Tytuł oryginału: *Digital Signal Processing: A Practical Guide for Engineers and Scientists by Steven W. Smith*
ISBN: 0-750674-44-X

Copyright © 2003, Steven W. Smith.

This edition of *Digital Signal Processing: A Practical Guide for Engineers and Scientists by Steven W. Smith* is published by arrangement with Elsevier Inc, 200 Wheleer Road, 6th Floor, Burlington, MA01803, USA.

Tłumaczenie: *dr inż. Michał Nadachowski*
Redaktor techniczny: *Delfina Korabiewska*

Polish edition Copyright 2007 by BTC Korporacja.
All rights reserved.

ISBN 978-83-60233-18-4 (83-60233-18-7)



Wydawnictwo BTC
ul. Inowłodzka 5
03-237 Warszawa
fax: (22) 814-13-02
http://www.btc.pl
e-mail: redakcja@btc.pl

Wydanie I.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawnictwo BTC dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawnictwo BTC nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentów niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Druk i oprawa: Łódzka Drukarnia Dziełowa S.A.

Przedmowa ... 9
Cele książki i sposoby ich realizacji ... 9
Podziękowania ... 10
1. Rozległość zastosowań i głębia problematyki DSP ... 11
Korzenie DSP ... 12
Telekomunikacja ... 14
Przetwarzanie sygnału fonicznego ... 15
Echolokacja ... 17
Przetwarzanie obrazów ... 18
2. Statystyka, rachunek prawdopodobieństwa i szumy ... 21
Terminologia sygnałów i wykresów ... 22
Wartość średnia i odchylenie standardowe ... 23
Sygnał a proces przezeń reprezentowany ... 26
Histogram, pmf, pdf ... 29
Rozkład normalny ... 35
Generacja szumu cyfrowego ... 38
Precyzja i dokładność ... 41
3. Przetwarzanie analogowo-cyfrowe i cyfrowo-analogowe ... 43
Kwantyzacja ... 44
Twierdzenie o próbkowaniu ... 47
Przetwarzanie cyfrowo-analogowe ... 53
Filtry analogowe w przetwarzaniu danych ... 56
Wybór filtru antyaliasingowego ... 63
Przetwarzanie o wielu częstotliwościach próbkowania (*multirate*) ... 66
Jednobitowe przetwarzanie danych ... 67
4. Oprogramowanie DSP ... 73
Liczby w komputerze ... 74
Zapis stałoprzecinkowy (liczby całkowite) ... 74
Zapis zmiennoprzecinkowy (liczby rzeczywiste) ... 76
Dokładność liczb ... 78
Szybkość wykonania – język programowania ... 81
Szybkość wykonywania – sprzęt ... 85
Szybkość wykonywania – triki programowe ... 89
5. Systemy liniowe ... 91
Sygnały i systemy ... 92
Warunki liniowości ... 93
Liniowość statyczna i wierność odtwarzania sinusoidy ... 96
Przykłady systemów liniowych i nieliniowych ... 99
Specjalne cechy liniowości ... 99
Zasada superpozycji – podstawa DSP ... 101

Metody rozkładu ... 103
Alternatywy liniowości ... 108
6. Splot ... 109
Funkcja delta i odpowiedź impulsowa ... 110
Splot ... 111
Algorytm od strony wejścia ... 113
Algorytm od strony wyjścia ... 117
Suma wejść ważonych ... 122
7. Właściwości splotu ... 123
Typowe odpowiedzi impulsowe ... 124
Właściwości matematyczne ... 132
Szybkość ... 139
8. Dyskretne przekształcenie Fouriera ... 141
Rodzina przekształceń Fouriera ... 142
Zapis i format rzeczywistego DFT ... 147
Zmienna niezależna w dziedzinie częstotliwości ... 148
Funkcje bazowe DFT ... 150
Synteza, obliczanie odwrotnej transformaty DFT ... 152
Analiza, obliczanie transformaty DFT ... 156
Dualizm ... 159
Zapis we współrzędnych biegunowych ... 160
Niedogodności współrzędnych biegunowych ... 162
9. Zastosowania DFT ... 167
Analiza widmowa sygnałów ... 168
Odpowiedź częstotliwościowa systemów ... 175
Splot przez dziedzinę częstotliwości ... 177
10. Właściwości przekształcenia Fouriera ... 183
Liniowość przekształcenia Fouriera ... 184
Właściwości fazowe ... 186
Okresowy charakter DFT ... 192
Kompresja i ekspansja, przetwarzanie o wielu częstotliwościach próbkowania ... 197
Mnożenie sygnałów (modulacja amplitudy) ... 200
Dyskretne w czasie przekształcenie Fouriera ... 202
Zależność Parsevala ... 204
11. Pary transformat Fouriera ... 205
Para transformat funkcji delta ... 206
Funkcja $sin(x)/x$ ... 208
Inne pary transformat ... 211
Efekt Gibbsa ... 213

Harmoniczne ... 216
Sygnały świergotowe ... 217
12. Szybkie przekształcenie Fouriera ... 219
Rzeczywiste DFT z zastosowaniem zespolonego DFT ... 220
Jak działa FFT ... 222
Programy FFT ... 227
Porównanie szybkości i dokładności ... 230
Dalszy wzrost szybkości ... 232
13. Przetwarzanie sygnałów ciągłych ... 237
Funkcja delta ... 238
Splot ... 240
Przekształcenie Fouriera ... 244
Szereg Fouriera ... 246
14. Wprowadzenie do filtrów cyfrowych ... 251
Podstawowe informacje o filtrach ... 252
Jak informacja jest reprezentowana w sygnałach ... 255
Parametry w dziedzinie czasu ... 255
Parametry w dziedzinie częstotliwości ... 256
Filtry górnoprzepustowe, pasmowoprzepustowe i pasmowozaporowe ... 260
Klasyfikacja filtrów ... 264
15. Filtry o zasadzie średniej kroczącej ... 265
Realizacja przez splot ... 266
Tłumienie szumu a odpowiedź skokowa ... 267
Odpowiedź częstotliwościowa ... 268
Filtry pokrewne filtrom o średniej kroczącej ... 269
Realizacja przez rekursję ... 271
16. Filtry okienkowane funkcją *sinc* ... 273
Sposób realizacji filtrów z okienkowaniem funkcją *sinc* ... 274
Projektowanie filtru ... 277
Przykłady filtrów okienkowanych funkcją *sinc* ... 280
Granice możliwości ... 283
17. Filtry dostosowane do szczególnych wymagań ... 285
Dowolna odpowiedź częstotliwościowa ... 286
Rozplot ... 290
Filtry optymalne ... 295
18. Splot FFT ... 299
Metoda nakładania z dodawaniem (*overlap-add*) ... 300
Splot FFT ... 302
Polepszenie szybkości ... 306

19. Filtry rekursywne 307
Metoda rekursywna 308
Jednobiegunowe filtry rekursywne 310
Filtry wąskopasmowe 314
Odpowiedź fazowa 316
Stosowanie liczb całkowitych 319
20. Filtry Czebyszewa 321
Odpowiedzi Czebyszewa i Butterwortha 322
Projektowanie filtru 323
Przerzut odpowiedzi skokowej 324
Stabilność 325
21. Porównanie filtrów 331
Pojedynek 1: Filtry analogowe przeciw cyfrowym 332
Pojedynek 2: Filtr okienkowany funkcją *sinc* przeciw filtrowi Czebyszewa 334
Pojedynek 3: Filtr o średniej kroczącej przeciw filtrowi jednobiegunowemu 337
22. Przetwarzanie sygnałów fonicznych 339
Słuch ludzki 340
Barwa dźwięku 343
Zależność jakości dźwięku od przepływności bitowej 346
Technika audio o wysokiej wierności 347
Kompansja 350
Synteza i rozpoznawanie mowy 351
Nieliniowe przetwarzanie sygnałów fonicznych 355
23. Tworzenie i wyświetlanie obrazów 359
Struktura obrazu cyfrowego 360
Aparaty fotograficzne i oczy 362
Telewizyjne sygnały wideo 369
Akwizycja i wyświetlanie innych obrazów 372
Regulacja jaskrawości i kontrastu 372
Przekształcenia skali szarości 376
Odkształcanie obrazów (*warping*) 378
24. Liniowe przetwarzanie obrazów 381
Splot 382
Modyfikacja krawędzi 3 × 3 386
Splot przez rozłączność 388
Przykład dużej funkcji PSF – spłaszczanie oświetlenia 390
Fourierowska analiza obrazu 394
Splot FFT 398
Bliższe spojrzenie na splot obrazowy 401

25. Specjalne techniki obrazowe 405
Rozdzielczość przestrzenna 406
Odstęp i apertura próbkowania 412
Stosunek sygnału do szumu 414
Morfologiczne przetwarzanie obrazu 418
Tomografia komputerowa 422
26. Sieci neuronowe (i więcej!) 431
Wykrywanie celu 432
Architektura sieci neuronowych 438
Dlaczego to działa? 442
Trenowanie sieci neuronowych 444
Ocena wyników 451
Projektowanie filtru rekursywnego 454
27. Kompresja danych 459
Sposoby kompresji danych 460
Kodowanie ciągów 461
Kodowanie Huffmana 463
Kodowanie delta 465
Kompresja LZW 466
JPEG (kompresja transformatowa) 472
MPEG 479
28. Cyfrowe procesory sygnałowe 481
Czym cyfrowe procesory sygnałowe różnią się od innych mikroprocesorów 482
Buforowanie cykliczne 484
Architektura cyfrowego procesora sygnałowego 487
DSP stałoprzecinkowe a DSP zmiennoprzecinkowe 491
Język C a język asemblera 496
Jak szybkie są cyfrowe procesory sygnałowe? 502
Rynek cyfrowych procesorów sygnałowych (DSP) 506
29. Rozpoczęcie pracy z DSP 509
Rodzina ADSP-2106x 510
Zestaw próbny SHARC EZ-KIT Lite 511
Przykład projektowania – filtr SOI do sygnałów fonicznych 513
Pomiary analogowe w systemie DSP 516
Inne spojrzenie na systemy stało- i zmiennoprzecinkowe 518
Zaawansowane narzędzia programowe 520
30. Liczby zespolone 527
System liczb zespolonych 528
Zapis we współrzędnych biegunowych 530
Stosowanie liczb zespolonych metodą zastępowania 533

Zespolona reprezentacja sinusoid 535
Zespolona reprezentacja systemów 536
Analiza obwodów elektrycznych 537
31. Zespolone przekształcenie Fouriera 541
Rzeczywiste DFT 542
Równoważność matematyczna 543
Zespolone DFT 544
Rodzina przekształceń Fouriera 548
Dlaczego stosuje się przekształcenie Fouriera 552
32. Przekształcenie Laplace'a 553
Właściwości dziedziny *s* 554
Metodyka przekształcenia Laplace'a 560
Analiza obwodów elektrycznych 564
Znaczenie zer i biegunów 567
Projektowanie filtrów w dziedzinie zmiennej zespolonej *s* 570
33. Przekształcenie Z 575
Właściwości dziedziny *z* 576
Analiza systemów rekursywnych 580
Stopnie kaskadowe i równoległe 584
Inwersja widmowa 587
Zmiany wzmocnienia 589
Projektowanie filtru Czebyszewa–Butterwortha 590
To, co najlepsze i najgorsze w DSP 596
Słownik pojęć 597
Indeks 613

Przedmowa

Cele książki i sposoby ich realizacji

Świat techniki ulega bardzo gwałtownym zmianom. W ciągu zaledwie 15 lat efektywność pracy komputerów osobistych wzrosła prawie *tysiąckrotnie* i wszystko wskazuje na to, że w ciągu następnych 15 lat wzrośnie znowu o czynnik równy *tysiąc*. Ta wielka efektywność komputerów zmienia sposób podejścia do rozwiązywania problemów badawczych i inżynierskich, a najlepszym tego przykładem jest cyfrowe przetwarzanie sygnałów.

We wczesnych latach 80. nauczano techniki DSP[1] na poziomie studiów magisterskich na wydziałach elektrycznych i elektroniki. Dziesięć lat później DSP stało się standardową częścią programu studiów inżynierskich. Teraz znajomość DSP jest *podstawową umiejętnością* wymaganą od naukowców i inżynierów w wielu dziedzinach. Niestety, postęp w nauczaniu DSP nie nadąża za rozwojem tej techniki. Prawie wszystkie książki o DSP są napisane w tradycyjnym stylu podręczników podstaw elektrotechniki, z dużą ilością trudnych, szczegółowych rozważań matematycznych. Technika DSP jest niewiarygodnie silnym narzędziem, lecz bez zrozumienia tej techniki nie można jej zastosować.

Tę książkę napisano dla naukowców i inżynierów pracujących w wielu różnych dziedzinach. Wymieńmy kilka z nich: fizyka, bioinżynieria, geologia, oceanografia, mechanika, elektrotechnika i elektronika. Celem książki jest takie przedstawienie praktycznych aspektów DSP, aby uniknąć barier, jakie tworzą szczegółowe wywody matematyczne i abstrakcyjne teorie. Dlatego trzymano się przy pisaniu tej książki trzech podstawowych zasad.

Po pierwsze, metody DSP są *wyjaśnione*, a nie tylko udowodnione na podstawie rozważań matematycznych. Wprawdzie podano wiele wzorów, ale nie one są głównym sposobem dostarczenia Czytelnikom informacji. Nie ma niczego lepszego niż dobrze napisany tekst wyjaśniający, poparty dobrymi rysunkami.

Po drugie, *liczby zespolone potraktowano jako temat trudniejszy*, a więc zagadnienie, które można opanować dopiero po zrozumieniu podstawowych zasad. W rozdziałach 1–29 podstawy techniki DSP są objaśnione przy użyciu algebry, w rzadkich tylko przypadkach z wykorzystaniem rachunku różniczkowego i całkowego. Z materiału zawartego w rozdziałach 30–33 wynika, jak bardzo zastosowanie liczb zespolonych zwiększa efektywność techniki DSP, umożliwiając tworzenie metod nierealizowalnych przy użyciu tylko liczb rzeczywistych. Wielu Czytelników potraktuje takie podejście do tematu jak herezję! Tradycyjne podręczniki DSP są pełne teorii liczb zespolonych, często już od pierwszego rozdziału.

Po trzecie, użyto *bardzo prostych programów komputerowych*. W praktyce większość programów związanych z DSP pisze się w C, Fortranie lub podobnych językach. Jednak *uczenie się* DSP stawia inne wymagania niż *stosowanie* DSP. Student powinien skoncentrować się na algorytmach i metodach, bez rozpraszania uwagi na różne chwyty poszczególnych języków. Efektywność i elastyczność nie są ważne; najważniejsza jest prostota. Programy w tej książce napisano po to, aby nauczyć DSP w sposób najprostszy, traktując inne problemy jako wtórne. Nie przywiązywano wagi do dobrego stylu programowania tam, gdzie można było w ten sposób uczynić program bardziej logicznym i klarownym. Na przykład:

- korzystano z uproszczonej wersji języka BASIC
- zamieszczano numery wierszy
- jako strukturę sterującą używano tylko pętli FOR-NEXT
- nie stosowano instrukcji wejścia-wyjścia.

[1] DSP (*Digital Signal Processing*) jest wprawdzie skrótem angielskim, lecz powszechnie stosowanym w światowej literaturze technicznej. Dlatego będziemy go używać w tej książce. (przyp. tłum.).

Jest to najprostszy styl programowania, jaki udało mi się znaleźć. Niektórzy mogą sądzić, że książka byłaby lepsza, gdyby programy napisano w języku C. Całkowicie się z tym nie zgadzam.

Spodziewany krąg czytelników

Ta książka jest zamierzona przede wszystkim jako materiał dla rocznych praktycznych szkoleń w technice DSP, których uczestnikami są osoby pracujące w różnych dziedzinach nauki i techniki. Proponowanym przygotowaniem wstępnym są:

- Kurs elektroniki praktycznej (wzmacniacze operacyjne, układy RC itd.)
- Kurs programowania komputerowego (Fortran lub podobne)
- Roczny wykład rachunku różniczkowego i całkowego

Pisząc tę książkę, miałem też na uwadze potrzeby osób profesjonalnie zajmujących się DSP. Opisano wiele często spotykanych zastosowań: filtry cyfrowe, sieci neuronowe, kompresję danych, przetwarzanie dźwięku i obrazu itd. Rozdziały książki o tej tematyce są w dużym stopniu niezależne, aby Czytelnik dla rozwiązania konkretnego problemu nie musiał przeszukiwać całej książki.

Podziękowania

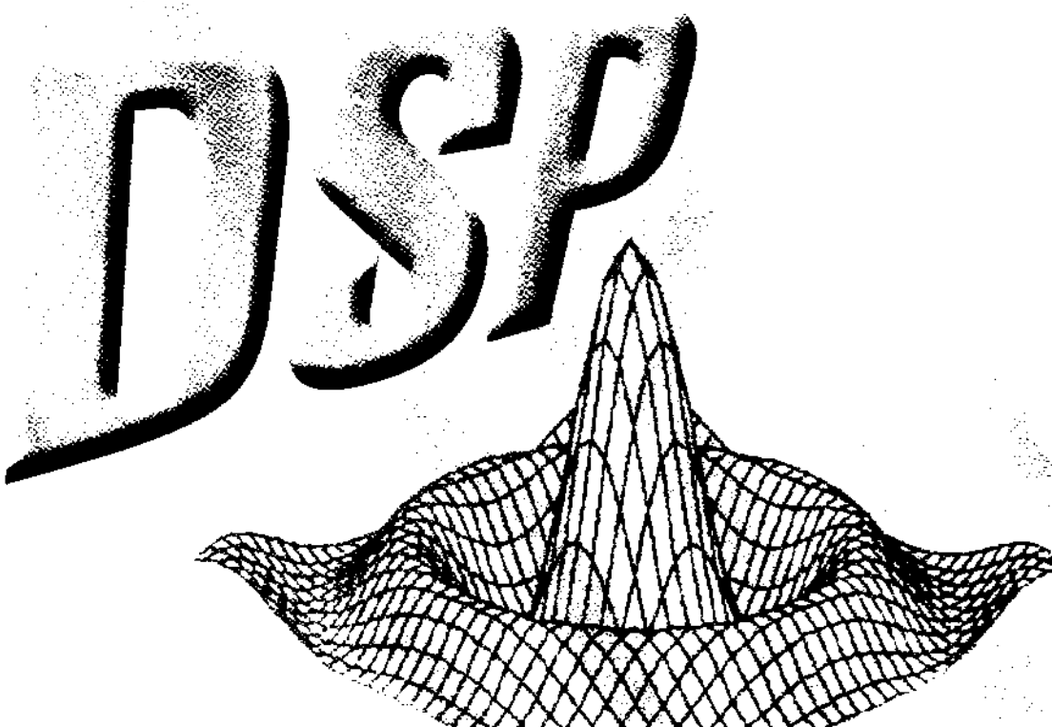
Wielu osobom składam szczególne podziękowania za uwagi i sugestie dotyczące tej książki. Ci pierwsi recenzenci, dzięki wspaniałomyślnemu poświęceniu swojego czasu i zdolności, przyczynili się do udoskonalenia książki. A oto oni: **Magnus Aronsson** (Wydział Elektryczny, Uniwersytet Utah); **Bruce B. Azimi** (Marynarka Wojenna USA), **Vernon L. Chi** (Wydział Informatyki, Uniwersytet Karoliny Północnej), **Manohar Das, Ph.D.** (Wydział Elektryki i Inżynierii Systemowej, Uniwersytet Oakland), **Carol A. Dean** (Analog Devices, Inc.), **Fred DePiero, Ph.D.** (Wydział Elektryczny, Uniwersytet Stanowy CalPoly), **Jose Fridman, Ph.D.** (Analog Devices, Inc.), **Frederick K. Duennebier, Ph.D.** (Wydział Geologii i Geofizyki, Uniwersytet Hawajów, Manoa), **D. Lee Fugal** (Space&Signals Technologies), **Filson H. Glanz, Ph.D.** (Wydział Elektryki i Techniki Komputerowej, Uniwersytet New Hampshire), **Kenneth H. Jacker** (Wydział Informatyki, Uniwersytet Stanowy Appałachian), **Rajiv Kapadia, Ph.D.** (Wydział Elektryczny, Uniwersytet Stanowy Mankato), **Dan King** (Analog Devices, Inc.), **Kevin Leary** (Analog Devices, Inc.), **A. Dale Magoun, Ph.D.** (Wydział Informatyki, Uniwersytet Północno-Wschodniej Luizjany), **Ben Mbugua** (Analog Devices, Inc.), **Bernard J. Maxum, Ph.D.** (Wydział Elektryczny, Uniwersytet Lamar), **Paul Morgan, Ph.D.** (Wydział Geologii, Uniwersytet Północnej Arizony), **Dale H. Mugler, Ph.D.** (Wydział Matematyki, Uniwersytet Akron), **Christopher L. Mullen, Ph.D.** (Wydział Inżynierii Lądowej, Uniwersytet Missisipi), **Cynthia L. Nelson, Ph.D.** (Sandia National Laboratories), **Branislava Perunicic-Drazenovic, Ph.D.** (Wydział Elektryczny, Uniwersytet Lamar), **John Schmeelk, Ph.D.**(Wydział Matematyczny, Uniwersytet Virginia Commonwealth), **Richard R. Schultz, Ph.D.** (Wydział Elektryczny, Uniwersytet Dakoty Północnej), **David Skolnick** (Analog Devices, Inc.), **Jay J. Smith, Ph.D.** (Centrum Techniki Kosmicznej, Uniwersytet Stanowy Weber), **Jeffrey Smith, Ph.D.** (Wydział Informatyki, Uniwersytet Georgia), **Oscar Yanez Suarez, Ph. D.** (Wydział Elektryczny, Metropolitan University, kampus Iztapalapa, Mexico City) oraz inni, którzy pragną pozostać anonimowi.

Teraz książka jest w rękach ostatecznych recenzentów, którymi jesteście Wy – jej Czytelnicy. Poświęćcie, proszę, trochę czasu na przekazanie mi swoich uwag i sugestii. Przyczynią się one do poprawienia dalszych wydań książki, aby mogła jeszcze lepiej służyć waszym potrzebom. Nie więcej niż dwie minuty zajmie wam wysłanie e-maila pod adresem: *Smith@DSPguide.com*. Dziękuję. Mam nadzieję, że książka się Wam spodoba.

Steve Smith

1

Rozległość zastosowań i głębia problematyki DSP



Cyfrowe przetwarzanie sygnałów (DSP) jest jedną z najważniejszych technik, które będą kształtować naukę i technikę w XXI wieku. Rewolucyjne przemiany zaszły już w *rozległym* obszarze zastosowań: w telekomunikacji, obrazowaniu w medycynie, radarach i sonarach, odtwarzaniu muzyki z bardzo dobrą wiernością, a także w wykrywaniu ropy naftowej, jeśli wymienić tylko kilka. W każdej z tych dziedzin nastąpił *głęboki* rozwój problematyki DSP, z własnymi algorytmami, matematyką i wyspecjalizowanymi metodami. To połączenie rozległości zastosowań z głębią problematyki powoduje, że nikt nie jest w stanie zapanować nad całą dotychczas opracowaną techniką DSP. Edukacja DSP obejmuje dwa zadania: poznanie koncepcji ogólnych dotyczących tej dziedziny jako całości i nauczenie się metod specjalistycznych związanych z konkretnym obszarem naszego zainteresowania. W tym rozdziale rozpoczynamy naszą podróż w świat cyfrowego przetwarzania sygnałów od omówienia silnego wpływu, jaki wywarło DSP na kilka bardzo różnorodnych dziedzin. Rewolucja się zaczęła.

Korzenie DSP

Cyfrowe przetwarzanie sygnałów wyróżnia się spośród innych dziedzin techniki komputerowej unikatowym rodzajem używanych danych: *sygnałami*. W większości przypadków te sygnały, jako dane z czujników, pochodzą ze świata rzeczywistego: z wibracji sejsmicznych, obrazów wizyjnych, fal dźwiękowych itd. DSP to matematyka, algorytmy i rozwiązania techniczne stosowane do operowania tymi sygnałami po ich przetworzeniu w postać cyfrową. DSP służy do realizacji wielu różnych celów, takich jak udoskonalanie obrazów, rozpoznawanie i generacja mowy, kompresja danych w celu ich magazynowania i przesyłania itd. Załóżmy, że do komputera dołączyliśmy przetwornik analogowo-cyfrowy i zebraliśmy pewną porcję danych ze świata rzeczywistego. Wtedy system DSP zapyta: *I co dalej?*

Korzenie DSP sięgają lat 60. i 70. ubiegłego wieku, gdy pojawiły się komputery cyfrowe. Komputery były wówczas bardzo drogie, a systemy DSP ograniczały się tylko do niewielu szczególnie ważnych zastosowań. Pionierskie prace zrealizowano w czterech dziedzinach o dużym znaczeniu. Były to: *radary* i *sonary*, od których zależało bezpieczeństwo narodowe; *poszukiwania ropy*, na których można było zarobić wielkie pieniądze, *badania kosmiczne*, w których dane są niepowtarzalne i *obrazowanie medyczne*, mogące ratować życie. Rewolucja w dziedzinie komputerów osobistych w latach 80. i 90. spowodowała eksplozję zastosowań DSP w nowych dziedzinach. Motywacją rozwoju DSP były już nie tylko potrzeby wojskowe i państwowe, ale nagle stały się nią także wymagania rynku. Każdy, kto sądził, że może zarobić na tej szybko rozwijającej się dziedzinie, od razu stawał się dostawcą systemów DSP. Technika DSP dotarła do szerokich kręgów społeczeństwa w takich urządzeniach, jak telefony komórkowe, odtwarzacze płyt kompaktowych i elektroniczna poczta głosowa. Niektóre z tych różnorodnych zastosowań zestawiono na rys. 1.1.

Rewolucja techniczna postępowała z góry do dołu. We wczesnych latach 80. DSP nauczano na poziomie *studiów magisterskich* na wydziałach elektrycznych i elektroniki. Dziesięć lat później DSP stało się standardową częścią programu *studiów inżynierskich*. Teraz znajomość techniki DSP jest *podstawową umiejętnością* wymaganą od inżynierów i naukowców w wielu dziedzinach. Szybki postęp DSP można porównać z poprzednią rewolucją techniczną – w *elektronice*. W dziedzinie elektroniki niemal wszyscy specjaliści mają podstawową wiedzę z zakresu projektowania układów. Bez tych wiadomości byliby zagubieni w świecie techniki. Taka sama przyszłość czeka DSP.

Dzieje rozwoju DSP to nie tylko ciekawostka, mają one znaczny wpływ na *wasze* możliwości nauczenia się i stosowania DSP. Załóżmy, że napotykacie problemy z zakresu DSP i szukacie potrzebnych rozwiązań w książkach i innych publikacjach. Na ogół znajdziecie tam całe stronice pełne wzorów, nieznanych symboli matematycznych i niejasnej terminologii. Prawdziwy koszmar! Większość literatury dotyczącej DSP wprawia w zakłopotanie nawet czytelników mających już pewne doświadczenie w tej dziedzinie. Nie są to, oczywiście, publikacje złe, lecz po prostu



Rys. 1.1. Technika DSP zrewolucjonizowała wiele dziedzin nauki i techniki. Na rysunku przedstawiono niektóre z tych rozmaitych zastosowań

przeznaczone dla wąskiego kręgu czytelników o ścisłej specjalizacji. Tego rodzaju szczegółowe rozważania matematyczne są potrzebne prowadzącym badania w zakresie DSP do zrozumienia teoretycznych implikacji tego zagadnienia.

Przy pisaniu tej książki przyjęto podstawowe założenie, że większości zagadnień praktycznych DSP można się nauczyć i stosować je bez tradycyjnej bariery tworzonej przez szczegółową matematykę i teorię. *Cyfrowe przetwarzanie sygnałów. Praktyczny poradnik dla inżynierów i naukowców* napisano dla tych, którzy traktują DSP jako *narzędzie*, a nie nową drogę *kariery* zawodowej.

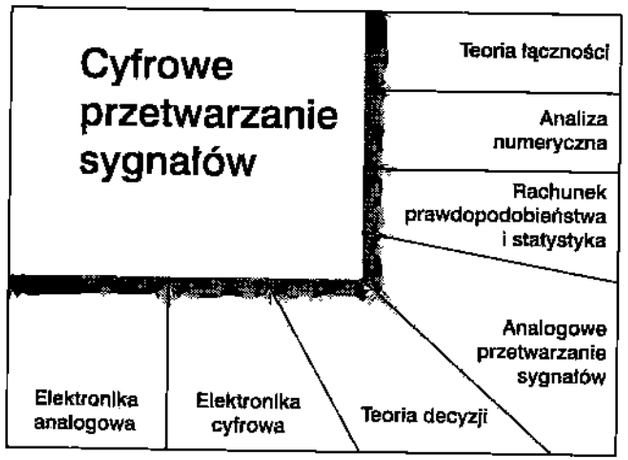
W dalszej części tego rozdziału będą przedstawione dziedziny, w których DSP spowodowało rewolucyjne zmiany. Poznając kolejne zastosowania, zauważcie, że DSP ma charakter wybitnie *interdyscyplinarny*, wykorzystując rozwiązania techniczne z wielu pokrewnych ze sobą dziedzin. Jak wynika z rys. 1.2, granice między DSP i innymi dziedzinami nie są ostre i wyraźnie określone, lecz raczej rozmyte i zachodzące na siebie. Jeśli chcecie specjalizować się w DSP, to pamiętajcie o tych pokrewnych dziedzinach, które też będziecie musieli poznać.

Telekomunikacja

Telekomunikacja to przesyłanie informacji na odległość. Przesyłana informacja może mieć wiele różnych form: rozmowy telefoniczne, sygnały telewizyjne, pliki komputerowe i inne typy danych. Do przesłania informacji jest konieczny *kanał* między dwiema lokalizacjami. Kanałem może być para przewodów, łącze radiowe, światłowód itd. Firmy telekomunikacyjne pobierają od swoich klientów *opłaty* za przesyłanie informacji, a w zamian muszą ponosić *koszty* utrzymania i konserwacji kanału. Rachunek ekonomiczny jest prosty: im więcej informacji firmy telekomunikacyjne są w stanie przesłać jednym kanałem, tym więcej pieniędzy mogą zarobić. Technika DSP zrewolucjonizowała telekomunikację w wielu obszarach: w generacji i wykrywaniu tonu sygnalizacji, przesuwania pasm częstotliwości (*frequency band shifting*), filtracji w celu usunięcia tętnień sieciowych itd. Omówimy teraz trzy charakterystyczne przykłady z dziedziny sieci telefonicznych – multipleksowanie, kompresję, a także tłumienie echa.

Multipleksowanie

Na świecie jest ok. *1 miliarda* telefonów. Systemy telefoniczne umożliwiają, za naciśnięciem kilku przycisków, połączenie każdego z tych telefonów z dowolnym innym. Ogrom tego zadania jest zdumiewający! Do lat 60. do uzyskania połączenia między dwoma telefonami było konieczne przesłanie analogowego sygnału głosowego przez przełączniki mechaniczne i wzmacniacze. Jedno połączenie wymagało jednej pary przewodów. Stosując zaś DSP, przetwarza się sygnały głosowe w strumień szeregowych danych cyfrowych. Ponieważ bity można łatwo przeplatać ze sobą i potem rozdzielać, więc jednym kanałem można przesłać wiele rozmów telefonicznych. Na przykład standard telefoniczny zwany *T-carrier* (USA) umożliwia jednoczesne przesyłanie 24 sygnałów głosowych. Każdy z nich jest próbkowany 8000 razy na sekundę przy użyciu kompanderowego, 8-bitowego przetwarzania analogowo-cyfrowego (z kompresją logarytmiczną). W rezultacie każdy sygnał głosowy jest reprezentowany przez strumień o przepływności bitowej 64 000 bit/s i wszystkie 24 kanały można zawrzeć w strumieniu o przepływności 1,544 Mbit/s. Taki sygnał można przesłać na odległość ok. 1800 m zwykłą linią telefoniczną z drutu miedzianego o średnicy ok. 0,64 mm. Jest to typowa odległość połączenia. Znaczne są korzyści ekonomiczne wynikające z przesyłania cyfrowego. Przełączniki analogowe są drogie, cyfrowe bramki logiczne zaś – tanie.



Rys. 1.2. Granice między cyfrowym przetwarzaniem sygnałów i innymi dziedzinami nauki, techniki i matematyki są rozmyte i zachodzą na siebie

Kompresja

Przy próbkowaniu sygnału głosowego z częstotliwością 8000 próbek/sekundę większość informacji cyfrowych to informacje *nadmiarowe* (redundancyjne). Informacja zawarta w jednej próbce jest bowiem w znacznym stopniu powielona w próbkach sąsiednich. Opracowano wiele algorytmów DSP do przetwarzania cyfrowych danych głosowych na strumień danych o mniejszej wymaganej przepływności bitowej (bit/s). Są to algorytmy **kompresji danych**. Równoważne im algorytmy **dekompresji** stosuje się do odtwarzania sygnału w jego oryginalnej postaci. Poszczególne algorytmy różnią się od siebie stopniem kompresji i uzyskiwaną jakością dźwięku. Ogólnie biorąc, zredukowanie przepływności bitowej danych z 64 kbit/s do 32 kbit/s nie powoduje pogorszenia jakości dźwięku. Przy kompresji do przepływności bitowej 8 kbit/s wpływ na jakość dźwięku jest już wprawdzie zauważalny, lecz taką kompresję można jeszcze stosować w sieciach telefonicznych dalekiego zasięgu. Najwyższa osiągalna kompresja do ok. 2 kbit/s powoduje znaczne zniekształcenia dźwięku, lecz jest przydatna w niektórych zastosowaniach – np. wojskowych i w łączności podmorskiej.

Tłumienie echa

Występowanie echa jest poważnym problemem w łączności telefonicznej dalekiego zasięgu. Gdy mówimy do mikrofonu aparatu telefonicznego, sygnał reprezentujący nasz głos wędruje do aparatu odbiorczego, z którego część sygnału powraca jako echo. Jeśli odległość wynosi kilkaset kilometrów, to czas powrotu echa jest równy kilka milisekund. Ucho jest przyzwyczajone do odbioru echa o tak małym opóźnieniu i połączenie jest traktowane jako prawidłowe. Przy większych odległościach echo staje się coraz bardziej słyszalne i irytujące. Opóźnienie może osiągać kilkaset milisekund w łączności międzykontynentalnej i jest wtedy szczególnie nieprzyjemne. Cyfrowe przetwarzanie sygnałów daje rozwiązanie tych trudności przez pomiar sygnału powrotnego i generowanie odpowiedniego *antysygnału* likwidującego kłopotliwe echo. Ta sama technika umożliwia użytkownikom aparatów telefonicznych z głośnikiem jednoczesne słuchanie i mówienie bez kłopotów ze sprzężeniem akustycznym (gwizdami interferencyjnymi). Może też być stosowana do tłumienia szumu otoczenia przez redukowanie go *antyszumem* generowanym cyfrowo.

Przetwarzanie sygnału fonicznego

Wzrok i słuch to dwa najważniejsze nasze zmysły. Właśnie dlatego znaczna część zastosowań DSP jest związana z przetwarzaniem obrazu i dźwięku. Słuchamy zarówno *muzyki*, jak i *mowy*. Technika DSP spowodowała rewolucyjne zmiany w obu tych dziedzinach.

Muzyka

Droga prowadząca od mikrofonu muzyka do głośnika audiofila jest długa. Dzięki cyfrowej reprezentacji przesyłanych danych zapobiega się pogorszeniu jakości związanemu zwykle z analogowym przetwarzaniem i zapamiętywaniem. Jest to oczywiste dla każdego, kto porównywał jakość muzyki uzyskiwanej z magnetofonu z muzyką z płyty kompaktowej. Utwór muzyczny jest w studiu dźwiękowym zazwyczaj nagrywany w wielu kanałach lub na wielu ścieżkach. W niektórych przypadkach nagrywa się osobno nawet poszczególne instrumenty i wokalistów. Inżynier dźwięku ma wtedy większą swobodę w tworzeniu nagrania finalnego. Złożony proces łączenia z sobą indywidualnych ścieżek nagraniowych nazywa się *miksowaniem*. Do realizacji kilku ważnych funkcji podczas miksowania można zastosować DSP. Są to: filtracja, sumowanie i odejmowanie sygnałów, edycja sygnałów i inne.

Jednym z najciekawszych zastosowań DSP w dziedzinie przygotowania muzyki jest *sztuczny pogłos*. Jeśli sygnały z poszczególnych kanałów są po prostu sumowane, to brzmienie dźwięku wydaje się rozrzedzone i kruche, jakby muzycy grali gdzieś w otwartej przestrzeni. Słuchacze odnoszą takie wrażenie, gdyż są w dużym stopniu przyzwyczajeni do echa i pogłosu zawartego w słuchanej muzyce, a te właśnie efekty są na ogół minimalizowane

w studiu dźwiękowym. Technika DSP daje podczas miksowania możliwość wprowadzania sztucznego echa i pogłosu w celu symulacji różnych pomieszczeń odsłuchowych. Echo z opóźnieniem kilkuset milisekund stwarza wrażenie słuchania w pomieszczeniu, takim jak katedra. Dodanie echa z opóźnieniem 10–20 ms daje wrażenie odsłuchu w pomieszczeniu o skromniejszych wymiarach.

Generacja mowy

Generacja i rozpoznawanie mowy znajdują zastosowanie w komunikacji człowiek–maszyna. Zamiast rąk i oczu można używać ust i uszu. Jest to bardzo wygodne w sytuacjach, gdy ręce i oczy są zajęte innymi czynnościami, takimi jak prowadzenie samochodu, wykonywanie operacji chirurgicznej lub (niestety) celowanie z broni do wroga. Są dwa sposoby komputerowej generacji mowy: *rejestracja cyfrowa* i *symulacja organu głosowego*. Przy rejestracji cyfrowej sygnał mowy jest przetwarzany do postaci cyfrowej i rejestrowany, zwykle w formie skomprymowanej. Podczas odtwarzania zapisane dane podlegają dekompresji i są z powrotem przetwarzane na sygnał analogowy. Jednogodzinny zapis mowy wymaga tylko ok. 3 megabajtów pamięci, a więc w granicach możliwości nawet małych systemów komputerowych. Jest to obecnie najczęściej stosowana metoda generacji mowy.

Symulatory organu głosowego są bardziej skomplikowane, gdyż są próbą naśladowania fizycznych mechanizmów tworzenia mowy. U człowieka organ głosowy jest wnęką akustyczną o częstotliwościach rezonansowych określonych rozmiarami i kształtem komór. Są dwa podstawowe źródła powstawania dźwięku w organie głosowym, dające głoski *dźwięczne* i *bezdźwięczne* (*np. trące*). Głoski dźwięczne powstają, gdy powietrze jest wypychane z płuc przez struny głosowe i dalej przez usta i/lub przez nos. Głoski bezdźwięczne powstają zaś w wyniku turbulencji powietrza w zwężeniach, takich jak zęby i wargi. Symulatory organu głosowego działają na zasadzie generacji sygnałów cyfrowych przypominających dwa wymienione rodzaje sygnałów pobudzających. Właściwości komory rezonansowej są symulowane przez przepuszczanie sygnału pobudzającego przez filtr o podobnych właściwościach rezonansowych. Taką metodę przyjęto w jednym z pierwszych uwieńczonych powodzeniem zastosowań DSP, jakim był sprzedawany w dużych ilościach system *Speak & Spell*, urządzenie dla dzieci, wspomagające nauczanie.

Rozpoznawanie mowy

Automatyczne rozpoznawanie mowy jest nieporównanie trudniejsze niż jej generacja. Rozpoznawanie mowy to klasyczny przykład zadania, które mózg ludzki realizuje dobrze, a komputer – słabo. Komputery cyfrowe są w stanie zapamiętać i przywołać z pamięci ogromne ilości danych, wykonywać obliczenia matematyczne z zadziwiającą szybkością i realizować wielokrotnie powtarzające się czynności bez znudzenia i pogorszenia wydajności. Niestety komputery współczesne bardzo słabo sobie radzą z danymi pochodzącymi bezpośrednio z czujników. Łatwo można nauczyć komputer comiesięcznego wysyłania rachunków za energię elektryczną, ale nauczenie tego samego komputera rozumienia naszego głosu jest trudnym przedsięwzięciem. Przy użyciu DSP problem rozpoznawania głosu jest rozwiązywany w dwóch etapach: *wydzielenie cech* i następnie *dopasowanie cech*. Każde słowo w sygnale akustycznym zostaje wyodrębnione i następnie przeanalizowane w celu zidentyfikowania rodzaju pobudzenia i częstotliwości rezonansowych. Te parametry są porównywane z wcześniejszymi przykładami wymawianych słów, aby wybrać najlepiej dopasowany odpowiednik. Działanie tych systemów ogranicza się często zaledwie do kilkuset słów i mogą one akceptować tylko mowę z wyraźnymi przerwami między słowami. Systemy muszą być przetestowane dla każdej osoby mówiącej. Takie rozpoznawanie mowy jest wystarczające w wielu zastosowaniach komercyjnych, lecz jego ograniczenia są znaczne w stosunku do możliwości ludzkiego słyszenia. W tej dziedzinie jest jeszcze bardzo dużo do zrobienia. Nagrodą dla tych, którzy z powodzeniem opracują i wprowadzą na rynek dobre urządzenia do rozpoznawania głosu, będą znaczne korzyści finansowe.

Echolokacja

Powszechnie stosowanym sposobem uzyskiwania informacji o odległym obiekcie jest *odbijanie* od niego jakichś fal. Na przykład radar działa na zasadzie wysyłania impulsów fal radiowych i analizy sygnału powracającego jako echo odbite od samolotu. W sonarze fale dźwiękowe są przesyłane w środowisku wodnym w celu wykrycia łodzi podwodnych i innych obiektów zanurzonych. Geofizycy od dawna badają kulę ziemską przez wywoływanie wybuchów i odsłuchiwanie sygnałów echa z głęboko położonych warstw skalnych. Chociaż idea działania jest jednakowa we wszystkich tych zastosowaniach, to każde z nich ma swoją specyfikę i wymagania. Cyfrowe przetwarzanie sygnałów wywołało rewolucyjne zmiany we wszystkich tych dziedzinach.

Radar

Nazwa radar jest akronimem *RAdio Detection And Ranging* (detekcja radiowa i wyznaczanie odległości). W najprostszym urządzeniu radarowym nadajnik radiowy wytwarza impuls fal radiowych o czasie trwania kilku mikrosekund. Impuls jest wprowadzany do anteny kierunkowej, z której fale są wysyłane z prędkością światła. Samolot, który znajdzie się na drodze tych fal, odbije niewielką część ich energii z powrotem w kierunku anteny odbiorczej umieszczonej blisko nadajnika. Odległość do obiektu jest obliczana na podstawie czasu, jaki upłynął od wysłania impulsu do odebrania echa. Prostsze jest wyznaczanie kierunku lokalizacji obiektu, gdyż znane jest *ustawienie* anteny kierunkowej w chwili odbioru echa.

Zakres działania radaru jest określony dwoma parametrami: energią zawartą w nadawanym impulsie i poziomem szumu w odborniku. Niestety zwiększanie energii impulsu wymaga na ogół *zwiększenia jego czasu trwania*. Szerszy impuls powoduje z kolei zmniejszenie dokładności pomiaru czasu opóźnienia. Występuje tu więc sprzeczność wymagań dotyczących dwóch ważnych parametrów: zdolności wykrywania obiektów na duże odległości i możliwości dokładnego określenia odległości od obiektu.

DSP zrewolucjonizowało radar w trzech aspektach, z których wszystkie są związane ze wspomnianą sprzecznością wymagań. Po pierwsze, przy użyciu DSP można impuls poddać *kompresji* po jego odbiorze, uzyskując lepsze wyznaczenie odległości bez zmniejszania zasięgu działania. Po drugie, system DSP może odfiltrować odebrany sygnał, tłumiąc szum. Dzięki temu zwiększa się zasięg bez pogorszenia dokładności pomiaru odległości. Po trzecie, DSP umożliwia szybką selekcję i generację impulsów o różnych kształtach i szerokościach. To daje możliwość m.in. optymalizacji impulsów dla różnych problemów detekcji. I teraz najważniejsze: większość tych zadań jest realizowana z częstotliwością próbkowania porównywalną ze stosowaną częstotliwością radiową, a więc kilkaset megaherców. Jeśli chodzi o radar, DSP jest równie ważne w postaci rozwiązań sprzętowych o dużej szybkości działania, jak i w formie algorytmów.

Sonar

Nazwa sonar jest akronimem: *SOund NAvigation and Ranging* (dźwiękowa nawigacja i wyznaczanie odległości). Istnieją dwa rodzaje sonaru: *aktywny* i *pasywny*. W sonarze aktywnym impulsy dźwiękowe o częstotliwości w zakresie od 2 do 40 kHz są wysyłane do środowiska wodnego a powracające sygnały echa są wykrywane i analizowane. Zastosowania sonaru aktywnego obejmują: wykrywanie i lokalizację obiektów podwodnych, nawigację, łączność oraz sporządzanie map dna morskiego. Typowy maksymalny zasięg roboczy wynosi od 10 do 100 km. Działanie sonaru pasywnego polega na *nasłuchu* dźwięków podwodnych, których źródłem mogą być naturalne turbulencje, życie podwodne, odgłosy mechaniczne z łodzi podwodnych i statków płynących na powierzchni morza. Sonar pasywny nie wysyła żadnej energii, dlatego jest doskonały do operacji ukrytych. To tak, jakbyśmy chcieli wykryć *jakiegoś faceta*, a on nie mógł wykryć *nas*. Najważniejszą

dziedziną zastosowania sonaru pasywnego są wojskowe systemy inwigilacyjne do wykrywania i śledzenia łodzi podwodnych. W sonarach pasywnych pracuje się na ogół na mniejszych częstotliwościach niż w aktywnych, gdyż takie sygnały są słabiej pochłaniane w wodzie. Zasięg wykrywania może wynosić nawet tysiące kilometrów.

DSP zrewolucjonizowało sonar w tych samych aspektach, co radar. Są to: generacja i kompresja impulsów oraz filtracja wykrywanych sygnałów. Pod pewnym względem sonar jest *łatwiejszy* od radaru – pracuje z mniejszymi częstotliwościami. Pod innym względem jest jednak trudniejszy, gdyż środowisko, w którym działa, jest mniej jednorodne i mniej stabilne. W systemach sonarowych stosuje się zazwyczaj rozległe sieci elementów nadawczych i odbiorczych, a nie tylko jeden kanał. Przez odpowiednie sterowanie sygnałami i ich mieszanie w tych wielu elementach, urządzenie sonarowe może wysyłać impulsy do żądanych miejsc i określać kierunki, z których są odbierane sygnały echa. Do obsługi tych wielu kanałów sonary wymagają DSP o takiej efektywności pracy, jak w radarach.

Sejsmologia odbiciowa

Już w latach 20. XX wieku geofizycy wykryli, że strukturę skorupy ziemskiej można badać falami dźwiękowymi. Poszukiwacze wywoływali wybuchy i rejestrowali sygnały echa pochodzące od warstw granicznych znajdujących się głębiej niż 10 km pod powierzchnią ziemi. Takie sejsmogramy odczytywano bezpośrednio wzrokowo i na ich podstawie odwzorowywano strukturę warstw podziemnych. Odbiciowe metody sejsmiczne stały się nagle głównym sposobem wyszukiwania złóż ropy i innych bogactw naturalnych, i takim pozostają do dziś.

W przypadku idealnym impuls dźwiękowy wysyłany w głąb Ziemi wytwarza pojedynczy sygnał echa od każdej warstwy granicznej, przez którą impuls przechodzi. Niestety, nie zawsze jest to takie proste. Każdy powracający sygnał echa musi znowu przejść przez wszystkie warstwy graniczne leżące powyżej miejsca jego powstania. Może to wywołać odbijanie się echa między warstwami i wykrywanie *ech echa* na powierzchni. Wykryty sygnał może, w wyniku ech wtórnych, być bardzo skomplikowany i trudny do interpretacji. Od lat 60. szeroko stosuje się DSP do wyizolowania echa pierwotnego z ech wtórnych w sejsmogramie odbiciowym. Jak geofizycy radzili sobie niegdyś bez DSP? Odpowiedź jest prosta – szukali *łatwych* miejsc, gdzie odbicia wielokrotne były minimalne. DSP umożliwia znalezienie ropy *w trudnych* miejscach, np. pod oceanem.

Przetwarzanie obrazów

Obrazy są sygnałami o szczególnych właściwościach. Po pierwsze, są one wyrażane parametrami *przestrzennymi* (odległościami), podczas gdy większość innych sygnałów jest wyrażana parametrami *czasowymi*. Po drugie, obrazy zawierają bardzo dużo informacji. Na przykład trzeba 10 MB pamięci do zarejestrowania 1 sekundy telewizyjnego sygnału wideo. Jest to ponad tysiąc razy więcej niż dla sygnału głosowego o podobnym czasie trwania. Po trzecie, ostateczną oceną jakości obrazu jest często subiektywna ocena widza, a nie kryterium obiektywne. Z powodu tych specyficznych właściwości przetwarzanie obrazów stanowi oddzielną grupę zastosowań w ramach DSP.

Medycyna

W roku 1895 Wilhelm Konrad Roentgen odkrył, że promieniowanie X może przenikać przez znaczne obszary materii. Możliwość zaglądania do wnętrza organizmu żywego człowieka zrewolucjonizowała medycynę. W ciągu zaledwie kilku lat medyczne urządzenia rentgenowskie rozprzestrzeniły się na całym świecie. Mimo oczywistego sukcesu obrazowanie przy użyciu promieni X miało cztery ograniczenia, zanim w latach 70. niewprowadzono DSP i podobnych technik. Po pierwsze, nakładające się na siebie tkanki

i narządy mogą pozostać ukryte. Na przykład fragmenty serca mogą być niewidoczne za żebrami. Po drugie, nie zawsze da się odróżnić tkanki podobne do siebie. Na przykład można odróżnić kości od tkanek miękkich, ale trudno – nowotwór od wątroby. Po trzecie, obrazy rentgenowskie pokazują *anatomię*, budowę organizmu, a nie *fizjologię*, czyli jego funkcjonowanie. Obraz rentgenowski człowieka żywego jest dokładnie taki sam jak martwego! Po czwarte, naświetlanie promieniowaniem X może spowodować raka, dlatego powinno być stosowane oszczędnie i tylko w uzasadnionych przypadkach.

Problem nakładających się tkanek i narządów rozwiązano w roku 1971, wprowadzając pierwszy **tomograf komputerowy** (najpierw nazywany CAT – komputerowy tomograf, lub skaner, osiowy – *Computed Axial Tomography*). Tomografia komputerowa (CT) jest klasycznym przykładem cyfrowego przetwarzania sygnałów. Promieniowanie X z wielu kierunków przechodzi przez badaną część ciała pacjenta. Zamiast prostego tworzenia obrazów na podstawie promieni X, odebrane sygnały są przetwarzane do postaci danych cyfrowych i gromadzone w komputerze. Następnie wykorzystuje się tę informację do *obliczeń* dających obrazy będące *przekrojami przez badany fragment ciała*. Na takich obrazach jest uwidocznionych dużo więcej szczegółów niż w konwencjonalnym badaniu rentgenowskim, dzięki czemu diagnoza i leczenie są znacznie skuteczniejsze. Wpływ tomografii komputerowej na medycynę był prawie tak duży, jak pierwsze wprowadzenie badań rentgenowskich. W ciągu kilku lat każdy większy szpital na świecie uzyskał dostęp do tomografu komputerowego. W roku 1979 dwaj główni twórcy tomografii komputerowej, Godfrey N. Hounsfield i Allan M. Cormack, otrzymali Nagrodę Nobla w dziedzinie medycyny. *To jest dobre DSP!*

Trzy dalsze wymienione ograniczenia badań rentgenowskich rozwiązano, stosując do penetracji organizmu rodzaje energii inne niż promieniowanie X, takie jak fale dźwiękowe i radiowe. We wszystkich tych metodach kluczową rolę odgrywa DSP. Na przykład w obrazowaniu przy użyciu rezonansu magnetycznego (MRI – *Magnetic Resonance Imaging*) do badania wnętrza ciała człowieka stosuje się pole magnetyczne w połączeniu z falami radiowymi. Przy właściwym doborze natężenia i częstotliwości tych pól jądra atomów w jakimś obszarze ciała wchodzą w rezonans, zmieniając energię między pewnymi skwantowanymi stanami. Rezonans daje emisję wtórnych fal radiowych wykrywanych przez antenę umieszczoną w pobliżu ciała pacjenta. Wartość i inne parametry odebranego sygnału dostarczają informacji o obszarze, w którym zachodzi rezonans. Dobór pola magnetycznego umożliwia przesuwanie badanego obszaru rezonansowego wzdłuż ciała, co daje odwzorowanie struktury wewnętrznej. Ta informacja jest zazwyczaj przedstawiana w formie obrazów, podobnie jak w tomografii komputerowej. Badanie rezonansem magnetycznym nie tylko daje doskonałe rozróżnianie między rozmaitymi rodzajami tkanek miękkich, ale dostarcza też informacji o zjawiskach fizjologicznych, np. o przepływie krwi w tętnicach. Metoda rezonansu magnetycznego jest nieodłącznie związana z cyfrowym przetwarzaniem sygnałów i bez DSP nie może być realizowana.

Badania kosmiczne

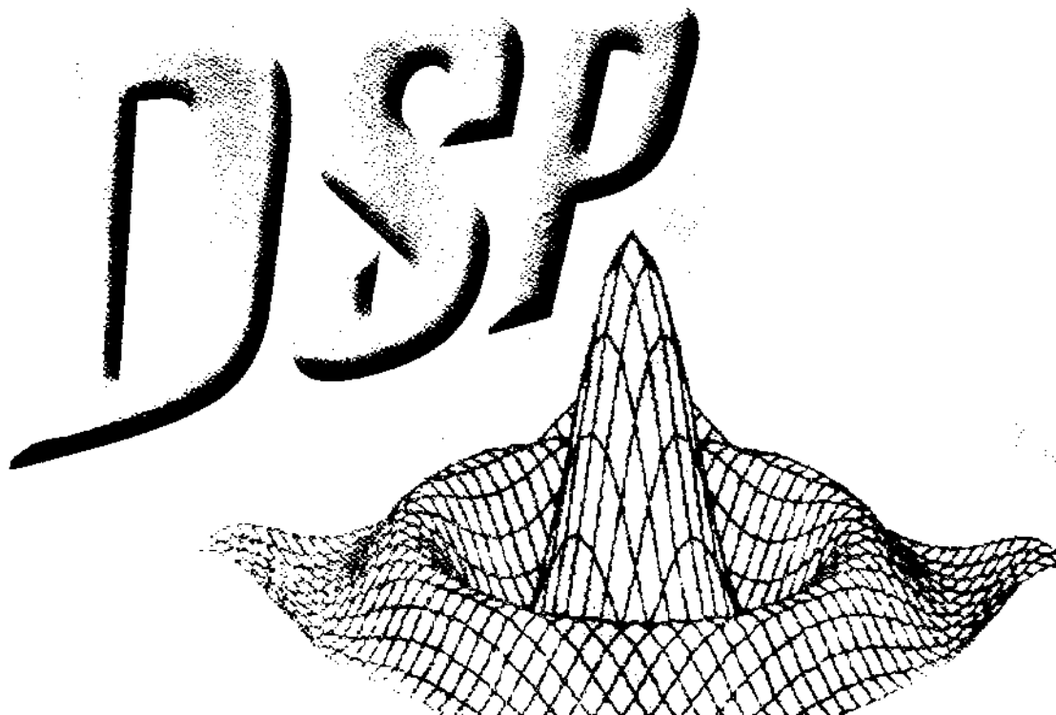
Czasem powstaje konieczność uzyskania jak największej ilości informacji z bardzo złego obrazu. Taka potrzeba często zachodzi w przypadku obrazów pochodzących z bezzałogowych satelitów i pojazdów kosmicznych. Nikt przecież nie wyśle na Marsa pracownika serwisu, aby tam pokręcił dwiema gałkami w kamerze! Przy użyciu DSP można kilkoma sposobami poprawić jakość obrazów uzyskiwanych w skrajnie niekorzystnych warunkach. Można dobierać jaskrawość i kontrast, wykrywać kontury, tłumić szum, dobierać ostrość, zmniejszać rozmycie wywołane ruchem itd. Można również *doprowadzać* do właściwej postaci obrazy obarczone zniekształceniami przestrzennymi, np. płaskie obrazy kulistej planety. W jedną bazę danych można łączyć wiele pojedynczych obrazów, co daje możliwość wyświetlania informacji w unikatowy sposób, np. w formie sekwencji wideo symulującej lot nad powierzchnią odległej planety.

Komercyjne urządzenia obrazowe

Duża ilość informacji zawarta w obrazach jest problemem w urządzeniach masowo sprzedawanych klientom. Urządzenia *komercyjne* (oferowane na rynku) powinny być *tanie*, co raczej nie idzie w parze z wymogami dużych pamięci i dużych szybkości przesyłania danych. Jednym z rozwiązań tego dylematu jest *kompresja obrazu*. Obraz, podobnie jak sygnał głosowy, zawiera dużą ilość informacji nadmiarowych. Dlatego dane obrazowe można przetwarzać przy użyciu algorytmów zmniejszających liczbę bitów informacji niezbędnych do reprezentowania obrazu. Do kompresji nadają się zwłaszcza obrazy telewizyjne i inne obrazy ruchome, gdyż większość informacji w kolejnych ramkach pozostaje taka sama. Dostępnymi na rynku urządzeniami, w których wykorzystano tę metodę, są na przykład: telefony wideo, programy komputerowe do wyświetlania obrazów ruchomych oraz telewizja cyfrowa.

2

Statystyka, rachunek prawdopodobieństwa i szumy



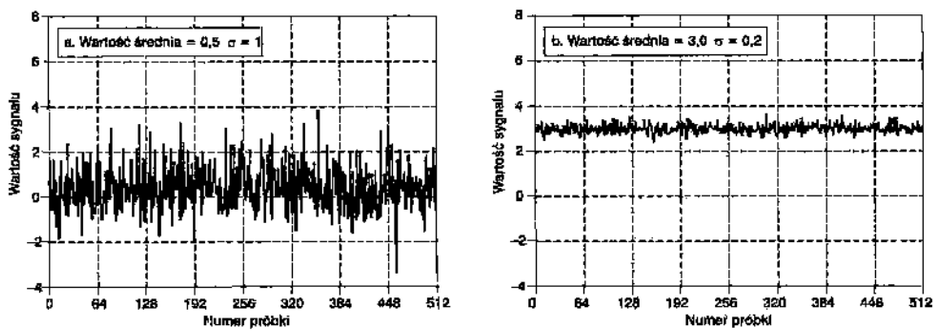
W cyfrowym przetwarzaniu sygnałów statystyka i rachunek prawdopodobieństwa są stosowane do opisu sygnałów i procesów, w których sygnały są wytwarzane. Na przykład podstawowym zastosowaniem DSP jest tłumienie zakłóceń, szumu i innych niepożądanych składników występujących w uzyskiwanych danych. Takie składniki mogą być nieodłączną częścią mierzonego sygnału lub mogą powstawać w wyniku niedoskonałości systemu akwizycji. Mogą też być wprowadzane jako niepożądany produkt uboczny niektórych operacji DSP. Statystyka i rachunek prawdopodobieństwa umożliwiają pomiar i klasyfikację tych niekorzystnych zjawisk, co jest pierwszym krokiem do opracowania metod usuwania takich kłopotliwych składników. W tym rozdziale będą omówione najważniejsze koncepcje statystyki, z podkreśleniem ich zastosowania do opisu sygnałów.

Terminologia sygnałów i wykresów

Sygnał jest opisem zależności jednego parametru od innego. Na przykład w elektronice analogowej najpospolitszym sygnałem jest *napięcie*, które zmienia się w funkcji *czasu*. Ponieważ oba te parametry mogą charakteryzować się ciągłym zakresem wartości, więc sygnał będziemy nazywać **ciągłym**. Poddanie zaś takiego sygnału przetwarzaniu analogowo-cyfrowemu powoduje *kwantyzację* obu parametrów. Jako przykład wyobraźmy sobie przetwarzanie 12-bitowe z częstotliwością próbkowania 1000 próbek/s. Reprezentacja napięcia jest ograniczona do 4096 (2^{12}) możliwych poziomów binarnych, a czas jest wyrażany tylko przyrostami równymi 1 ms. Sygnały utworzone na podstawie parametrów w ten sposób skwantowanych są nazywane sygnałami **dyskretnymi** lub wyrażonymi **cyfrowo**. Większość sygnałów występujących w przyrodzie ma charakter ciągły, dyskretne zaś są wewnątrz komputerów (choć w obu tych przypadkach bywają wyjątki). Mogą też istnieć sygnały, w których jeden parametr jest ciągły, a drugi dyskretny. Takie sygnały są dość rzadko spotykane, dlatego nie nadano im specjalnych nazw i trzeba wyraźnie określać charakter każdego z dwóch parametrów.

Na rys. 2.1 przedstawiono dwa sygnały dyskretne, które można spotkać w systemach akwizycji danych cyfrowych. **Oś pionowa** może wyrażać napięcie, natężenie światła, ciśnienie dźwięku lub nieskończoną liczbę innych parametrów. Ponieważ nie wiemy, co jest wyrażone na tej osi w każdym szczególnym przypadku, więc damy osi etykietę ogólną: **wartość sygnału**[2]. Ten parametr jest też nazywany **zmienną zależną**, a oś – **osią y** lub **osią rzędnych**.

Oś pozioma wyraża drugi parametr sygnału. Jest też nazywana **osią x** lub **osią odciętych**, a parametr – **zmienną niezależną**. Na osi poziomej sygnału najczęściej występuje *czas*, lecz w niektórych zastosowaniach mogą się pojawiać inne wielkości. Na przykład geofizycy mogą



Rys. 2.1. Przykłady dwóch sygnałów przetworzonych na cyfrowe, o różnych wartościach średnich i różnych odchyleniach standardowych

[2] Autor do określenia wartości chwilowych sygnału używa słowa *amplitude*. Po polsku zaś amplituda oznacza wartość maksymalną, np. okresowego przebiegu sinusoidalnego lub impulsu (przyp. tłum.).

mierzyć gęstość skał w równych, jednakowych odległościach na powierzchni Ziemi. W tej sytuacji parametrem opisującym oś poziomą będzie po prostu **numer próbki**. Gdyby to był sygnał ciągły, można by użyć innego opisu, jak np. *czas*, *odległość*, *x* itp.

Dwa parametry tworzące sygnał są na ogół niezamienialne. O parametrze na osi *y* (zmienna zależna) mówi się jako o **funkcji** parametru na osi x (zmiennej niezależnej). Inaczej mówiąc, zmienna niezależna opisuje *jak* i *kiedy* jest pobierana każda próbka, podczas gdy zmienna zależna jest wynikiem pomiaru. Wybierając jakąś wartość na osi x, możemy zawsze znaleźć odpowiadającą jej wartość na osi y, lecz stwierdzenie odwrotne na ogół nie obowiązuje.

Szczególną uwagę proszę zwrócić na słowo *dziedzina*, bardzo często używane w DSP. Na przykład o sygnale, dla którego zmienną niezależną (tj. parametrem na osi poziomej) jest czas, mówi się jako o sygnale w **dziedzinie czasu**. Dla innego, często występującego w DSP sygnału, zmienną niezależną jest częstotliwość; jest to sygnał **w dziedzinie częstotliwości**. Podobnie sygnały, dla których zmienną niezależną jest odległość, są sygnałami w **dziedzinie przestrzeni** (odległość jest wielkością związaną z przestrzenią). Rodzaj parametru na osi poziomej określa dziedzinę sygnału – to proste. A jak to jest w przypadku, gdy na osi poziomej jest wyrażona jakaś wielkość o charakterze ogólnym, jak np. *numer próbki*? Autorzy publikacji traktują na ogół taki sygnał jako sygnał *w dziedzinie czasu*. Próbkowanie odbywa się bowiem na ogół w równych odstępach czasu, a trudno znaleźć jakąś inną specjalną nazwę dla takich sygnałów.

Chociaż na rys. 2.1 przedstawiono sygnały dyskretne, to jednak narysowano je liniami ciągłymi. Próbek jest bowiem zbyt wiele, aby mogły być rozróżnialne jako pojedyncze punkty. Na wykresach wyrażających krótsze sygnały, powiedzmy o 100 próbkach, zwykle pokazuje się poszczególne punkty. Punkty mogą być lub nie łączone liniami ciągłymi, w zależności od sposobu, w jaki autor chce zaprezentować dane. Linia ciągła może na przykład sugerować, co się dzieje *pomiędzy* próbkami lub po prostu ułatwiać oku czytelnika śledzenie tendencji przebiegu wśród danych zaszumionych. Należy więc sprawdzić oznaczenie osi poziomej, aby stwierdzić, czy mamy do czynienia z sygnałem dyskretnym, czy ciągłym. Nie polegajcie tylko na zdolnościach ilustratora do zaznaczania punktów.

Zmienna N jest w DSP używana zazwyczaj do wyrażania całkowitej liczby próbek w sygnale. Na przykład dla sygnałów na rys. 2.1 $N = 512$. W celu uporządkowania danych przyporządkowuje się każdej próbce jej **numer**, czyli **indeks**. Te liczby pojawiają się wzdłuż osi poziomej. Stosuje się dwa rodzaje sposobu zapisu numerów próbek. W pierwszym zapisie numery próbek zmieniają się od 1 do N (np. od 1 do 512), a w drugim od 0 do $N-1$ (np. od 0 do 511). Matematycy często stosują pierwszy sposób (od 1 do N), a specjaliści od DSP – drugi (od 0 do $N-1$). W tej książce będziemy używać drugiego zapisu. Nie traktujcie tego problemu jako trywialnego. Na pewno nieraz sprawi wam *kłopot*. Zwracajcie więc uwagę na tę kwestię!

Wartość średnia i odchylenie standardowe

Średnią, oznaczaną jako μ (mała grecka litera *mi*), jest w potocznym języku statystyków nazywana średnia wartość sygnału. Wyznacza się ją tak, jak można się tego spodziewać, a więc przez zsumowanie wartości wszystkich próbek i podzielenie przez N. W postaci matematycznej wygląda to następująco:

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i$$

Wzór 2.1. Obliczanie wartości średniej sygnału. Sygnał zawiera się od x_0 do x_{N-1}, i jest indeksem zmieniającym się w tym zakresie wartości, μ jest wartością średnią

Wyrażając to słownie: trzeba zsumować wartości sygnału x_i, zmieniając indeks i od 0 do $N-1$. Kończymy obliczanie, dzieląc uzyskaną sumę przez N. Jest to taki sam wzór jak: $\mu_i = (x_0 + x_1 + x_2 + ... + x_{N-1})/N$. Jeśli nie znacie jeszcze oznaczenia Σ (duża litera grecka *sigma*) stosowanego jako symbol *sumowania*, dokładnie przestudiujcie te równania i porów-

najcie z programem komputerowym w tablicy 2.1. Takie sumowania często pojawiają się w DSP, dlatego trzeba je w pełni zrozumieć.

W elektronice *wartość średnia* jest zwykle nazywana wartością **DC** (składowa stałoprądowa). Wartość **AC** (składowa zmiennoprądowa) określa zaś, jak sygnał fluktuuje wokół wartości średniej. Jeśli sygnał jest prostym, powtarzającym się przebiegiem, takim jak sinusoida lub przebieg prostokątny, to jego odchylenia od średniej mogą być wyrażane przez amplitudę szczyt–szczyt. Niestety większość sygnałów nie ma wyraźnie określonej wartości szczyt–szczyt, lecz ma charakter losowy, tak jak sygnały na rys. 2.1. W takich przypadkach trzeba stosować metody ogólniejsze i parametr zwany **odchyleniem standardowym**, oznaczany σ (mała grecka litera *sigma*).

Zacznijmy od tego, że wyrażenie $|x_i - \mu|$ określa, jak dalece i-ta próbka *odchyla* się od średniej. *Odchylenie średnie* sygnału wyznacza się, sumując odchylenia wszystkich poszczególnych próbek i następnie dzieląc tę sumę przez liczbę próbek N. Zauważmy, że do sumowania bierzemy wartości bezwzględne odchyleń, w przeciwnym razie wartości ujemne i dodatnie dałyby w wyniku zero. Odchylenie średnie jest liczbą określającą typowe odchylenie próbki od średniej. Odchylenie średnie, choć proste i wygodne, prawie nigdy nie jest używane w statystyce. Wiąże się to z fizycznymi właściwościami sygnałów. W większości przypadków ważnym parametrem nie jest *odchylenie* od średniej, lecz *moc* wyrażona przez to odchylenie. Na przykład gdy sygnały losowe łączą się z sobą w układzie elektronicznym, to szum wypadkowy wynika z sumowania *mocy* poszczególnych sygnałów, a nie ich *amplitud*.

Odchylenie standardowe jest parametrem podobnym do *odchylenia średniego*, z tą wszakże różnicą, że w tym przypadku następuje uśrednianie mocy, a nie amplitud. Uzyskuje się to, podnosząc wartości odchyleń do kwadratu przed wyznaczeniem średniej (pamiętamy, że moc jest proporcjonalna do kwadratu napięcia). Następnie wyciąga się pierwiastek kwadratowy, aby „skompensować" poprzednie podnoszenie do kwadratu. Odchylenie standardowe można więc wyrazić wzorem:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - \mu)^2$$

Wzór 2.2. Obliczenie odchylenia standardowego sygnału. Sygnał jest zawarty w próbce x_i, μ jest średnią obliczoną z wzoru 2.1, N jest liczbą próbek a σ – odchyleniem standardowym

Powyższy wzór można też zapisać w postaci:

$$\sigma = \sqrt{(x_0 - \mu)^2 + (x_1 - \mu)^2 + ... + (x_{N-1} - \mu)^2 / (N-1)}$$

Proszę zauważyć, że wartość średnią uzyskuje się przez dzielenie przez $N - 1$ zamiast przez N. Tę delikatną sprawę przeanalizujemy w następnym podrozdziale. Wyraz σ^2 często występuje w statystyce i nosi nazwę **wariancji**. Odchylenie standardowe jest miarą wartości fluktuacji sygnału od średniej. Wariancja wyraża moc tych fluktuacji. Inną wielkością, z którą trzeba się zapoznać, jest wartość **rms** (wartość średniokwadratowa – *root mean square*), często stosowana w elektronice. Odchylenie standardowe, z definicji, określa część zmiennoprądową (AC) sygnału, podczas gdy wartość rms jest miarą zarówno składowej stałoprądowej (DC), jak i zmiennoprądowej (AC). Jeśli sygnał nie zawiera składowej stałej, to wartość średniokwadratowa (rms) jest identyczna jak odchylenie standardowe. Na rys. 2.2 przedstawiono zależności między odchyleniem standardowym i wartością międzyszczytową (*peak-to-peak*) dla kilku często spotykanych przebiegów.

W tablicy 2.1 podano komputerową procedurę obliczania wartości średniej i odchylenia standardowego przy użyciu wzorów 2.1 i 2.2. Programy w tej książce mają, w zamierzeniu autora, wyrażać *algorytmy* w najprostszy sposób. Wszystkie inne kwestie są traktowane jako wtórne. Rezygnuje się tu z dobrej techniki programowania, jeśli dzięki temu uzyskuje się program bardziej logiczny i o większej jasności. Na przykład: stosuje się uproszczoną wersję BASIC, podaje się numerację wierszy, jedyną dopuszczalną strukturą sterującą jest FOR-NEXT, nie

stosuje się instrukcji wejścia/wyjścia itd. Potraktujcie te programy jako alternatywny sposób zrozumienia wzorów używanych w DSP. Jeśli nie zrozumiecie jednego, to może któryś inny okaże się pomocny. W języku BASIC znak % na końcu nazwy zmiennej oznacza, że jest to liczba całkowita. Wszystkie inne zmienne są zmiennoprzecinkowe. Rodzaje tych zmiennych omówiono szczegółowo w rozdziale 4.

Opisana metoda obliczania wartości średniej i odchylenia standardowego nadaje się do wielu zastosowań; ma jednak dwa ograniczenia. Po pierwsze, jeśli wartość średnia jest znacznie większa niż odchylenie standardowe, to we wzorze 2.2 mamy odejmowanie dwóch liczb o wartościach bardzo zbliżonych do siebie. W obliczeniach może to prowadzić do zbyt dużych błędów zaokrąglania. Tą kwestią zajmiemy się szczegółowo w rozdziale 4. Po drugie, często trzeba ponownie obliczać wartość średnią i odchylenie standardowe w miarę pobierania nowych próbek i dodawania ich do sygnału. Ten rodzaj obliczeń będziemy nazywać **statystyką bieżącą**. Gdy metodę opisaną wzorami 2.1 i 2.2 chcemy zastosować do statystyki bieżącej, to w każdym nowym obliczeniu musimy uwzględniać *wszystkie* próbki. Jest to bardzo nieefektywne ze względu na możliwości obliczeniowe i pamięć komputera.

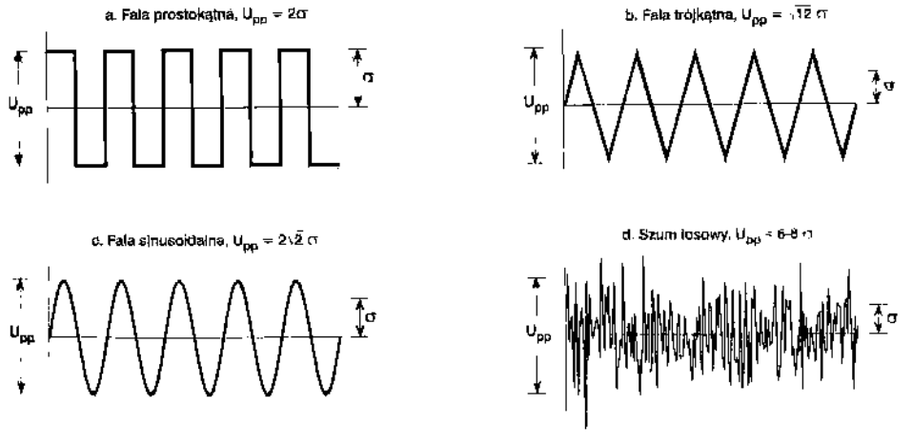
Tę trudność można pokonać, przekształcając wzory 2.1 i 2.2 w celu uzyskania innego wzoru do obliczania odchylenia standardowego:

$$\sigma^2 = \frac{1}{N-1}\left[\sum_{i=0}^{N-1} x_i^2 - \frac{1}{N}\left(\sum_{i=0}^{N-1} x_i\right)^2\right]$$

lub stosując prostszy sposób zapisu

$$\sigma^2 = \frac{1}{N-1}\left[\text{suma kwadratów} - \frac{\text{suma}^2}{N}\right]$$

Wzór 2.3. Obliczenie odchylenia standardowego przy użyciu statystyki bieżącej. Ten wzór daje taki sam wynik jak wzór 2.2, lecz z mniejszym błędem wywołanym zaokrąglaniem i z mniejszymi wymaganiami dotyczącymi możliwości obliczeniowych komputera. Sygnał jest wyrażony za pomocą trzech akumulowanych parametrów: N – całkowitej liczby próbek, sumy próbek, sumy kwadratów próbek. Następnie na podstawie tych parametrów akumulowanych oblicza się wartość średnią i odchylenie standardowe



Rys. 2.2. Stosunek wartości międzyszczytowej (peak-to-peak) do odchylenia standardowego dla kilku często spotykanych przebiegów. Dla fali prostokątnej ten stosunek jest równy 2; dla fali trójkątnej jest $\sqrt{12} = 3,46$; dla sinusoidy $2\sqrt{2} = 2,83$. Szum losowy nie ma dokładnie określonej wartości międzyszczytowej, jest ona w przybliżeniu od 6 do 8 razy większa od odchylenia standardowego

Tab. 2.1.

```
100 'OBLICZANIE WARTOŚCI ŚREDNIEJ I ODCHYLENIA STANDARDOWEGO
110 '
120 DIM X[511]            'Sygnał jest przechowywany w X[0] do X[511]
130 N% = 512              'N% jest liczbą punktów w sygnale
140
150 GOSUB XXXX            'Podprogram "mityczny" ładujący sygnał do X[ ]
160 '
170 MEAN =0               'Wyznacz wartość średnią z wzoru 2.1
180 FOR I% = 0 TO N%-1
190   MEAN = MEAN + X[I%]
200 NEXT I%
210 MEAN = MEAN/N%
220 '
230 VARIANCE = 0          'Wyznacz odchylenie standardowe z wzoru 2.2
240 FOR I% = 0 TO N%-1
250   VARIANCE = VARIANCE + (X[I%] - MEAN)^2
260 NEXT I%
270 VARIANCE = VARIANCE/(N%-1)
280 SD = SQR(VARIANCE)
290 '
300                       'Drukuj obliczoną wartość średnią i odchylenie standardowe
310 '
320 END
```

Uwaga: Pojęcie podprogramu „mitycznego" wyjaśniono przy tablicy 6.1.

W trakcie próbkowania sygnału jest pamiętany bieżący wynik obliczania trzech parametrów: 1/ liczby próbek już przetworzonych, 2/ sumy tych próbek, 3/ sumy kwadratów tych próbek (tzn. podnosi się do kwadratu wartość każdej próbki i dodaje wynik do wartości skumulowanej). Po przetworzeniu dowolnej liczby próbek można obliczyć wartość średnią i odchylenie standardowe, korzystając tylko z bieżącej wartości tych trzech parametrów. W tablicy 2.2 zamieszczono program, który w wyżej opisany sposób wyznacza wartość średnią i odchylenie standardowe po uwzględnieniu każdej nowej próbki. Ta metoda jest stosowana w przenośnych kalkulatorach do określania właściwości statystycznych ciągu liczb. Wartości trzech wymienionych parametrów są uaktualniane za każdym razem, gdy wprowadzamy nową liczbę i naciskamy przycisk sumowania. W każdej chwili można uzyskać wartość średnią i odchylenie standardowe bez konieczności przeliczania na nowo całego ciągu liczb.

Zanim zakończymy omawianie wartości średniej i odchylenia standardowego, powinniśmy jeszcze wspomnieć dwie inne wielkości. W niektórych sytuacjach *średnia* wyraża wartość zmierzoną, podczas gdy *odchylenie standardowe* wyraża szum i inne zakłócenia. W tych przypadkach odchylenie standardowe nie jest istotne samo w sobie, lecz tylko w *porównaniu* ze średnią. Stąd bierze się parametr **stosunek sygnału do szumu** (**SNR** – *Signal-to-Noise Ratio*) równy stosunkowi wartości średniej do odchylenia standardowego. Inny parametr to **współczynnik rozrzutu** (**CV** – *Coefficient of Variation*). Jest to stosunek odchylenia standardowego do wartości średniej, wyrażony w procentach. Na przykład sygnał (lub inna grupa zmierzonych wartości) o współczynniku rozrzutu 2% ma stosunek sygnału do szumu równy 50. „Lepsze" dane mają *większy* stosunek sygnału do szumu a *mniejszy* współczynnik rozrzutu.

Sygnał a proces przezeń reprezentowany

Statystyka jest dziedziną wiedzy służącą do interpretacji *danych liczbowych*, takich jak pobrane sygnały. **Prawdopodobieństwo** zaś jest stosowane w DSP do zrozumienia *procesów* wytwarzających sygnały. Choć **pobrany sygnał** i **proces** przez niego reprezentowany są ściśle powiązane, to jednak rozróżnienie między nimi jest kluczem do wielu metod DSP.

Wyobraźmy sobie na przykład utworzenie sygnału złożonego z 1000 punktów przez 1000-krotne rzucanie monety. Jeśli moneta upadnie orłem do góry, to próbka ma wartość 1; jeśli zaś reszką – wartość 0. *Proces* ten ma wartość średnią równą dokładnie 0,5; określoną przez wartości procentowych prawdopodobieństw każdego z rzutów: 50% orzeł i 50% reszka. Jest jednak nieprawdopodobne, żeby sygnał 1000-punktowy miał wartość średnią równą dokładnie 0,5. Losowy charakter procesu powoduje, że w każdym z tworzonych sygnałów liczby zer i jedynek będą się od siebie nieco różnić. *Prawdopodobieństwa* procesów są stałe, lecz *statystyka* pobieranego sygnału przy każdym powtórzeniu eksperymentu się zmienia. Ta losowa nieregularność występująca w danych rzeczywistych jest określana takimi nazwami jak: **odchylenie statystyczne, fluktuacja statystyczna** i **szum statystyczny**.

Powstaje tu pewien dylemat. Gdy spotykamy określenia: *odchylenie standardowe* i *wartość średnia*, to skąd mamy wiedzieć, czy autor ma na myśli statystykę rzeczywistego sygnału, czy prawdopodobieństwo w procesie, który jest reprezentowany przez sygnał? Niestety, można to stwierdzić tylko na podstawie kontekstu. Nie dotyczy to wszystkich pojęć stosowanych w statystyce i rachunku prawdopodobieństwa. Na przykład *histogram* i *funkcja masy prawdopodobieństwa* (omawiane w następnym podrozdziale) są dobrze pasującymi do siebie koncepcjami, którym nadano oddzielne nazwy.

Wróćmy teraz do wzoru 2.2 służącego do obliczania odchylenia standardowego. Jak już wspomniano, obliczając według tego wzoru średnią kwadratów odchyleń, wykonuje się dzielenie przez $N-1$, a nie po prostu przez N. Aby to zrozumieć, wyobraźmy sobie, że chcemy obliczyć wartość średnią i odchylenie standardowe jakiegoś *procesu* wytwarzającego sygnały. Pobieramy sygnał złożony z N próbek pochodzących z tego procesu i ze wzoru 2.1 obliczamy wartość średnią tego sygnału. Można ją później wykorzystać jako *estymatę* średniej wartości tego procesu. Zdajemy sobie jednak sprawę, że pojawia się tu pewien szum statystyczny.

Tab. 2.2.

```
100 'WARTOŚĆ ŚREDNIA I ODCHYLENIE STANDARDOWE PRZY UŻYCIU STATYSTYKI BIEŻĄCEJ
110 '
120 DIM X[511]          'Sygnał jest przechowywany w X[0] do X[511]
130 '
140 GOSUB XXXX          'Podprogram "mityczny" ładujący sygnał do X
150 '
160 N% = 0              'Zeruj trzy bieżące parametry
170 SUM = 0
180 SUMSQUARES = 0
190 '
200 FOR I% = 0 TO 511 'Pętla przez wszystkie próbki w sygnale
210   '
220   N% = N%+1         'Uaktualnij te trzy parametry
230   SUM = SUM + X[I%]
240   SUMSQUARES = SUMSQUARES + X[I%]^2
250   '
260   MEAN = SUM/N%   'Oblicz wartość średnią i odchylenie standardowe z wzoru 2.3
270   IF N% = 1 THEN SD = 0: GOTO 300
280   SD = SQR((SUMSQUARES - SUMA^2/N%) / (N%-1))
290   '
300   PRINT MEAN, SD  'Drukuj bieżące wartości średniej i odchylenia standardowego
310   '
320 NEXT I%
330 '
340 END
```

W szczególności, dla sygnałów losowych, typowy błąd między wartością średnią dla N punktów a wartością średnią procesu jest określony wzorem:

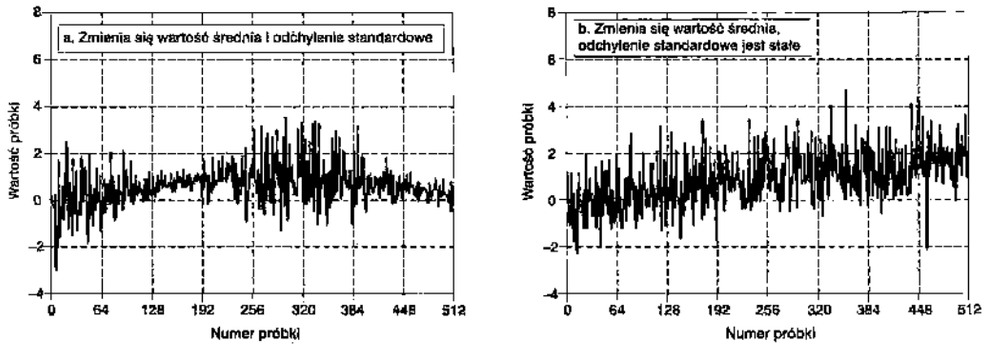
$$\text{Typowy błąd} = \frac{\sigma}{N^{\frac{1}{2}}}$$

Wzór 2.4. Typowy błąd obliczenia wartości średniej procesu przy wykorzystaniu skończonej liczby próbek N. Parametr σ jest odchyleniem standardowym

Jeśli N jest małe, to szum statystyczny w obliczonej średniej będzie bardzo duży. Inaczej mówiąc, nie mamy dostępu do ilości danych wystarczających do prawidłowego scharakteryzowania procesu. Im większa jest wartość N, tym mniejszy będzie spodziewany błąd. Z *mocnego prawa wielkich liczb*, będącego kamieniem milowym w rozwoju teorii prawdopodobieństwa, wynika, że błąd staje się równy zeru, gdy N dąży do nieskończoności.

W następnym kroku chcielibyśmy obliczyć odchylenie standardowe pobranego sygnału i zastosować je jako estymatę odchylenia standardowego procesu reprezentowanego przez sygnał. I tu pojawia się pewna trudność. Zanim obliczymy odchylenie standardowe z wzoru 2.2, powinniśmy już znać średnią μ. Jednak nie znamy średniej wartości procesu, a tylko średnią N-punktowego sygnału, *zawierającą błąd spowodowany szumem statystycznym*. Ten błąd może powodować zmniejszenie obliczonej wartości odchylenia standardowego. Aby to skompensować, zastępuje się N przez $N-1$. Jeśli N jest duże, to ta różnica jest bez znaczenia. Jeśli zaś N jest małe, to takie zastąpienie daje dokładniejszą estymatę standardowego odchylenia procesu. A więc wzór 2.2 jest *estymatą* odchylenia standardowego *procesu reprezentowanego przez sygnał*. Gdybyśmy we wzorze mieli dzielenie przez N, to uzyskalibyśmy odchylenie standardowe *pobranego sygnału*.

A teraz spójrzmy na rys. 2.3 będący ilustracją omawianych problemów i zadajmy pytanie: czy wahania występujące w tych sygnałach są wynikiem szumu statystycznego, czy zmian zachodzących w procesie będącym źródłem sygnału? Nietrudno chyba dojść do przekonania, że te zmiany są zbyt duże jak na zjawiska losowe i muszą wynikać z procesu będącego źródłem sygnału. Procesy, które zmieniają w ten sposób swoje właściwości nazywamy **niestacjonarnymi**. Sygnały zaś przedstawione poprzednio na rys. 2.1 były generowane w procesie stacjonarnym i wahania były powodowane wyłącznie szumem statystycznym. Na rys. 2.3 b) przedstawiono problem często spotykany w przypadku sygnałów niestacjonarnych: wolno zmieniająca się *wartość średnia* zakłóca obliczanie *odchylenia standardowego*. W tym przy-

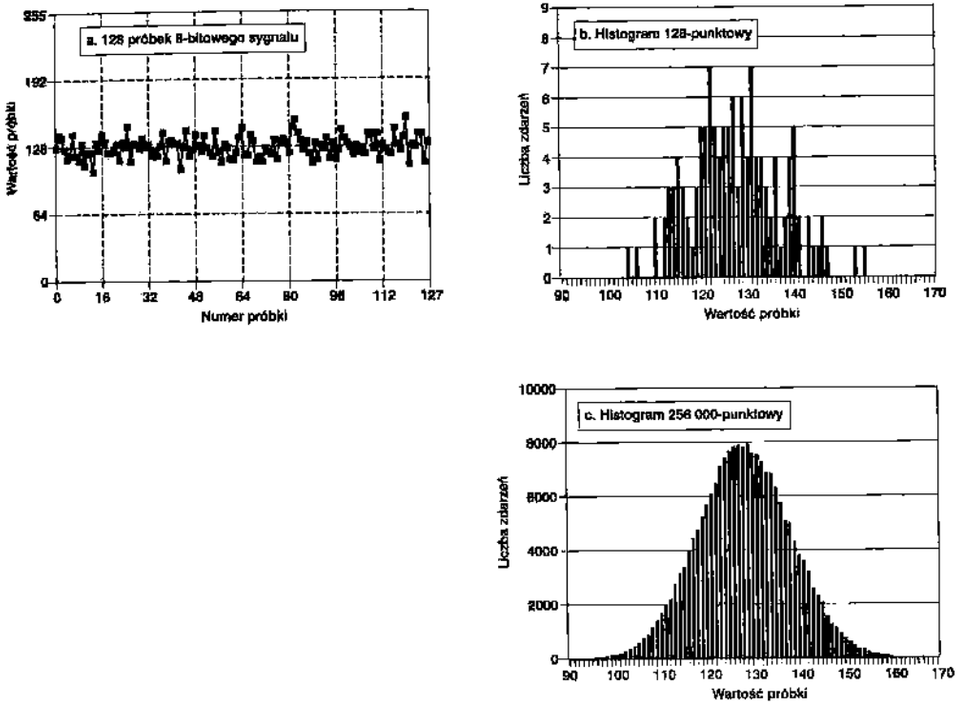


Rys. 2.3. Przykłady sygnałów generowanych w procesach niestacjonarnych. Na wykresie a) zmienia się zarówno wartość średnia, jak i odchylenie standardowe. Na wykresie b) odchylenie standardowe pozostaje stałe i jest równe 1, podczas gdy wartość średnia zmienia się od 0 do 2. Powszechnie stosowaną metodą analizy jest podział takich sygnałów na krótkie odcinki i obliczanie parametrów statystycznych każdego odcinka osobno

kładzie odchylenie standardowe sygnału w krótkim przedziale jest równe *jeden*. Jednak odchylenie standardowe całego sygnału jest równe 1,16. Ten błąd można niemal całkowicie wyeliminować dzieląc sygnał na krótkie odcinki i obliczając parametry statystyczne każdego odcinka osobno. Jeśli trzeba, można uśrednić odchylenia statystyczne wszystkich odcinków uzyskując jedną wartość.

Histogram, pmf, pdf

Załóżmy, że do komputera mamy przyłączony 8-bitowy przetwornik a/c i że zbieramy 256 000 próbek jakiegoś sygnału. Na rys. 2.4 a) przedstawiono, jako przykład, 128 próbek, które mogą być częścią tego zestawu danych. Każda z próbek może przyjmować jedną z 256 możliwych wartości: od 0 do 255. **Histogram** pokazuje, jakie są w sygnale *liczby próbek* o określonych wartościach, dla każdej z *możliwych wartości*. Na rys. 2.4 b) przedstawiono histogram dla 128 próbek z rys. 2.4 a). Są na przykład 2 próbki o wartości 110; 7 próbek o wartości 131; 0 próbek o wartości 170 itd. Oznaczmy histogram przez H_i, gdzie i jest indeksem zmieniającym się od 0 do $M-1$, a M jest liczbą możliwych wartości próbki. Na przykład H_{50} jest liczbą próbek mających wartość 50. Na rys. 2.4 c) przedstawiono histogram sygnału obejmujący wszystkie próbki, a więc 256k punktów. Jak można zauważyć, przy większej liczbie próbek histogram ma przebieg bardziej gładki. Tak jak w przypadku wartości średniej, szum statystyczny histogramu (jego chropowatość) jest odwrotnie proporcjonalny do pierwiastka kwadratowego z liczby użytych próbek.



Rys. 2.4. Przykłady histogramów. Na rys. a) przedstawiono 128 próbek z bardzo długiego sygnału, każda próbka może przyjmować wartość całkowitą od 0 do 255. Na rys. b), c) przedstawiono histogramy, stosując, odpowiednio, 128 i 256 000 próbek z tego długiego sygnału. Jak pokazano na rysunkach, histogram jest tym gładszy, im więcej próbek użyto

Z definicji histogramu wynika, że suma wszystkich liczb próbek w histogramie musi być równa liczbie punktów w sygnale:

$$N = \sum_{i=0}^{M-1} H_i$$

Wzór 2.5. Suma wszystkich liczb próbek w histogramie jest równa liczbie punktów w sygnale. W tym wzorze H_i jest histogramem, N – liczbą punktów w sygnale, a M – liczbą punktów w histogramie

Histogramu można użyć do efektywnego obliczania wartości średniej i odchylenia standardowego w bardzo dużych zestawach danych. Jest to ważne zwłaszcza w przypadku *obrazów*, które mogą zawierać miliony próbek. Histogram grupuje razem próbki mające takie same wartości. Umożliwia to obliczanie parametrów statystycznych z wykorzystaniem kilku grup zamiast wielkiej liczby próbek indywidualnych. Przy użyciu tego sposobu oblicza się wartość i odchylenie standardowe z wzorów:

$$\mu = \frac{1}{N} \sum_{i=0}^{M-1} i H_i$$

Wzór 2.6. Obliczanie wartości średniej na podstawie histogramu. Można to traktować jako łączenie w grupy wszystkich próbek o jednakowych wartościach i następnie obliczanie z wzoru 2.1 wartości średniej dla każdej grupy

$$\sigma^2 = \frac{1}{N-1} \sum_{i=0}^{M-1} (i-\mu)^2 H_i$$

Wzór 2.7. Obliczanie odchylenia standardowego na podstawie histogramu. Koncepcja jest taka sama jak we wzorze 2.2, poza tym że wszystkie próbki o jednakowych wartościach są od razu brane do obliczeń

W tablicy 2.3 zamieszczono program do obliczania histogramu, wartości średniej i odchylenia standardowego przy użyciu powyższych wzorów. Obliczanie histogramu jest bardzo szybkie, gdyż wymaga tylko indeksowania i inkrementacji. Obliczanie wartości średniej i odchylenia standardowego wymaga zaś czasochłonnych operacji dodawania i mnożenia. Strategia tego algorytmu polega na zastosowaniu tych wolnych operacji tylko do niewielu liczb w histogramie, a nie do wielu próbek w sygnale. Dzięki temu ten algorytm jest znacznie szybszy niż metody opisane poprzednio. Jest to czynnik ok. 10 dla sygnałów bardzo długich, w przypadku obliczeń na komputerach ogólnego zastosowania.

Bardzo ważne jest zwrócenie uwagi na fakt, że pobrany sygnał jest zaszumioną wersją procesu, który reprezentuje. To stwierdzenie jest tak istotne, że powoduje zróżnicowanie nazw niektórych pojęć. I tak *histogramem* jest wykres utworzony na podstawie pobranego sygnału. Odpowiadająca mu krzywa, opisująca proces, jest nazywana **funkcją masy prawdopodobieństwa** (**pmf** – *probability mass function*). Histogram jest zawsze obliczany przy użyciu skończonej liczby próbek, podczas gdy *pmf* jest wykresem, który *byłby* uzyskany z nieskończonej liczby próbek. Wykres *pmf* może być oszacowany (wywnioskowany) na podstawie histogramu lub wydedukowany metodami matematycznymi, jak w przykładzie z rzucaniem monetą.

Na rys. 2.5 przedstawiono przykład wykresu *pmf* oraz jednego z możliwych histogramów, który mógłby być z nim związany. Kluczem do zrozumienia tych pojęć są jednostki opisujące oś pionową. Jak to już poprzednio powiedziano, na osi pionowej histogramu jest przedstawiona *liczba występowania* określonych wartości sygnału. Oś pionowa wykresu *pmf* zawiera podobną informację, lecz wyrażoną *procentowo*. Inaczej mówiąc, aby uzyskać przybliżony wykres *pmf*, dzieli się każdą wartość histogramu przez całkowitą liczbę próbek. Wynika stąd, że wszystkie wartości wykresu *pmf* muszą mieścić się w granicach do 0 do 1 i że suma wszystkich wartości w tym wykresie jest równa 1.

Wykres *pmf* jest istotny, gdyż opisuje *prawdopodobieństwo* występowania określonych wartości. Wyobraźmy sobie, na przykład, sygnał o wykresie *pmf* z rys. 2.5 b) , taki jak poprzednio pokazano na rys. 2.4 a). Jakie jest prawdopodobieństwo, że próbka pobrana z tego sygnału będzie miała wartość 120? Odpowiedź daje wykres na rys. 2.5 b) – prawdopodobieństwo

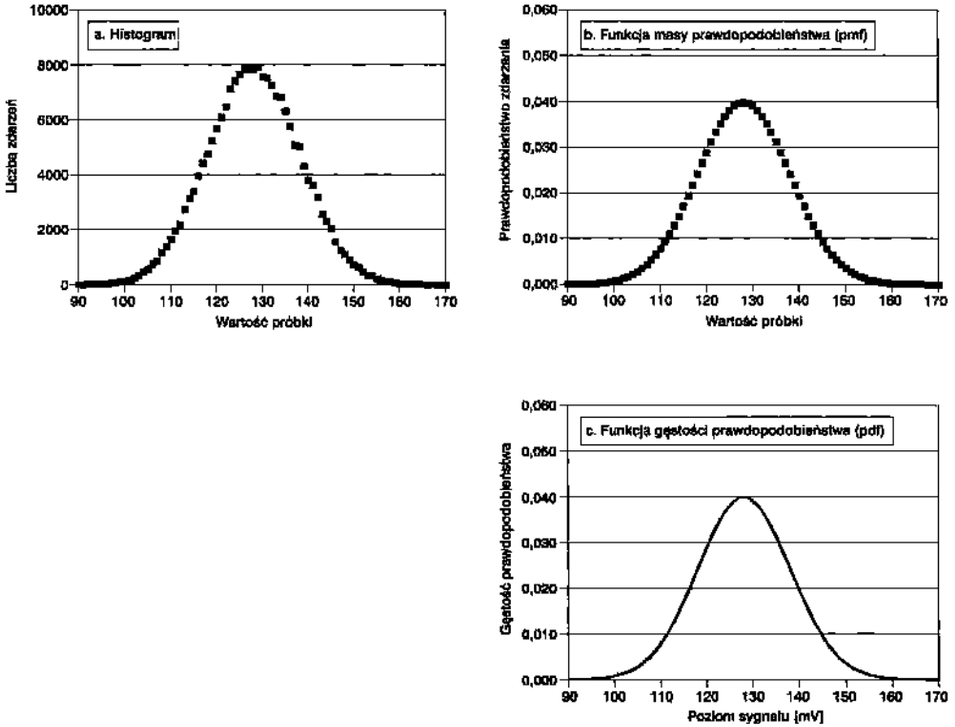
Tab. 2.3.

```
100 'OBLICZANIE HISTOGRAMU, WARTOŚCI ŚREDNIEJ I ODCHYLENIA STANDARDOWEGO
110 '
120 DIM X%[25000]      'Przetwarzany sygnał jest przechowywany od X%[0] do X%[25000]
130 DIM H%[255]        'Histogram jest przechowywany od H%[0] do H%[255]
140 N% = 25001         'Ustaw liczbę punktów w sygnale
150 '
160 FOR I% = 0 TO 255 'Wyzeruj histogram, aby mógł być użyty jako akumulator
170   H%[I%] = 0
180 NEXT I%
190 '
200 GOSUB XXXX         'Podprogram "mityczny" ładujący sygnał do X%[ ]
210 '
220 FOR I% = 0 TO 25000  'Oblicz histogram dla 25001 punktów
230   H%[X%[I%]] = H%[X%[I%]] + 1
240 NEXT I%
250 '
260 MEAN = 0           'Oblicz średnią z wzoru 2.6
270 FOR I% = 0 TO 255
280   MEAN = MEAN + I% * H%[I%]
290 NEXT I%
300 MEAN = MEAN / N%
310 '
320 VARIANCE = 0       'Oblicz odchylenie standardowe z wzoru 2.7
330 FOR I% = 0 TO 255
340   VARIANCE = VARIANCE + H%[I%] * (I%-MEAN)^2
350 NEXT I%
360 VARIANCE = VARIANCE / (N%-1)
370 SD = SQR(VARIANCE)
380 '
390 PRINT MEAN SD      'Drukuj obliczoną wartość średnią i odchylenie standardowe
400 '
410 END
```

jest 0,03, czyli w przybliżeniu 1 szansa na 34. A jakie jest prawdopodobieństwo, że losowo wybrana próbka będzie miała wartość większą niż 150? Sumowanie wartości wykresu *pmf* dla: 151, 152, 153, ... 255 daje wynik równy 0,0122, czyli w przybliżeniu 1 szansa na 82. A zatem można się spodziewać, że sygnał będzie przyjmował wartość większą niż 150 średnio co w 82. punkcie próbkowania. Jakie jest prawdopodobieństwo, że wartość każdej próbki będzie się mieściła w granicach między 0 i 255? Sumując wszystkie wartości wykresu *pmf*, uzyskujemy prawdopodobieństwo równe 1,00, czyli pewność zajścia takiego zdarzenia.

Histogramy i wykresy *pmf* można stosować tylko do danych dyskretnych, takich jak przetworzony na wartości cyfrowe sygnał przechowywany w komputerze. Podobne jak *pmf* pojęcie można też zastosować do sygnałów ciągłych, np. do napięć w elektronice analogowej. **Funkcja *pdf*** (*probability density function*), zwana też **funkcją rozkładu prawdopodobieństwa**, jest tym dla sygnałów analogowych, czym funkcja masy prawdopodobieństwa dla dyskretnych. Wyobraźmy sobie na przykład sygnał analogowy, który po przetworzeniu analogowo-cyfrowym daje sygnał cyfrowy przedstawiony na rys. 2.4 a). Załóżmy dla uproszczenia, że napięcie o wartości między 0 i 255 mV jest przetwarzane na liczby od 0 do 255. Wykres *pmf* tego sygnału jest *punktowo* przedstawiono na rys. 2.5 b). Podobnie wykres *pdf* tego sygnału przedstawiono linią *ciągłą* na rys. 2.5 c), co wskazuje na to, że sygnał może przyjmować w sposób ciągły wartości z pewnego zakresu, jak np. napięcie w układzie elektronicznym.

Wartości na osi pionowej wykresu *pdf* są wyrażone w jednostkach **gęstości prawdopodobieństwa**, a nie w jednostkach prawdopodobieństwa. Na przykład wartość *pdf* 0,03 dla 120,5 nie oznacza, że napięcie 120,5 mV pojawia się w ciągu 3% czasu. W istocie, prawdopodobieństwo, że sygnał ciągły będzie miał wartość dokładnie 120,5 mV, jest nieskończenie małe.

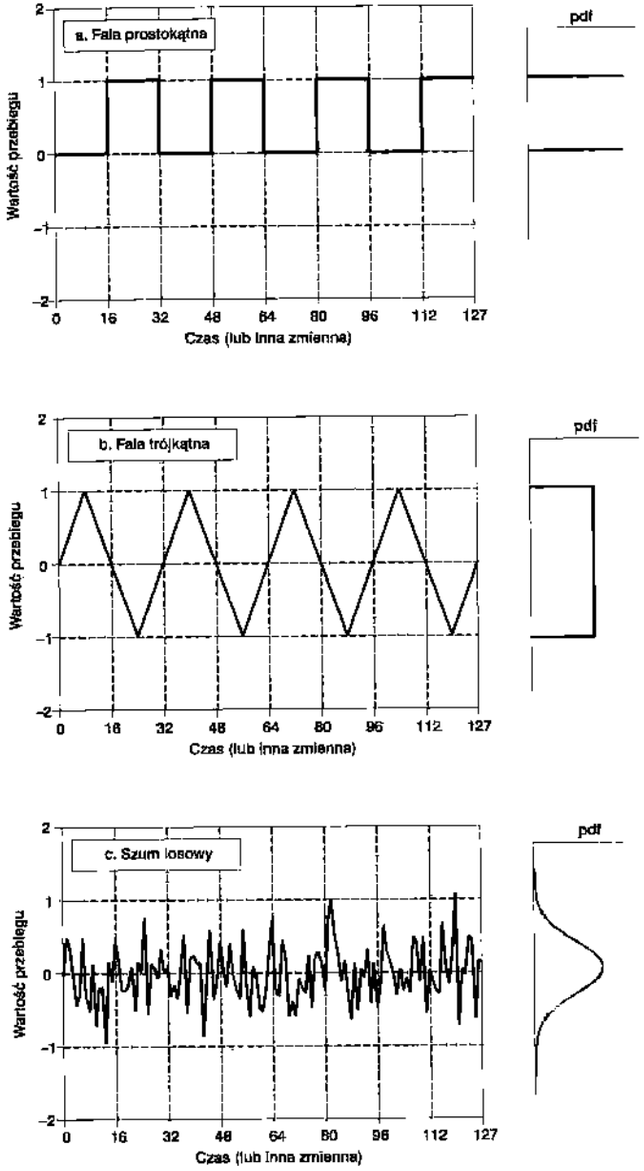


Rys. 2.5. Zależność między histogramem (a), funkcją masy prawdopodobieństwa pmf (b) i funkcją gęstości prawdopodobieństwa pdf (c). Histogram jest obliczany na podstawie skończonej liczby próbek. Wykres pmf opisuje proces reprezentowany przez sygnał. Wykres pdf jest podobny do pmf, lecz jest stosowany raczej do sygnałów ciągłych, a nie do dyskretnych. Chociaż na osiach pionowych wykresów b), c) są takie same wartości (0 do 0,06), jest to tylko zbieg okoliczności pojawiający się w tym przykładzie. Wartości na osiach pionowych tych trzech wykresów są określone przez: a) – sumę wartości w histogramie równą liczbie próbek w sygnale, b) – sumę wartości w pmf równą jeden, c) – powierzchnię pod krzywą pdf równą jeden

Jest tak, gdyż istnieje nieskończenie wiele możliwych wartości, między które sygnał musi podzielić swój czas: 120,49997; 120,49998; 120,49999 itd. Szansa, żeby sygnał miał właśnie wartość dokładnie 120,50000, jest doprawdy bardzo niewielka!

W celu obliczenia *prawdopodobieństwa* trzeba *gęstość prawdopodobieństwa* pomnożyć przez zakres wartości. Na przykład prawdopodobieństwo, że sygnał w każdej danej chwili ma wartość między 120 i 121, jest równe: (121–120) × 0,03 = 0,03. Prawdopodobieństwo, że wartość sygnału mieści się między 120,4 i 120,5 jest: (120,5–120,4) × 0,03 = 0,003 itd. Jeśli gęstość prawdopodobieństwa nie jest stała w interesującym nas zakresie, to iloczyn przekształca się w całkę *pdf* w danym zakresie, a więc powierzchnię pod wykresem *pdf* między określonymi wartościami. Ponieważ sygnał zawsze musi mieć *jakąś wartość*, więc całkowita powierzchnia pod wykresem, czyli całka od $-\infty$ do $+\infty$, będzie zawsze równa 1. Jest to analogiczne z faktem, że suma wszystkich wartości *pmf* jest równa 1 oraz że suma wszystkich wartości histogramu jest równa N.

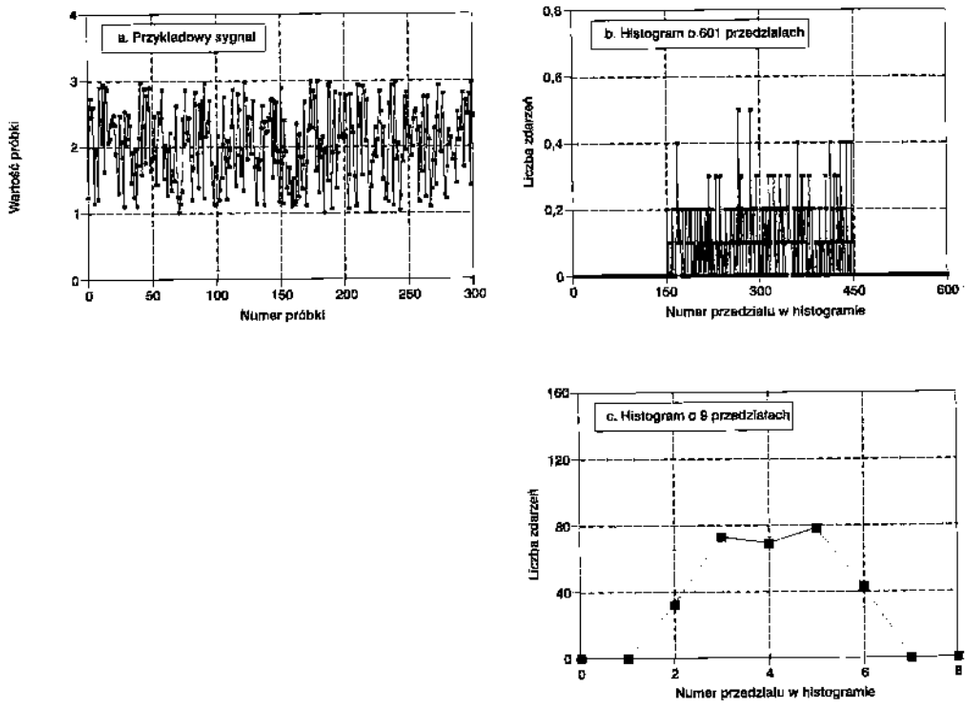
Histogram, *pmf* i *pdf* są pojęciami do siebie podobnymi. Matematycy je ściśle rozróżniają, lecz przez wielu naukowców i inżynierów są często stosowane wymiennie (a zatem błędnie). Na rys. 2.6 przedstawiono trzy przebiegi *ciągłe* i ich gęstości prawdopodobieństwa (*pdf*). Gdyby to były *sygnały dyskretne*, a więc mające numery próbek na osi poziomej, to zastosowalibyśmy wykresy masy prawdopodobieństwa (*pmf*).



Rys. 2.6. Trzy często spotykane przebiegi i ich funkcje gęstości prawdopodobieństwa. Wykres gęstości prawdopodobieństwa jest często, jak w tych przykładach, przedstawiany jako obrócony o 90° i umieszczony obok wykresu sygnału, który reprezentuje. Funkcja gęstości prawdopodobieństwa fali prostokątnej (a) składa się z dwóch nieskończenie wąskich szpilek odpowiadających sygnałowi, przyjmującymi tylko dwie możliwe wartości. Gęstość prawdopodobieństwa fali trójkątnej (b) ma stałą wartość w pewnym przedziale, jest to tzw. rozkład równomierny. Najbardziej interesujący jest wykres gęstości prawdopodobieństwa szumu losowego (c) będący funkcją o kształcie dzwonowym, znaną jako funkcja gaussowska

Trudność w obliczaniu histogramu pojawia się, gdy liczba wartości, jaką może przyjmować każda próbka, jest znacznie większa niż liczba próbek w sygnale. Dotyczy to zawsze sygnałów wyrażanych w zapisie *zmiennoprzecinkowym*, w którym każda próbka jest zapisywana jako wartość ułamkowa. W zapisie w liczbach całkowitych wartość próbki może być na przykład 3 lub 4, a w zapisie zmiennoprzecinkowym są miliony możliwych wartości *między* 3 i 4. Metoda obliczania histogramów opisana poprzednio wymaga obliczania liczby próbek mających każdą z możliwych wartości poziomów kwantyzacji. Jest to nie do zrealizowania w przypadku danych zmiennoprzecinkowych, gdyż istnieją *miliardy* możliwych poziomów, które trzeba uwzględnić. Co gorsza, prawie wszystkie z tych możliwych poziomów nie mają odpowiadających im próbek. Wyobraźmy sobie, na przykład, sygnał złożony z 10 000 próbek, z których każda ma miliard możliwych wartości. Konwencjonalny histogram zawierałby wtedy miliard punktów, z których wszystkie – oprócz 10 000 – miałyby wartość zero.

Rozwiązaniem tych trudności jest metoda polegająca na utworzeniu histogramu o dowolnie dobranej długości i punkcie początkowym, składającego się z przedziałów o określonych szerokościach (metoda zwana *binning*). Wybiera się taką długość histogramu, aby była równa jakiejś dogodnej liczbie, np. 1000 punktów; są to tzw. **przedziały** (*bins*). Wartość w każdym przedziale wyraża całkowitą liczbę próbek w sygnale, które mają wartości zawarte w *pewnym zakresie*. Weźmy na przykład sygnał zmiennoprzecinkowy o wartościach pomiędzy 0,0 i 10,0 i histogram o 1000 przedziałów. Przedział 0 w histogramie wyraża liczbę próbek sygnału mających wartość między 0 i 0,01, przedział 1 – liczbę próbek o wartościach między 0,01 i 0,02 itd. aż do przedziału 999 zawierającej liczbę próbek o wartościach między 9,99 i 10,0. W tablicy 2.4 podano program obliczania histogramu przedziałowego tą metodą.



Rys. 2.7. Przykłady histogramów przedziałowych. Jak pokazano na rys. a), sygnał zastosowany w tym przykładzie ma długość 300 próbek, które są liczbami zmiennoprzecinkowymi zawartymi w przedziale od 1 do 3. Na rys. b), c) przedstawiono histogramy przedziałowe tego sygnału z użyciem, odpowiednio, 601 i 9 przedziałów. Jak widać, duża liczba przedziałów powoduje słabą rozdzielczość wzdłuż osi pionowej, podczas gdy mała liczba przedziałów – słabą rozdzielczość na osi pionowej. Zastosowanie większej liczby przedziałów powoduje poprawę rozdzielczości w obu kierunkach

Tab. 2.4.

```
' OBLICZENIE HISTOGRAMU PRZEDZIAŁOWWEGO

DIM X[25000]                    'Sygnał zmiennoprzecinkowy jest przechowywany od X[0] do X[25000]
                                'a każda próbka ma wartość od 0.0 do 10.0
DIM H%[999]                     'Histogram przedziałowy jest przechowywany od H%[0]do H%[999]

FOR I% = 0 TO 999               'Wyzeruj histogram przedziałowy, aby go użyć jako akumulator
    H%[I%] = 0
NEXT I%

GOSUB XXXX                      'Podprogram "mityczny" ładujący sygnał do X%[ ]

FOR I% = 0 TO 25000             'Oblicz histogram przedziałowy dla 25001 punktów
    BINNUM% = INT( X[I%] * 100 )
    H%[BINNUM%] = H%[BINNUM%] + 1
NEXT I%

END
```

Ile przedziałów powinno się stosować? Konieczne jest pogodzenie sprzecznych wymagań. Z jednej strony, jak pokazano na rys. 2.7, zbyt wiele przedziałów utrudnia estymację *amplitudy pmf*, gdyż w każdym przedziale jest niewiele próbek, więc szum statystyczny jest bardzo duży. Z drugiej jednak strony, zbyt mało przedziałów utrudnia estymację *pmf* w kierunku *poziomym*. A zatem wybór liczby przedziałów powinien być kompromisem między rozdzielczością wzdłuż osi y i rozdzielczością na osi x.

Rozkład normalny

Sygnały powstające w wyniku procesów losowych mają zwykle przebieg funkcji gęstości prawdopodobieństwa o kształcie dzwonowym. Taki przebieg nazywany **rozkładem normalnym** lub **rozkładem Gaussa**, czyli **gaussowskim**. Nazwa pochodzi od nazwiska wielkiego niemieckiego matematyka Carla Friedricha Gaussa (1777–1855). Przyczynę, dla której taki rozkład tak często występuje w przyrodzie, krótko omówimy w powiązaniu z *generacją szumu cyfrowego*. Podstawowy kształt krzywej jest opisany *funkcją wykładniczą typu*:

$$y(x)=e^{-x^2}$$

Tę podstawową zależność można przekształcić w pełną krzywą gaussowską, wprowadzając dobieraną wartość średnią μ i odchylenie standardowe σ. Ponadto wzór musi być znormalizowany, aby powierzchnia całkowita pod krzywą była równa *jedności*. Jest to wymóg dotyczący wszystkich funkcji rozkładów prawdopodobieństwa. W rezultacie uzyskuje się ogólną postać wyrażenia opisującego rozkład normalny. Jest to jeden z najważniejszych wzorów w dziedzinie statystyki i rachunku prawdopodobieństwa:

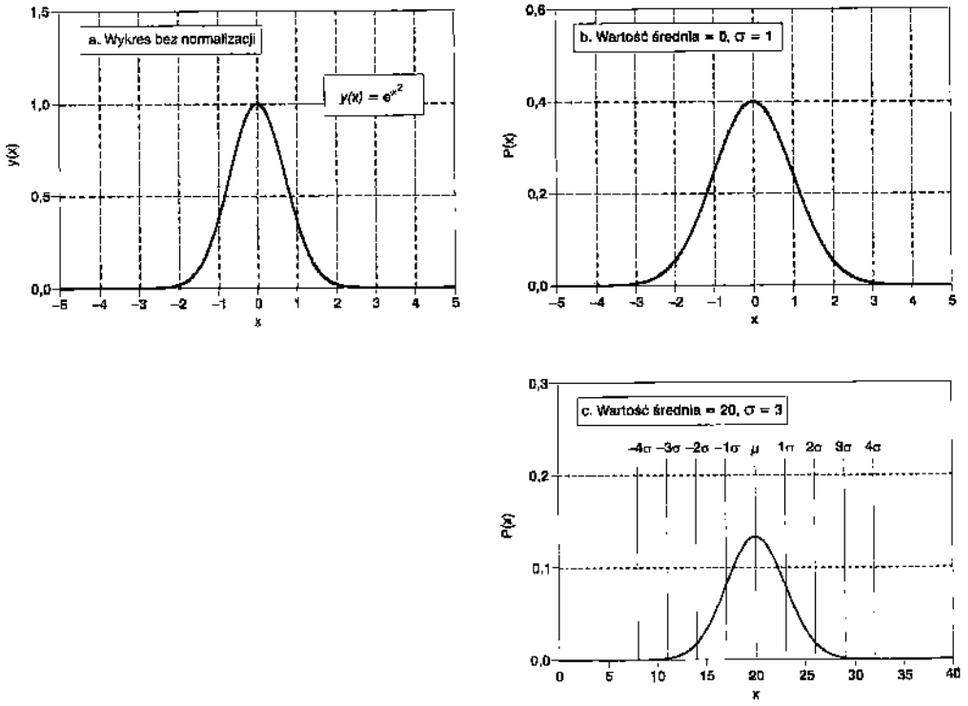
$$P(x)=\frac{1}{\sqrt{2\pi}\,\sigma}\,e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Wzór 2.8. Wzór opisuje rozkład normalny, zwany również rozkładem Gaussa lub po prostu gaussowskim. W tej zależności P(x) jest funkcją rozkładu prawdopodobieństwa, μ wartością średnią i σ odchyleniem standardowym

Na rys. 2.8 przedstawiono kilka przykładów rozkładów Gaussa o różnych wartościach średnich i odchyleniach standardowych. *Wartość średnia* wyznacza punkt środkowy krzywej, *odchylenie standardowe* zaś – szerokość krzywej dzwonowej.

Interesującą właściwością krzywej gaussowskiej jest bardzo gwałtowne opadanie jej zboczy do zera, znacznie szybsze niż w przypadku innych znanych funkcji, jak opadająca funkcja wykładnicza lub 1/x. Na przykład przy odchyleniach od średniej równych dwóm, czterem i sześciu odchyleniom standardowym, wartość krzywej Gaussa spada, odpowiednio, do ok. 1/19, 1/7563 i 1/166 666 666. Dlatego *odnosi się wrażenie*, że takie sygnały o rozkładzie normalnym, jak przedstawiony na rys. 2.6 c), mają tylko przybliżoną wartość międzyszczytową. W sygnałach tego rodzaju mogą, z zasady, pojawiać się wyskoki o nieograniczonej amplitudzie. W praktyce ostre opadanie gaussowskiej krzywej gęstości prawdopodobieństwa (*pdf*) powoduje, że takie skrajności prawie nigdy nie występują. W rezultacie przebieg jest dość ograniczony i ma wartość międzyszczytową ok. 6–8σ.

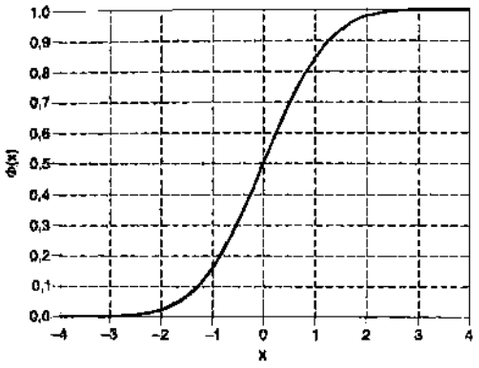
Jak poprzednio wykazano, całkę gęstości prawdopodobieństwa stosuje się do wyznaczania prawdopodobieństwa, że sygnał znajduje się w pewnym zakresie wartości. Dlatego całka gęstości prawdopodobieństwa jest na tyle ważna, że nadano jej specjalną nazwę – **zbiorowa funkcja rozkładu** (*cdf – cumulative distribution function*). Szczególnie przykrą trudnością z funkcją Gaussa jest niemożność jej całkowania metodami podstawowymi. Aby ominąć tę trudność, można całkę gaussowską obliczać przez *całkowanie numeryczne*. To wymaga bardzo dokładnego próbkowania ciągłej krzywej Gaussa, np. kilka milionów punktów próbkowanych między −10σ i +10σ. Próbki sygnału dyskretnego są następnie *sumowane*, aby symulować



Rys. 2.8. Przykłady rozkładów Gaussa: a) krzywa podstawowa – nieznormalizowana i bez wprowadzenia parametrów dobieranych, b), c) pełny rozkład Gaussa dla różnych wartości średnich i odchyleń standardowych

całkowanie. Taka krzywa dyskretna, będąca wynikiem symulowanego całkowania, jest następnie zapisywana w tablicy wykorzystywanej do obliczania wartości prawdopodobieństwa.

Zbiorową funkcję (*cdf*) rozkładu normalnego przedstawiono na rys. 2.9, a jej wartości liczbowe zestawiono w tablicy 2.5. Ta krzywa jest bardzo często używana w rachunku prawdopodobieństwa, dlatego nadano jej własny symbol: Φ(x) (duża grecka litera fi). Na przykład Φ(–2) ma wartość 0,0228. To wskazuje na prawdopodobieństwo 2,28%, że wartość sygnału znajdzie się między



x	Φ(x)
–3,4	0,0003
–3,3	0,0005
–3,2	0,0007
–3,1	0,0010
–3,0	0,0013
–2,9	0,0019
–2,8	0,0026
–2,7	0,0035
–2,6	0,0047
–2,5	0,0062
–2,4	0,0082
–2,3	0,0107
–2,2	0,0139
–2,1	0,0179
–2,0	0,0228
–1,9	0,0287
–1,8	0,0359
–1,7	0,0446
–1,6	0,0548
–1,5	0,0668
–1,4	0,0808
–1,3	0,0968
–1,2	0,1151
–1,1	0,1357
–1,0	0,1587
–0,9	0,1841
–0,8	0,2119
–0,7	0,2420
–0,6	0,2743
–0,5	0,3085
–0,4	0,3446
–0,3	0,3821
–0,2	0,4207
–0,1	0,4602
0,0	0,5000

x	Φ(x)
00	0,5000
0,1	0,5398
0,2	0,5793
0,3	0,6179
0,4	0,6554
0,5	0,6915
0,6	0,7257
0,7	0,7580
0,8	0,7881
0,9	0,8159
1,0	0,8413
1,1	0,8643
1,2	0,8849
1,3	0,9032
1,4	0,9192
1,5	0,9332
1,6	0,9452
1,7	0,9554
1,8	0,9641
1,9	0,9713
2,0	0,9772
2,1	0,9821
2,2	0,9861
2,3	0,9893
2,4	0,9918
2,5	0,9938
2,6	0,9953
2,7	0,9965
2,8	0,9974
2,9	0,9981
3,0	0,9987
3,1	0,9990
3,2	0,9993
3,3	0,9995
3,4	0,9997

Rys. 2.9. i tab. 2.5. Φ(x), zbiorowa funkcja rozkładu dla rozkładu normalnego (wartość średnia = 0, odchylenie standardowe = 1). Wartości rozkładu obliczono przez całkowanie numeryczne rozkładu normalnego przedstawionego na rys. 2.8 b). Φ(x) jest prawdopodobieństwem, że wartość sygnału o rozkładzie normalnym, w jakimś czasie wybranym losowo, będzie mniejsza od x. W tablicy wartość x jest wyrażona w jednostkach odchylenia standardowego, w odniesieniu do wartości średniej

$-\infty$ a wartością o dwa odchylenia standardowe mniejszą od wartości średniej, w każdym losowo wybranym czasie. Podobnie, wartość $\Phi(1) = 0{,}8413$ oznacza, że jest szansa 84,13% na znalezienie się wartości sygnału, w losowo wybranej chwili, między $-\infty$ i wartością o jedno odchylenie standardowe większej od średniej. W celu obliczenia prawdopodobieństwa, że sygnał znajduje się *między* dwiema wartościami, trzeba odjąć od siebie odpowiednie liczby odczytane z tablicy $\Phi(x)$. Na przykład prawdopodobieństwo, że wartość sygnału, w jakimś losowo wybranym czasie, będzie w zakresie między wartością o dwa odchylenia standardowe mniejszą od średniej a wartością o jedno odchylenie większą od średniej, jest dane jako: $\Phi(1) - \Phi(-2) = 0{,}8185$, czyli 81,85%.

W tej metodzie próbki pobrane z sygnału o rozkładzie normalnym będą w 68% czasu mieścić się w granicach $\pm 1\sigma$ wokół średniej. Będą się znajdowały w granicach $\pm 2\sigma$ od średniej przez 95% czasu i w granicach $\pm 3\sigma$ przez ok. 99,75% czasu. Prawdopodobieństwo, że sygnał ma wartość różniącą się od średniej o więcej niż 10 odchyleń standardowych, jest tak małe, że można oczekiwać pojawienia się takiej wartości tylko w ciągu kilku *mikrosekund* od początku Wszechświata, czyli od 10 miliardów lat!

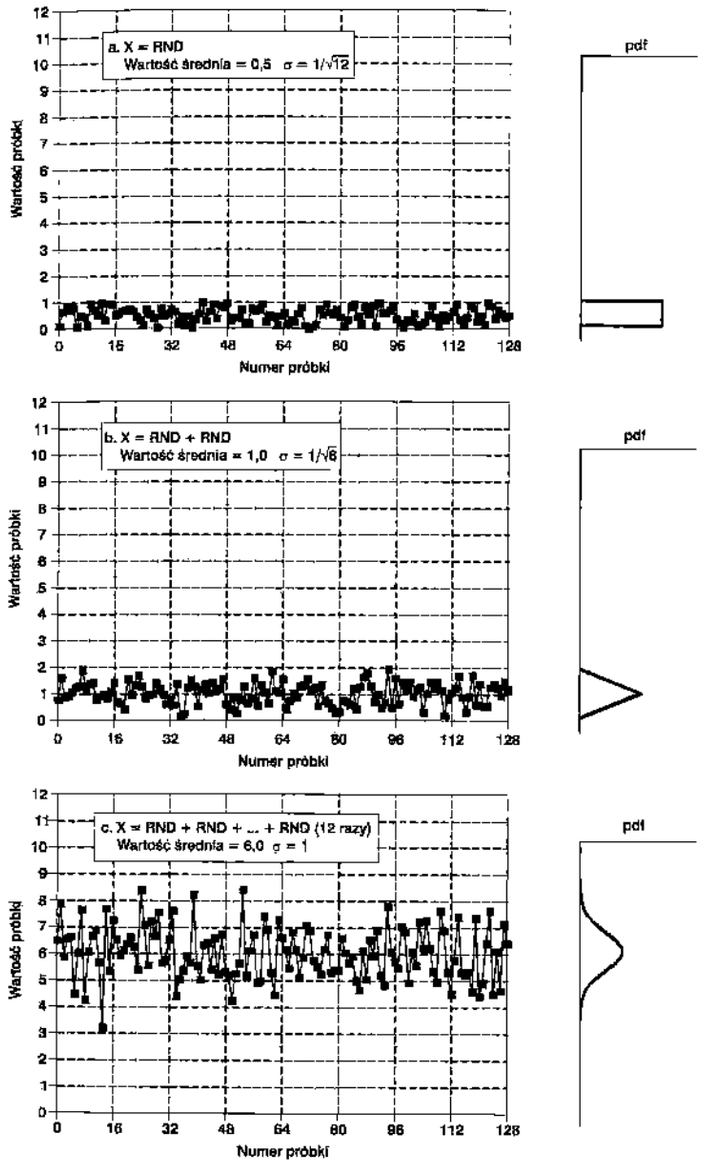
Wzór 2.8 można też wykorzystać do wyrażenia funkcji masy prawdopodobieństwa sygnałów *dyskretnych* o rozkładzie normalnym. W tym przypadku x jest jednym ze skwantyzowanych poziomów przyjmowanych przez sygnał, np. jedną z 4096 wartości binarnych z wyjścia przetwornika analogowo-cyfrowego. Pomińmy czynnik $1/\sqrt{2\pi}\sigma$, który jest stosowany tylko po to, aby całkowita powierzchnia pod krzywą gęstości prawdopodobieństwa (*pdf*) była równa *jedności*. Zamiast tego trzeba włączyć jakiś czynnik niezbędny do uczynienia sumy wszystkich wartość *pmf* równą *jedności*. W większości przypadków osiąga się to przez utworzenie krzywej bez martwienia się o znormalizowanie, sumując wszystkie nieznormalizowane wartości i następnie dzieląc wszystkie wartości przez tę sumę.

Generacja szumu cyfrowego

Szum losowy jest tematem ważnym zarówno w elektronice, jak i w DSP. Na przykład określa on, jak mały sygnał może być mierzony przez przyrząd pomiarowy, na jaką odległość system radiowy może utrzymywać łączność i jaka dawka promieniowania jest wymagana do wytworzenia obrazu rentgenowskiego. W DSP jest często konieczne generowanie sygnałów przypominających różne rodzaje szumu losowego. Jest to potrzebne do testowania algorytmów, które muszą *pracować* w obecności szumu.

Sercem systemu wytwarzającego szum cyfrowy jest **generator liczb losowych**. Większość języków programowania ma taki generator jako funkcję standardową. W BASIC-u instrukcja $X = RND$ ładuje zmienną X za każdym razem z nową liczbą losową. Każda liczba losowa ma wartość pomiędzy 0 i 1, z jednakowym prawdopodobieństwem występowania wszystkich wartości znajdujących się między tymi wartościami skrajnymi. Na rys. 2.10 przedstawiono sygnał utworzony przez pobranie 128 próbek z tego typu generatora liczb losowych. Średnia wartość procesu, który był źródłem tego sygnału, jest równa 0,5, odchylenie standardowe $1/\sqrt{12} = 0{,}29$ a rozkład między 0 i 1 jest równomierny.

Algorytmy powinny być testowane przy użyciu danych tego samego rodzaju, jak przy zwykłej pracy. To wywołuje konieczność generacji szumu cyfrowego o *gaussowskim* rozkładzie gęstości prawdopodobieństwa (*pdf*). Są dwa sposoby wytwarzania takich sygnałów w generatorze liczb losowych. Rys. 2.10 ilustruje pierwszą metodę. Na rys. 2.10 b) przedstawiono sygnał, w którym każdą próbkę uzyskano przez sumowanie dwóch liczb losowych, tj. $X = RND + RND$. Suma może zmieniać się w granicach od 0 do 2, gdyż każda z próbek zmienia się od 0 do 1. Wartość średnia jest teraz równa 1, a odchylenie standardowe $1/\sqrt{6}$ (pamiętajmy, że przy sumowaniu niezależnych sygnałów losowych wariancje też się sumują). Jak pokazano na rysunku, rozkład gęstości prawdopodobieństwa zmienił się z *równomiernego* na *trójkątny*. A więc sygnał dłużej ma wartość znajdującą się w pobliżu *jedności*, a krócej – w pobliżu wartości *zero* lub *dwa*.



Rys. 2.10. Przekształcenie rozkładu równomiernego w gaussowski. Na wykresie a) przedstawiono sygnał, w którym każda próbka jest wytwarzana przez generator liczb losowych. Jak wynika z rozkładu gęstości (pdf) wartości wszystkich próbek są równomiernie rozłożone między wartościami 0 i 1. W sygnale przedstawionym na rys. b) każda próbka jest tworzona przez sumowanie dwóch, a na rys. c) dwunastu wartości z generatora liczb losowych. Rozkład gęstości prawdopodobieństwa na rys. c) jest bardzo bliski gaussowskiemu, o wartości średniej sześć i odchyleniu standardowym równym jeden

Na rys. 2.10 c) przedstawiono, jak w tej metodzie można pójść o krok dalej, tworząc każdą próbkę przez sumowanie 12 liczb losowych. Wartość średnia jest teraz równa *sześć*, a odchylenie standardowe *jeden*. A najważniejsze jest, że funkcja *pdf* stała się wirtualnie *gaussowska*. Tę procedurę można stosować do tworzenia sygnału szumu o rozkładzie normalnym,

o dowolnych wartości średniej i odchyleniu standardowym. Dla każdej próbki sygnału trzeba: (1) zsumować 12 liczb losowych, (2) odjąć sześć, aby średnią uczynić równą zero, (3) pomnożyć przez żądane odchylenie standardowe oraz (4) dodać żądaną wartość średnią.

Matematycznych podstaw tego algorytmu trzeba szukać w **centralnym twierdzeniu granicznym**, jednej z najważniejszych koncepcji w rachunku prawdopodobieństwa. W swej najprostszej postaci centralne twierdzenie graniczne głosi, że rozkład *sumy* liczb losowych staje się tym bardziej zbliżony do normalnego, im więcej liczb losowych jest sumowanych. Centralne twierdzenie graniczne *nie zawiera* wymagania, aby poszczególne liczby losowe pochodziły z jakiegoś szczególnego rozkładu ani nawet żeby pochodziły z *takiego samego* rozkładu. Centralne twierdzenie graniczne przynosi odpowiedź na pytanie, dlaczego w przyrodzie tak często spotyka się sygnały o rozkładzie normalnym. Gdziekolwiek wiele różnych sił losowych oddziałuje wzajemnie na siebie, uzyskuje się w wyniku gaussowski rozkład gęstości prawdopodobieństwa.

W drugiej metodzie generowania liczb losowych o rozkładzie normalnym przywołuje się *dwukrotnie* generator liczb losowych w celu uzyskania R_1 i R_2. Liczbę o rozkładzie normalnym X można znaleźć z wzoru:

$$X = (-2 \ln R_1)^{\frac{1}{2}} \cos(2\pi R_2)$$

Wzór 2.9. Generowanie liczb losowych o rozkładzie normalnym. R_1 i R_2 są liczbami losowymi o rozkładzie równomiernym między zerem i jedynką. Jako wynik uzyskuje się liczbę X o rozkładzie normalnym, o wartości średniej równej zero i odchyleniu standardowym równym jeden. ln jest logarytmem naturalnym (o podstawie e), kosinus jest podany w radianach

Tak jak poprzednio, tą metodą można generować sygnały losowe o rozkładzie normalnym o dowolnej wartości średniej i dowolnym odchyleniu standardowym.

Generatory liczb losowych pracują, zaczynając od pewnej liczby początkowej zwanej **ziarnem** (*seed*), o wartości między 0 i 1. Gdy jest przywoływany generator liczb losowych, liczba początkowa jest poddawana przetworzeniu przez ustalony algorytm, dając w wyniku nową liczbę o wartości między 0 i 1. Ta nowa liczba jest generowaną *liczbą losową* i jest następnie wewnętrznie rejestrowana, aby mogła być użyta jako liczba początkowa przy następnym przywołaniu generatora liczb losowych. Algorytm przetwarzający liczbę początkową (ziarno) w liczbę losową ma często postać

$$R = (aS + b) \text{ modulo } c$$

Wzór 2.10. Często stosowany algorytm do generowania liczb losowych o rozkładzie równomiernym między 0 i 1. W tej metodzie S jest wartością początkową (ziarnem), R jest nową liczbą losową, a, b, c są odpowiednio wybranymi stałymi. Wyraźmy ten wzór słowami: wielkość aS + b trzeba podzielić przez c i reszta z tego dzielenia daje R

W ten sposób generuje się ciągłą sekwencję liczb losowych, z których wszystkie pochodzą od tej samej wartości początkowej. Dzięki temu można uruchamiać ten program wielokrotnie, stosując dokładnie takie same sekwencje liczb losowych. Jeśli chcemy zmienić sekwencję, to w większości programów mamy możliwość **powtórnego zainicjowania pracy** (*reseeding*) generatora liczb losowych z możliwością wyboru wartości początkowej. Często jako ziarno stosuje się *czas* (wskazywany przez zegar systemu), uzyskując w ten sposób nową sekwencję przy każdym uruchomieniu programu.

Z czysto matematycznego punktu widzenia liczby generowane w ten sposób nie mogą być całkowicie losowe, gdyż każda liczba jest w pełni określona przez liczbę *poprzednią*. Do opisu tej sytuacji stosuje się często termin **pseudolosowy**. Nie trzeba się jednak przejmować tą kwestią. Sekwencje liczb generowane przez generatory liczb losowych są w wystarczająco dużym stopniu statystycznie losowe. Jest bardzo mało prawdopodobne spotkanie sytuacji, w której te sekwencje nie są właściwe.

Precyzja i dokładność[3]

Precyzja (*precision*) i dokładność (*accuracy*) to terminy stosowane do opisu systemów i metod służących do *pomiaru*, *estymacji* lub *przewidywania*. W tych wszystkich przypadkach mamy do czynienia z jakimś parametrem, którego wartość chcemy poznać. Nazywamy ją **wartością prawdziwą** lub po prostu **prawdą**. Metoda daje **wartość zmierzoną**. Chcielibyśmy, aby ta wartość była możliwie jak najbliższa wartości prawdziwej. *Precyzja* i *dokładność* są sposobami opisu błędu, który może występować między tymi dwiema wartościami.

Precyzja i dokładność są, niestety, używane wymiennie w sensie nietechnicznym. Rzeczywiście, słowniki, wyjaśniając te słowa, podają odsyłacze od jednej z tych nazw do drugiej! A jednak w nauce i technice każda z tych wielkości ma bardzo dokładną definicję. Starajcie się stosować te określenia prawidłowo i spokojnie tolerować tych, którzy używają tych terminów w niewłaściwy sposób.

Jako przykład rozważmy oceanografa mierzącego głębokość wody przy użyciu *sonaru*. Ze statku są wysyłane krótkie impulsy dźwiękowe, odbijane następnie od dna morskiego i odbierane na powierzchni jako echo. Fale dźwiękowe przemieszczają się w wodzie ze względnie stałą prędkością, dzięki czemu można wyznaczyć głębokość na podstawie czasu, jaki upływa między impulsem wysłanym i odebranym. Jak we wszystkich pomiarach doświadczalnych, występuje pewien błąd między wartościami zmierzonymi i prawdziwymi. Na omawiany pomiar może wpływać wiele czynników, takich jak losowy szum aparatury elektronicznej, fale na powierzchni oceanu, roślinność na dnie oceanu, zmiany temperatury wody powodujące zmiany szybkości dźwięku itd.

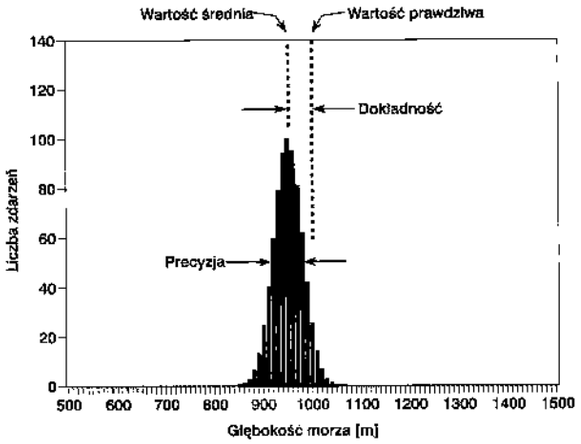
W celu zbadania tych wpływów oceanograf wykonuje szereg kolejnych odczytów pomiarowych w miejscu o znanej głębokości równej *dokładnie* 1000 metrów (wartość prawdziwa). Wyniki pomiarów są zestawiane w histogramie przedstawionym na rys. 2.11. Dane mają rozkład normalny, jak można tego było oczekiwać na podstawie centralnego twierdzenia granicznego. *Wartość średnia* jest pośrodku rozkładu i wyraża najlepszą estymatę na podstawie danych uzyskanych z pomiarów. *Odchylenie standardowe* określa szerokość rozkładu, opisując jak duże odchylenia występują między kolejnymi wynikami pomiarów.

W systemie mogą więc występować dwa ogólne rodzaje błędów. Po pierwsze, wartość średnia może być przesunięta w stosunku do wartości prawdziwej. Wartość tego przesunięcia jest nazywana **dokładnością** pomiaru. Po drugie, wyniki indywidualnych pomiarów mogą nie zgadzać się ze sobą, na co wskazuje szerokość rozkładu. Jest to nazywane **precyzją** pomiaru wyrażaną przez odchylenie standardowe, stosunek sygnału do szumu lub współczynnik rozrzutu CV.

Rozpatrzmy pomiar o dobrej dokładności, lecz słabej precyzji. Histogram jest wtedy wypośrodkowany wokół prawdziwej wartości, lecz bardzo szeroki. Chociaż pomiary są prawidłowe *jako grupa*, to każdy indywidualny wynik jest marną miarą prawdziwej wartości. Mamy tu do czynienia ze słabą *powtarzalnością*; zgodność wyników kolejnych pomiarów nie jest dobra. Słaba precyzja wynika z **błędów losowych**. Taką nazwę nadaje się błędom, które zmieniają się przy każdym powtórzeniu pomiaru. Uśrednianie kilku wyników pomiaru *zawsze* polepsza precyzję. Krótko mówiąc, *precyzja jest miarą szumu losowego*.

Teraz wyobraźmy sobie pomiar o bardzo dobrej precyzji a marnej dokładności. Takie parametry powodują, że histogram jest bardzo „wysmukły", lecz nie jest wypośrodkowany wokół wartości prawdziwej. Wartości kolejnych odczytów są bliskie sobie, jednak *wszystkie* są obarczone dużym błędem. Marna dokładność wynika z **błędów systematycznych**. Są to błędy powtarzające się ściśle w ten sam sposób przy każdym pomiarze. Dokładność zależy zwykle od sposobu *kalibracji* systemu. Na przykład przy pomiarze głębokości oceanu parametrem mierzonym bezpośrednio jest czas. Odcinek czasu jest przetwarzany na głębokość w wyniku procedury kalibracyjnej zamieniającej *milisekundy* na *metry*. Może to być pro-

[3] Patrz przypis od tłumacza zamieszczony w tekście na końcu podrozdziału



Rys. 2.11. Definicje dokładności i precyzji (patrz: przypis tłumacza na końcu tego podrozdziału). Dokładność jest określona różnicą między wartością prawdziwą a średnią wynikającą z procesu generującego dane pomiarowe. Dokładność jest rozrzutem wartości, opisanym odchyleniem standardowym, stosunkiem sygnału od szumu lub współczynnikiem rozrzutu (CV)

sta procedura polegająca na mnożeniu przez stałą szybkość lub tak skomplikowana, jak liczne korekcje drugiego rzędu. Uśrednianie kolejnych pomiarów nie poprawia dokładności. Krótko mówiąc, *dokładność jest miarą kalibracji.*

W praktyce jest wiele sposobów rozróżnienia dokładności od precyzji. Wyobraźmy sobie, na przykład, że skonstruowano wzmacniacz elektroniczny z rezystorami o tolerancji 1%. Taka tolerancja wskazuje, że wartość każdego rezystora mieści się w granicach odchyłki 1% od wartości nominalnej, w szerokim zakresie zmian warunków, takich jak temperatura, wilgotność, efekt starzenia itp. Ten błąd rezystancji wytwarza we wzmacniaczu równoważny błąd wzmocnienia. Czy ten błąd jest związany z dokładnością, czy z precyzją?

Odpowiedź na to pytanie zależy od sposobu, w jaki jest przeprowadzany pomiar. Przypuśćmy na przykład, że zbudowaliśmy *jeden* wzmacniacz i testujemy go kilkakrotnie w ciągu kilku minut. Błąd wzmocnienia pozostaje stały podczas każdego pomiaru i stąd wnioskujemy, że problemem jest *dokładność*. Dla porównania przypuśćmy, że zbudowaliśmy *tysiąc* wzmacniaczy. Wartości wzmocnienia w poszczególnych egzemplarzach fluktuują w sposób losowy i okazuje się, że problemem jest wtedy *precyzja*. Podobnie każdy z tych wzmacniaczy będzie wykazywał fluktuacje wzmocnienia w wyniku zmian temperatury i innych czynników środowiskowych. Tu znowu mamy do czynienia z *precyzją*.

Decydując, jaką nazwą określić problem występujący w pomiarach, trzeba sobie zadać dwa pytania. Pierwsze, czy uśrednianie kolejnych odczytów da lepszy pomiar? Jeśli tak, to błąd można nazwać błędem precyzji, a jeśli nie – to występuje tu błąd dokładności. Drugie pytanie: czy kalibracja skoryguje błąd? Jeśli tak, to mamy błąd dokładności, a jeśli nie – błąd precyzji. Ta metoda może wymagać pewnych przemyśleń, zwłaszcza dotyczących sposobu i częstości kalibracji urządzenia pomiarowego.

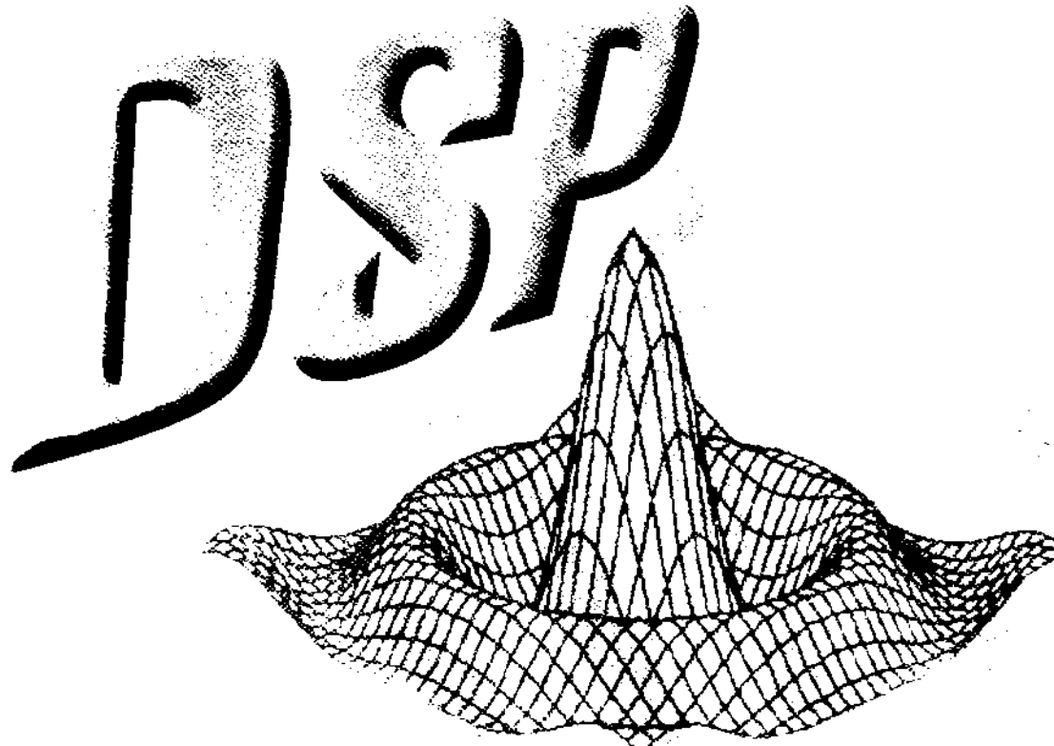
Przypis tłumacza

W polskiej terminologii technicznej nie stosuje się proponowanego przez autora rozróżnienia dokładności i precyzji. Według wiedzy tłumacza nie spotyka się też takiego rozróżnienia w podręcznikach metrologii. Wyróżnia się natomiast błędy systematyczne i błędy przypadkowe (losowe). Do błędów systematycznych zalicza się te, które można w konkretnym przypadku przewidzieć na podstawie znajomości (*a posteriori*) danego procesu pomiarowego. Błędy przypadkowe zaś to błędy, których nie można przewidzieć w kategoriach poszczególnego zdarzenia jednostkowego (pomiaru), lecz o których informacje mogą dać metody statystyczne (definicje według: P. H. Sydenham: *Podręcznik metrologii*, WKŁ Warszawa, 1988).

W tej książce autor wiąże dokładność z błędem systematycznym, a precyzję – z błędem losowym. Według polskiej terminologii można by rozróżnić dokładność wynikającą z błędu statystycznego i dokładność wynikającą z błędu losowego. Warto przypomnieć, że w przypadku pomiaru wielkości przetworzonych na cyfrowe jednym z parametrów wpływających na dokładność pomiaru jest rozdzielczość uwarunkowana ograniczoną liczbą bitów przetwornika a/c.

3

Przetwarzanie analogowo-cyfrowe i cyfrowo-analogowe



Sygnały występujące w badaniach i w technice są w większości sygnałami *ciągłymi*: natężenie światła zmieniające się z odległością, napięcie zmieniające się w funkcji czasu, szybkość reakcji chemicznej zależna od temperatury itd. Przetwarzanie analogowo-cyfrowe (ADC – *Analog-to-Digital Conversion*) i cyfrowo-analogowe (DAC – *Digital-to-Analog Conversion*) to procesy, dzięki którym komputery cyfrowe mogą operować takimi powszechnie spotykanymi sygnałami. Informacja cyfrowa różni się od swego analogowego odpowiednika pod dwoma ważnymi względami: jest *spróbkowana* i *skwantyzowana*. Obie te właściwości decydują o tym, ile informacji może zawierać sygnał cyfrowy. Ten rozdział poświęcono *zarządzaniu informacją*: zrozumieniu, jaką informację należy utrzymać i na utratę jakiej można sobie pozwolić. To z kolei decyduje o wyborze częstotliwości próbkowania, liczby bitów i typu filtracji analogowej – wymaganych do przechodzenia z dziedziny analogowej do cyfrowej i odwrotnie.

Kwantyzacja

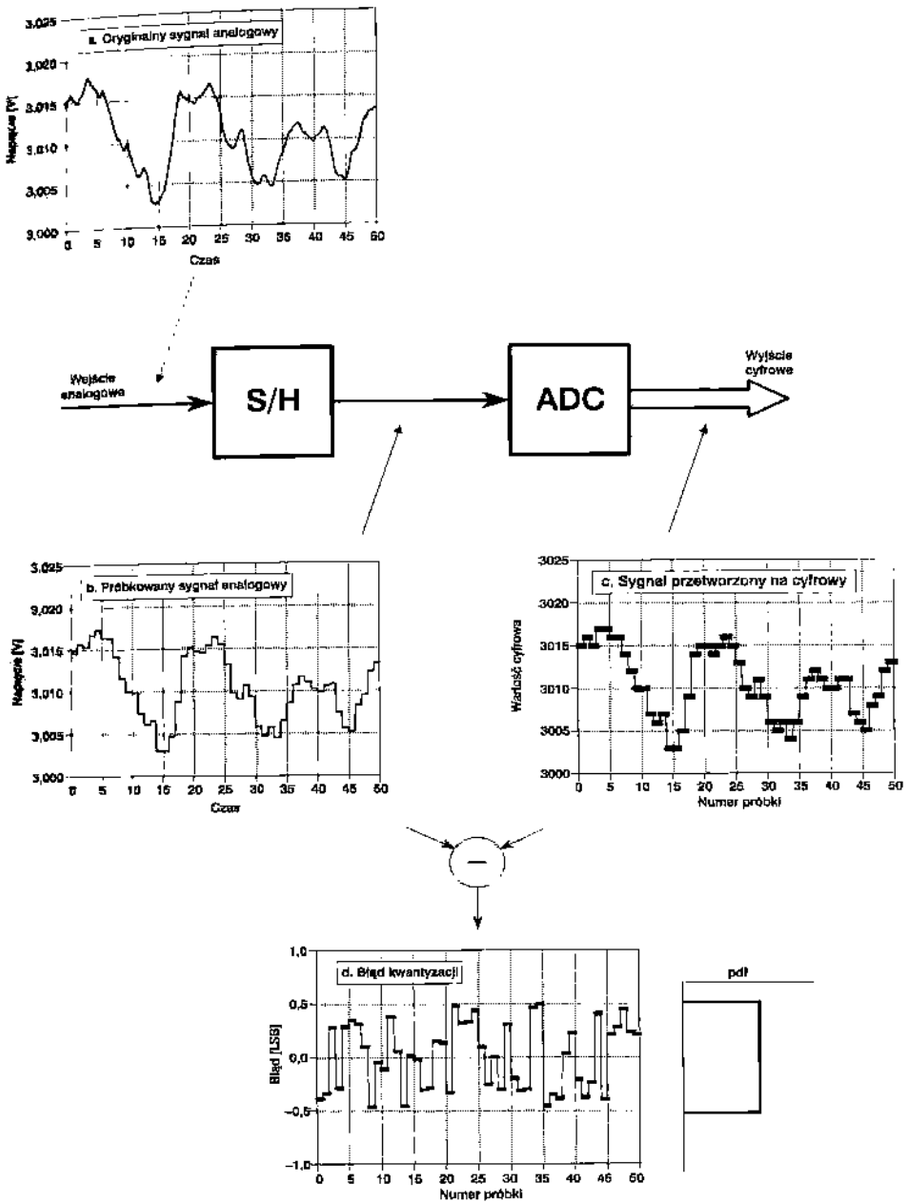
Najpierw trochę spraw oczywistych. Jak wiadomo, w języku angielskim używa się określenia *digital computer*, a nie *digit computer*. Przetwarzana informacja jest nazywana *digital data*, a nie *digit data*. Dlaczego więc przetwarzanie analogowo-cyfrowe jest po angielsku określane słowami *digitize* i *digitization*, a nie *digitalize* i *digitalization*? Odpowiedź jest zaskakująca. Gdy w elektronice zaczęto rozwijać technikę cyfrową, wiele dogodnych nazw było już, sto lat wcześniej, zaanektowanych przez nauki medyczne. Słowa *digitalize* i *digitalization* oznaczają podawanie choremu ziołowego środka pobudzającego pracę serca o nazwie *digitalis* (po polsku naparstnica)[4].

Na rys. 3.1 przedstawiono przebiegi elektryczne w typowym przetworniku analogowo-cyfrowym. Przebieg a) jest sygnałem, który ma być przetworzony. Jak wynika z opisu osi współrzędnych, jest to sygnał *napięciowy* zmieniający się w funkcji *czasu*. Dla uproszczenia załóżmy, że sygnał może się zmieniać od 0 do 4,095 V, co odpowiada liczbom od 0 do 4095 uzyskiwanym na wyjściu 12-bitowego digitalizatora. Proszę zauważyć, że schemat blokowy podzielono na dwie sekcje: układ próbkująco-pamiętający (S/H – *Sample-and-Hold*) i przetwornik analogowo-cyfrowy (ADC – *Analog-to-Digital Converter*). Jak już zapewne wiecie z wykładów elektroniki, układ próbkująco-pamiętający jest niezbędny do utrzymania, podczas przetwarzania, ustalonej wartości sygnału na wejściu przetwornika a/c. Jednak *nie z tego powodu* pokazano na rysunku ten układ – podział digitalizacji na dwa etapy daje ważny model teoretyczny ułatwiający zrozumienie tego procesu. Fakt, że taki podział można przedstawić w postaci dwóch układów elektronicznych, jest po prostu szczęśliwym zbiegiem okoliczności.

Analizując różnice między przebiegami 3.1 a) i b) widzimy, że sygnał na wyjściu układu próbkująco-pamiętającego może się zmieniać tylko w pewnych chwilach powtarzających się okresowo. W tych chwilach ten sygnał staje się identyczny z wartością chwilową sygnału wejściowego. Zmiany sygnału wejściowego pojawiające się w międzyczasie są całkowicie ignorowane. Zatem **próbkowanie** zamienia zmienną niezależną (w tym przypadku jest nią czas) z ciągłej na dyskretną.

Jak wynika z różnic między przebiegami na rys. 3.1 b), c), przetwornik a/c daje liczbę całkowitą o wartości od 0 do 4095 dla każdego płaskiego obszaru przebiegu b). To wprowadza pewien błąd, gdyż każde plateau może wyrażać *dowolną* wartość napięcia z zakresu między 0 i 4,095 V. Na przykład zarówno napięcie 2,56000 V, jak i 2,56001 V będą przetwarzane na liczbę 2560. Tak więc **kwantyzacja** przetwarza *zmienną zależną* (w tym przypadku jest nią napięcie) z ciągłej na dyskretną.

[4] Ta uwaga dotyczy terminologii angielskiej, po polsku używa się określeń: digitalizacja lub dyskretyzacja (przyp. tłum.).



Rys. 3.1. Przebiegi ilustrujące proces digitalizacji. Przetwarzanie rozdzielono na dwa stopnie, aby wpływ próbkowania wyodrębnić od wpływu kwantyzacji. Pierwszym stopniem jest układ próbkująco-pamiętający (S/H), po którym jedyną zachowaną informacją są wartości chwilowe sygnału próbkowanego okresowo. Drugi stopień, przetwornik a/c (ADC) przetwarza napięcie na najbliższą, równoważną mu wartość całkowitą. W rezultacie każda próbka sygnału przetworzonego ma wartość cyfrową obarczoną błędem do ±1/2LSB, jak to przedstawiono na przebiegu d). Dlatego modelem kwantyzacji jest zazwyczaj proste dodanie szumu do sygnału

Proszę zwrócić uwagę, że unikamy porównywania przebiegów 3.1 a) i c), gdyż wtedy kwantyzacja i próbkowanie byłyby traktowane wspólnie. Ważne jest, aby te dwa procesy analizować oddzielnie, ponieważ w różny sposób pogarszają właściwości sygnału i są uzależnione od różnych parametrów układów elektronicznych. Bywa też, że jeden z tych procesów jest stosowany oddzielnie, bez drugiego. Na przykład próbkowanie bez kwantyzacji jest stosowane w filtrach z przełączanymi pojemnościami.

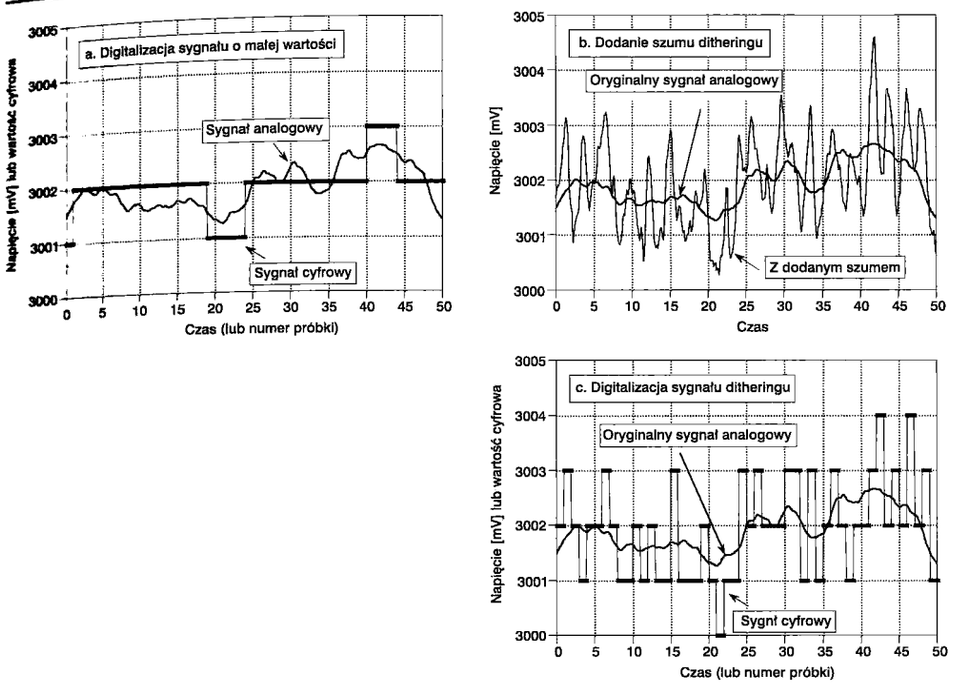
Najpierw rozpatrzymy wpływ kwantyzacji. Każda próbka w sygnale przetworzonym na liczbę może mieć maksymalny błąd ±1/2 **LSB (najmniej znaczącego bitu** – *Least Significant Bit*, odpowiadającemu odstępowi między kolejnymi poziomami kwantyzacji). Błąd kwantyzacji w tym konkretnym przykładzie, uzyskany przez odejmowanie przebiegów 3.1 b) i c) (z odpowiednią konwersją) przedstawiono na rys. 3.1 d). Można więc powiedzieć, że cyfrowy sygnał wyjściowy c) jest równoważny ciągłemu sygnałowi wejściowemu b) *plus* błąd kwantyzacji d). Z tej analizy wynika ważny fakt, że błąd kwantyzacji jest, jak się okazuje, bardzo podobny do *szumu losowego*.

Dzięki temu można utworzyć przydatny model błędu kwantyzacji. W większości przypadków kwantyzacja *powoduje tylko dodanie do sygnału pewnej specyficznej ilości szumu losowego*. Ten szum addytywny jest rozmieszczony równomiernie w zakresie ±LSB, ma wartość średnią równą zero i odchylenie standardowe $1/\sqrt{12}$ LSB ≈ 0,29 LSB. Poddanie sygnału analogowego przetwarzaniu, na przykład w 8-bitowym przetworniku a/c, powoduje powstanie szumu o wartości skutecznej 0,29/256 czyli ok. 1/900 pełnego zakresu. Przetwarzanie 12-bitowe dodaje szum równy 0,29/4096 ≈ 1/14000, a 16-bitowe 0,29/65536 ≈ 1/227000. Ponieważ błąd kwantyzacji jest szumem losowym, więc *liczba bitów* określa precyzję[5] danych. Można powiedzieć na przykład: „Zwiększyliśmy precyzję pomiaru z 8 do 12 bitów".

Ten model jest niezwykle dogodny, gdyż szum losowy wytwarzany w wyniku kwantyzacji po prostu dodaje się do jakiegokolwiek szumu już obecnego w sygnale analogowym. Wyobraźmy sobie np. sygnał o maksymalnej wartości 1,0 V i z szumem losowym 1,0 mV (wartość skuteczna). Wartość 1,0 V w wyniku 8-bitowej digitalizacji staje się liczbą 255, a 1,0 mV odpowiada 0,255 LSB. Jak już stwierdzono w poprzednim rozdziale, szumy losowe sumuje się przez dodawanie ich *wariancji*. Tak więc sumuje się kwadraty: $\sqrt{A^2+B^2}=C$. Całkowity szum w sygnale przetworzonym na wartość cyfrową jest równy $\sqrt{0,255^2+0,29^2}$ = 0,386LSB w stosunku do szumu, który już był w sygnale analogowym. Przetwarzanie 12-bitowe tego samego sygnału nie spowodowałoby praktycznie żadnego wzrostu szumu i nie byłoby *żadnej straty* spowodowanej kwantyzacją. Stając więc przed decyzją o liczbie bitów przetwornika, trzeba sobie zadać dwa pytania: 1) Jaki szum *już* jest w sygnale analogowym?, 2) Jak duży szum jest dopuszczalny w sygnale cyfrowym?
Kiedy taki model kwantyzacji nie jest słuszny? Tylko wtedy, gdy błędu kwantyzacji nie można traktować jako *wartości losowej*. Taka sytuacja powstaje tylko wówczas, gdy wartość sygnału analogowego pozostaje prawie stała w wielu kolejnych próbkach, jak na rys. 3.2 a). Ta sama wartość cyfrowa na wyjściu *pozostaje stała* podczas szeregu kolejnych próbek, nawet jeśli sygnał analogowy zmienia się w granicach ±1/2 LSB.W tym przypadku błąd kwantyzacji nie ma charakteru addytywnego szumu losowego, a wygląda raczej jak wpływ zjawiska progowania lub szumu granulacyjnego (zniekształcenia typu *weird*).

Często spotykaną techniką udoskonalenia procesu digitalizacji takich powolnych przebiegów jest **dithering**[6]. Do sygnału analogowego jest dodawany szum losowy o niewielkiej wartości (rys. 3.2 b). W podanym przykładzie szum losowy ma rozkład normalny o odchyleniu standardowym 2/3 LSB, co daje wartość międzyszczytową ok. 3 LSB. Na rys. 3.2 c) pokazano, jak dodawanie tego szumu ditheringu wpływa na sygnał przetworzony na wartość cyfrową. Nawet gdy sygnał analogowy zmienia się o wartość mniejszą niż ±1/2LSB, to szum dodany powoduje, że sygnał wyjściowy losowo zmienia swój stan między sąsiednimi poziomami.

[5] Autor precyzją nazywa dokładność związaną z błędami losowymi (przyp. tłum.).
[6] Dosłownie: dygotanie. W polskiej literaturze technicznej używa się oryginalnej nazwy angielskiej.



Rys. 3.2. Wyjaśnienie techniki ditheringu. Na rys. a) pokazano, jak sygnał analogowy zmieniający się o mniej niż ±1/2 LSB utrzymuje się na tym samym poziomie kwantyzacji podczas procesu digitalizacji. Dithering poprawia tę sytuację przez dodanie szumu losowego o niewielkiej wartości, jak to przedstawiono na rys. b). W tym przykładzie szum ma rozkład normalny o odchyleniu standardowym 2/3 LSB. Dodany szum powoduje zmiany stanu sygnału cyfrowego między sąsiednimi poziomami kwantyzacji (rys. c), co daje więcej informacji o sygnale oryginalnym

Aby zrozumieć korzyść wynikającą z tej techniki, wyobraźmy sobie, że sygnał wejściowy jest napięciem stałym równym 3,0001 V, co odpowiada wartości cyfrowej znajdującej się na 1/10 przedziału między poziomami 3000 i 3001. Bez ditheringu pobranie 10 000 próbek tego sygnału dałoby 10 000 identycznych liczb, wszystkie o wartości 3000. Powtórzmy ten eksperyment myślowy, dodając niewielki szum ditheringu. Wartości 10 000 próbek będą teraz oscylowały między dwoma (lub więcej) poziomami, przy czym ok. 90% próbek będzie miało wartość 3000, a 10% wartość 3001. Obliczając średnią 10 000 wyników, uzyskuje się wartość bliską 3000,1. Chociaż pojedynczy pomiar ma inherentne ograniczenie ±1/2LSB, to statystyka dużej liczby próbek daje znaczna poprawę. Mamy więc dziwną sytuację: *dodanie szumu dostarcza więcej informacji*.

Układy ditheringu mogą być dość wysublimowane, np. z wykorzystaniem komputera do generacji liczb losowych przekazywanych następnie do przetwornika a/c wytwarzającego dodawany szum. Po digitalizacji komputer może *odejmować* te liczby od sygnału cyfrowego z zastosowaniem arytmetyki zmiennoprzecinkowej. Ta elegancka technika nazywa się **ditheringiem subtraktywnym**, lecz jest stosowana tylko w systemach najbardziej rozbudowanych. Najprostszą, choć nie zawsze możliwą metodą jest wykorzystanie do ditheringu szumu już obecnego w sygnale analogowym.

Twierdzenie o próbkowaniu

Definicja *prawidłowego próbkowania* jest bardzo prosta. Załóżmy, że w jakiś sposób próbkujemy sygnał ciągły. Jeśli z pobranych próbek możemy dokładnie *odtworzyć* sygnał analogowy, to znaczy, że próbkowanie było *prawidłowe*. Nawet jeśli spróbkowane dane wydają się być mylące lub niekompletne, to kluczowa informacja została jednak uchwycona, jeśli odwrócenie procesu jest możliwe.

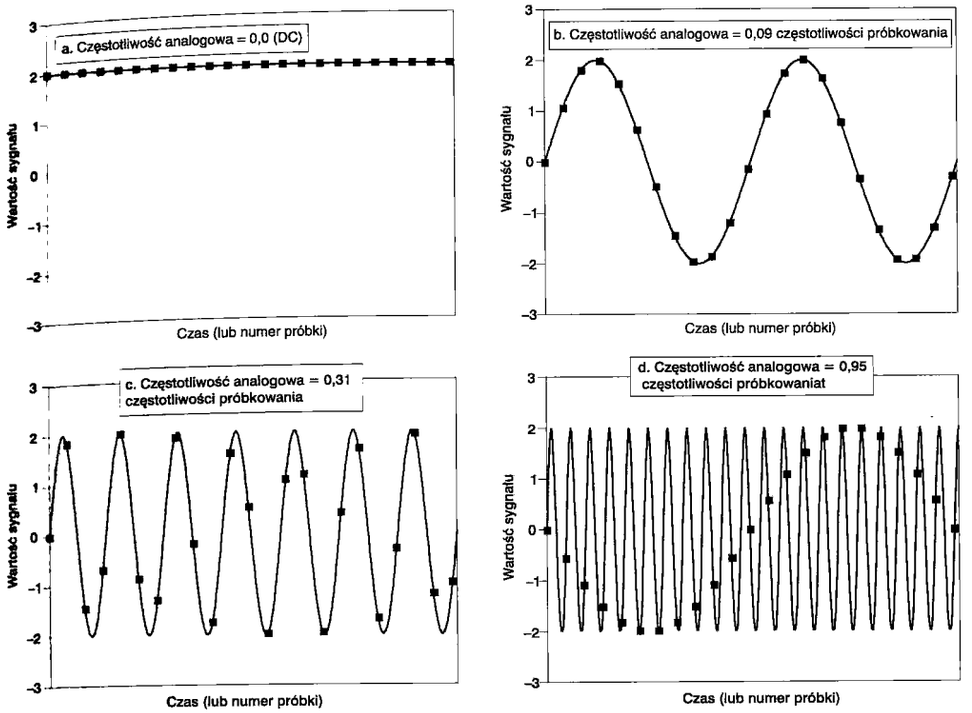
Na rys. 3.3 przedstawiono kilka przebiegów sinusoidalnych przed i po digitalizacji. Linia ciągła obrazuje sygnał analogowy wprowadzany do przetwornika a/c, a kwadratowe znaczniki – sygnał cyfrowy z tego przetwornika. Na rys. 3.3 a) sygnał analogowy jest napięciem stałym (DC), a więc jest kosinusoidą o częstotliwości *zero*. Ponieważ ten sygnał analogowy jest szeregiem linii prostych między poszczególnymi próbkami, więc w danych cyfrowych jest zawarta pełna informacja niezbędna do odtworzenia sygnału analogowego. Zgodnie z naszą definicją, jest to *próbkowanie prawidłowe*.

Przebieg sinusoidalny przedstawiony na rys. 3.3 b) ma częstotliwość równą 0,09 częstotliwości próbkowania. Może to być np. sinusoida o częstotliwości 90 Hz próbkowana z częstotliwością 1000 próbek/s. Wyrażając to inaczej, mamy 11,1 próbek pobieranych w każdym pełnym okresie sinusoidy. Jest to sytuacja bardziej skomplikowana niż w poprzednim przypadku, ponieważ nie da się zrekonstruować sygnału analogowego, prowadząc linie proste między punktami. Czy zebrane próbki prawidłowo reprezentują przebieg analogowy? Odpowiedź jest pozytywna, gdyż żadna inna sinusoida ani kombinacja sinusoid nie wytworzą takiego właśnie rozkładu próbek (z uwzględnieniem rozsądnych ograniczeń dalej wyszczególnionych). Te próbki odpowiadają tylko jednemu sygnałowi analogowemu, a zatem ten sygnał może być dokładnie odtworzony. Znowu mamy tu więc przykład *prawidłowego próbkowania*.

Na rys. 3.3 c) sytuacja jest trudniejsza, gdyż częstotliwość fali sinusoidalnej została zwiększona do 0,31 częstotliwości próbkowania. Wtedy uzyskuje się tylko 3,2 próbki na jeden okres sinusoidy. Tutaj próbki są tak od siebie odległe, że – jak się wydaje – nie reprezentują ogólnego charakteru sygnału analogowego. Czy te próbki w sposób właściwy reprezentują przebieg analogowy? Odpowiedź jest znowu pozytywna, i to z tego samego powodu, co poprzednio. Próbki są unikatową reprezentacją tego sygnału analogowego. W danych cyfrowych jest zawarta pełna informacja konieczna do odtworzenia przebiegu ciągłego. W dalszej części rozdziału będzie wyjaśnione, jak zrealizować taką rekonstrukcję sygnału. Oczywiście musi to być sposób bardziej wyszukany niż zwykłe prowadzenie linii prostych między punktami danych. Choć to może wydać się dziwne, to – zgodnie z naszą definicją – mamy tu *próbkowanie prawidłowe*.

W przebiegu na rys. 3.3 d) częstotliwość przebiegu analogowego zwiększono jeszcze bardziej – do wartości 0,95 częstotliwości próbkowania, uzyskując zaledwie 1,05 próbki na okres przebiegu sinusoidalnego. Czy te próbki prawidłowo reprezentują przebieg analogowy? *Nie, nie reprezentują.* Próbki reprezentują falę sinusoidalną *inną* niż występująca w sygnale wejściowym. Ściślej mówiąc, oryginalna fala sinusoidalna o częstotliwości 0,95 częstotliwości próbkowania jest w sygnale cyfrowym fałszywie przedstawiana jako fala sinusoidalna o częstotliwości równej 0,05 częstotliwości próbkowania. To zjawisko zmiany częstotliwości podczas próbkowania nazywamy **aliasingiem**. Tak jak kryminalista może używać przybranego nazwiska lub tożsamości (*alias*), sinusoida przyjmuje inną, nie swoją częstotliwość. Jednoznaczne odtworzenie sygnału analogowego nie jest możliwe, gdyż w tym przypadku dane cyfrowe nie są w sposób niepowtarzalny związane z jakimś szczególnym przebiegiem analogowym. W spróbkowanych danych nie ma żadnej informacji sugerującej, że oryginalny sygnał analogowy miał częstotliwość 0,95 a nie 0,05 częstotliwości próbkowania. Przebieg sinusoidalny całkowicie ukrył swoją prawdziwą tożsamość – popełniono zbrodnię doskonałą! Według naszej definicji mamy tu *próbkowanie nieprawidłowe*.

Ten tok rozumowania prowadzi do **twierdzenia o próbkowaniu**, będącego kamieniem milowym w dziedzinie DSP. Jest ono często nazywane twierdzeniem *Shannona* lub twierdzeniem *Nyquista*, autorów publikacji na ten temat w latach 40. ubiegłego wieku. Zgodnie z twierdzeniem o próbkowaniu, że sygnał ciągły może być prawidłowo spróbkowany, *tylko jeśli nie zawiera składowych o częstotliwościach większych niż połowa częstotliwości próbkowania*. Na przykład próbkowanie z częstotliwością 2000 próbek/s wymaga, aby sygnał analogowy zawierał tylko częstotliwości poniżej 1000 okresów/s. Jeśli w sygnale *występują* składowe o częstotliwościach większych od tej granicy, to będą one przypisane do zakresu częstotliwości między



Rys. 3.3. Przykłady próbkowania prawidłowego i nieprawidłowego. Sygnał ciągły jest spróbkowany prawidłowo, jeśli próbki zawierają pełną informację niezbędną do odtworzenia przebiegu oryginalnego. Na rysunkach a) ,b) ,c) przedstawiono próbkowanie prawidłowe trzech fal sinusoidalnych. Prawidłowość próbkowania z pewnością nie jest oczywista, gdyż wydaje się, że próbki w przebiegu (c) nawet nie wychwytują kształtu przebiegu. Niemniej każdy z tych sygnałów ciągłych stanowi unikatową, równoważną sobie wzajemnie, parę ze swym rozkładem próbek. To gwarantuje możliwość odtworzenia. Na rys. d) częstotliwość przebiegu sinusoidalnego jest większa od częstotliwości Nyquista (połowy częstotliwości próbkowania). To powoduje powstawanie aliasingu, a więc częstotliwość spróbkowanych danych jest różna od częstotliwości sygnału ciągłego. Ponieważ aliasing wywołuje zniekształcenie sygnału, więc sygnał oryginalny nie może zostać odtworzony na podstawie próbek

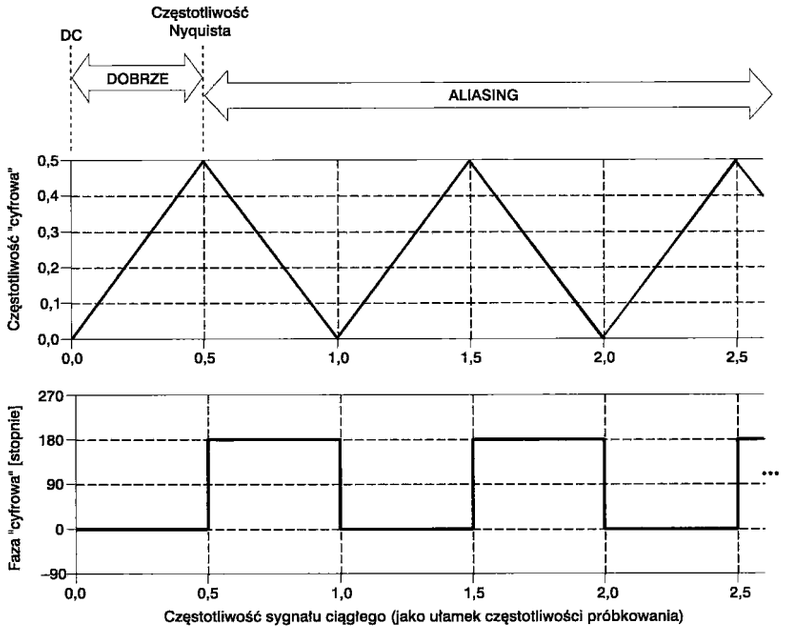
0 i 1000 Hz, dołączając się do informacji, która w sposób uprawniony jest zawarta w tym obszarze częstotliwości.

W związku z twierdzeniem o próbkowaniu szeroko używa się dwóch pojęć: **częstotliwość Nyquista** (*Nyquist frequency*) i **szybkość Nyquista** (*Nyquist rate*)[7]. Niestety ich znaczenia nie są znormalizowane. W celu zrozumienia tej kwestii rozważmy sygnał analogowy zawierający częstotliwości między DC (0) i 3 kHz. Prawidłowa digitalizacja takiego sygnału wymaga próbkowania z częstotliwością 6000 próbek/s (6 kHz) lub większą. Przyjmijmy, że wybieramy próbkowanie 8000 próbek/s (8 kHz), dające prawidłową reprezentację sygnału o częstotliwościach od 0 do 4 kHz. Mamy tu do czynienia z czterema ważnymi częstotliwościami: 1) największa częstotliwość w sygnale 3 kHz, 2) podwojenie tej częstotliwości 6 kHz, 3) szybkość próbkowania 8 kHz, 4) połowa szybkości próbkowania 4 kHz. Która z tych wielkości jest częstotliwością Nyquista, a która szybkością Nyquista? Zależy, kogo spytamy! Stosowane są wszystkie możliwe kombinacje. Na szczęście większość autorów

[7] W polskiej terminologii używa się raczej pojęcia „częstotliwość próbkowania"(przyp. tłum.).

definiuje sposób stosowania tych wielkości. W tej książce stosujemy pojęcie częstotliwości Nyquista oznaczającej *połowę częstotliwości próbkowania.*

Na rys. 3.4 przedstawiono, jak częstotliwości zmieniają się w wyniku aliasingu. Przede wszystkim trzeba pamiętać, że *nie jest możliwe*, aby sygnał cyfrowy zawierał częstotliwości większe od połowy częstotliwości próbkowania (tj. od częstotliwości Nyquista). Gdy częstotliwości w przebiegu ciągłym są poniżej częstotliwości Nyquista, to te same częstotliwości są w sygnale spróbkowanym. Jeśli jednak częstotliwość sygnału ciągłego jest powyżej częstotliwości Nyquista, to zjawisko aliasingu *zmienia* częstotliwość na taką, która może być reprezentowana w spróbkowanych danych. Jak pokazano za pomocą linii łamanej na rys. 3.4, każda częstotliwość w przebiegu ciągłym powyżej częstotliwości Nyquista ma odpowiadającą jej częstotliwość leżącą pomiędzy zerem a połową częstotliwości próbkowania. Jeśli zdarzy się, że w tym dolnym obszarze częstotliwości jest już jakiś przebieg sinusoidalny, to sygnał poddany aliasingowi doda się do tamtego przebiegu, powodując stratę informacji. Aliasing jest podwójnie szkodliwy: może być utracona informacja zarówno o większych, *jak i* małych częstotliwościach. Załóżmy, że mamy sygnał cyfrowy zawierający częstotliwość równą 0,2 częstotliwości próbkowania. Jeśli ten sygnał był uzyskany w wyniku prawidłowego próbkowania, to oryginalny sygnał analogowy *musiał* mieć częstotliwość 0,2 częstotliwości próbkowania. Jeśli podczas próbkowania nastąpił aliasing, to częstotliwość przebiegu cyfrowego 0,2 mogła powstać z każdej z nieskończonej liczby częstotliwości sygnału analogowego: 0,2; 0,8; 1,2; 1,8; 2,2, ...



Rys. 3.4. Konwersja częstotliwości „analogowej" na częstotliwość „cyfrową" podczas próbkowania. Sygnały ciągłe o częstotliwości mniejszej od połowy częstotliwości próbkowania są bezpośrednio przetwarzane na sygnały o właściwej częstotliwości „cyfrowej". Powyżej połowy częstotliwości próbkowania następuje aliasing, w wyniku którego częstotliwość jest błędnie interpretowana w sygnale cyfrowym. Aliasing zawsze zamienia większe częstotliwości w mniejsze, w zakresie od 0 do 0,5. Ponadto aliasing może przesunąć fazę sygnału o 180°

Aliasing podczas próbkowania może, podobnie jak częstotliwość, zmienić także *fazę*. Na przykład spójrzmy znowu na sygnał z aliasingiem na rys. 3.3 d). Sygnał z aliasingiem jest *odwrócony* w stosunku do oryginalnego sygnału analogowego; jeden jest sinusoidą, a drugi jej odwróceniem. Tak więc aliasing spowodował zmianę częstotliwości *oraz* przesunięcie fazy o 180°. Są możliwe tylko dwa przesunięcia fazy: 0° (bez przesunięcia) oraz 180° (odwrócenie). Zerowe przesunięcie fazy zachodzi dla częstotliwości przebiegu analogowego od 0 do 0,5; od 1,0 do 1,5; od 2,0 do 2,5 itd. Odwrócenie fazy następuje dla częstotliwości przebiegu analogowego: od 0,5 do 1,0; od 1,5 do 2,0; od 2,5 do 3,0 itd.

Teraz zagłębimy się w bardziej szczegółową analizę próbkowania i powstawania aliasingu. Naszym głównym celem jest zrozumieniem, co dzieje się z informacją, gdy sygnał podlega przetwarzaniu z postaci ciągłej w dyskretną. Problem polega na tym, że mamy do czynienia z dwiema bardzo różnymi rzeczami – *przebiegiem ciągłym* i *matrycą liczb*. Takie porównanie jak „jabłek z pomarańczami", czyni analizę bardzo trudną. Rozwiązaniem jest wprowadzenie teoretycznej koncepcji zwanej **ciągiem impulsów** (*impulse train*).

Na rys. 3.5 a) przedstawiono pewien przykład sygnału analogowego. Na rys. 3.5 c) jest pokazany sygnał spróbkowany przy użyciu *ciągu impulsów*. Ciąg impulsów jest sygnałem ciągłym złożonym z ciągu wąskich szpilek (impulsów), które w chwilach próbkowania są identyczne z sygnałem oryginalnym. Każdy impuls jest nieskończenie wąski; tę koncepcję omówimy w rozdziale 13. Między próbkowaniami wartość przebiegu jest zerowa. Pamiętajmy, że ciąg impulsów jest pojęciem *teoretycznym*, a nie przebiegiem, który może istnieć w układach elektronicznych. Ponieważ zarówno przebieg oryginalny, jak i ciąg impulsów są przebiegami ciągłymi, więc między nimi możemy już robić porównanie jak „jabłek z jabłkami".

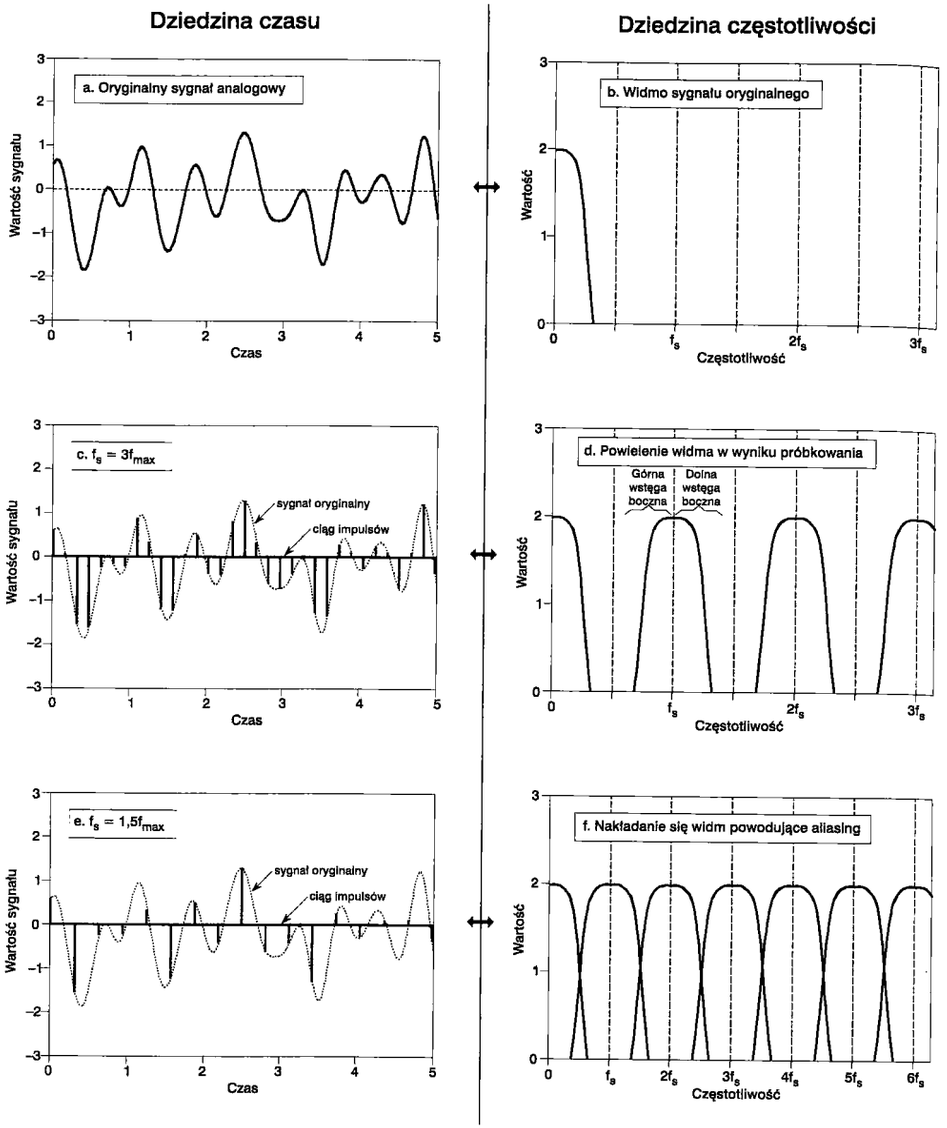
Teraz powinniśmy rozpatrzyć zależność między ciągiem impulsów i sygnałem dyskretnym (matryca liczb). Jest to łatwe; pod względem *zawartości informacji* te sygnały są *identyczne*. Jeśli jeden z tych sygnałów jest znany, to w łatwy sposób można obliczyć drugi. Można o nich pomyśleć jako o dwóch końcach mostu przerzuconego między światem analogowym i cyfrowym. Stąd wynika, że osiągniemy nasz główny cel, jeśli zrozumiemy konsekwencje zamiany przebiegu z rys. 3.5 a) na przebieg z rys. 3.5 c).

Trzy przebiegi ciągłe przedstawiono w lewej kolumnie rys. 3.5. Odpowiadające im *widma częstotliwościowe* pokazano w kolumnie prawej. Takie podejście jest znane w elektronice: każdy przebieg można przedstawić jako złożony z sinusoid o różnych amplitudach i częstotliwościach. Dziedzinę częstotliwości omówimy szczegółowo w dalszych rozdziałach. (Być może będziecie chcieli wrócić do tych rozważań po lepszym zapoznaniu się z widmami częstotliwościowymi).

Na rys. 3.5 a) przedstawiono sygnał analogowy, który chcemy próbkować. Jak wynika z widma częstotliwościowego tego sygnału (rys. 3.5 b), zawiera on tylko składowe częstotliwości z zakresu między 0 i $0{,}33f_s$, gdzie f_s jest częstotliwością próbkowania, jaką zamierzamy zastosować. Może to być np. sygnał mowy odfiltrowany w celu usunięcia wszystkich częstotliwości powyżej 3,3 kHz. Odpowiednio, f_s powinna być 10 kHz (10 000 próbek/s), to jest nasza zamierzona częstotliwość próbkowania.

Próbkowanie sygnału z rys. 3.5 a) ciągiem impulsów daje sygnał przedstawiony na rys. 3.5 c) o widmie częstotliwościowym pokazanym na rys. 3.5 d). To widmo jest *powielonym* widmem sygnału oryginalnego. Przy każdej wielokrotności częstotliwości próbkowania f_s, $2f_s$, $3f_s$, $4f_s$, itd. powstaje *kopia* widma, a po lewej stronie – *odwrócona kopia widma*. Kopia nazywa się **górną wstęgą boczną**, a kopia odwrócona – **dolną wstęgą boczną**. Próbkowanie spowodowało powstanie *nowych* częstotliwości. Czy jest to próbkowanie prawidłowe? Tak, ponieważ sygnał z rys. 3.5 c) może być z powrotem przetworzony na sygnał z rys. 3.5 a) przez usunięcie wszystkich częstotliwości większych od $0{,}5f_s$. A zatem filtr dolnoprzepustowy przetworzy ciąg impulsów (rys. 3.5 c`) z powrotem w sygnał analogowy (rys. 3.5 a).

Jeśli znacie już podstawy DSP, to podamy teraz bardziej techniczne wyjaśnienie przyczyn powielania widma. (Pomińcie ten fragment, jeśli jesteście nowicjuszami w dziedzinie DSP).



Rys. 3.5. Twierdzenie o próbkowaniu w dziedzinach czasu i częstotliwości. Na rys. a) i b) przedstawiono sygnał analogowy zawierający składowe o częstotliwościach między zerem i 0,33 częstotliwości próbkowania f_s. Na rys. c) sygnał analogowy został spróbkowany przez przetworzenie go na ciąg impulsów. W dziedzinie częstotliwości (rys. 3.5 d) powoduje to powielenie widma na nieskończenie dużą liczbę dolnych i górnych wstęg bocznych. Ponieważ oryginalne częstotliwości przebiegu z rys. b) są zachowane bez zniekształceń na rys. d), więc nastąpiło próbkowanie prawidłowe. Sygnał analogowy z rys. e) jest próbkowany z mniejszą częstotliwością, dlatego zawiera składowe od zera do $0{,}66f_s$ i następuje przekroczenie częstotliwości Nyquista. W rezultacie powstaje aliasing, na co wskazuje zachodzenie wstęg bocznych na siebie. Na rysunkach przez f_{max} oznaczono maksymalną częstotliwość w sygnale oryginalnym

W dziedzinie czasu próbkowanie jest realizowane przez mnożenie sygnału oryginalnego przez ciąg impulsów szpilkowych o *amplitudzie jednostkowej*. Widmo częstotliwościowe takiego ciągu jest też ciągiem impulsów o amplitudzie jednostkowej ze szpilkami znajdującymi się w miejscach wielokrotności częstotliwości próbkowania f_s, $2f_s$, $3f_s$, $4f_s$, itd. Przy mnożeniu dwóch sygnałów w dziedzinie czasu ich widma częstotliwościowe są poddawane operacji splotu. W wyniku tej operacji widmo oryginalne zostaje powielone w miejscu każdej szpilki w widmie ciągu impulsów. Traktując sygnał oryginalny jako zawierający zarówno częstotliwości dodatnie, jak i ujemne, widzimy przyczynę powstawania wstęg bocznych górnej i dolnej. Jest to taka sama sytuacja, jak w przypadku modulacji amplitudy omówionej w rozdziale 10.

Na rys. 3.5 e) podano przykład *próbkowania nieprawidłowego*, spowodowanego zbyt małą częstotliwością próbkowania. Sygnał analogowy zawiera, jak poprzednio, częstotliwości do 3,3 kHz, lecz częstotliwość próbkowania zmniejszono do 5 kHz. Zauważmy, że częstotliwości f_s, $2f_s$, $3f_s$, ... na osi poziomej są rozmieszone gęściej na rys. 3.5 f) niż na rys. 3.5 d). Powstająca trudność jest widoczna na rys. 3.5 f) – powielane części widma zachodzą na pasmo między zerem i połową częstotliwości próbkowania. Chociaż na rys. 3.5 f) te zachodzące na siebie częstotliwości przedstawiono jako zachowujące swoją odrębną tożsamość, to w praktyce następuje sumowanie powodujące całkowite pogmatwanie. Ponieważ nie ma sposobu oddzielenia tych zachodzących na siebie częstotliwości, więc informacja jest tracona i nie można odtworzyć widma oryginalnego. Takie nakładanie się częstotliwości następuje wówczas, gdy sygnał analogowy zawiera składowe o częstotliwościach większych od połowy częstotliwości próbkowania. Tak więc udowodniliśmy słuszność twierdzenia o próbkowaniu.

Przetwarzanie cyfrowo-analogowe

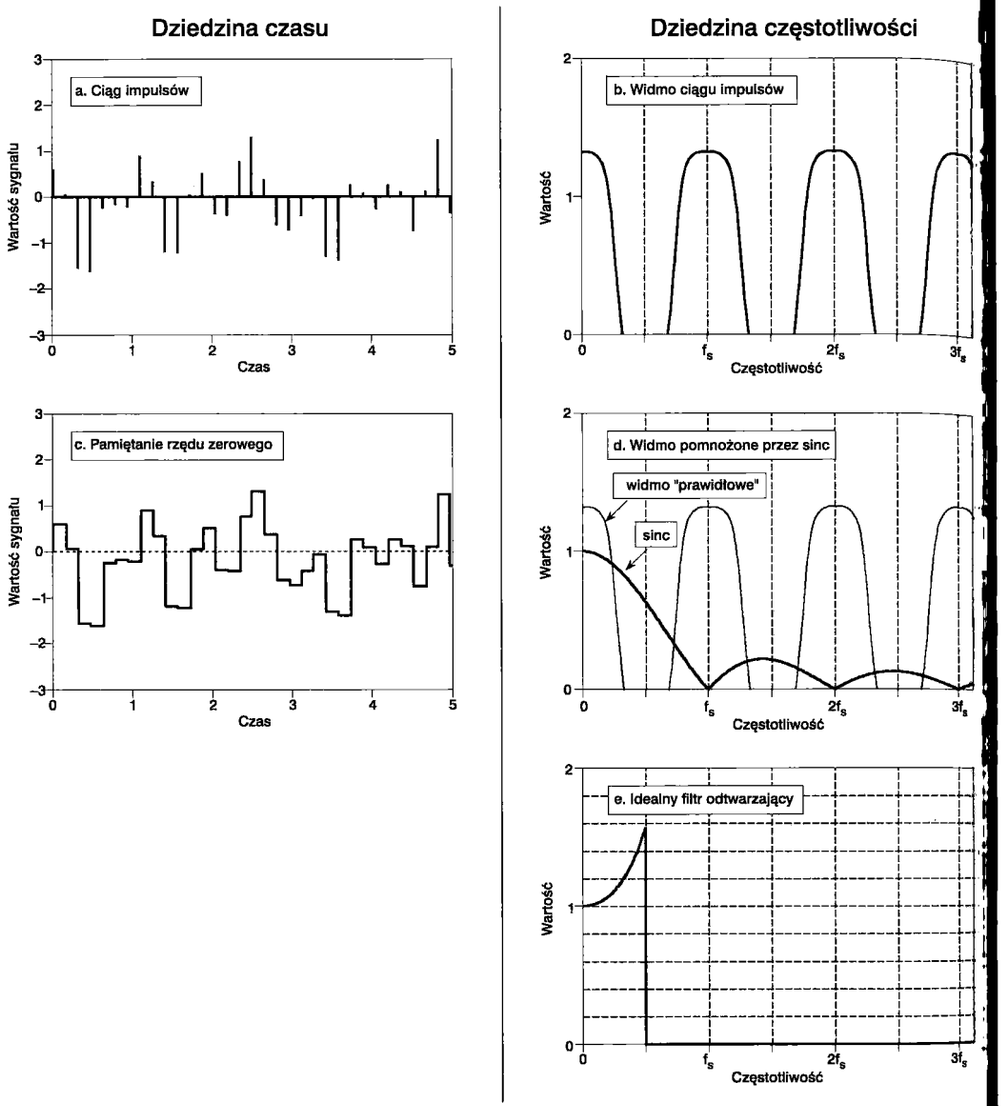
Teoretycznie najprostszą metodą przetwarzania cyfrowo-analogowego jest pobranie próbek z pamięci i przetworzenie ich na *ciąg impulsów*, jak to przedstawiono na rys. 3.6 a). Odpowiednie widmo częstotliwościowe pokazano na rys. 3.6 b). Jak już powiedziano, można doskonale odtworzyć oryginalny sygnał analogowy, przepuszczając ten ciąg impulsów przez filtr dolnoprzepustowy o częstotliwości odcięcia równej połowie częstotliwości próbkowania. Zatem sygnał oryginalny i ciąg impulsów mają identyczne widma częstotliwościowe poniżej częstotliwości Nyquista (połowy częstotliwości próbkowania). Przy większych częstotliwościach w ciągu impulsów jest zawarte powielenie tej informacji, podczas gdy sygnał oryginalny niczego takiego nie zawiera (zakładając, że nie nastąpił aliasing).

Ta metoda jest pod względem matematycznym doskonała, ale w praktyce trudno jest w układach elektronicznych wygenerować żądane wąskie impulsy. Aby ominąć tę trudność, w prawie wszystkich przetwornikach c/a ostatnia wartość próbki jest utrzymywana aż do czasu uzyskania następnej próbki. Jest to **pamiętanie rzędu zerowego** będące w dziedzinie przetwarzania c/a odpowiednikiem układu próbkująco-pamiętającego w przetworniku a/c. (W pamiętaniu zerowego rzędu mamy linie proste między punktami, w drugiego rzędu stosuje się parabole itd.). Pamiętanie zerowego rzędu daje przebieg o charakterze schodkowym, przedstawiony na rys. 3.6 c).

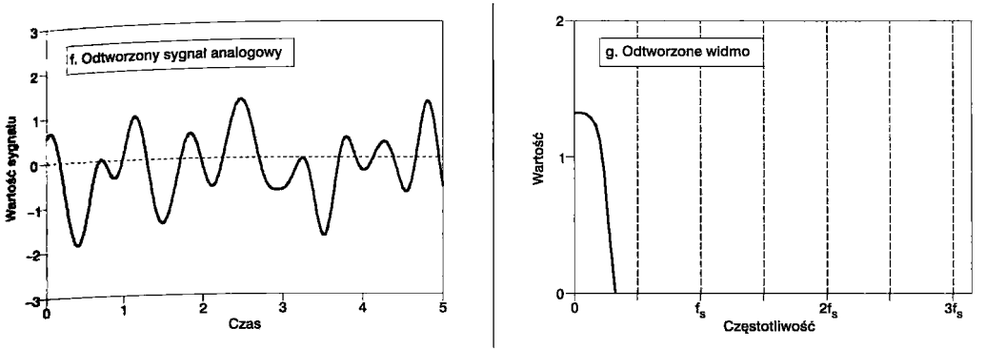
W dziedzinie częstotliwości pamiętanie rzędu zerowego powoduje *pomnożenie* widma ciągu impulsów przez przebieg zaznaczony pogrubioną linią na rys. 3.6 d), co można wyrazić wzorem:

$$H(f) = \left| \frac{\sin(\pi f / f_s)}{\pi f / f_s} \right|$$

Wzór 3.1. Ograniczenie amplitudy sygnału przy większych częstotliwościach w rezultacie pamiętania rzędu zerowego. Krzywą wyrażoną tym wzorem przedstawiono na rys. 3.6 d). Częstotliwość próbkowania jest oznaczona jako f_s. Dla $f = 0$, $H(f) = 1$



Rys. 3.6. Analiza przetwarzania cyfrowo-analogowego. Na rys. a) dane cyfrowe są przetworzone w ciąg impulsów, którego widmo przedstawiono na rys. b). Dane dają sygnał odtworzony (rys. f) w wyniku zastosowania elektronicznego filtru dolnoprzepustowego usuwającego częstotliwości powyżej połowy częstotliwości próbkowania (proszę porównać rysunki b) i g). Jednak w większości elektronicznych przetworników c/a uzyskuje się nie ciąg impulsów, lecz przebieg z pamiętaniem rzędu zerowego. Widmo przebiegu z pamiętaniem rzędu zerowego jest równe widmu ciągu impulsów pomnożonemu przez funkcję sinc pokazaną na rys. d). W celu przetworzenia sygnału z pamiętaniem rzędu zerowego w sygnał odtworzony, trzeba filtrem analogowym usunąć wszystkie częstotliwości powyżej częstotliwości Nyquista oraz dokonać korekty ze względu na funkcję sinc, jak to przedstawiono na rys. e)



Rys. 3.6. (cd.)

Jest to ogólna postać funkcji $\sin(\pi x) / (\pi x)$, oznaczanej jako ***sinc*** lub ***sinc***(x). Funkcja *sinc* jest często stosowana w dziedzinie DSP i omówimy ją szczegółowo w dalszych rozdziałach. Mając już pewną wiedzę w tej kwestii, można pamiętanie rzędu zerowego rozumieć jako splot ciągu impulsów i impulsu prostokątnego mającego szerokość równą okresowi próbkowania. W dziedzinie częstotliwości powoduje to *mnożenie* przez transformatę Fouriera impulsu prostokątnego, tj. przez funkcję *sinc*. Na rys. 3.6 d) cienką linią pokazano widmo częstotliwościowe ciągu impulsów (widmo „poprawne"), a linią pogrubioną – funkcję *sinc*. Widmo częstotliwościowe sygnału z pamiętaniem rzędu zerowego jest iloczynem tych dwóch krzywych.

Filtr analogowy zastosowany do przetwarzania sygnału z pamiętaniem rzędu zerowego (przebieg na rys. 3.6 c) na przebieg odtworzony (rys. 3.6 f) powinien wykonać dwa zadania: 1) usunąć wszystkie częstotliwości powyżej połowy częstotliwości próbkowania, 2) wzmocnić widmo częstotliwościowe o współczynnik równy odwrotności wpływu pamiętania rzędu zerowego, tj. $1/sinc(x)$. To oznacza wzmocnienie o ok. 36% przy połowie częstotliwości próbkowania. Idealną odpowiedź tego filtru w dziedzinie częstotliwości przedstawiono na rys. 3.6 e).

Wzmocnienie widma częstotliwościowego o współczynnik $1/sinc(x)$ można potraktować na jeden z czterech sposobów: 1) zignorować to i pogodzić się ze skutkami, 2) zaprojektować filtr z uwzględnieniem odpowiedzi $1/sinc(x)$, 3) zastosować pomysłową technikę przetwarzania o wielu częstotliwościach próbkowania (*multirate*) opisaną w dalszej części tego rozdziału, 4) wprowadzić odpowiednią korekcję do oprogramowania przed przetwarzaniem c/a (patrz rozdział 24).

Przed zakończeniem omawiania próbkowania trzeba rozwiać powszechnie przyjmowany mit dotyczący porównania sygnałów analogowych z cyfrowymi. Jak wykazano w tym rozdziale, ilość informacji przenoszonej w sygnale cyfrowym jest ograniczona dwiema przyczynami. Po pierwsze, liczba bitów na próbkę ogranicza rozdzielczość *zmiennej zależnej*. Małe zmiany wartości sygnału mogą być utracone w szumie kwantyzacji. Po drugie, częstotliwość próbkowania ogranicza rozdzielczość *zmiennej niezależnej*, to znaczy, że zdarzenia znajdujące się blisko siebie w sygnale analogowym mogą zostać utracone między próbkami. Jest to inny sposób stwierdzenia, że są tracone częstotliwości większe od połowy częstotliwości próbkowania.

A oto ten mit: „Ponieważ sygnały analogowe mają charakter ciągły, więc mają nieskończenie dobrą rozdzielczość zarówno zmiennej niezależnej, jak i zależnej". To nieprawda! Sygnały analogowe są ograniczone z dwóch przyczyn, tych samych co sygnały cyfrowe. A są nimi *szum* i *pasmo częstotliwości* (największa częstotliwość dopuszczalna w sygnale). Szum w sygnale analogowym ogranicza możliwość pomiaru wartości przebiegu, podobnie jak szum kwantyzacji w sygnale cyfrowym. W sygnale analogowym możliwość rozróżniania zdarzeń położonych blisko siebie zależy od największej dopuszczalnej częstotliwości w przebiegu.

Aby to zrozumieć, wyobraźmy sobie sygnał analogowy zawierający dwa położone blisko siebie impulsy. Jeśli przepuścimy ten sygnał przez filtr dolnoprzepustowy (usuwający większe częstotliwości), to impulsy zleją się ze sobą. Na przykład sygnał analogowy zawierający częstotliwości od 0 do 10 kHz będzie miał *dokładnie* taką samą rozdzielczość, jak sygnał cyfrowy powstający w wyniku próbkowania z częstotliwością 20 kHz. Musi tak być, gdyż zgodnie z twierdzeniem o próbkowaniu oba te sygnały zawierają tę samą informację.

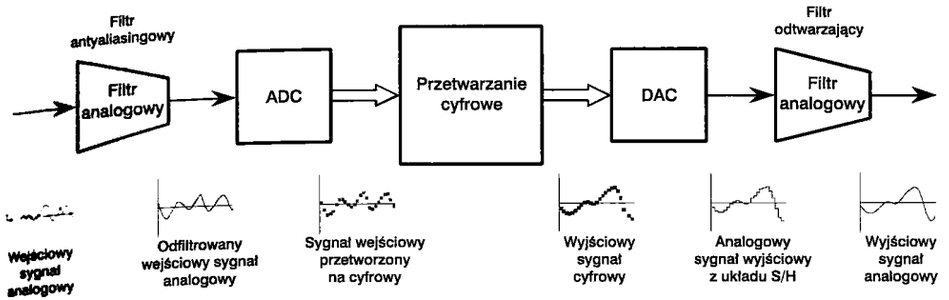
Filtry analogowe w przetwarzaniu danych

Na rys. 3.7 przedstawiono schemat blokowy systemu cyfrowego przetwarzania danych *zgodnego z wymogami twierdzenia o próbkowaniu*. Sygnał wejściowy, przed wprowadzeniem go do przetwornika a/c, przechodzi przez elektroniczny filtr dolnoprzepustowy usuwający wszystkie częstotliwości powyżej częstotliwości Nyquista (połowy częstotliwości próbkowania). To jest konieczne, aby zapobiec powstawaniu aliasingu podczas próbkowania i dlatego ten filtr jest nazywany **antyaliasingowym**. W dalszej części systemu sygnał, przetworzony już na cyfrowy, jest poddawany przetwarzaniu c/a i przechodzi do następnego filtru dolnoprzepustowego ustawionego na częstotliwość Nyquista. Ten filtr wyjściowy jest nazywany **odtwarzającym** (albo rekonstrukcyjnym) i może też realizować poprzednio opisane wzmacnianie widma częstotliwości związane z korektą wpływu pamiętania zerowego rzędu. Z tym prostym modelem jest, niestety, spory kłopot. Ograniczenia filtrów elektronicznych mogą być problemem tak samo poważnym, jak trudności, którym mają zapobiegać.

Jeśli interesujecie się głównie oprogramowaniem, to pewno uważacie, że nie warto czytać tego podrozdziału. *Nie macie racji*! Nawet jeśli przyrzekliście sobie nigdy nie dotknąć oscyloskopu, to zrozumienie właściwości filtrów analogowych jest bardzo ważne dla osiągnięcia sukcesu w dziedzinie DSP. Po pierwsze, właściwości każdego zdigitalizowanego sygnału, z jakim się spotkacie, będą zależały od rodzaju filtru antyaliasingowego zastosowanego przy pobieraniu sygnału. Jeśli nie rozumiecie natury filtru antyaliasingowego, to nie możecie zrozumieć natury sygnału cyfrowego. Po drugie, przyszłością DSP jest zastępowanie *rozwiązań układowych* przez *oprogramowanie*. Na przykład metoda przetwarzania o wielu częstotliwościach próbkowania, przedstawiona w dalszej części tego rozdziału, ogranicza konieczność stosowania filtrów antyaliasingowych i odtwarzających dzięki pomysłowym trikom programowym. Jeśli nie rozumiecie rozwiązań układowych, to nie będziecie umieli zaprojektować oprogramowania, które mogłoby je zastąpić. Po trzecie, wiele zagadnień DSP wiąże się z projektowaniem filtrów cyfrowych. Powszechnie przyjętym sposobem postępowania jest rozpoczynanie projektowania od równoważnego *filtru analogowego* i przekształcenie go na oprogramowanie. W materiale dalszych rozdziałów książki zakłada się, że Czytelnicy mają podstawowe wiadomości o technice filtrów analogowych.

Na ogół są stosowane trzy rodzaje filtrów analogowych: **Czebyszewa, Butterwotha i Bessela** (zwany też filtrem Thomsona). Każdy rodzaj filtru jest przewidziany do optymalizacji innych parametrów. Złożoność każdego filtru można dobierać przez wybór liczby **biegunów** i **zer.** Są to pojęcia matematyczne, które dalej objaśnimy. Im więcej jest biegunów w filtrze, tym więcej podzespołów elektronicznych on wymaga i tym lepiej działa. Każda z wymienionych nazw opisuje, co filtr *robi*, a nie określoną konfigurację rezystorów i kondensatorów. Na przykład 6-biegunowy filtr Bessela może być realizowany przy użyciu wielu różnych typów układów, które wszystkie mają taką samą charakterystykę wypadkową całkowitą. Do celów DSP charakterystyki tych filtrów są ważniejsze od ich konstrukcji. Niemniej jednak rozpoczniemy od krótkiego opisu elektronicznego projektowania tych filtrów, aby dać Czytelnikom ogólne pojęcie na ten temat.

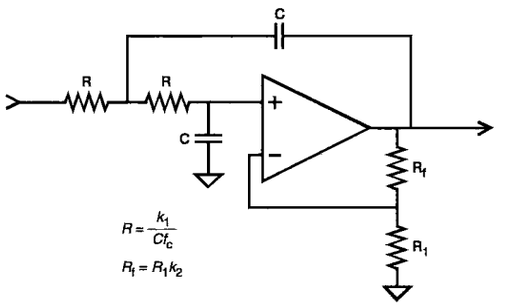
Na rys. 3.8 przedstawiono układ często stosowany jako element filtrów analogowych. Jest to zmodyfikowany układ Sallena-Keya. Nazwa pochodzi od nazwisk autorów artykułu z lat pięćdziesiątych, w którym opisali tę technikę. Przedstawiony układ jest dwubiegunowym filtrem dolnoprzepustowym, który można konfigurować jako każdy z trzech podstawowych typów filtru. W tablicy 3.1 podano informacje umożliwiające wybór odpowiednich rezystorów i kondensatorów. Na przykład projektując 2-biegunowy, 1 kHz filtr Butterwortha znajdujemy w tablicy 3.1 współczynniki $k_1 = 0{,}1592$ i $k_2 = 0{,}586$. W sposób dowolny wybierając $R_1 = 10\ k\Omega$ oraz $C = 0{,}01\ \mu F$ (typowe wartości w układach ze wzmacniaczami operacyjnymi), możemy obliczyć wartości R i R_f równe, odpowiednio 15,95 kΩ i 5,86 kΩ. Zaokrąglenie tych rezystancji do najbliższych wartości z szeregu rezystorów o tolerancji 1% daje: R = 15,8 kΩ i $R_f = 5{,}90\ k\Omega$. Wszystkie podzespoły powinny mieć tolerancję 1% lub lepszą.



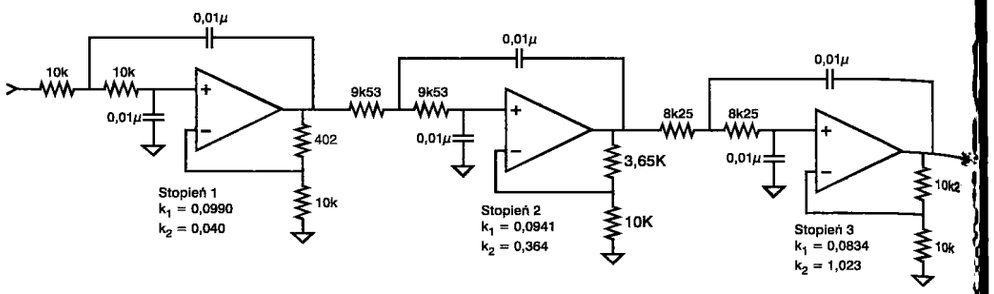
Rys. 3.7. Elektroniczne filtry analogowe stosowane w celu spełnienia wymogów twierdzenia o próbkowaniu. Filtr elektroniczny umieszczony przed przetwornikiem a/c jest nazywany filtrem antyaliasingowym. Służy on do usuwania składowych częstotliwości powyżej połowy częstotliwości próbkowania. Te składowe mogłyby wywołać zjawisko aliasingu podczas próbkowania. Filtr elektroniczny znajdujący się za przetwornikiem c/a nosi nazwę filtru odtwarzającego. On także eliminuje częstotliwości powyżej częstotliwości Nyquista i może wprowadzać korektę związaną z pamiętaniem zerowego rzędu

Wybór typu wzmacniacza operacyjnego nie jest krytyczny, o ile tylko częstotliwość przy wzmocnieniu jednostkowym jest od 30 do 100 razy większa od częstotliwości odcięcia filtru. Nie jest to wymaganie trudne do spełnienia, jeśli częstotliwość odcięcia jest mniejsza od ok. 100 kHz.

Filtry 4-, 6- i 8-biegunowe tworzy się przez łączenie kaskadowe 2, 3 lub 4 układów podstawowych. Na przykład na rys. 3.9 przedstawiono schemat 6-biegunowego filtru Bessela złożonego z 3 stopni kaskadowych. Każdy stopień ma różne wartości współczynników k_1 i k_2, zgodnie z tablicą 3.1, a więc różne są też wartości zastosowanych rezystorów i kondensatorów. Jeśli jest potrzebny filtr górnoprzepustowy, zamieńcie miejscami elementy C i R w układzie, nie ruszając R_f i R_1.



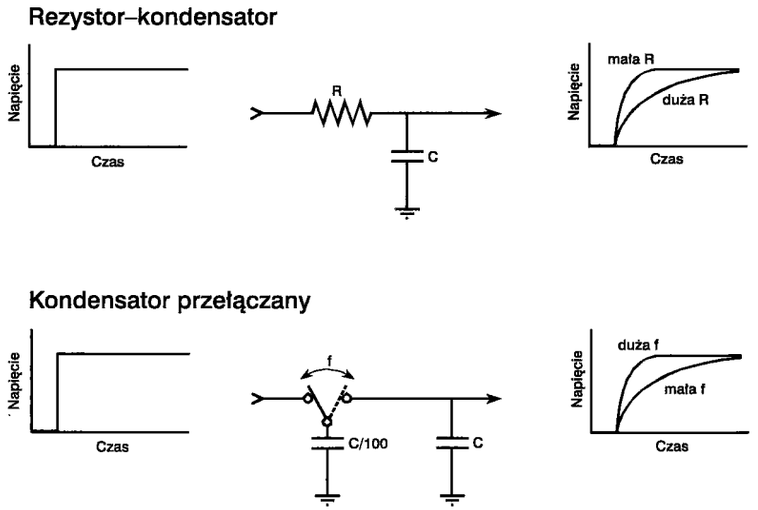
Rys. 3.8. Zmodyfikowany układ Sallena-Keya, podstawowy element filtrów aktywnych. Na rysunku przedstawiono realizację 2-biegunowego filtru dolnoprzepustowego. Filtry wyższych rzędów (o większej liczbie biegunów) można budować przez kaskadowe łączenie wielu stopni. Należy wybrać współczynniki k_1 i k_2 z tablicy 3.1, w dowolny sposób dobrać R_1 i C (np. 10 kΩ i 0,01 μF) i następnie obliczyć R i R_f z wzorów podanych na rysunku. Wielkość f_c jest częstotliwością odcięcia filtru (w Hz)



Rys. 3.9. 6-biegunowy filtr Bessela utworzony przez kaskadowe połączenie 3 układów Sallena-Keya. Jest to filtr dolnoprzepustowy o częstotliwości odcięcia 1 kHz

Tab. 3.1. Współczynniki do projektowania filtrów Bessela, Butterwortha i Czebyszewa (zafalowania 6%)

Liczba biegunów		Bessel		Butterworth		Czebyszew	
		k_1	k_2	k_1	k_2	k_1	k_2
2	Stopień 1	0,1251	0,268	0,1592	0,586	0,1293	0,842
4	Stopień 1	0,1111	0,084	0,1592	0,152	0,2666	0,582
	Stopień 2	0,0991	0,759	0,1592	1,235	0,1544	1,660
6	Stopień 1	0,0990	0,040	0,1592	0,068	0,4019	0,537
	Stopień 2	0,0941	0,364	0,1592	0,586	0,2072	1,448
	Stopień 3	0,0834	1,023	0,1592	1,483	0,1574	1,846
8	Stopień 1	0,0894	0,024	0,1592	0,038	0,5359	0,522
	Stopień 2	0,0867	0,213	0,1592	0,337	0,2657	1,379
	Stopień 3	0,0814	0,593	0,1592	0,889	0,1848	1,711
	Stopień 4	0,0726	1,184	0,1592	1,610	0,1582	1,913



Rys. 3.10. Objaśnienie działania filtru z przełączanymi pojemnościami. W filtrach tego rodzaju rezystory są zastąpione kondensatorami i przełącznikami. Jak wynika z odpowiedzi układów na funkcję skokową, dwa kondensatory i przełącznik mogą realizować taką samą funkcję, jak obwód rezystor–kondensator

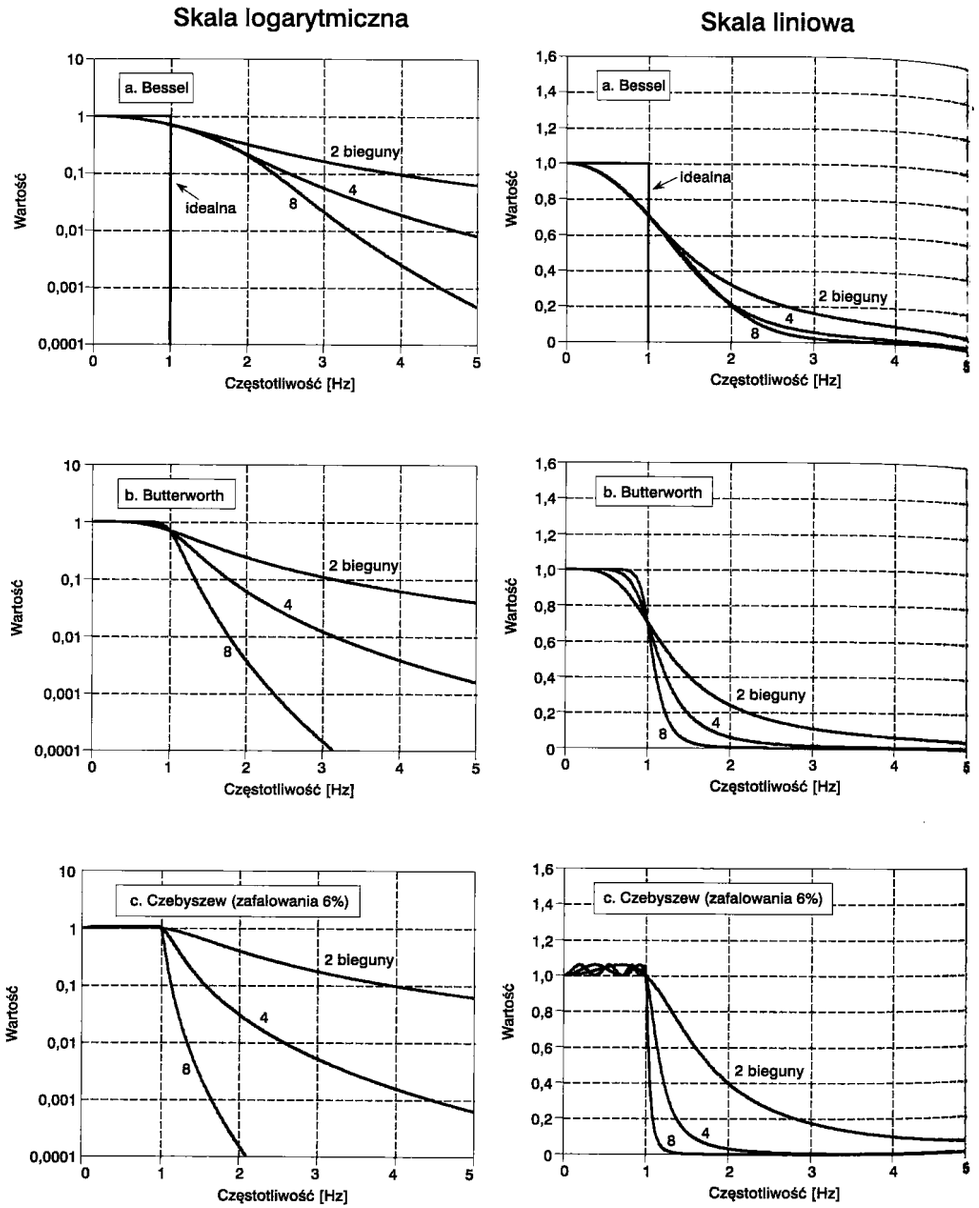
Ten rodzaj układu jest stosowany w produkcji małoseryjnej i w pracach badawczych. W urządzeniach produkowanych wielkoseryjnie konieczne są filtry wykonywane jako *układy scalone*. Pojawia się tu trudność z wykonywaniem rezystorów bezpośrednio w krzemie. Rozwiązaniem tego problemu są **filtry z przełączanymi pojemnościami**. Na rys. 3.10 zilustrowano zasadę działania takiego filtru przez porównanie go z prostym obwodem RC. Wprowadzając funkcję skokową na wejście dolnoprzepustowego filtru RC, uzyskuje się eksponencjalne narastanie napięcia wyjściowego aż do zrównania się z sygnałem wejściowym. Napięcie na kondensatorze nie zmienia się natychmiastowo, gdyż rezystor ogranicza przepływ ładunku elektrycznego.

W filtrze z przełączanymi pojemnościami podstawowy obwód rezystor–kondensator zastępuje się dwoma kondensatorami i przełącznikiem. Nowo dodany kondensator ma wartość znacznie mniejszą niż kondensator z obwodu RC, jest to, powiedzmy 1% pojemności tego kondensatora. Przełącznik dołącza mały kondensator na przemian do wejścia i wyjścia, z bardzo dużą częstotliwością, typowo 100 razy większą niż częstotliwość odcięcia filtru. Gdy do wejścia jest dołączony mały kondensator, ładuje się on szybko do chwilowej wartości napięcia wejściowego. Gdy przełącznik zostaje dołączony do wyjścia, ładunek z małego kondensatora jest przekazywany do dużego. W rezystorze szybkość przekazywania ładunku jest określona rezystancją. W układzie z przełączaną pojemnością szybkość przekazywania ładunku jest określona pojemnością małego kondensatora *oraz* częstotliwością przełączania. Stąd wynika bardzo korzystna cecha filtrów z przełączanymi pojemnościami: *częstotliwość odcięcia filtru jest wprost proporcjonalna do częstotliwości taktowania zastosowanej do sterowania przełącznikami*. To czyni filtr z przełączanymi pojemnościami idealnym rozwiązaniem do systemów akwizycji danych pracujących z więcej niż jedną częstotliwością próbkowania. Są to układy dostępne do powszechnego stosowania; za 10 dolarów można mieć 8-biegunowy filtr w postaci układu scalonego z 8 wyprowadzeniami.

Przejdźmy teraz do ważnej kwestii dotyczącej parametrów trzech klasycznych typów filtrów. Pierwszym parametrem, któremu poświęcimy nieco uwagi, jest **stromość spadku charakterystyki** w obszarze odcięcia. Filtr dolnoprzepustowy jest przeznaczony do tłumienia wszystkich częstotliwości powyżej częstotliwości odcięcia (pasmo zaporowe) i przepuszczania wszystkich częstotliwości poniżej odcięcia (pasmo przepustowe). Na rys. 3.11 pokazano w skali logarytmicznej (dB) odpowiedzi częstotliwościowe filtrów trzech podstawowych rodzajów. Wykresy przedstawiono dla filtrów o częstotliwości odcięcia 1 Hz, lecz można je łatwo przeskalować dla każdej innej wymaganej częstotliwości odcięcia. Jak wypada porównanie tych filtrów pod względem stromości spadku charakterystyki? Wyraźnie najlepszy jest filtr Czebyszewa, Butterwortha jest gorszy, a Bessela całkiem koszmarny! Jak pewno już zauważyliście, filtr Czebyszewa jest zaprojektowany tak, aby **spadek** amplitudy (*roll-off*) był możliwie jak najszybszy.

Niestety, nawet 8-biegunowy filtr Czebyszewa nie jest wystarczająco dobry do filtrów antyaliasingowych. Wyobraźmy sobie np. system 12-bitowy o częstotliwości próbkowania 10 000 próbek/s. Z twierdzenia o próbkowaniu wynika, że wszystkie częstotliwości powyżej 5 kHz podlegają aliasingowi, a tego właśnie chcemy uniknąć. Po krótkim namyśle decydujemy, że amplituda wszystkich składowych powyżej 5 kHz powinna być zredukowana o czynnik 100. Wtedy wszystkie składowe o częstotliwościach podlegających aliasingowi miałyby amplitudy poniżej 1%. Na podstawie rys. 3.11 c) stwierdzamy, że 8-bitowy filtr Czebyszewa o częstotliwości odcięcia 1 Hz osiąga współczynnika tłumienia (czyli redukcji sygnału) równy 100 przy częstotliwości ok. 1,35 Hz. Dokonując przeskalowania odpowiedniego dla naszego przykładu, trzeba ustalić częstotliwość odcięcia filtru na 3,7 kHz, aby wszystkie składowe powyżej 5 kHz zostały należycie stłumione. W rezultacie zakres od 3,7 do 5 kHz jest niepotrzebnie tracony z powodu niewłaściwego odcięcia filtrem analogowym.

I jeszcze delikatna kwestia. Współczynnik tłumienia 100 w tym przykładzie jest przypuszczalnie wystarczający, mimo że w 12 bitach mamy tu 4096 poziomów. Zgodnie z rys. 3.4 składowa 5100 Hz będzie się nakładała na 4900 Hz, 6000 Hz na 4000 Hz itd. Nie musimy

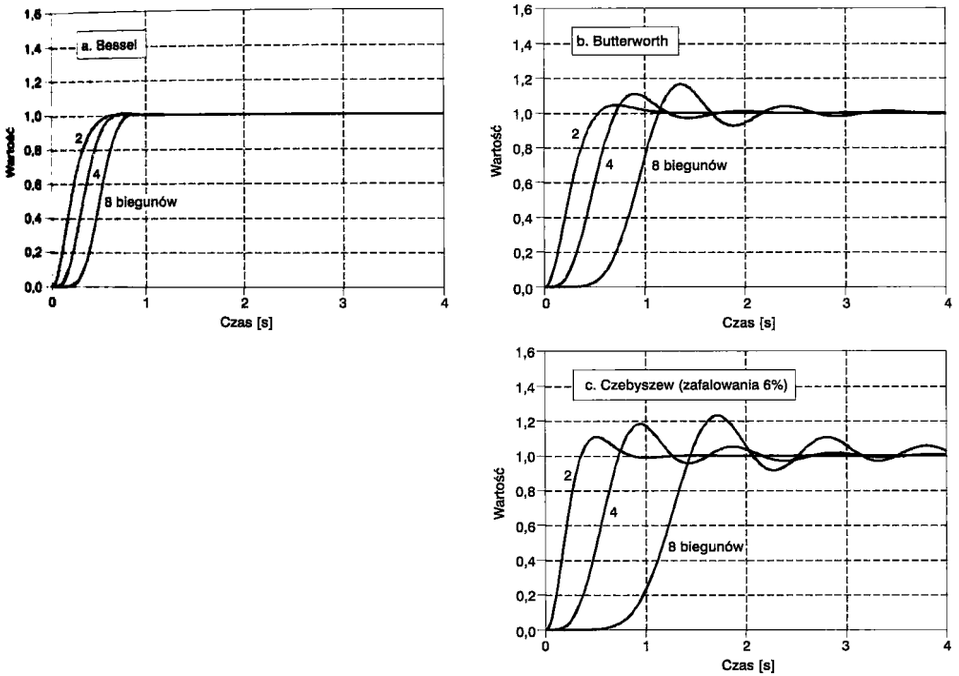


Rys. 3.11. Odpowiedź częstotliwościowa filtrów trzech podstawowych rodzajów, przedstawiona w skali logarytmicznej. Najbardziej strome opadanie zbocza ma filtr Czebyszewa

Rys. 3.12. Odpowiedź częstotliwościowa filtrów trzech podstawowych rodzajów, przedstawiona w skali liniowej. Filtr Butterwotha ma najbardziej płaską charakterystykę w paśmie przepustowym

się martwić, jakie są wartości składowych o częstotliwościach między 5000 Hz i 6300 Hz, ponieważ w wyniku aliasingu nałożą się one na częstotliwości w nieużywanym obszarze 3700 i 5000 Hz. Aliasingowi do pasma przepustowego filtru mogą ulegać częstotliwości większe od 6300 Hz, czyli od 1,7 częstotliwości odcięcia filtru równej 3700 Hz. Jak wynika z rys. 3.11 c), współczynnik tłumienia 8-biegunowego filtru Czebyszewa przy częstotliwości równej 1,7 częstotliwości odcięcia wynosi ok. 1300, a więc znacznie więcej niż wymagana wartość 100, od której zaczęliśmy naszą analizę. A morał z tej historii jest następujący: *W większości systemów pasmo częstotliwości pomiędzy ok. 0,4 do 0,5 częstotliwości próbkowania jest jałowym, bezużytecznym obszarem opadania charakterystyki filtru i nakładania się sygnałów ulegających aliasingowi.* Jest to bezpośredni skutek ograniczeń filtru analogowego.

Odpowiedź częstotliwościowa idealnego filtru dolnoprzepustowego powinna być *płaska* w całym paśmie przepustowym. Charakterystyki wszystkich filtrów na rys. 3.11 prezentują się pod tym względem doskonale, lecz tylko dzięki temu, że pionowa skala wykresów jest *logarytmiczna*. Inaczej to wygląda, gdy skalę zamieni się na *liniową*, co przedstawiono na rys. 3.12. Na charakterystyce filtru Czebyszewa widać **zafalowania w paśmie przepustowym** (faliste zmiany amplitudy). W istocie rzeczy w filtrze Czebyszewa uzyskuje się doskonałą charakterystykę spadku amplitudy kosztem dopuszczenia do zafalowań w paśmie przepustowym. Gdy są dopuszczalne większe zafalowania, można uzyskać szybszy spadek amplitudy w obszarze częstotliwości odcięcia. Wszystkie filtry Czebyszewa zaprojektowane według tablicy 3.1 mają zafalowania w paśmie przepustowym ok. 6% (0,5 dB) – dobry kompromis, na ogół dopuszczalny. W podobnym układzie, w **filtrze eliptycznym**, są dopuszczalne tętnienia zarówno w paśmie przepustowym, *jak i* w zaporowym. Filtry eliptyczne, choć trudne do zaprojektowania, dają jednak jeszcze lepszy kompromis między szybkością opadania i tętnieniami w paśmie przepustowym.

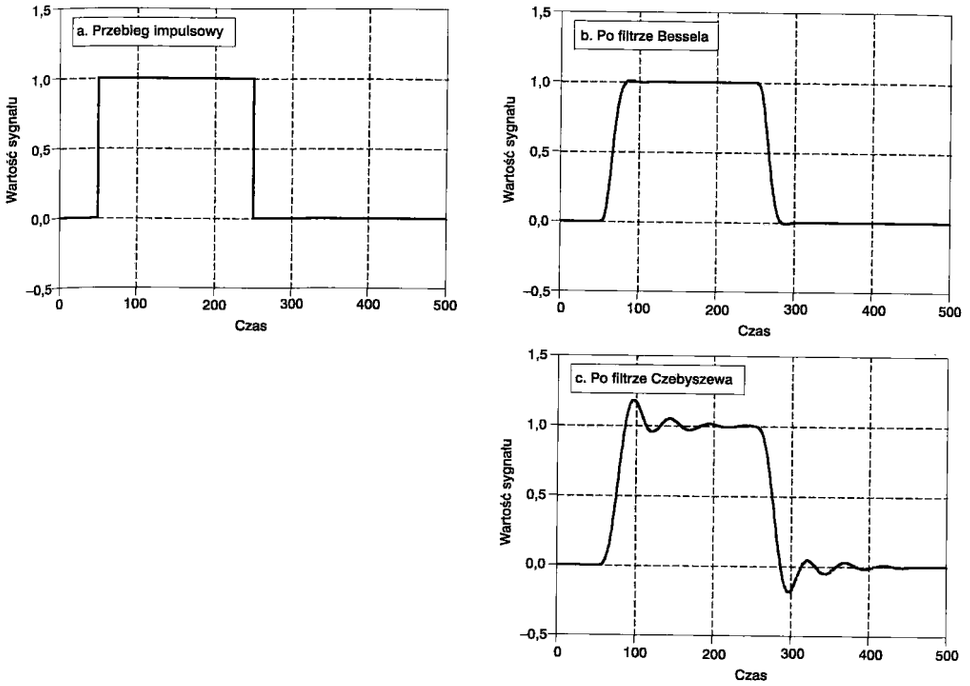


Rys. 3.13. Odpowiedzi skokowe trzech podstawowych filtrów. Skala czasu na osi poziomej odpowiada częstotliwości odcięcia 1 Hz. Filtr Bessela jest optymalny do tych zastosowań, w których przerzut i „dzwonienia" muszą być zminimalizowane

Filtr Butterwortha zaś jest zoptymalizowany pod względem najlepszej stromości spadku charakterystyki w obszarze częstotliwości odcięcia, lecz *bez* zafalowań w paśmie przepustowym. Jest zwykle nazywany *filtrem o maksymalnie płaskiej charakterystyce* i jest identyczny z filtrem Czebyszewa zaprojektowanym dla zerowych zafalowań w paśmie przepustowym. W filtrze Bessela nie ma tych zafalowań, ale charakterystyka opadania jest znacznie gorsza niż w filtrze Butterwortha.

Ostatnim parametrem, który poddamy ocenie, jest **odpowiedź skokowa**, czyli odpowiedź filtru na szybką zmianę sygnału wejściowego od jednej wartości do drugiej. Na rys. 3.13 przedstawiono odpowiedzi skokowe trzech podstawowych rodzajów filtrów. Oś pozioma jest pokazana dla filtrów o częstotliwości odcięcia 1 Hz, lecz może być przeskalowana dla większych częstotliwości odcięcia. Na przykład częstotliwość odcięcia 1000 Hz da odpowiedź na skok jednostkowy wyrażoną w *milisekundach*, a nie w *sekundach*. Odpowiedzi filtrów Butterwortha i Czebyszewa mają **przerzuty** i tzw. **dzwonienia** (oscylacje o wolno zmniejszającej się amplitudzie). W filtrze Bessela zaś nie ma tych nieprzyjemnych zjawisk.

Te bardzo korzystne właściwości filtru Bessela są też widoczne na rys. 3.14. Na rys. 3.14 a) przedstawiono przebieg impulsowy, który można traktować jako zbocze narastające i następujące po pewnym czasie zbocze opadające. Na rys. 3.14 b), c) przedstawiono, jaka jest odpowiedź na taki przebieg na wyjściach filtrów, odpowiednio, Bessela i Czebyszewa. Gdyby był to sygnał wizyjny, to zniekształcenie wprowadzane przez filtr Czebyszewa byłoby nie-



Rys. 3.14. Odpowiedź filtrów Bessela i Czebyszewa na przebieg impulsowy. Podstawową właściwością filtru Bessela jest to, że zbocza narastające i opadające odpowiedzi wyglądają podobnie. Ta właściwość jest w żargonie technicznym nazywana liniową fazą. Na rys. b) jest pokazany wynik przepuszczenia przebiegu z rys. a) przez 4-biegunowy filtr Bessela. Oba zbocza są wygładzone w podobny sposób. Na rys. c) pokazano wynik przepuszczenia przebiegu z rys. a) przez 4-biegunowy filtr Czebyszewa. Lewe zbocze ma przerzut na wierzchołku, a prawe – u dołu. Takie zniekształcenia są w wielu zastosowaniach niedopuszczalne

dopuszczalne. Przerzut zmieniłby jasność na *krawędziach* obiektów w stosunku do części *środkowych*. Co gorsza, lewa strona obiektu będzie się wydawała jasna, a prawa – ciemna. W wielu zastosowaniach kiepska odpowiedź skokowa jest niedopuszczalna. W tych zastosowaniach dominują zalety filtrów Bessela – brak przerzutu i symetryczne zbocza.

Wybór filtru antyaliasingowego

W tablicy 3.2 zestawiono właściwości trzech podstawowych filtrów. Na podstawie tych danych można stwierdzić, jak w każdym z filtrów jeden parametr jest optymalizowany kosztem innych. Filtr Czebyszewa jest zoptymalizowany pod względem *stromości opadania charakterystyki*, Butterwortha – jej *płaskości w paśmie przepustowym*, Bessela zaś pod względem odpowiedzi skokowej.

Wybór filtru antyaliasingowego prawie całkowicie zależy od jednej kwestii: *jak jest reprezentowana informacja w sygnałach, które mamy zamiar przetwarzać*. Jest wiele sposobów kodowania informacji w przebiegu analogowym, ale tylko dwa sposoby są powszechnie stosowane – **kodowanie w dziedzinie czasu** i **kodowanie w dziedzinie częstotliwości**. Różnica między tymi sposobami ma bardzo istotne znaczenie w DSP i będzie na kartach tej książki tematem wielokrotnie powracającym.

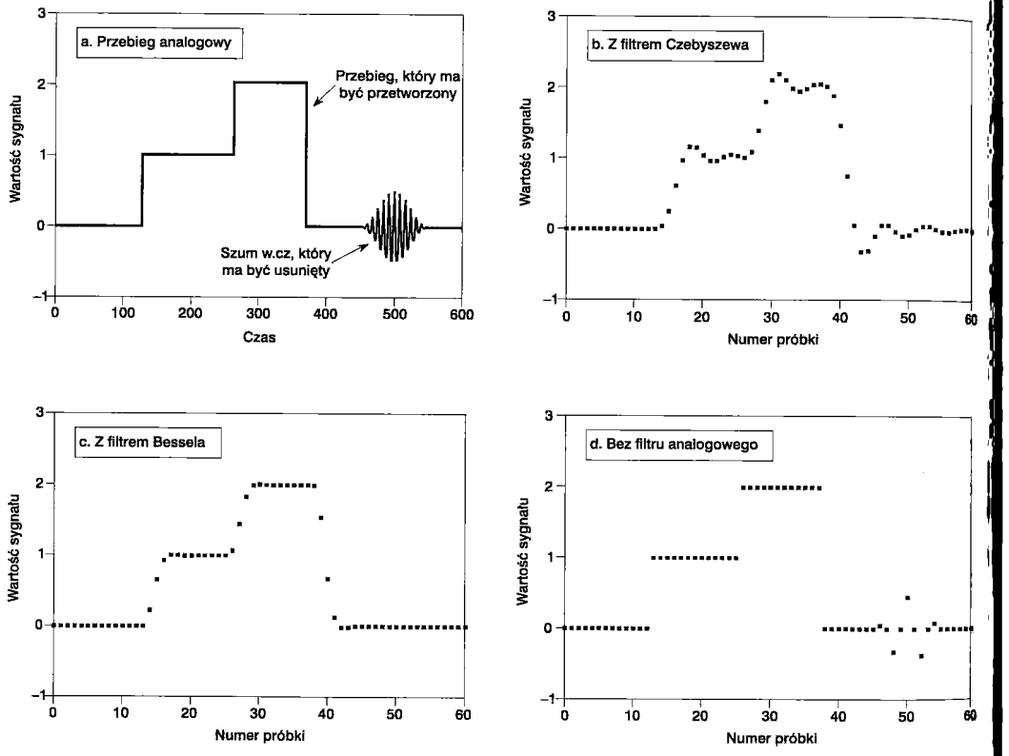
W przypadku *kodowania w dziedzinie częstotliwości* informacja jest zawarta w przebiegach sinusoidalnych, które zsumowane tworzą sygnał. Doskonałym przykładem są sygnały foniczne, zwane w skrócie sygnałami audio. Gdy człowiek słucha mowy lub muzyki, to odbierany dźwięk zależy od częstotliwości w nim występujących, a nie od jakiegoś szczególnego *kształtu* przebiegu. Można to wykazać, przepuszczając sygnał audio przez układ zmieniający fazy różnych sinusoid, lecz zachowujący ich częstotliwości i amplitudy.

Tab. 3.2. Parametry trzech podstawowych rodzajów filtrów. Filtr Bessela daje najlepszą odpowiedź skokową; dlatego jest stosowany do sygnałów kodowanych w dziedzinie czasu. Filtry Czebyszewa i Butterwortha skutecznie eliminują składowe częstotliwości w paśmie zaporowym, dlatego doskonale nadają się do sygnałów kodowanych w dziedzinie częstotliwości. Wartości w tablicy są podane w sekundach i hercach, dla częstotliwości odcięcia 1 Hz

	Stałoprądowe wzmocnienie napięciowe	Odpowiedź skokowa: Przerzut	Odpowiedź skokowa: Czas ustalania do 1%	Odpowiedź skokowa: Czas ustalania do 0,1%	Odpowiedź częstotliwościowa: Zafalowania w paśmie przepustowym	Odpowiedź częstotliwościowa: Częstotliwość, dla której współczynnik tłumienia jest równy 100	Odpowiedź częstotliwościowa: Częstotliwość, dla której współczynnik tłumienia jest równy 1000
Bessel							
2-biegunowy	1,27	0,4%	0,60	1,12	0%	12,74	40,4
4-biegunowy	1,91	0,9%	0,66	1,20	0%	4,74	8,45
6-biegunowy	2,87	0,7%	0,74	1,18	0%	3,65	5,43
8-biegunowy	4,32	0,4%	0,80	1,16	0%	3,35	4,53
Butterworth							
2-biegunowy	1,59	4,3%	1,06	1,66	0%	10,0	31,6
4-biegunowy	2,58	10,9%	1,68	2,74	0%	3,17	5,62
6-biegunowy	4,21	14,3%	2,74	3,92	0%	2,16	3,17
8-biegunowy	6,84	16,4%	3,50	5,12	0%	1,78	2,38
Czebyszew							
2-biegunowy	1,84	10,8%	1,10	1,62	6%	12,33	38,9
4-biegunowy	4,21	18,2%	3,04	5,42	6%	2,59	4,47
6-biegunowy	10,71	21,3%	5,86	10,4	6%	1,63	2,26
8-biegunowy	28,58	23,0%	8,34	16,4	6%	1,34	1,66

Utworzony w ten sposób sygnał *wygląda* na oscyloskopie zupełnie inaczej, lecz *brzmi* identycznie jak poprzedni. Istotna informacja pozostała nietknięta, chociaż kształt przebiegu został znacznie zmieniony. Ponieważ aliasing przemieszcza i nakłada na siebie składowe częstotliwości, więc bezpośrednio niszczy informację zakodowaną w dziedzinie częstotliwości. W rezultacie system digitalizacji tych sygnałów wymaga zastosowania filtru antyaliasingowego o charakterystyce z ostrym odcięciem, takich jak filtr Czebyszewa, eliptyczny lub Butterwortha. A co z marną odpowiedzią skokową tych filtrów? Ona nie ma tu znaczenia. Zakodowana informacja nie jest wrażliwa na ten rodzaj zniekształceń.

W *kodowaniu w dziedzinie czasu* do zapamiętania informacji wykorzystuje się *kształt przebiegu*. Na przykład lekarze mogą monitorować przebiegi elektryczne w sercu pacjenta, dołączając elektrody do jego klatki piersiowej i rąk (EKG, czyli elektrokardiogram). Kształt *przebiegu* EKG dostarcza szukanej informacji dotyczącej np. czasu skurczów komór serca. Innym przykładem tego rodzaju sygnału są obrazy. W obrazach informacja jest zawarta nie w przebiegu zmieniającym się w funkcji *czasu*, lecz raczej w kształcie przebiegu zmieniają-



Rys. 3.15. Trzy opcje filtru antyaliasingowego dla sygnałów kodowanych w dziedzinie czasu. Celem jest wyeliminowanie dużych częstotliwości (które uległyby zjawisku aliasingu) przy jednoczesnym zachowaniu ostrości zboczy (które niosą informację). Na rys. a) pokazano przykład sygnału analogowego zawierającego zarówno ostre zbocza, jak i paczkę szumu w.cz. Na rys. b) przedstawiono sygnał przetworzony na cyfrowy z użyciem filtru Czebyszewa. Wprawdzie duże częstotliwości zostały skutecznie usunięte, lecz zbocza uległy znacznemu zniekształceniu. Jest to złe rozwiązanie. Filtr Bessela (rys. c) daje łagodne zaokrąglenie zboczy i usuwa duże częstotliwości. Na rys. d) przedstawiono sygnał przetworzony na cyfrowy bez filtru antyaliasingowego. W tym przypadku zbocza zachowały doskonałą stromość, jednak paczka szumu w.cz. uległa aliasingowi, dając kilka nic nieznaczących próbek

cego się w funkcji *odległości*. Obrazy są tworzone z pewnych obszarów jaskrawości i barw oraz ich relacji w stosunku do innych obszarów jaskrawości i barw. Patrząc na Mona Lisę, nie powiesz przecież: „*Cóż za interesujący zbiór sinusoid*".

I tu pojawia się problem. Twierdzenie o próbkowaniu służy do analizy tego, co dzieje się w dziedzinie częstotliwości podczas digitalizacji. Dlatego doskonale nadaje się do wyjaśnienia przetwarzania analogowo-cyfrowego sygnałów zawierających informację zakodowaną w dziedzinie częstotliwości. Twierdzenie o próbkowaniu niewiele jednak pomoże w wyjaśnieniu, jak sygnały kodowane w dziedzinie czasu powinny być przetwarzane na cyfrowe. Przyjrzyjmy się temu bliżej.

Na rys. 3.15 przedstawiono kilka przykładów digitalizacji sygnałów kodowanych w dziedzinie czasu. Na rys. 3.15 a) widzimy przebieg analogowy, który ma być przetworzony na cyfrowy. W tym przypadku informacją, którą chcemy zachować, jest *kształt* impulsów prostokątnych. Ten przykładowy sygnał zawiera też krótką paczkę sinusoid o wielkiej częstotliwości. One reprezentują szum szerokopasmowy, zakłócenia i inne podobne śmiecie, które zawsze pojawiają się w sygnałach analogowych. Na dalszych rysunkach pokazano, jak będzie wyglądał sygnał po digitalizacji z użyciem różnych filtrów antyaliasingowych: Czebyszewa, Bessela i bez filtru.

Ważne jest zrozumienie, że *żadne* z tych rozwiązań nie umożliwia odtworzenia sygnału oryginalnego na podstawie spróbkowanych danych. Sygnał oryginalny w sposób inherentny zawiera bowiem składowe o częstotliwościach większych od połowy częstotliwości próbkowania. Ponieważ te częstotliwości nie mogą występować w sygnale przetworzonym na cyfrowy, więc nie może ich również zawierać sygnał odtworzony. Te duże częstotliwości pochodzą z dwóch źródeł, którymi są: 1) szum i zakłócenia, które chcielibyśmy wyeliminować, oraz 2) ostre zbocza przebiegu, zawierające przypuszczalnie informacje, które chcielibyśmy zachować.

W filtrze Czebyszewa (rys. 3.15 b) ten problem jest rozwiązywany przez ostre odcięcie wszystkich składowych o dużych częstotliwościach. W rezultacie uzyskuje się odfiltrowany sygnał analogowy, który *może* być próbkowany i potem doskonale odtworzony. Jednak odtworzony sygnał jest identyczny z sygnałem *odfiltrowanym* a nie z *oryginalnym*. Chociaż podczas próbkowania niczego się nie utraciło, to przebieg został poważnie zniekształcony przez filtr anyaliasingowy. Jak wynika z rys. 3.15 b), kuracja okazała się gorsza niż sama choroba! A więc nie działajmy w ten sposób.

W celu rozwiązania tego problemu zaprojektowano filtr Bessela (rys. 3.15 c). Przebieg na jego wyjściu jest bardzo zbliżony do przebiegu oryginalnego, ma tylko nieco zaokrąglone zbocza. Dobierając częstotliwość odcięcia filtru, można uzyskać kompromis między gładkością zboczy a eliminacją składowych w.cz. w sygnale. Zastosowanie większej liczby biegunów umożliwia *lepszy* kompromis między tymi dwoma parametrami. Ogólnym zaleceniem jest wybór częstotliwości odcięcia równej około jednej czwartej częstotliwości próbkowania. Daje to dwie próbki na każdym zboczu sygnału. Zwróćcie uwagę, że zarówno w filtrze Bessela, jak i Czebyszewa zostaje usunięta paczka szumu wielkoczęstotliwościowego istniejąca w sygnale oryginalnym.

Ostatnią możliwością jest rezygnacja z filtru antyaliasingowego (rys. 3.15 d). Takie rozwiązanie ma tę zaletę, że wartość każdej próbki jest *identyczna* z wartością sygnału oryginalnego. Sygnał ma więc doskonałą stromość zboczy, a zmiana sygnału oryginalnego natychmiast znajduje swoje odzwierciedlenie w danych cyfrowych. Aliasing może jednak zniekształcić sygnał, co jest wadą tego rozwiązania. Zniekształcenie może przyjmować dwie różne formy. Po pierwsze, wielkoczęstotliwościowe zakłócenia i szumy, jak np. pojawiająca się w naszym przykładzie paczka sinusoid, będą przetworzone na jakieś niemające sensu próbki (jak na rys. 3.15 d). To znaczy, że każdy szum w.cz. występujący w sygnale analogowym pojawi się w sygnale cyfrowym jako szum, który uległ aliasingowi. W sensie ogólniejszym nie jest to problem próbkowania, lecz problem niedoskonałości elektroniki analogowej. Ograniczanie szumów i zakłóceń nie jest zadaniem przetwornika a/c; funkcję tę powinny spełniać ukła-

dy elektroniczne jeszcze przed digitalizacją sygnału. Może się okazać, że w celu rozwiązania problemu filtr Bessela powinien być umieszczony przed przetwornikiem a/c. To jednak oznacza, że filtr trzeba traktować jako element przetwarzania analogowego, a nie jako układ działający na rzecz przetwarzania a/c.

Drugi objaw aliasingu jest trudniej uchwytny. Gdy w sygnale analogowym następuje jakieś zdarzenie (np. zbocze), to sygnał cyfrowy (rys. 3.15 d) wykrywa takie zdarzenie w *następnej* próbce. W danych cyfrowych nie ma żadnej informacji o tym, co dzieje się *między* próbkami. Teraz, w tym przykładzie, porównajmy sytuację w systemie *bez filtru* i w systemie *z filtrem Bessela*. Wyobraźmy sobie np. rysowanie prostych między próbkami na rys. 3.15 c). Chwila, gdy ta wykreślona linia przecina połowę amplitudy skoku, daje *podpróbkowe* oszacowanie czasu wystąpienia zbocza w sygnale analogowym. W systemie bez filtru taka informacja podpróbkowa jest całkowicie tracona. Nie musicie stosować jakichś zawiłych twierdzeń, aby oszacować, jak omawiany problem wpłynie na wasze konkretne zastosowanie. Wystarczy dobre zrozumienie tego, co zamierzacie zrobić z danymi po ich akwizycji.

Przetwarzanie o wielu częstotliwościach próbkowania (*multirate*)

W elektronice występuje silna tendencja zastępowania *układów analogowych* przez *algorytmy cyfrowe*. Doskonałym tego przykładem jest przetwarzanie danych. Rozpatrzmy projekt cyfrowego rejestratora głosu, czyli urządzenia, które digitalizuje sygnał głosowy, gromadzi dane w postaci cyfrowej i potem rekonstruuje sygnał w celu jego odtworzenia. Aby móc odtworzyć zrozumiały sygnał głosowy, urządzenie powinno wychwytywać sygnały o częstotliwości od 100 do 3000 Hz. Sygnał analogowy wytwarzany w mikrofonie zawiera jednak także znacznie większe częstotliwości, powiedzmy do 40 kHz. Najprostszym podejściem jest przepuszczenie sygnału analogowego przez 8-biegunowy, dolnoprzepustowy filtr Czebyszewa o częstotliwości odcięcia 3 kHz i następnie spróbkowanie sygnału z częstotliwością 8 kHz. Z drugiej strony systemu przetwornik c/a odtwarza sygnał analogowy z częstotliwością 8 kHz, z pamiętaniem rzędu zerowego. Do uzyskania końcowego sygnału głosowego jest zastosowany inny filtr Czebyszewa o częstotliwości odcięcia 3 kHz.

Wiele pożytecznych zalet ma próbkowanie *szybsze* niż to wynika z bezpośredniej analizy. Wyobraźmy sobie na przykład przeprojektowanie cyfrowego rejestratora głosu z zastosowaniem częstotliwości próbkowania 64 kHz. Zadanie filtru antyaliasingowego jest teraz łatwiejsze: przepuścić częstotliwości poniżej 3 kHz, usuwając wszystkie powyżej 32 kHz. Podobne uproszczenie dotyczy też filtru odtwarzającego. Krótko mówiąc, większa częstotliwość próbkowania umożliwia zastąpienie filtrów 8-biegunowych prostymi układami rezystor–kondensator (RC). Trudność polega na tym, że system cyfrowy jest teraz zapychany danymi z większą częstotliwością próbkowania.

Na następnym, wyższym szczeblu rozwoju są techniki przetwarzania, w których w jednym systemie stosuje się więcej niż jedną częstotliwość próbkowania (***multirate***). W przykładzie cyfrowego rejestratora głosu taka technika funkcjonuje następująco. Najpierw sygnał głosowy jest przepuszczany przez prosty dolnoprzepustowy filtr RC i próbkowany z częstotliwością 64 kHz. Uzyskane dane cyfrowe zawierają żądane pasmo głosu między 100 i 3000 Hz, lecz także nieużyteczne pasmo między 3 kHz i 32 kHz. Następnie usuwa się, w sposób *programowy*, wszystkie częstotliwości nieużyteczne, stosując *cyfrowy* filtr dolnoprzepustowy o częstotliwości odcięcia 3 kHz. Potem obniża się częstotliwość próbek w sygnale cyfrowym z 64 kHz do 8 kHz, pomijając po prostu każde 7 z 8 próbek. Taka procedura nosi nazwę **decymacji**. Uzyskane w ten sposób dane cyfrowe są równoważne danym wytwarzanym w wyniku ostrej filtracji analogowej i bezpośredniego próbkowania z częstotliwością 8 kHz.

Technika przetwarzania o wielu częstotliwościach próbkowania może też być stosowana w wyjściowej części rejestratora omawianego jako przykład. Dane 8 kHz są wyprowadzane z pamięci i przekształcane na częstotliwość próbkowania 64 kHz. Jest to procedura zwana **interpolacją**. Polega ona na umieszczaniu 7 próbek o wartości zerowej między kolejnymi próbkami otrzymywanymi z pamięci. Uzyskany sygnał jest cyfrowym *ciągiem impulsowym* zawierającym żądane pasmo głosu między 100 i 3000 Hz oraz powielane widma między 3 kHz i 32 kHz. Spójrzmy jeszcze raz na rys. 3.6 a), b), aby zrozumieć dlaczego tak jest. Wszystkie składowe powyżej 3 kHz są następnie usuwane *cyfrowym* filtrem dolnoprzepustowym. Po przetworzeniu na sygnał analogowy przetwornikiem c/a wystarczy prosty układ RC do wytworzenia końcowego sygnału głosowego.

Technika przetwarzania danych z użyciem wielu częstotliwości próbkowania jest cenna z dwóch powodów: 1) umożliwia zastąpienie podzespołów analogowych przez oprogramowanie, co jest oczywistą korzyścią ekonomiczną przy produkcji masowej oraz 2) umożliwia osiągnięcie lepszego działania w krytycznych zastosowaniach. Taka technika jest stosowana na przykład w odtwarzaczach płyt kompaktowych w celu uzyskania najlepszej możliwej jakości dźwięku. To udoskonalone działanie jest wynikiem zastąpienia elementów analogowych (dokładność 1%) algorytmami cyfrowymi (dokładność 0,0001% spowodowana błędami zaokrąglenia). Jak to wyniknie z rozważań w następnych rozdziałach, filtry cyfrowe w najważniejszych dziedzinach zastosowania działają *setki razy* lepiej niż filtry analogowe.

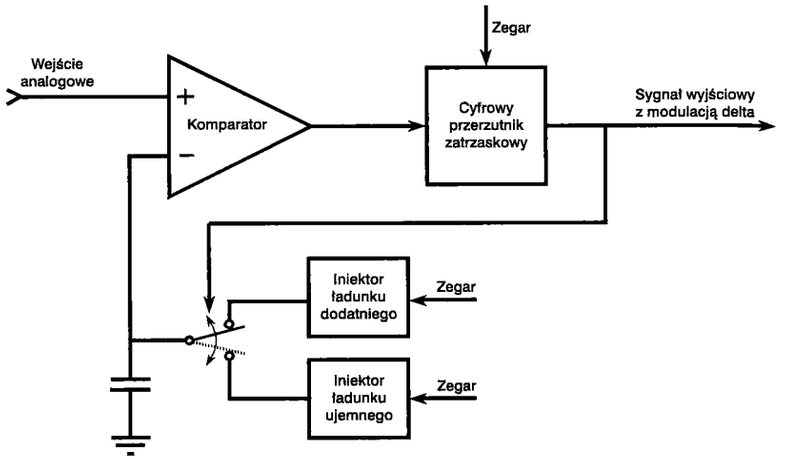
Jednobitowe przetwarzanie danych

Metodą często stosowaną w telekomunikacji i odtwarzaniu muzyki z bardzo dobrą wiernością jest **jednobitowe przetwarzanie a/c i c/a**. Jest to technika przetwarzania należąca do grupy metod o wielu częstotliwościach próbkowania, w której większą częstotliwość próbkowania osiąga się kosztem zmniejszenia liczby bitów. W skrajnym przypadku wystarcza tylko jeden bit na każdą próbkę. Jest wiele różnych rozwiązań układowych, lecz w większości z nich stosuje się **modulację delta**. Omówione dalej trzy przykłady rozwiązań będące dobrym wprowadzeniem w tę problematykę. Wszystkie trzy rozwiązania są zrealizowane jako układy scalone, nie trzeba się więc martwić o rozmieszczenie wszystkich tranzystorów i wzmacniaczy operacyjnych. Na pewno nikt was nie poprosi o skonstruowanie tych układów z elementów dyskretnych.

Na rys. 3.16 przedstawiono schemat blokowy typowego modulatora delta. Wejściowym sygnałem analogowym jest sygnał głosowy o amplitudzie kilku woltów, sygnałem wyjściowym zaś – sygnał cyfrowy będący strumieniem logicznych jedynek i zer. Komparator decyduje, które z dwóch napięć jest większe – wejściowy sygnał analogowy czy napięcie zapamiętane na kondensatorze. Ta decyzja, w postaci logicznego zera lub jedynki, jest wprowadzona na wejście przerzutnika zatrzaskowego. Za każdym impulsem zegarowym (typowo o częstotliwości kilkuset kHz) przerzutnik przenosi aktualny stan logiczny ze swojego wejścia na wyjście. Dzięki przerzutnikowi zatrzaskowemu wyjście jest zsynchronizowane z zegarem, wyznaczając w ten sposób szybkość próbkowania, tj. szybkość, z jaką może następować aktualizacja 1-bitowego wyjścia.

Cyfrowy sygnał wyjściowy steruje przełącznikiem elektronicznym, tworząc w ten sposób pętlę sprzężenia zwrotnego. Jeśli na wyjściu jest stan logiczny *jeden*, to przełącznik dołącza kondensator do *iniektora ładunku dodatniego*. Jest to układ, który zwiększa napięcie na kondensatorze o określoną wartość, np. o 1 mV na jeden impuls zegarowy. Może to być tylko rezystor dołączony do dużego napięcia dodatniego. Jeśli na wyjściu jest stan logiczny *zero*, to kondensator jest dołączany do *iniektora ładunku ujemnego*. Powoduje on *zmniejszanie* napięcia na kondensatorze o taką sama ustaloną wartość.

Na rys. 3.17 przedstawiono przebiegi sygnałów w tym układzie. W chwili zero zarówno wejściowy sygnał analogowy, jak i napięcie na kondensatorze są równe zeru. Jak pokazano na rys. 3.17 a), w chwili ósmego impulsu zegarowego napięcie wejściowe wzrasta nagle do 9,5 V.

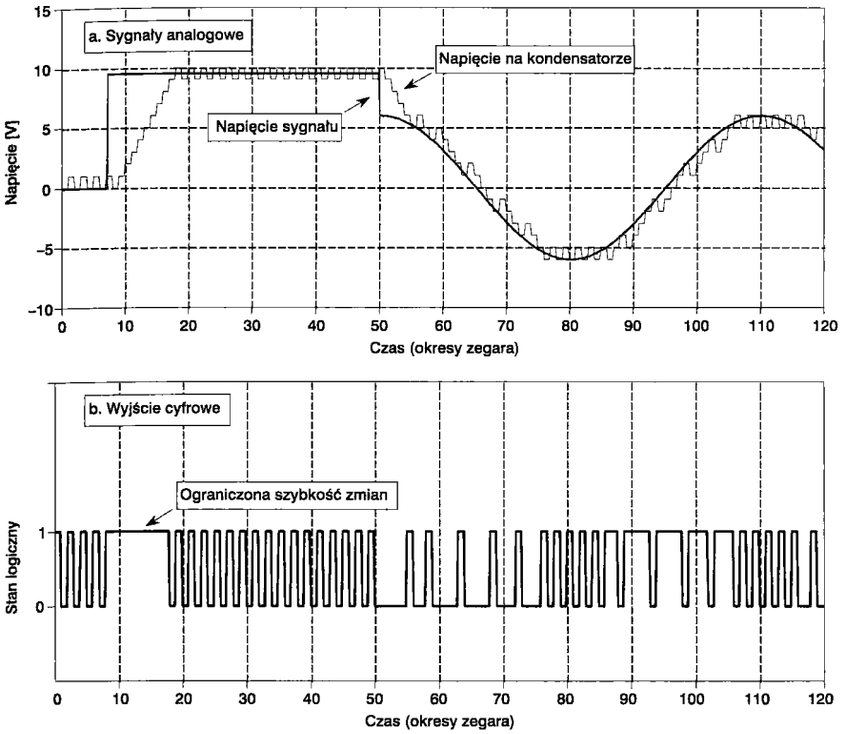


Rys. 3.16. Schemat blokowy modulatora delta. Napięcie wejściowe jest porównywane z napięciem zapamiętanym na kondensatorze. Zależnie od wyniku tego porównania do wejścia przerzutnika zatrzaskowego jest wprowadzany stan zero lub jeden. Stan wyjściowy przerzutnika jest aktualizowany synchronicznie z impulsami zegarowymi. Dzięki pętli sprzężenia zwrotnego napięcie na kondensatorze nadąża w ślad za napięciem wejściowym

Ponieważ sygnał wejściowy staje się wówczas większy od napięcia na kondensatorze, więc stan logiczny na wyjściu zmienia się na *jeden* (rys. 3.17 b). Kondensator zostaje wtedy dołączony do iniektora ładunku dodatniego i napięcie na nim wzrasta o niewielką wartość przy każdym impulsie zegarowym. Chociaż na rys. 3.17 a), jako przykład, pokazano wzrost o 1 V na jeden okres zegarowy, to bardziej typowa jest wartość 1 mV. Taki schodkowy wzrost napięcia na kondensatorze trwa tak długo, aż to napięcie stanie się większe od napięcia wejściowego. Wtedy system osiąga równowagę i sygnał wyjściowy oscyluje między stanami zero i jeden, powodując oscylacje napięcia na kondensatorze między wartościami 9 i 10 V. W ten sposób sprzężenie zwrotne wymusza, aby napięcie na kondensatorze nadążało w ślad za napięciem wejściowym. Jeśli sygnał wejściowy zmienia się bardzo gwałtownie, to napięcie na kondensatorze zmienia się ze stałą szybkością, aż do zrównania napięć. Ta stała szybkość zmiany napięcia jest w terminologii angielskiej nazywana **slew rate**, przez analogię do podobnych parametrów innych urządzeń elektronicznych, jak wzmacniacze operacyjne.

Zajmijmy się teraz właściwościami sygnału wyjściowego z modulacją delta. Gdy wartość wejściowego sygnału analogowego *wzrasta*, to sygnał wyjściowy zawiera więcej jedynek niż zer. Jeśli zaś ta wartość *maleje*, to w sygnale wyjściowym jest więcej zer niż jedynek. Jeśli sygnał wejściowy jest stały, to cyfrowy sygnał na wyjściu zmienia się między zerem i jedynką, przy czym zer jest tyle samo co jedynek. Uogólniając, można stwierdzić, że względna liczba zer w stosunku do jedynek jest wprost proporcjonalna do *nachylenia* (pochodnej) wejściowego sygnału analogowego.

Ten układ umożliwia tani sposób przetwarzania sygnału analogowego na szeregowy strumień jedynek i zer, który można przesyłać lub rejestrować w postaci cyfrowej. Szczególnie korzystną właściwością jest, że tu wszystkie bity mają takie samo znaczenie, w odróżnieniu od konwencjonalnego formatu szeregowego, gdzie mamy: *bit startu*, *LSB*, ..., *MSB*, *bit stopu*. W odbiorniku jest układ identyczny z układem w pętli sprzężenia w części nadawczej. Napięcie na kondensatorze w układzie odbiorczym, podobnie jak napięcie na kondensatorze w układzie nadawczym, nadąża w ślad za wejściowym napięciem analogowym. A więc przebieg napięcia na kondensatorze przedstawiony na rys. 3.17 a) wyraża również przebieg sygnału odtworzonego na wyjściu całego systemu.

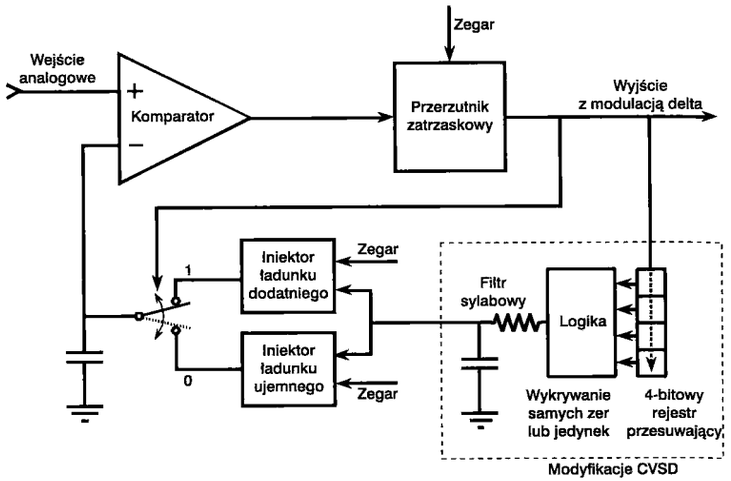


Rys. 3.17. Przykład sygnałów w modulatorze delta z rys. 3.16. Na rys. a) przedstawiono analogowy sygnał wejściowy i odpowiadające mu napięcie na kondensatorze. Na rys. b) pokazano sygnał wyjściowy z modulacją delta, będący strumieniem stanów logicznych zero i jeden

Krytycznym ograniczeniem jest w tym układzie konieczność kompromisu między: 1) maksymalną szybkością zmiany napięcia (*slew rate*), 2) wartością kroku kwantyzacji i 3) szybkością przekazywania danych. W szczególności, jeśli maksymalna szybkość zmiany napięcia oraz krok kwantyzacji dobrano według wymagań przesyłania sygnału głosowego, to przepływność bitowa nie przekracza zakresu megahercowego. Do celów komercyjnych jest to zbyt dużo. Na przykład konwencjonalne próbkowanie sygnału głosowego wymaga tylko 64 000 bit/s.

Rozwiązaniem tej trudności jest przedstawiony na rys. 3.18 modulator delta ze zboczem zmiennym w sposób ciągły (**CVSD** – *Continuosly Variable Slope Delta*). Tę technikę zastosowano w rodzinie układów MC3518 firmy Motorola. W tej metodzie częstotliwość zegara i krok kwantyzacji są ustawione na jakieś możliwe do przyjęcia wartości – powiedzmy 30 kHz i 2000 poziomów kwantyzacji. Daje to fatalną szybkość zmian napięcia na kondensatorze, którą koryguje się dodatkowymi układami. Rejestr przesuwający stale śledzi ostatnie cztery bity pojawiające się na wyjściu systemu. Jeśli układ jest w warunkach ograniczonej szybkości zmian napięcia, to ostanie cztery bity będą samymi jedynkami (zbocze dodatnie) lub samymi zerami (zbocze ujemne). Taką sytuacje wykrywają układy logiczne, wytwarzając sygnał analogowy, który zwiększa porcje ładunków generowanych w iniektorach ładunków. To powoduje zwiększenie szybkości zmian napięcia dzięki wzrostowi skoków napięcia na kondensatorze.

Filtr analogowy jest zwykle umieszczany między układami logicznymi a iniektorami ładunku. To powoduje zależność wartości skoku od czasu, w jakim układ był w warunkach ograniczenia szybkości zmian napięcia. Tak długo, jak układ ma tę szybkość ograniczoną,



Rys. 3.18. Schemat blokowy modulatora CVSD. Do podstawowego modulatora delta dodano układy logiczne w celu ulepszenia szybkości zmian napięcia

wartość skoku staje się coraz większa. Ten filtr jest często nazywany *sylabowym*, ponieważ jego właściwości zależą od średniej długości sylaby w sygnale mowy. Przy właściwej optymalizacji (zgodnie z zaleceniami producenta układu, a nie według naszych własnych pomysłów) częstotliwości przesyłania danych od 16 do 32 kHz dają sygnał mowy o akceptowalnej jakości. Ciągle zmieniająca się wartość skoku powoduje, że dane cyfrowe są trudne do zrozumienia, ale to zrozumienie, na szczęście, wcale nie jest potrzebne. Sygnał analogowy jest bowiem odtwarzany w odbiorniku, w którym jest umieszczony filtr sylabowy, identyczny z tym w nadajniku. Jeśli oba filtry są dopasowane do siebie, to zniekształcenia wywołane modulacją CVSD są niewielkie. Ta modulacja jest przypuszczalnie najłatwiejszym sposobem cyfrowego przesyłania sygnału głosowego.

Modulacja CVSD, choć świetna do kodowania sygnałów głosowych, nie może być stosowana do przetwarzania a/c ogólnego przeznaczenia. Jeśli nawet pominiemy fakt, że dane cyfrowe odnoszą się do *pochodnej* sygnału wejściowego, *zmieniający się skok kwantyzacji* wprowadzi zamieszanie niemożliwe do skorygowania. Ponadto, przy tej modulacji poziom stały sygnału wejściowego nie jest uwzględniony w uzyskiwanych danych cyfrowych.

Te trudności są wyeliminowane w przetworniku **delta-sigma** przedstawionym na rys. 3.19, w którym umiejętnie połączono elektronikę analogową z algorytmami DSP. Zauważmy, że napięcie na kondensatorze jest teraz porównywane z potencjałem masy. Zmodyfikowano również pętlę sprzężenia zwrotnego, tak aby napięcie na kondensatorze *zmniejszało się*, gdy na wyjściu układu jest stan logiczny *jeden* i *wzrastało* w przypadku stanu *zero*. Wzrastający i malejący wejściowy sygnał napięciowy próbuje wywołać wzrost lub zmniejszenie napięcia na kondensatorze. Ta zmiana napięcia jest wykrywana przez komparator, powodując, że iniektory generują ładunek *przeciwdziałający*, który utrzymuje napięcie na kondensatorze równe zeru.

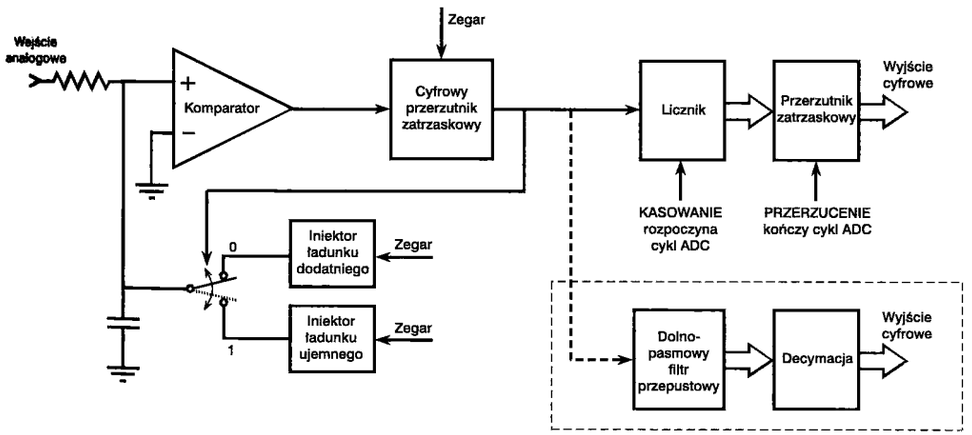
Jeśli napięcie wejściowe jest dodatnie, to w wyjściowym sygnale cyfrowym będzie więcej jedynek niż zer. Nadmiarowa liczba jedynek jest potrzebna do generowania *ujemnego* ładunku, który redukuje *dodatni* sygnał wejściowy. Podobnie, jeśli napięcie wejściowe jest ujemne, sygnał cyfrowy na wyjściu będzie zawierał więcej zer niż jedynek, powodując iniekcje wypadkowego ładunku dodatniego. Jeśli sygnał wejściowy jest równy zeru, to na wyjściu pojawiają się jednakowe liczby zer i jedynek, powodując zerową wypadkową iniekcję ładunku.

Względna liczba jedynek i zer na wyjściu jest teraz zależna od *poziomu* napięcia wejściowego, a nie od *nachylenia* jego zmian, jak to było w układzie poprzednio omówionym. Można na przykład utworzyć 12-bitowy przetwornik a/c, wprowadzając sygnał wyjściowy do licznika i wyznaczając liczbę *jedynek* w 4096 okresach zegarowych. Liczba 4095 będzie odpowiadała maksymalnemu dodatniemu napięciu wejściowemu. Liczba 0 będzie zaś odpowiadała maksymalnemu wejściowemu napięciu ujemnemu, a 2048 – napięciu wejściowemu równemu zero. Można teraz wyjaśnić pochodzenie nazwy metody *delta-sigma*: modulacja delta, po której następuje sumowanie (sigma).

Jedynki i zera generowane na wyjściu tego typu modulatora delta można łatwo przekształcić w sygnał analogowy. Do tego celu wystarczy analogowy filtr dolnoprzepustowy, którym może być prosty, pojedynczy obwód RC. Następuje w nim uśrednianie wysokich i niskich stanów napięciowych odpowiadających stanom logicznym 0 i 1, dając w wyniku poprawne napięcie analogowe. Załóżmy na przykład, że stany 1 i 0 są wyrażane napięciami 5 V i 0 V. Jeśli w strumieniu danych jest 80% *jedynek* a 20% *zer*, to na wyjściu filtru dolnoprzepustowego będzie napięcie 4 V.

Ta metoda przekształcania jednobitowego strumienia danych z powrotem w przebieg oryginalny jest z kilku przyczyn bardzo ważna. Po pierwsze, wynika z niej zgrabny sposób zastąpienia licznika w przetworniku a/c delta-sigma. Zamiast prostego zliczania impulsów z modulatora delta, sygnał binarny jest przepuszczany przez *cyfrowy* filtr dolnoprzepustowy i następnie poddawany *decymacji* w celu zredukowania częstotliwości próbkowania. Ta procedura może się zacząć na przykład od zamiany na 12-bitowe próbki każdego zera i jedynki w strumieniu cyfrowym. Jedynki stają się próbkami o wartości 4095, a zera – o wartości 0. Taki sygnał, poddany dolnoprzepustowej filtracji cyfrowej, daje cyfrową wersję sygnału oryginalnego, podobnie jak dolnoprzepustowy filtr analogowy dałby rekonstrukcję analogową. Decymacja redukuje następnie częstotliwość próbkowania przez odrzucenie większości próbek. W rezultacie uzyskuje się sygnał cyfrowy równoważny uzyskiwanemu przez bezpośrednie próbkowanie przebiegu oryginalnego.

Ta metoda jest stosowana w wielu dostępnych na rynku przetwornikach a/c służących do digitalizacji głosu i innych sygnałów fonicznych. Przykładem takiego 16-bitowego przetwornika a/c jest układ ADC16071 firmy National Semiconductor o częstotliwości prób-



Rys. 3.19. Schemat blokowy przetwornika analogowo-cyfrowego delta-sigma. W najprostszym przypadku impulsy z modulatora delta są zliczane w ciągu ustalonej liczby okresów zegarowych. Wyjście licznika przerzuca przerzutnik, kończąc przetwarzanie. W bardziej rozbudowanym układzie impulsy są przepuszczane przez cyfrowy filtr dolnoprzepustowy i następnie poddawane decymacji zmniejszającej częstotliwość próbkowania

kowania do 192 kHz. Przy częstotliwości próbkowania 100 kHz modulator delta pracuje z częstotliwością zegarową 6,4 MHz. Dolnoprzepustowy filtr cyfrowy jest 246-puntowym filtrem o skończonej odpowiedzi impulsowej (SOI), taki jak opisany w rozdziale 16. Filtr usuwa w danych cyfrowych wszystkie częstotliwości powyżej 50 kHz, a więc powyżej połowy częstotliwości próbkowania. Można na to spojrzeć jako na tworzenie sygnału cyfrowego z częstotliwością 6,4 MHz, w którym każda próbka jest reprezentowana przez 16 bitów. Ten sygnał podlega następnie decymacji od 64 MHz do 100 kHz, wykonywanej przez usuwanie każdych 63 z 64 próbek. W rzeczywistości, wewnątrz tego układu dzieje się znacznie więcej, niż tu w skrócie opisaliśmy.

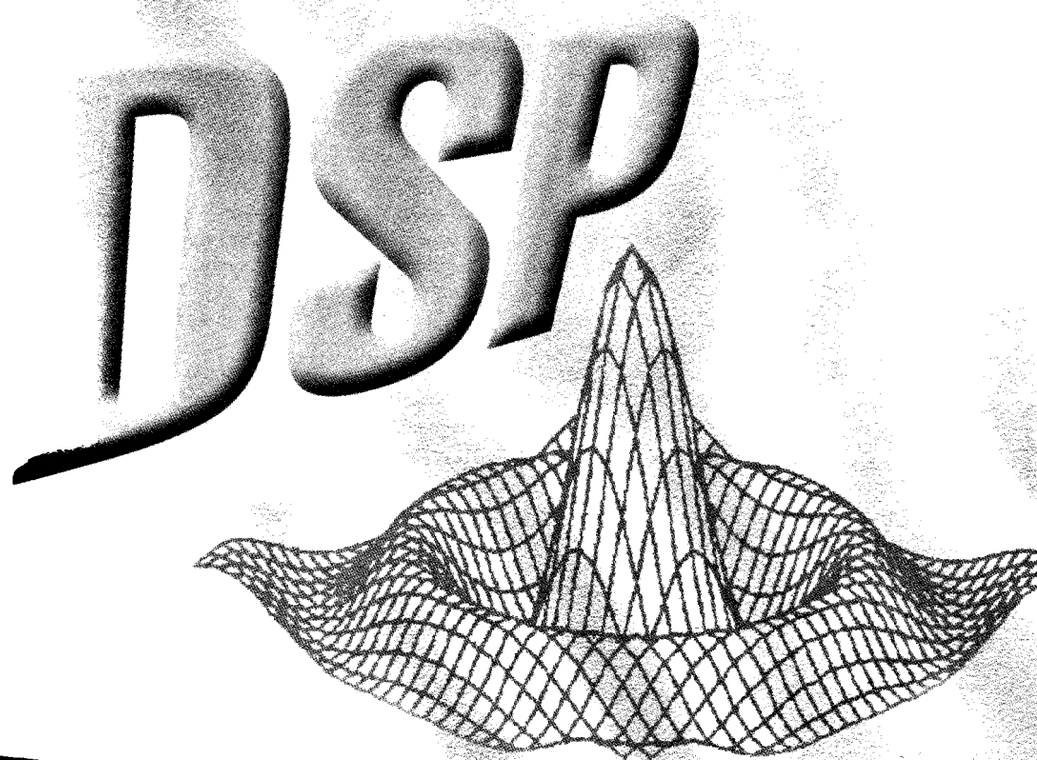
Przetworniki delta-sigma mogą być również używane do przetwarzania cyfrowo-analogowego sygnałów głosowych i innych fonicznych. Sygnał cyfrowy jest pobierany z pamięci i przetwarzany na strumień zer i jedynek z modulacją delta. Jak już wspomniano, taki sygnał jednobitowy można łatwo przekształcić w zrekonstruowany sygnał analogowy, stosując prosty dolnoprzepustowy filtr analogowy. Tak jak przy filtrze antyaliasingowym, zwykle jest wymagany tylko prosty układ RC, gdyż większość filtracji wykonują filtry cyfrowe o bardzo dobrych parametrach.

Przetworniki a/c delta-sigma mają kilka cech, które ograniczają ich zastosowanie do specyficznych zastosowań. Na przykład trudne jest multipleksowanie wejść tych przetworników. Po przełączeniu wejścia od jednego sygnału do innego, właściwe działanie ustala się dopiero, gdy filtr cyfrowy oczyści się z danych poprzedniego sygnału. Przetworniki delta-sigma są ograniczone jeszcze pod innym względem: nigdy nie wiadomo dokładnie, *kiedy* każda próbka jest pobierana. Każda pobrana próbka jest bowiem składnikiem informacji jednobitowej pobranej z pewnego segmentu sygnału wejściowego. Nie jest to problemem w przypadku sygnałów kodowanych w dziedzinie częstotliwości, takich jak sygnały foniczne, lecz jest poważnym ograniczeniem dla sygnałów kodowanych w dziedzinie czasu. W celu zrozumienia kształtu przebiegu sygnału, trzeba często dokładnie znać chwilę, w której została pobrana każda z próbek. I na koniec trzeba podkreślić, że większość układów delta-sigma jest specjalnie zaprojektowanych do zastosowań audio i zgodnie z tymi wymaganiami są definiowane parametry układów. Na przykład w 16-bitowym przetworniku a/c stosowanym do sygnałów głosowych każda próbka niekoniecznie musi mieć 16-bitową dokładność. Znacznie bardziej prawdopodobne jest stwierdzenie producenta, że *sygnały głosowe* mogą być przetwarzane na cyfrowe z 16-bitowym *zakresem dynamicznym*. Nie oczekujmy od tego układu uzyskania pełnych 16 bitów użytecznej informacji do akwizycji danych ogólnego przeznaczenia.

Powyższe wyjaśnienie działania i przykłady są wstępem do jednobitowych przetworników a/c i c/a. Trzeba podkreślić, że są to jedynie uproszczone opisy skomplikowanych systemów DSP i techniki układów scalonych. Nie możemy się spodziewać, aby wytwórcy zdradzili *konkurentom* sekrety swych opracowań, a więc nie powiedzą tego także Wam.

4

Oprogramowanie DSP



Zadania realizowane z wykorzystaniem DSP są na ogół programowane w tych samych językach, co inne zadania naukowe i techniczne np. w języku C, BASIC i asemblerze. Uniwersalność i możliwości języka C powodują, że on jest zwykle wybierany przez naukowców z dziedziny techniki komputerowej i przez innych programistów profesjonalnych. Z drugiej zaś strony prostota BASIC-a czyni go świetnym wyborem dla naukowców i inżynierów, którzy tylko okazjonalnie odwiedzają świat oprogramowania. Niezależnie od używanego języka, większość ważnych zagadnień związanych z programowaniem DSP jest na poziomie znacznie niższym – w sferze „wirujących" zer i jedynek. Są to m.in. takie problemy jak: sposoby wyrażania liczb przez zestawy bitów, błędy zaokrąglania w arytmetyce komputerowej, szybkość obliczeniowa różnych typów procesorów. Ten rozdział dotyczy kwestii, które możecie rozwiązywać na *wysokim poziomie*, aby uniknąć stłamszenia przez wewnętrzne problemy waszego komputera na *niskim poziomie*.

Liczby w komputerze

Komputery cyfrowe są bardzo sprawne w zapisywaniu i przywoływaniu liczb. Niestety ten proces nie odbywa się bez błędów. Na przykład wydajecie polecenie swojemu komputerowi, aby zapisał liczbę 1,41421356. Komputer wykonuje to polecenie najlepiej, jak umie i zapisuje najbliższą liczbę, jaką jest zdolny wyrazić: 1,41421354. W niektórych przypadkach taki błąd nie ma żadnego znaczenia, podczas gdy w innych jest fatalny. Inną ilustracją problemu są klasyczne błędy obliczeniowe wynikające z dodawania dwóch liczb o wartościach bardzo się od siebie różniących – np. 1 i 0,00000001. Chcielibyśmy, aby wynik był 1,00000001, ale komputer odpowiada: 1. Zrozumienie, w jaki sposób komputer zapisuje liczby i nimi operuje, pozwala przewidzieć i skorygować te trudności, *zanim* program wypluje z siebie bezsensowne dane.

Takie kłopoty powstają, ponieważ do zapisania każdej liczby jest przydzielona pewna ustalona liczba bitów, zwykle 8, 16, 32 lub 64. Rozważmy np. przypadek, gdy 8 bitów jest używanych do zapisania wartości pewnej zmiennej. Ponieważ jest $2^8 = 256$ możliwych sekwencji bitów, więc zmienna może przyjmować tylko 256 różnych wartości. Jest to fundamentalne ograniczenie, na które nic nie możemy poradzić. *Możemy* tylko decydować, jaką wartość każda sekwencja bitów ma reprezentować. W najprostszych przypadkach 256 sekwencji bitów może reprezentować liczby całkowite od 0 do 255, od 1 do 256, od –127 do 128 itd. W rzadziej spotykanym rozwiązaniu 256 sekwencji bitów może reprezentować 256 liczb powiązanych ze sobą wykładniczo: 1, 10, 100, 1000, ... , 10^{254}, 10^{255}. Każdy mający dostęp do danych powinien zdawać sobie sprawę, jaką wartość reprezentuje każda sekwencja bitów. Do tego celu służy zazwyczaj jakiś algorytm lub wzór umożliwiający konwersję między wartością reprezentowaną a odpowiadającą jej sekwencją bitów, i na odwrót.

Chociaż jest wiele możliwych sposobów kodowania, to tylko dwa formaty zapisu są w powszechnym użyciu: *stałoprzecinkowy* (zwany też zapisem liczb całkowitych) oraz *zmiennoprzecinkowy* (zwany też zapisem liczb rzeczywistych). W tej książce w programach w BASIC-u, zmienne stałoprzecinkowe są zaznaczane symbolem % jako ostatni znak w nazwie – np. I%, N%, SUM% itd. Wszystkie inne zmienne są wartościami zmiennoprzecinkowymi, np. X,Y, MEAN itd. Oceniając formaty przedstawione na następnych kilku stronach, spróbujcie rozważać je pod względem **zakresu** (najmniejsza i największa liczba, którą mogą reprezentować) oraz **dokładności** (rozmiar przerw między liczbami).

Zapis stałoprzecinkowy (liczby całkowite)

Zapis stałoprzecinkowy jest stosowany do wyrażania *liczb całkowitych*, dodatnich i ujemnych: ... –3, –2, –1, 0, 1, 2, 3, ... W programach w językach wysokiego poziomu, takich jak C i BASIC, zwykle jest przydzielonych 16 bitów do zapisu każdej liczby całkowitej. W najprostszym przypadku $2^{16} = 65\,536$ możliwych sekwencji bitów jest przypisanych liczbom od 0 do 65 535. Jest to zapis **liczb całkowitych bez znaku** (*unsigned integer*). Jego uproszczony przykład podano w tablicy na rys. 4.1 (z użyciem tylko 4 bitów). Konwersja między sekwencją bitów a liczbą reprezentowaną przez tę sekwencję polega po prostu na zamianie podstawy 2 (kod binarny) na podstawę 10 (kod dziesiętny). Wadą tego zapisu jest brak możliwości reprezentacji liczb ujemnych.

Zapis binarny przesunięty (*offset binary*) jest podobny do zapisu liczb całkowitych bez znaku, z tą różnicą, że wartości dziesiętne są *przesunięte*, co umożliwia zapis także liczb ujemnych. W 4-bitowym przykładzie w tablicy na rys. 4.1 liczby dziesiętne są przesunięte o *siedem*, co daje 16 sekwencji bitów reprezentujących liczby od –7 do 8. W ten sam sposób w zapisie 16-bitowym przesunięcie 32 767 daje zakres od –32 767 do 32 768. Zapis binarny przesunięty nie jest jedynym standardem, istnieją inne zapisy z przesunięciem. Najważniejszą dziedziną zastosowania zapisu binarnego przesuniętego są przetworniki a/c i c/a. Na przykład zakres napięcia wejściowego od –5 V do 5 V może być odwzorowany przez liczby od 0 do 4095 przy przetwarzaniu 12-bitowym.

Zapis znak-moduł jest innym prostym sposobem reprezentacji ujemnych liczb całkowitych. Pierwszy bit z lewej strony jest nazywany **bitem znaku** i zawiera *zero* dla liczb dodatnich i *jeden* dla ujemnych. Pozostałe bity są standardową binarną reprezentacją bezwzględnych wartości liczb. W ten sposób traci się jedną sekwencję bitów, gdyż są dwie reprezentacje zera, 0000 (zero dodatnie) i 1000 (zero ujemne). Zapis 16-bitowy znak-moduł daje reprezentację liczb od –32 767 do 32 767.

Te trzy wymienione sposoby zapisu są koncepcyjnie proste, lecz trudne do realizacji układowej. Pamiętajmy, że gdy wyrażenie A = B + C jest wprowadzane do programu komputerowego, to jakiś inżynier musi wymyślić, jak utworzyć sekwencję bitów reprezentującą B, połączyć go z sekwencją bitów wyrażającą C, aby uzyskać sekwencję bitów reprezentującą A.

Zapis uzupełnienia do 2 jest szczególnie lubiany przez inżynierów specjalizujących się w układach elektronicznych. W komputerach liczby całkowite są zwykle reprezentowane w tym zapisie. Aby zrozumieć sposób kodowania, spójrzmy najpierw w tablicy na rys. 4.1 na liczbę dziesiętną zero, która odpowiada zeru binarnemu 0000. Spojrzawszy wyżej, widzimy, że liczby dziesiętne są reprezentowane przez ich binarne równoważniki (0 = 0000, 1 = 0001, 2 = 0010, 3 = 0011 itd.). Przypomnijmy sobie teraz, że te cztery bity są zapisywane w rejestrze złożonym z 4 przerzutników. Jeśli znowu zaczniemy od 0000 i rozpoczniemy odejmowanie, to układy cy-

Liczba całkowita bez znaku		Binarny przesunięty		Znak-moduł		Uzupełnienie do 2	
Liczba dziesiętna	Sekwencja bitów	Liczba dziesiętna	Sekwencja bitów	Liczba dziesiętna	Sekwencja bitów	Liczba dziesiętna	Sekwencja bitów
15	1111	8	1111	7	0111	7	0111
14	1110	7	1110	6	0110	6	0110
13	1101	6	1101	5	0101	5	0101
12	1100	5	1100	4	0100	4	0100
11	1011	4	1011	3	0011	3	0011
10	1010	3	1010	2	0010	2	0010
9	1001	2	1001	1	0001	1	0001
8	1000	1	1000	0	0000	0	0000
7	0111	0	0111	0	1000	–1	1111
6	0110	–1	0110	–1	1001	–2	1110
5	0101	–2	0101	–2	1010	–3	1101
4	0100	–3	0100	–3	1011	–4	1100
3	0011	–4	0011	–4	1100	–5	1011
2	0010	–5	0010	–5	1101	–6	1010
1	0001	–6	0001	–6	1110	–7	1001
0	0000	–7	0000	–7	1111	–8	1000
Zakres 16-bitowy: 0 do 65535		Zakres 16-bitowy: –32767 do 32768		Zakres 16-bitowy: –32767 do 32767		Zakres 16-bitowy: –32768 do 32767	

Rys. 4.1. Najczęściej stosowane sposoby zapisu stałoprzecinkowego (liczby całkowite). Zapis liczby całkowitej bez znaku jest prostym formatem binarnym, lecz nie może reprezentować liczb ujemnych. Zapisy binarny przesunięty oraz znak-moduł umożliwiają reprezentację liczb ujemnych, lecz są trudne do realizacji w układach elektronicznych. Zapis uzupełnienia do 2, najłatwiejszy do realizacji układowej, jest najczęściej stosowanym sposobem zapisu w obliczeniach ogólnego przeznaczenia

frowe automatycznie obliczają uzupełnienia do 2: 0 = 0000, –1 = 1111, –2 = 1110, –3 = 1101 itd. Jest tu analogia do hodometru w nowym samochodzie. Przy jeździe do przodu daje on odczyty: 00000, 00001, 00002, 00003, itd., a przy jeździe do tyłu: 00000, 99999, 99998, 99997 itd.

W zapisie uzupełnienia do 2 przy użyciu 16 bitów są reprezentowane liczby od –32 768 do 32 767. Pierwszy bit z lewej strony jest 0, gdy liczba jest dodatnia lub równa zero, albo 1, gdy liczba jest ujemna. Dlatego pierwszy bit z lewej strony jest nazywany **bitem znaku**, podobnie jak w zapisie znak-moduł. Konwersja między liczbami dziesiętnymi a ich reprezentacją w zapisie uzupełnienia do 2 jest bezpośrednia dla liczb dodatnich – po prostu przejście od liczby dziesiętnej do binarnej. Dla liczb ujemnych często stosuje się następujący algorytm: 1) należy wziąć wartość bezwzględną liczby dziesiętnej, 2) przekształcić ją na wartość binarną, 3) odwrócić stany wszystkich bitów (czyli wyznaczyć komplement – tzn. wszystkie zera zamienić na jedynki, a jedynki na zera), 4) do uzyskanej liczby binarnej dodać 1. Na przykład: –5 → 5 → 0101 → 1010 → 1011. Zapis uzupełnienia do 2 jest trudny dla człowieka, łatwy zaś dla komputera.

Zapis zmiennoprzecinkowy (liczby rzeczywiste)

Sposób kodowania liczb zmiennoprzecinkowych jest bardziej skomplikowany niż stałoprzecinkowych. Podstawowa zasada jest taka sama, jak w zapisie wykładniczym, gdzie **mantysa** jest mnożona przez 10 podniesione do **wykładnika potęgi**. Na przykład 5,4321 × 10^6, gdzie 5,4321 jest *mantysą*, a 6 *wykładnikiem* potęgi. Zapis wykładniczy doskonale służy do reprezentacji liczb bardzo dużych i bardzo małych. Na przykład: 1,2 × 10^{50} – liczba atomów w kuli ziemskiej lub 2,6 × 10^{-23} – stosunek drogi przebywanej przez żółwia w ciągu jednej sekundy do średnicy naszej Galaktyki. Zauważmy, że liczby w zapisie wykładniczym są *znormalizowane* w taki sposób, że na lewo od przecinka dziesiętnego jest tylko jedna cyfra różna od zera. Osiąga się to przez odpowiedni dobór wykładnika.

Zapis zmiennoprzecinkowy jest podobny do wykładniczego z tą wszakże różnicą, że stosuje się w nim podstawę 2, a nie 10. Jest w użyciu kilka podobnych do siebie formatów zmiennoprzecinkowych. Najpopularniejszy jest format objęty normą ANSI/IEEE Std. 754-1985. Ta norma definiuje format dla liczb 32-bitowych, tzw. o **pojedynczej dokładności**, a także dla 64-bitowych (o **podwójnej dokładności**). Jak wynika z rys. 4.2, 32 bity stosowane w zapisie o pojedynczej dokładności są podzielone na trzy osobne grupy: bity od 0 do 22 tworzą mantysę, od 23 do 30 – wykładnik, a bit 31 jest bitem znaku. Te bity tworzą liczbę zmiennoprzecinkową v według zależności:

$$v = (-1)^S \times M \times 2^{E-127}$$

Wzór 4.1. Wzór do konwersji sekwencji bitów na liczbę zmiennoprzecinkową. Liczba jest wyrażona przez v, S jest wartością bitu znaku, M – wartością mantysy, a E – wykładnika

Wyrażenie $(-1)^S$ oznacza po prostu, że bit znaku S jest 0 dla liczby dodatniej i 1 dla ujemnej. Zmienna E jest liczbą między 0 i 255 wyrażoną przez 8 bitów wykładnika. Odejmując od tej liczby 127, uzyskuje się czynnik wykładniczy o zakresie od 2^{-127} do 2^{128}. Tak więc wykładnik jest wyrażony w zapisie *binarnym przesuniętym* z przesunięciem 127.

Mantysa jest tworzona z 23 bitów jako *ułamek binarny*. Na przykład ułamek dziesiętny 2,783 interpretuje się jako 2 + 7/10 + 8/100 + 3/1000. Ułamek binarny 1,0101 oznacza: 1 + 0/2 + 1/4 + 0/8 + 1/16. Liczby zmiennoprzecinkowe są *znormalizowane* w taki sam sposób jak w zapisie wykładniczym, tzn. jest tylko jedna cyfra niezerowa na lewo od przecinka dziesiętnego (w kodzie binarnym zwanego *przecinkiem binarnym*). Ponieważ jedyną cyfrą niezerową w kodzie binarnym jest 1, więc pierwszą cyfrą w mantysie zawsze będzie 1 i dlatego ta liczba nie musi być zapisywana. Usunięcie tej redundancji umożliwia poprawę dokładności liczby o 1 bit. Mantysę tworzą 23 bity: m_{22}, m_{21}, m_{20}, ..., m_0, zgodnie z wyrażeniem

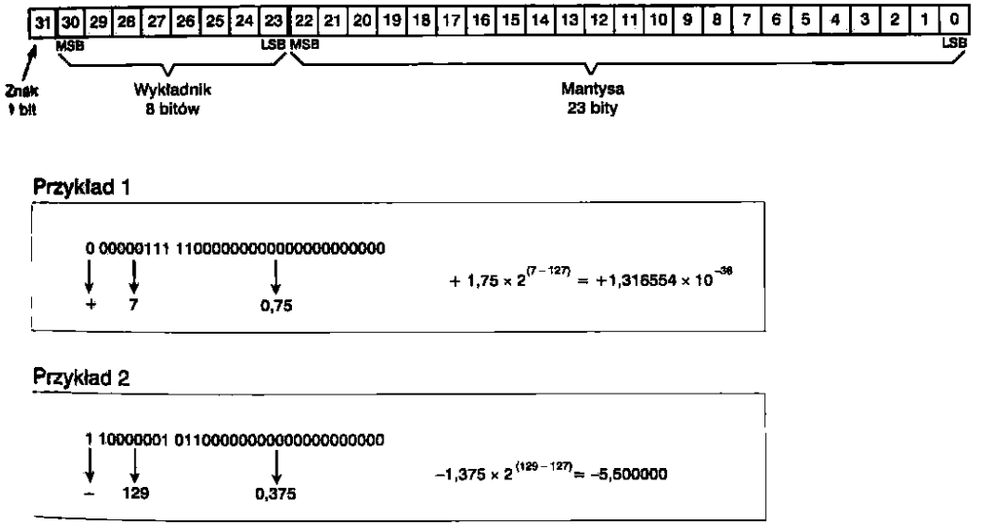
$$M = 1, m_{22}m_{21}m_{20}m_{19}...m_2m_1m_0$$

Wzór 4.2. Algorytm konwersji sekwencji bitów na mantysę M, zawartą we wzorze 4.1

W inny sposób można napisać: $M = 1 + m_{22}2^{-1} + m_{21}2^{-2} + m_{20}2^{-3}$ ... Jeśli bity od 0 do 22 mają wszystkie wartość *zero*, to M przyjmuje wartość 1. Jeśli wszystkie te bity mają wartość *jeden*, to M jest bardzo bliskie 2, tzn. $2 - 2^{-23}$.

Przy użyciu tego sposobu kodowania największa możliwa reprezentowana liczba jest: $\pm(2 - 2^{-23}) \times 2^{128} = \pm 6{,}8 \times 10^{38}$. Podobnie, najmniejsza liczba jest: $\pm 1{,}0 \times 2^{-127} = \pm 5{,}9 \times 10^{-39}$. Norma IEEE nieco ogranicza ten zakres liczb, aby zwolnić niektóre sekwencje bitów i przyznać im specjalne znaczenia. Najmniejsza i największa liczby dopuszczalne w normie są, odpowiednio: $\pm 3{,}4 \times 10^{38}$ oraz $\pm 1{,}2 \times 10^{-38}$. Zwolnione sekwencje bitów są przeznaczone dla trzech liczb klasy specjalnej: 1) ±0, gdy wszystkie bity mantysy i wykładnika mają wartość zero, 2) ±∞, gdy wszystkie bity mantysy mają wartość zero, a wszystkie bity wykładnika – jeden, 3) grupa bardzo małych, *nienormalizowanych* liczb między $\pm 1{,}2 \times 10^{-38}$ i $\pm 1{,}4 \times 10^{-45}$. Są to liczby o mniejszej dokładności, uzyskane przez rezygnację z wymogu, żeby pierwsza cyfra mantysy była 1. Oprócz tych trzech klas specjalnych są jeszcze sekwencje bitów, którym nie przedzielono znaczenia, zwykle nazywane NAN (*Not A Number*, nie liczba).

W normie IEEE o zapisie z podwójną dokładnością po prostu dodano więcej bitów zarówno do mantysy, jak i do wykładnika. Z 64 bitów stosowanych do zapisu liczby o podwójnej dokładności, bity od 0 do 51 tworzą mantysę, od 52 do 62 – wykładnik, a bit 63 jest bitem znaku. Jak poprzednio, mantysa mieści się w zakresie od 1 do prawie 2, tj. $M = 1 + m_{51}2^{-1} + m_{50}2^{-2} + m_{49}2^{-3}$... Jedenaście bitów wykładnika tworzy liczbę między 0 a 2047, z przesunięciem 1023, co daje wykładniki od 2^{-1023} do 2^{1024}. W tym zapisie najmniejsza i największa możliwe liczby to $\pm 1{,}8 \times 10^{308}$ i $\pm 2{,}2 \times 10^{-308}$. Są to liczby niewiarygodnie duże i małe! Niewiele jest zastosowań, w których pojedyncza dokładność nie byłaby wystarczająca, a przypuszczalnie nigdy nie spotkacie się z takim przypadkiem, w którym podwójna dokładność ograniczałaby wasze zamierzenia.



Rys. 4.2. Format zapisu zmiennoprzecinkowego o pojedynczej dokładności. Słowo 32-bitowe jest podzielone na trzy osobne grupy bitów: bit znaku, wykładnik i mantysę. We wzorach 4.1 i 4.2 podano, jak na podstawie tych grup bitów można znaleźć liczby przez nie reprezentowane. MSB i LSB oznaczają, odpowiednio, najbardziej i najmniej znaczący bit

Dokładność liczb

Błędy związane z reprezentacją liczb są bardzo podobne do błędów kwantyzacji powstających podczas przetwarzania analogowo-cyfrowego. *Chcemy* zapisywać ciągły przedział wartości, jednak *możemy* posługiwać się tylko skończoną liczbą skwantowanych poziomów. Za każdym razem, gdy pojawia się nowa liczba, na przykład w wyniku operacji matematycznych, musi ona zostać zaokrąglona do najbliższej wartości, która może być zapisana w zastosowanym formacie.

Jako przykład wyobraźmy sobie, że do zapisania liczby są przeznaczone 32 bity. Ponieważ jest dokładnie 2^{32} = 4 294 967 296 różnych możliwych sekwencji bitów, więc można wyrazić dokładnie 4 294 967 296 różnych liczb. W niektórych językach programowania jest dopuszczalna zmienna o nazwie **długa liczba całkowita** (*long integer*), zapisywana, jako stałoprzecinkowa, w 32 bitach w zapisie uzupełnienia do 2. Wynika stąd, że 4 294 967 296 możliwych sekwencji bitów reprezentuje liczby całkowite między –2 147 483 648 i 2 147 483 647. Warto wspomnieć, dla porównania, że przy zapisie zmiennoprzecinkowym o pojedynczej dokładności te 4 294 967 296 sekwencji bitów reprezentuje liczby w znacznie szerszym zakresie: od $-3{,}4 \times 10^{38}$ do $3{,}4 \times 10^{38}$.

W przypadku zmiennych stałoprzecinkowych odstępy między sąsiednimi liczbami są zawsze równe *dokładnie jeden*. W zapisie zmiennoprzecinkowym te odstępy zmieniają się zależnie od zakresu reprezentowanych liczb. Jeśli losowo wybierzemy jakąś liczbę zmiennoprzecinkową, to odstęp za tą liczbą jest w przybliżeniu *dziesięć milionów razy mniejszy* niż sama liczba (dokładnie stanowi od 2^{-24} do 2^{-23} tej liczby). Jest to kluczowa koncepcja zapisu zmiennoprzecinkowego – duże liczby mają duże odstępy między sobą, a małe – małe odstępy. Zilustrowano to na rys. 4.3, pokazując kolejne liczby zmiennoprzecinkowe i odstępy między nimi.

```
0,00001233862713
0,00001233862804   Odstęp = 0,00000000000091
0,00001233862895   (jedna 13-milionowa)
0,00001233862986
      ⋮

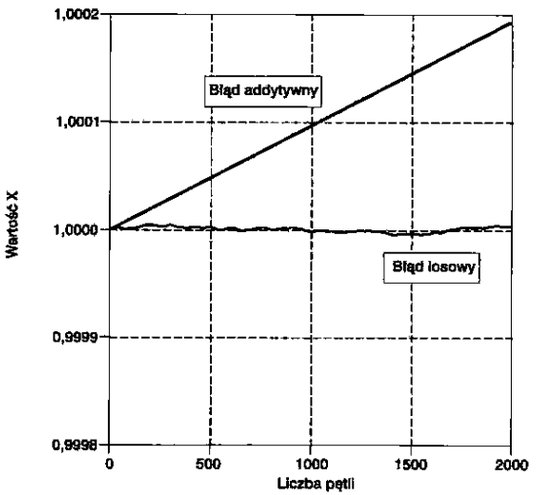
1,000000000
1,000000119        Odstęp = 0,000000119
1,000000238        (jedna 8-milionowa)
1,000000358
      ⋮

1,996093750
1,996093869        Odstęp = 0,000000119
1,996093988        (jedna 17-milionowa)
1,996094108
      ⋮

636,0312500
636,0313110        Odstęp = 0,0000610
636,0313720        (jedna 10-milionowa)
636,0314331
      ⋮

217063424,0
217063440,0        Odstęp = 16,0
217063456,0        (jedna 14-milionowa)
217063472,0
```

Rys. 4.3. Przykłady odstępów między liczbami zmiennoprzecinkowymi o pojedynczej dokładności. Odstęp między sąsiednimi liczbami zawsze mieści się od jednej 8-milionowej do jednej 17-milionowej części liczby



Rys. 4.4. Akumulacja błędu zaokrąglenia w zmiennych zmiennoprzecinkowych. Podane krzywe wygenerowano przy użyciu programu przedstawionego w tablicy 4.1. Gdy zmienna zmiennoprzecinkowa jest wielokrotnie stosowana w operacjach arytmetycznych, akumulowany błąd zaokrąglenia powoduje dryf wartości zmiennej. Jeśli błędy są zarówno dodatnie, jak i ujemne, to wartość zmiennej będzie losowo wzrastała i się zmniejszała. Jeśli zaś przeważają błędy zaokrąglenia tego samego znaku, to wartość będzie się zmieniała szybciej i bardziej równomiernie

Program przedstawiony w tablicy 4.1 ilustruje, jakie problemy w DSP powoduje **błąd zaokrąglenia** (błąd kwantyzacji w obliczeniach matematycznych). W pętli programowej dwie liczby losowe są dodawane do zmiennej zmiennoprzecinkowej X i potem od niej odejmowane. W idealnym przypadku *nic* się nie powinno zdarzyć. W rzeczywistości błąd zaokrąglenia w każdej operacji matematycznej powoduje stopniowe dryfowanie wartości X od jej wartości początkowej. Ten dryf może przyjmować jedną z dwóch form, zależnie od sposobu sumowania się błędów. Jeśli błędy zaokrąglenia są, losowo, dodatnie i ujemne, to wartość zmiennej będzie losowo wzrastać i się zmniejszać. Jeśli błędy są w głównej mierze jednego znaku, to wartość zmiennej będzie dryfować szybciej i bardziej równomiernie.

Tab. 4.1. Program ilustrujący akumulację błędów w przypadku stosowania liczb zmiennoprzecinkowych. Początkowo program ustawia wartość X na 1,000000 i następnie przebiega przez pętlę, która w przypadku idealnym nie powinna wykonywać niczego. Podczas każdej pętli są do X dodawane dwie liczby losowe A i B, które są znowu następnie odejmowane. Zakumulowany błąd tych dodawań i odejmowań powoduje, że X odbiega od swojej początkowej wartości. Jak pokazano na rys. 4.4, błąd może być losowy lub addytywny.

```
100 X= 1                                    'inicjalizuj X
110 '
120 FOR I% = 0 TO 2000
130    A = RND                              'ładuj liczby losowe
140    B = RND                              'do A i B
150    '
160    X = X + A                            'dodaj A i B do X
170    X = X + B
180    X = X - A                            'cofnij te dodawania
190    X = X - B
200    '
210    PRINT X                              'idealnie X powinno być 1
220 NEXT I%
230 END
```

Na rys. 4.4 przedstawiono, jak wartość zmiennej X dryfuje w tym programie podanym jako przykład. Jest oczywiste, że *błąd addytywny* jest znacznie gorszy niż *błąd losowy*. Błędy losowe mają bowiem tendencję do redukowania się wzajemnie, podczas gdy błędy addytywne po prostu się akumulują. Błąd addytywny jest w przybliżeniu równy błędowi zaokrąglenia pojedynczej operacji pomnożonemu przez całkowitą liczbę operacji. Błąd losowy zaś wzrasta tylko proporcjonalnie do *pierwiastka kwadratowego* liczby operacji. Jak pokazano w przykładzie, błąd addytywny może być setki razy większy niż błąd losowy w powszechnie stosowanych algorytmach DSP.

Niestety prawie niemożliwe jest przewidzieć, które z tych dwóch możliwych zachowań wystąpi w danym konkretnym algorytmie lub sterować trybem akumulacji błędów. Na przykład program podany w tablicy 4.1 generuje błąd addytywny. Można go zmienić na losowy przez wprowadzenie niewielkiej modyfikacji liczb dodawanych i odejmowanych. W ułamku właśnie sposób uzyskano krzywą błędu losowego (rys. 4.4), definiując: $A = EXP(RND)$ i $B = EXP(RND)$ zamiast $A = RND$ i $B = RND$. Liczby losowe A i B między 0 i 1 zastąpiono wartościami o rozkładzie wykładniczym leżącymi między 1 i 2,718. Nawet ta mała zmiana wystarcza, aby zmienić tryb akumulacji błędów.

Ponieważ nie możemy sterować sposobem akumulacji błędów zaokrąglania, więc uwzględniajmy najgorszy scenariusz. Oczekujmy, że każda liczba o pojedynczej dokładności będzie charakteryzowała się błędem równym około *jednej czterdziestomilionowej pomnożonym przez liczbę operacji, które na tej liczbie przeprowadzono*. To stwierdzenie opiera się na założeniu, że błąd jest addytywny i że średni błąd pojedynczej operacji jest równy jednej czwartej poziomu kwantyzacji. Przeprowadzając takie samo rozumowanie, dochodzimy do wniosku, że każda liczba o podwójnej dokładności charakteryzuje się błędem równym około *jednej czterdziestobiliardowej pomnożonym przez liczbę operacji.*

Tablica 4.2 jest ilustracją szczególnie irytującego problemu związanego z błędem zaokrąglania. Każdy z dwóch programów w tej tablicy realizuje takie samo zadanie: drukowanie 1001 liczb równomiernie rozmieszczonych między 0 i 10. W programie umieszczonym z lewej strony tablicy zastosowano zmiennoprzecinkową zmienną X jako indeks pętli. Komputer, otrzymując polecenie realizacji pętli, rozpoczyna od ustawienia indeksu na wartość początkową pętli

Tab. 4.2. Porównanie zastosowania zmiennych zmienno- i stałoprzecinkowych do sterowania pętlą. Program umieszczony z lewej strony tablicy steruje pętlą FOR-NEXT przy użyciu zmiennej zmiennoprzecinkowej X. Powoduje to, po zakończeniu programu, powstanie zakumulowanego błędu zaokrąglenia równego 0,000133, w wyniku którego zostaje pominięta ostatnia wartość pętli X = 10,0. W programie zaś z prawej strony tablicy jako indeks pętli zastosowano liczbę całkowitą I%. Daje to doskonałą dokładność i pewność, że będzie ukończona właściwa liczba cykli w pętli

```
100 'Zmiennoprzecinkowe sterowanie pętlą
110 FOR X = 0 TO 10 STEP 0.01
120   PRINT X
130 NEXT X


Dane na wyjściu programu:

    0.00
    0.01
    0.02
    0.03
     .
     .
     .
    9.960132
    9.970133
    9.980133
    9.990133
```

```
100 'Sterowanie pętlą przy użyciu liczb całkowitych
110 FOR I% = 0 TO 1000
120   X = I%/100
130   PRINT X
140 NEXT I%


Dane na wyjściu programu:

    0.00
    0.01
    0.02
    0.03
     .
     .
     .
    9.96
    9.97
    9.98
    9.99
   10.00
```


(w tym przykładzie 0). Na końcu każdego cyklu pętli pewna wartość skokowa (0,01 w tym przypadku) jest *dodawana* do indeksu. Wtedy jest podejmowana decyzja: czy jest konieczne więcej cykli pętli lub czy pętla jest zakończona. Pętla się kończy, gdy komputer stwierdzi, że wartość indeksu jest *większa* od wartości końcowej (w tym przykładzie 10,0). Jak wynika z danych na wyjściu programu, błąd zaokrąglenia w dodawaniach powoduje, że w wartości X, podczas wykonywania pętli, jest akumulowana znaczna rozbieżność. Faktycznie, zakumulowany błąd *powstrzymuje* wykonanie ostatniego cyklu pętli. Błąd powoduje, że zamiast wartości 10,0 w ostatnim cyklu, ostatnia wartość X jest równa 10,000133. Ponieważ wartość X jest większa od końcowej, więc komputer stwierdza, że jego praca jest wykonana i pętla kończy się przedwcześnie. Brakująca ostatnia wartość jest częstym błędem w wielu programach komputerowych.

W programie umieszczonym z prawej strony tablicy do sterowania pętlą zastosowano zmienną będącą liczbą całkowitą *I%*. Dodawanie, odejmowanie lub mnożenie dwóch liczb całkowitych daje zawsze inną liczbę całkowitą. Wynika stąd, że w zapisie stałoprzecinkowym w tych operacjach nie ma absolutnie żadnego błędu zaokrąglenia. Liczby całkowite idealnie nadają się do pętli sterujących, jak również do użycia jako inne zmienne podlegające wielokrotnym operacjom matematycznym. Tu mamy gwarancję wykonania ostatniego cyklu pętli! Zawsze należy stosować liczby całkowite jako indeksy pętli i liczników, jeśli nie ma jakiejś silnej motywacji przemawiającej przeciw temu.

Jeśli jako indeks pętli musicie użyć zmiennej zmiennoprzecinkowej, spróbujcie zastosować ułamki będące potęgami *dwóch* (takie jak np. 1/2, 1/4, 3/8, 27/16) zamiast potęg *dziesięciu* (takich jak np. 0,1; 0,6; 1,4; 2,3). Przykładowo lepiej jest zastosować: FOR X = 1 TO 10 STEP 0,125 zamiast FOR X = 1 TO 10 STEP 0,1. Dzięki temu indeks ma zawsze *dokładną* reprezentację binarną, co redukuje błąd zaokrąglenia. Na przykład liczba dziesiętna 1,125 może być dokładnie reprezentowana w kodzie binarnym jako $1{,}001000000000000000000000 \times 2^0$. Liczba dziesiętna 0,1 zaś znajdzie się *pomiędzy* dwiema liczbami zmiennoprzecinkowymi: 1,0999999046 i 1,1000000238 (w kodzie binarnym są to liczby: $1{,}00011001100110011001100 \times 2^0$ oraz $1{,}00011001100110011001101 \times 2^0$). To powoduje, że za każdym razem do programu jest wprowadzany inherentny błąd równy 1,1.

Warto zapamiętać pożyteczny fakt: w zapisie zmiennoprzecinkowym o pojedynczej dokładności mamy *dokładną* reprezentację każdej liczby całkowitej w zakresie ±16,8 milionów (ściśle mówiąc $\pm 2^{24}$). Powyżej tej wartości odstępy między poziomami są większe od jedności, co jest przyczyną braku niektórych liczb całkowitych. To umożliwia dodawanie, odejmowanie i mnożenie całkowitych liczb zmiennoprzecinkowych (w zakresie ±16,8 milionów) bez błędów zaokrąglenia.

Szybkość wykonania – język programowania

Programowanie DSP można z grubsza podzielić na trzy poziomy zaawansowania: *asemblacja*, *kompilacja* oraz *specjalistyczne oprogramowanie aplikacyjne*. Aby zrozumieć różnicę między nimi, musimy sięgnąć do podstaw elektroniki cyfrowej. Wszystkie mikroprocesory jako jeden z podstawowych elementów zawierają zestaw wewnętrznych rejestrów binarnych, czyli grupę przerzutników mogących zapamiętywać szeregi zer i jedynek. Na przykład procesor 8088, rdzeń oryginalnego IBM PC, ma *cztery* rejestry ogólnego przeznaczenia, z których każdy składa się z 16 bitów. Są one identyfikowane nazwami: AX, BX, CX i DX. Jest również *dziewięć* rejestrów dodatkowych do specjalnych celów. Oto ich nazwy: SI, DI, SP, BP, CS, DS, SS, ES i IP. Na przykład IP – wskaźnik instrukcji (*instruction pointer*) śledzi, gdzie w pamięci znajduje się następna instrukcja.

Przypuśćmy, że piszemy program mający zsumować liczby: 1234 i 4321. Gdy zaczyna się program, rejestr IP zawiera adres jakiejś sekcji pamięci, w której znajduje się pewna sekwencja zer i jedynek, jak to przedstawiono w tablicy 4.3. Chociaż dla większości ludzi wydaje się ona nie mieć żadnego sensu, ta sekwencja zer i jedynek zawiera jednak

wszystkie instrukcje i dane wymagane do realizacji postawionego zadania. Na przykład, gdy procesor napotka sekwencję bitów: 00000011 11000011, interpretuje ją jako instrukcję nakazującą pobrać 16 bitów danych zapisanych w rejestrze BX, dodać je binarnie do zawartości 16 bitów rejestru AX i zapisać wynik w rejestrze AX. Ten poziom programowania nosi nazwę **kodu maszynowego** lub **wewnętrznego** (*machine code*) i jest to poziom tylko o włos wyższy od pracy bezpośrednio z układami elektronicznymi.

Praca w kodzie binarnym może w końcu nawet najcierpliwszego inżyniera doprowadzić do szału, dlatego sekwencjom zer i jedynek nadano nazwy zgodne z wykonywanymi przez nie funkcjami. Ten poziom programowania nazywamy **asemblacją** (*assembly*), której przykład podano w tablicy 4.4. Chociaż program asemblerowy jest znacznie łatwiejszy do zrozumienia, to jest on w zasadzie taki sam, jak kod maszynowy. Istnieje bowiem odpowiedniość jeden do jednego między instrukcjami programu i czynnościami podejmowanymi przez mikroprocesor. Na przykład: ADD AX, BX przekłada się na: 00000011 11000011. Program zwany **asemblerem** stosuje się do konwersji kodu asemblerowego z tablicy 4.4 (zwanego **kodem źródłowym** – *source code*) na sekwencje zer i jedynek przedstawione w tablicy 4.3 (zwane **kodem wynikowym** lub **kodem wykonawczym** – *object code, executable code*). Ten kod może być bezpośrednio realizowany w mikroprocesorze. Programowanie asemblerowe wymaga, oczywiście, dogłębnego rozumienia wewnętrznej budowy mikroprocesora, który zamierzamy zastosować.

Programowanie asemblerowe obejmuje bezpośrednie operowanie elektroniką cyfrową: rejestrami, komórkami pamięciowymi, bitami stanu itd. Na następnym poziomie zaawansowania można operować zmiennymi *abstrakcyjnymi* bez żadnego odniesienia do jakichś konkretnych rozwiązań układowych. Są to języki **skompilowane** lub **wysokiego poziomu**. Kilkanaście z nich jest w powszechnym użyciu. Są to np. C, BASIC, FORTRAN, PASCAL, APL, COBOL, LISP. W tablicy 4.5 podano program w BASIC-u służący do dodawania 1234 i 4321. Programista zna tylko zmienne A, B i C i niczego nie musi wiedzieć o rozwiązaniach układowych.

Program zwany **kompilatorem** stosuje się do konwersji kodu źródłowego wysokiego poziomu bezpośrednio na kod maszynowy. W tym celu kompilator musi w układach pamięciowych *przydzielić* komórki każdej zmiennej abstrakcyjnej pojawiającej się w programie. Na przykład, gdy kompilator po raz pierwszy napotka zmienną *A* w tablicy 4.5 (wiersz 100), to przyjmuje, że programista stosuje ten symbol do oznaczenia zmiennej zmiennoprzecinkowej o pojedynczej dokładności. Kompilator wyznacza, odpowiednio, cztery bajty pamięci, które będą służyć tylko do przechowywania wartości tej zmiennej. Za każdym razem, gdy *A* pojawia się w programie, komputer wie, że trzeba zaktualizować zawartość tych czterech bajtów. Kompilator również rozkłada skomplikowane wyrażenia matematyczne, takie jak

Tab. 4.3. Maszynowy kod programu do dodawania liczb 1234 i 4321. To jest najniższy poziom programowania – bezpośrednie operowanie układami elektronicznymi. (Prawa kolumna jest przedłużeniem lewej)

10111001	00000000
11010010	10100001
00000100	00000000
10001001	00000000
00001110	10001011
00000000	00011110
00000000	00000010
10111001	00000000
11100001	00000011
00010000	11000011
10001001	10100011
00001110	00000100
00000010	00000000

Tab. 4.4. Program asemblerowy do dodawania liczb 1234 i 4321. Asembler jest programem konwertującym program asemblerowy na kod maszynowy

```
MOV CX, 1234          ;zapisz 1234 w rejestrze CX i następnie
MOV DS:[0],CX         ;przenieś to do komórki pamięci DS:[0]
MOV CX,4321           ;zapisz 4321 w rejestrze CX i następnie
MOV DS:[2],CX         ;przenieś to do komórki pamięci DS:[2]
MOV AX,DS:[0]         ;przenieś zmienne zapisane w pamięci
MOV BX,DS:[2]         ;w DS:[0] i DS:[2] do AX &BX
ADD AX,BX             ;dodaj AX i BX, sumę zapisz w AX
MOV DS:[4],AX         ;przenieś sumę do pamięci do DS:[4]
```

$Y = LOG(X_{cos(z)})$, na bardziej podstawowe operacje arytmetyczne. Mikroprocesory wiedzą tylko jak dodawać, odejmować, mnożyć i dzielić. Wszelkie działania bardziej skomplikowane muszą być wykonywane jako ciąg tych czterech podstawowych operacji.

Języki wysokiego poziomu izolują programistę od układów elektronicznych. To ułatwia programowanie i umożliwia przekazywanie kodu źródłowego między mikroprocesorami różnych typów. Najważniejsze jest wszakże to, że programista stosujący język kompilatora nie musi *niczego* wiedzieć o wewnętrznym działaniu komputera. Tę odpowiedzialność wziął na siebie inny programista – ten, który napisał kompilator.

Większość kompilatorów działa na zasadzie konwersji *całego* programu na kod maszynowy, *zanim* program jest wykonywany. Wyjątkiem od tej zasady jest kompilator zwany **interpreterem**. Najbardziej znanym przykładem jest **interpreter BASIC**. Interpreter przetwarza *pojedynczy wiersz* kodu źródłowego na kod maszynowy, wykonuje ten kod maszynowy i następnie przechodzi do następnego wiersza kodu źródłowego. Daje to prostym programom otoczenie interaktywne, choć szybkość wykonywania jest skrajnie mała (powiedzmy o czynnik 100).

Najwyższy poziom zaawansowania reprezentują specjalizowane **aplikacyjne** pakiety oprogramowania DSP. Mają one różną formę i są często dostarczane wraz ze specjalizowanym sprzętem. Przyjmijmy, że kupiliście nowo opracowany mikroprocesor DSP, aby zastosować go w właśnie realizowanym projekcie. Takie mikroprocesory często mają wiele właściwości specjalnie dostosowanych do DSP: wejścia i wyjścia analogowe, cyfrowe wejścia/wyjścia, filtry antyaliasingowe i odtwarzające itd. Powstaje pytanie: jak je programować? W najgorszym przypadku producent dostarczy wam *asembler* i będzie oczekiwał, że nauczycie się wewnętrznej architektury procesora. W bardziej typowym scenariuszu będzie dostarczony *kompilator C* umożliwiający zaprogramowanie mikroprocesora bez konieczności wnikania w sposób jego działania.

W najlepszym przypadku producent dostarczy wysublimowany pakiet programowy wspomagający opracowanie programu i zawierający: biblioteki algorytmów, napisane wcześniej procedury, narzędzia do usuwania błędów w programie itd. Można po prostu łączyć ikony, aby utworzyć żądany system, korzystając z łatwego w użyciu obrazowania graficznego. Operuje się wówczas drogami sygnałów, algorytmami do przetwarzania sygnałów, parametrami wejść i wyjść analogowych itd. Gdy jesteście już zadowoleni z projektu programu, jest on transformowany w odpowiedni kod maszynowy do wykonywania w układach elektronicznych. Inne typy pakietów aplikacyjnych są stosowane w przetwarzaniu obrazu, analizie widmowej, oprzyrządowaniu i sterowaniu, projektowaniu filtrów itd. To jest w znacznym stopniu sprawa przyszłości.

Tab. 4.5. Program BASIC do dodawania liczb 1234 i 4321. Kompilator jest programem konwertującym kod źródłowy wysokiego poziomu na kod maszynowy

```
100 A =1234
110 B = 4321
120 C = A+B
130 END
```

Rozróżnienie między trzema poziomami oprogramowania bywa bardzo niewyraźne. Na przykład języki o najwyższym stopniu kompilacji umożliwiają bezpośrednie operowanie układami elektronicznymi. Podobnie język wysokiego poziomu z dobrze zaopatrzoną biblioteką funkcji DSP jest bardzo bliski stania się pakietem aplikacyjnym. Główną zasadą podziału między trzema kategoriami oprogramowania jest zrozumienie, czym się operuje: 1) układami elektronicznymi, 2) zmiennymi abstrakcyjnymi czy 3) całymi procedurami i algorytmami.

Jest jeszcze jeden ważny aspekt tej klasyfikacji. Gdy stosujemy język wysokiego poziomu, polegamy na umiejętnościach programisty, który napisał kompilator w celu zrozumienia najlepszych sposobów operowania układami elektronicznymi. Także gdy korzystamy z pakietu aplikacyjnego, polegamy na wiedzy programisty, który opracował ten pakiet w celu zrozumienia najlepszych metod DSP. I tu jest kłopot, ci programiści nigdy nie zetknęli się z problemem, nad którym wy właśnie pracujecie. Dlatego nie zawsze mogą wam dostarczyć optymalne rozwiązanie. Gdy działacie na *wyższym* poziomie, bierzcie pod uwagę, że końcowy kod maszynowy będzie *mniej* efektywny pod względem wykorzystania pamięci, szybkości i dokładności.

Który język programowania powinniście stosować? To zależy od tego, *kim* jesteście i co zamierzacie zrobić. Większość specjalistów komputerowych i programistów stosuje język C (lub bardziej zaawansowany C++). Wydajność, elastyczność, modularność – te wszystkie zalety ma język C. C jest tak popularny, że można zapytać, po co miałby ktokolwiek programować systemy DSP w języku innym niż C. Przychodzą na myśl trzy odpowiedzi na takie pytanie. Po pierwsze, technika DSP rozwinęła się tak gwałtownie, że zaskoczyła niektóre osoby i organizacje przyzwyczajone do stosowania innych języków, takich jak FORTRAN i PASCAL. Dotyczy to zwłaszcza agend rządowych i instytucji wojskowych, jak wiadomo trudno podatnych na szybkie zmiany. Po drugie, niektóre zastosowania wymagają skrajnie wysokiej efektywności, osiągalnej tylko w programach asemblerowych. Należą one do kategorii „trochę więcej szybkości za o wiele więcej pracy". Po trzecie, C nie jest językiem łatwym do opanowania, zwłaszcza przez specjalistów, dla których programowanie nie jest głównym zajęciem. Dotyczy to wielu inżynierów i naukowców, którzy od czasu do czasu potrzebują metod DSP w swych badaniach i pracach projektowych. Ta grupa specjalistów często zwraca się ku BASIC-owi ze względu na jego prostotę.

Dlaczego w tej książce wybrano BASIC? Tutaj piszemy o *algorytmach*, a nie o stylu programowania. Czytelnicy mają koncentrować uwagę na technice DSP, a nie rozpraszać jej na triki poszczególnych języków. Na przykład wszystkie programy w tej książce zawierają *numery wierszy*. To ułatwia opisy działania programu: „w wierszu 100 wykonuje się to a to, a w wierszu 110 – to i tamto" itd. Oczywiście, przypuszczalnie nigdy nie będziecie używać numeracji wierszy w waszych użytkowych programach. Inne bowiem są wymagania dotyczące *uczenia DSP*, a inne – jego *stosowania*. Jest wiele książek, w których zamieszczono znakomite kody źródłowe do algorytmów DSP. Jeśli tylko szukacie już napisanych kodów, aby skopiować je w waszych programach, to w tej książce ich nie znajdziecie.

Wszelkie porównywanie szybkości działania układów elektronicznych z szybkością wykonywania programów jest niewdzięcznym zadaniem. Niezależnie od wyniku, przegrywający zawsze powie, że wyścig był nieuczciwy! Programiści, którzy lubią języki wysokiego poziomu (np. tradycyjni specjaliści komputerowi), będą argumentować, że asemblery są tylko o 50% szybsze od kodów skompilowanych, lecz pięciokrotnie bardziej kłopotliwe. Ci, którzy lubią asemblery (są to na ogół naukowcy i inżynierowie elektronicy), będą się upierać przy opinii przeciwnej: asembler jest pięciokrotnie szybszy, lecz tylko o 50% trudniejszy w stosowaniu. Jak przy większości takich kontrowersji, obie strony sporu mogą dostarczyć wybiórcze dane wspierające ich opinie.

Jako zasadę praktyczną można przyjąć, że podprogram napisany w asemblerze będzie od 1,5 do 3 razy szybszy niż porównywalny program w języku wysokiego poziomu. Jedynym sposobem poznania dokładnej wartości tego stosunku szybkości jest napisanie programów i przeprowadzenia testów praktycznych. Ponieważ szybkość działania komputerów osobistych wzrasta corocznie o ok. 40%, więc napisanie podprogramu w asemblerze jest równoważne skokowi do przodu o ok. 2 lata w rozwoju sprzętu.

Większość profesjonalnych programistów obraża się, słysząc propozycję użycia asemblera i śmieje się z sugestii zastosowania BASIC-a. Ich rozumowanie jest całkiem proste. Asembler i BASIC zniechęcają do stosowania dobrych praktyk w programowaniu. Dobry program powinien być *przenośny* (możliwy do przekazywania z jednego typu komputer na inny), *modularny* (złożony z dobrze zdefiniowanych podprogramów) i *łatwo zrozumiały* (wyposażony w wiele komentarzy i opisowe nazwy zmiennych). Słaba struktura asemblera i BASIC-a utrudnia spełnienie tych wymagań. Wiąże się to też z faktem, że osoby chętnie stosujące asembler i BASIC często mają niewiele doświadczenia w zakresie prawidłowej struktury oprogramowania i dokumentacji.

Zwolennicy asemblera odpowiadają na ten atak własną repliką. Przypuśćmy, że piszecie w języku C, a wasz konkurent pisze ten sam program w asemblerze. Pierwsze wrażenie końcowego użytkownika będzie takie, że wasz program to rupieć, gdyż jest dwa razy wolniejszy. Nikt przecież nie będzie sugerował, żeby pisać w asemblerze bardzo duże programy, lecz tylko części programu wymagające bardzo szybkiego wykonywania. Na przykład wiele funkcji w bibliotekach oprogramowania DSP jest napisanych w asemblerze i potem dostępnych z większych programów napisanych w C. Nawet najwierniejsi puryści oprogramowania będą *używać* asemblera, o ile tylko nie muszą go *napisać*.

Szybkość wykonywania – sprzęt

Moc obliczeniowa komputerów wzrasta tak gwałtownie, że każda *książka* na ten temat staje się przestarzała, zanim zostanie opublikowana. To jest koszmarny sen autorów! Oryginalny IBM PC wprowadzono na rynek w roku 1981. Jego podstawą był mikroprocesor 8088 z zegarem 4,77 MHz i z 8-bitową magistralą danych. Potem nastąpiły kolejne generacje komputerów osobistych wprowadzane co 3–4 lata: 8088›80286›80386›80486›80586 (Pentium). W każdym z tych nowych systemów szybkość obliczeń wzrastała o czynnik *pięć* w stosunku do rozwiązania poprzedniego. Do roku 1996 częstotliwość zegara wzrosła do 200 MHz, a magistralę rozszerzono do 32 bitów. Wraz z innymi udoskonaleniami dało to wzrost mocy obliczeniowej blisko *tysiąckrotny* w ciągu 15 lat! W ciągu *następnych* 15 lat należy oczekiwać dalszego wzrostu o *czynnik 1000*[8].

Jedynym źródłem uzyskania aktualnych informacji w tej gwałtownie zmieniającej się dziedzinie są bezpośredni producenci, którzy publikują reklamy, karty katalogowe, listy cenowe itp. Zapomnijcie o szukaniu takich informacji w książkach, sięgajcie do czasopism i nawet do pism codziennych. Bierzcie pod uwagę, że podstawowa szybkość obliczeniowa będzie się co najmniej podwajała co dwa lata. Nie wystarczy tylko poznawanie aktualnego stanu rozwoju techniki komputerowej. Trzeba śledzić ten rozwój i przewidywać jego dalszy trend.

Mając to na uwadze, możemy teraz przejść do omówienia, w jaki sposób szybkość działania jest ograniczana przez komputer. Komputery są zbudowane z wielu podsystemów, więc czas wykonywania zadań zależy od dwóch podstawowych czynników: 1) szybkości działania poszczególnych podsystemów, 2) czasu koniecznego do przesyłania danych między tymi blokami. Na rys. 4.5 przedstawiono uproszczony schemat najważniejszych, ograniczających szybkość działania, bloków typowego komputera osobistego. **Jednostka centralna** (CPU) jest sercem systemu. Jak już poprzednio powiedziano, składa się ona z kilkunastu rejestrów, każdy o długości 32 bitów (w komputerach obecnej generacji)[9]. W CPU są też układy cyfrowe potrzebne do operacji podstawowych, takich jak przesuwanie zawartości bitów i arytmetyka stałoprzecinkowa.

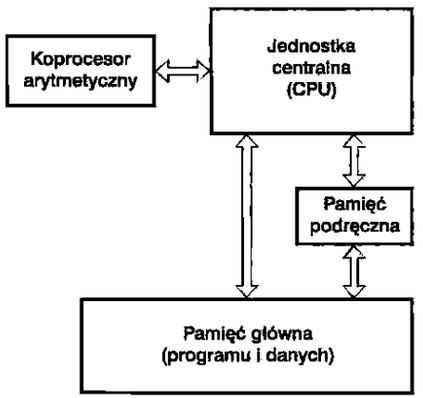
[8] Obecnie (rok 2006) standardowym procesorem stosowanym w popularnych komputerach PC jest 32-bitowy Pentium 4 produkowany przez firmę Intel oraz jego odpowiedniki produkowane przez firmy AMD-Athlon i Sempron. Komputery do zastosowań profesjonalnych są wyposażane w 64-bitowe procesory Itanium, Xeon, Athlon 64 i Opteron. Ich wewnętrzna częstotliwość taktowania dochodzi do 5 GHz (przyp. tłum.).

[9] Także obecnie – w roku 2006 (przyp. tłum.).

Bardziej rozbudowane działania matematyczne są wykonywane przez transfer danych do specjalnego układu nazywanego **koprocesorem arytmetycznym** (zwany też **jednostką arytmetyczno-logiczną ALU** – *Arithmetic Logic Unit*). Koprocesor arytmetyczny może znajdować się w tym samym układzie scalonym co jednostka centralna albo może stanowić oddzielny układ. Na przykład przy dodawaniu dwóch liczb zmiennoprzecinkowych jednostka centralna powinna przesłać 8 bajtów (4 na każdą liczbę) do koprocesora arytmetycznego, a także kilka bajtów określających, co należy zrobić z przesłanymi danymi. Po krótkim czasie obliczania, koprocesor arytmetyczny przekazuje cztery bajty z powrotem do CPU. Te bajty zawierają liczbę zmiennoprzecinkową będącą obliczoną sumą. Najtańsze systemy komputerowe nie mają koprocesora arytmetycznego lub mają go tylko jako opcję. Na przykład mikroprocesor 80486DX ma wewnętrzny koprocesor arytmetyczny, podczas gdy 80486SX go nie ma. W takich systemach o ograniczonym działaniu zastępuje się rozbudowę sprzętu oprogramowaniem. Każda funkcja matematyczna jest dzielona na elementarne operacje binarne, które mogą być wykonywane bezpośrednio przez jednostkę centralną. Uzyskany wynik jest wprawdzie taki sam, lecz czas wykonania znacznie dłuższy, o czynnik, powiedzmy od 10 do 20.

Większość oprogramowania komputerów osobistych może być używana z koprocesorem arytmetycznym lub bez niego. Uzyskuje się to dzięki posiadaniu kompilatora generującego kod maszynowy do obsługi obu tych sytuacji. Wszystko jest zapisane w końcowym wykonywalnym programie. Jeśli w właśnie stosowanym komputerze jest koprocesor arytmetyczny, to tylko jedna sekcja kodu jest realizowana. Jeśli zaś koprocesora nie ma, to będzie używana inna sekcja kodu. Kompilator może też być tak sterowany, żeby generował kod tylko dla jednej z wymienionych sytuacji. Można na przykład czasem spotkać program, który wymaga obecności koprocesora arytmetycznego i załamuje się, jeśli napotka komputer bez koprocesora. Takie zastosowania, jak przetwarzanie tekstów, zwykle nie czerpią żadnej korzyści z koprocesora arytmetycznego. Wymagają one bowiem przemieszczania danych w pamięci, a nie obliczania wyrażeń matematycznych. Obecność koprocesora arytmetycznego nie ma również wpływu na obliczenia obejmujące zmienne stałoprzecinkowe (liczby całkowite), gdyż są one realizowane wewnątrz jednostki centralnej. Z drugiej jednak strony, szybkość wykonywania DSP i innych programów obliczeniowych zawierających operacje zmiennoprzecinkowe może różnić się o rząd wielkości zależnie od tego, czy jest koprocesor arytmetyczny, czy go nie ma.

W większości systemów komputerowych jednostka centralna (CPU) i główna pamięć znajdują się w oddzielnych układach scalonych. Z oczywistych względów chcielibyśmy, żeby główna pamięć była bardzo duża i bardzo szybka. Niestety taka pamięć jest bardzo kosz-



Rys. 4.5. Architektura typowego systemu komputerowego. Szybkość działania komputera jest ograniczona przez: 1) szybkość pracy poszczególnych podsystemów, 2) szybkość, z jaką dane mogą być przesyłane między tymi podsystemami

towna. Przesyłanie danych między główną pamięcią i jednostką centralną jest na ogół wąskim gardłem ograniczającym szybkość. Jednostka centralna *prosi* pamięć główną o informację binarną zarejestrowaną pod określonym adresem pamięciowym i następnie *oczekuje* na otrzymanie tej informacji. Powszechnie stosowaną metodą ominięcia tej trudności jest użycie **pamięci podręcznej** (*cache memory*). Jest to niewielka, bardzo szybka pamięć będąca buforem między CPU i pamięcią główną. Pamięć podręczna ma, typowo, pojemność, kilkuset kilobajtów. Gdy CPU żąda od pamięci głównej dostarczenia danych binarnych z określonego adresu, to szybkie elektroniczne układy cyfrowe kopiują do pamięci podręcznej pewną *sekcję* pamięci głównej wokół tego adresu. Przy następnym żądaniu przez CPU informacji z pamięci jest bardzo prawdopodobne, że ta informacja będzie zawarta w pamięci podręcznej, dzięki czemu jej pozyskanie będzie bardzo szybkie. Wynika to z faktu, że w programach jest tendencja do zwracania się do komórek pamięci będących w najbliższym sąsiedztwie tych, z których poprzednio pobrano dane. W typowych zastosowaniach komputerów osobistych dodanie pamięci podręcznej może kilkakrotnie zwiększyć szybkość działania. Pamięć podręczna może znajdować się w tym samym układzie scalonym, co jednostka centralna lub może być zewnętrznym układem elektronicznym.

Szybkość, z jaką dane są przesyłane między podsystemami, zależy od liczby dostępnych równoległych linii danych oraz od maksymalnej szybkości przesyłania sygnałów cyfrowych wzdłuż każdej linii. Ogólnie biorąc, dane cyfrowe można znacznie szybciej przesyłać wewnątrz jednego układu scalonego niż między układami. Jeszcze wolniejsze przesyłanie następuje w ścieżkach danych wiodących przez złącza do innych płytek drukowanych (tj. struktura magistralowa). Jest więc silna motywacja do „upchnięcia" w jednostce centralnej tak dużo elektroniki, ile tylko można.

Szczególnie nieprzyjemnym problemem związanym z szybkością pracy komputera jest *kompatybilność wsteczna*. Firma komputerowa, wprowadzając nowy produkt (powiedzmy, kartę zbierania danych lub program), chce dla niego znaleźć możliwie jak największy rynek. To powoduje, że produkt musi być kompatybilny z większością aktualnie używanych komputerów, co obejmuje kilka generacji rozwoju technicznego. Takie wymaganie często ogranicza działanie sprzętu lub oprogramowania do poziomu odpowiadającego systemom znacznie starszym. Przypuśćmy na przykład, że kupujemy kartę wejścia/wyjścia jako moduł wtykany do magistrali osobistego komputera Pentium 1 GHz. Karta ma 8 linii do wysyłania i odbierania danych (jednorazowo jeden bajt). Następnie piszemy program asemblera do bardzo szybkiego przesyłania danych między naszym komputerem i jakimś urządzeniem zewnętrznym, takim jak eksperyment naukowy lub inny komputer. Ku naszemu zdziwieniu maksymalna przepływność danych jest tylko ok. 100 000 bajtów na sekundę, a więc *dziesięć tysięcy* razy wolniej niż częstotliwość zegarowa mikroprocesora! Winowajcą jest magistrala ISA – technika, która jest *wstecznie kompatybilna* z komputerami z wczesnych lat 80.

W tablicy 4.6 zestawiono czasy wykonania operacji w komputerach kilku generacji. Te dane trzeba, oczywiście, traktować jako zgrubne przybliżenie. Jeśli chcecie zrozumieć *wasz* system, to wykonajcie pomiary *waszego* systemu. To całkiem łatwe, trzeba napisać pętlę programową wykonującą *milion* razy jakąś operację i sprawdzić na zegarku, ile czasu zajmuje realizacja takiej pętli. Pierwsze trzy systemy wymienione w tablicy 4.6, a więc 80286, 80486 i Pentium, są standardowymi biurkowymi komputerami osobistymi, odpowiednio z lat 1986, 1993 i 1996. Czwarty to mikroprocesor z 2000 roku specjalnie zaprojektowany do zadań DSP – ADSP-TS101 firmy Analog Devices.

Pentium jest szybszy od systemu 80286 z czterech powodów: 1) szybszy zegar, 2) więcej linii w magistrali danych, 3) dodanie pamięci podręcznej, 4) bardziej efektywnie zaprojektowane układy wewnętrzne, wymagające mniej okresów zegarowych na wykonanie jednej instrukcji.

Jeśli Pentium było cadillakiem, to ADSP-TS101 można porównać do ferrari: mniej wygód, lecz oszałamiająca szybkość. Ten układ scalony jest reprezentatywny dla kilku mikroprocesorów zaprojektowanych specjalnie w taki sposób, aby zmniejszyć czas wykonania algorytmów DSP. Innymi układami tej kategorii są: TIC55xx firmy Texas Instruments

i DSP56800 firmy Motorola. Takie układy scalone są nazywane **mikroprocesorami DSP**, **cyfrowymi procesorami sygnałowymi** i **RISC** (*Reduced Instruction Set Computer*, komputerami o zredukowanej liczbie rozkazów). Ta ostatnia nazwa wyraża fakt, że zwiększono szybkość wynika z mniejszej liczby instrukcji na poziomie asemblera, udostępnionych programiście. Dla porównania, bardziej tradycyjne mikroprocesory, jak Pentium, są nazywane **CISC** (*Complex Instruction Set Computer*, komputery z rozbudowaną listą rozkazów).

Mikroprocesory DSP mają zastosowania dwojakiego rodzaju. Mogą pracować jako moduły podporządkowane (*slave*) pod kontrolą komputera bardziej konwencjonalnego albo jako procesory wbudowane w jakimś wydzielonym zastosowaniu, takim jak telefon komórkowy. W niektóre modelach mikroprocesorów DSP możliwe jest wykonywanie działań tylko na liczbach stałoprzecinkowych, a inne mogą operować liczbami zmiennoprzecinkowymi. Architektura wewnętrzna zastosowana w celu zwiększenia szybkości obejmuje: 1) wielką pamięci podręcznej zawartej wewnątrz układu scalonego, 2) oddzielne magistrale dla programu i dla danych, umożliwiające dostęp do obu z nich równocześnie (tzw. **architektura harwardzka**), 3) szybkie rozwiązania układowe do operacji matematycznych, zawarte bezpośrednio w mikroprocesorze oraz 4) *potokowy* tryb pracy (*pipeline*).

W architekturze **potokowej** rozdziela się *układy* wymagane do wykonania określonego zadania na kilka następujących po sobie stopni. Na przykład dodawanie dwóch liczb może być wykonywane w trzech stopniach połączonych potokowo. Pierwszy stopień pobiera tylko z pamięci liczby, które mają zostać zsumowane. Jedynym zadaniem stopnia drugiego jest zsumowanie obu liczb, a trzeciego – zapisanie wyniku w pamięci. Jeśli każdy stopień może zakończyć swoje zadanie podczas jednego okresu zegarowego, to wykonanie całej procedury zajmie trzy okresy zegarowe. Podstawową właściwością struktury potokowej jest możliwość rozpoczęcia następnego zadania, zanim poprzednie zostało zakończone. W tym przykładzie możemy rozpocząć sumowanie *innych* dwóch liczb, gdy tylko pierwszy stopień jest wolny, a więc przy końcu pierwszego okresu ze-

Tab. 4.6. Zmierzone czasy wykonania operacji w różnych komputerach. Czasy podano w mikrosekundach, jeśli nie zaznaczono inaczej. 80286, 80486 i Pentium to trzy generacje komputerów osobistych, podczas gdy TS101 jest mikroprocesorem zaprojektowanym specjalnie do zadań DSP. Wszystkie te komputery osobiste zawierają koprocesor arytmetyczny. Podane czasy należy traktować tylko jako ogólne oszacowanie. Czasy wykonania operacji w waszych komputerach będą się zmieniać zależnie od zastosowanych rozwiązań układowych i oprogramowania

	[illegible] (12 MHz)	[illegible] (33 MHz)	PENTIUM (100 MHz)	[illegible] (250 MHz)
LICZBY CAŁKOWITE				
A% = B% + C%	1,6	0,12	0,04	
A% = B% – C%	1,6	0,12	0,04	
A% = B% × C%	2,7	0,59	0,13	
A% = B% ÷ C%	64	9,2	1,5	
LICZBY ZMIENNOPRZECINKOWE				
A = B + C	33	2,5	0,50	4 ns
A = B – C	35	2,5	0,50	4 ns
A = B × C	35	2,5	0,50	4 ns
A = B ÷ C	49	4,5	0,87	32 ns
A = SQR(B)	45	5,3	1,3	56 ns
A = LOG(B)	186	19	3,4	196 ns
A = EXP(B)	246	25	5,5	152 ns
A = B^C	311	31	5,3	148 ns
A = SIN(B)	262	30	6,6	152 ns
A = ARCTAN(B)	168	21	4,4	244 ns

garowego. Przy wielkiej liczbie operacji szybkość systemu będzie określana jako jedno sumowanie na okres zegarowy, nawet jeśli przeprowadzenie sumowania każdej pary liczb wymaga trzech okresów zegarowych. Systemy potokowe są świetne pod względem szybkości, ale mogą być trudne do programowania. Algorytm powinien dopuszczać rozpoczęcie nowego obliczenia nawet gdy wyniki poprzednich obliczeń nie są dostępne (ponieważ jeszcze są w systemie potokowym).

Mikroprocesory DSP są bardziej szczegółowo omówione w rozdziałach 28 i 29. Są to zadziwiające urządzenia; ich duża moc obliczeniowa i niska cena spowodują wprowadzenie DSP do wielu zastosowań naukowych i w sprzęcie powszechnego użytku. Jest to jedna z tych technik, która ukształtuje XXI wiek.

Szybkość wykonywania – triki programowe

Układy elektroniczne komputera i języki programowania są ważne dla maksymalizacji szybkości działania, ale nie można ich zmieniać z dnia na dzień. To zaś, *jak programujemy*, można zmieniać w każdym czasie i wpłynie to radykalnie na czas wymagany do realizacji programu. Oto trzy sugestie.

Po pierwsze, tam, gdzie to możliwe, stosujcie liczby całkowite zamiast zmiennych zmiennoprzecinkowych. Konwencjonalne mikroprocesory, takie jak stosowane w komputerach osobistych, operują na liczbach całkowitych 10 do 20 razy szybciej niż na zmiennoprzecinkowych. W systemach z koprocesorem arytmetycznym różnica może być jak 200:1. Wyjątkiem jest dzielenie liczb całkowitych, które często przeprowadza się, konwertując wartości na zmiennoprzecinkowe. To czyni operację koszmarnie wolną w porównaniu z innymi obliczeniami na liczbach całkowitych. Szczegóły podano w tablicy 4.6.

Po drugie, unikajcie używania funkcji takich, jak sin(*x*), log(*x*), y^x itd. Te funkcje przestępne są obliczane jako szeregi dodawań, odejmowań i mnożeń. Na przykład szereg potęgowy MacLaurina daje:

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\frac{x^{11}}{11!}+\ldots$$

$$\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\frac{x^{10}}{10!}+\ldots$$

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\frac{x^5}{5!}+\ldots$$

Wzór 4.3. Rozwinięcia trzech funkcji przestępnych w szeregi potęgowe MacLaurina. W ten właśnie sposób komputery obliczają funkcje tego typu i dlatego trwa to tak długo

Długości tych rozwinięć są wprawdzie nieskończone, lecz kolejne wyrazy stają się szybko tak małe, że mogą być pominięte. Na przykład:

sin(1) = 1 – 0,166666 + 0,008333 – 0,000198 + 0,000002 – ...

Obliczanie tych funkcji wymaga czasu 10-krotnie dłuższego niż proste dodawanie lub mnożenie (patrz tablica 4.6). Można skorzystać z kilku trików, aby ominąć te obliczenia. Na przykład można zastosować wzory: $x^3 = x \cdot x \cdot x$; $\sin(x) \approx x$, gdy x jest bardzo małe; $\sin(-x) = -\sin(x)$, gdy już znamy jedną z wartości i chcemy określić drugą itd. W większości języków jest podawanych tylko kilka funkcji przestępnych, a inne programista powinien wyprowadzić z zależności podanych w tablicy 4.7. Takie obliczenia są oczywiście nawet jeszcze wolniejsze.

Inną możliwością jest wcześniejsze obliczenie tych „powolnych" funkcji i zapisanie ich wartości w **tablicy przeglądowej** (**LUT** – *Look Up Table*). Wyobraźmy sobie na przykład 8-bitowy system akwizycji danych stosowany do ciągłego monitorowania *napięcia* na rezystorze. Jeśli interesującym nas parametrem jest *moc* wydzielana w rezystorze, to zmierzone napięcie można wykorzystać do obliczenia mocy według wzoru: $P = U^2/R$. Jest jednak szybsza alternatywa.

Tab. 4.7. Obliczanie funkcji rzadko stosowanych na podstawie funkcji bardziej popularnych. Wszystkie kąty podano w radianach, ATN(XD) to arcus tangens, LOG(X) – logarytm naturalny, SGN(X) – funkcja znaku (tj. –1 dla X ≤ 0, 1 dla X > 0), EXP(X) to jest e^X

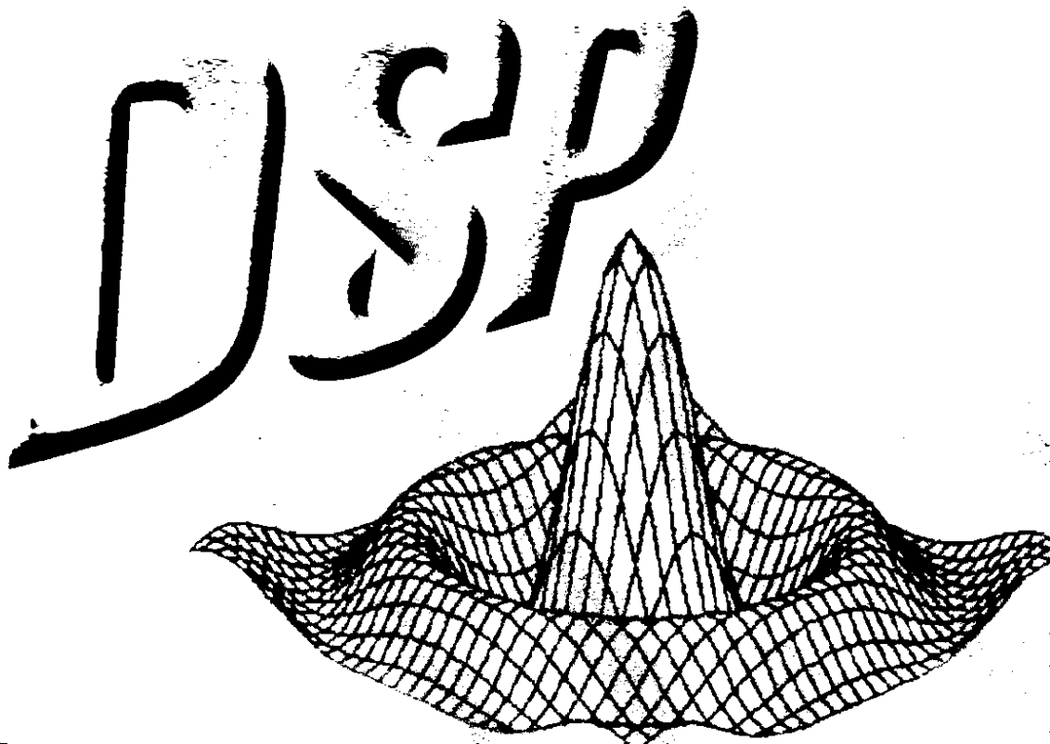
Secans (X) =	1/COS(X)
Cosecans (X) =	1/SIN(X)
Cotangens (X) =	1/TAN(X)
Arcus sinus (X) =	ATN(X/SQR(1–X*X))
Arcus cosinus (X) =	–ATN(X/SQR(1–X*X)) + PI/2
Arcus secans (X) =	ATN(SQR(X*X–1)) + (SGN(X)–1) * PI/2
Arcus cosecans (X) =	ATN(1/SQR(X*X–1)) + (SGN(X)–1) * PI/2
Arcus cotangens (X) =	–ATN(X) + PI/2
Sinus hiperboliczny (X) =	(EXP(X)–EXP(–X))/2
Cosinus hiperboliczny (X) =	(EXP(X)+EXP(–X))/2
Tangens hiperboliczny (X) =	(EXP(X)–EXP(–X))/(EXP(X)+EXP(–X))
Secans hipeboliczny (X) =	1/HYPERBOLIC COSINE
Cosecans hiperboliczny (X) =	1/HYPERBOLIC SINE
Cotangens hiperboliczny (X) =	1/HYPERBOLIC TANGENT
Arcus sinus hiperboliczny (X) =	LOG(X+SQR(X*X+1))
Arcus cosinus hiperboliczny (X) =	LOG(X+SQR(X*X–1))
Arcus tangens hiperboliczny (X) =	LOG((1+X)/(1–X))/2
Arcus secans hiperboliczny (X) =	LOG((SQR(1–X*X)+1)/X)
Arcus cosecans hiperboliczny (X) =	LOG(1+SGN(X)*SQR(1+X*X))/X
Arcus cotangens hiperboliczny (X) =	LOG((1+X)/(X–1))/2
LOG10(X) =	LOG(X)/LOG(10) = 0,4342945 LOG(X)
PI =	4*ATN(1) = 3,141592653589794

Wartość odpowiadająca każdej z możliwych 256 wartości napięcia może być obliczona wcześniej i zapisane w pamięci przeglądowej. Gdy system jest uruchomiony, zmierzone napięcie – będące liczbą między 0 i 255 – staje się indeksem w tablicy przeglądowej służącym do odszukania odpowiadającej mu mocy. Stosując tablice przeglądowe, można uzyskać szybkość setki razy większą niż w bezpośrednim obliczeniu.

Po trzecie, powinniście wiedzieć, co w waszym systemie jest *szybkie*, a co *wolne*. Ta wiedza przychodzi wraz z doświadczeniem i próbami. Zawsze pojawiają się jakieś niespodzianki. Szczególną uwagę zwróćcie na instrukcje graficzne i we/wy. Jest zwykle kilka sposobów realizacji tych instrukcji, dlatego szybkości ich wykonania mogą być bardzo zróżnicowane. Na przykład w BASIC-u instrukcja *BLOAD* przesyła plik danych bezpośrednio do jakiejś sekcji pamięci. Wczytywanie tego samego pliku do pamięci bajt po bajcie (w pętli) może być 100 razy wolniejsze. Innym przykładem może być instrukcja BASIC-a *LINE* stosowana do rysowania kolorowej ramki na ekranie wizyjnym. Rysowanie takiej samej ramki piksel po pikselu również może trwać 100 razy dłużej. Nawet wprowadzenie do pętli instrukcji *drukowania* (aby śledzić, co się dzieje) może spowolnić tę operację *tysiące razy*.

5

Systemy liniowe



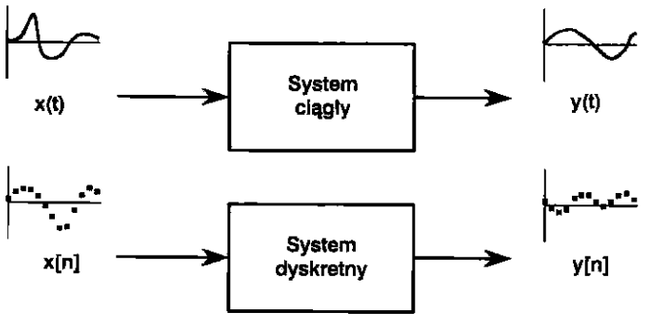
Podstawą większości metod DSP jest strategia „dziel i rządź", zwana zasadą *superpozycji*. Przetwarzany sygnał jest dzielony na proste składniki, po czym każdy z nich jest przetwarzany indywidualnie a wyniki są następnie ponownie łączone ze sobą. Takie podejście ma wielką zaletę, że umożliwia rozbicie jednego skomplikowanego problemu na wiele łatwych. Zasadę superpozycji można stosować tylko w odniesieniu do *systemów liniowych*, a więc tylko takich, dla których obowiązują pewne reguły matematyczne. Na szczęście większość zastosowań spotykanych w nauce i technice należy do klasy systemów liniowych. W tym rozdziale są rozpatrywane pytania fundamentalne dla DSP: co to znaczy, że system jest liniowy; jakie są sposoby dzielenia sygnałów na prostsze składowe i jak zasada superpozycji daje dużą rozmaitość metod przetwarzania sygnałów.

Sygnały i systemy

Sygnał jest opisem zmian jednego parametru przy zmianach innego – np. napięcie w układzie elektronicznym zmieniające się w funkcji czasu lub natężenie oświetlenia zmieniające się z odległością. **Systemem** jest każdy proces wytwarzający *sygnał wyjściowy* w odpowiedzi na *sygnał wejściowy*, co zilustrowano na schemacie blokowym na rys. 5.1. Systemy ciągłe mają na wejściu i na wyjściu sygnały ciągłe (np. analogowe układy elektroniczne). Systemy dyskretne mają dyskretne sygnały wejściowe i wyjściowe (np. programy komputerowe opisujące wartościami zarejestrowanymi w pamięciach).

W oznaczeniach sygnałów obowiązuje kilka reguł. Te reguły nie zawsze są przestrzegane w DSP, lecz należy je zapamiętać ze względu na powszechnie stosowanie. Bez jasnych i klarownych oznaczeń matematyka stałaby się trudniejsza. Po pierwsze, w oznaczeniu sygnałów *ciągłych* stosuje się nawiasy okrągłe, np. $x(t)$ i $y(t)$, a w *dyskretnych* – nawiasy kwadratowe, np. $x[n]$ i $y[n]$. Po drugie, do oznaczania sygnałów używa się małych liter. Duże litery są zarezerwowane dla sygnałów w dziedzinie częstotliwości, które będą omawiane w dalszych rozdziałach. Po trzecie, oznaczenie nadawane sygnałowi jest zwykle opisem parametru reprezentowanego przez sygnał. Na przykład *napięcie* w zależności od *czasu* jest oznaczane jako $u(t)$, a cena giełdowa wyznaczana każdego *dnia* – jako $p[d]$.

Sygnały i systemy są często rozpatrywane bez dokładnej znajomości parametrów, które reprezentują. Jest to podobne do stosowania w algebrze oznaczeń x i y bez przypisywania tym zmiennym jakiegokolwiek znaczenia fizycznego. Dochodzimy tu do czwartej reguły oznaczania sygnałów. Jeśli nie jest dostępne oznaczenie bardziej opisowe, to sygnał wejściowy



Rys. 5.1. Terminologia sygnałów i systemów. Systemem jest każdy proces wytwarzający sygnał wyjściowy w odpowiedzi na wejściowy. W oznaczeniach sygnałów ciągłych stosuje się zwykle nawiasy okrągłe, a dyskretnych – kwadratowe. Wszystkie sygnały są oznaczane małymi literami, gdyż duże zarezerwowano dla sygnałów w dziedzinie częstotliwości (omówionych w dalszych rozdziałach). Jeśli nie ma lepszego oznaczenia, to sygnał wejściowy jest oznaczany jako x(t) lub x[n], a wyjściowy y(t) lub y[n]

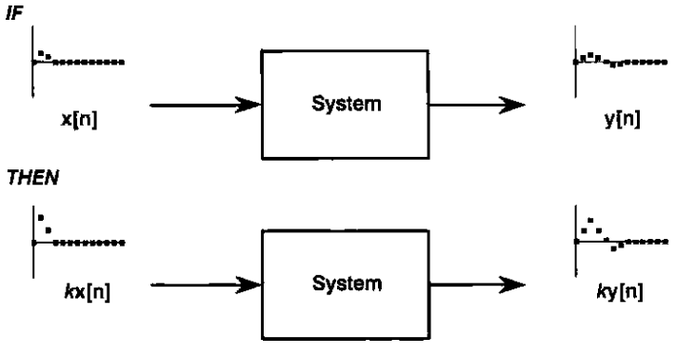
w systemie dyskretnym jest zazwyczaj oznaczany $x[n]$, a wyjściowy $y[n]$. W systemach ciągłych stosuje się oznaczenia $x(t)$ i $y(t)$.

Jest wiele powodów, dla których potrzebne jest rozumienie *systemu*. Można na przykład chcieć *zaprojektować* system tłumiący szum w elektrokardiogramie lub zwiększający ostrość obrazu, albo usuwający echo w nagraniach dźwięku. W innych przypadkach system może dawać zniekształcenia lub zakłócenia, które chcemy określić i zmierzyć. Na przykład, gdy rozmawiamy przez telefon, spodziewamy się, że odbiorca słyszy coś, co przypomina nasz głos. Niestety sygnał wejściowy linii przesyłowej rzadko jest identyczny z wyjściowym. Jeśli rozumiemy, w jaki sposób linia przesyłowa (system) powoduje zmiany w sygnale, to być może uda się skompensować ten wpływ. W jeszcze innych przypadkach system może reprezentować jakiś proces fizyczny, który chcemy badać lub analizować. Dobrymi przykładami są tu radar i sonar, w których porównuje się sygnał wysyłany z odbitym w celu określenia właściwości odległego obiektu. W kategoriach teorii systemów problem polega na znalezieniu takiego systemu, który zamienia sygnał wysyłany w sygnał odbierany.

Na pierwszy rzut oka może się wydawać, że zrozumienie wszystkich możliwych systemów jest zadaniem przytłaczającym. Na szczęście większość użytecznych systemów należy do klasy **systemów liniowych**. Jest to niezwykle ważny fakt. *Bez* koncepcji systemu liniowego bylibyśmy zmuszeni do analizowania indywidualnych właściwości wielu systemów niemających ze sobą nic wspólnego. *Mając* tę koncepcję, możemy skoncentrować uwagę na cechach wspólnych dla wszystkich systemów liniowych. Naszym pierwszym zadaniem jest stwierdzenie, jakie właściwości systemu czynią go systemem liniowym i w jaki sposób mają one odniesienie do stosowanych w praktyce układów elektronicznych, oprogramowania i innych systemów przetwarzania sygnałów.

Warunki liniowości

System jest nazywany *liniowym*, jeśli charakteryzują go dwie matematyczne właściwości: **homogeniczność** (czyli jednorodność) oraz **addytywność**. Jeśli można wykazać, że system ma dwie takie cechy, to jest udowodnione, że system jest liniowy. Jeśli się udowodni, że system nie ma jednej albo obu tych cech, to znaczy, że nie jest liniowy. Trzecia właściwość, **niezmienność względem przesunięcia**, nie jest ścisłym warunkiem liniowości, lecz cechą wymaganą w wielu metodach DSP. Gdy w dziedzinie DSP spotykamy pojęcie *system liniowy*, to powinniśmy zakładać, że charakteryzuje się on *niezmiennością względem przesunięcia*, jeśli nie znamy żadnych faktów świadczących przeciw temu. Wymienione trzy właściwości są matematyczną podstawą zdefiniowania i zastosowania teorii systemów liniowych. W dalszej



Rys. 5.2. Definicja homogeniczności. System jest homogeniczny, jeśli zmiana wartości sygnału na wejściu powoduje taką samą zmianę na wyjściu. To znaczy, że jeśli x[n] daje y[n], to kx[n] powinien dawać ky[n] dla dowolnego sygnału x[n] i dowolnej stałej k

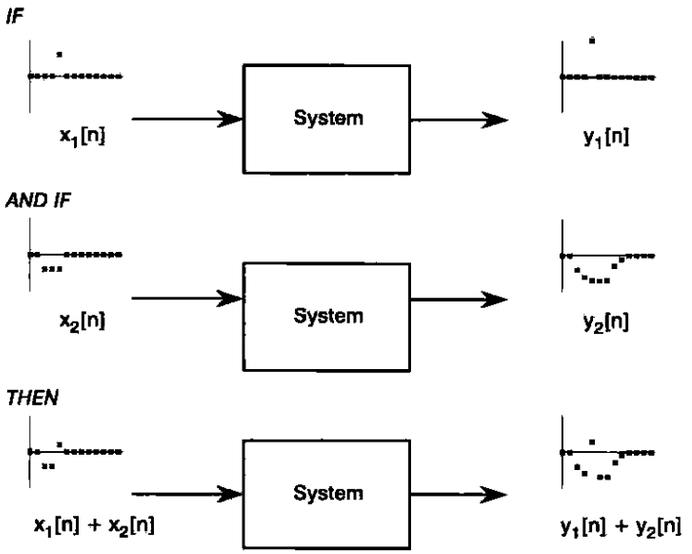
części tego rozdziału rozpatrzymy bardziej intuicyjne sposoby rozumienia liniowości. W razie rozważmy matematyczne właściwości systemów liniowych.

Jak to przedstawiono na rys. 5.2, homogeniczność oznacza, że zmiana wartości sygnału wejściowego powoduje odpowiadającą jej zmianę sygnału wyjściowego. Wyrażając to w formie matematycznej, można stwierdzić, że jeśli sygnał wejściowy $x[n]$ daje sygnał wyjściowy $y[n]$, to sygnał wejściowy $kx[n]$ powinien dawać na wyjściu sygnał $kx[n]$ dla dowolnego sygnału wejściowego i dowolnej stałej k.

Prosty rezystor może być dobrym przykładem systemu zarówno homogenicznego, jak i niehomogenicznego. Jeśli wejściem systemu jest napięcie $u(t)$ na rezystorze, a sygnałem wyjściowym prąd $i(t)$ płynący przez rezystor, to system jest homogeniczny. Wynika to z prawa Ohma – wzrost lub spadek napięcia wywołują proporcjonalny wzrost lub spadek prądu. Rozważmy teraz inny system, w którym sygnałem wejściowym jest napięcie $u(t)$ na rezystorze, a wyjściowym moc $p(t)$ rozpraszana w rezystorze. Ponieważ moc jest proporcjonalna do kwadratu napięcia, więc jeśli sygnał wejściowy wzrośnie *dwukrotnie*, to wyjściowy wzrośnie *czterokrotnie*. A zatem system nie jest homogeniczny, tak więc nie może być liniowy.

Właściwość addytywności zilustrowano na rys. 5.3. Rozpatrzmy system, w którym sygnał wejściowy $x_1[n]$ daje wyjściowy $y_1[n]$. Załóżmy następnie, że inny sygnał wejściowy $x_2[n]$ daje na wyjściu $y_2[n]$. System jest *addytywny*, jeśli sygnał wejściowy $x_1[n] + x_2[n]$ daje na wyjściu $y_1[n] + y_2[n]$ dla wszystkich możliwych sygnałów wejściowych. Mówiąc prościej, sygnały zsumowane na wejściu dają na wyjściu sygnały, które też się sumują.

Ważne jest, żeby sygnały *zsumowane* przechodziły przez system bez wzajemnego oddziaływania na siebie. Dla przykładu pomyślmy o rozmowie telefonicznej waszej ciotki Edny z wujkiem Berniem. Ciotka Edna rozpoczyna dość rozwlekłą opowieść o tym, jak świetnie w tym roku udały się jej rzodkiewki. W tle tej opowieści wujek Bernie krzyczy na psa, który właśnie narobił na ulubiony fotel wujka. Dwa sygnały głosowe są dodawane do siebie i przesyłane elektroniczne siecią telefoniczną. Ponieważ system jest addytywny, więc dźwięk, który



Rys. 5.3. Definicja addytywności. System jest addytywny, jeśli sygnały zsumowane przechodzą przez system bez wzajemnego oddziaływania na siebie. Formalnie znaczy to, że jeśli $x_1[n]$ daje na wyjściu $y_1[n]$, a $x_2[n]$ daje $y_2[n]$, to $x_1[n] + x_2[n]$ daje $y_1[n] + y_2[n]$

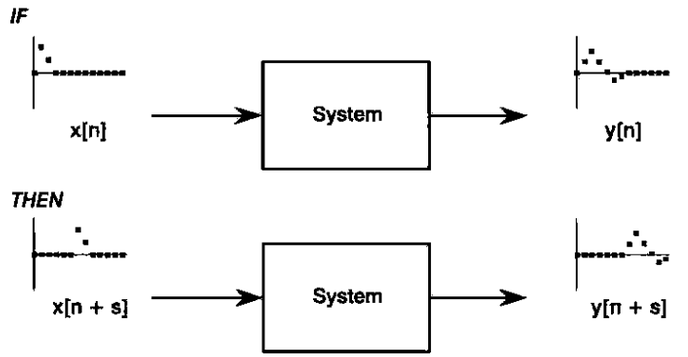
ry słyszycie, jest sumą obu tych głosów, takich, jakby brzmiały przesyłane indywidualnie. Słyszysz *Ednę* i *Berniego*, a nie jakiś dziwny twór *Ednabernie*.

Dobrym przykładem systemu *nieaddytywnego* jest stopień mieszający (modulator) w nadajniku radiowym. Są tam dwa sygnały: sygnał foniczny zawierający głos lub muzykę oraz fala nośna, która – doprowadzona do anteny – może rozchodzić się w przestrzeni. Oba sygnały są sumowane na elemencie o charakterystyce nieliniowej, takim jak np. dioda złączowa *pn*. W rezultacie następuje *złączenie* sygnałów dające w wyniku trzeci sygnał – zmodulowaną falę radiową, zdolną do przenoszenia informacji na duże odległości.

Jak wynika z rys. 5.4, niezmienność względem przesunięcia polega na tym, że przesunięcie sygnału wejściowego nie będzie powodowało niczego innego poza identycznym przesunięciem w sygnale wyjściowym. Ujmując to bardziej formalistycznie, stwierdzamy, że jeśli sygnał wejściowy $x[n]$ daje na wyjściu $y[n]$, to sygnał wejściowy $x[n + s]$ powinien dawać na wyjściu $y[n + s]$ dla dowolnego sygnału wejściowego i dowolnej stałej s. Proszę zwrócić szczególną uwagę na matematyczny zapis tego przesunięcia, gdyż będzie to potrzebne w dalszych rozdziałach. Przez dodanie stałej s do zmiennej niezależnej n przebieg może być przesunięty w przód lub w tył w kierunku poziomym. Na przykład, gdy $s = 2$, to sygnał jest przesunięty *w lewo* o dwie próbki, a gdy $s = -2$ – o dwie próbki *w prawo*.

Niezmienność względem przesunięcia jest ważną cechą, gdyż oznacza, że właściwości sygnału nie zmieniają się w funkcji czasu (lub jakiejkolwiek innej zmiennej niezależnej). Jeśli jakiś *blip* na wejściu wywołuje *blop* na wyjściu, to możemy być pewni, że inny *blip* da na wyjściu identyczny *blop*. Większość systemów, z którymi się spotykamy, charakteryzuje się niezmiennością względem przesunięcia. Jest to bardzo korzystne, gdyż trudno posługiwać się systemami, których właściwości zmieniają się podczas pracy. Wyobraźmy sobie na przykład, że zaprojektowaliśmy filtr cyfrowy kompensujący niekorzystny wpływ telefonicznej linii przesyłowej na przesyłany sygnał. Filtr powoduje, że głos brzmi bardziej naturalnie i jest łatwiejszy do zrozumienia. Tymczasem z nadejściem zimy, ku naszemu zdumieniu, właściwości linii uległy zmianie ze zmianą temperatury. Nasz filtr kompensujący staje się wówczas niedopasowany i pracuje niezbyt dobrze. Taka sytuacja może wymagać bardziej wysublimowanego algorytmu, *adaptującego się* do zmieniających warunków.

Dlaczego homogeniczność i addytywność mają krytyczne znaczenie dla liniowości, podczas gdy niezmienność względem przesunięcia ma tylko znaczenie uboczne? Jest tak dlatego, gdyż liniowość jest koncepcją bardzo szeroką, obejmującą obszar znacznie większy niż tylko sygnały i systemy. Rozważmy przykład farmy sprzedającej pomarań-



Rys. 5.4. Definicja niezmienności względem przesunięcia. Mówi się o systemie, że jest niezmienny względem przesunięcia, jeśli przesunięcie sygnału wejściowego powoduje identyczne przesunięcie sygnału wyjściowego. W ujęciu matematycznym: jeśli $x[n]$ daje $y[n]$, to $x[n + s]$ powinno dawać $y[n + s]$ dla dowolnego sygnału $x[n]$ i dowolnej stałej s

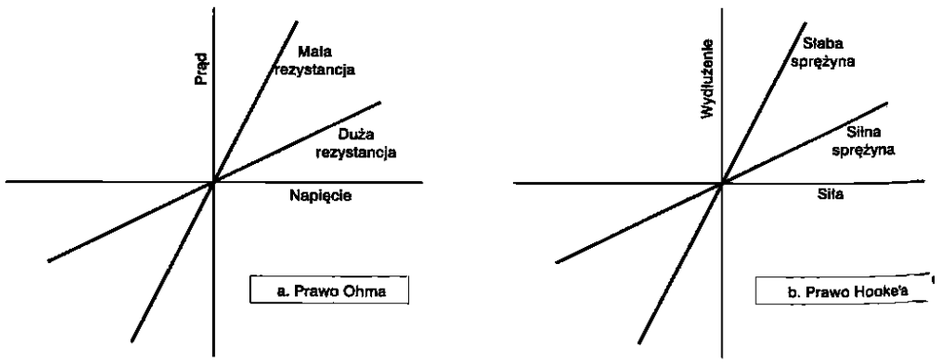
cze po 2 USD za skrzynkę i jabłka po 5 USD za skrzynkę. Jeśli farmer sprzedaje tyl pomarańcze, uzyskuje wtedy 20 USD za 10 skrzynek i 40 USD za 20 skrzynek, a w cała wymiana ma charakter *homogeniczny*. Jeśli sprzeda on 20 skrzynek pomarańcz i skrzynek jabłek, to otrzyma: 20 × 2 USD + 10 × 5 USD = 90 USD. Jest to taka sa kwota, jak w przypadku, gdyby pomarańcze i jabłka były sprzedane oddzielnie, a w transakcja jest *addytywna*. Będąc zarówno homegeniczną, jak i addytywną, ta sprzed towarów jest procesem liniowym. Ponieważ jednak w tym procesie nie ma żadnych s gnałów, więc nie mamy tu do czynienia z *systemem* i *niezmienność względem prze nięcia* nie ma żadnego sensu. Niezmienność względem przesunięcia można uważać dodatkowy aspekt liniowości, wymagany w przypadku sygnałów i systemów.

Liniowość statyczna i wierność odtwarzania sinusoidy

Homogeniczność, addytywność i niezmienność względem przesunięcia są ważnymi wł ściwościami, gdyż stanowią matematyczną podstawę definiowania systemów liniowy Niestety, same te cechy nie dają naukowcom i inżynierom intuicyjnego wyczucia, czym są systemy liniowe. Często pomocą mogą być **liniowość statyczna** i **wierność odtwarzania sinusoidy**. Nie są to właściwości szczególnie ważne z punktu widzenia matematyczne lecz odnoszą się do ludzkiego sposobu myślenia o systemach liniowych. Należy poświęc szczególną uwagę temu podrozdziałowi.

Liniowość statyczna określa, jak system liniowy reaguje na sygnały niezmieniające się, *stałoprądowe* (DC) lub *statyczne*. Odpowiedź statyczna systemu liniowego jest bardzo pros *sygnał wyjściowy jest sygnałem wejściowym pomnożonym przez pewną stałą*. A więc wyk wartości wyjściowych w funkcji wszystkich możliwych wartości wejściowych jest linią pros przechodzącą przez środek układu współrzędnych. Takie odpowiedzi przedstawiono na ry 5.5 dla dwóch często spotykanych systemów liniowych: prawa Ohma dla rezystorów i praw Hooke'a dla sprężyn. Dla porównania na rys. 5.6 przedstawiono charakterystyki statyczn dwóch systemów nieliniowych: diody złączowej *pn* i właściwości magnetycznych żelaza.

Wszystkie systemy liniowe mają właściwość *liniowości statycznej*. Stwierdzenie odwrotn też zwykle jest prawdziwe, lecz nie zawsze. Bywają systemy, które wykazują liniowo statyczną, lecz nie są liniowe w odniesieniu do sygnałów zmieniających się. Jednak ko rzystając tylko z liniowości statycznej, można w pełni zrozumieć dużą klasę system W tych systemach nie ma znaczenia, czy sygnał wejściowy jest statyczny, czy się zmien Są to tak zwane systemy **bezpamięciowe** (*memoryless*), gdyż wyjście zależy tylko od a



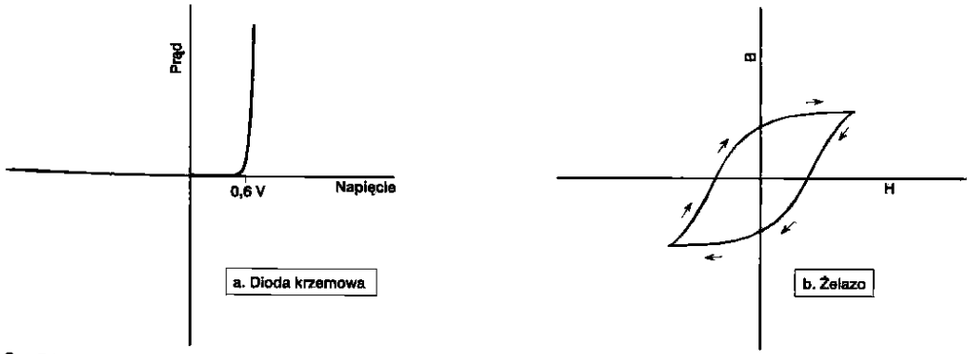
Rys. 5.5. Dwa przykłady liniowości statycznej: a) prawo Ohma – natężenie prądu płynącego przez rezystor jest równe napięciu na rezystorze podzielonemu przez rezystancję; b) Prawo Hooke'a – wydłużenie sprężyny jest proporcjonalne do przyłożonej siły pomnożonej przez współczynnik sprężystości sprężyny

tualnego stanu wejścia, a nie od jego historii. Na przykład chwilowa wartość prądu w rezystorze zależy tylko od chwilowej wartości spadku napięcia na nim, a nie od tego, w jaki sposób te wielkości uprzednio osiągnęły swoje aktualne wartości. Jeśli system charakteryzuje się liniowością statyczną i jest bezpamięciowy, to musi być systemem liniowym. To daje ważny sposób zrozumienia (i udowodnienia) liniowości tych prostych systemów.

Ważną cechą systemów liniowych jest ich zachowanie się w przypadku sygnałów sinusoidalnych. Jest to właściwość zwana **wiernością odtwarzania sinusoidy**. *Jeśli sygnałem wejściowym systemu liniowego jest przebieg sinusoidalny, to na wyjściu też będzie przebieg sinusoidalny o częstotliwości dokładnie takiej samej, jak przebiegu wejściowego*. Sinusoida jest jedynym przebiegiem, który ma taką właściwość. Nie ma na przykład żadnego powodu, aby spodziewać się, że przebieg prostokątny doprowadzony do wejścia systemu liniowego da na wyjściu również przebieg prostokątny. Chociaż wejściowy przebieg sinusoidalny z pewnością da sinusoidę na wyjściu, to jednak te dwie sinusoidy mogą różnić się *amplitudą* i *fazą*. Jest to w elektronice rzecz znana. Układ może być opisany przez *odpowiedź częstotliwościową*, czyli przez wykresy zmian wzmocnienia i kąta fazowego w funkcji częstotliwości.

Zastanówmy się teraz nad pytaniem odwrotnym – jeśli system zawsze daje na wyjściu przebieg sinusoidalny w odpowiedzi na sinusoidę na wejściu, to czy taki system jest na pewno liniowy? Odpowiedź jest „nie", choć wyjątki są rzadkie i na ogół oczywiste. Wyobraźmy sobie na przykład ukrytego wewnątrz systemu diabełka, który chce nas wprowadzić w błąd. Diabełek dysponuje oscyloskopem do obserwacji sygnału wejściowego oraz generatorem przebiegu sinusoidalnego do wytwarzania sygnału wyjściowego. Gdy wprowadzamy sygnał sinusoidalny na wejście, diabełek szybko mierzy częstotliwość i tak ustawia swój generator, aby wytwarzał odpowiedni sygnał wyjściowy. Oczywiście ten system nie jest liniowy, gdyż nie jest addytywny. Aby to wykazać, wprowadźmy do systemu sumę dwóch przebiegów sinusoidalnych. Diabełek może odpowiedzieć na wyjściu tylko jednym przebiegiem sinusoidalnym. Ten przykład wcale nie jest tak bardzo zmyślony, jak by się mogło wydawać. Na podobnej zasadzie działają układy z pętlą synchronizacji fazowej (PLL – *Phase Locked Loops*).

Aby lepiej uchwycić pojęcie liniowości, pomyślmy o techniku próbującym stwierdzić, czy jego urządzenie jest liniowe. Technik do wejścia urządzenia dołączy generator przebiegu sinusoidalnego, a do wyjścia – oscyloskop. Mając sinusoidę na wejściu urządzenia, technik będzie sprawdzał, czy występuje ona także na wyjściu. Przebieg, na przykład, nie powinien być obcięty ani od góry, ani od dołu, górna połówka nie może różnić się od



Rys. 5.6. Dwa przykłady nieliniowości DC: a) dioda krzemowa ma wykładniczą charakterystykę prądowo-napięciową; b) charakterystyka indukcji magnetycznej B w funkcji natężenia pola magnetycznego H w żelazie zależy od historii danej próbki, takie zachowanie nazywamy histerezą

dolnej, nie powinny występować zniekształcenia przy przekraczaniu zera itd. Następnie technik będzie zmieniał amplitudę przebiegu i obserwował wpływ tych zmian na przebieg wyjściowy. Jeśli system jest liniowy, to zmiany amplitudy na wyjściu powinny śledzić zmiany wejściowe. W końcu technik będzie zmieniał częstotliwość przebiegu i sprawdzał, czy zmiany częstotliwości na wyjściu są identyczne. Przy zmianach częstotliwości mogą na wyjściu być widoczne zmiany amplitudy i fazy, lecz to jest w pełni dopuszczalne w systemie liniowym. Przy pewnych częstotliwościach sygnał wyjściowy może być nawet *zerowy*, to znaczy może być sinusoidą o amplitudzie zero. Jeśli technik sprawdzi te wszystkie fakty, to dojdzie do wniosku, że system jest liniowy. Ten wniosek nie jest wprawdzie ścisłym dowodem matematycznym, lecz jego poziom ufności jest – w sposób uzasadniony – wysoki.

Tab. 5.1. Przykłady systemów liniowych i nieliniowych. Formalnie, systemy liniowe charakteryzują się homogenicznością, addytywnością i niezmiennością względem przesunięcia. Nieformalnie, większość naukowców i inżynierów myśli o systemach liniowych w kategoriach liniowości statycznej i wierności odtwarzania przebiegu sinusoidalnego

[illegible]
Propagacja fal, np. dźwiękowych lub elektromagnetycznych
Układy elektryczne złożone z rezystorów, kondensatorów i cewek indukcyjnych
Układy elektroniczne, takie jak wzmacniacze i filtry
Ruch mechaniczny wywołany oddziaływaniem mas, sprężyn i tłumików
Systemy opisywane równaniami różniczkowymi, np. sieci rezystor – kondensator – cewka indukcyjna
Mnożenie przez stałą wartość, tj. wzmacnianie lub tłumienie (dzielnikowanie) sygnałów
Zmiany zachodzące w sygnałach, takie jak echa, rezonanse, rozmazania obrazu
System o wzmocnieniu 1, w którym sygnał wyjściowy jest zawsze równy wejściowemu
System o wyjściu zerowym, w którym sygnał wyjściowy jest zawsze równy zero, niezależnie od wejścia
Różniczkowanie i całkowanie oraz analogiczne operacje – pierwsza różnica (first difference) i suma bieżąca (running sum) dla sygnałów dyskretnych
Małe odchylenia sygnału w systemach, które – ogólnie biorąc, są nieliniowe, np. mały sygnał wzmacniany w prawidłowo spolaryzowanym tranzystorze
Splot, operacja matematyczna, w której każda wartość wyjściowa jest wyrażana jako suma wartości na wejściu przemnażanych przez pewien zbiór współczynników wagowych
Rekursja, operacja podobna do splotu, z tą jednak różnicą, że oprócz wartości na wejściu stosuje się poprzednio obliczone wartości wyjściowe
[illegible]
Systemy, które nie charakteryzują się liniowością statyczną, np. napięcie i moc w rezystorze: $P = U^2/R$, zależność emisji energii promieniowanej z ciała nagrzanego w zależności od temperatury: $R = kT^4$, natężenie światła po przejściu przez materiał półprzezroczysty: $I = I_0 e^{-T}$ itd.
Systemy, które nie charakteryzują się wiernością odtwarzania sygnału sinusoidalnego, takie jak detektory wartości szczytowej, układy podnoszące do kwadratu, konwertery przebiegu sinusoidalnego na prostokątny, podwajacze częstotliwości itd.
Zniekształcenia często spotykane w układach elektronicznych, takie jak obcinanie, zniekształcenia skrośne i ograniczenie szybkości zmian sygnału
Mnożenie dwóch sygnałów, np. modulacja amplitudy i automatyczna regulacja wzmocnienia
Zjawiska związane z histerezą, np. zależność indukcji magnetycznej od natężenia pola magnetycznego w żelazie lub zależność między naprężeniem mechanicznym a odkształceniem w wulkanizowanej gumie
Nasycenie, np. w przesterowanych wzmacniaczach elektronicznych lub transformatorach
Systemy z progiem, np. cyfrowe bramki logiczne lub wibracje sejsmiczne na tyle silne, że mogące skruszyć przeszkadzające skały

Przykłady systemów liniowych i nieliniowych

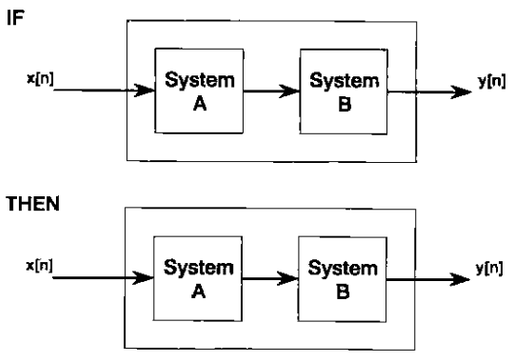
W tablicy 5.1 podano przykłady częstospotykanych systemów liniowych i nieliniowych. Przeglądając ten wykaz, pamiętajmy o matematycznych aspektach liniowości (*homogeniczność*, *addytywność* oraz *niezmienność względem przesunięcia*), jak również o nieformalnych sposobach określania liniowości stosowanych przez większość naukowców i inżynierów (*liniowość statyczna* i *wierność odtwarzania sinusoidy*).

Specjalne cechy liniowości

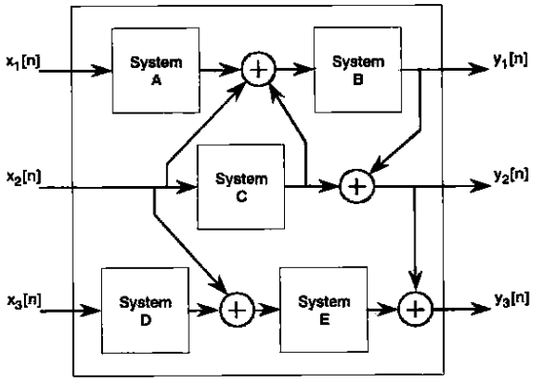
Liniowość jest cechą **przemienną**, a więc obejmującą dwa lub więcej systemów. Koncepcje przemienności przedstawiono ogólnie na rys. 5.7. Wyobraźmy sobie dwa systemy połączone **w kaskadę**, tzn. wyjście jednego systemu jest wejściem następnego. Jeśli każdy z systemów jest liniowy, to cały połączony system też będzie liniowy. Z cechy przemienności wynika, że zmiana kolejności systemów w kaskadzie nie wpływa na właściwości całego połączonego systemu. Przypuszczalnie stosowaliście już tę zasadę w układach elektronicznych. Wyobraźmy sobie na przykład układ składający się z dwóch stopni – wzmacniacza i filtru. Jak jest lepiej – najpierw wzmacniać, potem filtrować, czy filtrować i następnie wzmacniać? Jeśli oba stopnie są liniowe, to ich kolejność w kaskadzie nie ma żadnego znaczenia i sygnał uzyskany na wyjściu jest w obu przypadkach taki sam. Trzeba jednak pamiętać, że we współczesnej elektronice występuje wiele zjawisk o charakterze nieliniowym, które powodują, że kolejność układów w systemie jest istotna. Takie zjawiska to np.: zakłócenia, przesunięcie stałoprądowe (*offset DC*), szumy wewnętrzne, zniekształcenia spowodowane ograniczeniem szybkości zmian sygnału.

Na rys. 5.8 zilustrowano następny krok w teorii systemów liniowych – wielokrotne wejścia i wyjścia. System z wielokrotnymi wejściami i/lub wyjściami jest liniowy, jeśli składa się z *liniowych podsystemów* i *sumowań sygnałów*. Złożoność sytemu nie ma znaczenia, wszelkie *nieliniowości* są w nim jednak niedopuszczalne.

Przeprowadzimy pewien eksperyment myślowy ułatwiający zrozumienie, jakie jest znaczenie liniowości w systemach z wielokrotnymi wejściami i/lub wyjściami. Zacznijmy od umieszczenia sygnału na jednym z wejść, podczas gdy na pozostałych jest utrzymywany sygnał zerowy. Wtedy wielokrotne wyjścia odpowiedzą taką samą kombinacją sygnałów. Powtórzmy następnie tę procedurę, umieszczając inny sygnał na innym wejściu. Tak jak poprzednio, na pozostałych wejściach będzie utrzymywany sygnał zerowy. Ten drugi sy-



Rys. 5.7. Cecha przemienności systemów liniowych. Gdy dwa lub więcej systemów liniowych są połączone kaskadowo, to kolejność systemów w kaskadzie nie wpływa na właściwości całego połączonego systemu

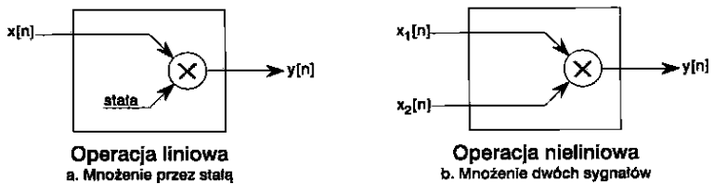


Rys. 5.8. Każdy system o wielokrotnych wejściach i/lub wyjściach jest liniowy, jeśli składa się z liniowych podsystemów i sumowań sygnałów

gnał wejściowy da w rezultacie inną kombinację sygnałów na wielokrotnych wyjściach. Kończąc nasz eksperyment, umieśćmy jednocześnie oba sygnały odpowiednio na dwóch wejściach. Sygnały pojawiające się na wyjściach będą *superpozycją* (sumą) *sygnałów wyjściowych* wytwarzanych w sytuacjach, gdy sygnały wejściowe były dołączane oddzielnie.

Często błędnie rozumiane jest stosowanie mnożenia w systemach liniowych. Przyczyną jest fakt, że mnożenie może być zarówno liniowe, jak i nieliniowe, zależnie od właściwości mnożonych sygnałów. Na rys. 5.9 przedstawiono dwa przypadki. System mnożący sygnał wejściowy przez pewną *stałą* jest liniowy. Taki system jest wzmacniaczem albo tłumikiem (dzielnikiem), zależnie od tego, czy stała jest większa, czy mniejsza od jedności. Natomiast mnożenie sygnału przez *drugi* sygnał jest nieliniowe. Wyobraźmy sobie mnożenie sinusoidy przez drugą sinusoidę o innej częstotliwości, wynikowy przebieg z pewnością nie jest sinusoidą.

Inna często błędnie rozumiana kwestia dotyczy sygnałów pasożytniczych występujących w układach elektronicznych, np. napięć niezrównoważenia, szumów cieplnych itd. Czy dodawanie się takich obcych sygnałów do sygnału użytecznego jest procesem liniowym, czy nieliniowym? Odpowiedź zależy od miejsca pochodzenia tych sygnałów. Jeśli traktuje się je jako pochodzące z *wnętrza* systemu, to proces jest nieliniowy, gdyż sinusoidalny sygnał wejściowy nie wytwarza idealnie sinusoidalnego sygnału wyjściowego. Można jednak też traktować obcy sygnał jako wchodzący *z zewnątrz* systemu do oddzielnego wejścia systemu wielowejściowego. Wtedy proces staje się liniowy, ponieważ w systemie jest wymagane tylko dodawanie.



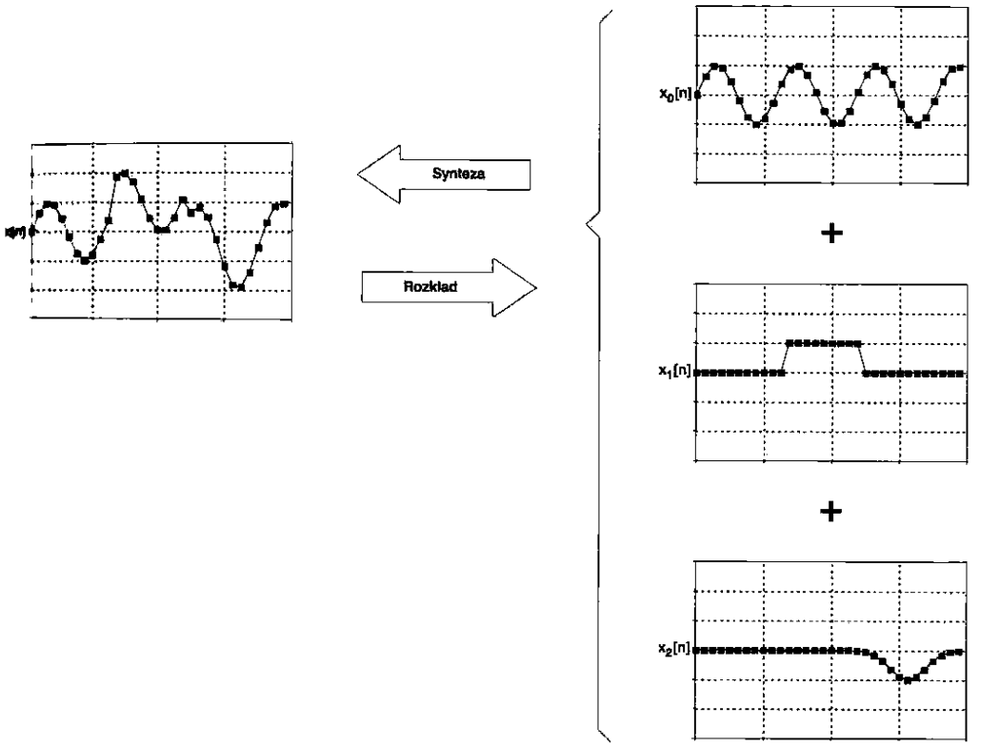
Rys. 5.9. Liniowość mnożenia. Mnożenie sygnału przez pewną stałą jest operacją liniową. Natomiast mnożenie dwóch sygnałów jest operacją nieliniową

Zasada superpozycji – podstawa DSP

W przypadku systemów liniowych jedynym sposobem łączenia sygnałów z sobą jest ich *skalowanie* (mnożenie sygnałów przez stałe współczynniki) i następujące po nim *sumowanie*. Nie można na przykład mnożyć jednego sygnału przez inny. Przykład przedstawiono na rys. 5.10. Trzy sygnały $x_0[n]$, $x_1[n]$ i $x_2[n]$ są sumowane, dając w wyniku czwarty sygnał $x[n]$. Taki proces łączenia sygnałów ze sobą drogą skalowania i sumowania jest nazywany **syntezą**.

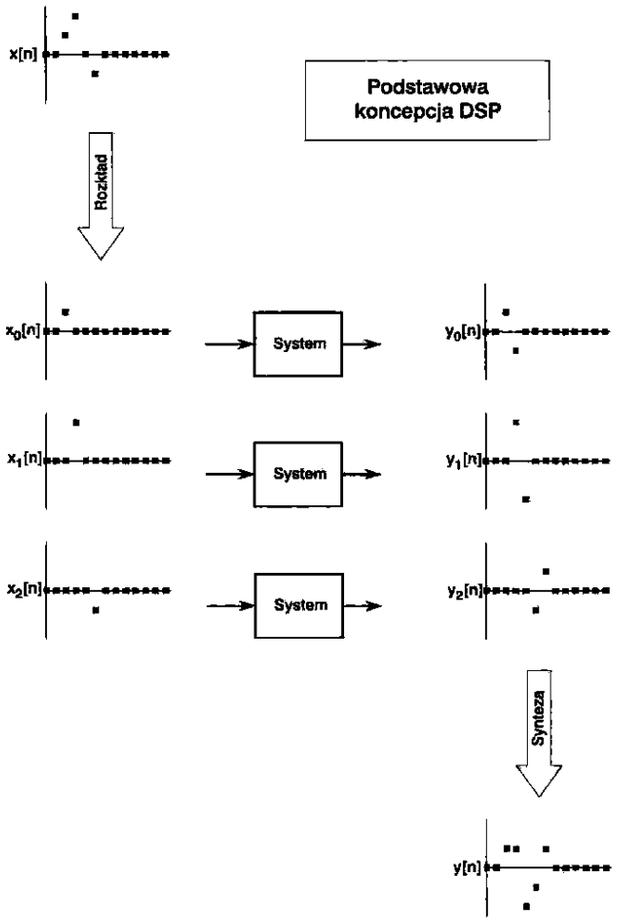
Rozkład (czyli dekompozycja) sygnału jest operacją odwrotną do syntezy. Polega na rozdzieleniu sygnału na dwie lub więcej składowych addytywnych. Jest to operacja bardziej skomplikowana niż synteza, jako że jest nieskończenie wiele możliwości rozkładu danego sygnału. Na przykład liczby 15 i 25 mogą, poddane syntezie, dać w wyniku tylko liczbę 40. Liczba 40 zaś ma wiele możliwości rozkładu: 1 + 39, 2 + 38 lub –30,5 + 60 + 10,5 itd.

Dochodzimy teraz do samego jądra DSP – do zasady superpozycji, która jest drogą do zrozumienia sposobów analizy sygnałów i systemów. Rozważmy sygnał oznaczony $x[n]$, który po przejściu przez system liniowy daje sygnał wyjściowy $y[n]$. Jak to przedstawiono na rys. 5.11, sygnał wejściowy może być poddany rozkładowi na jakąś grupę sygnałów prostszych: $x_0[n]$, $x_1[n]$ i $x_2[n]$ itd. Nazwiemy je **składowymi sygnału wejściowego**. Następnie każdą składową oddzielnie przepuszczamy przez system, uzyskując pewien **zestaw składowych sygnału wyjściowego**: $y_0[n]$, $y_1[n]$, $y_2[n]$ itd. Później te składowe poddaje się syntezie, otrzymując sygnał wyjściowy $y[n]$.



Rys. 5.10. Synteza i rozkład sygnału. W wyniku syntezy dwa dodawane do siebie sygnały tworzą inny sygnał. Odwrotną operacją jest rozkład polegający na rozdzieleniu jednego sygnału na dwa lub więcej addytywnych sygnałów składowych

I teraz ważna rzecz. Uzyskany w ten sposób sygnał wyjściowy jest *identyczny* z sygnałem powstającym przy przejściu bezpośrednio sygnału wejściowego przez system. Ta zasada daje wielkie możliwości. Zamiast prób zrozumienia, jak system zmienia sygnały *skomplikowane*, wystarczy zrozumieć, jakim zmianom ulegają sygnały *proste*. W terminologii stosowanej w dziedzinie przetwarzania sygnałów sygnały wejściowy i wyjściowy traktuje się jako *superpozycję* (sumę) prostszych przebiegów. Jest to podstawą prawie wszystkich metod przetwarzania sygnałów. Prostym przykładem zastosowania zasady superpozycji niech będzie mnożenie w pamięci liczby 2041 przez 4. Jak to zrobić? Można sobie wyobrazić 2041 zapałek fruwających w naszym umyśle, zwiększyć (myślowo) ich liczbę czterokrotnie i rozpocząć liczenie. Z pewnością jednak, dla uproszczenia sobie zadania, skorzystamy z zasady superpozycji. Przeprowadzimy rozkład liczby 2041 na 2000 + 40 + 1, każdą składową pomnożymy przez 4 i w wyniku syntezy uzyskamy końcowy rezultat, tj. 8000 + 160 + 4 = 8164.



Rys. 5.11. Podstawowa koncepcja DSP. Każdy sygnał, taki jak x[n], może być rozłożony na grupę składowych addytywnych, pokazanych na rysunku jako sygnały $x_0[n]$, $x_1[n]$ i $x_2[n]$. Przejście tych składowych przez system liniowy daje sygnały $y_0[n]$, $y_1[n]$ i $y_2[n]$. Synteza (dodawanie) tych sygnałów wyjściowych daje sygnał y[n], taki sam jak wytwarzany przy przejściu przez system sygnału x[n]

Metody rozkładu

Pamiętajmy, że celem tej operacji jest zastąpienie jednego skomplikowanego problemu kilkoma łatwymi. Jeśli rozkład nie upraszcza w jakiś sposób zadania, to niczego się nie zyskuje. Są dwie podstawowe metody rozkładu sygnałów stosowane w DSP – *rozkład impulsowy* oraz *fourierowski*. Będą one opisane w następnych kilku rozdziałach. Oprócz tego jest kilka innych, rzadziej spotykanych sposobów rozkładu sygnału.

Rozkład impulsowy

W rozkładzie impulsowym (rys. 5.12) rozdziela się sygnał N-próbkowy na N sygnałów składowych, z których każdy zawiera N próbek. Każdy z sygnałów składowych zawiera jeden punkt z sygnału oryginalnego, a pozostałe wartości są zerami. Pojedynczy punkt niezerowy w łańcuchu zer jest nazywany *impulsem*. Rozkład impulsowy ma duże znaczenie, gdyż daje możliwość sprawdzania sygnałów próbka po próbce. Również charakterystyki systemów określa się ich odpowiedziami na impulsy. Wiedząc, jaka jest odpowiedź systemu na impuls, można obliczyć sygnał wyjściowy dla każdego sygnału wejściowego. Taka metoda, nazywana *splotową*, będzie tematem dwóch następnych rozdziałów.

Rozkład skokowy

W rozkładzie skokowym (rys. 5.13) również następuje rozdzielanie sygnału N-próbkowego na N sygnałów składowych, z których każdy składa się z N próbek. Każdy sygnał składowy ma postać *skoku*, tzn. pierwsze próbki mają wartość zero, a ostatnie pewną wartość stałą. Rozpatrzmy rozkład N-punktowego sygnału $x[n]$ na składowe: $x_0[n]$, $x_1[n]$, $x_2[n]$, ...$x_{N-1}[n]$. Składowa k-ta, sygnał $x_k[n]$, składa się z zer dla punktów od 0 do $k-1$, a dla pozostałych punktów ma wartość $x[k] - x[k-1]$. Na przykład piąty sygnał składowy $x_5[n]$ składa się z zer dla punktów od 0 do 4, a pozostałe próbki mają wartość $x[5] - x[4]$, czyli wartość różnicy między próbkami 4 i 5 sygnału oryginalnego. Przypadkiem szczególnym jest $x_0[n]$, w którym wszystkie próbki są równe $x[0]$. O ile w rozkładzie impulsowym przegląda się sygnały punkt po punkcie, to w rozkładzie skokowym sygnały są charakteryzowane przez *różnice* między sąsiednimi próbkami. A zatem systemy są określane na podstawie ich odpowiedzi na *zmianę* sygnału wejściowego.

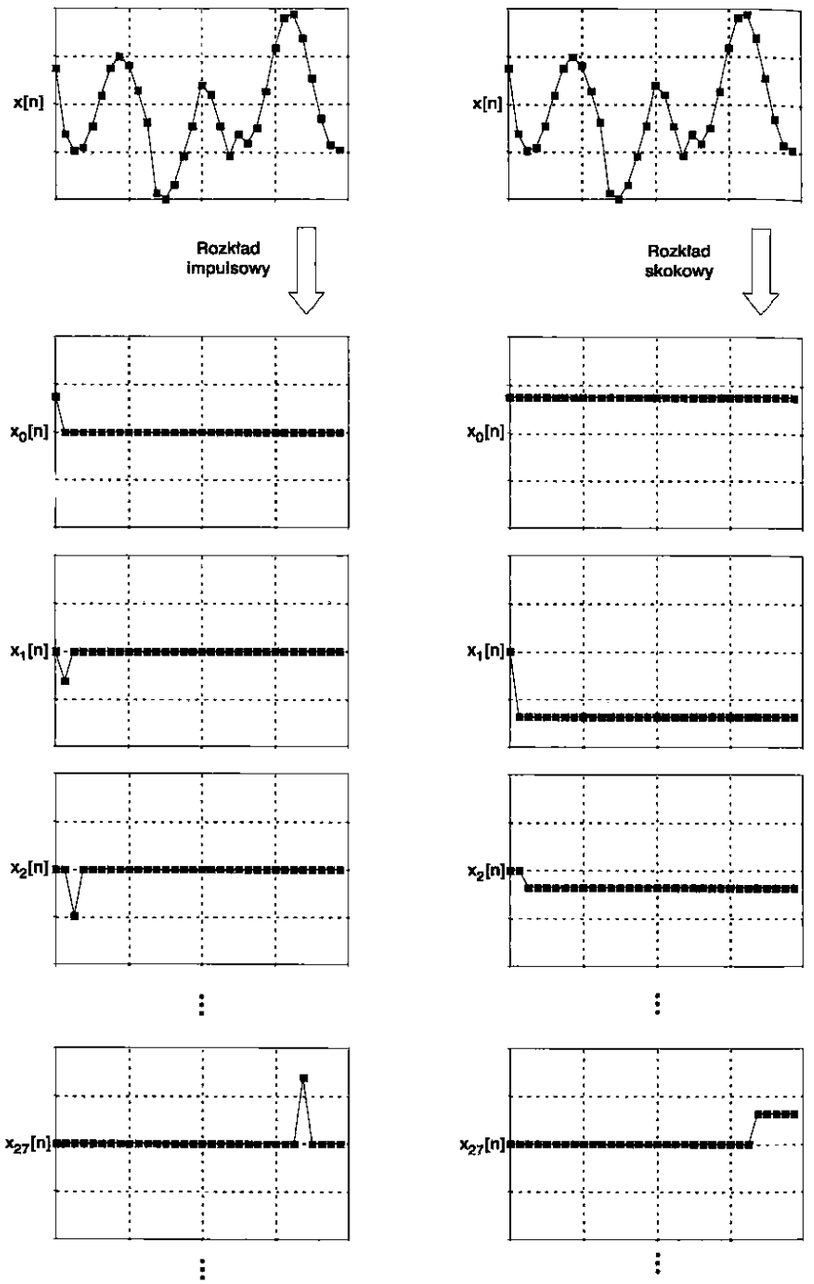
Rozkład parzysto-nieparzysty

W rozkładzie parzysto-nieparzystym, przedstawionym na rys. 5.14, rozdziela się sygnał na dwa sygnały składowe, jeden o symetrii **parzystej**, drugi – o **nieparzystej**. O sygnale złożonym z N punktów mówi się, że ma symetrię parzystą, jeśli daje zwierciadlane odbicie wokół punktu $N/2$. To znaczy, że próbka $x[N/2 + 1]$ musi być równa próbce $x[N/2 - 1]$, próbka $x[N/2 + 2]$ – próbce $x[N/2 - 2]$ itd. Symetria nieparzysta zaś zachodzi, gdy odpowiednie punkty mają równe wartości, lecz przeciwne znaki, tak jak $x[N/2 + 1] = -x[N/2 - 1]$, $x[N/2 + 2] = -x[N/2 - 2]$ itd. W tych definicjach zakłada się, że sygnał zawiera parzystą liczbę próbek i że indeksy mają wartości od 0 do $N-1$. Rozkład parzysto-nieparzysty jest opisany wzorami:

$$x_E[n] = \frac{x[n] + x[N-n]}{2}$$

$$x_O[n] = \frac{x[n] - x[N-n]}{2}$$

Wzór 5.1. Wzory wyrażające rozkład parzysto-nieparzysty. Na podstawie tych wzorów następuje rozdzielenie sygnału x[n] na część parzystą $x_E[n]$ oraz nieparzystą $x_O[n]$. Ponieważ ten rozkład opiera się na symetrii kołowej, więc zerowe próbki sygnału parzystego i nieparzystego są obliczane jako: $x_E[0] = x[0]$ oraz $x_O[0] = 0$. Wszystkie sygnały mają długość N próbek o indeksach zmieniających się od 0 do N – 1



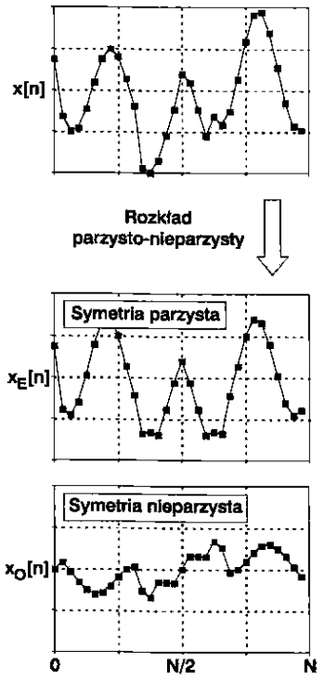
Rys. 5.12. Przykład rozkładu impulsowego. Sygnał N-próbkowy jest rozdzielany na N składowych, z których każda zawiera jeden punkt niezerowy

Rys. 5.13. Przykład rozkładu skokowego. Sygnał N-próbkowy jest rozdzielany na N sygnałów, z których każdy zawiera funkcję skokową

Definicja symetrii lewo-prawostronnej może wydawać się dziwna, gdyż dokładnym środkiem sygnału jest ($N/2 - 1/2$) (między dwiema próbkami), a nie $N/2$. Z takiej mimośrodowej symetrii wynika, że próbka zerowa wymaga specjalnego traktowania. O co tu chodzi?

Ten rozkład jest częścią ważnej w DSP koncepcji, zwanej **symetrią kołową**. Polega ona na założeniu, że *koniec* sygnału jest powiązany z jego *początkiem*. Jak punkt $x[4]$ jest obok punktu $x[5]$, tak punkt $x[N-1]$ jest obok punktu $x[0]$. Całkiem jak wąż zjadający własny ogon. Gdy w taki kołowy sposób spojrzymy na sygnały parzyste i nieparzyste, to mamy *dwie* linie symetrii, jedna w punkcie $x[N/2]$ i druga w punkcie $x[0]$. Na przykład w sygnale parzystym ta symetria wokół $x[0]$ oznacza, że punkt $x[1]$ ma wartość równą punktowi $x[N-1]$, punkt $x[2]$ punktowi $x[N-2]$ itd. W sygnale nieparzystym punkty 0 i $N/2$ zawsze mają wartość zero. W sygnale parzystym punkty 0 i $N/2$ są równe odpowiednim punktom sygnału oryginalnego.

Jaki jest cel traktowania ostatniej próbki w sygnale jako sąsiedniej w stosunku do zerowej? W konwencjonalnej akwizycji danych nie ma niczego, co by uzasadniało taki zapis kołowy. W rzeczywistości próbki pierwsza i ostatnia mają ze sobą mniej wspólnego niż którekolwiek inne punkty w szeregu. Tak mówi zdrowy rozsądek. Brakującym ogniwem w tej łamigłówce jest metoda DSP, zwana *analizą Fouriera*. Nieodłącznym elementem matematycznego ujęcia analizy Fouriera jest potraktowanie sygnału jako kołowego, choć zwykle nie ma to znaczenia fizycznego związanego z miejscem, z którego dane przychodzą. Bardziej szczegółowo rozpatrzymy tę kwestię w rozdziale 10. Na razie trzeba zrozumieć ten ważny fakt, że wzór 5.1 daje bardzo pożyteczny rozkład sygnału, po prostu dlatego, że części parzysta i nieparzysta po dodaniu do siebie odtwarzają sygnał oryginalny.



Rys. 5.14. Przykład rozkładu parzysto-nieparzystego. Sygnał N-punktowy jest rozdzielany na dwa sygnały N-punktowe, jeden o symetrii parzystej, drugi – nieparzystej

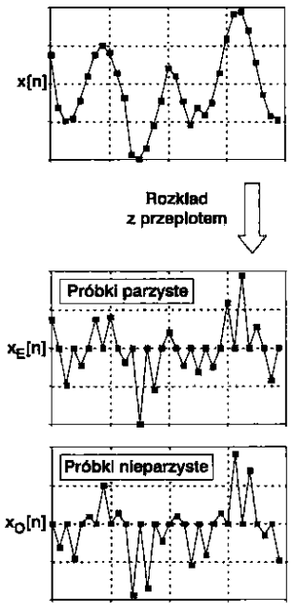
Rozkład z przeplotem

Jak przedstawiono na rys. 5.15, w rozkładzie z przeplotem sygnał jest rozkładany na dwa sygnały składowe – sygnał *próbek parzystych* oraz sygnał *próbek nieparzystych* (proszę nie mylić z sygnałami o symetrii parzystej i nieparzystej). Aby wyznaczyć sygnał próbek parzystych, trzeba zacząć od sygnału oryginalnego i ustawić na zero wszystkie próbki o numerach nieparzystych. Jest to proste.

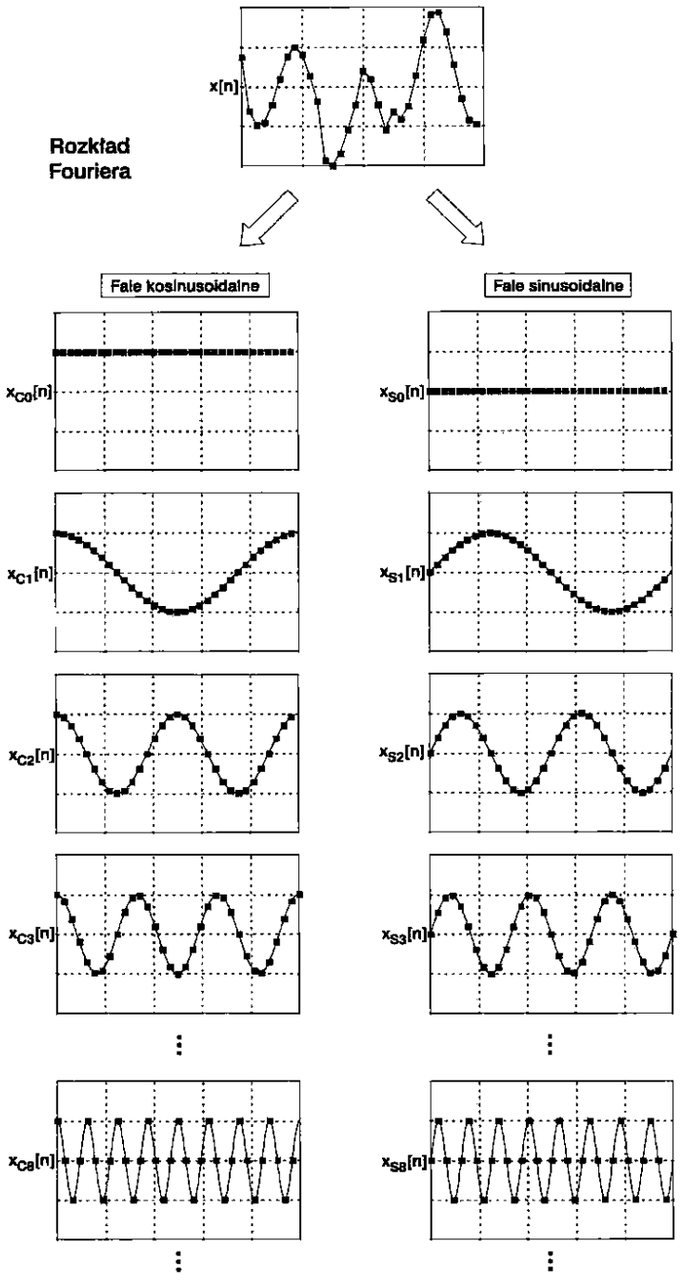
Taki rozkład może na pierwszy rzut oka wydawać się trywialny i nieinteresujący. Ale to tylko wrażenie pozorne, gdyż rozkład z przeplotem jest podstawą niezwykle ważnego w DSP algorytmu – szybkiego przekształcenia Fouriera (FFT). Procedura obliczeń rozkładu Fouriera jest znana od kilkuset lat. Niestety jest ona frustrująco powolna. Jej realizacja na współczesnych komputerach wymaga często minut lub godzin. FFT jest rodziną algorytmów opracowanych w latach 60. XX wieku w celu zredukowania czasu obliczeń. Zastosowana strategia jest świetnym przykładem DSP – zredukowanie sygnału do elementarnych składowych przez powtarzane stosowanie transformacji z przeplotem, obliczenie rozkładu Fouriera poszczególnych składowych oraz syntezę wyników dającą końcową odpowiedź. W ten sposób uzyskuje się rewelacyjną poprawę szybkości obliczeń – na ogół o czynnik rzędu *setek lub tysięcy* razy.

Rozkład Fouriera

Rozkład Fouriera ma charakter bardzo matematyczny i wcale nie jest oczywisty. Przykład tej metody przedstawiono na rys. 5.16. Każdy sygnał N-punktowy może być rozłożony na $N+2$ sygnałów, połowa z nich to przebiegi sinusoidalne a połowa kosinusoidalne. Fala kosinusoidalna o najmniejszej częstotliwości (oznaczona na rysunku jako $x_{C0}[n]$) zawiera *zero* pełnych okresów w N próbkach, jest to więc sygnał stałoprądowy. Następne składowe kosinusoidalne $x_{C1}[n]$, $x_{C2}[n]$ i $x_{C3}[n]$ zawierają, odpowiednio, 1, 2 lub pełne okresy w N próbkach. Taka zasada dotyczy też dalszych składowych zarówno kosinusoidalnych, jak i sinusoidalnych. Ponieważ częstotliwość każdej ze składowych jest ustalona, więc jedynym parametrem zmieniającym się dla różnych sygnałów jest *amplituda* każdego z przebiegów kosinusoidalnych i sinusoidalnych.



Rys. 5.15. Przykład rozkładu z przeplotem. Sygnał N-punktowy jest rozdzielany na dwa sygnały N-punktowe, jeden z próbkami nieparzystymi ustawionymi na zero, a drugi z parzystymi ustawionymi na zero



Rys. 5.16 Zilustrowanie rozkładu Fouriera. Sygnał N-punktowy jest rozkładany na N + 2 sygnałów, każdy z nich o N punktach. Połowa z tych sygnałów to przebiegi kosinusoidalne, a połowa – sinusoidalne. Częstotliwości tych sygnałów są ustalone, a amplitudy mogą się zmieniać

Rozkład Fouriera jest ważny z trzech powodów. Po pierwsze, bardzo wiele sygnałów jest z natury rzeczy, tworzonych przez nakładanie się sinusoid. Dobrym przykładem są sygnały foniczne. Rozkład Fouriera umożliwia bezpośrednią analizę informacji zawartych w sygnałach tego rodzaju. Po drugie, systemy liniowe w specyficzny sposób przenoszą przebiegi sinusoidalne; sinusoidalny przebieg wejściowy zawsze daje na wyjściu też przebieg sinusoidalny. Właściwości tych systemów są określane przez zmiany amplitudy i fazy przebiegów sinusoidalnych przez nie przenoszonych. Ponieważ sygnał wejściowy można rozłożyć na sinusoidy, więc znajomość reakcji systemu na przebiegi sinusoidalne umożliwia wyznaczenie odpowiedzi systemu. Po trzecie, rozkład Fouriera jest podstawą rozległej i ważnej dziedziny, zwanej *analizą fourierowską*, a także *przekształceń Laplace'a* i Z. W większości ciekawych algorytmów DSP korzysta się z tych metod.

Dlaczego jest możliwe rozłożenie dowolnego sygnału na przebiegi sinusoidalne i kosinusoidalne? Jak są wyznaczane amplitudy sinusoid dla poszczególnych sygnałów? Jakiego rodzaju systemy można projektować przy użyciu tej metody? Na te właśnie pytania odpowiemy w następnych rozdziałach. Szczegóły rozkładu Fouriera są zbyt skomplikowane, aby można je przedstawić w tym krótkim omówieniu. Na razie najważniejsze jest zrozumienie faktu, że można dokładnie odtworzyć przebieg oryginalny, dodając do siebie wszystkie składowe sinusoidalne. Więcej na ten temat powiemy w rozdziale 8.

Alternatywy liniowości

W celu docenienia ważności systemów liniowych zwróćmy uwagę, że jest tylko *jeden* główny sposób analizy tych systemów, które są nieliniowe. Sposób polega na uczynieniu systemów nieliniowych *podobnymi* do liniowych. Można to przeprowadzić jednym z trzech sposobów. Po pierwsze, można zignorować nieliniowość. Jeśli nieliniowość jest wystarczająco mała, to system można aproksymować jako liniowy. Błędy wynikające ze wstępnego założenia są traktowane jako szum lub po prostu pomijane.

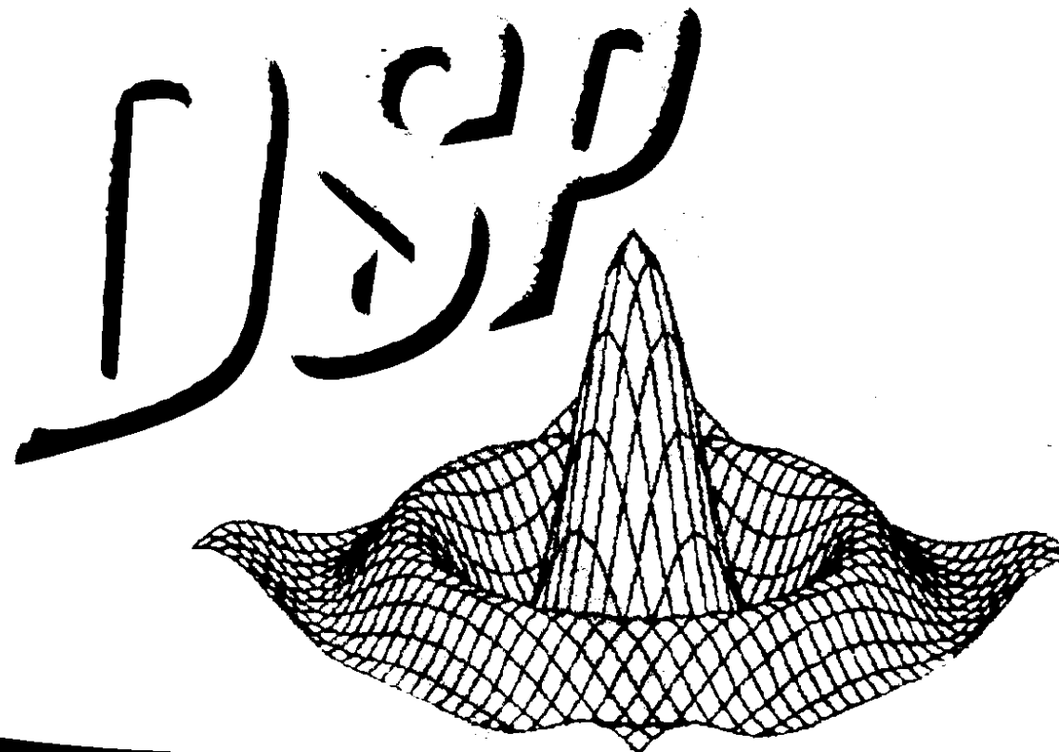
Po drugie, można przyjąć, że sygnały są bardzo małe. Wiele systemów nieliniowych można traktować jako liniowe, jeśli sygnały mają bardzo małe wartości. Na przykład tranzystory są bardzo nieliniowe w pełnym zakresie roboczym, lecz dają dokładne, liniowe wzmacnianie, gdy sygnały są nie większe niż kilka miliwoltów. Ta cecha jest w największym stopniu widoczna we wzmacniaczach operacyjnych. Stosując bardzo duże wzmocnienie z otwartą pętlą oraz ujemne sprzężenie zwrotne, utrzymuje się sygnał wejściowy wzmacniacza (a więc różnicę napięć między wejściem odwracającym i nieodwracającym) na poziomie kilku mikrowoltów. Ten bardzo mały sygnał wejściowy powoduje, że liniowość układu, w innych przypadkach bardzo nieliniowego, jest doskonała.

Po trzecie, można stosować przekształcenia linearyzujące. Rozważmy na przykład dwa sygnały, które po przemnożeniu dają trzeci: $a[n] = b[n] \times c[n]$. Zlogarytmowanie sygnałów zmienia nieliniowy proces mnożenia w liniowy proces dodawania: $\log(a[n]) = \log(b[n]) + \log(c[n])$. Ta metoda jest nazywana *homomorficznym* przetwarzaniem sygnałów. Na przykład obraz wizualny może być modelowany jako dwuwymiarowy sygnał odbity od oglądanej sceny pomnożony przez oświetlenie otoczenia (drugi sygnał również dwuwymiarowy). Dzięki technice homomorficznej sygnał oświetlenia można uczynić bardziej równomiernym, poprawiając w ten sposób jakość obrazu.

W następnych rozdziałach zajmiemy się dwiema najważniejszymi metodami przetwarzania sygnałów – analizą *fourierowską* i *splotową*. Obie są oparte na procedurze opisanej w tym rozdziale: 1) rozłożenie sygnału na proste składowe addytywne, 2) przetworzenie składowych w jakiś pożyteczny sposób oraz 3) synteza składowych w celu uzyskania wyniku końcowego. To jest właśnie DSP.

6

Splot



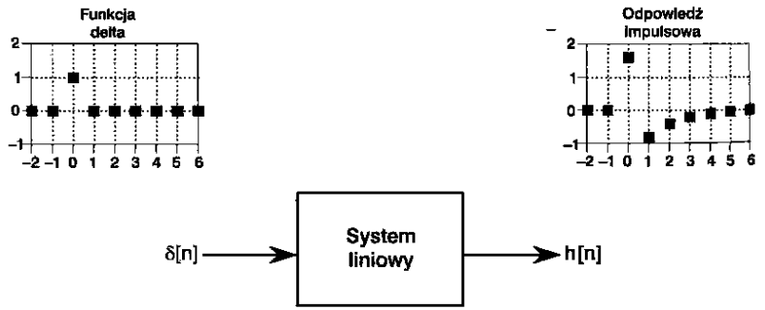
Splot jest matematyczną operacją łączenia dwóch sygnałów, dającą w wyniku trzeci sygnał. Jest to najważniejsza metoda w cyfrowym przetwarzaniu sygnałów. Stosując rozkład impulsowy, opisuje się systemy sygnałem zwanym *odpowiedzią impulsową*. Ważne znaczenie splotu wynika z faktu, że wiąże on ze sobą trzy istotne sygnały – wejściowy, wyjściowy i odpowiedź impulsową. W tym rozdziale przedstawiono splot z dwóch punktów widzenia, określanych jako algorytm od strony wejścia i algorytm od strony wyjścia. Splot jest matematyczną podstawą DSP. Nie ma w tej książce niczego ważniejszego niż splot.

Funkcja delta i odpowiedź impulsowa

W poprzednim rozdziale opisano, w jaki sposób sygnał może być rozkładany na grupę składowych zwanych **impulsami**. Impuls jest sygnałem złożonym z samych zer, oprócz pojedynczego punktu niezerowego. W rezultacie rozkład impulsowy daje możliwość analizy sygnału próbka po próbce. W poprzednim rozdziale przedstawiono też podstawową koncepcję DSP: sygnał wejściowy jest rozkładany na proste składowe addytywne; każda z tych składowych jest przepuszczana przez system liniowy i uzyskane sygnały wyjściowe są poddawane syntezie (sumowaniu). Sygnał będący wynikiem takiej procedury „dziel i rządź" jest identyczny z sygnałem uzyskiwanym przez bezpośrednie przejście sygnału oryginalnego przez system. Chociaż jest wiele możliwych sposobów rozkładu sygnału, to dwie metody tworzą trzon przetwarzania sygnałów. Są to: rozkład impulsowy i rozkład Fouriera. Przy stosowaniu rozkładu impulsowego można zastosowaną procedurę opisać operacją matematyczną zwaną **splotem**. W tym rozdziale (i w większości następnych) będziemy się zajmować tylko sygnałami *dyskretnymi*. Splot dotyczy także sygnałów *ciągłych*, lecz wtedy operacje matematyczne są bardziej skomplikowane. Na przetwarzanie sygnałów analogowych zwrócimy uwagę w rozdziale 13.

Na rys. 6.1 zdefiniowano dwa ważne pojęcia stosowane w DSP. Pierwszym jest **funkcja delta**, oznaczana grecką literą delta **δ[n]**. Funkcja delta jest *znormalizowanym* impulsem, którego próbka numer zero ma wartość jeden, a wszystkie pozostałe próbki wartość zero. Z tego powodu funkcja delta jest często nazywana **impulsem jednostkowym**.

Drugie pojęcie zdefiniowane na rys. 6.1 to **odpowiedź impulsowa**. Jak to już sugeruje nazwa, odpowiedź impulsowa jest sygnałem pojawiającym się na wyjściu systemu, gdy na jego wejściu jest funkcja delta (impuls jednostkowy). Jeśli dwa systemy różnią się od siebie pod jakimkolwiek względem, to ich odpowiedzi impulsowe będą różne. Podobnie jak sygnał wejściowy i wyjściowy oznacza się często przez $x[n]$ i $y[n]$, to odpowiedzi impulsowej nadaje się symbol $h[n]$ Oczywiście sposób oznaczania można zmienić, jeśli są możliwe symbole bardziej opisowe, np. $f[n]$ na oznaczenie odpowiedzi impulsowej *filtru*.



Rys. 6.1. Definicje funkcji delta i odpowiedzi impulsowej. Funkcja delta jest znormalizowanym impulsem. Wszystkie jej próbki mają wartość zero, oprócz próbki numer zero, która ma wartość jeden. Funkcja ta jest oznaczana jako δ[n]. Odpowiedź impulsowa systemu liniowego, zwykle oznaczana jako h[n], jest sygnałem wyjściowym systemu, gdy na jego wejściu jest funkcja delta

Każdy impuls może być wyrażony jako *przesunięta* lub *przeskalowana* funkcja delta. Rozważmy sygnał $a[n]$ złożony z samych zer oprócz próbki numer 8, która ma wartość –3. Jest to funkcja delta przesunięta w prawo o 8 próbek i pomnożona przez –3. Można to wyrazić wzorem: $a[n] = -3\delta[n-8]$. Postarajcie się zapamiętać ten sposób zapisu, gdyż jest on stosowany niemal we wszystkich wzorach dotyczących DSP.

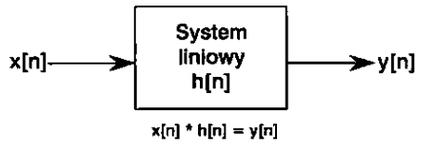
Co uzyskuje się na wyjściu systemu, jeśli na jego wejściu jest impuls, taki jak np. $-3\delta[n-8]$? Tu właśnie trzeba zastosować właściwości homogeniczności i niezmienności względem przesunięcia. Przeskalowanie i przesunięcie na wejściu powoduje takie samo przesunięcie i przeskalowanie na wyjściu. Jeśli $\delta[n]$ daje $h[n]$, to wynika stąd, że $-3\delta[n-8]$ daje $-3h[n-8]$. Innymi słowy, na wyjściu uzyskuje się wersję odpowiedzi impulsowej *przesuniętą* i *przeskalowaną* o takie same wartości, jak funkcja delta na wejściu. Jeśli znamy odpowiedź impulsową systemu, to od razu wiemy, jak będzie on reagował na *każdy* impuls.

Splot

Podsumujmy sposób rozumienia sposobu, w jaki system zmienia sygnał wejściowy w wyjściowy. Po pierwsze, sygnał wejściowy może być rozłożony na zbiór impulsów, z których każdy można rozważać jako przesuniętą i przeskalowaną funkcję delta. Po drugie, sygnał wyjściowy wywołany przez każdy z impulsów jest przesuniętą i przeskalowaną wersją odpowiedzi impulsowej. Po trzecie, całkowity sygnał wyjściowy można wyznaczyć, sumując te przesunięte i przeskalowane odpowiedzi impulsowe. Inaczej mówiąc, jeśli znamy odpowiedź impulsową systemu, to możemy obliczyć sygnał wyjściowy dla każdego możliwego sygnału wejściowego. Tak więc wiemy *wszystko* o systemie. Niczego już więcej nie możemy się dowiedzieć o właściwościach systemu liniowego. Jednak w następnych rozdziałach wykażemy, że ta informacja może być wyrażana w różnych formach.

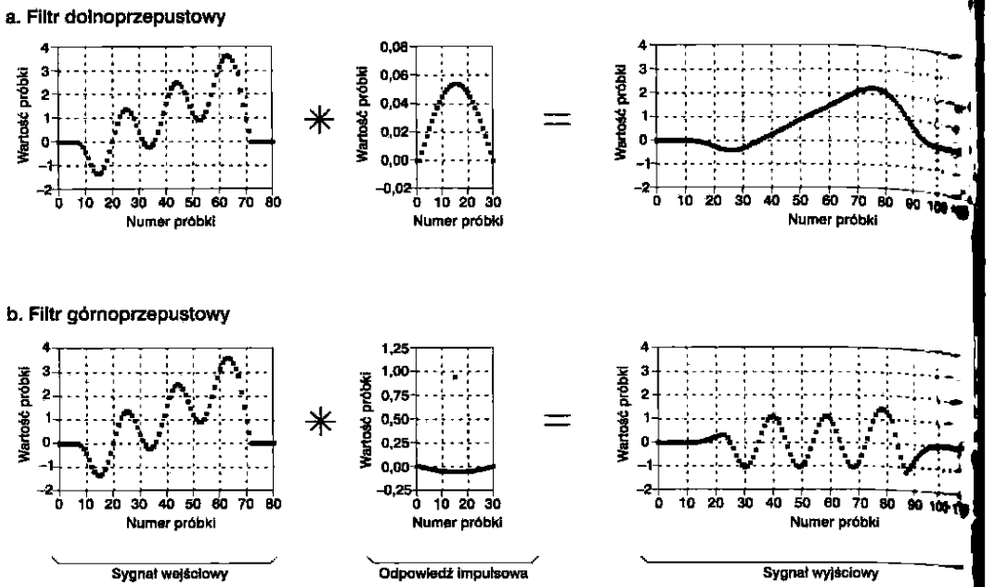
W niektórych zastosowaniach, w terminologii angielskojęzycznej, odpowiedź impulsowa pojawia się pod innymi nazwami. Jeśli rozpatrywanym systemem jest *filtr*, to odpowiedź impulsowa jest nazywana *filter kernel*[10] lub *convolution kernel*, a w dziedzinie przetwarzania obrazów – *Point Spread Function – PSF* (funkcja rozmycia punktu). Chociaż te pojęcia są używane na nieco różne sposoby, to wszystkie określają to samo – sygnał wytwarzany przez system, gdy na jego wejściu jest funkcja delta.

Splot jest operacją matematyczną, podobnie jak mnożenie, dodawanie i całkowanie. Wykonując dodawanie, bierze się dwie liczby i uzyskuje trzecią *liczbę*, w przypadku splotu zaś bierze się dwa *sygnały* i uzyskuje trzeci *sygnał*. Operacja splotu jest stosowana w wielu działach matematyki, jak rachunek prawdopodobieństwa i statystyka. W systemach liniowych stosuje się splot do opisu wzajemnej zależności między trzema sygnałami – sygnałem wejściowym, odpowiedzią impulsową i sygnałem wyjściowym.

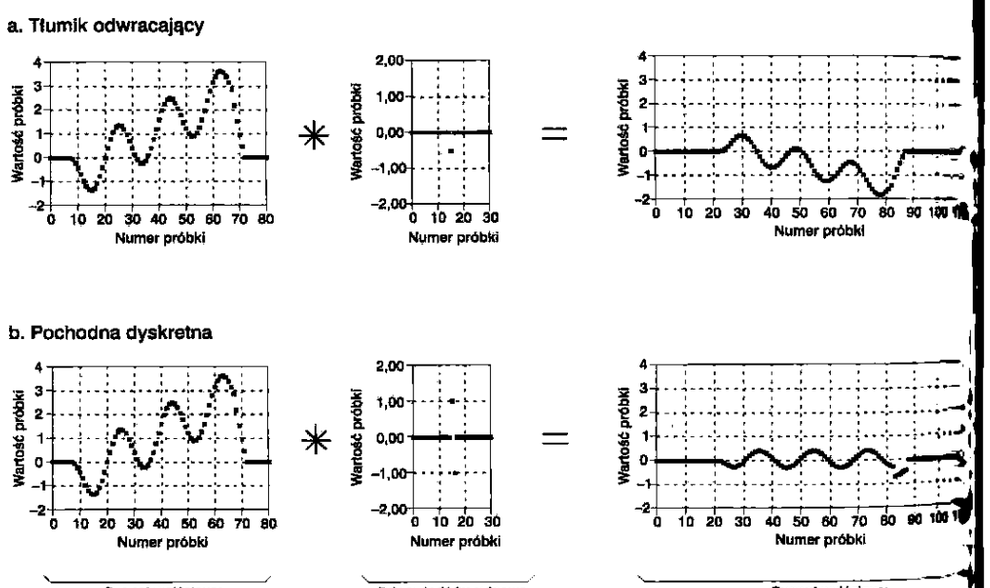


Rys. 6.2. Zastosowanie splotu w DSP. Sygnał na wyjściu systemu liniowego jest równy sygnałowi wejściowemu poddanemu operacji splotu z odpowiedzią impulsową systemu. We wzorach operację splotu oznacza się gwiazdką

[10] W terminologii polskiej spotyka się czasem pojęcie „jądro filtru" określające odpowiedź impulsową filtru (przyp. tłum.).



Rys. 6.3. Przykłady splotu zastosowanego w filtracji dolno- i górnoprzepustowej. Sygnał wejściowy w tym przykładzie jest sumą kilku okresów fali sinusoidalnej i wolno narastającego przebiegu piłowego. Rozdzielenie tych dwóch składowych następuje w wyniku właściwie dobranych odpowiedzi impulsowych



Rys. 6.4. Przykłady sygnałów przetwarzanych przy użyciu splotu. W wielu operacjach przetwarzania sygnałów stosuje się bardzo proste odpowiedzi impulsowe. Jak wynika z podanych przykładów, bardzo duże zmiany można uzyskać przy użyciu bardzo niewielu punktów niezerowych w odpowiedzi impulsowej

Na rys. 6.2 przedstawiono zapis używany przy stosowaniu splotu w systemach liniowych. Impuls wejściowy $x[n]$ jest wprowadzany do systemu liniowego o odpowiedzi impulsowej $h[n]$. Na wyjściu uzyskuje się sygnał wyjściowy $y[n]$. Można to wyrazić wzorem $x[n] * h[n] = y[n]$. Wyraźmy to słownie: impuls wejściowy spleciony z odpowiedzią impulsową jest równy sygnałowi wyjściowemu. Podobnie jak dodawanie jest oznaczane przez +, mnożenie przez ×, splot jest oznaczany gwiazdką *. Bardzo niefortunne jest stosowanie takiej gwiazdki, w większości języków programowania, do oznaczania mnożenia. Gwiazdka w programie komputerowym oznacza mnożenie, we wzorach zaś – operację splotu.

Na rys. 6.3 przedstawiono zastosowanie splotu w filtracji dolno- i górnoprzepustowej. Przykładowym sygnałem wejściowym jest suma dwóch składowych – trzech okresów fali sinusoidalnej (reprezentujących duże częstotliwości) i wolno narastającego przebiegu piłowego (zawierającego małe częstotliwości). Odpowiedź impulsowa filtru dolnoprzepustowego (rys. 6.3 a) jest gładką krzywą o kształcie łuku i daje w wyniku tylko wolno narastający przebieg piłowy, przepuszczany do wyjścia filtru. Filtr górnoprzepustowy (rys. 6.3 b) zaś przepuszcza tylko szybkozmienne sinusoidy.

Dwa dalsze przykłady zastosowania splotu w przetwarzaniu sygnałów przedstawiono na rys. 6.4. Tłumik odwracający (rys. 6.4 a) odwraca sygnał i zmniejsza jego amplitudę. Operacja pochodnej dyskretnej (rys. 6.4 b) daje sygnał wyjściowy zależny od nachylenia sygnału wejściowego.

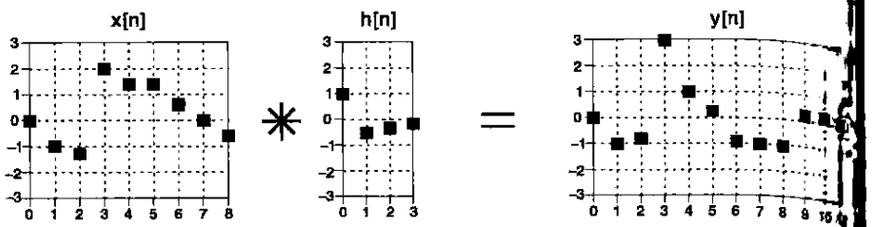
Zwróćmy uwagę na długość sygnałów na rys. 6.3 i 6.4. Sygnały wejściowe mają długość 81 próbek, a każda z odpowiedzi impulsowych – 31 próbek. W większości zastosowań DSP impulsy wejściowe mają długość setek, tysięcy lub nawet milionów próbek. Odpowiedź impulsowa jest zwykle znacznie krótsza – od kilku do kilkuset punktów. Matematyczne podstawy operacji splotu nie ograniczają długości tych sygnałów, wyznaczają jednak długość sygnału wyjściowego. Długość sygnału wyjściowego jest równa sumie długości sygnału wejściowego i odpowiedzi impulsowej, pomniejszonej o 1. Na rys. 6.3 i 6.4 sygnały wyjściowe mają długość 81 + 31 – 1 = 111 próbek. Sygnały wejściowe zawierają próbki od 0 do 80, odpowiedzi impulsowe – od 0 do 30, a sygnały wyjściowe – od 0 do 110.

Dochodzimy teraz do szczegółów matematycznych operacji splotu. Splot, jako operację stosowaną w cyfrowym przetwarzaniu sygnałów, można rozważać dwoma sposobami. Po pierwsze, możemy rozpatrywać splot **z punktu widzenia sygnału wejściowego**. To prowadzi do przeanalizowania, co *wnosi* każda próbka sygnału wejściowego do wielu punktów sygnału wyjściowego. Drugi sposób to rozpatrywanie splotu **z punktu widzenia sygnału wyjściowego**. Wtedy bada się, co *otrzymuje* każda próbka sygnału wyjściowego z wielu punktów sygnału wejściowego.

Pamiętajmy, że te dwa spojrzenia na splot z różnych perspektyw są różnymi sposobami myślenia o tej samej operacji matematycznej. Pierwszy punkt widzenia jest istotny, gdyż daje *pojęciowe* zrozumienie związku splotu z DSP. Drugi punkt widzenia daje zaś *matematyczny* opis splotu. Jest to przykładem jednego z najtrudniejszych zadań, jakie napotykamy w DSP – powiązania zrozumienia pojęciowego z gmatwaniną zależności matematycznych opisujących zachodzące procesy.

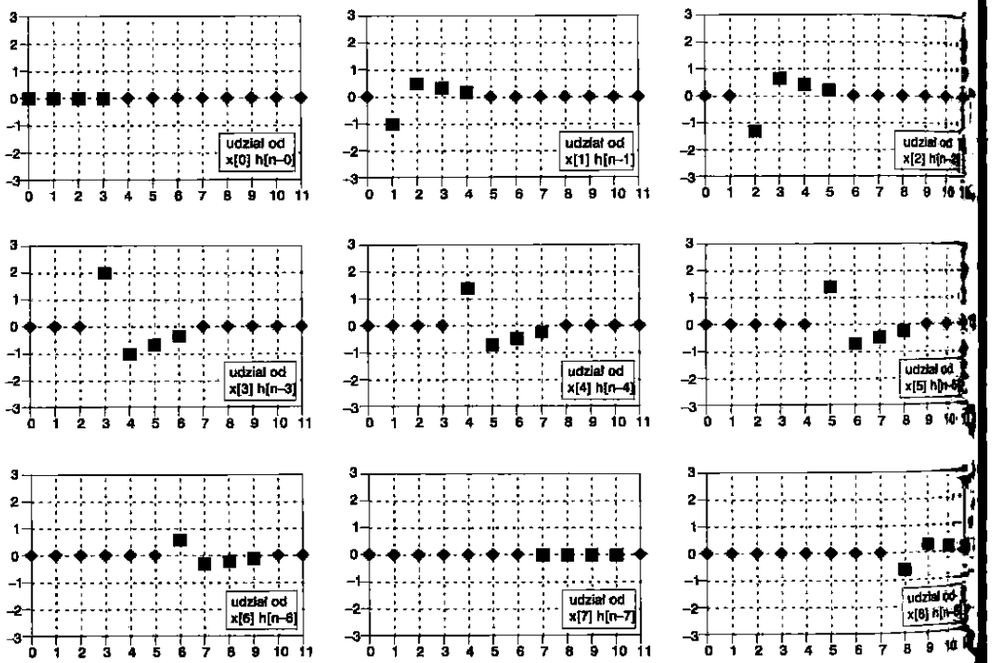
Algorytm od strony wejścia

Na rys. 6.5 przedstawiono prosty problem związany ze splotem. Sygnał 9-punktowy $x[n]$, przechodzący przez system o 4-punktowej odpowiedzi impulsowej $h[n]$ daje sygnał wyjściowy $y[n]$ 12-puntowy (9 + 4 – 1 = 12 punktów). W kategoriach matematycznych można powiedzieć, że $x[n]$ jest splatana z $h[n]$, dając $y[n]$. Taki sposób spojrzenia na splot wynika z podstawowej koncepcji DSP: rozłożenie sygnału, przepuszczenie jego składowych przez system i uzyskanie sygnału wyjściowego przez syntezę. W podanym przykładzie każda z 9 próbek sygnału wejściowego będzie wnosiła do sygnału wyjściowego przeskalowaną i przesuniętą wersję odpowiedzi impulsowej. Te dziewięć sygnałów pokazano na rys. 6.6. Dodając te 9 sygnałów, uzyskuje się sygnał wyjściowy $y[n]$.



Rys. 6.5. Przykład problemu związanego ze splotem. 9-punktowy sygnał wejściowy splecony z 4-punktową odpowiedzią impulsową daje 12-punktowy sygnał wyjściowy. Każdy punkt sygnału wejściowego daje wkład do sygnału wyjściowego w postaci przeskalowanej i przesuniętej odpowiedzi impulsowej. Te 9 przeskalowanych i przesuniętych odpowiedzi przedstawiono na rys. 6.6.

Przyjrzyjmy się bardziej szczegółowo kilku z tych dziewięciu sygnałów. Zacznijmy od próbki numer 4, a więc $x[4]$ w sygnale wejściowym. Ta próbka ma wartość 1,4. Gdy sygnał zostaje rozłożony, próbka przekształca się w impuls wyrażony jako $1{,}4\delta[n-4]$. Po przejściu przez system uzyskuje się składową wyjściową $1{,}4h[n-4]$. Ten sygnał przedstawiono w środkowej ramce wśród 9 sygnałów na rys. 6.6. Zauważmy, że jest to odpowiedź impulsowa $h[n]$ pomnożona przez 1,4 i przesunięta o 4 próbki w prawo. Zera wprowadzono do próbek od 0 do 3 oraz od 8 do 11 jako znaki-wypełniacze. Na rys. 6.6, dla jasności oznaczono *kwadratami* punkty danych pochodzące od przeskalowanej i przesuniętej odpowiedzi impulsowej, rombami zaś – dodane zera.

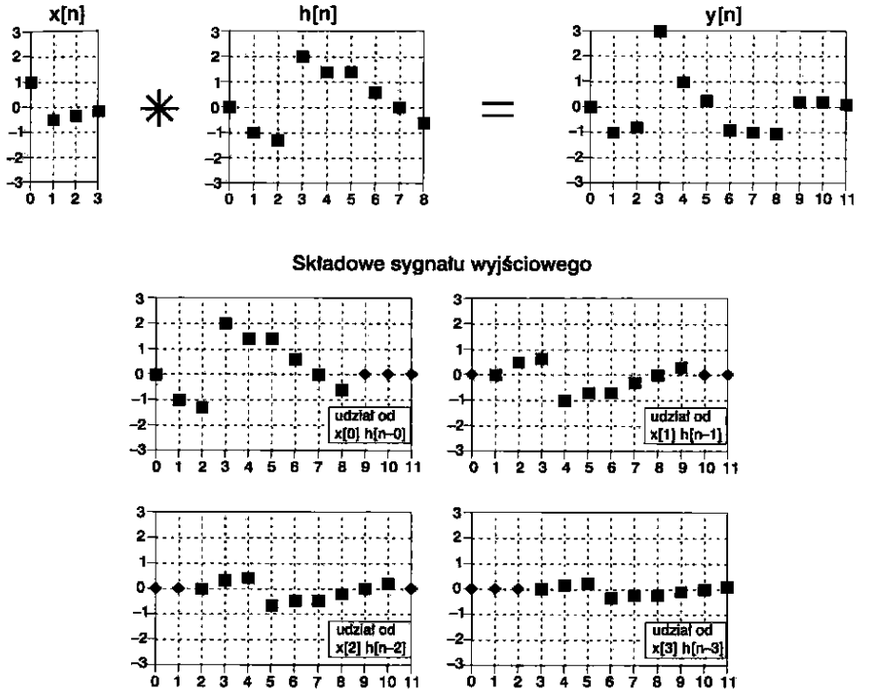


Rys. 6.6. Składowe sygnału wyjściowego dla splotu przedstawionego na rys. 6.5. Każdy punkt w tych sygnałach, wynikający z przeskalowanej i przesuniętej odpowiedzi impulsowej, jest oznaczony kwadratem. Pozostałe punkty danych, oznaczone rombami, są zerami dodanymi jako znaki-wypełniacze

Zbadajmy teraz próbkę $x[8]$ – ostatni punkt sygnału wejściowego. Jest to próbka o numerze 8 i wartości –0,5. Jak pokazano na wykresie w prawym, dolnym rogu na rys. 6.6, $x[8]$ daje odpowiedź impulsową przesuniętą w prawo o osiem punktów i pomnożoną przez –0,5. Zera jako znaki-wypełniacze wprowadzono do punktów od 0 do 7. Na koniec sprawdźmy wpływ punktów $x[0]$ i $x[7]$. Obie te próbki mają wartość zero, więc wytwarzają składowe wyjściowe złożone z samych zer.

W omówionym przykładzie $x[n]$ jest sygnałem 9-punktowym a $h[n]$ 4-punktowym. W następnym przykładzie, przedstawionym na rys. 6.7, odwrócimy tę sytuację, wybierając 4-punktowy sygnał $x[n]$ i 9-punktową odpowiedź impulsową $h[n]$. Zastosowano takie same dwa przebiegi, zamieniając je tylko miejscami. Jak wynika z przedstawionych składowych sygnału wyjściowego, cztery próbki $x[n]$ dają cztery przesunięte i przeskalowane wersje 9-punktowej odpowiedzi impulsowej. Tak jak poprzednio, z przodu i z tyłu dodano zera jako znaki-wypełniacze.

Ale przyjrzyjmy się tym rysunkom. Przecież sygnał wyjściowy na rys. 6.7 jest *identyczny* jak sygnał wyjściowy na rys. 6.5. To nie pomyłka, lecz ważna właściwość. Operacja splotu jest *przemienna*: $a[n] * b[n] = b[n] * a[n]$. Z matematycznego punktu widzenia nie ma znaczenia, który sygnał jest wejściowym, a który jest odpowiedzią impulsową. Ważne jest tylko, że dwa sygnały są poddawane operacji splotu. Zamiana sygnałów miejscami nie ma jednak żadnego fizycznego znaczenia, chociaż matematyka na nią zezwala. Sygnał wejściowy i odpowiedź impulsowa to dwie całkiem różne rzeczy i ich wzajemna zamiana nie ma sensu. Przemienność splotu jest *narzędziem matematycznym* służącym do przekształcania wzorów w celu uzyskania potrzebnych wyników.



Rys. 6.7. Inny przykład splotu. Przebiegi sygnału wejściowego i odpowiedzi impulsowej zamieniono miejscami w stosunku do rys. 6.5. Ponieważ splot jest operacją przemienną, więc sygnały wyjściowe w obu przykładach są jednakowe

Tab. 6.1

```
100 'SPLOT PRZY UŻYCIU ALGORYTMU OD STRONY WEJŚCIA
110 '
120 DIM X[80]                    'Sygnał wejściowy, 81 punktów
130 DIM H[30]                    'Odpowiedź impulsowa, 31 punktów
140 DIM Y[110]                   'Sygnał wyjściowy, 111 punktów
150 '
160 GOSUB XXXX                   'Podprogram "mityczny" do ładowania X[ ] i H[ ]
170 '
180 FOR I% = 0 TO 110            'Zeruj matrycę wyjściową
190   Y(I%) = 0
200 NEXT I%
210 '
220 FOR I% = 0 TO 80             'Pętla przez każdy punkt w X[ ]
230   FOR J% = 0 TO 30           'Pętla przez każdy punktu w H[ ]
240     Y[I%+J%] = Y[I%+J%] + X[I%] * H[J%]
250 NEXT J%
260 NEXT I%                      '(pamiętajcie, że * w programach oznacza mnożenie!)
270 '
280 GOSUB XXXX                   'Podprogram "mityczny" do zapisywania Y[ ]
290 '
300 END
```

W tablicy 6.1 podano program do obliczania splotu z wykorzystaniem algorytmu od strony wejścia. Proszę pamiętać, że programy zamieszczone w tej książce mają na celu przedstawienie *algorytmów* w najprostszej postaci, nawet kosztem dobrego stylu programowania. Na przykład wszystkie wejścia i wyjścia są obsługiwane przez pewne podprogramy, które nazywamy „**mitycznymi**" (wiersze 160 i 280), co oznacza że nie określamy, jak te operacje są przeprowadzane. Nie pomijajcie tych programów. Są one istotną częścią prezentowanego materiału i trzeba je zrozumieć szczegółowo.

Program realizuje splot 81-punktowego sygnału wejściowego przechowywanego w matrycy X[ ] z 31-punktową odpowiedzią impulsową przechowywaną w matrycy H[ ], dając w rezultacie 111-punktowy sygnał wyjściowy, przechowywany w matrycy Y[ ]. Są to takie same długości sygnałów, jak na rys. 6.3 i 6.4. Proszę zwrócić uwagę, że oznaczenia matryc są pisane dużymi literami. Jest to złamanie poprzednio podanej konwencji, zgodnie z którą duże litery są zarezerwowane dla sygnałów w dziedzinie częstotliwości. Niestety, prosty BASIC używany w tej książce nie dopuszcza oznaczania zmiennych małymi literami. Zauważmy też, że w wierszu 240 zastosowano gwiazdkę jako oznaczenie *mnożenia*. Pamiętajmy, że gwiazdka w programie oznacza mnożenie, a we wzorach – splot. Gwiazdka użyta w tekście (np. w dokumentacji lub w komentarzach do programu) może oznaczać jedno lub drugie.

„Mityczny" podprogram w wierszu 160 umieszcza sygnał wejściowy w X[ ] a odpowiedź impulsową w H[ ]. W wierszach 180–200 następuje ustawienie na zero wszystkich wartości w Y[ ]. Jest to konieczne, gdyż Y[ ] jest wykorzystywane jako akumulator do sumowania wyników w miarę ich obliczania. Wiersze od 220 do 260 to serce programu. Instrukcja FOR w wierszu 220 steruje pętlą przechodzącą przez wszystkie punkty sygnału wejściowego X[ ]. Dla każdej próbki sygnału wejściowego w pętli wewnętrznej (wiersze 230–250) następuje obliczenie przeskalowanej i przesuniętej wersji odpowiedzi impulsowej oraz dodawanie jej do matrycy Y[ ] akumulującej sygnał wyjściowy. Zagnieżdżona struktura pętli (jedna pętla w drugiej) jest charakterystyczną cechą programów splotowych; warto się z nią zaznajomić.

Zawiłości pojawiają się po dojściu z indeksowaniem do wiersza 240. Powiedzmy, że jesteśmy w połowie wykonywania programu, a więc zaczęliśmy właśnie działania na próbce X[40], tj. I% = 40. Pętla wewnętrzna przechodzi przez każdy punkt odpowiedzi impulsowej, wykonując trzy operacje. Po pierwsze, odpowiedź impulsowa zostaje *przeskalowana* w wyniku przemnożenia jej przez wartość próbki wejściowej. Gdyby to była jedyna czynność realizowana przez pętlę wewnętrzną, to wiersz 240 można by zapisać: Y[J%] = X[40] * H[J%]. Po drugie, przeskalowany impuls jest *przesuwany* o 40 próbek w prawo przez dodanie tej liczby do indeksu zastosowanego w sygnale wyjściowym. Ta druga operacja powoduje zmianę wiersza 240 na: Y[40 + J%] = X[40] * H[J%]. Po trzecie, matryca Y[ ] powinna gromadzić (*syntetyzować*) wszystkie sygnały wynikające z każdej próbki w sygnale wejściowym. Tak więc nowa informacja musi być dodana do informacji już zgromadzonej w matrycy. Z tego wynika ostatnia instrukcja: Y[40 + J%] = Y[40 + J%] + X[40] * H[J%]. Przeanalizujmy to dokładnie; jest to *bardzo* zagmatwane, ale także *bardzo* ważne.

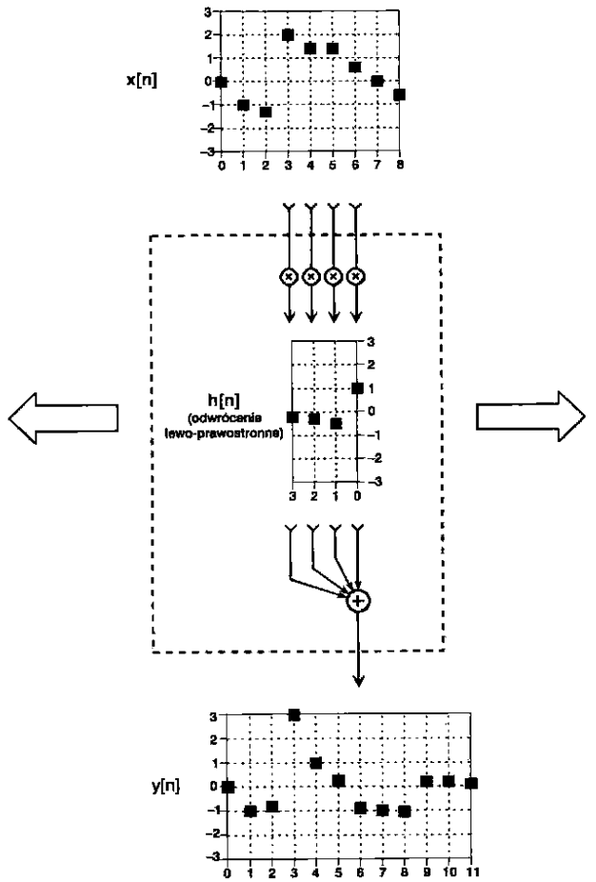
Algorytm od strony wyjścia

Pierwszy punkt widzenia przy rozpatrywaniu splotu polega na przeanalizowaniu, jak każda próbka *sygnału wejściowego* wpływa na wiele próbek w sygnale wyjściowym. Przy drugim punkcie widzenia odwracamy tę sytuację, rozpatrując indywidualne próbki w *sygnale wyjściowym* i wyszukując punkty w sygnale wejściowym, które wniosły wkład do tych próbek na wyjściu. Jest to ważne zarówno pod względem matematycznym, jak i praktycznym. Załóżmy, że mamy dany jakiś sygnał wejściowy i jakąś odpowiedź impulsową i chcemy znaleźć splot tych dwóch sygnałów. Najprostszą metodą byłoby napisanie programu, który pętlą obejmowałby *sygnał wyjściowy*, obliczając jedną próbkę w każdym cyklu pętli. Wzory są zapisane w postaci: $y[n]$ – *pewna kombinacja innych zmiennych*. To znaczy, że próbka n w sygnale wyjściowym jest równa jakiejś kombinacji wielu wartości w sygnale wejściowym i w odpowiedzi impulsowej. Potrzebna jest więc wiedza o tym, jak każda próbka w sygnale wyjściowym może być obliczona niezależnie od wszystkich innych próbek w sygnale wyjściowym. Takiej informacji dostarcza algorytm od strony wyjścia.

Na przykładzie rozważmy, jaki wpływ na jeden punkt w sygnale wyjściowym ma wiele punktów w sygnale wejściowym. Jako przykład weźmy punkt $y[6]$ na rys. 6.5. Ten punkt jest równy sumie wszystkich punktów numer 6 w dziewięciu składowych wyjściowych przedstawionych na rys. 6.6. Przyjrzyjmy się teraz dokładniej tym dziewięciu składowym wyjściowym i ustalmy, która z nich może mieć wpływ na $y[6]$. To znaczy, że trzeba sprawdzić, które z tych dziewięciu sygnałów zawierają niezerowe próbki na pozycji numer 6. Pięć spośród składowych wyjściowych ma na tej pozycji *dodane* zera (znaczniki w kształcie rombów), można je więc zignorować. Tylko cztery ze składowych wyjściowych mogą mieć niezerowe wartości na szóstej pozycji. Są to składowe wyjściowe utworzone z próbek wejściowych $x[3]$, $x[4]$, $x[5]$ i $x[6]$. Przez sumowanie próbek numer 6 z każdej z tych składowych wyjściowych uzyskuje się $y[6] = x[3]h[3] + x[4]h[2] + x[5]h[1] + x[6]h[0]$. Tak więc cztery próbki sygnału wejściowego są mnożone przez cztery próbki w odpowiedzi impulsowej i iloczyny są sumowane.

Na rys. 6.8 przedstawiono algorytm od strony wyjścia jako tzw. **maszynę splotową**, sieć działań, według której jest realizowana operacja splotu. Przyjmijmy, że sygnały wejściowy $x[n]$ i wyjściowy $y[n]$ są takie, jak właśnie omawiane. Maszyna splotowa (czyli wszystko co znajduje się w ramce zaznaczonej linią przerywaną) może się poruszać w lewo lub w prawo, zależnie od potrzeb. Maszyna splotowa jest umieszczana w taki sposób, aby jej wyjście było ustawione na wprost próbki wyjściowej, która jest właśnie obliczana. Cztery próbki z sygnału wejściowego wchodzą na wejścia maszyny splotowej. Wartości tych próbek są mnożone przez wskazane próbki w odpowiedzi impulsowej i iloczyny są sumowane. W ten sposób uzyskuje się wartość sygnału wyjściowego, który wpada na swoje właściwe miejsce. Jako przykład pokazano obliczanie $y[6]$ na podstawie czterech próbek wejściowych: $x[3]$, $x[4]$, $x[5]$ i $x[6]$.

W celu obliczenia $y[7]$ maszyna splotowa przesuwa się o jedną próbkę w prawo. Wtedy inny zestaw czterech próbek wchodzi do maszyny splotowej, a więc $x[4]$, $x[5]$, $x[6]$, $x[7]$ i wartość $y[7]$ wpada na swoje miejsce. Ta procedura jest powtarzana dopóty, dopóki nie zostaną obliczone wszystkie, przewidziane do wyznaczenia, punkty sygnału wyjściowego.



Rys. 6.8. Maszyna splotowa. Jest to sieć działań pokazująca, jaki wpływ na każdą próbkę w sygnale wyjściowym ma sygnał wejściowy i odpowiedź impulsowa. Szczegóły podano w tekście

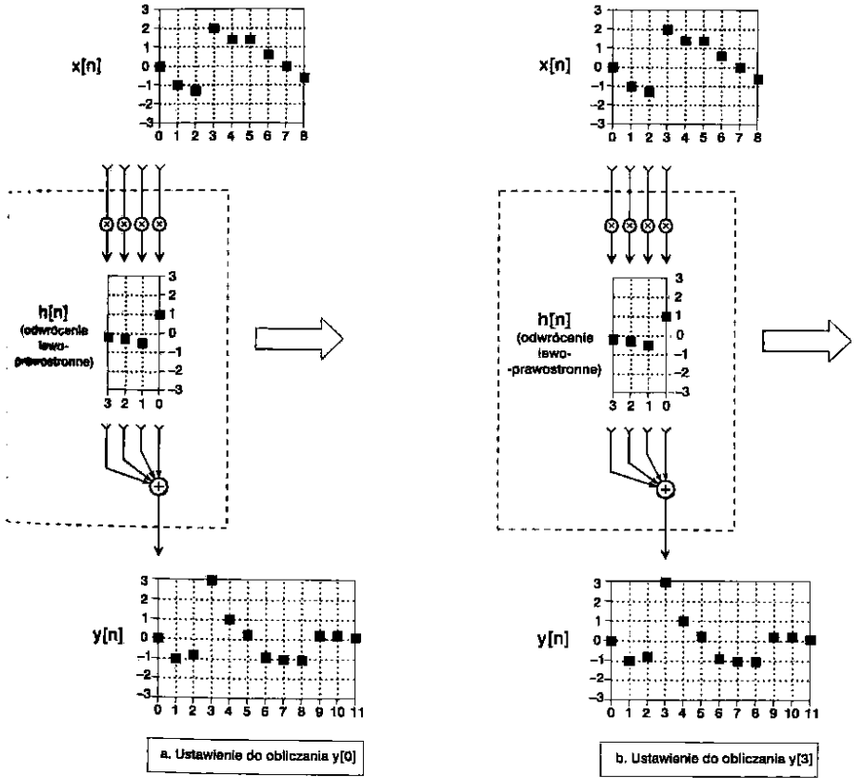
Bardzo ważną kwestią jest sposób ustawienia odpowiedzi impulsowej *wewnątrz* maszyny splotowej. Odpowiedź impulsowa jest *odwracana lewo-prawostronnie* Tak więc próbka numer zero jest umieszczona z prawej strony, a dodatnie numery próbek wzrastają w kierunku na lewo. Proszę porównać to z normalną odpowiedzią impulsową na rys. 6.5, aby zrozumieć geometrię tego odwracania. Dlaczego jest potrzebne takie odwracanie lewo-prawostronne? Wynika to wprost z matematyki. Odpowiedź impulsowa opisuje, jak każdy punkt w sygnale wejściowym wpływa na sygnał wyjściowy. Z tego wynika, że na każdy punkt w sygnale wyjściowym wpływają punkty w sygnale wejściowym ważone przez *przełączaną* odpowiedź impulsową.

Na rys. 6.9 przedstawiono maszynę splotową zastosowaną do obliczania kilku próbek w sygnale wyjściowym. Rysunek ilustruje też istotną niedogodność splotu. Na rys. 6.9 a maszyna splotowa jest w skrajnym lewym położeniu z wyjściem ustawionym na $y[0]$. W tym położeniu maszyna próbuje uzyskać wartości próbek wejściowych $x[-3]$, $x[-2]$, $x[-1]$ i $x[0]$. Problem jest w tym, że trzy z wymienionych próbek: $x[-3]$, $x[-2]$ i $x[-1]$ nie istnieją! Ten sam dylemat pojawia się na rys. 6.9 d, gdzie maszyna splotowa usiłuje pobrać próbki na prawo od zdefiniowanego sygnału wejściowego, punkty $x[9]$, $x[10]$ i $x[11]$.

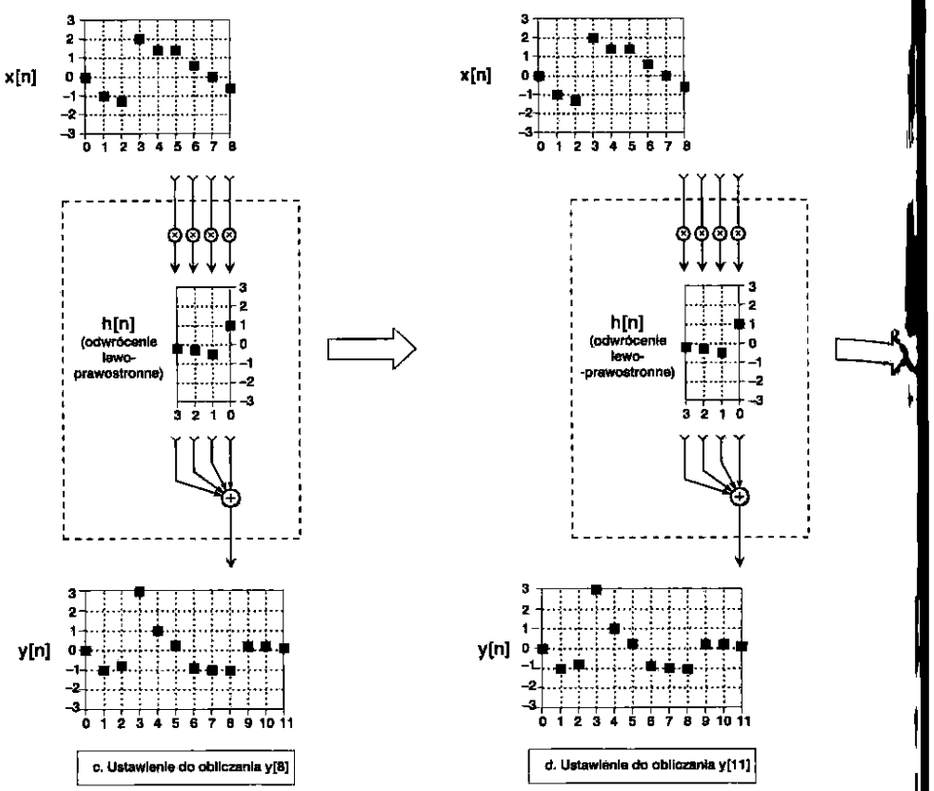
Jednym ze sposobów poradzenia sobie z tym problemem jest *wprowadzenie* nieistniejących próbek. Polega to na dodaniu próbek o wartości *zero* na końcu sygnału wejściowego. Jest to tzw. **dopełnianie** (*padding*) sygnału zerami. Zamiast usiłować uzyskać dostęp do jakiejś nieistniejącej wartości, maszyna splotowa otrzymuje próbkę o wartości zero. Ponieważ to zero jest eliminowane podczas mnożenia, więc wynik jest matematycznie taki sam, jak w przypadku *zignorowania* nieistniejących wejść.

Ważnym faktem jest, że próbki położone skrajnie na lewo i skrajnie na prawo w sygnale wyjściowym są obliczane na podstawie *niekompletnej* informacji. W potocznie używanej terminologii DSP mówi się, że **odpowiedź impulsowa jest nie w pełni zanurzona w sygnale wejściowym**. Jeśli odpowiedź impulsowa ma długość M punktów, to pierwsze *oraz* ostatnie $M-1$ próbek sygnału wyjściowego jest wyznaczone na podstawie mniejszej ilości informacji niż pozostałe próbki. Dzieje się to podobnie, jak w układzie elektronicznym wymagającym pewnego czasu do ustabilizowania swego działania po włączeniu zasilania. Różnica polega na tym, że w układzie elektronicznym można taki stan przejściowy łatwo pominąć, w DSP jest to zjawisko znaczące.

Na rys. 6.10 przedstawiono przykład trudności, jakie mogą spowodować te efekty krańcowe. Sygnałem wejściowym jest fala sinusoidalna z dodaną składową stałą. Zadaniem jest usunięcie składowej stałej z pozostawieniem nieznieksztalconej sinusoidy. To wymaga filtru górnoprzepustowego, takiego jak pokazana na rysunku odpowiedź impulsowa. Problem jest taki, że w pierwszych i w ostatnich 30 próbkach powstaje bałagan. Kształt przebiegu w tych



Rys. 6.9. Działanie maszyny splotowej. Na rysunkach a) d) przedstawiono maszynę splotową ustawioną do obliczania czterech różnych próbek w sygnale wyjściowym: y[0], y[3], y[8] oraz y[11]



Rys. 6.9. (cd.)

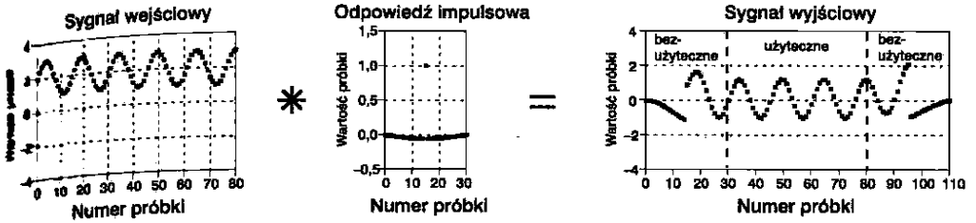
obszarach krańcowych można zrozumieć, wyobrażając sobie sygnał wejściowy dopełniony 30 zerami z lewej strony (próbki od $x[-1]$ do $x[-30]$) i 30 zerami z prawej (próbki od $x[81]$ do $x[110]$). Na sygnał wyjściowy trzeba więc spojrzeć jako na odfiltrowaną wersję tego dłuższego przebiegu. Takie kłopoty z efektami krańcowymi są często spotykane w cyfrowym przetwarzaniu sygnałów. Jako ogólną zasadę należy przyjąć, że próbki początkowe i końcowe w przetwarzanych sygnałach są całkiem bezużyteczne.

Teraz trochę matematyki. Stosując maszynę splotową jako wytyczną, możemy zapisać standardowy wzór wyrażający splot. Jeśli $x[n]$ jest sygnałem N-punktowym zawierającym próbki o numerach od 0 do $N-1$, a $h[n]$ jest sygnałem M-punktowym (od 0 do $M-1$), to splot tych dwóch sygnałów $y[n] = x[n] * h[n]$ jest sygnałem $(N+M-1)$-punktowym (od 0 do $N+M-2$), wyrażonym wzorem:

$$y[i] = \sum_{j=0}^{M-1} h[j]\,x[i-j]$$

Wzór 6.1. Obliczanie splotu. Jest to formalna definicja operacji splotu, zapisywanej w uproszczeniu jako y[n] = x[n] * h[n]. W tym wzorze h[n] jest sygnałem M-punktowym o indeksach od 0 do M – 1

Powyższe wyrażenie jest nazywane **sumą splotową**. Wzór umożliwia obliczenie każdego punktu w sygnale wyjściowym niezależnie od wszystkich innych punktów w tym sygnale. Indeks i określa, która próbka sygnału wyjściowego jest obliczana, a zatem wyznacza też



Rys. 6.10. Efekty krańcowe przy wyznaczaniu splotu. Gdy sygnał wejściowy jest poddawany splotowi z M-punktową odpowiedzią impulsową, to pierwsze i ostatnie M – 1 próbek może nie nadawać się do wykorzystania. W tym przykładzie odpowiedź impulsowa jest filtrem górnoprzepustowym użytym w celu usunięcia składowej stałej z sygnału wejściowego

położenie maszyny kodowej. W programach komputerowych realizujących splot, pętla programowa powoduje, że indeks się zmienia, obiegając wszystkie próbki sygnału wyjściowego. W celu obliczenia jednej z próbek wyjściowych stosuje się indeks j *wewnątrz* maszyny kodowej. Gdy j zmienia się od 0 do $M-1$, każda próbka w odpowiedzi impulsowej $h[j]$ jest mnożona przez odpowiednią próbkę z sygnału wejściowego $x[i-j]$. Wszystkie te iloczyny są sumowane, dając wartość obliczanej próbki wyjściowej. Proszę analizować wzór (6.1) aż do pełnego zrozumienia, jak jest on implementowany w maszynie kodowej. Z tego wzoru korzysta się w bardzo wielu zagadnieniach DSP.

Oznaczenie n w wyrażeniu $y[n] = x[n] * h[n]$ może być mylące; jest to po prostu znak-wypełniacz wskazujący, że *jakaś* zmienna jest indeksem w matrycy. Czasem ten wzór zapisuje się w postaci: $y[\,] = x[\,] * h[\,]$, aby uniknąć wprowadzania nic nieznaczącego symbolu.

W tablicy 6.2 zamieszczono program do realizacji splotu przy użyciu *algorytmu od strony wyjścia*, z bezpośrednim wykorzystaniem wzoru 6.1. Ten program daje taki sam sygnał wyjściowy, jak *algorytm od strony wejścia*, podany poprzednio w tablicy 6.1. Zwróćmy uwagę na główną różnicę między tymi programami: w algorytmie od strony wejścia pętla obejmuje każdą próbkę w *sygnale wejściowym* (wiersz 220 w tablicy 6.1), a w algorytmie od strony wyjścia – *w sygnale wyjściowym* (wiersz 180 w tablicy 6.2).

A oto szczegóły działania tego programu. Pętla FOR-NEXT w wierszach od 180 do 250 przechodzi przez każdą próbkę sygnału wyjściowego, z użyciem I% jako indeksu. Dla każdej

Tab. 6.2.

```
100 'SPLOT PRZY UŻYCIU ALGORYTMU OD STRONY WYJŚCIA
110 '
120 DIM X[80]                          'Sygnał wejściowy, 81 punktów
130 DIM H[30]                          'Odpowiedź impulsowa, 31 punktów
140 DIM Y[110]                         'Sygnał wyjściowy, 111 punktów
150 '
160 GOSUB XXXX                         'Podprogram "mityczny" ładujący X[ ] i H[ ]
170 '
180 FOR I% = 0 TO 110                  'Pętla przez każdy punkt w Y[ ]
190   Y[I%] = 0                        'Zeruj próbkę w matrycy wyjściowej
200   FOR J% = 0 TO 30                 'Pętla przez każdy punkt w H[ ]
210     IF(I%-J% < 0)  THEN GOTO 240
220     IF(I%-J% > 80) THEN GOTO 240
230     Y(I%) = Y(I%) + H(J%) * X(I%-J%)
240   NEXT J%
250 NEXT I%
260 '
270 GOSUB XXXX                         'Podprogram "mityczny" do zapisywania Y[ ]
280 '
290 END
```

z tych wartości, w pętli wewnętrznej (wiersze 200 do 230), jest obliczana wartość próbki wyjściowej Y[I%]. Wartość Y[I%] jest ustawiana na zero w wierszu 190, umożliwiając akumulowanie iloczynów wewnątrz maszyny splotowej. Pętla FOR-NEXT w wierszach od 200 do 240 jest dokładną implementacją wzoru 6.1. Indeks J% przechodzi kolejno przez każdą próbkę w odpowiedzi impulsowej. W wierszu 230 jest wykonywane mnożenie każdej próbki w odpowiedzi impulsowej H[J%] przez odpowiednią próbkę w sygnale wejściowym X[I% – J%] oraz dodawanie wyniku do akumulatora.

W wierszu 230 próbka pobrana z sygnału wejściowego jest X[I% – J%]. Wiersze 210 i 220 programu zapobiegają wyjściu poza zdefiniowaną matrycę X[0] do X[80]. Inaczej mówiąc, program *ignoruje* nieokreślone próbki w sygnale wejściowym. Inną możliwością byłoby zdefiniowanie matrycy sygnału wejściowego od X[–30] do X[110], dopuszczając, aby 30 zer dopełniało z każdej strony prawdziwe dane. Jako trzecią możliwość można by pętlę w wierszu 180 zmienić w taki sposób, aby obiegała od 30 do 80, zamiast od 0 do 110. Tak więc program obliczałby tylko te próbki w sygnale wyjściowym, dla których odpowiedź impulsowa jest *w pełni zanurzona* w sygnale wejściowym. Trzeba pamiętać, że zastosowanie jednego z trzech podanych rozwiązań jest niezbędne. Jeśli tego nie zrobimy, to program przestanie działać, usiłując odczytywać dane spoza zakresu.

Suma wejść ważonych

Właściwości systemu liniowego są w pełni opisane przez jego odpowiedź impulsową. To jest podstawą algorytmu od strony wejścia: każdy punkt w sygnale wejściowym wnosi do sygnału wyjściowego przeskalowaną i przesuniętą wersję odpowiedzi impulsowej. Matematyczne konsekwencje tego faktu prowadzą do algorytmu od strony wyjścia: każdy punkt w sygnale wyjściowym otrzymuje wkład z wielu punktów w sygnale wejściowym, mnożony przez *przesuniętą* odpowiedź impulsową. Chociaż to wszystko jest prawdą, to nie w pełni wyjaśnia, dlaczego splot jest tak ważny w przetwarzaniu sygnałów.

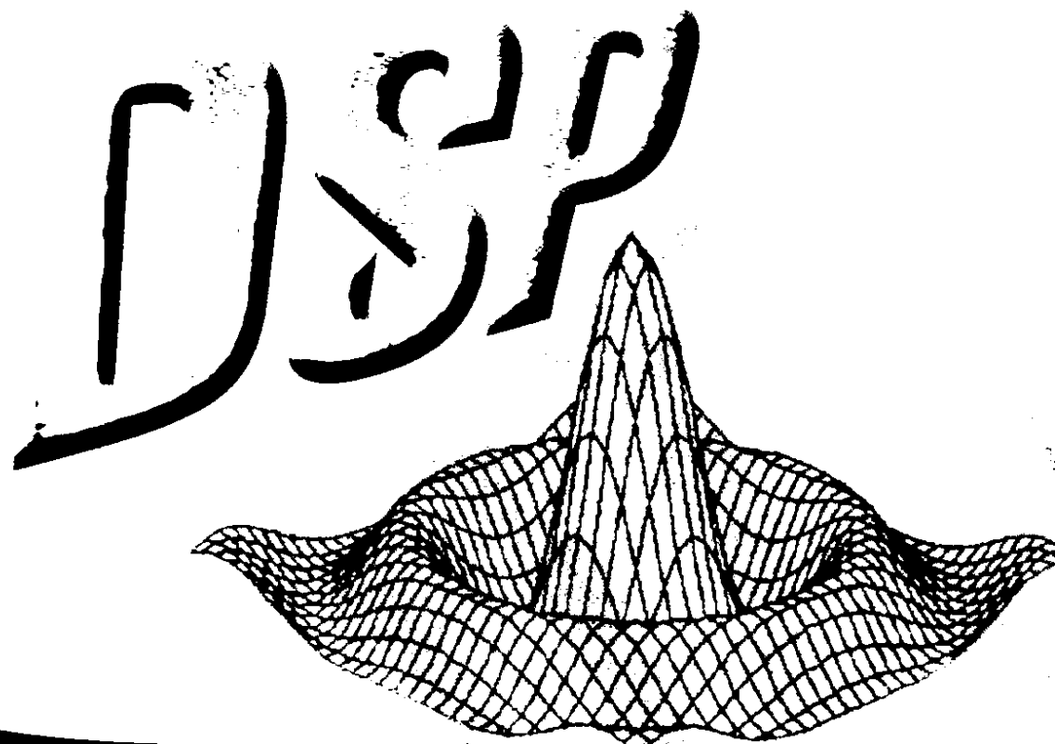
Spójrzmy raz jeszcze na maszynę splotową przedstawioną na rys. 6.8 i pomińmy fakt, że sygnał w ramce zaznaczonej linią przerywaną jest *odpowiedzią impulsową*. Pomyślmy o niej jako o zestawie **współczynników wagowych** wbudowanym w sieć działań. Przy takim podejściu każda próbka w sygnale wyjściowym jest równa *sumie wejść ważonych*. Na każdą próbkę w sygnale wyjściowym ma wpływ pewien obszar próbek w sygnale wejściowym, zgodnie w wybranymi współczynnikami wagowymi. Wyobraźmy sobie na przykład, że jest 10 współczynników wagowych, każdy o wartości 0,1. Wtedy każda próbka w sygnale wyjściowym jest *średnią* dziesięciu próbek wejściowych.

Idąc dalej, trzeba stwierdzić, że współczynniki wagowe nie muszą być ograniczone do znajdujących się *po lewej stronie* obliczanej próbki wyjściowej. Na przykład na rys. 6.8 przedstawiono obliczanie $y[6]$ na podstawie $x[3]$, $x[4]$, $x[5]$ i $x[6]$. Patrząc na maszynę kodową jako na sumę wejść ważonych, można wybierać współczynniki wagowe *symetrycznie* wokół próbki wyjściowej. Na przykład $y[6]$ może być obliczane na podstawie $x[4]$, $x[5]$, $x[6]$, $x[7]$ i $x[8]$. Przy zastosowaniu takiego samego zapisu indeksów jak na rys. 6.8 współczynniki wagowe dla tych pięciu wejść znajdowałyby się w $h[2]$, $h[1]$, $h[0]$, $h[-1]$ i $h[-2]$. A zatem odpowiedź impulsowa zgodna z naszym wyborem symetrycznych współczynników wagowych wymaga użycia *ujemnych indeksów*. Do tej kwestii wrócimy w następnym rozdziale.

Pod względem matematycznym koncepcja jest tylko jedna – splot określony wzorem 6.1. Jednak w różnych zastosowaniach naukowych i inżynierskich można podchodzić do problemu z dwóch różnych kierunków. Czasem chcemy myśleć o systemie w kategoriach jego odpowiedzi impulsowej, a czasem można zrozumieć system jako zestaw współczynników wagowych. Trzeba się oswoić z oboma podejściami i z przechodzeniem od jednego do drugiego.

7

Właściwości splotu



Właściwości systemu liniowego są w pełni określone przez jego odpowiedź impulsową, zgodnie z matematyczną operacją splotu. Jest to podstawą wielu metod przetwarzania sygnałów. A oto przykłady. Filtry cyfrowe tworzy się, *projektując* właściwe odpowiedzi impulsowe. Samoloty nieprzyjacielskie są wykrywane radarem przez *analizę* zmierzonych odpowiedzi impulsowych. Tłumienie echa w rozmowach telefonicznych na dużych odległościach realizuje się, tworząc odpowiedź impulsową, która *przeciwdziała* odpowiedzi impulsowej pogłosu. Tę listę można by ciągnąć dalej. Ten rozdział poświęcono właściwościom i zastosowaniu splotu w kilku dziedzinach. Po pierwsze, omówiono kilka najczęściej spotykanych rodzajów odpowiedzi impulsowych. Po drugie, opisano metody postępowania z kaskadowymi i równoległymi połączeniami systemów liniowych. Po trzecie, wprowadzono metodę *korelacji*. Po czwarte, poruszono nieprzyjemny problem związany ze splotem; czas obliczeń może stać się niedopuszczalnie długi przy użyciu konwencjonalnych algorytmów i komputerów.

Typowe odpowiedzi impulsowe

Funkcja delta

Najprostszą odpowiedzią impulsową jest po prostu funkcja delta, którą przedstawiono na rys. 7.1 a). Taka odpowiedź impulsowa oznacza, że impuls na wejściu wytwarza na wyjściu taki sam impuls. A zatem *wszystkie* sygnały przechodzą przez system *niezmienione*. Poddanie jakiegokolwiek sygnału splotowi z funkcją delta daje w rezultacie dokładnie taki sam sygnał. Można to zapisać matematycznie:

$$x[n] * \delta[n] = x[n]$$

Wzór 7.1. Funkcja delta jest tożsama ze splotem. Każdy sygnał poddany splotowi z funkcją delta pozostaje niezmieniony

Ta właściwość powoduje, że funkcja delta jest **elementem neutralnym** (*identity*) dla splotu. Jest to analogiczne do *zera* będącego elementem neutralnym dla dodawania ($a + 0 = a$) oraz do *jedności*, która jest elementem neutralnym dla mnożenia ($a \times 1 = a$). Na pierwszy rzut oka system tego rodzaju może wydawać się trywialny i nieinteresujący. Nic podobnego! Takie systemy świetnie się nadają do gromadzenia danych, łączności i pomiarów. W wielu zastosowaniach DSP mamy do czynienia z systemami przekazującymi informacje bez ich zmiany i pogorszenia parametrów.

Na rys. 7.1 b) przedstawiono niewielką modyfikację funkcji delta jako odpowiedzi impulsowej. Jeśli amplituda funkcji delta zostaje zwiększona lub zmniejszona, to uzyskuje się, odpowiednio, system **wzmacniacza** lub **tłumika**. Wyrażając to wzorem, uzyskujemy wzmacnianie, jeśli *k jest większe od 1*, a tłumienie, jeśli *k jest mniejsze od 1*:

$$x[n] * k\delta[n] = kx[n]$$

Wzór 7.2. System wzmacniający lub tłumiący ma jako odpowiedź impulsową przeskalowaną funkcję delta. W tym wzorze k określa wzmacnianie lub tłumienie

Odpowiedź impulsowa przedstawiona na rys. 7.1 c) jest funkcją delta **z przesunięciem**. Uzyskuje się wówczas system wprowadzający identyczne przesunięcie między sygnałem wejściowym i wyjściowym. Można to nazwać **opóźnieniem** lub **wyprzedzeniem** sygnału, zależnie od kierunku przesunięcia. Wyraźmy przesunięcie parametrem s. Wtedy możemy to napisać równanie:

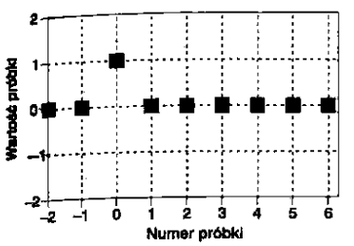
$$x[n] * \delta[n + s] = x[n + s]$$

Wzór 7.3. Względne przesunięcie między sygnałami wejściowym i wyjściowym wynika z odpowiedzi impulsowej będącej przesuniętą funkcją delta. Zmienna s w tym równaniu określa wartość przesunięcia

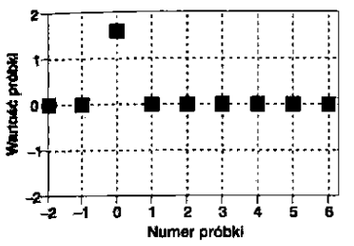
W nauce i technice jest mnóstwo przypadków przesunięcia jednej wersji sygnału w stosunku do innej. Jako przykład weźmy sygnał radiowy nadawany z odległej sondy kosmicznej oraz sygnał odbierany na Ziemi. Czas propagacji fal radiowych powoduje opóźnienie sygnału odbieranego w stosunku do nadawanego. W biologii sygnały elektryczne w sąsiadujących

komórkach nerwowych są względem siebie przesunięte o czas wynikający z przejścia potencjału czynnościowego przez złącze nerwowe między komórkami.

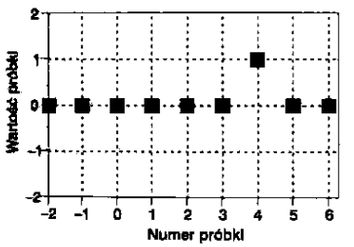
Na rys. 7.1 d) przedstawiono odpowiedź impulsową złożoną z funkcji delta oraz z przeskalowanej i przesuniętej funkcji delta. Zgodnie z zasadą superpozycji na wyjściu systemu uzyskuje się sygnał identyczny z wejściowym oraz jego opóźnioną wersję, czyli tzw. *echo*. Echo ma istotne znaczenie w wielu zastosowaniach DSP. Wprowadzanie echa jest ważnym



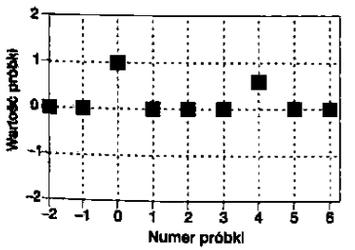
a. Element neutralny
Funkcja delta jest *elementem neutralnym* dla splotu. Splecenie sygnału z funkcją delta pozostawia sygnał niezmienionym. To właśnie jest celem systemów przesyłających lub gromadzących sygnały.



b. Wzmacnianie i tłumienie
Zwiększenie lub zmniejszenie amplitudy funkcji delta daje odpowiedź impulsową, która – odpowiednio – wzmacnia lub tłumi. Przedstawiona odpowiedź impulsowa spowoduje 1,6-krotne wzmacnianie sygnału.



c. Przesunięcie
Przesunięcie funkcji delta powoduje odpowiednie przesunięcie między sygnałami wyjściowym a wejściowym. Zależnie od kierunku może to być *opóźnienie* lub *wyprzedzenie*.
Przedstawiona odpowiedź impulsowa opóźnia sygnał o 4 próbki.



d. Echo
Odpowiedź impulsowa złożona z funkcji delta oraz przeskalowanej, przesuniętej funkcji delta powoduje dodanie echa do sygnału oryginalnego. W tym przykładzie echo jest opóźnione o 4 próbki i ma amplitudę równą 60% amplitudy sygnału oryginalnego.

Rys. 7.1. Proste odpowiedzi impulsowe będące przeskalowanymi i przesuniętymi funkcjami delta

elementem przy nagrywaniu dźwięku, który dzięki temu brzmi bardziej naturalnie i przyjemnie. W urządzeniach radarowych i sonarowych analizuje się sygnały echa w celu wykrycia samolotów i łodzi podwodnych. Geofizycy stosują echo do wykrywania złóż ropy. Zjawisko echa jest też istotne w sieciach telefonicznych, lecz tam chcemy go *uniknąć*.

Operacje podobne do różniczkowania i całkowania

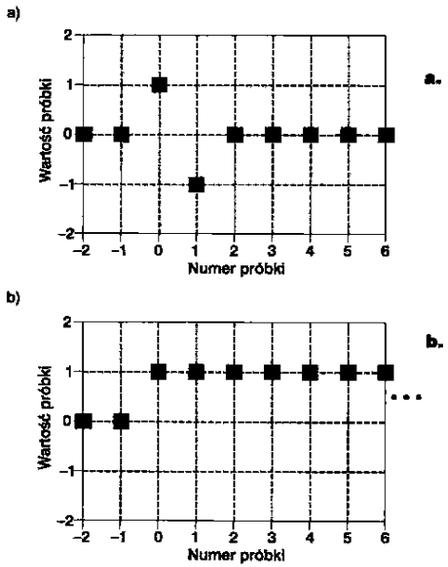
Splot może zmieniać sygnał dyskretny w sposób podobny do całkowania i różniczkowania. Ponieważ pojęcia „pochodna" i „całka" dotyczą operacji na sygnałach *ciągłych*, więc nie mają one inne nazwy ich odpowiednikom w przypadku sygnałów dyskretnych. Operacja dyskretna będąca naśladownictwem *pierwszej pochodnej* jest nazywana **pierwszą różnicą**. Podobnie dyskretna postać całki jest nazywana **sumą bieżącą**. Spotyka się też określenia tych operacji jako **pochodna dyskretna** oraz **całka dyskretna**, chociaż matematycy marszczą brwi, słysząc takie nieformalne nazwy.

Na rys. 7.2 przedstawiono odpowiedzi impulsowe realizujące pierwszą różnicę oraz sumę bieżącą, na rys. 7.3 zaś – przykład zastosowania tych operacji. Sygnał oryginalny (rys. 7.3 a) składa się z kilku fragmentów o zmieniającym się nachyleniu. Splecenie tego sygnału z odpowiedzią impulsową realizującą pierwszą różnicę daje sygnał przedstawiony na rys. 7.3 b). Tak jak w przypadku pierwszej pochodnej, wartość każdego punktu w sygnale pierwszej różnicy jest równa *nachyleniu* sygnału oryginalnego w tym punkcie. Suma bieżąca jest operacją odwrotną w stosunku do pierwszej różnicy. Tak więc poddanie splotowi sygnału z rys. 7.3 b) z odpowiedzią impulsową realizującą sumę bieżącą da sygnał przedstawiony na rys. 7.3 a).

Te odpowiedzi impulsowe są na tyle proste, że do ich implementacji nie jest konieczny pełny program obliczania splotu. Pomyślmy o nich raczej w sposób alternatywny, przyjmując, że każda próbka jest sumą ważonych próbek wejściowych. Na przykład pierwszą różnicę można obliczyć z wzoru:

$$y[n] = x[n] - x[n-1]$$

Wzór 7.4. Obliczanie pierwszej różnicy. W tej zależności x[n] jest sygnałem oryginalnym, y[n] – pierwszą różnicą



a. Pierwsza różnica
Jest to dyskretna wersja pierwszej pochodnej. Każda próbka w sygnale wyjściowym jest równa różnicy między sąsiednimi próbkami w sygnale wejściowym. Inaczej mówiąc, sygnał wyjściowy jest równy nachyleniu sygnału wejściowego.

b. Suma bieżąca
Suma bieżąca jest dyskretną wersją całki. Każda próbka w sygnale wyjściowym jest równa sumie wszystkich położonych na lewo próbek w sygnale wejściowym. Proszę zauważyć, że odpowiedź impulsowa rozciąga się aż do nieskończoności, co jest raczej niepożądaną cechą.

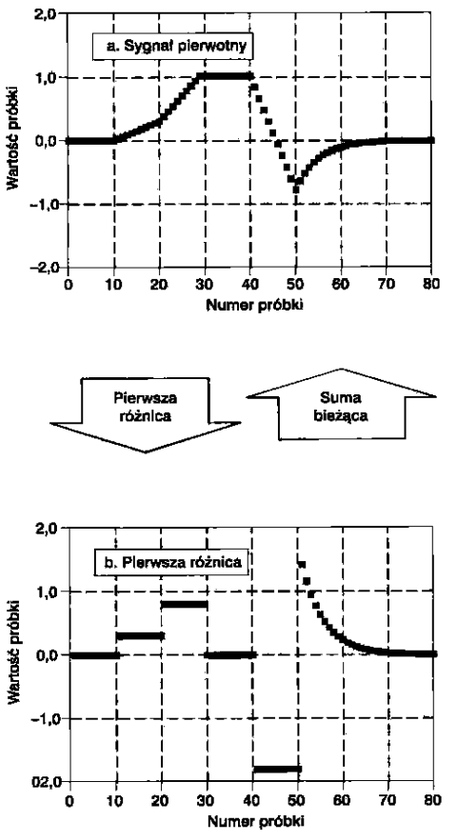
Rys. 7.2. Odpowiedzi impulsowe będące naśladownictwem operacji różniczkowania i całkowania

Z wzoru wynika, że każda próbka w sygnale wyjściowym jest równa różnicy między dwiema sąsiednimi próbkami w sygnale wejściowym. Na przykład $y[40] = x[40] - x[39]$. Trzeba zaznaczyć, że nie jest to jedyny sposób wyznaczania *pochodnej dyskretnej*. Inną, często stosowaną metodą, jest określanie nachylenia symetrycznie wokół wyznaczonego punktu, na przykład $y[n] = (x[n+1] - x[n-1])/2$.

Stosując takie samo podejście, można obliczyć każdą próbkę w sumie bieżącej przez sumowanie wszystkich punktów sygnału oryginalnego, znajdujących się *na lewo* od położenia próbki. Na przykład, jeśli $y[n]$ jest sumą bieżącą $x[n]$, to próbkę $y[40]$ oblicza się, sumując próbki od $x[0]$ do $x[40]$. W podobny sposób jest obliczana próbka $y[41]$ przez sumowanie próbek od $x[0]$ do $x[41]$. Obliczanie sum bieżących w taki sposób byłoby, oczywiście, bardzo nieefektywne. Można jednak – na przykład – jeśli $y[40]$ zostało już obliczone, obliczyć $y[41]$ przez proste dodawanie: $y[41] = x[41] + y[40]$. Postać wzoru ogólnego:

$$y[n] = x[n] + y[n-1]$$

Wzór 7.5. Obliczanie sumy bieżącej. W tej zależności x[n] jest sygnałem oryginalnym a y[n] sumą bieżącą



Rys. 7.3. Przykład operacji podobnych do różniczkowania i całkowania. Sygnał na rys. b) jest pierwszą różnicą sygnału z rys. a). Odpowiednio, sygnał z rys. a) jest sumą bieżącą sygnału z rys. b). Te metody przetwarzania są stosowane do sygnałów dyskretnych, tak samo jak różniczkowanie i całkowanie do sygnałów ciągłych

Tab. 7.1. Programy do obliczania pierwszej różnicy i sumy bieżącej. Sygnał oryginalny jest przechowywany w X[ ] a sygnał przetworzony (pierwsza różnica i suma bieżąca) – w Y[ ]. Obie metryce obejmują zakres od 0 do N% – 1

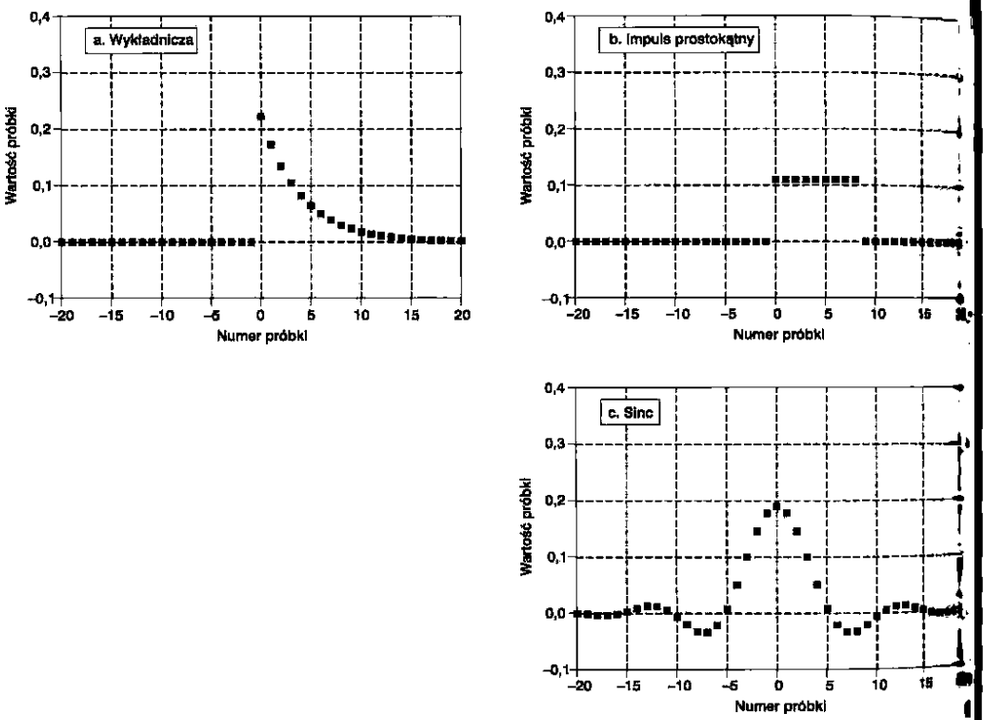
```
100 'Obliczenie pierwszej różnicy
110 Y[0]=0
120 FOR I%= 1 TO N%-1
130 Y[I%] = X[I%] - X[I%-1]
140 NEXT I%
```

```
100 'Obliczenie sumy bieżącej
110 Y[0]=X[0]
120 FOR I% = 1 TO N%-1
130 Y[I%] = Y[I%-1] + X[I%]
140 NEXT I%
```

Zależności tego rodzaju noszą nazwę **równań rekurencyjnych** lub **różnicowych**. Wrócimy do nich w rozdziale 19. Na razie ważne do zrozumienia jest, że te zależności są *identyczne* ze splotem z wykorzystaniem odpowiedzi impulsowych z rys. 7.2. W tablicy 7.1 zamieszczono programy komputerowe wykonujące operacje podobne do różniczkowania i całkowania.

Filtry dolno- i górnoprzepustowe

Projektowanie filtrów cyfrowych będzie szczegółowo omówione w dalszych rozdziałach. Na razie zadowólmy się zrozumieniem ogólnej postaci ich odpowiedzi impulsowych. Na rys. 7.4 przedstawiono kilka typowych odpowiedzi impulsowych filtrów dolnoprzepustowych. Ogólnie biorąc, te odpowiedzi impulsowe są złożone z grupy *dodatnich, sąsiadujących ze*

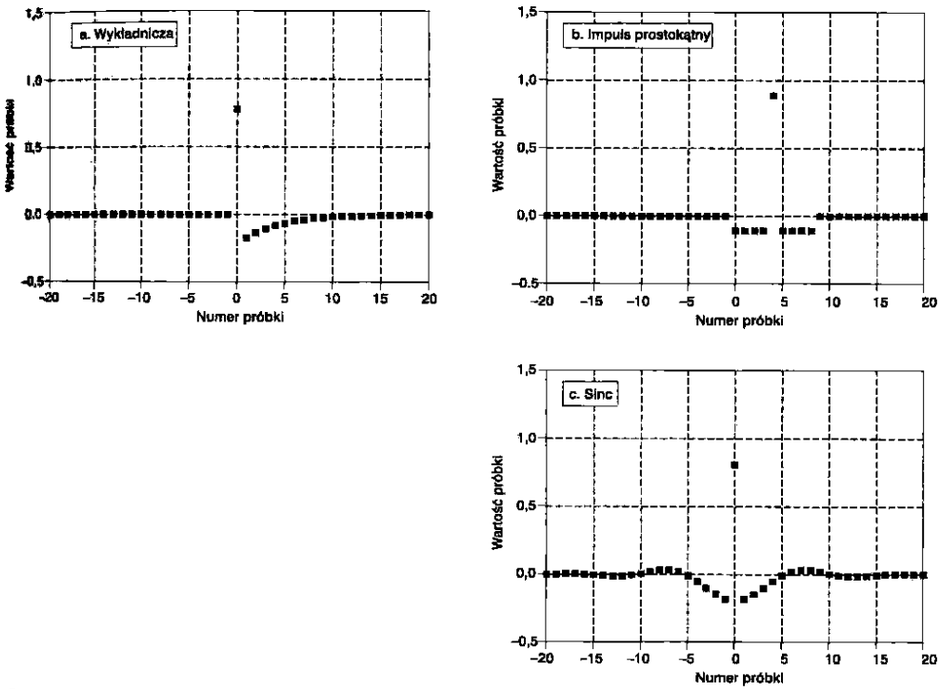


Rys. 7.4. Typowe odpowiedzi impulsowe cyfrowych filtrów dolnoprzepustowych. Odpowiedzi impulsowe tych filtrów są utworzone z grupy sąsiadujących ze sobą punktów o wartościach dodatnich, dzięki czemu uzyskuje się uśrednianie (wygładzanie) sygnału. Każda z tych odpowiedzi jest najlepsza do określonego celu, co będzie omówione w dalszych rozdziałach. Odpowiedź wykładnicza (rys. a) jest najprostszym filtrem rekursywnym. Odpowiedź o kształcie impulsu prostokątnego (rys. b) jest najbardziej odpowiednia do tłumienia szumu, utrzymując przy tym ostrość zboczy. Funkcja sinc (rys. c), czyli krzywa o kształcie sin(x)/x, jest stosowana do rozdzielania między sobą różnych pasm częstotliwości

sobą punktów. Dlatego każda próbka w sygnale wyjściowym jest średnią ważoną wielu sąsiadujących punktów w sygnale wejściowym. Takie uśredniania *wygładza* sygnał, a więc usuwa z niego składowe o dużych częstotliwościach. Jak to pokazano na przykładzie funkcji *sinc* (rys. 7.4 c), niektóre odpowiedzi częstotliwościowe filtrów zawierają kilka próbek o wartościach ujemnych. Podobnie jak w elektronice analogowej, dolnoprzepustowe filtry cyfrowe są stosowane do tłumienia szumu, kształtowania przebiegów itd.

Częstotliwość odcięcia filtru można zmieniać, zwężając lub rozszerzając odpowiedź impulsową filtru. Jeśli filtr dolnoprzepustowy ma wzmocnienie równe *jeden* dla przebiegu stałego (częstotliwość zerowa), to suma wszystkich punktów w odpowiedzi impulsowej musi być równa *jeden*. Jak to przedstawiono na rys. 7.4 a), c), niektóre odpowiedzi impulsowe filtrów *teoretycznie* rozciągają się aż do nieskończoności, nie schodząc do wartości zero. W praktyce ogony odpowiedzi impulsowej są obcinane po pewnej liczbie próbek, tak aby odpowiedź mogła być wyrażona skończoną liczbą punktów. Bo jakby inaczej dało się taką odpowiedź zarejestrować w komputerze?

Na rys. 7.5 przedstawiono trzy typowe odpowiedzi impulsowe filtrów górnoprzepustowych, utworzone z odpowiadających im odpowiedzi impulsowych filtrów dolnoprzepustowych z rys. 7.4. Jest to metoda często stosowana przy projektowaniu filtrów. Najpierw obmyśla się filtr dolnoprzepustowy i potem przekształca go w filtr wymagany – górnoprzepustowy, pasmowoprzepustowy, pasmozaporowy itd. Aby zrozumieć przekształcenie filtru dolnoprzepustowego w górnoprzepustowy, przypomnijmy sobie, że odpowiedź impulsowa w postaci funkcji delta przepuszcza cały sygnał, podczas gdy dolnoprzepustowa odpowiedź impulsowa przepuszcza tylko składowe małoczęstotliwościowe. W wyniku superpozycji

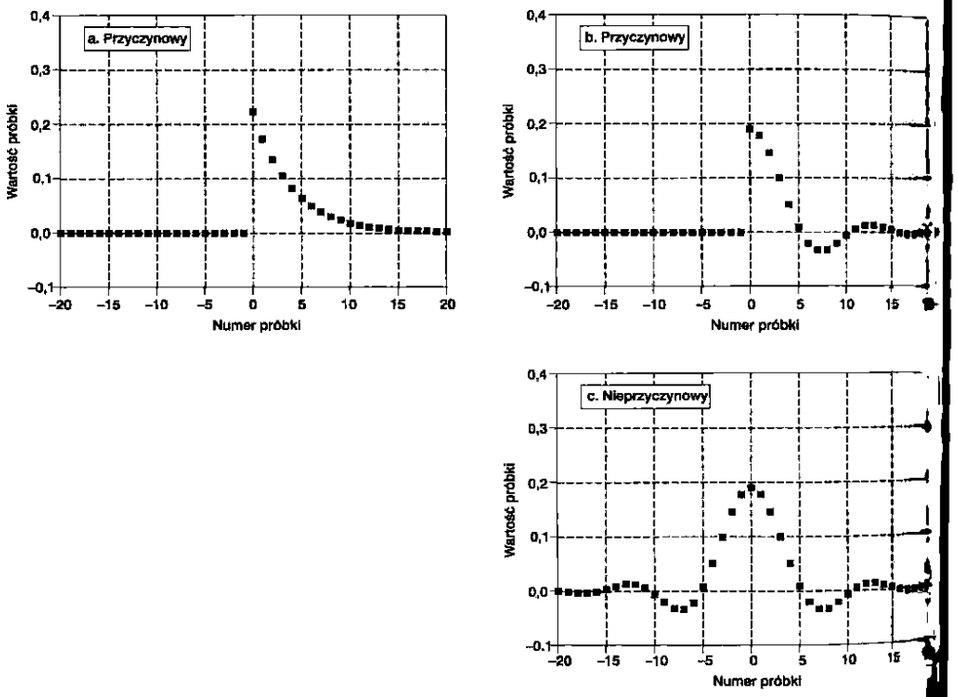


Rys. 7.5. Typowe odpowiedzi impulsowe filtrów górnoprzepustowych. Są one utworzone przez odjęcie odpowiednich odpowiedzi impulsowych filtrów dolnoprzepustowych (z rys. 7.4) od funkcji delta. Charakterystyczną cechą wyróżniającą odpowiedzi impulsowe filtrów górnoprzepustowych jest szpilkowy wyskok otoczony przez wiele sąsiednich próbek o wartościach ujemnych

odpowiedź impulsowa filtru, będąca różnicą funkcji delta i odpowiedzi dolnoprzepustowej, będzie przepuszczać różnicę całego sygnału i składowych małoczęstotliwościowych. A więc powstaje w ten sposób filtr górnoprzepustowy. Jak to pokazano na rys. 7.5, funkcja delta jest zwykle dodawana w środku symetrii odpowiedzi impulsowej albo przy próbce 0, jeśli odpowiedź nie jest symetryczna. Filtry górnoprzepustowe mają zerowe wzmocnienie przy częstotliwości zero (dla przebiegu stałego), co uzyskuje się przez uczynienie sumy wszystkich punktów w odpowiedzi filtru równą *zeru*.

Sygnały przyczynowe i nieprzyczynowe

Wyobraźmy sobie prosty układ elektroniki analogowej. Jeśli na jego wejście wprowadzimy krótki impuls, to na wyjściu zobaczymy odpowiedź układu. Jest to przykład zasady przyczyny i skutku, obowiązującej w naszym Wszechświecie. Jedną rzecz wiemy na pewno: *każdy skutek musi nastąpić po jego przyczynie*. Porównajmy to z działaniem systemu DSP, które zamienia sygnał wejściowy na wyjściowy, a oba te sygnały są gromadzone w pamięci komputera. Jeśli takie działanie jest naśladownictwem świata rzeczywistego, to musi zachować tę samą zasadę przyczynowości, jaka obowiązuje w świecie rzeczywistym. Na przykład wartość próbki numer 8 w sygnale wejściowym może w sygnale wyjściowym wpływać tylko na próbki o numerze 8 lub większym. Systemy działające zgodnie z tą zasadą nazywamy **przyczynowymi**. Przetwarzanie cyfrowe, oczywiście, niekoniecznie musi działać w ten sposób. Ponieważ zarówno sygnał wejściowy, jak i wyjściowy są zestawami liczb pamiętanymi w komputerze, więc każda wartość sygnału wejściowego może mieć wpływ na każdą wartość sygnału wyjściowego.



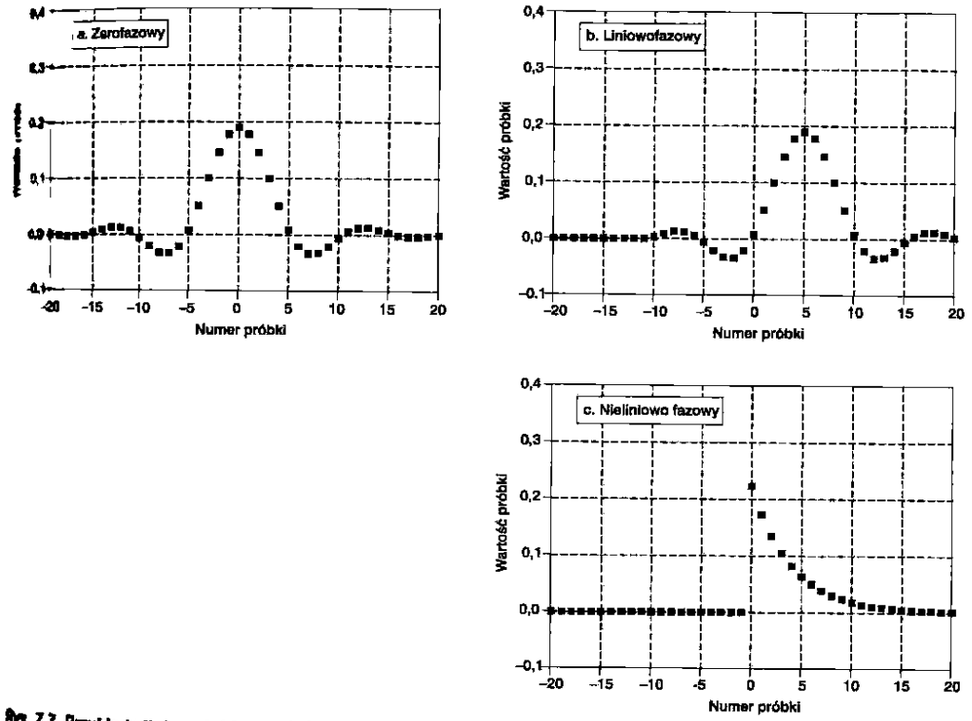
Rys. 7.6. Przykłady sygnałów przyczynowych. Odpowiedź impulsowa lub każdy inny sygnał jest określany jako przyczynowy, jeśli wszystkie jego próbki o numerach ujemnych mają wartość zero. Pokazano tu 3 przykłady. Każdy sygnał nieprzyczynowy o skończonej liczbie punktów można przekształcić w przyczynowy po prostu przez przesunięcie

Jak wynika z przykładów podanych na rys. 7.6, odpowiedź impulsowa systemu przyczynowego musi mieć wartości zerowe we wszystkich próbkach o *numerach ujemnych*. Pomyślmy o tym w aspekcie splotu rozważanego od strony wejścia. Impuls w sygnale wejściowym, w próbce numer *n* musi, aby sygnał był przyczynowy, wpływać w sygnale wyjściowym, tylko na próbki o numerach *n* lub większych. Na ogół określenia *przyczynowy* używa się do *każdego* sygnału, w którym wszystkie próbki o numerach ujemnych mają wartość zero, niezależnie pod tego, czy jest on odpowiedzią impulsową, czy nie.

Faza zerowa liniowa, liniowa i nieliniowa

Mówimy o sygnale, że jest **zerofazowy**, jeśli charakteryzuje się symetrią lewo-prawostronną wokół próbki o numerze zero. Przykład takiego sygnału przedstawiono na rys. 7.7 a). Sygnał jest **liniowofazowy**, jeśli ma symetrię lewo-prawostronną, lecz wokół jakiegoś punktu różnego od zera (rys. 7.7 b). Z tego wynika, że każdy sygnał liniowofazowy można zamienić na zerofazowy, po prostu przesuwając go w lewo lub w prawo. Wreszcie, mówimy o sygnale, że jest **nieliniowofazowy**, jeśli nie ma symetrii lewo-prawostronnej (rys. 7.7 c).

Może się wydawać, że wymienione nazwy nie mają nic wspólnego z ich definicjami. Bo cóż może mieć wspólnego *faza* z *symetrią*? Odpowiedzi na to pytanie należy szukać w widmie częstotliwościowym; w dalszych rozdziałach ta kwestia będzie omówiona bardziej szczegółowo. Mówiąc w skrócie, widmo częstotliwościowe każdego sygnału składa się z dwóch części – widm modułu i fazy. Widmo częstotliwościowe sygnału, które jest symetryczne wokół



Rys. 7.7. Przykłady liniowości fazowej. Sygnały charakteryzujące się symetrią lewo-prawostronną określa się jako liniowofazowe. Jeśli oś symetrii przechodzi przez próbkę numer zero, to dodatkowo określa się taki sygnał jako zerofazowy. Każdy sygnał liniowofazowy można przekształcić w zerofazowy, po prostu go przesuwając. Sygnały niemające symetrii lewo-prawostronnej są nieliniowofazowymi. Proszę nie mylić tych określeń z liniowością w systemach liniowych. To są dwa zupełnie różne pojęcia

zera, ma fazę zerową. Podobnie widmo częstotliwościowe sygnału, symetryczne wokół kiegoś punktu niezerowego, ma charakterystykę fazową będącą dokładnie linią prostą, a ma fazę liniową. Na koniec, widmo częstotliwościowe sygnału, które nie jest symetryczne ma charakterystykę fazową niebędącą linią prostą, tzn. ma fazę nieliniową.

Należy jeszcze zwrócić uwagę na pojęcia, które mogą być mylące – *faza liniowa* i *nieliniowa*. Co to ma wspólnego z pojęciem liniowości systemu, omawianym w poprzednich rozdziałach? Otóż zupełnie nic. Liniowość systemu jest pojęciem o szerokim znaczeniu, które jest podstawą niemal całego DSP (superpozycja, homogeniczność, addytywność itd.). *Liniowość* lub *nieliniowość fazy* określa, czy charakterystyka fazowa jest, czy nie jest linią prostą. W istocie system musi być *liniowy*, aby nawet tylko móc określić, czy faza jest zerowa, liniowa czy nieliniowa.

Właściwości matematyczne

Przemienność operacji splotu

Przemienność splotu wyraża się wzorem:

$$a[n] * b[n] = b[n] * a[n]$$

Wzór 7.6. Przemienność splotu. Oznacza ona, że można zmieniać kolejność, w jakiej sygnały są poddawane operacji splotu

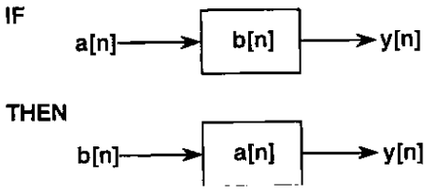
Powyższa zależność wyrażona słownie oznacza, że kolejność, w jakiej dwa sygnały są poddawane operacji splotu, nie ma znaczenia; wyniki są identyczne. Nabiera to, jak wynika z rys. 7.8, szczególnego sensu w teorii sygnałów. W każdym systemie liniowym sygnał wejściowy i odpowiedź impulsowa systemu mogą być *zamienione* miejscami bez zmiany sygnału wyjściowego. Jest to bardzo interesujące stwierdzenie, które zwykle nie ma jednak żadnego znaczenia fizycznego. Sygnał wejściowy i odpowiedź impulsowa to dwie zupełnie różne rzeczy. Z tego tylko powodu, że zależności matematyczne *zezwalają* na jakieś operacje, nie wynika, że należy je wykonywać. Na przykład załóżmy, że zarabiasz 10 USD/godz. × 2000 godz./rok = 20 000 USD/rok. Z przemienności mnożenia wynika, że można uzyskać taki sam roczny dochód, pracując tylko 10 godz./rok przy płacy 2000 USD/godz. Spróbujcie przekonać waszego szefa, że to ma sens! Mimo to cecha przemienności ma duże zastosowanie w DSP do przekształcenia równań, jak w zwykłej algebrze.

Łączność operacji splotu

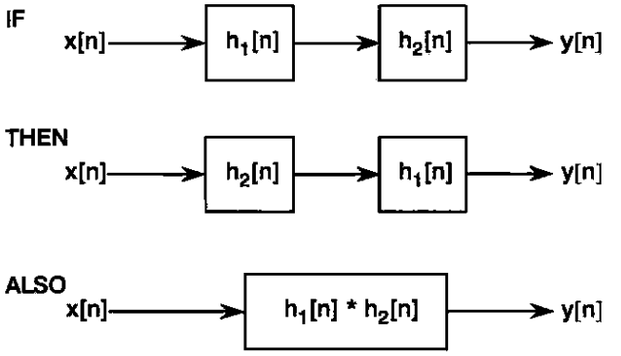
Czy można poddać operacji splotu trzy lub więcej sygnałów? Odpowiedź brzmi: tak, a z łączności operacji splotu wynika, jak to zrobić. Należy spleść dwa sygnały, aby uzyskać sygnał pośredni. Następnie trzeba ten sygnał pośredni spleść z trzecim sygnałem. Z faktu, że splot jest działaniem łącznym, wynika, iż kolejność splotów nie ma znaczenia. Można to wyrazić równaniem

$$(a[n] * b[n]) * c[n] = a[n] * (b[n] * c[n])$$

Wzór 7.7. Zasada łączności splotu określa, jak poddawać splotowi trzy lub więcej sygnałów



Rys. 7.8. Cecha przemienności w teorii systemów. Przemienność splotu zezwala na zamianę miejscami sygnału wejściowego i odpowiedzi impulsowej systemu, bez zmiany wyniku na wyjściu systemu. Co ciekawe, ta właściwość nie ma na ogół żadnego znaczenia fizycznego. (Na tym rysunku sygnał wewnątrz nazwany jak b[n] i a[n], wyraża odpowiedź impulsową systemu)



Rys. 7.9. Zasada łączności w teorii systemów. Ta zasada określa dwie ważne właściwości kaskadowych systemów liniowych. Po pierwsze, kolejność systemów można zmieniać bez zmiany działania kaskady jako całości. Po drugie, dwa lub więcej systemów kaskadowych można zastąpić systemem pojedynczym. Odpowiedź impulsową tego systemu określa się splotem odpowiedzi impulsowych poszczególnych stopni pierwotnych, które zastąpiono jednym systemem

Zasadę łączności stosuje się w teorii systemów do opisu zachowania **systemów kaskadowych** (rys. 7.9). Dwa lub więcej systemów jest połączonych *kaskadowo*, jeśli wyjście jednego systemu jest używane jako wejście następnego. Według zasady łączności kolejność systemów w kaskadzie można zmieniać bez zmiany odpowiedzi całego systemu kaskadowego. Ponadto, każdą liczbę systemów kaskadowych można zamienić *pojedynczym* systemem. Odpowiedź impulsową tego systemu wyznacza się jako splot odpowiedzi impulsowych wszystkich systemów pierwotnych.

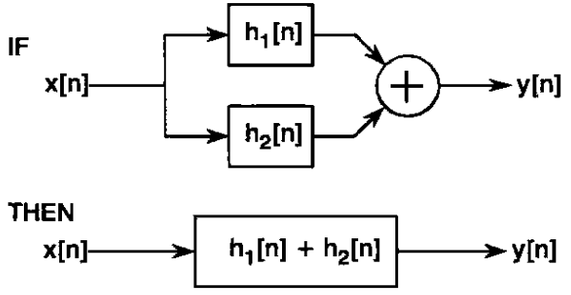
Rozdzielność splotu względem dodawania

Rozdzielność splotu względem dodawania wyraża się wzorem:

$$a[n] * b[n] + a[n] * c[n] = a[n] * (b[n] + c[n])$$

Wzór 7.8. Rozdzielność operacji splotu opisuje, jak analizować systemy równoległe

Z rozdzielności splotu wynika sposób analizy **systemów równoległych z sumowanymi wyjściami**. Jak to przedstawiono na rys. 7.10, dwa lub więcej systemów może mieć wspólne wejście $x[n]$, a ich sygnały wyjściowe są sumowane, dając $y[n]$. Dzięki temu, że splot jest operacją rozdzielną, można taką kombinację systemów zastąpić pojedynczym systemem o odpowiedzi impulsowej równej *sumie* odpowiedzi impulsowych systemów pierwotnych.



Rys. 7.10. Właściwość rozdzielności względem dodawania w teorii systemów. Ta właściwość powoduje, że systemy równoległe z sumowanymi wyjściami można zastępować pojedynczym systemem. Odpowiedź impulsowa tego systemu zastępczego jest równa sumie odpowiedzi impulsowych wszystkich systemów pierwotnych

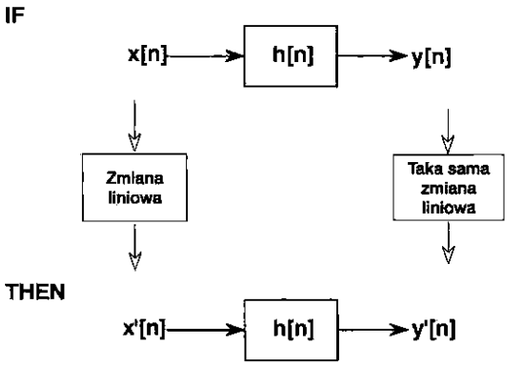
Przenoszenie między wejściem a wyjściem

Nie jest to formalna właściwość matematyczna, a raczej sposób myślenia o sytuacji często spotykanej w dziedzinie przetwarzania sygnałów. Wyobraźmy sobie system liniowy mający na wejściu sygnał $x[n]$ i wytwarzający na wyjściu sygnał $y[n]$, jak to przedstawiono na rys. 7.11. Przyjmijmy teraz, że sygnał wejściowy zmienia się w jakiś sposób liniowy, tworząc nowy sygnał wejściowy, który oznaczymy $x'[n]$. Jest wtedy wytwarzany nowy sygnał wyjściowy $y'[n]$. Powstaje pytanie, jaki jest związek zmiany sygnału wejściowego ze zmianą na wyjściu. Odpowiedź na to pytanie brzmi: *sygnał wyjściowy zmienia się dokładnie w ten sam, liniowy sposób, jak zmienił się sygnał wejściowy*. Na przykład, jeśli sygnał wejściowy został wzmocniony o czynnik 2, to sygnał wyjściowy też będzie wzmocniony o taki sam czynnik. Jeśli sygnał wejściowy jest zróżniczkowany, to i wyjściowy będzie zróżniczkowany. Jeśli sygnał wejściowy zostanie w jakiś sposób odfiltrowany, to wyjściowy też będzie odfiltrowany w identyczny sposób. Można to łatwo wykazać, korzystając z faktu, że splot jest operacją łączną.

Centralne twierdzenie graniczne

Centralne twierdzenie graniczne jest ważnym narzędziem w teorii prawdopodobieństwa, gdyż wyjaśnia ono matematycznie, dlaczego rozkład prawdopodobieństwa Gaussa jest tak często spotykany w przyrodzie. A oto przykłady. Wartość szumu cieplnego w układach elektronicznych podlega rozkładowi Gaussa; gaussowskie jest też natężenia wiązki laserowej w przekroju poprzecznym; charakter gaussowski ma nawet rozkład trafień wokół tzw. oka byka na tarczy do gry w strzałki. Zgodnie z najprostszą postacią centralnego twierdzenia granicznego rozkład prawdopodobieństwa Gaussa powstaje wówczas, gdy obserwowana zmienna jest sumą wielu procesów przypadkowych. Nawet jeśli procesy składowe nie mają rozkładu Gaussa, to suma będzie miała taki rozkład.

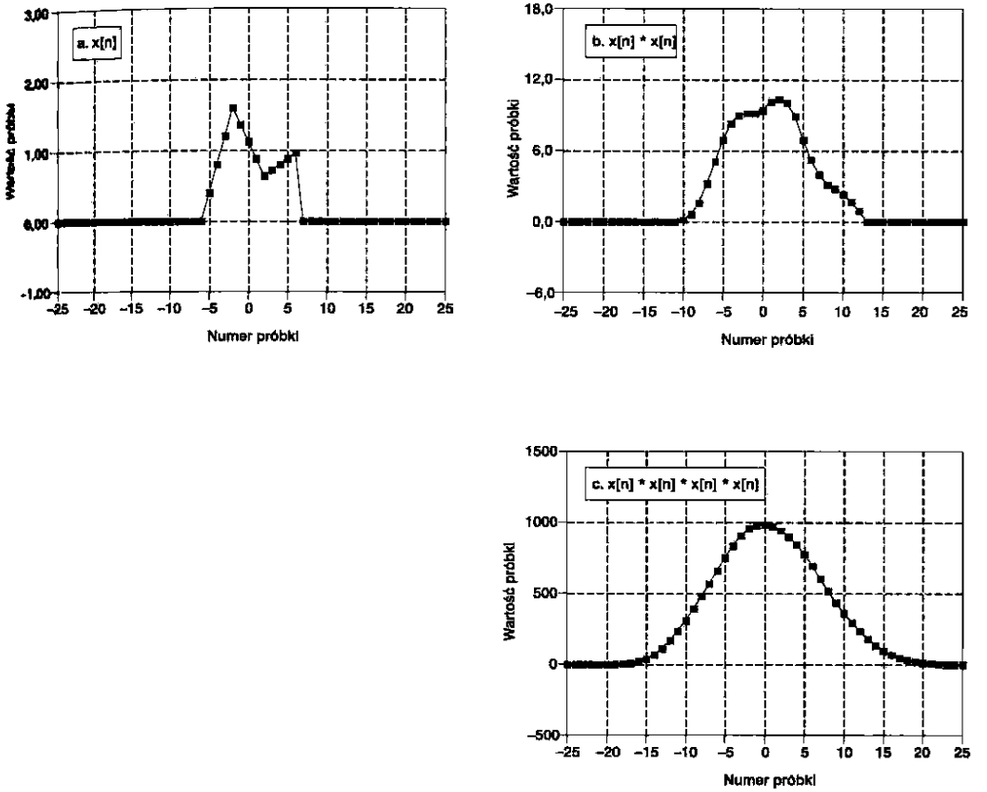
Ciekawe są implikacje centralnego twierdzenia granicznego związane ze splotem. Jeśli sygnał o kształcie podobnym do impulsu jest wielokrotnie splatany *sam za sobą*, to powstaje krzywa o kształcie rozkładu Gaussa. Przykład podano na rys. 7.12. Sygnał na rys. 7.12 a) jest nieregularnym impulsem, specjalnie tak dobranym, aby nie był podobny do przebiegu gaussowskiego. Na rys. 7.12 b) pokazano wynik *jednokrotnego* poddania tego sygnału splotowi z samego z sobą, a na rys. 7.12 c) – wynik po *trzykrotnym* wykonaniu tej operacji. Nawet po tylko trzech operacjach splotu przebieg jest bardzo zbliżony do krzywej Gaussa. W języku matematycznym mówi się, że taka procedura powoduje *bardzo szybką zbieżność* do krzywej gaussowskiej. Szerokość uzyskanego rozkładu Gaussa (tj. σ we wzorach 2.7 lub 2.8) jest równa szerokości orginalnego impulsu (wyrażonej jako σ we wzorze 2.7) pomnożonej przez pierwiastek kwadratowy z liczby wykonanych operacji splotu.



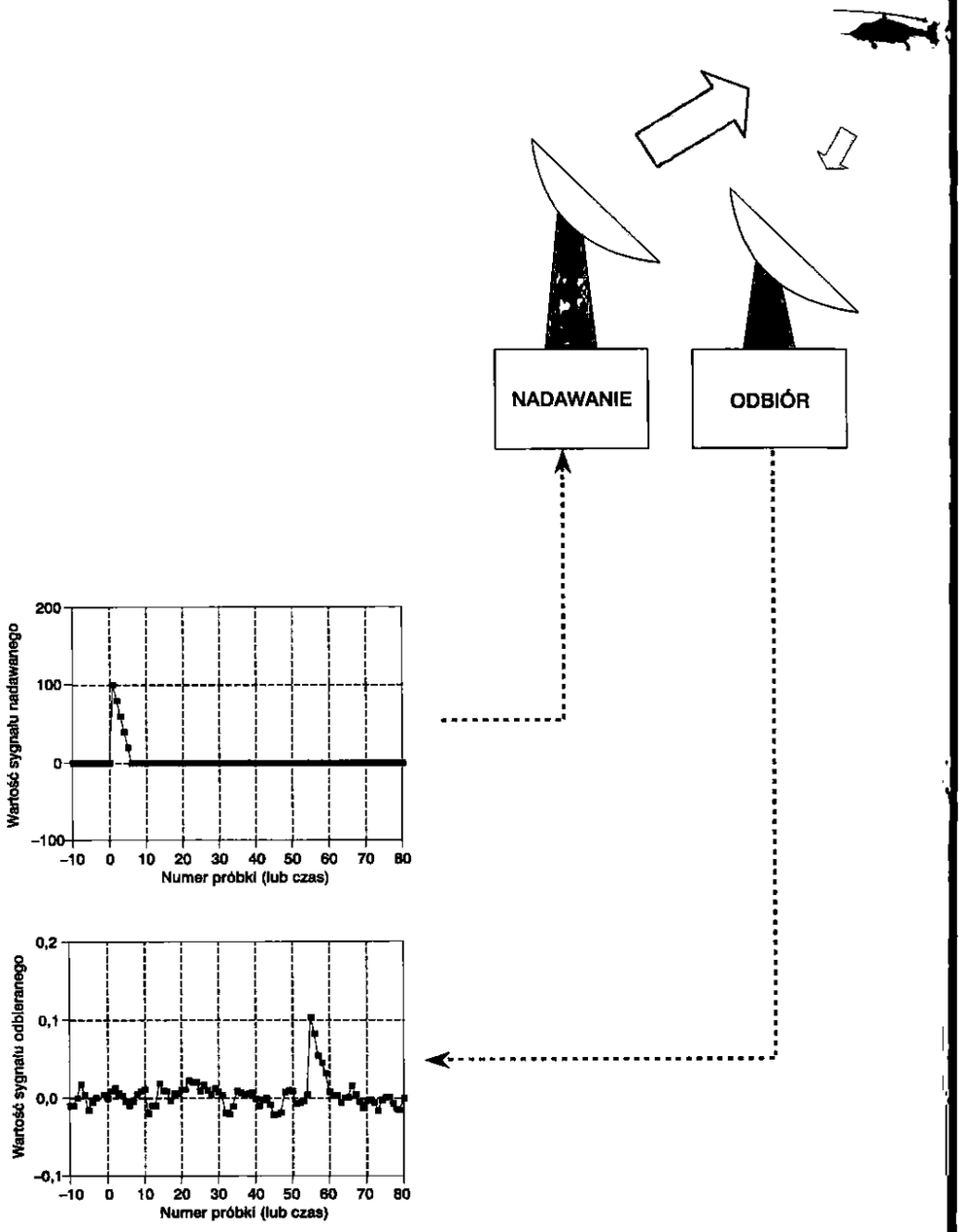
Rys. 7.11. Przenoszenie między wejściem a wyjściem. Jest to sposób myślenia o sytuacji często spotykanej w dziedzinie przetwarzania sygnałów. Liniowa zmiana wprowadzona do sygnału wejściowego skutkuje taką samą liniową zmianą sygnału na wyjściu

Korelacja

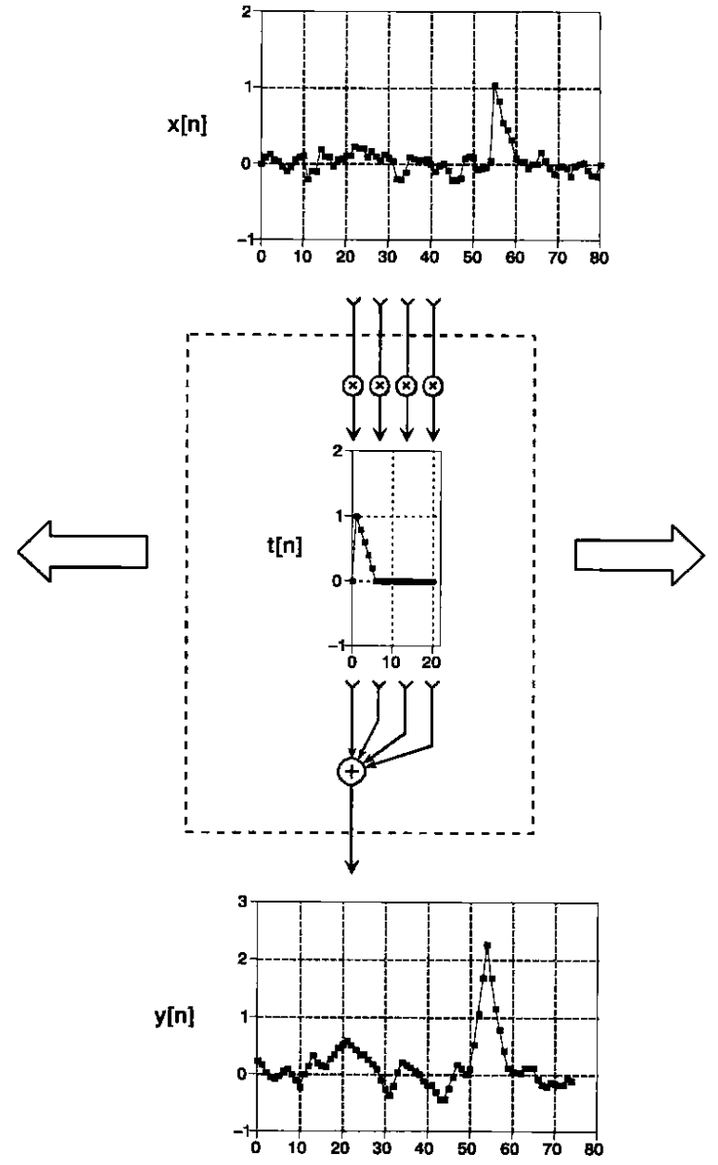
Pojęcie korelacji najlepiej przedstawić na przykładzie. Na rys. 7.13 pokazano najważniejsze elementy systemu radarowego. Specjalnie zaprojektowana antena wysyła w wybranym kierunku krótką wiązkę fal radiowych. Jeśli rozchodząca się fala napotka jakiś obiekt, np. pokazany na rysunku śmigłowiec, mała część energii jest odbijana wstecz w kierunku odbiornika fal radiowych, umieszczonego w pobliżu nadajnika. Przesyłany impuls ma określony kształt, który mu nadano, np. trójkątny, jak w omawianym przykładzie. Odebrany sygnał będzie się składał z dwóch części: z przesuniętej i przeskalowanej wersji wysłanego impulsu oraz z szumu losowego powstającego na skutek zakłóceń fal radiowych, szumu cieplnego w podzespołach elektronicznych itd. Ponieważ fale radiowe poruszają się ze znaną prędkością, czyli z prędkością światła, więc przesunięcie między impulsami nadanym i odebranym jest bezpośrednią miarą odległości od wykrytego obiektu. I tu jest problem do rozwiązania – mając sygnał o danym kształcie, trzeba znaleźć najlepszy sposób stwierdzenia, gdzie (i czy) ten sygnał mieści się w *innym* sygnale. Rozwiązaniem tego problemu jest korelacja.



Rys. 7.12. Przykład wykonywania operacji splotu przebiegu impulsowego z tym samym przebiegiem. Zgodnie z centralnym twierdzeniem granicznym przebieg gaussowski jest wytwarzany, gdy dowolnie ukształtowany impuls jest wielokrotnie poddawany splotowi z samym sobą. Na rys. a) podano przykład impulsu. Na rys. b) impuls, po jednokrotnym splocie z samym sobą, zaczyna być gładki i regularny, a na rys. c) po trzech takich operacjach jest dobrym przybliżeniem krzywej gaussowskiej



Rys. 7.13. Najważniejsze elementy systemu radarowego. Podobnie jak w innych systemach echolokacyjnych, nadajnik radarowy wysyła krótki impuls energii, który ulega odbiciu od poszukiwanego obiektu. Przebieg odebrany jest więc przesuniętą wersją przebiegu nadanego z dodanym szumem losowym. Wykrycie znanego przebiegu w zaszumionym sygnale jest podstawowym zadaniem w echolokacji. Rozwiązanie tego zagadnienia realizuje się przez korelację



Rys. 7.14. Maszyna korelacyjna. Jest to sieć działań pokazująca, w jaki sposób jest obliczana korelacja wzajemna dwóch sygnałów. W tym przykładzie y[n] jest korelacją wzajemną x[n] i t[n]. Zaznaczona linią przerywaną ramka jest przesuwana w lewo lub w prawo tak, aby jej wyjście wskazywało na tę próbkę w y[n], która właśnie jest obliczana. Wskazane próbki z x[n] są mnożone przez odpowiednie próbki w t[n] i iloczyny są sumowane. Maszyna korelacyjna jest identyczna z maszyną splotową (rys. 6.8 i 6.9), poza tym że sygnał wewnątrz ramki nie jest odwracany lewo-prawostronnie. W podanym przykładzie są obliczane tylko te próbki y[n], dla których t[n] jest całkowicie zanurzone w x[n]

Korelacja jest operacją matematyczną bardzo podobną do splotu. Podobnie jak w splocie w korelacji występują dwa sygnały dające w wyniku trzeci. Ten trzeci sygnał jest nazywany **korelacją wzajemną** dwóch sygnałów wejściowych. Jeśli sygnał jest poddany korelacji *ze sobą*, to uzyskany wynik nosi nazwę **autokorelacji**. W poprzednim rozdziale zaprezentowano maszynę splotową, aby pokazać, jak splot jest realizowany. Na rys. 7.14 w podobny sposób zaprezentowano tzw. **maszynę korelacyjną**. Sygnał odbierany $x[n]$ i sygnał korelacji wzajemnej $y[n]$ przedstawiono u góry i u dołu rysunku. Przebieg $t[n]$, którego szukamy w sygnale odbieranym – zwany **sygnałem docelowym** (*target*), jest zawarty *wewnątrz* maszyny korelacyjnej. Każda próbka w $y[n]$ jest obliczana przez przesuwanie maszyny korelacyjnej w lewo lub w prawo, aż wskaże próbkę właśnie obliczaną. Następnie wskazane próbki sygnału odbieranego wpadają do maszyny korelacyjnej i są przemnażane przez wartości odpowiednich punktów w sygnale docelowym. Suma tych iloczynów jest następnie przesuwana do odpowiedniej próbki w sygnale korelacji wzajemnej.

Wartość każdej próbki w sygnale korelacji wzajemnej jest miarą tego, jak dalece odbierany sygnał *jest podobny* do sygnału docelowego, *w tym miejscu*. Wynika stąd, że w sygnale korelacji wzajemnej pik będzie powstawał dla każdego sygnału docelowego, który jest obecny w sygnale odbieranym. Inaczej mówiąc, wartość korelacji wzajemnej ma maksimum, gdy sygnał docelowy jest *dopasowany* swymi właściwościami do takich samych cech sygnału odbieranego.

A co się dzieje, jeśli sygnał docelowy zawiera próbki o wartościach ujemnych? To niczego nie zmienia. Wyobraźmy sobie, że maszyna korelacyjna jest ustawiona w taki sposób, że sygnał docelowy jest idealnie dopasowany do odpowiedniego przebiegu w sygnale odbieranym. Gdy próbki z sygnału odbieranego wpadają do maszyny korelacyjnej, są wówczas mnożone przez zgodne z nimi próbki w sygnale docelowym. Jeśli pominąć szum, próbki o wartości dodatniej będzie mnożona przez samą siebie, dając w wyniku liczbę dodatnią. Nawet jeśli sygnał docelowy jest w całości ujemny, to pik przebiegu korelacji wzajemnej będzie jednak dodatni.

Jeśli sygnał odbierany zawiera szum, to szum będzie również w sygnale korelacji wzajemnej. Nie można uniknąć podobieństwa szumu losowego, w pewnym stopniu, do każdego – jakie byśmy wybrali – sygnału docelowego. Szum występujący w sygnale korelacji wzajemnej jest po prostu miarą tego podobieństwa. Pik wytwarzany na sygnale korelacji wzajemnej jest, poza tym szumem, symetryczny (lewa strona w stosunku do prawej). To stwierdzenie jest prawdziwe, nawet jeśli sygnał docelowy nie jest symetryczny. Ponadto szerokość tego piku jest dwa razy większa niż szerokość sygnału docelowego. Pamiętajmy, że sygnał korelacji wzajemnej próbuje *wykrywać* sygnał docelowy, a nie go *odtwarzać*. Nie ma żadnej przyczyny, aby spodziewać się, że ten pik będzie wyglądał podobnie jak sygnał docelowy.

Korelacja jest *optymalną* metodą wykrywania znanych przebiegów w szumie przypadkowym. Tak więc pik, przy użyciu korelacji, wznosi się bardziej powyżej szumu niż może być wytwarzany przez każdy inny system liniowy. (Mówiąc ideałnie poprawnie – metoda korelacji jest optymalna tylko dla *losowego szumu białego*). Stosowanie korelacji do wykrywania znanego przebiegu jest często nazywane **filtracją dopasowaną**. Powiemy o niej więcej w rozdziale 17.

Maszyny korelacyjna i splotowa są identyczne, poza jedną niewielką różnicą. Jak już opisano w poprzednim rozdziale, sygnał wewnątrz maszyny splotowej jest odwracany lewo-prawostronnie. To oznacza, że próbki o numerach 1,2,3,... są ustawione od prawej do lewej strony. W maszynie korelacyjnej nie ma takiego odwracania i ustawienie próbek jest w normalnym kierunku. Ponieważ lewo-prawostronne odwracanie sygnału jest jedyną różnicą między dwiema operacjami, więc można wyrazić *korelację*, stosując takie same wzory matematyczne jak dla splotu. Wymaga to *wstępnego lewo-prawostronnego odwrócenia* jednego lub dwóch sygnałów korelowanych w celu skasowania odwracania z

do prawej nieodłącznie związanego ze splotem. Na przykład, gdy $a[n]$ i $b[n]$ są splatane, dając w rezultacie $c[n]$, to piszemy równanie $a[n] * b[n] = c[n]$. Korelacja wzajemna $a[n]$ i $b[n]$ może więc być zapisana równaniem $a[n] * b[-n] = c[n]$. A zatem lewo-prawostronne odwracanie $b[n]$ jest realizowane przez odwrócenie znaku w indeksie, tj. $b[-n]$.

Niech was nie zmyli matematyczne podobieństwo między splotem i korelacją – to są dwie bardzo różne procedury DSP. Splot jest wzajemną zależnością między impulsem wejściowym systemu, sygnałem wyjściowym i odpowiedzią impulsową. Korelacja jest sposobem wykrywania znanego przebiegu w tle szumu. Podobieństwo operacji matematycznych jest tylko dogodną koincydencją.

Szybkość

Napisanie programu realizującego operację splotu między dwoma sygnałami jest prostym zadaniem; wymaga tylko kilku wierszy programu. Wykonywanie tego programu może być bardziej kłopotliwe. Trudnością jest duża liczba sumowań i mnożeń wymaganych w algorytmie, czego skutkiem jest długi czas wykonywania programu. Jak to już przedstawiono w programach w poprzednim rozdziale, operacja pochłaniająca czas zawiera mnożenie dwóch liczb i dodawanie wyniku do akumulatora. Inne części algorytmu, takie jak indeksacja matryc, są bardzo szybkie. Mnożenie – akumulacja jest podstawowym składowym blokiem DSP, który – jak zobaczymy – powtarza się w kilku innych ważnych algorytmach. Szybkość komputerów DSP jest często *określana* właśnie przez czas wykonywania operacji mnożenie-akumulacja.

Jeśli sygnał złożony z N próbek jest poddawany operacji splotu z sygnałem złożonym z M próbek, to trzeba wykonać $N \times M$ operacji mnożenia-akumulacji. Można to stwierdzić na podstawie programu w poprzednim rozdziale. Komputery osobiste połowy lat 90. wymagały ok. 1 mikrosekundy do wykonania operacji mnożenia-akumulacji (100 MHz Pentium z arytmetyką zmiennoprzecinkową o pojedynczej dokładności, patrz tablica 4.6)[11].

Tak więc wykonanie operacji splotu sygnału złożonego z 10 000 próbek z sygnałem 100-próbkowym wymagało ok. 1 sekundy. Przetworzenie sygnału o milionie próbek przy użyciu 3000-próbkowej odpowiedzi impulsowej wymagało prawie godziny. Dekadę wcześniej (80286, 12 MHz) na to obliczenie trzeba było trzech dni!

Problem zbyt długich czasów wykonania programów jest zwykle rozwiązywany jednym z trzech sposobów. Po pierwsze, trzeba się starać, aby długość sygnałów była jak najmniejsza i należy stosować liczby całkowite zamiast arytmetyki zmiennoprzecinkowej. Jeśli splot ma być obliczany tylko kilka razy, to ten sposób będzie przypuszczalnie najlepszym kompromisem między czasem wykonania i wysiłkiem włożonym w programowanie. Po drugie, można użyć komputera przeznaczonego specjalnie do DSP. Są dostępne mikroprocesory DSP o czasach operacji mnożenie-akumulacja równych kilkadziesiąt nanosekund[12]. Jest to dobra droga, jeśli przewiduje się wykonywanie operacji splotu wiele razy, tak jak przy projektowaniu wyrobów komercyjnych.

Trzecim rozwiązaniem jest zastosowanie lepszego algorytmu do realizacji splotu. W rozdziale 17 opisano bardzo wysublimowany algorytm zwany *splotem FFT*. Ten algorytm daje dokładnie takie same wyniki, jak algorytmy splotu zaprezentowane w poprzednim rozdziale;

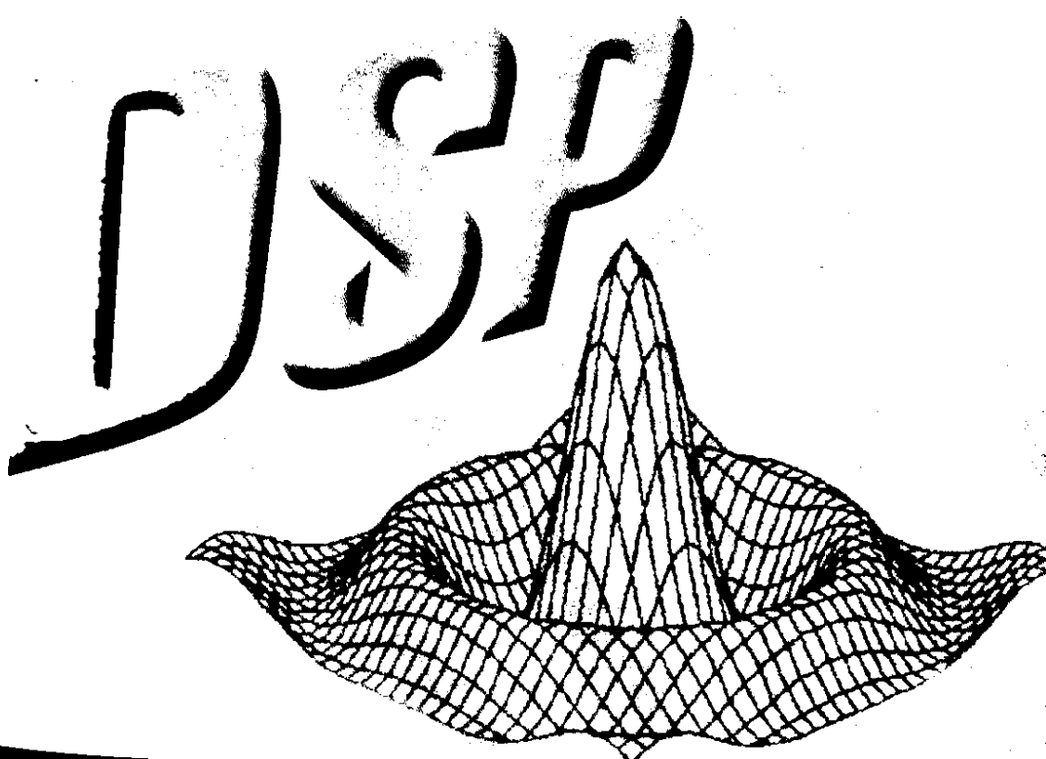
[11] Współczesne (2006 rok) komputery biurowe wykonują takie obliczenia (w zależności od zastosowanego oprogramowania) od 8 do 20 razy szybciej (dane za: *http://www.intel.com*) (przyp. tłum.).

[12] Jeden z najszybszych obecnie (początek 2006 roku) procesorów DSP, produkowany przez Texas Instruments TMS320C6455-1000, wykonuje do 8 milionów operacji mnożenie-akumulacja (operacji MAC) w ciągu sekundy, czyli jedna taka operacja trwa 0,125 ns (dane za: *http://www.ti.com*) (przyp. tłum.).

jednak czas wykonania jest znacznie skrócony. Dla sygnałów o tysiącach próbek splot FFT może być setki razy szybszy. Wadą tego programu jest jego złożoność. Nawet jeśli jesteście zaznajomieni z tą metodą, to musicie być przygotowani, że uruchomienie programu zajmie wam kilka godzin.

8

Dyskretne przekształcenie Fouriera



Analiza fourierowska to rodzina metod matematycznych, których podstawą jest rozkład sygnałów na sinusoidy. Do tej rodziny należy też dyskretne przekształcenie Fouriera (DFT – *Discrete Fourier Transform*), w którym operuje się sygnałami *przetworzonymi na cyfrowe*. Ten rozdział jest pierwszym z czterech poświęconych **rzeczywistemu DFT**, czyli wersji dyskretnego przekształcenia Fouriera, w której do reprezentowania sygnałów wejściowych i wyjściowych są stosowane liczby rzeczywiste. **Zespolone DFT**, metoda bardziej zaawansowana, w której stosuje się liczby zespolone, będzie omówiona w rozdziale 31. W tym rozdziale rozpatrzymy zależności matematyczne i algorytmy rozkładu Fouriera, który jest sercem DFT.

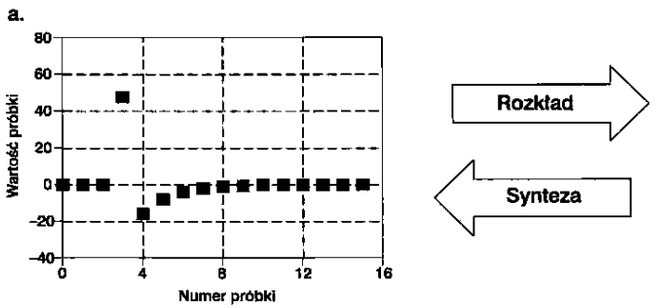
Rodzina przekształceń Fouriera

Analiza Fouriera wzięła nazwę od swego twórcy, którym był **Jean Baptiste Joseph Fourier** (1768–1830), francuski matematyk i fizyk. Chociaż wielu innych uczonych wniosło znaczący wkład w tę dziedzinę, to nazwą przekształcenia uhonorowano Fouriera za jego odkrycia matematyczne i wnikliwe spojrzenie na praktyczną przydatność metody. Fourier interesował się propagacją ciepła i przedstawił w roku 1807, w Institut de France, artykuł o zastosowaniu sinusoid do wyrażania rozkładów temperatury. Artykuł zawierał kontrowersyjną tezę, że każdy ciągły przebieg okresowy może być wyrażony jako suma odpowiednio dobranych fal sinusoidalnych. Wśród recenzentów było dwóch spośród najsławniejszych w historii matematyków – Joseph Louis Lagrange (1736–1813) i Pierre Simon de Laplace (1749–1827).

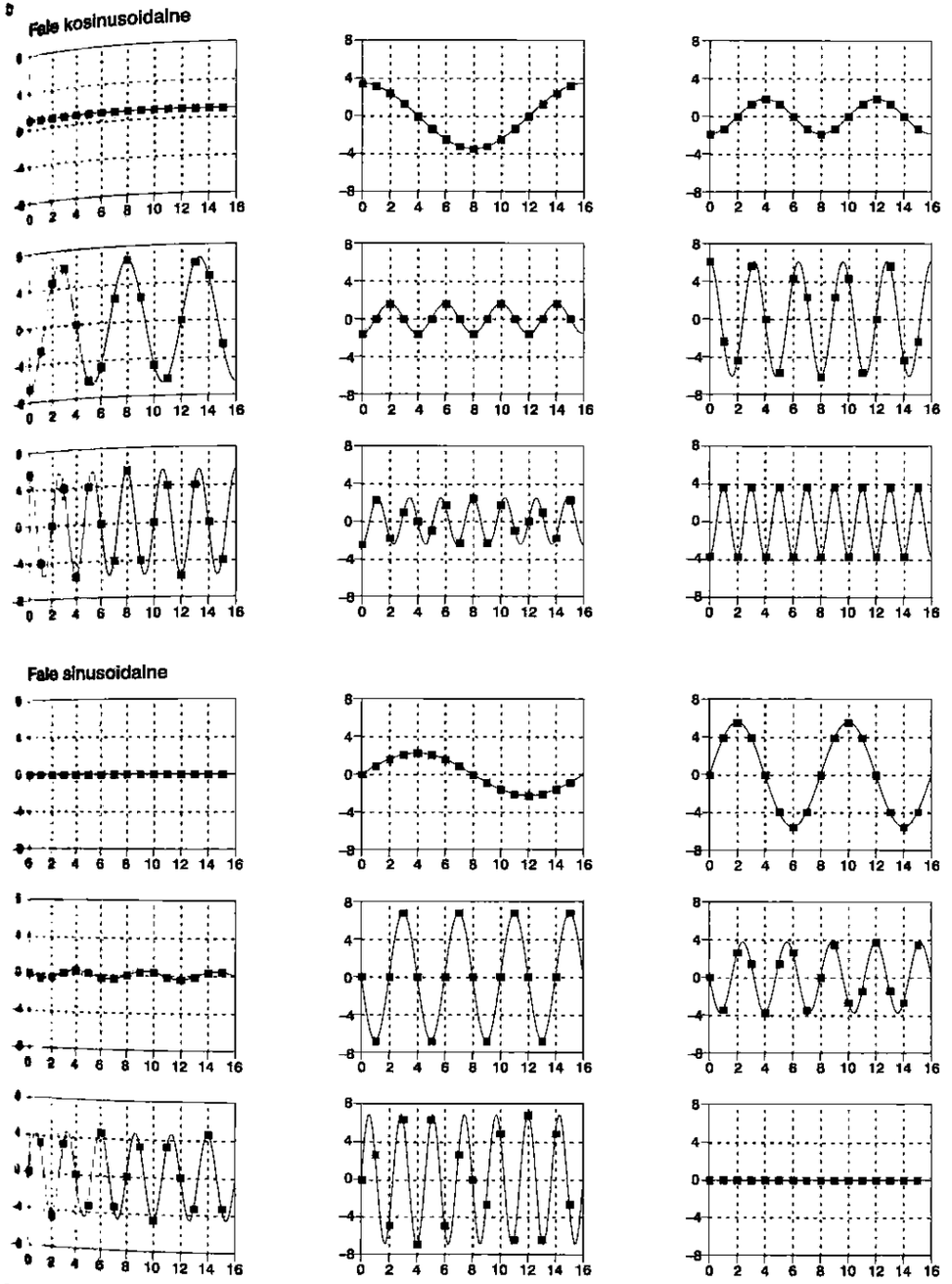
Podczas gdy Laplace i dwóch innych recenzentów głosowali za przyjęciem artykułu, Lagrange nieugięcie protestował. Przez prawie 50 lat Lagrange uparcie twierdził, że nie można stosować takiego podejścia do wyrażania sygnałów mających ostre *naroża*, np. nieciągłe zbocza, jak w fali prostokątnej. Institut de France uległ prestiżowi Lagrange'a i odrzucił artykuł Fouriera. Dopiero po śmierci Lagrange'a, 15 lat później, opublikowano wreszcie pracę Fouriera. Na szczęście Fouriera absorbowało także wiele innych zajęć, takich jak działalność polityczna, wyprawa do Egiptu z Napoleonem i próby uniknięcia gilotyny po rewolucji francuskiej (dosłownie!).

Kto miał rację? Odpowiedź nie jest całkiem prosta. Lagrange miał słuszność, twierdząc, że sumowanie sinusoid nie może utworzyć sygnału mającego ostre naroża. Jednak można uzyskać sygnał *bardzo* do niego zbliżony. Tak bardzo zbliżony, że różnica między obu sygnałami ma *zerową energię*. W tym sensie Fourier miał rację, chociaż w nauce XVIII wieku niewiele jeszcze wiedziano o pojęciu energii. Wspomniane zjawisko nosi teraz nazwę *efektu Gibbsa*, który będzie omówiony w rozdziale 11.

Na rys. 8.1 przedstawiono, jak można rozłożyć sygnał na przebiegi sinusoidalne i kosinusoidalne. Na rys. 8.1 a) mamy przykład sygnału o długości 16 punktów, zawierającego próbki o numerach od 0 do 15. Na rys. 8.1 b) pokazano rozkład Fouriera tego sygnału, 9 fal sinusoidalnych i 9 fal kosinusoidalnych, każda o innej częstotliwości i amplitudzie. Chociaż nie jest



Rys. 8.1. Przykład rozkładu Fouriera. Sygnał 8-punktowy (rys. a) jest rozłożony na 9 przebiegach sinusoidalnych i 9 kosinusoidalnych (rys. b). Częstotliwość każdego przebiegu jest ustalona, tylko amplituda się zmienia w zależności od kształtu przebiegu podlegającego rozkładowi

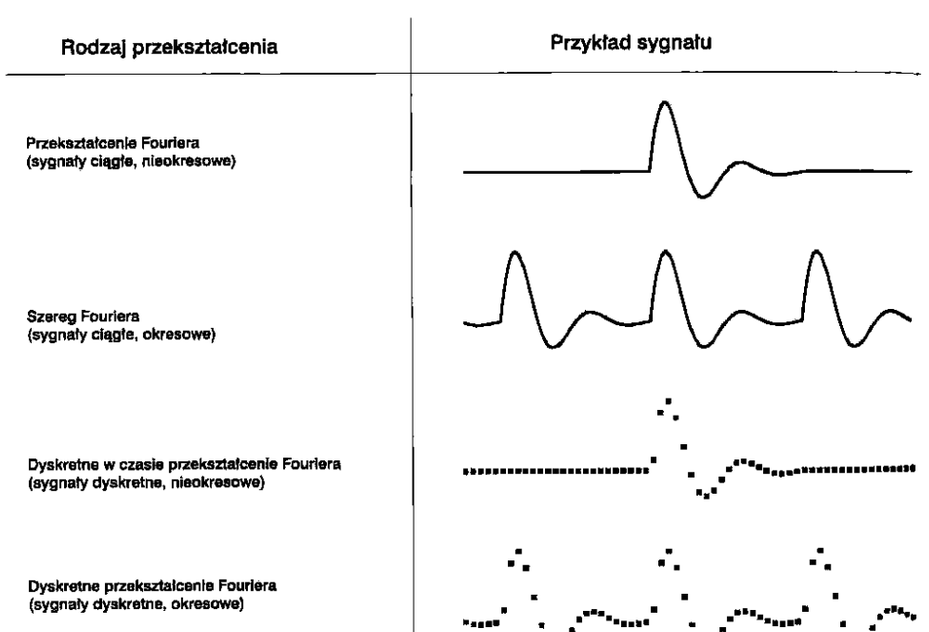


Rys. 8.1. (cd.)

to oczywiste, to jednak zsumowanie tych 18 przebiegów daje przebieg z rys. 8.1 a). Tr... podkreślić, że zastrzeżenie wysunięte przez Lagrange'a dotyczy tylko sygnałów *ciągłyc*... Dla sygnałów *dyskretnych* ten rozkład jest ścisły matematycznie. Nie ma żadnej różnicy mię... dzy sygnałem na rys. 8.1 a) i *sumą* sygnałów z rys. 8.1 b), tak jak nie ma żadnej różnicy ... między liczbą 7 i sumą 3 + 4.

Dlaczego stosuje się sinusoidy, a nie np. falę prostokątną lub trójkątną? Pamiętajmy, ... jest nieskończenie wiele sposobów rozkładu sygnału. Celem dokonania rozkładu sygna... łu jest uzyskanie przebiegu, którym można operować *łatwiej* niż przebiegiem oryginal... nym. Rozkład impulsowy, na przykład, umożliwia analizę sygnałów punkt po punkcie, ... co prowadzi do bardzo skutecznej metody splotowej. Sygnały składowe, sinusoidalne ... i kosinusoidalne, są prostsze niż sygnał oryginalny, gdyż mają pewną właściwość, ... której nie ma sygnał oryginalny – *wierność odtwarzania sinusoidy*. Jak to już rozważano ... w rozdziale 5, sygnał sinusoidalny na wejściu systemu da na wyjściu też z pewnością ... sygnał sinusoidalny. Amplituda i faza sygnału mogą ulec zmianie, ale kształt przebiegu ... musi pozostać taki sam. Sinusoida jest jedynym przebiegiem mającym tę pożyteczną ... właściwość. Rozkłady na przebiegi prostokątne lub trójkątne są *możliwe*, ale nie mają ... żadnego istotnego uzasadnienia.

W ogólnym pojęciu *przekształcenie Fouriera* można wyróżnić cztery kategorie wynikające ... z czterech podstawowych rodzajów sygnału. Sygnał może być zarówno *ciągły*, jak *dyskret... ny*; może być *okresowy* lub *nieokresowy*. Różne kombinacje tych dwóch cech tworzą cztery ... kategorie opisane poniżej i przedstawione na rys. 8.2



Rys. 8.2. Przykłady czterech rodzajów przekształcenia Fouriera. Sygnał może być ciągły lub dyskretny, okresowy lub nieokresowy. Z tego wynikają cztery możliwe kombinacje, z których każda daje inną wersję przekształcenia Fouriera. Nazwy nie są zbyt precyzyjne, trzeba je po prostu zapamiętać

Ciągłe przebiegi nieokresowe

Ta kategoria obejmuje takie sygnały, jak np. zanikający przebieg wykładniczy i krzywą Gaussa. Te sygnały rozciągają się zarówno do plus, jak i do minus nieskończoności *bez* powtarzania w formie sygnału okresowego. Przekształcenie fourierowskie dla tego rodzaju sygnałów nosi nazwę po prostu **przekształcenia Fouriera.**

Ciągłe przebiegi okresowe

Oto przykłady przebiegów tego rodzaju: fala sinusoidalna, fala prostokątna lub inny przebieg powtarzający się okresowo w zakresie od minus do plus nieskończoności. Tę wersję przekształcenia Fouriera nazywamy **szeregiem Fouriera.**

Dyskretne przebiegi nieokresowe

Te sygnały są określone tylko jako punkty dyskretne między minus i plus nieskończonością i nie powtarzają się w sposób okresowy. Ten rodzaj przekształcenia Fouriera nosi nazwę **dyskretnego w czasie przekształcenia Fouriera.**

Dyskretne przebiegi okresowe

Są to sygnały dyskretne, które powtarzają się w sposób okresowy w zakresie od minus do plus nieskończoności. Przekształcenie Fouriera tego rodzaju jest czasem nazywane dyskretnym szeregiem Fouriera, lecz najczęściej **dyskretnym przekształceniem Fouriera.**

Możecie pomyśleć, że nazwy nadane tym czterem rodzajom przekształcenia Fouriera są mylące i niezbyt precyzyjne. Macie rację, te nazwy powstawały w sposób raczej przypadkowy w ciągu 200 lat. Nie pozostaje wam nic innego, jak tylko nauczyć się ich na pamięć i zająć się dalszymi tematami.

Wszystkie cztery wymienione kategorie sygnałów obejmują zakres od minus do plus *nieskończoności*. Ale zaraz, zaraz, powiecie. A co będzie, jeśli mamy tylko skończoną liczbę próbek zgromadzonych w naszym komputerze, na przykład sygnał zawierający 1024 punktów. Czy nie ma jakiejś wersji przekształcenia Fouriera dla sygnałów o skończonej długości? Nie, nie ma. Fale sinusoidalna i kosinusoidalna są *zdefiniowane* jako trwające od minus nieskończoności do plus nieskończoności. Nie można zastosować grupy sygnałów nieskończenie długich do syntezy sygnału o skończonej długości. Sposobem ominięcia tego dylematu jest spowodowanie, aby dane skończone *wyglądały* jak sygnał o nieskończonej długości. W tym celu trzeba sobie wyobrazić, że sygnał ma nieskończoną liczbę próbek na lewo i na prawo od próbek rzeczywistych. Jeśli wszystkie te „wyobrażone" próbki mają wartość zero, to sygnał wygląda jak *dyskretny* i *nieokresowy* i można do niego zastosować dyskretne w czasie przekształcenie Fouriera. Rozwiązaniem alternatywnym jest uznanie, że wyobrażone próbki są duplikatem rzeczywistych 1024 punktów. W tym przypadku sygnał ma wygląd dyskretnego okresowego, o okresie 1024 próbek. Można więc skorzystać z dyskretnego przekształcenia Fouriera.

Jak się okazuje, do syntezy sygnału *nieokresowego* jest konieczna *nieskończona* liczba sinusoid. To powoduje, że obliczenie dyskretnej w czasie transformaty Fouriera jest niemożliwe przy użyciu algorytmu komputerowego. Drogą eliminacji dochodzimy więc do wniosku, że w DSP jedynym możliwym rodzajem przekształcenia Fouriera jest DFT. Inaczej mówiąc, komputery cyfrowe mogą pracować tylko z informacjami, które są *dyskretne* i mają *skończoną* długość. Walcząc z zagadnieniami teoretycznymi, zmagając się z trudnościami pracy domowej i rozszyfrowując tajniki matematyki, możecie korzystać z pierwszych trzech spośród poprzednio wymienionych przekształceń Fouriera. Ale gdy zasiądziecie do komputera, będziecie stosować tylko DFT. Na razie skoncentrujemy więc uwagę na dyskretnym przekształceniu Fouriera; inne omówimy krótko w dalszych rozdziałach.

Wróćmy do przykładu rozkładu DFT przedstawionego na rys. 8.1. Mamy tam 16-punktowy sygnał rozłożony na 18 sinusoid i kosinusoid, z których każda zawiera 16 punktów.

Podchodząc do problemu bardziej formalnie, trzeba rozpatrywać 16-punktowy sygnał z rys. 8.1 a) jako pojedynczy okres nieskończenie długiego sygnału okresowego. Podobnie każda z 18 sinusoid pokazanych na rys. 8.1 b) reprezentuje 16-punktowy segment nieskończenie długiej sinusoidy. Możemy rozpatrywać ten przykład jako syntezę 16-punktowego sygnału z 16-punktowych sinusoid lub jako syntezę nieskończenie długiego sygnału okresowego z nieskończenie długich sinusoid. Czy jest istotna różnica w tych dwóch podejściach? Odpowiedź brzmi: *zwykle nie, lecz czasem – tak*. W następnych rozdziałach spotkacie się z właściwościami DFT, które mogą wprawić w zakłopotanie, jeśli sygnały traktujemy jako skończone, lecz wydają się oczywiste, jeśli uwzględnimy okresowy charakter sygnałów. Najważniejszą sprawą jest zrozumienie, że okresowość jest wprowadzona w celu zastosowania *narzędzia matematycznego*, czyli DFT. Zwykle nie ma to żadnego związku z pochodzeniem sygnału ani ze sposobem jego uzyskania.

W każdym z czterech przekształceń Fouriera można wyróżnić wersję **rzeczywistą** i **zespoloną**. Najprostsza jest wersja rzeczywista, w której do syntezy i rozkładu stosuje się zwykłe liczby i algebrę. Przykładem **rzeczywistej DFT** jest przekształcenie przedstawione na rys. 8.1. Wersje zespolone wszystkich czterech przekształceń Fouriera są znacznie bardziej skomplikowane i wymagają zastosowania *liczb zespolonych*. Są to liczby takie, jak np. $3 + 4j$, gdzie j jest równe $\sqrt{-1}$ (elektrycy i elektronicy stosują oznaczenie j, matematycy zaś i). Skomplikowana matematyka może szybko przytłoczyć nawet tych, którzy specjalizują się w DSP. W istocie, głównym celem tej książki jest przedstawienie podstaw DSP *bez* stosowania zawiłej matematyki, dzięki czemu prezentowany materiał może być zrozumiały dla szerszego kręgu naukowców i inżynierów. Zespolone przekształcenia Fouriera są dziedziną dla specjalistów w zakresie DSP, którzy chcą się po szyję zanurzyć w grząskim gruncie matematyki. Jeśli macie takie skłonności, to doprowadzi was tam materiał zawarty w rozdziałach 30–33.

Termin matematyczny **przekształcenie** (*transform*)[13] jest szeroko używany w cyfrowym przetwarzaniu sygnałów. Są to m.in. przekształcenia Laplace'a, Z, Hilberta, dyskretne cosinusowe. Czym jest przekształcenie? Aby odpowiedzieć na to pytanie, przypomnijmy sobie, czym jest *funkcja*. Funkcja jest algorytmem lub procedurą zmieniającą jedną wartość w drugą. Na przykład $y = 2x + 1$ jest funkcją. Wybieramy jakąś wartość x, wstawiamy ją do równania i uzyskujemy wartość y. Funkcje mogą też zmieniać *kilka* wartości w inną wartość, np. $y = 2a + 3b + 4c$, gdzie a, b, c są zmieniane w pewną wartość y.

Przekształcenia, których wyniki są też nazywane *transformatami*, są bezpośrednim rozszerzeniem tego, co realizują funkcje. W tym przypadku jednak zarówno na wejściu, jak i na wyjściu może być *wiele* wartości. Załóżmy, że macie sygnał złożony ze 100 próbek. Jeśli wymyślicie jakieś równanie, algorytm lub procedurę powodujące zmianę tych 100 próbek w inne 100 próbek, to wymyśliliście nowe przekształcenie. Jeśli uważacie, że jest ono wystarczająco użyteczne, to macie pełne prawo dopisać do niego swoje nazwisko i zademonstrować kolegom zalety waszego przekształcenia. (Dałoby to najlepszy skutek, gdybyście byli pewnym wybitnym, XVIII-wiecznym matematykiem francuskim). Przekształcenia nie są ograniczone do jakiegoś specyficznego rodzaju liczb lub danych. Na przykład można mieć 100 próbek danych dyskretnych na wejściu i 200 próbek danych dyskretnych na wyjściu. Można także mieć sygnał ciągły na wejściu i ciągły na wyjściu. Dopuszczalne są sygnały mieszane – dyskretne na wejściu i ciągłe na wyjściu lub vice versa. W skrócie można stwierdzić, że przekształcenie jest ustaloną procedurą zmieniającą jedną porcję danych w inną taką porcję. Zobaczmy, jakie odniesienie ma to stwierdzenie do tematu, którym właśnie się zajmujemy – a więc do dyskretnego przekształcenia Fouriera.

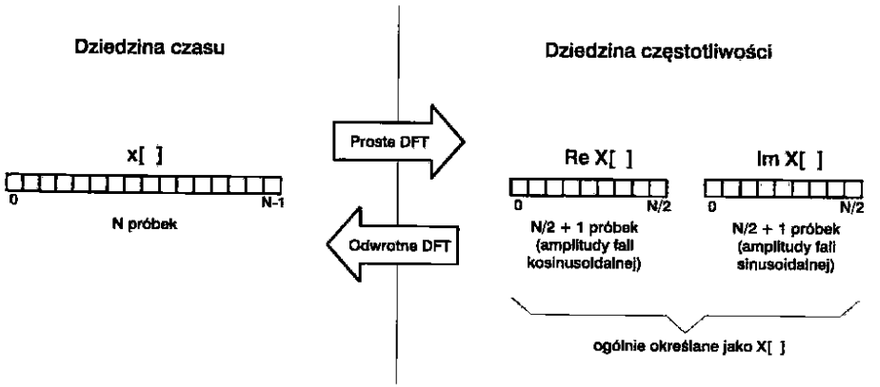
[13] W terminologii angielskiej słowo *transform* oznacza zarówno przekształcenie (np. Fouriera), jak i jego wynik, natomiast w języku polskim rozróżnia się te dwa pojęcia, nazywając transformatą wynik przekształcenia (przyp. tłum.).

Zapis i format rzeczywistego DFT

Dyskretne przekształcenie Fouriera (DFT), jak to przedstawiono na rys. 8.3, zmienia N-punktowy sygnał wejściowy w dwa $(N/2 + 1)$-punktowe sygnały wyjściowe. Sygnał wejściowy ma być poddany rozkładowi, a dwa sygnały wyjściowe zawierają *amplitudy* składowych sinusoidalnej i kosinusoidalnej (wyskalowanej w sposób, który później krótko omówimy). Mówimy, że sygnał wejściowy jest sygnałem **w dziedzinie czasu**. Jest tak, gdyż na wejściu DSP najczęściej spotykane są sygnały złożone z próbek pobieranych w regularnych odstępach *czasu*. Oczywiście na wejście systemu DSP można wprowadzić dane każdego rodzaju, niezależnie od tego, jak były one pozyskane. Gdy w analizie Fouriera spotykacie pojęcie „dziedzina czasu", to może ono odnosić się do próbek pobieranych w odstępach czasu lub może być ogólnym odniesieniem do każdego sygnału dyskretnego poddawanemu rozkładowi. Pojęcie **dziedzina częstotliwości** stosuje się do opisu amplitud fal sinusoidalnej i kosinusoidalnej (łącznie ze specjalnym skalowaniem, które – zgodnie z obietnicą – omówimy dalej).

W dziedzinie częstotliwości jest zawarta informacja dokładnie taka sama, jak w dziedzinie czasu, lecz w innej formie. Znając dane w jednej dziedzinie, można obliczyć je w drugiej. Jeśli jest dany sygnał w dziedzinie czasu, to proces obliczania sygnału w dziedzinie częstotliwości jest nazywany **prostym rozkładem, analizą** lub **przekształceniem DFT**, albo po prostu **DFT**. Jeśli zaś znamy dane w dziedzinie częstotliwości, to obliczenia w dziedzinie czasu są nazywane **syntezą** lub **odwrotnym DFT**. Zarówno synteza, jak i analiza mogą być wyrażane w postaci wzorów i algorytmów komputerowych.

Liczba próbek w dziedzinie czasu jest zwykle reprezentowana przez **zmienną** N. Chociaż N może być dowolną liczbą dodatnią, to zwykle wybiera się potęgę liczby 2, tj. 128, 256, 512, 1024 itd. Są dwie przyczyny takiego wyboru. Po pierwsze, w cyfrowych pamięciach danych stosuje się adresowanie dwójkowe, co czyni potęgi liczby dwa naturalną długością sygnału. Po drugie, w najbardziej efektywnym algorytmie do obliczeń DFT, szybkim przekształceniu Fouriera (FFT), zwykle stosuje się N będące potęgą dwóch. Typowo wybiera się N między 32 i 4096. W większości przypadków próbki mają numerację raczej od 0 do $N - 1$, a nie od 1 do N.



Rys. 8.3. Terminologia DFT. W dziedzinie czasu sygnał x[ ] zawiera N punktów o numeracji od 0 do N – 1. W dziedzinie częstotliwości DFT wytwarza dwa sygnały – część rzeczywistą zapisywaną jako Re X[ ] oraz część urojoną zapisywaną jako Im X[ ]. Każdy z tych sygnałów w dziedzinie częstotliwości ma długość N/2 +1 punktów, od 0 do N/2. Proste przekształcenie DFT zamienia sygnał w dziedzinie czasu na sygnał w dziedzinie częstotliwości, a odwrotne – z dziedziny częstotliwości do dziedziny czasu. (Uwaga: ten rysunek obrazuje rzeczywiste DFT. Zespolone DFT, które jest omówione w rozdziale 31, zamienia N punktów zespolonych w inny zestaw N punktów zespolonych)

W standardowym zapisie DSP używa się **małych liter** do wyrażania sygnałów w dziedzinie czasu, takich jak x[ ], y[ ] i z[ ]. Odpowiadające im **duże litery** są stosowane do wyrażania tych sygnałów w dziedzinie częstotliwości, a więc X[ ], Y[ ] i Z[ ]. Jako ilustrację tej zasady przyjmujmy, że N-punktowy sygnał w dziedzinie czasu jest zawarty w x[ ]. Ten sygnał w dziedzinie częstotliwości jest nazwany X[ ] i składa się z dwóch części, z których każda zawiera zestaw N/2 + 1 próbek. Jest to **część rzeczywista** X[ ], zapisywana jako *Re X[ ]* i **część urojona** X[ ], zapisywana jako *Im X[ ]*. Część *Re* [X] wyraża wartości fali kosinusoidalnej, a *Im* [X] – sinusoidalnej (pomijamy na razie kwestię czynników skalujących). Sygnał w dziedzinie czasu obejmuje zakres od x[0] do x[N – 1], a w dziedzinie częstotliwości – od *Re X*[0] do *Re X*[N/2] oraz od *Im X*[0] do *Im X*[N/2]. Dokładnie przestudiujcie te zapisy, mają one bardzo istotne znaczenie dla zrozumienia wzorów w DSP. Niektóre języki komputerowe nie rozróżniają, niestety, małych i dużych liter, pozostawiając indywidualnym programistom kwestię oznaczenia zmiennych. W programach w tej książce zastosowano matrycę XX[ ] do przechowywania sygnału w dziedzinie czasu oraz matryce *REX*[ ] i *IMX*[ ] – do sygnału w dziedzinie częstotliwości.

Nazwy *część rzeczywista* i *część urojona* pochodzą z zespolonego przekształcenia DFT, gdzie są stosowane do rozróżnienia liczb *rzeczywistych* i *urojonych*. Żadne takie komplikacje nie są konieczne w rzeczywistym przekształceniu DFT. Zanim dojdziemy do rozdziału 31, przyjmijcie po prostu, że „część rzeczywista" wyraża *wartości fali kosinusoidalnej*, podczas gdy „część urojona" – *fali sinusoidalnej*. Niech te sugestywne nazwy was nie zmylą, wszystko tutaj jest wyrażane w liczbach rzeczywistych.

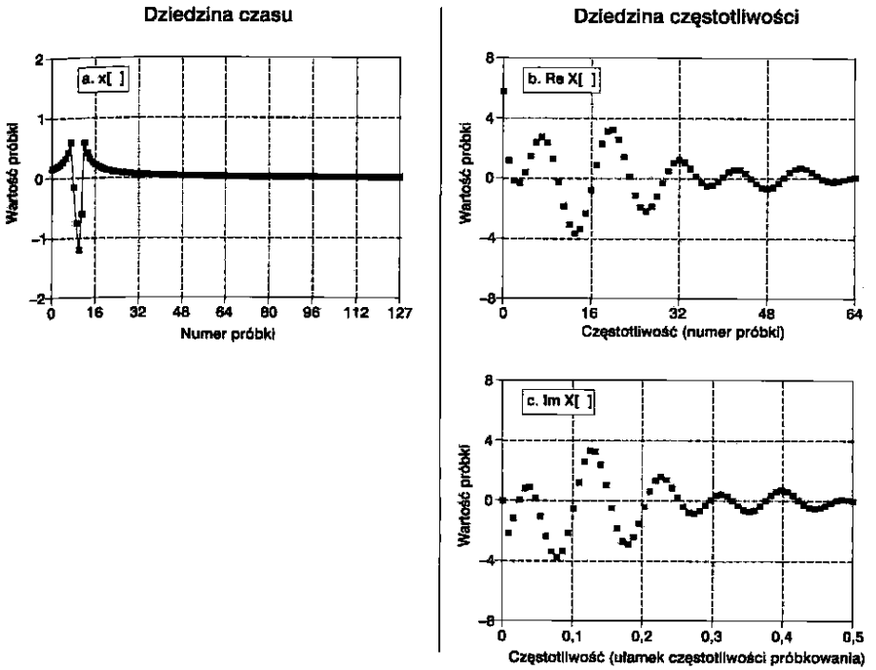
Nie dajcie się też zmylić *długością* sygnałów w dziedzinie częstotliwości. W publikacjach o DSP często spotyka się stwierdzenia, takie jak: „Przekształcenie DFT zamienia N-punktowy sygnał w dziedzinie czasu w N-punktowy sygnał w dziedzinie częstotliwości". To stwierdzenie nie odnosi się do *zespolonego przekształcenia* DFT, gdzie każdy „punkt" jest liczbą zespoloną (złożoną z części rzeczywistej i urojonej). Na razie skoncentrujmy uwagę na rzeczywistym przekształceniu DFT; trudna matematyka pojawi się później.

Zmienna niezależna w dziedzinie częstotliwości

Na rys. 8.4 przedstawiono przykład DFT z N = 128. Sygnał w dziedzinie czasu jest zawarty w matrycy x[0] do x[127]. Sygnały w dziedzinie częstotliwości umieszczono w dwóch matrycach: *Re X*[0] do *Re X*[64] i *Im X*[0] do *Im X*[64]. Zauważmy, że 128 punktów w dziedzinie czasu odpowiada 65 punktom w każdym z sygnałów w dziedzinie częstotliwości, indeksami zmieniającymi się od 0 do 64. Z tego wynika, że N punktom w dziedzinie czasu odpowiada N/2 + 1 punktów w dziedzinie częstotliwości (a nie N/2 punktów). Zapomnienie o tym dodatkowym jednym punkcie jest często spotykanym błędem w programach DFT.

Oś pozioma w dziedzinie częstotliwości może być wyskalowana jednym z **czterech różnych sposobów**; wszystkie są często spotykane w DSP. W pierwszej metodzie oś pozioma jest wyskalowana od 0 do 64, zgodnie z numeracją próbek w matrycach od 0 do N/2. Gdy stosuje się takie wyskalowanie, indeks w dziedzinie częstotliwości jest liczbą całkowitą, np. *Re X*[k] i *Im X*[k], gdzie k zmienia się od 0 do N/2 w krokach co jeden. Programiści lubią tę metodę, gdyż jest tu zgodność z pisaniem programu, gdzie stosują indeksy w celu adresowania matrycy. Takie wyskalowanie zastosowano na rys. 8.4 b).

W drugiej metodzie, stosowanej na rys. 8.4 c), oś pozioma jest wyskalowana w *jednostkach częstotliwości próbkowania*. To oznacza, że wartości wzdłuż osi poziomej zmieniają się od 0 do 0,5, gdyż dane dyskretne mogą zawierać częstotliwości tylko od 0 (prąd stały) do połowy częstotliwości próbkowania. Indeksem stosowanym w tym przypadku jest *f*, częstotliwość. Część rzeczywista i urojona są zapisywane jako *Re X*[f] i *Im X*[f], a f przyjmuje N/2 + 1 wartości rozmieszczonych w równych odstępach między 0 a 0,5. Przy przejść od poprzedniego zapisu (k) do drugiego (f), trzeba podzielić wartości skalowania



Rys. 8.4. Przykład przekształcenia DFT. To przekształcenie zamienia sygnał x[ ] w dziedzinie czasu na sygnały Re X[ ] i Im X[ ] w dziedzinie częstotliwości. Oś pozioma w dziedzinie częstotliwości może być wyskalowana jednym z trzech sposobów: 1) indeksem matrycy, od 0 do N/2, 2) ułamkiem częstotliwości próbkowania, od 0 do 0,5, 3) pulsacją, od 0 do π. W pokazanych przykładach zastosowano metodę pierwszą (rys. b) oraz drugą (rys. c)

poziomej przez N, a więc $f = k/N$. W tej książce w większości wykresów zastosowano tę drugą metodę, pamiętając, że sygnały dyskretne zawierają tylko częstotliwości z zakresu od 0 do 0,5 częstotliwości próbkowania.

Trzeci sposób jest podobny do drugiego, z tą wszakże różnicą, że wyskalowanie osi poziomej jest przemnożone przez 2π. Indeksem stosowanym przy tym skalowaniu jest ω. W tym przypadku części rzeczywistą i urojoną zapisuje się jako *Re X*[ω] i *Im X*[ω], gdzie ω przyjmuje N/2 + 1 wartości rozmieszczonych w równych odstępach między 0 i π. Wielkość ω jest nazywana **pulsacją**, a jej jednostką jest **radian**. W okręgu jest 2π radianów. Tę metodę lubią matematycy, gdyż daje ona skrócenie wzorów. Rozważmy na przykład, jak jest zapisywana fala kosinusoidalna przy trzech wymienionych metodach skalowania. Przy użyciu *k*: $c[n] = \cos(2\pi kn/N)$, przy użyciu *f*: $c[n] = \cos(2\pi fn)$, a stosując ω: $c[n] = \cos(\omega n)$.

W czwartej metodzie skaluje się oś poziomą według częstotliwości analogowej używanej w tym *szczególnym* zastosowaniu. Na przykład, jeśli badany system ma częstotliwość próbkowania 10 kHz (tzn. 10 000 próbek/s), to wykresy w dziedzinie częstotliwości będą wyskalowane od 0 do 5 kHz. Metoda ma tę zaletę, że częstotliwość jest prezentowana w kategoriach *świata rzeczywistego*. Wadą zaś jest powiązanie skalowania z pewną określoną częstotliwością próbkowania, co uniemożliwia stosowanie tej metody do opracowań ogólnych algorytmów DSP, takich jak np. projektowanie filtrów cyfrowych.

Wszystkie cztery wymienione sposoby zapisu są stosowane do skalowania w DSP i powinno się mieć łatwość konwersji pomiędzy nimi. To obejmuje zarówno wykresy, jak i wzory

matematyczne. W celu stwierdzenia, jakiego zapisu użyto, wystarczy spojrzeć na zm niezależną i jej zakres. Może to być jeden z czterech zapisów: k (lub jakaś inna liczba kowita) zmieniająca się od 0 do $N/2$; f zmieniające się od 0 do 0,5; ω zmieniająca się od 0 do π; lub częstotliwość wyrażona w hercach, zmieniająca od 0 do połowy zastosowanej częstotliwości próbkowania.

Funkcje bazowe DFT

Stosowane w dyskretnym przekształceniu Fouriera fale sinusoidalna i kosinusoidalna są z kle nazywane **funkcjami bazowymi** DFT. Można powiedzieć, że na wyjściu DFT uzy się zestaw liczb wyrażających amplitudy. Funkcje bazowe są zestawem fal sinusoidalnej i sinusoidalnej o amplitudzie *jednostkowej*. Jeśli przypisać każdą amplitudę (w dziedzinie stotliwości) do właściwej fali sinusoidalnej lub kosinusoidalnej (więc do funkcji bazowy to wynikiem jest zestaw *wyskalowanych* fal sinusoidalnych i kosinusoidalnych, które zsumować w celu utworzenia sygnału w dziedzinie czasu.

Funkcje bazowe DFT są opisane wzorami:

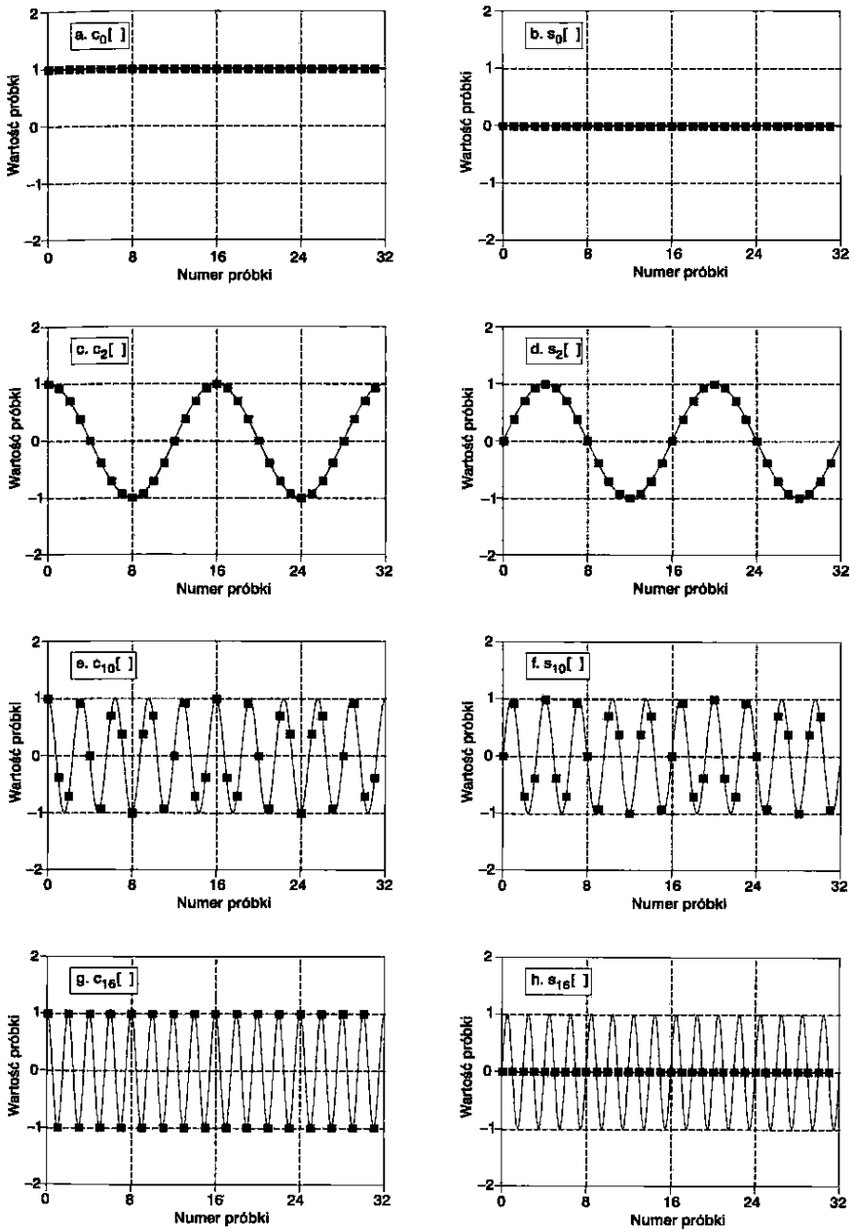
$$c_k[i]=\cos(2\pi ki/N)$$
$$s_k[i]=\sin(2\pi ki/N)$$

Wzór 8.1. Wzory wyrażające bazowe funkcje DFT. W tych wzorach $c_k[i]$ i $s_k[i]$ są falami sinusoidalną i kosinusoidalną, każda o długości N punktów od i = 0 do N – 1. Parametr k określa częstotliwość fali. W N-punktowym DFT współczynnik k przyjmuje wartości od 0 do N/2.

gdzie: $c_k[\ ]$ jest falą kosinusoidalną, której amplituda jest zawarta w $Re\ X[k]$, a $s_k[\ ]$ – fa nusoidalną o amplitudzie zawartej w $Im\ X[k]$. Na rys. 8.5, jako przykład, przedstawiono z 17 fal sinuoidalnych i 17 kosinusoidalnych stosowanych DFT o liczbie punktów $N=$ Ponieważ te przebiegi po zsumowaniu mają utworzyć sygnał wejściowy, więc muszą *długość* taką samą, jak sygnał wejściowy. W tym przypadku oba przebiegi mają 32 pu o numeracji od $i = 0$ do 31. Parametr k ustala częstotliwość każdej sinusoidy. W szczeg ści $c_1[\ ]$ jest falą kosinusoidalną, której *jeden* pełny okres mieści się w N punktach, $c_5[\ ]$ jest falą kosinusoidalną mającą *pięć* pełnych okresów itd. Jest to ważna koncepcja, konie do zrozumienia funkcji bazowych; parametr częstotliwościowy k jest równy liczbie pe okresów mieszczących się w N punktach sygnału.

Przyjrzyjmy się bardziej szczegółowo kilku z tych funkcji bazowych. Na rys. 8.5 pokazano falę kosinusoidalną $c_0[\ ]$. Jest to fala kosinusoidalna o częstotliwości a więc jest to stała wartość równa jedności. To znaczy, że $Re\ X[0]$ przechowuje śr wartość wszystkich punktów sygnału w dziedzinie czasu. W elektronice powiedzia się, że przechowuje wartość **przesunięcia napięcia stałego** (*DC offset*). Falę sinusoid $s_0[\ ]$ o częstotliwości zerowej pokazano na rys. 8.5 b). Jest to sygnał składający z samych *zer*. Ponieważ nie może to mieć żadnego wpływu na syntetyzowany sy w dziedzinie czasu, więc wartość $Im\ X[0]$ jest nieistotna i zawsze ustawiana na Wkrótce powiemy o tym więcej.

Na rysunkach 8.5 c), d) przedstawiono $c_2[\ ]$ i $s_2[\ ]$ – sinusoidy, których *dwa* pełne sy obejmują N punktów. Odpowiadają one $Re\ X[2]$ i $Im\ X[2]$. Podobnie na rysun 8.5 e), f) przedstawiono $c_{10}[\ ]$ i $s_{10}[\ ]$ – sinusoidy, których *dziesięć* pełnych okresów obejm N punktów. Te sinusoidy odpowiadają amplitudom przechowywanym w matrycach $Re\ X$ i $Im\ X[10]$. Problem jest taki, że próbki na rysunkach 8.5 e), f) *nie wyglądają* już jak si idy i kosinusoidy. Gdyby na tych rysunkach nie poprowadzono linii ciągłych, trudno b nawet określić kształt tych przebiegów. To może wydawać się nieco niepokojące. przejmujcie się tym. Z matematycznego punktu widzenia te próbki tworzą dyskretne s idy, nawet jeśli wzrokowo nie możemy prześledzić ich kształtu.



Rys. 8.5. Bazowe funkcje DFT. Przekształcenie DFT 32-punktowe ma jako funkcje bazowe 17 dyskretnych fal kosinusoidalnych i 17 dyskretnych fal sinusoidalnych. Osiem z tych przebiegów przedstawiono na rysunku. Są to sygnały dyskretne; linie ciągłe poprowadzono na tych rysunkach tylko po to, aby czytelnicy mogli łatwiej wzrokowo prześledzić przebiegi

Największe częstotliwości w funkcjach bazowych przedstawiono na rysunkach 8.5g, h. Są to $c_{N/2}[\,]$ i $s_{N/2}[\,]$, czyli w tym przykładzie $c_{16}[\,]$ i $s_{16}[\,]$. Dyskretna fala kosinusoidalna zmienia swoją wartość między 1 i –1, co można interpretować jako próbkowanie ciągłej sinusoidy w jej *wartościach szczytowych*. Sytuacja odwrotna jest z falą sinusoidalną, która zawiera same zera, co wynika z próbkowania przy *przejściach przez zero*. To powoduje, że wartość *Im X*[*N*/2] jest taka sama jak *Im X*[0], więc zawsze równa zeru i nie ma wpływu na syntezę sygnału w dziedzinie czasu.

A oto zagadka: jeśli mamy *N* próbek wchodzących do DFT i *N* + 2 wychodzących, to skąd się bierze ta dodatkowa informacja? Odpowiedź: dwie z próbek wyjściowych *nie zawierają* żadnej informacji, dzięki czemu pozostałe *N* próbek jest całkowicie niezależnych. Jak można było odgadnąć, punktami, które nie niosą żadnej informacji, są *Im X*[0] i *Im X*[*N*/2], a więc próbki mające zawsze wartość zero.

Synteza, obliczanie odwrotnej transformaty DFT

Zbierając razem wszystko, co dotychczas powiedziano, można napisać **równanie syntezy**:

$$x[i]=\sum_{k=0}^{N/2} Re\,\bar{X}[k]\cos(2\pi ki/N)+\sum_{k=0}^{N/2} Im\,\bar{X}[k]\sin(2\pi ki/N)$$

Wzór 8.2. Równanie syntezy. W tej zależności x[i] jest poddawanym syntezie sygnałem, którego indeks i zmienia się od 0 do N – 1. Amplitudy fal kosinusoidalnej i sinusoidalnej są przechowywane w Re X[k] i Im X[k], przy czym k zmienia się od 0 do N/2. We wzorze 8.3 jest dokonywana normalizacja w celu zamiany tego wzoru w odwrotne DFT

Wyraźmy to samo słownie: każdy sygnał *N*-punktowy *x*[*i*] może być utworzony przez zsumowanie *N*/2 + 1 fal kosinusoidalnych i *N*/2 + 1 fal sinusoidalnych. Amplitudy fal sinusoidalnych i kosinusoidlanych są przechowywane w matrycach, odpowiednio, $Im\,\bar{X}[k]$ i $Re\,\bar{X}[k]$.

W równaniu syntezy te amplitudy są mnożone przez funkcje bazowe, aby utworzyć zestaw wyskalowanych fal sinusoidalnych i kosinusoidalnych. Zsumowanie wyskalowanych fal sinusoidalnych i kosinusoidlanych tworzy sygnał w dziedzinie czasu *x*[*i*].

We wzorze 8.2 matryce są nazwane $Im\,\bar{X}[k]$ i $Re\,\bar{X}[k]$, a nie *Im X*[*k*] i *Re X*[*k*]. Przyczyną jest fakt, że *amplitudy wymagane do syntezy* (oznaczone w tych rozważaniach $Im\,\bar{X}[k]$ i $Re\,\bar{X}[k]$) są nieco inne niż *w dziedzinie częstotliwości sygnału* (oznaczane jako *Im X*[*k*] i *Re X*[*k*]). Chodzi tu o czynnik skalujący, o którym wcześniej mówiliśmy. Chociaż ta konwersja jest tylko prostą normalizacją, jest jednak przyczyną częstych pomyłek w programach komputerowych. Trzeba na to zwracać uwagę! W formie równań ta konwersja ma postać:

$$Re\,\bar{X}[k]=\frac{ReX[k]}{N/2}$$

$$Im\,\bar{X}[k]=-\frac{ImX[k]}{N/2}$$

oprócz dwóch przypadków szczególnych

$$Re\,\bar{X}[0]=\frac{ReX[0]}{N}$$

$$Re\,\bar{X}[N/2]=\frac{ReX[N/2]}{N}$$

Wzór 8.3. Konwersja między amplitudami sinusoid i wartościami w dziedzinie częstotliwości. W tych równaniach Re X[k] i Im X[k] przechowują amplitudy fal kosinusoidalnych i sinusoidalnych potrzebnych do syntezy, podczas gdy Im X[k] i Re X[k] – części rzeczywistą i urojoną w dziedzinie częstotliwości. Jak zwykle, N jest liczbą punktów w sygnale w dziedzinie czasu, k jest indeksem zmieniającym się od 0 do

Rys. 8.1.

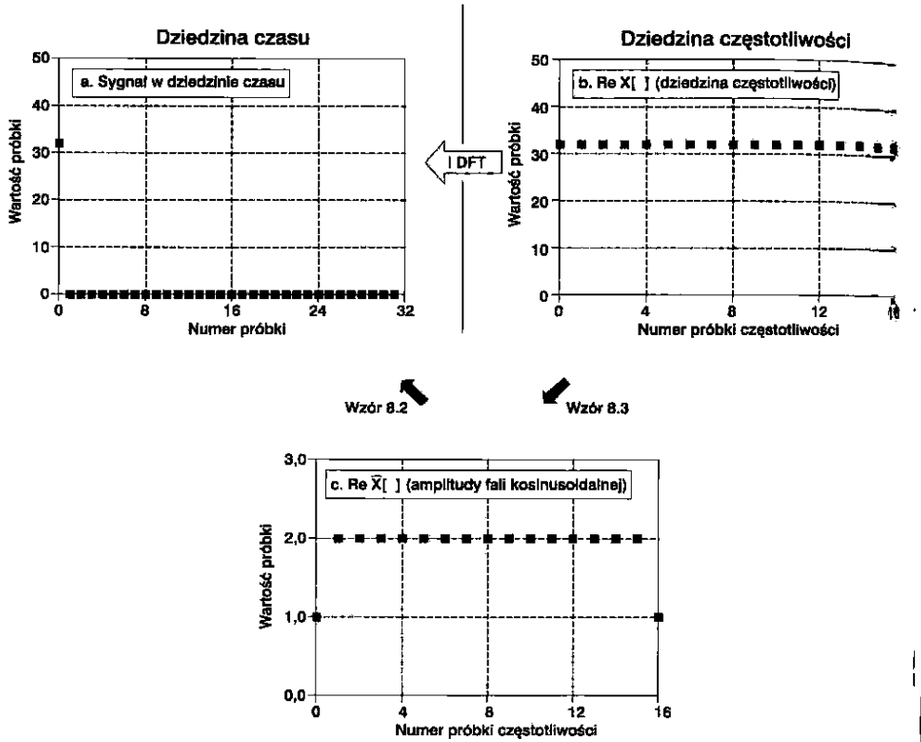
```
'ODWROTNE DYSKRETNE PRZEKSTAŁCENIE FOURIERA
'Sygnał w dziedzinie czasu, przechowywany w XX[ ], obliczany z sygnałów w dziedzinie częstotliwości
'przechowywanych w REX[ ] i IMX[ ],
'
DIM XX[511]             'XX[ ] przechowuje sygnał w dziedzinie czasu
DIM REX[256]            'REX[ ] przechowuje część rzeczywistą sygnału w dziedzinie częstotliwości
DIM IMX[256]            'IMX[ ] przechowuje część urojoną sygnału w dziedzinie częstotliwości
'
PI = 3.14159265         'Ustaw stałą PI
N% = 512                'N% jest liczbą punktów w XX[ ]
'
GOSUB XXXX              'Podprogram "mityczny" do ładowania danych do REX[ ] i IMX[ ]
'
'
'                       'Wyznacz amplitudy fali kosinusoidalnej i sinusoidalnej na podstawie wzoru 8.3
FOR K% = 0 TO 256
  REX[K%] = REX[K%] / (N%/2)
  IMX[K%] = -IMX[K%] / (N%/2)
NEXT K%
'
REX[0] = REX[0] / 2
REX[256] = REX[256] / 2
'
'
FOR I% = 0 TO 511       'Wyzeruj XX[ ], aby mógł być użyty jako akumulator
  XX[I%]=0
NEXT I%
'
'                       'Wzór 8.2 METODA SYNTEZY #1. Pętla przez każdą częstotliwość
'                       'generująca fale sinusoidalną i kosinusoidalną o całej długości
'                       'i dodawanie ich do sygnału w akumulatorze XX[ ]
'
FOR K% = 0 TO 256       'K% tworzy pętlę przez wszystkie próbki w REX[ ] i IMX[ ]
  FOR I% = 0 TO 511     'I% tworzy pętlę przez wszystkie próbki w XX[ ]
    '
    XX[I%] = XX[I%] + REX[K%] * COS(2*PI*K%*I%/N%)
    XX[I%] = XX[I%] + IMX[K%] * SIN(2*PI*K%*I%/N%)
    '
  NEXT I%
NEXT K%
'
END
```

Program alternatywny w wierszach od 380 do 510

```
'                       'WZÓR 8.2 METODA SYNTEZY #2. Pętla przez wszystkie próbki
'                       'w dziedzinie czasu i sumowanie odpowiednich próbek
'                       'z każdej fali kosinusoidalnej i sinusoidalnej
'
FOR I% = 0 TO 511       'I% tworzy pętlę przez wszystkie próbki w XX[ ]
  FOR K% = 0 TO 256     'K% tworzy pętlę przez wszystkie próbki w REX[ ] i IMX[ ]
    '
    XX[I%] = XX[I%] + REX[K%] * COS(2*PI*K%*I%/N%)
    XX[I%] = XX[I%] + IMX[K%] * SIN(2*PI*K%*I%/N%)
    '
  NEXT K%
NEXT I%
'
END
```

Załóżmy, że mamy daną reprezentację sygnału w dziedzinie częstotliwości i chcemy dok nać syntezy, aby uzyskać odpowiadający tej reprezentacji sygnał w dziedzinie czasu. Najp musimy znaleźć amplitudy fal sinusoidalnych i kosinusoidalnych. A więc mając *Im X*[*k*] *Re X*[*k*], powinniśmy wyznaczyć $Im\,\bar{X}[k]$ oraz $Re\,\bar{X}[k]$. W równaniach 8.3 przedstawi tę operację w postaci matematycznej. Aby wykonać to w formie programu komputerow trzeba podjąć trzy czynności. Po pierwsze, podzielić przez *N*/2 wszystkie wartości w dziedz częstotliwości. Po drugie, zmienić znak wszystkich wartości urojonych. Po trzecie, pod pierwszą i ostatnią próbkę w części rzeczywistej, *Re X*[0] i *Re X*[*N*/2], przez dwa. W ten spo otrzymuje się amplitudy potrzebne do przeprowadzenia syntezy opisanej wzorem 8.2. Łat oba wzory 8.2 i 8.3 *definiują* odwrotne dyskretne przekształcenie Fouriera (DFT).

Całkowite odwrotne DFT przedstawiono w programie komputerowym w tablicy 8.4 dwa sposoby zaprogramowania syntezy (zgodnie z wzorem 8.2) i oba są zaprezento W pierwszej metodzie wyskalowane sinusoidy są generowane jedna po drugiej i sumo w matrycy akumulacyjnej, dając w rezultacie sygnał w dziedzinie czasu. W drugiej metod próbki w dziedzinie czasu są obliczane jedna po drugiej, jako sumy odpowiednich pod w falach kosinuosidalnych i sinusoidalnych. Obie metody dają taki sam wynik. Różnica m dzy tymi dwoma programami jest niewielka; pętle wewnętrzna i zewnętrzna są zamien miejscami podczas syntezy.



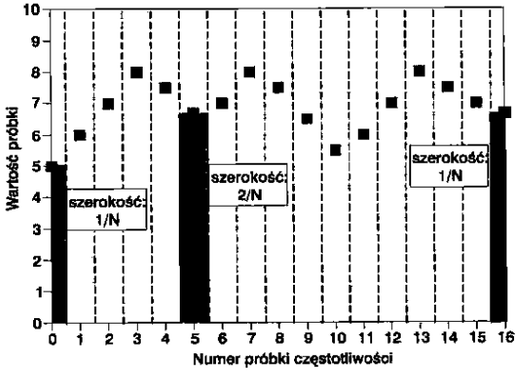
Rys. 8.6. Przykład odwrotnego, dyskretnego przekształcenia Fouriera. Na rys. a) przedstawiono przykład sygnału w dziedzinie czasu, jest to impuls w próbce zerowej o amplitudzie równej 32. Na rys. b) pokazano część rzeczywistą reprezentacji tego sygnału w dziedzinie częstotliwości, która ma stałą wartość 32. Część urojona (niepokazana) w dziedzinie częstotliwości zawiera same zera. Na rys. c) pokazano amplitudy fal kosinusoidalnych koniecznych do odtworzenia przebiegu z rys. a) przy użyciu wzoru 8.2. Wartości na rys. c) są wyznaczone na podstawie rys. b) przy użyciu wzoru 8.3

Na rys. 8.6 zilustrowano działanie odwrotnego DFT oraz drobne różnice między dziedziną częstotliwości i amplitudami koniecznymi do syntezy. Na rys. 8.6 a) przedstawiono przykład sygnału, który chcemy syntetyzować – impuls o amplitudzie 32 w próbce zerowej. Na rys. 8.6 b) pokazano reprezentację tego sygnału w dziedzinie częstotliwości. Część rzeczywista sygnału w dziedzinie częstotliwości ma stałą wartość równą 32. Część urojona (niepokazana) zawiera same zera. Jest to ważna *para* DFT – impuls w dziedzinie czasu odpowiada wartości stałej w dziedzinie częstotliwości. Tę kwestię przedyskutujemy w następnym rozdziale. Na razie ważne jest, że sygnał na rys. 8.6 b) jest przekształceniem DFT sygnału z rys. 8.6 a), przebieg zaś na rys. 8.6 a) jest *odwrotnym* przekształceniem DFT przebiegu z rys. 8.6 b).

Wzór 8.3 jest stosowany do przekształcenia sygnału w dziedzinie częstotliwości (rys. 8.6 b) na amplitudy fal kosinusoidalnych (rys. 8.6 c). Jak to pokazano na rysunku, wszystkie próbki mają wartość *dwa*, oprócz próbek 0 i 16 mających wartość *jeden*. W tym przykładzie nie pokazano amplitud fal sinusoidalnych, gdyż mają one wartość zero i są nieistotne. Równanie syntezy (wzór 8.2) jest następnie stosowane do przekształcenia amplitud fal kosinusoidalnych (rys. 8.6 c) na sygnał w dziedzinie czasu (rys. 8.6 a).

Powyżej opisano, *czym* sygnał w dziedzinie częstotliwości różni się od amplitud sinusoid, lecz nie powiedziano, *dlaczego* występuje ta różnica. Różnica powstaje, gdyż reprezentacja sygnału w dziedzinie częstotliwości jest określana jako **gęstość widmowa**. Na rys. 8.7 pokazano, na czym to polega. Na tym rysunku jako przykład wzięto część rzeczywistą reprezentacji sygnału 32-punktowego w dziedzinie częstotliwości. Jak trzeba się było spodziewać, są to próbki od 0 do 16, reprezentujące 17 częstotliwości rozmieszczonych w równych odstępach między 0 i 1/2 częstotliwości próbkowania. *Gęstość widmowa* opisuje, jak dużo sygnału (pod względem wartości) występuje *na jednostkę szerokości pasma*. Aby przetworzyć amplitudy sinusoid na gęstość widmową, należy podzielić każdą amplitudę przez reprezentowaną przez nią szerokość pasma. To prowadzi do następnej kwestii: jak określić szerokość pasma każdej częstotliwości dyskretnej w dziedzinie częstotliwości.

Szerokość pasma, jak to przedstawiono na rys. 8.7, można określić, rysując linie podziału między próbkami. Na przykład próbka numer 5 mieści się w paśmie między 4,5 i 5,5; próbka numer 6 w paśmie między 5,5 i 6,5 itd. Szerokość pasma każdej próbki, wyrażona jako ułamek pasma całkowitego (tzn. *N*/2), wynosi 2/*N*. Wyjątkiem są próbki skrajne, które mają połowę tego pasma, czyli 1/*N*. To można wyjaśnić współczynnikiem skalowania 2/*N* między



Rys. 8.7. Szerokość pasma próbek w dziedzinie częstotliwości. Każdą próbkę w dziedzinie częstotliwości można traktować jako zajmującą pasmo częstotliwości o szerokości 2/N, wyrażonej jako ułamek pasma całkowitego. Wyjątkiem od tej zasady są próbki skrajne, które mają szerokość pasma o połowę mniejszą, równą 1/N

amplitudami sinusoid i dziedziną częstotliwości, jak również dodatkowym czynnikiem równym 2 dla pierwszej i ostatniej próbki.

Skąd się bierze negacja części urojonej? To się robi wyłącznie w celu utrzymania zgodności *części rzeczywistej DFT* z „wielkim bratem", czyli z *zespolonym przekształceniem Fouriera* (*DFT*). W rozdziale 29 wykażemy, że ta negacja jest konieczna, aby prawidłowo zadziałały operacje matematyczne w zespolonym DFT. Wielu autorów, operując tylko z rzeczywistym DFT, w ogóle nie wprowadza tej negacji. Niektórzy autorzy nie wprowadzają nawet współczynnika skalowania $2/N$. Bądźcie więc przygotowani na to, że możecie spotkać takie pominięcia w niektórych analizach. Tutaj uwzględniamy te kwestie z niezwykle ważnego powodu; najskuteczniejszy sposób obliczania DFT prowadzi przez algorytm szybkiej transformaty Fouriera (FFT), przedstawiony w rozdziale 12. FFT tworzy transformatę w dziedzinie częstotliwości zgodnie z wzorami 8.2 i 8.3. Jeśli dopuścicie do bałaganu we współczynnikach skalowania, to wasze programy zawierające FFT nie zadziałają zgodnie z oczekiwaniami.

Analiza, obliczanie transformaty DFT

Transformatę DFT można obliczać trzema zupełnie różnymi metodami. Po pierwsze, można podejść do problemu jako do układu *równań równoważnych*. Ta metoda jest przydatna do zrozumienia DFT, lecz nie ma praktycznego znaczenia ze względu na zbyt małą efektywność. W drugiej metodzie korzysta się z pojęcia wprowadzonego w poprzednim rozdziale – z *korelacji*. Polega ona na wykrywaniu znanego przebiegu w innym sygnale. Trzecia metoda, zwana szybkim przekształceniem Fouriera (FFT), jest pomysłowym algorytmem, który rozkłada DFT o N punktach na N przekształceń DFT, każde o pojedynczym punkcie. Przekształcenie FFT jest, typowo, setki razy szybsze od innych metod. Pierwsze dwie metody będą tu omówione, a FFT będzie tematem rozdziału 12. Trzeba pamiętać, że w tych wszystkich trzech metodach uzyskuje się jednakowy wynik. Którą z nich powinno się stosować? W praktyce, metoda *korelacyjna* jest zalecana, jeśli DFT ma mniej niż ok. 32 punkty. W innych przypadkach stosuje się *FFT*.

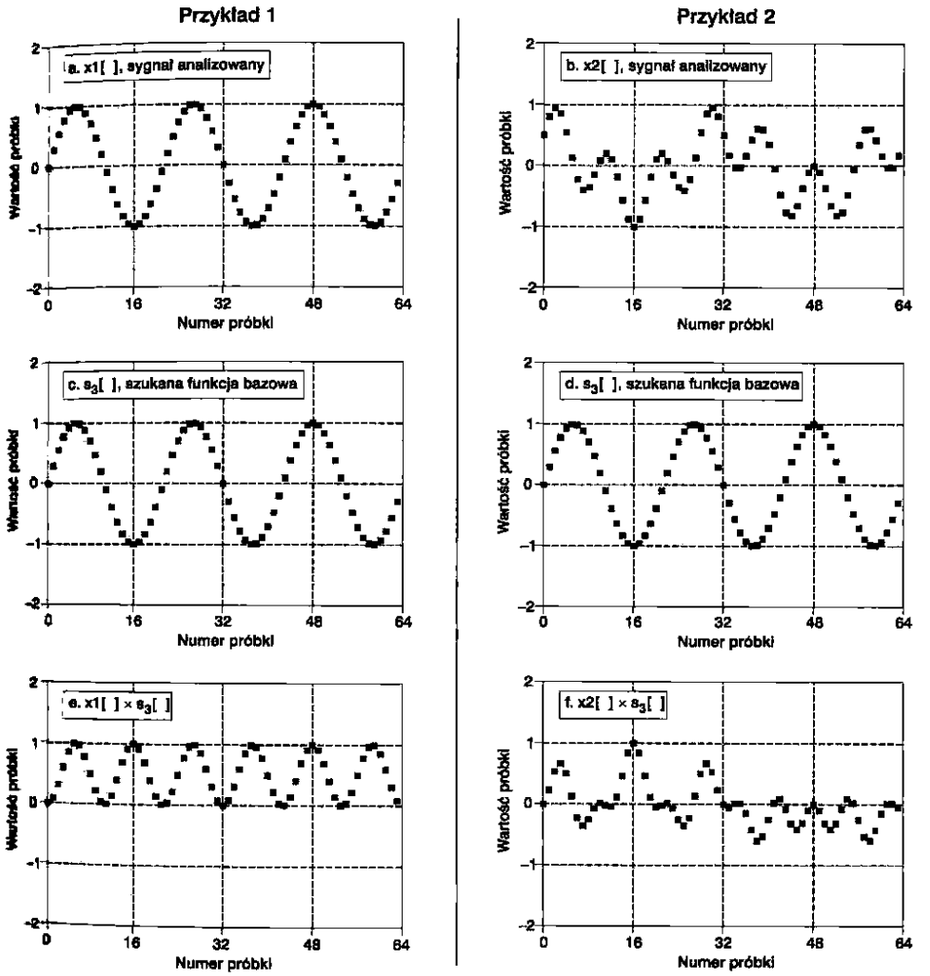
Metoda równań równoważnych

Podejdźmy do obliczania transformaty DFT w sposób następujący. Mamy dane N wartości w dziedzinie czasu i chcemy obliczyć N wartości w dziedzinie częstotliwości (pomijając dwie wartości z dziedziny częstotliwości, o których wiadomo, że muszą być zerowe). Pomocna tu będzie podstawowa algebra. Żeby znaleźć N niewiadomych, trzeba móc napisać N równań liniowo niezależnych. Aby to uczynić, weźmy pierwsze próbki z wszystkich sinusoid i zsumujmy je. Suma musi być równa pierwszej próbce w sygnale w dziedzinie czasu, uzyskujemy więc pierwsze równanie. Podobnie można zapisać równania dla każdego z pozostałych punktów w sygnale w dziedzinie czasu, uzyskując w wyniku żądane N równań. Rozwiązanie można uzyskać, stosując znane sposoby rozwiązywania równań równoważnych, takie, jak metody eliminacji Gaussa. Niestety ta metoda wymaga wielkiej liczby obliczeń i w zasadzie nie jest stosowana w DSP. Jest jednak ważna z innego powodu. Pokazuje bowiem, *dlaczego* jest możliwy rozkład sygnału na sinusoidy, jak *wiele* sinusoid jest potrzebnych oraz to, że funkcje bazowe muszą być liniowo niezależne. Wkrótce powiemy o tym więcej.

Metod korelacyjna

Przejdźmy teraz do lepszej metody – do *standardowego sposobu* obliczania transformaty DFT. Działanie metody objaśnimy na przykładzie. Przyjmijmy, że próbujemy obliczyć transformatę DFT sygnału 64-punktowego. Z tego wynika, że powinniśmy obliczyć 33 punkty w części rzeczywistej oraz 33 punkty w części urojonej w dziedzinie częstotliwości. W tym przykładzie przedstawimy tylko sposób obliczania jednej próbki *Im* $X[3]$, tj. amplitudy fali sinusoidalnej, której trzy pełne okresy mieszczą się między punktami 0 i 63. Wszystkie inne wartości w dziedzinie częstotliwości są obliczane w podobny sposób.

Na rys. 8.8 zilustrowano zastosowanie metody korelacyjnej do obliczania *Im* $X[3]$. Na rys. 8.8 a) i 8.8 b) przedstawiono dwa przykłady sygnałów w dziedzinie czasu zwanych, odpowiednio, $x1[\ ]$ i $x2[\ ]$. Pierwszy sygnał, $x[1]$, jest złożony tylko z fali sinusoidalnej, której trzy pełne okresy mieszczą się między punktami 0 i 63. Przeciwnie sygnał $x[2]$, złożony z kilku fal sinusoidalnych i kosinusoidalnych, z których *żadna* nie wykonuje trzech pełnych okresów między punktami 0 i 63. Z tych dwóch sygnałów można wywnioskować, co powinien realizować algorytm do obliczania *Im* $X[3]$. Po wprowadzeniu do niego $x1[\ ]$ algorytm powinien wygenerować wartość 32 – amplitudę fali sinusoidalnej występującej w sygnale (zmodyfikowaną przez współczynniki skalowania z wzorów 8.3). Gdy do algorytmu będzie wprowadzony inny sygnał, $x2[\ ]$, wówczas powinna być generowana wartość zero, wskazując, że ta właśnie fala sinusoidalna nie występuje w sygnale.



Rys. 8.8. Dwa sygnały (rys. a, b) są, jako przykłady, poddawane analizie mającej wykazać, czy zawierają one specyficzne funkcje bazowe przedstawione na rys. c), d). Na rys. e) ,f) pokazano wynik mnożenia każdego przykładowego sygnału przez funkcję bazową. Przebieg na rys. e) ma wartość średnią 0,5, co wskazuje na fakt, że x1[ ] zawiera funkcję bazową o amplitudzie 1,0. Przebieg na rys. f) ma zaś wartość średnią równą zero, wskazującą na brak funkcji bazowej w sygnale x2[ ]

Pojęcie korelacji wprowadziliśmy w rozdziale 7. Jak pamiętacie, aby wykryć znany przebieg zawarty w innym sygnale, trzeba pomnożyć oba sygnały przez siebie i zsumować wszystkie punkty sygnału wynikowego. Jedna liczba, będąca wynikiem tej procedury, jest miarą wzajemnego podobieństwa dwóch sygnałów. Tę metodę zilustrowano na rys. 8.8. Na dwóch rysunkach 8.8 c), d) przedstawiono sygnał, którego szukamy – falę sinusoidalną, której 3 okresy mieszczą się między próbkami 0 i 63. Na rys. 8.8 e) pokazano wynik mnożenia przebiegów a) i c). Podobnie na rys. 8.8 f) mamy rezultat mnożenia przebiegów b) i d). Suma wszystkich punktów w przebiegu e) wynosi 32, podczas gdy suma wszystkich punktów w przebiegu f) jest równa zero, co wskazuje na fakt, że znaleźliśmy żądany algorytm.

Inne próbki w dziedzinie częstotliwości są obliczane w taki sam sposób. Ta procedura jest sformalizowana w postaci *wzoru analitycznego* wyrażającego matematyczny sposób obliczania transformaty w dziedzinie częstotliwości na podstawie sygnału w dziedzinie czasu:

$$Re\,X[k]=\sum_{i=0}^{N-1} x[i]\cos(2\pi ki/N)$$

$$Im\,X[k]=-\sum_{i=0}^{N-1} x[i]\sin(2\pi ki/N)$$

Wzór 8.4. Wzory analityczne do obliczania transformaty DFT. W tych wzorach x [i] jest analizowanym sygnałem w dziedzinie czasu, a Re X[k] i Im X[k] są sygnałami w dziedzinie częstotliwości, które mają być obliczone. Indeks i zmienia się od 0 do N – 1, a indeks k od 0 do N/2

Wyraźmy to słownie: każdą próbkę w dziedzinie częstotliwości oblicza się, mnożąc sygnał w dziedzinie czasu przez falę kosinusoidalną lub sinusoidalną, której szukamy i sumując następnie obliczone wartości punktów. Gdyby ktoś was zapytał, co robicie, powiedzcie w zaufaniu: „Koreluję sygnał wejściowy z każdą funkcją bazową". W tablicy 8.2 podano program komputerowy do obliczania DFT tą metodą.

Wzory analityczne *nie* wymagają specjalnego potraktowania punktów pierwszego i ostatniego, jak to było we wzorach syntetycznych. Przy części urojonej we wzorze 8.4 jest jednak znak minus. Tak jak poprzednio, ta negacja powoduje, że *rzeczywiste* przekształcenie DFT jest zgodne z *zespolonym* DFT. Negacja nie zawsze jest uwzględniana.

Aby ten algorytm korelacyjny funkcjonował, funkcje bazowe muszą mieć interesującą właściwość – każda z nich musi być całkowicie *nieskorelowana* z wszystkimi pozostałymi. To znaczy, że jeśli mnożymy przez siebie którekolwiek dwie z tych funkcji bazowych, to suma wartości obliczonych punktów będzie równa zero. Funkcje bazowe mające taką właściwość nazywamy **ortogonalnymi**. Istnieje wiele innych ortogonalnych funkcji bazowych, takich jak fala prostokątna lub trójkątna, impulsy itd. Sygnały mogą być rozkładane na te inne ortogonalne funkcje bazowe przy użyciu metody korelacyjnej, podobnie jak to robiliśmy, stosując przebiegi sinusoidalne. Nie chodzi tu o sugestię, że to może być *użyteczne*, a tylko o stwierdzenie, że jest taka *możliwość*.

Jak to poprzednio przedstawiono w tablicy 8.1, w *odwrotnym* przekształceniu DFT istnieją dwa sposoby realizacji w postaci programu komputerowego. Różnica polega na *zamianie miejscami* pętli wewnętrznej i zewnętrznej podczas syntezy. Wprawdzie nie zmienia to końcowego wyniku uzyskiwanego w programie, lecz daje inne *spojrzenie* na sposób realizowania programu. Program DFT w tablicy 8.2 też można zmienić, zamieniając miejscami pętle wewnętrzną i zewnętrzną w wierszach od 310 do 380. Tak jak poprzednio, wynik uzyskiwany z programu się nie zmienia, lecz nasz sposób *myślenia* o wykonywanych obliczeniach jest różny. Te dwa różne sposoby spojrzenia na DFT i odwrotne DFT można opisać jako algorytmy „od strony wejścia" i „od strony wyjścia", tak jak przy splocie.

Program podany w tablicy 8.2 jest napisany w taki sposób, że opisuje, jak wszystkie próbki w dziedzinie czasu wpływają na jedną próbkę w dziedzinie częstotliwości. To znaczy, że program oblicza poszczególne wartości w dziedzinie częstotliwości kolejno, a nie jako grupę.

Tab. 8.2.

```
'DYSKRETNE PRZEKSZTAŁCENIE FOURIERA
'Wartości w dziedzinie częstotliwości przechowywane w REX[ ] i IMX[ ] są obliczane
' z sygnału w dziedzinie czasu przechowywanego w XX[ ].

DIM XX[511]                 'W XX[ ] jest przechowywany sygnał w dziedzinie czasu
DIM REX[256]                'W REX[ ] jest przechowywana część rzeczywista sygnału w dziedzinie częstotliwości
DIM IMX[256]                'W IMX[ ] jest przechowywana część urojona sygnału w dziedzinie częstotliwości

PI = 3.14159265             'Ustaw stałą , PI
N% = 512                    'N% jest liczbą punktów w XX[ ]

CALL MITY(XX[])             'Podprogram "mityczny" ładujący dane do XX[ ]

FOR I% = 0 TO 256           'Wyzeruj REX[ ] i IMX[ ], aby mogły być użyte jako akumulatory
  REX[I%] = 0
  IMX[I%] = 0
NEXT I%

                            'Skoreluj XX[ ] z falami kosinusoidalną i sinusoidalną, wzór 8.4

FOR K% = 0 TO 256           'K% tworzy pętlę przez wszystkie próbki w REX[ ] i IMX[ ]
  FOR I% = 0 TO 511         'I% tworzy pętlę przez wszystkie próbki w XX[ ]

    REX[K%] = REX[K%] + XX[I%] * COS(2*PI*K%*I%/N%)
    IMX[K%] = IMX[K%] - XX[I%] * SIN(2*PI*K%*I%/N%)

  NEXT I%
NEXT K%
END
```

Gdy pętle wewnętrzną i zewnętrzną zamieni się miejscami, to program przechodzi w pętli przez każdą próbkę w dziedzinie czasu, obliczając udział wartości tego punktu w sygnale w dziedzinie częstotliwości. Całkowity sygnał w dziedzinie częstotliwości jest wyznaczany przez sumowanie udziałów pochodzących od indywidualnych punktów w dziedzinie czasu. To prowadzi do następnego pytania: jakiego rodzaju udział w dziedzinę częstotliwości wnosi indywidualna próbka w dziedzinie czasu? Odpowiedź jest zawarta w interesującym aspekcie analizy Fouriera, zwanym *dualizmem*.

Dualizm

Równania syntezy i analizy (wzory 8.2 i 8.4) są do siebie uderzająco podobne. Przejście z jednej dziedziny do drugiej wymaga pomnożenia znanych wartości przez funkcje bazowe i zsumowania uzyskanych iloczynów. Fakt, że w *DFT* i w *odwrotnym DFT* stosuje się taki sam aparat matematyczny, jest rzeczywiście godny uwagi, jeśli uwzględnić zupełnie odmienne drogi, jakimi doszliśmy do tych dwóch procedur. Faktycznie, jedyną znacząca różnicą między tymi dwoma równaniami jest uzyskiwany wynik, który w dziedzinie czasu jest *jednym* sygnałem o N punktach, podczas gdy w dziedzinie częstotliwości są to *dwa* sygnały o $N/2 + 1$ punktach. W dalszych rozdziałach będzie mowa o tym, że w *zespolonym* DFT w obu dziedzinach – zarówno czasu, jak i częstotliwości – mamy sygnały zespolone, każdy o N punktach. To czyni obie dziedziny całkowicie symetrycznymi, a wzory służące do przechodzenia z dziedziny do dziedziny – pozornie *identycznymi*.

Taką symetrię między dziedzinami czasu i częstotliwości nazywamy **dualizmem**. Wynika z niego wiele interesujących właściwości. Na przykład pojedynczy punkt w dziedzinie częstotliwości

odpowiada jednej sinusoidzie w dziedzinie czasu. W wyniku dualizmu stwierdzenie odwrotne też jest słuszne; pojedynczy punkt w dziedzinie czasu odpowiada jednej sinusoidzie w dziedzinie częstotliwości. A oto inny przykład. Splot w dziedzinie czasu odpowiada mnożeniu w dziedzinie częstotliwości. W wyniku dualizmu słuszne jest też stwierdzenie odwrotne: splot w dziedzinie częstotliwości odpowiada mnożeniu w dziedzinie czasu. Te i inne wzajemne zależności wynikające z dualizmu będą bardziej szczegółowo rozpatrzone w rozdziałach 10 i 11.

Zapis we współrzędnych biegunowych

W dotychczasowych rozważaniach mieliśmy w dziedzinie częstotliwości grupę amplitud dla kosinusoidlanych i sinusoidalnych (z niewielkimi modyfikacjami wynikającymi ze skalowania). Jest to zapis we współrzędnych **prostokątnych**. Alternatywą jest wyrażenie sygnałów w dziedzinie częstotliwości w postaci **biegunowej**. W tym zapisie *Re X*[ ] i *Im X*[ ] są zastępowane dwiema innymi matrycami nazwanymi **Moduł *X*** (Magnitude X), zapisywany we wzorach jako ***Mag X***[ ] oraz **Faza X** (Phase *X*) zapisywana jako ***Phase X***[ ][14]. Moduł i faza ściśle zastępują części rzeczywistą i urojoną. Na przykład wartości *Mag X*[0] i *Phase X*[0] są obliczane przy wykorzystaniu tylko *Re X*[0] i *Im X*[0]. Podobnie *Mag* [14] i *Phase X*[14] są obliczane w z wykorzystaniem tylko *Re X*[14] i *Im X*[14] i tak dalej. W celu zrozumienia takiej konwersji rozważmy, co się dzieje w przypadku sumowaniu fal kosinusoidalnej i sinusoidalnej o tej samej częstotliwości. Wynikiem sumowania jest kosinusoida o tej samej częstotliwości, lecz o innej amplitudzie i innym przesunięciu fazy. Można to wyrazić równaniem:

$$A\cos(x)+B\sin(x)=M\cos(x+\theta)$$

Wzór 8.5. Sumowanie fal kosinusoidalnej i sinusoidalnej daje w wyniku falę kosinusoidalną o innej amplitudzie i przesunięciu fazy. Informacja zawarta w A i B jest przenoszona do dwóch innych zmiennych M i θ

Ważne jest, że w tym procesie konwersji żadna informacja nie jest tracona; mając daną jedną reprezentację sygnału, można obliczyć drugą. A więc informacja zawarta w amplitudach *A* i *B* jest też zawarta w zmiennych *M* i θ. Chociaż równanie 8.5 dotyczy fal sinusoidalnych i kosinusoidalnych, to jest ono zgodne z podobnymi równaniami dotyczącymi prostych wektorów. Na rys. 8.9 pokazano reprezentację wektorową, w której dwie zmienne *A* i *B* mogą być widziane we współrzędnych prostokątnych, podczas gdy *M* i θ są parametrami we współrzędnych biegunowych.

We współrzędnych biegunowych *Mag X*[ ] wyraża amplitudę fali kosinusoidalnej (*M* we wzorach 8.5 i na rys. 8.9), a faza *Phase X*[ ] wyraża kąt fazowy fali kosinusoidalnej (θ we wzorach 8.5 i na rys. 8.9). Poniższe wzory służą do konwersji sygnału w dziedzinie częstotliwości ze współrzędnych prostokątnych na biegunowe i vice versa:

$$Mag\,X[k]=\left(Re\,X[k]^2+Im\,X[k]^2\right)^{1/2}$$

$$Phase\,X[k]=\text{arctg}\left(\frac{Im\,X[k]}{Re\,X[k]}\right)$$

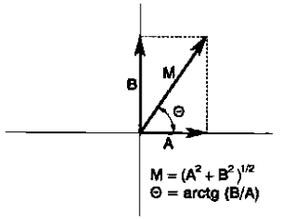
Wzór 8.6. Konwersja ze współrzędnych prostokątnych na biegunowe. Reprezentacja sygnału w dziedzinie częstotliwości we współrzędnych prostokątnych Re X [k] i Im X [k] jest przekształcana na reprezentację we współrzędnych biegunowych Mag X [k] i Phase X [k]

$$Re\,X[k]=Mag\,X[k]\cos\left(Phase\,X[k]\right)$$

$$Im\,X[k]=Mag\,X[k]\sin\left(Phase\,X[k]\right)$$

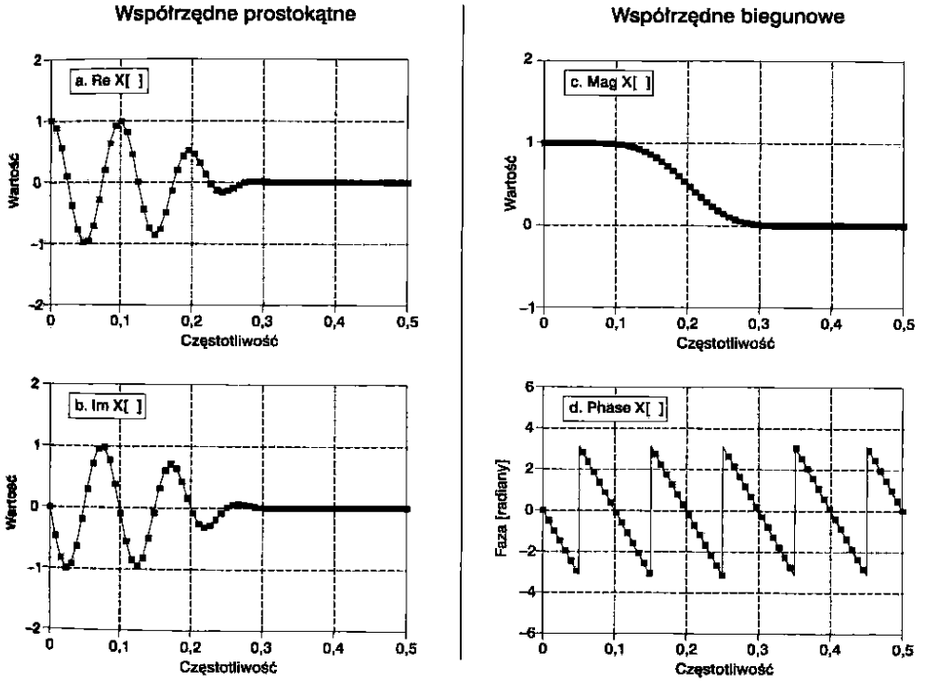
Wzór 8.7. Konwersja ze współrzędnych biegunowych na prostokątne. Dwie matryce Mag X[k] i Phase X [k] są przekształcane na Re X [k] i Im X [k]

[14] W tłumaczeniu zachowano angielskie symbole *Mag X*[ ] i *Phase X*[ ] ze względu na ich zastosowanie w przykładach programów komputerowych (przyp. tłum.).



Rys. 8.9. Konwersja ze współrzędnych prostokątnych na biegunowe. Dodawanie fal kosinusoidalnej i sinusoidalnej (o tej samej częstotliwości) podlega takim samym operacjom matematycznym, jak dodawanie prostych wektorów

Zależnie od sposobu zapisu – we współrzędnych prostokątnych lub biegunowych – można rozważać DFT na dwa różne sposoby. DFT wyrażone we współrzędnych prostokątnych daje rozkład sygnału *N*-punktowego na *N*/2 + 1 fal kosinusoidalnych i *N*/2 + 1 sinusoidalnych, każda z nich o określonej *amplitudzie*. W zapisie we współrzędnych biegunowych DFT daje rozkład sygnału *N*-punktowego na *N*/2 + 1 fal kosinusoidalnych, każda z nich o określonej amplitudzie (zwanej *modułem*) i *przesunięciu fazy*. Dlaczego w zapisie we współrzędnych biegunowych są stosowane fale kosinusoidalne zamiast sinusoidalnych? Fale sinusoidalne nie mogą wyrażać składowej stałej (*DC*) sygnału, ponieważ fala sinusoidalna o zerowej częstotliwości jest złożona z samych zer (rys. 8.5 a, b).



Rys. 8.10. Przykład reprezentacji sygnału w dziedzinie częstotliwości we współrzędnych prostokątnych i biegunowych. W tym przykładzie przedstawiono zarówno zapis we współrzędnych prostokątnych, jak i biegunowych. Tak jak w tym przykładzie, zapis we współrzędnych biegunowych daje zwykle obserwatorowi lepsze zrozumienie właściwości sygnału. Zapis we współrzędnych prostokątnych jest zaś prawie zawsze stosowany tam, gdzie są konieczne operacje matematyczne. Zwróćcie szczególną uwagę na fakt, że pierwsza i ostatnia próbka w wykresie przesunięcia fazy muszą być zerowe, podobnie jak to jest w części urojonej

Zapisy we współrzędnych prostokątnych i biegunowych zawierają wprawdzie dokładnie samą informację, lecz istnieje wiele takich problemów, w których jeden zapis jest łatwiejszy od drugiego. Na rys. 8.10 przedstawiono sygnał w dziedzinie częstotliwości zarówno we współrzędnych prostokątnych, jak i biegunowych. Ostrzeżenie: Nie próbujcie zrozumieć kształtu części rzeczywistej i urojonej; głowa wam pęknie! Natomiast wykresy we współrzędnych biegunowych są zrozumiałe w sposób bezpośredni; występują tylko częstotliwości poniżej ok. 0,25 i przesunięcie fazy jest w przybliżeniu proporcjonalne do częstotliwości. Jest to odpowiedź częstotliwościowa filtru dolnoprzepustowego.

Kiedy należy stosować zapis we współrzędnych prostokątnych, a kiedy w biegunowych? Współrzędne prostokątne są na ogół korzystniejsze do obliczeń, a więc we wzorach i w programach komputerowych. Wykresy zaś są prawie zawsze sporządzane we współrzędnych biegunowych. Jak to już wykazano na poprzednim przykładzie, dla *człowieka* jest prawie niemożliwe zrozumienie właściwości sygnału w dziedzinie częstotliwości przez oglądanie wykresów części rzeczywistej i urojonej. W typowym programie sygnały w dziedzinie częstotliwości są utrzymywane we współrzędnych prostokątnych dopóty, dopóki obserwator nie ma potrzeby ich obejrzenia, wtedy wykonuje się konwersję na współrzędne biegunowe.

Dlaczego sygnał w dziedzinie częstotliwości jest łatwiejszy do zrozumienia we współrzędnych biegunowych niż w prostokątnych? To pytanie sięga do samego sedna kwestii przed*latności* rozkładu sygnału na sinusoidy. Przypomnijmy sobie pojęcie wierności odtwarzania sinusoidy, o której była mowa w rozdziale 5. Jeśli sinusoidę wprowadzi się do systemu liniowego, to na wyjściu też uzyska się sinusoidę o dokładnie takiej samej częstotliwości, jak na wejściu. Zmienić się mogą tylko amplituda i faza. Zapis we współrzędnych biegunowych reprezentuje sygnały bezpośrednio przez amplitudę i fazę składowych fal kosinusoidalnych. A więc systemy mogą być reprezentowane przez stwierdzenie, w jaki sposób modyfikują amplitudę i fazę każdej z tych fal kosinusoidalnych.

Zastanówmy się teraz, co się stanie, jeśli zapis we współrzędnych prostokątnych zastosujemy w następującej sytuacji. Mieszanina sinusoid i kosinusoid jest wprowadzana do systemu liniowego, a więc na wyjściu też jest mieszanina sinusoid i kosinusoid. Problem polega na tym, że fala kosinusoidalna na wejściu może na wyjściu dać zarówno falę kosinusoidalną, jak i sinusoidalną. Podobnie wejściowa fala sinusoidalna może na wyjściu dać zarówno fale kosinusoidalną, jak i sinusoidalną. Chociaż to skrzyżowanie pojęć można jakoś wyprostować, to cała sprawa niezbyt pasuje do kwestii, dlaczego w pierwszej kolejności chcieliśmy stosować sinusoidy.

Niedogodności współrzędnych biegunowych

Jest wiele niedogodności związanych ze stosowaniem zapisu we współrzędnych biegunowych. Żadna z nich nie ma decydującego znaczenia, są tylko rzeczywiście kłopotliwe. W tablicy 8.3 podano program komputerowy do konwersji między zapisami we współrzędnych prostokątnych i biegunowych. W tym programie podano sposoby rozwiązania niektórych tych niedogodności.

Niedogodność 1 – radiany i stopnie

Fazę można wyrażać albo w *stopniach*, albo w *radianach*. Wartość fazy sygnału, podana w stopniach, mieści się w granicach od –180 do 180. Przy użyciu radianów każda wartość będzie w granicach od $-\pi$ do π, czyli między –3,141592 i 3,141592. W większości języków komputerowych jest wymagane stosowanie radianów w funkcjach trygonometrycznych, takich jak sinus, kosinus, arcus tangens itd. Praca z tymi długimi liczbami dziesiętnymi może być irytująca, a uzyskane dane – trudne do interpretacji. Na przykład, jeśli chcemy wprowadzić do sygnału 90-stopniowe przesunięcie fazy, to musimy dodać do jego fazy liczbę 1,570796. Chociaż wprowadzenie tej liczby do programu nie jest zadaniem ponad siły, to jednak bywa nużące. Najlepszym sposobem rozwiązania tej kwestii jest zdefiniowanie, na początku programu stałej równej *PI* = 3,141592. Wtedy 90-stopniowe przesunięcie fazy można zapisać jako *PI*/2. W DSP szeroko stosuje się równo stopnie, jak i radiany, dlatego trzeba umieć się posługiwać obiema tymi jednostkami.

Tab. 8.3.

```
'KONWERSJA WSPÓŁRZĘDNYCH PROSTOKĄTNYCH NA BIEGUNOWE I BIEGUNOWYCH NA PROSTOKĄTNE
'
DIM REX[256]                    'REX[ ] przechowuje część rzeczywistą
DIM IMX[256]                    'IMX[ ] przechowuje część urojoną
DIM MAG[256]                    'MAG[ ] przechowuje moduł
DIM PHASE[256]                  'PHASE[ ] przechowuje fazę
'
PI = 3.14159265
'
GOSUB XXXX                      'Podprogram "mityczny" ładujący dane do REX[ ] i IMX[ ]
'
'
'                               'Konwersja współrzędnych prostokątnych na biegunowe, wzór 8.6
FOR K% = 0 TO 256
  MAG[K%] = SQR( REX[K%]^2 + IMX[K%]^2 )   'z wzoru 8.6
  IF REX[K%] = 0 THEN REX[K%] = 1E-20      'zapobieganie dzieleniu przez zero (niedogodność 2)
  PHASE[K%] = ATN( IMX[K%] / REX[K%] )     'z wzoru 8.6
  '                                        'korekta arcus tangensa (niedogodność 3)
  IF REX[K%] < 0 AND IMX[K%] < 0 THEN PHASE[K%] = PHASE[K%] - PI
  IF REX[K%] < 0 AND IMX[K%] >= 0 THEN PHASE[K%] = PHASE[K%] + PI
NEXT K%
'
'
'                               'Konwersja współrzędnych biegunowych na prostokątne, wzór 8.7
FOR K% = 0 TO 256
  REX[K%] = MAG[K%] * COS( PHASE[K%])
  IMX[K%] = MAG[K%] * SIN( PHASE[K%])
NEXT K%
'
END
```

Niedogodność 2 – błąd dzielenia przez zero

Przechodząc z zapisu we współrzędnych prostokątnych na współrzędne biegunowe, często można znaleźć częstotliwości, przy których część rzeczywista ma wartość zerową, część urojona zaś jakąś wartość niezerową. To znaczy po prostu, że faza jest równa dokładnie 90° lub –90°. Spróbujcie powiedzieć to waszemu komputerowi! Gdy wasz program próbuje obliczyć fazę ze wzoru: *Phase X*[*k*] = arctg(*Im X*[*k*]/*Re X*[*k*]), to pojawia się *błąd dzielenia przez zero*. Nawet jeśli wykonywanie programu się nie zatrzyma, to uzyskana przy tej częstotliwości wartość fazy nie będzie prawidłowa. W celu uniknięcia tej przeszkody, trzeba przed wykonaniem operacji dzielenia sprawdzić, czy część rzeczywista nie jest zerowa. Jeśli jest ona równa zeru, to należy sprawdzić, czy część urojona jest dodatnia, czy ujemna w celu ustalenia wartości fazy, odpowiednio na $\pi/2$ lub $-\pi/2$. Wreszcie dzielenie powinno być ominięte. W realizacji wymienionych czynności nie ma niczego trudnego, poza potencjalnym zdenerwowaniem. Alternatywny sposób rozwiązania tego problemu pokazano w wierszu 250 programu w tablicy 8.3. Jeśli część rzeczywista jest równa zeru, to trzeba ją zamienić na liczbę pomijalnie małą, aby procesor matematyczny nie miał kłopotów z wykonaniem dzielenia.

Niedogodność 3 – nieprawidłowy arcus tangens

Rozważmy próbkę w dziedzinie częstotliwości, w której *Re X*[*k*] = 1 i *Im X*[*k*] = 1. Z wzoru 8.6 wynikają równoważne współrzędne biegunowe: *Mag X*[*k*] = 1,414 i *Phase X*[*k*] = 45°. Rozpatrzmy teraz inną próbkę, w której *Re X*[*k*] = –1 i *Im X*[*k*] = –1. Wzór 8.6 znowu daje *Mag X*[*k*] = 1,414 i *Phase X*[*k*] = 45°. Problem polega na tym, że wartość fazy jest błędna! Powinna być –135°. Ten błąd powstaje zawsze, gdy część rzeczywista jest ujemna. Można go skorygować, sprawdzając części rzeczywistą i urojoną po obliczeniu fazy. Jeśli obie części

– rzeczywista i urojona – są ujemne, to trzeba odjąć 180° (lub π radianów) od obliczonej fazy. W wierszach 340 i 350 programu w tablicy 8.3 pokazano, jak to jest realizowane. Jeśli pominie się tę kwestię, to obliczona wartość fazy będzie się mieścić w granicach od $-\pi/2$ do $\pi/2$, zamiast od $-\pi$ do π. Dobrze to zapamiętajcie. Jeśli widzicie, że faza mieści się w granicach ±1,5708, to znaczy, że zapomnieliście skorygować dwuznaczność w obliczeniach arcus tangensa.

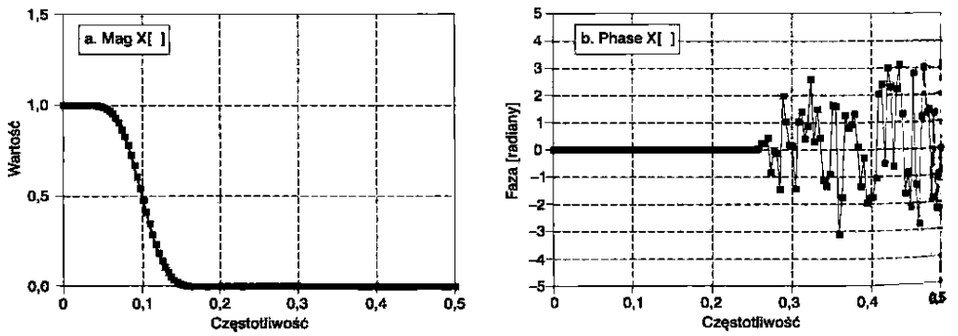
Niedogodność 4 – faza przy bardzo małych modułach

Wyobraźmy sobie następującą sytuację. Rozgryzacie właśnie jakiś problem DSP i nagle zauważacie, że część wyników dotyczących fazy jest chyba nie w porządku. Mogą być obarczone szumem, zmieniać się skokowo lub po prostu być *błędne*. Spędziwszy następną godzinę na przeszukiwaniu setek wierszy programu komputerowego, znajdujecie odpowiedź. Odpowiednie wartości modułu są tak małe, że są ukryte w szumie zaokrąglania. Jeśli moduł jest pomijalnie mały, faza nie ma żadnego znaczenia i może przyjmować niezwykłe wartości. Przykład przedstawiono na rys. 8.11. Jeśli *amplituda* sygnału ginie w szumie, to nie ma sygnału, przypuszcza się, że jej wartość jest bardzo mała, a więc bez znaczenia. Co innego z fazą. Gdy sygnał wyrażony we współrzędnych biegunowych jest obarczony szumem, to wartości fazy są liczbami losowymi z zakresu od $-\pi$ do π. Niestety, to często *wygląda jak* rzeczywisty sygnał, a nie jak nonsens, którym w istocie jest.

Niedogodność 5 – niejednoznaczność fazy równa 2π

Spójrzcie raz jeszcze na rys. 8.10 d) i zauważcie kilka nieciągłości w przedstawionych danych. Za każdym razem, gdy zdaje się, że punkt zejdzie poniżej wartości –3,141592, skacze on z powrotem do 3,141592. Jest to wynikiem okresowego charakteru sinusoidy. Na przykład przesunięcie fazy θ jest dokładnie takie samo, jak przesunięcie $\theta + 2\pi$, $\theta + 4\pi$, $\theta + 6\pi$ itd. Każda sinusoida pozostaje niezmieniona, jeśli do jej fazy doda się całkowitą wielokrotność 2π. Wyraźne nieciągłości są wywoływane programem komputerowym wybierającym swoje ulubione spośród nieskończonej liczby równoważnych możliwości. Zawsze jest wybierana najmniejsza możliwa wartość, co utrzymuje fazę w granicach od $-\pi$ do π.

Często łatwiej jest zrozumieć fazę, jeśli nie ma tych nieciągłości, nawet jeśli to znaczy, że faza wykracza powyżej π lub poniżej $-\pi$. Jest to nazywane **rozwijaniem fazy**, którego przykład pokazano na rys. 8.12. Jak to przedstawiono w tablicy 8.4, do każdej wartości fazy jest dodawane lub odejmowane 2π. Wartość dokładna jest określana przez algorytm, który minimalizuje różnicę między sąsiednimi próbkami.



Rys. 8.11. Faza sygnałów o małej wartości modułu. Dla częstotliwości, przy których moduł spada do bardzo małej wartości, szum zaokrąglania może spowodować znaczne odchylenia przebiegu fazy. Nie zróbcie błędu, myśląc, że jest to znaczący sygnał

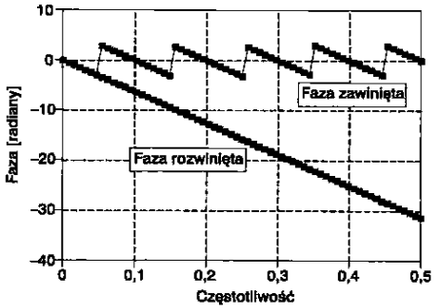
Tab. 8.4.

```
'ROZWIJANIE FAZY
'
DIM PHASE[256]                  'W PHASE [ ] jest przechowywana faza oryginalna
DIM UWPHASE[256]                'W UWPHASE [ ] jest przechowywana faza przed rozwinięciem
'
PI = 3.14159265
'
GOSUB XXXX                      'Podprogram "mityczny" ładujący dane do PHASE[ ]
'
UWPHASE[0] = 0                  'Pierwszy punkt wszystkich sygnałów fazy jest zero
'
                                'Przejdź przez algorytm rozwijania
FOR K% = 1 TO 256
  C% = CINT((UWPHASE[K%-1] - PHASE[K%]) / (2 * PI))
  UWPHASE[K%] = PHASE[K%] + C%*2*PI
NEXT K%
'
END
```

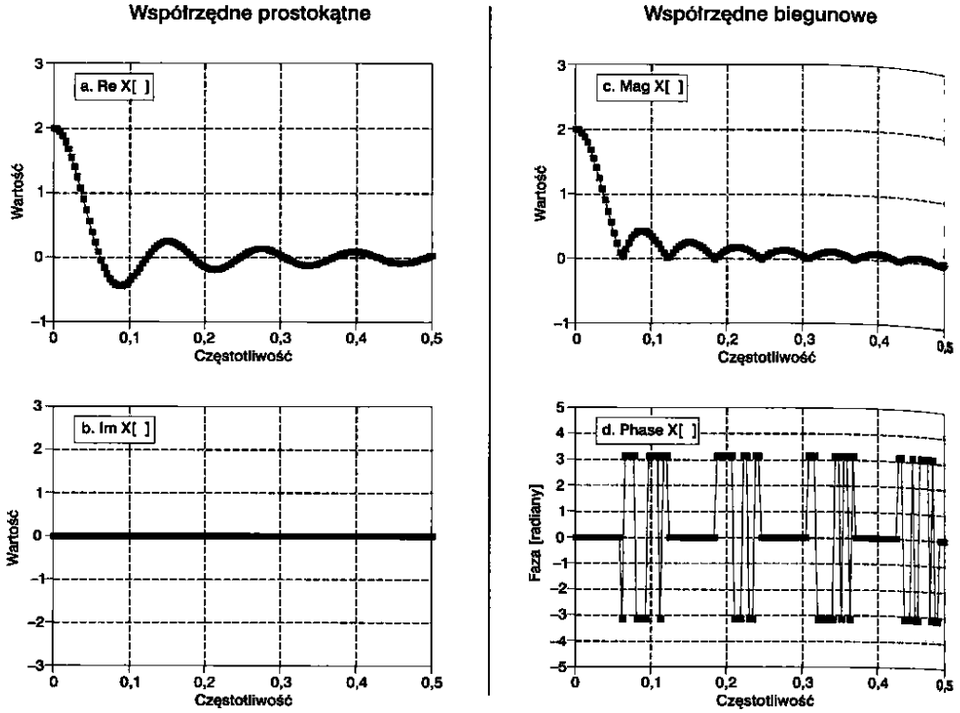
Niedogodność 6 – moduł jest zawsze dodatni (niejednoznaczność fazy równa π)

Na rys. 8.13 pokazano sygnał w dziedzinie częstotliwości we współrzędnych prostokątnych i biegunowych. Przebieg części rzeczywistej jest gładki i zupełnie łatwy do zrozumienia, podczas gdy część urojona jest całkowicie zerowa. Sygnały zaś we współrzędnych biegunowych zawierają nagłe nieciągłości i ostre naroża. Przyczyną jest moduł, który z *definicji* zawsze jest dodatni. Gdy tylko część rzeczywista staje się mniejsza od zera, moduł pozostaje dodatni przez zmianę fazy o π (lub $-\pi$, co daje taki sam skutek). Nieregularne krzywe nie są żadnym problemem w matematyce, ale mogą być trudne do interpretacji.

Jednym z rozwiązań tego problemu jest przyzwolenie na to, aby moduł miał wartości *ujemne*. W przykładzie z rys. 8.13 spowodowałoby to, że moduł miałby taki sam przebieg, jak część rzeczywista, podczas gdy faza byłaby całkowicie zerowa. Nie ma w tym nic złego, o ile tylko pomaga w zrozumieniu. Zwracajcie jednak uwagę, aby sygnału o ujemnych wartościach nie nazywać „modułem", gdyż łamałoby jego formalną definicję. W tej książce nazwa *rozwinięty moduł* jest używana do określenia „modułu", który może przyjmować wartości ujemne.



Rys. 8.12. Przykład rozwijania fazy. Górna krzywa jest typowym przebiegiem fazy sygnału otrzymanym przez konwersję współrzędnych prostokątnych na biegunowe. Wartości tego sygnału muszą mieścić się między $-\pi$ i π (tj. –3,14159 i 3,14159). Jak pokazano na dolnej krzywej, faza może zostać rozwinięta przez dodanie lub odjęcie od każdej próbki całkowitych wielokrotności 2π. Wielokrotność jest tak wybrana, aby zminimalizować nieciągłości między punktami



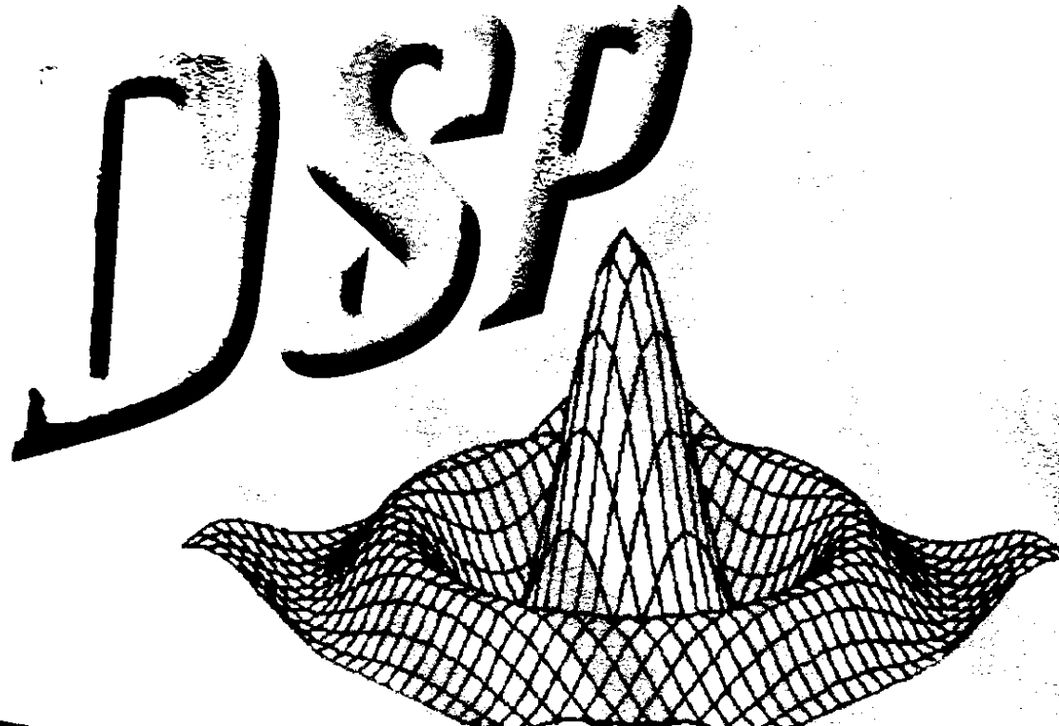
Rys. 8.13. Przykład sygnałów we współrzędnych prostokątnych i biegunowych. Ponieważ moduł zawsze musi być dodatni (z definicji), więc przebiegi modułu i fazy mogą zawierać nagłe nieciągłości i ostre naroża. Na rys. d) przedstawiono jeszcze jedną niedogodność współrzędnych biegunowych – szum losowy może powodować szybkie oscylacje fazy między π lub –π

Niedogodność 7 – wyskoki między π i –π

Ponieważ π i –π wyrażają takie samo przesunięcie fazy, więc szum zaokrąglania może powodować ostre przełączania sąsiednich punktów charakterystyki fazy między dwiema wartościami. Jak pokazano na rys. 8.13 d), może to wytwarzać ostre przerwy i wyskoki na gładkiej poza tym krzywej. Nie dajcie się omamić; w rzeczywistości faza nie ma takich nieciągłości.

9

Zastosowania DFT



Dyskretne przekształcenie Fouriera (DFT) jest jednym z najważniejszych narzędzi w cyfrowym przetwarzaniu sygnałów. W tym rozdziale będą omówione trzy typowe sposoby stosowania tego przekształcenia. Po pierwsze, stosując DFT, można obliczać *częstotliwościowe widmo* sygnału. Uzyskuje się je przez bezpośrednie wykorzystanie informacji zakodowanej w częstotliwości, fazie i amplitudzie sinusoid składowych. Na przykład w ludzkiej mowie i słyszeniu występują sygnały o tym typie zakodowania. Po drugie, za pomocą DFT można znaleźć odpowiedź częstotliwościową sygnału na podstawie jego odpowiedzi impulsowej i vice versa. Dzięki temu systemy mogą być analizowane *w dziedzinie częstotliwości*, podobnie jak splot daje możliwość analizy *w dziedzinie czasu*. Po trzecie, DFT można stosować jako krok pośredni w bardziej rozwiniętych metodach przetwarzania sygnałów. Klasycznym przykładem jest tu *splot FFT*, algorytm do realizacji splotu sygnałów, który jest setki razy szybszy od metod konwencjonalnych.

Analiza widmowa sygnałów

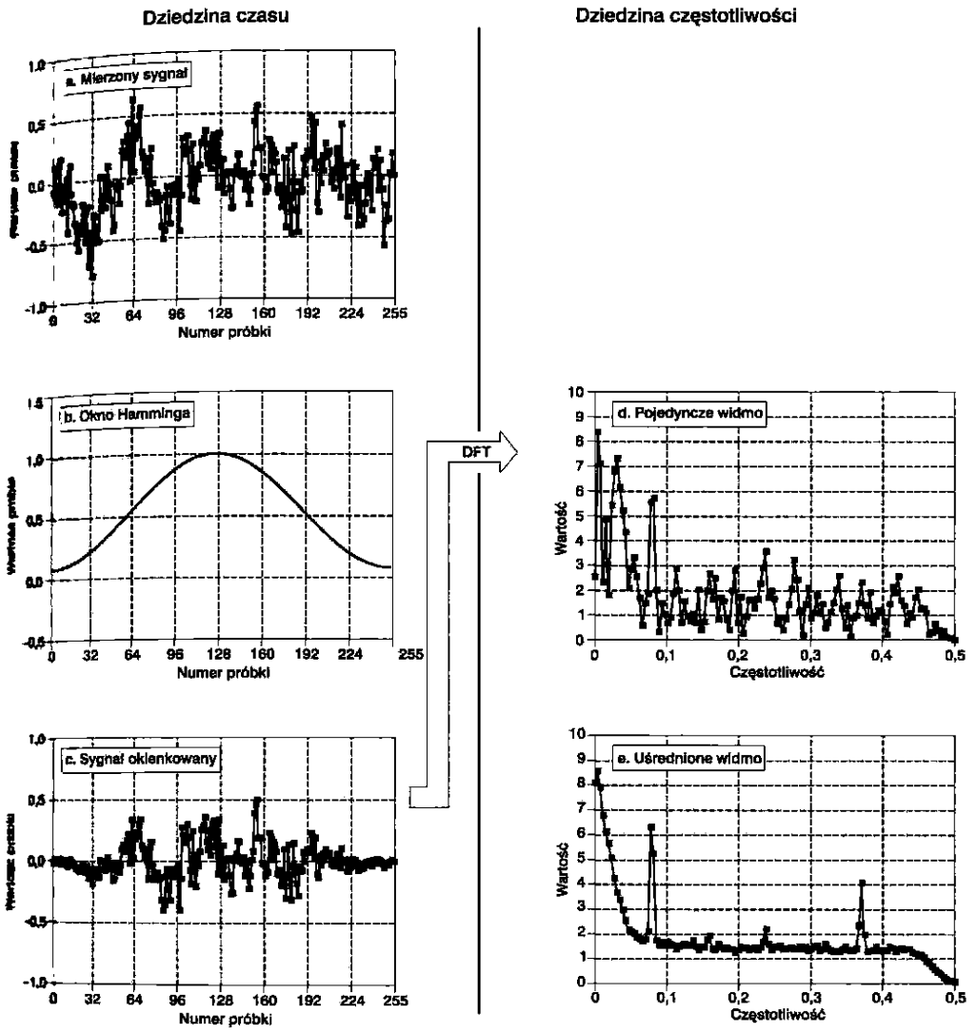
Bardzo często informacja bywa zakodowana w sinusoidach tworzących sygnał. Dotyczy to zarówno sygnałów powstających w sposób naturalny, jak i tworzonych przez ludzi. W naszym Wszechświecie wiele rzeczy oscyluje. Mowa, na przykład, jest wynikiem wibracji strun głosowych człowieka; gwiazdy i planety zmieniają swoją jasność, gdy obracają się wokół własnej osi lub jedna wokół drugiej; śruby okrętowe wytwarzają okresowe przemieszczenia wody i tak dalej. W tych sygnałach *kształt* przebiegu w dziedzinie czasu nie jest ważny. Główna informacja jest zawarta w *częstotliwości, fazie i amplitudzie sinusoid* składowych. Przekształcenie DFT jest stosowane do wydobycia tej informacji.

Rozważymy to na przykładzie. Załóżmy, że chcemy zbadać dźwięki rozchodzące się w oceanie. Najpierw umieszcza się w wodzie mikrofon i uzyskany sygnał elektroniczny wzmacnia do odpowiedniego poziomu, np. kilku woltów. Następnie analogowym filtrem dolnoprzepustowym usuwa się wszystkie częstotliwości powyżej 80 Hz, a więc sygnał można próbkować z częstotliwością 160 próbek/s. Zbieramy i rejestrujemy kilka tysięcy próbek, i co dalej?

W pierwszej kolejności trzeba po prostu *przyjrzeć się* uzyskanym danym. Na rys. 9.1 przedstawiono 256 próbek z naszego wymyślonego eksperymentu. Wszystko, co możemy zobaczyć, to zaszumiony przebieg, który ludzkiemu oku przekazuje niewiele informacji. Z powodów, które wkrótce przedstawimy, następnym krokiem jest pomnożenie tego sygnału przez gładką krzywą zwaną **oknem Hamminga** (rys. 9.1 b). Wzory opisujące okno Hamminga i inne okna będą podane w rozdziale 16 (wzory 16.1 i 16.2 oraz rys. 16.2 a). W wyniku mnożenia uzyskujemy 256-punktowy sygnał, w którym próbki bliskie końca sygnału mają zmniejszoną amplitudę, jak to przedstawiono na rys. 9.1 c).

Przeprowadzając przekształcenie DFT i przechodząc do współrzędnych biegunowych, otrzymujemy 129-punktowe widmo częstotliwościowe (rys. 9.1 d). Niestety i to widmo wygląda jak zaszumiona gmatwanina. Jest tak, gdyż w 256-punktowym sygnale oryginalnym nie ma dość informacji koniecznej do uzyskania wyraźnej krzywej. Nic na to nie pomoże zastosowanie przekształcenia DFT o większej liczbie punktów. Na przykład jeśli zastosujemy 2048-punktowe DFT, to widmo częstotliwościowe będzie miało 1025 punktów. Chociaż oryginalne 2048 punktów zawiera więcej informacji, to większa liczba próbek w widmie rozmywa informację o taki sam czynnik. Przekształcenie DFT o większej liczbie punktów daje lepszą rozdzielczość częstotliwości, lecz taki sam poziom szumu.

Odpowiedzią na te trudności jest zastosowanie większej ilości informacji o sygnale oryginalnym w sposób, który nie zwiększa liczby punktów w widmie częstotliwościowym. Można tego dokonać, dzieląc sygnał wejściowy na wiele *segmentów* 256-punktowych. Każdy z tych segmentów jest mnożony przez okno Hamminga i poddawany 256-punktowemu przekształceniu DFT i konwersji na współrzędne biegunowe. Uzyskane w ten sposób widma częstotliwościowe są następnie *uśredniane* i tworzą 129-punktowe widmo częstotliwościowe. Na rys. 9.1 e) przedstawiono przykład uśredniania 100 spośród widm częstotliwościowych zgodnych z rys. 9.1 d). Poprawa jest oczywista; szum został zredukowany do poziomu umożliwiającego dostrzeżenie interesujących właściwości obserwowanego sygnału. W ten sposób w dziedzinie częstotliwości jest uśredniany tylko *moduł*. *Faza* jest zwykle pomijana, gdyż nie zawiera użytecznych informacji. Szum losowy jest redukowany proporcjonalnie do *pierwiastka kwadratowego* z liczby segmentów. Chociaż 100 segmentów jest liczbą typową, to w niektórych zastosowaniach można uśredniać *miliony* segmentów, aby uwypuklić bardzo słabe cechy widma.



Rys. 9.1. Przykład analizy widmowej. Na rys. a) przedstawiono 256 próbek pobranych z (symulowanego) podwodnego mikrofonu z częstotliwością 160 próbek/s. Ten sygnał jest mnożony przez okno Hamminga przedstawione na rys. b) dając w wyniku sygnał okienkowany (rys. c). Widmo częstotliwościowe tego okienkowanego sygnału jest wyznaczone przy użyciu przekształcenia DFT i pokazane na rys. d) (tylko moduł). Uśrednienie 100 takich widm redukuje szum losowy, dając w rezultacie uśrednione widmo częstotliwościowe (rys. e)

Jest jeszcze druga metoda tłumienia szumu widmowego. Zaczynamy od wykonania przekształcenia DFT o bardzo dużej liczbie punktów, np. 16 384. Uzyskane widmo ma wysoką rozdzielczość (8193 próbki), lecz jest bardzo zaszumione. Dlatego stosuje się dolnoprzepustowy filtr cyfrowy służący do *wygładzenia* widma. Ten filtr tłumi szum kosztem rozdzielczości. Na przykład najprostszy filtr cyfrowy do wytworzenia każdej próbki w widmie odfiltrowanym może uśredniać 64 sąsiednie próbki widma oryginalnego. Wykonując odpowiednie obliczenia, uzyskuje się mniej więcej taki sam szum i rozdzielczość, jak w pierwszej metodzie z podziałem 16 384 punktów na 64 segmenty po 256 punktów.

Którą z metod należy stosować? Pierwsza jest łatwiejsza, gdyż nie wymaga filtru cyfrowego. Druga ma *potencjalne* możliwości lepszego działania, ponieważ filtr cyfrowy można tak dobrać, aby uzyskać najkorzystniejszy kompromis szumu z rozdzielczością. Jednak to ulepszone działanie jest rzadko kiedy warte zachodu, gdyż zarówno szum, jak i rozdzielczość można poprawić, wykorzystując *więcej* danych z sygnału wejściowego. Wyobraźmy sobie na przykład podział zebranych danych na 10 000 segmentów, po 16 384 próbki w każdym. Uzyskane w rezultacie widmo częstotliwościowe ma wysoką rozdzielczość (8193 punkty) i mały szum (10 000 uśrednień). A więc problem rozwiązany! Dlatego w tym omówieniu będziemy się zajmować tylko metodą z uśrednianiem segmentów.

Na rys. 9.2 przedstawiono przykład widma uzyskanego z mikrofonu podwodnego. To widmo ilustruje właściwości często występujące w widmach częstotliwościowych zbieranych sygnałów. Na chwilę pomińmy ostre piki. W obszarze od 10 do 70 Hz widmo ma dość płaski przebieg. Nazywamy to **szumem białym** z powodu jednakowej zawartości wszystkich częstotliwości, podobnie jak w świetle białym. Ten szum wynika z szumu w przebiegu w dziedzinie czasu, który jest *nieskorelowany* między poszczególnymi próbkami. Tak więc znajomość wartości szumu w którejkolwiek próbce nie dostarcza żadnej informacji o wartości szumu w innych próbkach. Szum biały jest generowany na przykład przez losowy ruch elektronów w układach elektronicznych. Przykładem szumu białego w życiu codziennym może być dźwięk strumienia wody spadającego na podłogę natrysku. Biały szum pokazany na rys. 9.2 może pochodzić z jednego z kilku możliwych źródeł, np. z analogowych układów elektronicznych lub z samego oceanu.

Powyżej 70 Hz amplituda szumu białego gwałtownie się zmniejsza. Jest to wynikiem wpływu nachylenia charakterystyki filtru antyaliasingowego. Filtr idealny powinien przepuszczać wszystkie częstotliwości poniżej 80 Hz i obcinać wszystkie częstotliwości powyżej tej granicy. Idealne obcinanie nie jest możliwe w praktyce, dlatego należy się spodziewać stopniowego opadania charakterystyki. Jeśli tego zjawiska nie ma, to można oczekiwać wystąpienia aliasingu.

Poniżej częstotliwości 10 Hz szum gwałtownie się zwiększa z powodu osobliwego zjawiska zwanego **szumem 1/f** . Szum 1/f jest swego rodzaju zagadką. Zmierzono go w wielu różnych systemach, takich jak gęstość ruchu na bezpłatnych autostradach i elektroniczny szum w tranzystorach. Prawdopodobnie można by istnienie tego szumu stwierdzić we wszystkich systemach, gdyby tylko poszukać go przy odpowiednio małych częstotliwościach. Mimo częstego występowania szumu typ 1/f, pełne jego wyjaśnienie i stworzenie ogólnej teorii dotychczas się naukowcom nie powiodło. Przyczyny powstawania tego szumu mogą być zidentyfikowane w niektórych specyficznych systemach, co jednak nie wyjaśnia, dlaczego szum 1/f jest wszędzie. W często stosowanych układach analogowych i większości systemów fizycznych przejście od szumu białego do szumu 1/f następuje w obszarze między 1 i 100 Hz.

Zajmijmy się teraz ostrymi pikami w wykresie na rys. 9.2. Najprostszym sposobem wyjaśnienia piku przy częstotliwości 60 Hz jest wpływ zakłóceń elektromagnetycznych pochodzących od sieci energetycznej[15]. Można się też spodziewać mniejszych pików przy wielokrotnościach tej częstotliwości (120, 180, 240 Hz itd.), gdyż przebieg napięcia w sieci

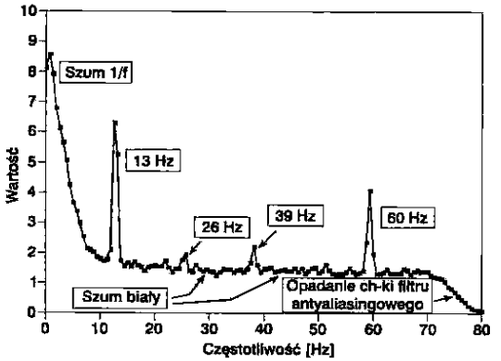
[15] 60 Hz – częstotliwość sieci energetycznej w USA. W Europie obowiązuje 50 Hz (przyp. tłum.).

energetycznej nie jest *idealną* sinusoidą. Często można też spotkać piki zakłócające w zakresie częstotliwości od 25 do 40 kHz, ulubionym przez projektantów zasilaczy impulsowych. Pobliskie stacje radiowe i telewizyjne mogą wytwarzać piki zakłócające w zakresie megahercowym. Piki przy małych częstotliwościach mogą być wywoływane wibracjami elementów systemu przy wstrząsach. Jest to tzw. efekt *mikrofonowania*, dający piki typowo w zakresie od 10 do 100 Hz.

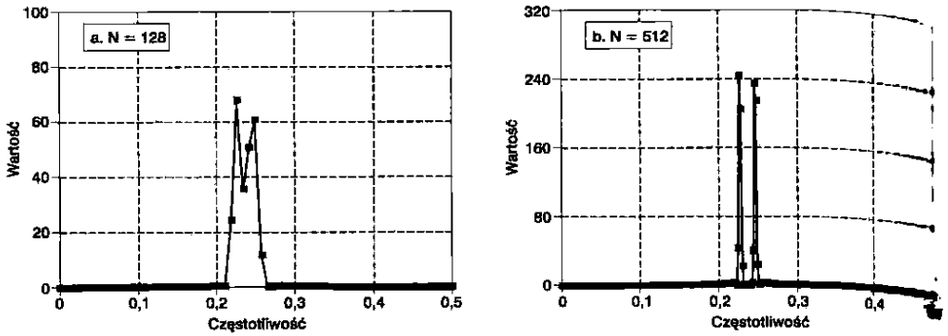
Przejdźmy teraz do sygnałów właściwych. Mamy silny pik przy 13 Hz i słabe przy 26 i 39 Hz. Zgodnie z rozważaniami w następnym rozdziale, jest to widmo częstotliwościowe niesinusoidalnego przebiegu okresowego. Częstotliwość 13 Hz nosi nazwę częstotliwości podstawowej, a 26 i 39 Hz to, odpowiednio – druga i trzecia harmoniczna. Można też spodziewać się znalezienia pików przy dalszych wielokrotnościach 13 Hz, a więc 52; 65; 78 Hz itd. Nie widać ich na rys. 9.2, gdyż giną w białym szumie. Sygnał 13 Hz może być generowany przez np. trzyskrzydłową śrubę napędową łodzi podwodnej wykonującą 4,33 obr/s. Omawiany system jest podstawą sonaru *pasywnego* identyfikującego dźwięki podwodne na podstawie ich częstotliwości i zawartości harmonicznych.

Załóżmy, że piki są położone bardzo blisko siebie, jak to pokazano na rys. 9.3. Są dwa czynniki ograniczające możliwą do uzyskania rozdzielczość częstotliwościową, a więc minimalną odległość pików, przy której nie zlewają się one w jedną całość. Pierwszym czynnikiem jest długość (liczba punktów) DFT. Widmo częstotliwościowe wytworzone przez N-punktowe przekształcenie DFT zawiera $N/2 + 1$ próbek równomiernie rozmieszczonych między zerem a połową częstotliwości próbkowania. Aby można było rozdzielić dwie bardzo do siebie zbliżone częstotliwości, odstęp między próbkami musi być *mniejszy* niż odległość między dwoma pikami. Na przykład 512-punktowe DFT jest wystarczające do rozdzielenia piku na rys. 9.3, podczas gdy 128-punktowe nie wystarcza.

Drugi czynnik ograniczający rozdzielczość jest bardziej subtelnej natury. Wyobraźmy sobie sygnał utworzony przez zsumowanie dwóch sinusoid o bardzo niewielkiej różnicy częstotliwości. W krótkim fragmencie tego sygnału, powiedzmy w ciągu kilku okresów, przebieg wygląda bardzo podobnie do *pojedynczego* przebiegu sinusoidalnego. Im mniej różnią się od siebie częstotliwości dwóch przebiegów, tym dłuższy jest fragment przebiegu, po którym dopiero można stwierdzić, że jest w nim więcej niż jedna częstotliwość. Inaczej mówiąc, *długość* sygnału ogranicza rozdzielczość częstotliwościową. Ten czynnik ograniczający rozdzielczość różni się od pierwszego, gdyż *długość sygnału wejściowego* nie musi



Rys. 9.2. Przykład widma częstotliwościowego. W widmach zbieranych sygnałów pojawiają się elementy trzech rodzajów: 1) szum losowy, jak szum biały i szum 1/f, 2) sygnały zakłócające od sieci energetycznej, zasilaczy impulsowych, nadajników radiowych i TV, zjawiska mikrofonowania itd., 3) sygnały właściwe, pojawiające się zwykle jako częstotliwość podstawowa i harmoniczne. Na podanym przykładzie widma (tylko modułu) jest widocznych kilka z tych elementów



Rys. 9.3. Rozdzielczość widma częstotliwościowego. Im większą długość (liczbę punktów) ma przekształcenie DFT, tym lepsza jest zdolność rozdzielania położonych blisko siebie elementów widma. W przedstawionym przykładzie stosując 128-punktowe DFT, nie można rozdzielić dwóch pików, a przy użyciu 512-punktowego można

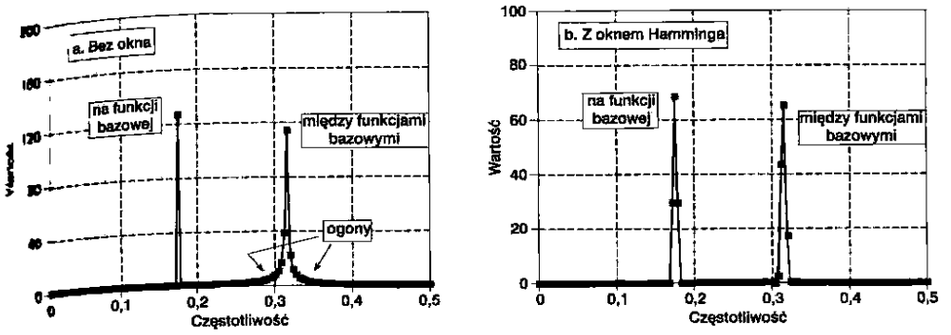
być tym samym, co *długość przekształcenia DFT*. Na przykład sygnał 256-punktowy może być dopełniony zerami, uzyskując długość 2048-punktową. Przekształcenie 2048-punktowe DFT daje widmo częstotliwościowe o 1025 próbkach. Dodane zera nie zmieniają kształtu widma, a wprowadzają tylko więcej próbek w dziedzinie częstotliwości. Mimo tego brak gęstego próbkowania, możliwość rozdzielenia bardzo bliskich pików poprawia się niewiele w stosunku do 256-punktowego DFT. Gdy przekształcenie DFT ma taka samą długość, jak sygnał wejściowy, to dwa czynniki w równym stopniu wpływają na ograniczenie rozdzielczości. Wkrótce wrócimy jeszcze do tej kwestii.

A teraz następne pytanie: Co się dzieje, gdy sygnał wejściowy zawiera sinusoidę o częstotliwości leżącej *pomiędzy* dwiema funkcjami bazowymi? Odpowiedź zawarto na rys. 9.4 a), na którym przedstawiono widmo częstotliwościowe sygnału złożonego z dwóch fal sinusoidalnych, z których jedna ma częstotliwość zgodną z funkcją bazową, a druga – częstotliwość znajdującą się *pomiędzy* dwiema z funkcji bazowych. Jak można się było spodziewać, pierwsza fala sinusoidalna jest reprezentowana przez pojedynczy punkt. Drugi pik trudniej zrozumieć. Ponieważ nie może być wyrażony przez pojedynczą próbkę, więc staje się pikiem z ogonami, które zwiększają minimalną odległość, przy jakiej piki są rozróżniane.

Jakie jest rozwiązanie tego problemu? Przed dokonaniem przekształcenia DFT pomnożymy sygnał przez okno Hamminga, jak to już poprzednio omówiono. Z rys. 9.4 b) wynika, że w wyniku użycia okna widmo zmieniło się na trzy sposoby. Po pierwsze, dwa piki stają się bardziej podobne do siebie. To dobrze. Po drugie, zmniejszyły się ogony. To też dobrze. Po trzecie, okno pogorszyło rozdzielczość widma w wyniku rozszerzenia pików. To źle. W języku stosowanym w DSP mówi się, że użycie okna prowadzi do kompromisu między *rozdzielczością* (szerokością piku) a *przeciekiem widmowym* (wielkością ogonów).

Aby bardziej szczegółowo wniknąć w teoretyczne aspekty tego problemu, wyobraźmy sobie nieskończenie długą dyskretną falę sinusoidalną o częstotliwości równej 0,1 częstotliwości próbkowania. Widmo częstotliwościowe tego sygnału jest nieskończenie wąskim pikiem, podczas gdy dla wszystkich innych częstotliwości ma wartość zero. Oczywiście ani ten sygnału, ani jego widma częstotliwościowego nie można wprowadzić do komputera ze względu na ich nieskończenie dużą lub nieskończenie małą długość. W celu ominięcia tej trudności zmieniamy sygnał dwoma sposobami. Oba te sposoby zniekształcają prawdziwe widmo częstotliwościowe.

W pierwszym sposobie *obcinamy* informację zawartą w sygnale, mnożąc go przez jakieś okno. Na przykład 256-punktowe *okno prostokątne* umożliwia zachowanie poprawnych wartości w 256 punktach, podczas gdy wszystkie inne próbki w nieskończenie długim sygnale zostają ustawione na wartość zero. Podobnie okno Hamminga *ukształtowałoby* zachowane próbki



Rys. 9.4. Przykład zastosowania okna w analizie widmowej. Na rys. a) przedstawiono widmo częstotliwościowe (tylko moduł) sygnału złożonego z dwóch fal sinusoidalnych. Jedna z sinusoid ma częstotliwość dokładnie równą częstotliwości funkcji bazowej, więc jest reprezentowana przez pojedynczą próbkę. Druga sinusoida ma częstotliwość leżącą pomiędzy dwiema funkcjami bazowymi, co powoduje występowanie ogonów na piku. Na rys. b) przedstawiono widmo częstotliwościowe tego samego sygnału, lecz z oknem Hamminga zastosowanym przed dokonaniem przekształcenia DFT. Okno powoduje, że piki stają się do siebie podobne, ogony zostają zmniejszone, lecz piki są szersze

nadto ustawiając na zero wszystkie punkty poza oknem. Sygnał nadal jest nieskończenie długi, lecz tylko skończona liczba próbek ma wartości niezerowe.

Jaki wpływ na sygnał w dziedzinie częstotliwości ma okienkowanie? Zgodnie z rozważaniami podanymi w rozdziale 10, operacja *mnożenia* dwóch sygnałów w dziedzinie czasu odpowiada operacji *splotu* w dziedzinie częstotliwości. Ponieważ widmem oryginalnym jest nieskończenie wąski pik (tj. funkcja delta), więc widmo sygnału okienkowanego jest widmem okna przesuniętym do położenia piku. Na rys. 9.5 pokazano, jak wygląda pik widma po zastosowaniu czterech różnych okien. (Jeśli chcecie odświeżyć swoją wiedzę o decybelach, zajrzyjcie do rozdziału 14). Widmo na rys. 9.5 a) jest wynikiem zastosowania okna prostokątnego, a na rys. 9.5 b), c) – dwóch popularnych okien, Hamminga i Blackmana (informację o nich podano we wzorach 16.1 i 16.2 oraz na rys. 16.2 a).

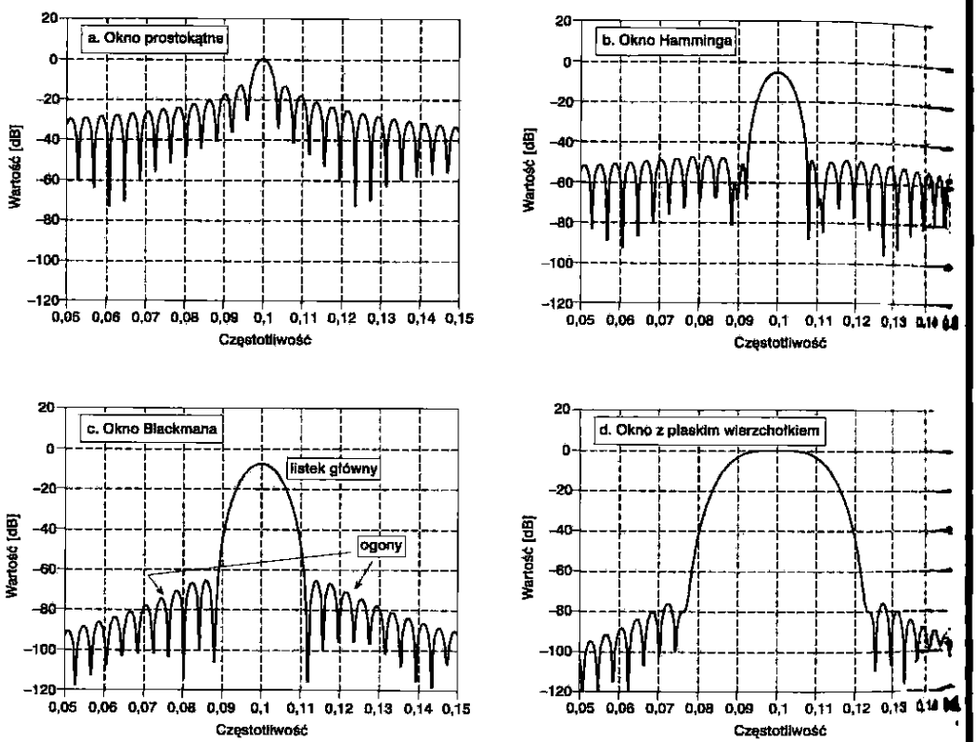
Z rys. 9.5 wynika, że te wszystkie okna zniekształciły widmo oryginalne przez rozszerzenie piku i dodanie ogonów złożonych z licznych listków bocznych. Jest to nieunikniony rezultat wykorzystania tylko części oryginalnego widma w dziedzinie czasu. Tu można porównać właściwości trzech okien. Okno Blackmana daje najszerszy listek główny (to jest źle), lecz najmniejsze ogony (dobrze). Okno prostokątne daje najwęższy listek główny (dobrze), lecz największe ogony (źle). Okno Hamminga ma właściwości pośrednie między tymi dwoma oknami.

Proszę zauważyć, że widma częstotliwościowe są krzywymi ciągłymi, a nie próbkami dyskretnymi. Po okienkowaniu sygnał w dziedzinie czasu jest nadal nieskończenie długi, chociaż większość próbek jest zerowych. To znaczy, że widmo częstotliwościowe składa się z $\infty/2 + 1$ próbek pomiędzy 0 i 0,5, a więc tak samo, jak linia ciągła.

To prowadzi do drugiego sposobu potrzebnego do takiego zmodyfikowania sygnału w dziedzinie czasu, aby mógł być wprowadzony do komputera. Metoda polega na *wyborze N punktów z sygnału*. W tych *N* punktach muszą być zawarte wszystkie punkty niezerowe zidentyfikowane przez okno, lecz może być w nich zawarta także dowolna liczba zer. Daje to efekt *próbkowania* ciągłej krzywej widma częstotliwościowego. Jeśli na przykład wybrano wartość *N* równą 1024, to krzywa ciągła widma będzie próbkowana 513 razy między 0 i 0,5. Jeśli wybrano *N* znacznie większe niż długość okna, to próbki w dziedzinie częstotliwości będą wystarczająco bliskie siebie, żeby piki i wgłębienia krzywej ciągłej zostały zachowane w nowym widmie. Jeśli wybrano *N* równe długości okna, to mniejsza liczba próbek w widmie powoduje, że regularny obraz pików i wgłębień zmienia się w nieregularne ogony, zależne od tego, gdzie trafiły próbki.

To wyjaśnia, dlaczego dwa piki na rys. 9.4 a) nie wyglądają podobnie. Każdy pik na rys. 9.4 a) jest *spróbkowaniem* krzywej z rys. 9.5 a). Istnienie lub brak ogonów zależy od usytuowania miejsca pobrania próbek w stosunku do położenia pików i wgłębień. Jeśli fala sinusoidalna jest dokładnie zgodna z funkcją bazową, to próbki następują dokładnie we wgłębieniach, eliminując ogony. Jeśli fala sinusoidalna mieści się między dwiema funkcjami bazowymi, to próbki następują gdzieś między pikami i wgłębieniami, dając w rezultacie ogony różnych kształtów.

W ten sposób dochodzimy do **okna o płaskim wierzchołku**, przedstawionego na rys. 9.5 d). W niektórych zastosowaniach jest konieczny bardzo dokładny pomiar *amplitudy* piku widmowego. Ponieważ widmo częstotliwościowe DFT jest tworzone z próbek, więc nie ma żadnej gwarancji, że próbka będzie miała miejsce dokładnie w szczycie piku. Jest więcej niż prawdopodobne, że najbliższa próbka będzie nieco przesunięta w stosunku do szczytu, dając wartość mniejszą niż prawdziwa amplituda. Rozwiązaniem tej kwestii jest zastosowanie okna wytwarzającego plik widmowy o *płaskim wierzchołku*, dzięki czemu jest pewne, że jedna lub więcej próbek będzie miała zawsze prawidłową wartość piku. Karą za to, jak widać na rys. 9.5 d), jest bardzo duża szerokość głównego listka powodująca słabą rozdzielczość częstotliwościową.



Rys. 9.5. Szczegółowy obraz piku widmowego po zastosowaniu różnych okien. Każdy pik w widmie częstotliwościowym jest tu środkowym listkiem otoczonym przez ogony złożone z listków bocznych. Przez zmianę kształtu okna wielkość listków bocznych może ulec zmniejszeniu kosztem rozszerzenia listka głównego. Okno prostokątne (rys. a) daje najwęższy listek główny, lecz największe listki boczne. Okna Hamminga (rys. b) i Blackmana (rys. c) dają mniejsze listki boczne kosztem szerszego listka głównego. Okno o płaskim wierzchołku (rys. d) jest stosowane wówczas, gdy amplituda piku musi być dokładnie zmierzona. Podane krzywe dotyczą okien 255-punktowych; dłuższe okna dają proporcjonalnie węższe piki

Okazuje się, że pożądany kształt okna o płaskim wierzchołku jest zgodny z kształtem odpowiedzi impulsowej (tzw. jądro) filtru dolnoprzepustowego. Teoretyczne przyczyny tego faktu rozważymy w dalszych rozdziałach. Na razie podajemy praktyczne wskazówki o stosowaniu tej metody. W rozdziale 16 znajdziecie omówienie filtru dolnoprzepustowego zwanego filtrem z *okienkowaniem funkcją sinc*. Wzór 16.4 opisuje, jak tworzyć odpowiedź impulsową (którą chcemy zastosować jako okno), a na rys. 16.4 a) pokazano typowy kształt tej krzywej. Aby móc skorzystać z tego wzoru, musimy znać wartości dwóch parametrów: M i f_c. Oblicza się je z zależności: $M = N - 2$ oraz $f_c = s/N$, gdzie N jest długością zastosowanego przekształcenia DFT, a s – liczbą próbek, które mają znaleźć się na płaskiej części piku (zwykle od 3 do 5). W tablicy 16.1 podano program do obliczania odpowiedzi impulsowej filtru (czyli naszego okna). Program zawiera dwa istotne elementy – stałą normalizacyjną K i sposób uniknięcia błędu dzielenia przez zero w próbce środkowej. Stosując tę metodę, trzeba pamiętać, że stała wartość równa *jedności* w dziedzinie czasu wytworzy pik o amplitudzie *jeden* w dziedzinie częstotliwości. Jednak sinusoida o amplitudzie *jeden* w dziedzinie czasu wytworzy pik widmowy o amplitudzie tylko *jedna druga* (1/2). To zagadnienie jest omówione w poprzednim rozdziale, w podrozdziale: Synteza, obliczanie odwrotnej transformaty DFT.

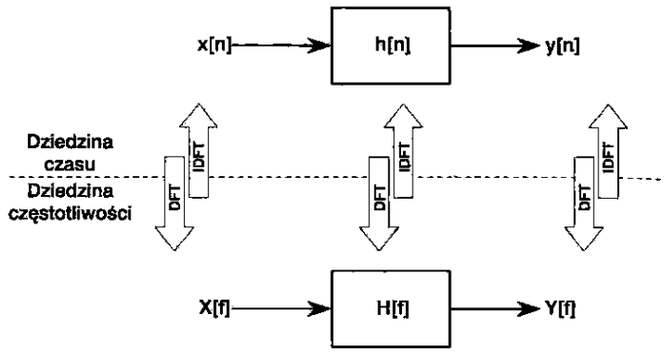
Odpowiedź częstotliwościowa systemów

W *dziedzinie czasu* systemy są analizowane przy użyciu splotu. Podobne analizy można przeprowadzać w *dziedzinie częstotliwości*. Każdy sygnał wejściowy może, przy użyciu przekształcenia Fouriera, być przedstawiony jako grupa fal kosinusoidalnych, każda o określonej amplitudzie i przesunięciu fazowym. Można też stosować przekształcenie DFT do przedstawienia sygnału wyjściowego w podobnej postaci. Wynika stąd, że każdy system liniowy może być *całkowicie* opisany przez zmiany amplitudy i fazy, jakie wywołuje w przechodzących przezeń falach kosinusoidalnych. Ta informacja jest nazywana **odpowiedzią częstotliwościową**. Ponieważ zarówno odpowiedź impulsowa, jak i odpowiedź częstotliwościowa zawierają pełną informację o systemie, więc musi być odpowiedniość jeden do jednego między obiema tymi charakterystykami. Mając daną jedną z odpowiedzi, można obliczyć drugą. Zależność między odpowiedzią impulsową i odpowiedzią częstotliwościową jest jedną z podstaw przetwarzania sygnałów. *Odpowiedź częstotliwościowa systemu jest przekształceniem Fouriera odpowiedzi impulsowej tego systemu*. Te zależności są zilustrowane na rys. 9.6.

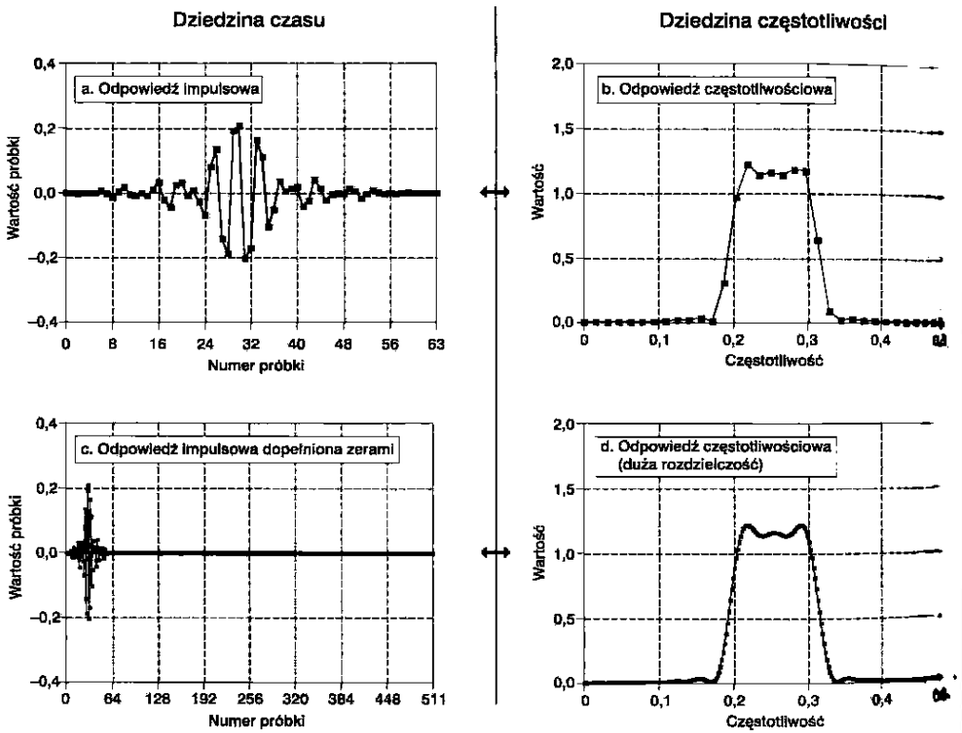
Utrzymując zasady przyjęte w DSP, odpowiedzi impulsowe oznacza się małymi literami, a odpowiadające im odpowiedzi częstotliwościowe – dużymi. Jako że $h[\ ]$ jest powszechnie stosowanym oznaczeniem odpowiedzi impulsowej, więc $H[\ ]$ stosuje się do odpowiedzi częstotliwościowej. W dziedzinie czasu systemy są opisywane przez splot $x[n] * h[n] = y[n]$. W dziedzinie częstotliwości widmo wejściowe jest *mnożone* przez odpowiedź częstotliwościową, dając w wyniku widmo wyjściowe, zgodnie z zależnością: $X[f] \times H[f] = Y[f]$. A zatem *splot* w dziedzinie czasu odpowiada *mnożeniu* w dziedzinie częstotliwości.

Na rys. 9.7 przedstawiono przykład zastosowania DFT do przetworzenia odpowiedzi impulsowej systemu na jego odpowiedź częstotliwościową. Na rys. 9.7 a) pokazano odpowiedź impulsową systemu. Oglądanie tej krzywej nie daje najmniejszego nawet pojęcia o tym, jak działa system. Wykonanie 64-punktowego przekształcenia DFT tej odpowiedzi impulsowej daje odpowiedź częstotliwościową systemu, przedstawioną na rys. 9.6 b). Teraz funkcja tego systemu staje się oczywista: przepuszcza on częstotliwości między 0,2 i 0,3 i odrzuca wszystkie pozostałe. Jest to więc filtr pasmowoprzepustowy. Może też zbadać *fazę* odpowiedzi częstotliwościowej, jednak charakterystyka fazowa jest *trudniejsza* do interpretacji i *mniej* interesująca. Będzie o niej mowa w dalszych rozdziałach.

Charakterystyka na rys. 9.7 b) jest bardzo poszarpana z powodu małej liczby próbek wyznaczających krzywą. Tę sytuację można poprawić przez **dopełnianie** odpowiedzi impulsowej zerami, przed przekształceniem DFT. Na przykład wprowadzenie zer w taki sposób, aby wydłużyć odpowiedź impulsową do 512 próbek (rys. 9.7 c), daje charakterystykę częstotliwościową o dużej rozdzielczości (rys. 9.7 d).



Rys. 9.6. Porównanie działania systemu w dziedzinie czasu i częstotliwości. W dziedzinie czasu sygnał wejściowy jest splatany z odpowiedzią impulsową, dając w wyniku sygnał wyjściowy, tzn. $x[n] * h[n] = y[n]$. W dziedzinie częstotliwości widmo wejściowe jest mnożone przez odpowiedź częstotliwościową, co w rezultacie daje widmo wyjściowe, tzn. $X[f] \times H[f] = Y[f]$. Wzajemne zależności między sygnałami w obu dziedzinach są wyrażone przez przekształcenia DFT i odwrotne DFT (oznaczone na rysunku jako IDFT)



Rys. 9.7. Wyznaczanie charakterystyki częstotliwościowej na podstawie odpowiedzi impulsowej. Przy zastosowaniu przekształcenia DFT odpowiedź impulsowa systemu (rys. a) może być przetworzona w jego odpowiedź częstotliwościową (rys. b). Dopełniając odpowiedź impulsową zerami (rys. c), możemy osiągnąć większą rozdzielczość odpowiedzi częstotliwościowej (rys. d). W tym przykładzie pokazano tylko moduł odpowiedzi częstotliwościowej, omówienie fazy pozostawiamy do następnego rozdziału

Jak dużą rozdzielczość odpowiedzi częstotliwościowej można uzyskać? Odpowiedź brzmi: *nieskończenie* dużą, jeśli jesteśmy skłonni dopełnić odpowiedź impulsową *nieskończoną* liczbą zer. Inaczej mówiąc, nie ma innego czynnika ograniczającego rozdzielczość częstotliwościową, prócz długości przekształcenia DFT. To prowadzi do ważnych wniosków. Chociaż odpowiedź impulsowa jest sygnałem *dyskretnym*, to odpowiadająca jej odpowiedź częstotliwościowa ma charakter *ciągły*. Przekształcenie N-punktowe DFT odpowiedzi impulsowej dostarcza $N/2 + 1$ *próbek* w krzywej ciągłej. Jeśli wydłużymy DFT, to rozdzielczość się polepszy i dowiemy się więcej o kształcie krzywej ciągłej. Pamiętajmy, że odpowiedź częstotliwościowa wyraża zmiany fazy i amplitudy, jakim ulega fala kosinusoidalna, przechodząc przez system. Ponieważ sygnał wejściowy może zawierać wszelkie częstotliwości między 0 a 0,5, więc odpowiedź częstotliwościowa systemu w tym zakresie *musi* być krzywą ciągłą.

Można to lepiej zrozumieć, wprowadzając inne przekształcenie z grupy transformat fourierowskich – **dyskretne w czasie przekształcenie Fouriera (DTFT** – *Discrete Time Fourier Transform*). Rozważmy sygnał złożony z N próbek poddawany N-punktowemu przekształceniu Fouriera, dającemu $N/2 + 1$ próbek charakterystyki w dziedzinie częstotliwości. Pamiętamy z poprzedniego rozdziału, że przekształcenie DFT wymaga, aby sygnał w dziedzinie czasu był *nieskończenie długi* i *okresowy*. Tak więc N punktów powinno być powtarzanych wielokrotnie od minus do plus nieskończoności. Rozważmy teraz, co się dzieje, gdy sygnał w dziedzinie czasu dopełniamy ciągle rosnącą liczbą zer, aby uzyskać coraz to gęściejsze próbkowanie w dziedzinie częstotliwości. Dodawanie zer *wydłuża* przebieg w dziedzinie czasu, przy jednoczesnym *zmniejszaniu odstępów między próbkami* w dziedzinie częstotliwości.

Rozważmy teraz skrajną sytuację – dodanie *nieskończonej* liczby zer do sygnału w dziedzinie czasu. Wtedy, pod dwoma względami, powstaje inna sytuacja. Po pierwsze, sygnał w dziedzinie czasu ma teraz okres nieskończenie długi. Tak więc stał się sygnałem *nieokresowym*. Po drugie, odstępy między próbkami w dziedzinie częstotliwości stają się nieskończenie małe. A zatem ten sygnał jest teraz sygnałem *ciągłym*. To jest przekształcenie DTFT, procedura zmieniająca dyskretny sygnał nieokresowy w dziedzinie czasu na sygnał będący krzywą ciągłą w dziedzinie częstotliwości. Odpowiedź częstotliwościową systemu określa się matematycznie, wyznaczając DTFT jego odpowiedzi impulsowej. Ponieważ nie da się tego zrobić w komputerze, więc stosuje się DFT do obliczania *próbek* prawdziwej odpowiedzi częstotliwościowej. Jest różnica między tym, co wykonuje komputer (DFT), a tym, co realizujemy przy użyciu wzorów matematycznych (DTFT).

Splot przez dziedzinę częstotliwości

Powiedzmy, że jesteśmy niechętni stosowaniu splotu. Co wtedy możemy zrobić, mając dany sygnał wejściowy i odpowiedź impulsową i chcąc znaleźć wynikowy sygnał wyjściowy? Odpowiedź znajdziecie na rys. 9.8 – trzeba oba sygnały poddać przekształceniu do dziedziny częstotliwości, pomnożyć je przez siebie i następnie przekształcić uzyskany wynik z powrotem do dziedziny czasu. Ta procedura zastępuje jedną operację splotu dwoma przekształceniami DFT, mnożeniem i odwrotnym DFT. Mimo że operacje pośrednie są w tej procedurze odmienne, to uzyskany wynik jest *identyczny* ze standardowym algorytmem splotu.

Czy ktokolwiek może aż tak bardzo nienawidzić splotu, aby podejmować tę kłopotliwą procedurę? Otóż tak. Splotu unika się z dwóch przyczyn. Po pierwsze, splot jest operacją *matematycznie* trudną do realizacji. Załóżmy na przykład, że mamy daną odpowiedź impulsową systemu i sygnał na jego wyjściu. Jak obliczyć sygnał wejściowy? Ta operacja jest nazywana **rozplotem** (*deconvolution*), który jest zupełnie niemożliwy do zrozumienia w dziedzinie czasu. Jednak rozplot w dziedzinie częstotliwości może być przeprowadzony jako proste *dzielenie*, operacja odwrotna do mnożenia. Dziedzina częstotliwości staje się godna uwagi, jeśli tylko zawiłości przekształcenia Fouriera są mniejsze niż operacji splotu. Nie jest to kwestia tego, którą z tych operacji bardziej lubimy, lecz tego, której mniej nienawidzimy.

Drugim powodem unikania splotu jest *szybkość obliczeń*. Przypuśćmy, że projektujemy filtr cyfrowy o odpowiedzi impulsowej zawierającej 512 próbek. W komputerze osobistym z procesorem 200 MHz i arytmetyką zmiennoprzecinkową obliczenie każdej próbki – przy zastosowaniu standardowego algorytmu splotu – wymaga czasu 1 ms. A zatem wydajność systemu jest tylko 1000 próbek/s. Jest to 40-krotnie zbyt wolno, jak na potrzeby aparatury audio hi-fi oraz 10 000 razy zbyt wolno dla aparatury wideo o telewizyjnej jakości obrazu.

Standardowy algorytm splotu jest wolny, ponieważ wymaga wielkiej liczby operacji mnożenia i dodawania. Proste przeniesienie problemu, przez DFT, w dziedzinę częstotliwości niczego wcale nie jest pomocne. Do obliczeń przekształceń DFT jest koniecznych tyle samo obliczeń, co do bezpośredniego obliczenia splotu. Przełom w tej kwestii nastąpił we wczesnych latach 60., gdy opracowano *szybkie przekształcenie Fouriera* (FFT).

FFT jest pomysłowym algorytmem do szybkiego obliczania transformaty DFT. Przy zastosowaniu FFT operacja splotu realizowana przez mnożenie w dziedzinie częstotliwości może być setki razy szybsza niż splot konwencjonalny. Dlatego właśnie przekształcenie FFT oraz przetwarzanie sygnałów w dziedzinie częstotliwości wywołały takie zainteresowanie. Przekształcenie FFT jest omówione w rozdziale 12, a metoda splotowa FFT – w rozdziale 18. Na razie skupimy uwagę na splataniu sygnałów przez mnożenie w dziedzinie częstotliwości.

Na początku trzeba zdefiniować sposób mnożenia dwóch sygnałów w dziedzinie częstotliwości, a więc wyjaśnić, co znaczy zapis: $X[f] \times H[f] = Y[f]$. We współrzędnych biegunowych moduły są mnożone $Mag\ Y[f] = Mag\ X[f] \times Mag\ H[f]$, a fazy dodawane $Phase\ Y[f] = Phase\ X[f] + Phase\ H[f]$. W celu zrozumienia tych zasad wyobraźmy sobie wchodzącą do systemu falę kosinusoidalną o pewnej amplitudzie i fazie. Również sygnał wyjściowy jest falą kosinusoidalną o pewnej amplitudzie i fazie. Odpowiedź częstotliwościowa we współrzędnych biegunowych opisuje bezpośrednio zależności między tymi dwiema amplitudami i dwiema fazami.

Gdy mnożenie w dziedzinie częstotliwości jest wykonywane we *współrzędnych prostokątnych* to występują powiązania wzajemne między częścią rzeczywista i urojoną. Na przykład fala kosinusoidalna wchodząca do systemu może dawać na wyjściu zarówno falę sinusoidalną, jak i falę sinusoidalną. Mnożenie w dziedzinie częstotliwości sygnałów zapisanych we współrzędnych prostokątnych wykonuje się zgodnie z wzorami:

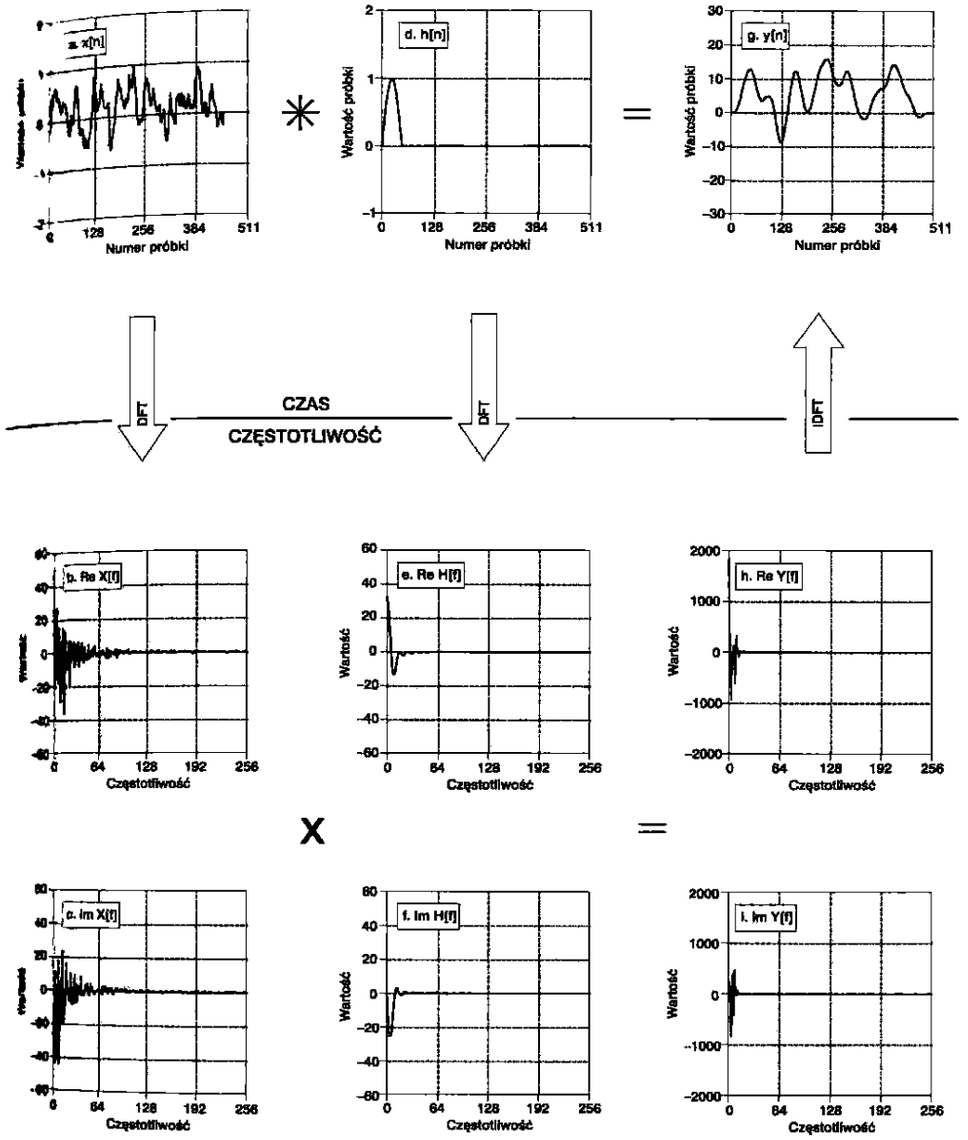
$$Re\,Y[f] = Re\,X[f]\,Re\,H[f] - Im\,X[f]\,Im\,H[f]$$
$$Im\,Y[f] = Im\,X[f]\,Re\,H[f] + Re\,X[f]\,Im\,H[f]$$

Wzór 9.1. Mnożenie w dziedzinie częstotliwości sygnałów zapisanych we współrzędnych biegunowych: Y [f] = X[f] × H[f]

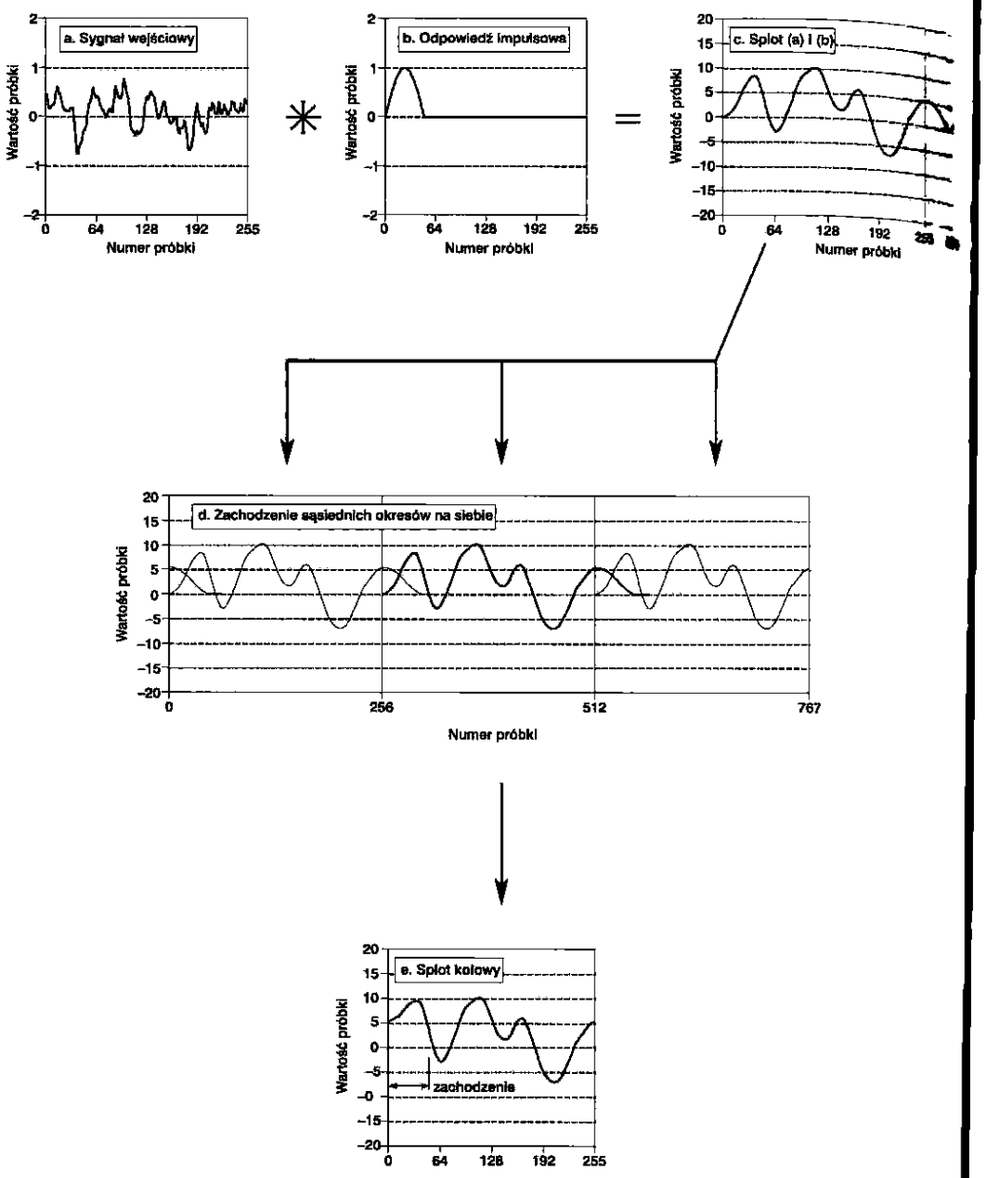
Proszę zwrócić szczególną uwagę na zrozumienie mnożenia we *współrzędnych biegunowych* oraz przechodzenia fal kosinusoidalnych przez system. Następnie trzeba po prostu zaakceptować fakt, że wzory są bardziej rozbudowane w przypadku przeprowadzania operacji we współrzędnych prostokątnych. Spójrzmy na przykład na *dzielenie* jednego sygnału w dziedzinie częstotliwości przez drugi. We współrzędnych biegunowych dzielenie tych sygnałów realizuje się przez operacje odwrotne do stosowanych w mnożeniu. W celu obliczenia: $H[f] = Y[f] / X[f]$, trzeba podzielić moduły i odjąć fazy, tj. $Mag\ H[f] = Mag\ Y[f] / Mag\ X[f]$ oraz $Phase\ H[f] = Phase\ Y[f] - Phase\ X[f]$. We współrzędnych prostokątnych konieczne jest skorzystanie z wzorów:

$$Re\,H[f] = \frac{Re\,Y[f]\,Re\,X[f] + Im\,Y[f]\,Im\,X[f]}{Re\,X[f]^2 + Im\,X[f]^2}$$
$$Im\,H[f] = \frac{Im\,Y[f]\,Re\,X[f] - Re\,Y[f]\,Im\,X[f]}{Re\,X[f]^2 + Im\,X[f]^2}$$

Wzór 9.2. Dzielenie w dziedzinie częstotliwości sygnałów zapisanych we współrzędnych prostokątnych: H[f] = Y[f] / X[f]



Rys. 9.8. Splot w dziedzinie częstotliwości. W dziedzinie czasu x[n] jest splatane z h[n], dając w wyniku y[n], jak to pokazano na rys. a), d), g). Tę samą procedurę można zastosować w dziedzinie częstotliwości. Zastosowano DFT w celu znalezienia widma częstotliwościowego sygnału wejściowego (rys. b, c) i odpowiedź częstotliwościową systemu (rys. e, f). Pomnożenie tych dwóch sygnałów w dziedzinie częstotliwości daje w rezultacie widmo częstotliwościowe sygnału wyjściowego (rys. h, i). Następnie zastosowano odwrotne przekształcenie DFT, aby znaleźć sygnał wyjściowy (rys. g). Na rysunku przez IDFT oznaczono odwrotne przekształcenie DFT



Rys. 9.9. Splot kołowy. Sygnał 256-próbkowy (rys. a) poddany splotowi z 51-próbkową odpowiedzią impulsową (rys. b) daje w rezultacie 306-próbkowy sygnał wyjściowy (rys. c). Jeśli ten splot jest realizowany w dziedzinie częstotliwości przy użyciu 256-punktowego przekształcenia DFT, to 306 punktów prawidłowego splotu nie może się zmieścić w dostępnych 256 próbkach. Jak pokazano na rys. d), próbki od 256 do 305 sygnału wyjściowego są wpychane do następnego okresu z prawej strony i tam dodawane do początkowego fragmentu sygnału w następnym okresie. Na rys. e przedstawiono pojedynczy okres sygnału uzyskanego jako wynik

Powróćmy teraz do splotu w dziedzinie częstotliwości. Zauważyliście, być może, że trochę oszukaliśmy na rys. 9.8. Przypomnijmy sobie, że splot sygnału o N punktach z M-punktową odpowiedzią impulsową daje w wyniku $(N+M-1)$ – punktowy sygnał wyjściowy. Oszukaliśmy, przyjmując sygnał wejściowy, w którym ostatnia część zawiera same *zera* i umożliwiając powstanie takiego wydłużenia przebiegu. W rezultacie przebieg na rys. 9.8 a) zawiera 453 próbek niezerowych, a na rys. 9.8 b) – 60 takich próbek. Wynika stąd, że splot tych dwóch przebiegów (rys. 9.8 c) dobrze mieści się w przewidzianych 512 punktach.

Rozważmy teraz przypadek bardziej ogólny przedstawiony na rys. 9.9. Sygnał wejściowy (rys. 9.9 a) ma długość 256 punktów, podczas gdy odpowiedź impulsowa zawiera 51 punktów niezerowych (rys. 9.9 b). Stąd wynika, że splot tych sygnałów ma długość 306 punktów (rys. 9.9 c). Problem polega na tym, że jeśli do realizacji splotu stosujemy mnożenie w dziedzinie częstotliwości, to w sygnale wyjściowym jest *dopuszczalnych* tylko 256 próbek. Zatem stosuje się 256-punktowe przekształcenie DFT, aby przenieść przebiegi na rys. 9.9 a), b) w dziedzinę częstotliwości. Po wymnożeniu używa się 256-punktowego odwrotnego przekształcenia DFT w celu wyznaczenia sygnału wyjściowego. W jaki sposób ścisnąć 306 wartości prawidłowego sygnału do 256 punktów udostępnionych przez algorytm w dziedzinie częstotliwości? Odpowiedź brzmi: nie da się tego zrobić. Te 256 punktów jest zniekształconą wersją prawidłowego sygnału. Ten proces nosi nazwę **splotu kołowego**. Jest on ważny z tego względu, że chcemy go *uniknąć*.

W celu zrozumienia splotu kołowego przypomnijmy sobie, że w N-punktowym przekształceniu DFT sygnał w dziedzinie czasu jest widziany jako nieskończenie długi przebieg okresowy o N próbkach na okres. Na rys. 9.9 d) pokazano, jak w tym przykładzie w trzech okresach sygnał wyjściowy jest widziany przez DFT. Ponieważ $N = 256$, więc każdy okres składa się z 256 punktów: 0–255, 256–511 i 512–767. Wykonanie splotu w dziedzinie częstotliwości powoduje próbę umieszczenia 306-punktowego *prawidłowego sygnału wyjściowego* (rys. 9.9 c) w każdym z 256-punktowych okresów. Powoduje to wepchnięcie 49 próbek do sąsiedniego okresu z prawej strony, gdzie zachodzą one na próbki znajdujące się tam w sposób uprawniony. Te zachodzące na siebie fragmenty sumują się, dając w wyniku w każdym z okresów *splot kołowy*, jak to pokazano na rys. 9.9 e).

Jak już się rozumie naturę splotu kołowego, to dość łatwo można go uniknąć. Trzeba po prostu każdy ze splatanych sygnałów dopełnić o wystarczającą liczbą zer, aby stworzyć dla sygnału wyjściowego miejsce do operowania $N+M-1$ punktami w prawidłowym splocie. Można na przykład sygnały na rys. 9.9 a), b) dopełnić zerami tak, aby uzyskały długości po 512 punktów, umożliwiając zastosowanie 512-punktowego przekształcenia DFT. Po realizacji splotu w dziedzinie częstotliwości sygnał wyjściowy zawierałby 306 próbek niezerowych plus 206 próbek o wartości zero. W rozdziale 18 szczegółowo opisano tę procedurę.

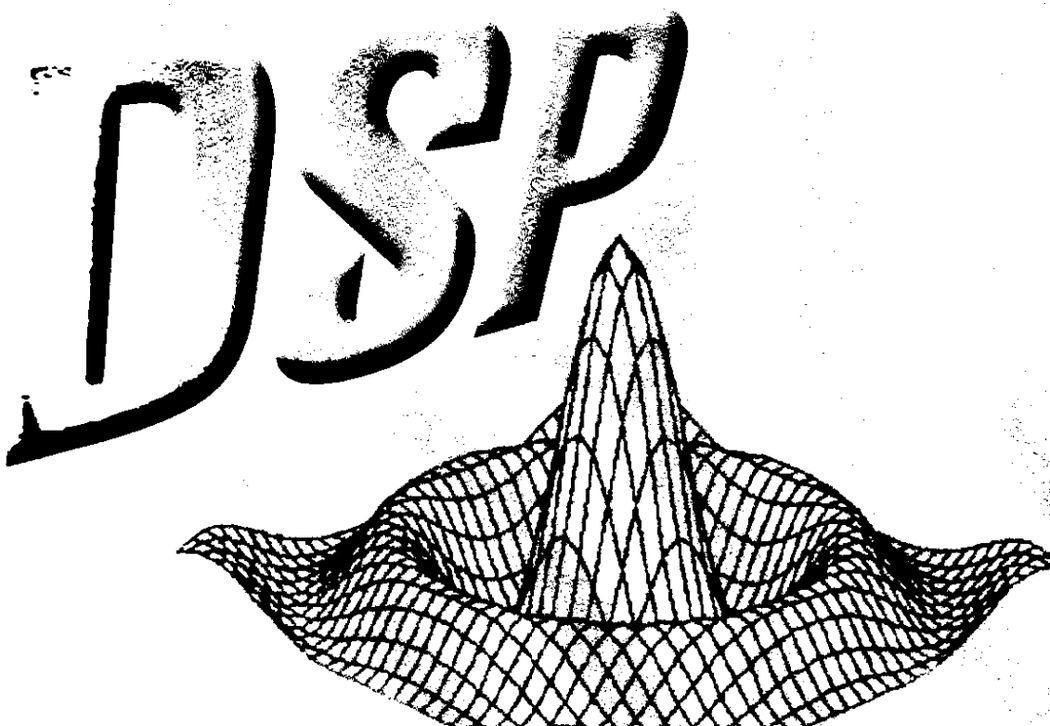
Dlaczego jest ona nazywana *splotem kołowym*? Wróćmy do rys. 9.9 d) i przyjrzyjmy się środkowemu okresowi (próbki od 256 do 511). Ponieważ wszystkie okresy są takie same, więc część sygnału, która wypływa z tego okresu *w prawo*, jest taka sama, jak wpływająca do niego z *lewej* strony. Jeśli rozpatrujemy tylko pojedynczy okres, taki jak na rys. 9.9 e), to *okazuje się*, że prawa strona sygnału jest w pewnym sensie *związana* z jego lewą stroną. Wyobraźmy sobie węża kąsającego własny ogon; próbka 255 jest zlokalizowana obok próbki 0, tak samo jak próbka 100 obok próbki 101. Gdy część sygnału wychodzi z tego okresu w prawo, to w magiczny sposób znowu się pojawia po lewej stronie. Tak więc N-punktowy sygnał w dziedzinie czasu zachowuje się tak, jakby był *kołowy*.

Poprzednio postawiliśmy pytanie, czy ma istotne znaczenie sposób widzenia DFT w dziedzinie czasu – jako sygnału N-punktowego, a nie jako nieskończenie długiego sygnału okresowego o okresie N. Splot kołowy jest przykładem sytuacji, gdy *ma to znaczenie*. Jeśli sygnał w dziedzinie czasu jest rozumiany jako *okresowy*, to zniekształcenia występujące w splocie

kołowym można po prostu wyjaśnić jako rozciąganie się sygnału z jednego okresu do następnego. Gdy się zaś rozważa tylko N-punktowy sygnał w dziedzinie czasu, to dochodzi się do dość dziwacznego wniosku. A mianowicie do konkluzji, że splot w dziedzinie częstotliwości działa tak, jakby przebieg w dziedzinie czasu zwijał się w jakiś kołowy pierścień z próbką 0 umieszczoną obok próbki $N-1$.

10

Właściwości przekształcenia Fouriera

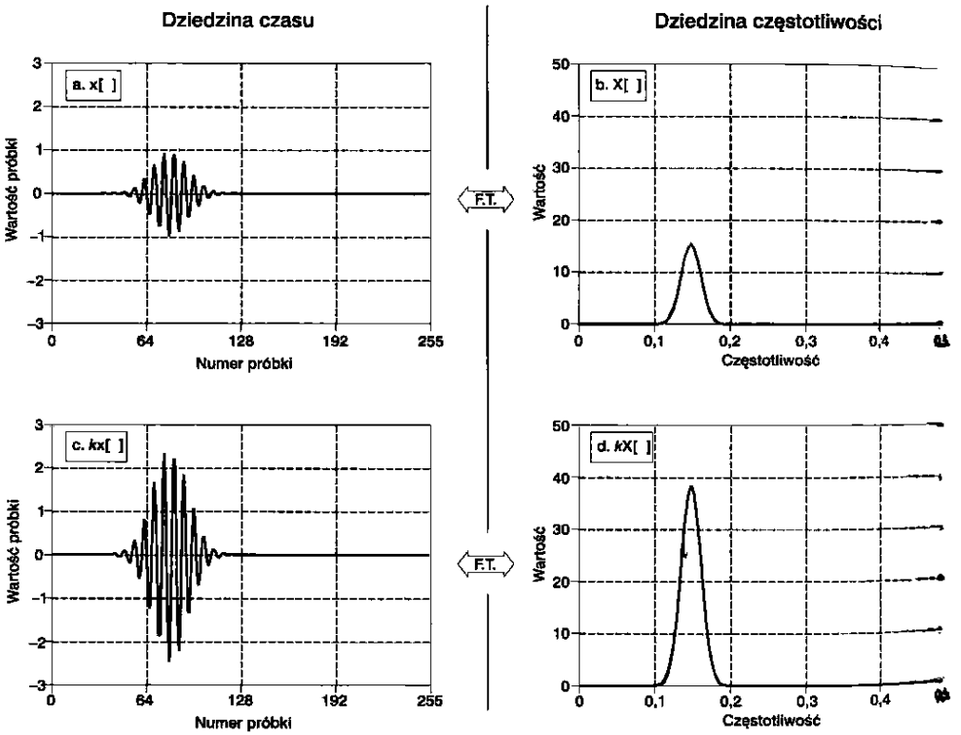


Sygnały mogą być reprezentowane alternatywnie – albo w dziedzinie czasu, albo w dziedzinie częstotliwości. Przekształcenie Fouriera wyraża matematyczną zależność między tymi dwiema reprezentacjami sygnału. Jeśli sygnał jest modyfikowany w jednej dziedzinie, to będzie również zmieniony w drugiej, chociaż na ogół nie w ten sam sposób. W poprzednim rozdziale wykazano na przykład, że *splatanie* sygnałów w dziedzinie czasu jest równoważne z *mnożeniem* ich widm częstotliwościowych. Inne operacje matematyczne, takie jak dodawanie, skalowanie, przesuwanie, też mają swoje odpowiedniki w drugiej dziedzinie. Te zależności nazywamy *właściwościami* przekształcenia Fouriera opisującymi, jak zmiana matematyczna w jednej dziedzinie wpływa na zmianę w drugiej.

Liniowość przekształcenia Fouriera

Przekształcenie Fouriera jest *liniowe*, charakteryzuje się więc *homogenicznością* i *addytywnością*. Obowiązuje to dla wszystkich czterech przekształceń z rodziny przekształceń Fouriera (przekształcenie Fouriera, szereg Fouriera, DFT i DTFT).

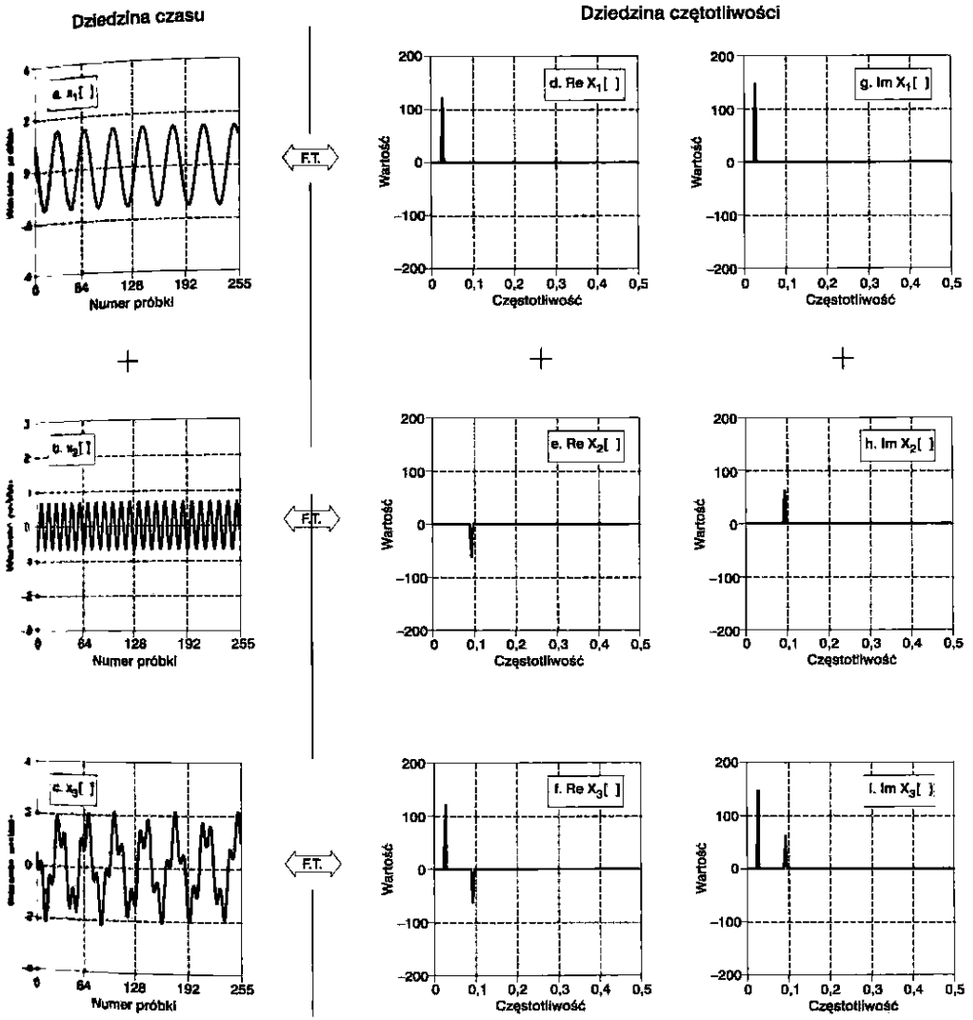
Na rys. 10.1 wyjaśniono na przykładzie, na czym polega właściwość homogeniczności przekształcenia Fouriera. Na rys. 10.1 a) przedstawiono jakiś dowolny sygnał w dziedzinie czasu, a na rys. 10.1 b) – odpowiadające mu widmo w dziedzinie częstotliwości. Oznaczmy te dwa sygnały, odpowiednio, jako $x[\ ]$ i $X[\ ]$. *Homogeniczność* oznacza, że zmiana amplitudy w jednej dziedzinie wytwarza taką samą zmianę w drugiej dziedzinie. Jest to zrozumiałe



Rys. 10.1. Homogeniczność przekształcenia Fouriera. Jeśli amplituda zmienia się w jednej dziedzinie, to tak samo zmienia się w drugiej. Inaczej mówiąc, skalowanie w jednej dziedzinie odpowiada skalowaniu w drugiej. Przez F.T. oznaczono na rysunku przekształcenie Fouriera

łe intuicyjnie; gdy zmienia się amplituda przebiegu w dziedzinie czasu, to o równoważną wartość powinny zmienić się amplitudy fal sinusoidalnych i kosinusoidalnych tworzących przebieg w dziedzinie częstotliwości.

Ujmując tę kwestię matematycznie, można stwierdzić, że jeśli $x[\ ]$ i $X[\ ]$ są parą transformat Fouriera, to $kx[\ ]$ i $kX[\ ]$ są też parą transformat Fouriera dla dowolnej stałej k. Jeśli przebieg w dziedzinie częstotliwości jest wyrażony we współrzędnych *prostokątnych*, to wyrażenie $kX[\ ]$ oznacza, że zarówno część rzeczywista, jak i urojona są mnożone przez k. Jeśli zaś ten przebieg jest wyrażony we współrzędnych *biegunowych*, to wyrażenie $kX[\ ]$ oznacza, że moduł jest mnożony przez k, a faza pozostaje niezmieniona.



Rys. 10.2. Addytywność przekształcenia Fouriera. Dodawanie dwóch lub więcej sygnałów w jednej dziedzinie jest równoważne dodawaniu odpowiadających im sygnałów w drugiej dziedzinie. Na rysunku sygnały w dziedzinie czasu a i b) są dodawane dając sygnał przedstawiony na rys. c). To powoduje sumowanie się części rzeczywistych i części urojonych odpowiednich widm częstotliwościowych

Addytywność przekształcenia Fouriera polega na tym, że *dodawanie* w jednej dziedzinie jest równoważne dodawaniu w drugiej. Przykład przedstawiono na rys. 10.2. Przebiegi na rys. 10.2 a), b) są sygnałami w dziedzinie czasu oznaczonymi $x_1[\ ]$ i $x_2[\ ]$. Dodawanie tych sygnałów daje trzeci sygnał w dziedzinie czasu, oznaczony x_3, przedstawiony na rys. 10.2 c. Każdy z tych trzech sygnałów ma widmo częstotliwościowe składające się z części rzeczywistej i urojonej (rys. 10.2 d ÷ i). Ponieważ dwa sygnały w dziedzinie czasu są do siebie dodawane, dając trzeci sygnał w dziedzinie czasu, więc dwa odpowiednie widma też są dodawane, dając trzecie widmo. Widma częstotliwościowe są dodawane w zapisie we współrzędnych prostokątnych przez sumowanie części rzeczywistych i sumowanie części urojonych. Jeśli $x_1[n] + x_2[n] = x_3[n]$, to $Re\ X_1[f] + Re\ X_2[f] = Re\ X_3[f]$ oraz $Im\ X_1[f] + Im\ X_2[f] = Im\ X_3[f]$. Przemyślmy to w kategoriach fal sinusoidalnych i kosinusoidalnych. Wszystkie fale kosinusoidalne sumują się (części rzeczywiste), a także sinusoidalne się sumują (części urojone) bez wzajemnego oddziaływania między sinusoidami i kosinusoidami.

Widm częstotliwościowych we współrzędnych biegunowych nie można sumować bezpośrednio. Muszą być przetwarzane do współrzędnych prostokątnych, sumowane i z powrotem przetwarzane do współrzędnych biegunowych. Można to też zrozumieć na podstawie zachowania się sinusoid. Wyobraźmy sobie sumowanie dwóch sinusoid o takiej samej częstotliwości, lecz o różnych amplitudach (A_1 i A_2) i fazach (ϕ_1 i ϕ_2). Jeśli zdarzy się, że obie fazy są takie same ($\phi_1 = \phi_2$), to amplitudy będą się *dodawać* do siebie ($A_1 + A_2$), gdy sinusoidy są sumowane. Jeśli zaś fazy będą przeciwne ($\phi_1 = -\phi_2$), to amplitudy będą *odejmować* się od siebie, gdy sinusoidy są sumowane. Wynika stąd, że jeśli sinusoidy (lub widma) są wyrażone w postaci biegunowej, to nie można ich sumować przez proste dodawanie modułów i faz.

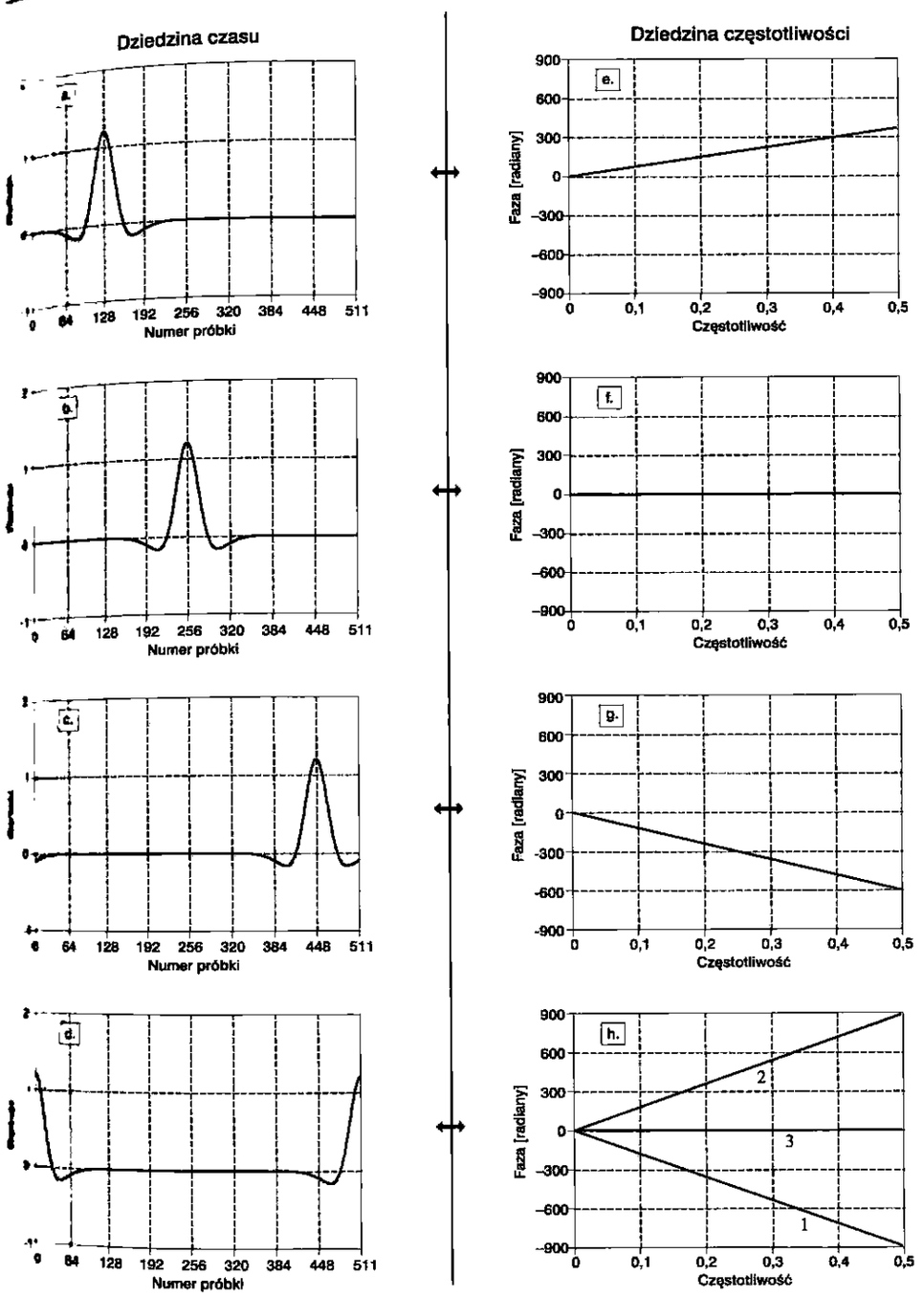
Przekształcenie Fouriera, mimo że liniowe, *nie jest* niezmienne względem przesunięcia. Wynika stąd, że przesunięcie w dziedzinie czasu *nie jest* równoważne przesunięciu w dziedzinie częstotliwości. Ta kwestia będzie tematem następnego podrozdziału.

Właściwości fazowe

Stosując pojęcia matematyczne, można napisać: jeśli $x[n] \leftrightarrow Mag\ X[f]$ i $Phase\ X[n]$, to przesunięcie w dziedzinie czasu daje w wyniku: $x[n+s] \leftrightarrow Mag\ X[f]$ i $Phase\ X[f] + 2\pi sf$, gdzie f jest wyrażone jako ułamek częstotliwości próbkowania mieszczący się w granicach od 0 do 0,5. Przesunięcie w dziedzinie czasu o s próbek pozostawia więc moduł niezmieniony, lecz powoduje dodanie do fazy liniowego składnika $2\pi sf$. Przyjrzyjmy się przykładowi ilustrującemu tę zasadę.

Na rys. 10.3 przedstawiono, jaki wpływ na fazę ma przesunięcie przebiegu w dziedzinie czasu w lewo lub w prawo. W tym przykładzie nie pokazano modułu, gdyż nie jest on tu interesujący. Moduł nie ulega bowiem zmianie przy przesunięciu w dziedzinie czasu. Na rysunkach od 10.3 a) do 10.3 d) przebieg jest stopniowo przesuwany od położenia, w którym jego maksimum znajdowało się w próbce 128 do położenia z maksimum w próbce 0. W tej sekwencji wykresów uwzględniono, że w przekształceniu DFT dziedzina czasu jest traktowana jako *kołowa* – gdy części przebiegu wychodzące poza wykres na prawo pojawiają się znowu z lewej strony.

Przebieg w dziedzinie czasu na rys. 10.3 jest symetryczny względem osi pionowej, tj. lewa i prawa strona są nawzajem względem siebie odbiciami lustrzanymi. Jak wspomniano w rozdziale 7, sygnały o symetrii tego rodzaju są nazywane *liniowofazowymi*, ponieważ ich charakterystyka fazy w funkcji częstotliwości jest *linią prostą*. Sygnały zaś niemające symetrii lewo-prawostronnej są nazywane nieliniowofazowymi i mają charakterystykę fazową odbiegającą od linii prostej. Na rysunkach od 10.3 e) do 10.3 h) przedstawiono charakterystyki fazowe sygnałów z rysunków od 10.3 a) do 10.3 d). Jak podano w rozdziale 7, te sygnały fazowe są *rozwijane*, dzięki czemu mogą występować bez nieciągłości związanych z mianowaniem wartości między π i $-\pi$.



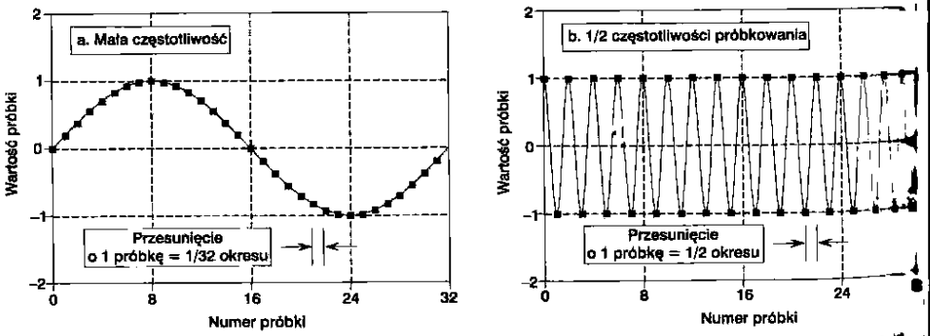
Rys. 10.3. Zmiany charakterystyki fazowej spowodowane przesunięciem w dziedzinie czasu

Gdy przebieg w dziedzinie czasu jest przesuwany w prawo, charakterystyka fazowa pozostaje linią prostą, lecz następuje *zmniejszenie* jej nachylenia. Przesuwanie w lewo powoduje *zwiększenie* nachylenia. Jest to ważna właściwość, którą z tego podrozdziału należy zapamiętać: przesunięcie w dziedzinie czasu odpowiada zmianie nachylenia charakterystyki fazy.

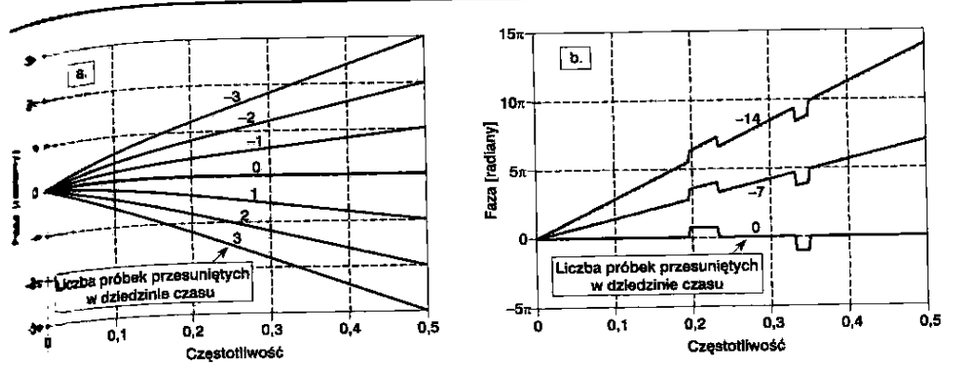
Na rys. 10.3 b), f) przedstawiono szczególny przypadek, gdy faza jest całkowicie zerowa. Taka sytuacja ma miejsce, gdy sygnał w dziedzinie czasu jest *symetryczny* wokół próbki *zero*. Na pierwszy rzut taka symetria na rys. 10.3 b) nie jest oczywista. Może się wydawać, że mamy tu raczej symetrię wokół próbki 256 (tj. $N/2$). Przypomnijmy sobie jednak, że w przekształceniu DFT dziedzina czasu jest widziana jako kołowa, z próbką zero inherentnie dołączoną do próbki $N-1$. Każdy sygnał symetryczny wokół próbki zero jest również symetryczny wokół próbki $N/2$ i vice versa. Jeśli stosuje się te przekształcenia z grupy transformat Fouriera, które nie traktują sygnałów w dziedzinie czasu jako okresowych (jak np. DTFT), symetria musi być wokół próbki zero, aby wytworzyć fazę zerową.

Rysunki 10.3 d), h) mają charakter nieco zagadkowy. Najpierw wyobraźmy sobie, że przebieg na rys. 10.3 d) powstał w wyniku przesunięcia nieco w prawo przebiegu z rys. 10.3 f). To znaczy, że charakterystyka fazowa na rys. 10.3 h) powinna mieć nachylenie trochę bardziej ujemne niż na rys. 10.3 g). Ta charakterystyka jest wykresem 1. Następnie wyobraźmy sobie, że przebieg na rys. 10.3 d) utworzono, rozpoczynając od przebiegu na rys. 10.3 e) i przesuwając go w lewo. W tym przypadku charakterystyka fazowa powinna mieć nachylenie nieco bardziej dodatnie niż na rys. 10.3 e), co zilustrowano wykresem 2. Wreszcie zauważmy, że przebieg na rys. 10.3 d) jest symetryczny wokół próbki $N/2$, więc powinien mieć fazę zero, jak na wykresie 3. Która z tych trzech charakterystyk fazowych jest poprawna? Wszystkie są poprawne, zależnie od potraktowania niejednoznaczności fazowej π i 2π (omówionej w rozdziale 8). Na przykład każda próbka na wykresie 2 różni się od odpowiadającej jej próbki na wykresie 1 o całkowitą wielokrotność 2π, co czyni je równymi sobie. Aby powiązać charakterystykę z wykresu 3 z wykresami 1 i 2, trzeba też wziąć pod uwagę niejednoznaczności π.

W celu zrozumienia omówionego przebiegu charakterystyk fazowych wyobraźmy sobie przesunięcie przebiegu o *jedną* próbkę w prawo. Z tego wynika, że również wszystkie sinusoidy tworzące przebieg muszą być przesunięte o *jedną* próbkę w prawo. Na rys. 10.4 przedstawiono dwie sinusoidy, które mogłyby być częścią przebiegu. Sinusoida na rys. 10.4 a) ma bardzo małą częstotliwość i przesunięcie o jedną próbkę stanowi tylko niewielki ułamek pełnego okresu. Sinusoida na rys. 10.4 b) ma częstotliwość równą połowie częstotliwości próbkowania, a więc największą jaka może występować w próbkowanych danych. Przy tej częstotliwości przesunięcie o jedną próbkę jest równe połowie pełnego okresu lub π radianów. Tak więc, jeśli rozważać przesunięcie jako zmianę fazy, to staje się ono *proporcjonalne* do częstotliwości przesuwanej sinusoidy.



Rys. 10.4. Zależność między próbkami i fazą. Na rysunkach a) i b) przedstawiono sinusoidy o małej i dużej częstotliwości. Na rys. a) przesunięcie o jedną próbkę jest równe 1/32 okresu, a na rys. b) – połowie okresu. Właśnie dlatego przesunięcie przebiegu zmienia fazę bardziej przy dużych niż przy małych częstotliwościach



Rys. 10.5. Zmiany fazy wynikające z przesunięcia w dziedzinie czasu o jedną próbkę w kierunku dodatnim (tj. w prawo) faza przy częstotliwości 0,5 zmniejsza się o π radianów. Przy każdym przesunięciu o jedną próbkę w kierunku ujemnym (tj. w lewo) faza przy częstotliwości 0,5 wzrasta o π radianów. Na rys. a) pokazano te zmiany dla liniowej charakterystyki fazowej (linia prosta), a na rys. b) dla nieliniowej

Rozważmy na przykład przebieg, który jest symetryczny wokół próbki zero, a zatem ma fazę zerową. Na rys. 10.5 a) pokazano, jak faza tego sygnału się zmienia, gdy sygnał jest przesuwany w lewo lub w prawo. Przy największej częstotliwości, równej połowie częstotliwości próbkowania, faza wzrasta o π przy każdym przesunięciu o jedną próbkę w lewo i zmniejsza się o π przy przesunięciu o jedną próbkę w prawo. Przy częstotliwości zerowej nie ma przesunięcia fazy. W całym zakresie, od 0 do częstotliwości maksymalnej (równej 0,5 częstotliwości próbkowania), zmiany fazy w funkcji częstotliwości są liniowe.

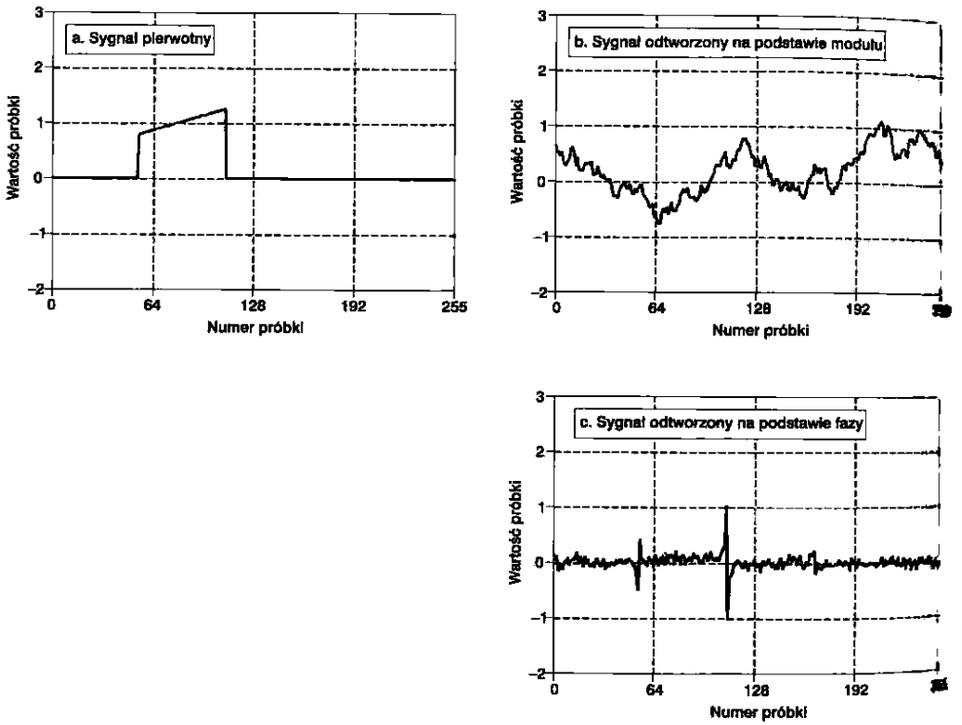
Wszystkie przykłady, które dotychczas podaliśmy, dotyczyły fazy *liniowej*. Na rys. 10.5 b) pokazano, że sygnały o *nieliniowej* charakterystyce fazowej reagują na przesunięcie w taki sam sposób, jak sygnały o charakterystyce liniowej. W tym przykładzie nieliniowa charakterystyka fazowa jest linią prostą z nałożonymi na nią dwoma impulsami prostokątnymi. Gdy następuje przesunięcie w dziedzinie czasu, to elementy nieliniowe są po prostu nakładane na prostą o zmienionym nachyleniu.

Co się dzieje z częścią *rzeczywistą* i *urojoną*, gdy przebieg w dziedzinie czasu ulega przesunięciu? Przypomnijmy sobie, że sygnały w dziedzinie częstotliwości zapisane we współrzędnych prostokątnych są w zasadzie niemożliwe do bezpośredniego zrozumienia. Części rzeczywista i urojona wyglądają na ogół jak przypadkowe oscylacje bez wyraźnego kształtu. Gdy sygnał w dziedzinie czasu ulega przesunięciu, te „pokręcone" przebiegi stają się jeszcze bardziej oscylacyjne i trudne do interpretacji. Nie traćcie czasu na próby zrozumienia tych sygnałów oraz charakteru ich zmian przy przesunięciach w dziedzinie czasu.

Na rys. 10.6 w ciekawy sposób przedstawiono, jaka informacja jest zawarta w *fazie* i jaka w *module*. Przebieg na rys. 10.6 a) ma dwie istotne cechy – zbocze narastające przy próbce 55 i zbocze opadające przy próbce 110. Zbocza są bardzo ważne, jeśli informacja jest zawarta w *kształcie* przebiegu. Zbocze wskazuje, *kiedy* coś się zdarza, rozdzielając to, co się dzieje na lewo od zbocza, od tego, co się dzieje na prawo. Jest to w dziedzinie czasu informacja w jej najczystszej formie. Na rys. 10.6 wykonano najpierw przekształcenie DFT sygnału z rys. 10.6 a) i uzyskane widmo częstotliwościowe wyrażono we współrzędnych biegunowych. Aby znaleźć sygnał przedstawiony na rys. 10.6 b), zastąpiono fazę losowo wybranymi liczbami z zakresu od $-\pi$ do π i wykonano odwrotne przekształcenie DFT w celu odtworzenia przebiegu w dziedzinie czasu. Tak więc przebieg na rys. 10.6 b) ma źródło tylko w informacji zawartej w *module*. W podobny sposób, zastępując moduł małymi liczbami losowymi przed wykonaniem odwrotnego przekształcenia DFT, wyznaczono przebieg przedstawiony na rys. 10.6 c). Odtworzenie przebiegu przedstawione na rys. 10.6 c) uzyskano więc tylko na podstawie informacji zawartych w *fazie*.

I jaki mamy wynik? Położenia zboczy są wyraźnie widoczne w przebiegu na rys. 10.6 c, a wcale ich nie ma na rys. 10.6 b). Jest tak dlatego, bo zbocze tworzy się tam, gdzie razem jednocześnie wiele sinusoid, co jest możliwe tylko w przypadku skoordynowania ich faz. Krótko mówiąc, dużo informacji o kształcie przebiegu w dziedzinie czasu jest zawarte raczej w *fazie* niż w *module*. Pozostaje to w przeciwieństwie do sygnałów, w których informacja jest zakodowana w dziedzinie częstotliwości – takich, jak sygnały foniczne. W tych sygnałach najważniejszy jest moduł, faza ma mniejsze znaczenie. W dalszych rozdziałach zobaczymy, że ten sposób rozumowania tworzy strategie projektowania filtrów i innych metod przetwarzania sygnałów. Zrozumienie, w jaki sposób informacja jest reprezentowana w sygnale, jest zawsze pierwszym krokiem do udanego DSP.

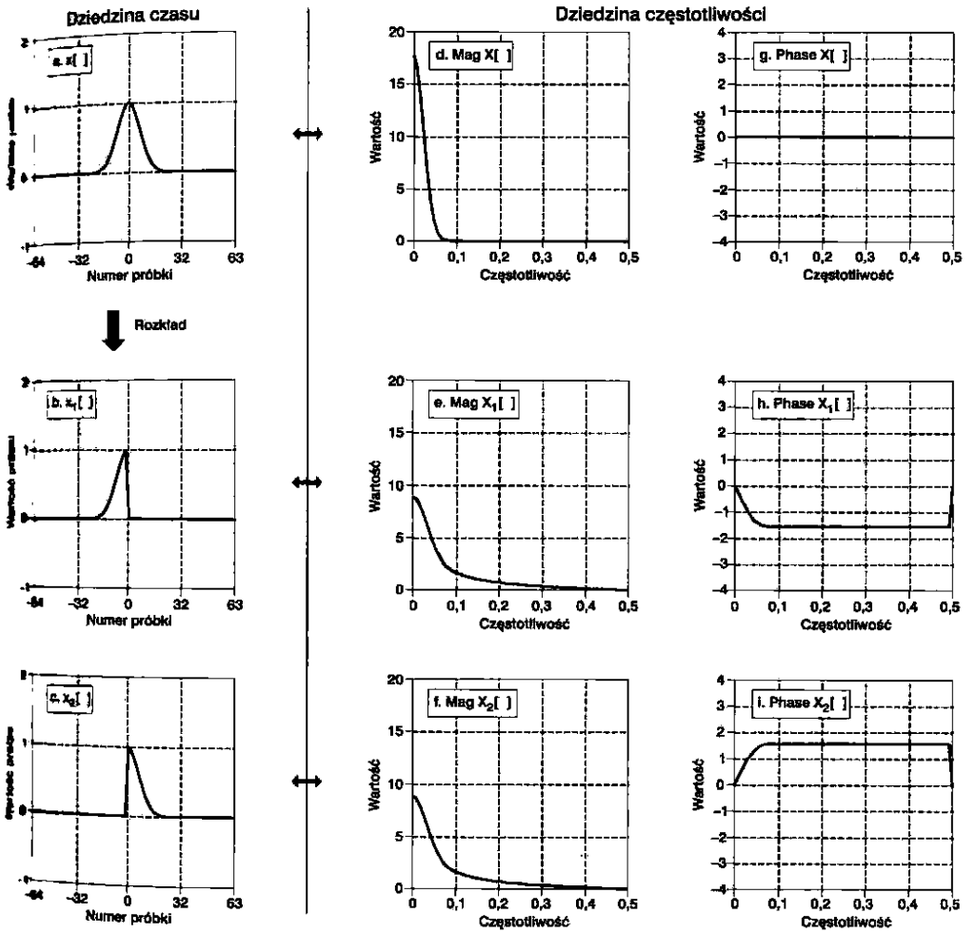
Dlaczego symetria lewo-prawostronna jest równoważna zerowej (lub liniowej) fazie? Odpowiedź znajdziecie na rys. 10.7. Taki sygnał może być rozłożony na prawą i lewą połówkę (rys. 10.7 a, b, c). Próbka w środku symetrii (w tym przypadku jest nim zero) zostaje podzielona równo między lewą i prawą połówkę, dzięki czemu obie strony są wzajemnie idealnymi odbiciami lustrzanymi. Moduły tych dwóch połówek będą *identyczne*, jak to przedstawiono na rys. 10.7 e), f), fazy zaś będą miały odwrotne znaki (rys. 10.7 h, i). Wynikają z tego dwa ważne stwierdzenia. Po pierwsze, każdy sygnał symetryczny lewo-prawostronnie będzie miał liniową charakterystykę fazową, *gdyż* nieliniowość fazy w lewej połówce dokładnie kompensuje nieliniowość prawej połówki.



Rys. 10.6. Informacja zawarta w fazie. Na rys. a) przedstawiono przebieg o charakterze impulsowym. Sygnał na rys. b) znaleziono, wykonując przekształcenie DFT przebiegu z rys. a), zastępując fazę losowymi liczbami i przeprowadzając odwrotne przekształcenie DFT. Sygnał na rys. c) znaleziono, wykonując DFT przebiegu z rys. a), zastępując moduł liczbami losowymi i przeprowadzając odwrotne przekształcenie DFT. Informacja o położeniu zboczy jest zachowana na rys. c), lecz nie na rys. b). Wynika stąd, że faza zawiera informację o położeniu zdarzeń w sygnale w dziedzinie czasu

Po drugie, wyobraźmy sobie lewo-prawostronne odwracanie przebiegu na rys. 10.7 b) tak, aby stawał się przebiegiem z rys. 10.7 c). Takie odwracanie w dziedzinie czasu nie ma wpływu na moduł, lecz zmienia znak każdego punktu charakterystyki fazowej. Podobnie, zmiana znaku fazy powoduje lewo-prawostronne odwrócenie sygnału w dziedzinie czasu. Jeśli sygnały są ciągłe, to odwrócenie następuje wokół zera, jeśli zaś dyskretne – to wokół próbki zero *oraz* jednocześnie wokół próbki *N*/2.

Zmiana znaku fazy jest operacją na tyle częstą, że nadano jej nazwę. Jest to tworzenie wartości **zespolonej sprzężonej**, oznaczanej przez postawienie gwiazdki u góry z prawej strony zmiennej. Na przykład jeśli *X*[*f*] składa się z *Mag X*[*f*] i *Phase X*[*f*], to *X**[*f*] nazywa się wartością *zespoloną sprzężoną* i składa się z: *Mag X*[*f*] i *–Phase X*[*f*]. We współrzędnych prostokątnych wartość zespoloną sprzężoną wyznacza się, pomijając część rzeczywistą i zmieniając znak części urojonej. Wyrażając to w kategoriach matematycznych: jeśli *X*[*f*] składa się z *Re X*[*f*] i *Im X*[*f*], to *X**[*f*] tworzy się z *Re X*[*f*] i *–Im X*[*f*].



Rys. 10.7. Charakterystyki fazowe sygnału lewo-prawostronnie symetrycznego. Sygnał o takiej symetrii, przedstawiony na rys. 10.7 a), może być rozłożony na połówki – prawą (rys. b) i lewą (rys. c). Moduły obu połówek są identyczne (rys. e oraz f), podczas gdy ich fazy różnią się znakiem (rys. h oraz i)

Oto kilka przykładów zastosowania wartości zespolonych sprzężonych w DSP. Jeśli $x[n]$ ma transformatę Fouriera $X[f]$, to wtedy $x[-n]$ ma transformatę Fouriera $X^*[f]$. To znaczy, że lewo-prawostronne odwrócenie sygnału w dziedzinie czasu odpowiada zmianie znaku fazy. Przy omówieniu następnego przykładu przypomnijmy sobie z rozdziału 7, że korelację można realizować jako splot. Wykonuje się to, odwracając lewo-prawostronnie jeden z sygnałów. Matematycznie $a[n] * b[n]$ jest splotem, a $a[n] * b[-n]$ jest korelacją. W dziedzinie częstotliwości te operacje są równoważne – odpowiednio – operacjom $A[f] \times B[f]$ oraz $A[f] \times B^*[f]$. Jako ostatni przykład rozpatrzmy dowolny sygnał $x[n]$ i jego widmo częstotliwościowe $X[f]$. Widmo częstotliwościowe można zamienić na *zerofazowe*, mnożąc je przez jego wartość zespoloną sprzężoną, czyli $X[f] \times X^*[f]$. Zatem faza $X[f]$, jakakolwiek by nie była, będzie zawsze redukowana przez dodanie jej odwrotności (pamiętajmy, że gdy widma częstotliwościowe są mnożone, to ich fazy się sumują). W dziedzinie czasu oznacza to, że $x[n] * x[-n]$, czyli sygnał spleciony z lewo-prawostronną wersją jego samego, będzie miał lewo-prawostronną symetrię wokół próbki zero, niezależnie od tego, czym jest $x[n]$.

Wielu inżynierów i matematyków uważa, że ten rodzaj operacji *to jest* DSP. Jeśli chcecie móc porozumiewać się z tą grupą, to musicie przyzwyczaić się do stosowania ich języka.

Okresowy charakter DFT

W odróżnieniu od pozostałych trzech przekształceń fourierowskich, w DFT traktuje się przebiegi zarówno w dziedzinie czasu, jak i częstotliwości jako *okresowe*. Może to być niekiedy godne i kłopotliwe, ponieważ sygnały stosowane w DSP w większości *nie są* okresowe. Niemniej jednak, jeśli chcemy stosować DFT, to musimy dostosować się do sposobu widzenia przebiegów przez DFT.

Na rys. 10.8 przedstawiono dwie różne interpretacje sygnału w dziedzinie czasu. Spójrzmy najpierw na rysunek górny, na którym sygnał w dziedzinie czasu jest widziany jako przebieg o długości N punktów. Reprezentuje to typowy sposób pobierania sygnałów cyfrowych w eksperymentach naukowych i zastosowaniach inżynierskich. Te 128 próbek mogło na przykład być pobranych przez próbkowanie jakiegoś parametru w regularnych odstępach *czasu*. Próbka 0 różni się od próbki 127 i jest w stosunku do niej odrębna, gdyż te próbki były pobrane w *różnym* czasie. Nie ma żadnego powodu, aby na podstawie ukształtowania sygnału sądzić, że próbki z lewej strony sygnału są w sposób parzysty związane z próbkami po stronie prawej.

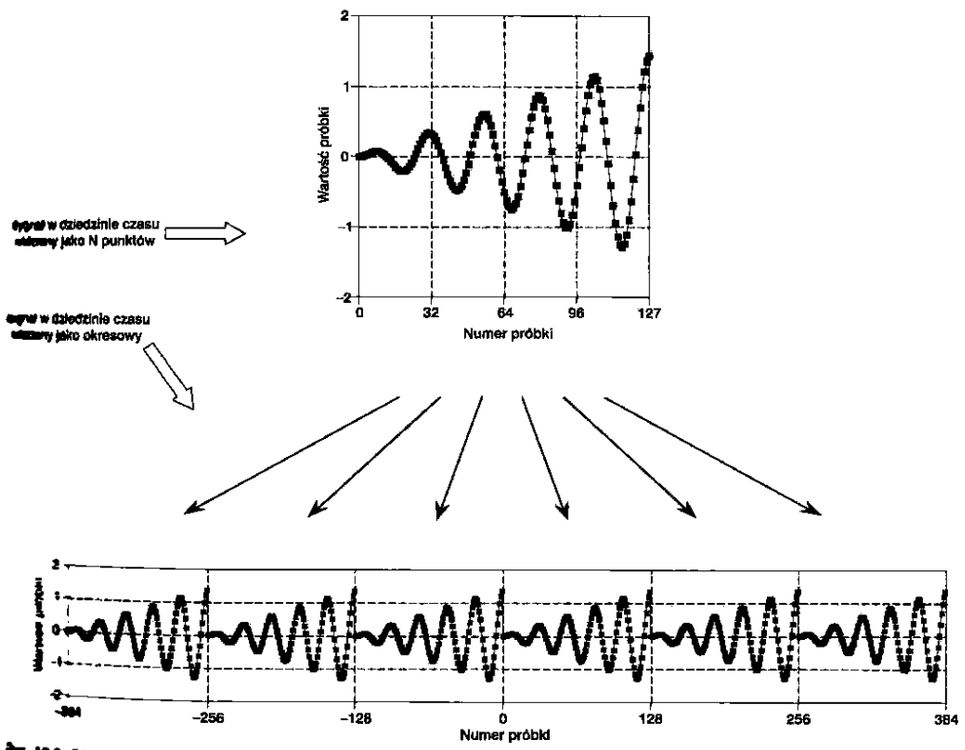
Niestety, w DFT problem nie jest widziany w ten sposób. Jak pokazano na dolnym rysunku, DFT traktuje te 128 punktów jako pojedynczy okres nieskończenie długiego sygnału okresowego. Wynika stąd, że lewa strona zebranego sygnału jest powiązana z prawą stroną sygnału powtórzonego. W podobny sposób prawa strona zebranego sygnału jest powiązana z lewą stroną identycznego okresu. Można to sobie też wyobrazić jako zawijanie prawej strony zebranego sygnału i dołączenie do jego lewej strony. Przy takim podejściu próbka 127 znajduje się obok próbki 0, w podobny sposób jak próbka 43 jest obok próbki 44. Jest to potraktowanie dziedziny czasu jako **kołowej**, co jest identyczne z sygnałem *okresowym*.

Najpoważniejszą konsekwencją okresowości w dziedzinie czasu jest tzw. **aliasing w dziedzinie czasu**. Aby zilustrować to zjawisko, załóżmy, że mamy sygnał w dziedzinie czasu i poddajemy go przekształceniu DFT w celu znalezienia jego widma częstotliwościowego. Możemy następnie natychmiast poddać to widmo odwrotnemu przekształceniu DFT, aby odtworzyć sygnał oryginalny w dziedzinie czasu. Cała taka procedura nie byłaby jednak dla nas interesująca. Zamiast tego, w pewien sposób zmodyfikujemy widmo częstotliwościowe przed zastosowaniem odwrotnego przekształcenia DFT. Można na przykład usunąć wybrane częstotliwości, zmienić amplitudy lub fazy, przesunąć itd. Są to operacje wykonywane w DSP nowo. Niestety, takie zmiany w dziedzinie częstotliwości mogą w dziedzinie czasu utworzyć sygnał zbyt długi, aby zmieścił się w pojedynczym okresie. To powoduje, że część sygnału zachodzi z jednego okresu na okresy sąsiednie. Jeśli przebieg w dziedzinie czasu jest widziany

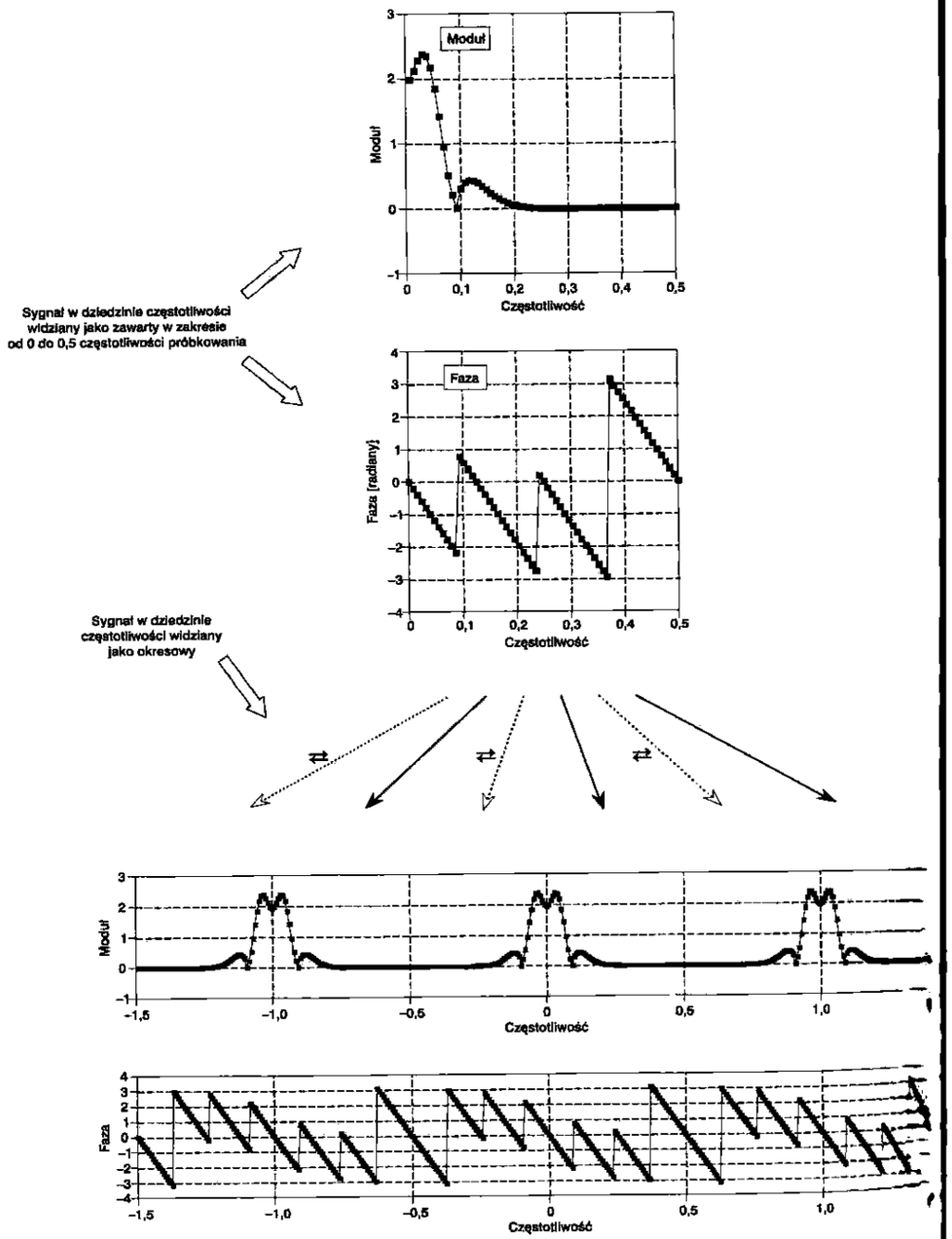
jako *kołowy*, to wydaje się, że części sygnału, które zachodzą na prawo, znowu pojawiają się po lewej stronie i vice versa. Tak więc części widma zachodzące na sąsiednie okresy przypisują się do nowej lokalizacji w dziedzinie czasu (tzw. zjawisko aliasingu). Jeśli w tej lokalizacji był już zawarty jakiś istniejący sygnał, to cała ta mieszanina sumuje się, powodując w rezultacie utratę informacji. Doskonałym przykładem aliasingu tego typu jest omówiony w rozdziale 9 splot kołowy będący wynikiem mnożenia w dziedzinie częstotliwości.

Okresowość w dziedzinie częstotliwości przejawia się w dużym stopniu w podobny sposób, lecz jest bardziej skomplikowana. Przykład podano na rys. 10.9. Na górnych rysunkach przedstawiono moduł i fazę widma częstotliwościowego, zobrazowane jako składające się z $N/2 + 1$ próbek rozmieszczonych między 0 i 0,5 częstotliwości próbkowania. Jest to najprostszy sposób zobrazowania widma częstotliwościowego, który jednak nie wyjaśnia wielu właściwości DFT.

Na dwóch dolnych rysunkach przedstawiono, w jaki sposób w przekształceniu DFT widmo częstotliwościowe jest widziane jako okresowe. Podstawową sprawą jest to, że widmo częstotliwościowe w zakresie częstotliwości od 0 do 0,5 ma *odbicie lustrzane* dla częstotliwości od 0 do –0,5. To lustrzane odbicie przy **częstotliwościach ujemnych** różni się nieco pod względem amplitudy i fazy sygnałów. Zarówno moduł sygnału, jak i jego faza są odwrócone lewo-prawostronnie, przy czym jednak *znak* fazy ulega zmianie. Jak pamiętacie, te dwa rodzaje symetrii mają swoje nazwy: o module mówi się, że jest sygnałem **parzystym** (o symetrii *parzystej*), podczas gdy faza jest sygnałem **nieparzystym** (o symetrii *nieparzystej*). Jeśli widmo częstotliwościowe zostaje przetworzone na część rzeczywistą i urojoną, to *część rzeczywista* zawsze będzie *parzysta*, a *część urojona* – zawsze *nieparzysta*.



Rys. 10.8. Okresowość sygnału w dziedzinie czasu w DFT. Sygnał w dziedzinie czasu można widzieć jako przebieg o długości N próbek (rysunek górny) lub jako nieskończenie długi sygnał okresowy (rysunek dolny)



Rys. 10.9. Okresowość DFT w dziedzinie częstotliwości. Przebieg w dziedzinie częstotliwości może być widziany jako zawarty w zakresie od 0 do 0,5 częstotliwości próbkowania (dwa górne rysunki) lub jako nieskończenie długi sygnał okresowy z lewo-prawostronnym odwróceniem każdego segmentu od 0 do 0,5 (dwa dolne rysunki)

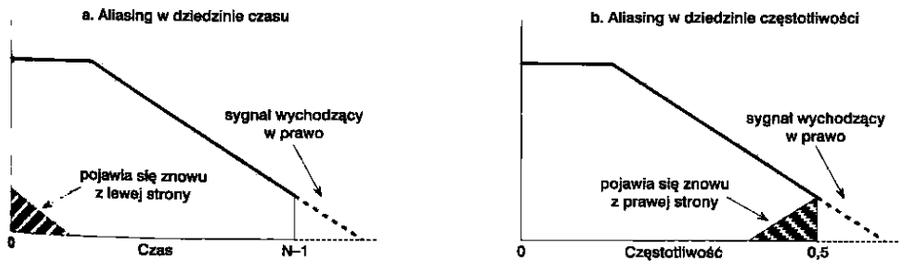
Z uwzględnieniem częstotliwości ujemnych, sygnały w dziedzinie częstotliwości w DFT są widziane jako okresowe, o okresie 1,0 pomnożonym przez częstotliwość próbkowania, tj. od –0,5 do 0,5 lub od 0 do 1,0. Długość okresu w dziedzinie częstotliwości, wyrażona w numerach próbek, jest równa N, taka samo jak w dziedzinie czasu.

Okresowość sygnałów w dziedzinie częstotliwości czyni je podatnymi na **aliasing w dziedzinie częstotliwości**, w pełni analogiczny do poprzednio opisanego aliasingu w dziedzinie czasu. Wyobraźmy sobie jakiś sygnał w dziedzinie czasu mający równoważne mu widmo częstotliwościowe. Gdy sygnał w dziedzinie czasu ulega modyfikacji, jest oczywiste, że jego widmo częstotliwościowe też się zmieni. Jeśli zmodyfikowane widmo częstotliwościowe nie może się zmieścić w obszarze przewidzianym, to jego część przejdzie do sąsiednich okresów. Tak jak poprzednio, taki aliasing powoduje dwa problemy: częstotliwości nie są umiejscowione tam, gdzie powinny być, a nakładające się sygnały z różnych okresów sumują się, niszcząc informację.

Aliasing w dziedzinie częstotliwości jest trudniejszy do zrozumienia niż w dziedzinie czasu, gdyż charakter okresowości jest w dziedzinie czasu bardziej skomplikowany. Rozważmy jakąś jedną częstotliwość, która zostaje zmuszona do przesunięcia się z pozycji 0,01 do 0,49 w dziedzinie częstotliwości. Odpowiadająca jej częstotliwość ujemna przemieszcza się więc od –0,01 do –0,49. Gdy częstotliwość dodatnia przechodzi przez barierę 0,5, to częstotliwość ujemna jest przemieszczana przez barierę –0,5. Ponieważ dziedzina częstotliwości ma charakter okresowy, więc takie same zdarzenia zachodzą w innych okresach, np. od 0,5 do 1,5. Klon częstotliwości dodatniej przekracza częstotliwość 1,5 od lewej do prawej strony, podczas gdy klon częstotliwości ujemnej przekracza 0,5 od prawej do lewej. Pomyślmy, jakby to było, gdyby widoczny był tylko zakres częstotliwości od 0 do 0,5. Wydawałoby się, że częstotliwości wychodzące z *prawej* strony znowu się pojawiają z *prawej* strony, lecz poruszają się w odwrotnym kierunku.

Na rys. 10.10 przedstawiono, jak aliasing przejawia się w dziedzinach czasu i częstotliwości, gdy widziany jest tylko pojedynczy okres. Jeśli jeden koniec sygnału w dziedzinie czasu jest zbyt długi (rys. 10.10 a), aby zmieścić się w pojedynczym okresie, to wystający kraniec jest *obcinany* i *dołączany* z drugiej strony. Jeśli zaś sygnał w dziedzinie częstotliwości wykracza poza pojedynczy okres, wystający koniec ulega *składaniu*. Niezależnie od tego, gdzie kończy się segment podlegający aliasingowi, dodaje się on do sygnału obecnego w miejscu składania. Informacja ulega więc zniszczeniu.

Przyjrzyjmy się bliżej temu dziwnemu pojęciu, jakim są *częstotliwości ujemne*. Powstaje pytanie, czy są tylko dziwacznym artefaktem matematyki, czy mają jakieś znaczenie w świecie rzeczywistym. Tę kwestię wyjaśniono na rys. 10.11, gdzie przedstawiono sygnał dyskretny złożony z 32 próbek (rys. 10.11 a). Powiedzmy, że postawiono nam zadanie znalezienia widma częstotliwościowego odpowiadającego tym 32 punktom. Aby ułatwić



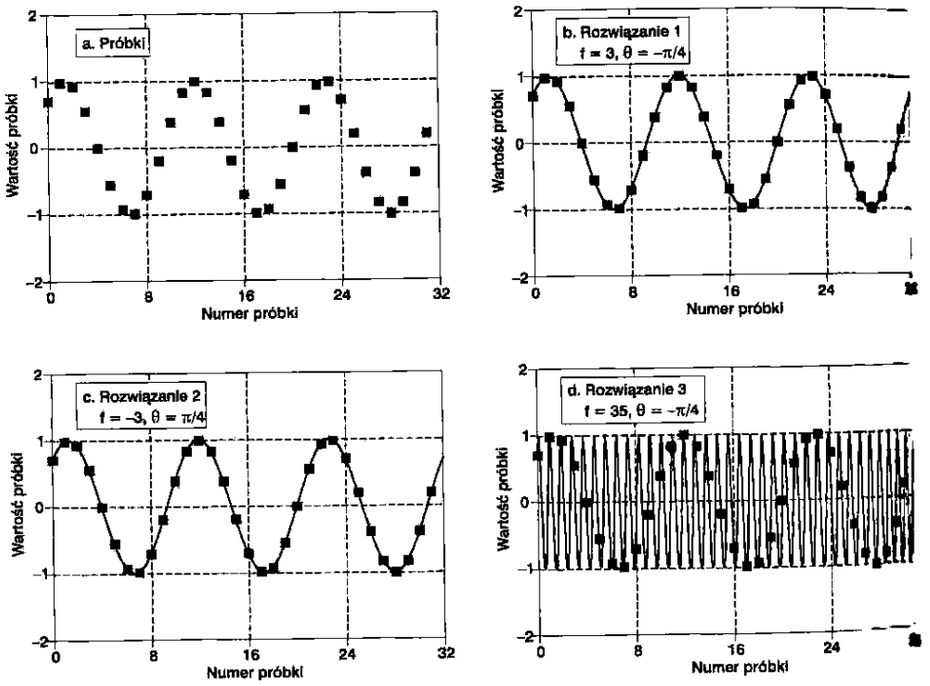
Rys. 10.10. Przykłady aliasingu w dziedzinach czasu i częstotliwości w przypadku, gdy tylko pojedynczy okres jest brany pod uwagę. W dziedzinie czasu (rys. a) segmenty sygnału wychodzące poza ten okres na prawo pojawiają się znowu z lewej strony. W dziedzinie częstotliwości (rys. b) segmenty sygnału wychodzące poza ten okres na prawo pojawiają się znowu też na prawo tak, jakby były składane

to zadanie, podano, że te punkty reprezentują dyskretną falę kosinusoidalną. Musimy wy znaleźć częstotliwość i przesunięcie fazy (f, θ) takie, żeby zależność $x[n] = \cos(2\pi fn/N + \theta$ odpowiadała danym próbkom. Nie trzeba wiele czasu, aby dojść do rozwiązania przedstawionego na rys. 10.11 b), tzn. $f = 3$ oraz $\theta = -\pi/4$.

Jeśli zakończymy w tym miejscu naszą analizę, to uzyskamy tylko 1/3 rozwiązania problemu. Są bowiem dwa inne rozwiązania, które pominęliśmy. Jak pokazano na rys. 10.11 c), drugim rozwiązaniem jest $f = -3$ oraz $\theta = \pi/4$. Nawet jeśli pojęcie *ujemnej częstotliwości* obraża nasz zdrowy rozsądek, nie zmienia to faktu, że jest to matematycznie poprawne rozwiązanie postawionego problemu. Każda sinusoida o częstotliwości *dodatniej* może być alternatywnie wyrażona jako sinusoida o częstotliwości *ujemnej*. Dotyczy to zarówno sygnałów ciągłych jak i dyskretnych.

Trzecie rozwiązanie nie daje jednej odpowiedzi, gdyż jest to raczej rodzina nieskończonej liczby rozwiązań. Sinusoida o $f = 35$ i $\theta = -\pi/4$ (rys. 10.11 d) przechodzi przez wszystkie punkty dyskretne, jest więc poprawnym rozwiązaniem. Nieco mylący może być fakt, że między próbkami występują oscylacje, lecz nie dyskwalifikuje to prawidłowości tego rozwiązania. Również rozwiązania $f = \pm 29$, $f = \pm 35$, $f = \pm 61$ oraz $f = \pm 67$ są rozwiązaniami o wiele oscylacjach między punktami.

Każde z tych trzech rozwiązań odpowiada innej sekcji widma częstotliwości. Dla sygnałów dyskretnych pierwsze rozwiązanie odpowiada częstotliwościom między 0 i 0,5 częstotliwości próbkowania. Drugie rozwiązanie daje w wyniku częstotliwości pomiędzy 0 i –0,5, a trzecie daje nieskończoną liczbę powtarzanych częstotliwości poniżej –0,5 i powyżej 0,5. Jeśli



Rys. 10.11. Znaczenie częstotliwości ujemnych. Postawionym zadaniem jest znalezienie widma częstotliwościowego sygnału dyskretnego przedstawionego na rys. 10.11 a). Chcemy więc wyznaczyć częstotliwość i fazę sinusoidy, która przechodzi przez wszystkie próbki. W rozwiązaniu na rys. 10.11 b) występują częstotliwości dodatnie, a na rys. 10.11 c) – ujemne. Na rys. 10.11 d) pokazano rodzinę rozwiązań tego problemu

lizowane sygnały są ciągłe, to pierwsze rozwiązanie daje częstotliwości od zera do dodatniej nieskończoności, podczas gdy drugie – od zera do ujemnej nieskończoności. Trzecia grupa rozwiązań nie istnieje dla sygnałów ciągłych.

W wielu metodach DSP nie ma konieczności stosowania częstotliwości ujemnych lub zrozumienia okresowości DFT. Przykłady dwóch takich często spotykanych metod opisano w poprzednim rozdziale. Była to *analiza widmowa* oraz *odpowiedź częstotliwościowa* systemu. W tych zastosowaniach zupełnie wystarcza rozpatrywanie sygnałów w dziedzinie czasu jako obejmujących próbki od 0 do $N-1$, a w dziedzinie częstotliwości – od zera do połowy częstotliwości próbkowania. W tych metodach można w prostszy sposób patrzeć na świat sygnałów, gdyż nigdy w rezultacie nie uzyskuje się tu zachodzenia części sygnału z jednego okresu na inne okresy. W takich przypadkach rozpatrywanie pojedynczego okresu jest równie dobre jak całego sygnału okresowego.

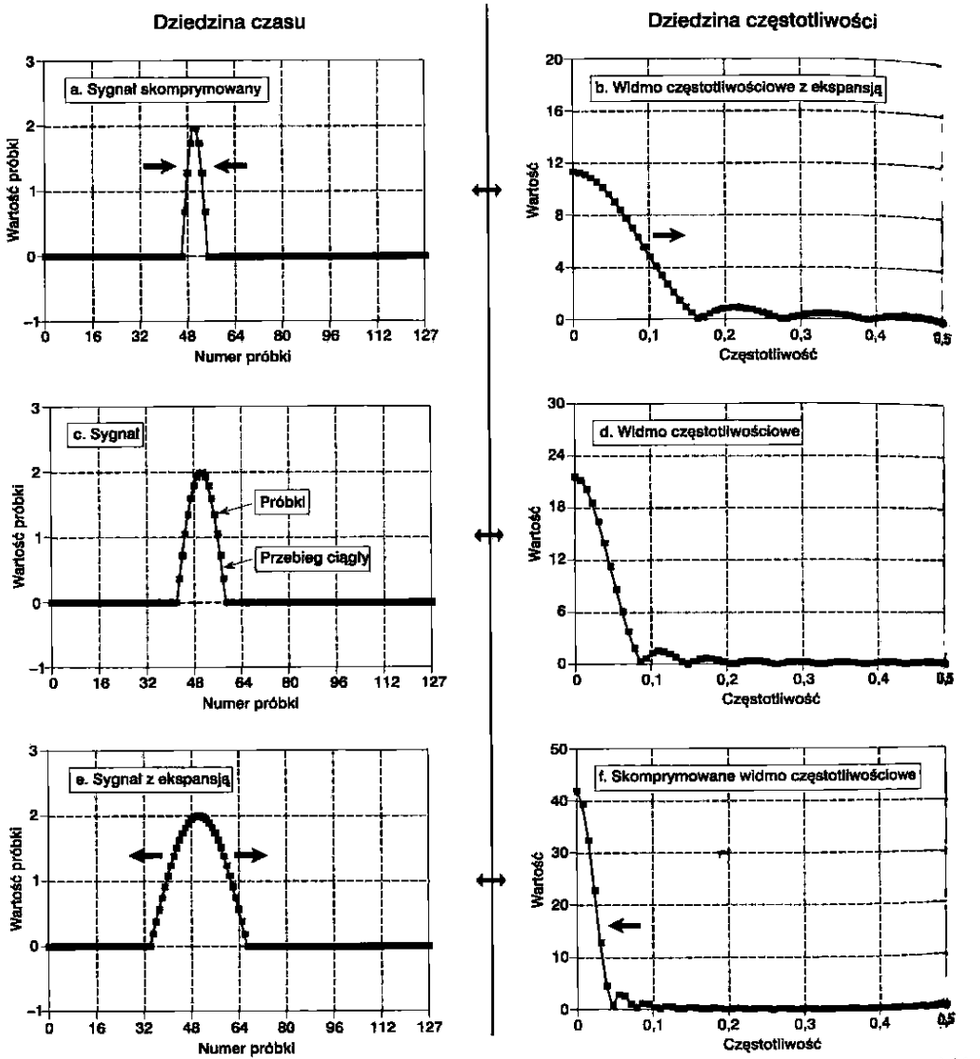
Jednakże niektóre procedury mogą być realizowane *tylko* z uwzględnieniem zachodzenia sygnału na inne okresy. Dwa takie przykłady już zaprezentowano – *splot kołowy* oraz *przetwarzanie analogowo-cyfrowe*. W przypadku splotu kołowego mnożenie widm częstotliwościowych daje w wyniku w dziedzinie czasu sygnały splatane. Jeśli sygnał uzyskany w dziedzinie czasu jest zbyt długi, aby mógł zmieścić się w pojedynczym okresie, to zachodzi na sąsiednie okresy, powodując *aliasing w dziedzinie czasu*. Przetwarzanie analogowo-cyfrowe zaś jest przykładem *aliasingu w dziedzinie częstotliwości*. W dziedzinie czasu jest przeprowadzana operacja nieliniowa – zamiana sygnału ciągłego w dyskretny przez próbkowanie. Trudność polega na tym, że widmo oryginalnego sygnału analogowego może być zbyt długie, aby zmieścić się wewnątrz widma. Forsowanie wykonania tej operacji może spowodować, że krańce widma będą wystawać poza pojedynczy okres i zachodzić na sąsiednie. Rozpatrzmy dwa dalsze przykłady, w których ważny jest okresowy charakter DFT. Są to *kompresja* i *ekspansja* sygnału oraz *modulacja amplitudy*.

Kompresja i ekspansja, przetwarzanie o wielu częstotliwościach próbkowania

Kompresja sygnału w jednej dziedzinie powoduje *ekspansję* w drugiej, jak to przedstawiono na rys. 10.12. Dla sygnałów ciągłych, jeśli $X(f)$ jest transformatą Fouriera sygnału $x(t)$, to $1/k \times X(f/k)$ jest transformatą Fouriera sygnału $x(kt)$, gdzie k jest parametrem opisującym ekspansję lub kompresję. Jeśli zdarzenie przebiega *szybciej* (czyli podlega kompresji w czasie), to musi być złożone z *większych* częstotliwości. Jeśli zdarzenie przebiega *wolniej* (podlega ekspansji w czasie), to musi się składać z *mniejszych* częstotliwości. Ta zasada dotyczy także sytuacji skrajnych. Tak więc, jeśli sygnał w dziedzinie czasu poddano takiej kompresji, że stał się on *impulsem*, to odpowiadające mu widmo częstotliwościowe ulega ekspansji tak daleko idącej, że staje się *wartością stałą*. Podobnie, jeśli sygnał w dziedzinie czasu poddano tak silnej ekspansji, że stał się wartością stałą, to przebieg w dziedzinie częstotliwości staje się impulsem.

Sygnały dyskretne zachowują się w podobny sposób, lecz trzeba tu dodać kilka szczegółów. Pierwszym zagadnieniem w przypadku sygnałów dyskretnych jest *aliasing*. Przyjmijmy, że impuls na rys. 10.12 a) jest poddany kompresji kilkakrotnie większej, niż to pokazano na rysunku. Widmo częstotliwościowe ulega ekspansji o taki sam czynnik i kilka garbów na rys. 10.12 b) jest zepchniętych do częstotliwości powyżej 0,5. Powstający aliasing łamie prostą zależność między ekspansją i kompresją. Aliasing tego rodzaju może również zachodzić w dziedzinie czasu. Wyobraźmy sobie, że widmo częstotliwościowe na rys. 10.12 f) ulega znacznie silniejszej kompresji. Może to spowodować rozciąganie się sygnału w dziedzinie czasu (rys. 10.12 e) na sąsiednie okresy.

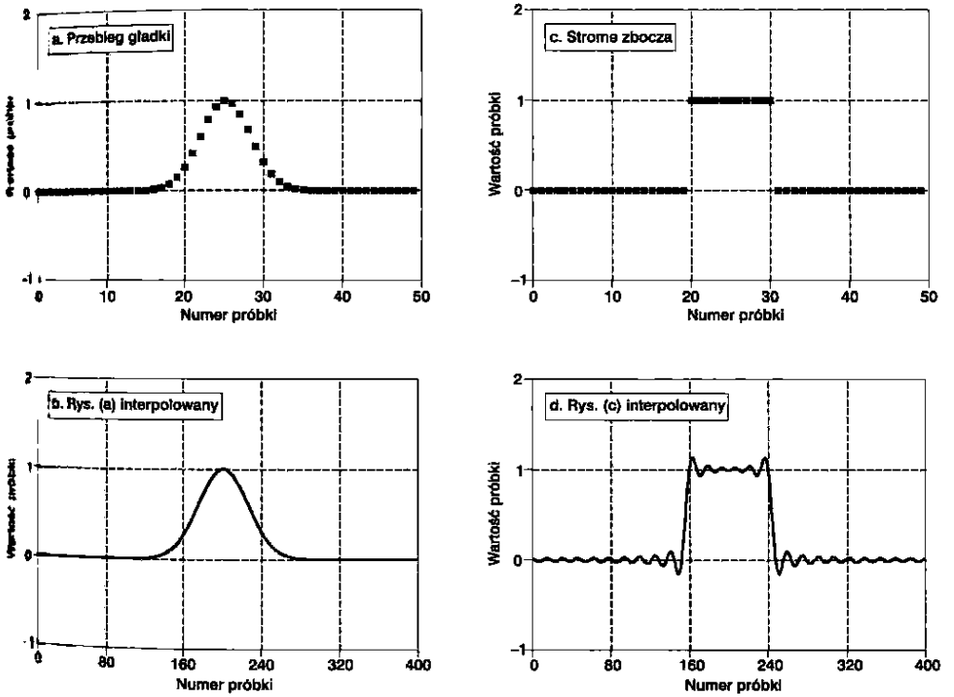
Drugim zagadnieniem jest ścisłe zdefiniowanie, co znaczy kompresja i ekspansja sygnału dyskretnego. Jak pokazano na rys. 10.12 a), sygnał dyskretny jest poddawany kompresji przez kompresję krzywej *ciągłej*, z której pochodzą próbki i następnie ponowne próbkowanie nowej krzywej ciągłej w celu uzyskania nowego sygnału dyskretnego. Taką samą procedurę dla ekspansji przedstawiono na rys. 10.12 e). Gdy sygnał dyskretny jest poddawany kompresji, zdarzenia w sygnale (takie jak szerokość impulsu) mieszczą się w *mniejszej* liczbie próbek. Natomiast zdarzenia w sygnale poddanym ekspansji obejmują *większą* liczbę próbek.



Rys. 10.12. Kompresja i ekspansja. Kompresja sygnału w jednej dziedzinie daje w rezultacie ekspansję w drugiej i odwrotnie. Na rysunkach c), d) przedstawiono sygnał dyskretny i jego widmo. Na rysunkach a), b) pokazano, jak kompresja w dziedzinie czasu powoduje ekspansję w dziedzinie częstotliwości. Na rysunkach e), f) przedstawiono proces odwrotny. Jak pokazano na tych rysunkach, sygnały dyskretne są poddawane ekspansji lub kompresji przez ekspansję lub kompresję sygnału ciągłego będącego ich źródłem. Ten sygnał ciągły jest następnie ponownie próbkowany w celu uzyskania nowego sygnału dyskretnego

Inne podejście do tej procedury polega na utrzymaniu takiego samego oryginalnego przebiegu ciągłego i na ponownym próbkowaniu go z inną częstotliwością. Spójrzmy na przykład z rys. 10.13 a), a więc na dyskretny przebieg gaussowski złożony z 50 próbek. Na rys. 10.13 b) ten sam przebieg jest wyrażony przez 400 próbek. Zmianę między rysunkami 10.13 a) i 10.13 b) można rozpatrywać jednym z dwóch sposobów: 1) utrzymanie stałej częstotliwości próbkowania i poddanie przebiegu oryginalnego ekspansji rozszerzającej go 8-krotnie, 2) utrzymanie niezmienionego przebiegu oryginalnego i zwiększenie częstotliwości próbkowania o czynnik 8. Sposób ze zmianą częstotliwości próbkowania nazywa się metodą **o wielu częstotliwościach próbkowania** (*multirate techniques*). Jeśli liczba próbek jest zwiększana, to mamy **interpolację**, a jeśli do reprezentowania sygnału wykorzystuje się mniejszą liczbę próbek, jest to **decymacja**. W rozdziale 3 opisano stosowanie metody o wielu częstotliwościach próbkowania w przetwornikach a/c i c/a.

I tu pojawia się problem. Jeśli mamy dowolny sygnał dyskretny, to skąd wiemy, jaki jest przebieg oryginalny? Zależy to od dziedziny, w której jest zakodowana informacji – *dziedziny czasu* lub *częstotliwości*. Dla sygnałów zakodowanych w dziedzinie czasu chcemy, aby oryginalny przebieg ciągły był gładką krzywą przechodzącą przez wszystkie próbki. W najprostszym przypadku możemy przeprowadzić linie proste między punktami i zaokrąglić ostre narożniki. Metodą bardziej zaawansowaną jest zastosowanie algorytmów dopasowania krzywej, takich jak funkcja składana lub dopasowanie wielomianowe. Nie ma żadnego jednego „poprawnego" rozwiązania tego problemu. Takie podejście jest oparte na minimalizacji nieregularności w przebiegu w *dziedzinie czasu* z całkowitym pominięciem dziedziny częstotliwości.



Rys. 10.13. Interpolacja przez dopełnianie przebiegu w dziedzinie częstotliwości. Przebiegi na rysunkach a) i c) zawierają po 50 próbek. Następuje ich interpolacja do 400 próbek przez dopełnianie zerami przebiegu w dziedzinie częstotliwości, co daje w rezultacie przebiegi przedstawione, odpowiednio, na rys. b) i d). Sygnały na rysunkach b) i d) są dyskretne, lecz narysowano je jako linie ciągłe z powodu dużej liczby próbek

Gdy sygnał jest zakodowany w dziedzinie częstotliwości, pomijamy przebieg w dzied nie czasu i koncentrujemy uwagę na widmie częstotliwościowym. Jak powiedziano w po przednim rozdziale, próbkowanie widma częstotliwościowego w mniejszych odstęp (czyli więcej próbek między częstotliwościami 0 i 0,5) można uzyskać przez dopełn zerami sygnału w dziedzinie czasu, przed poddaniem go przetwarzaniu DFT. Dualn powoduje, że stwierdzenie odwrotne też jest słuszne. Jeśli chcemy zrealizować pr kowanie w mniejszych odstępach w dziedzinie czasu (interpolacja), to trzeba dopeł widmo częstotliwościowe zerami, przed wykonaniem odwrotnego przekształcenia DFT Powiedzmy, że chcemy interpolować sygnał o 50 próbkach na sygnał o 400 próbk Można to zrobić w następujący sposób: 1) Bierzemy 50 próbek i dodajemy zera, aby utworzyć sygnał o długości 64 próbek. 2) Przeprowadzamy 64-punktowe przekształc DFT w celu wyznaczenia widma częstotliwościowego, które będzie zawierać 33-punkto wą część rzeczywistą i 33-punktową część urojoną. 3) Dopełniamy prawą stronę widm częstotliwościowego (zarówno część rzeczywistą, jak i urojoną) 224 zerami, aby widm uzyskało długość 257 punktów. 4) Wykonujemy 512-punktowe odwrotne przekształcen Fouriera, aby przenieść dane z powrotem do dziedziny czasu. W rezultacie uzyskuj sygnał złożony z 512 próbek, który jest wersją sygnału o 64 próbkach o większej ro dzielczości. Pierwsze 400 próbek tego sygnału jest interpolowaną wersją oryginalne sygnału o 50 próbkach.

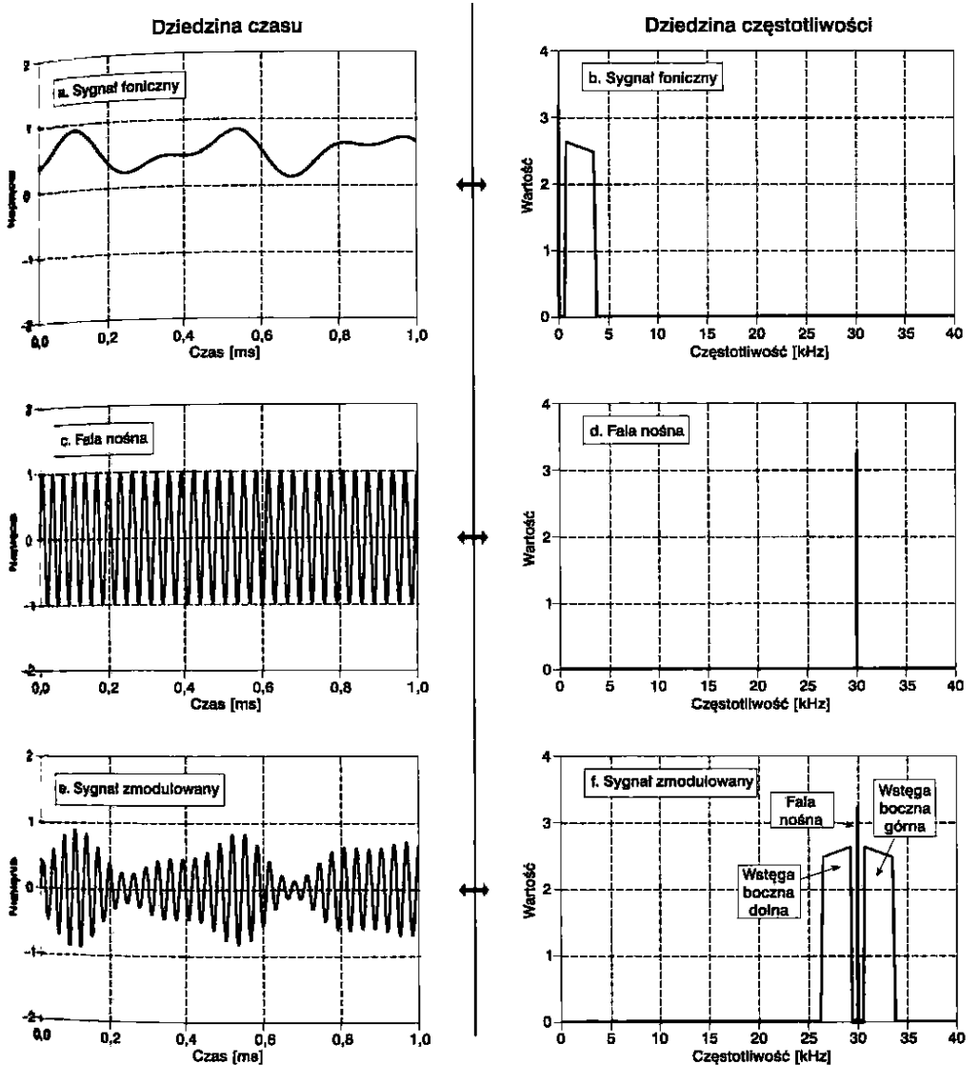
Najbardziej istotną właściwością metody jest to, że sygnał interpolowany zawiera *dokład* te same częstotliwości, co sygnał oryginalny. Może on dobrze pasować albo nie pasować dziedziny czasu. Na przykład na rysunkach 10.13 a), b) przedstawiono 50-próbkowy sygna interpolowany w 400-próbkowy przy użyciu omawianej metody. Ta interpolacja jest gładk dopasowaniem między punktami oryginalnymi, tak jakby zastosowano procedurę dopasowa nia krzywej. Na rysunkach 10.13 c), d) zaś przedstawiono inny przykład, gdzie w dziedzi nie czasu powstaje bałagan. Zjawiska oscylacyjne pokazane na rys. 10.13 d) powstają zboczach oraz przy innych nieciągłościach sygnału. Dotyczy to także nieciągłości między próbkami 0 i $N-1$, ponieważ przebiegi w dziedzinie czasu traktuje się jako kołowe. Przeregu przy nieciągłościach jest nazywany *efektem Gibbsa* i będzie omówiony w rozdziale 11, a metodę interpolacji w dziedzinie częstotliwości przedstawiono w rozdziale 3. Polega ona dodawaniu zer między próbkami w dziedzinie czasu i filtracji dolnoprzepustowej.

Mnożenie sygnałów (modulacja amplitudy)

Splot w jednej dziedzinie odpowiada *mnożeniu* w drugiej. Jest to ważna cecha przekształcenia Fouriera. Jedną część tego stwierdzenia omówiono w poprzednim rozdziale: sygnały w dziedzinie czasu mogą być splatane ze sobą przez mnożenie ich widm częstotliwościowych. Modulacja amplitudy jest przykładem sytuacji odwrotnej – mnożenie w dziedzinie czasu odpowiada splotowi w dziedzinie częstotliwości. Modulacja amplitudy jest, pomimo świetnym przykładem wprowadzenia nieuchwytnych częstotliwości *ujemnych* do codziennych zagadnień naukowych i problemów inżynierskich.

Sygnały foniczne świetnie nadają się do porozumiewania na krótkie odległości; gdy mówimy, ktoś z drugiej strony pokoju nas słyszy. Natomiast częstotliwości radiowe bardzo dobrze rozprzestrzeniają się na duże odległości. Jeśli, na przykład, wprowadzimy do anteny radiowej 100-woltowy sygnał sinusoidalny o częstotliwości 1 MHz, to powstająca fala radiowa może być odebrana w sąsiednim *pokoju*, w sąsiednim *kraju* i nawet na sąsiedniej *planecie*. **Modulacja** jest procesem łączenia dwóch sygnałów w celu utworzenia trzeciego, mającego pożądane właściwości obu sygnałów. Taka operacja zawsze wymaga procesów nieliniowych takich, jak mnożenie; nie można po prostu dodać obu sygnałów do siebie. W łączności radiowej modulacja daje sygnały radiowe, które mogą rozchodzić się na duże odległości i przenosić sygnały foniczne i inne informacje.

Łączność radiowa jest dziedziną wyjątkowo dobrze rozwiniętą. Opracowano wiele metod modulacji. Jedna z prostszych nosi nazwę **modulacji amplitudy**. Na rys. 10.14 przedstawiono przykład funkcjonowania modulacji amplitudy zarówno w dziedzinie czasu, jak i częstotliwości. W tym przykładzie użyto sygnałów ciągłych, gdyż ta modulacja jest stosowana głównie w elektronice analogowej. Jednak, jeśli trzeba, to cała ta procedura może być przeprowadzona również w postaci dyskretnej (pieśń przyszłości!).



Ilyc. 10.14. Modulacja amplitudy. W dziedzinie czasu modulację amplitudy realizuje się przez mnożenie sygnału fonicznego (rys. a) przez falę nośną (rys. c) w celu utworzenia sygnału zmodulowanego (rys. e). Ponieważ mnożenie w dziedzinie czasu odpowiada splotowi w dziedzinie częstotliwości, więc widmo sygnału zmodulowanego jest widmem sygnału fonicznego przesuniętym do częstotliwości nośnej

Na rys. 10.14 a) przedstawiono sygnał foniczny ze składową stałą, dzięki której sygnał zawsze ma wartość dodatnią. Jego widmo (rys. 10.14 b) zawiera częstotliwości od 300 Hz do 3 kHz, a więc zakres niezbędny do łączności głosowej oraz szpilkę reprezentującą składową stałą. Wszystkie inne częstotliwości usunięto przez filtrację analogową. Na rysunkach 10.14 c), d) przedstawiono **falę nośną**, czystą sinusoidę o częstotliwości znacznie większej niż sygnału fonicznego. W dziedzinie czasu modulacja jest *mnożeniem* sygnału fonicznego przez falę nośną. W wyniku mnożenia powstaje przebieg oscylacyjny (rys. 10.14 e) o chwilowej wartości amplitudy proporcjonalnej do oryginalnego sygnału fonicznego. Mówi się więc, że *obwiednia* przebiegu nośnego jest równa sygnałowi modulującemu. Uzyskany sygnał może być doprowadzony do anteny, przetworzony na fale radiowe a potem wykryty przez antenę odbiorczą. W rezultacie, w układach elektronicznych odbiornika radiowego uzyskuje się sygnał identyczny z przebiegiem z rys. 10.14 a). Następnie w *detektorze*, czyli *demodulatorze* przebieg zmodulowany jest z powrotem przetwarzany w sygnał foniczny, zgodny z sygnałem na rys. 10.14 a).

Ponieważ sygnały w dziedzinie czasu są mnożone, więc odpowiadające im widma częstotliwościowe są ze sobą splatane. Tak więc przebieg na rys. 10.14 f) jest wyznaczany jako splot przebiegów na rysunkach 10.14 b), d). Widmo fali nośnej jest przesuniętą funkcją delta, dlatego widmo sygnału zmodulowanego jest równe widmu sygnału fonicznego *przesuniętemu* do częstotliwości fali nośnej. W rezultacie widmo sygnału zmodulowanego składa się z trzech części: **fali nośnej**, **wstęgi bocznej górnej** i **wstęgi bocznej dolnej**.

Ta zawartość widma odpowiada trzem częściom oryginalnego sygnału fonicznego: składowej stałej, częstotliwościom dodatnim od 0,3 do 3 kHz i ujemnym od –0,3 do –3 kHz. Chociaż częstotliwości ujemne w oryginalnym sygnale fonicznym są czymś nieuchwytnym i abstrakcyjnym, to wynikające z nich częstotliwości dolnej wstęgi bocznej są jak najbardziej rzeczywiste. Widma przyjmują ludzką postać!

Tego rodzaju analizy są treścią życia specjalistów z dziedziny telekomunikacji. Rozpatrzmy na przykład widmo częstotliwościowe transmitowanego sygnału telewizyjnego. Standardowy sygnał TV ma widmo od składowej stałej do 6 MHz. Stosując metodę przesuwania częstotliwości, układa się obok siebie 82 takie kanały o szerokości 6 MHz. Na przykład kanał 3 zajmuje pasmo od 60 do 66 MHz, kanał 4 – od 66 do 72 MHz, a kanał 83 od 884 do 890 MHz itd. W odbiorniku telewizyjnym następuje przesuwanie wybranego kanału z powrotem do pasma od 0 do 6 MHz, aby można było wyświetlić obraz na ekranie. Jest to **multipleksowanie w dziedzinie częstotliwości**.

Dyskretne w czasie przekształcenie Fouriera

Dyskretne w czasie przekształcenie Fouriera (DTFT – *Discrete Time Fourier Transform*) jest jednym z rodziny przekształceń fourierowskich i jest przeznaczone do *nieokresowych sygnałów dyskretnych*. Najlepszym sposobem zrozumienia DTFT jest rozważenie jego związku z DFT. Na początku załóżmy, że zebraliśmy sygnał o N próbkach i chcemy znaleźć jego widmo częstotliwościowe. Stosując DFT, można rozłożyć sygnał na $N/2 + 1$ fal sinusoidalnych i kosinusoidalnych o częstotliwościach równomiernie rozmieszczonych w zakresie od 0 do połowy częstotliwości próbkowania. Jak to wynika z rozważań w poprzednim rozdziale, dopełnianie zerami sygnału w dziedzinie czasu *wydłuża* okres, a w dziedzinie częstotliwości *zwęża* odstępy między próbkami. Gdy N dąży do nieskończoności, sygnał w dziedzinie czasu staje się *nieokresowy*, a w dziedzinie częstotliwości staje się sygnałem *ciągłym*. To jest właśnie DTFT, przekształcenie Fouriera określające zależność między sygnałem *nieokresowym*, *dyskretnym* w dziedzinie czasu a *okresowym* widmem *ciągłym* w dziedzinie częstotliwości.

Matematyczne zależności DTFT można zrozumieć, wychodząc z syntezy i analizy wzorów opisujących DFT (wzory 8.2, 8.3 i 8.4) i zakładając, że N dąży do nieskończoności.

$$Re\,X(\omega)=\sum_{n=-\infty}^{+\infty} x[n]\cos(\omega n)$$

$$Im\,X(\omega)=\sum_{n=-\infty}^{+\infty} x[n]\sin(\omega n)$$

Wzór 10.1. Analiza DTFT. W tej zależności x[n] jest sygnałem w dziedzinie czasu z n zmieniającym się od 0 do N – 1. Widmo częstotliwościowe jest wyrażone przez Re X(ω) i Im X(ω), przy czym ω zawiera się od 0 do π

$$x[n]=\frac{1}{\pi}\int_0^{\pi}\left(Re\,X(\omega)\cos(\omega n)-Im\,X(\omega)\sin(\omega n)\right)d\omega$$

Wzór 10.2. Synteza DTFT

W powyższych zależnościach jest wiele subtelności. Po pierwsze, sygnał w dziedzinie czasu $x[n]$ jest jeszcze dyskretny i dlatego zawiera nawiasy *kwadratowe*. Natomiast sygnały w dziedzinie częstotliwości *Re* $X(\omega)$ i *Im* $X(\omega)$ są ciągłe i dlatego mają nawiasy *okrągłe*. Ponieważ sygnał w dziedzinie częstotliwości jest ciągły, więc wzór dotyczący syntezy jest zapisany jako całka, a nie jako sumowanie.

Było już powiedziane w rozdziale 8, że w przekształceniu DFT w dziedzinie częstotliwości częstotliwość jest wyrażana przez jedną z trzech zmiennych: k – indeks zmieniający się od 0 do $N/2$, f – ułamek częstotliwości próbkowania zmieniający się od 0 do 0,5 lub ω – ułamek częstotliwości próbkowania wyrażony w pulsacji i zmieniający się od 0 do π. Widmo DTFT jest ciągłe, więc można stosować zarówno f, jak i ω. Na ogół stosuje się ω, ponieważ wzory są wtedy krótsze, bez czynnika 2π. Pamiętajmy, że przy stosowaniu ω widmo częstotliwościowe obejmuje zakres od 0 do π, co odpowiada zakresowi od 0 do połowy częstotliwości próbkowania. Chyba po to, żeby wszystko jeszcze bardziej skomplikować, niektórzy autorzy dla częstotliwości w DTFT stosują oznaczenie Ω, zamiast ω.

Przy obliczaniu odwrotnej transformaty DFT wartości próbek o numerach 0 i $N/2$ muszą być dzielone przez 2 (wzór 8.3) przed wykonaniem syntezy według wzoru 8.2. Nie jest to konieczne w DTFT. Jak pamiętamy, ta operacja w DFT odnosi się do widma częstotliwościowego definiowanego przez *gęstość widmową*, tj. amplitudę na pasmo jednostkowe. Gdy widmo staje się ciągłe, specjalne traktowanie skrajnych punktów przestaje obowiązywać. Nadal jednak pozostaje konieczność użycia czynnika normalizacyjnego. Czynnik $2/N$ w DFT (wzór 8.3) staje się w DTFT czynnikiem $1/\pi$ (wzór 10.2). Niektórzy autorzy umieszczają ten czynnik przed wzorem *syntezy*, a inni – przed wzorem *analizy*. Przypuśćmy, że zaczynamy od jakiegoś sygnału w dziedzinie czasu. Po znalezieniu transformaty Fouriera i następnie po wykonaniu odwrotnego przekształcenia Fouriera, spodziewamy się dojść do tego samego sygnału, od jakiego rozpoczęliśmy. Zatem powinien gdzieś po drodze pojawić się czynnik $1/\pi$ (lub $2/N$), w przekształceniu prostym albo odwrotnym. Niektórzy autorzy nawet rozdzielają ten czynnik między dwoma przekształceniami, umieszczając czynnik $1/\sqrt{\pi}$ przed obu wzorami.

Przekształcenie DTFT nie może być obliczane w komputerze cyfrowym, gdyż zawiera całkowania i sumowania nieskończone. Jest stosowane głównie w problemach teoretycznych jako alternatywa DFT. Przypuśćmy na przykład, że chcemy wyznaczyć odpowiedź częstotliwościową systemu na podstawie jego odpowiedzi impulsowej. Jeśli odpowiedź impulsowa jest znana jako *matryca liczb*, uzyskana np. z pomiarów doświadczalnych lub symulacji komputerowej, to program DFT może być realizowany w komputerze. Otrzymuje się wtedy inną *matrycę liczb*, równomiernie rozmieszczonych między 0 i 0,5 częstotliwości próbkowania.

W innych przypadkach odpowiedź impulsowa może być znana w postaci *wzoru*, np. jako funkcja *sinc* (opisana w następnym rozdziale) lub jako zanikająca wykładniczo sinusoida. Wtedy DFT jest stosowane do matematycznego obliczenia sygnału w dziedzinie częstotliwości w formie innego *wzoru* wyrażającego całą krzywą ciągłą między 0 i 0,5. Przekształcenie DFT też mogłoby być stosowane do tych obliczeń, lecz dałoby wzór wyrażający *próbki* odpowiedzi częstotliwościowej, a nie całą krzywą.

Zależność Parsevala

Ponieważ w dziedzinach czasu i częstotliwości występują równoważne reprezentacje tego samego sygnału, więc powinny one mieć taką sama *energię*. Jest to nazwane zależnością Parsevala obowiązującą we wszystkich przekształceniach z rodziny transformat fourierowskich. Dla przekształcenia DFT zależność Parsevala ma postać:

$$\sum_{i=0}^{N-1} x[i]^2 = \frac{2}{N}\sum_{k=0}^{N/2} MagX[k]^2$$

Wzór 10.3. Zależność Parsevala. W tym wzorze x[i] jest sygnałem w dziedzinie czasu, gdzie i zmienia się od 0 do N – 1, a X[k] jest jego zmodyfikowanym widmem częstotliwościowym, gdzie k zmienia się od 0 do N/2. Zmodyfikowane widmo częstotliwościowe jest znajdowane przez przeprowadzenie przekształcenia DFT sygnału i podzielenie pierwszej i ostatniej częstotliwości widma (próbki 0 i N/2) przez pierwiastek kwadratowy z 2

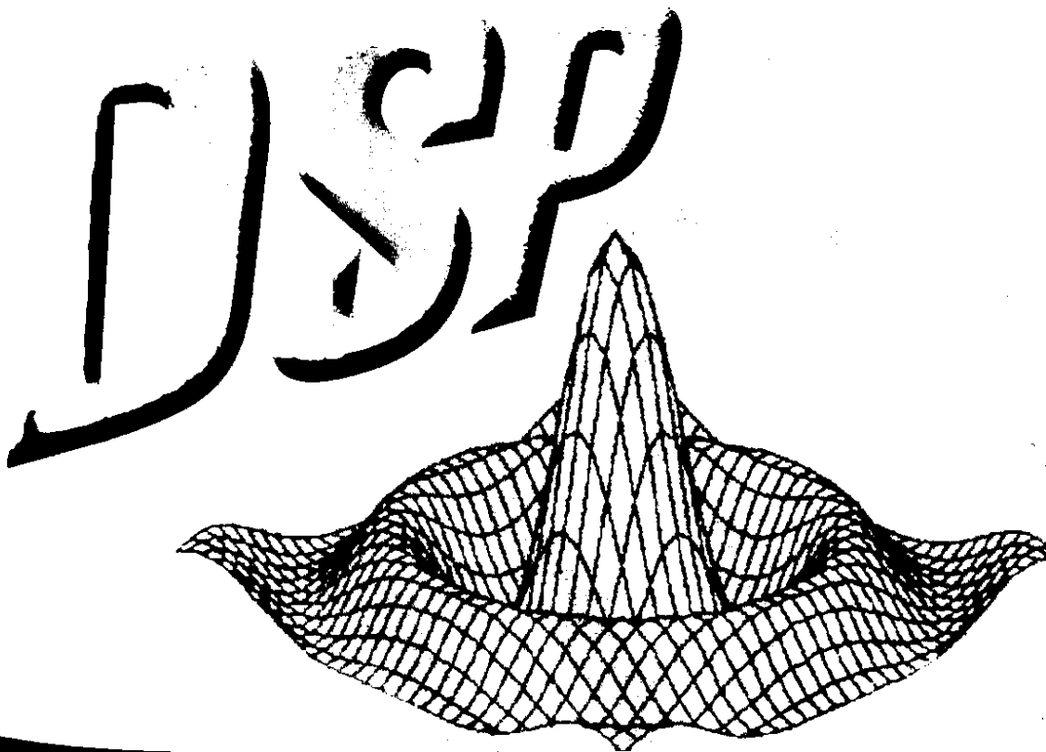
Lewa strona powyższego równania wyraża całkowitą energię zawartą w sygnale w dziedzinie czasu, obliczoną przez sumowanie energii N indywidualnych próbek. Prawa strona równania w podobny sposób wyraża energię zawartą w sygnale w dziedzinie częstotliwości, obliczoną przez sumowanie energii $N/2 + 1$ sinusoid. Przypomnijmy sobie z fizyki, że *energia* jest proporcjonalna do *kwadratu wartości* próbki, na przykład energia zmagazynowana w sprężynie jest proporcjonalna do *kwadratu* odkształcenia, a zgromadzona w kondensatorze – do kwadratu napięcia. We wzorze 10.3 $X[f]$ jest widmem częstotliwościowym $x[n]$, z jedną niewielką modyfikacją: pierwsza i ostatnia składowa częstotliwości $X[0]$ i $X[N/2]$ zostały podzielone przez $\sqrt{2}$. Ta modyfikacja, łącznie z czynnikiem $2/N$ z prawej strony równania, zalicza się do tych kilku niuansów obliczania i sumowania energii.

W celu zrozumienia tych modyfikacji zacznijmy od wyznaczenia reprezentacji sygnału w dziedzinie częstotliwości przy użyciu DFT. Następnie przetwórzmy sygnał w dziedzinie częstotliwości w amplitudy sinusoid koniecznych do odtworzenia sygnału, zgodnie z wzorem 8.3. Wykonuje się to przez podzielenie pierwszego i ostatniego punktu (próbki 0 i $N/2$) przez 2 i następnie podzielenie wszystkich próbek przez $N/2$. W ten sposób uzyskuje się amplitudy sinusoid, a więc ich wartości *szczytowe*, a nie wartości skuteczne potrzebne do obliczania energii. W przebiegu sinusoidalnym wartość szczytowa jest przetwarzana na skuteczną w wyniku podzielenia przez $\sqrt{2}$. Ta poprawka musi być wprowadzona dla wszystkich wartości w dziedzinie częstotliwości *oprócz* próbek 0 i $N/2$. Jest tak, gdyż te sinusoidy są wyjątkowe; jedna jest wartością stałą, a druga zawiera zmiany między dwiema wartościami stałymi. W tych dwóch przypadkach szczególnych wartość *szczytowa* jest równa *skutecznej*. Wszystkie wartości w dziedzinie częstotliwości są podnoszone do kwadratu i sumowane. Ostatnim krokiem jest podzielenie uzyskanej sumy przez N ze względu na to, że każda próbka w dziedzinie częstotliwości jest przetwarzana na sinusoidę, która pokrywa N wartości w dziedzinie czasu. Uwzględnienie tych wszystkich szczegółów prowadzi do sformułowania wzoru 10.3.

Wprawdzie zależność Parsevala jest interesująca ze względu na opisywane zjawisko fizyczne (zachowanie energii), lecz jej praktyczne zastosowanie w DSP jest niewielkie.

11

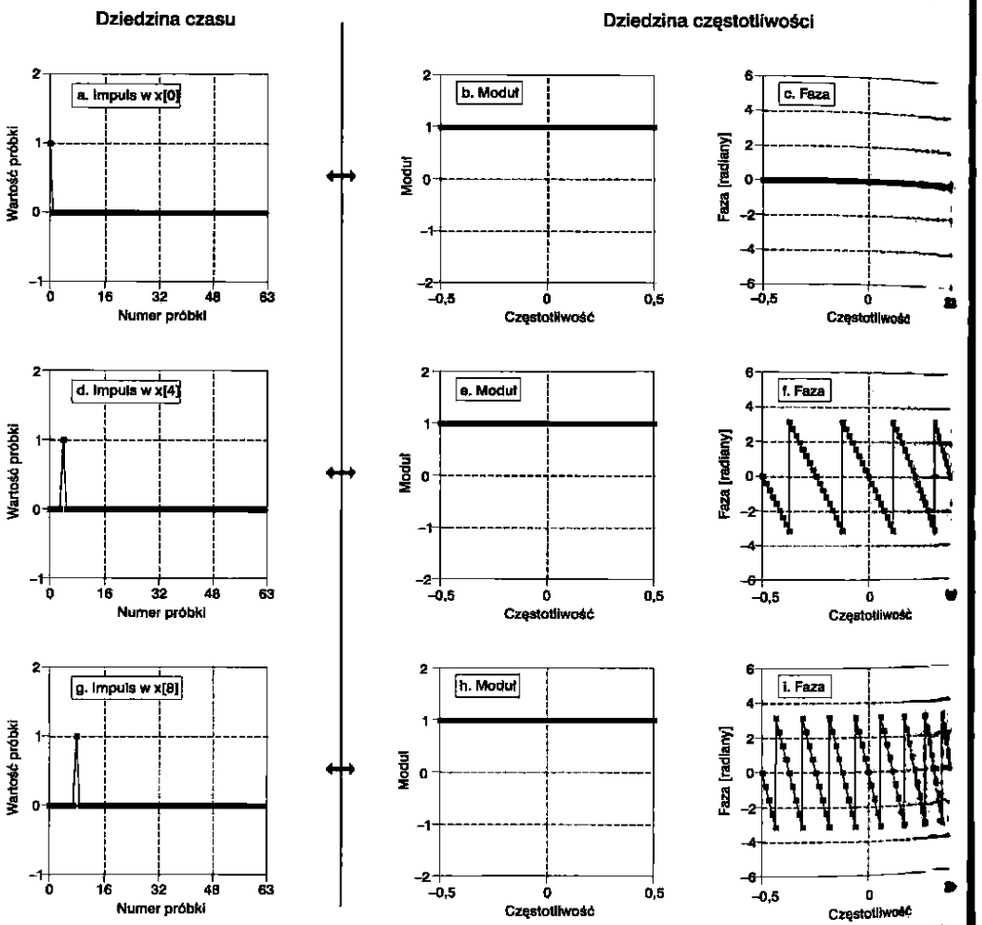
Pary transformat Fouriera



Każdy przebieg w dziedzinie czasu ma swój odpowiednik w dziedzinie częstotliwości i vice versa. Na przykład impulsowi prostokątnemu w dziedzinie czasu odpowiada w dziedzinie częstotliwości funkcja $sin(x)/x$, oznaczana też przez $sinc(x)$. Dualność powoduje, że prawdziwe jest też stwierdzenie odwrotne: odpowiednikiem impulsu prostokątnego w dziedzinie częstotliwości jest w dziedzinie czasu funkcja $sin(x)/x$. Przebiegi będące wzajemnie takimi odpowiednikami nazywamy *parami transformat Fouriera*. W tym rozdziale będzie przedstawionych kilka takich par.

Para transformat funkcji delta

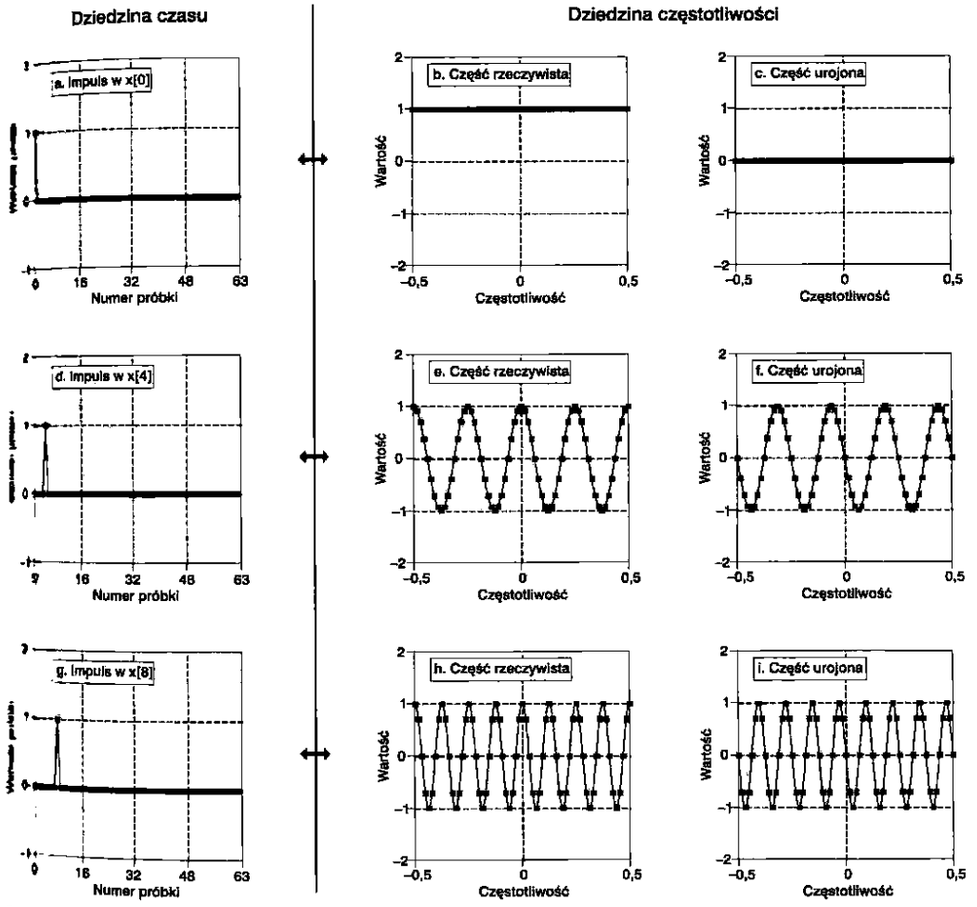
Wśród sygnałów dyskretnych funkcja delta jest prostym przebiegiem i ma równie prostą parę transformat Fouriera. Przebieg funkcji delta w dziedzinie czasu przedstawiono na rys. 11.1, a widmo częstotliwościowe – na rysunkach 11.1 b), c). Moduł ma wartość stałą, faza zaś jest zerowa. Można to wyjaśnić na podstawie właściwości ekspansji/kompresji, o czym była mowa w poprzednim rozdziale. Gdy przebieg w dziedzinie czasu jest poddawany kompresji, aż do stania się impulsem, to sygnał w dziedzinie częstotliwości ulega ekspansji, aż staje się wartością stałą.



Rys. 11.1. Para transformat funkcji delta we współrzędnych biegunowych. Impuls w dziedzinie czasu odpowiada stałemu modułowi i liniowej charakterystyce fazowej w dziedzinie częstotliwości

Na rysunkach 11.1 d), g) przebieg w dziedzinie czasu został przesunięty o cztery i o osiem próbek w prawo. Uwzględniając właściwości rozpatrywane w poprzednim rozdziale, wiadomo, że przesunięcie przebiegu w dziedzinie czasu nie wpływa na moduł, lecz dodaje składową liniową do charakterystyki fazowej. Charakterystyki fazowe przedstawione na rysunku nie zostały *rozwinięte*, więc obejmują tylko zakres od $-\pi$ do π. Zauważmy też, że oś pozioma w dziedzinie częstotliwości ma zakres –0,5 do 0,5. To znaczy, że w widmie są pokazane zarówno częstotliwości *ujemne*, jak i *dodatnie*. Częstotliwości ujemne są informacją redundancyjną, lecz często uwzględnia się je na wykresach w DSP. Trzeba się więc przyzwyczaić do ich oglądania.

Na rys. 11.2 podano te same informacje, co na rys. 11.1, lecz w dziedzinie częstotliwości zastosowano współrzędne *prostokątne*. Z tych wykresów można się nauczyć dwóch rzeczy. Po pierwsze, porównajmy sygnały w dziedzinie częstotliwości przedstawione we współrzędnych biegunowych i w prostokątnych. Wykresy we współrzędnych biegunowych są, jak zwykle, łatwiejsze do zrozumienia. Moduł jest wartością stałą, faza zaś linią prostą. We współrzędnych prostokątnych zaś części rzeczywista i urojona są oscylacjami sinusoidalnymi, którym trudno przypisać jakieś znaczenie.



Rys. 11.2. Para transformat funkcji delta we współrzędnych prostokątnych. Każda próbka w dziedzinie czasu daje falę sinusoidalną w części rzeczywistej i ujemną falę sinusoidalną w części urojonej w dziedzinie częstotliwości

Drugą interesującą sprawą na rys. 11.2 jest *dualność* DFT. Wiadomo, że w DFT ko... dej próbce w dziedzinie częstotliwości odpowiada sinusoida w dziedzinie czasu. Jed... słuszna jest też zależność odwrotna: każdej próbce w dziedzinie czasu odpowiada sin... ida w dziedzinie częstotliwości. Włączenie do wykresów częstotliwości ujemnych po... duje, że dualność staje się bardziej symetryczna. Na przykład z rysunków 11.2 d), e) i f) wynika, że impuls w próbce numer 4 w dziedzinie czasu daje 4 okresy fali kosinu... idalnej w części rzeczywistej fali widma częstotliwościowego i 4 okresy ujemnej fali sinusoidalnej w części urojonej. Zgodnie z zasadą dualności, impuls w próbce numer 4 w części rzeczywistej widma częstotliwościowego daje cztery okresy fali kosinusoidalnej w dziedzinie czasu. W podobny sposób impuls w próbce numer 4 w części urojonej widma częstotliwościowego daje 4 okresy ujemnej fali sinusoidalnej dodawane do fali w dziedzinie czasu.

Jak wspomniano w rozdziale 8, powyższe właściwości można wykorzystać jako inny spos... obliczania DFT (oprócz korelacji w dziedzinie czasu z sinusoidami). Każda próbka w dzie... dzinie czasu daje w wyniku falę kosinusoidalną dodawaną do części rzeczywistej w dziedzi... nie częstotliwości oraz ujemną falę sinusoidalną dodawaną do części urojonej. *Amplituda* każdej sinusoidy jest określona przez *amplitudę* próbki w dziedzinie czasu. *Częstotliwość* każdej sinusoidy jest wyznaczona przez *numer próbki* punktu w dziedzinie czasu. Algorytm zawiera: 1) krokowe przechodzenie przez wszystkie próbki w dziedzinie czasu, 2) obliczanie fal sinusoidalnych i kosinusoidalnych odpowiadających kolejnym próbkom oraz 3) sumowanie wszystkich sinusoid składowych. W rezultacie program jest prawie identyczny z programem w metodzie korelacyjnej (tablica 8.2), przy czym pętle zewnętrzna i wewnętrzna są zamienione.

Funkcja *sin(x)/x*

Na rys. 11.3 przedstawiono często spotykaną parę transformat: *impuls prostokątny* i *funkcję sin(x)/x*. Ta funkcja jest też oznaczana jako *sinc(x)*. Funkcja opisuje sinusoidę, której amplituda zmniejsza się proporcjonalnie do $1/x$. Na rys. 11.3 a) impuls prostokątny jest umieszczony symetrycznie wokół próbki zero, w wyniku czego połowa impulsu znajduje się z prawej strony wykresu, a druga połowa – z lewej. W przekształceniu DFT jest to traktowane jako pojedynczy impuls ze względu na okresowość w dziedzinie czasu. Transformaty DFT tego sygnału przedstawiono na rysunkach 11.3 b), c), a w wersji *rozwiniętej* – na rysunkach 11.3 d), e).

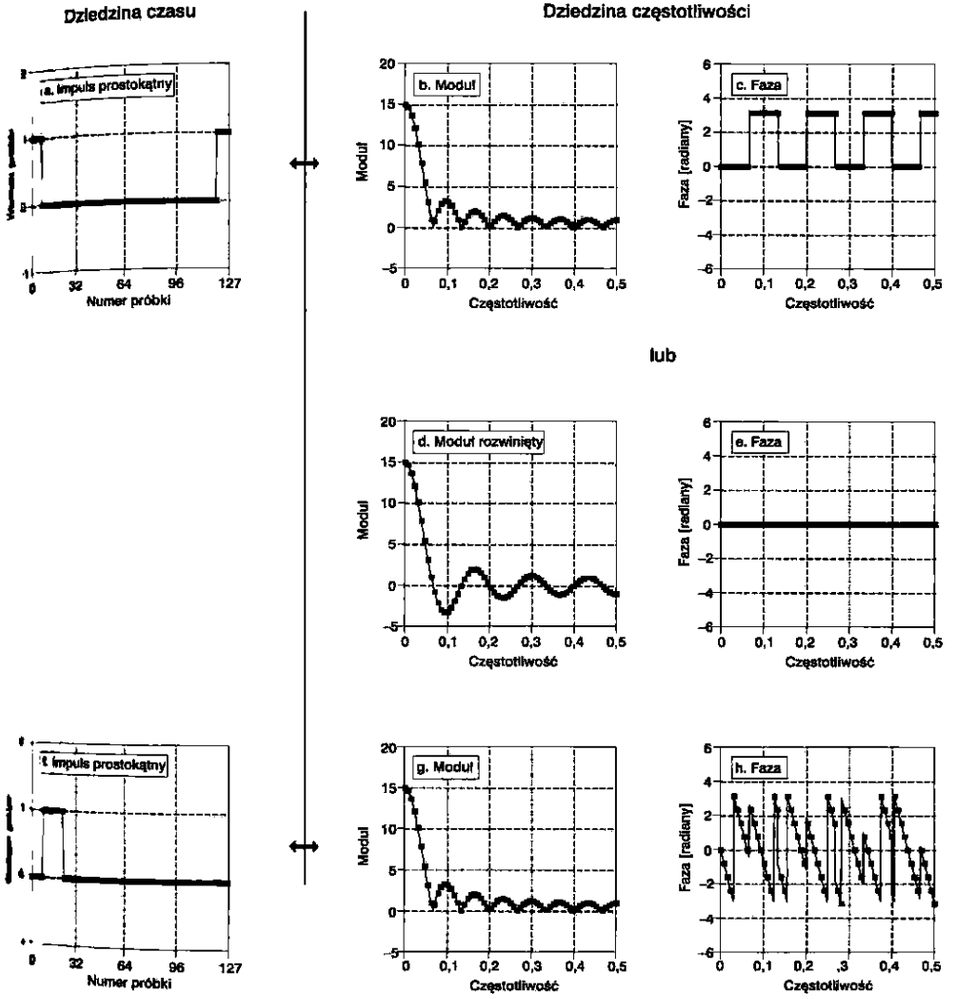
Spójrzmy najpierw na widmo rozwinięte (rysunki 11.3 d, e). *Moduł rozwinięty* jest oscylacją o amplitudzie zmniejszającej się w funkcji częstotliwości. Charakterystyka fazy jest złożona z samych zer, czego można było oczekiwać z powodu symetrii sygnału w dziedzinie czasu wokół próbki numer 0. Używamy nazwy *moduł rozwinięty*, aby zaznaczyć, że może on mieć wartości zarówno dodatnie, jak i ujemne. *Moduł* z definicji powinien być zawsze dodatni, jak to przedstawiono na rysunkach 11.3 b), c), gdzie uzyskuje się w całym zakresie moduł dodatni przez wprowadzenie przesunięcia fazy równego π dla wszystkich częstotliwości, przy których rozwinięty moduł jest ujemny na rys. 11.3 d).

Impuls prostokątny na rys. 11.3 f) jest przesunięty tak, że ma wygląd pojedynczego impulsu lecz nie jest już symetryczny w stosunku do próbki numer 0. Wprawdzie takie przesunięcie nie zmienia charakterystyki modułu w dziedzinie częstotliwości, lecz dodaje składową liniową do charakterystyki fazy, powodując w niej zupełny bałagan. A jak wygląda widmo częstotliwościowe w postaci części rzeczywistej i urojonej? Zbyt zagmatwane, żeby w ogóle się nim zajmować.

Złożony z N punktów sygnał w dziedzinie czasu, który zawiera impuls prostokątny o amplitudzie jednostkowej i o szerokości M punktów, ma widmo częstotliwościowe DFT określone wzorem:

$$Mag\,X[k]=\left|\frac{\sin\left(\frac{\pi k M}{N}\right)}{\sin\left(\frac{\pi k}{N}\right)}\right|$$

Wzór 11.1. Widmo DFT impulsu prostokątnego. W tym wzorze N jest liczbą punktów w sygnale w dziedzinie czasu. Wszystkie te punkty mają wartość zero, oprócz M sąsiadujących ze sobą punktów mających wartość jeden. Widmo częstotliwościowe jest zawarte w X[k], gdzie k zmienia się od 0 do N/2. Aby uniknąć dzielenia przez zero, zastosowano X[0] = M. Funkcja sin jest wyrażona w radianach, a nie w stopniach. W tym wzorze uwzględniono aliasing sygnału



Rys. 11.3. Przekształcenie DFT impulsu prostokątnego. Impuls prostokątny w jednej dziedzinie odpowiada funkcji sin(x)/x w drugiej

Alternatywnie można też zastosować przekształcenie DTFT do określenia widma częstotliwościowego w funkcji ułamków częstotliwości próbkowania *f*:

$$Mag\ X(f) = \left| \frac{\sin(\pi f M)}{\sin(\pi f)} \right|$$

Wzór 11.2. Widmo, jak we wzorze 11.1, lecz wyrażone w zależności od częstotliwości próbkowania. Parametr f jest ułamkiem częstotliwości próbkowania, zmieniającym się w sposób ciągły od 0 do 0,5. Aby uniknąć dzielenia przez zero, zastosowano Mag X(0) = M

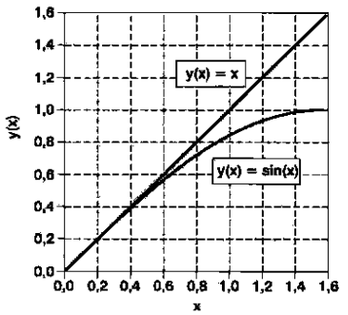
Innymi słowy, wzór 11.1 daje $N/2 + 1$ *próbek* widma częstotliwościowego, wzór 11.2 zaś daje *krzywą ciągłą*, która pokrywa się z próbkami. Na podstawie tych wzorów można wyznaczyć tylko moduł. Faza może być określona tylko przez lewo-prawostronne odwrócenie przebiegu w dziedzinie czasu, jak to opisano w poprzednim rozdziale.

Zauważmy, że amplituda oscylacji na rys. 11.3 b) nie zanika do zera przed osiągnięciem częstotliwości 0,5. Jak się można było spodziewać, przebieg nadal zachodzi na następny okres i tam ulega *aliasingowi*. To zmienia kształt charakterystyki częstotliwościowej; ten efekt jest uwzględniony we wzorach 11.1 i 11.2.

Często warto wiedzieć, jak wygląda widmo częstotliwościowe, gdy *nie ma* aliasingu. Ma to znaczenie, gdyż *sygnały dyskretne* są często stosowane do reprezentowania lub modelowania *sygnałów ciągłych*, a sygnały ciągłe nie ulegają aliasingowi. Aby usunąć aliasing z wzorów 11.1 i 11.2, trzeba w nich zmienić mianowniki, odpowiednio z $\sin(\pi k/N)$ na $\pi k/N$ oraz $\sin\pi f$ na πf. Na rys. 11.4 zilustrowano znaczenie tych zmian. Wyraz πf może się zmieniać tylko od 0 do 1,5708, ponieważ f może zmieniać się od 0 do 0,5. W tym zakresie nie ma dużej różnicy między przebiegami $\sin(\pi f)$ i πf. Przy częstotliwości zerowej wartości obu przebiegów są jednakowe, a dla częstotliwości 0,5 jest tylko 36% różnicy. Bez aliasingu krzywa z rys. 11.3 b) miałaby nieco mniejszą amplitudę w pobliżu prawego krańca wykresu, a powinna stałaby niezmieniona przy lewym krańcu.

Widmo częstotliwościowe impulsu prostokątnego, jeśli *nie* ulega aliasingowi (ponieważ przebieg w dziedzinie czasu jest ciągły albo aliasing ignorujemy), to ma postać ogólną $\sin(x)/x$. Dla sygnałów ciągłych *impuls prostokątny* i *funkcja* $\sin(x)/x$ są parą transformat Fouriera. Dla sygnałów dyskretnych jest to tylko przybliżenie, z błędem spowodowanym aliasingiem.

W przebiegu $\sin(x)/x$ pojawia się kłopot w punkcie $x = 0$, wartość funkcji jest wówczas zero dzielonym przez *zero*. Nie jest to trudny problem matematyczny; gdy x staje się bardzo małe to $\sin(x)$ zbliża się do wartości x (rys. 11.4). To zmienia funkcję $\sin(x)/x$ w x/x, która ma wartość *jeden*. Inaczej mówiąc, gdy x staje się coraz mniejsze, wartość $\sin(x)/x$ dąży do jedności, więc dla $x = 0$ mamy $\sin(x)/x = 1$. Spróbuj teraz powiedzieć komputerowi, żeby to obliczył! Wszystko, co komputer zauważy, to dzielenie przez zero i wtedy zastopuje wykonywanie



Rys. 11.4. Porównanie przebiegów x oraz sin(x). Funkcje y(x) = x oraz y(x) = sin(x) są zbliżone do siebie dla małych wartości x i różnią się od siebie tylko o 36% dla x = 1,57 (π/2). Te przebiegi opisują, jakie odchylenie widma impulsu prostokątnego od czystej funkcji sinx/x wywołuje aliasing

waszego programu. Ważną kwestią, którą należy zapamiętać, jest uwzględnienie w programie specjalnej obsługi wartości $x = 0$ przy obliczaniu funkcji $\sin(x)/x$.

Podstawową cechą funkcji $\sin(x)/x$ jest położenie **przejść przez zero**. Takie przejścia występują przy częstotliwościach, przy których całkowita liczba okresów sinusoidy mieści się w sposób parzysty w impulsie prostokątnym. Jeśli, na przykład, impuls prostokątny ma szerokość 20 punktów, to pierwsze zero w dziedzinie częstotliwości jest przy częstotliwości, dla której jeden okres mieści się dokładnie w 20 punktach. Drugie zero jest przy częstotliwości dającej dwa pełen okresy w 20 punktach itd. Można to zrozumieć, przypominając sobie sposób obliczania DFT metodą korelacji. Amplitudę składowej częstotliwościowej wyznacza się, mnożąc sygnał w dziedzinie czasu przez sinusoidę i dodając uzyskane w ten sposób próbki. Jest to takie same działanie, jak *dodawanie* próbek sinusoidy mieszczących się w granicach impulsu prostokątnego, w przypadku gdy przebieg w dziedzinie czasu jest takim impulsem o amplitudzie jednostkowej. Jeśli to sumowanie odbywa się w granicach całkowitej liczby okresów sinusoidy, to wynikiem będzie zero.

Funkcja $\sin(x)/x$ jest szeroko stosowana w DSP, bo tworzy parę transformat Fouriera z bardzo prostym przebiegiem, jakim jest impuls prostokątny. Tę funkcję stosuje się na przykład w *analizie widmowej*, omówionej w rozdziale 9. Rozważmy analizę nieskończenie długiego sygnału dyskretnego. Przekształcenie DFT można stosować tylko do sygnałów o *skończonej* długości, dlatego trzeba wybrać N próbek reprezentujących dłuższy sygnał. Rzecz polega na tym, że „wybranie N próbek z dłuższego sygnału" jest tym samym, co pomnożeniem dłuższego sygnału przez impuls prostokątny. *Jedynki* w impulsie prostokątnym zachowują odpowiadające im próbki, *zera* zaś je eliminują. Jaki ma to wpływ na widmo częstotliwościowe sygnału? Mnożenie przebiegu w dziedzinie czasu przez impuls prostokątny odpowiada w dziedzinie częstotliwości *splotowi* z funkcją $\sin(x)/x$. Ogranicza to rozdzielczość widma częstotliwościowego, jak to poprzednio przedstawiono na rys. 9.5 a).

Inne pary transformat

Na rysunkach 11.5 a), b) przedstawiono wyżej omówioną dualność: impuls prostokątny w dziedzinie częstotliwości odpowiada funkcji $\sin(x)/x$ (plus aliasing) w dziedzinie czasu. Uwzględniając wpływ aliasingu, można sygnał w dziedzinie czasu zapisać jako:

$$x[i] = \frac{1}{N} \frac{\sin(2\pi i(M - 1/2)/N)}{\sin(\pi i/N)}$$

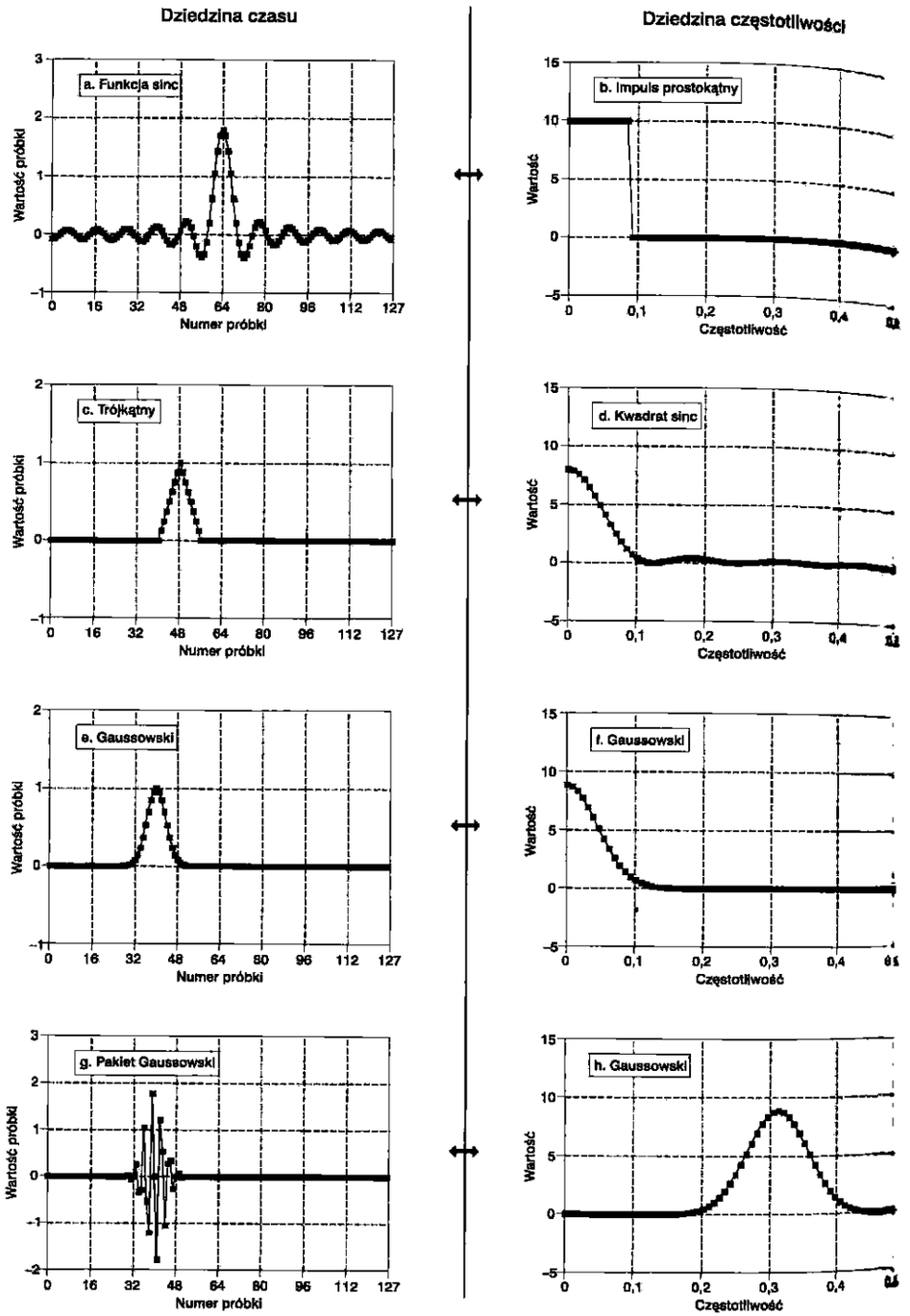
Wzór 11.3. Odwrotne przekształcenie DFT impulsu prostokątnego. W dziedzinie częstotliwości impuls ma amplitudę równą jedności i zawiera próbki od numeru 0 do M – 1. Parametr N jest długością DFT a x[i] jest sygnałem w dziedzinie czasu przy czym i zmienia się od 0 do N – 1. Aby uniknąć dzielenia przez zero, zastosowano x[0] = (2M – 1)/N

W celu wyeliminowania z tego wzoru wpływu aliasingu załóżmy, że sygnał w dziedzinie częstotliwości jest próbkowany z tak krótkim okresem próbkowania, że przemienia się w krzywą ciągłą. To powoduje, że przebieg w dziedzinie czasu staje się nieskończenie długi i nieokresowy. Trzeba tu zastosować przekształcenie DTFT, dające w wyniku sygnał w dziedzinie czasu wyrażony wzorem:

$$x[i] = \frac{\sin(2\pi f_c\, i)}{i\pi}$$

Wzór 11.4. Odwrotne przekształcenie DTFT impulsu prostokątnego. W dziedzinie częstotliwości impuls ma amplitudę równą jedności i obejmuje zakres od 0 do częstotliwości granicznej f_c o wartości między 0 i 0,5. Sygnałem w dziedzinie czasu jest x[i] przy czym i zmienia się od 0 do N – 1. Aby uniknąć dzielenia przez zero zastosowano $x[0] = 2f_c$

Ten wzór jest bardzo ważny w DSP, gdyż impuls prostokątny jest w dziedzinie częstotliwości idealnym *filtrem dolnoprzepustowym*. Zatem funkcja typu $\sin(x)/x$ występująca w tym równaniu jest odpowiedzią impulsową idealnego filtru dolnoprzepustowego. Jest to podstawą bardzo użytecznej klasy filtrów zwanych *filtrami okienkowanymi funkcją* $\sin(x)/x$, które będą opisane w rozdziale 15.



Rys.11.5. Często spotykane pary transformat

Na rysunkach 11.5 c), d) pokazano, że impuls trójkątny w dziedzinie czasu odpowiada w dziedzinie częstotliwości *kwadratowi* funkcji $sin(x)/x$ (plus aliasing). Tu ważna jest nie tyle ta para transformat, co raczej *przyczyna* jej słuszności. Trójkąt o długości $2M - 1$ punktów w dziedzinie czasu może być utworzony przez splot M-punktowego impulsu prostokątnego z samym sobą. Ponieważ splot w dziedzinie czasu daje mnożenie w dziedzinie częstotliwości, więc splatanie przebiegu z samym sobą powoduje *podnoszenie do kwadratu* w dziedzinie częstotliwości.

Czy istnieje jakiś przebieg, który jest swoją własną transformatą Fouriera? Tak, jest *tylko* jeden taki przebieg – gaussowski. Na rys. 11.5 e) pokazano krzywą Gaussa, a na rys. 11.5 f) – odpowiadające jej widmo częstotliwościowe, będące też krzywą gaussowską. Ta zależność jest słuszna tylko wtedy, gdy pomija się aliasing. Zależność między odchyleniami standardowymi w dziedzinach czasu i częstotliwości jest określona wzorem: $2\pi\sigma_f = 1/\sigma_t$. Na rys. 11.5 f) pokazano jedną część krzywej Gaussa; ujemne częstotliwości w widmie uzupełniają całą krzywą, której środkiem symetrii jest częstotliwość zerowa.

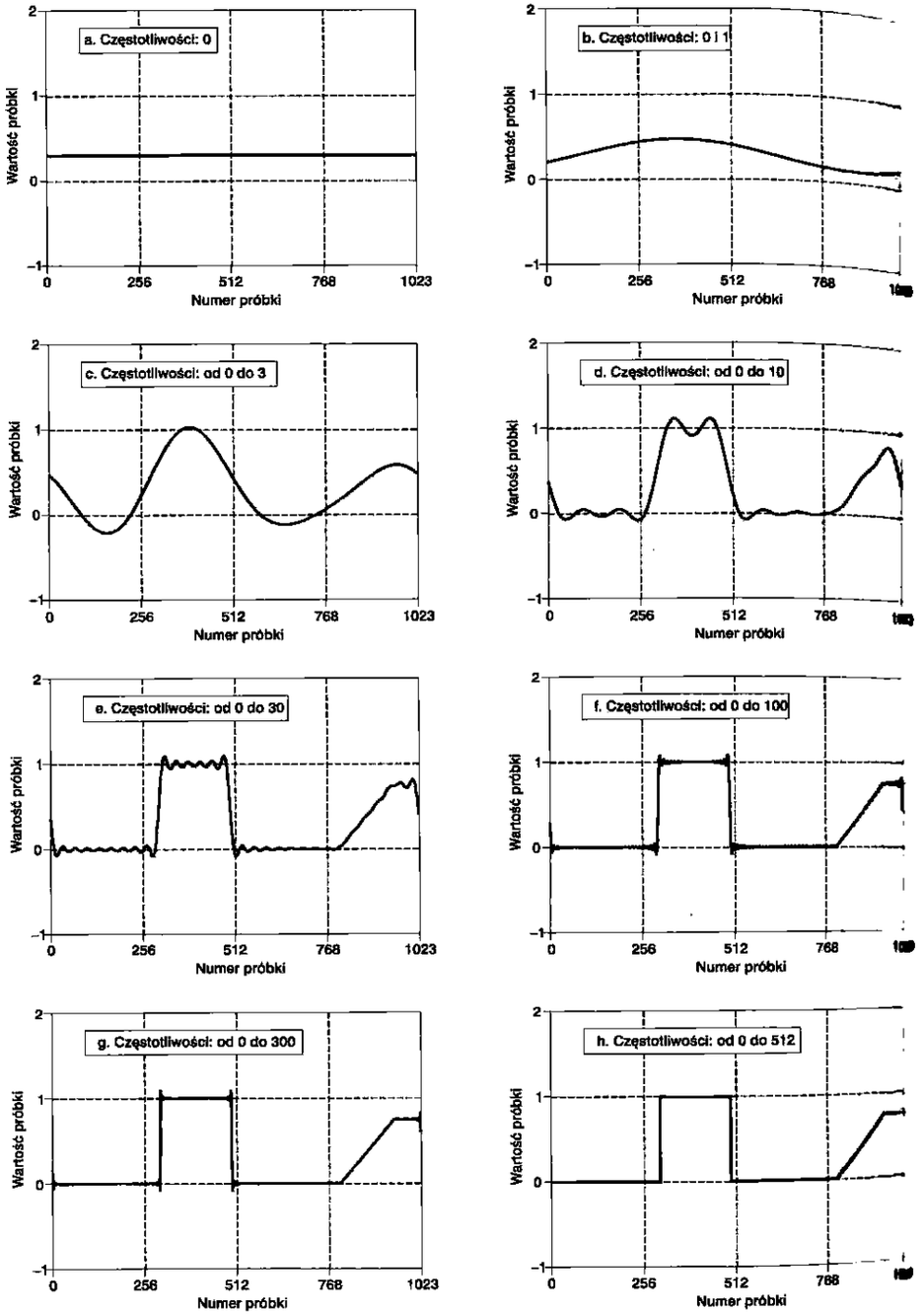
Na rys. 11.5 g) przedstawiono tak zwany **pakiet gaussowski**. Jest on utworzony przez pomnożenie fali sinusoidalnej przez krzywą Gaussa. Na przykład przebieg na rys. 11.5 g) jest falą sinusoidalną pomnożoną przez krzywą Gaussa taką, jak pokazana na rys. 11.5 e). Takiemu przebiegowi odpowiada w dziedzinie częstotliwości przebieg gaussowski o środku symetrii innym niż częstotliwość zerowa. Tak jak poprzednio, ważna jest nie tyle ta para transformat, co *sposób* jej uzasadnienia. Ponieważ sygnał w dziedzinie czasu jest w tym przypadku iloczynem dwóch przebiegów, więc przebieg w dziedzinie częstotliwości będzie splotem dwóch widm częstotliwościowych. Widmo częstotliwościowe fali sinusoidalnej jest funkcją delta o środku symetrii w punkcie częstotliwości fali sinusoidalnej. Widmo częstotliwościowe przebiegu gaussowskiego jest też przebiegiem gaussowskim o środku symetrii przy częstotliwości zerowej. Splot tych dwóch widm daje przebieg gaussowski o środku symetrii przy częstotliwości fali sinusoidalnej. Ten rezultat wydaje się nam już znany; jest to procedura identyczna z *modulacją amplitudy* opisaną w poprzednim rozdziale.

Efekt Gibbsa

Na rys. 11.6 przedstawiono sygnał w dziedzinie czasu syntetyzowany z sinusoid. Odtwarzany sygnał pokazano na ostatnim rysunku (rys. 11.6 h). Ponieważ ten sygnał ma długość 1024 punkty, więc do pełnego odtworzenia jest wymagane 513 indywidualnych częstotliwości. Na rysunkach 11.6 a) ÷ g) pokazano, jak wygląda sygnał odtworzony przy użyciu nie wszystkich wymaganych częstotliwości. Na przykład na rys. 11.5 f) zobrazowano sygnał odtworzony przy zastosowaniu częstotliwości od 0 do 100. Ten sygnał utworzono, wykonując przekształcenie DFT sygnału z rys. 11.6 h), ustawiając sygnały o częstotliwościach od 101 do 512 na wartość zerową i wykonując następnie odwrotne przekształcenie DFT w celu znalezienia wynikowego sygnału w dziedzinie czasu.

W miarę dodawania do syntezy coraz to większej liczby częstotliwości, odtworzony sygnał staje się coraz bardziej zbliżony do ostatecznego wyniku. Interesującą sprawą jest obserwacja, *jak* zmieniają się *zbocza* sygnału w miarę dochodzenia do końcowego wyniku. Na rys. 11.6 h) są trzy ostre zbocza, z których dwa są zboczami impulsu prostokątnego. Trzecie znajduje się między próbkami numer 1023 i 0, gdyż w przekształceniu DFT sygnały w dziedzinie czasu są traktowane jako okresowe. Gdy do odtwarzania używa się nie wszystkich wymaganych częstotliwości, na każdym zboczu pojawia się *przerzut* i tzw. *dzwonienie* (oscylacje zanikające). Jest to tzw. **efekt Gibbsa**, od nazwiska matematyka i fizyka Josiaha Gibbsa, który wyjaśnił to zjawisko w roku 1899.

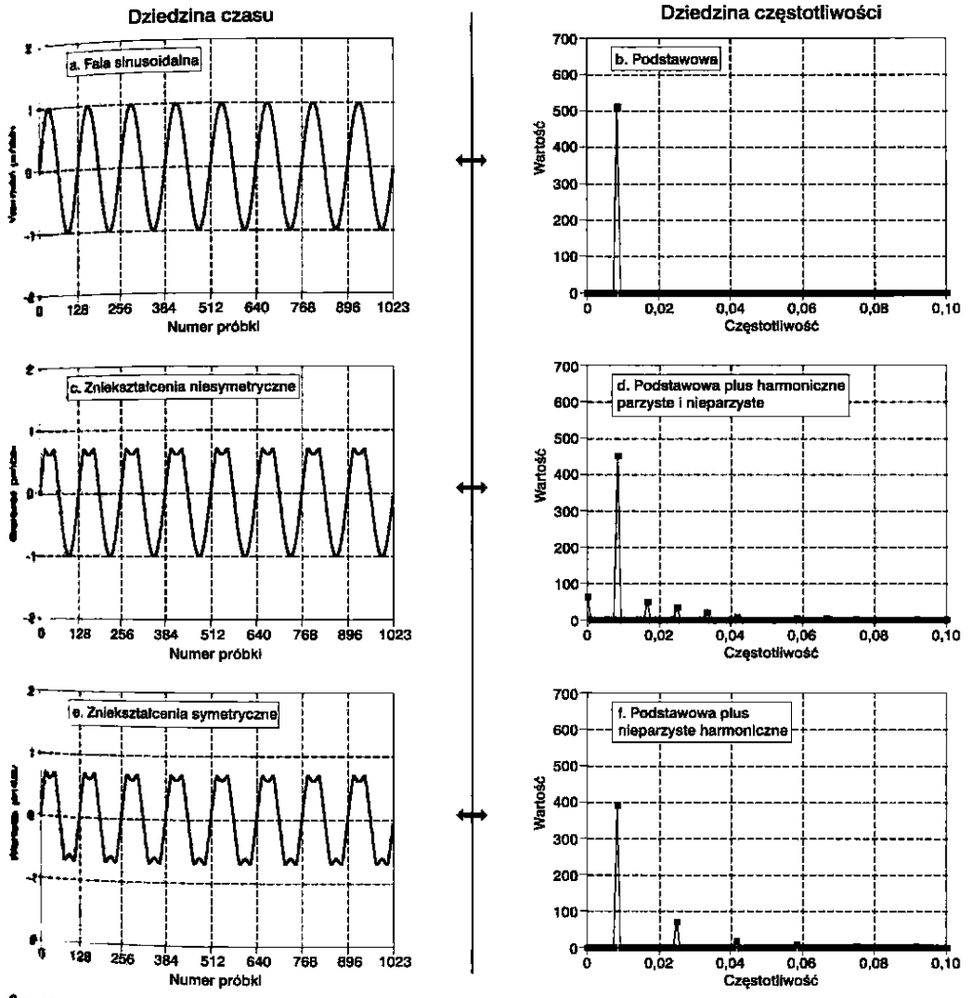
Przyjrzyjmy się dokładniej przerzutowi na rysunkach 11.6 e), f), g). W miarę dodawania większej liczby sinusoid przerzut staje się *węższy*, jednak amplituda przerzutu pozostaje w przybliżeniu taka sama i wynosi ok. 9% amplitudy impulsu. Nie ma kłopotu z sygnałami dyskretnymi; przerzut zostaje wyeliminowany wraz z dodaniem ostatniej wymaganej częstotliwości. Nie da się jednak tak łatwo wyjaśnić odtwarzania sygnału ciągłego. Do syntezy sygnału ciągłego trzeba by zastosować nieskończoną liczbę sinusoid. Problem w tym, że amplituda przerzutu nie zmniejsza się, gdy liczba sinusoid dąży do nieskończoności. Przerzut ciągle pozostaje ok. 9%. Uwzględniając tę sytuację (i inne argumenty), można się zastanawiać, czy *jest możliwe* odtworzenie zbocza przez sumowanie ciągłych sinusoid. Wspomnijmy tu spór między Lagrangem i Fourierem.



Rys. 11.6. Efekt Gibbsa

W rozwiązaniu tej zagadki podstawowym czynnikiem jest fakt, że *szerokość* przerzutu się zmniejsza w miarę dołączania większej liczby sinusoid. W przypadku nieskończonej liczby sinusoid przerzut nadal istnieje, lecz jego szerokość jest równa *zeru*. Dokładnie w miejscu nieciągłości wartość odtworzonego sygnału jest zbieżna do połowy wartości skoku. Jak wykazał Gibbs, suma sinusoid jest zbieżna do sygnału odtwarzanego w tym sensie, że *błąd* między nimi ma energię zerową.

W DSP często można się natknąć na problemy związane z efektem Gibbsa. Na przykład filtr dolnoprzepustowy *obcina* większe częstotliwości, co powoduje przerzut i dzwonienie na zboczach sygnału w *dziedzinie czasu*. Inną często spotykaną procedurą jest obcinanie krańców sygnału w dziedzinie czasu, aby zapobiec ich rozciąganiu się na sąsiednie okresy. To powoduje, w wyniku dualności, zniekształcenie zboczy w *dziedzinie częstotliwości*. Do tych kwestii powrócimy w dalszych rozdziałach przy omawianiu projektowania filtrów.



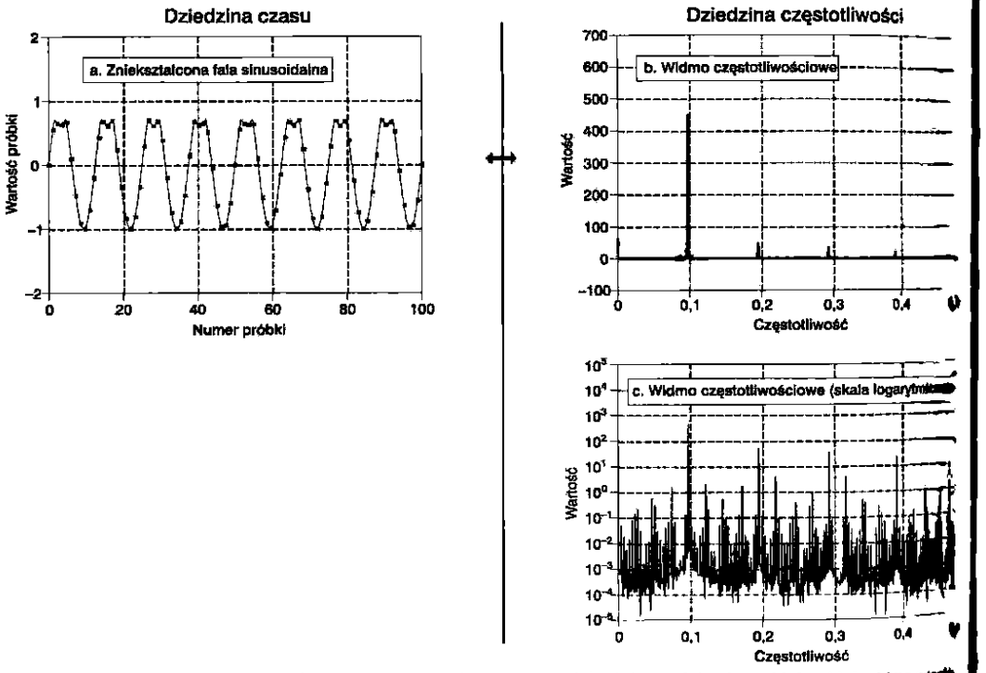
Rys. 11.7. Przykład harmonicznych. Zniekształcenie niesymetryczne (rys. c) daje harmoniczne parzyste i nieparzyste (rys. d), symetryczne zaś (rys. e) – tylko harmoniczne nieparzyste (rys. f)

Harmoniczne

W sygnale okresowym o częstotliwości *f* jedynymi częstotliwościami sinusoid tworzącymi sygnał są całkowite wielokrotności *f*, tj. *f*, 2*f*, 3*f*, 4*f* itd. Te częstotliwości nazywamy **harmonicznymi**. **Pierwszą harmoniczną** jest *f*, **drugą** 2*f*, **trzecią** 3*f* i tak dalej. Pierwsza harmoniczna ma też specjalną nazwę, jest to **częstotliwość podstawowa**. Przykłady podano na rys. 11.7. W rys. 11.7 a) przedstawiono czystą falę sinusoidalną, a na rys. 11.7 b) – jej transformatę DFT, która jest pojedynczym pikiem. Na rys. 11.7 c) fala sinusoidalna ma zniekształcone wierzchołki, a na rys. 11.7 d) pokazano, jak takie zniekształcenie wpływa na reprezentację sygnału w dziedzinie częstotliwości. Ponieważ przebieg zniekształcony jest okresowy i ma częstotliwość taką samą, jak oryginalna fala sinusoidalna, więc przebieg w dziedzinie częstotliwości zawiera piki przebiegu oryginalnego oraz harmoniczne. Harmoniczne mogą mieć różne amplitudy, na ogół zmniejszające się w miarę wzrostu częstotliwości. Tak jak dla każdego sygnału, *ostre zbocza* są reprezentowane przez *większe częstotliwości*. Rozważmy na przykład zwykłą bramkę logiczną TTL generującą falę prostokątną 1 kHz. Czas narastania zboczy jest równy kilka nanosekund, co daje harmoniczne do prawie 100 MHz – *dziesięciotysięczna harmoniczna*!

Na rys. 11.7 e) zilustrowano subtelności analizy harmonicznych. Jeśli sygnał jest symetryczny wokół osi poziomej, a więc górne garbki są odbiciem lustrzanym dolnych, to wszystkie parzyste harmoniczne mają wartość zero. Sygnał zawiera wtedy harmoniczną podstawową, trzecią, piątą itd. (rys. 11.7 f).

Wszystkie okresowe sygnały *ciągłe* mogą być reprezentowane przez sumowanie harmonicznych, tak jak to opisano. W przypadku *dyskretnych* sygnałów okresowych następuje złamanie tej prostej zależności. Jak się zapewne domyślacie, problemem jest tu *aliasing*. Na rys. 11.8 a) przedstawio-



Rys. 11.8. Aliasing harmonicznych. Na rysunkach a) i b) pokazano zniekształconą falę sinusoidalną i jej widmo częstotliwościowe. Składowe harmoniczne o częstotliwości większej niż 0,5 ulegną aliasingowi z częstotliwościami od 0 do 0,5. Na rys. c) to samo widmo częstotliwościowe wykreślono w skali logarytmicznej, uwypuklając wiele pików o małej amplitudzie ulegających aliasingowi

no falę sinusoidalną zniekształconą tak samo, jak poprzednio, tzn. na wierzchołkach. Ten przebieg wygląda na znacznie mniej regularny i gładki niż w poprzednim przykładzie, gdyż sinusoida ma znacznie większą częstotliwość, co powoduje, że liczba próbek na okres jest mniejsza. Widmo częstotliwościowe tego sygnału pokazano na rys. 11.8 b). Zgodnie z oczekiwaniem, można w nim wyróżnić częstotliwość podstawową i harmoniczne. Z tego przykładu wynika, że harmoniczne mogą wychodzić poza wartość 0,5 częstotliwości próbkowania i ulegać *aliasingowi* z częstotliwościami gdzieś w obszarze między 0 i 0,5. Nie można tego dostrzec na rys. 11.8 b), ponieważ amplitudy tych składowych są zbyt małe. Na rys. 11.8 c) przedstawiono widmo częstotliwościowe wykreślone w skali logarytmicznej w celu uwypuklenia tych małych amplitud pików ulegających aliasingowi. Na pierwszy rzut oka widmo wygląda jak szum losowy. Ale to nie jest taki szum, lecz wynik wielu nakładających się częstotliwości harmonicznych, które ulegają aliasingowi.

Trzeba zrozumieć, że ten przykład dotyczy zniekształcenia sygnału *po* utworzeniu jego cyfrowej reprezentacji. Gdyby takie zniekształcenie nastąpiło w sygnale analogowym, to można by usunąć nakładające się częstotliwości filtrem antyaliasingowym *przed* digitalizacją. Aliasing harmonicznych jest problemem tylko wówczas, gdy operacje nieliniowe są wykonywane bezpośrednio na sygnale dyskretnym. Nawet wtedy amplituda tych harmonicznych ulegających aliasingowi jest często wystarczająco mała, aby mogły być pominięte.

Pojęcie harmonicznych jest często użyteczne jeszcze z innego powodu. Wyjaśnia ono, dlaczego w przekształceniu DFT przebiegi w dziedzinie czasu i częstotliwości są widziane jako *okresowe*. W dziedzinie częstotliwości *N*-punktowe przekształcenie DFT zawiera *N*/2 + 1 częstotliwości rozmieszczonych w jednakowych odstępach. Częstotliwości *między* tymi próbkami można traktować jako mające wartość zerową lub jako nieistniejące. W obu tych przypadkach nie dają one swego udziału do syntezy sygnału w dziedzinie czasu. Innymi słowy, *dyskretne* widmo częstotliwościowe zawiera *harmoniczne*, a nie ciągły zakres częstotliwości. To powoduje, że sygnał w dziedzinie czasu powinien być *okresowy*, o częstotliwości równej najmniejszej częstotliwości sinusoid w dziedzinie częstotliwości, tzn. o częstotliwości podstawowej. Przy pominięciu składowej stałej, przebieg o najmniejszej częstotliwości reprezentowanej w dziedzinie częstotliwości wykonuje pełny okres co *N* próbek, co w dziedzinie czasu skutkuje przebiegiem okresowym o okresie *N*. Tak więc jeśli przebieg w jednej dziedzinie jest *dyskretny*, to w drugiej musi być *okresowy* i odwrotnie. To dotyczy wszystkich czterech przekształceń z rodziny fourierowskiej. Ponieważ w DFT przebiegi w obu dziedzinach są widziane jako dyskretne, więc też w obu dziedzinach muszą być traktowane jako okresowe. Próbki w każdej dziedzinie reprezentują harmoniczne o okresowości takiej, jak przebiegu w drugiej dziedzinie.

Sygnały świergotowe

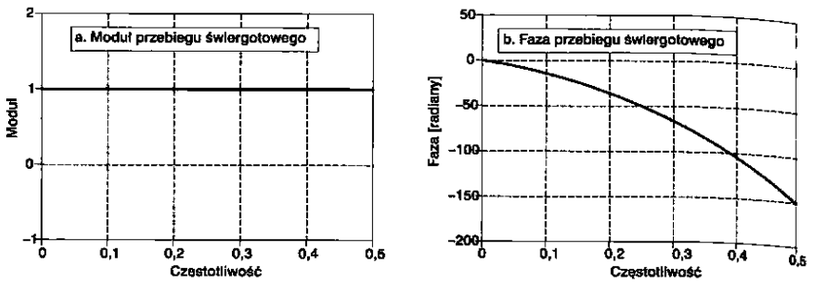
Sygnały świergotowe (*chirp signals*) są pomysłowym sposobem rozwiązania problemu spotykanego w praktyce, w systemach lokacyjnych, takich jak radary i sonary. Na rys. 11.9 przedstawiono odpowiedź częstotliwościową systemu świergotowego. Moduł ma wartość stałą równą jedności, podczas gdy faza ma charakterystykę paraboliczną:

$$Phase X[k] = \alpha k + \beta k^2$$

Wzór 11.5. Faza systemu świergotowego

Parametr α wprowadza do charakterystyki fazowej liniowe nachylenie, tzn. że po prostu powoduje przesunięcie odpowiedzi impulsowej w lewo lub w prawo, według życzenia. *Krzywizna* charakterystyki fazowej zależy od parametru β. Te dwa parametry trzeba wybrać w taki sposób, żeby faza przy częstotliwości 0,5 (tj. dla $k = N/2$) była wielokrotnością 2π. Pamiętajmy, że przy wszystkich bezpośrednich operacjach związanych z fazą, powinna ona przy częstotliwościach 0 i 0,5 być równa zeru (lub być wielokrotnością 2π, co na jedno wychodzi).

Na rys. 11.10 pokazano impuls wchodzący do systemu świergotowego oraz odpowiedź impulsową pobudzającą ten system. Odpowiedź impulsowa jest wiązką oscylacyjną, która zaczyna się od małych częstotliwości, i wzrastających do dużych w miarę wzrostu czasu. Taki przebieg nazywamy sygnałem *świergotowym* z bardzo prostej przyczyny – odtwarzany w głośniku brzmi jak świergot ptaków.



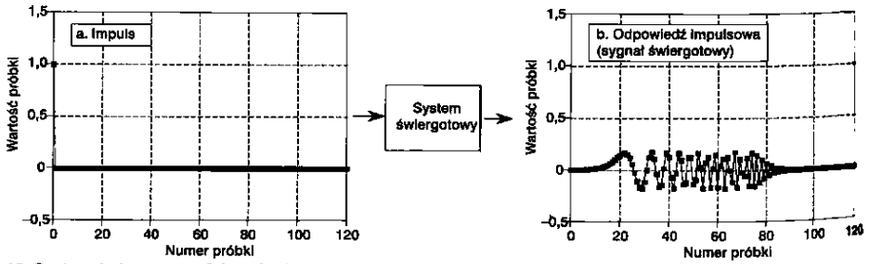
Rys. 11.9. Odpowiedź częstotliwościowa systemu świergotowego. Moduł jest stały, a charakterystyka fazowa jest paraboliczna

Główną cechą systemu świergotowego jest jego pełna *odwracalność*. Po przejściu sygnału świergotowego przez system *antyświergotowy*, sygnał znowu zamienia się w impuls. System antyświergotowy powinien więc mieć moduł równy jedności, a charakterystykę fazową *odwrotną* w stosunku do charakterystyki systemu świergotowego. To oznacza – zgodnie z rozważaniami w poprzednim rozdziale – że odpowiedź impulsowa systemu antyświergotowego jest określana przez lewo-prawostronne odwrócenie przebiegu odpowiedzi impulsowej systemu świergotowego. To interesujące, ale do czego może służyć?

Zastanówmy się, jak działa system radarowy. Krótka wiązka energii w.cz. jest wysyłana z anteny kierunkowej. Samolot lub inne obiekty odbijają część tej energii z powrotem do odbiornika umieszczonego blisko nadajnika. Ponieważ fale radiowe rozchodzą się ze stałą prędkością, więc czas upływający między sygnałami nadanym i odebranym jest miarą odległości do przeszkody. To narzuca na impuls pierwsze wymaganie – powinien on być możliwie jak najkrótszy. Na przykład impuls o czasie trwania 1 μs daje wiązkę radiową o długości ok. 300 m. Stąd wynika, że uzyskana w systemie informacja o odległości będzie miała w przybliżeniu taką właśnie rozdzielczość. Chcąc uzyskać lepszą rozdzielczość wyznaczania odległości, należy zastosować krótszy impuls.

Drugi wymóg jest oczywisty. Jeśli chcemy wykrywać obiekty znajdujące się w większej odległości, to konieczny jest impuls o większej energii. Niestety, *większa energia* i *krótszy impuls* to wymagania przeciwstawne. Moc elektryczna konieczna do wygenerowania impulsu jest równa energii impulsu podzielonej przez jego czas trwania. Jednoczesne wymaganie *zwiększonej energii* i *krótszego czasu trwania* powoduje, że wymagana moc elektryczna staje się poważnym ograniczeniem w systemie. Stopień wyjściowy nadajnika może dysponować tylko taką mocą, aby sam nie uległ zniszczeniu.

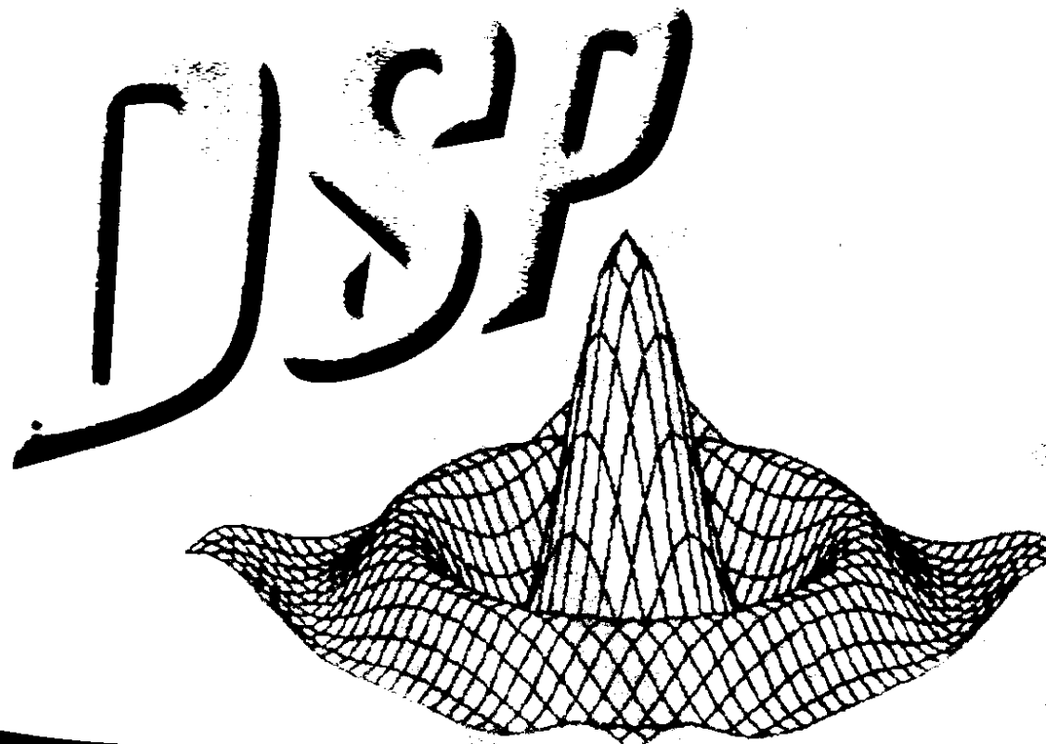
Tego ograniczenia można uniknąć, stosując sygnały świergotowe. Impuls, przed dojściem do stopnia końcowego nadajnika radarowego, jest przepuszczany przez system świergotowy. Zamiast impulsu, ulegającego odbiciu od wykrytego samolotu, stosuje się sygnał świergotowy. Po odebraniu echa świergotowego, sygnał jest przepuszczany przez system antyświergotowy, który przetwarza go znowu w impuls. Dzięki temu części systemu, które mierzą odległość, odbierają krótsze impulsy, a układy wymagające większej mocy mają do czynienia tylko z sygnałami o dłuższym czasie trwania. Takie kształtowanie przebiegów jest podstawową częścią nowoczesnych systemów radarowych.



Rys. 11.10. System świergotowy. Odpowiedź impulsowa systemu świergotowego jest sygnałem świergotowym

12

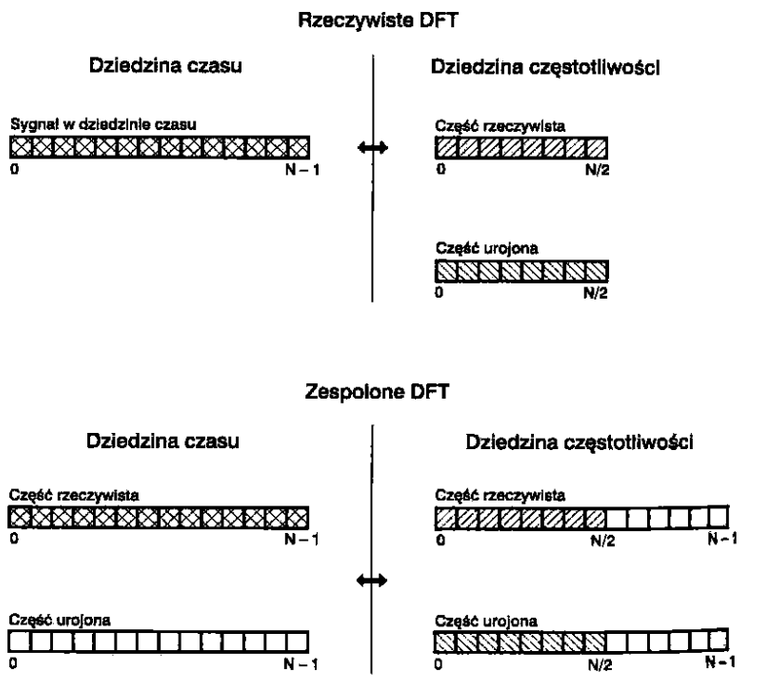
Szybkie przekształcenie Fouriera



Jest kilka metod obliczania dyskretnej transformaty Fouriera (DFT), takich jak rozwią- nie równoważnych równań liniowych lub metoda *korelacyjna* opisana w rozdziale 8. Szy- przekształcenie Fouriera (FFT – *Fast Fourier Transform*) jest inną metodą obliczania DFT Daje ona taki sam wynik, jak inne sposoby obliczania, lecz jest niewiarygodnie t efektywna, redukując czas obliczeń często o *setki* razy. Jest to taka różnica, jak między odrzutowcem a spacerem na piechotę! Wiele metod opisanych w tej książce nie mo być stosowanych w praktyce, gdyby nie istniało FFT. Chociaż FFT wymaga tylko kilku sięciu wierszy programu, to jest jednak jednym z najbardziej skomplikowanych algoryt w DSP. Lecz się nie martwcie. Możecie z łatwością korzystać z publikowanych pro programowych bez pełnego rozumienia ich wewnętrznego funkcjonowania.

Rzeczywiste DFT z zastosowaniem zespolonego DFT

Zasługa wprowadzenia FFT do techniki przypada J.W. Cooleyowi i J.W. Tukeyowi za ich p blikację *An algorithm for the machine calculation of complex Fourier Series*, „Mathem Computation", vol. 19, 1965, str. 297–301. Patrząc wstecz, trzeba jednak stwierdzić, że już dawno wynaleźli tę technikę. Na przykład wielki niemiecki matematyk Karl Fried Gauss (1777–1855) zastosował tę metodę prawie o stulecie wcześniej. Ta dawna praca zo ła w dużym stopniu zapomniana, gdyż brakowało wówczas narzędzia do jej wykorzy praktycznego, czyli *komputera*. Cooley i Tukey są tak honorowani, gdyż odkryli FFT w n powiednim czasie, u zarania rewolucji komputerowej.



Rys. 12.1. Porównanie rzeczywistego i zespolonego przekształcenia DFT. W rzeczywistym przekształ DFT z N-punktowego sygnału w dziedzinie czasu są tworzone dwa (N/2 + 1)-punktowe sygnały w dzied nie częstotliwości. W zespolonym DFT z dwóch N-punktowych sygnałów w dziedzinie czasu są tworzo dwa N-punktowe sygnały w dziedzinie częstotliwości. Obszary zakreskowane wskazują na wartości wsp ne dla obu przekształceń

Podstawą FFT jest *zespolone* przekształcenie DFT, bardziej skomplikowana wersja *rzeczywistego* DFT, które omawialiśmy w czterech poprzednich rozdziałach. Nazwy tych przekształceń wynikają ze sposobu reprezentacji danych – w postaci liczb rzeczywistych lub zespolonych. Określenie *zespolony* nie znaczy, że takie wyrażanie liczb jest trudne lub skomplikowane, lecz że wiąże się z zastosowanymi metodami matematycznymi. Matematyka liczb zespolonych często *jest* trudna i skomplikowana, lecz nie stąd pochodzi jej nazwa. Zespolone DFT jest omówione w rozdziale 29 i tam też są zawarte podstawowe informacje konieczne do zrozumienia szczegółów algorytmu FFT. Tematyka tego rozdziału jest prostsza. Będzie mowa o tym, jak zastosować FFT do obliczania rzeczywistego DFT, bez zagłębiania się w grząskie obszary matematyki wyższej.

Ponieważ FFT jest algorytmem do obliczania zespolonego DFT, więc ważną sprawą jest zrozumienie, jak transferować dane z *rzeczywistego DFT* do formatu *zespolonego DFT* i z tego formatu. Na rys. 12.1 porównano sposoby gromadzenia danych w rzeczywistym i zespolonym DFT. W rzeczywistym DFT N-punktowy sygnał w dziedzinie czasu jest przekształcany na dwa ($N/2 + 1$)-punktowe sygnały w dziedzinie częstotliwości. Sygnał w dziedzinie czasu jest nazywany po prostu *sygnałem w dziedzinie czasu*. Dwa sygnały w dziedzinie częstotliwości są nazywane *częścią rzeczywistą* i *częścią urojoną*, odnoszącymi się, odpowiednio, do wartości fali kosinusoidalnej i sinusoidalnej. To już wiadomo z poprzednich rozdziałów.

W zespolonym DFT następuje przekształcenie dwóch N-punktowych sygnałów w dziedzinie czasu na dwa N-punktowe sygnały w dziedzinie częstotliwości. Dwa sygnały w dziedzinie czasu są nazywane *częścią rzeczywistą* i *częścią urojoną*, a więc tak samo jak sygnały w dziedzinie częstotliwości. Mimo swych nazw, wszystkie wartości w tych matrycach są po prostu zwyczajnymi liczbami. (Jeśli znacie liczby zespolone, to warto powiedzieć, że j nie występuje w wartościach zawartych w tablicach; jest stosowne w operacjach *matematycznych*. Pamiętajcie, że symbol *Im*( ) oznacza wartość rzeczywistą).

Przyjmijmy, że mamy sygnał N-punktowy i chcemy zrealizować *rzeczywiste przekształcenie DFT*, stosując zespolone *przekształcenie* DFT (np. stosując algorytm FFT). Najpierw przesuńmy sygnał N-punktowy do części rzeczywistej zespolonego DFT w dziedzinie czasu, a następnie wszystkie próbki w części urojonej ustawmy na wartość *zero*. Obliczenie zespolonej transformaty DFT daje w rezultacie w dziedzinie częstotliwości sygnał rzeczywisty i sygnał urojony, a każdy z nich składa się z N punktów. Próbki od 0 do $N/2$ w tych sygnałach odpowiadają widmu rzeczywistego DFT.

Zgodnie z rozważaniami w rozdziale 10, widmo częstotliwościowe DFT jest okresowe, gdy są uwzględnione częstotliwości ujemne (patrz rys. 10.9). Wybór pojedynczego okresu jest dowolny; można wybrać okres między –1,0 i 0, –0,5 i 0,5, 0 i 1,0 lub każdy inny przedział o szerokości jednostkowej w odniesieniu do częstotliwości próbkowania. Zespolone widmo DFT obejmuje częstotliwości ujemne przy wyborze okresu od 0 do 1,0. Inaczej mówiąc, jeden pełny okres rozciąga się od próbki 0 do próbki $N – 1$, co odpowiada wielokrotności 0 lub 1 częstotliwości próbkowania. Częstotliwości dodatnie są między próbkami 0 i $N/2$, co odpowiada 0 i 0,5. Inne próbki, pomiędzy $N/2 + 1$ i $N – 1$, zawierają wartości dla częstotliwości ujemnych, które są na ogół pomijane.

Nieco trudniejsze jest wykonywanie *rzeczywistego odwrotnego przekształcenia DFT* przy użyciu *zespolonego odwrotnego przekształcenia DFT*. Przyczyną jest konieczność upewnienia się, czy częstotliwości ujemne wprowadzono we właściwym formacie. Pamiętajmy, że punkty od 0 do $N/2$ w zespolonym DFT są takie same, jak w rzeczywistym DFT, zarówno jeśli chodzi o część rzeczywistą, jak i urojoną. W części rzeczywistej punkt $N/2 + 1$ jest taki sam, jak punkt $N/2 – 1$, punkt $N/2 + 2$ taki sam jak punkt $N/2 – 2$ itd., aż do punktu $N – 1$, który jest taki sam, jak punkt 1. Podobna zasada obowiązuje dla części urojonej, z tym tylko, że następuje zmiana znaku. To znaczy, że punkt $N/2 + 1$ ma ujemną wartość punktu $N/2 – 1$, punkt $N/2 + 2$ – ujemną wartość punktu $N/2 – 2$ itd. Zwróćmy uwagę, że próbki 0 i $N/2$ nie mają swych równoważnych punktów w tym schemacie powtórzeń. Skorzystajcie z rys. 10.9 jako z przewodnika ułatwia-

Tab. 12.1.

```
6000 'GENERACJA CZĘSTOTLIWOŚCI UJEMNEJ
6010 'Ten podprogram tworzy sygnał w zespolonej dziedzinie częstotliwości z sygnału
6015 'w rzeczywistej dziedzinie częstotliwości.
6020 'Po wejściu do tego podprogramu N% zawiera liczbę punktów w sygnałach a
6030 'REX[ ] i IMX[ ] zawierają rzeczywiste widmo częstotliwościowe w próbkach od 0 do N%/2.
6040 'Po powrocie REX[ ] i IMX[ ] zawierają zespolone widmo częstotliwościowe w próbkach od 0 do N%-1
6060 FOR K% = (N%/2+1) TO (N%-1)
6070   REX[K%] = REX[N%-K%]
6080   IMX[K%] = -IMX[N%-K%]
6090 NEXT K%
6100 '
6110 RETURN
```

jącego zrozumienie tej symetrii. W praktyce wprowadza się widmo częstotliwościowe rzeczywistego DFT do próbek od 0 do $N/2$ w matrycach zespolonego DFT i następnie korzysta z podprogramu do wytworzenia częstotliwości ujemnych w próbkach od $N/2 + 1$ do $N - 1$. Ten program zamieszczono w tablicy 12.1. W celu sprawdzenia, czy istnieje właściwa symetria należy – po wykonaniu odwrotnego przekształcenia FFT – przyjrzeć się części urojonej w dziedzinie czasu. Powinna ona zawierać same zera, jeśli wszystko jest w porządku (poza szumem o wartości kilku części na milion, przy użyciu obliczeń o pojedynczej dokładności).

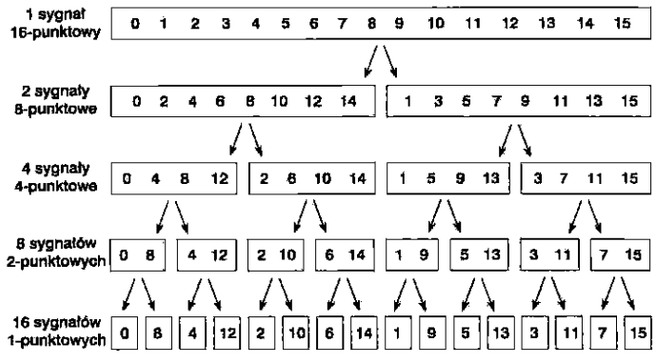
Jak działa FFT

FFT jest skomplikowanym algorytmem i poznanie jego szczegółów pozostawia się zwykle tym, którzy specjalizują się w tej problematyce. W tym podrozdziale będzie opisane ogólne działanie FFT, jednak z ominięciem jednego istotnego zagadnienia – zastosowania *liczb zespolonych*. Jeśli jednak macie odpowiednie przygotowanie z matematyki liczb zespolonych, to możecie między wierszami wyczytać prawdziwą naturę tego algorytmu. Nie martwcie się, jeśli nie wychwycicie niektórych szczegółów. Bardzo niewielu naukowców i inżynierów potrafi od ręki napisać program FFT.

W zapisie zespolonym każda z dziedzin – tak czasu, jak i częstotliwości – zawiera jedynie sygnał złożony z *N punktów zespolonych*. Każdy z tych punktów zespolonych zawiera dwie liczby, wyrażające część rzeczywistą i część urojoną. Na przykład, gdy mówimy o zespolonej próbce *X*[42], odnosi się to do kombinacji *Re X*[42] i *Im X*[42]. A zatem każda zmienna zespolona zawiera dwie liczby. W przypadku mnożenia dwóch zmiennych zespolonych trzeba wykonać kombinację czterech oddzielnych składników w celu utworzenia dwóch czynników iloczynu (takich jak we wzorze 9.1). W poniższych rozważaniach na temat „Jak działa FFT" zastosowano specyficzne słownictwo związane z zapisem zespolonym. W tym języku pojęcia *sygnał*, *punkt*, *próbka* i *wartość* odnoszą się do *kombinacji* części rzeczywistej i urojonej.

FFT działa na zasadzie rozłożenia *N*-punktowego sygnału w dziedzinie czasu na *N* sygnałów też w dziedzinie czasu, z których każdy zawiera jeden punkt. Następnym krokiem jest obliczenie *N* widm częstotliwościowych odpowiadających tym *N* sygnałom w dziedzinie czasu. Wreszcie, *N* widm podlega syntezie, tworząc jedno widmo częstotliwościowe.

Na rys. 12.2 przedstawiono przykład rozkładu w dziedzinie czasu z zastosowaniem FFT. W tym przykładzie 16-punktowy sygnał jest rozkładany w czterech etapach. W pierwszym etapie 16-punktowy sygnał jest rozdzielany na dwa sygnały 8-punktowe. W drugim etapie dane są rozkładane na cztery sygnały zawierające po 4 punkty. Takie rozkładanie jest kontynuowane aż do uzyskania *N* sygnałów zawierających po jednym punkcie. Za każdym razem, gdy sygnał jest dzielony na dwa, stosuje się **rozkład z przeplotem**, a więc sygnał jest rozdzielany na próbki o numerach parzystych i nieparzystych. Najlepszym sposobem



Rys. 12.2. Rozkład FFT. Sygnał N-punktowy jest rozkładany na N sygnałów, z których każdy zawiera pojedynczy punkt. W każdym etapie zastosowano rozkład z przeplotem, rozdzielając próbki o numerach parzystych i nieparzystych.

zrozumienia tej procedury jest dokładne przeanalizowanie rys. 12.2. W tym procesie rozkładania jest konieczne log_2N etapów, tzn. sygnał 16-punktowy (2^4) wymaga 4 etapów, sygnał 512-punktowy (2^7) – 7 etapów, sygnał 4096-punktowy (2^{12}) – 12 etapów itd. Zapamiętajmy to wyrażenie: log_2N; będziemy się do niego wielokrotnie odwoływać w tym rozdziale.

Teraz, kiedy już zrozumieliście procedurę rozkładania sygnału, można ją znacznie uprościć. Rozkład jest niczym innym, jak *zmianą uporządkowania* próbek w sygnale. Na rys. 12.3 pokazano wymagany schemat takiej zmiany. W lewej kolumnie zamieszczono listę numerów próbek sygnału oryginalnego, wraz z ich binarnymi równoważnikami. W prawej kolumnie podano listę numerów próbek po zmianie uporządkowania, także z binarnymi równoważnikami numerów. Ważną zasadą jest to, że liczby binarne przed i po zmianie uporządkowania są względem siebie *odwrócone*. Na przykład próbka 3 (0011) została zamieniona miejscami z próbką numer 12 (1100). Podobnie próbkę numer 14 (1110) zamieniono miejscami z próbką numer 7 (0111) i tak dalej. Rozkład FFT w dziedzinie czasu jest zwykle wykonywany przez algorytm **sortowania z odwróceniem bitowym**. Obejmuje on zmianę uporządkowania *N* próbek w dziedzinie czasu przez zliczanie binarne ze zmianą kolejności bitów (bit najmniej znaczący z lewej strony), jak to przedstawiono w prawej kolumnie na rys. 12.3.

Następnym krokiem w realizacji algorytmu jest wyznaczenie widm częstotliwościowych 1-punktowych sygnałów w dziedzinie czasu. Nic łatwiejszego, gdyż widmo częstotliwościowe sygnału 1-punktowego jest równe *temu sygnałowi*. Wynika stąd, że ten etap nie wymaga *żadnych* operacji. Chociaż nie trzeba w tym etapie wykonać żadnej pracy, to nie zapomnijmy, że każdy z 1-punktowych sygnałów jest teraz widmem częstotliwościowym, a nie sygnałem w dziedzinie czasu.

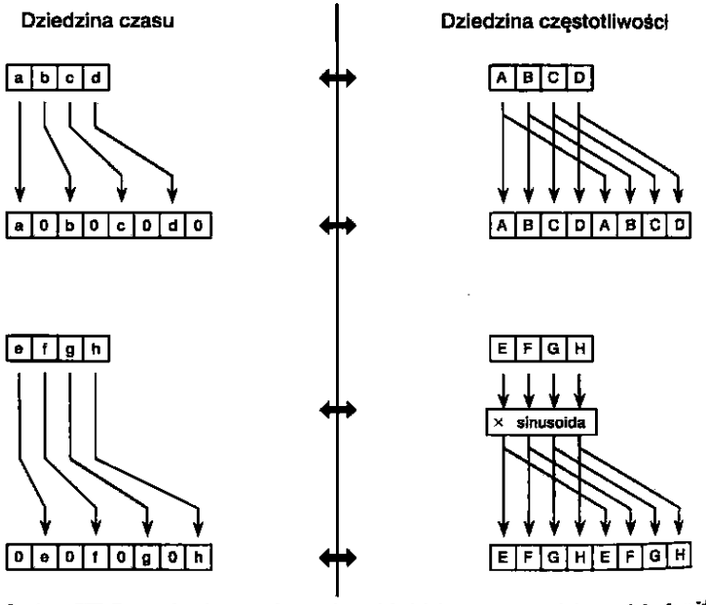
Ostatnim etapem w FFT jest połączenie ze sobą *N* widm częstotliwościowych w kolejności odwrotnej do tej, jaka miała miejsce przy rozkładzie w dziedzinie czasu. I tu w algorytmie powstaje pewien nieporządek. Niestety nie da się zastosować skróconego odwracania bitów, trzeba cofać się wstecz etap po etapie. W pierwszym etapie 16 widm częstotliwościowych 1-punktowych jest syntetyzowanych w 8 widm częstotliwościowych (każde po 2 punkty). W etapie drugim te 8 widm podlega syntezie w 4 widma częstotliwościowe 4-punktowe itd. Po ostatnim etapie uzyskuje się wynik FFT – 16-punktowe widmo częstotliwościowe.

Na rys. 12.4 pokazano, jak dwa 4-punktowe widma częstotliwościowe są ze sobą łączone w pojedyncze widmo o 8 punktach. Ta synteza musi *anulować* rozkład z przeplotem dokonany w dziedzinie czasu. Innymi słowy, operacja w dziedzinie częstotliwości powinna odpowiadać przeprowadzonej w dziedzinie czasu procedurze *łączenia* ze sobą dwóch sygnałów 4-punk-

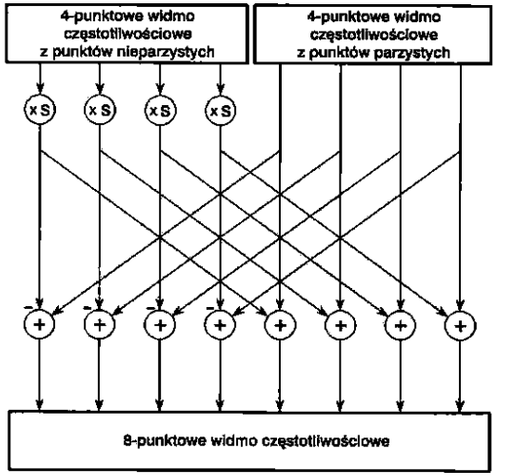
Numery próbek w normalnej kolejności		Numery próbek po odwróceniu bitowym	
Dziesiętne	Binarne	Dziesiętne	Binarne
0	0000	0	0000
1	0001	8	1000
2	0010	4	0100
3	0011	12	1100
4	0100	2	0010
5	0101	10	1010
6	0110	6	0100
7	0111	14	1110
8	1000	1	0001
9	1001	9	1001
10	1010	5	0101
11	1011	13	1101
12	1100	3	0011
13	1101	11	1011
14	1110	7	0111
15	1111	15	1111

Rys. 12.3. Sortowanie z odwróceniem bitowym w FFT. Rozkład FFT w dziedzinie czasu można wykonać przez posortowanie próbek zgodnie z odwróconą kolejnością bitów

towych z użyciem przeplotu. Rozpatrzmy dwa sygnały w dziedzinie czasu, *abcd* oraz *efgh*. Sygnał 8-punktowy w dziedzinie czasu może być utworzony w dwóch krokach: rozrzedzeniu zerami każdego 4-punktowego sygnału, aby uczynić z niego sygnał 8-punktowy i następnie zsumowanie uzyskanych sygnałów. To znaczy, że sygnał *abcd* staje się sygnałem *a0b0c0d0*, a *efgh* staje się sygnałem *0e0f0g0h*. Zsumowanie tych dwóch 8-punktowych sygnałów da w twarza sygnał *aebfcgdh*. Jak wynika z rys. 12.4, rozrzedzanie zerami sygnału w dziedzinie czasu jest *powielaniem* widma częstotliwościowego. Zatem w FFT widma częstotliwościowe są łączone ze sobą przez ich powielanie i następnie sumowanie powielonych widm.



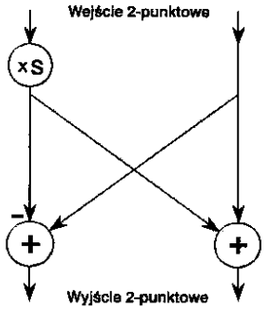
Rys. 12.4. Synteza FFT. Rozrzedzanie zerami sygnału w dziedzinie czasu powoduje powielanie widma w dziedzinie częstotliwości. Jeśli sygnał w dziedzinie czasu jest podczas rozrzedzania także przesunięty o jedną próbkę, to widmo będzie dodatkowo pomnożone przez sinusoidę



Rys. 12.5. Sieć przepływu sygnałów w syntezie FFT. Przedstawiono tu sposób łączenia ze sobą dwóch 4-punktowych widm częstotliwościowych w celu utworzenia pojedynczego, 8-punktowego widma częstotliwościowego. Operacja × S oznacza, że sygnał jest mnożony przez sinusoidę o odpowiednio wybranej częstotliwości

W celu dopasowania do siebie sumowanych widm, rozrzedzanie zerami sygnałów w dziedzinie czasu wykonuje się dwoma nieco różniącymi się sposobami. W jednym sygnale zerami są *punkty nieparzyste*, podczas gdy w drugim są nimi *punkty parzyste*. Tak więc jeden z sygnałów w dziedzinie czasu (*0e0f0g0h* na rys. 12.4) jest przesunięty w prawo o jedną próbkę. Takie przesunięcie w dziedzinie czasu odpowiada pomnożeniu widma przez *sinusoidę*. Aby to zrozumieć, przypomnijcie sobie, że przesunięcie w dziedzinie czasu jest równoważne splotowi sygnału z przesuniętą funkcją delta. Powoduje to mnożenie widma sygnału przez widmo przesuniętej funkcji delta. Widmo przesuniętej funkcji delta jest sinusoidą (patrz rys. 11.2).

Na rys. 12.5 przedstawiono sieć przepływu sygnałów, obrazującą łączenie dwóch widm 4-punktowych w pojedyncze widmo 8-punktowe. Aby jeszcze bardziej uprościć ten problem, zwróćcie uwagę na fakt, że rys. 12.5 jest utworzony przez wielokrotne powtarzanie podstawowej sieci z rys. 12.6.

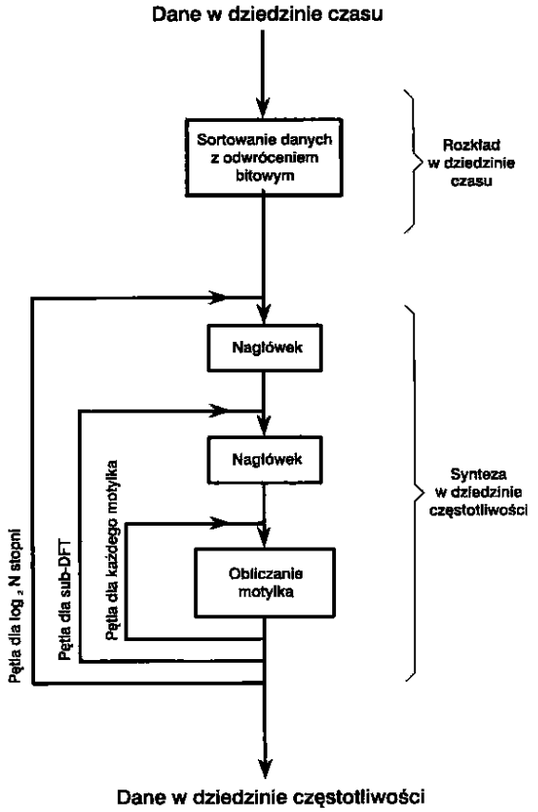


Rys. 12.6. „Motylek" FFT. Jest to podstawowy element obliczeniowy w FFT, który dwa punkty zespolone przetwarza na dwa inne punkty zespolone

Ta prosta sieć przepływu sygnałów jest, ze względu na swój kształt, nazywana motylkiem. Motylek jest podstawowym elementem obliczeniowym w FFT, przekształcającym dwa punkty zespolone w dwa inne punkty zespolone.

Na rys. 12.7 przedstawiono strukturę całego FFT. Rozkład w dziedzinie czasu jest realizowany algorytmem sortowania z odwróceniem bitowym. Przeniesienie poddanych rozkładowi danych w dziedzinę częstotliwości nie wymaga żadnych operacji i dlatego nie jest uwzględnione na rysunku.

Synteza w dziedzinie częstotliwości wymaga trzech pętli. Pętla zewnętrzna przechodzi przez log_2N stopni. Stopniem nazywamy tu każdy poziom na rys. 12.2, poczynając od dołu i przesuwając się ku górze. Środkowa pętla przechodzi przez każde przetwarzane indywidualne widmo częstotliwościowe w danym stopniu (tj. przez każdą z ramek w kolejnych poziomach na rys. 12.2). W pętli najbardziej wewnętrznej zastosowanu motylek do obliczania punktów w każdym widmie częstotliwościowym, a więc jest to pętla przechodząca przez próbki wewnątrz każdej ramki na rys. 12.2. Ramki „nagłówek" (*overhead*) na rys. 12.7 określają indeksy początkowe i końcowe dla pętli, jak również obliczania sinusoid w motylkach. Przejdziemy teraz do sedna tego rozdziału, a więc do programów FFT.



Rys. 12.7. Sieć przepływu sygnałów w FFT. Sieć zawiera trzy etapy: 1) rozkład N-punktowego sygnału w dziedzinie czasu na N sygnałów 1-punktowych, 2) wyznaczenie widma każdego z sygnałów N-punktowych (nie wymaga żadnej operacji) oraz 3) synteza N widm częstotliwościowych w pojedyncze widmo częstotliwościowe

Programy FFT

Rzeczywiste DFT, zgodnie z rozważaniami w rozdziale 8, może być wykonywane przez korelację sygnału w dziedzinie czasu z falami sinusoidalnymi i kosinusoidalnymi (patrz tablica 8.2). W tablicy 12.2 zamieszczono program do obliczania tą samą metodą *zespolonego DFT*. Przez porównanie można stwierdzić, że jest to program, który został ulepszony przez FFT.

W tablicach 12.3 i 12.4 podano dwa różne programy FFT, jeden w FORTRAN-ie, drugi w BASIC-u. Najpierw spójrzmy na podprogram BASIC w tablicy 12.4. Daje on dokładnie taki sam wynik, jak metoda korelacyjna przedstawiona w tablicy 12.2, z tą wszakże różnicą, że jest *znacznie szybszy*. Do rozróżnienia poszczególnych sekcji tego programu można się posłużyć rysunkiem 12.7. Dane są wprowadzane do tego podprogramu FFT w postaci matryc *Re X*[ ] i *Im X*[ ], każda obejmująca próbki od 0 do $N-1$. Po powrocie z tego podprogramu w miejsce tych danych są wpisywane dane dotyczące dziedziny częstotliwości. To jeszcze jeden aspekt daleko posuniętej optymalizacji FFT; te same matryce są stosowane jako pamięć danych wejściowych, pośrednia i danych wyjściowych. Takie wydajne wykorzystanie pamięci jest ważne w projektowaniu szybkich układów do obliczania FFT. Do opisania takiego wykorzystania pamięci jest stosowany angielski termin ***in-place computation***.

Chociaż wszystkie programy FFT dają taki sam wynik liczbowy, to są jednak subtelne różnice w programowaniu, którym trzeba się przyjrzeć. Kilka z tych ważnych różnic zilustrowano w listingu programu w FORTRAN-ie podanego w tablicy 12.3. W tym programie zastosowano algorytm zwany **decymacją w dziedzinie częstotliwości**, podczas gdy poprzednio opisany algorytm jest nazywany **decymacją w dziedzinie czasu**. W algorytmie decymacji w dziedzinie częstotliwości sortowanie z odwróceniem bitowym jest wykonywane *po* trzech zagnieżdżonych pętlach. Są także takie podprogramy FFT, w których całkowicie wyeliminowano sortowanie z odwróceniem bitowym. Żadna z tych odmian nie ulepsza w sposób znaczący działania FFT i nie trzeba się przejmować kwestią wyboru jednej z nich.

Tablica 12.2.

```
'ZESPOLONE DFT PRZEZ KORELACJĘ
'Na wejściu do programu N% zawiera liczbę punktów w DFT, a XR[ ] i XI[ ]
'zawierają części rzeczywistą i urojoną sygnału w dziedzinie czasu.
'Po powrocie REX[ ] i IMX[ ] zawierają dane w dziedzinie częstotliwości.
'Wszystkie sygnały zmieniają się w zakresie od 0 do N%-1.

PI = 3.14159265                 'Ustaw stałe.

FOR K% = 0 TO N%-1              'Zeruj REX[ ] i IMX[ ], aby podczas korelacji mogły
  REX[K%] = 0                   'być wykorzystane jako akumulatory.
  IMX[K%] = 0
NEXT K%

FOR K% = 0 TO N%-1              'Pętla przez wszystkie wartości w dziedzinie częstotliwości.
  FOR I% = 0 TO N%-1            'Korelacja z zespolonymi sinusoidami SR i SI.

    SR = COS(2*PI*K%*I%/N%)     'Oblicz zespoloną sinusoidę.
    SI = -SIN(2*PI*K%*I%/N%)
    REX[K%] = REX[K%] + XR[I%]*SR - XR[I%]*SI
    IMX[K%] = IMX[K%] +XR[I%]*SI + XI[I%]*SR

  NEXT I%
NEXT K%

RETURN
```

Ważne różnice między algorytmami FFT dotyczą sposobu wprowadzania i wyprowadzania danych do i z podprogramu. W programie w BASIC-u dane wchodzą do podprogramu i opuszczają go w postaci matryc REX[ ] i IMX[ ], o próbkach od 0 do $N-1$. W programie w FORTRAN-ie dane są przekazywane w zespolonej matrycy X( ), o próbkach od 1 do N. Ponieważ jest to matryca zmiennych zespolonych, więc każda próbka w X[ ] zawiera dwie liczby – część rzeczywistą i część urojoną. Informacja o długości DFT też powinna być wprowadzona do tych podprogramów. W programie w BASIC-u do tego celu służy zmienna N%. W programie w FORTRAN-ie zaś stosuje się zmienną M, która jest zdefiniowana jako równa $log_2 N$. Na przykład, M będzie równe 8 dla 256-punktowego DFT, 12 – dla 4096 -punktowego itd. Rzecz w tym, że programista piszący podprogram FFT ma do dyspozycji wiele opcji powiązania z programem głównym. Szczególnie kłopotliwe są matryce o próbkach od 1 do N, tak jak w programie w FORTRAN-ie. W większości publikacji o DSP (z tą książką włącznie) algorytmy są objaśniane przy założeniu, że matryce zawierają próbki od 0 do $N-1$. Na przykład jeśli matryce obejmują próbki od 1 do N, to w dziedzinie częstotliwości występuje symetria wokół punktów 1 i $N/2+1$, a nie wokół 0 i $N/2$. Stosowanie zespolonego DFT do obliczania rzeczywistego DFT ma jeszcze jedną ciekawą zaletę. Symetria zespolonego DFT między dziedzinami czasu i częstotliwości jest lepsza niż rzeczywistego DFT. Ta

Tablica 12.3. FFT w FORTRAN-ie. Dane są wprowadzane do tego podprogramu w postaci zmiennych X( ) i M. Liczba całkowita M jest logarytmem o podstawie 2 z długości DFT, tj. M = 8 dla 256-punktowego DFT, M = 12 dla 4096-punktowego itd. W matrycy zespolonej X( ) są przechowywane dane z dziedziny czasu aż do ich wprowadzenia do DFT. Po powrocie z tego podprogramu do X( ) w miejsce poprzednich danych wpisuje się dane z dziedziny częstotliwości. Uwaga: w tym podprogramie jest wymagane, żeby sygnały wejściowy i wyjściowy zmieniały się od X(1) do X(N), a nie od X(0) do X(N – 1), jak zwykle przyjęto

```
      SUBROUTINE FFT(X,M)
      COMPLEX X(4096),U,S,T
      PI=3.14159265
      N=2**M
      DO 20L=1,M
      LE=2**(M+1-L)
      LE2=LE/2
      U=(1.0,0.0)
      S=CMPLX(COS(PI/FLOAT(LE2)),-SIN(PI/FLOAT(LE2)))
      DO 20 J=1,LE2
      DO 10 I=J,N,LE
      IP=I+LE2
      T=X(I)+X(IP)
      X(IP)=(X(I)-X(IP))*U
10    X(I)=T
20    U=U*S
      ND2=N/2
      NM1=N-1
      J=1
      DO 50 I=1,NM1
      IF(I.GE.J) GO TO 30
      T=X(J)
      X(J)=X(I)
      X(I)=T
30    K=ND2
40    IF(K.GE.J) GO TO 50
      J=J-K
      K=K/2
      GO TO 40
50    J=J+K
      RETURN
      END
```


więc w zespolonym DFT **dualność** jest silniejsza. Wynika stąd, między innymi, że odwrotne DFT jest prawie identyczne jak proste DFT. Rzeczywiście najłatwiejszym sposobem obliczenia *odwrotnego* FFT jest obliczenia *prostego* FFT i potem dobranie danych. W tablicy 12.5 przedstawiono podprogram do obliczania odwrotnego FFT tym sposobem.

Prog. 12.4. FFT w BASIC-u

```
'SZYBKIE PRZEKSZTAŁCENIE FOURIERA
'Po wejściu do programu N% zawiera liczbę punktów w DFT,
'REX[ ] i IMX[ ] rzeczywistą i urojoną część sygnału wejściowego. Po powrocie REX[ ] i IMX[ ]
'zawierają wynik DFT. Wszystkie sygnały zmieniają się w zakresie od 0 do N%-1.
'
PI = 3.14159265                          'Ustaw stałe
NM1% = N%-1
ND2% = N%/2
M% = CINT(LOG(N%)/LOG(2))
J% = ND2%
'
FOR I% = 1 TO N%-2                       'Sortowanie z odwróceniem bitowym
  IF I% >= J% THEN GOTO 1190
  TR = REX[J%]
  TI = IMX[J%]
  REX[J%] = REX[I%]
  IMX[J%] = IMX[I%]
  REX[I%]=TR
  IMX[I%] = TI
  K% = ND2%
  IF K% > J% THEN GOTO 1240
  J% = J%-K%
  K% = K%/2
  GOTO 1200
  J% = J%+K%
NEXT I%
'
FOR L% = 1 TO M%                         'Pętla dla każdego stopnia
  LE% = CINT(2^L%)
  LE2% = LE%/2
  UR = 1
  UI = 0
  SR = COS(PI/LE2%)                      'Oblicz wartości sinusoidy i kosinusoidy
  SI = -SIN(PI/LE2%)
  FOR J% = 1 TO LE2%                     'Pętla dla każdego podprogramu DFT
    JM1% = J%-1
    FOR I% = JM1% TO NM1% STEP LE%       'Pętla dla każdego motylka
      IP% = I%+LE2%
      TR = REX[IP%]*UR - IMX[IP%]*UI     'Obliczanie motylka
      TI = REX[IP%]*UI + IMX[IP%]*UR
      REX[IP%] = REX[I%]-TR
      IMX[IP%] = IMX[I%]-TI
      REX[I%] =REX[I%]+TR
      IMX[I%] =IMX[I%]+TI
    NEXT I%
    TR = UR
    UR = TR*SR - UI*SI
    UI = TR*SI + UI*SR
  NEXT J%
NEXT L%
'
RETURN
```

Tab. 12.5.

```
2000 'PODPROGRAM SZYBKIEGO ODWROTNEGO PRZEKSZTAŁCENIA FOURIERA
2010 'Po wejściu do programu N% zawiera liczbę punktów w IDFT, a REX[ ] i IMX[ ] zawiera
2020 'części rzeczywistą i urojoną sygnału w zespolonej dziedzinie częstotliwości.
2030 'Po powrocie REX[ ] i IMX[ ] zawierają sygnał w zespolonej dziedzinie czasu.
2040 'Wszystkie sygnały zmieniają się w zakresie od 0 do N%-1.
2050 '
2060 FOR K% = 0 TO N%-1                    'Zmień znak IMX[ ].
2070 IMX[K%] = -IMX[K%]
2080 NEXT K%
2090 '
2100 GOSUB 1000                            'Oblicz prostą transformatę FFT (tablica
2110 '
2120 FOR I% = 0 TO N%-1                    'Podziel sygnał w dziedzinie czasu przez N
2130   REX[I%] = REX[I%]/N%                'i zmień znak IMX[ ].
2140   IMX[I%] = -IMX[I%]/N%
2150 NEXT I%
2160 '
2170 RETURN
```

Załóżmy, że kopiujecie jeden z tych algorytmów FFT do programu w waszym komputerze i go uruchamiacie. Skąd wiadomo, czy działa on poprawnie? Do wykrywania błędów stosuje się zwykle dwie sztuczki. Po pierwsze, zacznijcie od jakiegoś dowolnego sygnału w dziedzinie czasu, np. z generatora liczb losowych, i poddajcie go przekształceniu FFT. Potem uzyskane widmo częstotliwościowe poddajcie odwrotnemu przekształceniu FFT i porównajcie wynik z sygnałem oryginalnym. Powinny one być *identyczne*, z pominięciem szumu spowodowanego zaokrąglaniem (kilka części na milion, przy pojedynczej dokładności).

Drugim możliwym testem prawidłowego działania jest sprawdzenie, czy sygnały mają poprawną *symetrię*. Gdy część urojona sygnału w dziedzinie czasu składa się z samych zer (normalny przypadek), to zespolone DFT jest symetryczne wokół próbek 0 i $N/2$, jak to powyżej stwierdzono.

Również gdy występuje taka poprawna symetria w dziedzinie częstotliwości, to odwrotne DFT da sygnał w dziedzinie czasu mający część urojoną zawierającą same zera (plus szum zaokrąglania). Te metody szukania błędów są bardzo ważne w zastosowaniach FFT. Dobrze je poznajcie.

Porównanie szybkości i dokładności

Przy obliczaniu DFT metodą korelacji (jak w tablicy 12.2) zastosowano w programie dwie zagnieżdżone pętle, każda przechodząca przez N punktów. Z tego wynika, że całkowita liczba operacji jest proporcjonalna do N razy N. Czas konieczny do wykonania programu jest więc określony wzorem:

$$\text{Czas wykonania} = k_{DFT} N^2$$

Wzór 12.1. Czas wykonania programu obliczania DFT. Ten czas w przypadku obliczania przez korelację jest proporcjonalny do kwadratu długości DFT

gdzie N jest liczbą punktów w DFT, a k_{DFT} – współczynnikiem proporcjonalności. Jeśli wartości sinusoid i kosinusoid są obliczane *wewnątrz* zagnieżdżonych pętli, to k_{DFT} jest równe 25 µs w Pentium 100 MHz. Jeśli wartości sinusoid i kosinusoid są obliczone wcześniej i zapamiętane w tablicach przeglądowych, to k_{DFT} zmniejsza się do 7 µs. Na przykład 1024-punktowe DFT będzie wymagało ok. 25 s, czyli prawie 25 ms na jeden punkt. To jest długo![16]

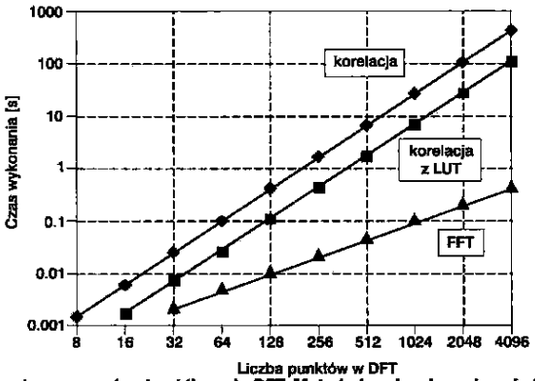
[16] Współczesne (rok 2006) komputery biurowe radzą sobie z obliczaniem 1024-punktowego DFT w czasie rzeczywistym (za: *http://www.intel.com*) (przyp. tłum.).

Stosując taką sama metodykę, można określić czas wykonania programu realizującego FFT. Czas konieczny do odwrócenia bitów jest pomijalnie mały. W każdym z $\log_2 N$ etapów jest $N/2$ obliczeń motylkowych. Zatem czas wykonania programu jest w przybliżeniu równy

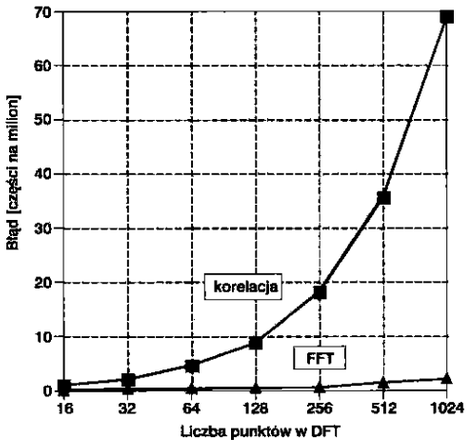
$$\text{Czas wykonania} = k_{FFT} N \log_2 N$$

Wzór 12.2. Czas wykonania programu obliczania FFT. Czas konieczny do obliczenia DFT przy użyciu FFT jest proporcjonalny do N pomnożonego przez logarytm z N

Współczynnik k_{FFT} jest równy ok. 10 µs w systemie Pentium 100 MHz. Obliczenie 1024-punktowego FFT wymaga więc ok. 70 ms, czyli 70 µs na jeden punkt. Jest to ponad 300 razy szybciej niż w przypadku DFT obliczanego metodą korelacji!



Rys. 12.8. Czasy wykonania programów do obliczania DFT. Metoda korelacyjna odpowiada algorytmowi podanemu w tablicy 12.2. Można przyśpieszyć tę metodę obliczając wcześniej wartości sinusoid i kosinusoid i rejestrując je w tablicy przeglądowej (LUT – Look-Up Table). FFT (tablica 12.3) jest algorytmem najszybszym, gdy DFT ma długość większą od 16 punktów. Czasy podane na rysunku dotyczą procesora Pentium 100 MHz



Rys. 12.9. Dokładność DFT. Przy użyciu FFT obliczanie DFT jest szybsze niż w metodzie korelacyjnej. Dlatego błędy zaokrąglania obliczeń też są mniejsze

Czynnik $Nlog_2N$ jest nie tylko mniejszy niż N^2, lecz wzrasta on znacznie wolniej ze wzrostem N. Na przykład 32-punktowe FFT jest ok. *dziesięciokrotnie* szybsze niż metoda korelacyjna. Jednak 4096-punktowe FFT jest już *tysiąckrotnie* szybsze. Dla małych wartości N (powiedzmy od 32 do 128) FFT jest bardzo przydatne, a dla dużych N (1024 i więcej) jest już absolutnie niezbędne. Na rys. 12.8 w formie graficznej porównano czasy wykonania obu algorytmów.

Przekształcenie FFT ma, oprócz szybkości, jeszcze jedną zaletę. FFT jest obliczane bardziej *dokładnie*, gdyż mniejsza liczba operacji powoduje mniejszy błąd zaokrąglania. Można to wykazać, obliczając FFT dowolnego sygnału i następnie poddając uzyskane widmo częstotliwościowe odwrotnemu FFT. Powoduje to odtworzenie oryginalnego sygnału w dziedzinie czasu, z *dodaniem* jednak szumu zaokrąglania pochodzącego z obliczeń. Wielkość charakteryzującą ten szum uzyskuje się, obliczając odchylenie standardowe różnicy między dwoma sygnałami. Dla porównania można powtórzyć taką samą procedurę, stosując DFT wyznaczane metodą korelacyjną i następnie odpowiadające mu odwrotne DFT. Jaki jest wynik takiego porównania szumu zaokrąglania w DFT i FFT? Zobaczcie sami na rys. 12.9.

Dalszy wzrost szybkości

Jest kilka sposobów jeszcze dalszego zwiększania szybkości FFT, jednakże dają one poprawę tylko ok. 20–40%. W jednym z tych sposobów rozkład w dziedzinie czasu jest zatrzymywany dwa etapy wcześniej – wówczas gdy każdy sygnał składa się tylko z czterech punktów. Zamiast obliczeń w dwóch ostatnich etapach stosuje się wysoce zoptymalizowany program do przeskoczenia od razu w dziedzinę częstotliwości, wykorzystując prostotę 4-punktowych fal sinusoidalnych i kosinusoidalnych.

W innym popularnym algorytmie wyeliminowano zbędne obliczenia związane z tym, że część urojona w dziedzinie czasu jest zerowa, a widmo częstotliwościowe – symetryczne. Innymi słowy, FFT jest tak zmodyfikowane, aby obliczać *rzeczywiste DFT* zamiast *zespolonego DFT*. Te algorytmy noszą nazwy **rzeczywistego FFT** oraz **rzeczywistego odwrotnego FFT** (lub podobne nazwy). Można od nich oczekiwać, że będą ok. 30% szybsze niż konwencjonalne programy FFT. W tablicach 12.6 i 12.7 zamieszczono programy dla tych logarytmów.

Tab. 12.6.

```
4000 'ODWROTNE FFT DLA SYGNAŁOW RZECZYWISTYCH
4010 'Po wejściu do programu N% zawiera liczbę punktów w IDFT, a REX[ ] i IMX[ ]
4020 'zawierają części rzeczywistą i urojoną sygnału w dziedzinie częstotliwości, o indeksie
4030 'zmieniającym się od 0 do N%/2. Pozostałe próbki w REX[ ] i IMX[ ] są pomijane.
4040 'Po powrocie REX[ ] zawiera sygnał w rzeczywistej dziedzinie czasu, a IMX[ ] zawiera zero.
4050 '
4060 '
4070 FOR K% = (N%/2+1) TO (N%-1)                'Symetryzuj sygnał w dziedzinie częstotliwości
4080   REX[K%] = REX[N%-K%]                     '(jak w tablicy 12.1).
4090   IMX[K%] = -IMX[N%-K%]
4100 NEXT K%
4110 '
4120 FOR K% = 0 TO N%-1                         'Dodaj do siebie części rzeczywiste i urojone
4130   REX[K%] = REX[K%]+IMX[K%]
4140 NEXT K%
4150 '
4160 GOSUB 3000                                 'Oblicz prostą rzeczywistą transformatę DSP.
4170 '
4180 FOR I% = 0 TO N%-1  'Dodaj do siebie części rzeczywiste i urojone
4190   REX[I%] = (REX[I%] + IMX[I%])/N%         'i podziel sygnał w dziedzinie czasu przez N
4200   IMX[I%] = 0
4210 NEXT I%
4220 '
4230 RETURN
```


Tab. 12.7.

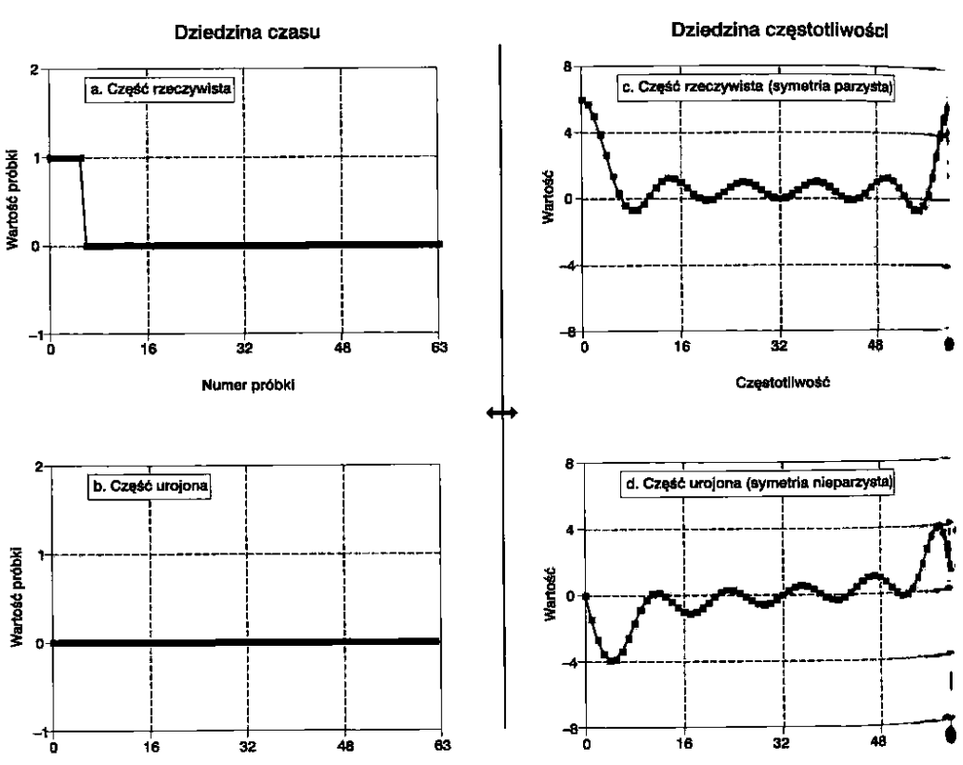
```
'FFT DLA SYGNAŁOW RZECZYWISTYCH
'Po wejściu do programu N% zawiera liczbę punktów w DFT, a REX[ ] zawiera
'rzeczywisty sygnał wejściowy a wartości IMX[ ] są pomijane. Po powrocie REX[ ] i IMX [ ]
'zawierają wynikową transformatę DFT. Wszystkie sygnały zmieniają się
'w zakresie od 0 do N%-1.
'
NH% = N%/2-1                            'Oddziel punkty nieparzyste od parzystych.
FOR I% = 0 TO NH%
  REX(I%) = REX(2*I%)
  IMX(I%) = REX(2*I%+1)
NEXT I%
'
N% = N%/2                               'Oblicz FFT o długości N%/2 punktów.
GOSUB 1000                              '(GOSUB 1000 to FFT w tablicy 12.3).
N% = N%*2
'
NM1% = N%-1                             'Rozkład parzysto-nieparzysty w dziedzinie częstotliwości
ND2% = N%/2
N4% = N%/4-1
FOR I% = 1 TO N4%
  IM% =ND2%-I%
  IP2% = I%+ND2%
  IPM% = IM%+ND2%
  REX(IP2%)= (IMX(I%) + IMX(IM%))/2
  REX(IPM%) = REX(IP2%)
  IMX(IP2%) = -(REX(I%) - REX(IM%))/2
  IMX(IPM%) = -IMX(IP2%)
  REX(I%) = (REX(I%) + REX(IM%))/2
  REX(IM%) = REX(I%)
  IMX(I%) = (IMX(I%) - IMX(IM%))/2
  IMX(IM%) =-IMX(I%)
NEXT I%
REX(N%*3/4) = IMX(N%/4)
REX(ND2%) = IMX(0)
IMX(N%*3/4) = 0
IMX(ND2%) = 0
IMX(N%/4) = 0
IMX(0) = 0
'
PI = 3.14159265                         'Zakończ ostatni stopień FFT.
L% = CINT(LOG(N%)/LOG(2))
LE% = CINT(2^L%)
LE2% = LE%/2
UR = 1
UI = 0
SR = COS(PI/LE2%)
SI = -SIN(PI/LE2%)
FOR J% = 1 TO LE2%
  JM1% = J%-1
  FOR I% = JM1 % TO NM1 % STEP LE%
    IP% = I%+LE2%
    TR = REX[IP%]*UR - IMX[IP%]*UI
    TI = REX[IP%]*UI + IMX[IP%]*UR
    REX[IP%] = REX[I%]-TR
    IMX[IP%] = IMX[I%]-TI
    REX[I%] =REX[I%]+TR
    IMX[I%] =IMX[I%]+TI
  NEXT I%
  TR = UR
  UR = TR*SR-UI*SI
  UI = TR*SI + UI*SR
NEXT J%
RETURN
```

Stosowanie *rzeczywistego FFT* ma dwie małe wady. Po pierwsze, kod programu jest tak
razy dłuższy. Po drugie, usuwanie błędów z takiego programu jest nieco trudniejsze, gdyż
nie można korzystać z symetrii jako sprawdzianu właściwego działania. W tych algorytmach
jest wymuszana zerowa część urojona w dziedzinie czasu oraz symetria lewo-prawostronna
w dziedzinie częstotliwości. W celu usunięcia błędów trzeba sprawdzić, czy te programy dają
taki sam wynik jak konwencjonalne algorytmy FFT.

Na rys. 12.10 i 12.11 pokazano, jak działa rzeczywiste FFT. Na rysunkach 12.10 a), b) przedstawiono sygnał w dziedzinie czasu, który składa się z impulsu w części rzeczywistej i z samych zer w części urojonej. Na rysunkach 12.10 c), d) przedstawiono odpowiadające tym przebiegom widmo częstotliwościowe. Jak już poprzednio powiedziano, część rzeczywista w dziedzinie częstotliwości ma symetrię *parzystą* wokół próbek 0 i $N/2$, część urojona – symetrię *nieparzystą* wokół tych samych punktów.

Teraz rozpatrzmy rys. 12.11, na którym impuls jest w części urojonej, a w części rzeczywistej są same zera. Symetria w dziedzinie częstotliwości jest tu *odwrócona*; część rzeczywista ma symetrię nieparzystą, podczas gdy urojona – parzystą. Taką sytuację omówimy w rozdziale 29. Na razie przyjmijcie do wiadomości, że tak właśnie zachowuje się zespolone DFT.

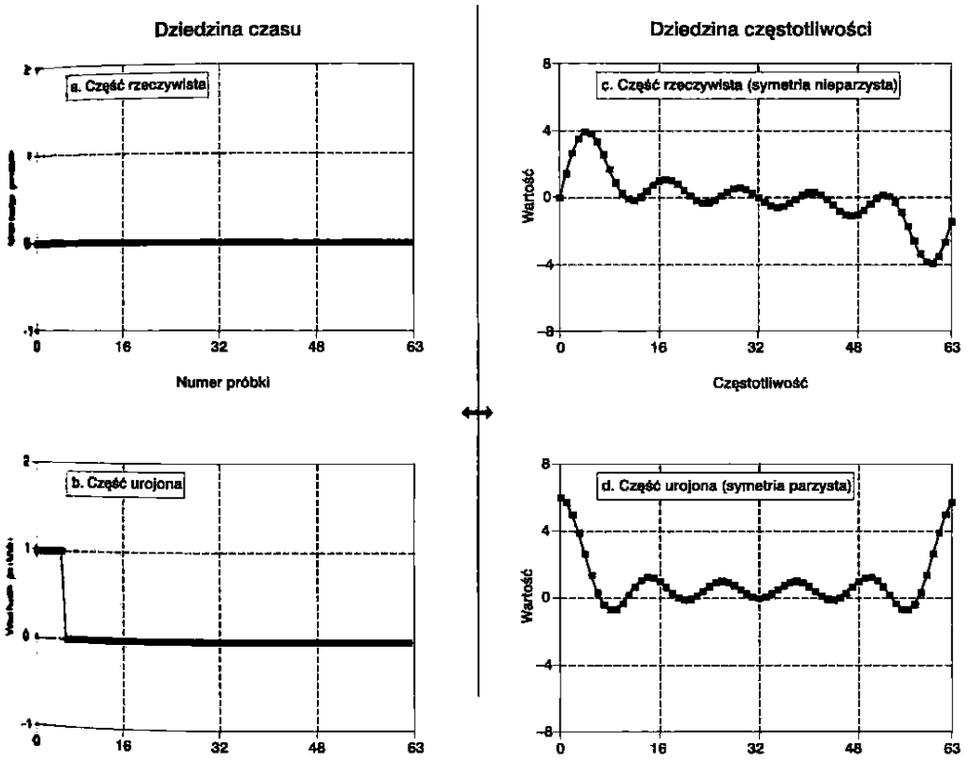
A co się dzieje, gdy sygnał znajduje się zarówno w części *rzeczywistej, jak i urojonej* w dziedzinie czasu? Zgodnie z zasadą addytywności sygnał w dziedzinie częstotliwości będzie *sumą* dwóch widm częstotliwościowych. A teraz najważniejsza sprawa – widmo częstotliwo-



Rys. 12.10. Symetria części rzeczywistej DFT

ściowe składane z widm o różnych rodzajach symetrii może być doskonale rozdzielone na te dwa sygnały składowe. Uzyskuje się to przez *rozkład parzysto-nieparzysty* opisany w rozdziale 5. Inaczej mówiąc, można wyznaczyć dwa rzeczywiste DFT przez pojedyncze FFT. Jeden z tych sygnałów jest usytuowany w części rzeczywistej w dziedzinie czasu, a drugi – w części urojonej. Po obliczeniu zespolonego DFT (oczywiście przez FFT) widma są rozdzielane przy użyciu rozkładu parzysto-nieparzystego. Ta metoda redukuje czas wykonania o ok. 40%, gdy dwa lub więcej sygnałów trzeba poddać przekształceniu FFT. To skrócenie czasu nie jest dokładnie o czynnik 2 ze względu na czas potrzebny do obliczeń związanych z rozkładem parzysto-nieparzystym. Jest to metoda stosunkowa prosta ze znacznie mniejszą liczbą pułapek czyhających na programistę niż przy pisaniu programu FFT od podstaw.

Dalszym krokiem jest modyfikacja algorytmu w taki sposób, aby *pojedyncze* DFT można było obliczać szybciej. To, co się robi w tym celu, wygląda niezbyt ładnie. Sygnał wejściowy jest rozdzielany na połowy przy użyciu rozkładu z przeplotem. Punkty parzyste (jest ich $N/2$) są umieszczone w części rzeczywistej sygnału w dziedzinie czasu, podczas gdy $N/2$ punktów nieparzystych wchodzi do części urojonej. Następnie jest obliczane $N/2$-punktowe FFT, co wymaga tylko połowy czasu potrzebnego do obliczenia FFT N-punktowego. Uzyskany w ten sposób przebieg w dziedzinie częstotliwości jest potem rozdzielany metodą rozkładu parzysto-nieparzystego, co w rezultacie daje widma częstotliwościowe dwóch przeplatanych sygnałów w dziedzinie czasu. Te dwa widma częstotliwościowe są następnie łączone ze sobą w pojedyncze widmo, tak właśnie jak w ostatnim etapie syntezy FFT.

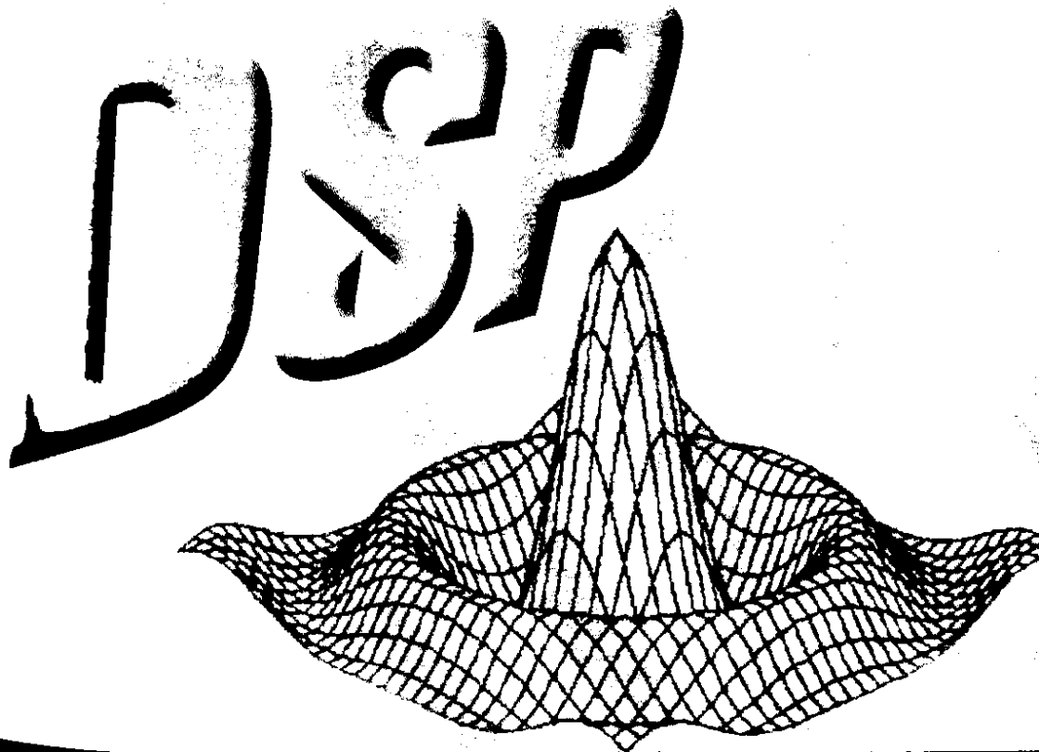


Rys. 12.11. Symetria części urojonej DFT

Podsumowując ten rozdział, trzeba podkreślić, że *FFT* jest dla *cyfrowego przetwarzania sygnałów* tym, czym *tranzystor dla elektroniki*. Przekształcenie FFT jest podstawą DSP i każdy zajmujący się tą techniką zna jego właściwości i sposoby stosowania. Jednak tylko niewielu specjalistów rzeczywiście rozumie szczegóły działania FFT.

13

Przetwarzanie sygnałów ciągłych



Przetwarzanie sygnałów ciągłych jest dziedziną podobną do DSP i stosowane metody w większości prawie identyczne. Na przykład, podstawą zarówno DSP, jak i przetwarzania sygnałów ciągłych są: liniowość, rozkład, splot i analiza Fouriera. Sygnały ciągłe nie mogą być bezpośrednio reprezentowane w komputerach cyfrowych, dlatego w tym rozdziale nie spotkacie się programów komputerowych. Przetwarzanie sygnałów ciągłych opiera się na *matyce*; sygnały są wyrażane w postaci równań, a systemy zmieniają jedne równania w Podobnie jak *komputer cyfrowy* jest podstawowym narzędziem w DSP, tak *rachunek różniczkowy i całkowy* jest takim narzędziem w przetwarzaniu sygnałów ciągłych. Te metody były stosowane od bardzo dawna, na długo przed wynalezieniem komputerów cyfrowych.

Funkcja delta

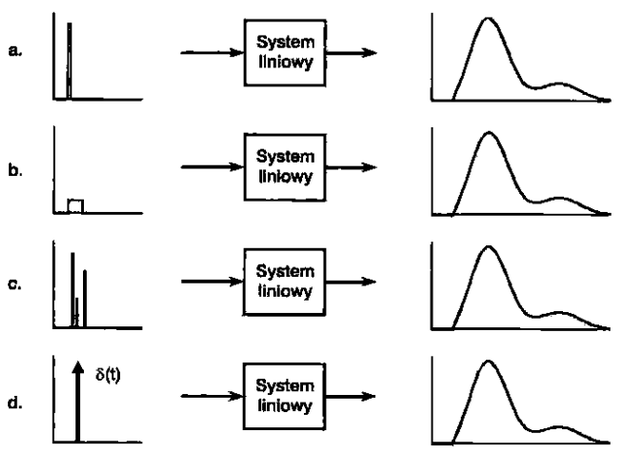
Sygnały ciągłe, podobnie jak dyskretne, mogą być poddawane rozkładowi na przeskalowane i przesunięte *funkcje delta*. Różnica polega na tym, że ciągła funkcja delta jest znacznie bardziej skomplikowana i matematycznie abstrakcyjna niż jej dyskretny odpowiednik. Zamiast określać, czym *jest* ciągła funkcja delta, zdefiniujemy ją przez *jej właściwości*.

Eksperyment myślowy umożliwi zrozumienie istoty tej funkcji. Wyobraźmy sobie układ elektroniczny złożony z elementów liniowych, takich jak rezystory, kondensatory i cewki indukcyjne. Do wejścia układu jest dołączony generator sygnałowy wytwarzający krótkie *impulsy* o różnych kształtach. Wyjście układu połączono z oscyloskopem, wyświetlającym przebiegi będące odpowiedziami układu na każdy z impulsów wejściowych. Chcemy odpowiedzieć na pytanie: *jak kształt impulsu wyjściowego jest związany z właściwościami impulsu wejściowego*? Aby uprościć to badanie, będziemy stosować tylko impulsy wejściowe znacznie krótsze niż uzyskiwane na wyjściu. Na przykład, jeśli układ odpowiada impulsami milisekundowymi, to jako impulsy wejściowe powinniśmy stosować impulsy o długości rzędu mikrosekund.

Po dokonaniu wielu pomiarów dochodzimy do trzech wniosków: Po pierwsze, *kształt* impulsu wejściowego nie ma wpływu na kształt impulsu na wyjściu. Na rys. 13.1 pokazano, że krótkie impulsy wejściowe o różnych kształtach dają na wyjściu impulsy o dokładnie jednakowym kształcie. Po drugie, kształt przebiegu wyjściowego jest całkowicie określony właściwościami systemu, tj. wartościami i konfiguracją rezystorów, kondensatorów i cewek indukcyjnych. Po trzecie, *amplituda* impulsu wyjściowego jest wprost proporcjonalna do *powierzchni* impulsu wejściowego. Na przykład impuls wyjściowy będzie miał taką samą amplitudę dla impulsów wejściowych mających amplitudy i czasy trwania: 1 V i 1 μs, 10 V i 0,1 μs albo 1000 V i 1 ns itd. Ta zależność dopuszcza też istnienie impulsów o powierzchni *ujemnej*. Wyobraźmy sobie na przykład kombinację impulsu o amplitudzie 2 V i czasie trwania 2 μs, po którym szybko następują dwa impulsy o amplitudzie –1 V, trwające 4 μs. W tym przypadku całkowita powierzchnia impulsu wejściowego jest równa *zero*, a więc na wyjściu nie uzyskuje się niczego.

Sygnały wejściowe, wystarczająco krótkie, aby mieć trzy wyżej wymienione właściwości, nazywamy **impulsami**. Impulsem jest więc każdy sygnał, który przez cały czas ma wartość zerową oprócz krótkiego *wyskoku* o dowolnym kształcie. Na przykład w układzie nadajnika mikrofalowego impulsem mogą być sygnały w zakresie *pikosekundowym*, gdyż układy elektroniczne dają odpowiedź w zakresie *nanosekundowym*. Natomiast wulkan, którego wybuch trwa parę lat, może być traktowany jako impuls dla przemian geologicznych trwających przez *tysiąclecia*.

Matematycy nie lubią być ograniczani żadnymi konkretnymi systemami i zwykle używają pojęcia *impuls* na określenie sygnału, który jest wystarczająco krótki, aby być impulsem w *każdym dowolnym systemie*. Jest to zatem sygnał *nieskończenie* krótki. **Ciągła funkcja delta** jest znormalizowaną wersją impulsu tego typu. Ciągła funkcja delta jest w szczególności definiowana matematycznie przez trzy wyidealizowane właściwości: 1) sygnał musi być nieskończenie krótki, 2) impuls musi pojawiać się w czasie zero, oraz 3) impuls musi mieć powierzchnię równą jedności.



Rys. 13.1. Ciągła funkcja delta. Jeśli sygnał na wejściu systemu liniowego jest krótki w porównaniu z sygnałem uzyskiwanym na wyjściu, to kształt przebiegu wyjściowego zależy tylko od właściwości systemu, a nie od kształtu sygnału na wejściu. Takie krótkie sygnały wejściowe są nazywane impulsami. Na rysunkach a), b), c) podano przykłady sygnałów wejściowych, które są impulsami dla tego szczególnego systemu. Termin funkcja delta jest stosowany do opisu impulsu znormalizowanego, tj. takiego, który występuje w chwili t = 0 i ma powierzchnię równą jedności. Symbole matematyczne funkcji delta przedstawiono na rys. d), są to – pionowa strzałka i δ(t)

Ponieważ funkcja delta jest zdefiniowana jako nieskończenie wąska *oraz* ma ustaloną powierzchnię, więc trzeba przyjąć, że jej amplituda jest *nieskończenie duża*. Nie przejmujcie się tym; to nie ma żadnego znaczenia. Ponieważ amplituda jest częścią *kształtu* impulsu, więc nigdy nie zetkniecie się z problemem, w którym istotny byłby fakt, czy amplituda jest, czy nie jest nieskończona. Funkcja delta jest pojęciem matematycznym, a nie sygnałem ze świata rzeczywistego. Realne sygnały, które *działają* jak funkcja delta, będą zawsze miały skończony czas trwania i amplitudę.

Tak jak w przypadku funkcji dyskretnej, ciągła funkcja delta jest oznaczana symbolem matematycznym δ( ). Sygnał wyjściowy systemu będący odpowiedzią na ciągłą funkcję delta jest nazywany **odpowiedzią impulsową** i jest często oznaczany przez *h*( ). Proszę zauważyć, że nawiasy okrągłe ( ) stosuje się do oznaczania sygnałów ciągłych, a kwadratowe [ ] – sygnałów dyskretnych. Taki zapis stosuje się w tej książce i prawie wszędzie w DSP, nie jest to jednak zapis uniwersalny. Na wykresach impulsy są przedstawiane jako pionowe strzałki (parz rys. 13.1 d) o *długościach* wyrażających *powierzchnie* impulsów.

Chcąc lepiej zrozumieć impulsy w świecie rzeczywistym, spójrzcie na nocnym niebie na jakąś *planetę* i na jakąś *gwiazdę*, np. na Marsa i Syriusza. Wydaje się, że obie mają – dla oka nieuzbrojonego – taką samą jasność i wymiar, Przyczyna takiego podobieństwa nie jest oczywista, bo geometria oglądania jest w obu przypadkach drastycznie różna. Mars ma ok. 6000 km średnicy i znajduje się 60 mln km od Ziemi, Syriusz zaś jest ok. 300 razy większy i ponad milion razy bardziej oddalony. Te liczby powinny dawać wrażenie, że Mars jest ponad *trzy tysiące* razy większy od Syriusza. Jak to jest możliwe, że oba te obiekty wyglądają podobnie?

Mars i Syriusz wydają się nam jednakowe, będąc na tyle małe, że dla ludzkiego układu wzrokowego są *impulsami*. Dostrzegany kształt jest odpowiedzią impulsową oka, a nie rzeczywistym obrazem gwiazdy lub planety. Ten fakt staje się oczywisty, gdy te dwa obiekt obejrzymy przez mały teleskop. Mars ma wtedy wygląd ciemnej tarczy, podczas gdy Syriusz nadal wydaje się być jasnym impulsem. Z tej także przyczyny gwiazdy migoczą, a planety nie. Obraz gwiazdy jest wystarczająco mały, aby mógł być na krótko blokowany w atmosferze przez pyłki lub turbulencje, których wpływ na obraz planety jest znacznie słabszy.

Splot

Podobnie jak w przypadku sygnałów dyskretnych, splot sygnału ciągłego może być rozpatrywany od strony *sygnału wejściowego* albo *wyjściowego*. Rozpatrywanie od strony wejścia jest najlepszym opisem *pojęciowym* wyjaśniającym działanie splotu. Rozpatrywanie od strony wyjścia zaś daje opis *matematyczny*. Te opisy są w istocie identyczne z zaprezentowanymi w rozdziale 6 dla sygnałów dyskretnych.

Na rys. 13.2 pokazano, jak splot jest widziany od strony wejścia. Sygnał wejściowy na przechodzi przez system charakteryzujący się odpowiedzią impulsową $h(t)$, dając na wyjściu sygnał $y(t)$. Można to zapisać znanym równaniem: $y(t) = x(t) * h(t)$. Sygnał wejściowy jest podzielony na wąskie kolumny, z których każda jest wystarczająco wąska, aby przez system być traktowana jako *impuls*. Innymi słowy, sygnał wejściowy zostaje rozłożony na nieskończoną liczbę przeskalowanych i przesuniętych wersji funkcji delta. Każdy z tych impulsów wytwarza w sygnale wyjściowym przeskalowaną i przesuniętą wersję odpowiedzi impulsowej. Uzyskany sygnał wyjściowy jest równy łącznemu wynikowi, a więc sumie wszystkich odpowiedzi indywidualnych.

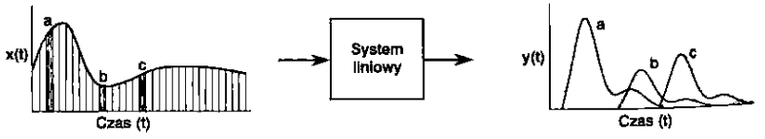
Przy takim sposobie działania szerokość kolumn musi być znacznie mniejsza od szerokości odpowiedzi systemu. Oczywiście matematycy doprowadzają to do skrajności, czyniąc te elementy wejściowe *nieskończenie* wąskimi i sprowadzając zagadnienie do problemu obliczeniowego. Tak więc rozpatrywanie od strony wejścia opisuje, jak pojedynczy punkt (lub wąski obszar) w sygnale wejściowym wpływa na większy fragment sygnału wyjściowego.

Rozpatrywanie zaś od strony wyjścia powoduje sprawdzenie, w jaki sposób pojedynczy punkt w sygnale wyjściowym jest wyznaczony przez różne wartości pochodzące z sygnału wejściowego. Tak jak w przypadku sygnałów dyskretnych, na każdą wartość chwilową sygnału wyjściowego ma wpływ jedna sekcja sygnału wejściowego, mnożona (czyli ważona) przez odpowiedź impulsową odwróconą lewo-prawostronnie. W technice dyskretnej sygnały są przemnażane i *sumowane*. Sygnały ciągłe są przemnażane i *całkowane*. Można to wyrazić wzorem:

$$y(t)=\int_{-\infty}^{+\infty} x(\tau)h(t-\tau)\,d\tau$$

Wzór 13.1. Całka splotowa. Ten wzór wyjaśnia znaczenie wyrażenia: y(t) = x(t) * h(t)

Powyższe wyrażenie jest nazywane całką splotową i jest bliźniaczą odmianą sumy splotowej (wzór 6.1) stosowanej do sygnałów dyskretnych. Rys. 13.3 ma ułatwić zrozumienie powyższego wzoru. Celem jest znalezienie wyrażenia do obliczenia wartości sygnału wyjściowego w dowolnej chwili t. Pierwszym krokiem jest zmiana zmiennej niezależnej stosowanej do przemieszczania się przez sygnał wejściowy i przez odpowiedź impulsową. Zamienia się więc zmienną t na τ. Wtedy $x(t)$ i $h(t)$ zmieniają się, odpowiednio, w $x(\tau)$ i $h(\tau)$. Taka zmiana nazw zmiennych jest potrzebna, gdyż t jest już używane do reprezentowania punktu w obliczanym sygnale wyjściowym. Następnym krokiem jest lewo-prawostronne odwrócenie odpowiedzi impulsowej, co daje zamianę jej w $h(-\tau)$. Przesunięcie odwróconej odpowiedzi impulsowej do położenia t daje wyrażenie $h(t-\tau)$. Sygnał wejściowy jest następnie mnożony (czyli ważony) przez odwróconą i przesuniętą odpowiedź impulsową, dając $x(\tau)\,h(t-\tau)$.



Rys. 13.2. Splot rozpatrywany od strony wejścia. Sygnał wejściowy x(t) jest dzielony na wąskie segmenty działające w systemie jako impulsy. Sygnał wyjściowy y(t) jest sumą uzyskanych przeskalowanych i przesuniętych odpowiedzi impulsowych. Na tym rysunku pokazano, jaki udział w sygnale wyjściowym mają trzy punkty sygnału wejściowego

Potem wartość sygnału wyjściowego jest wyznaczana przez całkowanie tego iloczynu w granicach od minus do plus nieskończoności, zgodnie z rys. 13.1.

Jeśli macie trudności ze zrozumieniem tych rozważań, powróćcie do rozdziału 6 i przypomnijcie sobie te same koncepcje zastosowane do sygnałów dyskretnych. Na rys. 13.3 podano inny sposób opisu maszyny splotowej z rys. 6.8. Jedyna różnica jest taka, że użyto całek zamiast sumowań. Nie ma tu więc niczego nowego, a tylko rozszerzenie już znanych informacji.

Na przykładzie pokażemy sposób zastosowania splotu ciągłego w problemach praktycznych oraz zastanowimy się, jakie operacje matematyczne są wtedy niezbędne. Na rys. 13.4 przedstawiono prosty system liniowy z sygnałami ciągłymi – elektroniczny filtr dolnoprzepustowy złożony z rezystora i kondensatora. Impuls wchodzący do tego systemu daje na wyjściu szybki skok sygnału do pewnej wartości i następnie wykładnicze opadanie w kierunku zera. Tak więc odpowiedź impulsowa tego prostego układu elektronicznego jest *jednostronną funkcją eksponencjalną*. Wyrażając matematycznie odpowiedź impulsową tego systemu, trzeba ją podzielić na dwa odcinki wyrażone wzorami:

$$h(t)=0 \qquad dla\ t<0$$

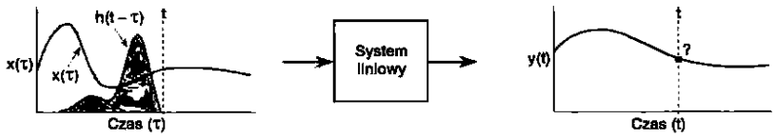
$$h(t)=\alpha e^{-\alpha t} \quad dla\ t\geq 0$$

gdzie $\alpha = 1/RC$ (R w omach, C w faradach, t w sekundach). Tak jak w przypadku sygnałów dyskretnych, ciągła odpowiedź impulsowa zawiera pełną informację o systemie, czyli o jego reakcji na wszystkie możliwe sygnały. Kontynuując ten przykład, załóżmy, że na wejściu układu jest tym razem impuls prostokątny (rys. 13.5) wyrażony przez:

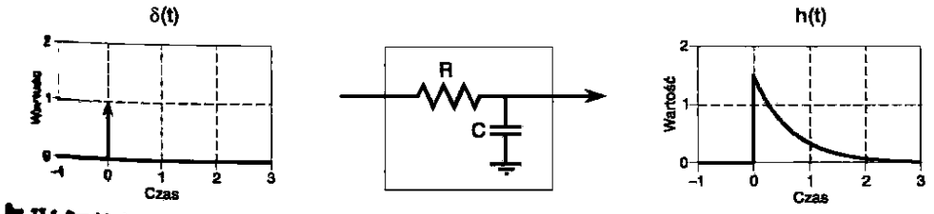
$$x(t)=1 \ \ dla\ 0\leq t\leq 1$$

$$x(t)=0 \ \ \text{we wszystkich innych przypadkach}$$

Ponieważ zarówno sygnał wejściowy, jak i odpowiedź impulsowa są dobrze znane w postaci wyrażeń matematycznych, więc sygnał wyjściowy $y(t)$ można obliczyć jako całkę splotową wg wzoru 13.1. Sprawę komplikuje fakt, że oba sygnały są zdefiniowane raczej w pewnych *obszarach*, a nie jako jedno wyrażenie matematyczne. Często tak bywa w przetwarzaniu sygnałów ciągłych. Zwykle robi się wykres pokazujący, jak dwa sygnały przesuwają się wobec siebie dla różnych wartości t. W omawianym przykładzie dwa sygnały wcale nie zachodzą



Rys. 13.3. Splot widziany od strony wyjścia. Na każdą wartość w sygnale wyjściowym ma wpływ wiele punktów z sygnału wejściowego. Na tym rysunku jest obliczana wartość sygnału wyjściowego w chwili t. Sygnał wejściowy x(τ) jest mnożony (czyli ważony) przez odwróconą lewo-prawostronnie i przesuniętą odpowiedź impulsową h(t − τ). Całkowanie tego ważonego sygnału wejściowego daje wartość punktu y(t) w sygnale wyjściowym



Rys. 13.4. Przykład systemu liniowego o sygnałach ciągłych. Ten układ elektroniczny jest filtrem dolnoprzepustowym złożonym z rezystora i kondensatora. Odpowiedzią impulsową tego systemu jest jednostronna funkcja eksponencjalna

na siebie dla $t < 0$ (rys. 13.6 a). To powoduje, że iloczyn obu sygnałów jest równy zer wszystkich położeń wzdłuż osi τ i sygnał wyjściowy jest:

$$y(t)=0 \quad dla\ t<0$$

Drugi przypadek (t od 0 do 1) pokazano na rys. 13.6 b). Tu dwa sygnały częściowo na bie zachodzą, w wyniku czego ich iloczyn ma wartości niezerowe między $\tau = 0$ oraz τ Ponieważ jest to jedyny obszar niezerowy, więc trzeba obliczyć całkę. A zatem sygnał ściowy dla $0 \le t \le 1$ jest określony przez:

$$y(t)=\int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \quad \text{(zaczynamy od wzoru 13.1)}$$

$$y(t)=\int_{0}^{t} 1\cdot\alpha e^{-\alpha(t-\tau)}d\tau \quad \text{(wstawiamy sygnały)}$$

$$y(t)=e^{-\alpha t}\left[e^{\alpha\tau}\right]\Big|_{0}^{t} \quad \text{(wprowadzamy całkowanie)}$$

$$y(t)=e^{-\alpha t}\left[e^{\alpha t}-1\right] \quad \text{(obliczamy całkę)}$$

$$y(t)=1-e^{-\alpha t} \quad \text{dla } 0 \le t \le 1$$

Na rys. 13.6 c) przedstawiono obliczanie trzeciego odcinka sygnału wyjściowego, w t $t > 1$. Tu zachodzenie sygnałów na siebie następuje między $\tau = 0$ a $\tau = 1$. Obliczenia są te same, jak w drugim odcinku sygnału, ze zmianą tylko granic całkowania:

$$y(t)=\int_{0}^{1} 1\cdot\alpha e^{-\alpha(t-\tau)}d\tau \quad \text{(wstawienie do wzoru 13.1)}$$

$$y(t)=e^{-\alpha t}\left[e^{\alpha\tau}\right]\Big|_{0}^{1} \quad \text{(wyznaczenie całki)}$$

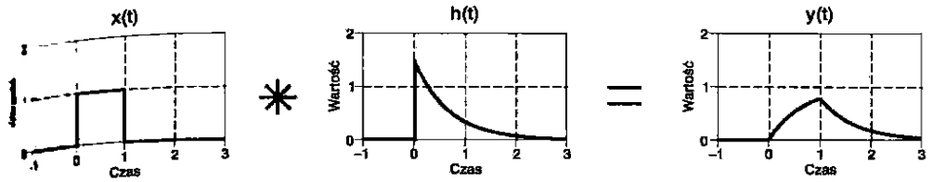
$$y(t)=\left[e^{\alpha}-1\right]e^{-\alpha t} \quad \text{dla } t>1$$

Przebieg uzyskany w każdym z trzech odcinków powinien być zgodny z waszą wiedz o układach elektronicznych:

1) Sygnał wyjściowy powinien być zerowy aż do chwili, gdy przebieg wejściowy stanie się niezerowy. Tak więc pierwszy odcinek jest opisany jako $y(t) = 0$ dla $t < 0$.
2) Gdy następuje skok, sygnał na wyjściu obwodu RC narasta eksponencjalnie dążąc do zgnięcia wartości sygnału wejściowego, zgodnie ze wzorem: $y(t) = 1 - e^{-t}$.
3) Gdy sygnał wejściowy powróci do poziomu zerowego, sygnał wyjściowy zaczyna opa eksponencjalnie dążąc do zera, zgodnie ze wzorem: $y(t) = ke^{-t}$ (gdzie $k = e^{-} - 1$, napięc na kondensatorze tuż przed rozpoczęciem rozładowywania).

W ten sam sposób można operować przebiegami bardziej skomplikowanymi, chociaż ż żoność operacji matematycznych może w pewnej chwili stać się trudna do pokonania. staniecie przed kłopotliwym zagadnieniem splotu ciągłego, powinniście poświęcić spor su na określeniu *strategii* rozwiązania problemu. Jeśli bez zastanowienia zaczniecie oblic całki, to możecie dojść do kompletnego bałaganu matematycznego. Często stosowana st gia polega na rozdzieleniu jednego z sygnałów na prostsze składowe addytywne, które *widualnie* można poddawać splotowi. Stosując zasady liniowości, można sumować uzys przebiegi i w ten sposób uzyskać rozwiązanie postawionego problemu.

Inną strategię przedstawiono na rys. 13.7. Modyfikuje się jeden z sygnałów w jakiś sposób, wykonuje się splot i potem anuluje się wprowadzoną modyfikację. W podanym kładzie modyfikacją jest *różniczkowanie*, a jego anulowanie polega na *scałkowaniu*. Po impulsu prostokątnego o jednostkowej amplitudzie są *dwa impulsy*, pierwszy o powie równej jeden, a drugi – minus jeden. Aby to lepiej zrozumieć, pomyślcie o odwrotnym



Rys. 13.5. Przykład splotu ciągłego. Na rysunku pokazano sygnał prostokątny na wejściu dolnoprzepustowego filtru RC (rys. 13.4). Sygnał wyjściowy uzyskuje się jako splot impulsu prostokątnego i odpowiedzi impulsowej systemu

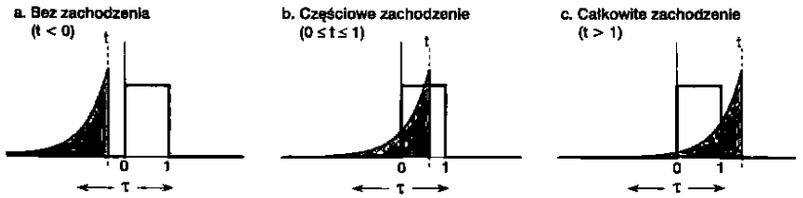
całkowania tych dwóch impulsów. Przy całkowaniu pierwszego impulsu wartość całki nagle wzrasta od zera do jedności, więc daje skok jednostkowy. Po przejściu impulsu ujemnego wartość całki sygnału nagle powraca od jedności do zera, dając w rezultacie impuls prostokątny.

Różniczkowanie upraszcza ten problem, ponieważ wyznaczanie splotu jest łatwe, gdy jeden z sygnałów jest złożony z impulsów. Każdy z dwóch impulsów w przebiegu $x'(t)$ wnosi przeskalowaną i przesuniętą wersję odpowiedzi impulsowej do pochodnej sygnału wyjściowego $y'(t)$. Tak więc: $y'(t) = h(t) - h(t - 1)$. Sygnał wyjściowy $y(t)$ można wyznaczyć przez wstawienie wyrażenia określającego $h(t)$ i scałkowanie uzyskanego wzoru.

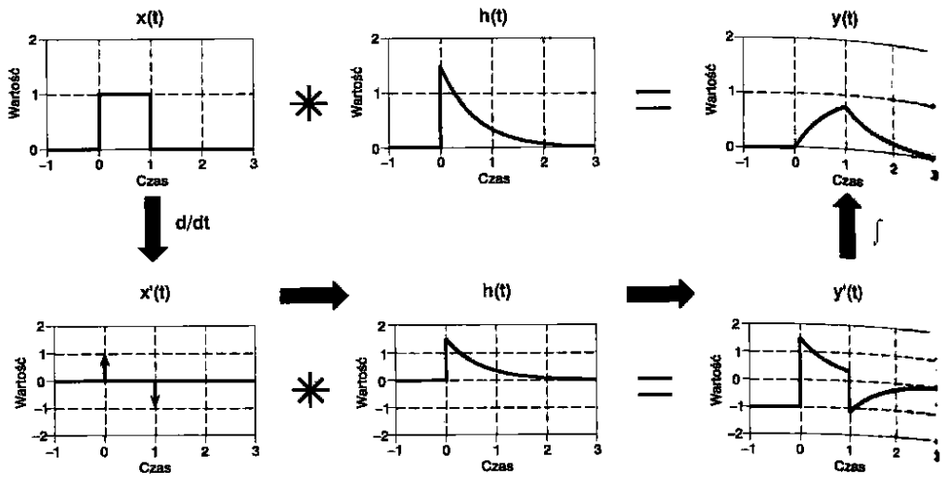
Drobną niedogodnością tej procedury jest, przy różniczkowaniu, utrata składowej stałej sygnału wejściowego. To może spowodować błąd wartości tej składowej w obliczonym sygnale wyjściowym. Matematycznie traktuje się tę składową jako dowolną stałą, którą można dodać podczas całkowania. Nie ma żadnego sposobu systematycznej identyfikacji tego błędu, lecz można go zwykle skorygować przez sprawdzenie całego obliczenia. Tak na przykład nie ma błędu składowej stałej w przykładzie na rys. 13.7. Jest to wiadome, gdyż obliczony sygnał wyjściowy ma prawidłową wartość składowej stałej, gdy t staje się bardzo duże. Gdy w jakimś szczególnym zadaniu wystąpi ten błąd, to można obliczenie uzupełnić o odpowiedni składnik stały.

Ta metoda działa także dla sygnałów, które można zredukować do impulsów przez *wielokrotne* różniczkowanie. Takie sygnały są w nazewnictwie stosowanym w DSP określane jako *sygnały odcinkowo-wielomianowe* (*piecewise polynomials*). Po wykonaniu splotu, początkowa operacja różniczkowania wielokrotnego jest anulowana przez wielokrotne całkowanie. Jedyna pułapka polega tu na tym, że utracona wartość składowej stałej musi być wyznaczana w każdym etapie całkowania przez znalezienie prawidłowej stałej całkowania.

Zanim zaczniecie rozwiązywać jakieś zadanie z wykorzystaniem trudnej metody ze splotem sygnałów ciągłych, powinniście rozważyć możliwość innego podejścia do postawionego problemu. Postawcie sobie pytanie: *Czy rzeczywiście musimy znać matematyczne wyrażenie sygnału wyjściowego, czy może wystarczy nam wykres przebiegu.* Jeśli wystarczy wykres, to lepiej będzie rozwiązywać problem metodami techniki *dyskretnej*. Trzeba wtedy aproksymować przebieg ciągły próbkami, na których operację splotu można realizować bezpośrednio programem komputerowym. Jest to sposób znacznie łatwiejszy, choć nie jest czysty matematycznie.



Rys. 13.6. Obliczanie splotu odcinkami. Ponieważ wiele sygnałów ciągłych jest definiowanych w pewnych obszarach, więc splot musi być obliczany obszar po obszarze. W tym przykładzie obliczanie sygnału wyjściowego jest rozdzielone na trzy odcinki: a) – bez zachodzenia na siebie sygnału wejściowego i przesuniętej, odwróconej lewo-prawostronnie odpowiedzi impulsowej, b) – z częściowym zachodzeniem tych sygnałów, c) – z całkowitym zachodzeniem



Rys. 13.7. Strategia splatania sygnałów. Problemy związane ze splotem można często uprościć przez sprytne wykorzystanie zasad rządzących systemami liniowymi. W tym przykładzie splot dwóch sygnałów jest uproszczony przez zróżniczkowanie jednego z nich. Po wykonaniu splotu różniczkowanie jest anulowane przez całkowanie

Przekształcenie Fouriera

W przekształceniu Fouriera dla sygnałów ciągłych można wyróżnić dwie kategorie: przekształcenie dla sygnałów *okresowych* oraz dla sygnałów *nieokresowych*. Do sygnałów okresowych stosuje się wersję przekształcenia Fouriera zwaną **szeregiem Fouriera**. Będzie ona omówiony w następnym podrozdziale. Przekształcenie Fouriera stosowane do sygnałów nieokresowych nosi nazwę po prostu **przekształcenia Fouriera**. W tym rozdziale te techniki fourierowskie są omawiane przy użyciu tylko liczb *rzeczywistych*, tak jak to uczyniono w poprzednich kilku rozdziałach w odniesieniu do sygnałów dyskretnych. Znacznie silniejsze narzędzie, jakim są liczby zespolone, pozostawiamy do rozpatrzenia w rozdziale 31.

Na rys. 13.8 przedstawiono przykład ciągłego sygnału nieokresowego i jego widmo częstotliwościowe. Sygnał w dziedzinie czasu obejmuje zakres od minus do plus nieskończoności, a każdy z sygnałów w dziedzinie częstotliwości ma zakres od zera do plus nieskończoności. To widmo częstotliwościowe zaprezentowano we współrzędnych prostokątnych (część rzeczywista i urojona), lecz współrzędne biegunowe (moduł i faza) też są stosowane do sygnałów ciągłych. Podobnie jak w przypadku sygnałów dyskretnych, **równanie syntezy** daje przepis na stworzenie sygnału w dziedzinie czasu na podstawie danych z dziedziny częstotliwości. Równanie ma postać matematyczną:

$$x(t)=\frac{1}{\pi}\int_{0}^{+\infty} ReX(\omega)\cos(\omega t)-ImX(\omega)\sin(\omega t)\,d\omega$$

Wzór 13.2. Przekształcenie Fouriera, równanie syntezy. W tym wzorze x(t) jest wyznaczanym przez syntezę sygnałem w dziedzinie czasu, a Re X(ω) i Im X(ω) to części rzeczywista i urojona widma częstotliwościowego

Omawiając słownie powyższe wyrażenie, trzeba stwierdzić, że sygnał w dziedzinie czasu jest tworzony przez sumowanie (drogą całkowania) nieskończonej liczby przeskalowanych fal sinusoidalnych i kosinusoidalnych. Część rzeczywista przebiegu w dziedzinie częstotliwości zawiera współczynniki skalowania fal kosinusoidalnych, a część urojona – fal sinusoidalnych. Tak jak w przypadku sygnałów dyskretnych, równanie syntezy jest zwykle zapisywane z *ujemnymi* falami sinusoidalnymi. Chociaż znak minus nie ma znaczenia w tych rozważaniach, to jest on konieczny, aby sposób zapisu uczynić zgodnym z rachunkiem na liczbach

zespolonych, omówionym w rozdziale 29. Należy pamiętać, że niektórzy autorzy stawiają ten znak minus we wzorze, a inni nie. Zauważcie też, że częstotliwość jest wyrażana symbolem ω. Jak pamiętacie, taki zapis oznacza **pulsację** (*natural frequency*), której jednostką są radiany na sekundę. Obowiązuje zależność $\omega = 2\pi f$, gdzie f jest częstotliwością w okresach na sekundę (Hz). Zapis z użyciem pulsacji preferują matematycy i ci wszyscy, którzy odpowiedź na problemy DSP znajdują przez *rozwiązywanie równań*, gdyż przy stosowaniu pulsacji liczba zapisywanych symboli jest na ogół mniejsza.

W **równaniach analizy** dla sygnałów ciągłych jest stosowana taka sama metoda, jak w przypadku sygnałów dyskretnych: *korelacja* z falami kosinusoidalnymi i sinusoidalnymi. Te równania mają postać:

$$ReX(\omega)=\int_{-\infty}^{+\infty} x(t)\cos(\omega t)\,dt$$

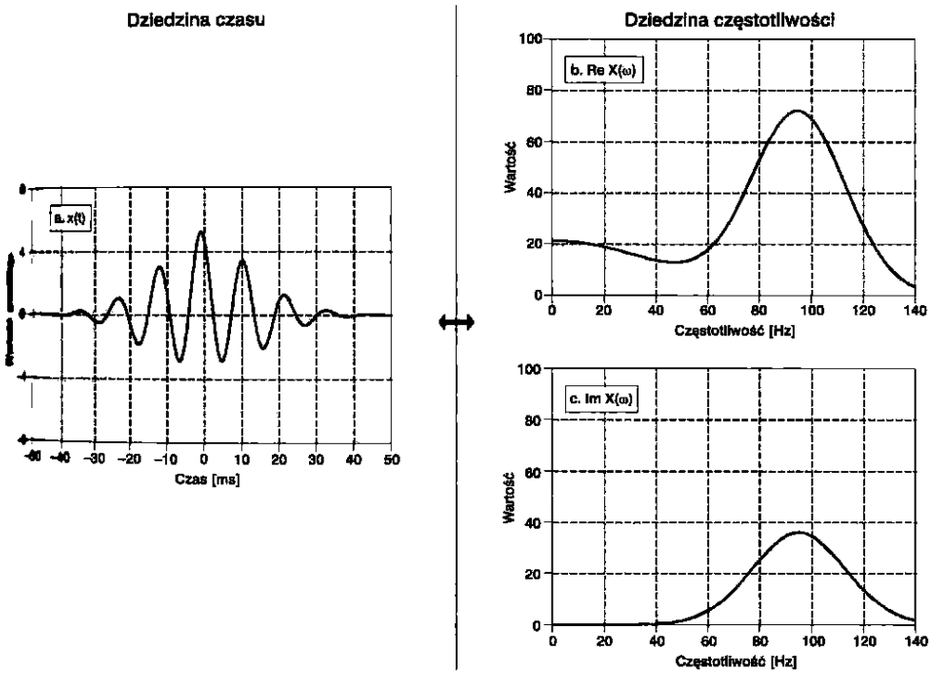
$$ImX(\omega)=-\int_{-\infty}^{+\infty} x(t)\sin(\omega t)\,dt$$

Wzór 13.3. Przekształcenie Fouriera, równania analizy. W tych wzorach Re X(ω) i Im X(ω) są częściami rzeczywistą i urojoną widma częstotliwościowego, a x(t) analizowanym sygnałem w dziedzinie czasu

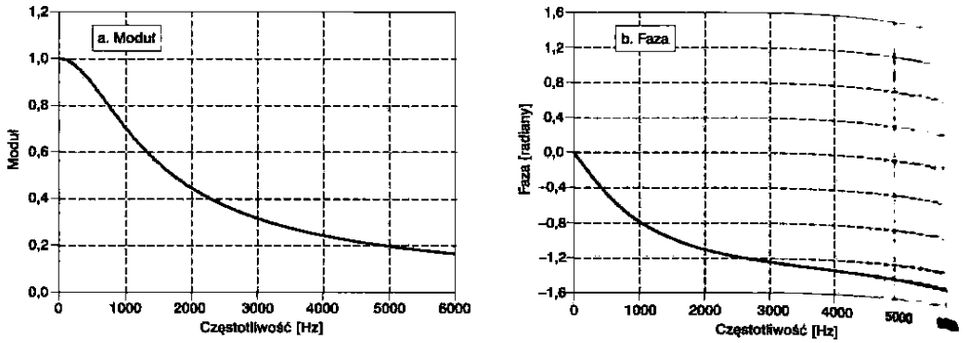
Jako przykład zastosowania równań analizy określimy odpowiedź częstotliwościową dolnoprzepustowego filtru RC. W tym celu wyznacza się transformatę Fouriera odpowiedzi impulsowej tego układu, już poprzednio przedstawionej na rys. 13.4 i opisanej wzorami:

$$h(t)=0 \qquad dla\ t<0$$

$$h(t)=\alpha e^{-\alpha t} \qquad dla\ t\geq 0$$



Rys. 13.8. Przykład przekształcenia Fouriera. Sygnał w dziedzinie czasu x(t) obejmuje zakres od minus do plus nieskończoności. Sygnał w dziedzinie częstotliwości składa się z części rzeczywistej Re X(ω) i części urojonej Im X(ω), z których każda obejmuje zakres od zera do nieskończoności dodatniej. Oś częstotliwości jest na tym wykresie wyskalowana w okresach na sekundę (w Hz). W celu przeliczenia tych wartości na pulsację trzeba je pomnożyć przez 2π



Rys. 13.9. Odpowiedź częstotliwościowa dolnoprzepustowego filtru RC. Te krzywe uzyskano, obliczając transformatę Fouriera odpowiedzi impulsowej i wyrażając ją następnie we współrzędnych biegunowych

Odpowiedź częstotliwościową znajduje się, wstawiając odpowiedź impulsową do równania analizy. Najpierw obliczamy część rzeczywistą:

$$Re\,H(\omega)=\int_{-\infty}^{+\infty} h(t)\cos(\omega t)\,dt \qquad \text{(rozpoczęcie od wzoru 13.3)}$$

$$Re\,H(\omega)=\int_{0}^{+\infty} \alpha e^{-\alpha t}\cos(\omega t)\,dt \qquad \text{(wstawienie wartości sygnału)}$$

$$Re\,H(\omega)=\frac{\alpha e^{-\alpha t}}{\alpha^2+\omega^2}\left[-\alpha\cos(\omega t)+\omega\sin(\omega t)\right]\Bigg|_{0}^{+\infty} \qquad \text{(wyznaczenie całki)}$$

$$Re\,H(\omega)=\frac{\alpha^2}{\alpha^2+\omega^2}$$

W taki sam sposób obliczamy część urojoną odpowiedzi częstotliwościowej:

$$Im\,H(\omega)=\frac{-\omega\alpha}{\alpha^2+\omega^2}$$

Zapis we współrzędnych prostokątnych jest, podobnie jak dla sygnałów dyskretnych, bardzo korzystny do operacji matematycznych, lecz trudny do zrozumienia. Można łatwo przejść na zapis we współrzędnych biegunowych, korzystając ze standardowych zależności: *Mag* $H(\omega) = [Re\,H(\omega)^2 + Im\,H(\omega)^2]^{1/2}$ oraz *Phase* $H(\omega) = \text{arctg}\,[Re\,H(\omega) / Im\,H(\omega)]$.

Korzystając z przekształceń algebraicznych, uzyskujemy odpowiedź częstotliwościową dla dolnoprzepustowego filtru *RC* zapisaną we współrzędnych biegunowych w postaci modułu i fazy:

$$Mag\,H(\omega)=\frac{\alpha}{\left[\alpha^2+\omega^2\right]^{1/2}}$$

$$Phase\,H(\omega)=\text{arctg}\left[-\frac{\omega}{\alpha}\right]$$

Na rys. 13.9 pokazano wykresy modułu i fazy dla częstotliwości odcięcia równej 1000 Hz (tj. $\alpha = 2\pi 1000$).

Szereg Fouriera

Dochodzimy teraz do ostatniego przekształcenia z rodziny transformat fourierowskich – do *szeregu Fouriera*. Sygnał w dziedzinie czasu stosowany w przekształceniu Fouriera jest okresowy i *ciągły*. Na rys. 13.10 podano kilka przykładów przebiegów ciągłych powtarzających

się od minus do plus nieskończoności. W rozdziale 11 wykazano, że sygnały okresowe mają widmo częstotliwościowe złożone z **harmonicznych**. Jeśli na przykład przebieg w dziedzinie czasu powtarza się z częstotliwością 1000 Hz (okres 1 ms), to widmo częstotliwościowe będzie zawierało pierwszą harmoniczną 1000 Hz, drugą 2000 Hz, trzecią 3000 Hz itd. Pierwsza harmoniczna, a więc częstotliwość, z jaką powtarza się sygnał w dziedzinie czasu, nosi nazwę **częstotliwości podstawowej**. Można więc widmo częstotliwościowe rozpatrywać dwoma sposobami: 1) widmo częstotliwościowe jest *ciągłe*, lecz równe zero dla wszystkich częstotliwości oprócz harmonicznych, lub 2) widmo częstotliwościowe jest *dyskretne*, lecz *określone* tylko dla częstotliwości harmonicznych. Innymi słowy, częstotliwości między harmonicznymi można traktować jako mające wartość zero lub po prostu jako nieistniejące. Ważne jest to, że nie wnoszą one swego udziału do tworzenia sygnału w dziedzinie czasu.

Równanie syntezy szeregu Fouriera daje ciągły sygnał okresowy o częstotliwości podstawowej f przez sumowanie przeskalowanych fal sinusoidalnych i kosinusoidalnych o częstotliwościach f, $2f$, $3f$, $4f$, itd. Amplitudy fal kosinusoidalnych są zawarte we współczynnikach: a_1, a_2, a_3, a_4, itd., a sinusoidalnych w: b_1, b_2, b_3, b_4, itd. Współczynniki a i b są więc, odpowiednio, rzeczywistymi i urojonymi częściami widma częstotliwościowego. Dodatkowo stosuje się współczynnik a_0, aby zachować składową stałą przebiegu w dziedzinie czasu. Można ten współczynnik traktować jako amplitudę fali kosinusoidalnej o częstotliwości zero (wartość stałą). Czasem współczynnik a_0 jest grupowany razem z innymi współczynnikami a, lecz często jest traktowany jako zupełnie oddzielny, gdyż wymaga specjalnych obliczeń. Nie jest potrzebny współczynnik b_0, gdyż fala sinusoidalna dla częstotliwości zero ma wartość stałą równą zero. Równanie syntezy ma postać:

$$x(t)=a_0+\sum_{n=1}^{\infty} a_n\cos(2\pi f\,t\,n)-\sum_{n=1}^{\infty} b_n\sin(2\pi f\,t\,n)$$

Wzór 13.4. Szereg Fouriera, równanie syntezy. Każdy sygnał okresowy x(t) może być odtworzony z fal sinusoidalnych i kosinusoidalnych o częstotliwościach będących wielokrotnościami częstotliwości podstawowej f. Współczynniki a_n, b_n zawierają amplitudy fal kosinusoidalnych i sinusoidalnych

W zapisie równań **analizy** dla szeregu Fouriera używa się zwykle *okresu* przebiegu, oznaczanego T, zamiast częstotliwości podstawowej f (przy czym $f = 1/T$). Ponieważ sygnał w dziedzinie czasu jest okresowy, więc korelację fal sinusoidalnych i kosinusoidalnych można wyznaczyć tylko na podstawie jednego okresu, tj. od $-T/2$ do $T/2$, od 0 do T, od $-T$ do 0 itd. Wybór różnych granic całkowania zmienia trochę tok obliczeń, lecz rezultat końcowy jest zawsze taki sam. Równania analizy dla szeregu Fouriera mają postać:

$$a_0=\frac{1}{T}\int_{-T/2}^{T/2} x(t)\,dt$$

$$a_n=\frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos\left(\frac{2\pi tn}{T}\right)dt$$

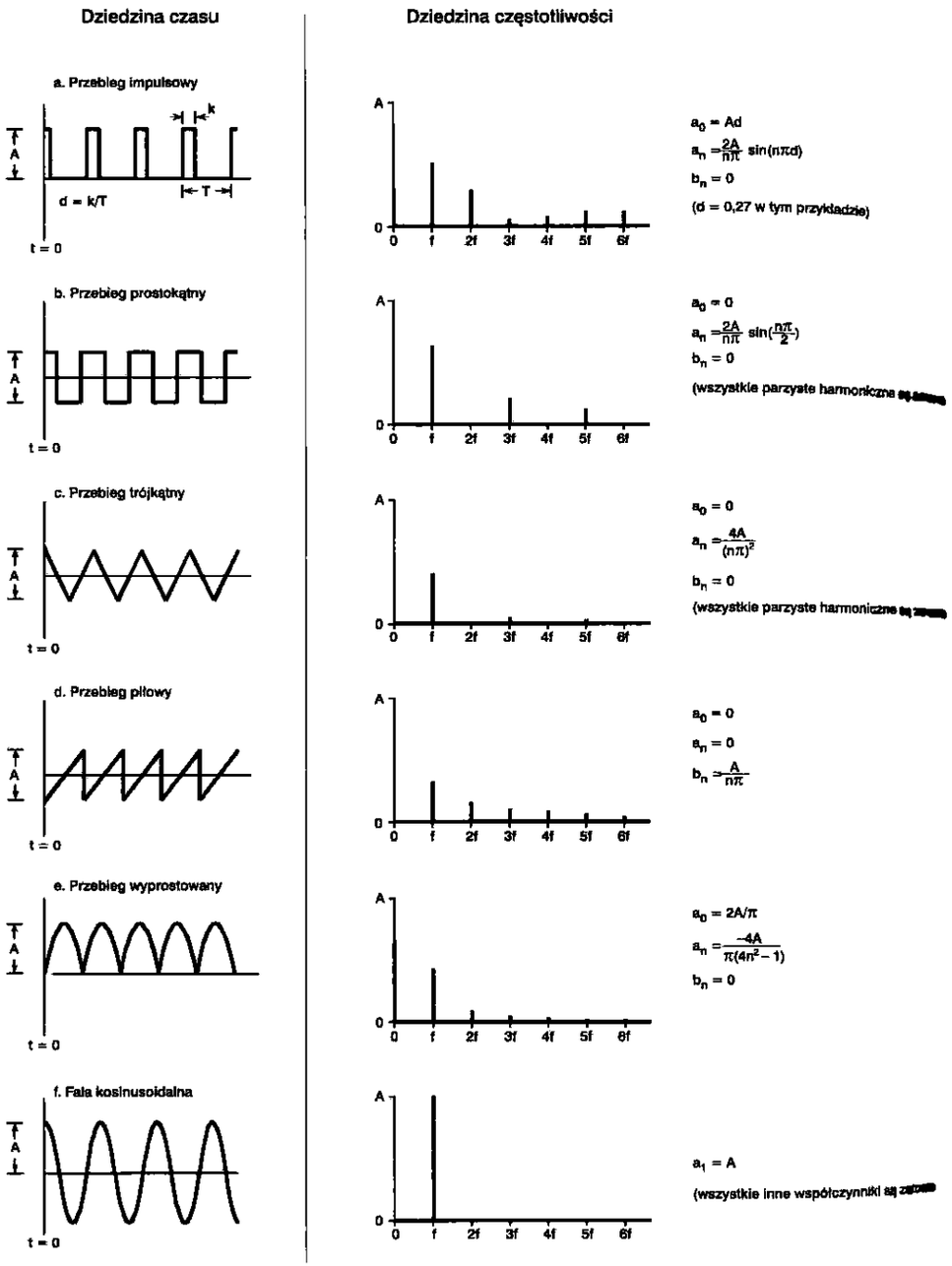
$$b_n=\frac{-2}{T}\int_{-T/2}^{T/2} x(t)\sin\left(\frac{2\pi tn}{T}\right)dt$$

Wzór 13.5. Szereg Fouriera, równania analizy. W tych wzorach x(t) jest sygnałem w dziedzinie czasu podlegającym rozkładowi, a_0 jest składową stałą, współczynniki a_n i b_n zawierają amplitudy fal kosinusoidalnych i sinusoidalnych, a T jest okresem sygnału, czyli odwrotnością częstotliwości podstawowej

Na rys. 13.11 podano przykład obliczania szeregu Fouriera z wykorzystaniem powyższych równań analizy. Analizowanym sygnałem w dziedzinie czasu jest *ciąg impulsów*, czyli fala prostokątna o niejednakowych czasach trwania stanu wysokiego i niskiego. Jeden okres tego przebiegu, od $-T/2$ do $T/2$, można opisać wzorami:

$$x(t)=A \quad dla\ -k/2\le t\le k/2$$

$$x(t)=0 \quad \text{we wszystkich innych przypadkach}$$



Rys. 13.10. Przykłady szeregów Fouriera. Pokazano sześć często spotykanych przebiegów w dziedzinie czasu oraz ich widma częstotliwościowe. Podano też wzory do obliczania współczynników a i b

Współczynnik wypełnienia tego przebiegu (procent czasu, w jakim przebieg jest w stanie wysokim) jest równy $d = k/T$. Współczynniki szeregu Fouriera obliczamy ze wzorów 13.5. Najpierw wyznaczamy składową stałą:

$$a_0 = \frac{1}{T}\int_{-T/2}^{T/2} x(t)\,dt \quad \text{(rozpoczęcie od wzoru 13.5)}$$

$$a_0 = \frac{1}{T}\int_{-k/2}^{k/2} A\,dt \quad \text{(wstawienie wartości sygnału)}$$

$$a_0 = \frac{Ak}{T} \quad \text{(wyznaczenie całki)}$$

$$a_0 = Ad \quad \text{(podstawienie d = k/T)}$$

Uzyskany wynik można by określić intuicyjnie; składowa stała jest po prostu średnią wartością sygnału. Podobne obliczenia prowadzą do wyznaczenia wartości współczynników a:

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos\left(\frac{2\pi t n}{T}\right)dt \quad \text{(rozpoczęcie od wzoru 13.4)}$$

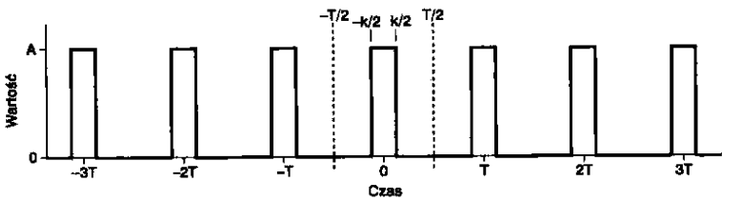
$$a_n = \frac{2}{T}\int_{-k/2}^{k/2} A\cos\left(\frac{2\pi t n}{T}\right)dt \quad \text{(wstawienie wartości sygnału)}$$

$$a_n = \frac{2A}{T}\left[\frac{T}{2\pi n}\sin\left(\frac{2\pi t n}{T}\right)\right]_{-k/2}^{k/2} \quad \text{(wyznaczenie całki)}$$

$$a_n = \frac{2A}{n\pi}\sin(\pi n d) \quad \text{(obliczenie wyniku)}$$

W ten sam sposób są obliczane współczynniki b; okazuje się jednak, że wszystkie mają wartość *zero*. Tak więc ten przebieg można odtworzyć, stosując tylko kosinusoidy, bez konieczności stosowania sinusoid.

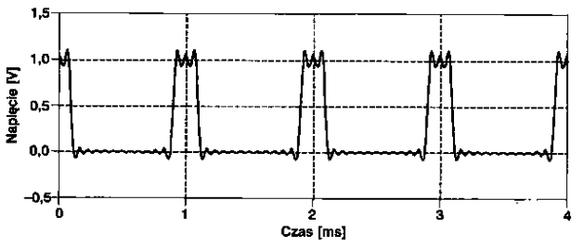
Współczynniki a i b zmienią się, gdy przebieg w dziedzinie czasu zostanie przesunięty w lewo lub w prawo. Na przykład współczynniki b w tym przypadku będą zerowe *tylko* wtedy, gdy jeden z impulsów ma środek symetrii w punkcie $t = 0$. Rozważmy to w następujący sposób. Jeśli przebieg jest *parzysty* (tzn. symetryczny względem $t = 0$), to jest on złożony wyłącznie z sinusoid *parzystych*, czyli z fal kosinusoidalnych. Z tego wynika, że wszystkie współczynniki b są wówczas równe zeru. Jeśli zaś przebieg jest *nieparzysty* (tj. symetryczny względem $t = 0$, lecz z odwróceniem znaku), to składa się on tylko z sinusoid *nieparzystych*, czyli z fal sinusoidalnych. To powoduje, że wtedy wszystkie współczynniki b są równe zeru. Gdy wykonamy konwersję współczynników na współrzędne biegunowe (np. na współczynniki M_n i θ_n), to przy przesunięciu w dziedzinie czasu moduł pozostanie niezmieniony, do fazy zaś będzie dodana składowa liniowa.



Rys. 13.11. Przykład obliczania szeregu Fouriera. Mamy tu ciąg impulsów o współczynniku wypełnienia d = k/T. Współczynniki szeregu Fouriera są obliczane przez korelację przebiegu z falami sinusoidalnymi i kosinusoidalnymi w jednym pełnym okresie. W tym przykładzie wybrano okres od –T/2 do T/2

Jako uzupełnienie tego przykładu wyobraźmy sobie ciąg impulsów występujący w jakimś układzie elektronicznym, mający częstotliwość 1 kHz, amplitudę 1 V i współczynnik wypełnienia 0,2. W tablicy na rys. 13.12 podano amplitudy wszystkich harmonicznych zawartych w tym przebiegu. Na tym rysunku przedstawiono też syntezę przebiegu z wykorzystaniem tylko *pierwszych czternastu* harmonicznych. Nawet z taką liczbą harmonicznych odtworzenie nie jest zbyt dobre. W języku matematycznym mówi się w takim przypadku, że szereg Fouriera jest bardzo *wolno zbieżny*. Jest to po prostu inny sposób stwierdzenia, że ostre zbocza przebiegu w dziedzinie czasu są przyczyną pojawienia się bardzo dużych częstotliwości w widmie. Zwróćcie też uwagę na przerzuty na ostrych zboczach, czyli tzw. efekt Gibbsa, o którym była mowa w rozdziale 11.

Ważnym zastosowaniem szeregu Fouriera jest elektroniczne **mnożenie częstotliwości**. Przypuśćmy, że chcemy utworzyć bardzo stabilny przebieg sinusoidalny o częstotliwości 150 MHz. Może to być potrzebne np. w nadajniku radowym pracującym z tą częstotliwością. Wysoka stabilność wymaga zastosowania generatora *kwarcowego*. Częstotliwość takiego generatora jest stabilizowana przez rezonujący kryształ kwarcowy będący elementem układu. Trudność polega na tym, że kryształy kwarcowe pracują tylko do ok. 10 MHz. Rozwiązaniem jest zbudowanie generatora stabilizowanego kwarcem o częstotliwości z zakresu między 1 a 10 MHz i następnie pomnożenie tej częstotliwości w celu uzyskania żądanej wartości. Tę operację realizuje się, *zniekształcając* przebieg sinusoidalny, np. przez obcinanie wierzchołków przy użyciu diod lub przepuszczenie przebiegu przez układ podnoszący do kwadratu. Następnie z tego zniekształconego przebiegu filtrem pasmowym wybiera się harmoniczne. Dzięki temu można częstotliwość podwoić, potroić lub pomnożyć przez jeszcze większe liczby całkowite. Najczęściej, w celu uzyskania potrzebnej częstotliwości, zamiast jednego stopnia mnożącego stosuje się kilka sekwencyjnych stopni podwajaczy i potrajaczy. W tego rodzaju układach szereg Fouriera ma istotne znaczenie, gdyż umożliwia wyznaczenie *amplitudy* tych sygnałów o mnożonej częstotliwości w zależności od rodzaju zniekształcenia i od wyboru harmonicznej.

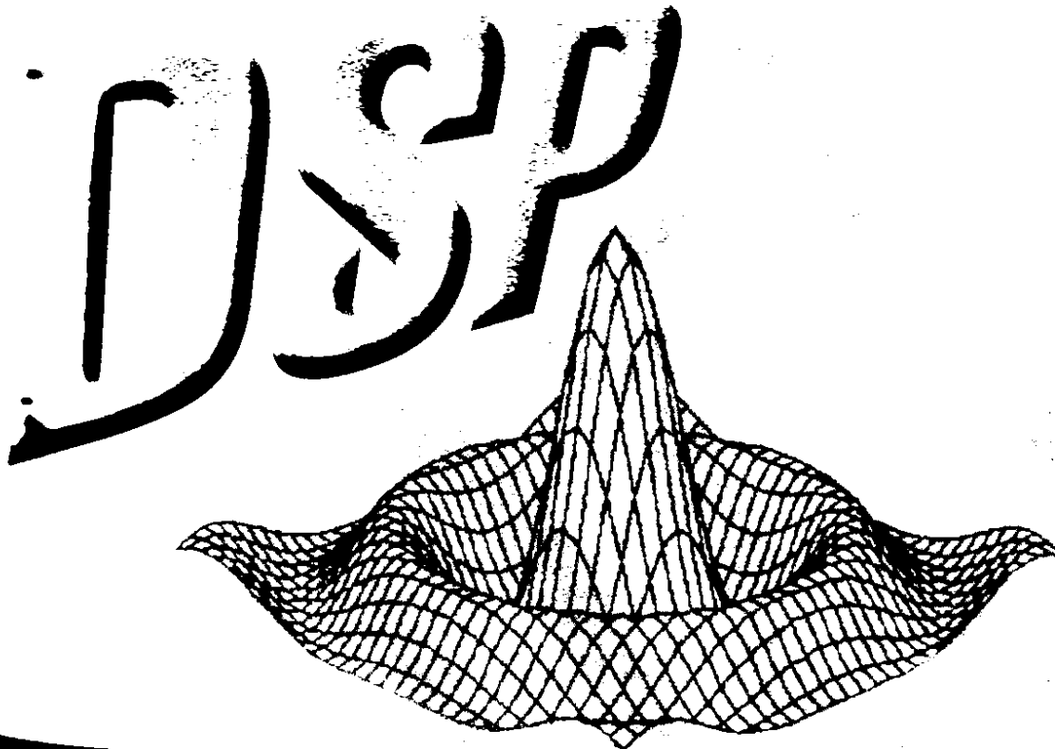


Częstotliwość	Napięcie [V]
DC	0,20000
1kHz	0,37420
2kHz	0,30273
3kHz	0,20182
4kHz	0,09355
5kHz	0,00000
6kHz	-0,06237
7kHz	-0,08648
8kHz	-0,07568
9kHz	-0,04156
10kHz	0,00000
11kHz	0,03402
12kHz	0,05046
.	.
.	.
.	.
123kHz	0,00482
124kHz	0,00302
125kHz	0,00000
126kHz	-0,00297
.	.
.	.
.	.
803kHz	0,00075
804kHz	0,00046
805kHz	0,00000
806kHz	-0,00046

Rys. 13.12. Przykład syntezy przebiegu przy użyciu szeregu Fouriera. Odtwarzany przebieg jest ciągiem impulsów o częstotliwości 1 kHz, amplitudzie 1 V i współczynniku wypełnienia 0,2 (jak to pokazano na rys. 13.11). W tablicy podano amplitudy harmonicznych, a na rysunku przedstawiono przebieg odtworzony z wykorzystaniem tylko pierwszych czternastu harmonicznych

14

Wprowadzenie do filtrów cyfrowych



Dwoma celami stosowania filtrów cyfrowych, ogólnie biorąc, są: rozdzielanie sygnałów wy... jących razem oraz odtwarzanie sygnałów, które zostały w jakiś sposób zniekształcone. Filtry ... gowe (elektroniczne) mogą być stosowane do tych samych celów, jednak przy użyciu filtrów cyf... wych uzyskuje się znacznie lepsze wyniki. W następnych siedmiu rozdziałach opisano najpopu... niejsze filtry cyfrowe i porównano ich właściwości. Ten rozdział ma charakter wstępny. Omówi... w nim parametry, które będą miały istotne znaczenie przy rozpatrywaniu różnych rodzajów filt...

Podstawowe informacje o filtrach

Filtry cyfrowe są bardzo ważnym działem DSP. W istocie, to właśnie ich doskonałe działa... jest jedną z głównych przyczyn, dla których DSP stało się tak popularne. Jak już wspomnia... no, filtry mają dwa zastosowania: *rozdzielanie* sygnałów i ich *odtwarzanie*. Rozdzielanie sy... gnałów jest konieczne, gdy jakiś sygnał został zanieczyszczony zakłóceniami, szumem lub ... innymi sygnałami. Wyobraźmy sobie na przykład urządzenie elektrokardiograficzne (EKG) ... do badań funkcjonowania serca dziecka będącego jeszcze w łonie matki. Podstawowy sygnał ... będzie zakłócony przez oddychanie i tętno matki. Można więc zastosować filtr, aby rozdzie... lić te sygnały i prawidłowo je indywidualnie przeanalizować.

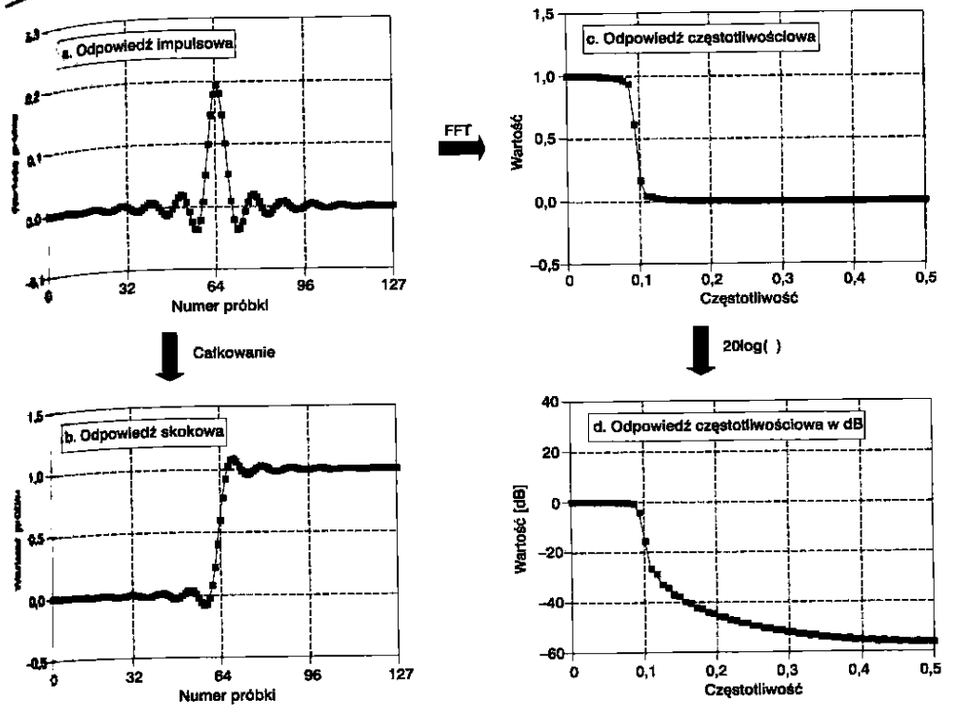
Odtworzenie sygnału stosuje się, gdy został on w jakiś sposób zniekształcony. Na przykład ... nagranie sygnału fonicznego wykonane przy użyciu aparatury słabej jakości można odfil... trować i uzyskać nagranie bardziej zbliżone do dźwięku oryginalnego. Innym przykładem ... jest usuwanie rozmazań w obrazie uzyskanym przy użyciu niewłaściwie zogniskowanych ... soczewek lub trzęsącej się kamery.

Te kłopoty można usuwać, stosując filtry zarówno analogowe, jak i cyfrowe. Które są lepsze? Filtry analogowe są tanie, szybkie i mają szeroki zakres dynamiczny tak amplitudy, jak i częstotliwości. Filtr cyfrowe zaś znacznie górują nad analogowymi doskonałą jakością swego działa... nia. Na przykład cyfrowy filtr dolnoprzepustowy przedstawiony w rozdziale 16 ma wzmocnienie 1 ± 0,0002 w zakresie od 0 do 1000 Hz i wzmocnienie poniżej 0,0002 dla częstotliwości powyżej 1001 Hz. Cały więc ten skok charakterystyki następuje w obszarze tylko 1 Hz. Nie spodziewa... się takich właściwości w układzie ze wzmacniaczem operacyjnym. Filtry cyfrowe mogą działać o *tysiące* razy lepiej niż analogowe. To powoduje zasadniczą różnicę w podejściu do zagadnie... filtracji. Przy projektowaniu filtrów analogowych chodzi przede wszystkim o poradzenie sobie z ograniczeniami elektroniki, takimi jak dokładność i stabilność rezystorów i kondensatorów. Filtry cyfrowe zaś są tak dobre, że samo działanie filtru jest często ignorowane. Nacisk kładzie się na ograniczenia *sygnałów* i zagadnienia *teoretyczne* dotyczące ich przetwarzania.

W DSP przyjmuje się zwykle, że sygnały wejściowe i wyjściowe filtrów są sygnałami *w dziedzinie czasu*. Wynika to z faktu, że sygnały są zazwyczaj tworzone przez próbkowanie w regularnych odstępach *czasu*. Lecz nie jest to jedyny możliwy sposób prowadzenia próbkowania. Drugą często spotykaną metodą jest próbkowanie *przestrzenne*. Wyobraźmy sobie na przykład jednoczesne odczyty z matrycy czujników odkształcenia wmontowanych w odstępach co 1 cm wzdłuż skrzydła samolotu. Jest możliwych jeszcze wiele innych dziedzin próbkowania, jednak najczęściej stosuje się próbkowanie w dziedzinie czasu lub przestrzeni. Gdy spotykacie w DSP pojęcie *dziedzina* czasu, to pamiętajcie, że może się ono odnosić do próbek pobieranych w funkcji czasu albo może być ogólnym odniesieniem do każdej dziedziny, w której próbki są pobierane.

Każdy filtr liniowy (rys. 14.1) charakteryzuje się **odpowiedziami**: **impulsową**, **skokową** i **częstotliwościową**. Każda z tych odpowiedzi zawiera pełną, lecz w różnych formach, informację o filtrze. Jeśli jedna z odpowiedzi jest znana, to dwie inne są też określone i mogą być bezpośrednio obliczone. Wszystkie te trzy charakterystyki są istotne, gdyż opisują, jak filtr będzie reagował w różnych okolicznościach.

Najbardziej bezpośrednim sposobem realizacji filtru cyfrowego jest wykonanie *splotu* sygnału wejściowego z *odpowiedzią impulsową* filtru cyfrowego. Wszystkie możliwe filtry cyfrowe mogą być realizowane w ten sposób. (To powinno być dla was oczywiste. Jeśli nie



Rys. 14.1. Parametry filtru. Każdy filtr liniowy charakteryzuje się odpowiedziami: impulsową, skokową i częstotliwościową. Odpowiedź skokową (rysunek b) można wyznaczyć przez całkowanie dyskretne odpowiedzi impulsowej (rysunek a). Odpowiedź częstotliwościową można określić jako szybką transformatę Fouriera (FFT) odpowiedzi impulsowej, wyrażoną w skali liniowej (rysunek c) lub logarytmicznej w dB (rysunek d)

jest, to znaczy, że nie macie wystarczającego przygotowania do zrozumienia tych rozdziałów o projektowaniu filtrów. Jeszcze raz przestudiujcie wcześniejsze rozdziały o podstawach DSP). Gdy w podany sposób korzysta się z *odpowiedzi impulsowej*, to projektanci filtrów nadają jej czasem specjalną nazwę: **jądro filtru** (*filter kernel*).

Jest jeszcze inny sposób realizacji filtrów cyfrowych, zwany **rekursją**. Podczas realizacji filtru metodą splotu każda próbka jest obliczana przez *ważenie* próbek na wejściu i ich sumowaniu. Filtry rekursywne są rozszerzeniem tej metody, z wykorzystaniem wartości poprzednio obliczonych z *wyjścia*, oprócz punktów pobranych z *wejścia*. Zamiast zastosowania odpowiedzi impulsowej filtru, filtr rekursywny określa się zbiorem **współczynników rekursywnych**. Ta metoda będzie szczegółowo omówiona w rozdziale 19. Na razie ważny jest fakt, że wszystkie filtry liniowe mają odpowiedź impulsową, nawet jeśli z niej nie korzystamy przy realizacji filtru. W celu znalezienia odpowiedzi impulsowej filtru rekursywnego trzeba po prostu wprowadzić impuls i zobaczyć, co się uzyska na wyjściu. Odpowiedzi impulsowe filtrów rekursywnych składają się z sinusoid o amplitudzie zmniejszającej się wykładniczo. Z tego w zasadzie wynika, że te odpowiedzi impulsowe są *nieskończenie długie*. W końcu jednak amplituda zmniejsza się poniżej szumu zaokrągleń systemu i dalsze próbki mogą być pominięte. Ze względu na tę właściwość filtry rekursywne są także nazywane filtrami **o nieskończonej odpowiedzi impulsowej NOI** (*IIR – Infinite Impulse Response*), a filtry realizowane metodą splotu są nazywane filtrami **o skończonej odpowiedzi impulsowej SOI** (*FIR – Finite Impulse Response*).

Jak wiadomo, *odpowiedź impulsowa* jest sygnałem na wyjściu systemu, gdy na jego wejściu jest *impuls jednostkowy*. Podobnie *odpowiedź skokowa* jest sygnałem na wyjściu, gdy na

wyjściu jest *skok jednostkowy*. Ponieważ skok jest całką impulsu, więc odpowiedź skokowa jest całką impulsowej. Z tego wynikają dwa sposoby wyznaczania odpowiedzi skokowej: 1) wprowadzenie przebiegu skokowego na wejście filtru i zobaczenie, co się pojawia na jego wyjściu, lub 2) scałkowanie odpowiedzi impulsowej. Dla matematycznej ścisłości trzeba stwierdzić, że pojęcie *całkowanie* jest używane w odniesieniu do sygnałów ciągłych, podczas gdy *całkowanie dyskretne*, tj. suma bieżąca, odnosi się do sygnałów dyskretnych. Odpowiedź częstotliwościową można określić, wyznaczając transformatę DFT (przy użyciu algorytmu FFT). Będzie to rozważane w dalszej części tego rozdziału. Przebieg odpowiedzi częstotliwości można wykreślać w pionowej skali liniowej (jak na rys. 14.1 c) lub w logarytmicznej (w dB, jak na rys. 14.1 d). W skali liniowej są lepiej widoczne zafalowania i spadki wzmocnienia w paśmie przepustowym, a w skali logarytmicznej – tłumienie w paśmie zaporowym.

Może nie pamiętacie decybeli? Oto krótkie przypomnienie. Jeden **bel** (na cześć Alexandra Grahama Bella) oznacza zmianę mocy o *czynnik 10*. Na przykład układ elektroniczny o wzmocnieniu 3 bele daje sygnał wyjściowy o mocy 10 × 10 × 10 = 1000 razy większej niż moc sygnału wejściowego. **Decybel (dB)** jest dziesiątą częścią bela. Tak więc wartości decybelowe –20 dB, –10 dB, 0 dB, 10 dB i 20 dB oznaczają, odpowiednio, stosunki mocy: 0,01; 0,1; 1; 10 i 100. Tak więc każde *dziesięć* decybeli oznacza, że moc zmienia się o czynnik 10.

I tu pojawia się pewien kłopot – zwykle chcemy posługiwać się wartością *napięcia sygnału*, a nie jego *mocą*. Weźmy na przykład wzmacniacz o wzmocnieniu 20 dB. Oznacza to, z definicji, że moc sygnału wzrasta o czynnik 100. Ponieważ wartość napięcia sygnału jest proporcjonalna do pierwiastka kwadratowego mocy, więc napięciowa wartość sygnału jest 10 razy większa na wyjściu niż na wejściu. Wartość 20 dB więc to czynnik 100 dla mocy, a tylko 10 dla napięcia. Każde *dwadzieścia* decybeli oznacza, że napięcie zmienia się o czynnik 10. Można to wyrazić wzorami:

$$\mathrm{dB} = 10 \log_{10} \frac{P_2}{P_1}$$

$$\mathrm{dB} = 20 \log_{10} \frac{U_2}{U_1}$$

Wzór 14.1. Definicja decybeli. Decybele są sposobem wyrażania stosunku dwóch sygnałów. Stosunek mocy (P_2/P_1) jest opisany innym wzorem niż stosunek wartości napięcia sygnału (U_2/U_1)

Powyższe wzory zawierają logarytm o podstawie 10. Jednak w wielu językach komputerowych jest stosowany tylko logarytm o podstawie *e* (logarytm naturalny, zapisywany jako $\log_e x$ lub $\ln x$). Można zastosować logarytm naturalny, modyfikując powyższe wzory w następujący sposób: $\mathrm{dB} = 4{,}342945 \log_e(P_2/P_1)$ oraz $\mathrm{dB} = 8{,}685890 \log_e(U_2/U_1)$.

Decybele są sposobem wyrażania stosunku dwóch sygnałów, dlatego doskonale nadają się do opisu wzmocnienia systemu, tj. stosunku między sygnałem wyjściowym i wejściowym. Inżynierowie stosują jednak decybele także do określania wartości napięcia (lub mocy) *pojedynczego* sygnału, odnosząc go do pewnej wartości odniesienia. I tak na przykład **dBV** oznacza, że sygnał jest odniesiony do poziomu 1 V wartości skutecznej. Podobnie **dBm** określa, że poziomem odniesienia jest sygnał wytwarzający moc 1 mW na obciążeniu 600 Ω (czyli wartość skuteczna ok. 0,78 V).

Jeśli nie rozumiecie niczego więcej na temat decybeli, to zapamiętajcie dwie rzeczy. Po pierwsze spadek –3 dB oznacza spadek wartości napięcia sygnału do 0,707 (a więc spadek mocy do 0,5). Po drugie, zapamiętajcie następującą tablicę konwersji decybeli na stosunek *wartości napięcia* sygnału:

60 dB = 1000
40 dB = 100
20 dB = 10
0 dB = 1
–20 dB = 0,1
–40 dB = 0,01
–60 dB = 0,001

Jak informacja jest reprezentowana w sygnałach

Najważniejszą sprawą przy rozwiązywaniu każdego problemu DSP jest zrozumienie, w jaki sposób *informacja* jest zawarta w sygnale, którym się posługujemy. Jest tu wiele różnych możliwości, zwłaszcza jeśli chodzi o sygnały wytwarzane sztucznie w urządzeniach technicznych. Rozważmy na przykład wszystkie możliwe sposoby modulacji: AM, FM, jednowstęgowa, kodowo-impulsowa, szerokości impulsu itd. Listę można by jeszcze znacznie wydłużyć. Na szczęście w sygnałach powstających drogą naturalną informacja jest najczęściej zawarta na jeden z dwóch sposobów. Będziemy je nazywać: **informacja reprezentowana w dziedzinie czasu** lub **reprezentowana w dziedzinie częstotliwości**.

Informacja w dziedzinie czasu opisuje, kiedy zaszło dane zdarzenie i jaka jest jego wartość. Wyobraźmy sobie na przykład doświadczenie polegające na badaniu natężenia światła słonecznego. Natężenie jest mierzone i rejestrowane co sekundę. Każda próbka sygnału wskazuje, co nastąpiło w danej chwili i jaka była wartość tego zdarzenia. Jeśli następuje rozbłysk słoneczny, to sygnał bezpośrednio dostarcza informację o czasie, w jakim rozbłysk nastąpił, o jego czasie trwania, jego zmianach w funkcji czasu itd. Każda próbka zawiera informację, którą można interpretować bez odniesienia do żadnej innej próbki. Nawet jeśli mamy tylko jedną próbkę sygnału, to już coś wiemy o mierzonym zjawisku. Jest to najprostszy sposób reprezentacji informacji w sygnale.

Informacja reprezentowana w dziedzinie częstotliwości nie ma tak bezpośredniego charakteru. W naszym Wszechświecie wiele rzeczy charakteryzuje się ruchem okresowym. Na przykład kieliszek do wina wibruje po stuknięciu go paznokciem, wydając dzwoniący dźwięk; wahadło starego zegara kołysze się w lewo i w prawo; gwiazdy i planety obracają się wokół swych osi, a także wokół siebie itd. Mierząc częstotliwość, fazę i amplitudę takiego ruchu okresowego, można często uzyskać informację o systemie wytwarzającym ten ruch. Powiedzmy, że próbkujemy dźwięk pochodzący z wibrującego kieliszka. Podstawowa częstotliwość i harmoniczne okresowej wibracji są związane z masą kieliszka i sprężystością materiału. Pojedyncza próbka, sama w sobie, nie zawiera żadnej informacji o ruchu okresowym, a więc też żadnej informacji o kieliszku. Informacja jest zawarta we *wzajemnych zależnościach* między wieloma punktami w sygnale.

W ten sposób doszliśmy do kwestii ważności odpowiedzi skokowej i częstotliwościowej. *Odpowiedź skokowa* opisuje, w jaki sposób informacja reprezentowana w *dziedzinie czasu* jest modyfikowana przez system. *Odpowiedź częstotliwościowa* pokazuje zaś, jak zmienia się informacja reprezentowana w *dziedzinie częstotliwości*. To rozróżnienie ma zasadnicze znaczenie w projektowaniu filtrów, gdyż nie jest możliwe zoptymalizowanie filtru dla obu tych odpowiedzi. Dobre działanie w dziedzinie czasu powoduje słabe działanie w dziedzinie częstotliwości i vice versa. Jeśli projektujecie filtr do usuwania szumu z sygnału EKG (a więc z informacji reprezentowanej w dziedzinie czasu), to istotnym parametrem jest odpowiedź skokowa, a odpowiedź częstotliwościowa ma niewielkie znaczenie. Jeśli zaś waszym zadaniem będzie zaprojektowanie filtru do aparatu słuchowego (informacja w dziedzinie częstotliwości), to najważniejsza będzie odpowiedź częstotliwościowa, a odpowiedź skokowa zupełnie bez znaczenia. Teraz zastanowimy się, co czyni filtr optymalnym do zastosowań w dziedzinie czasu, a co w dziedzinie częstotliwości.

Parametry w dziedzinie czasu

Może nie być całkiem oczywiste, dlaczego odpowiedź skokowa jest tak ważna w filtrach w dziedzinie czasu. Można by się zastanawiać, dlaczego odpowiedź impulsowa nie jest istotnym parametrem. Zależy to od sposobu rozumienia i przetwarzania informacji w ludzkim umyśle. Pamiętajmy, że wszystkie trzy odpowiedzi, skokowa, impulsowa i częstotliwościowa, zawierają identyczną informację, tylko podaną w inny sposób. Odpowiedź skokowa jest użyteczna w analizie w dziedzinie czasu, gdyż jest zgodna ze sposobem widzenia przez człowieka informacji zawartych w sygnałach.

Przyjmijmy na przykład, że dano wam jakiś sygnał nieznanego pochodzenia i poproszono o jego przeanalizowanie. Pierwszą rzeczą, jaką zrobimy, będzie podział sygnału na obszary o podobnych właściwościach. Nie możecie się od tego powstrzymać, wasz umysł wykona to automatycznie. Niektóre obszary mogą być gładkie, inne mogą zawierać duże piki, inne mogą być zaszumione. Taką segmentację przeprowadza się przez identyfikację punktów rozdzielających obszary. Wtedy właśnie zaczyna być potrzebna funkcja skokowa. Jest ona najczystszym sposobem reprezentowania podziału między dwoma odmiennymi obszarami. Funkcja skokowa może być zaznaczeniem początku, a także końca zdarzenia. Zgodnie z tym wszystko, co jest *na lewo*, różni się w jakiś sposób od tego, co jest *na prawo*. W ten właśnie sposób nasz umysł widzi informację w dziedzinie czasu – jako grupę funkcji skokowych dzielących informację na obszary o zbliżonych właściwościach. Odpowiedź skokowa, z tej lei, jest ważna, gdyż opisuje, jak te linie podziału są modyfikowane przez filtr.

Parametry odpowiedzi skokowej, mające istotne znaczenie w projektowaniu filtrów, przedstawiono na rys. 14.2. Aby móc rozróżnić zdarzenia w sygnale, czas trwania odpowiedzi skokowej musi być krótszy niż odstęp między zdarzeniami. Z tego wynika, że odpowiedź skokowa powinna być możliwie jak *najszybsza* (tak się mówi w żargonie stosowanym w DSP), jak pokazano na rysunkach 14.2 a), b). Najlepszym sposobem określenia **czasu narastania** jest podanie liczby próbek znajdujących się między 10% i 90% poziomu maksymalnego. Dlaczego nie zawsze jest możliwy krótki czas narastania? Jest wiele przyczyn – konieczność zredukowania szumu, ograniczenia nieodłącznie związane z systemem akwizycji danych, unikanie aliasingu itd.

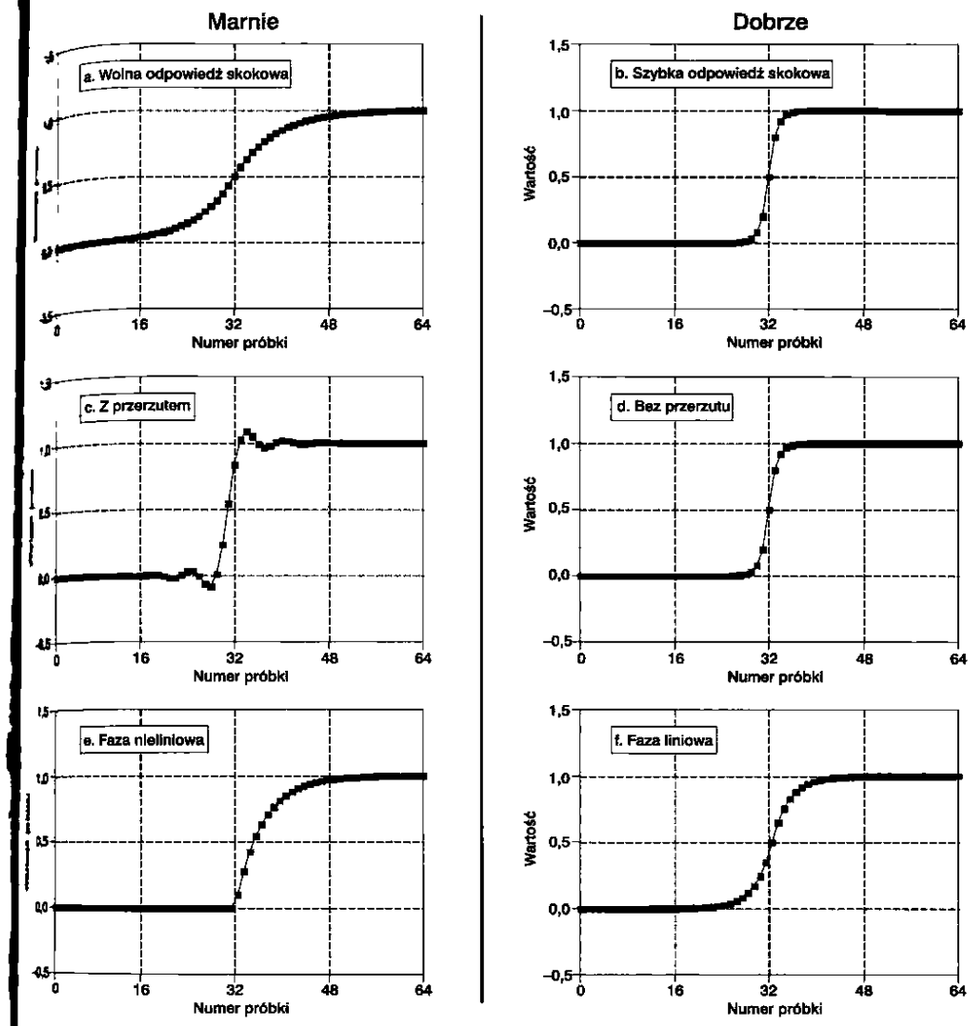
Rysunki 14.2 c), d) ilustrują następny ważny parametr – **przerzut** odpowiedzi skokowej. Ogólnie biorąc, przerzut trzeba eliminować, bo zmienia on amplitudy próbek w sygnale. Jest to podstawowe źródło zniekształceń informacji zawartej w dziedzinie czasu. Trzeba tu jednak zadać pytanie: czy obserwowany przerzut pochodzi z mierzonego sygnału, czy jest spowodowany zastosowanym filtrem?

Na koniec trzeba powiedzieć, że często jest wymagane, aby górna połówka odpowiedzi skokowej była symetryczna z dolną, jak to pokazano na rysunkach 14.2 e), f). Symetria jest konieczna po to, aby *zbocza narastające* były identyczne ze *zboczami opadającymi*. Taka symetria nosi nazwę **liniowości fazy**, gdyż odpowiedź częstotliwościowa ma wtedy charakterystykę fazową będącą linią prostą (omówienie – w rozdziale 19). Upewnijcie się, czy rozumiecie te trzy parametry; są one kluczem do obliczania filtrów w dziedzinie czasu.

Parametry w dziedzinie częstotliwości

Cztery najważniejsze parametry odpowiedzi częstotliwościowej przedstawiono na rys. 14.3. Zadaniem tych filtrów jest przepuszczenie niezmienionych sygnałów w pewnym zakresie częstotliwości przy całkowitym zablokowaniu innych częstotliwości. **Pasmo przepustowe** obejmuje te częstotliwości, które są przepuszczane, **zaporowe** zaś – częstotliwości blokowane. Między nimi jest pasmo przejściowe. Przy **szybkim opadaniu** charakterystyki pasmo przejściowe jest bardzo wąskie. Częstotliwość dzieląca pasma zaporowe i przepustowe jest nazywana **częstotliwością odcięcia** (lub **graniczną**). W filtrach analogowych częstotliwość odcięcia jest zwykle definiowana jako częstotliwość, przy której wartość odpowiedzi spada do 0,707 wartości maksymalnej (w paśmie przepustowym), tj. o –3 dB. Filtry cyfrowe nie są tak znormalizowane; można spotkać definicje częstotliwości odcięcia dla spadku do 99%, 90%, 70,7% i 50% wartości maksymalnej.

Na rys. 14.4 przedstawiono trzy parametry będące miarą jakości działania filtru w dziedzinie częstotliwości. Dla dobrego oddzielenia pasm częstotliwości filtr powinien mieć **szybkie opadanie charakterystyki** w paśmie przejściowym (rysunki 14.4 a, b). Aby sygnał w paśmie przepustowym przechodził przez filtr niezmieniony, nie powinno być **zafalowań** na charakterystyce w tym paśmie (rysunki 14.4 c, d). Wreszcie, dla dobrego blokowania sygnału w paśmie zaporowym, filtr powinien mieć dobry **współczynnik tłumienia** w tym paśmie (rysunki 14.4 e, f).



Rys. 14.2. Parametry określające działanie filtru w dziedzinie czasu. Odpowiedź skokowa jest stosowana do oceny jakości działania filtru w dziedzinie czasu. Ważne są trzy parametry: 1) szybkość odpowiedzi skokowej (czas narastania) (rysunki a), b), 2) przerzut – rysunki c) ,d), 3) liniowość fazy – rysunki e), f)

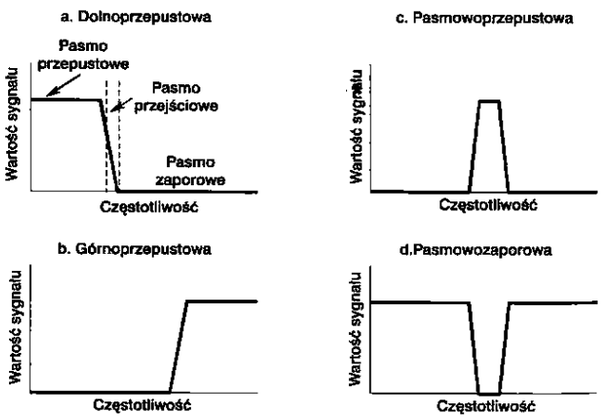
Dlaczego w tych parametrach w ogóle nie ma mowy o fazie? Po pierwsze, faza nie jest istotna w większości zastosowań w dziedzinie częstotliwości. Na przykład faza sygnału fonicznego ma charakter prawie całkowicie losowy i zawiera niewiele użytecznej informacji. Po drugie, jeśli faza ma istotne znaczenie, to bardzo łatwo można zrealizować filtry cyfrowe o doskonałej odpowiedzi fazowej, tj. takiej, żeby wszystkie częstotliwości przechodziły przez filtr z fazą zerową (wrócimy do tej kwestii w rozdziale 19). Natomiast filtry analogowe są pod tym względem bardzo niekorzystne.

W poprzednich rozdziałach rozważano, w jaki sposób stosując DFT, przekształca się odpowiedź impulsową systemu w jego odpowiedź częstotliwościową. Oto krótkie przypomnienie. Najszybszym sposobem obliczenia DFT jest zastosowanie algorytmu FFT omówionego

w rozdziale 12. Rozpoczynając od odpowiedzi impulsowej filtru o długości N próbek, oblicza się – za pomocą FFT – widmo częstotliwościowe składające się z N punktów części *rzeczywistej* i N punktów *części urojonej*. Użyteczną informację zawierają tylko próbki od 0 do $N/2$ części rzeczywistej i urojonej FFT. Pozostałe punkty są duplikatami (częstotliwości ujemne) i można je pominąć. Ponieważ części rzeczywista i urojona są trudne do zrozumienia, więc zwykle są przetwarzane do współrzędnych biegunowych, jak to opisano w rozdziale 8. W ten sposób uzyskuje się moduł i fazę sygnału, obejmujące próbki od 0 do $N/2$ (tzn. $N/2 + 1$ próbek w każdym sygnale). Na przykład odpowiedź impulsowa 256-punktowa da odpowiedź częstotliwościową obejmującą punkty od 0 do 128. Próbka zero reprezentuje składową stałą (DC), czyli częstotliwość zero. Próbka 128 reprezentuje częstotliwość równą połowie częstotliwości próbkowania. Pamiętajmy, że żadne częstotliwości większe od połowy częstotliwości próbkowania nie powinny się pojawiać w spróbkowanych danych.

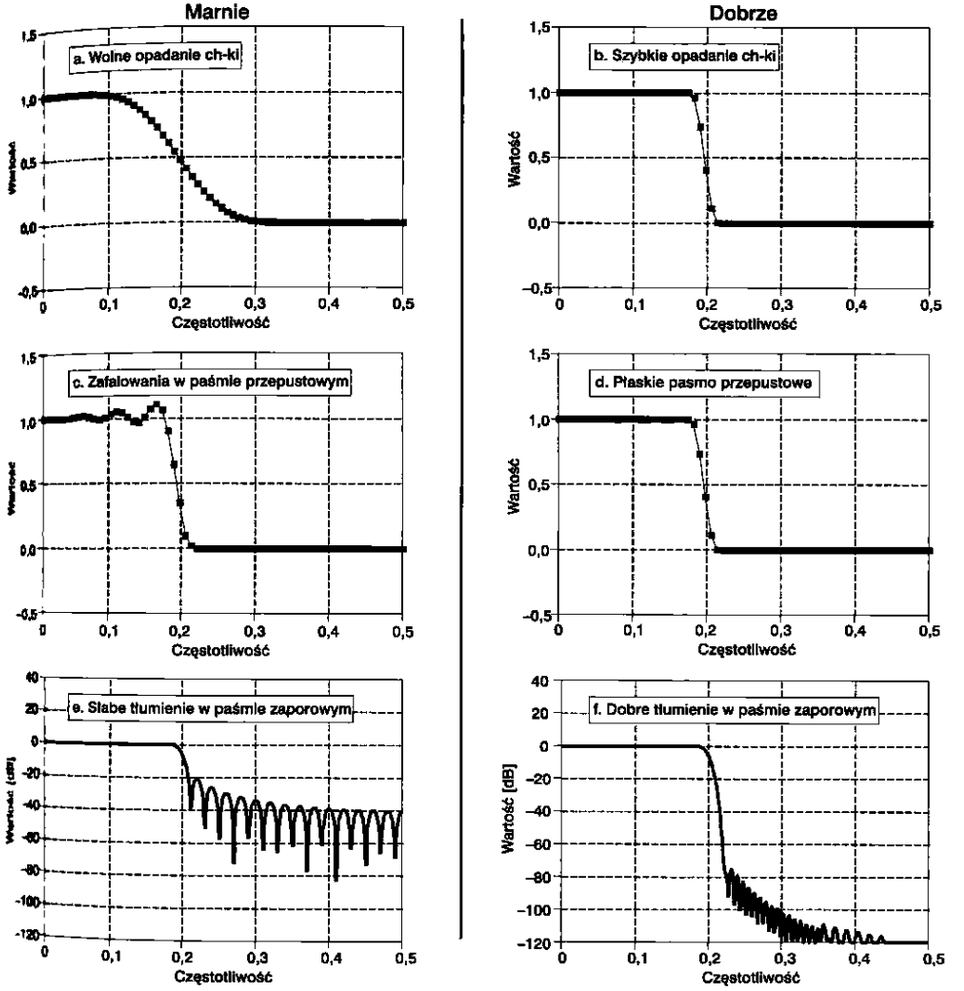
Liczba próbek użyta do reprezentowania odpowiedzi impulsowej może być dowolnie duża. Przyjmijmy na przykład, że chcemy wyznaczyć odpowiedź częstotliwościową filtru o odpowiedzi impulsowej zawierającej 80 punktów. Ponieważ FFT można stosować tylko do sygnałów o długości będącej potęgą dwóch, więc trzeba do sygnału dodać 48 zer, aby uzyskać długość 128 próbek. Takie *dopełnianie zerami* nie zmienia odpowiedzi impulsowej. Aby zrozumieć, dlaczego tak jest, pomyślcie, co dzieje się z tymi dodanymi zerami, gdy sygnał wejściowy jest splatany z odpowiedzią impulsową systemu. Dodane zera po prostu znikają podczas operacji splotu i nie mają wpływu na uzyskany wynik.

Idąc o krok dalej, można dodać *wiele* zer do odpowiedzi impulsowej, aby uzyskała długość, powiedzmy, 256, 512 lub 1024 punktów. Ważne jest to, że dłuższa odpowiedź impulsowa powoduje w odpowiedzi częstotliwościowej gęściejsze rozmieszenie punktów danych. To znaczy, że więcej próbek jest rozmieszczonych między częstotliwościami zerową i równą połowie częstotliwości próbkowania. W przypadku skrajnym, gdy odpowiedź impulsowa jest dopełniona *nieskończoną* liczbą zer, wówczas punkty danych w odpowiedzi częstotliwościowej są nieskończenie blisko siebie, tzn. tworzą linię ciągłą. Inaczej mówiąc, odpowiedź częstotliwościowa filtru jest rzeczywiście sygnałem *ciągłym* od częstotliwości zero do połowy częstotliwości próbkowania. Wynik DFT jest *spróbkowaniem* tej linii ciągłej. Jaką długość odpowiedzi impulsowej powinno się stosować, obliczając odpowiedź częstotliwościową filtru? Na początku spróbujcie $N = 1024$, lecz nie obawiajcie się zmienić tę wartość w razie potrzeby (np. w przypadku niewystarczającej rozdzielczości lub zbyt długiego czasu obliczeń).



Rys. 14.3. Cztery często spotykane odpowiedzi częstotliwościowe. Filtry w dziedzinie częstotliwości są, ogólnie biorąc, stosowane do przepuszczania pewnych częstotliwości (pasmo przepustowe) i blokowania innych (pasmo zaporowe). Najczęściej są stosowane czery rodzaje odpowiedzi: dolnoprzepustowa, górnoprzepustowa, pasmowoprzepustowa i pasmowozaporowa

Miejcie na uwadze, że parametry omawiane w tym rozdziale jako „dobre" lub „złe" to tylko uogólnienia. Wielu sygnałów nie mieści się ściśle w tych kategoriach. Weźmy na przykład sygnał EKG zawierający zakłócenia 50 Hz. Informacja jest zakodowana w *dziedzinie czasu*, lecz wszelkie operacje związane z zakłóceniem lepiej przeprowadzać w *dziedzinie częstotliwości*. W tym zastosowaniu projektowanie filtru polega na przyjęciu wielu kompromisów, które często mogą być sprzeczne z konwencjonalną wiedzą zawartą w tym rozdziale. Pamiętajcie o ogólnej regule zdobywania wiedzy: *Żaden rozdział tej książki nie daje wam licencji na zaprzestanie myślenia.*

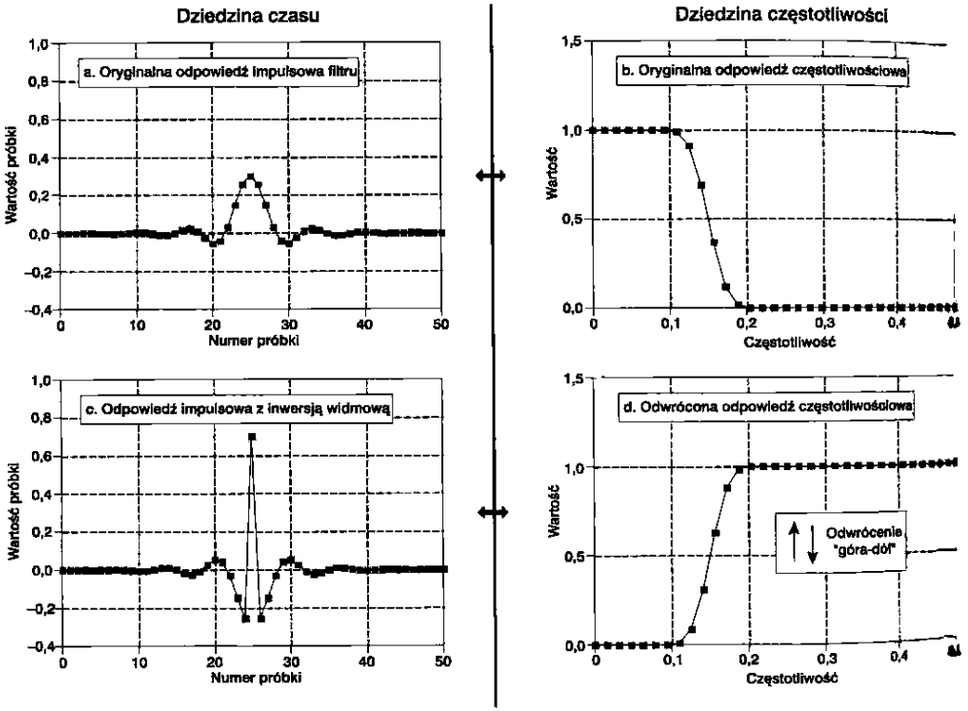


Rys. 14.4. Parametry określające działanie filtru w dziedzinie częstotliwości. Przedstawione odpowiedzi częstotliwościowe dotyczą filtrów dolnoprzepustowych. Ważne są trzy parametry: 1) stromość opadania charakterystyki – rysunki a), b), 2) zafalowania w paśmie przepustowym – rysunki c), d) oraz 3) współczynnik tłumienia w paśmie zaporowym – rysunki e), f)

Filtry górnoprzepustowe, pasmowoprzepustowe i pasmowozaporowe

Filtry górnoprzepustowe, pasmowoprzepustowe i pasmowozaporowe projektuje się, rozpoczynając od filtru dolnoprzepustowego i następnie przekształcając go w taki sposób, aby uzyskać żądaną odpowiedź. Dlatego w większości rozważań o projektowaniu filtrów są podane tylko przykłady dotyczące filtrów dolnoprzepustowych. Są dwie metody przekształcenia filtru dolnoprzepustowego w górnoprzepustowy: **inwersja widmowa** i **odwrotna inwersja widmowa**. Obie są jednakowe użyteczne.

Przykład *inwersji widmowej* przedstawiono na rys. 14.5. Na rys. 14.5 a) pokazano odpowiedź impulsową filtru dolnoprzepustowego zwaną funkcją okienkowanego $sinc(x)$ będącą przedmiotem rozważań w rozdziale 16. Ta odpowiedź impulsowa ma długość 51 punktów, chociaż wiele próbek ma wartości tak małe, że na rysunku wyglądają jak równe zeru. Wynikającą z tej odpowiedzi odpowiedź częstotliwościową pokazano na rys. 14.5 b). Wyznaczono ją, dopełniając odpowiedź impulsową 13 zerami i wykonując 64-punktowe przekształcenie FFT. Aby przemienić odpowiedź impulsową filtru dolnoprzepustowego w odpowiedź filtru górnoprzepustowego, trzeba zrobić dwie rzeczy. Po pierwsze, należy zmienić znak każdej próbki w odpowiedzi impulsowej. Po drugie, trzeba dodać *jeden* do próbki leżącej w środku symetrii. W rezultacie uzyskuje się odpowiedź impulsową filtru górnoprzepustowego (rys. 14.5 c).

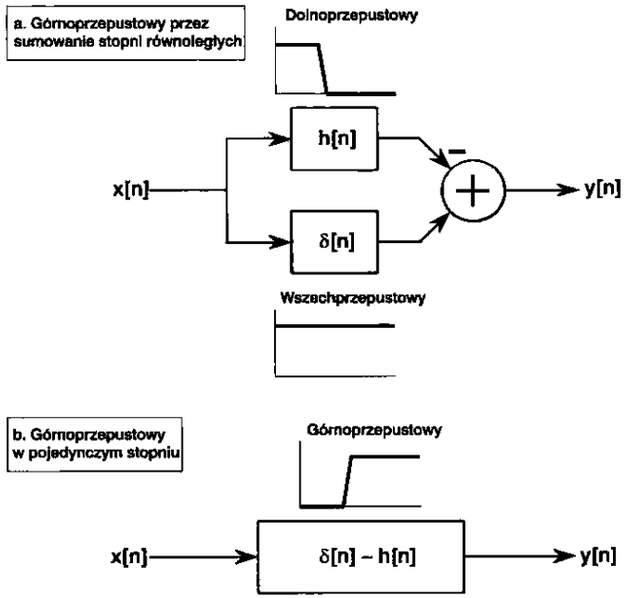


Rys. 14.5. Przykład inwersji widmowej. Odpowiedź impulsowa filtru dolnoprzepustowego (rys. a) ma odpowiedź częstotliwościową pokazaną na rys. b). Odpowiedź impulsową filtru górnoprzepustowego (rys. c) utworzono, zmieniając znak każdej próbki z rys. a) i dodając 1 do próbki leżącej w środku symetrii. Takie działanie w dziedzinie czasu powoduje inwersję widma częstotliwościowego (tj. odwrócenie góra–dół), jak to pokazano w odpowiedzi częstotliwościowej górnoprzepustowej (rys. d)

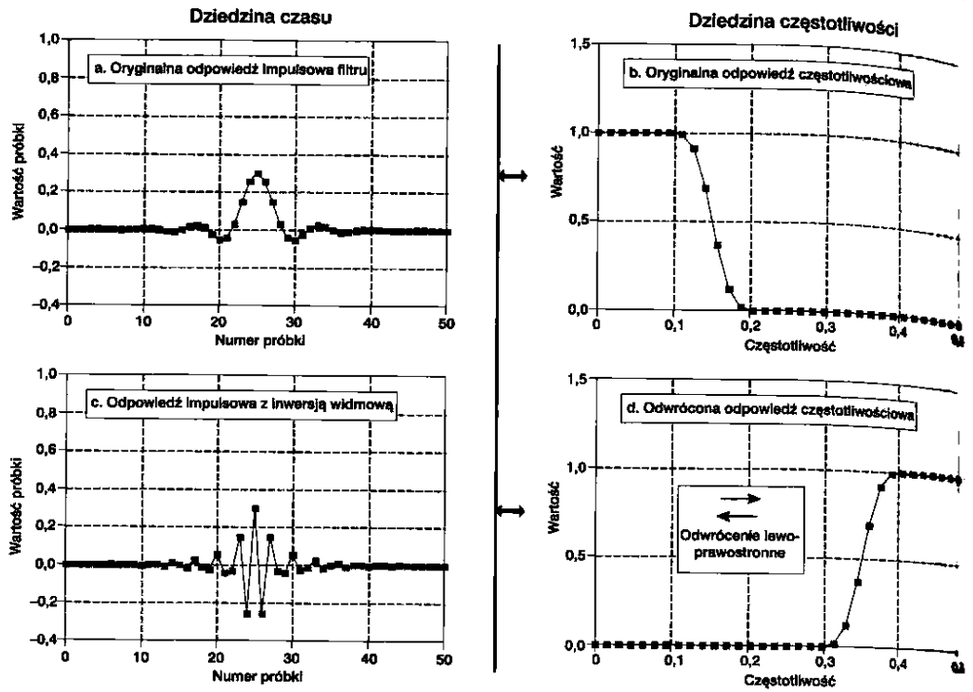
dającą odpowiedź częstotliwościową pokazaną na rys. 14.5 d). Inwersja widmowa powoduje *odwrócenie „góra–dół"* odpowiedzi częstotliwościowej, zmieniając pasma przepustowe na zaporowe i na odwrót. W ten sposób filtr dolnoprzepustowy zamienia się w górnoprzepustowy, górnoprzepustowy w dolnoprzepustowy, pasmowoprzepustowy w pasmowozaporowy oraz pasmowozaporowy w pasmowoprzepustowy.

Na rys. 14.6 przedstawiono, dlaczego taka dwustopniowa modyfikacja w dziedzinie czasu daje w wyniku odwrócenie widma częstotliwościowego. Sygnał wejściowy $x[n]$ – rys. 14.6 a) – jest wprowadzany do dwóch systemów połączonych równolegle. Jeden z systemów jest filtrem dolnoprzepustowym o odpowiedzi impulsowej $h[n]$. Drugi system w ogóle *nie zmienia* sygnału i dlatego ma odpowiedź impulsową będącą funkcją delta $\delta[n]$. Na wyjściu całego systemu uzyskuje się *różnicę* sygnałów wyjściowych systemu wszechprzepustowego i systemu dolnoprzepustowego. Ponieważ składowe o małych częstotliwościach są odejmowane od sygnału oryginalnego, więc na wyjściu pojawiają się tylko składowe o dużych częstotliwościach. Tak więc został utworzony filtr górnoprzepustowy.

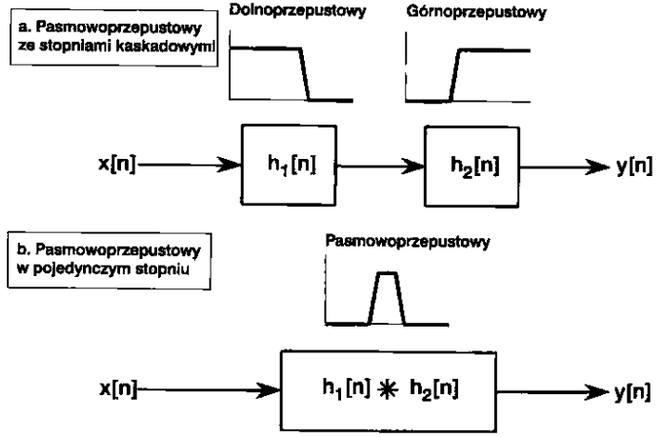
Można to wykonać jako dwuetapową operację w programie komputerowym polegającą na przeprowadzeniu sygnału przez filtr dolnoprzepustowy i następnie odjęciu sygnału odfiltrowanego od oryginalnego. Można jednak całą operację zrealizować w jednym etapie, łącząc ze sobą dwie odpowiedzi impulsowe. Jak podano w rozdziale 7, systemy równoległe o sumowanych wyjściach mogą być łączone w jeden stopień przez sumowanie ich odpowiedzi impulsowych. Odpowiedź impulsowa filtru górnoprzepustowego (rys. 14.6 b) jest opisana wzorem: $\delta[n] - h[n]$. A zatem trzeba zmienić znak wszystkich próbek i potem dodać 1 do próbki będącej w środku symetrii.



Rys. 14.6. Schemat blokowy inwersji widmowej. Na rysunku a) sygnał wejściowy x[n] jest wprowadzany do dwóch, równoległe połączonych systemów o odpowiedziach impulsowych h[n] i δ[n]. Cały system (rysunek b) ma odpowiedź impulsową δ[n] – h[n]. To oznacza, że odpowiedź częstotliwościowa całego systemu jest inwersją odpowiedzi częstotliwościowej h[n]



Rys. 14.7. Przykład odwrotnej inwersji widmowej. Filtr dolnoprzepustowy o odpowiedzi impulsowej pokazanej na rysunku a) ma odpowiedź częstotliwościową przedstawioną na rysunku b). Odpowiedź impulsowa filtru górnoprzepustowego (rysunek c) jest utworzona przez zmianę znaku co drugiej próbki w odpowiedzi z rysunku a). Ta operacja w dziedzinie czasu powoduje w dziedzinie częstotliwości odwrócenie lewo-prawostronne dające w wyniku odpowiedź impulsową filtru górnoprzepustowego (rysunek d)



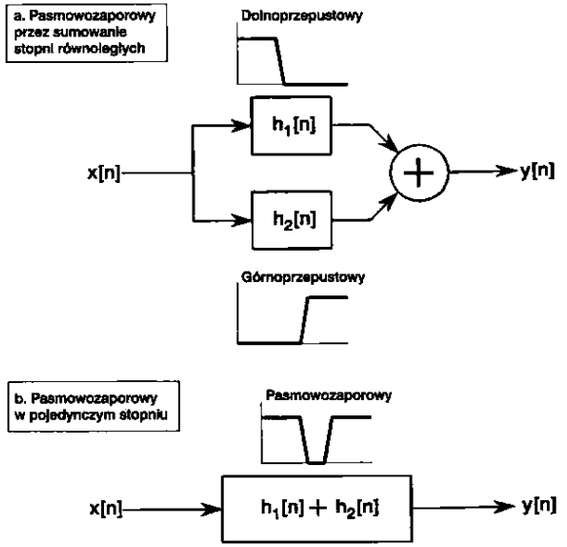
Rys. 14.8. Projektowanie filtru pasmowoprzepustowego. Filtr pasmowoprzepustowy można utworzyć, łącząc kaskadowo filtry dolno- i górnoprzepustowy (rysunek a). Można to zredukować do jednego stopnia, jak pokazano na rysunku b). Odpowiedź impulsowa tego pojedynczego stopnia jest równa splotowi odpowiedzi impulsowych filtrów dolno- i górnoprzepustowego

Aby ta metoda była skuteczna, składowe małoczęstotliwościowe wychodzące z filtru dolnoprzepustowego muszą mieć taką sama fazę, jak składowe małoczęstotliwościowe z systemu wszechprzepustowego. W innym razie nie mogłoby nastąpić pełne odejmowanie. Stąd wynikają dwa ograniczenia metody: 1) oryginalna odpowiedź impulsowa musi się charakteryzować symetrią lewo-prawostronną (tj. fazą zerową lub liniową) oraz 2) impuls musi być dodany w środku symetrii.

Drugą metodą zamiany filtru dolnoprzepustowego na górnoprzepustowy jest *odwrotna inwersja widmowa* (rys. 14.7). Tak jak poprzednio, odpowiedź impulsowa filtru dolnoprzepustowego (rys. 14.7 a) daje odpowiedź częstotliwościową przedstawioną na rys. 14.7 b). Odpowiedź częstotliwościowa filtru górnoprzepustowego (rys. 14.7 c) jest tworzona przez *zmianę znaku co drugiej próbki* na rys. 14.7 a). Jak pokazano na rys. 14.7 d), ta operacja powoduje *lewo-prawostronne* odwrócenie odpowiedzi częstotliwościowej: 0 staje się 0,5 a 0,5 staje się 0. Jeśli na przykład częstotliwość odcięcia filtru dolnoprzepustowego jest 0,15, to uzyskanego górnoprzepustowego będzie 0,35.

Zmiana znaku co drugiej próbki jest równoważna pomnożeniu odpowiedzi impulsowej filtru przez sinusoidę o częstotliwości 0,5. Skutkiem tego, jak stwierdzono w rozdziale 10, jest *przesunięcie* w dziedzinie częstotliwości o 0,5. Spójrzcie na rys. 14.7 b) i wyobraźcie sobie ujemne częstotliwości między –0,5 i 0 będące lustrzanym odbiciem częstotliwości między 0 i 0,5. Częstotliwości pojawiające się na rys. 14.7 d) są ujemnymi częstotliwościami z rys. 14.7 b) przesuniętymi o 0,5.

Na koniec, na rysunkach 14.8 i 14.9 pokazano, jak można łączyć ze sobą odpowiedzi częstotliwościowe filtrów dolno- i górnoprzepustowego, aby uzyskać filtry pasmowoprzepustowy i pasmowozaporowy. Krótko mówiąc, *sumowanie* tych odpowiedzi impulsowych tworzy filtr *pasmowozaporowy*, a ich *splatanie – pasmowoprzepustowy*. Wynika to z zasad łączenia kaskadowego i równoległego podanych w rozdziale 7. Można również stosować kombinacje wielokrotne tych połączeń. Na przykład filtr pasmowoprzepustowy można zaprojektować, tworząc najpierw filtr pamowozaporpwy przez sumowanie dwóch odpowiedzi impulsowych i następnie wykonanie *inwersji widmowej* lub *odwrotnej inwersji widmowej*, jak to opisano poprzednio. Wszystkie te metody funkcjonują bardzo dobrze, bez większych niespodzianek.



Rys. 14.9. Projektowanie filtru pasmowozaporowego. Filtr pasmowozaporowy jest tworzony przez połączenie równoległe filtrów dolno- i górnoprzepustowego, których sygnały wyjściowe są sumowane (rysunek a). Na rysunku b) pokazano tę operację zredukowaną do jednego stopnia. Odpowiedź impulsowa filtru jest wtedy wyznaczana przez sumowanie odpowiedzi impulsowych filtrów dolno- i górnoprzepustowego

Klasyfikacja filtrów

W tablicy 14.1 podano klasyfikację filtrów według ich *zastosowania* i sposobu *realizacji*. Zastosowania filtrów można podzielić na trzy kategorie: zastosowania *w dziedzinie czasu*, *w dziedzinie częstotliwości* oraz *dostosowane do szczególnych wymagań*. Jak już poprzednio powiedziano, filtry w dziedzinie czasu są stosowane wówczas, gdy informacja jest zakodowana w kształcie przebiegu sygnału. Filtracja w dziedzinie czasu jest używana np. do wygładzania przebiegów, usuwania składowej stałej, kształtowania przebiegów itd., a filtry w dziedzinie częstotliwości są stosowane, gdy informacja jest zawarta w amplitudzie, częstotliwości i fazie sinusoid składowych. Celem takich filtrów jest rozdzielenie jednego pasma częstotliwości od innego. Filtry dostosowane do wymagań szczególnych są przeznaczone do funkcji specjalnych, bardziej skomplikowanych niż podstawowe odpowiedzi filtrów (dolno- i górnoprzepustowe, środkowoprzepustowe i środkowozaporowe). W rozdziale 17 opisano na przykład, jak można stosować takie filtry specjalne do rozplotu (*deconvolution*), a więc do uniknięcia niechcianego splotu.

Filtry cyfrowe mogą być realizowane dwoma sposobami – przez *splot* (są też nazywane filtrami *o skończonej odpowiedzi impulsowej*, SOI) lub przez *rekursję* (*o nieskończonej odpowiedzi impulsowej*, NOI). Filtry realizowane przy użyciu splotu mogą mieć znacznie korzystniejsze właściwości od filtrów rekursywnych, lecz działają znacznie wolniej.

W następnych sześciu rozdziałach filtry cyfrowe będą opisane zgodnie z klasyfikacją podaną w tablicy 14.1. Najpierw zajmiemy się filtrami realizowanymi przez splot. Metoda *średniej kroczącej* (rozdz. 15) jest stosowana w dziedzinie czasu, *okienkowanie funkcją sinc* (rozdz. 16) – w dziedzinie częstotliwości, a metoda *SOI* – do filtrów o szczególnych wymaganiach (rozdz. 17). Na zakończenie rozważań nad filtrami przedstawimy w rozdziale 18 metodę zwaną splotem FFT. Jest to algorytm zwiększający szybkość splotu, dzięki czemu filtry SOI mogą działać szybciej.

Następnie przejdziemy do filtrów rekursywnych. Filtr rekursywny *jednobiegunowy* (rozdz. 19) jest stosowany w dziedzinie czasu, filtr *Czebyszewa* (rozdz. 20) zaś – w dziedzinie częstotliwości. Filtry rekursywne o odpowiedzi zgodnej ze szczególnymi wymaganiami są projektowane *metodą iteracyjną*. Dlatego z ich omówieniem powstrzymamy się aż do rozdziału 26, w którym będą opisane procedury iteracyjne innego rodzaju, a mianowicie sieci neuronowe.

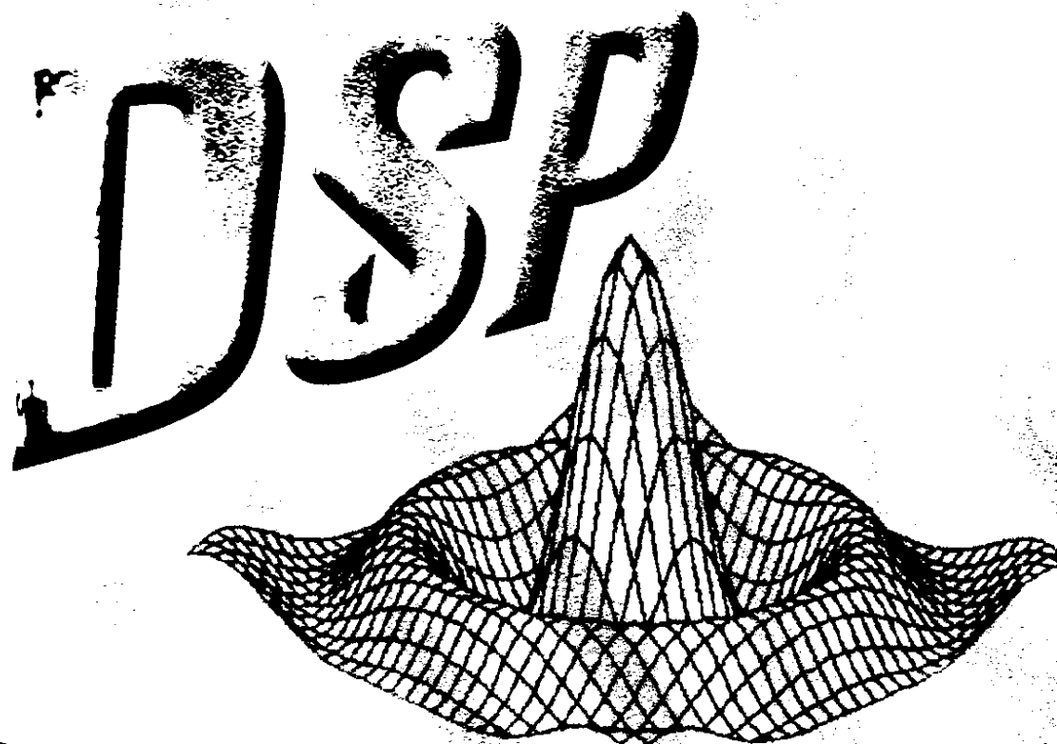
Jak wynika z tablicy 14.1 *splot* i *rekursja* są metodami konkurencyjnymi. W określonym zastosowaniu musimy skorzystać z jednej lub drugiej metody. Którą wybrać? W rozdziale 21 przedstawiono bezpośrednie porównanie obu metod, zarówno w dziedzinie czasu, jak i częstotliwości.

Tab. 14.1. Klasyfikacja filtrów według ich zastosowania i sposobu realizacji

		Splot Skończona odpowiedź impulsowa (SOI)	Rekursja Nieskończona odpowiedź impulsowa (NOI)
Zastosowanie	Dziedzina czasu (wygładzanie, usuwanie składowej stałej)	Średnia krocząca (rozdz. 15)	Jednobiegunowy (rozdz. 19)
	Dziedzina częstotliwości (rozdzielanie częstotliwości)	Okienkowanie funkcją *sinc* (rozdz. 16)	Czebyszewa (rozdz. 20)
	Dostosowane do szczególnych wymagań (rozplot)	SOI – do filtrów o szczególnych wymaganiach (rozdz. 17)	Iteracja (rozdz. 26)

15

Filtry o zasadzie średniej kroczącej



Filtry z wykorzystaniem metody średniej kroczącej są najpopularniejszymi filtrami w DSP przede wszystkim dlatego, że wśród filtrów cyfrowych są one najłatwiejsze do zrozumienia i stosowania. Mimo swej prostoty, filtr o średniej kroczącej jest *optymalny* do często występującego zadania, jakim jest zredukowanie szumu losowego z zachowaniem stromej odpowiedzi skokowej. To powoduje, że do sygnałów zakodowanych w dziedzinie czasu ten filtr jest używany najczęściej. Jednak filtr o średniej kroczącej jest zarazem *najgorszym* filtrem do sygnałów w dziedzinie częstotliwości, gdyż daje małe możliwości rozdzielenia jednego pasma częstotliwości od drugiego. Pokrewne filtrom o średniej kroczącej są tu filtry gaussowski i Blackmana oraz wielokrokowy filtr o średniej kroczącej. Te filtry pracują nieco lepiej w dziedzinie częstotliwości, kosztem dłuższego czasu obliczeń.

Realizacja przez splot

Jak sama nazwa wskazuje, filtr o średniej kroczącej działa na zasadzie uśredniania pewnej liczby punktów sygnału wejściowego w celu uzyskania jednego punktu w sygnale wyjściowym. Tę operację można zapisać wzorem:

$$y[i]=\frac{1}{M}\sum_{j=0}^{M-1}x[i+j]$$

Wzór 15.1. Wzór opisujący działanie filtru o średniej kroczącej. W tym wzorze x[ ] jest sygnałem wejściowym, y[ ] – sygnałem wyjściowym, M – liczbą punktów stosowanych w średniej kroczącej. W tym wzorze są wykorzystywane punkty tylko po jednej stronie właśnie obliczanej próbki wyjściowej

gdzie $x[\ ]$ jest sygnałem wejściowym, $y[\ ]$ – sygnałem wyjściowym, M – liczbą punktów w średniej. Na przykład w 5-punktowym filtrze o średniej kroczącej punkt 80 w sygnale wyjściowym oblicza się następująco:

$$y[80]=\frac{x[80]+x[81]+x[82]+x[83]+x[84]}{5}$$

Alternatywnie grupę punktów z sygnału wejściowego można wybrać *symetrycznie* wokół punktu wyjściowego:

$$y[80]=\frac{x[78]+x[79]+x[80]+x[81]+x[82]}{5}$$

Odpowiada to zmianie sumowania we wzorze 15.1 z: $j = 0$ do $M - 1$ na $j = -(M-1)/2$ do $-(M-1)/2$. Na przykład w 10-punktowym filtrze ze średnią kroczącą indeks j może zmieniać się od 0 do 11 (uśrednianie z jednej strony) lub od –5 do 5 (uśrednianie symetryczne). Przy

Tab. 15.1.

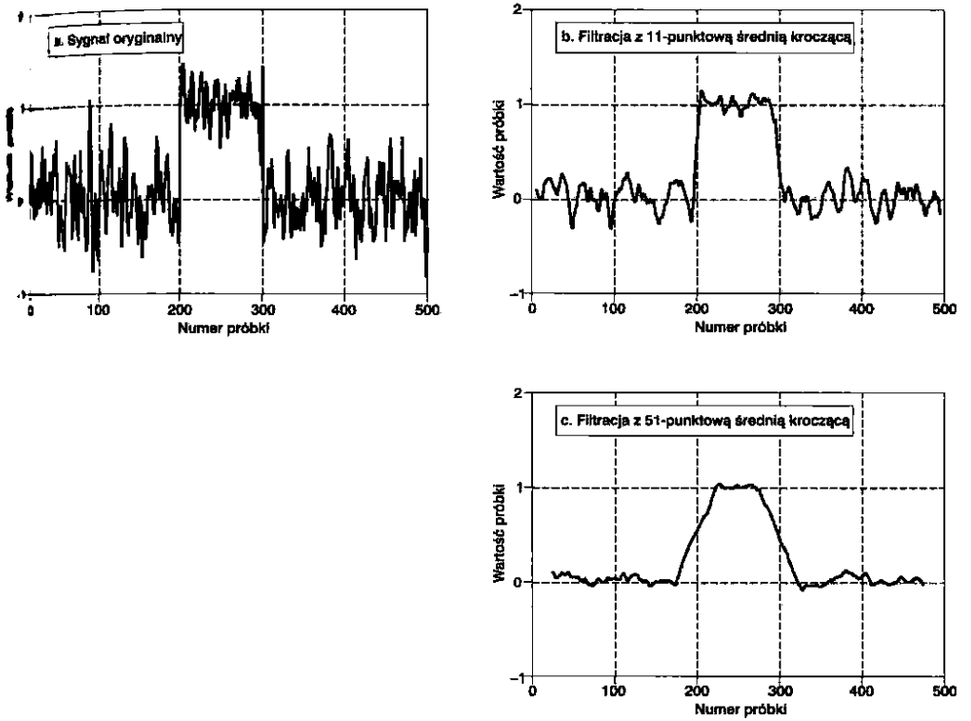
```
100 'FILTR O ZASADZIE ŚREDNIEJ KROCZĄCEJ
110 'Ten program filtruje 5000 próbek 101-punktowym filtrem o  średniej kroczącej,
120 'co daje w wyniku 4900 próbek danych przefiltrowanych.
130 '
140 DIM X[4999]                    'X[ ] przechowuje sygnał wejściowy
150 DIM Y[4999]                    'Y[ ] przechowuje sygnał wyjściowy
160'
170 GOSUB XXXX                     'Podprogram "mityczny" do ładowania X[ ]
180 '
190 FOR I% = 50 TO 4949            'Pętla przez wszystkie punkty sygnału wyjściowego
200   Y[I%] = 0                    'Zeruj, żeby mógł być użyty jako akumulator
210   FOR J% =-50 TO 50            'Oblicz sumowanie
220     Y[I%] = Y[I%] + X(I%+J%]
230   NEXT %
240   Y[I%]=Y[I%]/101              'Zakończ wyznaczanie  średniej przez podzielenie
250 NEXT I%
260 '
270 END
```


uśrednianiu symetrycznym M musi być liczbą *nieparzystą*. Programowanie jest nieco łatwiejsze, gdy punkty znajdują się tylko z jednej strony; jednak powstaje wtedy pewne względne przesunięcie między sygnałami wejściowymi i wyjściowymi.

Trzeba mieć na uwadze, że filtr o średniej kroczącej jest *splotem*, w którym zastosowano bardzo prostą odpowiedź impulsową filtru. Na przykład filtr 5-punktowy ma odpowiedź impulsową: ... 0, 0, 1/5, 1/5, 1/5, 1/5, 1/5, 0, 0 ... A zatem filtr o średniej kroczącej jest splotem sygnału wejściowego z *impulsem prostokątnym* o powierzchni *jeden*. W tablicy 15.1 podano program realizujący filtr o średniej kroczącej.

Tłumienie szumu a odpowiedź skokowa

Wielu naukowców i inżynierów ma poczucie winy, stosując filtr o średniej kroczącej. Ze względu na swoją prostotę, jest on często pierwszym, próbnym rozwiązaniem postawionego problemu. Nawet gdy problem został już w ten sposób całkowicie rozwiązany, ciągle ma się uczucie, że powinno się zrobić jeszcze coś więcej. Jest to raczej śmieszna sytuacja. Filtr o średniej kroczącej jest bowiem nie tylko dobry w wielu aplikacjach, lecz wręcz *optymalny* do tak podstawowego zastosowania, jakim jest stłumienie białego szumu losowego z utrzymaniem jak najbardziej stromej odpowiedzi skokowej.



Rys. 15.1. Przykład filtru o średniej kroczącej. Impuls prostokątny na rys. a) jest ukryty w szumie losowym. Sygnał na rysunkach b), c) jest filtrowany, odpowiednio, 11- i 51-punktowym filtrem o średniej kroczącej. W miarę wzrostu liczby punktów w filtrze szum się zmniejsza, jednak zbocza stają się mniej strome. Filtr o średniej kroczącej jest w przedstawionym problemie rozwiązaniem optymalnym, dającym możliwie najmniejszy szum przy danej stromości zboczy

Na rys. 15.1 podano przykład takiego działania filtru. Sygnał na rys. 15.1 a) jest impuls ukrytym w szumie losowym. Na rysunkach 15.1 a), b) jest widoczne, jak wygładzające dz łanie filtru o średniej kroczącej zmniejsza amplitudę szumu losowego (to dobrze), lecz ró nież pogarsza stromość zboczy impulsu (to źle). Ze wszystkich możliwych filtrów liniowych które mogłyby być użyte, właśnie filtr o średniej kroczącej daje najmniejszy szum przy ok ślonej stromości zboczy. Wielkość tłumienia szumu jest równa pierwiastkowi kwadratowe z liczby punktów podlegających uśrednianiu. Filtr 100-punktowy o średniej kroczącej da na przykład stłumienie szumu o czynnik 10.

Aby zrozumieć dlaczego metoda średniej kroczącej jest najlepszym rozwiązaniem, przy mijmy, że chcemy zaprojektować filtr o ustalonej stromości zboczy. Załóżmy na przykład że ustaliliśmy stromość zboczy w taki sposób, aby zbocze narastające odpowiedzi skok wej zawierało jedenaście punktów. Z tego wynika, że odpowiedź impulsowa filtru powin mieć jedenaście punktów. Trzeba postawić pytanie dotyczące optymalizacji: jak wybra jedenaście wartości w odpowiedzi impulsowej filtru, aby zminimalizować szum w sygnale wyjściowym? Ponieważ szum, który chcemy zredukować, ma charakter losowy, więc żad den punkt na wejściu nie jest szczególny, każdy jest tak zaszumiony, jak punkt sąsiedni Zatem zupełnie bezużyteczne jest preferencyjne traktowanie któregokolwiek z punktów wejściowych przez nadanie mu większego współczynnika w odpowiedzi impulsowej filtru. W dalszej części tego rozdziału wykażemy, że inne filtry są w zasadzie *równie* dobre Żaden jednak nie jest *lepszy* od prostego filtru o średniej kroczącej.

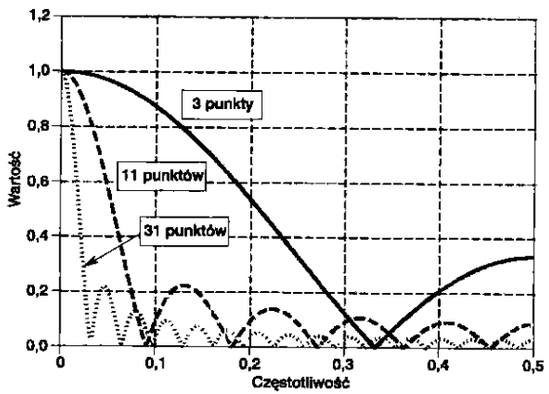
Odpowiedź częstotliwościowa

Na rys. 15.2 przedstawiono odpowiedź częstotliwościową filtru o średniej kroczącej. Jest ona opisana matematycznie transformatą Fouriera impulsu prostokątnego, jak to przedstawiono w rozdziale 11:

$$H[f]=\frac{\sin(\pi f M)}{M\sin(\pi f)}$$

Wzór 15.2. Odpowiedź częstotliwościowa M-punktowego filtru o średniej kroczącej. Częstotliwość f zmienia się od 0 do 0,5. Dla f = 0 trzeba zastosować H[f] = 1

Opadanie charakterystyki jest bardzo wolne, a tłumienie w paśmie zaporowym bardzo marne. Jasne jest, że filtr o średniej kroczącej nie może rozdzielać jednego pasma częstotliwości od

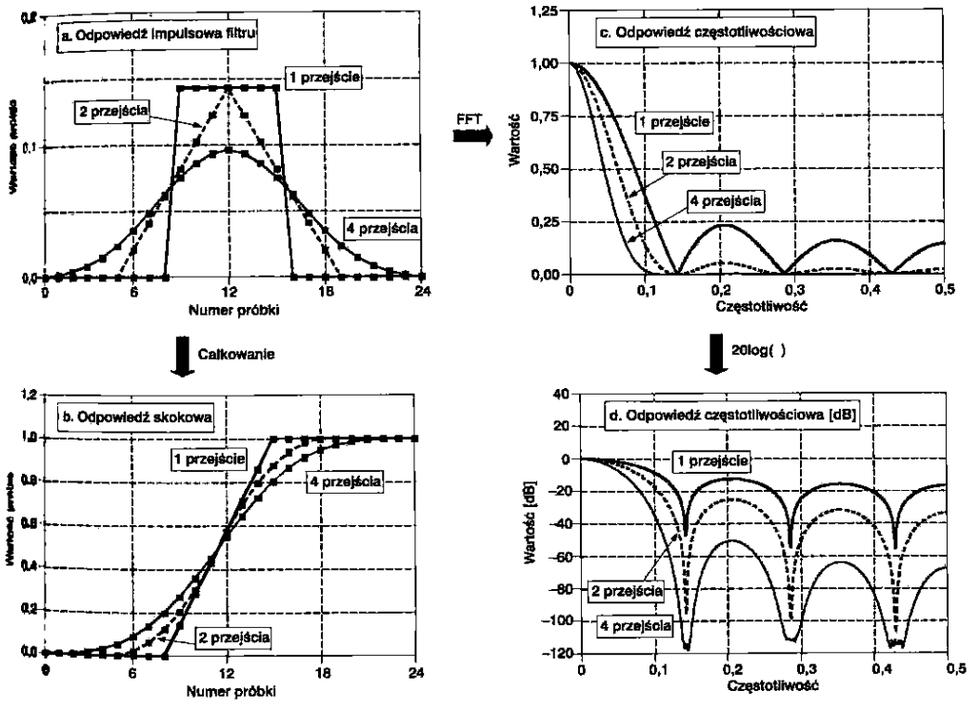


Rys. 15.2. Odpowiedź częstotliwościowa filtru o średniej kroczącej. Ten filtr jest bardzo marnym filtrem dolnoprzepustowym ze względu na wolne opadanie charakterystyki w paśmie przejściowym i słabe tłumienie w paśmie zaporowym. Krzywe na wykresie wyznaczono na podstawie wzoru 15.2

drugiego. Przypomnijcie sobie, że dobre działanie w dziedzinie czasu skutkuje złym działaniem w dziedzinie częstotliwości i vice versa. Podsumowując, trzeba stwierdzić, że filtr o średniej kroczącej jest wyjątkowo dobrym *filtrem wygładzającym* (działanie w dziedzinie czasu), lecz wyjątkowo złym filtrem *dolnoprzepustowym* (działanie w dziedzinie częstotliwości).

Filtry pokrewne filtrom o średniej kroczącej

W świecie idealnym projektanci filtrów mieliby do czynienia tylko z informacjami zakodowanym w dziedzinie czasu *lub* w dziedzinie częstotliwości, lecz nigdy z mieszaniną tych dwóch dziedzin w tym samym sygnale. Niestety istnieją jednak niektóre zastosowania, w których obie dziedziny są jednakowe ważne. Do tej trudnej kategorii należą na przykład sygnały telewizyjne. Informacja wizyjna jest kodowana w dziedzinie czasu, to znaczy, że kształt przebiegu odpowiada rozkładowi jaskrawości w obrazie. Jednak podczas przesyłania sygnału wizyjnego postępuje się z nim zgodnie z jego rozkładem częstotliwościowym. Istotne jest więc całkowite pasmo częstotliwości, sposób sumowania fal nośnych dźwięku i koloru, eliminacja i odtwarzanie składowej stałej itd. Innym przykładem mogą być zakłócenia elektromagnetyczne, które są najlepiej zrozumiałe w dziedzinie częstotliwości, mimo że informacja sygnałowa jest zakodowana w dziedzinie czasu. Na przykład sygnał monitora temperatury w eksperymencie naukowym może być zakłócony sygnałem 50 Hz z sieci energetycznej, 30 kHz z zasilacza impulsowego lub 1320 kHz z lokalnej stacji radiowej AM. Filtry pokrewne filtrom o średniej kroczącej mają lepsze właściwości w dziedzinie częstotliwości i mogą być przydatne w zastosowaniach związanych jednocześnie z obu dziedzinami – czasu i częstotliwości.

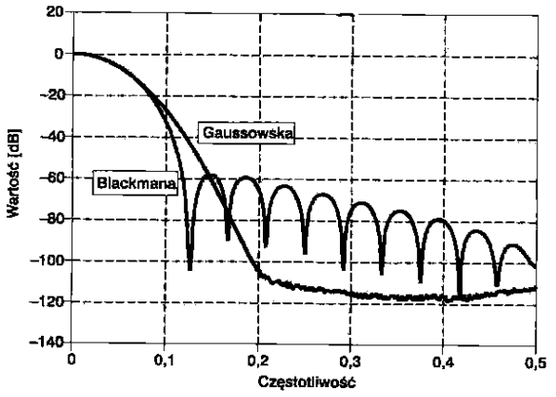


Rys. 15.3. Charakterystyki wielokrokowego filtru o średniej kroczącej. Na rys. a) przedstawiono odpowiedzi impulsowe filtru wynikające z jedno-, dwu- lub czterokrotnego przejścia sygnału przez 7-punktowy filtr o średniej kroczącej. Na rys. b) przedstawiono równoważne odpowiedzi skokowe, a na rys. c), d) – częstotliwościowe

Wielokrokowe filtry o średniej kroczącej działają na zasadzie przechodzenia sygnału wejściowego przez filtr dwa lub więcej razy. Na rys. 15.3 a) przedstawiono całkowitą odpowiedź impulsową filtru wynikającą z jednego, dwóch lub czterech kroków filtracji, czyli przejść sygnału przez filtr. Dwa kroki są równoważne zastosowaniu odpowiedzi impulsowej o kształcie *trójkątnym* (prostokątnej odpowiedzi impulsowej poddanej splotowi z samą sobą). Po czterech lub więcej krokach równoważna odpowiedź impulsowa ma kształt zbliżony do krzywej *gaussowskiej* (przypomnijcie sobie centralne twierdzenie graniczne). Jak pokazano na rys. 15.3 b), wielokrotne przejścia przez filtr dają odpowiedź skokową o kształcie litery „S", podczas gdy pojedyncze przejście daje linię prostą. Odpowiedzi częstotliwościowe (rysunki 15.3 c, d) wynikają z wzoru 15.2 *przemnażanego* przez siebie przy każdym kolejnym przejściu. Każdy splot w dziedzinie czasu powoduje mnożenie widm częstotliwościowych.

Na rys. 15.4 pokazano odpowiedzi częstotliwościowe dwóch innych filtrów pokrewnych filtrowi o średniej kroczącej. Jeśli jako odpowiedź filtru zastosować czystą krzywą Gaussa, to odpowiedź częstotliwościowa też będzie gaussowska, jak to rozważano w rozdziale 11. Krzywa Gaussa jest istotna, gdyż jest to kształt odpowiedzi impulsowych wielu systemów naturalnych i zbudowanych przez człowieka. Na przykład krótki impuls światła wchodzący do długiej światłowodowej linii przesyłowej wychodzi z niej jako impuls o kształcie gaussowskim, co wynika z różnic dróg przebytych przez fotony wewnątrz światłowodu. Gaussowska odpowiedź impulsowa jest też szeroko stosowana w *przetwarzaniu obrazu*, ponieważ jej symetryczne właściwości umożliwiają szybkie splioty dwuwymiarowe (patrz rozdział 24). Druga odpowiedź częstotliwościowa na rys.15.4 wynika z odpowiedzi impulsowej filtru zgodnej z **oknem Blackmana**. (Pojęcie *okno* nie ma tu żadnego znaczenia, jest po prostu częścią powszechnie przyjętej nazwy tej krzywej). Dokładny kształt okna Blackmana opisano w rozdziale 16 (wzór 16.2, rys. 16.2); wygląda ono bardzo podobnie jak krzywa gaussowska.

W jakiej mierze te filtry pokrewne są lepsze od samego filtru o średniej kroczącej? Są lepsze pod trzema względami. Po pierwsze i najważniejsze, te filtry mają *tłumienie w paśmie zaporowym* lepsze niż filtr o średniej kroczącej. Po drugie, odpowiedzi impulsowe tych filtrów są w swojej końcowej części *zbieżne* do mniejszych wartości. Pamiętajmy, że każdy punkt w sygnale wyjściowym jest sumą ważoną grupy próbek wejściowych. Jeśli odpowiedź impulsowa filtru jest zbieżna, to próbkom sygnału wejściowego leżącym dalej nadaje się mniejsze wagi niż próbkom leżącym bliżej. Po trzecie, odpowiedzi skokowe są raczej krzywymi *gładkimi*



Rys. 15.4. Odpowiedź częstotliwościowa okna Blackmana i filtru gaussowskiego. Oba te filtry dają tłumienie w paśmie zaporowym lepsze niż filtr o średniej kroczącej. Nie daje to żadnych korzyści w usuwaniu szumu losowego z sygnałów zakodowanych w dziedzinie czasu, lecz może być pożyteczne w problemach z dziedzin mieszanych. Wadą tych filtrów jest konieczność stosowania splotu, który jest algorytmem bardzo wolnym

a nie liniami prostymi o nagłych zmianach, jak w filtrze o średniej kroczącej. Dwie ostatnie zalety dają na ogół ograniczone korzyści, choć można spotkać zastosowania, gdzie te zalety są rzeczywiście korzystne.

Filtr o średniej kroczącej i filtry mu pokrewne mają bardzo podobne właściwości pod względem tłumienia szumu losowego z zachowaniem stromej odpowiedzi impulsowej. Niejednoznaczność wiąże się tylko ze sposobem pomiaru *czasu narastania* odpowiedzi skokowej. Jeśli czas narastania jest mierzony od 0 do 100% amplitudy skoku, to najlepszy z możliwych jest filtr o średniej kroczącej, jak to już podano poprzednio. Natomiast przy pomiarze od 10 do 90% amplitudy filtr z oknem Blackmana okazuje się *lepszy* niż filtr ze średnią kroczącą. Są to jednak tylko teoretyczne dyskusje; traktujcie ten parametr jako jednakowy w obu filtrach.

Największa różnica między omawianymi filtrami wiąże się z *szybkością działania*. Przy użyciu algorytmu rekursywnego (opisanego dalej) filtr o średniej kroczącej przeleci przez wasz komputer z szybkością błyskawicy. Jest to rzeczywiście *najszybszy* dostępny filtr cyfrowy. Wielokrotne przejścia przez filtr ze średnią kroczącą będą oczywiście odpowiednio wolniejsze, lecz taki filtr jest też bardzo szybki. Natomiast filtry gaussowski i Blackmana są irytująco powolne, gdyż trzeba w nich używać splotu. Chodzi tu o czynnik równy 10-krotności liczby punktów w odpowiedzi impulsowej filtru (gdyż mnożenie jest ok. 10 razy wolniejsze od dodawania). Można więc przewidywać, że 100-punktowy filtr gaussowski będzie 1000 razy wolniejszy niż filtr o średniej kroczącej z wykorzystaniem rekursji.

Realizacja przez rekursję

Niebywałą zaletą filtru o średniej kroczącej jest możliwość jego realizacji bardzo szybkim algorytmem. Aby zrozumieć ten algorytm, wyobraźmy sobie przejście sygnału wejściowego $x[\ ]$ przez 7-punktowy filtr o średniej kroczącej dające w rezultacie sygnał wyjściowy $y[\ ]$. Spójrzmy, jak są obliczane dwa sąsiednie punkty $y[50]$ i $y[51]$:

$$y[50]=x[47]+x[48]+x[49]+x[50]+x[51]+x[52]+x[53]$$
$$y[51]=x[48]+x[49]+x[50]+x[51]+x[52]+x[53]+x[54]$$

Są to prawie identyczne obliczenia. Przy obliczeniu $y[50]$ konieczne jest zsumowanie punktów od $x[48]$ do $x[53]$. Takie samo sumowanie następuje też przy obliczeniu $y[51]$. Jeśli punkt $y[50]$ został już obliczony, to najbardziej *efektywnym* sposobem obliczenia punktu $y[51]$ jest:

$$y[51]=y[50]+x[54]-x[47]$$

Po obliczeniu $y[51]$ na podstawie $y[50]$, można obliczyć $y[52]$ na podstawie $y[51]$ i tak dalej. Gdy w sygnale $y[\ ]$ został już obliczony pierwszy punkt, to wszystkie inne punkty można obliczyć przez pojedyncze dodawanie i odejmowanie dla każdego punktu. Wyraża się to wzorem:

$$y[i]=y[i-1]+x[i+p]-x[i-q]$$
$$\text{gdzie}: p=(M-1)/2$$
$$q=p+1$$

Wzór 15.3. Rekursywna realizacja filtru o średniej kroczącej. W tym wzorze x[ ] jest sygnałem wejściowym, y[ ] – wyjściowym, M – liczbą punktów w średniej kroczącej (liczba nieparzysta). Przed zastosowaniem tego wzoru trzeba pierwszy punkt w sygnale obliczyć przez standardowe sumowanie

Proszę zauważyć, że w tym wzorze do obliczenia każdego punktu sygnału wyjściowego korzysta się z dwóch źródeł danych: punktów z sygnału wejściowego *oraz* poprzednio obliczonych punktów sygnału wyjściowego. Jest to tzw. wzór **rekursywny**, co znaczy, że wynik jednego obliczenia jest wykorzystywany do dalszych obliczeń. (Nazwa „rekursywny" ma

jeszcze inne znaczenie, zwłaszcza w technice komputerowej). W rozdziale 19 bardziej szczegółowo omówiono filtry rekursywne różnych rodzajów. Trzeba zdawać sobie sprawę z tego, że filtr rekursywny ze średnią kroczącą bardzo różni się od typowych filtrów rekursywnych. W szczególności, większość filtrów rekursywnych ma nieskończenie długą odpowiedź impulsową (NOI) złożoną z sinusoid i przebiegów wykładniczych. Odpowiedź impulsowa filtru o średniej kroczącej jest zaś impulsem prostokątnym, a więc jest to skończona odpowiedź impulsowa (SOI).

Ten algorytm jest z kilku powodów szybszy od innych filtrów cyfrowych. Po pierwsze, wymaga tylko dwóch obliczeń na jeden punkt, niezależnie od długości odpowiedzi impulsowej filtru. Po drugie, jedynymi koniecznymi operacjami matematycznymi są dodawanie i odejmowanie, podczas gdy w większości filtrów cyfrowych niezbędne jest czasochłonne mnożenie. Po trzecie, schemat indeksowania jest bardzo prosty. Każdy indeks we wzorze 15.3 jest uzyskiwany przez dodawanie lub odejmowanie stałych o wartościach całkowitych, które mogą być obliczane przed rozpoczęciem filtrowania (tj. p i q). Po czwarte, cały algorytm może być zrealizowany przy użyciu liczb całkowitych. Zależnie od zastosowanych rozwiązań sprzętowych, obliczenia na liczbach całkowitych mogą być więcej niż o rząd wielkości szybsze niż zmiennoprzecinkowe.

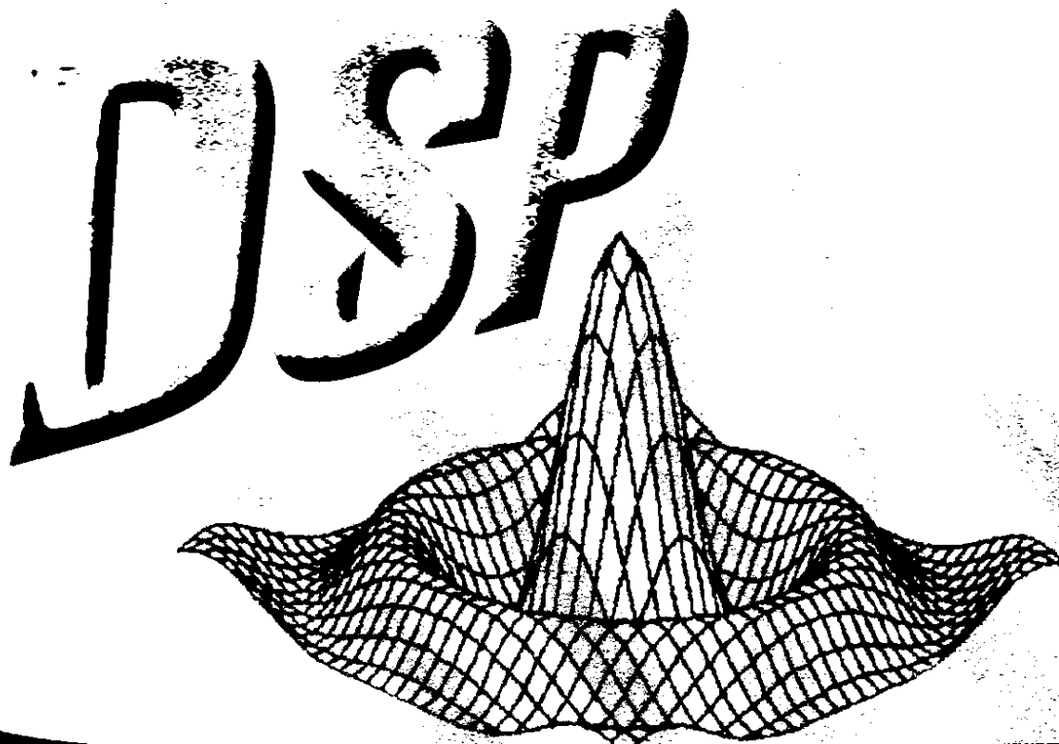
Zadziwiające jest, że w tym algorytmie obliczenia na liczbach całkowitych są nie tylko *szybsze*, lecz także *lepsze* niż przy zapisie zmiennoprzecinkowym. Błędy zaokrąglania w arytmetyce zmiennoprzecinkowej mogą bowiem dawać zupełnie nieoczekiwane wyniki, jeśli nie zachowa się odpowiednich środków ostrożności. Załóżmy, na przykład, że tą metodą jest filtrowany sygnał złożony z 10 000 próbek. Ostatnia próbka w filtrowanym sygnale zawiera skumulowany błąd z 10 000 sumowań i 10 000 odejmowań. To się pojawia na wyjściu jako zmieniające się przesunięcie. W przypadku liczb całkowitych nie ma tej trudności, gdyż w tej arytmetyce nie występują błędy zaokrąglania. Jeśli musicie jednak w tym algorytmie zastosować arytmetykę zmiennoprzecinkową, to można do eliminacji wspomnianego przesunięcia zastosować akumulator o podwójnej dokładności – jak to przedstawiono w programie podanym w tablicy 15.2.

Tab. 15.2

```
100 'FILTR O ŚREDNIEJ KROCZĄCEJ IMPLEMENTOWANY PRZEZ REKURSJĘ
110 'Ten program filtruje 5000 próbek 101-punktowym filtrem o  średniej kroczącej,
120 'co daje w wyniku 4900 próbek danych przefiltrowanych.
130 'Zastosowano akumulator o podwójnej dokładności, aby uniknąć przesunięcia spowodowanego
135 'zaokrąglaniem.
140 '
150 DIM X[4999]                    'X[ ] przechowuje sygnał wejściowy
160 DIM Y[4999]                    'Y[ ] przechowuje sygnał wyjściowy
170 DEFDBL ACC                     'Zdefiniuj ACC jako zmienną o podwójnej dokładności
180 '
190 GOSUB XXXX                     'Podprogram "mityczny" do ładowania X[ ]
200 '
210 ACC = 0                        'Wyznacz Y[50] przez uśrednianie punktów od X[0] do X[100]
220 FOR I% = 0 TO 100
230    ACC = ACC + X[I%]
240 NEXT I%
250 Y[50]=ACC/101
260 '                              'Filtr rekursywny o średniej kroczącej (wzór 15.3)
270 FOR I% = 51 TO 4949
280    ACC = ACC + X[I%+50] - X[I%-51]
290    Y[I%]=ACC/101
300 NEXT I%
310 '
320 END
```

16

Filtry okienkowane funkcją *sinc*



Filtry okienkowane funkcją *sinc* są stosowane do oddzielania jednego pasma częstotliwości od innego. Są one bardzo stabilne, bardzo rzadko powodują kłopotliwe niespodzianki i można w nich osiągnąć parametry o niewiarygodnie wysokim poziomie. Te wyjątkowe właściwości w dziedzinie częstotliwości są uzyskiwane kosztem słabego działania w dziedzinie czasu, m.in. dużych zafalowań i przerzutów w odpowiedzi skokowej. Filtry okienkowane funkcją *sinc*, realizowane przez standardowy splot, są łatwe do programowania, lecz wolne w działaniu. W rozdziale 18 opisano, w jaki sposób można zastosować FFT w celu znacznego polepszenia szybkości obliczeń w tych filtrach.

Sposób realizacji filtrów z okienkowaniem funkcją *sinc*

Na rys. 16.1 przedstawiono zasadę działania filtrów okienkowanych funkcją *sinc*. Na rys. 16.1 b pokazano odpowiedź częstotliwościową *idealnego* filtru dolnoprzepustowego. Wszystkie częstotliwości poniżej częstotliwości odcięcia f_c przechodzą z amplitudą równą jedności, a wyższe częstotliwości są tłumione. Charakterystyka w paśmie przepustowym jest idealnie płaska, tłumienie w paśmie zaporowym – nieskończenie duże, a obszar przejścia między obu pasmami nieskończenie mały.

Wyznaczając odwrotną transformatę Fouriera tej idealnej odpowiedzi częstotliwościowej, uzyskuje się idealną odpowiedź impulsową filtru (rys. 16.1 b). Jak to już poprzednio rozważano (rozdział 11, wzór 11.4) ta krzywa ma postać ogólną $\sin(x)/x$ zwaną **funkcją sinc**, daną wzorem:

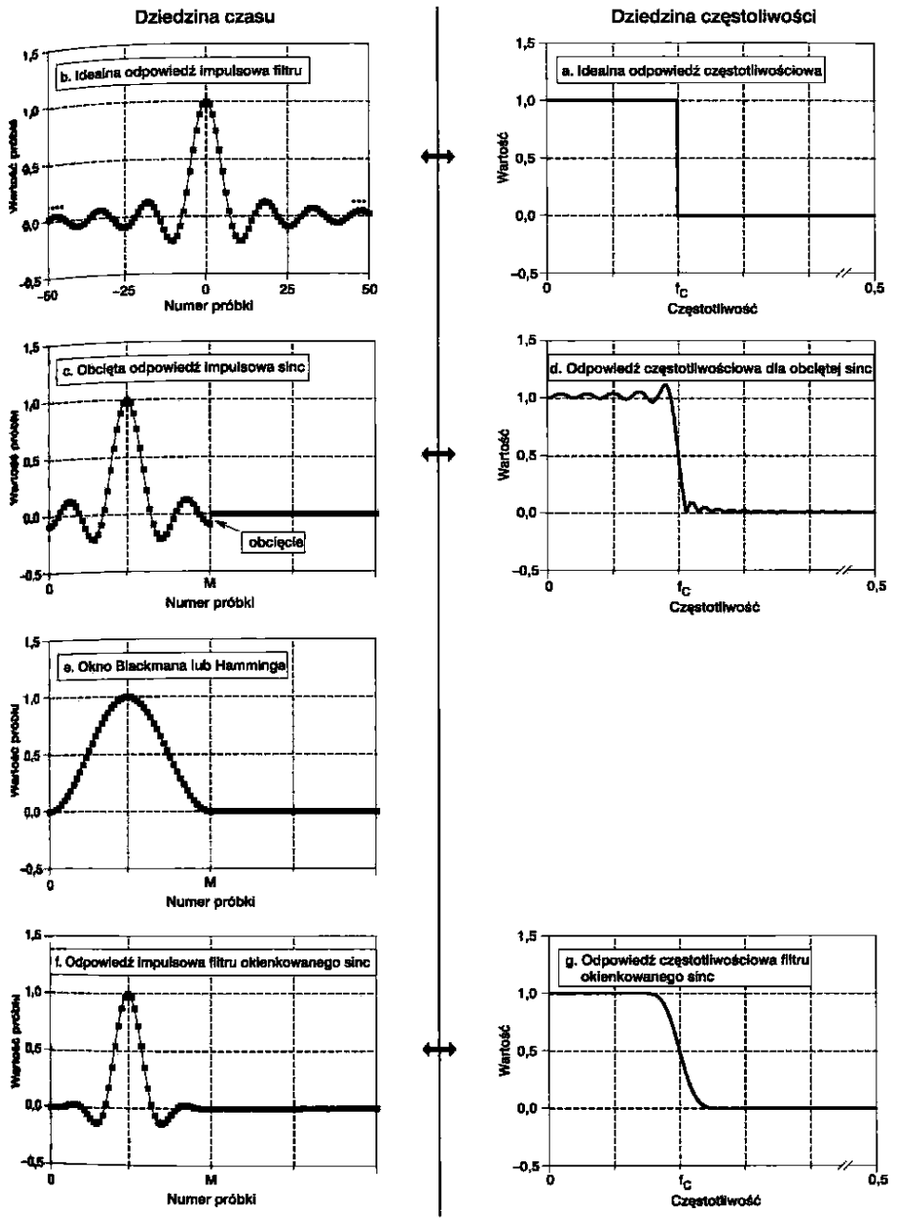
$$h[i]=\frac{\sin(2\pi f_c i)}{i\pi}$$

Splecenie sygnału wejściowego z taką odpowiedzią impulsową filtru tworzy *doskonały* filtr dolnoprzepustowy. Trudność polega na tym, że funkcja *sinc* przebiega od minus do plus nieskończoności bez schodzenia do wartości zero. O ile ta nieskończona długość nie jest problemem pod względem *matematycznym*, to jest przeszkodą dla *komputerów*.

Dla ominięcia tej trudności wprowadzimy dwie modyfikacje do funkcji *sinc* na rys. 16.1 b, uzyskując przebieg pokazany na rys. 16.1 c). Po pierwsze, przebieg jest obcinany do $M+1$ punktów, wybranych symetrycznie wokół głównego listka, przy czym M jest liczbą parzystą. Wszystkie próbki, poza tymi $M+1$ punktami, są ustawione na zero lub po prostu pomijane. Po drugie, cała sekwencja zostaje przesunięta tak, aby przebiegała od 0 do M. Dzięki temu odpowiedź impulsowa filtru może być wyrażona przy użyciu indeksów tylko *dodatnich*. Choć wiele języków programowania dopuszcza indeksy ujemne, to ich stosowanie jest jednak kłopotliwe. Jedynym skutkiem takiego przesunięcia o $M/2$ w odpowiedzi impulsowej filtru jest przesunięcie sygnału wyjściowego o tę samą wartość.

Ponieważ zmodyfikowana odpowiedź impulsowa filtru jest tylko przybliżeniem filtru idealnego, więc filtr nie będzie miał idealnej odpowiedzi częstotliwościowej. W celu wyznaczenia tej odpowiedzi trzeba określić transformatę Fouriera sygnału z rys. 16.1 c). W ten sposób uzyskuje się krzywą przedstawioną na rys. 16.1 d). Co za bałagan! Mamy tu nadmierne zafalowania w paśmie przepustowymi i słabe tłumienie w paśmie zaporowym (przypomnijcie sobie efekt Gibbsa omówiony w rozdziale 11). Te mankamenty są spowodowane gwałtowną nieciągłością na krańcach obciętej funkcji *sinc*. Wydłużanie odpowiedzi impulsowej filtru nie usuwa tych kłopotów; nieciągłość ma istotny wpływ niezależnie od długości M.

Istnieje, na szczęście, prosty sposób poprawy tej sytuacji. Na rys. 16.1 e) pokazano łagodnie zbieżną krzywą zwaną **oknem Blackmana**. Wymnażając obciętą funkcję *sinc* (rys. 16.1 c) przez okno Blackmana (rys. 16.1 e), uzyskuje się odpowiedź impulsową filtru **okienkowanego funkcją sinc** (rys. 16.1 f). Idea tej metody polega na usunięciu gwałtownego opadania na obciętych krańcach odpowiedzi impulsowej i poprawieniu w ten sposób odpowiedzi częstotliwościowej. To ulepszenie jest widoczne na rys. 16.1 g). Charakterystyka jest teraz płaska w paśmie przepustowym, a tłumienie w paśmie zaporowym jest tak dobre, że nie jest możliwe do określenia na podstawie wykresu.



Rys. 16.1. Wyznaczanie odpowiedzi impulsowej filtru okienkowanego funkcją sinc. Odpowiedź częstotliwościową idealnego filtru dolnoprzepustowego przedstawiono na rys. a), a wynikającą z niej odpowiedź impulsową (funkcję sinc) na rys. b). Ponieważ przebieg funkcji sinc jest nieskończenie długi, więc musi być obcięty (rys. c), aby umożliwić jego wykorzystanie w komputerze. Takie obcięcie wywołuje jednak niepożądane zmiany w odpowiedzi częstotliwościowej rys. d). Rozwiązaniem problemu jest przemnożenie obciętego przebiegu sinc przez jakieś okno o gładkim przebiegu rys. e), co daje w rezultacie odpowiedź impulsową filtru okienkowanego funkcją sinc (rys. f). Odpowiedź częstotliwościowa tego filtru (rys. g) jest gładka i korzystna. Przedstawiając przebiegi, nie zachowano skali

Jest możliwość wykorzystania kilku różnych okien, z których większość nosi nazwy pochodzące od nazwisk naukowców, którzy je opracowali w latach 50. Tylko dwa są warte stosowania – **okno Hamminga** i **okno Blackmana**. Są one wyrażone wzorami:

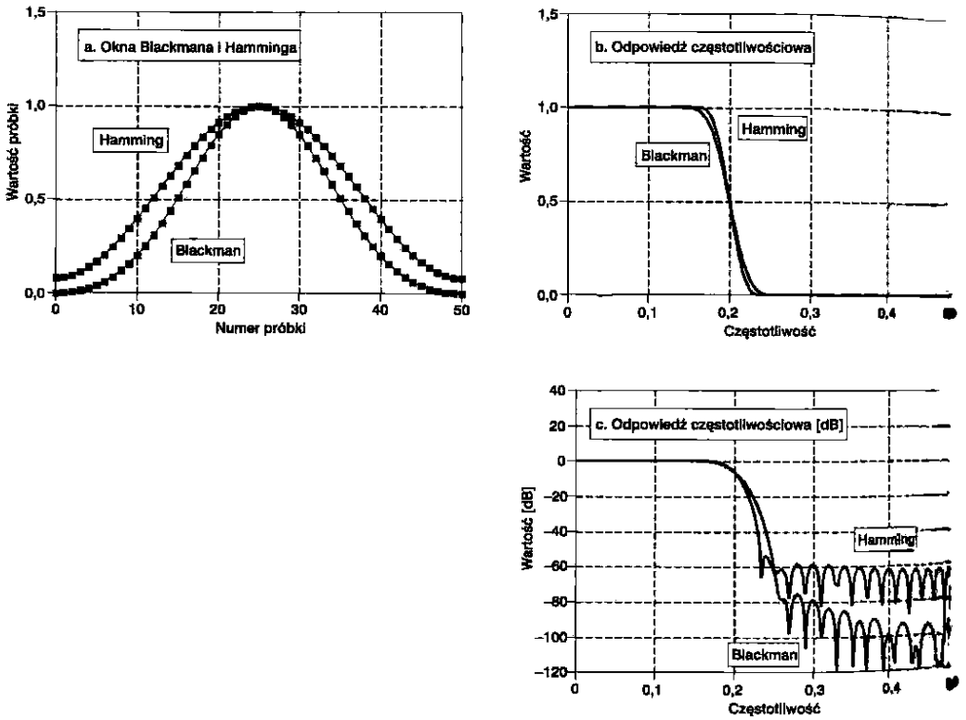
$$w[i]=0,54-0,46\cos\left(2\pi i/M\right)$$

Wzór 16.1. Okno Hamminga. Te okna obejmują zakres od i = 0 do M, całkowita liczba punktów jest M + 1

$$w[i]=0,42-0,5\cos\left(2\pi i/M\right)+0,08\cos\left(4\pi i/M\right)$$

Wzór 16.2. Okno Blackmana

Na rys. 16.2 a) pokazano kształty obu tych okien dla M = 50, tj. dla całkowitej liczby 51 punktów w krzywej. Które z tych okien powinniśmy stosować? Konieczny jest tu pewien kompromis w wyborze parametrów. Jak wynika z rys. 16.2 b), okno Hamminga ma opadanie charakterystyki o ok. 20% szybsze niż okno Blackmana. Jednak przebieg na rys. 16.2 c wskazuje, że okno Blackmana daje lepsze *tłumienie w paśmie zaporowym*. Ściśle mówiąc, tłumienie w filtrze z oknem Blackmana wynosi –74 dB (~0,02%), a z oknem Hamminga tylko –53dB (~0,2%). Chociaż nie widać tego na zamieszczonych wykresach, warto wiedzieć, że filtr z oknem Blackmana ma zafalowania w paśmie przepustowym równe tylko 0,02% podczas gdy Hamminga typowo 0,2%. Ogólnie biorąc, pierwszym wyborem przy projektowaniu powinien być filtr Blackmana, gdyż łatwiej sobie poradzić z wolnym opadaniem charakterystyki niż ze słabym tłumieniem w paśmie zaporowym.



Rys. 16.2. Właściwości okien Blackmana i Hamminga. Kształty obu okien przedstawiono na rys. a), zgodnie z wzorami 16.1 i 16.2. Jak pokazano na rys. b), okno Hamminga daje opadanie charakterystyki o ok. 20% szybsze niż okno Blackmana. Jednak okno Blackmana ma lepsze tłumienie w paśmie zaporowym (Blackman 0,02%, Hamming 0,2%) i mniejsze zafalowania w paśmie przepustowym (Blackman 0,02%, Hamming 0,2%)

Są jeszcze inne okna, o których mogliście słyszeć, choć nie dorównują one oknom Blackmana i Hamminga. **Okno Bartletta** ma kształt trójkątny, zbieżność jest uzyskana przy użyciu linii prostych. **Okno Hanninga**, zwane też oknem **z podwyższonym cosinusem,** jest opisane wzorem: $w[i] = 0,5 - 0,5\cos(2\pi i/M)$. Te dwa okna mają w przybliżeniu taką samą szybkość opadania charakterystyki, jak okno Hamminga, lecz gorsze tłumienie w paśmie zaporowym (Bartlett: –25 dB lub 5,6%, Hanning: –44 dB lub 0,63%). Mogliście też słyszeć o **oknie prostokątnym**. Jest ono tym samym, co *brak* jakiegokolwiek okna. Daje ono po prostu obcinanie ogonów charakterystyki (jak na rys. 16.1 c). Wtedy szybkość opadania jest ok. 2,5 razy większa niż dla okna Blackmana, lecz tłumienie w paśmie zaporowym wynosi tylko –21 dB (8,9%).

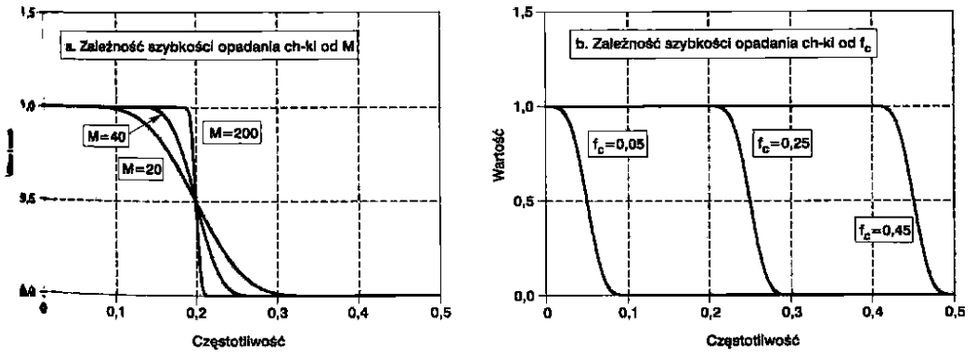
Projektowanie filtru

Przystępując do projektowania filtru okienkowanego funkcją *sinc*, trzeba wybrać dwa parametry: częstotliwość odcięcia f_c oraz długość odpowiedzi impulsowej filtru M. Częstotliwość odcięcia jest wyrażana jako ułamek częstotliwości próbkowania, a więc musi się mieścić w granicach od 0 do 0,5. Wartość M ustala *opadanie charakterystyki* w paśmie przejściowym zgodnie z przybliżonym wzorem:

$$M\approx\frac{4}{BW}$$

Wzór 16.3. Zależność między długością filtru M a szybkością opadania charakterystyki. Długość odpowiedzi impulsowej filtru M określa szerokość pasma przejściowego filtru BW. Jest to tylko wzór przybliżony, gdyż szybkość opadania zależy od zastosowanego okna

gdzie BW jest szerokością pasma przejściowego mierzonego od miejsca, gdzie zaczyna dopiero schodzić poniżej wartości 1 do miejsca, gdzie prawie osiąga zero (powiedzmy – od 99% do 1%). Szerokość pasma przejściowego jest też wyrażana jako ułamek częstotliwości próbkowania i musi się mieścić między 0 i 0,5. Na rys. 16.3 podano przykład zastosowania tego przybliżonego wzoru. Trzy przedstawione krzywe obliczono na podstawie odpowiedzi impulsowych o długościach M = 20, 40 i 200. Ze wzoru 16.3 wynika, że szerokości pasma przejściowego są równe, odpowiednio, BW = 0,2; 0,1 i 0,02. Na rys. 16.3 b) pokazano, że kształt odpowiedzi impulsowej nie zależy od wybranej częstotliwości odcięcia.



Rys. 16.3. Filtr okienkowany funkcją sinc – zależność między długością filtru i szybkością opadania charakterystyki. Jak pokazano na rys. a), dla M = 20, 40 i 200 szerokości pasma przejściowego są odpowiednio równe BW = 0,2; 0,1 i 0,02 częstotliwości próbkowania. Z rys. b) wynika, że kształt odpowiedzi częstotliwościowej nie zmienia się przy różnych częstotliwościach odcięcia. Dla przebiegów na rys. b): M = 60

Czas konieczny do wykonania splotu jest proporcjonalny do długości sygnałów, dlatego wzór 16.3 wyraża kompromisowy wybór między *czasem obliczeń* (zależnym od wartości *M*) i *mością charakterystyki* filtru (reprezentowaną przez *BW*). Na przykład wolniejsze o 20% opadanie charakterystyki okna Blackmana (w porównaniu z oknem Hamminga) skompensować, stosując o 20% dłuższą odpowiedź impulsową filtru. Innymi słowy, na powiedzieć, że realizacja okna Blackmana jest o 20% wolniejsza niż okna Hamminga o równoważnej szybkości opadania charakterystyki. Jest to ważne, gdyż już realizacja filtrów okienkowanych funkcją *sinc* jest bardzo wolna.

Jak pokazano na rys. 16.3 b), częstotliwość odcięcia filtru okienkowanego funkcją *sinc* mierzona na wysokości *połowy amplitudy*. Dlaczego zastosowano tu wartość 0,5 zamiast standardowej 0,707 (–3 dB) używanej w elektronice analogowej i w innych filtrach cyfrowych? Przyczyną jest fakt, że odpowiedź częstotliwościowa filtru okienkowanego funkcją *sinc* charakteryzuje się *symetrią* pasm przepustowego i zaporowego. Na przykład okno Hamminga daje zafalowania w paśmie przepustowym równe 0,2% i *identyczne* tłumienie w paśmie zaporowym (czyli zafalowania w paśmie zaporowym) równe 0,2%. Inne filtry nie wykazują takiej symetrii i nie mają zalety polegającej na stosowaniu punktu połowy amplitudy do wyznaczania częstotliwości odcięcia. Jak będzie o tym mowa w dalszej części tego rozdziału, symetria filtrów okienkowanych funkcją *sinc* powoduje, że są idealne do *inwersji widmowej*.

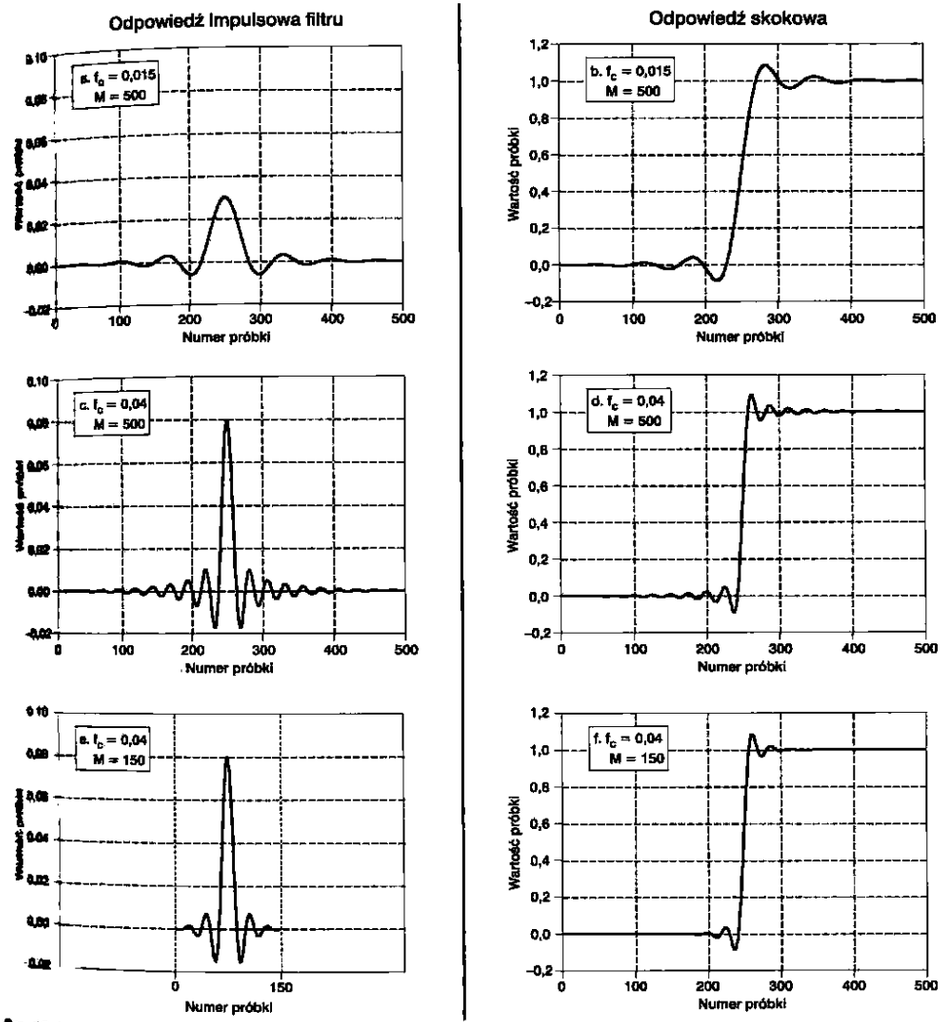
Tab. 16.1

```
100 'FILTR DOLNOPRZEPUSTOWY OKIENKOWANY FUNKCJĄ SINC
110 'Ten program filtruje 5000 próbek 101-punktowym filtrem okienkowanym funkcją sinc,
120 'czego wynikiem jest 4900 próbek danych przefiltrowanych.
130 '
140 DIM X[4999]              'X[ ] przechowuje sygnał wejściowy
150 DIM Y[4999]              'Y[ ] przechowuje sygnał wyjściowy
160 DIM H[100]               'H[ ] przechowuje odpowiedź impulsową filtru
170'
180 PI = 3.14159265
190 FC = .14                 'Ustaw częstotliwość odcięcia (między 0 a 0,5)
200 M% = 100                 'Ustaw długość filtru (101 punktów)
210 '
220 GOSUB XXXX               'Podprogram "mityczny" do ładowania X[ ]
230 '
240 '                        'Oblicz odpowiedź impulsową filtru dolnoprzepustowego ze wzoru 16.4
250 FOR I% = 0 TO 100
260    IF (I%-M%/2) = 0 THEN H[I%] = 2*PI*FC
270    IF (I%-M%/2) <> 0 THEN H[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
280    H[I%] = H[I%] * (0.54 - 0.46*COS(2*PI*I%/M%))
290 NEXT I%
300 '
310 SUM = 0                  'Znormalizuj odpowiedź impulsową filtru dolnoprzepustowego
320 FOR I% = 0 TO 100        'dla wzmocnienia jednostkowego dla DC (częstotliwości zero)
330    SUM = SUM + H[I%]
340 NEXT I%
350 '
360 FOR I% = 0 TO 100
370    H[I%] = H[I%] / SUM
380 NEXT I%
390 '
400 FOR J% = 100 TO 4999     'Wykonaj splot sygnału wejściowego z odpowiedzią impulsową filtru
410    Y[J%]= 0
420    FOR I% = 0 TO 100
430       Y[J%] = Y[J%] + X[J%-I%] * H[I%]
440    NEXT I%
450 NEXT J%
460 '
470 END
```


Po wyborze wartości f_c i M oblicza się odpowiedź impulsową filtru z zależności:

$$h[i]=K\frac{\sin\left(2\pi f_c\left(i-M/2\right)\right)}{i-M/2}\left[0{,}42-0{,}5\cos\left(\frac{2\pi i}{M}\right)+0{,}08\cos\left(\frac{4\pi i}{M}\right)\right]$$

Wzór 16.4. Odpowiedź impulsowa filtru okienkowanego funkcją sinc. Częstotliwość odcięcia f_C jest wyrażona jako ułamek częstotliwości próbkowania o wartości między 0 i 0,5. Długość odpowiedzi impulsowej filtru jest określona liczbą całkowitą M. Numer próbki i jest liczbą całkowitą zmieniającą się od 0 do M, co daje całkowitą liczbę punktów w odpowiedzi impulsowej równą M + 1. Stała K jest tak dobrana, żeby wzmocnienie było równe 1 przy częstotliwości 0. W celu uniknięcia błędu dzielenia przez zero trzeba dla i = M/2 zastosować h[i] = 2πf$_C$K.



Rys. 16.4. Filtr okienkowany funkcją sinc. Przykłady odpowiedzi impulsowych i wynikających z nich odpowiedzi skokowych. Częstotliwość oscylacji sinusoidalnych jest w przybliżeniu równa częstotliwości odcięcia f_C, M jest długością odpowiedzi impulsowej

Nie obawiajcie się tego wzoru! Na podstawie dotychczasowych rozważań powinniście być w stanie rozróżnić trzy składniki: *funkcję sinc*, *przesunięcie M/2* oraz *okno Blackmana*. Chcąc uzyskać wzmocnienie filtru równe jedności przy częstotliwości zero, trzeba tak dobrać stałą *K*, aby suma wszystkich próbek była równa jedności. W praktyce można zignorować stałą *K* podczas obliczania odpowiedzi impulsowej i potem *znormalizować* wszystkie próbki według potrzeby. W programie podanym w tablicy 16.1 pokazano, jak to zrobić. Proszę też zauważyć, jak prowadzone są obliczenia pośrodku funkcji *sinc*, przy $i = M/2$ obejmując dzielenie przez zero.

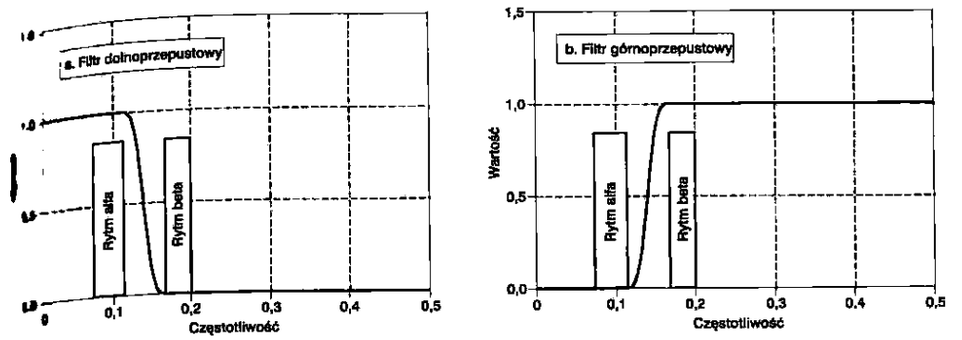
Wzór 16.4, choć długi, jest łatwy w użyciu. Po prostu wprowadźcie go do komputera i nie żecie o nim zapomnieć. Niech komputer prowadzi obliczenia. Bardzo źle byście postąpili próbując robić obliczenia według tego wzoru samodzielnie.

Sprecyzujmy dokładniej, gdzie odpowiedź impulsowa według wzoru 16.4 jest umieszczana w matrycy w waszym komputerze. Jako przykład wybierzmy $M = 100$. Pamiętajmy, że M musi być liczbą parzystą. Pierwszy punkt odpowiedzi impulsowej ma lokalizację 0 w matrycy, a ostatni – lokalizację 100. Wynika stąd, że cały sygnał ma długość 101 punktów. Środek symetrii jest w punkcie 50, tj. $M/2$. 50 punktów na lewo od punktu numer 50 jest symetrycznych w stosunku do 50 punktów leżących na prawo. Punkt 0 ma więc taką samą wartość jak punkt 100, a punkt 49 jak punkt 51. Jeśli trzeba mieć jakąś określoną liczbę punktów w odpowiedzi impulsowej, np. w celu wykonania FFT, to nie starczy po prostu dodać zera do jednego lub drugiego końca. Na przykład mając $M = 100$ można utworzyć próbki od 101 do 127, równe zero, uzyskując odpowiedź impulsową o długości 128 punktów.

Na rys. 16.4 podano przykłady odpowiedzi impulsowych filtrów okienkowanych funkcją *sinc* oraz wynikających z nich odpowiedzi skokowych. Próbki na początku i na końcu odpowiedzi impulsowych są tak małe, że nie można ich nawet zobaczyć na wykresach. Nie popełnijcie błędu, myśląc, że są one nieważne! Te próbki są małe pod względem wartości, lecz wspólnie mogą mieć znaczny wpływ na działanie filtru. Między innymi dlatego w realizacji filtrów okienkowanych funkcją *sinc* stosuje się zapis zmiennoprzecinkowy. Zapis w liczbach całkowitych zwykle nie ma wystarczającego zakresu dynamicznego, aby uchwycić wielkie zmiany wartości występujące w odpowiedzi impulsowej filtru. Jak filtr okienkowany funkcją *sinc* działa w dziedzinie czasu? Okropnie! Jego odpowiedź skokowa ma przerzut i zafalowania; *nie jest to* filtr odpowiedni do sygnałów o informacji zakodowanej w dziedzinie czasu.

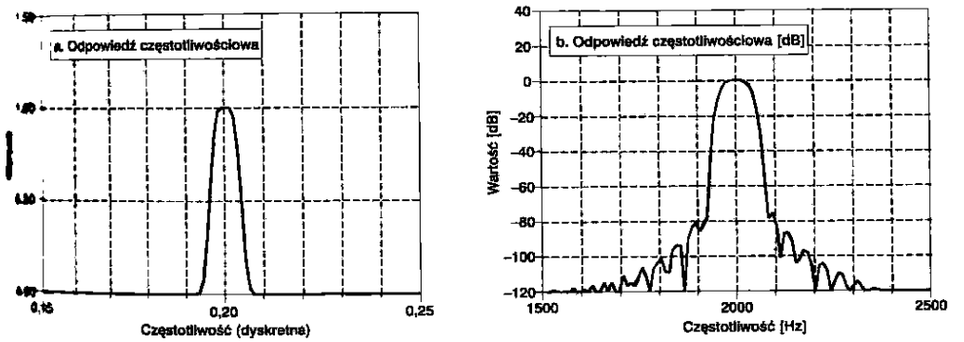
Przykłady filtrów okienkowanych funkcją *sinc*

Elektroencefalogram (EEG) jest wynikiem pomiaru potencjałów prądów czynnościowych mózgu. Są one wykrywane jako miliwoltowe sygnały na elektrodach przymocowanych do powierzchni głowy. Każda komórka nerwowa w mózgu wytwarza małe impulsy elektryczne. Elektroencefalogram jest łącznym wynikiem wielkiej liczby tych impulsów generowanych w pewien (miejmy nadzieję) skoordynowany sposób. Chociaż zależność między myśleniem a tą elektryczną koordynacją jest bardzo mało zrozumiała, to jednak różne częstotliwości w EEG mogą być identyfikowane z określonymi stanami psychicznymi. Kiedy zamkniesz oczy i wypoczywasz, dominującym przebiegiem w EEG będzie wolna oscylacja o częstotliwości między 7 i 12 Hz. Ten przebieg jest nazywany *rytmem alfa* i wiąże się z uczuciem zadowolenia i zmniejszonym poziomem uwagi. Otwarcie oczu i rozglądanie się wokół powoduje zmianę przebiegu EEG i pojawienie się w nim *rytmu beta* o częstotliwości od 7 do 20 Hz. Inne przebiegi i częstotliwości obserwuje się u dzieci, inne we śnie – zależnie od głębokości snu, a jeszcze inne w przypadkach nieprawidłowej pracy mózgu, takich jak padaczka.



Rys. 16.5. Przykład filtrów okienkowanych funkcją sinc. Rytmy alfa i beta w sygnale EEG są rozdzielane filtrami dolnoprzepustowym o długości M = 100. Program realizujący filtr dolnoprzepustowy podano w tablicy 16.1. Dla filtru górnoprzepustowego program jest identyczny, lecz konieczna jest inwersja widmowa odpowiedzi impulsowej filtru dolnoprzepustowego

W tym przykładzie założymy, że sygnał EEG został wzmocniony elektronicznymi układami analogowymi i następnie przetworzony na sygnał cyfrowy z częstotliwością próbkowania 100 próbek/s. Zbieranie danych w czasie 50 s tworzy sygnał zawierający 5000 punktów. Naszym celem jest rozdzielenie rytmu alfa od beta. Do realizacji tego zadania zaprojektujemy dolnoprzepustowy filtr cyfrowy o częstotliwości odcięcia 14 Hz, czyli równej 0,14 częstotliwości próbkowania. Obszar przejścia między pasmami ma mieć szerokość 4 Hz, czyli 0,04 częstotliwości próbkowania. Jak wynika z wzoru 16.3, odpowiedź impulsowa filtru powinna mieć długość ok. 101 punktów. W sposób dowolny wybieramy użycie okna Hamminga. W programie podanym w tablicy 16.1 pokazano, jak realizować taki filtr. Na rys. 16.5 przedstawiono odpowiedź częstotliwościową tego filtru, uzyskaną przez wyznaczenie transformaty Fouriera odpowiedzi impulsowej.



Rys. 16.6. Przykład pasmowoprzepustowego filtru okienkowanego funkcją sinc. Filtr zaprojektowano dla częstotliwości próbkowania 10 kHz. Odniesiona do sygnału analogowego, częstotliwość środkowa filtru pasmowoprzepustowego ma równe 2 kHz, pasmo przepustowe 80 Hz i pasma przejściowe 50 Hz. Zastosowano odpowiedź impulsową filtru okienkowanego funkcją sinc o długości 801 punktów, żeby uzyskać żądaną szybkość opadania charakterystyki. Użyto okna Blackmana w celu osiągnięcia dobrego tłumienia w paśmie zaporowym. Na rys. a) przedstawiono uzyskaną charakterystykę częstotliwościową w skali liniowej, a na rys b) – w decybelach. Oś odciętych na rys. a) jest wyskalowana w ułamkach częstotliwości próbkowania, a na rys. b) w częstotliwości sygnału analogowego przed przetworzeniem go na cyfrowy

Tab. 16.2

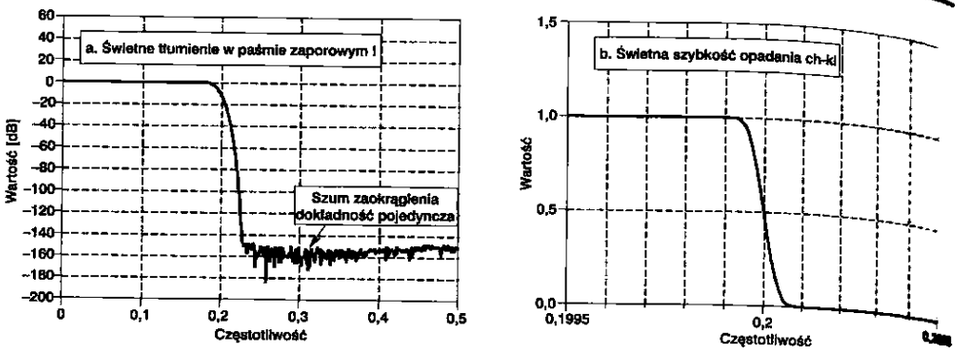
```
100 'FILTR PasmowoPRZEPUSTOWY OKIENKOWANY FUNKCJĄ SINC
110 'Ten program oblicza 801-punktową odpowiedź impulsową odpowiedź filtru pasmowoprzepu
120 '
130 DIM A[800]                      'A[ ] obszar roboczy dla dolnej częstotliwości odcięcia
140 DIM B[800]                      'B[ ] obszar roboczy dla górnej częstotliwości odcięcia
150 DIM H[800]                      'H[ ] przechowuje uzyskaną odpowiedź impulsową filtru
160 '
170 PI = 3.1415926
180 M% = 800                        'Ustaw długość odpowiedzi impulsowej filtru (801 punktów)
190 '
200 '                               'Oblicz pierwszą odpowiedź impulsową filtru z wzoru 16.4
210 FC = 0.196                      'o częstotliwości odcięcia 0.196, zapamiętaj w A[ ]
220 FOR I% = 0 TO 800
230    IF (I%-M%/2) = 0 THEN A[I%] = 2*PI*FC
240    IF (I%-M%/2) <> 0 THEN A[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
250    A[I%] = A[I%] * (0.42 - 0.5*COS(2*PI*I%/M%) + 0.08*COS(4*PI*I%/M%))
260 NEXT I%
270 '
280 SUM = 0                         'Znormalizuj pierwszą odpowiedź impulsową filtru dla wzmo
290 FOR I% = 0 TO 800               'jednostkowego dla DC (częstotliwości zero)
300    SUM = SUM + A[I%]
310 NEXT I%
320 '
330 FOR I% = 0 TO 800
340    A[I%] = A[I%] / SUM
350 NEXT I%
360 '                               'Oblicz drugą odpowiedź impulsową filtru z wzoru 16.4
370 FC = 0.204                      'o częstotliwości odcięcia 0.204, zapamiętaj w B[ ]
380 FOR I% = 0 TO 800
390    IF (I%-M%/2) = 0 THEN B[I%] = 2*PI*FC
400    IF (I%-M%/2) <> 0 THEN B[I%] = SIN(2*PI*FC * (I%-M%/2)) / (I%-M%/2)
410    B[I%] = B[I%] * (0.42 - 0.5*COS(2*PI*I%/M%) + 0.08*COS(4*PI*I%/M%))
420 NEXT I%
430 '
440 SUM = 0                         'Znormalizuj drugą odpowiedź impulsową filtru dla wzmocnienia
450 FOR I% = 0 TO 800               'jednostkowego dla DC (częstotliwości zero)
460    SUM = SUM + B[I%]
470 NEXT I%
480 '
490 FOR I% = 0 TO 800
500    B[I%] = B[I%] / SUM
510 NEXT I%
520 '
530 FOR I% = 0 TO 800               'Zamień odpowiedź impulsową filtru dolnoprzepustowego w B[ ]
540   B[I%] = -B[I%]                'odpowiedź filtru górnoprzepustowego stosując inwersję widmową
545                                 '(jak na rys. 14.5)
550 NEXT I%
560 B[400] = B[400] + 1
570 '
580 '
590 FOR I% = 0 TO 800               'Dodaj odpowiedź impulsową filtru dolonoprzepustowego
595                                 'w A[ ] do odpowiedzi
600    H[I%] = A[I%] + B[I%]        'filtru górnoprzepustowego w B[ ], aby utworzyć odpowiedź
610 NEXT I%                         'impulsową filtru pasmowozaporowego zapamiętaną w H[ ]
615                                 '(jak na rys. 14.8)
620 '
630 FOR I% = 0 TO 800               'Zamień odpowiedź impulsową filtru pasmowozaporowego na odpo
640    H[I%] = -H[I%]               'filtru pasmowoprzepustowego stosując inwersję widmową
650 NEXT I%
660 H[400] = H[400] + 1
670 '                               'Odpowiedź impulsowa filtru pasmowoprzepustowego
675                                 'znajduje się teraz w H[ ]
680 END
```


W drugim przykładzie zaprojektujemy filtr *pasmowoprzepustowy* oddzielający *ton sygnalizacji* w sygnale fonicznym. Chodzi tu np. o oddzielenie sygnału powstającego przy naciśnięciu przycisku w telefonie. Zakładamy, że sygnał jest przetwarzany na cyfrowy z częstotliwością 10 kHz, a celem jest wyizolowanie pasma o szerokości 80 Hz wokół częstotliwości 2 kHz. Wyrażając te wartości w ułamkach częstotliwości próbkowania, chcemy stłumić wszystkie częstotliwości poniżej 0,196 i powyżej 0,204 (co odpowiada częstotliwościom 1960 Hz i 2040 Hz). W celu uzyskania pasma przejściowego równego 50 Hz (0,05 częstotliwości próbkowania) przyjmiemy długość odpowiedzi impulsowej filtru równą 801 punktów i zastosujemy okno Blackmana. W tablicy 16.2 zamieszczono program do obliczenia odpowiedzi impulsowej filtru, a na rys. 16.6 przedstawiono jego odpowiedź częstotliwościową. Projektowanie składa się z kilku etapów. W pierwszym projektuje się *dwa* filtry dolnoprzepustowe o częstotliwościach odcięcia 0,196 oraz 0,204. Ten drugi filtr jest następnie poddawany *inwersji widmowej*, w wyniku której staje się filtrem górnoprzepustowym (patrz rozdział 14, rys. 14.6). Następnie te dwie odpowiedzi impulsowe są sumowane, co daje w rezultacie filtr pasmowozaporowy (patrz rys. 14.8). Wreszcie, wykonuje się następną *inwersje widmową* dającą żądany filtr pasmowoprzepustowy.

Granice możliwości

W filtrach okienkowanych funkcją *sinc* można, bez żadnych kłopotliwych niespodzianek, uzyskać niewiarygodnie dobre właściwości. Przyjmijmy na przykład, że chcemy wyodrębnić sygnał o wartości 1 mV pojawiający się na linii energetycznej 120 V. W tym przypadku filtr dolnoprzepustowy powinien mieć współczynnik tłumienia w paśmie zaporowym co najmniej –120 dB (jedna część na milion – wyjaśnienie dla tych, co nie znoszą decybeli). Jak już poprzednio wyjaśniono, okno Blackmana daje tylko –74 dB (jedna część na pięć tysięcy). Na szczęście łatwo można uzyskać lepsze tłumienie. Sygnał wejściowy można wstępnie odfiltrować, stosując konwencjonalną odpowiedź impulsową filtru okienkowanego funkcją *sinc* i uzyskując w ten sposób sygnał pośredni. Ten sygnał może następnie być jeszcze raz przepuszczony przez ten filtr, co daje dalsze zwiększenie współczynnika tłumienia do –148 dB (1 część na 30 milionów, nie byle co!). Można również te dwa stopnie połączyć w jeden filtr. Odpowiedź impulsowa tego łączonego filtru będzie równa *splotowi* odpowiedzi obu stopni. Z tego wynika też, że splatanie dowolnej odpowiedzi impulsowej filtru z *samą sobą* daje w rezultacie odpowiedź impulsową o znacznie lepszym tłumieniu w paśmie zaporowym. Ceną, jaką się za to płaci, jest dłuższa odpowiedź impulsowa filtru i wolniejsze opadanie charakterystyki. Na rys. 16.7 a) przedstawiono odpowiedź częstotliwościową 201-punktowego filtru dolnoprzepustowego utworzonego przez splecenie z samą sobą 101-punktowej odpowiedzi impulsowej filtru okienkowanego funkcją *sinc* z oknem Blackmana. Zadziwiająco dobre są właściwości takiego filtru! Jeśli rzeczywiście jest niezbędne tłumienie lepsze niż –100 dB, to trzeba stosować zapis o podwójnej dokładności. Przy pojedynczej dokładności szum zaokrągleń na sygnale w paśmie *przepustowym* może też błędnie pojawiać się w paśmie *zaporowym* z amplitudą w zakresie od –100 dB do –120 dB.

Na rys. 16.7 b) przedstawiono inny przykład filtru okienkowanego funkcją *sinc* o niewiarygodnie dobrym działaniu. Jest to filtr dolnoprzepustowy o 32 001-punktowej odpowiedzi impulsowej. Odpowiedź częstotliwościowa jest taka, jak można oczekiwać – z opadaniem równym 0,000125 częstotliwości próbkowania. Jak dobry jest taki filtr? Spróbujcie zbudować analogowy filtr elektroniczny przepuszczający sygnały od napięcia stałego (DC) do 1000 Hz z wahaniami mniejszymi od 0,02% i z blokowaniem wszystkich częstotliwości powyżej 1001 Hz z pozostałością nie większą niż 0,02%. To jest dopiero filtr! Jeśli rzeczywiście chcecie być pod wrażeniem to, pamiętajcie, że w obu filtrach na rys. 16.7 zastosowano zapis o *pojedynczej dokładności*. Zastosowanie *podwójnej dokładności* umożliwia zwiększenie poziomu uzyskiwanych parametrów o *milion* razy.



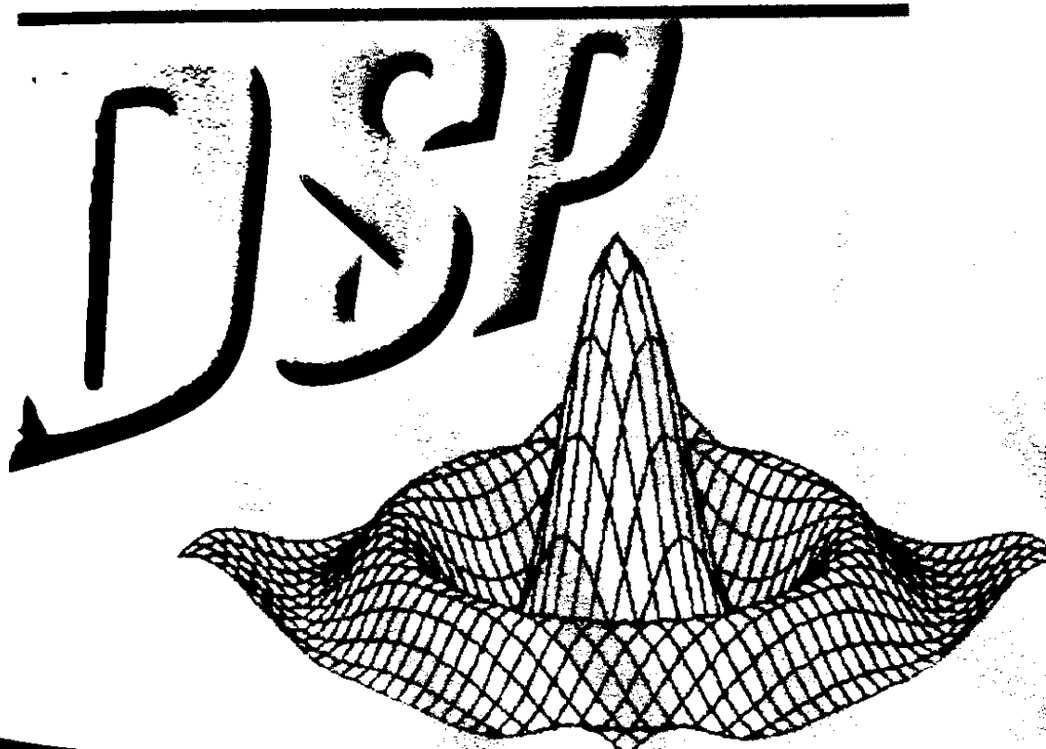
Rys. 16.7. Niewiarygodnie dobre właściwości filtru okienkowanego funkcją sinc. Na rys. a) pokazano odpowiedź częstotliwościową filtru okienkowanego funkcją sinc o zwiększonym tłumieniu w paśmie zaporowym. Osiągnięto to przez wykonanie splotu odpowiedzi impulsowej filtru okienkowanego funkcją sinc z samą sobą. Na rys. b) przedstawiono zaś bardzo szybkie opadanie charakterystyki w 32 001-punktowym filtrze okienkowanym funkcją sinc

Najsilniejszym ograniczeniem filtrów okienkowanych funkcją *sinc* jest *czas wykonania*. Może on się stać niedopuszczalnie długi, jeśli odpowiedź impulsowa zawiera bardzo wiele punktów i stosuje się algorytm ze standardowym splotem. Bardzo szybki algorytm do tego filtru (ze splotem FFT) podano w rozdziale 18. Filtry rekursywne (rozdział 19) też dają dobre rozdzielanie częstotliwości i są rozsądną alternatywą filtrów okienkowanych funkcją *sinc*.

Czy filtr okienkowany funkcją *sinc* jest optymalny do rozdzielania częstotliwości? Nie lepsze mogą być filtry o odpowiedziach impulsowych opracowanych przy użyciu bardziej skomplikowanych metod. Bądźcie jednak ostrożni! Zanim zdecydujecie się zastosować te bardzo matematyczne metody, dobrze się zastanówcie, co chcecie osiągnąć. Filtry okienkowane funkcją *sinc* mogą zapewnić parametry o *każdym* poziomie, jakiego moglibyście wymagać. Tym, co mogą dać metody bardziej skomplikowane, jest nieco krótsza odpowiedź impulsowa przy określonych parametrach. To z kolei może powodować nieco większą szybkość działania. Lecz ostrzegam, że możecie niewiele zyskać w zamian za włożony wysiłek.

17

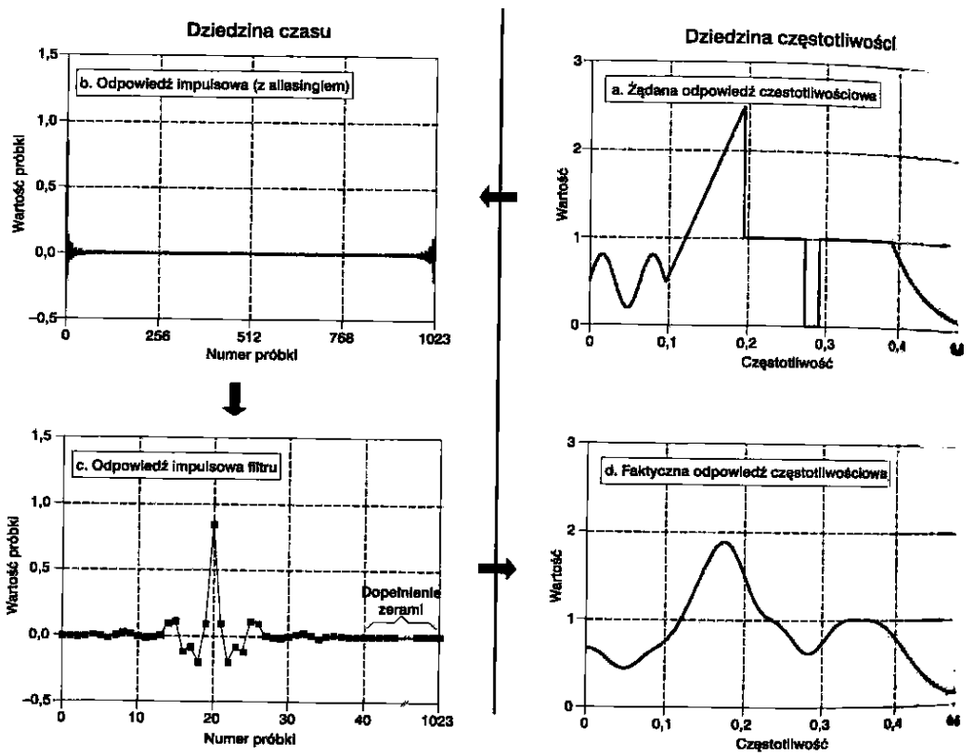
Filtry dostosowane do szczególnych wymagań



Większość filtrów charakteryzuje się jedną z czterech typowych odpowiedzi częstotliwo-
wych: dolnoprzepustową, górnoprzepustową, pasmowoprzepustową i pasmowozaporową. W
rozdziale przedstawiono ogólną metodę projektowania filtrów cyfrowych o *dowolnej* odpowie
częstotliwościowej, przystosowanej do szczególnych zastosowań. Technika DSP przodu
dziedzinie, dając rozwiązania problemów, które w znacznym stopniu przekraczają możliwo
elektroniki analogowej. Omówione będą dwie ważne dziedziny wykorzystania filtrów dos
wanych do szczególnych wymagań: *rozplot*, czyli sposób odtwarzania sygnałów, które uległy
pożądanemu splotowi oraz *filtracja optymalna*, a więc problem rozdzielania sygnałów, któ
widma nakładają się na siebie. W tych problemach zalety DSP są wyzyskane najlepiej.

Dowolna odpowiedź częstotliwościowa

Przedstawiony w poprzednim rozdziale sposób podejścia do zaprojektowania filtru okienk
wanego funkcją *sinc* może być też zastosowany do projektowania filtrów o w zasadzie tak
dej *dowolnej* odpowiedzi częstotliwościowej. Jedyną różnicą jest inna metoda przenoszen



Rys. 17.1. Przykład projektowania filtru SOI. Na rys. a) pokazano żądaną odpowiedź częstotliwościową zawierającą 513 próbek umieszczonych między 0 i 0,5 częstotliwości próbkowania. Wykonanie odwrotnego przekształcenia DFT daje przedstawioną na rys. b) odpowiedź impulsową z aliasingiem złożoną z 1024 punktów. W celu utworzenia odpowiedzi impulsowej filtru (rys. c) ta odpowiedź impulsowa z aliasingiem jest obcięta do M – 1 próbek, przesunięta w prawo o M/2 próbek i przemnożona przez okno Hamminga lub Blackmana. W tym przykładzie M jest równe 40. W programie w tablicy 17.1 przedstawiono, w jaki sposób to się realizuje. Odpowiedź impulsowa jest testowana przez dopełnienie jej zerami i wykonanie DFT. W ten sposób uzyskuje się rzeczywistą odpowiedź częstotliwościową filtru (rys. d)

żądanej odpowiedzi częstotliwościowej z dziedziny częstotliwości w dziedzinę czasu. W filtrze okienkowanym funkcją *sinc* zarówno odpowiedź częstotliwościowa, jak i odpowiedź impulsowa filtru są wyrażane *wzorami* i przejście z jednej dziedziny do drugiej jest realizowane matematycznie, przez przekształcenie Fouriera. Tymczasem w metodzie przedstawionej w tym rozdziale obie odpowiedzi są wyrażone *matrycami liczb*, a *program komputerowy* (FFT) jest stosowany do przejścia od jednej do drugiej.

Na rys. 17.1 pokazano, na czym polega ta metoda. Na rys. 17.1 a) przedstawiono żądaną odpowiedź częstotliwościową filtru. Jest ona, najogólniej mówiąc, bardzo nieregularna i byłaby z pewnością niemożliwa do zrealizowania przy użyciu elektroniki analogowej. Ta idealna odpowiedź częstotliwościowa jest *zdefiniowana* przez matrycę odpowiednio wybranych liczb, a nie przez jakiś wzór matematyczny. W tym przykładzie jest to 513 próbek rozmieszczonych między 0 i 0,5 częstotliwości próbkowania. Można zastosować więcej punktów, aby lepiej wyrazić żądaną odpowiedź częstotliwościową, podczas gdy mniejsza ich liczba może czasem być zalecana z uwagi na zredukowanie czasu obliczeń podczas projektowania filtru. Na ogół jednak ta ostatnia kwestia nie jest bardzo istotna i w większości zastosowań dobrym wyborem jest długość 513 punktów.

Oprócz matrycy zawierającej wartości *modułu* (rys. 17.1 a), musi być odpowiadająca jej matryca *fazy*, o takiej samej długości. W tym przykładzie faza żądanej odpowiedzi impulsowej jest całkowicie zerowa i matryca wartości fazy nie jest w ogóle pokazana na rys. 17.1. Podobnie jak do matrycy wartości modułu, także do matrycy wartości fazy może być wprowadzona dowolna krzywa, którą ma realizować filtr. Pamiętajcie jednak, że próbki pierwsza i ostatnia (tj. 0 i 512) w matrycy fazy muszą mieć wartość *zero* (lub wielokrotności 2π, co oznacza to samo). Odpowiedź częstotliwościowa może być też wyrażona we współrzędnych prostokątnych przez umieszczenie w matrycach *części rzeczywistej* i *urojonej*, zamiast modułu i fazy.

Tab. 17.1

```
' PROJEKT FILTRU DOSTOSOWANEGO DO SZCZEGÓLNYCH WYMAGAŃ
'Ten program przetwarza 1024-punktową odpowiedź impulsową z aliasingiem w (M + 1)
'punktową odpowiedź impulsową filtru (tak jak przetworzenie odpowiedzi z rys. 17.1 b) w tę
'na rys. 17.1 c)
'
DIM REX[1023]              'W REX[ ] jest przechowywany przetwarzany sygnał
DIM T[1023]                'T[ ] jest tymczasowym buforem pamięciowym
'
PI = 3.14159265
M% = 40                    'Ustaw odpowiedź impulsową filtru (cała ma 41 punktów)
'
GOSUB XXXX                 'Podprogram "mityczny" do ładowania odpowiedzi impulsowej do REX[ ]
'
FOR I% = 0 TO 1023         'Przesuń (obróć) sygnał o M/2 punktów w prawo
  INDEX% = I% + M%/2
  IF INDEX% > 1023 THEN INDEX% = INDEX%-1024
  T[INDEX%] = REX[I%]
NEXT I%
'
FOR I% = 0 TO 1023
  REX[I%] = T[I%]
NEXT I%
'                          'Obcinaj i okienkuj sygnał
FOR I% = 0 TO 1023
  IF I% <= M% THEN REX[I%] = REX[I%] * (0.54 - 0.46 * COS(2*PI*I%/M%))
  IF I% > M% THEN REX[I%] = 0
NEXT I%
'                          'Odpowiedź impulsowa filtru jest teraz umieszczona od REX[0] do REX[40]
END
```

Następnym krokiem jest wykonanie odwrotnego przekształcenia DFT, aby przenieść [illegible] w dziedzinę czasu. Najszybszym sposobem przeprowadzenia tej operacji jest przekształcenie sygnału w dziedzinie częstotliwości do współrzędnych prostokątnych i wykonanie następnie odwrotnego przekształcenia FFT. W rezultacie uzyska się 1024-próbkowy sygnał zmieniający się od próbki 0 do 1023, jak pokazano na rys. 17.1 b). Jest to odpowiedź impulsowa zgodna z odpowiedzią częstotliwościową, jaką chcemy uzyskać. Nie jest ona jednak odpowiednia do wykorzystania jako odpowiedź impulsowa filtru, o czym zaraz krótko powiemy. Podobnie jak to było w rozważaniach w poprzednim rozdziale, ta odpowiedź powinna zostać *przesunięta*, *obcięta* i *okienkowana*. W tym przykładzie zaprojektujemy odpowiedź impulsową filtru o $M = 40$, tj. 41 punktów zmieniających się od próbki 0 do 40. W tablicy 17.1 podano program komputerowy przetwarzający sygnał z rys. 17.1 b) w odpowiedź impulsową filtru pokazaną na rys. 17.1 c. Podobnie jak w filtrze okienkowanym funkcją *sinc*, punkty w pobliżu końca odpowiedzi impulsowej mają wartości tak małe, że na wykresie wyglądają jak równe zero. Nie pomylcie się, myśląc, że można je usunąć!

Ostatnim krokiem jest *testowanie* odpowiedzi impulsowej filtru. Robi się to, przeprowadzając DFT (z wykorzystaniem FFT), aby uzyskać faktyczną odpowiedź częstotliwościową, która pokazano na rys. 17.1 d). Aby uzyskać lepszą rozdzielczość w dziedzinie częstotliwości, trzeba przed wykonaniem FFT dopełnić odpowiedź impulsową filtru zerami. Na przykład stosując całkowitą liczbę próbek równą 1024 (41 w odpowiedzi impulsowej filtru oraz 983 zer), uzyskuje się w rezultacie 513 próbek między częstotliwościami 0 i 0,5.

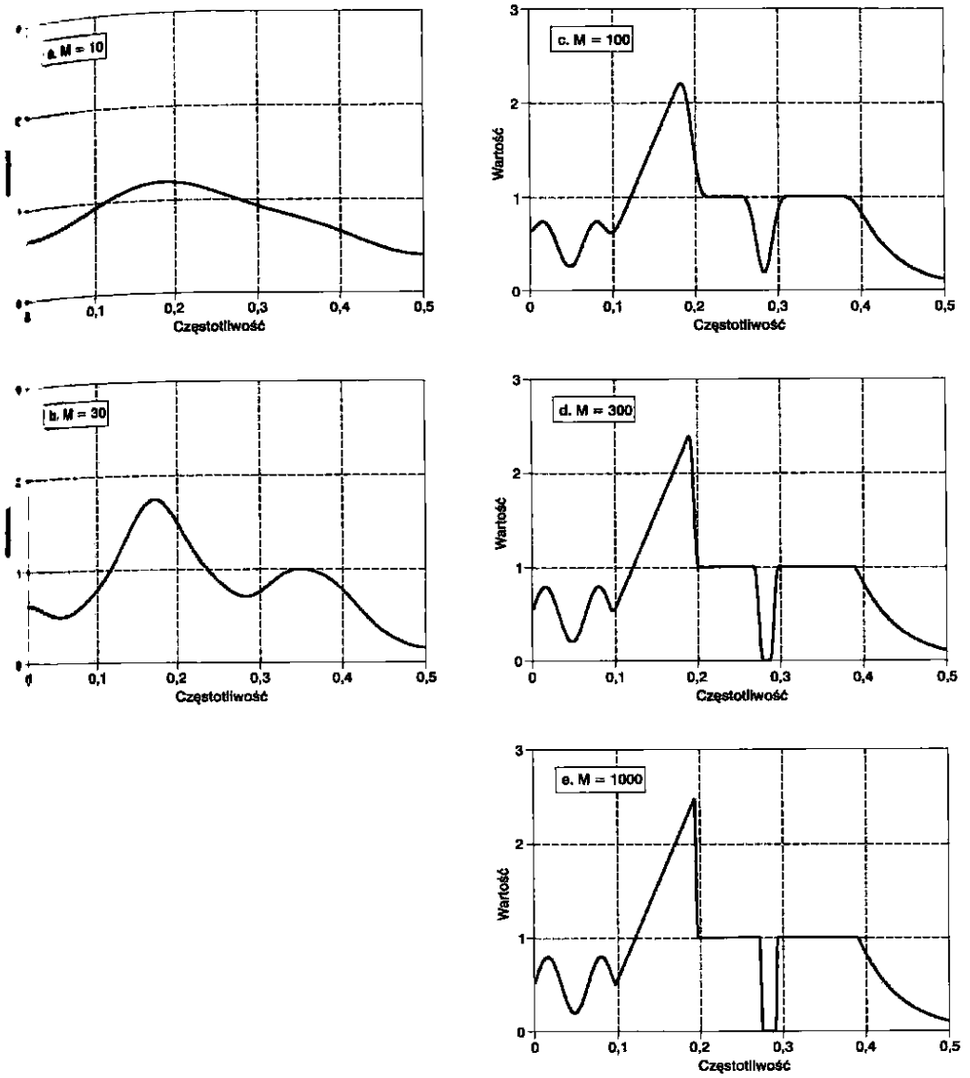
Długość odpowiedzi impulsowej filtru określa, jak dobrze *uzyskana* odpowiedź częstotliwościowa jest zgodna z *żądaną*, jak to przedstawiono na rys. 17.2. W oczywisty sposób jest widoczne wyjątkowe działanie filtrów cyfrowych SOI. Można w nich bowiem uzyskać każdą dowolną odpowiedź częstotliwościową, o ile tylko zastosuje się odpowiednio długą odpowiedź impulsową.

Opisano pełną metodę projektowania filtru. Jest jednak pewien subtelny problem teoretyczny, który należy wyjaśnić. Dlaczego nie można bezpośrednio zastosować odpowiedzi impulsowej pokazanej na rys. 17.1 b) jako odpowiedzi filtru? Przecież, jeśli sygnał na rys. 17.1 a) jest transformatą Fouriera sygnału na rys. 17.1 b), to czy splecenie sygnału wejściowego z przebiegiem na rys. 17.1 b) nie dałoby odpowiedzi częstotliwościowej dokładnie takiej, jak chcemy? Otóż *nie* i zaraz wyjaśnimy dlaczego.

Przy projektowaniu filtru, który ma spełniać szczególne wymagania, żądana odpowiedź częstotliwościowa jest określona wartościami zawartymi w matrycy. Zastanówmy się teraz nad pytaniem: jaki jest przebieg odpowiedzi częstotliwościowej *pomiędzy* poszczególnymi punktami? Dla uproszczenia można sobie wyobrazić dwa przypadki – „dobry" i „zły". W „dobrym" przypadku" odpowiedź częstotliwościowa jest gładką krzywą pomiędzy poszczególnymi próbkami. W „złym" przypadku między próbkami występują duże fluktuacje. Odpowiedź impulsowa na rys. 17.1 b) z pewnością daje „złą" odpowiedź impulsową. Można to udowodnić, dopełniając tę odpowiedź dużą liczbą zer i potem wykonując przekształcenie DFT. Otrzymana w ten sposób odpowiedź częstotliwościowa wykaże działanie błędne pomiędzy pierwotnie określonymi próbkami, i będzie wyglądała wręcz okropnie.

Aby to zrozumieć, wyobraźmy sobie, że wymuszamy żądaną odpowiedź częstotliwościową, opisując ją nieskończoną liczbą punktów między 0 i 0,5. W ten sposób tworzymy krzywą ciągłą. Następnie stosujemy odwrotne przekształcenie DTFT w celu wyznaczenia odpowiedzi impulsowej, która będzie *nieskończenie* długa. Innymi słowy, „dobra" odpowiedź częstotliwościowa koresponduje z przebiegiem, który nie może być reprezentowany w komputerze, gdyż jest odpowiedzią impulsową o długości nieskończenie dużej. Gdy widmo częstotliwościowe wyrażamy przez $N/2 + 1$ próbek, tylko N próbek jest dostępnych w dziedzinie czasu, uniemożliwiając prawidłowe pomieszczenie tam sygnału. W rezultacie nieskończenie długa odpowiedź impulsowa zachodzi na N punktów (ulega aliasingowi). Gdy następuje taki aliasing, odpowiedź częstotliwościowa zmienia się z „dobrej" na „złą". Na szczęście okienkowanie N-punktowej odpowiedzi impulsowej znacznie ogranicza aliasing, dając gładką krzywą między próbkami w dziedzinie częstotliwości.

Zaprojektowanie filtru cyfrowego dającego żądaną odpowiedź częstotliwościową jest całkiem proste. Trudną częścią projektu jest określenie, jaką odpowiedź częstotliwościową należy zastosować. Rozpatrzmy kilka różnych strategii stosowanych w DSP do projektowania filtrów dostosowanych do szczególnych wymagań.



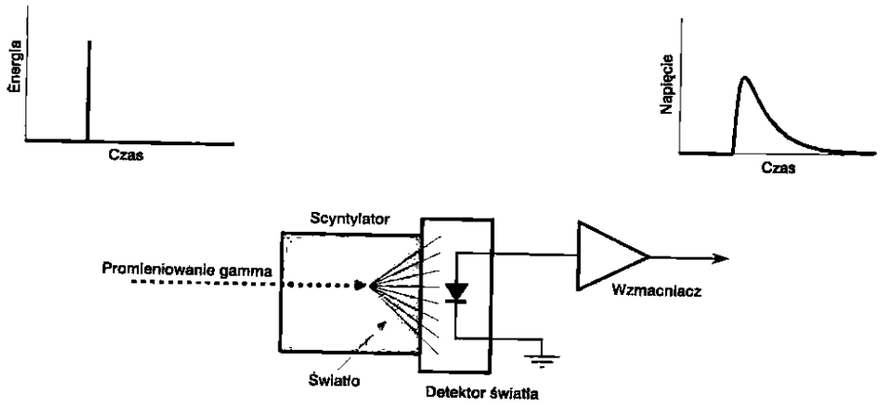
Rys. 17.2. Zależność odpowiedzi częstotliwościowej filtru od długości odpowiedzi impulsowej. Na rysunku przedstawiono odpowiedzi częstotliwościowe uzyskane przy różnych długościach odpowiedzi impulsowej filtrów. Liczba punktów w każdej odpowiedzi impulsowej jest $M + 1$, od próbki 0 do M. Czym więcej punktów zawiera odpowiedź impulsowa, tym lepiej uzyskana odpowiedź częstotliwościowa jest dopasowana do odpowiedzi żądanej. Żądaną odpowiedź impulsową można zobaczyć na rys. 17.1 a)

Rozplot

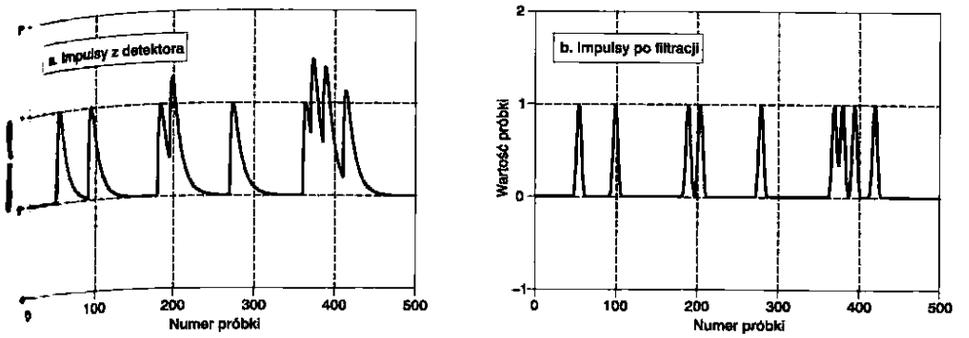
Niechciany splot jest problemem nieodłącznie związanym z przesyłaniem informacji cyfr gowej. Na przykład można jako splot modelować wszystkie wymienione zjawiska: niecen obrazu spowodowaną drżeniem kamery, echo w połączeniach telefonicznych na dużyc ległościach, skończoną szerokość pasma czujników analogowych i układów elektroni itd. Rozplot jest procesem filtracji sygnału kompensującym niepożądany splot. Celem plotu jest odtworzenie sygnału w postaci, jaką miał on, *zanim* nastąpił splot. Zwykle tego konieczna znajomość charakterystyki splotu (tj. odpowiedzi impulsowej lub częstotl ściowej). Trzeba taką sytuację odróżnić od tzw. **rozplotu „na oślep"** (*blind deconvolu* przy której charakterystyka pasożytniczego splotu *nie jest* znana. Rozplot „na oślep" znacznie trudniejszym problemem, niemającym ogólnego rozwiązania. Sposób postępow musi więc każdorazowo być dostosowany do danego zastosowania.

Rozplot jest prawie niemożliwy do zrozumienia w *dziedzinie czasu*, a jest całkiem p w *dziedzinie częstotliwości*. W każdej sinusoidzie wchodzącej w skład sygnału oryginalnego może nastąpić zmiana amplitudy i/lub fazy w wyniku niepożądanego splotu. W celu wyeli nienia sygnału oryginalnego filtr rozplatający musi *zlikwidować* te zmiany amplitudy i fazy. Jeśli na przykład splot zmienia amplitudę sinusoidy o czynnik 0,5 z przesunięciem fazy o 30° to filtr rozplatający musi wzmocnić sinusoidę o czynnik 2,0 ze zmianą fazy o –30°.

Przykładem ilustrującym rozplot niech będzie *detektor promieniowania gamma*. Urządz (rys. 17.3) składa się z dwóch części – *scyntylatora* i *detektora światła*. Scyntylator jest substancją przezroczystą o szczególnych właściwościach; może nim być np. jodek sodu lub ger manian bizmutu. W tych związkach chemicznych energia promieniowania gamma jest za mieniana na światło widzialne, które jest następnie przetwarzane na sygnał elektryczny w de tektorze światła, takim jak fotodioda lub fotopowielacz. Każdy impuls powstający w takim detektorze ma kształt *jednostronnie wykładniczy*, z zaokrągleniami w narożnikach. Ten kształt wynika z właściwości zastosowanego scyntylatora. Gdy kwant promieniowania gamma prze kazuje swoją energię scyntylatorowi, prawie wszystkie atomy są wzbudzane i przechodzą na wyższy poziom energetyczny. Następnie te atomy losowo powracają na poziom niższy. Każdy atom wysyła przy tym foton światła widzialnego. Sumarycznym wynikiem jest impuls świetl ny o wartości zanikającej w ciągu kilkuset nanosekund (w jodku sodu). Ponieważ pojawie



Rys. 17.3. Przykład nieuniknionego splotu. Detektor promieniowania gamma zawiera scyntylator umieszczony na detektorze światła. Gdy kwant promieniowania gamma uderza w scyntylator, jego energia jest zamieniana w impuls świetlny, który jest następnie w detektorze światła przetwarzany na impuls elektryczny. Kwant gamma jest impulsem, podczas gdy sygnał wyjściowy z detektora (tj. odpowiedź impulsowa) przypomina jednostronny przebieg wykładniczy



Rys. 17.4. Przykład rozplotu. Na rys. a) przedstawiono sygnał z detektora promieniowania gamma będący odpowiedzią ciąg pojawiających się losowo kwantów gamma. Zaprojektowano filtr rozplatający w taki sposób, aby przebieg z rys. a) przekształcić w przebieg na rys. b) przez ograniczenie szerokość impulsów. W ten sposób minimalizuje się zniekształcenie amplitudy spowodowane splętrzaniem impulsów

się każdego kwantu promieniowania gamma jest *impulsem*, więc impuls wyjściowy z detektora (tj. jednostronny przebieg wykładniczy) jest *odpowiedzią impulsową* systemu.

Na rys. 17.4 a) pokazano impulsy wytwarzane przez detektor w odpowiedzi na kwanty promieniowania gamma przychodzące w sposób losowy. Informacją, jaką chcemy wydobyć z tego sygnału wyjściowego, jest *amplituda* każdego z impulsów, proporcjonalna do *energii* kwantu gamma, który ten impuls wywołał. Taka informacja jest użyteczna, gdyż energia może wiele powiedzieć o pochodzeniu promieniowania gamma. Może na przykład dostarczyć informacji medycznej o pacjencie, umożliwić określenie wieku odległej galaktyki, wykryć bombę w bagażu lotniczym itd.

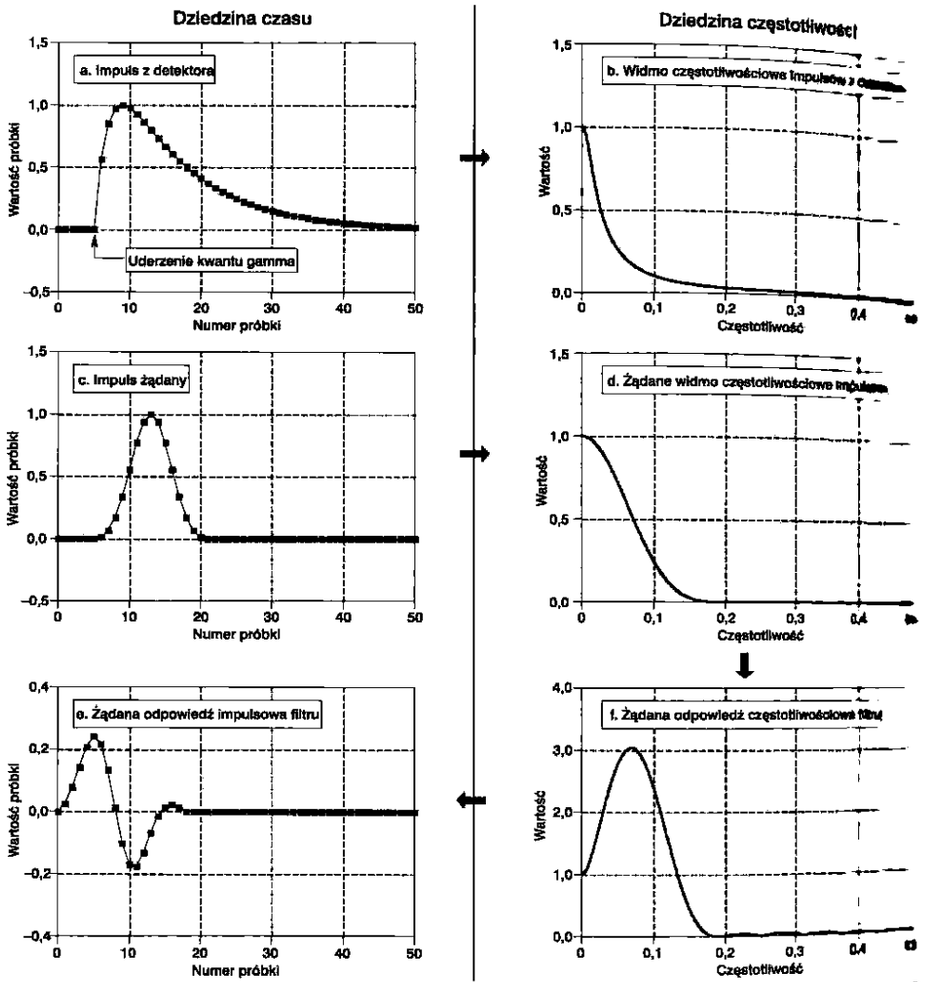
Wszystko byłoby łatwe, gdyby były wykrywane tylko pojedyncze kwanty gamma. Tak jednak zazwyczaj nie jest. Często dwa lub więcej impulsów może nakładać się na siebie (rys. 17.4 a), przesuwając mierzoną amplitudę. Jednym z rozwiązań tej trudności jest *rozplot* wyjściowego sygnału detektora, powodujący ich zwężenie, co ogranicza ich nakładanie. W idealnym przypadku chcielibyśmy, aby każdy impuls był zbliżony do impulsu oryginalnego. Jak się można spodziewać, nie jest to możliwe i musimy zadowolić się impulsami o skończonej szerokości, lecz znacznie krótszymi niż impuls wykrywany (rys. 17.4 b).

Chociaż sygnał z detektora zawiera informację zakodowaną w *dziedzinie czasu*, to większa część analizy musi być wykonywana w *dziedzinie częstotliwości*, w której ten problem jest łatwiej zrozumiały. Na rys. 17.5 a) przedstawiono powstający w detektorze sygnał, który już poznaliśmy. Na rys. 17.5 c) zaś mamy impuls, jaki chcielibyśmy uzyskać. Też go już poznaliśmy. Ten żądany impuls wybrano w sposób dowolny tak, aby miał kształt okna Blackmana i szerokość równą ok. 1/3 impulsu oryginalnego. Naszym celem jest znalezienie takiej odpowiedzi impulsowej filtru (rys. 17.5 e), która spleciona z sygnałem z rys. 17.5 a), daje sygnał pokazany na rys. 17.5 c). Można to wyrazić tak: jeśli $a * e = c$, to znajdźmy e mając dane a i c.

Gdyby te sygnały były ze sobą połączone, zamiast splotu, operacją dodawania lub mnożenia, to rozwiązanie byłoby proste: *odejmowanie* jest stosowane jako odwrotność dodawania, a *dzielenie* – mnożenia. Splot jest czymś innym, nie istnieje żadna prosta operacja, którą by można określić jako odwrotną w stosunku do splotu, czyli jako rozplot. Splot jest operacją zbyt skomplikowaną, aby rozplot można było wykonać przez proste operowanie sygnałami w dziedzinie czasu.

Ten problem jest, na szczęście, znacznie prostszy w dziedzinie częstotliwości. Przypomnijcie sobie, że *splot* w jednej dziedzinie jest równoważny *mnożeniu* w drugiej. Odnosząc się znowu do sygnału z rys. 17.5, mamy zagadnienie: jeśli $b \times f = d$, to znajdźmy f, mając dane b i d. Jest to problem łatwy do rozwiązania: odpowiedź częstotliwościowa (rys. 17.5 f) jest widmem czę-

stotliwościowym żądanego impulsu (rys. 17.5 d) *podzielonym* przez widmo częstotliwościowe impulsu wykrytego (rys. 17.5 b). Ponieważ impuls wykryty jest niesymetryczny, więc jego faza jest *niezerowa*. To oznacza, że trzeba zastosować dzielenie *zespolone* (tzn. moduł i faza podzielone przez inny moduł i fazę). W rozdziale 9 wyjaśniono, jak się wykonuje zespolone dzielenie jednego widma przez drugie. Żądaną odpowiedź impulsową filtru (rys. 17.5 e) można więc wyznaczyć na podstawie odpowiedzi częstotliwościowej metodą stosowaną do filtrów dostosowanych do szczególnych wymagań (odwrotne DFT, przesunięcie, obcięcie i pomnożenie przez okno).



Rys. 17.5. Rozplot w dziedzinach czasu i częstotliwości na przykładzie detektora promieniowania gamma. Odpowiedź impulsową detektora przedstawiono na rys. a), a pożądaną odpowiedź impulsową – na rys. c). Widmo częstotliwościowe tych dwóch sygnałów pokazano, odpowiednio, na rysunkach b), d). Filtr, który zamienia przebieg a) w przebieg c), ma odpowiedź częstotliwościową pokazaną na rys. f), równą przebiegowi d) podzielonemu przez przebieg b). Następnie wyznacza się odpowiedź impulsową tego filtru (rys. e) na podstawie jego odpowiedzi częstotliwościowej, korzystając z metody projektowania filtrów dostosowanych do szczególnych wymagań (odwrotne przekształcenie DFT, obcinanie, okienkowanie). Na rysunkach pokazano tylko moduły sygnałów w dziedzinie częstotliwości; jednak fazy są niezerowe i też trzeba je uwzględniać

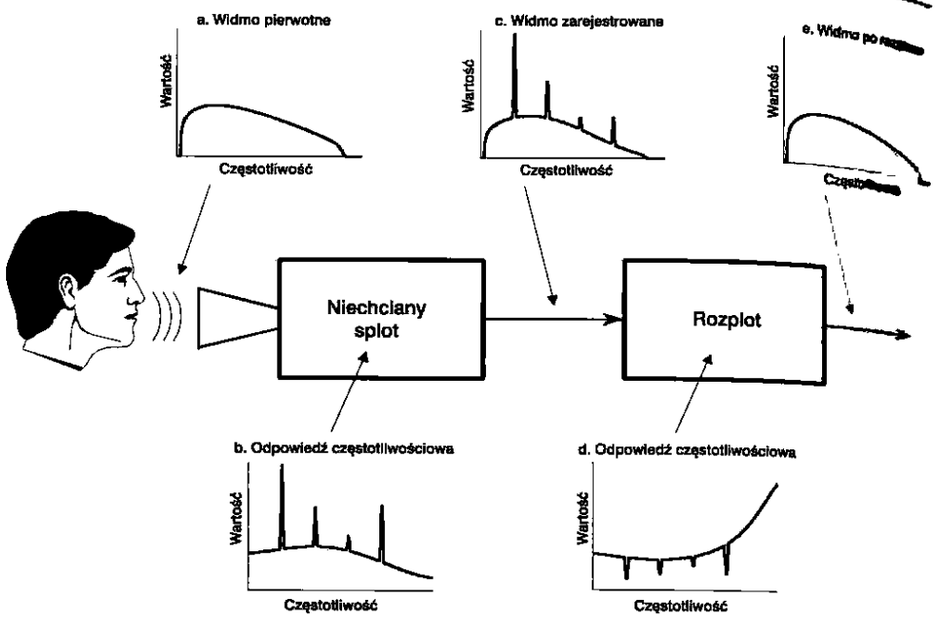
Są pewne granice udoskonaleń, które można osiągnąć przez rozplot. Można więc powiedzieć, że jeśli jesteśmy zbyt zachłanni, to zawodzi zastosowana metoda. W tym przypadku zachłanność oznacza próbę uzyskania nadmiernego zwężenia impulsu. Spójrzmy, co się wtedy dzieje. Jeśli żądany impuls zostaje zwężony, to jego widmo częstotliwościowe musi wtedy zawierać więcej składowych o dużych częstotliwościach. Ponieważ te składowe o dużych częstotliwościach mają w wykrywanym impulsie bardzo małe amplitudy, więc filtr powinien mieć bardzo duże wzmocnienie dla tych częstotliwości. Na przykład z rys. 17.5 f) wynika, że niektóre częstotliwości powinny zostać wzmocnione o czynnik *trzy*, aby można było uzyskać żądany impuls przedstawiony na rys. 17.5 c). Jeśli uczynić żądany impuls węższym, to wzmocnienie filtru rozplatającego przy dużych częstotliwościach musi być jeszcze większe.

Trudność polega na tym, że w tej sytuacji nawet małe błędy mszczą się bezlitośnie. Na przykład jeśli składowe o pewnych częstotliwościach będą wzmacniane o czynnik 30, podczas gdy wymagane jest tylko 28, to w sygnale rozplecionym powstanie przypuszczalnie bałagan. Chcąc przenieść rozplot na wyższy poziom działania, trzeba bardziej szczegółowo i ze znacznie większą dokładnością rozumieć właściwości splotu. W świecie rzeczywistych zastosowań zawsze są jakieś niewiadome spowodowane takimi kłopotliwymi zjawiskami, jak np. szumem elektronicznym, dryfem cieplnym, różnicami między poszczególnymi układami itp. Te właśnie niewiadome czynniki stawiają granice jakości działaniu rozplotu.

Nawet, jeśli niechciany splot jest *doskonale* rozumiany, to istnieje jednak pewien czynnik ograniczający działanie rozplotu – jest nim *szum*. Na przykład większość niechcianych splotów przyjmuje postać filtru dolnoprzepustowego, ograniczającego amplitudę składowych o dużych częstotliwościach w sygnale. Przez rozplot wprowadza się korektę, wzmacniając te częstotliwości. Jeśli jednak amplituda tych składowych obniży się poniżej poziomu nieuniknionego szumu systemu, to informacja w nich zawarta jest tracona. Nie odzyska się tych informacji przez żadne, choćby nie wiadomo jak rozbudowane, przetwarzanie sygnału. Jest stracona na zawsze. Żegnaj informacjo! Goodbye! Adios! Sayonara! Próba przywrócenia tych danych spowoduje tylko wzmocnienie szumu. W skrajnym przypadku wartość sygnału przy pewnych częstotliwościach może być sprowadzona do *zera*. To nie tylko wymazuje informację, lecz także będzie próbować wymusić *nieskończenie* duże wzmocnienie dla tych częstotliwości w filtrze rozplatającym. A oto rozwiązanie tych trudności: projektujcie mniej „agresywny" filtr rozplatający i/lub stawiajcie pewne ograniczenia na dopuszczalne wzmocnienie przy każdej z częstotliwości.

Jak daleko możemy się posunąć? Kiedy nasza „chciwość" jest zbyt duża? To całkowicie zależy od rozwiązywanego problemu. Gdy sygnał jest dość gładki i ma mały szum, wtedy można uzyskać znaczna poprawę (powiedzmy o czynnik 5–10). Gdy zaś sygnał bardzo się zmienia z upływem czasu, nie jest zbyt dobrze rozumiany lub jest zaszumiony, to poprawa będzie słabsza (powiedzmy o czynnik 1–2). Pomyślny rozplot wymaga sporego testowania. Gdy rozplot działa na pewnym poziomie, spróbujcie pójść nieco dalej i zauważycie, kiedy wszystko traci sens. Żadna analiza teoretyczna nie zastąpi takiego procesu iteracyjnego.

Rozplot można też zastosować do sygnałów zakodowanych w *dziedzinie częstotliwości*. Klasycznym przykładem jest rekonstrukcja starych nagrań sławnego śpiewaka operowego Enrica Carusa (1873–1921). Dokonywano tych nagrań na sprzęcie bardzo prymitywnym, jak na obecne standardy. Najpoważniejszym problemem są *rezonanse* długiej tuby rurowej służącej do zbierania dźwięku przy nagraniach. Za każdym razem, gdy śpiewak trafia w jedną z częstotliwości rezonansowych, głośność nagrania gwałtownie wzrasta. Rozplot cyfrowy poprawia subiektywną jakość tych nagrań przez stłumienie głośnych momentów muzyki. Opiszemy tu tylko zasadę ogólną. Opis szczegółowy można znaleźć w oryginalnej publikacji: T. Stockham, T. Cannon i R. Ingebretsen: *Blind Deconvolution Through Digital Signal Processing*, „Proc. IEEE", vol. 63, kwiecień 1975, str. 678–692.



Rys. 17.6. Rekonstrukcja starych nagrań fonograficznych metodą rozplotu. Oryginalne widmo częstotliwościowe sygnału głosowego śpiewaka pokazano na rys. a). Piki rezonansowe w prymitywnej aparaturze (rys. b) powodują zniekształcenie nagranego widma częstotliwościowego (rys. c). Odpowiedź częstotliwościową filtru rozplatającego (rys. d) zaprojektowano w taki sposób, aby przeciwdziałała niepożądanemu splotowi, odtwarzając widmo oryginalne (rys. e). Podane wykresy mają wyłącznie charakter ilustracyjny, nie są to rzeczywiste sygnały

Na rys. 17.6 przedstawiono ogólną zasadę działania metody. Widmo częstotliwościowe oryginalnego sygnału fonicznego pokazano na rys. 17.6 a). Na rys. 17.6 b) przedstawiono odpowiedź częstotliwościową aparatury nagrywającej – dość gładką krzywą, mającą jednak kilka ostrych pików. Widmo sygnału nagranego (rys. 17.6 c) jest równe widmu prawdziwemu (rys. 17.6 a) pomnożonemu przez niejednostajną odpowiedź częstotliwościową (rys. 17.6 b). Celem rozplotu jest *przeciwdziałanie* niepożądanemu splotowi. Innymi słowy, odpowiedź częstotliwościowa filtru rozplatającego (rys. 17.5 d) musi być odwróceniem odpowiedzi z rys. 17.6 b). To znaczy, że każdy pik na rys. 17.5 b) będzie kasowany przez odpowiadające mu wgłębienie na przeciwległej na rys. 17.6 d). Gdyby filtr zaprojektowano w sposób doskonały, to uzyskany sygnał miałby widmo (rys. 17.6 e) identyczne z sygnałem oryginalnym. Ale jest tu jednak pułapka. Aparatura używana do tych starych nagrań od dawna nie jest już stosowana i jej odpowiedź częstotliwościowa pozostaje zagadką. Mamy tu więc problem *rozplotu „na oślep"* – jak więc można określić przebieg z rys. 17.6 d), dysponując tylko widmem pokazanym na rys. 17.6 c)?

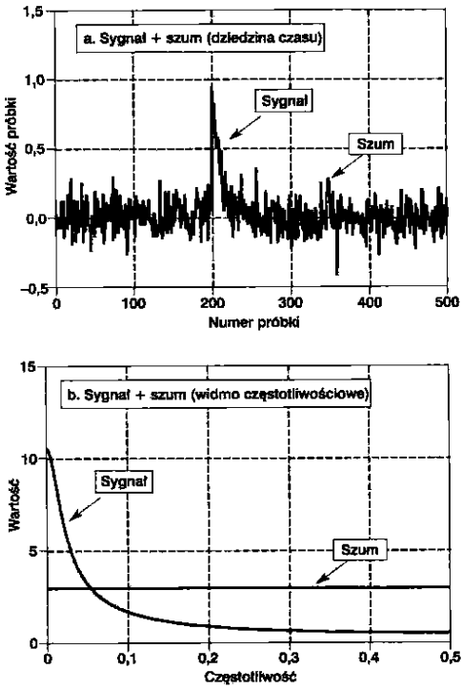
Zagadnienia związane z rozplotem „na oślep" są zwykle rozwiązywane przez przyjęcie formatu lub przypuszczalnych wartości nieznanych parametrów. W omawianym przykładzie trzeba założyć pewne *średnie widmo* muzyki oryginalnej dopasowane do *średniego widma* tej samej muzyki wykonywanej przez współczesnych śpiewaków przy użyciu nowoczesnego sprzętu. *Średnie widmo* określa się, stosując metodę podaną w rozdziale 9. Rozdziela się więc sygnał na wielką liczbę segmentów, wykonuje się przekształcenie DFT każdego segmentu, przechodząc do współrzędnych biegunowych i następnie uśrednia się moduły. W najprostszym przypadku przyjmuje się nieznaną odpowiedź częstotliwościową jako uśrednione widmo starych nagrań podzielone przez średnie widmo nagrań współczesnych. Metoda użyta przez Stockhama i in. była bardziej wysublimowana, gdyż wykorzystano w niej technikę zwaną przetwarzaniem *homomorficznym*, dającym lepszą estymatę właściwości aparatury nagrywającej.

Filtry optymalne

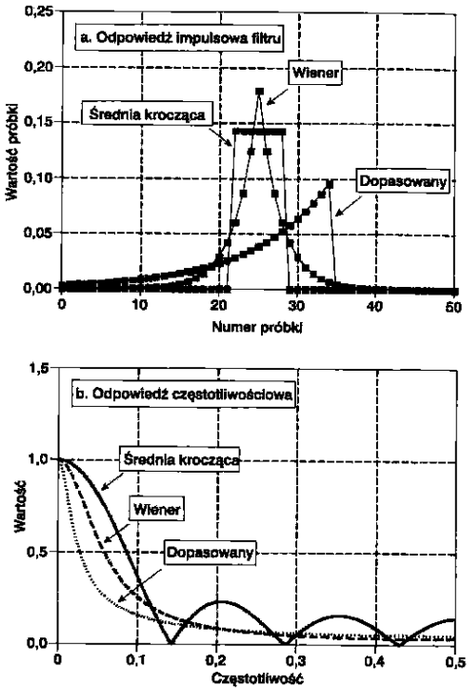
Na rys. 17.7 a) przedstawiono często spotykany problem wymagający filtracji. Chodzi tu o próbę wydobycia przebiegu (w tym przypadku impulsu wykładniczego) ukrytego w szumie losowym. Ten problem, jak pokazano na rys. 17.7 b), wcale nie jest łatwiejszy w dziedzinie częstotliwości. Sygnał ma widmo złożone głównie ze składowych o małych częstotliwościach. Szum zaś ma widmo *białe* (jednakowa amplituda przy wszystkich częstotliwościach). Ponieważ widma szumu i sygnału *nakładają się* na siebie, więc nie jest jasne, jak je można w najlepszy sposób rozdzielić. Faktycznie, istotną kwestią jest zdefiniowanie, co tutaj oznacza słowo „najlepszy". Rozpatrzymy trzy filtry, z których każdy jest w innym sensie „najlepszy" (optymalny). Na rys. 17.8 przedstawiono odpowiedzi impulsowe i częstotliwościowe wszystkich trzech filtrów, a na rys. 17.9 – wyniki ich użycia do filtracji przebiegu z rys. 17.7 a).

Filtr o średniej kroczącej był tematem rozdziału 15. Jak pamiętacie, każdy punkt w sygnale jest wyznaczany przez filtr o średniej kroczącej jako średnia pewnej liczby punktów z sygnału wejściowego. To powoduje, że odpowiedź impulsowa tego filtru jest impulsem prostokątnym o amplitudzie równej odwrotności liczby punktów w tej średniej. Filtr o średniej kroczącej jest optymalny w tym sensie, że daje najszybszą odpowiedź skokową przy danym tłumieniu szumu.

Filtr dopasowany omówiono już poprzednio w rozdziale 7. Jak pokazano na rys. 17.8 a), odpowiedź impulsowa filtru dopasowanego jest taka sama, jak sygnał, który ma być wykryty, z tą tylko różnicą, że jest odwrócony lewo-prawostronnie. Zasadą filtru dopasowanego jest *korelacja*, i takie odwrócenie jest niezbędne do przeprowadzenia *korelacji* przy użyciu



Rys. 17.7. Przykład filtracji optymalnej. Na rys. a) przedstawiono sygnał wykładniczy ukryty w szumie losowym, a na rys. b) - widma częstotliwościowe sygnału i szumu. Ponieważ sygnał i szum nakładają się na siebie zarówno w dziedzinie czasu, jak i częstotliwości, więc nie jest oczywiste określenie najlepszego sposobu ich rozdzielenia



Rys. 17.8. Przykład filtrów optymalnych. Na rys. a) przedstawiono trzy odpowiedzi impulsowe filtrów, z których każdy jest w jakimś sensie optymalny. Równoważne im odpowiedzi częstotliwościowe pokazano na rys. b). Filtr o średniej kroczącej zaprojektowano w taki sposób, aby jego odpowiedź impulsowa była impulsem prostokątnym. Odpowiedź impulsowa filtru dopasowanego zaś ma wygląd wykrywanego impulsu. Filtr Wienera został zaprojektowany w dziedzinie częstotliwości na podstawie względnych stosunków sygnału do szumu dla każdej częstotliwości

splotu. Amplituda każdego punktu w sygnale wyjściowym jest miarą tego, jak dobrze odpowiedź impulsowa filtru jest *dopasowana* do odpowiedniego segmentu sygnału wejściowego. Przypomnijcie sobie, że sygnał na wyjściu filtru dopasowanego niekoniecznie wygląda tak jak wykrywany sygnał. To, w istocie rzeczy, nie ma większego znaczenia. Jeśli stosujemy ten filtr dopasowany, to kształt docelowego impulsu wykrywanego musi już być znany. Filtr dopasowany jest optymalny w tym sensie, że szczyt piku sięga powyżej szumu znacznie wyżej niż to jest możliwe w jakimkolwiek innym filtrze liniowym (patrz rys. 17.9 b).

Filtr Wienera (nazwany tak od nazwiska twórcy teorii estymacji Norberta Wienera) rozdziela sygnały na podstawie ich widm częstotliwościowych. Jak pokazano na rys. 17.7 b, przy pewnych częstotliwościach występuje głównie sygnał, a przy innych – głównie szum. Wydaje się logiczne, że częstotliwości, przy których mamy głównie sygnał, powinny być przepuszczane przez filtr, a te z szumem – stłumione. Z taką zasadą w filtrze Wienera posunięto się jeszcze o krok dalej; przy *każdej częstotliwości* wzmocnienie filtru jest określone przez względny stosunek sygnału do szumu przy *tej częstotliwości*:

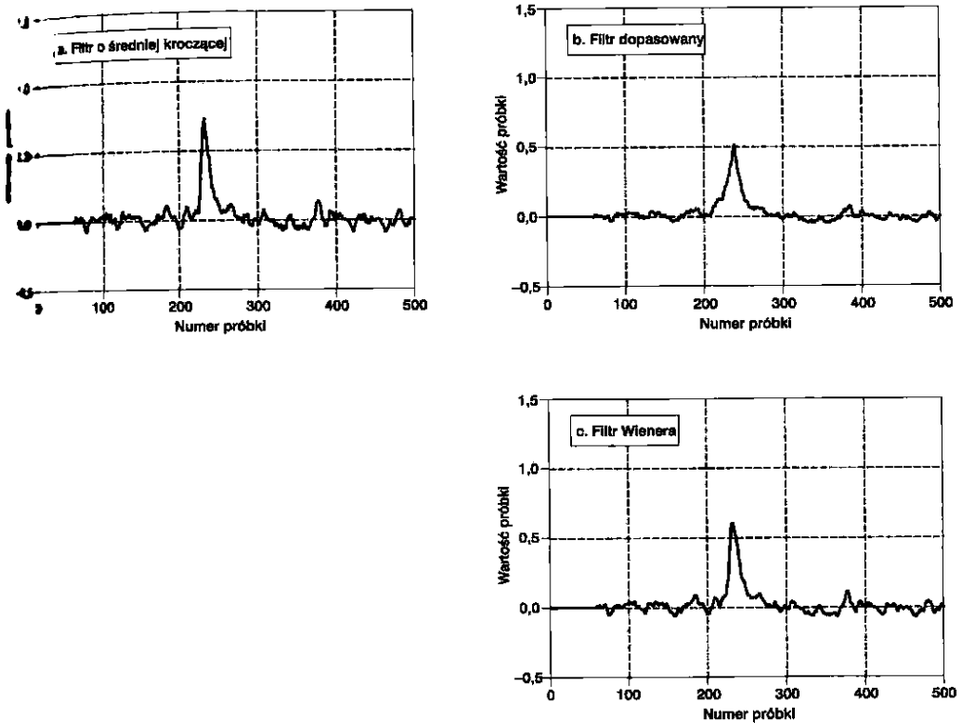
$$H[f]=\frac{S[f]^2}{S[f]^2+N[f]^2}$$

Wzór 17.1. Filtr Wienera. Odpowiedź częstotliwościowa H[f] jest określona przez widma częstotliwościowe szumu N[f] i sygnału S[f]. Istotne są tylko moduły, wszystkie fazy są zerowe

Powyższa zależność jest stosowana do przekształcenia widma z rys. 17.7 b) w odpowiedź częstotliwościową filtru Wienera z rys. 17.8 b). Filtr Wienera jest optymalny w tym sensie, że maksymalizuje stosunek mocy sygnału do mocy szumu (na całej szerokości sygnału, nie w poszczególnych jego punktach). Właściwa odpowiedź impulsowa filtru jest projektowana na podstawie odpowiedzi częstotliwościowej Wienera przy użyciu metod projektowania filtrów dostosowanych do szczególnych wymagań.

Chociaż pomysły dotyczące filtrów optymalnych są matematycznie eleganckie, to często zawodzą w praktyce. Nie wynika stąd, że nie powinny być w ogóle stosowane. Chodzi o to, żebyście nie przestawali myśleć, usłyszawszy słowo „optymalny". Przyjrzyjmy się kilku przyczynom, dla których możecie *nie* chcieć stosować takich filtrów.

Pierwszy powód: różnice między sygnałami na rys. 17.9 są niezbyt duże. Rzeczywiście, gdyby nie powiedziano, które parametry zostały zoptymalizowane, to prawdopodobnie nie dałoby się tego stwierdzić, patrząc na przebiegi sygnałów. Tak jest na ogół wtedy, gdy w problemie występują nakładające się widma częstotliwościowe. Niewielka poprawa działania osiągnięta dzięki filtrowi optymalnemu może nie być warta zwiększonej złożoności programu, dodatkowego wysiłku włożonego w oprogramowanie lub dłuższego czasu wykonania.



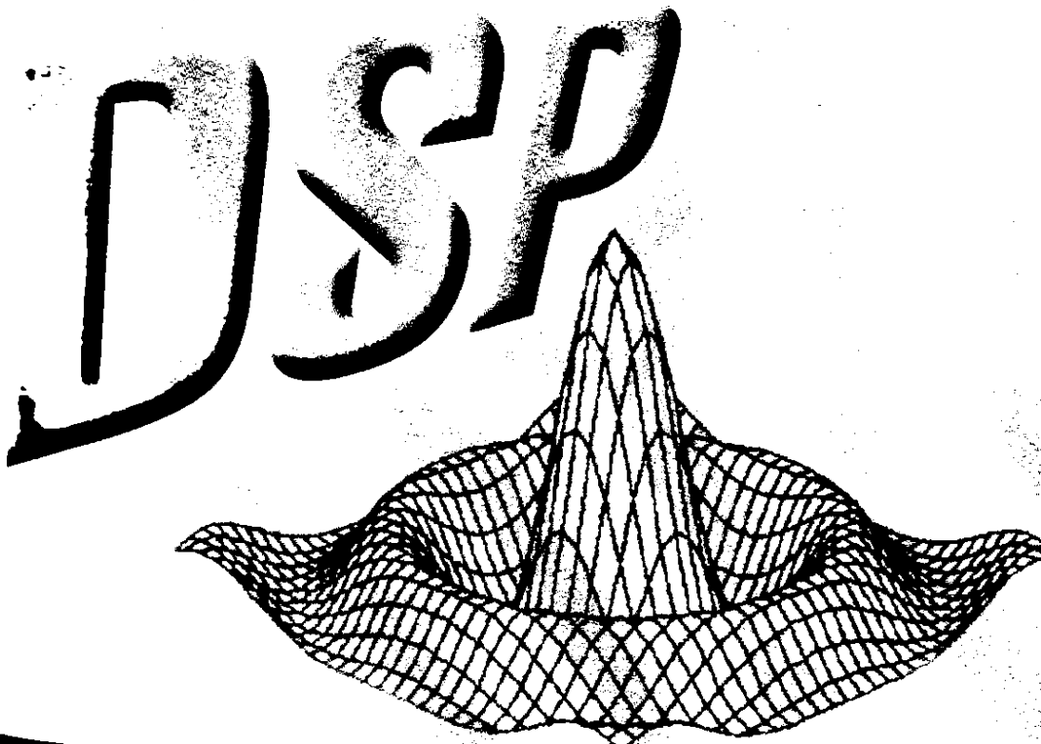
Rys. 17.9. Przykład zastosowania trzech filtrów optymalnych. Przedstawione sygnały są wynikiem filtracji przebiegu z rys. 17.7 filtrami przedstawionymi na rys. 17.8. Każdy z filtrów jest w jakimś sensie optymalny. Filtr o średniej kroczącej (rys. a) daje najostrzejszą odpowiedź na zbocze przy danym poziomie tłumienia szumu losowego. Filtr dopasowany (rys. b) daje pik położony nad resztkowym szumem znacznie wyżej niż w innych filtrach. Filtr Wienera (rys. c) jest optymalny pod względem stosunku sygnału do szumu

Drugi powód: Filtry Wienera i dopasowane są całkowicie określone przez właściwości danego problemu. Inne filtry, takie jak okienkowane funkcją *sinc* i ze średnią kroczącą, mogą być dostosowane do życzeń użytkownika. Zwolennicy filtrów optymalnych mogą twierdzić, że wszelkie dodatkowe manipulacje w tych innych filtrach mogą tylko zmniejszyć efektywność filtru. Jest to kwestia sporna. Pamiętajcie, że każdy z tych filtrów jest optymalny tylko pod jakimś jednym określonym względem. To rzadko wystarcza do stwierdzenia, że cały problem został zoptymalizowany, zwłaszcza jeśli uzyskane w rezultacie sygnały są interpretowane przez człowieka. Na przykład inżynier biomedyk mógłby zastosować filtr Wienera do maksymalizacji stosunku sygnału do szumu w elektrokardiogramie. Nie jest jednak oczywiste, czy nastąpi wówczas także optymalizacja możliwości lekarza w wykrywaniu nieregularności pracy serca na podstawie oglądanego sygnału.

Trzeci powód: Filtry Wienera i dopasowany muszą być realizowane przy użyciu *splotu*, przez co ich wykonywanie jest bardzo wolne. Nawet przy zastosowaniu sposobów poprawy szybkości opisanych w następnym rozdziale (splot FFT) czas wykonywania może być bardzo długi. Filtry *rekursywne* zaś (jak np. filtr o średniej kroczącej lub inne opisane w rozdziale 19) są znacznie szybsze, zapewniając jednocześnie wystarczająco dobry poziom jakości działania.

18

Splot FFT



W tym rozdziale będą przedstawione dwie ważne metody DSP – *metoda nakładania z dodawaniem* (*overlap-add*) oraz *splot FFT*. Metoda nakładania z dodawaniem jest stosowana do rozdzielenia długich sygnałów na mniejsze segmenty w celu łatwiejszego ich przetwarzania. W metodzie splotu FFT wykorzystuje się metodę nakładania z dodawaniem wraz z szybkim przekształceniem Fouriera, dzięki czemu sygnały mogą być splatane przez wymnażanie ich widm częstotliwościowych. Dla filtrów o odpowiedzi impulsowej dłuższej niż ok. 64 punkty splot FFT jest szybszy niż splot standardowy, dając dokładnie taki sam wynik.

Metoda nakładania z dodawaniem (*overlap-add*)

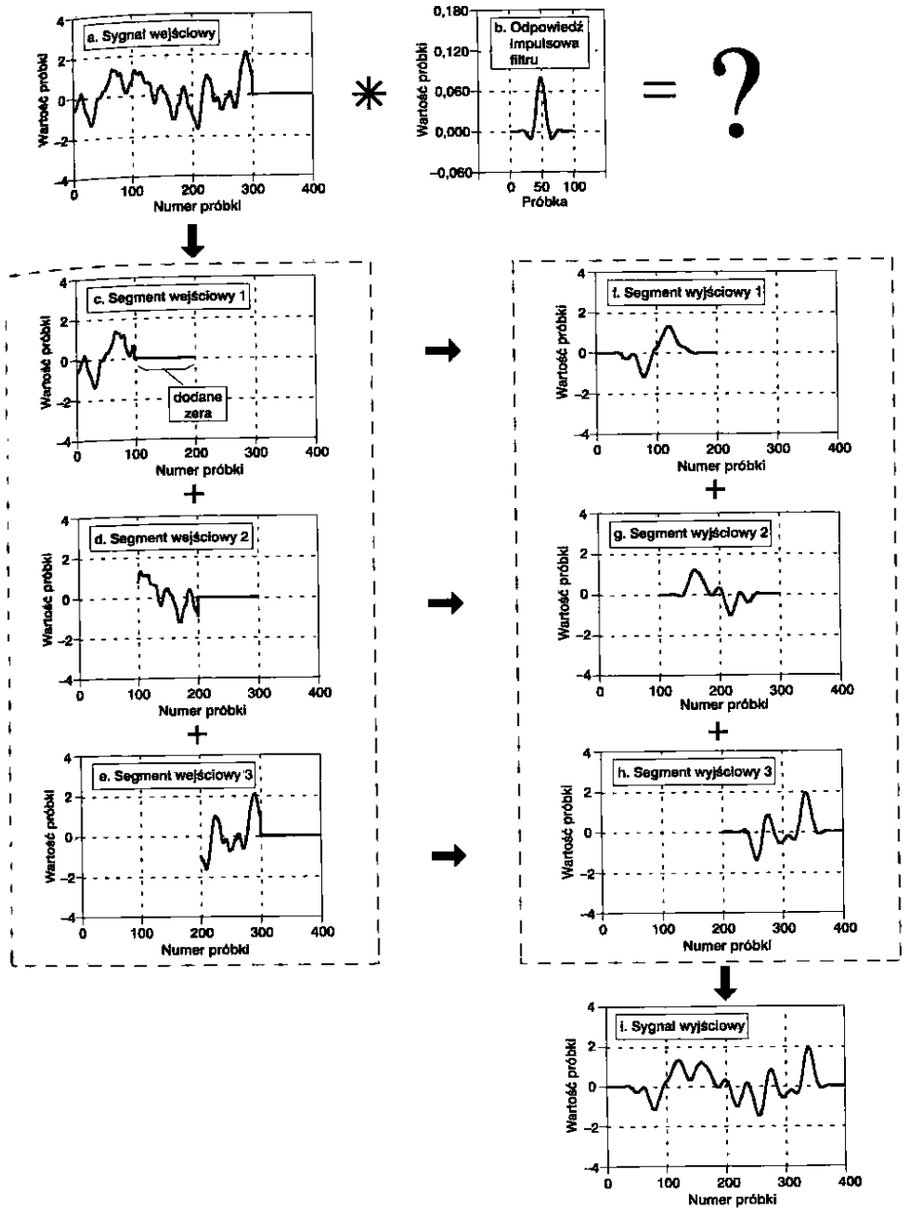
Jest wiele zastosowań DSP, w których długi sygnał musi być filtrowany *segmentami*. Na przykład cyfrowy sygnał *audio* o wysokiej wierności (hi-fi) wymaga przepływności bitowej 5 MB/min, podczas gdy cyfrowy sygnał *wideo* ok. 500 MB/min. Przy tak dużych szybkościach przesyłania danych często się zdarza, że pamięć komputera jest niewystarczająca do jednoczesnego przechowywania całego przetwarzanego sygnału. Są też systemy, które przetwarzają segment po segmencie, gdyż pracują w *czasie rzeczywistym*. Na przykład sygnały telefoniczne nie powinny być opóźniane o więcej niż kilkaset milisekund, co ogranicza ilość danych dostępnych do przetwarzania w każdej dowolnej chwili. Są jeszcze inne zastosowania, w których *przetwarzanie* może wymagać podziału sygnału na segmenty. Przykładem jest splot FFT będący głównym tematem tego rozdziału.

W metodzie nakładania z dodawaniem (*overlap-add*) korzysta się z podstawowej procedury DSP: 1) rozkład sygnału na proste składowe, 2) przetworzenie każdej składowej w jakiś użyteczny sposób, 3) połączenie przetworzonych składowych w końcowy sygnał. Na rys. 18.1 przedstawiono przykład realizacji takiej procedury w metodzie nakładania z sumowaniem. Na rys. 18.1 a) pokazano sygnał, który ma być filtrowany, a na rys. 18.1 b) – zastosowaną odpowiedź impulsową dolnoprzepustowego filtru okienkowanego funkcją *sinc*. Przechodząc na dół rysunku, widzimy sygnał odfiltrowany (rys. 18.1 i) będący wygładzoną wersją sygnału wejściowego. Kwestią zasadniczą w tej metodzie jest zależność *długości* tych sygnałów od splotu. Gdy sygnał N-próbkowy jest splatany z M-próbkową odpowiedzią impulsową, to sygnał wyjściowy ma długość równą $N + M - 1$ próbek. Na przykład sygnał wejściowy (rys. 18.1 a) ma długość 300 próbek (od 0 do 299), odpowiedź impulsowa (rys. 18.1 b) 101 próbek (od 0 do 100) i sygnał wyjściowy (rys. 18.1 i) 400 próbek (od 0 do 399).

Tak więc, gdy N-punktowy sygnał jest filtrowany, jest on *rozszerzany* o $M - 1$ punktów w prawo. (Tak jest przy założeniu, że odpowiedź impulsowa zmienia się od indeksu 0 do M. Jeśli w odpowiedzi impulsowej stosuje się indeksy ujemne, to rozszerzenie nastąpi także w lewo.) Na rys. 18.1 a) do sygnału dodano zera między próbkami 300 i 399, aby pokazać, gdzie nastąpuje rozszerzenie. Niech was nie dziwią małe wartości na krańcu sygnału wyjściowego (rys. 18.1 i). Jest to po prostu wynikiem odpowiedzi impulsowej filtru okienkowanego funkcją *sinc* mającą małe wartości na swych krańcach. Wszystkie 400 próbek na rys. 18.1 i) ma wartości niezerowe, choć niektóre z nich są zbyt małe, aby były widoczne na wykresie.

Na rysunkach 18 c), d), e) przedstawiono rozkład zastosowany w metodzie nakładania z dodawaniem. Sygnał zostaje rozdzielony na segmenty. Każdy z nich zawiera 100 próbek sygnału oryginalnego. Ponadto dodano po 100 zer z prawej strony każdego segmentu. W następnym kroku każdy segment jest indywidualnie filtrowany przez splatanie go z odpowiedzią impulsową filtru. W ten sposób uzyskuje się segmenty wyjściowe pokazane na rysunkach 18.1 f), g), h). Ponieważ każdy segment wejściowy ma długość 100 próbek, a odpowiedź impulsowa 101 próbek, więc każdy segment wyjściowy będzie miał długość 200 próbek. Rzeczą ważną do zrozumienia jest to, że 100 zer dodano do każdego segmentu po to, aby umożliwić rozszerzenie podczas operacji splotu.

Zauważmy, że rozszerzenie powoduje *nakładanie się* segmentów wyjściowych. Te nakładające się segmenty wyjściowe są sumowane, dając w wyniku sygnał wyjściowy (rys. 18.1 i)



Rys. 18.1. Metoda nakładania z dodawaniem. Celem jest wykonanie splotu sygnału wejściowego (rys. a) z odpowiedzią impulsową filtru (rys. b). Realizuje się to przez rozdzielenie sygnału wejściowego na pewną liczbę segmentów, takich jak na rys. c), d), e). Każdy z tych segmentów uzupełniono taką liczbą zer, aby umożliwiała rozszerzenie podczas wykonywania splotu. Splatając każdy segment wejściowy z odpowiedzią impulsową filtru, uzyskuje się segmenty wyjściowe rys. f, g, h). Następnie sygnał wyjściowy (rys. i) jest wyznaczany przez sumowanie nakładających się segmentów wyjściowych

Na przykład próbki od 200 do 299 są obliczane prze sumowanie odpowiednich próbek na rysunkach 18.1 g), h). Metoda nakładania z dodawaniem daje dokładnie taki sam sygnał wyjściowy, jak bezpośredni splot. Wadą jest znacznie większa złożoność programu ze względu na konieczność nadążania za nakładającymi się próbkami.

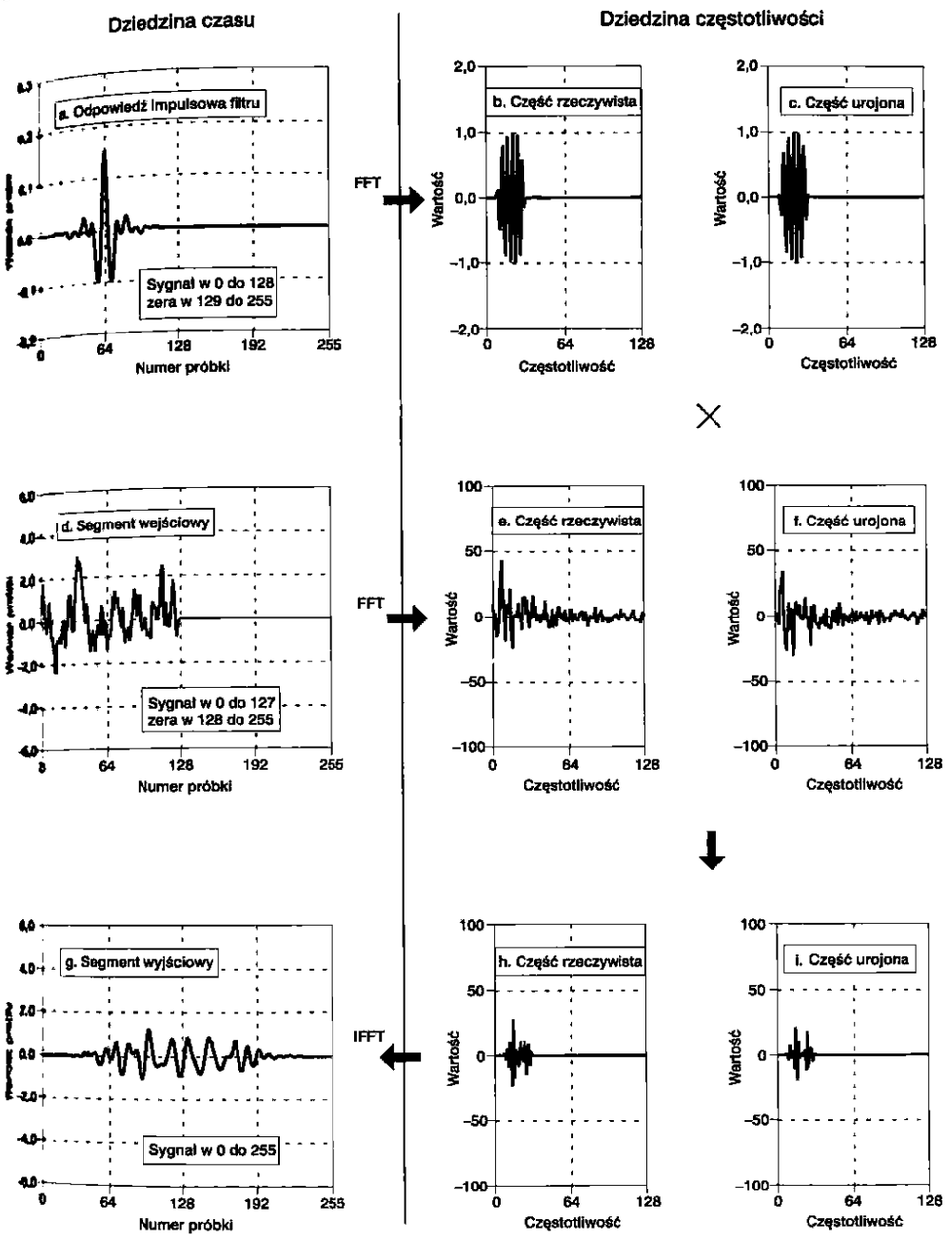
Splot FFT

W splocie FFT korzysta się z reguły, zgodnie z którą *mnożenie* w dziedzinie częstotliwości odpowiada *splotowi* w dziedzinie czasu. Sygnał wejściowy jest przenoszony w dziedzinę częstotliwości przez przekształcenie DFT, potem jest wymnażany przez odpowiedź częstotliwościową filtru i z powrotem przenoszony w dziedzinę czasu przez odwrotne przekształcenie DFT. Ta podstawowa metoda jest znana od czasów Fouriera. Wtedy jednak nikt się nią nie zajmował, bo czas obliczenia DFT był wówczas *dłuższy* niż czas potrzebny do bezpośredniego obliczania splotu. To się zmieniło w roku 1965 z chwilą opracowania szybkiej transformaty Fouriera (FFT). Przy zastosowaniu algorytmu FFT do przeprowadzania DFT splot realizowany przez dziedzinę częstotliwości może być *szybszy* niż bezpośrednie splatanie sygnałów w dziedzinie czasu. Końcowy wynik jest taki sam; zmieniła się tylko liczba obliczeń dzięki bardziej efektywnemu algorytmowi. Z tego powodu splot FFT jest nazywany szybkim **splotem**.

Przy wyznaczaniu splotu przez FFT stosuje się metodę nakładania z dodawaniem, przedstawioną na rys. 18.1. Zmieniono tylko sposób, w jaki segmenty wejściowe są przetwarzane na wyjściowe. Na rys. 18.2 pokazano przykład przetwarzania segmentu wejściowego na wyjściowy przez splot FFT. Najpierw określa się odpowiedź częstotliwościową filtru przez wyznaczenie transformaty DFT jego odpowiedzi impulsowej przy użyciu FFT. Na rys. 18.2 a) przedstawiono przykład odpowiedzi impulsowej pasmowoprzepustowego filtru okienkowanego funkcją *sinc*. Przez FFT następuje przekształcenie tej odpowiedzi na części rzeczywistą i urojoną odpowiedzi częstotliwościowej (rys. 18.2 b, c). Te sygnały w dziedzinie częstotliwości być może *nie wyglądają* jak filtr pasmowoprzepustowy, gdyż są wyrażone we współrzędnych prostokątnych. Przypomnijcie sobie, że najłatwiej zrozumieć problem, stosując współrzędne biegunowe, a do obliczeń matematycznych najkorzystniejsze są współrzędne prostokątne. Części rzeczywista i urojona zostają zapisane w komputerze do wykorzystania przy obliczeniach poszczególnych segmentów.

Na rys. 18.2 d) pokazano segment wejściowy, który ma być przetwarzany. Stosuje się FFT do znalezienia widma częstotliwościowego (rys. 18.2 e, f). Następnie wyznacza się widmo częstotliwościowe segmentu wyjściowego (rys. 18.2 h, i) przez wymnożenie odpowiedzi częstotliwościowych (rys. 18.2 b, c) przez widmo segmentu wejściowego (rys. 18.2 e, f). Te widma składają się z części rzeczywistych i urojonych, dlatego są wymnażane według wzoru 9.1 w rozdziale 9. Potem stosuje się odwrotne przekształcenie FFT w celu wyznaczenia segmentu wyjściowego (rys. 18.2 g) na podstawie jego widma częstotliwościowego (rys. 18.2 h, i). Trzeba przy tym pamiętać, że ten segment wyjściowy jest dokładnie taki sam, jaki by uzyskano przez bezpośredni splot segmentu wejściowego (rys. 18.2 d) i odpowiedzi częstotliwościowej filtru (rys. 18.2 a).

Przekształcenie FFT musi być wystarczająco długie, aby nie nastąpił *splot kołowy* (też opisany w rozdziale 9). To znaczy, że FFT powinno mieć długość taką samą, jak segment wyjściowy (rys. 18.2 g). W przykładzie z rys. 18.2 odpowiedź impulsowa filtru zawiera 128 punktów i każdy segment zawiera 128 punktów, co powoduje, że segment wyjściowy ma długość 256 punktów. To stawia wymaganie użycia 256-punktowego FFT. Odpowiedź impulsowa filtru (rys. 18.2 a) musi więc być dopełniona 127 zerami, aby osiągnęła całkowitą długość 256 punktów. Podobnie każdy z segmentów wejściowych (rys. 18.2 d) powinien być dopełniony 128 zerami. Jako inny przykład wyobraźmy sobie, że mamy zrealizować splot bardzo długiego sygnału z odpowiedzią impulsową filtru mającą 600 próbek. Jedną alternatywą byłoby zastosowanie segmentów po 425 punktów i 1024-punktowych FFT. Drugie – użycie segmentów po 1449 punktów i 2048-punktowyh FFT.



Rys. 18.2. Splot FFT. Odpowiedź impulsowa (rys. a) i segment sygnału (rys. d) są przez FFT przekształcane w odpowiadające im widma (rys. b, c oraz e, f). Te widma są wymnażane przez siebie, dając w wyniku widmo segmentu wyjściowego (rys. h, i). Następnie stosując odwrotne przekształcenie FFT, znajduje się segment wyjściowy (rys. g). Przez IFFT oznaczono odwrotne FFT

Tab. 18.1

```
100 'SPLOT FFT
110 'Ten program realizuje splot sygnału o 10 milionach punktów z 400-punktową odpowiedzią
120 'impulsową filtru. Sygnał wejściowy jest dzielony na 16000 segmentów, każdy po 625 punktów.
125 'Zastosowane jest 1024-punktowe przekształcenie FFT.
130'                        'INICJALIZACJA MATRYC
140 DIM XX[1023]            'sygnał w dziedzinie czasu (do FFT)
150 DIM REX[512]            'część rzeczywista sygnału w dziedzinie częstotliwości (do FFT)
160 DIM IMX[512]            'część urojona sygnału w dziedzinie częstotliwości (do FFT)
170 DIM REFR[512]           'część rzeczywista odpowiedzi częstotliwościowej filtru
180 DIM IMFR[512]           'część urojona odpowiedzi częstotliwościowej filtru
190 DIM OLAP[398]           'przechowuje próbki nakładające się z jednego segmentu na drugi
200 '
210 FOR I% = 0 TO 398       'zeruj matrycę zawierającą nakładające się próbki
220   OLAP[I%]=0
230 NEXT I%
240 '
250 '                       'WYZNACZ I ZAPAMIĘTAJ ODPOWIEDŹ CZĘSTOTLIWOŚCIOWĄ FILTRU
260 GOSUB XXXX              'Podprogram "mityczny" do ładowania odpowiedzi impulsowej filtru do XX
270 GOSUB XXXX              'Podprogram "mityczny" FFT: XX[ ]-- >REX[ ] & IMX[ ]
280 FOR F% = 0 TO 512       'Zapamiętaj odpowiedź częstotliwościową w REFR[ ] & IMFR[ ]
290   REFR[F%] = REX[F%]
300   IMFR[F%] = IMX[F%]
310 NEXT F%
320 '
330 '                       'PRZETWARZAJ KAŻDY Z 1600 SEGMENTÓW
340 FOR SEGMENT% = 0 TO 15999
350   '
360   GOSUB XXXX            'Podprogram "mityczny" do ładowania następnego segmentu wejścio
365                         'do XX[ ]
370   GOSUB XXXX            'Podprogram "mityczny" FFT: XX[ ] -- >REX[ ] & IMX[ ]
380   '
390   FOR F% = 0 TO 512     'Pomnóż widmo częstotliwościowe przez odpowiedź częstotliwościową
400     TEMP    = REX[F%]*REFR[F%]-IMX[F%]*IMFR[F%]
410     IMX[F%] = REX[F%]*IMFR[F%]+IMX[F%]*REFR[F%]
420     REX[F%] = TEMP
430   NEXT F%
440   '
450   GOSUB XXXX            'Podprogram "mityczny" IFFT: REX[ ] & IMX[ ] -- > XX[ ]
460   '
470   FOR I% = 0 TO 398     'Dodaj nałożenie się ostatniego segmentu do tego segmentu
480     XX[I%] = XX[I%] + OLAP[I%]
490   NEXT I%
500   '
510   FOR I% = 625 TO 1023  'Zapamiętaj próbki, które będą nakładały się na następny segment
520     OLAP[I%-625] = XX[I%]
530   NEXT I%
540   '
550   GOSUB XXXX            'Podprogram "mityczny" do wyprowadzania 625 próbek
560   '                     'przechowywanych od XX[0] do XX[624]
570   '
580 NEXT SEGMENT%
590 '
600 GOSUB XXXX              'Podprogram "mityczny" do wyprowadzania wszystkich 399 próbek do
610 END
```


W tablicy 18.1 podano przykład programu realizującego splot FFT. Program filtruje sygnał złożony z 10 milionów punktów przez splatanie go z 400-punktową odpowiedzią impulsową filtru. Jest to wykonywane przez rozdzielenie sygnału wejściowego na 16 000 segmentów zawierających po 625 punktów. Gdy każdy z tych segmentów jest splatany z odpowiedzią impulsową, to uzyskuje się segment wyjściowy złożony z 625 + 400 – 1 = 1024 punktów. Są więc stosowane 1024-punktowe przekształcenia FFT. Po zdefiniowaniu i zainicjowaniu matryc (wiersze od 130 do 230) pierwszym krokiem jest obliczenie i zarejestrowanie odpowiedzi impulsowej filtru (wiersze od 250 do 310). Wiersz 260 wywołuje „mityczny" podprogram ładujący odpowiedź impulsową do matrycy od XX[0] do XX[399] i ustawiający od XX[400] do XX[1023] na wartość zero. Podprogram w wierszu 270 jest przekształceniem FFT, przetwarzającym 1024 próbki przechowywane w XX[ ] na 513 próbek przechowywanych w REX[ ] oraz IMX[ ], a więc na części rzeczywistą i urojoną odpowiedzi częstotliwościowej. Te wartości są przekazywane do matryc REFR[ ] oraz IMFR[ ] (skrót od: REal and IMaginary Frequency Response), aby mogły być użyte w dalszej części programu.

Pętla FOR-NEXT między wierszami 340 i 580 steruje przetwarzaniem 16 000 segmentów. W wierszu 360 podprogram ładuje do macierzy od XX[0] do XX[624] następny segment, który ma być przetwarzany oraz ustawia od XX[625] do XX[1023] na wartość zero. W wierszu 370 podprogram FFT zastosowano do znalezienia widma częstotliwościowego tego segmentu. Część rzeczywista tego widma jest umieszczona w 513 punktach REX[ ], a część urojona w 513 punktach IMX[ ]. Wiersze od 390 do 430 zawierają wymnażanie odpowiedzi częstotliwościowej segmentu przechowywanej w REX[ ] i IMX[ ] przez odpowiedź częstotliwościową filtru przechowywaną w REFR[ ] i IMFR[ ]. Wynik tego mnożenia jest wpisany do REX[ ] i IMX[ ] w miejsce danych tam poprzednio przechowywanych. Ponieważ jest to teraz widmo częstotliwościowe segmentu wyjściowego, więc można zastosować IFFT, aby wyznaczyć segment wyjściowy. Korzysta się przy tym z „mitycznego" podprogramu w wierszu 450, który przetwarza 513 punktów przechowywanych w REX[ ] i IMX[ ] na 1024 punkty zapisane do XX[ ], które są segmentem wyjściowym.

Wiersze od 470 do 550 obsługują nakładanie się segmentów. Każdy segment wyjściowy jest dzielony na dwie sekcje. Pierwsze 625 punktów (od 0 do 624) trzeba połączyć z nałożeniem się *poprzedniego* segmentu wyjściowego i następnie wpisać do sygnału wyjściowego. Ostatnie 399 punktów (625 do 1023) należy zarejestrować, gdyż mogą się one nakładać na następny segment wyjściowy.

Aby to zrozumieć, spójrzmy raz jeszcze na rys. 18.1. Próbki od 100 do 199 na rys. 18.1 g) trzeba połączyć z nałożeniem od *poprzedniego* segmentu wyjściowego (rys. 18.1 f) i potem powinny być one przeniesione do sygnału wyjściowego (rys. 18.1 i). Próbki od 200 do 299 na rys. 18.1 g) trzeba zapamiętać, aby można je potem połączyć z *następnym* segmentem wyjściowym (rys. 18.1 h).

Teraz wróćmy do programu. Matryca OLAP[ ] jest używana do przechowywania 399 próbek, które z jednego segmentu nakładają się na następny. W wierszach od 470 do 490 te 399 wartości z tej matrycy (z poprzedniego segmentu wyjściowego) jest dodawanych do właśnie przetwarzanego segmentu wyjściowego, przechowywanego w XX[ ]. „Mityczny" podprogram w wierszu 550 dostarcza następnie 625 próbek zawartych w matrycy od XX[0] do XX[624] do zbioru obejmującego sygnał wyjściowy. Następnie te 399 próbek z bieżącego segmentu wyjściowego, które muszą być przetrzymane do następnego segmentu wyjściowego, są zapamiętywane w OLAP[ ] w wierszach od 510 do 530. Po przetworzeniu wszystkich segmentów od 0 do 15999, matryca OLAP[ ] będzie zawierać 399 próbek z segmentu 15 999, które powinny nałożyć się na segment 16 000. Ponieważ segment 16 000 nie istnieje (lub może być traktowany jako zawierający same zera), więc te 399 próbek wpisuje się do sygnału wyjściowego w wierszu 600. To powoduje, że sygnał wyjściowy ma długość 1600 × 625 + 399 = 10 000 399 punktów. Jest to równe sumie długości sygnału wejściowego i długości odpowiedzi impulsowej filtru minus 1.

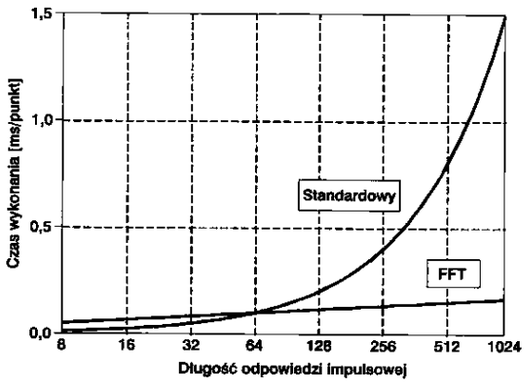
Polepszenie szybkości

Kiedy FFT jest szybsze od splotu standardowego? Odpowiedź na to pytanie zależy od długości odpowiedzi impulsowej filtru, jak to pokazano na rys. 18.3. Czas wykonania splotu standardowego jest wprost proporcjonalny do liczby punktów w odpowiedzi impulsowej filtru. Wymagany czas splotu FFT zaś rośnie bardzo wolno, tylko *logarytmicznie*, w miarę wzrostu liczby punktów w odpowiedzi impulsowej. Przecięcie się wykresów tych dwóch zależności następuje przy liczbie próbek w odpowiedzi równej ok. od 40 do 80 (zależnie od zastosowanych rozwiązań sprzętowych).

Ważna rzecz do zapamiętania: odpowiedzi impulsowe filtru krótsze niż ok. 60 punktów można realizować szybciej splotem standardowym i czas wykonania jest wtedy wprost proporcjonalny do długości odpowiedzi impulsowej filtru. Dłuższe odpowiedzi impulsowe można szybciej realizować splotem FFT. W tym przypadku można odpowiedź impulsową wybrać dowolnie długą, gdyż długość ma niewielki wpływ na czas wykonania. Na przykład odpowiedź 16 000-punktowa wymaga czasu tylko *dwukrotnie* dłuższego niż odpowiedź 64-punktowa.

Szybkość splotu decyduje również o *dokładności* obliczeń (jak to opisano przy omawianiu FFT w rozdziale 12). Przyczyną jest zależność błędu zaokrąglania w sygnale wyjściowym od całkowitej liczby obliczeń, która z kolei jest wprost proporcjonalna do czasu obliczania. Jeśli sygnał wyjściowy zostaje obliczony *szybciej*, to wtedy będzie on też obliczony *dokładniej*. Rozpatrzmy na przykład splot sygnału z 1000-punktową odpowiedzią częstotliwościową, z obliczaniem zmiennoprzecinkowym o pojedynczej dokładności. Stosując splot standardowy, można oczekiwać typowego błędu zaokrąglania ok. 1 część na 20 000 (jak wynika z wytycznych w rozdziale 4). Można się natomiast spodziewać, że splot FFT będzie o rząd wielkości *szybszy* i także o rząd wielkości *dokładniejszy* (tj. 1 część na 200 000).

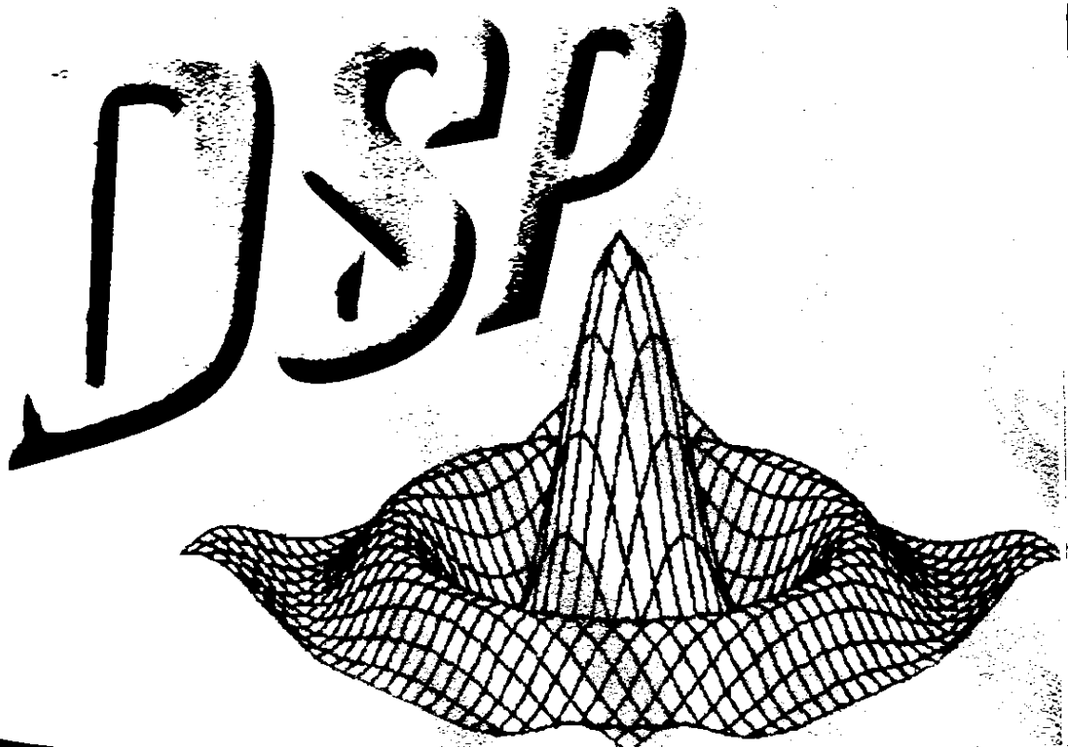
Splot FFT zachowajcie do tych sytuacji, gdy macie bardzo dużo danych do przetworzenia i musicie zastosować bardzo długą odpowiedź impulsową filtru. Trzeba tu myśleć w kategoriach sygnału o *milionie* próbek i odpowiedzi impulsowej o *tysiącu* próbek. Wszelkie mniejsze liczby próbek nie uzasadniałyby wkładania dodatkowego wysiłku w oprogramowanie. Jeśli nie chcecie pisać własnego podprogramu splotu FFT, zajrzyjcie do bibliotek oprogramowania i znajdźcie tam gotowy program.



Rys. 18.3. Czas wykonania splotu FFT. Splot FFT jest szybszy od standardowego, gdy odpowiedź impulsowa filtru jest dłuższa niż ok. 60 punktów. Podane czasy wykonania dotyczą komputera Pentium 100 MHz przy obliczeniach zmiennoprzecinkowych z pojedynczą dokładnością

19

Filtry rekursywne



Filtry rekursywne są skutecznym sposobem uzyskania długiej odpowiedzi impulsowej bez konieczności wykonywania długiego splotu. Te filtry działają bardzo szybko, lecz ich parametry i łatwość przystosowania do wymagań są gorsze niż w innych filtrach cyfrowych. Filtry rekursywne są też nazywane *filtrami o nieskończonej odpowiedzi impulsowej* (*IIR – Infinite Impulse Response*), gdyż ich odpowiedź impulsowa składa się z opadających funkcji wykładniczych. To odróżnia te filtry od filtrów cyfrowych realizowanych przez splot i nazywanych *filtrami o skończonej odpowiedzi impulsowej SOI* (*FIR – Finite Impulse Response*). Ten rozdział jest wprowadzeniem do filtrów rekursywnych wyjaśniającym jak działają te filtry i jak można je projektować w prosty sposób. Bardziej zaawansowane metody projektowania będą omówione w rozdziałach 20, 26 i 33.

Metoda rekursywna

Zaczynając rozważanie filtrów rekursywnych, wyobraźcie sobie, że chcecie wydobyć informację z jakiegoś sygnału $x[\ ]$. Wasza potrzeba uzyskania tej informacji jest tak wielka, że zatrudniacie starego profesora matematyki, żeby dla was przetwarzał dane. Zadaniem profesora jest odfiltrować $x[\ ]$, aby uzyskać $y[\ ]$, które – jak oczekujecie – zawiera interesującą was informację. Profesor zaczyna swoją pracę od obliczenia każdego punktu w $y[\ ]$ według pewnego algorytmu, który profesor ma szczegółowo zapisany w swoim, ponad miarę rozwiniętym, mózgu. W trakcie realizacji zadania następują wyjątkowo niefortunne zdarzenia. Profesor zaczyna coś pleść o analitycznych osobliwościach lub transformacjach fraktalnych i innych demonach rodem z koszmarnego, matematycznego snu. Staje się jasne, że profesor doznał pomieszania zmysłów. Patrzycie z niepokojem, jak kilku ludzi w białych fartuchach zabiera profesora wraz z waszym algorytmem.

Gorączkowo przeszukujecie notatki profesora, aby znaleźć algorytm, który stosował. Stwierdzacie, że zdążył ukończyć obliczenia punktów od $y[0]$ do $y[27]$ i właśnie miał zacząć obliczanie punktu $y[28]$. Jak pokazano na rys. 19.1, przyjmujemy wartość zmiennej równej n do reprezentowania punktu, który jest aktualnie obliczany. To znaczy, że $y[n]$ jest próbką 28 w sygnale wyjściowym, $y[n-1]$ – próbką 27, $y[n-2]$ – próbką 26 itd. Podobnie $x[n]$ jest punktem 28 w sygnale wejściowym, $x[n-1]$ – punktem 27 itd. Aby zrozumieć zastosowany algorytm, postawcie sobie pytanie: „Jaka informacja była dostępna profesorowi do obliczenia $y[n]$ – próbki aktualnie obliczanej?."

Najbardziej oczywistym źródłem informacji jest *sygnał wejściowy*, to znaczy wartości $x[n]$, $x[n-1]$, $x[n-2]$, ... Profesor mógł wymnażać każdy punkt sygnału wejściowego przez pewien współczynnik i sumować uzyskane iloczyny:

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + ...$$

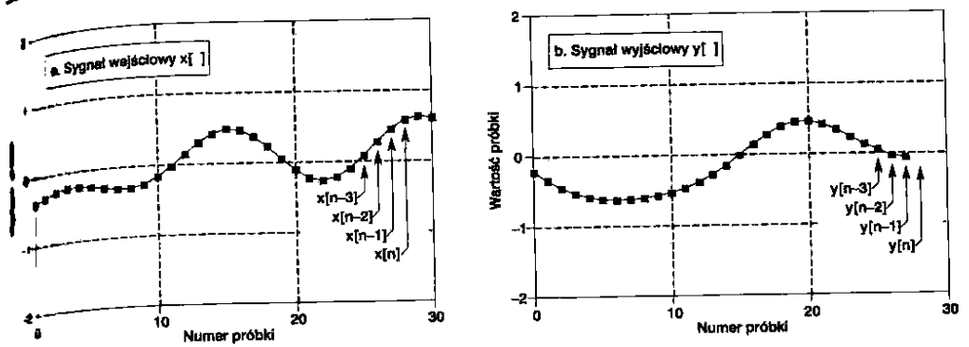
Łatwo rozpoznacie, że jest to nic innego, jak prosty splot, przy czym współczynniki a_0, a_1, a_2, ... tworzą w tym splocie odpowiedź impulsową. Jeśli to było wszystko, co robił profesor, to nie byłoby już niczego do dodania ani w tej historii, ani w tym rozdziale. Profesor miał jednak dostęp także do drugiego źródła informacji – do *poprzednio* obliczonych wartości sygnału, a więc $y[n-1]$, $y[n-2]$, $y[n-3]$, ... Przy wykorzystaniu tych dodatkowych informacji algorytm przyjmuje postać:

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + ...$$

$$b_1 y[n-1] + b_2 y[n-2] + b_3 y[n-3] + ...$$

Wzór 19.1. Wzór rekursywny, w którym x[ ] jest sygnałem wejściowym, y[ ] – wyjściowym, a, b – współczynnikami

Można więc powiedzieć, że każdy punkt w sygnale wyjściowym jest wyznaczany przez wymnażanie wartości sygnału wejściowego przez współczynniki „a" oraz wymnażanie poprzednio obliczonych wartości sygnału wyjściowego przez współczynniki „b" i następnie



Rys. 19.1. Zapis filtru rekursywnego. Obliczana próbka wyjściowa y[n] jest wyznaczana na podstawie wartości sygnału wejściowego x[n], x[n – 1], x[n – 2], ... , jak również poprzednio obliczonych wartości sygnału wyjściowego y[n – 1], y[n – 2], y[n – 3], ... Wykresy wykonano dla n = 28

sumowanie wszystkich uzyskanych iloczynów. Zauważcie, że nie ma tu wartości b_0, gdyż odpowiada ona próbce właśnie obliczanej. Wzór 19.1 jest nazywany **wzorem rekursywnym** a filtry obliczane przy jego użyciu – **filtrami rekursywnymi**. Wartości „a" i „b" określające filtr są nazywane **współczynnikami rekursywnymi**. W praktyce można stosować nie więcej niż kilkanaście współczynników, gdyż filtr mógłby stać się niestabilny (tj. sygnał wyjściowy byłby ciągle rosnący lub oscylujący). W tablicy 19.1 podano przykład programu filtru rekursywnego.

Filtry rekursywne są stosowane, gdyż *omija się* w nich dłuższy splot. Rozważmy na przykład, co się dzieje, gdy funkcja delta jest przepuszczana przez filtr rekursywny. Na wyjściu uzyskuje się wtedy *odpowiedź impulsową* filtru, która – typowo – jest oscylacją sinusoidalną zanikającą wykładniczo. Ponieważ odpowiedź impulsowa jest nieskończenie długa, więc filtry rekursywne są często nazywane filtrami o *nieskończonej odpowiedzi impulsowej* (NOI). Filtry rekursywne działają na zasadzie *splatania* sygnału wejściowego z bardzo długą odpowiedzią impulsową filtru, choć jest stosowanych niewiele współczynników.

Zależność między współczynnikami rekursywnymi i odpowiedzią impulsową filtru jest określona metodą matematyczną zwaną **przekształceniem Z**, które będzie tematem rozdziału 33. Przekształcenie Z może być stosowane do takich zadań jak: konwersja między współczynnikami rekursywnymi i odpowiedzią impulsową, łączenie stopni równoległych i kaskadowych w pojedynczy filtr, projektowanie systemów rekursywnych naśladujących filtry analogowe itd. Niestety przekształcenie Z jest trudne matematycznie i bardziej skomplikowane niż tego życzyłaby sobie większość użytkowników DSP. Jest to dziedzina dla tych, którzy specjalizują się w DSP.

Tab. 19.1

```
'FILTR REKURSYWNY
'
DIM X[499]                    'przechowuje sygnał wejściowy
DIM Y[499]                    'przechowuje odfiltrowany sygnał wyjściowy
'
GOSUB XXXX                    'podprogram "mityczny" do obliczania współczynników
'                             'rekursywnych: A0, A1, A2, B1, B2
'
GOSUB XXXX                    'podprogram "mityczny" do ładowania danych wejściowych do X[ ]
'
FOR I% = 2 TO 499
  Y[I%] = A0*X[I%] + A1*X[I%-1] + A2*X[I%-2] + B1*Y[I%-1] + B2*Y[I%-2]
NEXT I%
'
END
```

Są trzy sposoby określania współczynników rekursywnych bez potrzeby zrozumienia przekształcenia Z. Pierwszy sposób – podane w tym rozdziale wzory projektowe dla kilku typów prostych filtrów rekursywnych. Drugi sposób to zamieszczony w rozdziale 20, praktyczny program komputerowy typu *cookbook* do projektowania bardziej skomplikowanych dolnoprzepustowych filtrów *Czebyszewa*. Trzeci sposób opisano w rozdziale 26, jest to iteracyjna metoda projektowania filtrów rekursywnych o *dowolnej* odpowiedzi częstotliwościowej.

Jednobiegunowe filtry rekursywne

Na rys. 19.2 przedstawiono przykład tego, co nazywamy dolnoprzepustowym filtrem jednobiegunowym. W tym filtrze rekursywnym zastosowano tylko dwa współczynniki $a_0 = 0,15$ i $b_1 = 0,85$. W tym przykładzie sygnałem wejściowym jest funkcja skokowa. Jak można się tego spodziewać po filtrze dolnoprzepustowym, sygnał wyjściowy jest przebiegiem łagodnie narastającym do pewnego ustalonego poziomu. Na rysunku przedstawiono też układ znany z podstaw elektroniki. Ten dolnoprzepustowy filtr rekursywny jest bowiem w pełni równoważny z elektronicznym filtrem dolnoprzepustowym złożonym z rezystora i kondensatora.

Piękno metody rekursywnej polega na zdolności tworzenia wielkiej różnorodności odpowiedzi przez zmianę tylko kilku parametrów. Na przykład na rys. 19.3 pokazano filtr o trzech współczynnikach: $a_0 = 0,93$, $a_1 = -0,93$ oraz $b_1 = 0,86$. Jak wynika z podobieństwa odpowiedzi skokowych, ten filtr cyfrowy jest naśladownictwem elektronicznego górnoprzepustowego filtru RC.

Te jednobiegunowe filtry rekursywne z pewnością należą do podstawowego zestawu rozwiązań DSP. Można je stosować do przetwarzania sygnałów cyfrowych, podobnie jak obwody RC służą do przetwarzania elektronicznych sygnałów analogowych. To obejmuje wszystko, czego można by oczekiwać: usuwanie składowej stałej, tłumienie szumu w.cz., kształtowanie i wygładzanie przebiegów itd. Takie filtry są łatwe do programowania, szybkie w działaniu i dostarczają raczej niewielu przykrych niespodzianek. Współczynniki oblicza się z prostych wzorów:

$$a_0 = 1 - x$$
$$b_1 = x$$

Wzór 19.2. Dolnoprzepustowy filtr jednobiegunowy. Odpowiedź filtru jest określana parametrem x, o wartości między 0 i 1

$$a_0 = (1+x)/2$$
$$a_1 = -(1+x)/2$$
$$b_1 = x$$

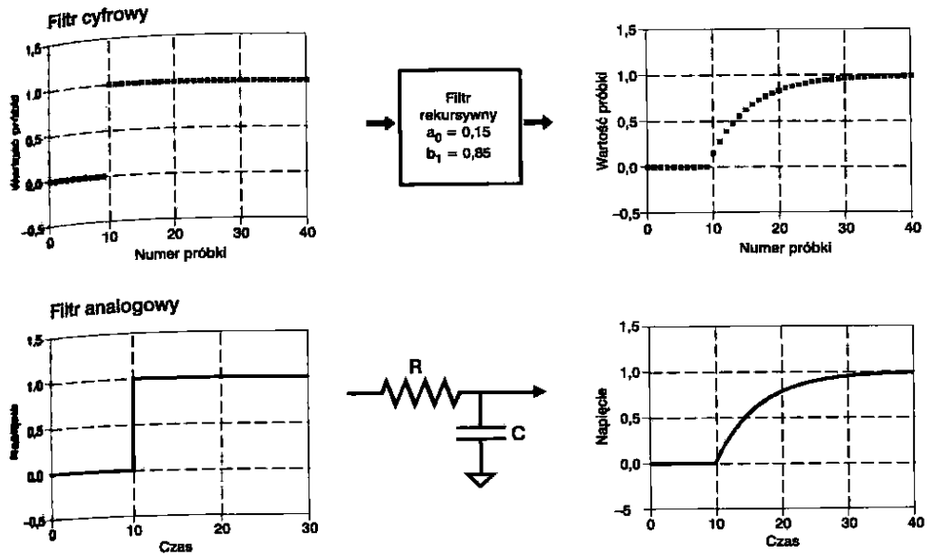
Wzór 19.3. Górnoprzepustowy filtr jednobiegunowy

Właściwości tych filtrów są ustalane przez parametr x, którego wartość mieści się między 0 i 1. Fizycznie x jest współczynnikiem wskazującym, jakie jest *opadanie* przebiegu mierzone między kolejnymi próbkami. Na przykład na rys. 19.3 parametr x jest równy 0,86, co znaczy, że wartość każdej następnej próbki w sygnale wyjściowym jest równa 0,86 wartości próbki poprzedniej. Czym większa jest wartość x, tym wolniejsze jest opadanie. Zauważmy, że filtr staje się *niestabilny*, jeśli x jest większe od jedności. Wtedy każda niezerowa wartość na wejściu powoduje wzrost sygnału wyjściowego aż do przekroczenia pełnego zakresu.

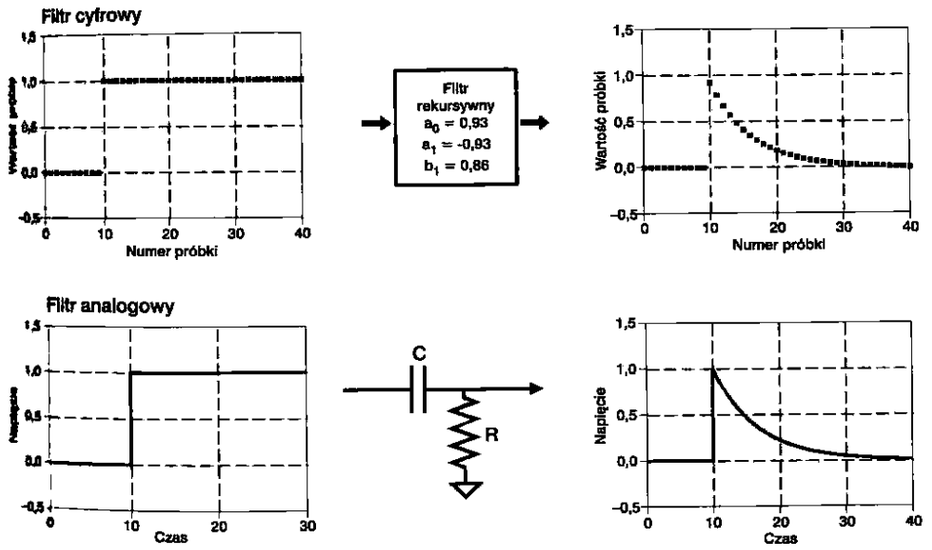
Wartość x może być bezpośrednio obliczona na podstawie wymaganej *stałej czasowej* filtru. Uwzględniając, że $R \times C$ jest wyrażonym w sekundach czasem opadania przebiegu na wyjściu obwodu RC do 36,8% wartości maksymalnej, a d jest liczbą próbek, podczas których sygnał z filtru rekursywnego opada do tego samego poziomu, mamy:

$$x = e^{-1/d}$$

Wzór 19.4. Stała czasowa filtrów jednobiegunowych. Ten wzór określa zależność między współczynnikiem x opadania przebiegu między kolejnymi próbkami a stałą czasową d będącą liczbą próbek, podczas których przebieg opada do 36,8% wartości maksymalnej



Rys. 19.2. Jednobiegunowy filtr dolnoprzepustowy. Rekursywne filtry cyfrowe mogą być naśladownictwem filtrów analogowych złożonych z rezystorów i kondensatorów. Jak pokazano w tym przykładzie, dolnoprzepustowy, jednobiegunowy filtr rekursywny w odpowiedzi na funkcję skokową na wejściu, podobnie jak elektroniczny filtr RC, daje na wyjściu przebieg łagodnie narastający



Rys. 19.3. Górnoprzepustowy filtr jednobiegunowy. Przy odpowiednim doborze współczynników filtr rekursywny może być naśladownictwem elektronicznego, górnoprzepustowego filtru RC. Takie jednobiegunowe filtry rekursywne mogą być stosowane w DSP, tak jak obwody RC w elektronice analogowej

Na przykład współczynnik opadania między sąsiednimi próbkami równy $x = 0{,}86$ odpowiada stałej czasowej $d = 6{,}63$ próbki (jak to pokazano na rys. 19.3). Istnieje ustalona zależność między x i –3-decybelową *częstotliwością odcięcia* f_c filtru cyfrowego:

$$x = e^{-2\pi f_c}$$

Wzór 19.5. Częstotliwość odcięcia filtrów jednobiegunowych. Jest to zależność między współczynnikiem x opadania przebiegu między sąsiednimi próbkami a częstotliwością odcięcia filtru f_c mieszczącą się w granicach od 0 do 0,5

W ten sposób mamy trzy sposoby wyznaczenia współczynników a i b – wychodząc ze stałej czasowej, częstotliwości odcięcia lub bezpośrednio wybierając x.

Na rys. 19.4 przedstawiono przykład zastosowania jednobiegunowego filtru rekursywnego. Na rys. 19.4 a) pokazano sygnał oryginalny będący gładką krzywą z nałożoną paczką przebiegu sinusoidalnego w.cz. Na rys. 19.4 b) pokazano ten przebieg po przejściu przez filtry dolnoprzepustowy i górnoprzepustowy. Sygnały zostały rozdzielone dość dobrze, choć nie idealnie – tak jak przy zastosowaniu prostego obwodu RC w technice analogowej.

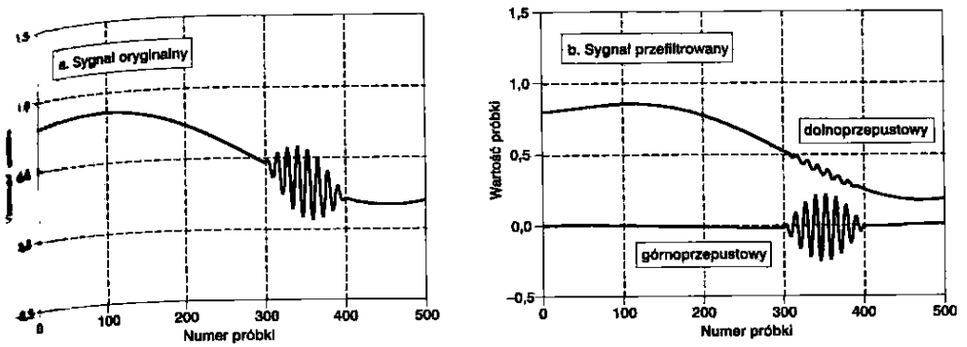
Na rys. 19.5 przedstawiono odpowiedzi częstotliwościowe różnych jednobiegunowych filtrów rekursywnych. Krzywe uzyskano, przepuszczając przez filtry funkcję delta w celu wyznaczenia odpowiedzi impulsowych. Następnie stosując FFT, przekształcono te odpowiedzi w odpowiedzi częstotliwościowe. Odpowiedź impulsowa jest, w zasadzie, nieskończenie długa, jednak po 15 do 20 stałych czasowych opada poniżej szumu błędu zaokrąglania przy pojedynczej dokładności. Na przykład, gdy stała czasowa filtru $d = 6{,}63$ próbki, to odpowiedź impulsowa może się zawierać w ok. 128 próbkach.

Najważniejszy wniosek, jaki można wyciągnąć na podstawie rys. 19.5, jest taki, że jednobiegunowe filtry rekursywne mają niewielką zdolność do rozdzielania jednego pasma częstotliwości od drugiego. Można więc powiedzieć, że dobrze działają w dziedzinie czasu a słabo w dziedzinie częstotliwości. Odpowiedź częstotliwościową można nieco poprawić, łącząc kaskadowo kilka stopni. Można to wykonać dwoma sposobami. Po pierwsze, można sygnał kilkakrotnie przepuścić przez filtr. Po drugie, można zastosować przekształcenie Z do wyznaczenia współczynników rekursywnych, które łączą kaskadę w pojedynczy stopień. Oba te sposoby dobrze funkcjonują i są często stosowane. Na rys. 19.5 c) pokazano odpowiedź częstotliwościową kaskady czterech filtrów dolnoprzepustowych. Chociaż osiągnięto zwiększenie tłumienia w pasie zaporowym, to jednak pozostała bardzo zła szybkość opadania charakterystyki. Jeśli chcecie uzyskać lepsze działanie w dziedzinie częstotliwości, zainteresujcie się filtrami Czebyszewa opisanymi w następnym rozdziale.

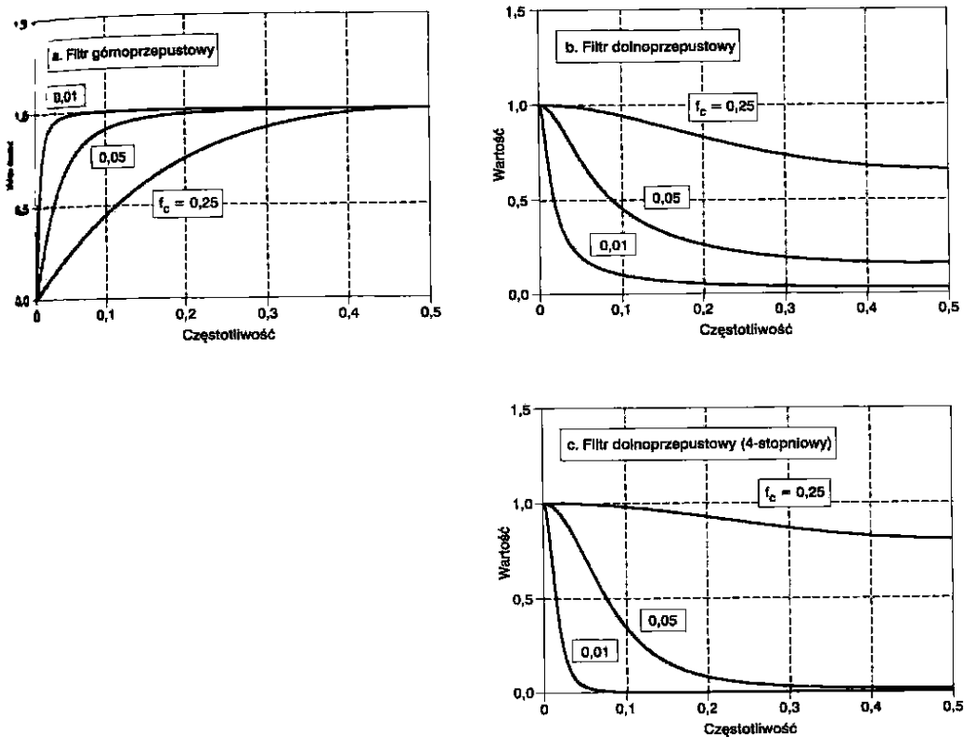
Filtr czterostopniowy jest porównywalny z filtrami Blackmana i gaussowskim (pokrewnymi filtrowi o średniej kroczącej, rozdział 15), lecz ma znacznie większą szybkość działania. Wzory do projektowania czterostopniowego filtru dolnoprzepustowego są następujące:

$$a_0 = (1-x)^4$$
$$b_1 = 4x$$
$$b_2 = -6x^2$$
$$b_3 = 4x^3$$
$$b_4 = -x^4$$

Wzór 19.6. Czterostopniowy filtr dolnoprzepustowy. Z wzorów można obliczyć współczynniki a i b dla kaskadowego, czterostopniowego, jednobiegunowego filtru dolnoprzepustowego. Zależność między x i częstotliwością odcięcia w takim filtrze jest dana wzorem 19.5, lecz 2π należy zamienić na 14,445



Rys. 19.4. Przykład jednobiegunowych filtrów rekursywnych. Na rys. a) na wolno zmieniającym się sygnale oryginalnym występuje paczka sinusoid w.cz. Na rys. b) pokazano rezultaty zastosowania jednobiegunowych filtrów górno- i dolnoprzepustowego do rozdzielenia obu składowych. W filtrze dolnoprzepustowym zastosowano x = 0,95, a w górnoprzepustowym x = 0,86



Rys. 19.5. Odpowiedzi filtrów jednobiegunowych. Na rys. a), b) pokazano odpowiedzi częstotliwościowe jednobiegunowych filtrów rekursywnych – górno- i dolnoprzepustowego, a na rys. c) – kaskady czterech filtrów dolnoprzepustowych. Odpowiedzi częstotliwościowe filtrów rekursywnych nie zawsze są zgodne z oczekiwaniami użytkowników, zwłaszcza jeśli w filtrze chce się osiągnąć skrajne wartości parametrów. Na przykład krzywa dla $f_c = 0{,}25$ na rys. c) jest zupełnie nieprzydatna w praktyce. Winić za to można wiele czynników, takich jak aliasing, szum zaokrąglania i nieliniowa odpowiedź fazowa

Filtry wąskopasmowe

W elektronice i DSP często powstaje potrzeba wyodrębnienia wąskiego pasma częstotliwości z sygnału o paśmie szerszym. Na przykład możemy chcieć wyeliminować w systemie pomiarowym zakłócenia 50 Hz lub odizolować ton sygnalizacji w sieci telefonicznej. Są możliwe dwa rodzaje odpowiedzi częstotliwościowej: *pasmowoprzepustowa* i *pasmowozaporowa* (zwana też filtrem wycinającym – *notch filter*). Na rys. 19.6 przedstawiono odpowiedzi częstotliwościowe tych filtrów, ze współczynnikami rekursywnymi obliczonymi według poniższych wzorów:

$$a_0 = 1 - K$$
$$a_1 = 2(K - R)\cos(2\pi f)$$
$$a_2 = R^2 - K$$
$$b_1 = 2R\cos(2\pi f)$$
$$b_2 = -R^2$$

Wzór 19.7. Filtr pasmowoprzepustowy. Przykład odpowiedzi częstotliwościowych podano na rys. 19.6 a) Aby skorzystać z tych wzorów, trzeba najpierw wybrać częstotliwość środkową f i szerokość pasma BW. Obie te wielkości są wyrażone jako ułamki częstotliwości próbkowania, mieszczą się więc w zakresie od 0 do 0,5. Następnie oblicza się R i K, a potem współczynniki rekursywne

$$a_0 = K$$
$$a_1 = -2K\cos(2\pi f)$$
$$a_2 = K$$
$$b_1 = 2R\cos(2\pi f)$$
$$b_2 = -R^2$$

gdzie:

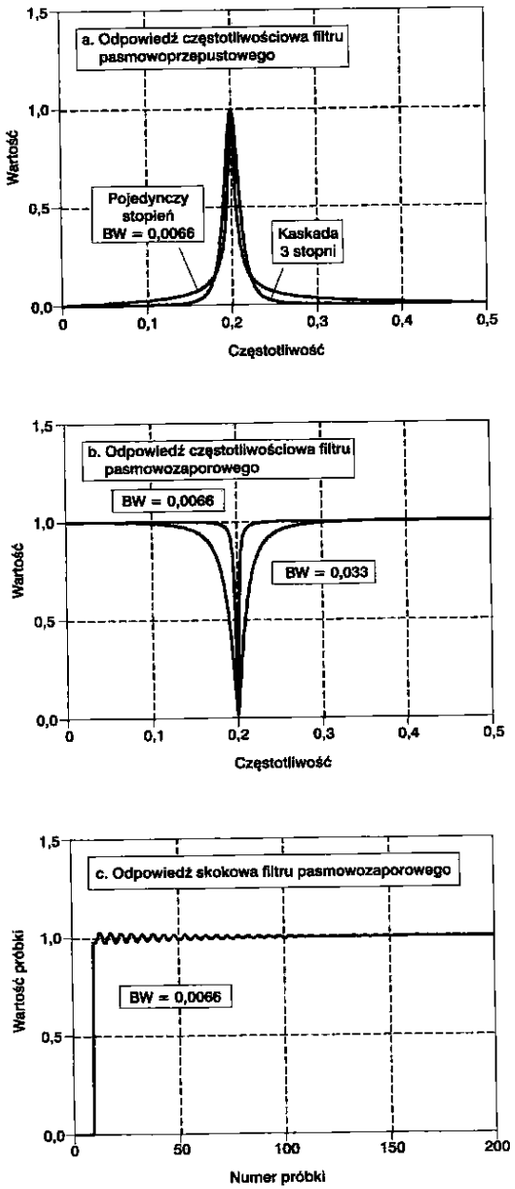
$$K = \frac{1 - 2R\cos(2\pi f) + R^2}{2 - 2\cos(2\pi f)}$$
$$R = 1 - 3BW$$

Wzór 19.8. Filtr pasmowozaporowy, zwany też filtrem wycinającym. Przykład odpowiedzi częstotliwościowych podano na rys. 19.6 b)

Przed zastosowaniem tych wzorów trzeba wybrać wartości dwóch parametrów: f – częstotliwość środkową oraz B – szerokość pasma mierzoną przy 0,707 amplitudy. Obie te wielkości są wyrażane jako ułamki częstotliwości próbkowania, a zatem ich wartości muszą się mieścić w przedziale od 0 do 0,5. Na podstawie tych dwóch wybranych wartości trzeba obliczyć zmienne pośrednie R i K i następnie współczynniki rekursywne.

Jak pokazano na rys. 19.6 a), przebieg odpowiedzi filtru pasmowoprzepustowego ma dość duże „*ogony*" odchodzące od głównego piku. Można to poprawić, łącząc kilka stopni kaskadowo. Ponieważ wzory obliczeniowe są dość długie, więc łatwiej jest zrealizować kaskadę raczej przez kilkakrotną filtracje sygnału niż przez próby obliczania współczynników potrzebnych dla filtru pojedynczego.

Na rys. 19.6 b) podano przykłady filtrów pasmowozaporowych. Najwęższe pasmo osiągalne w nich przy pojedynczej dokładności wynosi ok. 0,0003 częstotliwości próbkowania. Gdy wyjdzie się poza tę granicę, następuje pogorszenie tłumienia w paśmie zaporowym. Na rys. 19.6 c) pokazano odpowiedź skokową filtru pasmowozaporowego. Widać wyraźny przeregulowanie i oscylacje (tzw. dzwonienia), lecz ich amplituda jest niewielka. Dzięki temu filtr może usuwać zakłócenia wąskopasmowe (50 Hz i podobne), powodując tylko niewielkie zniekształcenia przebiegu w dziedzinie czasu.



Rys. 19.6. Charakterystyki filtrów wąskopasmowych. Na rys. a), b) pokazano odpowiedzi częstotliwościowe różnych filtrów pasmowoprzepustowych i pasmowozaporowych. Odpowiedź skokową filtru pasmowozaporowego przedstawiono na rys. c). Filtr pasmowozaporowy jest przydatny do usuwania zakłóceń 50 Hz i podobnych z przebiegów kodowanych w dziedzinie czasu

Odpowiedź fazowa

Filtr może mieć odpowiedź fazową jednego z trzech rodzajów: **faza zerowa**, **faza liniowa** i **faza nieliniowa**. Przykłady każdej z tych odpowiedzi zamieszczono na rys. 19.7. Filtr o *fazie zerowej* charakteryzuje się odpowiedzią impulsową symetryczną wokół próbki zero (rys. 19.7 a). Kształt odpowiedzi nie ma znaczenia; chodzi tylko o to, żeby próbki o numerach ujemnej były lustrzanym obiciem próbek o numeracji dodatniej. Gdy dokona się przetwarzania Fouriera takiego przebiegu symetrycznego, uzyska się przebieg fazy całkowicie zerowy (rys. 19.7 b).

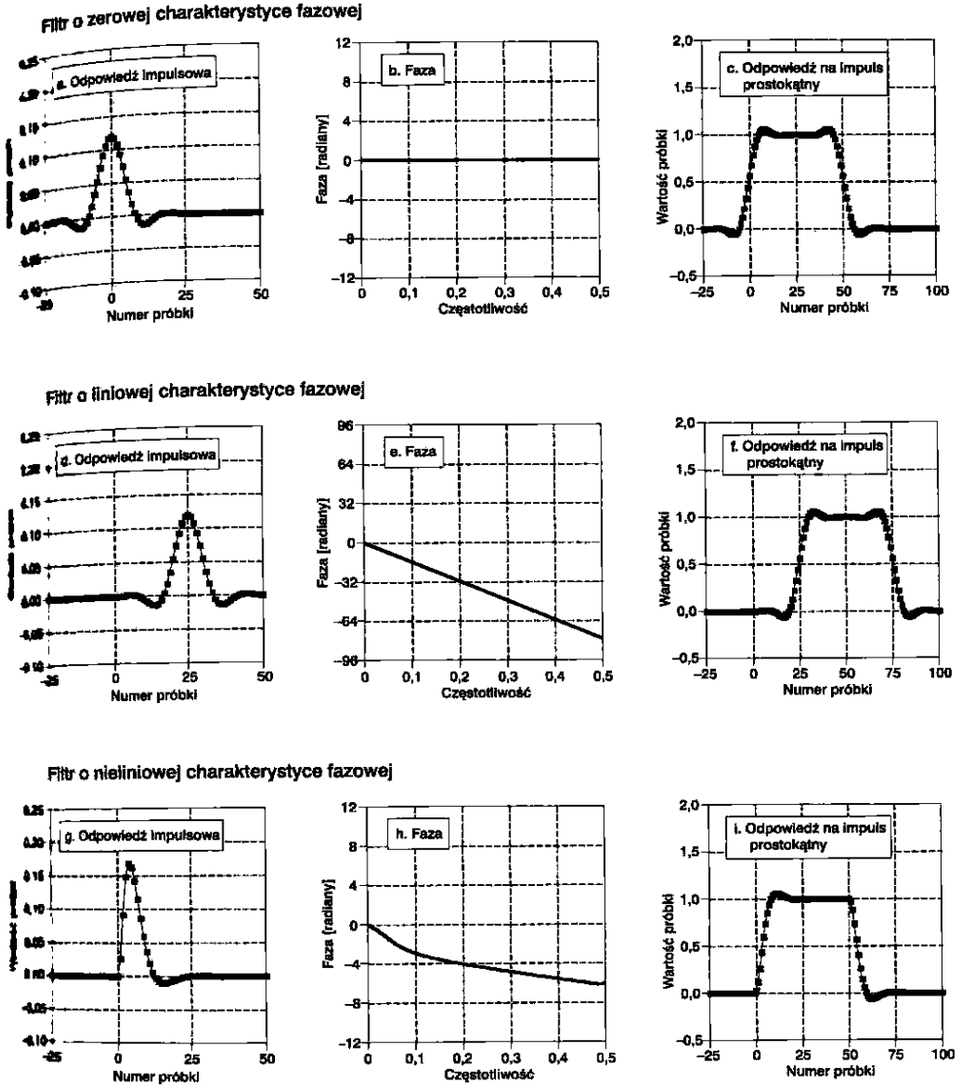
Wadą filtru o fazie zerowej jest konieczność użycia ujemnych indeksów, którymi posługiwanie się może być niewygodne. Można tego uniknąć, stosując filtr o fazie liniowej. Jego odpowiedź impulsowa (rys. 19.7 d) jest identyczna z pokazaną na rys. 19.7 a), lecz przesunięta tak, aby można było stosować tylko próbki o numeracji dodatniej. Ta odpowiedź impulsowa jest nadal symetryczna lewo-prawostronnie, lecz oś symetrii jest przesunięta w stosunku do zera. To przesunięcie powoduje, że charakterystyka fazowa jest *linią prostą* (rys. 19.7 e), dlatego mówimy o *fazie liniowej*. Nachylenie tej linii prostej jest wprost proporcjonalne do wartości przesunięcia odpowiedzi impulsowej. Ponieważ to przesunięcie nie powoduje niczego poza identycznym przesunięciem w sygnale wyjściowym, więc filtr o fazie liniowej jest w większości zastosowań równoważny filtrowi o fazie zerowej.

Odpowiedź impulsowa pokazana na rys. 19.7 g) *nie jest* lewo-prawostronnie symetryczna. Odpowiednio do tego, przebieg fazy (rys.19.7 h) *nie jest* linią prostą. Mamy tu więc fazę *nieliniową*. Nie pomylcie pojęć *nieliniowa i liniowa faza* z koncepcją *liniowości systemu* rozważaną w rozdziale 5. Chociaż w obu tych pojęciach występuje słowo *liniowy*, to jednak nie ma między nimi żadnego związku.

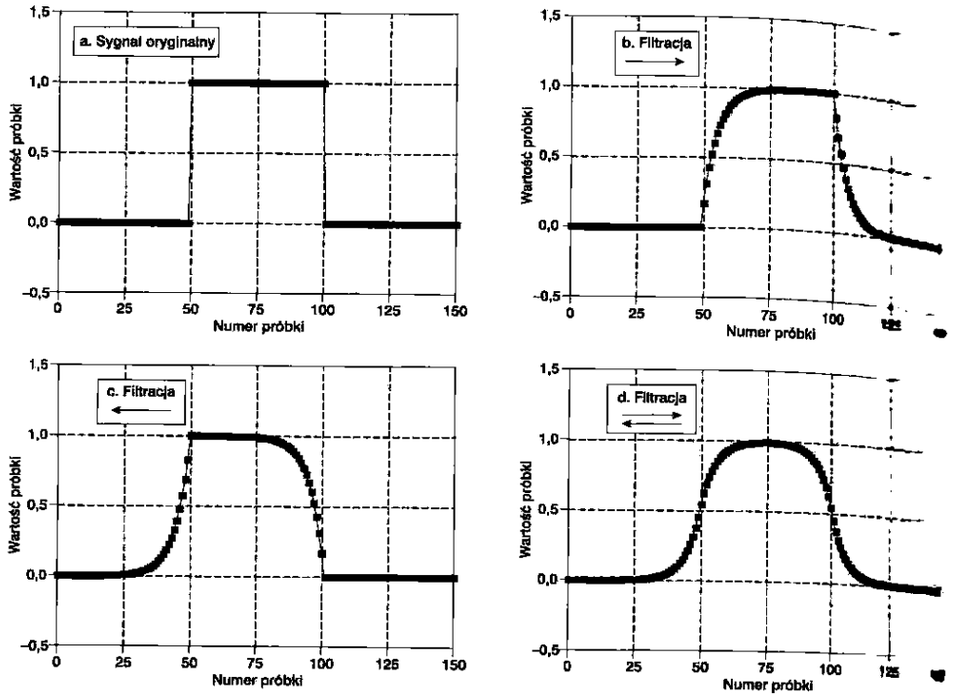
Dlaczego zastanawiamy się nad tym, czy charakterystyka fazy jest liniowa czy nieliniowa? Odpowiedź na to pytanie wynika z rys. 19.7 c), f), i). Na tych rysunkach przedstawiono **odpowiedzi na impuls prostokątny** każdego z trzech filtrów. Odpowiedzi te są niczym innym, jak odpowiedzią skokową o zboczu dodatnim, po której następuje odpowiedź skokowa o zboczu ujemnym. Pokazano tu odpowiedzi na impuls prostokątny, gdyż one obrazują, co dzieje się zarówno z narastającym, jak i opadającym zboczem sygnału. A teraz ważna informacja: filtry o fazie zerowej lub liniowej dają zbocze lewe i prawe wyglądające *jednakowo*, podczas gdy w filtrach o fazie nieliniowej te zbocza są *różne*. W wielu zastosowaniach taka różnica jest niedopuszczalna. Jednym z przykładów jest wyświetlacz oscyloskopu, gdzie ta różnica mogłaby być błędnie interpretowana jako cecha mierzonego sygnału. Innym przykładem jest przetwarzanie sygnału wizyjnego. Czy możesz sobie wyobrazić, że włączasz telewizor i widzisz, że jedno ucho ulubionego aktora różni się od drugiego?

Łatwo można zrealizować filtr o skończonej odpowiedzi impulsowej (SOI) mający liniową fazę. Odpowiedź impulsowa filtru SOI jest bowiem *określana* bezpośrednio w procesie projektowania. Konieczne jest tylko spowodowanie, aby ta odpowiedź miała symetrię lewo-prawostronną. Inna sytuacja jest z filtrami o nieskończonej odpowiedzi impulsowej (NOI), czyli rekursywnymi, gdyż w nich są określane współczynniki rekursywne, a nie odpowiedź impulsowa. Odpowiedź impulsowa filtru rekursywnego *nie jest* symetryczna lewo-prawostronnie, a więc ma fazę *nieliniową*.

Takie same problemy z odpowiedzią fazową występują w elektronicznych układach analogowych. Wyobraźcie sobie, że na swoim stole laboratoryjnym macie układ złożony z rezystorów i kondensatorów. Jeśli na wejściu układu jest zawsze zero, to i na wyjściu będzie zero. Gdy do wejścia zostanie doprowadzony impuls, kondensatory szybko się ładują do pewnej wartości, a potem zaczynają się rozładowywać wykładniczo przez rezystory. Odpowiedź impulsowa (tj. sygnał wyjściowy) jest kombinacją tych różnych zanikających przebiegów wykładniczych. Ta odpowiedź impulsowa *nie może* być symetryczna, ponieważ sygnał



Rys. 19.7. Filtry o zerowej, liniowej i nieliniowej charakterystyce fazowej. Filtr o fazie zerowej ma odpowiedź impulsową, która jest lewo-prawostronnie symetryczna wokół próbki zero (rys. a). To daje odpowiedź częstotliwościową, o której przebieg fazy zawiera same zera (rys. b). Odpowiedzi impulsowe o fazie zerowej są korzystne, gdyż w tym przypadku odpowiedzi skokowe mają symetrię góra–dół, dzięki czemu lewe i prawe zbocze impulsu wyglądają identycznie (rys. c). Filtry o fazie liniowej mają symetrię lewo-prawostronną, lecz nie wokół próbki zerowej (rys. 19.7 d). Daje to charakterystykę fazową, która jest linią prostą (rys. 19.7 e). Odpowiedź filtru o fazie liniowej na impuls prostokątny, pokazana na rys. f), ma wszelkie zalety takie, jak odpowiedź o fazie zerowej. Natomiast odpowiedzi impulsowe filtrów o fazie nieliniowej nie są symetryczne lewo-prawostronnie (rys. g), a przebieg fazy nie jest linią prostą (rys. h). Najgorszą cechą filtrów o fazie nieliniowej jest niejednakowość lewego i prawego zbocza odpowiedzi na impuls prostokątny (rys. 19.7 i)



Rys. 19.8. Dwukierunkowa filtracja rekursywna. Prostokątny impuls wejściowy pokazano na rys. a). Na rys. b) przedstawiono ten sygnał po przefiltrowaniu jednobiegunowym, dolnoprzepustowym filtrem rekursywnym, przy obliczaniu próbek od lewej do prawej. Sygnał na rys. c) został przetworzony w ten sam sposób, lecz implementacja filtru następowała od prawej ku lewej stronie. Sygnał przedstawiony na rys. d) został przefiltrowany zarówno od lewej do prawej, jak i od prawej do lewej. Stosując tę technikę, można każdy filtr rekursywny uczynić zerofazowym

wyjściowy był równy zero przed impulsem, a przebieg wykładniczy nigdy nie powraca całkiem do zera. W filtrach analogowych rozwiązuje się ten problem, stosując **filtr Bessela** omówiony w rozdziale 3. Filtr Bessela jest zaprojektowany w taki sposób, żeby charakterystyka fazy była jak najbardziej liniowa. Jednak jakość działania tego filtru jest daleko poniżej jakości filtrów cyfrowych. Możliwość realizacji *dokładnie* liniowej charakterystyki fazowej jest wyraźną zaletą filtrów cyfrowych.

Jest na szczęście prosty sposób modyfikacji filtrów rekursywnych tak, aby uzyskać *fazę zerową*. Na rys. 19.8 przedstawiono przykład działania takiego filtru. Na rys. 19.8 a) pokazano sygnał, który ma być filtrowany, a na rys. 19.8 b) – sygnał po przefiltrowaniu w jednobiegunowym filtrze dolnoprzepustowym. Ponieważ jest to filtr o fazie nieliniowej, więc prawe i lewe zbocze nie wyglądają jednakowo, mają kształty odwrócone względem siebie. Jak już poprzednio powiedziano, ten filtr rekursywny jest implementowany, rozpoczynając od próbki 0 zero i dochodząc aż do próbki 150, z kolejnym obliczeniem każdej próbki między tymi wartościami.

Teraz przypuśćmy, że zamiast poruszać się od próbki 0 do 150, zaczynamy od próbki 150 i posuwamy się w kierunku próbki 0. Inaczej mówiąc, każda próbka w sygnale wyjściowym jest obliczana na podstawie próbek wejściowych i wyjściowych leżących *na prawo* od próbki właśnie obliczanej. Wynika stąd, że wzór rekursywny 19.1 został zamieniony na wzór:

$$y[n]=a_0x[n]+a_1x[n+1]+a_2x[n+2]+a_3x[n+3]+\ldots$$
$$+b_1y[n+1]+b_2y[n+2]+b_3y[n+3]+\ldots$$

Wzór 19.9. Odwrotny wzór rekursywny. Jest on identyczny ze wzorem 19.1, z tym że sygnał jest filtrowany od lewej do prawej strony, zamiast od prawej do lewej

Na rys. 19.8 c) pokazano wynik takiej **odwrotnej filtracji**. Jest to analogiczne do przejścia sygnału analogowego przez układ RC założeniem, że czas biegnie wstecz. – !taiwśhcezsw ćąniwz eżom usazc einecórwdo agawU

Filtracja w odwrotnym kierunku, sama w sobie, nie daje żadnych korzyści; przefiltrowany sygnał nadal ma zbocza lewe i prawe różniące się od siebie. Cudowny rezultat uzyskuje się dopiero, stosując połączoną filtrację prostą i odwrotną. Na rys. 19.8 d) pokazano wynik kolejno wykonanych filtracji sygnału – prostej i odwrotnej. I oto mamy filtr rekursywny *o fazie zerowej*. W istocie, *każdy* filtr rekursywny może być przekształcony w zerofazowy przy użyciu tej techniki **dwukierunkowej filtracji**. Jedyną ceną, jaką płaci się za takie udoskonalenie działania, jest zwiększenie czasu wykonania i złożoności oprogramowania o czynnik 2.

Jak znaleźć odpowiedzi impulsową i częstotliwościową takiego złożonego filtru? Moduł odpowiedzi częstotliwościowej jest taki sam przy każdym kierunku filtracji, a fazy mają odwrotne znaki. Gdy połączymy ze sobą filtracje w obu kierunkach, to moduł trzeba *podnieść do kwadratu*, a fazy się znoszą, dając w wyniku *zero*. W dziedzinie czasu odpowiada to splotowi oryginalnej odpowiedzi impulsowej z jej wersją odwróconą lewo-prawostronnie. Na przykład odpowiedź impulsowa dolnoprzepustowego filtru jednobiegunowego jest jednostronną funkcją wykładniczą. Odpowiedź impulsowa równoważnego filtru dwukierunkowego jest splotem jednostronnej funkcji wykładniczej malejącej ku prawej stronie z taką samą funkcją malejącą w lewo. W wyniku operacji matematycznych uzyskuje się wynik tego splotu, którym jest dwustronna funkcja wykładnicza malejąca z obu stron według stałej czasowej takiej, jak filtru pierwotnego.

W niektórych zastosowaniach tylko część sygnału znajduje się w komputerze w danej chwili. Jest tak np. w systemach, które przemiennie pobierają i przekazują dane na zasadzie kontynuacji. W takich przypadkach można realizować filtrację dwukierunkową przez łączenie jej z metodą nakładania z dodawaniem opisaną w poprzednim rozdziale. Gdy dochodzicie do kwestii długości odpowiedzi impulsowej, nie mówcie, że jest ona „nieskończona". Gdybyście założyli długość nieskończenie dużą, musielibyście dopełniać każdy segment sygnału o *nieskończoną* liczbę zer. Pamiętajcie, że można obciąć odpowiedź impulsową, gdy zmniejszy się ona poniżej poziomu szumu zaokrąglania, tj. po 15 do 20 stałych czasowych. Każdy segment wymaga dopełnienia zerami zarówno z lewej, jak i z prawej strony, aby umożliwić rozwinięcie podczas filtracji dwukierunkowej.

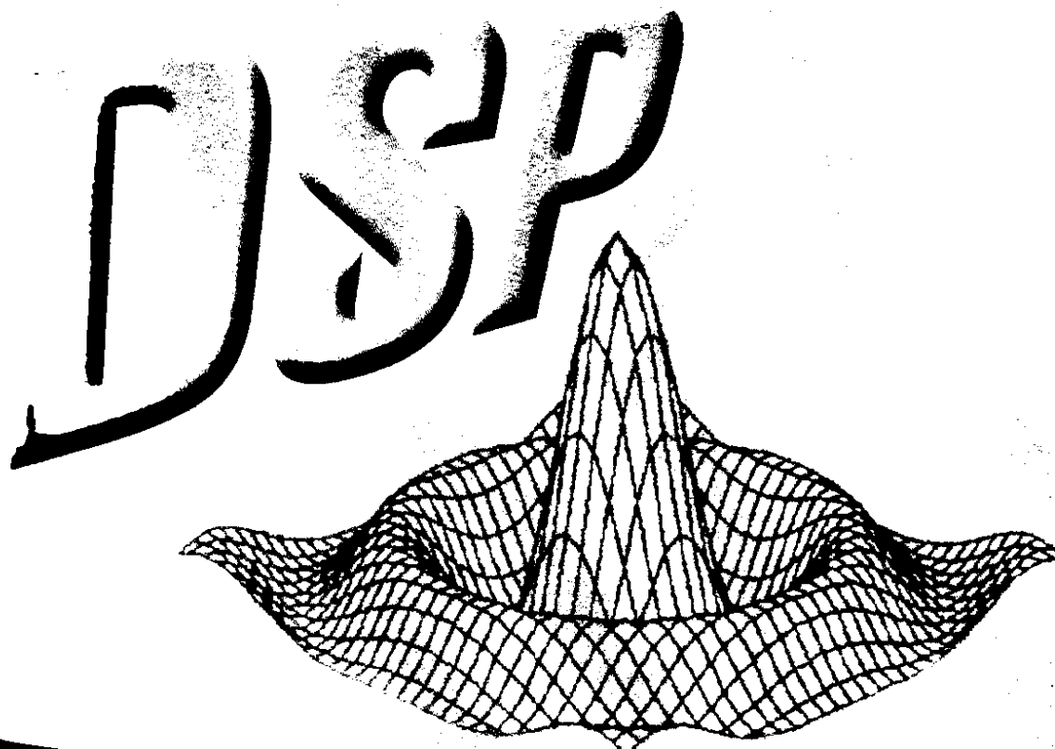
Stosowanie liczb całkowitych

Zapis zmiennoprzecinkowy o pojedynczej dokładności doskonale nadaje się do implementacji tych prostych filtrów rekursywnych. Zastosowanie liczb całkowitych jest możliwe, lecz znacznie trudniejsze. Pojawiają się wtedy dwie główne trudności. Po pierwsze, błąd zaokrąglenia przy ograniczonej liczbie bitów może pogorszyć odpowiedź filtru lub nawet wywołać jej niestabilność. Po drugie, trzeba przy użyciu liczb całkowitych operować ułamkowymi wartościami współczynników rekursywnych. Jednym ze sposobów przezwyciężenia tych trudności jest wyrażanie każdego współczynnika w postaci stosunku dwóch liczb. Na przykład 0,15 zamienia się na 19/128. Zamiast mnożyć przez 0,15, najpierw mnożymy przez 19, a potem dzielimy przez 128. Innym sposobem jest zastąpienie mnożenia tablicami przeglądowymi. Na przykład 12-bitowy przetwornik a/c daje próbki o wartościach między 0 a 4095. Zamiast wymnażania wszystkich próbek przez 0,15, przepuszczamy je przez tablicę

przeglądową zawierającą 4096 wpisów. Wartość uzyskana z tablicy przeglądowej jest równa wartości wchodzącej do tablicy pomnożonej przez 0,15. Ta metoda jest bardzo szybka, lecz wymaga dodatkowej pamięci; oddzielna tablica przeglądowa jest niezbędna dla każdego współczynnika. Zanim wypróbujecie któryś z tych sposobów operowania na liczbach całkowitych, upewnijcie się, czy waszych wymagań nie spełni rekursywny algorytm dla filtru o średniej kroczącej. Tamten algorytm *kocha* liczby całkowite.

20

Filtry Czebyszewa



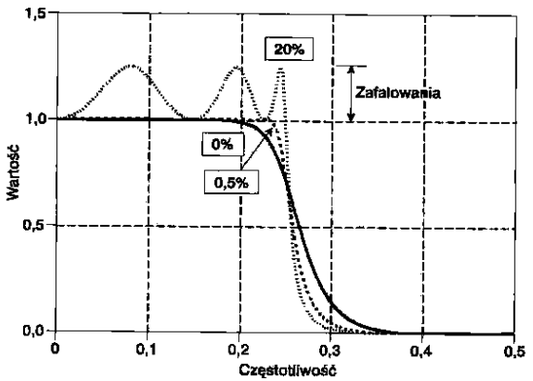
Filtry Czebyszewa są stosowane do rozdzielania pasm częstotliwości. Wprawdzie nie dorównują swym działaniem filtrom okienkowanym funkcją *sinc*, lecz w wielu zastosowaniach są wystarczające. Ich podstawową cechą jest szybkość, typowo o ponad rząd wielkości większa niż filtrów okienkowanych funkcją *sinc*. Wynika to z faktu, że filtry Czebyszewa są realizowane przez *rekursję*, a nie przez *splot*. W projektowaniu tych filtrów korzysta się z opisanej w rozdziale 33 metody matematycznej zwanej *przekształceniem* Z. W tym rozdziale zawarto informacje konieczne do *stosowania* filtrów Czebyszewa, bez wchodzenia w labirynt wyższej matematyki.

Odpowiedzi Czebyszewa i Butterwortha

Odpowiedź Czebyszewa jest narzędziem matematycznym służącym do osiągnięcia szybszego *opadania charakterystyki* przez dopuszczenie do *zafalowań* na odpowiedzi częstotliwościowej. Filtry analogowe i cyfrowe, w których zastosowano takie podejście do problemu, nazywane są filtrami Czebyszewa. Analogowe filtry Czebyszewa były użyte np. w rozdziale 3 przy przetwarzaniu analogowo-cyfrowym i cyfrowo-analogowym. Nazwa filtrów pochodzi od wielomianów *Czebyszewa*, które opracował rosyjski matematyk Pafnuti Czebyszew (1821–1894). W zachodniej literaturze technicznej spotyka się różne transkrypcje tego nazwiska, najczęściej Chebyshev.

Na rys. 20.1 przedstawiono odpowiedź częstotliwościową dolnoprzepustowego filtru Czebyszewa o zafalowaniach w paśmie przepustowym wynoszących 0%, 0,5% i 20%. Gdy zafalowania się zwiększają (to źle), to opadanie charakterystyki w paśmie przejściowym staje się szybsze (to dobrze). Odpowiedź Czebyszewa jest rozwiązaniem dającym optymalny kompromis między tymi dwoma parametrami. Jeśli zafalowania są ustalone na zero, to mamy filtr o charakterystyce **maksymalnie płaskiej**, zwany też **filtrem Butterwortha** (od nazwiska S. Butterwortha, brytyjskiego inżyniera, który opisał tę odpowiedź filtru w roku 1930). Dobrym wyborem w filtrach cyfrowych są często zafalowania równe 0,5%. Jest to wartość dopasowana do typowej dokładności układów elektroniki analogowej, przez które przeszedł sygnał.

Omówione w tym rozdziale filtry Czebyszewa są nazywane filtrami **typu 1**, co znaczy, że zafalowania odpowiedzi są dopuszczalne tylko w paśmie *przepustowym*. Natomiast filtry Czebyszewa **typu 2** mają zafalowania tylko w paśmie *zaporowym*. Filtry typu 2 są rzadko stosowane i nie będziemy ich rozpatrywać. Jest jednak pewne ważne rozwiązanie zwane **filtrem eliptycznym**. Ma



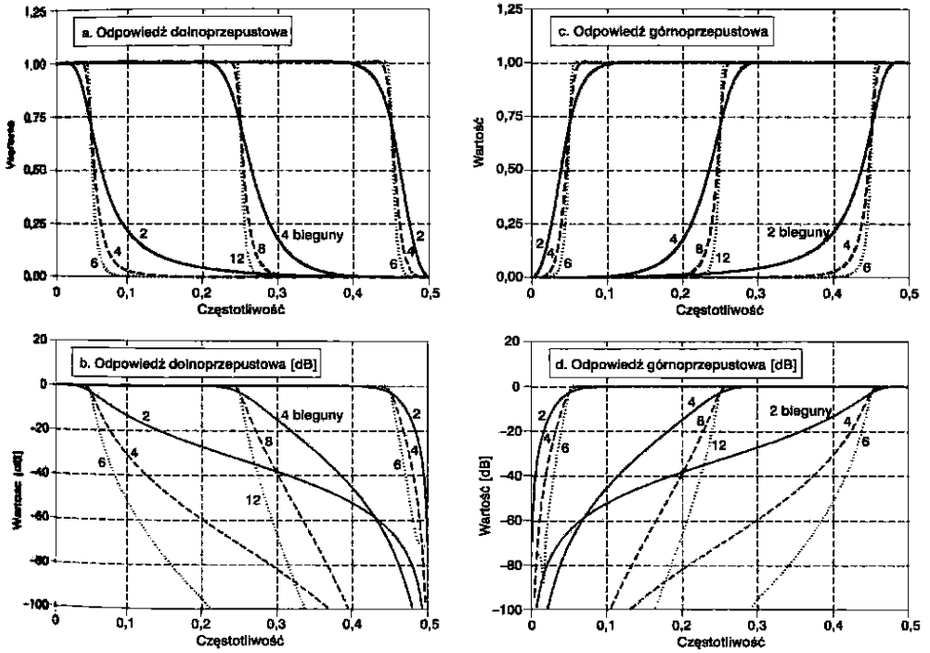
Rys. 20.1. Odpowiedź Czebyszewa. W filtrach Czebyszewa uzyskuje się szybsze opadanie charakterystyki dzięki dopuszczeniu do zafalowań w paśmie przepustowym. W przypadku, gdy zafalowania są ustalone na zero, filtr jest nazywany filtrem o maksymalnie płaskiej charakterystyce lub Butterwortha. W naszych projektach przyjmijcie zafalowania 0,5%; ten brak płaskości odpowiedzi jest tak mały, że pozostaje niezauważalny na wykresie, lecz szybkość opadania charakterystyki jest wtedy znacznie większa niż w filtrze Butterwortha

on zafalowania charakterystyki *tak* w paśmie przepustowym, *jak i* zaporowym. Filtry eliptyczne dają najszybsze opadanie charakterystyki przy danej liczbie biegunów, lecz ich projektowanie jest znacznie trudniejsze. Nie będziemy tu rozważać filtrów eliptycznych, lecz trzeba wiedzieć, że bywają one czasem rozwiązaniem najchętniej wybieranym przez profesjonalnych projektantów zarówno w elektronice analogowej, jak i w DSP. Jeśli jest wam potrzebny filtr o takim poziomie właściwości, kupcie pakiet oprogramowania do projektowania filtrów cyfrowych.

Projektowanie filtru

Przystępując do projektowania filtru Czebyszewa trzeba wybrać cztery parametry: 1) odpowiedź dolno- lub górnoprzepustową, 2) częstotliwość odcięcia, 3) procentową wartość zafalowań w paśmie przepustowym, 4) liczbę biegunów. A co to jest *biegun*? Są na to dwie odpowiedzi. Jeśli jedna wam się nie spodoba, to może druga wam się przyda.

Odpowiedź 1: Przekształcenie Laplace'a i przekształcenie Z są matematycznymi sposobami rozkładu odpowiedzi impulsowej na sinusoidy i opadające przebiegi wykładnicze. Jest to wykonywane przez wyrażenie charakterystyki systemu jako jednego wielomianu zespolonego podzielonego przez inny wielomian zespolony. Pierwiastki licznika takiego ilorazu noszą nazwę *zer*, a pierwiastki mianownika – *biegunów*. Zera i bieguny mogą być liczbami zespolonymi, dlatego przyjęto mówić, że mają „jakieś położenie" na płaszczyźnie zespolonej. Systemy rozbudowane mają więcej zer i biegunów niż systemy proste. Filtry rekursywne projektuje się, wybierając najpierw położenia zer i biegunów i wyznaczając następnie właściwe wartości współczynników rekursywnych (lub podzespołów analogowych). Na przykład filtry Butterwortha mają bieguny leżące na *okręgu* na płaszczyźnie zespolonej, a w filtrze Czebyszewa leżą one na *elipsie*. Ta kwestia jest tematem rozdziałów 32 i 33.



Rys. 20.2. Odpowiedzi częstotliwościowe Czebyszewa. Na rys. a), b) pokazano odpowiedzi częstotliwościowe dolnoprzepustowych filtrów Czebyszewa o zafalowaniach 0,5%, a na rys. c), d) takie odpowiedzi filtru górnoprzepustowego

Odpowiedź 2: Bieguny są pojemnikami wypełnionymi magicznym proszkiem. Czym biegunów w filtrze, tym lepiej on działa.

To tylko żart mający zwrócić uwagę na fakt, że możecie z powodzeniem stosować te nie wgłębiając się w trudną matematykę z nimi związaną. Projektowanie filtrów specjalizacji. W codziennej praktyce większość inżynierów, naukowców i programi przychyla się raczej ku odpowiedzi 2 niż 1.

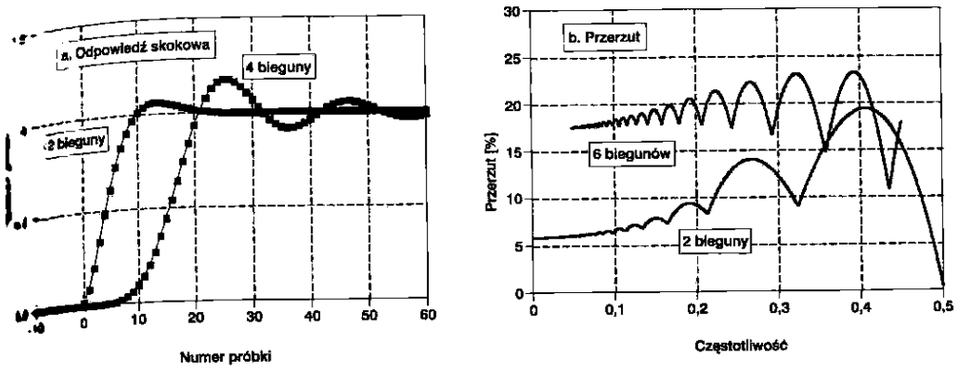
Na rys. 20.2 przedstawiono odpowiedzi częstotliwościowe kilku filtrów Czeby o zafalowaniach 0,5%. W metodzie tu stosowanej liczba biegunów musi być Częstotliwość odcięcia każdego filtru jest mierzona w punkcie opadania charakterystyki 0,707 (–3 dB) wartości maksymalnej. Filtry o częstotliwości odcięcia leżących w 0 lub 0,5 częstotliwości próbkowania mają ostrzejsze opadanie charakterystyki niż o częstotliwości odcięcia ze środka zakresu. Na przykład filtr dwubiegunowy ma dla tliwości $f_c = 0{,}05$ mniej więcej taką samą szybkość opadania jak filtr czterobiegunowy $f_c = 0{,}25$. To jest korzystne; w pobliżu wartości 0 i 0,5 można zastosować mniej bieg ze względu na szum zaokrąglenia. Później powiemy o tym więcej. Są dwie metody czenia współczynników rekursywnych bez używania przekształcenia Z. Pierwsza, dla tchórzy – skorzystanie z tablicy. W tablicach 20.1 i 20.2 podano wartości współcz ników rekursywnych dla filtrów dolno- i górnoprzepustowego z zafalowaniami w przepustowym równymi 0,5%. Jeśli chcecie wykonać szybki, przybliżony projekt, prostu skopiujcie do waszego programu odpowiednie współczynniki.

Dwie kwestie pojawiają się, gdy korzysta się z tablic przy projektowania filtrów pierwsze, wybór parametrów zawartych w tablicach jest ograniczony. W tablicy uwzględniono np. tylko 12 różnych wartości częstotliwości odcięcia, maksymalnie gunów na filtr i nie ma tam żadnej możliwości wyboru wartości zafalowań w przepustowym. Nie można *zoptymalizować* filtru bez możliwości wyboru parame z ciągłego zakresu wartości. Po drugie, współczynniki trzeba ręcznie wprowadzać blicy do programu. Jest to bardzo czasochłonne i może zniechęcić do wypróbowywa wartości alternatywnych.

Zamiast stosowania wartości stabelaryzowanych, rozważcie włączenie do progra podprogramu, który *oblicza* współczynniki. Taki program zamieszczono w tablicy Korzystna, dzięki swej prostocie, jest struktura programu. Po wprowadzeniu czterech rametrów filtru program daje współczynniki *a* i *b* w matrycach A[ ] i B[ ]. Niekorzy zaś jest wywołanie w programie podprogramu z tablicy 20.5. Na pierwszy rzut podprogram wygląda nieładnie. Nie martwcie się, nie jest on taki zły, na jaki wygląda! w nim jedno proste rozgałęzienie w wierszu 1120. Cała reszta podprogramu to po obliczanie liczb. Sześć zmiennych wchodzi do tego podprogramu, a pięć z niego wych Wewnątrz podprogramu jest stosowanych piętnaście zmiennych tymczasowych (oraz deksy). W tablicy 20.5 zawarto dwa zestawy danych testowych służących do sprawdz i usuwania błędów z podprogramu. Działanie tego programu będzie szczegółowo w rozdziale 33.

Przerzut odpowiedzi skokowej

Filtry Butterwortha i Czebyszewa mają odpowiedź skokową z przerzutem od 5 do zwiększającym się przy wzroście liczby biegunów. Na rys. 20.3 a) przedstawiono kłady dwóch odpowiedzi skokowych filtrów Czebyszewa. Na rys. 20.3 b) pok charakterystyki, które są unikatowe w filtrach cyfrowych i nie mają żadnego wiednika w elektronice analogowej. Charakterystyki te wykazują, w niewielkim niu, zależność przerzutu w odpowiedzi skokowej od częstotliwości odcięcia. Nadmi przerzut i oscylacje w odpowiedzi skokowej wynikają z tego, że filtr Czebyszewa zoptymalizowany w *dziedzinie częstotliwości* kosztem jego właściwości w *dziedzinie czasu*.



Rys. 20.3. Odpowiedź skokowa Czebyszewa. Przerzut w odpowiedzi impulsowej filtru Czebyszewa jest równy od 5 do w zależności od liczby biegunów (rys. a) i od częstotliwości odcięcia (rys. b). Wykresy na rys. a) są wykonane dla częstotliwości odcięcia 0,05 i mogą być przeskalowane dla innych częstotliwości odcięcia

Stabilność

Podstawowym ograniczeniem w filtrach cyfrowych realizowanych przez splot jest *czas wykonania*. Można uzyskać niemal dowolną odpowiedź filtru, o ile tylko możemy wystarczająco długo poczekać na uzyskanie wyniku. Z filtrami rekursywnymi sytuacja jest całkiem przeciwna. Działają jak błyskawica; jednak są ograniczone pod względem jakości działania. Jako przykład rozważmy 6-biegunowy filtr dolnoprzepustowy o zafalowaniach 0,5% i częstotliwości odcięcia 0,01. Współczynniki rekursywne dla takiego filtru można uzyskać z tablicy 20.1:

```
a0=1,391351E-10
a1=8,348109E-10      b1=5,883343E+00
a2=2,087027E-09      b2=-1,442798E+01
a3=2,782703E-09      b3=1,887786E+01
a4=2,087027E-09      b4=-1,389914E+01
a5=8,348109E-10      b5=5,459909E+00
a6=1,391351E-10      b6=-8,939932E-01
```

Przyjrzyjmy się dokładnie tym parametrom. Współczynniki *b* mają wartości bezwzględne ok. *dziesięć*. Stosując pojedynczą dokładność, uzyskuje się szum zaokrąglenia w każdej z tych liczb równy ok. jednej dziesięciomilionowej wartości współczynnika, tj. 10^{-6}. Spójrzmy teraz na współczynniki *a* o wartości ok. 10^{-9}. Coś tu jest w sposób oczywisty nie w porządku. Udział pochodzący od sygnału wejściowego (przez współczynniki *a*) będzie 1000 razy mniejszy od *szumu* z poprzednio obliczonego sygnału wyjściowego (przez współczynniki *b*). Taki filtr nie będzie działał. Krótko mówiąc, szum zaokrąglania ogranicza liczbę biegunów, jaką można zastosować w filtrze. Ta liczba będzie w niewielkim stopniu zależna od zafalowań i od rodzaju przepustowości filtru (dolno- czy górnoprzepustowy). Przybliżone liczby, przy pojedynczej dokładności, podano w tablicy 20.3.

Działanie filtru zaczyna się pogarszać, gdy dochodzi się do granic określonych w tablicy 20.3. Na odpowiedzi skokowej pojawi się wówczas zwiększony przerzut, tłumienie w paśmie zaporowym będzie słabe, a odpowiedź częstotliwościowa będzie miała nadmierne zafalowania. Jeśli, projektując filtr, posuniemy się zbyt daleko lub występuje jakiś błąd we współczynnikach, to na wyjściu przypuszczalnie pojawi się oscylacja, zanim nie nastąpi przekroczenie zakresu.

Tab. 20.1. Dolnoprzepustowe filtry Czebyszewa (zafalowania 0,5%)

f_c	2-biegunowy		4-biegunowy		6-biegunowy	
0,01	a0=8,663387E-04 a1=1,732678E-03 a2=8,663387E-04	b1=1,919129E+00 b2=-9,225943E-01	a0=4,149425E-07 a1=1,659770E-06 a2=2,489655E-06 a3=1,659770E-06 a4=4,149425E-07	(!! Unstable !!) b1=3,893453E+00 b2=-5,688233E+00 b3=3,695783E+00 b4=-9,010106E-01	a0=1,391351E-10 a1=8,348109E-10 a2=2,087027E-09 a3=2,787703E-09 a4=2,087027E-09 a5=8,348109E-10 a6=1,391351E-10	[illegible]
0,025	a0=5,112374E-03 a1=1,022475E-02 a2=5,112374E-03	b1=1,797154E+00 b2=-8,176033E-01	a0=1,504626E-05 a1=6,018503E-05 a2=9,027754E-05 a3=6,018503E-05 a4=1,504626E-05	b1=3,725385E+00 b2=-5,226004E+00 b3=3,270902E+00 b4=-7,705239E-01	a0=3,136210E-08 a1=1,881726E-07 a2=4,704314E-07 a3=6,272419E-07 a4=4,704314E-07 a5=1,881726E-07 a6=3,136210E-08	[illegible]
0,05	a0=1,868823E-02 a1=3,737647E-02 a2=1,868823E-02	b1=1,593937E+00 b2=-6,686903E-01	a0=2,141509E-04 a1=8,566037E-04 a2=1,284906E-03 a3=8,566037E-04 a4=2,141509E-04	b1=3,425455E+00 b2=-4,479272E+00 b3=2,643718E+00 b4=-5,933269E-01	a0=1,771089E-06 a1=1,062654E-05 a2=2,656634E-05 a3=3,542179E-05 a4=2,656634E-05 a5=1,062654E-05 a6=1,771089E-06	[illegible]
0,075	a0=3,869430E-02 a1=7,738860E-02 a2=3,869430E-02	b1=1,392667E+00 b2=-5,474446E-01	a0=9,726342E-04 a1=3,890537E-03 a2=5,835806E-03 a3=3,890537E-03 a4=9,726342E-04	b1=3,103944E+00 b2=-3,774453E+00 b3=2,111238E+00 b4=-4,562908E-01	a0=1,797538E-05 a1=1,078523E-04 a2=2,696307E-04 a3=3,595076E-04 a4=2,696307E-04 a5=1,078523E-04 a6=1,797538E-05	[illegible]
0,1	a0=6,372802E-02 a1=1,274560E-01 a2=6,372802E-02	b1=1,194365E+00 b2=-4,492774E-01	a0=2,780755E-03 a1=1,112302E-02 a2=1,668453E-02 a3=1,112302E-02 a4=2,780755E-03	b1=2,764031E+00 b2=-3,122854E+00 b3=1,664554E+00 b4=-3,502232E-01	a0=9,086148E-05 a1=5,451688E-04 a2=1,362922E-03 a3=1,817229E-03 a4=1,362922E-03 a5=5,451688E-04 a6=9,086148E-05	[illegible]
0,15	a0=1,254285E-01 a1=2,508570E-01 a2=1,254285E-01	b1=8,070778E-01 b2=-3,087918E-01	a0=1,180009E-02 a1=4,720034E-02 a2=7,080051E-02 a3=4,720034E-02 a4=1,180009E-02	b1=2,039039E+00 b2=-2,012961E+00 b3=9,897915E-01 b4=-2,046700E-01	a0=8,618665E-04 a1=5,171199E-03 a2=1,292800E-02 a3=1,723733E-02 a4=1,292800E-02 a5=5,171199E-03 a6=8,618665E-04	[illegible]
0,2	a0=1,997396E-01 a1=3,994792E-01 a2=1,997396E-01	b1=4,291048E-01 b2=-2,280633E-01	a0=3,224554E-02 a1=1,289821E-01 a2=1,934732E-01 a3=1,289821E-01 a4=3,224554E-02	b1=1,265912E+00 b2=-1,203878E+00 b3=5,405908E-01 b4=-1,185538E-01	a0=4,187408E-03 a1=2,512445E-02 a2=6,281112E-02 a3=8,374816E-02 a4=6,281112E-02 a5=2,512445E-02 a6=4,187408E-03	[illegible]
0,25	a0=2,858110E-01 a1=5,716221E-01 a2=2,858110E-01	b1=5,423258E-02 b2=-1,974768E-01	a0=7,015301E-02 a1=2,806120E-01 a2=4,209180E-01 a3=2,806120E-01 a4=7,015301E-02	b1=4,541481E-01 b2=-7,417536E-01 b3=2,361222E-01 b4=-7,096476E-02	a0=1,434449E-02 a1=8,606697E-02 a2=2,151674E-01 a3=2,868899E-01 a4=2,151674E-01 a5=8,606697E-02 a6=1,434449E-02	[illegible]
0,3	a0=3,849163E-01 a1=7,698326E-01 a2=3,849163E-01	b1=-3,249116E-01 b2=2,147536E-01	a0=1,335566E-01 a1=5,342263E-01 a2=8,013394E-01 a3=5,342263E-01 a4=1,335566E-01	b1=-3,904486E-01 b2=-6,784138E-01 b3=-1,412021E-02 b4=-5,392238E-02	a0=3,997487E-02 a1=2,398492E-01 a2=5,996231E-01 a3=7,994975E-01 a4=5,996231E-01 a5=2,398492E-01 a6=3,997487E-02	[illegible]
0,35	a0=5,001024E-01 a1=1,000205E+00 a2=5,001024E-01	b1=-7,158993E-01 b2=-2,645103E-01	a0=2,340973E-01 a1=9,363892E-01 a2=1,404584E+00 a3=9,363892E-01 a4=2,340973E-01	b1=-1,263672E+00 b2=-1,080487E+00 b3=-3,276296E-01 b4=-7,376791E-02	a0=9,792321E-02 a1=5,875393E-01 a2=1,468848E+00 a3=1,958464E+00 a4=1,468848E+00 a5=5,875393E-01 a6=9,792321E-02	[illegible]
0,40	a0=6,362308E-01 a1=1,272462E+00 a2=6,362308E-01	b1=-1,125379E+00 b2=-4,195441E-01	a0=3,896966E-01 a1=1,558787E+00 a2=2,338180E+00 a3=1,558787E+00 a4=3,896966E-01	b1=-2,161179E+00 b2=-2,033992E+00 b3=-8,789094E-01 b4=-1,610655E-01	a0=2,211834E-01 a1=1,327100E+00 a2=3,317751E+00 a3=4,423668E+00 a4=3,317751E+00 a5=1,327100E+00 a6=2,211834E-01	[illegible]
0,45	a0=8,001101E-01 a1=1,600220E+00 a2=8,001101E-01	b1 -1,556269E+00 b2 -6,441713E-01	a0=6,291693E-01 a1=2,516677E+00 a2=3,775016E+00 a3=2,516677E+00 a4=6,291693E-01	b1=-3,077062E+00 b2=-3,641323E+00 b3=-1,949229E+00 b4=-3,990945E-01	a0=4,760636E-01 a1=2,856381E+00 a2=7,140952E+00 a3=9,521270E+00 a4=7,140952E+00 a5=2,856381E+00 a6=4,760635E-01	[illegible]

Tab. 20.2. Górnoprzepustowe filtry Czebyszewa (zafalowania 0,5%)

f_c	2-biegunowy		4-biegunowy		6-biegunowy	
0,01	[illegible]	b1=1,911437E+00 b2=-9,155749E-01	a0=9,121579E-01 a1=-3,648632E+00 a2=5,472947E+00 a3=-3,648632E+00 a4=9,121579E-01	(!! Unstable !!) b1=3,815952E+00 b2=-5,465026E+00 b3=3,481295E+00 b4=-8,322529E-01	a0=8,630195E-01 a1=-5,178118E+00 a2=1,294529E+01 a3=-1,726039E+01 a4=1,294529E+01 a5=-5,178118E+00 a6=8,630195E-01	(!! Unstable !!) b1=5,705102E+00 b2=-1,356935E+01 b3=1,722231E+01 b4=-1,230214E+01 b5=4,689218E+00 b6=-7,451429E-01
0,025	[illegible]	b1=-1,777932E+00 b2=-8,022106E-01	a0=7,941874E-01 a1=-3,176750E+00 a2=4,765125E+00 a3=-3,176750E+00 a4=7,941874E-01	b1=3,538919E+00 b2=-4,722213E+00 b3=2,814036E+00 b4=-6,318300E-01	a0=6,912863E-01 a1=-4,147718E+00 a2=1,036929E+01 a3=-1,382573E+01 a4=1,036929E+01 a5=-4,147718E+00 a6=6,912863E-01	(!! Unstable !!) b1=5,261399E+00 b2=-1,157800E+01 b3=1,363599E+01 b4=-9,063840E+00 b5=3,223738E+00 b6=-4,793541E-01
0,05	[illegible]	b1=-1,556269E+00 b2=-6,441715E-01	a0=6,291694E-01 a1=-2,516678E+00 a2=3,775016E+00 a3=-2,516678E+00 a4=6,291694E-01	b1=3,077062E+00 b2=-3,641324E+00 b3=1,949230E+00 b4=-3,990947E-01	a0=4,760636E-01 a1=-2,856382E+00 a2=7,140954E+00 a3=-9,521272E+00 a4=7,140954E+00 a5=-2,856382E+00 a6=4,760636E-01	b1=4,522403E+00 b2=-8,676846E+00 b3=9,007515E+00 b4=-5,328431E+00 b5=1,702544E+00 b6=-2,303304E-01
0,075	[illegible]	b1=-1,338264E+00 b2=-5,185469E-01	a0=4,965350E-01 a1=-1,986140E+00 a2=2,979210E+00 a3=-1,986140E+00 a4=4,965350E-01	b1=2,617304E+00 b2=-2,749252E+00 b3=1,325548E+00 b4=-2,524546E-01	a0=3,259100E-01 a1=-1,955460E+00 a2=4,888651E+00 a3=-6,518201E+00 a4=4,888651E+00 a5=-1,955460E+00 a6=3,259100E-01	b1=3,787397E+00 b2=-6,288362E+00 b3=5,747801E+00 b4=-3,041570E+00 b5=8,808669E-01 b6=-1,122464E-01
0,1	[illegible]	b1=1,125379E+00 b2=-4,195440E-01	a0=3,896966E-01 a1=-1,558786E+00 a2=2,338179E+00 a3=-1,558786E+00 a4=3,896966E-01	b1=2,161179E+00 b2=-2,033991E+00 b3=8,789094E-01 b4=-1,610655E-01	a0=2,211833E-01 a1=-1,327100E+00 a2=3,317750E+00 a3=-4,423667E+00 a4=3,317750E+00 a5=-1,327100E+00 a6=2,211833E-01	b1=3,058671E+00 b2=-4,390464E+00 b3=3,523252E+00 b4=-1,684184E+00 b5=4,414878E-01 b6=-5,767508E-02
0,15	[illegible]	b1=7,158993E-01 b2=-2,645103E-01	a0=2,340973E-01 a1=-9,363892E-01 a2=1,404584E+00 a3=-9,363892E-01 a4=2,340973E-01	b1=1,263672E+00 b2=-1,080487E+00 b3=3,276296E-01 b4=-7,376791E-02	a0=9,792321E-02 a1=-5,875393E-01 a2=1,468848E+00 a3=-1,958464E+00 a4=1,468848E+00 a5=-5,875393E-01 a6=9,792321E-02	b1=1,627573E+00 b2=-1,955020E+00 b3=1,075051E+00 b4=-5,106501E-01 b5=7,239843E-02 b6=-2,639193E-02
0,2	[illegible]	b1=3,249116E-01 b2=-2,147536E-01	a0=1,335566E-01 a1=-5,342262E-01 a2=8,013393E-01 a3=-5,342262E-01 a4=1,335566E-01	b1=3,904484E-01 b2=-6,784138E-01 b3=1,412016E-02 b4=-5,392238E-02	a0=3,997486E-02 a1=-2,398492E-01 a2=5,996230E-01 a3=-7,994973E-01 a4=5,996230E-01 a5=-2,398492E-01 a6=3,997486E-02	b1=2,441149E-01 b2=-1,130306E+00 b3=-1,063169E-01 b4=-3,463299E-01 b5=-8,882996E-02 b6=-3,278741E-02
0,25	[illegible]	b1=-5,423243E-02 b2=-1,974768E-01	a0=7,015302E-02 a1=-2,806121E-01 a2=4,209182E-01 a3=-2,806121E-01 a4=7,015302E-02	b1=-4,541478E-01 b2=-7,417535E-01 b3=-2,361221E-01 b4=-7,096475E-02	a0=1,434450E-02 a1=-8,606701E-02 a2=2,151675E-01 a3=-2,868900E-01 a4=2,151675E-01 a5=-8,606701E-02 a6=1,434450E-02	b1=-1,076051E+00 b2=-1,662847E+00 b3=-1,191062E+00 b4=-7,403085E-01 b5=-2,752156E-01 b6=-5,722250E-02
0,3	[illegible]	b1=-4,291049E-01 b2=-2,280633E-01	a0=3,224553E-02 a1=-1,289821E-01 a2=1,934732E-01 a3=-1,289821E-01 a4=3,224553E-02	b1=-1,265912E+00 b2=-1,203878E+00 b3=-5,405908E-01 b4=-1,185538E-01	a0=4,187407E-03 a1=-2,512444E-02 a2=6,281111E-02 a3=-8,374815E-02 a4=6,281111E-02 a5=-2,512444E-02 a6=4,187407E-03	b1=-2,315806E+00 b2=-3,293726E+00 b3=-2,904827E+00 b4=-1,694129E+00 b5=-6,021426E-01 b6=-1,029147E-01
0,35	[illegible]	b1=-8,070777E-01 b2=-3,087918E-01	a0=1,180009E-02 a1=-4,720035E-02 a2=7,080051E-02 a3=-4,720035E-02 a4=1,180009E-02	b1=-2,039039E+00 b2=-2,012961E+00 b3=-9,897915E-01 b4=-2,046700E-01	a0=8,618665E-04 a1=-5,171200E-03 a2=1,292800E-02 a3=-1,723733E-02 a4=1,292800E-02 a5=-5,171200E-03 a6=8,618665E-04	b1=-3,455239E+00 b2=-5,754734E+00 b3=-5,645387E+00 b4=-3,394902E+00 b5=-1,177469E+00 b6=-1,936195E-01
0,40	[illegible]	b1=-1,194365E+00 b2=-4,492774E-01	a0=2,780754E-03 a1=-1,112302E-02 a2=1,668453E-02 a3=-1,112302E-02 a4=2,780754E-03	b1=-2,764031E+00 b2=-3,122854E+00 b3=-1,664554E+00 b4=-3,502233E-01	a0=9,086141E-05 a1=-5,451685E-04 a2=1,362921E-03 a3=-1,817228E-03 a4=1,362921E-03 a5=-5,451685E-04 a6=9,086141E-05	b1=-4,470118E+00 b2=-8,755595E+00 b3=-9,543712E+00 b4=-6,079377E+00 b5=-2,140062E+00 b6=-3,247363E-01
0,45	[illegible]	b1=-1,593937E+00 b2=-6,686903E-01	a0=2,141509E-04 a1=-8,566037E-04 a2=1,284906E-03 a3=-8,566037E-04 a4=2,141509E-04	b1=-3,425455E+00 b2=-4,479272E+00 b3=-2,643718E+00 b4=-5,933269E-01	a0=1,771089E-06 a1=-1,062654E-05 a2=2,656634E-05 a3=-3,542179E-05 a4=2,656634E-05 a5=-1,062654E-05 a6=1,771089E-06	b1=-5,330512E+00 b2=-1,196611E+01 b3=-1,447067E+01 b4=-9,937710E+00 b5=-3,673283E+00 b6=-5,707561E-01

Tablica 20.3. Maksymalna liczba biegunów przy pojedynczej dokładności

[illegible]	0,02	0,05	0,10	0,25	0,40	0,45	0,48
[illegible]	4	6	10	20	10	6	4

Tab. 20.4

```
100 'OBLICZANIE PARAMETRÓW REKURSYWNYCH FILTRU CZEBYSZEWA
110 '
120                                 'INICJALIZACJA ZMIENNYCH
130 DIM A[22]                       'przechowuje współczynniki "a" po zakończeniu programu
140 DIM B[22]                       'przechowuje współczynniki "b" po zakończeniu programu
150 DIM TA[22]                      'wewnętrzne zastosowanie do łączenia stopni
160 DIM TB[22]                      'wewnętrzne zastosowanie do łączenia stopni
170 '
180 FOR I% = 0 TO 22
190   A[I%] = 0
200   B[I%] = 0
210 NEXT I%
220 '
230 A[2] = 1
240 B[2] = 1
250 PI = 3.14159265
260                                 'WPROWADZENIE CZTERECH PARAMETRÓW FILTRU
270 INPUT "Enter cutoff frequency (0 to .5): ", FC
280 INPUT "Enter 0 for LP,   for HP filter:  ", LH
290 INPUT "Enter percent ripple  (0 to 29):  ", PR
300 INPUT "Enter number of poles (2,4,...20):", NP
310 '
320 FOR P% = 1 TO NP/2              'PĘTLA PRZEZ WSZYSTKIE PARY BIEGUNÓW
330 '
340   GOSUB 1000   Podprogram z tablicy 20.5
350 '
360   FOR I% = 0 TO 22              'Dodaj współczynniki do kaskady
370     TA[I%] = A[I%]
380     TB[I%] = B[I%]
390   NEXT I%
400 '
410   FOR I% = 2 TO 22
420     A[I%] = A0*TA[I%] + A1*TA[I%-1] + A2*TA[I%-2]
430     B[I%] = TB[I%] - B1*TB[I%-1] - B2*TB[I%-2]
440   NEXT I%
450 '
460 NEXT P%
470 '
480 B[2] = 0                        'Zakończenie łączenia współczynników
490 FOR I% = 0 TO 20
500   A[I%] = A[I%+2]
510   B[I%] = -B[I%+2]
520 NEXT I%
530 '
540 SA = 0                          'NORMALIZACJA WZMOCNIENIA
550 SB = 0
560 FOR I% = 0 TO 20
570   IF LH = 0 THEN SA = SA + A[I%]
580   IF LH = 0 THEN SB = SB + B[I%]
590   IF LH = 1 THEN SA = SA + A[I%]*(-1)^I%
600   IF LH = 1 THEN SB = SB + B[I%]*(-1)^I%
610 NEXT I%
620 '
630 GAIN = SA/(1 -SB)
640 '
650 FOR I% = 0 TO 20
660   A[I%] = A[I%] / GAIN
670 NEXT I%
680                                 'Końcowe wartości współczynników rekursywnych są w A[ ] i B[
690 END
```


Tablice 20.4 i 20.5. Program do obliczania współczynników a i b filtrów rekursywnych Czebyszewa. W wierszach 270–300 programu są wprowadzane cztery parametry. Częstotliwość odcięcia FC jest wyrażona jako ułamek częstotliwości próbkowania, musi się więc mieścić w zakresie od 0 do 0,5. Ustawia się wartość zmiennej LH równą jeden dla filtru górnoprzepustowego a zero dla dolnoprzepustowego. Wprowadzona wartość PR powinna być zawarta w zakresie od 0 do 29, co odpowiada wartościom od 0 do 29% zafalowań w odpowiedzi częstotliwościowej filtru. Liczba biegunów w charakterystyce filtru, ustawiana jako zmienna NP, musi być parzysta, z zakresu od 2 do 20. Na zakończenie programu współczynniki a i b są zapisane w matrycach A[ ] i B[ ] (a_0 = A[0], a_1 = A[1] itd.). Tablica 20.5 zawiera podprogram wywoływany z wiersza 340 programu głównego. Do tego podprogramu jest przekazywanych sześć zmiennych, a uzyskuje się z niego pięć. W tablicy 20.6 podano dwa zestawy danych pomagające w sprawdzaniu programu i usuwaniu błędów. W funkcjach COS i SIN użyto radianów, a nie stopni. Funkcja LOG jest logarytmem naturalnym o podstawie e. Zadeklarowanie wszystkich wielkości zmiennoprzecinkowych (łącznie z liczbą π) jako wartości o podwójnej dokładności umożliwia zastosowanie większej liczby biegunów. Tablice 20.1 i 20.2 utworzono, korzystając z tego programu. Mogą być one użyte do sprawdzania prawidłowości działania. Matematyczne operacje realizowane w tym programie omówiono w rozdziale 33

```
1000 'TEN PODPROGRAM JEST WYWOŁYWANY Z TABLICY 20.4, WIERSZ 340
1010 '
1020 ' Zmienne wchodzące do podprogramu:      PI, FC, LH, PR, HP, P%
1030 ' Zmienne wychodzące z podprogramu:      A0, A1, A2, B1, B2
1040 ' Zmienne pomocnicze                     RP, IP, ES, VX, KX, T, W, M, D, K,
1050 '                                        X0, X1, X2, Y1, Y2
1060 '
1070 '                                        'Oblicz położenie bieguna na okręgu jednostkowym
1080 RP = -COS(PI/(NP*2) + (P%-1) * PI/NP)
1090 IP = SIN(PI/(NP*2) + (P%-1) * PI/NP)
1100 '
1110 '                                        'Odkształcenie okręgu w elipsę
1120 IF PR = 0 THEN GOTO 1210
1130 ES = SQR( (100/(100-PR))^2-1 )
1140 VX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) + 1))
1150 KX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) - 1))
1160 KX = (EXP(KX) + EXP(-KX))/2
1170 RP = RP * ( (EXP(VX) - EXP(-VX) ) /2 ) / KX
1180 IP = IP * ((EXP(VX) + EXP(-VX) ) /2 ) / KX
1190 '
1200 '                                        'Konwersja dziedziny s w dziedzinę Z
1210 T =2 *TAN(1/2)
1220 W =2*PI*FC
1230 M =RP^2 + IP^2
1240 D = 4 - 4*RP*T + M*T^2
1250 X0 = T^2/D
1260 X1 = 2*T^2/D
1270 X2 = T^2/D
1280 Y1 = (8 - 2*M*T^2)/D
1290 Y2 = (-4 - 4*RP*T - M*T^2)/D
1300 '
1310 '                                        'Przekształcenie LP w LP lub LP w HP
1320 IF LH = 1 THEN K = -COS(W/2 + 1/2) / COS(W/2 - 1/2)
1330 IF LH = 0 THEN K = SIN(1/2 - W/2) / SIN(1/2 + W/2)
1340 D = 1 +Y1*K - Y2*K^2
1350 A0 = (X0 - X1*K + X2*K^2)/D
1360 A1 = (-2*X0*K + X1 + X1*K^2 - 2*X2*K)/D
1370 A2 = (X0*K^2 - X1*K + X2)/D
1380 B1 = (2*K + Y1 + Y1*K^2 - 2*Y2*K)/D
1390 B2 = (-(K^2) - Y1*K + Y2)/D
1400 IF LH = 1 THEN A1 = -A1
1410 IF LH = 1 THEN B1 = -B1
1420 '
1430 RETURN
```

Są dwa sposoby zwiększenia maksymalnej liczby biegunów, jaką możemy zastosować. Po pierwsze, skorzystajcie z podwójnej dokładności. Wymaga to zastosowania podwójnej dokładności również przy obliczaniu współczynników (z wartością π włącznie).

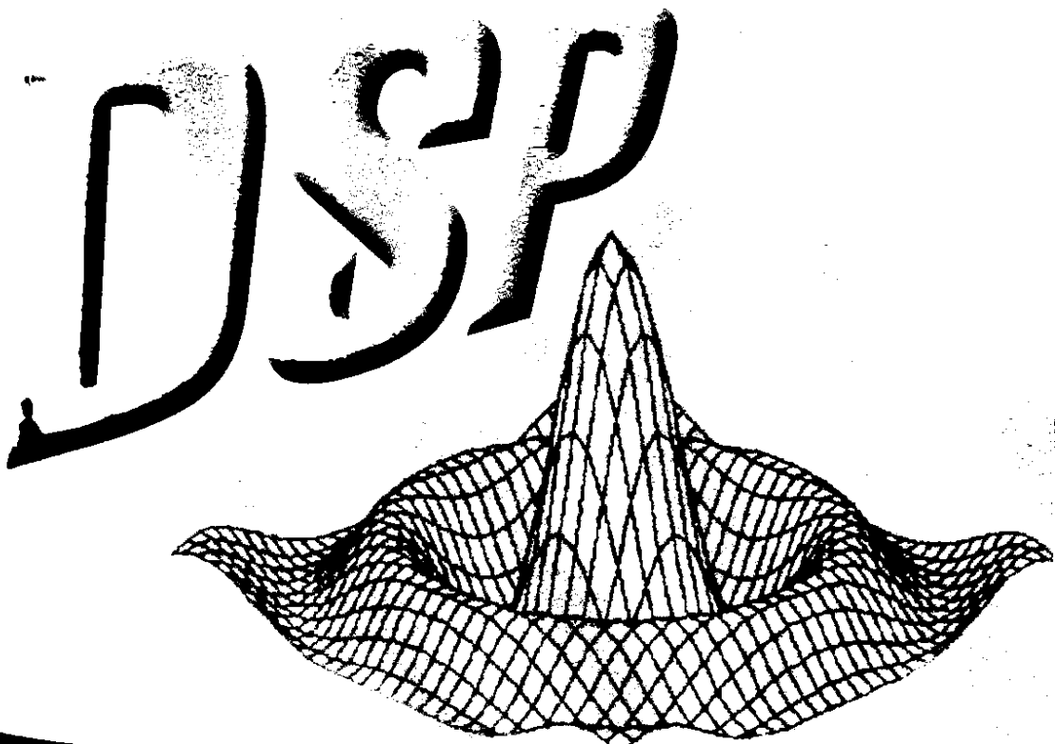
Drugim sposobem jest implementacja filtru przez realizację jego *stopni*. Na przykład realizacja filtru 6-biegunowego zaczyna się od kaskady trzech stopni, po dwa bieguny każdy. Program podany w tablicy 20.4 łączy te trzy stopnie w jeden zestaw współczynników rekursywnych w celu ułatwienia programowania. Filtr jest jednak bardziej stabilny, jeśli jest realizowany jako trzy oddzielne stopnie. To wymaga znajomości współczynników *a* i *b* dla każdego z tych stopni. Można je uzyskać z programu zamieszczonego w tablicy 20.4. Podprogram z tablicy 20.5 jest wywoływany dla każdego stopnia w kaskadzie. Na przykład dla filtru 6-biegunowego jest wywoływany trzy razy. Po zakończeniu wykonywania podprogramu pięć zmiennych powraca do programu głównego: *A*0, *A*1, *A*2, *B*1 *i B*2. Są to współczynniki rekursywne dla właśnie realizowanego stopnia dwubiegunowego i mogą być użyte do implementacji filtru przez realizację poszczególnych stopni.

Tab. 20.6. Dane do sprawdzania podprogramu i usuwania błędów. Tablica zawiera dwa zestawy danych służące do sprawdzania podprogramu z tablicy 20.5 i usuwania błędów

ZESTAW DANYCH 1	ZESTAW DANYCH 2
Wejdź do podprogramu z następującymi wartościami:	
FC= 0,1	FC= 0,1
LH= 0	LH= 1
PR= 0	PR= 10
NP= 4	NP= 4
P%= 1	P%= 2
PI= 3,141592	PI= 3,141592
Te wartości powinny być w wierszu 1200:	
RP= –0,923879	RP= –0,136178
IP= 0,382683	IP= 0,933223
ES= not used	ES= 0,484322
VX= not used	VX= 0,368054
KX= not used	KX= 1,057802
Te wartości powinny być w wierszu 1310:	
T= 1,092605	T= 1,092605
W= 0,628318	W= 0,628318
M= 1,000000	M= 0,889450
D= 9,231528	D= 5,656972
X0= 0,129316	X0= 0,211029
X1= 0,258632	X1= 0,422058
X2= 0,129316	X2= 0,211029
Y1= 0,607963	Y1= 1,038784
Y2= –0,125227	Y2= –0,789584
Te wartości powinny powrócić do programu głównego:	
A0= 0,061885	A0= 0,922919
A1= 0,123770	A1= –1,845840
A2= 0,061885	A2= 0,922919
B1= 1,048600	B1= 1,446913
B2= –0,296140	B2= –0,836653

21

Porównanie filtrów



Decyzje, decyzje, decyzje! Przy tak dużym wyborze filtrów, jak zdecydować, który wybrać? W tym rozdziale różne filtry będą bezpośrednio rywalizować ze sobą. Wytypujemy najlepsze filtry w poszczególnych grupach i pozwolimy im zmierzyć się ze sobą. W pierwszym pojedynku filtry *cyfrowe* wystąpią przeciw *analogowym*, aby rozstrzygnąć, która z tych technologii jest lepsza. W drugim – filtr okienkowany funkcją *sinc* będzie walczył z filtrem Czebyszewa o pozycję króla filtrów *w dziedzinie częstotliwości*. W końcowym boju, o mistrzostwo w *dziedzinie czasu*, filtr o średniej kroczącej spotka się z filtrem jednobiegunowym. Dość gadania. Niech rozpocznie się rywalizacja.

Pojedynek 1: Filtry analogowe przeciw cyfrowym

Większość sygnałów cyfrowych ma swoje źródło w elektronice analogowej. Jeśli sygnał wymaga filtracji, to czy lepiej jest zastosować filtr analogowy przed przetworzeniem sygnału na cyfrowy, czy filtr cyfrowy po takim przetworzeniu? Odpowiemy na to pytanie, gdy oba pretendenci wymienią pierwsze ciosy.

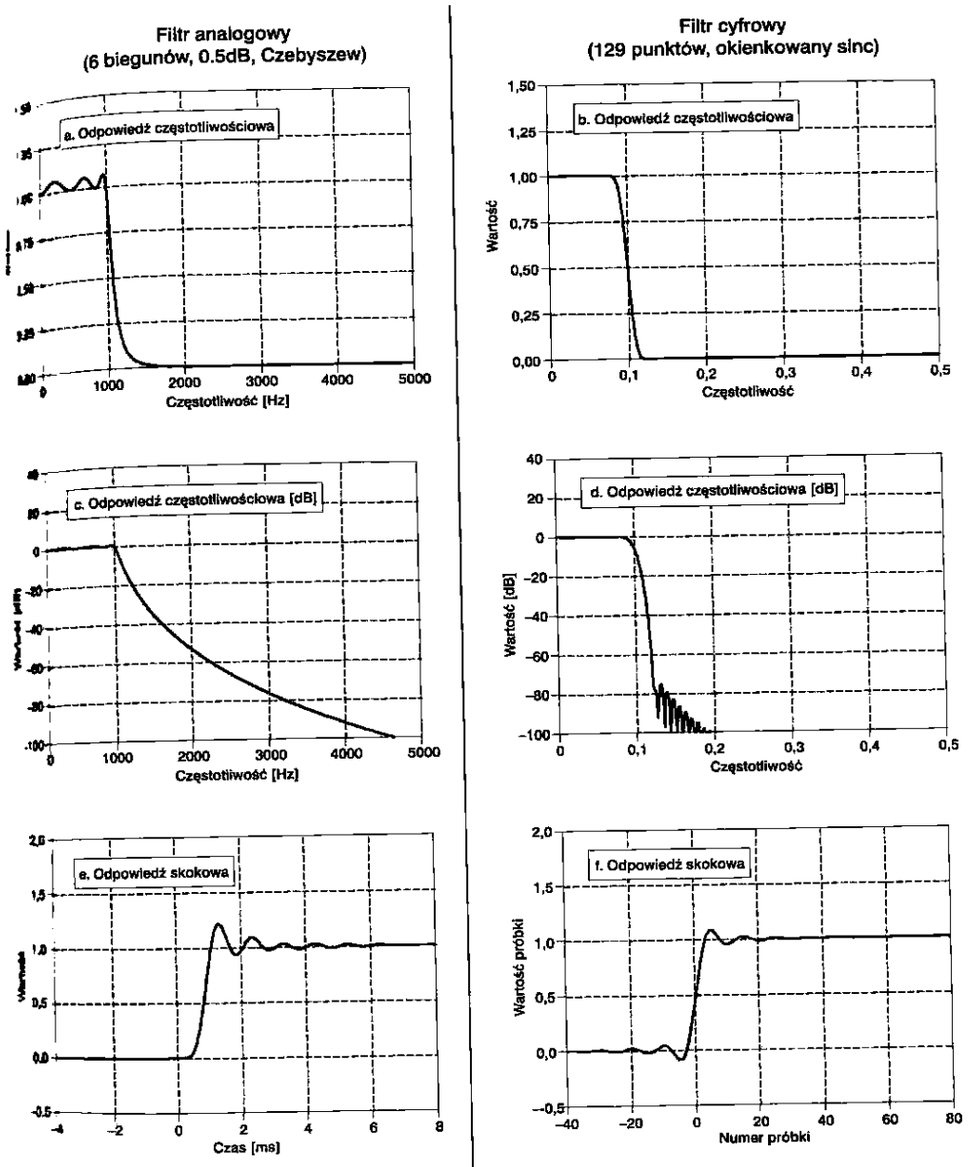
Celem rywalizacji jest uzyskanie filtru dolnoprzepustowego o częstotliwości odcięcia 1 kHz. Ze strony filtrów analogowych do pojedynku staje 6-biegunowy filtr Czebyszewa o zafalowaniach 0,5 dB (6%). Według opisu w rozdziale 3, można taki filtr skonstruować przy użyciu 3 wzmacniaczy operacyjnych, 12 rezystorów i 6 kondensatorów. W narożniku filtrów cyfrowych rozgrzewa się gotowy do walki filtr okienkowany funkcją *sinc*. Sygnał analogowy jest zamieniany na cyfrowy z częstotliwością próbkowania 10 kHz, więc częstotliwość odcięcia jest równa 0,1 na cyfrowej skali częstotliwości. Wybieramy długość odpowiedzi filtru okienkowanego funkcją *sinc* równą 129 punktów, co daje opadanie od 90% do 10%, takie samo jak w filtrze analogowym. Walka musi być czysta. Na rys. 21.1 przedstawiono odpowiedzi częstotliwościowe i skokowe obu tych filtrów.

Porównajmy oba filtry parametr po parametrze. Jak wynika z rysunków 21.1 a), b), filtr analogowy ma zafalowania 6% w paśmie przepustowym, podczas gdy filtr cyfrowy ma odpowiedź idealnie płaską (w granicach 0,02%). Projektant filtrów analogowych będzie argumentował, że wartość zafalowań może być *wybierana* podczas projektowania, ten argument jednak nie utrzymuje krytyki. Płaskość odpowiedzi uzyskiwana w filtrach analogowych jest bowiem ograniczona dokładnością użytych w nich rezystorów i kondensatorów. Nawet jeśli zaprojektuje się filtr o odpowiedzi Butterwortha (tj. o zafalowaniach 0%), to filtry o takiej złożoności będą miały jakieś resztkowe zafalowania, powiedzmy 1%. Z drugiej strony, płaskość odpowiedzi filtrów cyfrowych jest ograniczona głównie błędem zaokrąglania, co czyni tę odpowiedź *setki razy* bardziej płaską niż w ich analogowych odpowiednikach. Wynik: 1 punkt dla filtru cyfrowego.

Następnie przyjrzyjmy się odpowiedzi częstotliwościowej w skali logarytmicznej (rysunki 20.1c), d). Tu znowu filtr cyfrowy jest wyraźnym zwycięzcą, zarówno gdy chodzi o szybkość *opadania*, jak i *tłumienie w paśmie zaporowym*. Nawet jeśli działanie filtru analogowego można udoskonalić przez dodanie dodatkowych stopni, to i tak nie stanie się on porównywalny z filtrem cyfrowym. Załóżmy na przykład, że chcemy polepszyć te dwa parametry o czynnik 100. Można tego dokonać przez prostą modyfikację filtru okienkowanego funkcją *sinc*, lecz jest to faktycznie niemożliwe w układzie analogowym. A więc kolejny punkt dla filtrów cyfrowych.

Odpowiedzi skokowe tych dwóch filtrów pokazano na rysunkach 20.1 e), f). Odpowiedź skokowa filtru cyfrowego ma symetrię górnej i dolnej części skoku, tzn. że ma fazę liniową. Odpowiedź skokowa filtru analogowego *nie jest* symetryczna, tj. ma fazę nieliniową. Jeszcze jeden punkt dla filtru cyfrowego. Na koniec, filtr cyfrowy ma przerzut ok. 20% z jednej strony skoku, a cyfrowy ok. 10%, lecz po stronach skoku. Oba filtry są więc pod tym względem złe i żaden nie otrzymuje punktu.

Mimo wyniku tego pojedynku, ciągle jest wiele zastosowań, w których filtry analogowe powinny lub muszą być używane. Nie wiąże się to z rzeczywistym działaniem filtru (tj. z tym, co wchodzi do filtru i co z niego wychodzi), lecz z ogólnymi zaletami układów analogowych w stosunku



Rys. 21.1. Porównanie filtrów analogowych z cyfrowymi. Filtry cyfrowe działają lepiej pod względem wielu właściwości, takich jak zafalowania w paśmie przepustowym (porównajcie rysunki a) i b), szybkość opadania i tłumienia w paśmie zaporowym (porównajcie rysunki c) i d) oraz symetria odpowiedzi skokowej (porównajcie rysunki e) i f). W tym przykładzie filtr cyfrowy ma częstotliwość odcięcia 0,1 i częstotliwość próbkowania 10 kHz. To zapewnia rzetelne porównanie z filtrem analogowym o częstotliwości odcięcia 1 kHz

do cyfrowych. Pierwszą zaletą jest *szybkość*; układy cyfrowe są wolne, a analogowe szybkie. Na przykład komputer osobisty może filtrować dane z częstotliwością ok. 10 000 próbek/s przy użyciu splotu FFT. Nawet proste wzmacniacze operacyjne mogą działać przy częstotliwościach od 100 kHz do 1 MHz, a zatem od 10 do 100 razy szybciej niż system cyfrowy!

Drugą zaletą, charakterystyczną dla układów analogowych, jest *zakres dynamiczny*. Ten parametr występuje w dwóch aspektach. Pierwszy to **zakres dynamiczny amplitudy**, będący stosunkiem największego sygnału przechodzącego przez system do szumu wytwarzającego w systemie. Na przykład 12-bitowy przetwornik a/c ma poziom maksymalny (nasycenie) 4095 i wartość skuteczną szumu kwantyzacji równą 0,29 cyfry (LSB), a więc zakres dynamiczny wynosi ok. 14 000. Standardowy wzmacniacz operacyjny zaś ma napięcie nasycenia ok. 20 V i szum wewnętrzny ok. 2 μV, co daje zakres dynamiczny ok. *dziesięć milionów*. Tak jak poprzednio, pojedynczy wzmacniacz operacyjny nokautuje pod tym względem system cyfrowy.

Drugim aspektem omawianego parametru jest **zakres dynamiczny częstotliwości**. Łatwo można zaprojektować układ ze wzmacniaczem operacyjnym, działający w zakresie częstotliwości od 0,01 Hz do 100 kHz (siedem dekad). Gdyby w systemie cyfrowym próbkowanie pracować w takim zakresie, to komputer wkrótce zapchałby się danymi. Na przykład przy próbkowaniu z częstotliwością 200 kHz trzeba zarejestrować aż 20 milionów punktów, aby uchwycić pełny okres przebiegu o częstotliwości 0,01 Hz. Zapewne zauważyliście, że odpowiedzi częstotliwościowe filtrów cyfrowych wykreśla się w skali *liniowej*, filtrów analogowych zaś – zwykle w skali *logarytmicznej*. Filtry cyfrowe wymagają bowiem skali liniowej, aby można było pokazać w szczegółach ich wyjątkowe działanie filtrujące, a w filtrach analogowych konieczna jest skala logarytmiczna, aby objąć ich olbrzymi zakres dynamiczny.

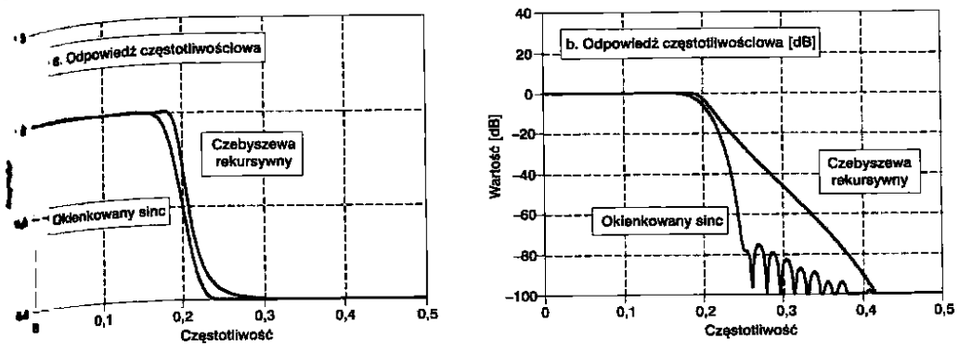
Pojedynek 2: Filtr okienkowany funkcją *sinc* przeciw filtrowi Czebyszewa

Zarówno filtry okienkowane funkcją *sinc*, jak i filtry Czebyszewa są projektowane pod kątem jak najlepszego rozdzielania pasm częstotliwości. Filtr okienkowany funkcją *sinc* jest filtrem o skończonej odpowiedzi impulsowej (SOI) implementowanym przez *splot*, a filtr Czebyszewa – filtrem o nieskończonej odpowiedzi impulsowej (NOI) realizowanym przez *rekursję*. Który filtr cyfrowy jest najlepszy w dziedzinie częstotliwości? Niech rozstrzygną o tym bezpośrednie starcie.

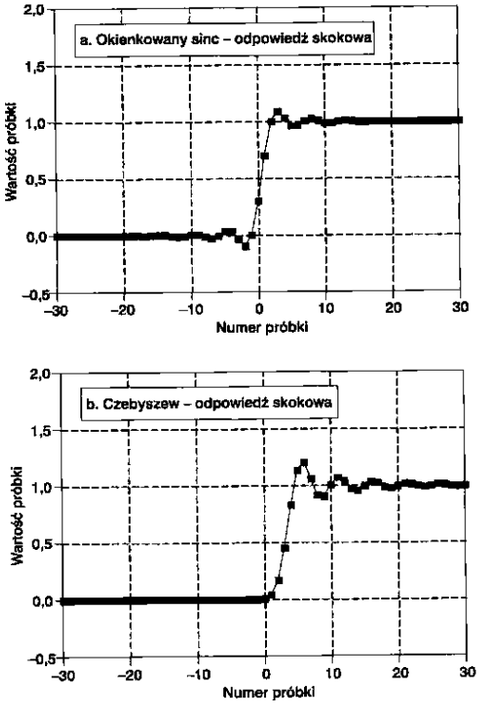
Pretendentem ze strony filtrów rekursywnych jest 6-biegunowy, dolnoprzepustowy filtr o zafalowaniach odpowiedzi 0,5%. Porównanie utrudnia fakt, że odpowiedź częstotliwościowa filtru Czebyszewa zmienia się przy zmianach częstotliwości odcięcia. Przyjmiemy częstotliwość odcięcia równą 0,2 i wybierzemy 51-punktową odpowiedź impulsową filtru okienkowanego funkcją *sinc*. Przy takim doborze parametrów oba filtry mają jednakową szybkość opadania od 90% do 10%, jak to przedstawiono na rys. 21.2 a).

I teraz zaczyna się wymiana ciosów. Filtr rekursywny ma zafalowania 0,5% w paśmie przepustowym, a odpowiedź filtru okienkowanego funkcją *sinc* jest płaska. Jednak możemy to two ustawić zafalowania filtru rekursywnego na 0%, jeśli jest taka potrzeba. Nie przyznamy więc punktu żadnemu filtrowi. Z rys. 21.2 b) wynika, że filtr okienkowany funkcją *sinc* ma znacznie lepsze tłumienie w paśmie zaporowym niż filtr Czebyszewa. Jeden punkt dla filtru okienkowanego *sinc*.

Odpowiedzi skokowe obu filtrów przedstawiono na rys. 21.3. Obie są niekorzystne, jak tego można było się spodziewać po filtrach przeznaczonych do dziedziny częstotliwości. Filtr rekursywny ma nieliniową fazę, lecz to można skorygować przez filtrację dwukierunkową. Ponieważ odpowiedzi skokowe obu filtrów są tak złe, więc uznajemy remis.



Rys. 21.2. Odpowiedzi częstotliwościowe filtru okienkowanego funkcją sinc i filtru Czebyszewa. Przedstawiono odpowiedzi częstotliwościowe dla 51-punktowego filtru okienkowanego funkcją sinc oraz dla 6-biegunowego rekursywnego filtru Czebyszewa o zafalowaniach 0,5%. Filtr okienkowany funkcją sinc ma lepszy współczynnik tłumienia w paśmie zaporowym, lecz oba filtry nadają się do zastosowań o umiarkowanych wymaganiach. Częstotliwość odcięcia obu filtrów wynosi 0,2, mierzona dla 0,5 (filtr okienkowany sinc) lub 0,707 wartości maksymalnej (filtr rekursywny)

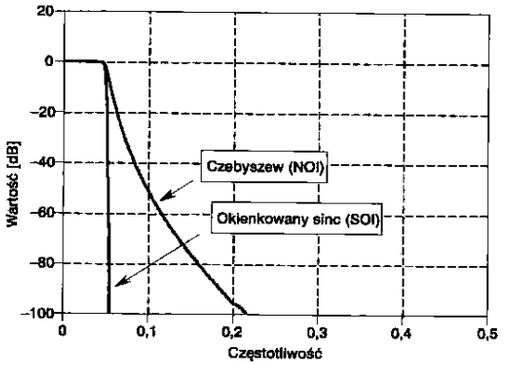


Rys. 21.3. Odpowiedzi skokowe filtrów: okienkowanego funkcją sinc i Czebyszewa. Pokazano odpowiedzi skokowe dla 51-punktowego filtru okienkowanego funkcją sinc oraz dla 6-biegunowego rekursywnego filtru Czebyszewa o zafalowaniach 0,5%. Oba filtry mają częstotliwość odcięcia 0,2. Filtr okienkowany funkcją sinc ma nieco lepszą odpowiedź skokową ze względu na mniejszy przerzut i zerową fazę

Jak dotąd nie widać większych różnic między obu porównywanymi filtrami; oba pracują zadowalająco, gdy stawiane wymagania nie są wysokie. Prawdziwa walka się zaczyna, gdy przechodzimy do dwóch krytycznych cech: *maksymalne możliwe parametry* i *szybkość działania*. Filtr okienkowany funkcją *sinc* jest potężny, podczas gdy filtr Czebyszewa jest szybki i zwarty. Przypuśćmy, że mamy rzeczywiście trudny problem rozdzielania częstotliwości, np. chcemy wydzielić 100 miliwoltowy sygnał o częstotliwości 61 Hz przenoszony po linii energetycznej 120 V, 60 Hz[17]. Na rys. 21.4 porównano oba filtry w sytuacji, gdy jest wymagane maksymalne wykorzystanie ich właściwości. Filtr rekursywny jest 6-biegunowym filtrem Czebyszewa o zafalowaniach 0,5%. Jest to maksymalna liczba biegunów, jaką można zastosować przy częstotliwości odcięcia 0,05 i przy pojedynczej dokładności. W filtrze okienkowanym funkcją *sinc* zastosowano 1001-punktową odpowiedź impulsową, utworzoną przez splecenie 501-punktowej odpowiedzi impulsowej z samą sobą. Jak wykazano w rozdziale 16, ta operacja daje zwiększenie tłumienia w paśmie zaporowym.

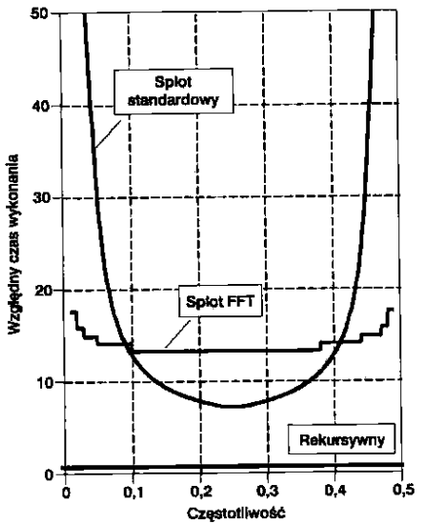
Jak więc wypada porównanie tych dwóch filtrów w sytuacji, gdy jest wymagane maksymalne działanie? Filtr okienkowany *sinc* pod tym względem wręcz miażdży Czebyszewa. Nawet gdyby udoskonalić filtry rekursywne (więcej biegunów, implementacja wielokrotna, podwójna dokładność itd.), to i tak nie dorównają one w działaniu filtrom SOI. To robi wrażenie, zwłaszcza jeśli uwzględnić, że filtry okienkowane funkcją *sinc* bardzo niedawno wkroczyły do walki. Filtry rekursywne mają bardzo ostre ograniczenia pod względem działania przy maksymalnych możliwych parametrach. Natomiast działanie filtrów okienkowanych funkcją *sinc* można „wyśrubować" do niewiarygodnego poziomu. Jest to oczywiście możliwe tylko pod warunkiem, że zechce się odpowiednio długo czekać na wynik. I tak dochodzimy do następnej krytycznej cechy – *szybkości działania*.

Porównywanie tych dwóch filtrów pod względem szybkości to jak porównanie wyścigowego Ferrari z gokartem. Na rys. 21.5 pokazano, o ile dłuższy jest czas wykonania w filtrze okienkowanym funkcją *sinc* niż w 6-biegunowym filtrze rekursywnym. Ponieważ filtr rekursywny ma większą szybkość opadania przy małych i dużych częstotliwościach, więc odpowiedź impulsowa filtru okienkowanego funkcją *sinc* musi być odpowiednio *wydłużona*, aby dorównać drugiemu filtrowi pod względem działania (tzn. aby porównanie było rzetelne). Z tego wynika dłuższy czas wykonania dla filtrów okienkowanych funkcji *sinc* przy częstotliwościach bliskich 0 lub 0,5. Na koniec ważny wniosek. Należy się spodziewać, że filtry SOI są o rząd wielkości wolniejsze niż porównywalne filtry NOI (gokart: 25 km/godz., ferrari: 250 km/godz.).



Rys. 21.4. Działanie filtrów SOI i NOI z maksymalnym wykorzystaniem ich parametrów. Odpowiedź częstotliwościowa filtru okienkowanego funkcją sinc może w zasadzie mieć dowolny kształt, a rekursywny filtr Czebyszewa jest pod tym względem bardzo ograniczony. Na rysunku porównano odpowiedzi częstotliwościowe 6-biegunowego, rekursywnego filtru Czebyszewa i 1001-punktowego filtru okienkowanego funkcją sinc

[17] 120 V, 60 Hz – napięcie skuteczne i częstotliwość w sieci energetycznej w USA (przyp. tłum.).



Rys. 21.5. Porównanie czasów wykonania filtrów SOI i NOI. Na wykresach pokazano względne czasy wykonania dla filtru okienkowanego funkcją sinc w porównaniu z równoważnym 6-biegunowym rekursywnym filtrem Czebyszewa. Wykresy dla filtru SOI dotyczą zarówno realizacji przez algorytm splotu standardowego, jak i splotu FFT. Czas wykonywania dla filtru okienkowanego funkcją sinc wzrasta przy małych i dużych częstotliwościach, ponieważ odpowiedź impulsowa musi być dłuższa, aby w tych obszarach częstotliwości dorównać lepszemu działaniu filtru rekursywnego. Ogólnie biorąc, filtry NOI są o rząd wielkości szybsze od filtrów SOI o porównywalnym działaniu

Pojedynek 3: Filtr o średniej kroczącej przeciw filtrowi jednobiegunowemu

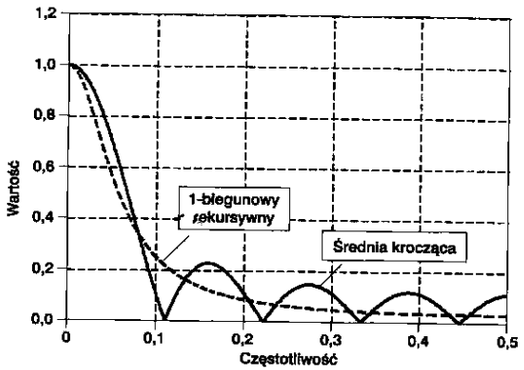
Nasz trzeci pojedynek będzie walką filtrów w dziedzinie czasu. Pierwszym zawodnikiem będzie 9-punktowy filtr o średniej kroczącej, a jego przeciwnikiem w tym pojedynku – jednobiegunowy filtr rekursywny działający z wykorzystaniem techniki dwukierunkowej. Aby osiągnąć porównywalną odpowiedź częstotliwościową, trzeba w filtrze jednobiegunowym zastosować współczynnik opadania przebiegu między kolejnymi próbkami równy $x = 0{,}70$. Walka rozpoczyna się na rys. 21.6, gdzie przedstawiono odpowiedzi częstotliwościowe obu filtrów. Żadna z nich nie jest imponująca, lecz oczywiście rozdzielanie pasm częstotliwości nie jest dziedziną zastosowania tych filtrów. Nie przyznajemy punktów żadnemu filtrowi.

Na rys. 21.7 przedstawiono odpowiedzi skokowe rywalizujących filtrów. Odpowiedź filtru o średniej kroczącej (rys. 21.7 a) jest linią prostą, a więc tworzy najszybszą drogę przejścia z jednego poziomu na inny. Odpowiedź filtru rekursywnego (rys. 21.7 b) jest łagodniejsza, co może być korzystniejsze w niektórych zastosowaniach. Dajemy po jednym punkcie dla każdego filtru.

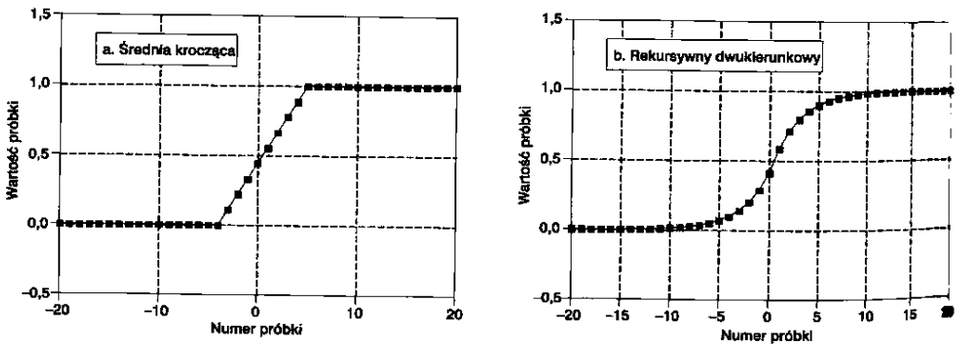
Porównywane filtry są w zasadzie równorzędne pod względem działania i wybór jednego z nich jest często wynikiem osobistych preferencji projektanta. Są jednak dwa przypadki, gdzie jeden filtr ma niewielką przewagę nad drugim. Chodzi o takie sytuacje, gdy jest konieczny kompromis między czasem *opracowania* i czasem *wykonania*. W pierwszym z tych przypadków chcecie skrócić czas opracowania filtru i jesteście skłonni zaakceptować jego wolniejsze działanie. Może na przykład być potrzebny filtr o kilku tysiącach punktów do jed-

norazowego zastosowania. Ponieważ wykonanie całego programu trwa tylko kilka sekund, więc nie ma sensu poświęcać wiele czasu na optymalizację algorytmu. Z całą pewnością zostaną tu użyte obliczenia zmiennoprzecinkowe. Do wyboru macie filtr o średniej kroczącej realizowany przez splot lub jednobiegunowy filtr rekursywny. Zwycięzcą jest tutaj filtr rekursywny. Będzie on nieco łatwiejszy do zaprojektowania, a także do modyfikacji i będzie działać znacznie szybciej.

Drugi przypadek jest przeciwieństwem pierwszego; filtr musi działać możliwie jak najszybciej i aby to osiągnąć, chcecie poświęcić trochę dodatkowego czasu na projektowanie. Ten filtr może na przykład być częścią produktu komercyjnego, z potencjalną możliwością powtarzania *miliony* razy. Przypuszczalnie zastosujecie liczby całkowite, aby osiągnąć możliwie największą szybkość. W tym przypadku wasz wybór padnie na filtr o średniej kroczącej, realizowany przez *rekursję* lub jednobiegunowy filtr rekursywny realizowany z wykorzystaniem tablic przeglądowych albo obliczeń na liczbach całkowitych. Zwycięzcą jest filtr o średniej kroczącej. Będzie on działał szybciej i podczas projektowania i wykonywania nie będzie podatny na problemy związane z arytmetyką liczb całkowitych.



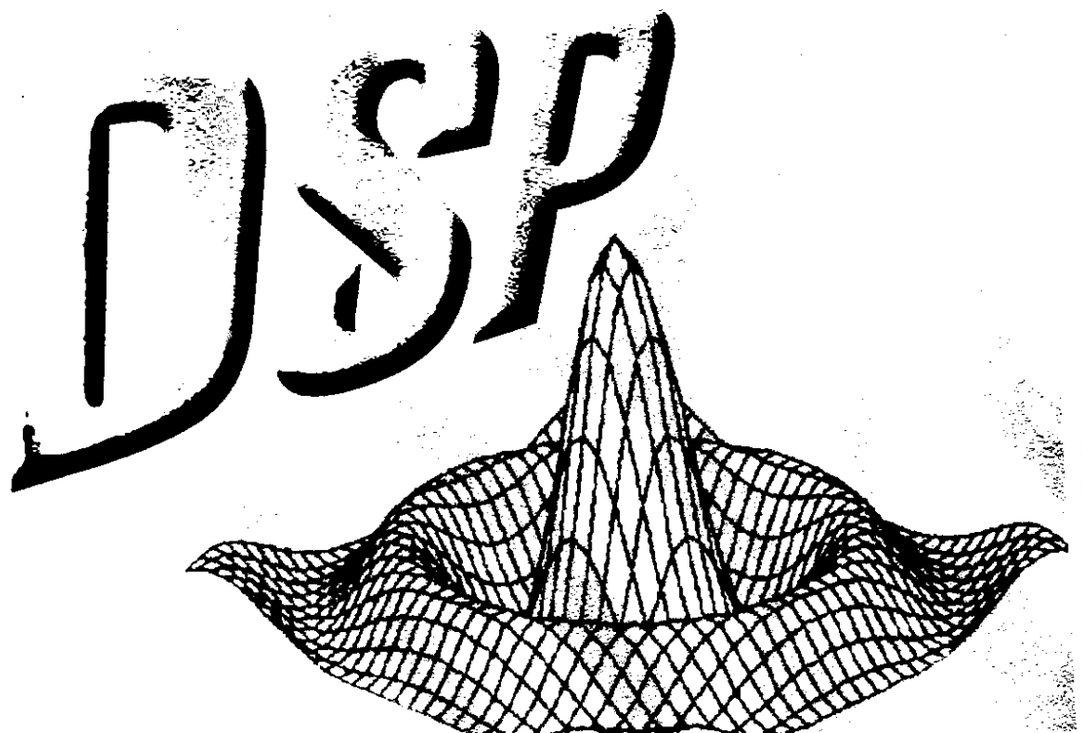
Rys. 21.6. Odpowiedzi częstotliwościowe filtru o średniej kroczącej i jednobiegunowego. Oba filtry mają marną odpowiedź częstotliwościową, czego można było się spodziewać po filtrach przeznaczonych do pracy w dziedzinie czasu



Rys. 21.7. Odpowiedzi skokowe filtrów: o średniej kroczącej i jednobiegunowego dwukierunkowego. Odpowiedź skokowa filtru o średniej kroczącej następuje podczas mniejszej liczby próbek, odpowiedź zaś filtru jednobiegunowego jest łagodniejsza

22

Przetwarzanie sygnałów fonicznych



Przetwarzanie sygnałów fonicznych, czyli sygnałów muzyki i mowy, obejmuje różno dz ny; wszystkie one są związane z prezentacją dźwięku słuchaczom. Trzy obszary zastos są szczególnie ważne: 1) *odtwarzanie muzyki z bardzo wysoką wiernością*, np. z płyt kom paktowych audio, 2) *telekomunikacja głosowa* oraz 3) *synteza mowy*, a więc komputer generacja i rozpoznawanie ludzkiego głosu. Chociaż te aplikacje mają różne cele i wyma w nich odmienne problemy, to we wszystkich przypadkach wytwarzane sygnały są odbier ne przez ludzkie ucho. Cyfrowe przetwarzanie sygnałów wprowadziło rewolucyjne zm w tych i innych dziedzinach cyfrowej techniki fonicznej.

Słuch ludzki

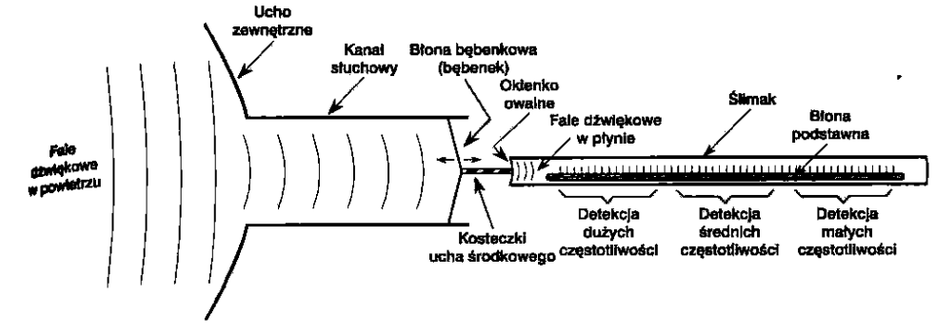
Ludzkie ucho jest organem wyjątkowo skomplikowanym. Zrozumienie jego działania utrud nia dodatkowo fakt, że informacje z *obu uszu* łączą się ze sobą w bardzo zawikłanej sie neuronowej, jaką jest ludzki mózg. Trzeba podkreślić, że zamieszczone poniżej omówie jest bardzo skrótowe. W ludzkim słyszeniu jest bowiem wiele zjawisk bardzo złożonych i w pełni jeszcze poznanych.

Na rys. 22.1 przedstawiono główne elementy i procesy tworzące ucho człowieka. *Ucho wnętrzne* składa się z dwóch części – widocznej małżowiny usznej utworzonej z chrząst okrytej skórą oraz z *przewodu słuchowego* o średnicy ok. 0,5 cm wchodzącego na ok. 3 c w głąb czaszki. Te struktury kierują zewnętrzny dźwięk do czułych organów – *ucha środ wego* i *wewnętrznego*, ukrytych w głębi czaszki. Na końcu przewodu słuchowego znajdu się napięta *błona bębenkowa*. Fale dźwiękowe, uderzając w błonę bębenkową, wywołu jej drgania. Ucho środowe zawiera kosteczki słuchowe przenoszące te drgania do *ślimak* (w uchu wewnętrznym), gdzie są przetwarzane na impulsy neuronowe. Ślimak jest wypełn ną płynem rurką o średnicy ok. 2 mm i długości 3 cm. Chociaż na rys. 22.1 pokazano ślima jako prosty, w rzeczywistości jest spiralny i wygląda jak mała muszla ślimaka.

Gdy fala dźwiękowa próbuje przejść z powietrza do płynu, tylko mała część dźwięku prz chodzi przez tę granicę, reszta energii dźwięku ulega odbiciu. Przyczyną jest to, że powietrz ma *małą* impedancję mechaniczną (małe ciśnienie akustyczne i duża prędkość ruchu czą tek spowodowana małą gęstością i dużą ściśliwością), ciecz zaś ma *dużą* impedancję me chaniczną. Wyrażając się mniej technicznie, można powiedzieć, że machanie ręką w wodz wymaga więcej wysiłku niż w powietrzu. Ta różnica impedancji mechanicznych powodu że większość dźwięku jest odbijana od granicy powietrze/ciecz.

Ucho środkowe jest elementem *dopasowującym impedancje*, zwiększającym ułamek energ dźwięku przechodzącej do płynu w uchu wewnętrznym. Ryby, na przykład, nie mają a błony bębenkowej, ani ucha środkowego, gdyż nie mają potrzeby słuchania w powietrz Dopasowanie impedancji wynika w głównej mierze z różnic *powierzchni* błony bębenko wej (odbierającej dźwięk z powietrza) i *okienka owalnego* (przekazującego dźwięk do płyn w ślimaku, patrz rys. 22.1). Błona bębenkowa ma powierzchnię ok. 60 mm^2, a okienko owa ne – ok. 4 mm^2. Ciśnienie jest stosunkiem siły do powierzchni, dlatego ta różnica powierzch ni powoduje ok. 15-krotne zwiększenie ciśnienia fali dźwiękowej.

W ślimaku znajduje się błona podstawna, stanowiąca podstawę dla ok. 12 000 receptoró słuchowych tworzących *nerw ślimakowy*. Błona podstawna jest najsztywniejsza w pobliu okienka owalnego i staje się coraz bardziej elastyczna w kierunku końca. Dzięki temu błon działa jak *analizator widma częstotliwościowego*. Błona podstawna, poddana oddziały sygnału o dużej częstotliwości, drga w swojej sztywnej części, pobudzając komórki nerwow w pobliżu okienka owalnego. Małe częstotliwości zaś pobudzają komórki nerwowe w ku cowej części błony. To powoduje, że specyficzne włókna w nerwie ślimakowym reagują n pewne określone częstotliwości. Jest to tzw. **teoria miejsca** (*place principle*), zgodnie z któ największe znaczenie w wytwarzanym wrażeniu wysokości dźwięku ma miejsce pobudze ślimaka (miejsce na błonie podstawnej).



Rys. 22.1. Schemat funkcjonalny ludzkiego ucha. Ucho zewnętrzne zbiera fale dźwiękowe z otoczenia i kieruje je do *zewnętrznej błony bębenkowej, która drga synchronicznie z falą dźwiękową. Kosteczki słuchowe w uchu środkowym (młoteczek, kowadełko i strzemiączko) przekazują te wibracje do okienka owalnego, w którym stopa strzemiączka drga, pobudzając do drgań płyn wypełniający ślimak. W ślimaku znajduje się błona podstawna, będącą podstawą dla ok. 12 000 komórek nerwowych[18] tworzących nerw ślimakowy. Dzięki temu, że sztywność błony podstawnej zmienia się wzdłuż jej długości, każdy komórka nerwowa reaguje tylko na wąski zakres częstotliwości dźwięku. W ten sposób ucho jest analizatorem widma częstotliwościowego*

W ludzkim słyszeniu występuje też jeszcze inny mechanizm kodowania, zwany **teorią czasu** (*volley priciple*). Komórki nerwowe przesyłają informacje przez wytwarzanie krótkich impulsów elektrycznych, zwanych *potencjałami czynnościowymi*. Komórka nerwowa na błonie podstawnej może kodować informację foniczną, wytwarzając potencjał czynnościowy w odpowiedzi na każdy okres drgań. Na przykład fala dźwiękowa 200 Hz może być odbierana przez neuron generujący 200 potencjałów czynnościowych na sekundę. Taki mechanizm działania dotyczy tylko częstotliwości poniżej ok. 500 Hz, a więc maksymalnej częstotliwości, z jaką neurony mogą generować potencjały czynnościowe. Ucho ludzkie pokonuje tę trudność przez umożliwienie realizacji każdego takiego pojedynczego zadania na przemian przez kilka komórek nerwowych. Na przykład ton 3000 Hz może być odbierany przez *dziesięć* komórek nerwowych na przemian pobudzanych 300 razy na sekundę. To rozciąga zakres obowiązywania teorii czasu do częstotliwości ok. 4 kHz, powyżej której jest stosowana wyłącznie teoria miejsca.

W tablicy 22.1 przedstawiono zależność między natężeniem dźwięku a odczuwalną głośnością. Przyjęto wyrażać natężenie dźwięku w skali logarytmicznej w **decybelach**. W tej skali 0 dB jest falą dźwiękową o gęstości mocy 10^{-16} W/cm^2, jest to w przybliżeniu najniższy poziom natężenia dźwięku wykrywany przez ludzkie ucho. Normalna mowa ma ok. 60 dB, a bolesne uszkodzenie ucha następuje przy poziomie ok. 140 dB.

Różnica w natężeniu najgłośniejszego i najcichszego dźwięku, słyszalnego ludzkim uchem, wynosi ok. 120 dB, co pod względem amplitudy daje stosunek jeden do miliona. Słuchacze mogą odczuć *zmianę* głośności, gdy poziom natężenia dźwięku zmienia się o ok. 1 dB (12% zmiany amplitudy). Tak więc jest odczuwanych tylko ok. 120 poziomów natężenia dźwięku – od najcichszego szeptu do najgłośniejszego grzmotu. Czułość ucha jest zadziwiająca, przy słyszeniu bardzo słabych dźwięków bębenek w uchu drga z amplitudą mniejszą od średnicy pojedynczej cząsteczki!

Percepcja głośności zależy od poziomu natężenia dźwięku ze współczynnikiem proporcjonalności w przybliżeniu równym $10^{1/3}$. Jeśli na przykład zwiększamy poziom natężenia dźwięku o czynnik *dziesięć*, to słuchacze odniosą wrażenie, że głośność wzrosła o czynnik ok. *dwa* ($10^{1/3} \approx 2$). Z tego wynika główna trudność eliminowania niepożądanych dźwięków z otoczenia, np. hałaśliwej muzyki u sąsiadów. Powiedzmy, że starannie pokryjecie 99% waszych ścian materiałem

[18] Na błonie podstawnej znajduje się ok. 28 000 zewnętrznych i ok. 3500 wewnętrznych komórek rzęsatych różniących się morfologicznie i funkcją przy odbiorze dźwięku (przyp. tłum.).

Tablica 22.1. Jednostki natężenia dźwięku. Natężenie dźwięku jest wyrażane jako moc na jednostkę powierzchni (np. W/cm²) lub – częściej – w skali logarytmicznej w decybelach. Jak wynika z tablic, słuch ludzki ma największą czułość w zakresie między 1 kHz a 4 kHz

Głośniej → Ciszej	10^{-2}	140 dB	Ból
	10^{-3}	130 dB	
	10^{-4}	120 dB	Nieprzyjemne odczucie
	10^{-5}	110 dB	Wiertarka udarowa i koncert rockowy
	10^{-6}	100 dB	
	10^{-7}	90 dB	Granica dopuszczalnego hałasu przemysłowego (wg norm USA)
	10^{-8}	80 dB	
	10^{-9}	70 dB	
	10^{-10}	60 dB	Zwykła rozmowa
	10^{-11}	50 dB	
	10^{-12}	40 dB	Najsłabszy dźwięk słyszalny przy 100 Hz
	10^{-13}	30 dB	
	10^{-14}	20 dB	Najsłabszy dźwięk słyszalny przy 10 kHz
	10^{-15}	10 dB	
	10^{-16}	0 dB	Najsłabszy dźwięk słyszalny przy 3 kHz
	10^{-17}	–10 dB	
	10^{-18}	–20 dB	

doskonale dźwiękochłonnym, zostawiając tylko 1% powierzchni niepokrytej na drzwi, narożnik wentylatory itd. Mimo że zredukowaliście poziom natężenia dźwięku do zaledwie 1% poprzedniej wartości, to odczuwalna głośność zmniejszyła się tylko do ok. $0{,}01^{1/3} \approx 0{,}2$, czyli do 20%.

Ogólnie przyjmuje się, że zakres ludzkiego słyszenia obejmuje częstotliwości od 20 Hz do 20 kHz, lecz z największą czułością w zakresie od 1 do 4 kHz. Człowiek odczuwa już dźwięki o poziomie natężenia 0 dB przy częstotliwości 3 kHz, lecz trzeba poziomu natężenia aż 40 dB, aby usłyszeć dźwięk o częstotliwości 100 Hz (amplituda wzrasta stukrotnie). Słuchający może stwierdzić, że dwa tony różnią się wysokością, jeśli różnica ich częstotliwości jest co najmniej 0,3% przy częstotliwości 3 kHz. Ta wymagana różnica wzrasta do 3% dla częstotliwości 100 Hz. Dla porównania warto wiedzieć, że częstotliwość tonów dwóch sąsiednich klawiszy fortepianu różni się o ok. 6%.

Główną zaletą posiadania *dwojga* uszu jest możliwość ustalania *kierunku* dźwięku. Człowiek jest w stanie wykryć różnicę między dwoma źródłami o kierunkach dochodzenia dźwięku różniących się zaledwie o 3 stopnie, a więc źródeł rozstawionych mniej więcej na szerokość człowieka w odległości 10 m. Informacja kierunkowa jest uzyskiwana dwiema oddzielnymi drogami. Po pierwsze, sygnały o częstotliwości powyżej ok. 1 kHz są silnie *zacieniane* przez głowę. Inaczej mówiąc, ucho znajdujące się od strony źródła dźwięku odbiera silniejszy sygnał niż ucho po przeciwnej stronie głowy. Drugi wskaźnik kierunkowości polega na tym, że ucho położone z bardziej odległej strony głowy odbiera dźwięk nieco *później* niż drugie ucho, ze względu na większą odległość od źródła dźwięku. Przyjmując typowy rozmiar głowy (ok. 22 cm) i prędkość rozchodzenia się dźwięku (ok. 340 m/s), można obliczyć, że rozróżnialność kierunku wynosząca 3 stopnie wymaga dokładności określania czasu ok. 30 mikrosekund. Ponieważ takie dokładne rozróżnianie czasu wymaga mechanizmu opartego na teorii czasu, więc ten wskaźnik kierunkowości jest wykorzystywany przede wszystkim do dźwięków o częstotliwości poniżej ok. 1 kHz.

Oba te mechanizmy wyznaczania kierunkowości są w dużym stopniu wspomagane obserwowaniem zmian sygnału przy poruszaniu głową. Ciekawe odczucia mają miejsce wówczas, gdy słuchacz odbiera dwa zupełnie identyczne dźwięki w obu uszach, np. słuchając obu uszami jednakowego dźwięku doprowadzonego do słuchawek nagłownych. Mózg wnioskuje wtedy, że dźwięk dochodzi z punktu położonego w środku głowy słuchacza!

Ludzki słuch może określać *kierunek* dochodzenia dźwięku, lecz słabo identyfikuje *odległość* do źródła dźwięku. W fali dźwiękowej jest bowiem niewiele wskazówek zawierających informacje na ten temat. Ludzki słuch w pewnym stopniu reaguje na dźwięki o dużej częstotliwości jako na dochodzące z bliska, a na dźwięki o małych częstotliwościach – jako na dochodzące z daleka. Przyczyną takiego odczuwania jest rozpraszanie fal dźwiękowych o większych częstotliwościach podczas rozprzestrzeniania się dźwięku na duże odległości. Inną, raczej słabą, wskazówką dotyczącą odległości jest zawartość echa w sygnale, jeśli mówimy o odbiorze na odległościach w skali rozmiarów pokoju. Na przykład dźwięk w dużym audytorium zawiera echo o opóźnieniu ok. 100 ms, a w małym pokoju biurowym – typowo ok. 10 ms. Niektóre żywe organizmy rozwiązały ten problem, stosując *sonar aktywny*, czyli *echolokację*. Nietoperze i delfiny wydają z siebie mlaśnięcia i piski, które odbijają się od pobliskich obiektów. Mierząc czas między wysłanym sygnałem i odebranym echem, te zwierzęta mogą lokalizować obiekty z rozdzielczością ok. 1 cm. Eksperymenty wykazały, że niektórzy ludzie, zwłaszcza niewidomi, potrafią też, w niewielkim stopniu, korzystać z echolokacji.

Barwa dźwięku

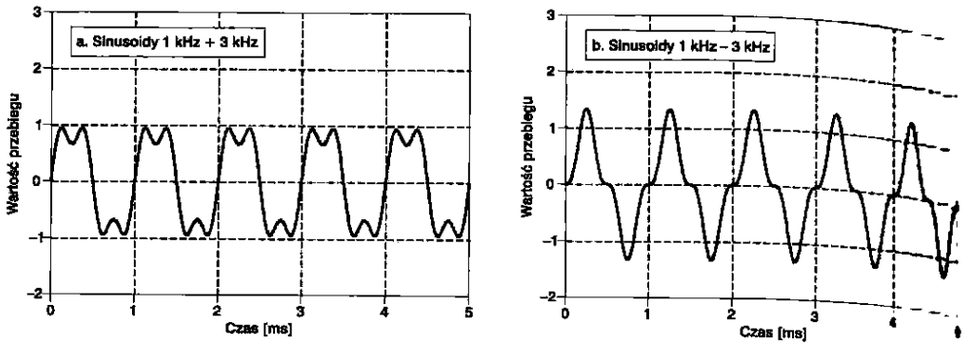
Percepcja ciągłego dźwięku, takiego jak ton instrumentu muzycznego, jest często określana trzema subiektywnymi cechami dźwięku: jego **głośnością, wysokością i barwą**. *Głośność* jest miarą omówionego już poprzednio natężenia fali dźwiękowej. *Wysokość dźwięku* odpowiada częstotliwość składowej podstawowej w fali dźwiękowej. Wprawdzie istnieją pewne trudno uchwytne zjawiska w odbiorze tych właściwości dźwięku, są one jednak łatwo charakteryzowane przez odpowiednie wielkości fizyczne[19].

Bardziej skomplikowanym problemem jest *barwa* dźwięku, którą określa się przez *zawartość harmonicznych* w sygnale. Na rys. 22.2 przedstawiono dwa przebiegi, każdy utworzony przez zsumowanie fali sinusoidalnej 1 kHz o amplitudzie *jeden* z falą sinusoidalną 3 kHz o amplitudzie równej *jedna druga*. Różnica między oboma przebiegami polega na tym, że przy tworzeniu przebiegu na rys. 22 b) sinusoidę o większej częstotliwości przed sumowaniem poddano *inwersji*. Inaczej to wyrażając: trzecia harmoniczna (3 kHz) jest przesunięta w fazie o 180° w stosunku do pierwszej (1 kHz). Mimo tak różniących się od siebie przebiegów w dziedzinie czasu, te sygnały – jako dźwięk – brzmią *identycznie*, jako że słyszenie jest związane z odbiorem amplitud składowych o różnych częstotliwościach, a jest całkiem nieczułe na ich *fazę*. *Kształt* przebiegu w dziedzinie czasu ma tylko pośredni związek ze słyszeniem i zwykle nie jest w ogóle uwzględniany w systemach audio.

Nieczułość ucha na fazę można zrozumieć, rozważając, jak dźwięk rozprzestrzenia się w środowisku. Powiedzmy, że słuchacie osoby mówiącej do was w małym pokoju. Do waszych uszu dociera, po części, dźwięk bezpośredni, a po części – odbity od ścian, sufitu i podłogi. Ponieważ rozprzestrzenianie się dźwięku (takie właściwości jak tłumienie, odbicie i rezonans) zależy od częstotliwości, więc składowe o różnych częstotliwościach docierają do waszych uszu różnymi drogami. To powoduje, że względne fazy każdej składowej częstotliwościowej zmieniają się, gdy poruszacie się po pokoju. Ucho nie uwzględnia tych zmian fazy, dlatego odbieracie głos jako *niezmienny*, gdy zmieniacie swoje położenie. Z punktu widzenia fizyki faza sygnału fonicznego nabiera charakteru losowego w miarę rozprzestrzeniania się sygnału w złożonym środowisku. Inaczej mówiąc, ucho jest nieczułe na fazę, gdyż zawiera ona mało pożytecznej informacji[20].

[19] Odpowiednio: natężenie, częstotliwość podstawowa i widmo częstotliwościowe, które są obiektywnymi cechami dźwięku (przyp. tłum.).

[20] Badania wskazują jednak na wrażliwość ucha na fazę, która jest różna w zależności od konkretnej struktury widmowej dźwięku (przyp. tłum.).



Rys. 22.2. Detekcja fazy przez ludzkie ucho. Ludzkie ucho jest nieczułe na względną fazę sinusoid składowych. Pokazane na rysunku przykładowe przebiegi będą brzmiały jednakowo, gdyż amplitudy ich składowych są jednakowe, mimo jeśli ich względne fazy się różnią

Nie można jednak powiedzieć, że ucho jest całkiem głuche na fazę. Zmiana fazy może bowiem inaczej ustawiać *sekwencję czasową* w sygnale fonicznym. Przykładem jest system świergotowy (rozdział 11) zmieniający impuls w sygnał o znacznie dłuższym czasie trwania. Są to jednak na ogół osobliwości, a nie zjawiska występujące przy słuchaniu w normalnym środowisku.

Przypuśćmy, że poprosiliśmy skrzypka, aby zagrał jakiś ton, np. *A* poniżej środkowego *C*[21]. Gdy wyświetlimy ten przebieg na oscyloskopie, wygląda on jak przebieg piłokształtny pokazany na rys. 22.3 a). Jest to spowodowane lepką kalafonią, jaką skrzypek posmarował włosy swojego smyczka. Pokazany na rysunku przebieg powstaje w wyniku pociągania smyczkiem po strunie, potem pociągania w drugim kierunku i ewentualnego odrywania smyczka od struny. Ten cykl ciągle się powtarza, dając przebieg piłokształtny.

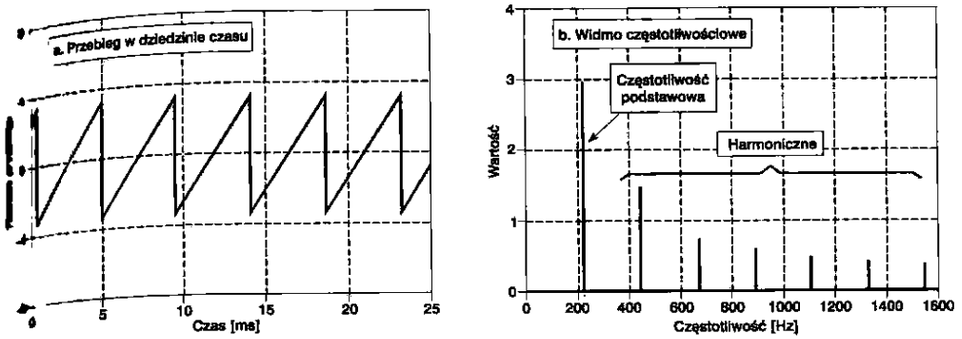
Na rys. 22.3 b) przedstawiono, jak ten dźwięk jest odbierany przez ucho jako sygnał o częstotliwości 220 Hz z harmonicznymi 440, 660, 880 Hz itd. Gdyby ten ton zagrano na jakimś innym instrumencie, przebieg miałby inny *wygląd*; ucho jednak też by słyszało częstotliwość 220 Hz i jej harmoniczne. Oba instrumenty brzmią podobnie, gdyż wytwarzają taką samą częstotliwość podstawową zagranego tonu. Mówi się więc, że dają dźwięki o takiej samej *wysokości*. Ponieważ jednak względne amplitudy *harmonicznych* są różne, więc instrumenty mają inną *barwę* dźwięku.

Często się sądzi, że barwa dźwięku zależy od kształtu przebiegu. Jest to stwierdzenie słuszne, choć nieco mylące. Percepcja barwy dźwięku jest wynikiem detekcji harmonicznych przez ucho. Zawartość harmonicznych jest określona kształtem przebiegu, lecz nieczułość ucha na fazę czyni tę zależność bardzo jednostronną. To znaczy, że określony przebieg daje tylko jedną barwę dźwięku, lecz określona barwa dźwięku może być rezultatem nieskończonej liczby różnych przebiegów.

Ucho jest bardzo przyzwyczajone do słuchania dźwięku o częstotliwości podstawowej z harmonicznymi. Dźwięk będący kombinacją sinusoid o częstotliwościach 1 kHz i 3 kHz wydaje się słuchaczowi naturalny i miły. Dźwięk zaś złożony z sinusoid 1 kHz i 3,1 kHz wydaje się nieprzyjemny.

Taki odbiór dźwięku jest podstawą standardowej skali muzycznej, którą przy użyciu klawiatury fortepianu zilustrowano na rys. 22.4. Uderzenie w klawisz położony najbardziej na lewo wytwarza częstotliwość podstawową 27,5 Hz z harmonicznymi 55, 110, 440, 880 Hz itd. (są tam też harmoniczne leżące pomiędzy tymi wartościami, lecz nie są one istotne w tych rozważaniach). Harmoniczne są zgodne z częstotliwościami podstawowymi wytwarzanymi przy uderzeniach w inne klawisze. Ściślej mówiąc, każdy *co siódmy* biały klawisz daje harmoniczną dźwięku dostarczanego przez skrajny, lewy klawisz. Tak więc ósmy klawisz od lewej strony daje częstotliwość

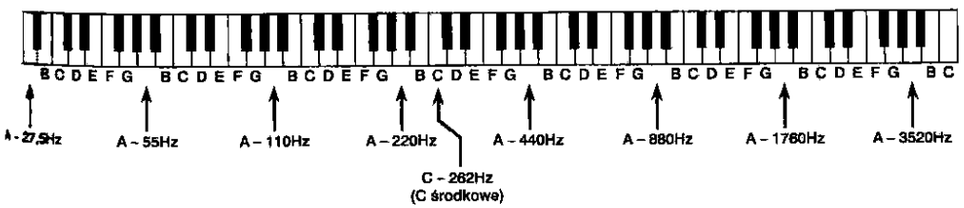
[21] W polskim nazewnictwie środkowe *C* jest określane jako *c'*, a *A* poniżej jako *a* (przyp. tłum.).



Rys. 22.3. Przebieg sygnału dźwiękowego skrzypiec. Skrzypce dają przebieg piłokształtny (rys. a). Dźwięk słyszany takim pokazano na rys. b), jest to częstotliwość podstawowa z harmonicznymi

podstawową 55 Hz, piętnasty – 110 Hz itd. Dźwięki odpowiadające tym klawiszom, będące wzajemnie swymi harmonicznymi, brzmią podobnie do siebie, a zagrane unisono dźwięczą harmonijnie i melodyjnie. Dlatego *wszystkie* one są nazywane dźwiękiem *A*. W ten sam sposób biały klawisz znajdujący się bezpośrednio na prawo od każdego z klawiszy *A* jest nazywany *B*, i wszystkie klawisze *B* dają dźwięki będące względem siebie harmonicznymi. Ten sposób oznaczania powtarza się dla siedmiu dźwięków: *A*, *B*, *C*, *D*, *E*, *F* i *G*. (W Polsce stosuje się nieco inne nazewnictwo, podstawowa różnica polega na tym, że *B* jest zastąpione przez *H* – przyp. tłum.).

Pojęcie **oktawa** wyraża *czynnik dwa na skali częstotliwości*. Na klawiaturze fortepianu jedna oktawa obejmuje osiem białych klawiszy i stąd pochodzi jej nazwa (*octo* to po łacinie *osiem*). Inaczej mówiąc, częstotliwość dźwięku wytwarzanego przez fortepian podwaja się co siódmy biały klawisz, a cała klawiatura fortepianu obejmuje trochę ponad siedem oktaw. Zakres słuchu ludzkiego ogólnie się przyjmuje jako obejmujący częstotliwości od 20 Hz do 20 kHz, co wykracza o ok. 1/2 oktawy na lewo i o dwie oktawy na prawo poza skalę klawiatury fortepianu. Ponieważ oktawy oznaczają podwajanie częstotliwości co pewną określoną liczbę klawiszy, więc wyrażają one częstotliwość w skali *logarytmicznej*. Jest to ważne, gdyż informacja o sygnale fonicznym charakteryzuje się też na ogół takim rozkładem. Na przykład tyle samo tej informacji jest przenoszonych w oktawie od 50 do 100 Hz, co w oktawie od 10 do 20 kHz. Chociaż fortepian pokrywa tylko ok. 20% zakresu częstotliwości dźwięku słyszanego przez człowieka (4 kHz z zakresu 20 kHz), to jednak może wytwarzać więcej niż 70% informacji, jaką człowiek może odebrać (7 z 10 oktaw). Trzeba też uwzględnić, że największa częstotliwość dźwięku słyszanego przez człowieka zmniejsza się od 20 do 10 kHz w miarę trwania życia. Jest to wszakże strata tylko ok. 10% zdolności słyszenia (jedna oktawa z dziesięciu). Jak dalej będzie wykazane, ten logarytmiczny rozkład informacji ma bezpośredni wpływ na wymaganą częstotliwość próbkowania sygnałów fonicznych.



Rys. 22.4. Klawiatura fortepianu. Jest ona logarytmiczną skalą częstotliwości, na której częstotliwość podstawowa podwaja się co siedem białych klawiszy. Te białe klawisze odpowiadają dźwiękom A, B, C, D, E, F i G. Jest to nazewnictwo anglosaskie. W Polsce stosuje się nieco inne – m.in. zamiast oznaczenia B używa się H (przyp. tłum.).

Zależność jakości dźwięku od przepływności bitowej

Przy projektowaniu cyfrowego systemu audio nasuwają się dwa pytania: 1) Jak dobra powinna być jakość dźwięku? oraz 2) jaka jest dopuszczalna bitowa przepływność danych? Odpowiedź na te pytania mieści się zazwyczaj w jednej z trzech kategorii. Pierwsza, **muzyka o wysokiej wierności odtwarzania**, gdzie jakość dźwięku ma pierwszorzędne znaczenie i prawie każda przepływność danych jest akceptowalna. Druga kategoria to **łączność telefoniczna**, wymagająca naturalnego brzmienia mowy oraz małej przepływności w celu ograniczenia kosztów sytemu. Do trzeciej kategorii należy mowa **poddana kompresji**, gdzie zredukowanie bitowej przepływności danych jest bardzo ważne, może zaś być dopuszczalna pewna nienaturalność brzmienia mowy, ale przy zachowaniu jej zrozumiałości. Obejmuje to łączność wojskową, telefonię komórkową oraz sygnał mowy przechowywany w formie cyfrowej do wykorzystania w poczcie głosowej i multimediach.

W tablicy 22.2 pokazano, jak w trzech wymienionych kategoriach uzyskuje się kompromisowy doboru parametrów – jakości dźwięku i bitowej przepływności danych. W systemach muzycznych o wysokiej wierności odtwarzania próbkowanie musi być odpowiednio szybkie (z częstotliwością 44,1 kHz) i kwantowanie próbek dokładne (rozdzielczość 16-bitowa), aby można było wychwycić wszystkie dźwięki słyszalne dla człowieka. Taką wspaniałą jakość dźwięku uzyskuje się za cenę dużej przepływności bitowej 44,1 kHz × 16 bitów = 706 kbit/s (dla 2 kanałów 1411,2 kbit/s – przyp. tłum.).

Sygnał muzyczny wymaga pasma 20 kHz, podczas gdy sygnał mowy tylko 3,2 kHz. Chociaż pasmo częstotliwości ograniczono do zaledwie 16% (3,2 kHz w stosunku do 20 kHz), to sygnał nadal zawiera 80% informacji zawartej w sygnale oryginalnym (8 z 10 oktaw). Typowa częstotliwość próbkowania w systemach telekomunikacyjnych wynosi 8 kHz, co wystarcza do naturalnego brzmienia mowy, lecz znacznie pogarsza jakość muzyki. Na pewno te różnice w jakości muzyki nie są wam obce: stacje radiowe FM (z modulacją częstotliwości) nadają z pasmem sygnału fonicznego prawie 20 kHz, a stacje AM (z modulacją amplitudy) mają to pasmo ograniczone do ok. 3,2 kHz. Na stacjach AM głos brzmi dobrze, lecz jakość muzyki nie jest zadowalająca.

W systemach przekazujących tylko sygnał mowy ogranicza się też dokładność z 16 do 12 bitów na próbkę, co powoduje mało zauważalną zmianę w jakości dźwięku. Przy zastosowaniu kwantyzacji nierównomiernej można liczbę bitów zredukować nawet do 8 na próbkę. Jest to szeroko rozpowszechniona technika zwana **kompansją** (*companding*), czyli technika kompresji połączonej z ekspansją. Omówimy ją w dalszej części tego rozdziału. Częstotliwość próbkowania 8 kHz i rozdzielczość przetwornika a/c 8 bitów/próbkę dają w rezultacie bitową przepływność danych 64 kbit/s. Naturalne brzmienie mowy można jednak uzyskać przy znacząco mniejszej przepływności. Warto zwrócić uwagę, że mowa wymaga mniej niż 10% bitowej przepływności danych koniecznej dla muzyki o wysokiej wierności.

Bitowa przepływność danych 64 kbit/s jest wynikiem bezpośredniego zastosowania tego próbkowania i kwantyzacji do sygnałów fonicznych. W metodach dalszego zmniejszania przepływności korzysta się z *kompresji* strumienia danych przez usuwanie redundancji z sygnału mowy. Kompresja danych będzie przedmiotem rozważań w rozdziale 27. Jednym z najbardziej skutecznych sposobów kompresji sygnału fonicznego jest **liniowe kodowanie predykcyjne** (**LPC** – *Linear Predictive Code*). Ta metoda ma kilka odmian i podgrup. Zależnie od wymaganej jakości mowy można metodą LPC zredukować przepływność danych do zaledwie 2-6 kbit/s. Do kompresji LPC powrócimy jeszcze w tym rozdziale przy omawianiu *syntezy* mowy.

Tablica 22.2. Zależność jakości dźwięku od bitowej przepływności danych. Jakość dźwięku zależy od formatu cyfrowego sygnału fonicznego, czyli od przepływności bitowej będącej iloczynem częstotliwości próbkowania i liczby bitów na próbkę. Można tu wyróżnić trzy kategorie sygnałów: muzyka o wysokiej wierności odtwarzania (705,6 kbit/s), mowa o jakości odpowiedniej do łączności telefonicznej (64 kbit/s) i mowa poddana kompresji (4 kbit/s)

Wymagana jakość dźwięku	Pasmo	Częstotliwość próbkowania	Liczba bitów	Bitowa przepływność danych [bit/s]	Komentarze
Muzyka o wysokiej wierności odtwarzania (płyta kompaktowa)	od 5 Hz do 20 kHz	44,1 kHz	16	705,6 k (jeden kanał)	Zadowala nawet najwybredniejszych audiofilów. Jakość lepsza niż możliwości ludzkiego słuchu.
Jakość mowy odpowiednia do łączności telefonicznej	od 200 Hz do 3,2 kHz	8 kHz	12	96 k	Dobra jakość mowy, lecz bardzo słaba muzyki.
Jakość mowy odpowiednia do łączności telefonicznej (z kompansją)	od 200 Hz do 3,2 kHz	8 kHz	8	64 k	Nieliniowy przetwornik a/c zmniejsza bitową przepływność danych o 50%. Technika powszechnie stosowana.
Sygnał mowy kodowany predykcyjnie (LPC)	od 200 Hz do 3,2 kHz	8 kHz	12	4 k	Kompresja sygnału mowy metodami DSP. Bardzo mała bitowa przepływność danych, słaba jakość mowy.

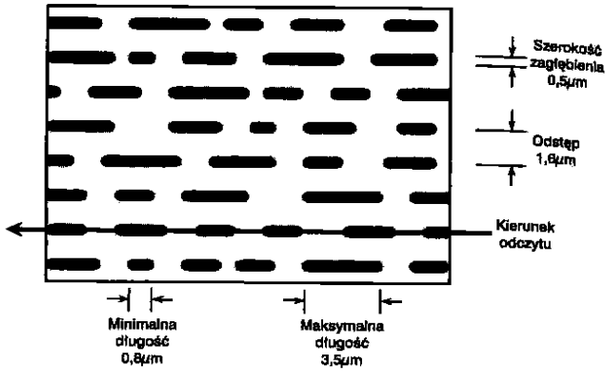
Technika audio o wysokiej wierności

Audiofile wymagają skrajnie wysokiej jakości dźwięku i wszystkie inne czynniki są traktowane jako drugorzędne. Gdybyśmy mieli jednym słowem opisać ich podejście do tej kwestii, powiedzielibyśmy – *przesada*. Zamiast dopasowywać takie systemy audio do możliwości ludzkiego ucha, projektuje się je tak, aby *przekraczały* granice słyszenia. Jest to jedyny sposób dający pewność, że odtwarzana muzyka jest całkowicie zgodna z oryginałem[22]. Cyfrowa technika audio weszła do masowego zastosowania dzięki **kompaktowej płycie laserowej, czyli CD**. Była to rewolucja w muzyce, gdyż jakość dźwięku z płyty kompaktowej znacznie przekracza jakość osiąganą ze starszych nośników, takich jak płyty gramofonowe i taśmy magnetofonowe. DSP stanowi przednią linię frontu tej rewolucji cyfrowej.

Na rys. 22.5 pokazano powierzchnię kompaktowej płyty laserowej widzianą przez mikroskop o dużym powiększeniu. Główna powierzchnia płyty jest błyszcząca (odbijająca światło), a informacja cyfrowa jest zapisana w postaci szeregu ciemnych zagłębień (*pits*) wytłoczonych w warstwie informacyjnej. Informacja jest rejestrowana na pojedynczej, spiralnej ścieżce – od środka ku brzegowi, a więc odwrotnie niż na płycie gramofonowej. Prędkość kątowa płyty zmienia się w miarę odczytu informacji od ok. 480 obr/min do ok. 240 obr/min. Dzięki temu jest utrzymywana stała prędkość liniowa wynosząca 1,2 m/s. Dla porównania warto przypomnieć, że dla płyt gramofonowych prędkość kątowa była stała, równa 33 1/3, 45 lub 78 obr/min. Podczas odtwarzania czujnik optyczny sprawdza, czy powierzchnia w danym miejscu odbija światło, czy nie i na tej podstawie jest odczytywana odpowiednia informacja binarna.

Jak wynika z geometrii pokazanej na rys. 22.5, na płycie CD gęstość zapisu informacji wynosi ok. 1 bit na μm^2, co odpowiada 1 milionowi bitów na 1 mm^2 i ok. 15 miliardom bitów na całej płycie. Są to podobne wymiary, jak używane przy produkcji układów scalonych,

[22] Nie można się zgodzić z tym stwierdzeniem autora (przyp. tłum.).

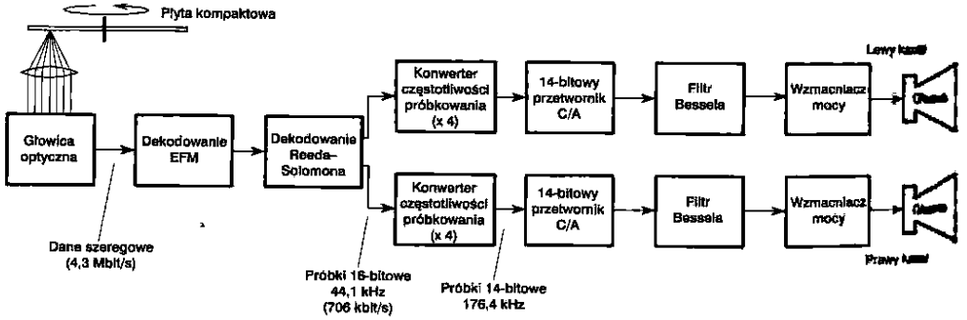


Rys. 22.5. Powierzchnia płyty kompaktowej. Na powierzchni płyty są wytłoczone zagłębienia o wymiarach mikronowych, reprezentujące zera i jedynki. Daje to gęstość zapisu danych 1 bit/μm² lub milion bitów/mm². Głębokość zagłębienia wynosi 0,16 μm

i to nie bez powodu. Jedną z właściwości światła jest niemożność jego zogniskowania do rozmiarów mniejszych od połowy długości fali, czyli ok. 0,3 μm. Ponieważ zarówno tłoczenie scalone, jak i płyty laserowe są tworzone metodami optycznymi, więc nieostrość nie może poniżej 0,3 μm ogranicza minimalne rozmiary.

Na rys. 22.6 przedstawiono schemat blokowy typowego odtwarzacza płyt kompaktowych. Bitowa przepływność danych w sygnale wynosi ok. 4,3 milionów bitów/s, co odpowiada 1 bitowi na 0,28 μm długości ścieżki. Pozostaje to jednak w konflikcie ze specyficzną geometrią CD. Każde zagłębienie nie powinno być krótsze niż 0,8 μm i nie dłuższe niż 3,5 μm. Inaczej mówiąc, każdy binarny stan *jeden* musi być częścią grupy od 3 do 13 *jedynek*. Dało to zaletę zmniejszenia stopy błędów powstających podczas pracy głowicy optycznej. Jak jednak dostosować dane cyfrowe do takiego dziwnego pogrupowania?

Odpowiedzią jest sposób kodowania, zwany **modulacją osiem-do-czternastu (EFM** – *Eight-to-Fourteen Modulation*)[23]. Zamiast bezpośredniej rejestracji bajtu danych na płycie, przepuszcza się 8 bitów przez tablicę przeglądową dającą w wyniku 14 bitów. Są one odpowiednio grupowane i zapisywane na płycie kompaktowej. Podczas odtwarzania wartości bitowe



Rys. 22.6. Schemat blokowy odtwarzacza płyt kompaktowych. Informacja cyfrowa jest odczytywana z płyty przy użyciu głowicy optycznej, następnie demodulowana przez dekodowanie EFM i korygowana w dekoderze Reeda-Solomona oraz przetwarzana na analogowe sygnały stereofoniczne

[23] Modulacja EFM jest też nazywana kodowaniem kanałowym. Informacja na płycie jest zapisana w postaci bitów kanałowych (przyp. tłum.).

odczytywane z płyty przechodzą przez odwrotną tablicę przeglądową EFM, w wyniku czego każda grupa 14 bitów jest z powrotem przekształcana na prawidłowe 8 bitów.

Oprócz zastosowania modulacji EFM, korzysta się z kodowania danych w formacie nazywanym **dwupoziomowym kodowaniem Reeda-Solomona**[24]. Obejmuje ono połączenie danych lewego i prawego kanału stereo z danymi do wykrywania i korekcji błędów. Błędy cyfrowe wykryte podczas odtwarzania są – dzięki wprowadzeniu danych redundancyjnych podczas kodowania – albo *poprawiane*, albo *ukrywane* przez interpolację między sąsiednimi próbkami lub też *wyciszane* przez ustawianie wartości próbki na zero. Taki sposób kodowania powoduje *potrojenie* bitowej przepływności danych, tj. do 4,3 Mbit/s dla danych zapisanych na płycie w stosunku do 1,4 Mbit/s dla stereofonicznego sygnału fonicznego.

Po zdekodowaniu i korekcji błędów sygnały foniczne są reprezentowane przez 16-bitowe próbki (słowa) o częstotliwości powtarzania 44,1 kHz. W najprostszym systemie te sygnały mogłyby być przetwarzane w 16-bitowym przetworniku c/a i następnie filtrowane w analogowym filtrze dolnoprzepustowym. Wymagałoby to jednak użycia filtru wysokiego rzędu mogącego przepuścić częstotliwości poniżej 20 kHz i stłumić wszystkie częstotliwości powyżej 22,05 kHz, a więc powyżej połowy częstotliwości próbkowania. Częściej spotykana metoda polega na **zwielokrotnianiu częstotliwości** (nadpróbkowaniu), czyli konwersji częstotliwości próbkowania na większą przed przetwarzaniem c/a. Stosuje się zwykle współczynnik nadpróbkowania równy 4, przechodząc z częstotliwości 44,1 kHz do 176,4 kHz. Jest to tzw. **interpolacja**, którą można objaśnić jako proces dwuetapowy (chociaż wcale nie musi być realizowana w ten sposób). Najpierw po trzy próbki o wartościach zerowych umieszcza się między próbkami oryginalnymi, zwiększając w ten sposób częstotliwość próbkowania. W dziedzinie częstotliwości daje to efekt trzykrotnego powielenia widma od 0 do 22,05 kHz, co daje częstotliwości od 22,05 do 44,1 kHz, od 44,1 do 66,15 kHz oraz od 66,15 do 88,2 kHz. W drugim etapie te nowo wprowadzone częstotliwości są usuwane skutecznym dolnoprzepustowym filtrem *cyfrowym*.

Większa częstotliwość próbkowania powoduje skrócenie okresu próbkowania, dzięki czemu przebieg z przetwornika c/a staje się gładszy. Sygnał nadal zawiera częstotliwości między 20 Hz i 20 kHz, jednak częstotliwość Nyquista została zwiększona o czynnik 4. Z tego wynika, że filtr analogowy musi teraz tylko przepuszczać częstotliwości poniżej 20 kHz a tłumić powyżej 88,2 kHz. Realizuje się to zwykle za pomocą trzybiegunowego filtru Bessela. Po co stosować filtr Bessela, skoro ucho jest nieczułe na fazę? To właśnie ta przesada, o której już mówiliśmy.

Ponieważ próbek jest cztery razy więcej, więc liczbę bitów w próbce można zmniejszyć z 16 do 15 bez pogorszenia jakości dźwięku. Korekcja $\sin(x)/x$, konieczna do skompensowania pamiętania rzędu zerowego w przetworniku c/a, może być częścią filtru analogowego lub cyfrowego.

Systemy audio mające więcej niż jeden kanał są nazywane **stereofonicznymi** (od greckiego słowa *stereos* oznaczającego *mocny*, *trwały* lub *trójwymiarowy*). Dźwięk z wielu kanałów dochodzi do słuchacza z różnych kierunków, zapewniając dokładniejsze odtwarzanie oryginalnej muzyki. Muzyka odtwarzana w systemie jednokanałowym (monofonicznym) często brzmi nienaturalnie i płasko. Natomiast przy dobrym odtwarzaniu stereofonicznym słuchacz ma wrażenie, jakby muzycy byli kilka metrów od niego. Od lat 60. w muzyce o dużej wierności stosuje się dwa kanały (lewy i prawy), a w udźwiękowieniu filmów cztery kanały (lewy, prawy, środkowy i otaczający tzw. *surround*). We wczesnych nagraniach stereofonicznych (np. Beatlesów lub Mamas and Papas) poszczególni wokaliści są często słyszalni tylko w jednym z kanałów. To spowodowało szybki rozwój bardziej zaawansowanego **miksowania w dół**, w którym dźwięk z wielu mikrofonów w studiu nagraniowym jest łączony w dwa kanały. Takie *miksowanie* to pewnego rodzaju sztuka, której celem jest wytworzenie u słuchacza wrażenia, że jest *tam*, gdzie grają muzycy.

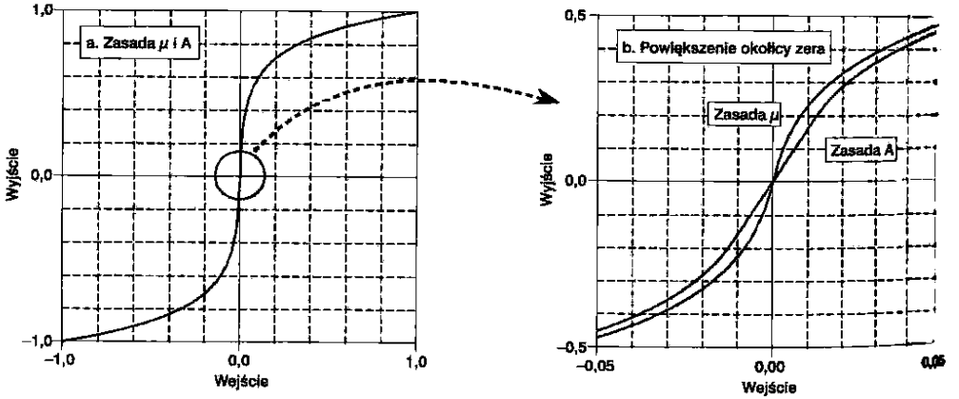
[24] Ściślej mówiąc, kodowanie to jest nazywane kodowaniem protekcyjnym z użyciem kodu CIRC (*Cross-Interleaved Reed-Solomon Code*) (przyp. tłum.).

Dźwięk czterokanałowy stosowany w filmach nosi nazwę **Dolby Stereo**, a jego wersja do mowa – **Dolby Surround Pro Logic** (*Dolby* i *Pro Logic* są znakami firmowymi Dolby Laboratories Licensing Corp.). Cztery kanały są zakodowane w standardowym lewym i prawym kanale, aby można było odtwarzać muzykę w zwykłych, dwukanałowych systemach stereo. Dekoder Dolby stosuje się podczas odtwarzania w celu zrekonstruowania dźwięku w czterech kanałach. Kanały lewy i prawy, z głośnikami umieszczonymi po każdej stronie ekranu kinowego lub telewizyjnego, są podobne jak w zwykłym, dwukanałowym systemie stereofonicznym. Głośnik kanału środkowego jest zazwyczaj umieszczany bezpośrednio nad ekranem lub pod nim. Jego zadaniem jest odtwarzanie dialogów lub innych dźwięków bezpośrednio związanych z obrazem i utrzymywanie tych dźwięków zawsze pośrodku ekranu, niezależnie od umiejscowienia widza-słuchacza. Głośniki otaczające (*surround*) są umieszczane na lewo i na prawo od słuchacza. Tych głośników w dużym pomieszczeniu może być nawet do dwudziestu. Kanał otaczający przekazuje tylko średnie częstotliwości (powiedzmy od 100 Hz do 7 kHz) i ma *opóźnienie* o 15 do 30 ms. To opóźnienie stwarza u słuchacza wrażenie, że mowa dociera z ekranu, a nie z boków. Słuchacz słyszy bowiem najpierw sygnał mowy przychodzący z przodu, a potem opóźnioną wersję sygnału mowy przychodzącą z boków. Mózg słuchacza interpretuje sygnał opóźniony jako odbicie od ścian i go ignoruje.

Kompansja

W telekomunikacji przepływność bitowa ma istotne znaczenie, bo wprost proporcjonalnie wpływa na *koszt* przesyłania sygnału. Oszczędzanie bitów jest więc równoważne z oszczędzaniem pieniędzy. **Kompansja** jest często stosowaną techniką służącą do zmniejszenia przepływności bitowej sygnałów audio przez kwantyzację z *niejednakowymi* przedziałami. Jak już wspomniano poprzednio, najgłośniejszy wytrzymywany przez ucho dźwięk (120 dB) ma amplitudę ok. milion razy większą niż najsłabszy dźwięk wykrywany (0 dB). Ucho nie może jednak rozróżnić dźwięków różniących się od siebie o mniej niż o 1 dB (12% amplitudy). Tak więc jest tylko ok. 120 różnych poziomów głośności, które mogą być wykrywane przez słuch ludzki. Poziomy te są logarytmicznie rozmieszczone w pełnym zakresie amplitudy o dynamice jeden do miliona.

Jest to stwierdzenie ważne dla digitalizacji sygnałów fonicznych. Jeśli poziomy kwantyzacji są rozmieszczone równomiernie, to trzeba zastosować 12 bitów dla uzyskania jakości odpowiadającej sygnałowi mowy w telefonii. Wystarczy jednak tylko 8 bitów, jeśli zastosuje się *niejednakowe* poziomy kwantyzacji, dopasowując je do charakterystyki słuchu ludzkiego.



Rys. 22.7. Krzywe kompansji. Krzywe zgodne z zasadami: μ255 lub A są prawie identyczne; różnią się tylko w obszarze w okolicy środka układu współrzędnych. Kompansja zwiększa amplitudę, gdy sygnał jest mały, a zmniejsza, gdy jest duży

To jest intuicyjnie zrozumiałe. Jeśli sygnał jest mały, poziomy powinny być umieszczone blisko siebie. Gdy zaś sygnał jest duży, stosuje się większe odstępy.

Kompansja może być wykonywana trzema sposobami, 1) przepuszczenie sygnału analogowego przez układ nieliniowy umieszczony przed 8-bitowym przetwornikiem a/c, 2) zastosowanie 8-bitowego przetwornika a/c o nierównomiernej charakterystyce przetwarzania, 3) zastosowanie liniowego, 12-bitowego przetwornika a/c i umieszczenie za nim cyfrowej tablicy przeglądowej (12 bitów na wejściu, 8 na wyjściu). W każdej z tych trzech opcji jest wymagana taka sama nieliniowość, tylko że realizowana w innym miejscu: w układzie analogowym, w przetworniku a/c lub w układzie cyfrowym.

Stosuje się dwa prawie identyczne standardy opisujące krzywą kompansji: **zasadę μ255** (nazywaną też **zasadą μ**) stosowaną w Ameryce Północnej oraz **zasadę A** stosowaną w Europie. W obu tych standardach korzysta się z nieliniowości o charakterze logarytmicznym, gdyż układ o takiej właściwości przetwarza charakterystykę rozmieszczenia przedziałów słyszalności ucha na charakterystykę liniową. Krzywe, według zasad μ255, jak i A, są opisane wzorami:

$$y=\frac{\ln(1+\mu x)}{\ln(1+\mu)} \quad \text{dla } 0\le x\le 1$$

Wzór 22.1. Kompansja według zasady μ255. Wzór opisuje nieliniowość zgodną z zasadą kompansji μ255. Stała μ ma wartość 255, co znajduje wyraz w nazwie standardu

$$y=\frac{1+\ln(Ax)}{1+\ln(A)} \quad \text{dla } 1/A\le x\le 1$$

$$y=\frac{Ax}{1+\ln(A)} \quad \text{dla } 0\le x\le 1/A$$

Wzór 22.2. Kompansja według zasady A. Stała A ma wartość 87,6

Na rys. 22.7 wykreślono krzywe według wzorów 22.1 i 22.2 dla zmiennej wejściowej x o wartościach między –1 i +1, dających zmienną wyjściową także o wartościach od –1 do +1. Wzory operują tylko dodatnimi wartościami wejściowymi; części krzywych dla ujemnych wartości wejściowych wyznaczono na zasadzie symetrii. Jak wynika z rys. 22.7 a), krzywe według zasad μ255 i A są prawie identyczne. Jedyna wyraźna różnica występuje w okolicy początku układu współrzędnych, co przedstawiono w powiększeniu na rys. 22.7 b). W tym obszarze zasada μ255 daje gładką krzywą, a zasada A – linię prawie prostą.

Wytworzenie stabilnej nieliniowości jest w elektronice analogowej trudnym zadaniem. Jedna z metod polega na wykorzystaniu zależności logarytmicznej między prądem i napięciem w diodzie złączowej *pn*, z zastosowaniem dodatkowego układu do korekcji dużych zmian cieplnych. W większości układów kompansji przyjęto inną strategię – aproksymowania nieliniowości przez wiele linii prostych. Typowym rozwiązaniem jest aproksymacja krzywej logarytmicznej przez 16 **segmentów** linii prostych, zwanych **krzywą odcinkowo-liniową** (*cords*). Pierwszy bit 8-bitowego słowa wyjściowego wskazuje, czy sygnał wejściowy jest dodatni, czy ujemny. Następne 3 bity określają, który z 8 segmentów dodatnich lub 8 ujemnych jest zastosowany. Cztery ostatnie bity dają podział każdego segmentu na 16 równomiernie rozmieszczonych przyrostów. Tak jak większość układów scalonych, układy służące do kompansji mają bardzo złożoną i zastrzeżoną patentami budowę wewnętrzną. Zamiast analizowania, co się dzieje wewnątrz takiego układu, skoncentrujcie uwagę raczej na rozmieszczeniu końcówek i parametrach podanych w kartach katalogowych układu.

Synteza i rozpoznawanie mowy

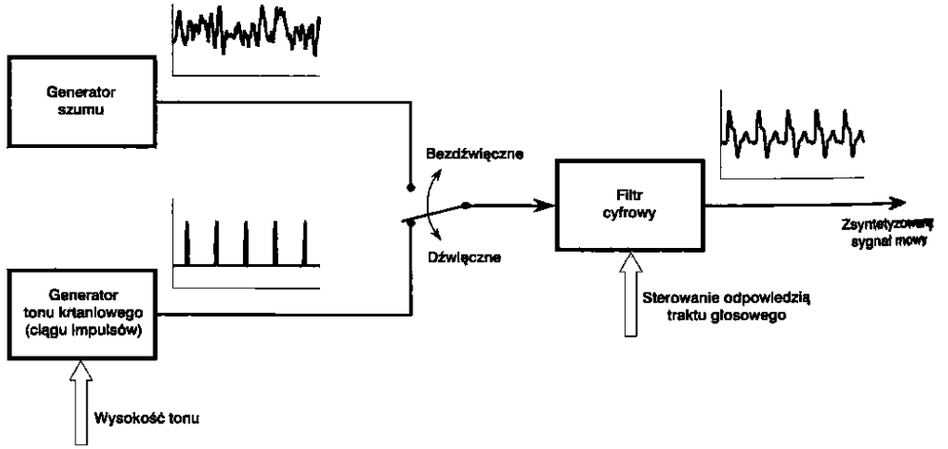
Komputerowa generacja i rozpoznawanie mowy to trudne problemy. Próbowano wielu rozwiązań, lecz tylko z umiarkowanym powodzeniem. Ta dziedzina jest obecnie przedmiotem wielu prac prowadzonych w ramach badań z użyciem DSP, które będą z pewnością kontynuowane

przez wiele najbliższych lat. Będziecie bardzo rozczarowani, jeśli oczekujecie, że w tym pod- rozdziale znajdziecie opisy budowy układów syntezy i rozpoznawania mowy. Przedstawimy tu tylko krótkie wprowadzenie do typowych rozwiązań problemu. Na początku trzeba podkre- ślić, że większość dostępnych na rynku urządzeń wytwarzających głos brzmiący jak ludzki *syntezuje* go, lecz po prostu odtwarza nagrany cyfrowo fragment głosu. Ta metoda daje bardzo dobrą jakość dźwięku, lecz jest ograniczona do wcześniej nagranych słów i zwrotów.

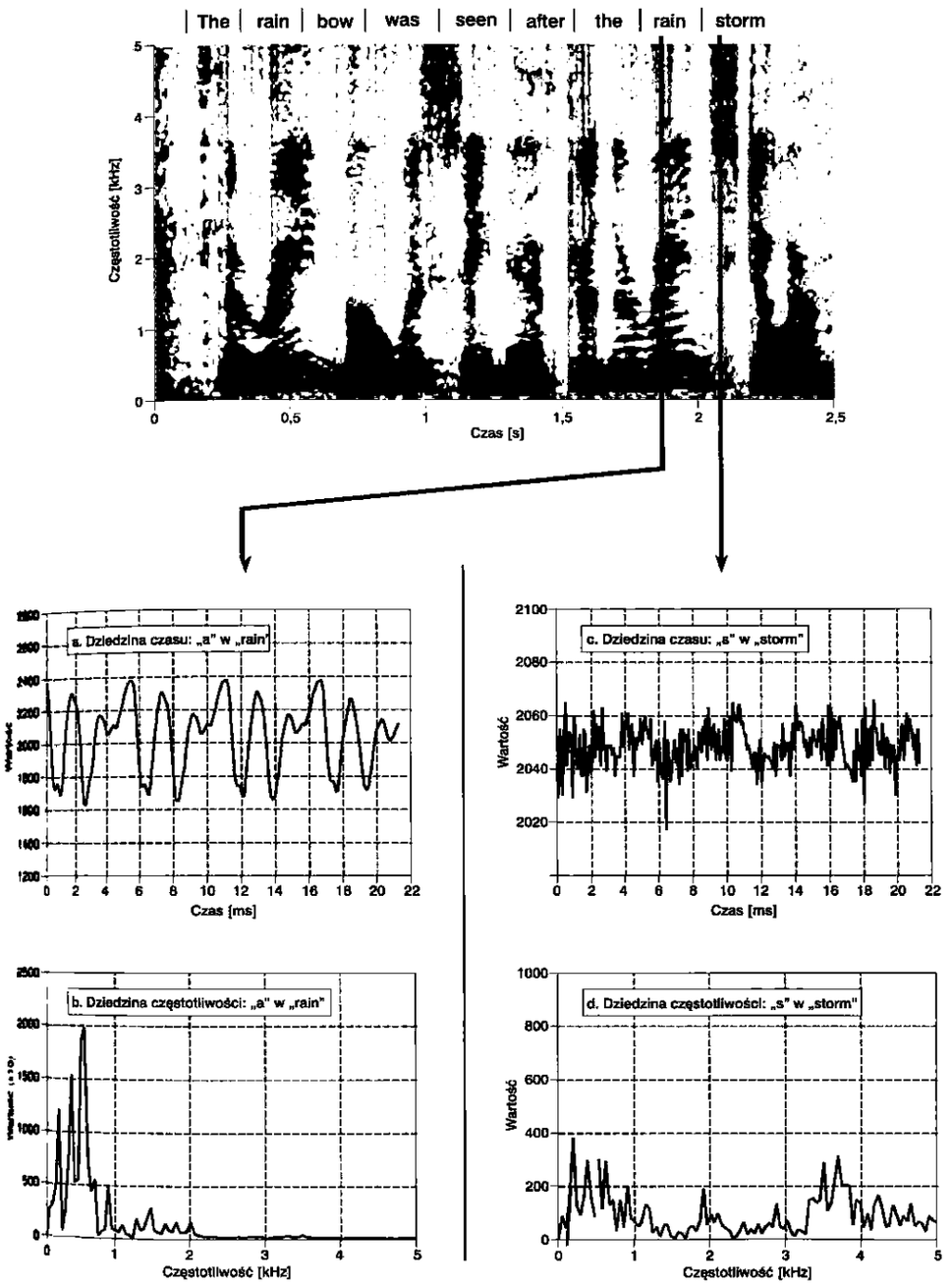
Podstawą prawie wszystkich metod syntezy i rozpoznawania mowy jest model ludzkiej mowy przedstawiony na rys. 22.8. Większość dźwięków tworzących mowę ludzką można zakwalifikować jako głoski **dźwięczne** lub **bezdźwięczne**. Głoski dźwięczne powstają, gdy powietrze jest wypychane z płuc przez struny głosowe i dalej przez usta i/lub przez nos. Struny głosowe są dwiema fałdami tkanki umieszczonymi na drodze przypływu powietrza zaraz za jabłkiem Adama. W odpowiedzi na zmieniające się napięcie mięśni struny głosowe drgają z częstotliwością od 50 do 1000 Hz, dając w rezultacie okresowe wydmuchy powietrza do gardła. Przykładem głosek dźwięcznych są samogłoski. Na rys. 22.8 głoski dźwięczne są reprezentowane przez generator ciągu impulsów z regulowaną wysokością tonu (tj. częstotliwością podstawową przebiegu, czyli częstotliwością tonu krtaniowego).

Głoski *bezdźwięczne* (np. trące) są tworzone jako szum losowy, a nie w wyniku drgań strun głosowych. Powstają one, gdy przepływ powietrza jest prawie zablokowany przez język, wargi i/lub zęby. Wywołuje to turbulencje powietrza w okolicy przewężenia. Głoski trące (*fricative*) to (w języku angielskim) m.in. *s*, *f*, *sh*, *z*, *v* oraz *th*. W modelu na rys. 22.8 głoski trące są reprezentowane przez *generator szumu*.

Oba te źródła głosek są modyfikowane przez wnęki akustyczne (komory rezonujące) tworzone przez język, wargi, jamę ustną, gardło i kanały nosowe. Ponieważ rozchodzenie się głosu w tych strukturach jest procesem liniowym, więc może być przedstawione jako liniowy filtr o odpowiednio dobranej odpowiedzi impulsowej. W większości przypadków w tym modelu jest stosowany *rekursywny* filtr cyfrowy ze współczynnikami rekursji określającymi charakterystykę filtru. Ponieważ wnęki akustyczne mają wymiary kilku centymetrów, więc odpowiedź częstotliwościowa zawiera przede wszystkim szereg częstotliwości rezonansowych w zakresie kiloherc- cowym. W terminologii stosowanej w dziedzinie przetwarzania sygnałów te maksima rezonan- sowe są nazywane **częstotliwościami formantowymi** lub **formantami**. Zmieniając wzajemne ustawienie języka i warg, można zmieniać zarówno amplitudę, jak i częstotliwości formantów.



Rys. 22.8. Model sygnału mowy. W krótkim przedziale czasu, od ok. 2 do 40 ms, sygnał mowy można modelować trzema parametrami: 1) wyborem pobudzenia okresowego lub szumowego, 2) wysokością tonu pobudzenia okresowego oraz 3) współczynnikami rekursywnego filtru liniowego modelującego trakt głosowy



Rys. 22.9. Spektrogram sygnału mowy. Jest to spektrogram zdania angielskiego: The rainbow was seen after the storm. Na rys. a), b) pokazano w dziedzinach czasu i częstotliwości sygnały dźwięcznej głoski a w słowie rain, a na rys. c), d) – bezdźwięcznej, trącej głoski s w słowie storm

Na rys. 22.9 przedstawiono często spotykany sposób przedstawiania sygnałów mowy – spek- **trogram mowy** lub tzw. **odbitkę głosu** (*voice print*). Sygnał foniczny jest dzielony na krótkie segmenty, o czasie trwania, powiedzmy, od 2 do 40 ms. Następnie korzysta się z przekształcenia FFT w celu znalezienia widma częstotliwościowego każdego segmentu. Uzyskane widma umieszcza się obok siebie i przetwarza na obraz o różnej skali szarości (mniejsze amplitudy są białe, a większe stają się coraz ciemniejsze). To umożliwia obserwację zmian zawartości częstotliwości w sygnale mowy w funkcji czasu. Długość segmentu wybrano kompromisowo z uwzględnieniem *rozdzielczości częstotliwościowej* (korzystniejszej dla dłuższych segmentów) oraz *rozdzielczości czasowej* (korzystniejszej dla krótszych segmentów).

Jak zademonstrowano na przykładzie głoski *a* w słowie *rain*, głoski dźwięczne charakteryzują się przebiegiem okresowym w dziedzinie czasu (rys. 22.9 a) oraz widmem częstotliwościowym będącym szeregiem równomiernie rozmieszczonych harmonicznych (rys. 22.9 b). Natomiast na przykładzie głoski *s* w słowie *storm* można stwierdzić, że głoski trące mają bogaty w szum sygnał w dziedzinie czasu (rys. 22.9 c) i także zaszumione widmo częstotliwościowe (rys. 22.9 d). Na widmach jest też widoczne ukształtowanie formantów w takich dźwiękach. Można również zauważyć, że obrazy słowa *rain* zarówno w dziedzinie czasu jak i częstotliwości, są bardzo podobne w obu przypadkach wymawiania tego słowa.

W krótkim odcinku czasowym, np. 25 ms, sygnał mowy można aproksymować, określając trzy parametry: 1) rodzaj pobudzenia – szumowe lub okresowe, 2) częstotliwość pobudzenia okresowego (jeśli jest stosowana), 3) współczynniki filtru cyfrowego symulującego odpowiedź traktu głosowego. Następnie można syntetyzować ciągły sygnał mowy przez stałe uaktualnianie tych trzech parametrów ok. 40 razy na sekundę. Ta metoda przyczyniła się do jednego z wczesnych komercyjnych sukcesów DSP – *Speak & Spell*, sprzedawanego w dużych ilościach urządzenia do elektronicznego wspomagania edukacyjnego dla dzieci. Jakość dźwięku w tego rodzaju syntezie mowy jest marna. Brzmienie jest raczej mechaniczne, nie zupełnie przypominające brzmienie głosu ludzkiego. Taka synteza wymaga jednak tylko bardzo małej bitowej przepływności danych, typowo kilka kbit/s.

Synteza tego rodzaju jest też podstawą **liniowego kodowania predykcyjnego** (LPC – *Linear Predictive Coding*), będącego metodą kompresji sygnału mowy. Zarejestrowany cyfrowo sygnał mowy jest dzielony na krótkie segmenty, z których każdy jest scharakteryzowany zgodnie z trzema parametrami modelu. Do tego jest konieczne kilkanaście bajtów na segment, czyli od 2 do 6 kB/s. Informacja z segmentu jest przesyłana lub rejestrowana, zgodnie z potrzebami i potem odtwarzana syntetyzatorem mowy.

Dalszym krokiem w omawianej dziedzinie są algorytmy rozpoznawania mowy służące do rozpoznawania fragmentów sygnału mowy na podstawie uzyskanych parametrów. Polega to na ogół na porównaniu segmentu informacji z szablonami dźwięków poprzednio zarejestrowanych w celu zidentyfikowania wymawianych słów. Ta metoda nie działa jednak zbyt dobrze. Jest przydatna w niektórych zastosowaniach, lecz daleka od dorównania możliwościom słuchowym człowieka. Aby zrozumieć, dlaczego rozpoznawanie mowy jest dla komputera tak trudnym zadaniem, wyobraźcie sobie, że ktoś nagle wypowiada następujące zdanie:

Larger run medical buy dogs fortunate almost when.

Nie zrozumiecie, oczywiście, znaczenia tego zdania, gdyż ono wcale go nie ma (także po angielsku – przyp. tłum.). Co ważne, przypuszczalnie nawet nie zrozumiecie poszczególnych słów, gdy są one wymawiane. Zasadą odbioru i zrozumienia mowy jest bowiem rozpoznawanie słów nie tylko na podstawie ich brzmienia, lecz także ich *kontekstu* w zdaniu i *oczekiwań* słuchacza. Wyobraźmy sobie na przykład, że słuchamy dwóch zdań:

The child wore a spider ring on Halloween.

(Dziecko nosiło naszyjnik z pająków podczas Halloween).

He was an American spy during the war.

(On był amerykańskim szpiegiem podczas wojny).

Podkreślone słowa mają w języku angielskim identyczne brzmienie, a jednak ze względu na ich kontekst *słyszymy* poprawne słowa. Korzystając z naszej zgromadzonej przez lata wiedzy o świecie, wiemy, że dzieci nie są tajnymi agentami a ludzie w czasie wojny nie noszą dziwacznej biżuterii. Na ogół nie jest to akt świadomy, lecz nieodłączna część ludzkiego słyszenia.

W większość algorytmów rozpoznawania mowy wykorzystuje się tylko dźwięk pojedynczych słów, a nie ich kontekst. Te algorytmy próbują *rozpoznawać słowa*, a nie *zrozumieć mowę*. Jest to ich poważną wadą w porównaniu ze słuchaczem-człowiekiem. W popularnych systemach rozpoznawania mowy pojawiają się trzy kłopotliwe trudności: 1) Rozpoznawana mowa musi mieć wyraźne przerwy między słowami. To uwalnia algorytm od konieczności zajmowania się zwrotami brzmiącymi podobnie, lecz złożonymi z innych słów (np. *spider ring* i *spy during*). Takie rozpoznawanie jest więc powolne i niewygodne dla ludzi przyzwyczajonych do nieprzerwanego strumienia słów. 2) Słownik bywa często ograniczony do kilkuset słów. To znaczy, że algorytm przeszukuje tylko ograniczony zestaw słów, aby znaleźć najlepiej dopasowane. W miarę rozszerzania słownika, wzrasta czas rozpoznawania i stopa błędów. 3) Algorytm musi być *przetrenowany* dla każdego mówiącego. Każda osoba stosującą system musi więc wypowiedzieć każde słowo, które ma podlegać rozpoznawaniu, a często trzeba je powtarzać pięć do dziesięciu razy. Taka spersonalizowana baza danych znacznie zwiększa dokładność rozpoznawania słów, lecz jest bardzo niewygodna i czasochłonna.

Opracowanie udanej techniki rozpoznawania mowy stanie się wielkim osiągnięciem. Mowa jest najszybszym i najskuteczniejszym sposobem komunikowania się ludzi. Rozpoznawanie mowy ma potencjalne możliwości zastąpienia pisania, wprowadzania tekstów przez klawiaturę i sterowania elektronicznego przy użyciu gałek i przełączników. Rozpoznawanie mowy musi działać jeszcze lepiej, aby zostało zaakceptowane przez rynek komercyjny. Postęp w tej dziedzinie przyjdzie przypuszczalnie z obszarów sztucznej inteligencji i sieci neuronowych, w tym samym stopniu co z samego DSP. I nie należy tego kojarzyć z *trudnościami* technicznymi, lecz z *szansą* dla techniki.

Nieliniowe przetwarzanie sygnałów fonicznych

Stosując filtrację cyfrową, można wieloma sposobami udoskonalać sygnały foniczne. Na przykład *filtrację Wienera* stosuje się do oddzielania tych częstotliwości, które należą głównie do sygnału od tych, które są głównie szumem (patrz rozdział 17). Przy użyciu *rozplotu* można kompensować niepożądany splot, np. w rekonstrukcji starych nagrań (też omówionej w rozdziale 17). Tego rodzaju metody liniowe stanowią trzon DSP. W przetwarzaniu sygnałów fonicznych przydatne też są niektóre metody *nieliniowe*. Dwie z nich będą tu krótko omówione.

Pierwsza z tych metod nieliniowych jest stosowana do tłumienia szumu szerokopasmowego w sygnale mowy. Ten typ szumu obejmuje: szum taśmy magnetofonowej, szum elektroniczny w układach analogowych, podmuchy wiatru w mikrofonach, oklaski tłumu itd. Filtracja liniowa nie na wiele tu się przyda, ponieważ częstotliwości szumu całkowicie nakładają się na częstotliwości sygnału głosowego, obie częstotliwości obejmują zakres od 200 Hz do 3,2 kHz. Jak można rozdzielić sygnały, jeśli nakładają się *zarówno* w dziedzinie czasu, jak i częstotliwości?

A oto, jak wykonuje się takie rozdzielanie. W krótkim segmencie mowy wartości składowych częstotliwościowych są w znacznym stopniu *zróżnicowane*. Na rys. 22.10 a) przedstawiono, jako przykład, widmo częstotliwościowe 16-milisekundowego segmentu mowy (tj. 128 próbek przy próbkowaniu z częstotliwością 8 kHz). Większość energii sygnału jest zawarta w kilku składowych częstotliwościowych o dużych wartościach. Na rys. 22.10 b) zaś pokazano widmo samego szumu losowego. Jest ono bardzo nieregularne, lecz ma bardziej równomierny rozkład przy małych wartościach.

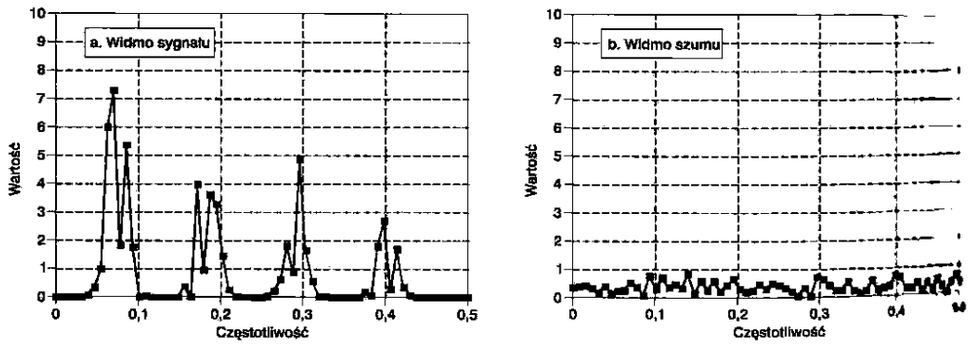
A teraz kluczowa koncepcja – jeśli występuje zarówno sygnał, jak i szum, to mogą one być częściowo rozdzielone przez obserwację *wartości* każdej składowej częstotliwościowej. Jeśli wartość składowej jest duża, to najprawdopodobniej jest to składowa sygnału, należy ją więc

pozostawić. Jeśli zaś wartość składowej jest mała, to tę składową można z największym prawdopodobieństwem przyporządkować szumowi i powinna, wobec tego, być pomniejszona, tj. wyzerowana. Składowe o średnich wartościach są dobierane w pewien sposób pośredni między przypadkami skrajnymi.

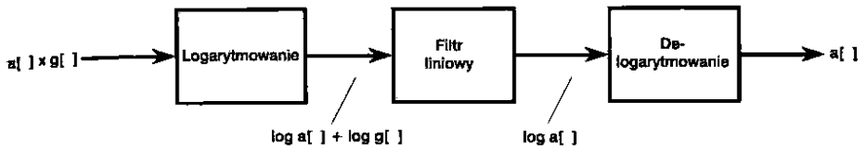
Przedstawiony powyżej sposób postępowania jest podobny do działania zmiennego w czasie *filtru Wienera*. Jak pamiętacie, z odpowiedzi częstotliwościowej filtru Wienera wynika, że przepuszczane są częstotliwości, które w większości należą do sygnału, a odrzucane te, które głównie są szumem. To wymaga *wcześniejszej* wiedzy o widmach sygnału i szumu, aby można było określić odpowiedź częstotliwościową filtru. W metodzie nieliniowej, którą omawiamy, korzysta się z tej samej zasady, z tą tylko różnicą, że odpowiedź częstotliwościową filtru Wienera jest ponownie obliczana *dla każdego segmentu*. Tak więc odpowiedź częstotliwościowa filtru zmienia się od segmentu do segmentu, co wynika z właściwości samego sygnału.

Jedną z trudności w implementacji tej (i innych) technik nieliniowych jest to, że przy filtracji długich sygnałów nie obowiązuje metoda dodawania z nakładaniem (*overlap-add*). Ponieważ odpowiedź częstotliwościowa się zmienia, więc przebieg w dziedzinie czasu w każdym z segmentów nie będzie pasował do segmentów sąsiednich. Tę trudność można pokonać, pamiętając, że informacja o sygnale fonicznym jest zakodowana w zmieniającym się w czasie widmie częstotliwościowym, a nie w kształcie przebiegu czasowego. Typowym podejściem jest podział oryginalnego sygnału w dziedzinie czasu na *nakładające się* segmenty (podobnie z zakładkami). Po przetwarzaniu każdy z nakładających się segmentów jest poddawany wygładzającemu okienkowaniu, przed ich połączeniem ze sobą. Daje to łagodne przejście wartości ma częstotliwościowego z jednego segmentu do następnego.

Druga metoda nieliniowa nosi nazwę **homomorficznego** przetwarzania sygnałów. *Homomorficzny* znaczy dosłownie: *o takiej samej strukturze*. Dodawanie nie jest jedyną możliwością łączenia szumu i zakłóceń z sygnałem użytecznym. Innymi, często stosowanymi sposobami mieszania ze sobą sygnałów mogą być mnożenie lub splot. Gdy sygnały są łączone w sposób nieliniowy (a więc w każdy inny sposób oprócz dodawania), to nie da się ich rozdzielić przez filtrację liniową. Metoda homomorficzna polega na próbie rozdzielenia sygnałów połączonych ze sobą w sposób nieliniowy w taki sposób, aby to zadanie stało się problemem liniowym. Innymi słowy, zadanie zostaje tak przekształcone, że jego struktura staje się *taka sama*, jak systemu liniowego.



Rys. 22.10. Widma sygnału mowy i szumu. Widma częstotliwościowe mowy i szumu, ogólnie biorąc, zachodzą na siebie; jakaś separacja następuje przy wyborze wystarczająco krótkich segmentów sygnału. Na rys. a) przedstawiono widmo 16-milisekundowego segmentu mowy, z którego można wnioskować, że wiele częstotliwości z tego segmentu niesie niewiele informacji o sygnale mowy. Na rys. b) przedstawiono widmo źródła szumu losowego; wszystkie składowe mają małą amplitudę. Pokazane wykresy nie są rzeczywistymi sygnałami, lecz ilustracją metody tłumienia szumu



Rys. 22.11. Homomorficzne rozdzielanie pomnożonych sygnałów. Logarytmowanie sygnału wejściowego przekształca mnożenie składowych na ich sumowanie. Składowe mogą być następnie rozdzielane przez filtrację liniową i delogarytmowanie

Weźmy na przykład sygnał audio przesyłany jako fala radiowa z modulacją AM. Przy zmianie warunków atmosferycznych poziom odbieranego sygnału wzrasta lub maleje, co powoduje powolne zmiany głośności w funkcji czasu. To zjawisko można modelować jako sygnał audio $a[\ ]$ *pomnożony* przez jakiś sygnał wolnozmienny $g[\ ]$, który reprezentuje zmieniające się wzmocnienie. Ten problem jest na ogół rozwiązywany przez zastosowanie układu *automatycznej regulacji wzmocnienia ARW* (*AGC – Automatic Gain Control*), lecz korektę można też przeprowadzić nieliniowym układem DSP.

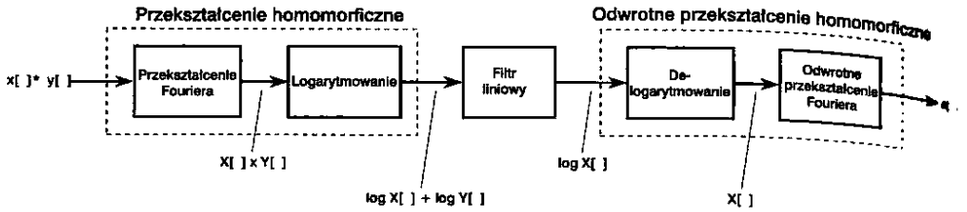
Jak pokazano na rys. 22.11, sygnał wejściowy $a[\ ] \times g[\ ]$ jest poddawany logarytmowaniu. Z zależności $\log(xy) = \log x + \log y$ wynika, że dwa sygnały można połączyć ze sobą przez sumowanie logarytmów, tj. $\log a[\ ] + \log g[\ ]$. Można więc powiedzieć, że *logarytm* jest homomorficznym przekształceniem, zamieniającym nieliniową operację *mnożenia* w liniową operację *dodawania*.

Zsumowane sygnały są następnie rozdzielane konwencjonalnym filtrem liniowym, a więc niektóre częstotliwości są przepuszczane, a inne tłumione. W układzie ARW sygnał wzmocnienia $g[\ ]$ składa się z bardzo małych częstotliwości, leżących daleko poniżej pasma głosowego obejmującego zakres od 200 Hz do 3,2 kHz. Logarytm tych sygnałów będzie miał bardziej skomplikowane widmo, lecz idea jest ta sama – stosuje się filtr górnoprzepustowy, aby wyeliminować z sygnału wolnozmienną składową wzmocnienia. W rezultacie z $\log a[\ ] + \log g[\ ]$ pozostaje tylko $\log a[\ ]$. W ostatnim kroku następuje delogarytmowanie przy użyciu funkcji wykładniczej e^x. W ten sposób uzyskuje się żądany sygnał wyjściowy $a[\ ]$.

Na rys. 22.12 przedstawiono system homomorficzny do rozdzielania sygnałów, które zostały ze sobą *splecione*. Pożytecznym zastosowaniem tego systemu okazało się usuwania echa z sygnałów fonicznych. Sygnał foniczny jest poddawany splotowi z odpowiedzią impulsową zawierającą funkcję delta oraz przesuniętą i przeskalowaną funkcję delta. Przekształcenie homomorficzne splotu jest wykonywane w dwóch etapach. Najpierw wykonuje się *przekształcenie Fouriera* zamieniające splot w mnożenie, a następnie *logarytmowanie* przekształcające mnożenie w dodawanie. Tak jak poprzednio, po rozdzieleniu sygnałów filtrem liniowym wykonuje się odwrotne przekształcenie homomorficzne.

Interesującą cechą systemu przedstawionego na rys. 22.12 jest to, że w filtracji liniowej operuje się sygnałami w dziedzinie częstotliwości w taki sam sposób, w jaki zwykle są przetwarzane sygnały w dziedzinie czasu. Innymi słowy, nastąpiła konwersja zastosowań dziedzin czasu i częstotliwości. Na przykład, gdyby splot FFT zastosować do przeprowadzenia filtracji liniowej, to mnożone „widma" byłyby w *dziedzinie czasu*. Takie odwrócenie ról narodziło specyficzny żargon. Na przykład *cepstrum* (przestawienie liter słowa *spectrum* – widmo) oznacza transformatę Fouriera logarytmu transformaty Fouriera. Podobnie, mówi się o filtrach *długo-* i *krótkoprzepustowych*, zamiast o dolno- i górnoprzepustowych. Niektórzy autorzy używają nawet określenia *Quefrency Alanysis and liftering* (przestawienie liter w nazwie *Frequency Analysis and Filtering* – analiza częstotliwościowa i filtracja – przyp. tłum.).

Miejcie na uwadze, że podane tu opisy skomplikowanych algorytmów DSP mają charakter bardzo uproszczony; rzeczywiste przetwarzanie homomorficzne zawiera wiele subtelnych szczegółów. Na przykład logarytmowanie musi operować zarówno dodatnimi, jak i ujemnymi wartościami sygnału wejściowego, gdyż taki jest charakter sygnałów fonicznych. To



Rys. 22.12. Homomorficzne rozdzielanie sygnałów splecionych. W wyniku przekształcenia Fouriera i logarytmowania składowe splecione są przetwarzane na składowe zsumowane. Po filtracji liniowej, w celu rozdzielenia składowych zsumowanych, wykonuje się działania odwrotne, czyli delogarytmowanie i odwrotne przekształcenie Fouriera

wymaga użycia *logarytmowania zespolonego*, operacji bardziej złożonej niż logarytmowanie stosowane na co dzień w nauce i technice. Gdy filtrowanie liniowe jest ograniczone do filtra *zerofazowego*, to logarytm zespolony jest wyznaczany przez zwykły logarytm wartości bezwzględnej sygnału. Po przepuszczeniu sygnału przez filtr zerofazowy znak sygnału oryginalnego jest na nowo wprowadzany do sygnału odfiltrowanego.

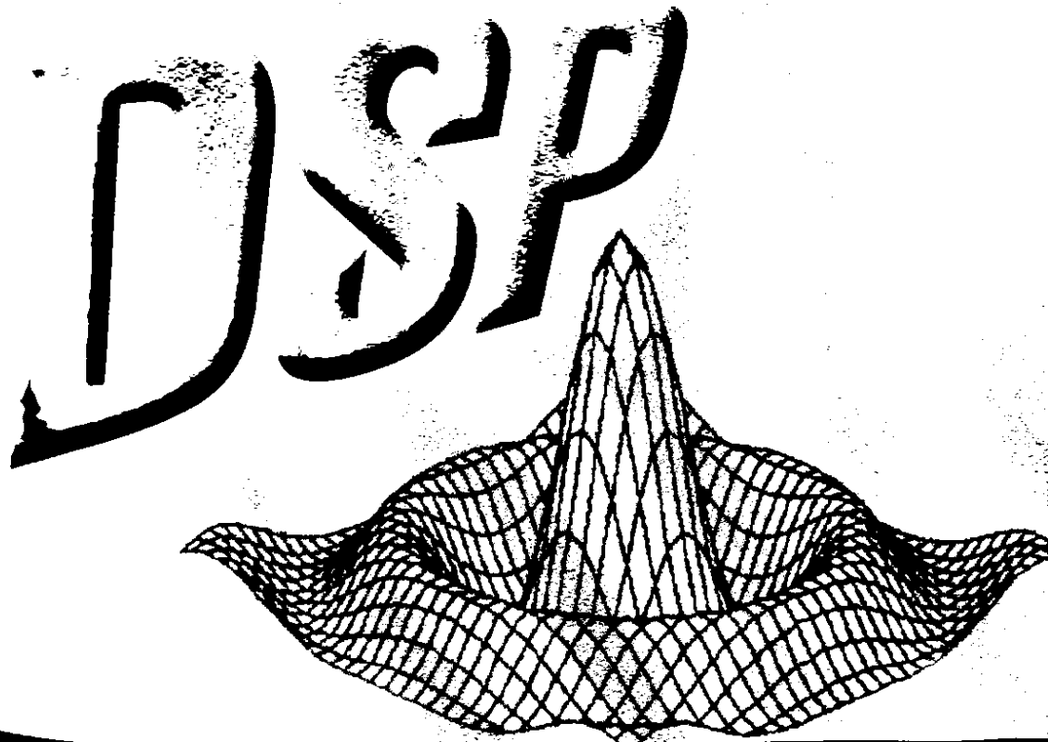
Innym zagadnieniem jest *aliasing* zachodzący podczas operacji logarytmowania. Wyobraźmy sobie na przykład digitalizację ciągłej *fali sinusoidalnej*. Zgodnie z twierdzeniem o próbkowaniu wystarczą dwie lub więcej próbek na okres sinusoidy. Rozpatrzmy teraz digitalizację logarytmu tej ciągłej fali sinusoidalnej. Ostre narożniki tego przebiegu wymagają o wiele więcej próbek na okres, aby przebieg właściwie zdyskretyzować w czasie, tzn. zapobiec aliasingowi. Wymagana częstotliwość próbkowania może być nawet 100 razy większa po logarytmowaniu niż przed nim. Co więcej, jest bez znaczenia, czy wykonuje się logarytmowanie sygnału ciągłego, czy jego cyfrowej reprezentacji – wynik jest taki sam. Aliasing następuje, gdy częstotliwość próbkowania nie jest wystarczająco duża, aby można było „uchwycić" ostre narożniki przebiegu spowodowane nieliniowością. W rezultacie sygnały foniczne mogą wymagać próbkowania z częstotliwością 100 kHz lub większą, zamiast standardowej o wartości tylko 8 kHz.

Nawet jeśli uwzględni się te wszystkie szczegóły, to nie ma gwarancji, że zlinearyzowane sygnały będą *mogły* być rozdzielone filtrem liniowym. Przyczyną jest możliwość nakładania się widm sygnałów zlinearyzowanych, nawet wtedy, gdy widma sygnałów oryginalnych się nie nakładają. Jako przykład weźmy dodawanie dwóch fal sinusoidalnych o częstotliwościach 1 kHz i 2 kHz. Te sygnały nie nakładają się na siebie w dziedzinie częstotliwości, więc mogą być całkowicie rozdzielone przez filtrację liniową. Wyobraźmy sobie teraz, że oba te przebiegi sinusoidalne zostały przemnożone przez siebie. Stosując przekształcenie homomorficzne, wyznacza się logarytm tego połączonego sygnału, uzyskując jako rezultat sumę logarytmów obu sinusoid. Problem leży w tym, że logarytm fali sinusoidalnej zawiera dużo harmonicznych. Ponieważ harmoniczne obu sygnałów nakładają się na siebie, więc całkowite rozdzielenie nie jest możliwe.

Mimo tych przeszkód, proces homomorficzny zawiera ważne pouczenie: sygnały powinny być przetwarzane w sposób *zgodny* ze sposobem tworzenia tych sygnałów. Zatem pierwszym krokiem w rozwiązywaniu zadania z dziedziny DSP powinno być zrozumienie, w jaki sposób informacja jest reprezentowana w sygnałach, które mają być przetwarzane.

23

Tworzenie i wyświetlanie obrazów



Obraz jest opisem zmian jakiegoś parametru na pewnej powierzchni. Na przykład standardowe obrazy wizyjne powstają w wyniku zmian natężenia światła na płaszczyźnie dwuwymiarowej. Światło wszakże nie jest jedynym parametrem stosowanym w obrazowaniu w nauce. Można na przykład utworzyć obraz na podstawie *temperatury* w układzie scalonym, *gęstości przepływu krwi* w arterii pacjenta, *emisji promieniowania rentgenowskiego* z odległej galaktyki, *ruchów skorupy ziemskiej* podczas trzęsienia ziemi itd. Te nietypowe obrazy są zwykle przetwarzane na konwencjonalne (tj. obrazy świetlne), aby mogły być oceniane ludzkim okiem. W tym pierwszym rozdziale dotyczącym przetwarzania obrazów opisano sposoby tworzenia obrazów i ich prezentacji obserwatorom.

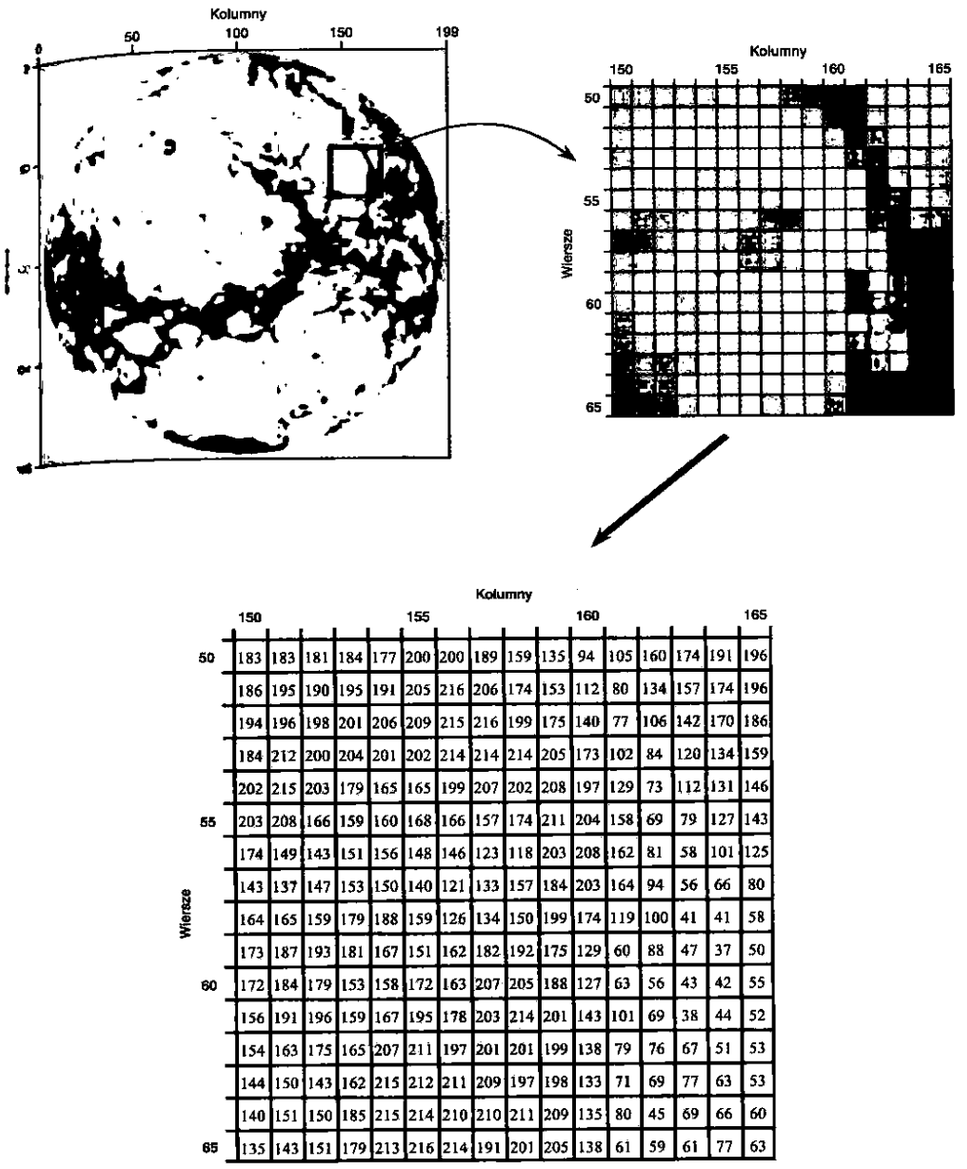
Struktura obrazu cyfrowego

Strukturę obrazu cyfrowego przedstawiono na rys. 23.1. Przykładem jest obraz planety Wenus uzyskany za pomocą mikrofalowego radaru z sondy kosmicznej umieszczonej na orbicie wokół planety. Obrazowanie mikrofalowe jest tutaj niezbędne, gdyż gęsta atmosfera Wenus pochłania światło widzialne, uniemożliwiając fotografię standardową. Pokazany obraz jest utworzony z 40 000 próbek ułożonych w dwuwymiarową matrycę zawierającą 200 kolumn i 200 wierszy. Podobnie jak w przypadku sygnałów jednowymiarowych, te kolumny i wiersze można ponumerować od 0 do 199 lub od 1 do 200. Każda próbka jest nazywana **pikselem** (od słów *picture element*). Każdy *piksel* w tym przykładzie jest pojedynczą liczbą od 0 do 255. Podczas pobierania obrazu, każdej z tych liczb przyporządkowano ilość energii mikrofalowej odbitej od odpowiadającego temu pikselowi położenia na powierzchni planety. Aby te informacje przedstawić w postaci obrazu, wyraża się wartość każdego piksela w **skali szarości** (*grayscale*), gdzie 0 oznacza czerń, 255 biel, a wartości pośrednie są różnymi odcieniami szarości.

Informacja o obrazie jest zakodowana w **dziedzinie przestrzeni**, obrazowym równoważniku dziedziny czasu. Inaczej mówiąc, cechy obrazu są reprezentowane przez *zbocza sygnału*, a nie przez *sinusoidy*. Wynika z tego, że rozstaw i liczba pikseli zależą raczej od wielkości szczegółów obrazu, które powinny być widoczne, niż od formalnych ograniczeń twierdzenia o próbkowania. Aliasing *może* wprawdzie występować w obrazach, lecz jest na ogół traktowany jako pewna niedogodność, a nie poważny problem. Na przykład garnitury w prążki fatalnie wyglądają w telewizji, gdyż powtarzający się wzór ma częstotliwość większą od częstotliwości Nyquista. Częstotliwości aliasingowe pojawiają się jako jasne i ciemne pasy przesuwające się po garniturze, gdy osoba go nosząca zmienia pozycję.

„Typowy" obraz cyfrowy składa się z ok. 500 wierszy i 500 kolumn. Taka jakość obrazu jest spotykana w telewizji, komputerach osobistych i w typowych badaniach naukowych. Obrazy o mniejszej liczbie pikseli, powiedzmy 250 na 250, są uważane za obrazy o niezwykle słabej rozdzielczości. W miarę rozwoju techniki i nowych zastosowań techniki obrazowej dodaje się więcej pikseli. Te obrazy o małej rozdzielczości mają zauważalnie nienaturalny wygląd, często są widoczne poszczególne piksele. Z drugiej strony, obrazy o liczbie pikseli większej niż 1000 na 1000 są uważane za wyjątkowo dobre. Odpowiada to jakości najlepszej grafiki komputerowej, telewizji wysokiej jakości (*high definition TV*) i filmów 35-milimetrowych. Są też aplikacje wymagające jeszcze większej rozdzielczości, np. obrazu o każdym z boków równym kilku tysiącom pikseli. Są to zdigitalizowane zdjęcia rentgenowskie, fotografie z kosmosu i lśniące reklamy w magazynach ilustrowanych.

Najsilniejszą motywacją do stosowania obrazów o niższej rozdzielczości jest *mniejsza liczba* pikseli, którymi trzeba operować. Nie jest to sprawa trywialna; jednym z najtrudniejszych problemów w przetwarzaniu obrazów jest posługiwanie się ogromną ilością danych. Jedna sekunda cyfrowego sygnału audio wymaga, na przykład, ok. 8 *kilobajtów*, podczas gdy jedna sekunda sygnału telewizyjnego ok. 8 *megabajtów*. Przesyłanie obrazu 500 × 500 pikseli modemem o przepływności bitowej 33,6 kbit/s wymaga prawie minuty! Przejście do obrazu o wymiarze 1000 × 1000 powoduje zwiększenie tych problemów o *czynnik cztery*.



Rys. 23.1. Struktura obrazu cyfrowego. Przykładem jest obraz planety Wenus uzyskany z odbitych mikrofal. Obraz cyfrowy jest reprezentowany przez dwuwymiarową matrycę liczb, zwanych pikselami. W tym obrazie matryca ma 200 wierszy i 200 kolumn, a piksele mają wartości od 0 do 255. Przy pobieraniu tego obrazu wartość każdego piksela odpowiadała poziomowi odbitej energii mikrofalowej. Obraz jest wyrażany w skali szarości przez przyporządkowanie liczb od 0 do 255 do zmieniających się odcieni szarości

W przetwarzaniu obrazów stosuje się na ogół 256 **poziomów szarości** (poziomów kwantyza-cji), co jest równoważne jednemu bajtowi na piksel. Jest kilka przyczyn takiego wybo... Po pierwsze, jeden bajt jest wygodny do operowania danymi, gdyż komputery na ogół w... w bajtach gromadzą dane. Po drugie, duża liczba pikseli w obrazie kompensuje w pewnym stopniu ograniczoną liczbę poziomów kwantyzacji. Rozpatrzmy na przykład grupę sąsiadu-jących pikseli o cyfrowych wartościach zmieniających się między 145 i 146. Oko ludzkie odbiera jaskrawość tego obszaru jako równą 145,5. Inaczej mówiąc, obrazy *drżą*. Po trzecie i najważniejsze, wielkość skoku jaskrawości równa 1/256 (0,39%) jest mniejsza od rozdziel-czości oka. Obraz prezentowany widzowi nie polepszy się więc przy użyciu liczby poziomów kwantyzacji większej niż 256.

Jednak niektóre obrazy powinny być rejestrowane z kwantyzacją powyżej 8 bitów na piksel. Pamiętajmy, że większość obrazów, jakie można spotykać w DSP, wyraża parametry nie-wizualne. Wtedy większą liczbę poziomów kwantyzacji można wykorzystać w obrazie do uchwycenia drobnych szczegółów sygnału. Nie należy się jednak spodziewać, żeby ludzkie oko zobaczyło całą informację zawartą w tych tak gęsto rozmieszczonych poziomach. Do kwestii związanych z tym faktem wrócimy przy omawianiu jaskrawości i kontrastu.

Wartość każdego piksela w obrazie cyfrowym reprezentuje mały *obszar* w obrazie ciągłym przetwarzanym na cyfrowy. Załóżmy dla przykładu, że sonda wenusjańska, krążąc nad pla-netą, pobiera próbki co 10 metrów wzdłuż jej powierzchni. To wyznacza kwadratową prze-strzeń **próbkowania** oraz **siatkę próbkowania**, przy czym każdy piksel reprezentuje obszar 10 m × 10 m. Pomyślmy teraz, co się dzieje przy pojedynczym pomiarze odbitych mikrofal. Sonda emituje silnie zogniskowaną wiązkę energii mikrofalowej, uderzającą w powierzchnię planety na kołowym obszarze, o średnicy na przykład 15 metrów. Każdy piksel zawiera więc informację o tym obszarze kołowym, niezależnie od wymiarów siatki próbkowania.

Obszar obrazu ciągłego mający swój udział w wartości pikselu jest nazywany **aperturą próbkowania**. Rozmiar apertury próbkowania jest często związany z charakterystycznymi właściwościami konkretnych stosowanych systemów obrazujących. Ograniczeniem, na przykład mikroskopów, są układy optyczne i długość fali świetlnej, elektronicznych aparatów fotograficznych i kamer zaś – losowe rozpraszanie elektronów w czujniku ob-razowym itd. W większości przypadków realizuje się siatkę próbkowania w przybliżeniu taką samą, jak apertura próbkowania w systemie. Rozdzielczość końcowego obrazu cyfro-wego ogranicza większa z tych dwóch wartości – siatka próbkowania lub jego apertura. Powrócimy do tego tematu w rozdziale 25 przy omawianiu rozdzielczości przestrzennej obrazów cyfrowych.

Do obrazów cyfrowych dodaje się barwę przez przypisanie każdemu pikselowi trzech liczb, wyrażających natężenie światła trzech barw podstawowych: czerwonej, zielonej i niebieskiej. Mieszanie tych trzech barw daje wszystkie możliwe barwy, które mogą być odbierane przez oko ludzkie. Do zapisu natężenia każdej z barw podstawowych stosuje się na ogół jeden bajt, dzięki czemu w obrazie może występować 256 × 256 × 256 = 16,8 mln różnych barw.

Barwa jest bardzo ważna, gdy celem jest przedstawienie widzowi prawdziwego obrazu świa-ta, np. w telewizji i w fotografii. Na ogół nie o to chodzi w nauce i technice. W tych dzie-dzinach celem jest analiza sygnału dwywymiarowego z wykorzystaniem – jako *narzędzia* – ludzkiego systemu widzenia. Do tego zupełnie wystarczające są obrazy czarno-białe.

Aparaty fotograficzne i oczy

Oko i elektroniczny aparat fotograficzny omawiamy wspólnie, gdyż ich budowa i działanie są bardzo podobne. Oba te systemy optyczne zawierają zespół soczewek oraz czujnik obra-zowy. Zespół soczewek wychwytuje część światła emitowanego przez obiekt i ogniskuje je na czujniku obrazowym. Następnie czujnik obrazowy przekształca obraz świetlny na sygnał wideo – elektroniczny lub neuronowy.

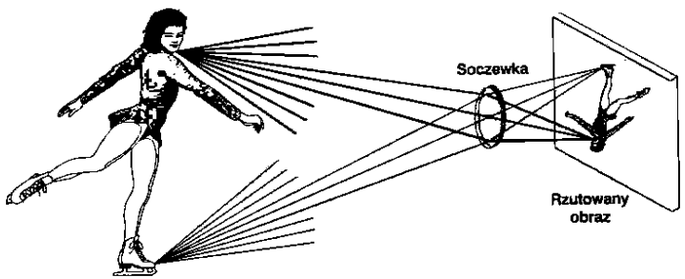
Działanie soczewki przedstawiono na rys. 23.2. W podanym przykładzie obraz łyżwiarki jest zogniskowany na ekranie. Pojęcie *zogniskowanie* oznacza, że powstaje dopasowanie je-den do jednego każdego punktu łyżwiarki z odpowiadającym mu punktem na ekranie. Jako przykład weźmy obszar 1 mm × 1 mm na czubku stopy łyżwiarki. W jasnym świetle na ten obszar o powierzchni 1 mm^2 pada 100 bilionów fotonów światła w ciągu sekundy. Zależnie od właściwości powierzchni, od 1 do 99% padających fotonów odbija się losowo w róż-nych kierunkach. Tylko niewielka część tych odbitych fotonów przejdzie przez soczewkę. Na przykład tylko ok. milionowa część odbitego światła przechodzi przez soczewkę o średnicy 1 cm umieszczoną w odległości 3 m od obiektu.

Załamanie światła w soczewce zmienia kierunek rozchodzenia się pojedynczych fotonów, zależnie od położenia punktu, w którym uderzają w granicę szkło/powietrze oraz od kąta padania. Te zmiany kierunków powodują, że światło rozchodzące się z jednego punktu po-wraca do jednego punktu na ekranie projekcyjnym. Wszystkie fotony, które odbijają się od czubka stopy *oraz* przechodzą przez soczewkę, są z powrotem gromadzone na „czubku stopy" rzutowanym na obrazie. Podobnie porcje światła pochodzące ze *wszystkich* punktów obiektu przechodzą przez soczewkę i są ogniskowane w odpowiadających im punktach na rzutowanym obrazie.

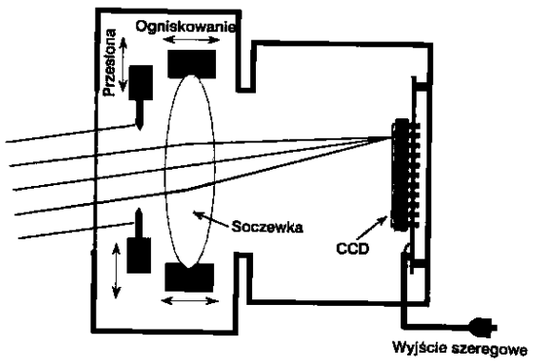
Na rys. 23.3 i 23.4 przedstawiono schematycznie budowę elektronicznego aparatu fotogra-ficznego i ludzkiego oka. Oba te przyrządy optyczne są światłoszczelnymi komorami z so-czewką umieszczoną z jednej strony i czujnikiem światła z drugiej. W aparacie znajduje się powietrze, a oko jest wypełnione przezroczystym płynem. Systemy soczewek mają w obu przypadkach dwa parametry regulowane: **ogniskową** i **średnicę źrenicy** (w aparacie fotogra-ficznym – **średnicę przesłony**).

Jeśli soczewka nie jest prawidłowo zogniskowana, to każdy punkt obiektu będzie rzutowany na czujnik obrazowy jako kolisty obszar, powodując rozmycie obrazu. W aparacie ogniskowanie wykonuje się, przesuwając soczewkę bliżej lub dalej od czujnika obrazowego. Oko zaś zawiera dwie soczewki – wypukłą rogówkę w przedniej części gałki ocznej i regulowaną soczewkę wewnątrz oka. Załamanie światła następuje głównie w rogówce, która ma jednak ściśle ustalony kształt i położenie. Regulacja ogniskowania następuje więc w soczewce we-wnętrznej – elastycznej strukturze mogącej zmieniać kształt pod wpływem *mięśni akomo-dacyjnych*. Mięśnie, kurcząc się, powodują spłaszczanie soczewki w taki sposób, aby obiekt dawał ostry obraz w ognisku.

W obu systemach optycznych istnieje regulacja ilości światła padającego na soczewkę, a więc jaskrawości obrazu rzutowanego na czujnik obrazowy. W oku do tego celu służy *źrenica* będącą okrągłym otworem pośrodku tęczówki, a w aparacie fotograficznym ilość światła reguluje się *przesłoną*. Tęczówka oka jest utworzona z nieprzezroczystej tkanki mięśniowej, która może się kurczyć, rozszerzając *źrenicę*. W aparacie funkcję źrenicy pełni odpowiedni mechanizm zwany przesłoną.

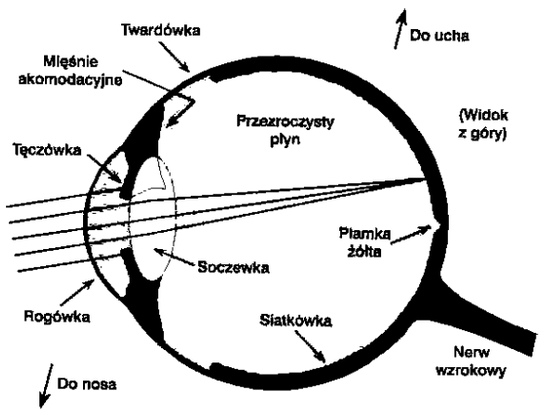


Rys. 23.2. Ogniskowanie za pomocą soczewki. Soczewka zbiera światło rozchodzące się ze źródła punktowego i skupia je w jakimś innym punkcie. Dzięki temu obraz może być przez soczewkę rzutowany na ekran



Rys. 23.3. Budowa elektronicznego aparatu fotograficznego. Ogniskowanie osiąga się przez przesuwanie soczewki w kierunku do i od czujnika obrazowego. Ilość światła dochodzącego do czujnika jest regulowana przesłoną – mechanicznym urządzeniem zmieniającym efektywną średnicę soczewki. Obecnie w aparatach i kamerach najczęściej stosowanym czujnikiem obrazowym jest CCD, dwuwymiarowa matryca elementów światłoczułych

Wzajemne oddziaływanie parametrów systemów optycznych może następować na wiele czasem niespodziewanych sposobów. Zastanówmy się, na przykład, jak ilość światła i cz ilość czujnika światła wpływają na *ostrość* uzyskanego obrazu. *Średnica źrenicy* (lub prze słony) i *czas ekspozycji* są tak dobierane, żeby do czujnika światła trafiała właściwa ilo światła z oglądanego widoku. Jeśli dochodzi zbyt dużo światła, to średnica źrenicy musi zostać zmniejszona, powodując zwiększenie *głębi ostrości* (zakresu odległości od aparatu przy której obiekt pozostaje w ognisku). Większa głębia ostrości zapewnia ostrzejszy obraz w sytuacji, gdy obiekty znajdują się w różnych odległościach od aparatu. Ponadto, obniżo światła umożliwia skrócenie czasu ekspozycji, co zmniejsza rozmycie obrazu spowodowane



Rys. 23.4. Budowa ludzkiego oka. Oko ma kształt kuli o średnicy ok. 3 cm, wypełnionej płynem i osłoniętej twardą powłoką zewnętrzną zwaną twardówką. Ogniskowanie światła następuje głównie w wypukłej rogówce tworzącej stałą soczewkę z przodu oka. Regulowane ogniskowanie następuje w wyniku działania mięśni akomodacyjnych na elastyczną soczewkę wewnątrz oka. Ilość światła wchodzącego do oka jest regulowana przez źrenicę będącą okrągłym otworem umieszczonym pośrodku tęczówki utworzonej z nieprzezroczystej tkanki mięśniowej. Tylna półkula oka zawiera siatkówkę, która jest warstwą światłoczułych komórek nerwowych przetwarzających obraz na sygnał neuronowy w nerwie wzrokowym

drganiami aparatu i ruchami obiektu. W systemach optycznych jest wiele takich parametrów wymagających kompromisowego doboru.

Regulacja średnicy źrenicy (w oku) lub przesłony (w aparacie) jest konieczna, gdyż zmiany natężenia światła w otoczeniu są znacznie większe niż te, które mogą być bezpośrednio przetwarzane w czujniku światła. Na przykład stosunek natężeń światła słonecznego i księżycowego jest jak ok. jeden do miliona. Uwzględniając jeszcze współczynnik odbicia światła, który może zmieniać się od 1 do 99%, uzyskujemy zakres zmian natężenia światła prawie *jeden do stu milionów*.

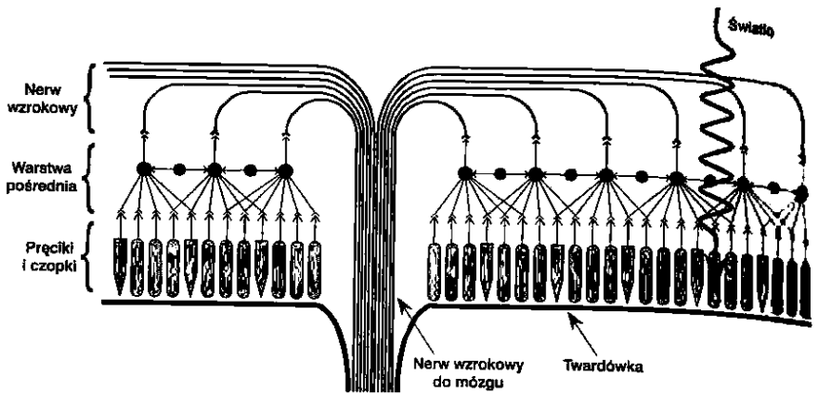
Zakres dynamiczny elektronicznego aparatu fotograficznego wynosi typowo od 300 do 1000. Jest on definiowany jako największy sygnał, jaki można zmierzyć, podzielony przez szum własny urządzenia. Patrząc na to w inny sposób, możemy powiedzieć, że maksymalny wytwarzany sygnał jest 1 V, a wartość skuteczna szumu ciemnego wynosi ok. 1 mV. Typowa soczewka aparatu ma przesłonę zmieniającą powierzchnię otworu świetlnego o czynnik ok. 300. Z tego wynika zakres dynamiczny równy kilkaset tysięcy. Jest jasne, że ciemną nocą będzie całkiem bezużyteczny aparat fotograficzny z zespołem soczewek takim samym, jak stosowany w jasnym świetle słonecznym.

Oko zaś działa w zakresie dynamicznym, który niemal pokrywa duży zakres zmian natężenia światła w otoczeniu. To zaskakujące, że wcale nie zmiany źrenicy są głównym sposobem uzyskania tego ogromnego zakresu dynamicznego. Przy przejściu od ciemności do jasnego światła powierzchnia źrenicy zmienia się tylko o czynnik ok. 20. Pozostała część zakresu dynamicznego jest uzyskiwana w wyniku stopniowej adaptacji czułości komórek nerwowych wykrywających światło. Na przykład trzeba kilku minut, aby nasze oczy dostosowały się do słabego nawet światła po wyjściu z ciemnego kina.

Jednym ze sposobów, w jaki DSP może ulepszać obrazy, jest redukowanie zakresu dynamicznego, jaki musi być odbierany przez obserwatora. Nie chcemy bowiem mieć bardzo jasnych i bardzo ciemnych obszarów w tym samym obrazie. Widziany obraz tworzy się z dwóch sygnałów obrazowych: dwuwymiarowego obrazu pochodzącego z *bezpośredniego oświetlenia* oglądanego widoku i dwuwymiarowego obrazu *odbitego*. Obraz odbity ma zakres dynamiczny mniejszy od 100, ponieważ zwykle materiały odbijają od 1% do 99% padającego światła. W świetle odbitym jest zawarta większość *informacji o obrazie*, a więc informacji o położeniu obiektu i o właściwościach jego powierzchni. Natomiast sygnał od oświetlenia bezpośredniego zależy od źródeł światła umiejscowionych wokół obiektów, ale nie od samych obiektów. Sygnał od oświetlenia bezpośredniego może mieć zakres dynamiczny nawet rzędu milionów, ale typowa wartość dla pojedynczego obrazu to 10 do 100. Ten sygnał niesie niewiele interesujących informacji, lecz może pogorszyć jakość końcowego obrazu przez nadmierne zwiększenie jego zakresu dynamicznego. DSP może poprawić tę sytuację, wytłumiając sygnał oświetlenia bezpośredniego w taki sposób, żeby w obrazie dominował sygnał odbity. W następnym rozdziale przedstawimy sposób implementacji takiego algorytmu.

Tylną powierzchnię gałki ocznej pokrywa światłoczuła **siatkówka**. Siatkówka może być podzielona na trzy główne warstwy wyspecjalizowanych komórek nerwowych (rys. 23.5). Jedna z warstw to komórki przetwarzające światło na sygnały neuronowe, druga warstwa służy do przetwarzania obrazów, a trzecia – do przekazywania informacji do nerwu wzrokowego prowadzącego do mózgu. U prawie wszystkich zwierząt te warstwy są ułożone *odwrotnie*. To znaczy, że komórki światłoczułe znajdują się w ostatniej warstwie, a światło, przed detekcją, musi przejść przez inne warstwy.

Są dwa typy komórek wykrywających światło: **pręciki** oraz **czopki**, których nazwy pochodzą od wyglądu pod mikroskopem. Pręciki są wyspecjalizowane do odbioru światła bardzo słabego, na przykład światła z nocnego nieba. Widzenie prawie w zupełnej ciemności jest bardzo *zaszumione*; obrazy wydają się być pokryte ciągle zmieniającym się ziarnistym wzorem. Przyczyną nie są ograniczenia oka, lecz bardzo słaby sygnał obrazu. Tak mało energii świetlnej wchodzi do oka, że detekcja indywidualnych fotonów następuje w sposób losowy.



Rys. 23.5. Siatkówka w oku ludzkim. Siatkówka zawiera trzy główne warstwy: 1) receptory światła – pręciki i czopki, 2) warstwę pośrednią służącą do redukcji danych i przetwarzania obrazu oraz 3) włókna nerwu wzrokowego prowadzącego do mózgu. Struktura tych warstw jest odwrócona – światło musi przejść przez inne warstwy, zanim dotrze do receptorów

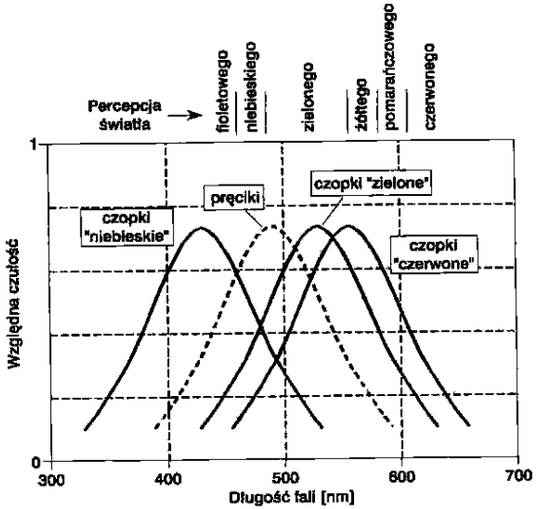
Jest to *szum statystyczny*, występujący we wszystkich obrazach o małej ilości światła, np. w noktowizorach wojskowych. Wrócimy do tego tematu w rozdziale 25. Ponieważ pręciki nie wykrywają barw, więc widzenie przy słabym oświetleniu jest czarno-białe.

Receptory czopkowe są wyspecjalizowane w rozróżnianiu barw, lecz działają tylko przy wystarczająco dobrym oświetleniu. W oku są trzy rodzaje czopków – tzw. „czerwone", „zielone" lub „niebieskie", a więc najczulsze na te właśnie barwy. Czopki każdego z tych rodzajów zawierają inny fotopigment, czyli związek chemiczny absorbujący inną długość fali świetlnej. Na rys. 23.6 przedstawiono zakresy długości fal świetlnych, na jakie reagują receptory tych trzech rodzajów. Jest to tzw. **kodowanie RGB** określające, w jaki sposób informacja o barwie opuszcza oko przez nerw wzrokowy. Ludzka percepcja barw komplikuje się w wyniku przetwarzania sygnałów wzrokowych na dolnych poziomach mózgu. Kodowanie RGB przechodzi w inną formę kodowania, w którym barwy są klasyfikowane jako: czerwona *lub* zielona, niebieska *lub* żółta i jasna *lub* ciemna.

Kodowanie RGB jest ważnym ograniczeniem ludzkiego widzenia; długości fali świetlnych występujące w otoczeniu są zaliczane tylko do trzech szerokich kategorii. Natomiast specjalizowane aparaty fotograficzne są w stanie dzielić widmo światła na setki lub tysiące indywidualnych barw. Takie rozróżnienie barw może służyć np. do klasyfikacji komórek na zdrowe lub rakowe, do zrozumienia zjawisk fizycznych na odległej gwieździe lub do wykrywania maskujących się żołnierzy ukrytych w lesie. Dlaczego oko jest tak ograniczone pod względem wykrywania barw? Widocznie wszystkie ludzkie istoty do przetrwania potrzebują znaleźć *czerwone* jabłko wśród *zielonych* liści rysujące się na tle *niebieskiego* nieba.

Czopki i pręciki mają szerokość ok. 3 μm i są gęsto ścieśnione na całej powierzchni siatkówki 3 × 3 cm. Siatkówka jest więc złożona z matrycy ok. 10 000 × 10 000 = 100 milionów receptorów. Nerw wzrokowy zaś ma tylko ok. miliona włókien nerwowych, które mogą być dołączone do tych komórek receptorowych. Każde włókno nerwu wzrokowego jest dołączone – przez warstwę pośrednią – do średnio ok. 100 receptorów światła. Ponadto, w celu lepszego skonsolidowania informacji, warstwa pośrednia wzbogaca obraz przez zaostrzanie krawędzi i stłumienie składowej, która w oglądanym widoku pochodzi od oświetlenia bezpośredniego. To biologiczne przetwarzanie obrazu będzie omówione w następnym rozdziale.

Dokładnie pośrodku siatkówki znajduje się tzw. **plamka żółta** służąca do widzenia o bardzo dużej rozdzielczości (patrz rys. 23.4). Plamka żółta pod kilkoma względami różni się od reszty

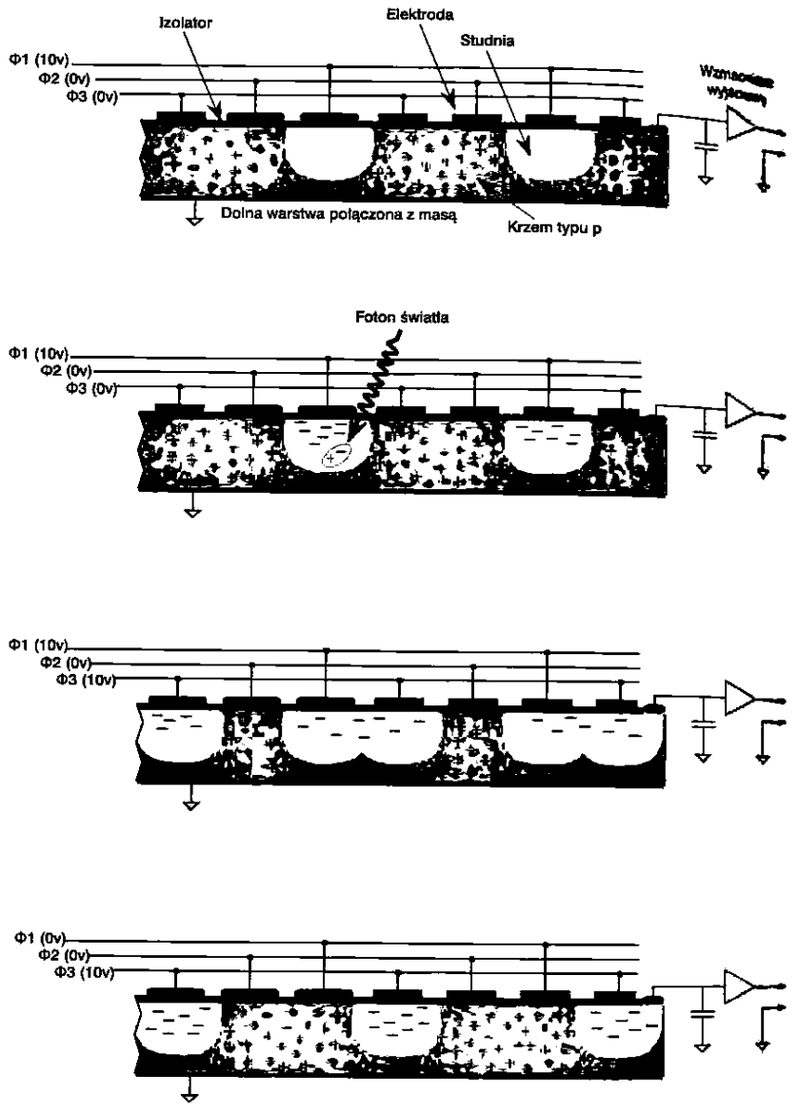


Rys. 23.6. Charakterystyka widmowa oka. W ludzkim oku czopki trzech rodzajów są czułe na częstotliwości z trzech obszarów widma światła, odpowiadających w przybliżeniu barwom czerwonej, zielonej i niebieskiej. Kombinacje tych trzech barw tworzą wszystkie barwy, które może odbierać ludzki wzrok. Czopki nie są wystarczająco czułe, aby mogły służyć do widzenia przy słabym oświetleniu. W tych warunkach do detekcji obrazu służą pręciki. Właśnie dlatego barwy są tak słabo widoczne w nocy

siatkówki. Po pierwsze, do bocznej części plamki żółtej przylega nerw wzrokowy i warstwy pośredniczące, dzięki czemu receptory są bardziej bezpośrednio wystawione na działanie padającego światła. To powoduje, że plamka żółta ma wygląd niewielkiego wgłębienia w siatkówce. Po drugie, w plamce żółtej znajdują się tylko czopki, upakowane znacznie gęściej niż w pozostałej części siatkówki. Brak pręcików w plamce żółtej wyjaśnia, dlaczego widzenie nocne jest często lepsze, gdy na obiekt patrzy się z *boku*, zamiast na wprost. Po trzecie, na każde włókno nerwu wzrokowego oddziałuje tylko kilka czopków, co daje dobrą zdolność lokalizacji. Plamka żółta jest zadziwiająco mała. Przy zwykłej odległości, jak przy czytaniu, plamka żółta widzi tylko obszar o średnicy 1 mm, mniejszy od rozmiaru pojedynczej litery! W tym obszarze rozdzielczość jest równoważna siatce ok. 20 × 20 pikseli.

Wzrok ludzki przezwycięża problem małych wymiarów plamki żółtej przez nagłe ruchy gałki ocznej. Te gwałtowne ruchy powodują, że dająca dużą rozdzielczość plamka żółta szybko przebiega pole widzenia, poszukując właściwych informacji. Ponadto, w wyniku tych nagłych ruchów oka pręciki i czopki otrzymują ciągle zmieniający się obraz świetlny. Jest to ważne z powodu naturalnej zdolności siatkówki do przystosowywania się do zmian natężenia światła. Rzeczywiście, jeśli zmusimy oko do zatrzymania się przez jakiś czas na tym samym widoku, to szczegóły i barwy obrazu zaczną blaknąć po kilku sekundach.

Czujnikiem obrazu najczęściej stosowanym w elektronicznych aparatach fotograficznych i kamerach jest układ ze sprzężeniem ładunkowym CCD (*Charge Coupled Device*). Jest to układ scalony, który w latach 80. zastąpił większość lamp próżniowych stosowanych w kamerach jako czujniki; podobnie jak dwadzieścia lat wcześniej tranzystory zastąpiły wzmacniacze lampowe. Sercem układu CCD jest cienka płytka krzemowa o powierzchni zwykle ok.1 cm^2. Jak pokazano na przekroju na rys. 23.7, dolna powierzchnia jest powleczona cienką warstwą metalu połączoną z masą. Powierzchnia górna jest pokryta cienką warstwą izolatora elektrycznego. Na niej umieszczono elektrody, według powtarzającego się wzoru. Najpopularniejszym rodzajem układów CCD są układy **z odczytem trzyfazowym**, w których



Rys. 23.7. Działanie układu ze sprzężeniem ładunkowym (CCD). Jak przedstawiono na tych przekrojach, cienka płytka z krzemu typu p jest pokryta warstwą izolatora i matrycą elektrod. Elektrody są połączone ze sobą w grupach po trzy. Mogą więc być do nich dołączane trzy niezależne napięcia Φ1, Φ2 oraz Φ3. Gdy do elektrody jest dołączone napięcie dodatnie, to dziury (czyli nośniki ładunku dodatniego oznaczone „+") są odpychane. W rezultacie powstaje obszar zubożony w dziury, zwany studnią. Padające światło generuje dziury i elektrony, przy czym zbierane elektrony (oznaczone „–") zostają uwięzione w poszczególnych studniach. Przez odpowiednie operowanie napięciami na trzech elektrodach uzyskuje się przesuwanie elektronów z każdej studni aż do końca płytki krzemowej, gdzie znajduje się wzmacniacz ładunkowy przetwarzający ładunek na napięcie

co trzecie elektrody są ze sobą połączone. Stosuje się krzem *typu p*, a więc krzem z nadmiarem nośników dodatnich zwanych *dziurami*. W tych rozważaniach przyjmiemy, że dziura jest dodatnio naładowaną cząstką, mogącą swobodnie poruszać się w krzemie. Na rysunku dziury oznaczono symbolem „+".

Na rys. 23.7 a) do jednej z trzech faz dołączono napięcie +10 V. Pozostałe dwie fazy są utrzymywane na potencjale 0 V. To powoduje, że dziury oddalają się od każdej co trzeciej elektrody, gdyż ładunki dodatnie są odpychane przez dodatnie elektrody. Pod tymi elektrodami powstają więc obszary zwane **studniami**, co jest skróconą wersją pojęcia fizycznego *studnia potencjału*.

Każda studnia w układzie CCD jest czujnikiem światła o bardzo dużej sprawności. Jak pokazano na rys. 23.7 b), pojedynczy foton uderzając w krzem, traci swoją energię na utworzenie dwóch cząstek naładowanych – jednego elektronu i jednej dziury. Dziura jest odpychana od elektrody, pozostawiając elektron tkwiący w studni, utrzymywany tam przez dodatni potencjał elektrody. Elektrony na rysunku są oznaczone symbolem „–" . Podczas **okresu całkowania** wzór obrazu światła padającego na układ CCD jest przekształcany na wzór ładunków w studniach CCD. Ciemniejsze źródła światła wymagają dłuższych okresów całkowania. Dla standardowego obrazu telewizyjnego okres całkowania jest 1/60 s (w Polsce 1/50 s – przyp. tłum.), a w astrofotografii światło można zbierać nawet przez wiele godzin.

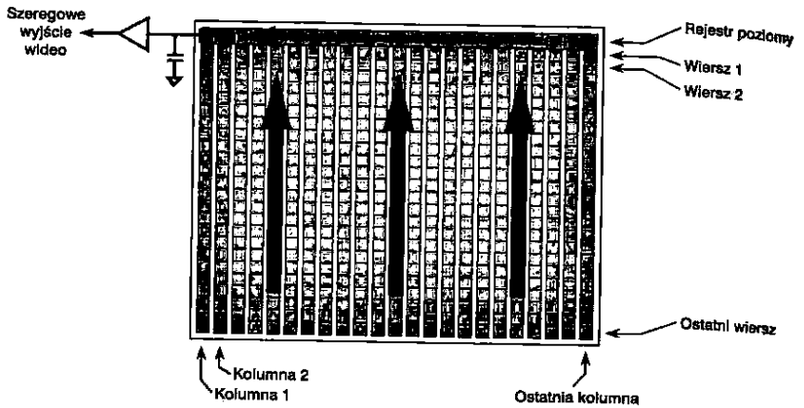
Odczyt obrazu elektronicznego jest całkiem prosty. Elektrony zebrane w każdej studni są *wypychane* w kierunku wzmacniacza wyjściowego. Napięcie dodatnie, jak pokazano na rys. 23.7 c), zostaje dołączone do *dwóch* linii fazowych. W ten sposób każda studnia rozszerza się w prawo. Następnym krokiem jest zdjęcie napięcia z pierwszej fazy (rys. 23.7 d), co powoduje, że poprzednio utworzone studnie zanikają. W ten sposób zebrane elektrony przesunęły się o jedną studnię w prawo. Powtarzając taką sekwencję taktowania w trzech liniach fazy, przesuwa się zgromadzone elektrony w prawo, aż dotrą do **wzmacniacza ładunkowego**. Ta nazwa dotyczy kondensatora połączonego ze wzmacniaczem o wzmocnieniu +1. Elektrony wypychane z ostatniej studni wpływają do kondensatora, wytwarzając w nim pewne napięcie. Pojemność kondensatora musi być bardzo mała (zwykle poniżej 1 pF), aby czułość wzmacniacza była duża. Ten kondensator wraz ze wzmacniaczem są częścią układu scalonego CCD i są wykonane z tego samego kawałka krzemu. Sygnał uzyskiwany z układu CCD jest sekwencją poziomów napięcia proporcjonalnych do ilości światła padającego na kolejne studnie.

Na rys. 23.8 przedstawiono, w jaki sposób obraz dwuwymiarowy jest odczytywany z CCD. Po okresie całkowania ładunek zgromadzony w każdej studni jest przesuwany w górę kolumny, za każdym razem o jeden wiersz. Na przykład wszystkie studnie w wierszu 15 są najpierw przesuwane do wiersza 14, potem do 13, do 12 itd. Za każdym razem, gdy wiersze są przesuwane w górę, wszystkie studnie z wiersza o numerze 1 są przekazywane do **rejestru poziomego**. W tym wierszu znajdują się specjalne studnie CCD, które bardzo szybko przesuwają ładunek w kierunku poziomym do wzmacniacza ładunkowego.

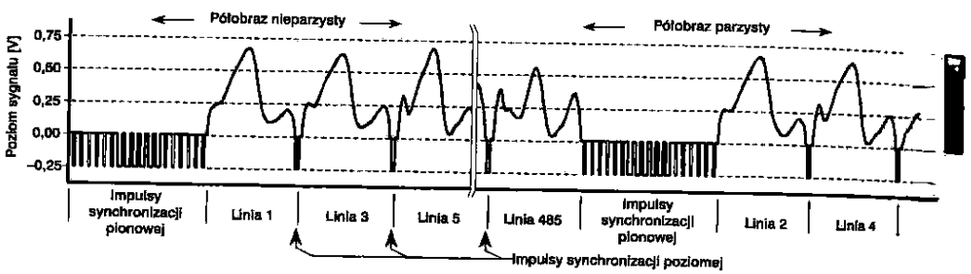
Zwróćcie uwagę, że taka architektura układu umożliwia przetwarzanie dwuwymiarowej matrycy danych w szeregowy strumień danych o określonej kolejności. Pierwszy odczytany piksel jest lewym górnym narożnikiem obrazu. Następnie odczyt przesuwa się od lewej do prawej strony w pierwszym wierszu, a potem od lewej ku prawej w kolejnych wierszach. Jest to tzw. odczyt z **główną kolejnością wg wierszy** (*row major order*), prawie zawsze obowiązującą przy konwersji matrycy (obrazu) dwuwymiarowej na dane szeregowe.

Telewizyjne sygnały wideo

Standardowy sygnał telewizyjny, choć ma już przeszło 50 lat, jest ciągle jednym z najpopularniejszych sposobów przesyłania obrazów. Na rys. 23.9 przedstawiono, jak sygnał telewizyjny wygląda na oscyloskopie. Jest to tzw. **całkowity sygnał wizji** zawierający impulsy synchronizacji poziomej i pionowej zmieszane z faktyczną informacją obrazową. Te impulsy są wy-



Rys. 23.8. Architektura układu CCD. Studnie układu CCD są ustawione w kolumny. Podczas odczytu ładunek z każdej studni jest przesuwany w górę kolumny aż do rejestru poziomego. Następnie ten rejestr jest sczytywany do przedwzmacniacza ładunkowego



Rys. 23.9. Całkowity sygnał wizji. Sygnał wideo w standardzie NTSC (obowiązującym w USA) składa się z 30 pełnych ramek (obrazów), z których każda zawiera od 480 do 486 linii wizyjnych. Każda ramka dzieli się na dwa półobrazy, z których jeden zawiera linie nieparzyste, a drugi – parzyste. Każdy półobraz zaczyna się grupą impulsów synchronizacji pionowej, po których następują kolejne linie informacji wizyjnej rozdzielane impulsami synchronizacji poziomej. (Na wykresie oś pozioma nie jest narysowana w skali)

korzystywane w odbiorniku telewizyjnym do synchronizacji układów odchylania poziomego i pionowego, aby można było wyświetlić obraz. Każda sekunda standardowego sygnału wizyjnego zawiera 30 pełnych obrazów[25], powszechnie nazywanych **ramkami**. Specjaliści od telewizji powiedzieliby, że każda ramka zawiera 525 **linii**, które w języku oprogramowania nazwalibyśmy raczej *wierszami*. Ta liczba linii jest nieco myląca, gdyż tylko 480 do 486 tych linii zawiera informacje wideo, pozostałe 39 do 45 linii zarezerwowano dla impulsów synchronizacji, aby utrzymać synchronizację działania układów telewizora z sygnałem wideo.

W standardzie telewizyjnym stosuje się **wybieranie międzyliniowe** zmniejszające *migotanie* wyświetlanego obrazu. Przy takim wybieraniu najpierw są przesyłane wszystkie linie nieparzyste każdej ramki, a potem linie parzyste. Grupa linii nieparzystych jest nazywana **półobrazem nieparzystym**, a parzystych – **półobrazem parzystym**. Każda ramka składa się z dwóch półobrazów. Jest więc w sygnale wizyjnym jest nadawanych 60 półobrazów na sekundę. Każdy półobraz rozpoczyna się od dość złożonej serii impulsów synchronizacji pionowej, trwającej 1,3 ms. Potem na

[25] Podane informacje dotyczą standardu NTSC obowiązującego w USA. W obowiązującym obecnie w Polsce standardzie PAL jest 25 pełnych obrazów na sekundę, a obraz zawiera 625 linii (przyp. tłum.)

stępują parzyste lub nieparzyste linie sygnału wizyjnego. Każda linia trwa 63,5 µs, w tym 10,2 µs dla impulsu synchronizacji poziomej, oddzielającego linię poprzednią od następnej. Napięcie analogowe w granicach każdej linii wyraża poziom skali szarości obrazu. Większe wartości napięcia odpowiadają jaśniejszemu obrazowi. Dlatego impulsy synchronizacji znajdują się poza obszarem czerni. Mówi się potocznie, że impulsy synchronizacji są *czarniejsze niż czerń*.

Układy stosowane do przetwarzania analogowo-cyfrowego sygnałów wideo noszą w terminologii angielskiej nazwę *frame grabber*. Mają zwykle formę karty z układem elektronicznym, wstawianej do komputera i połączonej z kamerą kablem koncentrycznym. Wykonując instrukcję programu, układ czeka na rozpoczęcie następnego obrazu, sygnalizowane impulsami synchronizacji pionowej. Podczas kolejnych dwóch półobrazów każda linia wideo jest wielokrotnie próbkowana, typowo 512, 640 lub 720 próbek na linię, 8 bitów na próbkę. Te próbki są gromadzone w pamięci jako jeden wiersz obrazu cyfrowego.

Taki sposób zbierania obrazu cyfrowego jest przyczyną ważnej różnicy między kierunkami poziomym i pionowym na obrazie. Każdy wiersz na obrazie cyfrowym odpowiada jednej linii w sygnale wizyjnym, a więc jednemu szeregowi studni w układzie CCD. Niestety z kolumnami sprawa nie jest tak prosta. W CCD każdy szereg zawiera od ok. 400 do 800 studni (a więc tyleż kolumn), zależnie od typu zastosowanego układu. Gdy szereg studni jest odczytywany z układu CCD, to uzyskana linia sygnału wizyjnego jest filtrowana, dając wygładzony sygnał analogowy, taki jak na rys. 23.9. Innymi słowy, sygnał wizyjny nie zależy od liczby kolumn w CCD. Rozdzielczość w kierunku poziomym jest ograniczona szybkością zmian sygnału analogowego. Ta szybkość w telewizji kolorowej odpowiada 3,2 MHz, z czego wynika czas narastania ok.100 ns, tj. ok. 1/500 linii wideo trwającej 53,2 µs.

Sygnał wizyjny, podczas digitalizacji w układzie *frame rabber*, jest z powrotem przetwarzany na kolumny. Jednak te kolumny w obrazie cyfrowym nie mają *żadnego związku* z kolumnami w CCD. Liczba kolumn w obrazie cyfrowym zależy wyłącznie od liczby próbek pobranych z każdej linii sygnału wizyjnego. Na przykład CCD może mieć 800 studni w wierszu, a zdigitalizowany obraz – tylko 512 pikseli (czyli kolumn) na wiersz.

Liczba kolumn w obrazie cyfrowym jest ważna jeszcze z jednej przyczyny. Standardowy obraz telewizyjny ma **współczynnik kształtu obrazu** 4:3, tj. szerokość obrazu jest nieco większa od jego wysokości. Filmy kinowe mają obraz szerszy, współczynnik kształtu jest 25:9. Układy CCD w zastosowaniach naukowych mają często współczynnik kształtu 1;1, a więc obraz jest kwadratowy. W każdym razie współczynnik kształtu w układzie CCD jest ustalony przez ustawienie elektrod i nie może być zmieniany. Jednak stosunek kształtu obrazu cyfrowego zależy od liczby próbek na linię. Jest to problemem, gdy obraz jest wyświetlany na monitorze lub drukowany. Jeśli współczynnik kształtu nie jest odpowiednio zachowany, to obraz jest ściśnięty w pionie lub w poziomie.

Opisany tu 525-liniowy sygnał telewizyjny jest nazywany sygnałem **NTSC** (*National Television Systems Committee*); jest to standard zdefiniowany w roku 1954. Ten system obowiązuje w USA i Japonii. W Europie są dwa inne standardy TV zwane **PAL** (*Phase Alternation by Line*) oraz **SECAM** (**Sequential Chrominance And Memory**). Podstawowa koncepcja jest taka sama, jak w NTSC, lecz liczba linii jest inna. Zarówno PAL, jak i SECAM charakteryzują się 25 ramkami (z wybieraniem międzyliniowym) na sekundę i 625 liniami na ramkę. Podobnie jak w NTSC, niektóre z tych linii występują podczas synchronizacji pionowej i w rezultacie tylko ok. 576 linii niesie informacje obrazowe. Inne, mniej istotne różnice dotyczą sposobu dodawania do sygnału informacji o barwie i dźwięku.

Najprostszym sposobem transmisji telewizji kolorowej byłoby przesyłanie trzech oddzielnych sygnałów analogowych, po jednym dla każdej z trzech barw, które wykrywa ludzkie oko: czerwonej, zielonej i niebieskiej. Niestety, historyczny rozwój telewizji uniemożliwił takie proste rozwiązanie. Sygnał telewizji kolorowej musiał bowiem być opracowany w taki sposób, aby umożliwiał dalsze korzystanie z istniejących odbiorników telewizji

czarno-białej. Osiągnięto ten cel, zachowując taki sam sygnał dla informacji o jaskrawości i dodając oddzielny sygnał z informacją o kolorze. Informacje o jaskrawości znajdują się w sygnale *luminancji*, a o kolorze – *chrominancji*. Sygnał chrominancji jest zawarty w fali o częstotliwości nośnej 3,58 MHz dodawanej do czarno-białego sygnału wizyjnego. W taki sam sposób wprowadzany jest dźwięk na fali nośnej 4,5 MHz. Odbiornik telewizyjny rozdziela te trzy sygnały, przetwarza je oddzielnie i łączy ze sobą w końcowej prezentacji.

Akwizycja i wyświetlanie innych obrazów

Nie wszystkie obrazy są odbierane w taki sposób, że za jednym razem jest odbierana jedna ramka. Inny często spotykany sposób nosi nazwę **skanowania liniowego**. Stosuje się wtedy detektor zawierający jednowymiarową matrycę pikseli, np. o długości 2048 pikseli i szerokości 1 piksela. Obraz jest zbierany linia po linii w miarę przesuwania się obiektu przed detektorem. Skanowanie liniowe jest stosowane w urządzeniach faksowych i rentgenowskich skanerach bagażu na lotniskach. Odmianą metody jest przesuwanie detektora wzdłuż nieruchomego obiektu. Jest to bardzo dogodne w sytuacjach, gdy detektor jest wmontowany w jakimś ruchomym obiekcie, takim jak samolot pobierający obrazy z ziemi pod nim. Zaletą skanowania liniowego polega na tym, że *prostotę detektora* uzyskano tu kosztem *szybkości*. Na przykład faksowanie całej strony tekstu może zająć kilka sekund, lecz uzyskany obraz zawiera tysiące wierszy i kolumn.

Jeszcze prostszą metodą jest zbieranie obrazu **punkt po punkcie**. Na przykład mikrofalowy obraz Wenus pobierano piksel po pikselu. Innym przykładem jest *sondujący mikroskop skaningowy*, który może dawać obrazy pojedynczych atomów. Mała sonda, często zawierająca tylko jeden atom na swym ostrzu, jest doprowadzana bardzo blisko próbki, która ma być zobrazowana. W ten sposób wykrywa się zjawiska mechaniki kwantowej zachodzące między sondą i próbką. To umożliwia zatrzymanie sondy w dokładnie określonej odległości od powierzchni próbki. Sonda następnie przesuwa się ponad powierzchnią próbki, utrzymując stałą odległość i śledząc uwypuklenie i wgłębienia. W końcowym obrazie wartość każdego piksela wyraża wartość wzniesienia odpowiadającego temu pikselowi miejsca na powierzchni próbki.

Obrazy drukowane dzieli się na dwie kategorie – prezentowane w **skali szarości** (*grayscale*) lub prezentowane **rastrowo** (*halftone*). W obrazowaniu w skali szarości każdy piksel ma pewien odcień szarości między czernią a bielą, jak np. na fotografii. W obrazowaniu rastrowym obraz jest tworzony z wielu pojedynczych *kropek*, z których każda jest albo całkiem czarna albo całkiem biała. Odcienie szarości uzyskuje się, zmieniając liczby kropek białych i czarnych. Wyobraźmy sobie drukarkę laserową o rozdzielczości 600 kropek na cal. Aby można było odtworzyć 256 poziomów jasności między bielą i czernią, każdy piksel powinien być równoważny matrycy drukowanych kropek, mającej wymiar 16 × 16. Czarne piksele powstają, gdy wszystkie 256 kropek w matrycy jest czarnych, a białe piksele – gdy jest 256 białych kropek. Średni odcień szarości uzyskuje się, gdy połowa kropek jest czarnych i połowa białych. Pojedyncze kropki są tak małe, że patrząc z normalnej odległości, nie można ich widzieć oddzielnie. Oko jest więc oszukiwane i odbiera obraz tak, jakby miał różne odcienie szarości.

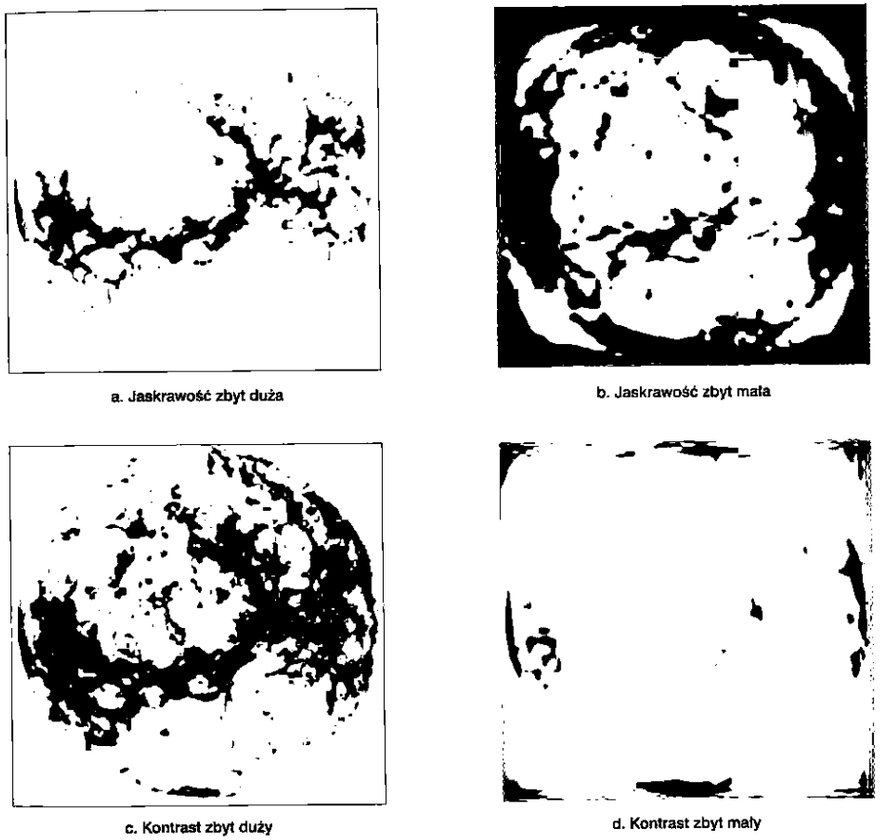
Obrazy rastrowe są łatwiejsze do realizacji w drukarkach i kopiarkach. Ich wadą jest jakość obrazu, często gorsza niż przy obrazowaniu w skali szarości.

Regulacja jaskrawości i kontrastu

Obraz, aby mógł być dobrze oglądany, powinien mieć odpowiednią **jaskrawość** i **kontrast**. Jaskrawość określa, czy obraz jest jaśniejszy, czy ciemniejszy. Kontrast jest *różnicą* jaskrawości między obiektami lub obszarami na obrazie. Na przykład biały królik biegnący po zaśnieżonym polu ma *słaby* kontrast, a czarny pies na tym samym tle – *dobry*

Na rys. 23.10 przedstawiono cztery możliwości złego doboru jaskrawości i kontrastu. Gdy jaskrawość jest zbyt duża (rys. 23.10 a), najbielsze piksele są nasycone, powodując zanikanie szczegółów w tych obszarach. Odwrotną sytuację pokazano na rys. 23.10 b), gdzie jaskrawość jest zbyt mała i nasycone są najczarniejsze piksele. Z powodu zbyt dużego kontrastu (rys. 23.10 c) obszary czarne stają się zbyt czarne, a białe – zbyt białe. Wreszcie, na rys. 23.10 d) mamy sytuację, gdy kontrast jest zbyt mały; wszystkie piksele mają średni odcień szarości i obraz się zaciera.

Na rys. 23.11 i 23.12 bardziej szczegółowo zilustrowano *jaskrawość* i *kontrast*. Na rys. 23.12 przedstawiono obrazy testowe o sześciu różnych poziomach jaskrawości i kontrastu. Na rys. 23.11 pokazano strukturę obrazu testowego. Jest to matryca o wymiarze 80 × 32 pikseli, które mogą mieć wartości od 0 do 255. Tło obrazu testowego jest wypełnione szumem losowym, rozmieszczonym równomiernie między 0 i 255. Na obrazie są trzy kwadraty o wartościach pikseli 75, 150 i 225, patrząc od lewej strony do prawej. Wewnątrz każdego kwadratu znajdują się dwa trójkąty z pikselami o wartościach niewiele różniących się od otoczenia. Innymi słowy, na obrazie mamy obszary ciemny, średnio ciemny i jasny, każdy z nich ze słabo wyróżniającymi się szczegółami.



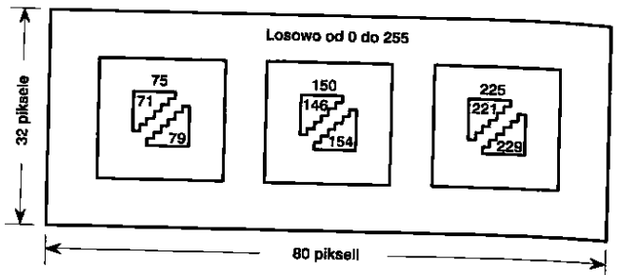
a. Jaskrawość zbyt duża

b. Jaskrawość zbyt mała

c. Kontrast zbyt duży

d. Kontrast zbyt mały

Rys. 23.10. Regulacja jaskrawości i kontrastu. Zwiększanie jaskrawości powoduje, że każdy piksel w obrazie staje się jaśniejszy. Natomiast zwiększania kontrastu powoduje, że jasne obszary stają się jaśniejsze, a ciemne ciemniejsze. Podane obrazy są przykładami złej jaskrawości i kontrastu



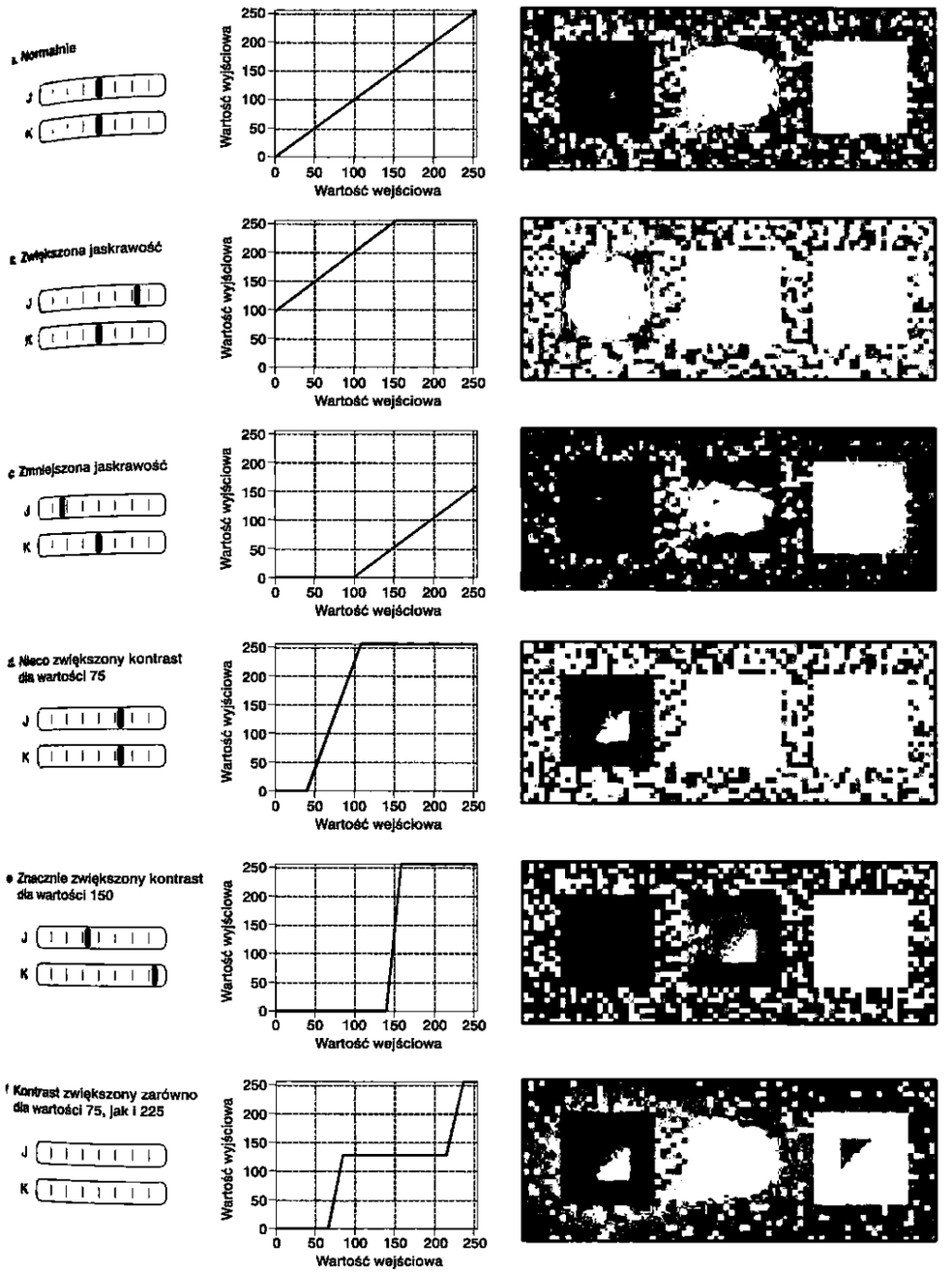
Rys. 23.11. Obraz do testowania jaskrawości i kontrastu. Takie obrazy zastosowano w przykładach podanych na rys. 23.12. Trzy kwadraty są obiektami – ciemny, średnio ciemny i jasny. Każdy z nich zawiera dwa trójkąty o małym kontraście. Liczby wskazują cyfrowe wartości pikseli w każdym obszarze

Na rys. 23.12 przedstawiono, jak ustawienie kontrastu i jaskrawości może uwidaczniać różne cechy obrazu. Na rys. 23.12 a) jaskrawość i kontrast są ustawione na *normalnym* poziomie, jak to wskazują **nastawniki suwakowe J i K** (jaskrawość i kontrast) z lewej strony rysunku. Zwróćmy teraz uwagę na wykresy pokazane przy każdym obrazie, zwane **charakterystyką wyjście/wejście, wyjściową tablicą przeglądową** lub **krzywą gamma**. Według tych zależności są sterowane układy ustawiające jaskrawość i kontrast. Wartość każdego piksela w zarejestrowanym obrazie, a więc pewna liczba z zakresu od 0 do 255, jest wprowadzana do takiej tablicy przeglądowej, która w odpowiedzi daje inną liczbę też z zakresu od 0 do 255. Ta nowa wartość cyfrowa steruje układem jaskrawości, w którym liczby od 0 do 255 są, odpowiednio, przetwarzane na różne odcienie od czerni do bieli. Tak więc tablica przeglądowa odwzorowuje zarejestrowane liczby w postaci jaskrawości obrazu.

Na rys. 23.12 a) pokazano, jak wygląda obraz, gdy tablica przeglądowa *niczego* nie wprowadza, to znaczy daje na wyjściu to samo, co ma na wejściu. Wszystkie piksele w zaszumionym tle mają losowo rozmieszczone różne odcienie szarości od bieli do czerni. Trzy kwadraty, wyświetlane jako jasny, średnio ciemny i ciemny, wyraźnie różnią się od siebie. Problem leży w tym, że trójkąty wewnątrz każdego kwadratu nie są dobrze widoczne. Kontrast jest zbyt mały, aby oko mogło odróżnić te obszary od ich otoczenia.

Rysunki 23.12 b), c) obrazują wpływ zmian jaskrawości. Wzrost jaskrawości powoduje przesuwanie charakterystyki wyjście/wejście *w lewo*, a jej zmniejszanie – przesuwanie *w prawo*. Zwiększanie jaskrawości powoduje, że *każdy* piksel na obrazie staje się jaśniejszy, a przy jej zmniejszaniu każdy piksel staje się ciemniejszy. Te zmiany mogą poprawić widoczność obszarów skrajnie jasnych bądź skrajnie ciemnych, lecz – jeśli doprowadzone są zbyt daleko – powodują **nasycenie** obrazu. Na przykład wszystkie piksele w kwadracie z prawej strony na rys. 23.12 b) są wyświetlane z pełną jaskrawością, tj. 255. Przeciwny efekt pokazano na rys. 23.12 c), gdzie wszystkie piksele w kwadracie z lewej strony są wyświetlane w najczarniejszej czerni, a więc mają wartość cyfrową 0. Ponieważ wszystkie piksele w tych obszarach mają jednakowe wartości, więc trójkąty są całkowicie wymazane. Zauważcie też, że *żaden* z trójkątów na rysunkach 23.12 b), c) nie jest widoczny lepiej niż trójkąty na rys. 23.12 a). Zmiana jaskrawości niewiele pomaga (lub wcale nie pomaga) w wyróżnianiu obiektów o małym kontraście z ich otoczenia.

Na rys. 23.12 d) przedstawiono wyświetlanie obrazu zoptymalizowane w celu uwidocznienia wartości pikseli wokół cyfrowej wartości 75. Optymalizację osiągnięto przez zmianę *kontrastu*, powodującą zmianę *nachylenia* charakterystyki wyjście/wejście. Na przykład zarejestrowane wartości pikseli 71 i 75 stają się na wyświetlanym obrazie wartościami 100 i 116, dając czterokrotnie większy kontrast. Wartości pikseli między 46 i 109 obejmują zakres wyświetlania od najczarniejszej czerni do najbielszej bieli. Ceną, jaką się płaci za poprawę kontrastu, jest to, że piksele o wartościach wejściowych od 0 do 45 są nasycone w czerni, a od 110 do 255 – nasycone w bieli. Jak pokazano na rys. 23.12 d), zwiększony kontrast umożliwia widoczność trójkątów w lewym kwadracie kosztem nasycenia kwadratu środkowego i prawego.



Rys. 23.12

Na rys. 23.12 e) przedstawiono wpływ jeszcze dalszego wzrostu kontrastu, który powoduje, że tylko 16 z możliwych 256 poziomów jest wyświetlanych jako nienasycone. Zwiększa też jaskrawość tak, że 16 nadających się do użytku poziomów znajduje się wokół wartości wejściowej 150. W środkowym kwadracie szczegóły są teraz dobrze widoczne, jednak poza tym prawie wszystko na obrazie jest nasycone. Spójrzmy, na przykład, na szum przy granicach obrazu. Jest tam bardzo niewiele pikseli o pośrednich odcieniach szarości. Prawie wszystkie piksele w tym obszarze są albo całkiem czarne, albo całkiem białe. Taka technika stosowania dużego kontrastu w celu zobaczenia tylko kilku poziomów szarości jest czasem nazywana **rozciąganiem skali szarości** (*grayscale stretch*).

Ustawianie kontrastu jest sposobem *rozszerzania* (*zoom*) jakiegoś małego zakresu wartości pikseli. Ustawianie jaskrawości powoduje umieszczenie rozszerzonego obszaru na wybranym wartościach pikseli. W większości cyfrowych systemów obrazujących możliwe jest ustawianie jaskrawości i kontrastu właśnie w taki sposób; często jest też wyświetlany wykres funkcji wyjście/wejście (jak na rys. 23.12). Inne rozwiązania zastosowano w telewizorach i monitorach ekranowych, gdzie do ustawiania jaskrawości i kontrastu służą *układy analogowe*, mogące działać w sposób odmienny od omówionego. Na przykład ustawianie kontrastu w monitorze może polegać na regulacji wzmocnienia sygnału analogowego, a jaskrawości – na dodawaniu lub odejmowaniu pewnego stałego poziomu. Wniosek z tego taki, że nie trzeba się dziwić, jeśli te regulacje analogowe nie zadziałają w sposób zgodny z oczekiwaniem.

Przekształcenia skali szarości

Ostatni obraz, przedstawiony na rys. 23.12 f), różni się od pozostałych. Wykres wyjście/wejście ma pewne nachylenie nie w *jednym*, a w *dwóch* obszarach wartości wejściowych. Dzięki temu na obrazie mogą być jednocześnie dobrze widoczne trójkąty zarówno w lewym, jak i w prawym kwadracie. Oczywiście powoduje to nasycenie pikseli *nieleżących* blisko wartości cyfrowych odpowiadających tym trójkątom. Zauważcie, że na tym rysunku nie pokazano położeń nastawników suwakowych jaskrawości i kontrastu; ustawienia są bowiem poza granicami możliwości tych nastawników.

Rozwijając dalej omówioną metodę, dochodzi się do mającej bardzo duże możliwości techniki udoskonalania wyglądu obrazów, noszącej nazwę **przekształcenia skali szarości**. Istota tej techniki jest zwiększanie kontrastu dla niektórych wartości pikseli, kosztem tych, które nie są interesujące. Realizuje się to, określając względną ważność każdej z możliwych wartości pikseli, od 0 do 255. Czym ważniejsza jest wartość piksela, tym większy kontrast jest ustawiany na obrazie. Implementację tej procedury przedstawimy na przykładzie.

Obraz pokazany na rys. 23.13 a) pobrano w zupełnej ciemności, stosując aparat fotograficzny CCD czuły na daleką podczerwień. Obrazowanym parametrem jest *temperatura*. Im obiekt jest cieplejszy, tym więcej emituje energii w podczerwieni i tym jaśniejszy jest na obrazie. Dlatego tło jest bardzo czarne (zimne), postać ludzka jest szara (ciepła) a maskownica silnika ciężarówki biała (gorąca). Takie systemy są świetne dla wojska i policji; można obserwować kogoś, kto nawet samego siebie nie może widzieć! Obraz na rys. 23.13 a) jest trudny do oglądania z powodu nierównomiernego rozkładu wartości pikseli. Większa część obrazu jest tak ciemna, że nie są widoczne żadne szczegóły. Z drugiej strony biel maskownicy silnika jest bliska nasycenia.

Histogram tego obrazu jest przedstawiony na rys. 23.14 a). Na tym histogramie tło, postać ludzka i maskownica silnika ciężarówki mają wartości wyraźnie różniące się od siebie. W tym przykładzie zwiększymy kontrast tła i maskownicy silnika kosztem pozostałych części obrazu, włącznie z postacią ludzką. Taki sposób podejścia zilustrowano na rys. 23.14 b). Deklarujemy, że piksele o najmniejszych wartościach, czyli tło, będą miały względny kontrast równy 12. Piksele zaś o największych wartościach, a więc maskownica silnika, będą miały względny kontrast 6, a postać ludzka – kontrast 1, a przejścia między obszarami powinny mieć przebieg schodkowy. Wszystkie te wartość są określane metodą prób i błędów



a. Pierwotny obraz w podczerwieni b. Po przekształceniu skali szarości

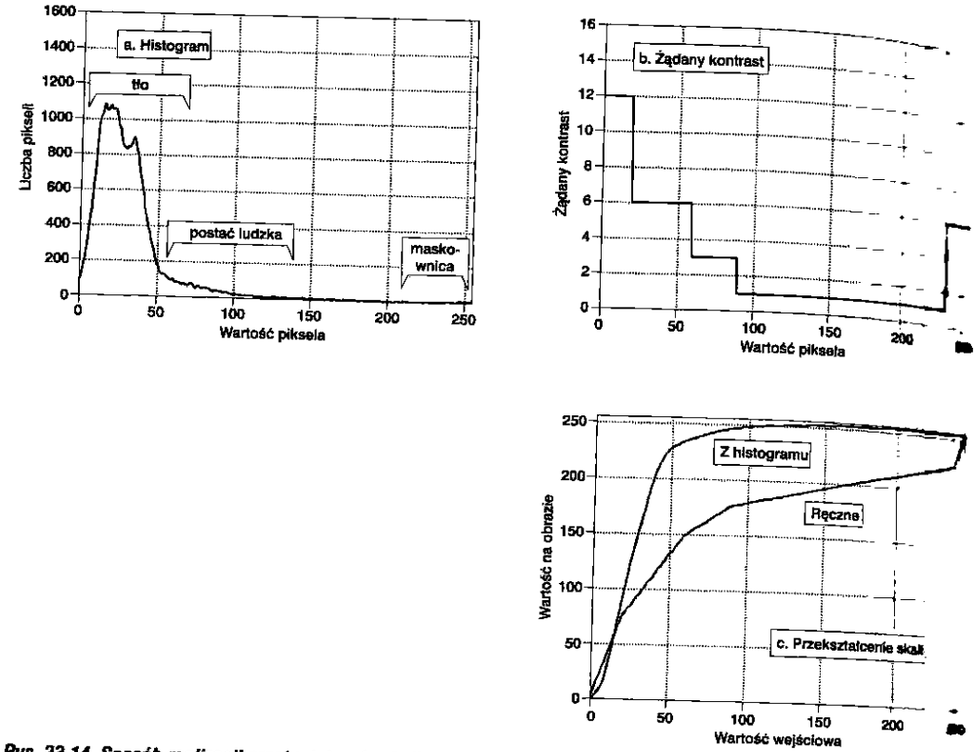
Rys. 23.13. Przetwarzanie obrazu w skali szarości. Obraz a) pobrano w zupełnej ciemności aparatem fotograficznym czułym na podczerwień. Jaskrawość obrazu zależy od temperatury obiektów, co wyraża wygląd ciepłej postaci ludzkiej i gorącej maskownicy silnika ciężarówki. Obraz pokazany na rys. b) został przetworzony zgodnie z krzywą ręcznego przekształcenia skali szarości przedstawioną na rys. 23.14 c)

Przekształcenie skali szarości wynikające z opisanej metody pokazano na rys. 23.14 c) na wykresie z napisem *ręczne*. Tę krzywą wyznaczono, obliczając sumę bieżącą (tj. całkę dyskretną) krzywej z rys. 23.14 b) i następnie normalizując ją w taki sposób, aby z prawej strony wykres miał wartość 255. Dlaczego trzeba obliczać *całkę*, aby wyznaczyć żądaną krzywą? Pomyślmy o tym w następujący sposób. Kontrast przy jakiejś konkretnej wartości piksela jest równy nachyleniu krzywej wyjście/wejście. A więc chcemy, żeby krzywa na rys. 23.14 b) był pochodną (nachyleniem) krzywej z rys. 23.14 c). Z tego wynika, że krzywa na rys. 23.14 c) musi być całką krzywej z rys. 23.14 b).

Przepuszczając obraz z rys. 23.13 a) przez takie, ręcznie określone przekształcenie skali szarości, uzyskujemy obraz pokazany na rys. 23.13 b). Tło stało się *jaśniejsze*, maskownica silnika *ciemniejsza* i oba te obiekty uzyskały lepszy kontrast. Te ulepszenia osiągnięto kosztem kontrastu postaci ludzkiej, uzyskując mniej szczegółowy obraz intruza (choć nie może już być dużo gorszy niż na obrazie oryginalnym).

Przekształcenia skali szarości mogą w znacznym stopniu poprawić widoczność obrazu. Problem w tym, że metoda wymaga bardzo wielu prób i błędów. Sposobem automatyzacji tego procesu jest **wyrównywanie histogramu**. Zauważcie, że histogram na rys. 23.14 a) i krzywa ważona kontrastu mają taki sam kształt ogólny. W metodzie wyrównywania histogramu stosuje się bezpośrednio histogram jako krzywą ważoną kontrastu, eliminując konieczność oceny człowieka. A zatem wykres wyjście/wejście jest wyznaczany przez całkowanie i normalizację *histogramu*, a nie przez ręcznie generowaną krzywą. W rezultacie uzyskuje się większy kontrast dla tych wartości, które występują w największej liczbie pikseli.

Wyrównywanie histogramu jest interesującą procedurą matematyczną, gdyż maksymalizuje *entropię* obrazu, a więc miarę tego, ile informacji jest przesyłanych w ustalonej liczbie bitów. Wadą wyrównywania histogramu jest pomylenie zwykłej *liczby* pikseli o pewnej wartości z *ważnością* pikseli o tej wartości. Na przykład maskownica silnika ciężarówki i postać intruza są najistotniejszymi cechami obrazu na rys. 23.13. Mimo to wyrównywanie histogramu powoduje prawie zupełne pominięcie tych obiektów, gdyż zawierają one stosunkowo mało pikseli. Wyrównywanie histogramu jest szybkie i łatwe. Miejcie na uwadze, że jeśli nie działa ono dobrze, to krzywa generowana ręcznie przypuszczalnie zadziała lepiej.



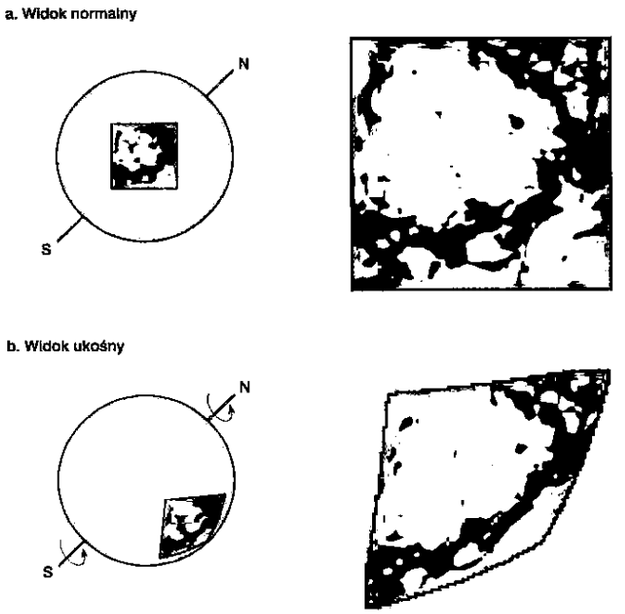
Rys. 23.14. Sposób realizacji przekształcenia skali szarości. Na rys. a) pokazano histogram obrazu z rys. 23.13 a), Krzywą na rys. b) utworzono ręcznie, pokazując, jaki kontrast jest pożądany przy każdej wartości piksela. Następnie, przez całkowanie i normalizację wykresu b) jest wyznaczana tablica przeglądowa przekształcenia skali, dając krzywą oznakowaną jako ręczna na rys. c). W procesie wyrównywania histogram obrazu oryginalnego (rys. a) jest całkowany i normalizowany, aby wyznaczyć tablicę przeglądową pokazaną na rys. c)

Odkształcanie obrazów (*warping*)

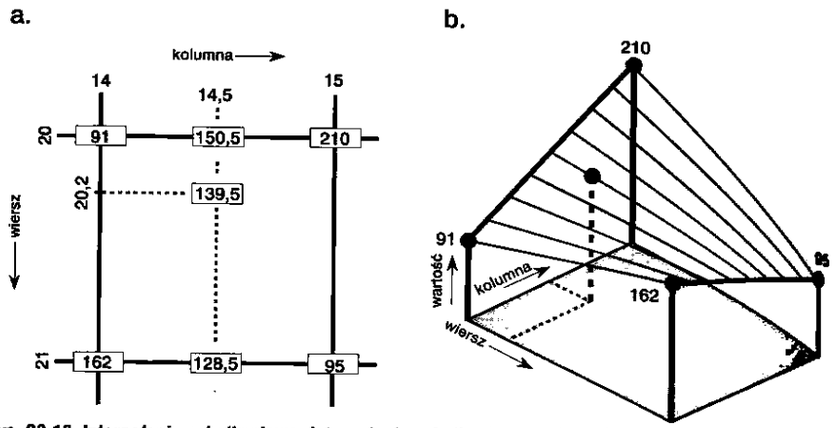
Przy fotografowaniu powierzchni planet jednym z problemów jest zniekształcenie spowodowane krzywizną powierzchni sferycznej. Powiedzmy, że stosujemy teleskop do fotografowania kwadratowego obszaru w pobliżu środka powierzchni jakiejś planety (rys. 23.15 a). Planeta obraca się wokół własnej osi, więc po kilku godzinach będziemy mieli widok przedstawiony na rys. 23.15 b). Obszar poprzednio sfotografowany wygląda jak mocno odkształcony, gdyż jest zakrzywiony w pobliżu horyzontu planety. Każdy z tych dwóch obrazów zawiera pełną informację o obszarze, lecz widzianą z innej perspektywy. Często wykonuje się fotografię taką, jak na rys. 23.15 a), lecz chciałoby się, żeby obraz wyglądał jak na rys. 23.15 b) lub vice versa. Na przykład wykonanie mapy powierzchni planety przez satelitę może wymagać tysięcy zdjęć zrobionych prosto z góry, jak na rys. 23.15 a). Aby wykonać obraz planety wyglądający naturalnie, tak jak obraz Wenus na rys. 23.1, trzeba każdy obraz odkształcić i umieścić we właściwym miejscu. Jako inny przykład rozważmy satelitę meteorologicznego prowadzącego obserwacje huraganu, trwającego nie bezpośrednio pod satelitą. Nie pozostaje nic innego, jak pobrać obraz skośnie, jak na rys. 23.15 b). Obraz jest potem przetwarzany na taki, jakby był fotografowany prosto z góry, czyli jak na rys. 23.15 a).

Takie transformacje przestrzenne nazywamy **odkształceniami** (*warping*). Najczęstszym, choć nie jedynym zastosowaniem odkształcania jest fotografia kosmiczna. Na przykład w wielu lampowych detektorach obrazu występują, w różnym stopniu, odkształcenia przestrzenne. Dotyczy to stosowanych w wojsku aparatów fotograficznych z noktowizorami oraz medycznych detektorów promieniowania rentgenowskiego. Można stosować odkształcanie cyfrowe w celu skorygowania zniekształceń nieodłącznie związanych z tymi urządzeniami. Także twórcy filmowych efektów specjalnych bardzo lubią odkształcanie obrazów. Na przykład technika zwana **morfingiem** daje stopniowe przechodzenie jednego obrazu w drugi w ciągu obrazów. W ten sposób osiąga się iluzje takie, jak przetwarzanie się dziecka w dorosłego człowieka lub mężczyzny w wilka.

Odkształcanie polega na pobraniu *obrazu oryginalnego* (matryca dwuwymiarowa) i generowaniu *obrazu odkształconego* (inna matryca dwuwymiarowa). Wykonuje się to, tworząc pętlę wokół każdego piksela w obrazie odkształconym i zapytaniu: Jaka jest właściwa wartość piksela, który powinien się tu znajdować? Mając dane numery wiersza i kolumny obliczone w obrazie odkształconym, znajdujemy odpowiadające im numery wiersza i kolumny w obrazie oryginalnym. Wartość piksela z obrazu oryginalnego jest przez algorytm przekształcana na wartość w obrazie odkształconym. W języku używanym w dziedzinie przetwarzania obrazów wiersz i kolumna, *z których* pochodzi piksel w obrazie oryginalnym, są nazywane **adresem pochodzenia** (*comes-from address*). Przeniesienie każdego piksela z obrazu oryginalnego do odkształconego jest sprawą łatwą. Trudną kwestią zaś jest obliczenie *adresu pochodzenia* związanego z każdym pikselem w obrazie odkształconym. Jest to zwykle problem czysto matematyczny, który często może być dość zawikłany. Proste rozciągnięcie obrazu w kierunku poziomym lub pionowym jest łatwiejsze, gdyż wymaga tylko mnożenia numeru wiersza i/lub kolumny, aby określić adres pochodzenia.



Rys. 23.15. Odkształcanie obrazu. Jak wynika z rys. a) normalny widok małego obszaru wydaje się raczej wolny od odkształceń. Natomiast obraz ukośny (rys. b) wykazuje znaczne odkształcenia przestrzenne. Odkształcanie jest techniką zamiany jednego z tych obrazów w drugi



Rys. 23.16. Interpolacja subpikselowa. Interpolacja subpikselowa służąca do odkształcania obrazu jest zwykle wykonywana przy użyciu interpolacji dwuliniowej. Dwie wartości pośrednie są obliczane przez interpolację liniową w kierunku poziomym. Następnie wartość końcowa jest obliczana przez interpolację liniową w kierunku pionowym między dwiema wartościami pośrednimi. Jak pokazano na trójwymiarowej ilustracji b) ta procedura określa wszystkie wartości między czterema znanymi pikselami w każdym z narożników

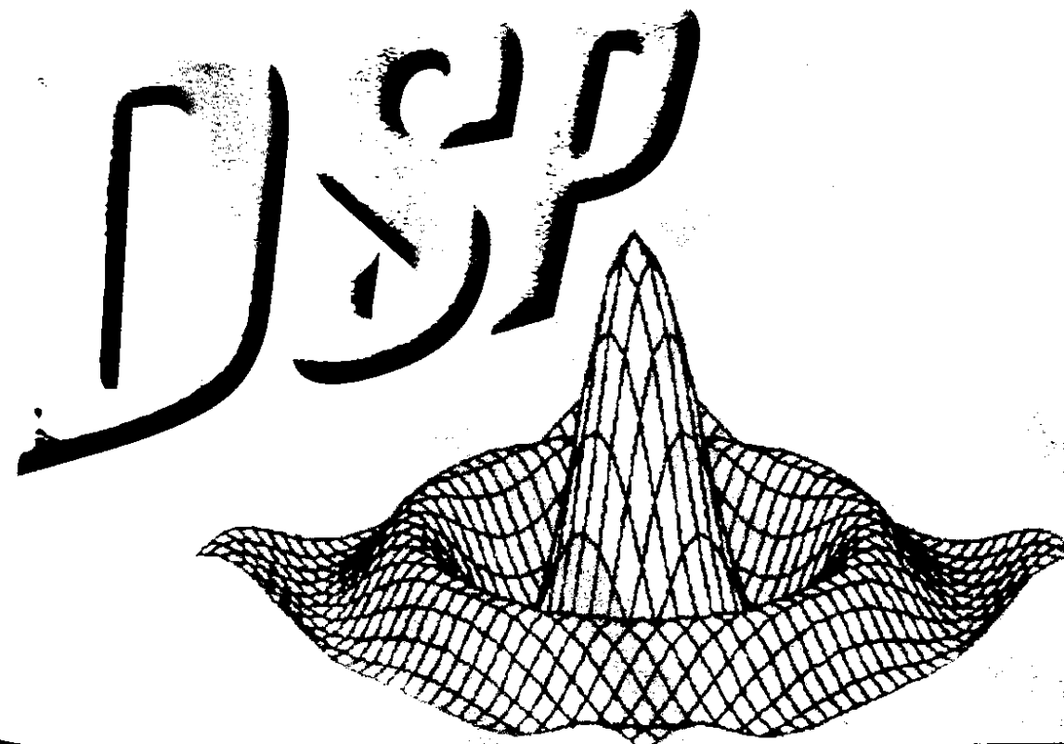
Jedną z technik stosowanych w odkształcania obrazu jest **interpolacja subpikselowa**. Załóżmy na przykład, że opracowaliście układ równań, które przekształcają adres wiersza i kolumny w obrazie odkształconym na adres pochodzenia w obrazie oryginalnym. Rozważcie, co się może zdarzyć, gdy spróbujecie znaleźć wartość piksela w wierszu 10 i kolumnie 20 w obrazie odkształconym. Wstawiacie informację: *wiersz = 10, kolumna = 20* do waszych równań i w zamian uzyskujemy: *wiersz pochodzenia = 20,2, kolumna pochodzenia = 14,5*. Przypuszczalnie zastosowaliście obliczenia zmiennoprzecinkowe i dlatego adresy pochodzenia nie są liczbami całkowitymi. W tej sytuacji najłatwiejszą metodą jest zastosowanie algorytmu **najbliższego sąsiedztwa**, który po prostu zaokrągla adresy do najbliższej liczby całkowitej. To jest proste, lecz może wytwarzać ziarnistość na krańcach obiektów, gdzie piksele mogą wyglądać na trochę niewłaściwie umieszczone.

Interpolacja dwuliniowa wymaga wprawdzie trochę więcej wysiłku, lecz daje znacznie lepsze obrazy. Działanie tej metody zilustrowano na rys. 23.16. Znacie wartości czterech pikseli *wokół* adresu ułamkowego, tj. wartości pikseli w wierszach 20 i 21 oraz w kolumnach 14 i 15. W tym przykładzie założymy, że wartości pikseli są 91, 210, 162 i 95. Trzeba wykonać interpolację między tymi czterema wartościami. Robi się to w dwóch krokach. Po pierwsze, przeprowadza się interpolację w kierunku *poziomym*, między kolumnami 14 i 15. To daje dwie wartości pośrednie; 150,5 w wierszu 20 oraz 128,5 w wierszu 21. Po drugie, przeprowadza się interpolację między tymi wartościami pośrednimi w kierunku pionowym. To daje interpolowaną dwuliniowo wartość piksela równą 139,5[26], która jest następnie przekazywana do odkształcanego obrazu. Dlaczego interpolujemy najpierw w kierunku poziomym a *następnie* w pionowym, zamiast odwrotnie? To nie ma znaczenia; końcowy wynik nie zależy od kolejności tych interpolacji.

[26] Zdaniem tłumacza powinna to być wartość 146,1.

24

Liniowe przetwarzanie obrazów



Podstawą liniowego przetwarzania obrazów są te same dwie metody, które stosuje się w konwencjonalnym DSP – *splot* i *analiza Fouriera*. Ważniejszą z nich jest metoda splotu, gdyż obrazy mają informację zakodowaną raczej w dziedzinie przestrzeni niż w dziedzinie częstotliwości. Filtracja liniowa może ulepszać obrazy kilkoma sposobami, przez: wyostrzanie krawędzi obiektów, tłumienie szumu losowego, korekcję nierównego oświetlenia, rozplot lub usuwanie obrazy rozmyte, poruszone itd. Te procedury są przeprowadzane przez splatanie obrazu oryginalnego z właściwą odpowiedzią impulsową filtru. W ten sposób uzyskuje się przefiltrowany obraz. W realizacji splotu obrazowego poważną trudnością jest konieczność wykonania ogromnej liczby obliczeń, z czego wynikają bardzo długie, niemożliwe do zaakceptowania czasy wykonania. W tym rozdziale przedstawiono metody projektowania odpowiedzi impulsowych filtrów dla różnych zadań związanych z przetwarzaniem obrazów. Opisano też dwie ważne techniki skracające czas wykonania: *splot przez rozłączność* oraz *splot FFT*.

Splot

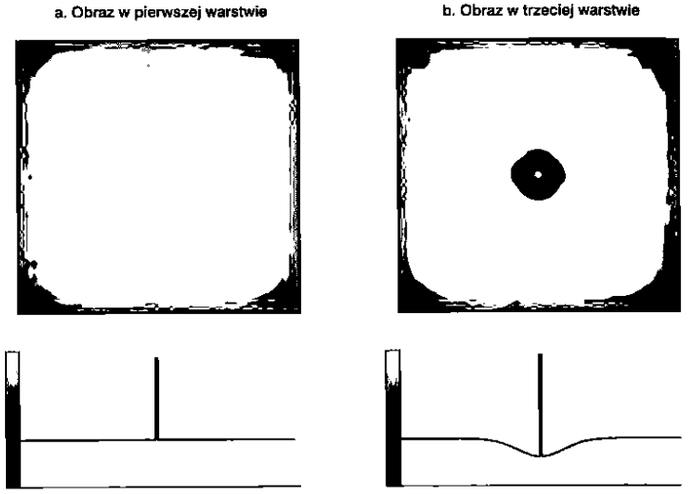
Splot obrazowy działa w ten sam sposób, jak splot jednowymiarowy. Na przykład, na obrazy można patrzeć jak na sumowanie *impulsów*, tj. przeskalowanych i przesuniętych funkcji delta. Również *systemy liniowe* są charakteryzowane ich odpowiedziami na *impulsy*, czyli przez *odpowiedzi impulsowe*. Jak można się było spodziewać, obraz uzyskany na wyjściu systemu jest równy *splotowi* obrazu wejściowego i odpowiedzi impulsowej systemu.

Dwuwymiarowa funkcja delta jest obrazem złożonym z samych zer, oprócz jednego piksela znajdującego się w *wierszu* = 0, *kolumnie* = 0, który ma wartość *jeden*. Załóżmy na razie, że indeksy wierszy i kolumn mogą mieć wartości zarówno dodatnie, jak i ujemne, tak więc ta jedynka jest umieszczona pośrodku ogromnego morza zer. Gdy funkcja delta przechodzi przez system liniowy, pojedynczy punkt niezerowy będzie zamieniony na jakiś inny obraz dwuwymiarowy. Ponieważ jedyną rzeczą, jaka się może zdarzyć punktowi, jest jego *rozciągnięcie się*, więc ta odpowiedź impulsowa jest często w żargonie specjalistów w dziedzinie przetwarzania obrazów nazywana **funkcją rozmycia punktu PSF** (*Point Spread Function*).

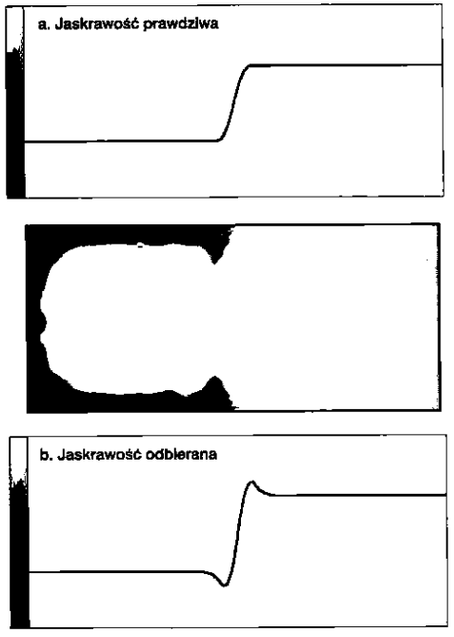
Doskonałym przykładem tych koncepcji jest ludzkie oko. Jak napisano w poprzednim rozdziale, pierwsza warstwa siatkówki przetwarza obraz wyrażony jakimś rozkładem światła na obraz będący pewnym rozkładem w impulsach nerwowych. W drugiej warstwie siatkówki następuje *przetwarzanie* tego obrazu neuralnego i przekazanie go do trzeciej warstwy – tkanki tworzącej nerw wzrokowy. Wyobraźmy sobie, że obraz rzutowany na siatkówkę jest bardzo małą plamką świetlną pośrodku ciemnego tła. A więc do oka jest wprowadzany *impuls*. Przy założeniu, że system jest liniowy, przetwarzanie obrazu następujące w siatkówce może być określone przez sprawdzenie obrazu pojawiającego się na nerwie wzrokowym. Innymi słowy, chcemy wyznaczyć *funkcję rozmycia punktu* tego przetwarzania. Do założenia dotyczącego liniowości oka wrócimy jeszcze w dalszej części tego rozdziału.

Ten eksperyment naszkicowano na rys. 24.1. W siatkówkę trafia impuls świetlny (rys. 24.1 a), dając obraz pojawiający się na nerwie wzrokowym. Środkowa warstwa oka przepuszcza jasny impuls szpilkowy, lecz wytwarza kolisty obraz *ciemniejszy*. Oko realizuje to przez proces zwany wzbranianiem bocznym (*lateral inhibition*). Jeśli jest pobudzana komórka nerwowa w warstwie środkowej, to zmniejsza ona możliwość pobudzenia innych komórek znajdujących się w najbliższym sąsiedztwie. Gdy oko widzi cały obraz, każdy punkt w obrazie wnosi przeskalowaną i przesuniętą wersję odpowiedzi impulsowej do obrazu pojawiającego się na nerwie wzrokowym. Innymi słowy, obraz wzrokowy podlega *splotowi* z funkcją rozmycia punktu (PSF) oka, dając obraz neuronowy przekazywany do mózgu. Nasuwa się oczywiste pytanie – jak splatanie obrazu wzrokowego z PSF poprawia zdolność oka do rozumienia świata?

Ludzie i zwierzęta korzystają ze wzroku do identyfikacji pobliskich obiektów, takich jak wrogowie, pożywienie i przyjaciele. Identyfikacja jest realizowana przez rozróżnianie poszczególnych obszarów obrazu na podstawie różnic jaskrawości i barwy. Pierwszym krokiem w rozpo-



Rys. 24.1. Funkcja rozmycia punktu (PSF) oka. Środkowa warstwa siatkówki zamienia impuls pokazany na rys. a) na impuls otoczony obszarem ciemnym (rys. b). Taka funkcja rozmycia punktu wyostrza krawędzie obiektów



Rys. 24.2. Pasma Macha. Przetwarzanie obrazu w siatkówce powoduje, że powoli zmieniające się krawędzie (rys. a) ulegają wyostrzeniu (rys. b). Dzięki temu rozdzielanie obiektów na obrazie jest łatwiejsze, powstaje jednak złudzenie optyczne zwane pasmami Macha. W wyniku przerzutu niedaleko krawędzi obszar ciemny wydaje się ciemniejszy, a jasny – jaśniejszy. Powstają więc pasma jasne i ciemne, biegnące równolegle do krawędzi

znawaniu obiektu jest więc identyfikacja jego *krawędzi*, czyli nieciągłości oddzielającej od tła. Środkowa warstwa siatkówki wspomaga to zadanie, zaostrzając krawędzie ogo obrazu. Jako ilustrację takiego działania, na rys. 24.2 przedstawiono obraz, który przemienia się z ciemnego w jasny, wytwarzając rozmycia i słabo zaznaczone krawędzie. rys. 24.2 a) pokazano profil natężenia światła tego obrazu, czyli profil jaskrawości padającego na oko, a na rys. 24.2 b) – profil jaskrawości obrazu pojawiającego się na wzrokowym i przekazywanego do mózgu. Przetwarzanie obrazu w siatkówce powoduje, krawędzie między obszarami jasnymi i ciemnymi stają się ostrzejsze, wyraźniej podkreślone różnice między dwoma obszarami.

Przerzut pojawiający się w odpowiedzi na krawędzie tworzy ciekawe złudzenie optyczne. Wydaje się, że ciemny obszar obok krawędzi jest niezwykle ciemny, a jasny obszar – nie zwykle jasny. Wynikające z tego paski jasne i ciemne są nazywane **pasmami Macha**, od nazwiska Ernsta Macha (1838–1916), austriackiego fizyka, który pierwszy je opisał.

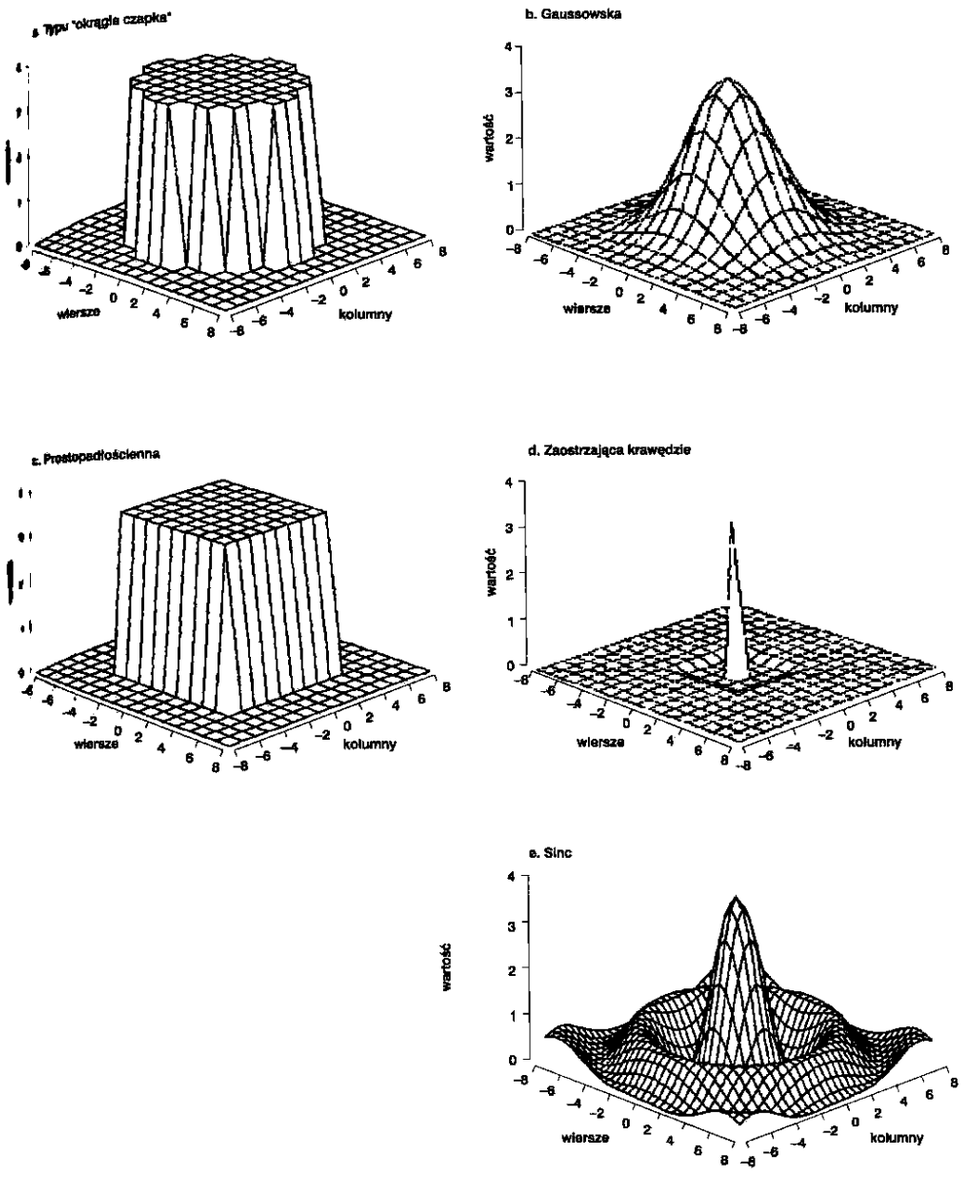
Podobnie jak w przypadku sygnałów jednowymiarowych, na splot obrazowy można spojrzeć w dwojaki sposób – od strony wejścia lub od strony wyjścia. Od strony wejścia każdy piksel w obrazie wejściowym ma swój udział w obrazie wyjściowym, w postaci przeskalowanej i przesuniętej wersji funkcji rozmycia punktu. Przy rozpatrywaniu od strony wyjścia na każdy piksel w obrazie wyjściowym ma wpływ grupa pikseli z sygnału wejściowego. Dla sygnałów jednowymiarowych ten obszar wpływu jest odpowiedzią impulsową odwróconą *lewo-prawostronnie* oraz *góra–dół*. Ponieważ stosowane w DSP funkcje PSF są w większości symetryczne wokół osi poziomej i pionowej, więc takie odwrócenie niczego nie zmienia i może być pominięte. W dalszym ciągu tego rozdziału poświęcimy nieco uwagi niesymetrycznym funkcjom PSF, przy których trzeba uwzględniać takie odwracanie.

Na rys. 24.3 przedstawiono kilka często spotkanych funkcji PSF. Funkcja typu „okrągła czapka" (*pillbox*) (rys. 24.3 a) ma okrągły wierzchołek i proste boki. Jeśli, na przykład, soczewka aparatu fotograficznego nie jest prawidłowo zogniskowana, to każdy punkt w obrazie będzie rzutowany jako okrągła plamka na czujniku obrazowym (wróćcie do rys. 23.2 i rozważcie wpływ ruchu ekranu projekcyjnego w kierunku do i od soczewki). Można powiedzieć, że funkcja rozmycia punktu typu „okrągła czapka" jest funkcją źle zogniskowanych soczewek.

Funkcja gaussowska (rys. 24.3 b) jest funkcją PSF systemu obrazowego ograniczonego doskonałościami *losowymi*. Na przykład obraz z teleskopu jest rozmyty przez turbulencje atmosferyczne, co powoduje, że każdy punkt świetlny staje się w końcowym obrazie przebiegiem gaussowskim. Działanie czujników obrazowych, takich jak CCD i siatkówka, jest często ograniczone rozpraszaniem światła i/lub elektronów. Z centralnego twierdzenia granicznego wynika, że rozmycie gaussowskie jest spowodowane procesami losowymi tego rodzaju.

Rozkłady typu „okrągła czapka" i gaussowski są stosowane w przetwarzaniu obrazów w taki sposób, jak *filtr o zasadzie średniej kroczącej* jest stosowany do sygnałów jednowymiarowych. Obraz poddany splotowi z tymi funkcjami PSF zrobi się rozmyty i będzie miał słabiej zarysowane krawędzie. Będzie jednak zawierał mniej szumu losowego. Te funkcje są zwane **filtrami wygładzającymi**, z racji ich działania w dziedzinie czasu lub **filtrami dolnoprzepustowymi** pod względem działania w dziedzinie częstotliwości. **Prostopadłościenna** funkcja PSF (rys. 24.3 c) też może być używana jako filtr wygładzający, lecz nie ma symetrii kołowej. Dlatego rozmycia będą inne w kierunku po przekątnej niż w kierunku poziomym lub pionowym. To może mieć znaczenie lub nie w zależnie od zastosowania.

Przeciwieństwem filtru wygładzającego jest filtr **zaostrzający krawędzie**, czyli **filtr górnoprzepustowy**. Technika inwersji widmowej, opisana w rozdziale 14, jest stosowana do przekształcania jednych filtrów w drugie. Jak pokazano na rys. 24.3 d), odpowiedź impulsowa filtru zaostrzającego krawędzie jest określana przez wykonanie *negacji* filtru wygładzającego i dodanie funkcji delta pośrodku. Przykładem filtru tego rodzaju jest przetwarzanie obrazu zachodzące w siatkówce.



Rys. 24.3. Często spotykane funkcje rozmycia punktu (PSF). Przedstawione na rys. a), b), c) funkcje: typu „okrągła czapka", gaussowska oraz prostopadłościenna są stosowane jako filtry wygładzające (dolnoprzepustowe). Filtry zaostrzające krawędzie (górnoprzepustowe) są tworzone przez odejmowanie dolnoprzepustowej odpowiedzi impulsowej filtru od impulsu (rys. d). Funkcja sinc (rys. e) jest rzadko stosowana w przetwarzaniu obrazu, ponieważ informacja obrazowa jest zakodowana w dziedzinie przestrzeni, a nie częstotliwości

Na rys. 24.3 e) przedstawiono dwuwymiarową funkcję *sinc*. W przetwarzaniu sygnału jedno-wymiarowego stosuje się okienkowanie funkcją *sinc* w celu rozdzielania pasm częstotliwości. Ponieważ informacja obrazowa nie jest zakodowana w dziedzinie częstotliwości, więc funkcja *sinc* jest rzadko stosowana jako odpowiedź impulsowa filtru obrazu, chociaż jest przydatna w niektórych zagadnieniach teoretycznych. Funkcja *sinc* jest trudna w stosowaniu, bo amplituda jej „ogonów" maleje bardzo wolno, (jak $1/x$), a więc przebieg funkcji musi być traktowany jako nieskończenie długi. Natomiast „ogony" funkcji gaussowskiej zmniejszają się bardzo gwałtownie, (jak e^{-x^2}) i mogą być obcięte bez żadnych złych skutków.

W tych wszystkich odpowiedziach impulsowych filtrów są stosowane *ujemne* indeksy, dzięki czemu funkcja PSF może być wypośrodkowana dla *wiersza* = 0 i *kolumny* = 0. W jedno-wymiarowym DSP ujemne indeksy są często eliminowane przez przesuwanie odpowiedzi impulsowej w prawo tak dalece, aż wszystkie niezerowe próbki będą miały indeksy dodatnie. To przesunięcie przesuwa też sygnał wyjściowy o taką samą wartość, co zwykle nie ma istotnego znaczenia. Natomiast przesunięcie między obrazami wyjściowym a wejściowym jest zwykle nie do przyjęcia. Dlatego w przetwarzaniu obrazów są z zasady stosowane indeksy ujemne w odpowiedziach impulsowych filtrów.

Trudnością, jaką się napotyka w splocie obrazów, jest bardzo wielka liczba obliczeń. Gdy na przykład obraz 512 × 512 pikseli jest poddawany splotowi z funkcją PSF 64 × 64 pikseli, to trzeba wykonać ponad *miliard* mnożeń i dodawań (tj. 64 × 64 × 512 ×512). Długi czas wykonania może uczynić tę metodę nieużyteczną w praktyce. Skrócić ten czas można w trojaki sposób.

Pierwszym rozwiązaniem jest zastosowanie bardzo małej funkcji PSF, często tylko 3 × 3 piksele. Realizuje się to przez pętlę w każdej próbce obrazu wyjściowego, zastosowanie zoptymalizowanego programu do mnożenia i sumowania odpowiednich dziewięciu pikseli z obrazu wyjściowego. Używając funkcji PSF tylko 3 × 3, można wykonać zadziwiająco dużo przetwarzania, a w każdym razie na tyle dużo, żeby wpłynąć na *krawędzie* obrazu.

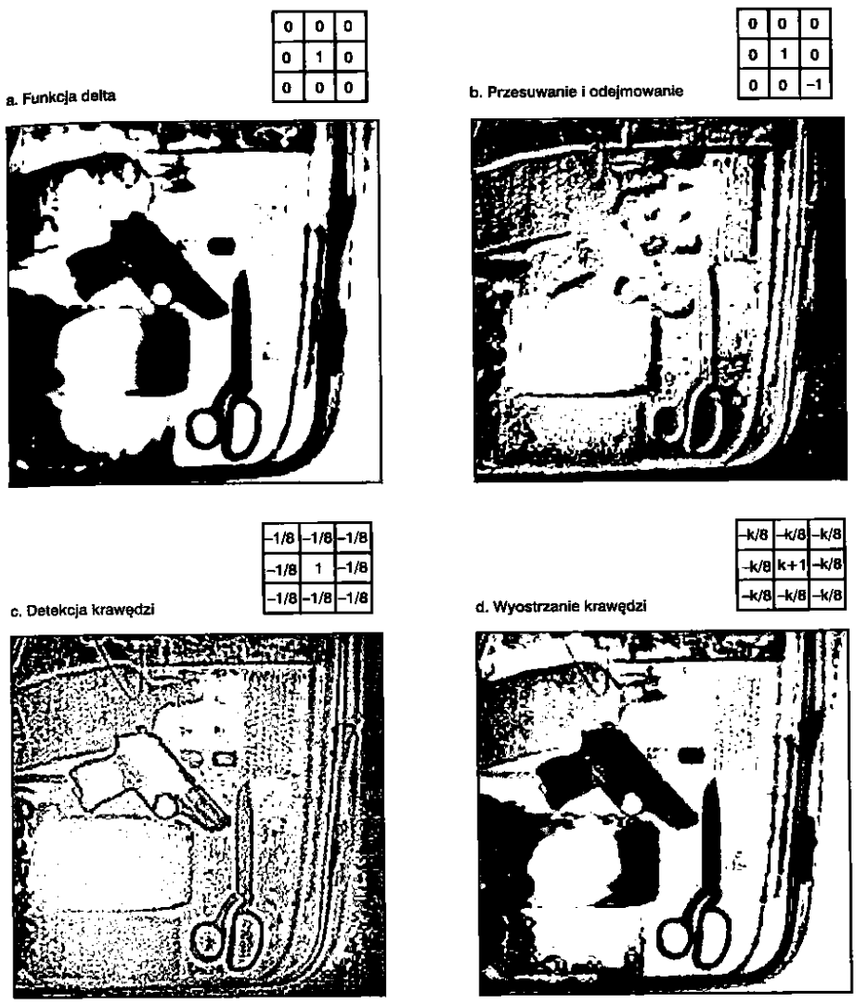
Z drugiego sposobu korzysta się, gdy jest konieczna duża funkcja PSF, lecz jej kształt nie jest krytyczny. To wymaga odpowiedzi impulsowej, która jest *rozłączna* (*separable*). Ta właściwość umożliwia wykonanie splotu obrazu przez szereg operacji jednowymiarowych. W ten sposób można *setki* razy zwiększyć szybkość wykonywania.

Trzecim rozwiązaniem jest splot FFT, stosowany, gdy odpowiedź impulsowa filtru jest duża i ma specyficzny kształt. Nawet przy poprawie szybkości uzyskanej przez FFT o wysokiej skuteczności, czasy wykonania będą paskudne. Przyjrzyjmy się szczegółowo tym trzem sposobom i rozważmy przykłady ich zastosowania w przetwarzaniu obrazów.

Modyfikacja krawędzi 3 × 3

Na rys. 24.4 przedstawiono kilka przykładów operacji 3 × 3. Na rys. 24.4 a) pokazano obraz uzyskany w lotniskowym, rentgenowskim skanerze bagażu. Gdy ten obraz zostanie spleciony z funkcją delta 3 × 3 (*jedynka* w otoczeniu 8 zer), obraz będzie niezmieniony. Chociaż ten fakt sam w sobie nie jest interesujący, to jest on podstawą dla innych odpowiedzi impulsowych filtrów.

Na rys. 24.4 b) przedstawiono ten obraz spleciony z odpowiedzią impulsową filtru 3 × 3 złożoną z jedynki, ujemnej jedynki i z 7 zer. Ta operacja nosi nazwę **przesuwanie i odejmowanie**, gdyż *przesuniętą* wersję obrazu (odpowiadającego –1) *odejmuje się* od obrazu oryginalnego (odpowiadającego 1). Takie przetworzenie daje złudzenie optyczne, że niektóre obiekty są bliżej lub dalej niż tło, tworząc efekt trójwymiarowości lub obrazu wytłaczanego. Mózg interpretuje obrazy tak, jakby światło padało *z góry*, gdyż to jest normalnie spotykany sposób oświetlenia. Jeśli krawędzie obiektu są jasne u góry i ciemne u dołu, obiekt jest postrzegany jako wystający ponad tło. Aby zobaczyć inny interesujący efekt, przekręćcie obraz do góry nogami i obiekty będą wtedy jakby *wepchnięte* w tło.



Rys. 24.4. Modyfikacja 3 × 3. Obraz oryginalny (rys. a) uzyskano z lotniskowego, rentgenowskiego skanera bagażu. Operacja przesuwania i odejmowania (rys. b) daje efekt pseudotrójwymiarowości. Operacja wykrywania krawędzi (rys. c) całkiem usuwa kontrast, pozostawiając tylko informacje o krawędziach. Filtr zaostrzający krawędzie (rys. d) powoduje sumowanie obrazów z rysunków a), c) w różnym ich stosunku, zależnym od stałej k. Przy tworzeniu tego obrazu zastosowano k = 2

Na rys. 24.4 c) pokazano funkcję PSF **detekcji krawędzi** i wynikający z niej obraz. Każda krawędź obrazu oryginalnego jest przekształcana na wąskie ciemne i jasne pasma, biegnące równolegle do oryginalnych krawędzi. Stosując w obrazie odpowiednie progi, można wyizolować albo ciemne, albo jasne pasma, uzyskując prosty algorytm do detekcji krawędzi w obrazie.

Stosowaną na ogół technikę przetwarzania – **wyostrzanie krawędzi**, przedstawiono na rys. 24.4 d). Obiekty na rys. 24.4 a) mają dobry kontrast (odpowiednie poziomy obszarów jasnych i ciemnych), lecz bardzo rozmyte krawędzie. Obiekty na rys. 24.4 c) wcale nie mają kontrastu, lecz mają bardzo ostre krawędzie. Metoda polega na pomnożeniu obrazu z *dobrymi krawędziami* przez pewną stałą k i dodanie go do obrazu z *dobrym kontrastem*. Jest to równoważne

ze splataniem obrazu oryginalnego z funkcją PSF 3 × 3 pokazaną na rys. 24.4 d). Jeśli umieścimy $k = 0$, to PSF staje się funkcją delta i obraz pozostaje bez zmian. Jeśli będzie wybrana większa wartość k, to obraz będzie miał wyraźniej zarysowane krawędzie. W obrazie na rys. 24.4 d) zastosowano $k = 2$, czyli sumowanie w stosunku: dwie części obrazu z rys. 24.4 c) do jednej części obrazu z rys. 24.4 a). Taka operacja jest naśladownictwem zdolności oka do zaostrzania krawędzi, dzięki której obiekty mogą być łatwiej odróżniane od tła.

Splot z którąkolwiek z tych funkcji PSF może spowodować pojawienie się ujemnych pikseli w obrazie końcowym. Nawet jeśli program może posługiwać się ujemnymi wartościami pikseli, to wyświetlanie obrazu nie powinno z nich korzystać. Najprostszym sposobem uniknięcia tej trudności jest dodanie pewnego przesunięcia do każdego z obliczonych pikseli, tak jak w obrazach. Alternatywą jest obcinanie wartości wychodzących poza zakres.

Splot przez rozłączność

Jest to metoda szybkiego splotu, realizowana, gdy funkcja PSF jest **rozłączna**. PSF jest *rozłączna*, jeśli może być rozdzielona na dwa sygnały jednowymiarowe – rzuty pionowy i poziomy. Na rys. 24.5 przedstawiono przykład obrazu rozłącznego – prostopadłościenną funkcję PSF. Ściślej mówiąc, wartość każdego piksela w obrazie jest równa odpowiadającemu mu punktowi w rzucie poziomym pomnożonemu przez odpowiedni punkt w rzucie pionowym. Można to wyrazić wzorem:

$$x[r,c]=vert[r]\times horz[c]$$

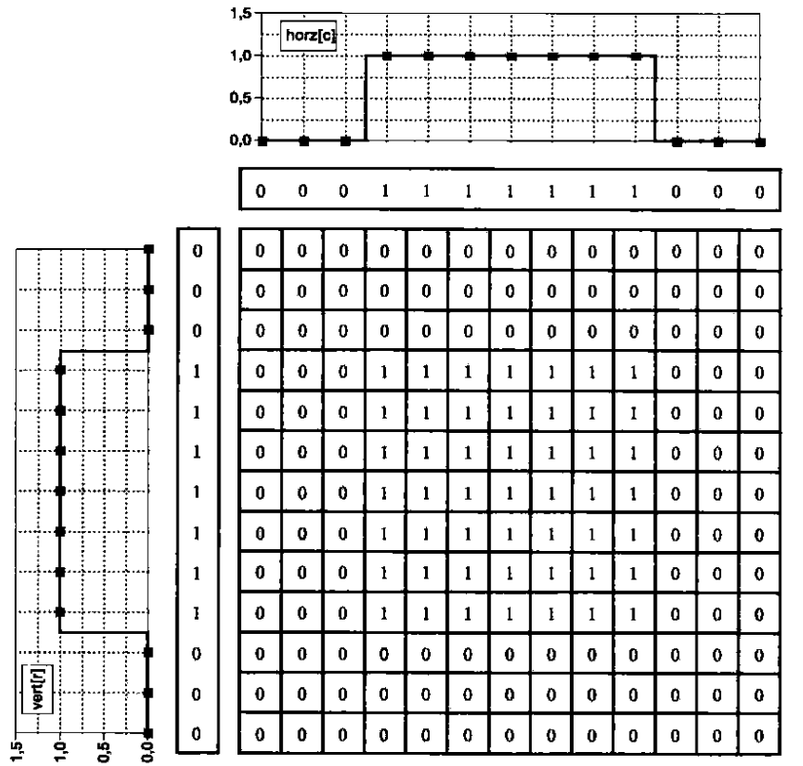
Wzór 24.1. Rozdzielanie obrazu. Obraz traktuje się jako rozłączny, jeśli może zostać rozłożony na rzuty pionowy i poziomy

gdzie $x[r,c]$ jest obrazem dwuwymiarowym, $vert[r]$ i $horz[c]$ są rzutami jednowymiarowymi. Oczywiście, większość obrazów nie spełnia tego wymagania. Na przykład funkcja typu „okrągła czapka" nie jest rozłączna. Istnieje jednak *nieskończona* liczba obrazów rozłącznych. Można to zrozumieć, wytwarzając dowolne rzuty poziome i pionowe i znajdując odpowiadający im obraz. Przedstawiono to na rys. 24.6 na przykładzie profilów będących dwustronnymi funkcjami wykładniczymi. Następnie z wzoru 24.1 jest wyznaczany obraz odpowiadający tym profilom. Po wyświetleniu obraz ma kształt romboidu opadającego wykładniczo w miarę wzrostu odległości od początku układu współrzędnych.

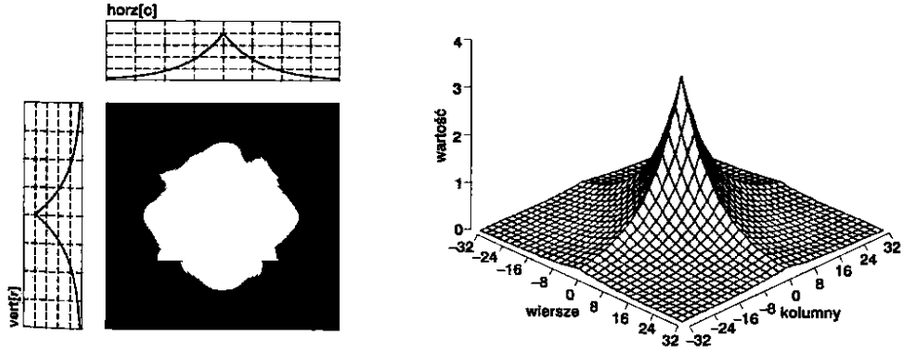
W większości zadań DSP idealna funkcja PSF jest *kołowo symetryczna*, jak np. funkcja typu „okrągła czapka". Chociaż obrazy w postaci cyfrowej są zwykle rejestrowane i przetwarzane w prostokątnym formacie wierszy i kolumn, to pożądane jest modyfikowanie obrazu, który jest jednakowy we wszystkich kierunkach. To rodzi pytanie, czy istnieje funkcja PSF mająca symetrię kołową i *jednocześnie* rozłączna. Odpowiedź brzmi – tak, lecz jest tylko jedna taka funkcja – *gaussowska*. Jak pokazano na rys. 24.7, dwuwymiarowy obraz gaussowski ma rzuty, które też są gaussowskie. Obraz i rzuty gaussowskie mają *takie same* odchylenia standardowe.

W celu wykonania splotu obrazu z rozłączną odpowiedzią impulsową filtru splatamy każdy *wiersz* w obrazie z *rzutem poziomym*, uzyskując obraz pośredni. Następnie splatamy każdą *kolumnę* tego obrazu pośredniego z *rzutem pionowym* funkcji PSF. Uzyskany obraz jest identyczny z bezpośrednim splotem obrazu oryginalnego i odpowiedzi impulsowej filtru. Gdybyście chcieli poddać splotowi najpierw kolumny, a potem wiersze, to wynik będzie taki sam.

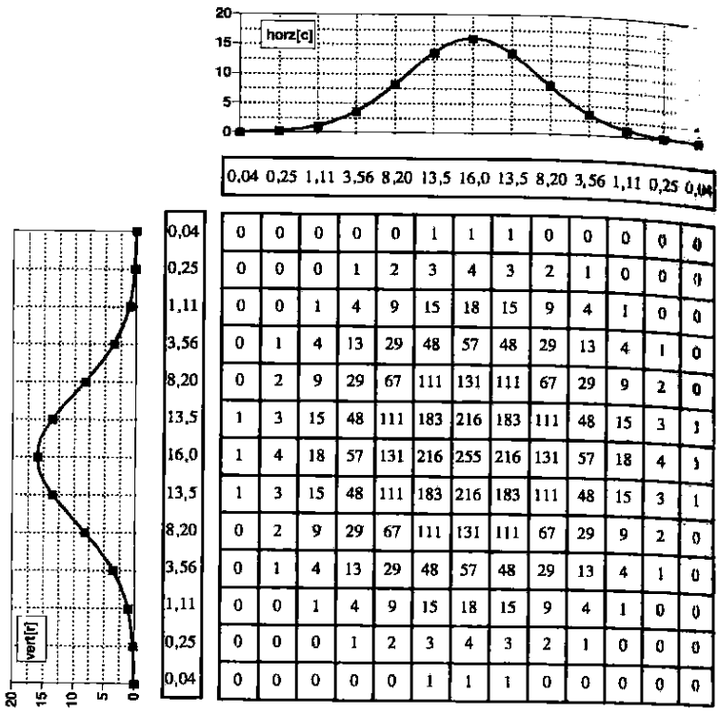
Przeprowadzenie splotu obrazu $N \times N$ z odpowiedzią impulsową $M \times M$ wymaga czasu proporcjonalnego do N^2M^2. Tak więc każdy piksel w obrazie wyjściowym zależy od *wszystkich* pikseli w odpowiedzi impulsowej filtru. Natomiast splot przez rozłączność wymaga tylko czasu proporcjonalnego do N^2M. Dla odpowiedzi impulsowych filtru o długościach setek pikseli ta metoda redukuje czas wykonania o czynnik rzędu *setek*.



Rys. 24.5. Rozłączność prostopadłościennej funkcji PSF. PSF jest rozłączna, jeśli może być rozdzielona na profile poziomy i pionowy. Rozłączne funkcje PSF mają istotne znaczenie, gdyż mogą być poddawane bardzo szybkiej operacji splotu



Rys. 24.6. Tworzenie rozłącznej funkcji PSF. Można utworzyć nieskończoną liczbę rozłącznych funkcji PSF, definiując dowolne rzuty i obliczając następnie odpowiadającą im funkcję dwuwymiarową. W tym przykładzie jako profile wybrano dwustronne funkcje wykładnicze, uzyskując PSF o kształcie romboidalnym



0,04	0	0	0	0	0	1	1	1	0	0	0	0	0
0,25	0	0	0	1	2	3	4	3	2	1	0	0	0
1,11	0	0	1	4	9	15	18	15	9	4	1	0	0
3,56	0	1	4	13	29	48	57	48	29	13	4	1	0
8,20	0	2	9	29	67	111	131	111	67	29	9	2	0
13,5	1	3	15	48	111	183	216	183	111	48	15	3	1
16,0	1	4	18	57	131	216	255	216	131	57	18	4	1
13,5	1	3	15	48	111	183	216	183	111	48	15	3	1
8,20	0	2	9	29	67	111	131	111	67	29	9	2	0
3,56	0	1	4	13	29	48	57	48	29	13	4	1	0
1,11	0	0	1	4	9	15	18	15	9	4	1	0	0
0,25	0	0	0	1	2	3	4	3	2	1	0	0	0
0,04	0	0	0	0	0	1	1	1	0	0	0	0	0

Rys. 24.7. Rozłączność funkcji gaussowskiej. Funkcja gaussowska jest jedyną funkcją PSF, która jest kołowo symetryczna oraz rozłączna. Dlatego jest często stosowana w przetwarzaniu obrazów jako odpowiedź impulsowa filtru

A może być jeszcze lepiej. Jeśli chcecie zastosować *prostopadłościenną* (rys. 24.5) lub *dwustronnie wykładniczą* funkcję PSF (rys. 24.6), to obliczenia są jeszcze sprawniejsze. Wynika to z faktu, że sploty jednowymiarowe są, odpowiednio, filtrem *o średniej kroczącej* (rozdział 15) i *dwukierunkowym filtrem jednobiegunowym* (rozdział 19). Oba te filtry jednowymiarowe mogą być bardzo szybko realizowane przez rekursję. W rezultacie uzyskuje się czas wykonania proporcjonalny tylko do N^2, całkowicie niezależny od rozmiaru PSF. Inaczej mówiąc, obraz może być splatany zgodnie z potrzebami, z dowolnie długą funkcją PSF, z kilkoma tylko operacjami na liczbach całkowitych na piksel. Na przykład splot obrazu 512 × 512 wymaga tylko kilkuset milisekund na komputerze osobistym. To bardzo szybko! Nie podoba się wam kształt tych dwóch odpowiedzi impulsowych filtru? Wykonajcie splot obrazu z jedną z tych odpowiedzi *kilkakrotnie*, aby aproksymować gaussowską funkcję PSF (zgodnie z centralnym twierdzeniem granicznym, rozdział 7). To są świetne algorytmy, które można z sukcesem realizować. Warto o nich pamiętać.

Przykład dużej funkcji PSF – spłaszczanie oświetlenia

Częstym zastosowaniem wymagającym dużej funkcji PSF jest ulepszanie obrazów o nierównym oświetleniu. Splot przez rozłączność jest algorytmem idealnie nadającym się do przeprowadzenia takiego przetwarzania. Z kilkoma zaledwie wyjątkami obrazy widziane przez oko są tworzone ze światła *odbitego*. Wynika stąd, że widziany obraz jest proporcjonalny do współczynnika odbicia pomnożonego przez oświetlenie pochodzące z otoczenia. Model tworzenia w ten sposób obrazu przedstawiono na rys. 24.8. Na rys. 24.8 a) pokazano rozkład

współczynnika odbicia w oglądanym widoku; w tym przypadku jest to szereg pasm jasnych i ciemnych. Przykład sygnału oświetlenia pokazano na rys. 24.8 b). Jest to rozkład światła padającego na obiekt z rys. 24.8 a). Jak w świecie rzeczywistym, oświetlenie w całym obszarze widoku nie jest jednakowe. Na rys. 24.8 c) widzimy obraz widziany przez oko, równy rozkładowi współczynnika odbicia (rys. 24.8 a) pomnożonemu przez rozkład oświetlenia (rys. 24.8 b). Obszary słabo oświetlone w widoku na rys. 24.8 c) są trudne do zobaczenia, gdyż są zbyt ciemne i mają zbyt słaby kontrast (różnicę między maksimami i minimami).

W celu zrozumienia, jakie ma to odniesienie do zagadnienia codziennego widzenia, wyobraźcie sobie, że patrzycie na dwóch identycznie ubranych mężczyzn. Jeden z nich stoi w jasnym świetle słonecznym, a drugi w cieniu pobliskiego drzewa. Procent światła padającego odbity od obu tych ludzi jest taki sam. Ich twarze mogą odbijać np. 80% padającego światła, ich szare koszule 40%, a ciemne spodnie 5%. Jednak ich oświetlenie może różnić się, powiedzmy, dziesięciokrotnie. To powoduje, że obraz mężczyzny w cieniu jest dziesięć razy ciemniejszy niż tego w słońcu, a i kontrast (między twarzą, koszulą i spodniami) jest dziesięciokrotnie mniejszy.

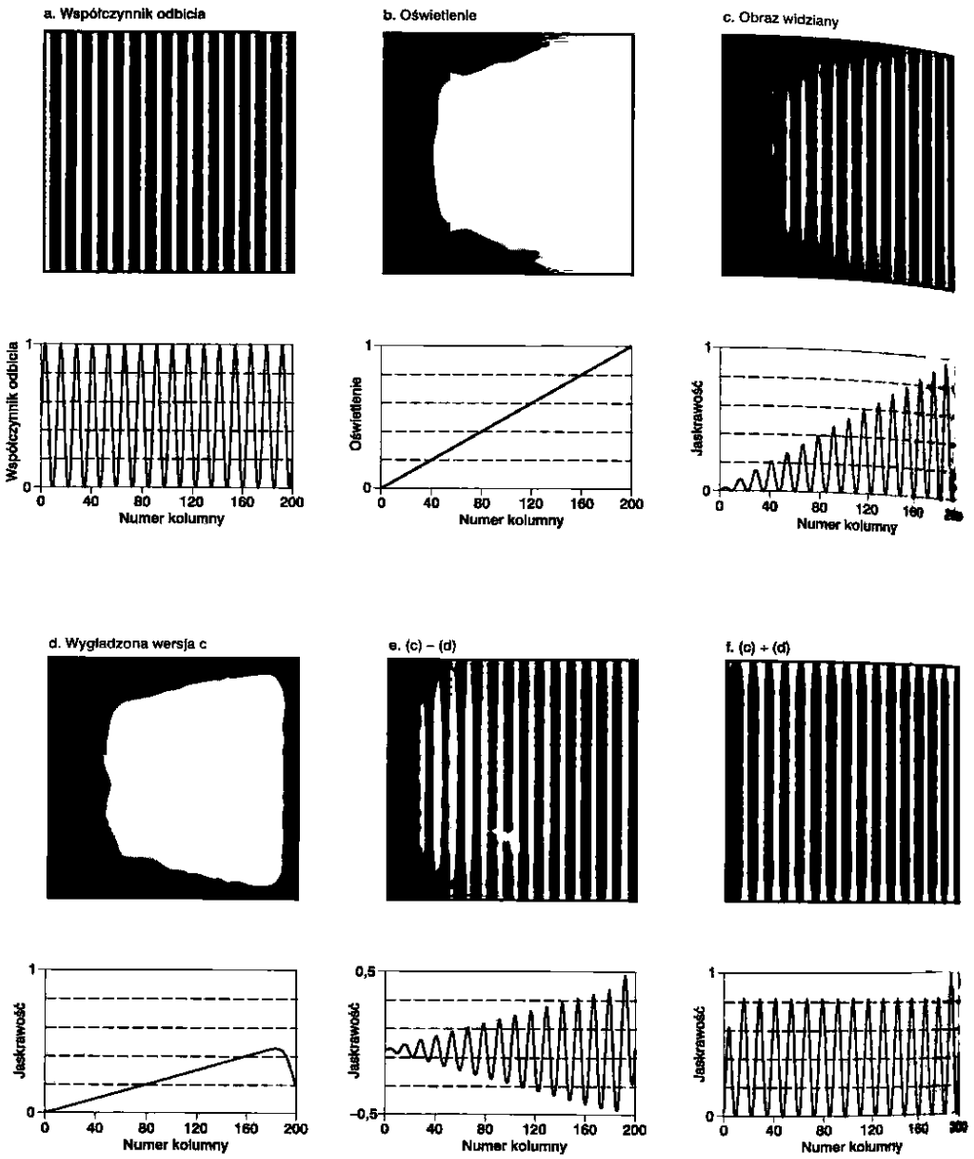
Celem przetwarzania obrazu jest w tym przypadku *spłaszczenie* oświetlenia poszczególnych elementów pobieranego obrazu. Inaczej mówiąc, chcielibyśmy, aby obraz końcowy był reprezentacją współczynników odbicia, a nie warunków oświetlenia. Można powiedzieć, że mając obraz z rys. 24.8 c), chcemy uzyskać obraz z rys. 24.8 a). Jest to zagadnienie filtracji nieliniowej, gdyż składowe obrazu są uzyskiwane przez mnożenie, a nie przez dodawanie. Rozdzielenia nie da się przeprowadzić idealnie, można jednak uzyskać bardzo znaczną poprawę jakości obrazu.

Rozpoczniemy od wykonania splotu obrazu z rys. 24.8 c) z dużą funkcją PSF, o rozmiarze równym jednej piątej całego obrazu. Celem jest wyeliminowanie ostrych zmian na rys. 24.8 c) i uzyskanie przybliżenia oryginalnego sygnału oświetlenia z rys. 24.8 b). Tu właśnie stosuje się splot przez rozłączność. Dokładny kształt PSF nie jest ważny, funkcja powinna tylko być znacznie większa niż szczegóły w obrazie odbiciowym. W wyniku, stosując gaussowską odpowiedź filtru, uzyskuje się obraz pokazany na rys. 24.8 d).

Ponieważ filtr wygładzający daje przybliżenie obrazu wynikającego z oświetlenia, więc użyjemy filtru zaostrzającego krawędzie w celu uzyskania obrazu odbiciowego. A zatem obraz z rys. 24.8 c) trzeba poddać splotowi z odpowiedzią impulsową filtru będącą różnicą funkcji delta i gaussowskiej. Aby zmniejszyć czas wykonania, robi się to przez odejmowanie wygładzonego obrazu z rys. 24.8 d) od obrazu oryginalnego z rys. 24.8 c). Wynik przedstawiono na rys. 24.8 e). To nie działa dobrze! Wprawdzie ciemne obszary zostały prawidłowo oświetlone, ale kontrast w tych obszarach jest nadal bardzo zły.

Filtracja liniowa w tym zastosowaniu działa słabo, gdyż sygnały odbicia i oświetlenia były pierwotnie połączone przez mnożenie, a nie przez dodawanie. Przez filtrację liniową nie można rozdzielić sygnałów, które zostały ze sobą połączone w wyniku operacji nieliniowej. W celu rozdzielenia takich sygnałów musi być wykonana operacja *odwrotna do mnożenia*. Trzeba więc obraz oryginalny *podzielić* przez obraz wygładzony, jak to pokazano na rys. 24.8 f). W ten sposób koryguje się jaskrawość i odtwarza odpowiedni poziom kontrastu.

Taka procedura dzielenie obrazów przez siebie jest ściśle związana z **przetwarzaniem homomorficznym**, opisanym już w rozdziale 22. Przetwarzanie homomorficzne jest sposobem operowania sygnałami, które zostały ze sobą połączone przez operację nieliniową. Podejściem do tej kwestii jest zamiana problemu nieliniowego w liniowy, przez odpowiednią operację matematyczną. Gdy dwa sygnały są ze sobą połączone przez mnożenie, to przetwarzanie homomorficzne trzeba rozpocząć od *zlogarytmowania* pobranego sygnału. Przez tożsamość $\log(a \times b) = \log a + \log b$, problem rozdzielenia sygnałów powstałych przez *mnożenie* zamienia się w problem rozdzielenia sygnałów powstałych przez *dodawanie*. Na przykład, logarytmując obraz z rys. 24.8 c), można zastosować liniowy filtr górnoprzepustowy do rozdzielenia logarytmu od obrazu odbiciowego. Podobnie jak poprzednio, najszybszym sposobem realizacji filtru górnoprzepustowego jest odjęcie wygładzonej wersji obrazu. Następnie stosuje się antylogarytm (a więc funkcję wykładniczą), uzyskując żądane przybliżenie obrazu odbiciowego.



Rys. 24.8. Model tworzenia obrazu. Obraz widziany (rys. c) powstaje w wyniku wymnożenia rozkładu oświetlenia (rys. b) przez rozkład współczynników odbicia (rys. a). Celem przetwarzania jest takie zmodyfikowanie obrazu c), żeby uczynić go bardziej podobnym do obrazu a). Jest to realizowane na rys. d), e), f). Obraz na rys. d) jest wygładzoną wersją obrazu z rys. c), stosowaną jako aproksymacja sygnału oświetlenia. Na rys. e) pokazano aproksymację obrazu odbiciowego, utworzoną przez odjęcie obrazu wygładzonego od obrazu widzianego. Na rys. f) przedstawiono lepszą aproksymację uzyskaną w wyniku nieliniowego procesu dzielenia przez siebie tych dwóch obrazów

Co jest lepsze – dzielenie czy skorzystanie z drogi przetwarzania homomorficznego? Obie metody są prawie identyczne, ponieważ logarytmowanie i odejmowanie są *równoważne* dzieleniu. Jedyną różnicą jest aproksymacja zastosowana do obrazu uzyskanego na podstawie oświetlenia. W jednej metodzie stosuje się wygładzoną wersję pobranego obrazu, a w drugiej – wygładzoną wersję logarytmu pobranego obrazu.

Ta technika spłaszczania sygnału oświetlenia jest tak użyteczna, że jest włączona do neuronowej struktury oka. Przetwarzanie zachodzące w środkowej warstwie siatkówki opisano poprzednio jako wyostrzanie krawędzi lub jako filtr górnoprzepustowy. Choć jest to prawda, to jednak nie cała. Pierwsza warstwa oka działa w sposób nieliniowy; dokonuje – w przybliżeniu – *logarytmowania* pobieranego obrazu. To czyni z oka procesor homomorficzny. Jak już powiedziano, logarytmowanie i następnie filtracja zaostrzająca krawędzie powodują spłaszczanie składowej oświetleniowej, co umożliwia oku widzenie w warunkach złego oświetlenia. Inne interesujące zastosowanie przetwarzania homomorficznego pojawia się w fotografii. Gęstość (naświetlenie) negatywu jest równe logarytmowi jaskrawości w końcowym zdjęciu. To oznacza, że wszystkie operacje z negatywem podczas jego wywoływania są pewnym rodzajem przetwarzania homomorficznego.

Przed zakończeniem omawiania tego przykładu trzeba wspomnieć o pewnej niedogodności. Jak już stwierdzono w rozdziale 6, gdy N-punktowy sygnał jest splatany z M-punktową odpowiedzią impulsową filtru, to uzyskuje się sygnał o długości $(N + M - 1)$ punktów. Podobnie, gdy obraz $M \times M$ jest splatany z odpowiedzią impulsową $N \times N$, to wynikiem jest obraz $(M + N - 1) \times (M + N - 1)$. Kłopot z tym, że czasem trudno sobie poradzić ze zmieniającymi się rozmiarami obrazu. Musi być zmieniana np. przydzielona pojemność pamięci, dobierany wyświetlacz, ewentualnie zmieniane indeksowanie matrycy itd. Zazwyczaj te kwestie po prostu *się pomija*. Jeśli zaczynamy od obrazu 512 × 512, to chcemy zakończyć też na obrazie 512 × 512. Piksele niepasujące do pierwotnych granic są odrzucane.

W ten sposób utrzymuje się wprawdzie takie same rozmiary obrazu, lecz nie rozwiązuje się całego problemu. Ciągle pozostają *warunki graniczne* dla splotu. Wyobraźmy sobie, na przykład, że próbujemy obliczyć wartość piksela w prawym, górnym rogu w obrazie na rys. 24.8 d). Wykonuje się to przez wypośrodkowanie gaussowskiej funkcji PSF na prawym, górnym rogu obrazu na rys. 24.8 c). Każdy piksel obrazu na rys. 24.8 c) jest następnie wymnażany przez odpowiadający mu piksel w nakładanej PSF i iloczyny są sumowane. Trudność polega na tym, że trzy czwarte PSF leży na zewnątrz zdefiniowanego obrazu. Najłatwiejszym rozwiązaniem jest przypisanie niezdefiniowanym pikselom wartości zero. Tak właśnie utworzono obraz na rys. 24.8 d), co wyjaśnia pochodzenie ciemnego paska na obwodzie obrazu. Tak więc jaskrawość łagodnie maleje aż do zerowej wartości piksela na zewnątrz zdefiniowanego obrazu.

Na szczęście ten ciemny obszar wokół granicy można skorygować, choć nie zrobiono tego w podanym przykładzie. Korekcję wprowadza się, dzieląc każdy piksel na obrazie z rys. 24.8 d) przez współczynnik korygujący. Ten współczynnik jest takim ułamkiem funkcji PSF, jaki był wprowadzony do obrazu, gdy obliczano wartość piksela. Tak więc aby skorygować pojedynczy piksel w obrazie z rys. 24.8 d), wyobraźmy sobie, że funkcja PSF jest wypośrodkowana na odpowiednim pikselu na rys. 24.8 c). Na przykład górny, prawy piksel na rys. 24.8 c) jest tylko w 25% wynikiem funkcji PSF nakładanej na obraz wejściowy. A zatem korygujemy ten piksel na obrazie z rys. 24.8 d), dzieląc go przez czynnik 0,25. Z tego wynika, że piksele w środku obrazu na rys. 24.8 d) pozostaną niezmienione, lecz ciemne piksele na obwodzie zostaną rozjaśnione. Aby wyznaczyć współczynniki korygujące, wyobraźmy sobie splatanie odpowiedzi impulsowej filtru z obrazem, w którym wszystkie piksele mają wartość równą *jeden*. W uzyskanym w ten sposób obrazie piksele mają wartości równe współczynnikom korygującym, koniecznym do usunięcia zjawisk krawędziowych.

Fourierowska analiza obrazu

W przetwarzaniu obrazów analiza fourierowska jest stosowana w podobny sposób, jak w przypadku sygnałów jednowymiarowych. Jednak obrazy nie mają informacji zakodowanych w dziedzinie częstotliwości, co czyni tę analizę mniej użyteczną. Na przykład realizując przekształcenie Fouriera sygnału *audio*, zamienia się trudny do interpretacji przebieg czasowy na łatwiejsze do zrozumienia widmo częstotliwościowe. Wykonując zaś przekształcenie Fouriera obrazu, zamienia się jasną informację w dziedzinie przestrzeni na zagmatwaną w dziedzinie częstotliwości. Krótko mówiąc, nie spodziewajcie się, że przekształcenie Fouriera pomoże wam w zrozumieniu informacji zakodowanych w obrazach.

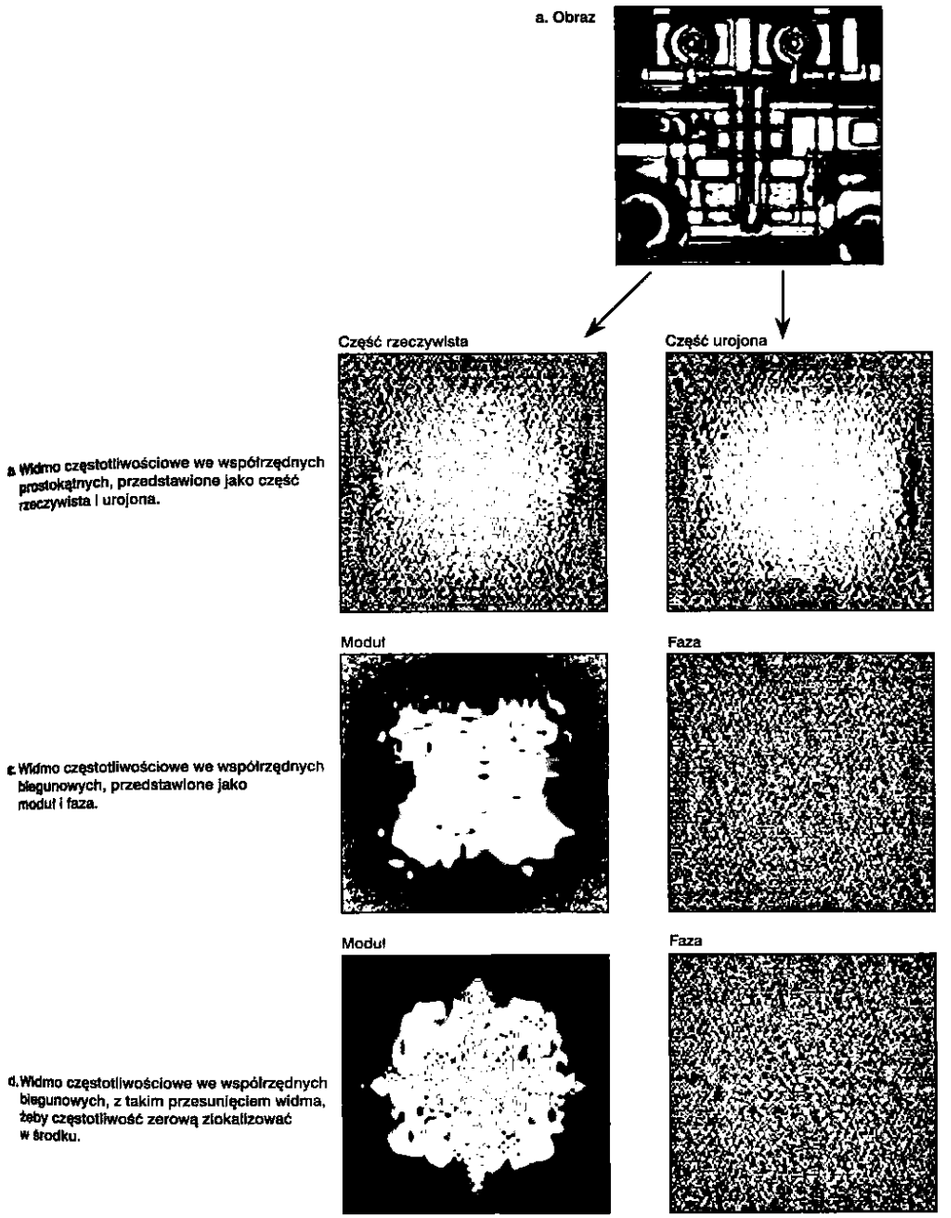
Nie zaglądajcie też do dziedziny częstotliwości przy projektowaniu filtrów do przetwarzania obrazów. Najważniejszą cechą w obrazie jest krawędź, linia oddzielająca jeden *obiekt* lub *obszar* od drugiego *obiektu* lub *obszaru*. Próby modyfikacji krawędzi przez operowanie widmem częstotliwościowym są na ogół mało skuteczne, gdyż krawędź zawiera bardzo szeroki zakres częstotliwości składowych. Filtry obrazowe są zwykle projektowane w dziedzinie przestrzeni, gdzie informacja jest zakodowana w najprostszej formie. Trzeba tu myśleć raczej w kategoriach *wygładzania* i *wyostrzania krawędzi*, a nie filtrów górno- i dolnoprzepustowych (w dziedzinie częstotliwości).

Mimo to fourierowska analiza obrazów ma kilka pożytecznych właściwości. Na przykład *splot* w dziedzinie przestrzeni odpowiada *mnożeniu* w dziedzinie częstotliwości. Jest to ważne, gdyż mnożenie jest operacją matematyczną prostszą niż splot. Podobnie jak dla sygnałów jednowymiarowych, ta właściwość umożliwia splot FFT i różne metody rozplotu. Inną korzystną cechą dziedziny częstotliwości jest twierdzenie Fouriera o projekcji (*Fourier Slice Theorem*) wyrażające zależność między obrazem i jego rzutami (obrazem widzianym z jego boków). Jest ono podstawą *tomografii komputerowej*, rentgenowskiej techniki obrazowej szeroko stosowanej w medycynie i przemyśle.

Widmo częstotliwościowe obrazu można obliczać kilkoma różnymi sposobami, ale znaczenie praktyczne ma tylko przedstawiona tutaj metoda FFT. Obraz oryginalny powinien być złożony z N wierszy i N kolumn, gdzie N jest potęgą 2, tj. 256, 512, 1024 itd. Jeśli wymiar obrazu oryginalnego nie jest potęgą 2, to dodaje się piksele o wartości zero, aby skorygować ten wymiar. Dwuwymiarową matrycę zawierającą obraz będziemy nazywać **matrycą rzeczywistą**. Dodatkowo potrzebna jest jeszcze druga matryca o takich samych wymiarach, którą będziemy nazywać **matrycą urojoną**.

Recepta na obliczanie transformaty Fouriera obrazu jest całkiem prosta: wyznacza się jednowymiarowe FFT każdego wiersza, a następnie – każdej kolumny. Zacznijcie od wyznaczenia transformaty FFT dla wartości N pikseli w wierszu 0 matrycy rzeczywistej. Część rzeczywista uzyskanej transformaty FFT jest z powrotem umieszczana w wierszu 0 matrycy rzeczywistej, podczas gdy część urojona tej transformaty jest umieszczana w wierszu matrycy urojonej. Po powtórzeniu tej procedury dla wierszy od 1 do $N-1$, zarówno matryca rzeczywista, jak i urojona zawierają obraz pośredni. Potem procedurę trzeba powtórzyć dla każdej *kolumny* danych pośrednich. Weźcie N wartości pikseli z kolumny 0 matrycy rzeczywistej *oraz* N wartości pikseli z kolumny 0 matrycy urojonej i obliczcie FFT. Część rzeczywista uzyskanej transformaty FFT jest umieszczana z powrotem w kolumnie 0 matrycy rzeczywistej, a część urojona – z powrotem w kolumnie 0 matrycy urojonej. Po powtórzeniu tych operacji na kolumnach od 1 do $N-1$ do obu matryc zostało wpisane widmo częstotliwościowe obrazu.

Ponieważ w obrazie kierunki poziomy i pionowy są sobie równoważne, więc ten algorytm można także realizować, poddając przekształceniu najpierw kolumny, a potem wiersze. Niezależnie od przyjętej kolejności, wynik jest taki sam. Ze względu na drogę śledzenia danych przez FFT, amplitudy składowych małoczęstotliwościowych znajdują swoje miejsce w narożnikach widma dwuwymiarowego, a duże częstotliwości – w jego środku. Odwrotna transformata Fouriera obrazu jest obliczana przez wyznaczenie odwrotnego FFT każdego wiersza i następnie każdej kolumny (lub vice versa).



Rys. 24.9. Częstotliwościowe widmo obrazu. Jako przykład pokazano na rys. a) obraz mikroskopowy krzemowej powierzchni układu scalonego. Widmo częstotliwościowe można przedstawić jako część rzeczywistą i urojoną (rys. b) lub jako moduł i fazę (rys. c). Na rysunkach b), c) małe częstotliwości pokazano w narożnikach, a duże – pośrodku. Ponieważ sygnały w dziedzinie są okresowe, więc sposób prezentacji można zmienić, odwracając usytuowanie małych i dużych częstotliwości. Pokazano to na rys. d), gdzie moduł i faza są przedstawione w taki sposób, że małe częstotliwości są zlokalizowane pośrodku, a duże – w narożnikach

Na rys. 24.9 przedstawiono przykład obrazu poddanego przekształceniu Fouriera. Na rys. 24.9 a) pokazano obraz oryginalny – mikroskopowy widok stopnia wejściowego scalonego wzmacniacza operacyjnego typu 741, a na rys. 24.9 b) – części rzeczywistą i urojoną widma częstotliwościowego obrazu. Ponieważ w dziedzinie częstotliwości mogą występować piksele o wartościach ujemnych, więc wartości skali szarości tych obrazów są przesunięte w taki sposób, że wartości ujemne są ciemne, zero jest szare, a wartości dodatnie są jasne. W obu razie amplitudy składowe małoczęstotliwościowe są zwykle znacznie większe niż składowe o dużych częstotliwościach. To wyjaśnia istnienie bardzo jasnych i bardzo ciemnych pikseli w czterech rogach obrazu na rys. 24.9 b). Natomiast widma *typowych* obrazów nie mają żadnego dostrzegalnego porządku; ich charakter wydaje się losowy. Oczywiście można wam *wymyślać* obrazy o dowolnym, pożądanym widmie.

Jak wynika z rys. 24.9 c), widmo obrazu w postaci biegunowej jest tylko trochę łatwiejsze do zrozumienia. Moduły składowych małoczęstotliwościowych mają duże wartości dodatnie (białe narożniki), a składowych o dużych częstotliwościach – małe wartości dodatnie (czarny środek). Charakterystyka fazy wygląda jednakowo przy małych i dużych częstotliwościach i jest losowo rozłożona między wartościami $-\pi$ i π radianów.

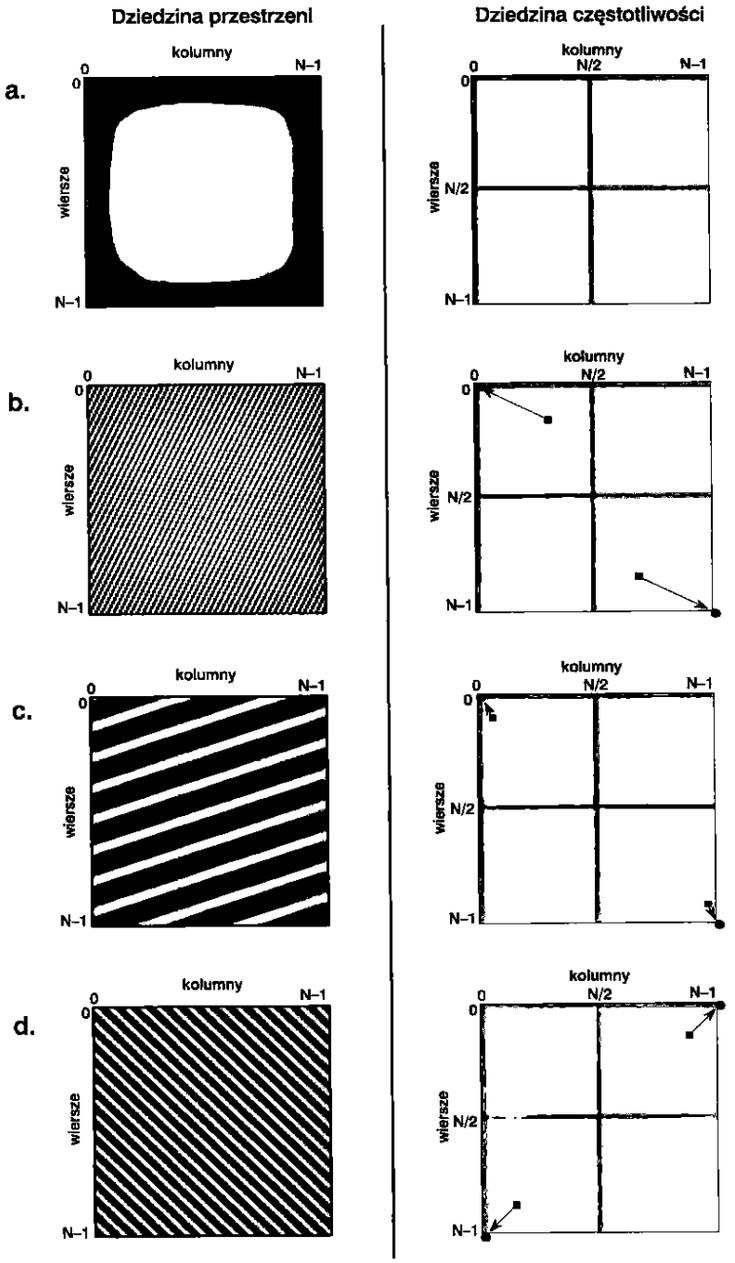
Na rys. 24.9 d) przedstawiono alternatywny sposób prezentacji widma obrazu. Ponieważ w dziedzinie przestrzeni są zawarte sygnały *dyskretne*, więc w dziedzinie częstotliwości są one *okresowe*. Innymi słowy, matryce w dziedzinie częstotliwości są powtarzane nieskończoną liczbę razy w lewo, w prawo, w górę i w dół. Jako przykład weźmy ścianę wyłożoną kafelkami, w której każdy kafelek ma wymiary $N \times N$ (jak na rys. 24.9 c). Na rys. 24.9 d) też jest fragment $N \times N$ takiej ściany, lecz obejmuje on fragmenty czterech kafelków w taki sposób, że środek obrazu jest punktem styku tych czterech kafelków. Tak więc obraz na rys. 24.9 d) jest taki, jak na rys. 24.9 c), z tą wszakże różnicą, że został przesunięty w poziomie o $N/2$ pikseli (w prawo lub w lewo) i w pionie też o $N/2$ pikseli (w górę lub w dół) w okresowym widmie częstotliwościowym. W ten sposób jasne piksele z czterech narożników obrazu na rys. 24.9 c) są umieszczone obok siebie pośrodku obrazu na rys. 24.9 d).

Na rys. 24.10 przedstawiono sposób organizacji dwuwymiarowego widma częstotliwościowego (z małymi częstotliwościami umieszczonymi w narożnikach). Wiersz $N/2$ i kolumna $N/2$ dzielą to widmo na cztery ćwiartki. Pod względem części rzeczywistej i modułu górna prawa ćwiartka jest lustrzanym odbiciem dolnej lewej, a górna lewa - dolnej prawej. Ta symetria jest także zachowana dla części urojonej i fazy, tyle tylko że wartości pikseli będących odbiciem lustrzanym mają odwrócony znak. Zatem każdy punkt w widmie częstotliwościowym ma odpowiadający mu punkt umieszczony symetrycznie z drugiej strony środka obrazu (wierszu $N/2$, kolumna $N/2$). Jeden z tych punktów to częstotliwość *dodatnia*, podczas gdy drugi – odpowiadająca jej częstotliwość *ujemna*, jak to omówiono w rozdziale 10 dla sygnałów jednowymiarowych. Matematycznie ta symetria wyraża się wzorem:

$$Re\,X[r,c] = Re\,X[N-r, N-c]$$
$$Im\,X[r,c] = -Im\,X[N-r, N-c]$$

Wzór 24.2. Symetria widma dwuwymiarowego w dziedzinie częstotliwości. Wzory można stosować dla obu formatów – zarówno gdy małe częstotliwości są prezentowane w narożnikach, jak i wówczas, gdy w wyniku przesunięcia zostały umieszczone pośrodku. We współrzędnych biegunowych moduł ma symetrię taką samą, jak część rzeczywista, faza zaś – taką jak część urojona

W powyższych wzorach uwzględniono okresowość widma częstotliwościowego, które powtarza się co każde N próbek z indeksami zmieniającymi się od 0 do $N-1$. Dlatego $X[r,N]$ powinno być interpretowane jako $X[r,0]$, $X[N,c]$ jako $X[0,c]$ oraz $X[N,N]$ jako $X[0,0]$. Ta symetria powoduje, że cztery punkty są *swymi własnymi* odpowiednikami. Te punkty mają następujące lokalizacje: [0, 0], [0, $N/2$], [$N/2$, 0] i [$N/2$, $N/2$].



Rys. 24.10. Sinusoidy dwuwymiarowe. Dwuwymiarowe fale sinusoidalne i kosinusoidalne mają zarówno częstotliwość, jak i kierunek. Tutaj pokazano cztery przykłady. Te widma są prezentowane w taki sposób, że małe częstotliwości są w narożnikach. Kółeczka na widmach wskazują położenie częstotliwości zerowej

Każda para punktów w dziedzinie częstotliwości odpowiada jednej sinusoidzie w dziedzinie przestrzeni. Wartość dla [0, 0] odpowiada sinusoidzie o *częstotliwości zerowej* w dziedzinie przestrzeni (rys. 24.10 a), tzn. składowej stałej w obrazie. Na rysunku pokazano tylko jeden punkt, gdyż jest to jeden z tych punktów, które są odpowiednikami same dla siebie. Na rysunkach 24.10 b), c), d) inne pary punktów są odpowiednikami sinusoid dwuwymiarowych, które wyglądają jak fale na oceanie. Sinusoidy jednowymiarowe charakteryzują się częstotliwością, *fazą* i *amplitudą*. Sinusoidy dwuwymiarowe charakteryzują się jeszcze *kierunkiem*.

Częstotliwość i kierunek każdej sinusoidy jest wyznaczony lokalizacją pary punktów w dziedzinie częstotliwości. Jak pokazano, trzeba przeprowadzić linię od każdego punktu do *częstotliwości zerowej* umiejscowionej w zewnętrznym rogu tej ćwiartki, w której znajduje się punkt. Są to więc lokalizacje [0, 0], [0, *N*/2], [*N*/2, 0] lub [*N*/2, *N*/2], zaznaczone kółkami na rysunku. Kierunek tej linii określa kierunek przestrzenny sinusoidy, a jej długość jest proporcjonalna do częstotliwości fali. W rezultacie małe częstotliwości są lokalizowane blisko rogów, a duże – w pobliżu środka.

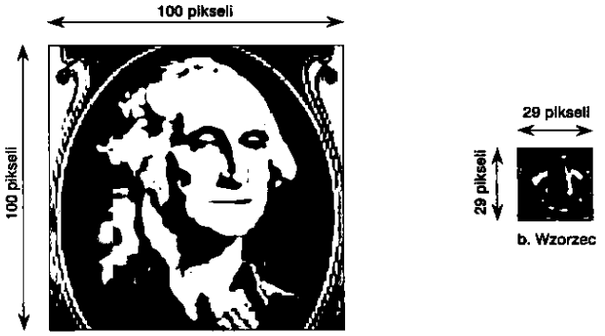
Gdy widmo jest prezentowane z częstotliwością zerową w środku (rys. 24.9 d), wówczas linia od każdej pary punktów jest prowadzona do wartości odpowiadającej przebiegowi stałemu (częstotliwości zerowej) w *środku* obrazu, tj. w punkcie [*N*/2, *N*/2]. Obraz prezentowany w ten sposób jest łatwiejszy do zrozumienia i obróbki, ponieważ wszystkie linie są przeprowadzane do tego samego punktu. Inną zaletą umieszczenia zera w środku jest dopasowanie takiej prezentacji do widm częstotliwościowych obrazów *ciągłych*. Gdy sygnał w dziedzinie przestrzeni jest ciągły, to w dziedzinie częstotliwości jest *nieokresowy*. To lokalizuje częstotliwość zerową w środku, przy czym częstotliwość wzrasta we wszystkich kierunkach aż do nieskończoności. Ogólnie biorąc, widma częstotliwościowe obrazów dyskretnych są prezentowane z częstotliwością zerową w środku, ilekroć ogląda się te dane w podręcznikach, artykułach i dokumentacjach algorytmów. Jednak większość obliczeń prowadzi się, rejestrując dane w komputerze, w matrycach w innym formacie (małe częstotliwości w narożnikach). Jest to bowiem format stosowany w przekształceniu FFT.

Nawet przy zastosowaniu FFT, czas wymagany do obliczenia przekształcenia Fouriera jest bardzo kłopotliwym wąskim gardłem w przetwarzaniu obrazów. Na przykład przekształcenie Fouriera obrazu 512 × 512 wymaga kilku minut pracy komputera osobistego. Jest to szybkość ok. 10 000 razy mniejsza niż wymagana do przetwarzania obrazów w czasie rzeczywistym, 30 obrazów/s. Ten długi czas wykonywania jest spowodowany ogromną ilością informacji zawartą w obrazach. Dla porównania, w typowym obrazie jest mniej więcej tyle samo *pikseli*, ile *słów* w tej książce. Przetwarzanie obrazów przez dziedzinę częstotliwości stanie się bardziej popularne, gdy komputery staną się szybsze. To jest technika XXI wieku: na pewno wkrótce się rozwinie.

Splot FFT

Chociaż przekształcenie Fouriera jest wolne, to jest jednak najszybszym sposobem przeprowadzenia splotu obrazu z dużą odpowiedzią impulsową filtru. Na przykład wykonanie splotu obrazu 512 × 512 z funkcją PSF 50 × 50 jest przy użyciu FFT ok. 20 razy szybsze niż realizacja splotu konwencjonalnego. W rozdziale 18 rozważano splot FFT dla sygnałów jednowymiarowych. Wersja dwuwymiarowa jest tylko prostym rozszerzeniem.

Splot FFT zademonstrujemy na przykładzie algorytmu, który na obrazie ma odszukać wcześniej ustalony wzorzec. Przyjmijmy, że tworzymy system do sprawdzania banknotów jednodolarowych, który mógłby być stosowany np. do kontroli jakości druku, wykrywania fałszerstw lub weryfikacji płatności w sprzedaży z automatu. Z banknotu pobiera się obraz 100 × 100 pikseli wypośrodkowany na portrecie Jerzego Waszyngtona (rys. 24.11). Celem jest odszukanie na obrazie znanego wzorca, którym w tym przykładzie jest obraz twarzy o rozmiarach 29 × 29 pikseli. Mając pobrany obraz i znany wzorzec, trzeba się zastanowić



a. Obraz przeszukiwany

Rys. 24.11. Przykład odszukiwania wzorca. Celem jest przeszukanie obrazu 100 × 100 pikseli przedstawiającego Jerzego Waszyngtona (rys. a) w celu wykrycia wzorca docelowego, którym jest obraz twarzy 29 × 29 pikseli (rys. b). Rozwiązaniem optymalnym tego problemu jest korelacja, która jest przeprowadzana przez splot

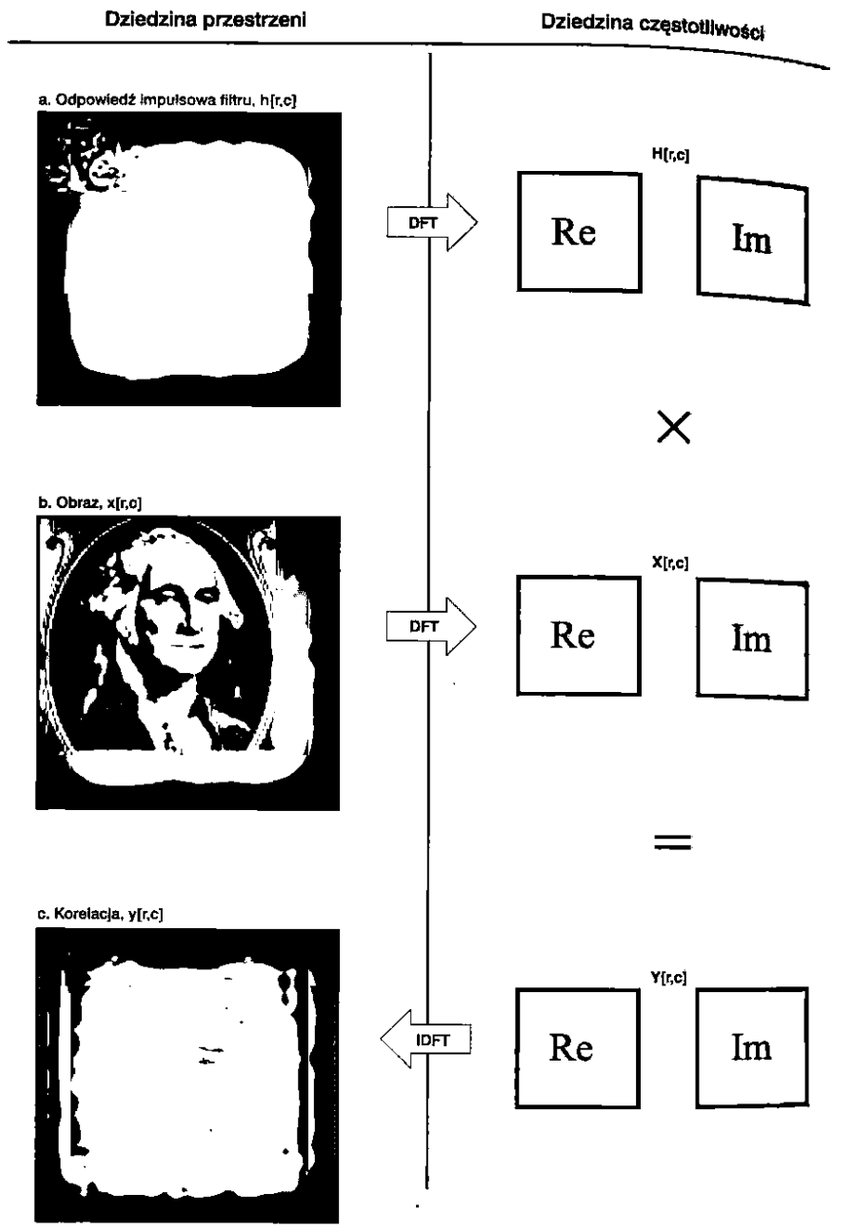
jaki jest najbardziej efektywny sposób stwierdzenia, czy i gdzie na obrazie znajduje się ten wzorzec. Jeśli uważnie czytaliście rozdział 6, to wiecie, że rozwiązaniem tego problemu jest *korelacja* (filtr dopasowany) i że może ona być zrealizowana przy użyciu *splotu*.

Przed przeprowadzeniem splotu trzeba zrobić dwie modyfikacje, aby zamienić docelowy obraz w funkcję PSF. Przedstawiono to na rys. 24.12. Na rys. 24.12 a) pokazano sygnał docelowy, a więc wzorzec, który próbujemy wykryć. Na rys. 24.12 b) obraz został obrócony o 180°, tak jakby był odwrócony lewo-prawostronnie i góra–dół. Jak stwierdzono w rozdziale 7, przy wykonywaniu *korelacji* przy użyciu *splotu* sygnał docelowy powinien być odwrócony, aby przeciwdziałać odwróceniu zachodzącemu podczas operacji splotu. Wkrótce wrócimy do tego zagadnienia.

Druga modyfikacja to „sztuczka" poprawiająca efektywność algorytmu. Zamiast wykrywania twarzy na obrazie oryginalnym, bardziej skuteczne jest wykrywanie *krawędzi twarzy* w *krawędziach obrazu oryginalnego*. Krawędzie są bowiem ostrzejsze niż inne cechy obrazu, dzięki czemu korelacja daje wyraźniejsze maksimum. Ta modyfikacja nie jest konieczna, lecz daje wyniki znacznie lepsze. W najprostszej formie, przed przeprowadzeniem korelacji stosuje się filtr 3 × 3 do wykrywania krawędzi zarówno w sygnale oryginalnym, jak i docelowym. Ze względu na zasadę łączności operacji splotu, taka procedura jest równorzędna *dwukrotnemu* zastosowaniu filtru wykrywającego krawędzie do sygnału docelowego, bez zajmowania się obrazem oryginalnym. W praktyce na ogół wystarcza jednokrotne użycie filtru 3 × 3 wykrywającego krawędzie. W ten sposób obraz b) na rys. 24.12 jest zamieniany w obraz c). Obraz c) jest funkcją PSF, która ma być zastosowana w splocie.



Rys. 24.12. Opracowanie odpowiedzi impulsowej filtru korelacyjnego. Sygnał docelowy pokazano na rys. a). Rys. b) obrócono o 180°, aby skompensować odwrócenie wynikające ze splotu i móc wykonać korelację. Zastosowanie filtru wykrywającego krawędzie daje obraz c), który w tym przykładzie jest zastosowany jako odpowiedź impulsowa filtru



Rys. 24.13. Sieć działań obrazowego splotu FFT. Obrazy z rys. a) i b) są przekształcane do dziedziny częstotliwości przy użyciu FFT. Następnie te dwa widma częstotliwościowe są wymnażane i przez odwrotne przekształcenie FFT powraca się do dziedziny przestrzeni. W tym przykładzie, obrazy oryginalne wybrane i wstępnie przygotowano do wykonania korelacji przy użyciu splotu. Na rysunku jako IDFT oznaczono odwrotne przekształcenie DFT

Szczegóły splotu FFT przedstawiono na rys. 24.13. W tym przykładzie wykonujemy splot obrazu a) z obrazem b), aby uzyskać obraz c). Fakt, że te obrazy zostały wybrane i wstępnie przygotowane do przeprowadzenia *korelacji*, nie jest tu istotny; jest to ogólna sieć działań *splotu*. Pierwszym krokiem jest dopełnienie obu splatanych sygnałów odpowiednią liczbą zer, żeby ich rozmiary były potęgami dwóch i żeby były dostatecznie duże do zmieszczenia obrazu końcowego. Gdy na przykład splotowi są poddawane obrazy 100 × 100 pikseli i 29 × 29 pikseli, wynikowy obraz będzie miał rozmiar 128 × 128 pikseli. Trzeba zatem dodać wystarczająco dużo zer do obrazów na rysunkach 24.13 a), b), aby osiągnęły one rozmiary 128 × 128 pikseli. Jeśli tego nie uczynimy, to nastąpi splot kołowy i obraz końcowy będzie zniekształcony. Jeśli macie kłopoty ze zrozumieniem tych problemów, wróćcie do rozdziału 18, gdzie są one szczegółowo przeanalizowane dla sygnału jednowymiarowego.

Do przekształcenia obrazów z rysunków 24.13 a), b) w dziedzinę częstotliwości stosujemy algorytm FFT. W rezultacie uzyskujemy *cztery* matryce 128 × 128 – części rzeczywiste i urojone dwóch obrazów poddawanych splotowi. Wymnażanie części rzeczywistych i urojonych obu tych obrazów daje część rzeczywistą i urojoną obrazu pokazanego na rys. 24.13 c). Jeśli musicie sobie przypomnieć, jak to się robi, spójrzcie na wzór 9.1. Wykonanie odwrotnego przekształcenia FFT zakańcza ten algorytm, dając końcowy, spleciony obraz.

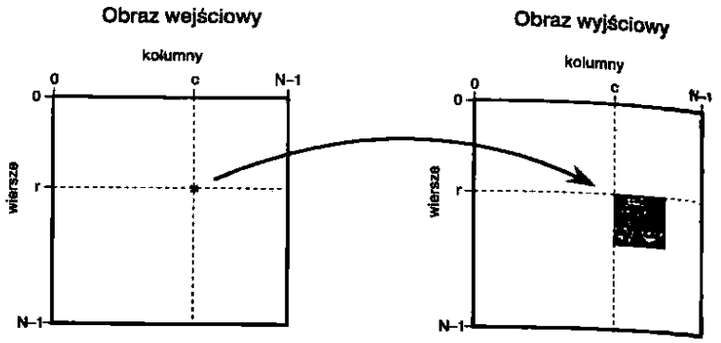
Wartość każdego piksela w obrazie korelacyjnym jest miarą zgodności – *w tym punkcie* – obrazu docelowego z szukanym. W omawianym przykładzie obraz korelacyjny zawiera szum oraz pojedynczy jasny pik wskazujący dobre dopasowanie do sygnału docelowego. Przez proste zlokalizowanie najjaśniejszego piksela w tym obrazie określamy wykryte współrzędne twarzy. Gdybyśmy nie zastosowali modyfikacji polegającej na wykrywaniu krawędzi, to też uzyskalibyśmy pik, lecz znacznie mniej wyraźny.

Korelacja jest wprawdzie silnym narzędziem w przetwarzaniu obrazów, lecz ma jedno istotne ograniczenie: obraz docelowy musi mieć dokładnie takie same *rozmiary* i *ustawienie kątowe*, jak odpowiedni obszar w obrazie poszukiwanym. Szum i inne zmiany wartości każdego piksela są stosunkowo mało ważne, a krytyczne znaczenie ma dokładne dopasowanie przestrzenne. To powoduje, że metoda jest prawie bezużyteczna na przykład do wykrywania nieprzyjacielskich czołgów na zdjęciach z rozpoznania wojskowego, guzów nowotworowych na obrazach medycznych i broni ręcznej przy skanowaniu bagażu na lotniskach. Jednym ze sposobów jest *wielokrotne* korelowanie obrazu z różnymi kształtami i ustawieniami kątowymi obrazu docelowego. W zasadzie ta metoda działa, lecz czas wykonania spowoduje, że będziecie musieli zrezygnować z jakiegokolwiek pośpiechu.

Bliższe spojrzenie na splot obrazowy

Ten właśnie omówiony przykład wykorzystajmy jeszcze do bardziej szczegółowego przyjrzenia się splotowi obrazowemu. Podobnie, jak w przypadku sygnałów jednowymiarowych, splot obrazowy może być widziany albo od *strony wejścia*, albo *wyjścia*. Jak pamiętacie z rozdziału 6, punkt widzenia od strony wejścia jest najlepszym opisem działania splotu. Natomiast przy analizach matematycznych i opracowaniach algorytmów stosuje się zwykle opis od strony wyjścia. Powinniście znać oba sposoby spojrzenia na tę operację.

Na rys. 24.14 zilustrowano opis splotu obrazowego od strony wejścia. Każdy piksel w obrazie wejściowym powoduje dodawanie do obrazu wyjściowego przeskalowanej i przesuniętej funkcji PSF. Następnie obraz wyjściowy jest obliczany jako suma wszystkich PSF mających swój udział w tym obrazie. Na rysunku pokazano, jaki udział w obrazie wyjściowym ma punkt o lokalizacji [r, c] w obrazie wejściowym. Funkcja PSF jest przesunięta w taki sposób, że piksel [0, 0] w PSF jest ustawiony zgodnie z pikselem [r, c] w obrazie wyjściowym. Jeśli PSF jest zdefiniowana tylko dla indeksów dodatnich, jak w tym przykładzie, to przesunięta PSF będzie się znajdowała tylko w dół i na prawo od [r, c]. Nie dziwcie się, że twarz na tym rysunku pojawia się odwrócona. Właśnie ta odwrócona twarz *jest* funkcją PSF, którą



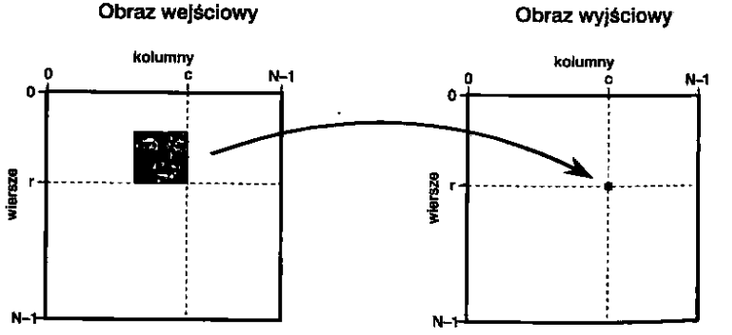
Rys. 24.14. Splot obrazowy widziany od strony wejścia. Każdy piksel w obrazie wejściowym wnosi przeskalowaną i przesuniętą funkcję PSF jako wkład do obrazu wyjściowego. Obraz wyjściowy jest sumą tych wkładów. Twarz na tym rysunku jest odwrócona, gdyż taką właśnie PSF stosujemy

stosujemy w tym przykładzie (rys. 24.13 a). W widzeniu od strony wejścia nie ma żadnego obrócenia PSF; jest ona po prosu przesunięta.

Splot obrazowy widziany od strony wyjścia przedstawiono na rys. 24.15. Każdy piksel w obrazie wyjściowym, taki jak pokazany na rysunku w położeniu [r, c], otrzymuje udziały od wielu pikseli z obrazu wejściowego. Funkcja PSF jest obracana o 180° wokół piksela [0, 0] i następnie przesuwana w taki sposób, żeby piksel [0, 0] w PSF był ustawiony zgodnie z pikselem [r, c] w obrazie wejściowym. Jeśli w PSF są użyte tylko piksele dodatnie, to PSF znajdzie się powyżej i na lewo od pozycji [r, c] w obrazie wejściowym. Wartość piksela [r, c] w obrazie wyjściowym oblicza się przez wymnożenie pikseli w obróconej PSF z odpowiednimi pikselami w obrazie wejściowym oraz zsumowanie uzyskanych iloczynów. Ta procedura jest opisana wzorem 24.3 i programem podanym w tablicy 24.1.

$$y[r,c]=\sum_{k=0}^{M-1}\sum_{j=0}^{M-1} h[k,j]x[r-k,c-j]$$

Wzór 24.3. Splot obrazowy. Obrazy x[,] i h[,] są splatane w celu utworzenia obrazu y[,]. Obraz h[,] ma rozmiary M × M pikseli, o indeksach zmieniających się od 0 do M – 1. W tym wzorze pojedynczy piksel w obrazie wyjściowym y[r, c] jest obliczany zgodnie ze spojrzeniem od strony wyjścia. Indeksy j oraz k są stosowane do utworzenia pętli przez wiersze i kolumny h[,] w celu obliczenia sumy iloczynów



Rys. 24.15. Splot obrazowy widziany od strony wyjścia. Każdy piksel w sygnale wyjściowym jest równy sumie pikseli w obróconej PSF przemnożonych przez odpowiadające im piksele w obrazie wejściowym

24.1

```
KONWENCJONALNY SPLOT OBRAZOWY

DIM X[99,99]              'przechowuje obraz wejściowy, 100×100 pikseli
DIM H[28,28]              'przechowuje odpowiedź impulsową filtru, 29×29 pikseli
DIM Y[ 127,127]           'przechowuje obraz wyjściowy, 128×128 pikseli

FOR R% = 0 TO 127         'pętla przez wszystkie wiersze i kolumny na wyjściu
FOR C% = 0 TO 127         'obliczanie piksela obrazowego z wzoru 24.3

  Y[R%,C%] = 0            'zerowanie piksela, aby mógł być użyty jako akumulator

  FOR J% = 0 TO 28        'wymnażanie każdego piksela w odpowiedzi impulsowej filtru
  FOR K% = 0 TO 28        'przez odpowiadający mu piksel w obrazie wejściowym i dodanie wyniku
                          'do akumulatora
    Y[R%,C%] = Y[R%,C%] + H[J%,K%] * X[R%-J%,C%-J%]
  NEXT K%
  NEXTJ%

NEXT C%
NEXT R%

END
```

Zwróćcie uwagę, że obrót PSF wynikający ze splotu skompensował odwrócenie dokonane przy projektowaniu PSF. Dlatego twarz na rys. 24.15 nie jest odwrócona, mając takie samo ukierunkowanie, jak wzorzec wykrywany w obrazie wejściowym. A zatem z powodzeniem zastosowaliśmy *splot* do przeprowadzenia *korelacji*. Porównajcie rys. 24.13 c) z rys. 24.15, aby zobaczyć, jak jasna plama w obrazie korelacyjnym wskazuje na to, że obraz docelowy został wykryty.

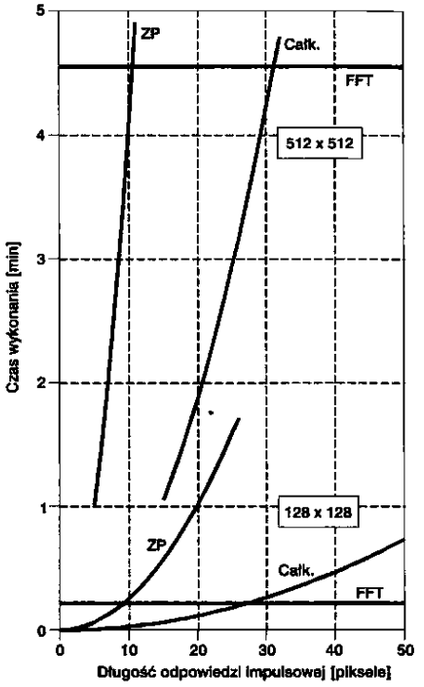
Splot FFT daje taki sam obraz wyjściowy, jak program konwencjonalnego splotu podany w tablicy 24.1. Czy skrócony czas wykonania osiągany w splocie FFT jest rzeczywiście wart dodatkowej złożoności programu? Przyjrzyjmy się temu bliżej. Na rys. 24.16 przeprowadzono porównanie czasów wykonania dla: splotu konwencjonalnego z operacjami *zmiennoprzecinkowymi* (oznaczenie ZP), splotu konwencjonalnego na liczbach całkowitych (oznaczenie CAŁK) oraz splotu FFT z operacjami zmiennoprzecinkowymi (oznaczenie FFT). Przedstawiono dane dla dwóch różnych rozmiarów obrazu, 512 × 512 oraz 128 × 128.

Po pierwsze, zauważcie, że czas wykonania splotu FFT nie zależy od rozmiarów odpowiedzi impulsowej filtru, co obrazują płaskie krzywe na wykresie. Na komputerze osobistym Pentium 100 MHz obraz 128 × 128 może być poddany splotowi w czasie ok. 15 sekund przy użyciu splotu FFT, a obraz 512 × 512 wymaga ponad 4 minut[27]. Zsumowanie liczby obliczeń pokazuje, że czas wykonania splotu FFT jest proporcjonalny do $N^2 \log_2 N$ dla obrazu $N \times N$. Tak więc obraz 512 × 512 wymaga czasu ok. 20 razy dłuższego niż obraz 128 × 128.

Splot konwencjonalny ma czas wykonania proporcjonalny do N^2M^2 dla obrazu $N \times N$ splatanego z odpowiedzią impulsową filtru $M \times M$. Można to zrozumieć, rozpatrzywszy program podany w tablicy 24.1. Można powiedzieć, że czas wykonania splotu konwencjonalnego *bardzo silnie* zależy od rozmiarów odpowiedzi impulsowej filtru. Jak pokazano na rysunku, splot FFT jest szybszy od konwencjonalnego zmiennoprzecinkowego, jeśli odpowiedź impulsowa filtru jest większa niż ok. 10 × 10 pikseli. W większości przypadków można do splotu konwencjonalnego stosować liczby całkowite, podwyższając ten punkt zrównania szybkości do

[27] Obecnie (rok 2006) czas trwania podobnej operacji skrócono, w zależności od zastosowanego oprogramowania, do kilkunastu lub nawet kilku sekund. Zgodnie z danymi katalogowymi, MatLAb w wersji 14 zainstalowany na komputerze z procesorem Pentium 4 HT 2,8 GHz i pamięcią 1 GB wykona takie obliczenie w czasie nie dłuższym niż 1 do 2 sekund (za: *http://www.intel.com*) (przyp. tłum.).

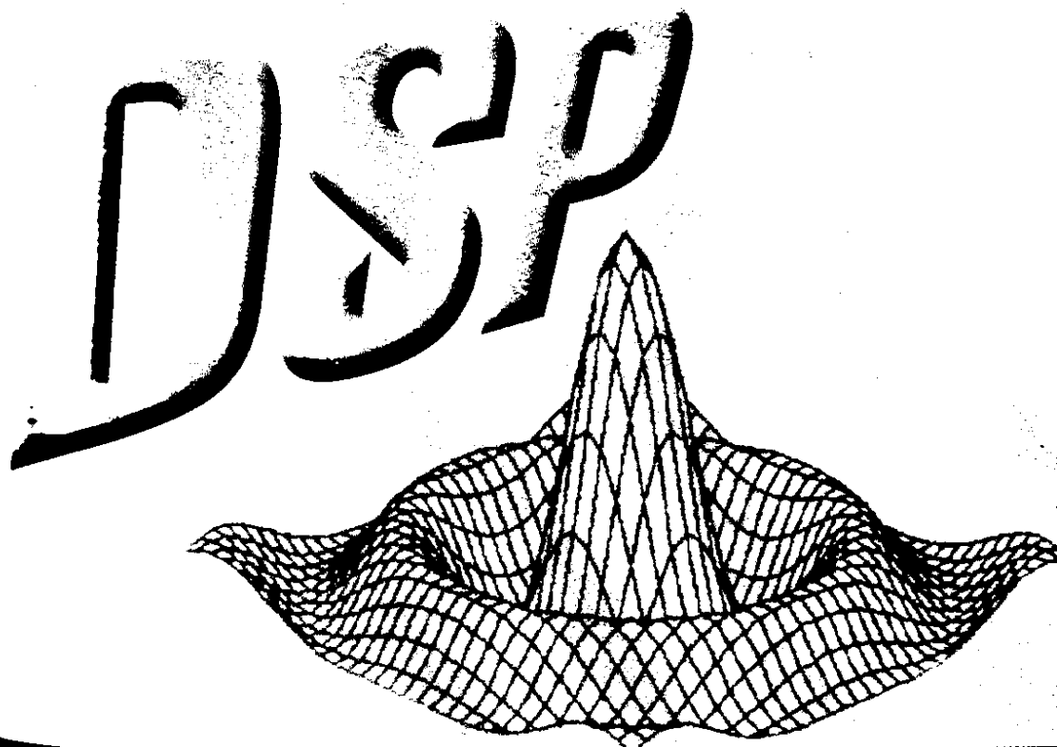
30 × 30 pikseli. Położenia tych punktów zrównania szybkości w niewielkim stopniu zależą od rozmiarów obrazu poddawanego splotowi, jak to pokazano na wykresie. Trzeba zapamiętać, że splot FFT jest użyteczny tylko dla *dużych* odpowiedzi impulsowych filtru.



Rys. 24.16. Czasy wykonania splotów obrazowych. Wykres pokazuje czasy wykonania na procesorze Pentium 100 MHz dla trzech metod realizacji splotu obrazowego: splotu konwencjonalnego przeprowadzanego przy użyciu operacji zmiennoprzecinkowych (ZP), splotu konwencjonalnego na liczbach całkowitych (CAŁK) oraz zmiennoprzecinkowego splotu FFT (oznaczenie FFT). Dwa zestawy krzywych dotyczą rozmiarów obrazu wejściowego 512 × 512 oraz 128 × 128 pikseli. Przy stosowaniu splotu FFT czas zależy tylko od rozmiarów obrazu, a nie od rozmiarów odpowiedzi impulsowej filtru. Natomiast dla splotu konwencjonalnego czas zależy zarówno od rozmiarów obrazu, jak i odpowiedzi filtru

25

Specjalne techniki obrazowe



W tym rozdziale przedstawiono cztery szczególne aspekty przetwarzania obrazów. Najpierw są rozważane sposoby definiowania *rozdzielczości przestrzennej*. Określa ona, jaki musi być minimalny rozmiar obiektu, aby był on widoczny na obrazie. Następnie jest rozpatrzony *stosunek sygnału do szumu*, decydujący o tym, jak słaby obiekt może jeszcze być odkrywany. W dalszej części rozdziału wprowadzono techniki *morfologiczne*. Są to operacje nieliniowe stosowane do obróbki obrazów binarnych, w których każdy piksel jest albo biały, albo czarny. Opisano też szczególnie godną uwagi technikę *tomografii komputerowej*. Zrewolucjonizowała ona diagnostykę medyczną, udostępniając szczegółowe obrazy wnętrza ciała ludzkiego.

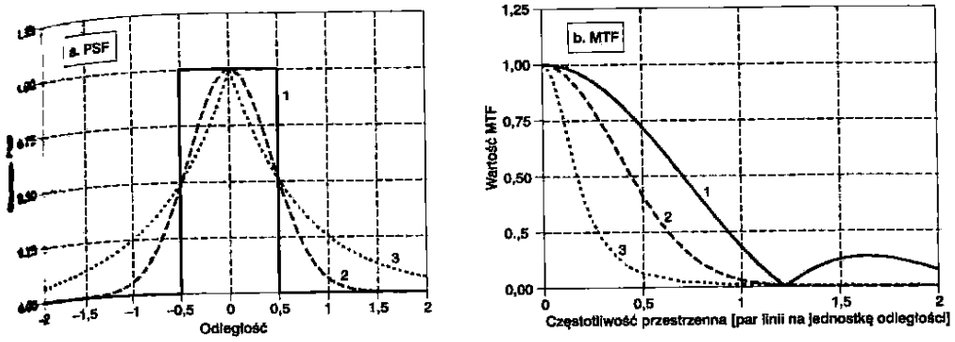
Rozdzielczość przestrzenna

Przypuśćmy, że chcemy porównać dwa systemy obrazowe pod względem ich rozdzielczości przestrzennej. Chcemy więc się dowiedzieć, który system jest zdolny do wykrycia najmniejszego obiektu. Dla uproszczenia problemu chcielibyśmy uzyskać odpowiedź w postaci jednej *liczby* charakteryzującej każdy system. Umożliwi to bezpośrednie porównania i ułatwi decyzję o wyborze systemu. Niestety, jeden parametr nie zawsze wystarcza do scharakteryzowania wszystkich subtelnych aspektów obrazowania. Sprawę komplikuje fakt, że rozdzielczość przestrzenna jest ograniczona dwoma wyraźnymi i wzajemnie powiązanymi wpływami, *odstępem* i *aperturą próbkowania*. Ten podrozdział poświęcono dwom głównym zagadnieniom: 1) jaki pojedynczy parametr może najlepiej scharakteryzować rozdzielczość przestrzenną, 2) jaka jest zależność między odstępem i aperturą próbkowania.

Na rys. 25.1 a) przedstawiono profile trzech kołowo symetrycznych funkcji PSF - typu „okrągła czapka", gaussowskiej i wykładniczej. Są to rodzaje PSF najczęściej stosowane w systemach obrazowych. Jak stwierdzono w poprzednim rozdziale, kształt funkcji PSF typu „okrągła czapka" może wynikać z niewłaściwego zogniskowania układu soczewek. Funkcja gaussowska zaś powstaje, gdy kumulują się błędy losowe, np. przy oglądaniu gwiazd przez zaburzoną atmosferę. Wykładnicza PSF jest wytwarzana na przykład wówczas, gdy elektrony lub promieniowanie rentgenowskie padają na warstwę luminoforu, gdzie są przetwarzane na światło. Jest stosowana w detektorach promieniowania, wzmacniaczach noktowizyjnych i monitorach ekranowych z lampami elektronopromieniowymi. W tych rozważaniach dokładny kształt trzech wymienionych funkcji PSF nie jest ważny. One tylko reprezentują wszystkie funkcje PSF spotykane w zastosowaniach praktycznych.

Funkcja PSF zawiera pełną informację o rozdzielczości przestrzennej. Aby wyrazić rozdzielczość jedną liczbą, możemy pominąć *kształt* PSF i po prostu zmierzyć *szerokość* wykresu tej funkcji. Najczęściej stosowanym sposobem określania szerokości jest wartość FWHM (*Full Width at Half Maximum*), czyli szerokości w połowie maksymalnej wysokości. Na przykład wszystkie PSF na rys. 25.1 a) mają szerokość FWHM równą 1 jednostce.

Ten sposób, niestety, ma dwie istotne wady. Po pierwsze, nie jest on zgodny z innymi miarami określającymi rozdzielczość przestrzenną, włącznie z subiektywną oceną obserwatorów oglądających obrazy. Po drugie, zazwyczaj bardzo trudno jest bezpośrednio pomierzyć PSF. Wyobraźcie sobie wprowadzanie impulsu do systemu obrazowego, czyli uzyskanie obrazu będącego bardzo małą, białą plamką na czarnym tle. Pobrany obraz będzie, z definicji, funkcją PSF systemu. Trudność polega na tym, że zmierzona funkcja PSF będzie zawierała tylko kilka pikseli, a kontrast będzie mały. Jeśli nie będziecie wystarczająco staranni, to szumy stłumią cały pomiar. Jako przykład wyobraźmy sobie, że obraz impulsowy jest matrycą 512 × 512 pikseli złożoną z samych zer oprócz jednego piksela mającego wartość 255. Porównajmy to teraz z normalnym obrazem, gdzie wszystkie 512 × 512 pikseli mają średnią wartość ok. 128. Według przybliżonego oszacowania, sygnał w obrazie impulsowym jest ok. 100 000 razy słabszy niż normalny obraz. Nic więc dziwnego, że stosunek sygnału do szumu jest zły: tam nie ma prawie żadnego sygnału!



Rys. 25.1. Wartości szerokości w połowie wysokości (FWHM) dla różnych funkcji MFT. Na rys. a) pokazano profile trzech funkcji PSF często spotykanych w systemach obrazowych: 1 – typu „okrągła czapka", 2 – gaussowskiej, 3 – wykładniczej. Każda z nich ma FWHM równą jednej jednostce. Odpowiednie funkcje MTF pokazano na rys. b). Podobne wartości FWHM nie odpowiadają, niestety, podobnym krzywym MTF

Przez całą treść tej książki przewija się podstawowa teza, że sygnały trzeba rozważać w tej dziedzinie, w której jest zakodowana ich informacja. Na przykład wszelkie operacje z sygnałami audio powinno się prowadzić w dziedzinie częstotliwości, a z sygnałami obrazowymi – w dziedzinie przestrzeni. Mimo to, jeden ze sposobów pomiaru rozdzielczości obrazu polega na rozpatrzeniu *odpowiedzi impulsowej*. To wprawdzie przeczy podstawowej filozofii tej książki, jest jednak metodą często stosowaną, którą należy poznać.

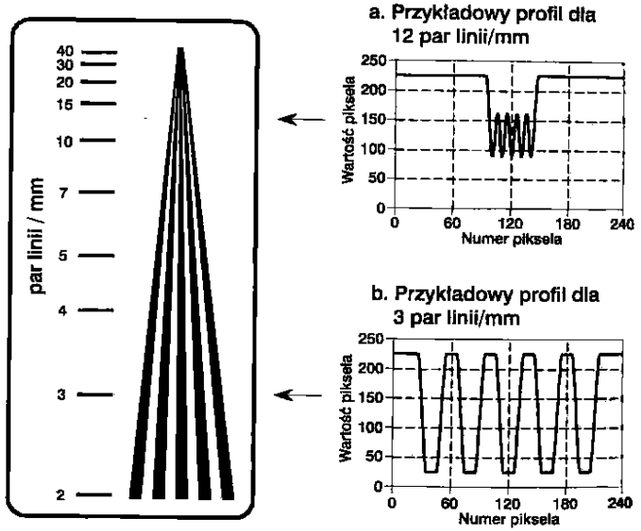
Przeprowadzenie dwuwymiarowego przekształcenia Fouriera funkcji PSF daje dwuwymiarową odpowiedź częstotliwościową. Jeśli PSF ma symetrię kołową, to odpowiedź częstotliwościowa też będzie miała taką symetrię. W tym przypadku pełna informacja o odpowiedzi częstotliwościowej jest zawarta w profilu PSF. Tak więc po wyznaczeniu odpowiedzi częstotliwościowej metodą FFT potrzebne są tylko kolumny od 0 do $N/2$ w wierszu 0. Taka prezentacja odpowiedzi częstotliwościowej jest nazywana **funkcją przenoszenia modulacji MTF** (*Modulation Transfer Function*). Na rys. 25.1 b) przedstawiono funkcje MTF dla trzech funkcji PSF przedstawionych na rys. 25.1 a). W przypadku, gdy PSF nie jest kołowo symetryczna, informację zawiera cała dwuwymiarowa odpowiedź częstotliwościowa. Na ogół jednak wystarczy znać krzywe MTF w kierunku poziomym i pionowym (tj. kolumny od 0 do $N/2$ w wierszu 0 oraz wiersze od 0 do $N/2$ w kolumnie 0). Zauważcie, że ta procedura wyodrębniania wiersza lub kolumny z dwuwymiarowego widma częstotliwościowego *nie jest* równoważna z wykonywaniem jednowymiarowego przekształcenia FFT profilów pokazanych na rys. 25.1 a). Wkrótce powrócimy do tej kwestii. Jak wynika z rys. 25.1, podobne wartości FWHM nie odpowiadają podobnym krzywym MTF.

Na rys. 25.2 pokazano **podziałkę z parami linii** (*line pair gauge*), urządzenie służące do pomiaru rozdzielczości obrazu przy użyciu MTF. Podziałki z parami linii mają różną postać, zależnie od konkretnego zastosowania. Na przykład wzór czarno-biały pokazany na tym rysunku może być bezpośrednio stosowany do testowania kamer wideo. W rentgenowskich systemach obrazowych żeberka podziałki mogą być wykonane z ołowiu, z umieszczonym między nimi materiałem przepuszczającym promieniowanie rentgenowskie. Główna właściwość tej podziałki polega na tym, że odstępy między liniami zmniejszają się w jednym kierunku. Gdy pobiera się obraz takiej podziałki z parami linii, to linie na końcu o gęstym rozmieszczeniu będą rozmyte, a na drugim końcu – wyraźne. Gdzieś pośrodku linie będą ledwo rozróżnialne. Obserwator, patrząc na obraz, identyfikuje to miejsce na podziałce i na kalibrowanej skali odczytuje odpowiadającą mu rozdzielczość.

Sposób, w jaki żeberka zlewają się ze sobą, jest ważny dla zrozumienia ograniczeń tej metody pomiaru. Wyobraźcie sobie pobieranie obrazu pokazanego na rys. 25.2. Na rysunku 25.2 a), b) przedstawiono przykłady małej i dużej częstotliwości przestrzennej. Przy małych częstotliwościach (rys. 25.2b) krzywa jest płaska na wierzchołkach i u dołu, lecz krawędzie są rozmazane. Przy większych częstotliwościach (rys. 25.2 a) *amplituda* modulacji się zmniejsza. Jest to zgodne z przebiegiem funkcji MTF na rys. 25.1 b) – przy większych częstotliwościach przestrzennych amplitudy są mniejsze. Pojedyncze żeberka są rozróżnialne w obrazie tak długo, jak amplituda pozostaje większa niż ok. od 3 do 10% wysokości pierwotnej. Jest to związane ze zdolnością oka do postrzegania małych różnic kontrastu między maksimami i minimami w obecności szumu obrazu.

Dużymi zaletami pomiaru podziałką z parami linii są prostota i szybkość. Największą wadą jest zależność wyniku od właściwości ludzkiego oka, a więc pewien składnik subiektywny. Nawet jeśli jest zmierzona cała krzywa MFT, to najczęściej stosowanym sposobem wyrażania rozdzielczości systemu jest podanie częstotliwości, przy której MFT jest zmniejszona do 3, 5 lub 10%. Niestety nie zawsze jest zdefiniowane, której z tych wartości użyto. W danych katalogowych są często podawane dość mgliste parametry, takie jak „rozdzielczość ograniczająca" (*limiting frequency*). Ponieważ producenci pragną, aby parametry ich wyrobów były możliwie jak najlepsze (niezależnie od tego, jak dobre są naprawdę te wyroby), więc najbezpieczniej jest interpretować dwuznaczne wartości jako wyznaczone przy redukcji krzywej do 3% krzywej MTF.

Delikatną kwestią, na którą trzeba zwrócić uwagę, jest definiowanie MTF w kategoriach fali *sinusoidalnej*, podczas gdy w podwójnej podziałce liniowej stosuje się fale *prostokątne*. Wynika to stąd, że żeberka są równomiernie rozłożonymi obszarami ciemnymi i jasnymi. Taki rodzaj ułatwia wyprodukowanie podziałki; trudno byłoby wytworzyć linie o sinusoidalnie zmieniającym się stopniu szarości. Jakie są konsekwencje stosowania fali prostokątnej do pomiaru MFT?



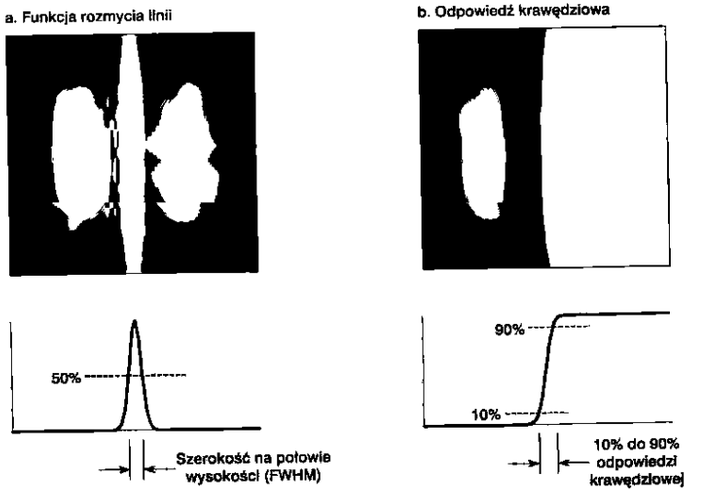
Rys. 25.2. Podziałka z parami linii. Podziałka jest narzędziem stosowanym do pomiaru rozdzielczości systemów obrazowych. Szereg czarnych i białych, zbiegających się żeber tworzy kontinuum częstotliwości przestrzennych. Rozdzielczość systemu wyznacza się jako częstotliwość, przy której oko przestaje rozróżniać poszczególne żebra. Ten przykład podziałki z parami linii pokazano w kilkakrotnym powiększeniu w stosunku do kalibrowanej skali

Przy dużych częstotliwościach przestrzennych wszystkie składowe częstotliwościowe, oprócz składowej podstawowej fali prostokątnej, są usuwane. To powoduje, że modulacja wydaje się być sinusoidalną, jak to pokazano na rys. 25.2 a). Przy małych częstotliwościach (jak na rys. 25.2 b) wydaje się, że fala jest prostokątna. Podstawowa fala sinusoidalna zawarta w fali prostokątnej ma amplitudę $4/\pi = 1{,}27$ razy większą od amplitudy fali prostokątnej (patrz tablica 13.10). W rezultacie podziałka z parami linii daje niewielkie przeszacowanie prawdziwej rozdzielczości systemu, poczynając od tego, że efektywna amplituda jest większa niż mierzona od czystej czerni do czystej bieli. Jest to sprawa ciekawa, lecz prawie zawsze ignorowana.

Ponieważ do pomiaru MFT są przemiennie używane fale prostokątna i sinusoidalna, więc powstała specjalna terminologia. Zamiast słowa „okres" w technice obrazowej używa się pojęcia **para linii** (linia ciemna obok jasnej). Na przykład częstotliwość przestrzenna jest określana jako *25 par linii na milimetr*, zamiast *25 okresów na milimetr*.

Szerokość przebiegu funkcji PSF nie jest dokładnie zgodna z percepcją człowieka i jest trudna do pomiaru. Metody MTF są realizowane w *dziedzinie niewłaściwej* dla zrozumienia wpływu rozdzielczości na zakodowaną informację. Czy jest jakaś korzystniejsza alternatywa? Tak, jest nią **funkcja rozmycia linii LSF** (*Line Spread Function*) oraz **odpowiedź krawędziowa**. Funkcja rozmycia linii (rys. 25.3) jest odpowiedzią systemu na cienką linię w poprzek obrazu. Podobnie odpowiedź krawędziowa wskazuje, jak system odpowiada na ostrą, prostoliniową nieciągłość (krawędź). Ponieważ linia jest pochodną (lub pierwszą różnicą) krawędzi, więc funkcja LSF jest pochodną (lub pierwszą różnicą) odpowiedzi krawędziowej. Zastosowanym tutaj pojedynczym parametrem pomiaru jest odległość konieczna do wzrostu odpowiedzi krawędziowej od 10 do 90%.

Stosowanie odpowiedzi krawędziowej do pomiaru rozdzielczości ma wiele zalet. Po pierwsze, pomiar ma tę samą postać, w jakiej jest zakodowana informacja obrazowa. Faktycznie, główną przyczyną, dla której chce się poznać wartość rozdzielczości systemu, jest chęć zrozumienia, jak są *rozmywane* krawędzie obrazu. Drugą zaletą jest łatwość pomiaru odpowiedzi krawędziowej spowodowana łatwym generowaniem krawędzi w obrazie. Jeśli trzeba, można łatwo określić funkcję LSF, wyznaczając pierwszą różnicę odpowiedzi krawędziowej.

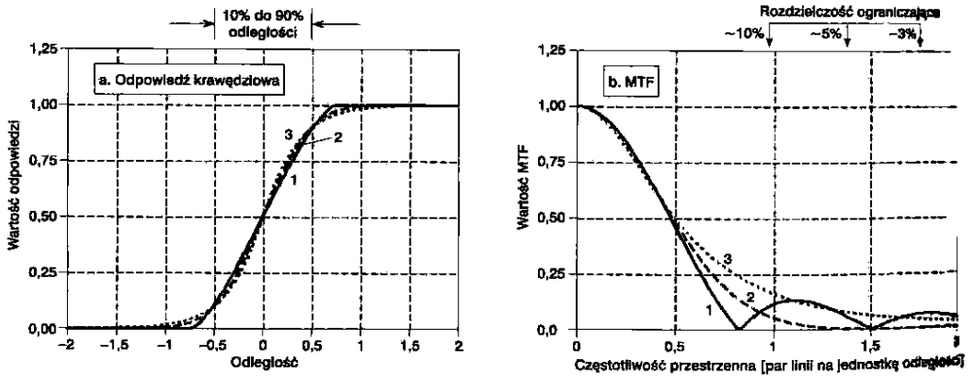


Rys. 25.3. Funkcja rozmycia linii i odpowiedź krawędziowa. Funkcja rozmycia linii (LSF) jest pochodną odpowiedzi krawędziowej. Szerokość wykresu LSF jest zwykle wyrażana jako szerokość na połowie maksymalnej wysokości (FWHM). Szerokość odpowiedzi krawędziowej jest na ogół podawana jako odległość mierzona od 10 do 90% pełnej wartości

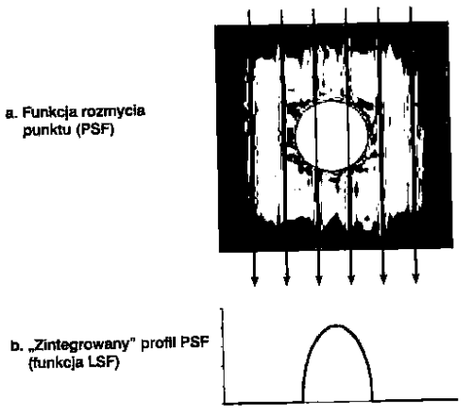
Trzecią zaletą jest podobieństwo kształtów wszystkich odpowiedzi krawędziowych, nawet jeśli mogą one pochodzić od drastycznie różnych funkcji PSF. Przedstawiono to na rys. 25.4 a), gdzie wykreślono odpowiedzi krawędziowe dla funkcji PSF typu „okrągła czapka", gaussowskiej i wykładniczej. Ze względu na podobieństwo kształtu, odległość mierzona od 10 do 90% amplitudy jest doskonałym pojedynczym parametrem określającym rozdzielczość. Czwartą zaletą jest możliwość bezpośredniego wyznaczania funkcji MTF przez wykonanie jednowymiarowego przekształcenia FFT funkcji LSF (inaczej niż przy obliczaniu MTF na podstawie PSF, gdzie konieczne jest zastosowanie dwuwymiarowego przekształcenia Fouriera). Na rys. 25.4 b) pokazano przebiegi funkcji MTF odpowiadających odpowiedziom krawędziowym z rys. 25.4 a). Tak więc krzywe z rys. 25.4 a) przetworzono na krzywe z rys. 25.4 b) przez wyznaczanie pierwszej różnicy (w celu znalezienia LSF) i następnie obliczenie FFT.

Piąta zaleta wynika z faktu, że podobne odpowiedzi krawędziowe mają podobne krzywe MTF, co jest widoczne na rysunkach 25.4 a), b). To ułatwia przechodzenie od jednego pomiaru do drugiego. W szczególności system, który ma odpowiedź krawędziową od 10 do 90% równą odległości x, ma rozdzielczość ograniczającą (kontrast 10%) równy ok. 1 parze linii na odległości x. Jednostki „odległości" są zależne od typu rozpatrywanego systemu. Jako przykład rozważmy trzy różne systemy obrazowe mające odpowiedzi krawędziowe od 10 do 90% równe 0,05 mm, 0,2 miliradiana i 3,3 piksela. Poziom kontrastu 10% na odpowiednich krzywych MTF będzie występował, odpowiednio, przy ok. 20 parach linii/mm, 5 parach linii/miliradian oraz 0,33 par linii/piksel.

Na rys. 25.5 przedstawiono zależność matematyczną między funkcjami PSF i LSF. Na rys. 25.5 a) pokazano funkcje PSF typu „okrągła czapka" będącą kołowym obszarem o wartości 1, pokazanym na biało, otoczonym przez obszar samych zer, pokazany jako szary. Profil tej funkcji PSF (tj. wykres wartości pikseli leżących wzdłuż linii poprowadzonej przez środek obrazu) będzie impulsem prostokątnym. Na rys. 25.5 b) pokazano odpowiadający temu profilowi przebieg funkcji LSF. Matematycznie, LSF jest równa *profilowi zintegrowanemu* PSF. Jest on określany przez przecięcia obrazu w kierunkach pokazanych na rysunku strzałkami. Każda wartość zintegrowanego profilu jest *sumą* wartości pikseli wzdłuż odpowiedniego przekroju.



Rys. 25.4. Odpowiedź krawędziowa i funkcja MTF. Na rys. a) pokazano odpowiedzi krawędziowe dla trzech funkcji PSF: 1 – typu „okrągła czapka", 2 – gaussowskiej, 3 – wykładniczej. Każda odpowiedź krawędziowa ma odległość od 10 do 90% równą jednej jednostce. Na rys. b) pokazano odpowiadające tym odpowiedziom krzywe MTF, które są do siebie podobne powyżej 10% wartości maksymalnej. Rozdzielczość ograniczająca jest niezbyt ściśle zdefiniowanym parametrem wyrażającym częstotliwość, przy której MTF ma wartość od 3 do 10%



Rys. 25.5. Zależność między funkcjami PSF i LSF. Na rys. a) pokazano funkcję PSF typu „okrągła czapka". Każdy wiersz lub kolumna poprowadzone przez biały obszar będą dawały impuls prostokątny. Na rys. b) pokazano funkcję LSF, równoważną zintegrowanemu profilowi PSF. To znaczy, że funkcja LSF jest wyznaczana przez przemiatanie obrazu w poprzek w jakimś kierunku i sumowanie (całkowanie) wartości pikseli wzdłuż każdego przekroju. W pokazanym przykładzie są sumowane piksele w każdej kolumnie

W tym przykładzie, gdzie linie przecięcia są pionowe, każdy punkt w zintegrowanym profilu jest wyznaczany przez sumowanie wszystkich wartości pikseli w każdej kolumnie. To koresponduje z funkcją LSF takiej linii, która w obrazie jest *pionowa*. Funkcję LSF linii, która w obrazie jest *pozioma*, wyznacza się przez sumowanie wszystkich wartości pikseli w każdym *wierszu*. Dla obrazów ciągłych koncepcja jest taka sama, lecz sumy są zamieniane całkami.

Jak pokazano w tym przykładzie, funkcję LSF można obliczyć bezpośrednio z PSF. Jednak PSF nie zawsze może być obliczone z LSF. Przyczyna jest w tym, że PSF zawiera informacje o rozdzielczości przestrzennej we *wszystkich kierunkach*, podczas gdy LSF jest ograniczona do tylko jednego, szczególnego kierunku. System ma tylko jedną funkcję PSF, lecz nieskończoną liczbę funkcji LSF, jedną dla każdego kąta. Weźmy jako przykład system, który ma funkcję PSF o kształcie podłużnym. To powoduje, że inna będzie rozdzielczość przestrzenna w kierunku poziomym, a inna w pionowym, a więc inna będzie funkcja LSF w tych kierunkach. Mierząc LSF dla jednej wartości kąta, nie uzyskuje się informacji wystarczającej do obliczenia pełnej funkcji PSF, poza tymi szczególnymi przypadkami, gdy funkcja PSF jest kołowo symetryczna. Wielokrotne pomiary LSF dla różnych wartości kąta umożliwiają obliczenie niekołowej funkcji PSF, lecz operacje matematyczne są wówczas zawiłe i zwykle niewarte wysiłku. Problem obliczania PSF na podstawie pewnej liczby pomiarów LSF jest dokładnie takim samym problemem, jaki występuje w *tomografii komputerowej*, która będzie omówiona w dalszej części tego rozdziału.

W praktyce, w większości systemów obrazowych funkcje LSF i PSF nie różnią się od siebie w sposób zasadniczy. Często można spotkać zastosowanie jednej funkcji jako aproksymacji drugiej. Tym bardziej usprawiedliwione jest więc stwierdzenie, że w dwóch często spotykanych przypadkach te funkcje są identyczne: prostokątna funkcja PSF ma prostokątną LSF (o takiej samej szerokości) oraz gaussowska PSF ma gaussowską LSF (o takim samym odchyleniu standardowym).

Przedstawione koncepcje można podsumować przez umiejętność odpowiedzi na dwa pytania: jak *oszacować* wyspecyfikowaną wartość rozdzielczości oraz jak samemu *zmierzyć* rozdzielczość. Przypuśćmy, że natknęliście się na ogłoszenie o treści: „Ten system rozróżnia 40 par linii na milimetr". Zinterpretowalibyście to następująco: „Sinusoida o 40 podwójnych liniach na milimetr będzie miała amplitudę zredukowaną do 3–10% prawdziwej wartości i na obrazie będzie ledwo widzialna". Powinniście też od razu przeliczyć w pamięci, że 40

par linii na milimetr przy kontraście 10% jest równoważne odpowiedzi krawędziowej (od 10 do 90%) wynoszącej 1/40 par linii/mm = 0,025 mm. Jeśli MTF jest określone dla pewnego kontrastu 3%, to odpowiedź impulsowa będzie od ok. 1,5 do 2 razy szersza.

Mierząc rozdzielczość przestrzenną systemu obrazowego, wyżej opisane kroki wykonuje się w odwrotnej kolejności. Trzeba umieścić ostrą krawędź w obrazie i zmierzyć wynikającą z tego odpowiedź krawędziową. Na tej krzywej odległość odpowiadająca narastaniu od 10% do 90% jest najlepszą, wyrażoną jednym parametrem, miarą rozdzielczości systemu. Wystarczy teraz obliczyć pierwszą różnicę odpowiedzi krawędziowej, aby wyznaczyć LSF i następnie stosując FFT, określić MTF.

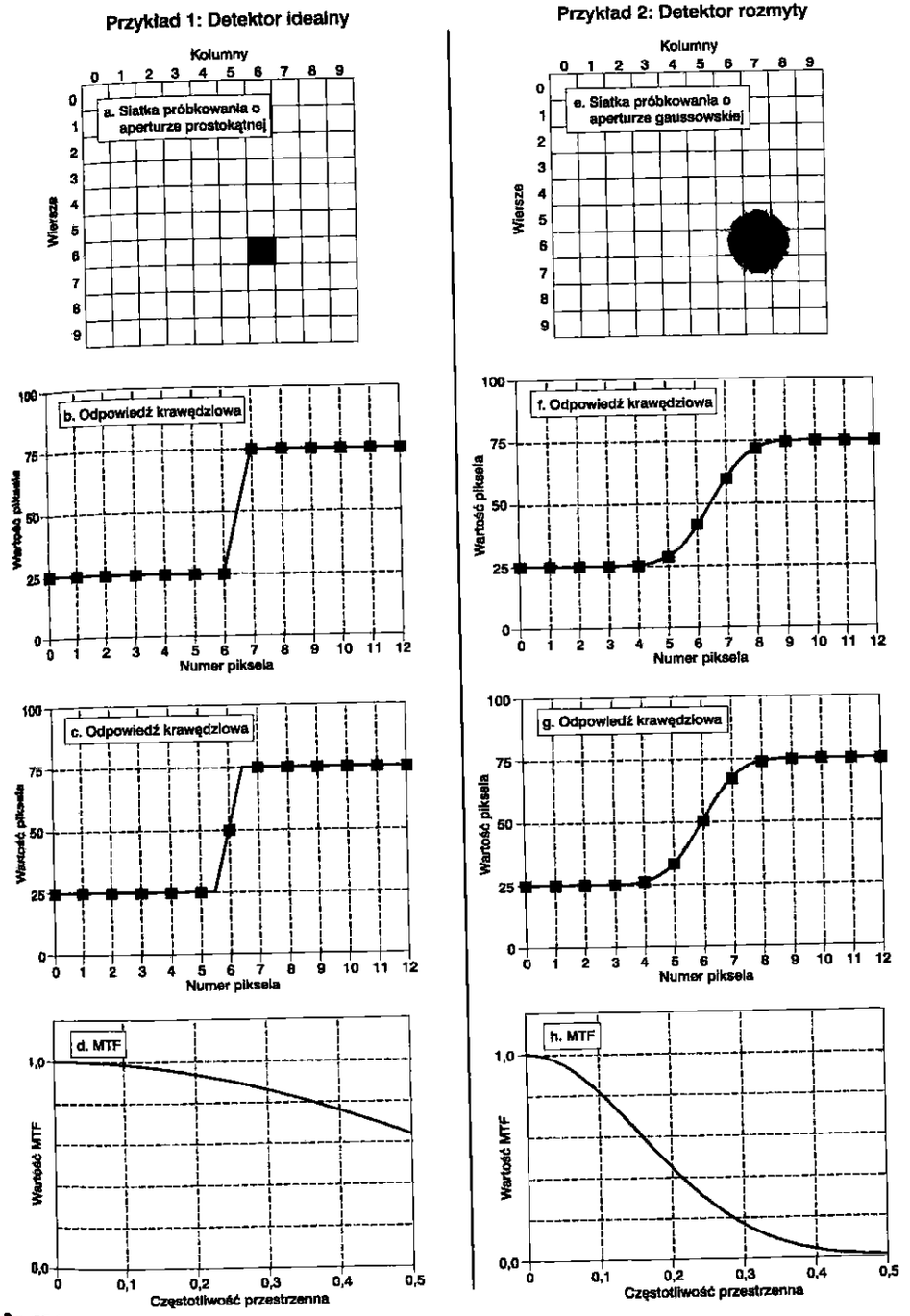
Odstęp i apertura próbkowania

Na rys. 25.6 pokazano dwa skrajne przypadki próbkowania, które będziemy nazywać **detektorem idealnym** oraz **detektorem rozmytym**. Wyobraźcie sobie, że na rys. 25.6 a) przed stawiono powierzchnię detektora obrazowego, np. CCD. Światło, trafiając gdziekolwiek w granicach jednego z kwadratowych pikseli, będzie miało wpływ na wartość *tylko* tego właśnie piksela, a nie żadnych innych. Zaznaczono to na rysunku na czarno jako aperturę próbkowania dokładnie wypełniającą jeden z kwadratowych pikseli. Dla detektora obrazu jest to sytuacja optymalna, gdyż *całe* światło ulega detekcji i nie ma *żadnego* nakładania się ani przesłuchu między sąsiednimi pikselami. Inaczej mówiąc, apertura próbkowania jest dokładnie równa odstępowi próbkowania.

Przykład alternatywny przedstawiono na rys. 25.6 e). Tutaj apertura próbkowania jest znacznie większa od odstępu próbkowania i jest zgodna z rozkładem gaussowskim. Tak więc wartość każdego piksela w detektorze uzyskuje pewien wkład od światła padającego na detektor w obszarze *wokół* piksela. To stwierdzenie powinno się wydać już znane, gdyż jest to zgodne z rozpatrywaniem splotu od strony wyjścia. Przy rozpatrywaniu od strony wejścia, wąska wiązka światła padająca na detektor miałaby udział w wartościach kilku sąsiadujących pikseli, także zgodnie z rozkładem gaussowskim.

Zwróćmy teraz uwagę na odpowiedzi impulsowe w tych dwóch podanych przykładach. Znaczniki na każdym z wykresów wskazują aktualne wartości pikseli, które można znaleźć na obrazie, a linie łączące pokazują *krzywą* podlegającą próbkowaniu. Ważne jest, że kształt tej krzywej jest określony *tylko* przez aperturę próbkowania. Wynika stąd, że rozdzielczość końcowego obrazu może być ograniczona dwoma sposobami. Po pierwsze, próbkowana krzywa może mieć słabą rozdzielczość spowodowaną zbyt dużą aperturą próbkowania. Po drugie, odstęp próbkowania może być zbyt duży, w wyniku czego są tracone drobne szczegóły między próbkami. Dla każdego z dwóch przykładów przedstawiono odpowiedzi krawędziowe, z których wynika, że pobieranie próbki może następować w różnych dowolnych miejscach wzdłuż próbkowanej krzywej. A zatem obrazowana krawędź może znajdować się na pikselu albo obejmować dwa piksele. Zauważcie, że detektor idealny ma na narastającej części krawędzi zero próbek lub jedną próbkę. Natomiast detektor rozmyty ma *trzy* lub cztery próbki na narastającej części krawędzi.

Jaka jest rozdzielczość ograniczająca w tych dwóch systemach? Odpowiedzi dostarcza *twierdzenie o próbkowaniu*. Jak to rozważano w rozdziale 3, próbkowanie wychwytuje wszystkie składowe o częstotliwościach leżących poniżej połowy częstotliwości próbkowania. Większe częstotliwości są tracone z powodu zjawiska aliasingu. Spójrzcie teraz na funkcję MTF na rys. 25.6 h). Apertura próbkowania detektora rozmytego spowodowała usunięcie wszystkich częstotliwości większych od połowy częstotliwości próbkowania, a zatem *nic* nie jest tracone podczas próbkowania. To znaczy, że rozdzielczość tego systemu jest ograniczona całkowicie przez aperturę, a nie przez odstęp próbkowania. Patrząc na to z innej strony, można stwierdzić, że apertura próbkowania zadziałała jak filtr antyaliasingowy, umożliwiając bezstratne próbkowanie.



Rys. 25.6.

Natomiast z krzywej MTF na rys. 25.6 d) wynika, że rozdzielczość tego systemu jest ograniczona *obiema* wymienionymi wielkościami. Opadanie krzywej MTF przy dużych częstotliwościach wyraża utratę informacji spowodowaną *aperturą próbkowania*. Ponieważ wykres MTF nie opadł do zera poniżej częstotliwości 0,5, więc jakaś część informacji jest również tracona w wyniku ograniczonej wartości *odstępu próbkowania*. Który z tych parametrów, odstęp czy apertura, bardziej ogranicza rozdzielczość? Na to pytanie trudno odpowiedzieć jakąś liczbą, ponieważ każdy z tych dwóch parametrów pogarsza rozdzielczość w inny sposób. Zadowólcie się stwierdzeniem, że rozdzielczość detektora idealnego (przykład 1) jest ograniczona przede wszystkim odstępem próbkowania.

Chociaż te rozważania wydają się trudne, to sprowadzają się one do bardzo prostej reguły, przydatnej w zastosowaniach praktycznych. Rozważcie system, w którym odległość dla narastania od 10 do 90% odpowiedzi krawędziowej jest równa np. 1 mm. Jeśli odstęp próbkowania jest większy niż 1 mm (czyli wzdłuż krawędzi jest mniej niż jedna próbka), to ograniczeniem systemu będzie *odstęp próbkowania*. Jeśli odstęp próbkowania jest mniejszy od 0,33 mm (tak krawędzi jest więcej niż 3 próbki), to rozdzielczość będzie ograniczana *aperturą próbkowania*. Gdy system daje na krawędzi od 1 do 3 próbek, to ograniczać będą oba czynniki.

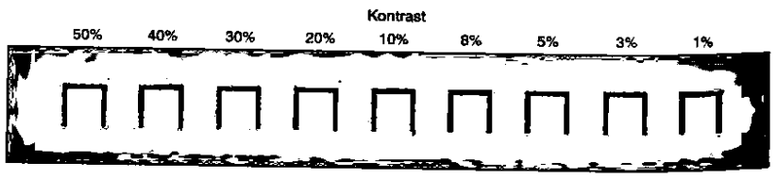
Stosunek sygnału do szumu

Obiekt jest widoczny na obrazie, ponieważ ma jaskrawość inną niż otoczenie. Tak więc kontrast obiektu (czyli sygnału) powinien przeważać nad szumem obrazu. Można tu wyróżnić ograniczenia *oka* i ograniczenia *danych*.

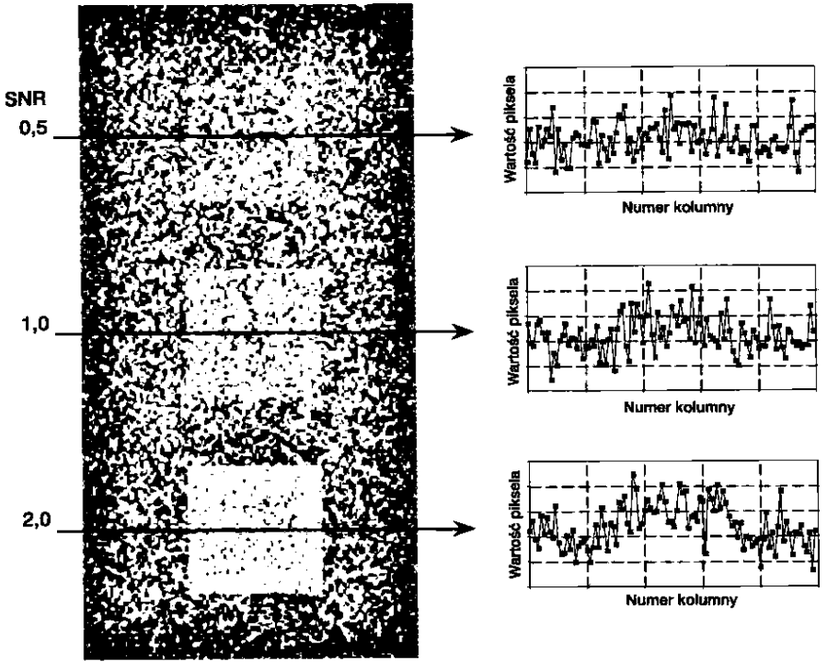
Eksperyment przedstawiony na rys. 25.7 służy do pomiaru zdolności oka do wykrywania słabych sygnałów. Zależnie od warunków obserwacji oko ludzkie może wykrywać minimalny kontrast od 0,5 do 5%. Człowiek potrafi więc rozróżniać między od ok. 20 do 200 odcieniami szarości – od najczarniejszej czerni do najbielszej bieli. Dokładna liczba tych odcieni zależy od różnych czynników, takich jak jaskrawość oświetlenia otoczenia, odległość między dwoma porównywanymi obszarami i sposób tworzenia obrazu w skali szarości (monitor wizyjny, fotografia, chemigrafia itd.).

Opisane w rozdziale 23 przekształcenie skali szarości może być wykorzystane do zwiększenia kontrastu w wybranym zakresie wartości pikseli. Jest to cenne narzędzie w pokonywaniu ograniczeń ludzkiego wzroku. Kontrast przy jednym poziomie jaskrawości jest zwiększany kosztem zmniejszenia kontrastu przy innych jaskrawościach. Jednak jest to skuteczne tylko po warunkiem, że kontrast obiektu nie ginie w losowym szumie obrazowym. Wtedy sytuacja jest trudniejsza; *sygnał* nie zawiera wystarczająco dużo informacji, aby móc ujawnić obiekt, niezależnie od działania oka.

Na rys. 25.8 pokazano obraz z trzema kwadratami o kontrastach 5, 10 i 20%. Tło zawiera szum losowy o rozkładzie normalnym z odchyleniem standardowym równym ok. 10% kontrastu. Stosunek sygnału do szumu jest definiowany jako kontrast podzielony przez odchylenie standardowe szumu. Trzy kwadraty mają więc stosunek sygnału do szumu o wartościach 0,5; 1,0 oraz 2,0. Kłopoty zaczynają się na ogół wówczas, gdy stosunek sygnału do szumu zmniejsza się poniżej wartości ok. 1,0.



Rys. 25.7. Wykrywanie kontrastu. Oko ludzkie potrafi wykrywać minimalny kontrast od ok. 0,5 do 5%, zależnie od warunków obserwacji. Kontrast 100% jest różnicą między czystą czernią i czystą bielą



Rys. 25.8. Minimalny wykrywalny stosunek sygnału do szumu. Obiekt jest widoczny w obrazie tylko wówczas, gdy jego kontrast jest wystarczająco duży, aby nie ginąć w losowym szumie obrazowym. W tym przykładzie trzy kwadraty mają stosunek sygnału do szumu o wartościach 2,0; 1,0 oraz 0,5, przy czym stosunek ten jest definiowany jako iloraz kontrastu obiektu i odchylenia standardowego szumu

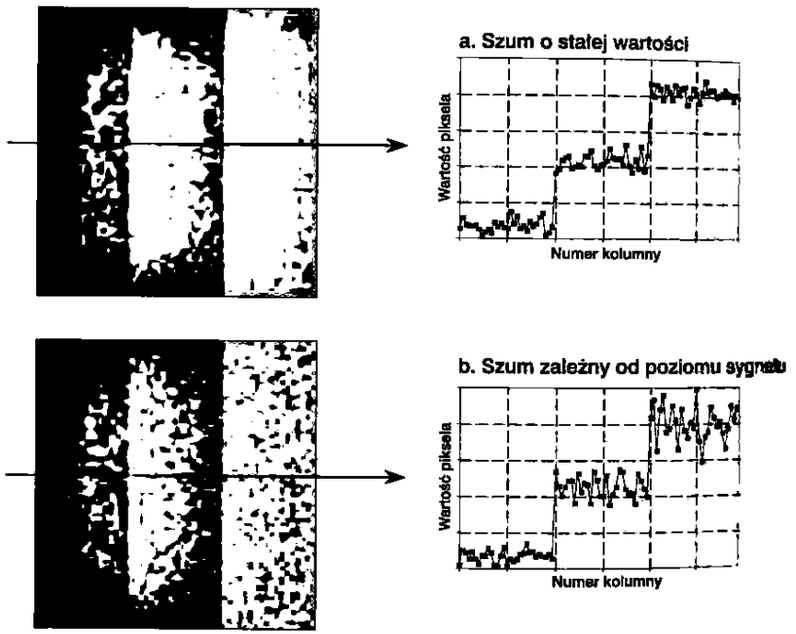
Dokładna wartość minimalnego wykrywalnego stosunku sygnału do szumu zależy od *rozmiaru* obiektu – im większy jest obiekt, tym łatwiej go wykryć. Aby to zrozumieć, wyobraźcie sobie wygładzanie obrazu z rys. 25.8 przy użyciu filtru o prostokątnej odpowiedzi impulsowej 3 × 3. Wygładzanie pozostawia kontrast niezmieniony, lecz tłumi szum o czynnik trzy (tj. o pierwiastek kwadratowy liczby pikseli w odpowiedzi impulsowej). Ponieważ stosunek sygnału do szumu jest zwiększony trzykrotnie, więc stają się widoczne obiekty o mniejszych rozmiarach. Aby móc zobaczyć obiekty jeszcze słabsze, trzeba jeszcze bardziej powiększyć odpowiedź impulsową filtru. Na przykład filtr o odpowiedzi impulsowej 5 × 5 poprawia stosunek sygnału do szumu o czynnik $\sqrt{25} = 5$. Taką procedurę można kontynuować tak długo, aż odpowiedź impulsowa filtru zrówna się swym rozmiarem z wykrywanym obiektem. Z tego wynika, że zdolność wykrywania obiektu jest proporcjonalna do *pierwiastka kwadratowego* jego *powierzchni*. Przy podwojeniu średnicy obiektu, może on być wykrywany na tle dwa razy większego szumu.

Podobne procesy następują w mózgu podczas przetwarzania obrazu. Wygładzanie oglądanego obrazu przez filtry o odpowiedziach impulsowych o różnych rozmiarach jest próbą rozpoznawania obiektów o małym kontraście. Trzy profile na rys. 25.8 mają wykazać, jak dobry jest ludzki wzrok w wykrywaniu obiektów znajdujących się w otoczeniu szumu. Chociaż obiekty są na profilach trudne do zidentyfikowania, to są w sposób oczywisty widoczne na obrazie. Jeśli chcecie w pełni docenić możliwości ludzkiego systemu widzenia, spróbujcie napisać algorytmy przeznaczone do przetwarzania obrazów przy małym stosunku sygnału do szumu. Wtedy z pokorą przyznacie, że nasz mózg potrafi zrobić wiele rzeczy, których nie jest w stanie wykonać napisany przez was program!

Losowy szum obrazowy występuje zwykle w dwóch postaciach. Szum pierwszego typu (rys. 25.9 a) ma stałą amplitudę, a więc jasne i ciemne obszary obrazu są jednakowo zaszumione. Natomiast szum pokazany na rys. 25.9 b) *wzrasta przy wzrastającym poziomie sygnału*, w wyniku czego jasne obszary są bardziej zaszumione niż ciemne. W większości obrazów pojawia się szum obu rodzajów, lecz jeden z nich jest zwykle dominujący. Na przykład, zazwyczaj szum zmniejsza się wraz z poziomem sygnału aż do osiągnięcia pewnego plateau stałej wartości szumu.

Źródłem szumu o stałej amplitudzie jest zwykle *przedwzmacniacz*. Wszystkie elektroniczne układy analogowe generują szum. Szum jest jednak najbardziej szkodliwy tam, gdzie wzmacniany sygnał jest jeszcze bardzo mały, bezpośrednio w CCD lub innym czujniku obrazowym. Szum przedwzmacniacza pochodzi od losowych ruchów elektronów w tranzystorach. Dlatego wartość szumu zależy od sposobu zaprojektowania układu, lecz nie zależy od poziomu wzmacnianego sygnału. Na przykład typowy aparat fotograficzny CCD ma stosunek sygnału do szumu od ok. 300 do 1000 (40 do 60 dB), definiowany jako pełny zakres poziomu sygnału podzielony przez odchylenie standardowe szumu o stałej amplitudzie.

Szum wzrastający z poziomem sygnału powstaje, gdy obraz jest reprezentowany przez małą liczbę indywidualnych zdarzeń. Mogą to być np. *fotony promieniowania rentgenowskiego* przechodzące przez ciało pacjenta, *fotony światła* wchodzące do aparatu fotograficznego lub *elektrony* znajdujące się w studni układu CCD. Reguły matematyczne rządzące tymi zjawiskami noszą nazwę **statystyki zliczania** lub **statystyki Poissona**. Przypuśćmy, że powierzchnia układu CCD jest równomiernie oświetlona tak, że w każdej studni jest średnio generowane 10 000 elektronów. W wyniku zwykłego przypadku w niektórych studniach może



Rys. 25.9. Szum obrazowy. Szum losowy w obrazie występuje w dwóch postaciach. Na rys. a) amplituda szumu pozostaje stała, gdy wzrasta poziom sygnału. Jest to typowa właściwość szumu elektronicznego. Na rys. b) amplituda szumu wzrasta proporcjonalnie do pierwiastka kwadratowego poziomu sygnału. Szum tego typu powstaje w wyniku detekcji małej liczby zdarzeń, takich jak fotony światła, elektrony, fotony promieniowania rentgenowskiego

znaleźć więcej elektronów, a w innych mniej. Ściślej mówiąc, liczba elektronów będzie miała rozkład normalny, czyli gaussowski o wartości średniej 10 000, o pewnym odchyleniu standardowym opisującym, jak duże są wahania liczby elektronów w poszczególnych studniach. W statystycznym rozkładzie Poissona odchylenie standardowe jest równe pierwiastkowi kwadratowemu z liczby indywidualnych zdarzeń. Jest to podstawowa cecha tego rozkładu. Wynika z niej, że jeśli jest N pikseli w każdym pikselu, to wartość średnia jest równa N, a odchylenie standardowe $\sqrt{N}$. Stosunek sygnału do szumu jest zatem równy $N/\sqrt{N}$ lub po prostu $\sqrt{N}$. Wyraża się to wzorami:

$$\mu = N$$
$$\sigma = \sqrt{N}$$
$$SNR = \sqrt{N}$$

Wzór 25.1. Rozkład statystyczny Poissona. W tym rozkładzie μ jest średnią liczbą indywidualnych zdarzeń, N. Odchylenie standardowe σ jest równe pierwiastkowi kwadratowemu średniej liczby indywidualnych zdarzeń. Stosunek sygnału do szumu jest wartością średnią podzieloną przez odchylenie standardowe

W podanym jako przykład układzie CCD odchylenie standardowe jest $\sqrt{10000} = 100$, stosunek sygnału do szumu jest też równy $\sqrt{10000} = 100$. Jeśli średnia liczba elektronów na studnię wzrośnie do miliona, to zarówno odchylenie standardowe, jak i stosunek do szumu wzrastają do 1000. Szum wzrasta więc ze wzrostem sygnału, jak widać to na rys. 25.9 b). Sygnał jednak wzrasta *szybciej* niż szum, powodując polepszenie stosunku sygnału do szumu. Nie należy mylnie sądzić, że mniejszy sygnał da mniejszy szum, a więc lepszą informację. Trzeba pamiętać, że celem *nie jest* tłumienie szumu, lecz wyodrębnienie sygnału z *szumu*. Dlatego podstawowym parametrem jest stosunek sygnału do szumu.

Wiele systemów obrazowych działa na zasadzie konwersji jednej cząstki (lub fotonu) w inną. Zastanówmy się na przykład, jakie zjawiska zachodzą w medycznym, rentgenowskim systemie obrazowym. W lampie rentgenowskiej *elektrony* uderzają w metalową tarczę, wytwarzając promieniowanie *rentgenowskie*. Po przeniknięciu przez ciało pacjenta promienie rentgenowskie trafiają w detektor z lampą próżniową, zwany wzmacniaczem obrazu. Tutaj promienie rentgenowskie są kolejno przetwarzane w *fotony świetlne*, następnie w *elektrony*, a potem znowu w *fotony świetlne*. Te fotony wchodzą do kamery, gdzie generują elektrony w studni układu CCD. W każdej z tych pośrednich postaci obraz jest reprezentowany przez skończoną liczbę cząstek (lub fotonów), z czego wynika dodany szum zgodny z wzorem 25.1. Całkowity stosunek sygnału do szumu wyraża łączny szum *wszystkich* stopni, jednak zwykle jeden stopień jest dominujący. Jest to stopień o *najgorszym* stosunku sygnału do szumu z powodu *najmniejszej* liczby cząstek (lub fotonów). Ten dominujący stopień ograniczający jest w terminologii angielskiej nazywany *quantum sink*.

W systemach noktowizyjnych takim dominującym ograniczeniem jest liczba fotonów, które może wychwycić kamera. Im noc jest ciemniejsza, tym bardziej zaszumiony jest uzyskany obraz. Podobnym przykładem jest medyczny, rentgenowski system obrazowy – ograniczeniem dominującym jest liczba fotonów (kwantów) promieniowania rentgenowskiego uderzających w detektor. Wyższy poziom promieniowania daje obrazy z mniejszym szumem kosztem silniejszego napromieniowania pacjenta.

Nasuwa się pytanie, kiedy statystyczny szum Poissona jest głównym szumem w obrazie. Otóż jest on dominujący zawsze wtedy, gdy szum pochodzący od dominującego stopnia ograniczającego (*quantum sink*) jest większy od szumów z innych źródeł w systemie, takich jak np. układy elektroniczne. Rozważmy na przykład typowy aparat fotograficzny CCD, w którym stosunek sygnału do szumu wynosi 300. To znaczy, że szum z przedwzmacniacza CCD jest 1/300 częścią pełnego zakresu sygnału. Równoważny szum byłby wytwarzany, gdyby dominujący stopień ograniczający systemu charakteryzował się liczbą 90 000 cząstek na piksel. Jeśli dominujący stopień ograniczający wyraża się mniejszą liczbą cząstek,

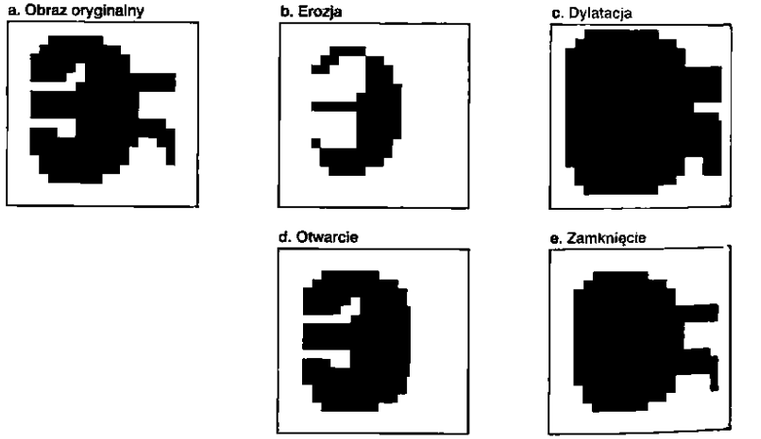
to szum o rozkładzie Poissona zdominuje system. Jeśli zaś ten stopień charakteryzuje [illegible] większą liczbą cząstek, to będzie przeważał szum przedwzmacniacza. Zgodnie z tym [illegible] szość układów CCD projektuje się z pełną pojemnością studni od 100 000 do 1 000 000 elektronów, minimalizując szum Poissona.

Morfologiczne przetwarzanie obrazu

Identyfikacja obiektów na obrazie może być zadaniem bardzo trudnym. Jednym ze sposobów uproszczenia tego problemu jest zamiana obrazu ze skalą szarości na **obraz binarny**, w którym każdy piksel może mieć wartość tylko 0 lub 1. Techniki stosowane do przetwarzania obrazów binarnych noszą nazwy: **analiza bloków** (*blob*), **analiza spójności** oraz **morfologiczne przetwarzanie obrazu** (od greckiego słowa *morphe* oznaczającego kształt lub formę). Matematyczną podstawą przetwarzania morfologicznego jest *teoria mnogości*, jednak rzadko jest konieczny taki poziom rozważań teoretycznych. Większość algorytmów morfologicznych to proste operacje logiczne realizowane ad hoc. Innymi słowy, każde zastosowanie wymaga rozwiązania wykonywanego na konkretne zamówienie, metodą prób i błędów. Więcej w tym na ogół sztuki niż nauki. Korzysta się raczej z całego mnóstwa różnych sztuczek niż ze standardowych algorytmów i formalnych reguł matematycznych. Oto kilka przykładów.

Przykład obrazu binarnego pokazano na rys. 25.10 a). Może to być np. zdjęty w podczerwieni obraz nieprzyjacielskiego czołgu, asteroida na zdjęciu kosmicznym lub guz na obrazie rentgenowskim. Każdy piksel tła jest obrazowany jako biały, a każdy piksel obiektu – jako czarny. Niejednokrotnie obrazy binarne są tworzone przez wprowadzenie pewnego progu dla obrazów o skali szarości. Wtedy piksele o wartościach większych od progu są ustawiane na wartość 1, a poniżej progu – na wartość 0. Przed operacją progową obraz o skali szarości jest zazwyczaj przetwarzany metodami liniowymi. Na przykład stosując *spłaszczanie oświetlenia* (opisane w rozdziale 24), można często poprawić jakość początkowego obrazu binarnego.

Na rys. 25.10 b), c) pokazano zmiany obrazu w wyniku dwóch najbardziej znanych operacji morfologicznych – **erozji** i **dylatacji**. W operacji erozji każdy piksel obiektu, który styka się z pikselem tła, jest zamieniany na piksel tła. W operacji dylatacji zaś każdy piksel tła stykający się z pikselem obiektu jest zamieniany na piksel obiektu. Erozja zmniejsza obiekt i może rozdzielić jeden obiekt na kilka. Dylatacja powiększa obiekt i może połączyć kilka obiektów w jeden.



Rys. 25.10. Operacje morfologiczne. W przetwarzaniu obrazów binarnych stosuje się cztery podstawowe operacje morfologiczne: erozję, dylatację, otwarcie oraz zamknięcie. Na rys. a) przedstawiono przykład obrazu binarnego, a na rys. b)÷e) – wyniki zastosowania tych operacji do obrazu z rys. a)

Jak to przedstawiono na rys. 25.10 d), **otwarcie** jest definiowane jako erozja, po której nastąpiła dylatacja. Na rys. 25.10 e) pokazano operację odwrotną, zwaną **zamknięciem**, złożoną z dylatacji i następującej po niej erozji. Z podanych na rysunkach przykładów wynika, że operacja *otwarcia* usuwa małe wyspy i wąskie nitki złożone z *pikseli obiektowych*. Operacja *zamknięcia* zaś usuwa wyspy i cienkie włókna złożone z *pikseli tła*. Te metody są przydatne do przetwarzania zaszumionych obrazów, gdzie niektóre piksele mają błędną wartość binarną. Może być z góry wiadome, że na przykład obiekt nie zawiera żadnej „dziury" lub że jego granice muszą być gładkie.

Na rys. 25.11 podano przykład przetwarzania morfologicznego. Na rys. 25.11 a) pokazano binarny obraz linii papilarnych. Są opracowane algorytmy do analizy wzoru tych linii, umożliwiające porównanie ich zgodności z odpowiednimi wzorcami w bazie danych. Jako jeden z kroków w realizacji tych algorytmów często stosuje się operację zwaną **szkieletowaniem** (*skeletonization*). Upraszcza ona obraz dzięki *usunięciu* nadmiarowych pikseli, czyli przez zamianę odpowiednich pikseli z czarnych na białe. To powoduje zamianę każdego grzbietu w linię o szerokości tylko jednego piksela.

W tablicach 25.1 i 25.2 podano program operacji szkieletowania. Chociaż obraz linii papilarnych jest binarny, to jest on jednak rejestrowany w pamięci, w której każdy piksel może mieć wartość zmieniającą się od 0 do 255. Czarny piksel jest oznaczony jako 0, a biały jako 255. Algorytm w tablicy 25.1 składa się z 6 iteracji, które wprowadzają stopniową erozję grzbietów przekształcającą je w cienkie linie. Liczba iteracji jest dobierana metodą prób i błędów. Alternatywą może być wstrzymanie programu, gdy iteracja nie wprowadza już żadnych zmian.

Podczas iteracji każdy piksel w obrazie jest szacowany w celu stwierdzenia, czy nadaje się *do usunięcia*. Piksel musi spełnić pewien zestaw kryteriów, aby mógł być zamieniony z czarnego na biały. W wierszach od 200 do 240 zapisano pętlę biegnącą przez wszystkie piksele, a w tablicy 25.2 podano podprogram realizujący oszacowanie. Jeśli rozpatrywany piksel jest nieusuwalny, to podprogram nie podejmuje żadnego działania. Jeśli piksel powinien być usunięty, to podprogram zmienia jego wartość z 0 na 1. Ten piksel pozostaje jeszcze czarny i zostaje zamieniony na biały po zakończeniu iteracji. Po oszacowaniu wszystkich pikseli, instrukcje zapisane w wierszach 260-300 zamieniają wartości zaznaczonych pikseli z 1 na 255. W wyniku tego dwustopniowego procesu grube grzbiety ulegają erozji jednakowo ze wszystkich kierunków, a nie na zasadzie zgodnej ze skanowaniem wierszy i kolumn.

Tab. 25.1.

```
'PROGRAM DO SZKIELETOWANIA
'Piksele obiektu mają wartość 0 (wyświetlane jako czarne)
'Piksele tła mają wartość 255 (wyświetlane jako białe)
'
DIM X%[149,149]                    'W X%[ , ] jest przechowywany przetwarzany obraz
'
GOSUB XXXX                         'Podprogram "mityczny" do ładowania X%[ , ]
'
FOR ITER% = 0 TO 5                 'Przejście przez sześć pętli iteracyjnych
'
  FOR R% = 1 TO 148                'Pętla przez wszystkie piksele w obrazie
  FOR C% = 1 TO 148                'Podprogram 5000 (tablica 25.2) wskazuje,
    GOSUB 5000                     'które piksele mogą być zamienione z czarnych na białe,
  NEXT C%                          'przez zaznaczenie pikseli o wartości 1.
  NEXT R%
'
  FOR R% = 0 TO 149                'Pętla przez wszystkie piksele w obrazie zamieniająca
  FOR C% = 0 TO 149                'zaznaczone piksele z czarnych na białe.
    IF X%(R%,C%) = 1 THEN X%(R%,C%) = 255
  NEXT C%
  NEXT R%
'
NEXT ITER%
'
END
```



Rys. 25.11. Szkieletowanie binarne. Binarny obraz linii papilarnych (rys. a) zawiera grzbiety o szerokości wielu pikseli, a obraz po szkieletowaniu (rys. b) – grzbiety o szerokości tylko jednego piksela

Tab. 25.2.

```
5000 , PODPROGRAM DO OKREŚLANIA, CZY PIKSEL X%[R%,C%] MOŻE BYĆ USUNIĘTY
5010 , Jeśli wszystkie cztery warunki są spełnione, to X%(R%,C%) jest ustawiany na wartość 1,
5020 , co wskazuje, że powinien on zostać usunięty na końcu iteracji.
5030 '
5040 'WARUNEK #1: Nic nie rób, jeśli piksel jest już biały
5050 IF X%(R%,C%) = 255 THEN RETURN
5060 '
5070 '
5080 'WARUNEK #2: Nic nie rób, jeśli wszystkie piksele - bliscy sąsiedzi są czarne
5090 IF X%[R%-1,C% ] <> 255 AND X%[R% ,C%+1] <> 255 AND
     X%[R%+1,C% ] <>255 AND X%[R% ,C%-1] <> 255 THEN RETURN
5100 '
5110 '
5120 'WARUNEK #3: Nic nie rób, jeśli tylko pojedynczy piksel sąsiedni jest czarny
5130 COUNT% = 0
5140 IF X%[R%  -1,C%-1   ] = 0 THEN COUNT% = COUNT% + 1
5150 IF X%[R%  -1,C% ] = 0 THEN COUNT% = COUNT% + 1
5160 IF X%[R% -1,  C%+1   ] = 0 THEN COUNT% = COUNT% + 1
5170 IF X%[R%, C%+1   ] = 0 THEN COUNT% = COUNT% + 1
5180 IF X%[R%  +1,C%+1   ] = 0 THEN COUNT% = COUNT% + 1
5190 IF X%[R%+1,   C% ] = 0 THEN COUNT% = COUNT% + 1
5200 IF X%[R%+1,   C%-1   ] = 0 THEN COUNT% = COUNT% + 1
5210 IF X%[R%, C% -1  ]       = 0 THEN COUNT% = COUNT% + 1
5220 IF COUNT% = 1 THEN RETURN
5230 '
5240 '
5250 'WARUNEK #4: Nic nie rób, jeśli piksele sąsiednie są niepołączone
5260 'Określ to obliczając liczbę przejść z czerni do bieli
5270 'poruszając się zgodnie z ruchem wskazówek zegara przez 8 sąsiednich pikseli.
5280 COUNT% = 0
5290 IF X%[R% -1,C% -1    ] = 0 AND X%[R% -1,C%     ] > 0 THEN COUNT% = COUNT% + 1
5300 IF X%[R% -1,C%    ] = 0 AND X%[R% -1,C%+1] > 0   AND X%[R% ,C% +1] > 0 THEN COUNT% = COUNT% + 1
5310 IF X%[R% -1,C%+1 ] = 0 AND X%[R%     ,C%+1] > 0  THEN COUNT% = COUNT% + 1
5320 IF X%[R%     ,C%+1    ] = 0 AND X%[R%+1,C%+1] > 0AND X%[R%+1,C% ] > 0 THEN COUNT% = COUNT% + 1
5330 IF X%[R%+1,C%+1  ] = 0 AND X%[R%+1,C%     ] > 0  THEN COUNT% = COUNT% + 1
5340 IF X%[R%+1,C%     ] = 0 AND X%[R%+1,C% -1] > 0   AND X%[R% ,C% -1] > 0 THEN COUNT% = COUNT% + 1
5350 IF X%[R%+1,C% -1 ] = 0 AND X%[R%     ,C% -1] > 0 THEN COUNT% = COUNT% + 1
5360 IF X%[R%     ,C% -1] = 0 AND X%[R% -1,C% -1] > 0   AND X%[R% -1,C% ] > 0 THEN COUNT% = COUNT% + 1
5370 IF COUNT% > 1 THEN RETURN
5380 '
5390 '
5400 'Jeśli wszystkie reguły są spełnione, zaznacz piksel, który ma być ustawiony jako biały
5405 'na końcu iteracji
5410 X%(R%,C%)= 1
5420 '
5430 RETURN
```

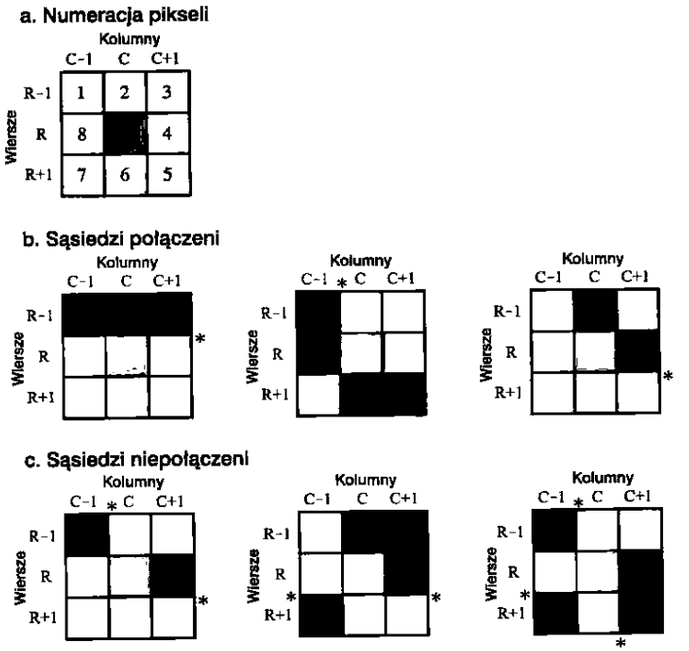

Podstawą decyzji o usunięciu piksela są cztery warunki zawarte w podprogramie w tablicy 25.2. *Wszystkie* te warunki muszą być spełnione, aby piksel został zamieniony z czarnego w biały. Pierwsze trzy warunki są dość proste, a czwarty raczej skomplikowany. Jak pokazano na rys. 25.12 a), każdy piksel o lokalizacji [R,C] ma ośmiu sąsiadów (czyli piksele sąsiednie). Cztery piksele sąsiednie w kierunku poziomym i pionowym (oznaczone 2, 4, 6, 8) są często nazywane **bliskimi sąsiadami**. Piksele sąsiadujące ukośnie (oznaczone 1, 3, 5, 7) są **dalekimi sąsiadami**. Cztery warunki są następujące:

Warunek 1: Rozpatrywany piksel musi być czarny. Jeśli piksel już jest biały, to nie jest podejmowane żadne działanie.

Warunek 2: Co najmniej jeden z pikseli będących bliskimi sąsiadami musi być biały. To gwarantuje, że erozja grubych grzbietów będzie się odbywała od zewnątrz. Inaczej mówiąc, jeśli piksel jest czarny i jest otoczony ze wszystkich stron czarnymi pikselami, to nie powinien być poddawany tej iteracji. Dlaczego są sprawdzani tylko *bliscy*, a nie *wszyscy* sąsiedzi? Odpowiedź jest prosta. Przeprowadzono próby z dwoma algorytmami i ten z bliskimi sąsiadami okazał się lepszy. Pamiętajmy, że w metodach iteracyjnych bardzo często korzysta się z metody prób i błędów w celu określenia, czy jedna metoda jest lepsza od drugiej.

Warunek 3: Piksel musi mieć więcej niż jednego czarnego sąsiada. Jeśli ma tylko jednego, to znaczy, że znajduje się na końcu linii i dlatego nie powinien zostać usunięty.

Warunek 4: Piksela nie można usunąć, gdyby miało to spowodować *rozłączenie* pikseli sąsiednich. Chodzi bowiem o to, aby każdy grzbiet zamienić w linię ciągłą, a nie w grupę rozerwanych segmentów. Jak wynika z przykładów na rys. 25.12, pojęcie *połączone* oznacza, że wszystkie czarne piksele będące sąsiadami stykają się ze sobą, *niepołączone* zaś, że czarni sąsiedzi tworzą dwie lub więcej grup.



Rys. 25.12. Sąsiadujące piksele. Piksel o lokalizacji [R,C] ma osiem pikseli sąsiednich (sąsiadów), które na rys. a) ponumerowano od 1 do 8. Na rys. b), c) pokazano przykłady sąsiednich pikseli połączonych i niepołączonych. Z tej definicji korzysta się w czwartym warunku w algorytmie szkieletowania

Algorytm do określania, czy piksele–sąsiedzi są połączone, czy niepołączone, polega na sprawdzaniu przejść z czerni do bieli sąsiadów położonych obok siebie, *w kierunku obrotu wskazówek zegara*. Na przykład jeśli piksel 1 jest czarny, a piksel 2 biały, to mamy jedno przejście typu czerń–biel. Takie jedno przejście mamy także wówczas, gdy piksel 2 jest czarny, a piksele 3 i 4 są białe. Ogółem jest osiem lokalizacji, gdzie mogą zajść przejścia czerń–biel. Aby jeszcze lepiej zilustrować tę definicję, w przykładach na rys. 25.12 b), c) zamieszczono gwiazdki przy każdym przejściu czerń–biel. Kluczem do tego algorytmu jest fakt, że gdy czas nie ma przejść czerń–biel lub jest tyko jedno takie przejście, gdy piksele–sąsiedzi są *połączone*. Więcej przejść niż jedno wskazuje, że piksele-sąsiedzi są *niepołączone*.

Jako dodatkowe przykłady przetwarzania obrazu binarnego rozważmy te typy algorytmów, które mogą być przydatne do przetwarzania obrazu po operacji jego szkieletowania. Wadą zaimplementowanego algorytmu jest pozostawienie znacznej ilości rozmyć, krótkich odrośli wystających na końcach dłuższych segmentów. Do kwestii likwidacji tych rozmyć można podejść na kilka różnych sposobów. Na przykład można zapętlić program wokół obrazu w taki sposób, aby usuwać piksele na końcach linii. Te piksele są identyfikowane przez fakt, że mają tylko jednego czarnego sąsiada. Po kilkakrotnym powtórzeniu takiej procedury usuwa się rozmycia kosztem skrócenia wszystkich prawidłowych linii. Lepsza metoda polega na utworzeniu pętli programowej, która obejmuje obraz i identyfikuje *piksele gałęziowe*, czyli piksele mające więcej niż dwóch sąsiadów. Rozpoczynając od każdego z pikseli gałęziowych, zlicza się liczby pikseli w każdej odrośli. Jeśli liczba pikseli w odrośli jest mniejsza od pewnej wartości (powiedzmy 5), to stwierdza się, że jest to rozmycie i piksele w tej gałęzi zmienia się z czarnych w białe.

Inny algorytm może zamieniać format danych z *mapy bitów* w *mapę wektorową*. To pociąga za sobą stworzenie listy grzbietów zawartych w obrazie i pikseli zawartych w każdym grzbiecie. W mapie wektorowej każdy grzbiet w obrazie linii papilarnych ma swoją unikalną dualną tożsamość, w przeciwieństwie do obrazu złożonego z wielu niezwiązanych ze sobą pikseli. Taki algorytm można zrealizować, obejmując obraz pętlą i szukając końcówek każdej linii, a więc pikseli mających tylko jednego czarnego sąsiada. Zaczynając od końcówki, trzeba prześledzić każdą linię, przechodząc od piksela do piksela łączącego. Po osiągnięciu przeciwnego końca linii, deklaruje się wszystkie prześledzone piksele jako *obiekty* pojedyncze i tak są one traktowane w realizowanych później algorytmach.

Tomografia komputerowa

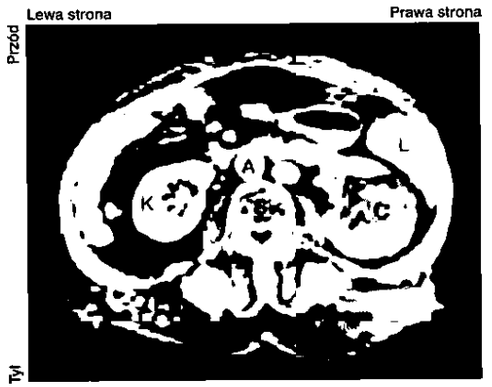
Główną trudnością przy tworzeniu obrazów rentgenowskich (lub pochodzących od innego promieniowania przenikliwego) jest *dwuwymiarowość obrazowania trójwymiarowych obiektów*. To powoduje, że struktury mogą na siebie zachodzić w uzyskanym obrazie, chociaż w obiekcie są całkowicie oddzielne. Jest to kłopotliwe zwłaszcza w diagnostyce medycznej, gdzie nakładające się obrazy wielu organów anatomicznych mogą lekarzowi utrudniać interpretację. W latach 30. usiłowano przezwyciężyć tę trudność przez przesuwanie, podczas tworzenia obrazu, źródła promieniowania rentgenowskiego i detektora. Ruchy są wykonywane w pewien skoordynowany sposób. Geometria tego ruchu powoduje, że tylko *jedna* płaszczyzna w ciele pacjenta jest zogniskowana, a struktury leżące poza tą płaszczyzną stają się rozmyte. Jest to analogiczne do aparatu fotograficznego, który zogniskowany na obiekcie w odległości np. 2 metry, daje rozmycie obiektów znajdujących się w odległości np. 1 lub 20 metrów. Te powiązane ze sobą techniki, polegające na rozmywaniu w wyniku ruchu, są teraz objęte ogólną nazwą **tomografii klasycznej**. Słowo *tomografia* znaczy dosłownie *obraz płaszczyzny*.

Prace nad tomografią klasyczną prowadzono przez ponad 50 lat, ale mimo to jest ona rzadko stosowana. Przyczyną jest zasadnicza wada tej metody – zakłócające obrazy nie są *usuwane*, lecz tylko *rozmywane*. Uzyskiwane obrazy nie mają zwykle praktycznego zastosowania z powodu zbyt słabej jakości. Długo poszukiwanym rozwiązaniem okazał się system mający tworzyć obraz reprezentujący przekrój 2D (dwuwymiarowy) obiektu 3D (trójwymiarowego) *bez żadnych* zakłóceń pochodzących od innych struktur w obiekcie 3D.

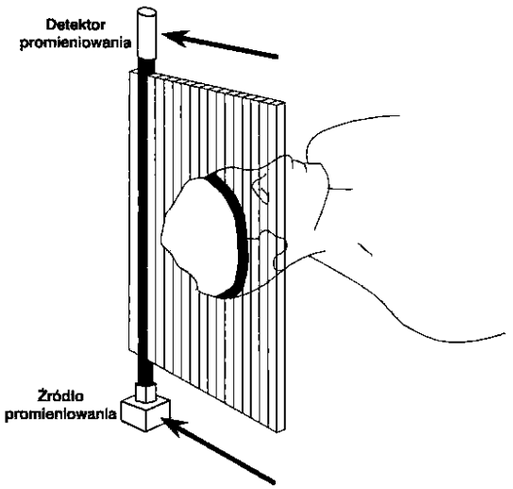
Problem rozwiązano na początku lat 70. wprowadzając technikę zwaną **tomografią komputerową** (CT – *Computer Tomography*). Ta tomografia zrewolucjonizowała medyczne metody rentgenowskie dzięki swym bezprecedensowym możliwościom wizualizacji anatomicznej budowy ciała. Na rys. 25.13 przedstawiono typowy obraz medyczny uzyskany metodą tomografii komputerowej. Początkowo tomografia komputerowa była wprowadzona na rynek pod nazwą *komputerowa tomografia osiowa* lub *skaner CAT* (*Computer Axial Tomography Scanner*). W środowisku medycznym zarzucono już stosowanie tych nazw, lecz ciągle można je jeszcze usłyszeć w języku potocznym.

Na rys. 25.14 przedstawiono prostą geometrię pozyskiwania w tomografii komputerowej widoku przekroju przechodzącego przez środek głowy. Wąska wiązka promieniowania rentgenowskiego przechodzi od źródła do detektora promieniowania. Wartość zmierzona w detektorze jest więc zależna od całkowitej ilości substancji znajdującej się *gdziekolwiek* na drodze wiązki. Substancje takie, jak kości lub zęby pochłaniają więcej promieniowania rentgenowskiego, co powoduje, że w detektorze uzyskuje się sygnał mniejszy niż w przypadku tkanek miękkich i tłuszczu. Jak to widać na rysunku, zespół źródła i detektora są przesuwane, aby zarejestrować **widok** pod pewnym określonym kątem. Na rysunku pokazano tylko jeden pobrany widok; pełne skanowanie obejmuje od 300 do 1000 widoków pobranych przy obrocie o przyrostach od 0,3° do 1,0°. Takie skanowanie realizuje się, montując źródło i detektor na obrotowej dźwigni poruszającej się wokół pacjenta. Najważniejszą cechą zbierania danych w tomografii komputerowej jest to, że promienie rentgenowskie przechodzą *tylko* przez właśnie badany poprzeczny wycinek ciała. Inaczej jest w tomografii klasycznej, gdzie promienie Roentgena przechodzą przez struktury, które próbujemy wytłumić w końcowym obrazie. W tomografii komputerowej nie dopuszcza się do tego, aby informacja z nieistotnych dla badania miejsc w ogóle się dostawała do zbieranych danych.

Przed rekonstrukcją obrazu trzeba na ogół wykonać kilka etapów przetwarzania wstępnego. Na przykład wynik każdego pomiaru promieniowania rentgenowskiego powinien zostać zlogarytmowany, gdyż natężenie tego promieniowania przy przechodzeniu przez jakiś materiał zmniejsza się wykładniczo. Złogarytmowanie daje sygnał zależny liniowo od właściwości mierzonego materiału. Inne etapy przetwarzania wstępnego służą do kompensacji *polichromatyczności* stosowanego promieniowania (mającego więcej niż jedną energię) oraz detektorów *wieloelementowych* (w przeciwieństwie do pojedynczego elementu na rys. 25.14). Chociaż te operacje są istotne w całej metodzie, nie są jednak związane z algorytmem rekonstrukcyjnym i nie będziemy więcej o nich mówić.



Rys. 25.13. Obraz uzyskany metodą tomografii komputerowej. Jest to widok przekroju (warstwy) brzucha człowieka na wysokości pępka. Jest widocznych wiele organów, takich jak wątroba (oznaczona L), nerka (K), aorta (A), kręgosłup (S), torbiel pokrywająca prawą nerkę (C). Tomografia komputerowa wizualizuje anatomię wewnętrzną znacznie lepiej niż konwencjonalne prześwietlenia rentgenowskie



Rys. 25.14. Zbieranie danych w tomografii komputerowej. W prostym systemie wąska wiązka promieniowania rentgenowskiego przechodzi przez ciało pacjenta od źródła do detektora. Źródło i detektor są potem przesuwane, aby można było uzyskać pełny obraz. Pozostałe widoki otrzymuje się, obracając źródło i detektor z przyrostami kątowymi o ok. 1° i powtarzając proces przesuwania

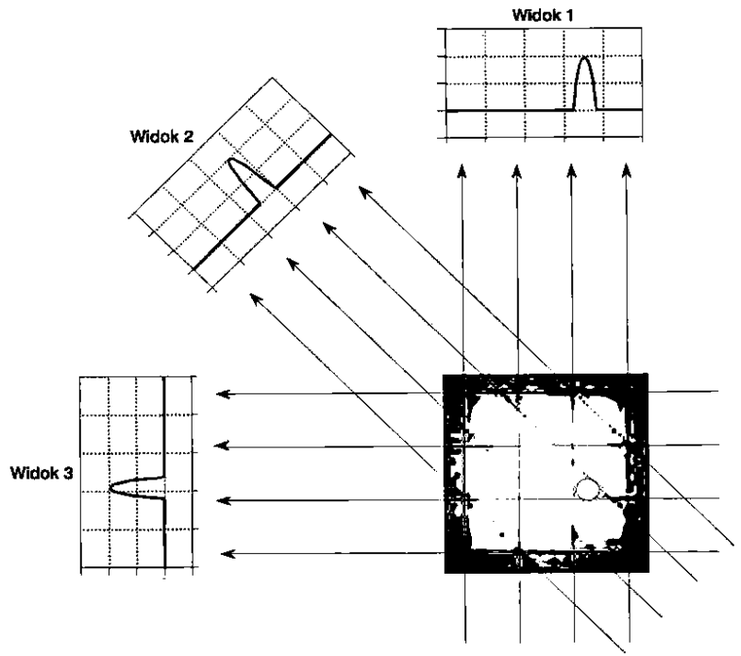
Na rys. 25.15 zilustrowano zależność między zmierzonymi widokami i odpowiadającym mu obrazem. Każda próbka pobierana w systemie tomografii komputerowej jest równa sumie wartości obrazu wzdłuż promienia wskazującego tę próbkę. Na przykład widok 1 znaleziono, sumując piksele w każdej kolumnie. Widok 3 zaś jest znajdowany przez sumowanie w wierszach. Inne widoki, takie jak widok 2, uzyskano, sumując piksele wzdłuż promieni biegnących pod jakimś określonym kątem.

Istnieją cztery główne sposoby obliczania obrazu przekroju, gdy ma się dany zestaw jego widoków. Są to tzw. **algorytmy rekonstruujące tomografii komputerowej**. Pierwszy sposób jest zupełnie nieprzydatny w praktyce, lecz daje lepsze zrozumienie problemu. Polega on na rozwiązywaniu wielu **równoważnych równań liniowych**. Dla każdego pomiaru można napisać jedno równanie. To znaczy, że określona próbka w określonym profilu jest sumą określonej grupy pikseli w obrazie. Aby obliczyć N^2 nieznanych zmiennych (tj. wartości pikseli obrazu), trzeba mieć N^2 niezależnych równań, a więc N^2 pomiarów. Większość komputerowych skanerów tomograficznych pobiera o ok. 50% próbek więcej, niż wynika z ostrych wymagań tej analizy. Na przykład w celu zrekonstruowania obrazu 512 × 512 system może pobrać 700 widoków zawierających po 600 próbek. Przez takie „*przedobrzenie*" problemu uzyskuje się zmniejszenie szumów i artefaktów. Trudności w tej pierwszej metodzie rekonstrukcji obrazu tomografii komputerowej są związane z czasem obliczeń. Rozwiązywanie kilkuset równoważnych równań liniowych to paskudne zadanie.

W drugiej metodzie rekonstrukcji w tomografii komputerowej stosuje się technikę **iteracyjną**, aby małymi krokami obliczyć końcowy obraz. Jest kilka odmian tej metody: technika rekonstrukcji algebraicznej (ART – *Algebraic Reconstruction Technique*), technika równoczesnej rekonstrukcji iteracyjnej (SIRT – *Simultaneous Iterative Reconstructiom Technique*) oraz technika iteracji metodą najmniejszych kwadratów (ILST – *Iterative Least Squares Technique*). Różnice między tymi metodami leżą w sposobie wykonywania kolejnych korekcji, a więc – odpowiednio – promień po promieniu, piksel po pikselu, następuje jedna korekcja całego zestawu danych. Jako przykład tych technik omówimy ART.

Na początku algorytmu ART wszystkie piksele w matrycy obrazu są ustawiane na jakąś dowolną wartość. Następnie stosuje się procedurę iteracyjną, stopniowo zmieniając matrycę obrazową tak, aby odpowiadała ona profilom. Cykl iteracyjny zawiera pętlę przebiegającą przez każdy punkt w zmierzonych danych. Przy każdej wartości zmierzonej stawia się pytanie: *Jak można zmienić wartości pikseli w matrycy, aby stały się zgodne z tym konkretnym pomiarem*? Inaczej mówiąc, zmierzona próbka jest porównywana z sumą pikseli obrazowych wzdłuż promienia wskazującego próbkę. Jeśli suma wzdłuż promienia jest mniejsza niż zmierzona próbka, to wartości wszystkich pikseli wzdłuż tego promienia ulegają zwiększeniu. Natomiast są one zmniejszane, jeśli suma pikseli wzdłuż promienia jest większa od zmierzonej próbki. Po pierwszym pełnym cyklu iteracyjnym ciągle jeszcze pozostanie pewien błąd między sumami wzdłuż promienia i wartościami zmierzonymi. Przyczyna jest taka, że zmiany wprowadzone do jednego pomiaru niszczą wszystkie korekcje wprowadzone poprzednio. Zasada polega na tym, że błędy stają się coraz mniejsze w miarę powtarzanych iteracji, aż obraz osiągnie właściwą postać.

Metody iteracyjne są – ogólnie biorąc – wolne, lecz przydatne, gdy lepsze algorytmy nie są dostępne. Faktycznie, technikę ART stosowano w pierwszym komercyjnym skanerze CT z 1972 roku – EMI Mark I. Do metod iteracyjnych powrócimy w następnym rozdziale przy omawianiu sieci neuronowych. Rozwój trzeciego i czwartego spośród wspomnianych czterech sposobów obliczania obrazu przekroju spowodował, że prawie całkowicie zastąpiły one techniki iteracyjne w komercyjnej aparaturze do tomografii komputerowej.

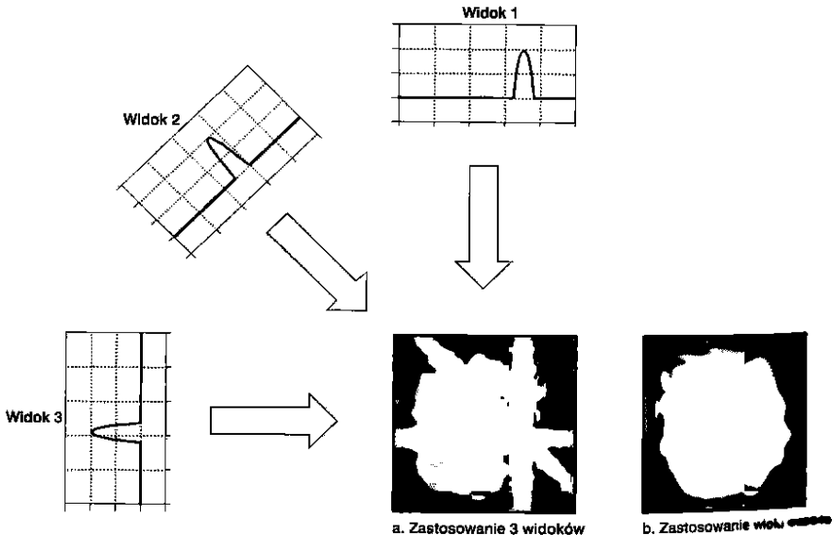


Rys. 25.15. Widoki uzyskiwane w tomografii komputerowej. Na podstawie uzyskanego zestawu widoków rekonstruuje się odpowiadający im obraz. Każda próbka w widoku jest równa sumie wartości obrazu wzdłuż promienia wskazującego na tę próbkę. W tym przykładzie obraz jest obiektem typu „okrągła czapka" otoczonym przez zera. Tutaj pokazano tylko trzy widoki, a w typowym skanowaniu w tomografii komputerowej stosuje się setki widoków zdejmowanych pod nieco różniącymi się kątami

Dwa ostatnie spośród czterech algorytmów rekonstruujących oparto na formalnych... tematycznych rozwiązaniach problemu. Są to eleganckie przykłady DSP. Trzecia metoda nosi nazwę **filtrowanej projekcji wstecznej**. Jest to modyfikacja starszej techniki, zwanej **projekcją wsteczną** lub **prostą projekcją wsteczną**. Prosta projekcja wsteczna (rys. 25.16) jest rozwiązaniem mało wyszukanym, opartym na zdrowym rozsądku. Pojedynczy punkt jest rzutowana wstecznie przez ustawienie na jednakową wartość wszystkich pikseli w promienia wskazującego próbkę. Mniej technicznym językiem można powiedzieć, że projekcja jest tworzona przez *przemiatanie* każdego widoku wstecz przez obraz w kierunku, z którego był on pierwotnie pobrany. Końcowy, rzutowany wstecznie obraz jest więc prawie jako suma wszystkich widoków rzutowanych wstecznie.

Projekcja wsteczna jest wprawdzie koncepcyjnie prosta, lecz jednak nie rozwiązuje problemu w prawidłowy sposób. Uzyskany w wyniku projekcji wstecznej obraz jest bardzo *rozmyty*. Pojedynczy punkt *prawdziwego* obrazu jest rekonstruowany jako obszar kołowy o jaskrawości zmniejszającej się wraz ze wzrostem odległości od środka koła. W kategoriach bardziej formalnych trzeba stwierdzić, że funkcja rozmycia punktu PSF projekcji wstecznej jest kołowo symetryczna i zmniejsza się odwrotnie proporcjonalnie do promienia koła.

Filtrowana projekcja wsteczna jest techniką korygującą rozmycia powstające przy prostej projekcji wstecznej. Każdy widok, jak to pokazano na rys. 25.17, jest przed projekcją wsteczną *filtrowany*, aby przeciwdziałać rozmywającej funkcji PSF. Każdy z jednowymiarowych widoków jest więc splatany z jednowymiarową odpowiedzią impulsową filtru w celu utworzenia zestawu *widoków przefiltrowanych*. Te widoki są następnie rzutowane wstecznie, dając zrekonstruowany obraz będący bliskim przybliżeniem obrazu „prawidłowego". Obraz uzyskiwany przez filtrowaną projekcję wsteczną jest rzeczywiście *identyczny* z obrazem „prawidłowym" wówczas, gdy tak liczba widoków, jak i liczba punktów w widoku są *nieskończenie duże*.

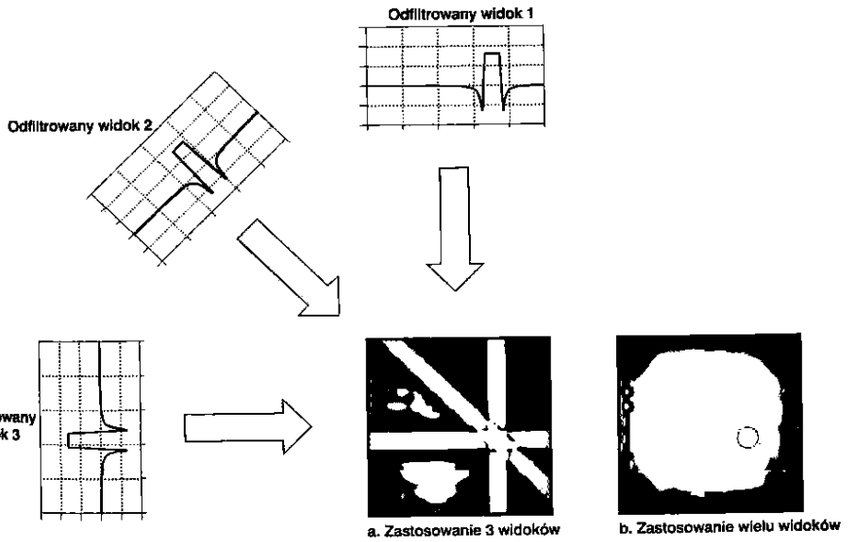


Rys. 25.16. Projekcja wsteczna. Następuje tu rekonstrukcja obrazu przez pobieranie każdego obrazu i przemiatanie go wzdłuż ścieżki, z której poprzednio był pobrany. Uzyskany obraz jest rozmytą wersją obrazu prawidłowego

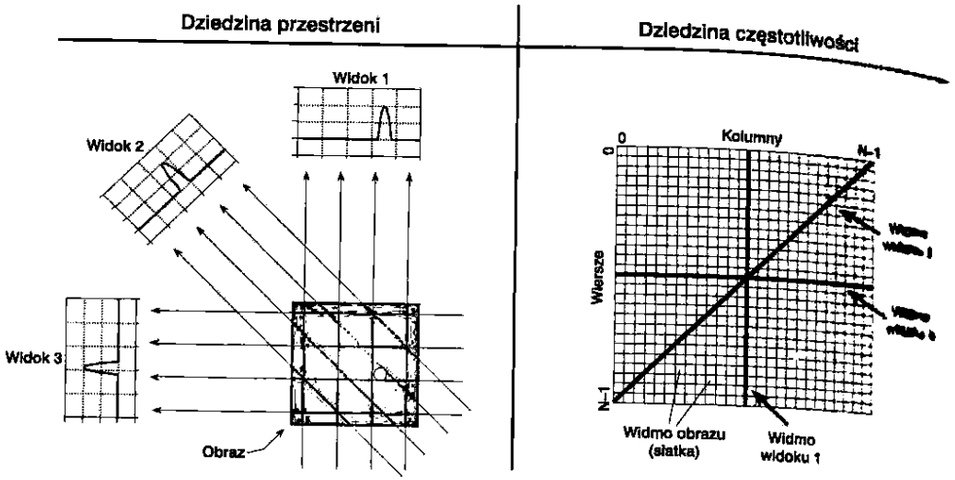
Odpowiedź impulsowa filtru stosowana w tej technice będzie omówiona nieco dalej. Na razie zwróćcie uwagę, jak filtracja zmieniła profile. W tym przykładzie obraz jest jednolitym białym kołem otoczonym ciemnym tłem (funkcja typu „okrągła czapka"). Każdy z pobranych widoków ma płaskie tło z okrągłym obszarem reprezentującym białe koło. Filtracja wprowadza do widoków dwie znaczące zmiany. Po pierwsze, wierzchołek impulsu ulega spłaszczeniu, dzięki czemu w obrazie końcowym z projekcją wsteczną poziom sygnału wewnątrz koła jest *równomierny*. Po drugie, zostały wprowadzone ujemne szpilki po bokach impulsu. Przy projekcji wstecznej te ujemne obszary przeciwdziałają rozmyciom.

Czwarta metoda jest nazywana **rekonstrukcją Fouriera**. Do rekonstrukcji komputerowego obrazu tomograficznego w dziedzinie przestrzeni jest potrzebna zależność między dwuwymiarowym obrazem a jego zestawem widoków jednowymiarowych. Problem może być badany w dziedzinie częstotliwości po przeprowadzeniu dwuwymiarowego przekształcenia Fouriera obrazu i jednowymiarowego przekształcenia Fouriera każdego widoku. Okazuje się, że zależność między obrazem i jego widokami jest znacznie prostsza w dziedzinie częstotliwości niż w dziedzinie przestrzeni. Analiza tego problemu w dziedzinie częstotliwości jest kamieniem milowym w rozwoju tomografii komputerowej i nosi nazwę **twierdzenia Fouriera o projekcji** (*Fourier slice theorem*).

Na rys. 25.18 zilustrowano to zagadnienie w dziedzinach przestrzeni i częstotliwości. W dziedzinie przestrzeni każdy widok jest określany przez scałkowanie obrazu wzdłuż promieni biegnących pod pewnym określonym kątem. W dziedzinie częstotliwości widmo obrazu jest na rysunku wyrażone przez siatkę dwuwymiarową. Widmo każdego widoku (sygnał jednowymiarowy) jest reprezentowane przez ciemną linię nałożoną na siatkę. Jak to wyrażono przez umiejscowienie linii na siatce, zgodnie z twierdzeniem Fouriera o przekrojach widmo widoku jest identyczne z widmem wzdłuż linii (przekroju) poprzez widmo obrazu. Na przykład widmo widoku 1 jest takie samo, jak środkowa kolumna w widmie obrazu, a widmo widoku 3 jest takie samo, jak środkowy wiersz widma obrazu. Zauważcie, że widmo każdego widoku umiejscowiono na siatce pod takim samym kątem, pod jakim widok został pierwotnie pobrany. Te wszystkie widma częstotliwościowe zawierają częstotliwości ujemne i są przedstawiane z częstotliwości zerową pośrodku.



Rys. 25.17. Filtrowana projekcja wsteczna. W tej metodzie rekonstruuje się obraz, filtrując każdy widok przed projekcją wsteczną. W ten sposób usuwa się rozmycia widoczne w przypadku prostej projekcji wstecznej. Filtrowana projekcja wsteczna jest najczęściej stosowanym algorytmem w systemach tomografii komputerowej



Rys. 25.18. Twierdzenie Fouriera o projekcji. Opisuje ono zależność między obrazem i jego widokami w dziedzinie częstotliwości. W dziedzinie przestrzeni każdy widok jest wyznaczany przez całkowanie obrazu wzdłuż promieni przebiegających pod określonymi kątami. W dziedzinie częstotliwości widmo każdego widoku jest jednowymiarowym „wycinkiem" dwuwymiarowego widma obrazu

Fourierowska rekonstrukcja obrazu tomografii komputerowej wymaga wykonania trzech kroków. Po pierwsze, przeprowadza się jednowymiarowe przekształcenie FFT każdego widoku. Po drugie, te widma widoków wykorzystuje się do obliczania dwuwymiarowego częstotliwościowego widma obrazu, jak to wynika z twierdzenia Fouriera o przekroju. Ponieważ widma widoków są rozmieszczane *promieniowo*, a prawidłowe widmo obrazu jest rozmieszczone *prostokątnie*, więc do wykonania konwersji jest konieczna procedura interpolacyjna. Po trzecie, wykonuje się odwrotne przekształcenie FFT widma obrazu dla uzyskania obrazu zrekonstruowanego.

Konwersja „promieniowe na prostokątne" jest też kluczem do zrozumienia filtrowanej projekcji wstecznej. Rozmieszczenie promieniowe jest widmem obrazu rzutowanego wstecznie, podczas gdy rozmieszczenie prostokątne jest widmem obrazu prawidłowego. Jeśli porównamy jeden mały obszar widma promieniowego z odpowiadającym mu obszarem siatki prostokątnej, to stwierdzimy, że wartości próbek są identyczne. Mają one jednak różne *gęstości próbkowania*. W widmie prawidłowym punkty są rozmieszczone równomiernie, jak to przedstawiono przez jednakowe odstępy w siatce prostokątnej. Widmo rzutowane wstecz ma zaś większą gęstość próbkowania w pobliżu środka z powodu promienistego rozmieszczenia. Można by powiedzieć, że odległości między szprychami koła są najmniejsze w pobliżu piasty. Ten fakt nie ma żadnego wpływu na rekonstrukcję fourierowską, gdyż interpolację wykonuje się na podstawie *wartości* najbliższych pikseli–sąsiadów, a nie ich *gęstości*.

W projekcji wstecznej filtracja likwiduje tę nierówną gęstość próbkowania. W szczególności odpowiedź częstotliwościowa filtru musi być *odwrotnością* gęstości próbkowania. Ponieważ rzutowane wstecznie widmo ma gęstość $1/f$, więc właściwy filtr powinien mieć odpowiedź częstotliwościową $H[f] = f$. Tę odpowiedź częstotliwościową pokazano na rys. 25.19 a). Następnie wyznacza się odpowiedź impulsową filtru przez wykonanie odwrotnego przekształcenia Fouriera (rys. 25.19 b). Odpowiedź impulsowa jest opisana wzorami:

$$h[0]=1$$

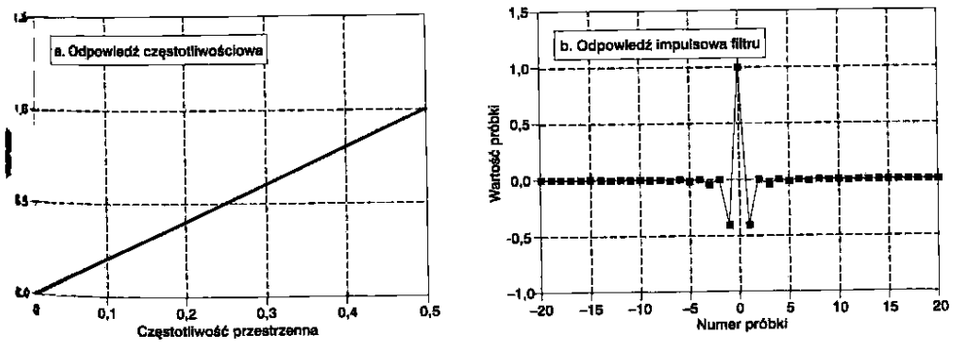
$$h[k]=0 \qquad \text{dla parzystych wartości k}$$

$$h[k]=\frac{-4/\pi^2}{k^2} \qquad \text{dla nieparzystych wartości k}$$

Wzór 25.2. Odpowiedź impulsowa filtru stosowana do filtrowanej projekcji wstecznej. Wykres tej odpowiedzi pokazano na rys. 25.19 b)

Przed rozstaniem się z tematem tomografii komputerowej trzeba wspomnieć, że w medycynie istnieje kilka innych, podobnych technik obrazowych. W tych wszystkich technikach jest szeroko stosowane DSP. W **pozytonowej tomografii emisyjnej PET** (*Positron Emission Tomography*) wstrzykuje się pacjentowi substancję promieniotwórczą o niewielkiej aktywności, emitującą *pozytony*. Natychmiast po emisji pozyton anihiluje z elektronem, wysyłając dwa fotony promieniowania gamma, które opuszczają ciało pacjenta w dwóch dokładnie przeciwnych kierunkach. Detektory promieniowania rozmieszczone wokół pacjenta wyszukują te pary fotonów i na tej podstawie następuje identyfikacja położenia *linii*, wzdłuż której przebiegają fotony gamma. Ponieważ punkt, z którego zostały wysłane fotony, musi leżeć gdzieś na tej linii, więc można zastosować algorytm rekonstruujący, podobny do używanego w tomografii komputerowej. W ten sposób uzyskuje się obraz podobny jak w tomografii komputerowej, z tą różnicą, że *jaskrawość* jest zależna od ilości substancji promieniotwórczej znajdującej się w danym miejscu ciała pacjenta. Unikatową zaletą obrazowania PET jest możliwość włączania związku promieniotwórczego do różnych substancji pełniących ważne funkcje w procesach życiowych, np. do glukozy. Wtedy zrekonstruowany obraz jest związany z koncentracją takiej substancji biologicznej. Dzięki temu staje się możliwe obrazowanie procesów *fizjologicznych*, a nie tylko *anatomii*. Można na przykład uzyskiwać obrazy pokazujące, jakie funkcje mózgu są realizowane w poszczególnych jego częściach.

Bardziej bezpośrednim konkurentem tomografii komputerowej jest **obrazowanie za pomocą rezonansu magnetycznego MRI** (*Magnetic Resonance Imaging*), stosowane obecnie w większości dużych szpitali. Ta technika była pierwotnie opracowana pod nazwą **jądrowego rezonansu magnetycznego NMR** (*Nuclear Magnetic Resonance*). Zmiana nazwy nastąpiła ze względu na promocję metody. Niektóre władze lokalne nie chciały bowiem, aby w ich społecznościach stosowano jakiekolwiek urządzenia *jądrowe*. Często okazywało się niemożliwe wytłumaczenie ludziom, że termin *jądrowy* odnosi się w tym przypadku do faktu, że każdy atom zawiera *jądro*. Skanowanie MRI przeprowadza się, umieszczając pacjenta wewnątrz silnego magnesu. Fale elektromagnetyczne wielkiej częstotliwości w połączeniu z polem magnetycznym wywołują rezonans jąder w wybranych atomach w ciele pacjenta,

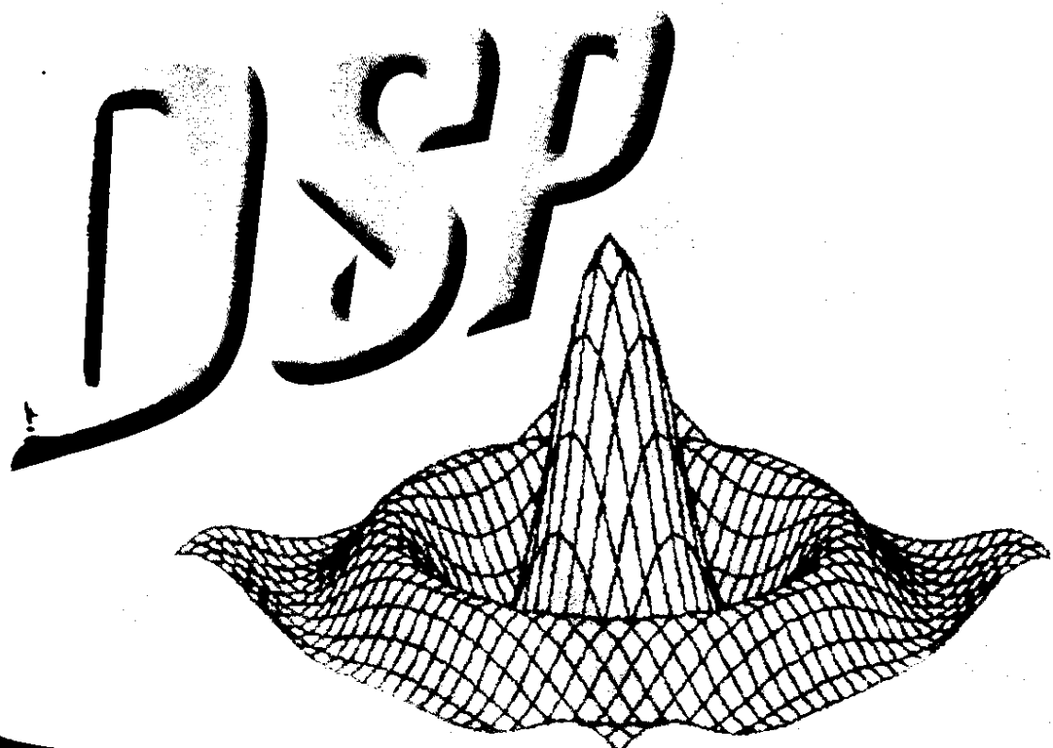


Rys. 25.19. Filtr projekcji wstecznej. Odpowiedź częstotliwościową filtru projekcji wstecznej pokazano na rys. a), a odpowiadającą jej odpowiedź impulsową na rys. b). Wartości odpowiedzi impulsowej obliczono z wzoru 25.2

co z kolei powoduje emisję wtórnych fal wielkiej częstotliwości. Te fale są przekształcane do postaci cyfrowej, tworząc zestaw danych przetwarzanych przez rekonstruujące algorytmy MRI. Wynikiem jest zestaw obrazów, bardzo podobnych do otrzymywanych w tomografii komputerowej. Obrazowanie MRI ma liczne zalety: dobrą rozróżnialność tkanek miękkich, łatwy wybór różnych przekrojów i niestosowanie potencjalnie szkodliwego promieniowania rentgenowskiego. Są i cechy ujemne MRI – koszty większe niż tomografii komputerowej oraz słabe obrazowanie kości i innych tkanek twardych. Tomografia komputerowa i obrazowanie za pomocą pola magnetycznego będą z pewnością filarami obrazowania medycznego przez wiele najbliższych lat.

26

Sieci neuronowe (i więcej!)



Podstawą tradycyjnego DSP są *algorytmy* przekształcające dane z jednej postaci w inną przez procedury wykonywane krok po kroku. Do realizacji większości tych metod konieczne są także pewne *parametry*. Na przykład w filtrach rekursywnych stosuje się współczynniki rekursywne, wykrywanie cech może być realizowane przez korelację i *progi*, a wyświetlanie obrazu zależy od ustawień *jaskrawości* i *kontrastu* itd. Algorytmy opisują, co ma być wykonywane, a parametry dostarczają wzorca do oceny danych. Właściwy wybór parametrów jest często ważniejszy od samego algorytmu. W sieciach neuronowych ta zasada jest doprowadzona do skrajności przez zastosowanie bardzo prostych algorytmów i wielu wysoce zoptymalizowanych parametrów. Jest to rewolucyjne odejście od tradycyjnej nauki i techniki – od logiki matematycznej i teorii popartej eksperymentem. W sieciach neuronowych te sposoby rozwiązywania problemów są zastępowane przez metodę prób i błędów, rozwiązania pragmatyczne oraz metodologię typu „to działa lepiej niż tamto". W tym rozdziale przedstawiono różne kwestie dotyczące wyboru parametrów tak w sieciach neuronowych, jak i w tradycyjnych algorytmach DSP.

Wykrywanie celu

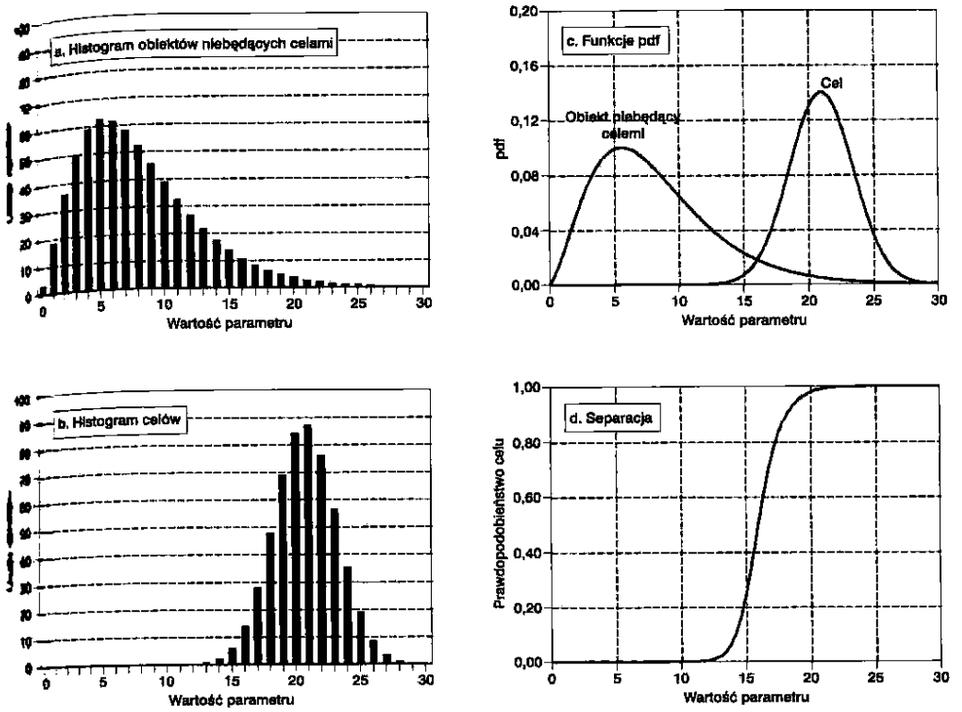
Naukowcy i inżynierowie często chcą się dowiedzieć, czy istnieje dany obiekt lub warunek. Na przykład geofizycy poszukują ropy naftowej, lekarze badają pacjentów, szukając chorób, astronomowie badają Wszechświat w poszukiwaniu inteligencji pozaziemskiej itd. Te kwestie wymagają zwykle porównania zebranych danych z jakimś progiem. Jeśli próg jest przekroczony, to **cel** (poszukiwany obiekt lub warunek) uważa się za istniejący.

Załóżmy na przykład, że wynaleźliście urządzenie do wykrywania raka. Aparat jest przesuwany po ciele pacjenta i na ekranie monitora wyskakuje liczba od 0 do 30. Małe liczby dotyczą pacjentów zdrowych, a duże oznaczają obecność tkanki nowotworowej. Stwierdzacie, że urządzenie działa dość dobrze, lecz nie jest jednak doskonałe i od czasu do czasu popełnia błędy. Można więc postawić pytanie, jak stosować taki system z pożytkiem dla badanego pacjenta.

Sposób systematycznej analizy tej sytuacji przedstawiono na rys. 26.1. Przyjmijmy, że urządzenie przetestowano na dwóch grupach: kilkuset ochotników, o których wiadomo, że są zdrowi (nie są więc poszukiwanym celem) i kilkuset ochotników, o których wiadomo, że są chorzy na raka (są celem). Wyniki testów w postaci histogramów pokazano na rys. 26.1 a), b). Osobników zdrowych, ogólnie biorąc, charakteryzuje liczba mniejsza niż charakteryzująca tych mających raka (co jest prawidłowe), lecz oba rozkłady w pewnym stopniu zachodzą na siebie (co jest nieprawidłowe).

Jak już powiedziano w rozdziale 2, histogram może być traktowany jako oszacowanie funkcji **rozkładu prawdopodobieństwa (pdf)** – rys. 26.1 c). Wyobraźmy sobie na przykład, że urządzeniem przebadano wybranego losowo, zdrowego osobnika. Zgodnie z rys. 26.1 c) jest szansa ok. 8%, że wynik będzie równy 3, a ok. 1%, że będzie 18 itd. (W tym przykładzie nie określono, czy chodzi o liczby rzeczywiste wymagane w *pdf*, czy całkowite wymagane w *pmf*. Tym się nie przejmujcie, to tutaj nie jest ważne).

Pomyślcie teraz, co się dzieje, gdy urządzeniem przebada się pacjenta o nieznanym stanie zdrowia. Jaki wniosek wyciągniecie, gdy osoba, której nigdy przedtem nie widzieliście, uzyska w badaniu wynik na przykład równy 15? Czy pacjent ma raka, czy nie? Prawdopodobieństwo, że pacjent o wyniku 15 jest zdrowy, wynosi – jak wiemy – 2,1%. Jednocześnie, z prawdopodobieństwem 0,7% pacjent chory na raka może uzyskać wynik równy 15. Jeśli nie są dostępne żadne inne informacje, to wywnioskujemy, iż trzy razy bardziej prawdopodobne jest, że pacjent jest zdrowy, niż że jest chory. A zatem wynik testu równy 15 prowadzi do wniosku, że prawdopodobieństwo przynależności pacjenta do grupy docelowej wynosi 25%. Tę metodę można uogólnić, tworząc krzywą pokazaną na rys. 26.1 d), wyrażającą prawdopodobieństwo raka na podstawie liczb uzyskiwanych z urządzenia [matematycznie: $pdf_1 / (pdf_1 + pdf_2)$]



Rys. 26.1. Prawdopodobieństwo wykrycia celu. Na rys. a), b) pokazano histogramy grupy będącej celem i grupy niebędącej celem w odniesieniu do pewnej wartości parametru. Na podstawie tych histogramów można oszacować funkcje gęstości prawdopodobieństwa obu grup (rys. c). Korzystając tylko z tej informacji, można obliczyć krzywą pokazaną na rys. d) wyznaczającą prawdopodobieństwo znalezienia celu na podstawie określonej wartości parametru

Gdybyśmy w tym miejscu zakończyli analizę, to popełnilibyśmy jeden z najczęściej spotykanych (i poważnych) błędów w wykrywaniu celu. Aby nadać sens wykresowi na rys. 26.1 d), trzeba na ogół uwzględnić inne źródło informacji. Taką informacją jest względna liczba celów w stosunku do liczby obiektów niebędących celem, w badanej populacji. Możemy na przykład stwierdzić, że tylko jedna na tysiąc osoba ma raka, którego próbujemy wykryć. Aby włączyć informację do analizy, tak dobiera się amplitudę wykresu *pdf* na rys. 26.1 c) dotyczącego obiektów niebędących poszukiwanym celem, żeby powierzchnia pod krzywą była 0,999. Amplituda wykresu *pdf* dotyczącego poszukiwanego celu zaś jest tak dobierana, żeby powierzchnia pod krzywą była 0,001. Następnie, podobnie jak poprzednio, oblicza się krzywą z rys. 26.1 d), uzyskując wartości prawdopodobieństwa, że pacjent ma raka.

Pominięcie tej dodatkowej informacji jest poważnym błędem, gdyż w istotny sposób wpływa na interpretację wyników testu. Krzywa na rys. 26.1 d) zmienia się bardzo znacznie w wyniku uwzględnienia informacji o rozpowszechnieniu raka. Jeśli na przykład ułamek populacji mającej raka jest 0,001, to wynik testu równy 15 daje prawdopodobieństwo, że pacjent ma raka, równe tylko 0,025%. Ten wynik bardzo się różni od prawdopodobieństwa 25% wyznaczonego przy wykorzystaniu tylko wyników uzyskanych z urządzenia.

Ta metoda przetwarzania uzyskanej liczby na wartość prawdopodobieństwa może być pożyteczna dla zrozumienia problemu, lecz nie jest głównym sposobem realizacji wykrywania celu. W większości zastosowań jest wymagana decyzja typu tak/nie dotycząca istnienia lub

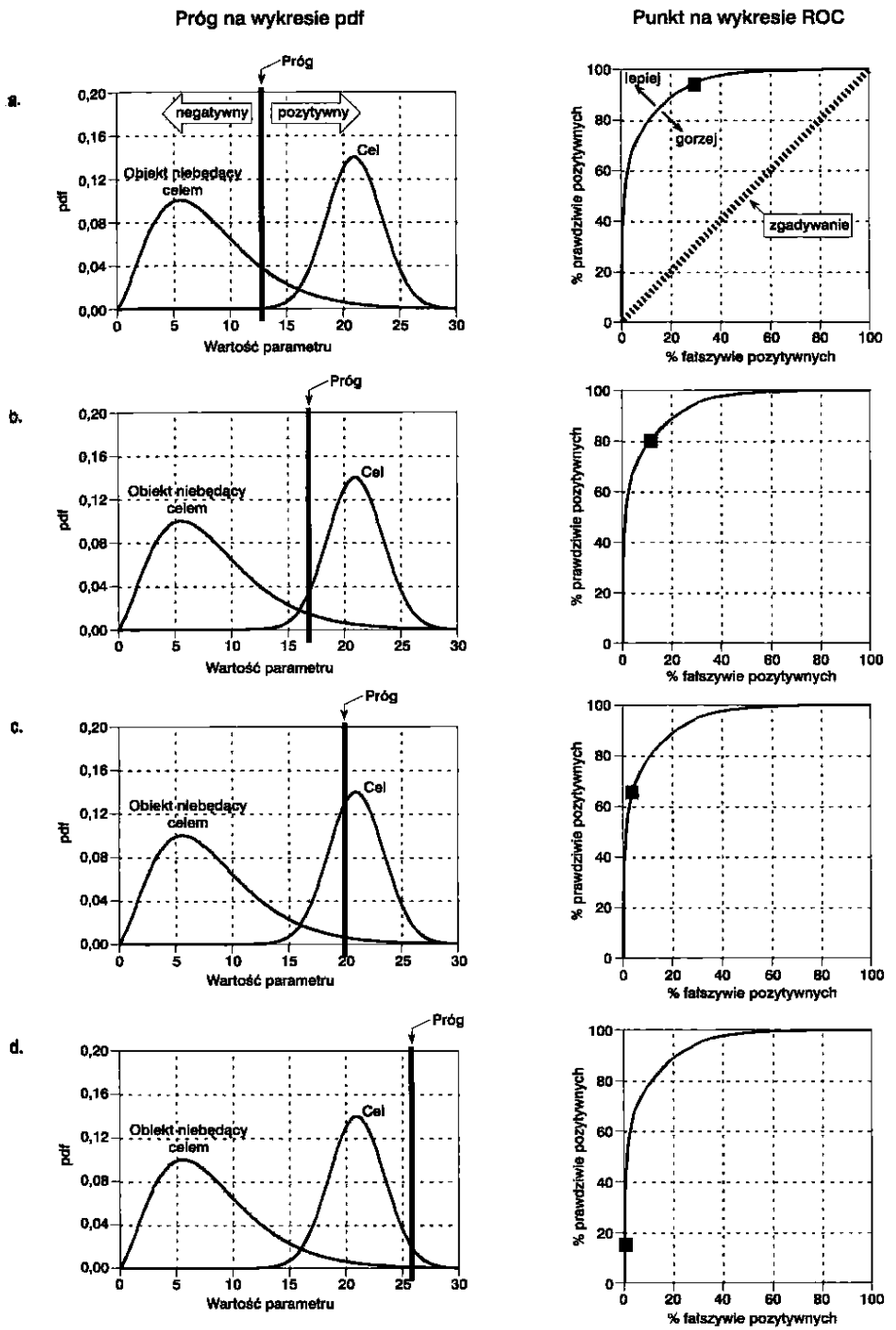
nieistnienia wykrywanego celu, gdyż wynik *tak* ma powodować jakieś działanie, a *nie* – inne. Takie decyzje uzyskuje się, porównując uzyskaną wartość wyjściową testu z pewnym progiem. Jeśli wartość jest większa od progu, to mówi się, że test jest **pozytywny**, a więc cel istnieje. Jeśli zaś wartość jest mniejsza od progu, to test jest **negatywny**, co wskazuje, że celu nie ma. W omawianym przykładzie poszukiwania raka negatywny wynik testu powoduje odesłanie pacjenta do domu ze stwierdzeniem, że jest zdrowy. W przypadku testu pozytywnego prowadzi się badania dodatkowe, jak na przykład pobranie próbki tkanki igłą do biopsji.

Niektóre wyniki testów nie będą prawidłowe, bo rozkłady prawdopodobieństwa dotyczące istnienie lub nieistnienia celu nakładają się na siebie. Tak więc niektórzy pacjenci odesłani do domu w rzeczywistości mają raka, a niektórzy wysłani na dodatkowe badania są zdrowi. W terminologii stosowanej w systemach wykrywania celu klasyfikację *prawidłową* nazywa się **prawdziwą**, a *nieprawidłową* – **fałszywą**. Na przykład jeśli pacjent ma raka i test prawidłowo wykrywa jego kondycję, to mówi się, że jest **prawdziwie pozytywny**. Jeśli zaś pacjent nie ma raka i test też to wykazuje, to zachodzi przypadek **prawdziwie negatywny**. Przypadek **fałszywie pozytywny** ma miejsce wówczas, gdy pacjent nie ma raka, a test błędnie wykazuje, że go ma. To powoduje niepotrzebne zmartwienia, a także ból i koszty związane z dodatkowymi badaniami. Jeszcze gorszy scenariusz mamy w przypadku **fałszywie negatywnym**, gdy rak jest, a test wykazuje, że pacjent jest zdrowy. Jak wszyscy wiemy, nieleczony nowotwór może być przyczyną wielu poważnych kłopotów zdrowotnych, a nawet przedwczesnego zgonu.

Ludzkie cierpienia wywołane tymi błędami dwóch rodzajów powodują, że zrównoważony wybór progu jest ważną i delikatną sprawą. Ile przypadków *fałszywie pozytywnych* można tolerować, aby zredukować liczbę przypadków *fałszywie negatywnych*. Na rys. 26.2 zaprezentowano graficzny sposób oceny tego problemu – tzw. **krzywą ROC** (*Receiver Operating Characteristics*). Krzywa ROC wyraża zależność procentowej części sygnałów będących celem i uznawanych za pozytywne (im większa ta część, tym lepiej) od procentowej części sygnałów niebędących celem a błędnie uznawanych za pozytywne (im mniejsza ta część, tym lepiej). Inaczej mówiąc, każdy punkt na krzywej ROC wyraża jeden z możliwych kompromisów między liczbą wyników prawdziwie pozytywnych i fałszywie pozytywnych.

Na rysunkach 26.2 a)÷d) pokazano cztery ustawienia progu w omawianym przykładzie wykrywania raka. Spójrzcie, na przykład, na rys. 26.2 b), gdzie próg ustawiono na wartość 17. Pamiętajcie, że każdy test dający wartość wyjściową *większą* od progu jest traktowany jako wynik *pozytywny*. Około 13% powierzchni rozkładu przypadków niebędących celem znajduje się powyżej progu (tzn. *na prawo* od progu). Spośród wszystkich pacjentów niemających raka 87% będzie uznanych za negatywnych (tj. prawdziwie negatywnych), podczas gdy 13% będzie uznanych za pozytywnych (tj. fałszywie pozytywnych). Natomiast w rozkładzie przypadków będących celem ok. 80% powierzchni znajduje się powyżej progu. To znaczy, że 80% chorych na raka da pozytywny wynik testu (tj. prawdziwie pozytywny). Pozostałe 20% mających raka będzie błędnie uznanych za negatywnych (tj. fałszywie negatywnych). Jak pokazano na krzywej ROC na rys. 26.2 b), tak dobrany próg daje punkt na tej krzywej o współrzędnych: *udział pozytywnych wyników dla przypadków niebędących celem = 13%* oraz *udział pozytywnych wyników dla przypadków będących celem = 80%*.

Proces wykrywania jest tym bardziej skuteczny, im bardziej krzywa ROC zagina się w kierunku górnego, lewego kąta wykresu. Wyniki uzyskane przez czyste zgadywanie tworzą linię prostą biegnącą po przekątnej pod kątem 45°. Względnie niskie ustawienie progu (rys. 26.2 a) powoduje, że prawie wszystkie przypadki będące celem zostają wykryte. Uzyskuje się to kosztem wielu fałszywych alarmów (wyników fałszywie pozytywnych). Ustawienie zaś progu względnie wysoko (rys. 26.2 d) wytwarza sytuację odwrotną – mało fałszywych alarmów, lecz wiele chybionych celów.

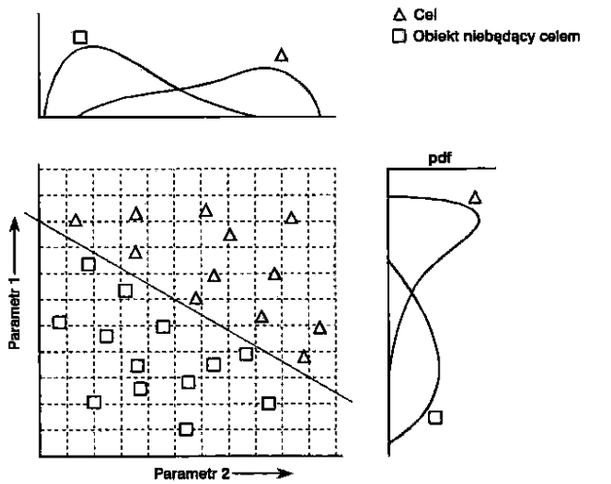


Rys. 26.2. Zależność między krzywymi ROC i pdf

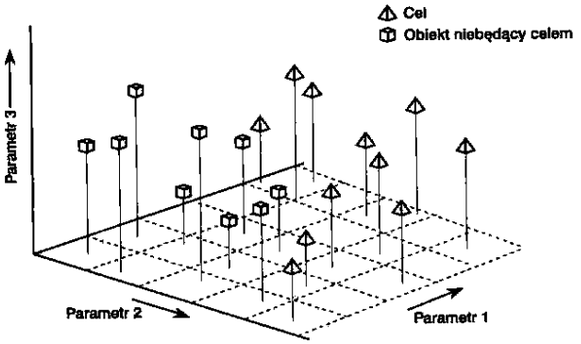
Takie metody analizy są użyteczne dla zrozumienia konsekwencji wyboru progu, lecz ko
cowa decyzja zależy od tego, co *ludzie* zaakceptują. Przypuśćmy, że początkowo ustaw
cie próg urządzenia do wykrywania raka na pewną wartość, która w waszym odczuciu jest
właściwa. Po przebadaniu urządzeniem wielu pacjentów porozmawiacie z kilkunastoma pa
cjentami, którzy uzyskali wyniki fałszywie pozytywne. Słysząc, jak *wasze* urządzenie niepo
trzebnie utrudniło życie tym ludziom, uzyskujecie silną motywację do zwiększenia progu.
Przypuszczalnie natkniecie się też na sytuację, wobec której poczujecie się jeszcze gorzej
– porozmawiacie z pacjentem nieuleczalnie chorym na raka, którego *wasze* urządzenie nie
zdołało wykryć. Na takie trudne przeżycie zareagujecie *znacznym* obniżeniem progu. W mia
rę upływu czasu i wielokrotnego powtarzania się takich zdarzeń próg stopniowo ustala się na
pewnej *wartości równowagi*. To znaczy, że częstotliwość pojawiania się wyników fałszywie
pozytywnych pomnożona przez jakiś współczynnik istotności (obniżanie progu) jest rów
noważona częstotliwością wyników fałszywie negatywnych pomnożoną przez pewien inny
współczynnik istotności (podwyższanie progu).

Tę analizę można rozszerzyć na urządzenia o więcej niż jednym wyjściu. Przypuśćmy na
przykład, że system do wykrywania raka działa na zasadzie pobierania obrazu rentgenow
skiego pacjenta i automatycznej analizy tego obrazu przy użyciu algorytmów identyfikują
cych nowotwór. Algorytmy identyfikują obszary podejrzane i następnie mierzą najważniej
sze właściwości, aby pomóc w ocenie wyniku. Załóżmy na przykład, że mierzymy średnicę podejrzanego obszaru (parametr 1) i *jaskrawość* obrazu w tym obszarze (parametr 2). Przyjmijmy, że według przeprowadzonych badań guzy są na ogół większe i jaśniejsze od zwykłych tkanek. Najpierw, jako pierwszą próbę, przeprowadźmy poprzednio opisaną analizę ROC dla każdego parametru i określmy akceptowalny próg dla każdego z nich. Wtedy możemy wynik testu określić jako pozytywny tylko jeśli spełnia oba kryteria: parametr 1 większy od pewnego progu *oraz* parametr 2 większy od innego progu.

Ta metoda oddzielnego wyznaczania progów poszczególnych parametrów i przywoływania następnie funkcji logicznych (I, LUB itd.) jest bardzo często stosowana. Niemniej jednak jest ona mało wydajna i istnieją znacznie lepsze metody. Tę kwestię wyjaśniono na rys. 26.3.



Rys. 26.3. Przykład przestrzeni dwuparametrycznej. Grupy obiektów będących celem (trójkąty) i niebędących celem (kwadraty) są całkowicie rozdzielone w dwóch wymiarach. Następuje jednak nakładanie się poszczególnych parametrów. Nakładanie się jest widoczne na jednowymiarowych funkcjach pdf przedstawionych wzdłuż każdej osi



Rys. 26.4. Przykład przestrzeni trójparametrycznej. Tak jak przestrzeń dwuparametryczna tworzy płaską płaszczyznę, tak przestrzeń trójparametryczna może być wyrażana graficznie przy użyciu konwencjonalnych osi x, y, z. Podział przestrzeni trójparametrycznej na obszary wymaga wyznaczenia rozdzielającej płaszczyzny lub powierzchni zakrzywionej

Każdy trójkąt na rysunku wyraża pojedyncze pojawienie się celu (pacjenta z rakiem), zaznaczone w miejscu zgodnym z wartościami dwóch parametrów tego celu. Każdy kwadrat zaś wyraża pojedyncze pojawienie się obiektu niebędącego celem (pacjenta niemającego raka). Jak wynika z pokazanych obok każdej osi wykresów funkcji *pdf*, rozkłady obu parametrów dla obiektów będących i niebędących celem w znacznym stopniu zachodzą na siebie. Tak więc każdy parametr, potraktowany oddzielnie, jest kiepskim wskaźnikiem prognozy raka. Powiązanie obu parametrów prostymi funkcjami logicznymi daje tylko niewielką poprawę. Jest to fakt szczególnie interesujący, gdyż oba parametry zawierają informację mogącą *doskonale* oddzielić obiekty będące celem od tych, które nim nie są. Robi się to, wykreślając linię ukośną między obiema grupami, w sposób pokazany na rysunku.

Układ współrzędnych tego rodzaju jest w języku specjalistów nazywany **przestrzenią parametryczną**. Na przykład płaszczyznę dwuwymiarową w tym przykładzie można nazwać przestrzenią średnica–jaskrawość. Chodzi tu o to, żeby obiekty będące celem zajmowały jeden obszar przestrzeni parametrycznej, a obiekty niebędące celem – drugi. Oddzielenie obu obszarów może być tak nieskomplikowane jak linia prosta lub tak skomplikowane jak zamknięte obszary o nieregularnych granicach. Następny stopień złożoności przedstawiono na rys. 26.4. Jest to przestrzeń trójparametryczna o osiach współrzędnych x, y oraz z. W ten sposób może być przedstawiony system wykrywania raka mierzący *średnicę*, *jaskrawość* i jakiś trzeci parametr, powiedzmy *ostrość krawędzi*. Tak samo jak w przypadku przestrzeni dwuwymiarowej obiekty należące do grupy celów i grupy obiektów niebędących celami powinny zajmować oddzielne obszary przestrzeni, co umożliwia ich rozdzielenie. W przestrzeni trójwymiarowej obszary są rozdzielane płaszczyznami i zakrzywionymi powierzchniami. Pojęcie **hiperprzestrzeni** (wokół, ponad lub za normalną przestrzenią) jest często stosowane do opisu przestrzeni parametrycznych o więcej niż trzech wymiarach. Matematycznie hiperprzestrzenie nie różnią się od przestrzeni jedno-, dwu- i trójwymiarowych; w praktyce nie można ich jednak obrazować w formie graficznej w naszym trójwymiarowym świecie.

Próg dobrany do zadania jednoparametrycznego nie może (na ogół) być klasyfikowany jako właściwy lub niewłaściwy. Każda wartość progu daje bowiem inną unikatową kombinację przypadków fałszywie-pozytywnych i fałszywie-negatywnych, tj. jakiś punkt na krzywej ROC. Jest to po prostu kwestia kompromisu liczby jednych i drugich przypadków, nie ma tu odpowiedzi bezwzględnie poprawnej. Z drugiej strony, w przestrzeniach parametrycznych o dwóch lub więcej parametrach mogą występować zdecydowanie nieprawidłowe podziały między obszarami. Wyobraźmy sobie na przykład wzrastającą liczbę punktów danych na rys. 26.3,

wykazującą pewne nałożenie się grup przypadków będących i niebędących celem. Można przesuwać linię progu między grupami, aby zmniejszać liczbę przypadków fałszywie pozytywnych kosztem fałszywie negatywnych lub na odwrót. Trzeba wtedy linię ukośną przesuwać w kierunku prawego górnego lub lewego dolnego narożnika. Nie należy jednak wykonywać obrotu tą linią, gdyż wtedy zwiększy się liczba błędnych przypadków *obu* rodzajów.

Jak wynika z omówionych przykładów, konwencjonalne podejście do detekcji celu (zwanej czasem rozpoznawaniem wzorca lub obrazu) jest procesem dwuetapowym. Pierwszy etap nosi nazwę **wyodrębniania cech**. Do tego celu stosuje się algorytmy redukujące dane pierwotne do kilku parametrów, takich jak średnica, jaskrawość, ostrość krawędzi itd. Te parametry są często nazywane **cechami** albo **klasyfikatorami**. Wyodrębnianie cech jest konieczne do zredukowania ilości danych. Medyczny obraz rentgenowski może, na przykład, zawierać więcej niż milion pikseli. Zadaniem wyodrębniania cech jest przedestylowanie informacji do postaci bardziej skoncentrowanej i łatwiejszej do dalszego przetwarzania. Opracowanie algorytmu tego rodzaju jest zadaniem bardziej z dziedziny sztuki niż nauki. Wymaga wielkiego doświadczenia i umiejętności spojrzenia na problem i powiedzenia: *Oto klasyfikatory, które umożliwią najlepsze wychwycenie informacji*. Ważną rolę gra tu metoda prób i błędów.

W drugim etapie dokonuje się oceny klasyfikatorów w celu stwierdzenia, czy cel jest obecny czy nie. Inaczej mówiąc, trzeba zastosować jakąś metodę służącą do podziału przestrzeni parametrycznej na obszary odpowiadające celom i obiektom niebędącym celami. Jest to całkiem proste zadanie w przypadku przestrzeni jedno- lub dwuparametrycznej; punkty o znanych danych są nanoszone na układ współrzędnych (jak na rys. 26.3) i obszary można wówczas rozdzielić nawet wzrokowo. Ten podział zostaje następnie wpisany do komputera w postaci wzoru lub w jakiś inny sposób definiujący te obszary. W zasadzie taką samą procedurę można zastosować do trójwymiarowej przestrzeni parametrycznej. Pojawia się jednak wtedy trudność wynikająca z faktu, że wykresy trójwymiarowe są dla człowieka bardzo trudne do zrozumienia i wizualizacji (jak na rys. 26.4). Ostrzeżenie: Nie próbujcie tego z hiperprzestrzenią, bo wasze mózgi eksplodują!

Krótko mówiąc, potrzebna jest maszyna mogąca dokonywać podziału przestrzeni wielowymiarowej, zgodnie z przykładami sygnałów będących i niebędących celem. Taki system wykrywania celu jest szczególnie bliski głównemu tematowi tego rozdziału – sieciom neuronowym.

Architektura sieci neuronowych

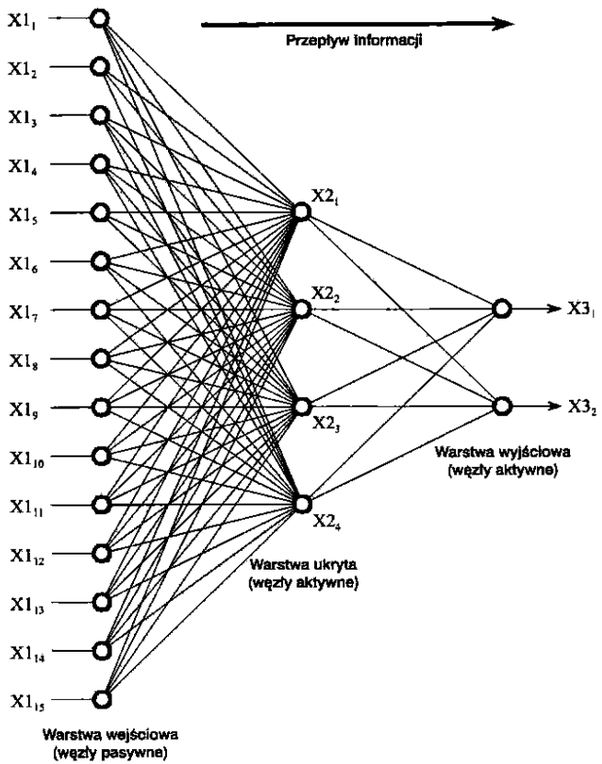
Ludzie i zwierzęta przetwarzają informacje *sieciami neuronowymi*. Tworzą je *biliony* **neuronów** (komórek nerwowych) wymieniających między sobą impulsy elektryczne zwane **potencjałami czynnościowymi**. Algorytmy komputerowe naśladujące struktury biologiczne są formalnie nazywane **sztucznymi sieciami neuronowymi** dla odróżnienie od sieci w organizmach ludzi i zwierząt. Jednak większość naukowców i inżynierów nie podchodzi do tej kwestii tak formalnie i stosuje nazwę *sieci neuronowe* tak do systemów biologicznych, jak i niebiologicznych.

Motywacją do prowadzenia badań w dziedzinie sieci neuronowych były dwa pragnienia: osiągnięcie lepszego zrozumienia ludzkiego mózgu i opracowanie komputerów zdolnych do rozwiązywania zadań abstrakcyjnych i słabo zdefiniowanych. Komputery konwencjonalne mają na przykład kłopoty z rozumieniem mowy i rozpoznawaniem ludzkich twarzy. A ludzie te zadania wykonują z łatwością.

Wypróbowano wiele różnych struktur sieci neuronowych. Niektóre były naśladownictwem tego, co biolodzy widzą pod mikroskopem, w innych korzystano z matematycznej analizy problemu. Strukturę najczęściej stosowaną przedstawiono na rys. 26.5. Ta sieć składa się z trzech warstw zwanych **warstwą wejściową**, **ukrytą i wyjściową**. Każda warstwa zawiera jeden lub więcej **węzłów**, które na rysunku zaznaczono kółeczkami. Linie między węzłami pokazują przepływ informacji od jednego węzła do następnego (a więc z lewa na prawo). Sieci neuronowe innych rodzajów mają bardziej zawiłe połączenia, takie jak pętle sprzężenia zwrotnego.

Węzły w sieci wejściowej są **pasywne**, co znaczy, że nie modyfikują danych. Odbierają pojedynczą wartość na wejściu i powielają tę wartość na swych wielokrotnych wyjściach. Węzły w warstwach ukrytej i wyjściowej zaś są **aktywne**. To znaczy, że modyfikują dane, jak to przedstawiono na rys. 26.6. Zmienne $X1_1$, $X1_2$, ... $X1_{15}$ zawierają dane, które mają być oceniane (rys. 26.5). Mogą to być np. wartości pikseli z jakiegoś obrazu, próbki sygnału fonicznego, ceny akcji na giełdzie w kolejnych dniach. Mogą to też być dane wyjściowe uzyskane z jakiegoś algorytmu, takie jak klasyfikatory z omówionego przykładu wykrywania raka: średnica, jaskrawość, ostrość krawędzi itd.

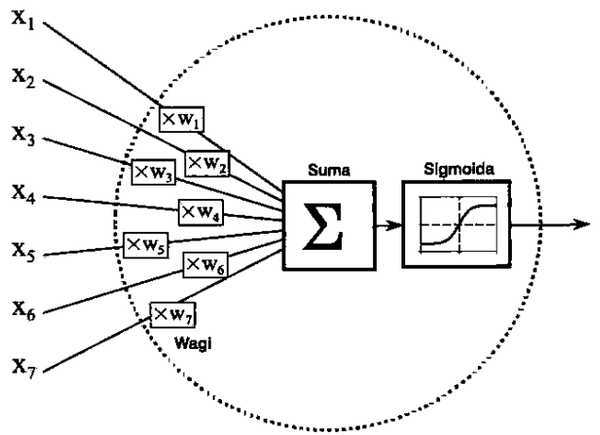
Każda wartość z warstwy wejściowej jest powielana i wysyłana do *wszystkich* węzłów ukrytych. Jest to struktura zwana **całkowicie połączoną**. Wartości wchodzące do węzła ukrytego są mnożone przez **wagi** (rys. 26.6) będące zestawem ustalonych z góry liczb zarejestrowanych w programie. Wejścia ważone są następnie sumowane, dając w rezultacie jedną liczbę. Na rysunku sumowanie zaznaczono symbolem Σ. Przed opuszczeniem węzła ta liczba jest poddawana nieliniowej funkcji matematycznej zwanej *sigmoidą*. Wykres tej funkcji ma kształt litery *S*, co ogranicza sygnał wyjściowy z węzła. Sygnał wejściowy sigmoidy jest wartością między $-\infty$ i $+\infty$, a wyjściowy może mieścić się tylko w granicach od 0 do 1.



Rys. 26.5. Architektura sieci neuronowej. Jest to najczęściej spotykana struktura sieci neuronowych: trzy warstwy w pełni połączone. Węzły sieci wejściowej są pasywne, tylko przekazują wartości ze swego pojedynczego wejścia do wielokrotnych wyjść. Natomiast węzły warstw ukrytej i wyjściowej są aktywne i modyfikują sygnały zgodnie z rys. 26.6. Działanie tej sieci neuronowej jest określone wagami użytymi w węzłach warstw ukrytej i wyjściowej

Tab. 26.1.

```
100 'SIEĆ NEURONOWA (DO SIECI PRZEPŁYWU Z RYS. 26.5)
110 '
120 DIM X1[15]                    'przechowuje wartości wejściowe
130 DIM X2[4]                     'przechowuje wartości wychodzące z warstwy ukrytej
140 DIMX3[2]                      'przechowuje wartości wychodzące z warstwy wyjściowej
150 DIM WH[4,15]                  'przechowuje wagi warstwy ukrytej
160 DIM WO[2,4]                   'przechowuje wagi warstwy wyjściowej
170 '
180 GOSUB XXXX                    'podprogram "mityczny" do ładowania danych wejściowych X X1[ ,
190 GOSUB XXXX                    'podprogram "mityczny" do ładowania wag WH[ , ]&WO[ , ]
200 '
210 '                             'WYZNACZ WARTOŚCI WĘZŁÓW UKRYTYCH, X2[ ]
220 FOR J% = 1 TO 4               'pętla przez wszystkie węzły warstwy ukrytej
230    ACC = 0                    'wyzeruj zmienną akumulatora, ACC
240    FOR I%= 1 TO 15            'pomnóż przez wagi i zsumuj wszystkie węzły wejściowe
250      ACC = ACC + X1 [I%] * WH[J%,I%]
260    NEXT I%
270    X2[J%] = 1 /(1 + EXP(-ACC)  'przepuść zsumowaną wartość przez sigmoidę
280 NEXT J%
290 '
300 '                             'WYZNACZ WARTOŚCI WĘZŁÓW WYJŚCIOWYCH X3[ ]
310 FOR J% = 1 TO 2               'pętla przez wszystkie węzły warstwy wyjściowej
320    ACC = 0                    'wyzeruj zmienną akumulatora, ACC
330    FOR I% = 1 TO 4            'pomnóż przez wagi i zsumuj wszystkie węzły ukryte
340      ACC = ACC + X2[I%] * WO[J%,I%]
350    NEXT I%
360    X3[J%] =1/(1+ EXP(-ACC) )  'przepuść zsumowaną wartość przez sigmoidę
370 NEXT J%
380 '
390 END
```



Rys. 26.6. Węzeł aktywny sieci neuronowej. Jest to sieć przepływu węzłów aktywnych stosowanych w warstwach ukrytej i wyjściowej. Wszystkie wartości wejściowe są wymnażane przez pewne wagi (wartości w_N) i następnie sumowane. W ten sposób jest generowana pojedyncza wartość, poddawana potem nieliniowej funkcji o kształcie litery S zwanej sigmoidą. Sigmoidę pokazano szczegółowo na rys. 26.7

Wyjścia warstwy ukrytej są oznaczone na sieci przepływu (rys. 26.5) jako zmienne $X2_1$, $X2_2$, $X2_3$ i $X2_4$. Podobnie jak poprzednio, każda z tych wartości jest powielana i wprowadzana do następnej warstwy. Węzły aktywne warstwy wyjściowej łączą dane i modyfikują je, wytwarzając dwie wartości wyjściowe tej sieci $X3_1$ i $X3_2$.

Sieci neuronowe mogą mieć dowolną liczbę warstw i dowolną liczbę węzłów w każdej sieci. Najczęściej stosuje się sieci trójwarstwowe o liczbie węzłów nie większej niż kilkaset. Warstwa ukryta ma zwykle rozmiar ok. 10% warstwy wejściowej. W przypadku wykrywania celu w warstwie wyjściowej wystarczy tylko pojedynczy węzeł. Sygnał wyjściowy z tego węzła jest poddawany progowi, dając pozytywną lub negatywną wskazówkę o obecności lub nieobecności celu w danych wejściowych.

W tablicy 26.1 zamieszczono program realizujący sieć przepływu z rys. 26.5. Co najważniejsze, ta architektura jest bardzo prosta i bardzo uogólniona. Ta sama sieć przepływu może być wykorzystywana do wielu zadań, niezależnie od ich specyfiki. Zdolność sieci neuronowej do pożytecznego operowania danymi polega na właściwym doborze *wag*. Jest to spektakularne odejście od konwencjonalnego przetwarzania informacji, gdzie rozwiązania są opisywane procedurami realizowanymi krok po kroku.

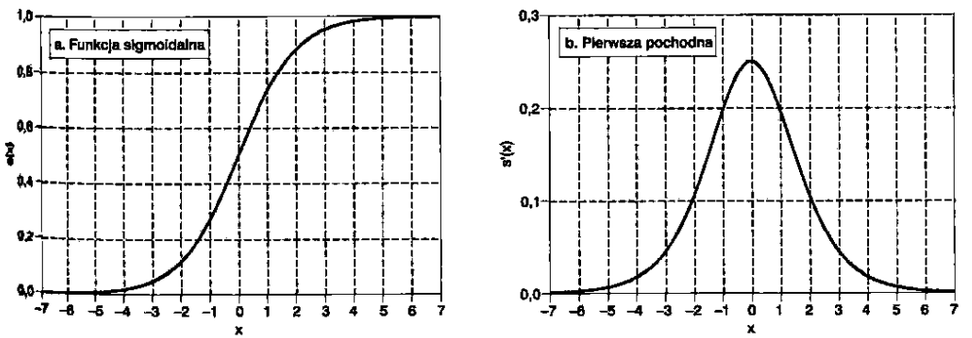
Weźmy jako przykład sieć neuronową służącą do rozpoznawania obiektów w jakimś sygnale. Przypuśćmy, że w komputerze zgromadzono 1000 próbek z sygnału. Jak komputer określi, czy te dane reprezentują łódź podwodną, wieloryba, górę podmorską czy zgoła nic? W konwencjonalnym DSP rozwiązywałoby się ten problem, korzystając z matematyki i algorytmów, takich jak analiza widma częstotliwościowego i korelacja. W przypadku sieci neuronowej te 1000 próbek wprowadza się po prostu do warstwy wejściowej, w wyniku czego jakieś wartości „wyskakują" z warstwy wyjściowej. Przez dobór właściwych wag można skonfigurować wyjście tak, aby dostarczało szeroki zakres informacji. Mogą to być na przykład wyjścia typu: łódź podwodna (tak/nie), wieloryb (tak/nie), podmorska góra (tak/nie) itd.

Przy innych wagach, wyjścia mogą klasyfikować obiekty jako: metal czy niemetal, obiekt biologiczny czy niebiologiczny, wróg czy sojusznik itd. Nie ma tu algorytmów, nie ma reguł, nie ma procedur, a tylko zależność między wejściem i wyjściem podyktowana przez wybrane wartości wag.

Na rys. 26.7 dokładniej pokazano przebieg sigmoidy, matematycznie opisanej wzorem:

$$s(x)=\frac{1}{1+e^{-x}}$$

Wzór 26.1. Funkcja sigmoidalna. Jest ona stosowana w sieciach neuronowych jako łagodny próg. Przebieg tej funkcji pokazano na rys. 26.7 a)



Rys. 26.7. Funkcja sigmoidalna i jej pochodna. Te krzywe są opisane wzorami 26.1 i 26.2

Dokładny kształt sigmoidy nie jest istotny; ważne, że jest to **łagodny próg**. Przypomnijmy, że **prosty próg** daje wartość *jeden*, gdy $x > 0$ i wartość *zero*, gdy $x < 0$. Sigmoida realizuje taką samą podstawową funkcję progową, lecz jest też *różniczkowalna*, jak to pokazano na rys. 26.7 b). Wprawdzie ta pochodna nie jest stosowana w sieci przepływu (rys. 25.5), to jest najważniejszym elementem w poszukiwaniu właściwych wag. Wkrótce więcej o tym powiemy. Zaletą sigmoidy jest możliwość obliczania jej pochodnej w skrócony sposób:

$$s'(x)=s(x)\left[1-s(x)\right]$$

Wzór 26.2. Pierwsza pochodna sigmoidy. Oblicza się ją korzystając z wartości samej sigmoidy

Na przykład jeśli $x = 0$, to $s(x) = 0{,}5$ (ze wzoru 26.1) i pierwszą pochodną obliczamy jako $s'(x) = 0{,}5(1 - 0{,}5) = 0{,}25$. Nie jest to jakiś pomysł o zasadniczym znaczeniu, a tylko krótki, upraszczający obliczenia algebraiczne.

Czy sieć neuronowa miałaby większą elastyczność działania, gdyby sigmoidę można było przesunąć w lewo lub w prawo, środkując ją na jakiejś wartości innej niż $x = 0$? Odpowiedź brzmi „tak", w większości sieci neuronowych jest to możliwe. Można to łatwo zrealizować, wprowadzając do warstwy wejściowej dodatkowy węzeł, którego wejście ma zawsze wartość *jeden*. Ta wartość wejściowa przemnażana przez wagi warstwy ukrytej daje pewną *polaryzację* (przesunięcie o składową stałą). Ten dodatkowy węzeł nazywamy **polaryzującym**. Jest on traktowany tak samo, jak pozostałe węzły, z tym tylko że na wejściu ma wartość stałą.

Czy można zrealizować sieć neuronową bez sigmoidy lub podobnej nieliniowości? Aby na to powiedzieć na to pytanie, spójrzmy na trójwarstwową sieć na rys. 26.5. Gdyby nie było sigmoidy, to trzy warstwy *zredukowałyby* się do dwóch. Innymi słowy, sumowanie i wagi warstw ukrytej i wyjściowej mogłyby zostać połączone w pojedynczą warstwę, a więc sieć stałaby się tylko dwuwarstwowa.

Dlaczego to działa?

Wagi są niezbędne do tego, aby sieć neuronowa mogła spełnić wyznaczone jej zadanie. Określa się ich wartości stosując **algorytm uczący się** wraz z **przykładami** pokazującymi, jak system *powinien* działać. Takie przykłady, powiedzmy w systemie sonaru, są zawarte w bazie danych w postaci kilkuset (lub więcej) z 1000 próbkowanych segmentów. Niektóre z przykładowych segmentów dotyczyłyby łodzi podwodnych, inne – wielorybów, jeszcze inne – szumu losowego itd. W algorytmach uczących się te przykłady są wykorzystywane do obliczania zestawu wag odpowiednich do przewidywanego zadania. Pojęcie **uczenie się** (lub trenowanie) jest szeroko stosowane w dziedzinie sieci neuronowych do opisania tego procesu; jednak lepszym opisem mogłoby być: *określanie optymalnego zestawu wag na podstawie statystyki przykładów*. Niezależnie od użytej nazwy metody, uzyskane w jej wyniku wagi są dla człowieka, w istocie, zupełnie niezrozumiałe. W niektórych rzadkich przypadkach można w nich zauważyć jakieś prawidłowości, lecz na ogół są to liczby losowe. W sieciach neuronowych, w których zastosowano te wagi, można zaobserwować prawidłowe zależności między wyjściem a wejściem, lecz trudno pojąć, *dlaczego* te właśnie wagi dobrze działają. Te tajemnicze właściwości sieci neuronowych zniechęcają do nich wielu naukowców i inżynierów. Pamiętacie te wszystkie filmy *science fiction* o zdegenerowanych komputerach zdobywających władzę nad światem?

Mimo to, ciągle słyszy się obrońców sieci neuronowych głoszących stwierdzenia, takie jak: *sieci neuronowe są łatwo zrozumiałe*. Aby zbadać prawdziwość takiego stwierdzenia, wykażemy najpierw, że jest możliwy wybór wag sieci neuronowej tradycyjnymi metodami DSP. Następnie pokażemy, że algorytmy uczące się dają rozwiązania *lepsze* niż metody tradycyjne. Wprawdzie to nie wyjaśnia, *dlaczego* dobrze działa jakiś określony zestaw wag, lecz wzbudza zaufanie do metody.

Patrząc na sieci neuronowe w sposób bardziej wysublimowany, można powiedzieć, że jest to metoda etykietowania różnych obszarów w *przestrzeni parametrycznej*. Jako przykład rozważmy system sonaru będący siecią neuronową o 1000 wejść i pojedynczym wyjściu. Przy prawidłowym wyborze wag sygnał wyjściowy będzie bliski *jeden*, jeśli sygnał wejściowy jest echem odbitym od łodzi podwodnej i bliski *zeru*, jeśli jest sygnał wejściowy jest tylko szumem. W ten sposób powstaje hiperprzestrzeń parametryczna o 1000 wymiarach. Sieć neuronowa jest metodą przypisania jakiejś wartości każdemu położeniu w tej hiperprzestrzeni. Tak więc 1000 wartości wejściowych określa *lokalizację* w tej hiperprzetrzeni, sygnał zaś wyjściowy z sieci – *wartość* dla tej lokalizacji. To zadanie wykonałaby doskonale tablica przeglądowa mająca zarejestrowaną wartość wyjściową dla każdego możliwego adresu wejściowego. Różnica leży w tym, że sieć neuronowa *oblicza* wartość dla każdej lokalizacji (adresu), a nie realizuje wielkiego ponad możliwości zadania *zarejestrowania* każdej wartości. Faktycznie, architektury sieci neuronowych często ocenia się na podstawie tego, jak dobrze rozdzielają one hiperprzestrzeń przy danej liczbie wag.

Przy takim podejściu do problemu uzyskuje się również wskazówkę dotyczącą wymaganej liczby węzłów w warstwie ukrytej. Przestrzeń parametryczna N-wymiarowa wymaga N liczb do opisania lokalizacji. Identyfikacja *obszaru* w hiperprzestrzeni wymaga $2N$ wartości (tj. wartości minimalnych i maksymalnych na każdej osi, wyznaczających hiperprzestrzenną bryłę prostopadłościenną). Te proste obliczenia mogą – na przykład – wykazać, że sieć neuronowa o 1000 wejść wymaga 2000 wag do odróżnienia jednego obszaru hiperprzestrzeni od drugiego. W sieci całkowicie połączonej wymagałoby to dwóch węzłów ukrytych. Liczba niezbędnych obszarów zależy od określonego problemu, lecz można oczekiwać, że będzie ona znacznie mniejsza niż liczba wymiarów w przestrzeni wieloparametrycznej. Choć jest to tylko grube przybliżenie, to wyjaśnia ono ogólnie, dlaczego większość sieci neuronowych może działać z warstwą ukrytą stanowiącą od 2 do 3% rozmiaru warstwy wejściowej.

W zupełnie inny sposób można rozumieć sieci neuronowych, korzystając ze stosowanego w DSP pojęcia *korelacji*. Jak stwierdzono w rozdziale 7, korelacja jest optymalnym sposobem wykrycia, czy dany wzorzec jest zawarty w sygnale. Korelację przeprowadza się przez wymnażanie sygnału i poszukiwanego wzorca i następnie sumowanie iloczynów. Czym suma jest większa, tym bardziej sygnał jest podobny do wzorca. Spójrzcie teraz na rys. 26.5 i pomyślcie o każdym ukrytym węźle jako o poszukującym specyficznego wzorca w danych wejściowych. To znaczy, że każdy z ukrytych węzłów *koreluje* dane wejściowe z zestawem wag związanych z tym węzłem ukrytym. Jeśli wzorzec jest w danych wejściowych, to suma przekazywana do sigmoidy będzie duża, w innym razie – będzie mała.

Działanie sigmoidy jest, z tego punktu widzenia, całkiem interesujące. Wróćcie do rys. 26.1 d) i zwróćcie uwagę, że krzywa prawdopodobieństwa rozdzielająca dwa rozkłady o kształcie dzwonowym przypomina sigmoidę. Gdybyście w sposób najprostszy projektowali sieć neuronową, moglibyście sygnał wyjściowy każdego ukrytego węzła uczynić równym *ułamkowemu prawdopodobieństwu* tego, czy dany wzorzec jest w danych wejściowych. Warstwa wyjściowa powtarza tę operację, powodując, że cała trójwarstwowa struktura staje się korelacją korelacji, siecią szukającą *wzorców wzorców*.

Podstawą konwencjonalnego DSP są dwie metody – *splot* oraz *analiza Fouriera*. To jest pocieszające, że sieci neuronowe mogą wykonywać obie te operacje, i jeszcze *wiele więcej*. Wyobraźcie sobie sygnał N-próbkowy filtrowany w celu uzyskania innego sygnału N-próbkowego. Zgodnie ze splotem rozpatrywanym od strony wyjścia, każda próbka w sygnale wyjściowym jest sumą ważoną próbek z wejścia. A teraz, weźmy dwuwarstwową sieć neuronową o N węzłach w każdej warstwie. Wartość wytwarzana przez każdy z węzłów warstwy wyjściowej jest również sumą ważoną wartości wejściowych. Jeśli w każdym węźle sieci wyjściowej zastosowano wagi takie same, jak we wszystkich pozostałych węzłach wyjściowych, to sieć będzie realizować splot liniowy. Przekształcenie DFT zaś może być obliczone przez dwuwarstwową sieć neuronową o N węzłach w każdej warstwie. Każdy węzeł

warstwy wyjściowej wyznacza amplitudę jednej składowej częstotliwościowej. Wykonując się to zadanie, ustalając wagi każdego węzła warstwy wyjściowej takie same, jak sinusoida, której się szuka. Uzyskana w ten sposób sieć koreluje sygnał wejściowy z każdą podstawową funkcją sinusoidalną, obliczając DFT. Oczywiście, dwuwarstwowa sieć neuronowa ma możliwości znacznie mniejsze niż standardowa architektura trójwarstwowa. To znaczy, że sieci neuronowe mogą realizować zarówno przetwarzanie *nieliniowe*, jak i *liniowe*.

Przypuśćmy, że do zaprojektowania wag sieci neuronowej zastosowano jedną z konwencjonalnych metod DSP. Czy można twierdzić, że taka sieć jest *optymalna*? Tradycyjne algorytmy DSP opierają się zwykle na założeniach dotyczących właściwości sygnału wejściowego. Na przykład filtracja Wienera jest optymalna dla maksymalizacji stosunku sygnału do szumu *przy założeniu*, że widma sygnału i szumu są znane; korelacja jest optymalna do wykrywania celów *przy założeniu*, że szum jest biały; rozplot przeciwdziała niepożądanemu splotowi *przy założeniu*, że odpowiedź impulsowa rozplotu jest odwrotnością oryginalnej odpowiedzi impulsowej splotu itd. Trudność polega na tym, że naukowcy i inżynierowie rzadko mają doskonałą wiedzę o sygnałach wejściowych, które można napotkać. Chociaż operacje matematyczne mogą być eleganckie, to działanie całego systemu jest ograniczone poziomem dobrego rozumienia danych.

Jako przykład weźcie testowanie tradycyjnego algorytmu DSP aktualnymi sygnałami wejściowymi. Następnie powtórzcie ten test z algorytmem nieco zmienionym, na przykład przez zwiększenie jednego parametru o 1%. Jeśli wynik drugiego testu jest lepszy niż pierwszego, to oryginalny algorytm nie jest optymalizowany dla aktualnie wykonywanego zadania. Niemal wszystkie konwencjonalne algorytmy DSP można znacznie ulepszyć przez szacowanie – metodą prób i błędów – małych zmian parametrów i procedur algorytmu. To jest właśnie sposób działania sieci neuronowych.

Trenowanie sieci neuronowych

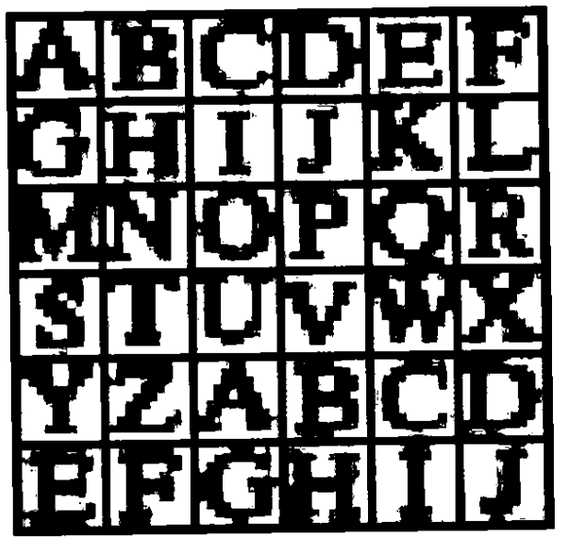
Projektowanie sieci neuronowych można najlepiej objaśnić na przykładzie. Problem, który chcemy rozwiązać, przedstawiono na rys. 26.8. Jest to identyfikowanie pojedynczych liter w obrazie tekstu. Takiemu zadaniu, będącemu rozpoznawaniem obrazu, poświęca się wiele uwagi. Jest ono na tyle łatwe, że w wielu próbach jego rozwiązania osiągnięto częściowy sukces, ale też na tyle trudne, że nie ma rozwiązań idealnych. Rozwiązanie tego problemu wykorzystano z powodzeniem w wielu produktach handlowych, takich jak urządzenia do odczytywania adresów w obrocie pocztowym, a także do wprowadzania dokumentów do procesorów tekstów.

Pierwszym krokiem w opracowaniu sieci neuronowej jest utworzenie bazy danych złożonej z przykładów. W problemie rozpoznawania tekstu tworzy się taką bazę przez 50-krotne wydrukowanie na kartce papieru 26 dużych liter: A, B, C, D ... Y, Z. Następnie te 1300 liter przetwarza się na obraz cyfrowy, stosując jedno z wielu urządzeń skanujących dostępnych wraz z komputerami osobistymi. Ten duży obraz cyfrowy jest następnie dzielony na małe obrazy 10 × 10 pikseli, z których każdy zawiera pojedynczą literę. Ta informacja zostaje zarejestrowana jako 1,3-megabajtowa baza danych: 1300 obrazów, 100 pikseli na obraz, 8 bitów na piksel. Pierwsze 260 obrazów z tej bazy będzie użytych do *trenowania* sieci neuronowej (tj. określania wag), a pozostałe obrazy – do *testowania działania* sieci. Baza danych musi też zawierać sposób identyfikacji litery zawartej w każdym z obrazów. Można na przykład do każdego obrazu 10 × 10 dodać dodatkowy bajt zawierający kod ASCII danej litery. W innym rozwiązaniu położenie każdego obrazu 10 × 10 w bazie danych może wskazywać, jaka to jest litera. Na przykład wszystkie obrazy od 0 do 49 mogą być literą *A*, obrazy 50–99 literą *B* itd.

W omawianym przykładzie będzie projektowana sieć neuronowa do dowolnego zadania – określenia, które obrazy 10 × 10 zawierają samogłoskę A, E, I, O lub U. Takie rozwiązanie może nie mieć żadnego praktycznego zastosowania, lecz dobrze ilustruje zdolność sieci neuronowych do uczenia się bardzo abstrakcyjnych problemów rozpoznawania obrazu. Przez włączenie dziesięciu przykładów każdej litery do zestawu trenującego sieć będzie (można mieć nadzieję) się uczyła podstawowych cech rozróżniających obrazy będące celem od tych, które nim nie są.

Zastosowana w tym przykładzie sieć neuronowa jest typową siecią z trzema warstwami w pełni połączonymi, taka jak pokazano na rys. 26.5 i 26.6. Jest 101 węzłów w warstwie wejściowej (100 wartości pikseli oraz węzeł polaryzujący), 10 węzłów w warstwie ukrytej i 1 węzeł w warstwie wyjściowej. Gdy na wejście sieci jest podany obraz 100-pikselowy, to chcielibyśmy uzyskać na wyjściu wartość bliską *jeden*, jeśli jest samogłoska oraz bliską *zero*, gdy jej nie ma. Nie kłopoczcie się tym, że sygnał wejściowy został pobrany jako matryca dwuwymiarowa (10 × 10), podczas gdy wejście sieci neuronowej jest matrycą jednowymiarową. To jest *wasz* sposób rozumienia współzależności między pikselami; *sieć neuronowa* po swojemu wyszuka te zależności.

W tablicy 26.2 podano główny program do obliczania wag sieci neuronowych, a w tablicy 26.3 – trzy podprogramy wywoływane z programu głównego. W elementach matrycy od X1[1] do X1[100] są przechowywane wartości warstwy wejściowej. Dodatkowo, w elemencie X1[101] jest zawsze przechowywana wartość 1, stanowiącą wejście węzła polaryzującego. Wyjściowe wartości węzłów ukrytych są zawarte w elementach matrycy od X2[1] do X2[10]. Zmienna X3 zawiera wartość wyjściową sieci. Wagi warstwy ukrytej znajdują się w matrycy WH[,], gdzie pierwszy indeks określa węzeł ukryty (od 1 do 10), a drugi indeks jest węzłem warstwy wejściowej (od 1 do 101). Wagi warstwy wyjściowej są przechowywane w matrycy od WO[1] do WO[10]. Tak więc działanie sieci jest określone przez 1020 wartości wag.



Rys. 26.8. Przykład obrazu tekstu. Identyfikacja liter w obrazach tekstowych jest jednym z klasycznych zadań w dziedzinie rozpoznawania obrazów. W tym przykładzie każda litera jest zawarta w obrazie o rozmiarze 10 × 10 pikseli o 256 poziomach szarości na piksel. Baza danych zastosowana do trenowania i testowania tej przykładowej sieci neuronowej składa się z 50 zestawów 26 dużych liter, co daje w sumie 1300 obrazów. Obrazy pokazane na rysunku są częścią tej bazy danych

Tab. 26.2.

```
100 'TRENOWANIE SIECI NEURONOWYCH (Określanie wag)
110 '
120                              'INICJALIZACJA
130 MU = .000005                 'wartość kroku iteracji
140 DIM X1[101]                  'przechowuje sygnał warstwy wejściowej + wyraz polaryzujący
150 DIM X2[10]                   'przechowuje sygnał warstwy ukrytej
160 DIM WH[10,101]               'przechowuje wagi warstwy ukrytej
170 DIM WO[10]                   'przechowuje wagi warstwy wyjściowej
180 '
190 FOR H% = 1 TO 10             'USTAW LOSOWE WARTOŚCI WAG
200    WO[H%] = (RND-0.5)        'wagi warstwy wyjściowej: -0.5 do 0.5
210    FOR I% = 1 TO 101         'wagi warstwy ukrytej: -0.0005 do 0.0005
220       WH[H%,I%] = (RND-0.5)/1000
230    NEXT I%
240 NEXT H%
250 '
260                              'PĘTLA ITERACJI
270 FOR ITER% = 1 TO 800         'pętla przez 800 iteracji
280 '
290    ESUM = 0                  'wyzeruj akumulator błędu, ESUM
300 '
310    FOR LETTER% = 1 TO 260    'pętla przez wszystkie litery w zestawie trenującym
320       GOSUB 1000             'ładuj zestaw trenujący do X1[ ]
330       GOSUB 2000             'wyznacz błąd dla tej litery, ELET
340       ESUM = ESUM + ELET^2   'akumuluj błąd dla tej iteracji
350       GOSUB 3000             'wyznacz nowe wagi

360 NEXT LETTER%
370
380    PRINT ITER% ESUM          'drukuj postęp iteracji na ekranie wideo
390 '
400 NEXT ITER%
410 '
420 GOSUB XXXX                   'podprogram "mityczny" do zapamiętania wag
430 END
```

Pierwszym działaniem programu jest ustawienie każdej wagi na pewną wartość początkową przy użyciu generatora liczb losowych. Jak pokazano w wierszach od 190 do 240, wagom warstwy ukrytej przypisano wartości początkowe między –0,0005 i 0,0005, a warstwy wyjściowej między –0,5 i 0,5. Wartości zakresów wybiera się w taki sposób, aby ich rząd wielkości był taki sam, jaki *muszą* mieć obliczone wagi. Określa się te wartości na podstawie 1) zakresu wartości sygnału wejściowego, 2) liczby wejść sumowanych w każdym węźle oraz 3) zakresu wartości wynikających z zakresu zmian sigmoidy (wejście ok. $-5 < x < 5$ a wyjście od 0 do 1). Gdy na przykład 101 sygnałów wejściowych o typowej wartości 100 jest mnożonych przez typową wartość wagi 0,0002, to suma iloczynów jest ok. 2, a więc mieści się w wejściowym zakresie zmian sigmoidy.

Gdybyśmy oceniali działanie sieci neuronowej, stosując te wagi losowe, to moglibyśmy oczekiwać wyniku takiego, jak przy losowym zgadywaniu. Algorytm uczący się polepsza działanie sieci przez stopniowe zmiany każdej wagi we właściwym kierunku. Jest to procedura **iteracyjna**, w programie realizowana w pętli FOR-NEXT w wierszach 270–400. Każda iteracja powoduje, że wagi stają się trochę bardziej skuteczne w rozdzielaniu obiektów będących i niebędących celem. Pętla iteracyjna jest zwykle wykonywana tak długo, aż przestaje się uzyskiwać dalszą poprawę. W typowych sieciach neuronowych może to być liczba iteracji gdzieś pomiędzy wartościami dziesięć i dziesięć tysięcy, na ogół iteracji jest kilkaset. W omawianym przykładzie przeprowadzono 800 iteracji.

Tab. 26.3.

```
'PODPROGRAM DO ŁADOWANIA OBRAZÓW Z BAZY DANYCH DO X1[ ]]
'Zmienne wchodzące do podprogramu: LETTER%
'Zmienne wychodzące z podprogramu: X1[1] do X1[100], X1[101] = 1, CORRECT
'
'Zmienna LETTER%, między 1 a 256, wskazuje, który obraz w bazie danych ma powrócić
'do X1[1] do X1[100]. Węzeł polaryzujący X1[101] zawsze ma wartość 1. Zmienna CORRECT
'ma wartość jeden, jeśli obraz wracający jest samogłoską, a w innych przypadkach - wartość zero.
'(Szczegóły tego podprogramu nie są istotne i nie są podane w tym listingu).
RETURN

'PODPROGRAM DO OBLICZANIA BŁĘDU Z BIEŻĄCYMI WARTOŚCIAMI WAG
'Zmienne wchodzące do podprogramu: X1[1], X[2], WI[ , ], WH[ ], CORRECT
'Zmienne wychodzące z podprogramu: ELET
'
'                                    'WYZNACZ WARTOŚĆI WĘZŁÓW UKRYTYCH , X2[ ]
FOR H% = 1 TO 10                     'pętla przez wszystkie węzły ukryte
   ACC = 0                           'wyzeruj akumulator
   FOR I% = 1 TO 101                 'pomnóż przez wagi i zsumuj wszystkie węzły wejściowe
      ACC = ACC + X1[I%] * WH[H%,I%]
   NEXT I%
   X2[H%] = 1/(1+ EXP(-ACC) )        'przepuść zsumowaną wartość przez sigmoidę
NEXT H%
'
'                                    'WYZNACZ WARTOŚĆI WYJŚCIOWE: X3
ACC = 0                              'wyzeruj akumulator
FOR H% = 1 TO 10                     'pomnóż przez wagi i zsumuj wszystkie węzły ukryte
   ACC = ACC + X2[H%] * WO[H%]
NEXT H%
X3 = 1 / (1 + EXP(-ACC))             'przepuść zsumowaną wartość przez sigmoidę
'
'                                    'WYZNACZ BŁĄD DLA TEJ LITERY, ELET
ELET = (CORRECT - X3)                'wyznacz błąd
IF CORRECT = 1 THEN ELET = ELET*5    'przydziel dodatkową wagę celom
'
RETURN

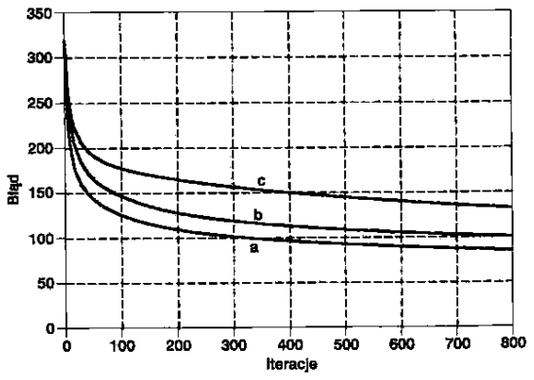
'PODPROGRAM DO WYZNACZANIA NOWYCH WAG
'Zmienne wchodzące do podprogramu: X1[ ], X2[ ], X3, WI[ , ], WH[ , ], ELET, MU
'Zmienne wychodzące z podprogramu: WI[ , ], WH[ , ]
'
'                                    'WYZNACZ NOWE WAGI DLA WARSTWY UKRYTEJ
FOR H%= 1 TO 10
   FOR I%= 1 TO 101
      SLOPEO = X3 * (1 - X3)
      SLOPEH = X2(H%) * (1 - X2[H%])
      DX3DW = X1[I%] * SLOPEH * WO[H%] * SLOPEO
      WH[H%,I%] = WH[H%,I%] + DX3DW * ELET * MU
   NEXT I%
NEXT H%
'
'                                    'WYZNACZ NOWE WAGI DLA WARSTWY WYJŚCIOWEJ
FOR H%= 1 TO 10
   SLOPEO = X3 * (1 - X3)
   DX3DW = X2[H%] * SLOPEO
   WO[H%] = WO[H%] + DX3DW * ELET * MU
NEXT H%
'
RETURN
```

Aby taka procedura iteracyjna działała, musi być jakiś *pojedynczy* parametr opisujący, jak dobrze w danym momencie działa system. Tę rolę w programie wypełnia zmienna ESUM (*error sum* – suma błędów). Pierwszą czynnością wewnątrz pętli iteracyjnej jest ustawienie zmiennej ESUM na zero (wiersz 290), aby mogła być ona użyta jako akumulator. Na końcu każdej iteracji wartość ESUM jest przekazywana do monitora wizyjnego (wiersz 380), aby operator wiedział, czy jest jakiś postęp w realizacji programu. Wartość ESUM najpierw staje się duża i zmniejsza się stopniowo w miarę trenowania sieci w rozpoznawaniu celów. Na rys. 26.9 pokazano przykłady zmniejszania się wartości ESUM wraz z przebiegającym procesem iteracji.

Podczas każdej iteracji ocenia się wszystkie 260 obrazów zawartych w zestawie trenującym w pętli FOR-NEXT, wiersze 310–360. Do wyszukiwania obrazów z bazy danych zawierających przykłady stosuje się podprogram 1000. Ponieważ szczegóły podprogramu nie są tu dla nas szczególnie interesujące, więc opiszemy tylko parametry wprowadzane do podprogramu i uzyskiwane z niego. Do podprogramu 1000 wchodzi się z parametrem LETTER% o wartościach między 1 i 260. Po powrocie, wartości węzłów wejściowych od X1[1] do X1[100] zawierają wartości pikseli obrazu w bazie danych odpowiadającego LETTER%. Wartość węzła polaryzującego X1[101] zawsze powraca ze stałą wartością *jeden*. Z podprogramu 1000 uzyskuje się też inny parametr, CORRECT. Zawiera on żądaną wartość wyjściową sieci dla tej określonej litery. A zatem, jeśli litera w obrazie jest samogłoską, to parametr CORRECT powróci z podprogramu z wartością *jeden*; a z wartością *zero*, jeśli litera w obrazie nie jest samogłoską.

Gdy opracowywany obraz zostanie już zapisany do matrycy od X1[1] do X1[100], podprogram 2000 przepuszcza dane przez aktualnie istniejącą sieć neuronową w celu uzyskania wartości węzła wyjściowego X3. Innymi słowy, podprogram 2000 jest taki sam, jak program w tablicy 26.1, z tą różnicą, że liczby węzłów w warstwach są różne. Ten podprogram oblicza też, jak dobrze aktualnie istniejąca sieć określa, czy litera jest, czy nie jest poszukiwanym celem. W wierszu 2210 zmienna ELET (*error-letter* – błąd litery) jest obliczana jako różnica między wartością wyjściową aktualnie generowaną X3 a pożądaną wartością CORRECT. Ta daje wartości ELET z zakresu od –1 do 1. Na podstawie wszystkich 260 wartości ELET uzyskuje się (wiersz 340) wartość ESUM, całkowity błąd kwadratowy sieci dla całego zestawu trenującego.

W wierszu 2220 pokazano opcję, która jest często włączana do obliczeń w celu przydzielania różnego stopnia *ważności* błędom dla obiektów będących lub niebędących celem. Przypomnijcie sobie na przykład omawiany wcześniej w tym rozdziale system wykrywania raka oraz konsekwencje wynikające z błędów typu „fałszywie pozytywny" w porównaniu



Rys. 26.9. Zbieżność sieci neuronowej. Na wykresach pokazano, jak błąd sieci neuronowej (wartość ESUM) zmniejsza się w miarę procesu iteracji. Przedstawiono wyniki trzech prób, z których każda zaczyna się od innej początkowej wartości wagi

z błędami typu „fałszywie negatywny". W rozważanym teraz przykładzie przyjmijmy w sposób dowolny, że błąd w wykryciu celu jest *pięciokrotnie* gorszy niż błąd w wykryciu obiektu niebędącego celem. W rezultacie to narzuca na sieć wymaganie, żeby lepiej działała w poszukiwaniu celu nawet kosztem gorszego wykrywania obiektów niebędących celem.

Podprogram 3000 jest sercem strategii stosowanej w sieciach neuronowych, algorytmem do zmiany wag po każdej iteracji. W celu wyjaśnienia podstaw matematycznych skorzystamy z pewnej analogii. Rozważmy dylemat skoczka spadochronowego zrzuconego między linie nieprzyjacielskie. Ląduje w nieznanym terenie, w takich ciemnościach, że widoczność jest ograniczona do kilku metrów. Otrzymał rozkaz posuwania się w kierunku dna najbliższej doliny, gdzie ma rozpocząć dalszą część swojej misji. Pojawia się jednak trudność – jak przy tak słabej widoczności znaleźć drogę do dna doliny. Można powiedzieć, że potrzebny jest algorytm określający położenie *x* i *y* spadochroniarza na powierzchni ziemi, aby móc *zminimalizować* wysokość tego położenia. Jest to zagadnienie analogiczne do ustawiania wag sieci neuronowej w taki sposób, żeby błąd sieci ESUM był minimalizowany.

W celu rozwiązania tego problemu weźmiemy pod uwagę dwa algorytmy: **ewolucji** (*evolution*) oraz **najszybszego spadku** (*steepest descent*). Według algorytmu ewolucji spadochroniarz wykonuje skokowy ruch w jakimś kierunku wybranym losowo. Jeśli nowe położenie znajduje się *wyżej* niż poprzednie, to spadochroniarz klnie i wraca do położenia początkowego. Jeśli nowe położenie znajduje się *niżej*, to spadochroniarz odczuwa początek sukcesu i powtarza procedurę, zaczynając od tego nowego położenia. Wreszcie osiągnie dno doliny, choć bardzo nieskuteczną i przypadkowo wybraną drogą. Tę metodę nazywa się *ewolucją*, gdyż jest to algorytm taki sam, jaki występuje w przyrodzie, w procesie ewolucji biologicznej. W każdym nowym pokoleniu danego gatunku pojawiają się jakieś losowe zmiany w stosunku do pokolenia poprzedniego. Takie zmiany, które działają na korzyść danego gatunku, mają większą szansę na to, aby były utrzymane i przekazane *następnemu* pokoleniu. Takie udoskonalenia umożliwiają zwierzętom zdobywać więcej pożywienia, skuteczniej uciekać przed wrogami, wydawać na świat więcej potomstwa itd. Jeśli zaś nowa cecha okaże się szkodliwa, to zwierzę nią dotknięte stanie się obiadem jakiegoś drapieżnika i ta zmiana zostaje odrzucona. W tym sensie każde nowe pokolenie jest iteracją w ewolucyjnej procedurze optymalizacji.

Przy zastosowaniu ewolucji jako algorytmu trenującego każda waga w sieci neuronowej jest nieco zmieniana przez dodanie pewnej wartości z generatora liczb losowych. Jeśli zmodyfikowane wagi ulepszają sieć (tj. dają mniejszą wartość zmiennej ESUM), to takie zmiany są utrzymywane, w innym razie są odrzucane. Wprawdzie ta procedura działa, ale jej **zbieżność** jest bardzo powolna. Pojęcie zbieżności jest używane do opisu ciągle postępującego ulepszania w kierunku rozwiązania optymalnego (dna doliny). Prościej mówiąc, na osiągnięcie rozwiązania program może potrzebować raczej dni, a nie minut czy godzin.

Na szczęście algorytm *najszybszego spadku* działa znacznie szybciej. Według tego algorytmu spadochroniarz zachowałby się w sposób naturalny. Oceniłby, która droga prowadzi *w dół* i poruszałby się w tym kierunku. Pomyślmy o tej sytuacji w sposób następujący. Spadochroniarz poruszy się o jeden krok na północ i zanotuje zmianę wysokości swojego położenia. Po powrocie do położenia początkowego zrobi jeden krok w kierunku wschodnim i znowu zarejestruje zmianę wysokości. Korzystając z tych uzyskanych wartości, może określić, jaki kierunek prowadzi w dół. Przyjmijmy że spadochroniarz obniżył się o 10 cm, robiąc krok na północ, a o 20 cm, robiąc krok ku wchodowi. Aby wędrował prosto w dół, jego droga wzdłuż każdej osi powinna być proporcjonalna do nachylenia wzdłuż tej osi. W omawianym przypadku mógłby przesunąć się na północ o 10 kroków i na wschód o 20 kroków. Dzięki temu przemieści się po najbardziej stromej części stoku o $\sqrt{10^2+20^2}=22{,}4$ kroków. Mógłby, alternatywnie, poruszać się po linii prostej do nowego położenia, robiąc 22,4 kroków po przekątnej. Najważniejsze jest to, że: *najszybszy spadek uzyskuje się przez poruszanie się wzdłuż każdej osi na odległość proporcjonalną do nachylenia wzdłuż tej osi.*

Taki sam algorytm najszybszego spadku zaimplementowano dla wag sieci neuronowej w podprogramie 3000. Przed wejściem w podprogram 3000, jeden z przykładów obrazu jest wprowadzany do warstwy wejściowej i ta informacja przechodzi na wyjście. To znaczy, że wszystkie wartości dla X1[ ], X2[ ] i X[3] są określone, podobnie jak aktualnie wartości wag WH[,] i WO[ ]. Ponadto znamy błąd ELET, jaki powstaje w sieci dla tego właśnie obrazu. Wagi warstwy ukrytej są uaktualniane w wierszach od 3050 do 3120, wagi zaś warstwy wyjściowej w wierszach od 3150 do 3190. *Jest to realizowane przez obliczanie nachylenia dla każdej wagi i następnie zmienianie każdej wagi o wartość proporcjonalną do tego nachylenia*. W przypadku spadochroniarza nachylenie wzdłuż osi jest wyznaczane przez przesuwanie się o małą odległość wzdłuż osi (powiedzmy, Δx), pomiar zmiany wysokości (powiedzmy, ΔE) i następnie podzielenie tych dwóch wartości przez siebie ($\Delta E/\Delta x$). Nachylenie wagi sieci neuronowej można wyznaczyć w taki sam sposób – przez dodanie małego przyrostu do wartości wagi (Δw), określenie powstałej zmiany sygnału wyjściowego ($\Delta X3$) i podzielenie tych dwóch wartości ($\Delta X3/\Delta w$). W dalszej części tego rozdziału rozpatrzymy przykład, w którym nachylenie jest obliczane właśnie w ten sposób. Jednak w przykładzie teraz omawianym zastosujemy skuteczniejszą metodę.

Powiedzieliśmy poprzednio, że funkcja nieliniowa (sigmoida) powinna być *różniczkowalna*. Teraz właśnie skorzystamy z tej właściwości. Jeśli znamy nachylenie w każdym punkcie przebiegu nieliniowego, możemy bezpośrednio zapisać wzór na nachylenie każdej z wag ($\Delta X3/\Delta w$) bez konieczności jej zaburzania. Rozważmy pewną określoną wagę, np. WO[1], odpowiadającą pierwszemu wejściu węzła wyjściowego. Spójrzcie na struktury przedstawione na rys. 26.5 i 26.6 i spytajcie, jaki wpływ na wyjście ($X3$) będzie miała niewielka zmiana tej określonej wagi (w), jeśli wszystko poza tym pozostanie bez zmian. A oto odpowiedź na to pytanie:

$$\frac{\Delta X3}{\Delta w} = X2[1]\text{SLOPE}_0$$

Wzór 26.3. Nachylenie wag warstwy wyjściowej. Wzór zapisano dla wagi WO[1]

gdzie SLOPE_0 jest pierwszą pochodną sigmoidy warstwy wyjściowej, obliczoną na krzywej w punkcie pracy. Innymi słowy, SLOPE_0 opisuje, o ile zmienia się sygnał *wyjściowy* sigmoidy w odpowiedzi na zmianę na *wejściu* tej sigmoidy. Wartość SLOPE_0 można obliczyć z wzoru na podstawie aktualnej wartości X3 sigmoidy. To obliczenie przedstawiono w wierszu 3160. W wierszu 3170 nachylenie tej wagi jest obliczane z wzoru 26.3 i rejestrowane jako zmienna DX3DW (tj. $\Delta X3/\Delta w$).

Podobnie oblicza się nachylenie wagi warstwy ukrytej, takiej jak WH[1,1], korzystając z wzoru

$$\frac{\Delta X3}{\Delta w} = X1[1]\text{SLOPE}_{H1}WO[1]\text{SLOPE}_0$$

Wzór 26.4. Nachylenie wag warstwy ukrytej. Wzór zapisano dla wagi WH[1,1]

W powyższym wzorze SLOPE_{H1} jest pierwszą pochodną sigmoidy warstwy ukrytej, obliczoną w punkcie pracy na tej krzywej. Inne wielkości, X1[1] i WO[1], są po prostu pewnymi stałymi, napotykanymi przez zmianę wagi w jej drodze do wyjścia sieci. W wierszach 3070 i 3080 następuje obliczanie nachyleń sigmoidy według wzoru 26.2. Nachylenie wagi warstwy ukrytej DX3DW jest obliczane w wierszu 3090 z wzoru 26.4.

Teraz, znając już *nachylenie* każdej z wag, możemy popatrzeć, jak każda waga zmienia się w wyniku następnej iteracji. Nową wartość każdej wagi wyznacza się, biorąc wagę aktualną i dodając do niej pewną wartość proporcjonalną do nachylenia:

$$w_{new} = w_{old} + \frac{\Delta X3}{\Delta w}\text{ELET MU}$$

Wzór 26.5. Uaktualnianie wag. Każda z wag jest uaktualniania przez dodanie do niej pewnej wartości proporcjonalnej do nachylenia tej wagi

To obliczenie jest wykonywane w wierszu 3180 dla warstwy wyjściowej. Stała proporcjonalności składa się z dwóch czynników: ELET – błędu sieci dla tego, określonego sygnału wejściowego oraz MU – stałej ustalonej na początku programu. W celu zrozumienia potrzeby użycia wartości błędu ELET w tym obliczeniu, wyobraźmy sobie, że obraz umieszczony na wejściu wytwarza *mały* błąd w sygnale wyjściowym. Następnie wyobraźmy sobie, że inny obraz wprowadzony na wejście wytwarza *duży* błąd wyjściowy. Ustawiając wagi, chcemy bardziej zmienić działanie sieci dla drugiego obrazu niż dla pierwszego. Jeśli coś kiepsko działa, to chcemy to zmienić; jeśli zaś działa dobrze, to zostawimy sprawę w spokoju. Tę ideę realizuje się przez zmienianie każdej wagi proporcjonalnie do aktualnej wartości błędu ELET.

Aby zrozumieć, jak stała MU wpływa na system, wróćcie do przykładu ze spadochroniarzem. Gdy spadochroniarz określi kierunek opadania stoku, musi zdecydować, jak daleko powinien przesunąć się w tym kierunku przed ponownym oszacowaniem nachylenia terenu. Jeśli zdecyduje się na przesunięcie na małą odległość, np. 1 metr, to będzie z dużą dokładnością śledził kontury terenu i zawsze przemieszczał się w optymalnym kierunku. Trudność polega na tym, że wtedy spadochroniarz musiałby większość czasu poświęcać na szacowanie nachylenia stoku, zamiast na poruszanie się po nim. Może też wybrać odległość dużą, np. 1000 metrów. To wprawdzie umożliwi spadochroniarzowi bardzo szybkie poruszanie się w terenie, ale spowoduje niebezpieczeństwo przegapienia drogi w dół. Zbyt duża odległość może spowodować skakanie po całym terenie bez uzyskania żądanego postępu.

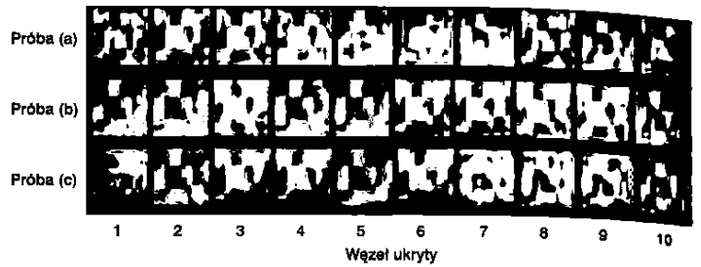
W sieci neuronowej stała MU decyduje o zmianach wag przy każdej iteracji. Zastosowana wartość zmiany zależy od rozwiązywanego problemu i może być nawet tak mała jak 10^{-6} i tak duża jak 0,1. Przez analogię z zadaniem ze spadochroniarzem można się obawiać, że zbyt mała wartość zmiany spowoduje, że zbieżność sieci będzie bardzo powolna. Natomiast zbyt duża zmiana spowoduje zbieżność nierównomierną, wywołując chaotyczne oscylacje wokół końcowego rozwiązania. Niestety sposób reakcji sieci neuronowej na różne wartości MU może być trudny do zrozumienia lub przewidzenia. To powoduje, że bardzo ważne jest monitorowanie błędu (tj. ESUM) podczas treningu, np. przekazywanie wartości ESUM do monitora wizyjnego po zakończeniu każdej iteracji. Jeśli system nie wykazuje właściwej zbieżności, to trzeba zatrzymać program i spróbować zastosować inną wartość MU.

Ocena wyników

A więc jak to działa? Program trenujący do rozpoznawania samogłosek wykonano trzykrotnie, stosując różne wartości losowe jako wagi początkowe. Trzeba ok. 10 minut do zrealizowania 800 iteracji na komputerze osobistym Pentium 1 GHz.

Na rys. 26.9 pokazano, jak w tym czasie zmienia się błąd sieci ESUM. Stopniowe opadanie przebiegu wskazuje, że sieć uczy się zadania i że wagi osiągają wartości bliskie optymalnych po kilkuset iteracjach. Każda próba daje inne rozwiązanie problemu, o różnym końcowym wyniku. Jest to analogiczne do spadochroniarza rozpoczynającego z różnych położeń, a więc i kończącego na dnie różnych dolin. Ponieważ niektóre doliny są głębsze od innych, więc niektóre rozwiązania sieci neuronowych są lepsze od innych. Wynika stąd, że algorytm uczący się powinien być wykonywany kilka razy, a najlepsza realizacja wybrana jako rozwiązanie końcowe.

Na rys. 26.10 pokazano, jako wyświetlone obrazy, wagi warstwy ukrytej dla trzech różnych rozwiązań. Pierwszym działaniem podejmowanym przez sieć neuronową jest korelacja (mnożenie i sumowanie) tych obrazów z sygnałem wejściowym. Wyglądają one jak szum losowy! Można *pokazać*, że te wagi działają, lecz to, *dlaczego* działają, ma w sobie coś z tajemnicy. Jeszcze jedna sprawa jest zastanawiająca. Ludzki mózg składa się z ok. 100 *bilionów* neuronów, z których każdy ma średnio 10 000 połączeń. Jeśli nie jesteśmy w stanie zrozumieć tej prostej sieci neuronowej w tym przykładzie, to jak moglibyśmy przeanalizować system będący co najmniej 100 000 000 000 000 razy bardziej skomplikowany. To są badania na miarę XXI wieku!



Rys. 26.10. Przykład wag sieci neuronowej. Na tym rysunku wagi dla trzech różnych rozwiązań (prób) są wyświetlone jako obrazy. Patrząc na te obrazy, odnosi się wrażenie, że wszystkie trzy rozwiązania mają charakter losowy

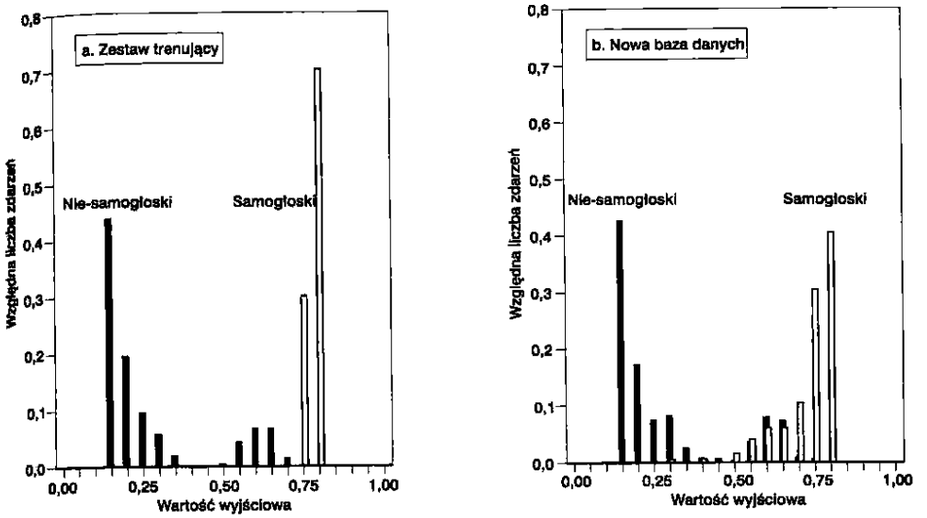
Na rys. 26.11 a) pokazano histogram sygnału wyjściowego sieci neuronowej dla 260 liter w zestawie trenującym. Pamiętajcie, że wagi wybrano w taki sposób, aby sygnał wyjściowy był bliski *jeden* dla obrazów samogłosek, a w innych przypadkach bliski *zeru*. Uzyskano doskonałe rozdzielenie, bez żadnego nakładania się obu rozkładów. Zauważcie też, że rozkład samogłosek jest węższy niż nie-samogłosek. Przyczyną tego jest zadeklarowanie przez nas, że błąd w wykryciu celu jest pięciokrotnie ważniejszy od błędu w wykryciu obiektu niebędącego celem (patrz wiersz 2220).

Na rys. 26.11 b) pokazano histogramy dla obrazów od 261 do 1300 w bazie danych. Wprawdzie rozkłady dla obiektów będących i niebędących celem są dość wyraźne, to nie są one jednak całkiem rozdzielone. Dlaczego sieć neuronowa działa lepiej na pierwszych 260 literach niż na ostatnich 1040? Rys. 26.11 a) jest mylący! Łatwo wykonać test, gdy już się widziało odpowiedzi. Inaczej mówiąc, sieć neuronowa rozpoznaje specyficzne obrazy w zestawie trenującym, a nie ogólne wzorce rozróżniające samogłoski od nie-samogłosek.

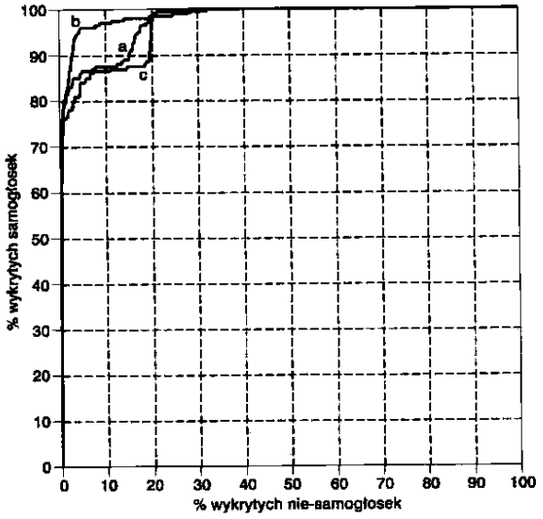
Na rys. 26.12 przedstawiono, w postaci wykresów ROC, wyniki uzyskane przy użyciu tych trzech rozwiązań. Próba b) daje sieć wyraźnie lepszą niż dwie pozostałe próby (a, c). Jest to kwestia losowego przypadku zależnego od zastosowanych wag początkowych. Przy jednym ustawieniu progu sieć neuronowa zaprojektowana w wyniku próby b) może wykryć 24 z 25 celów (tj. 96% obrazów samogłosek), z występowaniem fałszywych alarmów tylko raz na 25 obiektów niebędących celami (tj. 4% obrazów nie-samogłosek). To nieźle, jeśli uwzględnić abstrakcyjny charakter tego problemu i bardzo ogólne rozwiązanie, jakie zastosowano.

I jeszcze kilka końcowych komentarzy na temat sieci neuronowych. Uzyskanie zbieżności sieci neuronowej podczas jej trenowania bywa trudne. Jeśli błąd sieci (ESUM) nie zmniejsza się w sposób ustalony, to program trzeba zakończyć, zmienić i rozpocząć jego wykonywanie od nowa. Może to wymagać kilku prób, zanim się powiedzie. W celu osiągnięcia zbieżności można zmieniać trzy wielkości: 1) stałą MU, 2) wartość początkowych wag losowych oraz 3) liczbę ukrytych węzłów. Te wielkości podano tu w takiej kolejności, w jakiej powinny być zmieniane.

Najbardziej krytyczną sprawą w opracowaniach sieci neuronowych jest *ważność* przykładów trenujących. Na przykład podczas opracowywania nowych produktów rynkowych jedyne dostępne dane testowe pochodzą z prototypów, symulacji, przypuszczeń itd. Sieć neuronowa trenowana na podstawie tych wstępnych informacji może nie działać prawidłowo w końcowym zastosowaniu. Każda różnica między trenującą bazą danych a danymi ostatecznymi pogorszy działanie sieci neuronowej (prawo Murphy'ego dla sieci neuronowych). Z tą siecią neuronową nie próbuj po raz drugi takiego zgadywania; nie potrafisz!



Rys. 26.11. Działanie sieci neuronowej. Są to histogramy wartości sygnałów wyjściowych sieci neuronowej; a) dla danych trenujących, b) dla pozostałych obrazów. Sieć neuronowa lepiej działa w przypadku danych trenujących, ponieważ widziała już odpowiedzi na ten test



Rys. 26.12. Analiza krzywych ROC na przykładach sieci neuronowych. Te krzywe dotyczą trzech sieci neuronowych zaprojektowanych do wykrywania obrazów samogłosek. Najlepszym rozwiązaniem jest wykres b), gdyż krzywa znajduje się najbliżej prawego górnego rogu wykresu. Ta sieć może prawidłowo wykrywać 24 z 25 celów, dając tylko jeden fałszywy alarm na 25 obiektów niebędących celami. Jest to więc punkt na krzywej ROC o współrzędnych x = 4% i y = 96%

Projektowanie filtru rekursywnego

W rozdziałach 19 i 20 pokazano, jak się projektuje filtry rekursywne o standardowych odpowiedziach częstotliwościowych: górno- , dolno-, pasmowoprzepustowe itd. A co zrobić, jeśli trzeba wykonać filtr według specjalnych wymagań? Trzeba wtedy zaprojektować filtr rekursywny tak, jakby się projektowało sieć neuronową – zacząć od ogólnego zestawu współczynników rekursywnych i zastosować iterację, aby powoli ukształtować je w to, czego chcemy. Ta metoda jest ważna z dwóch powodów. Po pierwsze, umożliwia projektowanie filtrów rekursywnych bez konieczności szamotania się z matematycznymi operacjami przekształcenia Z. Po drugie, wykazuje, że łącząc ze sobą pomysły z konwencjonalnego DSP i z sieci neuronowych, uzyskuje się świetne algorytmy.

Główny program do tej metody podano w tablicy 26.4, a dwa podprogramy – w tablicy 26.5. W matrycy T[ ] jest przechowywana żądana odpowiedź częstotliwościowa, czyli jakiś wykres, który ręcznie zaprojektowaliśmy. Ponieważ w programie korzysta się z FFT, więc długości sygnałów muszą być potęgami dwóch. W programie zastosowano przekształcenie FFT o długości 256, co zostało zdefiniowane przez zmienną N% w wierszu 130. To znaczy, że komórki od T[0] do T[128] odnoszą się do częstotliwości między 0 a 0.5 częstotliwości próbkowania. W tej matrycy są zapisane tylko moduły, faza w tym przykładzie nie jest wyznaczana i będzie, jaka będzie.

Tab. 26.4.

```
100 'ITERACYJNE PROJEKTOWANIE FILTRU REKURSYWNEGO
110 '
120                                     'INICJALIZACJA
130 N% = 256                            'liczba punktów w FFT
140 NP% = 8                             'liczba pól w filtrze
150 DELTA = .00001                      'przyrost zakłócenia
160 MU = .2                             'wielkość kroku iteracji
170 DIM REX[255]                        'część rzeczywista sygnału podczas FFT
180DIM IMX[255]                         'część urojona sygnału podczas FFT
190 DIM T[128]                          'żądana odpowiedź częstotliwościowa (tylko moduł)
200 DIM A[8]                            'współczynniki rekursywne "a"
210 DIM B[8]                            'współczynniki rekursywne "b"
220 DIM SA[8]                           'nachylenie współczynników "a"
230 DIM SB[8]                           'nachylenie dla współczynników "b"
240 '
250 GOSUB XXXX                          'podprogram "mityczny" do ładowania T[ ]
260 '
270 FOR P% = 0 TO NP%                   'inicjalizuj współczynniki dla systemu tożsamościowego
280   A[P%] = 0
290   B[P%] = 0
300 NEXT P%
310 A[0] = 1
320 '
330 '                                   'PĘTLA ITERACJI
340 FOR ITER%= 1 TO 100                 'pętla dla żądanej liczby iteracji
350   GOSUB 2000                        'oblicz nowe współczynniki
360    PRINT  ITER%  ENEW  MU           'pokaż stan aktualny na monitorze ekranowym
370    IF ENEW > EOLD THEN MU = MU/2    'dobierz wartość MU
380 NEXT ITER%
390 '
400 '
410 FOR P% = 0 TO NP%                   'WYDRUKUJ WSPÓŁCZYNNIKI
420    PRINT  A[P%]     B[P%]
430 NEXT P%
440 '
450 END
```


Tab. 26.5.

```
2000 'PODPROGRAM DO OBLICZANIA NOWYCH WSPÓŁCZYNNIKÓW REKURSYWNYCH
2010 'Zmienne wchodzące do podprogramu: A[ ], B[ ], DELTA, MU
2020 'Zmienne wychodzące z podprogramu: A[ ], B[ ], EOLD, ENEW
2030 '
2040 GOSUB 3000                         'WYZNACZ BIEŻĄCĄ WARTOŚĆ BŁĘDU
2050 EOLD = ER                          'zapamiętaj bieżący błąd w zmiennej EOLD
2060 '
2070                                    'WYZNACZ NACHYLENIA BŁĘDÓW
2080 FOR P% = 0 TO NP%                  'pętla przez wszystkie współczynniki "a"
2090   A[P%] = A[P%] + DELTA            'dodaj mały przyrost do współczynnika
2100   GOSUB 3000                       'wyznacz wartość błędu z tą zmianą
2110   SA[P%] = (ER-EOLD)/DELTA         'oblicz nachylenie błędu, zapamiętaj w SA[ ]
2120   A[P%] = A[P%] - DELTA            'wróć do oryginalnej wartości współczynnika
2130 NEXT P%
2140 '
2150 FOR P% = 1 TO NP%                  'powtórz tę procedurę dla każdego współczynnika "b"
2160   B[P%] = B[P%] + DELTA
2170   GOSUB 3000
2180   SB[P%] = (ER-EOLD)/DELTA         'oblicz nachylenie błędu, zapamiętaj w SB[ ]
2190   B[P%] = B[P%] - DELTA
2200 NEXT P%
2210 '                                  'OBLICZ NOWE WSPÓŁCZYNNIKI
2220 FOR P% = 0 TO NP%                  'pętla przez wszystkie współczynniki
2230   A[P%] = A[P%] - SA[P%] * MU      'dobierz współczynniki tak, aby poruszać się "w dół"
2240   B[P%] = B[P%] - SB[P%] * MU
2250 NEXT P%
2260 ,
2270 GOSUB 3000                         'WYZNACZ NOWY BŁĄD
2280 ENEW = ER                          'zapamiętaj nowy błąd w zmiennej ENEW
2290 '
2300 RETURN

3000 'PODPROGRAM DO OBLICZANIA BŁĘDU W DZIEDZINIE CZĘSTOTLIWOŚCI
3010 'Zmienne wchodzące do podprogramu: A[ ], B[ ], T[ ]
3020 'Zmienne wychodzące z podprogramu: ER
3030 '
3040 FOR I% = 0 TO N%-1                 'ŁADUJ PRZESUNIĘTY IMPULS DO IMX[ ]
3050 REX[I%] = 0
3060 IMX[I%] = 0
3070 NEXT I%
3080 IMX[12]= 1
3090 '                                  'OBLICZ ODPOWIEDŹ IMPULSOWĄ
3100 FOR I%= 12  TO N%-1
3110    FOR J% = 0 TO NP%
3120       REX[I%] = REX[I%] + A[J%] * IMX[I%-J%] + B[J%] * REX[I%-J%]
3130    NEXT J%
3140 NEXT I%
3150 IMX[12] = 0
3160 '                                  'OBLICZ FFT
3170 GOSUB 1000                         'Tablica 12.4, używa REX[ ], IMX[ ], N%
3180'
3190                                    'WYZNACZ BŁĄD W DZIEDZINIE CZĘSTOTLIWOŚCI
3200 ER = 0                             'wyzeruj ER, aby użyć go jako akumulatora
3210 FOR I% = 0 TO N%/2                 'pętla przez wszystkie częstotliwości dodatnie
3220   MAG = SQR(REX[I%]^2 + IMX[I%]^2)'konwersja: prostokątne -- > biegunowe
3230   ER = ER + ( MAG - T[I%] )^2      'oblicz i zakumuluj kwadrat błędu
3240 NEXT I%
3250 ER = SQR( ER/(N%/2+1) )            'zakończ obliczanie błędu, ER
3260 '
3270 RETURN
```

W wierszach 270–310 ustawia się współczynniki rekursywne na ich wartości początkowe, typowo wybrane w taki sposób, aby tworzyły system *tożsamościowy*. Nie używajcie tu liczb losowych, gdyż początkowy filtr będzie wtedy niestabilny. Współczynniki rekursywne są przechowywane w matrycach A[ ] i B[ ]. Zmienna NP% ustala liczbę biegunów w projektowanym filtrze. Na przykład, jeśli NP% jest 5, to współczynnik „a" zmienia się od A[0] do A[5], a współczynnik „b" – od B[1] do B[5].

Jak wspomniano poprzednio, procedura iteracyjna wymaga *pojedynczej* wartości opisującej, jak dobrze działa aktualny system. Uzyskuje się to przez zmienną ER (błąd) obliczaną w podprogramie 3000. W wierszach od 3040 do 3080 następuje ładowanie impulsu do matrycy IMX[ ]. W następnych wierszach 3100–3150 ten impuls jest wykorzystywany jako sygnał wejściowy filtru rekursywnego określonego bieżącymi wartościami z A[ ] i B[ ]. Wyjście tego filtru jest więc *odpowiedzią impulsową* aktualnego systemu, która jest przechowywana w matrycy REX[ ]. Następnie jest znajdowana odpowiedź częstotliwościowa systemu przez wyznaczenie przekształcenia FFT, jak to pokazano w wierszu 3170. Podprogram 1000 jest programem FFT podanym w tablicy 12.4 w rozdziale 12. Ten podprogram FFT powraca do odpowiedzi częstotliwościowej we współrzędnych prostokątnych, nadpisując dane do matryc REX[ ] i IMX[ ].

Następnie w wierszach 3200–3250 jest obliczana wartość ER, *średni błąd kwadratowy* – błąd między modułami bieżącej i żądanej odpowiedzi częstotliwościowej. Szczególną uwagę zwróćcie na sposób obliczania błędu. Działanie iteracyjne programu optymalizuje ten błąd, w wyniku czego bardzo ważny staje się sposób jego zdefiniowania. Pętla FOR–NEXT przebiega przez każdą częstotliwość w odpowiedzi częstotliwościowej. Dla każdej częstotliwości, w wierszu 3220 następuje obliczenie modułu bieżącej odpowiedzi częstotliwościowej na podstawie danych we współrzędnych prostokątnych. W wierszu 3230 jest wyznaczany błąd dla tej częstotliwości, przez odejmowanie żądanego modułu T[ ] od jego wartości bieżącej MAG. Ten błąd jest następnie podnoszony do kwadratu i dodawany do zmiennej akumulatora ER. Po przebiegnięciu pętli przez wszystkie częstotliwości, w wierszu 3250 obliczanie zostaje zakończone, dając w wyniku ER – średni błąd kwadratowy całej odpowiedzi częstotliwościowej.

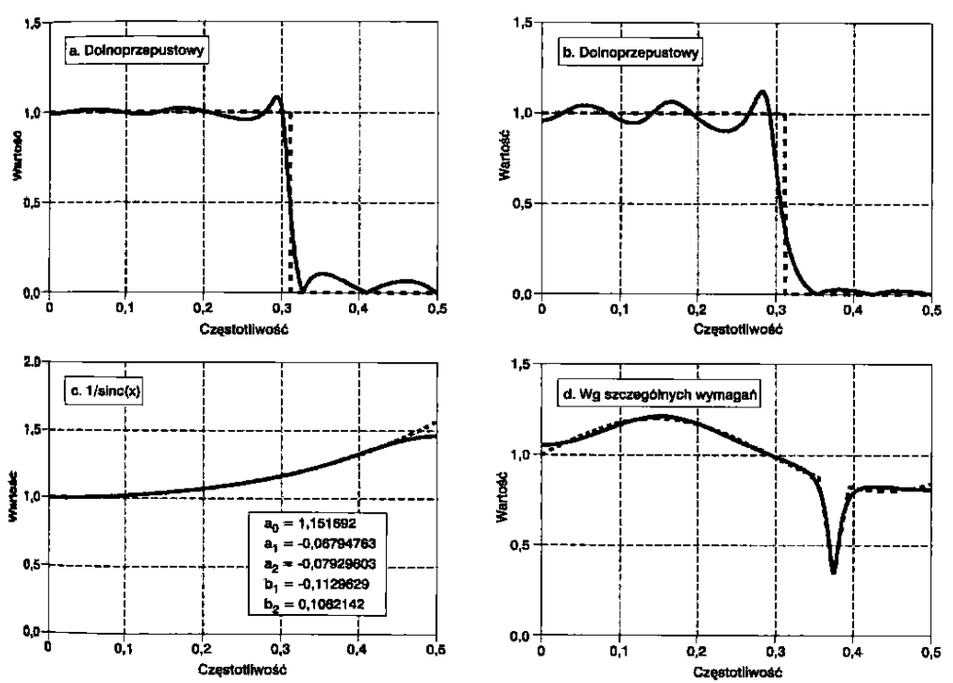
Wiersze od 340 do 380 sterują pętlą iteracyjną programu. Zmiany współczynników rekursywnych są wykonywane w podprogramie 2000. Pierwszym działaniem w tym podprogramie jest określenie bieżącej wartości ER i zapisanie jej w innej zmiennej EOLD (wiersze 2040 i 2050). Gdy podprogram już uaktualni współczynniki, to wartość ER jest określana na nowo i przypisywana do zmiennej ENEW (wiersze 2270 i 2280).

Zmienna MU decyduje o rozmiarze kroku iteracji, tak jak w poprzednio omówionym programie sieci neuronowej. W tym programie zastosowano pewne udoskonalenie: *automatyczne* dopasowanie do wartości MU. Właśnie dlatego mamy tu dwie zmienne, EOLD i ENEW. Na początku programu zmienna MU jest ustawiana na dość dużą wartość 0,2 (wiersz 160). Umożliwia to wprawdzie szybką zbieżność, lecz ogranicza bliskość dojścia filtru do rozwiązania optymalnego. W miarę kolejnych iteracji dochodzi się do punktów, gdzie nie ma już żadnej poprawy, co jest identyfikowane faktem, że ENEW jest *większe* niż EOLD. Za każdym razem, gdy to nastąpi, wiersz 370 powoduje zmniejszenie wartości MU.

Podprogram 2000 uaktualnia współczynniki rekursywne według metody najszybszego spadku – dla każdego współczynnika jest obliczane nachylenie i następnie ten współczynnik jest zmieniany o wartość proporcjonalną do jego nachylenia. W wierszach 2080–2130 są obliczane nachylenia dla współczynników „a", rejestrowane w matrycy SA[ ]. Podobnie, w wierszach 2150–2200 są obliczane nachylenia współczynników „b", rejestrowane w matrycy SB[ ]. Potem w wierszach 2220–2250 następuje modyfikacja wszystkich współczynników rekursywnych o wartości proporcjonalnej do ich nachyleń. W tym programie współczynnikiem proporcjonalności jest po prostu wartość kroku MU. W stałej proporcjonalności nie jest potrzebna żadna składowa błędu, gdyż jest tu tylko *jeden* przykład, do którego trzeba się dopasować – żądana odpowiedź częstotliwościowa.

Ostatnią kwestią jest sposób obliczania nachyleń współczynników rekursywnych. W przykładzie sieci neuronowej był wyprowadzony *wzór* na nachylenie. Tu nie można zastosować takiej procedury, gdyż wymagałoby to obliczania pochodnej *przez* DFT. Zamiast tego stosuje się prostszą metodę – zmienia się współczynnik rekursywny o niewielki przyrost i potem bezpośrednio oblicza nową wartość ER. Nachylenie wyznacza się wtedy jako zmianę ER podzieloną przez wartość przyrostu. Bieżąca wartość ER jest znajdowana w wierszach 2040–2050 i przechowywana w zmiennej EOLD. Pętla w wierszach 2080–2130 przebiega przez każdy ze współczynników „a". Pierwszym działaniem wewnątrz tej pętli jest dodanie małego przyrostu DELTA do współczynnika właśnie zmienianego (wiersz 2090). Podprogram 3000 jest przywoływany w wierszu 2100 w celu znalezienia wartości ER dla zmodyfikowanego współczynnika. Następnie w wierszu 2110 jest obliczane nachylenie tego współczynnika jako: *(ER – EOLD)/DELTA*. W wierszu 2120 wartość współczynnika jest odtwarzana przez odjęcie wartości DELTA.

Na rys. 26.13 przedstawiono kilka przykładów filtrów zaprojektowanych przy użyciu tego programu. Linia przerywana jest żądaną odpowiedzią częstotliwościową, a linia ciągła odpowiedzią częstotliwościową zaprojektowanego filtru. Każdy z tych filtrów wymaga około minuty na osiągnięcie zbieżności, na komputerze Pentium 1 GHz. Na rys. 26.13 a) przedstawiono 8-biegunowy filtr dolnoprzepustowy, w którym błąd jest równo ważony w całym widmie częstotliwości (taki program, jaki mamy napisany). Na rys. 26.13 b) pokazano taki sam filtr, z tą różnicą, że błąd w paśmie zaporowym jest pomnożony przez *osiem* przy obliczaniu ER. To wymusza w filtrze mniejsze pofalowania w paśmie zaporowym, kosztem większych pofalowań w paśmie przepustowym.

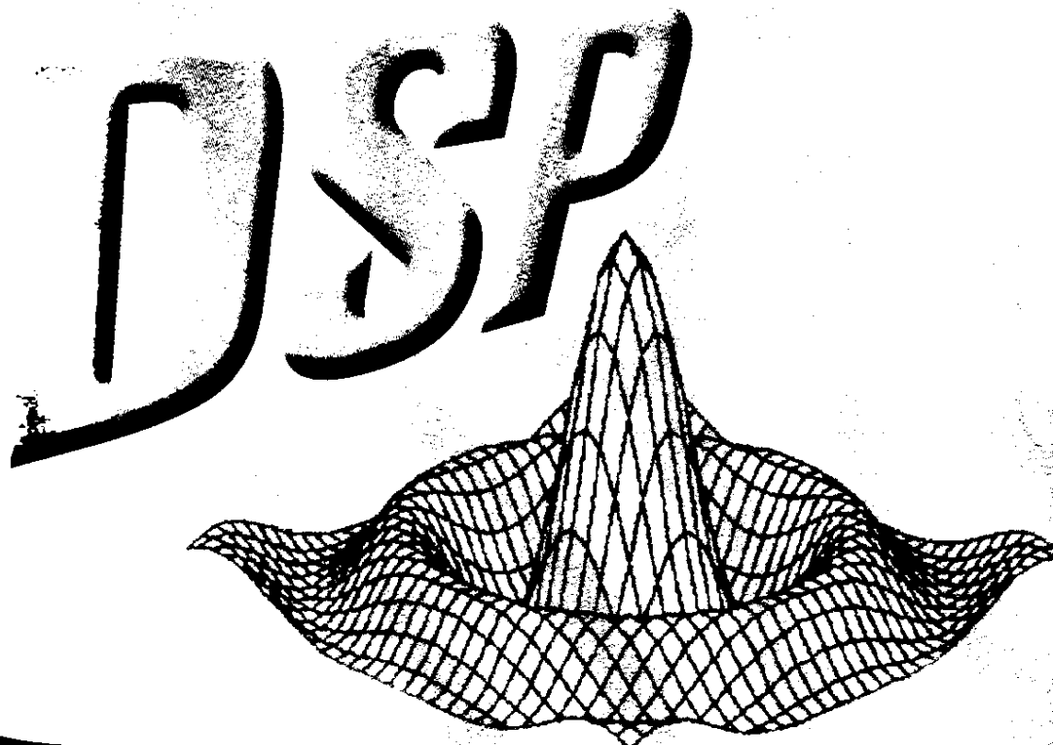


Rys. 26. 13. Iteracyjne projektowanie filtrów rekursywnych. Na rys. a) pokazano 8-biegunowy filtr dolnoprzepustowy z błędem równomiernie rozłożonym między 0 i 0,5. Na rys. b) błąd wyważono w taki sposób, aby uzyskać lepsze działanie w paśmie zaporowym kosztem błędu w paśmie przepustowym. Na rys. c) pokazano filtr 2-biegunowy stosowany do korekcji 1/sinc(x) w przetwarzaniu cyfrowo-analogowym. Odpowiedź częstotliwościowa na rys. d) jest zaprojektowana całkowicie według szczególnych wymagań. Na każdym rysunku żądana odpowiedź częstotliwościowa jest wykreślona linią przerywaną, a uzyskana – linią ciągłą

Na rys. 26.13 c) przedstawiono filtr 2-biegunowy dla $1/sinc(x)$. Jak już powiedziano w rozdziale 3, taki filtr może być stosowany do przeciwdziałania wpływowi pamiętania rzędu zerowego podczas przetwarzania cyfrowo-analogowego (patrz rys. 3.6). Błąd w tym filtrze został zsumowany tylko między 0 i 0,45, dając lepsze dopasowanie w tym obszarze kosztem gorszego między 0,45 i 0,5. Wreszcie na rys. 26.13 d) pokazano bardzo nieregularną, 6-biegunową odpowiedź częstotliwościową zawierającą ostre wgłębienie. Dla uzyskania zbieżności ustawiono początkowo współczynniki rekursywne na wartości współczynników filtru pasmowozaporowego (tzw. filtru wycinającego).

27

Kompresja danych



Przesyłanie i przechowywanie danych wymaga kosztów, tym większych, im więcej informacji jest przesyłanych i przechowywanych. Mimo to większość danych cyfrowych nie jest przechowywana w postaci najbardziej zagęszczonej. Są raczej przechowywane w sposób najdogodniejszy do ich wykorzystania, jak np. tekst ASCII z procesorów tekstu, kod dwójkowy – łatwy do operowania w komputerach, indywidualne próbki z systemu zbierania danych itd. Takie, dogodne do zastosowań, metody kodowania wymagają zazwyczaj dwa razy większej objętości danych niż rzeczywiście konieczna do reprezentowania określonej informacji. Kompresja danych to pojęcie ogólne, obejmujące różne algorytmy i programy opracowywane w celu przezwyciężenia tych trudności. *Program kompresji* jest stosowany do przetwarzania danych z formatu dogodnego do zastosowań na format zoptymalizowany pod względem gęstości upakowania. *Program dekompresji* zaś przywraca informacji jej postać oryginalną. W tym rozdziale najpierw omówimy pięć metod kompresji danych. Pierwsze trzy są prostymi sposobami kodowania, zwanymi kodowaniem ciągów (lub kompresją grupową, *run-length coding*), kodowaniem Huffmana i kodowaniem delta. Dwie ostatnie metody to procedury bardziej rozbudowane, które stały się standardami komercyjnymi – LZW i JPEG.

Sposoby kompresji danych

W tablicy 27.1 przedstawiono dwa sposoby podziału algorytmów kompresji danych na kategorie. W części a) tablicy metody sklasyfikowano jako **bezstratne** albo **stratne**. W technice bezstratnej odtworzony plik danych jest *identyczny* z oryginalnym. Jest to bezwzględnie konieczne dla wielu rodzajów danych, takich jak kody wykonywalne, pliki tekstowe, tablice liczbowe itd. W tego typu informacjach nie można sobie pozwolić na niewłaściwe umieszczenie nawet pojedynczego bitu. Natomiast pliki danych reprezentujących obrazy lub inne zbierane sygnały nie muszą być zachowane w tak doskonałym stanie podczas przesyłania lub przechowywania. Wszystkie pomiary wykonywane w realnym świecie są w sposób nieodłączny obarczone pewnym *szumem*. Jeśli zmiany, jakim jest poddawany sygnał, przypominają niewielki dodatkowy szum, to nie jest to szkodliwe. Metody kompresji, w których dopuszczalne jest takie pogorszenie jakości sygnału, są nazywane **stratnymi**. Takie rozróżnienie jest istotne, gdyż metody stratne dają kompresję znacznie skuteczniejszą niż bezstratne. Im większy jest współczynnik kompresji, tym więcej jest szumu dodawanego do danych.

Obrazy przesyłane w ogólnoświatowej sieci komputerowej *WWW* są świetnym przykładem znaczenia kompresji danych. Przypuśćmy, że chcemy pobrać cyfrową fotografię kolorową przez modem komputera o przepływności 33,6 kbit/s. Jeśli obraz nie jest poddany kompresji (na przykład plik *TIF*), to będzie zawierał ok. 600 kB danych. Jeśli poddano go kompresji przy użyciu jednej z metod *bezstratnych* (takiej np. jak w formacie *GIF*), to danych będzie mniej więcej o połowę mniej, czyli 300 kB. W przypadku zastosowania kompresji *stratnej* (plik JPEG), będzie to ok. 50 kB. Problem w tym, że czasy pobierania dla tych trzech rów-

Tablica 27.1. Klasyfikacja metod kompresji. Metody kompresji danych można dzielić dwoma sposobami. W tablicy a) metody podzielono na bezstratne i stratne. W metodach bezstratnych skomprymowane dane są odtwarzane dokładnie w takiej samej postaci, jaką miały dane oryginalne. W metodach stratnych zaś uzyskuje się tylko dane przybliżone. W tablicy b) metody sklasyfikowano według kryterium ustalonego lub zmiennego rozmiaru grupy pobieranej z pliku oryginalnego i wpisywanej do pliku skomprymowanego

Metody bezstratne	Metody stratne
Kodowanie ciągów	CS&Q
Kodowanie Huffmana	JPEG
Kodowanie delta	MPEG
LZW	

a. Metody stratne i bezstratne

Metoda		
CS&Q	ustalony	ustalony
Huffmana	ustalony	zmienny
Arytmetyczna	zmienny	zmienny
Kodowanie ciągów, LZW	zmienny	ustalony

b. Ustalony lub zmienny rozmiar grupy

noważnych plików są, odpowiednio, 142 s, 71 s i 12 s. To bardzo duża różnica. Kompresja JPEG jest najlepsza do fotografii cyfrowych, format GIF zaś jest używany do *rysunków*, takich jak logo firmy, w których występują duże obszary o jednej barwie.

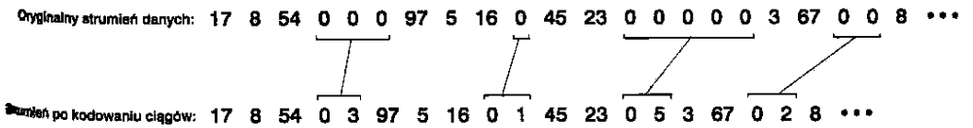
Drugi sposób klasyfikacji metod kompresji danych podano w części b) tablicy 27.1. Większość programów kompresji danych działa na zasadzie pobierania grup danych z pliku oryginalnego, poddawania ich kompresji w jakiś sposób i potem wpisywania tej skomprymowanej grupy do pliku wyjściowego. Jedną z metod podanych w tablicy jest np. CS&Q – skrót od *Coarser Sampling and/or Quantization* (zgrubne próbkowanie i/lub kwantyzacja). Przypuśćmy, że poddajemy kompresji jakiś przetworzony do postaci cyfrowej przebieg, taki jak sygnał audio przetworzony na cyfrowy z rozdzielczością 12 bitów. Możemy z sygnału oryginalnego odczytać dwie sąsiednie próbki (24 bity), odrzucić całkowicie jedną z próbek, odrzucić też 4 najmniej znaczące bity z drugiej próbki i następnie wpisać pozostałe 8 bitów do pliku wyjściowego. Mając 24 bity na wejściu a 8 bitów na wyjściu, wprowadziliśmy współczynnik kompresji 3:1 przy użyciu algorytmu stratnego. Choć taki sposób, sam w sobie, wydaje się dość prymitywny, to jest jednak bardzo skuteczny, gdy stosuje się go wraz z techniką zwaną *kompresją transformatową*. Jest ona podstawą JPEG, o czym będzie mowa w dalszej części tego rozdziału.

W tablicy 27.1b podano, że w metodzie CS&Q rozmiar zarówno grupy wejściowej, jak i wyjściowej jest ustalony. To znaczy, że ustaloną liczbę bitów odczytuje się z pliku wejściowego, a pewną mniejszą, też ustaloną liczbę bitów wpisuje się do pliku wyjściowego. W innych metodach kompresji są dopuszczalne zmienne liczby bitów odczytywanych lub wpisywanych. Zapoznając się z opisami kolejnych metod kompresji, sięgajcie do tej tablicy, aby zrozumieć, jak dana metoda plasuje się w podanej klasyfikacji. JPEG i MPEG nie są wymienione w części b) tablicy, gdyż są to algorytmy składane, łączące w sobie wiele innych metod. Są one zbyt skomplikowane, aby mogły być sklasyfikowane według tych prostych kategorii.

Kodowanie ciągów

Pliki danych często zawierają ten sam znak powtarzany wielokrotnie w jednym wierszu. Na przykład w plikach tekstowych używa się wielokrotnych spacji do oddzielania zdań, wcięć akapitów, formatowań tablic i wykresów itd. *Sygnały* wyrażane cyfrowo też często zawierają powtarzające się wartości, świadczące o tym, że sygnał się nie zmienia. Na przykład obraz nocnego nieba będzie zawierał długie grupy jednego lub kilku znaków, reprezentujące czarne tło. Podobnie sygnał muzyczny w postaci cyfrowej może mieć długie ciągi zer między piosenkami. Kodowanie ciągów jest prostym sposobem kompresji tego rodzaju plików.

Na rys. 27.1 przedstawiono kodowanie ciągów w przypadku sekwencji danych zawierającej wiele grup *zer*. Za każdym razem, gdy w danych wejściowych pojawia się zero, do pliku wyjściowego są wpisywane *dwie* wartości. Pierwsza z tych wartości jest zerem – flagą oznaczającą, że rozpoczyna się kodowanie ciągów. Drugą wartością jest liczba zer w grupie. Jeśli średnia długość grupy jest większa niż 2, to następuje kompresja. W innym razie, wiele pojedynczych zer w danych mogłoby spowodować, że plik zakodowany stanie się większy od oryginalnego.



Rys. 27.1. Przykład kodowania ciągów. Każda grupa zer jest zastępowana przez dwa znaki w pliku skomprymowanym: zero wskazuje, że zachodzi kompresja, po nim następuje liczba zer w grupie

Tablica 27.2. Kod ASCII. Jest to od dawna stosowana norma określająca sposób zapisu liter i liczb w postaci cyfrowej. Każdemu znakowi, który może być drukowany, przyporządkowuje się liczbę od 32 do 127, liczby zaś od 0 do 31 są stosowane do różnych działań kontrolnych. Chociaż zdefiniowano tylko 128 słów kodowych, to znaki ASCII są zwykle wyrażane jako pełny bajt (8 bitów). Wartości niezdefiniowane (od 128 do 255) są często wykorzystywane do liter greckich, symboli matematycznych i różnych wzorców geometrycznych; nie są to jednak zastosowania znormalizowane. Wiele spośród znaków kontrolnych (od 0 do 31) dotyczy starych sieci telekomunikacyjnych i nie znajduje zastosowania w technice komputerowej

0	null	32	space	64	@	96	`
1	start heading	33	!	65	A	97	a
2	start of text	34	"	66	B	98	b
3	end of text	35	#	67	C	99	c
4	end of xmit	36	$	68	D	100	d
5	enquiry	37	%	69	E	101	e
6	acknowledge	38	&	70	F	102	f
7	bell, beep	39	'	71	G	103	g
8	backspace	40	(	72	H	104	h
9	horz. tab	41	)	73	I	105	i
10	line feed	42	*	74	J	106	j
11	vert. tab, home	43	+	75	K	107	k
12	form feed, cls	44	,	76	L	108	l
13	carriage return	45	-	77	M	109	m
14	shift out	46	.	78	N	110	n
15	shift in	47	/	79	O	111	o
16	data line esc	48	0	80	P	112	p
17	device control 1	49	1	81	Q	113	q
18	device control 2	50	2	82	R	114	r
19	device control 3	51	3	83	S	115	s
20	device control 4	52	4	84	T	116	t
21	negative ack.	53	5	85	U	117	u
22	synch. idle	54	6	86	V	118	v
23	end xmit block	55	7	87	W	119	w
24	cancel	56	8	88	X	120	x
25	end of medium	57	9	89	Y	121	y
26	substitute	58	:	90	Z	122	z
27	escape	59	;	91	[	123	{
28	file separator	60	<	92	\	124	|
29	group separator	61	=	93	]	125	}
30	record separator	62	>	94	^	126	~
31	unit separator	63	?	95	_	127	del

Opracowano wiele różnych sposobów kodowania ciągów. Dane wejściowe można, na przykład, traktować jako indywidualne bajty albo jako grupy bajtów reprezentujące coś bardziej rozbudowanego, takiego jak liczby zmiennoprzecinkowe. Kodowanie ciągów może być zastosowane do *jednego* tylko znaku (jak w powyższym przykładzie z *zerem*), do *kilku* znaków lub do *wszystkich* znaków.

Dobrym przykładem uogólnionej metody ciągów jest PacBits, stworzona dla użytkowników komputerów Macintosh. Każdy bajt (osiem bitów) z pliku wejściowego jest zastępowany przez dziewięć bitów w pliku skomprymowanym. Dodatkowy, dziewiąty bit jest interpretowany jako *znak* liczby (plus lub minus). A zatem każdy znak odczytywany z pliku wejściowego znajduje się między 0 a 255, a każdy znak wpisany do pliku zakodowanego jest między –255 i 255. Aby zrozumieć działanie tej metody, rozważmy plik wejściowy: 1, 2, 3, 4, 2, 2, 2, 2, 4 oraz skomprymowany plik wygenerowany przez algorytm PackBits: 1, 2, 3, 4, 2, –3, 4. Program kompresji po prostu przenosi z pliku wejściowego do pliku skomprymowanego każdą liczbę, oprócz grupy 2, 2, 2, 2. Ta grupa jest reprezentowana w pliku skomprymowanym przez dwie liczby: 2, –3. Pierwsza liczba (2) wskazuje, z jakich znaków składa się grupa, druga (–3) zaś – liczbę znaków w grupie, wyznaczaną przez wzięcie wartości bezwzględnej i dodanie do niej jedynki. Na przykład 4, –2 oznacza 4, 4, 4; a 21, –4 oznacza 21, 21, 21, 21, 21 itd.

Niedogodnością techniki PackBits jest konieczność reformatowania dziewięciu bitów na standardowe bajty ośmiobitowe stosowane w komputerach do przesyłania i przechowywania danych. Można zrobić pożyteczną modyfikację tej techniki, ograniczając wejście do tekstu ASCII. Jak podano w tablicy 27.2, każdy znak ASCII jest zwykle przechowywany jako pełny bajt (osiem bitów), lecz naprawdę do zidentyfikowania znaku wykorzystuje się tylko *siedem* bitów. Innymi słowy, wartości od 127 do 255 nie są definiowane żadnym znormalizowanym określeniem i nie muszą być przesyłane lub przechowywane. To umożliwia wykorzystanie ósmego bitu do wskazywania, czy kodowanie ciągów jest w toku.

Kodowanie Huffmana

Nazwa metody pochodzi od D.A. Huffmana, który opracował ją w latach 50. XX wieku. Na rys. 27.2 przedstawiono histogram wartości bajtów w dużym pliku ASCII. Ponad 96% tego pliku zawiera tylko 31 rodzajów znaków: małe litery, spacje, przecinki, kropki i powroty karetki. To spostrzeżenie można wykorzystać do przygotowania schematu kompresji właściwego dla tego pliku. Najpierw przypiszemy każdemu z tych 31 popularnych znaków 5-bitowe binarne słowo kodowe: 00000 = „a", 00001 = „b", 00010 = „c" itd. To umożliwia zredukowanie 96% pliku pod względem rozmiaru o 5/8. Ostatnie ze słów kodowych 5-bitowych 11111 będzie flagą wskazującą, że przesyłany znak nie jest jednym z 31 popularnych znaków. Wtedy następne osiem bitów wskazuje, jaki to jest znak, według standardowego przyporządkowania ASCII. To w rezultacie daje tylko 4% znaków w pliku wejściowym wymagających 5 + 8 = 13 bitów. Zasada tego kodowania polega na przyznawaniu mniejszej liczby bitów znakom często stosowanym, a większej – znakom stosowanym rzadko. W tym przykładzie *średnia* liczba bitów wymaganych na jeden znak w pliku oryginalnym jest: 0,96 × 5 + 0,04 × 13 = 5,32. Inaczej mówiąc, całkowity współczynnik kompresji jest: 8 bitów/5,32 bitów, czyli ok. 1,5:1.

W kodowaniu Huffmana doprowadzono tę zasadę do skrajności. Znakom występującym najczęściej, takim jak spacja i kropka, można przyporządkować nawet tylko jeden lub dwa bity. Znaki rzadko spotykane, takie jak !, @, #, $ i % mogą wymagać kilkunastu lub więcej bitów. Optymalną, pod względem matematycznym, sytuację osiąga się, gdy liczba bitów zastosowanych do każdego znaku jest proporcjonalna do logarytmu z prawdopodobieństwa występowania danego znaku.

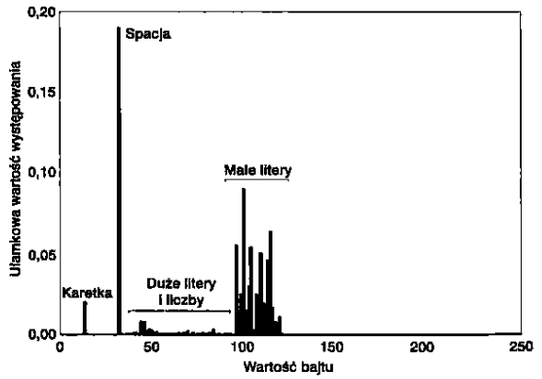
Pomysłowym rozwiązaniem w kodowaniu Huffmana jest sposób pakowania razem ze sobą słów kodowych o różnej długości. Wyobraźmy sobie odbieranie szeregowego strumienia danych złożonego z zer i jedynek. Jeśli każdy znak jest reprezentowany przez osiem bitów, to można łatwo oddzielić każdy znak od następnego przez robienie przerw między 8-bitowymi porcjami. Rozważmy teraz strumień danych zakodowany metodą Huffmana, gdzie każdy znak może być reprezentowany przez zmienną liczbę bitów. Jak w tym przypadku oddzielić dany znak od następnego? Odpowiedź leży w prawidłowej selekcji słów kodowych Huffmana, umożliwiającej prawidłowe rozdzielanie. Działanie tej metody zilustrujemy przykładem.

Na rys. 27.3 przedstawiono uproszczony schemat kodowania Huffmana. Znaki od *A* do *G* występują w oryginalnym strumieniu danych z prawdopodobieństwami podanymi w tablicy. Ponieważ znak *A* występuje najczęściej, więc będzie reprezentowany pojedynczym bitem jako 1. Następny, pod względem częstości występowania, znak *B* dostaje dwa bity, jest on prezentowany przez 01. I tak kontynuujemy przydzielanie bitów aż do znaku występującego najrzadziej, którym jest *G*. Temu znakowi przydzielamy sześć bitów, 000011. Jak pokazano na rys. 27.3, słowa kodowe o zmiennej długości są na nowo sortowane w grupy ośmiobitowe, będące standardem w komputerach.

Podczas dekompresji (czyli rozpakowywania skomprymowanych plików) wszystkie ośmiobitowe grupy są umieszczane jedna za drugą, tworząc długi ciąg zer i jedynek. Przyjrzyjcie się dokładnie tablicy kodowej na rys. 27.3 i zwróćcie uwagę, że każde zakodowane słowo składa się z dwóch części: pewnej liczby zer przed *jedynką* i opcjonalnego binarnego słowa kodowego po *jedynce*. To umożliwia podział tego strumienia danych dwójkowych na słowa kodowe bez konieczności stosowania ograniczników lub innych znaczników między poszczególnymi słowami. Program dekompresji przegląda strumień zer i jedynek aż do pojawienia się prawidłowego słowa kodowego i potem rozpoczyna poszukiwanie następnego znaku. Sposób tworzenia słów kodowych daje pewność, że podczas rozdzielania nie powstaną niejednoznaczności.

Bardziej zaawansowana wersja kodowania, z wykorzystaniem metody Huffmana, nosi nazwę **kodowania arytmetycznego**. W tym sposobie kodowania *sekwencje* znaków są reprezentowane przez indywidualne słowa kodowe, według prawdopodobieństwa występowania. Zaletą tego sposobu jest kompresja danych lepsza o, powiedzmy, 5–10%. Często stosuje się też kodowanie ciągów, po którym następuje albo kodowanie Huffmana, albo arytmetyczne. Jak można się było spodziewać, algorytmy tego typu są bardzo skomplikowane i ich opracowanie powierza się specjalistom z dziedziny kompresji danych.

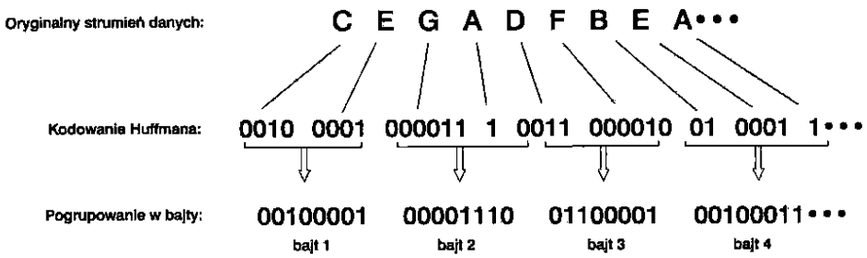
Aby można było zrealizować kodowanie Huffmana lub arytmetyczne, algorytmy kompresji i dekompresji muszą być zgodne pod względem binarnych słów kodowych zastosowanych do wyrażania każdego znaku (lub grupy znaków). Można to uzyskać dwoma sposobami. Najprostszym jest zastosowanie wcześniej zdefiniowanej tablicy kodowania, która zawsze pozostaje taka sama, niezależnie od informacji poddawanej kompresji. W metodach bardziej złożonych stosuje się kodowanie zoptymalizowane dla tych danych, które są aktualnie używane. To wymaga włączenia tablicy kodowania do pliku podlegającego kompresji, aby móc skorzystać z tej tablicy w programie dekompresji. Oba sposoby są często stosowane.



Rys. 27.2. Histogram tekstu. Jest to histogram wartości ASCII tekstu pochodzącego z rozdziału w tej książce. Najczęściej spotykanymi znakami są małe litery, spacje i powroty karetki

Przykład tablicy kodowej

Litera	Prawdopodobieństwo	Kod Huffmana
A	0,154	1
B	0,110	01
C	0,072	0010
D	0,063	0011
E	0,059	0001
F	0,015	000010
G	0,011	000011

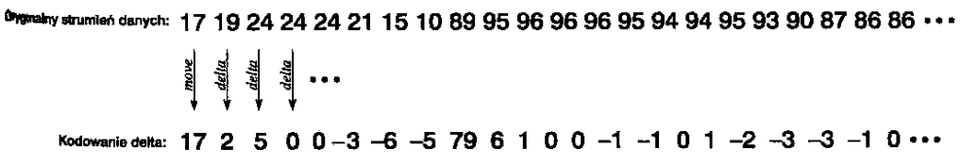


Rys. 27.3. Kodowanie Huffmana. Tablica kodowa przypisuje każdej z siedmiu liter stosowanych w tym przykładzie zmienną długość binarnego słowa kodowego, w zależności od częstotliwości występowania tych liter. Oryginalny strumień danych złożony z tych 7 znaków jest w tablicy kodowej przełożony na dane w kodzie Huffmana. Ponieważ długości słów kodowych Huffmana są różne, więc dane binarne wymagają przegrupowania na standardowe bajty 8-bitowe, dogodne do przesyłania i przechowywania

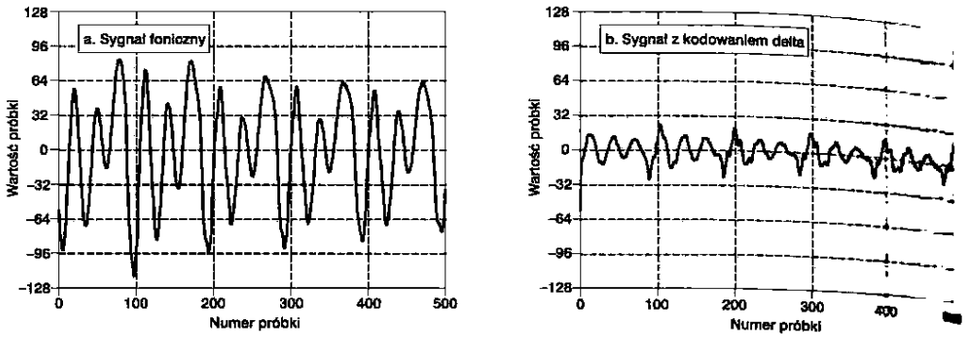
Kodowanie delta

W nauce, technice i matematyce grecka litera *delta* (Δ) jest stosowana do oznaczania *zmiany* jakiejś wielkości. Pojęcie *kodowanie delta* odnosi się do kilku metod, w których dane są gromadzone jako *różnice* między kolejnymi próbkami (lub znakami), a nie same próbki. Przykład przedstawiono na rys. 27.4. Pierwsza wartość w pliku kodowanym metodą delta jest taka sama jak pierwsza wartość w danych oryginalnych. Wszystkie następne wartości w kodowanym pliku są równe różnicom (delta) między odpowiednią wartością w pliku wejściowym i wartością *poprzednią* w tym pliku.

Kodowanie delta można stosować do kompresji danych, gdy zmiany wartości w pliku wejściowym są *łagodne*, a więc różnice między sąsiednimi wartościami są niewielkie. Takiej właściwości nie mają teksty ASCII ani kody wykonywalne, natomiast jest to cecha bardzo często spotykana



Rys. 27.4. Przykład kodowania delta. Pierwsza wartość w zakodowanym pliku jest taka sama, jak pierwsza wartość w pliku oryginalnym. Następnie każda próbka w zakodowanym pliku jest różnicą między aktualną i poprzednią próbką w pliku oryginalnym



Rys. 27.5. Przykład kodowania delta. Na rys. a) pokazano sygnał foniczny przetworzony na cyfrowy z rozdzielczością 8 bitów. Na rys. b) pokazano wersję tego sygnału zakodowanego metodą delta. Kodowanie delta jest przydatne do kompresji danych, jeśli sygnał kodowany zmienia się między próbkami bardzo wolno

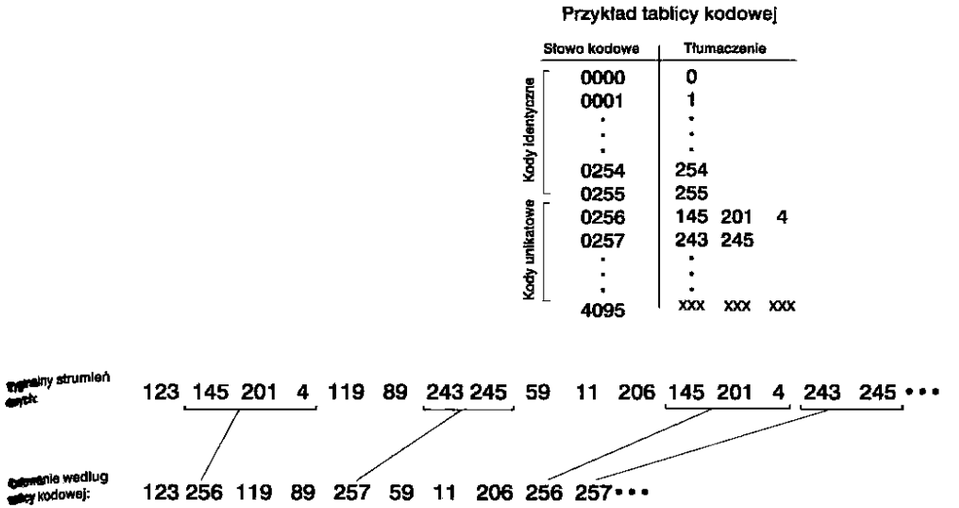
w tych przypadkach, gdy plik reprezentuje jakiś *sygnał*. Na przykład na rys. 27.5 a) pokazano segment sygnału fonicznego, przetworzonego na sygnał cyfrowy z rozdzielczością 8-bitową, wtedy wartości próbek mieszczą się w granicach od –127 do 127. Na rys. 27.5 b) pokazano ten sygnał zakodowany metodą delta. Podstawową właściwością sygnału po kodowaniu delta jest mniejsza *amplituda* niż w sygnale oryginalnym. Inaczej mówiąc, kodowanie delta zwiększyło prawdopodobieństwo, że wartość każdej próbki będzie bliska zeru, a zmniejszyło, że będzie daleka od zera. To niejednakowe prawdopodobieństwo jest właśnie tą cechą, która jest niezbędną do działania kodowania Huffmana. Jeśli sygnał oryginalny się nie zmienia lub wykres jego zmian jest linią prostą, to kodowanie delta da wynik w postaci ciągu próbek mających jednakową wartość.

Tego wymaga kodowanie ciągów. Dlatego często stosowanym sposobem kompresji sygnałów jest kodowanie delta z następującym po nim kodowaniem Huffmana i/lub kodowaniem ciągów.

Metodę stosowaną w kodowaniu delta można rozszerzyć, uzyskując technikę bardziej skomplikowaną zwaną **liniowym kodowaniem predykcyjnym** (**LPC** – *Linear Predictive Coding*). Aby zrozumieć LPC, wyobraźmy sobie, że pierwsze 99 próbek z sygnału wejściowego zostało już zakodowanych i zabieramy się do próbki numer 100. Zadajemy sobie wtedy pytanie: jaka jest najbardziej prawdopodobna wartość próbki 100 wyznaczona na podstawie wartości pierwszych 99 próbek? W przypadku kodowania delta najbardziej prawdopodobna wartość próbki 100 jest taka sama jak wartość poprzedniej próbki 99. Ta wartość oczekiwana jest stosowana jako odniesienie przy kodowaniu próbki 100. To znaczy, że w kodowanym pliku umieszcza się *różnicę* między próbką i wartością spodziewaną. W kodowaniu LPC ten sposób ulega rozszerzeniu przez lepsze odgadywanie najbardziej prawdopodobnej wartości próbki. Realizuje się to, uwzględniając nie tylko jedną, lecz kilka ostatnich próbek. Algorytmy stosowane w LPC są podobne do filtrów rekursywnych z wykorzystaniem przekształcenia Z i innych metod z bogatym aparatem matematycznym.

Kompresja LZW

Nazwa kompresji LZW pochodzi od nazwisk jej twórców, którymi są A. Lempel i J. Ziv, z późniejszymi modyfikacjami, które wprowadził Terry A. Welch. Dzięki swojej prostocie i uniwersalności LZW jest podstawową metodą kompresji danych, o ogólnym przeznaczeniu. Można oczekiwać, że metoda LZW, zastosowana typowo do tekstów, kodów wykonywalnych i plików danych podobnego typu, spowoduje ich kompresję do ok. połowy rozmiaru oryginalnego. LZW jest też skuteczna w przypadku plików danych o bardzo dużej redundancji, takich jak stabelaryzowane liczby, źródłowe kody komputerowe i zbierane sygnały. W tych przypadkach współczynnik kompresji jest



Rys. 27.6. Przykład tablicy kodowej stosowanej do kompresji danych. Jest ona podstawą popularnej metody kompresji LZW. Kodowanie odbywa się przez identyfikację sekwencji bajtów z pliku oryginalnego, które występują w tablicy kodowej. 12-bitowe słowo kodowe reprezentujące sekwencję jest umieszczane w pliku skomprymowanym zamiast tej sekwencji. Pierwsze 256 wpisów w tablicy odpowiada wartościom pojedynczych bajtów, od 0 do 255, a pozostałe wpisy odpowiadają sekwencjom bajtów. Algorytm LZW jest wydajnym sposobem generowania tablicy kodowej na podstawie danych poddawanych kompresji. (Przedstawiona tu tablica kodowa jest przykładem uproszczonym, a nie generowanym w rzeczywistości przez algorytm LZW)

zwykle ok. 5:1. Kompresja LZW w komputerach osobistych jest podstawą niektórych programów użytkowych, które – jak twierdzą ich twórcy – *podwoją pojemność waszego twardego dysku*.

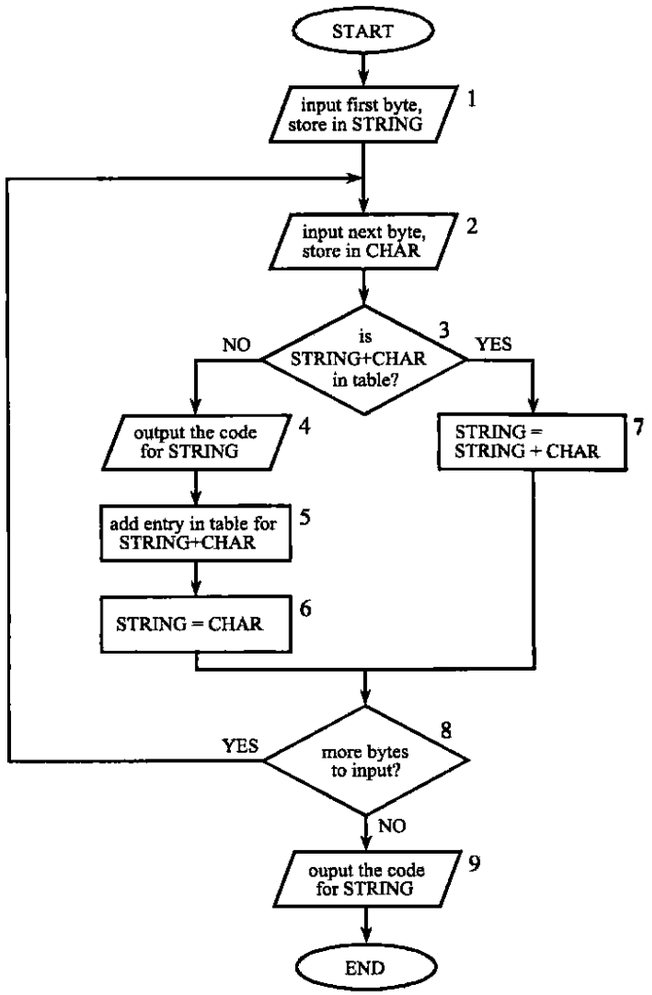
Kompresja LZW jest zawsze stosowana w plikach obrazowych GIF i oferowana jako opcja w TIFF i PostScript. Kompresja LZW jest chroniona patentem USA nr 4,558,302, przyznanym 10 grudnia 1985 r. firmie Sperry Corporation (obecnie Unisys Corporation). O informacje w sprawie licencji można się zwracać do: Welch Licensing Department, Law Department, M/SC2SW1, Unisys Corporation, Blue Bell, Pennsylvania, 19424-0001.

W kompresji LZW korzysta się z **tablicy kodowej**, jak to pokazano na rys. 27.6. Tablica zawiera na ogół 4096 wpisów. W tym przypadku dane zakodowane w LZW zawierają wyłącznie 12-bitowe słowa kodowe, z których każde odnosi się do jednego wpisu w tablicy kodowej. Dekompresję realizuje się, pobierając każde zakodowane słowo z pliku skomprymowanego i tłumacząc je za pomocą tablicy kodowej, aby stwierdzić, jaki znak lub zestaw znaków reprezentuje. Słowa kodowe od 0 do 255 w tablicy są zawsze przeznaczone do reprezentowania pojedynczych bajtów z pliku wejściowego. Gdyby na przykład zastosowano tylko te pierwsze 256 słów kodowych, to każdy bajt w pliku oryginalnym byłby przetworzony na 12 bitów w pliku zakodowanym LZW, co dałoby plik większy o 50%. Podczas dekompresji wszystkie słowa 12-bitowe byłyby tłumaczone tablicą kodową z powrotem na pojedyncze bajty. Oczywiście, takie rozwiązanie nie byłoby użyteczne.

W metodzie LZW kompresję uzyskuje się, stosując słowa kodowe od 256 do 4095 do reprezentowania *sekwencji* bitów. Na przykład 523 może reprezentować sekwencję trzech bajtów: 231 124 234. Za każdym razem, gdy algorytm kompresji napotka tę sekwencję w pliku wejściowym, słowo kodowe 523 jest umieszczane w zakodowanym pliku. Podczas dekompresji słowo kodowe 523 jest tłumaczone przez tablicę kodową w celu odtworzenia prawdziwej 3-bajtowej sekwencji. Im dłuższa jest sekwencja przypisana do danego słowa kodowego i im częściej ta sekwencja się powtarza, tym silniejszą kompresję się uzyskuje.

Chociaż ten sposób jest prosty, to są jednak do pokonania dwie przeszkody: 1) jak określić, które sekwencje powinny być w tablicy kodowej oraz 2) jak wyposażyć program dekompresji w taką samą tablicę kodową, jak stosowana w programie kompresji. W algorytmie LZW świetnie rozwiązano oba te problemy.

Gdy program LZW zaczyna kodować plik, to tablica kodowa zawiera tylko pierwszych 256 wpisów; reszta tablicy pozostaje pusta. To znaczy, że pierwsze słowa kodowe wpisywane do pliku zakodowanego są po prostu pojedynczymi bajtami z pliku wejściowego przetworzonymi do 12 bitów. W miarę dalszego kodowania algorytm LZW identyfikuje sekwencje powtarzające się w danych i dodaje je do tablicy kodowej. Kompresja się zaczyna, gdy algorytm po raz drugi napotka taką sekwencję. Podstawową sprawą jest to, że sekwencja z pliku wejścio-



Rys. 27.7. Sieć działań kompresji LZW. Zmienna CHAR jest pojedynczym bajtem. Zmienna STRING jest sekwencją bajtów o zmiennej długości. Dane są odczytywane z pliku wejściowego (ramki 1 i 2) jako pojedyncze bajty i wpisywane do pliku skomprymowanego (ramka 4) jako kody 12-bitowe. W tablicy 27.3 podano przykład tego algorytmu

wego nie jest dodawana do tablicy kodowej, jeśli nie została przedtem umieszczona w pliku skomprymowanym jako pojedyncze znaki (słowa kodowe od 0 do 255). Jest to ważne, ponieważ umożliwia programowi dekompresji *odtwarzanie* tablicy kodowej bezpośrednio z danych skomprymowanych, bez konieczności przesyłania oddzielnie tablicy kodowej.

Na rys. 27.7 przedstawiono sieć działań kompresji LZW. W tablicy 27.3 zaś podano krok po kroku szczegóły przykładu dotyczącego pliku wejściowego złożonego z 45 bajtów, będącego strumieniem tekstowym ASCII: *the/rain/in/Spain/falls/mainly/in/the/plain*[28]. Gdy mówimy, że algorytm LZW odczytuje znak „a" z pliku wejściowego, to mamy na myśli odczyt wartości 01100001 (liczba 97 wyrażona jako słowo 8-bitowe), gdzie 97 jest znakiem „a" w ASCII. Gdy mówimy, że algorytm wpisuje znak „a" do pliku zakodowanego, to myślimy o wpisaniu 000001100001 (liczba 97 wyrażona jako słowo 12-bitowe).

W algorytmie kompresji zastosowano dwie zmienne: *CHAR* i *STRING*. W zmiennej *CHAR* jest przechowywany pojedynczy znak, tj. wartość pojedynczego bajtu między 0 a 255. Zmienna *STRING* jest zmienną długości strumienia danych, czyli grupy jednego lub więcej znaków, przy czym każdy znak jest pojedynczym bajtem. W ramce 1 na rys. 27.7 wykonywanie programu rozpoczyna się od pobrania pierwszego bajtu z pliku wejściowego i umieszczenie go w zmiennej *STRING*. W tablicy 27.3 to działanie pokazano w wierszu 1. Potem następuje pętla algorytmu dla każdego dodatkowego bajtu w pliku wejściowym, sterowana w sieci działań przez ramkę 8. Za każdym razem, gdy z pliku wejściowego jest odczytywany jakiś bajt (ramka 2), jest on zapisywany w zmiennej *CHAR*. Tablica danych jest następnie przeszukiwana w celu określenia, czy takiemu połączeniu ze sobą dwóch zmiennych *STRING* + *CHAR* przyporządkowano już jakieś słowo kodowe (ramka 3).

Jeśli w tablicy kodowej *nie* znaleziono dopasowania, to podejmuje się trzy działania, jak to pokazano w ramkach 4, 5 i 6. W ramce 4 słowo kodowe 12-bitowe odpowiadające zmiennej *STRING* jest wpisywane do pliku skomprymowanego. W ramce 5 jest w tablicy tworzone nowe słowo kodowe dla połączenia *STRING* + *CHAR*. W ramce 6 zmienna *STRING* przyjmuje wartość zmiennej *CHAR*. Przykład tych działań podano w wierszach od 2 do 10 w tablicy 27.3, dla pierwszych 10 bajtów przykładowego pliku.

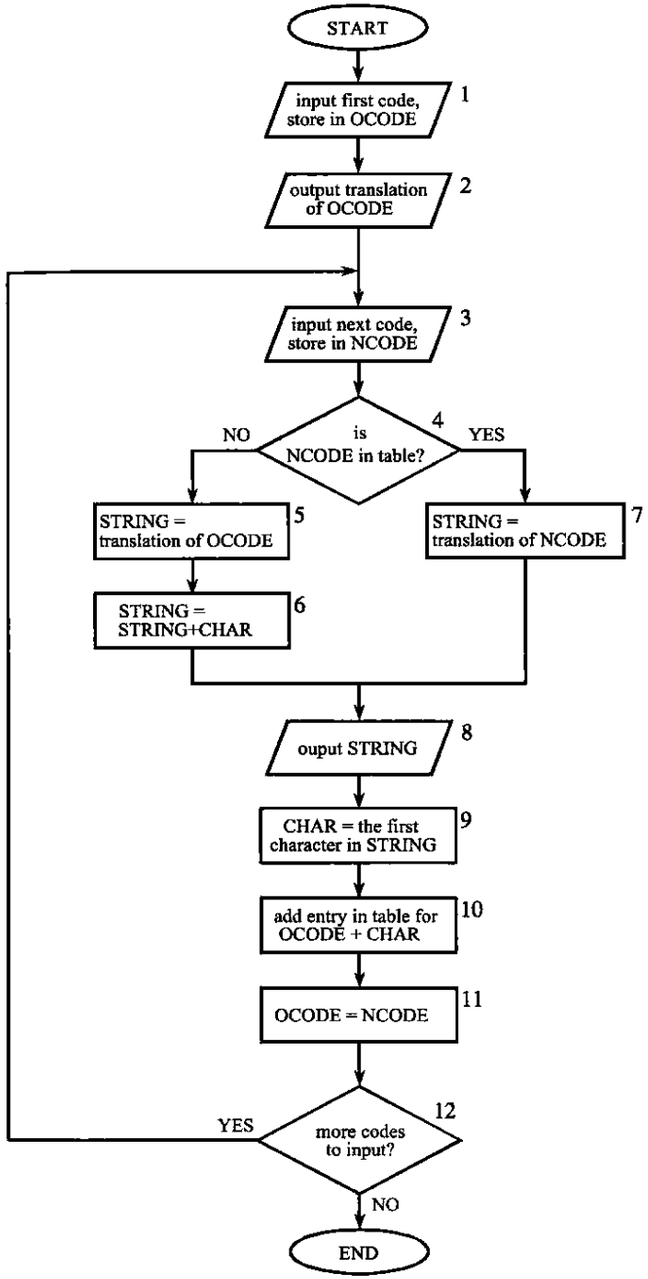
Gdy zaś dopasowanie w tablicy kodowej *zostało* znalezione (ramka 3), to połączenie *STRING* + *CHAR* jest wpisywane do zmiennej *STRING* bez podejmowania żadnych dalszych działań (ramka 7). Tak więc, jeśli pasująca sekwencja została znaleziona w tablicy, nie podejmuje się żadnych działań przed stwierdzeniem, czy jest w tablicy jakaś *dłuższa* pasująca sekwencja. Przykład takiego działania pokazano w wierszu 11, gdzie sekwencja *STRING* + *CHAR* = *in* jest zidentyfikowana jako już mająca słowo kodowe w tablicy. W wierszu 12 następny znak z pliku wejściowego, /, jest dodawany do sekwencji i w tablicy kodowej jest szukana sekwencja *in/*. Ponieważ tej dłuższej sekwencji nie ma w tablicy, więc program *dodaje* ją do tablicy, wyprowadzając słowo kodowe dla krótszej sekwencji, które *jest już* w tablicy (słowo 262) i zaczyna poszukiwania sekwencji rozpoczynających się od znaku /. Taki ciąg działań jest kontynuowany, aż nie będzie już więcej znaków w kodzie wejściowym. Program kończy się słowem kodowym odpowiadającym bieżącej wartości *STRING* wpisywanej do skomprymowanego pliku (jak to zilustrowano w ramce 9 na rys. 27.7 i wierszu 45 tablicy 27.3).

Sieć działań algorytmu dekompresji LZW przedstawiono na rys. 27.8. Każde słowo kodowe jest odczytywane ze skomprymowanego pliku i porównywane z tablicą kodową w celu uzyskania tłumaczenia. Wszystkie słowa kodowe są poddane takiej procedurze; tablica kodowa jest uaktualniana, dzięki czemu ciągle jest zgodna z tą stosowaną podczas kompresji. Jest jednak mała komplikacja związana z dekompresją. Istnieją pewne kombinacje danych powodujące, że algorytm dekompresji odbiera słowo kodowe, którego jeszcze nie ma w tablicy. Takie przypadki są obsługiwane w ramkach 4, 5 i 6.

[28] Znany fragment z musicalu *My Fair Lady* (przyp. tłum.).

Tab. 27.3. Przykład kompresji LZW. Przedstawiono tu kompresję zdania: the/rain/in/Spain/falls/mainly/in/the/plain/

	char	string + char	Czy jest w tablicy?	Wyjście	Dodaj do tablicy	Klucz STRING	Uwagi
1	t	t					Pierwszy znak – bez działania
2	h	th	nie	t	256 = th	h	
3	e	he	nie	h	257 = he	e	
4	/	e/	nie	e	258 = e/	/	
5	r	/r	nie	/	259 = /r	r	
6	a	ra	nie	r	260 = ra	a	
7	i	ai	nie	a	261 = ai	i	
8	n	in	nie	i	262 = in	n	
9	/	n/	nie	n	263 = n/	/	
10	i	/i	nie	/	264 = /i	i	
11	n	in	tak (262)			in	Pierwsze zmodyfikowane dopasowanie
12	/	in/	no	262	265 = in/	/	
13	S	/S	nie	/	266 = /S	S	
14	p	Sp	nie	S	267 = Sp	p	
15	a	pa	nie	p	268 = pa	a	
16	i	ai	tak (261)			ai	Dopasowanie *ai*, *ain* nie ma jeszcze w tablicy
17	n	ain	nie	261	269 = ain	n	*ain* dodane do tablicy
18	/	n/	tak (263)			n/	
19	f	n/f	nie	263	270 = n/f	f	
20	a	fa	nie	f	271 = fa	a	
21	l	al	nie	a	272 = al	l	
22	l	ll	nie	l	273 = ll	l	
23	s	ls	nie	l	274 = ls	s	
24	/	s/	nie	s	275 = s/	/	
25	m	/m	nie	/	276 = /m	m	
26	a	ma	nie	m	277 = ma	a	
27	i	ai	tak (261)			ai	Dopasowanie *ai*
28	n	ain	tak (269)			ain	Dopasowanie dłuższej sekwencji *ain*
29	l	ainl	nie	269	278 = ainl	l	
30	y	ly	nie	l	279 = ly	y	
31	/	y/	nie	y	280 = y/	/	
32	o	/o	nie	/	281 = /o	o	
33	n	on	nie	o	282 = on	n	
34	/	n/	tak (263)			n/	
35	t	n/t	nie	263	283 = n/t	t	
36	h	th	tak (256)			th	Dopasowanie *th*, *the* nie ma jeszcze w tablicy
37	e	the	nie	256	284 = the	e	*the* dodano do tablicy
38	/	e/	yes			e/	
39	p	e/p	nie	258	285 = e/p	p	
40	l	pl	nie	p	286 = pl	l	
41	a	la	nie	l	287 = la	a	
42	i	ai	tak (261)			ai	Dopasowanie *ai*
43	n	ain	tak (269)			ain	Dopasowanie dłuższej sekwencji *ain*
44	/	ain/	nie	269	288 = ain/	/	
45	EOF	/		/			Koniec pliku, wyjście STRING



Rys. 27.8. Sieć działań dekompresji LZW. W zmiennych OCODE i NCODE (stare i nowe słowo kodowe – oldcode i newcode) są przechowywane 12-bitowe słowa kodowe z pliku skomprymowanego, w CHAR jest przechowywany pojedynczy bajt, a w STRING ciąg bajtów

Większość podstawowych programów LZW ma długość kilkudziesięciu wierszy. Prawdziwa trudność leży w efektywnym wykorzystywaniu tablicy kodowej. Zbyt uproszczone podejście do tej kwestii powoduje, że wymagana jest bardzo duża pamięć, a wykonanie programu jest powolne. W komercyjnych programach LZW wprowadzono kilka trików, aby ulepszyć ich działanie. Na przykład problem pojemności pamięci powstaje, gdyż nie wiadomo z góry, jak długi będzie ciąg znaków dla każdego ze słów kodowych. W większości programów LZW tę kwestię rozwiązuje się, korzystając z redundancyjnego charakteru tablicy kodowej. Spójrzcie na przykład, na wiersz 29 w tablicy 27.3, gdzie słowo kodowe 278 jest definiowane jako *ainl*. Zamiast zapisywania tych czterech bajtów, słowo 278 można zapisać jako: *słowo kodowe 269 + l*, gdzie słowo kodowe 269 było poprzednio zdefiniowane jako *ain* w wierszu 17. Podobnie słowo kodowe 269 może być zapisywane jako: *słowo kodowe 261 + n*, gdzie słowo kodowe 261 było poprzednio zdefiniowane jako *ai* w wierszu 7. Taki schemat działania jest zawsze możliwy – każde słowo kodowe może być wyrażone jako słowo poprzednie plus jeden nowy znak.

Czas wykonania algorytmu kompresji jest ograniczony przeszukiwaniem tablicy kodowej w celu określenia, czy jest w niej poszukiwany odpowiednik. Przez analogię wyobraźcie sobie, że chcecie stwierdzić, czy nazwisko waszego przyjaciela jest w książce telefonicznej. Kłopot w tym, że jedyna książka, jaką macie, nie jest ułożona alfabetycznie, lecz według numerów telefonów. To wymaga przeszukiwania strona po stronie w celu znalezienia nazwiska, które jest wam potrzebne. To mało efektywne działanie jest dokładnie takie samo, jak przeszukiwanie wszystkich 4096 słów kodowych w celu znalezienia odpowiednika dla danego ciągu znaków. Rozwiązaniem jest taka organizacja tablicy kodowej, żeby to, czego szukamy, samo powiedziało wam, gdzie szukać (jak w ułożonej częściowo alfabetycznie książce telefonicznej). Innymi słowy, nie przyporządkowujcie 4096 słów kodowych kolejnym miejscom pamięci. Podzielcie raczej pamięć na sekcje, według umieszczonych w nich sekwencji. Przypuśćmy, na przykład, że chcemy stwierdzić, czy sekwencja *słowo kodowe 329 + x* jest w tablicy kodowej. Tablica powinna być tak zorganizowana, żeby *x* wskazywało, od którego miejsca należy rozpocząć poszukiwanie. Jest wiele sposobów tego typu organizacji pamięci i stają się one dość skomplikowane.

To przywodzi nas do komentarza na temat LZW i podobnych metod kompresji: *to jest dziedzina, w której istnieje silna konkurencja*. O ile podstawy kompresji danych są stosunkowo proste, o tyle różnego typu programy, sprzedawane jako produkty komercyjne, są skrajnie wysublimowane. Firmy zarabiają, sprzedając programy wykonujące kompresję i zazdrośnie strzegą swych tajemnic przez zastrzeżenia patentowe. Nie spodziewajcie się, że w ciągu kilkugodzinnej pracy osiągniecie taki sam poziom działania, jak te programy firmowe.

JPEG (kompresja transformatowa)

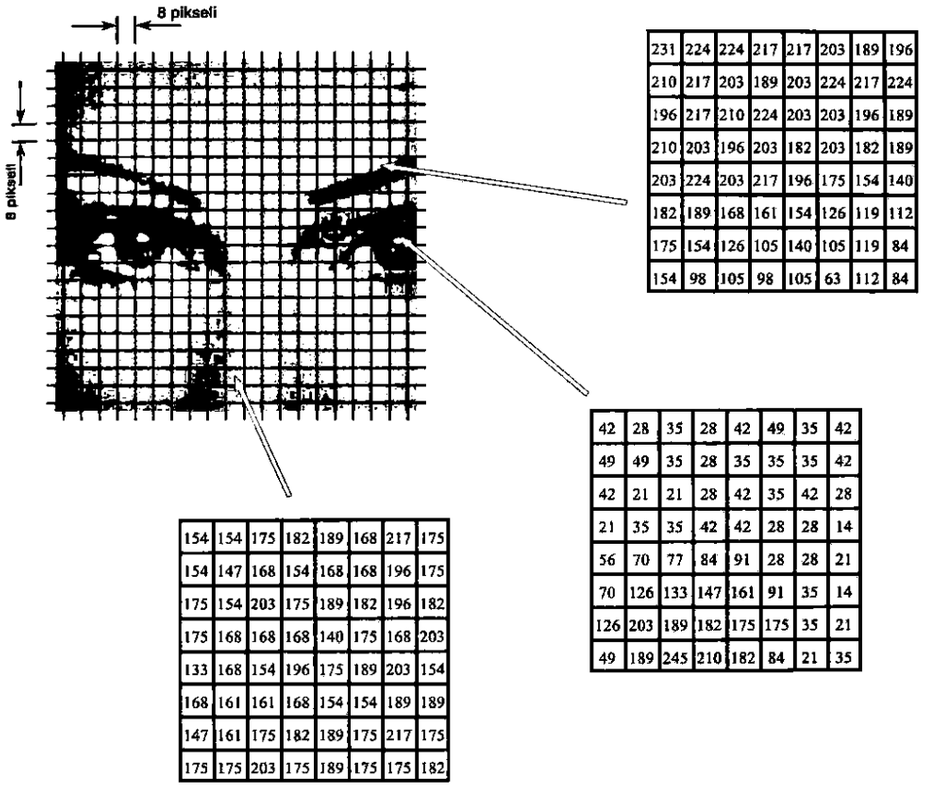
Opracowano wiele metod kompresji stratnej, jednak najbardziej wartościowa okazała się rodzina metod zwana *kompresją transformatową*. Najlepszym przykładem kompresji transformatowej jest popularny standard kodowania obrazów JPEG. Nazwa standardu pochodzi od grupy roboczej, która go opracowała: *Joint Photographic Expert Group*. Na przykładzie opisu działania JPEG objaśnimy zasady pracy kompresji stratnej.

Omówiliśmy już prostą metodę kompresji stratnej – *próbkowanie zgrubne i/lub kwantyzację* (CS&Q, tablica 27.1). Polega ona na zredukowaniu liczby bitów w próbce lub zupełnym pominięciu niektórych próbek. Obie te procedury dają żądany skutek – plik danych staje się mniejszy kosztem jakości sygnału. Jak można się tego spodziewać, te proste metody nie działają zbyt dobrze.

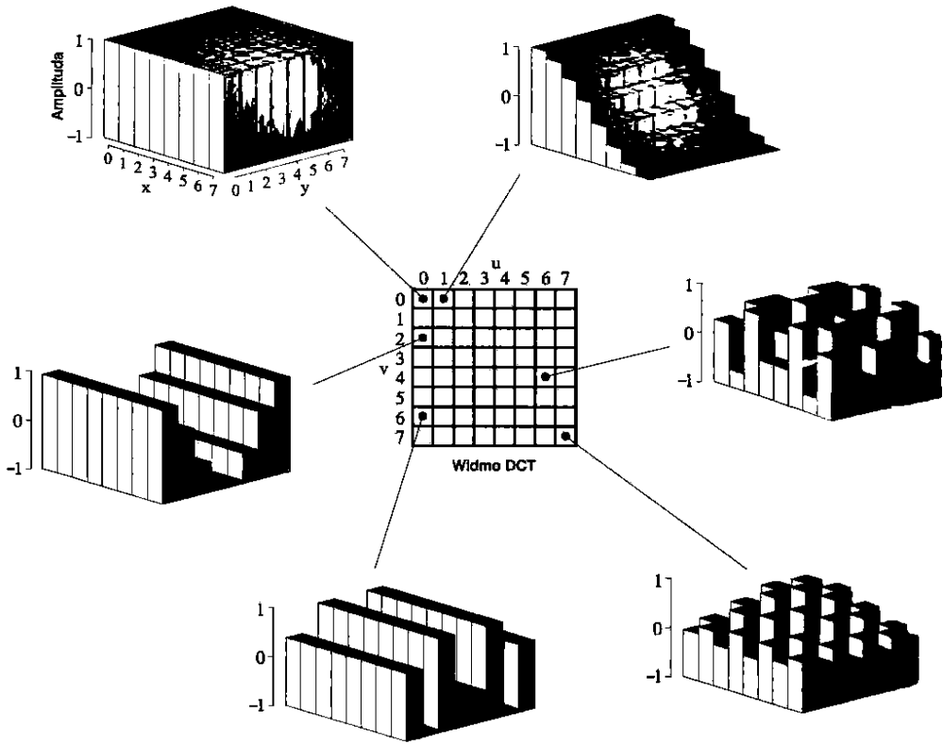
Kompresja transformatowa jest oparta na prostej przesłance: gdy sygnał oryginalny jest poddawany przekształceniu Fouriera (lub innemu), to uzyskiwane wartości danych nie są już równorzędne pod względem ich wartości jako nośnika informacji. A w szczególności

składowe małoczęstotliwościowe sygnału są ważniejsze niż składowe o dużych częstotliwościach. Usuwając 50% bitów ze składowych o dużych częstotliwościach, możemy usunąć – powiedzmy – tylko 5% zakodowanej informacji.

Kompresja JPEG (rys. 27.9) rozpoczyna się od rozdzielenia obrazu na grupy po 8 × 8 pikseli. W pełnym algorytmie JPEG może być akceptowany szeroki zakres liczby bitów na piksel, włącznie z informacją o kolorze. W tym przykładzie każdy piksel jest pojedynczym bajtem o wartości w skali szarości między 0 a 255. Te grupy 8 × 8 bitów są podczas kompresji traktowane niezależnie. Każda grupa jest więc na początku reprezentowana przez 64 bajty. Po wykonaniu przekształcenia i usunięciu danych dotyczących dużych częstotliwości każda grupa jest reprezentowana przez, przypuśćmy, od 2 do 20 bajtów. Podczas dekompresji wykonuje się przekształcenie odwrotne 2 do 20 bajtów, aby utworzyć aproksymację oryginalnej grupy 8 × 8. Te aproksymowane grupy są następnie łączone ze sobą, dając zdekomprymowany obraz. Dlaczego mamy stosować grupy 8 × 8 pikseli, a nie np. 16 × 16? Grupowanie 8 × 8 wyniknęło z maksymalnego rozmiaru grupy, jaką mogły przetwarzać układy scalone produkowane w czasie, gdy powstawał standard JPEG. W każdym razie kompresja JPEG z takim rozmiarem grupy działa dobrze i rozmiar może być albo nie być zmieniony w przyszłości.



Rys. 27.9. Podział obrazu do kompresji JPEG. Kompresja przez przekształcenie JPEG rozpoczyna się od rozdzielenia obrazu na grupy 8 × 8 zawierające po 64 piksele. Na rysunku powiększono trzy z tych grup 8 × 8, z podaniem wartości poszczególnych pikseli, mieszczących się w pojedynczym bajcie – od 0 do 255



Rys. 27.10. Funkcje podstawowe przekształcenia DCT. Widmo DCT jest matrycą 8 × 8, której każdy element ma wartość równą amplitudzie jednej z 64 funkcji bazowych. Na rysunku pokazano sześć z tych funkcji bazowych, ze wskazaniem miejsca, gdzie dana amplituda jest zlokalizowana

Przebadano różne przekształcenia pod względem ich przydatności do kompresji danych, a niektóre przekształcenia wymyślono specjalnie do tego celu. Na przykład w przekształceniu *Karhunena–Loevego* uzyskuje się najlepszy z możliwych współczynnik kompresji, lecz implementacja tego przekształcenia jest trudna. *Przekształcenie Fouriera* jest łatwe do stosowania, lecz nie daje wystarczającej kompresji. W tej silnej konkurencji różnych przekształceń zwycięzcą jest pokrewne przekształceniu Fouriera **dyskretne przekształcenie kosinusowe DCT** (*Discrete Cosine Transform*).

O ile w przekształceniu Fouriera stosuje się fale sinusoidalne i kosinusoidalne, to w DCT są stosowane tylko fale kosinusoidalne. Jest kilka wersji przekształcenia DCT, różniących się nieco od siebie pod względem matematycznym. Jako przykład jednej z wersji wyobraźmy sobie sygnał 129-punktowy, o próbkach od 0 do 128. Teraz zróbmy z niego sygnał 256-punktowy powielając próbki od 1 do 127 i uzupełniając nimi sygnał jako próbkami od 255 do 130. Mamy więc: 0, 1, 2, ..., 127, 128, 127, ..., 2, 1. Przeprowadziwszy przekształcenie Fouriera tego sygnału 256-punktowego, uzyskuje się 129-punktowe widmo częstotliwościowe, rozmieszczone od 0 do 128. Ponieważ wymuszono symetrię sygnału w dziedzinie czasu, więc część urojona widma będzie zawierała same zera. Innymi słowy, zaczęliśmy od 129-punktowego sygnału w dziedzinie czasu, a skończyliśmy na widmie częstotliwościowym o 129 punktach, z których każdy jest amplitudą fali kosinusoidalnej. Voilá, oto DCT!

Wykonując dyskretne przekształcenie kosinusowe grupy 8 × 8, uzyskuje się w wyniku widmo 8 × 8. Inaczej mówiąc, 64 liczby zostają zamienione na inne 64 liczby. Wszystkie te wartości są *rzeczywiste*, nie ma tu matematyki liczb zespolonych. Tak jak w analizie Fouriera, każda wartość w widmie jest amplitudą **funkcji bazowej**. Na rys. 27.10 przedstawiono 6 z 64 funkcji bazowych stosowanych w DCT 8 × 8, zgodnie z położeniami amplitud tych funkcji w widmie. Funkcje bazowe przekształcenia DCT 8 × 8 są określone wzorem:

$$b[x,y]=\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Wzór 27.1. Funkcje bazowe przekształcenia DCT. Zmienne x i y są indeksami w dziedzinie przestrzeni, a u i v indeksami w widmie częstotliwościowym. Dla DCT 8 × 8 wszystkie indeksy zmieniają się od 0 do 7

Grupa oryginalna

a. Brew

231	224	224	217	217	203	189	196
210	217	203	189	203	224	217	224
196	217	210	224	203	203	196	189
210	203	196	203	182	203	182	189
203	224	203	217	196	175	154	140
182	189	168	161	154	126	119	112
175	154	126	105	140	105	119	84
154	98	105	98	105	63	112	84

b. Oko

42	28	35	28	42	49	35	42
49	49	35	28	35	35	35	42
42	21	21	28	42	35	42	28
21	35	35	42	42	28	28	14
56	70	77	84	91	28	28	21
70	126	133	147	161	91	35	14
126	203	189	182	175	175	35	21
49	189	245	210	182	84	21	35

c. Nos

154	154	175	182	189	168	217	175
154	147	168	154	168	168	196	175
175	154	203	175	189	182	196	182
175	168	168	168	140	175	168	203
133	168	154	196	175	189	203	154
168	161	161	168	154	154	189	189
147	161	175	182	189	175	217	175
175	175	203	175	189	175	175	182

Widmo DCT

d. Widmo brwi

174	19	0	3	1	0	-3	1
52	-13	-3	-4	-4	-4	5	-8
-18	-4	8	3	3	2	0	9
5	12	-4	0	0	-5	-1	0
1	2	-2	-1	4	4	2	0
-1	2	1	3	0	0	1	1
-2	5	-5	-5	3	2	-1	-1
3	5	-7	0	0	0	-4	0

e. Widmo oka

70	24	-28	-4	-2	-10	-1	0
-53	-35	43	13	7	13	1	3
23	9	-10	-8	-7	-6	5	-3
6	2	-2	8	2	-1	0	-1
-10	-2	-1	-12	2	1	-1	4
3	0	0	11	-4	-1	5	6
-3	-5	-5	-4	3	2	-3	5
3	0	4	5	1	2	1	0

f. Widmo nosa

174	-11	-2	-3	-3	6	-3	4
-2	-3	1	2	0	3	1	2
3	0	-4	0	0	0	-1	9
-4	-6	-2	1	-1	4	-10	-3
1	2	-2	0	0	-2	0	-5
3	-1	3	-2	2	1	1	0
3	5	2	-2	3	0	4	3
4	-3	-13	3	-4	3	-5	3

Błąd kwantyzacji

g. Użycie 10 bitów

0	0	0	0	-1	0	0	0
-1	0	0	0	0	0	0	-1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	-1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

h. Użycie 8 bitów

0	-3	-1	-1	1	0	0	-1
1	0	-1	-1	0	0	0	-1
-1	-2	1	0	-2	0	-2	-2
-1	-2	-1	2	0	2	0	1
0	-2	1	0	0	1	0	0
0	-4	-1	0	1	0	0	0
0	-2	0	1	-1	-1	1	-1
-1	-3	1	1	1	-3	-2	-1

i. Użycie 5 bitów

-13	-7	1	4	0	0	10	-2
-22	6	-13	5	-5	2	-2	-13
-9	-15	0	-17	-8	8	12	25
-9	16	1	9	1	-5	-5	13
-20	-3	-13	-16	-19	-1	-4	-22
-11	6	-8	16	-9	-3	-7	6
-14	10	-9	4	-15	3	3	-4
-13	19	12	9	18	5	-5	10

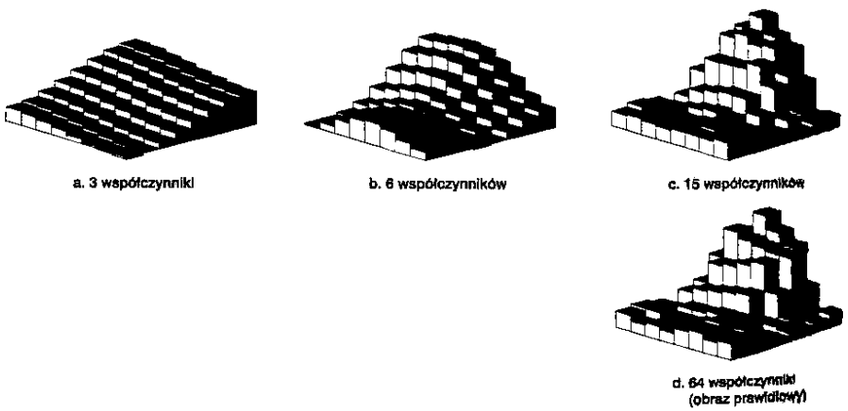
Rys. 27.11. Przykład kodowania JPEG. W lewej szpalcie przedstawiono trzy grupy po 8 × 8 pikseli, te same, co na rys. 27.9. W szpalcie środkowej pokazano widma DCT tych trzech grup. W trzeciej szpalcie podano błędy wartości pikselu bez kompresji, wynikające ze skończonej liczby bitów użytej do reprezentowania widma

Małe częstotliwości są usytuowane w górnym, lewym rogu widma, a duże – w dolnym, prawym. Składowa stała jest w punkcie [0,0], jest to wartość znajdująca się najbardziej z lewej strony, u góry. Funkcja bazowa dla [0,1] jest połową okresu fali kosinusoidalnej w jednym kierunku i wartością stałą – w drugim. Funkcja bazowa dla [1,0] jest podobna, lecz obrócona o 90°.

W przekształceniu DCT widmo jest obliczane przez *korelowanie* grupy pikseli 8 × 8 z każdą z funkcji bazowych. To znaczy, że każda wartość w widmie jest znajdowana przez przemnożenie odpowiedniej funkcji bazowej przez grupę pikseli 8 × 8 i następnie zsumowanie iloczynów. Do zakończenia obliczania DCT są jeszcze konieczne dwie korekty (tak jak w przekształceniu Fouriera). Po pierwsze, trzeba podzielić 15 wartości widma w wierszu 0 i kolumnie 0 przez *dwa*. Po drugie, należy podzielić wszystkie 64 wartości widma przez 16. Odwrotne przekształcenie DCT jest wykonywane przez przyporządkowanie każdej wartości widma właściwej funkcji bazowej i zsumowanie w celu odtworzenia dziedziny przestrzeni. Nie są potrzebne żadne dodatkowe kroki. Są to koncepcje takie same, jak w przekształceniu Fouriera, lecz z użyciem innych funkcji bazowych.

Na rys. 27.11 przedstawiono kodowanie JPEG dla trzech grup 8 × 8 zidentyfikowanych na rys. 27.9. W lewej szpalcie (rysunki 27.11 a, b, c) podano oryginalne wartości pikseli, a w środkowej (rys. 27.11 d, e, f) – widma DCT tych grup. W prawej kolumnie (rys. 27.11 g, h, i) zobrazowano wpływ redukowania liczby bitów stosowanych do reprezentowania każdej składowej w widmie częstotliwościowym. Na przykład grupa g) została utworzona przez obcinanie każdej z próbek w grupie d) do 10 bitów, wykonanie odwrotnego przekształcenia DCT i następnie odjęcie odtworzonego obrazu od oryginalnego. Podobnie grupy h) oraz i) tworzy się przez obcinanie każdej próbki w widmie, odpowiednio, do ośmiu i pięciu bitów. Jak się można było spodziewać, błąd odtwarzania wzrasta, gdy mniej bitów wykorzystuje się do reprezentowania danych. Jako przykład takiego obcinania bitowego, widma pokazane w środkowej szpalcie są reprezentowane przez 8 bitów na każdą wartość w widmie, ustawionych jako 0 do 255 dla składowej stałej i 127 do 127 dla innych wartości.

Druga metoda kompresji w dziedzinie częstotliwości polega na odrzuceniu niektórych z 64 wartości widma. Jak wynika z widm na rys. 27.11, prawie cały sygnał jest zawarty w składowych o małych częstotliwościach. Można więc wyeliminować składowe o większych częstotliwościach, tylko w niewielkim stopniu pogarszając jakość sygnału. Na rys. 27.12 przedstawiono przykład zniekształceń obrazu powstających w przypadku usuwania różnej liczby składowych o dużych częstotliwościach. Grupa 8 × 8 pikseli zastosowana w tym przykładzie jest obrazem



Rys. 27.12. Przykład odtwarzania JPEG. Zastosowana w tym przykładzie grupa 8 × 8 pikseli jest obrazem z rys. 27.9. Jak pokazano na rysunku, do uzyskania dobrego przybliżenia obrazu prawidłowego jest koniecznych mniej niż 1/4 z 64 wartości

a. Słaba kompresja

1	1	1	1	1	2	2	4
1	1	1	1	1	2	2	4
1	1	1	1	2	2	2	4
1	1	1	1	2	2	4	8
1	1	2	2	2	2	4	8
2	2	2	2	2	4	8	8
2	2	2	4	4	8	8	16
4	4	4	4	8	8	16	16

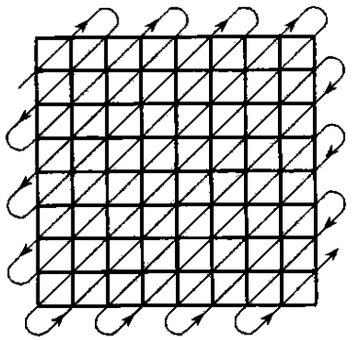
b. Silna kompresja

1	2	4	8	16	32	64	128
2	4	4	8	16	32	64	128
4	4	8	16	32	64	128	128
8	8	16	32	64	128	128	256
16	16	32	64	128	128	256	256
32	32	64	128	128	256	256	256
64	64	128	128	256	256	256	256
128	128	128	256	256	256	256	256

Rys. 27.13. Tablice kwantyzacji JPEG. Podano dwa przykłady tablic kwantyzacji JPEG, które mogą być stosowane podczas kompresji. Każda wartość w widmie DCT jest dzielona przez odpowiadającą jej wartość w tablicy kwantyzacji, a wynik zaokrągla się do najbliższej liczby całkowitej

oka z rys. 27.10. Na rys. 27.12 d) pokazano odtworzenie prawidłowe przy użyciu wszystkich 64 wartości widma. Pozostałe rysunki obrazują odtworzenie przy użyciu wskazanych liczb współczynników dotyczących najmniejszych częstotliwości. Usunięcie nawet 3/4 składowych o największych częstotliwościach daje niewielki błąd w odtworzonym obrazie (rys. 27.12 c). I to jest korzystne, że powstający błąd wygląda bardzo podobnie, jak szum losowy.

JPEG jest dobrym przykładem tego, jak w celu uzyskania lepszej skuteczności można połączyć ze sobą kilka sposobów kompresji danych. Cała procedura JPEG składa się z następujących kroków. Po pierwsze, obraz jest dzielony na grupy 8 × 8. Po drugie, dla każdej grupy wykonuje się przekształcenie DCT. Po trzecie, każde widmo 8 × 8 jest poddawane kompresji opisanymi powyżej metodami – przez redukowanie liczby bitów i eliminowanie niektórych składowych. To odbywa się w jednym kroku sterowanym **tablicą kwantyzacji**. Dwa przykłady tablic kwantyzacji podano na rys. 27.13 Każda wartość widma jest dzielona przez odpowiadającą jej wartość w tablicy kwantyzacji, a wynik jest zaokrąglany do najbliższej liczby całkowitej. Na przykład skrajna górna lewa wartość w tablicy kwantyzacji wynosi *jeden*, co powoduje, że wartość składowej stałej pozostanie niezmieniona. Natomiast skrajna dolna prawa wartość na rys. 27.13 a) jest 16, co oznacza, że oryginalny zakres od –127 do 127 zostaje zredukowany do tylko od –7 do 7. Inaczej mówiąc, wartość, pod względem jej dokładności, zredukowano z 8 do 4 bitów. W bardziej krańcowym przypadku skrajny dolny prawy zapis na rys. 27.13 b) jest równy 256, co całkowicie eliminuje wartość widmową.



Rys. 27.14. Przetwarzanie szeregowe JPEG. Uporządkowanie zygzakowate jest używane przy konwersji widma DCT 8 × 8 na sekwencję liniową złożoną z 64 wartości. W ten sposób wszystkie składowe o dużych częstotliwościach są grupowane razem tam, gdzie duża liczba zer może z dobrą skutecznością być poddana kompresji metodą kodowania ciągów



a. Obraz oryginalny



b. Z kompresją 10:1



c. Z kompresją 45:1

Rys. 27.15. Przykład zniekształceń spowodowanych kompresją JPEG. Na rys. a) pokazano obraz oryginalny, a na b) i c) obrazy odtworzone przy użyciu kompresji, odpowiednio, 10:1 i 45:1. Duży współczynnik kompresji zastosowany w obrazie c) powoduje, że każda grupa 8 × 8 pikseli jest reprezentowana przez mniej niż 12 bitów

W czwartym kroku kodowania JPEG zmodyfikowane widmo jest przetwarzane z matrycy 8 × 8 na sekwencję liniową. W tym kroku kodowania stosuje się uporządkowanie zygzakowate pokazane na rys. 27.14, umieszczając wszystkie duże częstotliwości razem na końcu sekwencji liniowej. W ten sposób *zera* z elementów usuwanych grupuje się w długie ciągi. W piątym kroku te długie ciągi zer poddaje się kompresji metodą kodowania ciągów. W szóstym kroku sekwencja jest kodowana przez kodowanie Huffmana lub arytmetyczne, dając końcowy plik skomprymowany.

Uruchamiając program kompresji JPEG, można wybrać wielkość kompresji i wynikające z niej pogorszenie jakości obrazu. Na rys. 27.15 pokazano ten rodzaj zniekształcenia obrazu, który wynika z dużych współczynników kompresji. Przy przedstawionym na rysunku współczynniku kompresji 45:1 każda z grup 8 × 8 jest reprezentowana przez tylko ok. 12 bitów. Dokładniejsze obejrzenie tego obrazu wykazuje, że sześć funkcji bazowych o najmniejszych częstotliwościach jest reprezentowanych do pewnego stopnia.

Dlaczego przekształcenie DCT jest lepsze do kompresji obrazu niż przekształcenie Fouriera? Główna przyczyna jest taka, że DCT ma półokresowe funkcje bazowe, tj. S[0,1] i S[1,0]. Jak pokazano na rys. 27.10, te funkcje delikatnie „nachylają się" od jednej strony matrycy ku drugiej. Natomiast najmniejsze częstotliwości w przekształceniu Fouriera tworzą *jeden pełny okres*. Obrazy prawie zawsze zawierają obszary, w których jaskrawość zmienia się stopniowo. Stosowanie funkcji bazowej dopasowanej do takiego wzorca bazowego umożliwia lepszą kompresję.

MPEG

MPEG jest standardem kompresji cyfrowych sekwencji sygnałów wizyjnych takich, jakie są stosowane w wideo, w komputerze i w sieciach telewizji cyfrowej. Ponadto, MPEG daje również kompresję ścieżki dźwiękowej towarzyszącej sygnałowi wizyjnemu. Nazwa standardu pochodzi od grupy roboczej, która go opracowała: *Moving Pictures Experts Group*. Jeśli uważacie, że JPEG jest skomplikowany, to MPEG jest koszmarem! MPEG po prostu się kupuje; nie próbujcie pisać takiego programu. Przyszłością tej techniki jest kodowanie algorytmów kompresji i dekompresji bezpośrednio w układach scalonych. Potencjalne możliwości MPEG są ogromne. Pomyślcie o tysiącach kanałów wideo przesyłanych do waszych domów pojedynczym kablem światłowodowym. Jest to podstawowa technika XXI wieku.

MPEG ma kilka innych ważnych właściwości, oprócz redukowania ilości przesyłanych danych. Film można odgrywać *w przód* i *w tył*, z szybkością *normalną* lub *zwiększoną*. Jest *bezpośredni dostęp* do zakodowanych informacji, co znaczy, że każda indywidualna ramka w sekwencji może być łatwo wyświetlona jako obraz nieruchomy. Temu również towarzyszy możliwość *redagowania* filmów, polegająca na tym, że krótkie segmenty z filmu można kodować tylko w odniesieniu do nich samych, a nie do całej sekwencji. Kompresję MPEG zaprojektowano w taki sposób, żeby była odporna na błędy. Ostatnią rzeczą, jaką chcielibyście napotkać podczas kompresji, jest błąd pojedynczego bitu powodujący przerwanie filmu.

Sposoby użyte w MPEG można podzielić na dwa typy kompresji: *wewnątrzramkową* i *międzyramkową*. W kompresji wewnątrzramkowej pojedyncze ramki tworzące sekwencję sygnału wizyjnego są kodowane tak, jakby były zwykłymi obrazami nieruchomymi. Taka kompresja jest wykonywana przy użyciu standardu JPEG, z kilkoma zmianami. W terminologii MPEG ramka zakodowana w ten sposób jest nazywana kodowaną wewnętrznie (*intra-coded*) lub **obrazem typu I.**

Większość pikseli w sekwencji sygnału wizyjnego niewiele się zmienia między jedną ramką a następną. Jeśli kamera się nie porusza, to większość obrazu zajmuje tło, które pozostaje stałe w kilkunastu kolejnych ramkach. W standardzie MPEG wykorzystano ten fakt, wprowadzając bardzo wysublimowaną postać *kodowania delta*, w celu dokonania kompresji informacji redundancyjnej *między ramkami*. Po dokonaniu kompresji jednej z ramek jako obrazu typu I,

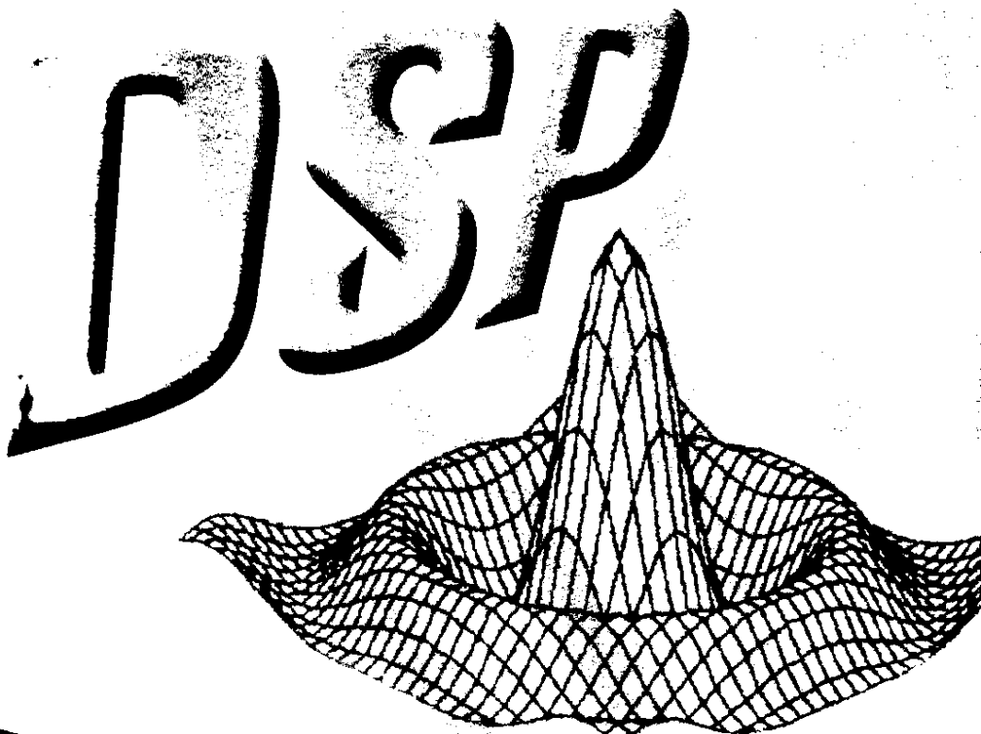
w MPEG koduje się kolejne ramki jako kodowane predykcyjnie, czyli **obrazy typu P**. Tak więc w obrazie typu P są zawarte tylko te piksele, które uległy zmianie po obrazie typu I.

Chociaż te dwa typy kompresji są podstawą MPEG, to jednak aktualna implementacja tej kompresji jest znacznie bardziej skomplikowana niż tu opisana. Na przykład obraz typu P może być odniesiony do obrazu typu I, który został *przesunięty* ze względu na ruch obiektów w sekwencji obrazowej. Są też obrazy kodowane predykcyjnie dwukierunkowo, czyli **obrazy typu B**. Mają one odniesienie zarówno do poprzedniego, jak i przyszłego obrazu typu I. W ten sposób operuje się w tych obszarach obrazu, które zmieniają się stopniowo podczas wielu ramek. Pojedyncze ramki mogą też być nie po kolei zapisywane w danych skomprymowanych, aby ułatwić właściwe uszeregowanie obrazów typu I, P i B. Dodanie barwy i dźwięku jeszcze bardziej komplikuje taką kompresję.

Największe zniekształcenia związane z MPEG występują wówczas, gdy duże segmenty obrazu szybko się zmieniają. W rezultacie jest potrzebny spory pakiet informacji, aby nadążyć za gwałtownie zmieniającymi się scenami. Jeśli bitowa przepływność danych jest ustalona, to występuje tzw. efekt blokowy – pojawianie się wyraźnych fałszywych krawędzi w miejscach podziału obrazu na segmenty. Można zminimalizować to zjawisko w sieciach przesyłających jednocześnie wiele kanałów wideo, takich jak telewizja kablowa. Nagle pojawiający się pakiet informacji potrzebny do podtrzymania szybko zmieniającej się sceny w jednym kanale wideo jest uśredniany przez umiarkowane wymagania dość statycznych scen w innych kanałach.

28

Cyfrowe procesory sygnałowe



Cyfrowe przetwarzanie sygnałów jest przeprowadzane przy użyciu operacji matematycznych w odróżnieniu na przykład od przetwarzania tekstów lub innych podobnych programów, w których tylko przemieszcza się zgromadzone dane. Wynika stąd, że komputery zaprojektowane do zastosowań w ekonomii i innych aplikacji ogólnych nie są zoptymalizowane dla algorytmów takich, jak filtracja cyfrowa i analiza fourierowska. *Cyfrowe procesory sygnałowe* są mikroprocesorami specjalnie zaprojektowanymi do realizacji zadań cyfrowego przetwarzania sygnałów. W ciągu ostatnich kilkunastu lat nastąpił niebywały rozwój tych urządzeń, które znalazły zastosowanie wszędzie – od telefonii komórkowej aż po wysokiej klasy aparaturę badawczą. W istocie projektanci układów stosują skrót DSP na określenie *cyfrowego procesora sygnałowego*, a projektanci oprogramowania – *cyfrowego przetwarzania sygnałów*. W tym rozdziale będzie mowa o tym, czym procesory DSP różnią się od innych rodzajów mikroprocesorów, a także o tym, jak stwierdzić, czy dany procesor jest odpowiedni do konkretnego zastosowania oraz jak rozpocząć pracę w tej nowej ekscytującej dziedzinie. W następnym rozdziale omówimy bardziej szczegółowo jeden z tych bardzo zaawansowanych technicznie produktów – *rodzinę SHARC®* firmy Analog Devices.

Czym cyfrowe procesory sygnałowe różnią się od innych mikroprocesorów

W latach 60. przewidywano, że sztuczna inteligencja zrewolucjonizuje współdziałanie człowieka z komputerem i innymi urządzeniami. Wierzono, że z końcem XX wieku będą istniały już roboty do sprzątania domów, komputery kierujące samochodami i interfejsy głosowe eliminujące gromadzeniem i wyszukiwaniem informacji. Tak się jednak nie stało. Te zadania okazały się znacznie bardziej skomplikowane, niż przewidywano i bardzo trudne do wykonania przy użyciu funkcji logicznych realizowanych krok po kroku przez komputery cyfrowe.

Jednakże, w czasie ostatnich czterdziestu lat okazało się, że komputery mają niezwykłe możliwości w dwóch wielkich dziedzinach, którymi są: 1) **operowanie danymi**, takie jak przetwarzanie tekstu i zarządzanie bazami danych, 2) **obliczenia matematyczne**, stosowane w nauce, technice i w cyfrowym przetwarzaniu sygnałów. Wszystkie mikroprocesory mogą wykonywać oba te zadania. Jednak trudne (i kosztowne) jest opracowanie urządzenia *zoptymalizowanego* do obu zadań. W projektowaniu układów są konieczne kompromisy uwzględniające przeciwstawne wymogi związane z różnymi parametrami – np. z rozmiarem zestawu instrukcji i łatwością obsługi przerwań. Jeszcze ważniejsze są problemy związane z wprowadzaniem mikroprocesora na *rynek* – koszt opracowania i wytwarzania, konkurencyjność, czas życia wyrobu itd. Ogólnie biorąc, te problemy dotyczą także tradycyjnych mikroprocesorów takich jak Pentium®, ukierunkowanych przede wszystkim na operowanie danymi. Procesory sygnałowe zaś są przeznaczone do obliczeń matematycznych niezbędnych w cyfrowym przetwarzaniu sygnałów.

Na rys. 28.1 zestawiono najważniejsze różnice między tymi dwiema kategoriami mikroprocesorów. Operowanie danymi obejmuje zapamiętywanie i sortowanie informacji. Rozważmy na przykład program do przetwarzania tekstu. Podstawowym jego zadaniem jest zapamiętywanie informacji (wprowadzanej z klawiatury przez operatora), organizowanie informacji (wycinanie i wstawianie, sprawdzanie pisowni, układ stron itd.), a następnie odzyskiwanie informacji (np. zapisanie informacji na dyskietce lub wydrukowanie jej przez drukarkę laserową). Te zadania są realizowane przez *przenoszenie* danych z jednej lokalizacji do innej i *sprawdzanie* nierówności, czyli porównywania ($A = B$, $A < B$ itd.). Jako przykład weźmy sortowanie listy słów w porządku alfabetycznym. Każde słowo jest reprezentowane przez liczbę 8-bitową, wartość ASCII pierwszej litery słowa. Układanie w porządku alfabetycznym polega na przestawianiu kolejności słów tak długo aż wartości ASCII będą w sposób ciągły wzrastały od początku do końca listy. Można tego dokonać przez wielokrotne powtarzanie dwóch kroków aż do zakończenia porządkowania według alfabetu. Po pierwsze, sprawdza

	Operowanie danymi	Obliczenia matematyczne
Typowe zastosowania	Przetwarzanie tekstów, zarządzanie bazami danych, arkusze kalkulacyjne, systemy operacyjne itd.	Cyfrowe przetwarzanie sygnałów, sterowanie ruchem, symulacja w nauce i technice itd.
Najważniejsze operacje	Przenoszenie danych (A → B) Instrukcje warunkowe (if $A = B$ then ...)	Dodawanie ($A + B = C$) Mnożenie ($A \times B = C$)

Rys. 28.1. Operowanie danymi a obliczenia matematyczne. Komputery cyfrowe są stosowane do realizacji dwóch głównych zadań – operowania danymi i obliczeń matematycznych. Operowanie danymi polega na ich przenoszeniu i sprawdzaniu nierówności (czyli porównywaniu), podczas gdy w operacjach matematycznych korzysta się z mnożeń i dodawań

się, czy dwa sąsiednie zapisy są ustawione w kolejności alfabetycznej (IF A > B THEN...). Po drugie, jeśli te dwa zapisy nie są w kolejności alfabetycznej, to trzeba je przesunąć, aby tak było (A ⇄ B). Jeśli ten dwukrokowy proces jest powtarzany wiele razy na wszystkich sąsiadujących parach, to lista w końcu stanie się alfabetyczna.

Jako inny przykład, rozpatrzmy drukowanie dokumentu z procesora tekstu. Komputer w sposób ciągły testuje urządzenie wejściowe (myszkę lub klawiaturę), sprawdzając, czy nie pojawi się kod oznaczający „drukuj dokument". Po wykryciu tego kodu program przenosi dane z pamięci komputera do drukarki. Mamy tu znowu te same dwie operacje podstawowe – przenoszenie danych i sprawdzanie nierówności. W zastosowaniach tego rodzaju operacje matematyczne są stosowane od czasu do czasu, lecz niezbyt często i nie mają znaczącego wpływu na całkowity czas wykonania programu.

Natomiast szybkość wykonywania większości algorytmów DSP jest ograniczona prawie całkowicie przez wymaganą liczbę mnożeń i dodawań. Na przykład na rys. 28.2 przedstawiono implementację filtru cyfrowego SOI, a więc najczęściej spotykaną technikę DSP. Przy zastosowaniu zapisu standardowego sygnał wejściowy jest oznaczany jako $x[\ ]$, a wyjściowy jako $y[\ ]$. Naszym zadaniem jest obliczenie próbki w sygnale wyjściowym o położeniu n, tj. $y[n]$. Filtr SOI wykonuje to obliczenie, przemnażając odpowiednie próbki sygnału wejściowego przez grupę współczynników oznaczonych jako: a_0, a_1, a_2, a_3, ... i następnie sumując uzyskane iloczyny. Sygnał wyjściowy $y[n]$ można więc wyrazić wzorem:

$$y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + a_3x[n-3] + a_4x[n-4] + \ldots$$

Jest to po prostu stwierdzenie, że sygnał wejściowy został poddany *splotowi* z odpowiedzią impulsową filtru składającą się z: a_0, a_1, a_2, a_3, ... Zależnie od zastosowania odpowiedź impulsowa filtru może zawierać tylko kilka współczynników albo nawet kilka tysięcy. W tym algorytmie główny wpływ na czas wykonania mają operacje matematyczne, choć są tu też operacje przesuwania danych i szacowania nierówności, używane na przykład do utrzymywania rezultatów pośrednich i do pętli sterowania.

Procesor DSP, prócz tego że powinien wykonywać operacje matematyczne bardzo szybko, to musi mieć *przewidywalny* czas wykonania algorytmu. Powiedzmy, że dajecie waszemu komputerowi zadanie np. przekształcenia jakiegoś dokumentu z jednej postaci w inną. Wtedy nie ma znaczenia, czy przetwarzanie trwa dziesięć *milisekund*, czy dziesięć *sekund*. Po prostu czekacie, aż zadanie będzie wykonane i wtedy możecie zadać komputerowi następną pracę.

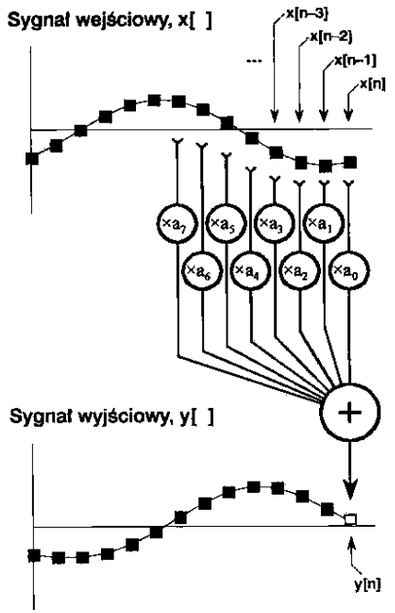
Natomiast DSP stosuje się głównie w takich aplikacjach, gdzie przetwarzanie jest *ciągłe*, niemające ani początku, ani końca. Powiedzmy, że inżynier projektuje system DSP dla sygnału fonicznego, np. do aparatów słuchowych. Jeśli sygnał cyfrowy jest pobierany z szybkością 20 000 próbek/s, to DSP musi być w stanie utrzymać przepływność 20 000 próbek/s. Są jednak

ważne przyczyny przemawiające za tym, aby nie projektować systemu na szybkość większą od koniecznej. Ze wzrostem szybkości wzrasta bowiem także *koszt*, *pobór mocy*, *trudność projektowania* itd. To powoduje, że znajomość dokładnej wartości czasu wykonania jest niezbędna w celu wyboru właściwego układu, jak również algorytmów, które mogą być zastosowane.

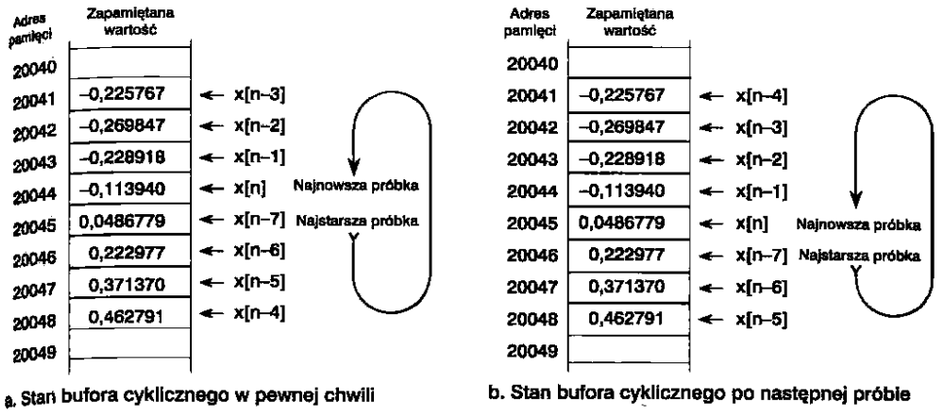
Buforowanie cykliczne

Cyfrowe procesory sygnałowe są zaprojektowane w taki sposób, aby mogły szybko realizować filtry SOI i podobne zadania. Aby zrozumieć *rozwiązania układowe*, musimy najpierw zrozumieć *algorytmy*. W tym podrozdziale przedstawimy szczegółową listę kroków koniecznych do implementacji filtru SOI. W następnym podrozdziale zobaczymy, jak projektuje się procesory DSP, aby realizowały te kroki w sposób możliwie najskuteczniejszy.

Na początku musimy rozróżnić przetwarzanie **autonomiczne**, czyli **off-line** i przetwarzanie **w czasie rzeczywistym**. W przypadku przetwarzania autonomicznego *cały* sygnał wejściowy jednocześnie znajduje się w komputerze. Na przykład geofizyk może wykorzystać sejsmometr do rejestracji ruchów skorupy ziemskiej podczas trzęsienia ziemi. Gdy wstrząsy ustaną, zebrana informacja może być wczytana do komputera i w jakiś sposób analizowana. Innym przykładem przetwarzania autonomicznego jest obrazowanie medyczne, np. tomografia komputerowa i obrazowanie za pomocą rezonansu magnetycznego. Zestaw danych jest pobierany, gdy pacjent jest wewnątrz urządzenia, a odtworzenie obrazu może być wykonane z opóźnieniem. Kluczowy jest tu fakt, że *cała* informacja jest jednocześnie dostępna dla programu przetwarzającego. Taka sytuacja występuje często w badaniach naukowych i w technice, lecz nie w produktach powszechnego użytku. Przetwarzanie autonomiczne (off-line) jest domeną zarówno osobistych, jak i dużych komputerów.



Rys. 28.2. Filtr cyfrowy SOI. W filtracji SOI każda próbka w sygnale wyjściowym y[n] jest wyznaczana przez przemnażanie próbek sygnału wejściowego x[n], x[n – 1], x[n – 2], ... przez współczynniki odpowiedzi impulsowej filtru a_0, a_1, a_2, a_3, ... i zsumowanie uzyskanych iloczynów



Rys. 28.3. Działanie bufora cyklicznego. Bufory cykliczne są stosowane do przechowywania ostatnich poprzednich wartości sygnału, który jest ciągle uaktualniany. Na rysunku przedstawiono stan 8-próbkowego bufora cyklicznego w jakiejś chwili (rys. a) i jego stan o jedną próbkę później (rys. b)

W przypadku przetwarzania w czasie rzeczywistym sygnał wyjściowy jest wytwarzany w tym samym czasie, gdy jest pobierany sygnał wejściowy. Jest to niezbędne na przykład w łączności telefonicznej, aparatach słuchowych i radarze. W tych zastosowaniach informacja musi być dostępna natychmiast, choć może być nieco opóźniona. Na przykład opóźnienie o 10 ms w rozmowie telefonicznej pozostanie niezauważone zarówno przez mówiącego, jak i słuchającego. Również nie ma znaczenia opóźnienie sygnału radarowego o kilka sekund przed jego wyświetleniem operatorowi. W zastosowaniach z przetwarzaniem w czasie rzeczywistym następuje pobranie próbki, wykonanie algorytmu i wyprowadzenie próbki, i tak wielokrotnie. Może być też pobranie grupy próbek, wykonanie algorytmu i wyprowadzenie grupy próbek. To jest właśnie świat cyfrowych procesorów sygnałowych.

Wróćcie teraz do rys. 28.2 i wyobraźcie sobie, że jest to filtr SOI realizowany *w czasie rzeczywistym*. Aby obliczyć próbkę wyjściową, musimy mieć dostęp do pewnej liczby ostatnich poprzednich próbek wejściowych. Przypuśćmy, że w filtrze stosujemy osiem współczynników a_0, a_1, ... a_7. To znaczy, że musimy znać wartości ostatnich ośmiu próbek sygnału wejściowego $x[n]$, $x[n-1]$, ... $x[n-7]$. Te osiem próbek trzeba przechowywać w pamięci i ciągle uaktualniać w miarę pobierania nowych próbek. Jaki jest najlepszy sposób operowania tymi zapamiętanymi próbkami? Odpowiedź brzmi – **buforowanie cykliczne**.

Na rys. 28.3 przedstawiono 8-próbkowy bufor cykliczny. Umieściliśmy ten bufor w ośmiu kolejnych adresach w pamięci od 20 041 do 20 048. Na rys. 28.3 a) pokazano, w jaki sposób osiem próbek wejściowych może być zapamiętanych w jakiejś określonej chwili, na rys. 28.3 b) zaś przedstawiono zmiany, jakie nastąpiły po pobraniu następnej próbki. Zasada buforowania cyklicznego polega na tym, że koniec matrycy liniowej jest dołączony do jej początku; lokacja pamięci 20 041 jest widziana jako sąsiednia z 20 048, tak samo jak 20 044 jest sąsiednią z 20 045. Śledzi się działanie pamięci za pomocą **wskaźnika** (zmiennej, której wartość jest zawarta w *adresie*) pokazującego, gdzie znajduje się najnowsza próbka. Na przykład na rys. 28.3 a) wskaźnik pokazuje adres 2044, a na rys. 28.3 b) – adres 2045. Nowo pobrana próbka zastępuje najstarszą próbkę w matrycy i wskaźnik jest przesuwany o jeden adres w przód. Bufory cykliczne są użyteczne, gdyż przy pobraniu nowej próbki wystarczy zmienić tylko jedną wartość. Do posługiwania się buforem cyklicznym są konieczne cztery parametry. Po pierwsze, musi być wskaźnik pokazujący początek bufora cyklicznego w pamięci (w tym przykładzie 20 041). Po drugie, potrzebny jest wskaźnik pokazujący koniec matrycy (na przykład 20 048) lub zmienna zawierająca długość matrycy (np. 8). Po trzecie,

musi być określona wartość kroku adresowania pamięci. Na rys. 28.3 wartość kroku wynosi *jeden*, np. pod adresem 20 043 jest zawarta jedna próbka, pod 20 044 następna itd. Często bywa inaczej. Na przykład adresowanie może być wykonywane zgodnie z bajtami i do przechowywania wartości każdej próbki może być wymagane dwa lub cztery bajty. W takich przypadkach wartość kroku mogłaby być, odpowiednio, dwa lub cztery.

Wartości tych trzech parametrów określają rozmiar i konfigurację bufora cyklicznego i nie zmieniają się podczas działania programu. Czwarta wartość, wskaźnik ostatnio pobranej próbki powinien być modyfikowany przy pobraniu każdej kolejnej próbki. Innymi słowy musi istnieć logika programowa sterująca sposobem uaktualniania wskaźnika ostatnio pobranej próbki, na podstawie wartości trzech poprzednio wymienionych parametrów. Ta logika jest wprawdzie całkiem prosta, ale musi być bardzo szybka. I to jest podstawowy punkt rozważań; procesor DSP powinien być zoptymalizowany pod względem posługiwania się buforem cyklicznym w taki sposób, żeby osiągnąć możliwie jak największą szybkość wykonywania programu.

Buforowanie cykliczne jest dodatkowo przydatne w przetwarzaniu *autonomicznym* (*off-line*). Rozważmy program, w którym tak sygnał wejściowy, jak i wyjściowy są w całości zawarte w pamięci. Buforowanie cykliczne nie jest tu konieczne do obliczania splotu, ponieważ wszystkie próbki są w każdej chwili natychmiast dostępne. Jednak wiele algorytmów jest realizowanych *etapowo*, z sygnałem pośrednim tworzonym między kolejnymi etapami. W ten sposób działa na przykład filtr rekursywny realizowany jako szereg stopni bikwadratowych. Metodą najprostszą jest przechowanie w pamięci każdego sygnału pośredniego w pełnej jego długości. Buforowanie cykliczne daje inną możliwość – zapamiętanie tylko tych pośrednich próbek, które są niezbędne do natychmiastowych obliczeń. W ten sposób redukuje się wymaganą wielkość pamięci kosztem bardziej skomplikowanego algorytmu. Ważny wniosek: bufory cykliczne są *przydatne* w przetwarzaniu off-line, a *niezbędne* w systemach pracujących w czasie rzeczywistym.

Spójrzmy teraz, jakie kroki trzeba wykonać, aby zrealizować filtr SOI, stosując bufory cykliczne zarówno dla sygnału wejściowego, jak i dla współczynników. Ta lista może wydać się trywialna i zbyt szczegółowa – ale wcale tak nie jest! Skuteczne posługiwanie się poszczególnymi zadaniami jest tym, co odróżnia DSP od tradycyjnego mikroprocesora. Dla każdej nowej próbki powinny zostać podjęte kroki wymienione w tablicy 28.1.

Tablica 28.1. Kroki konieczne do realizacji filtru SOI

1.	Pobierz próbkę z przetwornika a/c; wygeneruj sygnał przerwania
2.	Wykryj i zrealizuj przerwanie
3.	Przenieś próbkę do bufora cyklicznego sygnału wejściowego
4.	Zaktualizuj wskaźnik bufora cyklicznego sygnału wejściowego
5.	Wyzeruj akumulator
► 6.	Steruj pętlą przechodzącą przez każdy ze współczynników
7.	Pobierz współczynnik z cyklicznego bufora współczynników
8.	Uaktualnij wskaźnik dla cyklicznego bufora współczynników
9.	Pobierz próbkę z cyklicznego bufora sygnału wejściowego
10.	Uaktualnij wskaźnik dla cyklicznego bufora sygnału wejściowego
11.	Pomnóż współczynnik przez próbkę
└ 12.	Uzyskany iloczyn dodaj do akumulatora
13.	Przenieś próbkę wyjściową (akumulator) do bufora wyjściowego
14.	Przenieś próbkę wyjściową z bufora wyjściowego do przetwornika c/a

Celem jest szybkie wykonanie kolejnych kroków. Szczególną uwagę trzeba zwrócić na kroki od 6 do 12, gdyż są one wykonywane wielokrotnie – tyle razy, ile jest współczynników filtru. W mikroprocesorach tradycyjnych tych 14 kroków musi być realizowanych *szeregowo*, procesor DSP zaś jest tak zaprojektowany, aby wykonywać je *równolegle*. W niektórych przypadkach wszystkie operacje objęte pętlą (kroki 6–12) mogą być całkowicie zrealizowane w ciągu *jednego okresu zegarowego*. Przyjrzyjmy się architekturze wewnętrznej, która umożliwia takie wspaniałe działanie.

Architektura cyfrowego procesora sygnałowego

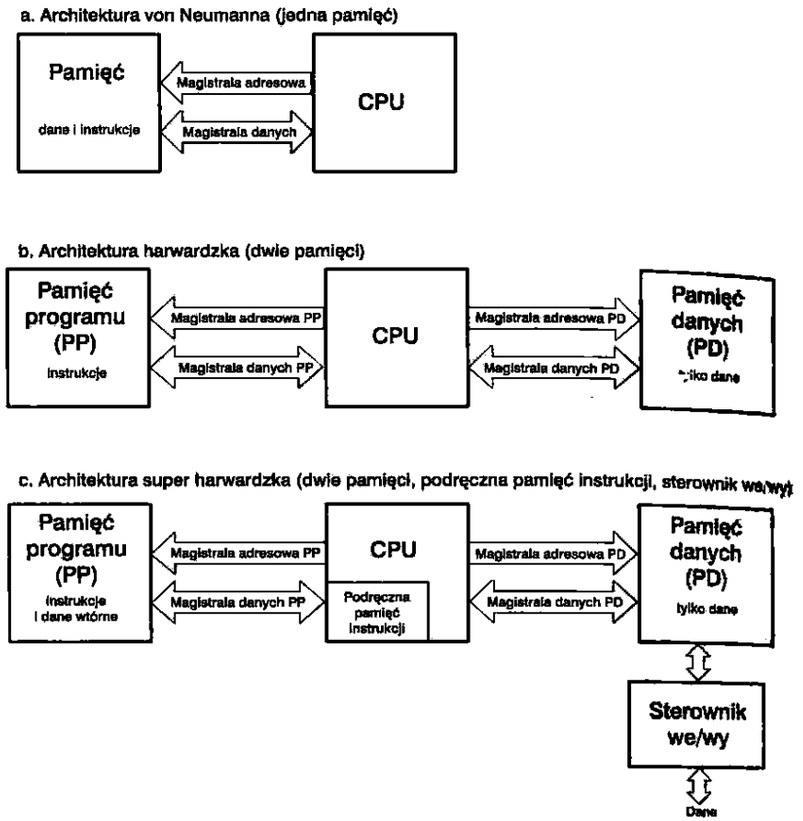
Jednym z ważniejszych „wąskich gardeł" w realizacji algorytmów DSP jest przesyłanie informacji do i z pamięci. To dotyczy *danych*, takich jak próbki sygnału wejściowego i współczynniki filtru, a także *instrukcji programu*, czyli słów binarnych kodów binarnych wchodzących do sekwensera programowego. Powiedzmy, że chcemy przemnożyć przez siebie dwie liczby umieszczone gdzieś w pamięci. Aby to wykonać, musimy pobrać z pamięci trzy wartości binarne, a więc dwie liczby przemnażane oraz instrukcję programową określającą, co ma być zrobione.

Na rys. 28.4 a) przedstawiono, w jaki sposób to proste zadanie jest wykonywane w tradycyjnym mikroprocesorze. Te mikroprocesory mają tzw. **architekturę von Neumanna**, biorącą swoją nazwę od błyskotliwego amerykańskiego matematyka Johna von Neumanna (1903–1957). Von Neumann kierował opracowaniami matematycznymi wielu ważnych odkryć pierwszej połowy XX wieku. Do jego licznych osiągnięć należą: opracowanie koncepcji komputera sterowanego programem umieszczonym w pamięci, matematyczne sformalizowanie mechaniki kwantowej i prace nad bombą atomową. Jeśli pojawiało się coś nowego i ekscytującego, to von Neumann tam był.

System wg architektury von Neumanna (rys. 28.4 a) zawiera pojedynczą pamięć i pojedynczą magistralę do przesyłania danych do i z jednostki centralnej (CPU). Mnożenie dwóch liczb wymaga co najmniej trzech okresów zegarowych, po jednym na przesłanie magistralą każdej z trzech liczb z pamięci, do CPU. Nie wliczamy tu czasu na przesłanie wyniku z powrotem do pamięci zakładając, że pozostaje on w CPU do dodatkowych operacji, takich jak sumowanie iloczynów w filtrze SOI. System von Neumanna jest całkiem zadowalający, gdy wystarcza wam realizacja wszystkich wymaganych zadań w sposób szeregowy. W istocie większość obecnie stosowanych komputerów ma architekturę von Neumanna. Inna architektura jest potrzebna tylko wówczas, gdy przetwarzanie ma być bardzo szybkie i gdy jesteśmy skłonni uzyskać to kosztem większej złożoności systemu.

Ta myśl prowadzi nas do **architektury harwardzkiej** (rys. 28.4 b). Jej nazwa pochodzi od prac wykonywanych na Uniwersytecie Harvarda, w latach 40., pod kierunkiem Howarda Aikena (1900–1973). Jak pokazano na rysunku, Aiken wprowadził oddzielne pamięci dla danych i dla instrukcji programu wraz z oddzielnymi magistralami. Ponieważ magistrale działają oddzielnie, więc instrukcje programu i dane mogą być pobierane w tym samym czasie, co poprawia szybkość w stosunku do pracy z pojedynczą magistralą. Taką architekturę z podwójną magistralą stosuje się teraz w większości procesorów DSP.

Na rys. 28.4 c) zilustrowano jeszcze wyższy poziom zaawansowania technicznego – **architekturę superharwardzką**. Tę nazwę wymyślono w firmie Analog Devices, aby opisać wewnętrzne działanie rodzin cyfrowych procesorów sygnałowych ADSP-2106x i ADSP-211xx tej firmy. Są to tzw. **SHARC®DSP**, jako skrót od Super Harvard ARChitecture. Udoskonalono działanie architektury harwardzkiej, wprowadzając kilka nowych rozwiązań. Chociaż architektura superharwardzka jest zoptymalizowana bardzo wieloma sposobami, to dwa rozwiązania są tak ważne, że zostały uwzględnione na rys. 28.4 c). Są to *podręczna pamięć instrukcji* (*instruction cache*) oraz *sterownik wejście-wyjście*.



Rys. 28.4. Architektura mikroprocesorów. W architekturze von Neumanna stosuje się jedną pamięć do przechowywania zarówno danych, jak i instrukcji. W architekturze harwardzkiej zaś są oddzielne pamięci dla danych i instrukcji, dzięki czemu uzyskuje się większą szybkość działania. Architektura superharwardzka jest udoskonaleniem harwardzkiej przez dodanie podręcznej pamięci instrukcji i sterownika wejście-wyjście

Najpierw zastanówmy się, jak podręczna pamięć instrukcji ulepsza działanie systemu o architekturze harwardzkiej. Niekorzystną cechą podstawowej struktury harwardzkiej jest fakt, że magistrala pamięci danych jest bardziej zajęta niż magistrala pamięci programu. Podczas mnożenie dwóch liczb dwie wartości binarne (dwie liczby) muszą być przesłane magistralą pamięci danych, a tylko jedna (instrukcja programu) – magistralą pamięci programu. Chcąc poprawić tę sytuację, zaczynamy od przemieszczenia części „danych" do pamięci programu. Możemy umieścić na przykład współczynniki filtru w pamięci programu, pozostawiając sygnał wejściowy w pamięci danych. Na rysunku te przemieszczone dane nazwano „danymi wtórnymi". Na pierwszy rzut oka wydaje się, że to nie poprawia sytuacji; musimy teraz przesłać jedną wartość magistralą pamięci danych (próbkę sygnału wejściowego), lecz dwie (instrukcję programu i współczynnik) – magistralą pamięci programu. Tak więc, w zasadzie gdybyśmy wykonywali instrukcje wybrane losowo, to sytuacja wcale by się nie polepszyła.

Jednak większość czasu wykonania algorytmów DSP zajmują pętle, takie jak instrukcje 6–13 w tablicy 28.1. Z tego powodu ten sam zestaw instrukcji programu będzie ciągle przechodził z pamięci programu do jednostki centralnej. W architekturze superharwardzkiej wykorzystano

tę sytuację, włączając do jednostki centralnej **podręczną pamięć instrukcji**. Jest to mała pamięć zawierająca ok. 32 ostatnich instrukcji programu. Przy pierwszym przejściu pętli instrukcje muszą być przesłane przez magistralę pamięci programu. To powoduje spowolnienie działania ze względu na to, że współczynniki też muszą być pobierane tą drogą. Jednak przy dalszych wykonaniach pętli instrukcje programu mogą być pobierane z pamięci podręcznej. Dzięki temu całe przesyłanie informacji z pamięci do jednostki centralnej może być zrealizowane podczas jednego okresu zegarowego – próbka sygnału wejściowego przechodzi przez magistralę pamięci danych, współczynnik – przez magistralę pamięci programu, a instrukcja programu jest pobierana z podręcznej pamięci instrukcji. W potocznym języku technicznym takie wydajne przesyłanie danych nazywa się *szerokopasmowym dostępem do pamięci* (*high memory access bandwidth*).

Na rys. 28.5 bardziej szczegółowo przedstawiono architekturę procesorów rodziny SHARC, pokazując **sterownik wejścia-wyjścia** dołączony do pamięci danych. Tą drogą sygnały wchodzą do systemu i z niego wychodzą. Na przykład procesory DSP SHARC mają porty komunikacyjne tak szeregowe, jak i równoległe. To są połączenia dające skrajnie dużą szybkość. Na przykład przy częstotliwości zegara 40 MHz[29] są dwa porty szeregowe, każdy działający z przepływnością bitową 40 Mbit/s, podczas gdy każdy z sześciu portów równoległych ma przepływność bitową 40 MB/s. Gdy wykorzystuje się wszystkie sześć portów równoległych razem, to osiąga się bardzo dużą przepływność bitową 240 MB/s.

To wystarczyłoby do przesłania całego tekstu tej książki w ciągu zaledwie 2 ms! Równie ważne jest to, że specjalnie zaprojektowane układy umożliwiają przesyłanie tych strumieni danych bezpośrednio do pamięci (bezpośredni dostęp do pamięci – DMA) bez konieczności przekazywania przez rejestry jednostki centralnej (CPU). Inaczej mówiąc, zadania 1 i 14 z naszej listy w tablicy 28.1 są realizowane niezależnie i równocześnie z innymi zadaniami, nie zajmując żadnych okresów zegara jednostki centralnej. Główne magistrale (pamięci programu i pamięci danych) są również dostępne na zewnątrz układu scalonego, tworząc dodatkowy interfejs do pamięci zewnętrznych i urządzeń peryferyjnych. To umożliwia procesorom sygnałowym SHARC stosowanie pamięci o pojemności czterech gigasłów (16 GB), dostępnej z przepływnością bitową 40 megasłów/s (160 MB/s) dla danych 32-bitowych. To jest coś!

Tego rodzaju duża szybkość działania układu wejście-wyjście jest charakterystyczna dla procesorów DSP. Dominującym celem jest wprowadzenie danych, wykonanie operacji matematycznych i wyprowadzenie danych, zanim pojawi się następna próbka. Wszystko poza tym ma znaczenie drugorzędne. Niektóre procesory DSP zawierają w jednej strukturze także przetworniki a/c i c/a. Takie procesory są nazywane układami o **sygnałach mieszanych**. Wszystkie procesory DSP mają możliwość połączenia z przetwornikami zewnętrznymi przez porty szeregowe lub równoległe.

Zajrzyjmy teraz do wnętrza jednostki centralnej (CPU). W górnej części schematu (rys. 28.5) są dwa bloki nazwane **generator adresów danych** (DAG – *Data Address Generator*), po jednym dla każdej z dwóch pamięci. Sterują one adresami przesyłanymi do pamięci programu i danych, określając, skąd informacja ma być odczytywana i gdzie wpisywana. W prostszych mikroprocesorach to zadanie jest realizowane jako nieodłączna część sekwensera programu i jest zupełnie przejrzysta dla programisty. Jednak procesory DSP są zaprojektowane do pracy z *buforami cyklicznymi* i uzyskuje się w nich korzyści dzięki dodatkowym układom, skutecznie operującym tymi buforami. W ten sposób unika się wykorzystywania cennych okresów zegarowych CPU na śledzenie przebiegu gromadzenia danych. Na przykład w procesorach sygnałowych SHARC każdy z dwóch generatorów adresów danych steruje *ośmioma* buforami cyklicznymi. To znaczy, że w każdym DAG są przechowywane 32 zmienne (4 na bufor) wraz z niezbędnymi funkcjami logicznymi.

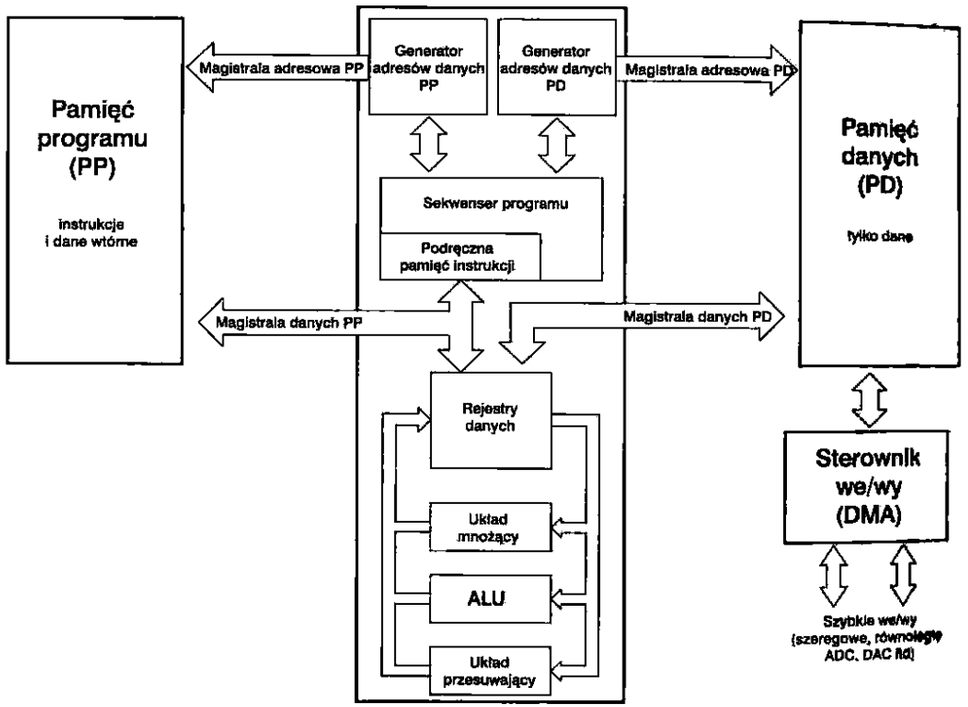
Po co tak wiele buforów cyklicznych? Niektóre algorytmy DSP są najlepiej realizowane etapami. Filtry SOI, na przykład, są bardziej stabilne, gdy się je realizuje jako kaskadę stopni

[29] Współcześnie produkowane procesory z rodziny SHARC są taktowane przebiegami zegarowymi o częstotliwości od 150 MHz (ADSP-21261) aż do 400 MHz (ADSP-21368 i ADSP-21369).

bikwadratowych. Stopnie wielokrotne, aby pracowały z dużą szybkością, wymagają wielokrotnych buforów cyklicznych. Generatory adresów w procesorach SHARC są zaprojektowane również w taki sposób, aby skutecznie można było realizować *szybkie przekształcenie Fouriera*. W tym rodzaju pracy generatory adresów są tak skonfigurowane, aby dawały **adresy z odwracaniem bitowym** do buforów cyklicznych, co jest niezbędną częścią algorytmu FFT. Ponadto obfitość buforów cyklicznych znacznie upraszcza pisanie programów dla DSP – zarówno programiście – człowiekowi, jak i kompilatorom języków wysokiego poziomu, takich jak C.

Rejestry danych CPU są wykorzystywane w taki sam sposób, jak w tradycyjnych mikroprocesorach. W procesorze sygnałowym ADSP-2106x SHARC jest 16 rejestrów ogólnego przeznaczenia, każdy 40-bitowy. Mogą one przechowywać dane przy obliczeniach pośrednich, przygotowywać dane dla procesora matematycznego, służyć jako bufor do przesyłania danych, przechowywać flagi do sterowania programem i tak dalej. W razie potrzeby te rejestry mogą też być stosowane do sterowania pętlami i licznikami; jednak procesory SHARC mają dodatkowe układy rejestrów do wykonywania wielu z tych funkcji.

Układy przetwarzania matematycznego można podzielić na trzy sekcje – **układ mnożący**, **jednostkę arytmetyczno-logiczną** (**ALU** – *Arithmetic Logic Unit*) oraz **szybki rejestr przesuwający** (typu *barrel shifter*). Układ mnożący pobiera wartości z dwóch rejestrów, wymnaża je i umieszcza wynik w jeszcze innym rejestrze. Jednostka arytmetyczno-logiczna



Rys. 28.5. Typowa architektura procesora DSP. Cyfrowe procesory sygnałowe są zaprojektowane do równoległego wykonywania zadań. Przedstawiono uproszczony schemat cyfrowego procesora sygnałowego SHARC firmy Analog Devices. Porównajcie tę architekturę z zadaniami niezbędnymi do realizacji filtru SOI podanymi w tablicy 28.1. Wszystkie zadania objęte pętlą mogą być wykonane w jednym okresie zegarowym

wykonuje dodawanie, odejmowanie, wyznaczanie wartości bezwzględnej, operacje logiczne (AND, OR, XOR, NOT), konwersję między formatami stało- i zmiennoprzecinkowym i podobne funkcje. W rejestrze przesuwającym są wykonywane elementarne operacje binarne, takie jak przesuwanie, obracanie, wydzielanie i przenoszenie segmentów itd. Stwarzającą wielkie możliwości cechą rodziny procesorów SHARC jest równoległy dostęp do układu mnożącego i do ALU. W ciągu jednego okresu zegarowego dane z rejestrów 0–7 mogą być przekazane do układu mnożącego, dane z rejestrów 8–15 – do ALU i oba uzyskane wyniki przekazane z powrotem do jednego z 16 rejestrów.

Rozszerzeniem pierwotnej architektury SHARC jest w rodzinie ADSP-2116x SHARC możliwość pracy typu SIMD (pojedyncza instrukcja – wiele danych, *Single Instruction Multiple Data*), jak w ADSP-TS101. Mówiąc w skrócie, operacja SIMD daje możliwość realizacji w procesorze DSP tej samej instrukcji jednocześnie z dwoma zestawami danych.

O ile w rodzinie 2106x był tylko jeden element procesorowy, to 2116x oraz TS101 zawierają po dwa takie elementy – PE x i PE y w procesorze 2116x i po prostu X i Y w TS101. PE X działa w sposób konwencjonalny i gdy jest w trybie pracy SIMD, te same instrukcje są wykonywane na danych w PE x (w rejestrze plik od R0 do R15) *oraz* na danych w PE y (w rejestrze plik od S0 do S15). Dalszym wzbogaceniem procesora jest operacja MIMD – wiele instrukcji – wiele danych (*Multiple Instructions Multiple Data*). Umożliwia ona układom komputera wykonywanie różnych operacji na różnych fragmentach danych w tym samym okresie zegara. Ta operacja jest dostępna w procesorze sygnałowym ADSP-TS101 TigerSHARC firmy Analog Devices.

W architekturze DSP Blackfin™, o dużej szybkości działania i dużych możliwościach, bloki komputera są tak zaprojektowane, żeby zmaksymalizować liczbę operacji matematycznych, które mogą być wykonane w jednym okresie zegara. Sercem procesora o architekturze Blackfin jest jednostka arytmetyczna danych (*data arithmetic unit*) zawierająca dwa 16-bitowe akumulatory układu mnożącego (MAC – *Multiplier Accumulator*), dwie 40-bitowe jednostki arytmetyczno-logiczne (ALU), cztery 8-bitowe wizyjne ALU i pojedynczy 40-bitowy szybki rejestr przesuwający. W tej architekturze 8-, 16- i 32-bitowe słowa danych mogą być przetwarzane w prosty sposób z maksymalną wydajnością.

Porównując te różne techniki dochodzimy do ciekawego spostrzeżenia. *W prawie wszystkich technikach DSP jest wymagane wykonanie od 4 do 400 instrukcji (impulsów zegarowych w procesorach rodziny SHARC).* Co do drugiej generacji procesorów sygnałowych SHARC ADSP2116x pracujących z częstotliwością 100 MHz możemy od razu wyciągnąć wniosek, że jego przepływność danych będzie od 250k do 25M próbek na sekundę, zależnie do stopnia skomplikowania algorytmu. W procesorze ADSP-TS101, 250 MHz przepływność zaś będzie między 625k i 62,5M próbek na sekundę, też zależnie od skomplikowania algorytmu. W tych wartościach przepływności nie wzięto pod uwagę korzyści wynikających z zastosowania trybu SIMD skracającego czas przetwarzania o połowę, podwajając przepływność danych.

DSP stałoprzecinkowe a DSP zmiennoprzecinkowe

Cyfrowe procesory sygnałowe można podzielić na dwie kategorie: **stałoprzecinkowe** i **zmiennoprzecinkowe**. Taki podział dotyczy formatu stosowanego do przechowywania liczb w urządzeniu i operowania nimi. W stałoprzecinkowych DSP każda liczba jest zwykle reprezentowana przez co najmniej 16 bitów, chociaż bywają też stosowane inne długości reprezentowanych liczb. Na przykład firma Motorola produkuje rodzinę 24-bitowych stałoprzecinkowych procesorów sygnałowych. Istnieją cztery często stosowane sposoby reprezentowania liczb przez tych $2^{16} = 65\,536$ możliwych ustawień. W przypadku reprezentowania **liczb całkowitych bez znaku** zapisana liczba może przyjmować wartości od 0 do 65 535. W metodzie reprezentowania **liczby całkowitej ze znakiem** stosuje się zapis uzupełnienia do dwóch, aby uzyskać zakres obejmujący także liczby ujemne od –32 768 do 32 767. Przy

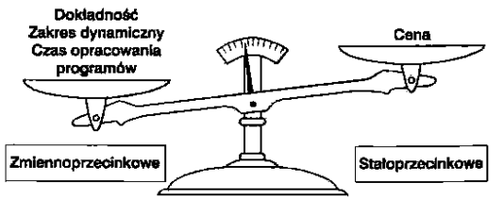
zapisie **ułamka bez znaku** 65 536 poziomów jest rozmieszczonych równomiernie między 0 a 1. Wreszcie, format **ułamka ze znakiem** umożliwia reprezentowanie także liczb ujemnych, równo rozmieszczonych między –1 a 1.

Natomiast w zmiennoprzecinkowych DSP do reprezentowania każdej wartości stosuje się typowo co najmniej 32 bity. To daje znacznie więcej możliwych kombinacji niż dla formatu stałoprzecinkowego, a ściśle $2^{32} = 4\ 294\ 967\ 296$. Główną cechą zapisu zmiennoprzecinkowego jest *nierównomierne* rozmieszczenie reprezentowanych liczb. W najczęściej stosowanym formacie (według normy ANSI/IEEE Std. 754-1985) największa i najmniejsza liczba to $\pm 3{,}4 \times 10^{38}$ oraz $\pm 1{,}2 \times 10^{-38}$. Reprezentowane wartości są nierównomiernie rozmieszczone między tymi dwiema wartościami skrajnymi tak, że odstęp między dwiema dowolnymi liczbami jest ok. dziesięć milionów razy mniejszy niż wartość tych liczb. Zapis zmiennoprzecinkowy omówiono bardziej szczegółowo w rozdziale 4.

Wszystkie zmiennoprzecinkowe procesory sygnałowe mogą też operować liczbami stałoprzecinkowymi, co jest konieczne do implementacji liczników, pętli oraz dla sygnałów przychodzących z przetworników a/c i wychodzących do przetworników c/a. Nie znaczy to jednak, że matematyczne operacje stałoprzecinkowe mogą być wykonywane równie szybko, jak zmiennoprzecinkowe; to zależy od architektury wewnętrznej. Na przykład procesory sygnałowe SHARC są zoptymalizowane tak dla operacji zmiennoprzecinkowych, jak i stałoprzecinkowych i wykonują je z jednakową wydajnością. Dlatego procesory SHARC są często określane raczej jako „32-bitowe DSP", a nie po prostu jako „zmiennoprzecinkowe".

Na rys. 28.6 przedstawiono porównanie podstawowych zalet procesorów sygnałowych stało- i zmiennoprzecinkowych. W rozdziale 3 podkreśliliśmy, że arytmetyka stałoprzecinkowa jest znacznie szybsza niż zmiennoprzecinkowa w komputerach ogólnego przeznaczenia. Jednak w DSP szybkość jest mniej więcej jednakowa, z powodu rozwiązań układowych silnie zoptymalizowanych dla operacji matematycznych. Wewnętrzna architektura DSP zmiennoprzecinkowych jest bardziej skomplikowana niż stałoprzecinkowych. Wszystkie rejestry i magistrale danych muszą być 32-bitowe, a nie tylko 16-bitowe; układ mnożący i jednostka arytmetyczno-logiczna powinny móc szybko wykonywać zmiennoprzecinkowe operacje arytmetyczne, zestaw instrukcji musi być większy (aby mogły operować liczbami zarówno zmienno-, jak i stałoprzecinkowymi) itd. Obliczenia zmiennoprzecinkowe (32 bity) mają lepszą dokładność i większy zakres dynamiczny niż stałoprzecinkowe (16 bitów). Ponadto, czas opracowania programów zmiennoprzecinkowych jest często krótszy, gdyż programista – ogólnie biorąc – nie musi się martwić o takie sprawy, jak przepełnienie, wyjście poza dolną granicę i błąd zaokrąglenia.

Z drugiej strony, stałoprzecinkowe procesory DSP są tradycyjnie tańsze od urządzeń zmiennoprzecinkowych. W elektronice ceny zmieniają się bardzo szybko; informacje w cennikach często dezaktualizują się już w chwili ich wydrukowania. Niemniej jednak koszt jest czynnikiem istotnym dla zrozumienia rozwoju cyfrowych procesorów sygnałowych i trzeba mieć chociaż ogólną orientację w tej kwestii. Kiedy kończono pisanie tej książki w 2002 roku, cena stałoprzecinkowych procesorów DSP mieściła się w granicach od 5 do 100 USD, zmiennoprzecinkowych zaś



Rys. 28.6. DSP stałoprzecinkowe a DSP zmiennoprzecinkowe. Procesory sygnałowe stałoprzecinkowe są na ogół tańsze, a zmiennoprzecinkowe mają lepszą dokładność, większy zakres dynamiczny i krótszy czas opracowania programu

– od 10 do 300 USD. Różnica w cenie jest związana z różnym stopniem złożoności obu rodzajów procesorów. Jeśli chcecie znać *obecne* ceny, to musicie zajrzeć do *aktualnych* cenników.

Zajmijmy się teraz *działaniem* procesorów; tym, co może wykonać zmiennoprzecinkowy system 32-bitowy, a czego nie może stałoprzecinkowy 16-bitowy. Ta kwestia wiąże się ze **stosunkiem sygnału do szumu**. Załóżmy, że zapisujemy liczbę w 32-bitowym formacie zmiennoprzecinkowym. Jak poprzednio wspomniano, odstęp między tą liczbą a najbliższą sąsiednią wynosi ok. jedną dziesięciomilionową wartości tej liczby. Aby można było zapisać tę liczbę, musi ona zostać zaokrąglona w górę lub w dół o maksimum połowę tego odstępu. Inaczej mówiąc, za każdym razem gdy zapisujemy liczbę w formacie zmiennoprzecinkowym, dodajemy do sygnału pewien *szum*.

To samo się dzieje, gdy liczba jest zapisywana w 16-bitowym formacie stałoprzecinkowym, z tą wszakże różnicą, że dodany szum jest znacznie gorszy. Przyczyną są znacznie większe odstępy między sąsiednimi liczbami. Załóżmy na przykład, że zapisujemy liczbę 10 000 jako liczbę całkowitą ze znakiem (w zakresie od –32 768 do 32 767). Odstęp między liczbami wynosi jedną dziesięciotysięczną część zapisywanej liczby. Jeśli chcemy zapisać liczbę 1000, to odstęp między liczbami jest równy tylko jednej tysięcznej wartości liczby.

Szum w sygnale jest zazwyczaj wyrażany przez *odchylenie standardowe*, o czym szczegółowo była mowa w rozdziale 2. W tych rozważaniach ważny jest fakt, że odchylenie standardowe tego **szumu kwantyzacji** jest równe ok. jednej trzeciej rozmiaru odstępu między sąsiednimi liczbami. Z tego wynika, że stosunek sygnału do szumu w przypadku zapisu liczby zmiennoprzecinkowej wynosi ok. 30 milionów do 1, a dla liczby stałoprzecinkowej tylko ok. 10 000 do 1. A zatem przy zapisie zmiennoprzecinkowym szum kwantyzacji jest w przybliżeniu 3000 razy mniejszy niż przy zapisie stałoprzecinkowym.

I tu dochodzimy do ważnej różnicy między procesorami DSP i mikroprocesorami tradycyjnymi. Przyjmijmy, że realizujemy filtr SOI przy użyciu zapisu stałoprzecinkowego. Aby to wykonać, prowadzimy pętlę przez każdy ze współczynników, przemnażamy współczynnik przez odpowiednią próbkę sygnału wejściowego i uzyskany iloczyn dodajemy do akumulatora. I tu jest problem. W tradycyjnym mikroprocesorze ten akumulator jest po prostu jeszcze jedną 16-bitową zmienną stałoprzecinkową. Aby uniknąć przepełnienia, trzeba zmieniać skalę dodawanych wartości i w każdym kroku następuje wtedy dodawanie szumu kwantyzacji. W najgorszym przypadku ten szum kwantyzacji po prostu się sumuje, znacznie obniżając stosunek sygnału do szumu w systemie. Na przykład w filtrze SOI o 500 współczynnikach szum w każdej próbce wyjściowej może być 500 razy większy od szumu w każdej próbce wejściowej. Wtedy stosunek sygnału do szumu zmniejszy się z wartości *dziesięć tysięcy do jednego* do koszmarnej wartości *dwadzieścia do jednego*. Chociaż jest to skrajny przypadek, ilustruje on najważniejszą kwestię – jest bardzo, rzeczywiście bardzo niedobrze, gdy na każdej próbce przeprowadza się wiele operacji. Więcej szczegółów znajdziecie w rozdziale 3.

W procesorach DSP przezwyciężono tę trudność, stosując **akumulator o rozszerzonej dokładności**. Jest to specjalny rejestr mający 2–3 razy więcej bitów niż w innych lokalizacjach pamięci. Na przykład w 16-bitowym DSP taki akumulator może mieć od 32 do 40 bitów, a w procesorach sygnałowych SHARC zawiera on 80 bitów dla operacji stałoprzecinkowych. Ten rozszerzony zakres eliminuje szum zaokrąglania, gdy akumulacja jest w toku. Jedyny odczuwalny błąd zaokrąglenia powstaje przy przeskalowaniu zawartości akumulatora i zapisaniu jej w pamięci 16-bitowej. Ta metoda działa bardzo dobrze, choć ogranicza sposób realizacji niektórych algorytmów. Natomiast przy stosowaniu formatu zmiennoprzecinkowego błąd kwantyzacji jest tak mały, że ta technika zwykle nie jest konieczna.

Inną zaletą systemów zmiennoprzecinkowych, prócz mniejszego szumu kwantyzacji, jest łatwiejsze opracowywanie algorytmów dla tych systemów. Podstawą większości metod DSP są powtarzane mnożenia i dodawania. W systemach stałoprzecinkowych po każdej operacji trzeba uwzględniać możliwość powstania przepełnienia lub wyjścia poza dolną granicę zakresu. Programista musi stale śledzić wartości liczb, akumulację błędów kwantyzacji, a także wie-

dzieć, jakie przeskalowanie jest konieczne. Te kwestie nie występują w systemach zmiennoprzecinkowych, liczby same „troszczą się o siebie" (poza niektórymi rzadkimi przypadkami).

Aby umożliwić lepsze zrozumienie tych problemów, na rys. 28.7 pokazano tablicę zaczerpniętą z podręcznika użytkownika procesorów SHARC. Podano w niej sposoby przeprowadzania mnożenia zarówno w formacie stało-, jak i zmiennoprzecinkowym. Najpierw spójrzmy, jak są mnożone liczby zmiennoprzecinkowe; jest tylko jeden sposób: Fn = Fx * Fy, gdzie Fn, Fx i Fy są dowolnymi z 16 rejestrów danych. Jeszcze prościej być nie może. Dla porównania, przyjrzyjmy się wszystkim możliwym instrukcjom mnożenia liczb stałoprzecinkowych. Jest tam wiele opcji koniecznych do skutecznego rozwiązania problemów zaokrąglania, skalowania i formatu.

Na rys. 28.7 Rn, Rx, Ry dotyczą dowolnych z 16 rejestrów, a MRF i MRB są akumulatorami 80-bitowymi. Linie pionowe wskazują *opcje*. Na przykład lewy górny zapis w tablicy oznacza, że wszystkie następujące instrukcje są ważne: Rn = Rx * Ry, MRF = Rx * Ry, i MRB = Rx * Ry. Innymi słowy, wartości dwóch dowolnych rejestrów mogą być przemnożone, a wynik umieszczony w innym rejestrze lub w jednym z akumulatorów o rozszerzonej dokładności. Z tablicy wynika też, że liczby mogą być ze znakiem (S – *Signed*) lub bez znaku (U – *Unsigned*) oraz ułamkowe (F – *Fractional*) lub całkowite (I – *Integer*). Opcje RND i SAT są sposobami kontroli zaokrąglania i przepełnienia rejestru.

Tablica zawiera jeszcze inne szczegóły i opcje, lecz nie są one istotne w naszych obecnych rozważaniach. Ważne jest to, że programista systemu stałoprzecinkowego musi rozumieć *kilkanaście* sposobów realizacji tak podstawowego zadania, jakim jest mnożenie. Natomiast programista systemów zmiennoprzecinkowych może cały swój czas poświęcić algorytmowi

Stałoprzecinkowe	Zmiennoprzecinkowe
Rn / MRF / MRB = Rx * Ry (S/U, S/U, F/I/FR)	Fn = Fx * Fy
Rn / Rn / MRF / MRB = MRF / MRB + Rx * Ry (S/U, S/U, F/I/FR)	
Rn / Rn / MRF / MRB = MRF / MRB − Rx * Ry (S/U, S/U, F/I/FR)	
Rn / Rn / MRF / MRB = SAT MRF / MRB (SI) (UI) (SF) (UF)	
Rn / Rn / MRF / MRB = RND MRF / MRB (SF) (UF)	
MRF / MRB = 0	
MRxF / MRxB = Rn	
Rn = MRxF / MRxB	

Rys. 28.7. Instrukcje stałoprzecinkowe a zmiennoprzecinkowe. Są to instrukcje mnożenia stosowane w cyfrowych procesorach sygnałowych SHARC. O ile w zapisie zmiennoprzecinkowym jest potrzebna tylko jedna instrukcja, to w zapisie stałoprzecinkowym jest niezbędnych wiele opcji. Te opcje wyjaśniono w tekście

W jaki sposób dokonacie wyboru procesora stało- lub zmiennoprzecinkowego, znając już ich zalety i wady? Trzeba tu uwzględnić kilka kwestii. Najpierw zobaczcie, ile bitów zastosowano w przetwornikach a/c i c/a. W wielu zastosowaniach 12–14 bitów na próbkę jest rozgraniczeniem używania procesora stało- lub zmiennoprzecinkowego. Na przykład w technice TV i dla innych sygnałów wizyjnych stosuje się, typowo, przetworniki a/c i c/a 8-bitowe i dokładność operacji stałoprzecinkowych jest wystarczająca. Natomiast w profesjonalnych systemach fonicznych próbkowanie może być nawet 20- lub 24-bitowe i prawie na pewno jest konieczne użycie procesora zmiennoprzecinkowego, aby móc uzyskać duży zakres dynamiczny.

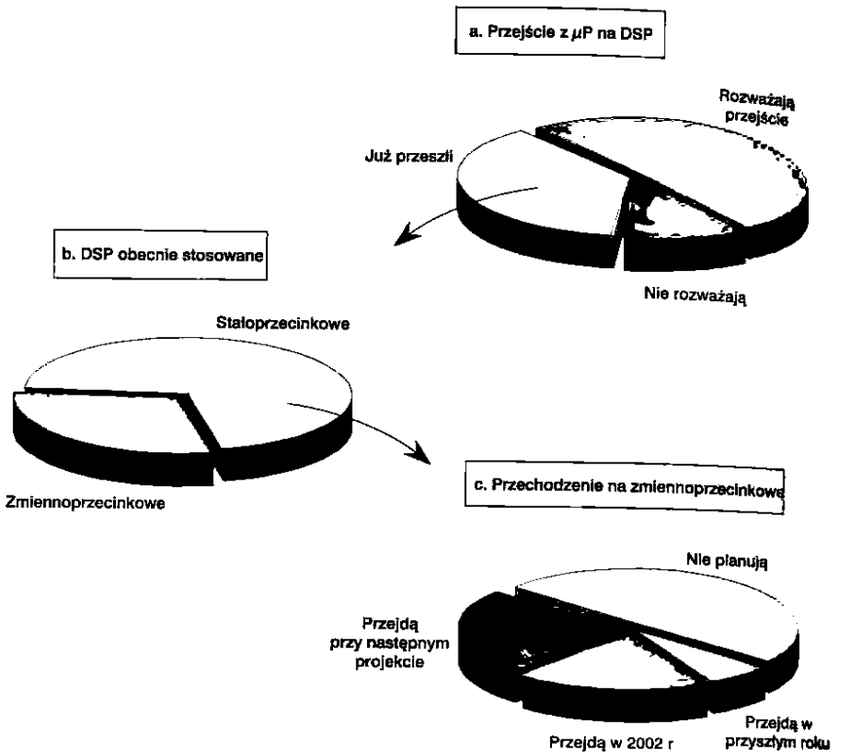
Następną kwestią do rozważenia jest złożoność algorytmu, który ma być zastosowany. Jeśli jest dość prosty, to pomyślcie o formacie stałoprzecinkowym; a jeśli bardziej skomplikowany – to o zmiennoprzecinkowym. Na przykład filtracja SOI i inne operacje w dziedzinie czasu wymagają tylko kilkudziesięciu wierszy kodu programu i dlatego są odpowiednie do procesorów stałoprzecinkowych. Algorytmy w dziedzinie częstotliwości zaś, takie jak analiza widmowa i splot FFT, zawierają wiele szczegółów i mogą być znacznie trudniejsze do zaprogramowania. Mogą wprawdzie być napisane w formacie stałoprzecinkowym, lecz zastosowanie formatu zmiennoprzecinkowego znacznie skróci czas opracowania programu.

W końcu pomyślcie też o pieniądzach – jak istotny jest koszt produktu, a jak ważny koszt opracowania programu? Przy wyborze zapisu stałoprzecinkowego koszt produktu będzie zredukowany, lecz koszt opracowania będzie przypuszczalnie większy z powodu trudniejszych algorytmów. Odwrotnie przy zapisie zmiennoprzecinkowym – opracowanie będzie szybsze i tańsze, lecz produkt końcowy – droższy.

Kilka głównych trendów rozwoju procesorów DSP zaprezentowano na rys. 28.8. Na rys. 28.8 a) przedstawiono wpływ, jaki cyfrowe procesory sygnałowe wywarły na rynek systemów *wbudowanych* (*embedded*). Są to takie zastosowania, w których mikroprocesor jest wykorzystywany do bezpośredniego sterowania jakimś większym systemem, np. telefonem komórkowym, kuchenką mikrofalową lub samochodową tablicą rozdzielczą. W odniesieniu do takich układów jest często stosowana nazwa „mikrokontroler" (lub po polsku bardziej prawidłowo – mikrosterownik – przyp. tłum.), w odróżnieniu od mikroprocesorów stosowanych w komputerach osobistych. Jak wynika z rys. 28.8 a), ok. 38% projektantów systemów wbudowanych już zaczęło stosować DSP, a rozważa to dalsze 49%. Duża wydajność i możliwości obliczeniowe procesorów DSP powodują, że są często wybierane jako najlepsze rozwiązanie w projektach układów wbudowanych.

Jak pokazano na rys. 28.8 b), inżynierów stosujących obecnie procesory sygnałowe stałoprzecinkowe jest dwa razy więcej niż tych stosujących procesory zmiennoprzecinkowe. Zależy to jednak w dużym stopniu od rodzaju aplikacji. Procesory stałoprzecinkowe są bardziej popularne w rynkowym sprzęcie powszechnego użytku, gdzie koszt elektroniki musi być bardzo niski. Dobrym przykładem są tu telefony komórkowe. Gdy producent musi konkurować z innymi, aby sprzedać miliony sztuk swojego wyrobu, to kilkudolarowa różnica kosztu może decydować o sukcesie lub porażce. Procesory zmiennoprzecinkowe są zaś częściej używane wówczas, gdy jest wymagane lepsze działanie, a koszt nie jest istotny. Przyjmijcie na przykład, że projektujecie medyczny system obrazowy, taki jak tomograf komputerowy. Sprzedaje się tylko kilkaset sztuk danego modelu takiego urządzenia, w cenie po kilkaset tysięcy dolarów. W takiej aplikacji koszt procesora DSP jest nieznaczący, ale jego dobre działanie ma decydujące znaczenie. Najszybciej rozwijającym się sektorem rynku DSP są jednak procesory zmiennoprzecinkowe, mimo że obecnie w użyciu jest wielka liczb procesorów stałoprzecinkowych. Jak pokazano na rys. 28.8 c), więcej niż połowa inżynierów stosujących układy 16-bitowe zamierza przejść w najbliższej przyszłości na systemy zmiennoprzecinkowe.

Przed odejściem od tego tematu powinniśmy raz jeszcze podkreślić, że w procesorach zmienno- i stałoprzecinkowych stosuje się na ogół – odpowiednio – 32 bity lub 16 bitów – *lecz nie zawsze*. Na przykład w procesorach rodziny SHARC liczby są reprezentowane w 32-bitowym zapisie stałoprzecinkowym, a więc w trybie przyjętym w cyfrowych systemach audio.



Rys. 28.8. Główne trendy rozwoju procesorów DSP. (Opisano tu wprawdzie sytuację w roku 2000, ale podane trendy są nadal aktualne – przyp. tłum.) Na rys. a) pokazano, że ok. 38% projektantów systemów wbudowanych przeszło już z konwencjonalnych mikroprocesorów na DSP, a dalsze 49% rozważa taką zmianę. Z rys. b) wynika, że inżynierów stosujących obecnie procesory sygnałowe stałoprzecinkowe jest około dwa razy więcej niż tych stosujących procesory zmiennoprzecinkowe. Jest to związane głównie z wyrobami powszechnego użytku – np. telefonami komórkowymi – w których koszt elektroniki musi być bardzo niski. Jednak, jak pokazano na rys. c), najszybciej rozwijającym się sektorem rynku DSP są procesory zmiennoprzecinkowe; ponad połowa inżynierów stosujących obecnie układy 16-bitowe planuje przejść na procesory DSP zmiennoprzecinkowe

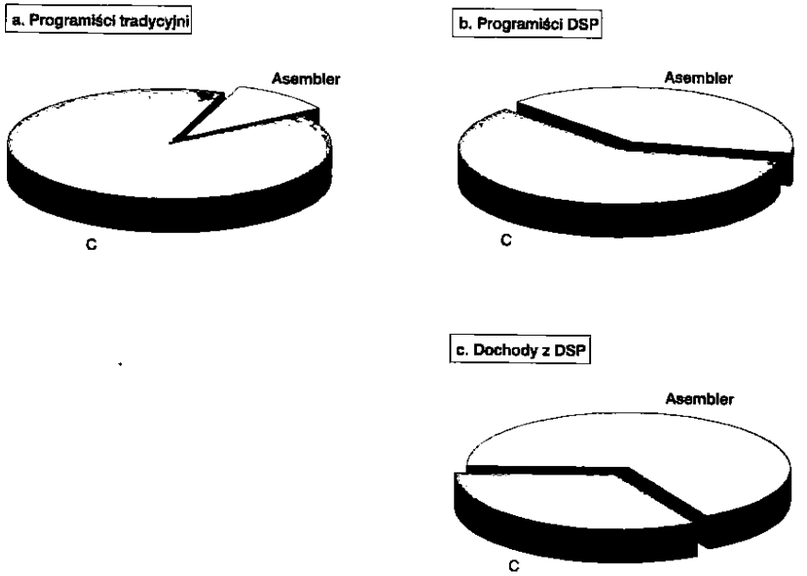
To daje 2^{32} poziomów kwantyzacji rozmieszczonych równomiernie w stosunkowo małym zakresie, powiedzmy od –1 do 1. Przy zapisie zmiennoprzecinkowym zaś 2^{32} poziomów kwantyzacji jest rozmieszczonych logarytmicznie w ogromnym zakresie, typowo $\pm 3{,}4 \times 10^{38}$. To daje procesorom stałoprzecinkowym lepszą *dokładność*, a więc mniejszy błąd kwantyzacji w każdej próbce. Jednak 32-bitowe procesory zmiennoprzecinkowe mają większy *zakres dynamiczny*, co znaczy, że większa jest różnica między najmniejszą i największą liczbą, które mogą być reprezentowane.

Język C a język asemblera

Procesory DSP są programowane w tych samych językach, które są używane w innych zastosowaniach w nauce i technice, zwykle w języku *asemblera* lub w *C*. Programy napisane w asemblerze mogą być wykonywane szybciej, a programy w języku C są łatwiejsze do opracowania i utrzymywania. W tradycyjnych zastosowaniach, takich jak programy wykonywane na dużych komputerach i na komputerach osobistych, prawie zawsze wybiera się język C. Jeśli w ogóle stosuje się asembler, to jego wykorzystanie jest ograniczone do krótkich podprogramów, które muszą być wykonywane ze skrajnie dużą szybkością. Przedstawiono to graficznie na rys. 28.9 a). Na jednego programistę pracującego tradycyjnie w asemblerze przypada około *dziesięciu* stosujących język C.

Programy DSP różnią się jednak pod dwoma ważnymi względami od tradycyjnych zadań programistycznych. Po pierwsze, programy DSP są zwykle znacznie krótsze, powiedzmy sto wierszy w stosunku do dziesięciu tysięcy. Po drugie, w wielu zastosowaniach DSP szybkość wykonywania ma znaczenie decydujące. Bo przecież wielu użytkowników stosuje DSP właśnie ze względu na bardzo dużą szybkość w tej technice. Dwa wymienione czynniki motywują wielu inżynierów-programistów do przechodzenia z języka C na asembler przy programowaniu cyfrowych procesorów sygnałowych. Ten fakt zilustrowano na rys 28.9 b) – prawie tyle samo programistów DSP stosuje asembler, co DSP.

Ta sytuacja potwierdza się na rys. 28.9 c), na którym przedstawiono dochód przynoszony przez produkty DSP. Na każdy dolar dochodu z produktów DSP programowanych w C przypada dwa dolary z produktów programowanych w asemblerze. Przyczyna tego jest prosta: pieniądze się robi, wygrywając w konkurencji. Jeśli uwzględnić takie właściwości, jak szybkość wykonywania i koszt wytwarzania, asembler prawie zawsze wykazuje przewagę nad językiem C. Na przykład program napisany w C zwykle wymaga pamięci większej niż asembler, co podraża część układową systemu. Jednak rynek DSP ciągle się zmienia. Na wzrost rynku producenci odpowiedzą opracowaniem procesorów DSP *zoptymalizowanych* do programowania w języku C. Język C jest, na przykład, znacznie bardziej skuteczny, gdy istnieje duży zestaw rejestrów ogólnego przeznaczenia i zunifikowana przestrzeń pamięciowa.



Rys. 28.9. Porównanie stosowania języka C i języka asemblera. Jak pokazano na rys. a), tylko 10% tradycyjnych programistów (takich, którzy pracują na komputerach dużych i osobistych) stosuje asembler. Jednak, jak wynika z rys. b), asembler jest popularniejszy w cyfrowych procesorach sygnałowych. Przyczyna leży w tym, że programy DSP muszą być wykonywane możliwie jak najszybciej i są zwykle dość krótkie. Na rys. c) pokazano, że asemblery są jeszcze częściej używane w produktach dających duże dochody

Te przyszłe udoskonalenia zminimalizują różnicę czasów wykonania programów w języku C i asemblerze i umożliwią wprowadzenie języka C do większej liczby zastosowań.

W celu lepszego zrozumienia decyzji wyboru języka C lub asemblera przyjrzyjmy się typowemu zadaniu DSP zaprogramowanemu w każdym z tych języków. Przykładem będzie obliczenie *iloczynu skalarnego* dwóch matryc *x*[ ] i *y*[ ]. Jest to prosta operacja matematyczna – mnożymy każdy współczynnik w jednej matrycy przez odpowiadający mu współczynnik w drugiej i sumujemy uzyskane iloczyny, tj. $x[0] \times y[0] + x[1] \times y[1] + x[2] \times y[2] + \ldots$ To wyrażenie powinno się wydawać bardzo znajome; jest to bowiem podstawowa operacja w filtrach SOI. A więc każda próbka w sygnale wyjściowym jest wyznaczana przez wymnożenie zarejestrowanych próbek sygnału wejściowego (w jednej matrycy) przez współczynniki filtru (w drugiej) i sumowanie iloczynów.

W tablicy 28.2 podano, jak iloczyn skalarny jest obliczany w języku C. W wierszach 001–004 definiujemy długości dwóch matryc, *x*[ ] i *y*[ ] równe 20 elementom. Definiujemy też *result* (wynik) – zmienną, w której jest przechowywany obliczony iloczyn skalarny, po zakończeniu programu. Wiersz 011 steruje pętlą 20 konieczną do obliczeń, wykorzystując zmienną *n* jako licznik pętli. Jedyną instrukcją w pętli jest wiersz 012, w którym następuje mnożenie odpowiednich współczynników z obu matryc i dodanie iloczynu do zmiennej akumulatora. (Jeśli nie znacie języka C, to warto się dowiedzieć, że instrukcja: $s += x[n] * y[n]$ znaczy to samo, co $s = s + x[n] * y[n]$). Po pętli, wartość akumulatora jest przekazywana do zmiennej wyjściowej *result* w wierszu 013.

Podstawową korzyścią ze stosowania języków wysokiego poziomu (jak C, Fortran lub Basic) jest uniknięcie konieczności rozumienia przez programistę architektury zastosowanego mikroprocesora. Znajomość tej architektury jest pozostawiona kompilatorowi. Natomiast w podanym krótkim programie w języku C zastosowano kilka zmiennych: *n*, *s*, *result* oraz matryce: *x*[ ] i *y*[ ]. Wszystkim tym zmiennym musi być przyporządkowana jakaś „rezydencja" w układach procesora, aby można było śledzić ich wartości. W zależności od mikroprocesora pamięciami dla tych zmiennych mogą być rejestry danych ogólnego przeznaczenia, lokacje w pamięci głównej lub specjalne rejestry przeznaczone do poszczególnych funkcji. Jednak osoba pisząca program w języku wysokiego poziomu wie niewiele lub nic o takiej organizacji pamięci; to zadanie przydzielono programiście, który napisał program kompilujący. Problem w tym, że ci dwaj ludzie nigdy się nie spotkali, komunikują się tylko poprzez zestaw wcześniej ustalonych reguł. Języki wysokiego poziomu są łatwiejsze od języka asemblera, gdyż stosując je, zlecamy komu innemu połowę pracy. Języki wysokiego poziomu są jednak mniej efektywne, gdyż nie jesteśmy całkiem pewni, jak działa ta część pracy, która została zlecona.

Tablica 28.2. Iloczyn skalarny w języku C. Program oblicza iloczyn skalarny dwóch matryc x[ ] i y[ ] i umieszcza wynik w zmiennej result

```
001          #defineLEN20
002             float dm x[LEN];
003             float pm y [LEN];
004             float result;
005
006          main()
007
008          {
009          int n;
010          float s;
011          for (n=0;n<LEN;n++)
012             s+=x[n]*y[n];
013          result = s
014          }
```


W tablicy 28.3 podano, dla porównania, program obliczania iloczynu skalarnego napisany w języku asemblera dla DSP rodziny SHARC. Języki asemblera dla cyfrowych procesorów sygnałowych firmy Analog Devices (zarówno 16-bitowych stałoprzecinkowych, jak i 32-bitowych SHARC) są znane ze swojej prostej, algebro podobnej składni. Omówimy tu tylko ogólne ich działanie, nie wchodząc we wszystkie szczegóły. Zauważcie, że w tym programie *wszystko* odnosi się do układów procesora; nie ma tu żadnych zmiennych abstrakcyjnych, a tylko rejestry danych i lokacje pamięci.

Każdy średnik w programie reprezentuje okres zegara. Matryce *x*[ ] i *y*[ ] są przechowywane w buforach cyklicznych w pamięci głównej. W wierszach 001 i 002 rejestry i4 oraz i12 wskazują początkowe lokacje tych matryc. Następnie wykonujemy 20 cykli pętli sterowanych przez wiersz 004. W formacie tej instrukcji wykorzystano dostępną w procesorach SHARC możliwość sterowania pętlą przez zerowanie rejestru (*zero-overhead looping*). Wszystkie zmienne potrzebne do sterowania pętlą są więc przechowywane w układach specjalnych rejestrów działających równolegle z innymi operacjami odbywającymi się wewnątrz procesora. W tym przypadku rejestr *lcntr* (licznik pętli – *loop counter*) jest załadowany początkową wartością 20 zmniejszaną o 1 po każdym wykonaniu pętli. Pętla się kończy, gdy *lcntr* osiąga wartość zero (wskazaną przez instrukcję *lce – loop counter expired*, licznik pętli wyzerowany). Pętla obejmuje wiersze od 004 do 008, zgodnie ze sterowaniem przez instrukcję (pc,4). A więc pętla kończy się cztery wiersze za bieżącą wartością licznika programu.

Wewnątrz pętli w wierszu 005 następuje ładowanie wartości z *x*[ ] do rejestru danych f2, a w wierszu 006 – wartości z *y*[ ] do rejestru danych f4. Symbole „dm" i „pm" wskazują, czy te wartości są pobierane przez magistralę pamięci danych („dm"), czy przez magistralę pamięci programu („pm"). Zmienne i4, m6, i12 oraz m14 są rejestrami w generatorach danych adresowych, które zarządzają buforami cyklicznymi przechowującymi *x*[ ] i *y*[ ]. Dwie wartości, w f2 i f4, są wymnażane w wierszu 007, a iloczyn jest wpisywany do rejestru danych f8. W wierszu 008 iloczyn z f8 jest dodawany do akumulatora, czyli rejestru danych f12. Po zakończeniu pętli zawartość akumulatora z f12 jest przekazywana do pamięci.

Ten program prawidłowo oblicza iloczyn skalarny, lecz nie wykorzystuje tej właściwości procesorów SHARC, jaką jest ich struktura w wysokim stopniu równoległa. W tablicy 28.4 podano ten sam program, lecz napisany w formie bardzo zoptymalizowanej, z wieloma operacjami wykonywanymi równolegle. Najpierw zauważcie, że w wierszu 007 określono wykonywanie tylko 18 pętli, a nie 20. Zwróćcie też uwagę, że ta pętla zawiera tylko jeden wiersz (008), w którym jednak zawarto wiele operacji. Chodzi o to, aby uczynić pętlę jak najbardziej efektywną; w tym przypadku jeden wiersz może być wykonywany podczas jednego okresu zegarowego. Aby móc to zrobić, powinniśmy niewielką część kodu programu przeznaczyć na „pierwsze załadowanie" rejestrów w pierwszej pętli (wiersze 004 i 005) i następną małą część kodu – na zakończenie ostatniej pętli (wiersze 010 i 011).

Tab. 28.3. Obliczanie iloczynu skalarnego w programie asemblera (niezoptymalizowanym). Program oblicza iloczyn skalarny dwóch matryc x [ ] i y[ ] i umieszcza wynik w zmiennej result. Jest to program asemblera dla DSP rodziny SHARC firmy Analog Devices. Szczegóły podano w tekście

```
001    i12 = _y;                                /* i12 wskazuje początek y[ ]*/
002    i4 = _x;                                 /* i4 wskazuje początek x[ ]*/
003
004    lcntr = 20, do (pc,4) until lce;         /* pętla przez 20 zapisów w matrycy*/
005    f2 = dm(i4,m6);                          /* ładuj wartość x[ ] do rejestru f2*/
006    f4 = pm(i12,m14);                        /* ładuj wartość y[ ] do rejestru f4*/
007    f8 = f2*f4;                              /* przemnóż obie te wartości, wpisz do f8*/
008    f12 = f8 + f12;                          /* dodaj ten iloczyn do akumulatora w f12*/
009
010    dm(_result) = f12;                       /*wpisz zawartość akumulatora do pamięci*/
```

Tablica 28.4. Obliczanie iloczynu skalarnego w programie asemblera (zoptymalizowanym). Jest to zoptymalizowana wersja programu z tablicy 28.2, zaprojektowana z wykorzystaniem równoległej, w wysokim stopniu, struktury procesorów SHARC

```
001     il2 = _y;                                   /* il2 wskazuje początek y[ ]*/
002     i4 = _x;                                    /* i4 wskazuje początek x[ ]*/
003
004     f2 = dm(i4,m6), f4 = pm(i12,m14)            /* pierwsze załadowanie rejestrów*/
005     f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);
006
007     lcntr = 18, do (pc,1) intil lce;            /* pętla główna o wysokiej efektywności*/
008     f12 = f8 + f12, f8 = f2*f4, f2 = dm(i4,m6), f4 = pm(i12,m14);
009
010     f12 = f8 + f12, f8 = f2*f4;                 /* zakończ ostatnią pętlę*/
011     f12 = f8 + f12;
012
013     dm(_result) = f12;                          /*wynik umieść w pamięci*/
```

Aby zrozumieć, jak to działa, przyjrzyjmy się wierszowi 008, jedynej instrukcji wewnątrz pętli. W tej pojedynczej instrukcji są równolegle wykonywane *cztery* operacje: 1) wartość *x*[ ] jest przekazywana z bufora cyklicznego w pamięci programu do f2, 2) wartość *y*[ ] jest przekazywana z bufora cyklicznego w pamięci programu do f4, 3) poprzednie wartości f2 i f4 są wymnażane i umieszczane w f8, oraz 4) poprzednia wartość w f8 jest dodawana do akumulatora w f12.

Na przykład przy wykonywaniu instrukcji z wiersza 008 po raz piąty, *x*[7] i *x*[8] są pobierane z pamięci i zapisywane w f2 i f4. W tym samym czasie *x*[6] i *y*[6] (które były w f2 i f4 na początku tego okresu) są wymnażane i umieszczane w f8. Ponadto, wartość *x*[5] × *y*[5] (która była w f8 na początku tego okresu) jest dodawana do wartości f12.

Porównajmy teraz liczbę okresów zegarowych wymaganych do wykonania programów niezoptymalizowanego i zoptymalizowanego. Pamiętajcie, że w programie bez optymalizacji jest 20 pętli z czterema działaniami wykonywanymi w każdej pętli. Ten program wymaga więc 80 okresów zegarowych do wykonania działań wewnątrz pętli oraz jeszcze dodatkowo 5 okresów zegarowych, co razem daje 85 okresów. Natomiast w programie zoptymalizowanym wykonuje się 18 pętli w ciągu 18 okresów zegarowych, lecz koniecznych jest jeszcze dodatkowo 11 okresów na pierwsze załadowanie rejestrów i zakończenie ostatniej pętli. Daje to całkowity czas wykonania równy 29 okresom zegarowym, a więc około trzy razy krótszy niż w prostym programie bez optymalizacji.

I tu pojawia się ważne pytanie. Jaka jest szybkość wykonania programu w języku C w porównaniu z programem asemblerowym? Czy – gdy program z tablicy 28.2 zostanie skompilowany – to uzyskany kod wykonywalny będzie podobny do naszych przykładów asemblera *efektywnego* lub *niefektywnego*? Można na to odpowiedzieć, że kompilator daje *efektywny* kod. Trzeba sobie jednak zdawać sprawę z tego, że iloczyn skalarny jest przykładem bardzo prostym. Dla kompilatora znacznie trudniejszym zadaniem jest wytworzenie zoptymalizowanego kodu, gdy program staje się bardziej skomplikowany, np. z wielokrotnymi pętlami zagnieżdżonymi i niespodziewanymi skokami do podprogramów. Jeśli robicie coś w sposób bezpośredni, to możecie oczekiwać, że kompilator dostarczy wam rozwiązanie prawie optymalne. Jeśli zaś robicie coś dziwacznego lub skomplikowanego, to spodziewajcie się, że program asemblerowy będzie wykonywany znacznie szybciej niż ten w języku C. W najgorszym przypadku może to być czynnik 2–3. Jak już wspomniano, efektywność języka C w porównaniu z asemblerem zależy w dużym stopniu od zastosowanego procesora sygnałowego. Przy stosowaniu języków wysokiego poziomu, jak C, procesory zmiennoprzecinkowe mogą ogólnie biorąc, być programowane bardziej efektywnie niż stałoprzecinkowe. Oczywiście do tego celu konieczne są właściwe narzędzia programowe, takie jak debugery z właściwościami profilowania, pomagające zrozumieć, jak długiego czasu wymaga wykonanie poszczególnych segmentów kodu.

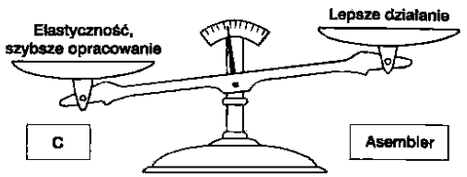
Jest jeszcze pewien sposób, w którym możecie wyzyskać najlepsze cechy programowania zarówno w języku C, jak i w asemblerze. Napiszcie program w C, a potem zastosujcie asembler do krytycznych fragmentów, które muszą być wykonywane bardzo szybko. Jest to jedna z przyczyn, dla której program C jest tak popularny w nauce i w technice. Działa jak język wysokiego poziomu, lecz daje też możliwość bezpośredniego operowania częścią układową, jeśli jest taka potrzeba. Nawet jeśli macie zamiar programować tylko w języku C, to przypuszczalnie będziecie potrzebować jakichś wiadomości na temat architektury procesora sygnałowego i zestawu instrukcji asemblera. Spójrzcie, na przykład, znowu na wiersze 002 i 003 w tablicy 28.2, w programie obliczania iloczynu skalarnego w języku C. Tam „dm" oznacza, że *x*[ ] ma być zapisane do pamięci danych, podczas gdy „pm" wskazuje, że *y*[ ] będzie umieszczone w pamięci programu. Chociaż program jest napisany w języku wysokiego poziomu, to jednak dla uzyskania najlepszego działania systemu jest niezbędna podstawowa wiedza o rozwiązaniach układowych procesora.

Który język jest najlepszy do *waszego* zastosowania? Wybór zależy od tego, co jest dla was ważniejsze. Jeśli potrzebujecie elastyczności oraz szybkiego opracowania programu, to wybierzcie język C. Natomiast jeśli chcecie osiągnąć możliwie jak najlepsze działanie systemu, to zastosujcie asembler. Przy tym wyborze nie unikniecie kompromisów, jak to zilustrowano na rys. 28.10. Oto kilka kwestii, które powinniście uwzględnić.

- Jak skomplikowany jest program? Jeśli jest długi i skomplikowany, to będziecie przypuszczalnie chcieli zastosować język C. Jeśli zaś jest krótki i prosty, to asembler może się okazać dobrym wyborem.
- Czy nalegacie na uzyskanie maksymalnej szybkości procesora sygnałowego. Jeśli tak, to asembler wyciśnie z procesora „ostatnie soki". W zastosowaniach o mniejszych wymaganiach asembler nie ma szczególnych zalet i możecie rozważyć zastosowanie C.
- Ilu programistów będzie pracować razem? Jeśli projekt jest na tyle duży, że wymaga więcej niż jednego programisty, to skłaniajcie się raczej ku językowi C i zastosujcie wbudowany asembler tylko w segmentach krytycznych pod względem szybkości wykonania.
- Co jest ważniejsze – *koszt produktu* czy *koszt opracowania*? Jeśli koszt produktu, wybierzcie asembler, a jeśli opracowania – wybierzcie C.
- Jakie jest wasze przygotowanie w dziedzinie programowania? Jeśli macie doświadczenie w asemblerach (na innych mikroprocesorach), to wybierzcie asembler do waszego DSP. Jeśli wasze poprzednie prace były w języku C, zdecydujcie się na C.
- Zastosowanie którego programu sugeruje producent cyfrowego procesora sygnałowego?

Ten ostatni punkt jest bardzo ważny. Przypuśćmy, że pytacie producenta procesora DSP, jaki język macie zastosować, a tam wam mówią: *„Możecie stosować albo język C, albo asembler, ale my rekomendujemy C"*. Powinniście skorzystać z tej rady. To, co oni naprawdę powiedzieli, brzmi: *„Nasz procesor jest tak trudny do programowania w asemblerze, że będziecie potrzebować 6 miesięcy ćwiczeń, aby móc napisać program"*. Z drugiej jednak strony, niektóre procesory sygnałowe są łatwe do programowania w asemblerze. Do tej kategorii zaliczają się na przykład układy firmy Analog Devices. Zapytajcie zresztą o to ich inżynierów, oni są z tego bardzo dumni.

Jednym z najlepszych sposobów podejmowania decyzji o układach i oprogramowaniu DSP jest konsultacja z inżynierami, którzy już je stosowali. Spytajcie producenta o referencje firm stosujących wyroby tego producenta lub poszukajcie w Internecie ludzi, do których moglibyście wysłać listy pocztą elektroniczną. Nie wahajcie się – inżynierowie bardzo lubią wyrażać swoje opinie o wyrobach, które stosują. Będą się czuli wyróżnieni, że się do nich zwracacie.



Rys. 28.10. Porównanie asemblera z językiem C. Programy w języku C są bardziej elastyczne i szybciej się je opracowuje. Programy asemblerowe zaś często działają lepiej i z większą szybkością; wykorzystują mniej pamięci, co obniża koszty

Jak szybkie są cyfrowe procesory sygnałowe?

Najważniejszą przyczyną stosowania procesorów DSP zamiast tradycyjnych mikroprocesorów jest *szybkość*, zdolność pobrania próbek do procesora, przeprowadzenia niezbędnych operacji matematycznych i wyprowadzenia przetworzonych danych. I tu powstaje pytanie, jak szybkie są procesory DSP. Normalnym sposobem odpowiedzi na takie pytanie są **wzorcowe testy porównawcze** (*benchmarks*) umożliwiające wyrażenie szybkości mikroprocesora w formie liczbowej. Na przykład szybkość działania systemów stałoprzecinkowych jest często podawana w **MIPS** – milionach operacji na liczbach całkowitych na sekundę. Szybkość systemów zmiennoprzecinkowych jest wyrażana w **MFLOPS** – milionach operacji na liczbach zmiennoprzecinkowych na sekundę.

Sto pięćdziesiąt lat temu premier brytyjski Beniamin Disraeli stwierdził, że są trzy rodzaje kłamstwa: *łgarstwo*, *wierutne łgarstwo* i *statystyka*. Gdyby Israeli żył teraz i pracował z mikroprocesorami, dodałby jeszcze *wzorcowe testy porównawcze* jako czwartą kategorię. Takie testy mają, dzięki bezpośredniemu porównaniu, wykazywać, który procesor jest najlepszy. Niestety, w praktyce ta metoda często zawodzi, gdyż różne mikroprocesory przodują w różnych obszarach zastosowań. Wyobraźcie sobie pytanie: Który samochód jest lepszy – cadillac czy ferrari? Odpowiedź zależy od tego, do czego jest wam ten samochód potrzebny.

Nieporozumienia wokół wzorcowych testów porównawczych pogłębiają się z powodu konkurencyjności w przemyśle elektronicznym. Wytwórcy chcą przedstawiać swoje produkty w jak najlepszym świetle i wszelkie niejednoznaczności powstające przy testowaniu obracają na swoją korzyść. Jest stare powiedzenie w elektronice: *Piszący dane katalogowe uzyskuje z układu działanie dwa razy lepsze niż inżynier*. Ci ludzi nie są kłamcami, lecz po prostu są opłacani za to, żeby mieli bujną wyobraźnię. Testy porównawcze trzeba traktować jako jedno z *narzędzi* w realizacji skomplikowanego zadania. Jeśli nie macie doświadczenia w korzystaniu z tego narzędzia, to możecie dojść do błędnych wniosków. Lepszym sposobem jest przyjrzenie się informacjom o szybkości wykonania algorytmów, z których macie zamiar skorzystać. Gdy wasze zastosowanie wymaga np. filtru SOI, to zwróćcie uwagę na dokładną liczbę okresów zegara potrzebnych do realizacji tego konkretnego zadania w procesorze.

Korzystając z takiego podejścia, zobaczcie, jakie czasy są potrzebne na wykonanie różnych algorytmów przez prezentowane tu procesory sygnałowe rodziny SHARC firmy Analog Devices. Nie zapominajcie też, że szybkość działania mikroprocesorów podwaja się co trzy lata. Wynika stąd, powinniście zwrócić szczególną uwagę na *metodę* zastosowaną w tym przykładzie. Aktualne dane liczbowe ciągle się zmieniają i powinniście powtarzać wasze obliczenia za każdym razem, gdy zaczynacie nowy projekt. W świecie techniki XXI wieku wszystko w okamgnieniu staje się przestarzałe.

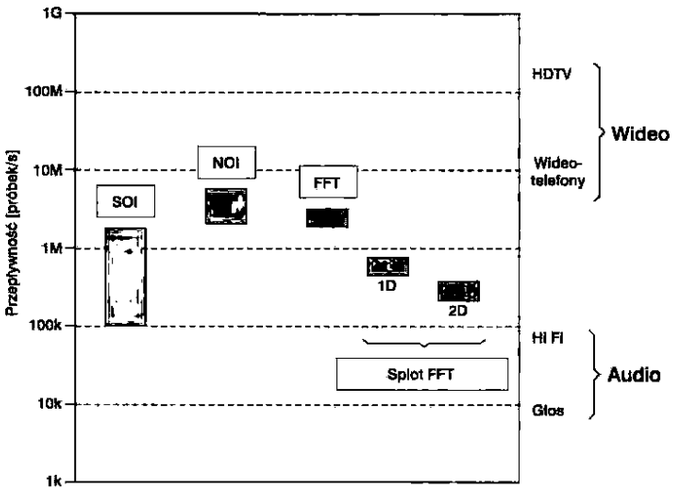
Jeśli chodzi o możliwość zrozumienia czasu wykonania, to najłatwiejszymi pod tym względem są procesory rodziny SHARC. Mogą one bowiem wykonywać operację mnożenia z akumulacją podczas jednego okresu zegarowego. Ponieważ większość filtrów SOI zawiera od 25 do 400 współczynników, więc na każdą przetwarzaną próbkę jest koniecznych od 25 do

400 okresów zegara. Jak już powiedziano poprzednio, jakiś naddatek jest niezbędny do osiągnięcia skuteczności tych pętli (zainicjowanie pierwszej pętli i zakończenie ostatniej), lecz ten naddatek jest pomijalnie mały, jeśli liczba pętli jest tak duża. Aby obliczyć przepływność filtru, możemy podzielić częstotliwość zegarową procesorów SHARC (obecnie 40 MHz) przez liczbę okresów zegara wymaganych na jedną próbkę. To nam daje maksymalną bitową przepływność danych w filtrze SOI równą ok. 100k do 1,6 Mpróbek/s. Trudno o obliczenie prostsze od tego! Wartości przepływności filtru SOI zaznaczono na rys. 28.11.

Te obliczenia są łatwe także dla filtrów rekursywnych. Typowe filtry NOI mają od ok. 5 do 17 współczynników. Ponieważ te pętle są względnie krótkie, więc dodamy jakąś nadwyżkę, powiedzmy 3 okresy na próbkę. Stąd wynika, że na jedną próbkę przetwarzanych danych są potrzebne od 8 do 20 okresów. Przy częstotliwości zegara 40 MHz uzyskuje się maksymalną przepływność od 1,8 M do 3,1 Mpróbek/s. Te wartości dla filtru NOI też podano na rys. 28.11.

Teraz przejdziemy do metod w dziedzinie częstotliwości, w których korzysta się z przekształcenia Fouriera. Podprogramy FFT są prawie zawsze dostarczane przez producenta procesora sygnałowego. Są to procedury wysoce zoptymalizowane, napisane w języku asemblera. Z danych katalogowych procesora sygnałowego SHARC ADSP-21062 wynika, że 1024-próbkowe zespolone przekształcenie FFT wymaga 18 221 okresów zegarowych, czyli ok. 0,46 ms przy 40 MHz. W celu obliczenia przepływności łatwiej jest uwzględniać, że potrzebnych jest 17,8 okresów zegarowych/próbkę. Ta wartość, obliczona dla jednej próbki, niewiele się zmienia dla dłuższych lub krótszych FFT. Na przykład 256-próbkowe FFT wymaga ok. 14,2 okresów zegarowych na próbkę, a 4096-próbkowe FFT – 21,4 okresów zegarowych na próbkę. Rzeczywiste przekształcenia FFT mogą być wykonywane 40% szybciej niż zespolone. Z tego wynika, że wszystkie procedury FFT zajmują w przybliżeniu od 10 do 22 okresów zegarowych na próbkę, co odpowiada przepływności ok. 1,8 M do 3,3 Mpróbek/s.

Splot FFT jest szybkim sposobem realizacji filtrów SOI. W typowym przypadku z wejścia jest pobierany segment 512 próbek, dopełniony 512 zerami i przetwarzany na widmo częstotliwościowe przez 1024-punktowe FFT. Po wymnożeniu tego widma przez żądaną odpowiedź częstotliwościową stosuje się 1024-punktowe odwrotne przekształcenie FFT, aby po-



Rys. 28.11. Szybkość DSP. Przepływność w danym algorytmie DSP można wyznaczyć, dzieląc częstotliwość zegara przez wymaganą na jedną próbkę liczbę okresów zegarowych. Na rysunku przedstawiono zakresy przepływności dla czterech często stosowanych algorytmów wykonywanych na procesorze sygnałowym SHARC o częstotliwości zegara 40 MHz

wrócić do dziedziny czasu. Uzyskane 1024 punkty są łączone z sąsiednimi przetworzonymi segmentami metodą nakładania z dodawaniem (*ovelap-add*). To daje 512 punktów sygnału wyjściowego.

Ile zajmuje to okresów zegarowych? Każdy segment 512-próbkowy wymaga dwóch 1024-punktowych przekształceń FFT z jeszcze pewnym niewielkim naddatkiem. Jest to w przybliżeniu o czynnik 5 więcej niż dla pojedynczego FFT 512-punktowego. Ponieważ rzeczywiste FFT zajmuje około 12 okresów zegarowych na próbkę, więc splot FFT może być przeprowadzany w ciągu ok. 60 okresów zegarowych na próbkę. W procesorze 2106x SHARC, 40 MHz jest to równoważne przepływności 660 kpróbek/s.

Zauważcie, że jest to mniej więcej tyle samo, co dla filtru SOI o 60 współczynnikach realizowanego przez konwencjonalny splot. Innymi słowy, jeśli filtr SOI ma mniej niż 60 współczynników, to może być realizowany szybciej przez splot standardowy. Jeśli zaś ma więcej niż 60 współczynników, to splot FFT jest szybszy. Podstawowa zaleta splotu FFT polega na tym, że czas wykonania wzrasta tylko logarytmicznie w funkcji liczby współczynników. Na przykład 4096-punktowa odpowiedź impulsowa filtru wymaga tylko o ok. 30% dłuższego czasu wykonania niż 512-punktowa.

Splot FFT może też być stosowany do problemów dwuwymiarowych, takich jak przetwarzanie obrazów. Przyjmijmy na przykład, że w dziedzinie częstotliwości chcemy przetwarzać obraz 800 × 600 pikseli. Najpierw dopełnijmy obraz zerami, aby miał rozmiar 1024 × 1024. Następnie oblicza się dwuwymiarowe widmo częstotliwościowe przez wyznaczanie FFT każdego wiersza, po czym – FFT każdej wynikającej z tego kolumny. Po przemnożeniu tego widma 1024 × 1024 przez żądaną odpowiedź częstotliwościową wykonuje się dwuwymiarowe, odwrotne przekształcenie FFT. Tę operację przeprowadza się przez wyznaczanie odwrotnego FFT każdego wiersza i następnie każdej powstałej w rezultacie kolumny. Przez zsumowanie liczb okresów zegarowych i podzielenie przez liczbę próbek obliczamy, że cała procedura zajmuje z grubsza 150 okresów zegarowych na piksel. Dla ADSP-2106, 40 MHz jest to równoważne przepływności danych ok. 260k próbek/s.

Patrząc na porównanie tych trzech różnych metod na rys. 28.11, możemy zrobić ważne spostrzeżenie. *Prawie wszystkie metody DSP do swojej realizacji wymagają od 4 do 400 instrukcji (czyli okresów zegarowych w rodzinie cyfrowych procesorów sygnałowych SHARC).* W przypadku procesorów SHARC o częstotliwości zegarowej 40 MHz możemy od razu wywnioskować, że ich przepływność danych będzie się mieścić w granicach od 100k do 10M próbek/s, w zależności od stopnia złożoności algorytmu.

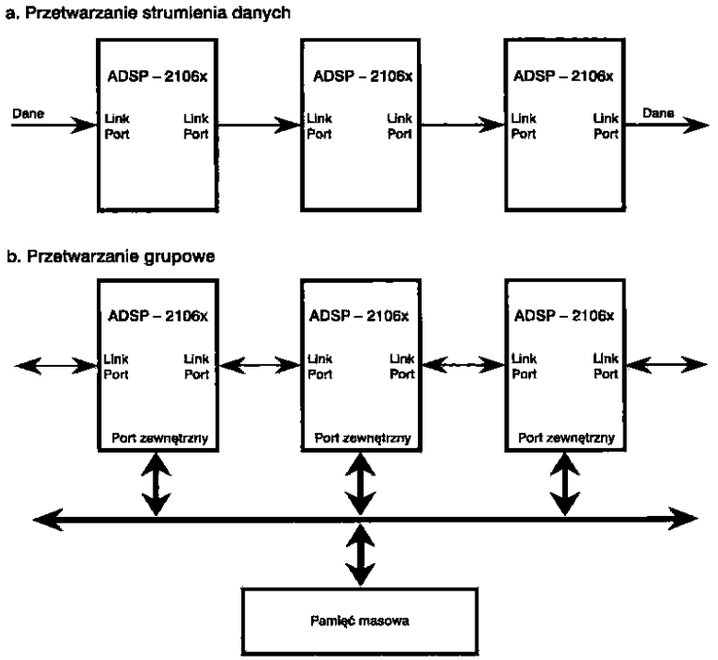
Teraz, gdy już rozumiemy, jak szybko procesor DSP *może* przetwarzać sygnały, spójrzmy na tę kwestię z innej strony – jaka szybkość przetwarzania danych jest nam *potrzebna*? To zależy, oczywiście, od zastosowania. Przyjrzyjmy się dwom najpowszechniejszym – przetwarzaniu sygnałów audio i wideo.

Dla sygnałów fonicznych wymagana bitowa przepływność danych zależy od żądanej jakości odtwarzanego sygnału. W dolnym obszarze jakości mamy telefoniczny sygnał mowy, wymagający wychwycenia zakresu częstotliwości od ok. 100 Hz do 3,2 kHz. Z tego wynika niezbędna częstotliwość próbkowania ok. 8k próbek/s. Natomiast muzyka o dużej wierności odtwarzania musi zawierać pełny zakres częstotliwości odbierany przez ludzkie ucho, więc od 20 Hz do 20 kHz. Często stosuje się częstotliwość próbkowania 44,1 kHz zarówno dla kanału lewego, jak i prawego, dające próbkowanie pełnego sygnału hi-fi z częstotliwością 88,2 kpróbek/s. Jak procesory sygnałowe rodziny SHARC spełniają te wymagania? Jak pokazano na rys. 28.11, te procesory mogą z łatwością przetwarzać sygnały foniczne o wysokiej wierności lub przetwarzać kilkadziesiąt sygnałów głosowych jednocześnie.

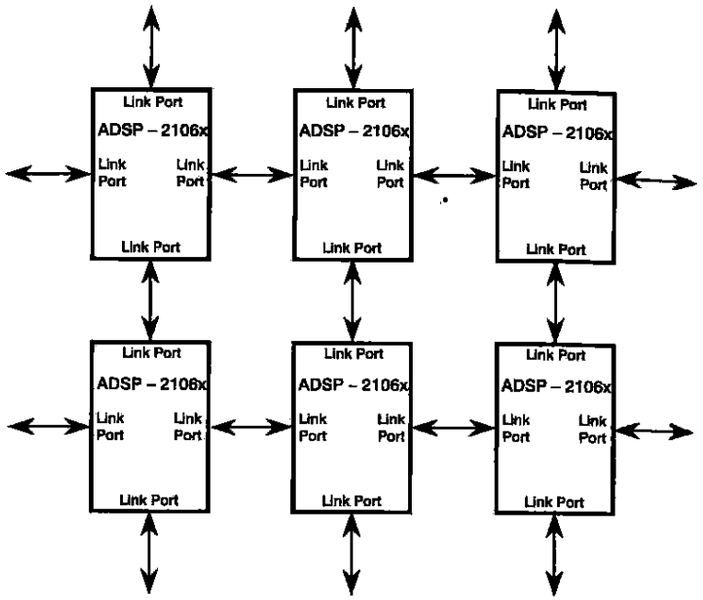
Sygnały wizyjne to zupełnie inna historia. One wymagają przepływności bitowej ok. *tysiąc krotnie* większej niż sygnały audio. Dobrym przykładem sygnału wizyjnego niskiej jakości jest sygnał według standardu CIF (*Common Interface Format*) do wideofonów. Stosuje się

w nim 352 × 288 pikseli, z trzema kolorami na piksel i 30 ramkami na sekundę, co daje przepływność 9,1 miliona próbek na sekundę. W obszarze bardzo dobrej jakości mamy zaś HDTV (telewizja wysokiej jakości), w której stosuje się 1920 × 1080 pikseli, z 3 kolorami na piksel i 30 ramkami na sekundę. Taki sygnał wymaga przepływności ponad 186 milionów próbek na sekundę. Takie przepływności danych przekraczają możliwości pojedynczego procesora sygnałowego rodziny SHARC (rys. 28.11). Są też inne zastosowania, które też wymagają tak dużych przepływności danych, np. radar, sonar i różne aplikacje wojskowe takie, jak zdalne sterowanie pociskami.

Do wykonywania tak trudnych zadań można łączyć kilka procesorów w jeden system. Jest to tzw. **wieloprocesorowość** lub **przetwarzanie równoległe**. Procesory rodziny SHARC projektowano z uwzględnieniem tego rodzaju wieloprocesorowości i wyposażono je w specjalne właściwości ułatwiające w maksymalnym stopniu taki rodzaj pracy. Na przykład do połączenia ze sobą zewnętrznych magistral wielu procesorów SHARC nie są wymagane żadne zewnętrzne układy logiczne. Wszystkie układy służące do tego celu są już zawarte w procesorach. Alternatywą jest zastosowanie portów *Link Port* (4 bity, równolegle) do łączenia wielu procesorów w różnych konfiguracjach. Typowe sposoby łączenia procesorów sygnałowych SHARC w systemy wieloprocesorowe przedstawiono na rys. 28.12. Na rys. 28.12 a) algorytm podzielono na kolejne etapy, przy czym każdy procesor wykonuje jeden z kroków metodą „linii montażowej". Na rys. 28.12 b) procesory współdziałają za pośrednictwem wspólnej pamięci ogólnej dostępnej przez magistralę równoległą (tj. przez port zewnętrzny). Inny sposób łączenia dużej liczby procesorów w jeden system pokazano na rys. 28.13. System jest „oczkiem siatki" 2D lub 3D. Każda z przedstawionych konfiguracji ma względne zalety i wady w odniesieniu do określonego zastosowania.



Rys. 28.12. Konfiguracje wieloprocesorowe. W systemach wieloprocesorowych typowo stosuje się jeden z dwóch sposobów komunikacji między węzłami procesorowymi : a) przez dedykowane kanały komunikacji typu punkt-punkt lub b) przez wspólną pamięć ogólną, dostępną przez magistralę równoległą



Rys. 28.13. Konfiguracja wieloprocesorowa typu „oczko sieci". W zastosowaniach takich jak obrazowanie radarowe najlepszym sposobem koordynacji działania dużej liczby procesorów może być matryca 2D lub 3D

Aby życie programisty uczynić łatwiejszym, w rodzinie procesorów SHARC zastosowano *ujednoliconą przestrzeń adresową* (*unified address space*). To oznacza, że przestrzeń adresowa 4G słów, dostępna przez 32-bitową magistralę adresową, jest podzielona między różne, pracujące wspólnie procesory. Przesłanie informacji z jednego procesora do drugiego wymaga po prostu odczytu lub zapisu w odpowiednich lokacjach pamięci. O resztę „zatroszczą się" wewnętrzne układy logiczne procesorów SHARC, przesyłając dane między procesorami z dużą przepływnością bitową 240 MB/s (przy zegarze 40 MHz).

Rynek cyfrowych procesorów sygnałowych (DSP)

Rynek procesorów DSP jest bardzo duży i gwałtownie się rozwija. Jego wartość wynosiła 8–10 miliardów USD/rok na przełomie stuleci i rosła w tempie 30–40% rocznie (rys. 28.14). Siłą napędową tego rynku jest nieustające zapotrzebowanie na lepsze i tańsze urządzenia powszechnego użytku, takie jak telefony komórkowe, komputery multimedialne i wysokiej klasy urządzenia do odtwarzania muzyki. Rynek kształtują te właśnie urządzenia przynoszące bardzo duży dochód, podczas gdy dziedziny mniej dochodowe, jak aparatura badawcza, „płyną na fali" nowej techniki.

DSP można kupować w trzech postaciach – jako **rdzeń**, jako **procesor** oraz jako **układ z płytką drukowaną**. W DSP pojęcie „rdzeń" odnosi się do tej części procesora, w której są realizowane jego główne zadania, co obejmuje rejestry danych, układ mnożący, jednostkę arytmetyczno-logiczną (ALU), generator adresów i sekwenser programu. Pełny **procesor** wymaga jeszcze połączenia rdzenia z pamięcią i interfejsami służącymi do kontaktów ze światem zewnętrznym. Chociaż rdzeń i te układy peryferyjne są projektowane oddzielnie, to są wytwarzane w tym samym kawałku krzemu, dzięki czemu *procesor* jest pojedynczym układem scalonym.

Przypuśćmy, że konstruujecie telefony komórkowe i chcecie zastosować DSP. Prawdopodobnie będziecie chcieli kupić DSP w postaci *procesora*, a więc układu scalonego zawierającego rdzeń, pamięć i inne niezbędne cechy wewnętrzne. Na przykład procesor sygnałowy SHARC ADSP-21060 jest wytwarzany w 240-końcówkowej obudowie typu „Metric PQFP", o rozmiarach tylko 35 × 35 × 4 mm. Aby wprowadzić ten układ scalony do waszego produktu, projektujecie płytkę drukowaną, w której procesor będzie wlutowany obok innych układów elektronicznych. To jest najczęstszy sposób zastosowania DSP.

A teraz przyjmijmy, że firma, dla której pracujecie, produkuje własne układy scalone. W tym przypadku możecie nie potrzebować całego *procesora*, a wystarczy wam projekt samego *rdzenia*. Po zawarciu stosownych umów licencyjnych możecie zacząć wytwarzanie układów scalonych w wysokim stopniu dostosowanych do waszego określonego zastosowania. Dzięki temu uzyskujecie elastyczność wyboru wielkości pamięci, która ma znaleźć się w tej strukturze scalonej oraz wyboru sposobu wprowadzania i wyprowadzania danych z układu, a także rodzaju obudowy itd. Układy tego rodzaju wykonywane na zamówienie stają się coraz ważniejszym segmentem rynku DSP.

Jest też wreszcie kilkadziesiąt firm sprzedających cyfrowe procesory sygnałowe już zmontowane na płytce drukowanej. Mają one takie właściwości, jak dodatkowa pamięć, przetworniki a/c i c/a, gniazda do układów EPROM, wiele procesorów na jednej płytce itd. Niektóre z tych płytek są przewidziane do pracy jako samodzielne komputery, lecz większość skonfigurowano w taki sposób, żeby były wtykane do gniazda w jednostce głównej, np. w komputerze osobistym. Firmy produkujące tego rodzaju płytki są nazywane **wytwórcami pośredniczącymi**. Najlepszym sposobem znalezienia takich wytwórców jest zapytanie o nich w firmach wytwarzających te procesory DSP, które macie zamiar zastosować. Przejrzyjcie strony internetowe producentów; jeśli nie znajdziecie tam interesującej was listy, wyślijcie do nich list pocztą elektroniczną. Będą bardzo zadowoleni, mogąc poinformować was, kto stosuje ich produkty i jaki jest kontakt z tymi wytwórcami.

Rynek cyfrowych procesorów sygnałowych był w roku 2002 zdominowany przez dwie firmy. Poniżej podano te firmy i ogólną zasadę, według której oznaczają swoje produkty:

Analog Devices (*www.analog.com/dsp*)

ADSP-218x	16-bitowy, stałoprzecinkowy
ADSP-219x	16-bitowy, stałoprzecinkowy, podwójne działanie procesora 218x
ADSP-2106x	32-bitowy, zmiennoprzecinkowy
ADSP-2116x	32-bitowy, zmiennoprzecinkowy, rdzeń SIMD
ADSP-2153x	16-bitowy, o dużej wydajności
ADSP-TSxxx	z możliwością pracy 8-, 16-, 32-bitowej, stało- i zmiennoprzecinkowy, MIMD, o bardzo dużej wydajności.

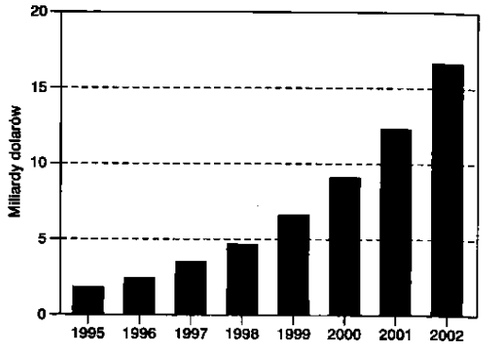
Texas Instruments (*www.ti.com*)

TMS320C6xx	16-bitowy, zmiennoprzecinkowy
TMS320C5xx	32-bitowy, zmiennoprzecinkowy
TMS320C2xx	32-bitowy, stałoprzecinkowy.

Pamiętajcie o tym, że rozróżnienie między cyfrowym procesorem sygnałowym (DSP) a innymi mikroprocesorami nie zawsze jest wyraźne. Spójrzmy, na przykład, jak Intel opisuje wprowadzenie techniki MMX do swojego procesora Pentium:

Inżynierowie Intela wprowadzili 57 silnych, nowych instrukcji, specjalnie opracowanych do skutecznego operowania i przetwarzania sygnałów wizyjnych i fonicznych oraz danych graficznych. Te instrukcje są ukierunkowane na powtarzalne sekwencje w wielkim stopniu równoległe, często spotykane w operacjach multimedialnych.

W przyszłości z pewnością pojawi się więcej funkcji zbliżonych do DSP, a włączanych do tradycyjnych mikroprocesorów i mikrosterowników. Siłą napędową tych zmian są aplikacje związane z Internetem i multimediami. Te zastosowania rozwijają się tak szybko, że za dwadzieścia lat, być może, cyfrowy procesor sygnałowy *stanie się* „tradycyjnym" mikroprocesorem.

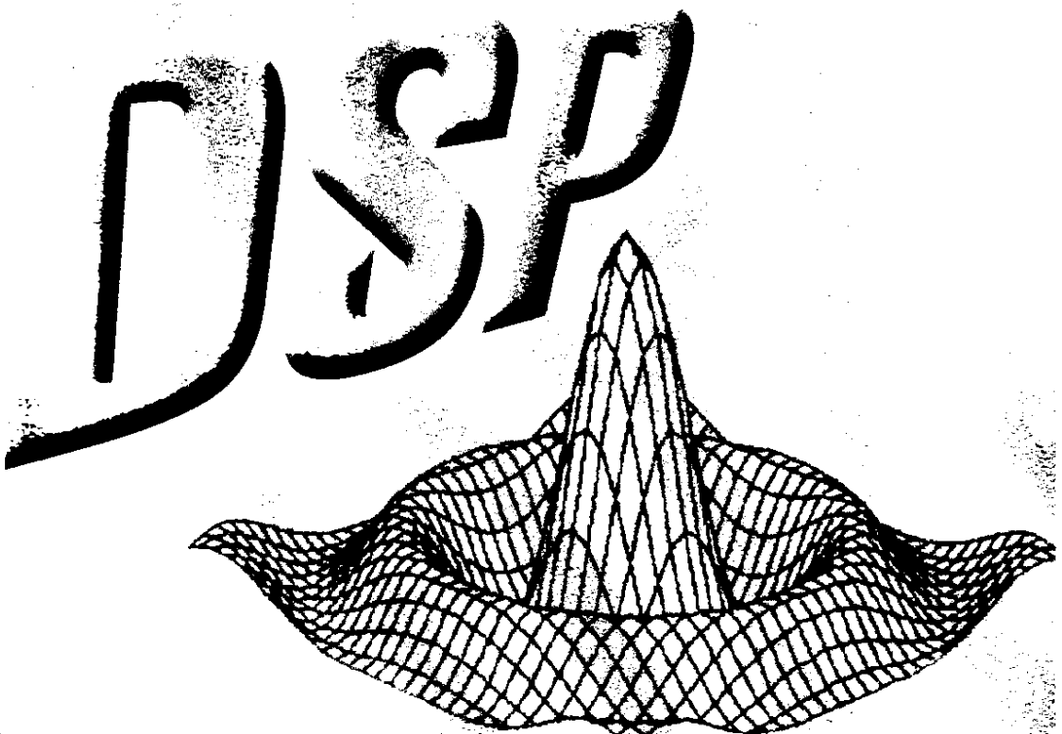


Rys. 28.14. Rynek DSP. Na przełomie stuleci rynek DSP miał wartość 8-10 miliardów USD/rok i rozwijał się w tempie ok. 30–40% rocznie

Jak nadążyć za tak szybko zmieniającą się dziedziną? Najlepszym sposobem jest czytanie fachowych czasopism śledzących rynek DSP, takich jak EDN („Electronic Design News", *http//: www.ednmag.com*) i ECN („Electronic Component News", *http//: www. ecnmag.com*). Są one rozprowadzane gratisowo i zawierają najaktualniejsze informacje o tym, co jest dostępne w sprzedaży oraz o kierunkach rozwoju. Lektura fachowym czasopism jest niezbędna dla wszystkich poważnie zainteresowanych dziedziną DSP. Powinniście też się znaleźć na listach wysyłkowych kilku producentów DSP. Wtedy będziecie otrzymywać zapowiedzi nowych produktów, cenniki i oferty specjalne (takie jak darmowe oprogramowanie i tanie zestawy próbne). Niektórzy producenci wydają też periodyczne biuletyny informacyjne. Na przykład Analog Devices ma witrynę *www*, która jest ciągle uaktualniana nowymi informacjami. Możecie zaprenumerować kilka biuletynów kwartalnych: *http://www.analog.com/misc/enewsletters_jumppage.html*. Zawsze możecie rozpocząć od przejrzenia witryn internetowych producentów i wysłania do nich listów pocztą elektroniczną z prośbą o dostarczenie potrzebnych wam informacji.

29

Rozpoczęcie pracy z DSP



Gdy już zdecydujecie, że cyfrowy procesor sygnałowy jest potrzebny do waszego zastosowania, to powinniście wiedzieć, od czego zacząć. Wielu producentów przyśle wam tanie zestawy próbne umożliwiające zdobycie pierwszych doświadczeń z ich produktami. Są to świetne narzędzia edukacyjne. Niezależnie od tego, czy jesteście nowicjuszami, czy profesjonalistami, taki zestaw jest najlepszym sposobem zapoznania się z konkretnym procesorem sygnałowym. Na przykład firma Analog Devices dostarcza zestawy EZ-KIT® Lite, aby potencjalnych klientów uczyć pracy z cyfrowymi procesorami sygnałowymi rodziny SHARC®. Za niewielką cenę otrzymujecie wszystkie układy i oprogramowanie niezbędne do tego, aby zobaczyć DSP w działaniu. Obejmuje to także standardowe oprogramowanie dostarczane wraz z zestawem, jak również przykłady aplikacji, do których sami możecie napisać programy w języku asemblerowym lub C. Przypuśćmy, że kupiliście jeden z tych zestawów Analog Devices i bawicie się nim przez kilka dni. Ten rozdział zawiera przegląd tego, co możecie znaleźć w takim zestawie i czego możecie się nauczyć.

Rodzina ADSP-2106x

W poprzednim rozdziale omówiliśmy ogólnie działanie rodziny cyfrowych procesorów sygnałowych ADSP-2106x rodziny SHARC. W tablicy 29.1 zestawiono różne procesory należące do tej rodziny. Wszystkie te układy mają taką samą architekturę, lecz różne wielkości pamięci wewnętrznej w strukturze monolitycznej, a właśnie wielkość pamięci jest czynnikiem decydującym o wyborze procesora do danego zastosowania. Dostęp do pamięci jest na ogół wąskim gardłem systemów DSP. W procesorach SHARC rozwiązano ten problem dzięki umieszczonym w strukturze monolitycznej dużym, dwuwejściowym pamięciom SRAM. Jednak ostatnią rzeczą, jaką chcielibyście zrobić, byłoby płacenie za więcej pamięci niż jest potrzebne. Procesory sygnałowe są często stosowane w urządzeniach, w których niskie koszty są bardzo ważne, np. w telefonach komórkowych i odtwarzaczach CD. Można więc powiedzieć, że zestaw procesorów w rodzinie SHARC jest podyktowany zarówno *technologią*, jak i *marketingiem*.

SHARC, *EZ-KIT*, *EZ-LAB*, *VisualDSP*, *EZ-ICE*, logo SHARC, logo Analog Devices i logo VisualDSP są zastrzeżonymi znakami towarowymi firmy Analog Devices, Inc.

Przeglądając tablicę 29.1, zwróćcie uwagę, że nawet tańsze układy mają całkiem dużo pamięci. Na przykład ADSP-21065L ma 544 kilobitów wewnętrznej pamięci SRAM. To wystarcza do zarejestrowania 6–8 sekund sygnału mowy przetworzonego na cyfrowy (8 kilopróbek 8-bitowych). Wśród najdroższych układów z tej rodziny ADSP-21060 ma 4 megabity pamięci. Jest to więcej, niż trzeba do zarejestrowania całego obrazu w postaci cyfrowej (512 × 512 pikseli, 8 bitów na piksel). Jeśli potrzebujecie jeszcze więcej pamięci, możecie łatwo, do każdego z tych układów dodać zewnętrzną pamięć SRAM (lub pamięć wolniejszą).

Różnice między procesorami rodziny SHARC są nie tylko w pamięciach, lecz również w układach wejścia/wyjścia. Układy wyższej klasy, ADSP-21060 oraz ADSP-21062, mają po 6 portów *Link Port*. Są to równoległe połączenia 4-bitowe służące do tworzenia systemów wieloprocesorowych i do innych zastosowań wymagających elastycznego, bardzo szybkiego wejścia/wyjścia. Układy nieco niższej klasy (ADSP-21061 i ADSP-21065L) nie mają portów *Link Port*, lecz charakteryzują się większą liczbą kanałów DMA wspomagających działanie *portów szeregowych*. Przy oznaczeniach typów tych układów spotkacie też litery L lub M, np. ADSP-21060L. To oznacza, że procesor pracuje przy napięciu niższym niż tradycyjne 5,0 V. Na przykład ADSP-21060L pracuje z napięciem zasilającym 3,3 V, a ADSP-21160M z napięciem tylko 2,5 V.

W czerwcu 1998 roku firma Analog Devices zaprezentowała drugą generację swych procesorów o architekturze SHARC, wprowadzając na rynek ADSP-21160. Charakteryzuje się on rdzeniem o architekturze SIMD (pojedyncze rozkazy – wiele danych) pracującym z częstotliwością 100 MHz, magistralą pamięci o rozszerzonym paśmie dającym przepływność 1600 MB/s, dwiema 64-bitowymi magistralami danych i czterema 80-bitowymi akumulatorami do obliczeń stałoprzecinkowych. Dzięki takim właściwościom procesor ADSP-21160M wykonuje 1024-punktowe przekształcenie FFT w czasie zaledwie 46 µs. Cyfrowy procesor sygnałowy SIMD zawiera drugi zestaw jednostek obliczeniowych (jednostkę arytmetyczno-logiczną, szybki rejestr przesuwający (*barrel shifter*), tablicę rejestrów danych i układ mnożący), dzięki czemu procesor zachowuje wsteczną kompatybilność programu z rodziną ADSP-2106x, a stwarza możliwość działania dziesięciokrotnie lepszego niż tamte układy.

Tab. 29.1. Cyfrowe procesory sygnałowe rodziny SHARC

ADSP-TS101	6 Mbit	Rdzeń SIMD (pojedyncze rozkazy – wiele danych), bardzo szybki i o szerokich możliwościach, zoptymalizowany dla pracy wieloprocesorowej, rdzeń o małym napięciu zasilającym i zmniejszonym poborze mocy.
ADSP-21160	4 Mbit	Rdzeń SIMD (pojedyncze rozkazy – wiele danych), zoptymalizowany do pracy wieloprocesorowej z portami Link Port, 64-bitową magistralą zewnętrzną i 14 kanałami DMA.
ADSP-21161	1 Mbit	Tańsza wersja ADSP-21160, z mniejszą liczbą portów Link Port i mniejszą pamięcią wewnętrzną, lecz z dodatkowymi protokołami portów szeregowych (SPI, I²S).
ADSP-21060	4 Mbit	Rdzeń SISD (pojedyncze rozkazy–pojedyncze dane), porty Link Port do przesyłania danych z dużą przepływnością i do pracy wieloprocesorowej, 48-bitowa magistrala zewnętrzna.
ADSP-21062	2 Mbit	Takie same właściwości jak ADSP-21060, lecz z mniejszą pamięcią wewnętrzną (SRAM) – dzięki temu jest tańszy.
ADSP-21061	1 Mbit	Tańsza wersja, stosowana w EZ-KIT Lite, mniej pamięci, nie ma portów Link Port, dodatkowe możliwości w DMA dla portów szeregowych.
ADSP-21065	544 kbit	Szybszy i nie tak drogi, jak inne procesory 2106x, zainteresuje rodziną SHARC użytkowników z obszarów zastosowań stałoprzecinkowych i audio.

Zestaw próbny SHARC EZ-KIT Lite[30]

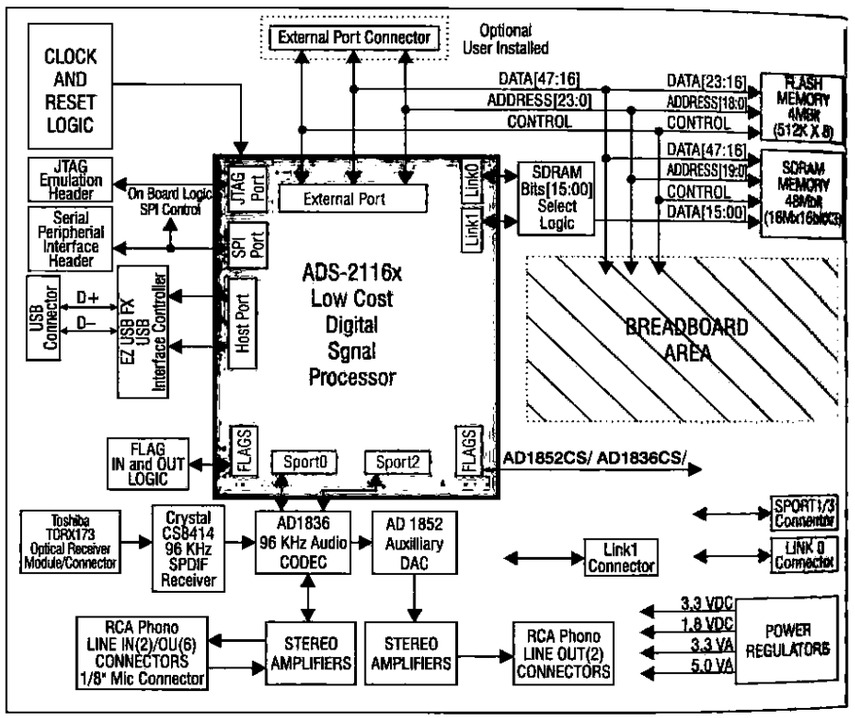
W zestawie próbnym EZ-kit Lite znajdziecie wszystko, co jest niezbędne do nauczenia się pracy z cyfrowymi procesorami sygnałowymi SHARC, a więc część sprzętową, oprogramowanie i podręcznik. Na rys. 29.1 przedstawiono schemat blokowy układu dostarczanego w zestawie EZ-KIT Lite, którego najważniejszym elementem jest cyfrowy procesor sygnałowy ADSP-21161. Układy są zmontowane na płycie drukowanej 13,5 × 20 cm. Połączeń, którymi musicie się zająć, jest niewiele – zasilanie napięciem stałym, łącze USB do komputera osobistego oraz sygnały wejściowy i wyjściowy. Zasilacz i kabel USB są zresztą dostarczane w zestawie. Sygnały wejściowy i wyjściowy są na poziomie sygnałów akustycznych, o amplitudzie ok. 1 V. Myśl jest tu taka, żeby dołączyć mikrofon do wejścia, a do wyjścia – kilka wzmacniających głośników (takich, jak stosowane w komputerach osobistych). W ten sposób możecie *usłyszeć* efekty działania różnych algorytmów DSP.

Przetwarzanie analogowo-cyfrowe i cyfrowo-analogowe jest realizowane w koderze-dekoderze (*codecu*) AD1836 firmy Analog Devices. Jest to koder-dekoder audio 96 kHz zawierający również przetwornik c/a AD1852.

Przyciski na płycie umożliwiają użytkownikowi generowanie przerwań, kasowanie procesora i przełączanie kilku bitów flagowych, które mogą być czytane przez system. Sześć diod elektroluminescencyjnych wmontowanych na płycie może być włączanych i wyłączanych przez przełączanie stanów bitów. Jeśli jesteście ambitni, to możecie wykorzystać znajdujące się na płycie sekcje dające dostęp do portu szeregowego, portów *Link Port*, interfejsu optycznego SPDIF i magistrali procesora.

[30] Oferta zestawów ewaluacyjnych dla procesorów DSP firmy Analog składała się z 8 typów dla różnych rodzin DSP. W skład zestawów wchodzi m.in. 90-dniowa licencja na pakiet programistyczny VisualDSP++. Szczegółowe informacje są dostępne pod adresem: *http://www.analog.com/processors/sharc/evaluationDevelopment/evaluationKits.html*.

A oto jak to działa. Po włączeniu zasilania procesor inicjuje pracę przez pobranie programu z pamięci *flash* umieszczonej na płycie drukowanej. Ten program nawiązuje łączność USB JTAG z waszym komputerem osobistym. Następnie uruchamiacie na komputerze program *VisualDSP++* umożliwiający ładowanie programów i pobieranie danych z DSP oraz tworzenie, kompilowanie i asemblację programów przeznaczonych do wykonania w zestawie próbnym. Jest też program do ładowania pamięci *flash*, dzięki czemu możecie spowodować, żeby wasze programy stały się ładowalne, a więc abyście mogli realizować aplikacje autonomiczne. Wraz z zestawem EZ-KIT Lite jest dostarczanych kilka wcześniej napisanych programów. Na przykład dzięki programowi pasmowoprzepustowemu możecie mówić do mikrofonu i słyszeć rezultat uzyskany po przepuszczeniu sygnału przez filtr pasmowoprzepustowy. Te programy są pożyteczne z dwóch przyczyn: 1) umożliwiają szybkie uruchomienie systemu i wykonanie czegoś interesującego, co na pewno będzie działać, 2) są szablonem do tworzenia waszych własnych programów. Teraz dochodzimy więc do następnego tematu – przykładu projektowania z zastosowaniem zestawu EZ-KIT Lite. Przykład projektowania zrealizowano, stosując starszy zestaw próbny ADSP-21061 EZ-KIT Lite. Jest on jednak programowo kompatybilny z ADSP-21161 EZ-KIT, a lepsze działanie nowego, szybszego układu ADSP-21161 polega na krótszym okresie wykonywania instrukcji programowych.

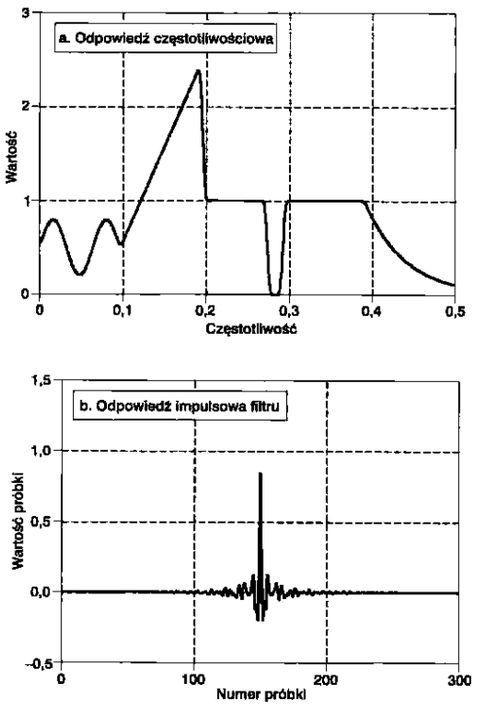


Rys. 29.1. Schemat blokowy płytki drukowanej zestawu próbnego EZ-KIT Lite. Potrzebnych jest tylko kilka połączeń: kilka opcji wejść/wyjść audio, kilkukanałowy wtyk foniczny RCA, interfejs optyczny SPDIF, zasilanie oraz kabel magistrali USB (dostarczany z zestawem) służący do połączenia z komputerem osobistym. (Ze względu na firmowy charakter zestawu zachowano nazewnictwo angielskie – przyp. tłum.)

Przykład projektowania – filtr SOI do sygnałów fonicznych

Gdy już trochę poeksperymentujecie z wcześniej napisanymi programami, dostarczonymi z zestawem, będziecie chcieli je zmodyfikować, aby zdobyć nowe doświadczenia w programowaniu. Programy można pisać albo w asemblerze, albo w języku C. Zestaw EZ-KIT Lite zawiera narzędzia programowe dla obu tych języków. W dalszej części tego rozdziału przyjrzymy się zaawansowanym metodom programowania, takim jak *symulacja*, *debugowanie* i praca ze *zintegrowanym środowiskiem tworzenia aplikacji*. Na razie skoncentrujemy uwagę na najłatwiejszym sposobie uruchomienia programu. Będziemy więc posuwać się do przodu małymi krokami.

Ponieważ kod źródłowy jest w ASCII, więc standardowy edytor tekstu jest wszystkim, co niezbędne do wprowadzania zmian w istniejących plikach lub do tworzenia całkiem nowych programów. W tablicy 29.2 podano przykład programu filtru SOI napisanego w asemblerze. Choć, jak na razie, jest to jedyny potrzebny wam program, to miejcie na uwadze, że są jeszcze inne pliki niezbędne do stworzenia pełnego programu. Obejmują one „plik opisu architektury" (określający konfigurację układową i alokację pamięci), ustawienie tablicy wektorów przerwań oraz podprogram inicjalizacji kodera-dekodera. W końcu będziecie musieli zrozumieć działanie tych części programu, lecz teraz po prostu skopiujecie je z programów napisanych wcześniej.



Rys. 29.2. Przykład filtru SOI. Na rys. a) pokazano odpowiedź częstotliwościową bardzo wyspecjalizowanego filtru. Odpowiadająca jej odpowiedź impulsowa filtru jest przedstawiona na rys. b). Ten filtr zaprojektowano w rozdziale 17, aby wykazać, że w filtrach cyfrowych SOI można uzyskać w zasadzie każdą odpowiedź częstotliwościową

Tab. 29.2. Program filtru SOI napisany w asemblerze

```
Przed wejściem do programu głównego powinny być zdefiniowane następujące stałe i zmienne:
NR_COEF        Liczba współczynników w odpowiedzi impulsowej filtru (w tym przykładzie 301)
dline[NR_COEF] Bufor cykliczny przechowujący poprzednie próbki wejściowe, w pamięci danych
coef[NR_COEF]  Bufor cykliczny przechowujący współczynniki filtru, w pamięci programu

001 /*************************************************************
002 ***************        PROGRAM GŁÓWNY        ******************
003 *************************************************************/
004 main:
005
006 /* INICJALIZACJA GENERATORÓW ADRESÓW DANYCH (DAG) DO STEROWANIA BUFORAMI CYKLICZNYMI
007
008 b0 = dline;                  /* ustaw dline[ ], bufor przechowujący poprzednie próbki wej
009 l0 = @dline;
010 m0 = 1;
011 b8 = coef;                   /* ustaw coef[ ], bufor przechowujący współczynniki filtru
012 l8 = @coef;
013 m8 = 1;
014
015 /* WEJŚCIE W PĘTLĘ, CZEKANIE NA PRZERWANIE "PRÓBKA GOTOWA"*/
016
017 wait:
018 idle;
019 jump wait;
020
021
022 /*************************************************************/
023 *********      PODPROGRAM DO PRZETWARZANIA JEDNEJ PRÓBKI ********
024 *************************************************************/
025 sample_ready:
026
027 /* POBIERZ PRÓBKĘ WEJŚCIOWĄ, PRZETWÓRZ NA LICZBĘ ZMIENNOPRZECINKOWĄ*/
028
029 r0 = dm(rx_buf+ 1);          /* przenieś próbkę wejściową do r0*/
030 r0 = lshift r0 by 16;        /* przesuń do najstarszych 16 bitów, aby zachować znak*/
031 r1 =-31;                     /*ustaw skalowanie przetwarzania*/
032 f0 = float r0 by r1;         /* przetwórz z formatu stało- na zmiennoprzecinkowy*/
033 dm(i0,m0) = f0;              /* zapamiętaj nową próbkę w dline[ ] i zeruj f12*/
034
035 /* OBLICZ PRÓBKĘ WYJŚCIOWĄ FILTRU SOI*/
036
037  f12 = 0;                    /* pierwsze załadowanie rejestrów*/
038 f2 = dm(i0,m0), f4 = pm(i8,m8);
039 f8 = f2*f4, f2 = dm(i0,m0), f4 = pm(i8,m8);
040                              /* efektywna pętla główna*/
041 lcntr = NR_COEF-2, do (pc,1) until lce;
042 f8 = f2*f4, f12 = f8+f12, f2 = dm(i0,m0), f4 = pm(i8,m8);
043
044 f8 = f2*f4, f12 = f8+f12;    /* zakończenie ostatniej pętli*/
045 f12 = f8+f12;
046
047 /* PRZETWÓRZ PRÓBKĘ WYJŚCIOWĄ DO FORMATU STAŁOPRZECINKOWEGO I WYPROWADŹ PRÓBKĘ DO WYJŚCIA
048
049 r1 =31;                      /* ustaw skalowanie przetwarzania*/
050 r8 = fix f12 by r1           /* przetwórz z formatu zmienno- do stałoprzecinkowego*/
051 rti(db);                     /* wyjdź z przerwania, lecz wykonaj następne 2 wiersze*/
052 r8 = lshift r8 by -16        /* przesuń do najmłodszych 16 bitów*/
053 dm(tx_buf + 1) = r8;         /* przenieś próbkę na wyjście*/
```


Jak podano w górnej części tablicy 29.2, trzeba – przed przejściem do głównej części programu – zdefiniować trzy zmienne. Są to: liczba punktów w odpowiedzi impulsowej filtru **NR_COEF**; bufor cykliczny **dline[ ]** przechowujący poprzednie próbki sygnału wejściowego oraz bufor cykliczny przechowujący odpowiedź impulsową filtru **coef[ ]**. Musimy też dostarczyć programowi dwie dalsze informacje – częstotliwość próbkowania w układzie kodująco-dekodującym oraz nazwę pliku zawierającego odpowiedź impulsową filtru, aby mogła być ona wpisana do coef[ ]. Wszystkie te kroki są łatwe, każdy z nich to nie więcej niż jeden wiersz programu. Nie pokazujemy ich w tym przykładzie, gdyż są zawarte w tych sekcjach programu, które dla uproszczenia pomijamy.

Na rys. 29.2 przedstawiono odpowiedź impulsową filtru, która posłuży do przetestowania programu. Jest to taki sam filtr, jaki – zgodnie ze szczególnymi wymaganiami – zaprojektowaliśmy w rozdziale 17. Jak pamiętacie, ten filtr miał mieć bardzo nieregularną odpowiedź częstotliwościową w celu zwrócenia uwagi na fakt, że w filtrach cyfrowych SOI można uzyskać każdą żądaną odpowiedź częstotliwościową. Tę odpowiedź testowego filtru pokazano na rys. 29.2 a), na rys. 29.2 b) zaś – odpowiadającą jej odpowiedź impulsową. Ta 301-punktowa odpowiedź impulsowa filtru jest zapisana jako plik ASCII i jest łączona z innymi sekcjami programu podczas tworzenia jednego wykonywalnego programu.

W głównej sekcji programu są wykonywane dwie funkcje. W wierszach od 6 do 13 generatory adresów danych (DAG) są konfigurowane do zarządzania buforami cyklicznymi dline[ ] i coef[ ]. Jak już stwierdzono w poprzednim rozdziale, dla każdego bufora są niezbędne trzy parametry: początkowa lokacja bufora w pamięci (b0 i b8), długość bufora (l0 i l8) oraz rozmiar porcji danych gromadzonej w buforze (m0 i m8). Te parametry sterujące buforami cyklicznymi są zapisane w rejestrach sprzętowych w generatorach adresów danych, dzięki czemu mogą mieć skuteczny dostęp do danych i bardzo efektywnie nimi zarządzać.

Drugim działaniem programu głównego jest pętla typu „kręcenie młynka palcami", realizowana w wierszach od 15 do 19. Nie robi ona niczego innego poza czekaniem na przerwanie wskazujące, że próbka wejściowa została pobrana. Całe przetwarzanie w tym programie odbywa się na zasadzie **próbka po próbce**. Za każdym razem, gdy zostaje odczytana próbka z wejścia, próbka sygnału wyjściowego jest obliczana i następnie kierowana do kodera-dekodera. Większość algorytmów w dziedzinie czasu, takich jak filtry SOI i NOI, zalicza się do tej kategorii. Alternatywą jest przetwarzanie systemem **ramka po ramce**, wymagane w technikach realizowanych w dziedzinie częstotliwości. W metodzie ramka po ramce z wejścia jest odczytywana *grupa* próbek, potem są przeprowadzane obliczenia i *grupa* próbek jest wpisywana do wyjścia.

Podprogram obsługujący przerwanie „próbka gotowa" składa się z trzech sekcji. W pierwszej sekcji (wiersze od 27 do 33) następuje pobranie z kodera-dekodera próbki jako liczby stałoprzecinkowej i przetworzenie jej w liczbę zmiennoprzecinkową. W języku asemblera SHARC rejestr danych zawierający liczby stałoprzecinkowe jest oznaczany przez „r" (np. r0, r8, r15 itd.) i przez „f" (tj. f0, f8, lub f15), jeśli zawiera liczby zmiennoprzecinkowe. Na przykład w wierszu 32 liczba stałoprzecinkowa w rejestrze danych 0 (tj. r0) jest przetwarzana na liczbę zmiennoprzecinkową i wpisywana do rejestru danych 0 (tj. f0). To przetworzenie jest wykonywane zgodnie ze skalowaniem wyspecyfikowanym przez liczbę stałoprzecinkową w rejestrze danych 1 (tj. r1). W trzeciej sekcji programu (wiersze od 47 do 53) następuje działanie odwrotne – liczba zmiennoprzecinkowa z próbki wyjściowej jest przetwarzana na stałoprzecinkową i przesyłana do kodera-dekodera.

Filtr SOI przetwarzający próbki wejściowe na wyjściowe jest zawarty w wierszach od 35 do 45. Wszystkie obliczenia są przeprowadzane na liczbach zmiennoprzecinkowych, dzięki czemu nie trzeba się kłopotać skalowaniem i przepełnieniem. Jak to opisano w poprzednim rozdziale, ta sekcja programu jest zoptymalizowana pod kątem wykorzystania możliwości wykonywania instrukcji wielokrotnych w jednym okresie zegarowym w cyfrowych procesorach sygnałowych SHARC.

Gdy mamy już napisany program asemblerowy i zaprojektowaną odpowiedź impulsową filtru, to jesteśmy gotowi do stworzenia programu, który mógłby być wykonywany na procesorze sygnałowym SHARC. W tym celu uruchamia się *kompilator*, *asembler* i następnie *program łączący* (*linker*) – trzy programy dostarczane z zestawem EZ-KIT Lite. Kompilator przekształca program w języku C na program asemblerowy SHARC. Jeśli piszecie program bezpośrednio w asemblerze, jak w tym przykładzie, to ten etap pomijacie. Asembler i program łączący przekształcają program i pliki zewnętrzne (takie jak plik architektury, podprogramy inicjalizujące koder-dekoder, odpowiedź impulsowa filtru itd.) do końcowej postaci plików nadających się do wykonania. Wszystko to zajmuje ok. 30 sekund, a końcowym wynikiem jest program SHARC na twardym dysku waszego komputera osobistego. Następnie program jest wykonywany w komputerze głównym, którym jest EZ-KIT Lite. Po prostu kliknij na ten plik, który chcesz uruchomić w DSP, a procesor EZ-KIT Lite „zatroszczy" się o resztę, ładując program i rozpoczynając jego wykonywanie.

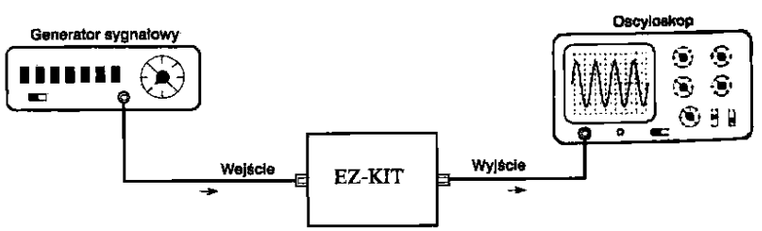
Tu dochodzimy do dwóch pytań. Po pierwsze, jak przetestujemy nasz filtr audio, aby się upewnić, że działa zgodnie z naszym projektem, a po drugie – co to, do licha, jest za firma, która nazywa się **Analog** Devices, a produkuje cyfrowe procesory sygnałowe?

Pomiary analogowe w systemie DSP

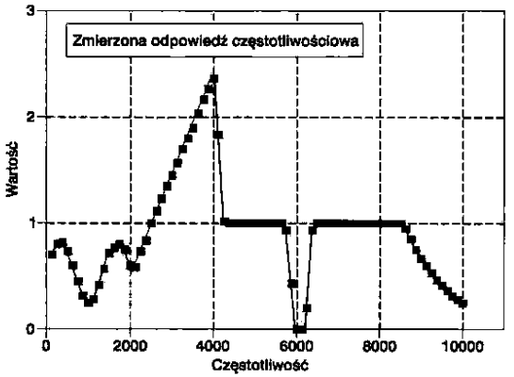
Zapomnijmy na chwilę, że zajmujemy się tu techniką *cyfrową*. Spójrzmy na problem z punktu widzenia inżyniera specjalizującego się w technice *analogowej*. Jego nie obchodzi, co znajduje się wewnątrz EZ-KIT Lite, a tylko to, że zestaw ma analogowe wejście i wyjście. Mógłby nawet sięgnąć po tradycyjną w technice analogowej metodę analizy „czarnej skrzynki", doprowadzając do wejścia sygnał z generatora i obserwując wyjście na oscyloskopie (rys. 29.3).

I do jakich wniosków dojdzie nasz analogowy guru? Najpierw stwierdzi, że system jest *liniowy* (przynajmniej w tym stopniu, w jakim to można wykazać za pomocą prostego testu). Jeśli na wejście wprowadzimy falę sinusoidalną, to taką falę zaobserwujemy też na wyjściu. Jeśli amplituda lub częstotliwość sygnału wejściowego ulega zmianie, to odpowiednią zmianę zobaczymy na wyjściu. Powoli zwiększając częstotliwość sygnału wejściowego, dojdziemy do punktu, w którym amplituda wyjściowej fali sinusoidalnej gwałtownie zmniejszy się do zera. To następuje nieco poniżej połowy częstotliwości próbkowania w wyniku zadziałania filtru antyaliasingowego w przetworniku a/c.

Następnie inżynier, o którym mówimy, zauważy rzecz w świecie analogowym niezwykłą – system ma doskonale *liniową charakterystykę fazy*. Inaczej mówiąc, jest stałe opóźnienie między zdarzeniem zachodzącym w sygnale wejściowym a wynikiem tego zdarzenia w sygnale wyjściowym. Rozważmy na przykład odpowiedź impulsową filtru z rys. 29.2. Ponieważ środek symetrii jest na próbce 150, to sygnał wyjściowy będzie opóźniony o 150 próbek w stosunku do sygnału wejściowego. Jeśli system próbkuje z częstotliwością na przykład 8 kHz, to opóźnienie wyniesie 18,75 ms. Ponadto, niewielkie stałe opóźnienie doda jeszcze przetwornik sigma-delta.



Rys. 29.3. Testowanie zestawu próbnego EZ-KIT Lite. Inżynierowie-analogowcy sprawdzają działanie systemu, doprowadzając do jego wejścia sygnał z generatora, a na wyjściu dołączając oscyloskop. Testowany w ten sposób system DSP (taki jak np. zestaw EZ-KIT Lite) sprawia wrażenie doskonałego systemu analogowego



Rys. 29.4. Zmierzona odpowiedź częstotliwościowa. Na wykresie przedstawiono punkty wyrażające zmierzone wartości odpowiedzi częstotliwościowej przykładowego filtru SOI. Te zmierzone punkty mają dokładność znacznie mniejszą niż zaprojektowana odpowiedź częstotliwościowa na rys. 29.2 a)

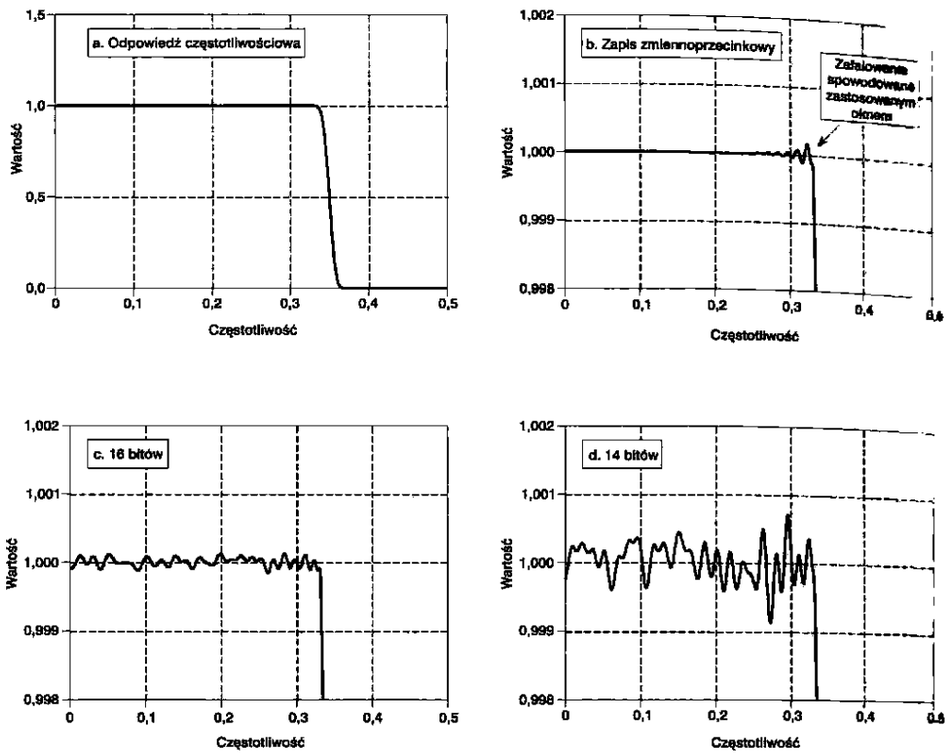
Nasz inżynier będzie bardzo poruszony, widząc tę liniowość fazy. Sygnały pojawiają się nie w taki sposób, jak się spodziewał, i inżynier zacznie gorączkowo kręcić gałkami. Będzie narzekał, że wyzwalanie nie działa dobrze i mamrotał pod nosem, że „to nie ma sensu", „co tu się dzieje?" i „kto się bawił moim oscyloskopem?". Działanie sytemu DSP jest tak dobre, że inżynier dopiero po paru minutach zrozumie, co naprawdę ogląda.

Aby zrobić jeszcze większe wrażenie, poprośmy naszego inżyniera, żeby w sposób ręczny zmierzył charakterystykę systemu. W tym celu będzie ustawiał na generatorze kolejno wszystkie częstotliwości między 125 Hz a 10 kHz w skokach po 125 Hz. Przy każdej częstotliwości zmierzy wartość sygnału wyjściowego i podzieli ją przez wartość sygnału wejściowego. (Oczywiście najłatwiejszym sposobem będzie utrzymywanie stałej wartości sygnału wejściowego). Do tego testu ustawimy częstotliwość próbkowania w EZ-KIT Lite na wartość 22 kHz. Innymi słowy, zakres cyfrowy częstotliwości od 0 do 0,5 z rys. 29.2 a) jest odwzorowywany jako zakres od 0 do 11 kHz w pomiarach w świecie rzeczywistym.

Na rys. 29.4 pokazano wynik pomiaru wykonanego przy użyciu zestawu EZ-KIT Lite. Trudno sobie wyobrazić lepszy wynik! Punkty pomiarowe zgadzają się z krzywą teoretyczną w granicach błędu pomiaru. Z czymś takim inżynier analogowy, o którym mówimy, *nigdy* się nie spotkał w filtrach wykonanych z rezystorów, kondensatorów i cewek indukcyjnych.

Jednak nawet to nie daje systemom DSP tak dobrej opinii, na jaką zasługują. Pomiary analogowe przy użyciu oscyloskopów i woltomierzy cyfrowych charakteryzują się typową rozdzielczością i dokładnością w przybliżeniu od 0,1 do 1%. Dokładność systemu DSP zaś jest ograniczona tylko błędem zaokrąglania ~0,001% w 16-bitowym koderze-dekoderze, gdyż operacje wewnętrzne są wykonywane na liczbach zmiennoprzecinkowych. Tak więc oceniane przez nas urządzenie jest *sto razy* dokładniejsze niż narzędzia pomiarowe użyte do jego testowania. Właściwa ocena charakterystyki częstotliwościowej wymagałaby specjalizowanej aparatury, takiej jak komputerowy system zbierania danych z 20-bitowym przetwornikiem a/c. Znając te fakty, nie dziwimy się, że cyfrowe procesory sygnałowe są często stosowane w aparaturze pomiarowej po to, aby uzyskać dużą dokładność pomiarów.

Teraz możemy już odpowiedzieć na pytanie: Dlaczego ***Analog** Devices* sprzedaje *cyfrowe procesory sygnałowe*? Zaledwie kilkanaście lat temu ówczesne przetwarzanie sygnałów było realizowane przy zastosowaniu dokładnych wzmacniaczy operacyjnych i podobnych do nich układów tranzystorowych. Teraz przetwarzanie *analogowe* najwyższej klasy realizuje się, stosując technikę *cyfrową*. Analog Devices może być przykładem dla innych i dla indywidualnych specjalistów – trzymajcie się swojej wizji i celów i nie obawiajcie się przystosowania do zmieniającej się techniki!



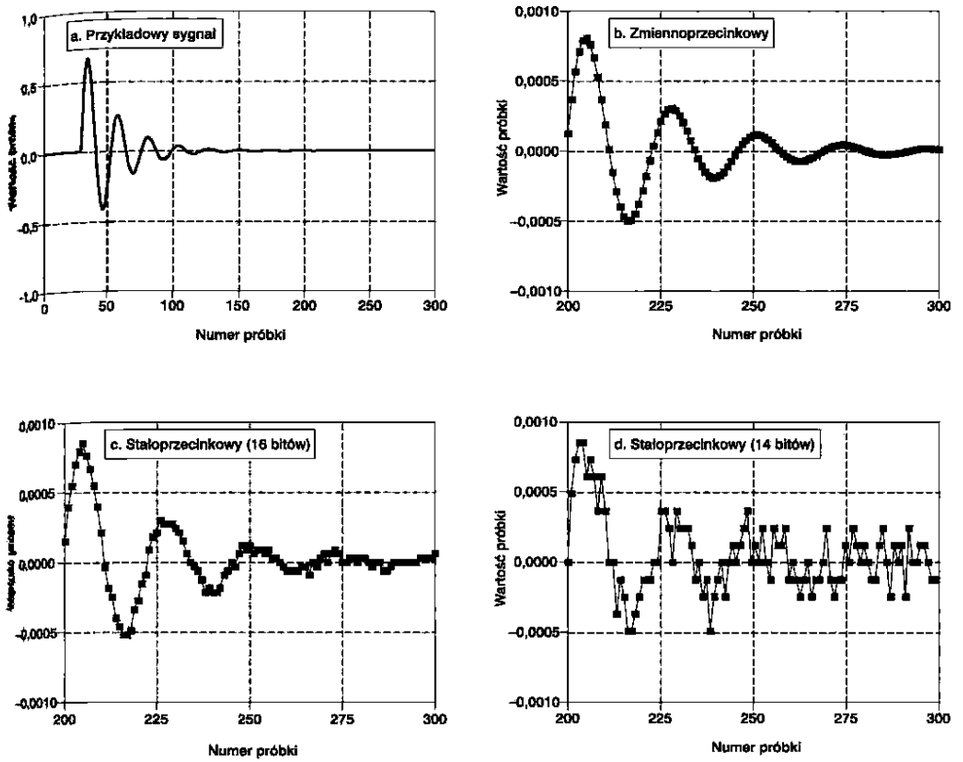
Rys. 29.5. Szum w odpowiedzi częstotliwościowej spowodowany błędami zaokrąglania. Na rys. a) pokazano odpowiedź częstotliwościową filtru dolnoprzepustowego okienkowanego funkcją sinc, z użyciem okna Blackmana i 150-punktowej odpowiedzi impulsowej filtru. Na rys. b), c), d) pokazano wykresy tej odpowiedzi bardziej szczegółowo, rozciągając skalę pionową. Gdy odpowiedź impulsowa filtru jest wyrażona w formacie zmiennoprzecinkowym (rys. b), to szum zaokrąglania jest pomijalny w stosunku do niedoskonałości okienkowania funkcją sinc. Jak wynika z rys. c), d), odnoszących się do stałoprzecinkowej odpowiedzi impulsowej filtru, szum zaokrąglania jest w tym przypadku niedoskonałością dominującą

Inne spojrzenie na systemy stało- i zmiennoprzecinkowe

W tym ostatnim przykładzie wykorzystamy jedną z głównych cech procesorów sygnałowych SHARC, jaką jest zdolność wykonywania obliczeń zmiennoprzecinkowych. Chociaż próbki pobierane z kodera-dekodera i do niego przekazywane są w formacie stałoprzecinkowym, to zadajemy sobie trud przekształcenia ich na format zmiennoprzecinkowy, z którego skorzystamy w pośrednim algorytmie filtracji SOI. Zgodnie z rozważaniami z poprzedniego rozdziału, istnieją dwie przyczyny, dla których jest wskazane przetwarzanie danych przy zastosowaniu operacji matematycznych zmiennoprzecinkowych: *łatwość programowania* i lepsze *działanie algorytmu*. Czy to rzeczywiście czyni jakąś różnicę?

Dla programisty różnica jest bardzo duża. Program zmiennoprzecinkowy jest znacznie łatwiejszy do napisania. Spójrzmy znowu na program asemblerowy w tablicy 29.2. Główna filtracja SOI zawiera tam tylko dwa wiersze (41 i 42). Natomiast programista stosujący liczby stałoprzecinkowe musi dodać instrukcje zarządzające danymi podczas każdej operacji matematycznej. Aby uniknąć przepełnienia lub niedomiaru, wartości muszą być sprawdzane pod względem rozmiaru i – w razie potrzeby – odpowiednio przeskalowane. Wyniki pośrednie powinny być przechowywane w akumulatorze o rozszerzonej dokładności w celu uniknięcia fatalnych skutków wywołanych powtarzającymi się błędami zaokrąglenia.

Sprawa jakości działania jest znacznie bardziej delikatna. Na rys. 29.5 a) jako przykład przedstawiono filtr dolnoprzepustowy SOI o umiarkowanie stromym odcięciu, taki jak opisany w rozdziale 16. Ta krzywa wykreślona „w dużej skali" wygląda tak samo niezależnie od tego, czy w obliczeniach stosowano format stało- czy zmiennoprzecinkowy. Aby zauważyć różnicę między dwoma sposobami obliczeń, trzeba rozciągnąć wykres w skali pionowej o czynnik kilkaset, jak pokazano na rys. 29.5 b), c), d). Tu widzimy wyraźną różnicę. Realizacja zmiennoprzecinkowa (rys. 29.5 b) ma tak mały szum wynikający z błędów zaokrągleń, że działanie tej metody jest ograniczone tylko przez sposób, w jaki zaprojektowaliśmy odpowiedź impulsową filtru. Przerzut 0,02% obok odcięcia jest właściwością okna Blackmana zastosowanego w filtrze. Istota problemu jest taka, że jeśli chcemy ulepszyć działanie, to



Rys. 29.6. Szum zaokrąglania w dziedzinie czasu. Na rys. a) przedstawiono przykład sygnału o amplitudzie zmniejszającej się wykładniczo. Na rys. b), c), d) pokazano wykresy bardziej szczegółowo przez rozciągnięcie skali pionowej. Gdy sygnał jest wyrażony w formacie zmiennoprzecinkowym (rys. b), szum zaokrąglania jest tak mały, że nie można go zobaczyć na wykresie. Na rys. c), d) dotyczących sygnału w formacie stałoprzecinkowym poziom tego szumu jest znacznie wyższy

musimy popracować nad *algorytmem*, a nie nad jego *implementacją*. Na wykresach na rys. 29.5 c), d) jest widoczny szum zaokrąglania wprowadzony w przypadku, gdy każdy punkt w odpowiedzi impulsowej filtru jest wyrażany przez wartość 16-bitową lub przez 14-bitową. Lepszy algorytm nie poprawiłby tych wykresów w żaden sposób; kształt odpowiedzi impulsowej jest bowiem zepsuty przez szum.

Na rys. 29.6 pokazano, jakie różnice między operacjami zmienno- i stałoprzecinkowymi występują *w dziedzinie czasu*. Na rys. 26.9 a) widzimy oscylacyjny sygnał o amplitudzie malejącej wykładniczo. Może on wyrażać na przykład falę dźwiękową z szarpniętej struny albo sygnał wstrząsów ziemi pochodzący z odległego wybuchu. Tak jak poprzednio, przebieg w dużej skali będzie wyglądał jednakowo, niezależnie od tego, czy w reprezentacji próbek zastosowano format stało- czy zmiennoprzecinkowy. Aby zobaczyć różnicę, trzeba rozciągnąć skalę pionową (rys. 29.6 b, c, d). Jak już powiedziano w rozdziale 3, kwantyzacja przejawia się tu jako addytywny szum losowy ograniczający wykrywalność małych składników sygnału.

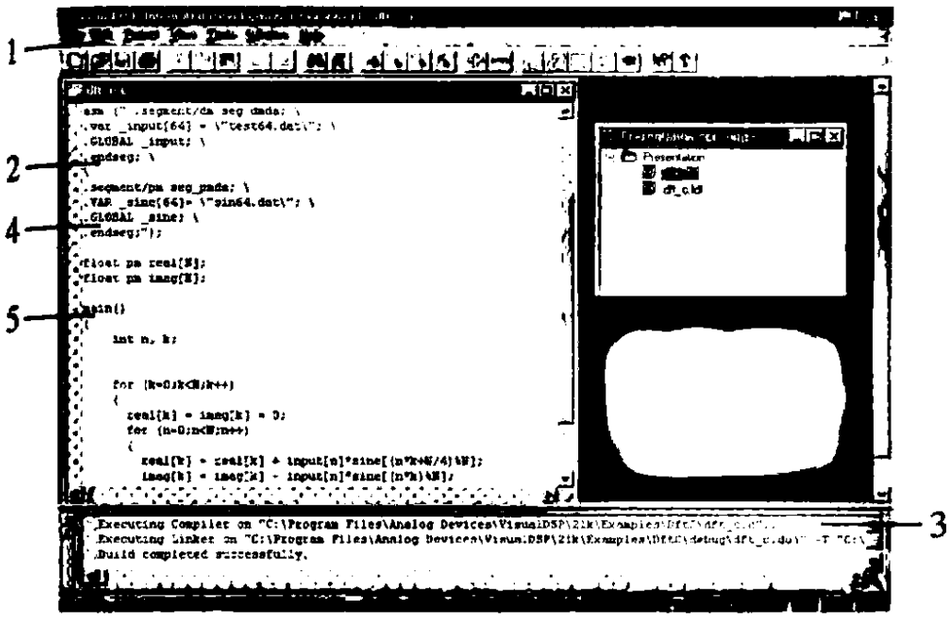
Takie różnice w działaniu między operacjami stało- i zmiennoprzecinkowymi są często nieistotne. Mogą nawet pozostać niezauważone w pokazanych w dużej skali sygnałach z rys. 29.5 a i 29.6 a. Są jednak takie zastosowania, gdzie dodatkowe zalety pracy zmiennoprzecinkowej są pomocne i nawet mogą mieć znaczenie krytyczne. Na przykład w systemach audio hi-fi, takich jak odtwarzacz CD, próbki sygnału są wyrażane w formacie stałoprzecinkowym z rozdzielczością 16-bitową. W większości przypadków taka jakość znacznie przekracza możliwości ludzkiego słuchu. Jednakże w najlepszych profesjonalnych systemach fonicznych sygnały są próbkowane z rozdzielczością od 20 do 24 bitów, wykluczając absolutnie wszelkie artefakty, które mogłyby zakłócić muzykę. W algorytmach przetwarzających takie sygnały cyfrowe o dużej dokładności prawie idealna jest technika zmiennoprzecinkowa.

Lepsze działanie systemu zmiennoprzecinkowego jest też potrzebne w *algorytmach* szczególnie wrażliwych na szumy. Filtry SOI, na przykład, są prawie niewrażliwe na wpływ zaokrągleń. Jak pokazano na rys. 29.5, szum spowodowany błędem zaokrągleń nie zmienia ogólnego kształtu odpowiedzi częstotliwościowej; krzywa staje się tylko bardziej zaszumiona. Inna historia jest z filtrami NOI, gdzie zaokrąglanie może powodować wszelkiego rodzaju spustoszenia, z niestabilnością filtru włącznie. Technika zmiennoprzecinkowa umożliwia lepsze działanie tych algorytmów pod względem ostrości odcięcia charakterystyki, tłumienia w paśmie zaporowym i przerzutu odpowiedzi skokowej.

Zaawansowane narzędzia programowe

Nasz przykład filtru realizowanego według specjalnych wymagań wskazuje *najłatwiejszą* drogę do uzyskania programu działającego na cyfrowych procesorach sygnałowych SHARC: edycja, asemblacja, łączenie i ładowanie wykonywane przez indywidualne programy. Ta metoda jest dobra do prostych zadań, lecz istnieją lepsze narzędzia dla programistów o większych wymaganiach. Spójrzmy więc, jakie oprogramowanie jest dostępne, gdy naprawdę poważnie weźmiecie się do DSP.

Pierwszym narzędziem, które chcemy omówić, jest **kompilator C**. Jak powiedziano w poprzednim rozdziale, zarówno asembler, jak i język C są na ogół stosowane do programowania DSP. Ogromną zaletą stosowania języka C jest biblioteka funkcji oraz algorytmów DSP. W tablicy 29.3 przedstawiono częściową listę biblioteki funkcji języka C do cyfrowych procesorów sygnałowych SHARC. Grupa funkcji matematycznych obejmuje wiele funkcji często stosowanych w DSP, takich jak funkcje trygonometryczne (*sin*, *cos*, *tg* itd.), logarytmiczne i wykładnicze. Jeśli tego typu funkcje są



1. Łatwe poruszanie się między działaniami związanymi z edycją, budową programu i debugowaniem
2. Mieszanie asemblera i C dla stworzenia wspólnego pliku źródłowego
3. Obejrzenie rezultatów „budowy"
4. Silny edytor rozumie składnię
5. Łatwy dostęp przez zakładki

Rys. 29.7. Przykładowy ekran VisualDSP. Daje on zintegrowane środowisko tworzenia aplikacji do programów na procesory SHARC. Wszystkie następujące funkcje są dostępne z tego pojedynczego interfejsu: edycja, kompilacja, łączenie, symulacja, debugowanie, ładowanie, tworzenie PROM

potrzebne w waszym programie, już samo to będzie chyba wystarczającą motywacją do zastosowania języka C zamiast asemblera. Szczególną uwagę zwróćcie na procedury „przetwarzanie sygnałów" w tablicy 29.3. Tam znajdziecie kluczowe algorytmy DSP, w tym rzeczywiste i zespolone FFT, filtry SOI i NOI i funkcje statystyczne, jak np. wartości średnia i skuteczna oraz wariancja. Wszystkie te procedury są oczywiście napisane w asemblerze, dzięki czemu są bardzo efektywne pod względem wykorzystania tak szybkości, jak i pamięci.

W tablicy 29.4 podano przykład programu w języku C zaczerpnięty z „C Compiler and Reference Manual" firmy Analog Devices. Ten program generuje echo przez dodawanie do sygnału jego opóźnionej wersji. Ostatnie poprzednie 4000 próbek sygnału wejściowego jest gromadzone w buforze cyklicznym. Za każdym pobraniem próbki stan bufora cyklicznego jest uaktualniany, najnowsza próbka jest dodawana do przeskalowanej wersji najstarszej próbki i powstała w ten sposób próbka jest kierowana na wyjście.

Tablica 29.3. Funkcje biblioteki języka C. Jest to częściowa lista funkcji dostępnych, gdy język C jest stosowany do programowania cyfrowych procesorów sygnałowych SHARC firmy Analog Devices

abs	absolute value	wartość bezwzględna
acos	arc cosine	arcus cosinus
asin	arc sine	arcus sinus
atan	arc tangent	arcus tangens
atan2	arc tang of quotient	arcus tangens ilorazu
cabsf	complex absolute value	zespolona wartość bezwzględna
cexpf	complex exponential	zespolona funkcja wykładnicza
cos	cosine	cosinus
cosh	hyperbolic cosine	cosinus hiperboliczny
cot	cotangent	cotangens
div	division	dzielenie
exp	exponential	funkcja wykładnicza
fmod	modulus	moduł
log	natural logarithm	logarytm naturalny
log10	base 10 logarithm	logarytm o podstawie 10
matadd	matrix addition	dodawanie macierzowe
matmul	matrix multiplication	mnożenie macierzowe
pow	raise to a power	podnoszenie do potęgi
rand	random number generator	generator liczb losowych
sin	sine	sinus
sinh	hyperbolic sine	sinus hiperboliczny
sqrt	square root	pierwiastek kwadratowy
srand	random number seed	wartość początkowa w generatorze liczb losowych
tan	tangent	tangens
tanh	hyperbolic tangent	tangens hiperboliczny
abort	abnormalprogram end	nieprawidłowy koniec programu
calloc	allocate/initialize memory	przydzielenie/inicjalizacja pamięci
free	deallocate memory	zwalnianie pamięci
idle	processor idle instruction	jałowa instrukcja procesora
interrupt	define interrupt handling	określenie obsługi przerwania
poll_flag_in	test input flag	sprawdź flagę (wskaźnik stanu) wejściową
set_flag	sets the processors flags	ustaw flagi procesorów
timer_off	disable processor timer	unieruchamianie zegara procesora
timer_on	enable processor timer	zezwolenie dla zegara procesora
timer_set	initialize processor timer	inicjalizacja zegara procesora
atoi	convert string to integer	przetwórz ciąg na liczbę całkowitą
bsearch	binary search of array	binarne szukanie matrycy
isalnum	detect alphanumeric character	wykryj znak alfanumeryczny
isalpha	detect alphabetic character	wykryj znak alfabetyczny

Tablica 29.3. (cd.)

iscntrl	detect control character	wykryj znak sterujący
isdigit	detect decimal digit	wykryj znak dziesiętny
isgraph	detect printable character	wykryj znak drukowalny
islower	detect lowercase character	wykryj znak z dolnego rejestru klawiatury
isprint	detect printable character	wykryj znak drukowalny
ispunct	detect punctuation character	wykryj znak przestankowy
isspace	detect white space character	wykryj znak odstępu
isupper	detect uppercase character	wykryj znak z górnego rejestru klawiatury
isxdigit	detect hexadecimal digit	wykryj cyfrę heksadecymalną (szesnastkową)
memchr	find first occurence of character	wykryj pierwsze wystąpienie znaku
memcpy	copy character	kopiuj znak
strcat	concatenate strings	łączenie ciągów
strcmp	compare strings	porównanie ciągów
strerror	get error message	pobierz informację o błędzie
strlen	string length	długość ciągu
strncmp	compare characters	porównanie znaków
strrchr	find last occurence of character	znajdź ostatnie wystąpienie znaku
strstr	find string within string	znajdź ciąg wewnątrz ciągu
strtok	convert string to tokens	przetwórz ciąg na tokeny
system	sent string to operating system	przesłanie ciągu do systemu operacyjnego
tolower	change uppercase to lowercase	zamień znak z górnego rejestru klawiatury na znak z dolnego rejestru
toupper	change lowercase to uppercase	zamień znak z dolnego rejestru klawiatury na znak z górnego rejestru
a_compress	A-law compressing	kompresja według zasady A
a-espand	A-law expansion	ekspansja według zasady A
autocorr	autocorrelation	autokorelacja
biquad	biquad filter section	bikwadratowa sekcja filtru
cfftN	complex FFT	zespolone FFT
crosscorr	cross-correlation	korelacja wzajemna
fir	FIR filter	filtr SOI
histogram	histogram	histogram
ifftN	inverse complex FFT	odwrotne zespolone FFT
iir	IIR filter	filtr NOI
mean	mean of an array	średnia matrycy
mu_compress	mu law compression	kompresja według zasady μ
mu_expand	mu law expansion	ekspansja według zasady μ
rfftN	real FFT	rzeczywiste FFT
rms	rms value of an array	wartość średnia kwadratowa

Następnym, zaawansowanym narzędziem programowym, z którego warto korzystać, jest **zintegrowane środowisko tworzenia aplikacji**. Jest to nieco fantazyjne określenie oznaczające, że wszystko, co jest potrzebne do programowania i testowania DSP, połączono w jeden, harmonijnie działający pakiet. Analog Devices dostarcza zintegrowane środowisko tworzenia aplikacji w postaci produktu zwanego **VisualDSP®** pracującego pod Windows®)95 i WindowsNT™. Na rys. 29.7 pokazano przykład ekranu głównego zapewniającego „bezszwowy" sposób edycji, budowy i debugowania programów.

Tab. 29.4.

```
/* CIRCBUF.>C */
/* Jest to program generowania echa napisany w języku C na ADSP-21061 EZ-KIT Lite. */
/* Program echa pobiera program talkthru i dodaje system buforowania cyklicznego.*/
/* Bufor cykliczny jest definiowany funkcjami CIRCULAR_BUFFER, BASE oraz LENGTH. */
/* Echo jest realizowane przez dodawanie bieżącej wartości sygnału wejściowego do najstarszej *
/* jego wartości. Opóźnienie echa można modyfikować zmieniając BUFF_LENGTH. */
#include <21020.h>                          /* dla instrukcji idle( ) */
#include <signal.h>                         /* dla instrukcji przerwania */
#include <macros.h>                         /* dla CIRCULAR_BUFFER i funkcji segmentowych
#define BUFF_LENGTH 4000
                                            /* definiuje echo jako 21k DAG1 rejestr il */
CIRCULAR_BUFFER (float,1,echo)              /* wskaźnik DM do bufora cyklicznego */

volatile float in_port segment (hip_reg0);  /* hip_reg0 hip_reg2 są */
volatile float out_port segment (hip_reg2); /* stosowane w pliku struktury */

void process_input (int);
void main (void)

{               /* Uczyń to macierzą o zmiennej długości. Jeśli emulator jest zatrzymany na zajm*
                /* i BUFFLENGTH w oknie dm jest zmodyfikowane, to opóźnienie echa zostaje */
                /* zmodyfikowane. Nie róbcie BUFF_LENGTH większej niż rozmiar stosu! */
float data_buff[BUFF_LENGTH];
interrupt (SIG_IRQ3, process.input);

BASE (echo) = data_buff;                    /* Ładuj buff start adr do b1 i il */
LENGTH (echo) = BUFF_LENGTH;                /* Ładuj długość bufora do L1 */

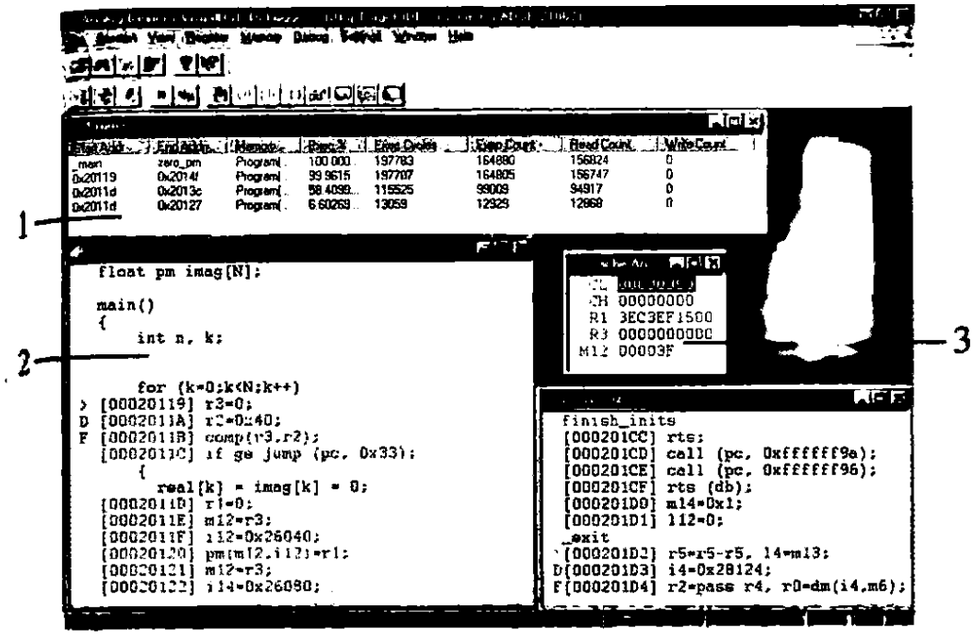
                /* Gdy macierz jest zapełniona, n-ta lokacja zawiera wartość najnowszą, */
                /* a (n+1)lokacja - najstarszą */

while (1)
   {                                        /* echo wysyła sumę ostatnich poprzednich wartości *
    float oldest, newest;                   /* i najstarszej wartości*/
    idle();

                /* Echo wskazuje na n-tą lokację po podprogramie przerwań*/
                /* Umieść nową wartość w zmiennej "newest'. Następnie uaktualnij wskaźnik */
                /* o jeden, aby wskazywał lokacje n+1 */
    CIRC_READ (echo, 1, newest, dm);

                /* Teraz echo wskazuje na n+1. Odczytaj lokację i umieść wartość w zmiennej */
                /* "oldest". Nie aktualizuj wskaźnika, gdyż wskazuje on teraz na nową lokację *
                /* dla obsługi przerwań. */
    CIR_READ (echo, 0, oldest, dm);

                /* dodaj najstarszą wartość i ostatnią poprzednią i wyślij do portu */
    out_port = oldest + newest;
    }
}
void process_input (int int_number)
{
                /* Najnowsza wartość wejściowa jest wpisywana w miejsce najstarszej */
                /* wartości w n-tej lokacji i wskaźnik nie jest uaktualniany */
    CIRC_WRITE (echo, 0, in_port, dm);
}
```

1. **Kod parametrów użytkownika (profilowy) do identyfikacji „wąskich gardeł"**
2. **Oglądanie zmieszanych listingów C i asemblera**
3. **Stworzenie żądanego okna rejestrów**

Rys. 29.8. Ekran debugowania Visual DSP. Jest to zwykły interfejs tak do symulacji, jak i do emulacji. Może pokazywać program w języku C przeplatany uzyskanym kodem asemblera, śledzić wykonywanie instrukcji, sprawdzać rejestry (sprzętowe i programowe oraz pamięci), śledzić funkcjonowanie magistral i realizować wiele innych zadań

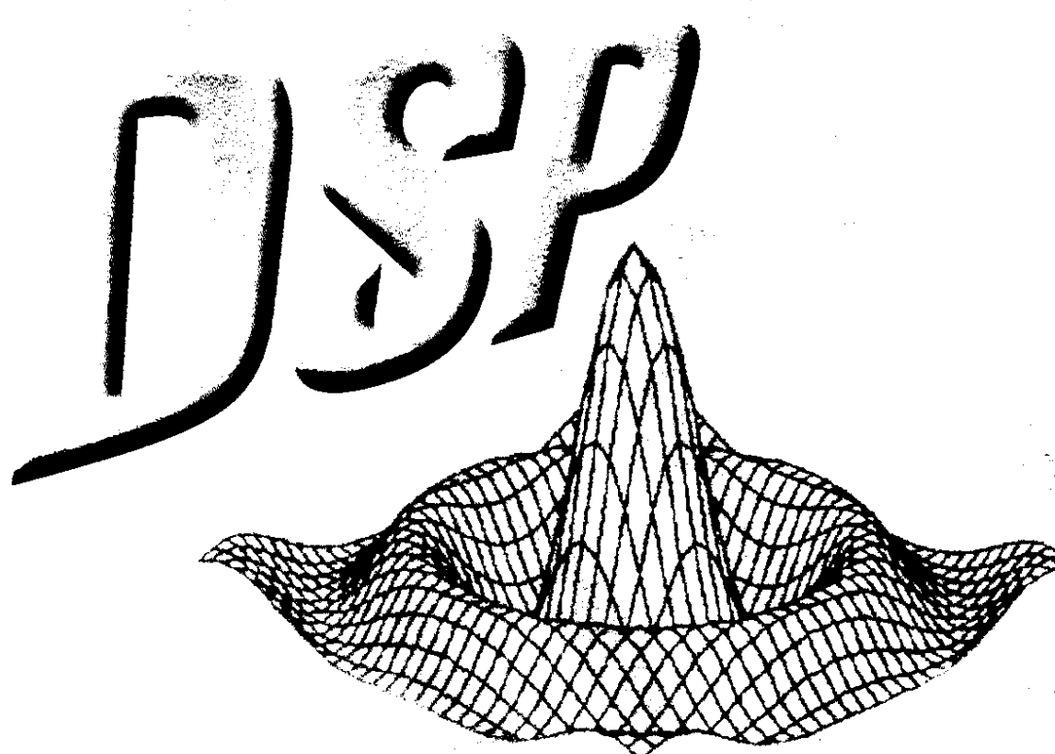
A oto kilka podstawowych cech VisualDSP, z których wynika, dlaczego zintegrowane środowisko tworzenia aplikacji jest tak ważne dla szybkiego opracowania programu. Edytor jest wyspecjalizowany w tworzeniu programów w języku C, w asemblerze i w mieszance obu tych języków. Edytor rozumie, na przykład, *składnię* tych języków, umożliwiając wyświetlanie różnych rodzajów instrukcji w różnych kolorach. Można też edytować więcej niż jeden plik za jednym razem. Jest to bardzo dogodne, gdyż docelowy program jest tworzony przez łączenie razem kilku plików.

Na rys. 29.8 pokazano przykład ekranu z debugera Visual DSP. Jest to interfejs do dwóch różnych typów narzędzi: symulatorów i emulatorów. **Symulatory** testują program *w ramach komputera osobistego*; obecność DSP nie jest konieczna. Jest to zwykle pierwszy, po napisaniu programu, etap debugowania. Symulator naśladuje architekturę i działanie części układowej DSP, włącznie ze strumieniami danych wejściowych, przerwaniami i innymi operacjami we/wy. **Emulatory** (takie jak Analog Devices EZ-ICE®) sprawdzają działanie programu *na istniejącej części układowej DSP*. Program emulatora (na waszym komputerze osobistym) musi więc być zdolny do monitorowania sygnałów elektrycznych wewnątrz DSP. Aby to umożliwić, procesory SHARC mają testowy port dostępu IEEE 1140.1 JTAG, który umożliwia urządzeniu zewnętrznemu śledzenie wewnętrznych funkcji procesora.

Gdy już popracowaliście z zestawem próbnym i przemyśleliście kwestię zakupu zaawansowanych narzędzi programowych, to powinniście rozważyć ewentualne wzięcie udziału w kursie szkoleniowym. Wielu producentów DSP organizuje takie kursy. Na przykład Analog Devices oferuje 3-dniowe szkolenia, przeprowadzane kilka razy do roku, w kilku różnych miejscach. Jest to świetny sposób nauczenia się DSP od ekspertów. Szczegóły znajdziecie na internetowych stronach producentów.

30

Liczby zespolone



Liczby zespolone są rozszerzeniem zwykłych liczb stosowanych w codziennej matematyce. Mają one unikatową właściwość reprezentowania *dwóch* zmiennych i operowania nimi jako *pojedynczą* wielkością. To w bardzo naturalny sposób świetnie pasuje do analizy Fouriera, w której sygnał w dziedzinie częstotliwości składa się z dwóch sygnałów – części rzeczywistej i części urojonej. Liczby zespolone dają uproszczenie wzorów stosowanych w DSP i umożliwiają stosowanie metod, które są bardzo trudne lub niemożliwe do realizacji przy użyciu tylko liczb rzeczywistych. Na przykład szybkie przekształcenie Fouriera jest oparte na liczbach zespolonych. Niestety metody na liczbach zespolonych są trudniejsze pod względem matematycznym. Dlatego trzeba sporo nauki i praktyki, aby można efektywnie z nich korzystać. Wielu naukowców i inżynierów uważa, że metody na liczbach zespolonych są granicą dzielącą DSP traktowane jako *narzędzie* od DSP traktowanego jako dziedzina *kariery zawodowej*. W tym rozdziale zajmiemy się matematyką liczb zespolonych i podstawowymi sposobami stosowania ich w nauce i technice. W następnych trzech rozdziałach omówimy trzy ważne techniki, w których korzysta się z liczb zespolonych: *zespolone przekształcenie Fouriera*, *przekształcenie Laplace'a* oraz *przekształcenie* Z. Te przekształcenia zespolone są sercem teoretycznego DSP. Przygotujcie się, teraz nachodzi matematyka!

System liczb zespolonych

Jako ilustrację liczb zespolonych wyobraźmy sobie dziecko podrzucające w górę piłkę. Przypuśćmy na przykład, że piłka została podrzucona prosto w górę z prędkością 9,8 m/s. W sekundę po opuszczeniu dłoni dziecka piłka osiągnęła wysokość 4,9 m i przyśpieszenie ziemskie (9,8 m/s^2) zredukowało jej prędkość do zera. Następnie piłka ruchem przyśpieszonym spada ku ziemi, aby po 2 sekundach od chwili wyrzucenie zostać złapaną przez dziecko. Z podstawowych zależności fizycznych wynika, że wysokość położenia piłki w dowolnej chwili jest określona wzorem:

$$h=\frac{-gt^2}{2}+vt$$

gdzie h jest wysokością ponad powierzchnię Ziemi (w metrach), g – przyśpieszeniem ziemskim (9,8 m/s^2), v – prędkością początkową (w naszym przykładzie 9,8 m/s) a t – czasem (w sekundach).

Teraz przyjmijmy, że chcemy wiedzieć *kiedy* piłka przekroczy pewną określoną wysokość. Wstawiając znane wartości i przekształcając powyższy wzór, uzyskujemy:

$$t=1\pm\sqrt{1-\frac{h}{4,9}}$$

Na wysokości na przykład 3 metrów piłka jest *dwukrotnie*, *dla* t = 0,38 s (przy wznoszeniu) oraz t = 1,62 s (przy opadaniu).

Dopóki zadajemy sensowne pytania, dopóty powyższe wzory dają sensowne odpowiedzi. A co się stanie, gdy zadamy nierozsądne pytania? Na przykład: po jakim czasie piłka osiągnie wysokość 10 metrów? To pytanie nie ma żadnej odpowiedzi w dziedzinie rzeczywistej, gdyż piłka *nigdy* nie osiągnie tej wysokości. Niemniej jednak wstawienie wartości h = 10 m do powyższego wzoru daje dwie odpowiedzi: $t=1+\sqrt{-1,041}$ oraz $t=1-\sqrt{-1,041}$. Oba te rozwiązania zawierają pierwiastek kwadratowy z liczby ujemnej, a więc coś nieistniejącego w znanym nam świecie. Tę niezwykłą właściwość równań wielomianowych jako pierwszy zastosował włoski matematyk Girolamo Cardano (1501–1576). Dwa stulecie później wielki matematyk niemiecki Carl Friedrich Gauss (1777–1855) wprowadził pojęcie **liczb zespolonych** i utworzył drogę do współczesnego zrozumienia tej dziedziny.

Każda liczba zespolona jest sumą dwóch składowych: **części rzeczywistej** oraz **części urojonej**. Część rzeczywista jest **liczbą rzeczywistą**, jedną z tych zwykłych liczb, o których wszyscy uczyliśmy się od dzieciństwa. Część urojona jest **liczbą urojoną**, czyli *pierwiastkiem kwadratowym z liczby ujemnej*. W celu znormalizowania zapisu wyraża się liczbę urojoną jako liczbę rzeczywistą pomnożoną przez pierwiastek kwadratowy z minus jeden. Na przykład liczba zespolona $t=1+\sqrt{-1,041}$ jest najpierw przekształcana do postaci $t=1+\sqrt{1,041}\sqrt{-1}$ i następnie do postaci końcowej $t = 1 + 1,02\sqrt{-1}$. Część rzeczywista tej liczby zespolonej jest więc równa 1, a urojona $1,02\sqrt{-1}$. Taki sposób zapisu daje możliwość nadania specjalnego symbolu abstrakcyjnemu czynnikowi $\sqrt{-1}$. Matematycy przez długi czas stosowali symbol i na oznaczenie $\sqrt{-1}$. Inżynierowie elektrycy zaś stosują symbol j, ponieważ i jest stosowane jako oznaczenie prądu elektrycznego. W DSP są stosowane oba te symbole. W tej książce przyjęto konwencję inżynierów elektryków, a więc symbol j.

Na przykład wszystkie następujące liczby są liczbami zespolonymi: $1 + 2j$, $1 - 2j$, $-1 + 2j$, $3,14159 + 2,7183j$, $(4/3) + (19/2)j$ itd. Wszystkie zwykłe liczby, takie jak 2; 6,34 oraz –1,414 można traktować jako liczby zespolone z częścią urojoną równą *zero*, tj. $2 + 0j$, $6,34 + 0j$, oraz $-1,414 + 0j$.

Liczby rzeczywiste mogą być opisywane ich położeniami na osi liczbowej, liczby zespolone zaś są reprezentowane położeniami na płaszczyźnie dwuwymiarowej zwanej **płaszczyzną zespoloną**. Oś pozioma tej płaszczyzny (rys. 30.1) jest osią rzeczywistą liczb zespolonych, a pionowa – osią urojoną. Ponieważ liczby rzeczywiste są takimi liczbami zespolonymi, które mają część urojoną równą zero, więc *oś liczb rzeczywistych* jest tym samym, co *oś x* na płaszczyźnie zespolonej.

We wzorach matematycznych liczba zespolona jest wyrażana jako jedna zmienna, mimo że składa się z dwóch części. Na przykład trzy liczby zmienne z rys. 30.1 można zapisać jako:

$$A = 2 + 6j$$
$$B = -4 - 1,5j$$
$$C = 3 - 7j$$

gdzie A, B, C są zmiennymi zespolonymi. To ilustruje zarówno istotną *zaletę*, jak i istotną *wadę* stosowania liczb zespolonych. Zaletą jest możliwość uproszczonego zapisu dwóch wartości jednym symbolem. Wadą jest konieczność pamiętania, które zmienne są zespolone, a które są zwykłymi liczbami.

W matematycznym zapisie rozdzielającym liczbę zespoloną na jej część rzeczywistą i urojoną stosuje się operatory *Re*( ) i *Im*( ). Dla wyżej podanych przykładów liczb zespolonych mamy więc:

$$Re\ A = 2 \qquad Im\ A = 6$$
$$Re\ B = -4 \qquad Im\ B = -1,5$$
$$Re\ C = 3 \qquad Im\ C = -7$$

Zwróćcie uwagę, że argument operatora matematycznego *Im*( ) nie zawiera j. Na przykład $Im(3 + 4j)$ jest równe 4, a nie $4j$.

W dziedzinie liczb zespolonych obowiązują takie same zasady algebraiczne, jak dla zwykłych liczb, z potraktowaniem wielkości j jako pewnej stałej. Na przykład działania dodawania, odejmowania, mnożenia i dzielenia są opisane równaniami:

$$(a+bj)+(c+dj)=(a+c)+j(b+d)$$

Wzór 30.1. Dodawanie liczb zespolonych

$$(a+bj)-(c+dj)=(a-c)+j(b-d)$$

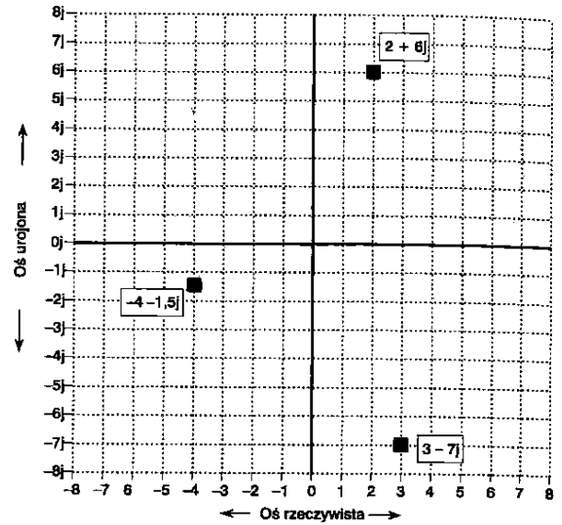
Wzór 30.2. Odejmowanie liczb zespolonych

$$(a+bj)(c+dj)=(ac-bd)+j(bc+ad)$$

Wzór 30.3. Mnożenie liczb zespolonych

$$\frac{(a+bj)}{(c+dj)}=\left(\frac{ac+bd}{c^2+d^2}\right)+j\left(\frac{bc-ad}{c^2+d^2}\right)$$

Wzór 30.4. Dzielenie liczb zespolonych



Rys. 30.1. Płaszczyzna zespolona. Każda liczb zespolona ma jedno i jednoznaczne położenie na płaszczyźnie zespolonej, co zilustrowano trzema przykładami. Oś pozioma jest osią rzeczywistą, pionowa zaś – urojoną

Operując równaniami, takimi jak powyższe, korzysta się z dwóch sztuczek. Po pierwsze, gdziekolwiek pojawia się wyraz j^2, tam zastępuje się go przez –1. Wynika to z definicji j, tzn. $j^2 = (\sqrt{-1})^2 = -1$. Druga sztuczka dotyczy eliminowania wyrazu j z mianownika ułamka. Na przykład lewa strona równania 30.4 ma mianownik $c + dj$. Trzeba licznik i mianownik pomnożyć przez czynnik $(c - jd)$, usuwając w ten sposób z mianownika wszelkie wyrazy urojone. Zmiana znaku części urojonej liczby zespolonej daje liczbę nazywaną **liczbą zespoloną sprzężoną**. Oznacza się ją gwiazdką w prawym, górnym rogu zmiennej. Na przykład jeśli $Z = a + bj$, to $Z^* = a - bj$. Tak więc wzór 30.4 wyprowadzono, mnożąc zarówno licznik, jak i mianownik przez zespoloną liczbę sprzężoną mianownika.

Niżej podane podstawowe zależności algebraiczne są słuszne także wówczas, gdy zmienne A, B, C są liczbami zespolonymi. Te zależności można udowodnić, rozdzielając każdą zmienną na jej część rzeczywistą i urojoną i wykonując działania algebraiczne.

$$AB = BA$$

Wzór 30.5. Zasada przemienności mnożenia

$$(A + B) + C = A + (B + C)$$

Wzór 30.6. Zasada łączności dodawania

$$A(B + C) = AB + AC$$

Wzór 30.7. Zasada rozdzielności mnożenia względem dodawania

Zapis we współrzędnych biegunowych

Liczby zespolone mogą być wyrażane nie tylko we *współrzędnych prostokątnych* (o czym już powiedziano), lecz także we *współrzędnych biegunowych*. Na rys. 30.2 podano trzy przykłady liczb zespolonych w postaci biegunowej. Są to te same liczby, które przedstawiono poprzednio na rys. 30.1. **Moduł** jest długością wektora zaczynającego się w początku układu współrzędnych, a kończącego w punkcie oznaczającym liczbę zespoloną. **Kąt fazowy** jest mierzony między tym wektorem a dodatnią osią x. Przekształcenia liczb zespolonych ze współrzędnych

prostokątnych na biegunowe i na odwrót wykonuje się według następujących wzorów (nie zapominając o niedogodnościach zapisu biegunowego rozpatrywanych w rozdziale 8):

$$M = \sqrt{(Re\,A)^2 + (Im\,A)^2}$$

$$\theta = \text{arctg}\left[\frac{Im\,A}{Re\,A}\right]$$

Wzór 30.8. Konwersja współrzędnych prostokątnych na biegunowe. Zmienna zespolona A może być przekształcona z zapisu we współrzędnych prostokątnych: Re A i Im A na współrzędne biegunowe: M i θ

$$\text{Re}\,A = M\cos(\theta)$$

$$\text{Im}\,A = M\sin(\theta)$$

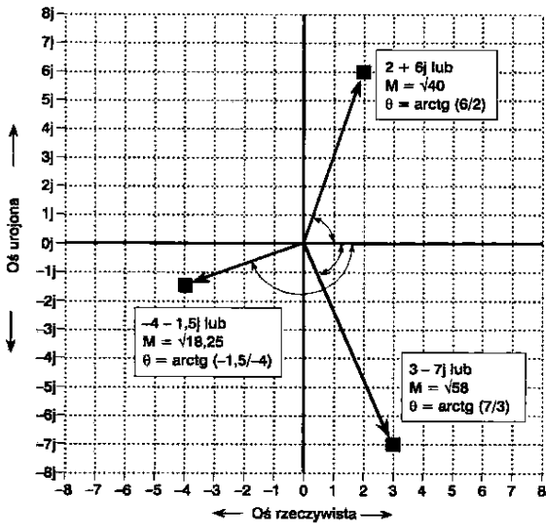
Wzór 30.9. Konwersja współrzędnych biegunowych na prostokątne. Jest to przekształcenie liczby zespolonej ze współrzędnych M, θ do współrzędnych Re A, Im A

I tu mamy ogromny skok w naszych rozważaniach matematycznych (co znaczy, że musicie bardzo wytężyć uwagę). Liczba zespolona zapisana we współrzędnych prostokątnych ma postać: $a + bj$. Informacja jest zawarta w dwóch zmiennych a oraz b, lecz właściwą liczbę zespoloną tworzy całe wyrażenie: $a + bj$. We współrzędnych biegunowych istotna informacja jest zawarta w M oraz θ, lecz jakie jest pełne wyrażenie właściwej liczby zespolonej w tych współrzędnych?

Kluczem do odpowiedzi na to pytanie jest wzór 30.9 opisujący konwersję współrzędnych biegunowych na prostokątne. Jeśli zaczniemy rozważanie od liczby zespolonej $a + bj$ i zastosujemy wzór 30.9, to uzyskamy:

$$a + jb = M(\cos\theta + j\sin\theta)$$

Wzór 30.10. Liczby zespolone we współrzędnych prostokątnych i biegunowych. Lewa strona równania wyraża liczbę zespoloną we współrzędnych prostokątnych, a prawa – w biegunowych. Konwersja między M i θ oraz a i b jest opisana wzorami 30.8 i 30.9



Rys. 30.2. Liczby zespolone we współrzędnych biegunowych. Trzy przykładowe punkty na płaszczyźnie zespolonej opisano we współrzędnych biegunowych. Na rys. 30.1 pokazano te same punkty we współrzędnych prostokątnych

Wyrażenie z lewej strony równania jest właściwym opisem liczby zespolonej we współrzędnych *prostokątnych*, a po prawej – we współrzędnych *biegunowych*.

Zanim przejdziemy do dalszych rozważań, zastanówmy się, jak doszliśmy do obecnego ich punktu. Po pierwsze, nadaliśmy formę graficzną liczbom zespolonym we współrzędnych biegunowych, a więc określiliśmy ich położenie na płaszczyźnie zespolonej. Po drugie, zdefiniowaliśmy pojęcia M i θ tak, aby były zgodne z naszymi poprzednimi wiadomościami o zależnościach między współrzędnymi biegunowymi i prostokątnymi (wzory 30.8 i 30.9). Po trzecie, konsekwentnie wyciągnęliśmy dalsze wnioski z poprzednich działań i przez operacje matematyczne doszliśmy do wyrażenia będącego prawidłową postacią liczby zespolonej we współrzędnych biegunowych, czyli do wyrażenia $M(\cos\theta + j\sin\theta)$. Chociaż całe to rozumowanie jest proste i logiczne, to wynik trudno byłoby przewidzieć intuicyjnie. Niestety, dalej jest jeszcze gorzej z takim przewidywaniem.

Jednym z najważniejszych równań w matematyce liczb zespolonych jest **zależność Eulera**, biorąca swoją nazwę od zdolnego i bardzo twórczego matematyka szwajcarskiego Leonharda Eulera (1707–1783):

$$e^{jx} = \cos x + j\sin x$$

Wzór 30.11. Zależność Eulera. To równanie ma podstawowe znaczenie w zastosowaniach liczb zespolonych w nauce i technice

Jeśli jesteście ciekawi szczegółów, to wiedzcie, że powyższą zależność można wyprowadzić, rozwijając składnik wykładniczy w szereg Taylora:

$$e^{jx} = \sum_{n=0}^{\infty} \frac{(jx)^n}{n!} = \left[\sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}\right] + j\left[\sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}\right]$$

Dwa składniki w nawiasach po prawej stronie tego wyrażenia są rozwinięciami w szereg Taylora funkcji $\cos x$ i $\sin x$. Nie traćcie zbyt wiele czasu na analizowanie tego wyprowadzenia; nie mamy zamiaru go stosować do czegokolwiek.

Przekształcając równanie 30.10 z wykorzystaniem zależności Eulera, uzyskujemy najczęściej stosowaną postać zapisu liczb zespolonych we współrzędnych biegunowych. Jest to **liczba zespolona w postaci wykładniczej**:

$$a + jb = Me^{j\theta}$$

Wzór 30.12. Wykładnicza postać zapisu liczb zespolonych. Liczba zespolona we współrzędnych prostokątnych (lewa strona równania) jest równa liczbie w zapisie wykładniczym (prawa strona)

Liczby zespolone zapisywane w postaci wykładniczej są trzonem rozważań matematycznych w DSP. Opanowanie tych problemów zacznijcie od zapamiętania równań od 30.8 do 30.12. Wielką zaletą zapisu wykładniczego liczb zespolonych jest łatwość mnożenia i dzielenia liczb w tym zapisie:

$$M_1 e^{j\theta_1} \, M_2 e^{j\theta_2} = M_1 M_2 e^{j(\theta_1 + \theta_2)}$$

Wzór 30.13. Mnożenie liczb zespolonych

$$\frac{M_1 e^{j\theta_1}}{M_2 e^{j\theta_2}} = \left[\frac{M_1}{M_2}\right] e^{j(\theta_1 - \theta_2)}$$

Wzór 30.14. Dzielenie liczb zespolonych

Jak wynika z powyższych wzorów, mnożenie liczb zespolonych we współrzędnych biegunowych wykonuje się, mnożąc ich moduły i sumując kąty fazowe. Najłatwiejszym sposobem wykonania dodawania lub odejmowania liczb zespolonych zapisanych we współrzędnych biegunowych jest przekształcenie tych liczb na współrzędne prostokątne, wykonanie żądanej operacji i powrotna konwersja do zapisu biegunowego. W procedurach komputerowych liczby zespolone są zazwyczaj wyrażane we współrzędnych prostokątnych, lecz przy operowaniu równaniami stosuje się współrzędne biegunowe. Tak jak *Re*( ) i *Im*( ) są stosowane do wydzielenia składowych prostokątnych z liczby zespolonej, tak operatory *Mag*( ) oraz *Phase*( ) służą do wydzielenia składowych biegunowych – modułu i przesunięcia fazowego. Na przykład jeśli $A = 5e^{j\pi/7}$, to $Mag(A) = 5$ i $Phase(A) = \pi/7$.

Stosowanie liczb zespolonych metodą zastępowania

Podsumujmy dotychczasowe rozważania. Rozwiązania zwykłych równań algebraicznych często zawierają pierwiastki kwadratowe z liczb ujemnych. Noszą one nazwę *liczb zespolonych* i wyrażają rozwiązania, które nie mogą istnieć w takim świecie, jaki znamy. Liczby zespolone są wyrażane w jednej z dwóch postaci: $a + bj$ (współrzędne prostokątne) lub $Me^{j\theta}$ (współrzędne biegunowe), przy czym j jest symbolem wyrażającym $\sqrt{-1}$. W każdym z tych zapisów jedna liczba zespolona zawiera dwie informacje – albo a oraz b, albo M oraz θ. Mimo swojej dość nieuchwytnej natury, liczby zespolone podlegają prawom matematycznym podobnym (lub identycznym) do praw rządzących zwykłymi liczbami.

W ten sposób opisaliśmy, czym są liczby zespolone i jakie jest ich miejsce w czystej matematyce. Naszym następnym zadaniem jest omówienie sposobów pożytecznego wykorzystania tych liczb w problemach naukowych i technicznych. Jak jest możliwe stosowanie matematyki niemającej żadnego związku z naszym codziennym doświadczeniem? A oto odpowiedź: *Jeśli narzędziem, którym dysponujemy, jest młotek, to spowodujmy, żeby problem miał wygląd gwoździa.* Innymi słowy, *przetwarzamy* problem fizyczny do postaci liczb zespolonych, operujemy na tych liczbach i następnie przez *powrotne przetworzenie* wracamy do rozwiązania problemu fizycznego.

Są dwa sposoby wyrażania problemów fizycznych przy użyciu liczb zespolonych: metoda prostego **zastępowania** oraz bardziej elegancka metoda, którą nazwiemy **równoważnością matematyczną**. W następnym rozdziale, obejmującym *zespolone przekształcenie Fouriera*, będzie omówiona metoda równoważności matematycznej. Pozostała część tego rozdziału jest poświęcona metodzie zastępowania.

W metodzie zastępowania bierze się dwa rzeczywiste parametry fizyczne i umieszcza jeden z nich w części rzeczywistej liczby zespolonej, a drugi – w części urojonej. To umożliwia operowanie tymi dwiema wartościami jako jedną całością, tj. jako jedną liczbą zespoloną. Po wykonaniu żądanych operacji matematycznych rozdziela się liczbę zespoloną na jej części rzeczywistą i urojoną, co znowu odpowiada interesującym nas parametrom fizycznym.

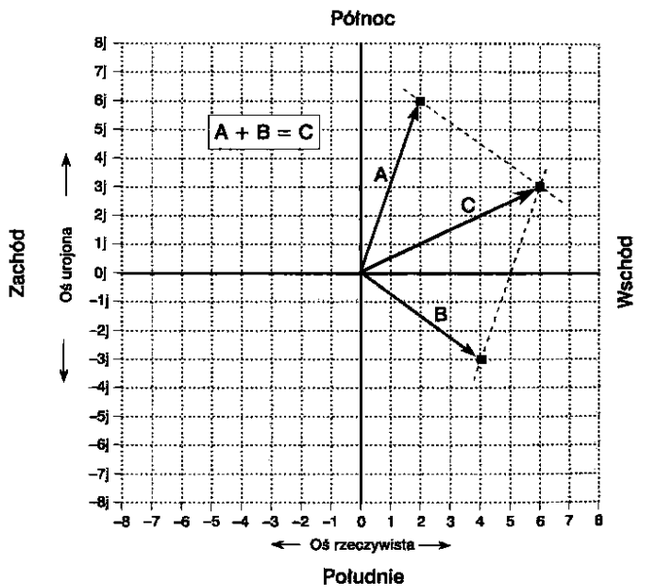
Przedstawimy tę metodę na prostym przykładzie. Jak pamiętacie z podstaw fizyki, *wektory* mogą reprezentować takie wielkości, jak siła, prędkość, przyspieszenie itd. Wyobraźmy sobie żaglówkę pchaną w jednym kierunku przez wiatr, a w drugim – przez prąd morski. Wynikowa siła działająca na łódź jest sumą wektorową dwóch indywidualnych wektorów sił. Ten przykład przedstawiono na rys. 30.3, na którym dwa wektory A i B są sumowane według zasady równoległoboku, dając wektor C.

Możemy to zagadnienie wyrazić w liczbach zespolonych, umieszczając współrzędną wschód/zachód w części rzeczywistej liczby zespolonej, a północ/południe – w części urojonej. W ten sposób możemy każdy wektor traktować jako jedną liczbę zespoloną, chociaż składa się ona z dwóch części. Na przykład siła wiatru, wektor A, może w 2 częściach mieć kierunek wschodni, a w 6 częściach – północny, co reprezentuje liczba zespolona $2 + 6j$. Podobnie siła prądu morskiego, wektor B, może mieć w 4 częściach kierunek wschodni, a w 3 – południowy, co wyraża liczba zespolona $4 - 3j$. Te dwa wektory można zsumować według wzoru 30.1, uzyskując liczbę zespoloną $6 + 3j$ reprezentującą wektor C. Przenosząc ten wynik z powrotem do pojęć świata fizycznego, uzyskuje się kierunek wypadkowej siły działającej na żaglówkę: 6 części w kierunku północnym i 3 – we wschodnim.

Czy można by rozwiązać to zadanie bez liczb zespolonych? Oczywiście tak. Liczby zespolone dają tylko formalną drogę operowania *dwiema* składowymi tworzącymi *jeden* wektor. Trzeba więc pamiętać, że niektóre problemy fizyczne można wyrażać w liczbach zespolonych, po prostu dodając j do jednej ze składowych. Powrót do świata fizycznego jest niczym innym, jak usunięciem tego j. To jest istota metody *zastępowania*.

I tu pojawia się pewna trudność. Skąd mamy wiedzieć, czy reguły i prawa obowiązujące w matematyce liczb zespolonych stosują się też do danego zagadnienia fizycznego? Zastosowaliśmy np. wzór 30.1 do sumowania wektorów sił w zadaniu o żaglówce. Skąd wiemy, że dodawanie liczb zespolonych daje taki sam wynik, jak dodawanie wektorów sił? W większości przypadków wiemy, że matematyka liczb zespolonych może być stosowana do danej aplikacji, gdyż *ktoś inny* twierdzi, że tak jest. Błyskotliwi i cenieni matematycy i inżynierowie pracowali nad szczegółami tego zagadnienia i opublikowali rezultaty. Trzeba pamiętać, że nie możemy zastępować każdego *dowolnego* problemu jego reprezentacją w liczbach zespolonych i spodziewać się sensownej odpowiedzi. Analizę na liczbach zespolonych można stosować tylko w tych zagadnieniach, dla których zasadność użycia tej analizy już wykazano.

Rozważmy teraz taki przykład, w którym *nie działa* metoda zastępowania liczbami zespolonymi. Powiedzmy, że kupujecie jabłka po 5 złotych za skrzynkę i pomarańcze po 10 złotych za skrzynkę. Wyrażacie to liczbą zespoloną $5 + 10j$. Podczas któregoś tygodnia kupujecie 6 skrzynek jabłek i 2 skrzynki pomarańcz, co wyrażacie liczbą zespoloną $6 + 2j$. Całkowita cena, jaką musicie zapłacić za te owoce, jest równa liczbie skrzynek pomnożonej przez cenę każdej z nich, czyli $(5 + 10j)(6 + 2j) = 10 + 70j$. Z działań na liczbach zespolonych wynika więc, że musicie zapłacić za jabłka 10 złotych i za pomarańcze 70 złotych. Otóż ta odpowiedź jest całkiem błędna! Zasady matematyki liczb zespolonych *nie są* zgodne z zasadami tego postawionego zadania.



Rys. 30.3. Sumowanie wektorów z zastosowaniem liczb zespolonych. Wektory A i B reprezentują siły zmierzone w odniesieniu do kierunków wschód/zachód i północ/południe. Wartość na osi wschód/zachód jest zastąpiona przez część rzeczywistą liczby zespolonej, a na osi północ/południe – przez jej część urojoną. Takie zastąpienie umożliwia zastosowanie matematyki liczb zespolonych do problemu całkiem rzeczywistego

Zespolona reprezentacja sinusoid

Liczby zespolone znalazły swoją niszę w elektronice i przetwarzaniu sygnałów, gdyż są zręcznym sposobem reprezentowania i operowania najbardziej użytecznymi ze wszystkich przebiegów – falami sinusoidalną i kosinusoidalną. Konwencjonalny sposób wyrażania sinusoidy to: $M\cos(\omega t + \phi)$ lub $A\cos(\omega t) + B\sin(\omega t)$, odpowiednio – w zapisie we współrzędnych biegunowych i prostokątnych. Zauważcie, że częstotliwość wyrażamy przez *pulsację* ω w *radianach na sekundę*. Jeśli wolicie, to możecie ω zastąpić przez wielkość $2\pi f$ w Hz. Jednak większość wzorów w DSP jest formułowanych z użyciem krótszego zapisu i trzeba się do niego przyzwyczaić. Ponieważ reprezentacja pojedynczej sinusoidy wymaga dwóch parametrów (tj. A i B lub M i ϕ), więc zastosowanie liczb zespolonych do reprezentowania tych ważnych przebiegów jest rzeczą naturalną. Przy użyciu metody zastępowania zamiana konwencjonalnego wyrażania sinusoidy na reprezentację w liczbach zespolonych jest bezpośrednia. We współrzędnych prostokątnych:

$$A\cos(\omega t) + B\sin(\omega t) \quad \rightleftarrows \quad a + jb$$

(reprezentacja konwencjonalna) (liczby zespolone)

gdzie $A \rightleftarrows a$ *oraz* $B \rightleftarrows -b$. Wyrażając to słownie – amplituda fali kosinusoidalnej staje się częścią rzeczywistą liczby zespolonej, a amplituda *ujemnej* fali sinusoidalnej – częścią urojoną. Ważne jest, aby zrozumieć, że *nie jest to* równanie, lecz po prostu wskazanie sposobu, w jaki liczby zespolone *reprezentują* sinusoidę. Zastępowanie można być też stosowane do zapisu biegunowego:

$$M\cos(\omega t + \Phi) \quad \rightleftarrows \quad Me^{j\theta}$$

(reprezentacja konwencjonalna) (liczby zespolone)

gdzie $M \rightleftarrows M$ oraz $\theta \rightleftarrows -\phi$. Wyrażając to słownie – zastępowanie pozostawia moduł bez zmiany, lecz zmienia znak kąta fazowego.

Dlaczego trzeba zmieniać znak części urojonej i kąta fazowego? Chodzi o to, aby zastępowanie miało taką samą formę, jak w *zespolonym przekształceniu Fouriera* omawianym w następnym rozdziale. Rozpatrywana w tym rozdziale metoda *zastępowania* nie zyskuje niczego na tej zmianie znaku, lecz prawie zawsze się ją wprowadza, aby zachować zgodność z metodami matematycznie bardziej zaawansowanymi.

Zastosowanie liczb zespolonych do reprezentowania fal sinusoidalnych i kosinusoidalnych jest metodą powszechnie stosowaną w analizie obwodów elektrycznych i w DSP. Wynika to z faktu, że wiele (choć nie wszystkie) reguł i praw rządzących liczbami zespolonymi jest takich samych, jak te, które rządzą przebiegami sinusoidalnymi. Możemy więc fale sinusoidalne i kosinusoidalne wyrażać w liczbach zespolonych, operować tymi liczbami na różne sposoby i uzyskiwać wyniki pasujące do zachowań przebiegów sinusoidalnych.

Musimy jednak być ostrożni i stosować tylko te operacje matematyczne, które są naśladownictwem reprezentowanych zagadnień fizycznych (w tym przypadku sinusoid). Przypuśćmy na przykład, że stosujemy zmienne zespolone A i B do reprezentowania dwóch sinusoid o takiej samej częstotliwości, lecz o różnych amplitudach i przesunięciach fazy. Przy *dodawaniu* dwóch liczb zespolonych powstaje trzecia liczba zespolona. Podobnie, trzecia sinusoida powstaje przy dodawaniu dwóch sinusoid. Zgodnie z naszym oczekiwaniem ta trzecia liczba zespolona będzie reprezentowała trzecią sinusoidę. Dodawanie zespolone wykazuje więc zgodność z systemem fizycznym.

A teraz wyobraźmy sobie mnożenie liczb zespolonych A i B dające w wyniku inną liczbę zespoloną. Czy to będzie zgodne z tym, co się dzieje przy mnożeniu dwóch sinusoid? Nie! Wymnożenie przez siebie dwóch sinusoid *nie wytwarza* innej sinusoidy. Mnożenie zespolone nie spełnia więc oczekiwań zgodności z systemem fizycznym i wobec tego nie może być stosowane.

Operacje właściwe do reprezentowania przebiegów sinusoidalnych są, na szczęście, jasno zdefiniowane. Muszą być spełnione dwa warunki. Po pierwsze, wszystkie sinusoidy muszą mieć *taką samą częstotliwość*. Na przykład, jeśli liczby zespolone $1 + 1j$ i $2 + 2j$ reprezentują sinusoidy o takiej samej częstotliwości, wtedy suma tych sinusoid jest reprezentowana przez liczbę zespoloną $3 + 3j$. Jeśli jednak liczby $1 + 1j$ i $2 + 2j$ reprezentują sinusoidy o różnych częstotliwościach, to niczego nie da się zrobić z ich reprezentacją zespoloną. W tym przypadku suma liczb zespolonych $3 + 3j$ nie ma żadnego sensu.

Mimo to, przy stosowaniu liczb zespolonych częstotliwość może być pozostawiona jako zmienna, lecz musi to być wszędzie *ta sama* częstotliwość. Jest np. całkiem poprawne dodawanie $2\omega + 3\omega j$ i $3\omega + 1j$ dające w wyniku $5\omega + (3\omega + 1)j$. Te wartości reprezentują sinusoidy, w których amplituda i faza zmieniają się przy zmianie częstotliwości. Chociaż nie wiemy, *jaka* jest ta określona częstotliwość, to wiemy na pewno, że wszędzie jest *taka sama*, tj. równa ω.

Drugie wymaganie mówi o tym, że reprezentowane operacje muszą być *liniowe*, jak to rozważano w rozdziale 5. Na przykład sinusoidy można łączyć ze sobą przez dodawanie lub odejmowanie, ale nie można przez mnożenie lub dzielenie. Podobnie jest z systemami, które mogą być wzmacniaczami, dzielnikami, filtrami górno- lub dolnoprzepustowymi, ale nie układami takimi, jak podnoszące do kwadratu, obcinające lub progowe. Zapamiętajcie – nawet splot i analiza Fouriera są słuszne tylko dla systemów liniowych.

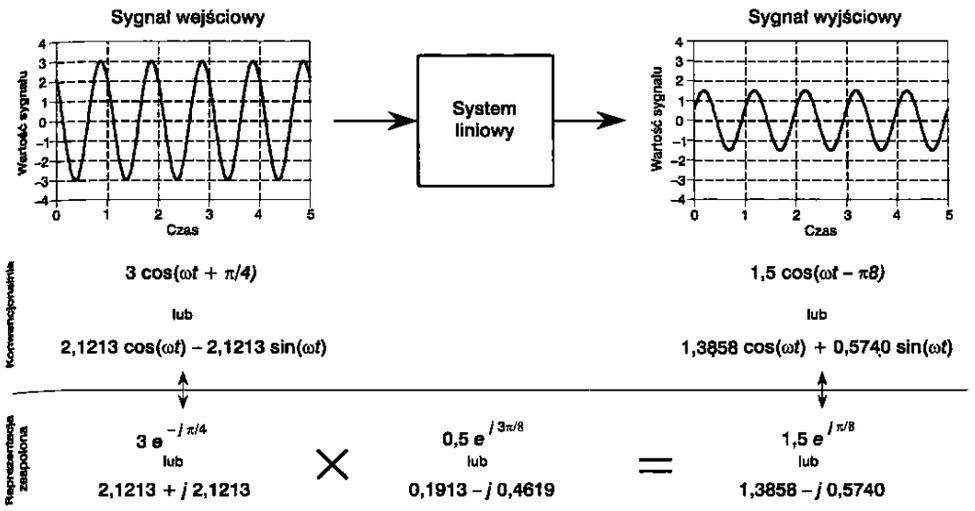
Zespolona reprezentacja systemów

Na rys. 30.4 przedstawiono przykład zastosowania liczb zespolonych do reprezentowania sinusoidy przechodzącej przez system liniowy. W tym przykładzie będziemy korzystać z sygnałów ciągłych, chociaż w ten sam sposób operuje się sygnałami dyskretnymi. Ponieważ sygnał wejściowy jest sinusoidalny, a system jest liniowy, więc na wyjściu też uzyska się sinusoidę o częstotliwości takiej samej jak na wejściu. Jak już wykazano, konwencjonalną reprezentacją przykładowego sygnału wejściowego jest: $3\cos(\omega t + \pi/4)$ lub wyrażenie równoważne: $2{,}1213\cos(\omega t) - 2{,}1213\sin(\omega t)$. Wyrażone w liczbach zespolonych wyrażenia te przyjmują postać: $3e^{-j\pi/4}$ lub $2{,}1213 + j\,2{,}1213$. Podobnie, konwencjonalna reprezentacja sygnału wyjściowego jest: $1{,}5\cos(\omega t - \pi/8)$ lub w formie alternatywnej: $1{,}3858\cos(\omega t) + 0{,}5740\sin(\omega t)$. To można wyrazić liczbą zespoloną $1{,}5e^{j\pi/8}$ lub $1{,}3858 - j\,0{,}5740$.

Charakterystyka systemu też może być wyrażona przez pewną liczbę zespoloną. Moduł tej liczby zespolonej jest stosunkiem modułów sygnałów wyjściowego i wejściowego (tj. M_{out}/M_{in}). Kąt fazowy zaś jest *negacją* różnicy między kątami sygnałów wyjściowego i wejściowego (tj. $-[\phi_{out} - \phi_{in}]$). W omawianym przykładzie system jest opisany liczbą zespoloną $0{,}5e^{j\,3\pi/8}$. A więc amplituda sinusoidy została zredukowana o 0,5, a kąt fazowy zmieniony o $-3\pi/8$.

Liczbę zespolona reprezentującą system można zamienić na współrzędne prostokątne jako: $0{,}1913 - j\,0{,}4619$, lecz musimy być ostrożni w interpretacji znaczenia tego rezultatu. Ta reprezentacja systemu nie znaczy, że fala sinusoidalna przechodząc przez system, zmienia swoją amplitudę o 0,1913 lub że kosinusoida zmienia się o –0,4619. Ogólnie biorąc, czyste fale sinusoidalne lub kosinusoidalne wchodzące do systemu liniowego są przekształcane w *mieszaninę* fal sinusoidalnych i kosinusoidalnych.

Na szczęście, matematyka liczb zespolonych śledzi te przekształcenia. Gdy sinusoida przechodzi przez system liniowy, liczby zespolone reprezentujące sygnał wejściowy oraz system są *mnożone* przez siebie, dając w wyniku liczbę zespoloną reprezentującą sygnał wyjściowy. Jeśli którekolwiek dwie liczby zespolone są znane, to trzecią można obliczyć. Obliczenia można prowadzić tak przy użyciu zapisu we współrzędnych prostokątnych, jak i w biegunowych, jak to przedstawiono na rys. 30.4.



Ilys. 30.4. Sinusoidy reprezentowane przez liczby zespolone. Liczby zespolone są często stosowane w DSP i elektronice, gdyż dają dogodny sposób reprezentowania sinusoid i operowania nimi. Jak pokazano w tym przykładzie, sinusoidalne sygnały wejściowy i wyjściowy mogą być reprezentowane przez liczby zespolone, wyrażane we współrzędnych prostokątnych lub biegunowych. Ponadto zmiana, jakiej ulega przebieg sinusoidalny w systemie liniowym, też może być wyrażona jako liczba zespolona

W poprzednich rozdziałach opisaliśmy już, jak w przekształceniu Fouriera sygnał może być rozłożony na fale sinusoidalne i kosinusoidalne. Amplitudy fal kosinusoidalnych są nazywane *częścią rzeczywistą*, a fal sinusoidalnych – *urojoną*. Podkreślaliśmy tam, że te wartości są liczbami rzeczywistymi, a pojęcia *rzeczywisty* i *urojony* były po prostu nazwami, które mają zaznaczyć odrębność tych dwóch wartości. Teraz, gdy wprowadziliśmy już liczby zespolone, stało się całkiem jasne, jakie jest pochodzenie tych nazw. Wyobraźmy sobie na przykład 1024-punktowy sygnał rozłożony na 513 fal kosinusoidalnych i 513 sinusoidalnych. Stosując metodę zastępowania, możemy wyrazić to widmo przez 513 liczb zespolonych. Nie dajcie się jednak zmylić i nie pomyślcie, że jest to *zespolone przekształcenie Fouriera* – temat rozdziału 31. Jest to bowiem ciągle jeszcze *rzeczywiste przekształcenie Fouriera*, z tym tylko że widmo umieszczono w formacie zespolonym, stosując metodę zastępowania.

Analiza obwodów elektrycznych

Ta metoda zastępowania fal kosinusoidalnych i sinusoidalnych liczbami zespolonymi jest nazywana **przekształceniem fazorowym** (w polskiej terminologii używa się nazwy **metoda symboliczna** – przyp. tłum.). Jest to główne narzędzie stosowane do analizy sieci złożonych z rezystorów, kondensatorów i cewek indukcyjnych. (Inżynierowie elektrycy definiują przekształcenie fazorowe bardziej formalnie jako mnożenie przez czynnik zespolony $e^{j\omega t}$ i wzięcie części rzeczywistej. Umożliwia to zapisanie tej procedury jako równania, co ułatwia operacje matematyczne. „Zastępowanie" daje taki sam wynik końcowy, lecz jest mniej eleganckie).

Pierwszym krokiem w tych rozważaniach jest zrozumienie wzajemnych zależności między prądem a napięciem w trzech wymienionych elementach podstawowych. W rezystorze tę zależność wyraża prawo Ohma: $u = iR$, gdzie i jest chwilową wartością prądu w rezystorze, u – chwilową wartością napięcia na rezystorze, a R – rezystancją. Natomiast zjawiskami w kondensatorze i cewce indukcyjnej rządzą równania różniczkowe: $i = C\,du/dt$ oraz $u = L\,di/dt$,

gdzie C to pojemność a L – indukcyjność. W najogólniejszym przypadku analizy obwodów te kłopotliwe równania różniczkowe są łączone ze sobą według konfiguracji układu i następnie rozwiązywane w celu wyznaczenia żądanych parametrów. Wprawdzie ta metoda powie *wszystko* o obwodzie, lecz obliczenia matematyczne powodują prawdziwy mętlik.

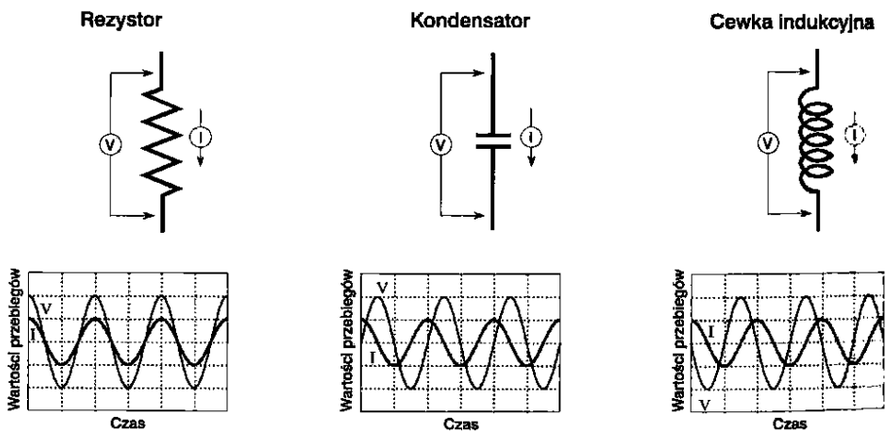
Metodę można znacznie uprościć, ograniczając sygnały tylko do sinusoidalnych. Wyrażając te sinusoidy w liczbach zespolonych, można trudne równania *różniczkowe* bezpośrednio zastąpić znacznie prostszymi równaniami *algebraicznymi*. Taki sposób działania zilustrowano na rys. 30.5. Każdy z trzech elementów podstawowych (rezystor, kondensator, cewka indukcyjna) traktujemy jako *system*. Wejściem do systemu jest prąd sinusoidalny płynący przez element, a wyjściem – napięcie między dwiema końcówkami tego elementu. Z tego wynika, że wejście i wyjście systemu mogą być reprezentowane przez dwie zmienne zespolone, odpowiednio I (dla prądu) oraz U (dla napięcia). Zależność między wejściem a wyjściem też może być wyrażona przez liczbę zespoloną. Tę liczbę zespoloną nazywamy **impedancją** i oznaczamy symbolem Z. A więc:

$$I \times Z = U$$

Można to wyrazić słownie: liczba zespolona reprezentująca napięcie sinusoidalne jest równa liczbie zespolonej reprezentującej prąd sinusoidalny pomnożonej przez impedancję (inną liczbę zespoloną). Znając dowolne dwie z tych liczb, można wyznaczyć trzecią. We współrzędnych biegunowych moduł impedancji jest stosunkiem amplitud U i I. Podobnie, faza impedancji jest różnicą faz U i I.

O tej zależności można pomyśleć, że jest to *prawo Ohma dla sinusoid*. Prawo Ohma ($u = iR$) opisuje, w jaki sposób *rezystancja* określa zależność między prądem a napięciem w rezystorze. Gdy sygnały są sinusoidalne, zależność przyjmuje postać: $U = I\,Z$. Tak więc *impedancja* wyznacza zależność między prądem a napięciem. Rezystancja jest zwykłą liczbą, gdyż jest związana z dwiema zwykłymi liczbami. Impedancja jest liczbą zespoloną, gdyż daje zależność między dwiema liczbami zespolonymi. Impedancja zawiera więcej informacji niż rezystancja, gdyż określa zarówno amplitudy, *jak i* kąty fazowe.

Na podstawie równań różniczkowych opisujących działanie rezystora, kondensatora i cewki indukcyjnej można wykazać, że impedancje tych elementów są, odpowiednio: R, $-j/\omega C$ oraz $j\omega L$. Przyjmijmy dla przykładu, że prąd w każdym z tych elementów jest falą kosinusoidalną o jednostkowej amplitudzie (rys. 30.5). Stosując metodę zastępowania, można wyrazić taką falę jako $1 + 0j$. Napięcie na rezystorze będzie więc: $U = I\,Z = (1 + 0j)\,R = R + 0j$. Inaczej



Rys. 30.5. Definicja impedancji. Stosunek napięcia i prądu wyrażonych w liczbach zespolonych jest nazywany impedancją i oznaczany jako zmienna zespolona Z. Rezystory, kondensatory i cewki indukcyjne mają, odpowiednio, impedancje R, –j/ωC oraz jωL

mówiąc, jest to fala kosinusoidalna o amplitudzie R. Następnie wyznaczamy napięcie na kondensatorze: $U = I\,Z = (1 + 0j)\,(-j/\omega C)$. To wyrażenie redukuje się do $(0 - j/\omega C)$, a więc do sinusoidy o amplitudzie $1/\omega C$. W podobny sposób oblicza się napięcie na cewce indukcyjnej: $U = I\,Z = (1 + 0j)(j\omega L)$. To wyrażenie redukuje się do $(0 + j\omega L)$, a więc ujemnej sinusoidy o amplitudzie ωL.

Uroda tej metody polega na tym, że obwody *RLC* można analizować bez uciekania się do równań różniczkowych. *Impedancja* rezystorów, kondensatorów i cewek indukcyjnych jest traktowana tak samo jak *rezystancja* w obwodzie prądu stałego. To obowiązuje też dla wszystkich kombinacji podstawowych, takich jak rezystory połączone szeregowo lub równolegle, dzielniki napięcia itd.

Na rys. 30.6 przedstawiono, jako przykład, obwód *RLC* zwany **filtrem wycinającym** (pasmowozaporowym, tzw. filtr typu *notch*) służącym do usuwania wąskiego pasma częstotliwości. Taki filtr może np. eliminować zakłócenia o częstotliwości 60 Hz (w Europie 50 Hz, przyp. tłum.) w sygnale fonicznym albo pomiarowym. Gdyby ten obwód składał się z 3 rezystorów (zamiast rezystora, kondensatora i cewki), to zależność między napięciem wyjściowym a wejściowym byłaby określona wzorem stosowanym dla dzielników napięcia: $U_{out}/U_{in} = (R_2 + R_3) / (R_1 + R_2 + R_3)$. Ponieważ obwód zawiera kondensator i cewkę indukcyjną, więc rezystancje we wzorze trzeba zastąpić impedancjami:

$$\frac{U_{out}}{U_{in}} = \frac{Z2+Z3}{Z1+Z2+Z3}$$

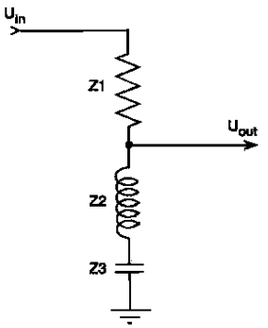
gdzie: U_{out}, U_{in}, Z, $Z2$, $Z3$ są zmiennymi zespolonymi. Wstawiając impedancje każdego elementu:

$$\frac{U_{out}}{U_{in}} = \frac{j\omega L - \frac{j}{\omega C}}{R + j\omega L - \frac{j}{\omega C}}$$

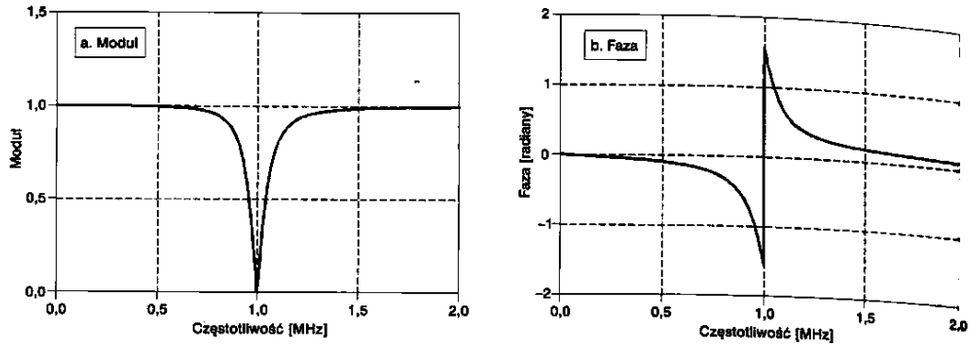
Następnie wykonujemy operacje algebraiczne, aby oddzielić wszystkie wyrazy zawierające j od wszystkich, które j nie zawierają. Innymi słowy, rozdzielamy to wyrażenie na jego części rzeczywistą i urojoną. Te działania algebraiczne mogą być nużące i długie, lecz alternatywą byłoby napisanie i rozwiązanie *równań różniczkowych*, a więc zadanie jeszcze przykrzejsze. Po wydzieleniu części rzeczywistej i urojonej wyrażenie zespolone opisujące filtr wycinający przyjmuje postać:

$$\frac{U_{out}}{U_{in}} = \frac{k^2}{R^2+k^2} + j\frac{Rk}{R^2+k^2}$$

$$gdzie: k = \omega L - 1/\omega C$$



Rys. 30.6. Filtr RLC pasmowozaporowy (filtr wycinający, zwany też filtrem typu notch). Ten obwód usuwa z sygnału wąskie pasmo częstotliwości. Zastosowanie metody zastępowania liczbami zespolonymi znacznie upraszcza analizę tego i jemu podobnych obwodów



Rys. 30.7. Odpowiedź częstotliwościowa filtru pasmowozaporowego (wycinającego). Wykresy wykonano dla wartości elementów: R = 50 Ω, C = 470 pF, L = 54 μH

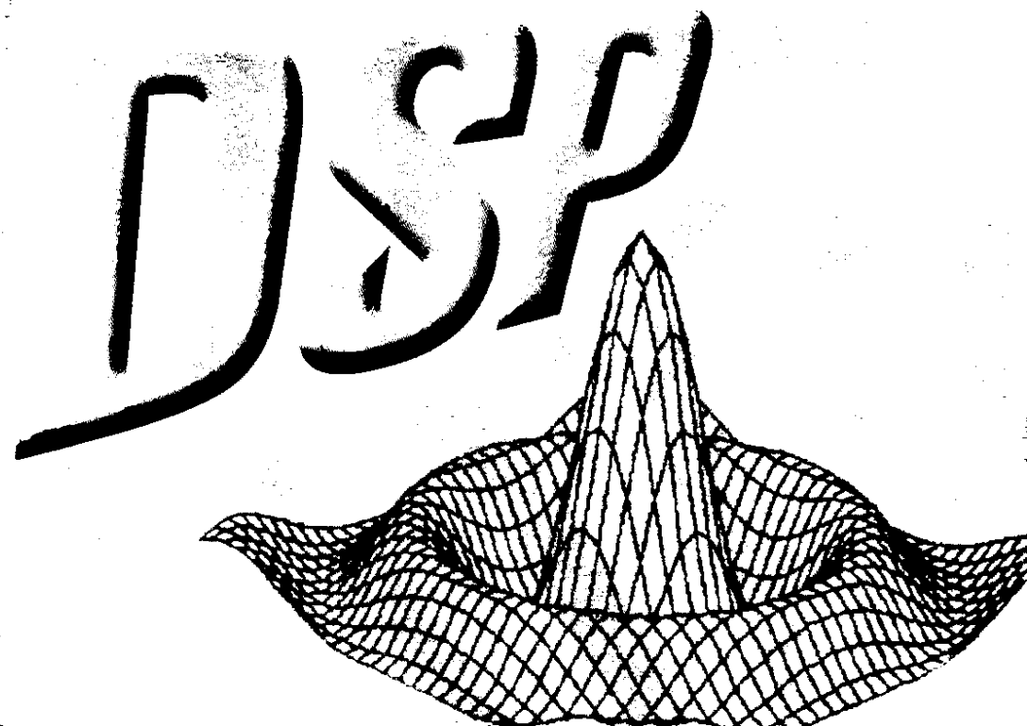
W końcu przekształcamy to na współrzędne biegunowe i wykonujemy wykres (rys. 30.7):

$$Mag = \frac{\omega L - 1/\omega C}{\left(R^2 + \left[\omega L - 1/\omega C\right]^2\right)^{1/2}} \quad Phase = \text{arctg}\left[\frac{R}{\omega L - 1/\omega C}\right]$$

W tych przykładach istotną do zapamiętania rzeczą jest sposób, w jaki *zastępowanie* umożliwia reprezentację problemów świata rzeczywistego przez liczby zespolone. W następnym rozdziale zajmiemy się trudniejszym sposobem stosowania liczb zespolonych w nauce i technice – *równoważnością matematyczną*.

31

Zespolone przekształcenie Fouriera



Liczby zespolone, choć zasadniczo niezwiązane ze światem rzeczywistym, mogą być stosowane do rozwiązywania zagadnień naukowych i technicznych. Są w zasadzie dwa sposoby wykorzystania liczb zespolonych w problemach naukowo-technicznych. Pierwszy to zastępowanie parametrów świata rzeczywistego liczbami zespolonymi, jak to omówiono w poprzednim rozdziale. W drugiej metodzie, znacznie elegantszej i skuteczniejszej, liczby zespolone czyni się matematycznie równoważnymi problemom fizycznym. Takie podejście prowadzi do *zespolonego przekształcenia Fouriera*, które jest bardziej wysublimowaną wersją *rzeczywistego przekształcenia Fouriera* opisanego w rozdziale 8. Zespolone przekształcenie Fouriera, samo w sobie bardzo ważne, jest także ważnym krokiem na drodze ku skuteczniejszym metodom zespolonym – *przekształceniu Laplace'a* i *przekształceniu* Z. Te przekształcenia zespolone są podstawą teoretycznego DSP.

Rzeczywiste DFT

Wszystkie cztery przekształcenia z rodziny transformat Fouriera (DFT, DTFT, przekształcenie Fouriera i szereg Fouriera) mogą być wykonywane zarówno na liczbach rzeczywistych, jak i na zespolonych. Jako przykładem posłużymy się DFT, gdyż to przekształcenie jest przede wszystkim związane z DSP. Przed wejściem w matematykę liczb zespolonych przyjrzyjmy się przekształceniu DFT ze szczególnym zwróceniem uwagi na te sprawy, które są matematycznie kłopotliwe. W rozdziale 8 zdefiniowaliśmy wersję *rzeczywistą* dyskretnego przekształcenia Fouriera zgodnie z równaniami:

$$Re\,X[k]=\frac{2}{N}\sum_{n=0}^{N-1}x[n]\cos\left(\frac{2\pi kn}{N}\right)$$

$$Im\,X[k]=\frac{-2}{N}\sum_{n=0}^{N-1}x[n]\sin\left(\frac{2\pi kn}{N}\right)$$

Wzór 31.1. Rzeczywiste przekształcenie DFT. Jest to przekształcenie proste – obliczanie charakterystyki w dziedzinie częstotliwości na podstawie przebiegu w dziedzinie czasu. Mimo korzystania z nazw część rzeczywista i część urojona, te wzory dotyczą tylko zwykłych liczb. Indeks częstotliwości k zmienia się od 0 do N/2. Są to te same wzory, co wzory 8.4 z tą różnicą, że do transformaty prostej wprowadzono czynnik 2/N

Zgodnie z wzorami 31.1 sygnał $x[n]$ złożony z N próbek w dziedzinie czasu jest rozkładany na $(N/2) + 1$ fal kosinusoidalnych i $(N/2) + 1$ fal sinusoidalnych, o częstotliwościach określonych indeksem k. Wartości fal kosinusoidalnych są zawarte w *Re X*[*k*], a sinusoidalnych – w *Im X*[*k*]. Te wzory działają na zasadzie *korelacji* poszczególnych fal kosinusoidalnych i sinusoidalnych z sygnałem w dziedzinie czasu. Mimo użytych nazw *część rzeczywista i część urojona*, nie ma w tych wzorach żadnych liczb zespolonych. Nigdzie nie widać *j*! Do wzorów włączono też czynnik normalizujący $2/N$. Pamiętajcie, że może on być umieszczony albo przed równaniem analizy, albo syntezy, lub też wprowadzony jako oddzielna operacja (jak to podano we wzorze 8.3). Te równania powinny być wam dobrze znane z poprzednich rozdziałów. Jeśli ich już nie pamiętacie, to wróćcie do tamtych analiz i uporządkujcie swoją wiedzę na ten temat przed kontynuowaniem tych rozważań. Jeśli nie rozumiecie *rzeczywistego* DFT, to nigdy nie będziecie w stanie zrozumieć DFT *zespolonego*.

Chociaż w rzeczywistym DFT stosuje się tylko liczby rzeczywiste, to metoda *zastępowania* umożliwia *reprezentację* widm w dziedzinie częstotliwości przez liczby zespolone. Jak sugerują nazwy matryc, *Re X*[*k*] staje się częścią rzeczywistą zespolonego widma częstotliwościowego, a *Im X*[*k*] – jego częścią urojoną. Inaczej mówiąc, dopisujemy *j* do każdej wartości w części urojonej i wynik dodajemy do części rzeczywistej. Nie pomylcie się jednak i nie sądźcie, że macie do czynienia z „zespolonym DFT". Jest to nic więcej, jak *rzeczywiste DSP z zespolonym zastępowaniem*.

Rzeczywiste DSP, chociaż przydatne do wielu zastosowań w nauce i technice, jest jednak niewygodne matematycznie z trzech względów. Po pierwsze, można w nim korzystać z liczb zespolonych, tylko stosując **zastępowanie**. Dla matematyków jest to niedogodne; oni chcieliby móc powiedzieć: *to jest równe tamtemu*, a nie *to reprezentuje tamto*. Wyobraźmy sobie, dla przykładu, że mamy stwierdzenie matematyczne: *A jest równe B*. Z tego od razu wynikają niezliczone konsekwencje: $5A = 5B$, $1 + A = 1 + B$, $A/x = B/x$ itd. A teraz przypuśćmy, że mamy stwierdzenie: *A* reprezentuje *B*. Wtedy, bez jakichś dodatkowych informacji, nie wiemy absolutnie nic! Gdy dwie wielkości są równe, to takie stwierdzenie uprawnia nas do korzystania z czterech tysięcy lat wiedzy matematycznej. Jeśli zaś te wielkości tylko reprezentują siebie nawzajem, to musimy od podstaw tworzyć nowe definicje. Gdy, na przykład, sinusoidy są reprezentowane przez liczby zespolone, to zezwalamy na dodawanie i odejmowanie, ale zakazujemy mnożenia i dzielenia.

Drugą słabą stroną rzeczywistego przekształcenia Fouriera jest część widma o **częstotliwościach ujemnych**. Jak pamiętacie z rozdziału 10, fale sinusoidalne i kosinusoidalne można opisywać jako mające częstotliwość *dodatnią* lub *ujemną*. Ponieważ obie te części widma są identyczne, więc w rzeczywistym przekształceniu Fouriera pomija się częstotliwości ujemne. Bywają jednak takie zastosowania, w których ujemne częstotliwości są ważne. Ma to miejsce wówczas, gdy składowe o ujemnych częstotliwościach są zmuszane do przesunięcia się do części widma o częstotliwościach dodatnich. Można by powiedzieć, że duch przyjmuje wówczas ludzką postać. Takie zjawisko następuje np. w przypadku aliasingu, splocie kołowym i modulacji amplitudy. Ponieważ w rzeczywistym przekształceniu Fouriera nie stosuje się częstotliwości ujemnych, więc możliwości pokonania wymienionych problemów są w tym przekształceniu bardzo ograniczone.

Trzecią niedogodnością rzeczywistego przekształcenia Fouriera jest **specjalne operowanie *Re X*[0] i *Re X*[N/2]** – pierwszymi i ostatnim punktem w widmie częstotliwościowym. Załóżmy, że mamy N-punktowy sygnał $x[n]$. Wykonanie przekształcenia DFT daje widmo częstotliwościowe zawarte w *Re X*[*k*] i *Im X*[*k*], gdzie *k* zmienia się od 0 do *N*/2. To nie są jednak wartości, mogące służyć do odtworzenia przebiegu w dziedzinie czasu. Próbki *Re X*[0] i *Re X*[*N*/2] trzeba najpierw podzielić przez 2. (Dla przypomnienia wróćcie do podrozdziału 8.3). Taką operację dzielenia łatwo się wykonuje w programie komputerowym, lecz jest ona niewygodna przy operowaniu wzorami.

Zespolone przekształcenie Fouriera jest eleganckim rozwiązaniem tych problemów. Liczby zespolone w sposób naturalny są związane z częstotliwościami ujemnymi. Zobaczmy, jak to działa.

Równoważność matematyczna

Pierwszym naszym krokiem jest przedstawienie, w jaki sposób można fale sinusoidalne i kosinusoidalne zapisać przy użyciu *wzorów* zawierających liczby zespolone. Podstawą jest zależność Eulera, podana w poprzednim rozdziale:

$$e^{jx}=\cos(x)+j\sin(x)$$

Wzór 31.2. Zależność Eulera

Na pierwszy rzut oka nie wydaje się, aby ten wzór był bardzo pomocny; jedno wyrażenie zespolone jest równe drugiemu wyrażeniu zespolonemu. Niemniej jednak, po niewielkich operacjach algebraicznych, można tę zależność przekształcić w dwie inne:

$$\cos(x)=\frac{e^{jx}+e^{-jx}}{2}\qquad \sin(x)=\frac{e^{jx}-e^{-jx}}{2j}$$

Wzór 31.3. Zależności Eulera dla sinusa i kosinusa

Uzyskany wynik jest niezwykle ważny. Daje on bowiem sposób zapisu *zależności* między zwykłymi sinusoidami a liczbami zespolonymi. Chociaż zależności 31.3 mają postać standardowych tożsamości, to do dalszych rozważań będą one użyteczniejsze po zamianie niektórych wyrazów:

$$\cos(\omega t)=\frac{1}{2}e^{j(-\omega)t}+\frac{1}{2}e^{j\omega t}$$

$$\sin(\omega t)=\frac{1}{2}e^{j(-\omega)t}-\frac{1}{2}je^{j\omega t}$$

Wzór 31.4. Sinusoidy zapisane jako liczby zespolone. Stosując liczby zespolone, można fale sinusoidalne i kosinusoidalne zapisać jako sumę przebiegów o częstotliwościach dodatnich i ujemnych

Każde z powyższych wyrażeń jest sumą dwóch przebiegów wykładniczych: jednego zawierającego częstotliwości *dodatnie* (ω) i drugiego, który zawiera częstotliwości ujemne (–ω). Tak więc fale sinusoidalne i kosinusoidalne są zapisane w postaci liczb zespolonych, przy czym część widma o częstotliwościach ujemnych jest automatycznie uwzględniana. Częstotliwości dodatnie i ujemne są traktowane na równych prawach. Utworzenie pełnego przebiegu wymaga połowy informacji o częstotliwościach dodatnich i połowy o ujemnych.

Zespolone DFT

Proste, zespolone przekształcenie DFT, zapisane we współrzędnych biegunowych, jest opisane wzorem:

$$X[k]=\frac{1}{N}\sum_{n=0}^{N-1}x[n]e^{\frac{-j2\pi kn}{N}}$$

Wzór 31.5. Proste, zespolone DFT. Zarówno sygnał w dziedzinie czasu x[n], jak i częstotliwości X[k] są matrycami liczb zespolonych, w których k i n zmieniają się od 0 do N – 1. Wzór jest wyrażony we współrzędnych biegunowych, najczęściej stosowanych w DSP

Zależność Eulera może też służyć do zapisu transformaty we współrzędnych prostokątnych:

$$X[k]=\frac{1}{N}\sum_{n=0}^{N-1}x[n]\left(\cos\left(\frac{2\pi kn}{N}\right)-j\sin\left(\frac{2\pi kn}{N}\right)\right)$$

Wzór 31.6. Proste, zespolone DFT (we współrzędnych prostokątnych)

Na początek porównajmy ten wzór opisujący *zespolone przekształcenie Fouriera* z wzorem opisującym *rzeczywiste przekształcenie Fouriera* (wzór 31.1). Na pierwszy rzut oka oba wzory wydają się identyczne i tylko niewielkie operacje algebraiczne są niezbędne do przekształcenia wzoru 31.6 we wzór 31.1. To odczucie jest jednak mylące. Różnice między oboma wzorami są bardzo subtelne i łatwo je przeoczyć, lecz są one niezwykle ważne. Przeanalizujmy je szczegółowo.

Po pierwsze, rzeczywiste przekształcenie Fouriera transformuje sygnał rzeczywisty w dziedzinie czasu $x[n]$ na dwa sygnały rzeczywiste w dziedzinie częstotliwości *Re* $X[k]$ i *Im*$[k]$. Przy zastosowaniu metody zastępowania liczbami zespolonymi sygnał w dziedzinie częstotliwości może być reprezentowany jedną matrycą zespoloną $X[k]$. W zespolonym przekształceniu Fouriera zarówno $x[n]$, jak i $X[k]$ są matrycami liczb zespolonych. Uwaga praktyczna: chociaż sygnał w dziedzinie częstotliwości jest zespolony, nic nas *nie zmusza* do wykorzystania części urojonej. Powiedzmy, że chcemy przetwarzać sygnał rzeczywisty, taki jak szereg wyników pomiarów napięcia wykonanych w pewnym czasie. Ta grupa danych staje się częścią rzeczywistą sygnału w dziedzinie czasu, podczas gdy część urojona składa się z zer.

Po drugie, w rzeczywistym przekształceniu Fouriera operuje się tylko częstotliwościami *dodatnimi*. To znaczy indeks w dziedzinie częstotliwości zmienia się tylko od 0 do *N*/2. Zespolone przekształcenie Fouriera obejmuje zaś częstotliwości tak *dodatnie*, jak i *ujemne*. To znaczy, że k zmienia się od 0 do $N-1$. Częstotliwości między 0 a $N/2$ są dodatnie, a od $N/2$ do $N-1$ ujemne. Pamiętajcie, że widmo częstotliwościowe sygnału dyskretnego jest okresowe, co powoduje, że sygnał dla częstotliwości ujemnych między $N/2$ a $N-1$ jest taki sam, jak sygnał między $-N/2$ a 0. Próbki 0 oraz $N/2$ rozgraniczają obszary częstotliwości ujemnych i dodatnich. Jeśli chcecie odświeżyć swoje wiadomości na ten temat, zajrzyjcie ponownie do rozdziałów 10 i 12.

Po trzecie, w rzeczywistym przekształceniu Fouriera z metodą zastępowania dodaje się j do wyrazów sinusoidalnych, dzięki czemu widmo częstotliwościowe jest reprezentowane przez liczby zespolone. Aby przeprowadzić konwersję odwrotną, z powrotem do zwykłych fal sinusoidalnych i kosinusoidalnych, możemy po prostu odrzucić j. Jest to pewna niedbałość, która może zaistnieć wówczas, gdy jedna rzecz tylko *reprezentuje* drugą. Zespolone DFT (wzór 31.5) jest natomiast formalnym równaniem matematycznym, którego integralną częścią jest j. W tej sytuacji nie możemy arbitralnie dodawać lub usuwać j, podobnie jak nie możemy w równaniach ani dodawać, ani usuwać innych zmiennych.

Po czwarte, w przekształceniu rzeczywistym występuje współczynnik skalowania *dwa* na początku wyrażenia, a w zespolonym przekształceniu Fouriera go nie ma. Przypuśćmy, że wykonujemy rzeczywiste przekształcenie DFT fali kosinusoidalnej o amplitudzie *jeden*. Wartość prążka widmowego odpowiadająca fali kosinusoidalnej jest również *jeden*. Powtórzmy teraz ten proces, stosując zespolone DFT. W tym przypadku fala kosinusoidalna odpowiada *dwóm* wartościom prążka widmowego, częstotliwości ujemnej i dodatniej. Obie te częstotliwości charakteryzują się wartością 1/2. Innymi słowy, sygnał o częstotliwości dodatniej i o amplitudzie 1/2 łączy się z sygnałem o częstotliwości ujemnej o amplitudzie 1/2, wytwarzając kosinusoidę o amplitudzie *jeden*.

Po piąte, rzeczywiste przekształcenie Fouriera wymaga specjalnego operowania dwiema próbkami z dziedziny częstotliwości *Re* $X[0]$ i *Re* $X[N/2]$, a w przekształceniu zespolonym nie ma takiego wymogu. Przyjmijmy, że rozpoczynamy od sygnału w dziedzinie czasu i wyznaczamy jego DFT, aby uzyskać sygnał w dziedzinie częstotliwości. Odwracając ten proces, wyznaczamy odwrotne DFT sygnału w dziedzinie częstotliwości w celu odtworzenia oryginalnego sygnału w dziedzinie czasu. Jednak, aby sygnał odtworzony był identyczny z oryginalnym, konieczne jest skalowanie. W zespolonym przekształceniu Fouriera trzeba gdzieś „po drodze" wprowadzić czynnik $1/N$. Może on być doczepiony do przekształcenia prostego lub odwrotnego albo potraktowany jako oddzielny krok między oboma przekształceniami. W rzeczywistym przekształceniu Fouriera jest wymagany dodatkowy czynnik równy *dwa* ($2/N$), jak to poprzednio powiedziano. Jednak rzeczywiste przekształcenie Fouriera wymaga też dodatkowego kroku dla skalowania: *Re* $X[0]$ i *Re* $X[N/2]$ muszą gdzieś w trakcie obliczeń zostać podzielone przez *dwa*. Tak więc współczynnik skalowania $1/N$ stosuje się do tych dwóch próbek, a $2/N$ – do reszty widma. Jak stwierdzono poprzednio, to kłopotliwe skalowanie jest jednym z mankamentów rzeczywistego przekształcenia Fouriera.

Dlaczego rzeczywiste i zespolone DFT różnią się od siebie pod względem obsługi tych dwóch szczególnych punktów? Aby odpowiedzieć na to pytanie, przypomnijcie sobie, że fala kosinusoidalna (lub sinusoidalna) w dziedzinie czasu zostaje w zespolonym widmie DFT rozdzielona między częstotliwość dodatnią i ujemną. Są jednak dwa wyjątki – wartości widma dla 0 i $N/2$. Odpowiadają one częstotliwościom 0 (DC) i Nyquista (połowa częstotliwości próbkowania). Ponieważ te punkty rozgraniczają części widma o częstotliwościach dodatnich i ujemnych, więc nie mają swojego odpowiednika. Jako że nie są powiązane z inną wartością, w sposób inherentny wnoszą do dziedziny czasu tylko udział połówkowy w stosunku do innych częstotliwości.

Zespolone widmo częstotliwościowe DFT przedstawiono na rys. 31.1. Na tym rysunku założono, że sygnał w dziedzinie czasu jest całkowicie *rzeczywisty*, to znaczy, że jego część urojona jest równa *zero*. Krótko rozważymy kwestię sygnałów urojonych w dziedzinie czasu. Są dwa powszechnie stosowane sposoby prezentacji widma zespolonego widma częstotliwościowego. Jak pokazano na rysunku, częstotliwość zero może być umieszczona pośrodku wykresu, częstotliwości dodatnie – na prawo od niej, a ujemne – na lewo. Jest to najlepszy sposób *myślenia* o widmie *zespolonym* i *jedyny* sposób, w jaki można przedstawić widmo nieokresowe.

Problem polega na tym, że widmo sygnału dyskretnego jest *okresowe* (jak w DFT i DTFT). To znaczy, że widmo mieszczące się między –0,5 i 0,5 powtarza się nieskończoną ilość razy ku prawej i ku lewej stronie. W tym przypadku widmo między 0 i 1,0 zawiera taką samą infor-

mację, jak od –0,5 do 0,5. Przy wykreślaniu wykresów, takich jak na rys. 31.1, przyjmuje się zwykle konwencję od –0,5 do 0,5. Jednak w wielu równaniach i programach stosuje się formę od 0 do 1,0. Na przykład we wzorach 31.5 i 31.6 indeks częstotliwości k zmienia się od 0 do $N-1$ (co jest równoważne zakresowi od 0 do 1,0). Moglibyśmy jednak tak zapisać te wzory, żeby indeks zmieniał się od $-N/2$ do $(N/2)-1$ (co jest równoważne zakresowi od –0,5 do 0,5), gdybyśmy tak chcieli.

Stosując jako punkt wyjścia widmo z rys. 31.1, możemy sprawdzić, jak odwrotne zespolone przekształcenie DFT odtwarza sygnał w dziedzinie czasu. Odwrotne zespolone DFT, zapisane we współrzędnych biegunowych, jest opisane wzorem:

$$x[n]=\sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi kn}{N}}$$

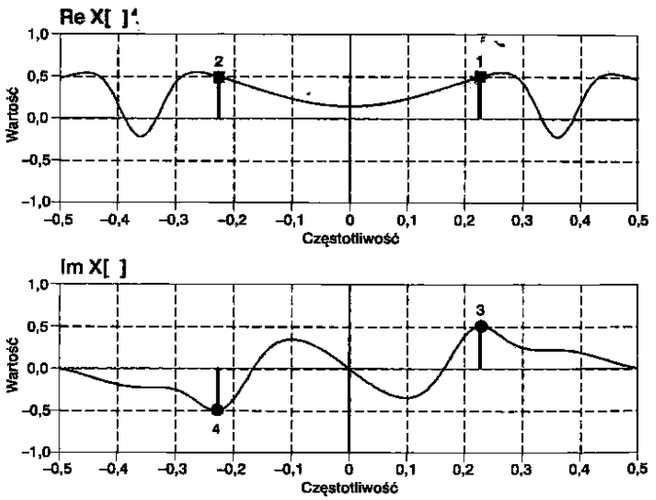
Wzór 31.7. Odwrotne, zespolone przekształcenie DFT. Jest to odpowiednik wzoru 31.5 na proste, zespolone przekształcenie DFT

Stosując zależność Eulera, można te wzory zapisać we współrzędnych biegunowych jako:

$$x[n]=\sum_{k=0}^{N-1} Re\,X[k]\left(\cos\left(\frac{2\pi kn}{N}\right)+j\sin\left(\frac{2\pi kn}{N}\right)\right)$$
$$-\sum_{k=0}^{N-1} Im\,X[k]\left(\sin\left(\frac{2\pi kn}{N}\right)-j\cos\left(\frac{2\pi kn}{N}\right)\right)$$

Wzór 31.8. Odwrotne, zespolone przekształcenie DFT. Jest to wzór 31.7 zapisany w taki sposób, żeby pokazać, jak każda wartość w widmie częstotliwościowym wpływa na sygnał w dziedzinie czasu

Odwrotne przekształcenie DFT jest zwykle zapisywane w zwartej formie (wzór 31.7), chociaż wersja rozszerzona (wzór 31.8) jest łatwiejsza do zrozumienia. Opisując wzór słowami, można stwierdzić, że każda wartość w rzeczywistej części sygnału w dziedzinie częstotliwości wnosi do dziedziny czasu rzeczywistą falę kosinusoidalną *oraz* urojoną sinusoidalną.



Rys. 31.1. Zespolone widmo częstotliwościowe. Te krzywe dotyczą całkowicie rzeczywistego sygnału w dziedzinie czasu, gdyż część rzeczywista widma ma symetrię parzystą, a część urojona – nieparzystą. Dwa kwadratowe znaczniki w części rzeczywistej odpowiadają fali kosinusoidalnej o amplitudzie równej jeden i częstotliwości 0,23. Dwa znaczniki okrągłe w części urojonej odpowiadają fali sinusoidalnej o amplitudzie równej jeden i częstotliwości 0,23

Podobnie, każda wartość w części urojonej sygnału w dziedzinie częstotliwości wnosi rzeczywistą falę sinusoidalną *oraz* urojoną kosinusoidalną. Sygnał w dziedzinie czasu znajduje się, sumując te wszystkie rzeczywiste i urojone sinusoidy. Ważne jest to, że każda wartość sygnału w dziedzinie częstotliwości wytwarza w dziedzinie czasu zarówno sinusoidę *rzeczywistą*, jak i *urojoną*.

Przypuśćmy na przykład, że chcemy odtworzyć falę kosinusoidalną o amplitudzie jednostkowej przy częstotliwości $2\pi k/N$. Wymaga to wykorzystania tak częstotliwości dodatniej, jak i ujemnej, obu z rzeczywistej części widma częstotliwościowego. Dwa kwadratowe znaczniki na rys. 31.1 są tego przykładem, z częstotliwościami ustawionymi na $k/N = 0{,}23$. Sygnał o dodatniej częstotliwości odpowiadającej wartości 0,23 (oznaczonej 1 na rys. 31.1) wnosi do dziedziny czasu falę kosinusoidalną i urojoną falę sinusoidalną:

$$\frac{1}{2}\cos(2\pi 0{,}23n)+\frac{1}{2}j\sin(2\pi 0{,}23n)$$

Podobnie sygnał o częstotliwości ujemnej odpowiadającej wartości –0,23 (oznaczenie 2 na rys. 31.1) również wnosi do dziedziny czasu kosinusoidę i urojoną sinusoidę:

$$\frac{1}{2}\cos(2\pi(-0{,}23)n)+\frac{1}{2}j\sin(2\pi(-0{,}23)n)$$

Ujemny znak w argumentach cosinusa i sinusa można wyeliminować, korzystając z zależności: $\cos(-x) = \cos x$ oraz $\sin(-x) = -\sin x$. Można więc udział częstotliwości ujemnej zapisać jako:

$$\frac{1}{2}\cos(2\pi 0{,}23n)-\frac{1}{2}j\sin(2\pi 0{,}23n)$$

Zsumowanie udziałów od częstotliwości ujemnych i dodatnich daje odtworzenie sygnału w dziedzinie czasu:

udział od częstotliwości dodatniej $\rightarrow \frac{1}{2}\cos(2\pi 0{,}23n)+\frac{1}{2}j\sin(2\pi 0{,}23n)$

udział od częstotliwości ujemnej $\rightarrow \frac{1}{2}\cos(2\pi 0{,}23n)-\frac{1}{2}j\sin(2\pi 0{,}23n)$

wynikowy sygnał w dziedzinie czasu $\rightarrow \cos(2\pi 0{,}23n)$

W ten sam sposób możemy zsyntezować falę sinusoidalną w dziedzinie czasu. W tym przypadku potrzebna jest częstotliwość dodatnia i ujemna z części urojonej widma częstotliwościowego. Zaznaczono je okrągłymi znacznikami na rys. 31.1. Z wzoru 31.8 wynika, że te wartości widma wnoszą do dziedziny czasu falę sinusoidalną i urojoną falę kosinusoidalną. Urojone fale kosinusoidalne się kasują, a rzeczywiste sinusoidalne sumują:

udział od częstotliwości dodatniej $\rightarrow \frac{1}{2}\sin(2\pi 0{,}23n)-\frac{1}{2}j\cos(2\pi 0{,}23n)$

udział od częstotliwości ujemnej $\rightarrow \frac{1}{2}\sin(2\pi 0{,}23n)+\frac{1}{2}j\cos(2\pi 0{,}23n)$

wynikowy sygnał w dziedzinie czasu $\rightarrow -\sin(2\pi 0{,}23n)$

Zauważcie, że jest generowana *ujemna* fala sinusoidalna, chociaż sygnał o dodatniej częstotliwości miał wartość, która była *dodatnia*. Odwrócenie znaku jest nierozerwalnie związane z operacjami matematycznymi zespolonego DSP. Jak pamiętacie, takie samo odwrócenie znaku jest zwykle stosowane w rzeczywistym DFT. To znaczy, że *dodatnia* wartość w części urojonej widma częstotliwościowego odpowiada *ujemnej* fali sinusoidalnej. Większość autorów włącza to odwrócenie znaku do definicji rzeczywistego przekształcenia Fouriera, aby było ono zgodne z jego zespolonym odpowiednikiem. Rzecz jest w tym, że takie odwrócenie znaku musi być stosowane w zespolonym przekształceniu Fouriera, a w rzeczywistym jest tylko opcją.

Bardzo ważna jest *symetria* zespolonego przekształcenia Fouriera. Jak pokazano na rys. 31.1, rzeczywisty sygnał w dziedzinie czasu odpowiada widmu częstotliwościowemu z *parzystą* częścią rzeczywistą i *nieparzystą* częścią urojoną. Inaczej mówiąc, sygnały przy częstotliwościach ujemnych i dodatnich mają takie same znaki w części rzeczywistej (jak punkty 1 i 2 na rys. 31.1), a przeciwne w części urojonej (punkty 3 i 4).

To prowadzi do innego tematu – do kwestii części urojonej w dziedzinie czasu. Dotychczas zakładaliśmy, że sygnał w dziedzinie czasu jest całkowicie rzeczywisty, czyli że część urojona jest zerowa. Nie jest to jednak wymagane w zespolonym przekształceniu Fouriera. Jaki jest więc sens fizyczny urojonego sygnału w dziedzinie czasu? Zwykle nie ma to żadnego sensu fizycznego, jest to po prostu rzecz dopuszczalna w matematyce liczb zespolonych, bez żadnego odniesienia do świata, w którym żyjemy. Są jednak takie zastosowania, gdzie ten fakt może być wykorzystany do celów matematycznych.

Taki przykład podano w rozdziale 12. Część urojona w dziedzinie czasu daje widmo częstotliwościowe o nieparzystej części rzeczywistej i parzystej urojonej. Jest to przeciwieństwo widma dawanego przez część rzeczywistą sygnału w dziedzinie czasu (rys. 31.1). Podczas gdy sygnał w dziedzinie czasu zawiera zarówno część rzeczywistą, jak i urojoną, to widmo częstotliwościowe jest sumą tych dwóch widm, które zostały oddzielnie obliczone. W rozdziale 12 opisano, jak zastosować tę zasadę, aby algorytm FFT obliczał od razu widma częstotliwościowe *dwóch* sygnałów rzeczywistych. Jeden sygnał jest umieszczony w części rzeczywistej dziedziny czasu, drugi zaś – w części urojonej. Po obliczeniu FFT widma dwóch sygnałów są rozdzielane metodą rozkładu parzysto-nieparzystego.

Rodzina przekształceń Fouriera

Podobnie jak DFT, również inne przekształcenia z rodziny fourierowskiej mają wersje rzeczywiste i zespolone. Mamy więc cały „ogród zoologiczny" wzorów, których zestawienie podano w tablicy 31.1. Zamiast analizować poszczególne równania, spróbujcie je zrozumieć jako dobrze zorganizowaną, symetryczną *grupę*. Niżej podane komentarze opisują organizację grupy przekształceń Fouriera. Komentarze są szczegółowe, powtarzające się i nudne. Niemniej są one podstawą dla zrozumienia teoretycznego DSP. Przestudiujcie je dokładnie.

1. Cztery przekształcenia Fouriera

Sygnał w dziedzinie czasu może być *ciągły* albo *dyskretny*; może też być *okresowy* lub *nieokresowy*. Z tego wynikają cztery typy przekształceń Fouriera: **dyskretne przekształcenie Fouriera** (sygnał dyskretny, okresowy), **dyskretne w czasie przekształcenie Fouriera** (sygnał dyskretny, nieokresowy), **szereg Fouriera** (sygnał ciągły, okresowy) i **przekształcenie Fouriera** (sygnał ciągły, nieokresowy).

Jeśli sygnał jest dyskretny w jednej dziedzinie, to będzie okresowy w drugiej. Podobnie, jeśli sygnał jest ciągły w jednej dziedzinie, to będzie nieokresowy w drugiej. Sygnały ciągłe są oznaczane nawiasami okrągłymi ( ), dyskretne – kwadratowymi [ ]. Nie ma żadnego zapisu odróżniającego sygnał okresowy od nieokresowego.

2. Przekształcenia rzeczywiste i zespolone

Każde z wymienionych czterech przekształceń Fouriera ma swoją wersję zespoloną i rzeczywistą. W zespolonych wersjach przekształcenia mamy sygnał zespolony tak w dziedzinie czasu, jak i częstotliwości. W wersjach rzeczywistych mamy zaś sygnał rzeczywisty w dziedzinie czasu i dwa sygnały rzeczywiste w dziedzinie częstotliwości. W przypadku przekształceń zespolonych stosuje się zarówno częstotliwości dodatnie, jak i ujemne, podczas gdy w przekształceniach rzeczywistych są stosowane tylko częstotliwości dodatnie. Przekształcenia zespolone są zwykle zapisywane w postaci wykładniczej. Można jednak skorzystać z zależności Eulera, aby – w razie potrzeby – przetworzyć je do postaci kosinusoidalnej lub sinusoidalnej.

3. Analiza i synteza

Dla każdego przekształcenia mamy równania analizy (zwane też przekształceniem prostym) oraz syntezy (zwane też przekształceniem odwrotnym). Równania analizy opisują sposób obliczania każdej wartości w dziedzinie częstotliwości na podstawie wszystkich wartości w dziedzinie czasu. Równania syntezy zaś opisują obliczanie każdej wartości w dziedzinie czasu na podstawie wszystkich wartości w dziedzinie częstotliwości.

4. Zapis w dziedzinie czasu

W dziedzinie czasu sygnały ciągłe są oznaczane jako $x(t)$, a sygnały dyskretne jako $x[n]$. W przekształceniach zespolonych te sygnały są zespolone, a w rzeczywistych – rzeczywiste. Wszystkie sygnały w dziedzinie czasu mają zakres od minus nieskończoności do plus nieskończoności. W przypadku jednak, gdy sygnał w dziedzinie czasu jest okresowy, jesteśmy zainteresowani tylko pojedynczym okresem, gdyż reszta jest informacją nadmiarową. Zmienne T i N oznaczają, odpowiednio, okres sygnału ciągłego i dyskretnego w dziedzinie czasu.

5. Zapis w dziedzinie częstotliwości

Sygnały ciągłe w dziedzinie częstotliwości są oznaczane jako $X(\omega)$, jeśli są zespolone, a $Re\ X(\omega)$ i $Im\ X(\omega)$, gdy są rzeczywiste. Sygnały dyskretne w dziedzinie częstotliwości są oznaczane jako $X[k]$, jeśli są zespolone oraz $Re\ X[k]$ i $Im\ X[k]$, gdy są rzeczywiste. W przekształceniach zespolonych występują częstotliwości ujemne o zakresie od minus nieskończoności do zera oraz częstotliwości dodatnie – od zera do plus nieskończoności. W przekształceniach rzeczywistych stosuje się tylko częstotliwości dodatnie. Jeśli sygnał w dziedzinie częstotliwości jest okresowy, to jesteśmy zainteresowani tylko pojedynczym okresem, gdyż reszta jest informacją nadmiarową. Dla sygnałów ciągłych w dziedzinie częstotliwości pełny okres zmiennej niezależnej ω zawiera się w granicach od $-\pi$ do π. Dla sygnałów dyskretnych stosujemy okres, w którym k zmienia się od 0 do $N-1$.

6. Równania analizy

Równania analizy operują na zasadzie *korelacji*, tzn. mnożenia sygnału w dziedzinie czasu przez sinusoidę i całkowania (sygnału ciągłego w dziedzinie czasu) lub sumowania (sygnału dyskretnego w dziedzinie czasu) we właściwym zakresie czasu. Jeśli sygnał w dziedzinie czasu jest nieokresowy, to ten zakres czasu będzie się rozciągał od minus do plus nieskończoności. Jeśli zaś sygnał jest okresowy, to zakresem właściwym będzie pełny okres sygnału. Podane w zestawieniu równania zapisano z całkowaniem (lub sumowaniem) w pełnym okresie od 0 do T (lub od 0 do $N-1$). Taki sam wynik dałby jednak każdy inny pełny okres, tj. od $-T$ do 0, od $-T/2$ do $T/2$ itd.

7. Równania syntezy

Równania syntezy opisują, jak jest obliczana pojedyncza wartość w dziedzinie czasu na podstawie wszystkich punktów w dziedzinie częstotliwości. Wykonuje się to, mnożąc sygnał w dziedzinie częstotliwości przez sinusoidę i całkując (dla ciągłego sygnału w dziedzinie częstotliwości) lub sumując (dla sygnału dyskretnego) we właściwym zakresie częstotliwości. Jeśli sygnał w dziedzinie częstotliwości jest zespolony i nieokresowy, to właściwy zakres częstotliwości rozciąga się od minus do plus nieskończoności. Jeśli zaś sygnał w dziedzinie częstotliwości jest zespolony i okresowy, to właściwym zakresem będzie pełny okres, czyli od $-\pi$ do π (dla sygnału ciągłego w dziedzinie częstotliwości) lub od 0 do $N-1$ (dla sygnału dyskretnego). Jeśli sygnał w dziedzinie częstotliwości jest rzeczywisty i nieokresowy, to właściwy będzie zakres od 0 do plus nieskończoności, a więc obejmujący tylko częstotliwości dodatnie. Na koniec, jeśli sygnał w dziedzinie częstotliwości jest rzeczywisty i okresowy, to właściwy zakres obejmuje połowę okresu zawierającą częstotliwości dodatnie, tj. albo do 0 do π (dla sygnału ciągłego w dziedzinie częstotliwości), albo od 0 do $N/2$ (dla sygnału dyskretnego).

Tab. 31.1. Przekształcenia Fouriera

Dyskretne przekształcenie Fouriera (DFT)

Przekształcenie zespolone	Przekształcenie rzeczywiste
Synteza $x[n]=\sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi kn}{N}}$	Synteza $x[n]=\sum_{k=0}^{N/2} Re\,X[k]\cos\left(\frac{2\pi kn}{N}\right) - Im\,X[k]\sin\left(\frac{2\pi kn}{N}\right)$
Analiza $X[k]=\frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{\frac{-j2\pi kn}{N}}$	Analiza $Re\,X[k]=\frac{2}{N}\sum_{n=0}^{N-1} x[n]\cos\left(\frac{2\pi kn}{N}\right)$ $Im\,X[k]=\frac{-2}{N}\sum_{n=0}^{N-1} x[n]\sin\left(\frac{2\pi kn}{N}\right)$
Dziedzina czasu: x[n] jest zespolony, dyskretny i okresowy n zmienia się w ciągu jednego okresu, od 0 do N – 1	Dziedzina czasu: x[n] jest rzeczywisty, dyskretny i okresowy n zmienia się od w ciągu jednego okresu, od 0 do N – 1
Dziedzina częstotliwości: X[k] jest zespolony, dyskretny i okresowy k zmienia się w ciągu jednego okresu, od 0 do N – 1 dla k od 0 do N/2 częstotliwości są dodatnie dla k od N/2 do N – 1 częstotliwości są ujemne	Dziedzina częstotliwości: Re X[k] jest rzeczywisty, dyskretny i okresowy Im X[k] jest rzeczywisty, dyskretny i okresowy k zmienia się w ciągu połowy okresu, od 0 do N/2 Uwaga: Przed zastosowaniem równania syntezy wartości Re X[0] i Re X[N/2] należy podzielić przez dwa.

Ciągłe w czasie przekształcenie Fouriera (CTFT)

Przekształcenie zespolone	Przekształcenie rzeczywiste
Synteza $x[n]=\int_0^{2\pi} X(\omega)e^{j\omega n}d\omega$	Syntèza $x[n]=\int_0^{\pi} Re\,X(\omega)\cos(\omega n) - Im\,X(\omega)\sin(\omega n)d\omega$
Analiza $X(\omega)=\frac{1}{2\pi}\sum_{n=-\infty}^{+\infty} x[n]e^{-j\omega n}$	Analiza $Re\,X(\omega)=\frac{1}{\pi}\sum_{n=-\infty}^{+\infty} x[n]\cos(\omega n)$ $Im\,X(\omega)=\frac{-1}{\pi}\sum_{n=-\infty}^{+\infty} x[n]\sin(\omega n)$
Dziedzina czasu: x[n] jest zespolony, dyskretny i nieokresowy n zmienia się od minus do plus nieskończoności	Dziedzina czasu: x[n] jest rzeczywisty, dyskretny i nieokresowy n zmienia się od minus do plus nieskończoności
Dziedzina częstotliwości: X(ω) jest zespolony, ciągły i okresowy ω zmienia się w ciągu jednego okresu, od 0 do 2π dla ω od 0 do π częstotliwości są dodatnie dla ω od π do 2π częstotliwości są ujemne	Dziedzina częstotliwości: Re X(ω) jest rzeczywisty, ciągły i okresowy Im X(ω) jest rzeczywisty, ciągły i okresowy ω zmienia się w ciągu połowy okresu, od 0 do π

Szereg Fouriera

Przekształcenie zespolone	Przekształcenie rzeczywiste
Synteza $x(t)=\sum_{k=-\infty}^{+\infty} X[k]e^{\frac{j2\pi kt}{T}}$	Synteza $x(t)=\sum_{k=0}^{+\infty} Re\,X[k]\cos\left(\frac{2\pi kt}{T}\right) - Im\,X[k]\sin\left(\frac{2\pi kt}{T}\right)$
Analiza $X[k]=\frac{1}{T}\int_0^T x(t)e^{\frac{-j2\pi kt}{T}}dt$	Analiza $Re\,X[k]=\frac{2}{T}\int_0^T x(t)\cos\left(\frac{2\pi kt}{T}\right)dt$ $Im\,X[k]=\frac{-2}{T}\int_0^T x(t)\sin\left(\frac{2\pi kt}{T}\right)dt$
Dziedzina czasu: x(t) jest zespolony, ciągły i okresowy t zmienia się w ciągu jednego okresu, od 0 do T	Dziedzina czasu: x(t) jest rzeczywisty, ciągły i okresowy t zmienia się w ciągu jednego okresu, od 0 do T
Dziedzina częstotliwości: X[k] jest zespolony, dyskretny i nieokresowy k zmienia się od minus do plus nieskończoności dla k > 0 częstotliwości są dodatnie dla k < 0 częstotliwości są ujemne	Dziedzina częstotliwości: Re X[k] jest rzeczywisty, dyskretny i nieokresowy Im X[k] jest rzeczywisty, dyskretny i nieokresowy k zmienia się od minus do plus nieskończoności Uwaga: Przed zastosowaniem równania syntezy wartość Re X[0] należy podzielić przez dwa.

Przekształcenie Fouriera

Przekształcenie zespolone	Przekształcenie rzeczywiste
Synteza $x(t)=\int_{-\infty}^{+\infty} X(\omega)e^{j\omega t}d\omega$	Synteza $x(t)=\int_0^{+\infty} Re\,X(\omega)\cos(\omega t) - Im\,X(\omega)\sin(\omega t)dt$
Analiza $X(\omega)=\frac{1}{2\pi}\int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt$	Analiza $Re\,X(\omega)=\frac{1}{\pi}\int_{-\infty}^{+\infty} x(t)\cos(\omega t)dt$ $Im\,X(\omega)=\frac{-1}{\pi}\int_{-\infty}^{+\infty} x(t)\sin(\omega t)dt$
Dziedzina czasu: x(t) jest zespolony, ciągły i nieokresowy t zmienia się od minus do plus nieskończoności	Dziedzina czasu: x(t) jest rzeczywisty, ciągły i nieokresowy t zmienia się od minus do plus nieskończoności
Dziedzina częstotliwości: X(ω) jest zespolony, ciągły i nieokresowy ω zmienia się od minus do plus nieskończoności dla ω > 0 częstotliwości są dodatnie dla ω < 0 częstotliwości są ujemne	Dziedzina częstotliwości: Re X(ω) jest rzeczywisty, ciągły i nieokresowy Im X(ω) jest rzeczywisty, ciągły i nieokresowy ω zmienia się od minus do plus nieskończoności

8. Skalowanie

Aby równania analizy i syntezy się nawzajem anulowały, trzeba do jednego z nich dodać współczynnik skalowania. W tablicy 31.1 współczynniki skalowania umieściliśmy przy równaniach analizy. W przypadku przekształceń zespolonych współczynniki skalowania są: $1/N$, $1/T$ lub $1/2\pi$. Ponieważ w przekształceniach rzeczywistych nie stosuje się częstotliwości ujemnych, więc współczynniki skalowania są dwukrotnie większe: $2/N$, $2/T$ lub $1/\pi$. Przekształcenia rzeczywiste zawierają też ujemny znak w obliczeniu części urojonej widma częstotliwościowego (opcja stosowana w celu osiągnięcia większej spójności przekształceń rzeczywistych z zespolonymi). Wreszcie równania syntezy dla rzeczywistego DFT i rzeczywistego szeregu Fouriera są zaopatrzone w specjalne zalecenie skalowania dotyczące $Re\ X(0)$ i $Re\ X(N/2)$.

9. Różnice

Te równania w innych publikacjach mogą wyglądać nieco inaczej. Oto kilka różnic, na które należy zwrócić uwagę:

- Stosowanie f zamiast ω w zależności $\omega = 2\pi f$
- Całkowanie w innych okresach, np. od $-T$ do 0, od $-T/2$ do $T/2$ lub od 0 do T
- Przeniesienie, całkowicie lub częściowo, współczynnika skalowania do równania syntezy
- Zastąpienie okresu częstotliwością podstawową: $f_0 = 1/T$
- Stosowanie innych oznaczeń zmiennych, np. zamiast ω może być Ω w DTFT, a $Re\ X[k]$ i $Im\ X[k]$ mogą być oznaczone jako a_k i b_k w szeregu Fouriera.

Dlaczego stosuje się przekształcenie Fouriera

Z tego rozdziału wynika bezsprzecznie, że zespolone DFT jest znacznie bardziej skomplikowane niż rzeczywiste DFT. Czy korzyści wynikające z zespolonego DFT są rzeczywiście warte wysiłku uczenia się zawiłej matematyki? Odpowiedź na takie pytanie zależy od tego, kim jesteście i do czego macie zamiar zastosować DSP. Podstawowym przesłaniem tej książki jest udowodnienie, że najpraktyczniejsze metody DSP można zrozumieć i stosować bez uciekania się do przekształceń zespolonych. Jeśli uczycie się DSP po to tylko, aby was wspomogło w waszych badaniach i projektach technicznych, to poznawanie zespolonego DFT byłoby chyba niepotrzebną przesadą.

Niemniej jednak, matematyka liczb zespolonych jest głównym językiem dla tych, którzy specjalizują się w DSP. Jeśli nie rozumiecie tego języka, to nie możecie porozumieć się z profesjonalistami w tej dziedzinie. To dotyczy także możliwości zrozumienia literatury o DSP – książek, artykułów, publikacji technicznych itd. Dlaczego metody zespolone są tak popularne wśród rzeszy profesjonalistów DSP?

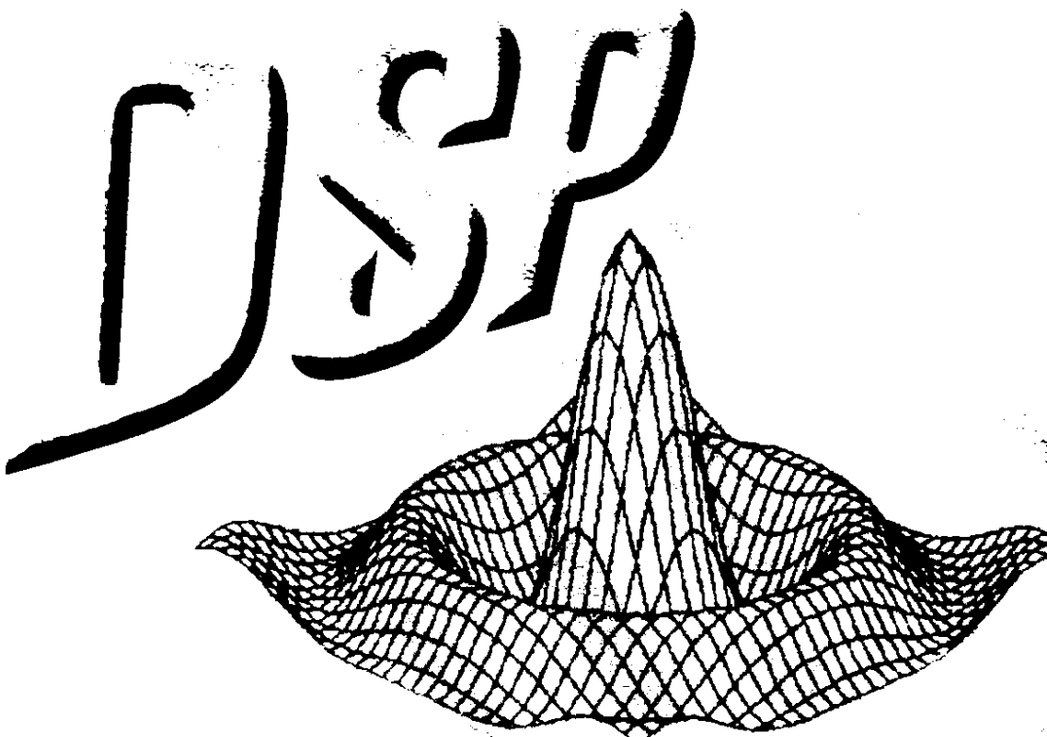
Jest kilka przyczyn, które już wymieniliśmy: wzory o zwartej formie, symetria równań analizy i syntezy, symetria dziedzin czasu i częstotliwości, włączenie częstotliwości ujemnych oraz fakt, że te metody są krokiem prowadzącym do przekształceń Z i Laplace'a, itd.

Jest jeszcze bardziej filozoficzna przyczyna, o której nie mówiliśmy, a którą czasem nazywa się *prawdą*. Rozpoczęliśmy ten rozdział od wyszczególnienia kilku przyczyn, dla których rzeczywiste przekształcenie Fouriera jest kłopotliwe. Po wprowadzeniu zespolonego przekształcenia Fouriera te kłopoty znikły. Powiedzieliśmy: świetnie, zespolone przekształcenie Fouriera rozwiązało trudności.

Chociaż to prawda, to jednak nie w pełni oddaje zalety zespolonego przekształcenia Fouriera. Spójrzmy na tę sprawę w sposób następujący. Mimo swej abstrakcyjnej natury, zespolone przekształcenie Fouriera *prawidłowo* opisuje, jak się zachowują systemy fizyczne. Jeśli operacje matematyczne ograniczamy do liczb rzeczywistych, to pojawiają się problemy. Można więc powiedzieć, że te problemy nie są *rozwiązywane* przez zespolone przekształcenie Fouriera, lecz są one *wprowadzane* przez rzeczywiste przekształcenie Fouriera. Matematycznie, przekształcenie zespolone daje więcej prawdy niż rzeczywiste. W tym jest wielka zasługa matematyków i innych uczonych – grupy, która rozszerza ludzką wiedzę, a nie tylko po prostu rozwiązuje problemy bieżące.

32

Przekształcenie Laplace'a



Dwie główne metody stosowane w przetwarzaniu sygnałów – splot i analiza Fouriera – uczą nas, że działanie systemu liniowego może być w pełni rozumiane na podstawie jego odpowiedzi impulsowej lub częstotliwościowej. Takie podejście jest bardzo uogólnione, gdyż te odpowiedzi mogą przyjmować prawie każdy możliwy kształt lub postać. W istocie, jest to podejście *zbyt* ogólne dla wielu zastosowań w nauce i technice. Wzajemne oddziaływania wielu parametrów w naszym Wszechświecie są opisane równaniami różniczkowymi. Na przykład napięcie na cewce indukcyjnej jest proporcjonalne do pochodnej prądu płynącego w tej cewce. Siła przyłożona do ciała o pewnej masie jest proporcjonalna do pochodnej jego prędkości. W fizyce jest mnóstwo zależności tego rodzaju. Odpowiedzi częstotliwościowa i impulsowa tych systemów nie mogą być dowolne, lecz powinny być zgodne z rozwiązaniami tych równań różniczkowych. To znaczy, że ich odpowiedzi impulsowe mogą składać się tylko z *przebiegów wykładniczych* i *sinusoidalnych*. Przekształcenie Laplace'a jest metodą analizy tych szczególnych systemów w przypadku sygnałów *ciągłych*. Przekształcenie Z jest podobną metodą stosowaną w przypadku sygnałów *dyskretnych*.

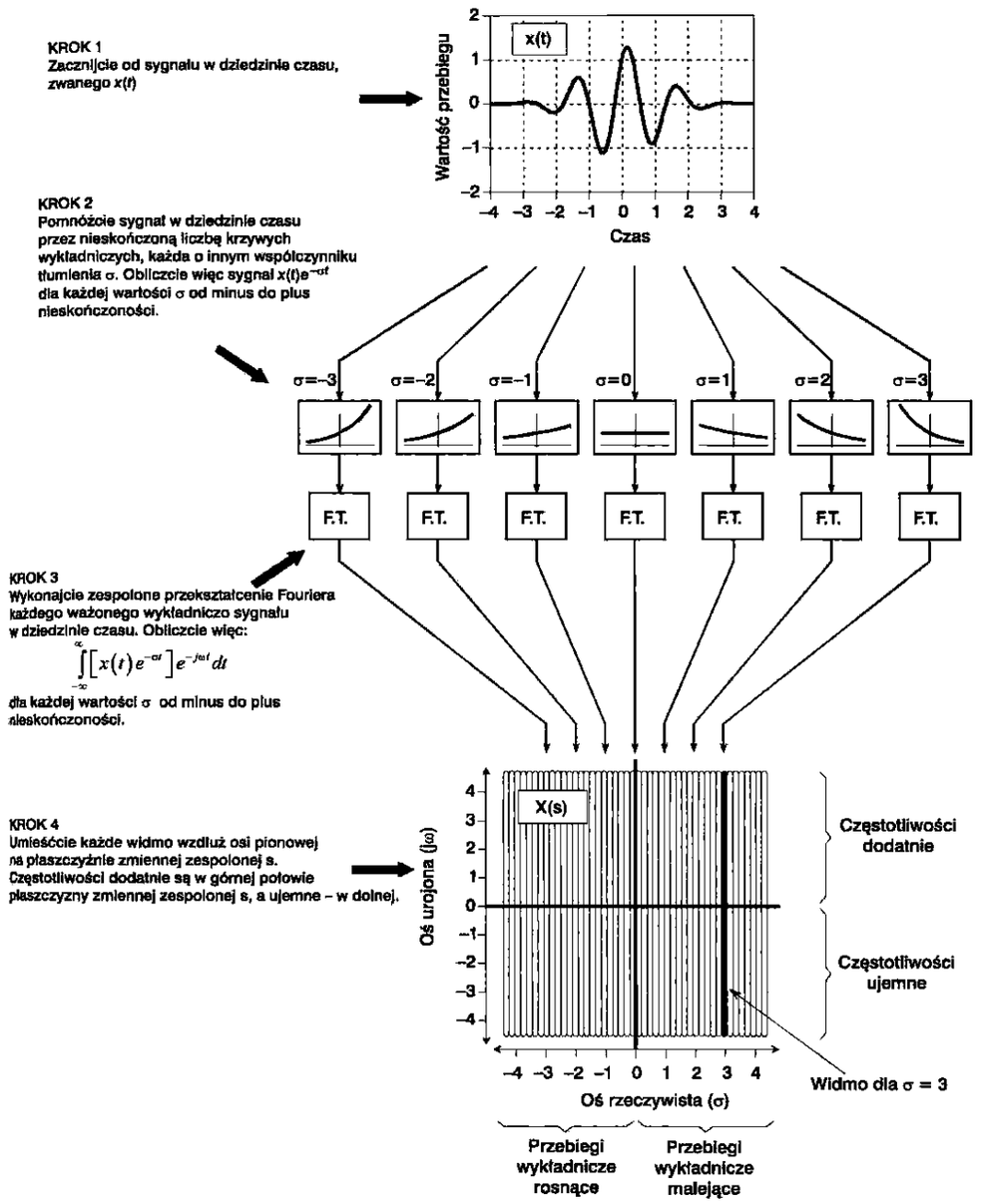
Właściwości dziedziny *s*

Przekształcenie Laplace'a jest dobrze znaną matematyczną metodą rozwiązywania równań różniczkowych. Metodę nazwano na cześć wielkiego matematyka francuskiego Pierre'a Simona de Laplace'a (1749–1827). Podobnie jak inne transformaty, przekształcenie Laplace'a zmienia jeden sygnał w inny, zgodnie z pewnym ustalonym zestawem reguł lub równań. Przekształcenia Laplace'a zamienia sygnał w dziedzinie czasu na sygnał w **dziedzinie *s***, zwanej też **płaszczyzną zmiennej zespolonej *s*** (rys. 32.1). Sygnał w dziedzinie czasu jest ciągły i ma zakres od minus do plus nieskończoności i może być okresowy lub nieokresowy. W przekształceniu Laplace'a sygnał w dziedzinie czasu może być *zespolony*, lecz w dziedzinie przetwarzania sygnałów rzadko jest to potrzebne. W tym omówieniu i w prawie wszystkich zastosowaniach praktycznych sygnał w dziedzinie czasu jest całkowicie rzeczywisty.

Dziedzina *s*, jak to przedstawiono na rys. 32.1, jest płaszczyzną zmiennej zespolonej, tzn. są tam liczby rzeczywiste wzdłuż osi poziomej i urojone wzdłuż pionowej. Odległość wzdłuż osi rzeczywistej jest wyrażana jako zmienna σ, a wzdłuż urojonej – jako ω, czyli pulsacja. Ten układ współrzędnych umożliwia oznaczenie położenia dowolnego punktu przez współrzędne σ i ω. Przy użyciu zapisu zespolonego każde położenie jest reprezentowane zmienną zespoloną *s*, przy czym $s = \sigma + j\omega$. Podobnie jak w przekształceniu Fouriera, sygnały w dziedzinie *s* są zapisywane dużymi literami. Na przykład sygnał w dziedzinie czasu $x(t)$ jest przekształcany na sygnał $X(s)$ w dziedzinie *s*, lub zapisując inaczej $X(\sigma, \omega)$. Płaszczyzna zmiennej zespolonej *s* jest ciągła i rozciąga się do nieskończoności we wszystkich czterech kierunkach.

Każdy punkt, oprócz położenia na płaszczyźnie zmiennej zespolonej *s* określonego przez jakąś liczbę zespoloną, ma też *wartość*, która jest też liczbą zespoloną. Innymi słowy, każde położenie na płaszczyźnie zmiennej zespolonej *s* ma część rzeczywistą i urojoną. Tak jak dla wszystkich liczb zespolonych, zamiast części rzeczywistej i urojonej można określać moduł i fazę.

O ile w przekształceniu Fouriera analizuje się sygnały, wyrażając je sinusoidami, to w analizie z przekształceniem Laplace'a wyraża się je sinusoidami *oraz* funkcjami wykładniczymi. Z matematycznego punktu widzenia przekształcenie Fouriera jest więc *podzbiorem* bardziej skomplikowanego przekształcenia Laplace'a. Na rys. 32.1 zobrazowano graficznie zależność między dziedziną *s* a dziedziną czasu. Aby znaleźć wartości wzdłuż osi pionowej na płaszczyźnie zmiennej zespolonej *s* (dla określonego σ), mnoży się najpierw sygnał w dziedzinie czasu przez przebieg wykładniczy $e^{-\sigma t}$. W lewej połówce płaszczyzny zmiennej zespolonej *s* następuje mnożenie sygnału w dziedzinie czasu przez przebiegi wykładnicze *rosnące* w funkcji czasu ($\sigma < 0$), a w prawej – przez *malejące* w funkcji czasu ($\sigma > 0$). Następnie wykonuje się zespolone przekształcenie Fouriera sygnału ważonego wykładniczo. Uzyskane widmo umieszcza się wzdłuż linii pionowej na płaszczyźnie zmiennej zespolonej *s*, przy czym górna połówka płaszczyzny zmiennej zespolonej *s* zawiera częstotliwości dodatnie,



Rys. 32.1. Przekształcenie Laplace'a. To przekształcenie daje konwersję sygnału x(t) w dziedzinie czasu na sygnał X(s) lub X(σ,ω) w dziedzinie s. Wartości wzdłuż każdej linii pionowej na płaszczyźnie zmiennej zespolonej s mogą być wyznaczone przez wymnożenie sygnału w dziedzinie czasu przez krzywą wykładniczą o współczynniku tłumienia σ i wykonanie zespolonego przekształcenia Fouriera. Jeśli sygnał w dziedzinie czasu jest całkowicie rzeczywisty, to górna połowa płaszczyzny zmiennej zespolonej s jest lustrzanym odbiciem dolnej

a dolna – ujemne. Zwróćcie szczególną uwagę na fakt, że wartości na osi y płaszczyzny zmiennej zespolonej s ($\sigma = 0$) są dokładnie równe wartościom przekształcenia Fouriera sygnału w dziedzinie czasu.

Jak podano w poprzednim rozdziale, zespolone przekształcenie Fouriera jest opisane wzorem:

$$X(\omega)=\int_{-\infty}^{\infty}x(t)e^{-j\omega t}dt$$

To wyrażenie można rozwinąć w przekształcenie Laplace'a, mnożąc najpierw sygnał w dziedzinie czasu przez wyraz wykładniczy:

$$X(\sigma,\omega)=\int_{-\infty}^{\infty}\left[x(t)e^{-\sigma t}\right]e^{-j\omega t}dt$$

Chociaż nie jest to najprostsza postać przekształcenia Laplace'a, jest to przypuszczalnie najlepszy opis strategii i sposobu działania w tej metodzie.

W celu uzyskania prostszej postaci tego wzoru można połączyć ze sobą oba wyrazy wykładnicze:

$$X(\sigma,\omega)=\int_{-\infty}^{\infty}x(t)\,e^{-(\sigma+j\omega)t}dt$$

Wreszcie, *położenie* na płaszczyźnie zmiennej zespolonej może być reprezentowane przez zmienną zespoloną s, przy czym $s = \sigma + j\omega$. Dzięki temu wzór można zredukować do jeszcze bardziej zwartej postaci:

$$X(s)=\int_{-\infty}^{\infty}x(t)\,e^{-st}dt$$

Wzór 32.1. Przekształcenie Laplace'a. Ten wzór określa zależność między sygnałem w dziedzinie czasu x(t) a sygnałem X(s) w dziedzinie s. Zmienne w dziedzinie s, a więc s i X( ) są zespolone. Sygnał w dziedzinie czasu zaś może być zespolony, ale na ogół jest rzeczywisty

Jest to końcowa postać przekształcenia Laplace'a – jeden z najważniejszych wzorów w dziedzinie przetwarzania sygnałów i w elektronice. Szczególną uwagę zwróćcie na wyraz e^{-st}, zwany *zespoloną funkcją wykładniczą*. Jak wynika z powyższego wyprowadzenia, dzięki zespolonej funkcji wykładniczej można w najbardziej zwartej postaci jednym wzorem wyrazić zarówno sinusoidy, jak i funkcje wykładnicze.

Chociaż objaśniliśmy przekształcenie Laplace'a jako pewien proces dwustopniowy (mnożenie przez krzywą wykładniczą, a po nim przekształcenie Fouriera), to pamiętajcie, że ma to być tylko ułatwieniem w uczeniu się i sposobem podziału wzoru 32.1 na prostsze składniki. Przekształcenie Laplace'e jest jednym równaniem wyrażającym zależność między $x(t)$ i $X(s)$, a nie procedurą wielokrokową. Wzór 32.1 określa, jak obliczać każdy *punkt* na płaszczyźnie zmiennej zespolonej s (opisany współrzędnymi σ i ω) na podstawie wartości σ i ω oraz sygnału w dziedzinie czasu $x(t)$. Zastosowanie przekształcenia Fouriera do *jednoczesnego* obliczenia wszystkich punktów wzdłuż osi pionowej jest po prostu udogodnieniem, ale nie wymogiem. Jednak trzeba koniecznie pamiętać, że wartości na płaszczyźnie zmiennej zespolonej s wzdłuż osi y ($\sigma = 0$) są *dokładnie* równe wartościom przekształcenia Fouriera. Jest to podstawowa cecha, dzięki której przekształcenie Laplace'a jest tak użyteczne. Będziemy jeszcze o tym mówić w dalszej części tego rozdziału.

W celu lepszego poznania natury wzoru 32.1, wybierzmy kilka punktów na płaszczyźnie zmiennej zespolonej s i sprawdźmy, jak wartości w tych punktach są powiązane z sygnałem w dziedzinie czasu. Najpierw przypomnijcie sobie, w jaki sposób poszczególne punkty *w dziedzinie częstotliwości* zależą od sygnału w dziedzinie czasu. Każdy punkt w dziedzinie częstotliwości, identyfikowany przez jakąś specyficzną wartość ω, odpowiada dwóm sinusoidom $\cos(\omega t)$ i $\sin(\omega t)$. Część rzeczywistą wyznacza się, mnożąc sygnał w dziedzinie czasu przez falę kosinusoidalną i następnie całkując uzyskane wyrażenie w granicach od $-\infty$ do ∞. Część urojoną wyznacza się w taki sam sposób, lecz z wykorzystaniem fali sinusoidalnej. Jeśli mamy do czynienia z *zespolonym* przekształceniem Fouriera, to dla częstotliwości

ujemnych, $-\omega$, mamy zespolone wartości sprzężone (taka sama część rzeczywista, ujemna część urojona) w stosunku do wartości dla częstotliwości dodatnich ω. Przekształcenie Laplace'a jest po prostu rozszerzeniem tych samych koncepcji.

Na rys. 32.2 pokazano trzy *pary* punktów na płaszczyźnie zmiennej zespolonej s: A i A', B i B' oraz C i C'. Tak samo jak w zespolonym widmie częstotliwościowym, punkty A, B i C (częstotliwości dodatnie) są zespolonymi wartościami sprzężonymi punktów A', B' i C' (częstotliwości ujemne). Górna połowa płaszczyzny zmiennej zespolonej s jest lustrzanym odbiciem dolnej połowy i obie połowy są konieczne do uzyskania odpowiedniości z rzeczywistym sygnałem w dziedzinie czasu. Inaczej mówiąc, traktując te punkty jako pary, omija się matematykę liczb zespolonych, dzięki czemu możemy operować w dziedzinie czasu, stosując liczby tylko rzeczywiste.

Ponieważ każda z tych par punktów ma określone wartości σ i $\pm\omega$, więc istnieją dwa przebiegi związane z każdą parą: $\cos(\omega t)\,e^{-\sigma t}$ oraz $\sin(\omega t)\,e^{-\sigma t}$. Punkty, np. A i A' mają położenie $\sigma = 1{,}5$ i $\omega = \pm 40$, a więc odpowiadają im przebiegi: $\cos(40t)\,e^{-1,5t}$ oraz $\sin(40t)\,e^{-1,5t}$. Są to sinusoidy o amplitudzie *tłumionej* wykładniczo w funkcji czasu (rys. 32.2). Natomiast fale sinusoidalna i kosinusoidalna związane z punktami B i B' mają *stałą* amplitudę, gdyż σ jest równa zero. Amplituda zaś fal sinusoidalnej i kosinusoidalnej związanych z punktami C i C' *wzrasta*, bo σ jest ujemna.

Wartość każdego położenia na płaszczyźnie zmiennej zespolonej s składa się z *części rzeczywistej* i *części urojonej*. Część rzeczywistą określa się, mnożąc sygnał w dziedzinie czasu przez kosinusoidę ważoną wykładniczo i następnie wykonując całkowanie w granicach od $-\infty$ do ∞. Część urojoną wyznacza się w ten sam sposób, stosując jednak wykładniczo ważoną falę sinusoidalną. Można to przedstawić w postaci równania na przykładzie części rzeczywistej A i A':

$$Re\,X(\sigma=1{,}5\ \ \omega=\pm 40)=\int_{-\infty}^{\infty}x(t)\cos(40t)e^{-1,5t}dt$$

Na rys. 32.3 przedstawiono przykład przebiegu w dziedzinie czasu, jego widmo częstotliwościowe i jego reprezentację w dziedzinie s. Przykładowy sygnał jest impulsem prostokątnym o szerokości *dwa* i wysokości *jeden*. Jak wynika z rysunku, zespolona transformata Fouriera tego sygnału jest funkcją *sinc* w części rzeczywistej i sygnałem całkowicie zerowym w części urojonej. Na płaszczyźnie zmiennej zespolonej s jest to pofalowany sygnał dwuwymiarowy, pokazany tu jako powierzchnie topograficzne części rzeczywistej i urojonej. Można to wyrazić matematycznie:

$$X(s)=\int_{-\infty}^{\infty}x(t)\,e^{-st}dt=\int_{-1}^{1}1\,e^{-st}dt$$

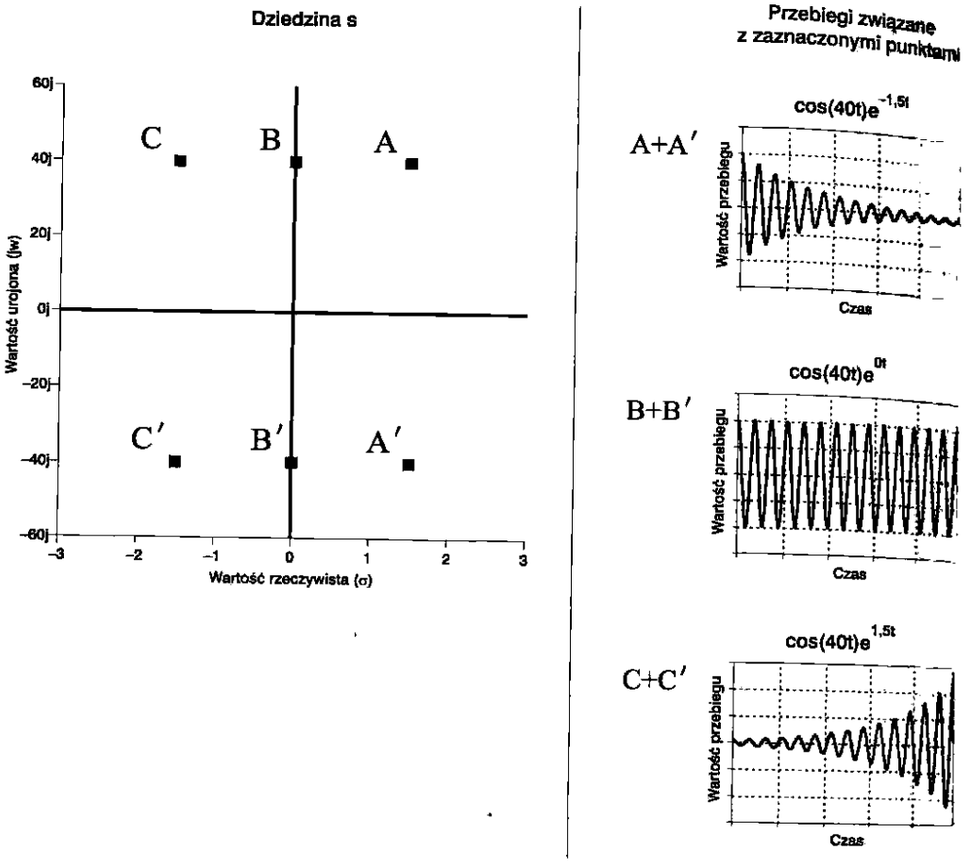
Rozpoczynamy tu od definicji przekształcenia Laplace'a (wzór 32.1), wstawiamy wartość jednostkową jako $x(t)$ i zmieniamy granice tak, aby były dopasowane do długości niezerowej części sygnału w dziedzinie czasu. Obliczenie całki daje sygnał w dziedzinie s, wyrażony jako zespolone położenie s i wartość zespolona $X(s)$:

$$X(s)=\frac{e^{s}-e^{-s}}{s}$$

Chociaż jest to najbardziej zwarta postać odpowiedzi, to jednak zastosowanie zmiennych zespolonych powoduje, że jest ona trudna do zrozumienia i niemożliwa do zobrazowania tak jak na rys. 32.3. Rozwiązaniem tych niedogodności jest zastąpienie zmiennej zespolonej s przez $\sigma + j\omega$ i rozdzielenie następnie części rzeczywistej od urojonej:

$$Re\,X(\sigma,\omega)=\frac{\sigma\cos(\omega)\left[e^{\sigma}-e^{-\sigma}\right]+\omega\sin(\omega)\left[e^{\sigma}+e^{-\sigma}\right]}{\sigma^{2}+\omega^{2}}$$

$$Im\,X(\sigma,\omega)=\frac{\sigma\sin(\omega)\left[e^{\sigma}+e^{-\sigma}\right]-\omega\cos(\omega)\left[e^{\sigma}-e^{-\sigma}\right]}{\sigma^{2}+\omega^{2}}$$

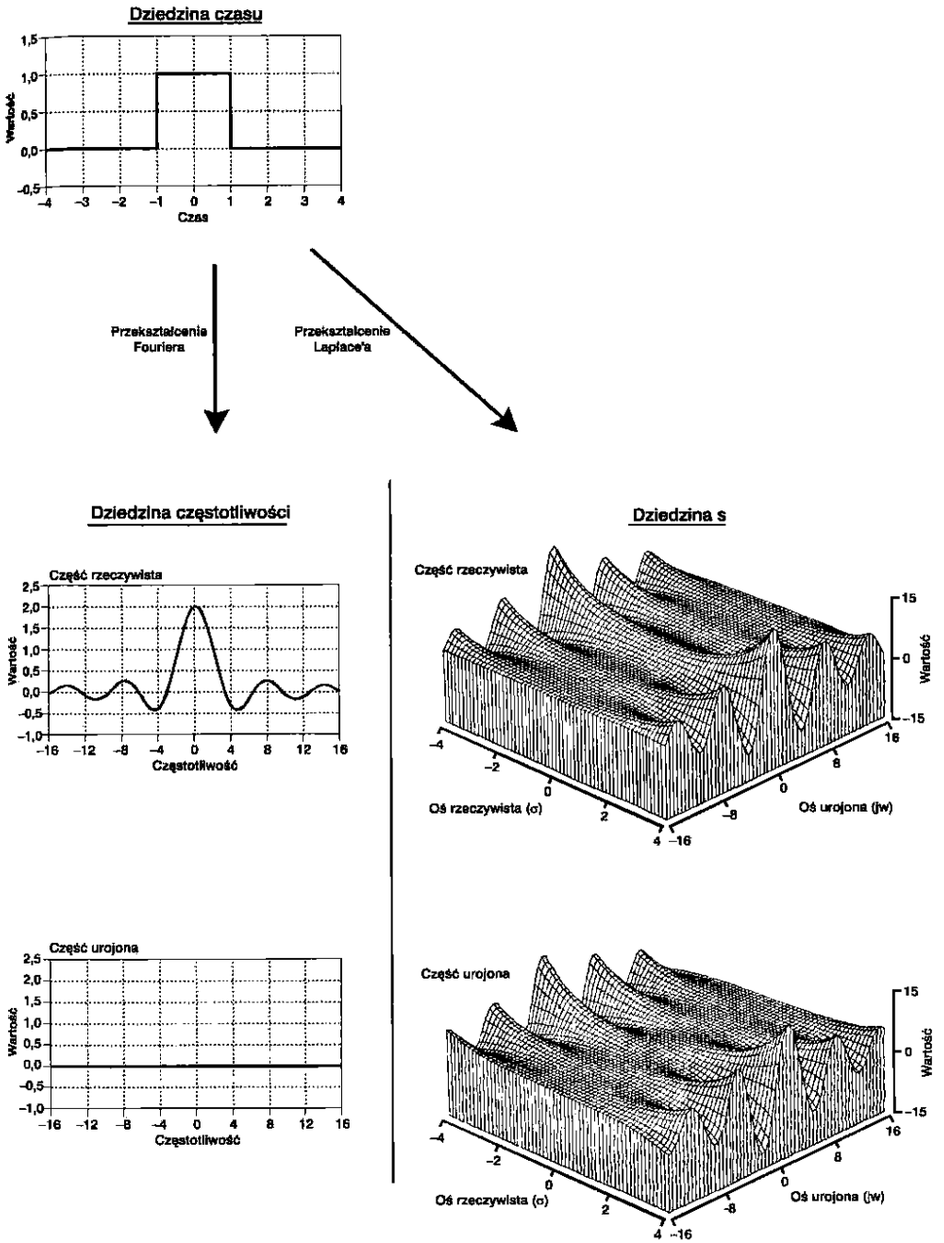


Rys. 32.2. Powiązanie przebiegów czasowych z płaszczyzną zmiennej zespolonej s. Każde położenie na tej płaszczyźnie jest identyfikowane dwoma parametrami: σ i ω. Te parametry definiują też dwa przebiegi związane z każdym położeniem. Jeśli rozważamy tylko pary punktów (takie jak A i A', B i B', C i C'), to dwa przebiegi związane z każdym położeniem na płaszczyźnie zmiennej zespolonej s są falami sinusoidalną i kosinusoidalną o częstotliwości ω i o amplitudzie zmieniającej się wykładniczo według stałej σ

Powierzchnie topograficzne na rys. 32.3 są wykresami powyższych równań. Te wzory są dość długie, a ich matematyczne wyprowadzenie jest bardzo żmudne. To prowadzi do kwestii praktycznej – skąd mamy pewność, czy nie zrobiliśmy błędu w obliczeniach, stosując algebrę liczb zespolonych? Sprawdzianem jest zweryfikowanie, czy te równania redukują się do *transformaty Fouriera* wzdłuż osi *y*. Robi się to, wstawiając w równaniach σ równe zero i je upraszczając:

$$Re\,X(\sigma,\omega)\Big|_{\sigma=0} = \frac{2\sin(\omega)}{\omega} \qquad Im\,X(\sigma,\omega)\Big|_{\sigma=0} = 0$$

Są to, jak pokazano na rys. 32.3, prawidłowe sygnały w dziedzinie częstotliwości, takie same jak wyznaczone przez bezpośrednie obliczenie transformaty Fouriera przebiegu w dziedzinie czasu.



Rys. 32.3. Dziedziny czasu, częstotliwości i dziedzina s. Sygnał w dziedzinie czasu (impuls prostokątny) jest przenoszony w dziedzinę częstotliwości przez przekształcenie Fouriera, a w dziedzinę s przez przekształcenie Laplace'a

Metodyka przekształcenia Laplace'a

Analogia dopomoże w wyjaśnieniu, jak przekształcenie Laplace'a jest wykorzystywane w przetwarzaniu sygnałów. Wyobraźcie sobie, że podróżujecie nocą pociągiem jadącym z jednego miasta do drugiego. Mapa wskazuje, że droga jest bardzo prosta, lecz noc jest tak ciemna, że nie widzicie otaczającego was krajobrazu. Nie mając niczego innego do roboty, zauważacie wysokościomierz na ścianie wagonu pasażerskiego i decydujecie się śledzić wzdłuż trasy zmiany wzniesienia nad poziom morza.

Po paru godzinach, znudziwszy się, zaczynacie rozmowę z konduktorem. „Ciekawy teren", mówicie. „Wydaje się, że na ogół się wznosimy, lecz zaobserwowałem kilka interesujących nieregularności". Nie zwracając uwagi na oczywisty brak zainteresowania ze strony konduktora, kontynuujecie: „Niedaleko początku naszej podróży minęliśmy coś w rodzaju gwałtownego wzrostu wysokości, po którym nastąpił równie gwałtowny spadek. Potem napotkaliśmy płytkie obniżenie". W obawie, że możecie być niespełna rozumu lub niebezpieczni, konduktor decyduje się na odpowiedź: „Tak, myślę, że to prawda. Cel naszej podróży znajduje się u podnóża pasma górskiego, sądząc na podstawie ciągłego wzrostu wzniesienia nad poziom morza. Jednak na naszej drodze przejeżdżamy przez obrzeża dużej góry i przez środek doliny".

Pomyślcie teraz, jak wy zrozumieliście zależność między wzniesieniem a odległością wzdłuż trasy, w porównaniu z tym, jak rozumie tę zależność konduktor. Ponieważ bezpośrednio mierzyliście wzniesienie nad poziom morza wzdłuż trasy, możecie twierdzić, że wiecie *wszystko* o tej zależności. Konduktor zaś ma tę samą pełną informację, lecz w prostszej i bardziej intuicyjnej formie – wie, gdzie są wzgórza i doliny, które *powodują* te garby i wgłębienia na drodze pociągu. Chociaż wasz opis sygnału może się składać z wyników tysięcy pojedynczych pomiarów, to opis konduktora może zawierać tylko kilka parametrów.

Aby zauważyć analogię tego opisu z przetwarzaniem sygnałów, wyobraźcie sobie, że próbujecie zrozumieć parametry jakiegoś układu elektrycznego. Pomagając sobie w badaniach, starannie mierzycie odpowiedź impulsową i/lub odpowiedź częstotliwościową. Jak wykazano w poprzednich rozdziałach, odpowiedzi impulsowa i częstotliwościowa zawierają *pełną* informację o danym systemie liniowym. Nie znaczy to jednak, że macie w ten sposób informację w jej *najprostszej* postaci. W szczególności, rozumiecie odpowiedź częstotliwościową jako zestaw wartości zmieniających się z częstotliwością. Tak jak w naszej analogii z pociągiem, odpowiedź częstotliwościową można łatwiej zrozumieć w kategoriach terenu *otaczającego* tę odpowiedź – a więc przez właściwości płaszczyzny zmiennej zespolonej *s*.

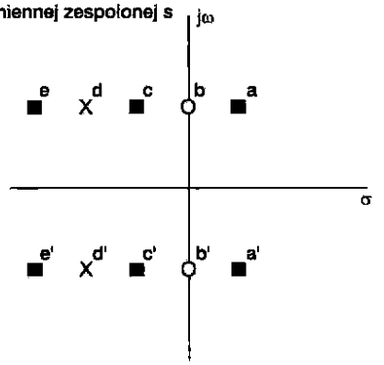
Mając w pamięci analogię z pociągiem, wróćcie do rys. 32.3 i zadajcie pytanie: jak kształt przebiegu na płaszczyźnie zmiennej zespolonej *s* dopomaga w zrozumieniu odpowiedzi częstotliwościowej? Odpowiedź jest: nie pomaga. W tym przykładzie przebieg sygnału na płaszczyźnie zmiennej zespolonej *s* tworzy ładny wykres, lecz nie daje żadnego wglądu w przyczyny takiego, a nie innego zachowania się sygnału w dziedzinie częstotliwości. Przyczyną jest to, że przekształcenie Laplace'a jest przeznaczone do analizy specyficznej klasy sygnałów w dziedzinie czasu: *odpowiedzi impulsowych, które składają się z sinusoid i funkcji wykładniczych*. Jeśli zostaje wyznaczona transformata Laplace'a jakiegoś innego przebiegu (takiego jak impuls prostokątny z rys. 32.3), to wynik uzyskany na płaszczyźnie zmiennej zespolonej *s* nie ma żadnego sensu.

Jak wspomniano we wprowadzeniu do tego rozdziału, systemy należące do wymienionej klasy niezwykle często występują w nauce i technice. Sinusoidy i funkcje wykładnicze są bowiem rozwiązaniami *równań różniczkowych*, a więc zależności opisujących większość zjawisk naszego świata fizycznego. Oto przykłady systemów rządzonych przez równania różniczkowe: obwody elektryczne, propagacja fal, ruch liniowy i obrotowy, pola elektryczne i magnetyczne oraz przepływ ciepła.

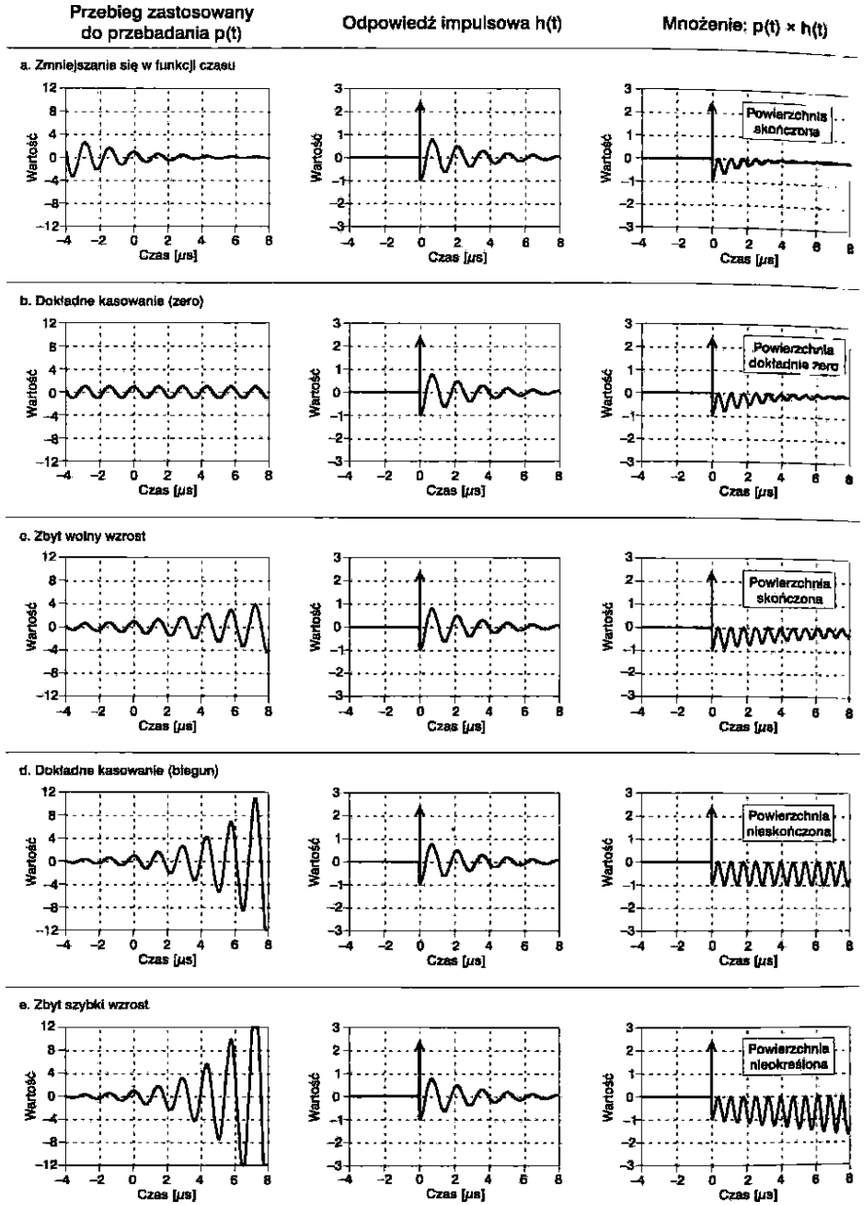
Przypuśćmy, że próbujemy zrozumieć pewien system liniowy, którego działanie opisują równania różniczkowe. Niech to będzie np. jakiś obwód elektryczny. Rozwiązanie równań różniczkowych jest matematycznym sposobem wyznaczenia odpowiedzi impulsowej. Alternatywnie możemy mierzyć odpowiedź impulsową, stosując odpowiednie generatory impulsów, oscyloskopy, rejestratory itd. Zanim poznamy świeżo wyznaczoną odpowiedź impulsową, spytajmy samych siebie, co *spodziewamy* się zobaczyć. Jest kilka właściwości przebiegu, które znamy nawet bez obejrzenia tej odpowiedzi. Po pierwsze, odpowiedź impulsowa musi być *przyczynowa*. Inaczej mówiąc, odpowiedź impulsowa musi mieć wartość zero, zanim sygnał wejściowy stanie się niezerowy w chwili $t = 0$. Jest to zasada przyczyny i skutku, na której oparty jest nasz Wszechświat.

Drugą, znaną nam właściwością odpowiedzi impulsowej jest fakt, że składa się ona z *sinusoid* i *funkcji wykładniczych*. Takie są bowiem rozwiązania równań różniczkowych rządzących działaniem systemu. Choćbyśmy nie wiem jak się starali, nigdy nie znajdziemy tego typu systemu mającego odpowiedź impulsową w postaci np. przebiegu prostokątnego lub trójkątnego. Trzecia znana właściwość to *nieskończona* długość odpowiedzi impulsowej. A więc ta odpowiedź ma wartości niezerowe od $t = 0$ do $t = +\infty$. Fale kosinusoidalna i sinusoidalna mają bowiem stałą amplitudę, a funkcje wykładnicze maleją, dążąc do zera, ale nigdy go nie osiągając. Jeśli badany przez nas system jest **stabilny**, to wartość odpowiedzi impulsowej zmniejsza się w funkcji czasu, osiągając wartość zero dla $t = +\infty$. Jest też możliwość, że system jest **niestabilny** – np. wzmacniacz samorzutnie oscylujący w wyniku zbyt silnego sprzężenia zwrotnego. W takim przypadku odpowiedź impulsowa *wzrasta* w funkcji czasu, stając się nieskończenie duża. W takim systemie nawet najmniejsze zaburzenie spowoduje nieograniczenie duży sygnał wyjściowy.

Ogólne zasady matematyczne przekształcenia Laplace'a są bardzo podobne do zasad przekształcenia Fouriera. W obu przypadkach wcześniej określone przebiegi są mnożone przez sygnał w dziedzinie czasu i uzyskany wynik jest całkowany. Na pierwszy rzut oka może się wydawać, że metodyka przekształcenia Laplace'a jest taka sama, jak Fouriera – skorelowanie sygnału w dziedzinie czasu z zestawem funkcji bazowych w celu uzyskania rozkładu przebiegu. Ale to nieprawda! Chociaż operacje matematyczne są prawie takie same, to racjonalne uzasadnienia obu metod są w znacznym stopniu różne. W przekształceniu Laplace'a następuje *badanie* przebiegu w dziedzinie czasu w celu zidentyfikowania jego głównych cech – *częstotliwości* sinusoid oraz *współczynników tłumienia* funkcji wykładniczych. Te kwestie wyjaśnimy na przykładzie.



Rys. 32.4. Przykład biegunów i zer. Filtr pasmowozaporowy (tzw. filtr typu notch) ma dwa bieguny (oznaczone x) i dwa zera (oznaczone kółkami). Na płaszczyźnie zmiennej zespolonej s pokazano pięć położeń, które w tym przykładzie „przebadamy" w celu wykonania analizy systemu. (Dalszym ciągiem tego przykładu jest rys. 32.5)



Rys. 32.5. Przebadanie odpowiedzi impulsowej. Na przekształcenie Laplace'a można spojrzeć jako na przebadanie odpowiedzi impulsowej systemu różnymi sinusoidami tłumionymi wykładniczo. Te użyte do przebadania przebiegi, które dają kasowanie, nazywamy biegunami i zerami. Na rysunku pokazano pięć przebiegów użytych do przebadania (lewa kolumna) wprowadzonych do odpowiedzi impulsowej filtru pasmowozaporowego (kolumna środkowa). Położenia na płaszczyźnie zmiennej zespolonej s odpowiadające tym pięciu przebiegom pokazano na rys. 32.4

W środkowej kolumnie na rys. 32.5 przedstawiono odpowiedź impulsową pasmowozaporowego filtru RLC (tzw. filtru wycinającego lub typu *notch*), o którym była mowa w rozdziale 30. Zawiera ona impuls dla $t = 0$, po którym następuje sinusoida tłumiona wykładniczo. Jak pokazano na rysunkach od 32.5 a) do 32.5 e), *przeprowadzamy przebadanie* tej odpowiedzi impulsowej różnymi sinusoidami tłumionymi wykładniczo. Każdą z tych sinusoid charakteryzują dwa parametry – ω, która określa częstotliwość sinusoidy, oraz σ, określająca szybkość tłumienia. A więc każdy użyty do przebadania przebieg odpowiada innemu położeniu na płaszczyźnie zmiennej zespolonej s (rys. 32.4). Przebadanie odpowiedzi impulsowej polega na *mnożeniu* jej przez przebiegi użyte do tego celu i następnie całkowaniu uzyskanego wyniku w granicach od $t = -\infty$ do $t = +\infty$. To działanie przedstawiono na rysunku w prawej kolumnie. Naszym celem jest znalezienie takich kombinacji σ i ω, które dokładnie *kasują* badaną odpowiedź impulsową. Takie skasowanie może nastąpić dwoma sposobami – powierzchnia pod krzywą może być albo *zero*, albo *prawie nieskończona*. Wszystkie inne wyniki nie są interesujące i można je pominąć. Te położenia na płaszczyźnie zmiennej zespolonej s, które dają kasowanie do zera, są nazywane **zerami** systemu. Te zaś położenia, które dają kasowanie typu „prawie nieskończoność", są nazywane **biegunami**. Bieguny i zera są analogiczne do gór i dolin w naszej opowieści o pociągu i obrazują teren „wokół" odpowiedzi impulsowej.

Na początku rozważmy, co się dzieje, gdy amplituda przebiegów użytych do przebadania zmniejsza się w funkcji czasu, jak na rys. 32.5 a). Ma to miejsce, gdy $\sigma > 0$ (prawa połówka płaszczyzny zmiennej zespolonej s). Ponieważ zarówno odpowiedź impulsowa, jak i przebieg próbkujący stają się coraz mniejsze z upływem czasu, więc taką samą właściwość będzie miał ich iloczyn. Gdy iloczyn dwóch przebiegów zostaje scałkowany w granicach od minus do plus nieskończoności, to wynikiem będzie jakaś liczba, która nie jest szczególnie interesująca. Zmniejszający się przebieg służący do przebadania *nie może*, w szczególności, skasować zmniejszającej się odpowiedzi impulsowej. Z tego wynika, że system stabilny nie będzie miał żadnych biegunów, jeśli $\sigma > 0$. Wszystkie bieguny dla systemu stabilnego są więc zlokalizowane w lewej połówce płaszczyzny s. Jeśli w prawej połówce znajdują się bieguny, to znaczy, że system jest niestabilny (tj. odpowiedź impulsowa *wzrasta* w funkcji czasu).

Na rys. 32.5 b) pokazano jeden z takich przypadków szczególnych, jakich szukamy. Gdy ten przebieg zostaje pomnożony przez odpowiedź impulsową, uzyskana całka wyniku mnożenia ma wartość zero. Wynika to z faktu, że obszar pod wykresem nad osią x (pochodzący od funkcji impulsowej delta) jest dokładnie równy obszarowi pod wykresem (od wyprostowanej sinusoidy). Wartości σ i ω dające kasowanie tego rodzaju są nazywane *zerem* systemu. Zera na wykresie płaszczyzny s na rys. 32.4 oznaczono małymi kółkami.

Na rys. 32.5 c) przedstawiono następne przebadanie. Tu stosujemy sinusoidę o amplitudzie *rosnącej* z czasem wykładniczo, lecz *wolniej* niż *zmniejsza się* amplituda odpowiedzi impulsowej w funkcji czasu. Iloczyn tych dwóch przebiegów też ma amplitudę zmniejszającą się w funkcji czasu. Podobnie jak na rys. 32.5 a), całka tego iloczynu jest jakąś nieinteresującą nas liczbą rzeczywistą. Nie zachodzi tu bowiem żaden rodzaj dokładnego kasowania.

Odstępując nieco od kolejności rysunków, spójrzcie na rys. 32.5 e). Jest to przebadanie przebiegiem o amplitudzie wzrastającej *szybciej* niż zmniejsza się amplituda odpowiedzi impulsowej. Amplituda sygnału uzyskanego w wyniku pomnożenia tych przebiegów wzrasta w funkcji czasu. To znaczy, że powierzchnia pod krzywą wzrasta w funkcji czasu i całkowita powierzchnia od $t = -\infty$ do $t = +\infty$ jest nieokreślona. W języku matematycznym mówi się, że *całka nie jest zbieżna*. Tak więc nie wszystkie powierzchnie na płaszczyźnie zmiennej zespolonej s mają wartość skończoną. Część płaszczyzny, w której całka ma określoną wartość, jest nazywana **obszarem zbieżności**. W niektórych metodach matematycznych ważna jest informacja, które części płaszczyzny zmiennej zespolonej s są w granicach obszaru zbieżności. Ta informacja nie jest wszakże konieczna do zastosowań, które omawiamy w tej książce. W naszych rozważaniach interesujące są tylko kwestie dokładnego kasowania.

Na rys. 32.5 d) amplituda przebiegu użytego do przebadania wzrasta z dokładnie z taką samą szybkością, z jaką jest tłumiona amplituda przebiegu odpowiedzi impulsowej. To powoduje, że iloczyn obu przebiegów ma stałą amplitudę. Jest to zatem linia podziału między przypadkami przedstawionymi na rys. 32.5 c), e). Powierzchnia pod wykresem ma tu wartość *prawie nieokreśloną* (niech mi matematycy wybaczą to niezbyt precyzyjne określenie). Ściślej mówiąc, ten punkt znajduje się na granicy obszaru zbieżności. Jak wspomniano, wartości σ i ω dające ten typ dokładnego kasowania są nazywane *biegunami* systemu. Bieguny zaznaczono krzyżykami (×) na płaszczyźnie zmiennej zespolonej *s*.

Analiza obwodów elektrycznych

Wprowadziliśmy przekształcenie Laplace'a, zaczynając od postaci graficznej, opisując, jak wyglądają przebiegi i w jaki sposób można nimi operować. Jest to najbardziej intuicyjny sposób zrozumienia tej metody, lecz bardzo różniący się od sposobu, w jaki metoda jest rzeczywiście stosowana. Przekształcenie Laplace'a jest w swojej istocie metodą matematyczną i jego stosowanie polega na pisaniu *równań* i operowaniu nimi. Trudność leży w tym, że łatwo się pogubić w algebrze liczb zespolonych z uwagi na jej abstrakcyjny charakter i utracić wszelkie związki ze światem rzeczywistym. Waszym zadaniem jest scalić ze sobą oba te sposoby podejścia do przekształcenia Laplace'a. Przekształcenie Laplace'a jest podstawową metodą analizy obwodów elektrycznych. Pamiętajcie, że w ten sam sposób można też przeanalizować *każdy* system, którego działanie jest opisane równaniami różniczkowymi. Obwody elektryczne są po prostu przykładem, z którego korzystamy.

Najprostszym podejściem jest rozwiązanie równań różniczkowych rządzących działaniem systemu i uzyskanie jego odpowiedzi impulsowej. Odpowiedź impulsową można następnie przekształcić, przenosząc ją do dziedziny *s* zgodnie ze wzorem 32.1. Na szczęście jest lepszy sposób – przekształcenie każdego elementu w jego równoważnik w dziedzinie *s* i następnie ocena, jak one współdziałają. Jest to bardzo podobne do przedstawionej w rozdziale 30 metody symbolicznej, gdzie rezystory, cewki indukcyjne i kondensatory są reprezentowane przez R, $j\omega L$ oraz $1/j\omega C$. W przekształceniu Laplace'a rezystory, cewki indukcyjne i kondensatory stają się zmiennymi zespolonymi: R, sL oraz $1/sc$. Zauważcie, że metoda symboliczna (zwana przekształceniem fazorowym) jest *podzbiorem* przekształcenia Laplace'a. To znaczy, że jeśli wstawi się σ równe zero do wyrażenia $s = \sigma + j\omega$, to R pozostaje jako R, sL staje się $j\omega$L, a $1/sC$ staje się $1/j\omega C$.

Podobnie jak w rozdziale 30 potraktujemy każdy z trzech elementów podstawowych jako oddzielny system, w którym przebieg prądowy jest sygnałem wejściowym, a napięciowy – wyjściowym. Gdy mówimy, że rezystory, cewki indukcyjne i kondensatory stają się równe R, sL oraz $1/sC$ w dziedzinie *s*, to odnosi się to do stosunku sygnału wyjściowego do wejściowego. Inaczej mówiąc, tym wyrażeniem jest równa transformata Laplace'a *przebiegu napięciowego* podzielona przez transformatę Laplace'a *przebiegu prądowego*.

Załóżmy, dla przykładu, że w cewce indukcyjnej wymuszamy prąd będący przebiegiem kosinusoidalnym o amplitudzie jednostkowej i częstotliwości ω_0. Powstający na cewce *przebieg napięciowy* można wyznaczyć, rozwiązując równanie różniczkowe rządzące tymi zjawiskami.

$$u(t) = L\frac{d}{dt}i(t) = L\frac{d}{dt}\cos(\omega_0 t) = \omega_0 L\sin(\omega_0 t)$$

Jeśli przebieg prądowy rozpoczyna się w chwili $t = 0$, to przebieg napięciowy też rozpocznie się w tej samej chwili, tzn. $i(t) = 0$ oraz $u(t) = 0$ dla czasów $t < 0$. Te przebiegi prądu i napięcia są przekształcane na ich reprezentacje w dziedzinie *s* według wzoru 32.1:

$$I(s) = \int_0^\infty \cos(\omega_0 t)e^{-st}dt = \frac{\omega_0}{\omega_0^2 + s^2}$$

$$U(s) = \int_0^\infty \omega_0 L\sin(\omega_0 t)e^{-st}dt = \frac{\omega_0 Ls}{\omega_0^2 + s^2}$$

Aby zakończyć ten przykład, podzielimy napięcie w dziedzinie *s* przez prąd w dziedzinie *s*, korzystając z prawa Ohma ($R = U/I$):

$$\frac{U(s)}{I(s)} = \frac{\dfrac{\omega_0 Ls}{\omega_0^2 + s^2}}{\dfrac{\omega_0}{\omega_0^2 + s^2}} = sL$$

W ten sposób stwierdziliśmy, że w dziedzinie *s* stosunek reprezentacji napięcia do reprezentacji prądu w cewce indukcyjnej jest równy sL. To jest *zawsze* prawda, niezależnie od przebiegu prądowego, od którego zaczynamy. W dziedzinie *s* ten stosunek dla rezystorów jest równy R, dla kondensatorów zaś $1/sC$.

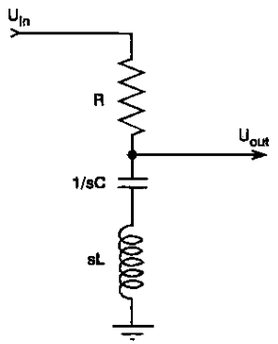
Na rys. 32.6 podano przykładowy obwód, który przeanalizujemy, stosując przekształcenie Laplace'a. Jest to filtr *RLC* pasmowozaporowy (tzw. wycinający lub typu *notch*) omawiany już w rozdziale 30. Ponieważ metodyka analizy jest jednakowa dla wszystkich obwodów elektrycznych, przedstawimy ją w formie kolejnych kroków:

Krok 1: Przekształćcie każdy z elementów obwodu w jego reprezentację w dziedzinie s. Trzeba więc zastąpić wartość każdego rezystora przez R, każdej cewki indukcyjnej przez sL, a każdego kondensatora przez $1/sC$. Przedstawiono to na rys. 32.6.

Krok 2: Wyznaczcie H(s), czyli stosunek sygnału wyjściowego do wejściowego. Jak podano w rozdziale 30, tę operację wykonuje się przez potraktowanie każdego elementu zgodnie z prawem Ohma, z „rezystancjami" określonymi przez R, sL, $1/sC$. To umożliwia skorzystanie ze standardowych równań na połączenia szeregowe i równoległe rezystorów, na dzielnik napięciowy itd. Traktując obwód *RLC* w tym przykładzie jako dzielnik napięciowy (jak w rozdziale 30), uzyskujemy H(s):

$$H(s) = \frac{U_{out}(s)}{U_{in}(s)} = \frac{sL + 1/sC}{R + sL + 1/sC} = \frac{sL + 1/sC}{R + sL + 1/sC}\left[\frac{s}{s}\right] = \frac{Ls^2 + 1/C}{Ls^2 + Rs + 1/C}$$

Jak pamiętacie z analizy *Fouriera*, stosunek widm częstotliwościowych sygnału wyjściowego i wyjściowego jest równy *odpowiedzi częstotliwościowej*, oznaczonej symbolem $H(\omega)$. Wyżej podany wzór jest rozszerzeniem tego stwierdzenia na dziedzinę *s*. Wielkość $H(s)$ jest nazywana **funkcją przenoszenia** systemu i jest równa stosunkowi reprezentacji sygnałów wyjściowego do wejściowego w dziedzinie *s*. Ponadto $H(s)$ jest równa *transformacie Laplace'a* odpowiedzi impulsowej, podobnie jak $H(\omega)$ jest równe *transformacie Fouriera* odpowiedzi impulsowej.



Rys. 32.6. Analiza filtru pasmowozaporowego w dziedzinie s. W tej procedurze pierwszym krokiem jest zastąpienie rezystora, cewki indukcyjnej i kondensatora ich równoważnikami w dziedzinie s

Jak dotąd nasze postępowanie jest identyczne, jak w poprzednim rozdziale, z tą tylko różnicą, że zamiast $j\omega$ stosujemy s. Różnica między obiema metodami staje się widoczna dopiero w dalszej procedurze, którą właśnie będziemy omawiać. Występuje teraz, gdy nie będziemy już mogli posługiwać się $j\omega$. Możemy wykreślać odpowiedź częstotliwościową lub badać ją w jakiś sposób, lecz ze względu na zasady matematyki nie możemy jej stosować w dalszych krokach analizy. Pojawią się zaś interesujące aspekty analizy Laplace'a. Wyznaczenie $H(s)$ jest kluczem do analizy Laplace'a. Jednak funkcja przenoszenia $H(s)$, aby była użyteczna, musi być wyrażona w pewnej szczególnej postaci. Wymaga to operacji algebraicznych realizowanych w dwóch następnych krokach analizy.

Krok 3: Przekształćcie $H(s)$ w taki sposób, aby było stosunkiem dwóch wielomianów. Wtedy funkcję przenoszenia można zapisać jako:

$$H(s)=\frac{as^2+bs+c}{as^2+bs+c}$$

Wzór 32.2. Funkcja przenoszenia w postaci wielomianowej

Wyrażenie funkcji przenoszenia w takiej postaci jest zawsze możliwe, *jeśli* działanie systemu jest rządzone równaniami różniczkowymi. Na przykład impuls prostokątny pokazany na rys. 32.3 nie jest rozwiązaniem równania różniczkowego i jego transformata Laplace'a nie może być zapisana w takiej postaci. Natomiast może być tak zapisana funkcja przenoszenia każdego obwodu elektrycznego złożonego z rezystorów, cewek indukcyjnych i kondensatorów. W przypadku filtru *RLC* pasmowozaporowego, analizowanego w tym przykładzie, operacje algebraiczne przedstawione w *kroku 2* od razu dały funkcję przenoszenia we właściwej postaci, to znaczy:

$$H(s)=\frac{as^2+bs+c}{as^2+bs+c}=\frac{Ls^2+1/C}{Ls^2+Rs+1/C}$$

gdzie $a=L$, $b=0$, $c=1/C$;

oraz $a=L$, $b=R$, $c=1/C$

Krok 4: Rozłóż na czynniki wielomiany w liczniku i mianowniku (jest to tzw. faktoryzacja). To znaczy rozbijcie wielomiany w liczniku i mianowniku na czynniki, z których każdy zawiera pojedyncze s. Wszystkie te czynniki pomnożone przez siebie muszą dawać wielomian z licznika lub mianownika. Równanie przybiera więc postać:

$$H(s)=\frac{(s-z_1)(s-z_2)(s-z_3)...}{(s-p_1)(s-p_2)(s-p_3)...}$$

Wzór 32.3. Funkcja przenoszenia rozłożona na czynniki w dziedzinie s. Wzór w takiej postaci może być wyrażony przez zera i bieguny

Pierwiastki licznika, z_1, z_2, z_3 ... są **zerami** równania, a pierwiastki mianownika – **biegunami**. Są to te same zera i bieguny, z którymi spotkaliście się już wcześniej w tym rozdziale. Ich zastosowanie omówimy w dalszej części rozdziału.

Rozkładanie na czynniki w dziedzinie s jest proste, jeśli licznik i mianownik są *wielomianami drugiego rzędu* (lub niższego).

Inaczej mówiąc, łatwo możemy operować wyrazami s i s^2, lecz nie s^3, s^4, s^5, ... Wynika to z łatwości wyznaczania pierwiastków wielomianu drugiego rzędu $ax^2 + bx + c$ z dobrze znanego wzoru na pierwiastki równania kwadratowego: $x=-b\pm\sqrt{b^2-4ac}/2a$. Stosując tę metodę, można funkcję przenoszenia analizowanego filtru rozłożyć na czynniki w sposób następujący:

$$H(s)=\frac{(s-z_1)(s-z_2)}{(s-p_1)(s-p_2)}$$

gdzie:

$$z_1=\frac{j}{\sqrt{LC}} \qquad p_1=\frac{-R+\sqrt{R^2-4L/C}}{2L}$$

$$z_2=-\frac{j}{\sqrt{LC}} \qquad p_2=\frac{-R-\sqrt{R^2-4L/C}}{2L}$$

System drugiego rzędu, taki jak w analizowanym przykładzie, ma maksimum dwa zera i dwa bieguny. Liczba biegunów w systemie jest równa liczbie niezależnych elementów gromadzących energię. Na przykład kondensatory i cewki indukcyjne gromadzą energię, rezystory zaś – nie. Liczba zer jest mniejsza lub równa liczbie biegunów.

Wielomianów rzędu wyższego niż drugi nie można, ogólnie mówiąc, rozkładać na czynniki metodami algebraicznymi; trzeba stosować bardziej skomplikowane metody numeryczne. Jako alternatywę można projektować układy będące *kaskadami sekcji drugiego rzędu*. Dobrym tego przykładem jest rodzina filtrów analogowych omówiona w rozdziale 3. Filtr ośmiobiegunowy, na przykład, projektuje się, łącząc kaskadowo cztery sekcje, z których każdy ma dwa bieguny. Warto podkreślić, że takie układy wielosekcyjne stosuje się w celu przezwyciężenia ograniczeń w *matematyce*, a nie w *elektronice*.

Znaczenie zer i biegunów

Chcąc uczynić nasze rozważania mniej abstrakcyjnymi, użyjemy rzeczywistych wartości elementów w filtrze pasmowozaporowym, który właśnie analizowaliśmy: $R = 220\ \Omega$, $L = 54\ \mu H$, $C = 470\ pF$. Wstawiając te wartości do wzorów z poprzedniego podrozdziału, uzyskujemy położenia zer i biegunów:

$$z_1=0+j6{,}277\times10^6 \qquad p_1=-2{,}037\times10^6+j5{,}937\times10^6$$

$$z_2=0-j6{,}277\times10^6 \qquad p_1=-2{,}037\times10^6-j5{,}937\times10^6$$

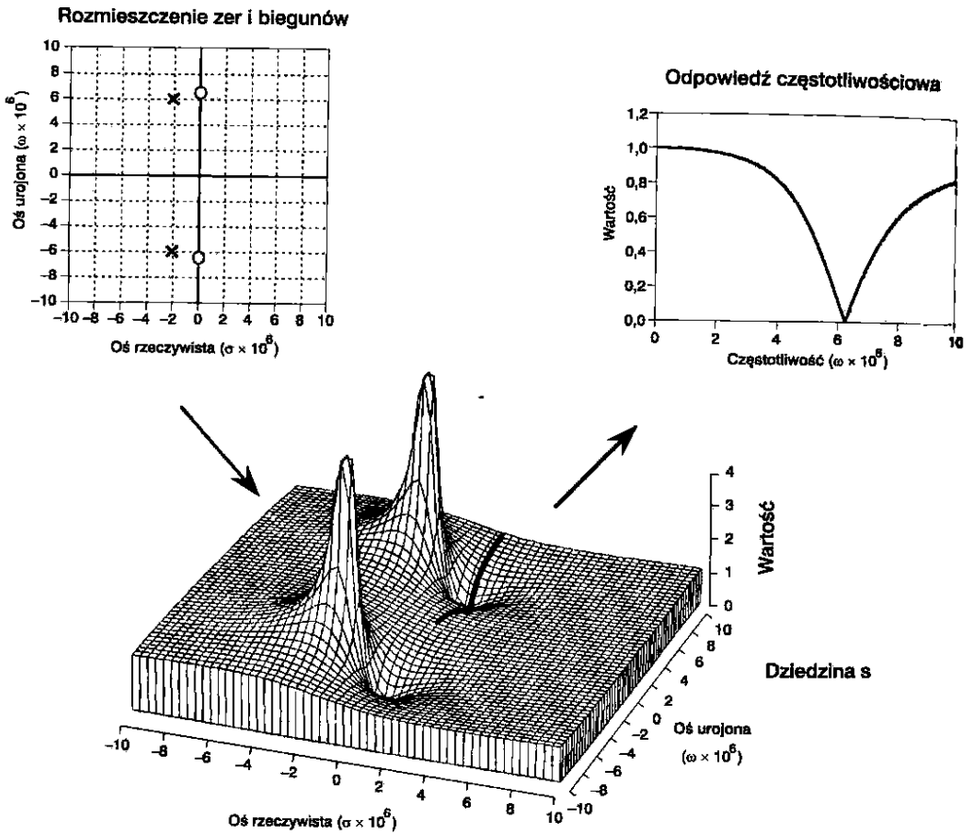
Takie rozmieszczenie zer i biegunów przedstawiono na rys. 32.7. Każde zero jest zaznaczone kółkiem, a każdy biegun krzyżykiem. Jest to wykres zer i biegunów będący najczęstszą formą prezentacji płaszczyzny zmiennej zespolonej s. Na rys. 32.7 pokazano również obraz topograficzny płaszczyzny zmiennej zespolonej s. Dla uproszczenia przedstawiono tylko moduł, ale nie zapominajcie, że jest też odpowiadająca mu faza. Tak jak góry i doliny określają kształt powierzchni ziemi, bieguny i zera określają kształt przebiegu na płaszczyźnie zmiennej zespolonej s. W odróżnieniu od gór i dolin, każdy biegun i zero mają dokładnie taki sam kształt i rozmiar, jak wszystkie inne bieguny i zera. Jedyną cechą charakterystyczną dla każdego bieguna lub zera jest jego *położenie*. Bieguny i zera są ważne, gdyż dają zwięzłą reprezentację wartości *każdego punktu na płaszczyźnie zmiennej zespolonej s*. To znaczy, że możemy w pełni opisać właściwości systemu przy użyciu tylko *niewielu parametrów*. W przypadku pasmowozaporowego filtru RLC wystarczy określić tylko cztery parametry zespolone charakteryzujące system: z_1, z_2, p_1, p_2 (każdy z nich składa się z części rzeczywistej i urojonej).

Dla lepszego zrozumienia biegunów i zer wyobraźmy sobie mrówkę poruszającą się po płaszczyźnie zmiennej zespolonej s. W każdym położeniu na płaszczyźnie, w którym znajdzie się mrówka (tj. dla jakiejś wartości s), istnieje odpowiadająca mu wartość funkcji przenoszenia $H(s)$. Ta wartość jest pewną liczbą zespoloną, którą można wyrazić jako moduł i fazę lub jako część rzeczywistą i urojoną. Teraz pozwólmy mrówce, żeby zaprowadziła nas do jednego z zer na płaszczyźnie zmiennej zespolonej s. W tym położeniu zmierzone

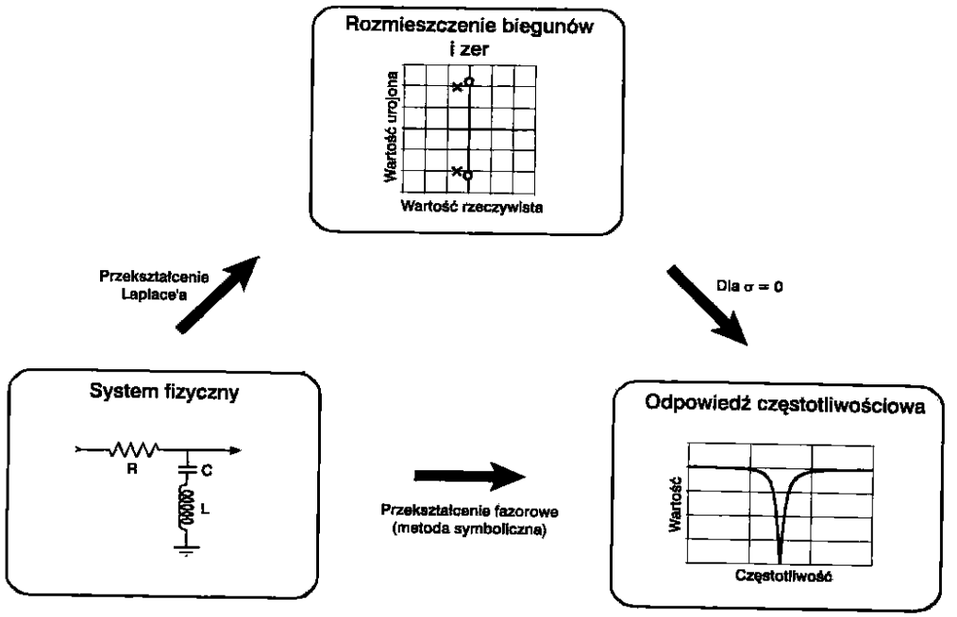
przez nas wartości części rzeczywistej i urojonej będą *zero*. Można to zrozumieć, rozpatrując wzór 32.3 wyrażający $H(s)$. Jeśli położenie s jest równe któremukolwiek z zer, to jeden z wyrazów w liczniku będzie równy zeru. To czyni całe wyrażenie równym zeru, niezależnie od pozostałych wartości.

Następnie wędrówka naszej mrówki prowadzi nas do jednego z biegunów, gdzie znowu mierzymy wartości części rzeczywistej i urojonej $H(s)$. Mierzona wartość staje się coraz to większa i większa w miarę, jak coraz bardziej zbliżamy się do dokładnego położenia bieguna (stąd też jego nazwa). To też można zrozumieć na podstawie wzoru 32.3. Jeśli położenie s jest równe jednemu z p w mianowniku, to mianownik będzie równy zeru, a dzielenie przez zero powoduje, że całe wyrażenie staje się nieskończenie duże.

Zbadawszy te położenia szczególne, nasza mrówka wędruje teraz w sposób przypadkowy po płaszczyźnie zmiennej zespolonej s. Wartość $H(s)$ w każdym położeniu zależy całkowicie od rozmieszczenia zer i biegunów, ponieważ na tym dziwnym terenie nie są dozwolone *żadne* cechy innego rodzaju. Jeśli jesteśmy blisko bieguna, wartość jest duża, a w pobliżu zera – mała.



Rys. 32.7. Bieguny i zera w dziedzinie s. Na rysunkach przedstawiono zależność między rozmieszczeniem zer i biegunów, wykresem na płaszczyźnie zmiennej zespolonej s oraz odpowiedzią częstotliwościową. Wykresy dotyczą filtru pasmowozaporowego o wartościach elementów: R = 220 Ω, L = 54 μH, C = 470 pF. Takie wartości dają pulsację środkową pasma zaporowego filtru ω = 6,277 · 10^6, co odpowiada częstotliwości ok. 1 MHz



Rys. 32.8. Metodyka stosowania przekształcenia Laplace'a. Metoda symboliczna (przekształcenie fazorowe) przedstawiona w rozdziale 30 (czyli metoda, w której stosuje się R, jωL i –j/ωC) umożliwia bezpośrednie obliczenie odpowiedzi częstotliwościowej na podstawie parametrów systemu fizycznego. Natomiast stosując przekształcenie Laplace'a, oblicza się reprezentację systemu fizycznego w dziedzinie s, przedstawianą na ogół w formie rozmieszczenia zer i biegunów. Następnie można określić odpowiedź częstotliwościową, wyznaczając wartości funkcji przenoszenia wzdłuż osi urojonej na płaszczyźnie liczb zespolonych s. Chociaż końcowy wynik jest w obu metodach taki sam, to pośredni krok z rozmieszczeniem zer i biegunów daje wyjaśnienie przebiegu odpowiedzi częstotliwościowej

Wzór 32.3 opisuje również, jak *wielokrotne* bieguny i zera współdziałają w tworzeniu sygnału w dziedzinie s. Jak pamiętacie, odejmując dwie liczby zespolone, uzyskuje się *odległość* między nimi na płaszczyźnie zmiennej zespolonej. Na przykład $(s - z_0)$ jest odległością między dowolnym położeniem s i zerem znajdującym się w położeniu z_0. A zatem wzór 32.3 określa, że wartość $H(s)$ w każdym położeniu s jest równa *iloczynowi* odległości tego położenia od wszystkich zer, podzielonemu przez *iloczyn* odległości od wszystkich biegunów.

To nas doprowadza do sedna tematyki tego rozdziału, a więc do kwestii, w jaki sposób rozmieszczenie zer i biegunów umożliwia głębsze zrozumienie *odpowiedzi częstotliwościowej* systemu. Odpowiedź częstotliwościową tworzą wartości funkcji przenoszenia $H(S)$ znajdujące się wzdłuż osi urojonej, oznaczone ciemną linią na obrazie topograficznym na rys. 32.7. Wyobraźmy sobie naszą mrówkę wyruszającą z początku układu współrzędnych i posuwającą się wzdłuż osi liczb urojonych. W pobliżu początku układu współrzędnych odległość do zer jest w przybliżeniu równa odległości do biegunów. Z tego powodu licznik i mianownik we wzorze 32.3 są sobie równe, dając odpowiedź częstotliwościową równą 1 przy małych częstotliwościach. Taka sytuacja nie zmienia się w sposób znaczący, dopóki mrówka nie dotrze w okolice zera i bieguna. Przy zbliżaniu się do położenia zera wartość $H(s)$ spada gwałtownie, uzyskując wartość *zero*, gdy mrówka jest w położeniu *zera*. Gdy mrówka minie już parę biegun–zero, wartość $H(s)$ znowu wraca do jedności. Stosując taki rodzaj obrazowania, można zauważyć, że szerokość wcięcia (pasma zaporowego) na charakterystyce zależy od odległości między zerem a biegunem.

Na rys. 32.8 podsumowano metodykę stosowania przekształcenia Laplace'a. Zaczynamy od systemu fizycznego, takiego jak obwód elektryczny. Jeśli chcemy, to metodą symboliczną możemy bezpośrednio uzyskać odpowiedź częstotliwościową obwodu, zgodnie z opisem w rozdziale 30. Alternatywą jest wykonanie przekształcenia Laplace'a w czterech krokach metodą poprzednio omówioną. W wyniku uzyskuje się wyrażenie matematyczne na funkcję przenoszenia H(*s*), które można przedstawić w formie wykresu zer i biegunów. Następnie można znaleźć odpowiedź częstotliwościową, wyznaczając przebieg funkcji przenoszenia wzdłuż osi liczb urojonych, a więc zastępując każde *s* przez *jω*. Aczkolwiek obie metody dają jednakowy wynik, to pośredni wykres zer i biegunów umożliwia zrozumienie, *dlaczego* system zachowuje się tak, a nie inaczej oraz jak to można zmienić.

Projektowanie filtrów w dziedzinie zmiennej zespolonej *s*

Najważniejszym zastosowaniem przekształcenia Laplace'a jest projektowanie systemów *bezpośrednio* w dziedzinie *s*. Takie projektowanie obejmuje dwa kroki. Po pierwsze, projektuje się system w dziedzinie *s*, określając liczbę i położenia biegunów i zer. Jest to problem czysto matematyczny, a celem jest uzyskanie najlepszej odpowiedzi częstotliwościowej. W drugim kroku określa się układ elektroniczny mający taką reprezentacje na płaszczyźnie zmiennej zespolonej *s*. Ta metoda ma w sobie coś ze sztuki, gdyż jest wiele konfiguracji układowych charakteryzujących się danym rozmieszczeniem zer i biegunów.

Jak już poprzednio wspomniano, *krok 4* w metodzie przekształcenia Laplace'a jest bardzo trudny, jeśli system ma więcej niż dwa bieguny lub dwa zera. Najczęściej stosowanym rozwiązaniem jest realizacja *wielokrotnych* biegunów i zer w kilku *kolejnych* sekcjach. Na przykład filtr 6-biegunowy można zrealizować jako trzy kolejne sekcje, z których każdy ma nie więcej niż dwa bieguny i dwa zera. Ponieważ każda z tych sekcji może być wyrażona w dziedzinie *s* jako stosunek wielomianów drugiego stopnia w liczniku i mianowniku, więc ta metoda jest nazywana projektowaniem z sekcjami **bikwadratowymi**.

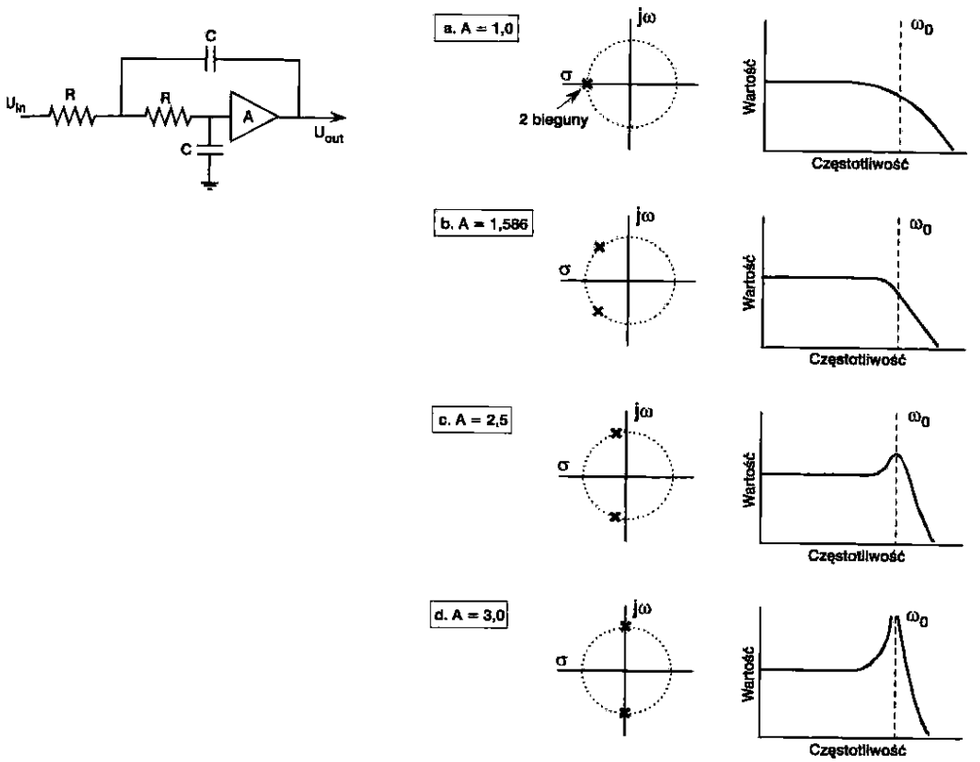
Na rys. 32.9 pokazano typowy układ bikwadratowy stosowany w metodzie projektowania opisanej w rozdziale 3. Nosi on nazwę układu **Sallena-Keya** od nazwisk R.P. Sallena i E.L. Keya, autorów artykułu, w którym opisali tę metodę w latach 50. XX wieku. Istnieje wprawdzie kilka odmian układu, ale w najpopularniejszej stosuje się dwa rezystory o jednakowych wartościach, dwa kondensatory o jednakowych wartościach i wzmacniacz o wzmocnieniu od 1 do 3. Teraz można zastosować niedrogi wzmacniacz operacyjny z odpowiednimi rezystorami w sprzężeniu zwrotnym. W czasach Sallena i Keya takie wzmacniacze oczywiście nie były jeszcze dostępne. Wykonując czterokrokową procedurę analizy układu, uzyskuje się zależności położenia dwóch biegunów od wartości elementów układu:

$$\sigma = \frac{A-3}{2RC}$$

$$\omega = \frac{\pm\sqrt{-A^2+6A-5}}{2RC}$$

Wzór 32.4. Położenia biegunów w układzie Sallena-Keya. Te wzory opisują zależność położenia biegunów ω i σ od wzmocnienia wzmacniacza A, rezystora R i pojemności kondensatora C

Z powyższych wzorów wynika, że dwa bieguny zawsze leżą gdzieś na okręgu o promieniu 1/RC. Dokładna lokalizacja na okręgu zależy od wzmocnienia wzmacniacza. Jak pokazano na rys. 32.9 a), przy wzmocnieniu 1 oba bieguny znajdują się na osi rzeczywistej. Odpowiedź częstotliwościowa takiej konfiguracji jest charakterystyką filtru dolnoprzepustowego o dość łagodnym przejściu między pasmem przepustowym a zaporowym. Częstotliwość odcięcia –3 dB (0,707) tego układu, oznaczona ω_0, jest w tym miejscu, gdzie okrąg przecina się z osią urojoną, tj. $\omega_0 = 1/RC$.



Rys. 32.9. Charakterystyki układu Sallena-Keya. Układ daje dwa bieguny leżące na okręgu o promieniu 1/RC. Gdy wzmocnienie wzmacniacza jest zwiększane, bieguny oddalają się od osi rzeczywistej (jak na rys.a) ku osi urojonej (jak na rys.d)

Gdy wzmocnienie wzmacniacza jest zwiększane, bieguny przesuwają się po okręgu, co powoduje także odpowiednią zmianę odpowiedzi częstotliwościowej. Przy wzmocnieniu 1,586 oba bieguny są rozmieszczone pod kątem 45° (rys. 32.9 b), co powoduje bardziej strome przejście na charakterystyce. Zwiększanie wzmocnienia przesuwa bieguny jeszcze bliżej osi urojonej, powodując powstanie przerzutu na charakterystyce. Tę sytuację przedstawiono na rys. 32.9 c przy wzmocnieniu ustalonym na 2,5. Amplituda przerzutu nadal wzrasta w miarę wzrostu wzmocnienia, aż wzmocnienie osiągnie wartość 3. Przy tym wzmocnieniu zachodzi przypadek szczególny (rys. 32.9 d), gdy bieguny są zlokalizowane bezpośrednio na osi urojonej. Odpowiedź częstotliwościowa ma wówczas wartość nieskończenie dużą w punkcie przerzutu. W praktyce oznacza to, że układ stał się generatorem. Dalsze zwiększanie wzmocnienia spycha bieguny głębiej w prawą połówkę płaszczyzny zmiennej zespolonej *s*. Jak wspomniano poprzednio, odpowiada to niestabilności systemu (generacji samorzutnej).

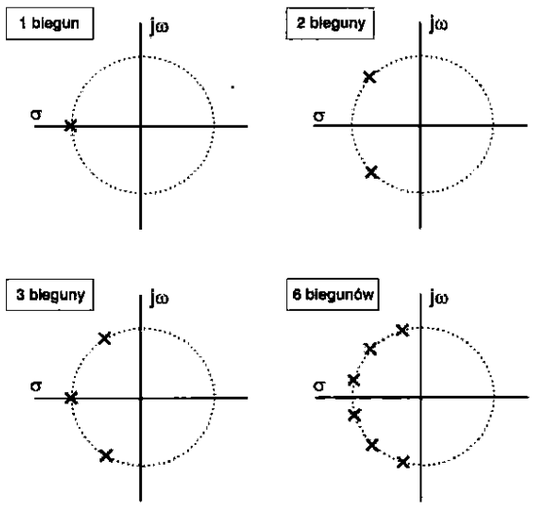
Można zrealizować wiele filtrów różnych rodzajów, stosując układ Sallena-Keya jako człon podstawowy. Na przykład dolnoprzepustowy **filtr Butterwortha** projektuje się, rozmieszczając wybraną liczbę biegunów symetrycznie wokół lewej połówki okręgu (rys. 32.10). Każde dwa bieguny w tej konfiguracji wymagają jednego stopnia Sallena-Keya. Filtr Butterwortha ma charakterystykę maksymalnie płaską (patrz rozdział 3), tzn. ma najostrzejsze przejście między pasmem przepustowym a zaporowym *bez przerzutów* na odpowiedzi częstotliwościowej. Im więcej biegunów zastosujemy, tym bardziej strome przejście uzyskamy. Ponieważ

wszystkie bieguny w filtrze Butterwortha leżą na tym samym okręgu, więc we wszystkich połączonych kaskadowo stopniach stosuje się takie same wartości R i C. Jedyną różnicą między stopniami jest wzmocnienie. Dlaczego takie kołowe rozmieszczenie biegunów zapewnia odpowiedź optymalnie płaską? Nie szukajcie żadnej oczywistej ani intuicyjnej odpowiedzi na to pytanie; to po prostu wynika z analizy matematycznej.

Na rys. 32.11 pokazano, jak można zmodyfikować położenia biegunów w taki sposób, aby utworzyć **filtr Czebyszewa**. Jak wynika z rozważań w rozdziale 3, w filtrze Czebyszewa osiąga się ostrzejsze przejście między pasmami kosztem zwiększenia zafalowań w paśmie przepustowym. Na płaszczyźnie zmiennej zespolonej s odpowiada to spłaszczeniu *w elipsę* okręgu, na którym są bieguny. Im bardziej spłaszczona elipsa, tym więcej jest zafalowań w paśmie przepustowym i tym ostrzejsze jest przejście. Taki filtr, tworzony jako kaskada stopni Sallena-Keya, wymaga w każdym stopniu różnych wartości rezystorów i kondensatorów.

Na rys. 32.11 przedstawiono też następny poziom zaawansowania w metodyce projektowania filtrów – **filtr eliptyczny**. W tym filtrze uzyskuje się najostrzejsze z możliwych przejście między pasmami z jednoczesnym dopuszczeniem do zafalowań w paśmie tak przepustowym, jak i zaporowym. Na płaszczyźnie zmiennej zespolonej s odpowiada to rozmieszczeniu zer na osi urojonej, przy czym pierwsze znajduje się blisko częstotliwości odcięcia. Filtry eliptyczne mają kilka odmian i są w projektowaniu znacznie trudniejsze niż konfiguracje Butterwortha czy Czebyszewa. Przyczyną jest fakt, że rozmieszczenie biegunów i zer filtru eliptycznego nie jest prostą figurą geometryczną, lecz jest określone matematycznie z udziałem funkcji eliptycznych i całek (stąd nazwa filtru).

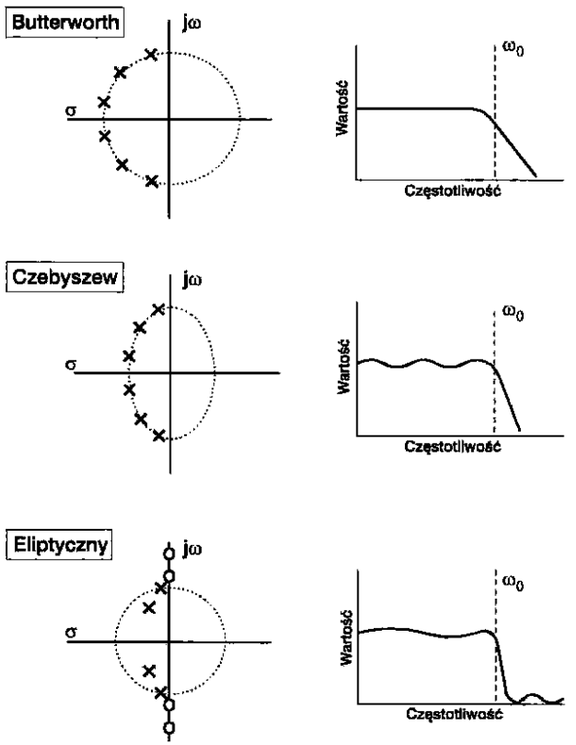
Ponieważ każdy stopień bikwadratowy wytwarza dwa bieguny, więc **filtry rzędu parzystego** (liczba biegunów 2, 4, 6 itd.) mogą być tworzone przez kaskadowe łączenie stopni bikwadratowych. Jednak **filtry rzędu nieparzystego** (liczba biegunów 1, 3, 5 itd.) wymagają tego, czego stopnie bikwadratowe nie mogą zapewnić – pojedynczego bieguna na osi rzeczywistej. Jest to nic innego, jak prosty obwód RC dodany do kaskady. Na przykład filtr 9-biegunowy



Rys. 32.10. Rozmieszczenie biegunów na płaszczyźnie zmiennej zespolonej s dla filtru Butterwortha. Dolnoprzepustowy filtr Butterwortha jest realizowany przez rozmieszczenie biegunów równomiernie wokół lewej połówki okręgu. Im więcej zastosowano biegunów w charakterystyce filtru, tym szybsze jest opadanie charakterystyki

może być zrealizowany z 5 stopni: 4 stopni bikwadratowych Sallena-Keya oraz jednego stopnia złożonego z kondensatora i rezystora.

Te klasyczne rozmieszczenia biegunów i zer dotyczą filtrów dolnoprzepustowych, lecz mogą być modyfikowane dla innych odpowiedzi częstotliwościowych. Modyfikacje wykonuje się, projektując filtr dolnoprzepustowy i przeprowadzając następnie transformację matematyczną w dziedzinie s. Zaczynamy od obliczenia położeń filtru dolnoprzepustowego i następnie napisania funkcji przenoszenia $H(s)$ w postaci takiej, jak we wzorze 32.3. Funkcję przenoszenia równoważnego filtru górnoprzepustowego uzyskuje się, zamieniając każde „s" przez „$1/s$" i tak przekształcając uzyskane wyrażenie, aby znowu miało postać zero-biegunową, jak we wzorze 32.3. Wtedy mamy nowe rozmieszczenie biegunów i zer, będące implementacją filtru górnoprzepustowego. Przez bardziej skomplikowane przekształcenia w dziedzinie s można realizować filtry pasmowoprzepustowe i pasmowozaporowe, wychodząc z początkowego projektu filtru dolnoprzepustowego. Ten rodzaj operacji matematycznych w dziedzinie s jest głównym zagadnieniem projektowania filtrów, któremu są poświęcone całe książki. Projektowanie filtrów analogowych to 90% *matematyki* i tylko 10% *elektroniki*.

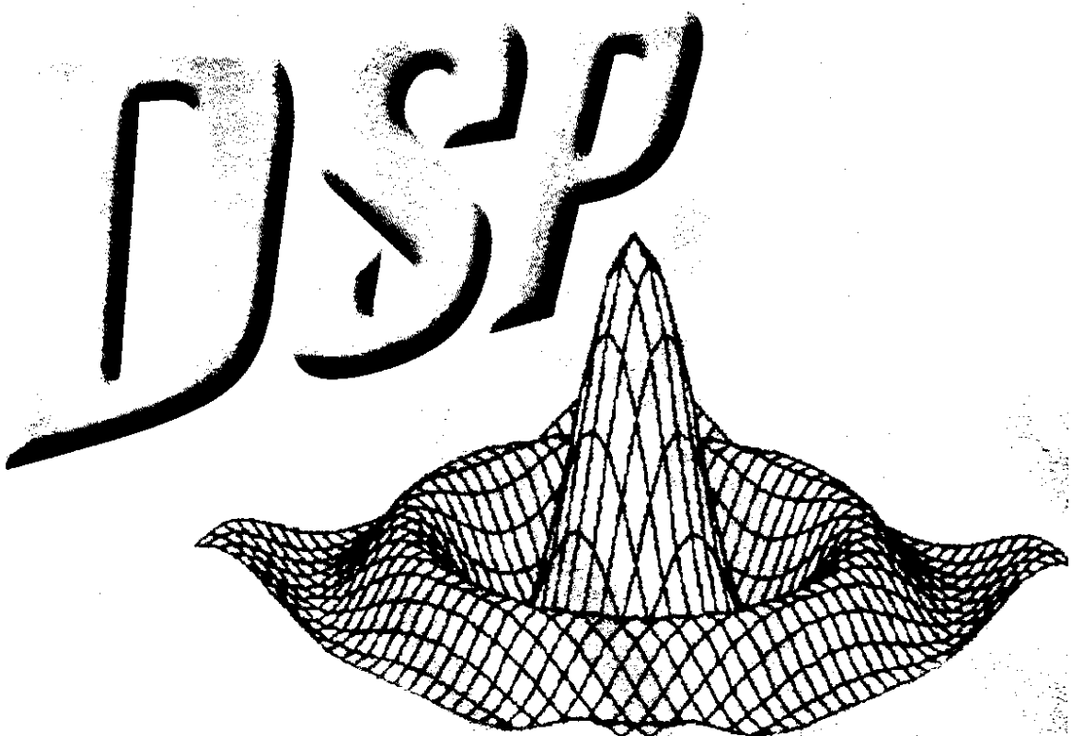


Rys. 32.11. Klasyczne rozmieszczenia zer i biegunów. Są to trzy klasyczne rozmieszczenia występujące w projektowaniu filtrów. Filtry Butterwortha mają bieguny rozmieszczone równomiernie na okręgu, co daje odpowiedź maksymalnie płaską. W filtrach Czebyszewa bieguny są rozmieszczone na elipsie, co daje ostrzejsze przejście na charakterystyce uzyskane kosztem występowania zafalowań w paśmie przepustowym. Filtry eliptyczne mają dodane zera w paśmie przepustowym. Uzyskuje się w nich charakterystykę z szybszym przejściem, lecz z zafalowaniami w paśmie przepustowym, a także w zaporowym

Na szczęście, projektowanie filtrów górnoprzepustowych z wykorzystaniem stopni Sallena-Keya nie wymaga takich operacji matematycznych. Zastąpienie „s przez „$1/s$" w dziedzinie s odpowiada zamianie miejscami kondensatorów i rezystorów w układzie. Na płaszczyźnie zmiennej zespolonej s ta zamiana powoduje umieszczenie biegunów w nowym położeniu i dodanie dwóch zer bezpośrednio w początku układu współrzędnych. To daje w wyniku odpowiedź częstotliwościową o wartości zero dla przebiegu stałego (DC, częstotliwość zero), czego zresztą można się było spodziewać po filtrze górnoprzepustowym. W ten sposób układ Sallena-Keya osiąga swoje pełne możliwości – implementację dwóch biegunów *oraz* dwóch zer.

33

Przekształcenie Z



Tak jak filtry analogowe projektuje się, stosując przekształcenie Laplace'a, to rekursywne filtry cyfrowe są opracowywane przy użyciu metody będącej odpowiednikiem przekształcenia Laplace'a, a zwanej przekształceniem Z. Ogólna metodyka obu tych przekształceń jest taka sama – wykonuje się przebadanie odpowiedzi impulsowej sinusoidami i przebiegami wykładniczymi w celu znalezienia zer i biegunów systemu. W przekształceniu Laplace'a posługujemy się równaniami różniczkowymi, dziedziną s i płaszczyzną zmiennej zespolonej s. W przekształceniu Z zaś posługujemy się, odpowiednio, równaniami różnicowymi, dziedziną z i płaszczyzną zmiennej z. Obie metody nie są jednak wzajemnym odbiciem lustrzanym; płaszczyzna zmiennej zespolonej s jest ujęta w prostokątny układ współrzędnych, podczas gdy na płaszczyźnie zmiennej z stosuje się współrzędne biegunowe. Rekursywne filtry cyfrowe często się projektuje, rozpoczynając od jednego z klasycznych filtrów analogowych, takiego jak filtr Butterwortha, Czebyszewa lub eliptyczny. Następnie przeprowadza się szereg przekształceń matematycznych w celu uzyskania żądanego filtru cyfrowego. Przekształcenie Z tworzy ogólną strukturę tych operacji matematycznych. Takie podejście zastosowano w programie do projektowania filtru Czebyszewa przedstawionym w rozdziale 20. W tym rozdziale ta metoda będzie omówiona szczegółowo.

Właściwości dziedziny z

W celu uzasadnienia stwierdzenia, że przekształcenia Z i Laplace'a są odpowiednikami, wykażemy, w jaki sposób można przekształcenia Laplace'a przemienić w przekształcenie Z. Według informacji podanych w poprzednim rozdziale przekształcenie Laplace'a definiuje się zależnością między sygnałami w dziedzinie czasu i w dziedzinie s:

$$X(s)=\int_{t=-\infty}^{\infty} x(t)e^{-st}dt$$

gdzie $x(t)$ i $X(s)$ są – odpowiednio – reprezentacjami sygnału w dziedzinach czasu i s. Jak powiedziano w poprzednim rozdziale, ten wzór daje analizę sygnału w dziedzinie czasu jako sumy fal sinusoidalnych i kosinusoidalnych o amplitudach zmieniających się w sposób wykładniczy. Można to zrozumieć, zamieniając zmienną zespoloną s przez równoważne wyrażenie $\sigma + j\omega$. Przy tym alternatywnym zapisie przekształcenie Laplace'a przyjmuje postać:

$$X(\sigma,\omega)=\int_{t=-\infty}^{\infty} x(t)e^{-\sigma t}e^{-j\omega t}dt$$

Jeśli interesują nas tylko *rzeczywiste* sygnały w dziedzinie czasu (najczęstszy przypadek), to górna i dolna połówka płaszczyzny zmiennej zespolonej s są wzajemnie odbiciami lustrzanymi i wyraz $e^{-j\omega t}$ redukuje się do prostych przebiegów sinusoidalnych i kosinusoidalnych. Ten wzór określa każde *położenie* na płaszczyźnie zmiennej zespolonej s dwoma parametrami σ i ω. *Wartość* każdego położenia jest liczbą zespoloną składającą się z części rzeczywistej i urojonej. W celu wyznaczenia części rzeczywistej mnoży się sygnał w dziedzinie czasu przez falę kosinusoidalną o pulsacji ω i o amplitudzie zmieniającej się wykładniczo zgodnie ze współczynnikiem tłumienia σ. Wartość części rzeczywistej $X(\sigma,\omega)$ jest równa całce uzyskanego przez mnożenie przebiegu. W podobny sposób uzyskuje się wartość części urojonej, stosując jednak falę sinusoidalną. Jeśli to wszystko wydaje się wam mało znane, to przed przejściem do dalszej części materiału powinniście przejrzeć poprzedni rozdział.

Przekształcenie Laplace'a można zamienić w przekształcenie Z w trzech krokach. Pierwszy krok jest najbardziej oczywisty – zamiana sygnału ciągłego na dyskretny. Wykonuje się to, zastępując zmienną czasową t numerem próbki n i zamieniając całkę na sumowanie:

$$X(\sigma,\omega)=\sum_{n=-\infty}^{\infty} x[n]e^{-\sigma n}e^{-j\omega n}$$

Zauważcie, że w wyrażeniu $X(\sigma, \omega)$ zastosowano nawiasy okrągłe, co wskazuje, że sygnał jest ciągły, a nie dyskretny. Chociaż operujemy już sygnałem dyskretnym w dziedzinie czasu

$x[n]$, to parametry σ i ω nadal mogą przybierać ciągły zakres wartości. Drugim krokiem jest zapisanie wyrazu wykładniczego w innej formie. Sygnał wykładniczy może być wyrażony matematycznie jednym z dwóch sposobów:

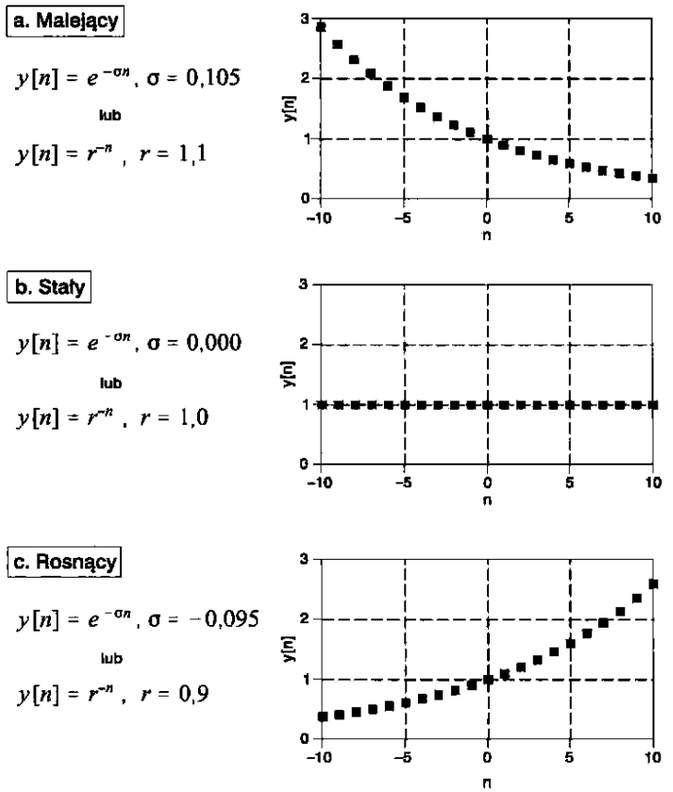
$$y[n]=e^{-\sigma n} \text{ lub } y[n]=r^{-n}$$

Oba powyższe wzory dają krzywą wykładniczą (rys. 33.1). W pierwszym wyrażeniu szybkość opadania sygnału jest określona współczynnikiem tłumienia σ. Jeśli ten współczynnik σ jest dodatni, to wartość przebiegu *się zmniejsza* w miarę wzrostu numeru próbki n. Krzywa jest zaś *rosnąca*, gdy σ jest ujemny. Jeśli σ jest równy dokładnie zero, to sygnał będzie miał stałą wartość równą *jeden*.

W drugim wyrażeniu zastosowano parametr r określający opadanie przebiegu. Przebieg maleje, jeśli $r > 1$, a wzrasta, gdy $r < 1$. Sygnał ma wartość stałą, gdy $r = 1$. Te dwa wzory są po prostu dwoma sposobami wyrażania tej samej rzeczy. Można przechodzić z jednego sposobu do drugiego, korzystając z zależności:

$$r^{-n}=\left[e^{\ln(r)}\right]^{-n}=e^{-n\ln(r)}=e^{-\sigma n}$$

$$\text{gdzie: } \sigma=\ln(r)$$



Rys. 33.1. Sygnały wykładnicze. Przebiegi wykładnicze można wyrażać matematycznie w dwóch różnych postaciach. W przekształceniu Laplace'a stosuje się jedną postać, a w przekształceniu Z – drugą

Drugi krok konwersji przekształcenia Laplace'a w przekształcenie Z jest wykonywany z zastosowaniem *drugiej* postaci przebiegu wykładniczego:

$$X(r,\omega)=\sum_{n=-\infty}^{\infty} x[n]r^{-n}e^{-j\omega n}$$

Jest to wprawdzie idealnie prawidłowe wyrażenie przekształcenia Z, ale nie jest to postać najbardziej dogodna do zapisu zespolonego. W przekształceniu Laplace'a przezwyciężono tę trudność, wprowadzając nową zmienną zespoloną $s = \sigma + j\omega$. W ten sam sposób zdefiniujemy nową zmienną dla przekształcenia Z:

$$z = re^{j\omega}$$

Jest to zdefiniowanie zmiennej zespolonej z będącej zapisem biegunowym kombinacji dwóch zmiennych rzeczywistych r i ω. Trzecim krokiem w wyprowadzeniu przekształcenia Z jest zastąpienie r i ω przez z. W ten sposób dochodzimy do standardowej postaci przekształcenia Z:

$$X(z)=\sum_{n=-\infty}^{\infty} x[n]z^{-n}$$

Wzór 33.1. Przekształcenie Z. Określa ono zależność między sygnałem x[n] w dziedzinie czasu i sygnałem X(z) w dziedzinie z

Dlaczego w przekształceniu Z stosuje się r^n zamiast $e^{-\sigma n}$ oraz z zamiast s? Jak podano w rozdziale 19, filtry rekursywne są implementowane przez zestaw *współczynników rekursywnych*. W celu analizy tych systemów w dziedzinie z trzeba mieć możliwość konwersji tych współczynników rekursywnych na *funkcję przenoszenia* w dziedzinie z i na odwrót. Jak to pokrótce wykażemy, zdefiniowanie przekształcenia Z w ten sposób (r^n i z) daje najprostszy sposób poruszania się między tymi dwoma ważnymi sposobami reprezentacji systemu. Zdefiniowanie w ten sposób przekształcenia Z powoduje w istocie, że przechodzenie od jednego do drugiego sposobu reprezentacji staje się *trywialne*.

Na rys. 33.2 zilustrowano różnicę między płaszczyznami zmiennej zespolonej s przekształcenia Laplace'a i płaszczyzną zmiennej zespolonej z w przekształceniu Z. Położenia na płaszczyźnie zmiennej zespolonej s są określane dwoma parametrami – współczynnikiem tłumienia wzdłuż osi poziomej σ i pulsacją ω wzdłuż osi pionowej. Innymi słowy, te dwa rzeczywiste parametry są umieszczone w układzie współrzędnych *prostokątnych*. Taka geometria wynika z definicji s, zmiennej zespolonej reprezentującej położenie na płaszczyźnie zmiennej zespolonej s przez zależność $s = \sigma + j\omega$.

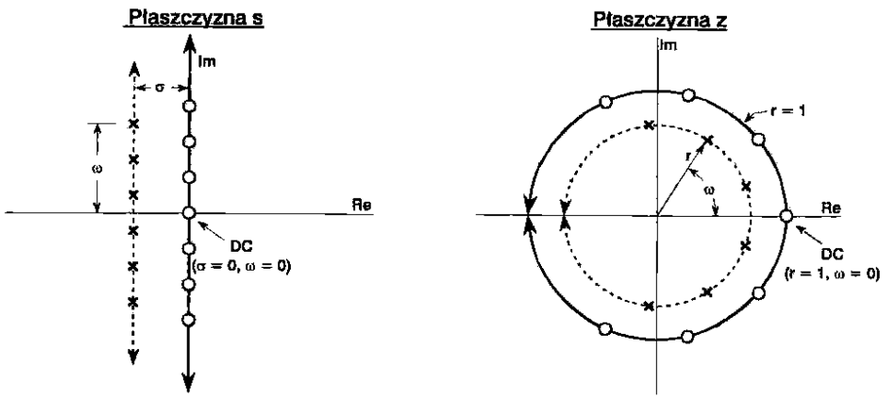
W przekształceniu Z zaś stosuje się zmienne r i ω, umieszczone we współrzędnych *biegunowych*. Odległość r od początku układu współrzędnych jest wartością współczynnika opadania wykładniczego. Odległość kątowa mierzona od dodatniej półosi poziomej ω jest częstotliwością kątową, czyli pulsacją. Taka geometria wynika z definicji z wyrażonej wzorem $z = re^{-j\omega}$. Można więc powiedzieć, że zmienna zespolona wyrażająca położenie na płaszczyźnie z jest utworzona jako kombinacja dwóch parametrów rzeczywistych we współrzędnych biegunowych.

Te różnice między przekształceniami skutkują tym, że *linie* pionowe na płaszczyźnie s odpowiadają *okręgom* na płaszczyźnie z. Na płaszczyźnie s na rys. 33.2 pokazano jako przykład rozmieszczenie biegunów i zer, w którym wszystkie bieguny i zera leżą na liniach pionowych. Równoważne bieguny i zera na płaszczyźnie z leżą na okręgach współśrodkowych w stosunku do początku układu współrzędnych. Można to zrozumieć, analizując podaną wcześniej zależność: $\sigma = -\ln(r)$. Na przykład oś pionowa płaszczyzny s (tj. $\sigma = 0$) odpowiada **okręgowi jednostkowemu** na płaszczyźnie z (tj. $r = 1$). Linie pionowe w lewej połówce płaszczyzny s odpowiadają okręgom położonym wewnątrz okręgu jednostkowego na płaszczyźnie z. Linie pionowe w prawej połówce płaszczyzny s odpowiadają zaś okręgom na zewnątrz koła jednostkowego na płaszczyźnie z. A zatem lewa i prawa strona płaszczyzny s są równoważne – odpowiednio – obszarom wewnątrz i na zewnątrz okręgu jednostkowego. System ciągły, na przykład, jest niestabilny, gdy bieguny zajmują *prawą połówkę* płaszczyzny s. W ten sam sposób system dyskretny jest niestabilny, gdy bieguny są *na zewnątrz* okręgu jednostkowego na płaszczyźnie z. Gdy sygnał w dziedzinie czasu jest

całkowicie rzeczywisty (najczęstszy przypadek), to dolna i górna połówka płaszczyzny z są wzajemnie odbiciami lustrzanymi, tak samo jak w dziedzinie s.

Zwróćcie szczególną uwagę na sposób zastosowania zmiennej częstotliwościowej ω w obu przekształceniach. Sinusoida ciągła może mieć dowolną częstotliwość z zakresu od 0 (DC) do nieskończoności. To znaczy, że ω na płaszczyźnie s musi mieć możliwość zmian od minus do plus nieskończoności. Natomiast sinusoida *dyskretna* może mieć częstotliwość tylko od 0 (DC) do połowy częstotliwości próbkowania. To znaczy częstotliwość wyrażona ułamkiem częstotliwości próbkowania musi się mieścić w granicach od 0 do 0,5, a wyrażona jako pulsacja (tj. $\omega = 2\pi f$) – w granicach od 0 do π. Jest to zgodne z geometrią płaszczyzny z, na której interpretujemy ω jako kąt wyrażony w *radianach*. To znaczy, że częstotliwości dodatnie odpowiadają kątom od 0 do π radianów, a ujemne – od 0 do $-\pi$ radianów. Ponieważ częstotliwość na płaszczyźnie z jest wyrażona w inny sposób niż na płaszczyźnie s, więc niektórzy autorzy używają odmiennych oznaczeń dla odróżnienia tych dwóch wielkości. Często stosowany jest zapis Ω dla oznaczenia częstotliwości w dziedzinie z oraz ω – dla częstotliwości w dziedzinie s. W tej książce stosujemy symbol ω dla częstotliwości w obu przekształceniach, ale w wielu innych publikacjach o DSP spotkacie dwa różne symbole.

Na płaszczyźnie s wartości znajdujące się wzdłuż osi pionowej tworzą odpowiedź częstotliwościową sytemu. Z tego wynika, że transformata Laplace'a wyznaczona dla $\sigma = 0$ jest równa transformacie Fouriera. W analogiczny sposób można stwierdzić, że odpowiedź częstotliwościową w dziedzinie z można wyznaczyć wzdłuż okręgu jednostkowego. Można to stwierdzić, określając ze wzoru 33.1 transformatę z dla $r = 1$, w wyniku czego wzór upraszcza się do dyskretnej w czasie transformaty Fouriera (DTFT). To powoduje umieszczenie częstotliwości zerowej w punkcie o wartości *jeden* na osi poziomej na płaszczyźnie z. Dodatnie częstotliwości widma są rozmieszczone w prawo od tego punktu w kierunku przeciwnym do ruchu wskazówek zegara i zajmują górną połowę okręgu. Częstotliwości ujemne są zaś rozmieszczone od punktu częstotliwości zerowej w kierunku zgodnym z ruchem wskazówek zegara, tworząc dolną połówkę okręgu. Dodatnie i ujemne częstotliwości widma spotykają się we wspólnych punktach $\omega = \pi$ i $\omega = -\pi$. Ta geometria kołowa dotyczy też widma częstotliwościowego sygnału dyskretnego, który jest *okresowy*. To znaczy, że gdy kąt częstotliwości przekroczy wartość π, to występują takie same wartości jak między 0 a π. Poruszając się wokół okręgu, ciągle na nowo widzicie powtarzające się te same wartości.



Rys. 33.2 Zależność między płaszczyznami s i z. Płaszczyzna s jest w układzie współrzędnych prostokątnych, w którym σ wyraża odległość wzdłuż osi rzeczywistej (poziomej) a ω – wzdłuż osi urojonej (pionowej). Płaszczyzna z zaś jest we współrzędnych biegunowych, gdzie r jest odległością od początku układu współrzędnych a ω – kątem mierzonym w stosunku do dodatniej półosi poziomej. Linie pionowe na płaszczyźnie s, takie jak pokazane na rysunku jako przykład biegunów i zer, odpowiadają okręgom na płaszczyźnie z

Analiza systemów rekursywnych

Jak podano w rozdziale 19, filtr rekursywny jest opisany **równaniem różnicowym:**

$$y[n]=a_0x[n]+a_1x[n-1]+a_2x[n-2]+...+b_1y[n-1]+b_2y[n-2]+b_3y[n-3]+...$$

Wzór 33.2. Równanie różnicowe. Szczegóły można znaleźć w rozdziale 19

gdzie $x[\ ]$ i $y[\ ]$ są sygnałami wejściowym i wyjściowym, a wyrazy a, b – **współczynnikami rekursywnymi**. Oczywistym zastosowaniem tego wzoru jest opisanie, w jaki sposób programista może zaimplementować filtr. Równie ważnym aspektem tego wzoru jest wyrażenie zależności matematycznej między wejściem a wyjściem, która musi być ciągle spełniona. O ile działanie systemów ciągłych jest opisane równaniami *różniczkowymi*, to systemy dyskretne działają zgodnie z równaniami *różnicowymi*. Z podanej zależności możemy wyprowadzić najważniejsze charakterystyki systemu – odpowiedzi: impulsową, skokową i częstotliwościową, rozmieszczenie biegunów i zer itd.

Rozpoczniemy analizę od poddania przekształceniu Z (wzór 33.1) obu stron równania 33.2 Chcemy bowiem zobaczyć, jak ta zależność wygląda w dziedzinie z. Stosując trochę algebry, możemy uzyskać zależność typu $Y[z]/X[z]$, czyli stosunek sygnału wyjściowego do wejściowego, przy czym oba sygnały są wyrażone w dziedzinie z. Tak samo jak w przekształceniu Laplace'a, ten stosunek nosi nazwę **funkcji przenoszenia systemu**, oznaczanej jako $H[z]$. A oto, co uzyskujemy:

$$H[z]=\frac{a_0+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}+...}{1-b_1z^{-1}-b_2z^{-2}-b_3z^{-3}-...}$$

Wzór 33.3. Funkcja przenoszenia w postaci wielomianowej. W tej zależności współczynniki rekursywne są łatwe do bezpośredniej identyfikacji

Jest to jeden z dwóch możliwych sposobów zapisu funkcji przenoszenia. Ta forma zapisu jest ważna, gdyż w takim zapisie są bezpośrednio zawarte współczynniki rekursywne. Załóżmy, na przykład, że znamy współczynniki rekursywne filtru cyfrowego, na przykład takie, jak w poniższej tabelce:

$a_0 = 0{,}389$	
$a_1 = -1{,}558$	$b_1 = 2{,}161$
$a_2 = 2{,}338$	$b_2 = -2{,}033$
$a_3 = -1{,}558$	$b_3 = 0{,}878$
$a_4 = 0{,}389$	$b_4 = -0{,}161$

Nie wdając się w zawiłości algebry liczb zespolonych, możemy bezpośrednio zapisać funkcję przenoszenia systemu:

$$H[z]=\frac{0{,}389-1{,}558z^{-1}+2{,}338z^{-2}-1{,}558z^{-3}+0{,}389z^{-4}}{1-2{,}161z^{-1}+2{,}033z^{-2}-0{,}878z^{-3}+0{,}161z^{-4}}$$

Zwróćcie uwagę, że współczynniki „b" wchodzą do wzoru ze znakiem *ujemnym* przed nimi. Niektórzy autorzy zapisują ten wzór inaczej – stosując we wzorze dodawanie, lecz zmieniając znak wszystkich współczynników „b". I tu pojawia się trudność. Jeśli macie dany zestaw współczynników rekursywnych (taki jak w powyższej tabelce lub jak w programie do programowania filtrów), to jest szansa 50% na 50%, że współczynniki „b" będą miały odwrotne, niż się spodziewacie, znaki. Jeśli nie wychwycicie tej rozbieżności, to filtr będzie przerażająco niestabilny.

Wzór 33.3 wyraża funkcję przenoszenia z zastosowaniem *ujemnych* wykładników potęg z, tzn. z^{-1}, z^{-2}, z^{-3} itd. Po wstawieniu właściwego zestawu współczynników rekursywnych możemy przekształcić wzór do wygodniejszej postaci z potęgami dodatnimi, tzn. z, z^2, z^3 itd. W rozważanym przykładzie mnożąc licznik i mianownik przez z^4, uzyskujemy:

$$H[z]=\frac{0{,}389z^4-1{,}558z^3+2{,}338z^2-1{,}558z+0{,}389}{z^4-2{,}161z^3+2{,}033z^2-0{,}878z+0{,}161}$$

Dodatnie wykładniki potęg są często łatwiejsze w stosowaniu, a są *wymagane* w niektórych metodach przekształcenia Z. Dlaczego nie zapisać wzoru 33.3, stosując dodatnie wykładniki i całkiem zapomnieć o ujemnych? Nie możemy tego zrobić. Tę sztuczkę z pomnożeniem licznika i mianownika przez z w najwyższej potędze (jak z^4 w omawianym przykładzie) możemy zastosować tylko wówczas, gdy jest już znana liczba współczynników rekursywnych. Wzór 33.3 zaś jest zapisany dla *dowolnej* liczby współczynników. Prawdą jest, że w DSP są rutynowo stosowane zarówno wykładniki dodatnie, jak i ujemne i musicie wiedzieć, jak przechodzić od jednych do drugich.

Funkcja przenoszenia systemu rekursywnego jest użyteczna, gdyż można nią operować takimi sposobami, których nie da się zastosować do współczynników rekursywnych. To obejmuje takie zadania, jak kaskadowe i równoległe łączenie stopni w jeden system, projektowanie filtrów przez określenie położeń biegunów i zer, przekształcanie filtrów analogowych w cyfrowe itd. Te operacje realizuje się przez działania algebraiczne wykonywane w dziedzinie z – takie, jak mnożenie, dodawanie i rozkładanie na czynniki. Po wykonaniu tych operacji nadaje się funkcji przenoszenia postać taką, jak we wzorze 33.3, co umożliwia określenie nowych współczynników rekursywnych.

Ważną cechą dziedziny s, podobnie jak dziedziny z, jest możliwość wyrażenie funkcji przenoszenia przez **bieguny** i **zera**. Jest to druga ogólna postać funkcji przenoszenia w dziedzinie z:

$$H[z]=\frac{(z-z_1)(z-z_2)(z-z_3)}{(z-p_1)(z-p_2)(z-p_3)}$$

Wzór 33.4. Funkcja przenoszenia wyrażona przez bieguny i zera

Każdy z biegunów (p_1, p_2, p_3, ...) i każde z zer (z_1, z_2, z_3, ...) są liczbami zespolonymi. Aby przejść od wzoru 33.4 do 33.3, trzeba wykonać wymnożenie i gromadzenie wyrazów podobnych. Wymaga to wprawdzie żmudnych operacji algebraicznych, ale w swej istocie są one proste i łatwe do zapisania w formie procedury komputerowej. Przechodzenie z wzoru 33.3 do 33.4 jest trudniejsze, gdyż wymaga *rozkładania wielomianów na czynniki*. Jak to rozważano w rozdziale 32, równanie kwadratowe można wykorzystać do rozkładania na czynniki, jeśli funkcja przenoszenia jest rzędu drugiego lub mniejszego (tzn. nie ma potęg z większych niż z^2). W przypadku systemów rzędu wyższego niż drugi nie mogą – ogólnie biorąc – być stosowane metody algebraiczne; trzeba korzystać z metod numerycznych. Taka potrzeba, na szczęście, rzadko się pojawia; projektowanie filtrów cyfrowych *zaczyna się* od rozmieszczenia biegunów i zer (wzór 33.4) i *kończy* na współczynnikach rekursywnych (wzór 33.3), a nie odwrotnie.

Tak jak wszystkie liczby zespolone, położenia zer i biegunów mogą być wyrażane bądź we współrzędnych prostokątnych, bądź w biegunowych. Zaletą zapisu biegunowego jest lepsza zgodność z naturalną organizacją płaszczyzny z. Natomiast zapis we współrzędnych prostokątnych jest na ogół preferowany w operacjach matematycznych. Łatwiej jest bowiem operować wyrażeniem $\sigma + j\omega$ niż $re^{j\omega}$.

Jak przykład zastosowania wzorów 33.3 i 33.4 zaprojektujemy filtr pasmowozaporowy. Projektowanie będzie się składać z następujących kroków: 1) określenie rozmieszczenia biegunów i zer na płaszczyźnie z; 2) zapisanie funkcji przenoszenia w postaci zgodnej z wzorem 33.4; 3) przekształcenie funkcji przenoszenia do postaci według wzoru 33.3 oraz 4) odczytanie z tego wzoru współczynników rekursywnych koniecznych do implementacji filtru. Na rys. 33.3 przedstawiono przykład, z którego będziemy korzystać – filtr pasmowozaporowy utworzony z dwóch biegunów i dwóch zer o położeniach:

We współrzędnych biegunowych:	We współrzędnych prostokątnych:
$z_1 = 1{,}00\, e^{j(\pi/4)}$	$z_1 = 0{,}7071 + j\,0{,}7071$
$z_2 = 1{,}00\, e^{j-(\pi/4)}$	$z_1 = 0{,}7071 - j\,0{,}7071$
$p_1 = 0{,}90\, e^{j(\pi/4)}$	$p_1 = 0{,}6364 + j\,0{,}6364$
$p_2 = 0{,}90\, e^{j-(\pi/4)}$	$p_2 = 0{,}6364 - j\,0{,}6364$

W celu zrozumienia, dlaczego jest to filtr pasmowozaporowy, porównajcie to rozmieszczenie biegunów i zer z rys. 32.6 obrazującym filtr pasmowozaporowy na płaszczyźnie s.

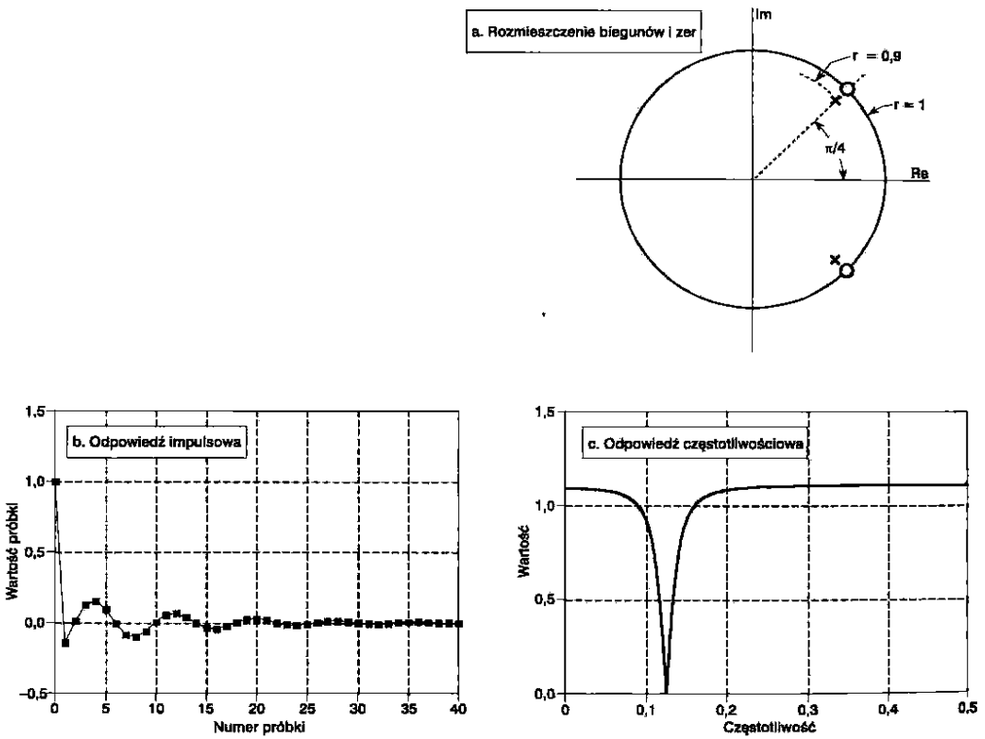
Jedyna różnica jest taka, że aby wyznaczyć odpowiedź częstotliwościową na płaszczyźnie z, poruszamy się po *okręgu jednostkowym*; po *osi pionowej* zaś, gdy wyznaczamy tę odpowiedź na płaszczyźnie s. Z biegunowego rozmieszczenia biegunów i zer wnioskujemy, że wycięcie na charakterystyce (pasmo zaporowe) ma miejsce przy pulsacji $\pi/4$, co odpowiada 0,125 częstotliwości próbkowania.

Ponieważ położenia biegunów i zer są znane, więc funkcję przenoszenia można zapisać według wzoru 33.4 przez proste wstawienie wartości:

$$H(z)=\frac{[z-(0,7071+j0,7071)][z-(0,7071-j0,7071)]}{[z-(0,6364+j0,6364)][z-(0,6364-j0,6364)]}$$

Trzeba przekształcić to wyrażenie do postaci według wzoru 33.3, aby można było określić współczynniki rekursywne implementujące ten filtr. Najpierw rozwińmy wyrażenie, wymnażając wszystkie jego wyrazy:

$$H(z)=\frac{z^2-0,7071z+j0,7071z-0,7071z+0,7071^2-j0,7071^2-j0,7071z+j0,7071^2-j^2\,0,7071^2}{z^2-0,6364z+j0,6364z-0,6364z+0,6364^2-j0,6364^2-j0,6364z+j0,6364^2-j^2\,0,6364^2}$$



Rys. 33.3. Projektowanie filtru pasmowozaporowego w dziedzinie z. Projektowanie rozpoczyna się od umiejscowienia dwóch biegunów i dwóch zer na płaszczyźnie z (rys. a). Uzyskane odpowiedzi – impulsowa i częstotliwościowa – są przedstawione, odpowiednio, na rys. b), c). Stromość wycięcia na charakterystyce jest ustalana odległością biegunów od zer

Następnie gromadzimy wyrazy podobne i przeprowadzamy ich redukcję. Jeśli górna połówka płaszczyzny z jest lustrzanym odbiciem dolnej (co ma zawsze miejsce, gdy mamy do czynienia z *rzeczywistą* odpowiedzią impulsową), to wszystkie wyrazy zawierające j ulegną redukcji:

$$H[z]=\frac{1,000-1,414z+1,000z^2}{0,810-1,273z+1,000z^2}$$

Chociaż jest to stosunek dwóch wielomianów, to nie ma w nim ujemnych wykładników potęg z, czego wymaga wzór 3.33. Można zmienić tę sytuację dzieląc licznik i mianownik przez z w najwyższej potędze w tym wyrażeniu, a więc przez z^2:

$$H[z]=\frac{1,000-1,414z^{-1}+1,000z^{-2}}{1,000-1,273z^{-1}+0,810z^{-2}}$$

Funkcja przenoszenia ma teraz postać zgodną ze wzorem 33.3, więc można z tej zależności wypisać współczynniki rekursywne:

$a_0 = 1,000$

$a_1 = -1,414$ $\quad b_1 = 1,273$

$a_2 = 1,000$ $\quad b_2 = -0,810$

Ten przykład obrazuje ogólną metodykę wyznaczania współczynników rekursywnych na podstawie rozmieszczenia biegunów i zer. W szczególnych przypadkach jest możliwe wyprowadzanie prostszych wzorów dających bezpośrednie zależności między współczynnikami rekursywnymi a położeniami biegunów i zer. Na przykład dla systemu zawierającego dwa bieguny i dwa zera, zwanego **bikwadratowym**, zależności te są następujące:

$$a_0 = 1$$
$$a_1 = -2r_0\cos(\omega_0)$$
$$a_2 = r_0^2$$
$$b_1 = 2r_p\cos(\omega_p)$$
$$b_2 = -r_p^2$$

Wzór 33.5. Wzory do projektowania filtru bikwadratowego. Wzory określają zależności współczynników rekursywnych a_0, a_1, a_2, b_1, b_2 od położeń biegunów r_p i ω_p i zer r_0 i ω_0

Jak określimy odpowiedź częstotliwościową, mając już wyznaczoną funkcję przenoszenia? Są trzy metody: jedna matematyczna i dwie komputerowe (programowe). Metoda matematyczna polega na znajdowaniu na płaszczyźnie z wartości leżących na okręgu jednostkowym. Wykonuje się to przez określanie wartości funkcji przenoszenia $H(z)$ dla $r = 1$. Zaczynamy od zapisania funkcji przenoszenia w postaci takiej, jak we wzorze 33.4 lub 33.4. Następnie każde z zastępujemy przez $e^{-j\omega}$ (czyli $re^{-j\omega}$ dla $r = 1$). W ten sposób uzyskujemy wyrażenie matematyczne na odpowiedź częstotliwościową $H(\omega)$. Trudność leży w tym, że uzyskane wyrażenie ma bardzo niedogodną postać. Trzeba wielu operacji algebraicznych, aby uzyskać jakieś rozpoznawalne wielkości, takie jak np. moduł i fazę. Ta metoda daje wprawdzie ścisły wzór na odpowiedź częstotliwościową, lecz jest trudna do zautomatyzowania programami komputerowymi, takimi jakie są wymagane w pakietach do programowania filtrów.

W drugiej metodzie określania odpowiedzi częstotliwościowej też wyznacza się wartości na kole jednostkowym na płaszczyźnie z. Różnica polega na tym, że w tej metodzie obliczamy tylko *próbki* odpowiedzi częstotliwościowej, a nie matematyczne wyrażenie opisujące całą krzywą. Program komputerowy przebiega w pętli przez, powiedzmy, 1000 częstotliwości rozmieszczonych równomiernie między $\omega = 0$ a $\omega = \pi$. Wyobraźcie sobie mrówkę poruszającą się między 1000 dyskretnymi punktami w górnej połówce okręgu jednostkowego na płaszczyźnie z. W każdym z tych punktów określa się moduł i fazę odpowiedzi częstotliwościowej przez wyznaczanie wartości funkcji przenoszenia.

Ta metoda działa dobrze i jest często stosowana w pakietach projektowania filtrów. Jej głównym ograniczeniem jest nieuwzględnianie *szumu zaokrąglania* wpływającego na charakterystyki systemu. Nawet jeśli wyznaczona tą metodą odpowiedź częstotliwościowa wygląda doskonale, to zrealizowany system może być całkowicie niestabilny!

To nas prowadzi do trzeciej metody – wyznaczania odpowiedzi częstotliwościowej na podstawie współczynników rekursywnych stosowanych do implementacji filtru. Rozpoczynamy od znalezienia odpowiedzi impulsowej filtru, przepuszczając impuls przez filtr. W następnym kroku poddajemy odpowiedź impulsową przekształceniu DFT (oczywiście stosując FFT), aby wyznaczyć odpowiedź częstotliwościową systemu. W tej procedurze jedynym punktem krytycznym, o którym trzeba pamiętać, jest konieczność zebrania wystarczająco dużej liczby próbek odpowiedzi impulsowej, tak aby próbki pominięte *nie miały znaczenia*. Aczkolwiek na temat kryteriów teoretycznych związanych z tą kwestią można by napisać całe książki, to jednak reguły praktyczne są znacznie prostsze. Stosujcie tak wiele próbek, ile *uznajecie za* konieczne. Po wyznaczeniu odpowiedzi częstotliwościowej cofnijcie się i powtórzcie procedurę, stosując dwukrotnie większą liczbę próbek. Jeśli obie uzyskane odpowiedzi impulsowe są wystarczająco podobne, to możecie być pewni, że ograniczenie odpowiedzi impulsowej nie wprowadziło was w błąd.

Stopnie kaskadowe i równoległe

Bardziej skomplikowane filtry rekursywne projektuje się zwykle stopień po stopniu, aby uniknąć żmudnych operacji algebraicznych w dziedzinie z. Na rys. 33.4 przedstawiono dwa sposoby łączenia pojedynczych stopni – łączenie kaskadowe lub łączenie równoległe z sumowaniem wyjść. Można na przykład połączyć kaskadowo filtry dolno- i górnoprzepustowy, aby uzyskać filtr *pasmowoprzepustowy*. W podobny sposób równoległe połączenie filtrów dolno- i górnoprzepustowego może dać filtr *pasmowozaporowy*. Dwa filtry, które łączymy ze sobą, nazwiemy *systemem 1* i *systemem 2*, a ich współczynniki rekursywne oznaczymy, odpowiednio, przez a_0, a_1, a_2, b_1, b_2 oraz A_0, A_1, A_2, B_1, B_2. Naszym celem jest połączenie ze sobą tych stopni (kaskadowo lub równolegle) w jeden filtr rekursywny, który nazwiemy *systemem 3*, o współczynnikach rekursywnych a_0, a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4.

Jak pamiętacie z poprzednich rozdziałów, odpowiedź częstotliwościową systemu filtrów łączonych kaskadowo uzyskuje się przez *mnożenie* odpowiedzi poszczególnych stopni, a w przypadku połączenia równoległego – przez ich *dodawanie*. Takie same reguły obowiązują dla funkcji przenoszenia w dziedzinie z. Dzięki temu można łączyć ze sobą systemy rekursywne, przenosząc cały problem do dziedziny z, wykonując wymagane mnożenie lub dodawanie i powracając następnie do współczynników rekursywnych systemu finalnego.

Jako przykład tej metody przeprowadzimy operacje algebraiczne łączące dwa stopnie bikwadratowe w kaskadę. Funkcję przenoszenia każdego stopnia określamy, wpisując do wzoru 33.3 odpowiednie współczynniki rekursywne. Następnie znajdujemy funkcję przenoszenia całego systemu $H[z]$, wymnażając funkcje przenoszenia obu stopni:

$$H[z]=\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1-b_1z^{-1}-b_2z^{-2}}\times\frac{A_0+A_1z^{-1}+A_2z^{-2}}{1-B_1z^{-1}-B_2z^{-2}}$$

Wymnażając wielomiany i grupując wyrazy podobne, uzyskujemy:

$$H[z]=\frac{a_0A_0+(a_0A_1+a_1A_0)z^{-1}+(a_0A_2+a_1A_1+a_2A_0)z^{-2}+(a_1A_2+a_2A_1)z^{-3}+(a_2A_2)z^{-4}}{1-(b_1+B_1)z^{-1}-(b_2+B_2-b_1B_1)z^{-2}-(-b_1B_2-b_2B_1)z^{-3}-(-b_2B_2)z^{-4}}$$

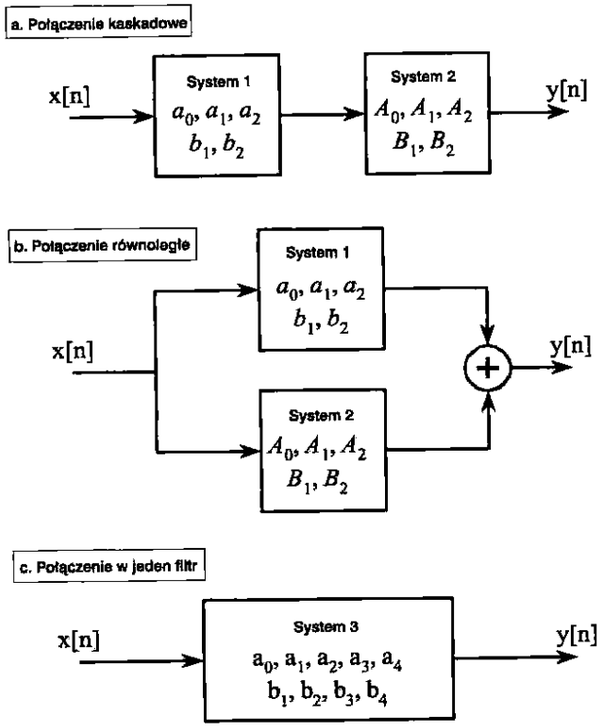
Ponieważ powyższe wyrażenie ma postać zgodną ze wzorem 33.3, więc możemy bezpośrednio wypisać współczynniki rekursywne implementujące cały system kaskadowy:

$$\begin{aligned}
a_0&=a_0A_0 & b_1&=b_1+B_1\\
a_1&=a_0A_1+a_1A_0 & b_2&=b_2+B_2-b_1B_1\\
a_2&=a_0A_2+a_1A_1+a_2A_0 & b_3&=-b_1B_2-b_2B_1\\
a_3&=a_1A_2+a_2A_1 & b_4&=-b_2B_2\\
a_4&=a_2A_2
\end{aligned}$$

W tej metodzie oczywistą trudnością jest duża liczba operacji algebraicznych koniecznych do wymnożenia wielomianów i pogrupowania wyrazów. Cały algorytm można, na szczęście, wyrazić prostym programem komputerowym podanym w tablicy 33.1. Chociaż połączenia kaskadowe i równoległe wymagają innych operacji matematycznych, to zastosowane programy są prawie identyczne. Oba algorytmy różnią się tylko jednym wierszem programu, dlatego można je łatwo połączyć w jeden program.

Program działa, zamieniając współczynniki rekursywne każdego ze stopni w funkcje przenoszenia w postaci zgodnej z wzorem 33.3 (wiersze 220–270). Po połączeniu ze sobą – w odpowiedni sposób – tych funkcji przenoszenia (wiersze 290–380), informacja wraca w formie współczynników rekursywnych (wiersze od 400 do 430).

Istotą tego programu jest sposób wyrażania wielomianów funkcji przenoszenia i ich łączenia ze sobą. Na przykład licznik funkcji przenoszenia pierwszego z łączonych ze sobą stopni jest: $a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}$... Ten wielomian jest w programie wyrażony przez zapisanie współczynników a_0, a_1, a_2, a_3... w matrycy: A1[0], A1[1], A1[2], A1[3] ... Podobnie licznik funkcji przenoszenia drugiego z łączonych stopni jest wyrażony przez wartości zapi-



Rys. 33.4. Kaskadowe i równoległe łączenie stopni. Przekształcenie Z umożliwia łączenie stopni rekursywnych kaskadowo (rys. a) lub równolegle (rys. b) w jeden system (rys. c)

sane w: A2[0], A2[1], A2[2], A2[3] ..., a licznik funkcji przenoszenia systemu uzyskanego w wyniku połączenia w: A3[0], A3[1], A3[2], A3[3] ... Zasada programu polega na wyrażaniu *wielomianów* i operowaniu nimi z odniesieniem tylko do ich *współczynników*. Powstaje pytanie, jak obliczyć A[3], wiedząc, że A[1], A[2] i A[3] wyrażają wielomiany. Odpowiedź jest znana – przy mnożeniu wielomianów trzeba ich współczynniki poddać operacji *splotu*. Można to wyrazić wzorem: A1[ ] * A2[ ] = A3[ ]. Stosując standardowy algorytm splotu, można więc wyznaczyć funkcję przenoszenia stopni połączonych w kaskadę przez poddanie splotowi dwóch matryc liczników oraz dwóch matryc mianowników.

Nieco bardziej skomplikowana jest procedura równoległego łączenia stopni. Według zasad algebry ułamki są dodawane według wzoru:

$$\frac{w}{x}+\frac{y}{z}=\frac{w\cdot z+x\cdot y}{x\cdot z}$$

Ponieważ każda funkcja przenoszenia ma postać ułamka (stosunku dwóch wielomianów), więc przy równoległym łączeniu stopni trzeba pomnożyć przez siebie mianowniki funkcji przenoszenia, a licznik obliczyć jako sumę iloczynów liczników i mianowników wziętych przemiennie z obu ułamków. Z tego wynika, że mianownik funkcji przenoszenia oblicza się

Tablica 33.1. Kaskadowe i równoległe łączenie stopni. Ten program łączy kaskadowo lub równolegle współczynniki rekursywne stopni. Współczynniki rekursywne obu łączonych ze sobą stopni wprowadza się do programu w matrycach: A1[ ], B1[ ] i A[2], B2[ ]. Po wykonaniu programu, w matrycach A3[ ] i B3[ ] uzyskuje się współczynniki rekursywne implementujące cały system

```
100 'ŁĄCZENIE ZE SOBĄ WSPÓŁCZYNNIKÓW REKURSYWNYCH STOPNI KASKADOWYCH I RÓWNOLEGŁYCH
110 '
120 '                              'INICJALIZACJA ZMIENNYCH
130 DIM A1 [8], B1 [8]             'Współczynniki a i b dla systemu 1, jeden ze stopni
140 DIM A2[8], B2[8]               'Współczynniki a i b dla systemu 2, jeden ze stopni
150 DIM A3[16], B3[16]             'Współczynniki a i b dla systemu 3, system połączony
160 '
170                                'Wskaż połączenie kaskadowe lub równoległe
180 INPUT "Wprowadź 0 dla kaskady a 1 dla połączenia równoległego:", CP%
190 '
200 GOSUB XXXX                     'Podprogram "mityczny" do ładowania A1[ ], B1[ ], A2[ ], B2[ ]
210'
220 FOR I% = 0 TO 8                'Przekształć współczynniki rekursywne w funkcje przenoszenia
230   B2[I%] = -B2[I%]
240   B1[I%] = -B1[I%]
250 NEXT I%
260 B1[0] = 1
270 B2[0] = 1
280 '
290 FOR I% = 0 TO 16               'Przemnóż wielomiany metodą splotu
300   A3[I%] = 0
310   B3[I%] = 0
320   FOR J% = 0 TO 8
330     IF I%-J% < 0 OR I%-J% > 8 THEN GOTO 370
340     IF CP% = 0 THEN A3[I%] = A3[I%] + A1[J%] * A2[I%-J%]
350     IF CP% = 1 THEN A3[I%] = A3[I%] + A1 [J%] * B2[I%-J%] + A2[J%] * B1 [I%-J%]
360     B3[I%] = B3[I%] + B1[J%] * B2[I%-J%]
370   NEXT J%
380 NEXT I%
390 '
400 FOR I% = 0 TO 16               'Przekształć funkcje przenoszenia w współczynniki rekursywne.
410   B3[I%] = -B3[I%]
420 NEXT I%
430 B3[0] = 0
440 '                              'Współczynniki rekursywne połączonego systemu
450 END                            'znajdują się obecnie w A3[ ] i B3[ ].
```


tak samo, jak dla połączenia kaskadowego, lecz sposób obliczania licznika jest bardziej złożony. W wierszu 340 programu liczniki funkcji przenoszenia stopni łączonych *kaskadowo* są poddawane operacji splotu w celu znalezienia licznika funkcji przenoszenia systemu filtrów połączonych. W wierszu 350 licznik funkcji przenoszenia stopni łączonych *równolegle* jest obliczany jako suma dwóch liczników poddanych operacji splotu z dwoma mianownikami. W wierszu 360 następuje obliczenie mianownika dla obu rodzajów połączenia.

Inwersja widmowa

Technikę filtrów SOI, zwaną *inwersją widmową*, opisano w rozdziale 14. Jest to sposób zmiany odpowiedzi impulsowej filtru tak, aby odpowiedź częstotliwościowa została odwrócona. Wszystkie pasma przepustowe zamieniają się wtedy w zaporowe i vice versa. Na przykład filtr dolnoprzepustowy zamienia się w górnoprzepustowy, pasmowoprzepustowy w pasmowozaporowy itd. Podobną procedurę można przeprowadzić dla filtrów rekursywnych, lecz ze znacznie mniejszym powodzeniem.

Inwersja widmowa jest realizowana przez odejmowanie sygnału wyjściowego od sygnału oryginalnego (rys. 33.5). Na tę procedurę można spojrzeć jako na równoległe łączenie dwóch stopni, przy czym jeden z systemów jest systemem tożsamościowym (tzn. sygnał wyjściowy jest identyczny z wejściowym). Korzystając z takiego podejścia, można wykazać, że współczynniki „b" pozostają niezmienione, a zmodyfikowane współczynniki „a" są określone wzorami:

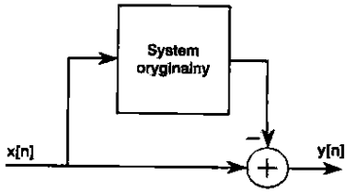
$$\begin{aligned} a_0 &= 1 - a_0 \\ a_1 &= -a_1 - b_1 \\ a_2 &= -a_2 - b_2 \\ a_3 &= -a_3 - b_3 \\ &\vdots \end{aligned}$$

Wzór 33.6. Inwersja widmowa. Odpowiedź częstotliwościowa filtru rekursywnego może zostać odwrócona (góra – dół) przez modyfikację współczynników „a" zgodnie z tymi wzorami. Współczynniki oryginalne zapisano drukiem pochyłym, a zmodyfikowane – prostym. Współczynniki „b" pozostają niezmienione. Ta metoda daje na ogół kiepskie wyniki

Na rys. 33.6 przedstawiono inwersję widmową dla dwóch często spotykanych odpowiedzi częstotliwościowych: filtru dolnoprzepustowego (rys. 33.6 a) i filtru pasmowozaporowego (wycinającego – rys. 33.6 c). W wyniku inwersji uzyskuje się, odpowiednio, filtr górnoprzepustowy (rys. 33.6 b) i pasmowoprzepustowy (rys. 33.6 d). Jak wyglądają te uzyskane odpowiedzi częstotliwościowe? Filtr górnoprzepustowy jest naprawdę okropny! Wprawdzie filtr pasmowoprzepustowy jest lepszy, lecz pik nie jest tak ostry, jak w filtrze pasmowozaporowym, z którego ten filtr został utworzony. Takie mierne wyniki szczególnie rozczarowują w porównaniu z doskonałymi parametrami przedstawionymi w rozdziale 14. Skąd taka różnica? Odpowiedź kryje się w charakterystyce, o której często zapomina się przy projektowaniu filtrów. Chodzi o *odpowiedź fazową*.

Aby wykazać, że faza jest winowajcą, rozważmy system zwany **transformatorem Hilberta**. Nie jest to jakieś specjalne urządzenie, lecz każdy system mający charakterystykę częstotliwościową: moduł = 1 i faza = 90 stopni w całym zakresie częstotliwości. Z tego wynika, że każda sinusoida po przejściu przez transformator Hilberta ma amplitudę niezmienioną, lecz fazę przesuniętą o ćwierć okresu. Transformatory Hilberta mogą być analogowe lub dyskretne (tzn. sprzętowe lub programowe) i są zwykle stosowane w telekomunikacji w różnych metodach modulacji i demodulacji.

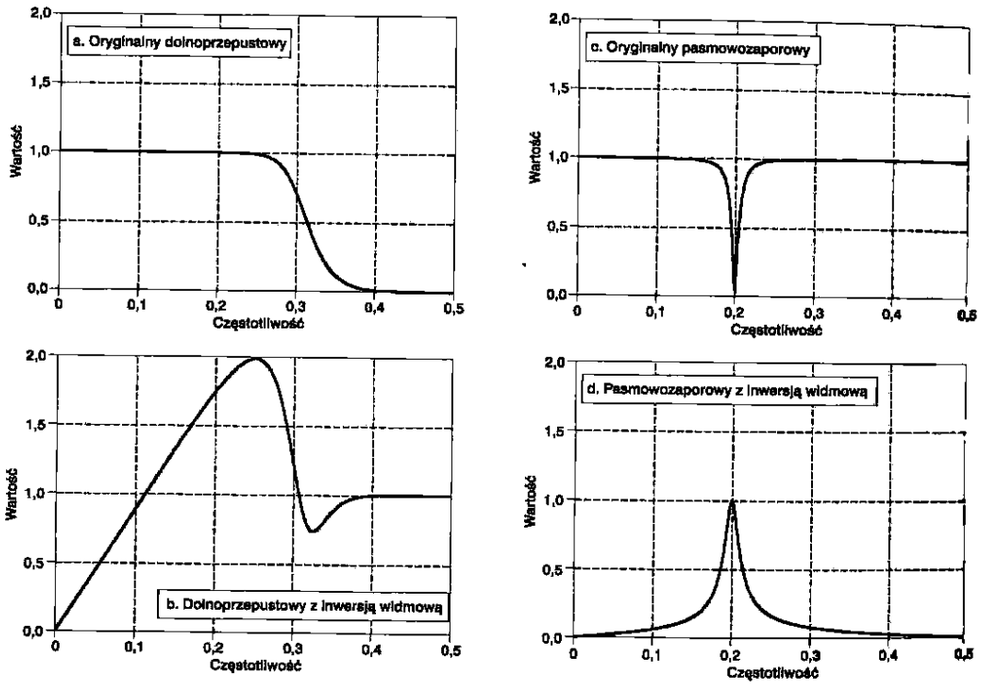
Przyjmujmy teraz, że dokonujemy inwersji widmowej transformatora Hilberta przez odjęcie jego sygnału wyjściowego od sygnału oryginalnego. Patrząc tylko na *moduły* odpowiedzi częstotliwościowych, doszlibyśmy do wniosku, że cały system ma na wyjściu *zero*. Moduł sygnału na wyjściu transformatora Hilberta jest bowiem identyczny z modułem sygnału oryginalnego i oba te sygnały będą się kasować. Jest to oczywiście stwierdzenie całkowicie niepoprawne.



Rys. 33.5. Inwersja widmowa. Jest to taka sama procedura, jak odejmowanie sygnału na wyjściu systemu od sygnału oryginalnego

Dwie sinusoidy będą się wzajemnie kasować tylko wówczas, jeśli mają takie same moduły oraz fazy. W rzeczywistości odpowiedź częstotliwościowa takiego systemu złożonego ma moduł $\sqrt{2}$ i przesunięcie fazy o –45 stopni. Sygnał wyjściowy – wbrew naszym naiwnym przypuszczeniom – nie jest równy zeru, lecz jest *większy* pod względem amplitudy niż sygnał wejściowy!

Inwersja widmowa dobrze zadziałała w rozważaniach w rozdziale 14 z powodu specyficznego rodzaju używanych tam filtrów. Są to filtry *zerofazowe*. To oznacza, że odpowiedź impulsowa filtru ma symetrię lewo – prawostronną. Jeśli system nie wprowadza żadnego przesunięcia fazy, to odejmowanie sygnału wyjściowego od wejściowego jest całkowicie uzależnione od modułów. Ponieważ mankamentem filtrów rekursywnych jest przesunięcie fazy, więc przez inwersję widmową uzyskuje się na ogół niezadowalające właściwości filtrów.



Rys. 33.6 Przykłady inwersji widmowej. Na rys. a) przedstawiono odpowiedź częstotliwościową 6-biegunowego, dolnoprzepustowego filtru Butterwortha. Na rys. b) pokazano odpowiadającą jej odpowiedź filtru górnoprzepustowego uzyskaną w wyniku inwersji widmowej – powstał tu zupełny mętlik! Bardziej udany przykład inwersji pokazano na ryd. c) oraz d), gdzie filtr pasmowozaporowy (typu notch) zostaje przekształcony w odpowiedź filtru pasmowoprzepustowego

Zmiany wzmocnienia

Przypuśćmy, że mamy filtr rekursywny i chcemy zmodyfikować współczynniki rekursywne w taki sposób, żeby zmienić amplitudę sygnału wyjściowego. Może to być potrzebne na przykład po to, aby filtr miał wzmocnienie równe jedności w paśmie przepustowym. Metoda osiągnięcia tego celu jest prosta – trzeba pomnożyć współczynniki „a" przez taki czynnik, o jaki chcemy zmienić wzmocnienie, a współczynniki „b" pozostawić niezmienione.

Przed ustawieniem wzmocnienia chcielibyśmy zapewne znać jego aktualną wartość. Procedura określenia tej wartości zależy od rodzaju filtru, gdyż wzmocnienie musi być wyznaczone dla jakiejś częstotliwości w paśmie przepustowym. W filtrach dolnoprzepustowych mierzy się wzmocnienie przy częstotliwości *zero*, a w górnoprzepustowych stosuje się częstotliwość równą 0,5, czyli połowie dopuszczalnej częstotliwości maksymalnej. Łatwo jest wyprowadzić wzory na wzmocnienie dla obu tych szczególnych częstotliwości. A oto, jak się dochodzi do tych wzorów.

Najpierw wyprowadzimy wzór na wzmocnienie przy częstotliwości zero. Metoda polega na wymuszeniu we wszystkich próbkach wejściowych wartości *jeden*, co spowoduje, że wszystkie próbki wyjściowe będą miały wartość wzmocnienia systemu G, które właśnie próbujemy określić. Zaczniemy od zapisania równania rekursywnego, a więc zależności matematycznej między sygnałami wyjściowym a wejściowym:

$$y[n] = a_0x[n] + a_1x[n-1] + a_2x[n-2] + ... + b_1y[n-1] + b_2y[n-2] + b_3y[n-3] + ...$$

Następnie podstawiamy *jeden* w miejsce każdej próbki wejściowej i G – każdej wyjściowej. Inaczej mówiąc, zmuszamy system do pracy przy częstotliwości zero. Uzyskujemy wyrażenie:

$$G = a_0 + a_1 + a_2 + a_3 + ... + b_1G + b_2G + b_3G + b_4G...$$

Przez rozwiązanie powyższego równania uzyskujemy wartość wzmocnienia G dla częstotliwości zero wyrażoną przez współczynniki rekursywne:

$$G = \frac{a_0 + a_1 + a_2 + a_3 ...}{1 - (b_1 + b_2 + b_3 ...)}$$

Wzór 33.7. Wzmocnienie filtrów rekursywnych przy częstotliwości zero. Jest to zależność tego wzmocnienia od współczynników rekursywnych

Aby uzyskać filtr o wzmocnieniu *jeden* dla częstotliwości zero, należy obliczyć na podstawie tej zależności wzmocnienie istniejące i następnie podzielić wszystkie współczynniki „a" przez G.

W podobny sposób wyznacza się wzmocnienie dla częstotliwości 0,5. Wymuszamy pracę sygnałów wejściowych i wyjściowych przy tej częstotliwości i patrzymy, jaka jest odpowiedź systemu. Przy częstotliwości 0,5 wartości próbek sygnału wejściowego są przemiennie –1 i 1, a więc kolejne próbki są: 1, –1, 1, –1, 1, –1, 1 itd. Wynikający z nich sygnał wyjściowy też zmienia znak, lecz ma wartość równą wzmocnieniu systemu: G, $-G$, G, $-G$, G, $-G$ itd. Podstawiając te wartości do równania rekursywnego, mamy:

$$G = a_0 - a_1 + a_2 - a_3 + ... - b_1G + b_2G - b_3G + b_4G...$$

Wyznaczając z powyższego równania G, uzyskujemy wzmocnienie systemu dla częstotliwości 0,5 wyrażone przez współczynniki rekursywne:

$$G = \frac{a_0 - a_1 + a_2 - a_3 + a_4 ...}{1 - (-b_1 + b_2 - b_3 + b_4 ...)}$$

Wzór 33.8. Wzmocnienie przy częstotliwości maksymalnej. Ten wzór określa zależność wzmocnienia przy częstotliwości 0,5 od współczynników rekursywnych systemu

Podobnie jak poprzednio, filtr można znormalizować dla wzmocnienia jednostkowego, dzieląc wszystkie współczynniki „a" przez obliczoną wartość G. Obliczanie wzoru 33.8 programem komputerowym wymaga metody generowania znaku minusa dla współczynników nieparzystych i plusa dla parzystych. Sposobem najczęściej stosowanym jest mnożenie każdego współczynnika przez $(-1)^k$, gdzie k jest indeksem kolejnego współczynnika. A zatem dla wartości k: 0, 1, 2, 3, 4 , 5, 6 itd., wyrażenie $(-1)^k$ przyjmuje wartości; 1, –1, 1, –1, 1, –1, 1 itd.

Projektowanie filtru Czebyszewa-Butterwortha

Powszechnie stosowaną metodę projektowania rekursywnych filtrów cyfrowych przedstawiono w rozdziale 20, podając program komputerowy dla filtrów Czebyszewa-Butterwortha. Program zaczyna się od rozmieszczenia biegunów i zer filtru *analogowego* na płaszczyźnie *s*. Program przetwarza następnie to rozmieszczenie w żądany filtr *cyfrowy* przez kilka *przekształceń* matematycznych. W celu uproszczenia operacji algebraicznych, filtr projektuje się jako kaskadowe połączenie kilku stopni, z których każdy implementuje jedną parę biegunów. Współczynniki rekursywne poszczególnych stopni są następnie łączone ze sobą, dając współczynniki rekursywne całego filtru. Jest to bardzo wysublimowany i skomplikowany algorytm, akurat świetny jako zakończenie tej książki. A oto, jak on działa.

Pętla

Na rys. 33.7 przedstawiono program i sieć działań tej metody, powtórzone z rozdziału 20. Po inicjalizacji i wprowadzeniu parametrów następuje główna część programu, którą jest pętla obejmująca wszystkie pary biegunów w filtrze. Ta pętla jest sterowana blokiem 11 w sieci działań oraz pętlą FOR-NEXT w wierszach 320 i 460 programu. Na przykład w filtrze 6-biegunowym pętla będzie obiegana trzy razy, z indeksem pętli P% przyjmującym wartości 1, 2, 3 Tak więc filtr 6-biegunowy będzie implementowany w trzech stopniach, z dwoma biegunami na stopień.

Łączenie współczynników

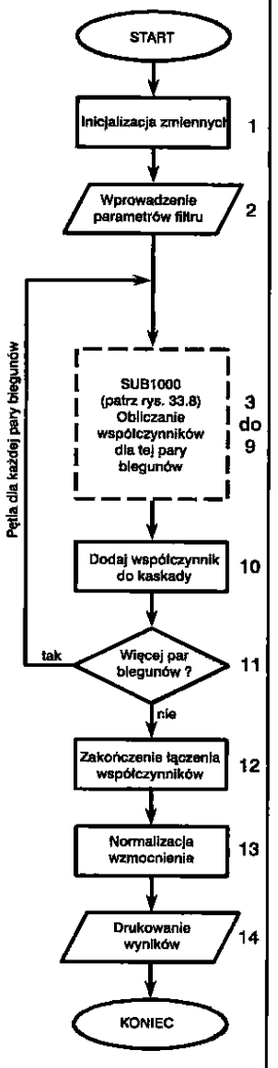
Podczas każdej pętli, w podprogramie 1000 (listing podano na rys. 33.8) są obliczane współczynniki rekursywne *danego stopnia*. Jako wynik podprogramu pojawiają się one w pięciu zmiennych: A0, A1, A2, B1, B2. W bloku 10 sieci działań (wiersze 360–440) te współczynniki są łączone ze współczynnikami wszystkich poprzednich stopni, przechowywanymi w matrycach A[ ] i B[ ]. Na końcu pierwszej pętli w A[ ] i B[ ] znajdują się współczynniki pierwszego stopnia filtru. Na końcu drugiej pętli są tam współczynniki kaskady złożonej ze stopni pierwszego i drugiego. Po wykonaniu wszystkich pętli w A[ ] i B[ ] są zapisane współczynniki konieczne do zaimplementowania całego filtru.

Współczynniki są łączone ze sobą tak, jak to poprzednio wskazano w tablicy 33.1, z kilkoma modyfikacjami dającymi większą zwartość programu. Po pierwsze, indeks matryc A[ ] i B[ ] zostaje przesunięty o *dwa* podczas pętli. Na przykład a_0 jest zapisywane w A[2], a_1 i b_1 są zapisywane w A[3] i B[3] itd. Robi się to dlatego, aby program nie próbował uzyskiwać dostępu do wartości poza ustalonymi matrycami. To przesunięcie jest usuwane w bloku 12 (wiersze 480–520), dzięki czemu uzyskane końcowe wartości współczynników rekursywnych znajdują się w A[ ] i B[ ] z indeksami bez przesunięcia.

Po drugie, A[ ] i B[ ] muszą być inicjalizowane współczynnikami odpowiadającymi systemowi tożsamościowemu, a nie samymi zerami. To jest wykonywane w wierszach od 180 do 240. Podczas pierwszej pętli współczynniki dla pierwszego stopnia filtru są łączone z informacją istniejącą początkowo w tych matrycach. Jeśli na początku były same zera, to w tych matrycach zawsze pozostaną zera. Po trzecie, stosuje się dwie matryce tymczasowe TA[ ] i TB[ ]. Są w nich przechowywane stare wartości A[ ] i B[ ] podczas operacji splotu, dzięki czemu matryce A[ ] i B[ ] są wolne i można w nich przechowywać nowe wartości.

Na zakończenie programu w bloku 13 (wiersze 540–670) filtr jest tak ustawiany, żeby miał w paśmie przepustowym wzmocnienie równe jedności. To odbywa się w sposób opisany poprzednio – ze wzorów 33.7 lub 33.8 jest obliczane istniejące wzmocnienie i przez tę wartość są dzielone wszystkie współczynniki „a" w celu ich znormalizowania. Zmienne pośrednie SA i SB są, odpowiednio, sumami współczynników „a" i „b".


```
100 'FILTR CZEBYSZEWA - OBLICZANIE WSPÓŁCZYNNIKÓW
110 '
120                             'INICJALIZACJA ZMIENNYCH
130 DIM A[22]                   'przechowuje współczynniki "a"
140 DIM B[22]                   'przechowuje współczynniki "b"
150 DIM TA[22]                  'zastosowanie wewnętrzne do łączenia stopni
160 DIM TB[22]                  'zastosowanie wewnętrzne do łączenia stopni
170 '
180 FOR I% = 0 TO 22
190   A[I%] = 0
200   B[I%] = 0
210 NEXT I%
220 '
230 A[2] = 1
240 B[2] = 1
250 PI = 3.14159265
260                             WPROWADZENIE PARAMETRÓW FILTRU
270 INPUT "Enter cutoff frequency (0 to .5): ", FC
280 INPUT "Enter 0 for LP,  1 for HP filter: ", LH
290 INPUT "Enter percent ripple     (0 to 29): ", PR
300 INPUT "Enter number of poles (2,4,...20):  ", NP
310 '
320 FOR P% = 1 TO NP/2          'PĘTLA DLA KAŻDEJ PARY BIEGUN-ZERO
330   '
340   GOSUB 1000                'Podprogram z rys. 33.8
350   '
360   FOR I% = 0 TO 22          'Dodaj współczynniki do kaskady
370    TA[I%] = A[I%]
380    TB[I%] = B[I%]
390   NEXT I%
400   '
410   FOR I% = 2 TO 22
420     A[I%]=A0*TA[I%] + A1*TA[I%-1] + A2*TA[I%-2]
430    B[I%] = TB[I%] - B1*TB[I%-1] - B2*TB[I%-2]
440   NEXT I%
450 '
460 NEXT P%
470 '
480 B[2] = 0                    'Zakończ łączenie współczynników
490 FOR I% = 0 TO 20
500   A[I%] = A[I%+2]
510   B[I%] = -B[I%+2]
520 NEXT I%
530 '
540 SA = 0                      'ZNORMALIZUJ WZMOCNIENIE
550 SB = 0
560 FOR I% = 0 TO 20
570   IF LH = 0 THEN SA   = SA + A[I%]
580   IF LH = 0 THEN SB   = SB + B[I%]
590    IF LH = 1 THEN SA = SA + A[I%] * (-1)^I%
600   IF LH= 1 THEN SB    = SB + B[I%] * (-1)^I%
610 NEXT I%
620 '
630 GAIN = SA/(1 - SB)
640 '
650 FOR I% = 0 TO 20
660   A[I%] = A[I%] / GAIN
670 NEXT I%
680 '                           'Końcowe wartości współczynników rekursywnych znajdują się
690 END                         'w A[ ] i B[ ]
```

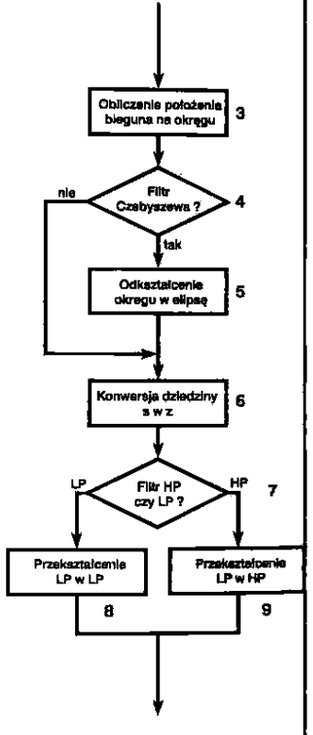


Rys. 33.7. Projektowanie filtru Czebyszewa-Butterwortha. Ten program przedstawiono już poprzednio w tablicach 20.4 i 20.5 w rozdziale 20. Na rys. 33.8 podano podprogram 1000 wywoływany z wiersza 340 tego programu głównego, a także sieć działań tego podprogramu

Obliczanie rozmieszczenia biegunów na płaszczyźnie *s*

Niezależnie od typu projektowanego filtru ten program zaczyna się od dolnoprzepustowego filtru Butterwortha na płaszczyźnie *s* z częstotliwością odcięcia ω = 1. Jak podano w poprzednim rozdziale, filtry Butterwortha mają bieguny równomiernie rozmieszczone wokół okręgu na płaszczyźnie *s*. Nie stosuje się żadnych zer, bo filtr jest dolnoprzepustowy. Promień tego okręgu jest równy *jeden*, co odpowiada częstotliwości odcięcia ω = 1. W bloku 3 w sieci działań (wiersze 1080 i 1090) są obliczane położenia każdej pary biegunów we współrzędnych prostokątnych. Zmienne programowe RP i IP to części rzeczywista i urojona położenia bieguna. Te zmienne programowe odpowiadają σ i ω, gdzie para biegunów ma położenie $\sigma \pm j\omega$. To położenie biegunów jest obliczone na podstawie liczby biegunów w filtrze (zmienna programowa NP) i w właśnie projektowanym stopniu (zmienna P%).

```
1000 'TEN PODPROGRAM JEST WYWOŁYWANY Z RYS. 33.7, WIERSZ 340
1010 '
1020 'Zmienne wchodzące do podprogramu:  PI, FC, LH, PR, HP, P%
1030 'Zmienne wychodzące z podprogramu:  A0, A1, A2, B1, B2
1040 'Zmienne stosowane wewnętrznie:     RP, IP, ES, VX, KX, T, W, M, D, K,
1050 '                                   X0, X1, X2, Y1, Y2
1060 '
1070 '          'Oblicz położenie bieguna na okręgu jednostkowym
1080 RP = -COS(PI(NP*2) + (P%-1) * PI/NP)
1090 IP =   SIN(PI/(NP*2) + (P%-1) * PI/NP)
1100 '
1110 '          'Odkształć okrąg w elipsę
1120 IF PR = 0 THEN GOTO 1210
1130 ES = SQR( (100 / (100-PR))^2 -1)
1140 VX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) + 1) )
1150 KX = (1/NP) * LOG( (1/ES) + SQR( (1/ES^2) - 1) )
1160 KX = (EXP(KX) + EXP(-KX))/2
1170 RP = RP * ( (EXP(VX) - EXP(-VX) ) /2 ) / KX
1180 IP = IP * ( (EXP(VX) + EXP(-VX) ) /2 ) / KX
1190 '
1200 '          'Konwersja z dziedziny s do dziedziny z
1210 T = 2*TAN(1/2)
1220 W = 2*PI*FC
1230 M = RP^2 + IP^2
1240 D = 4 - 4*RP*T + M*T^2
1250 X0 = T^2/D
1260 X1=2*T^2/D
1270 X2 = T^2/D
1280 Y1 = (8-2*M*T^2)/D
1290 Y2 = (-4 - 4"RPT - M*T^2)/D
1300 '
1310 '          'LP do LP lub LP do HP
1320 IF LH = 1 THEN K = -COS(W/2 + 1/2) / COS(W/2 - 1/2)
1330 IF LH = 0 THEN K = SIN(1/2 - W/2) / SIN(1/2 + W/2)
1340 D = 1 + Y1*K - Y2*K^2
1350 A0 = (X0 - X1*K + X2*K^2)/D
1360 A1 = (-2*X0*K + X1 +X1*K^2-2*X2*K)/D
1370 A2 = (X0*K^2 - X1 *K + X2)/D
1380 B1 = (2*K + Y1 + Y1*K^2 - 2*Y2*K)/D
1390 B2 = (-K^2 - Y1*K + Y2)/D
1400 IF LH = 1 THEN A1 = -A1
1410 IF LH = 1 THEN B1 = -B1
1420 '
1430 RETURN
```



Rys. 33.8. Podprogram wywoływany z rys. 33.7. Przez LP oznaczono filtr dolnoprzepustowy, a przez HP – górnoprzepustowy

Odkształcanie okręgu w elipsę

W celu implementacji filtru Czebyszewa to *kołowe* rozmieszczenie biegunów musi być przekształcone w rozmieszczenie *eliptyczne*. Względna płaskość elipsy określa, ile zafalowań będzie w paśmie przepustowym filtru. Jeśli położenie bieguna na okręgu oznaczymy jako σ i ω, to odpowiadające mu położenie na elipsie σ' i ω' jest określone wzorami:

$$\sigma' = \frac{\sigma \sinh(v)}{k}$$

$$\omega' = \frac{\omega \cosh(v)}{k}$$

gdzie:

$$v = \frac{\sinh^{-1}\left(\frac{1}{\epsilon}\right)}{NP}$$

$$k = \cosh\left(\frac{1}{NP}\cosh^{-1}\frac{1}{\epsilon}\right)$$

$$\epsilon = \left[\left(\frac{100}{100-PR}\right)^2 - 1\right]^{\frac{1}{2}}$$

Wzór 33.9. Przekształcenie okręgu w elipsę. Te wzory zamieniają położenie bieguna na okręgu w odpowiadające mu położenie na elipsie. Zmienna NP to liczba biegunów w filtrze, a PR – procentowa wartość zafalowań w paśmie przepustowym. Położenie na okręgu jest określone przez σ i ω, a na elipsie przez σ' i ω'. Zmienne ∈, v i k zastosowano tylko w celu skrócenia wzorów

W powyższych wzorach do opisywania elipsy stosuje się funkcje sinusa hiperbolicznego i kosinusa hiperbolicznego, podobnie jak zwykłe sinus i kosinus służą do opisu okręgu. Płaskość elipsy jest uzależniona od zmiennej *PR*, która liczbowo jest równa procentowej wartości zafalowań w paśmie przepustowym filtru. Zmienne ∈, v i *k* zastosowano, aby ograniczyć złożoność wzorów. Te zmienne są w programie reprezentowane, odpowiednio, przez *ES*, *VX* oraz *KX*. Powyższe wzory nie tylko przekształcają okrąg w elipsę, lecz także korygują położenia biegunów, tak aby zachować jednostkową częstotliwość odcięcia. Stosuje się niżej podane tożsamości, gdyż wiele języków programowania nie uwzględnia funkcji hiperbolicznych:

$$\sinh(x) = \frac{e^x - e^{-x}}{2}$$

$$\cosh(x) = \frac{e^x + e^{-x}}{2}$$

$$\sinh^{-1}(x) = \log_e\left[x + \left(x^2+1\right)^{\frac{1}{2}}\right]$$

$$\cosh^{-1}(x) = \log_e\left[x + \left(x^2-1\right)^{\frac{1}{2}}\right]$$

Te wzory dają niedozwolone operacje dla $PR \geq 30$ i $PR = 0$. Aby móc zastosować ten program do obliczania filtrów Butterwortha (tzn. zerowe zafalowania, $PR = 0$), należy pominąć te wiersze programu, gdzie występuje $PR = 0$ (wiersz 1120).

Konwersja sygnału ciągłego na dyskretny

Najczęściej stosowaną metodą konwersji rozmieszczenia biegunów i zer z dziedziny *s* do dziedziny *z* jest **przekształcenie biliniowe**. Jest to matematyczna metoda *odwzorowania konforemnego* (wiernokątnego), gdzie jedna płaszczyzna zespolona jest algebraicznie zniekształ-

cana lub odkształcana w inną płaszczyznę zespoloną. Przekształcenie biliniowe zamienia $H(s)$ w $H(z)$ przez podstawienie:

$$s \rightarrow \frac{2\left(1-z^{-1}\right)}{T\left(1+z^{-1}\right)}$$

Wzór 33.10. Przekształcenie biliniowe. To podstawienie odwzorowuje każdy punkt na płaszczyźnie s na odpowiadający mu punkt na płaszczyźnie z

Piszemy więc wzór na $H(s)$ i następnie zastępujemy każde s wyrażeniem z wzoru 33.10. W większości przypadków stosuje się $T = 2\text{tg}(1/2) = 1{,}093$. W wyniku uzyskuje się odwzorowanie zakresu częstotliwości od 0 do π radianów/sekundę w dziedzinie s na zakres od 0 do nieskończoności radianów w dziedzinie z. Nie wchodząc w szczegóły można powiedzieć, że przekształcenie biliniowe ma właściwości wymagane do konwersji z dziedziny s w dziedzinę z – takie jak np. odwzorowanie linii w okręgi. Oto przykład działania tego przekształcenia. Dla systemu ciągłego o pojedynczej parze biegunów o położeniach $p_1 = \sigma + j\omega$ oraz $p_2 = \sigma - j\omega$ funkcja przenoszenia w dziedzinie s jest dana wzorem:

$$H(s)=\frac{1}{(s-p_1)(s-p_2)}$$

Przekształcenie biliniowe zamienia tę funkcję w system dyskretny przez zastąpienie każdego s wyrażeniem z wzoru 33.10. Powstaje wówczas funkcja przenoszenia w dziedzinie z, też zawierająca dwa bieguny. Jest tu pewna trudność – w wyniku podstawienia uzyskuje się funkcję przenoszenia w postaci bardzo nieprzyjaznej:

$$H(z)=\frac{1}{\left(\frac{2\left(1-z^{-1}\right)}{T\left(1+z^{-1}\right)}-(\sigma+j\omega)\right)\left(\frac{2\left(1-z^{-1}\right)}{T\left(1+z^{-1}\right)}-(\sigma-j\omega)\right)}$$

Przeprowadzając długie i żmudne przekształcenia algebraiczne można powyższe wyrażenie doprowadzić do postaci takiej, jak we wzorze 33.3, a współczynniki rekursywne określić jako:

$$a_0=\frac{T^2}{D}$$

$$a_1=\frac{2T^2}{D}$$

$$a_2=\frac{T^2}{D}$$

$$b_1=\frac{\left(8-2MT^2\right)}{D}$$

$$b_2=\frac{\left(-4-4\sigma T-MT^2\right)}{D}$$

gdzie:

$$M=\sigma^2+\omega^2$$

$$T=2tg\left(\frac{1}{2}\right)$$

$$D=4-4\sigma T+MT^2$$

Wzór 33.11. Przekształcenie biliniowe dla dwóch biegunów. Para biegunów ma położenie σ ± ω na płaszczyźnie s, a a_0, a_1, a_2, b_1, b_2 są współczynnikami rekursywnymi dla systemu dyskretnego

Zmienne M, T i D nie mają żadnego znaczenia fizycznego; dzięki nim wzory stają się krótsze.

W wierszach 1200–1290 powyższe wzory są wykorzystane do konwersji położenia pary biegunów w dziedzinie s (przechowywanego w zmiennych RP i IP) bezpośrednio na współczynniki rekursywne przechowywane w zmiennych X0, X1, X2, Y1, Y2. Innymi słowy, obliczyliśmy wynik pośredni – współczynniki rekursywne dla jednego stopnia filtru *dolnoprzepustowego* o częstotliwości odcięcia *jeden*.

Zmiana częstotliwości odcięcia filtru dolnoprzepustowego

Zmiana częstotliwości filtru rekursywnego jest realizowana również metodą odwzorowania konforemnego. Przyjmijmy, że znamy funkcję przenoszenia dolnoprzepustowego filtru rekursywnego o częstotliwości odcięcia równej jeden. Funkcję przenoszenia podobnego filtru dolnoprzepustowego o nowej częstotliwości odcięcia W uzyskuje się, stosując odpowiednie przekształcenie, wykonywane też tylko przez *podstawienie zmiennych*, tak jak w przekształceniu biliniowym. Rozpoczynamy od zapisania funkcji przenoszenia filtru o częstotliwości odcięcia równej jeden i następnie zamiast każdego wyrazu z^{-1} podstawiamy następujące wyrażenie:

$$z^{-1} \rightarrow \frac{z^{-1}-k}{1-kz^{-1}}$$

gdzie:

$$k=\frac{\sin\left(\frac{1}{2}-\frac{W}{2}\right)}{\sin\left(\frac{1}{2}+\frac{W}{2}\right)}$$

Wzór 33.12. Zmiana częstotliwości odcięcia filtru dolnoprzepustowego. Jest to metoda zmiany częstotliwości odcięcia filtrów dolnoprzepustowych. Filtr oryginalny ma częstotliwość odcięcia równą jeden, a nowy filtr – częstotliwość odcięcia W – z zakresu od 0 do π

Wzór określa funkcję przenoszenia filtru o nowej częstotliwości odcięcia. Podane poniżej wzory projektowe są wynikiem zastosowania tego podstawienia do sekcji bikwadratowej, tzn. mającej nie więcej niż dwa bieguny i dwa zera:

$$a_0=\frac{\left(a_0-a_1k+a_2k^2\right)}{D}$$

$$a_1=\frac{\left(-2a_0k+a_1+a_1k^2-2a_2k\right)}{D}$$

$$a_2=\frac{\left(a_0k^2-a_1k+a_2\right)}{D}$$

$$b_1=\frac{\left(2k+b_1+b_1k^2-2b_2k\right)}{D}$$

$$b_2=\frac{\left(-k^2-b_1k+b_2\right)}{D}$$

gdzie:

$$D=1+b_1k-b_2k^2$$

$$k=\frac{\sin\left(\frac{1}{2}-\frac{W}{2}\right)}{\sin\left(\frac{1}{2}+\frac{W}{2}\right)}$$

Wzór 33.13. Konwersja filtru dolnoprzepustowego na filtr dolnoprzepustowy o innej częstotliwości odcięcia. Drukiem pochyłym zapisano współczynniki rekursywne filtru o częstotliwości równej jeden, a drukiem prostym – filtru o częstotliwości odcięcia W

Zmiana częstotliwości odcięcia filtru dolnoprzepustowego na górnoprzepustowy

Omówioną wyżej metodę zmiany częstotliwości można tak zmodyfikować, aby uzyskać zamianę odpowiedzi dolnoprzepustowej systemu na górnoprzepustową z jednoczesną zmianą częstotliwości odcięcia. Realizuje się to przez podstawienie:

$$z^{-1} \rightarrow \frac{-z^{-1}-k}{1+kz^{-1}}$$

gdzie:

$$k = -\frac{\cos\left(\frac{W}{2}+\frac{1}{2}\right)}{\cos\left(\frac{W}{2}-\frac{1}{2}\right)}$$

Wzór 33.14. Zmiana częstotliwości odcięcia filtru dolnoprzepustowego na górnoprzepustowy. To podstawienie zamienia filtr dolnoprzepustowy w górnoprzepustowy. Częstotliwość odcięcia filtru dolnoprzepustowego jest równa jeden, a filtru górnoprzepustowego W

Tak jak poprzednio, można wyprowadzić wzory projektowe dotyczące sekcji bikwadratowej. Okazuje się, że te wzory są identyczne z wzorami 33.14, z dwiema tylko zmianami. Wartość k jest inna (zgodnie ze wzorem 33.14), a dwa współczynniki a_1 i b_1 mają odwrotne znaki. Obliczenia według tych wzorów są wykonywane w wierszach od 1330 do 1410; w wyniku uzyskuje się częstotliwość odcięcia i możliwość wyboru odpowiedzi dolno- lub górnoprzepustowej.

To, co najlepsze i najgorsze w DSP

Pisząc tę książkę, przyjęto proste założenie: **większość metod DSP można zastosować i zrozumieć z użyciem minimalnego aparatu matematycznego**. Zamierzeniem autora jest dostarczenie naukowcom i inżynierom *narzędzi* do rozwiązywania problemów z dziedziny DSP występujących w ich pracach badawczych i projektowych.

W ostatnich czterech rozdziałach ukazano drugą stronę medalu, a więc takie metody DSP, które można zrozumieć, *tylko* korzystając z zaawansowanej matematyki. Jako przykład weźmy opisaną w tym rozdziale analizę filtru Czebyszewa-Butterwortha. Jest ona tym, co *najlepsze* w DSP – szeregiem eleganckich operacji matematycznych prowadzących do optymalnego rozwiązania. Istnieją też jednak *najgorsze* strony DSP – metody projektowania tak skomplikowane, że większość naukowców i inżynierów szuka jakiejś alternatywy.

A jakie jest twoje, Czytelniku, miejsce w tej problematyce DSP? To zależy od tego, *kim* jesteś i do *czego* chcesz zastosować DSP. Informacje zawarte w czterech ostatnich rozdziałach stanowią teoretyczne podstawy przetwarzania sygnałów. Jeśli masz zamiar *specjalizować się* w DSP, to musisz doskonale zrozumieć te podstawy matematyczne. Natomiast specjaliści z innych dziedzin nauki i techniki powinni tylko wiedzieć, jak się *stosuje* DSP, a nie muszą znać jego *uzasadnień* matematycznych. Dla tej grupy czytelników materiał teoretyczny jest bardziej tłem niż tematem głównym.

Słownik pojęć

Addytywność	*Additivity*	Właściwość matematyczna, którą musi mieć system, aby był liniowy. Jeśli sygnał wejściowy *a* daje na wyjściu sygnał *p*, a sygnał wejściowy *b* daje na wyjściu *q*, to sygnał wejściowy (*a* + *b*) daje na wyjściu (*p* + *q*).
Algorytm uczący się	*Learning algoritm*	Procedura stosowana w celu znalezienia zestawu wag sieci neuronowej na podstawie przykładów pokazujących, jak sieć powinna działać.
Aliasing	*Aliasing*	Proces, w którym sinusoida zmienia swoją częstotliwość na inną w wyniku próbkowania lub innych działań nieliniowych. Zwykle powoduje utratę informacji o sygnale. Inna definicja (przyp. tłum.): Niejednoznaczność sygnału w dziedzinie częstotliwości, która powstaje w wyniku nakładania się widma powielonego na widmo oryginalne w przypadku zbyt małej częstotliwości próbkowania (niespełniającej warunku Nyquista).
Aliasing w dziedzinie czasu	*Time domain aliasing*	Aliasing, który zachodzi w dziedzinie czasu, gdy operacje są wykonywane w dziedzinie częstotliwości. Przykładem jest splot kołowy.
Aliasing w dziedzinie częstotliwości	*Frequency domain aliasing*	Aliasing zachodzący w dziedzinie częstotliwości w wyniku działań podejmowanych w dziedzinie czasu. Przykładem jest aliasing podczas próbkowania.
Analiza widmowa	*Spectral analysis*	Analiza sygnału przez zbadanie amplitud, częstotliwości i faz jego sinusoid składowych. Podstawowym narzędziem analizy widmowej jest przekształcenie Fouriera.
Apertura próbkowania	*Sampling aperture*	Obszar w obrazie ciągłym, który jest wyrażany pojedynczym pikselem podczas przetwarzania na obraz cyfrowy. Na ogół ma w przybliżeniu taki sam rozmiar jak odstęp próbkowania.
Architektura harwardzka	*Harvard Architecture*	Wewnętrzna struktura komputera, w której program i dane są umieszczone w oddzielnych pamięciach, dostępnych przez oddzielne magistrale. Ta architektura jest powszechnie stosowana w mikroprocesorach używanych w DSP. Dla porównania proszę zobaczyć: *architektura von Neumanna.*
Architektura von Neumanna	*Von Neumann Architecture*	Wewnętrzna struktura komputera, w której zarówno program, jak i dane są umieszczone w tej samej pamięci; architektura bardzo często stosowana. Dla porównania proszę zobaczyć: *architektura harwardzka.*
ASCII	*ASCII*	Metoda wyrażania liter i liczb w postaci binarnej. Każdy znak ma przypisaną liczbę do 0 do 127. Metoda szeroko stosowana w technice komputerowej i w telekomunikacji.
Asembler	*Assembly*	Język programowania niskiego poziomu, który operuje bezpośrednio rejestrami i wewnętrznymi układami mikroprocesora. Dla porównania proszę zobaczyć: *język wysokiego poziomu.*
Autokorelacja	*Autocorrelation*	Skorelowanie sygnału z samym sobą. Jest pożyteczna, gdyż transformata Fouriera autokorelacji jest widmem mocy sygnału oryginalnego.
Barwa dźwięku	*Timbre*	Percepcja harmonicznych częstotliwości dźwięku przez słuch ludzki. Dla porównania proszę zobaczyć: *wysokość dźwięku.*
BASIC	*BASIC*	Język programowania wysokiego poziomu znany z powodu swej prostoty, lecz także z powodu licznych słabości. Większość programów w tej książce napisano w języku BASIC.
Bezpamięciowe	*Memoryless*	Systemy, w których bieżąca wartość sygnału wyjściowego zależy tylko od bieżącej wartości wejściowej, a nie od wartości poprzednich.
Biegun	*Pole*	Pojęcie stosowane w przekształceniu Laplace'a i w przekształceniu Z. Gdy funkcja przenoszenia w dziedzinie *s* lub w dziedzinie *z* jest zapisana jako iloraz dwóch wielomianów, to pierwiastki mianownika są *biegunami*, a licznika – *zerami*.

Bikwadratowy (system)	*Biquad*	System analogowy lub cyfrowy o dwóch biegunach i nie więcej niż dwóch zerach. Takie systemy są często łączone w kaskady w celu uzyskania filtrów spełniających bardziej skomplikowane wymogi.
Bitowe odwracanie adresów	*Bit reversal sorting*	Algorytm stosowany w FFT do uzyskania rozkładu z przeplotem. Realizowany przez zliczanie binarne z bitami odwróconymi lewo-prawostronnie.
Błąd kwantyzacji	*Quantization error*	Błąd wprowadzany przez kwantyzację sygnału. W większości przypadków maksymalnybłądkwantyzacjijestrówny ±½LSB,aodchyleniestandardowe $1/\sqrt{12}$ LSB. Nazywany także *szumem kwantyzacji.*
Błąd losowy	*Random error*	Błędy w pomiarach (lub w ich przewidywaniach), które nie powtarzają się przy kolejnych próbach. Dla porównania proszę zobaczyć: *błąd systematyczny.*
Błąd systematyczny	*Systematic error*	Błędy w pomiarach (lub w ich przewidywaniach), które powtarzają się przy kolejnych próbach. Dla porównania proszę zobaczyć: *błąd losowy.*
Błona podstawna	*Basiliar membrane*	Mały organ w uchu działający jak analizator widma. Powoduje, że specyficzne włókna w nerwie ślimakowym są pobudzane przez różne częstotliwości.
Bufor cykliczny	*Circular buffer*	Metoda gromadzenia danych stosowana w przetwarzaniu w czasie rzeczywistym; każdą nowo pobraną próbka zastępuje w pamięci próbkę najstarszą.
Całka dyskretna	*Discrete integral*	Operacja na sygnałach dyskretnych analogiczna z całką dla sygnałów ciągłych. Lepsza nazwa: *suma bieżąca.*
Całka splotowa	*Convolution integral*	Wzór matematyczny definiujący splot w systemach ciągłych; jest analogią sumy splotowej w systemach dyskretnych.
Całkowity sygnał wizyjny	*Composite video*	Analogowy sygnał telewizyjny zawierający impulsy synchronizacji rozdzielające półobrazy lub ramki.
Centralne twierdzenie graniczne	*Central Limit Theorem*	Ważne twierdzenie w statystyce. W jednej ze swych postaci brzmi ono: suma wielu liczb losowych daje gaussowską funkcję gęstości prawdopodobieństwa niezależnie od funkcji gęstości prawdopodobieństwa poszczególnych liczb losowych.
Cepstrum	*Cepstrum*	Słowo powstałe przez przestawienie liter w słowie *spectrum*. Nazwa stosowana w przetwarzaniu homomorficznym do opisania widma przy zamianie dziedzin czasu i częstotliwości.
Ciąg impulsów	*Impulse train*	Sygnał zawierający szereg impulsów rozmieszczonych w jednakowych odstępach.
Cooley i Tukey	*Cooley and Tukey*	J.W. Cooleyowi i J.W. Tukeyowi przypisuje się zasługę wprowadzenia do nauki i techniki szybkiego przekształcenia Fouriera (FFT), w artykule opublikowanym przez nich w roku 1965.
Częstotliwości ujemne	*Negative frequencies*	Sinusoidy można zapisywać jako przebiegi o częstotliwościach dodatnich: $\cos(\omega t)$ lub ujemnych: $\cos(-\omega t)$. Częstotliwości ujemne są uwzględnione w zespolonym przekształceniu Fouriera, rozszerzając jego możliwości.
Częstotliwość Nyquista	*Nyquist frequency, Nyquist rate*	To pojęcie odnosi się do twierdzenia o próbkowaniu, lecz jest w różny sposób stosowane przez różnych autorów. Może oznaczać cztery różne wielkości: największą częstotliwość zawartą w sygnale, podwojenie tej częstotliwości, częstotliwość próbkowania lub połowę tej częstotliwości.
Częstotliwość odcięcia	*Cutoff frequency*	W filtrach analogowych i cyfrowych częstotliwość rozdzielająca pasmo przepustowe od zaporowego. Zwykle mierzona w punkcie charakterystyki, gdzie amplituda maleje do 0,707 (o –3 dB). Nazywana też częstotliwością graniczną.
Częstotliwość odcięcia –3 dB	*–3 dB cutoff frequency*	Granica między pasmem przepustowym a pasmem zaporowym filtru. Częstotliwość odcięcia jest określana przez punkt –3 dB (0,707) spadku charakterystyki amplitudowej w stosunku do wartości maksymalnej w paśmie przepustowym.
Częstotliwość podstawowa	*Fundamental frequency*	Częstotliwość, z jaką powtarza się przebieg okresowy. Dla porównania proszę zobaczyć: *częstotliwości harmoniczne.*
Część rzeczywista	*Real part*	Składnik liczby urojonej, który występuje bez czynnika j, np. 3 w liczbie $3 + 2j$. W rzeczywistym przekształceniu Fouriera część rzeczywista odnosi się do tej części dziedziny częstotliwości, która zawiera amplitudy fal kosinusoidalnych, nawet gdy nie występuje żaden czynnik j.

Część urojona	*Imaginary part*	Składnik liczby urojonej występujący z czynnikiem j, np. 2 w liczbie $3 + 2j$. W rzeczywistym przekształceniu Fouriera część urojona odnosi się również do tej części dziedziny częstotliwości, która zawiera amplitudy fal sinusoidalnych, nawet jeśli wyrazy j nie są stosowane.
Decybel	*Decibel SPL*	Skala logarytmiczna stosowana m.in. do wyrażania natężenia fali dźwiękowej. Dźwięk o poziomie 0 dB jest prawie niesłyszalny, 60 dB – to normalna mowa, a 140 dB powoduje uszkodzenie ucha. W terminologii angielskiej – decibel SPL (*sound power level*).
Decymacja	*Decimation*	Zredukowanie częstotliwości próbkowania sygnału przetworzonego na cyfrowy. Na ogół wymaga filtracji dolnoprzepustowej, po której następuje odrzucenie niektórych próbek. Dla porównania proszę zobaczyć: *interpolacja.*
Delta-sigma	*Delta-sigma*	Metoda przetwarzania a/c popularna w przetwarzaniu sygnałów głosu i muzyki. W tej metodzie stosuje się bardzo dużą częstotliwość próbkowania, ale próbki są tylko jednobitowe. Po próbkowaniu przeprowadza się decymację.
Dithering	*Dithering*	Dodawanie szumu do sygnału analogowego przed przetwarzaniem a/c, aby zapobiec podczas przetwarzania utrzymywaniu się sygnału na tym samym poziomie kwantyzacji. Dosłownie: dygotanie.
Dokładność podwójna	*Double precision*	Standard zapisu zmiennoprzecinkowego, w którym do wyrażenia każdej liczby stosuje się 64 bity. Dla porównania proszę zobaczyć: *dokładność pojedyncza.*
Dylatacja	*Dilation*	Operacja morfologiczna. Gdy jest zastosowana do obrazów binarnych, powiększa obiekty i może w jedną całość łączyć obiekty rozłączone.
Dyskretne przekształcenie Fouriera (DFT)	*Discrete Fourier Transform (DFT)*	Jedno z przekształceń z rodziny fourierowskiej operujące *dyskretnymi i okresowymi* sygnałami w dziedzinie czasu.
Dyskretne przekształcenie kosinusowe (DCT)	*Discrete cosine transform (DCT)*	Jest pokrewne z przekształceniem Fouriera. Daje rozkład sygnału na fale kosinusoidalne. Stosowane w kompresji danych.
Dyskretne w czasie przekształcenie Fouriera (DTFT)	*Discrete time Fourier transform (DTFT)*	Jedno z przekształceń z rodziny fourierowskiej operujące *dyskretnymi i nieokresowymi* sygnałami w dziedzinie czasu.
Dziedzina	*Domain*	Niezależna zmienna sygnału. Na przykład napięcie zmieniające się w funkcji czasu należy do *dziedziny czasu*. Inne znane dziedziny to *dziedzina przestrzeni* (np. obrazy) i *dziedzina częstotliwości* (np. transformata uzyskana w wyniku przekształcenia Fouriera).
Dziedzina czasu	*Time domain*	Należą do niej sygnały, dla których zmienną niezależną jest czas. Stosowana również jako ogólne odniesienie do każdej dziedziny, w której są zbierane dane.
Dziedzina częstotliwości	*Frequency domain*	Należą do niej sygnały, dla których zmienną niezależną jest częstotliwość, na przykład – transformata Fouriera.
Dziedzina przestrzeni	*Spatial domain*	Należą do niej sygnały, dla których zmienną niezależną jest odległość (przestrzeń). Obrazy są sygnałami w dziedzinie przestrzeni.
Dziedzina s	*s-domain*	Dziedzina zdefiniowana przekształceniem Laplace'a. Zwana też płaszczyzną s.
Dziedzina z	*z-domain*	Dziedzina zdefiniowana przekształceniem Z.
Efekt Gibbsa	*Gibbs effect*	Gdy sygnał jest obcinany w jednej dziedzinie, to w innej pojawiają się oscylacje i przerzuty na krańcach i w narożnikach.
Efekty krańcowe	*End effects*	Nieprawidłowe krańce odfiltrowanego sygnału wyjściowego spowodowane tym, że odpowiedź impulsowa filtru nie jest w pełni zanurzona w sygnale wejściowym.
Erozja	*Erosion*	Operacja morfologiczna. Gdy jest zastosowana do obrazów binarnych, powoduje zmniejszanie obiektów i może rozdzielać obiekty na dwie lub więcej części.
Eulera zależność	*Euler's relation*	Najważniejsze równanie w matematyce liczb zespolonych określające zależność między falami sinusoidalną i kosinusoidalną a zespolonymi funkcjami wykładniczymi.

Fala nośna	*Carrier wave*	Nazwa związana z modulacją amplitudy sygnałów radiowych. Dotyczy fali sinusoidalnej w.cz., która jest łączona z sygnałem m.cz. niosącym informację.
Filtracja dopasowana	*Matched filtering*	Metoda stosowana do określania, czy i gdzie w sygnale występuje znany wzorzec. Podstawą filtracji dopasowanej jest korelacja, lecz implementowana przez splot.
Filtracja dwukierunkowa	*Bidirectional filtering*	Metod rekursywna stosowana do realizacji filtrów zerofazowych. Sygnał jest najpierw filtrowany według kolejności próbek od lewej ku prawej, a potem uzyskany sygnał pośredni jest filtrowany w kolejności od prawej ku lewej stronie.
Filtr antyaliasingowy	*Antialias filter*	Dolnoprzepustowy filtr analogowy umieszczony przed przetwornikiem a/c. Usuwa składowe sygnału o częstotliwości większej niż połowa częstotliwości próbkowania, które podczas przetwarzania wywołałyby zjawisko aliasingu.
Filtr Bessela	*Bessel filter*	Filtr analogowy zoptymalizowany pod względem liniowości fazy. Jego odpowiedź skokowa prawie wcale nie ma przerzutu, a zbocza narastające i opadające są do siebie podobne. Stosowany do wygładzania sygnałów w dziedzinie czasu.
Filtr Butterwortha	*Butterworth filter*	Rozdziela dwa pasma częstotliwości; ma najszybsze opadanie charakterystyki z utrzymaniem jej płaskiego przebiegu w paśmie przepustowym. Może być analogowy lub cyfrowy. Proszę też zobaczyć: *filtr o maksymalnie płaskiej charakterystyce.*
Filtr Czebyszewa	*Chebyshev filter*	Rozdziela dwa pasma częstotliwości. Daje szybsze opadanie charakterystyki niż filtr Butterwortha, lecz dopuszcza do zafalowań na charakterystyce w paśmie przepustowym. Może być analogowy lub cyfrowy.
Filtr eliptyczny	*Elliptic filter*	Rozdziela dwa pasma częstotliwości. Uzyskuje się w nim strome opadanie charakterystyki na granicy pasm, dopuszczając zafalowania na charakterystyce w pasmach przepustowym i zaporowym.
Filtr odtwarzający	*Reconstruction filter*	Analogowy filtr dolnoprzepustowy umieszczony za przetwornikiem cyfrowo-analogowym. Wygładza przebieg schodkowy, usuwając składowe o częstotliwościach większych od połowy częstotliwości próbkowania.
Filtr okienkowany funkcją *sinc*	*Windowed-sinc*	Filtr cyfrowy służący do oddzielania jednego pasma częstotliwości od drugiego.
Filtr optymalny	*Optimal filter*	Filtr, który jest „najlepszy" w pewien specyficzny sposób. Na przykład filtry Wienera dają optymalny stosunek sygnału do szumu, a filtry dopasowane są optymalne do wykrywania celu.
Filtrowana projekcja wsteczna	*Filtered backprojection*	Metoda stosowana w tomografii komputerowej do odtwarzania obrazu na podstawie jego widoków. Widoki są filtrowane i następnie rzutowane wstecznie.
Filtr parzystego rzędu	*Even order filter*	Filtr analogowy i cyfrowy o parzystej liczbie biegunów.
Filtr rzędu nieparzystego	*Odd order filter*	Filtr analogowy lub cyfrowy mający nieparzystą liczbę biegunów.
Filtr o średniej kroczącej	*Moving average filter*	W takim filtrze każda próbka w sygnale wyjściowym jest średnią wielu próbek wejściowych. Może być realizowany przez splot lub rekursję.
Filtr Wienera	*Wiener filter*	Filtr optymalny pod względem zwiększenia stosunku sygnału do szumu na podstawie widm częstotliwościowych sygnału i szumu.
Filtr z przełączanymi pojemnościami	*Switched capacitor filter*	Filtr analogowy, w którym zamiast rezystorów zastosowano szybkie przełączanie. Te filtry są realizowane jako łatwe w zastosowaniu układy scalone. Często używane jako filtry antyaliasingowe w przetwornikach a/c i filtry odtwarzające w przetwornikach c/a.
Frame grabber (stosuje się nazwę z terminologii angielskiej)	*Frame grabber*	Przetwornik analogowo-cyfrowy stosowany do zmiany na postać cyfrową i do rejestracji obrazów (ramek) z sygnału telewizyjnego.

Funkcja delta	*Delta function*	Znormalizowany impuls. Dyskretna funkcja delta jest sygnałem złożonym z samych zer, prócz próbki w zerze, która ma wartość jeden. Funkcja delta dla sygnałów ciągłych jest podobna, lecz ma charakter bardziej abstrakcyjny.
Funkcja masy prawdopodobieństwa	*Probability mass function (PMF)*	Wyraża prawdopodobieństwo, że zmienna *dyskretna* przyjmie określoną wartość. Dla porównania proszę zobaczyć: *funkcja gęstości prawdopodobieństwa.*
Funkcja przenoszenia	*Transfer function*	Stosunek sygnału wyjściowego do wejściowego. Może być stosowana w różnych postaciach, w zależności od tego, jak reprezentowane są sygnały. Na przykład w dziedzinach *s* i *z* ta funkcja jest ilorazem dwóch wielomianów i może być wyrażona jako *bieguny i zera.*
Funkcja przenoszenia modulacji (MTF)	*Modulation transfer function (MTF)*	Inna nazwa *odpowiedzi częstotliwościowej*, stosowana w terminologii techniki obrazowej.
Funkcja rozkładu (lub gęstości) prawdopodobieństwa (pdf)	*Probability distribution (or density) function (pdf)*	Wyraża prawdopodobieństwo, że zmienna ciągła przyjmie określoną wartość. Dla porównania proszę zobaczyć: *funkcja masy prawdopodobieństwa.*
Funkcja rozmycia linii (LSF)	*Line spread function (LSF)*	Odpowiedź systemu na cienką linię na obrazie wejściowym.
Funkcja rozmycia punktu (PSF)	*Point spread function (PSF)*	Nazwa odpowiedzi impulsowej stosowana w terminologii przetwarzania obrazów.
Funkcja *sinc*	*Sinc function*	Formalnie definiowana zależnością: $\mathrm{sinc}(a) = \sin(\pi a)/\pi a$. Stała π jest często ukryta w jednej ze zmiennych, co daje postać ogólną: $\sin(x)/x$. Ta funkcja jest ważna, gdyż jest transformatą Fouriera impulsu prostokątnego.
Funkcje bazowe	*Basis functions*	Zestaw przebiegów stosowanych do rozkładu sygnału. Na przykład funkcjami bazowymi dla rozkładu Fouriera są fale sinusoidalne i kosinusoidalne o amplitudach równych jedności.
GIF	*GIF (graphics interchange format)*	Często stosowany format pliku obrazowego, w którym stosuje się bezstratną kompresję LZW. Szeroko stosowany w grafice w ogólnoświatowej sieci WWW. Dla porównania proszę zobaczyć: *TIFF i JPEG.*
Głoski bezdźwięczne (np. trące)	*Fricative*	Dźwięk w mowie ludzkiej pochodzący z losowego szumu powstającego w wyniku turbulencji powietrza, np. *s, f, sz*, z w angielskim także np. *th*. Dla porównania proszę zobaczyć: głoski dźwięczne.
Głoski dźwięczne	*Voiced*	Dźwięki w mowie ludzkiej powstające w wyniku wypychania powietrza z płuc przez struny głosowe i dalej przez nos i/lub usta. Przykładem głosek dźwięcznych są samogłoski. Dla porównania proszę zobaczyć: *głoski bezdźwięczne (trące).*
Harmoniczne	*Harmonics*	Składowe częstotliwości sygnału okresowego, będące zawsze całkowitymi wielokrotnościami częstotliwości podstawowej. Częstotliwość podstawowa jest pierwszą harmoniczną, dwukrotność podstawowej – drugą harmoniczną itd.
Hiperprzestrzeń	*Hyperspace*	Pojęcie stosowane w metodzie wykrywania celu i w analizie sieci neuronowych. Jeden parametr może być graficznie interpretowany jako *linia*, dwa – jako *płaszczyzna*, trzy – jako *przestrzeń*, a więcej niż trzy parametry – jako *hiperprzestrzeń.*
Histogram	*Histogram*	Obrazuje rozkład wartości sygnału. Na osi x są przedstawione możliwe wartości próbek, a na osi y – liczby próbek mających daną wartość.
Histogram przedziałowy	*Binned histogram*	Histogram tworzony w przypadkach, gdy dane (lub sygnał) mają wiele poziomów kwantyzacji, jak np. liczby zmiennoprzecinkowe.
Homogeniczność	*Homogeneity*	Matematyczna właściwość wszystkich systemów liniowych. Jeśli sygnał wejściowy $x[n]$ daje na wyjściu $y[n]$, to sygnał wejściowy $kx[n]$ daje na wyjściu $ky[n]$, dla każdej wartości stałej k.
Homomorficzne przetwarzanie	*Homomorphic*	Metoda DSP służąca do rozdzielania sygnałów połączonych ze sobą w sposób nieliniowy np. przez mnożenie lub splot. Zadanie nieliniowe jest zamieniane w liniowe przez odpowiednie przekształcenie.

Impuls	*Impulse*	Sygnał złożony z samych zer, prócz bardzo krótkiego piku. W przypadku sygnałów dyskretnych ten pik obejmuje pojedynczą próbkę niezerową. W przypadku sygnałów ciągłych szerokość impulsu musi być znacznie mniejsza niż odpowiedzi wszystkich systemów, w których ten sygnał może być stosowany.
Impuls jednostkowy	*Unit impulse*	Inna nazwa *funkcji delta*.
Interpolacja	*Interpolation*	Zwiększanie częstotliwości próbkowania sygnału przetwarzanego na cyfrowy. Zwykle wykonywana przez wstawienie zer między próbkami oryginalnymi i zastosowanie filtru dolnoprzepustowego. Dla porównania proszę zobaczyć: *decymacja*.
Inwersja widmowa	*Spectral inversion*	Metoda zmiany odpowiedzi impulsowej filtru w taki sposób, że odpowiadająca jej odpowiedź częstotliwościowa ulega odwróceniu góra–dół. W ten sposób można zmienić filtr dolnoprzepustowy w górnoprzepustowy, pasmowoprzepustowy w pasmowozaporowy itd.
Iteracja	*Iterative*	Metoda dochodzenia do rozwiązania przez stopniowy dobór zmiennych we właściwym kierunku aż do uzyskania zbieżności. Stosowana w rekonstrukcji obrazów w tomografii komputerowej oraz w sieciach neuronowych.
Jaskrawość (luminancja)	*Brightness*	Wielkość fotometryczna charakteryzująca świecenie ciał (źródeł światła lub powierzchni oświetlonych) w danym kierunku. Dla porównania proszę zobaczyć: *kontrast*.
Jądro filtru	*Filter kernel*	Czasem używana nazwa odpowiedzi impulsowej filtru implementowanego przez splot. Znana też jako *odpowiedź splotowa*.
Jednobiegunowe filtry cyfrowe	*Single-pole digital filters*	Proste filtry rekursywne będące naśladownictwem elektronicznych dolno- i górnoprzepustowych filtrów RC.
Język C	*C*	Popularny język programowania stosowany w nauce, technice i w DSP. Jego bardziej zaawansowana wersja ma nazwę C++.
Język wysokiego poziomu	*High-level language*	Języki programowania, takie jak C, BASIC i FORTRAN.
JPEG	*JPEG*	Często stosowany format pliku obrazowego z zastosowaniem kompresji (stratnej). Dla porównania proszę zobaczyć: *GIF i TIFF*.
Kaskada	*Cascade*	Połączenie dwóch lub więcej stopni, w którym wyjście jednego stopnie staje się wejściem następnego.
Klasyfikatory	*Classifiers*	Parametry reprezentujące duży zestaw danych i z niego wyodrębnione. Na przykład – rozmiar obszaru, amplituda piku, stromość zbocza itd. Stosowane w rozpoznawaniu obrazów.
Kodowanie ciągów	*Run-length encoding*	Prosta metoda kompresji danych, mająca wiele odmian. Znaki, które wielokrotnie kolejno się powtarzają, zastępuje się słowami kodowymi zawierającymi informację, jaki to jest znak i jak długi jest jego ciąg.
Kodowanie delta	*Delta encoding*	Szerokie pojęcie obejmujące metody, w których zbierane dane są różnicami między sąsiednimi próbkami. Stosowane w przetwarzaniu a/c, w kompresji danych i w wielu innych aplikacjach.
Kodowanie Huffmana	*Huffman encoding*	Metoda kompresji danych, w której często występującym znakom przydziela się mniej bitów niż znakom pojawiającym się rzadko.
Kodowanie RGB	*RGB encoding*	Wyrażanie obrazu barwnego przez wyspecyfikowanie ilości czerwieni, zieleni i koloru niebieskiego w każdym pikselu.
Kodowanie w dziedzinie czasu	*Time domain encoding*	Jeden z dwóch głównych sposobów, w jaki można zakodować informację w sygnale. Informacja jest zawarta w kształcie przebiegu. Dla porównania proszę zobaczyć: *kodowanie w dziedzinie częstotliwości*.
Kodowanie w dziedzinie częstotliwości	*Frequency domain encoding*	Jeden z dwóch głównych sposobów, w jaki można zakodować informację w sygnale. Informacja jest zawarta w amplitudzie, częstotliwości i fazie sinusoidalnych składowych sygnału. Najlepszym przykładem są sygnały foniczne (audio). Dla porównania proszę zobaczyć: *kodowanie w dziedzinie czasu*.

Kod źródłowy	*Source code*	Program komputerowy w postaci napisanej przez programistę; w odróżnieniu od *programu wykonywalnego* – postaci programu nadającej się do bezpośredniego wykonywania w komputerze.
Kolejność głównego rzędu	*Row major order*	Kolejność stosowana do konwersji obrazu do postaci szeregowej.
Kołowość	*Circularity*	Wrażenie, że koniec sygnału jest połączony z jego początkiem. Wrażenie takie powstaje, gdy rozważa się tylko pojedynczy okres sygnału okresowego.
Kompansja	*Companding*	Nieliniowość o kształcie litery „S" umożliwiająca 8-bitową digitalizację sygnałów głosowych zamiast 12-bitowej. W Europie stosuje się zasadę „A", w Stanach Zjednoczonych zaś zasadę „μ".
Kompresja bezstratna	*Lossless compression*	Metody kompresji danych dające dokładne odtworzenie danych oryginalnych, np. kompresja LZW.
Kompresja stratna	*Lossy compression*	Metody kompresji danych dające tylko przybliżone odtworzenie danych oryginalnych. W ten sposób można uzyskać większe współczynniki kompresji. Przykładem jest *JPEG*.
Kompresja transformatowa	*Transform compression*	Metoda kompresji danych polegająca na przydzielaniu mniejszej liczby bitów składowym o wyższych częstotliwościach. Najlepszym przykładem jest *JPEG*.
Kontrast	*Contrast*	Różnica między jaskrawościami obiektu i tła. Dla porównania proszę zobaczyć: *jaskrawość*.
Korelacja	*Correlation*	Operacja matematyczna realizowana tak samo, jak splot, lecz z lewo-prawostronnym odwróceniem jednego sygnału. Jest to optymalny sposób wykrycia znanego przebiegu w sygnale.
Korelacja wzajemna	*Cross-correlation*	Sygnał utworzony w wyniku korelacji jednego sygnału z innym. Maksima na tym sygnale wskazują na podobieństwa między sygnałami oryginalnymi. Proszę też zobaczyć: *autokorelacja*.
Krzywa gamma	*Gamma curve*	Funkcja matematyczna lub tablica przeglądowa zawierająca zarejestrowaną wartość piksela i jaskrawość, z jaką pojawia się on w wyświetlanym obrazie. Jest też nazywana *przekształceniem skali szarości*.
Krzywa gaussowska	*Gaussian*	Krzywa o kształcie dzwonowym, w ogólnej postaci e^{x^2}. Krzywa gaussowska ma wiele wyjątkowych właściwości. Zwana też *rozkładem normalnym*.
Krzywa ROC	*ROC (receiver operating characteristic) curve*	Wykres obrazujący, jak wybór progu wpływa na rozwiązywanie problemu wykrywania celu.
Liczba zespolona sprzężona	*Complex conjugation*	Dwie liczby zespolone nazywa się sprzężonymi, jeśli mają jednakowe części rzeczywiste, a części urojone różnią się tylko znakiem. Liczba sprzężona jest często oznaczana gwiazdką umieszczoną za zmienną. Przykład: jeśli $A = 3 + 2j$, to $A^* = 3 - 2j$.
Liczba zespolona w postaci wykładniczej	*Complex exponential*	Liczba zespolona w postaci: e^{a+bj}. Taka postać jest dogodna w nauce i technice, gdyż dzięki zależności Eulera umożliwia wyrażanie sinusoid.
Liczby całkowite	*Integers*	Liczby ... –2, –1, 0, 1, 2, ... To określenie dotyczy też liczb zarejestrowanych przy użyciu zapisu stałoprzecinkowego. Dla porównania proszę zobaczyć: *zapis zmiennoprzecinkowy*.
Liczby zespolone	*Complex numbers*	Liczby w postaci $z = a + jb$, gdzie a i b są liczbami rzeczywistymi, a j tzw. jednostką urojoną o właściwości: $j = \sqrt{-1}$. Liczbę a nazywa się częścią rzeczywistą liczby zespolonej, a liczbę b – *jej częścią urojoną*. Przykład: $3 + 2j$.
Liniowa faza	*Linear phase*	Właściwość systemu, w którym charakterystyka fazy jest linią prostą. Na ogół jest to istotne, gdyż oznacza, że odpowiedź impulsowa ma symetrię lewo-prawostronną, dzięki czemu zbocza narastające w sygnale wyjściowym wyglądają tak samo, jak opadające.
Liniowość statyczna	*Static linearity*	Dotyczy działania systemu wówczas, gdy sygnały nie zmieniają się (tzn. są to sygnały stałoprądowe DC lub *statyczne*).W tym przypadku sygnał wyjściowy jest równy sygnałowi wejściowemu pomnożonemu przez pewną stałą.

Łączność operacji splotu	*Associative property of convolution*	Właściwość zapisywana jako: $(a[n] * b[n]) * c[n] = a[n] * (b[n] * c[n])$. Ta cecha jest istotna w przetwarzaniu sygnałów, gdyż opisuje zachowanie się stopni łączonych w kaskadę.
Metoda symboliczna (przekształcenie fazorowe)	*Phasor transform*	Metoda, w której liczby zespolone stosuje się do wyznaczania odpowiedzi częstotliwościowej obwodów *RLC*. Impedancje rezystorów, kondensatorów i cewek indukcyjnych są, odpowiednio, wyrażane jako: R, $1/j\omega C$ oraz $j\omega L$. W polskiej terminologii używa się raczej nazwy: *metoda symboliczna*.
MFLOPS	*MFLOPS*	*Million–Floating-Point-Operations-Per-Second* – milionów operacji zmiennoprzecinkowych na sekundę – często stosowany sposób wyrażania szybkości działania komputera. Dla porównania proszę zobaczyć: *MIPS*.
Mikroprocesor DSP	*DSP microprocessor*	Rodzaj mikroprocesora zaprojektowanego do szybkich operacji matematycznych. Często ma architekturę potokową (*pipeline*) i/lub harwardzką.
MIPS	*MIPS*	*Million-Instructions Per-Second* – milionów instrukcji na sekundę – często stosowany sposób wyrażania szybkości działania komputera. Dla porównania proszę zobaczyć: *MFLOPS*.
Modulacja amplitudy	*Amplitude modulation*	Metoda stosowana w łączności radiowej do łączenia sygnału niosącego informację (taką jak np. sygnał audio) z falą nośną. Zwykle realizowana przez mnożenie dwóch sygnałów.
Modulacja CVSD	*CVSD*	*Continuosly variable slope delta* – modulacja delta ze zboczem zmiennym w sposób ciągły, technika stosowana do przetwarzania sygnału głosowego na ciągły strumień binarny.
Morfing	*Morphing*	Stopniowe przechodzenie jednego obrazu w inny. Technika stosowana do efektów specjalnych, takich jak człowiek przekształcający się w wilka.
Morfologiczny	*Morphological*	Pojęcie odnoszące się zwykle do prostych operacji nieliniowych wykonywanych na obrazach binarnych, np. erozja i dylatacja.
Motylek	*Butterfly*	Podstawowy element obliczeniowy w FFT. Zamienia dwie liczby zespolone na dwie inne liczby zespolone.
MPEG	*MPEG*	Standard kompresji sygnału wizyjnego, np. w telewizji cyfrowej.
Multipleksowanie	*Multiplexing*	Łączenie razem ze sobą dwóch lub więcej sygnałów w celu ich przesłania. Może być realizowane wieloma różnymi sposobami.
Multipleksowanie w dziedzinie częstotliwości	*Frequency domain multiplexing*	Metoda łączenia sygnałów w celu ich jednoczesnego przesyłania. Polega na przesuwaniu sygnałów do różnych obszarów widma częstotliwości.
Najszybszy spadek	*Steepest descent*	Metodyka stosowana w projektowaniu algorytmów iteracyjnych. Analogiczna do znajdowania dna doliny przez poruszanie się stale w kierunku w dół.
Nakładanie z dodawaniem (raczej stosuje się nazwę angielską: overlap add))	*Overlap add*	Metoda stosowana do dzielenia długich sygnałów na mniejsze segmenty w celu ich przetwarzania.
Nerw ślimakowy	*Cochlear nerve*	Nerw przekazujący informację foniczną z ucha do mózgu.
Nieskończona odpowiedź impulsowa (NOI)	*Infinite impulse response (IIR)*	Odpowiedź impulsowa zawierająca nieskończoną liczbę wartości niezerowych, taka jak opadająca funkcja wykładnicza. Często stosowana do wykazania, że filtr jest realizowany metodą rekursywną, a nie przy użyciu splotu.
Niezmienność przesunięcia	*Shift invariance*	Właściwość wielu systemów. Przesunięcie w sygnale wejściowym nie powoduje w sygnale wyjściowym niczego innego poza przesunięciem. Oznacza, że właściwości systemu nie zmieniają się w funkcji czasu (lub jakiejkolwiek innej zmiennej niezależnej).
NTSC	*NTSC*	Standard telewizyjny stosowany w USA, Japonii i niektórych innych krajach. Dla porównania proszę zobaczyć: *PAL* i *SECAM*.

Obrazowanie rastrowe	*Halftone*	Powszechnie stosowany sposób drukowania obrazów na papierze. Różne odcienie szarości są uzyskiwane przez różne rozmieszczenia małych czarnych kropek. W obrazach kolorowych stosuje się kropki czerwone, zielone i niebieskie.
Obrazowanie w skali szarości	*Grayscale: image*	Obraz cyfrowy, w którym piksele są wyświetlane w różnych odcieniach szarości – od czarnego do białego.
Obszar zbieżności	*Region-of-convergence*	Pojęcie stosowane w przekształceniach Z i Laplace'a. Są to te obszary płaszczyzny s i z, które mają określoną wartość.
Odchylenie standardowe	*Standard deviation*	Sposób wyrażenia fluktuacji sygnału wokół jego średniej wartości. Definiowany jako pierwiastek kwadratowy ze średniej odchyleń podniesionych do kwadratu, przy czym odchylenie jest różnicą między wartością próbki a wartością średnią. Dla porównania proszę zobaczyć: wartość średniokwadratowa *rms*.
Odpowiedź częstotliwościowa	*Frequency response*	Zmiany modułu i fazy, jakim ulegają przebiegi sinusoidalne, przechodząc przez system liniowy. Zwykle wyrażana w funkcji częstotliwości. Często wyznaczana przez znajdowanie transformaty Fouriera odpowiedzi impulsowej.
Odpowiedź impulsowa	*Impulse response*	Sygnał wyjściowy systemu, gdy na wejściu jest impuls znormalizowany (funkcja delta).
Odpowiedź krawędziowa	*Edge response*	W przetwarzaniu obrazów sygnał wyjściowy systemu, gdy sygnałem wejściowym jest zbocze. Ostrość odpowiedzi krawędziowej stosuje się często jako miarę rozdzielczości systemu.
Odpowiedź na impuls prostokątny	*Pulse response*	Sygnał wyjściowy systemu, gdy na wejściu jest impuls prostokątny.
Odpowiedź skokowa	*Step response*	Odpowiedź na wyjściu sygnału, gdy na jego wejściu jest sygnał o kształcie funkcji skokowej.
Odtwarzanie sygnału	*Signal restoration*	Przywrócenie sygnału do jego oryginalnej postaci po tym, jak uległ jakimś zmianom lub uszkodzeniom. Jedno z głównych zastosowań *filtracji*.
Odwrotna inwersja widmowa	*Spectral reversal*	Metoda zmiany odpowiedzi impulsowej w taki sposób, że równoważna jej odpowiedź częstotliwościowa ulega odwróceniu lewo-prawostronnemu. W ten sposób filtr dolnoprzepustowy jest zamieniany w górnoprzepustowy.
Okno Blackmana	*Blackman window*	Krzywa o gładkiej charakterystyce stosowana w projektowaniu filtrów i w analizie widmowej, obliczana z wzoru: $0{,}42 - 0{,}5\cos(2\pi n/M) + 0{,}08\cos(4\pi n/M)$, gdzie n zmienia się od 0 do M.
Okno Hamminga	*Hamming window*	Gładka krzywa stosowana w projektowaniu filtrów i w analizie widmowej, obliczana z wzoru: $0{,}54 - 0{,}46\cos(2\pi n/M)$, gdzie n zmienia się od 0 do M.
Okno o płaskim wierzchołku	*Flat-top window*	Okno stosowane w analizie widmowej, zapewnia dokładny pomiar amplitud składowych widma. Może być stosowana odpowiedź impulsowa filtru z okienkowaniem *sinc*.
Okno prostokątne	*Rectangular window*	Sygnał zawierający grupę sąsiadujących ze sobą punktów o wartości jeden, a wszędzie poza tym obszarem – zero. Zwykle jest mnożony przez inny sygnał w celu wybrania, do przetwarzania, pewnego fragmentu tego sygnału.
Okrąg jednostkowy	*Unit circle*	Okrąg na płaszczyźnie z o promieniu $r = 1$. Wartości wzdłuż tego okręgu tworzą odpowiedź częstotliwościową systemu.
„Okrągła czapka"	*Pillbox*	Kształt odpowiedzi impulsowej filtru stosowanej w przetwarzaniu obrazów: obszar okrągły o wartości stałej, otoczony przez zera.
Oktawa	*Octave*	Czynnik równy dwa na skali częstotliwości.
Otwarcie	*Opening*	Operacja morfologiczna definiowana jako erozja, po której nastąpiła dylatacja.
PAL	*PAL*	Standard telewizyjny stosowany w Europie. Dla porównania proszę zobaczyć: *NTSC*.
Pamiętanie rzędu zerowego	*Zeroth-order hold*	Pojęcie stosowane w przetwornikach c/a dla stwierdzenia faktu, że między kolejnymi przetwarzaniami jest utrzymywana stała wartość sygnału analogowego, czego wynikiem jest przebieg schodkowy.
Para linii	*Line pair*	W technice obrazowej pojęcie stosowane zamiast okresu. Na przykład 5 okresów na milimetr zastępuje się przez 5 par linii na milimetr.

Para transformat Fouriera	*Fourier transform pair*	Przebiegi w dziedzinie czasu i częstotliwości będące wzajemnie swymi odpowiednikami. Na przykład – impuls prostokątny i funkcja *sinc*.
Pasmo przejściowe	*Transition band*	W terminologii filtracji pasmo częstotliwości między pasmem przepustowym i zaporowym. W tym paśmie następuje opadanie charakterystyki.
Pasmo przepustowe	*Passband*	Pasmo częstotliwości, które filtr ma przenosić bez zmian.
Pasmo zaporowe	*Stopband*	Pasmo częstotliwości, które filtr powinien tłumić.
Pierwsza różnica	*First diffference*	Operacja na sygnałach dyskretnych będąca naśladownictwem pierwszej pochodnej sygnałów ciągłych, nazywana też *pochodną dyskretną*.
Piksel	*Pixel*	Element obrazowy, skrót od *picture element*. Pojedyncza próbka w obrazie cyfrowym.
Plamka żółta	*Fovea*	Mały obszar na siatkówce oka zoptymalizowany do widzenia z bardzo dużą rozdzielczością.
Płaszczyzna zmiennej zespolonej	*Complex plane*	Graficzna interpretacja liczb zespolonych z częścią rzeczywistą na osi x i urojoną na osi y. Jest to analogia do *osi liczbowej* stosowanej do liczb rzeczywistych.
Pochodna dyskretna	*Discrete derivative*	Operacja na sygnałach dyskretnych analogiczna z pochodną dla sygnałów ciągłych. Lepszą nazwą jest *pierwsza różnica*.
Podprogram „mityczny" (przyp. tłum.)	*Mythical subroutine*	Tak są w książce nazywane podprogramy, dla których nie określono, jak są przeprowadzane operacje.
Podziałka z parami linii	*Line pair gauge*	Urządzenie stosowane do pomiaru rozdzielczości systemu obrazowego. Zawiera szereg linii jasnych i ciemnych. Odstępy między nimi zmniejszają się w kierunku jednego z końców podziałki.
Pojedyncza dokładność	*Single precision*	Zapis zmiennoprzecinkowy, w którym do wyrażenie każdej liczby stosuje się 32 bity. Dla porównania proszę zobaczyć: *dokładność podwójna*.
Półobraz	*Field*	W telewizji z wybieraniem międzyliniowym są wyświetlane linie parzyste każdego obrazu (ramki), a następnie linie nieparzyste. Linie parzyste są nazywane *półobrazem parzystym*, a nieparzyste – *półobrazem nieparzystym*.
Projekcja wsteczna	*Backprojection*	Technika stosowana w tomografii komputerowej do odtwarzania obrazu na podstawie jego widoków. W wyniku daje kiepski obraz, jeśli nie jest stosowana wraz z jakąś bardziej zaawansowaną technicznie metodą.
Przeciek widmowy	*Spectral leakage*	Pojęcie stosowane w analizie widmowej. Ponieważ przekształceniu DFT można poddawać tylko sygnał o skończonej długości, więc widmo częstotliwościowe jest pikiem z „ogonami" po bokach. Te „ogony" są traktowane jako *przeciek* z widma głównego.
Przekształcenie	*Transform*	Procedura, równanie lub algorytm, który zmienia jedną grupę danych w inną grupę danych.
Przekształcenie biliniowe	*Bilinear transform*	Metoda stosowana do odwzorowywania płaszczyzny s na płaszczyźnie z. Umożliwia konwersję filtrów analogowych na równoważne im filtry cyfrowe.
Przekształcenie fazorowe	*Phasor transform*	Patrz: *metoda symboliczna*.
Przekształcenie Fouriera	*Fourier transform*	Rodzina metod matematycznych polegających na rozkładzie sygnałów na sinusoidy. W wersji zespolonej sygnały podlegają rozkładowi na zespolone przebiegi wykładnicze.
Przekształcenie Laplace'a	*Laplace transform*	Matematyczna metoda analizy systemów, których działanie jest opisane równaniami różniczkowymi. Główne narzędzie do projektowania układów elektrycznych, takich jak filtry analogowe. Zamienia sygnał w dziedzinie czasu na sygnał w dziedzinie *s*.
Przekształcenie odwrotne	*Inverse transform*	Równanie syntetyczne przekształcenia Fouriera, służące do obliczania przebiegu w dziedzinie czasu na podstawie przebiegu w dziedzinie czasu. Dla porównania proszę zobaczyć: *przekształcenie proste*.

Przekształcenie proste	*Forward transform*	Równanie analityczne przekształcenia Fouriera, służące do obliczania przebiegu w dziedzinie częstotliwości na podstawie przebiegu w dziedzinie czasu. Dla porównania proszę zobaczyć: *przekształcenie odwrotne*.
Przekształcenie skali szarości	*Grayscale transform*	Funkcja konwersji między wartością zarejestrowanego piksela i jaskrawością, z jaką pojawia się na wyświetlanym obrazie. Zwana też krzywą gamma.
Przekształcenie Z	*z-transform*	Metoda matematyczna stosowana do analizy systemów dyskretnych, których działanie opisują równania różnicowe – np. filtrów rekursywnych (NOI). To przekształcenie zamienia sygnał w dziedzinie czasu na sygnał w dziedzinie *z*.
Przemienność operacji splotu	*Commutative property of convolution*	Właściwość splotu zapisywana następująco: $a[n] * b[n] = b[n] * a[n]$
Przestrzeń parametryczna	*Parameter space*	Nazwa używana w terminologii techniki obrazowej. Jeden parametr może być graficznie interpretowany jako *linia*, dwa – jako *płaszczyzna*, trzy – jako *przestrzeń*, a więcej niż trzy parametry – jako *hiperprzestrzeń*.
Przestrzeń próbkowania	*Sample spacing*	Odstęp między próbkami, gdy obraz ciągły jest przetwarzany na cyfrowy. Definiowany jako odległość między środkami pikseli.
Przesuwanie i odejmowanie	*Shift and substracct*	W przetwarzaniu obrazów – operacja tworząca efekt 3D (trójwymiarowości) lub efekt obrazu wytłaczanego.
Przetwarzanie w czasie rzeczywistym	*Real time processing*	Przetwarzanie danych w miarę ich pobierania, a nie gromadzenia ich do późniejszego wykorzystania. Przykład: Algorytmy DSP do tłumienia echa w rozmowach telefonicznych na duże odległości.
Przypadek fałszywie negatywny	*False-negative*	Jeden z czterech możliwych do uzyskania wyników w metodzie wykrywania celu. Cel istnieje, lecz nieprawidłowo wykryto, że go nie ma.
Przypadek fałszywie pozytywny	*False-positive*	Jeden z czterech możliwych do uzyskania wyników w metodzie wykrywania celu. Celu nie ma, lecz nieprawidłowo wykryto jego istnienie.
Przypadek prawdziwie negatywny	*True-negative*	Jeden z czterech możliwych do uzyskania wyników w metodzie wykrywania celu. Cel nie istnieje i prawidłowo wykryto, że go nie ma.
Przypadek prawdziwie pozytywny	*True-positive*	Jeden z czterech możliwych do uzyskania wyników w metodzie wykrywania celu. Cel istnieje i prawidłowo wykryto jego istnienie.
Pulsacja	*Natural frequency*	Częstotliwość wyrażona w radianach na sekundę, a nie w okresach na sekundę (Hz). Trzeba częstotliwość (Hz) pomnożyć przez 2π, aby uzyskać pulsację.
Radar	*Radar*	Radio Detection And Ranging. System radiolokacyjny, w którym stosuje się fale radiowe do wykrywania samolotów.
Ramka (obraz)	*Frame*	Pojedynczy obraz w sygnale telewizyjnym. W amerykańskim standardzie telewizyjnym NTSC stosuje się 30 ramek/s, a w obowiązującym w Polsce standardzie PAL – 25 ramek/s.
Rekonstrukcja Fouriera	*Fourier reconstruction*	Jedna z metod stosowanych w tomografii komputerowej do obliczania obrazu na podstawie jego widoków.
RISC	*RISC*	Reduced Instruction Set Computer – komputer o zredukowanej liczbie rozkazów, zwany też mikroprocesorem DSP. Mniejsza liczba rozkazów programowych umożliwia zwiększenie szybkości obliczeń matematycznych. Jego przeciwieństwem jest komputer z rozbudowaną listą rozkazów, taki jak Pentium.
Rozciąganie skali szarości	*Grayscale stretch*	Znaczne zwiększenie kontrastu obrazu cyfrowego w celu umożliwienia szczegółowego obejrzenia poziomów kwantyzacji w małym zakresie. Poziomy kwantyzacji poza tym obszarem są wyświetlane jako nasycone na biało lub na czarno.
Rozdzielczość częstotliwościowa	*Frequency resolution*	Zdolność rozróżniania lub rozdzielania częstotliwości leżących blisko siebie.
Rozdzielny (obraz)	*Separable*	Obraz, który może być reprezentowany jako iloczyn swych rzutów – pionowego i poziomego. Stosowany do zwiększania szybkości wykonywania splotów obrazów.

Rozkład (czyli dekompozycja)	*Decomposition*	Proces rozdzielenia sygnału na dwie lub więcej składowych addytywnych. Często to pojęcie odnosi się zwłaszcza do prostego *przekształcenia Fouriera*, w którym sygnał podlega rozkładowi na sinusoidy.
Rozkład impulsowy	*Impulse decomposition*	Rozdzielenie sygnału *N*-próbkowego na *N* sygnałów składowych, z których każdy zawiera pojedynczą próbkę z sygnału oryginalnego i pozostałe próbki zerowe. Jest to podstawa splotu.
Rozkład normalny	*Normal distribution*	Krzywa o kształcie dzwonowym w postaci: e^{x^2}. Zwany też *rozkładem gaussowskim*.
Rozkład parzysto-nieparzysty	*Even/odd decomposition*	Sposób rozdzielenia sygnału na dwa inne, z których jeden ma symetrię parzystą, a drugi nieparzystą.
Rozkład z przeplotem	*Interlaced decomposition*	Rozdzielenie sygnału na dwa sygnały składowe – sygnał próbek parzystych oraz sygnał próbek nieparzystych. Stosowany w FFT.
Rozmieszczenie zer i biegunów	*Pole-zero diagram*	Pojęcie stosowane w przekształceniu Laplace'a i w przekształceniu Z. Graficzne przedstawienie położeń biegunów i zer na płaszczyźnie *s* lub na płaszczyźnie *z*.
Rozplot	*Deconvolution*	Operacja odwrotna do splotu: jeśli $x[n] * h[n] = y[n]$, to trzeba wyznaczyć $x[n]$, mając dane tylko $h[n]$ i $y[n]$. Rozplot jest zwykle przeprowadzany przez dzielenie widm częstotliwościowych.
Równanie rekursywne	*Recursion equation*	Równanie określające zależność między przeszłymi i przyszłymi próbkami sygnału wyjściowego a przeszłymi i przyszłymi próbkami sygnału wejściowego. Zwane też *równaniem różnicowym*.
Równanie różnicowe	*Difference equation*	Równanie opisujące zależność przeszłych i obecnych próbek sygnału wyjściowego od przeszłych i obecnych próbek sygnału wejściowego. Zwane też *równaniem rekursywnym*.
Równoważność matematyczna	*Mathematical equivalence*	Sposób zastosowania liczb zespolonych do wyrażania zagadnień rzeczywistych. Jej podstawą jest zależność Eulera określająca równoważność sinusoid i zespolonych funkcji wykładniczych. Dla porównania proszę zobaczyć: *zastępowanie*.
Rzeczywiste DFT	*Real DFT*	Dyskretne przekształcenie Fouriera, w którym stosuje się tylko liczby rzeczywiste. Metoda o mniejszych możliwościach niż zespolone DFT, lecz prostsza. Dla porównania proszę zobaczyć: *zespolone przekształcenie DFT*.
Rzeczywiste FFT	*Real FFT*	Zmodyfikowana wersja FFT, o ok. 30% szybsza od standardowego przekształcenia FFT, w której sygnał w dziedzinie czasu jest całkowicie rzeczywisty (czyli część urojona w dziedzinie czasu jest równa zero).
Rzeczywiste przekształcenie Fouriera	*Real Fourier transform*	Każde przekształcenie z grupy fourierowskiej, w którym stosuje się tylko liczby rzeczywiste (w odróżnieniu od liczb zespolonych lub urojonych). Dla porównania proszę zobaczyć: *zespolone przekształcenie Fouriera*.
SECAM	*SECAM*	Standard telewizyjny stosowany w Europie. Dla porównania proszę zobaczyć: *NTSC*.
Sigmoida	*Sigmoid*	Krzywa o kształcie litery S, stosowana m.in. w sieciach neuronowych.
Skończona odpowiedź impulsowa (SOI)	*Finite impulse response (FIR)*	Odpowiedź impulsowa mająca skończoną liczbę wartości niezerowych. Często stosowana do wykazania, że filtr jest realizowany przy użyciu splotu, a nie metodą rekursywną.
Sonar	*Sonar*	Sound Navigation And Ranging. Zastosowanie dźwięku do wykrywania łodzi podwodnych i innych obiektów podwodnych. W sonarach aktywnych stosuje się echolokację, w pasywnych zaś tylko odbiór sygnałów.
Sonar pasywny	*Passive sonar*	Wykrywanie łodzi podwodnych i innych obiektów znajdujących się pod wodą za pomocą dźwięków wytwarzanych przez to urządzenia. Stosowany do ukrytego nadzoru obszaru wodnego.
Spektrogram głosu	*Voice spectrogram*	Wynik pomiaru zmian widma częstotliwościowego sygnału fonicznego w funkcji czasu. Zwykle jest wyświetlany jako obraz. Zwany też *odbitką głosu*.

Splot FFT	*FFT convolution*	Metoda określania splotu dwóch sygnałów przez wymnożenie ich widm częstotliwościowych. Nadano jej tę nazwę, gdyż przekształcenie FFT umożliwia łatwe poruszanie się między dziedzinami czasu i częstotliwości.
Splot kołowy	*Circular convolution*	Zjawisko nakładania się (aliasingu) mogące wystąpić w dziedzinie czasu, gdy sygnały w dziedzinie częstotliwości są przez siebie mnożone. Każdy okres w dziedzinie czasu zachodzi na okresy sąsiednie.
Splot o dużej szybkości	*High speed convolution*	Inna nazwa splotu FFT.
Splotowa odpowiedź impulsowa filtru	*Convolution kernel*	Odpowiedź impulsowa filtru implementowanego przez splot. Znana też jako splotowe jądro filtru (*filter kernel*).
Splot w dziedzinie częstotliwości	*Frequency domain convolution*	Splot wykonany przez wymnożenie widm częstotliwościowych sygnałów.
Statystyka Poissona	*Poisson statistics*	Zmiany wartości sygnału wynikające z faktu, że ta wartość jest wyrażana przez skończoną liczbę pewnych elementów, takich jak np. kwanty promieniowania rentgenowskiego, fotony światła lub elektrony. Zwana też szumem Poissona lub szumem statystycznym.
Stopnie równoległe	*Parallel stages*	Połączenie dwóch lub więcej stopni mających to samo wejście i sumowane wyjścia.
Stromość opadanie charakterystyki	*Roll-off*	W ten sposób określa się stromość przejścia między pasmami przepustowym i zaporowym filtru. *Szybkie* opadanie oznacza, że przejście jest strome, a *wolne* – że następuje stopniowo.
Studnia	*Well*	Skrócona nazwa *studni potencjału*; najbardziej czuły na światło obszar w układzie ze sprzężeniem ładunkowym (CCD).
Suma bieżąca	*Running sum*	Operacja wykonywana na sygnałach dyskretnych, będąca naśladownictwem całkowania sygnałów ciągłych.
Suma splotowa	*Convolution sum*	Wzór matematyczny definiujący splot systemów dyskretnych.
Sygnał	*Signal*	Opis zmian jednego parametru w funkcji zmian drugiego parametru. Na przykład: *napięcie*, które zmienia się w funkcji *czasu*.
Sygnał ciągły	*Continuous signal*	Sygnał utworzony ze zmiennych ciągłych (w przeciwieństwie do *sygnału dyskretnego*). Przykład: napięcie zmieniające się w funkcji czasu. Nazwa stosowana często zamiennie z *sygnałem analogowym*.
Sygnał dyskretny	*Discrete signal*	Sygnał, w którym są użyte zmienne dyskretne, taki jak sygnał przetworzony na cyfrowy i zarejestrowany w komputerze.
Sygnał mieszany	*Mixed signal*	Pojęcie używane w odniesieniu do układów scalonych zawierających zarówno elementy analogowe, jak i cyfrowe, takich jak np. przetwornik a/c umieszczony w cyfrowym procesorze sygnałowym.
Sygnał przyczynowy	*Causal signal*	Każdy sygnał mający wartość zero dla wszystkich próbek o ujemnych numerach.
System	*System*	Każdy proces wytwarzający sygnał wyjściowy w odpowiedzi na wejściowy.
System liniowy	*Linear system*	Z definicji – system charakteryzujący się addytywnością i homogenicznością.
System przyczynowy	*Causal system*	System, który ma na wyjściu zero tak długo, dopóki nie pojawi się wartość niezerowa na jego wejściu (tzn. sygnał wejściowy jest *przyczyną* wyjściowego). Odpowiedź impulsowa systemu przyczynowego jest sygnałem przyczynowym.
System świergotowy	*Chirp system*	Stosowany w radarach i sonarach. Impuls, przed wysłaniem go, jest przetwarzany na sygnał o dłuższym czasie trwania i po odbiorze ponownie zwężany do postaci impulsu.
Systemy o wielu częstotliwościach próbkowania	*Multirate*	Systemy, w których stosuje się więcej niż jedną częstotliwość próbkowania. Często stosowane w przetwornikach a/c i c/a w celu poprawy ich działania przy użyciu mniejszej liczby układów elektronicznych.
Szereg Fouriera	*Fourier series*	Jedno z przekształceń z rodziny fourierowskiej operujące *ciągłymi i okresowymi* sygnałami w dziedzinie czasu.

Szerokość w połowie maksymalnej wysokości	*Full-width-at-half-maximum (FWHM)*	Powszechnie stosowany sposób wyrażania szerokości piku w sygnale. Szerokość piku jest mierzona w połowie jego amplitudy.
Szum 1/f	*1/f noise*	Rodzaj szumu losowego, którego wartość wzrasta przy małych częstotliwościach. Jest często spotykany w systemach fizycznych, lecz jego pochodzenie nie jest dobrze wyjaśnione. Dla porównania proszę zobaczyć: *szum biały*.
Szum biały	*White noise*	Szum losowy o płaskim widmie częstotliwościowym. Występuje wtedy, gdy znajomość wartości szumu w którejkolwiek próbce nie dostarcza żadnej informacji o wartości szumu w innych próbkach. Dla porównania proszę zobaczyć: *szum 1/f*.
Szum statystyczny	*Statistical noise*	Zmiany wartości sygnału wynikające z faktu, że ta wartość jest wyrażana przez skończoną liczbę pewnych elementów, takich jak kwanty promieniowania rentgenowskiego, fotony światła lub elektrony. Zwany też *szumem Poissona* lub *statystyką Poissona*.
Szum zaokrąglenia	*Round-off noise*	Błąd spowodowany zaokrąglaniem wyników operacji matematycznych do najbliższego poziomu kwantyzacji.
Szybkie przekształcenie Fouriera (FFT)	*Fast Fourier Transform (FFT)*	Efektywny algorytm do obliczania dyskretnej transformaty Fouriera (DFT). W niektórych przypadkach zmniejsza czas wykonania nawet *kilkusetkrotnie*.
Ślimak	*Cochlea*	Część ucha, w której dźwięk jest przetwarzany na sygnał neuronowy.
Średnia	*Mean*	Wartość średnia sygnału lub innej grupy danych.
TIFF	*TIFF*	Format pliku obrazowego często stosowany w przetwarzaniu tekstów i w podobnych programach. Zwykle niepodlegający kompresji, chociaż opcją jest kompresja LZW. Dla porównania proszę zobaczyć: *GIF* i *JPEG*.
Tomografia komputerowa	*Computed tomography (CT)*	Metoda stosowana do odtwarzania wnętrza obiektu na podstawie rzutów obrazów rentgenowskich. Szeroko stosowana w medycynie. Jest jednym z najwcześniejszych zastosowań DSP. Stara nazwa: skaner CAT.
Transformator Hilberta	*Hilbert transformer*	System mający odpowiedź częstotliwościową: moduł = 1, faza = 90° dla wszystkich częstotliwości. Stosowany do modulacji w systemach telekomunikacyjnych. Może być analogowy lub cyfrowy.
Twierdzenie o próbkowaniu	*Sampling theorem*	Jeśli sygnał ciągły zawierający częstotliwości mniejsze niż *f* jest próbkowany z częstotliwością *2f*, to cała informacja znajdująca się w sygnale ciągłym będzie również obecna w sygnale spróbkowanym. Często zwane *twierdzeniem o próbkowaniu Shannona lub Nyquista*.
Układ ze sprzężeniem ładunkowym (CDD)	*Charge-coupled device (CCD)*	Taki układ jest zwykle czujnikiem światła w elektronicznych aparatach fotograficznych. Zbudowany jest z cienkiej warstwy krzemu zawierającej dwuwymiarową matrycę obszarów światłoczułych zwanych *studniami*.
Wartość średniokwadratowa	*Root-mean-square (rms)*	Wielkość stosowana do określania fluktuacji sygnału wokół zera. Często używana w elektronice. Definiowana jako pierwiastek kwadratowy ze średniej kwadratów. Dla porównania proszę zobaczyć: *odchylenie standardowe*.
Wierność odtwarzania sinusoidy	*Sinusoidal fidelity*	Ważna właściwość systemów liniowych. Wejściowy sygnał sinusoidalny powinien na wyjściu dawać też tylko sygnał sinusoidalny. Amplituda i faza mogą się zmieniać, lecz częstotliwość musi pozostać taka sama.
Wskaźnik	*Pointer*	Zmienna, której wartość jest adresem innej zmiennej.
Współczynniki rekursywne	*Recursion coefficients*	Wartości wagowe stosowane w równaniu rekursywnym. Współczynniki rekursywne opisują właściwości filtru rekursywnego (NOI).
Współczynnik rozrzutu	*Coefficient of variation (CV)*	Często spotykany sposób wyrażania odchyleń (szumu) danych. Określony jako: 100% × odchylenie standardowe / wartość średnia.
Współczynnik tłumienia w paśmie zaporowym	*Stopband attenuation*	Wielkość tłumienia sygnału o częstotliwościach znajdujących się w paśmie zaporowym filtru, zwykle wyrażana w decybelach. Stosowany do opisu działania filtrów.

Wybieranie międzyliniowe	*Interlaced video*	Dla sygnału wideo z wybieraniem międzyliniowym najpierw są wyświetlane linie parzyste każdego obrazu, a następnie linie nieparzyste. To wybieranie jest stosowane w telewizji, a opracowane zostało w celu zredukowania migotania.
Wykrywanie celu	*Target detection*	Decydowanie, na podstawie zmierzonych wartości, czy jakiś obiekt lub warunek jest obecny.
Wyostrzanie	*Sharpening*	W przetwarzaniu obrazów – operacja dająca bardziej ostre krawędzie.
Wyostrzanie krawędzi	*Edge enhancement*	Każdy z algorytmów przetwarzających, który czyni krawędzie bardziej wyraźnymi.
Wyrównywanie histogramu	*Histogram equalization*	Przetwarzanie obrazu z wykorzystaniem scałkowanego histogramu jako przekształcenia skali szarości. Daje dużym obszarom obrazu większy kontrast niż obszarom małym.
Wysoka wierność	*High fidelity*	Bardzo dobra jakość odtwarzania muzyki, np. w odtwarzaczach CD.
Wysokość dźwięku	*Pitch*	Percepcja podstawowej częstotliwości dźwięku przez słuch ludzki. Dla porównania proszę zobaczyć: *barwa dźwięku*.
Zakres dynamiczny	*Dynamic range*	Stosunek największej wartości sygnału, którą system może prawidłowo operować, do własnego szumu systemu. Również używany do określania liczby bitów zastosowanych w przetworniku a/c. Pojęcie to może być też używane do innych parametrów niż wielkość sygnału, proszę zobaczyć: *dynamiczny zakres częstotliwości*.
Zakres dynamiczny częstotliwości	*Frequency dynamic range*	Stosunek największej do najmniejszej częstotliwości, z jakimi system może prawidłowo operować.
Zależność Parsevala	*Parseval's relation*	Równanie określające zależność między energią w dziedzinie czasu a energią w dziedzinie częstotliwości.
Zamknięcie	*Closing*	Operacja morfologiczna określana jako dylatacja, po której następuje erozja.
Zapis stałoprzecinkowy	*Fixed point*	Jeden z dwóch powszechnie stosowanych sposobów zapisu liczb w komputerach, zwykle używany do zapisu liczb całkowitych. Dla porównania proszę zobaczyć: *zapis zmiennoprzecinkowy*.
Zapis we współrzędnych biegunowych	*Polar form*	Wyrażanie sinusoid przez ich moduł i fazę: $M\cos(\omega t+\Phi)$, gdzie M jest modułem a Φ fazą. Dla porównania proszę zobaczyć: zapis we *współrzędnych prostokątnych*.
Zapis we współrzędnych prostokątnych	*Rectangular form*	Wyrażanie przebiegów sinusoidalnych w postaci: $A\cos(\omega t) + B\sin(\omega t)$, gdzie A nosi nazwę części rzeczywistej, a B – części urojonej (chociaż nie są to liczby urojone).
Zapis zmiennoprzecinkowy	*Floating point*	Jeden z dwóch powszechnie stosowanych sposobów zapisu liczb w komputerach. Stosuje się tu rodzaj zapisu naukowego, w którym mantysa jest podnoszona do jakiejś potęgi. Dla porównania proszę zobaczyć: *zapis stałoprzecinkowy*.
Zasada „A"	*„A" law*	Standard kompansji używany w Europie. Dzięki użyciu niejednakowych odstępów między poziomami kwantyzacji umożliwia reprezentację cyfrowych sygnałów głosowych za pomocą tylko 8 bitów, zamiast 12. Dla porównania proszę zobaczyć: *zasada „μ"*.
Zasada „μ"	*„μ" law*	Standard kompansji używany w USA. Dzięki użyciu niejednakowych odstępów między poziomami kwantyzacji umożliwia reprezentację cyfrowych sygnałów głosowych za pomocą tylko 8 bitów, zamiast 12. Dla porównania proszę zobaczyć: *zasada „A"*.
Zastępowanie	*Substitution*	Sposób zastosowania liczb zespolonych do wyrażania problemów fizycznych, takich jak projektowanie obwodów elektrycznych. W tej metodzie wprowadzane są symbole j, aby problem fizyczny przetworzyć w problem z dziedziny liczb zespolonych i następnie usunąć symbole j w celu powrotu do zagadnienia fizycznego. Dla porównania proszę zobaczyć: *równoważność matematyczna*.
Zbieżność	*Converge*	Pojęcie stosowane w metodach iteracyjnych w celu wskazania na postęp w kierunku osiągnięcia rozwiązania („algorytm jest zbieżny") albo stwierdzenia, że rozwiązanie już uzyskano („algorytm osiągnął zbieżność").

Zero	*Zero*	Pojęcie stosowane w przekształceniach Z i Laplace'a. Gdy funkcja przenoszenia w dziedzinie s lub w dziedzinie z jest zapisana jako iloraz dwóch wielomianów, to pierwiastki licznika są zerami systemu. Proszę też zobaczyć: *biegun*.
Zerofazowy (system)	*Zero phase*	System mający fazę całkowicie zerową. Taka sytuacja jest tylko wówczas, gdy odpowiedź impulsowa ma symetrię lewo-prawostronną w stosunku do początku układu współrzędnych. Proszę też zobaczyć: *liniowa faza*.
Zespolone przekształcenie DFT	*Complex DFT*	Dyskretne przekształcenie Fouriera, w którym stosuje się liczby zespolone. Metoda bardziej skomplikowana i o większych możliwościach niż *rzeczywiste* DFT.
Zespolone przekształcenie Fouriera	*Complex Fourier transform*	Każde z czterech przekształceń z rodziny fourierowskiej zapisane z użyciem liczb zespolonych. Dla porównania proszę zobaczyć: *rzeczywiste przekształcenie Fouriera*
Zmienna niezależna i zależna	*Independent variable*	Wartość zmiennej zależnej w sygnale zależy od wartości zmiennej niezależnej. Na przykład: gdy napięcie zmienia się w funkcji czasu, czas jest zmienną niezależną, napięcie zaś – zależną.

Indeks

A

ADC16071 71
addytywność 93, 94, 99, 184, 186
  — przekształcenia Fouriera 185
adresy z odwracaniem bitowym 490
ADSP-2106x 510
ADSP-TS101 87
akumulacja błędu zaokrąglenia 79
akumulator
  — układu mnożącego (MAC) 491
  — o rozszerzonej dokładności 493
algorytm
  — ewolucji 449
  — od strony wejścia 113
  — uczący się 442
aliasing 48, 61, 193, 197, 288, 358
  — harmonicznych 216
  — w dziedzinie czasu 192, 197
  — w dziedzinie częstotliwości 195, 197
analiza
  — bloków 418
  — DTFT 203
  — Fouriera 105, 382, 108
  — splotowa 108
  — spójności 418
  — widmowa 169, 211
Analog Devices 507, 508, 510, 517
aparaty fotograficzne 362
apertura próbkowania 362, 414
architektura
  — potokowa 88
  — sieci neuronowej 439
  — superharwardzka 487
  — von Neumanna 487, 488
asemblacja 81
asembler 499, 501
autokorelacja 138
automatyczna regulacja
  wzmocnienia ARW 357

B

barwa 343, 344
  — dźwięku 343
bieguny 563, 567, 568
błąd
  — addytywny 80
  — dzielenia przez zero 163
  — losowy 41, 80
  — systematyczny 41
  — zaokrąglenia 25, 79
błona bębenkowa 340
budowa ludzkiego oka 364
buforowanie cykliczne 484, 485

C

całka
  — dyskretna 126
  — splotowa 240
całkowity sygnał wizji 369
centralne twierdzenie graniczne 40, 134, 135
cepstrum 357
charakterystyka
  — fazowa 188, 191
  — widmowa oka 367
ciągła funkcja delta 238, 239
ciąg impulsów 51, 53
CS&Q 460, 461
cyfrowy procesor sygnałowy 482
czas
  — narastania 256
  — wykonania DFT 230, 231
  — wykonania splotów obrazowych 404
część
  — rzeczywista 221, 528, 234
  — urojona 221, 528, 234
częstotliwościowe widmo obrazu 395
częstotliwości ujemne 193, 195, 196, 543
częstotliwość
  — Nyquista 49, 50, 53
  — odcięcia 256, 312

— podstawowa 216, 247
— próbkowania 148
czopki 365

D

decybel (dB) 254, 341
decymacja 66, 71, 199
— w dziedzinie czasu 227
— w dziedzinie częstotliwości 227
dekoder Reeda-Solomona 348
dekodowanie EFM 348
dekompresja LZW 471
detekcja
— fazy 344
— krawędzi 387
detektor promieniowania gamma 290
dithering 46, 47
— subtraktywny 47
długa liczba całkowita 78
długość przekształcenia DFT 172
dokładność 41, 42
— liczb 78
Dolby
— Stereo 350
— Surround Pro Logic 350
dolna wstęga boczna 51
dopełnianie sygnału zerami 119, 175, 258
DSP
— 56800 88
— stałoprzecinkowe 491, 492
— zmiennoprzecinkowe 491, 492
dualizm 159
dwukierunkowa filtracja rekursywna 318, 319
dylatacja 418
dyskretne przekształcenie
— Fouriera 142, 144, 145, 548, 550
— kosinusowe DCT 474
dyskretne w czasie przekształcenie Fouriera (DTFT) 144, 145, 177, 202, 548, 550
dziedzina
— czasu 23, 147, 168, 559
— częstotliwości 23, 147, 168, 559
— przestrzeni 23, 360
— s 554, 559
— z 576
dzielenie liczb zespolonych 532
dzwonienia 62

E

echo 125
echolokacja 17
efekty krańcowe 121
efekt Gibbsa 142, 213, 214
ekspansja 197, 198
elektroencefalogram (EEG) 280
elektroniczny aparat fotograficzny 364
element neutralny 124, 125
emulatory 525
entropia obrazu 377
erozja 418

F

fala nośna 202
faza 190
— liniowa 316
— nieliniowa 316
— zerowa 316
filtracja
— dopasowana 138
— optymalna 286, 295
— Wienera 355
filtr
— analogowy 57, 333
— antyaliasingowy 56, 63, 64
— Bessela (zwany też filtrem Thomsona) 56, 58, 62, 63, 65, 66
— bikwadratowy 583
— Butterwortha 56, 58, 60, 62, 63, 322, 324, 571
— Czebyszewa 56, 58, 59, 60, 61, 62, 63, 65, 322, 323, 324, 332, 335, 572
— dolnoprzepustowy 112, 128, 211, 281, 310, 384
— dolnoprzepustowy Czebyszewa 326
— dopasowany 295
— eliptyczny 61, 322, 572
— gaussowski 270

— górnoprzepustowy 112, 128, 260, 384
— górnoprzepustowy Czybyszewa 327
— jednobiegunowy 310, 332
— jednobiegunowy dwukierunkowy 338
— jednobiegunowy górnoprzepustowy 310, 311
— jednobiegunowy rekursywny 313, 337
— korelacyjny 399
— NOI (o nieskończonej odpowiedzi impulsowej) 264, 308, 336, 337, 503
— odtwarzający 56
— okienkowany funkcją sinc 211, 274, 280, 281, 284, 332, 335
— okienkowany funkcją sin(x)/x 211
— optymalny 296, 297
— o charakterystyce maksymalnie płaskiej 322
— o średniej kroczącej 266, 267, 270, 295, 332, 337, 338
— pasmowoprzepustowy 260, 262, 281, 283, 314
— pasmowozaporowy 260, 263, 314, 582
— rekonstrukcyjny 56
— rekursywny 308, 309, 454, 457
— SOI (o skończonej odpowiedzi impulsowej) 264, 286, 308, 336, 337, 486, 503, 513
— wąskopasmowy 315
— Wienera 296, 356
— wycinający 539, 540
— wygładzający 384
— zaostrzający krawędzie 384
— zerofazowy 358, 588
— z przełączanymi pojemnościami 58, 59
fluktuacja statystyczna 27
formanty 352
fortran 228
frame grabber 371
funkcja
— bazowa (DFT) 150, 151, 475
— delta 110, 124, 206, 238
— gęstości prawdopodobieństwa (pdf) 32
— LSF 411
— masy prawdopodobieństwa (pmf) 27, 29, 30, 32
— ortogonalna 158
— przenoszenia 565, 580, 581
— przenoszenia modulacji (MTF) 407, 410
— przenoszenia w postaci wielomianowej 566
— rozkładu prawdopodobieństwa (pdf) 31, 432
— funkcja rozmycia linii (LSF) 409
— funkcja rozmycia punktu (PSF) 382, 383, 385
— sigmoidalna 441
— sinc 55, 274, 386
— sin(x)/x 55, 208
FWHM 406

G

generacja
— mowy 16
— szumu cyfrowego 35, 38
generator
— adresów danych 489
— liczb losowych 38, 40
gęstość
— prawdopodobieństwa 31, 32, 33
— widmowa 155, 203
głębia ostrości 364
głośność 343
górna wstęga boczna 51

H

harmoniczne 216, 247
hiperprzestrzeń 437
histogram 27, 29, 30, 32
— przedziałowy 34
— tekstu 464
homogeniczność 93, 94, 99, 184
homomorficzne przetwarzanie sygnałów 108, 356

I

identyfikacja liter 445
iloczyn skalarny 498, 499
impedancja 538
impuls jednostkowy 110
interpolacja 67, 199, 349
— dwuliniowa 380
— przez dopełnianie 199

— subpikselowa 380
interpreter 83
— Basic 83
inwersja widmowa 260, 261, 263, 283, 587, 588

J

jądro filtru 111, 253
jaskrawość 372, 373
Jean Baptiste Joseph Fourier 142
jednobitowe przetwarzanie
— a/c i c/a 67
— danych 67
jednostka
— arytmetyczno-logiczna (ALU) 86, 490, 491
— jednostka centralna 85
język
— asembler 496, 497
— Basic 9, 25, 38, 83, 84, 85, 90, 229
— C 9, 10, 84, 496, 497, 501, 502, 521
— Fortran 9, 10, 84
— Pascal 84
Joseph Louis Lagrange 142
Josiah Gibbs 213
JPEG 472, 460

K

kaskadowe łączenie 585
kąt fazowy 530
klasyfikacja filtrów 264
klawiatura fortepianu 345
krzywe kompansji 350
kodowanie
— arytmetyczne 464
— ciągów 460, 461
— delta 460, 465
— Huffmana 460, 463, 464, 465
— JPEG 475
— Reeda-Solomona 349
— RGB 366
— w dziedzinie czasu 63, 64
— w dziedzinie częstotliwości 63
kod
— ASCII 462, 482
— maszynowy 82
— wewnętrzny 82
— wykonawczy 82
— wynikowy 82
— źródłowy 82
kompaktowa płyta laserowa 347, 348
kompansja 346, 350, 351
kompilacja 81
kompilator 82
— C 520
kompresja 15, 17, 197, 198
— danych 459, 460
— głosu 13
— JPEG 473
— LZW 466, 467, 470
— obrazu 13, 20
— sygnału 197
— transformatowa 461
kontrast 372, 373, 374
konwersja
— filtru 595
— współrzędnych biegunowych na prostokątne 531
— współrzędnych prostokątnych na biegunowe 160, 161, 531
koprocesor arytmetyczny 86
korelacja 124, 135, 156, 399, 401
— wzajemna 138
krzywa
— gamma 374
— Gaussa 270
— ROC 434, 435, 453
kwantyzacja 44, 45

L

liczba urojona 528
liczby zespolone 528, 531
liniowe kodowanie predykcyjne (LPC) 346, 354, 466
liniowość
— fazy 256
— przekształcenia Fouriera 184
— statyczna 96, 99

LSB (najmniej znaczący bit) 46
LSF 411
LZW 460

Ł

łączenie
— kaskadowe 584
— równoległe 584, 585
łączność
— dodawania 530
— operacji splotu 132
łagodny próg 442

M

mantysa 76
masa prawdopodobieństwa 32
maszyna
— korelacyjna 137, 138
— splotowa 117, 118, 119, 120
matryca
— rzeczywista 394
— urojona 394
metoda
— iteracyjna 264
— korelacyjna 156
— nakładania z dodawaniem 301
— nakładania z dodawaniem (overlap-add) 300
— o wielu częstotliwościach próbkowania 199
— równań równoważnych 156
— SOI 264
— symboliczna 537
— średniej kroczącej 264
metodyka przekształcenia Laplace'a 560
metody
— bezstratne 460
— stratne 460
MFLOPS 502
mikrofonowanie 171
miksowanie 349
MIMD (wiele instrukcji – wiele danych) 491
MIPS 502
mnożenie
— częstotliwości 250
— liczb zespolonych 532
mocne prawo wielkich liczb 28
model tworzenia obrazu 392
moduł 530
modulacja
— amplitudy 197, 200, 201
— delta 67
— osiem-do-czternastu (EFM) 348
modulator
— delta 68, 69
— delta ze zboczem zmiennym w sposób ciągły (CVSD) 69, 70
modyfikacja krawędzi 386
morfing 379
morfologiczne przetwarzanie obrazu 418
„motylek" FFT 225, 226
MPEG 460, 479
multipleksowanie 14
— w dziedzinie częstotliwości 202

N

najszybszy spadek 449
natężenia dźwięku 342
nerw ślimakowy 340
neurony 438
niedogodności współrzędnych biegunowych 162
niejednoznaczność 164
nieskończona odpowiedź impulsowa NOI 253
niezmienność względem przesunięcia 93, 95, 99
NTSC 371

O

obliczanie splotu 120
— odcinkami 243
obliczanie współczynników filtrów rekursywnych Czebyszewa 329
obrazowanie
— medyczne 12
— rastrowe 372

obraz
— cyfrowy 360, 361
— typu B 480
— typu I 479
— typu P 480
obszar zbieżności 563
odbitka głosu 354
odchylenie
— średnie 24
— standardowe 23, 24, 25, 26, 27, 28, 30, 36, 41, 493
— statystyczne 27
odkształcanie
— obrazów (warping) 378, 379
— okręgu w elipsę 593
odpowiedź
— częstotliwościowa 97, 175, 252
— fazowa 587
— impulsowa 110, 119, 122, 239, 252, 253, 275, 279
— krawędziowa 409, 410
— na impuls prostokątny 316
— skokowa 62, 252, 253
odstęp próbkowania 414
odtwarzanie
— JPEG 476
— sygnałów 252
odwrotna inwersja widmowa 260, 262, 263
odwrotne
— dyskretne przekształcenie Fouriera 154
— przekształcenie DFT 158, 211
— zespolone przekształcenie DFT 546
ogniskowa źrenicy 363
okienkowanie funkcją sinc 175, 264
okienko owalne 340
okno
— Bartletta 277
— Blackmana 173, 270, 274, 276
— Hamminga 168, 172, 173, 276
— Hanninga 277
— o płaskim wierzchołku 174
— prostokątne 172
oko 362
okrąg jednostkowy 578
oktawa 345

P

pakiet gaussowski 213
PAL 371
pamięć podręczna 87
pamiętanie rzędu zerowego 53
para linii 409
pary transformat Fouriera 206, 207
pasma Macha 383, 384
pasmo
— przepustowe 256
— zaporowe 256
pdf 29, 32
Pierre Simon de Laplace 142
pierwsza różnica 126
piksel 360, 374
plamka żółta 366
płaszczyzna
— s 579
— z 579
— zespolona 529
— zmiennej zespolonej 554
pmf 29, 32
pochodna dyskretna 126
podprogram „mityczny" 116
podręczna pamięć instrukcji 489
podstawowa koncepcja DSP 102
podwójna dokładność 76, 283
podziałka z parami linii 407, 408
pojedyncza dokładność 76
pomiary analogowe 516
potencjały czynnościowe 438
powielanie widma częstotliwościowego 224
poziomy szarości 362
półobraz nieparzysty 370
próbkowanie 44, 45
prawdopodobieństwo wykrycia celu 433
pręciki 365
precyzja 41, 42
procedura iteracyjna 446
procesory
— DSP 496
— SHARC 492, 494, 495, 499, 502, 510, 511, 515

— SHARC® DSP 487
— niestacjonarne 28
programy FFT 227
projekcja wsteczna 426
projektowanie filtru Czebyszewa-Butterwortha 590
przeciek widmowy 172
przekształcenie 146
— DCT 474, 475
— skali szarości 376
— biliniowe 593, 594
— DFT 209
— fazorowe 537
— Fouriera 144, 145, 244, 245, 357, 548, 551, 552
— Laplace'a 108, 528, 553, 554, 555, 556
— Z 108, 309, 528, 575, 578
przemienność
— mnożenia 530
— operacji splotu 132
— systemów liniowych 99
przerzut 62, 256
przesłona 363
przestrzeń
— dwuparametryczna 436
— parametryczna 437
przetwarzanie
— analogowo-cyfrowe 43, 44
— autonomiczne (off-line) 484, 486
— cyfrowo-analogowe 43, 44, 53, 54
— homomorficzne 294, 391
— o wielu częstotliwościach próbkowania (multirate) 55, 66
— sygnałów fonicznych 15, 340
— w czasie rzeczywistym 484
przetwornik
— analogowo-cyfrowy 44
— analogowo-cyfrowy delta-sigma 71
— delta-sigma 70, 72
pulsacja 149, 245

R

radar 12, 13, 17, 124, 136
radian 149, 162, 579
ramki 370
rdzeń 506
rekonstrukcja
— Fouriera 427
— nagrań fonograficznych 294
rekursja 271
rezonans magnetyczny 13, 19, 429
rozciąganie skali szarości 376
rozdzielanie sygnałów 252
rozdzielczość
— mnożenia względem dodawania 530
— przestrzenna 406
— splotu względem dodawania 133
— widma częstotliwościowego 172
rozkład 101
— FFT 223
— Fouriera 106, 107, 142, 143
— Gaussa 35, 36, 39, 134
— gęstości prawdopodobieństwa 38
— impulsowy 103, 104
— normalny 35
— parzysto-nieparzysty 103, 105, 235
— skokowy 103, 104
— statystyczny Poissona 417
— z przeplotem 106, 222
rozmieszczenie
— biegunów 572, 592
— zer i biegunów 573
rozplot 177, 264, 286, 290, 291, 292, 355
rozpoznawanie mowy 16
rozwijanie fazy 164, 165
rozwinięty moduł 165
równania
— analizy 245, 247
— równania rekurencyjne 128
— równania syntezy 152, 244, 247
równoważność matematyczna 533, 540
rzeczywiste
— DFT 142, 146, 220, 221, 232, 542
— FFT 234
— odwrotne przekształcenie DFT 221
— przekształcenie Fouriera 537, 544

S

SECAM 371
sejsmologia odbiciowa 18

siatka próbkowania 362
siatkówka 365, 366
sieci neuronowe 431
SIMD (pojedyncza instrukcja – wiele danych) 491
sinusoidy dwuwymiarowe 397
skala szarości 360, 372
skaner CAT 423
skanowanie liniowe 372
składowe sygnału wejściowego 101
skok jednostkowy 254
skończona odpowiedź impulsowa SOI 253
skrzypce 345
słuch ludzki 340
sonar 12, 13, 17
— pasywny 171
sortowanie z odwróceniem bitowym 223, 224
specjalistyczne oprogramowanie aplikacyjne 81
spektrogram mowy 354
splot 109, 110, 111, 113, 114, 115, 175, 240, 291, 382
— ciągły 243
— FFT 299, 300, 302, 303, 306, 398
— kołowy 180, 181
— obrazowy 401, 402
— od strony wejścia 240
— od strony wyjścia 241
— przez dziedzinę częstotliwości 177
— przez rozłączność 388
— w dziedzinie częstotliwości 179
spłaszczanie oświetlenia 390
stała czasowa filtrów jednobiegunowych 310
statystyka
— bieżąca 25
— Poissona 416
sterownik wejścia-wyjścia 489
stosunek sygnału do szumu (SNR) 26, 414, 415, 493
stromość spadku charakterystyki 59
studnia potencjału 369
suma bieżąca 126, 127
sygnał 92
— ciągły 22, 92
— docelowy 138
— dyskretny 22, 92
— liniowofazowy 131
— mieszany 489
— mowy 352, 353, 356
— nieliniowofazowy 131
— nieparzysty 193
— odcinkowo-wielomianowy 243
— parzysty 193
— przyczynowy 130
— świergotowy 217
— zerofazowy 131
symetria kołowa 105
symulacja organu głosowego 16
symulatory 525
synteza 101
— DTFT 203
— FFT 224
— mowy 340
— i rozkład sygnału 101
— i rozpoznawanie mowy 351
system 92
— bezpamięciowy 96
— liniowy 92, 98
— nieliniowy 98
— przyczynowy 130
— wbudowany (embedded) 495
— dyskretny 92
— liniowy 93
— świergotowy 217, 218
szereg Fouriera 144, 145, 244, 246, 247, 248, 551
szerokopasmowy dostępem do pamięci 489
szerokość w połowie wysokości (FWHM) 407
szkieletowanie 419
— binarne 420
sztuczny pogłos 15
szum 293
— 1/f 170
— biały 170
— kwantyzacji 493
— losowy 25
— obrazowy 416

— statystyczny 27
— zaokrąglania 519
szybkie przekształcenie Fouriera (FFT) 178, 220
szybki splot 302
szybkość
— Nyquista 49
— opadania 332
— wykonania 81, 85, 89

Ś

średnica źrenicy 363

T

tablica
— kodowa 467
— przeglądowa (LUT) 89
technika
— iteracji metodą najmniejszych kwadratów (ILST) 424
— rekonstrukcji algebraicznej (ART) 424, 425
— równoczesnej rekonstrukcji iteracyjnej (SIRT) 424
telekomunikacja głosowa 340
telewizja wysokiej jakości 360
teoria
— czasu 341
— miejsca 340
terminologia DFT 147
Texas Instruments 507
TIC55xx 87
tłumienie
— echa 15
— szumu 267
— w paśmie zaporowym 332
tłumik odwracający 112
tomografia komputerowa 406, 411, 422, 423, 424
tomograf komputerowy 19
transformata 146
— Fouriera 55
— DFT 158
transformator Hilberta 587
trenowanie sieci neuronowych 444
twierdzenie
— Fouriera o projekcji 427, 428
— Nyquista 48
— o próbkowaniu 48, 52, 412
— Shannona 48

U

ucho
— środowe 340
— zewnętrzne 340
układ
— z pętlą synchronizacji fazowej (PLL) 97
— CCD 370
— próbkująco-pamiętający 44
— Sallena-Keya 56, 57, 58, 570, 571, 574
— ze sprzężeniem ładunkowym (CCD) 368
— ze sprzężeniem ładunkowym CCD 367

W

wagi sieci neuronowej 452
wariacja 24
warstwa
— ukryta 438, 450
— wejściowa 438
— wyjściowa 438, 450
wartość
— międzyszczytowa 24, 25
— rms 24
— średnia 23, 24, 25, 26, 27, 28, 30, 36, 41
— średniokwadratowa 24
— zespolona sprzężona 191
węzeł aktywny 440
widmo
— częstotliwościowe 171
— DFT 209
wieloprocesorowość 505
wierność odtwarzania sinusoidy 96, 97, 99, 144
współczynnik
— rekursywny 253, 309, 580
— kształtu obrazu 371
— rozrzutu (CV) 26
— tłumienia 256
— wagowy 122

współrzędne
— biegunowe 160, 166
— prostokątne 160, 166
wstęga boczna
— dolna 202
— górna 202
wybieranie międzyliniowe 370
wygładzanie krawędzi 394
wyjściowa tablica przeglądowa 374
wyodrębnianie cech 438
wyostrzanie krawędzi 387, 394
wyrównywanie histogramu 377
wysokość 343, 344
wzbranianie boczne 382
wzmacniacz ładunkowy 369
wzmocnienie filtrów 589
wzór rekursywny 308

Z

zafalowania w paśmie przepustowym 61
zakres
— dynamiczny amplitudy 334
— dynamiczny częstotliwości 334
zależność
— Eulera 532, 543
— Parsevala 204
zapis
— binarny przesunięty 75, 76
— liczb całkowitych bez znaku 74
— o pojedynczej dokładności 283
— stałoprzecinkowy 74
— uzupełnienia do 2 75
— zmiennoprzecinkowy 74, 76, 77
— znak-moduł 75
— z podwójną dokładnością 77
zasada
— superpozycji 101
— μ255 lub A 350, 351
zastosowania DFT 167
zawartość harmonicznych 343
zbieżność 449
zbiorowa funkcja rozkładu (cdf) 36, 37
zera 563, 567, 568
zespolone
— DFT 142, 220, 221, 232, 544
— odwrotne przekształce DFT 221
— przekształcenie Fouriera 528, 533, 537, 541, 542
— widmo częstotliwościowe 546
zestaw
— EZ-KIT 513, 510
— EZ-KIT Lite 511, 512
ziarno 40
zwielokrotnianie częstotliwości 349

Pod maską SPICE'a – metody i algorytmy analizy układów elektronicznych



W książce przedstawiono obszerny opis języka symulacyjnego SPICE, który jest jednym z podstawowych narzędzi komputerowej analizy obwodów elektronicznych, zarówno analogowych, jak i cyfrowych. Po wprowadzeniu, zawierającym teoretyczne podstawy algorytmów stosowanych w języku SPICE, zaprezentowano opis podstawowych rodzajów analiz dostępnych w programie i przykłady ich użycia.

Książka jest adresowana do studentów wydziałów elektronicznych i elektrycznych wyższych uczelni technicznych oraz do wszystkich elektroników zainteresowanych komputerową analizą układów elektronicznych.

Autor: Andrzej Dobrowolski
Format: B5, 232 str. (twarda oprawa)
Cena: 54,00 zł

Projektowanie układów analogowych. Poradnik praktyczny



Książka równie niepowtarzalna jak jej autor. Jest to jedyny w swoim rodzaju przewodnik po tajnikach projektowania układów analogowych, napisany zgodnie z duchem doświadczonego praktyka, którego podstawowym celem jest zrozumieć dlaczego coś nie działa tak jak powinno.

Fragment z książki:

„(...) Dlaczego interesuje nas usuwanie usterek? Dlatego, że nawet najlepsi inżynierowie podejmują się wykonywania projektów, tak trudnych i stanowiących tak duże wyzwania, że opracowane układy często nie działają tak jak miały działać – a przynajmniej nie od razu(...)"

Autor: Robert A. Pease
Format: B5, 270 str. (twarda oprawa)
Cena: 56,00 zł

Wzmacniacze operacyjne



Książka jest kompendium wiedzy na temat wzmacniaczy operacyjnych, w którym szczególny nacisk położono na zagadnienia praktyczne, przede wszystkim: projektowanie filtrów aktywnych i generatorów przebiegów funkcyjnych, źródła szumów w układach analogowych, omówienie znaczenia parametrów charakteryzujących wzmacniacze operacyjne, a także dobór wartości elementów dla wybranych aplikacji. Niezbędna porcja rozważań teoretycznych jest jedynie uzupełnieniem wiadomości praktycznych, które powinien posiadać każdy elektronik zajmujący się układami analogowymi. Z tego powodu książka powinna być „niezbędnikiem" zarówno w profesjonalnych pracowniach konstrukcyjnych, jak i w domowych laboratoriach elektronicznych.

Autor: Piotr Górecki
Format: B5, 252 str. (twarda oprawa)
Cena: 43,00 zł