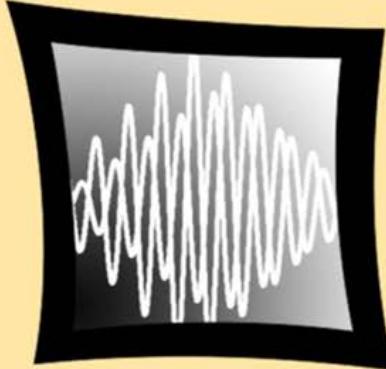




А. Б. Сергиенко

# Цифровая обработка сигналов

3-е издание



bhv®



**А. Б. Сергиенко**

# **Цифровая обработка сигналов**

**3-е издание**

Рекомендовано Учебно-методическим объединением вузов Российской Федерации по образованию в области радиотехники, электроники, биомедицинской техники и автоматизации в качестве учебного пособия для студентов высших заведений, обучающихся по направлению 210300 «Радиотехника»

Санкт-Петербург  
«БХВ-Петербург»  
2011

УДК 681.3.06(075.8)

ББК 32.973(я73)

C32

## Сергиенко А. Б.

С32      Цифровая обработка сигналов: учеб. пособие. — 3-е изд. — СПб.: БХВ-Петербург, 2011. — 768 с.: ил. — (Учебная литература для вузов)

ISBN 978-5-9775-0606-9

Учебное пособие представляет собой базовый курс по цифровой обработке сигналов. Изложены основы теории дискретных сигналов и систем, рассмотрены методы спектрального анализа и фильтрации дискретных сигналов, алгоритмы синтеза дискретных фильтров, влияние эффектов квантования и конечной точности вычислений на работу цифровых устройств, описаны методы модуляции, применяемые для передачи цифровой информации, обсуждаются адаптивные фильтры и многоскоростная обработка сигналов. Вводные главы посвящены основам анализа сигналов и теории аналоговых систем. Материал изложен так, чтобы наглядно продемонстрировать сущность алгоритмов, их взаимосвязь и области применения. Теоретические сведения сопровождаются примерами реализации обсуждаемых алгоритмов с помощью системы MATLAB и ее пакетов расширения Signal Processing, Communications, Filter Design и Fixed-Point. Третье издание книги отражает изменения, произошедшие в последних версиях MATLAB (вплоть до Release 2010a).

*Для студентов технических вузов, инженеров и специалистов,  
работающих в области обработки сигналов*

УДК 681.3.06(075.8)

ББК 32.973(я73)

## Группа подготовки издания:

Главный редактор	Екатерина Кондукова
Зам. главного редактора	Евгений Рыбаков
Зав. редакцией	Григорий Добин
Редактор	Юрий Рожко
Компьютерная верстка	Ольги Сергиенко
Корректор	Зинаида Дмитриева
Дизайн серии	Инны Тачиной
Оформление обложки	Елены Беляевой
Фото	Кирилла Сергеева
Зав. производством	Николай Тверских

## РЕЦЕНЗЕНТЫ:

- А. И. Солонина, канд. техн. наук, доц., проф. кафедры цифровой обработки сигналов  
Санкт-Петербургского государственного университета телекоммуникаций  
им. проф. А. М. Бонч-Бруевича
- Э. Л. Муро, канд. техн. наук, доц. кафедры радиоприемных устройств Московского  
энергетического института (Технического университета)

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.10.10.

Формат 70×100<sup>1</sup>/16. Печать офсетная. Усл. печ. л. 61,92.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию  
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой  
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов  
в ГУП "Типография "Наука"  
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0606-9

© Сергиенко А. Б., 2010

© Оформление, издательство "БХВ-Петербург", 2010

# Оглавление

<b>Введение .....</b>	<b>1</b>
Структура книги.....	2
Чего нет в этой книге.....	5
Благодарности.....	5
Информация от компании The MathWorks, Inc.....	6
От издательства.....	6
<b>Глава 1. Основы анализа сигналов .....</b>	<b>7</b>
Классификация сигналов .....	8
Энергия и мощность сигнала .....	11
Ряд Фурье .....	12
Синусно-косинусная форма .....	13
Вещественная форма .....	14
Комплексная форма .....	14
Примеры разложения сигналов в ряд Фурье .....	16
Преобразование Фурье.....	22
Примеры расчета преобразования Фурье .....	24
Свойства преобразования Фурье .....	33
Фурье-анализ неинтегрируемых сигналов.....	37
Корреляционный анализ .....	40
Корреляционная функция .....	40
Взаимная корреляционная функция .....	42
Связь между корреляционными функциями и спектрами сигналов .....	44
Энергетические расчеты в спектральной области .....	45
Комплексная огибающая.....	45
Преобразование Гильберта .....	47
Спектр аналитического сигнала .....	48
Пространство сигналов .....	50
Метрическое пространство .....	50
Линейное пространство.....	51
Нормированное линейное пространство .....	52
Пространство со скалярным произведением.....	53
Дискретные представления сигналов.....	53
Интегральные представления сигналов .....	56

Случайные сигналы .....	58
Ансамбль реализаций .....	58
Модели случайных процессов .....	59
Вероятностные характеристики случайных процессов .....	61
Корреляционные функции случайных процессов .....	66
Стационарные и эргодические случайные процессы .....	69
Спектральные характеристики случайных процессов .....	73
Теорема Винера — Хинчина .....	75
Узкополосный случайный процесс .....	78
<b>Глава 2. Аналоговые системы .....</b>	<b>85</b>
Классификация систем .....	85
Характеристики линейных систем .....	86
Импульсная характеристика .....	86
Переходная характеристика .....	87
Условие физической реализуемости .....	87
Комплексный коэффициент передачи .....	88
Коэффициент передачи по мощности .....	88
Фазовая и групповая задержка .....	88
Взаимный спектр выходного и входного сигналов .....	89
Взаимная корреляция между входом и выходом .....	89
Преобразование случайного процесса в линейной системе .....	90
Спектральная плотность мощности .....	90
Корреляционная функция .....	90
Дисперсия .....	90
Плотность вероятности .....	91
Частный случай белого шума .....	91
Способы описания линейных систем .....	91
Дифференциальное уравнение .....	92
Функция передачи .....	92
Нули и полюсы .....	93
Полюсы и вычеты .....	95
Пространство состояний .....	98
Функции MATLAB для расчета линейных цепей .....	101
Расчет частотных характеристик .....	101
Построение графиков фазочастотных характеристик .....	103
Построение голографа функции передачи .....	104
Преобразование способов описания линейных цепей .....	105
Расчет аналоговых фильтров-прототипов .....	109
Частотные преобразования фильтров .....	122
Расчет аналоговых фильтров .....	130
Выбор порядка фильтра .....	132
Расчет групповой задержки .....	134
<b>Глава 3. Дискретные сигналы .....</b>	<b>138</b>
Аналоговые, дискретные и цифровые сигналы .....	138
Аналого-цифровое и цифроаналоговое преобразование .....	139
Частота Найквиста .....	140
Спектр дискретного сигнала .....	141
Влияние формы дискретизирующих импульсов .....	147

Теорема Котельникова .....	149
Восстановление радиосигнала по отсчетам видеосигнала .....	153
Квадратурная дискретизация узкополосных сигналов .....	154
Субдискретизация сигнала .....	155
Z-преобразование .....	159
Примеры вычисления z-преобразования .....	159
Связь z-преобразования с преобразованиями Лапласа и Фурье .....	161
Свойства z-преобразования .....	162
Обратное z-преобразование .....	164
Пространство дискретных сигналов .....	165
Дискретные случайные сигналы .....	165
Корреляционная матрица .....	166
Дискретный белый шум .....	167
Дискретные сигналы в MATLAB .....	167
Расчет временных функций .....	168
Функции генерации одиночных импульсов .....	172
Генерация последовательности импульсов .....	179
Функции генерации периодических сигналов .....	182
Генерация сигнала с меняющейся частотой .....	186
Формирование случайных сигналов .....	189
Получение данных из внешних источников .....	194
Чтение WAV-файлов .....	195
Запись WAV-файлов .....	199
Воспроизведение звука .....	199
Запись звука .....	201
Готовые записи сигналов .....	202
Пакет расширения Data Acquisition .....	203
<b>Глава 4. Дискретные системы .....</b>	<b>206</b>
Сущность линейной дискретной обработки .....	206
Способы описания дискретных систем .....	210
Импульсная характеристика .....	210
Функция передачи .....	212
Нули и полюсы .....	213
Полюсы и вычеты .....	218
Пространство состояний .....	220
Фильтры первого и второго порядка .....	221
Фильтры первого порядка .....	221
Условие устойчивости для систем второго порядка .....	224
Резонатор второго порядка .....	225
Режектор второго порядка .....	230
Преобразование случайного сигнала в дискретной системе .....	232
Рекурсивные и нерекурсивные дискретные фильтры .....	234
Нерекурсивные фильтры .....	234
Рекурсивные фильтры .....	237
Формы реализации дискретных фильтров .....	238
Каноническая форма .....	239
Транспонированная форма .....	240
Последовательная (каскадная) форма .....	242
Параллельная форма .....	243

Дискретная фильтрация в MATLAB .....	244
Дискретная свертка.....	244
Обращение свертки.....	245
Функция дискретной фильтрации .....	245
Доступ к внутреннему состоянию фильтра .....	246
Компенсация фазового сдвига .....	247
Расчет импульсной характеристики .....	249
Расчет переходной характеристики.....	250
Расчет частотных характеристик.....	250
Отображение нулей и полюсов фильтра.....	258
Свертка как матричное умножение .....	259
Преобразование способов описания дискретных фильтров .....	260
Объекты дискретных фильтров .....	266
Функции расчета резонаторов второго порядка .....	274
Некоторые идеализированные фильтры.....	275
Дискретное преобразование Гильберта .....	276
Идеальный дифференцирующий фильтр .....	280
Идеальный фильтр задержки .....	282
Визуализатор фильтров.....	285
<b>Глава 5. Спектральный анализ.....</b>	<b>287</b>
Дискретное преобразование Фурье.....	288
Свойства дискретного преобразования Фурье .....	289
Восстановление непрерывного сигнала с помощью ДПФ .....	291
Матрица ДПФ .....	292
Связь ДПФ и спектра дискретного сигнала.....	293
Алгоритм быстрого преобразования Фурье .....	294
БПФ с прореживанием по времени .....	295
БПФ с прореживанием по частоте .....	300
Основание алгоритма БПФ .....	302
Вычислительные затраты при произвольной размерности БПФ.....	303
Выводы .....	304
Взаимосвязь ДПФ и фильтрации .....	304
ДПФ как дискретная фильтрация .....	304
Дискретная фильтрация с помощью ДПФ .....	308
"Идеальная" фильтрация путем обнуления части спектральных отсчетов.....	313
Растекание спектра .....	314
Весовые функции .....	318
Спектр дискретного случайного процесса .....	319
Непараметрические методы.....	320
Периодограмма .....	320
Метод Уэлча.....	321
Параметрические методы .....	322
Авторегрессионная модель .....	322
Метод MUSIC.....	335
Метод EV .....	337
Функции спектрального анализа в MATLAB.....	337
Прямое и обратное ДПФ .....	338
Функция <i>fftsift</i> .....	338

Блоковая фильтрация в частотной области .....	339
Окна .....	339
Функции непараметрического спектрального анализа.....	352
Параметрический спектральный анализ в MATLAB.....	359
Реализация метода MUSIC.....	366
Реализация метода EV .....	369
Хранение и отображение спектральных данных.....	369
<b>Глава 6. Проектирование дискретных фильтров .....</b>	<b>371</b>
Синтез рекурсивных фильтров по аналоговому прототипу.....	371
Метод билинейного $z$ -преобразования.....	372
Метод инвариантной импульсной характеристики.....	373
Прямые методы синтеза .....	376
Оптимальные методы .....	376
Минимизация квадратичной ошибки ( $p = 2$ ).....	377
Минимаксная оптимизация ( $p = \infty$ ) .....	379
Субоптимальные методы .....	383
Субоптимальный синтез нерекурсивных фильтров.....	383
Синтез с использованием окон .....	384
Фильтры с косинусоидальным сглаживанием АЧХ .....	389
Синтез дискретных фильтров в MATLAB.....	393
Функции, использующие билинейное $z$ -преобразование .....	396
Функция <i>impinvvar</i> .....	401
Функции прямого синтеза рекурсивных фильтров .....	402
Функции синтеза с использованием окон.....	408
Функции расчета ФНЧ с косинусоидальным сглаживанием .....	412
Функция расчета рекурсивного фильтра Гильберта .....	415
Функции минимизации среднеквадратической ошибки.....	415
Реализация метода Ремеза.....	422
Функции пакета Filter Design .....	429
Графическая среда для синтеза и анализа фильтров .....	433
Объекты спецификаций фильтров и среда FilterBuilder .....	443
<b>Глава 7. Эффекты квантования в цифровых системах.....</b>	<b>446</b>
Форматы представления чисел.....	446
Представление отрицательных чисел .....	447
Формат с фиксированной запятой.....	448
Формат с плавающей запятой .....	450
Процесс квантования.....	453
Шум квантования.....	453
Неравномерное квантование .....	455
Эффекты квантования в цифровых фильтрах .....	456
Квантование коэффициентов цифровых фильтров.....	456
Масштабирование коэффициентов цифровых фильтров .....	460
Переполнение разрядной сетки в процессе вычислений .....	462
Округление промежуточных результатов вычислений .....	465
Аналитическая модель собственного шума в фильтрах с фиксированной запятой.....	466
Пределевые циклы.....	470

Учет эффектов конечной точности вычислений в MATLAB .....	473
Функции квантования.....	473
Объекты квантователей.....	477
Квантованные фильтры .....	485
Анализ предельных циклов.....	493
Расширение программы FDA Tool.....	494
Квантованное БПФ .....	497
<b>Глава 8. Модуляция и демодуляция.....</b>	<b>502</b>
Амплитудная модуляция .....	503
Однотональная АМ .....	505
АМ-сигнал в общем случае.....	509
Энергетические соотношения в АМ-сигнале .....	511
Демодуляция АМ .....	512
Разновидности амплитудной модуляции .....	514
АМ с подавленной несущей.....	514
Однополосная модуляция .....	516
Полярная модуляция.....	520
Угловая модуляция .....	524
Фазовая и частотная модуляция .....	524
Гармоническая угловая модуляция .....	526
Спектр сигнала с гармонической угловой модуляцией.....	528
Ширина спектра сигнала с гармонической УМ .....	532
Демодуляция УМ .....	536
Квадратурная модуляция .....	538
Спектр сигнала с квадратурной модуляцией.....	539
Демодуляция сигнала с квадратурной модуляцией .....	539
Способы модуляции, используемые при передаче цифровой информации .....	541
Частотная манипуляция.....	542
Амплитудная манипуляция .....	547
Фазовая манипуляция .....	547
Квадратурная модуляция.....	548
Широтно-импульсная модуляция.....	558
Функции модуляции и демодуляции пакета Signal Processing.....	562
Амплитудная модуляция .....	563
АМ с подавленной несущей.....	565
Однополосная модуляция .....	565
Фазовая модуляция .....	565
Частотная модуляция.....	566
Квадратурная модуляция.....	567
Широтно-импульсная модуляция .....	567
Времяимпульсная модуляция .....	568
Функции модуляции и демодуляции пакета Communications .....	569
Аналоговая модуляция .....	569
Цифровая модуляция .....	578
<b>Глава 9. Адаптивные фильтры .....</b>	<b>591</b>
Основные понятия адаптивной обработки сигналов .....	592
Оптимальный фильтр Винера.....	593

Градиентный поиск оптимального решения .....	598
Адаптивный алгоритм LMS .....	599
Детерминированная задача оптимальной фильтрации.....	601
Адаптивный алгоритм RLS.....	602
Экспоненциальное забывание.....	606
Применение адаптивных фильтров.....	607
Идентификация систем.....	607
Линейное предсказание.....	607
Подавление шума.....	608
Выравнивание частотной характеристики канала связи .....	609
Эхоподавление .....	610
Объекты адаптивной фильтрации пакета Filter Design.....	611
Примеры реализации адаптивной фильтрации .....	617
Идентификация системы .....	617
Линейное предсказание .....	619
Шумоподавление .....	620
Компенсация искажений, вносимых каналом связи .....	622
<b>Глава 10. Многоскоростная обработка сигналов .....</b>	<b>624</b>
Изменение частоты дискретизации.....	624
Прореживание .....	625
Интерполяция.....	626
Передискретизация .....	628
Многокаскадная реализация прореживания и интерполяции.....	629
Структуры "интегратор — гребенчатый фильтр".....	631
Полифазные структуры .....	632
Идея полифазного представления сигналов .....	632
Полифазная реализация процесса интерполяции.....	634
Полифазная реализация процесса прореживания .....	635
Банки фильтров.....	638
Банк анализа .....	639
Банк синтеза .....	643
Функции изменения частоты дискретизации в MATLAB.....	647
Функции изменения частоты дискретизации .....	647
Функция прореживания.....	648
Функция интерполяции .....	650
Функции передискретизации .....	652
<b>Приложение 1. Основы работы с MATLAB.....</b>	<b>655</b>
Установка .....	655
Работа в интерактивном режиме .....	656
Справочная система.....	661
Интерфейс главного окна .....	662
Массивы .....	664
Другие типы данных .....	667
Многомерные массивы.....	667
Строки.....	668
Структуры.....	669
Массивы ячеек .....	670

Программирование .....	670
Программы и функции .....	671
Редактор/отладчик M-файлов .....	672
Создание функций .....	672
Путь поиска .....	674
Логические условия .....	675
Условный оператор .....	675
Оператор выбора.....	676
Циклы.....	677
Функции с переменным числом параметров.....	678
Ввод и вывод данных .....	679
Работа с отладчиком .....	681
Оптимизация MATLAB-программ.....	682
Графика.....	689
Двумерная графика .....	689
Трехмерная графика .....	692
Настройка внешнего вида графиков .....	696
Одновременный вывод нескольких графиков.....	700
Дальнейшее использование графиков.....	702
Дополнительные источники информации .....	703
<b>Приложение 2. Обзор функций MATLAB .....</b>	<b>704</b>
Audiovideo .....	704
Datafun .....	704
Datatypes .....	705
Demos.....	705
Elfun .....	706
Elmat.....	707
Funfun.....	709
General .....	709
Graph2d, Graph3d .....	709
Graphics.....	709
Guide .....	709
Iofun .....	710
Lang.....	710
Matfun .....	710
Ops.....	711
Polyfun .....	711
Sparfun .....	712
Specfun .....	712
Specgraph .....	713
Strfun .....	713
Timefun .....	713
Timeseries.....	714
Uitools .....	714
Winfun .....	714
<b>Приложение 3. Компоненты MATLAB .....</b>	<b>715</b>
MATLAB .....	715
Пакеты расширения MATLAB .....	715
Математика и оптимизация.....	715

Статистика и анализ данных .....	716
Анализ и синтез систем управления .....	716
Обработка сигналов и телекоммуникации.....	717
Обработка изображений.....	717
Тесты и измерения .....	717
Вычислительная биология .....	718
Финансовое моделирование и анализ .....	718
Разработка приложений .....	718
Целевые платформы для разработки приложений.....	719
Связь с базами данных и генерирование отчетов .....	719
<b>Simulink.....</b>	<b>719</b>
Наборы блоков Simulink.....	720
Физическое моделирование .....	720
Имитационная графика .....	721
Анализ и синтез систем управления .....	721
Обработка сигналов и телекоммуникации.....	721
Генерирование исполняемого кода .....	722
Быстрое прототипирование и аппаратно-программное моделирование (Hardware-in-the-Loop).....	722
Верификация, проверка и тестирование моделей .....	722
<b>Приложение 4. Программа SPTool.....</b>	<b>724</b>
Загрузка сигнала .....	725
Просмотр графика сигнала .....	726
Спектральный анализ сигнала .....	727
Расчет фильтра.....	728
Просмотр характеристик фильтра.....	728
Фильтрация сигнала .....	728
Сохранение результатов работы.....	729
<b>Литература .....</b>	<b>731</b>
Радиотехника.....	731
Цифровая обработка сигналов .....	732
MATLAB .....	734
Разное .....	734
<b>Предметный указатель .....</b>	<b>736</b>

# Введение

Цифровая обработка сигналов (*ЦОС*; английский термин — *Digital Signal Processing, DSP*) как направление развития науки и техники зародилась в 1950-х годах и поначалу представляла собой довольно экзотическую отрасль радиоэлектроники, практическая ценность которой была далеко не очевидной. Однако за прошедшие пятьдесят лет благодаря успехам микроэлектроники системы цифровой обработки сигналов не только воплотились в реальность, но и вошли в нашу повседневную жизнь в виде CD- и DVD-проигрывателей, модемов, сотовых телефонов и многое другое. Более того, в некоторых прикладных областях цифровая обработка сигналов стала вытеснять "традиционную" (анalogовую). Это уже произошло в аудиотехнике, в настоящее время интенсивно идет процесс перехода телевизионного вещания на цифровую основу.

Бурное развитие цифровых технологий во многом изменило как смысл самого понятия "радиотехника", так и требования, предъявляемые к подготовке специалистов в этой области, сделав необходимыми новые знания и умения.

Для приобретения знаний нужна соответствующая литература, причем для каждого читателя своя — в соответствии с его потребностями, интересами и даже темпераментом. Данное учебное пособие является попыткой *понятно объяснить* теоретические основы базовых алгоритмов цифровой обработки сигналов, продемонстрировать их сущность, взаимосвязь и области применения. Иногда это делается, и вполне сознательно, в некоторый ущерб математической строгости, чтобы обилие математических деталей не заслоняло сути вещей. Насколько удачным оказался найденный баланс, судить читателю.

Книга предназначена для широкого круга читателей — студентов, преподавателей, научных работников, программистов и вообще всех, кто интересуется компьютерной обработкой сигналов и иных данных. Требуемая для понимания материала математическая подготовка примерно соответствует первому курсу технического вуза.

В качестве средства создания иллюстрационных примеров выбрана система MATLAB, созданная фирмой The MathWorks, Inc. и являющаяся мировым стандартом в области научных и технических расчетов. Базовая библиотека MATLAB, а также пакеты расширения Signal Processing, Filter Design, Communications и Fixed-

Point содержат большое количество функций, позволяющих легко и быстро осуществлять разнообразные расчеты, связанные с цифровой обработкой сигналов. Для читателей, не знакомых с этой системой, в *приложении 1* приводится краткое описание принципов работы с ней.

Вопросы, касающиеся MATLAB и его пакетов расширения, изложены применительно к версиям программного обеспечения, входящим в весенний выпуск 2010 г. (Release 2010a):

- MATLAB 7.10;
- Signal Processing Toolbox 6.13;
- Filter Design Toolbox 4.7;
- Communications Toolbox 4.5;
- Fixed-Point Toolbox 3.1.

## Структура книги

Книга состоит из десяти глав и четырех приложений. Ее тематический охват, взаимосвязь отдельных глав и их соотношение с темами, не затронутыми в данном учебном пособии, наглядно иллюстрируются рис. В1. Толстая пунктирная линия показывает те границы, в пределах которых сосредоточены обсуждаемые в книге вопросы. Тонкие пунктирные линии объединяют разделы, входящие в родственные или смежные области, так или иначе связанные с ЦОС. Схема никоим образом не претендует на абсолютную полноту, ее цель — показать место, занимаемое рассматриваемыми темами книги в общей структуре разделов ЦОС.

Далее приведен краткий обзор содержания книги.

*Главы 1 и 2*, "Основы анализа сигналов" и "Аналоговые системы", являются вводными и посвящены не цифровым, а аналоговым сигналам и системам. Поскольку методы анализа цифровых и аналоговых сигналов и систем тесно связаны друг с другом, хорошее понимание обсуждаемых в этих главах вопросов необходимо для освоения материала последующих глав.

В *главе 3*, "Дискретные сигналы", рассматриваются принципы математического описания и анализа числовых последовательностей, которые и являются дискретными сигналами. Здесь же обсуждается разница между понятиями аналоговых, дискретных и цифровых сигналов.

В *главе 4*, "Дискретные системы", изложены принципы линейной обработки дискретных сигналов (дискретной фильтрации), рассмотрены способы описания дискретных систем и формы их реализации.

*Глава 5*, "Спектральный анализ", как явствует из ее названия, посвящена вопросам спектрального анализа дискретных сигналов. Здесь рассматриваются дискретное преобразование Фурье и быстрые алгоритмы его вычисления, способы реализации фильтрации в частотной области, а также непараметрические и параметрические методы оценивания спектра дискретного случайного процесса.

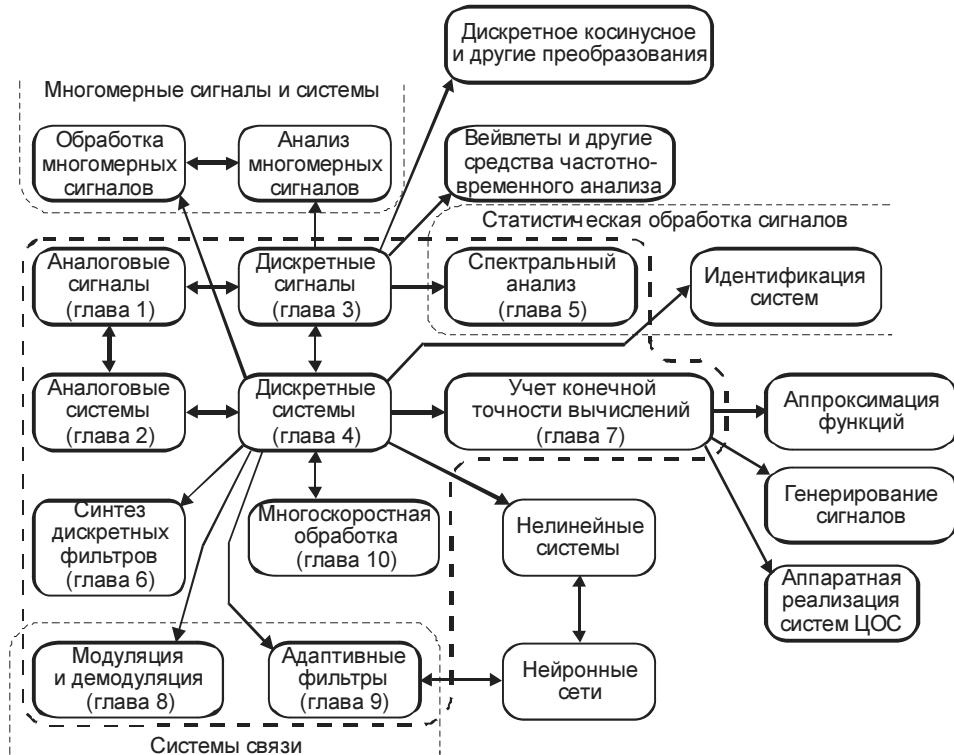


Рис. В1. Основные разделы цифровой обработки сигналов

В главе 6, "Проектирование дискретных фильтров", рассматриваются методы синтеза дискретных фильтров, т. е. способы расчета дискретных систем, удовлетворяющих заданным требованиям. Обсуждаются как методы, связанные с использованием аналоговых фильтров-прототипов, так и прямые методы, основанные на решении оптимизационных задач.

Глава 7, "Эффекты квантования в цифровых системах", посвящена эффектам, возникающим при практической реализации алгоритмов цифровой обработки сигналов из-за конечной точности представления чисел в вычислительных устройствах. Здесь рассматриваются такие вопросы, как шум квантования, округление коэффициентов и промежуточных результатов вычислений в цифровых фильтрах, возможности возникновения переполнения при вычислениях, а также предельные циклы.

В главе 8, "Модуляция и демодуляция", речь идет о способах преобразования сигналов для их передачи по каналам связи. Рассматриваются методы модуляции, используемые для передачи как аналоговых сигналов (амплитудная, угловая и квадратурная модуляция), так и цифровой информации (амплитудная, частотная, фазовая и квадратурная манипуляция).

Глава 9, "Адаптивные фильтры", посвящена устройствам, автоматически подстраивающим свои параметры под статистические свойства обрабатываемого сигнала. Такие фильтры получили широкое распространение, прежде всего, в системах циф-

ровой связи. В данной главе рассматриваются вопросы оптимальной фильтрации, а также два наиболее известных алгоритма адаптации — *LMS* (Least Mean Square, метод наименьшего квадрата, в отечественных источниках иногда используется аббревиатура МНК) и *RLS* (Recursive Least Squares, рекурсивный метод наименьших квадратов, в отечественных источниках — аббревиатура РНК).

Наконец, в *главе 10*, "Многоскоростная обработка сигналов", речь идет о системах, изменяющих частоту дискретизации сигнала. Рассматриваются алгоритмы прореживания и интерполяции сигнала, а также обсуждаются принципы построения банков фильтров.

Три из четырех приложений посвящены системе MATLAB. В *приложении 1* изложены основы работы с MATLAB, *приложение 2* содержит краткий обзор базовой библиотеки функций MATLAB, а в *приложении 3* перечислены компоненты MATLAB.

В *приложении 4* приводится описание графической среды *SPTool* (Signal Processing Tool), имеющейся в одном из рассматриваемых в книге пакетов расширения MATLAB — Signal Processing Toolbox. Эта программа позволяет просматривать графики сигналов, производить спектральный анализ, рассчитывать дискретные фильтры и осуществлять фильтрацию сигнала. Данные операции относятся к тематике *глав 3—6*, поэтому, чтобы не разбивать описание программы SPTool на несколько частей, соответствующая информация помещена в отдельное приложение.

Во всех главах, начиная со второй, важное место отводится описанию функций MATLAB, реализующих рассматриваемые алгоритмы. Как правило, в первой половине главы содержится теоретическая информация, а вторая половина посвящена соответствующим средствам MATLAB. При этом демонстрационные примеры в теоретической части реализованы в основном с помощью общематематических функций MATLAB, без использования специализированных средств обработки сигналов. Это сделано для того, чтобы более наглядно продемонстрировать внутреннюю сущность реализуемых алгоритмов. Однако упомянутое разделение материала не является абсолютным — иллюстрации, необходимые при изложении теоретических вопросов, не всегда удается получить без специальных средств MATLAB, а при описании функций MATLAB иногда возникает необходимость привести дополнительные теоретические сведения. Таким образом, теоретическая и "компьютерная" части глав книги дополняют друг друга, предоставляя читателю возможность взглянуть на обсуждаемые вопросы с разных точек зрения.

В книге много иллюстраций, полученных с помощью MATLAB, при этом во всех случаях приводится и соответствующий программный код. Разумеется, при описании многих функций MATLAB также демонстрируется их использование. Таким образом, книга содержит множество примеров реализации алгоритмов обработки сигналов средствами MATLAB, так что заинтересованный читатель сможет освоить эту систему и использовать ее для решения собственных задач.

Следующее необходимое замечание касается терминологии. Наука и техника интернациональны по самой своей природе, и бурное развитие информационных технологий лишь сделало это еще более очевидным. Поэтому любому, кто занимается

обработкой сигналов, приходится иметь дело с множеством зарубежных, и прежде всего англоязычных, источников информации. Помимо собственно владения английским языком это требует знания терминологии, которая не всегда соответствует принятой в отечественной литературе. Чтобы облегчить читателю работу с англоязычной литературой, при введении новых понятий в книге приводятся соответствующие английские термины.

## Чего нет в этой книге

Книга является *базовым* курсом по цифровой обработке сигналов, поэтому в ней обсуждаются фундаментальные положения и отсутствуют разделы, касающиеся более сложных вопросов, таких как решетчатые фильтры, вейвлет-анализ, идентификация систем, двумерная обработка сигналов, нейронные сети и т. д. Задача книги — лишь *подготовить* читателя к восприятию этих актуальных и современных тем. Чтобы быстро прикинуть, рассматриваются ли в книге интересующие вас вопросы, взгляните на приведенный ранее рис. В1, где показано место, занимаемое книгой в общей структуре разделов ЦОС.

Данная книга не является справочником по рассматриваемым пакетам расширения MATLAB, хотя в ней описана значительная часть функций пакетов Signal Processing, Filter Design и Communications. Следует иметь в виду, что при описании некоторых функций MATLAB приведены не все, а лишь наиболее употребительные способы их вызова. Наиболее полную информацию о функциях можно получить лишь из документации MATLAB и пакетов расширения.

## Благодарности

Прежде всего хотелось бы поблагодарить самых близких мне людей — жену Юлию и сына Андрея — за терпение и понимание.

Вот уже более двадцати лет моя жизнь связана с факультетом радиотехники и телекоммуникаций Санкт-Петербургского государственного электротехнического университета "ЛЭТИ". Спасибо учителям и коллегам, во многом определившим мой путь как преподавателя и специалиста — без них эта книга никогда не была бы написана.

Хочу сказать большое спасибо участникам форума на сайте <http://matlab.exponenta.ru> — за все вопросы, заданные мне за почти десять лет существования этого ресурса. Сетевое общение с широкой аудиторией — это замечательная для преподавателя возможность понять, как воспринимаются те или иные вопросы и как именно их следует объяснить.

Огромная благодарность также всем сотрудникам издательства "БХВ-Петербург", принявшим участие в подготовке и выпуске книги.

Особой благодарности заслуживает фирма The MathWorks, Inc. — за создание замечательной компьютерной системы MATLAB и за предоставление в распоряже-

ние автора последних версий MATLAB и описываемых в книге пакетов расширения.

## Информация от компании The MathWorks, Inc.

MATLAB® — зарегистрированная торговая марка, принадлежащая компании The MathWorks, Inc. The MathWorks не гарантирует отсутствия ошибок в тексте и программном коде, приведенных в этой книге. Обсуждение в данной книге пакета MATLAB® и иных программных продуктов не означает какого-либо одобрения предлагаемых способов их использования со стороны The MathWorks, Inc. Далее приводится контактная информация компании The MathWorks, Inc.:

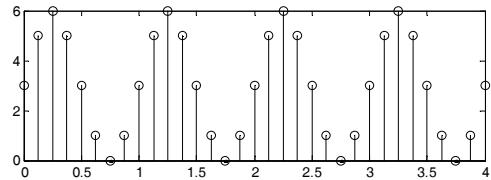
The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA, 01760-2098 USA  
Phone: 508-647-7000  
Fax: 508-647-7001  
E-mail: [info@mathworks.com](mailto:info@mathworks.com)  
Web: [www.mathworks.com](http://www.mathworks.com)

## От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты [mail@bvh.ru](mailto:mail@bvh.ru). Мы будем рады узнать ваше мнение!

Подробную информацию о наших книгах вы найдете на web-сайте издательства <http://www.bhv.ru>.

# ГЛАВА 1



## Основы анализа сигналов

Анализ — один из ключевых компонентов обработки сигналов. Его теоретические основы рассматриваются в большом числе книг — от учебников и учебных пособий по курсу "Радиотехнические цепи и сигналы" [1—4] до фундаментальных монографий, таких как [5]. В данной главе будут приведены основополагающие понятия и методы анализа сигналов.

Эта глава является вводной, и в ней рассматриваются аналоговые сигналы, а не дискретные или цифровые. Однако методы анализа аналоговых и дискретных сигналов тесно взаимосвязаны, поэтому полноценное освоение изложенного здесь материала необходимо для глубокого понимания главы 3, посвященной дискретным сигналам. Глава получилась весьма объемной, поскольку в ней вводится и обсуждается большое количество базовых понятий. Средства MATLAB в данной главе используются в основном для построения графиков.

Основной целью анализа является сравнение сигналов друг с другом для выявления их сходства и различия. Можно выделить три основных составляющих анализа сигналов:

- **измерение числовых параметров сигналов.** К таким параметрам прежде всего относятся энергия, средняя мощность и среднеквадратическое значение, а речь об их расчете пойдет в разд. "Энергия и мощность сигнала" этой главы;
- **разложение сигнала на элементарные составляющие** для их рассмотрения по отдельности либо для сравнения свойств различных сигналов. Такое разложение производится с использованием рядов и интегральных преобразований, важнейшими среди которых являются ряд Фурье и преобразование Фурье. Им будут посвящены одноименные разделы;
- **количественное измерение степени "похожести" различных сигналов.** Такое измерение производится с применением аппарата корреляционного анализа, который будет рассмотрен в соответствующем разделе.

Кроме того, во многих случаях полезным оказывается представление сигналов в виде *векторов*, для которых определены понятия размера, взаимного расстояния и т. п. Об этом пойдет речь в разд. "Пространство сигналов" этой главы.

## Классификация сигналов

Прежде чем приступать к рассмотрению задач анализа сигналов, выделим некоторые классы сигналов, которые будут часто встречаться нам в дальнейшем. Это необходимо по двум причинам. Во-первых, проверка принадлежности сигнала к конкретному классу сама по себе является процедурой анализа. Во-вторых, для представления и анализа сигналов разных классов зачастую приходится использовать разные средства и подходы.

Итак, что же такое сигнал? В наиболее общей формулировке это зависимость одной величины от другой (т. е. с математической точки зрения *сигнал* является *функцией*). Чаще всего рассматриваются зависимости от времени, хотя это не обязательно. Например, в системах оптической обработки информации сигналом может являться зависимость интенсивности света от пространственных координат. Физическая природа сигнала может быть весьма различной. Очень часто это напряжение, несколько реже — ток, возможны и многие другие физические величины.

В данной книге подразумевается (если иное не оговорено специально), что *сигнал* представляет собой зависимость *напряжения от времени*.

А теперь обратимся к собственно классификации.

В зависимости от того, известен ли нам сигнал *точно*, различают детерминированные и случайные *сигналы*. *Детерминированный сигнал* полностью известен — его значение в любой момент времени можно определить точно. *Случайный же сигнал* в любой момент времени представляет собой случайную величину, которая принимает конкретные значения с некоторой *вероятностью*. Специфические свойства случайных сигналов будут рассмотрены в конце данной главы, а основы анализа сигналов мы будем обсуждать применительно к сигналам детерминированным.

Следующий важный класс сигналов — сигналы с *интегрируемым квадратом*. Еще их называют сигналами с *ограниченной энергией* (почему, станет ясно из разд. "Энергия и мощность сигнала" этой главы). Для таких сигналов  $s(t)$  выполняется соотношение

$$\int_{-\infty}^{\infty} s^2(t) dt < \infty. \quad (1.1)$$

Многие важные соотношения теории сигналов получены в предположении о конечности энергии анализируемых сигналов. Если это условие не выполняется, приходится менять подходы к решению задачи (см., например, определения понятия корреляционной функции для сигналов с конечной и бесконечной энергией в разд. "Корреляционная функция" этой главы) или прибегать к использованию аппарата обобщенных функций (см. разд. "Фурье-анализ неинтегрируемых сигналов" в этой главе).

Еще один признак классификации сигналов, существенно влияющий на методы их анализа, — *периодичность*. Для периодического сигнала с периодом  $T$  выполняется соотношение

$$s(t + nT) = s(t) \text{ при любом } t,$$

где  $n$  — произвольное целое число. Если величина  $T$  является периодом сигнала  $s(t)$ , то периодами для него будут и кратные ей значения:  $2T, 3T$  и т. д. Как правило, говоря о периоде сигнала, имеют в виду минимальный из возможных периодов.

Величина, обратная периоду, называется *частотой повторения сигнала*:  $f = 1/T$ . В теории сигналов также часто используется понятие *круговой частоты*  $\omega = 2\pi f$ , измеряемой в радианах в секунду.

Очевидно, что любой периодический сигнал (за исключением сигнала, тождественно равного нулю) имеет бесконечную энергию.

Следующий класс — сигналы *конечной длительности* (их еще называют *финитными сигналами*). Такие сигналы отличны от нуля только на ограниченном промежутке времени. Иногда говорят, что сигнал *существует* на конечном временном интервале.

Очевидно, что сигнал конечной длительности будет иметь и конечную энергию — если только он не содержит разрывов второго рода (с уходящими в бесконечность ветвями функции).

Перейдем к более узким классам сигналов. Очень важную роль в технике обработки сигналов играют гармонические колебания, которые в самом общем виде записываются следующим образом:

$$s(t) = A \cos(\omega t + \varphi).$$

Гармонический сигнал полностью определяется тремя числовыми параметрами: *амплитудой*  $A$ , *частотой*  $\omega$  и *начальной фазой*  $\varphi$ .

Гармонический сигнал является одним из широко распространенных *тестовых* сигналов, применяющихся для анализа характеристик цепей. Кроме него к тестовым относятся еще две очень важных в радиотехнике функции: дельта-функция и функция единичного скачка.

*Дельта-функция*  $\delta(t)$ , или *функция Дирака*, представляет собой бесконечно узкий импульс с бесконечной амплитудой, расположенный при нулевом значении аргумента функции. "Площадь" импульса тем не менее равна единице:

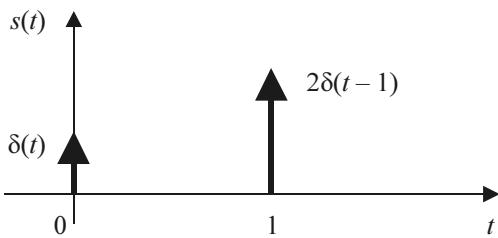
$$\delta(t) = \begin{cases} 0, & t \neq 0, \\ \infty, & t = 0, \end{cases} \quad \int_{-\infty}^{\infty} \delta(t) dt = 1.$$

Разумеется, сигнал в виде дельта-функции невозможно реализовать физически, однако эта функция очень важна для теоретического анализа сигналов и систем.

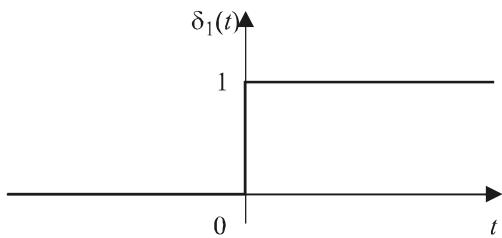
На графиках дельта-функция обычно изображается жирной стрелкой, высота которой пропорциональна множителю, стоящему перед дельта-функцией (рис. 1.1).

Одно из важных свойств дельта-функции — так называемое *фильтрующее свойство*. Оно состоит в том, что если дельта-функция присутствует под интегралом в качестве множителя, то результат интегрирования будет равен значению остального подынтегрального выражения в той точке, где сосредоточен дельта-импульс:

$$\int_{-\infty}^{\infty} f(t) \delta(t - t_0) dt = f(t_0). \quad (1.2)$$



**Рис. 1.1.** График сигнала  
 $s(t) = \delta(t) + 2\delta(t - 1)$



**Рис. 1.2.** Функция единичного скачка

### ЗАМЕЧАНИЕ

Пределы интегрирования в (1.2) не обязательно должны быть бесконечными, главное, чтобы в интервал интегрирования попадало значение  $t_0$ ; в противном случае интеграл будет равен нулю.

Из того факта, что интеграл от дельта-функции дает безразмерную единицу, следует, что размерность самой дельта-функции обратна размерности ее аргумента. Например, дельта-функция времени имеет размерность  $1/c$ , т. е. размерность частоты.

Функция единичного скачка  $\delta_1(t)$ , она же *функция Хевисайда*, она же *функция включения*, равна нулю для отрицательных значений аргумента и единице — для положительных. При нулевом значении аргумента функцию считают либо неопределенной, либо равной  $1/2$ :

$$\delta_1(t) = \begin{cases} 0, & t < 0, \\ 1/2, & t = 0, \\ 1, & t > 0. \end{cases} \quad (1.3)$$

В MATLAB данную функцию можно смоделировать с помощью оператора сравнения, возвращающего значение 0 или 1:

```
d1 = double(t >= 0);
```

Отличие такой реализации функции включения от формулы (1.3) состоит только в том, что при нулевом значении аргумента результат равен единице; впрочем, в большинстве случаев это отличие несущественно.

График функции единичного скачка приведен на рис. 1.2.

Функцию единичного скачка удобно использовать при создании математических выражений для сигналов конечной длительности. Простейшим примером является формирование прямоугольного импульса с амплитудой  $A$  и длительностью  $T$ :

$$s(t) = A(\delta_1(t) - \delta_1(t - T)).$$

Вообще, любую кусочно-заданную зависимость можно записать в виде единого математического выражения с помощью функции единичного скачка.

Сигналы, отражающие поведение физических величин, являются вещественными функциями времени. Однако в ряде случаев оказывается удобно вводить в рассмотр-

рение комплексные сигналы (см., например, разд. "Комплексная огибающая" далее в этой главе). Одним из простейших комплексных сигналов является комплексное гармоническое колебание

$$\dot{s}(t) = A \exp(j(\omega t + \phi)) = \dot{A}_m \exp(j\omega t),$$

где  $\dot{A}_m = A \exp(j\phi)$  — комплексная амплитуда данного сигнала.

## Энергия и мощность сигнала

В начале этой главы в качестве одной из составляющих анализа сигналов было названо измерение их количественных параметров. На практике очень часто используются такие параметры, как энергия и мощность сигнала. Их определения, принятые в теории сигналов, отличаются от обычных, а потому требуют некоторых комментариев.

Начнем с обычных, "физических" понятий мощности и энергии. Если к резистору с сопротивлением  $R$  приложено постоянное напряжение  $U$ , то выделяющаяся в резисторе мощность будет равна

$$P = \frac{U^2}{R}.$$

За время  $T$  в этом резисторе выделится тепловая энергия, равная

$$E = \frac{U^2 T}{R}.$$

Пусть теперь к тому же резистору приложено не постоянное напряжение, а сигнал  $s(t)$ . Рассеивающаяся в резисторе мощность при этом тоже будет зависеть от времени, т. е. в данном случае речь идет о *мгновенной* мощности (instantaneous power):

$$p(t) = \frac{s^2(t)}{R}.$$

Чтобы вычислить выделяющуюся за время  $T$  энергию, мгновенную мощность необходимо проинтегрировать:

$$E = \int_0^T p(t) dt = \frac{1}{R} \int_0^T s^2(t) dt.$$

Можно ввести также понятие *средней* мощности (average power) за заданный промежуток времени, разделив энергию на длительность временного интервала:

$$P_{cp} = \frac{E}{T} = \frac{1}{RT} \int_0^T s^2(t) dt.$$

Все приведенные формулы входят сопротивление нагрузки  $R$ . Однако если энергия и мощность интересуют нас не как физические величины, а как средство *сравнения* различных сигналов, этот параметр можно из формул исключить (принять

$R = 1$ ). Тогда мы получим определения энергии, мгновенной мощности и средней мощности, принятые в теории сигналов:

$$E = \int_0^T s^2(t)dt, \quad p(t) = s^2(t), \quad P_{\text{cp}} = \frac{1}{T} \int_0^T s^2(t)dt. \quad (1.4)$$

"Мощность" здесь имеет размерность  $\text{В}^2$  (вольт в квадрате), а "энергия" —  $\text{В}^2 \cdot \text{с}$ .

### ЗАМЕЧАНИЕ

Данные параметры иногда называют *удельной мощностью* и *энергией*, чтобы подчеркнуть подразумеваемое при этом единичное значение сопротивления нагрузки.

Энергия сигнала может быть конечной или бесконечной. Например, любой сигнал конечной длительности будет иметь конечную энергию (если только он не содержит дельта-функций или ветвей, уходящих в бесконечность). А любой периодический сигнал, напротив, имеет бесконечную энергию.

Если энергия сигнала бесконечна, можно определить его среднюю *мощность* на всей временной оси. Для этого нужно воспользоваться формулой (1.4) и выполнить предельный переход, устремив интервал усреднения в бесконечность:

$$P_{\text{cp}} = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} s^2(t)dt. \quad (1.5)$$

Средняя мощность в такой формулировке представляет собой *средний квадрат* сигнала (английский термин — mean square, *MSQ*).

Квадратный корень из средней мощности дает *среднеквадратическое (действующее) значение* сигнала (английский термин — root mean square, *RMS*):

$$\sigma_s = \sqrt{P_{\text{cp}}} = \sqrt{\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} s^2(t)dt}. \quad (1.6)$$

Для периодического сигнала предельный переход в формулах (1.5) и (1.6) выполнить не обязательно — достаточно выполнить усреднение по периоду  $T$ .

## Ряд Фурье

Разложению в ряд Фурье могут подвергаться *периодические* сигналы. При этом они представляются в виде суммы гармонических функций либо комплексных экспонент с частотами, имеющими некоторый общий делитель. Чтобы такое разложение существовало, фрагмент сигнала длительностью в один период должен удовлетворять *условиям Дирихле*:

- не должно быть разрывов второго рода (с уходящими в бесконечность ветвями функций);
- число разрывов первого рода (скачков) должно быть конечным;

- число экстремумов должно быть конечным (в качестве примера функции, которая на конечном интервале имеет бесконечное число экстремумов, можно привести  $\sin(1/x)$  в окрестности нуля).

В зависимости от конкретной формы базисных функций различают несколько форм записи ряда Фурье.

### **ЗАМЕЧАНИЕ**

Ряд Фурье может быть применен для представления не только периодических сигналов, но и сигналов конечной длительности. При этом оговаривается временной интервал, для которого строится ряд Фурье, а в остальные моменты времени сигнал считается равным нулю. Для расчета коэффициентов ряда такой подход фактически означает *периодическое продолжение* сигнала за границами рассматриваемого интервала. С таким продолжением применительно к дискретному сигналу мы столкнемся также в главе 5 при рассмотрении дискретного преобразования Фурье.

## **Синусно-косинусная форма**

В этом варианте ряд Фурье имеет следующий вид:

$$s(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos(k\omega_1 t) + b_k \sin(k\omega_1 t)). \quad (1.7)$$

Здесь  $\omega_1 = 2\pi/T$  — круговая частота, соответствующая периоду повторения сигнала, равному  $T$ . Входящие в формулу кратные ей частоты  $k\omega_1$  называются гармониками; гармоники нумеруются в соответствии с индексом  $k$ ; частота  $\omega_k = k\omega_1$  называется  $k$ -й гармоникой сигнала. Коэффициенты ряда  $a_k$  и  $b_k$  рассчитываются по формулам

$$a_k = \frac{2}{T} \int_{-T/2}^{T/2} s(t) \cos(k\omega_1 t) dt, \quad b_k = \frac{2}{T} \int_{-T/2}^{T/2} s(t) \sin(k\omega_1 t) dt.$$

Константа  $a_0$  рассчитывается по общей формуле для  $a_k$ . Ради этой общности и введена несколько странная на первый взгляд форма записи постоянного слагаемого (с делением на два). Само же это слагаемое представляет собой среднее значение сигнала на периоде:

$$\frac{a_0}{2} = \frac{1}{T} \int_{-T/2}^{T/2} s(t) dt.$$

### **ЗАМЕЧАНИЕ**

Пределы интегрирования не обязательно должны быть такими, как в приведенных выше формулах (от  $-T/2$  до  $T/2$ ). Интегрирование может производиться по любому интервалу длиной  $T$  — результат от этого не изменится. Конкретные пределы выбираются из соображений удобства вычислений; например, может оказаться удобнее выполнять интегрирование от 0 до  $T$ .

Если  $s(t)$  является четной функцией, то все  $b_k$  будут равны нулю и в формуле ряда Фурье будут присутствовать только косинусные слагаемые. Если  $s(t)$  является нечетной функцией, равны нулю будут, наоборот, косинусные коэффициенты  $a_k$ , и в формуле останутся лишь синусные слагаемые.

## Вещественная форма

Некоторое неудобство синусно-косинусной формы ряда Фурье состоит в том, что для каждой гармоники с частотой  $k\omega_1$  в формуле фигурируют два слагаемых — синус и косинус. Воспользовавшись формулами тригонометрических преобразований, сумму этих двух слагаемых можно трансформировать в косинус той же частоты с иной амплитудой и некоторой начальной фазой:

$$s(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} A_k \cos(k\omega_1 t + \varphi_k), \quad (1.8)$$

где

$$A_k = \sqrt{a_k^2 + b_k^2}, \quad \varphi_k = \begin{cases} -\operatorname{arctg} \frac{b_k}{a_k}, & a_k \geq 0, \\ \operatorname{arctg} \frac{b_k}{a_k} \pm \pi, & a_k < 0. \end{cases} \quad (1.9)$$

Если  $s(t)$  является четной функцией, фазы  $\varphi_k$  могут принимать только значения 0 и  $\pi$ , а если  $s(t)$  — функция нечетная, то возможные значения для фазы равны  $\pm\pi/2$ .

Данная форма представления ряда Фурье является наиболее естественной с "инженерной" точки зрения, однако ее неудобство заключается в том, что параметры  $A_k$  и  $\varphi_k$  не рассчитываются напрямую — они выражаются либо через синусные и косинусные коэффициенты согласно (1.9), либо через коэффициенты рассматриваемой далее комплексной формы ряда Фурье (см. далее формулу (1.11)).

## Комплексная форма

Данная форма представления ряда Фурье является, пожалуй, наиболее употребимой в радиотехнике. Она получается из вещественной формы представлением косинуса в виде полусуммы комплексных экспонент (такое представление вытекает из формулы Эйлера  $e^{ix} = \cos x + j \sin x$ ):

$$\cos x = \frac{1}{2}(e^{ix} + e^{-ix}).$$

Применив данное преобразование к вещественной форме ряда Фурье, получим суммы комплексных экспонент с положительными и отрицательными показателями:

$$s(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \frac{A_k}{2} (\exp(jk\omega_1 t + j\varphi_k) + \exp(-jk\omega_1 t - j\varphi_k)).$$

А теперь будем трактовать экспоненты со знаком "минус" в показателе как члены ряда с отрицательными номерами. В рамках этого же общего подхода постоянное слагаемое  $a_0/2$  станет членом ряда с нулевым номером. В результате получится комплексная форма записи ряда Фурье:

$$s(t) = \sum_{k=-\infty}^{\infty} \dot{C}_k e^{-jk\omega_l t}. \quad (1.10)$$

Комплексные коэффициенты ряда связаны с амплитудами  $A_k$  и фазами  $\varphi_k$ , фигурирующими в вещественной форме записи ряда Фурье (1.8), следующими несложными соотношениями:

$$\dot{C}_k = \frac{1}{2} A_k e^{j\varphi_k},$$

$$A_k = 2 |\dot{C}_k|, \quad \varphi_k = \arg(\dot{C}_k). \quad (1.11)$$

Несложно выглядят и формулы связи с коэффициентами  $a_k$  и  $b_k$  синусно-косинусной формы ряда Фурье (1.7):

$$\begin{aligned} \dot{C}_k &= \frac{a_k}{2} - j \frac{b_k}{2}, \\ a_k &= 2 \operatorname{Re}(\dot{C}_k), \quad b_k = -2 \operatorname{Im}(\dot{C}_k). \end{aligned}$$

Отсюда сразу же следует и формула непосредственного расчета коэффициентов  $\dot{C}_k$  ряда Фурье в комплексной форме:

$$\dot{C}_k = \frac{1}{T} \int_{-T/2}^{T/2} s(t) \exp(-jk\omega_l t) dt. \quad (1.12)$$

Если  $s(t)$  является *четной* функцией, коэффициенты ряда  $\dot{C}_k$  будут чисто *вещественными*, а если  $s(t)$  — функция *нечетная*, коэффициенты ряда окажутся чисто *мнимыми*.

Совокупность амплитуд гармоник ряда Фурье часто называют *амплитудным спектром*, а совокупность их фаз — *фазовым спектром*. Эти понятия не следует путать с амплитудно- и фазочастотными *характеристиками*, которые относятся не к сигналам, а к системам, осуществляющим обработку сигналов.

Если анализируемый сигнал  $s(t)$  является вещественным, то его амплитудный и фазовый спектры обладают соответственно четной и нечетной симметрией:

$$A_{-k} = A_k, \quad \varphi_{-k} = -\varphi_k,$$

а коэффициенты комплексного ряда Фурье — комплексно-сопряженной симметрией:

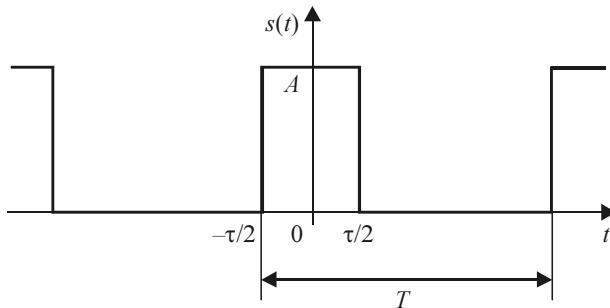
$$\dot{C}_{-k} = \dot{C}_k^*.$$

## Примеры разложения сигналов в ряд Фурье

В данном разделе мы применим ряд Фурье для анализа конкретных сигналов.

### Последовательность прямоугольных импульсов

Первым рассматриваемым сигналом будет последовательность прямоугольных импульсов с амплитудой  $A$ , длительностью  $\tau$  и периодом повторения  $T$ . Начало отсчета времени примем расположенным в середине импульса (рис. 1.3).



**Рис. 1.3.** Периодическая последовательность прямоугольных импульсов ( $t$ )

Данный сигнал является четной функцией, поэтому для его представления удобнее использовать синусно-косинусную форму ряд Фурье — в ней будут присутствовать только косинусные слагаемые  $a_k$ , равные

$$a_k = \frac{2}{T} \int_{-\tau/2}^{\tau/2} A \cos\left(\frac{2\pi k}{T}t\right) dt = \frac{2A}{\pi k} \sin\left(\frac{\pi k \tau}{T}\right).$$

Длительность импульсов и период их следования входят в полученную формулу не обособленно, а исключительно в виде отношения. Этот параметр — отношение периода к длительности импульсов — называют *скважностью* последовательности импульсов. Обозначим ее буквой  $q$  и введем этот параметр ( $q = T/\tau$ ) в полученную формулу для коэффициентов ряда Фурье, а затем приведем формулу к виду  $\sin(x)/x$ :

$$a_k = \frac{2A}{\pi k} \sin\left(\frac{\pi k}{q}\right) = \frac{2A}{q} \frac{\sin\left(\frac{\pi k}{q}\right)}{\frac{\pi k}{q}}. \quad (1.13)$$

#### ЗАМЕЧАНИЕ

В зарубежной литературе вместо скважности используется обратная величина, называемая *коэффициентом заполнения* (duty cycle) и равная  $\tau/T$ .

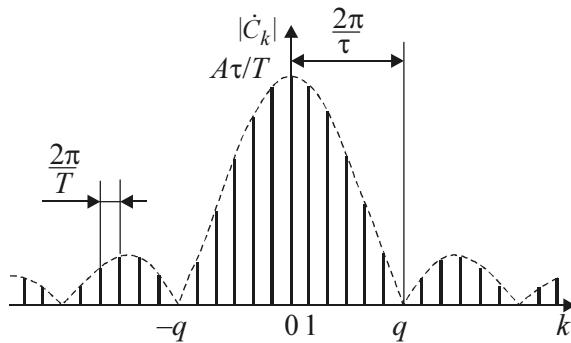
При такой форме записи становится хорошо видно, чему равно значение постоянного слагаемого ряда: поскольку при  $x \rightarrow 0 \sin(x)/x \rightarrow 1$ , то

$$\frac{a_0}{2} = \frac{A}{q} = \frac{A\tau}{T}.$$

Теперь можно записать и само представление последовательности прямоугольных импульсов в виде ряда Фурье:

$$s(t) = \frac{A}{q} + \sum_{k=1}^{\infty} \frac{2A}{\pi k} \sin\left(\frac{\pi k}{q}\right) \cos\left(\frac{2\pi k}{T}t\right).$$

Амплитуды гармонических слагаемых ряда зависят от номера гармоники по закону  $|\sin(x)/x|$  (рис. 1.4).



**Рис. 1.4.** Коэффициенты ряда Фурье  
для последовательности прямоугольных импульсов

График функции  $\sin(x)/x$  имеет лепестковый характер. Говоря о ширине этих лепестков, следует подчеркнуть, что для графиков дискретных спектров периодических сигналов возможны два варианта градуировки горизонтальной оси — в номерах гармоник и в частотах. На рис. 1.4 градуировка оси соответствует номерам гармоник, а частотные параметры спектра нанесены на график с помощью размерных линий.

Ширина лепестков, измеренная в количестве гармоник, равна скважности последовательности (при  $k=nq$  имеем  $a_k=a_{nq}=0$ , если  $n \neq 0$ ). Отсюда следует важное свойство спектра последовательности прямоугольных импульсов — в нем отсутствуют (имеют нулевые амплитуды) гармоники с номерами, кратными скважности.

Расстояние по частоте между соседними гармониками равно частоте следования импульсов —  $2\pi/T$ . Ширина лепестков спектра, измеренная в единицах частоты, равна  $2\pi/\tau$ , т. е. обратно пропорциональна длительности импульсов. Это, как мы увидим далее, проявление общего закона — чем короче сигнал, тем шире его спектр.

## Меандр

Важным частным случаем предыдущего сигнала является *меандр* — последовательность прямоугольных импульсов со скважностью, равной двум, когда длительности импульсов и промежутков между ними становятся равными (рис. 1.5).

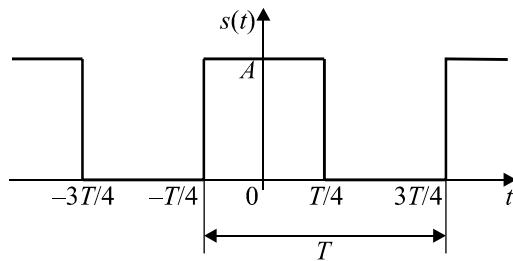


Рис. 1.5. Меандр

Подставив  $q = 2$  в формулу (1.13), получим

$$a_k = A \frac{\sin(\pi k / 2)}{\pi k / 2} = \begin{cases} A, & k = 0, \\ 0, & k = 2m, \quad m \neq 0, \\ \frac{2A}{\pi k}, & k = 4m + 1, \\ -\frac{2A}{\pi k}, & k = 4m - 1. \end{cases}$$

Здесь  $m$  — произвольное целое число.

Таким образом, в спектре меандра присутствуют только нечетные гармоники. Это согласуется с общим правилом, приведенным выше. Представление меандра в виде ряда Фурье с учетом этого может быть записано следующим образом:

$$s(t) = \frac{A}{2} + \frac{2A}{\pi} \left( \cos\left(\frac{2\pi}{T}t\right) - \frac{1}{3} \cos\left(3\frac{2\pi}{T}t\right) + \frac{1}{5} \cos\left(5\frac{2\pi}{T}t\right) - \dots \right).$$

Гармонические составляющие, из которых складывается меандр, имеют амплитуды, обратно пропорциональные номерам гармоник, и чередующиеся знаки.

Покажем на примере меандра, как складывается заданный сигнал из отдельных гармоник (рис. 1.6):

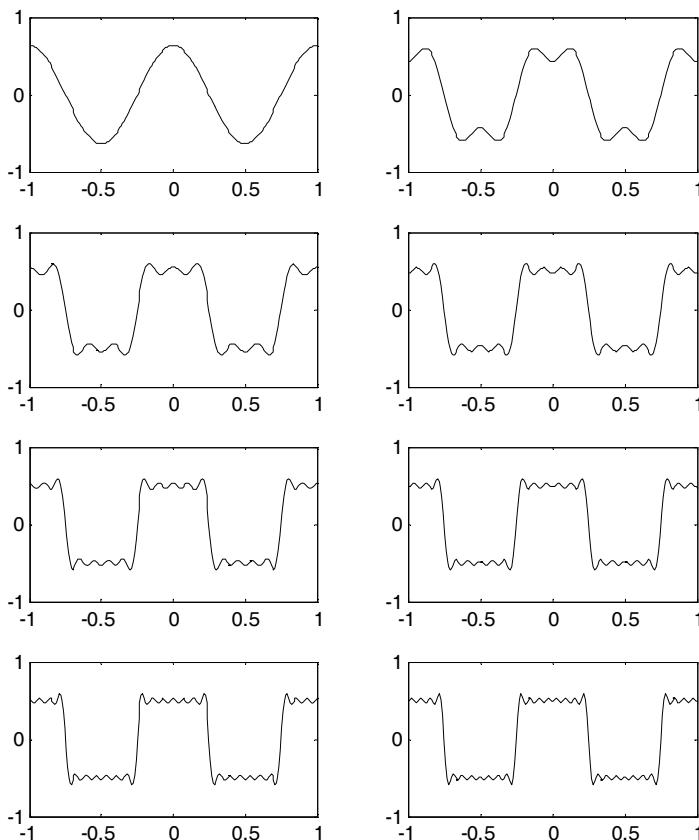
```
>> N = 8; % число ненулевых гармоник
>> t = -1:0.01:1; % вектор моментов времени
>> A = 1; % амплитуда
>> T = 1; % период
>> nh = (1:N)*2-1; % номера ненулевых гармоник
>> % строки массива — колебания отдельных гармоник
>> harmonics = cos(2*pi.*nh'.*t/T); % амплитуды гармоник
>> Am = 2/pi./nh; % чередование знаков
>> Am(2:2:end) = -Am(2:2:end);
>> s1 = harmonics .* repmat(Am', 1, length(t));
>> % строки — частичные суммы гармоник
>> s2 = cumsum(s1);
>> for k=1:N, subplot(ceil(N/2), 2, k), plot(t, s2(k,:)), end
```

Вообще, последовательность прямоугольных импульсов плохо подходит для представления рядом Фурье — она содержит скачки, а сумма любого числа гармониче-

ских составляющих с любыми амплитудами всегда будет непрерывной (и более того — бесконечно дифференцируемой во всех точках) функцией. Поэтому поведение ряда Фурье в окрестностях разрывов представляет особый интерес. На рисунках хорошо видно, что в окрестности точки разрыва суммирование ряда Фурье дает наклонный участок, причем крутизна наклона возрастает с ростом числа суммируемых гармоник. В самой точке разрыва ряд Фурье сходится к полусумме правого и левого пределов:

$$s'(t_0) = \frac{1}{2} \left( \lim_{t \rightarrow t_0^-} s(t) + \lim_{t \rightarrow t_0^+} s(t) \right).$$

Здесь  $s(t)$  — исходный сигнал,  $s'(t)$  — сумма ряда Фурье для него.



**Рис. 1.6.** Промежуточные стадии суммирования ряда Фурье для меандра

На примыкающих к разрыву участках сумма ряда Фурье дает заметные пульсации, причем на графиках рис. 1.6 заметно, что амплитуда этих пульсаций не уменьшается с ростом числа суммируемых гармоник — пульсации лишь сжимаются по горизонтали, приближаясь к точке разрыва. Это явление, присущее рядам Фурье для любых сигналов с разрывами первого рода (скачками), называется *эффектом*

*Гиббса.* Можно показать, что амплитуда первого (самого большого) выброса составляет примерно 18% от величины скачка.

## Пилообразный сигнал

Следующий сигнал, который мы рассмотрим, — пилообразный (рис. 1.7). В пределах периода он описывается линейной функцией:

$$s(t) = \frac{2A}{T}(t - kT), \quad (k - \frac{1}{2})T < t \leq (k + \frac{1}{2})T.$$

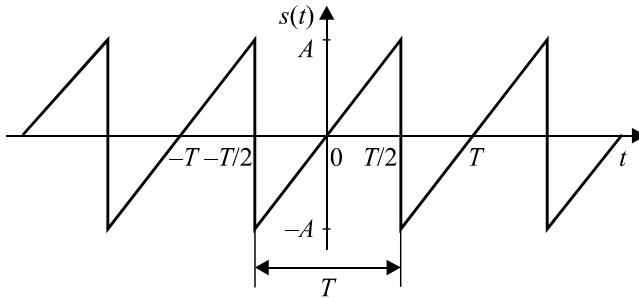


Рис. 1.7. Пилообразный сигнал

Данный сигнал является нечетной функцией, поэтому его ряд Фурье в синусно-косинусной форме (1.7) будет содержать только синусные слагаемые:

$$b_k = \frac{2}{T} \int_{-T/2}^{T/2} \frac{2A}{T} t \sin\left(\frac{2\pi k}{T} t\right) dt = \frac{2A}{\pi k} (-1)^{k+1}.$$

Сам ряд Фурье для пилообразного сигнала выглядит следующим образом:

$$s(t) = \frac{2A}{\pi} \left( \sin\left(\frac{2\pi}{T} t\right) - \frac{1}{2} \sin\left(2 \frac{2\pi}{T} t\right) + \frac{1}{3} \sin\left(3 \frac{2\pi}{T} t\right) - \frac{1}{4} \sin\left(4 \frac{2\pi}{T} t\right) + \dots \right).$$

У рассмотренных выше спектров прямоугольного и пилообразного периодических сигналов есть одна общая черта — амплитуды гармоник с ростом их номеров убывают пропорционально  $k$ . У следующего сигнала скорость затухания спектра будет иной, а почему, мы обсудим после расчета коэффициентов ряда Фурье для него.

## Последовательность треугольных импульсов

Очередной сигнал, для которого мы получим разложение в ряд Фурье, представляет собой периодическую последовательность треугольных импульсов. Строго говоря, импульсы в предыдущем сигнале тоже были треугольными, но в данном случае они будут иметь не пилообразную, а симметричную форму (рис. 1.8):

$$s(t) = A \left( 1 - 4 \frac{|t - kT|}{T} \right), \quad \left( k - \frac{1}{2} \right) T \leq t < \left( k + \frac{1}{2} \right) T.$$

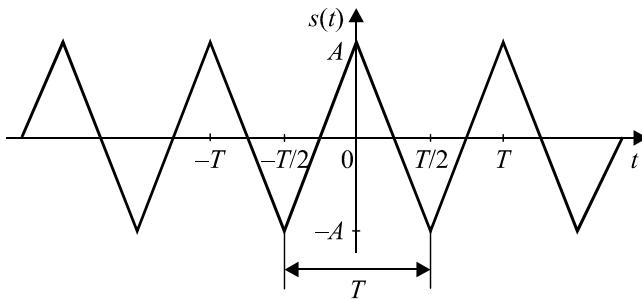


Рис. 1.8. Последовательность треугольных импульсов

Вычислим коэффициенты ряда Фурье (сигнал является четной функцией, поэтому в синусно-косинусной форме ряда Фурье (1.7) будут присутствовать только косинусные слагаемые):

$$a_k = \frac{2}{T} \int_{-T/2}^{T/2} A \left(1 - 4 \frac{|t|}{T}\right) \cos\left(\frac{2\pi k}{T} t\right) dt = \frac{4A}{(\pi k)^2} \left(1 - (-1)^k\right) = \begin{cases} 0, & k \text{ четно}, \\ \frac{8A}{(\pi k)^2}, & k \text{ нечетно}. \end{cases}$$

Как и в случае меандра, здесь присутствуют только нечетные гармоники. Сам ряд Фурье имеет следующий вид:

$$s(t) = \frac{8A}{\pi^2} \left( \cos\left(\frac{2\pi}{T} t\right) + \frac{1}{3^2} \cos\left(3 \frac{2\pi}{T} t\right) + \frac{1}{5^2} \cos\left(5 \frac{2\pi}{T} t\right) + \dots \right).$$

Как видите, в отличие от последовательностей прямоугольных и пилообразных импульсов, для треугольного периодического сигнала амплитуды гармоник убывают пропорционально *второй* степени номеров гармоник  $k$ . Это проявление общего правила, гласящего, что *скорость убывания спектра зависит от степени гладкости сигнала*. Прямоугольный и пилообразный сигналы имеют *разрывы первого рода* (скакки), и в их спектрах присутствует множитель  $1/k$ . Треугольный сигнал является *непрерывной* функцией (но ее первая производная имеет разрывы), и амплитуды гармоник его ряда Фурье содержат множитель  $1/k^2$ . Экстраполировав эту зависимость, получим следующее правило: если  $N$  — номер последней непрерывной производной сигнала, то спектр этого сигнала будет убывать со скоростью  $1/k^{N+2}$ . Предельным случаем является гармонический сигнал, дифференцировать который без потери непрерывности можно бесконечно. Согласно общему правилу, это даст бесконечную скорость убывания спектра, что вполне соответствует действительности (ряд Фурье для гармонического сигнала содержит только одну гармонику).

### ЗАМЕЧАНИЕ

То же самое правило действует и для рассматриваемого далее преобразования Фурье, только там спектральная функция будет убывать пропорционально соответствующей степени частоты  $\omega$ , а не номера гармоники.

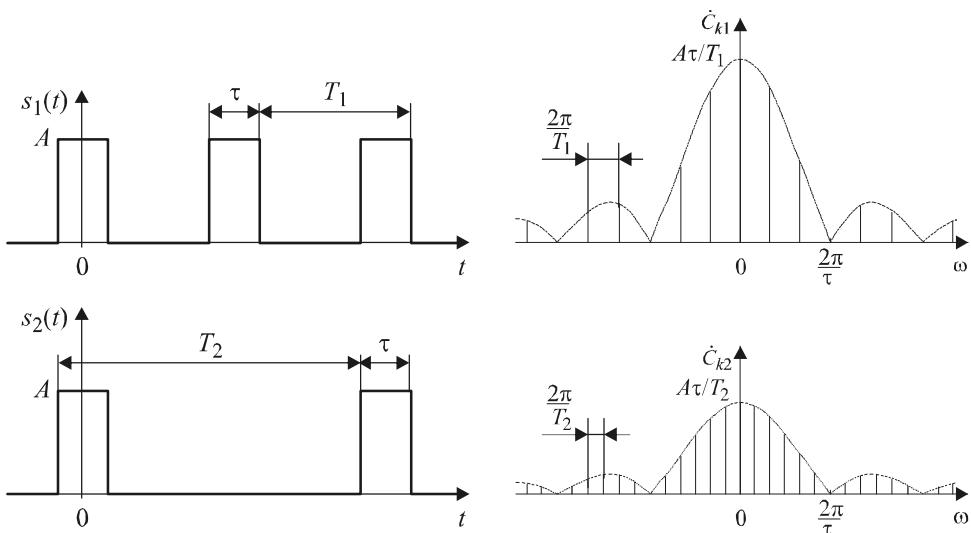
## Преобразование Фурье

Преобразование Фурье (Fourier transform) является инструментом спектрального анализа *непериодических* сигналов. Впрочем, чуть позже мы увидим, что его можно применять и к сигналам периодическим, но это потребует использования аппарата обобщенных функций.

Для наглядной иллюстрации перехода от ряда Фурье к преобразованию Фурье часто используется не вполне строгий математически, но зато понятный подход. Представим себе периодическую последовательность импульсов произвольного вида и сформируем ряд Фурье для нее. Затем, не меняя формы одиночных импульсов, увеличим период их повторения (заполнив промежутки нулевым значением) и снова рассчитаем коэффициенты ряда Фурье. Формула (1.12) для расчета коэффициентов ряда показывает, что нам придется вычислить *тот же самый* интеграл, но для более тесно расположенных частот  $\omega_k = k\omega_1$ . Изменение пределов интегрирования не играет роли — ведь на добавившемся между импульсами пространстве сигнал имеет нулевое значение. Единственное дополнительное изменение будет состоять в уменьшении общего уровня гармоник из-за деления результата интегрирования на увеличившийся период  $T$ .

На рис. 1.9 описанные изменения иллюстрируются на примере двукратного увеличения периода следования прямоугольных импульсов. Обратите внимание на то, что горизонтальная ось спектральных графиков проградуирована в значениях частот, а не номеров гармоник.

Итак, с ростом периода следования импульсов гармоники располагаются ближе друг к другу по частоте, а общий уровень спектральных составляющих становится все меньше. При этом вид вычисляемого интеграла (1.12) не меняется.



**Рис. 1.9.** Изменение спектра последовательности импульсов при двукратном увеличении периода их следования

Наконец, если устремить период к бесконечности (превратив тем самым периодическую последовательность в одиничный импульс), гармоники спектра будут плотно занимать всю частотную ось, а их амплитуды упадут до нуля (станут бесконечно малыми). Однако *взаимное соотношение* между уровнями гармоник остается неизменным и определяется все тем же интегралом (1.12). Поэтому при спектральном анализе непериодических сигналов формула для расчета коэффициентов комплексного ряда Фурье модифицируется следующим образом:

- частота перестает быть дискретно меняющейся и становится непрерывным параметром преобразования (т. е.  $k\omega_1$  в формуле (1.12) заменяется на  $\omega$ );
- удаляется множитель  $1/T$ ;
- результатом вычислений вместо нумерованных коэффициентов ряда  $\dot{C}_k$  является функция частоты  $\dot{S}(\omega)$  — *спектральная функция* сигнала  $s(t)$ . Иногда ее называют также *спектральной плотностью*.

В результате перечисленных модификаций формула (1.12) превращается в формулу *прямого преобразования Фурье*:

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt. \quad (1.14)$$

В формуле самого ряда Фурье суммирование, естественно, заменяется интегрированием (и, кроме того, перед интегралом появляется деление на  $2\pi$ ). Получающееся выражение называется *обратным преобразованием Фурье*:

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{S}(\omega) e^{j\omega t} d\omega. \quad (1.15)$$

### ЗАМЕЧАНИЕ

Если использовать не круговую, а обычную частоту  $f = \omega/(2\pi)$ , формулы прямого и обратного преобразования Фурье становятся еще более симметричными, отличаясь лишь знаком в показателе экспоненты:

$$\dot{S}(f) = \int_{-\infty}^{\infty} s(t) e^{-j2\pi ft} dt, \quad (1.16)$$

$$s(t) = \int_{-\infty}^{\infty} \dot{S}(f) e^{j2\pi ft} df. \quad (1.17)$$

Чтобы преобразование Фурье было применимо, сигнал должен удовлетворять следующим требованиям:

- на произвольном интервале конечной длительности должны выполняться условия Дирихле (см. разд. "Ряд Фурье" ранее в этой главе);
- сигнал должен быть *абсолютно интегрируемым*. Это означает, что интеграл от его модуля должен быть конечной величиной:

$$\int_{-\infty}^{\infty} |s(t)| dt < \infty. \quad (1.18)$$

Однако с привлечением математического аппарата обобщенных функций возможно выполнение Фурье-анализа и для некоторых сигналов, не удовлетворяющих этим требованиям (речь об этом пойдет далее в разд. "Фурье-анализ неинтегрируемых сигналов" этой главы).

### **ЗАМЕЧАНИЕ**

Выполнение приведенных условий гарантирует, что интеграл прямого преобразования Фурье (1.14) сходится при любой частоте  $\omega$ . Если допустить наличие некоторых частот, на которых интеграл (1.14) не существует, можно заменить условие абсолютной интегрируемости (1.18) более слабым условием конечности энергии сигнала (1.1), что и сделано в ряде источников.

Если анализируемый сигнал  $s(t)$  — вещественная функция, то соответствующая спектральная функция  $\dot{S}(\omega)$  является "сопряженно-симметричной" относительно нулевой частоты. Это означает, что значения спектральной функции на частотах  $\omega$  и  $-\omega$  являются комплексно-сопряженными по отношению друг к другу:

$$\dot{S}(-\omega) = \dot{S}^*(\omega).$$

Если  $s(t)$  — четная вещественная функция, то, как и в случае ряда Фурье, спектр будет чисто вещественным (и, следовательно, будет являться четной функцией). Если, напротив,  $s(t)$  — функция нечетная, то спектральная функция  $\dot{S}(\omega)$  будет чисто мнимой (и нечетной).

Модуль спектральной функции часто называют *амплитудным спектром*, а ее аргумент — *фазовым спектром*. Легко показать, что для вещественного сигнала амплитудный спектр является четной, а фазовый — нечетной функцией частоты:

$$|\dot{S}(-\omega)| = |\dot{S}(\omega)|, \quad \phi_s(-\omega) = -\phi_s(\omega).$$

Примеры расчета спектральных функций конкретных сигналов и соответствующие графики будут приведены далее.

Итак, преобразование Фурье (1.14) ставит в соответствие сигналу, заданному во времени, его спектральную функцию. При этом осуществляется переход из *временной области в частотную*. Преобразование Фурье является взаимно-однозначным, поэтому представление сигнала в частотной области (спектральная функция) содержит ровно столько же информации, сколько и исходный сигнал, заданный во временной области.

## **Примеры расчета преобразования Фурье**

В этом разделе будут рассмотрены примеры расчета преобразования Фурье для некоторых сигналов, часто встречающихся при решении различных задач.

## Прямоугольный импульс

Начнем с прямоугольного импульса, центрированного относительно начала отсчета времени (рис. 1.10):

$$s(t) = \begin{cases} A, & |t| \leq \tau/2, \\ 0, & |t| > \tau/2. \end{cases}$$

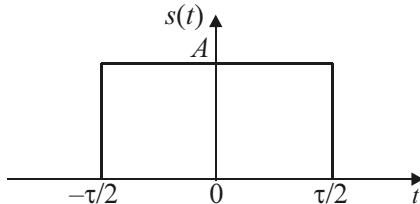


Рис. 1.10. Прямоугольный импульс

Вычисляем спектральную функцию:

$$\dot{S}(\omega) = \int_{-\tau/2}^{\tau/2} Ae^{-j\omega t} dt = \frac{2A}{\omega} \sin\left(\frac{\omega\tau}{2}\right) = A\tau \frac{\sin(\omega\tau/2)}{\omega\tau/2}.$$

Как видите, спектр представляет собой функцию вида  $\sin(x)/x$  (рис. 1.11). Амплитудный спектр имеет лепестковый характер, и ширина лепестков равна  $2\pi/\tau$ , т. е. обратно пропорциональна длительности импульса. Значение спектральной функции на нулевой частоте равно площади импульса —  $A\tau$ . Спектральная функция является вещественной, поэтому фазовый спектр принимает лишь два значения — 0 и  $\pi$ , в зависимости от знака функции  $\sin(x)/x$ . Значения фазы  $\pi$  и  $-\pi$  неразличимы, разные знаки для фазового спектра при  $\omega > 0$  и  $\omega < 0$  использованы лишь с целью представить его в виде нечетной функции.

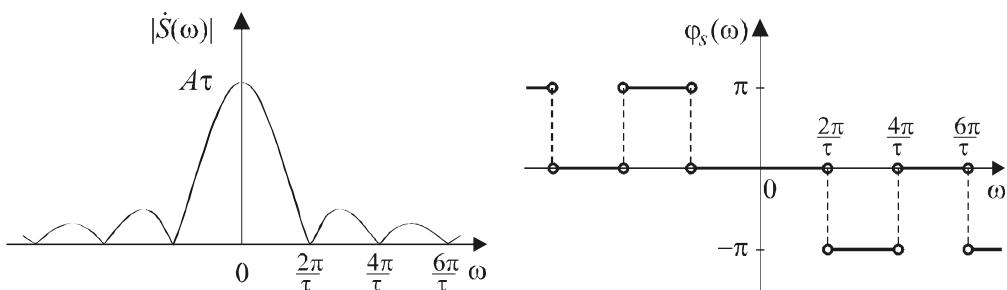
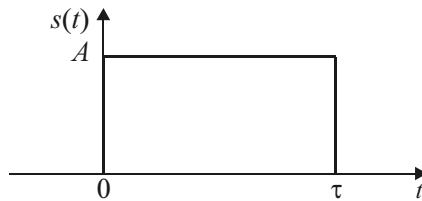


Рис. 1.11. Амплитудный (слева) и фазовый (справа) спектры прямоугольного импульса

Теперь посмотрим, что изменится после сдвига импульса во времени. Пусть импульс начинается в нулевой момент времени (рис. 1.12):

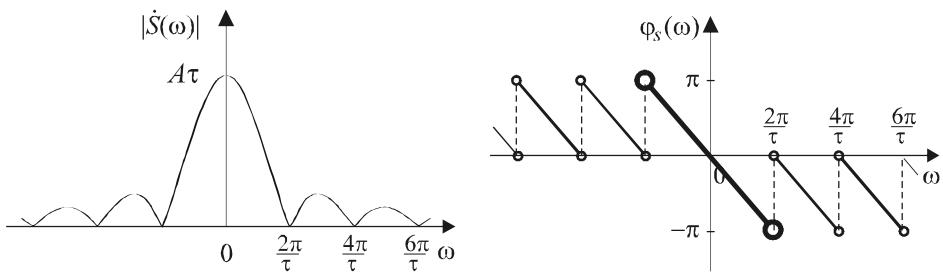
$$s(t) = \begin{cases} A, & 0 \leq t \leq \tau, \\ 0, & t < 0, \quad t > \tau. \end{cases} \quad (1.19)$$



**Рис. 1.12.** Прямоугольный импульс, задержанный во времени

Вычисляем преобразование Фурье и строим графики амплитудного и фазового спектров (рис. 1.13):

$$\dot{S}(\omega) = \int_0^{\tau} A e^{-j\omega t} dt = \frac{A}{j\omega} (1 - e^{-j\omega\tau}) = A\tau \frac{\sin(\omega\tau/2)}{\omega\tau/2} \exp\left(-j\frac{\omega\tau}{2}\right). \quad (1.20)$$



**Рис. 1.13.** Амплитудный (слева) и фазовый (справа) спектры задержанного прямоугольного импульса

Из формулы (1.20) и графиков рис. 1.13 видно, что после сдвига импульса во времени его амплитудный спектр остался прежним, а фазовый приобрел сдвиг, линейно зависящий от частоты.

#### ЗАМЕЧАНИЕ

Этот пример демонстрирует проявление свойства преобразования Фурье, касающееся изменения спектра при сдвиге сигнала во времени. Это свойство в общем виде рассмотрено далее в разд. "Свойства преобразования Фурье" этой главы.

Строго говоря, спектр данного сигнала простирается до бесконечности, лишь постепенно затухая. Поэтому вводят понятие *эффективной (практической) ширины спектра*. При лепестковом характере спектра за эффективную ширину спектра можно принять ширину главного лепестка. Из графиков видно, что она составляет  $2\pi/\tau$ , т. е. обратно пропорциональна длительности импульса. Это общее соотношение: чем короче сигнал, тем шире его спектр. Произведение же эффективных значений длительности сигнала и ширины его спектра (оно называется *базой сигнала*) остается равным некоторой константе, зависящей от конкретного способа определения этих параметров. В нашем примере это произведение, очевидно, равно  $2\pi$ . Вообще, для сигналов *простой формы* (не имеющих сложной внутриимпульсной структуры) величина базы независимо от способа определения эффективных значений длительности и ширины спектра составляет несколько единиц.

Длительность сигнала и ширина его спектра подчиняются *соотношению неопределенности*, гласящему, что произведение этих параметров (база сигнала) не может быть меньше единицы. Ограничений максимального значения базы сигнала не существует. Отсюда следует, что можно сформировать сигнал большой длительности, одновременно имеющий и широкий спектр (такие сигналы называют *широкополосными*, или *сложными*, или *сигналами с большой базой*); такие сигналы нашли широкое применение в радиолокационных, радионавигационных и телекоммуникационных системах. А вот короткий сигнал с узким спектром, согласно соотношению неопределенности, существовать не может.

## Несимметричный треугольный импульс

Далее рассмотрим несимметричный треугольный импульс (рис. 1.14):

$$s(t) = \begin{cases} A \frac{t}{T}, & 0 \leq t \leq T, \\ 0, & t < 0, \quad t > T. \end{cases}$$

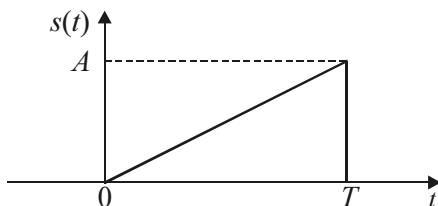


Рис. 1.14. Несимметричный треугольный импульс

Рассчитываем спектр и строим графики (рис. 1.15):

$$\dot{\hat{S}}(\omega) = \int_0^T A \frac{t}{T} e^{-j\omega t} dt = \frac{A}{(j\omega)^2 T} - \frac{A}{(j\omega)^2 T} e^{-j\omega T} - \frac{A}{j\omega} e^{-j\omega T}.$$

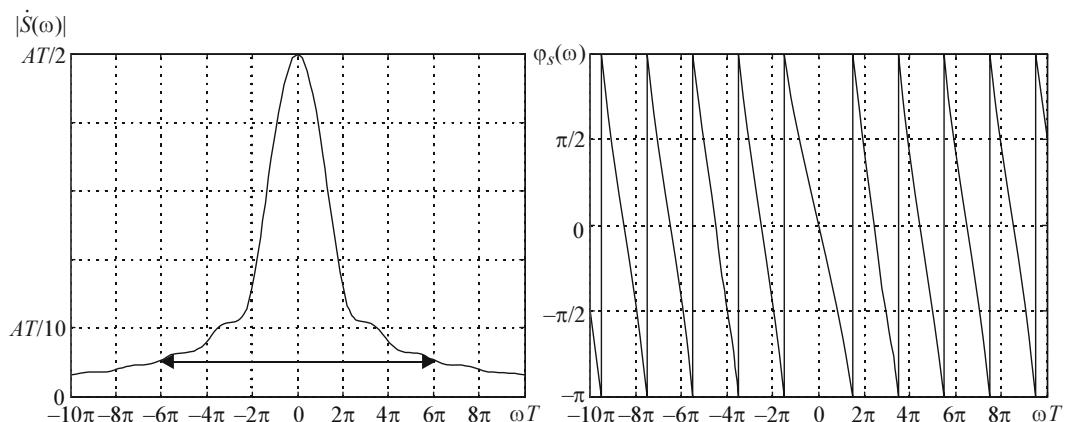


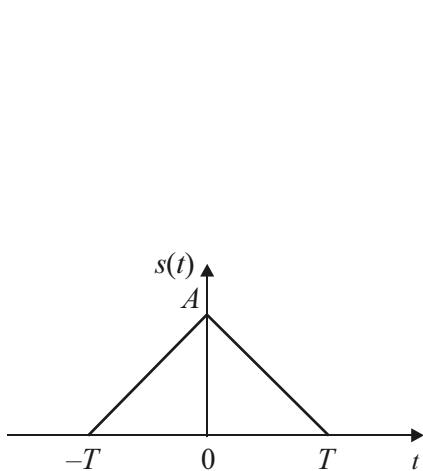
Рис. 1.15. Амплитудный (слева) и фазовый (справа) спектры несимметричного треугольного импульса

В этом случае амплитудный спектр не содержит ярко выраженных лепестков, поэтому для определения его эффективной ширины необходим иной критерий. Будем определять эффективную ширину спектра по уровню 0,1 от максимума. Из графика видно (см. рис. 1.15), что эта ширина (она показана стрелкой) составляет примерно  $6\pi/T$ . База сигнала, таким образом, равна  $6\pi$ .

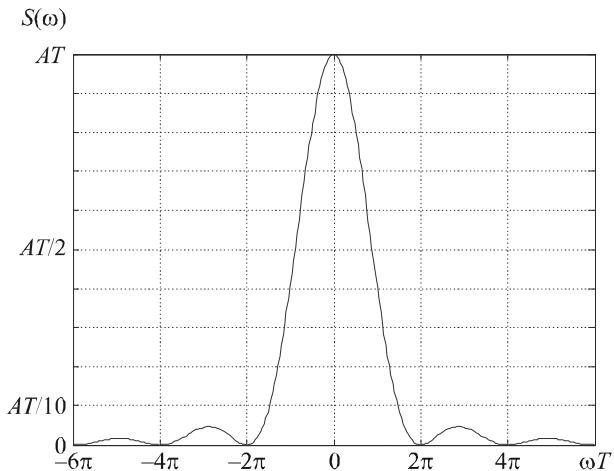
## Симметричный треугольный импульс

Следующий сигнал — симметричный треугольный импульс (рис. 1.16):

$$s(t) = \begin{cases} A\left(1 - \frac{|t|}{T}\right), & |t| \leq T, \\ 0, & |t| > T. \end{cases}$$



**Рис. 1.16.** Симметричный треугольный импульс



**Рис. 1.17.** Амплитудный спектр симметричного треугольного импульса

Рассчитываем спектральную функцию:

$$\hat{s}(\omega) = \int_{-T}^T A\left(1 - \frac{|t|}{T}\right) e^{-j\omega t} dt = AT \frac{\sin^2(\omega T / 2)}{(\omega T / 2)^2}.$$

Далее строим график амплитудного спектра (рис. 1.17). Спектральная функция оказывается не только вещественной (это сразу же следует из четности сигнала), но и неотрицательной, поэтому фазовый спектр в данном случае чисто нулевой и строить его график не имеет смысла.

Из графика видно, что спектр опять имеет лепестковую структуру, и ширина главного лепестка составляет  $2\pi/T$ , как и в случае прямоугольного импульса. Однако длительность сигнала в данном случае вдвое больше ( $2T$ ), и база оказывается равной  $4\pi$ .

Далее перейдем от сигналов конечной длительности к бесконечно протяженным сигналам.

## Односторонний экспоненциальный импульс

Первым из сигналов бесконечной длительности будет рассмотрен односторонний экспоненциальный импульс (рис. 1.18):

$$s(t) = \begin{cases} Ae^{-at}, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

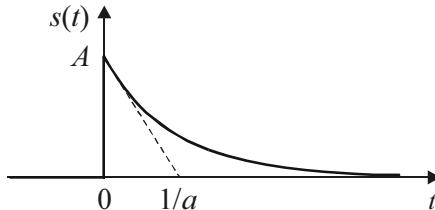


Рис. 1.18. Односторонний экспоненциальный импульс

### ЗАМЕЧАНИЕ

Пунктирная линия на рис. 1.18 и 1.20 иллюстрирует геометрический способ определения множителя в показателе экспоненты — касательная к графику функции вида  $Ae^{-at}$  пересекает горизонтальную ось на расстоянии (по горизонтали)  $1/a$  от точки касания.

Рассчитываем преобразование Фурье:

$$\hat{S}(\omega) = \int_0^{\infty} Ae^{-at} e^{-j\omega t} dt = \frac{A}{a + j\omega}.$$

Далее строим графики амплитудного и фазового спектров (рис. 1.19):

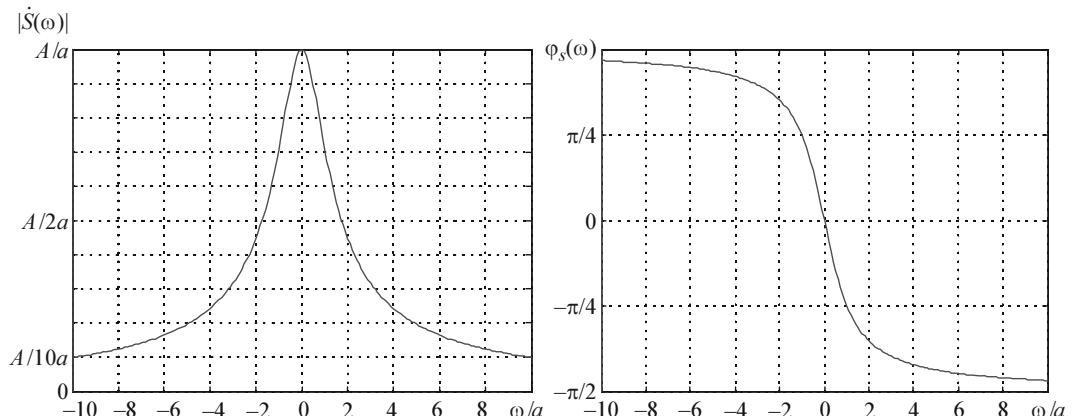
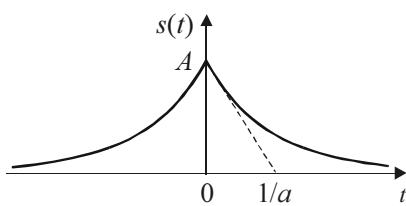


Рис. 1.19. Амплитудный (слева) и фазовый (справа) спектры одностороннего экспоненциального импульса

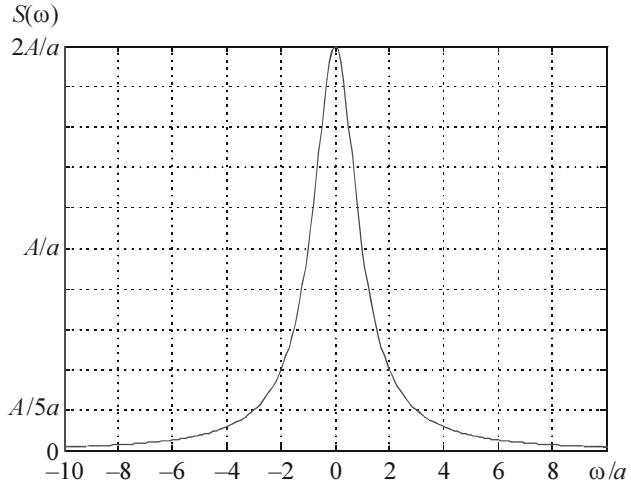
## Двусторонний экспоненциальный импульс

Теперь пусть экспоненциальный импульс будет двусторонним (симметричным, рис. 1.20):

$$s(t) = Ae^{-a|t|}.$$



**Рис. 1.20.** Двусторонний экспоненциальный импульс



**Рис. 1.21.** Амплитудный спектр двустороннего экспоненциального импульса

Рассчитываем преобразование Фурье:

$$\hat{S}(\omega) = \int_{-\infty}^{\infty} Ae^{-a|t|} e^{-j\omega t} dt = \frac{2Aa}{a^2 + \omega^2}.$$

Спектр в данном случае чисто вещественный, поэтому строить график фазового спектра нет смысла (рис. 1.21).

## Гауссов импульс

Следующий очень важный сигнал — гауссов импульс (рис. 1.22, слева). Как и предыдущий, он имеет бесконечную протяженность в обоих направлениях временной оси:

$$s(t) = Ae^{-a^2 t^2}.$$

Его спектр (вещественный, т. к. сигнал четный) тоже представляет собой гауссову функцию:

$$\hat{S}(\omega) = \int_{-\infty}^{\infty} Ae^{-a^2 t^2} e^{-j\omega t} dt = \frac{A\sqrt{\pi}}{a} \exp\left(-\frac{\omega^2}{4a^2}\right).$$

График амплитудного спектра приведен на рис. 1.22, справа.

Гауссов импульс имеет бесконечную протяженность как во временной, так и в частотной области. Определим его эффективную длительность и ширину спектра по

уровню  $1/e$  от максимума:  $\tau = 2/a$ ,  $\Delta\omega = 2a$ . База сигнала, таким образом, равна четырем.

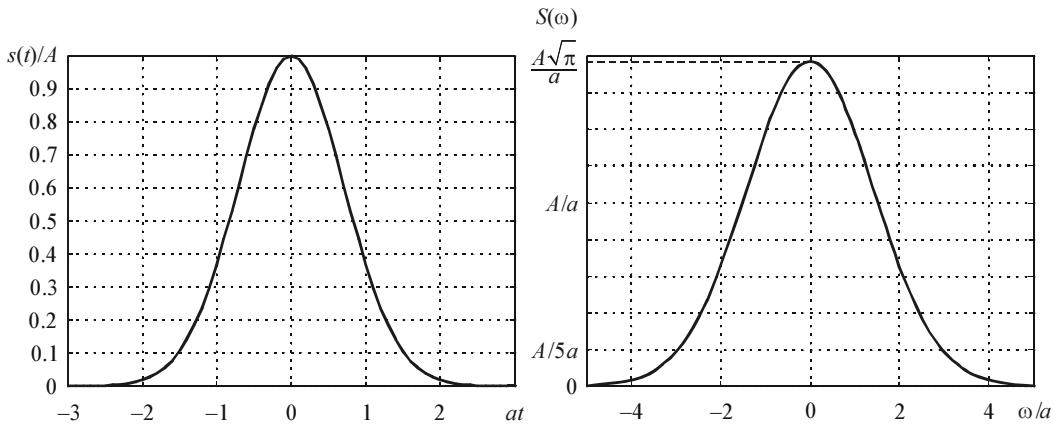


Рис. 1.22. Гауссов импульс (слева) и его амплитудный спектр (справа)

### Сигнал вида $\sin(x)/x$

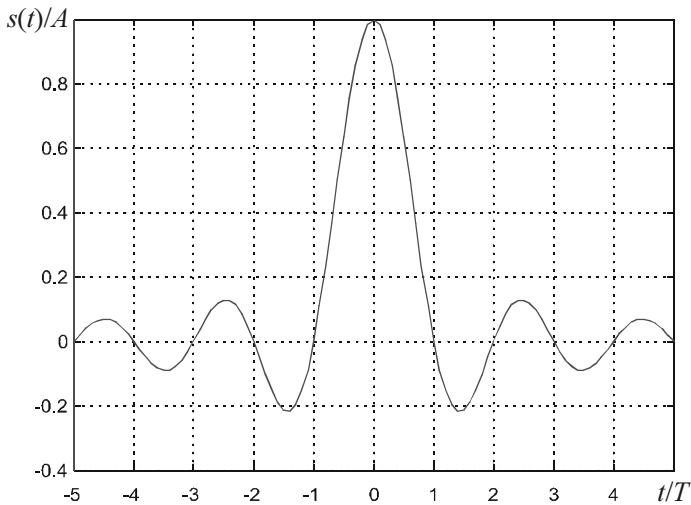
Следующий пример призван продемонстрировать *дuality* преобразования Фурье. Если сравнить формулы прямого и обратного преобразования Фурье, можно заметить, что они отличаются друг от друга лишь знаком в показателе комплексной экспоненты и множителем перед интегралом. Отсюда следует, что если четной функции времени  $f(t)$  соответствует спектральная функция  $g(\omega)$  (она будет также четной), то функции времени  $g(t)$  будет соответствовать спектральная функция  $2\pi f(\omega)$ . Проверим это на конкретном примере. В начале этого раздела мы выяснили, что прямоугольному импульсу соответствует спектральная функция вида  $\sin(\omega)/\omega$ . Теперь же рассмотрим временной сигнал вида  $\sin(t)/t$  и проверим, будет ли его спектральная функция прямоугольной.

Итак, задаем временной сигнал (используем параметр  $T$  для обозначения полупериода функции  $\sin$ ) (рис. 1.23):

$$s(t) = A \frac{\sin(\pi t / T)}{\pi t / T}.$$

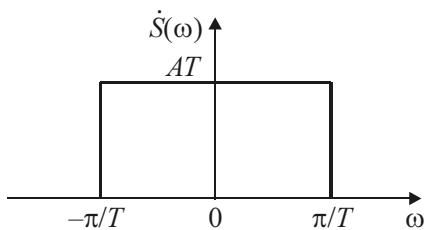
Рассчитываем спектр:

$$\begin{aligned} \dot{S}(\omega) &= \int_{-\infty}^{\infty} A \frac{\sin(\pi t / T)}{\pi t / T} e^{-j\omega t} dt = A \int_{-\infty}^{\infty} \frac{\sin(\pi t / T) \cos \omega t}{\pi t / T} dt = \\ &= \frac{AT}{2\pi} \int_{-\infty}^{\infty} \frac{\sin\left(\omega + \frac{\pi}{T}\right)t + \sin\left(\omega - \frac{\pi}{T}\right)t}{t} dt = \\ &= \frac{AT}{2\pi} \int_{-\infty}^{\infty} \frac{\sin\left(\omega + \frac{\pi}{T}\right)t}{t} dt + \frac{AT}{2\pi} \int_{-\infty}^{\infty} \frac{\sin\left(\omega - \frac{\pi}{T}\right)t}{t} dt. \end{aligned}$$

Рис. 1.23. Сигнал вида  $\sin(at)/(at)$ 

Значение каждого из двух получившихся интегралов равно  $\pm\pi$  в зависимости от знака множителей  $(\omega \pm \pi/T)$ . Поэтому результат суммирования интегралов зависит от частоты следующим образом (график показан на рис. 1.24):

$$\dot{S}(\omega) = \begin{cases} AT, & |\omega| \leq \frac{\pi}{T}, \\ 0, & |\omega| > \frac{\pi}{T}. \end{cases}$$

Рис. 1.24. Сигнал вида  $\sin(at)/(at)$   
имеет прямоугольный спектр

Как видите, дуальность (симметрия) преобразования Фурье получила наглядное подтверждение.

Сигнал данного вида имеет идеальный низкочастотный спектр — спектральная функция постоянна в некоторой полосе частот, начинающейся от нулевой частоты, и равна нулю за пределами этой полосы. Мы вновь встретимся с этим сигналом в главе 3 при обсуждении разложения сигналов в ряд Котельникова.

### ЗАМЕЧАНИЕ

Для данного сигнала не выполняется условие абсолютной интегрируемости (1.18). Вследствие этого интеграл Фурье для него не сходится на двух частотах, соответствующих краям прямоугольного спектра:  $\omega = \pm\pi/T$ .

## Свойства преобразования Фурье

Под свойствами преобразования Фурье подразумевается взаимное соответствие трансформаций сигналов и их спектров. Хорошее знание свойств преобразования Фурье позволяет предсказывать примерный (а иногда и точный) вид спектра анализируемого сигнала и таким образом контролировать правдоподобность результата, выдаваемого компьютером.

В этом разделе мы будем рассматривать два абстрактных сигнала,  $f(t)$  и  $g(t)$ , и считать, что их спектральные функции равны  $\dot{F}(\omega)$  и  $\dot{G}(\omega)$ .

### Линейность

Преобразование Фурье является *линейным* интегральным преобразованием. Смысл свойства линейности можно сформулировать так: спектр суммы равен сумме спектров. Говоря математическим языком, линейная комбинация сигналов имеет спектр в виде линейной комбинации (с теми же коэффициентами) их спектральных функций:

$$\text{если } s(t) = \alpha f(t) + \beta g(t), \text{ то } \dot{S}(\omega) = \alpha \dot{F}(\omega) + \beta \dot{G}(\omega).$$

### Задержка

А теперь посмотрим, как оказывается на спектральной функции задержка сигнала во времени. Итак, пусть  $\tau$  — время задержки:

$$s(t) = f(t - \tau),$$

Тогда спектральная функция изменится следующим образом:

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} f(t - \tau) e^{-j\omega t} dt = \int_{-\infty}^{\infty} f(t - \tau) e^{-j\omega(t - \tau)} d(t - \tau) e^{-j\omega\tau} = \dot{F}(\omega) e^{-j\omega\tau}.$$

Результат показывает, что спектр исходного сигнала оказался умноженным на комплексную экспоненту вида  $e^{-j\omega\tau}$ . Таким образом, амплитудный спектр сигнала не меняется (ведь модуль такой комплексной экспоненты равен 1; к тому же здравый смысл подсказывает, что соотношение между *амплитудами* спектральных составляющих из-за сдвига сигнала во времени измениться не должно). Фазовый спектр приобретает дополнительное слагаемое  $-\omega\tau$ , линейно зависящее от частоты.

### ЗАМЕЧАНИЕ

Если в результате какого-либо преобразования сигнала его спектр умножается на некоторую функцию, не зависящую от преобразуемого сигнала, это означает, что данное преобразование может быть выполнено линейной системой с постоянными параметрами. Речь о системах данного класса пойдет в главе 2.

### Изменение масштаба оси времени

Рассматривая конкретные примеры, мы уже познали на практике общее правило: чем короче сигнал, тем шире его спектр. Теперь взглянем на это правило со строгих

теоретических позиций. Если изменить длительность сигнала  $f(t)$ , сохраняя его форму, то новый сигнал  $s(t)$  следует записать как

$$s(t) = f(at).$$

При  $|a| > 1$  сигнал сжимается, при  $|a| < 1$  — растягивается. Если  $a < 0$ , дополнительно происходит зеркальное отражение сигнала относительно вертикальной оси. Посмотрим, как такое преобразование оказывается на спектре:

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} f(at)e^{-j\omega t} dt = \frac{1}{a} \int_{-\infty}^{\infty} f(at)e^{-j\frac{\omega}{a}at} d(at) = \frac{1}{a} \dot{F}\left(\frac{\omega}{a}\right).$$

Итак, изменение длительности сигнала приводит к изменению ширины спектра в противоположную сторону (аргумент  $t$  на  $a$  умножается, а  $\omega$  делится) в сочетании с увеличением (при растяжении,  $a < 1$ ) или уменьшением (при сжатии,  $a > 1$ ) уровня спектральных составляющих.

Полученная формула справедлива для  $a > 0$ . При  $a < 0$  использованная замена переменной  $t \rightarrow at$  вызовет перестановку пределов интегрирования и, как следствие, изменение знака у результата:

$$\dot{S}(\omega) = -\frac{1}{a} \dot{F}\left(\frac{\omega}{a}\right), \quad a < 0.$$

Объединяя оба случая, можно записать

$$\dot{S}(\omega) = \frac{1}{|a|} \dot{F}\left(\frac{\omega}{a}\right), \quad a \neq 0.$$

В частном случае  $a = -1$  полученная формула дает следующее:

$$\dot{S}(\omega) = \dot{F}(-\omega) = \dot{F}^*(\omega).$$

Итак, зеркальное отражение сигнала относительно начала отсчета времени приводит к зеркальному отражению спектра относительно нулевой частоты. Для вещественного сигнала это соответствует комплексному сопряжению спектра.

### ЗАМЕЧАНИЕ

В данном случае результат не сводится к умножению исходного спектра на некоторую функцию. В соответствии с предыдущим замечанием это означает, что изменение длительности сигнала *не может* быть осуществлено линейной системой с постоянными параметрами.

## Дифференцирование сигнала

Посмотрим, как влияет на спектр дифференцирование сигнала во временной области. Для этого продифференцируем обе части формулы обратного преобразования Фурье (1.15) по времени:

$$f(t) = \frac{ds}{dt} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{S}(\omega) j\omega e^{j\omega t} d\omega.$$

Полученная формула показывает, что производная сигнала по времени получается обратным преобразованием Фурье от спектра, имеющего вид

$$\dot{F}(\omega) = j\omega \dot{S}(\omega).$$

Итак, спектр производной получается путем умножения спектра исходного сигнала на  $j\omega$ . Таким образом, при дифференцировании низкие частоты ослабляются, а высокие усиливаются. Фазовый спектр сигнала сдвигается на  $90^\circ$  для положительных частот и на  $-90^\circ$  для отрицательных.

Множитель  $j\omega$  называют *оператором дифференцирования сигнала в частотной области*.

## Интегрирование сигнала

Интегрирование, как известно, является операцией, обратной дифференцированию. Поэтому, исходя из результатов, полученных в предыдущем разделе, казалось бы, можно ожидать следующий результат:

$$\text{если } s(t) = \int_{-\infty}^t f(t')dt', \text{ то } \dot{S}(\omega) = \frac{\dot{F}(\omega)}{j\omega}.$$

Однако детальный анализ, выполненный, например, в [1], показывает, что эта формула справедлива лишь для сигналов, не содержащих постоянной составляющей, у которых

$$\dot{F}(0) = \int_{-\infty}^{\infty} f(t)dt = 0.$$

В общем же случае результат должен содержать дополнительное слагаемое в виде дельта-функции на нулевой частоте. Множитель перед дельта-функцией пропорционален постоянной составляющей сигнала:

$$\dot{S}(\omega) = \frac{\dot{F}(\omega)}{j\omega} + \pi \dot{F}(0)\delta(\omega). \quad (1.21)$$

Итак, при интегрировании исходного сигнала высокие частоты ослабляются, а низкие усиливаются. Фазовый спектр сигнала смещается на  $-90^\circ$  для положительных частот и на  $90^\circ$  для отрицательных. Множитель  $1/(j\omega)$  называют *оператором интегрирования в частотной области*.

## Спектр свертки сигналов

Свертка сигналов является очень часто используемой в радиотехнике интегральной операцией, поскольку она описывает, в частности, прохождение сигнала через линейную систему с постоянными параметрами (подробнее это будет обсуждаться в главе 2):

$$s(t) = \int_{-\infty}^{\infty} f(t')g(t-t')dt'.$$

Подвернем такую конструкцию преобразованию Фурье:

$$\begin{aligned}\dot{S}(\omega) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(t')g(t-t')dt'e^{-j\omega t}dt = \\ &= \int_{-\infty}^{\infty} f(t')e^{-j\omega t'} \int_{-\infty}^{\infty} g(t-t')e^{-j\omega(t-t')}d(t-t')dt' = \dot{F}(\omega)\dot{G}(\omega).\end{aligned}\quad (1.22)$$

Полученный результат очень важен, он часто используется на практике: *спектр свертки равен произведению спектров*.

## Спектр произведения сигналов

Дуальность преобразования Фурье и соотношение (1.22), полученное в предыдущем разделе, позволяют легко предугадать результат. Однако все-таки получим его:

$$s(t) = f(t)g(t),$$

тогда

$$\begin{aligned}\dot{S}(\omega) &= \int_{-\infty}^{\infty} f(t)g(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty} \left( \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{F}(\omega')e^{j\omega't}d\omega' \right) g(t)e^{-j\omega t}dt = \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{F}(\omega') \int_{-\infty}^{\infty} g(t)e^{-j(\omega-\omega')t}dt d\omega' = \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{F}(\omega')\dot{G}(\omega-\omega')d\omega'.\end{aligned}\quad (1.23)$$

Как и следовало ожидать, *спектр произведения представляет собой свертку спектров*. Единственной дополнительной тонкостью является множитель  $1/(2\pi)$  перед интегралом свертки, он появляется из-за использования в формуле круговой частоты  $\omega$ .

### ЗАМЕЧАНИЕ

При выводе соотношения (1.23) мы представили сигнал  $f(t)$  с помощью обратного преобразования Фурье (1.15) от его спектральной функции.

## Умножение сигнала на гармоническую функцию

Умножим исходный сигнал, спектр которого нам известен, на гармоническую функцию:

$$s(t) = f(t)\cos(\omega_0 t + \phi_0).$$

Посмотрим, что произошло со спектром сигнала:

$$\begin{aligned}\dot{S}(\omega) &= \int_{-\infty}^{\infty} f(t)\cos(\omega_0 t + \phi_0)e^{-j\omega t}dt = \int_{-\infty}^{\infty} f(t) \frac{e^{j\omega_0 t + j\phi_0} + e^{-j\omega_0 t - j\phi_0}}{2} e^{-j\omega t}dt = \\ &= \frac{1}{2} \int_{-\infty}^{\infty} f(t)e^{j\phi_0}e^{-j(\omega-\omega_0)t}dt + \frac{1}{2} \int_{-\infty}^{\infty} f(t)e^{-j\phi_0}e^{-j(\omega+\omega_0)t}dt = \\ &= \frac{1}{2} e^{j\phi_0} \dot{F}(\omega - \omega_0) + \frac{1}{2} e^{-j\phi_0} \dot{F}(\omega + \omega_0).\end{aligned}\quad (1.24)$$

Как видите, спектр "раздвоился" — распался на два слагаемых вдвое меньшего уровня (множитель  $1/2$ ), смещенных на  $\omega_0$  вправо ( $\omega - \omega_0$ ) и влево ( $\omega + \omega_0$ ) по оси частот. Кроме того, при каждом слагаемом имеется множитель, учитывающий начальную фазу гармонического колебания. С практическим применением этого свойства мы столкнемся в главе 8 при обсуждении свойств сигналов с амплитудной модуляцией.

## Связь преобразования Фурье и коэффициентов ряда Фурье

Пусть  $s(t)$  — сигнал конечной длительности, а  $\dot{S}(\omega)$  — его спектральная функция. Получим на основе  $s(t)$  периодический сигнал, взяв период повторения  $T$  не меньше длительности сигнала:

$$s_T(t) = \sum_{k=-\infty}^{\infty} s(t - kT).$$

Сравнивая формулы (1.14) для расчета преобразования Фурье сигнала  $s(t)$  и (1.12) для расчета коэффициентов ряда Фурье сигнала  $s_T(t)$ , можно заметить, что эти формулы предполагают вычисление одного и того же интеграла. Различие состоит в том, что для расчета коэффициентов ряда Фурье в подынтегральное выражение подставляются не произвольные, а дискретные значения частоты  $\omega_k = 2\pi k/T$  и, кроме того, результат интегрирования делится на период сигнала  $T$ .

Таким образом, между спектральной функцией  $\dot{S}(\omega)$  одиночного импульса и коэффициентами  $\dot{C}_k$  ряда Фурье для периодической последовательности таких импульсов существует простая связь:

$$\dot{C}_k = \frac{1}{T} \dot{S}\left(\frac{2\pi k}{T}\right).$$

### ЗАМЕЧАНИЕ

Данная формула справедлива и в том случае, если период повторения импульсов меньше их длительности (т. е. если соседние импульсы периодической последовательности перекрываются).

## Фурье-анализ неинтегрируемых сигналов

При введении понятия преобразования Фурье были указаны условия его применимости: выполнение условий Дирихле и абсолютная интегрируемость сигнала. Однако в ряде случаев можно применить преобразование Фурье и к сигналам, этим условиям не удовлетворяющим, и получить при этом вполне осмысленный и практически полезный результат.

Итак, в данном разделе мы воспользуемся преобразованием Фурье для спектрального анализа таких сигналов, к которым оно формально неприменимо.

## Дельта-функция

Прежде всего вычислим преобразование Фурье для сигнала в виде дельта-функции (о ее свойствах шла речь в разд. "Классификация сигналов", и фильтрующее свойство (1.2) нам сейчас как раз понадобится):

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} \delta(t) e^{-j\omega t} dt = 1.$$

Спектр дельта-функции представляет собой константу, т. е. является равномерным в бесконечной полосе частот. Это вполне согласуется с общим соотношением между длительностью сигнала и шириной его спектра: дельта-импульс имеет бесконечно малую длительность, а его спектр бесконечно широк.

Из полученного результата следует, что дельта-функцию можно записать в виде обратного преобразования Фурье следующим образом:

$$\delta(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{j\omega t} d\omega. \quad (1.25)$$

Это полезное соотношение мы используем при анализе следующего сигнала.

## Постоянный во времени сигнал (константа)

Поскольку мы уже знаем, что спектром дельта-функции является константа, благодаря дуальности преобразования Фурье можно сразу же сказать, что спектром константы  $s(t) = A$  будет дельта-функция частоты. Проверим это, воспользовавшись только что полученным соотношением (1.25):

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} A e^{-j\omega t} dt = 2\pi A \delta(\omega). \quad (1.26)$$

Наши предположения полностью подтвердились. Здесь опять хорошо прослеживается обратная пропорциональность между длительностью сигнала и шириной его спектра: бесконечно протяженный сигнал имеет бесконечно узкий спектр.

## Функция единичного скачка

Функция единичного скачка (1.3) (см. разд. "Классификация сигналов") представляет собой интеграл от дельта-функции, поэтому, в соответствии со свойствами преобразования Фурье (см. предыдущий раздел), мы получаем

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} \delta_1(t) e^{-j\omega t} dt = \pi \delta(\omega) + \frac{1}{j\omega}.$$

Поскольку спектр дельта-функции на нулевой частоте равен единице, в соответствии с формулой (1.21) в спектре появляется дополнительное слагаемое в виде дельта-функции на нулевой частоте.

## Гармонический сигнал

Рассчитаем спектр гармонического сигнала общего вида:

$$s(t) = A \cos(\omega_0 t + \phi_0).$$

Данный сигнал можно представить как умножение константы  $A$  на гармоническую функцию, после чего применение формул (1.26) и (1.24) сразу же дает окончательный результат:

$$\dot{S}(\omega) = A\pi e^{j\phi_0} \delta(\omega - \omega_0) + A\pi e^{-j\phi_0} \delta(\omega + \omega_0). \quad (1.27)$$

Результат представляет собой пару дельта-функций, расположенных на частотах  $\pm\omega_0$ . Множители при них отражают амплитуду и начальную фазу (т. е. комплексную амплитуду) гармонического сигнала.

## Комплексная экспонента

Впервые в этой книге мы рассматриваем сигнал, не являющийся вещественным:

$$s(t) = A \exp(j\omega_0 t).$$

Результат вычисления его спектра легко предугадать: только что рассмотренный гармонический сигнал дал спектральную функцию в виде двух дельта-функций, а косинус с помощью формулы Эйлера можно представить в виде полусуммы двух комплексных экспонент. Значит, спектром комплексной экспоненты должна являться *одиночная дельта-функция*:

$$\dot{S}(\omega) = \int_{-\infty}^{\infty} A e^{j\omega_0 t} e^{-j\omega t} dt = 2A\pi\delta(\omega - \omega_0). \quad (1.28)$$

Результат, как видите, не обманул наших ожиданий. Обратите внимание на то, что, поскольку сигнал не является вещественным, его спектр теряет свойство симметрии.

На первый взгляд, польза от рассмотрения комплексных сигналов невелика. Однако они оказываются очень удобным средством для анализа модулированных сигналов, особенно когда у них одновременно меняются и амплитуда, и начальная фаза. С такими сигналами нам предстоит познакомиться далее в разд. "Комплексная огибающая" этой главы).

## Произвольный периодический сигнал

Как мы уже знаем, периодический сигнал с периодом  $T$  может быть представлен в виде ряда Фурье (1.10):

$$s(t) = \sum_{k=-\infty}^{\infty} \dot{C}_k e^{\frac{j2\pi k}{T}t}.$$

После вычисления спектров гармонического сигнала (1.27) и комплексной экспоненты (1.28) читателю уже должно быть ясно, что спектральная функция такого

сигнала представляет собой набор дельта-функций, расположенных на частотах гармоник ряда Фурье:

$$\dot{S}(\omega) = \sum_{k=-\infty}^{\infty} 2\pi \dot{C}_k \delta\left(\omega - \frac{2\pi k}{T}\right).$$

Множители при дельта-функциях равны соответствующим коэффициентам ряда Фурье  $\dot{C}_k$ , умноженным на  $2\pi$ .

## Корреляционный анализ

Корреляционный анализ, наряду со спектральным, играет большую роль в теории сигналов. Говоря кратко, его смысл состоит в количественном измерении *степени сходства* различных сигналов. Для этого служат корреляционные функции, с рассмотрения которых мы и начнем этот раздел.

### Корреляционная функция

*Корреляционная функция* (*КФ*; английский термин — correlation function, CF) детерминированного сигнала с конечной энергией представляет собой интеграл (в бесконечных пределах) от произведения двух копий сигнала, сдвинутых друг относительно друга на время  $\tau$ :

$$B_s(\tau) = \int_{-\infty}^{\infty} s(t)s(t-\tau)dt .$$

Корреляционная функция показывает степень сходства между сигналом и его сдвинутой копией — чем больше значение корреляционной функции, тем это сходство сильнее. Кроме того, корреляционная функция обладает следующими свойствами:

1. Значение КФ при  $\tau = 0$  равно *энергии сигнала*, т. е. интегралу от его квадрата:

$$B_s(0) = \int_{-\infty}^{\infty} s^2(t)dt = E .$$

2. КФ является четной функцией своего аргумента  $\tau$ :

$$B_s(-\tau) = B_s(\tau) .$$

3. Значение КФ при  $\tau = 0$  является максимально возможным значением:

$$|B_s(\tau)| \leq B_s(0) .$$

4. С ростом абсолютного значения  $\tau$  КФ сигнала с конечной энергией затухает:

$$\lim_{|\tau| \rightarrow \infty} B_s(\tau) = 0 .$$

5. Если сигнал  $s(t)$  не содержит особенностей в виде дельта-функций, его КФ не может иметь разрывов (т. е. обязана быть непрерывной функцией).
6. Если сигнал — напряжение, то размерность его КФ равна  $\text{В}^2 \cdot \text{с}$ .

В качестве примера вычислим КФ прямоугольного импульса (1.19), показанного ранее на рис. 1.12:

- при  $0 \leq \tau \leq T$   $B_s(\tau) = \int_{\tau}^T A^2 dt = A^2(T - \tau);$
- при  $-T \leq \tau < 0$   $B_s(\tau) = \int_0^{T+\tau} A^2 dt = A^2(T + \tau);$
- при  $|\tau| > T$   $B_s(\tau) = 0.$

Объединяя результаты, можно записать

$$B_s(\tau) = \begin{cases} A^2(T - |\tau|), & |\tau| \leq T, \\ 0, & |\tau| > T. \end{cases}$$

График КФ прямоугольного импульса показан на рис. 1.25.

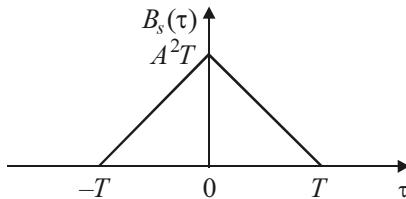


Рис. 1.25. Корреляционная функция прямоугольного импульса

В случае периодического сигнала (и вообще любого сигнала с бесконечной энергией) воспользоваться приведенным определением не удастся. Поэтому КФ периодического сигнала с периодом  $T$  вычисляют, усредняя произведение сдвинутых копий в пределах одного периода:

$$B_s(\tau) = \frac{1}{T} \int_{-T/2}^{T/2} s(t)s(t - \tau)dt.$$

Набор свойств такой КФ несколько меняется:

1. Значение при  $\tau = 0$  равно не энергии, а *средней мощности* анализируемого сигнала:

$$B_s(0) = \frac{1}{T} \int_{-T/2}^{T/2} s^2(t)dt = P_{\text{cp}}.$$

2. Свойство четности сохраняется:  $B_s(\tau) = B_s(-\tau).$

3. Значение КФ при  $\tau = 0$  по-прежнему является максимально возможным:  $|B_s(\tau)| \leq B_s(0)$ .
4. КФ периодического сигнала является периодической функцией с тем же периодом, что и сам сигнал:  $B_s(\tau + T) = B_s(\tau)$ .
5. Если сигнал не содержит дельта-функций, его КФ будет непрерывной функцией.
6. Размерность КФ периодического сигнала — квадрат размерности сигнала ( $B^2$ , если сигнал — напряжение).

В качестве примера вычислим КФ гармонического сигнала с частотой  $\omega_0$ :

$$s(t) = A \cos(\omega_0 t + \phi_0).$$

Вычисляем корреляционный интеграл, учитывая, что период такого сигнала равен  $2\pi/\omega_0$ :

$$B_s(\tau) = \frac{\omega_0}{2\pi} \int_{-\pi/\omega_0}^{\pi/\omega_0} A \cos(\omega_0 t + \phi_0) A \cos(\omega_0(t - \tau) + \phi_0) dt = \frac{A^2}{2} \cos(\omega_0 \tau).$$

Как видите, КФ гармонического сигнала тоже является гармонической функцией. Еще очень важен тот факт, что полученный результат не зависит от начальной фазы гармонического сигнала (параметр  $\phi$  в полученное выражение не вошел). Это проявление общего свойства всех КФ, о котором речь пойдет далее в разд. "Связь между корреляционными функциями и спектрами сигналов" этой главы.

## Взаимная корреляционная функция

Если КФ показывает степень сходства между сдвинутыми копиями *одного и того же* сигнала, то *взаимная корреляционная функция* (ВКФ; английский термин — cross-correlation function, CCF) позволяет измерить аналогичную величину для сдвинутых экземпляров двух *разных* сигналов.

Общий вид формулы КФ сохраняется, но под интегралом стоит произведение двух разных сигналов, один из которых задержан на время  $\tau$ :

$$B_{12}(\tau) = \int_{-\infty}^{\infty} s_1(t) s_2(t - \tau) dt.$$

### ЗАМЕЧАНИЕ

Очевидно, КФ является частным случаем ВКФ, когда оба сигнала одинаковы:  $s_1(t) = s_2(t) = s(t)$ .

В качестве примера вычислим ВКФ прямоугольного и треугольного импульсов (см. рис. 1.12 и 1.14):

$$s_1(t) = \begin{cases} A, & 0 \leq t \leq T, \\ 0, & t < 0, t > T. \end{cases} \quad s_2(t) = \begin{cases} A \frac{t}{T}, & 0 \leq t \leq T, \\ 0, & t < 0, t > T. \end{cases}$$

□ при  $0 \leq \tau \leq T$   $B_{12}(\tau) = \int_{\tau}^T A^2 \frac{t-\tau}{T} dt = \frac{A^2}{2T}(T-\tau)^2$ ;

□ при  $-T \leq \tau \leq 0$   $B_{12}(\tau) = \int_0^{-\tau} A^2 \frac{t-\tau}{T} dt = \frac{A^2}{2T}(T^2 - \tau^2)$ ;

□ при  $|\tau| > T$   $B_{12}(\tau) = 0$ .

Объединяя результаты, можно записать

$$B_{12}(\tau) = \begin{cases} \frac{A^2}{2T}(T-\tau)^2, & 0 \leq \tau \leq T, \\ \frac{A^2}{2T}(T^2 - \tau^2), & -T \leq \tau < 0, \\ 0, & |\tau| > T. \end{cases}$$

График полученной ВКФ представлен на рис. 1.26.

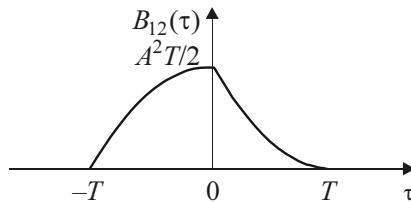


Рис. 1.26. ВКФ прямоугольного и треугольного импульсов

Свойства ВКФ несколько отличаются от свойств КФ:

- Значение ВКФ при  $\tau = 0$  ничем не выделяется; максимум может быть расположен в любом месте оси  $\tau$ .
- $B_{12}(-\tau) = B_{21}(\tau)$ , т. е. изменение знака  $\tau$  равносильно взаимной перестановке сигналов.
- $|B_{12}(\tau)| \leq \sqrt{E_1 E_2}$ , где  $E_1$  и  $E_2$  — энергии сигналов  $s_1(t)$  и  $s_2(t)$ . Это следует из рассматриваемого далее, в разд. "Пространство сигналов", неравенства (1.41).
- С ростом абсолютного значения  $\tau$  ВКФ сигналов с конечной энергией затухает:

$$\lim_{|\tau| \rightarrow \infty} B_{12}(\tau) = 0$$

- Если сигналы  $s_1(t)$  и  $s_2(t)$  не содержат особенностей в виде дельта-функций, их ВКФ не может иметь разрывов (т. е. обязана быть непрерывной функцией).
- Если сигналы — напряжение, то размерность их ВКФ равна  $B^2 \cdot c$ .

Для периодических сигналов понятие ВКФ обычно не применяется, хотя оно может быть введено в случае, если сигналы  $s_1(t)$  и  $s_2(t)$  имеют одинаковый период.

## Связь между корреляционными функциями и спектрами сигналов

Поскольку как корреляционные функции, так и спектры являются интегральными преобразованиями анализируемых сигналов, логично предположить, что эти характеристики как-то связаны друг с другом. Для выявления этой связи подвернем взаимную корреляционную функцию преобразованию Фурье, считая, что сигналы  $s_1(t)$  и  $s_2(t)$  имеют спектральные функции  $\dot{S}_1(\omega)$  и  $\dot{S}_2(\omega)$ :

$$\begin{aligned} \int_{-\infty}^{\infty} B_{12}(\tau) e^{-j\omega\tau} d\tau &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s_1(t)s_2(t-\tau) e^{-j\omega\tau} dt d\tau = \\ &= \int_{-\infty}^{\infty} s_1(t) e^{-j\omega t} \int_{-\infty}^{\infty} s_2(t-\tau) e^{j\omega(t-\tau)} d(t-\tau) dt = \dot{S}_1(\omega) \dot{S}_2^*(\omega). \end{aligned}$$

Полученный результат очень прост: ВКФ связана преобразованием Фурье с так называемым *взаимным спектром* сигналов, определяемым следующим образом:

$$\dot{S}_{12}(\omega) = \dot{S}_1(\omega) \dot{S}_2^*(\omega). \quad (1.29)$$

Отсюда можно сделать очень важный вывод: чтобы ВКФ сигналов была равна нулю *при любых временных сдвигах*  $\tau$ , необходимо и достаточно, чтобы был равен нулю их взаимный спектр, т. е. чтобы их спектры *не перекрывались на частотной оси*.

Приняв  $s_1(t) = s_2(t) = s(t)$ , получаем аналогичный результат для КФ, показывающий, что она связана преобразованием Фурье с квадратом модуля спектральной функции, т. е. с *энергетическим спектром* сигнала:

$$\begin{aligned} \int_{-\infty}^{\infty} B_s(\tau) e^{-j\omega\tau} d\tau &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(t)s(t-\tau) e^{-j\omega\tau} dt d\tau = \\ &= \int_{-\infty}^{\infty} s(t) e^{-j\omega t} \int_{-\infty}^{\infty} s(t-\tau) e^{j\omega(t-\tau)} d(t-\tau) dt = \\ &= \dot{S}(\omega) \dot{S}^*(\omega) = |\dot{S}(\omega)|^2. \end{aligned} \quad (1.30)$$

Отсюда следует еще один важный факт: *КФ сигнала не зависит от его фазового спектра*. Следовательно, сигналы, амплитудные спектры которых одинаковы, а фазовые различаются, будут иметь одинаковую КФ. Еще одно следствие заключается в том, что по КФ нельзя восстановить исходный сигнал (опять же из-за утраты информации о фазе).

## Энергетические расчеты в спектральной области

Ранее в разд. "Связь между корреляционными функциями и спектрами сигналов" мы показали, что ВКФ двух сигналов связана преобразованием Фурье с их взаимным спектром. Запишем эту связь в виде формулы обратного преобразования Фурье:

$$B_{12}(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{S}_{12}(\omega) e^{j\omega\tau} d\omega.$$

Теперь подставим в эту формулу значение  $\tau = 0$  и раскроем выражения для ВКФ и взаимного спектра. Получится соотношение, именуемое *теоремой Рэлея*:

$$\int_{-\infty}^{\infty} s_1(t)s_2(t)dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \dot{S}_1(\omega) \dot{S}_2^*(\omega) d\omega \quad (1.31)$$

Если теперь принять сигналы одинаковыми ( $s_1(t) = s_2(t) = s(t)$ ), получится соотношение, позволяющее вычислять энергию сигнала как во временной, так и в частотной области и называемое *равенством Парсеваля*:

$$E = \int_{-\infty}^{\infty} s^2(t)dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\dot{S}(\omega)|^2 d\omega. \quad (1.32)$$

Последнее, на чем следует остановиться в этом разделе, — это вычисление средней мощности периодического сигнала по коэффициентам его ряда Фурье. Запишем периодический сигнал  $s(t)$  в виде ряда Фурье в комплексной форме (1.10):

$$s(t) = \sum_{k=-\infty}^{\infty} \dot{C}_k e^{j\frac{2\pi k}{T}t}.$$

А теперь применим к этому выражению формулу для расчета средней мощности за период:

$$P_{cp} = \frac{1}{T} \int_0^T s^2(t)dt = \frac{1}{T} \sum_{k=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \dot{C}_k \dot{C}_m \int_0^T e^{j\frac{2\pi(k+m)}{T}t} dt.$$

Промежуток  $0...T$  соответствует целому числу периодов стоящей под интегралом комплексной экспоненты, поэтому интеграл будет равен нулю при всех  $k \neq -m$ . При  $k = -m$  экспонента становится константой, и интеграл будет равен  $T$ :

$$P_{cp} = \sum_{k=-\infty}^{\infty} |\dot{C}_k|^2.$$

Результат оказывается очень простым: *средняя мощность периодического сигнала равна сумме квадратов модулей коэффициентов его ряда Фурье*.

## Комплексная огибающая

В различных системах передачи информации часто применяются узкополосные сигналы, спектр которых сосредоточен в окрестности некоторой частоты  $\omega_0$ . При

анализе таких сигналов удобно пользоваться понятиями *комплексной огибающей*, *амплитудной огибающей* и *фазовой функции* сигнала. Эти понятия и будут рассмотрены в данном разделе.

Рассмотрим сигнал, представленный в виде колебаний с частотой  $\omega_0$ , у которых меняются во времени как амплитуда, так и начальная фаза:

$$s(t) = A(t) \cos(\omega_0 t + \phi(t)). \quad (1.33)$$

Множитель  $A(t)$  называется *амплитудной огибающей*, а начальная фаза  $\phi(t)$  — *фазовой функцией* сигнала  $s(t)$ . Весь аргумент функции  $\cos$  называют *полной фазой* сигнала:

$$\Psi(t) = \omega_0 t + \phi(t).$$

Сигнал (1.33) можно представить как вещественную часть комплексной функции, заменив косинус комплексной экспонентой:

$$s(t) = \operatorname{Re} \left( A(t) \exp(j(\omega_0 t + \phi(t))) \right).$$

В комплексном выражении, стоящем под функцией  $\operatorname{Re}$ , можно выделить два множителя:  $\exp(j\omega_0 t)$  представляет немодулированное несущее колебание и является быстро меняющимся, а  $A(t) \exp(j\phi(t))$  меняется, как правило, значительно медленнее и содержит информацию об амплитудной огибающей и начальной фазе одновременно. Этот медленно меняющийся множитель и называется *комплексной огибающей* сигнала:

$$\dot{A}_m(t) = A(t) \exp(j\phi(t)).$$

### **ЗАМЕЧАНИЕ**

Комплексная огибающая, объединяя в себе информацию об амплитуде и фазе сигнала, является обобщением понятия *комплексной амплитуды*, широко используемого в теоретической электротехнике.

Рассмотрим теперь другую задачу — представим произвольный сигнал  $s(t)$  в форме (1.33), т. е. выделим его амплитудную огибающую и фазовую функцию. Ясно, что способов сделать это бесконечно много, поскольку мы хотим одной функции  $s(t)$  поставить в соответствие набор из двух функций  $A(t)$  и  $\phi(t)$ . Однако искомое представление должно удовлетворять нескольким очевидным ограничениям. В частности, для гармонического сигнала  $s(t) = A \cos(\omega_0 t + \phi_0)$  искомая процедура должна давать  $A(t) = A$  и  $\Psi(t) = \omega_0 t + \phi_0$ . Кроме того, разумно потребовать, что фазовая функция не должна меняться при умножении сигнала на произвольный постоянный множитель. С учетом этих требований способ выделения амплитудной огибающей и фазовой функции из произвольного сигнала оказывается единственным: такое выделение производится с помощью преобразования Гильберта, речь о котором пойдет в следующем разделе.

### **ЗАМЕЧАНИЕ**

Подробное и интересное обсуждение данной проблемы можно найти в [6]. Вообще, с этой замечательной книгой, рассказывающей о различных парадоксах и заблужде-

ниях, связанных с теорией связи, должен познакомиться каждый, кто занимается обработкой сигналов, радиотехникой или телекоммуникациями.

## Преобразование Гильберта

Для выделения амплитуды и фазы произвольный сигнал  $s(t)$  представляется как вещественная часть комплексного сигнала  $\dot{s}_a(t)$  (он называется *аналитическим сигналом*):

$$s(t) = \operatorname{Re}(\dot{s}_a(t)).$$

Вещественная часть аналитического сигнала, естественно, должна совпадать с исходным сигналом  $s(t)$ . Мнимая же часть  $s_\perp(t)$  называется *сопряженным сигналом* или *квадратурным дополнением*:

$$\dot{s}_a(t) = s(t) + js_\perp(t).$$

Сопряженный сигнал получается из исходного с помощью *преобразования Гильберта*, вычисляемого следующим образом:

$$s_\perp(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{s(t')}{t - t'} dt'. \quad (1.34)$$

Данный интеграл представляет собой свертку сигнала  $s(t)$  и функции  $1/(\pi t)$ . Это означает, что преобразование Гильберта может быть выполнено линейной системой с постоянными параметрами (такие системы будут описаны в главе 2; забегая вперед, скажем, что система, осуществляющая преобразование Гильберта, является физически нереализуемой, поскольку ее импульсная характеристика имеет бесконечную протяженность в обоих направлениях временной оси). Из этого, в свою очередь, следует, что мы можем определить *частотную характеристику* преобразования Гильберта:

$$\dot{K}_\perp(\omega) = \int_{-\infty}^{\infty} \frac{1}{\pi t} e^{-j\omega t} dt = \begin{cases} j, & \omega < 0, \\ 0, & \omega = 0, \\ -j, & \omega > 0. \end{cases} \quad (1.35)$$

Итак, амплитудно-частотная характеристика (АЧХ) преобразования Гильберта равна единице всюду, кроме нулевой частоты, т. е. преобразование Гильберта не меняет амплитудных соотношений в спектре сигнала, лишь удаляя из него постоянную составляющую. Фазы всех спектральных составляющих в области положительных частот уменьшаются на  $90^\circ$ , в области отрицательных частот — увеличиваются на  $90^\circ$ .

Таким образом, устройство, осуществляющее преобразование Гильберта, должно представлять собой идеальный *фазовращатель*, вносящий на всех частотах фазовый сдвиг, равный  $90^\circ$ .

Очевидно, что обратное преобразование Гильберта должно вносить такой же фазовый сдвиг, но с обратным знаком, опять же при сохранении амплитудных соотношений в спектре преобразуемого сигнала.

Математически это будет выглядеть так:

$$\dot{K}'_{\perp}(\omega) = \frac{1}{\dot{K}_{\perp}(\omega)} = \begin{cases} -j, & \omega < 0, \\ 0, & \omega = 0, \\ j, & \omega > 0. \end{cases} \quad (1.36)$$

Сравнение формул (1.36) и (1.35) показывает, что  $\dot{K}'_{\perp}(\omega) = -\dot{K}'_{\perp}(\omega)$ , следовательно, формулы обратного и прямого преобразований Гильберта различаются лишь знаком:

$$s(t) = -\frac{1}{\pi} \int_{-\infty}^{\infty} \frac{s_{\perp}(t')}{t-t'} dt'.$$

## Спектр аналитического сигнала

Ранее уже было сказано, что аналитический сигнал получается путем добавления к вещественному сигналу  $s(t)$  мнимой части в виде его преобразования Гильберта:

$$\dot{s}_a(t) = s(t) + js_{\perp}(t).$$

Вычислим спектр аналитического сигнала, учитывая, что преобразование Гильберта является линейным и его коэффициент передачи определяется формулой (1.35):

$$\dot{S}_a(\omega) = \dot{S}(\omega) + j\dot{S}_{\perp}(\omega) = \dot{S}(\omega)(1 + j\dot{K}_{\perp}(\omega)) = \begin{cases} 0, & \omega < 0, \\ \dot{S}(0), & \omega = 0, \\ 2\dot{S}(\omega), & \omega > 0. \end{cases}$$

Полученный результат довольно любопытен. В области положительных частот спектры вещественного сигнала и добавленной мнимой части (с учетом дополнительного 90-градусного фазового сдвига, вносимого множителем  $j$ ) складываются, давая удвоенный результат. В области же отрицательных частот эти спектры оказываются противофазными и взаимно уничтожаются. В результате спектр аналитического сигнала оказывается *односторонним* (рис. 1.27, а, б).

Итак, чтобы для произвольного сигнала определить амплитудную огибающую и фазовую функцию, необходимо прежде всего сформировать аналитический сигнал, получив его мнимую часть с помощью преобразования Гильберта. Далее амплитудная огибающая находится как модуль аналитического сигнала:

$$A(t) = |\dot{s}_a(t)| = \sqrt{s^2(t) + s_{\perp}^2(t)}.$$

Полная фаза представляет собой аргумент комплексного аналитического сигнала:

$$\Psi(t) = \arg \dot{s}_a(t).$$

Чтобы получить начальную фазу сигнала, нужно выделить из полной фазы линейное слагаемое  $\omega_0 t$ . Для этого, в свою очередь, нужно знать значение центральной частоты  $\omega_0$ . После этого можно будет получить начальную фазу и комплексную огибающую:

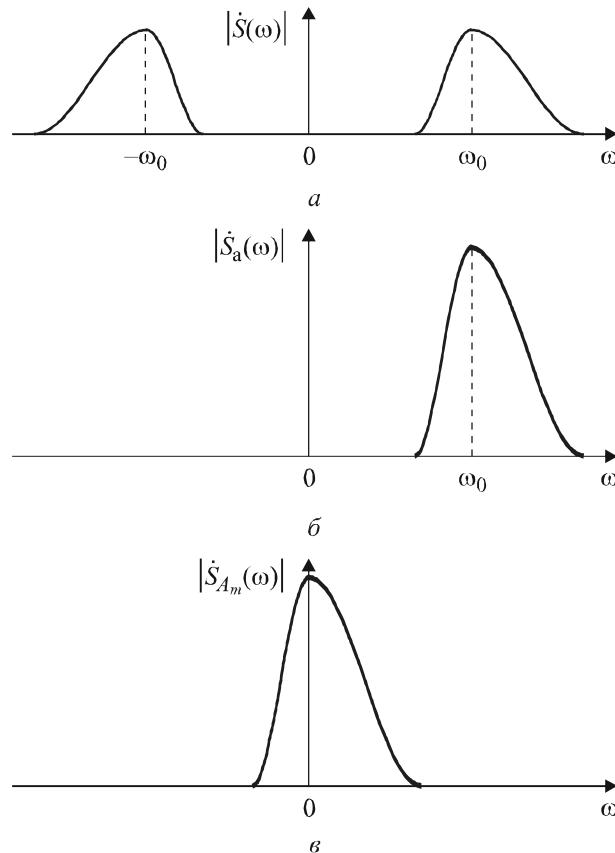
$$\varphi(t) = \Psi(t) - \omega_0 t,$$

$$\dot{A}_m(t) = A(t)e^{j\varphi(t)}.$$

Спектр комплексной огибающей представляет собой сдвинутый на  $\omega_0$  спектр аналитического сигнала:

$$\dot{S}_{A_m}(\omega) = \dot{S}_a(\omega + \omega_0).$$

В общем случае спектр комплексной огибающей не является симметричным относительно нулевой частоты (см. рис. 1.27, в).



**Рис. 1.27.** Амплитудные спектры вещественного сигнала (а), соответствующего ему аналитического сигнала (б) и его комплексной огибающей (в)

Выбор центральной частоты  $\omega_0$ , вообще говоря, является произвольным. Для узкополосных сигналов существует "разумное" значение  $\omega_0$ , при использовании которого аналитическая запись комплексной огибающей оказывается наиболее простой. Например, для гармонического сигнала

$$s(t) = A \cos(\Omega t + \varphi)$$

аналитический сигнал имеет вид

$$\dot{s}_a(t) = A \exp(j\Omega t + j\varphi).$$

Амплитудная огибающая равна  $A$ , полная фаза —  $\Omega t + \varphi$ . В общем случае, выбрав произвольное значение "средней" частоты  $\omega_0$ , мы получаем начальную фазу

$$\varphi(t) = (\Omega - \omega_0)t + \varphi$$

и комплексную огибающую

$$\dot{A}_m(t) = A \exp(j(\Omega - \omega_0)t + j\varphi).$$

Если выбранная "средняя" частота  $\omega_0$  будет совпадать с частотой гармонического сигнала ( $\omega_0 = \Omega$ ), комплексная огибающая станет константой:

$$\dot{A}_m(t) = A \exp(j\varphi).$$

Метод замены исходных функций их комплексными огибающими для анализа прохождения сигналов через различные цепи называется *методом низкочастотного эквивалента*. При этом принципиально то, что все комплексные огибающие должны вычисляться относительно одной и той же центральной частоты  $\omega_0$ , даже если ее значение для некоторых сигналов будет выглядеть "неестественным".

В целом же следует помнить, что понятие комплексной огибающей имеет смысл только при указании частоты  $\omega_0$ , относительно которой эта комплексная огибающая получена.

## Пространство сигналов

Чтобы анализировать свойства и преобразования сигналов максимально общим образом, в теории сигналов вводится в рассмотрение *пространство сигналов (signal space)* — множество, отдельными элементами которого являются сигналы, непрерывные или дискретные. Понятие "пространство" означает наделение элементов этого множества некоторыми дополнительными свойствами. В частности, различают следующие пространства (см., например, [5]):

- метрические пространства;
- линейные пространства;
- нормированные пространства;
- пространства со скалярным произведением.

## Метрическое пространство

*Метрическое пространство* — это множество сигналов, между которыми можно вычислять расстояния с помощью метрики. *Метрика (metric)* — это отображение пары сигналов  $(x, y)$  в вещественное число  $d(x, y)$ , обладающее следующими свойствами:

1.  $d(x, y) \geq 0$ , причем  $d(x, y) = 0$  только при  $x = y$ .
2.  $d(x, y) = d(y, x)$  (симметрия).
3.  $d(x, z) \leq d(x, y) + d(y, z)$  (неравенство треугольника).

Для непрерывных во времени сигналов наибольшее распространение получила  $p$ -метрика, согласно которой расстояние между сигналами  $x(t)$  и  $y(t)$  рассчитывается следующим образом:

$$d_p(x, y) = \sqrt[p]{\int_T |x(t) - y(t)|^p dt}. \quad (1.37)$$

Здесь  $T$  символизирует рассматриваемый промежуток времени.

Чаще всего используются три значения  $p$ : 1, 2 и  $\infty$ . При этом общая формула (1.37) принимает следующие частные формы:

$$\begin{aligned} d_1(x, y) &= \int_T |x(t) - y(t)| dt, \\ d_2(x, y) &= \sqrt{\int_T |x(t) - y(t)|^2 dt}, \\ d_\infty(x, y) &= \max_{t \in T} |x(t) - y(t)|. \end{aligned} \quad (1.38)$$

#### **ЗАМЕЧАНИЕ**

Метрика  $d_2(x, y)$ , определяемая формулой (1.38), называется *евклидовой*.

## **Линейное пространство**

*Линейное пространство* — это множество сигналов, наделенных векторными свойствами, т. е. для них определены операции *сложения* и *умножения на скаляр*.

#### **ЗАМЕЧАНИЕ**

В приводимых далее формулах векторы, согласно общепринятым соглашениям, обозначаются прямым полужирным шрифтом.

Операция сложения обладает следующими свойствами:

1.  $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$  для любых  $\mathbf{x}$  и  $\mathbf{y}$  (коммутативность).
2.  $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$  для любых  $\mathbf{x}$ ,  $\mathbf{y}$  и  $\mathbf{z}$  (ассоциативность).
3. Имеется единственный нулевой вектор  $\mathbf{0}$ , такой что  $\mathbf{x} + \mathbf{0} = \mathbf{x}$  для любого  $\mathbf{x}$ .
4. Для любого  $\mathbf{x}$  имеется единственный противоположный вектор  $(-\mathbf{x})$ , такой что  $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$ .

В качестве скаляров может использоваться любое множество, элементы которого образуют *поле* (т. е. для них определены операции сложения и умножения). Обычно в качестве скаляров используются вещественные или комплексные числа. Для обозначения скаляров в приводимых ниже формулах применены греческие буквы.

Операция умножения вектора на скаляр обладает следующими свойствами:

1.  $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$  для любых скаляров  $\alpha, \beta$  и вектора  $\mathbf{x}$  (ассоциативность).
2.  $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$  для любого скаляра  $\alpha$  и векторов  $\mathbf{x}$  и  $\mathbf{y}$  (дистрибутивность векторная).
3.  $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$  для любых скаляров  $\alpha, \beta$  и вектора  $\mathbf{x}$  (дистрибутивность скалярная).

Для сигналов непрерывного времени эти две операции определяются естественным образом: сложение  $\mathbf{x} + \mathbf{y}$  сводится к суммированию соответствующих функций,  $x(t) + y(t)$ , а умножение на скаляр — к умножению функции, описывающей сигнал, на скалярную константу:  $\alpha x(t)$ .

## Нормированное линейное пространство

*Нормированное линейное пространство* — это линейное пространство, в котором дополнительно определен "размер" каждого элемента. Этот размер называется *нормой* вектора (Norm). Нормой, обозначаемой как  $\|\mathbf{x}\|$ , может служить любое отображение, сопоставляющее элементам пространства вещественные числа таким образом, что выполняются следующие свойства:

1.  $\|\mathbf{x}\| \geq 0$  для любого  $\mathbf{x}$ , причем  $\|\mathbf{x}\| = 0$  только если  $\mathbf{x} = \mathbf{0}$ .
2.  $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$  для любых  $\mathbf{x}$  и  $\mathbf{y}$  (неравенство треугольника).
3.  $\|\alpha\mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|$  для любых скаляра  $\alpha$  и вектора  $\mathbf{x}$ .

Сравнивая свойства нормы и метрики, легко увидеть взаимосвязь между ними. Так, норма разности сигналов,  $\|\mathbf{x} - \mathbf{y}\|$ , может служить метрикой. В свою очередь, если в линейном пространстве задана метрика, то нормой может служить расстояние между вектором и началом координат (нулевым вектором), т. е.  $d(\mathbf{x}, \mathbf{0})$  есть норма вектора  $\mathbf{x}$ . Таким образом, приведенная ранее  $p$ -метрика (1.38) порождает соответствующую  $p$ -норму:

$$\|\mathbf{x}\|_p = \sqrt[p]{\int_T |x(t)|^p dt}. \quad (1.39)$$

Для чаще всего используемых на практике вариантов  $p = 1$ ,  $p = 2$  и  $p = \infty$  общая формула (1.39) принимает следующие формы:

$$\begin{aligned} \|\mathbf{x}\|_1 &= \int_T |x(t)| dt, \\ \|\mathbf{x}\|_2 &= \sqrt{\int_T |x(t)|^2 dt}, \\ \|\mathbf{x}\|_\infty &= \max_{t \in T} |x(t)|. \end{aligned} \quad (1.40)$$

### ЗАМЕЧАНИЕ

Норма  $\|\mathbf{x}\|_2$ , определяемая формулой (1.40), как и породившая ее метрика  $d_2(x, y)$ , называется *евклидовой*.

## Пространство со скалярным произведением

*Пространство со скалярным произведением* — это линейное пространство, для упорядоченных пар элементов которого определена операция *скалярного произведения* (*scalar product*, *dot product* или *inner product*). Эта операция, обозначаемая как  $(\mathbf{x}, \mathbf{y})$ , отображает упорядоченную пару сигналов в число (в общем случае комплексное) так, что при этом выполняются следующие свойства:

1.  $(\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \mathbf{x})^*$  для любых векторов  $\mathbf{x}$  и  $\mathbf{y}$  (звездочка обозначает комплексное сопряжение).
2.  $(\alpha\mathbf{x} + \beta\mathbf{y}, \mathbf{z}) = \alpha(\mathbf{x}, \mathbf{z}) + \beta(\mathbf{y}, \mathbf{z})$  для любых скаляров  $\alpha, \beta$  и векторов  $\mathbf{x}, \mathbf{y}, \mathbf{z}$  (дистрибутивность).
3.  $(\mathbf{x}, \mathbf{x}) \geq 0$  для любого вектора  $\mathbf{x}$ , причем  $(\mathbf{x}, \mathbf{x}) = 0$  только если  $\mathbf{x} = \mathbf{0}$ .

Из свойства 1 следует, что  $(\mathbf{x}, \mathbf{x})$  — вещественное число, а из свойств 1 и 2 — что  $(\alpha\mathbf{x}, \mathbf{y}) = \alpha(\mathbf{x}, \mathbf{y})$ , но  $(\mathbf{x}, \alpha\mathbf{y}) = \alpha^*(\mathbf{x}, \mathbf{y})$ , т. е. скалярные множители из второго аргумента скалярного произведения выносятся с комплексным сопряжением, а из первого — нет.

На основе скалярного произведения можно ввести норму векторов следующим образом:

$$\|\mathbf{x}\| = \sqrt{(\mathbf{x}, \mathbf{x})}.$$

А норма, как было сказано ранее, в свою очередь порождает метрику:

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\| = \sqrt{(\mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y})}.$$

Для пространства аналоговых сигналов скалярное произведение вводится следующим образом:

$$(\mathbf{x}, \mathbf{y}) = \int_T x(t)y^*(t)dt.$$

Легко убедиться, что введенное таким образом скалярное произведение порождает евклидову норму (1.40) и метрику (1.38).

С понятием скалярного произведения связано очень важное неравенство, в различных вариантах формулировки связываемое с именами Коши, Буняковского и Шварца:

$$|(\mathbf{x}, \mathbf{y})|^2 \leq (\mathbf{x}, \mathbf{x})(\mathbf{y}, \mathbf{y}). \quad (1.41)$$

Векторы, скалярное произведение которых равно нулю, называются *ортогональными* (*orthogonal*):

$$(\mathbf{x}, \mathbf{y}) = 0.$$

## Дискретные представления сигналов

Под словом "представление" (*representation*) понимается способ описания сигнала с помощью функций или наборов чисел.

В случае дискретного представления сигнала записывается в виде *линейной комбинации*

$$\mathbf{x} = \sum_{i=1}^N \alpha_i \phi_i . \quad (1.42)$$

Здесь  $\{\phi_i\}$ ,  $i = 1, \dots, N$  — базисные векторы, которые должны быть *линейно независимыми*. Линейная независимость означает, что равенство

$$\sum_{i=1}^N \alpha_i \phi_i = \mathbf{0}$$

выполняется только при нулевых значениях всех скалярных коэффициентов  $\alpha_i$ .

Набор коэффициентов  $\{\alpha_i\}$ , входящих в выражение (1.42), называется *представлением* сигнала  $\mathbf{x}$  по отношению к базису  $\{\phi_i\}$ .

Понятие скалярного произведения позволяет выразить коэффициенты  $\alpha_i$  через исходный сигнал  $\mathbf{x}$ . Умножим обе части (1.42) справа на один из векторов базиса —  $\phi_j$ .

$$(\mathbf{x}, \phi_j) = \sum_{i=1}^N \alpha_i (\phi_i, \phi_j) .$$

Проделав это для всех  $j = 1 \dots N$ , получаем систему линейных уравнений относительно  $\alpha_i$ , записать которую в матричном виде можно следующим образом:

$$\underbrace{\begin{bmatrix} (\phi_1, \phi_1) & (\phi_1, \phi_2) & \dots & (\phi_1, \phi_N) \\ (\phi_2, \phi_1) & (\phi_2, \phi_2) & \dots & (\phi_2, \phi_N) \\ \vdots & \vdots & \ddots & \vdots \\ (\phi_N, \phi_1) & (\phi_N, \phi_2) & \dots & (\phi_N, \phi_N) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}}_{\alpha} = \underbrace{\begin{bmatrix} (\mathbf{x}, \phi_1) \\ (\mathbf{x}, \phi_2) \\ \vdots \\ (\mathbf{x}, \phi_N) \end{bmatrix}}_{\beta} . \quad (1.43)$$

Решение этой системы дает

$$\alpha = \Phi^{-1} \beta . \quad (1.44)$$

Вычисления сильно упрощаются, если базис  $\{\phi_i\}$  является *ортонормальным* (*orthonormal*), т. е. если для него выполняются соотношения

$$(\phi_i, \phi_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases}$$

Тогда  $\Phi$  в (1.43) становится единичной матрицей и мы получаем

$$\alpha_i = (\mathbf{x}, \phi_i) , \quad (1.45)$$

а представление сигнала в виде линейной комбинации принимает вид

$$\mathbf{x} = \sum_{i=1}^N (\mathbf{x}, \phi_i) \phi_i . \quad (1.46)$$

Равенства (1.42) и (1.46) являются точными только в том случае, когда сигнал  $\mathbf{x}$  принадлежит множеству всевозможных линейных комбинаций базисных векторов  $\{\phi_i\}$ . Если же это не так, линейная комбинация с коэффициентами  $\alpha_i$ , рассчитанными согласно (1.44) или (для ортонормального базиса) (1.45), дает сигнал  $\hat{\mathbf{x}}$ , который является наилучшим *приближением* сигнала  $\mathbf{x}$ , представимым в базисе  $\{\phi_i\}$ . Приближение является наилучшим в том смысле, что оно обеспечивает минимальную возможную норму ошибки представления сигнала:

$$\|\mathbf{x} - \hat{\mathbf{x}}\| = \left\| \mathbf{x} - \sum_{i=1}^N \alpha_i \phi_i \right\| \rightarrow \min.$$

Ошибка представления  $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$  ортогональна всем базисным векторам, а значит, и всем их линейным комбинациям:

$$\mathbf{e} = \mathbf{x} - \sum_{i=1}^N \alpha_i \phi_i,$$

$$(\mathbf{e}, \phi_i) = 0 \text{ для всех } i = 1, \dots, N.$$

Поэтому приближенное представление  $\hat{\mathbf{x}}$  называется *ортогональной проекцией* вектора  $\mathbf{x}$  на пространство линейных комбинаций базисных векторов  $\{\phi_i\}$ .

Благодаря указанной ортогональности между нормами исходного вектора  $\mathbf{x}$ , его приближения  $\hat{\mathbf{x}}$  и ошибки представления  $\mathbf{e}$  существует простая связь:

$$\|\mathbf{e}\|^2 = \|\mathbf{x}\|^2 - \|\hat{\mathbf{x}}\|^2. \quad (1.47)$$

В свою очередь, для ортонормального базиса  $\{\phi_i\}$  квадрат нормы линейной комбинации (1.46) равен сумме квадратов модулей коэффициентов разложения:

$$\|\hat{\mathbf{x}}\|^2 = \sum_{i=1}^N |\alpha_i|^2 = \sum_{i=1}^N |(\mathbf{x}, \phi_i)|^2. \quad (1.48)$$

Из (1.47) и (1.48) следует *неравенство Бесселя*:

$$\sum_{i=1}^N |(\mathbf{x}, \phi_i)|^2 \leq \|\mathbf{x}\|^2.$$

Размерность базиса  $N$  может быть конечной либо бесконечной. Если бесконечный ортонормальный базис нельзя расширить, т. е. в рассматриваемом пространстве не существует ненулевых векторов, которые были бы ортогональны всем уже имеющимся векторам базиса (а значит, и всем их линейным комбинациям), то он называется *полной ортонормальной системой* (*complete orthonormal system*).

Если  $N = \infty$  и  $\{\phi_i\}$  является полной ортонормальной системой, норма ошибки представления становится равна нулю (т. е. представление становится точным), а неравенство Бесселя превращается в равенство:

$$\sum_{i=1}^{\infty} |(\mathbf{x}, \phi_i)|^2 = \|\mathbf{x}\|^2.$$

**ВНИМАНИЕ!**

Не любая бесконечная система ортонормальных векторов будет являться полной. С примером системы такого рода мы встретимся в главе 3 при рассмотрении теоремы Котельникова.

Рассмотренный в начале этой главы комплексный ряд Фурье (см. формулы (1.10) и (1.12)) в данном обобщенном контексте является дискретным представлением сигнала с использованием следующей бесконечной системы базисных функций, рассматриваемых на произвольном интервале временной оси длиной  $T$ :

$$\varphi_k(t) = \exp\left(j \frac{2\pi k}{T} t\right).$$

Эта система является полной и ортогональной, но не нормированной (евклидова норма базисных функций в данном случае равна  $T$ ), поэтому формула расчета коэффициентов представления (1.12) отличается от общей формулы (1.45) наличием множителя  $1/T$ .

## Интегральные представления сигналов

Идею перехода от дискретного к интегральному описанию сигнала можно представить примерно так же, как ранее в этой главе был представлен переход от ряда Фурье к преобразованию Фурье: дискретно меняющийся индекс  $i$  превращается в параметр  $s$ , принимающий значения из некоторой непрерывной области; соответственно, счетная последовательность коэффициентов  $\{\alpha_i\}$  превращается в функцию параметра  $s - u(s)$ , а набор базисных функций  $\{\varphi_i\}$  — в функцию двух переменных  $\varphi(t, s)$ , называемую ядром (*kernel*) преобразования; наконец, сумма заменяется интегралом. В результате получается следующая формула:

$$x(t) = \int_S u(s)\varphi(t, s)ds. \quad (1.49)$$

Здесь  $u(s)$  — функция, представляющая исходный сигнал  $x(t)$  в области параметра  $s$ ;  $\varphi(t, s)$  — ядро преобразования;  $S$  — область значений параметра  $s$ .

Аналогичный вид имеет и формула, позволяющая получить представление  $u(s)$ , зная сигнал  $x(t)$ :

$$u(s) = \int_T x(t)\theta(s, t)dt. \quad (1.50)$$

Здесь  $T$  — область определения сигнала  $x(t)$ ,  $\theta(s, t)$  — взаимное (используются также термины "обратное" и "сопряженное") ядро преобразования.

Установим связь между ядрами преобразований  $\varphi(t, s)$  и  $\theta(s, t)$ , которая должна существовать, чтобы формулы (1.49) и (1.50) были взаимно обратными. Для этого подставим, например, в (1.49) выражение для  $u(s)$  в виде (1.50):

$$x(t) = \int_S \left[ \int_T x(t')\theta(s, t')dt' \right] \varphi(t, s)ds = \int_T x(t') \left[ \int_S \varphi(t, s)\theta(s, t')ds \right] dt'.$$

Из последней формулы следует, что интегрирование по  $t'$  из функции  $x(t')$  "вырезает" ее значение при  $t' = t$ . Это возможно только в том случае, если результат вычислений в квадратных скобках дает дельта-функцию  $\delta(t - t')$  (см. формулу (1.2)). Отсюда получаем условие связи прямого и обратного ядер преобразования:

$$\int_S \phi(t, s)\theta(s, t')ds = \delta(t - t'). \quad (1.51)$$

Аналогично, если подставить в (1.50) выражение для  $x(t)$  в виде (1.49), получается второе условие связи:

$$\int_T \theta(s, t)\phi(t, s')dt = \delta(s - s'). \quad (1.52)$$

В монографии [5] можно найти более подробное рассмотрение условий связи между ядрами прямого и обратного интегральных преобразований для случаев, когда ядра зависят от разности либо произведения аргументов  $t$  и  $s$ .

Рассмотренные ранее в этой главе преобразования в данном обобщенном контексте являются интегральными преобразованиями со следующими ядрами:

□ преобразование Фурье:

$$\phi(t, s) = \frac{1}{2\pi} e^{jst}, \quad \theta(s, t) = e^{-jst}. \quad (1.53)$$

□ преобразование Гильберта:

$$\phi(t, s) = \frac{1}{\pi(s - t)}, \quad \theta(t, s) = \frac{-1}{\pi(t - s)}.$$

Аналогом ортонормального базиса для интегральных преобразований являются *самосопряженные ядра*, подчиняющиеся, помимо (1.51) и (1.52), соотношению

$$\phi(t, s) = \theta^*(s, t).$$

Интегральное преобразование с таким ядром не меняет значения скалярного произведения сигналов, т. е. если  $u_1(s)$  и  $u_2(s)$  — представления сигналов  $x_1(t)$  и  $x_2(t)$  соответственно, то

$$(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{u}_1, \mathbf{u}_2),$$

или, в интегральной записи,

$$\int_T x_1(t)x_2^*(t)dt = \int_S u_1(s)u_2^*(s)ds. \quad (1.54)$$

Из выражения (1.53) видно, что преобразование Фурье в рассматриваемой в данной книге формулировке, строго говоря, не является самосопряженным, т. к. в одном из ядер присутствует дополнительный постоянный множитель (его наличие обусловлено использованием круговой частоты — см. сделанное ранее замечание, предваряющее формулы (1.16) и (1.17)). Однако такая асимметрия приводит лишь к необходимости учесть этот множитель в правой или левой части формулы (1.54). Мы

уже видели наличие этого множителя ранее, в формулах (1.31) и (1.32), которые представляют собой частные случаи формулы (1.54) для преобразования Фурье. Если перейти от круговой частоты к обычной, равенство (1.54) будет выполняться без дополнительного множителя.

Помимо рассмотренных частотных представлений сигнала с использованием рядов Фурье и преобразования Фурье, а также временного представления сигнала с помощью преобразования Гильберта, в последнее время в теории сигналов значительное внимание уделяется *частотно-временному анализу* (*time-frequency analysis*). Одним из инструментов такого анализа является *вейвлет-преобразование* (*wavelet transform*), которое, подобно преобразованию Фурье, существует как в дискретном, так и в интегральном виде. Рассмотрение вейвлет-преобразования в данном учебном пособии не представляется возможным; заинтересованный читатель может найти информацию в книгах [27, 41, 42], минимальные сведения содержатся также в [2].

## Случайные сигналы

В отличие от детерминированных сигналов, форму которых мы знаем точно, мгновенные значения случайных сигналов заранее не известны и могут быть предсказаны лишь с некоторой вероятностью, меньшей единицы. Характеристики таких сигналов являются *статистическими*, т. е. имеют вероятностный вид.

В радиотехнике существует два основных класса сигналов, неждающихся в вероятностном описании. Во-первых, это *шумы* — хаотически изменяющиеся во времени электромагнитные колебания, возникающие в разнообразных физических системах из-за беспорядочного движения носителей заряда. Во-вторых, случайными являются все сигналы, несущие *информацию*, поэтому для описания закономерностей, присущих осмысленным сообщениям, также прибегают к вероятностным моделям.

## Ансамбль реализаций

Математическая модель изменяющегося во времени случайного сигнала называется *случайным процессом*. По определению, случайный процесс  $X(t)$  — это функция особого вида, характеризующаяся тем, что значения, принимаемые ею в любой момент времени  $t$ , являются случайными величинами.

### ЗАМЕЧАНИЕ

В технической литературе термины "случайный сигнал" и "случайный процесс" часто используются как синонимы.

До регистрации (приема) случайный сигнал следует рассматривать именно как случайный процесс, представляющий собой совокупность (*ансамбль*) функций времени  $x_i(t)$ , подчиняющихся некоторой общей для них статистической закономерности. Одна из этих функций, ставшая полностью известной после приема сообщения, называется *реализацией* случайного процесса. Эта реализация является уже не случайной, а детерминированной функцией времени. На рис. 1.28 приведен пример нескольких реализаций случайного процесса.

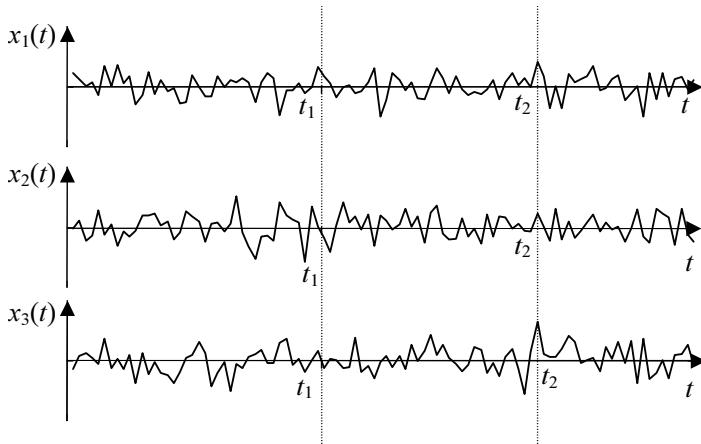


Рис. 1.28. Реализации случайного процесса

## Модели случайных процессов

Для анализа свойств и характеристик случайного процесса, а также различных его преобразований, необходимо задать *математическую модель* случайного процесса. Такая модель может представлять собой описание возможных реализаций случайного процесса в сочетании с указанием относительной частоты их появления. Приведем несколько примеров моделей случайных процессов, задаваемых таким образом.

### Гармонический сигнал со случайной начальной фазой

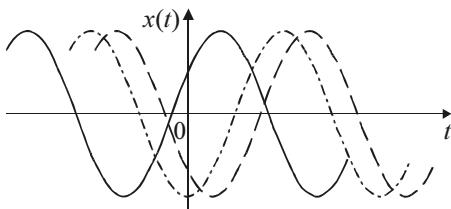
Во многих практических задачах используется модель случайного процесса, реализации которого представляют собой гармонические колебания с известными (детерминированными) амплитудой и частотой, но случайной начальной фазой. Таким образом, реализация рассматриваемого случайного процесса может быть записана как

$$x(t) = A \cos(\omega_0 t + \varphi),$$

где  $A$  — амплитуда (детерминированная),  $\omega_0$  — частота (детерминированная) и  $\varphi$  — случайная начальная фаза, которая в большинстве практически интересных случаев может считаться равномерно распределенной на интервале  $0..2\pi$ , т. е. имеющей следующую плотность вероятности:

$$p_\varphi(\varphi) = \begin{cases} \frac{1}{2\pi}, & 0 \leq \varphi < 2\pi, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Графики нескольких реализаций данного случайного процесса, представляющие собой синусоиды, смещенные друг относительно друга по временной оси, показаны на рис. 1.29.



**Рис. 1.29.** Реализации гармонического сигнала со случайной начальной фазой

Как видите, конкретный вид реализации *процесса* в данном случае определяется значением всего лишь одной случайной величины — начальной фазы.

#### ЗАМЕЧАНИЕ

Случайные процессы, конкретный вид реализаций которых определяется значениями конечного числа параметров (случайных величин), иногда называют *квазидетерминированными*.

### Случайный телеграфный сигнал

Таким сигналом (см., например, [5]) называют случайный процесс, реализации которого принимают значения +1 и -1, причем перепады уровня происходят в случайные моменты времени и число  $N$  перепадов уровня, происходящих за время  $\tau$ , является случайной величиной с дискретным распределением вероятности, описываемым законом Пуассона:

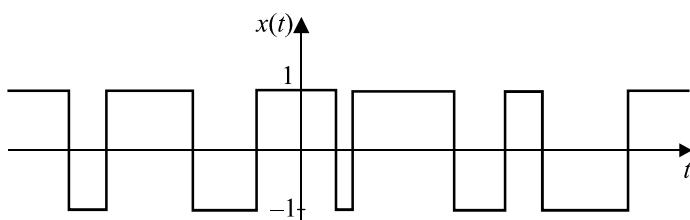
$$P(N, \tau) = \frac{(\lambda\tau)^N}{N!} e^{-\lambda\tau}. \quad (1.55)$$

Здесь  $\lambda$  — неотрицательный параметр, определяющий среднюю частоту возникновения перепадов уровня.

#### ЗАМЕЧАНИЕ

Прописной буквой  $P$  обозначается вероятность некоторого события, указываемого в скобках. В формуле (1.55)  $P(N, \tau)$  — это вероятность того, что за время  $\tau$  произойдет  $N$  перепадов уровня сигнала.

Скачки уровня происходят в случайные моменты времени  $t_k$ , поэтому аналитически записать формулу для отдельной реализации данного случайного процесса оказывается весьма затруднительно, а изобразить ее график можно лишь условно (рис. 1.30).



**Рис. 1.30.** Реализация случайного телеграфного сигнала

В данном случае конкретная реализация задается бесконечным множеством случайных величин — моментов перепадов уровня  $t_k$ , а характеристики случайного процесса определяются статистическими свойствами этих случайных величин.

Итак, полное описание случайного процесса дает его *ансамбль реализаций*. Однако для решения практических задач часто достаточно более простых характеристик, выражающихся в виде числовых параметров и детерминированных функций. Об этом речь пойдет далее.

## Вероятностные характеристики случайных процессов

Пусть  $X(t)$  — случайный процесс, заданный ансамблем реализаций  $\{x_1(t), x_2(t), \dots, x_k(t), \dots\}$ . Выбрав произвольный момент времени  $t_1$ , зафиксируем значения, принимаемые всеми реализациями:  $\{x_1(t_1), x_2(t_1), \dots, x_k(t_1), \dots\}$  (см. рис. 1.28). Совокупность этих значений образует *одномерное сечение* случайного процесса и представляет собой *случайную величину*  $X(t_1)$ . Напомним кратко основные характеристики случайных величин, отметив при этом, что для одномерных сечений случайных процессов они в общем случае зависят от выбранного момента времени  $t_1$ .

### Функциональные характеристики

*Функция распределения вероятности* (*cumulative distribution function*, CDF), обозначаемая как  $F(x, t_1)$ , равна вероятности того, что в момент времени  $t_1$  значение случайного процесса не превосходит  $x$ :

$$F(x, t_1) = P(X(t_1) \leq x).$$

$F(x, t_1)$  является неубывающей функцией, значения которой лежат в диапазоне  $0 \leq F(x, t_1) \leq 1$ . Для предельных значений  $x$  выполняются следующие соотношения:  $F(-\infty, t_1) = 0$  и  $F(\infty, t_1) = 1$ .

Вероятность попадания значения случайного процесса в интервал  $(a, b]$  равна разности значений функции распределения на концах этого интервала:

$$P(a < X(t_1) \leq b) = F(b, t_1) - F(a, t_1).$$

*Одномерная плотность вероятности* (*probability density function*, PDF) обозначается  $p(x, t_1)$  и представляет собой производную от функции распределения:

$$p(x, t_1) = \frac{dF(x, t_1)}{dx}, \text{ соответственно, } F(x, t_1) = \int_{-\infty}^x p(x, t_1) dx. \quad (1.56)$$

Произведение  $p(x, t_1) dx$  равно вероятности попадания значения случайного процесса  $X(t_1)$  в бесконечно малый интервал шириной  $dx$  в окрестности  $x$ :

$$p(x, t_1) dx = P\left(|X(t_1) - x| \leq \frac{dx}{2}\right),$$

откуда следует, что плотность вероятности является неотрицательной функцией:  $p(x, t_1) \geq 0$ . Чтобы рассчитать вероятность попадания значения  $X(t_1)$  в произвольный интервал  $(a, b]$ , необходимо вычислить следующий интеграл:

$$P(a < X(t_1) \leq b) = \int_a^b p(x, t_1) dx .$$

Наконец, поскольку случайная величина обязательно принимает *какое-нибудь* значение, должно выполняться условие нормировки:

$$\int_{-\infty}^{\infty} p(x, t_1) dx = P(-\infty < X(t_1) < \infty) = 1 . \quad (1.57)$$

## Числовые характеристики

Знание одномерной плотности вероятности  $p(x, t_1)$  позволяет произвести статистическое усреднение как самой величины  $X(t_1)$ , так и любой функции от нее. Под *статистическим усреднением* (*ensemble averaging*) подразумевается усреднение по множеству (по ансамблю реализаций) в каком-либо сечении процесса, т. е. в фиксированный момент времени.

Для практических приложений наибольшее значение имеют следующие параметры случайного процесса:

- *математическое ожидание* (*mean value*), которое служит теоретической оценкой среднего взвешенного значения случайного процесса в момент времени  $t$ :

$$m_x(t) = M\{X(t)\} = \int_{-\infty}^{\infty} xp(x, t) dx ; \quad (1.58)$$

### ЗАМЕЧАНИЕ

Во многих практических задачах приходится вычислять математическое ожидание некоторой функции  $f$  от случайной величины  $x$ , имеющей плотность вероятности  $p_x(x)$ . Такое вычисление выполняется по следующей несложной формуле:

$$M\{f(x)\} = \int_{-\infty}^{\infty} f(x) p_x(x) dx . \quad (1.59)$$

Формула для математического ожидания (1.58) является частным случаем (1.59) при  $f(x) = x$ .

- *дисперсия* (*variance*), характеризующая среднюю мощность отклонений случайного процесса от его среднего значения  $m_x(t)$ , называемых *флуктуациями* (*fluctuation*):

$$\sigma_x^2(t) = M\{[X(t) - m_x(t)]^2\} = M\{X^2(t)\} - m_x^2(t) = \int_{-\infty}^{\infty} x^2 p(x, t) dx - m_x^2(t) ; \quad (1.60)$$

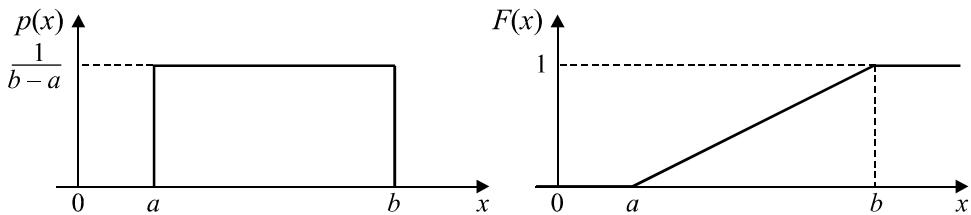
- *среднее квадратическое отклонение (standard deviation)*, представляющее собой квадратный корень из дисперсии и служащее амплитудной мерой разброса значений случайного процесса в момент времени  $t$  относительно математического ожидания:

$$\sigma_x(t) = \sqrt{\sigma_x^2(t)} = \sqrt{M\{[X(t) - m_x(t)]^2\}} = \sqrt{M\{X^2(t)\} - m_x^2(t)}. \quad (1.61)$$

## Равномерное распределение

Одним из часто используемых на практике законов распределения случайных величин является равномерное распределение (*uniform distribution*). При этом плотность вероятности является константой на некотором интервале  $[a, b]$  (рис. 1.31, слева). Величина этой константы, согласно условию нормировки (1.57), должна быть равна  $1/(b - a)$ :

$$p(x) = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b, \\ 0, & x < a, \quad x > b. \end{cases}$$



**Рис. 1.31.** Плотность вероятности (слева) и функция распределения (справа) случайной величины с равномерным распределением

Функция распределения, согласно (1.56), на интервале  $[a, b]$  линейно возрастает от 0 до 1 (см. рис. 1.31, справа):

$$F(x) = \begin{cases} 0, & x < a, \\ \frac{x-a}{b-a}, & a \leq x \leq b, \\ 1, & x > b. \end{cases}$$

Математическое ожидание, как и предсказывает интуиция, равно середине интервала возможных значений случайной величины:

$$m_x = \int_a^b x \frac{1}{b-a} dx = \frac{a+b}{2}.$$

### ЗАМЕЧАНИЕ

Если функция плотности вероятности имеет симметричный вид, то значение математического ожидания совпадает с положением оси симметрии.

Для расчета дисперсии необходимо сначала определить средний квадрат:

$$M\{X^2\} = \int_a^b x^2 \frac{1}{b-a} dx = \frac{b^3 - a^3}{3(b-a)} = \frac{a^2 + ab + b^2}{3}.$$

Теперь можно рассчитать дисперсию согласно (1.60):

$$\sigma_x^2 = \frac{a^2 + ab + b^2}{3} - \left(\frac{a+b}{2}\right)^2 = \frac{(b-a)^2}{12}.$$

Итак, дисперсия равна одной двенадцатой квадрата ширины интервала. Среднее квадратическое отклонение, естественно, оказывается пропорциональным этой ширине:

$$\sigma_x = \sqrt{\sigma_x^2} = \frac{b-a}{2\sqrt{3}}.$$

## Нормальное распределение

Нормальный (гауссов) закон распределения случайных величин (*normal distribution, Gaussian distribution*) очень удобен для анализа и часто встречается на практике, особенно он характерен для помех канала связи. Одномерная плотность вероятности нормальной случайной величины определяется выражением

$$p(x) = \frac{1}{\sigma_x \sqrt{2\pi}} \exp\left(-\frac{(x-m_x)^2}{2\sigma_x^2}\right), \quad (1.62)$$

где  $m_x$  и  $\sigma_x^2$  — соответственно математическое ожидание и дисперсия.

На рис. 1.32 приведен график плотности вероятности нормальной случайной величины, построенный согласно (1.62) при  $m_x = 0$  и  $\sigma_x = 1$ .

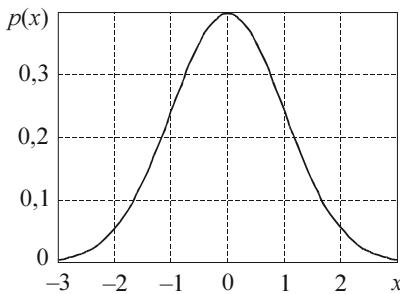


Рис. 1.32. Плотность вероятности случайной величины с нормальным распределением

Функция распределения для закона Гаусса, к сожалению, не выражается через элементарные функции. В отечественной литературе принято выражать ее через так называемый *интеграл вероятности*:

$$\Phi(x) = \int_{-\infty}^x \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x')^2}{2}\right) dx'. \quad (1.63)$$

Для нормального закона с математическим ожиданием  $m_x$  и дисперсией  $\sigma_x^2$  функция распределения выражается через интеграл вероятности следующим образом:

$$F(x) = \Phi\left(\frac{x - m_x}{\sigma_x}\right).$$

В зарубежной литературе большее распространение получила так называемая *функция ошибок (error function)* erf:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt. \quad (1.64)$$

Связь между функцией ошибок (1.64) и интегралом вероятности (1.63) выражается с помощью линейных преобразований аргументов функций и их результатов:

$$\text{erf}(x) = 2\Phi(x\sqrt{2}) - 1, \quad \Phi(x) = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{x}{\sqrt{2}}\right).$$

Функция распределения для нормального закона с математическим ожиданием  $m_x$  и дисперсией  $\sigma_x^2$  выражается через функцию ошибок (1.64) следующим образом:

$$F(x) = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{x - m_x}{\sigma_x \sqrt{2}}\right).$$

### ЗАМЕЧАНИЕ

В MATLAB имеется функция `erf`, реализующая формулу (1.64). Есть также функция `erfc`, возвращающая значение  $1 - \text{erf}(x)$ , и обратная функция `erfinv`.

Непосредственно вычислять функциональные характеристики нормального закона распределения можно с помощью функций `normpdf` (плотность вероятности), `normcdf` (функция распределения) и `norminv` (обратная функция распределения), входящих в пакет Statistics:

```
y = normpdf(x, m, sigma);
y = normcdf(x, m, sigma);
x = norminv(y, m, sigma);
```

Здесь  $m$  — математическое ожидание,  $\sigma$  — среднее квадратическое отклонение.

Кроме того, в пакете Communications Toolbox имеется функция `qfunc`, вычисляющая дополнение интеграла вероятности до единицы,  $1 - F(x)$ .

Широкое распространение нормального закона распределения в природе объясняется тем, что при суммировании достаточно большого числа равнозначных статистически независимых случайных величин, имеющих произвольные плотности распределения вероятности, плотность распределения суммы стремится к нормальному. Это положение носит название *центральной предельной теоремы*.

Весьма полезным для математического анализа свойством нормального распределения является то, что из *некоррелированности* гауссовых случайных величин сле-

дует их *статистическая независимость* (о разнице между этими понятиями см. далее в разд. "Корреляционные функции случайных процессов").

Одномерная плотность вероятности и связанные с ней числовые характеристики позволяют получить важную информацию о свойствах случайного процесса. Однако для решения многих задач таких сведений недостаточно, т. к. они дают вероятностное представление о случайном процессе  $X(t)$  только в отдельные моменты и ничего не говорят о том, как он изменяется во времени. Для описания его динамических свойств необходимо использовать корреляционную функцию или привлечь для этого спектральные характеристики. Упомянутые способы описания случайных процессов рассмотрены далее.

## Корреляционные функции случайных процессов

Как отмечалось в разд. "Вероятностные характеристики случайных процессов", одномерной плотности вероятности недостаточно для описания поведения случайного процесса во времени. Гораздо больше сведений можно получить, располагая двумя сечениями случайного процесса в произвольные моменты времени  $t_1$  и  $t_2$  (см. рис. 1.28). Совокупность этих двух сечений образует *двумерную* случайную величину  $\{X(t_1), X(t_2)\}$ , которая описывается двумерной плотностью вероятности  $p(x_1, x_2, t_1, t_2)$ . Произведение  $p(x_1, x_2, t_1, t_2) dx_1 dx_2$  представляет собой вероятность того, что реализация случайного процесса  $X(t)$  в момент времени  $t_1$  попадает в бесконечно малый интервал шириной  $dx_1$  в окрестности  $x_1$ , а в момент времени  $t_2$  — в бесконечно малый интервал шириной  $dx_2$  в окрестности  $x_2$ :

$$p(x_1, x_2, t_1, t_2) dx_1 dx_2 = P \left\{ \left| X(t_1) - x_1 \right| \leq \frac{dx_1}{2}, \quad \left| X(t_2) - x_2 \right| \leq \frac{dx_2}{2} \right\}.$$

Естественным обобщением является  $n$ -мерное сечение случайного процесса, приходящее к  $n$ -мерной плотности вероятности  $p(x_1, x_2, \dots, x_n, t_1, t_2, \dots, t_n)$ . При  $n \rightarrow \infty$  такая функция является исчерпывающей вероятностной характеристикой случайного процесса.

Описание свойств случайных процессов с помощью многомерных плотностей вероятности высокой размерности может быть весьма подробным, однако на этом пути часто встречаются серьезные математические трудности. К счастью, многие задачи, связанные с описанием случайных сигналов, удается решить на основе двумерной плотности вероятности.

В частности, задание двумерной плотности вероятности  $p(x_1, x_2, t_1, t_2)$  позволяет определить важную характеристику случайного процесса — его *корреляционную функцию* (*correlation function*)

$$R_x(t_1, t_2) = M \{ x(t_1) x(t_2) \}.$$

Согласно этому определению, корреляционная функция случайного процесса  $X(t)$  представляет собой статистически усредненное произведение значений случайной функции  $X(t)$  в моменты времени  $t_1$  и  $t_2$ .

Для каждой реализации случайного процесса произведение  $x(t_1)x(t_2)$  является некоторым числом. Совокупность реализаций образует множество случайных чисел, распределение которых характеризуется двумерной плотностью вероятности  $p(x_1, x_2, t_1, t_2)$ . Если эта плотность известна, операция усреднения по множеству осуществляется по формуле

$$R_x(t_1, t_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 p(x_1, x_2, t_1, t_2) dx_1 dx_2.$$

Часто при анализе случайных процессов основной интерес представляет их флюктуационная составляющая. В таких случаях применяется *ковариационная функция* (*covariance function*), представляющая собой статистически усредненное произведение значений *центрированной* случайной функции  $X(t) - m_x(t)$  в моменты времени  $t_1$  и  $t_2$ :

$$\begin{aligned} K_x(t_1, t_2) &= M\{(x(t_1) - m_x(t_1))(x(t_2) - m_x(t_2))\} = \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x(t_1) - m_x(t_1))(x(t_2) - m_x(t_2)) p(x_1, x_2, t_1, t_2) dx_1 dx_2 = \\ &= R_x(t_1, t_2) - m_x(t_1)m_x(t_2). \end{aligned}$$

Ковариационная функция характеризует степень статистической связи тех значений случайного процесса, которые наблюдаются при  $t = t_1$  и  $t = t_2$ . При  $t_1 = t_2 = t$  последнее выражение соответствует определению дисперсии случайного процесса  $X(t)$  (см. формулу (1.60)). Следовательно, при совмещении сечений ковариационная функция равна дисперсии:

$$K_x(t, t) = R_x(t, t) - m_x^2(t) = \sigma_x^2(t). \quad (1.65)$$

### ЗАМЕЧАНИЕ

Так сложилось, что в ряде источников используется обратная терминология —  $K_x(t_1, t_2)$  называется *корреляционной*, а  $R_x(t_1, t_2)$  *ковариационной* функцией. Во избежание недоразумений об этом следует помнить при работе с литературой. Впрочем, при анализе центрированных (имеющих нулевое математическое ожидание) случайных процессов корреляционная и ковариационная функции совпадают.

В качестве примера рассчитаем корреляционную функцию гармонического сигнала со случайной равномерно распределенной начальной фазой (см. разд. "Модели случайных процессов" ранее в этой главе).

Можно легко убедиться, что данный случайный процесс является *центризованным*, т. е. его математическое ожидание не зависит от времени и равно нулю:

$$m_x(x) = \int_{-\infty}^{\infty} x(t) p_\phi(\phi) d\phi = \int_0^{2\pi} A \cos(\omega_0 t + \phi) \frac{1}{2\pi} d\phi = 0. \quad (1.66)$$

Поэтому ковариационная и корреляционная функции данного процесса совпадают и могут быть найдены следующим образом (поскольку реализации данного случайного процесса представляют собой функции, зависящие от одной случайной величины, для усреднения произведения нет необходимости прибегать к двумерной плотности вероятности — достаточно воспользоваться формулой (1.59), позволяющей произвести усреднение произвольной функции от случайной величины):

$$\begin{aligned} R_x(t_1, t_2) &= K_x(t_1, t_2) = \int_{-\infty}^{\infty} x(t_1)x(t_2)p_{\phi}(\phi)d\phi = \\ &= \int_0^{2\pi} A \cos(\omega_0 t_1 + \phi) A \cos(\omega_0 t_2 + \phi) \frac{1}{2\pi} d\phi = \\ &= \frac{A^2}{2\pi} \left( \int_0^{2\pi} \frac{1}{2} \cos(\omega_0(t_1 + t_2) + 2\phi) d\phi + \int_0^{2\pi} \frac{1}{2} \cos(\omega_0(t_1 - t_2)) d\phi \right). \end{aligned}$$

В первом слагаемом интегрирование производится по двум периодам функции  $\cos$ , поэтому данный интеграл равен нулю. Во втором слагаемом подынтегральная функция не зависит от переменной интегрирования  $\phi$ , так что результат интегрирования равен произведению подынтегрального выражения и длины промежутка интегрирования, равной  $2\pi$ . Окончательно получаем

$$R_x(t_1, t_2) = K_x(t_1, t_2) = \frac{A^2}{2} \cos(\omega_0(t_1 - t_2)). \quad (1.67)$$

Как видите, корреляционная функция данного случайного процесса гармонически зависит от расстояния между анализируемыми моментами времени. При совпадении моментов времени  $t_1$  и  $t_2$  мы получаем значение дисперсии случайного процесса:

$$\sigma_x^2(t) = R_x(t, t) = \frac{A^2}{2}. \quad (1.68)$$

## Некоррелированность и статистическая независимость

Если совместно рассматривать две случайные величины  $X_1$  и  $X_2$ , между ними может существовать либо не существовать *статистическая связь*. Отсутствие такой связи означает, что плотность вероятности одной случайной величины не зависит от того, какое значение принимает другая случайная величина. Двумерная плотность вероятности при этом представляет собой произведение одномерных плотностей:

$$p(x_1, x_2) = p_1(x_1)p_2(x_2).$$

Это условие называется *условием статистической независимости*.

При наличии статистической связи между случайными величинами статистические свойства каждой из них зависят от значения, принимаемого другой случайной ве-

личиной. Эта связь может быть сильной или слабой, линейной или нелинейной. Мерой линейной статистической связи между случайными величинами является коэффициент корреляции:

$$r_{12} = \frac{M\{X_1 X_2\} - M\{X_1\} M\{X_2\}}{\sqrt{\sigma_{X_1}^2 \sigma_{X_2}^2}}. \quad (1.69)$$

Можно показать, что  $|r_{12}| \leq 1$ . Предельные значения  $\pm 1$  достигаются, если реализации случайных величин жестко связаны линейным соотношением  $x_2 = ax_1 + b$ , где  $a$  и  $b$  — некоторые константы. Знак коэффициента корреляции при этом совпадает со знаком множителя  $a$ .

Равенство коэффициента корреляции нулю свидетельствует об отсутствии линейной статистической связи между случайными величинами (при этом говорят об их некоррелированности). Как видно из (1.69), при этом математическое ожидание произведения случайных величин равно произведению их математических ожиданий:

$$M\{X_1 X_2\} = M\{X_1\} M\{X_2\}.$$

Легко показать, что из статистической независимости следует некоррелированность случайных величин. Обратное утверждение в общем случае неверно — некоррелированные случайные величины могут быть зависимыми.

#### **ЗАМЕЧАНИЕ**

Классическим примером этого является пара случайных величин  $x_1 = \cos \varphi$  и  $x_2 = \sin \varphi$ , где  $\varphi$  — случайная величина, равномерно распределенная на интервале  $0\dots 2\pi$ . Очевидно, что  $x_1$  и  $x_2$  зависят друг от друга; однако их коэффициент корреляции оказывается равным нулю.

## **Стационарные и эргодические случайные процессы**

В общем случае, как уже говорилось, вероятностные и корреляционные характеристики случайных процессов зависят от одного или нескольких моментов времени, в которые эти характеристики определяются. Однако существует класс случайных процессов, у которых зависимость характеристик от времени отсутствует. Кроме того, для некоторых случайных процессов не обязательно производить усреднение по ансамблю реализаций — можно ограничиться рассмотрением одной реализации и ее усреднением во времени.

Такие случайные процессы и рассмотрены в данном разделе.

## **Стационарные случайные процессы**

Так принято называть случайные процессы, статистические характеристики которых одинаковы во всех временных сечениях.

Говорят, что случайный процесс *строго стационарен* (или *стационарен в узком смысле*), если его многомерная плотность вероятности  $p(x_1, x_2, \dots, x_n, t_1, t_2, \dots, t_n)$

произвольной размерности  $n$  не изменяется при одновременном сдвиге всех временных сечений  $t_1, t_2, \dots, t_n$  вдоль оси времени на одинаковую величину  $\tau$ :

$$p(x_1, x_2, \dots, x_n, t_1, t_2, \dots, t_n) = p(x_1, x_2, \dots, x_n, t_1 + \tau, t_2 + \tau, \dots, t_n + \tau) \text{ при любом } \tau.$$

Если же ограничить требования тем, чтобы от временного сдвига не зависели лишь одномерная и двумерная плотности вероятности, то такой случайный процесс будет *стационарен в широком смысле*. Понятно, что из стационарности в узком смысле следует стационарность в широком смысле, но не наоборот.

Для стационарного случайного процесса математическое ожидание и дисперсия не зависят от времени, а корреляционная функция зависит не от самих моментов времени, а только от интервала между ними  $\tau = t_2 - t_1$ :

$$R_x(t_1, t_2) = R_x(t_2 - t_1) = R_x(\tau).$$

По этой причине при записи статистических параметров стационарного случайного процесса можно опускать обозначения фиксированных моментов времени:  $m_x, D_x, K_x(\tau), R_x(\tau)$ .

Легко убедиться, что корреляционная функция стационарного случайного процесса является четной:

$$R_x(-\tau) = R_x(\tau).$$

Кроме того, абсолютные значения этой функции при любых  $\tau$  не превышают ее значения при  $\tau = 0$  (напомним, что это значение равно дисперсии случайного процесса):

$$|R_x(\tau)| \leq R_x(0) = \sigma_x^2.$$

Часто удобно использовать *коэффициент корреляции* (его также называют *нормированной корреляционной функцией*)

$$r_x(\tau) = \frac{R_x(\tau)}{R_x(0)} = \frac{R_x(\tau)}{\sigma_x^2}.$$

Для коэффициента корреляции выполняются соотношения  $r_x(0) = 1, |r_x(\tau)| \leq 1$  и  $r_x(-\tau) = r_x(\tau)$ .

Функции  $R_x(\tau)$  и  $r_x(\tau)$  характеризуют связь (корреляцию) между значениями  $X(t)$ , разделенными промежутком  $\tau$ . Чем медленнее убывают эти функции с ростом абсолютного значения  $\tau$ , тем больше промежуток, в течение которого наблюдается статистическая связь между мгновенными значениями случайного процесса, и тем медленнее, плавнее изменяются во времени его реализации.

Легко заметить, что гармонический случайный процесс со случайной начальной фазой (см. разд. "Модели случайных процессов" и вычисление характеристик этого процесса в разд. "Корреляционные функции случайных процессов") является стационарным в широком смысле. Действительно, связанные с одномерной плотностью вероятности математическое ожидание (1.66) и дисперсия (1.68) не зависят

от времени, а корреляционная функция (1.67), определяемая двумерной плотностью вероятности, зависит лишь от интервала между рассматриваемыми моментами времени:

$$R_x(\tau) = K_x(\tau) = \frac{A^2}{2} \cos(\omega_0 \tau). \quad (1.70)$$

Коэффициент корреляции такого случайного процесса равен

$$r_x(\tau) = \frac{R_x(\tau)}{\sigma_x^2} = \cos(\omega_0 \tau).$$

### ЗАМЕЧАНИЕ

Здесь следует отметить, что стационарным будет *любой* случайный процесс, реализации которого являются периодическими функциями, идентичными по форме и различающимися лишь "начальной фазой", т. е. положением начала отсчета времени в пределах периода. При этом принципиальной является *равномерность* распределения начальной фазы в пределах периода. Действительно, пусть у гармонического процесса начальная фаза равномерно распределена в пределах *половины* окружности — на интервале  $0 \dots \pi$ . Математическое ожидание процесса в этом случае будет равно

$$m_x(x) = \int_{-\infty}^{\infty} x(t) p_\phi(\phi) d\phi = \int_0^\pi A \cos(\omega_0 t + \phi) \frac{1}{\pi} d\phi = -\frac{2A \sin(\omega_0 t)}{\pi}.$$

Результат вычислений показывает, что математическое ожидание процесса зависит от времени, следовательно, он не является стационарным.

## Эргодические случайные процессы

Дальнейшее упрощение анализа случайных процессов достигается при использовании условия *эргодичности* процесса. Стационарный случайный процесс называется эргодическим (*ergodic*), если при определении его статистических характеристик усреднение по множеству (ансамблю) реализаций эквивалентно усреднению по времени (*time averaging*) одной, теоретически бесконечно длинной, реализации.

Обозначив усреднение по времени угловыми скобками, можно записать следующие выражения, позволяющие вычислить важнейшие статистические характеристики эргодического случайного процесса по его единственной реализации  $x(t)$  (еще раз обращаем внимание на то, что эргодический случайный процесс обязательно является и стационарным, но не наоборот):

$$m_x = \langle x(t) \rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t) dt,$$

$$\sigma_x^2 = \left\langle [x(t) - m_x]^2 \right\rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} x^2(t) dt - m_x^2,$$

$$R_x(\tau) = \langle x(t)x(t - \tau) \rangle = \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t)x(t - \tau) dt.$$

Математическое ожидание эргодического случайного процесса равно *постоянной составляющей* любой его реализации, а дисперсия имеет наглядный физический смысл *мощности флуктуационной составляющей*.

Условия эргодичности для разных статистических характеристик являются разными. Например, достаточным условием *эргодичности по математическому ожиданию* является стремление к нулю корреляционной функции случайного процесса с ростом временного сдвига  $\tau$ :

$$\lim_{\tau \rightarrow \infty} R_x(\tau) = 0. \quad (1.71)$$

При экспериментальном исследовании случайных процессов доступно, как правило, наблюдение одной реализации сигнала, а не всего ансамбля. Если изучаемый процесс является эргодическим, то его реализация достаточной длины является "типичным" представителем статистического ансамбля. Согласно приведенным выше формулам по этой единственной реализации можно определить математическое ожидание, дисперсию и корреляционную функцию эргодического случайного процесса. На практике интегрирование выполняется, естественно, не в бесконечных пределах, а на конечном интервале, длина которого должна быть тем больше, чем выше требования к точности результатов измерения.

В качестве примера проверим эргодичность гармонического процесса со случайной начальной фазой (стационарность такого процесса была проверена ранее). Его корреляционная функция (1.70) с ростом  $\tau$  не стремится к нулю, так что условие (1.71) не выполняется. Однако это лишь *достаточное*, но не необходимое условие, поэтому его невыполнение еще не означает неэргодичности процесса. Проверим эргодичность согласно определению, вычислив усредненные по времени параметры:

$$\begin{aligned} \langle x(t) \rangle &= \frac{\omega_0}{2\pi} \int_{-\pi/\omega_0}^{\pi/\omega_0} A \cos(\omega_0 t + \varphi) dt = 0 = m_x, \\ \left\langle [x(t) - m_x]^2 \right\rangle &= \frac{\omega_0}{2\pi} \int_{-\pi/\omega_0}^{\pi/\omega_0} A^2 \cos^2(\omega_0 t + \varphi) dt = \frac{A^2}{2} = \sigma_x^2, \\ \left\langle x(t)x(t-\tau) \right\rangle &= \frac{\omega_0}{2\pi} \int_{-\pi/\omega_0}^{\pi/\omega_0} A^2 \cos(\omega_0 t + \varphi) \cos(\omega_0(t-\tau) + \varphi) dt = \\ &= \frac{A^2}{2} \cos(\omega_0 \tau) = R_x(\tau). \end{aligned}$$

### ЗАМЕЧАНИЕ

Тот факт, что реализации рассматриваемого процесса являются периодическими функциями, позволяет упростить вычисления, заменив усреднение по бесконечному (в пределе) промежутку времени усреднением по одному периоду, равному в данном случае  $2\pi/\omega_0$ .

Итак, параметры, вычисленные усреднением по времени, совпали с параметрами, полученными ранее путем статистического усреднения (формулы (1.66)–(1.68)). Следовательно, гармонический случайный процесс со случайной начальной фазой является эргодическим.

### **ЗАМЕЧАНИЕ**

Здесь также следует отметить, что *любой* случайный процесс, реализации которого являются периодическими функциями, идентичными по форме и различающимися лишь равномерно распределенной в пределах периода "начальной фазой", будет не только стационарным, но и *эргодическим*.

## **Спектральные характеристики случайных процессов**

Каждая отдельно взятая реализация случайного процесса представляет собой детерминированную функцию, и к ней можно применить преобразование Фурье. При этом различные реализации будут, естественно, иметь различные спектры. Нас же интересуют *статистически усредненные* характеристики случайных процессов. Попытаемся найти среднее значение спектральной плотности случайного процесса (горизонтальной чертой здесь и далее обозначается операция статистического усреднения по ансамблю реализаций):

$$\overline{S_x(\omega)} = \overline{\int_{-\infty}^{\infty} x(t)e^{-j\omega t} dt} = \int_{-\infty}^{\infty} \overline{x(t)}e^{-j\omega t} dt = \int_{-\infty}^{\infty} m_x(t)e^{-j\omega t} dt.$$

Как видите, усредненная спектральная плотность случайного процесса представляет собой спектр его *детерминированной составляющей* (математического ожидания). Для центрированных процессов  $m_x(t) = 0$  и  $\overline{S_x(\omega)} = 0$ . Таким образом, усредненное значение спектральной плотности не несет никакой информации о флюктуационной, т. е. собственно случайной, составляющей случайного процесса. Это говорит о том, что фазы спектральных составляющих в различных реализациях процесса случайны и независимы.

Можно, однако, рассмотреть спектральную плотность *мощности* случайного процесса, поскольку мощность не зависит от соотношения фаз спектральных составляющих.

Рассмотрим центрированный случайный процесс и выделим из его ансамбля какую-либо реализацию  $x(t)$ , ограничив ее длительность конечным интервалом времени  $[-T/2; T/2]$ . Применив затем к этой реализации прямое преобразование Фурье, найдем ее спектральную функцию  $\dot{X}_T(\omega)$ . Энергию  $E_T$  рассматриваемого отрезка реализации, согласно равенству Парсеваля (1.32), можно вычислить как

$$E_T = \int_{-T/2}^{T/2} x^2(t) dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\dot{X}_T(\omega)|^2 d\omega.$$

Разделив эту энергию на  $T$ , получим среднюю мощность  $P_T$  реализации на данном временном интервале:

$$P_T = \frac{E_T}{T} = \left\langle x^2(t) \right\rangle \Big|_{|t| \leq T/2} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{|\dot{X}_T(\omega)|^2}{T} d\omega.$$

При увеличении длительности промежутка времени  $T$  энергия отрезка реализации неограниченно возрастает, а средняя мощность стремится к некоторому пределу. Совершив предельный переход  $T \rightarrow \infty$  и произведя усреднение по ансамблю реализаций, получим

$$\overline{\left\langle x^2(t) \right\rangle} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lim_{T \rightarrow \infty} \frac{|\dot{X}_T(\omega)|^2}{T} d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} W(\omega) d\omega, \quad (1.72)$$

где функция

$$W(\omega) = \lim_{T \rightarrow \infty} \frac{|\dot{X}_T(\omega)|^2}{T} \quad (1.73)$$

представляет собой спектральную плотность средней мощности случайного процесса  $X(t)$ .

### ЗАМЕЧАНИЕ 1

Часто говорят "спектральная плотность мощности" или "спектр мощности"; соответствующий английский термин — *power spectral density, PSD*.

### ЗАМЕЧАНИЕ 2

Может показаться, что в случае эргодического процесса усреднение по ансамблю реализаций в формуле (1.73) не является обязательным, потому что один и тот же результат будет получен при рассмотрении любой отдельно взятой реализации. Однако при ближайшем рассмотрении оказывается, что без статистического усреднения предел в правой части (1.73) просто не существует.

После статистического усреднения левая часть (1.72) становится равной дисперсии рассматриваемого случайного процесса. Таким образом,

$$\sigma_x^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} W(\omega) d\omega. \quad (1.74)$$

$W(\omega)$  — вещественная функция, она не содержит информации о фазах спектральных составляющих и не позволяет восстановить отдельные реализации случайного процесса. Кроме того, из определения спектральной плотности (1.73) очевидно, что  $W(\omega)$  является неотрицательной и четной (для вещественных случайных процессов) функцией частоты.

### ЗАМЕЧАНИЕ

Здесь не показаны примеры расчета спектра случайного процесса согласно приведенному определению, поскольку такой расчет редко необходим на практике. Как

правило, вычисление спектра случайного процесса производится на основе его корреляционной функции с помощью теоремы Винера — Хинчина, речь о которой пойдет в следующем разделе.

## Теорема Винера — Хинчина

Как распределение спектральной плотности мощности, так и вид корреляционной функции связаны со скоростью изменения случайного процесса во времени. Найдем связь между этими двумя характеристиками.

Как известно, корреляционная функция детерминированного сигнала связана преобразованием Фурье с его энергетическим спектром. Применим это свойство к отрезку реализации случайного процесса длительностью  $T$ :

$$\int_{-T/2}^{T/2} x(t)x(t-\tau)dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |\dot{X}_T(\omega)|^2 e^{j\omega\tau} d\omega.$$

Разделим обе части этого равенства на  $T$ , устремим  $T$  к бесконечности и произведем статистическое усреднение:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \overline{x(t)x(t-\tau)} dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \lim_{T \rightarrow \infty} \frac{|\dot{X}_T(\omega)|^2}{T} e^{j\omega\tau} d\omega. \quad (1.75)$$

В левой части полученного равенства под интегралом стоит константа, равная корреляционной функции процесса,  $R(\tau)$ . В правой части под интегралом содержится выражение (1.73) для спектральной плотности мощности случайного процесса. С учетом этого

$$R(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} W(\omega) e^{j\omega\tau} d\omega. \quad (1.76)$$

Таким образом, корреляционная функция случайного процесса и его спектральная плотность мощности связаны друг с другом преобразованием Фурье. Это соотношение носит название *теоремы Винера — Хинчина*.

Для вещественных случайных процессов  $R(\tau)$  и  $W(\omega)$  являются четными вещественными функциями, поэтому можно отказаться от комплексной формы записи преобразования Фурье и перейти к полу бесконечным пределам интегрирования:

$$R(\tau) = \frac{1}{\pi} \int_0^{\infty} W(\omega) \cos(\omega\tau) d\omega, \quad W(\omega) = 2 \int_0^{\infty} R(\tau) \cos(\omega\tau) d\tau.$$

Очень часто используемая модель случайного процесса оказывается такова, что воспользоваться непосредственно определением (1.73) для расчета спектральной плотности мощности не представляется возможным. Если при этом удается вычислить корреляционную функцию, получить спектральную информацию позволяет теорема Винера — Хинчина.

В качестве примера рассмотрим случайный телеграфный сигнал (см. разд. "Модели случайных процессов"). Поскольку скачки уровня происходят в случайные моменты

времени, для данного случайного процесса затруднительно даже изобразить график отдельной реализации, не говоря уже о расчете спектра ее ограниченного во времени фрагмента.

Однако рассчитать корреляционную функцию для данного процесса оказывается совсем несложно. Действительно, произведение значений случайного процесса, разнесенных во времени на  $\tau$ , может быть равно +1 (если эти значения имеют одинаковый знак) или -1 (если знаки противоположны). Но совпадение знаков означает, что за интервал  $\tau$  произошло *четное* количество перепадов уровня, а несовпадение знаков соответствует *нечетному* количеству перепадов. Итак, чтобы найти вероятности для двух возможных значений произведения  $x(t)x(t-\tau)$ , нужно просуммировать значения, представленные формулой (1.55), отдельно для четных и нечетных  $N$ :

$$\begin{aligned} P(x(t)x(t-\tau)=1) &= \sum_{k=0}^{\infty} P(2k, |\tau|) = \sum_{k=0}^{\infty} \frac{(\lambda|\tau|)^{2k}}{(2k)!} e^{-\lambda|\tau|} = \\ &= \frac{1}{2}(e^{\lambda|\tau|} + e^{-\lambda|\tau|})e^{-\lambda|\tau|} = \frac{1}{2} + \frac{1}{2}e^{-2\lambda|\tau|}, \\ P(x(t)x(t-\tau)=-1) &= \sum_{k=0}^{\infty} P(2k+1, |\tau|) = \sum_{k=0}^{\infty} \frac{(\lambda|\tau|)^{2k+1}}{(2k+1)!} e^{-\lambda|\tau|} = \\ &= \frac{1}{2}(e^{\lambda|\tau|} - e^{-\lambda|\tau|})e^{-\lambda|\tau|} = \frac{1}{2} - \frac{1}{2}e^{-2\lambda|\tau|}. \end{aligned}$$

Полученные результаты позволяют рассчитать среднее значение произведения  $x(t)x(t-\tau)$ :

$$R_x(\tau) = 1 \cdot \left( \frac{1}{2} + \frac{1}{2}e^{-2\lambda|\tau|} \right) + (-1) \cdot \left( \frac{1}{2} - \frac{1}{2}e^{-2\lambda|\tau|} \right) = e^{-2\lambda|\tau|}.$$

Итак, корреляционная функция данного случайного процесса экспоненциально затухает с ростом абсолютного значения  $\tau$ . Теперь с помощью теоремы Винера — Хинчина можно найти спектральную плотность мощности:

$$W_x(\omega) = \int_{-\infty}^{\infty} R_x(\tau) e^{-j\omega\tau} d\tau = \int_{-\infty}^{\infty} e^{-2\lambda|\tau|} e^{-j\omega\tau} d\tau = \frac{4\lambda}{4\lambda^2 + \omega^2}.$$

## Интервал корреляции

Случайные процессы, встречающиеся в задачах обработки сигналов и изучаемые в радиотехнике, часто обладают следующим свойством: их функция корреляции стремится к нулю с увеличением временного сдвига  $\tau$  (напомним, что это является достаточным условием эргодичности процесса по математическому ожиданию). Чем быстрее убывает функция  $R(\tau)$ , тем слабее оказывается статистическая связь между мгновенными значениями случайного сигнала в два несовпадающих момента времени.

Числовой характеристикой, служащей для оценки "скорости изменения" реализаций случайного процесса, является *интервал корреляции*  $\tau_k$ , один из способов определения которого имеет вид

$$\tau_k = \frac{1}{R(0)} \int_0^\infty R(\tau) d\tau = \int_0^\infty r(\tau) d\tau.$$

Если известна информация о поведении какой-либо реализации случайного процесса "в прошлом", то возможен вероятностный прогноз случайного процесса на время порядка  $\tau_k$ .

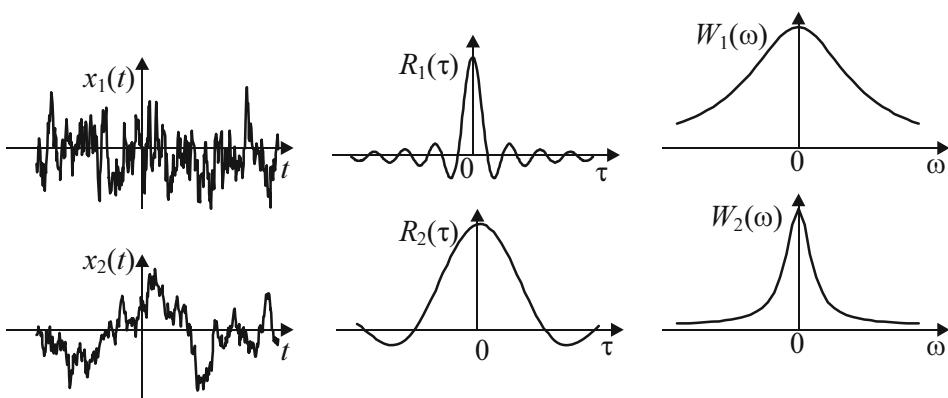
### Эффективная ширина спектра

Пусть исследуемый случайный процесс характеризуется спектром плотности мощности  $W(\omega)$ , имеющим максимальное значение  $W_{\max}$ . Заменим мысленно данный случайный процесс другим, у которого спектральная плотность мощности постоянна и равна  $W_{\max}$  в пределах некоторой полосы частот, выбираемой из условия равенства дисперсий (т. е. средних мощностей) обоих процессов. Ширина этой полосы частот называется *эффективной шириной спектра случайного процесса*:

$$W_{\max} \Delta\omega_{\text{eff}} = \int_{-\infty}^{\infty} W(\omega) d\omega, \quad \Delta\omega_{\text{eff}} = \frac{1}{W_{\max}} \int_{-\infty}^{\infty} W(\omega) d\omega.$$

Эффективную ширину спектра случайного процесса можно определить и другими способами, например, исходя из условия уменьшения значений спектра мощности на границе этого частотного интервала до некоторого уровня от максимума. В любом случае величины  $\tau_k$  и  $\Delta\omega_{\text{eff}}$  связаны известным из свойств преобразования Фурье соотношением неопределенности

$$\Delta\omega_{\text{eff}} \tau_k \sim 2\pi.$$



**Рис. 1.33.** Взаимосвязь между видом реализаций случайных процессов (слева), их корреляционными функциями (в центре) и спектрами (справа)

Для иллюстрации этого соотношения на рис. 1.33 в центре приведены примеры реализаций двух случайных процессов, слева — корреляционные функции этих процессов, а справа — их спектры плотности средней мощности.

## Белый шум

В радиотехнике белым шумом (*white noise*) называют стационарный случайный процесс, спектральная плотность мощности которого постоянна на всех частотах:

$$W(\omega) = W_0 = \text{const}.$$

Согласно теореме Винера — Хинчина, корреляционная функция белого шума представляет собой дельта-функцию:

$$R(\tau) = \frac{W_0}{2\pi} \int_{-\infty}^{\infty} e^{j\omega\tau} d\omega = W_0 \delta(\tau),$$

то есть равна нулю всюду, кроме точки  $\tau = 0$ . Дисперсия белого шума бесконечно велика.

В несовпадающие моменты времени значения белого шума некоррелированы — как бы ни был мал интервал  $\tau$ , сигнал за это время может измениться на любую величину.

Белый шум является абстрактной математической моделью и физически существовать не может. Это объясняется прежде всего бесконечностью его дисперсии (т. е. средней мощности). Однако в тех случаях, когда полоса пропускания исследуемой системы существенно уже эффективной ширины спектра шума, который на нее воздействует, можно для упрощения анализа приблизенно заменить реальный случайный процесс белым шумом.

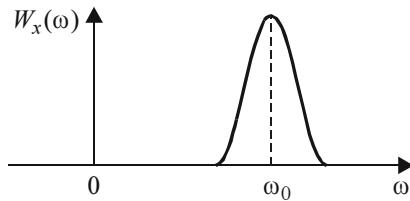
### ЗАМЕЧАНИЕ

Отметим еще раз, что вероятностные и корреляционные (или спектральные) характеристики случайного процесса — это совершенно разные и не связанные между собой функции. Так, например, нормальный случайный процесс может иметь разнообразную спектральную плотность мощности, а белый шум — произвольную функцию распределения. Единственная "точка соприкосновения" вероятностных и корреляционных характеристик — это возможность расчета дисперсии случайного процесса как на основе одномерной плотности вероятности (формула (1.60)), так и на основании корреляционной функции (формула (1.65)).

## Узкополосный случайный процесс

Важную роль в радиотехнике играет особый класс случайных процессов, спектр которых сосредоточен в относительно узкой полосе вблизи некоторой частоты  $\omega_0$ . Рассмотрим статистические свойства таких процессов.

Итак, пусть  $X(t)$  — случайный процесс, спектр плотности мощности которого  $W_x(\omega)$  имеет узкополосный характер (рис. 1.34). Будем также считать этот случайный процесс стационарным, нормальным и центрированным.



**Рис. 1.34.** Спектральная плотность мощности узкополосного случайного процесса

Согласно теореме Винера — Хинчина (см. формулу (1.76)), корреляционная функция и спектральная плотность мощности случайного процесса связаны друг с другом преобразованием Фурье. Узкополосный характер спектра  $W_x(\omega)$  говорит о том, что корреляционная функция  $R_x(\tau)$  имеет вид узкополосного радиосигнала:

$$R_x(\tau) = R_0(\tau) \cos(\omega_0 \tau + \phi_0(\tau)),$$

где  $R_0(\tau)$  и  $\phi_0(\tau)$  — медленно (по сравнению с  $\cos(\omega_0 \tau)$ ) меняющиеся функции.

Узкополосный спектр и осциллирующий характер корреляционной функции означают, что отдельные реализации узкополосного случайного процесса представляют собой *квазигармонические колебания*:

$$x(t) = A(t) \cos(\omega_0 t + \phi(t)),$$

у которых как огибающая  $A(t)$ , так и начальная фаза  $\phi(t)$  являются *случайными* функциями, медленно (по сравнению с  $\cos(\omega_0 t)$ ) изменяющимися во времени.

Чтобы определить статистические параметры огибающей и начальной фазы, рассмотрим комплексный аналитический сигнал  $\dot{Z}(t)$ , соответствующий вещественному случайному процессу  $X(t)$  (см. разд. "Комплексная огибающая" ранее в этой главе):

$$Z(t) = X(t) + jX_{\perp}(t),$$

где  $x_{\perp}(t)$  — сопряженный случайный процесс, реализации которого связаны с реализациями процесса  $x(t)$  преобразованием Гильберта, рассмотренным ранее в разд. "Преобразование Гильберта":

$$x_{\perp}(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(t')}{t - t'} dt'. \quad (1.77)$$

В этом же разделе было показано, что с помощью сопряженного сигнала можно определить мгновенные значения огибающей и полной фазы узкополосного сигнала:

$$A(t) = |\dot{z}(t)| = \sqrt{x^2(t) + x_{\perp}^2(t)},$$

$$\phi(t) = \arg \dot{z}(t) = \begin{cases} \arctg[x_{\perp}(t) / x(t)], & x(t) \geq 0, \\ \arctg[x_{\perp}(t) / x(t)] + \pi, & x(t) < 0. \end{cases}$$

Рассмотрим статистические свойства сопряженного процесса. Во-первых, определим его математическое ожидание, применив усреднение к формуле (1.77) и затем поменяв усреднение и интегрирование местами:

$$\overline{x_{\perp}(t)} = \overline{\frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(t')}{t-t'} dt'} = \frac{1}{\pi} \int_{-\infty}^{\infty} \overline{\frac{x(t')}{t-t'}} dt' = 0.$$

Результат равен нулю, т. к. процесс  $X(t)$  является центрированным.

Далее, поскольку процесс  $X(t)$  нормальный, а преобразование Гильберта является линейным интегральным преобразованием, то нормальным будет и сопряженный процесс  $X_{\perp}(t)$ .

Из свойств преобразования Гильберта (см. ранее разд. "Преобразование Гильберта", формула (1.35)) следует, что спектры конкретных реализаций процессов  $x(t)$  и  $x_{\perp}(t)$  связаны следующим образом:

$$\dot{S}_{x_{\perp}}(\omega) = \begin{cases} -j\dot{S}_x(\omega), & \omega > 0, \\ 0, & \omega = 0, \\ j\dot{S}_x(\omega), & \omega < 0, \end{cases}$$

откуда видно, что энергетические спектры реализаций процессов  $x(t)$  и  $x_{\perp}(t)$  совпадают, а следовательно, совпадают и спектральные плотности мощности этих процессов:  $W_{x_{\perp}}(\omega) = W_x(\omega)$ . Корреляционные функции связаны со спектрами плотности мощности обратным преобразованием Фурье, поэтому они тоже равны:  $R_{x_{\perp}}(\tau) = R_x(\tau)$ .

Нам осталось выяснить, имеется ли статистическая связь между процессами  $X(t)$  и  $X_{\perp}(t)$ . Ограничимся при этом расчетом корреляции между ними в совпадающие моменты времени, т. е. вычислим  $R_{xx_{\perp}}(0)$ :

$$R_{xx_{\perp}}(0) = \overline{x(t)x_{\perp}(t)} = \overline{x(t) \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(t')}{t-t'} dt'} = \frac{1}{\pi} \int_{-\infty}^{\infty} \overline{\frac{x(t)x(t')}{t-t'}} dt'.$$

Далее, как и ранее, внесем операцию статистического усреднения под знак интеграла, а затем используем замену переменной  $\tau = t - t'$ :

$$R_{xx_{\perp}}(0) = \frac{1}{\pi} \int_{-\infty}^{\infty} \overline{\frac{x(t)x(t')}{t-t'}} dt' = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{R_x(t-t')}{t-t'} dt' = -\frac{1}{\pi} \int_{-\infty}^{\infty} \frac{R_x(\tau)}{\tau} d\tau = 0.$$

Результат интегрирования равен нулю, т. к.  $R_x(\tau)$  является четной функцией, а все подынтегральное выражение, следовательно — нечетной. Таким образом, процессы  $X(t)$  и  $X_{\perp}(t)$  в совпадающие моменты времени некоррелированы. Поскольку они, кроме того, являются нормальными, то из некоррелированности следует статистическая независимость.

## Огибающая и полная фаза узкополосного случайного процесса

Мгновенное значение комплексного случайного процесса  $\dot{Z}(t)$  можно графически изобразить в виде вектора на комплексной плоскости (рис. 1.35). Проекции этого вектора на оси Re и Im равны мгновенным значениям процессов  $X(t)$  и  $X_{\perp}(t)$  соответственно. Эти мгновенные значения статистически независимы и имеют нормальные распределения с нулевым средним и одинаковыми дисперсиями (равенство дисперсий следует из равенства корреляционных функций). Поэтому совместная плотность вероятности процессов  $X(t)$  и  $X_{\perp}(t)$  равна произведению их одномерных плотностей вероятности, каждая из которых имеет вид (1.62):

$$p_{xx_{\perp}}(x, x_{\perp}) = p_x(x)p_{x_{\perp}}(x_{\perp}) = \frac{1}{2\pi\sigma_x^2} \exp\left(-\frac{x^2 + x_{\perp}^2}{2\sigma_x^2}\right). \quad (1.78)$$

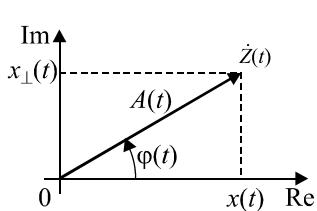


Рис. 1.35. Комплексный случайный процесс в виде вектора на комплексной плоскости

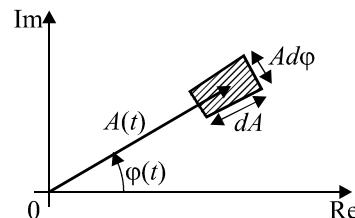


Рис. 1.36. Переход от декартовой системы координат к полярной

Для определения статистических свойств огибающей и фазы необходимо перейти в выражении (1.78) от декартовой  $(x, x_{\perp})$  к полярной системе координат  $(A, \phi)$  (см. рис. 1.35) и определить совместную плотность вероятности  $p_{A\phi}(A, \phi)$ . Связь между этими двумя системами координат выражается следующими формулами:

$$\begin{aligned} x &= A \cos \phi, \\ x_{\perp} &= A \sin \phi. \end{aligned}$$

Кроме того, вероятность попадания в бесконечно малую область в окрестности каждой точки комплексной плоскости при смене системы координат должна, очевидно, остаться неизменной. Площадь такой бесконечно малой области в декартовых координатах выражается как  $dx dx_{\perp}$ , а в полярных — как  $A dA d\phi$  (рис. 1.36). Таким образом, получаем

$$p_{xx_{\perp}}(x, x_{\perp}) dx dx_{\perp} = p_{xx_{\perp}}(A \cos \phi, A \sin \phi) A dA d\phi = p_{A\phi}(A, \phi) dA d\phi.$$

Отсюда видно, что искомая плотность вероятности выражается как

$$p_{A\phi}(A, \phi) = Ap_{x\hat{x}}(A \cos \phi, A \sin \phi) = \frac{A}{2\pi\sigma_x^2} \exp\left(-\frac{A^2}{2\sigma_x^2}\right). \quad (1.79)$$

Чтобы найти одномерные плотности вероятности для огибающей и фазы, нужно проинтегрировать двумерную плотность (1.79) по "лишним" координатам:

$$p_A(A) = \int_0^{2\pi} p_{A\phi}(A, \phi) d\phi, \quad p_\phi(\phi) = \int_0^\infty p_{A\phi}(A, \phi) dA.$$

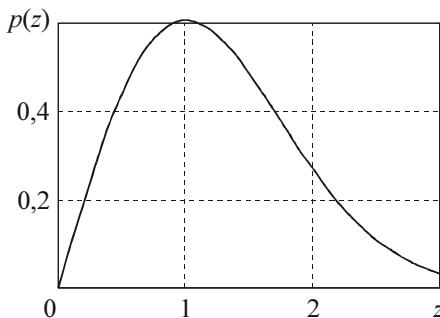
Так как двумерная плотность (1.79) не зависит от фазы  $\phi$ , плотность вероятности амплитуды рассчитывается элементарно:

$$p_A(A) = \int_0^{2\pi} \frac{A}{2\pi\sigma_x^2} \exp\left(-\frac{A^2}{2\sigma_x^2}\right) d\phi = \frac{A}{\sigma_x^2} \exp\left(-\frac{A^2}{2\sigma_x^2}\right). \quad (1.80)$$

Целесообразно перейти к безразмерной переменной  $z = A/\sigma_x$ , относительно которой

$$p(z)\sigma_x = z \exp(-z^2/2). \quad (1.81)$$

Плотность вероятности, описывающаяся законом (1.80) или (1.81), носит название *закона Рэлея (Rayleigh)*. График этого распределения, соответствующий формуле (1.81), приведен на рис. 1.37. Из графика видно, что наиболее вероятны некоторые средние (порядка  $\sigma_x$ ) значения огибающей. В то же время маловероятно, чтобы огибающая принимала значения как близкие к нулю, так и значительно превосходящие среднеквадратический уровень  $\sigma_x$  узкополосного процесса.



**Рис. 1.37.** Плотность вероятности огибающей узкополосного случайного процесса (закон Рэлея)

### ЗАМЕЧАНИЕ

Для работы с законом распределения Рэлея в статистическом пакете расширения MATLAB (Statistics Toolbox) имеются следующие функции: `raylrnd` (генерация случайных чисел, распределенных по закону Рэлея), `raylpdf` (расчет плотности вероятности), `raylcdf` (расчет функции распределения), `raylinv` (расчет обратной функции распределения) и `raylstat` (расчет математического ожидания и дисперсии).

Формула (1.80) позволяет известным способом (см. разд. "Вероятностные характеристики случайных процессов") вычислить среднее значение и дисперсию огибающей:

$$M\{A\} = \int_0^\infty A p_A(A) dA = \int_0^\infty \frac{A^2}{\sigma_x^2} \exp\left(-\frac{A^2}{2\sigma_x^2}\right) dA = \sqrt{\frac{\pi}{2}} \sigma_x \approx 1,253 \sigma_x,$$

$$\sigma_A^2 = M\{A^2\} - M^2\{A\} = \int_0^\infty \frac{A^3}{\sigma_x^2} \exp\left(-\frac{A^2}{2\sigma_x^2}\right) dA - \frac{\pi}{2} \sigma_x^2 = \left(2 - \frac{\pi}{2}\right) \sigma_x^2 \approx 0,429 \sigma_x^2.$$

Так как двумерная плотность (1.79) не зависит от фазы  $\phi$ , фаза имеет равномерное распределение на интервале  $[0, 2\pi]$ :

$$p_\phi(\phi) = \frac{1}{2\pi}, \quad 0 \leq \phi < 2\pi. \quad (1.82)$$

Физически это означает отсутствие какого-либо преимущественного значения полной фазы у отдельных реализаций узкополосного случайного процесса.

Из (1.79), (1.80) и (1.82) видно, что

$$p_{A\phi}(A, \phi) = p_A(A)p_\phi(\phi),$$

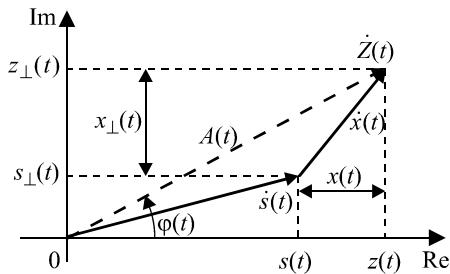
следовательно, амплитуда и полная фаза узкополосного случайного процесса в один и тот же момент времени являются статистически независимыми.

## Узкополосный случайный процесс при наличии детерминированной составляющей

Рассмотрим теперь ситуацию, когда к узкополосному шуму  $n(t)$  добавлен узкополосный детерминированный сигнал  $s(t)$ . Комплексный случайный процесс в данном случае будет иметь следующий вид:

$$\dot{Z}(t) = s(t) + js_\perp(t) + x(t) + jx_\perp(t).$$

Изображение мгновенного значения  $\dot{Z}(t)$  на комплексной плоскости будет отличаться от рис. 1.35 наличием детерминированного вектора  $\dot{s}(t)$  (рис. 1.38).



**Рис. 1.38.** Комплексный случайный процесс в виде вектора на комплексной плоскости при наличии детерминированной составляющей

Совместная плотность вероятности вещественной и мнимой частей этого комплексного процесса будет отличаться от (1.78) только смещением аргументов на  $s(t)$  и  $s_\perp(t)$ :

$$p_{zz_\perp}(z, z_\perp) = p_z(z)p_{z_\perp}(z_\perp) = \frac{1}{2\pi\sigma_x^2} \exp\left(-\frac{(z-s(t))^2 + (z_\perp-s_\perp(t))^2}{2\sigma_x^2}\right).$$

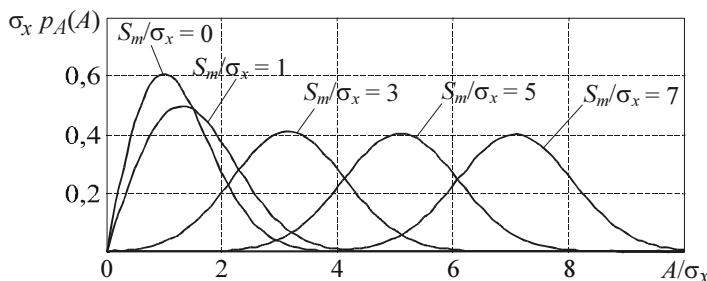
Переход от декартовой системы координат к полярной, аналогичный рассмотренному ранее (см. (1.79) и рис. 1.36), дает следующее:

$$\begin{aligned} p_{A\phi}(A, \phi) &= Ap_{zz_\perp}(A \cos \phi, A \sin \phi) = \\ &= \frac{A}{2\pi\sigma_x^2} \exp\left(-\frac{(A \cos \phi - s(t))^2 + (A \sin \phi - s_\perp(t))^2}{2\sigma_x^2}\right). \end{aligned}$$

Интегрирование этой двумерной плотности по фазе  $\phi$  дает одномерную плотность вероятности для амплитуды данного случайного процесса (промежуточные выкладки опущены):

$$p_A(A) = \int_0^{2\pi} p_{A\phi}(A, \phi) d\phi = \frac{A}{\sigma_x^2} \exp\left(-\frac{A^2 + S_m^2}{2\sigma_x^2}\right) I_0\left(\frac{AS_m}{\sigma_x^2}\right), \quad (1.83)$$

где  $S_m = \sqrt{s^2(t) + s_\perp^2(t)}$  — амплитудная огибающая детерминированного сигнала в данный момент времени,  $I_0(x)$  — модифицированная функция Бесселя первого рода нулевого порядка. Плотность вероятности (1.83) носит название *закона распределения Рэлея — Райса*. На рис. 1.39 показаны графики данной плотности вероятности, соответствующие разным отношениям сигнал/шум, т. е. разным значениям  $S_m/\sigma_x$ . При  $S_m = 0$  из (1.83) получается плотность вероятности, соответствующая закону Рэлея (1.81). При  $S_m/\sigma_x \gg 1$ , как видно из графиков, распределение огибающей приближается к нормальному закону.

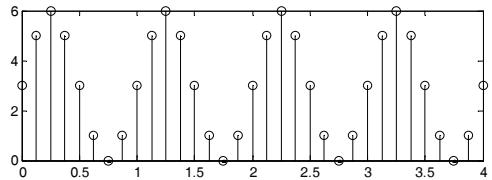


**Рис. 1.39.** Плотность вероятности огибающей узкополосного случайного процесса при наличии детерминированной составляющей (закон Рэлея — Райса)

### ЗАМЕЧАНИЕ

Для расчета плотности вероятности закона Рэлея — Райса с помощью MATLAB придется непосредственно воспользоваться формулой (1.83), поскольку пакет расширения Statistics не содержит специальных средств для этого. Функция распределения закона Рэлея — Райса выражается через так называемую  $Q$ -функцию Маркума. Для ее расчета в пакете расширения Signal Processing имеется функция `marcumq`.

## ГЛАВА 2



# Аналоговые системы

Данная глава, так же как и предыдущая, посвящена не цифровой, а аналоговой обработке сигналов. Ее цель — дать читателю представление о характеристиках и способах описания аналоговых систем. Понимание этих вопросов необходимо для более глубокого восприятия теории дискретных систем, поскольку многие методы анализа аналоговых и дискретных систем находятся в тесном родстве. Кроме того, в основе ряда методов проектирования дискретных фильтров лежит использование аналоговых прототипов, поэтому квалифицированное применение этих методов также требует знакомства с теорией аналоговых систем.

Итак, данная глава носит обзорный характер, чем и объясняются сжатое изложение и отсутствие конкретных примеров анализа прохождения сигналов через аналоговые системы.

### ЗАМЕЧАНИЕ

Положения, приводимые в данной главе, в литературе часто называют теорией линейных *цепей* с постоянными параметрами (см., например, [1, 2]). В данной книге используется термин "система", чтобы подчеркнуть высокогенеральный характер рассмотрения — система описывается только своими числовыми и функциональными характеристиками, без привлечения конкретных принципиальных схем.

## Классификация систем

Системы, используемые для преобразования сигналов, имеют самые разнообразные физические характеристики и могут классифицироваться по различным признакам.

Важнейшим классификационным признаком является линейность или нелинейность системы. *Линейными* называются системы, для которых выполняется принцип суперпозиции: реакция на сумму сигналов равна сумме реакций на эти сигналы, поданные на вход по отдельности. Системы, для которых принцип суперпозиции не выполняется, называются *нелинейными*.

Следующим критерием классификации систем является постоянство или непостоянство их характеристик во времени. Если произвольная задержка подаваемого на

вход сигнала приводит лишь к такой же задержке выходного сигнала, не меняя его формы, система называется *стационарной*, или *системой с постоянными параметрами*. В противном случае система называется *нестационарной, параметрической* или *системой с переменными параметрами*.

Два указанных способа классификации делят системы на четыре класса. В данной книге речь в основном пойдет о линейных стационарных системах. Исключение составляет *глава 9*, посвященная адаптивным фильтрам, которые являются системами с переменными параметрами.

## Характеристики линейных систем

Для рассматриваемых в этой главе линейных систем с постоянными параметрами справедливы принципы суперпозиции и стационарности. Это сильно упрощает анализ прохождения сигналов через такие системы, позволяя использовать для этого различные характеристики, речь о которых пойдет в данном разделе.

### Импульсная характеристика

Линейность и стационарность позволяют легко найти реакцию системы на любой входной сигнал, зная всего одну функцию — реакцию системы на поданную на вход дельта-функцию. Эта реакция, определяемая при нулевых начальных условиях, называется *импульсной характеристикой* системы и обозначается  $h(t)$ .

Любой сигнал может быть представлен в виде свертки самого себя с дельта-функцией (см. фильтрующее свойство дельта-функции (1.2) в разд. "Классификация сигналов" главы 1):

$$s_{\text{вх}}(t) = \int_{-\infty}^{\infty} s_{\text{вх}}(t') \delta(t - t') dt'.$$

Линейная система преобразует относительно переменной  $t$  все функции, входящие в это выражение. Входной сигнал  $s_{\text{вх}}(t)$  при этом превращается в выходной сигнал  $s_{\text{вых}}(t)$ , а дельта-функция  $\delta(t - t')$  — в импульсную характеристику  $h(t - t')$ . Функция  $s_{\text{вх}}(t')$  от  $t$  не зависит и поэтому остается без изменений. В результате получается формула, показывающая, что выходной сигнал линейной системы с постоянными параметрами равен свертке выходного сигнала и импульсной характеристики системы:

$$s_{\text{вых}}(t) = \int_{-\infty}^{\infty} s_{\text{вх}}(t') h(t - t') dt'. \quad (2.1)$$

#### ЗАМЕЧАНИЕ

Если входной и выходной сигналы системы имеют одинаковую физическую природу (т. е. одинаковую размерность), то импульсная характеристика, как и дельта-функция времени, имеет размерность частоты.

Формирование выходного сигнала можно пояснить следующим образом. Бесконечно малый "кусочек" входного сигнала  $s_{\text{вх}}(t')$  шириной  $dt'$  порождает на выходе отклик, представляющий собой импульсную характеристику, умноженную на  $s_{\text{вх}}(t') dt'$  и задержанную по времени на  $t'$ , т. е.  $s_{\text{вх}}(t') h(t - t') dt'$  (рис. 2.1). Чтобы получить значение выходного сигнала в момент времени  $t$ , нужно сложить вклады от всех этих бесконечно малых "кусочков", т. е. выполнить интегрирование по  $t'$ , что и дает приведенную выше формулу свертки (2.1)

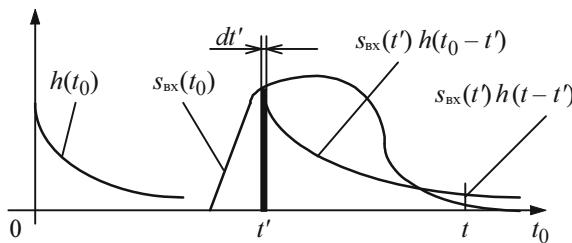


Рис. 2.1. Формирование выходной реакции цепи

#### ЗАМЕЧАНИЕ

Системы, имеющие вещественную импульсную характеристику, в дальнейшем будут называться *вещественными системами*.

## Переходная характеристика

*Переходной характеристикой*  $g(t)$  называют реакцию системы на поданную на вход функцию единичного скачка. Так же как и импульсная характеристика, переходная характеристика определяется при нулевых начальных условиях.

Поскольку дельта-функция — это производная от единичного скачка, импульсная и переходная характеристики связаны друг с другом операциями дифференцирования и интегрирования:

$$h(t) = \frac{dg(t)}{dt}, \quad g(t) = \int_{-\infty}^t h(t') dt'.$$

#### ЗАМЕЧАНИЕ

Если входной и выходной сигналы системы имеют одинаковую размерность, то переходная характеристика является безразмерной.

## Условие физической реализуемости

Любая физически реализуемая система обладает свойством *причинности* — выходная реакция не может возникнуть раньше входного сигнала. Отсюда следует, что для физически реализуемой системы импульсная и переходная характеристики должны быть равны нулю при  $t < 0$ .

### **ЗАМЕЧАНИЕ**

Под физической реализуемостью здесь понимается только выполнение свойства причинности, а не возможность создать систему в каком-нибудь конкретном виде. Дополнительные требования, которым должны удовлетворять характеристики систем, реализуемых в виде цепей с сосредоточенными параметрами, рассмотрены далее в разд. "Способы описания линейных систем" этой главы.

## **Комплексный коэффициент передачи**

Выходной сигнал линейной системы, как было показано выше, представляет собой свертку входного сигнала и импульсной характеристики. Преобразование Фурье от свертки дает произведение спектров сворачиваемых сигналов (см. формулу (1.22)), так что в частотной области прохождение сигнала через линейную систему описывается очень просто:

$$\dot{S}_{\text{вых}}(\omega) = \dot{S}_{\text{вх}}(\omega) \dot{K}(\omega).$$

Здесь  $\dot{K}(\omega)$  — преобразование Фурье импульсной характеристики системы:

$$\dot{K}(\omega) = \int_{-\infty}^{\infty} h(t) e^{-j\omega t} dt.$$

Эта функция называется *комплексным коэффициентом передачи* или *комплексной частотной характеристикой* системы, а ее модуль и фаза — соответственно амплитудно-частотной (АЧХ) и фазочастотной (ФЧХ) характеристиками системы.

Значение  $\dot{K}(\omega)$  показывает, как изменяется при прохождении через систему комплексная амплитуда синусоиды с частотой  $\omega$ . АЧХ показывает, во сколько раз изменится амплитуда синусоиды, а ФЧХ — каков будет полученный ею фазовый сдвиг.

## **Коэффициент передачи по мощности**

Мощность гармонического сигнала пропорциональна квадрату его амплитуды и не зависит от его фазы. Поэтому коэффициент передачи по мощности равен квадрату модуля комплексного коэффициента передачи, т. е. квадрату АЧХ:

$$K_{\text{мощ}}(\omega) = \dot{K}(\omega) \dot{K}^*(\omega) = |\dot{K}(\omega)|^2.$$

## **Фазовая и групповая задержка**

При преобразовании сигнала линейной системой различают два вида задержки. *Фазовая задержка* (*phase delay*) на частоте  $\omega$  — это задержка гармонического колебания с частотой  $\omega$ , проходящего через систему. Значение фазовой задержки равно фазовому сдвигу, вносимому системой, деленному на частоту гармонического колебания, с обратным знаком:

$$\tau_{\phi}(\omega) = -\varphi_K(\omega)/\omega. \quad (2.2)$$

*Групповая задержка (group delay)* на частоте  $\omega$  — это задержка *огибающей* узкополосного сигнала со средней частотой  $\omega$ . Групповая задержка равна производной от ФЧХ системы с обратным знаком:

$$\tau_{\text{гр}}(\omega) = -\frac{d\phi_K(\omega)}{d\omega}. \quad (2.3)$$

Пример, поясняющий разницу между фазовой и групповой задержкой, будет приведен применительно к дискретным системам (см. далее разд. "Групповая задержка" главы 4).

## Взаимный спектр выходного и входного сигналов

Взаимный спектр выходного и входного сигналов линейной системы легко найти, исходя из определения взаимного спектра (1.29) (см. разд. "Связь между корреляционными функциями и спектрами сигналов" главы 1):

$$\dot{S}_{\text{вых\_вх}}(\omega) = \dot{S}_{\text{вых}}(\omega) \dot{S}_{\text{вх}}^*(\omega) = \dot{S}_{\text{вх}}(\omega) \dot{K}(\omega) \dot{S}_{\text{вх}}^*(\omega) = \left| \dot{S}_{\text{вх}}(\omega) \right|^2 \dot{K}(\omega). \quad (2.4)$$

Отсюда следует, что комплексный коэффициент передачи системы равен отношению взаимного спектра выходного и входного сигналов к энергетическому спектру входного сигнала:

$$\dot{K}(\omega) = \frac{\dot{S}_{\text{вых\_вх}}(\omega)}{\left| \dot{S}_{\text{вх}}(\omega) \right|^2}.$$

### ЗАМЕЧАНИЕ

Аналогичная формула, связывающая коэффициент передачи со спектрами *случайных* сигналов, имеет большое значение для идентификации систем, т. е. оценки комплексного коэффициента передачи по результатам совместного наблюдения входного и выходного сигналов.

## Взаимная корреляция между входом и выходом

Применив обратное преобразование Фурье к формуле (2.4), получим выражение для ВКФ выходного и входного сигналов (используем при этом формулу (1.30) из разд. "Связь между корреляционными функциями и спектрами сигналов" главы 1):

$$B_{\text{вых\_вх}}(\tau) = \int_{-\infty}^{\infty} B_{\text{вх}}(\tau') h(\tau - \tau') d\tau'.$$

Итак, ВКФ выходного и входного сигналов линейной системы представляет собой свертку КФ входного сигнала с импульсной характеристикой системы.

## Преобразование случайного процесса в линейной системе

Как говорилось в *главе 1*, случайный процесс представляет собой ансамбль реализаций. Каждая отдельная реализация является детерминированным сигналом, и ее преобразование линейной системой анализируется с помощью формул, приведенных в этой главе ранее. В данном же разделе будет рассмотрено именно преобразование *статистических* характеристик случайного процесса. При этом подразумевается стационарный случайный процесс с нулевым математическим ожиданием.

### Спектральная плотность мощности

Поскольку спектром случайного процесса считается спектр его *мощности*, он преобразуется в линейной системе пропорционально коэффициенту передачи по мощности:

$$W_{\text{вых}}(\omega) = W_{\text{вх}}(\omega)K_{\text{мощ}}(\omega) = W_{\text{вх}}(\omega)|\dot{K}(\omega)|^2. \quad (2.5)$$

### Корреляционная функция

Согласно теореме Винера — Хинчина, корреляционная функция случайного процесса связана с его спектром преобразованием Фурье (см. разд. "Теорема Винера — Хинчина" главы 1). Применение преобразования Фурье к формуле (2.5) дает свертку:

$$R_{\text{вых}}(\tau) = \int_{-\infty}^{\infty} R_{\text{вх}}(\tau')B_h(\tau - \tau')d\tau'. \quad (2.6)$$

Здесь  $B_h(\tau)$  — результат обратного преобразования Фурье от коэффициента передачи по мощности  $|\dot{K}(\omega)|^2$ . Согласно разд. "Связь между корреляционными функциями и спектрами сигналов" главы 1, это преобразование дает корреляционную функцию импульсной характеристики системы:

$$B_h(\tau) = \int_{-\infty}^{\infty} h(t)h(t - \tau)dt.$$

### Дисперсия

Дисперсия случайного процесса равна значению его корреляционной функции при  $\tau = 0$ . Подстановка этой величины в формулу (2.6) для выходной корреляционной функции дает:

$$\sigma_{\text{вых}}^2 = R_{\text{вых}}(0) = \int_{-\infty}^{\infty} R_{\text{вх}}(\tau')B_h(\tau')d\tau'. \quad (2.7)$$

Можно рассчитать дисперсию и в частотной области (см. формулу (1.74) в разд. "Спектральные характеристики случайных процессов" главы 1). Воспользовавшись приведенной выше формулой (2.5) для выходного спектра, получаем:

$$\sigma_{\text{вых}}^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} W_{\text{вх}}(\omega) |\dot{K}(\omega)|^2 d\omega . \quad (2.8)$$

## Плотность вероятности

В общем случае плотность вероятности случайного процесса на выходе линейной системы не поддается расчету простыми средствами. Исключение составляет частный случай нормального случайного процесса, поскольку нормальное распределение остается нормальным при любых линейных преобразованиях. Поэтому нормальный случайный процесс с нулевым средним значением после прохождения через линейную систему сохранит свою нормальность и нулевое математическое ожидание, а его дисперсия может быть рассчитана по одной из формул (2.7), (2.8) предыдущего раздела.

### ЗАМЕЧАНИЕ

Если протяженность импульсной характеристики системы существенно превышает интервал корреляции входного случайного процесса, плотность вероятности сигнала на выходе, согласно центральной предельной теореме (см. разд. "Нормальное распределение" главы 1), приближается к нормальной.

## Частный случай белого шума

Если входной случайный процесс является белым шумом (см. разд. "Теорема Винера — Хинчина" главы 1), все приведенные ранее формулы существенно упрощаются:

$$\begin{aligned} W_{\text{вых}}(\omega) &= W_0 |\dot{K}(\omega)|^2 , \\ R_{\text{вых}}(\tau) &= W_0 B_h(\tau) = W_0 \int_{-\infty}^{\infty} h(t)h(t-\tau)dt , \\ D_{\text{вых}} &= W_0 B_h(0) = W_0 \int_{-\infty}^{\infty} h^2(t)dt = \frac{W_0}{2\pi} \int_{-\infty}^{\infty} |\dot{K}(\omega)|^2 d\omega . \end{aligned}$$

## Способы описания линейных систем

В данном разделе рассматриваются различные эквивалентные способы представления характеристик линейных систем, реализуемых в виде цепей с сосредоточенными параметрами. Понимание сущности этих вариантов представления и способов перехода от одного представления к другому важно для правильного использования соответствующих функций MATLAB.

## Дифференциальное уравнение

Связь между входным и выходным сигналами линейной цепи с сосредоточенными параметрами может быть выражена в виде дифференциального уравнения (ДУ) вида:

$$\begin{aligned} a_n \frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_{n-2} \frac{d^{n-2} y}{dt^{n-2}} + \dots + a_1 \frac{dy}{dt} + a_0 y(t) = \\ = b_m \frac{d^m x}{dt^m} + b_{m-1} \frac{d^{m-1} x}{dt^{m-1}} + b_{m-2} \frac{d^{m-2} x}{dt^{m-2}} + \dots + b_1 \frac{dx}{dt} + b_0 x(t). \end{aligned} \quad (2.9)$$

Здесь  $x(t)$  — входной сигнал,  $y(t)$  — выходной сигнал,  $a_i$  и  $b_i$  — постоянные коэффициенты. Таким образом, цепь описывается наборами коэффициентов  $\{a_i\}$  и  $\{b_i\}$ .

Должно выполняться неравенство  $m \leq n$ , т. е. максимальный порядок производной входного сигнала не может превышать максимального порядка производной выходного сигнала. Это связано с невозможностью реализации операции "чистого" дифференцирования аналоговой цепью. Значение  $n$  называется *порядком* цепи.

Если задать конкретный вид входного сигнала  $x(t)$ , получится линейное неоднородное дифференциальное уравнение с постоянными коэффициентами. Решение этого ДУ дает выходной сигнал  $y(t)$ .

## Функция передачи

Если применить к обеим частям приведенного в предыдущем разделе ДУ (2.9) преобразование Лапласа, получится выражение для *операторного коэффициента передачи*, или *функции передачи* цепи (*transfer function*):

$$H(p) = \frac{b_m p^m + b_{m-1} p^{m-1} + b_{m-2} p^{m-2} + \dots + b_1 p + b_0}{a_n p^n + a_{n-1} p^{n-1} + a_{n-2} p^{n-2} + \dots + a_1 p + a_0}. \quad (2.10)$$

Здесь  $a_i$  и  $b_i$  — те же постоянные коэффициенты, что и в приведенном ранее ДУ.

### ЗАМЕЧАНИЕ

Преобразование Лапласа [1—3] можно рассматривать как обобщение преобразования Фурье, при котором частота может принимать комплексные значения. Рассчитывается прямое преобразование Лапласа как  $F(p) = \int_{-\infty}^{\infty} f(t)e^{-pt} dt$  (сравните эту формулу с формулой прямого преобразования Фурье (1.14)). В этой главе для нас важно только то, что преобразование Лапласа является линейным и при дифференцировании сигнала во времени его преобразование Лапласа умножается на комплексную частоту  $p$ .

Комплексный коэффициент передачи получается из функции передачи (2.10) путем подстановки  $p = j\omega$ :

$$\dot{K}(\omega) = \frac{b_m (j\omega)^m + b_{m-1} (j\omega)^{m-1} + b_{m-2} (j\omega)^{m-2} + \dots + b_1 (j\omega) + b_0}{a_n (j\omega)^n + a_{n-1} (j\omega)^{n-1} + a_{n-2} (j\omega)^{n-2} + \dots + a_1 (j\omega) + a_0}.$$

## Нули и полюсы

Разложив числитель и знаменатель функции передачи (2.10) на множители, мы получим функцию передачи в следующем виде:

$$H(p) = k \frac{(p - z_m)(p - z_{m-1})(p - z_{m-2}) \dots (p - z_1)}{(p - p_n)(p - p_{n-1})(p - p_{n-2}) \dots (p - p_1)}. \quad (2.11)$$

Здесь  $k = b_m/a_n$  — коэффициент усиления (*gain*),  $z_i$  — нули функции передачи (*zero*),  $p_i$  — полюсы функции передачи (*pole*). В точках нулей  $H(z_i) = 0$ , а в точках полюсов  $H(p_i) \rightarrow \infty$ .

В данном случае цепь описывается набором параметров  $\{z_i\}$ ,  $\{p_i\}$ ,  $k$ .

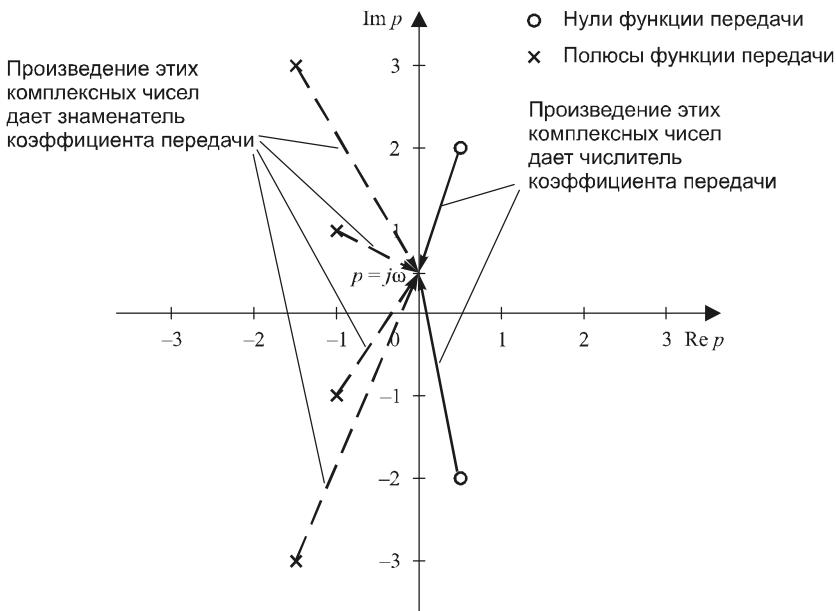
Для вещественных систем нули функции передачи являются вещественными либо составляют комплексно-сопряженные пары. То же относится и к полюсам. Коэффициент усиления таких систем всегда вещественный. В случае комплексных систем нули, полюсы и коэффициент передачи могут принимать произвольные комплексные значения.

## Связь АЧХ с расположением нулей и полюсов

Формула (2.11) дает возможность наглядно показать, как влияет расположение нулей и полюсов на АЧХ цепи. Разности вида  $(p - z_i)$ , произведение которых дает числитель формулы (2.11), можно представить на комплексной плоскости в виде векторов, соединяющих точки  $z_i$  и точку  $p = j\omega$ , расположенную на мнимой оси. Аналогичным образом можно показать на комплексной плоскости и разности  $(p - p_i)$ , произведение которых дает знаменатель формулы (2.11) (рис. 2.2).

При изменении частоты  $\omega$  соответствующая точка  $p = j\omega$  будет перемещаться вдоль мнимой оси, поэтому о поведении АЧХ системы можно сказать следующее:

- когда точка  $p = j\omega$  находится вблизи одного из нулей функции передачи  $z_i$ , соответствующая разность  $(p - z_i)$  окажется малой по сравнению с другими, в результате чего АЧХ в данной области частот будет иметь *провал*. Если нуль лежит на мнимой оси, АЧХ на соответствующей частоте будет иметь нулевое значение;
- когда точка  $p = j\omega$  находится вблизи одного из полюсов функции передачи  $p_i$ , соответствующая разность  $(p - p_i)$  окажется малой по сравнению с другими, в результате чего АЧХ в данной области частот будет иметь *подъем*. Если полюс лежит на мнимой оси, АЧХ на соответствующей частоте будет стремиться к бесконечности. Забегая немного вперед, скажем, что такая система находится на *границе устойчивости*;
- чем ближе к мнимой оси расположен нуль (полюс), тем более выраженным будет соответствующий провал (подъем) АЧХ.



**Рис. 2.2.** Влияние расположения нулей и полюсов на форму АЧХ системы

Таким образом, в приведенном на рис. 2.2 примере расположение нулей и полюсов позволяет сделать вывод о том, что АЧХ системы должна иметь пики в районе частот  $\omega = 1$  и  $\omega = 3$  и провал в районе  $\omega = 2$ . При этом пик в районе частоты  $\omega = 3$  должен быть выражен ярче, чем пик в районе частоты  $\omega = 1$ , т. к. соответствующая пара полюсов расположена ближе к мнимой оси. Проверим это:

```

>> z = [0.5+2i 0.5-2i]'; % нули функции передачи
>> p = [-1+1i -1-1i -1.5+3i -1.5-3i]'; % полюсы функции передачи
>> k = 1; % общий коэффициент усиления
>> [b, a] = zp2tf(z, p, k); % ищем функцию передачи
>> w = 0:0.01:5; % сетка частот для анализа
>> h = freqs(b, a, w); % расчет коэффициента передачи
>> plot(w, abs(h)) % график АЧХ
>> grid on

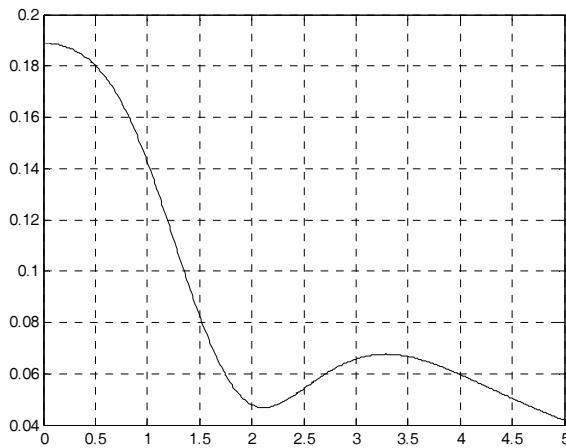
```

Результат, показанный на рис. 2.3, в целом подтверждает сделанные предположения. Правда, пики при  $\omega = \pm 1$  оказываются расположены слишком близко друг к другу, в результате чего они сливаются, образуя в итоге один пик на нулевой частоте.

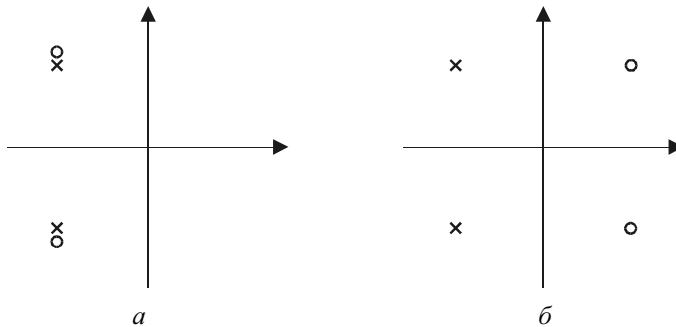
Приведенные соображения позволяют определить лишь самый общий характер поведения АЧХ системы. При расположении нулей рядом с полюсами их влияние на АЧХ может взаимно компенсироваться. То же самое имеет место при симметричном относительно мнимой оси расположении нуля и полюса.

Так, система, нули и полюсы которой расположены, как показано на рис. 2.4, а, имеет почти постоянные АЧХ и ФЧХ, а система, нули которой связаны с полюсами

как  $z_i = -p_i^*$  (см. рис. 2.4,  $\delta$ ), имеет строго постоянную АЧХ (к ФЧХ это в данном случае отнюдь не относится).



**Рис. 2.3.** АЧХ цепи подтверждает предсказания, сделанные исходя из расположения нулей и полюсов



**Рис. 2.4.** Компенсация влияния нулей и полюсов на АЧХ цепи

#### ЗАМЕЧАНИЕ

Системы типа рис. 2.4,  $\delta$ , имеющие строго постоянную АЧХ, называются *всепропускающими фильтрами (all-pass filter)* или *фазовыми звенями*.

## Полюсы и вычеты

Еще одним способом преобразования дробно-рациональной функции передачи (2.10) является ее представление в виде суммы простых дробей. При отсутствии кратных корней у знаменателя такое представление имеет следующий вид:

$$H(p) = \frac{r_n}{p - p_n} + \frac{r_{n-1}}{p - p_{n-1}} + \frac{r_{n-2}}{p - p_{n-2}} + \dots + \frac{r_1}{p - p_1} + C_0. \quad (2.12)$$

Здесь  $p_i$  — полюсы функции передачи, а числа  $r_i$  называются *вычетами*;  $C_0$  — целая часть функции передачи, отличная от нуля только в случае равенства степеней полиномов числителя и знаменателя.

В данном случае цепь описывается набором параметров  $\{r_i\}$ ,  $\{p_i\}$ ,  $C_0$ .

Для вещественных систем полюсы функции передачи могут быть вещественными либо составлять комплексно-сопряженные пары. Вычеты, соответствующие комплексно-сопряженным полюсам, при этом также являются комплексно-сопряженными.

При наличии кратных полюсов функции передачи разложение на простые дроби становится сложнее. Каждый  $m$ -кратный полюс  $p_i$  дает  $m$  слагаемых следующего вида:

$$\frac{r_{i1}}{p - p_i} + \frac{r_{i2}}{(p - p_i)^2} + \frac{r_{i3}}{(p - p_i)^3} + \dots + \frac{r_{im}}{(p - p_i)^m}. \quad (2.13)$$

## Расчет импульсной характеристики

Представление функции передачи в виде суммы простых дробей позволяет вычислить импульсную характеристику системы, поскольку каждое слагаемое функции передачи вида  $r_i / (p - p_i)$  соответствует слагаемому импульсной характеристики вида

$$r_i e^{p_i t}, t \geq 0.$$

Пара комплексно-сопряженных полюсов дает пару слагаемых импульсной характеристики в виде комплексно-сопряженных экспонент. Сумма таких слагаемых представляет собой синусоиду с экспоненциально меняющейся амплитудой:

$$\begin{aligned} r_i \exp(p_i t) + r_i^* \exp(p_i^* t) &= 2 \operatorname{Re}[r_i \exp(p_i t)] = \\ &= 2 \operatorname{Re}[|r_i| \exp(j \arg(r_i)) \exp(\operatorname{Re}(p_i)t) \exp(j \operatorname{Im}(p_i)t)] = \\ &= 2 |r_i| \exp(\operatorname{Re}(p_i)t) \cos(\operatorname{Im}(p_i)t + \arg(r_i)). \end{aligned}$$

Здесь  $\arg(r_i)$  — фаза комплексного числа  $r_i$ .

Что касается кратных полюсов, то  $m$ -кратный полюс  $p_i$  даст в выражении для импульсной характеристики  $m$  слагаемых следующего вида:

$$r_{i1} e^{p_i t} + r_{i2} t e^{p_i t} + r_{i3} \frac{t^2}{2} e^{p_i t} + \dots + r_{im} \frac{t^{m-1}}{(m-1)!} e^{p_i t}.$$

## Устойчивость линейных систем

Система называется *устойчивой*, если при нулевом входном сигнале выходной сигнал затухает при любых начальных условиях:

$$\lim_{t \rightarrow \infty} s_{\text{вых}}(t) = 0 \text{ при } s_{\text{вх}}(t) = 0.$$

Это требование равносильно требованию затухания импульсной характеристики:

$$\lim_{t \rightarrow \infty} h(t) = 0.$$

В предыдущем разделе было показано, что импульсная характеристика системы в общем случае содержит слагаемые вида

$$r_i \frac{t^k}{k!} e^{p_i t},$$

где  $p_i$  — полюсы функции передачи системы,  $r_i$  — соответствующие им вычеты,  $k$  — целые числа в диапазоне от нуля до значения, на единицу меньшего кратности полюса  $p_i$ .

Такие слагаемые при  $t \rightarrow \infty$  затухают, если вещественная часть полюса  $p_i$  является отрицательной:

$$\operatorname{Re}(p_i) < 0.$$

Отсюда получаем общее условие: линейная система является устойчивой тогда и только тогда, когда полюсы ее функции передачи лежат в левой комплексной полуплоскости.

### Частотная характеристика неустойчивой системы

К сожалению, довольно широко распространено представление, что АЧХ неустойчивой системы должна иметь пики, уходящие в бесконечность. Однако формула (2.11) показывает, что это не так. Из нуль-полюсного представления функции передачи следует, что АЧХ стремится к бесконечности при наличии полюса, расположенного на мнимой оси. Согласно сформулированному в предыдущем разделе условию это означает, что система находится на грани устойчивости.

Если же полюсов на мнимой оси нет, частотная характеристика системы на всех частотах будет конечной, независимо от выполнения условия устойчивости.

Какой же смысл имеет частотная характеристика неустойчивой системы? Напомню, что комплексный коэффициент передачи показывает, как изменяются амплитуда и начальная фаза синусоиды с некоторой частотой  $\omega$ . При этом имеется в виду синусоида, бесконечная во времени, т. е. существующая при  $-\infty < t < \infty$ . Таким образом, выражения

$$x(t) = A \cos(\omega t + \phi), \quad (2.14)$$

$$y(t) = A |\dot{K}(\omega)| \cos(\omega t + \phi + \phi_K(\omega)) \quad (2.15)$$

должны удовлетворять дифференциальному уравнению системы при всех  $t$ . Бесконечное во времени решение такого вида может быть найдено как для устойчивой, так и для неустойчивой системы.

Если же мы подаем на вход системы синусоиду, начиная с некоторого момента времени, то выходной сигнал будет зависеть не только от входного, но и от начальных условий (внутреннего состояния) системы. Для устойчивой системы при любых начальных условиях переходный процесс является затухающим, и при вход-

ном сигнале (2.14) выходной сигнал будет стремиться к (2.15) при  $t \rightarrow \infty$ . Для неустойчивой системы результат зависит от начальных условий — в одних случаях переходный процесс также затухает и выходной сигнал стремится к (2.15), в других — выходной сигнал неограниченно возрастает.

## Пространство состояний

Еще одним способом описания линейной цепи является ее представление в *пространстве состояний* (*state space*). При этом состояние цепи описывается *вектором состояния*  $\mathbf{s}(t)$ , а собственные колебания цепи и ее реакция на входной сигнал  $x(t)$  характеризуются следующим образом:

$$\begin{aligned}\mathbf{s}'(t) &= \mathbf{A}\mathbf{s}(t) + \mathbf{B}x(t), \\ y(t) &= \mathbf{C}\mathbf{s}(t) + Dx(t).\end{aligned}\quad (2.16)$$

Если размерность вектора состояния  $\mathbf{s}(t)$  равна  $N$  ( $\mathbf{s}(t)$  — вектор-столбец), а входной  $x(t)$  и выходной  $y(t)$  сигналы являются скалярными, то размерность параметров в этих формулах будет следующей:  $\mathbf{A}$  — матрица  $N \times N$ ,  $\mathbf{B}$  — столбец  $N \times 1$ ,  $\mathbf{C}$  — строка  $1 \times N$ ,  $D$  — скаляр. Если входной и/или выходной сигналы являются векторными, размерность параметров соответствующим образом изменяется.

Описанием цепи в данном случае является набор параметров  $\mathbf{A}, \mathbf{B}, \mathbf{C}, D$ .

От представления цепи в пространстве состояний можно легко перейти к функции передачи цепи. Если применить преобразование Лапласа к уравнениям состояния (2.16), а затем выразить из них операторный коэффициент передачи, получится следующее:

$$H(p) = D - \mathbf{C}(\mathbf{A} - s\mathbf{I})^{-1}\mathbf{B}. \quad (2.17)$$

Здесь  $\mathbf{I}$  — единичная матрица  $N \times N$ .

С обратным преобразованием дело обстоит не так просто. Прежде всего заметим, что это преобразование не является однозначным — оно зависит от выбора вектора состояния. Действительно, пусть у нас есть описание некоторой системы в пространстве состояний в форме (2.16). Введем новый вектор состояния  $s_1(t)$ , получаемый из исходного вектора  $\mathbf{s}(t)$  линейным преобразованием, т. е. путем умножения на квадратную невырожденную матрицу  $\mathbf{L}$ :

$$s_1(t) = \mathbf{L} s(t), \text{ и, следовательно, } s(t) = \mathbf{L}^{-1} s_1(t).$$

Подставив последнее выражение для  $\mathbf{s}(t)$  в (2.16), получаем

$$\begin{aligned}\mathbf{L}^{-1}\mathbf{s}'_1(t) &= \mathbf{A}\mathbf{L}^{-1}\mathbf{s}_1(t) + \mathbf{B}x(t), \\ y(t) &= \mathbf{C}\mathbf{L}^{-1}\mathbf{s}_1(t) + Dx(t).\end{aligned}$$

Чтобы привести эти уравнения к форме (2.16), необходимо умножить первое из них на  $\mathbf{L}$  слева:

$$\begin{aligned}\mathbf{s}'_1(t) &= \mathbf{L}\mathbf{A}\mathbf{L}^{-1}\mathbf{s}_1(t) + \mathbf{L}\mathbf{B}x(t), \\ y(t) &= \mathbf{C}\mathbf{L}^{-1}\mathbf{s}_1(t) + Dx(t).\end{aligned}$$

Сравнивая полученные равенства с (2.16), получаем выражения для параметров пространства состояний, соответствующих новому вектору состояния:

$$\mathbf{A}_1 = \mathbf{L} \mathbf{A} \mathbf{L}^{-1}, \quad \mathbf{B}_1 = \mathbf{L} \mathbf{B}, \quad \mathbf{C}_1 = \mathbf{C} \mathbf{L}^{-1}, \quad D_1 = D.$$

Таким образом, при смене вектора состояния параметры  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $D$  соответствующим образом модифицируются.

Теперь рассмотрим один из возможных вариантов преобразования наборов коэффициентов  $\{a_i\}$ ,  $\{b_i\}$  в параметры пространства состояний системы.

### ЗАМЕЧАНИЕ

Здесь рассматривается именно тот вариант преобразования, который используется соответствующей функцией MATLAB.

Введем обозначение  $y_0(t)$  для сигнала, являющегося решением дифференциального уравнения (2.9), из которого удалены производные от входного сигнала в правой части:

$$a_n \frac{d^n y_0}{dt^n} + a_{n-1} \frac{d^{n-1} y_0}{dt^{n-1}} + a_{n-2} \frac{d^{n-2} y_0}{dt^{n-2}} + \dots + a_1 \frac{dy_0}{dt} + a_0 y_0(t) = b_0 x(t). \quad (2.18)$$

Выходной сигнал системы  $y(t)$  связан с  $y_0(t)$  следующим соотношением (для его доказательства достаточно продифференцировать (2.18) по времени и рассмотреть линейную комбинацию производных необходимых порядков):

$$y(t) = y_0(t) + \frac{b_1}{b_0} \frac{dy_0}{dt} + \frac{b_2}{b_0} \frac{d^2 y_0}{dt^2} + \dots + \frac{b_m}{b_0} \frac{d^m y_0}{dt^m}. \quad (2.19)$$

Далее реализуем представление в пространстве состояний для дифференциального уравнения (2.18). В качестве  $n$  элементов вектора состояния выберем сигнал  $y_0(t)a_n/b_0$  и  $n - 1$  его производных, причем  $(n - 1)$ -я производная будет первым элементом вектора, а сигнал  $y_0(t)a_n/b_0$  — последним:

$$s_k(t) = \frac{a_n}{b_0} \frac{d^{n-k} y_0}{dt^{n-k}}, \quad k = 1, 2, \dots, n. \quad (2.20)$$

При таком выборе вектора состояния уравнение (2.18) можно переписать следующим образом:

$$\frac{ds_1}{dt} + \frac{a_{n-1}}{a_n} s_1(t) + \frac{a_{n-2}}{a_n} s_2(t) + \dots + \frac{a_1}{a_n} s_{n-1}(t) + \frac{a_0}{a_n} s_n(t) = x(t).$$

Отсюда получаем формулу для расчета производной первого элемента вектора состояния:

$$\frac{ds_1}{dt} = -\frac{a_{n-1}}{a_n} s_1(t) - \frac{a_{n-2}}{a_n} s_2(t) - \dots - \frac{a_1}{a_n} s_{n-1}(t) - \frac{a_0}{a_n} s_n(t) + x(t). \quad (2.21)$$

Производные остальных элементов вектора состояния, согласно (2.20), равны значениям соседних элементов этого вектора:

$$\frac{ds_n}{dt} = s_{n-1}(t), \quad \frac{ds_{n-1}}{dt} = s_{n-2}(t), \quad \dots, \quad \frac{ds_3}{dt} = s_2(t), \quad \frac{ds_2}{dt} = s_1(t). \quad (2.22)$$

Объединяя (2.22) и (2.21), получаем систему, соответствующую первому матричному уравнению описания системы в пространстве состояний (2.16). Переписав эту систему в матричном виде, получаем значения матриц **A** и **B**:

$$\underbrace{\begin{bmatrix} \frac{ds_1}{dt} \\ \frac{ds_2}{dt} \\ \frac{ds_3}{dt} \\ \vdots \\ \frac{ds_{n-1}}{dt} \\ \frac{ds_n}{dt} \end{bmatrix}}_{\mathbf{s}'(t)} = \underbrace{\begin{bmatrix} -\frac{a_{n-1}}{a_n} & -\frac{a_{n-2}}{a_n} & \dots & -\frac{a_2}{a_n} & -\frac{a_1}{a_n} & -\frac{a_0}{a_n} \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} s_1(t) \\ s_2(t) \\ s_3(t) \\ \vdots \\ s_{n-1}(t) \\ s_n(t) \end{bmatrix}}_{\mathbf{s}(t)} + \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{B}} x(t).$$

Осталось разобраться со вторым уравнением системы (2.16), т. е. с формированием выходного сигнала  $y(t)$ . Уравнение (2.19) с учетом введенного вектора состояния принимает следующий вид:

$$y(t) = \frac{b_0}{a_n} s_n(t) + \frac{b_1}{a_n} s_{n-1}(t) + \frac{b_2}{a_n} s_{n-2}(t) + \dots + \frac{b_m}{a_n} s_{n-m}(t). \quad (2.23)$$

Если  $m < n$ , отсюда очевидным образом получаются выражения для параметров **C** и **D**:

$$\mathbf{C} = \left[ \begin{array}{ccccc} \frac{b_{n-1}}{a_n} & \frac{b_{n-2}}{a_n} & \dots & \frac{b_1}{a_n} & \frac{b_0}{a_n} \end{array} \right], \quad D = 0. \quad (2.24)$$

Если  $m = n$ , то последнее слагаемое (2.23) равно  $\frac{b_n}{a_n} \frac{ds_1}{dt}$ , и для его вычисления придется воспользоваться выражением (2.21):

$$\begin{aligned} \frac{b_n}{a_n} \frac{ds_1}{dt} &= -\frac{b_n a_{n-1}}{a_n^2} s_1(t) - \frac{b_n a_{n-2}}{a_n^2} s_2(t) - \dots - \\ &\quad - \frac{b_n a_1}{a_n^2} s_{n-1}(t) - \frac{b_n a_0}{a_n^2} s_n(t) + \frac{b_n}{a_n} x(t) \end{aligned} \quad (2.25)$$

Комбинируя (2.23) и (2.25), получаем

$$\begin{aligned} y(t) = & \left( \frac{b_0}{a_n} - \frac{b_n a_0}{a_n^2} \right) s_n(t) + \left( \frac{b_1}{a_n} - \frac{b_n a_1}{a_n^2} \right) s_{n-1}(t) + \left( \frac{b_2}{a_n} - \frac{b_n a_2}{a_n^2} \right) s_{n-2}(t) + \dots + \\ & + \left( \frac{b_{n-2}}{a_n} - \frac{b_n a_{n-2}}{a_n^2} \right) s_2(t) + \left( \frac{b_{n-1}}{a_n} - \frac{b_n a_{n-1}}{a_n^2} \right) s_1(t) + \frac{b_n}{a_n} x(t). \end{aligned}$$

Это выражение дает следующие значения для параметров **C** и **D**:

$$\mathbf{C} = \left[ \left( \frac{b_{n-1}}{a_n} - \frac{b_n a_{n-1}}{a_n^2} \right) \left( \frac{b_{n-2}}{a_n} - \frac{b_n a_{n-2}}{a_n^2} \right) \dots \left( \frac{b_1}{a_n} - \frac{b_n a_1}{a_n^2} \right) \left( \frac{b_0}{a_n} - \frac{b_n a_0}{a_n^2} \right) \right], \quad D = \frac{b_n}{a_n}.$$

Данный вариант является более общим — более простые выражения (2.24) получаются из него в случае, когда  $b_n = 0$ .

## Функции MATLAB для расчета линейных цепей

MATLAB и его пакеты расширения ориентированы прежде всего на цифровую обработку сигналов, поэтому функции, связанные с расчетом аналоговых цепей, рассматриваются как вспомогательные. В основном они предназначены для вызова из других функций, использующих аналоговые прототипы при синтезе цифровых фильтров. Однако и сами по себе эти функции могут быть весьма полезны.

Большая часть рассматриваемых в данном разделе функций относится к пакету Signal Processing.

### Расчет частотных характеристик

Как уже говорилось, для расчета комплексного коэффициента передачи необходимо подставить в функцию передачи мнимый аргумент:  $p = j\omega$ . Соответствующие расчеты выполняются с помощью функции `freqs`. В простейшем виде она имеет следующий синтаксис:

```
freqs(b, a);
```

Здесь **b** и **a** — векторы коэффициентов полиномов, соответственно, числителя и знаменателя функции передачи. Коэффициенты следуют в порядке *убывания степеней*, заканчивая постоянным слагаемым.

Для расчета характеристики по умолчанию выбираются 200 частот, логарифмически равномерно распределенных в диапазоне, автоматически выбираемом функцией таким образом, чтобы отразить значимый фрагмент частотных характеристик ("за кадром" при этом остаются лишь участки, где поведение графиков определяется очевидными асимптотами:  $\dot{K}(\omega) \approx b_0 / a_0$  при  $\omega \rightarrow 0$  и  $\dot{K}(\omega) \approx b_m / (a_n(j\omega)^{n-m})$  при  $\omega \rightarrow \infty$ ).

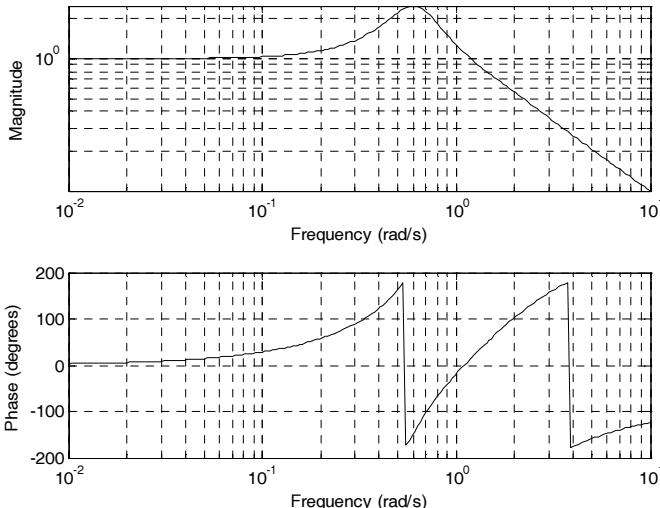
При отсутствии выходных параметров функция `freqs` строит графики АЧХ и ФЧХ. АЧХ выводится в логарифмическом масштабе (но без пересчета в децибелы), ФЧХ — в градусах.

Пусть анализируемая цепь имеет функцию передачи

$$H(p) = \frac{p^3 + 3p^2 + 3p + 1}{p^4 - 3p^3 + 4p^2 - 2p + 1}. \quad (2.26)$$

Строим ее АЧХ и ФЧХ (рис. 2.5):

```
>> b = [1 3 3 1];
>> a = [1 -3 4 -2 1];
>> freqs(b, a)
```



**Рис. 2.5.** Частотные характеристики, построенные функцией freqs

Чтобы вместо построения графика получить вектор рассчитанных значений комплексного коэффициента передачи, нужно присвоить результат, возвращаемый функцией freqs, какой-либо переменной:

```
h = freqs(b, a);
```

Если использовать второй выходной параметр, то в нем функция возвратит вектор частот, для которых рассчитаны значения импульсной характеристики:

```
[h, w] = freqs(b, a);
```

Рассчитаем вектор значений частотной характеристики цепи из нашего примера и построим график ее АЧХ в линейном, а не логарифмическом масштабе (рис. 2.6):

```
>> [h, w] = freqs(b, a);
>> plot(w, abs(h))
>> grid on
```

Можно принудительно задать количество частотных точек для анализа, используя третий входной параметр n (при этом частоты по-прежнему логарифмически распределены в автоматически выбранном диапазоне):

```
[h, w] = freqs(b, a, n);
```

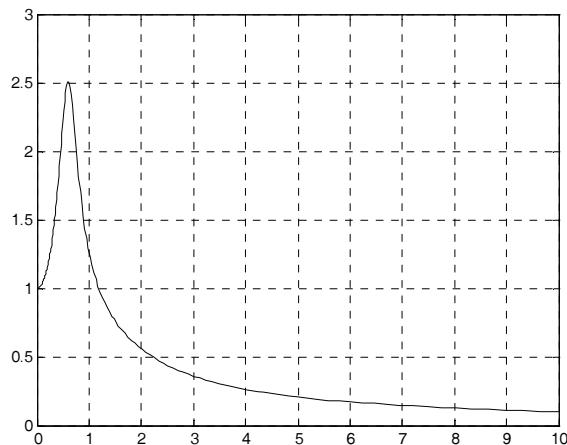


Рис. 2.6. АЧХ цепи в линейном масштабе

Наконец, можно принудительно задать частоты для анализа — также с помощью третьего входного параметра, которым в данном случае является вектор круговых частот  $w$ . Признаком, отличающим данную ситуацию от предыдущей, является векторный, а не скалярный, характер третьего входного параметра  $w$ :

```
h = freqs(b, a, w);
```

#### ЗАМЕЧАНИЕ

Для дискретных линейных систем, функция передачи которых задана в виде отношения полиномов в  $z$ -области, аналогичные расчеты и построения выполняются функцией freqz (см. главу 4).

### Построение графиков фазочастотных характеристик

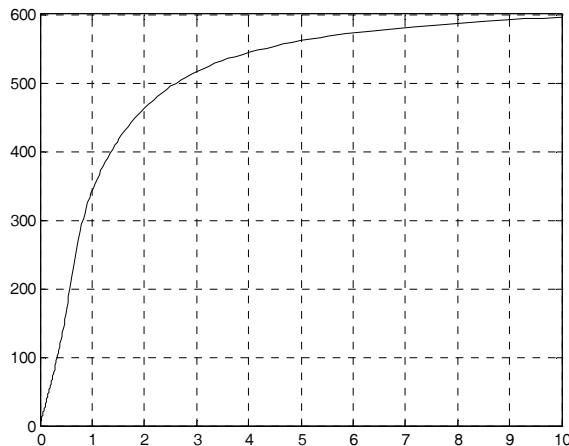
Как видно из рис. 2.5, ФЧХ цепи содержит большое количество разрывов (скачков). Те из них, величины которых равны  $180^\circ$ , действительно являются скачками ФЧХ, а остальные, величина которых составляет  $360^\circ$ , являются "фиктивными". Они возникают только из-за того, что результаты вычисления фазы комплексного числа всегда лежат в диапазоне  $\pm 180^\circ$ . Наличие этих многочисленных скачков затрудняет восприятие истинной формы ФЧХ и маскирует скачки "настоящие".

Избавиться от лишних разрывов позволяет функция unwrap, которая ищет в переданном ей векторе скачки между соседними элементами, превышающие заданную пороговую величину (по умолчанию равную  $\pi$ ) и сдвигает соответствующие фрагменты вектора на  $\pm 2\pi$  нужное число раз.

Продемонстрируем действие функции unwrap, построив график ФЧХ для цепи из примера предыдущего раздела (рис. 2.7):

```
>> [h, w] = freqs(b, a);
>> phi = angle(h); % ФЧХ
```

```
>> phi = unwrap(phi); % устранение скачков
>> plot(w, phi*180/pi) % отображаем в градусах
>> grid on
```



**Рис. 2.7.** ФЧХ цепи с устранимыми скачками на  $360^\circ$

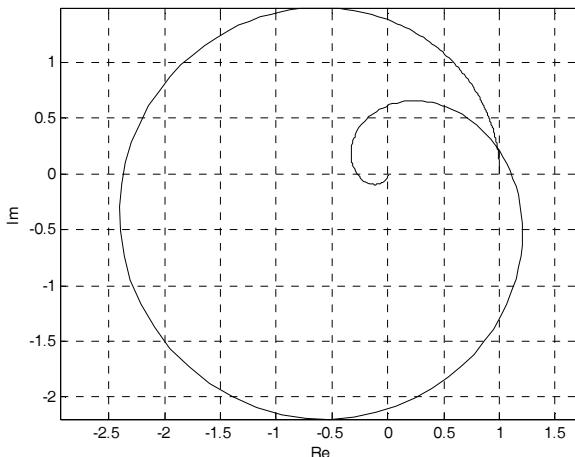
Сравнение графиков ФЧХ на рис. 2.5 и 2.7 наглядно демонстрирует сущность функции `unwrap`.

## Построение годографа функции передачи

*Годографом* называется траектория, которую описывает на комплексной плоскости коэффициент передачи системы при изменении частоты. Поскольку функция `freqs` при вызове с выходными параметрами возвращает вектор значений комплексного коэффициента передачи, годограф можно построить очень просто — функция `plot` при передаче ей вектора комплексных чисел сделает как раз то, что нужно. Таким образом, простейший вариант кода для построения годографа выглядит так:

```
plot(freqs(b, a))
```

Однако этот вариант построит только половину годографа, поскольку функция `freqs` рассчитывает коэффициент передачи только для положительных частот. Если необходим полный годограф, можно воспользоваться свойством комплексно-сопряженной симметрии коэффициента передачи вещественной системы, дополнив вектор значений зеркально перевернутой комплексно-сопряженной половиной. Кроме того, при заданном по умолчанию числе точек, рассчитываемых функцией `freqs`, годограф может выглядеть угловатым. В таком случае следует увеличить число точек, указав его с помощью третьего входного параметра функции `freqs`, или вообще задать вектор частот вручную. В последнем случае полезно включить в него нулевое и бесконечное значения. Размещаемый между ними вектор частот, как правило, целесообразно формировать в виде логарифмически равномерной сетки с помощью функции `logspace`. Крайние значения частот этой сетки, равно как и число точек, подбираются экспериментально — полученный график годографа не должен иметь заметных на глаз изломов.



**Рис. 2.8.** Годограф системы, имеющей функцию передачи (2.26)

В качестве примера построим годограф для рассматривавшейся ранее системы (см. (2.26) и рис. 2.5—2.7):

```
>> b = [1 3 3 1]; % коэффициенты числителя функции передачи
>> a = [1 -3 4 -2 1]; % коэффициенты знаменателя функции передачи
>> w = [0 logspace(-2, 2, 500) inf]; % вектор частот
>> plot(freqs(b, a, w)) % построение годографа
>> axis equal % выравнивание масштаба осей
>> grid on % вывод сетки
>> xlabel('Re')
>> ylabel('Im')
```

Результат построения показан на рис. 2.8.

#### **ЗАМЕЧАНИЕ**

При построении годографа целесообразно выровнять масштаб по вещественной и мнимой осям с помощью команды `axis equal`, а также вывести на график сетку с помощью команды `grid on`.

## **Преобразование способов описания линейных цепей**

В теоретической части данной главы было рассмотрено несколько эквивалентных способов описания линейных цепей. В пакете Signal Processing существует ряд функций, предназначенных для преобразования описаний из одной формы в другую. Имена этих функций имеют вид `xx2yy`, где `xx` — обозначение исходной формы описания, а `yy` — обозначение целевой формы описания цепи.

#### **ЗАМЕЧАНИЕ**

По-английски цифра "2" (two) произносится сходно с частицей "to", служащей, среди прочего, для указания на конечную цель или результат какого-либо процесса. Поэтому цифра "2" традиционно используется в середине идентификаторов функций, осуществляющих различные преобразования.

Формы описания цепей в именах функций обозначаются следующим образом:

- `tf` — коэффициенты полиномов числителя и знаменателя функции передачи (transfer function);
- `zp` — нули и полюсы (zeros and poles);
- `ss` — описание в пространстве состояний (state-space).

Необходимость в преобразовании описаний часто возникает из-за того, что функции расчета цепей (такие как рассматриваемые далее функции расчета фильтров-прототипов) дают результат в одной форме, а функция, например, построения частотной характеристики требует задания входных параметров в другой форме.

Далее кратко рассматриваются конкретные функции преобразования описаний цепей. Для входных и выходных параметров используются следующие обозначения:

- функция передачи:
  - `b` — вектор-строка коэффициентов (в порядке убывания степеней) числителя функции передачи;
  - `a` — вектор-строка коэффициентов (в порядке убывания степеней) знаменателя функции передачи;
- нули и полюсы:
  - `z` — вектор нулей (столбец);
  - `p` — вектор полюсов (столбец);
  - `k` — коэффициент усиления (скаляр);
- пространство состояний:
  - `A` — квадратная матрица связи вектора состояния и его производной;
  - `B` — вектор-столбец связи входного сигнала и производной вектора состояния;
  - `C` — вектор-строка связи выходного сигнала и вектора состояния;
  - `D` — скалярный коэффициент связи выходного и входного сигналов.

### **ЗАМЕЧАНИЕ**

Размерности параметров указаны для случая цепи с одним входом и одним выходом. Функции преобразования могут обрабатывать описания цепей с несколькими выходными сигналами (с векторным выходом). В этом случае размерности параметров соответствующим образом изменяются.

## **Функция `tf2zp`**

Функция `tf2zp` преобразует наборы коэффициентов полиномов числителя и знаменателя функции передачи в векторы нулей и полюсов, рассчитывая также значение общего коэффициента усиления:

`[z, p, k] = tf2zp(b, a);`

Преобразование производится путем вычисления корней полиномов числителя и знаменателя функции передачи с помощью функции `roots`. Коэффициент усиления  $k$  рассчитывается как отношение  $b(1) / a(1)$ .

### **Функция `zp2tf`**

Функция `zp2tf` является обратной по отношению к функции `tf2zp`: она осуществляется преобразование коэффициента усиления, а также векторов нулей и полюсов функции передачи в коэффициенты полиномов ее числителя и знаменателя:

```
[b, a] = zp2tf(z, p, k);
```

Преобразование производится с помощью функции `poly`, предназначеннной для вычисления коэффициентов полинома по заданным его корням. В завершение вектор коэффициентов числителя умножается на  $k$ .

### **Функция `tf2ss`**

Функция `tf2ss` преобразует наборы коэффициентов полиномов числителя и знаменателя функции передачи в параметры представления цепи в пространстве состояний:

```
[A, B, C, D] = tf2ss(b, a);
```

Преобразование производится согласно формулам, приведенным ранее в разд. "Пространство состояний" этой главы.

### **Функция `ss2tf`**

Функция `ss2tf` является обратной по отношению к функции `tf2ss`: она преобразует параметры пространства состояний в коэффициенты полиномов функции передачи цепи:

```
[b, a] = ss2tf(A, B, C, D);
```

Преобразование производится согласно формуле (2.17), приведенной ранее в разд. "Пространство состояний" этой главы.

### **Функция `zp2ss`**

Функция `zp2ss` преобразует нули, полюсы и коэффициент усиления цепи в ее параметры пространства состояний:

```
[A, B, C, D] = zp2ss(z, p, k);
```

Преобразование производится путем последовательного вызова функций `zp2tf` и `tf2ss`.

### **Функция `ss2zp`**

Функция `ss2zp` является обратной по отношению к функции `zp2ss`, преобразуя параметры пространства состояний в нули, полюсы и коэффициент усиления цепи:

```
[z, p, k] = ss2zp(A, B, C, D);
```

Полюсы системы, равные собственным числам матрицы A, вычисляются с помощью функции `eig`. Нули являются конечными решениями обобщенной задачи нахождения собственных чисел и рассчитываются следующим образом:

```
z = eig([A B;C D], diag([ones(1,n) 0]));
```

## Функция `residue`

Идентификатор последней функции, позволяющей преобразовывать описания линейных цепей, выпадает из общего ряда. Это связано с тем, что данное преобразование не является специфическим для обработки сигналов — оно сводится к разложению дробно-рациональной функции на простейшие дроби и часто применяется в математике. По этой же причине данная функция относится не к пакету Signal Processing, а к базовой библиотеке MATLAB.

Преобразование функции передачи, заданной в виде коэффициентов полиномов числителя и знаменателя, в сумму простых дробей производится с помощью функции `residue`. Она же осуществляет и обратное преобразование; нужное направление преобразования выбирается в зависимости от числа входных параметров.

При двух входных параметрах производится разложение функции передачи на простые дроби:

```
[r, p, k] = residue(b, a)
```

Здесь b и a — коэффициенты полиномов числителя и знаменателя функции передачи соответственно. Выходные параметры — векторы-столбцы полюсов (p) и соответствующих им вычетов (r), а также строка коэффициентов целой части k.

Разложение производится согласно формулам (2.12) и (2.13), приведенным ранее в разд. "Полюсы и вычеты" этой главы, включая случай наличия кратных полюсов.

При использовании трех входных параметров функция `residue` производит преобразование вычетов, полюсов и коэффициентов целой части в коэффициенты числителя и знаменателя функции передачи, т. е. осуществляет суммирование простых дробей:

```
[b, a] = residue(r, p, k)
```

В качестве примера разложим на простые дроби функцию передачи вида

$$H(p) = \frac{p^2 - 2p + 2}{p^4 + 4p^3 + 7p^2 + 6p + 2}.$$

```
>> b = [1 -2 2];
>> a = [1 4 7 6 2];
>> [r, p, k] = residue(b, a)
r =
    2.0000 + 2.0000i
    2.0000 - 2.0000i
   -4.0000
    5.0000
```

```
p =
-1.0000 + 1.0000i
-1.0000 - 1.0000i
-1.0000
-1.0000
k =
[]
```

Результаты преобразования показывают, что данная система имеет двукратный полюс, равный  $-1$ , и пару комплексно-сопряженных полюсов, равных  $-1 \pm j$ . Поскольку степень полинома числителя меньше, чем степень полинома знаменателя, целая часть их отношения равна нулю, и в параметре `k` функция возвращает пустую матрицу. Рассчитанные значения вычетов позволяют записать представление функции передачи в виде суммы простых дробей (с учетом наличия кратного полюса):

$$H(p) = \frac{2+j2}{p+1-j} + \frac{2-j2}{p+1+j} - \frac{4}{p+1} + \frac{5}{(p+1)^2}.$$

Произведем с помощью функции `residue` обратное преобразование вычетов и полюсов в коэффициенты полиномов числителя и знаменателя функции передачи:

```
[b1, a1] = residue(r, p, k)
b1 =
-0.0000    1.0000   -2.0000    2.0000
a1 =
1.0000    4.0000    7.0000    6.0000    2.0000
```

Полученные значения совпадают с исходными. Знак "минус" при нулевом значении `b1(1)` обусловлен вычислительными погрешностями — после обратного преобразования получается не ровно нуль, а малое отрицательное число (примерно  $7 \cdot 10^{-15}$ ).

### **ВНИМАНИЕ!**

Вычислительная задача разложения дробно-рациональной функции на простые дроби плохо обусловлена. Это означает, что если функция передачи имеет полюсы, близкие к кратным, то малые вариации исходных данных могут приводить к большим погрешностям вычисления полюсов и вычетов. В таких случаях документация MATLAB рекомендует использовать описание системы в виде нулей и полюсов либо в пространстве состояний.

## **Расчет аналоговых фильтров-прототипов**

Одной из часто возникающих на практике задач является создание фильтров, пропускающих сигналы в определенной полосе частот и задерживающих остальные частоты. При этом различают:

- **фильтры нижних частот** (*ФНЧ*; английский термин — *low-pass filter*), пропускающие частоты, меньшие некоторой частоты среза  $\omega_0$ ;
- **фильтры верхних частот** (*ФВЧ*; английский термин — *high-pass filter*), пропускающие частоты, большие некоторой частоты среза  $\omega_0$ ;

- **полосовые фильтры** (*ПФ*; английский термин — *band-pass filter*), пропускающие частоты в некотором диапазоне  $\omega_1 \dots \omega_2$  (они могут также характеризоваться средней частотой  $\omega_0 = (\omega_1 + \omega_2)/2$  и шириной полосы пропускания  $\Delta\omega = \omega_2 - \omega_1$ );
- **режекторные фильтры** (другие возможные названия — *заграждающий фильтр*, *фильтр-пробка*, *полосно-задерживающий фильтр*; английский термин — *band-stop filter*), пропускающие на выход *все* частоты, *кроме* лежащих в некотором диапазоне  $\omega_1 \dots \omega_2$  (они тоже могут характеризоваться средней частотой  $\omega_0 = (\omega_1 + \omega_2)/2$  и шириной полосы задерживания  $\Delta\omega = \omega_2 - \omega_1$ ).

Идеальная форма АЧХ фильтров этих четырех типов показана на рис. 2.9.

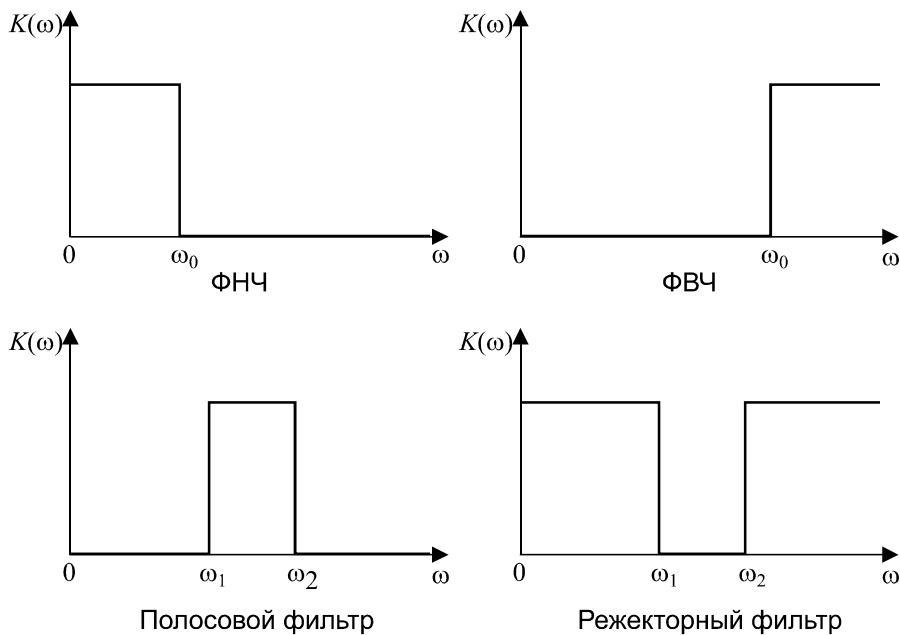


Рис. 2.9. АЧХ фильтров различного типа

Однако такая идеальная (прямоугольная) форма АЧХ не может быть физически реализована. Поэтому в теории аналоговых фильтров разработан ряд методов *аппроксимации* прямоугольных АЧХ. Функции, реализующие эти методы, рассмотрены далее.

Кроме того, рассчитав ФНЧ, можно несложными преобразованиями изменить его частоту среза, превратить его в ФВЧ, полосовой либо режекторный фильтр с заданными параметрами. Поэтому расчет аналогового фильтра начинается с расчета так называемого *фильтра-прототипа*, представляющего собой ФНЧ с частотой среза, равной 1 рад/с. Далее применяются функции преобразования частоты среза и преобразования типов фильтров, которые также рассмотрены далее.

При сравнении результатов использования фильтров различного типа в какой-либо системе обработки сигналов следует помнить о том, что частота среза для разных фильтров определяется по-разному. Подробная информация об этом приводится в следующих разделах, посвященных конкретным фильтрам. Немного забегая вперед, скажем, что для фильтра Баттервортча частота среза определяется по уровню  $1/\sqrt{2}$ , для фильтра Чебышева первого рода и эллиптического фильтра — по уровню пульсаций в полосе пропускания, для фильтра Чебышева второго рода — по уровню пульсаций в полосе задерживания. Наконец, для фильтра Бесселя само понятие частоты среза является весьма условным.

Помимо частоты среза, физически реализуемые фильтры характеризуются границами полос пропускания и задерживания:

- *полоса пропускания* фильтра (*passband*) — это диапазон частот, в котором вносимое фильтром ослабление сигнала *не превышает* заданной величины;
- *полоса задерживания* фильтра (*stopband*) — это диапазон частот, в котором вносимое фильтром ослабление сигнала *не меньше* заданной величины;
- диапазоны частот, не вошедшие в первые две категории, относятся к *переходным зонам* (*transition band*), в которых ослабление сигнала больше, чем в полосе пропускания, но меньше, чем в полосе задерживания.

Понятия частоты среза и границ полос пропускания и задерживания не следует смешивать. Под частотой среза в литературе, как правило, имеется в виду некая характерная частота, входящая в формулу, описывающую частотную характеристику фильтра. Границы же полос пропускания и задерживания определяются исходя из произвольно задаваемых уровней ослабления сигнала. Поэтому для одного и того же фильтра можно определить разные границы полос пропускания и задерживания, меняя определяющие их уровни ослабления сигнала.

Все функции MATLAB для расчета аналоговых прототипов возвращают векторы столбцы нулей и полюсов функции передачи, а также значение коэффициента усиления.

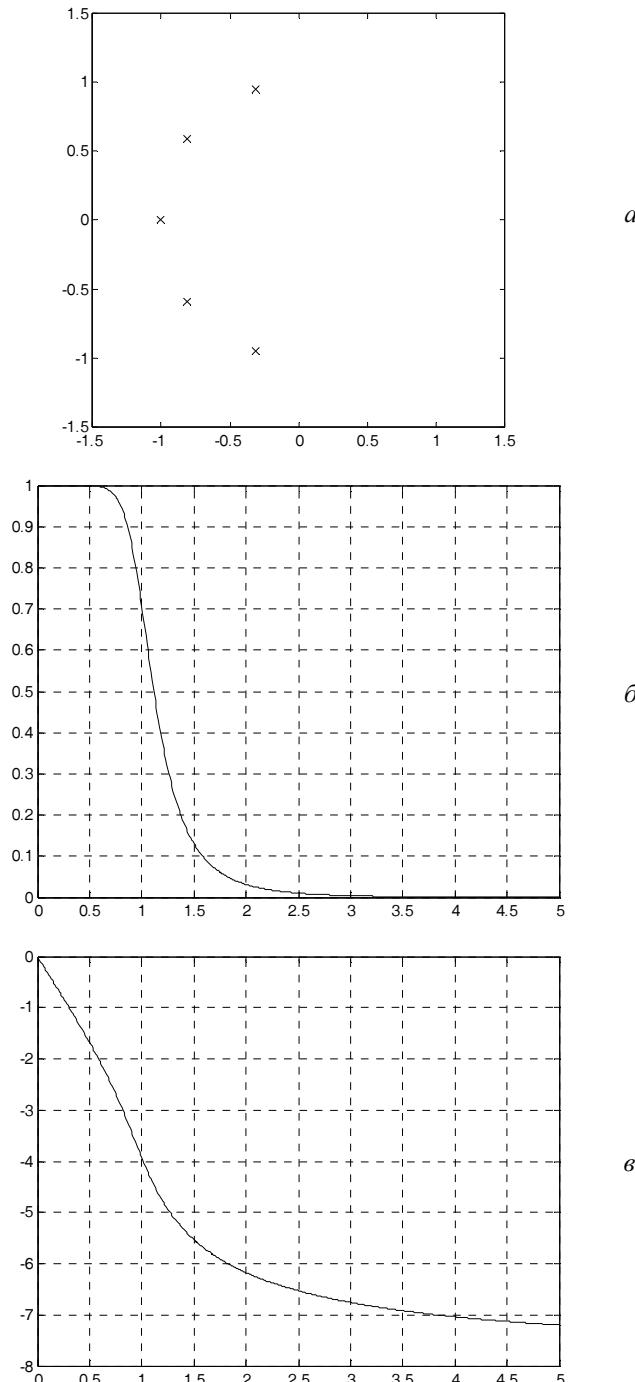
## Фильтр Баттервортча

Функция передачи фильтра-прототипа Баттервортча (Butterworth filter) не имеет нулей, а ее полюсы равномерно расположены на  $p$ -плоскости в левой половине окружности единичного радиуса (рис. 2.10, *a*).

Благодаря такому размещению полюсов формула для АЧХ фильтра Баттервортча оказывается очень простой:

$$K(\omega) = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_0}\right)^{2n}}},$$

где  $\omega_0$  — частота среза (для фильтра-прототипа она равна 1 рад/с),  $n$  — порядок фильтра.



**Рис. 2.10.** Характеристики фильтра Баттервортса 5-го порядка:  
*а* — расположение полюсов на комплексной плоскости, *б* — АЧХ, *в* — ФЧХ

Коэффициент передачи на нулевой частоте равен 1, а на частоте среза независимо от порядка фильтра составляет  $1/\sqrt{2} \approx 0,707 \approx -3$  дБ. При  $\omega \rightarrow \infty$  АЧХ стремится к нулю.

АЧХ фильтра Баттервортса (рис. 2.10, б) является *максимально плоской* при  $\omega = 0$  и  $\omega \rightarrow \infty$ . Это означает, что в данных точках равны нулю  $2n - 1$  производных АЧХ по частоте.

В целом АЧХ монотонно спадает от единицы до нуля при изменении частоты от нуля до бесконечности.

В MATLAB расчет аналогового фильтра-прототипа Баттервортса производится с помощью функции `buttap`:

```
[z, p, k] = buttap(n);
```

Входной целочисленный параметр  $n$  — это порядок фильтра.

На рис. 2.10 показано расположение полюсов фильтра Баттервортса 5-го порядка на комплексной плоскости, а также его АЧХ и ФЧХ:

```
>> [z, p, k] = buttap(5); % нули и полюсы прототипа
>> plot(p, 'x') % график расположения полюсов
>> axis equal
>> axis([-1.5 1.5 -1.5 1.5])
>> w = 0:0.01:5;
>> [b, a] = zp2tf(z, p, k); % коэффициенты функции передачи
>> h = freqs(b, a, w); % комплексный коэффициент передачи
>> figure
>> plot(w, abs(h)), grid % график АЧХ
>> figure
>> plot(w, unwrap(angle(h))), grid % график ФЧХ
```

## Фильтр Чебышева первого рода

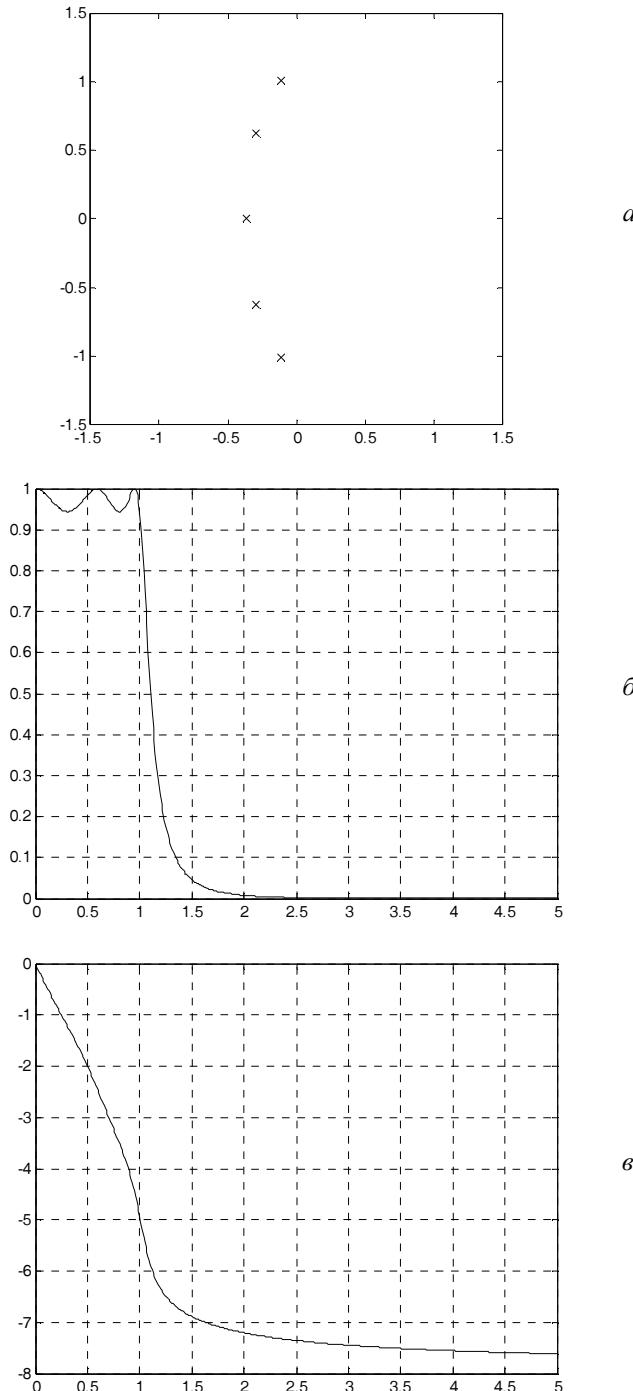
Функция передачи фильтра Чебышева первого рода (Chebyshev type I filter) также не имеет нулей, а ее полюсы расположены в левой половине эллипса на  $p$ -плоскости (рис. 2.11, а).

АЧХ фильтра Чебышева первого рода описывается следующим образом:

$$K(\omega) = \frac{1}{\sqrt{1 + \varepsilon^2 T_n^2(\omega / \omega_0)}}.$$

Здесь  $\omega_0$  — частота среза,  $T_n(x)$  — полином Чебышева  $n$ -го порядка,  $n$  — порядок фильтра,  $\varepsilon$  — параметр, определяющий величину пульсаций АЧХ в полосе пропускания.

Полином Чебышева  $T_n(x)$  при  $|x| \leq 1$  колеблется в диапазоне  $-1 \dots +1$ , а при  $|x| > 1$  неограниченно возрастает по абсолютной величине. Поэтому АЧХ фильтра Чебышева первого рода в полосе пропускания (при  $|\omega| \leq \omega_0$ ) колеблется между значе-



**Рис. 2.11.** Характеристики фильтра Чебышева первого рода 5-го порядка с уровнем пульсаций в полосе пропускания 0,5 дБ: *а* — расположение полюсов на комплексной плоскости, *б* — АЧХ, *в* — ФЧХ

ниями  $1/\sqrt{1+\varepsilon^2}$  и 1, а вне полосы пропускания (при  $|\omega| > \omega_0$ ) монотонно затухает до нуля (рис. 2.11, б).

Коэффициент передачи на нулевой частоте равен 1 при нечетном порядке фильтра и  $1/\sqrt{1+\varepsilon^2}$  — при четном. На частоте среза коэффициент передачи фильтра равен  $1/\sqrt{1+\varepsilon^2}$ , т. е. уровню пульсаций АЧХ в полосе пропускания. При  $\omega \rightarrow \infty$  АЧХ стремится к нулю.

По сравнению с фильтром Баттервортса того же порядка фильтр Чебышева обеспечивает более крутой спад АЧХ в области перехода от полосы пропускания к полосе задерживания.

Значение параметра  $\varepsilon$  и уровень пульсаций  $R_p$  (в децибелах) связаны следующим образом:

$$R_p = 20 \lg(\sqrt{1 + \varepsilon^2}) = 10 \lg(1 + \varepsilon^2) \text{ дБ}, \quad \varepsilon = \sqrt{10^{R_p/10} - 1}.$$

При  $\omega \rightarrow \infty$  АЧХ фильтра Чебышева первого рода является максимально плоской.

В MATLAB фильтр-прототип Чебышева первого рода рассчитывается с помощью функции `cheblap`:

```
[z, p, k] = cheblap(n, Rp)
```

Здесь  $n$  — порядок фильтра,  $R_p$  — уровень пульсаций в полосе пропускания (в децибелах).

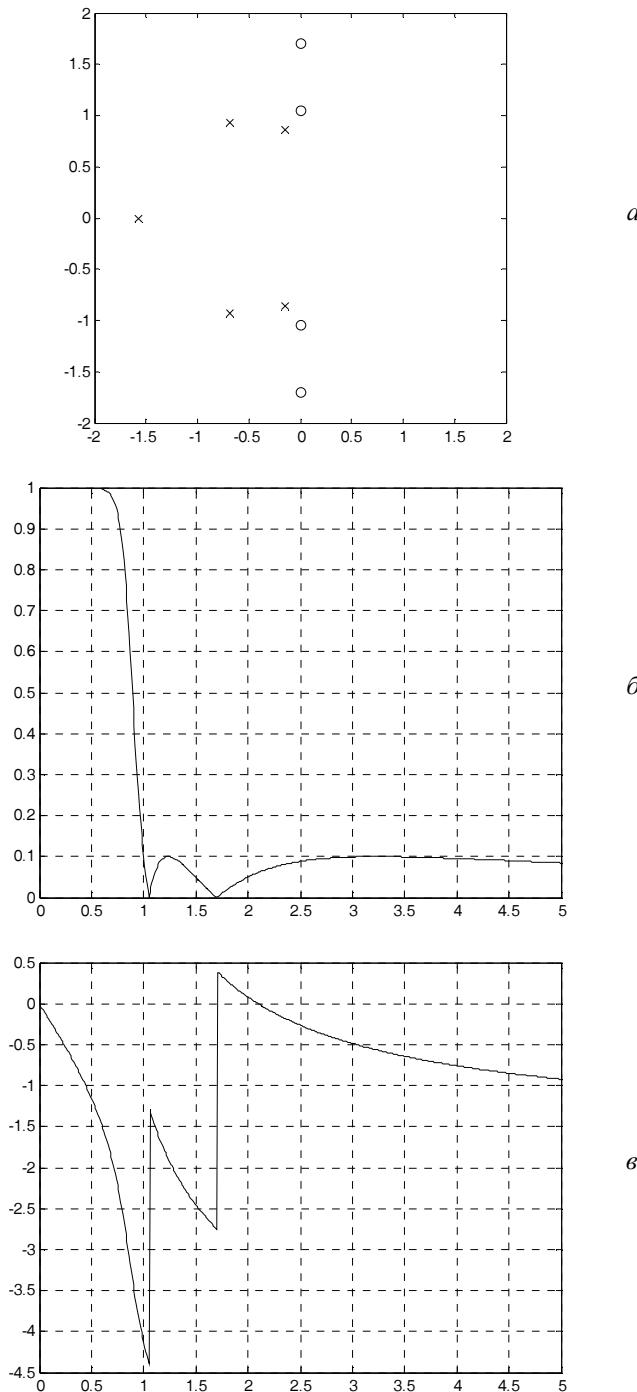
На рис. 2.11 показано расположение на комплексной плоскости полюсов фильтра Чебышева первого рода 5-го порядка с уровнем пульсаций 0,5 дБ, а также его АЧХ и ФЧХ:

```
>> [z, p, k] = cheblap(5, 0.5); % нули и полюсы прототипа
>> plot(p, 'x') % график расположения полюсов
>> axis equal
>> axis([-1.5 1.5 -1.5 1.5])
>> w = 0:0.01:5;
>> [b, a] = zp2tf(z, p, k); % коэффициенты функции передачи
>> h = freqs(b, a, w); % комплексный коэффициент передачи
>> figure
>> plot(w, abs(h)), grid % график АЧХ
>> figure
>> plot(w, unwrap(angle(h))), grid % график ФЧХ
```

## Фильтр Чебышева второго рода

Функция передачи фильтра Чебышева второго рода (Chebyshev type II filter), в отличие от предыдущих случаев, имеет и нули, и полюсы (см. рис. 2.12, а). Она связана с функцией передачи фильтра Чебышева первого рода следующим образом:

$$H_2(p) = 1 - H_1(1/p).$$



**Рис. 2.12.** Характеристики фильтра Чебышева второго рода 5-го порядка с уровнем пульсаций в полосе задерживания 20 дБ: *a* — расположение нулей и полюсов на комплексной плоскости, *б* — АЧХ, *в* — ФЧХ

Здесь  $H_1(p)$  и  $H_2(p)$  — функции передачи фильтров-прототипов Чебышева первого и второго рода соответственно.

Полюсы функции передачи фильтров-прототипов Чебышева первого и второго рода ( $p_{1i}$  и  $p_{2i}$  соответственно) связаны друг с другом соотношением  $p_{2i} = 1/p_{1i}$ .

По этой причине фильтры Чебышева второго рода называют еще *инверсными фильтрами Чебышева* (inverse Chebyshev filter).

АЧХ фильтра Чебышева второго рода описывается следующим образом:

$$K(\omega) = \frac{1}{\sqrt{1 + \frac{\varepsilon^2}{T_n^2(\omega_0 / \omega)}}}.$$

Здесь  $\omega_0$  — частота среза,  $T_n(x)$  — полином Чебышева  $n$ -го порядка,  $n$  — порядок фильтра,  $\varepsilon$  — параметр, определяющий величину пульсаций АЧХ в полосе задерживания.

В результате указанного ранее преобразования функции передачи АЧХ фильтра Чебышева второго рода ведет себя следующим образом: в полосе пропускания она монотонно затухает, а в полосе задерживания колеблется между нулем и значением  $1/\sqrt{1+\varepsilon^2}$  (см. рис. 2.12,  $\delta$ ).

### **ВНИМАНИЕ!**

Частотой среза фильтра Чебышева второго рода считается не конец полосы пропускания, а *начало полосы задерживания*.

Коэффициент передачи фильтра на нулевой частоте равен 1, на частоте среза — заданному уровню пульсаций в полосе задерживания. При  $\omega \rightarrow \infty$  коэффициент передачи равен нулю при нечетном порядке фильтра и уровню пульсаций — при четном.

Значение параметра  $\varepsilon$  и уровень пульсаций  $R_s$  (в децибелах) связаны следующим образом:

$$R_s = 20 \lg \left( \sqrt{1 + \varepsilon^2} \right) = 10 \lg \left( 1 + \varepsilon^2 \right) \text{ дБ}, \quad \varepsilon = \sqrt{10^{R_s/10} - 1}.$$

При  $\omega = 0$  АЧХ фильтра Чебышева второго рода является максимально плоской.

В MATLAB фильтр-прототип Чебышева второго рода рассчитывается с помощью функции `cheb2ap`:

```
[z, p, k] = cheb2ap(n, Rs)
```

Здесь  $n$  — порядок фильтра,  $Rs$  — уровень пульсаций в полосе задерживания (в децибелах).

На рис. 2.12 показано расположение на комплексной плоскости полюсов фильтра Чебышева второго рода 5-го порядка с уровнем пульсаций 20 дБ, а также его АЧХ и ФЧХ:

```

>> [z, p, k] = cheb2ap(5, 20);      % нули и полюсы прототипа
>> plot(p, 'x')                  % график расположения полюсов
>> hold on
>> plot(z, 'o')                  % график расположения нулей
>> hold off
>> axis equal
>> axis([-2 2 -2 2])
>> w = 0:0.01:5;
>> [b, a] = zp2tf(z, p, k);      % коэффициенты функции передачи
>> h = freqs(b, a, w);          % комплексный коэффициент передачи
>> figure
>> plot(w, abs(h)), grid        % график АЧХ
>> figure
>> plot(w, unwrap(angle(h))), grid % график ФЧХ

```

## Эллиптический фильтр

Эллиптические фильтры (фильтры Золотарева — Кауэра; английские термины — *elliptic filter*, *Cauer filter*) в некотором смысле объединяют в себе свойства фильтров Чебышева первого и второго рода, поскольку АЧХ эллиптического фильтра имеет пульсации заданной величины как в полосе пропускания, так и в полосе задерживания (см. рис. 2.13, *б*). За счет этого удается обеспечить максимально возможную (при фиксированном порядке фильтра) крутизну ската АЧХ, т. е. переходной зоны между полосами пропускания и задерживания.

Функция передачи эллиптического фильтра имеет как полюсы, так и нули. Нули, как и в случае фильтра Чебышева второго рода, являются чисто мнимыми и образуют комплексно-сопряженные пары (см. рис. 2.13, *а*). Количество нулей функции передачи равно максимальному четному числу, не превосходящему порядка фильтра.

АЧХ эллиптического фильтра описывается следующей формулой:

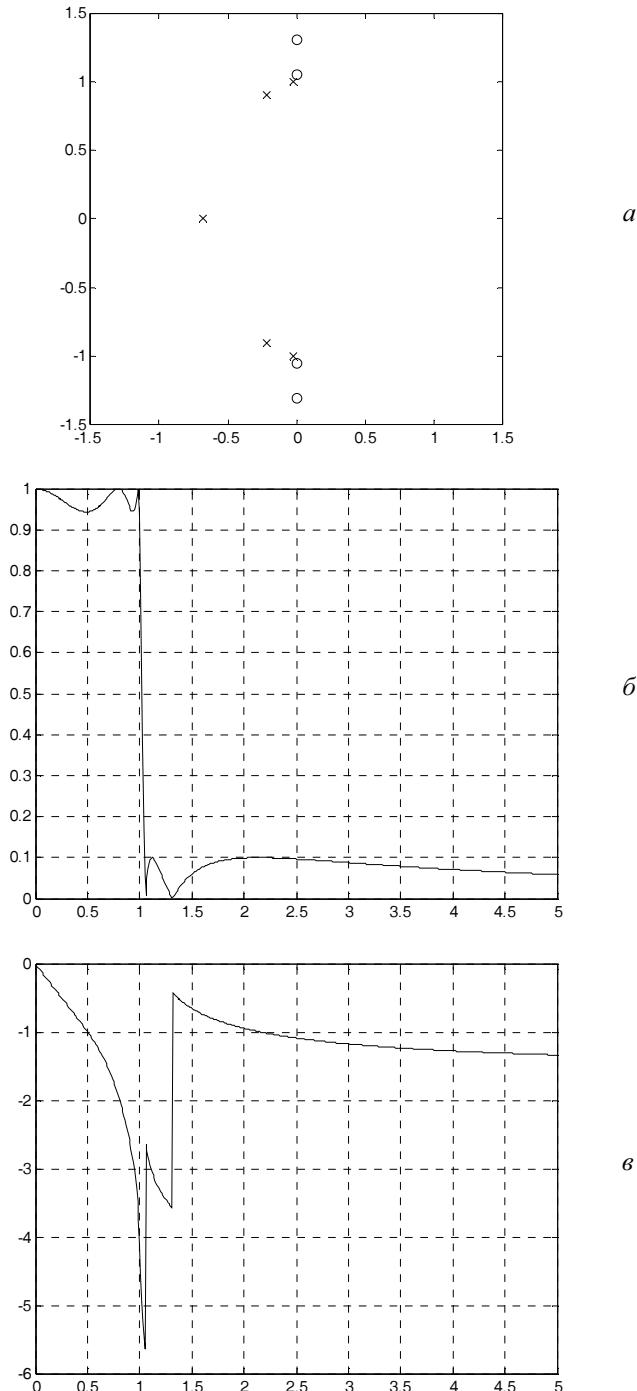
$$K(\omega) = \frac{1}{\sqrt{1 + \varepsilon^2 R_n^2(\omega / \omega_0, L)}}.$$

Здесь  $\omega_0$  — частота среза,  $n$  — порядок фильтра,  $R_n(x)$  — рациональная эллиптическая функция  $n$ -го порядка,  $\varepsilon$  и  $L$  — параметры, определяющие величину пульсаций в полосах пропускания и задерживания.

В MATLAB эллиптический фильтр-прототип рассчитывается с помощью функции *ellipap*:

```
[z, p, k] = ellipap(n, Rp, Rs)
```

Здесь  $n$  — порядок фильтра,  $Rp$  — уровень пульсаций в полосе пропускания,  $Rs$  — уровень пульсаций в полосе задерживания. Уровни пульсаций указываются в децибелах.



**Рис. 2.13.** Характеристики эллиптического фильтра 5-го порядка с уровнем пульсаций в полосе пропускания 0,5 дБ и в полосе задерживания 20 дБ: *а* — расположение нулей и полюсов на комплексной плоскости, *б* — АЧХ, *в* — ФЧХ

На рис. 2.13 показано расположение на комплексной плоскости нулей и полюсов эллиптического фильтра 5-го порядка с уровнем пульсаций 0,5 дБ в полосе пропускания и 20 дБ в полосе задерживания, а также его АЧХ и ФЧХ:

```
>> [z, p, k] = ellipap(5, 0.5, 20); % нули и полюсы прототипа
>> plot(p, 'x') % график расположения полюсов
>> hold on
>> plot(z, 'o') % график расположения нулей
>> hold off
>> axis equal
>> axis([-1.5 1.5 -1.5 1.5])
>> w = 0:0.01:5;
>> [b, a] = zp2tf(z, p, k); % коэффициенты функции передачи
>> h = freqs(b, a, w); % комплексный коэффициент передачи
>> figure
>> plot(w, abs(h)), grid % график АЧХ
>> figure
>> plot(w, unwrap(angle(h))), grid % график ФЧХ
```

## Фильтр Бесселя

В отличие от фильтров предыдущих типов, фильтры Бесселя (Bessel filter) не аппроксимируют прямоугольную АЧХ — их АЧХ (см. рис. 2.14, б) по форме близка к гауссовой кривой (точнее, стремится к ней с ростом порядка фильтра). Практическая ценность фильтров Бесселя определяется тем, что для них зависимость группового времени задержки от частоты является максимально гладкой в точке  $\omega = 0$  и групповая задержка очень мало меняется в полосе пропускания.

Функция передачи фильтра Бесселя имеет только полюсы, лежащие на окружности с центром в положительной области вещественной оси (см. рис. 2.14, а). Сама функция передачи имеет следующий вид:

$$H(p) = \frac{d_0}{\sum_{k=0}^n d_k p^k}.$$

Коэффициенты полинома знаменателя рассчитываются по следующей формуле:

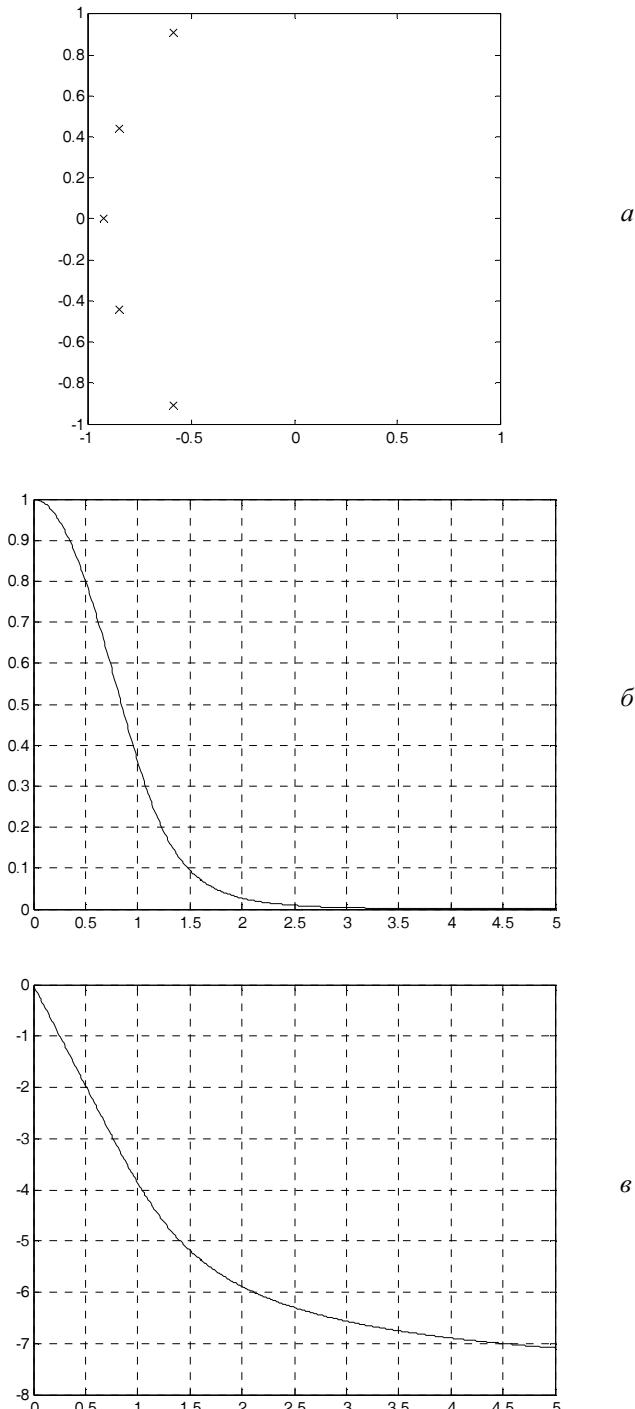
$$d_k = \frac{(2n-k)!}{2^{n-k} k!(n-k)!}.$$

В MATLAB фильтр-прототип Бесселя рассчитывается с помощью функции `besselap`:

```
[z, p, k] = besselap(n)
```

Здесь  $n$  — порядок фильтра.

На рис. 2.14 показано расположение на комплексной плоскости нулей и полюсов фильтра Бесселя 5-го порядка, а также его АЧХ и ФЧХ.



**Рис. 2.14.** Характеристики фильтра Бесселя 5-го порядка: *a* — расположение полюсов на комплексной плоскости, *б* — АЧХ, *в* — ФЧХ

```

>> [z, p, k] = besselap(5);           % нули и полюсы прототипа
>> plot(p, 'x')                   % график расположения нулей
>> axis equal
>> axis([-1 1 -1 1])
>> w = 0:0.01:5;                  % коэффициенты функции передачи
>> [b, a] = zp2tf(z, p, k);       % комплексный коэффициент передачи
>> h = freqs(b, a, w);           % комплексный коэффициент передачи
>> figure
>> plot(w, abs(h)), grid         % график АЧХ
>> figure
>> plot(w, unwrap(angle(h))), grid % график ФЧХ

```

## Частотные преобразования фильтров

Под частотными преобразованиями фильтров понимаются такие преобразования передаточной функции, при которых форма частотных характеристик меняется только в "горизонтальном" направлении, так что:

$$\dot{K}_2(\omega) = \dot{K}_1(f_\omega(\omega)),$$

где  $\dot{K}_1(\omega)$  — комплексный коэффициент передачи исходного фильтра,  $\dot{K}_2(\omega)$  — комплексный коэффициент передачи преобразованного фильтра,  $f_\omega(\omega)$  — функция преобразования частоты.

Такие преобразования осуществляются путем замены переменной  $p$  в функции передачи исходного фильтра на некоторую функцию  $f_p(p)$ :

$$H_2(p) = H_1(f_p(p)).$$

Эта функция должна удовлетворять двум требованиям:

- т. к. функция передачи после подстановки должна сохранить дробно-рациональную форму, подстановочная функция  $f_p(p)$  также должна быть дробно-рациональной;
- поскольку комплексный коэффициент передачи получается из функции передачи при подстановке  $p = j\omega$ , подстановочная функция  $f_p(p)$  при чисто мнимом аргументе должна давать чисто мнимый результат. Функция преобразования частоты в таком случае определяется как

$$f_\omega(\omega) = \frac{1}{j} f_p(j\omega).$$

При дробно-рациональном характере функции  $f_p(p)$  выполнение данного условия обеспечивается в том и только том случае, если один из полиномов (числителя и знаменателя) содержит только четные, а другой — только нечетные степени переменной  $p$ .

Для чаще всего необходимых на практике преобразований оказывается достаточно дробно-рациональных функций  $f_p(p)$ , содержащих в числите и знаменателе полиномы не более чем второй степени. Далее эти случаи будут рассмотрены подробнее.

В принципе, преобразования рассматриваемого типа могут применяться к произвольным фильтрам, но чаще всего они используются для превращения фильтра-прототипа в фильтр заданного вида с требуемыми частотами среза. Для этого используются приведенные ниже четыре функции MATLAB. Принцип составления их имен следующий: сначала идет сокращение  $1p$ , означающее, что исходным фильтром является ФНЧ (low-pass), потом следует символ преобразования  $2$  и в конце стоит обозначение типа результирующего фильтра:

- $1p2lp$  — изменение частоты среза ФНЧ (low-pass);
- $1p2hp$  — преобразование ФНЧ в ФВЧ (high-pass);
- $1p2bp$  — преобразование ФНЧ в полосовой фильтр (band-pass);
- $1p2bs$  — преобразование ФНЧ в режекторный фильтр (band-stop).

Все эти функции могут преобразовывать фильтры, заданные двумя способами — в виде коэффициентов полиномов числителя и знаменателя функции передачи либо в пространстве состояний. Различаются эти два варианта по числу входных и выходных параметров.

## Изменение частоты среза ФНЧ (изменение масштаба частотной оси)

Изменение частоты среза ФНЧ-прототипа сводится к простому масштабированию частотной оси и выполняется путем следующей замены переменной  $p$  в выражении для функции передачи:

$$f_p(p) = \frac{p}{\omega_0},$$

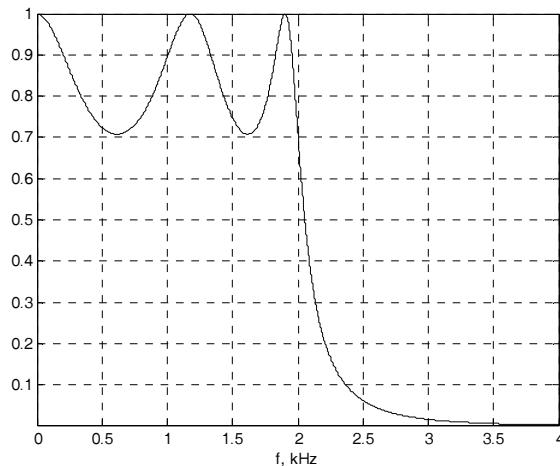
где  $\omega_0$  — требуемая частота среза ФНЧ. Функция трансформации частоты в данном случае также является линейной:

$$f_\omega(\omega) = \frac{\omega}{\omega_0}.$$

Данная подстановка приводит к линейному изменению масштаба частотной оси (для ФНЧ это означает изменение частоты среза в  $\omega_0$  раз, что нам в данном случае и требуется).

В MATLAB такое преобразование производится функцией  $1p2lp$ :

```
[b1, a1] = lp2lp(b, a, w0)
[A1, B1, C1, D1] = lp2lp(A, B, C, D, w0)
```



**Рис. 2.15.** Результат масштабирования частотной оси

Входными параметрами функции являются описание фильтра (в виде коэффициентов полиномов числителя и знаменателя функции передачи —  $b$ ,  $a$ , или в пространстве состояний —  $A$ ,  $B$ ,  $C$ ,  $D$ ) и требуемая частота среза  $\omega_0$ .

Возвращаемый результат — пересчитанные параметры фильтра.

Рассчитаем в качестве примера ФНЧ Чебышева первого рода 5-го порядка с частотой среза 2 кГц и уровнем пульсаций в полосе пропускания 3 дБ:

```
>> [z, p, k] = cheb1ap(5, 3);      % ФНЧ-прототип
>> [b, a] = zp2tf(z, p, k);       % функция передачи
>> f0 = 2e3;                      % частота среза
>> [b, a] = lp2lp(b, a, 2*pi*f0); % ФНЧ с нужной частотой среза
>> f = 0:1:4e3;                   % вектор частот для расчета
>> h = freqs(b, a, 2*pi*f);      % частотная характеристика
>> plot(f/1000, abs(h)), grid    % график АЧХ
>> axis tight
>> xlabel('f, kHz')
```

Построенный график АЧХ приведен на рис. 2.15.

### Преобразование ФНЧ в ФВЧ (инверсия частотной оси)

Преобразование ФНЧ-прототипа в ФВЧ требует инверсии частотной оси и выполняется путем следующей замены переменной  $p$  в выражении для функции передачи:

$$f_p(p) = \frac{\omega_0}{p},$$

где  $\omega_0$  — требуемая частота среза ФВЧ. Функция трансформации частоты в данном случае также является обратной пропорциональностью:

$$f_\omega(\omega) = -\frac{\omega_0}{\omega}.$$

Данная подстановка приводит к тому, что точки АЧХ, соответствующие нулевой и бесконечной частотам, меняются местами (для ФНЧ это означает превращение в ФВЧ, что нам в данном случае и требуется).

В MATLAB такое преобразование производится функцией `lp2hp`:

```
[b1, a1] = lp2hp(b, a, w0)
[A1, B1, C1, D1] = lp2hp(A, B, C, D, w0)
```

Входными параметрами функции являются описание фильтра (в виде коэффициентов полиномов числителя и знаменателя функции передачи —  $b$ ,  $a$ , или в пространстве состояний —  $A$ ,  $B$ ,  $C$ ,  $D$ ) и требуемая частота среза  $w_0$ .

Возвращаемый результат — пересчитанные параметры фильтра.

Рассчитаем в качестве примера ФВЧ Чебышева первого рода 5-го порядка с частотой среза 2 кГц и уровнем пульсаций в полосе пропускания 3 дБ:

```
>> [z, p, k] = cheb1ap(5, 3);      % ФНЧ-прототип
>> [b, a] = zp2tf(z, p, k);       % функция передачи
>> f0 = 2e3;                      % частота среза
>> [b, a] = lp2hp(b, a, 2*pi*f0); % ФВЧ с нужной частотой среза
>> f = 0:1:4e3;                   % вектор частот для расчета
>> h = freqs(b, a, 2*pi*f);       % частотная характеристика
>> plot(f/1000, abs(h)), grid     % график АЧХ
>> axis tight
>> xlabel('f, kHz')
```

Построенный график АЧХ приведен на рис. 2.16.

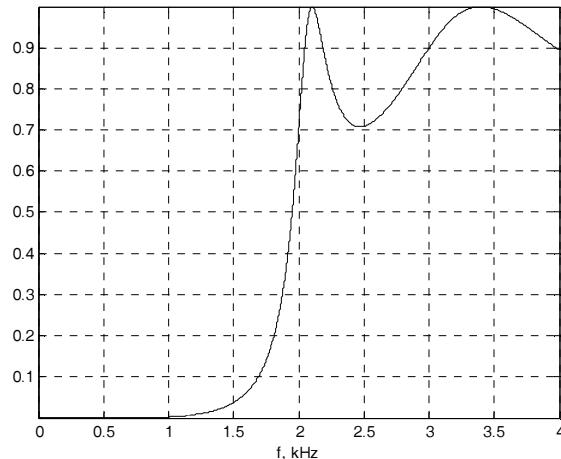


Рис. 2.16. Результат инверсии частотной оси

## Преобразование ФНЧ в полосовой фильтр

Для такого преобразования необходима более сложная трансформации частотной оси, чем в предыдущих случаях. Чтобы превратить ФНЧ в полосовой фильтр, настроенный на частоту  $\omega_0$ , нужно, чтобы функция  $f_\omega(\omega)$  отображала частоты следующим образом:

$$f_\omega(0) = \infty \text{ или } -\infty,$$

$$f_\omega(\pm\omega_0) = 0,$$

$$f_\omega(\pm\infty) = \infty \text{ или } -\infty.$$

Простейшая функция, удовлетворяющая этим и перечисленным ранее требованиям, имеет вид

$$f_\omega(\omega) = a \frac{\omega^2 - \omega_0^2}{\omega}. \quad (2.27)$$

Входящий в (2.27) параметр  $a$  регулирует крутизну ветвей функции преобразования, что в конечном счете определяет полосу пропускания фильтра. Выявим связь параметра  $a$  с полосой пропускания. Если преобразованию подвергается фильтр-прототип, его частота среза равна единице. Поэтому на границах полосы пропускания полосового фильтра функция  $f_\omega(\omega)$  должна давать единичное значение. Решение уравнения  $f_\omega(\omega) = 1$  дает

$$\omega = \frac{1 \pm \sqrt{1 + 4a^2\omega_0^2}}{2a}.$$

Однако не следует забывать, что мы рассматриваем также и отрицательные частоты, в области которых частота среза фильтра-прототипа равна минус единице. Таким образом, получаем уравнение  $f_\omega(\omega) = -1$ , решение которого дает

$$\omega = \frac{-1 \pm \sqrt{1 + 4a^2\omega_0^2}}{2a}.$$

Из приведенных формул очевидно, что пара решений каждого из двух уравнений имеет разные знаки. В результате получаем две пары граничных частот полосы пропускания:

□ в области положительных частот:  $\omega_1 = \frac{-1 + \sqrt{1 + 4a^2\omega_0^2}}{2a}$  и  $\omega_2 = \frac{1 + \sqrt{1 + 4a^2\omega_0^2}}{2a}$ ;

□ в области отрицательных частот:  $\omega_3 = \frac{-1 - \sqrt{1 + 4a^2\omega_0^2}}{2a}$  и  $\omega_4 = \frac{1 - \sqrt{1 + 4a^2\omega_0^2}}{2a}$ .

Отсюда видно, что ширина полосы пропускания составляет  $1/a$ ; кроме того, частота  $\omega_0$  равна *среднему геометрическому* краев полосы пропускания:

$$\omega_0 = \sqrt{\omega_1\omega_2} = \sqrt{\omega_3\omega_4}.$$

Таким образом, окончательно формулу преобразования частоты в данном случае можно записать так:

$$f_{\omega}(\omega) = \frac{\omega^2 - \omega_0^2}{\Delta\omega \omega}, \quad (2.28)$$

где  $\Delta\omega$  — требуемая ширина полосы пропускания фильтра. Получил распространение также еще один вариант формулы (2.28), в котором используется безразмерный параметр добротности ( $Q$ ), обычно применяемый для измерения качества колебательных систем. В частности, *добротность* одиночного колебательного контура равна отношению его резонансной частоты к полосе пропускания по уровню  $-3$  дБ. Если мы используем аналогичное определение ( $Q = \omega_0 / \Delta\omega$ ), формула (2.28) примет следующий вид:

$$f_{\omega}(\omega) = Q \frac{\omega^2 - \omega_0^2}{\omega_0 \omega}. \quad (2.29)$$

Из (2.28) и (2.29) легко получить соответствующие варианты функции подстановки для переменной  $s$ :

$$f_p(p) = \frac{p^2 + \omega_0^2}{\Delta\omega p} = Q \frac{p^2 + \omega_0^2}{\omega_0 p}. \quad (2.30)$$

Наконец, для наглядности можно перейти в формулах (2.29) и (2.30) к нормированной частоте, разделив числитель и знаменатель на  $\omega_0^2$ :

$$f_{\omega}(\omega) = Q \frac{(\omega/\omega_0)^2 - 1}{\omega/\omega_0}, \quad f_p(p) = Q \frac{(p/\omega_0)^2 + 1}{p/\omega_0}. \quad (2.31)$$

В MATLAB такое преобразование выполняется функцией `lp2bp`:

```
[b1, a1] = lp2bp(b, a, w0, Bw)
[A1, B1, C1, D1] = lp2bp(A, B, C, D, w0, Bw)
```

Входными параметрами функции являются описание фильтра (в виде коэффициентов полиномов числителя и знаменателя функции передачи —  $b$ ,  $a$ , или в пространстве состояний —  $A$ ,  $B$ ,  $C$ ,  $D$ ), средняя частота  $w0$  и ширина  $Bw$  полосы пропускания фильтра (в радианах в секунду).

### **ВНИМАНИЕ!**

Обратите внимание на то, что средняя частота полосы пропускания — это среднее *геометрическое*, а не среднее арифметическое частот среза:  $w0 = \sqrt{w1 * w2}$ . Полоса пропускания рассчитывается без особенностей:  $Bw = w2 - w1$ .

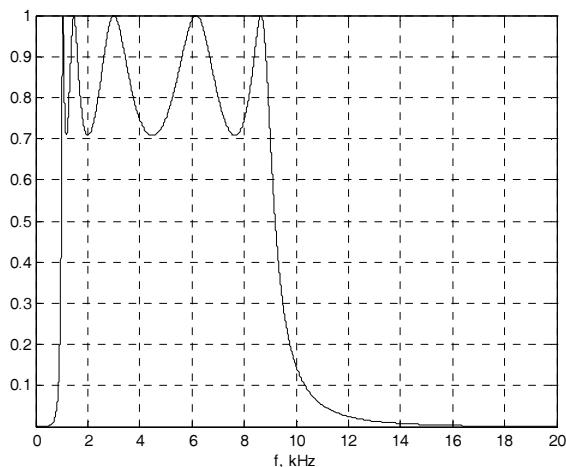
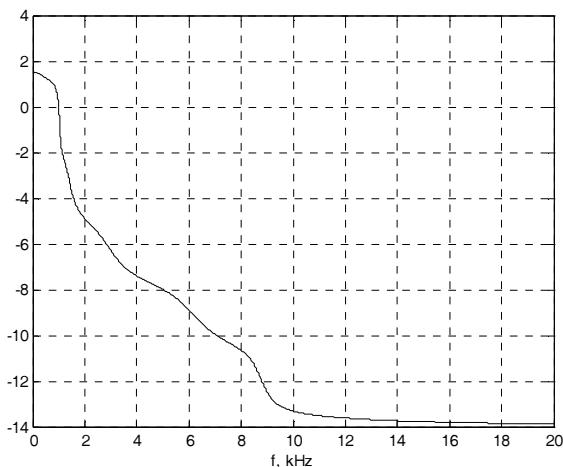
Возвращаемый результат — пересчитанные параметры фильтра.

Рассчитаем в качестве примера полосовой фильтр Чебышева первого рода 5-го порядка с полосой пропускания от 1 до 9 кГц и уровнем пульсаций в полосе пропускания 3 дБ:

```

>> [z, p, k] = cheb1ap(5, 3);      % ФНЧ-прототип
>> [b, a] = zp2tf(z, p, k);       % функция передачи
>> f1 = 1e3;                      % нижняя частота среза
>> f2 = 9e3;                      % верхняя частота среза
>> w0 = 2 * pi * sqrt(f1 * f2);    % средняя частота
>> Bw = 2 * pi * (f2 - f1);        % полоса пропускания
>> [b, a] = lp2bp(b, a, w0, Bw);   % полосовой фильтр
>> f = 0:1:20e3;                  % вектор частот для расчета
>> h = freqs(b, a, 2*pi*f);       % частотная характеристика
>> plot(f/1000, abs(h)), grid     % график АЧХ
>> xlabel('f, kHz')
>> axis tight
>> figure
>> plot(f/1000, unwrap(angle(h))), grid % график ФЧХ
>> xlabel('f, kHz')

```

*a**б*

**Рис. 2.17.** При большом отношении граничных частот полосы пропускания полосового фильтра отчетливо наблюдается асимметрия частотных характеристик

Далее приведены построенные графики АЧХ (рис. 2.17, а) и ФЧХ (рис. 2.17, б). Обратите внимание на то, что характеристики фильтра несимметричны — они сжаты слева и растянуты справа. Если проанализировать формулу замены переменной, использованную для преобразования ФНЧ в полосовой фильтр, окажется, что частоты, на которых коэффициент передачи имеет одинаковые значения, связаны соотношением  $\omega_1\omega_2 = \omega_0^2$ . Поэтому графики станут симметричными, если использовать логарифмический масштаб по оси частот.

Асимметрия частотных характеристик проявляется тем сильнее, чем больше отношение граничных частот полосы пропускания фильтра. Именно по этой причине в рассмотренном примере специально использованы столь сильно (в 9 раз) различающиеся частоты среза.

## Преобразование ФНЧ в режекторный фильтр

Чтобы превратить ФНЧ в режекторный фильтр, настроенный на частоту  $\omega_0$ , необходимо, чтобы функция  $f_\omega(\omega)$  отображала частоты следующим образом:

$$f_\omega(0) = 0,$$

$$f_\omega(\pm\omega_0) = \infty \text{ или } -\infty,$$

$$f_\omega(\pm\infty) = 0.$$

Рассуждения, аналогичные приведенным в предыдущем разделе, дают следующий результат:

$$f_\omega(\omega) = \frac{\Delta\omega \omega}{\omega^2 - \omega_0^2} = \frac{\omega_0 \omega}{Q(\omega^2 - \omega_0^2)} = \frac{\omega/\omega_0}{Q((\omega/\omega_0)^2 - 1)}, \quad (2.32)$$

$$f_p(p) = \frac{\Delta\omega p}{p^2 + \omega_0^2} = \frac{\omega_0 p}{Q(p^2 + \omega_0^2)} = \frac{p/\omega_0}{Q((p/\omega_0)^2 + 1)}. \quad (2.33)$$

В MATLAB такое преобразование выполняется функцией `lp2bs`:

```
[b1, a1] = lp2bs(b, a, w0, Bw)
[A1, B1, C1, D1] = lp2bs(A, B, C, D, w0, Bw)
```

Входными параметрами функции являются описание фильтра (в виде коэффициентов полиномов числителя и знаменателя функции передачи —  $b$ ,  $a$ , или в пространстве состояний —  $A$ ,  $B$ ,  $C$ ,  $D$ ), средняя частота  $w_0$  и ширина  $Bw$  полосы задерживания фильтра (в радианах в секунду).

Возвращаемый результат — пересчитанные параметры фильтра.

Рассчитаем в качестве примера режекторный фильтр Чебышева первого рода 5-го порядка с полосой задерживания от 8 до 12 кГц и уровнем пульсаций в полосе пропускания 3 дБ:

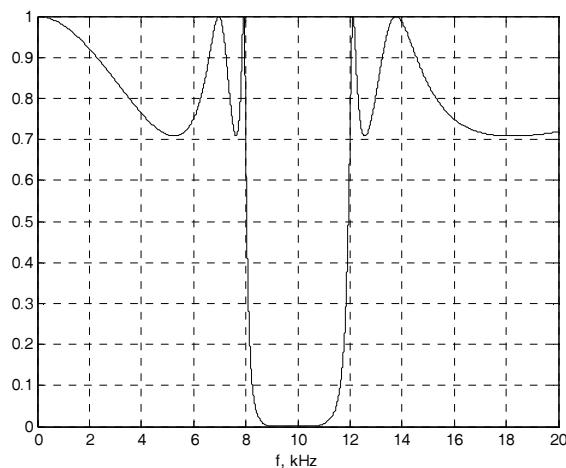
```
>> [z, p, k] = cheblap(5, 3);      % ФНЧ-прототип
>> [b, a] = zp2tf(z, p, k);       % функция передачи
```

```

>> f1 = 8e3; % нижняя частота среза
>> f2 = 12e3; % верхняя частота среза
>> w0 = 2 * pi * sqrt(f1 * f2); % средняя частота
>> Bw = 2 * pi * (f2 - f1); % полоса пропускания
>> [b, a] = lp2bs(b, a, w0, Bw); % режекторный фильтр
>> f = 0:1:20e3; % вектор частот для расчета
>> h = freqs(b, a, 2*pi*f); % частотная характеристика
>> plot(f/1000, abs(h)), grid % график АЧХ
>> ylim([0 1])
>> xlabel('f, kHz')

```

Построенный график АЧХ приведен на рис. 2.18.



**Рис. 2.18.** Результат преобразования ФНЧ в режекторный фильтр

#### ЗАМЕЧАНИЕ

Все сказанное ранее (применительно к полосовым фильтрам) о связи средней частоты с частотами среза и об асимметрии характеристик справедливо и для режекторных фильтров.

## Расчет аналоговых фильтров

Как уже говорилось, для расчета аналогового фильтра необходимо выполнить две основные операции: рассчитать ФНЧ-прототип и преобразовать его к нужному типу фильтра с заданными частотами среза. Требуемая последовательность действий оформлена в виде следующих функций MATLAB:

- `butter(n, w0, type, 's')` — расчет фильтров Баттервортта;
- `cheby1(n, Rp, w0, type, 's')` — расчет фильтров Чебышева первого рода;
- `cheby2(n, Rs, w0, type, 's')` — расчет фильтров Чебышева второго рода;

- `ellip(n, Rp, Rs, w0, type, 's')` — расчет эллиптических фильтров;
- `besself(n, w0, type)` — расчет фильтров Бесселя.

Параметры всех функций задаются одинаково, поэтому обсуждать функции по отдельности не будем.

Перечисленные функции, за исключением функции `besself`, позволяют рассчитывать как аналоговые, так и дискретные фильтры. Признаком аналогового расчета служит строка '`s`', использованная в качестве последнего входного параметра. Об использовании этих функций для расчета дискретных фильтров речь пойдет в [главе 4](#).

Параметры `n`, `Rp`, `Rs` (их состав зависит от типа фильтра) — это параметры фильтра-прототипа: `n` — порядок фильтра, `Rp` — уровень пульсаций в полосе пропускания (в децибелах), `Rs` — уровень пульсаций в полосе задерживания (в децибелах). Более подробное описание этих параметров было приведено ранее, в разделах, посвященных фильтрам-прототипам.

Параметры `w0` и `type` используются совместно для задания типа фильтра и значений его частот среза (в радианах в секунду):

- ФНЧ: `w0` — скаляр, параметр `type` отсутствует;
- ФВЧ: `w0` — скаляр, `type='high'`;
- полосовой фильтр: `w0` — двухэлементный вектор частот среза `[w1 w2]`, параметр `type` отсутствует;
- режекторный фильтр: `w0` — двухэлементный вектор частот среза `[w1 w2]`, `type='stop'`.

### **ВНИМАНИЕ!**

Следует подчеркнуть, что в данных функциях при разработке полосовых и режекторных фильтров в качестве параметров задаются именно *частоты среза*, а не средняя частота и полоса пропускания, как в функциях преобразования фильтров-прототипов.

В зависимости от того, сколько выходных параметров указано при вызове, функции могут возвращать результаты расчета в виде коэффициентов полиномов числителя и знаменателя функции передачи (два выходных параметра), нулей и полюсов (три выходных параметра) либо параметров пространства состояний (четыре выходных параметра):

```
[b, a] = ...  
[z, p, k] = ...  
[A, B, C, D] = ...
```

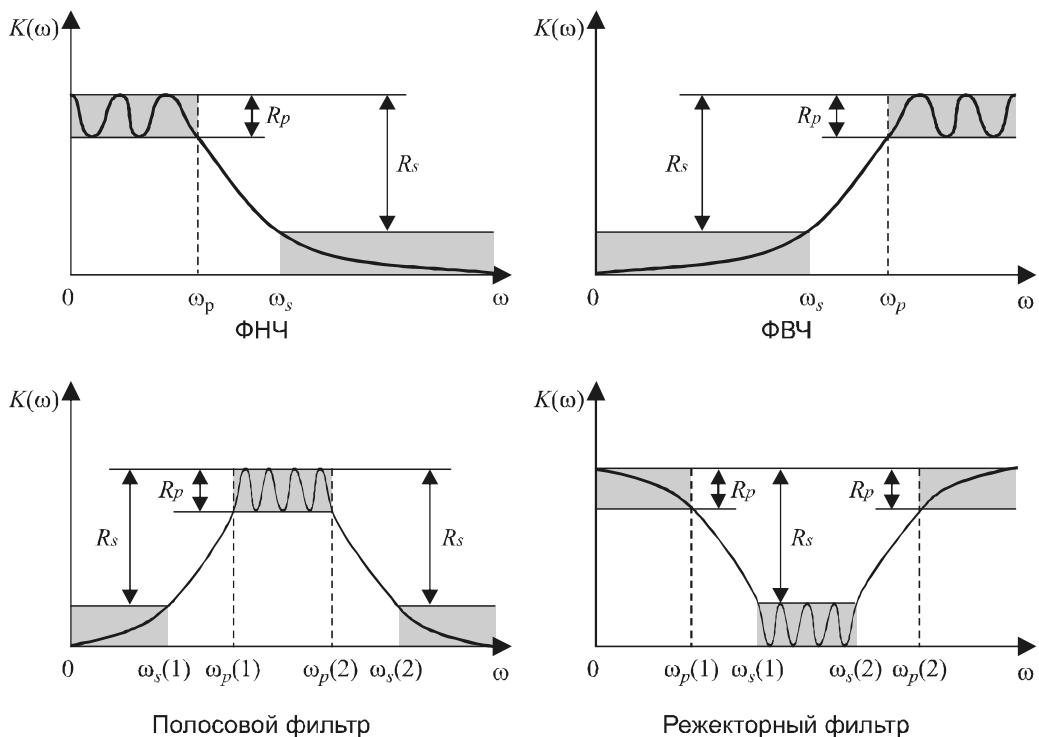
С учетом всего сказанного перечислим действия, выполняемые функциями расчета аналоговых фильтров:

1. Производится расчет фильтра-прототипа с заданными параметрами АЧХ.
2. Полученные нули и полюсы преобразуются в параметры пространства состояний.

3. Производится преобразование фильтра-прототипа к требуемому типу с заданными частотами среза.
4. Выполняется преобразование описания фильтра к заданному при вызове виду.

## Выбор порядка фильтра

Рассмотренные ранее функции расчета фильтров требуют задания в качестве входных параметров порядка фильтра и его частоты среза. При этом понятие частоты среза для фильтров разных типов определяется по-разному. Однако исходными данными при разработке фильтров, как правило, являются другие параметры: частотные границы полос пропускания ( $\omega_p$ ) и задерживания ( $\omega_s$ ), а также допустимая неравномерность АЧХ в полосе пропускания ( $R_p$ ) и минимально необходимое затухание в полосе задерживания ( $R_s$ ) (рис. 2.19). Серые области на рисунке демонстрируют допуски, в которые должна укладываться АЧХ фильтра в полосах пропускания и задерживания.



**Рис. 2.19.** Задание исходных параметров для расчета ФНЧ, ФВЧ, полосовых и режекторных фильтров

Выбрать минимально необходимый порядок фильтра позволяют следующие однотипные функции пакета Signal Processing:

```
[n, Wn] = buttord (Wp, Ws, Rp, Rs, 's')
[n, Wn] = cheblord (Wp, Ws, Rp, Rs, 's')
[n, Wn] = cheb2ord (Wp, Ws, Rp, Rs, 's')
[n, Wn] = ellipord (Wp, Ws, Rp, Rs, 's')
```

### ЗАМЕЧАНИЕ

Данные функции позволяют выбирать порядок как для аналоговых, так и для дискретных фильтров. Последний параметр 's' является признаком аналогового расчета. Использование этих функций для определения требуемого порядка дискретных фильтров обсуждается в разд. "Функции выбора порядка фильтров" главы 6.

Входной параметр  $R_p$  — допустимый уровень пульсаций в полосе пропускания (в децибелах),  $R_s$  — минимально необходимое затухание в полосе задерживания (в децибелах). Параметры  $W_p$  и  $W_s$  задают границы полос пропускания и задерживания, способ задания этих параметров зависит от типа проектируемого фильтра:

- ФНЧ:  $W_p$  и  $W_s$  — числа, при этом должно выполняться неравенство  $W_p < W_s$ ;
- ФВЧ:  $W_p$  и  $W_s$  — числа, при этом должно выполняться неравенство  $W_p > W_s$ ;
- полосовой фильтр:  $W_p$  и  $W_s$  — двухэлементные векторы, при этом должны выполняться неравенства  $W_s(1) < W_p(1) < W_p(2) < W_s(2)$ ;
- режекторный фильтр:  $W_p$  и  $W_s$  — двухэлементные векторы, при этом должны выполняться неравенства  $W_p(1) < W_s(1) < W_s(2) < W_p(2)$ .

Значения параметров  $W_p$  и  $W_s$  были обозначены на рис. 2.19 как  $\omega_p$  и  $\omega_s$  соответственно.

Выходными параметрами являются минимально необходимый для выполнения заданных требований порядок фильтра  $n$  и частота среза фильтра  $W_n$ . Эти параметры должны затем использоваться при вызове функции расчета фильтра. Возврат значения  $W_n$  избавляет пользователя от забот, связанных с тем, что при расчете разных фильтров понятие частоты среза имеет разный смысл. Для эллиптических фильтров и фильтров Чебышева первого рода  $W_n = W_p$ , для фильтров Чебышева второго рода  $W_n = W_s$ , а для фильтров Баттерворта значение  $W_n$  (напомним, что оно определяет частоту среза по уровню 3 dB) зависит от заданного уровня пульсаций  $R_p$  следующим образом:

$$\omega_n = \frac{\omega_p}{\sqrt[2n]{10^{R_p/10} - 1}}.$$

Поскольку порядок фильтра — величина целочисленная, то обычно оказывается, что фильтр минимально необходимого порядка обеспечивает некоторый запас по исходным параметрам. Этот запас можно использовать по-разному — либо сделать пульсации в полосе пропускания точно равными заданным, но увеличить затухание в полосе задерживания, либо точно выдержать заданное затухание в полосе задерживания, уменьшив при этом пульсации в полосе пропускания. Для эллиптических фильтров возможен еще один вариант — сужение переходной зоны за счет расширения полосы задерживания. Поведение функций выбора порядка фильтра в этом аспекте определяется тем, что при расчете фильтра должны будут использоваться

те же параметры пульсаций  $R_p$  и  $R_s$ , что и при выборе порядка фильтра. Поэтому для фильтров Баттервортса и Чебышева первого рода будет увеличиваться затухание в полосе задерживания, для фильтров Чебышева второго рода — уменьшаться пульсации в полосе пропускания, а для эллиптических фильтров — расширяться полоса задерживания.

## Расчет групповой задержки

В MATLAB нет функции, позволяющей вычислять групповую задержку для аналоговых систем (функция `grpdelay` предназначена только для дискретных систем; речь о ней пойдет в разд. "Расчет групповой задержки дискретной системы" главы 4). Поэтому в качестве небольшого примера рассмотрим возможную реализацию такой функции. Назовем ее, скажем, `grp_del`.

Прежде всего следует определиться с входными и выходными параметрами. Пусть функция получает векторы коэффициентов  $b$  и  $a$  полиномов числителя и знаменателя функции передачи, а также вектор значений круговой частоты  $w$  для расчета групповой задержки. Возвращаемый результат  $\tau_{\text{grp}}$  — вектор рассчитанных значений групповой задержки. Итак, функция будет использоваться следующим образом:

```
tau = grp_del(b, a, w)
```

Теперь нужно разобраться с алгоритмом работы функции, т. е. получить формулы, по которым будут выполняться вычисления. Для этого вспомним, что функция передачи выражается в виде отношения двух полиномов и что при делении комплексных чисел их фазы вычитаются. Если все это учесть в определении групповой задержки (см. формулу (2.3) в разд. "Фазовая и групповая задержка" этой главы), получим следующее:

$$\tau_{\text{rp}}(\omega) = \frac{d}{d\omega} \arg \frac{A(j\omega)}{B(j\omega)} = \frac{d}{d\omega} \arg A(j\omega) - \frac{d}{d\omega} \arg B(j\omega).$$

Фаза комплексного числа равна арктангенсу отношения мнимой и вещественной частей (добавление константы  $\pm\pi$ , необходимое в случае отрицательной вещественной части, не имеет значения, т. к. оно не влияет на результат дифференцирования по  $\omega$ ):

$$\tau_{\text{rp}}(\omega) = \frac{d}{d\omega} \operatorname{arctg} \frac{\operatorname{Im} A(j\omega)}{\operatorname{Re} A(j\omega)} - \frac{d}{d\omega} \operatorname{arctg} \frac{\operatorname{Im} B(j\omega)}{\operatorname{Re} B(j\omega)}.$$

Наконец, дифференцирование арктангенса дроби дает следующее:

$$\begin{aligned} \tau_{\text{rp}}(\omega) = & \frac{\operatorname{Re} A(j\omega) \frac{d}{d\omega} \operatorname{Im} A(j\omega) - \operatorname{Im} A(j\omega) \frac{d}{d\omega} \operatorname{Re} A(j\omega)}{|A(j\omega)|^2} - \\ & - \frac{\operatorname{Re} B(j\omega) \frac{d}{d\omega} \operatorname{Im} B(j\omega) - \operatorname{Im} B(j\omega) \frac{d}{d\omega} \operatorname{Re} B(j\omega)}{|B(j\omega)|^2}. \end{aligned} \quad (2.34)$$

Поскольку числитель и знаменатель функции передачи являются полиномами относительно переменной  $p = j\omega$ , выделение вещественной и мнимой частей, так же как и дифференцирование, сводится к простым манипуляциям с векторами коэффициентов полиномов. Комментарии на этот счет приводятся непосредственно в листинге функции:

```
function tau = grp_del(b, a, w)

% размещаем коэффициенты в порядке возрастания степеней
b = fliplr(b);
a = fliplr(a);
% выравниваем длины векторов
[b, a] = eqtflength(b, a);
% выделяем вещественную часть числителя
Br = b;
Br(2:2:end) = 0; % только четные степени
Br(3:4:end) = -Br(3:4:end); % через раз – знак "минус"
% выделяем мнимую часть числителя
Bi = b;
Bi(1:2:end) = 0; % только нечетные степени
Bi(4:4:end) = -Bi(4:4:end); % через раз – знак "минус"
% выделяем вещественную часть знаменателя
Ar = a;
Ar(2:2:end) = 0; % только четные степени
Ar(3:4:end) = -Ar(3:4:end); % через раз – знак "минус"
% выделяем мнимую часть знаменателя
Ai = a;
Ai(1:2:end) = 0; % только нечетные степени
Ai(4:4:end) = -Ai(4:4:end); % через раз – знак "минус"
%
Ar=fliplr(Ar);
Ai=fliplr(Ai);
Br=fliplr(Br);
Bi=fliplr(Bi);
% вычисляем знаменатели слагаемых
den_B = conv(Br, Br) + conv(Bi, Bi);
den_A = conv(Ar, Ar) + conv(Ai, Ai);
% вычисляем числители слагаемых
[num_B, tmp] = polyder(Bi, Br);
[num_A, tmp] = polyder(Ai, Ar);
% вычисляем групповую задержку
tau = -polyval(num_B, w) ./ polyval(den_B, w) + ...
    polyval(num_A, w) ./ polyval(den_A, w);
```

Этот программный код в точности реализует формулу (2.34). Приведем ряд дополнительных комментариев. Переворачивание векторов-строк "задом наперед" с по-

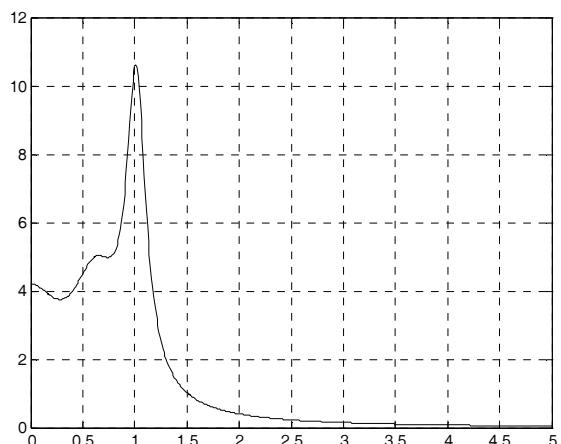
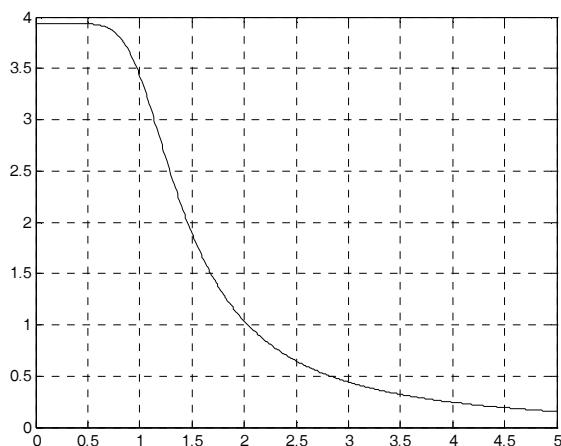
мощью функции `fliplr` производится исключительно для удобства выделения вещественных и мнимых частей — отсчитывать коэффициенты при этом проще от начала вектора, чем от конца. Функция `eqtflength` выравнивает длины переданных ей векторов, дополняя более короткий из них нулями в конце. Функция `polyder`, вызванная с двумя входными и выходными параметрами, вычисляет производную отношения двух полиномов, возвращая коэффициенты числителя и знаменателя результата. Нас в данном случае интересует только первый выходной параметр, поскольку числители слагаемых в (2.34) совпадают с числителями производных от дробей  $\text{Im } B(j\omega)/\text{Re } B(j\omega)$  и  $\text{Im } A(j\omega)/\text{Re } A(j\omega)$ . Умножение полиномов эквивалентно свертке векторов их коэффициентов, выполняемой с помощью функции `conv` (см. разд. "Дискретная свертка" главы 4). Для вычисления значений полиномов в заданных точках используется функция `polyval`.

Мы реализовали лишь простейший вариант — вычисление групповой задержки в заданных частотных точках. В принципе, функцию можно расширить, реализовав возможности, аналогичные тем, которыми обладает функция `freqs`: вывод графика при отсутствии выходных параметров, задание вектора частот по умолчанию и т. д. Заинтересованный читатель может сделать это, изучив исходный код функции `freqs` (файл `Program Files\ MATLAB\ R2010a\ toolbox\ signal\ signal\ freqs.m`).

В качестве примера применения разработанной функции (не забудьте сохранить ее в виде файла с именем `grp_del.m`) построим с ее помощью графики зависимости групповой задержки от частоты для двух рассмотренных ранее фильтров-прототипов — Чебышева первого рода и Бесселя (рис. 2.20):

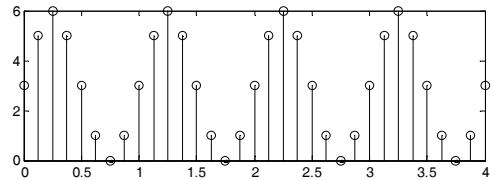
```
>> % фильтр Чебышева первого рода
>> [z, p, k] = cheb1ap(5, 0.5); % нули и полюсы прототипа
>> [b, a] = zp2tf(z, p, k); % коэффициенты функции передачи
>> w = 0:0.01:5;
>> tau = grp_del(b, a, w); % групповая задержка
>> plot(w, tau), grid % график групповой задержки
>> % фильтр Бесселя
>> [z, p, k] = besselap(5); % нули и полюсы прототипа
>> [b, a] = zp2tf(z, p, k); % коэффициенты функции передачи
>> w = 0:0.01:5;
>> tau = grp_del(b, a, w); % групповая задержка
>> figure
>> plot(w, tau), grid % график групповой задержки
```

Результаты расчетов показывают, что частотная зависимость групповой задержки для фильтра Чебышева первого рода имеет весьма изрезанный вид, а аналогичный параметр для фильтра Бесселя, как и было отмечено при рассмотрении соответствующего фильтра-прототипа, в полосе пропускания практически не меняется.

*a**б*

**Рис. 2.20.** Частотная зависимость групповой задержки для фильтров Чебышева первого рода (*а*) и Бесселя (*б*)

## ГЛАВА 3



## Дискретные сигналы

После двух вводных глав, посвященных основам теории сигналов и линейных систем, мы наконец приступаем к рассмотрению вопросов, непосредственно связанных с предметом данной книги — цифровой обработкой сигналов.

Сущность цифровой обработки состоит в том, что *физический сигнал* (напряжение, ток и т. д.) преобразуется в последовательность *чисел*, которая затем подвергается *математическим* преобразованиям в вычислительном устройстве. Трансформированный цифровой сигнал (последовательность чисел) при необходимости может быть преобразован обратно в напряжение или ток.

В данной главе рассмотрены принципы математического описания и анализа дискретных сигналов. Прежде всего следует пояснить некоторые терминологические тонкости, которые, возможно, уже были замечены внимательным читателем: в названии книги упоминается *цифровая* обработка сигналов, а данная глава посвящена *дискретным* сигналам. С обсуждения разницы между этими понятиями мы и начнем.

## **Аналоговые, дискретные и цифровые сигналы**

Исходный физический сигнал является *непрерывной* функцией времени. Такие сигналы, определенные во все моменты времени, называют *аналоговыми* (*analog*). Последовательность чисел, представляющая сигнал при цифровой обработке, является *дискретным рядом* (*discrete series*) и не может полностью соответствовать аналоговому сигналу. Числа, составляющие последовательность, являются значениями сигнала в отдельные (дискретные) моменты времени и называются *отсчетами* сигнала (*samples*). Как правило, отсчеты берутся через равные промежутки времени  $T$ , называемые *периодом дискретизации* (или *интервалом, шагом дискретизации* — *sample time*). Величина, обратная периоду дискретизации, называется *частотой дискретизации* (*sampling frequency*):  $f_d = 1/T$ . Соответствующая ей круговая частота определяется следующим образом:  $\omega_d = 2\pi/T$ .

Ясно, что в общем случае представление сигнала набором дискретных отсчетов приводит к потере информации, т. к. мы ничего не знаем о поведении сигнала в

промежутках между отсчетами. Однако, как будет показано далее в разд. "Теорема Котельникова" этой главы, существует класс аналоговых сигналов, для которых такой потери информации не происходит и которые могут быть *точно восстановлены* по значениям своих дискретных отсчетов.

Процесс преобразования аналогового сигнала в последовательность отсчетов называется *дискретизацией (sampling)*, а результат такого преобразования — *дискретным сигналом*.

При обработке сигнала в вычислительных устройствах его отсчеты представляются в виде двоичных чисел, имеющих ограниченное число разрядов. Вследствие этого отсчеты могут принимать лишь конечное множество значений и, следовательно, при представлении сигнала неизбежно происходит его округление. Процесс преобразования отсчетов сигнала в числа называется *квантованием по уровню (quantization)*, а возникающие при этом ошибки округления — *ошибками (или шумами) квантования (quantization error, quantization noise)*.

Сигнал, дискретный во времени, но не квантованный по уровню, называется *дискретным (discrete-time)* сигналом. Сигнал, дискретный во времени и квантованный по уровню, называют *цифровым (digital)* сигналом. Сигналы, квантованные по уровню, но непрерывные во времени, на практике встречаются редко. Разницу между аналоговыми, дискретными и цифровыми сигналами иллюстрирует рис. 3.1.

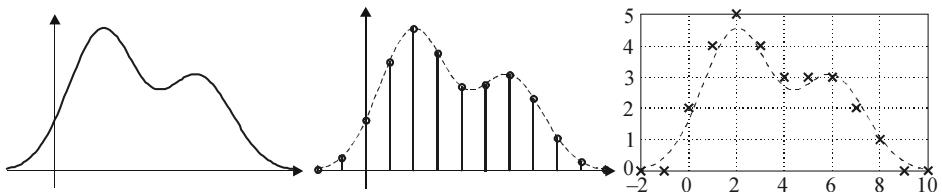


Рис. 3.1. Аналоговый (слева), дискретный (в центре) и цифровой (справа) сигналы

Вычислительные устройства, предназначенные для обработки сигналов, могут оперировать только цифровыми сигналами. Существуют также устройства, построенные в основном на базе аналоговой схемотехники, которые работают с дискретными сигналами, представленными в виде импульсов различной амплитуды или длительности. Чтобы подчеркнуть отсутствие квантования по уровню, такие устройства иногда называют *дискретно-аналоговыми (ДАУ)*.

В большей части книги речь пойдет о *дискретных* сигналах и методах их обработки, поскольку эффекты, связанные с квантованием по уровню, в большинстве случаев не будут приниматься во внимание. Однако этим эффектам, а также другим проблемам, связанным с конечной точностью представления чисел в вычислительных устройствах, специально посвящена глава 7.

## Аналого-цифровое и цифроаналоговое преобразование

Обобщенная структура системы цифровой обработки сигналов приведена на рис. 3.2. На вход поступает аналоговый сигнал  $s_{\text{вх}}(t)$ . Его временная дискретизация

и квантование по уровню производятся в *аналого-цифровом преобразователе* (АЦП; английский термин — *Analog-to-Digital Converter, ADC*). Вообще эти два процесса — дискретизация и квантование — являются независимыми друг от друга, но они, как правило, выполняются внутри одной микросхемы. Выходным сигналом АЦП является последовательность чисел, поступающая в *цифровой процессор* (ЦП), выполняющий требуемую обработку. Процессор осуществляет различные математические операции над входными отсчетами; ранее полученные отсчеты и промежуточные результаты могут сохраняться в памяти процессора для использования в последующих вычислениях. Результатом работы процессора является новая последовательность чисел, представляющих собой отсчеты выходного сигнала. Аналоговый выходной сигнал  $s_{\text{вых}}(t)$  восстанавливается по этой последовательности чисел с помощью *цифроаналогового преобразователя* (ЦАП; английский термин — *Digital-to-Analog Converter, DAC*). Напряжение на выходе ЦАП имеет ступенчатую форму (это также показано на рис. 3.2); при необходимости оно может быть преобразовано в плавно меняющийся выходной сигнал с помощью *сглаживающего фильтра*  $\Phi$ .

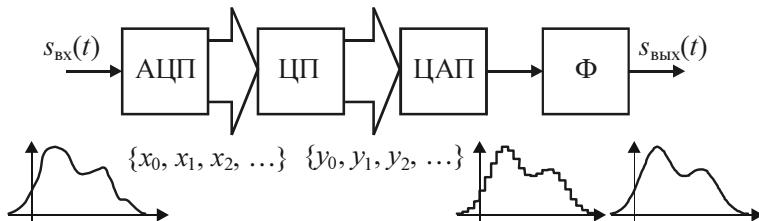


Рис. 3.2. Структурная схема системы цифровой обработки сигналов

Устройства, реализуемые с помощью структуры типа рис. 3.2, могут иметь самый разнообразный характер. В цифровой форме можно создавать фильтры, анализаторы спектра, нелинейные преобразователи сигналов и многое другое.

### ЗАМЕЧАНИЕ

Использование входных и выходных сигналов в аналоговой форме (и, следовательно, наличие АЦП и ЦАП) не всегда является необходимым. Так, при реализации цифрового генератора сигналов не нужен входной аналоговый сигнал, а ЦАП может отсутствовать, если конечный результат необходим только в цифровой форме.

## Частота Найквиста

Гармонический сигнал может быть адекватно представлен дискретными отсчетами, если его частота не превышает половины частоты дискретизации (этота частота называется *частотой Найквиста* (*Nyquist frequency*) —  $f_N = f_d/2 = 1/(2T)$ ;  $\omega_N = \omega_d/2 = \pi/T$ ). Происхождение этого ограничения поясняет рис. 3.3. В зависимости от соотношения между частотой дискретизируемого гармонического сигнала и частотой Найквиста возможны три случая.

- Если частота гармонического сигнала *меньше* частоты Найквиста, дискретные отсчеты позволяют правильно восстановить аналоговый сигнал (рис. 3.3, а).
- Если частота гармонического сигнала *равна* частоте Найквиста, то дискретные отсчеты позволяют восстановить аналоговый гармонический сигнал с той же частотой, но амплитуда и фаза восстановленного сигнала (он показан пунктирной линией) могут быть искажены (рис. 3.3, б). В худшем случае все дискретные отсчеты синусоиды могут оказаться равными нулю.
- Если частота гармонического сигнала *больше* частоты Найквиста, восстановленный по дискретным отсчетам аналоговый сигнал (как и в предыдущем случае, он показан пунктирной линией) будет также гармоническим, но с иной частотой (рис. 3.3, в). Данный эффект носит название *появления ложных частот (aliasing)*, мы еще вернемся к его рассмотрению в разд. "Спектр дискретного сигнала" этой главы.

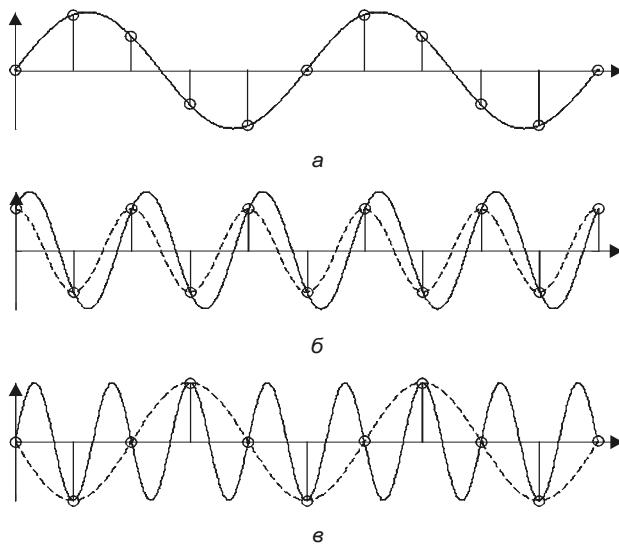


Рис. 3.3. Дискретизация гармонических сигналов с разной частотой

### **ЗАМЕЧАНИЕ**

Эффекты, связанные с дискретизацией периодических процессов, наглядно проявляются при кино- и видеосъемке вращающихся объектов (таких, например, как колеса автомобилей). Из-за недостаточно высокой частоты дискретизации (частоты смены кадров) быстро вращающееся колесо может выглядеть неподвижным либо медленно поворачивающимся (причем в любую сторону).

## **Спектр дискретного сигнала**

Преобразование Фурье позволяет вычислить спектральную плотность сигнала, представляющего собой *функцию* (как правило, времени либо пространственных

координат). Дискретный же сигнал является *последовательностью* чисел, поэтому для анализа его спектра обычными (аналоговыми) средствами необходимо сопоставить этой последовательности некоторую функцию.

Традиционным способом такого сопоставления является представление отсчетов в виде дельта-функций с соответствующими множителями и задержками. Для последовательности отсчетов  $\{x(k)\}$  получится следующий сигнал:

$$s(t) = \sum_{k=-\infty}^{\infty} x(k)\delta(t - k). \quad (3.1)$$

Преобразование Фурье линейно, спектр дельта-функции равен единице, а задержка сигнала во времени приводит к умножению спектра на комплексную экспоненту. Все это (см. разд. "Свойства преобразования Фурье" главы 1) позволяет сразу же записать спектр дискретного сигнала:

$$\dot{S}(\omega) = \sum_{k=-\infty}^{\infty} x(k)e^{-j\omega k}. \quad (3.2)$$

Из этой формулы видно главное свойство спектра любого дискретного сигнала: спектр является периодическим, и его период в данном случае равен  $2\pi$  (т. е. круговой частоте дискретизации, поскольку, составляя сигнал из дельта-функций, мы выбрали единичный интервал между ними, что дает  $\omega_d = 2\pi$ ):

$$\dot{S}(\omega \pm 2\pi) = \dot{S}(\omega).$$

Следует также обратить внимание на размерность спектральной функции дискретного сигнала: она совпадает с размерностью отсчетов. Это связано с тем, что дельта-функции времени, из которых был составлен сигнал (3.1), имеют размерность частоты (см. разд. "Классификация сигналов" главы 1).

Формула (3.2) позволяет вычислить спектральную функцию по известным отсчетам  $x(k)$ . При конечном числе ненулевых отсчетов этот расчет несложен; он может быть выполнен с помощью функции MATLAB `freqz` (подробнее об этой функции см. разд. "Расчет частотных характеристик" главы 4).

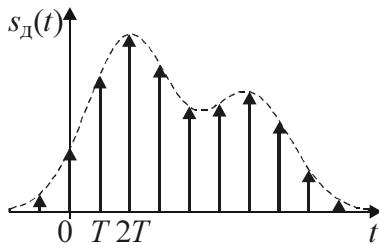
Теперь рассмотрим несколько иную задачу. Пусть значения  $x(k)$  являются отсчетами аналогового сигнала  $s(t)$ , взятыми с периодом  $T$ :

$$x(k) = s(kT).$$

Выясним, как в этом случае спектр дискретного сигнала (3.2) связан со спектром аналогового сигнала  $\dot{S}(\omega)$ .

Итак, мы рассматриваем *дискретизированный* сигнал в виде последовательности дельта-функций, "взвешенной" значениями отсчетов  $s(kT)$  аналогового сигнала  $s(t)$  (рис. 3.4):

$$s_d(t) = \sum_{k=-\infty}^{\infty} s(kT)\delta(t - kT). \quad (3.3)$$



**Рис. 3.4.**  
Дискретизированный сигнал  
в виде последовательности  
дельта-функций

### ЗАМЕЧАНИЕ

Термин "дискретизированный" в данном контексте подчеркивает, что последовательность отсчетов получена именно в результате дискретизации аналогового сигнала.

Так как функция  $\delta(t - kT)$  равна нулю всюду, кроме момента  $t = kT$ , можно заменить в выражении (3.3) константы  $s(kT)$  на исходный непрерывный сигнал  $s(t)$ :

$$s_d(t) = s(t) \sum_{k=-\infty}^{\infty} \delta(t - kT). \quad (3.4)$$

Далее заметим, что сумма, входящая в выражение (3.4), является периодическим сигналом, а потому может быть представлена в виде ряда Фурье. Коэффициенты этого ряда, согласно (1.12), равны

$$\hat{C}_n = \frac{1}{T} \int_{-T/2}^{T/2} \delta(t) e^{-j\omega_n t} dt = \frac{1}{T}. \quad (3.5)$$

В формуле (3.5) было учтено, что в интервал интегрирования  $(-T/2, T/2)$  попадает только одна дельта-функция, соответствующая  $k = 0$ .

Таким образом, периодическая последовательность дельта-функций может быть представлена в виде комплексного ряда Фурье (1.10):

$$\sum_{k=-\infty}^{\infty} \delta(t - kT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j\omega_n t}, \quad (3.6)$$

где  $\omega_n = 2\pi n/T$ . Подставив (3.6) в (3.4), получим

$$s_d(t) = \frac{s(t)}{T} \sum_{n=-\infty}^{\infty} e^{j\omega_n t} = \frac{1}{T} \sum_{n=-\infty}^{\infty} s(t) e^{j\omega_n t}.$$

Умножение сигнала на  $\exp(j\omega_n t)$  соответствует сдвигу спектральной функции на  $\omega_n$ , поэтому спектр дискретизированного сигнала можно записать следующим образом:

$$\hat{S}_d(\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \hat{S}\left(\omega - \frac{2\pi n}{T}\right). \quad (3.7)$$

Таким образом, спектр дискретизированного сигнала представляет собой бесконечный ряд сдвинутых копий спектра исходного непрерывного сигнала  $s(t)$  (рис. 3.5). Расстояние по частоте между соседними копиями спектра равно частоте дискретизации  $\omega_d = 2\pi/T$ .

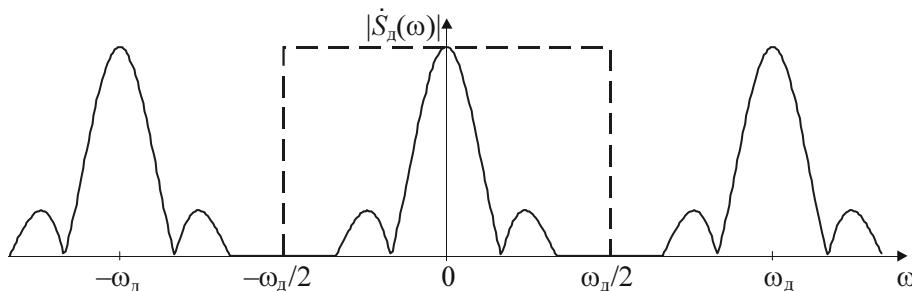


Рис. 3.5. Спектр дискретизированного сигнала

Следует также отметить, что из-за наличия в формуле (3.7) множителя  $1/T$  спектр дискретизированного сигнала имеет размерность, совпадающую с размерностью сигнала (как уже говорилось, это связано с тем, что функция  $\delta(t)$  имеет размерность частоты).

Характер спектра дискретизированного сигнала еще раз демонстрирует частотно-временную дуальность преобразования Фурье:

- периодический сигнал  $\rightarrow$  дискретный спектр;
- периодический спектр  $\rightarrow$  дискретный сигнал.

В начале данного раздела мы получили формулу (3.2), позволяющую рассчитать спектр последовательности отсчетов  $\{x(k)\}$ , никак не связывая эти отсчеты с аналоговым сигналом. Только что полученная формула (3.7) предполагает, что отсчеты  $\{x(k)\}$  получены путем дискретизации аналогового сигнала  $s(t)$ , и показывает связь между спектрами дискретизированного и аналогового сигналов. Нужно подчеркнуть, что эти две формулы дают *одинаковый результат*.

Отсюда следует еще один важный факт. Соединить отсчеты  $\{x(k)\}$  для получения аналогового сигнала можно произвольным образом. В каждом случае аналоговый сигнал будет, разумеется, иметь свой спектр. Однако результат *суммирования* сдвинутых копий спектров по формуле (3.7) всегда будет одним и тем же, поскольку определяется только значениями дискретных отсчетов  $\{x(k)\} = \{s(kT)\}$  и формулой (3.2).

Рисунок 3.5 наглядно демонстрирует и способ восстановления непрерывного сигнала по дискретным отсчетам. Для этого необходимо пропустить дискретный сигнал через идеальный фильтр нижних частот (ФНЧ) с частотой среза, равной половине частоты дискретизации. АЧХ такого фильтра показана на рис. 3.5 пунктиром.

Очевидно, что точное восстановление сигнала возможно, если сдвинутые копии спектра не перекрываются. Из рис. 3.5 видно, что для этого необходимо, чтобы час-

тота дискретизации как минимум в два раза превышала верхнюю граничную частоту в спектре сигнала:

$$\omega_d > 2\omega_b. \quad (3.8)$$

Спектральное представление дискретного сигнала позволяет объяснить появление ложных частот (*aliasing*), речь о которых шла в разд. "Частота Найквиста" этой главы. Пусть дискретизации подвергается гармонический сигнал с частотой  $\omega_0$ , превышающей частоту Найквиста, но меньшей частоты дискретизации. Спектр такого сигнала показан на рис. 3.6, сверху. Сдвинутые копии спектра, возникающие при дискретизации, создают попадающие в полосу восстановления (от нуля до частоты Найквиста) составляющие с частотой  $\omega_d - \omega_0$  (рис. 3.6, снизу). Спектры, получающиеся после дискретизации гармонических сигналов с частотами  $\omega_0$  и  $\omega_d - \omega_0$ , оказываются идентичными. Данный рисунок иллюстрирует в частотной области процесс дискретизации гармонического сигнала, показанный ранее на рис. 3.3.

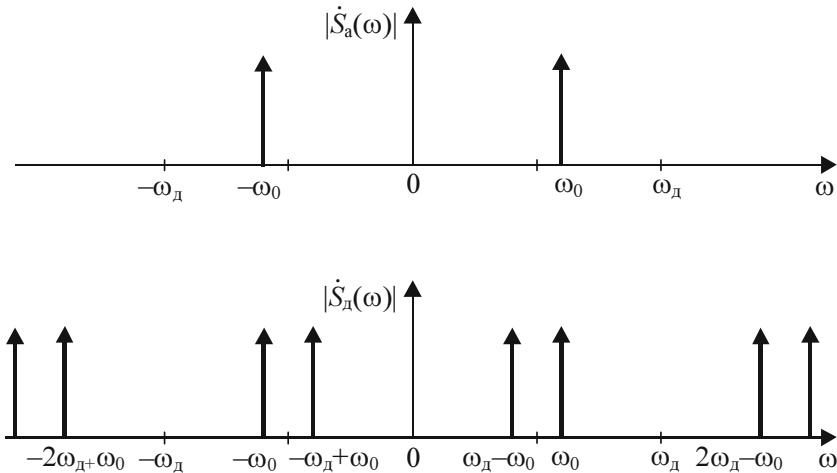
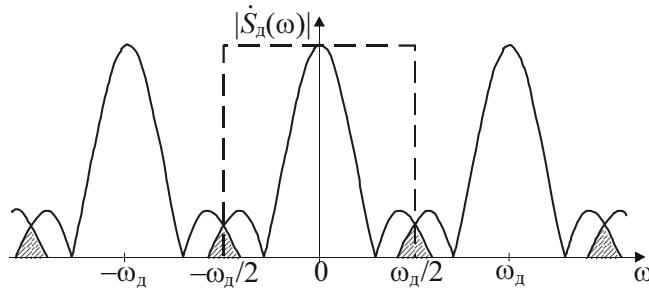


Рис. 3.6. Спектры аналоговой (сверху) и дискретизированной (снизу) синусоиды с частотой, превышающей частоту Найквиста

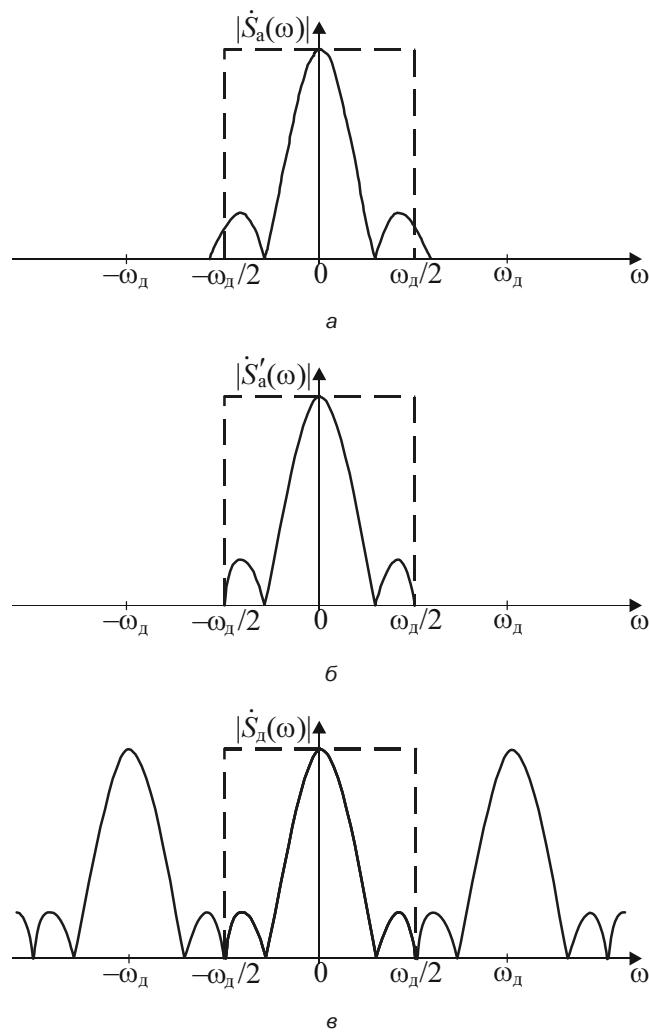
В случае произвольного сигнала, если условие (3.8) не выполняется, сдвинутые копии спектра будут накладываться друг на друга, что приведет к неизбежным искажениям при восстановлении непрерывного сигнала (рис. 3.7).

Эти искажения вызваны тем, что спектральные составляющие сигнала с частотами, превышающими частоту Найквиста, равную  $\omega_d/2$ , не могут быть восстановлены правильно — вместо этого они вызывают наложение "хвостов" соседних сдвинутых копий спектра и появление ложных частот.

Если подлежащий дискретизации сигнал может содержать спектральные составляющие с частотами, превышающими частоту Найквиста, полезно предварительно пропустить его через ФНЧ с частотой среза, равной частоте Найквиста (рис. 3.8).



**Рис. 3.7.** Перекрытие сдвинутых копий спектра при недостаточно высокой частоте дискретизации



**Рис. 3.8.** При дискретизации сигнала, содержащего высокочастотные составляющие (а), полезно пропустить его через ФНЧ (б), чтобы избежать появления ложных частот (в)

При этом все равно будут потеряны высокочастотные составляющие — сохранить их можно лишь путем повышения частоты дискретизации. Однако в этом случае благодаря отсутствию наложения "хвостов" не произойдет появления ложных частот и диапазон частот  $0 \dots \omega_d/2$  будет представлен в дискретном сигнале без искажений.

## Влияние формы дискретизирующих импульсов

Запишем выражение (3.3) в более общей форме, используя вместо дельта-функций импульсы  $s_0(t)$  произвольной формы:

$$s_d(t) = \sum_{k=-\infty}^{\infty} s(kT)s_0(t-kT). \quad (3.9)$$

Получить выражение для спектральной функции сигнала (3.9) будет проще всего, если рассмотреть этот сигнал как результат прохождения последовательности дельта-функций (3.3) через линейную стационарную систему с импульсной характеристикой  $s_0(t)$ . Действительно, при этом каждая из дельта-функций в (3.3) будет порождать на выходе цепи сигнал  $s_0(t)$  с соответствующими временной задержкой и амплитудным множителем, так что в результате получится именно выражение (3.9).

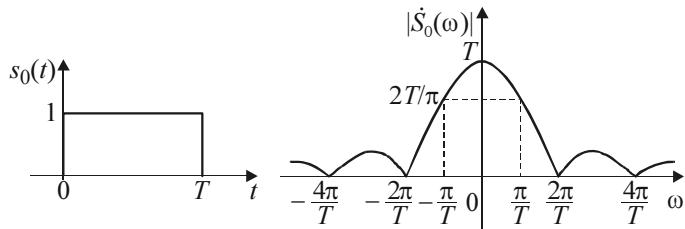
Поскольку при прохождении сигнала через линейную систему с постоянными параметрами его спектр умножается на комплексный коэффициент передачи этой системы (см. главу 2), спектр сигнала (3.9) будет отличаться от выражения (3.7) лишь дополнительным множителем — спектром импульса  $s_0(t)$ :

$$\dot{S}_d(\omega) = \frac{\dot{S}_0(\omega)}{T} \sum_{n=-\infty}^{\infty} \dot{S}\left(\omega - \frac{2\pi n}{T}\right). \quad (3.10)$$

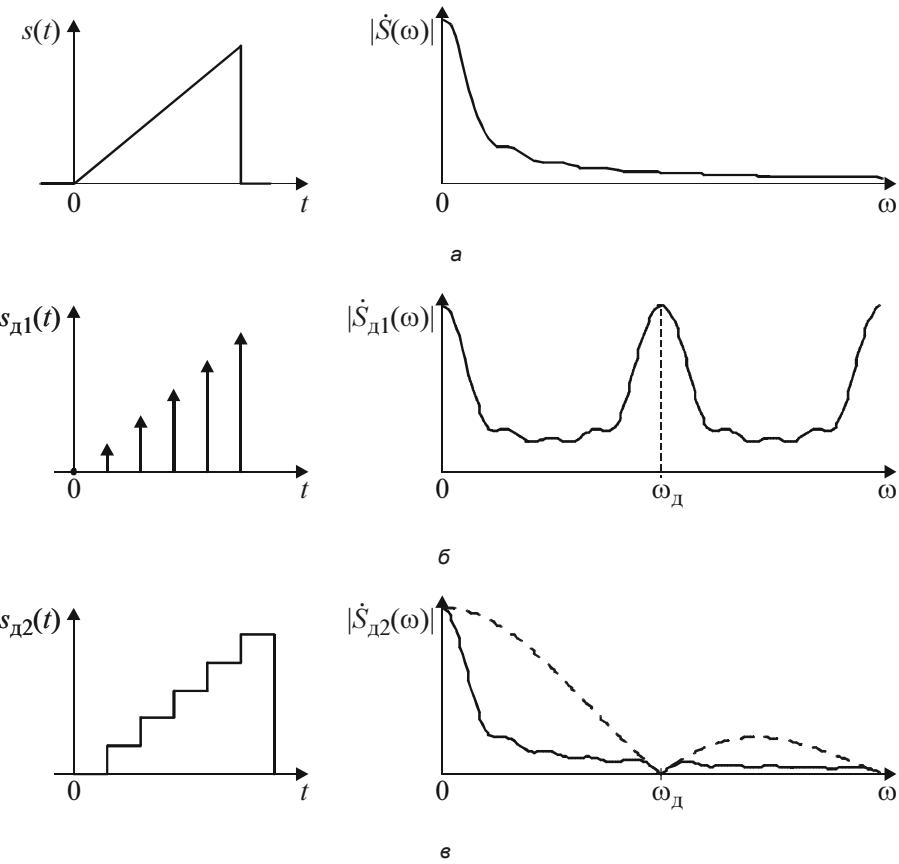
Итак, отличие формы дискретизирующих импульсов от дельта-функций вызывает *мультипликативные* искажения спектра дискретного сигнала. Спектральная функция импульса, имеющего конечную энергию, затухает с ростом частоты, поэтому возникающие при дискретизации сдвинутые копии спектра сигнала  $s(t)$  оказываются ослабленными.

Рассмотрим важный с практической точки зрения случай, когда  $s_0(t)$  представляет собой прямоугольный импульс с единичной амплитудой и длительностью, равной периоду дискретизации (рис. 3.9, слева). В данном случае дискретный сигнал приобретает ступенчатую форму, что характерно для сигнала на выходе ЦАП перед сглаживающим фильтром (см. рис. 3.2). Искажения спектра при этом описываются множителем  $\dot{S}_0(\omega)$  следующего вида:

$$\dot{S}_0(\omega) = \int_{-\infty}^{\infty} s_0(t)e^{-j\omega t} dt = T \frac{\sin(\omega T / 2)}{\omega T / 2} e^{-j\omega T / 2}.$$



**Рис. 3.9.** Прямоугольный дискретизирующий импульс (слева) и его амплитудный спектр (справа)



**Рис. 3.10.** Дискретизация треугольного импульса (слева — сигналы, справа — спектры): *a* — исходный аналоговый сигнал, *б* — дискретный сигнал в виде последовательности дельта-функций, *в* — ступенчатый сигнал (пунктирной линией показана форма амплитудного спектра дискретизирующих импульсов)

График модуля функции  $\dot{S}_0(\omega)$  приведен на рис. 3.9, справа. Спад амплитудного спектра на частоте Найквиста, равной  $\pi/T$ , составляет

$$\frac{|\dot{S}_0(\pi/T)|}{|\dot{S}_0(0)|} = \frac{\sin(\pi/2)}{\pi/2} = \frac{2}{\pi} \approx 0,637 \approx -3,9 \text{ дБ.}$$

В качестве примера на рис. 3.10 приведены результаты дискретизации треугольного импульса дельта-функциями и рассмотренными прямоугольными импульсами.

Из графиков видно, что ЦАП сам по себе является фильтром нижних частот, однако с весьма невысокой степенью подавления сдвинутых копий спектра. Кроме того, поскольку АЧХ такого фильтра весьма далека от прямоугольной, он обладает сильной неравномерностью в полосе пропускания и заметно ослабляет высокочастотные составляющие сигнала (на частоте Найквиста ослабление составляет почти 4 дБ).

## Теорема Котельникова

В начале данной главы уже было сформулировано утверждение о том, что некоторые сигналы могут быть без потерь информации представлены своими дискретными отсчетами. Полученное в предыдущем разделе выражение для спектра дискретизированного сигнала позволяет выделить тот класс сигналов, для которых это возможно, и описать способ такого восстановления.

Согласно (3.7), спектр дискретизированного сигнала представляет собой сумму сдвинутых копий спектра исходного сигнала, при этом шаг сдвига равен частоте дискретизации  $\omega_d$ . Из рис. 3.5 видно, что если в спектре аналогового сигнала не содержится составляющих с частотами, превышающими частоту Найквиста ( $\omega_d/2$ ), то сдвинутые копии спектра не будут перекрываться. В этом случае использование идеального ФНЧ с прямоугольной АЧХ позволит выделить исходную (несдвинутую) копию спектра, сосредоточенную в окрестностях нулевой частоты, и, таким образом, в точности восстановить исходный аналоговый сигнал.

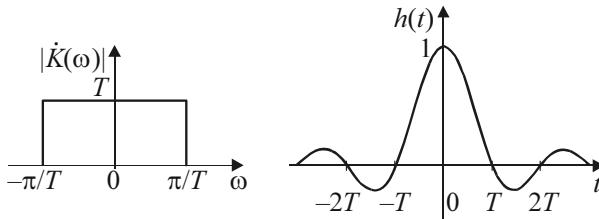
АЧХ идеального ФНЧ, восстанавливающего аналоговый сигнал, приведена на рис. 3.11, слева. Коэффициент передачи в полосе пропускания равен  $T$ , а не единице, чтобы компенсировать множитель  $1/T$  в формуле (3.7). С помощью обратного преобразования Фурье найдем импульсную характеристику фильтра (рис. 3.11, справа):

$$h(t) = \frac{\sin\left(\pi \frac{t}{T}\right)}{\pi \frac{t}{T}}. \quad (3.11)$$

Дискретизированный сигнал (3.3) представляет собой сумму дельта-функций. При прохождении такого сигнала через восстанавливающий ФНЧ каждая дельта-функция породит на выходе соответствующим образом сдвинутую и масштабиро-

ванную копию импульсной характеристики фильтра. Выходной сигнал (в точности соответствующий исходному аналоговому сигналу), таким образом, будет представлять собой сумму сдвинутых и умноженных на отсчеты сигнала копий импульсных характеристик идеального ФНЧ (3.11):

$$s(t) = \sum_{k=-\infty}^{\infty} s(kT) \frac{\sin\left(\pi \frac{t - kT}{T}\right)}{\pi \frac{t - kT}{T}}. \quad (3.12)$$



**Рис. 3.11.** Амплитудно-частотная (слева) и импульсная (справа) характеристики идеального восстанавливающего фильтра

Подводя итог всему сказанному, сформулируем *теорему Котельникова*: любой сигнал  $s(t)$ , спектр которого не содержит составляющих с частотами выше некоторого значения  $\omega_b = 2\pi f_b$ , может быть без потерь информации представлен своими дискретными отсчетами  $\{s(kT)\}$ , взятыми с интервалом  $T$ , удовлетворяющим следующему неравенству:

$$T \leq \frac{1}{2f_b} = \frac{\pi}{\omega_b}. \quad (3.13)$$

Восстановление исходного непрерывного сигнала  $s(t)$  по набору его дискретных отсчетов  $\{s(kT)\}$  производится по формуле (3.12).

### ЗАМЕЧАНИЕ

В зарубежных источниках данная теорема называется *теоремой Найквиста (Nyquist theorem)*, *теоремой Шеннона (Shannon theorem)* или *теоремой дискретизации (sampling theorem)*.

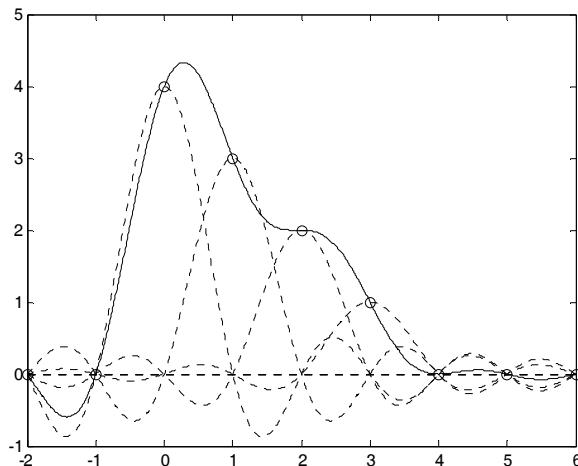
Формула (3.12) представляет собой разложение сигнала  $s(t)$  в ряд по системе функций  $\{\varphi_k(t)\}$ , называемой *базисом Котельникова*:

$$\varphi_k(t) = \frac{\sin\left(\frac{\pi}{T}(t - kT)\right)}{\frac{\pi}{T}(t - kT)}.$$

Формирование непрерывного сигнала по его дискретным отсчетам поясняет рис. 3.12. Пунктиром показаны графики отдельных слагаемых формулы (3.12),

сплошной линией — восстановленный сигнал. Далее приводится код MATLAB, использованный при построении рисунка:

```
>> t = -2:0.01:6; % время для восстановленного сигнала
>> td = -2:6; % номера отсчетов
>> s = [0 0 4 3 2 1 0 0 0]; % дискретный сигнал
>> d = [td' s']; % данные для функции pulstran
>> y = pulstran(t, d, 'sinc'); % восстановленный сигнал
>> plot(td, s, 'o', t, y) % график восстановленного сигнала
>> hold on % выводим графики отдельных sinc-импульсов
>> for k=1:length(s), plot(t, s(k)*sinc(t-td(k)), ':'), end
>> hold off
```



**Рис. 3.12.** Восстановление непрерывного сигнала по его дискретным отсчетам

### ЗАМЕЧАНИЕ

При построении рис. 3.12 была использована функция `pulstran`. Она позволяет сформировать сигнал в виде суммы конечного числа импульсов произвольной формы с заданными задержками и множителями, что делает ее очень удобной при построении графиков сигналов, восстановленных по дискретным отсчетам согласно теореме Котельникова. Эта функция подробно рассмотрена далее в разд. "Генерация последовательности импульсов" этой главы.

Рисунок 3.12 наглядно демонстрирует главное свойство сигнала с ограниченным спектром — его бесконечность во времени. Хотя отличны от нуля лишь несколько отсчетов показанного сигнала, аналоговый сигнал оказывается бесконечно колеблющимся — между нулевыми отсчетами (на рисунке это отсчеты с номерами -2, -1, 4, 5, 6) его значения отличны от нуля. Эти колебания нигде не заканчиваются, хотя их амплитуда стремится к нулю.

Иногда можно встретить примерно следующее "объяснение" сущности теоремы Котельникова: "если брать отсчеты достаточно часто, в промежутках между ними сигнал с ограниченным спектром не успеет сильно измениться, и мы сможем точно

восстановить его". Такая трактовка является принципиально неправильной. Замечательное обсуждение этого вопроса содержится в [6], здесь же ограничимся краткими пояснениями и одним примером.

Когда мы говорим об ограниченной полосе частот сигнала, имеется в виду спектральная функция *всего сигнала, имеющего бесконечную длительность*. При этом *мгновенные спектры отдельных фрагментов* сигнала могут содержать сколь угодно высокие частоты.

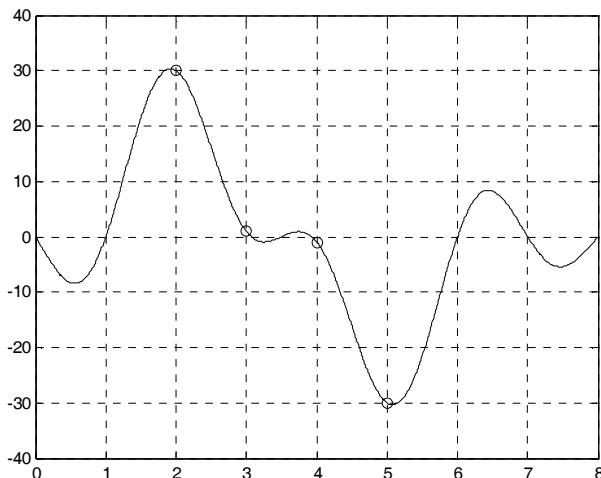
### ЗАМЕЧАНИЕ

Под мгновенным спектром подразумевается спектральная функция "вырезанного" из сигнала фрагмента конечной длительности.

В частности, в отдельном промежутке между соседними отсчетами сигнал с ограниченным спектром может иметь сколь угодно сложную форму, например произвольное число раз менять знак.

В качестве примера восстановим по формуле (3.12) непрерывный сигнал на основе последовательности, содержащей четыре ненулевых отсчета: ..., 0, 0, 30, 1, -1, -30, 0, 0, ... (рис. 3.13):

```
>> t = 0:0.01:8; % время для восстановленного сигнала
>> td = 2:5; % номера ненулевых отсчетов
>> s = [30 1 -1 -30]; % дискретный сигнал
>> d = [td' s']; % данные для функции pulstran
>> y = pulstran(t, d, 'sinc'); % восстановленный сигнал
>> plot(td, s, 'o', t, y)
>> grid
```



**Рис. 3.13.** Сигнал с ограниченным спектром, содержащий фрагмент с колебаниями высокой частоты

Результат, показанный на рис. 3.13, свидетельствует о том, что фрагмент восстановленного сигнала между 3 и 4 отсчетами представляет собой колебание с перио-

дом, равным интервалу дискретизации (т. е. с частотой, вдвое превышающей частоту Найквиста!). Однако *весь* сигнал, составленный из сдвинутых функций базиса Котельникова, не содержит спектральных составляющих с частотами, большими частоты Найквиста.

### ЗАМЕЧАНИЕ

Подобным образом можно получить произвольное число колебаний сигнала между двумя его соседними отсчетами, однако значения остальных отсчетов при этом быстро возрастают. Например, чтобы получить между отсчетами полтора колебания вместо одного, можете повторить приведенный пример, использовав следующую последовательность значений: {600, 360, 1, 1, 360, 600}. Чтобы увидеть колебания сигнала между единичными отсчетами, придется увеличить масштаб отображения этого фрагмента графика.

## Восстановление радиосигнала по отсчетам видеосигнала

Если для восстановления сигнала воспользоваться не ФНЧ, а идеальным полосовым фильтром со средней частотой  $n\omega_d$  и шириной полосы пропускания, равной  $\omega_d$ , будет выделена пара *сдвинутых* копий спектра  $\dot{S}\left(\omega \pm \frac{2\pi n}{T}\right)$ . Из свойств преобразования Фурье (см. формулу (1.24) в разд. "Умножение сигнала на гармоническую функцию" главы I) следует, что такой спектр соответствует радиосигналу вида

$$s_p(t) = 2s(t)\cos(n\omega_d t) \quad (3.14)$$

(предполагается, что условие теоремы Котельникова выполнено и сдвинутые копии спектра не перекрываются). Таким образом, с помощью полосового фильтра можно из дискретных отсчетов видеосигнала получить аналоговый радиосигнал.

Импульсная характеристика идеального полосового фильтра с указанными параметрами имеет вид

$$h(t) = \frac{\sin\left(\frac{\pi}{T}t\right)}{\frac{\pi}{T}t} \cos\left(\frac{2\pi n}{T}t\right).$$

Поэтому во временной области формирование радиосигнала по отсчетам видеосигнала описывается следующим образом:

$$s(t) = \left( \sum_{k=-\infty}^{\infty} s(kT) \frac{\sin\left(\frac{\pi}{T}(t-kT)\right)}{\frac{\pi}{T}(t-kT)} \right) \cos\left(\frac{2\pi n}{T}t\right) \quad (3.15)$$

**ЗАМЕЧАНИЕ**

В формулах (3.14) и (3.15) имеется множитель  $\cos(n\omega_d t)$ , что свидетельствует о чисто амплитудной модуляции получаемого радиосигнала. Подробнее о модуляции см. в главе 8.

## Квадратурная дискретизация узкополосных сигналов

Пусть дискретизации необходимо подвергнуть узкополосный сигнал, который в общем случае может иметь как амплитудную, так и угловую модуляцию:

$$s(t) = A(t) \cos(\omega_0 t + \phi(t)),$$

где  $\omega_0$  — несущая частота,  $A(t)$  и  $\phi(t)$  — законы амплитудной и фазовой модуляции соответственно, представляющие собой медленно (по сравнению с  $\cos(\omega_0 t)$ ) меняющиеся функции.

Частоту дискретизации для такого сигнала можно выбрать исходя непосредственно из теоремы Котельникова, при этом она должна быть не меньше, чем  $2(\omega_0 + \Delta\omega/2) = 2\omega_0 + \Delta\omega$ , где  $\Delta\omega$  — ширина спектра сигнала  $s(t)$ . Однако возможен другой вариант (правда, требующий предварительной обработки сигнала) — *квадратурная дискретизация*.

Запишем аналитический сигнал для  $s(t)$  (аналитический сигнал рассмотрен в разд. "Комплексная огибающая" главы 1):

$$\dot{s}(t) = A(t) e^{j\phi(t)} e^{j\omega_0 t}.$$

Информация, переносимая сигналом, заключена в его комплексной огибающей

$$\dot{A}(t) = A(t) e^{j\phi(t)}.$$

Спектр этого комплексного сигнала занимает полосу частот  $-\Delta\omega/2 \dots \Delta\omega/2$ . Поэтому, согласно теореме Котельникова, минимально допустимая частота дискретизации для этого сигнала равна  $\Delta\omega$ , что значительно меньше, чем в случае прямой дискретизации. Однако отсчеты такого сигнала, естественно, будут *комплексными*.

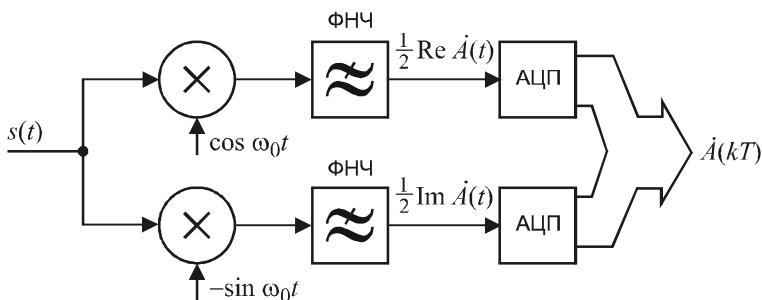


Рис. 3.14. Квадратурная дискретизация

Рассмотрим способ реализации квадратурной дискретизации (рис. 3.14). Входной сигнал умножается на колебания двух генераторов (*гетеродинов, heterodyne*) с час-

тотой  $\omega_0$ , сдвинутые по фазе друг относительно друга на  $90^\circ$  —  $\cos \omega_0 t$  и  $-\sin \omega_0 t$ . Результат каждого из умножений содержит две составляющие — низкочастотную и высокочастотную (на частоте  $2\omega_0$ ):

$$\begin{aligned}s(t) \cos \omega_0 t &= A(t) \cos(\omega_0 t + \varphi(t)) \cos \omega_0 t = \\&= \frac{1}{2} A(t) \cos \varphi(t) + \frac{1}{2} A(t) \cos(2\omega_0 t + \varphi(t)) = \\&= \frac{1}{2} \operatorname{Re} \dot{A}(t) + \frac{1}{2} A(t) \cos(2\omega_0 t + \varphi(t)), \\-s(t) \sin \omega_0 t &= -A(t) \cos(\omega_0 t + \varphi(t)) \sin \omega_0 t = \\&= \frac{1}{2} A(t) \sin \varphi(t) - \frac{1}{2} A(t) \cos(2\omega_0 t - \varphi(t)) = \\&= \frac{1}{2} \operatorname{Im} \dot{A}(t) - \frac{1}{2} A(t) \cos(2\omega_0 t - \varphi(t)).\end{aligned}$$

Таким образом, для получения вещественной и мнимой частей комплексной огибающей сигнала нужно после перемножения пропустить результаты через фильтры нижних частот, чтобы устраниТЬ вторую гармонику несущей. Полученные в результате сигналы, пропорциональные  $\operatorname{Re} \dot{A}(t)$  и  $\operatorname{Im} \dot{A}(t)$ , подвергаются дискретизации с частотой не ниже, чем

$$\frac{\Delta\omega}{2} \cdot 2 = \Delta\omega.$$

### ЗАМЕЧАНИЕ

Данная процедура сдвига спектра сигнала путем умножения его на опорное гармоническое колебание с последующим выделением нужных спектральных составляющих с помощью фильтра называется *гетеродинированием* (*heterodyning*).

Рассмотренная схема получила широкое распространение при обработке радиосигналов. Квадратурное гетеродинирование просто и дешево реализуется аналоговыми средствами, цифровая часть благодаря низкой частоте дискретизации также оказывается сравнительно несложной.

## Субдискретизация сигнала

В предыдущем разделе мы видели, как квадратурная обработка позволяет дискретизировать полосовой сигнал с частотой, определяемой не верхней границей, а шириной спектра сигнала. Однако в ряде случаев возможен и другой подход, когда с пониженнной частотой дискретизируется сам исходный сигнал, без предварительной обработки. Рассмотрим подробнее идею этого метода, называемого *субдискретизацией* (*undersampling*) или *преобразованием частоты на АЦП*.

Итак, пусть аналоговый полосовой сигнал имеет спектр, расположенный в диапазоне между частотами  $f_1$  и  $f_2$  (рис. 3.15, а; не будем забывать и о симметричной

половине спектра в области отрицательных частот). При обычном подходе частота дискретизации должна превышать  $2f_2$ . В результате возникающие при дискретизации сдвинутые копии спектра сигнала расположатся так, как показано на рис. 3.15, б. Однако для точного восстановления сигнала по его дискретным отсчетам нам необходимо лишь обеспечить отсутствие перекрытия сдвинутых копий спектра. В данном случае это дает дополнительные возможности для выбора частоты дискретизации (рис. 3.15, в). Восстановление сигнала в данном случае, естественно, должно производиться с помощью полосового фильтра, а не ФНЧ (АЧХ восстанавливающих фильтров показаны на рис. 3.15, б, в пунктиром).

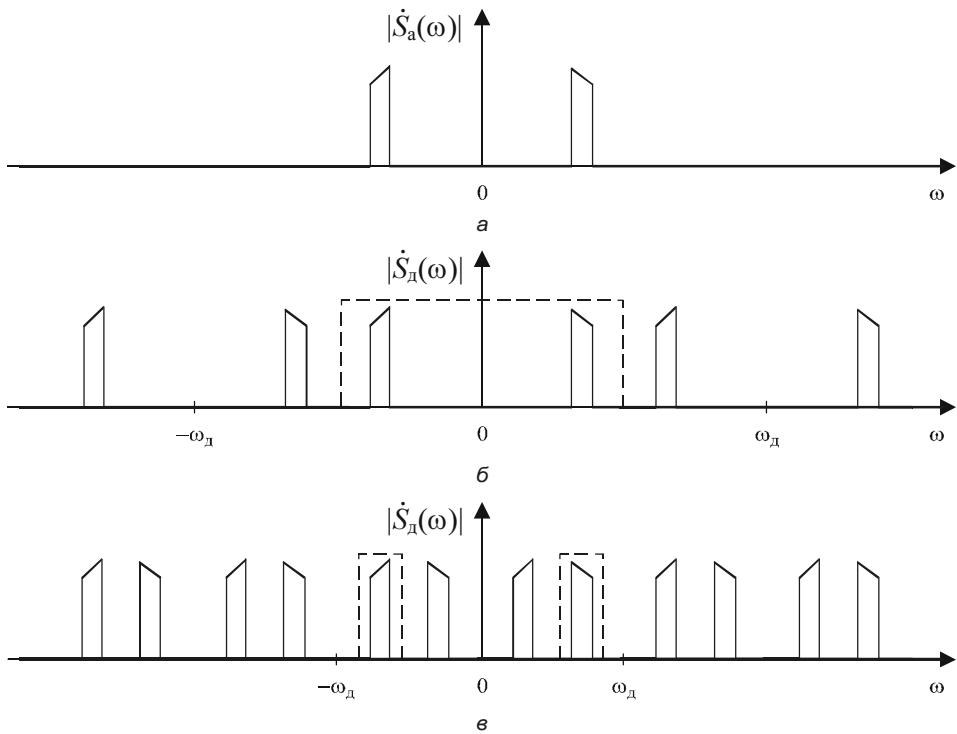


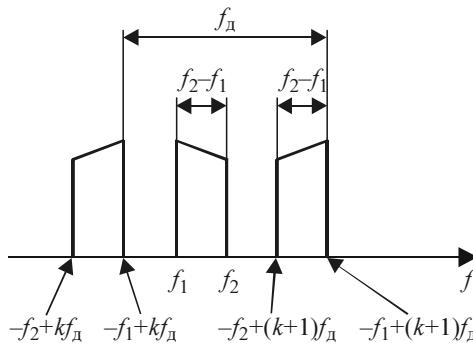
Рис. 3.15. Спектр аналогового сигнала (а), дискретного сигнала при обычной дискретизации (б) и при субдискретизации (в)

Определим условия, при которых возможна дискретизация сигнала таким образом. При некотором целом  $k$  зеркальная половина спектра должна оказаться расположена между  $k$ -й и  $(k + 1)$ -й сдвинутыми копиями спектра (рис. 3.16).

Из этого рисунка сразу же видна нижняя граница для частоты дискретизации:  $f_d > 2(f_2 - f_1)$ . Таким образом, в отличие от случая квадратурной дискретизации, в данном варианте частота дискретизации ограничена снизу удвоенной шириной спектра сигнала. Кроме того, из рис. 3.16 мы получаем пару неравенств:

$$-f_1 + k f_d < f_1,$$

$$-f_2 + (k + 1)f_d > f_2.$$



**Рис. 3.16.** Условия, необходимые для субдискретизации сигнала

Выразив из них частоту дискретизации, получаем для нее двойное неравенство:

$$\frac{2f_2}{k+1} < f_d < \frac{2f_1}{k}. \quad (3.16)$$

Чтобы этот диапазон существовал, правая часть неравенства должна быть больше левой:

$$\frac{2f_1}{k} > \frac{2f_2}{k+1}.$$

Отсюда получаем ограничение на возможные значения  $k$ :

$$k < \frac{f_1}{f_2 - f_1}. \quad (3.17)$$

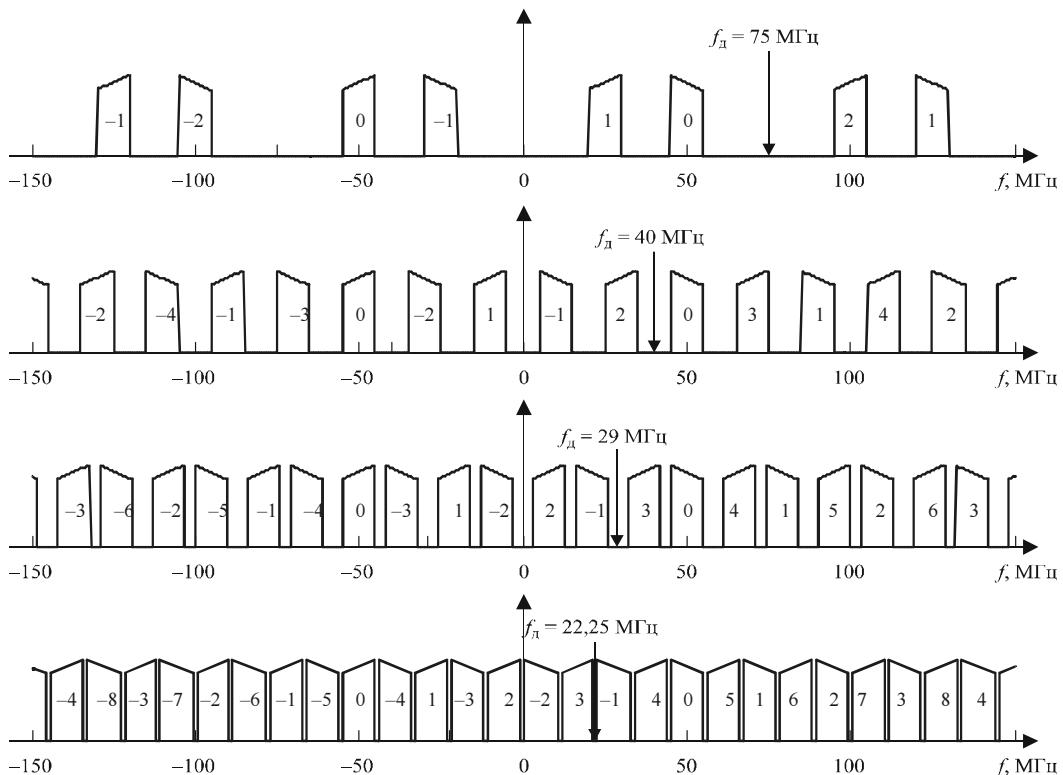
В качестве примера рассмотрим дискретизацию сигнала со средней частотой 50 МГц и шириной полосы 10 МГц. Границы занимаемой сигналом полосы частот в этом случае равны  $f_1 = 45$  МГц и  $f_2 = 55$  МГц, а минимальное значение  $k$ , согласно (3.17), определяется следующим неравенством:

$$k < \frac{f_1}{f_2 - f_1} = \frac{45}{10} = 4,5.$$

Для удовлетворяющих неравенству целочисленных значений  $k = 1\dots4$ , согласно (3.16), имеем следующие диапазоны возможных частот дискретизации:

- $k = 1$ :  $f_d = \frac{2 \cdot 55}{2} \dots \frac{2 \cdot 45}{1}$  МГц = 55...90 МГц;
- $k = 2$ :  $f_d = \frac{2 \cdot 55}{3} \dots \frac{2 \cdot 45}{2}$  МГц = 36,67...45 МГц;
- $k = 3$ :  $f_d = \frac{2 \cdot 55}{4} \dots \frac{2 \cdot 45}{3}$  МГц = 27,5...30 МГц;
- $k = 4$ :  $f_d = \frac{2 \cdot 55}{5} \dots \frac{2 \cdot 45}{4}$  МГц = 22...22,5 МГц.

На рис. 3.17 показаны спектры дискретизированного сигнала, получающиеся при выборе частот дискретизации, равных 75 МГц ( $k = 1$ ), 40 МГц ( $k = 2$ ), 29 МГц ( $k = 3$ ), 22,25 МГц ( $k = 4$ ). Внутри полос частот, занимаемых копиями спектра, подписаны номера этих копий (индекс  $n$  из формулы (3.7)).



**Рис. 3.17.** Спектр субдискретизированного сигнала при выборе частот дискретизации, соответствующих различным значениям  $k$  (сверху вниз):

$$\begin{aligned} k &= 1, \quad f_d = 75 \text{ МГц}; \\ k &= 2, \quad f_d = 40 \text{ МГц}; \\ k &= 3, \quad f_d = 29 \text{ МГц}; \\ k &= 4, \quad f_d = 22,25 \text{ МГц} \end{aligned}$$

#### ЗАМЕЧАНИЕ

Значение  $k = 0$ , исходя из (3.16), дает диапазон частоты дискретизации от  $2f_2$  до бесконечности и, таким образом, соответствует обычной дискретизации сигнала согласно теореме Котельникова.

Если после субдискретизации восстановить аналоговый сигнал обычным образом (с помощью идеального ФНЧ с частотой среза, равной частоте Найквиста), получится полосовой сигнал с той же или комплексно-сопряженной комплексной огибающей, что у исходного сигнала, но с более низкой средней частотой  $\omega'_0$ , определяемой следующим образом:

- если  $k$  четно, то  $\omega'_0 = \omega_0 - \frac{k}{2}\omega_d$  и комплексная огибающая сигнала не меняется;
- если  $k$  нечетно, то  $\omega'_0 = \frac{k+1}{2}\omega_d - \omega_0$ , а комплексная огибающая сигнала становится комплексно-сопряженной.

## Z-преобразование

Удобным способом анализа дискретных последовательностей является  $z$ -преобразование ( $z$ -transform). Смысл его заключается в том, что последовательности чисел  $\{x(k)\}$  ставится в соответствие функция комплексной переменной  $z$ , определяемая следующим образом:

$$X(z) = \sum_{k=-\infty}^{\infty} x(k)z^{-k}. \quad (3.18)$$

Разумеется, функция  $X(z)$  определена только для тех значений  $z$ , при которых ряд (3.18) сходится.

$Z$ -преобразование играет для дискретных сигналов и систем такую же роль, как преобразование Лапласа — для аналоговых. Определяющим при этом является тот факт, что, как мы увидим в главе 4,  $z$ -преобразование импульсной характеристики дискретной системы является дробно-рациональной функцией переменной  $z$ .

## Примеры вычисления z-преобразования

Вычислим  $z$ -преобразование для некоторых часто встречающихся на практике дискретных сигналов.

### Единичная импульсная функция

Единичная импульсная функция является дискретным аналогом дельта-функции (см. разд. "Классификация сигналов" главы 1) и представляет собой одиночный отсчет с единичным значением:

$$x_0(k) = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0. \end{cases} \quad (3.19)$$

Расчет его  $z$ -преобразования не представляет сложности:

$$X_0(z) = \sum_{k=-\infty}^{\infty} x_0(k)z^{-k} = 1 \cdot z^{-0} = 1.$$

Функция  $X_0(z)$  сходится на всей комплексной плоскости.

## Единичный скачок

Дискретный единичный скачок по смыслу полностью соответствует своему аналоговому прообразу (см. разд. "Классификация сигналов" главы 1):

$$x(k) = \begin{cases} 0, & k < 0, \\ 1, & k \geq 0. \end{cases}$$

Используя определение  $z$ -преобразования (3.18), получаем

$$X(z) = \sum_{k=-\infty}^{\infty} x(k)z^{-k} = \sum_{k=0}^{\infty} 1 \cdot z^{-k}. \quad (3.20)$$

Ряд (3.20) является суммой бесконечной геометрической прогрессии с первым членом  $1 \cdot z^{-0} = 1$  и знаменателем  $z^{-1}$ . Как известно, такой ряд сходится при  $|z^{-1}| < 1$ , т. е. при  $|z| > 1$ , и его сумма равна

$$X(z) = \frac{1}{1 - z^{-1}}. \quad (3.21)$$

### ЗАМЕЧАНИЕ

Можно записать результат и в виде  $X(z) = z/(z - 1)$ , но в теории дискретных сигналов принято использовать отрицательные степени  $z$ .

## Дискретная экспоненциальная функция

Дискретная экспоненциальная функция определяется следующим образом:

$$x(k) = \begin{cases} 0, & k < 0, \\ a^k, & k \geq 0. \end{cases}$$

Для вычисления  $z$ -преобразования нужно вычислить сумму следующего ряда:

$$X(z) = \sum_{k=-\infty}^{\infty} x(k)z^{-k} = \sum_{k=0}^{\infty} a^k z^{-k} = \sum_{k=0}^{\infty} (a^{-1}z)^{-k}.$$

Как и в предыдущем случае, этот ряд представляет собой сумму геометрической прогрессии. Первый член равен 1, знаменатель равен  $az^{-1}$ . Таким образом, ряд сходится при  $|az^{-1}| < 1$ , т. е. при  $|z| > |a|$ , и его сумма равна

$$X(z) = \frac{1}{1 - az^{-1}}. \quad (3.22)$$

## Дискретная затухающая синусоида

Последняя из рассматриваемых здесь дискретных последовательностей представляет собой отсчеты синусоиды с произвольными частотой и начальной фазой и экспоненциально меняющейся амплитудой:

$$x(k) = a^k \cos(\omega k + \phi).$$

Для вычисления  $z$ -преобразования можно представить косинус по формуле Эйлера в виде полусуммы двух комплексных экспонент, а потом воспользоваться уже готовым результатом (3.22):

$$X(z) = \frac{\cos \varphi - a \cos(\omega - \varphi) z^{-1}}{1 - 2a \cos \omega z^{-1} + a^2 z^{-2}}.$$

Так же, как и в случае дискретной экспоненты, ряд сходится при  $|z| > |a|$ .

## Связь $z$ -преобразования с преобразованиями Лапласа и Фурье

Дискретное  $z$ -преобразование очень просто связано с преобразованиями Лапласа и Фурье. Рассмотрим последовательность, определенную при  $k \geq 0$ , и сопоставим ей временной сигнал в виде набора дельта-функций:

$$s(t) = \sum_{k=0}^{\infty} x(k) \delta(t - kT), \quad (3.23)$$

где  $T$  — интервал дискретизации. Преобразование Лапласа для сигнала (3.23) равно:

$$S(p) = \int_0^{\infty} s(t) e^{-pt} dt = \int_0^{\infty} \sum_{k=0}^{\infty} x(k) \delta(t - kT) e^{-pt} dt = \sum_{k=0}^{\infty} x(k) \int_0^{\infty} \delta(t - kT) e^{-pt} dt.$$

Воспользовавшись фильтрующим свойством дельта-функции (см. формулу (1.2) в разд. "Классификация сигналов" главы 1), получим

$$S(p) = \sum_{k=0}^{\infty} x(k) e^{-pkT}.$$

Эта формула переходит в формулу (3.18), определяющую  $z$ -преобразование, если выполнить подстановку  $z = e^{pT}$ .

Таким образом, взаимное соответствие между  $z$ -преобразованием  $X(z)$  и преобразованием Лапласа  $S(p)$  описывается следующим образом:

$$X(z) = S\left(\frac{1}{T} \ln z\right), \quad S(p) = X(e^{pT}).$$

Похожими формулами описывается и связь  $z$ -преобразования  $X(z)$  с преобразованием Фурье  $\dot{S}(\omega)$  (заметим, что при рассмотрении этой связи нет необходимости считать последовательность односторонней):

$$X(z) = \dot{S}\left(\frac{1}{jT} \ln z\right), \quad \dot{S}(\omega) = X(e^{j\omega T}). \quad (3.24)$$

## Свойства z-преобразования

Тесная связь z-преобразования с преобразованиями Фурье и Лапласа обуславливает и подобие свойств этих преобразований. Однако имеется и некоторая специфика, возникающая из-за дискретного характера рассматриваемых сигналов.

### Линейность

Z-преобразование, согласно определению (3.18), является линейной комбинацией отсчетов последовательности, поэтому оно подчиняется принципу суперпозиции:

$$\begin{aligned} \text{если } & \{x_1(k)\} \leftrightarrow X_1(z) \text{ и } \{x_2(k)\} \leftrightarrow X_2(z), \\ \text{то } & \{ax_1(k) + bx_2(k)\} \leftrightarrow aX_1(z) + bX_2(z). \end{aligned}$$

### Задержка

Если z-преобразование последовательности  $\{x(k)\}$  равно  $X(z)$ , то z-преобразование последовательности, задержанной на  $k_0$  тактов ( $y(k) = x(k - k_0)$ ), будет иметь вид

$$\begin{aligned} Y(z) &= \sum_{k=-\infty}^{\infty} y(k)z^{-k} = \sum_{k=-\infty}^{\infty} x(k - k_0)z^{-k} = \\ &= z^{-k_0} \sum_{k=-\infty}^{\infty} x(k - k_0)z^{-(k-k_0)} = z^{-k_0} \sum_{n=-\infty}^{\infty} x(n)z^{-n} = X(z)z^{-k_0}. \end{aligned}$$

Таким образом, при задержке последовательности на  $k_0$  тактов необходимо умножить ее z-преобразование на  $z^{-k_0}$ . Множитель  $z^{-k_0}$  является *оператором задержки* дискретной последовательности на  $k_0$  тактов.

### Свертка

*Свертка* двух бесконечных дискретных последовательностей  $\{x_1(k)\}$  и  $\{x_2(k)\}$  определяется следующим образом:

$$y(k) = \sum_{n=-\infty}^{\infty} x_1(n)x_2(k-n).$$

Вычислим z-преобразование для последовательности  $\{y(k)\}$ :

$$\begin{aligned} Y(z) &= \sum_{k=-\infty}^{\infty} y(k)z^{-k} = \sum_{k=-\infty}^{\infty} \left[ \sum_{n=-\infty}^{\infty} x_1(n)x_2(k-n)z^{-k} \right] = \\ &= \sum_{k=-\infty}^{\infty} \left[ \sum_{n=-\infty}^{\infty} x_1(n)x_2(k-n)z^{-n}z^{-(k-n)} \right] = \\ &= \sum_{n=-\infty}^{\infty} \left[ x_1(n)z^{-n} \underbrace{\sum_{k=-\infty}^{\infty} x_2(k-n)z^{-(k-n)}}_{X_2(z)} \right] = \\ &= X_2(z) \sum_{n=-\infty}^{\infty} x_1(n)z^{-n} = X_1(z)X_2(z). \end{aligned} \tag{3.25}$$

Итак, свертка дискретных последовательностей соответствует произведение их  $z$ -преобразований.

### ЗАМЕЧАНИЕ

Рассматриваемая здесь свертка бесконечных дискретных последовательностей называется *линейной сверткой*; ее не следует путать с *круговой сверткой периодических последовательностей*, речь о которой пойдет при описании свойств дискретного преобразования Фурье в главе 5.

## Чередование знаков сигнала

Пусть у элементов исходной последовательности с нечетными номерами изменен знак:  $\{x(0), -x(1), x(2), -x(3), x(4), \dots\}$ . Элементы такой последовательности можно записать как  $y(k) = x(k) \cdot (-1)^{-k}$ , поэтому найти ее  $z$ -преобразование оказывается очень легко:

$$Y(z) = \sum_{k=-\infty}^{\infty} x(k)(-1)^{-k} z^{-k} = \sum_{k=-\infty}^{\infty} x(k)(-z)^{-k} = X(-z).$$

Итак, чередование знаков сигнала приводит к смене знака у аргумента  $z$ -преобразования.

Если изменить знаки у элементов последовательности с *четными* номерами, это будет соответствовать простому изменению знака у последовательности, рассмотренной ранее, а  $z$ -преобразование, соответственно, окажется равным  $-X(-z)$ .

## Инвертирование последовательности во времени

Пусть последовательность  $\{y(k)\}$  связана с последовательностью  $\{x(k)\}$  следующим образом:

$$y(k) = x(-k).$$

Тогда  $z$ -преобразование последовательности  $\{y(k)\}$  будет иметь вид

$$Y(z) = \sum_{k=-\infty}^{\infty} x(-k)z^{-k} = \sum_{m=-\infty}^{\infty} x(m)z^m = \sum_{m=-\infty}^{\infty} x(m)(z^{-1})^{-m} = X(z^{-1}).$$

Итак, инверсия последовательности отсчетов во времени соответствует замене  $z$  на  $1/z$  в формуле ее  $z$ -преобразования.

## Вставка нулей

Пусть последовательность  $\{y(k)\}$  получается из последовательности  $\{x(k)\}$  путем вставки  $N - 1$  нулей между соседними элементами:

$$y(k) = \begin{cases} x(k/N), & k = Nm, \text{ где } m \text{ — любое целое число,} \\ 0, & k \neq Nm. \end{cases}$$

Найдем  $z$ -преобразование:

$$Y(z) = \sum_{k=-\infty}^{\infty} y(k)z^{-k} = \sum_{m=-\infty}^{\infty} x(m)z^{-Nm} = \sum_{m=-\infty}^{\infty} x(m)(z^N)^{-m} = X(z^N).$$

Таким образом, рассматриваемая вставка нулей соответствует замене  $z$  на  $z^N$  в формуле  $z$ -преобразования.

## Обратное $z$ -преобразование

Соответствие между дискретной последовательностью чисел и ее  $z$ -преобразованием является взаимно-однозначным. Формула перехода от  $z$ -преобразования к последовательности чисел называется *обратным  $z$ -преобразованием* и формально записывается следующим образом:

$$x(k) = \frac{1}{j2\pi} \oint X(z) z^{k-1} dz. \quad (3.26)$$

Интеграл в (3.26) берется по произвольному замкнутому контуру, расположенному в области сходимости функции  $X(z)$  и охватывающему все ее полюсы.

Практическое вычисление обратного  $z$ -преобразования чаще производится путем разложения функции  $X(z)$  на простые дроби. Поясним это на несложном примере. Пусть

$$X(z) = \frac{1}{\frac{1}{2}z^{-2} - \frac{3}{2}z^{-1} + 1}.$$

Представим  $X(z)$  в виде суммы простых дробей:

$$X(z) = \frac{1}{(1-z^{-1})(1-\frac{1}{2}z^{-1})} = \frac{2}{1-z^{-1}} - \frac{1}{1-\frac{1}{2}z^{-1}}. \quad (3.27)$$

Из сравнения слагаемых (3.27) с примерами  $z$ -преобразований (3.21) и (3.22) видно, что первое слагаемое соответствует скачку с амплитудой, равной 2, а второе — дискретной показательной функции  $-2^{-k}$ ,  $k \geq 0$ . Итак, искомая последовательность имеет вид:

$$x(k) = \begin{cases} 2 - 2^{-k}, & k \geq 0, \\ 0, & k < 0. \end{cases}$$

### ЗАМЕЧАНИЕ

Чтобы рассчитать обратное  $z$ -преобразование, кроме функции  $X(z)$  нужно задать область ее определения. Результат, полученный в данном примере, соответствует области определения  $|z| > 1$ . При области определения  $|z| < 1/2$  получится последовательность, экспоненциально затухающая в направлении отрицательных номеров отсчетов  $k$ .

## Пространство дискретных сигналов

Для дискретных сигналов можно ввести те же понятия, что были рассмотрены в разд. "Пространство сигналов" главы 1 применительно к сигналам аналоговым. При этом формулы соответствующим образом модифицируются — интегралы превращаются в суммы. Приведем получающиеся при этом основные выражения. Для большей общности в них предполагается, что дискретный сигнал  $x(k)$  является комплексным, а диапазон индексов при суммировании не указывается (этот диапазон может быть как конечным, так и бесконечным):

□  $p$ -метрика и ее частные случаи:

$$d_p(x, y) = \sqrt[p]{\sum_k |x(k) - y(k)|^p},$$

$$d_1(x, y) = \sum_k |x(k) - y(k)|,$$

$$d_2(x, y) = \sqrt{\sum_k |x(k) - y(k)|^2},$$

$$d_\infty(x, y) = \max_k |x(k) - y(k)|;$$

□  $p$ -норма и ее частные случаи:

$$\|\mathbf{x}\|_p = \sqrt[p]{\sum_k |x(k)|^p},$$

$$\|\mathbf{x}\|_1 = \sum_k |x(k)|,$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_k |x(k)|^2},$$

$$\|\mathbf{x}\|_\infty = \max_k |x(k)|;$$

□ скалярное произведение:

$$(\mathbf{x}, \mathbf{y}) = \sum_k x(k)y^*(k).$$

## Дискретные случайные сигналы

Когда дискретизации подвергается случайный процесс, получаемая последовательность отсчетов будет зависеть от конкретной реализации дискретизируемого процесса, и, следовательно, она должна анализироваться статистическими методами.

Что касается одномерной плотности вероятности и связанных с ней статистических характеристик, здесь нет никаких отличий от случая аналогового сигнала — просто возможные одномерные сечения случайного процесса соответствуют моментам

дискретизации:  $t_1 = kT$ , и поэтому для привязки статистических параметров ко времени можно использовать номер отсчета:  $p_x(x, k)$ ,  $m_x(k)$  и т. д.

*Двумерные* сечения дискретного случайного процесса также могут браться только в моменты дискретизации:  $t_1 = kT$ ,  $t_2 = nT$ . Поэтому двумерная плотность вероятности и связанные с ней характеристики случайного процесса зависят от двух номеров отсчетов  $k$  и  $n$ :  $p_x(x_1, x_2, k, n)$ ,  $R_x(k, n)$  и т. д.

В случае *стационарного* в широком смысле случайного процесса одномерные характеристики не зависят от момента времени (номера отсчета), а двумерные зависят лишь от *промежутка* между моментами времени (в дискретном случае — от разности номеров отсчетов  $\Delta k = k - n$ ):

$$\begin{aligned} p_x(x, k) &= p_x(x), \quad m_x(k) = m_x, \quad D_x(k) = D_x, \\ p_x(x_1, x_2, k, n) &= p_x(x_1, x_2, \Delta k), \quad R_x(k, n) = R_x(\Delta k). \end{aligned}$$

Таким образом, для стационарного дискретного случайного процесса корреляционная функция является дискретной, т. е. представляет собой последовательность чисел  $\{R_x(\Delta k)\}$ .

В общем случае  $R_x(k, n) = R_x^*(n, k)$ ; для вещественного случайного процесса  $R_x(k, n) = R_x(n, k)$ , а если процесс еще и стационарный, то  $R_x(-k) = R_x(k)$ .

## Корреляционная матрица

Во многих задачах необходимо рассматривать конечный во времени фрагмент случайного процесса длиной  $N$  отсчетов. В этом случае корреляционная функция  $R_x(k, n)$  может быть представлена в виде матрицы, называемой *корреляционной матрицей* случайного процесса (для простоты будем считать, что случайный процесс имеет нулевое математическое ожидание):

$$\mathbf{R}_x = [R_x(k, n)] = \begin{bmatrix} \overline{x(0)x(0)} & \overline{x(0)x(1)} & \cdots & \overline{x(0)x(N-1)} \\ \overline{x(1)x(0)} & \overline{x(1)x(1)} & \cdots & \overline{x(1)x(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{x(N-1)x(0)} & \overline{x(N-1)x(1)} & \cdots & \overline{x(N-1)x(N-1)} \end{bmatrix}.$$

Черты сверху здесь обозначено усреднение по ансамблю реализаций.

Если процесс нестационарный, все элементы матрицы могут быть различными. В случае стационарного процесса корреляционная матрица полностью определяется своими первыми строкой и столбцом, поскольку вдоль всех диагоналей, параллельных главной, стоят одинаковые элементы:

$$\mathbf{R}_x = \begin{bmatrix} R_x(0) & R_x(1) & \ddots & R_x(N-2) & R_x(N-1) \\ R_x(-1) & R_x(0) & \ddots & R_x(N-3) & R_x(N-2) \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ R_x(-N+2) & R_x(-N+3) & \ddots & R_x(0) & R_x(1) \\ R_x(-N+1) & R_x(-N+2) & \ddots & R_x(-1) & R_x(0) \end{bmatrix}.$$

**ЗАМЕЧАНИЕ**

Матрицы, обладающие таким свойством, называются *матрицами Тэплица (Toeplitz matrix)*. В MATLAB сгенерировать матрицу Тэплица по первым строке и столбцу позволяет функция `toeplitz`.

В случае вещественного случайного процесса, как уже отмечалось,  $R_x(-k) = R_x(k)$ , и корреляционная матрица становится симметричной:

$$\mathbf{R}_x = \begin{bmatrix} R_x(0) & R_x(1) & \ddots & R_x(N-2) & R_x(N-1) \\ R_x(1) & R_x(0) & \ddots & R_x(N-3) & R_x(N-2) \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ R_x(N-2) & R_x(N-3) & \ddots & R_x(0) & R_x(1) \\ R_x(N-1) & R_x(N-2) & \ddots & R_x(1) & R_x(0) \end{bmatrix}. \quad (3.28)$$

Важным свойством матриц вида (3.28) является то, что они всегда являются *неотрицательно определенными*, т. е. все их *собственные числа* вещественны и неотрицательны.

**ЗАМЕЧАНИЕ**

Если стационарный случайный процесс является комплексным,  $R_x(-k) = R_x^*(k)$  и корреляционная матрица является не симметричной, а самосопряженной. Свойство неотрицательной определенности сохраняется и в этом случае.

## Дискретный белый шум

Так называется стационарный дискретный случайный процесс, отсчеты которого некоррелированы друг с другом:

$$R_x(\Delta k) = \begin{cases} \sigma_x^2, & \Delta k = 0, \\ 0, & \Delta k \neq 0. \end{cases}$$

В отличие от случая аналогового белого шума, дисперсия дискретного белого шума не является бесконечной, а потому такой шум является физически реализуемым.

Корреляционная матрица дискретного белого шума будет, очевидно, иметь диагональную структуру:

$$\mathbf{R}_x = \sigma_x^2 \mathbf{I},$$

где  $\mathbf{I}$  — единичная матрица.

## Дискретные сигналы в MATLAB

Дискретный сигнал представляет собой последовательность чисел, а потому в MATLAB он представляется в виде *вектора*. Если необходимо реализовать многоканальную обработку сигналов, для этого удобно использовать второе измерение, представив набор сигналов в виде *матрицы*. Многоканальная обработка поддерживается многими функциями MATLAB.

Если сигнал одномерный, то в большинстве случаев функции MATLAB правильно обработают его при любой ориентации вектора — как в виде строки, так и в виде столбца. Однако в многоканальном случае, когда входной сигнал представлен в виде матрицы, обработка производится *по столбцам*.

Таким образом, столбцы матрицы трактуются как сигналы разных каналов, а строки — как отдельные векторные отсчеты многоканального сигнала. Для избежания путаницы рекомендуется и в одноканальном случае задавать сигналы в виде столбцов.

Отсчеты дискретного сигнала могут быть получены двумя путями. Первый вариант — расчет значений сигнала (*моделирование сигнала*), второй — получение сигнала извне путем считывания его *записи*. Работу с записями сигналов в виде WAV-файлов мы рассмотрим ближе к концу главы, а сейчас обсудим моделирование сигналов.

## Расчет временных функций

Аналоговый сигнал, как уже говорилось, с математической точки зрения представляет собой *функцию* (как правило — функцию времени), и при его дискретизации мы получаем отсчеты, являющиеся значениями этой функции, вычисленными в дискретные моменты времени. Поэтому для расчета дискретизированного сигнала необходимо, прежде всего, сформировать вектор дискретных значений времени. Для этого удобно задать значение частоты дискретизации  $F_s$  (*sampling frequency*) и использовать обратную величину в качестве шага временного ряда:

```
>> Fs = 8e3; % частота дискретизации 8 кГц
>> t = 0:1/Fs:1; % одна секунда дискретных значений времени
>> t = t'; % преобразуем строку в столбец
```

Сформировав вектор опорных значений времени, можно вычислять значения сигнала, используя этот вектор в соответствующих формулах. Большинство математических функций MATLAB обрабатывают векторный аргумент поэлементно, так что с этим проблем не возникает. Однако следует помнить о том, что операции умножения, деления и возвведения в степень в MATLAB имеют матричный смысл, поэтому при расчете одномерных функций времени следует использовать поэлементные версии этих операций ( $.*$ ,  $./$  и  $.^$ ). Приведем несколько примеров:

```
>> A = 2; % амплитуда — два вольта
>> f0 = 1e3; % частота — 1 кГц
>> phi = pi/4; % начальная фаза — 45°
>> s1 = A * cos(2*pi*f0*t + phi); % гармонический сигнал
>> alpha = 1e3; % скорость затухания
>> s2 = exp(-alpha*t) .* s1; % затухающая синусоида
```

Для визуализации дискретных сигналов могут использоваться различные графические средства MATLAB в зависимости от конкретной ситуации. Часто вполне допустимым является соединение дискретных отсчетов линиями, т. е. построение графика с помощью функции *plot* (рис. 3.18, сверху слева). При этом хорошо видна

общая форма сигнала (фактически в этом случае мы получаем график аналогового сигнала, полученного путем линейной интерполяции), но незаметны отсчетные точки. Если необходимо отобразить именно их, можно отказаться от соединения точек линиями (рис. 3.18, сверху справа).

Помимо функции `plot` существуют другие графические функции, специально предназначенные для отображения дискретных последовательностей. Функция `stem` изображает сигнал в виде "стебельков" (рис. 3.18, снизу слева), а функция `stairs` — в ступенчатом виде (рис. 3.18, снизу справа). В последнем случае изображается аналоговый сигнал с кусочно-постоянной интерполяцией (интерполяцией нулевого порядка). Такой сигнал получается на выходе ЦАП при отсутствии сглаживающего фильтра. Вот последовательность команд MATLAB, при помощи которой был получен рис. 3.18:

```
>> subplot(2, 2, 1); plot(s2(1:50))
>> subplot(2, 2, 2); plot(s2(1:50), '.')
>> subplot(2, 2, 3); stem(s2(1:50))
>> subplot(2, 2, 4); stairs(s2(1:50))
```

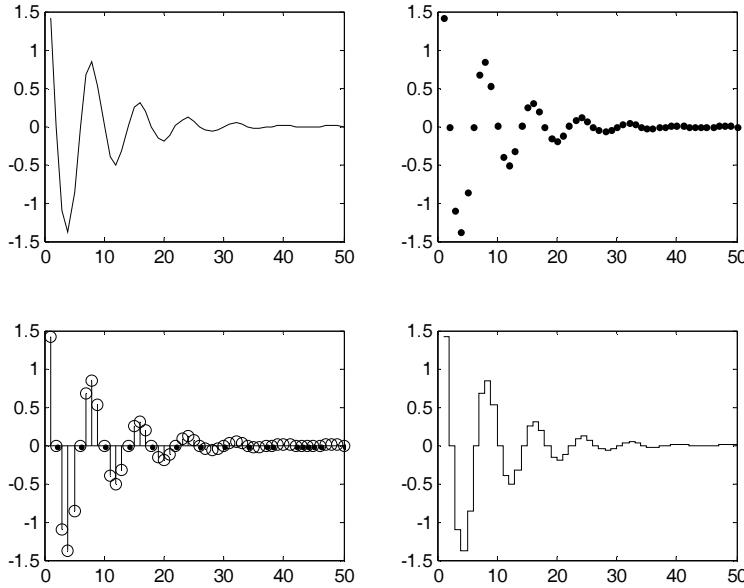


Рис. 3.18. Различные формы представления графиков дискретного сигнала

### ЗАМЕЧАНИЕ

Еще один способ визуализации сигналов обеспечивает функция `strips`. Она будет рассмотрена далее в разд. "Чтение WAV-файлов" этой главы.

Горизонтальная ось на приведенных графиках проградуирована в номерах отсчетов. Чтобы показать на этой оси значения времени, при вызове графических функ-

ций следует использовать два параметра, передав в первом из них соответствующий временной вектор, например, так:

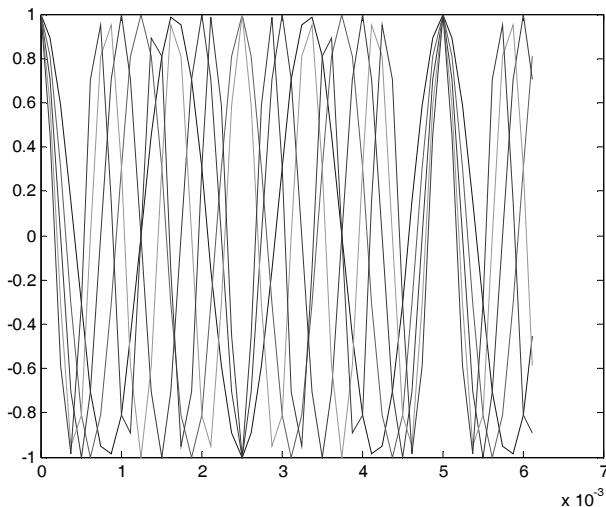
```
plot(t(1:50), s2(1:50))
```

Если необходимо сгенерировать многоканальный сигнал, каналы которого описываются одной и той же формулой, но с разными числовыми значениями параметров, для этого можно эффективно использовать средства матричных операций MATLAB. Простейший пример — генерация набора синусоид с заданными частотами (в приводимом далее коде используется вектор  $t$ , сформированный ранее):

```
>> f = [600 800 1000 1200 1400]; % вектор частот (строка!)
>> s3 = cos(2*pi*t*f); % пятиканальный сигнал
```

Здесь *столбец* значений времени  $t$  умножается на *строку* частот  $f$ . В результате получается *матрица*, содержащая все необходимые значения произведения времени и частоты. Далее эта матрица подвергается поэлементным преобразованиям (умножение на  $2\pi$  и вычисление косинуса). Результат вычислений показан на рис. 3.19:

```
>> plot(t(1:50), s3(1:50,:))
```



**Рис. 3.19.** Многоканальный сигнал

### ЗАМЕЧАНИЕ

Напомним, что при визуализации матрицы функция `plot` строит графики именно по столбцам.

Если вычисление значений сигнала производится однократно, необходимые вычисления можно выполнить в ходе интерактивного сеанса работы либо описать в тексте MATLAB-программы. Если же в ходе решения задачи необходимо повторно обращаться к вычислениям, целесообразно оформить их в виде функции

(см. разд. "Программирование" приложения 1). При этом следует иметь в виду, что для использования многочисленных численных алгоритмов, эффективно реализованных в MATLAB, функция для расчета значений сигнала должна быть способна принимать *векторный* аргумент, содержащий *набор* значений времени.

## Кусочные зависимости

Довольно часто возникает необходимость моделирования отсчетов сигнала, который для разных интервалов времени описывается разными формулами. Простейший случай такого рода — моделирование импульсов конечной длительности, значения которых описываются одной или несколькими формулами в пределах заданного интервала, а в остальное время равны нулю.

В MATLAB возможно несколько способов задания кусочных функций. Первый вариант — воспользоваться тем, что операции сравнения возвращают 1 при выполнении неравенства и 0 в противном случае, причем в случае векторного аргумента возвращается вектор результатов сравнения. Зададим таким образом односторонний экспоненциальный импульс (см. рис. 1.18):

```
s = A * exp(-alpha * t) .* (t >= 0);
```

Прямоугольный импульс, центрированный относительно начала отсчета времени (см. рис. 1.10):

```
s = A * (abs(t) <= T/2);
```

Несимметричный треугольный импульс (см. рис. 1.14):

```
s = A * t / T .* (t >= 0) .* (t <= T);
```

В приведенных примерах следует обратить внимание на использование оператора `*` для поэлементного перемножения векторов.

Недостаток данного способа состоит в том, что значения сигнала сначала вычисляются по заданной формуле для *всех* значений аргумента, и лишь затем "вырезается" нужная область. Помимо возрастания времени вычислений это может привести к выдаче сообщения об ошибке — например, в первом из приведенных примеров при вычислении экспоненты для большого отрицательного значения аргумента `t` может произойти переполнение.

Второй способ — выполнять вычисления только для тех моментов времени, для которых это действительно необходимо. Выделение набора элементов вектора `t`, для которых выполняется заданное условие, производится с помощью логической маски, создаваемой с помощью операторов сравнения. В этом случае следует принять во внимание два аспекта:

- необходимо заранее, до вычислений, создать вектор сигнала такого же размера, как вектор моментов времени `t` (обычно этот вектор заполняют нулями);
- поскольку логическую маску придется использовать несколько раз, целесообразно сохранить ее в переменной.

Покажем, как данным способом генерируются те же три кусочно-заданных сигнала.

**Односторонний экспоненциальный импульс:**

```
% заполняем вектор сигнала нулями
s = zeros(size(t));
% формируем маску для неотрицательных элементов вектора t
mask = (t >= 0);
% рассчитываем сигнал только в нужных точках
s(mask) = A * exp(-alpha * t(mask));
```

Прямоугольный видеоимпульс (поскольку для расчета значений такого импульса не требуется вычислений с вектором  $t$ , переменную  $mask$  можно не создавать — она понадобилась бы лишь один раз):

```
s = zeros(size(t));
s(abs(t) <= T/2) = A;
```

**Несимметричный треугольный импульс:**

```
s = zeros(size(t));
mask = (t >= 0) & (t <= T);
s(mask) = A * t(mask) / T;
```

Как видите, команда в данном случае нужно больше, но зато при этом не будет выполняться лишних вычислений и не появятся упомянутые выше сообщения об ошибках.

## Функции генерации одиночных импульсов

В пакете Signal Processing имеется ряд функций, генерирующих часто встречающиеся на практике непериодические сигналы:

- `rectpuls` — прямоугольный импульс;
- `tripuls` — треугольный импульс;
- `sinc` — импульс вида  $\sin(\pi t)/(\pi t)$ ;
- `gauspuls` — радиоимпульс с гауссовой огибающей;
- `pulstran` — последовательность из конечного числа импульсов произвольной формы.

Далее эти функции рассматриваются более подробно.

### Прямоугольный импульс

Для формирования одиночного прямоугольного импульса с единичной амплитудой служит функция `rectpuls`:

```
y = rectpuls(t, width)
```

Здесь  $t$  — вектор значений времени,  $width$  — ширина (длительность) импульса.

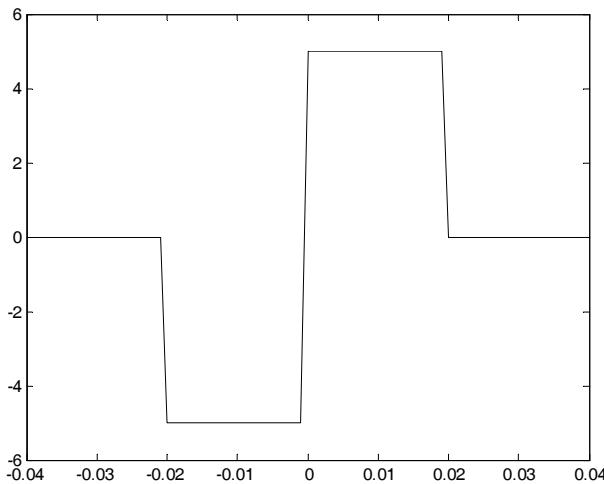
Возвращаемый результат  $y$  — вектор рассчитанных значений сигнала, определяемых по следующей формуле:

$$y = \begin{cases} 1, & -\frac{\text{width}}{2} \leq t < \frac{\text{width}}{2}, \\ 0, & t < -\frac{\text{width}}{2}, \quad t \geq \frac{\text{width}}{2}. \end{cases}$$

Параметр `width` можно опустить, при этом его значение по умолчанию равно 1 и функция `rectpuls` производит результат, соответствующий математической функции `rect`.

В качестве примера сформируем пару разнополярных прямоугольных импульсов с амплитудой 5 В и длительностью 20 мс каждый, расположенных справа и слева от начала отсчета времени. Частоту дискретизации выберем равной 1 кГц. Результат показан на рис. 3.20:

```
>> Fs = 1e3; % частота дискретизации
>> t = -40e-3:1/Fs:40e-3; % дискретное время
>> T = 20e-3; % длительность импульсов
>> A = 5; % амплитуда
>> s = -A * rectpuls(t+T/2, T) + A * rectpuls(t-T/2, T);
>> plot(t, s)
>> ylim([-6 6])
```



**Рис. 3.20.** Сигнал, сформированный с использованием функции `rectpuls`

## Треугольный импульс

Для формирования одиночного треугольного импульса с единичной амплитудой служит функция `tripuls`:

```
y = tripuls(t, width, skew)
```

Здесь `t` — вектор значений времени, `width` — ширина (длительность) импульса, `skew` — коэффициент асимметрии импульса, определяющий положение его вершины.

ны. Пик импульса расположен при  $t = \text{width} * \text{skew} / 2$ . Параметр `skew` должен лежать в диапазоне от  $-1$  до  $1$ .

Возвращаемый результат `y` — вектор рассчитанных значений сигнала, определяемых по следующей формуле:

$$y = \begin{cases} \frac{2t + \text{width}}{\text{width}(\text{skew} + 1)}, & -\frac{\text{width}}{2} \leq t < \frac{\text{width} \cdot \text{skew}}{2}, \\ \frac{2t - \text{width}}{\text{width}(\text{skew} - 1)}, & \frac{\text{width} \cdot \text{skew}}{2} \leq t < \frac{\text{width}}{2}, \\ 0, & |t| > \frac{\text{width}}{2}. \end{cases}$$

Параметры `skew` или `skew` и `width` можно опустить, при этом используются их значения по умолчанию: `skew = 0` (симметричный импульс) и `width = 1`.

В качестве примера сформируем симметричный трапециевидный импульс с амплитудой 10 В и размерами верхнего и нижнего оснований 20 и 60 мс соответственно. Частоту дискретизации выберем равной 1 кГц. Результат показан на рис. 3.21:

```
>> Fs = 1e3; % частота дискретизации
>> t = -50e-3:1/Fs:50e-3; % дискретное время
>> A = 10; % амплитуда
>> T1 = 20e-3; % верхнее основание
>> T2 = 60e-3; % нижнее основание
>> s = A*(T2*tripuls(t, T2) - T1*tripuls(t, T1))/(T2-T1);
>> plot(t, s)
```

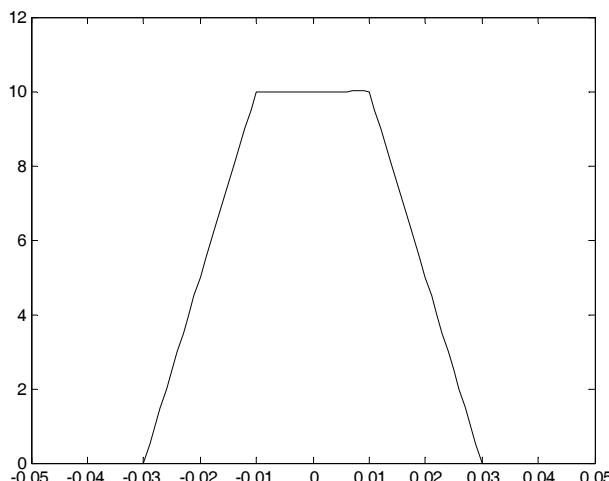


Рис. 3.21. Сигнал, сформированный с использованием функции `tripuls`

## Импульс с ограниченной полосой частот

Для формирования сигнала, имеющего прямоугольный, т. е. ограниченный по частоте спектр, служит функция `sinc`:

$$y = \text{sinc}(t)$$

Единственным входным параметром является вектор значений времени  $t$ .

Возвращаемый результат  $y$  — вектор рассчитанных значений сигнала, определяемых по следующей формуле:

$$y = \frac{\sin(\pi x)}{\pi x}.$$

### **ВНИМАНИЕ!**

В отечественной литературе под функцией `sinc` чаще всего понимается выражение  $\sin(x)/x$  — без умножения аргумента на  $\pi$ . Средствами MATLAB такая функция может быть рассчитана как `sinc(x/pi)`.

Спектральная функция сигнала, генерируемого функцией `sinc`, имеет прямоугольный вид:

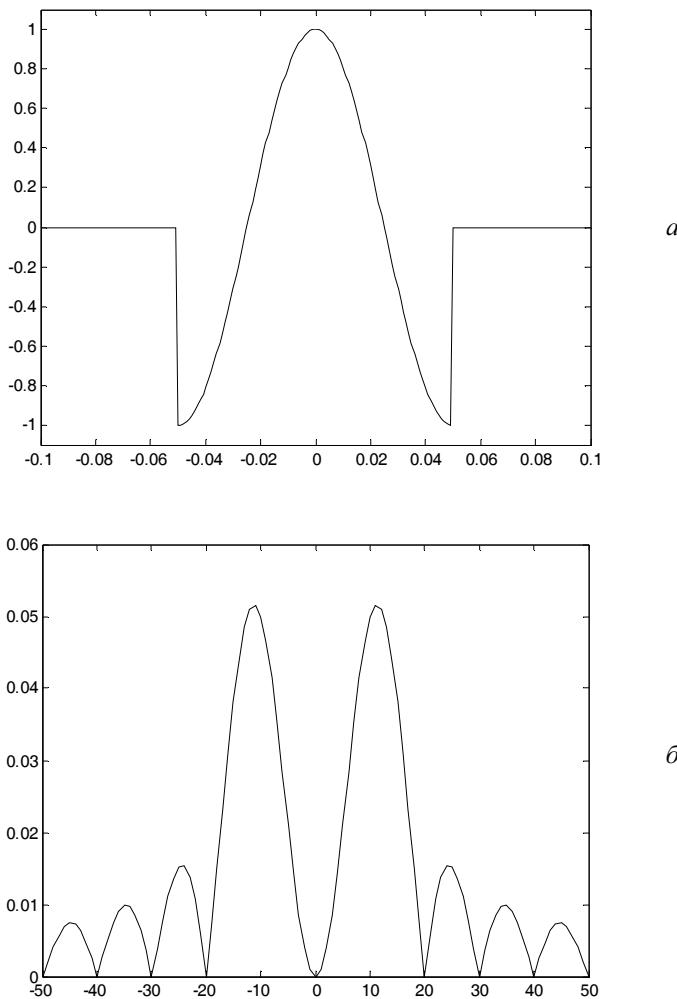
$$\dot{Y}(\omega) = \begin{cases} 1, & |\omega| < \pi, \\ 0, & |\omega| > \pi. \end{cases}$$

В качестве примера построим с помощью функции `sinc` график амплитудного спектра очень короткого радиоимпульса, на длительности которого укладывается лишь один период синусоидального заполнения. Согласно свойствам преобразования Фурье (см. главу 1) спектр такого сигнала должен представлять собой сумму двух спектров прямоугольного импульса, сдвинутых на величину частоты заполнения в сторону положительных и отрицательных частот. Спектр прямоугольного импульса, в свою очередь, описывается именно функцией `sinc` (см. формулу (1.20) в разд. "Примеры расчета преобразования Фурье" главы 1). Результат показан на рис. 3.22:

```
>> Fs = 1e3; % частота дискретизации
>> t = -0.1:1/Fs:0.1; % дискретное время
>> f0 = 10; % частота заполнения
>> T = 1/f0; % длительность радиоимпульса
>> s = rectpuls(t, T) .* cos(2*pi*f0*t); % радиоимпульс
>> f = -50:50; % вектор частот для расчета спектра
>> sp = T/2 * (sinc((f-f0)*T) + sinc((f+f0)*T));
>> plot(t, s) % график сигнала
>> ylim([-1.1 1.1])
>> figure
>> plot(f, abs(sp)) % график амплитудного спектра
```

Как видите, из-за наложения друг на друга "хвостов" функции `sinc` спектр оказывается существенно несимметричным относительно частоты заполнения радиоим-

пульса. Более подробно этот эффект будет обсуждаться в главе 8, посвященной модуляции.



**Рис. 3.22.** Короткий радиоимпульс (*а*) и его амплитудный спектр (*б*), построенный с помощью функции `sinc`

### Гауссов радиоимпульс

Для формирования одиночного радиоимпульса с гауссовой огибающей и единичной амплитудой служит функция `gauspuls`:

`y = gauspuls(t, fc, bw, bwr)`

Здесь `t` — вектор значений времени, `fc` — несущая частота в герцах, `bw` — относительная ширина спектра (ширина спектра, деленная на несущую частоту), `bwr` — уровень (в децибелах), по которому производится измерение ширины спектра.

Возвращаемый результат  $y$  — вектор рассчитанных значений сигнала, определяемых по следующей формуле:

$$y = \exp(-at^2) \cos(2\pi f_c t). \quad (3.29)$$

Коэффициент  $a$  управляет длительностью импульса и, соответственно, шириной его спектра. Сигнал (3.29) имеет спектральную функцию, равную (см. разд. "Примеры расчета преобразования Фурье" главы I)

$$\dot{S}(\omega) = \frac{1}{2} \sqrt{\frac{\pi}{a}} \left( \exp\left(-\frac{(\omega + 2\pi f_c)^2}{4a}\right) + \exp\left(-\frac{(\omega - 2\pi f_c)^2}{4a}\right) \right).$$

Если  $f_c \gg \sqrt{a}$ , можно пренебречь наложением "хвостов" сдвинутых копий спектра. Тогда параметр  $a$  связан с относительной шириной спектра и уровнем (в децибелах), по которому она определяется, следующим образом:

$$a = -\frac{5(2\pi f_c \cdot \text{bw})^2}{\text{bwr} \cdot \ln 10}.$$

Параметры  $\text{bwr}$ ,  $\text{bw}$  и  $f_c$  можно опустить, при этом используются их значения по умолчанию:  $\text{bwr} = -6$  дБ,  $\text{bw} = 0,5$  и  $f_c = 1000$  Гц.

При вызове функции `gauspuls` можно использовать от одного до трех выходных параметров:

```
[y, yq, ye] = gauspuls(...)
```

В втором выходном параметре  $yq$  функция возвращает *квадратурное (quadrature) дополнение* для рассчитанного радиоимпульса  $y$ . Вектор  $yq$  отличается от вектора  $y$  фазовым сдвигом несущего колебания на  $90^\circ$  (подробнее о понятии квадратурного дополнения в разд. "Преобразование Гильберта" главы I).

В третьем выходном параметре  $ye$  функция возвращает *огибающую (envelope)* сформированного радиоимпульса. Вектор  $ye$  представляет собой первый множитель формулы (3.29) (отсутствует умножение на косинус).

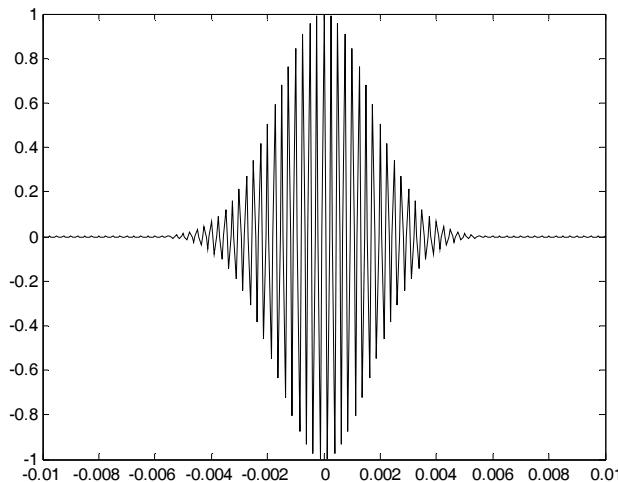
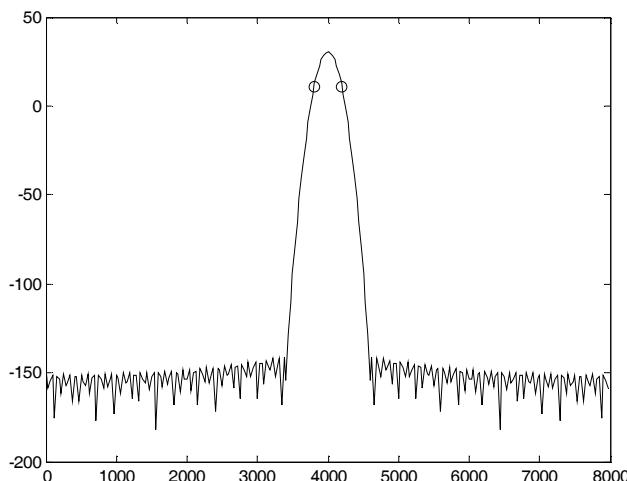
В качестве примера сформируем гауссов радиоимпульс с несущей частотой 4 кГц и относительной шириной спектра 10%, измеренной по уровню  $-20$  дБ, а затем построим график его спектра, чтобы убедиться в правильности расчетов. Частоту дискретизации примем равной 16 кГц. Результат показан на рис. 3.23:

```
>> Fs = 16e3; % частота дискретизации
>> t = -10e-3:1/Fs:10e-3; % дискретное время
>> Fc = 4e3; % несущая частота
>> bw = 0.1; % относительная ширина спектра
>> bwr = -20; % уровень измерения ширины спектра
>> s = gauspuls(t, Fc, bw, bwr);
>> Nfft = 2^nextpow2(length(s));
>> sp = fft(s, Nfft); % спектр
>> sp_dB = 20*log10(abs(sp)); % амплитудный спектр в децибелах
>> f = (0:Nfft-1)/Nfft*Fs; % вектор частот спектра
```

```

>> plot(t, s) % график сигнала
>> figure
>> % график амплитудного спектра
>> plot(f(1:Nfft/2), sp_dB(1:Nfft/2))
>> % максимальный уровень спектра в децибелах
>> sp_max_db = 20*log10(max(abs(sp)));
>> edges = Fc * [1-bw/2 1+bw/2]; % граничные частоты
>> % отображаем заданные при расчете границы спектра
>> hold on
>> plot(edges, sp_max_db([1 1])+bwr, 'o')
>> hold off

```

*a**б*

**Рис. 3.23.** Гауссов радиоимпульс, сформированный функцией `gauspuls` (*а*), и его амплитудный спектр (*б*)

Из графика спектра (см. рис. 3.23, б) видно, что заданные при вызове функции `gauspuls` параметры спектра выдержаны точно.

Наконец, у функции `gauspuls` есть еще один вариант использования — она позволяет рассчитать *время*, за которое огибающая гауссового импульса упадет до заданного уровня относительно максимума. При этом вместо вектора значений времени в качестве первого входного параметра используется строка '`cutoff`':

```
tc = gauspuls('cutoff', fc, bw, bwr, tpe)
```

Входные параметры `fc`, `bw` и `bwr` имеют тот же смысл, что и раньше, а `tpe` — уровень огибающей (в децибелах относительно максимума), момент достижения которого нужно определить.

Возвращаемый результат `tc` — момент достижения огибающей уровня `tpe` (т. е. полуширина импульса, измеренная по уровню `tpe`).

В качестве примера рассчитаем время, за которое огибающая сформированного в предыдущем примере импульса уменьшается на 6 дБ (примерно в два раза):

```
>> gauspuls('cutoff', Fc, bw, bwr, -6)
ans =
0.0020
```

Полученный результат соответствует графику сигнала, приведенному ранее на рис. 3.23, а.

## Генерация последовательности импульсов

Функция `pulstran` служит для генерации конечной последовательности импульсов (*pulse train*) одинаковой формы с произвольно задаваемыми задержками и уровнями. Самые импульсы могут задаваться одним из двух способов: именем функции, генерирующей импульс, либо уже рассчитанным вектором отсчетов.

Если импульсы задаются именем генерирующей функции, функция `pulstran` вызывается следующим образом:

```
y = pulstran(t, d, 'func', p1, p2, ...)
```

Здесь `t` — вектор значений времени, `d` — вектор задержек, '`func`' — имя функции, генерирующей одиночный импульс. В качестве этой функции могут использоваться, например, `rectpuls`, `tripuls`, `gauspuls`, а также любые другие функции (в том числе и "самодельные"), принимающие в качестве первого входного параметра вектор моментов времени и возвращающие вектор рассчитанных отсчетов сигнала. Оставшиеся параметры `p1`, `p2`, ... — дополнительные, они передаются функции `func` при ее вызове.

Таким образом, функция `pulstran` в данном варианте использования генерирует выходной сигнал следующим образом:

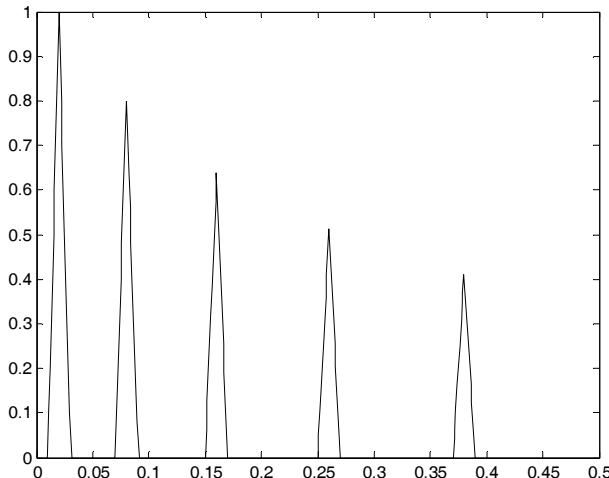
```
y = func(t-d(1), p1, p2, ...) + ...
    func(t-d(2), p1, p2, ...) + ...
    func(t-d(3), p1, p2, ...) + ...
    ...
    ...
```

Если  $d$  — двухстолбцовая матрица, то первый столбец трактуется как задержки импульсов, а второй — как их уровни. При этом выходной сигнал формируется так:

```
y = d(1,1) * func(t-d(1,2), p1, p2, ...) + ...
d(2,1) * func(t-d(2,2), p1, p2, ...) + ...
d(3,1) * func(t-d(3,2), p1, p2, ...) + ...
...
...
```

В качестве примера сформируем последовательность из пяти симметричных треугольных импульсов, интервалы между которыми линейно увеличиваются, а амплитуды экспоненциально уменьшаются. Частоту дискретизации выберем равной 1 кГц. Длительность импульса — 20 мс. Результат показан на рис. 3.24:

```
>> Fs = 1e3; % частота дискретизации
>> t = 0:1/Fs:0.5; % дискретное время
>> tau = 20e-3; % длительность импульса
>> d = [20 80 160 260 380]' * 1e-3; % задержки импульсов
>> d(:,2) = 0.8.^(:4:); % амплитуды импульсов
>> y = pulstran(t, d, 'tripuls', tau);
>> plot(t, y)
```



**Рис. 3.24.** Последовательность треугольных импульсов, сформированная с помощью функции pulstran

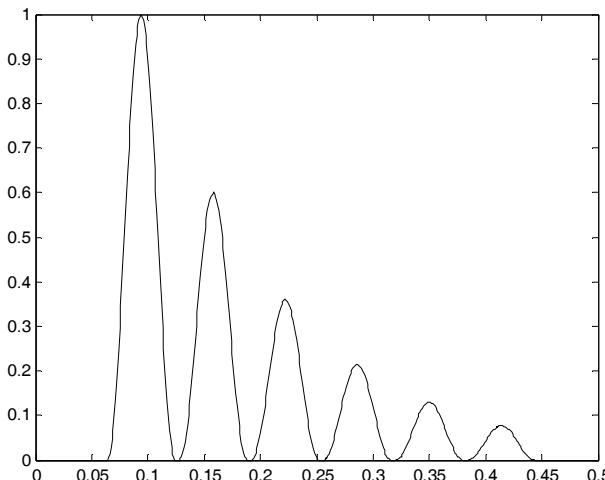
Если для генерации одиночного импульса нет готовой функции, можно рассчитать вектор отсчетов импульса, а затем использовать второй вариант вызова функции pulstran:

```
y = pulstran(t, d, p, fs, 'method')
```

Смысл входных параметров  $t$  и  $d$  тот же, что и раньше. Вектор  $p$  должен содержать отсчеты одиночного импульса, а параметр  $fs$  указывать частоту дискретизации, использованную при расчете этого вектора. Считается, что первый отсчет из вектора  $p$  соответствует нулевому моменту времени.

Поскольку частота  $fs$  может не совпадать с шагом значений вектора  $\tau$  (в принципе, они даже не обязаны представлять собой равномерную последовательность) и задержки из вектора  $d$  тоже не обязательно кратны этому шагу, для пересчета задержанных импульсов к сетке моментов времени  $\tau$  в общем случае необходимо использование интерполяции. Метод интерполяции может быть явно задан с помощью строкового параметра '*method*'. Возможны все методы, поддерживаемые функцией *interp1*: '*nearest*', '*linear*', '*spline*', '*pchip*', '*cubic*' и '*v5cubic*'.

Параметры  $fs$  и '*method*' при вызове могут опускаться, в этом случае используются их значения по умолчанию:  $fs = 1$  и '*method*' = '*linear*'.



**Рис. 3.25.** Последовательность импульсов, сформированная функцией *pulstran* из вектора отсчетов одиночного импульса

В качестве примера сформируем последовательность из шести импульсов, имеющих форму одного периода функции  $\sin^2$ . Пусть длительность импульса равна 60 мс, а частота его дискретизации — 400 Гц. Расстояние между центрами импульсов будет одинаковым и равным 64 мс, а частота дискретизации выходного сигнала — 1 кГц. Импульсы будут экспоненциально затухать с ростом номера. Результат показан на рис. 3.25:

```
>> % генерируем вектор отсчетов одиночного импульса
>> Fs0 = 400; % частота дискретизации импульса
>> tau = 60e-3; % длительность импульса
>> t0 = 0:1/Fs0:tau; % дискретное время для импульса
>> s0 = sin(pi*t0/tau).^2; % вектор отсчетов импульса
>> % генерируем последовательность импульсов
>> Fs = 1e3; % частота дискретизации последовательности
>> t = 0:1/Fs:0.5; % дискретное время для последовательности
>> d = (1:6)' * 64e-3; % задержки импульсов
```

```
>> d(:,2) = 0.6.^((0:5)'; % амплитуды импульсов
>> % последовательность импульсов
>> y = pulstran(t, d, s0, Fs0);
>> plot(t, y)
```

## Функции генерации периодических сигналов

Эти функции, входящие в пакет Signal Processing, позволяют формировать отсчеты периодических сигналов различной формы:

- `square` — последовательность прямоугольных импульсов;
- `sawtooth` — последовательность треугольных импульсов;
- `diric` — функция Дирихле (периодическая sinc-функция).

Последняя рассматриваемая в данном разделе функция `chirp` генерирует не периодический сигнал, а колебания с меняющейся частотой.

Далее эти функции рассматриваются более подробно.

### Последовательность прямоугольных импульсов

Для формирования последовательности прямоугольных импульсов служит функция `square`. В простейшем случае эта функция принимает один входной параметр — вектор значений времени `t`:

```
y = square(t)
```

Генерируемая при этом последовательность импульсов имеет период  $2\pi$  и скважность 2 (т. е. длительность импульса равна половине периода). Последовательность является двуполярной — сигнал принимает значения  $-1$  и  $1$ .

Сформировать последовательность с периодом  $T$  можно следующим образом:

```
y = square(2*pi*t/T)
```

С помощью второго входного параметра `duty` можно регулировать скважность получаемой последовательности. Однако этот параметр задает не саму скважность, а обратную ей величину — *коэффициент заполнения* (в процентах), т. е. отношение длительности импульса к периоду:

```
y = square(t, duty)
```

По умолчанию значение параметра `duty` равно 50, т. е. генерируется меандр.

#### **ВНИМАНИЕ!**

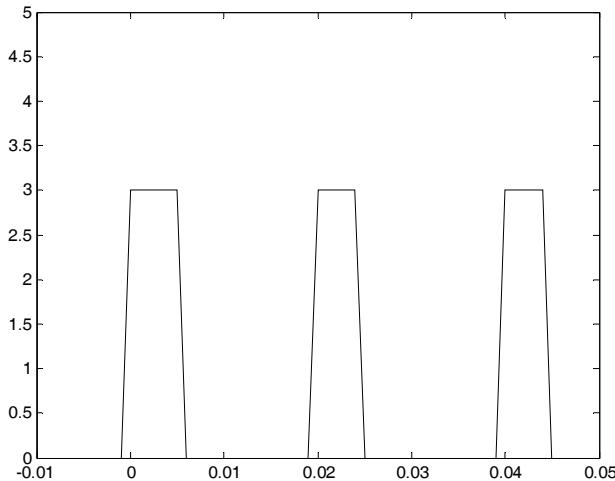
Обратите внимание на то, что значение параметра `duty` задается именно в *процентах*, а не в дробных единицах.

В качестве примера сформируем последовательность однополярных прямоугольных импульсов с амплитудой 3 В, частотой следования 50 Гц и длительностью 5 мс. Будем использовать частоту дискретизации 1 кГц и временной интервал  $-10 \dots 50$  мс (рис. 3.26):

```

>> Fs = 1e3; % частота дискретизации
>> t = -10e-3:1/Fs:50e-3; % дискретное время
>> A = 3; % амплитуда
>> f0 = 50; % частота следования импульсов
>> tau = 5e-3; % длительность импульсов
>> s = (square(2*pi*t*f0, f0*tau*100) + 1) * A/2;
>> plot(t, s)
>> ylim([0 5])

```



**Рис. 3.26.** Последовательность прямоугольных импульсов, полученная с помощью функции square

### ЗАМЕЧАНИЕ

На рис. 3.26 видно, что импульсы имеют неодинаковую ширину. Поскольку длительность импульса в данном случае равна ровно пяти интервалам дискретизации ( $5 \text{ мс} \times 1 \text{ кГц} = 5$ ), из-за погрешностей представления значений времени в компьютере каждый импульс может быть представлен либо пятью, либо шестью ненулевыми отсчетами.

### Последовательность треугольных импульсов

Для формирования последовательности треугольных импульсов служит функция sawtooth. В простейшем случае эта функция принимает один входной параметр — вектор значений времени  $t$ :

$y = \text{sawtooth}(t)$

Генерируемая при этом последовательность импульсов имеет период  $2\pi$ . На протяжении периода сигнала линейно нарастает от  $-1$  до  $1$ .

Сформировать последовательность с периодом  $T$  можно следующим образом:

$y = \text{sawtooth}(2*\pi*t/T)$

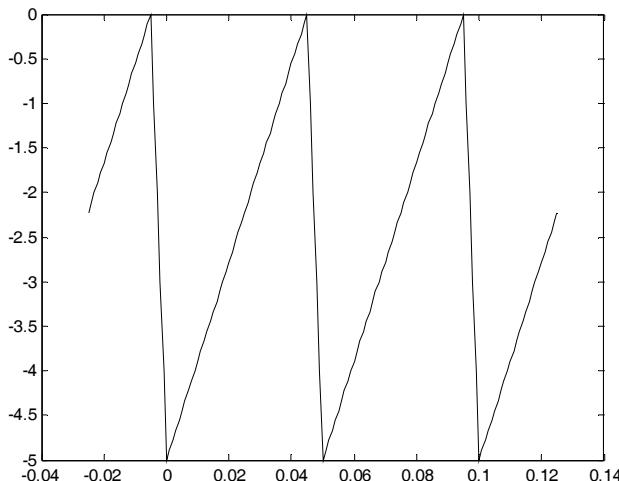
С помощью второго входного параметра `width` можно регулировать длительность "обратного хода" — промежутка, на котором уровень сигнала линейно падает от 1 до  $-1$ . При указании параметра `width` сигнал линейно возрастает от  $-1$  до 1 за время  $2\pi \text{width}$ , а затем за время  $2\pi(1 - \text{width})$  линейно убывает от 1 до  $-1$ :

```
y = sawtooth(t, width)
```

По умолчанию значение параметра `width` равно 1. При `width = 0,5` получится последовательность симметричных треугольных импульсов.

В качестве примера сформируем последовательность треугольных импульсов отрицательной полярности с амплитудой 5 В, периодом 50 мс и длительностью падающего участка 5 мс. Будем использовать частоту дискретизации 1 кГц и временной интервал  $-25\dots125$  мс (рис. 3.27):

```
>> Fs = 1e3; % частота дискретизации
>> t = -25e-3:1/Fs:125e-3; % дискретное время
>> A = 5; % амплитуда
>> T = 50e-3; % период
>> t1 = 5e-3; % длительность падающего участка
>> s = (sawtooth(2*pi*t/T, 1-t1/T) - 1) * A/2;
>> plot(t, s)
```



**Рис. 3.27.** Последовательность треугольных импульсов, полученная с помощью функции `sawtooth`

## Функция Дирихле

Функция Дирихле описывается формулой

$$\text{diric}(x) = \frac{\sin(nx / 2)}{n \sin(x / 2)},$$

где  $n$  — целое положительное число.

Функция имеет пульсирующий вид: пульсации максимального уровня расположены при  $x = 2\pi k$ , значение функции в этих точках равно  $(-1)^{k(n-1)}$ . Между этими главными пульсациями расположены пульсации меньшего уровня. При нечетном  $n$  все главные пульсации имеют положительную полярность, и период функции равен  $2\pi$ . При четном  $n$  полярность главных пульсаций чередуется, и период функции оказывается вдвое больше —  $4\pi$ .

Для расчета функции Дирихле в MATLAB служит функция `diric`. Синтаксис ее вызова следующий:

```
y = diric(x, n)
```

Назначение входных параметров  $x$  и  $n$  соответствует приведенной выше формуле.

Функцию Дирихле называют еще *периодической sinc-функцией* (о функции `sinc` ранее в разд. "Импульс с ограниченной полосой частот" этой главы). При нечетном  $n$  это действительно так, и функцию Дирихле можно представить следующим образом:

$$\text{diric}(x) = \sum_{k=-\infty}^{\infty} \text{sinc}\left(n\left(\frac{t}{2\pi} - k\right)\right).$$

При четном  $n$  функция Дирихле является суммой сдвинутых во времени sinc-функций с чередующимися знаками:

$$\text{diric}(x) = \sum_{k=-\infty}^{\infty} (-1)^k \text{sinc}\left(n\left(\frac{t}{2\pi} - k\right)\right).$$

Поскольку функция `sinc` имеет равномерный спектр в полосе частот от нуля до  $\pi$ , представление функции Дирихле в виде ряда Фурье (такое представление легко получить с помощью формулы, приведенной в разд. "Связь преобразования Фурье и коэффициентов ряда Фурье" главы 1) имеет очень простой вид — количество гармонических слагаемых конечно, а их амплитуды одинаковы:

□ при нечетном  $n$

$$\text{diric}(x) = \frac{1}{n} + \frac{2}{n} \cos(x) + \frac{2}{n} \cos(2x) + \dots + \frac{2}{n} \cos\left(\frac{n-1}{2}x\right);$$

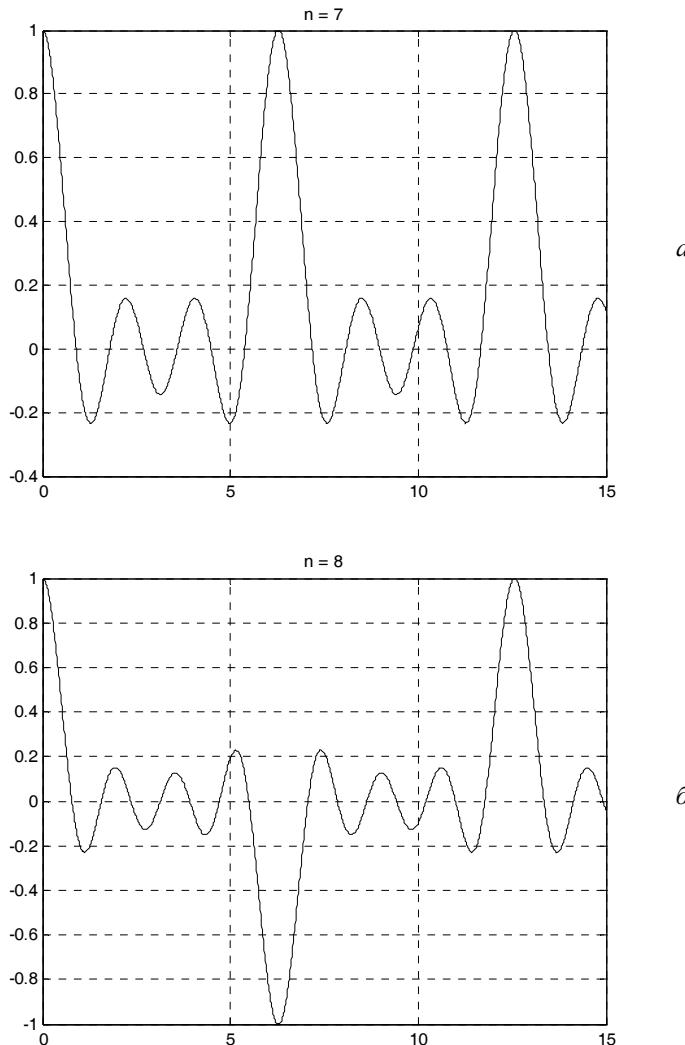
□ при четном  $n$

$$\text{diric}(x) = \frac{2}{n} \cos\left(\frac{x}{2}\right) + \frac{2}{n} \cos\left(\frac{3x}{2}\right) + \dots + \frac{2}{n} \cos\left(\frac{n-1}{2}x\right).$$

В качестве примера построим графики функции Дирихле при нечетном и четном значениях  $n$  ( $n = 7$  и  $n = 8$ , рис. 3.28):

```
>> x = 0:0.01:15;
>> plot(x, diric(x, 7))
>> grid on
>> title('n = 7')
```

```
>> figure
>> plot(x, diric(x, 8))
>> grid on
>> title('n = 8')
```



**Рис. 3.28.** Функция Дирихле нечетного (*а*) и четного (*б*) порядка

## Генерация сигнала с меняющейся частотой

Функция `chirp` предназначена для генерации колебаний с единичной амплитудой, мгновенная частота которых меняется по заданному закону:

```
y = chirp(t, f0, t1, f1, 'method', phi)
```

Здесь  $t$  — вектор значений времени,  $\text{phi}$  — начальная фаза колебания. Остальные параметры определяют закон изменения частоты.

### ЗАМЕЧАНИЕ

Подробнее понятие мгновенной частоты будет обсуждаться в разд. "Угловая модуляция" главы 8.

Строковый параметр 'method' определяет тип зависимости мгновенной частоты от времени — 'linear', 'quadratic' или 'logarithmic'. Числовые параметры  $f_0$ ,  $t_0$  и  $f_1$  создают опорные точки для расчетов: в нулевой момент времени мгновенная частота равна  $f_0$ , а в момент времени  $t_1$  она равна  $f_1$ .

Математически закон изменения мгновенной частоты выглядит следующим образом:

'linear':

$$f(t) = f_0 + (f_1 - f_0) \frac{t}{t_1};$$

'quadratic':

$$f(t) = f_0 + (f_1 - f_0) \left( \frac{t}{t_1} \right)^2;$$

'logarithmic' на деле противоречит своему названию — зависимость мгновенной частоты от времени при этом не логарифмическая, а экспоненциальная:

$$f(t) = f_0 \left( \frac{f_1}{f_0} \right)^{t/t_1}.$$

### ЗАМЕЧАНИЕ

Диапазон значений времени в векторе  $t$  может не включать в себя значений 0 и  $t_1$ .

Параметры  $\text{phi}$  и 'method' при вызове функции можно опускать, тогда будут использованы их значения по умолчанию:  $\text{phi} = 0$  и 'method' = 'linear'.

Ограничение сигнала по длительности не производится, колебания генерируются для всех значений времени, переданных функции в векторе  $t$ .

В качестве примера сформируем три сигнала, определенных на промежутке 0...1 с и имеющих разные законы изменения мгновенной частоты. В нулевой момент времени все сигналы имеют мгновенную частоту 1 кГц, а в момент времени 1 с — 3 кГц. Частоту дискретизации выберем равной 8 кГц:

```
>> Fs = 8e3; % частота дискретизации
>> t = 0:1/Fs:1; % дискретное время
>> f0 = 1e3;
>> t1 = 1;
>> f1 = 3e3;
>> s1 = chirp(t, f0, t1, f1, 'linear');
```

```

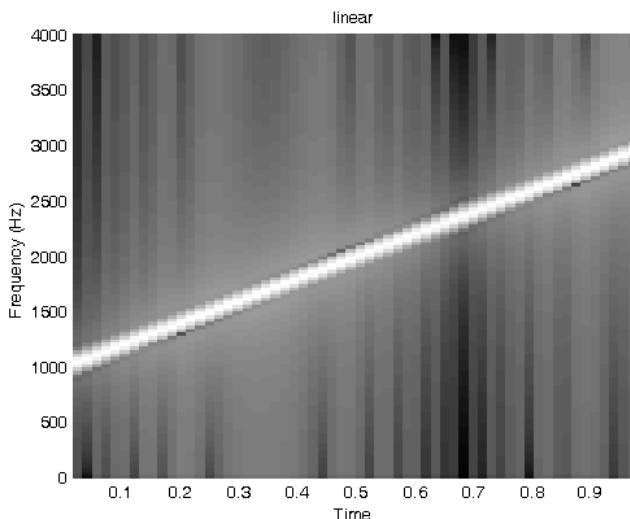
>> s2 = chirp(t, f0, t1, f1, 'quadratic');
>> s3 = chirp(t, f0, t1, f1, 'logarithmic');
>> spectrogram(s1, 256, [], [], Fs, 'yaxis')
>> title('linear')
>> colormap gray
>> figure
>> spectrogram(s2, 256, [], [], Fs, 'yaxis')
>> title('quadratic')
>> colormap gray
>> figure
>> spectrogram(s3, 256, [], [], Fs, 'yaxis')
>> title('logarithmic')
>> colormap gray

```

На рис. 3.29—3.31 показаны спектрограммы сформированных сигналов, наглядно демонстрирующие характер изменения мгновенной частоты при различных значениях параметра '`method`'. Обратите внимание на то, что значения мгновенной частоты при  $t = 0$  и  $t = 1$  для всех трех сигналов совпадают.

### ЗАМЕЧАНИЕ

Функция `spectrogram` строит *спектрограмму*, т. е. зависимость мгновенного амплитудного спектра сигнала от времени. Величина модуля спектральной функции отображается цветом в координатах "время — частота". Подробнее о функции `spectrogram` рассказано в главе 5. Команда `colormap gray` устанавливает для графиков палитру оттенков серого цвета; в противном случае при черно-белом воспроизведении цветных спектрограмм зависимость оттенка от уровня амплитудного спектра оказалась бы немонотонной.



**Рис. 3.29.** Спектрограмма сигнала, сформированного функцией `chirp` при линейном законе изменения мгновенной частоты

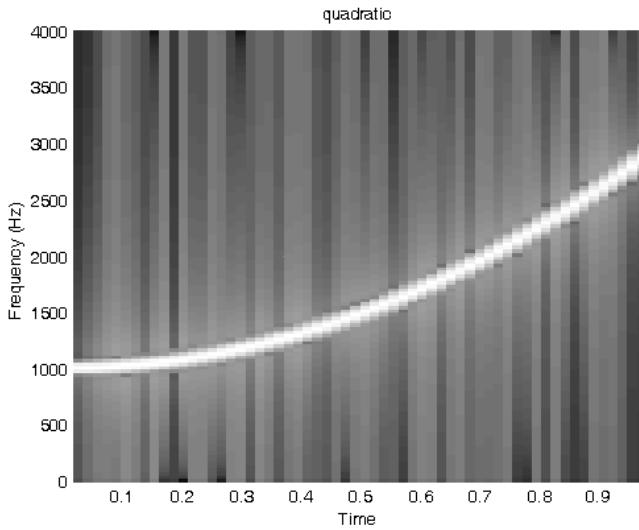


Рис. 3.30. Спектрограмма сигнала, сформированного функцией `chirp` при квадратичном законе изменения мгновенной частоты

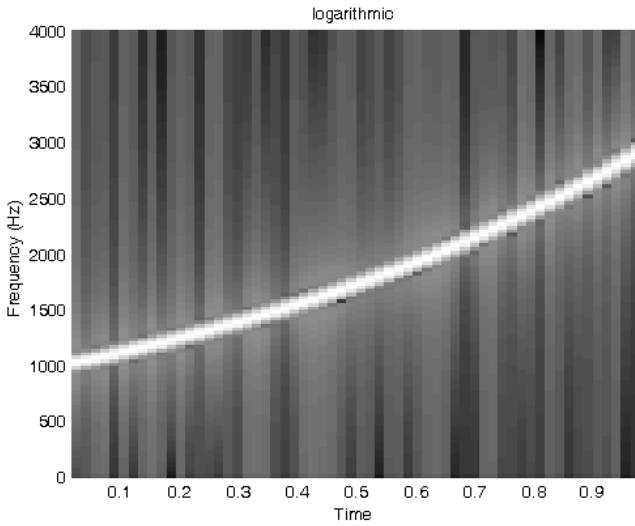


Рис. 3.31. Спектрограмма сигнала, сформированного функцией `chirp` при экспоненциальном законе изменения мгновенной частоты

## Формирование случайных сигналов

Для генерации случайных чисел в MATLAB служат функции `rand` (равномерное распределение) и `randn` (нормальное распределение), а также средства пакета рас-

ширения Statistics, которые реализуют множество разных законов распределения вероятности. В данном разделе мы кратко рассмотрим формирование дискретных случайных сигналов с заданными распределением вероятности и корреляционными свойствами.

## **Реализация заданного закона распределения вероятности**

Как уже говорилось, средства генерации случайных чисел с различными законами распределения вероятности имеются в пакете Statistics. Справочная информация об этом пакете имеется, например, в книге [25], здесь же мы рассмотрим лишь общий способ получения случайных чисел с заданной функцией распределения на основе равномерно распределенных случайных чисел.

Пусть  $X$  — случайная величина, равномерно распределенная на интервале 0...1 (см. разд. "Равномерное распределение" главы 1). Для получения случайной величины  $Y$ , имеющей функцию распределения  $F_y(y)$ , случайную величину  $X$  необходимо подвергнуть следующему нелинейному преобразованию:

$$Y = F_y^{(-1)}(X), \quad (3.30)$$

где  $F_y^{(-1)}$  — функция, обратная по отношению к  $F_y(y)$ .

Действительно, при таком расчете вероятность того, что  $Y$  не превышает значения  $y$ , равна  $P(Y \leq y) = P(X \leq F_y(y))$ . Но  $X$  имеет равномерное распределение, поэтому  $P(X \leq F_y(y)) = F_y(y)$ . Таким образом,  $P(Y \leq y) = F_y(y)$ , т. е.  $Y$  действительно имеет требуемую функцию распределения.

Формулу (3.30) можно обобщить на случай произвольного преобразования функции распределения. Если случайная величина  $X$  имеет функцию распределения  $F_x(x)$ , а нужно получить случайную величину  $Y$  с функцией распределения  $F_y(y)$ , искомое преобразование следует записать следующим образом:

$$Y = F_y^{(-1)}(F_x(X)).$$

Преобразование  $F_x(X)$  делает распределение случайной величины равномерным, а преобразование  $F_y^{(-1)}(\dots)$  формирует случайную величину с заданным распределением вероятности.

### **ВНИМАНИЕ!**

С теоретической точки зрения рассмотренный подход является универсальным, однако на практике он может оказаться неудобным, поскольку обратные функции распределения для многих законов (например, для нормального) не выражаются через элементарные функции. В таких случаях приходится использовать более изощренные методики.

В качестве примера сгенерируем по формуле (3.30) случайные числа с рэлеевским законом распределения (1.80). Функция распределения для закона Рэлея получается интегрированием его плотности вероятности:

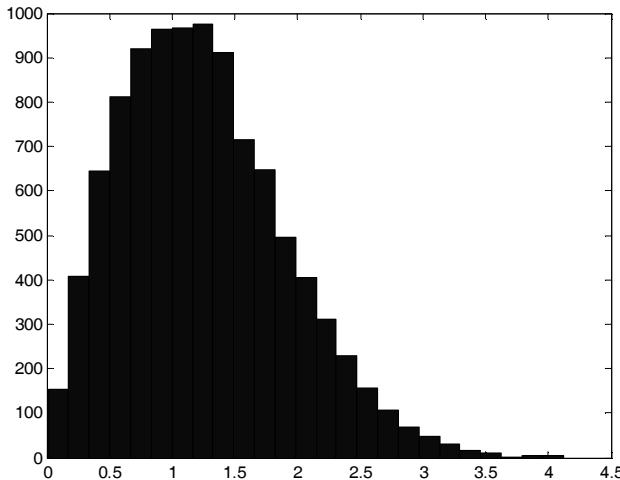
$$F_y(y) = \int_0^y \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = 1 - \exp\left(-\frac{y^2}{2\sigma^2}\right), \quad y \geq 0.$$

Обратная функция будет иметь вид

$$F_y^{(-1)}(x) = \sigma \sqrt{-2 \ln(1-x)}, \quad 0 \leq x < 1. \quad (3.31)$$

Генерируем равномерно распределенные случайные числа с помощью функции `rand`, производим их преобразование по формуле (3.31) и строим гистограмму с помощью функции `hist` (рис. 3.32):

```
>> N = 10000; % количество чисел
>> sigma = 1; % параметр рэлеевского закона
>> X = rand(1, N); % равномерное распределение
>> Y = sigma * sqrt(-2 * log(1 - X)); % закон Рэлея
>> hist(Y, 25) % гистограмма по 25 интервалам
```



**Рис. 3.32.** Гистограмма случайных чисел с рэлеевским распределением

Полученная гистограмма показывает хорошее соответствие с графиком рэлеевской плотности вероятности, показанным ранее на рис. 1.37.

### Реализация заданной корреляционной функции

Поскольку корреляционная функция случайного сигнала взаимно-однозначно связана с его спектром (см. разд. "Теорема Винера — Хинчина" главы 1), для формирования сигнала с заданной КФ можно взять отсчеты белого шума и пропустить их через фильтр с соответствующей АЧХ (ФЧХ фильтра не имеет значения, поскольку

к случайнм процессам неприменимо понятие фазового спектра). О теории работы дискретных фильтров речь пойдет в главе 4, а о методах их расчета — в главе 6. Здесь же мы, не сколько забегая вперед и используя материал последующих глав, рассмотрим лишь один пример, сформировав сигнал с экспоненциальной функцией корреляции:

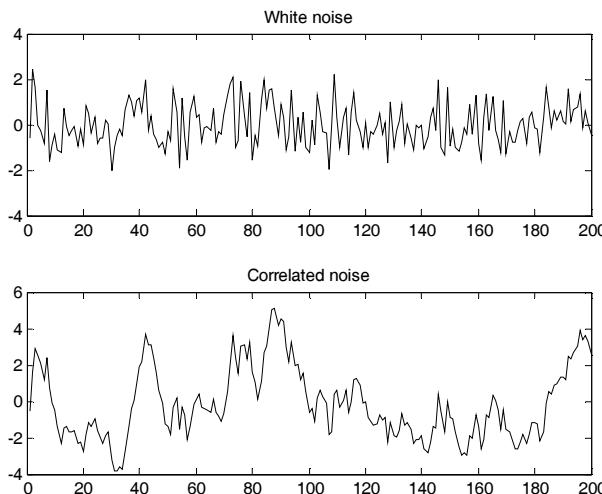
$$R_x(\Delta k) = \sigma_x^2 a^{|\Delta k|}, \quad |a| < 1. \quad (3.32)$$

Для формирования сигнала с такой КФ необходимо пропустить белый шум через дискретный фильтр, корреляционная функция импульсной характеристики которого имеет вид (3.32). Этому требованию удовлетворяет импульсная характеристика вида

$$h(k) = \sigma_x \sqrt{1-a^2} a^k, \quad k \geq 0.$$

Такая характеристика соответствует рекурсивному фильтру первого порядка с коэффициентом обратной связи, равным  $a$ . Требуемую фильтрацию осуществим с помощью функции `filter` (она также будет рассмотрена в главе 4):

```
>> X0 = randn(1, 1000); % независимые нормальные отсчеты
>> a = 0.9; % параметр экспоненциальной корреляции
>> sigma = 2; % среднеквадратическое значение результата
>> X = filter(sigma*sqrt(1-a^2), [1 -a], X0); % фильтрация
>> subplot(2, 1, 1)
>> plot(X0(1:200)) % график белого шума
>> title('White noise')
>> subplot(2, 1, 2)
>> plot(X(1:200)) % график коррелированного шума
>> title('Correlated noise')
```



**Рис. 3.33.** Белый (сверху) и коррелированный (снизу) шум

Графики белого шума и экспоненциально коррелированного сигнала показаны на рис. 3.33. Хорошо видно, что коррелированный шум сильно сглажен по сравнению с белым шумом.

## Генерация дискретного нормального белого шума

Дискретный белый шум с нормальным распределением можно сгенерировать с помощью функции `randn`, имеющей следующий синтаксис:

```
X = randn(m, n)
```

В результате вызова функции генерируется массив `X`, содержащий `m` строк и `n` столбцов псевдослучайных чисел, имеющих нормальное распределение с нулевым математическим ожиданием и единичной дисперсией.

Параметр `n` при вызове можно опустить, по умолчанию используется значение `n = m`, т. е. генерируется квадратная матрица.

В задачах, связанных с обработкой сигналов, для генерации дискретного нормального белого шума удобнее использовать функцию `wgn` (*white Gaussian noise*) пакета Communications, поскольку она позволяет удобным образом задавать уровень генерируемого шума. Синтаксис вызова функции следующий:

```
y = wgn(m, n, p, imp, state, 'powertype', 'outputtype');
```

Здесь `m` и `n` — как и ранее, размеры генерируемой матрицы, а `p` — мощность генерируемого шума в единицах, задаваемых параметром '`'powertype'`' (по умолчанию — в децибелах). Остальные параметры являются необязательными и имеют значения по умолчанию.

Параметр `imp` задает импеданс нагрузки в омах (предполагается, что генерируются отсчеты случайного *напряжения* на этой нагрузке). По умолчанию используется импеданс нагрузки, равный 1 Ом.

Целочисленный параметр `state` позволяет принудительно задавать начальное состояние генератора гауссовских случайных чисел MATLAB (функция `randn`). По умолчанию используется текущее состояние.

Строковый параметр '`'powertype'`' задает единицы измерения мощности, использованные при указании параметра `p`. Возможны следующие значения:

- '`'dB'` — мощность `p` задается в децибелах относительно 1 Вт, значению 0 дБ соответствует дисперсия, равная `imp^2`;
- '`'dBm'` — мощность `p` задается в децибелах относительно 1 мВт, значению 0 дБ соответствует дисперсия, равная `imp^2/1000`;
- '`'linear'` — мощность `p` задается в ваттах, дисперсия генерируемого шума равна `p*imp`.

Строковый параметр '`'outputtype'`' позволяет задавать генерацию вещественного или комплексного шума. Возможны значения '`'real'`' (вещественный шум; генерируется по умолчанию) и '`'complex'`' (комплексный шум). Если генерируется комплексный шум, его вещественная и мнимая части имеют мощности `p/2`.

## Добавление белого шума к сигналу

В том же пакете Communications имеется еще одна полезная функция, работающая с нормальным белым шумом. Это функция `awgn`, реализующая *канал связи с аддитивным белым гауссовым шумом* (*additive white Gaussian noise channel, AWGN channel*), т. е. добавляющая к сигналу белый шум с заданным уровнем. Синтаксис вызова функции следующий:

```
y = awgn(x, snr, sigpower, state, 'powertype');
```

Здесь `x` — вектор отсчетов сигнала. Скаляр `snr` задает отношение сигнал/шум в единицах, задаваемых параметром '`powertype`' (по умолчанию — в децибелах). Остальные параметры являются необязательными и имеют значения по умолчанию.

Параметр `sigpower` указывает мощность сигнала `x` в единицах, задаваемых параметром '`powertype`' (по умолчанию — в децибелах). По умолчанию предполагается, что мощность сигнала равна 0 дБ, т. е. средний квадрат модуля значений из вектора `x` равен единице. Параметр `sigpower` может также принимать строковое значение '`measured`', при этом мощность сигнала *автоматически измеряется*.

Целочисленный параметр `state` позволяет принудительно задавать начальное состояние генератора гауссовых случайных чисел MATLAB (функция `randn`). По умолчанию используется текущее состояние.

Строковый параметр '`powertype`' задает единицы измерения мощности, использованные при указании параметров `snr` и `sigpower`. Возможны следующие значения:

- '`'dB'`' — мощность сигнала и отношение сигнал/шум задаются в децибела;
- '`'linear'`' — мощность сигнала задается в ваттах, отношение сигнал/шум — в разах.

При расчете мощности сигнала предполагается, что значения вектора `x` представляют собой отсчеты напряжения на нагрузке с импедансом 1 Ом.

Результатом работы является вектор "зашумленных" отсчетов `y`. Если значения `x` являются вещественными, функция `awgn` добавляет вещественный шум, если комплексными — комплексный шум.

## Получение данных из внешних источников

Преобразование аналогового сигнала в цифровой и обратно — это процессы, которые выполняются аппаратными средствами. MATLAB же, будучи программным продуктом, может лишь взаимодействовать с соответствующим оборудованием (такое взаимодействие осуществляется, например, с помощью пакета Data Acquisition). Кроме того, в MATLAB предусмотрены средства для воспроизведения и записи звука, а также для работы со звуковыми файлами формата WAV. В данном разделе мы подробно рассмотрим считывание и запись WAV-файлов, а также воспроизведение звука. Детально рассматривать вопросы взаимодействия с оборудованием ввода/вывода данных из-за ограниченного объема книги не представляет-

ся возможным, поэтому в конце главы будет приведен лишь краткий обзор возможностей и демонстрационных примеров пакета Data Acquisition.

## Чтение WAV-файлов

Для считывания WAV-файлов в MATLAB имеется функция `wavread`. В простейшем случае она может быть использована следующим образом:

```
y = wavread('filename');
```

Здесь `filename` — имя звукового файла (расширение `.wav` указывать не обязательно). В имя файла необходимо включить полный путь, за исключением тех случаев, когда файл находится в текущем (для MATLAB) каталоге или в одном из каталогов, входящих в список поиска MATLAB.

В результате вызова функции в переменную `y` будет помещено *все* содержимое указанного файла. Строки матрицы `y` соответствуют отсчетам сигнала, столбцы — каналам, которых в WAV-файле может быть несколько.

В звуковых файлах отсчеты сигнала представлены целыми числами, лежащими в диапазоне  $-128\dots+127$  (8 бит на отсчет) либо  $-32\,768\dots+32\,767$  (16 бит на отсчет). Управлять нормировкой считываемых отсчетов можно с помощью дополнительного строкового параметра '`fmt`', добавляемого в конце списка входных параметров функции `wavread`. При принятом по умолчанию варианте '`double`' значения отсчетов приводятся к диапазону  $-1\dots+1$ , а при значении '`'native'`' функция возвращает целые числа в том виде, в каком они хранятся в WAV-файле.

Помимо собственно отсчетов сигнала, в WAV-файлах хранится еще и служебная информация о частоте дискретизации, количестве бит на отсчет и т. п. Узнать частоту дискретизации можно, используя при вызове функции второй выходной параметр:

```
[y, Fs] = wavread('filename');
```

При этом переменная `Fs` получает значение, равное частоте дискретизации в герцах.

Чтобы узнать, сколько уровней сигнала содержится в звуковом файле (точнее, число бит на отсчет), необходимо добавить третий выходной параметр:

```
[y, Fs, bits] = wavread('filename');
```

Еще два параметра, которые часто хочется знать заранее, — это число отсчетов и каналов записи. Для получения данной информации нужно вызвать функцию `wavread` с двумя *входными* параметрами. Первый — это по-прежнему имя файла, вторым же должна быть текстовая строка '`size`':

```
wavesize = wavread('filename', 'size');
```

В отличие от предыдущих вариантов вызова функции, в данном случае не производится считывания самих звуковых данных. Из WAV-файла лишь извлекается служебная информация, которая возвращается в виде двухэлементного вектора-строки

(в приведенном примере — `wavesize`). Первый элемент вектора содержит число отсчетов, второй — число каналов.

Наконец, имеется и важнейшая возможность считывания данных из WAV-файла не целиком, а отдельными фрагментами (без этого нельзя было бы работать с большими файлами). Для этого также используется второй входной параметр функции `wavread`. Если этот параметр является числом, будет считано соответствующее количество отсчетов (начиная с первого):

```
y = wavread('filename', N);
```

Если же нужный фрагмент расположен не в начале файла, придется указать его начало и конец. В этом случае второй входной параметр функции `wavread` должен представлять собой двухэлементный вектор:

```
y = wavread('filename', [n1 n2]);
```

В результате в переменную `y` будут считаны отсчеты с номерами от `n1` до `n2` включительно (нумерация отсчетов, как и элементов матриц в MATLAB, начинается с единицы). При этомчитываются все каналы звукозаписи. Возможности считывания информации из отдельных каналов не предусмотрено.

### **СОВЕТ**

Как видите, получить информацию о частоте дискретизации и числе бит на отсчет можно только одновременно со считыванием звуковых данных. Если необходимо получить эту информацию заранее, до основной работы со звуком, можно попросить MATLAB считать всего один отсчет:

```
[y, Fs, bits] = wavread('filename', 1);
```

Рассмотрим работу с WAV-файлами на конкретном примере. Возьмем для этого бодрый аккорд `tada.wav`, входящий в стандартный набор звуков Windows.

Прежде всего узнаем число отсчетов и каналов:

```
>> wavesize = wavread('c:\windows\media\tada', 'size')
wavesize =
    42752          2
```

Как видим, эта стереозапись (2 канала) содержит 42 752 отсчета.

Далее извлекаем информацию о частоте дискретизации и числе бит на отсчет:

```
>> [y, Fs, bits] = wavread('c:\windows\media\tada', 1)
y =
    0      0
Fs =
    22050
bits =
    16
```

Полученные результаты очевидны — частота дискретизации 22,05 кГц, используется 16 бит на отсчет (65 536 уровней сигнала). "В нагрузку" мы получили значения первого отсчета из обоих каналов (вектор `y`).

Теперь мы можем узнать о файле кое-что еще, например определить продолжительность звучания. Для этого нужно разделить число отсчетов (первый элемент полученного ранее вектора wavesize) на частоту дискретизации Fs:

```
>> wavesize(1)/Fs  
ans =  
    1.9389
```

Результат — длительность звука в секундах.

Чтобы узнать, сколько памяти потребуется MATLAB для хранения всей записи, нужно перемножить число отсчетов и число каналов, а затем увеличить результат еще в 8 раз. Можно сразу же поделить полученное значение на 1024, чтобы получить ответ в килобайтах, или на  $1024^2$ , если ожидаемое число лучше измерять мегабайтами:

```
>> prod(wavesize)*8/1024  
ans =  
    668
```

Расход памяти оказывается небольшим (668 Кбайт), так что можно считать файл в память целиком (не забудьте про точку с запятой в конце строки):

```
>> y = wavread('c:\windows\media\tada');
```

Теперь все содержимое файла считано в матрицу y. Проверим ее размеры:

```
>> size(y)  
ans =  
    42752          2
```

Как видите, размеры совпадают с полученной ранее информацией о файле: 42 752 отсчета (строк), два канала (столбца).

Чтобы "окинуть взглядом" звуковой сигнал, выведем его в виде графика — отдельно для правого и левого каналов стереозаписи, используя для этого функцию subplot (подробнее об использовании этой функции речь пойдет в разд. "Одновременный вывод нескольких графиков" приложения 1). Результат показан на рис. 3.34:

```
>> subplot(2, 1, 1)  
>> plot(y(:, 1))  
>> subplot(2, 1, 2)  
>> plot(y(:, 2))
```

Если при выводе графика разрешение по горизонтали имеет большее значение, чем по вертикали, можно воспользоваться функцией strips, специально предназначеннной для отображения длинных сигналов в "нарезанном" на фрагменты виде (фрагменты выводятся друг под другом). Синтаксис вызова функции strips следующий:

```
strips(x, N)
```

Здесь  $x$  — вектор отсчетов сигнала (двумерные массивы не допускаются),  $N$  — число отсчетов в каждом фрагменте (этот параметр можно опустить, по умолчанию размер фрагмента составляет 200 отсчетов).

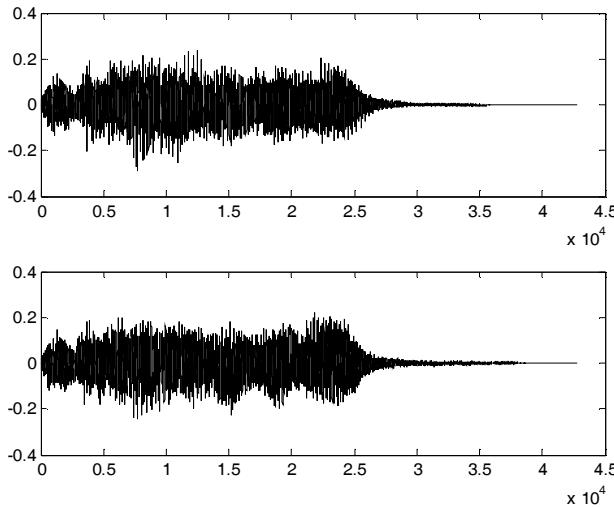


Рис. 3.34. График звукового сигнала

В качестве примера выведем с помощью функции `strips` график первого (левого) канала сигнала `tada.wav` (рис. 3.35):

```
>> strips(y(:,1), 10000)
```

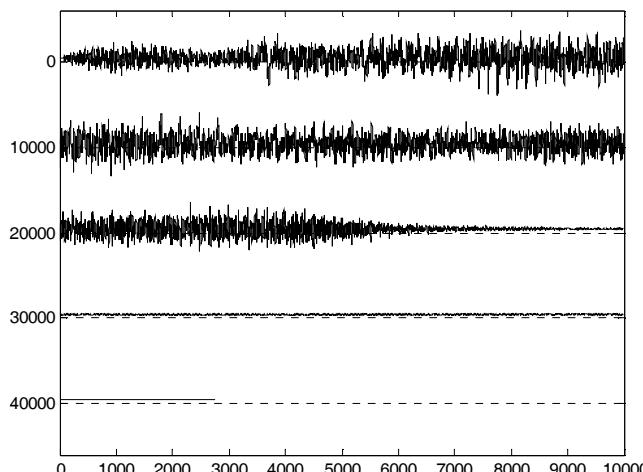


Рис. 3.35. График сигнала, выведенный с помощью функции `strips`

Над помещенным в переменную MATLAB звуковым сигналом можно выполнять любые преобразования, а затем воспроизвести полученный звук или сохранить его в виде нового WAV-файла. Обо всем этом речь пойдет в нескольких следующих разделах. Можно также сохранить сигнал на диске как переменную MATLAB (в виде MAT-файла) — см. разд. "Ввод и вывод данных" приложения 1.

Функция `wavread` не является встроенной — она целиком написана на языке MATLAB с использованием средств работы с двоичными файлами. Если вас интересует структура WAV-файлов и организация их считывания, попробуйте разобраться в тексте функции `wavread` — файл `wavread.m` расположен в папке `Program Files\ MATLAB\R2010a\toolbox\matlab\audiovideo`.

MATLAB умеет работать только с несжатыми WAV-файлами (формат PCM — Pulse Code Modulation). При попытке считать файл, в котором использован какой-либо из методов сжатия информации, будет выдано сообщение "Data compression format (...) is not supported".

## Запись WAV-файлов

Чтобы записать вектор (или матрицу) на диск в виде WAV-файла, используется функция `wavwrite`:

```
wavwrite(y, Fs, N, 'filename')
```

Здесь `y` — записываемые данные (вектор для монофонической записи, двухстолбцовая матрица — для создания стереофайла), `Fs` — частота дискретизации в герцах, `N` — число бит на отсчет (8 или 16), `'filename'` — имя создаваемого файла. Выходных параметров у данной функции нет.

Параметры `N` и `Fs` можно опускать, при этом используются значения по умолчанию — `N = 16` и `Fs = 8000`:

```
wavwrite(y, Fs, 'filename')  
wavwrite(y, 'filename')
```

Записываемые данные должны быть вещественными и лежать в диапазоне  $-1\dots 1$ . Значения, выходящие из этого диапазона, будут "обрезаны" и сделаны равными  $\pm 1$ .

## Воспроизведение звука

Если ваш компьютер оборудован звуковой картой, то, помимо работы с WAV-файлами, вы имеете и возможность воспроизведения векторов и матриц в звуковом виде. Для этого есть целых три функции — `sound`, `soundsc` и `wavplay`. В простейшем случае все три функции вызываются одинаково и обеспечивают воспроизведение вектора `y` (или двухстолбцовой матрицы — для стереозвука), содержащего отсчеты сигнала, с заданной частотой дискретизации `Fs` (в герцах):

```
sound(y, Fs)  
soundsc(y, Fs)  
wavplay(y, Fs)
```

Различие между этими функциями заключается в предоставляемых ими дополнительных возможностях.

## Функция **sound**

Функция **sound** обеспечивает воспроизведение сигнала с заданными частотой дискретизации и числом уровней (бит на отсчет):

```
sound(y, Fs, bits)
```

Здесь  $y$  — вектор или двухстолбовая матрица отсчетов сигнала,  $Fs$  — частота дискретизации в герцах,  $bits$  — число бит на отсчет (8 или 16).

Параметры  $bits$  и  $Fs$  можно опускать, при этом будут использоваться их значения по умолчанию:  $Fs = 8192$  и  $bits = 16$ .

Воспроизводимые данные  $y$  должны быть вещественными и лежать в диапазоне  $-1\dots1$ . Значения, выходящие из этого диапазона, "обрезаются" и делаются равными  $\pm 1$ .

Выходных параметров  $y$  функции нет. После вызова она передает вектор  $y$  звуковой карте для воспроизведения и сразу же, не дожидаясь окончания звука, возвращает управление командной строке MATLAB.

Если следующая команда **sound** будет использована до окончания предыдущего звука, будет выдано сообщение об ошибке "Unable to open sound device" (Невозможно открыть звуковое устройство).

## Функция **soundsc**

Функция **soundsc** (Sound scaled) отличается от функции **sound** лишь тем, что производит предварительное масштабирование отсчетов сигнала. Для управления масштабированием добавляется четвертый входной параметр  $s_{lim}$ :

```
soundsc(y, Fs, bits, s_lim)
```

Здесь входные параметры  $y$ ,  $Fs$ ,  $bits$  имеют то же назначение, что и для функции **sound**. Параметр  $s_{lim}$  должен быть двухэлементным вектором  $[s_{low} s_{high}]$ , он задает диапазон значений, который будет линейно преобразован к интервалу  $-1\dots1$ . Преобразование, таким образом, производится по формуле

$$y' = \frac{2y - (s_{low} + s_{high})}{s_{high} - s_{low}}.$$

При  $s_{lim} = [-1 1]$  функция **soundsc** эквивалентна функции **sound**.

Параметры  $s_{lim}$ ,  $bits$  и  $Fs$  при вызове можно опускать, при этом используются их значения по умолчанию:  $Fs = 8192$ ,  $bits = 16$  и  $s_{lim} = [\min(y) \max(y)]$ . Значение по умолчанию для  $s_{lim}$  обеспечивает точное приведение полного диапазона значений сигнала к интервалу  $-1\dots1$ .

## Функция *wavplay*

Наконец, третья функция, предназначенная для воспроизведения звука, имеет имя *wavplay*:

```
wavplay(y, Fs, 'mode')
```

Входные параметры *y* и *Fs* имеют тот же смысл, что и у предыдущих функций, а параметр '*mode*' управляет режимом воспроизведения. Этот параметр может принимать два значения:

- '*sync*' — синхронный режим, означающий, что функция вернет управление интерпретатору MATLAB только после окончания звука;
- '*async*' — асинхронный режим, при котором функция передает данные для воспроизведения звуковым драйверам Windows и сразу же возвращает управление системе MATLAB, не дожидаясь окончания звука.

Параметры '*mode*' и *Fs* при вызове можно опускать, при этом используются их значения по умолчанию: *Fs* = 11025 и '*mode*' = '*async*'.

Синхронный режим позволяет организовать в MATLAB-программе выдачу нескольких звуков подряд, не заботясь о расчете и соблюдении временных интервалов между соответствующими командами. В асинхронном же режиме можно одновременно с воспроизведением звука продолжать выполнение программы. К сожалению, при этом нет возможности программным путем проверить, закончилось ли воспроизведение звука.

### ЗАМЕЧАНИЕ 1

Такая программная проверка будет возможна, если для воспроизведения звука воспользоваться средствами пакета Data Acquisition (краткая информация о его возможностях приводится далее).

### ЗАМЕЧАНИЕ 2

Помимо перечисленных, воспроизводить звук позволяет также функция *audioplayer*, реализующая MATLAB-интерфейс для аудиопроигрывателя системы Windows.

## Запись звука

Функция *wavrecord* позволяет записать звук в переменную MATLAB с помощью звуковой карты компьютера:

```
y = wavrecord(n, Fs, ch, 'dtype')
```

Здесь *n* — число записываемых отсчетов, *Fs* — частота дискретизации в герцах, *ch* — число каналов записи, '*dtype*' — тип записываемых данных.

Возвращаемый результат *y* — матрица, каждый столбец которой соответствует одному каналу записи. При стереозаписи первый столбец — левый канал, второй — правый.

Для параметра 'dtype' возможны следующие значения:

- 'double' — 16-битовая запись, данные масштабируются к диапазону -1...1 и представляются в 8-байтовом формате с плавающей запятой;
- 'single' — 16-битовая запись, данные масштабируются к диапазону -1...1 и представляются в 4-байтовом формате с плавающей запятой;
- 'int16' — 16-битовая запись, данные представляются в двухбайтовом целочисленном формате (диапазон -32 768...32 767);
- 'uint8' — 8-битовая запись, данные представляются в однобайтовом беззнаковом целочисленном формате (диапазон 0...255, нулевому напряжению на входе соответствует значение 128).

Входные параметры 'dtype', ch и Fs можно опускать, при этом будут использоваться их значения по умолчанию: Fs = 11025, ch = 1, 'dtype' = 'double'.

### ЗАМЕЧАНИЕ

Еще одна функция, позволяющая осуществлять запись звука — audiorecorder. Она реализует MATLAB-интерфейс для программы звукозаписи, входящей в состав системы Windows.

## Готовые записи сигналов

В разделе **Audio** базовой библиотеки MATLAB и в пакете расширения **Signal Processing** имеется несколько готовых записей сигналов, сохраненных в виде MAT-файлов. Эти сигналы используются в качестве исходных данных в примерах, приводимых в документации, и демонстрационных программах. Возможно, вам они пригодятся для тестирования собственных программ и алгоритмов. Сведения об имеющихся сигналах приведены в табл. 3.1.

**Таблица 3.1.** Записи сигналов, имеющиеся в MATLAB

Имя файла	Пакет	Имя переменной	Размер	Частота дискретизации	Характер звука
chirp.mat	MATLAB	у	13 129 (1,6 с)	8192 Гц	Импульсы с меняющейся частотой
gong.mat	MATLAB	у	42 028 (5,1 с)	8192 Гц	Удар гонга
handel.mat	MATLAB	у	73 113 (8,9 с)	8192 Гц	Хор
laughter.mat	MATLAB	у	52 634 (6,4 с)	8192 Гц	Смех
splat.mat	MATLAB	у	10 001 (1,2 с)	8192 Гц	Тон с меняющейся частотой, затем звук удара
train.mat	MATLAB	у	12 880 (1,5 с)	8192 Гц	Гудок поезда

Таблица 3.1 (окончание)

Имя файла	Пакет	Имя переменной	Размер	Частота дискретизации	Характер звука
mtlb.mat	SP	mtlb	4001 (0,54 с)	7418 Гц	Произнесенное слово "MATLAB"
vcosig.mat	SP	vcosig	19661 (2,4 с)	8192 Гц	Данный сигнал имеет очень забавную спектрограмму

## Пакет расширения Data Acquisition

Пакет расширения Data Acquisition позволяет непосредственно из MATLAB работать с оборудованием аналогового и цифрового ввода/вывода данных. В комплект поставки входят драйверы для следующих устройств:

- звуковых карт, поддерживаемых операционной системой Windows;
- параллельных портов персонального компьютера (LPT);
- плат ввода/вывода фирм Advantech, Measurement Computing и National Instruments.

Разработать драйверы для других устройств можно с помощью Data Acquisition Toolbox Adaptor Kit; дополнительную информацию можно получить на сайте фирмы The MathWorks, Inc. по адресу <http://www.mathworks.com/products/daq/>.

Пакет Data Acquisition (разумеется, в сочетании с перечисленным оборудованием) предоставляет следующие возможности:

- аналоговый ввод и вывод информации в реальном масштабе времени, включая возможность одновременного выполнения аналого-цифрового и цифроаналогового преобразований;
- цифровой ввод и вывод информации в реальном масштабе времени;
- запись вводимых данных на диск либо их загрузка непосредственно в рабочую область памяти MATLAB в виде переменных;
- буферизацию данных для осуществления ввода в фоновом режиме;
- триггеры, управляемые программно или аппаратно генерируемыми событиями (это дает возможность осуществлять синхронизацию — например, запускать процесс аналогового ввода при достижении входным сигналом некоторого уровня).

Подробно рассматривать пакет Data Acquisition в рамках данной книги не представляется возможным, поэтому покажем лишь пример использования его возможностей в двух демонстрационных программах, поставляемых в составе пакета.

Первая программа реализует осциллограф, позволяя просматривать графики сигналов, получаемых от имеющихся в системе устройств ввода данных. Данная про-

грамма может быть вызвана из командной строки (ее имя — `daqscope`) либо из окна справочной системы (команда меню **Help | Product Help**, раздел оглавления справки `Data Acquisition Toolbox\Demos\GUI Demos\Analog Input Oscilloscope`).

Вид окна осциллографа приведен на рис. 3.36. Имеющиеся в окне элементы управления позволяют выбирать источник сигнала (справа в верхнем углу окна), задавать частоту дискретизации (флажок и поле ввода **Sample Rate**), устанавливать масштаб изображения по вертикали (переключатели **Volts/Div** и **Autoset** и поле ввода для ручного ввода вертикального масштаба) и горизонтали (ползунок **X-Axis Range**). Две кнопки на панели инструментов окна управляют запуском и остановкой процесса приема и отображения сигнала.

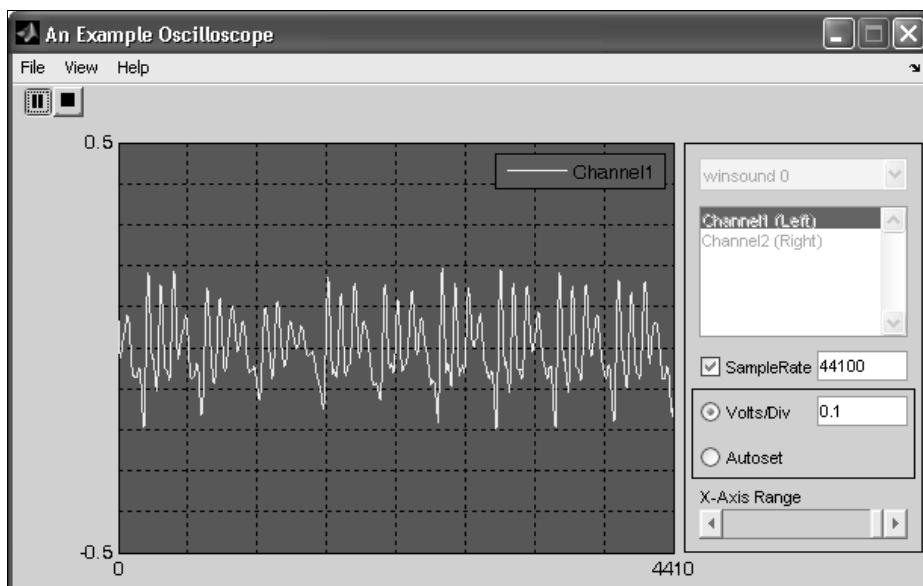


Рис. 3.36. Осциллограф — демонстрационный пример пакета Data Acquisition

### **ЗАМЕЧАНИЕ**

Более серьезный вариант осциллографа реализован в виде еще одной функции пакета Data Acquisition — `softscope`.

Второй пример — генератор сигналов различной формы. Имя программы для вызова из командной строки — `daqfcngen`. При использовании окна справочной системы после вызова команды меню **Help | Product Help** необходимо выбрать в оглавлении справки раздел `Data Acquisition Toolbox\Demos\GUI Demos\Analog Output Function Generator`.

Вид окна генератора приведен на рис. 3.37. В списке, расположеннном в левой части окна над полем графика, выбирается тип генерируемого сигнала:

- Sine** — гармонический сигнал;
- Sinc** — периодическая sinc-функция (функция Дирихле);

- Square** — последовательность прямоугольных импульсов;
- Triangle** — последовательность симметричных треугольных импульсов;
- Sawtooth** — последовательность пилообразных импульсов;
- Random** — случайный сигнал;
- Chirp** — колебания с плавно меняющейся частотой.

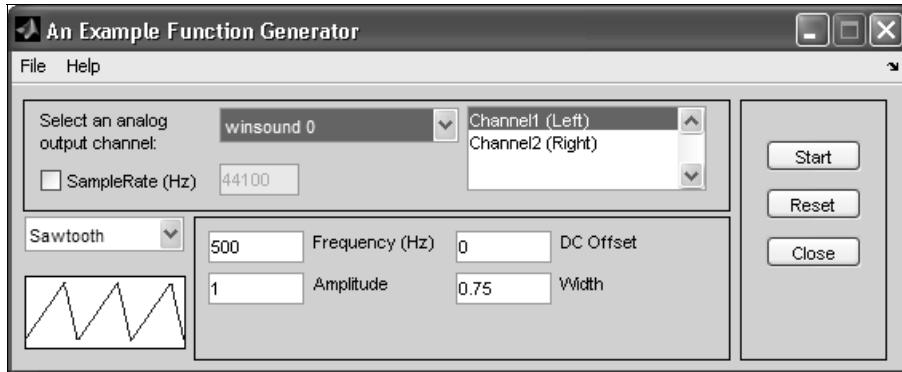
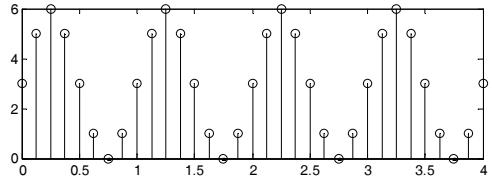


Рис. 3.37. Функциональный генератор — демонстрационный пример пакета Data Acquisition

Набор параметров, настраиваемых в нижней части окна, зависит от выбранного типа сигнала.

# ГЛАВА 4



## Дискретные системы

Теоретические сведения, приводимые в данной главе, в литературе довольно часто фигурируют под заголовками "Дискретные фильтры", "Принципы цифровой фильтрации" и т. п. (см., например, [1, 2, 4]). В принципе, "дискретная система" и "дискретный фильтр" — это одно и то же, однако понятие "фильтр", сознательно или подсознательно, тесно связывается с системами, которые одни частоты *пропускают*, а другие *задерживают* (см. рис. 2.9 в главе 2, где были показаны идеализированные АЧХ фильтров такого рода). Такой подход может создать ложное, ограниченное представление о назначении и возможностях дискретных линейных систем, которые способны выполнять и иные задачи, нежели выделение из сигнала определенной полосы частот. По этой причине в названии данной главы использован термин "дискретные системы". Однако в тексте главы слова "фильтр" и "система" будут использоваться как синонимы.

### ЗАМЕЧАНИЕ

Далее, в главе 6, где пойдет речь о расчете (синтезе) систем, реализующих заданные (во многих случаях действительно полосовые) частотные характеристики, будет употребляться исключительно слово "фильтр".

## Сущность линейной дискретной обработки

Вообще, *дискретный фильтр* — это произвольная система обработки дискретного сигнала, обладающая свойствами линейности и стационарности. Под этими свойствами понимается то же, что и в аналоговом случае (см. разд. "Классификация систем" главы 2): *линейность* означает, что выходная реакция на сумму сигналов равна сумме реакций на эти сигналы, поданные на вход по отдельности, а *стационарность* — что задержка входного сигнала приводит лишь к такой же задержке выходного сигнала, не меняя его формы.

### ЗАМЕЧАНИЕ

Существуют и фильтры с *переменными параметрами*, не обладающие свойством стационарности. Это, например, адаптивные фильтры, речь о которых пойдет в главе 9.

Любой фильтр обладает определенной частотной характеристикой. Чтобы она была нетривиальной, т. е. чтобы коэффициент передачи фильтра на разных частотах был разным, выходной сигнал фильтра  $y(k)$  должен зависеть от *нескольких* отсчетов входного сигнала  $x(k)$ . Таким образом, дискретный фильтр должен обладать *памятью*.

Чтобы обеспечить линейность и стационарность, производимые фильтром математические операции должны ограничиваться сложением и умножением на константы.

Рассмотрим простейший пример. Пусть выходной сигнал фильтра равен сумме двух последних отсчетов входного сигнала:

$$y(k) = x(k) + x(k - 1).$$

Убедимся в том, что эта система по-разному пропускает на выход сигналы разных частот. Для начала подадим на вход фильтра серию одинаковых отсчетов (т. е. сигнал нулевой частоты):

Вход	Выход
...	...
1	...
1	$1 + 1 = 2$
1	$1 + 1 = 2$
1	$1 + 1 = 2$
...	...

Как видите, уровень постоянного сигнала фильтр увеличил в два раза. Теперь подадим на вход отсчеты, одинаковые по модулю, но с чередующимися знаками (т. е. гармонический сигнал с частотой Найквиста):

Вход	Выход
...	...
1	...
-1	$-1 + 1 = 0$
1	$1 + (-1) = 0$
-1	$-1 + 1 = 0$
...	...

В отличие от постоянного сигнала, сигнал с частотой Найквиста на выход просто не прошел. Далее попробуем что-нибудь промежуточное, например сигнал с частотой, равной половине частоты Найквиста:

Вход	Выход
...	...
1	...
0	$0 + 1 = 1$
-1	$-1 + 0 = -1$
0	$0 + (-1) = -1$
1	$1 + 0 = 1$
0	$0 + 1 = 1$
-1	$-1 + 0 = -1$
...	...

На выходе в данном случае получаются отсчеты синусоиды, имеющей в  $\sqrt{2}$  раз большую амплитуду и некоторый фазовый сдвиг по сравнению с входным сигналом. Рассмотренный пример представляет собой простейший случай *нерекурсивного фильтра*. Такие фильтры суммируют некоторое число входных отсчетов, умножая их при этом на постоянные весовые коэффициенты.

Теперь заметим, что, помимо выходных отсчетов, мы можем использовать для вычислений и ранее рассчитанные значения *выходного* сигнала. Попробуем просто суммировать входной отсчет и предыдущий выходной отсчет:

$$y(k) = x(k) + y(k-1).$$

Подаем на вход постоянный сигнал (начальное состояние фильтра считаем нулевым):

Вход	Выход
1	$1 + 0 = 1$
1	$1 + 1 = 2$
1	$1 + 2 = 3$
1	$1 + 3 = 4$
...	...

Так, очевидно, будет продолжаться и далее — выходной сигнал будет линейно нарастать, что рано или поздно приведет к переполнению разрядной сетки вычислительного устройства. Это сразу же демонстрирует нам главную отличительную черту фильтров, использующих при вычислениях предыдущие отсчеты выходного сигнала (их называют *рекурсивными фильтрами*) — из-за наличия обратных связей они могут быть *неустойчивыми*.

Попробуем уменьшить влияние обратной связи, разделив предыдущий отсчет выходного сигнала на 2:

$$y(k) = x(k) + 0,5 y(k-1).$$

Снова подаем на вход постоянный сигнал:

Вход	Выход
1	$1 + 0,5 \cdot 0 = 1$
1	$1 + 0,5 \cdot 1 = 1,5$
1	$1 + 0,5 \cdot 1,5 = 1,75$
1	$1 + 0,5 \cdot 1,75 = 1,875$
1	$1 + 0,5 \cdot 1,875 = 1,9375$
...	...

Как видим, ситуация радикально изменилась — теперь выходной сигнал с уменьшающейся скоростью стремится к значению 2. Таким образом, переходный процесс в фильтре является *бесконечным*. Это еще одна отличительная черта рекурсивных фильтров.

Итак, рекурсивные фильтры суммируют при расчетах не только входные, но и некоторое количество предыдущих выходных отсчетов сигнала, умножая их при этом на постоянные весовые коэффициенты.

В общем случае дискретный фильтр суммирует (с весовыми коэффициентами) некоторое количество входных отсчетов (включая последний) и некоторое количество предыдущих выходных отсчетов:

$$y(k) = b_0x(k) + b_1x(k-1) + \dots + b_mx(k-m) - \\ - a_1y(k-1) - a_2y(k-2) - \dots - a_ny(k-n), \quad (4.1)$$

где  $a_j$  и  $b_i$  — вещественные коэффициенты.

Данная формула называется *алгоритмом дискретной фильтрации*. Если по-иному сгруппировать слагаемые, чтобы с одной стороны от знака равенства были только входные отсчеты, а с другой — только выходные, получим форму записи, называемую *разностным уравнением*:

$$y(k) + a_1y(k-1) + a_2y(k-2) + \dots + a_ny(k-n) = \\ = b_0x(k) + b_1x(k-1) + \dots + b_mx(k-m). \quad (4.2)$$

### ВНИМАНИЕ!

В ряде источников (см., например, [1, 2, 4]) обозначения  $a_i$  и  $b_i$  используются наоборот: коэффициенты  $a_i$  относятся к отсчетам *входного* сигнала, а коэффициенты  $b_i$  — *выходного*. Об этом следует помнить при работе с литературой.

Структура разностного уравнения похожа на структуру дифференциального уравнения аналоговой линейной системы (см. формулу (2.9) в разд. "Способы описания линейных систем" главы 2), только вместо операции дифференцирования в форму-

ле фигурируют задержки дискретных последовательностей. Как мы увидим далее, этим определяется и общность подходов к описанию аналоговых и дискретных систем. Однако весьма существенным является то, что в дискретной системе не существует каких-либо принципиальных ограничений на соотношение между  $t$  и  $n$  — количествами входных и выходных отсчетов, используемых при вычислениях.

## Способы описания дискретных систем

Дискретные системы, как и аналоговые, могут описываться различными способами. Благодаря сходству свойств  $z$ -преобразования со свойствами преобразований Лапласа и Фурье способы описания аналоговых и дискретных систем в основном похожи друг на друга.

Кроме того, практически каждый способ описания дискретной системы соответствует определенной структурной схеме из числа рассмотренных в разд. "Формы реализации дискретных фильтров" этой главы.

### Импульсная характеристика

В случае линейных систем с постоянными параметрами для анализа прохождения любого сигнала достаточно знать результат прохождения элементарного импульса в виде дельта-функции. Для дискретных систем также можно ввести в рассмотрение единичную импульсную функцию  $x_0(k)$  (см. разд. "Примеры вычисления  $z$ -преобразования" и формулу (3.19) в главе 3).

Выходная реакция на единичный импульс  $x_0(k)$ , определяемая при нулевых начальных условиях, называется *импульсной характеристикой* дискретной системы и обозначается  $h(k)$ .

Как и в случае линейных систем с постоянными параметрами, знание импульсной характеристики позволяет проанализировать прохождение через дискретную систему любого сигнала. Действительно, прежде всего заметим, что произвольный сигнал  $\{x(k)\}$  можно представить в виде линейной комбинации единичных отсчетов:

$$x(k) = \sum_{m=-\infty}^{\infty} x(m)x_0(k-m).$$

Выходной сигнал, исходя из линейности и стационарности рассматриваемой системы, должен представлять собой линейную комбинацию импульсных характеристик:

$$y(k) = \sum_{m=-\infty}^{\infty} x(m)h(k-m). \quad (4.3)$$

Выражение (4.3) называется *дискретной сверткой* (точнее, *дискретной линейной сверткой* — ее не следует путать с *круговой сверткой*, которая рассмотрена далее, при обсуждении свойств дискретного преобразования Фурье в главе 5). Для физи-

чески реализуемой системы  $h(k) = 0$  при  $k < 0$ , поэтому верхний предел суммирования в формуле (4.3) можно заменить на  $k$ :

$$y(k) = \sum_{m=-\infty}^k x(m)h(k-m).$$

Это означает, что система при вычислении очередного отсчета может оперировать только прошлыми значениями входного сигнала и еще ничего не знает о будущих.

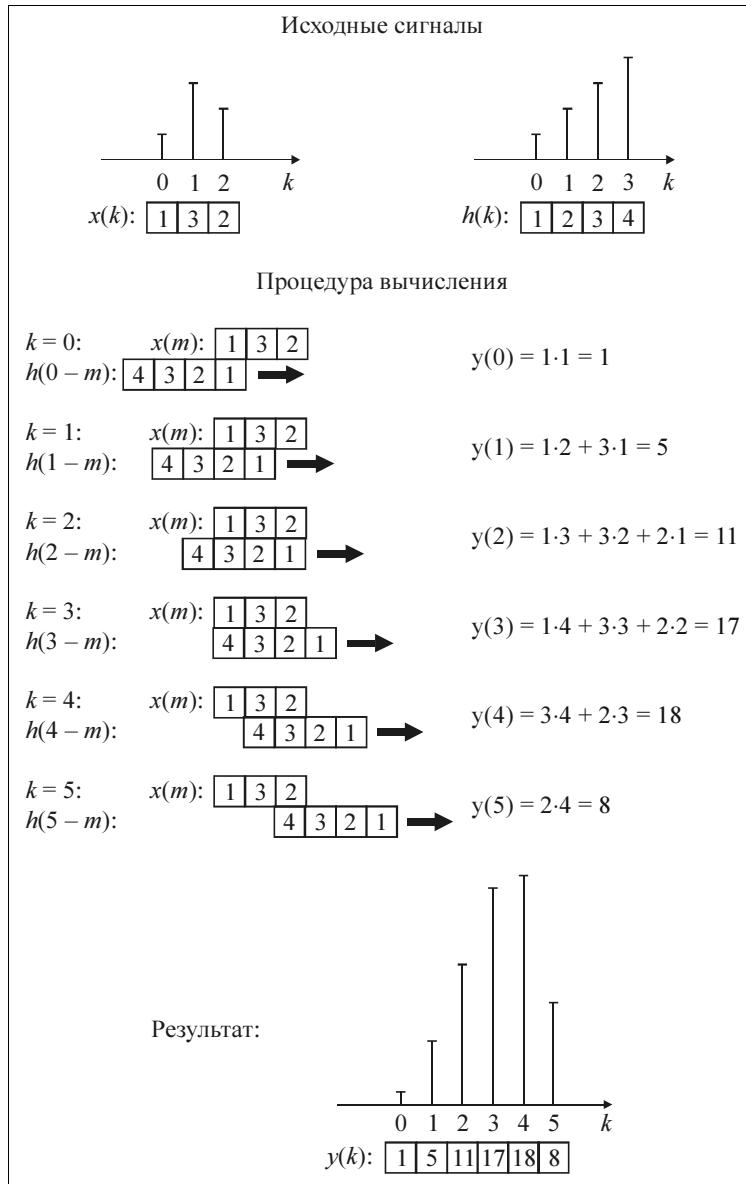


Рис. 4.1. Вычисление дискретной свертки

Пример вычисления дискретной свертки, подробно иллюстрирующий выполняемые при этом математические операции, приведен на рис. 4.1.

### ЗАМЕЧАНИЕ

Как и в аналоговом случае, мы будем называть *вещественными* дискретные системы, имеющие вещественную импульсную характеристику.

## Функция передачи

Применим рассмотренное в главе 3 *z*-преобразование к уравнению дискретной фильтрации (4.3). Так как это уравнение представляет собой дискретную свертку, то, согласно свойствам *z*-преобразования (см. формулу (3.25)), результатом будет являться произведение *z*-преобразований:

$$Y(z) = X(z) H(z). \quad (4.4)$$

Входящая в (4.4) функция  $H(z)$ , равная отношению *z*-преобразований выходного и входного сигналов и представляющая собой *z*-преобразование импульсной характеристики системы, называется *функцией передачи (transfer function)* или *системной функцией* дискретной системы:

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{\infty} h(k) z^{-k}. \quad (4.5)$$

Применив *z*-преобразование к обеим частям разностного уравнения (4.2), получим

$$\begin{aligned} Y(z)(1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_n z^{-n}) &= \\ &= X(z)(1 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-m}). \end{aligned}$$

Отсюда легко получить вид функции передачи:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \dots + b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_n z^{-n}}. \quad (4.6)$$

Таким образом, функция передачи *физически реализуемой* дискретной системы может быть представлена в виде отношения полиномов по отрицательным степеням переменной *z*.

## Частотная характеристика

Чтобы получить комплексный коэффициент передачи (частотную характеристику) дискретной системы, воспользуемся формулой (3.24) (см. разд. "Z-преобразование" главы 3), описывающей связь *z*-преобразования и преобразования Фурье:

$$\dot{K}(\omega) = H(e^{j\omega T}) = \sum_{k=0}^{\infty} h(k) e^{-j\omega kT}. \quad (4.7)$$

Из (4.7) видно, что частотная характеристика дискретной системы, так же как и спектры дискретизированных сигналов (см. формулу (3.2) в главе 3), является пе-

риодической функцией частоты с периодом, равным частоте дискретизации  $\omega_d = 2\pi/T$ .

## Нули и полюсы

Так же как и в аналоговом случае, разложив числитель и знаменатель функции передачи на множители, мы получим функцию передачи в следующем виде:

$$H(z) = k \frac{(1 - z_1 z^{-1})(1 - z_2 z^{-1}) \dots (1 - z_m z^{-1})}{(1 - p_1 z^{-1})(1 - p_2 z^{-1}) \dots (1 - p_n z^{-1})}. \quad (4.8)$$

Здесь  $k = b_0$  — коэффициент усиления (*gain*),  $z_i$  — нули функции передачи (*zero*),  $p_i$  — полюсы функции передачи (*pole*). В точках нулей  $H(z_i) = 0$ , а в точках полюсов  $H(p_i) \rightarrow \infty$ . Некоторая специфика формулы разложения связана лишь с тем, что при записи функции передачи дискретной системы используются отрицательные степени переменной  $z$ .

В данном случае дискретная система описывается набором параметров  $\{z_i\}$ ,  $\{p_i\}$ ,  $k$ .

Для вещественных систем нули функции передачи являются вещественными либо составляют комплексно-сопряженные пары. То же относится и к полюсам. Коэффициент усиления при этом всегда вещественный. В случае комплексных систем никаких ограничений на значения рассматриваемых параметров не накладывается.

Представление дискретной системы в виде наборов нулей и полюсов соответствует последовательной (каскадной) форме ее реализации (см. разд. "Формы реализации дискретных фильтров" далее в этой главе).

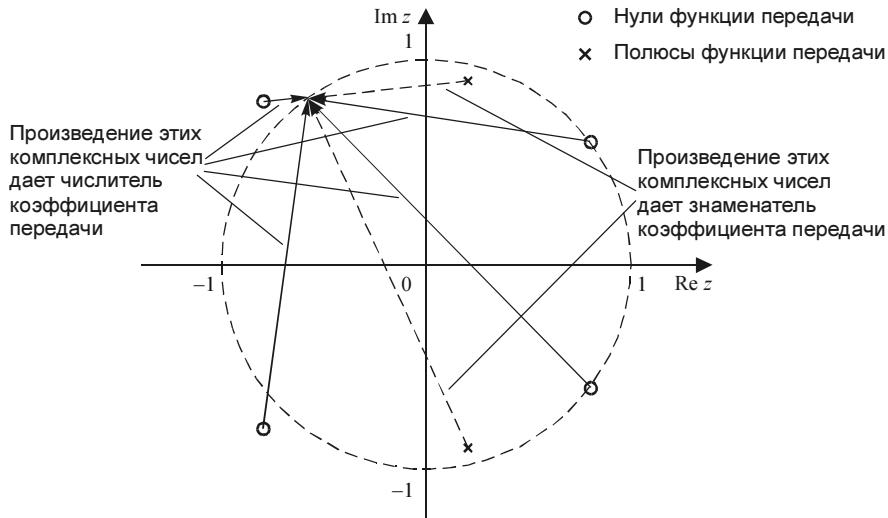
## Связь АЧХ с расположением нулей и полюсов

Подобно тому как это было сделано в главе 2 для аналоговых систем, мы можем установить связь между расположением нулей и полюсов функции передачи на комплексной плоскости и формой АЧХ системы. Отличие от аналогового случая заключается в том, что при расчете частотной характеристики точка, изображающая аргумент функции передачи на комплексной плоскости, движется не вдоль мнимой оси, а по единичной окружности:  $z = \exp(j\omega T)$  (рис. 4.2). Следовательно, выводы, сделанные в главе 2, для дискретной системы формулируются следующим образом:

- когда точка  $z = \exp(j\omega T)$  находится вблизи одного из нулей функции передачи  $z_i$ , соответствующая разность  $(z - z_i)$  окажется малой по сравнению с другими, в результате чего АЧХ в данной области частот будет иметь *провал*. Если нуль лежит на единичной окружности, АЧХ на соответствующей частоте будет иметь нулевое значение;
- когда точка  $z = \exp(j\omega T)$  находится вблизи одного из полюсов функции передачи  $p_i$ , соответствующая разность  $(z - p_i)$  окажется малой по сравнению с другими, в результате чего АЧХ в данной области частот будет иметь *подъем*. Если

полюс лежит *на* единичной окружности, АЧХ на соответствующей частоте будет стремиться к бесконечности. Забегая немного вперед, скажем, что такая система находится *на грани устойчивости*;

- чем ближе к единичной окружности расположен нуль (полюс), тем более выраженным будет соответствующий провал (подъем) АЧХ.



**Рис. 4.2.** Влияние расположения нулей и полюсов на форму АЧХ дискретной системы

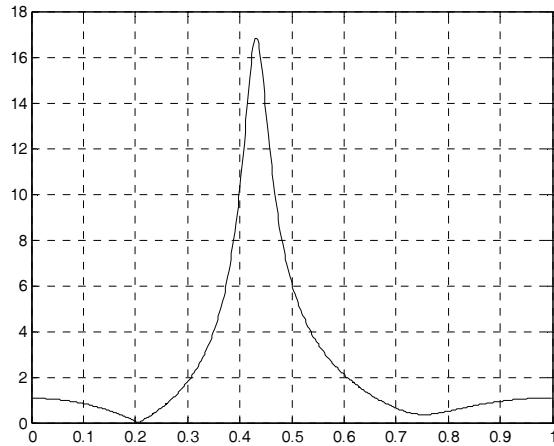
В качестве примера рассмотрим систему, имеющую пару полюсов, равных  $0,2 \pm 0,9j$ , и четыре нуля, два из которых  $(0,8 \pm 0,6j)$  расположены *на* единичной окружности, а еще два  $(-0,8 \pm 0,8j)$  — вблизи нее (см. рис. 4.2). Согласно перечисленным ранее принципам, АЧХ этой системы должна иметь пик в районе частоты 0,45, нуль в районе частоты 0,2 и провал в районе частоты 0,75 (приведены значения частот, нормированные к частоте Найквиста). Чтобы проверить это, построим график АЧХ системы:

```
>> z = [0.8+0.6i -0.8+0.8i]'; % нули функции передачи (столбец)
>> z = [z; conj(z)]; % создаем комплексно-сопряженные пары
>> p = 0.2+0.9i; p = [p; conj(p)]; % полюсы функции передачи (столбец)
>> k = 1; % общий коэффициент усиления
>> [b, a] = zp2tf(z, p, k); % ищем функцию передачи
>> [h, f] = freqz(b, a); % расчет коэффициента передачи
>> plot(f/pi, abs(h)) % график АЧХ
>> grid on
```

Результат, показанный на рис. 4.3, полностью подтверждает сделанные предположения.

Как и в аналоговом случае, при расположении нулей рядом с полюсами их влияние на АЧХ может взаимно компенсироваться. А вот свойство, соответствующее симметричному (относительно мнимой оси) расположению нуля и полюса аналоговой

системы, формулируется более сложным образом и будет подробно рассмотрено в следующем разделе.



**Рис. 4.3.** АЧХ цепи подтверждает предсказания, сделанные исходя из расположения нулей и полюсов

## Всепропускающие фильтры

Всепропускающими (*allpass filter*) называют фильтры, АЧХ которых равна единице на всех частотах. Такие фильтры изменяют только фазы спектральных составляющих входного сигнала и могут использоваться, например, для линеаризации ФЧХ дискретных систем (т. е. для выравнивания групповой задержки, вносимой системой).

### ЗАМЕЧАНИЕ

Используются также термины "*фазовый фильтр*" (*phase filter*) и "*фазовое звено*".

Представление функции передачи дискретной системы в виде разложения на множители позволяет легко получить условия, при выполнении которых рекурсивный фильтр будет всепропускающим. Рассмотрим функцию передачи фильтра первого порядка, имеющего один полюс  $p_1$  и один нуль  $z_1$  (в общем случае все три параметра  $z_1$ ,  $p_1$  и  $k$  являются комплексными):

$$H(z) = k \frac{1 - z_1 z^{-1}}{1 - p_1 z^{-1}}. \quad (4.9)$$

Квадрат АЧХ такого фильтра равен

$$\begin{aligned} |\dot{K}(\omega)|^2 &= H(e^{j\omega T}) H^*(e^{j\omega T}) = |k|^2 \frac{(1 - z_1 e^{-j\omega T})(1 - z_1^* e^{j\omega T})}{(1 - p_1 e^{-j\omega T})(1 - p_1^* e^{j\omega T})} = \\ &= |k|^2 \frac{1 + z_1 z_1^* - z_1 e^{-j\omega T} - z_1^* e^{j\omega T}}{1 + p_1 p_1^* - p_1 e^{-j\omega T} - p_1^* e^{j\omega T}}. \end{aligned} \quad (4.10)$$

Мы хотим, чтобы квадрат АЧХ был равен единице на всех частотах. Для этого необходимо, чтобы коэффициенты при всех степенях  $e^{j\omega T}$  в числителе и знаменателе (4.10) были одинаковы:

$$|k|^2(1 + z_1 z_1^*) = 1 + p_1 p_1^*, \quad |k|^2 z_1 = p_1, \quad |k|^2 z_1^* = p_1^*.$$

Если подставить в первое уравнение значение  $p_1$  из второго уравнения, получится квадратное уравнение относительно  $|k|^2$ . Его решениями являются  $|k|^2 = 1$  (тривиальный случай  $|k| = 1$  и  $z_1 = p_1$ ) и  $|k|^2 = 1/|z_1|^2$ . Второе решение и дает интересующий нас результат:

$$z_1 = \frac{1}{p_1^*}, \quad |k| = \sqrt{p_1 p_1^*} = |p_1|. \quad (4.11)$$

Фаза коэффициента усиления ( $\arg k$ ) может быть произвольной, поскольку она не влияет на АЧХ фильтра.

Понятно, что если включить последовательно произвольное количество фильтров, удовлетворяющих (4.11), то АЧХ системы в целом будет по-прежнему равна единице на всех частотах. Таким образом мы получаем условия, при которых фильтр является всепропускающим:

- число нулей равно числу полюсов;
- значения нулей являются обратными и комплексно-сопряженными по отношению к полюсам (иными словами,  $\arg p_i = \arg z_i$  и  $|p_i| \cdot |z_i| = 1$ );
- коэффициент усиления  $k$  равен произведению модулей полюсов фильтра.

Убедимся в справедливости сказанного, рассчитав АЧХ и ФЧХ фильтра, удовлетворяющего указанным условиям (рис. 4.4):

```
>> p = [0.5+0.5i; 0.5-0.5i; -0.7]; % полюсы фильтра
>> z = 1./conj(p); % нули фильтра
>> k = prod(abs(p)); % коэффициент усиления
>> [b, a] = zp2tf(z, p, k); % функция передачи
>> freqz(b, a) % частотные характеристики
```

Как видите, фильтр имеет единичную АЧХ (0 дБ) и нелинейную ФЧХ. Регулируя количество и расположение полюсов, можно получить ФЧХ весьма сложной формы.

### **ЗАМЕЧАНИЕ**

В пакете Filter Design имеется функция `iirgrpdelay`, позволяющая рассчитывать всепропускающие фильтры с заданной зависимостью групповой задержки от частоты (см. разд. "Функции пакета *Filter Design*" главы 6).

Помимо условия (4.11) для нулей, полюсов и коэффициента усиления, всепропускающие фильтры обладают очень простой связью между коэффициентами полинома

мов числителя и знаменателя функции передачи. Подставив (4.11) в формулу для функции передачи (4.9), получим

$$\begin{aligned} H(z) &= |p_1| e^{j \arg k} \frac{1 - \frac{1}{p_1^*} z^{-1}}{1 - p_1 z^{-1}} = p_1^* e^{j(\arg k + \arg p_1)} \frac{1 - \frac{1}{p_1^*} z^{-1}}{1 - p_1 z^{-1}} = \\ &= e^{j(\pi + \arg k + \arg p_1)} \frac{-p_1^* + z^{-1}}{1 - p_1 z^{-1}}, \end{aligned}$$

где  $\arg p_1$  — фаза полюса  $p_1$ .

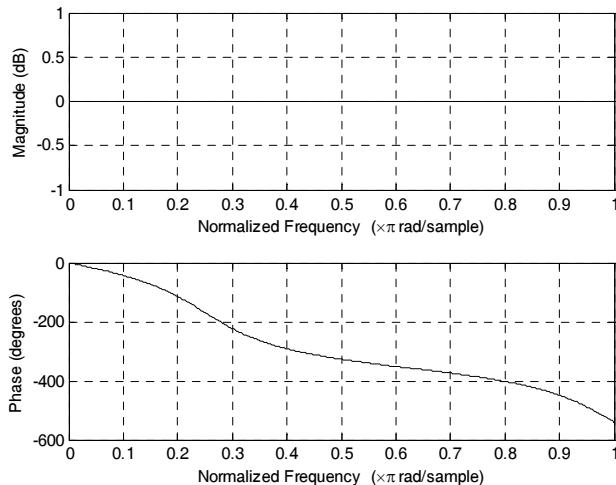


Рис. 4.4. АЧХ и ФЧХ всепропускающего фильтра

Итак, в случае фильтра первого порядка векторы коэффициентов полиномов числителя и знаменателя функции передачи связаны между собой комплексным сопряжением и зеркальным отражением (т. е. имеют противоположный порядок следования элементов). При этом функция передачи может иметь произвольный дополнительный комплексный множитель с единичным модулем.

Легко проверить, что зеркальное соотношение между векторами коэффициентов полиномов числителя и знаменателя сохранится и для фильтров произвольного порядка. Таким образом, в общем случае функция передачи всепропускающего фильтра имеет вид

$$H(z) = e^{j\varphi} \frac{a_n^* + a_{n-1}^* z^{-1} + a_{n-2}^* z^{-2} + \dots + a_1^* z^{-(n-1)} + z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_{n-1} z^{-(n-1)} + a_n z^{-n}},$$

где  $\varphi$  — произвольный угол. В вещественном случае формула становится еще проще:

$$H(z) = \pm \frac{a_n + a_{n-1} z^{-1} + a_{n-2} z^{-2} + \dots + a_1 z^{-(n-1)} + z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_{n-1} z^{-(n-1)} + a_n z^{-n}}.$$

Проверим это свойство, выведя на экран значения векторов  $b$  и  $a$ , полученных в предыдущем примере:

```
>> b, a
b =
    0.3500   -0.2000   -0.3000    1.0000
a =
    1.0000   -0.3000   -0.2000    0.3500
```

Как видим, зеркальное соотношение между векторами коэффициентов полиномов числителя и знаменателя функции передачи действительно имеет место.

## Полюсы и вычеты

Идея представления дробно-рациональной функции передачи в виде суммы простых дробей была рассмотрена в главе 2 применительно к аналоговым системам. В дискретном случае такое представление имеет несколько иной вид и при отсутствии кратных корней у знаменателя может быть записано следующим образом:

$$H(z) = \frac{r_1}{1 - p_1 z^{-1}} + \frac{r_2}{1 - p_2 z^{-1}} + \dots + \frac{r_n}{1 - p_n z^{-1}} + k_0 + k_1 z^{-1} + \dots + k_{m-n} z^{-(m-n)}. \quad (4.12)$$

Здесь, как и в аналоговом случае,  $p_i$  и  $r_i$  — полюсы функции передачи и соответствующие им *вычеты*. Поскольку на соотношение степеней полиномов числителя и знаменателя в дискретном случае не накладывается никаких ограничений, целая часть функции передачи, представленная коэффициентами  $k_i$ , может содержать не только константу.

В данном случае цепь описывается набором параметров  $\{r_i\}$ ,  $\{p_i\}$ ,  $\{k_i\}$ .

Для вещественных систем полюсы функции передачи являются вещественными либо составляют комплексно-сопряженные пары. Вычеты, соответствующие комплексно-сопряженным полюсам, при этом также являются комплексно-сопряженными.

Если полюс  $p_i$  имеет кратность  $m$ , то в разложении на простые дроби он порождает  $m$  слагаемых следующего вида:

$$\frac{r_{i1}}{1 - p_i z^{-1}} + \frac{r_{i2}}{(1 - p_i z^{-1})^2} + \dots + \frac{r_{im}}{(1 - p_i z^{-1})^m}. \quad (4.13)$$

Представление дискретной системы в виде наборов полюсов и вычетов соответствует параллельной форме ее реализации (см. разд. "Формы реализации дискретных фильтров" далее в этой главе).

## Расчет импульсной характеристики

Представление функции передачи в виде суммы простых дробей позволяет вычислить импульсную характеристику системы, поскольку каждое слагаемое функции

передачи вида  $r_i/(1-p_i z^{-1})$ , согласно формуле (3.22), соответствует экспоненциальному слагаемому импульсной характеристики вида

$$r_i p_i^k, \quad k \geq 0.$$

Пара комплексно-сопряженных полюсов дает пару слагаемых импульсной характеристики в виде комплексно-сопряженных экспонент. Сумма таких слагаемых представляет собой синусоиду с экспоненциально меняющейся амплитудой:

$$\begin{aligned} r_i(p_i)^n + r_i^*(p_i^*)^n &= 2 \operatorname{Re}[r_i(p_i)^n] = 2 \operatorname{Re}\left[|r_i| \cdot |p_i|^n \exp(j(\arg r_i + n \arg p_i))\right] = \\ &= 2|r_i| \cdot |p_i|^n \cos(n \arg p_i + \arg r_i). \end{aligned}$$

Здесь  $\arg r_i$  и  $\arg p_i$  — фазы комплексных чисел  $r_i$  и  $p_i$ .

Что касается кратных полюсов, то  $m$ -кратный полюс  $p_i$  даст в выражении для импульсной характеристики  $m$  слагаемых следующего вида:

$$A_0 p_i^k + A_1 k p_i^k + A_2 k^2 p_i^k + \dots + A_{m-1} k^{m-1} p_i^k, \quad k \geq 0.$$

Коэффициенты  $A_j$  представляют собой рациональные дроби и для каждой конкретной кратности  $k$  могут быть рассчитаны индивидуально. Вот несколько конкретных примеров:

$$\begin{aligned} \frac{A}{(1-pz^{-1})^2} &\leftrightarrow A(1+k)p^k, \quad k \geq 0, \\ \frac{A}{(1-pz^{-1})^3} &\leftrightarrow A\left(1+\frac{3}{2}k+\frac{1}{2}k^2\right)p^k, \quad k \geq 0, \\ \frac{A}{(1-pz^{-1})^4} &\leftrightarrow A\left(1+\frac{11}{6}k+k^2+\frac{1}{6}k^3\right)p^k, \quad k \geq 0. \end{aligned}$$

## Устойчивость дискретных систем

При отсутствии входного сигнала в дискретной системе могут существовать *свободные колебания*. Их вид зависит от начальных условий, т. е. значений, хранящихся в элементах памяти системы в момент отключения входного сигнала. Система называется *устойчивой*, если при любых начальных условиях свободные колебания являются затухающими, т. е. если при  $x(k) = 0$

$$\lim_{k \rightarrow \infty} y(k) = 0.$$

Любой сигнал на выходе линейной стационарной системы представляет собой линейную комбинацию ее задержанных во времени импульсных характеристик. Поэтому для затухания свободных колебаний необходимо, чтобы была затухающей импульсная характеристика системы  $h(k)$ :

$$\lim_{k \rightarrow \infty} h(k) = 0.$$

В предыдущем разделе было показано, что импульсная характеристика системы в общем случае содержит слагаемые вида

$$Ap_i^k k^n,$$

где  $p_i$  — полюсы функции передачи системы,  $n$  — неотрицательные целые числа, меньшие кратности полюса  $p_i$ ,  $A$  — некая константа.

Такие слагаемые при  $k \rightarrow \infty$  затухают, если полюс  $p_i$  по модулю меньше единицы:

$$|p_i| < 1. \quad (4.14)$$

Теперь мы можем окончательно сформулировать условие устойчивости: чтобы дискретная система была устойчива, полюсы функции передачи должны находиться на комплексной плоскости внутри круга единичного радиуса.

## Пространство состояний

Сущность представления дискретной системы в пространстве состояний та же, что и в аналоговом случае — имеется вектор параметров, описывающих внутреннее состояние системы, и две формулы, согласно которым производится изменение этого состояния и формирование выходного сигнала в зависимости от текущего состояния и входного сигнала:

$$\begin{aligned} s(k+1) &= As(k) + Bx(k), \\ y(k) &= Cs(k) + Dx(k). \end{aligned}$$

Здесь  $s(k)$  — вектор состояния,  $x(k)$  и  $y(k)$  — соответственно отсчеты входного и выходного сигналов,  $A$ ,  $B$ ,  $C$  и  $D$  — параметры, описывающие систему. Если  $x$  и  $y$  — скалярные сигналы и размерность вектора состояния равна  $N$ , то размерность параметров будет следующей:  $A$  — матрица  $N \times N$ ,  $B$  — столбец  $N \times 1$ ,  $C$  — строка  $1 \times N$ ,  $D$  — скаляр. Если входной и/или выходной сигналы являются векторными, размерность матриц соответствующим образом изменяется.

Преобразование параметров пространства состояний в функцию передачи осуществляется по формуле, аналогичной (2.17), приведенной в главе 2 применительно к аналоговым системам:

$$H(z) = C(zI - A)^{-1}B + D.$$

Из этой формулы видно, в частности, что полюсы функции передачи являются собственными числами матрицы  $A$ .

Как указывалось ранее применительно к аналоговым системам, преобразование коэффициентов функции передачи в параметры пространства состояний не является однозначным. Например, двум возможным вариантам представления одной и той же функции передачи в пространстве состояний соответствуют каноническая и транспонированная формы реализации соответствующего фильтра (см. разд. "Формы реализации дискретных фильтров" далее в этой главе).

## Фильтры первого и второго порядка

На практике рекурсивные фильтры часто строятся в виде последовательно (каскадно) или параллельно включенных секций первого и второго порядка, поэтому важно понимать, какими свойствами обладают такие секции. В данном разделе будут рассмотрены основные свойства дискретных фильтров первого и второго порядка с вещественными коэффициентами. Итак, речь пойдет о системах с функцией передачи следующего вида:

□ фильтр первого порядка:

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}; \quad (4.15)$$

□ фильтр второго порядка:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \quad (4.16)$$

Коэффициенты  $b_0, b_1, b_2, a_1, a_2$  в данном разделе будут считаться вещественными.

### Фильтры первого порядка

Функция передачи (4.15) имеет единственный полюс  $p_1 = -a_1$ . Сопоставляя это с условием устойчивости (4.14), сразу же получаем, что система первого порядка будет устойчива при  $|a_1| < 1$ .

Нуль функции передачи первого порядка также единственный:  $z_1 = -b_1/b_0$ .

Импульсная характеристика системы первого порядка имеет экспоненциальный вид. Для ее нахождения представим функцию передачи (4.15) в форме (4.12), выделив целую часть:

$$H(z) = \frac{b_1}{a_1} + \frac{b_0 - \frac{b_1}{a_1}}{1 + a_1 z^{-1}}.$$

Сравнивая это с приведенными в главе 2 примерами вычисления  $z$ -преобразования, получаем

$$h(k) = \begin{cases} 0, & k < 0, \\ b_0, & k = 0, \\ \left( b_0 - \frac{b_1}{a_1} \right) (-a_1)^k, & k > 0. \end{cases}$$

Частотная характеристика и ее модуль (т. е. АЧХ) для фильтра первого порядка рассчитываются следующим образом:

$$\dot{K}(\omega) = H(e^{j\omega T}) = \frac{b_0 + b_1 e^{-j\omega T}}{1 + a_1 e^{-j\omega T}},$$

$$|\dot{K}(\omega)| = \left| \frac{b_0 + b_1 e^{-j\omega T}}{1 + a_1 e^{-j\omega T}} \right| = \sqrt{\frac{b_0^2 + b_1^2 + 2b_0 b_1 \cos(\omega T)}{1 + a_1^2 + 2a_1 \cos(\omega T)}}.$$

Можно показать, что в рабочем диапазоне частот от нуля до частоты Найквиста АЧХ фильтра либо монотонно возрастает, либо монотонно убывает. С взаимным расположением нуля и полюса фильтра это связано следующим образом:

- при  $a_1 < 0$  (т. е. если полюс  $p_1 > 0$ ), АЧХ с частотой *убывает*, если нуль  $z_1 < p_1$  или  $z_1 > 1/p_1$ , и *возрастает*, если  $p_1 < z_1 < 1/p_1$ ;
- при  $a_1 > 0$  (т. е. если полюс  $p_1 < 0$ ), АЧХ с частотой *возрастает*, если нуль  $z_1 < 1/p_1$  или  $z_1 > p_1$ , и *убывает*, если  $1/p_1 < z_1 < p_1$ .

### ЗАМЕЧАНИЕ

При  $z_1 = p_1$  функция передачи представляет собой константу, а при  $z_1 = 1/p_1$  получается всепропускающий фильтр.

Таким образом, с помощью системы первого порядка можно получить простейший фильтр нижних или верхних частот, но не полосовой или режекторный фильтр.

Значения АЧХ фильтра на ключевых частотах (нулевой частоте, частоте Найквиста  $\pi/T$  и половине частоты Найквиста  $\pi/(2T)$ ) определяются следующим образом:

$$|\dot{K}(0)| = \frac{b_0 + b_1}{1 + a_1}, \quad \left| \dot{K}\left(\frac{\pi}{2T}\right) \right| = \sqrt{\frac{b_0^2 + b_1^2}{1 + a_1^2}}, \quad \left| \dot{K}\left(\frac{\pi}{T}\right) \right| = \frac{b_0 - b_1}{1 - a_1}.$$

Рассмотрим свойства простейших фильтров нижних и верхних частот первого порядка. Их описания легко получить исходя из того, что ФНЧ должен иметь нулевое значение АЧХ на максимальной рабочей частоте (т. е. на частоте Найквиста), а ФВЧ — на нулевой частоте. Отсюда сразу же следует, что функция передачи ФНЧ первого порядка должна иметь нуль при  $z = -1$ :

$$H(z) = \frac{1-p}{2} \frac{1+z^{-1}}{1-pz^{-1}}, \quad -1 < p < 1.$$

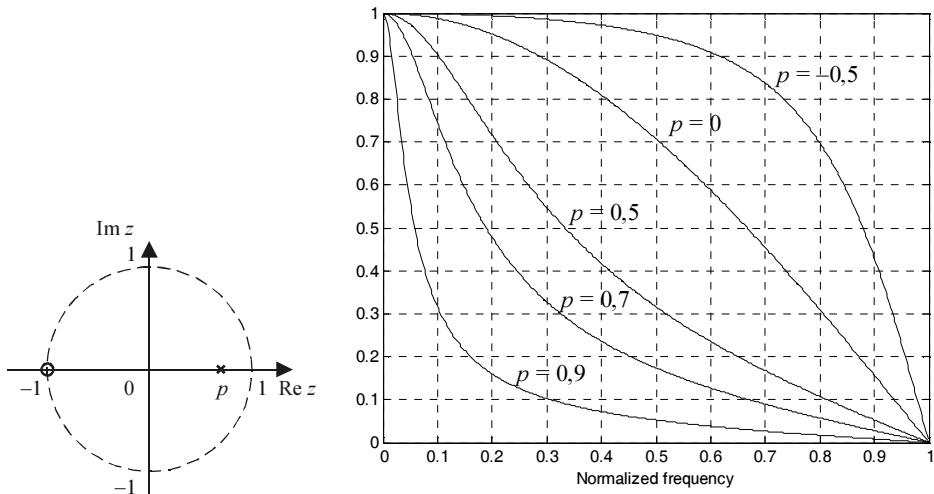
Постоянный множитель в этой функции передачи выбран так, чтобы обеспечить единичный коэффициент передачи на нулевой частоте, а показанное справа неравенство представляет собой приведенное ранее условие устойчивости системы первого порядка.

Параметр  $p$ , входящий в выражение для функции передачи, регулирует полосу пропускания ФНЧ — чем ближе  $p$  к единице, тем уже полоса пропускания. На рис. 4.5 показано расположение нулей и полюсов ФНЧ, а также приведены графики АЧХ рассматриваемого фильтра, полученные при различных значениях  $p$ . Для получения частотных характеристик использован следующий код MATLAB:

```

>> b = [1 1]; % числитель функции передачи
>> p = [-0.5 0 0.5 0.7 0.9]; % вектор значений параметра p
>> hold off
>> for k = 1:length(p)
>> a = [1 -p(k)]; % цикл по значениям p
>> k0 = (1-p(k))/2; % знаменатель функции передачи
>> [h(:,k), f] = freqz(b*k0, a); % постоянный множитель
>> plot(f/pi, abs(h(:,k))) % расчет частотной характеристики
>> plot(f/pi, abs(h(:,k))) % график АЧХ
>> hold on
>> end
>> hold off
>> grid on
>> xlabel('Normalized frequency')

```



**Рис. 4.5.** Фильтр нижних частот первого порядка: расположение нулей и полюсов (слева) и АЧХ при различных значениях параметра  $p$  (справа)

Аналогичный подход позволяет получить и функцию передачи простейшего фильтра верхних частот: она должна иметь нуль при  $z = 1$ , чтобы обеспечить нулевой коэффициент передачи на нулевой частоте. В результате получаем следующее (постоянный коэффициент, как и в предыдущем случае, выбран так, чтобы пиковое значение АЧХ было равно единице):

$$H(z) = \frac{1-p}{2} \frac{1-z^{-1}}{1+pz^{-1}}, \quad -1 < p < 1.$$

Параметр  $p$ , как и раньше, регулирует полосу пропускания фильтра. На рис. 4.6 показано расположение нулей и полюсов ФВЧ, а также приведены графики АЧХ рассматриваемого фильтра, полученные при различных значениях  $p$ . Для получения частотных характеристик использован следующий код MATLAB:

```

>> b = [1 -1]; % числитель функции передачи
>> p = [-0.5 0 0.5 0.7 0.9]; % вектор значений параметра p
>> hold off
>> for k = 1:length(p)
>> a = [1 p(k)]; % цикл по значениям p
>> k0 = (1-p(k))/2; % знаменатель функции передачи
>> [h(:,k), f] = freqz(b*k0, a); % постоянный множитель
>> plot(f/pi, abs(h(:,k))) % расчет частотной характеристики
>> hold on
>> xlabel('Normalized frequency')
>> end
>> hold off
>> grid on

```

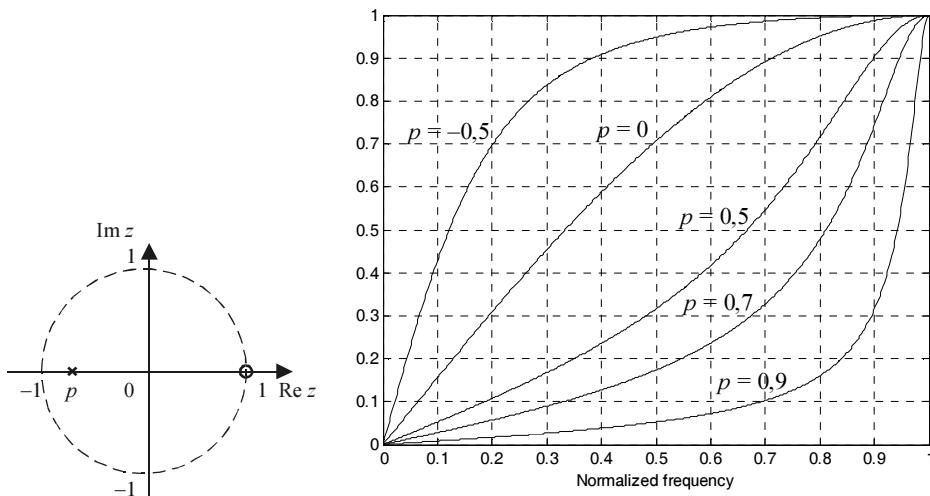


Рис. 4.6. Фильтр верхних частот первого порядка: расположение нулей и полюсов (слева) и АЧХ при различных значениях параметра  $p$  (справа)

## Условие устойчивости для систем второго порядка

В данном разделе мы определим, при каких условиях является устойчивой вещественная дискретная система второго порядка с функцией передачи (4.16). Согласно общему критерию (4.14), полюсы функции передачи устойчивого фильтра должны быть меньше единицы по модулю. Здесь следует отдельно рассмотреть два случая — вещественных и комплексных корней знаменателя. Характер корней определяется знаком дискриминанта квадратного уравнения

$$1 + a_1 z^{-1} + a_2 z^{-2} = 0, \text{ или } z^2 + a_1 z + a_2 = 0:$$

- $a_1^2 \geq 4a_2$  — корни вещественные;
- $a_1^2 < 4a_2$  — корни комплексные.

В комплексном случае оба полюса имеют одинаковые модули, равные  $a_2$ . Таким образом, область устойчивости при этом имеет вид

$$a_2 < 1, \quad (4.17)$$

а приведенное выше условие комплексного характера полюсов дает ограничение величины  $a_1$ :

$$|a_1| < 2. \quad (4.18)$$

В вещественном случае нужно рассмотреть значения обоих полюсов, определяемых по обычной формуле корней квадратного уравнения:

$$z = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2}}{2}.$$

Легко убедиться, что при любом сочетании знаков и абсолютных величин коэффициентов  $a_1$  и  $a_2$  больший из двух модулей полюсов равен

$$|z|_{\max} = \frac{|a_1| + \sqrt{a_1^2 - 4a_2}}{2}.$$

Решение неравенства  $|z|_{\max} < 1$  дает

$$a_2 > |a_1| - 1. \quad (4.19)$$

Объединяя (4.17), (4.18) и (4.19), получаем

$$\begin{cases} |a_1| < 2, \\ |a_1| - 1 < a_2 < 1. \end{cases} \quad (4.20)$$

Графически область устойчивости фильтра второго порядка показана на рис. 4.7. Пунктирной линией на этом рисунке изображена граница, разделяющая области вещественных и комплексных полюсов функции передачи.

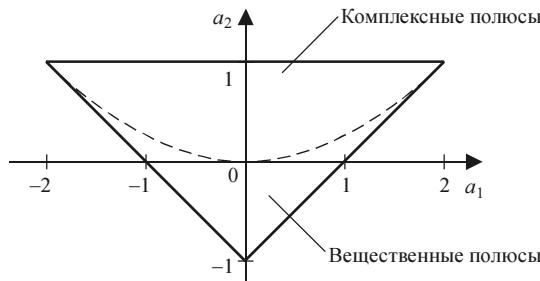


Рис. 4.7. Область устойчивости рекурсивного фильтра второго порядка

## Резонатор второго порядка

Дискретным *резонатором второго порядка* называется система, аналогичная по своим свойствам одиночному колебательному контуру, т. е. представляющая собой

простейший полосовой фильтр. Иными словами, АЧХ резонатора обладает следующими свойствами:

- на нулевой частоте АЧХ равна нулю:  $\dot{K}(0) = 0$ ;
- на частоте Найквиста АЧХ равна нулю:  $\dot{K}(\omega_d / 2) = 0$ ;
- на некоторой частоте  $\omega_0$  АЧХ достигает максимума, равного единице:  $|\dot{K}(\omega_0)| = 1$ .

Кроме того, очевидно, что существует множество систем, АЧХ которых удовлетворяет перечисленным требованиям, но имеет различную "остроту" максимума, т. е. разную полосу пропускания. Таким образом, полоса пропускания, измеренная по заданному уровню АЧХ, наряду со средней частотой  $\omega_0$  является исходным параметром при расчете резонатора.

Выведем формулы, связывающие вещественные коэффициенты фильтра  $b_0, b_1, b_2, a_1, a_2$ , входящие в выражение для функции передачи (4.16), с центральной частотой и шириной полосы пропускания резонатора.

Первые два из перечисленных требований выполнить очень просто — они означают, что функция передачи резонатора должна иметь нули в точках единичной окружности, соответствующих нулевой частоте (при этом  $z = e^{j0} = 1$ ) и частоте Найквиста ( $z = e^{j(\pi/T)T} = e^{j\pi} = -1$ ). Отсюда сразу же получаем, что числитель функции передачи должен иметь вид

$$b_0(1 - z^{-1})(1 + z^{-1}) = b_0(1 - z^{-2}),$$

а вся функция передачи может быть записана как

$$H(z) = b_0 \frac{1 - z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \quad (4.21)$$

Теперь займемся выполнением третьего требования к АЧХ. Комплексный коэффициент передачи с учетом (4.21) имеет вид

$$\dot{K}(\omega) = H(e^{j\omega T}) = b_0 \frac{1 - e^{-j2\omega T}}{1 + a_1 e^{-j\omega T} + a_2 e^{-j2\omega T}}.$$

Определим, на какой частоте (точнее, при каком значении произведения  $\omega T$ ) АЧХ резонатора имеет максимум. Для упрощения расчетов будем искать максимум *квадрата* АЧХ, который может быть записан следующим образом:

$$\begin{aligned} |\dot{K}(\omega)|^2 &= \dot{K}(\omega)\dot{K}^*(\omega) = |b_0|^2 \frac{(1 - e^{-j2\omega T})(1 - e^{j2\omega T})}{(1 + a_1 e^{-j\omega T} + a_2 e^{-j2\omega T})(1 + a_1 e^{j\omega T} + a_2 e^{j2\omega T})} = \\ &= |b_0|^2 \frac{2 - 2\cos(2\omega T)}{1 + a_1^2 + a_2^2 + 2a_1(a_2 + 1)\cos(\omega T) + 2a_2\cos(2\omega T)}. \end{aligned}$$

Для удобства дальнейших вычислений преобразуем косинус двойного угла по формуле  $\cos(2\omega T) = 2\cos^2(\omega T) - 1$ . С учетом этого

$$|\dot{K}(\omega)|^2 = |b_0|^2 \frac{4 - 4\cos^2(\omega T)}{1 + a_1^2 + a_2^2 - 2a_2 + 2a_1(a_2 + 1)\cos(\omega T) + 4a_2\cos^2(\omega T)}.$$

Произведем замену переменной  $w = \cos(\omega T)$  и продифференцируем по  $w$  квадрат АЧХ:

$$|\dot{K}(w)|^2 = 4|b_0|^2 \frac{1 - w^2}{1 + a_1^2 + a_2^2 - 2a_2 + 2a_1(a_2 + 1)w + 4a_2w^2}, \quad (4.22)$$

$$\begin{aligned} & \frac{d}{dw} |\dot{K}(w)|^2 = \\ &= 4|b_0|^2 \frac{-2w(1 + a_1^2 + a_2^2 - 2a_2 + 2a_1(a_2 + 1)w + 4a_2w^2) - (1 - w^2)(2a_1(a_2 + 1) + 8a_2w)}{(1 + a_1^2 + a_2^2 - 2a_2 + 2a_1(a_2 + 1)w + 4a_2w^2)^2}. \end{aligned}$$

Приравнивая производную к нулю, получаем уравнение

$$2w(1 + a_1^2 + a_2^2 - 2a_2 + 2a_1(a_2 + 1)w + 4a_2w^2) + (1 - w^2)(2a_1(a_2 + 1) + 8a_2w) = 0.$$

Раскрытие скобок и приведение подобных слагаемых дает

$$a_1(a_2 + 1)w^2 + (a_1^2 + (a_2 + 1)^2)w + a_1(a_2 + 1) = 0.$$

Решение этого квадратного уравнения приводит к следующему результату:

$$w_1 = -\frac{a_1}{a_2 + 1}, \quad w_2 = -\frac{a_2 + 1}{a_1}.$$

Поскольку  $w = \cos(\omega T)$ , модуль  $w$  не превышает единицы. В сочетании с условием устойчивости системы второго порядка (см. (4.20) и рис. 4.7) это позволяет отбросить один из корней и окончательно записать

$$w = \cos(\omega T) = -\frac{a_1}{a_2 + 1}. \quad (4.23)$$

Итак, чтобы получился резонатор с центральной частотой  $\omega_0$ , соотношение между коэффициентами  $a_1$  и  $a_2$  должно быть следующим:

$$a_1 = -(a_2 + 1)\cos(\omega_0 T). \quad (4.24)$$

### ЗАМЕЧАНИЕ

Найденный экстремум является именно *максимумом*, поскольку мы рассматриваем квадрат АЧХ — неотрицательную непрерывную функцию, и нам известно, что она равна нулю при  $w = \pm 1$ .

С учетом (4.24) функция передачи резонатора принимает вид

$$H(z) = b_0 \frac{1 - z^{-2}}{1 - (a_2 + 1)\cos(\omega_0 T)z^{-1} + a_2 z^{-2}}.$$

Чтобы найти квадрат АЧХ резонатора на частоте  $\omega_0$ , подставим в (4.22) найденное значение переменной  $w$  (4.23):

$$\left| \dot{K}(\omega_0) \right|^2 = 4b_0^2 \frac{1 - \frac{a_1^2}{(a_2 + 1)^2}}{1 + a_1^2 + a_2^2 - 2a_2 + 2a_1(a_2 + 1) \frac{-a_1}{a_2 + 1} + 4a_2 \frac{a_1^2}{(a_2 + 1)^2}} = \left( \frac{2b_0}{a_2 - 1} \right)^2.$$

Таким образом, чтобы обеспечить на частоте  $\omega_0$  единичный коэффициент передачи, необходимо выполнение соотношения

$$b_0 = \frac{1 - a_2}{2}.$$

Итак, функция передачи резонатора с центральной частотой  $\omega_0$  имеет вид

$$H(z) = \frac{1 - a_2}{2} \frac{1 - z^{-2}}{1 - (a_2 + 1)\cos(\omega_0 T)z^{-1} + a_2 z^{-2}}. \quad (4.25)$$

Оставшийся параметр  $a_2$  регулирует полосу пропускания резонатора. Чтобы система была устойчивой, необходимо выполнение условия  $|a_2| < 1$ .

Качественно рассмотрим влияние величины  $a_2$  на ширину полосы пропускания. При  $a_2 = -1$  числитель и знаменатель функции передачи (4.25) становятся совпадающими, а множитель перед дробью принимает значение, равное 1. Таким образом, функция передачи в данном случае превращается в константу:  $H(z) = 1$ . Частотно-избирательные свойства полностью утрачиваются. Если же значение  $a_2$  только приближается к минус единице, мы получаем резонатор с широкой полосой пропускания.

При  $a_2 = 1$  у функции передачи появляется полюс на единичной окружности, из-за которого АЧХ на частоте  $\omega_0$  стала бы равной бесконечности, если бы не множитель перед дробью, который оказывается в данном случае равным нулю. В результате на всех частотах, кроме  $\omega_0$ , АЧХ оказывается равной нулю, а на частоте  $\omega_0$  имеем раскрываемую неопределенность, равную единице. Если же значение  $a_2$  только приближается к единице, мы получаем резонатор с узкой полосой пропускания.

При  $a_2 = 0$  функция передачи становится равной

$$H(z) = -\frac{1}{2} \frac{1 - z^{-2}}{1 - \cos(\omega_0 T)z^{-1}}.$$

Квадрат АЧХ в этом случае после несложных преобразований можно представить в следующем виде:

$$\left| \dot{K}(\omega) \right|^2 = \frac{1 - \cos^2(\omega T)}{1 + \cos^2(\omega_0 T) - 2 \cos(\omega_0 T) \cos(\omega T)} = \frac{1}{1 + \frac{(\cos(\omega T) - \cos(\omega_0 T))^2}{1 - \cos^2(\omega T)}}.$$

Определим полосу пропускания по уровню  $\left| \dot{K}(\omega) \right|^2 = \frac{1}{2}$ . Это приводит к следующему уравнению:

$$(\cos(\omega T) - \cos(\omega_0 T))^2 = 1 - \cos^2(\omega T).$$

После раскрытия скобок получается квадратное уравнение относительно  $\cos(\omega T)$ , решение которого дает

$$\cos(\omega_1 T) = \frac{\cos(\omega_0 T) + \sqrt{2 - \cos^2(\omega_0 T)}}{2}, \quad \cos(\omega_2 T) = \frac{\cos(\omega_0 T) - \sqrt{2 - \cos^2(\omega_0 T)}}{2}.$$

Можно легко убедиться в том, что найденные корни удовлетворяют соотношению

$$\cos^2(\omega_1 T) + \cos^2(\omega_2 T) = 1.$$

Отсюда получается, что  $\cos^2(\omega_1 T) = \sin^2(\omega_2 T)$  и, следовательно,  $\omega_2 T = \omega_1 T + \pi/2$  (остальные варианты можно отбросить, поскольку мы рассматриваем рабочий диапазон частот дискретной системы  $\omega = 0 \dots \pi/T$  и, кроме того, пара найденных значений  $\cos(\omega T)$  всегда имеет разные знаки). Итак, полоса пропускания резонатора при  $a_2 = 0$  оказывается равной

$$\Delta\omega = \omega_2 - \omega_1 = \frac{\pi}{2T} = \frac{\omega_d}{4}$$

независимо от центральной частоты  $\omega_0$ .

Построим графики АЧХ резонатора с центральной частотой, равной 0,1 от частоты дискретизации (0,2 от частоты Найквиста) при трех значениях  $a_2$ , равных  $-0,9; 0$  и  $0,9$ . Для удобства формирования однотипных векторов коэффициентов создадим *анонимные функции*, а для визуализации характеристик резонаторов воспользуемся функцией `fvtool`:

```
>> b = @(a2) (a2-1)/2*[1 0 -1]';
>> a = @(a2,w0) [1 -(a2+1)*cos(w0) a2]';
>> w0 = 0.2 * pi; % центральная частота
>> fvtool(b(-0.9), a(-0.9, w0), b(0), a(0, w0), b(0.9), a(0.9, w0));
```

Результат работы этого кода представлен на рис. 4.8. Из графиков видно, что с ростом значения  $a_2$  от  $-1$  до  $1$  полоса пропускания резонатора постепенно сужается.

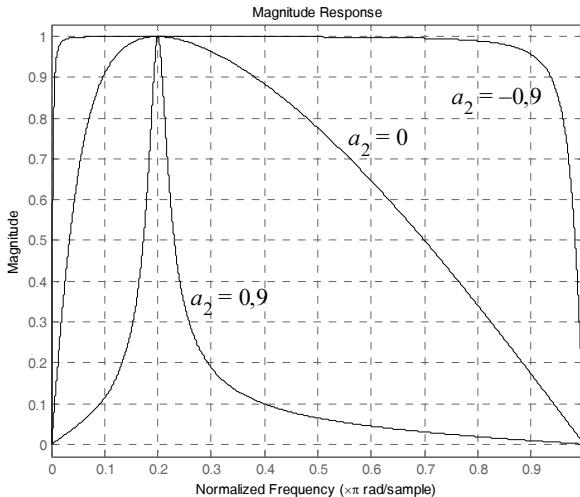


Рис. 4.8. АЧХ резонатора второго порядка при различных значениях  $a_2$

## Режектор второго порядка

Дискретным режектором второго порядка называется система, дуальная по отношению к рассмотренному ранее резонатору, т. е. представляющая собой простейший режекторный фильтр. Иными словами, АЧХ режектора обладает следующими свойствами:

- на нулевой частоте АЧХ равна единице:  $\dot{K}(0) = 1$ ;
- на частоте Найквиста АЧХ равна единице:  $\dot{K}(\omega_d / 2) = 1$ ;
- на некоторой частоте  $\omega_0$  АЧХ имеет минимум, равный нулю:  $\dot{K}(\omega_0) = 0$ .

Кроме того, очевидно, что существует множество систем, АЧХ которых удовлетворяет перечисленным требованиям, но имеет различную "остроту" минимума, т. е. разную полосу режекции. Таким образом, ширина полосы задерживания, измеренная по заданному уровню АЧХ, наряду со средней частотой  $\omega_0$  является исходным параметром при расчете режектора.

Выведем формулы, связывающие вещественные коэффициенты фильтра  $b_0$ ,  $b_1$ ,  $b_2$ ,  $a_1$ ,  $a_2$ , входящие в выражение для функции передачи (4.16), с центральной частотой и шириной полосы задерживания режектора.

Третье из перечисленных выше требований выполнить очень просто — оно означает, что функция передачи режектора должна иметь пару комплексно-сопряженных нулей в точках единичной окружности, соответствующих частоте режекции, т. е. при  $z = \exp(\pm j\omega_0 T)$ . Отсюда сразу же получаем, что числитель функции передачи должен иметь вид

$$b_0(1 - e^{-j\omega_0 T} z^{-1})(1 - e^{j\omega_0 T} z^{-1}) = b_0 \left(1 - 2 \cos(\omega_0 T) z^{-1} + z^{-2}\right),$$

а вся функция передачи может быть записана как

$$H(z) = b_0 \frac{1 - 2 \cos(\omega_0 T) z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}. \quad (4.26)$$

Теперь займемся выполнением первых двух требований к АЧХ. Для комплексного коэффициента передачи на нулевой частоте и частоте Найквиста с учетом (4.26) получаются следующие уравнения:

$$\begin{aligned} \dot{K}(0) = H(1) &= \frac{b_0 (1 - 2 \cos(\omega_0 T) + 1)}{1 + a_1 + a_2} = 1, \\ \dot{K}\left(\frac{\omega_d}{2}\right) = H(-1) &= \frac{b_0 (1 + 2 \cos(\omega_0 T) + 1)}{1 - a_1 + a_2} = 1. \end{aligned}$$

Отсюда после приравнивания друг к другу числителя и знаменателя каждой дроби получаем следующую систему уравнений:

$$\begin{cases} 2b_0 (1 - \cos(\omega_0 T)) = 1 + a_1 + a_2, \\ 2b_0 (1 + \cos(\omega_0 T)) = 1 - a_1 + a_2. \end{cases}$$

Чтобы получить решение этой системы относительно  $b_0$  и  $a_1$ , вычислим полусумму и полуразность уравнений:

$$\begin{cases} 2b_0 = a_2 + 1, \\ 2b_0 \cos(\omega_0 T) = -a_1. \end{cases}$$

Отсюда следует окончательный результат:

$$\begin{cases} b_0 = \frac{a_2 + 1}{2}, \\ a_1 = -(a_2 + 1) \cos(\omega_0 T). \end{cases}$$

Итак, функция передачи режектора с центральной частотой  $\omega_0$  имеет вид

$$H(z) = \frac{a_2 + 1}{2} \frac{1 - 2 \cos(\omega_0 T) z^{-1} + z^{-2}}{1 - (a_2 + 1) \cos(\omega_0 T) z^{-1} + a_2 z^{-2}}. \quad (4.27)$$

Оставшийся параметр  $a_2$  регулирует ширину полосы задерживания режектора. Чтобы система была устойчивой, необходимо выполнение условия  $|a_2| < 1$ .

Рассуждения, аналогичные тем, что были приведены ранее, позволяют показать, что при  $a_2 \rightarrow -1$  получается режектор с *широкой* полосой задерживания, при  $a_2 \rightarrow 1$  полоса задерживания становится узкой, а при  $a_2 = 0$  она, так же как и у резонатора, равна  $\omega_d/4$  (по уровню половинной мощности).

Модифицируем приведенный ранее MATLAB-код, чтобы построить графики АЧХ режектора с той же центральной частотой и при тех же значениях  $a_2$ , что и для рассматривавшегося резонатора:

```
>> b1 = @(a2,w0) (a2+1)/2*[1 -2*cos(w0) 1]';  
>> a1 = @(a2,w0) [1 -(a2+1)*cos(w0) a2]';  
>> w0 = 0.2 * pi; % центральная частота  
>> fvtool(b1(-0.9, w0), a1(-0.9, w0), b1(0, w0), a1(0, w0), b1(0.9, w0),  
a1(0.9, w0));
```

Результат работы этого кода представлен на рис. 4.9. Из графиков видно, что с ростом значения  $a_2$  от  $-1$  до  $1$  полоса задерживания режектора постепенно сужается.

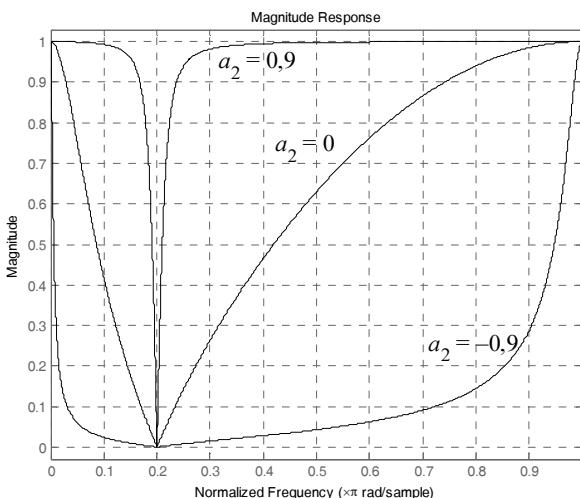


Рис. 4.9. АЧХ режектора второго порядка при различных значениях  $a_2$

## Преобразование случайного сигнала в дискретной системе

Пусть на вход дискретной системы с импульсной характеристикой  $h(k)$  поступает дискретный стационарный центрированный случайный процесс  $\{x(k)\}$  с корреляционной функцией  $R_x(\Delta k)$ . Найдем корреляционную функцию выходного случайного процесса. Для этого воспользуемся определением корреляционной функции (см. разд. "Дискретные случайные сигналы" главы 3) и формулой (4.3), описывающей преобразование сигнала, осуществляющее системой:

$$R_y(\Delta k) = \overline{y(k)y(k + \Delta k)} = \sum_{n_1=-\infty}^{\infty} x(n_1)h(k - n_1) \sum_{n_2=-\infty}^{\infty} x(n_2)h(k + \Delta k - n_2).$$

Преобразуем произведение сумм в двойную сумму, воспользуемся линейностью операции статистического усреднения и заметим, что  $\overline{x(n_1)x(n_2)} = R_x(n_2 - n_1)$ :

$$\begin{aligned} R_y(\Delta k) &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} \overline{x(n_1)x(n_2)} h(k - n_1) h(k + \Delta k - n_2) = \\ &= \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} R_x(n_1 - n_2) h(k - n_1) h(k + \Delta k - n_2). \end{aligned} \quad (4.28)$$

Произведем замену второго индекса суммирования:  $m = n_2 - n_1$ . С учетом этого формула (4.28) принимает вид

$$\begin{aligned} R_y(\Delta k) &= \sum_{n_1=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} R_x(m) h(k - n_1) h(k + \Delta k - m - n_1) = \\ &= \sum_{m=-\infty}^{\infty} \left[ R_x(m) \sum_{n_1=-\infty}^{\infty} h(k - n_1) h(k + \Delta k - m - n_1) \right]. \end{aligned} \quad (4.29)$$

Внутренняя сумма в (4.29) представляет собой корреляционную функцию импульсной характеристики системы:

$$\sum_{n_1=-\infty}^{\infty} h(k - n_1) h(k + \Delta k - m - n_1) = \sum_{n=-\infty}^{\infty} h(n) h(n + \Delta k - m) = B_h(\Delta k - m).$$

Таким образом, окончательно получаем, что корреляционная функция случайного сигнала на выходе системы представляет собой свертку корреляционной функции входного шума и корреляционной функции импульсной характеристики системы:

$$R_y(\Delta k) = \sum_{m=-\infty}^{\infty} R_x(m) B_h(\Delta k - m). \quad (4.30)$$

### ЗАМЕЧАНИЕ

Полученная формула является дискретным аналогом формулы (2.6), описывающей преобразование корреляционной функции аналогового шума в линейной цепи.

Дисперсия выходного случайного процесса может быть рассчитана как

$$\sigma_y^2 = R_y(0) = \sum_{m=-\infty}^{\infty} R_x(m) B_h(m).$$

В этой формуле учтена четность функции  $B_h(m)$ .

Если входной случайный процесс является дискретным белым шумом, формула (4.30) упрощается:

$$R_y(\Delta k) = \sigma_x^2 B_h(\Delta k).$$

Дисперсия выходного шума при этом составляет

$$\sigma_y^2 = \sigma_x^2 B_h(0) = \sigma_x^2 \sum_{k=0}^{\infty} h^2(k).$$

Таким образом, при воздействии на вход системы дискретного белого шума дисперсия выходного сигнала пропорциональна сумме квадратов отсчетов импульсной характеристики системы.

## Рекурсивные и нерекурсивные дискретные фильтры

В начале главы, рассматривая идею обработки сигнала дискретной линейной системой, мы уже ввели понятия нерекурсивных и рекурсивных фильтров. Настало время вернуться к этому вопросу и обсудить его более обстоятельно. Для этого рассмотрим структурные схемы устройств, реализующих уравнение (4.1), в общем виде описывающее процесс обработки сигнала дискретной системой.

### Нерекурсивные фильтры

Прежде всего следует отметить, что в общем случае при вычислении очередного выходного отсчета  $y(k)$  используется информация двух типов: некоторое количество отсчетов *входного* сигнала и некоторое количество предыдущих отсчетов *выходного* сигнала. Ясно, что хотя бы один отсчет входного сигнала должен участвовать в вычислениях; в противном случае выходной сигнал не будет зависеть от входного. В противоположность этому, предыдущие отсчеты выходного сигнала могут и не использоваться. Уравнение фильтрации (4.1) в этом случае приобретает следующий вид:

$$y(k) = \sum_{i=0}^m b_i x(k-i). \quad (4.31)$$

Количество используемых предыдущих отсчетов  $m$  называется *порядком фильтра* (*filter order*).

#### ЗАМЕЧАНИЕ

Следует обратить внимание на различие между терминами *порядок фильтра* и *длина фильтра*. Под длиной фильтра имеется в виду общее число отсчетов, участвующих в вычислениях, включая текущий отсчет  $x(k)$ , поэтому длина дискретного фильтра на единицу больше, чем его порядок.

Структурная схема, реализующая алгоритм (4.31), приведена на рис. 4.10. Некоторое количество предыдущих отсчетов входного сигнала хранится в ячейках памяти, которые образуют дискретную линию задержки. Эти отсчеты умножаются на коэффициенты  $b_i$  и суммируются, формируя выходной отсчет  $y(k)$ .

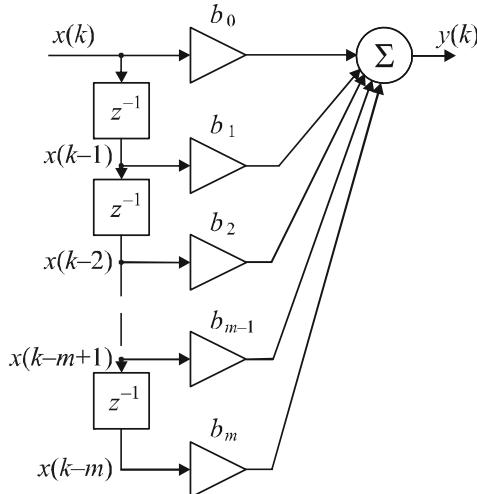


Рис. 4.10. Нерекурсивный фильтр

### ЗАМЕЧАНИЕ

Согласно свойствам  $z$ -преобразования (см. разд. "Z-преобразование" главы 3), задержка дискретной последовательности на один такт соответствует умножению ее  $z$ -преобразования на  $z^{-1}$ . Поэтому элементы памяти, осуществляющие такую задержку, обозначены на структурной схеме как " $z^{-1}$ ".

Так как при вычислениях не используются предыдущие отсчеты выходного сигнала, в схеме отсутствуют обратные связи. Поэтому такие фильтры называются *нерекурсивными* (*nonrecursive*). Применяется также термин "*трансверсальный фильтр*" (от английского *transversal* — поперечный).

Импульсная характеристика нерекурсивного фильтра определяется очень просто. Подставим в уравнение (4.31) единичный импульс  $x_0(k)$  в качестве входного сигнала:

$$h(k) = \sum_{i=0}^m b_i x_0(k-i).$$

Но отсчет  $x_0(k-i)$  равен нулю для всех  $k$ , кроме  $k = i$ , когда этот отсчет равен единице. Поэтому мы получаем очень простой результат:

$$h(k) = b_k,$$

то есть коэффициенты  $b_i$  являются отсчетами импульсной характеристики фильтра. Это можно наглядно пояснить с помощью рис. 4.10. При подаче на вход единичного импульса он будет перемещаться по линии задержки, умножаться на коэффициенты  $b_0, b_1, b_2, \dots$  и проходить на выход устройства (ведь все остальные входные сигналы сумматора при этом равны нулю). Очевидно, что в реальном устройстве линия задержки содержит конечное число элементов, поэтому импульсная характеристика нерекурсивного фильтра также является *конечной* по длительности. Это обусловило еще одно название таких фильтров — фильтры с *конечной им-*

*пульсной характеристикой* (КИХ-фильтры; английский термин — *finite impulse response, FIR*).

### ЗАМЕЧАНИЕ

Вследствие отсутствия обратных связей любой нерекурсивный фильтр является устойчивым — ведь каковы бы ни были начальные условия (т. е. отсчеты, хранящиеся в линии задержки), при отсутствии сигнала на входе ( $x(k) = 0$ ) выходной сигнал (свободные колебания) будет отличен от нуля в течение не более чем  $m$  тактов, необходимых для очистки линии задержки.

Простота анализа и реализации, а также наглядная связь коэффициентов фильтра с отсчетами его импульсной характеристики и абсолютная устойчивость привели к тому, что нерекурсивные фильтры широко применяются на практике. Однако для получения хороших частотных характеристик (например, полосовых фильтров с высокой прямоугольностью АЧХ) необходимы нерекурсивные фильтры высокого порядка — до нескольких сотен и даже тысяч.

## Симметричные фильтры

Очень важное значение имеет тот факт, что нерекурсивные фильтры позволяют легко обеспечить линейную ФЧХ, а значит, постоянные (не зависящие от частоты) групповую и фазовую задержки. Для этого необходима лишь симметрия импульсной характеристики. Эта симметрия может быть двух типов:

- *четная симметрия (even symmetry)*:  $b_k = b_{N-k}$  для всех  $k = 0, 1, \dots, N$ ;
- *нечетная симметрия (odd symmetry)*:  $b_k = -b_{N-k}$  для всех  $k = 0, 1, \dots, N$ .

### ЗАМЕЧАНИЕ

Иногда под *симметричными* подразумевают только характеристики с четной симметрией, а для нечетной симметрии используют термин "антисимметричные".

Доказать линейность ФЧХ для симметричных фильтров очень легко, если воспользоваться для этого свойствами преобразования Фурье. Действительно, их импульсные характеристики представляют собой четные или нечетные (в зависимости от типа симметрии) функции, сдвинутые вправо на  $N/2$  отсчетов. Спектр четной функции является чисто вещественным, а нечетной — чисто мнимым (см. разд. "Преобразование Фурье" главы 1), временной сдвиг же приводит к появлению в ФЧХ слагаемого, линейно зависящего от частоты. Таким образом, ФЧХ симметричных фильтров описывается следующим образом:

- в случае четной симметрии  $\phi_K(\omega) = -\omega T \frac{N}{2}$ ;
- в случае нечетной симметрии  $\phi_K(\omega) = -\frac{\pi}{2} \text{sign}(\omega) - \omega T \frac{N}{2}$ .

### ЗАМЕЧАНИЕ

Приведенные формулы являются не вполне строгими — в них полагается, что преобразование Фурье от четной (нечетной) функции дает чисто вещественный (мнимый)

спектр с *неотрицательной* вещественной (мнимой) частью. В общем же случае ФЧХ может содержать не учтенные в приведенных формулах скачки на  $\pm 180^\circ$ .

Групповая задержка для симметричных фильтров не зависит от частоты и равна  $N/2$  отсчетам.

При четном  $N$  и нечетной симметрии импульсной характеристики, очевидно, ее средний отсчет должен быть равен нулю:  $b_{N/2} = 0$ . Кроме того, четность или нечетность порядка фильтра и наличие того или иного типа симметрии накладывают определенные ограничения на коэффициенты передачи фильтра на нулевой частоте и на частоте Найквиста. Эти ограничения легко получить из условий симметрии и формулы (4.7) для комплексного коэффициента передачи фильтра. Сочетание четности порядка фильтра и типа симметрии дает четыре типа симметричных фильтров, перечисленных в табл. 4.1 вместе с указанными ограничениями значений АЧХ. Приведенные в таблице номера типов часто используются в зарубежной литературе.

**Таблица 4.1.** Типы симметричных фильтров

Тип	Порядок фильтра	Тип симметрии	$K(0)$	$K(\omega_a/2)$
I	Четный	Четная	Любой	Любой
II	Нечетный	Четная	Любой	0
III	Четный	Нечетная	0	0
IV	Нечетный	Нечетная	0	Любой

## Рекурсивные фильтры

Если уравнение фильтрации имеет общий вид (4.1), т. е. содержит как входные, так и выходные отсчеты, для реализации такого фильтра в схему, приведенную на рис. 4.10, необходимо добавить вторую линию задержки — для хранения выходных отсчетов  $y(k-i)$ . Получающаяся при этом структура показана на рис. 4.11.

Так как при вычислениях используются предыдущие отсчеты выходного сигнала, в схеме присутствуют обратные связи. Поэтому такие фильтры называют *рекурсивными* (*recursive*).

### ЗАМЕЧАНИЕ

Количество предыдущих входных и выходных отсчетов, используемых для вычислений, может не совпадать. В таком случае порядком фильтра считается *максимальное* из чисел  $m$  и  $n$ .

Импульсная характеристика рекурсивного фильтра рассчитывается значительно сложнее, чем для нерекурсивного. Рассмотрим формирование лишь нескольких первых ее отсчетов. При поступлении на вход единичного импульса он умножается на  $b_0$  и проходит на выход. Таким образом,

$$h(0) = b_0 .$$

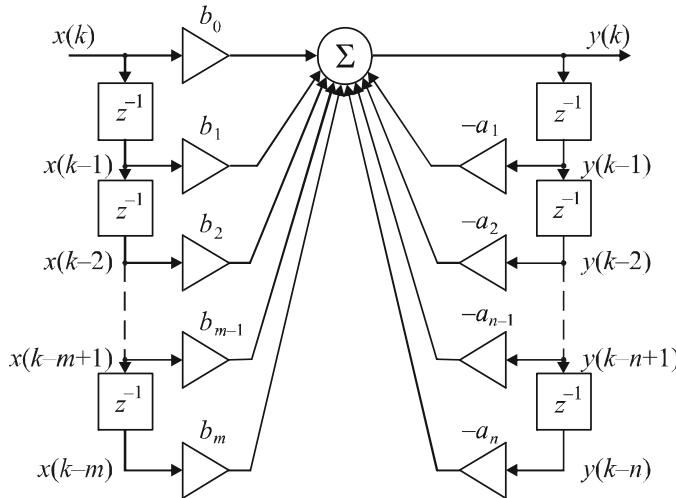


Рис. 4.11. Рекурсивный фильтр — прямая реализация

Далее входной единичный импульс попадает во входную линию задержки, а выходной отсчет, равный  $b_0$ , — в выходную линию задержки. В результате второй отсчет импульсной характеристики будет формироваться как

$$h(1) = b_1 - a_1 h(0) = b_1 - b_0 a_1.$$

Продолжив рассмотрение перемещения входного единичного импульса вдоль входной линии задержки и заполнения выходными отсчетами выходной линии задержки, можно получить

$$h(2) = b_2 - a_2 h(0) - a_1 h(1) = b_2 - b_0 a_2 - a_1 (b_1 - b_0 a_1) = b_2 - b_1 a_1 - b_0 a_2 + b_0 a_1^2.$$

Видно, что по мере того, как выходная линия задержки заполняется отсчетами импульсной характеристики, сложность аналитических формул быстро возрастает.

Наличие в схеме обратных связей позволяет получить бесконечную импульсную характеристику, поэтому рекурсивные фильтры называют также фильтрами с бесконечной импульсной характеристикой (БИХ-фильтры; английский термин — *infinite impulse response, IIR*). По этой же причине рекурсивные фильтры могут быть неустойчивыми.

#### ЗАМЕЧАНИЕ

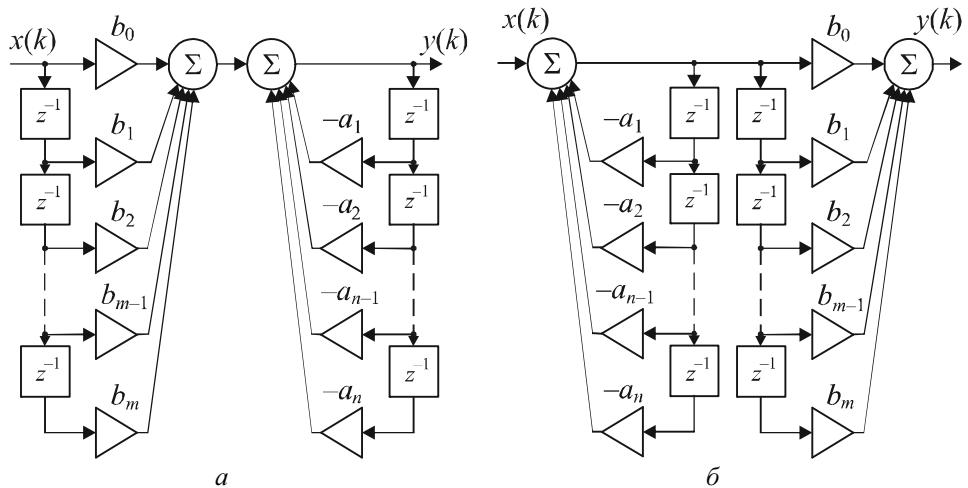
С помощью рекурсивной схемы можно реализовать и конечную импульсную характеристику. Пример такого рода будет приведен в главе 10 при описании структур "Интегратор — гребенчатый фильтр".

## Формы реализации дискретных фильтров

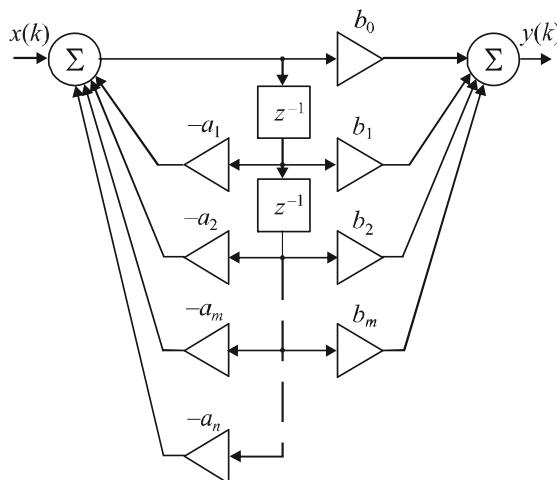
Структурная схема, показанная ранее на рис. 4.11, называется *прямой формой реализации* рекурсивного фильтра (*direct form I*) и не является единственной возможной. Рассмотрим еще несколько вариантов.

## Каноническая форма

Разделим общий сумматор в схеме рис. 4.11 на два отдельных — для рекурсивной и нерекурсивной частей фильтра (рис. 4.12, *a*). В результате получаем два последовательно соединенных фильтра, один из которых является нерекурсивным, а другой, напротив, содержит только рекурсивную часть. Так как результат последовательного прохождения сигнала через ряд линейных стационарных устройств не зависит от последовательности их соединения, мы можем поменять местами две "половинки" нашего фильтра (рис. 4.12, *б*). Теперь остается заметить, что в обе линии задержки подается один и тот же сигнал, поэтому они будут содержать одинаковые наборы отсчетов. Это позволяет объединить линии задержки. Полученная в результате схема изображена на рис. 4.13, она называется *канонической формой реализации рекурсивного фильтра* (*canonic form* или *direct form II*).



**Рис. 4.12.** Перестановка рекурсивной и нерекурсивной частей фильтра — путь к получению канонической реализации



**Рис. 4.13.** Рекурсивный фильтр — каноническая реализация

С теоретической точки зрения эти варианты эквивалентны. Однако при практической реализации необходимо обратить внимание на ряд особенностей, присущих этим схемам. С одной стороны, при канонической реализации используется общая линия задержки, что уменьшает число необходимых ячеек памяти. Однако при этом абсолютные величины отсчетов, "бегающих" в линии задержки, могут существенно превосходить амплитуду входного и выходного сигналов. Это приводит к необходимости увеличивать разрядность представления чисел в линии задержки по сравнению с разрядностью входного и выходного сигналов, что усложняет реализацию устройства. При прямой реализации в линиях задержки хранятся непосредственно отсчеты входного и выходного сигналов, т. е. повышенная разрядность линий задержки не требуется. Единственным элементом, требующим повышенной разрядности, в данном случае является сумматор, и это учтено в архитектуре микропроцессоров, специально предназначенных для обработки сигналов в реальном времени.

#### **ЗАМЕЧАНИЕ**

Каноническая форма реализации соответствует одному из возможных вариантов представления дискретной системы в *пространстве состояний*. Вектор состояния при этом представляет собой набор значений выходов элементов задержки.

## **Транспонированная форма**

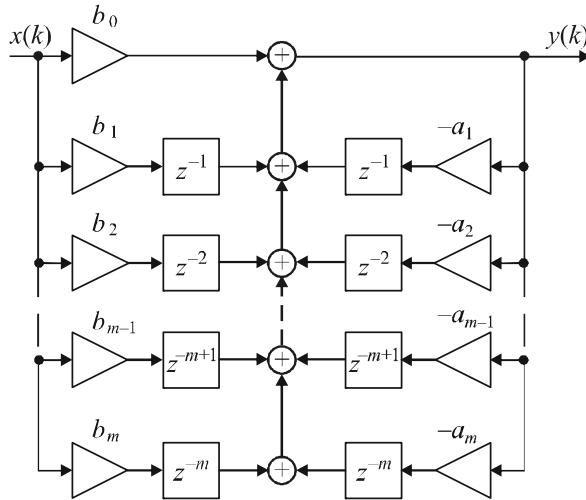
Поменяем в схеме рис. 4.10 последовательность выполнения операций умножения и задержки, используя в каждой ветви отдельную линию задержки на нужное количество тактов. Разделим также общий сумматор на несколько двухходовых сумматоров. Получившаяся структура показана на рис. 4.14. Теперь, рассмотрев любую пару соседних сумматоров, можно заметить, что суммируемые ими сигналы претерпевают некоторую общую задержку. Это дает возможность поменять местами операции суммирования и задержки. Получившаяся схема, показанная на рис. 4.15, называется *транспонированной* реализацией дискретного фильтра (*direct transposed form II*).

#### **ЗАМЕЧАНИЕ**

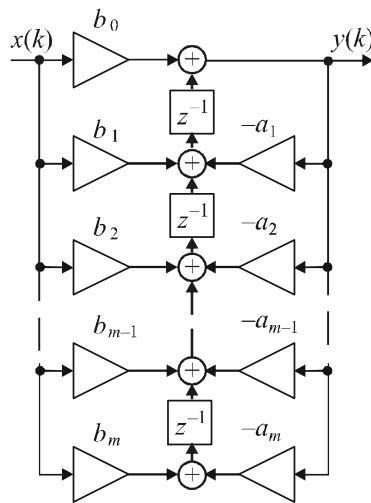
Разумеется, в транспонированной форме может быть реализован и нерекурсивный фильтр. Для этого в структурной схеме рис. 4.15 необходимо удалить все ветви с коэффициентами  $a_i$ .

Транспонированная схема позволяет эффективно распараллелить вычисления и потому применяется при реализации дискретных фильтров в виде специализированных интегральных схем. Действительно, при реализации фильтра в форме рис. 4.10 или рис 4.11 можно одновременно выполнять все операции умножения, но для получения выходного результата необходимо дождаться окончания выполнения *всех* операций сложения. В транспонированной же схеме, помимо умножения, можно одновременно выполнять и все операции *сложения*, поскольку они являются независимыми (т. е. не используют в качестве суммируемых величин результаты дру-

гих сложений). Как видно из схемы рис. 4.15, собственно для расчета выходного сигнала необходимо выполнить *одно умножение и одно сложение*; все остальные операции производят подготовку промежуточных результатов для вычисления последующих выходных отсчетов.



**Рис. 4.14.** Изменение последовательности выполнения операций умножения и задержки — путь к получению транспонированной реализации фильтра

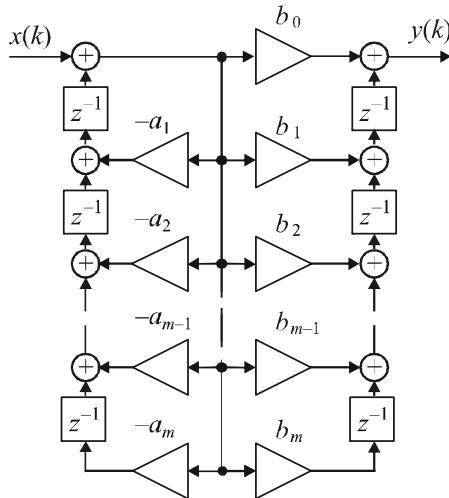


**Рис. 4.15.** Транспонированная реализация дискретного фильтра

### ЗАМЕЧАНИЕ

Транспонированная форма реализации, показанная на рис. 4.15, соответствует еще одному варианту представления дискретной системы в *пространстве состояний*. *Вектор состояния* при этом представляет собой набор значений выходов элементов задержки.

Если применить описанные преобразования к канонической структуре, показанной на рис. 4.13, получится еще один вариант транспонированной реализации фильтра (*direct transposed form I*) (рис. 4.16). В отличие от предыдущей схемы, данная структура содержит большее число элементов памяти.



**Рис. 4.16.** Транспонированная реализация, полученная из канонической формы дискретного фильтра

## Последовательная (каскадная) форма

В разд. "Нули и полюсы" этой главы было показано, что числитель и знаменатель функции передачи физически реализуемого дискретного фильтра можно разложить на линейные (относительно  $z^{-1}$ ) множители. Перемножение функций передачи соответствует последовательному (каскадному) включению соответствующих фильтров, поэтому такое представление дает реализацию фильтра в виде последовательно включенных фильтров 1-го порядка (при этом некоторые из них могут иметь комплексные коэффициенты) либо фильтров 1-го и 2-го порядка с вещественными коэффициентами.

Рассмотрим конкретный пример, задавшись численными значениями коэффициентов фильтра:

$$\begin{aligned} H(z) &= \frac{0,0985 + 0,2956z^{-1} + 0,2956z^{-2} + 0,0985z^{-3}}{1 - 0,5772z^{-1} + 0,4218z^{-2} - 0,0563z^{-3}} = \\ &= \frac{0,0985(1+z^{-1})(1+2z^{-1}+z^{-2})}{(1-0,1584z^{-1})(1-0,4188z^{-1}+0,3554z^{-2})}. \end{aligned} \quad (4.32)$$

### ЗАМЕЧАНИЕ

Данный фильтр является фильтром Баттервортса 3-го порядка с частотой среза, равной  $1/5$  частоты дискретизации, синтезированным методом билинейного  $z$ -преобразования (о данном методе синтеза дискретных фильтров см. в главе 6).

Структурная схема получившейся последовательной реализации фильтра представлена на рис. 4.17.

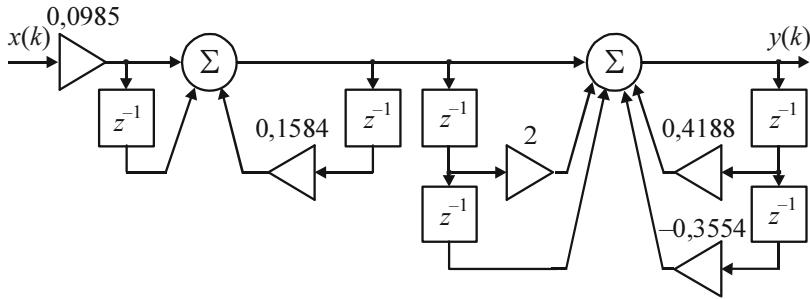


Рис. 4.17. Последовательная реализация дискретного фильтра

Последовательная реализация часто используется на практике, поскольку она позволяет ослабить нежелательные эффекты, связанные с ошибками округления [8]. Такие эффекты обсуждаются в главе 7.

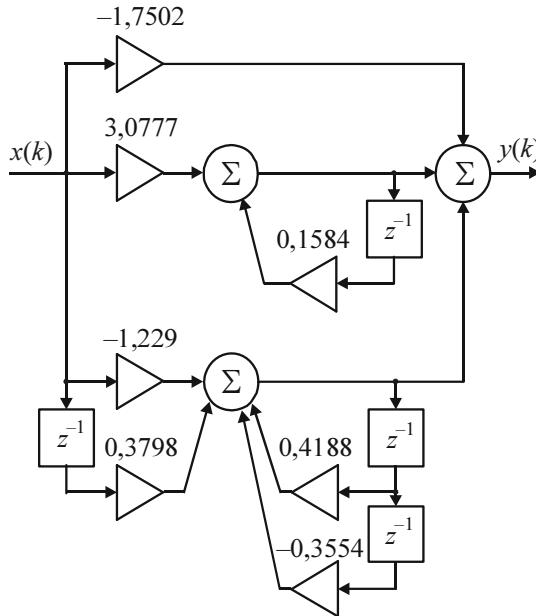
## Параллельная форма

Еще один способ преобразования функции передачи физически реализуемого фильтра — представление ее в виде суммы простейших дробей. Об этом уже шла речь в разд. "Полюсы и вычеты" этой главы (см. формулу (4.12)). Каждое из слагаемых при таком представлении соответствует функции передачи рекурсивного фильтра 1-го порядка (возможно, с комплексными коэффициентами) либо 1-го или 2-го порядка (если используется представление в виде суммы простейших дробей только с вещественными коэффициентами). Сама операция сложения эквивалентна параллельному соединению этих фильтров с суммированием выходных результатов.

Рассмотрим конкретный пример, используя ту же функцию передачи (4.32), что и раньше:

$$\begin{aligned} H(z) &= \frac{0,0985 + 0,2956z^{-1} + 0,2956z^{-2} + 0,0985z^{-3}}{1 - 0,5772z^{-1} + 0,4218z^{-2} - 0,0563z^{-3}} = \\ &= -1,7502 + \frac{3,0777}{1 - 0,1584z^{-1}} + \frac{-1,229 + 0,3798z^{-1}}{1 - 0,4188z^{-1} + 0,3554z^{-2}}. \end{aligned}$$

Структурная схема получившейся параллельной реализации фильтра представлена на рис. 4.18. Постоянному слагаемому соответствует верхняя ветвь структурной схемы, содержащая только умножитель.



**Рис. 4.18.** Параллельная реализация дискретного фильтра

## Дискретная фильтрация в MATLAB

В данном разделе рассматриваются средства MATLAB, предназначенные для выполнения дискретной фильтрации и анализа параметров дискретных фильтров. Следует отметить, что сфера применения технологий дискретной фильтрации в настоящее время не ограничивается обработкой сигналов — это технологии обработки любых данных. Данный факт нашел свое отражение и в том, что в MATLAB базовые функции дискретной фильтрации не относятся к пакету расширения Signal Processing, а являются встроенными в ядро программы.

Помимо многочисленных функций расчета параметров дискретных фильтров в составе пакета Signal Processing имеется программа *FDATool* (Filter Design & Analysis Tool) — графическая среда для анализа дискретных фильтров. Эта программа позволяет не только анализировать, но и рассчитывать (синтезировать) фильтры с заданными свойствами, поэтому она будет описана в главе 6, посвященной проектированию дискретных фильтров.

## Дискретная свертка

Дискретная свертка, являющаяся основой алгоритма дискретной фильтрации, в MATLAB вычисляется с помощью функции `conv`:

```
z = conv(x, y);
```

Длина выходного вектора равна `length(x)+length(y)-1`.

Вычислим с помощью функции `conv` свертку тех же числовых последовательностей, что рассматривались в качестве примера при пояснении идеи дискретной свертки в разд. "Импульсная характеристика" этой главы:

```
>> x = [1 3 2];
>> h = [1 2 3 4];
>> y = conv(x, h)
y =
    1      5     11     17     18      8
```

Результат, разумеется, соответствует тому, что было получено ранее (см. рис. 4.1).

## Обращение свертки

Зная результат свертки и один из сворачиваемых векторов, можно найти второй. Если перейти к  $z$ -преобразованиям векторов, данная операция (она называется *обращением свертки* — *deconvolution*) сводится к делению полиномов. В MATLAB эта операция реализуется с помощью функции `deconv`, имеющей следующий синтаксис:

```
[q, r] = deconv(b, a)
```

Здесь `b` — результат свертки (коэффициенты полинома-числителя), `a` — один из сворачиваемых векторов (коэффициенты полинома-знаменателя).

Выходные параметры имеют следующий смысл: `q` — результат деления полиномов (*частное*; *quotient*), т. е. искомый второй вектор свертки, `r` — *остаток* (*remainder*) от деления полиномов (если вектор `b` действительно является сверткой вектора `a` с чем-нибудь, остаток от деления будет нулевым).

В общем случае исходные данные `b` и `a` связаны с результатами расчета следующим образом:

```
b = conv(q, a) + r
```

Продемонстрируем использование функции `deconv`, взяв данные из предыдущего примера:

```
>> [q, r] = deconv(y, h)
q =
    1      3      2
r =
    0      0      0      0      0      0
```

Поскольку вектор `y` был получен в результате свертки, мы получили исходный сигнал (см. вектор `x` в предыдущем примере) и нулевой остаток.

## Функция дискретной фильтрации

Основная функция, реализующая дискретную фильтрацию в MATLAB, носит имя `filter`. В простейшем виде она имеет следующий синтаксис:

```
y = filter(b, a, x);
```

Здесь  $b$  — вектор коэффициентов нерекурсивной части фильтра (числителя функции передачи),  $a$  — вектор коэффициентов рекурсивной части фильтра (знаменателя функции передачи),  $x$  — входной сигнал.

Возвращаемой величиной является вектор отсчетов выходного сигнала фильтра.

Если первый элемент вектора  $a$  не равен 1, значения векторов  $b$  и  $a$  нормируются — делятся на  $a(1)$ .

Между функциями `filter` и `conv` есть два основных различия. Во-первых, функция `conv` требует задания импульсной характеристики фильтра в виде вектора, поэтому данная импульсная характеристика должна иметь конечную длину. Фактически это означает, что с помощью функции `conv` можно реализовать только нерекурсивный фильтр. Функция `filter` позволяет задавать как нерекурсивные, так и рекурсивные фильтры, реализуя в том числе и бесконечные импульсные характеристики. Во-вторых, функция `filter` возвращает результат, длина которого *равна* длине входного сигнала: это, как мы увидим далее, позволяет организовать блоковую обработку сигнала.

Повторим пример, рассмотренный ранее при описании функции `conv`, на сей раз, реализовав расчеты с помощью функции `filter`. Поскольку мы используем непосредственно импульсную характеристику фильтра, она должна быть передана функции в качестве вектора  $b$  коэффициентов нерекурсивной части фильтра. Вектор  $a$  рекурсивной части фильтра в этом случае представляет собой скаляр, равный единице:

```
>> y = filter(h, 1, x)
y =
    1      5     11
```

Как видите, результат фильтрации совпадает с полученными в предыдущем разделе значениями. Однако теперь вектор  $y$  не содержит затухающего "хвоста". Дело в том, что функция `conv` предполагала завершенность входного сигнала, а функция `filter` позволяет при необходимости продолжить его обработку, подавая на вход фильтра новые отсчеты (см. далее разд. *"Доступ к внутреннему состоянию фильтра"* этой главы).

Чтобы получить результат, полностью совпадающий с тем, что дает функция `conv`, можно дополнить вектор  $x$  соответствующим количеством нулей:

```
>> y = filter(h, 1, [x 0 0 0])
y =
    1      5     11     17     18      8
```

Входной сигнал  $x$  может быть матрицей. При этом ее столбцы фильтруются независимо.

## Доступ к внутреннему состоянию фильтра

Функция `filter` позволяет задавать и считывать внутреннее состояние дискретного фильтра. Для этого используются дополнительные входной и выходной параметры:

```
[y, z2] = filter(b, a, x, z1);
```

Здесь  $z_1$  — начальное внутреннее состояние фильтра (входной параметр),  $z_2$  — конечное внутреннее состояние фильтра (выходной параметр).

По умолчанию предполагается нулевое внутреннее состояние фильтра.

Чтобы понимать, что именно представляет собой вектор внутреннего состояния фильтра, нужно знать, как организованы вычисления в функции `filter`. Данная функция реализует рекурсивный дискретный фильтр в транспонированной форме (*direct transposed form II*, см. рис. 4.15).

Возможность доступа к внутреннему состоянию фильтра позволяет при необходимости реализовать обработку сигнала по частям:

```
% 1-й блок данных
[y(1:N), z] = filter(b, a, x(1:N));
% 2-й блок данных
[y(N+1:2*N), z] = filter(b, a, x(N+1:2*N), z);
...
...
```

### **ЗАМЕЧАНИЕ**

Если текущее внутреннее состояние фильтра не известно, но известны предыдущие отсчеты входного и выходного сигналов, состояние фильтра можно получить с помощью функции `filtic` ("ic" обозначает начальное состояние — *initial conditions*).

## **Компенсация фазового сдвига**

При фильтрации сигналов в ряде случаев возникает требование сохранения фазовых соотношений в исходном сигнале. Для этого требуется фильтр с линейной ФЧХ или, что то же самое, с постоянной (не зависящей от частоты) фазовой задержкой.

Как говорилось в разд. "Симметричные фильтры" этой главы, для получения линейной ФЧХ фильтр должен иметь симметричную импульсную характеристику. Поэтому линейную ФЧХ невозможно реализовать в нерекурсивном фильтре — для этого нужна только симметрия набора коэффициентов фильтра:  $b_i = b_{m-i}$  или  $b_i = -b_{m-i}$ .

Из симметрии импульсной характеристики фильтра и соображений его физической реализуемости следует *конечность* импульсной характеристики. Поэтому рекурсивные фильтры, обладающие бесконечной импульсной характеристикой, принципиально не могут иметь линейную ФЧХ.

Однако при обработке сигнала не в реальном масштабе времени можно обойти ограничение физической реализуемости и использовать фильтр, для которого *не выполняется условие причинности*.

### **ЗАМЕЧАНИЕ**

Еще один, менее шокирующий подход к данной ситуации можно сформулировать так. Нам известен *весь* входной сигнал, и мы знаем, что он уже *закончился*, т. е. на

вход фильтра в дальнейшем будут поступать только нулевые значения. Поэтому можно считать, что условие причинности по-прежнему соблюдается, но фильтр вносит очень большую задержку, превышающую длительность всего входного сигнала.

Для реализации фильтрации с нулевым фазовым сдвигом в MATLAB предназначена функция `filtfilt`. Ее название символизирует смысл выполняемых ею действий: сначала входной сигнал фильтруется как обычно, а затем производится *второй проход* фильтрации, при котором сигнал обрабатывается *от конца к началу*. Если заданный в качестве параметров функции фильтр имеет функцию передачи  $H(z)$ , импульсную характеристику  $h(k)$  и АЧХ  $K(\omega)$ , то фильтр, эквивалентный двум проходам обработки сигнала, будет иметь функцию передачи  $H(z)H(z^{-1})$ , АЧХ  $|K(\omega)|^2$  и нулевую ФЧХ. Импульсная характеристика представляет собой корреляционную функцию исходной импульсной характеристики:

$$h'(k) = \sum_{m=-\infty}^{\infty} h(m)h(m-k), \quad -\infty < k < \infty.$$

Синтаксис вызова функции `filtfilt` такой же, как для функции `filter`:

```
y = filtfilt(b, a, x)
```

Приведем пример, демонстрирующий компенсацию фазового сдвига с помощью функции `filtfilt`. Создадим сигнал в виде прямоугольного импульса, дополненного с обеих сторон нулями:

```
>> s = [zeros(50, 1); ones(100, 1); zeros(50, 1)];
```

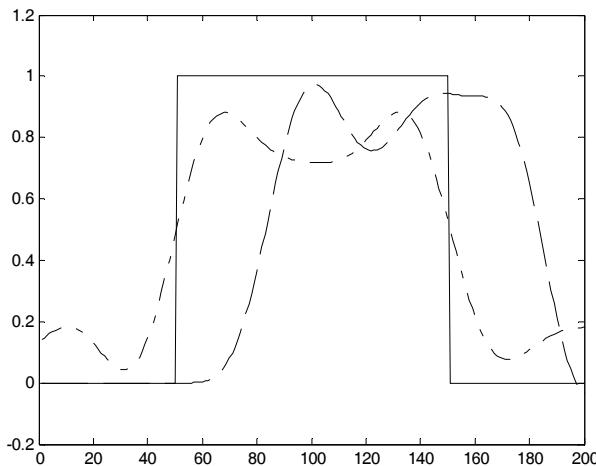
В качестве фильтра используем ФНЧ Чебышева первого рода с достаточно сильными пульсациями АЧХ в полосе пропускания (3 дБ) и низкой частотой среза (0,05 частоты Найквиста), чтобы выходной импульс был заметно искажен:

```
>> [b, a] = cheby1(5, 3, 0.05);
```

Обрабатываем сигнал с помощью функций `filter` и `filtfilt` и строим графики входного и выходных сигналов (рис. 4.19):

```
>> s1 = filter(b, a, s);
>> s2 = filtfilt(b, a, s);
>> plot(s)
>> hold on
>> plot(s1, '--')
>> plot(s2, '-.')
>> hold off
```

На рисунке хорошо видно, что при фильтрации обычным ФНЧ сигнал приобретает существенную задержку по времени. После повторной фильтрации в обратном направлении эта задержка компенсируется. Форма сигнала при втором проходе фильтрации также несколько изменяется, поскольку суммарное действие двух ФНЧ эквивалентно двукратному увеличению порядка фильтра.



**Рис. 4.19.** Входной сигнал (сплошная линия), результат обычной фильтрации (пунктирная линия) и фильтрации с компенсацией фазового сдвига (штрихпунктирная линия)

## Расчет импульсной характеристики

Чтобы получить импульсную характеристику, необходимо подать на вход фильтра единичный отсчет, дополненный некоторым количеством нулей:

```
filter(b, a, [1 zeros(1, N)])
```

Для удобства такой расчет реализован в функции `impz`, которая к тому же обладает рядом дополнительных возможностей.

В простейшем виде синтаксис вызова функции `impz` следующий:

```
h = impz(b, a);
```

Входные параметры `b` и `a` — коэффициенты полиномов числителя и знаменателя функции передачи соответственно.

Возвращаемое значение `h` — вектор отсчетов импульсной характеристики. Число рассчитываемых отсчетов выбирается автоматически (для этого используется функция `impzlength`) и зависит от поведения импульсной характеристики. Более подробную информацию об этом можно найти в документации пакета Signal Processing.

Чтобы явно задать число рассчитываемых отсчетов импульсной характеристики, следует использовать третий входной параметр `n`:

```
h = impz(b, a, n);
```

Теперь обратимся к выходным параметрам функции. В приведенных ранее вариантах использовался один выходной параметр — вектор отсчетов импульсной характеристики. Если выходные параметры отсутствуют, функция `impz` строит график импульсной характеристики с использованием графической функции `stem`.

Построим график импульсной характеристики фильтра Баттервортса 5-го порядка с частотой среза, равной 0,1 частоты дискретизации (рис. 4.20):

```
>> [b, a] = butter(5, 0.2);
>> impz(b, a)
```

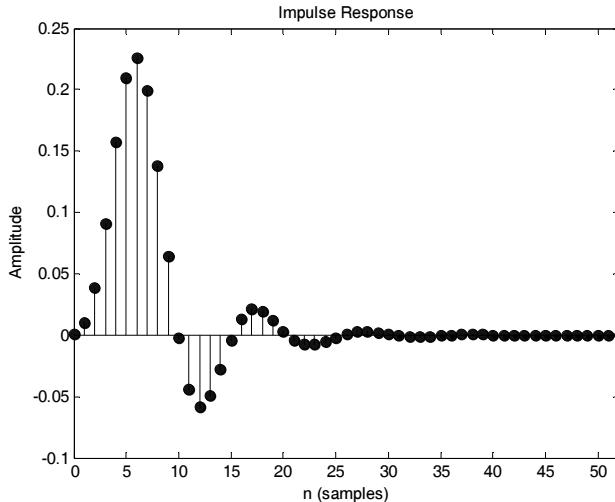


Рис. 4.20. График импульсной характеристики, построенный функцией `impz`

## Расчет переходной характеристики

Чтобы получить переходную характеристику, необходимо подать на вход фильтра последовательность единичных отсчетов:

```
filter(b, a, ones(1, N))
```

Для удобства такой расчет реализован в функции `stepz`, использование которой полностью аналогично только что рассмотренной функции `impz`:

```
g = stepz(b, a);
g = stepz(b, a, n);
```

Построим график переходной характеристики того же фильтра Баттервортса 5-го порядка, что использовался в предыдущем примере (рис. 4.21):

```
>> [b, a] = butter(5, 0.2);
>> stepz(b, a)
```

## Расчет частотных характеристик

В пакете Signal Processing имеется целый ряд функций, позволяющих рассчитывать зависимости различных характеристик дискретных фильтров от частоты:

- `freqz` — расчет комплексного коэффициента передачи;
- `phasez` — расчет фазочастотной характеристики;

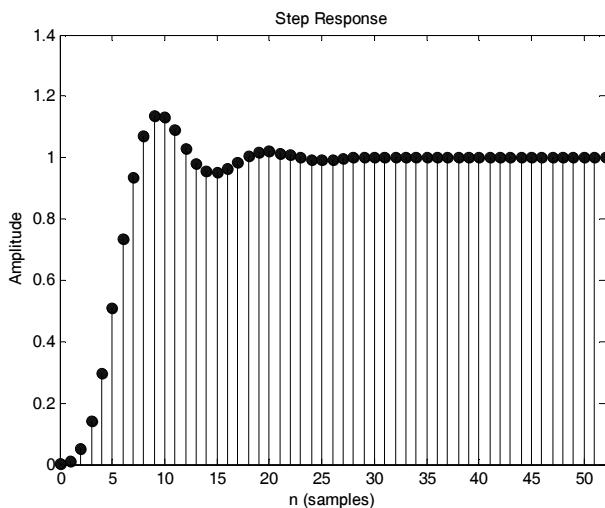


Рис. 4.21. График переходной характеристики, построенный функцией stepz

- phasedelay — расчет фазовой задержки;
- grpdelay — расчет групповой задержки.

Входные параметры перечисленных функций задаются одинаково, поэтому возможные варианты синтаксиса вызова будут рассмотрены на примере одной из них — freqz. Для остальных функций приведем лишь краткие комментарии и примеры использования.

### Комплексный коэффициент передачи

Частотная характеристика дискретного фильтра рассчитывается с помощью функции freqz, которая очень похожа на функцию freqs, выполняющую аналогичные расчеты для аналоговых цепей.

В простейшем виде функция freqz используется следующим образом:

```
freqz(b, a)
```

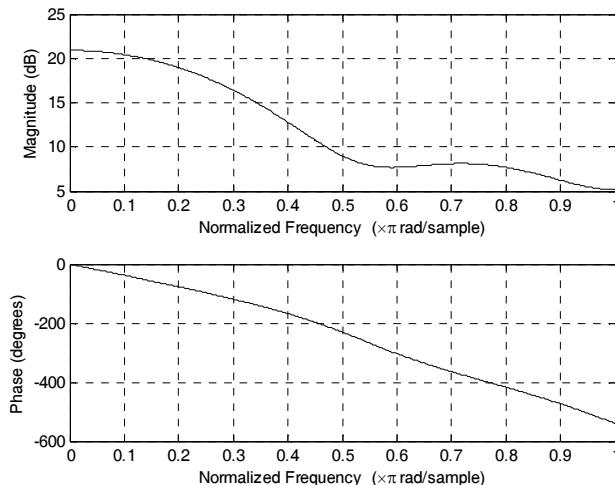
Входные параметры b и a — векторы коэффициентов полиномов числителя и знаменателя функции передачи фильтра.

При расчете используются нормированные значения частот, измеряемые в радианах на отсчет (при такой нормировке частота дискретизации равна  $2\pi$ , а частота Найквиста —  $\pi$ ). По умолчанию выбирается 512 частотных точек, равномерно распределенных в диапазоне  $0 \dots \pi$ .

При отсутствии выходных параметров функция freqz строит графики АЧХ (в децибелах) и ФЧХ (в градусах) фильтра.

Построим графики частотных характеристик фильтра, использованного при демонстрации дискретной свертки, но добавим к этому фильтру рекурсивную ветвь с коэффициентом передачи 0,1. Результат показан на рис. 4.22:

```
>> b = [1 2 3 4];
>> a = [1 -0.1];
>> freqz(b, a)
```



**Рис. 4.22.** Частотная характеристика дискретного фильтра, построенная функцией freqz

Если при вызове функции указаны выходные параметры, построение графика не производится:

```
[h, w] = freqz(b, a);
```

В векторе *h* возвращаются рассчитанные значения комплексного коэффициента передачи, а в векторе *w* — использованные при расчете значения нормированных частот.

Чтобы задать количество частотных точек для расчета, используется третий входной параметр *n* (частоты при этом по-прежнему равномерно распределяются в диапазоне 0... $\pi$ ):

```
freqz(b, a, n);
```

С помощью четвертого входного параметра можно задать частоту дискретизации:

```
freqz(b, a, n, Fs);
```

Можно заставить функцию freqz использовать ряд частот, равномерно распределенных на интервале 0... $2\pi$ , т. е. вплоть до частоты дискретизации. Это может быть необходимо при анализе фильтров с комплексными коэффициентами, частотные характеристики которых не являются симметричными. Такое указание дается функции freqz с помощью четвертого входного параметра — строки 'whole':

```
freqz(b, a, n, 'whole');
```

Наконец, можно задать вектор нормированных круговых частот *w* для расчета частотной характеристики:

```
h = freqz(b, a, w);
```

Использование второго выходного параметра в этом случае не имеет смысла, поскольку вектор частот задан на входе.

### ЗАМЕЧАНИЕ 1

При выводе графиков ФЧХ фильтров функция freqz применяет к вектору рассчитанных фазовых сдвигов функцию unwrap, чтобы устраниТЬ "фиктивные" разрывы, о которых шла речь в разд. "Построение графиков фазочастотных характеристик" главы 2.

### ЗАМЕЧАНИЕ 2

В составе пакета Signal Processing имеется еще одна функция, предназначенная для расчета частотных характеристик — zerophase. Она возвращает *вещественную* частотную характеристику, получаемую путем компенсации *непрерывной* фазовой функции в комплексной частотной характеристике. Полученная частотная зависимость не всегда совпадает с АЧХ — в отличие от последней, она может принимать и отрицательные значения. Вызывается и используется функция zerophase аналогично функции freqz.

## Задание вектора частот

Для некоторого повышения удобства задания вектора частот при расчете частотных характеристик служит функция freqspace. Эта функция очень проста и имеет следующий синтаксис вызова:

```
y = freqspace(n)
```

Возвращаемый вектор f содержит значения, не превосходящие частоту Найквиста и нормированные к этой частоте. При этом подразумевается, что на n равных частей делится частотный диапазон от нуля до *частоты дискретизации* (после нормировки к частоте Найквиста она равна двум). Значения частот в векторе f рассчитываются как  $0:2/n:1$ , так что этот вектор содержит  $\text{floor}(n/2+1)$  элементов.

При необходимости сформировать набор значений в диапазоне от нуля до двух (т. е. до частоты дискретизации) следует использовать второй входной параметр — строку 'whole':

```
y = freqspace(n, 'whole')
```

При этом вектор y содержит n элементов и рассчитывается так:

```
y = (0:n-1)*2/n
```

## Фазочастотная характеристика

Фазочастотная характеристика дискретного фильтра рассчитывается с помощью функции phasez. В простейшем виде она используется следующим образом:

```
phi = phasez(b, a)
```

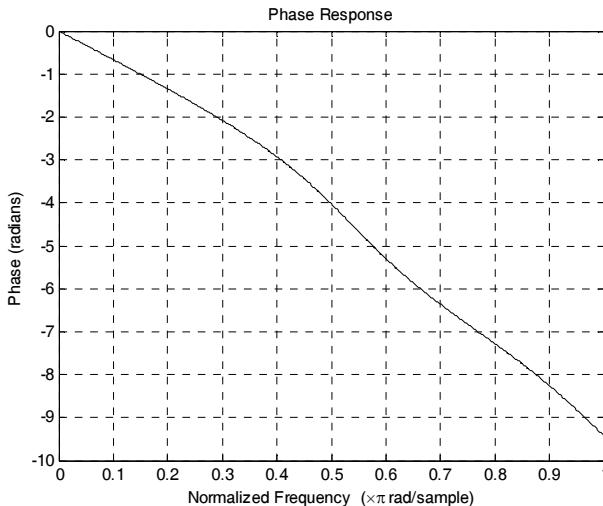
Входные параметры b и a — векторы коэффициентов полиномов числителя и знаменателя функции передачи фильтра. Выходной результат phi — вектор значений

ФЧХ (в радианах). Если выходной параметр не указан, функция строит график ФЧХ.

Дополнительные входные и выходные параметры у данной функции такие же, как у рассмотренной ранее функции `freqz`.

Построим график ФЧХ для того же фильтра, что использовался при демонстрации работы функции `freqz`:

```
>> b = [1 2 3 4];
>> a = [1 -0.1];
>> phasez(b, a)
```



**Рис. 4.23.** Фазочастотная характеристика дискретного фильтра, построенная функцией `phasez`

Результат, приведенный на рис. 4.23, показывает, что функция `phasez`, так же как и функция `freqz`, при расчете ФЧХ вызывает функцию `unwrap` для устранения скачков (см. разд. "Построение графиков фазочастотных характеристик" главы 2).

## Фазовая задержка

Фазовая задержка, вносимая дискретным фильтром, рассчитывается с помощью функции `phasedelay`. В простейшем виде она используется следующим образом:

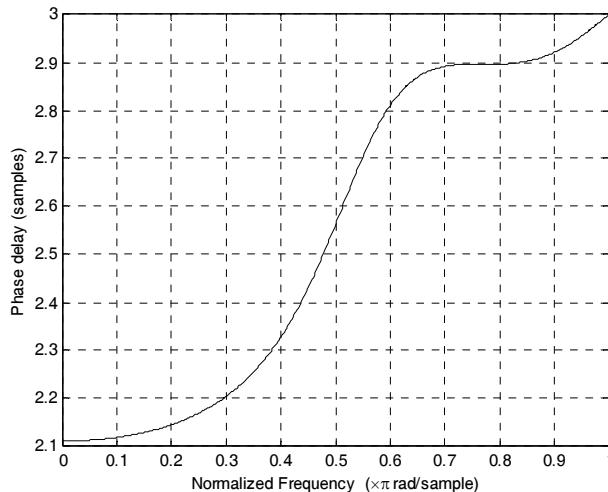
```
tau_phi = phasedelay(b, a)
```

Входные параметры `b` и `a` — векторы коэффициентов полиномов числителя и знаменателя функции передачи фильтра. Выходной результат `tau_phi` — вектор значений фазовой задержки (в отсчетах). Если выходной параметр не указан, функция строит график зависимости фазовой задержки от частоты. Расчет производится согласно формуле (2.2).

Дополнительные входные и выходные параметры у данной функции такие же, как у рассмотренной ранее функции `freqz`. При указании частоты дискретизации фазовая задержка рассчитывается и выводится на график в радианах на герц (рад/Гц), т. е. в секундах, умноженных на  $2\pi$ . Чтобы перевести этот результат в секунды, его нужно разделить на  $2\pi$ , а чтобы получить его в отсчетах — разделить на  $2\pi$  и умножить на частоту дискретизации.

Построим график ФЧХ для того же фильтра, что использовался при демонстрации работы функций `freqz` и `phasez` (результат показан на рис. 4.24):

```
>> b = [1 2 3 4];
>> a = [1 -0.1];
>> phasedelay(b, a)
```



**Рис. 4.24.** Фазовая задержка, вносимая дискретным фильтром, рассчитана функцией `phasedelay`

## Групповая задержка

Для вычисления групповой задержки дискретного фильтра в MATLAB служит функция `grpdelay`. В простейшем виде она вызывается так:

```
grpdelay(b, a)
```

Здесь `b` и `a` — векторы коэффициентов числителя и знаменателя передаточной функции фильтра соответственно. Расчет производится согласно формуле (2.3), задержка рассчитывается в отсчетах.

Дополнительные входные параметры используются точно так же, как и для рассмотренной ранее функции расчета частотной характеристики `freqz`.

При отсутствии выходных параметров функция `grpdelay` строит график зависимости группового времени задержки от частоты. При указании одного выходного па-

раметра функция возвращает вектор рассчитанных значений групповой задержки. При использовании двух выходных параметров во втором из них возвращается вектор использованных при расчете значений частот (второй выходной параметр можно использовать, если частоты для расчета не задаются принудительно среди входных параметров):

```
tau = grpdelay(b, a, ...);
[tau, w] = grpdelay(b, a, ...);
```

Как было сказано в разд. "Фазовая и групповая задержка" главы 2, при прохождении узкополосного сигнала через линейную систему его огибающая и несущее колебание приобретают *разные* задержки. Задержка несущей — это фазовая задержка, а задержка огибающей — групповая задержка. Продемонстрируем это на непростом примере, показав заодно и использование функции `grpdelay`.

Сформируем сигнал в виде радиоимпульса с треугольной огибающей. Обратите внимание на фазу несущего колебания — максимум огибающей совпадает с максимумом внутреннего заполнения (рис. 4.25, сплошная линия):

```
>> Fs = 1000; % частота дискретизации
>> t = -1:1/Fs:1.5; % вектор значений времени
>> Fc = 5; % несущая частота
>> A = (1 - abs(t)).*(abs(t)<=1); % огибающая сигнала
>> s = A.*cos(2*pi*Fc*t); % входной радиоимпульс
```

Далее пропускаем этот сигнал через фильтр нижних частот Баттерворт с частотой среза, равной несущей частоте сигнала, синтезированный методом билинейного z-преобразования (подробнее об этом и других методах проектирования дискретных фильтров см. далее *главу 6*).

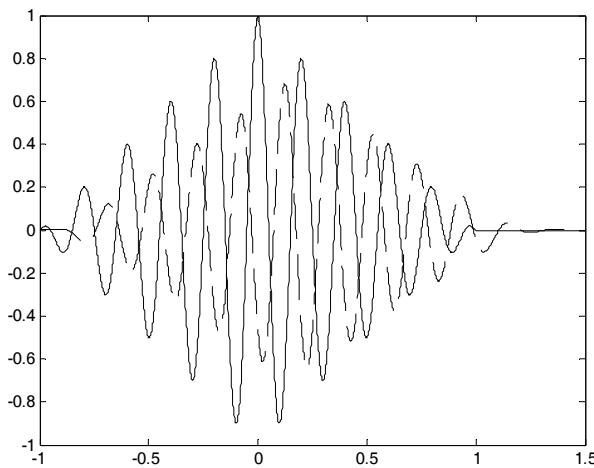
```
>> [b, a] = butter(5, Fc*2/Fs); % фильтр нижних частот
>> s1 = filter(b, a, s); % выходной сигнал фильтра
>> plot(t, s)
>> hold on
>> plot(t, s1, '--')
>> hold off
```

На рис. 4.25 пунктирной линией показан график выходного сигнала фильтра. На рисунке видно, что максимумы огибающей и несущего колебания больше не совпадают — между ними возникает некоторый временной сдвиг.

Посмотрим теперь, как это согласуется с величинами фазовой и групповой задержки:

```
>> f = 0.01:0.01:10; % частоты для расчета (в герцах)
>> h = freqz(b, a, f, Fs); % частотная характеристика
>> tau_phase = phasedelay(b, a, f, Fs); % фазовая задержка
>> tau_phase = tau_phase/2/pi; % пересчет в секунды
>> tau_group = grpdelay(b, a, f, Fs)/Fs; % групповая задержка
>> plot(f, tau_phase, '--', f, tau_group, '-')
>> ylim([0 0.2])
>> xlabel('Frequency, Hz')
```

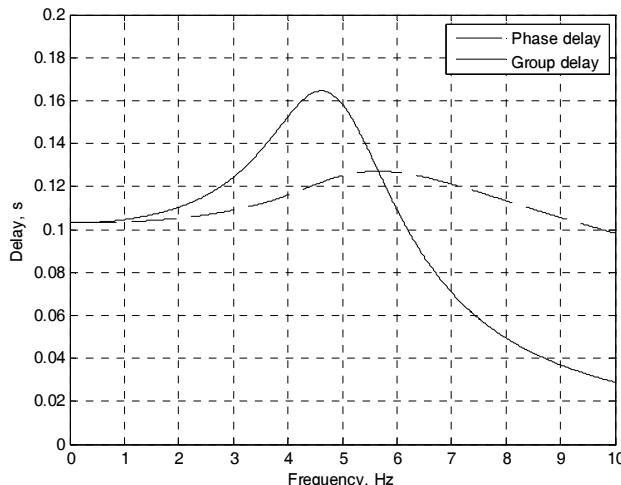
```
>> ylabel('Delay, s')
>> legend('Phase delay', 'Group delay')
>> grid on
```



**Рис. 4.25.** Входной (сверху) и выходной (снизу) сигналы полосового фильтра демонстрируют разницу между групповой и фазовой задержками

Построенные графики фазовой (пунктирная линия) и групповой (сплошная линия) задержек показаны на рис. 4.26. Видно, что на частоте 5 Гц, равной несущей частоте нашего сигнала, эти задержки различаются примерно на 0,04 с.

Таким образом, мы наглядно продемонстрировали эффект сдвига огибающей относительно несущей, вызванный различием между фазовой и групповой задержками.



**Рис. 4.26.** Фазовая (пунктирная линия) и групповая (сплошная линия) задержки, вносимые фильтром низких частот

## Отображение нулей и полюсов фильтра

Для отображения нулей и полюсов функции передачи фильтра на комплексной плоскости предназначена функция `zplane`:

```
zplane(z, p)
zplane(b, a)
```

Входными параметрами являются векторы-столбцы нулей и полюсов (`z, p`) либо векторы-строки коэффициентов полиномов числителя и знаменателя функции передачи (`b, a`). Различие этих двух случаев производится именно по ориентации передаваемых функции векторов.

Поскольку функция просто вычисляет (при необходимости) и отображает корни полиномов на комплексной плоскости, она может использоваться как для дискретных ( $z$ -плоскость), так и для аналоговых ( $s$ -плоскость) фильтров. Единственным неудобством может быть отображение единичной окружности, несущественной для аналоговых фильтров.

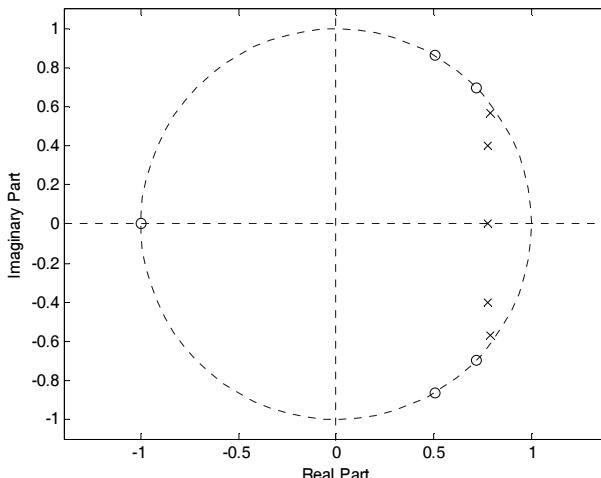
В качестве примера покажем расположение на комплексной плоскости нулей и полюсов дискретного эллиптического фильтра нижних частот 5-го порядка с уровнем пульсаций 1 дБ в полосе пропускания и затуханием 40 дБ в полосе задерживания при частоте среза, равной 0,2 частоты Найквиста (рис. 4.27):

```
>> [b, a] = ellip(5, 1, 40, 0.2);
>> zplane(b, a)
```

Нули функции передачи отображаются кружочками, полюсы — крестиками.

### ЗАМЕЧАНИЕ

Аналоговые эллиптические фильтры рассматривались в главе 2. Дискретный фильтр в данном примере рассчитан по аналоговому прототипу методом билинейного  $z$ -преобразования, которое будет обсуждаться в главе 6.



**Рис. 4.27.** Нули и полюсы эллиптического фильтра, выведенные функцией `zplane`

При необходимости можно получить дескрипторы созданных функцией графических объектов, используя выходные параметры:

```
[hz, hp, ht] = zplane(...)
```

Здесь  $hz$  — вектор дескрипторов нулей,  $hp$  — вектор дескрипторов полюсов,  $ht$  — вектор дескрипторов осей, единичной окружности и текстовых объектов.

## Свертка как матричное умножение

Дискретная свертка (4.3) представляет собой сумму поэлементных произведений, поэтому при конечной длине импульсной характеристики эту операцию можно представить как скалярное произведение двух векторов. Одним вектором при этом служит импульсная характеристика фильтра, а другим — набор отсчетов входного сигнала  $\mathbf{u}(k)$ , определяемый как

$$\mathbf{u}(k) = [x(k), x(k-1), x(k-2), \dots, x(k-N)].$$

Считая импульсную характеристику вектором-столбцом, выражение для  $k$ -го отсчета выходного сигнала можно записать следующим образом:

$$y(k) = \mathbf{u}(k) \mathbf{h}.$$

При расчете разных отсчетов выходного сигнала используются соответствующие векторы  $\mathbf{u}(k)$ . Для получения всего выходного сигнала отдельные строки  $\mathbf{u}(k)$  организуются в матрицу

$$\mathbf{U} = \begin{bmatrix} x(0) & 0 & 0 & \dots & 0 \\ x(1) & x(0) & 0 & \dots & 0 \\ x(2) & x(1) & x(0) & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ x(N) & x(N-1) & x(N-2) & \dots & x(0) \\ x(N+1) & x(N) & x(N-1) & \dots & x(1) \\ \dots & \dots & \dots & \dots & \dots \\ x(M) & x(M-1) & x(M-2) & \dots & x(M-N) \\ 0 & x(M) & x(M-1) & \dots & x(M-N+1) \\ 0 & 0 & x(M) & \dots & x(M-N+2) \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & x(M) \end{bmatrix}.$$

Эта матрица называется *матрицей свертки* (convolution matrix), а выходной сигнал фильтра может быть получен путем матрично-векторного умножения:

$$\mathbf{y} = \mathbf{U} \mathbf{h}.$$

Вычисление матрицы свертки в MATLAB производится с помощью функции `convmtx`, имеющей следующий синтаксис:

```
U = convmtx(x, n)
```

Здесь  $x$  — вектор отсчетов сигнала,  $n$  — размерность свертки (длина второго участка в свертке вектора).

Результатом работы функции является матрица свертки  $U$ . Ее размер зависит от ориентации вектора  $x$ :

- если  $x$  — столбец, то матрица  $U$  имеет  $\text{length}(x) + n - 1$  строк и  $n$  столбцов;
- если  $x$  — строка, то матрица  $U$  имеет  $n$  строк и  $\text{length}(x) + n - 1$  столбцов.

Нерекурсивный фильтр с импульсной характеристикой  $h$  с помощью матричного умножения может быть реализован следующим образом:

```
y = convmtx(x, length(h)) * h           % x, y и h — столбцы
y = h * convmtx(x, length(h))           % x, y и h — строки
```

Однако такой вариант требует значительно большего расхода памяти, чем при использовании функций `conv` и `filter`. Поэтому функцию `convmtx` следует использовать только для реализации каких-либо специфических алгоритмов обработки сигналов.

Приведем пример формирования матрицы свертки:

```
>> h = [1 2 3 4];
>> convmtx(h, 4)
ans =
    1     2     3     4     0     0     0
    0     1     2     3     4     0     0
    0     0     1     2     3     4     0
    0     0     0     1     2     3     4
```

Как видите, по краям, там, где при фильтрации формируются "хвосты" сигнала, недостающие входные данные дополняются нулями.

## Преобразование способов описания дискретных фильтров

Преобразование описаний дискретных фильтров между тремя вариантами — коэффициентами полиномов числителя и знаменателя функции передачи, наборами нулей и полюсов и параметрами пространства состояний — выполняются теми же шестью функциями, что и для аналоговых цепей (см. разд. "Преобразование способов описания линейных цепей" главы 2). Единственная особенность их использования в дискретном случае состоит в том, что векторы  $b$  и  $a$ , содержащие коэффициенты полиномов числителя и знаменателя функции передачи, должны иметь одинаковую длину. Поэтому для удобства преобразования описаний дискретных фильтров в пакете Signal Processing имеется функция `eqtflength`, дополняющая нулями в конце более короткий из переданных ей двух векторов:

```
[b1, a1] = eqtflength(b, a);
```

Что касается разложения на простые дроби, здесь преобразование дискретной системы несколько отличается от аналогового случая, поэтому для этого предусмотрена специальная функция `reziduez`.

Наконец, для дискретных систем имеет большое практическое значение последовательная, или каскадная, реализация, когда система представляется в виде последо-

вательного соединения *секций второго порядка* (second order sections). Для преобразования описания дискретной системы в эту форму и из нее служат функции `sos2ss`, `sos2tf`, `sos2zp`, `ss2sos`, `zp2sos` и `tf2sos`.

Кратко напомним имена и синтаксис функций преобразования, общих для аналоговых и дискретных систем:

- преобразование коэффициентов полиномов числителя и знаменателя функции передачи в нули, полюсы и коэффициент усиления:

$[z, p, k] = \text{tf2zp}(b, a);$

- преобразование нулей, полюсов и коэффициента усиления в коэффициенты полиномов числителя и знаменателя функции передачи:

$[b, a] = \text{zp2tf}(z, p, k);$

- преобразование коэффициентов полиномов числителя и знаменателя функции передачи в параметры пространства состояний:

$[A, B, C, D] = \text{tf2ss}(b, a);$

- преобразование параметров пространства состояний в коэффициенты полиномов числителя и знаменателя функции передачи:

$[b, a] = \text{ss2tf}(A, B, C, D);$

- преобразование параметров пространства состояний в нули, полюсы и коэффициент усиления:

$[z, p, k] = \text{ss2zp}(A, B, C, D);$

- преобразование нулей, полюсов и коэффициента усиления в параметры пространства состояний:

$[A, B, C, D] = \text{zp2ss}(z, p, k);$

Далее переходим к описанию функций преобразования, специфических для дискретных систем.

## Разложение на простые дроби

Функция `residuez` обеспечивает преобразование функции передачи дискретного фильтра в сумму простейших дробей и обратно (см. разд. "Полюсы и вычеты" этой главы). Ее отличие от функции `residue`, которая осуществляет такое преобразование для аналоговых систем, состоит в форме представления простейших дробей. Для уяснения разницы между функциями `residue` и `residuez` полезно сравнить используемые ими формулы разложения. Для функции `residue` это формула (2.12) (см. главу 2), для функции `residuez` — формула (4.12).

Как уже говорилось, функция `residuez` может выполнять преобразование в обе стороны. Требуемый тип преобразования определяется количеством входных параметров:

$[r, p, k] = \text{residuez}(b, a)$

$[b, a] = \text{residuez}(r, p, k)$

Здесь  $b$  и  $a$  — векторы коэффициентов полиномов числителя и знаменателя функции передачи соответственно,  $r$  — вектор вычетов,  $p$  — вектор полюсов,  $k$  — вектор коэффициентов целой части функции передачи.

Разложение функции передачи фильтра на простые дроби соответствует параллельной реализации фильтра (см. разд. "Параллельная форма" этой главы). Каждой простой дроби соответствует рекурсивный фильтр первого порядка; выходные сигналы этих фильтров суммируются.

В качестве примера разложим на простые дроби функцию передачи фильтра Баттерворта 3-го порядка, параллельная реализация которого была показана ранее на рис. 4.18:

```
>> [b, a] = butter(3, 0.4)
b =
    0.0985    0.2956    0.2956    0.0985
a =
    1.0000   -0.5772    0.4218   -0.0563
>> [r, p, k] = residuez(b, a)
r =
    -0.6145 - 0.1096i
    -0.6145 + 0.1096i
    3.0777
p =
    0.2094 + 0.5582i
    0.2094 - 0.5582i
    0.1584
k =
    -1.7502
```

Как видите, постоянное слагаемое и вещественный полюс с вычетом соответствуют двум верхним ветвям схемы на рис. 4.18. Для пар комплексно-сопряженных полюсов и вычетов такое сравнение выполнить сложнее, ведь на рисунке они были скомбинированы в секцию второго порядка с вещественными коэффициентами. Чтобы получить коэффициенты этой секции, выполним с помощью функции `residuez` обратное преобразование, задав в качестве входных параметров только эти пары полюсов и вычетов:

```
>> [b1, a1] = residuez(r(1:2), p(1:2), [])
b1 =
    -1.2290    0.3798
a1 =
    1.0000   -0.4189    0.3554
```

Теперь видно, что результат объединения комплексно-сопряженных пар полюсов и вычетов соответствует секции второго порядка, показанной на рис. 4.18.

### **ЗАМЕЧАНИЕ**

Хотя в последнем расчете не нужно было задавать целую часть функции передачи, нам пришлось указать соответствующий параметр в виде пустой матрицы. Без этого

функция `residuez` "не поймет", что мы хотим осуществить преобразование полюсов и вычетов в коэффициенты числителя и знаменателя функции передачи, а не наоборот.

## Функции, работающие с секциями второго порядка

В разд. "Последовательная (каскадная) форма" этой главы мы уже говорили, что разбиение структуры фильтра на последовательно включенные блоки часто используется на практике. В том, что это действительно уменьшает проблемы, связанные с конечной точностью вычислений, нам предстоит убедиться в главе 7, а пока рассмотрим функции MATLAB, работающие с таким представлением дискретных систем.

Казалось бы, для создания последовательной реализации системы достаточно рас считать ее нули и полюсы, после чего представить систему в виде каскадно включенных звеньев первого порядка. Однако часть этих звеньев (или все) может оказаться с комплексными коэффициентами. По этой причине при каскадной реализации вещественных фильтров их делят на *секции второго порядка (second-order sections)*. При этом пары комплексно-сопряженных нулей и полюсов объединяются и образуют каскады второго порядка. Легко убедиться, что для расчета выходного сигнала *вещественного* фильтра второго порядка необходимо выполнить меньше операций, чем для расчета выходного сигнала двух последовательно соединенных комплексных фильтров первого порядка.

В пакете Signal Processing имеется шесть функций, осуществляющих преобразование описания дискретной системы между набором секций второго порядка и другими формами описания. Имена этих функций построены так же, как и имена других функций преобразования описаний систем; аббревиатура последовательной формы — `sos` (second-order sections):

- преобразование секций второго порядка в коэффициенты полиномов числителя и знаменателя функции передачи:

`[b, a] = sos2tf(sos, g)`

- преобразование секций второго порядка в нули, полюсы и коэффициент усиления:

`[z, p, k] = sos2zp(sos, g)`

- преобразование секций второго порядка в параметры пространства состояний:

`[A, B, C, D] = sos2ss(sos, g)`

- преобразование коэффициентов полиномов числителя и знаменателя функции передачи в секции второго порядка:

`[sos, g] = tf2sos(b, a, 'order', 'scale')`

- преобразование нулей, полюсов и коэффициента усиления в секции второго порядка:

`[sos, g] = zp2sos(z, p, k, 'order', 'scale')`

- преобразование параметров пространства состояний в секции второго порядка:

```
[sos, g] = ss2sos(A, B, C, D, 'order', 'scale')
```

Набор секций второго порядка представляется с помощью 6-столбцовой матрицы *sos*, каждая строка которой соответствует одной секции и устроена следующим образом:

$$[b_0 \ b_1 \ b_2 \ 1 \ a_1 \ a_2].$$

Такой строке соответствует функция передачи вида

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}.$$

Параметр *g* — дополнительный множитель (*gain*), добавляемый к функции передачи, составленной из секций второго порядка. Этот параметр при вызове может опускаться, для входного параметра *g* это означает использование значения по умолчанию *g* = 1, а для выходного — учет этого коэффициента в первой секции фильтра.

Преобразование описания *из* набора секций второго порядка в какую-либо другую форму является однозначным, а обратное — нет. Ведь можно разными способами группировать нули и полюсы для формирования секций, менять порядок включения секций и по-разному распределять коэффициент усиления между ними (например, если у одной секции все коэффициенты *b<sub>i</sub>* в одинаковое число раз увеличить, а у другой — во столько же раз уменьшить, общая функция передачи фильтра останется без изменений).

Группировка нулей и полюсов функциями *xx2sos* выполняется следующим образом. Прежде всего выделяются комплексно-сопряженные пары нулей и полюсов. Эти пары образуют числители и знаменатели секций, причем для каждой секции выбирается пара нулей, расположенных ближе всего к паре полюсов секции. Для вещественных полюсов и нулей применяется аналогичный подход, при этом группируются в пары полюсы, максимально близкие по модулю.

Последовательностью включения секций можно управлять с помощью входного параметра 'order', имеющегося у функций *xx2sos*. Этот параметр определяет режим сортировки строк матрицы *sos* и может принимать одно из двух значений:

- 'up' — строки матрицы *sos* расположены в порядке приближения полюсов секций к единичной окружности (этот вариант принят по умолчанию);
- 'down' — строки матрицы *sos* расположены в порядке удаления полюсов секций от единичной окружности.

Наконец, входной параметр 'scale', имеющийся у функций *xx2sos*, управляет распределением коэффициента усиления между секциями фильтра и множителем *g*. Этот параметр может принимать одно из трех значений:

- 'none' — никакого дополнительного масштабирования не производится, во всех секциях  $b_0 = 1$ , а необходимый коэффициент усиления формируется с помощью общего множителя *g* (этот вариант принят по умолчанию);

- 'inf' — масштабирование коэффициентов  $b_i$  производится исходя из  $L_\infty$ -нормы АЧХ секций ( $L_\infty$ -норма равна максимальному абсолютному значению функции);
- 'two' — масштабирование коэффициентов  $b_i$  производится исходя из  $L_2$ -нормы АЧХ секций ( $L_2$ -норма равна корню квадратному из интеграла от квадрата модуля функции).

### ЗАМЕЧАНИЕ

Понятие нормы было рассмотрено в разд. "Пространство сигналов" главы 1. Более подробная информация о масштабировании последовательно включенных секций содержится в [8].

Естественно, масштабирование и порядок включения секций не влияют на общую функцию передачи фильтра. Однако при практической реализации фильтра с использованием вычислений с фиксированной запятой это обязательно следует принимать во внимание. Эффектам, связанным с конечной точностью вычислений, посвящена глава 7, здесь же приведем лишь два соображения, упоминаемых в документации MATLAB применительно к функциям `xx2sos`:

- при 'order' = 'up' и 'scale' = 'inf' минимизируется вероятность переполнения в процессе промежуточных вычислений;
- при 'order' = 'down' и 'scale' = 'two' минимизируется пиковое значение шума округления.

В качестве иллюстрации выполним разложение на секции второго порядка для фильтра, использованного для примера в разд. "Последовательная (каскадная) форма" этой главы (напомним, что это был фильтр Баттерворта 3-го порядка с частотой среза, равной 0,4 частоты Найквиста, синтезированный методом билинейного  $z$ -преобразования):

```
>> [b, a] = butter(3, 0.4)
b =
    0.0985    0.2956    0.2956    0.0985
a =
    1.0000   -0.5772    0.4218   -0.0563
>> [sos, g] = tf2sos(b, a)
sos =
    1.0000    1.0000        0    1.0000   -0.1584        0
    1.0000    2.0000    1.0000    1.0000   -0.4189    0.3554
g =
    0.0985
```

Полученный результат соответствует приведенному ранее на рис. 4.17. Продемонстрируем теперь варианты масштабирования и расположения секций, упомянутые выше:

```
>> [sos, g] = tf2sos(b, a, 'up', 'inf')
sos =
    0.3919    0.3919        0    1.0000   -0.1584        0
    0.2987    0.5975    0.2987    1.0000   -0.4189    0.3554
```

```
g =
0.8416
```

Как видите, изменились только коэффициенты нерекурсивных частей секций (первые три столбца матрицы `sos`) и коэффициент усиления `g`.

А теперь испытаем второй вариант масштабирования и расположения:

```
>> [sos, g] = tf2sos(b, a, 'down', 'two')
sos =
    0.3185    0.6370    0.3185    1.0000   -0.4189    0.3554
    0.3480    0.3480        0    1.0000   -0.1584        0
g =
0.8889
```

Теперь секции поменялись местами, а их коэффициенты усиления несколько изменились.

### **ЗАМЕЧАНИЕ**

В MATLAB можно непосредственно осуществлять обработку сигнала фильтром, представленным в виде секций второго порядка, не преобразуя описание фильтра к стандартной форме функции передачи. Для этого служит функция `sosfilt`, вызываемая следующим образом: `y = sosfilt(sos, x)`. Здесь `sos` — описание фильтра в рассмотренном ранее формате, `x` и `y` — соответственно входной и выходной сигналы.

## **Объекты дискретных фильтров**

В последнее время в прикладных пакетах расширения MATLAB различные алгоритмы и модели устройств все чаще реализуются в виде *объектов*. Первый класс объектов, с которым мы сталкиваемся в этой книге, предназначен для реализации дискретных фильтров. Не вдаваясь в обсуждение концепций объектно-ориентированного программирования (для этого есть много других книг), заметим, что объекты имеют некоторый набор *свойств*, а обычные функции `filter`, `freqz` и некоторые другие расширены таким образом, что они могут принимать объекты дискретных фильтров в качестве одного из параметров.

## **Общая информация об объектах MATLAB**

Поскольку объекты MATLAB встречаются нам впервые, перед рассмотрением конкретного их класса приведем некоторую общую информацию.

Перед использованием объект нужно создать. Это делается с помощью функции-конструктора. Так, для дискретного фильтра имеется целый ряд конструкторов, соответствующих различным формам реализации фильтра. Функция конструктора имеет имя `dfilt`, после которого через точку указывается идентификатор желаемой структуры. В простейшем варианте конструктор может вызываться без параметров:

```
>> hd = dfilt.structure;
```

При этом будет создан объект дискретного фильтра с именем `hd`, все свойства которого, в том числе и коэффициенты, принимают значения по умолчанию (такой

фильтр не изменяет входной сигнал, т. е. имеет функцию передачи, тождественно равную единице). Для установки нужных значений свойств используется функция `set`:

```
>> set(hd, 'name', value);
```

Здесь `hd` — идентификатор объекта, `'name'` — строка с именем свойства, `value` — задаваемое для него значение.

Считать значение свойства можно с помощью функции `get`:

```
>> value = get(hd, 'name')
```

Смысл параметров `hd` и `'name'` здесь тот же, что и для функции `set`, а результат `value` содержит текущее значение запрашиваемого свойства.

Свойства объектов доступны также через синтаксис обращения к полям структуры:

```
hd.name = value; % присваивание значения  
value = hd.name; % считывание значения
```

Конструктор объекта может вызываться с параметрами, задающими значения свойств сразу же при создании объекта. Подробнее об этом речь пойдет далее.

После создания объекта и настройки его свойств объект можно использовать, указывая его в качестве параметров различных функций (*методов объекта*). Например, для выполнения обработки сигнала дискретным фильтром уже знакомая нам функция `filter` используется следующим образом:

```
y = filter(hd, x)
```

Как видите, вместо двух векторов коэффициентов в списке параметров фигурирует *объект* дискретного фильтра.

Возможен и другой, более привычный для объектно-ориентированных языков синтаксис вызова методов объектов:

```
y = hd.filter(x)
```

В пакетах, использующих объекты, имеется множество функций, предназначенных для работы с ними. Некоторые из таких функций просто позволяют получать значения отдельных свойств более удобным путем, чем с помощью функции `get`, другие производят более специфические действия. Например, с помощью функции `info` можно получить информацию об объекте дискретного фильтра в виде строковой матрицы.

После этой вводной информации перейдем к рассмотрению конкретного класса объектов, предназначенного для реализации дискретных фильтров.

## Объекты класса `dfilt`

Для создания объекта дискретного фильтра служит функция `dfilt`. При ее вызове необходимо указать форму реализации фильтра — для этого после имени функции ставится точка и указывается соответствующий идентификатор метода (конструктора):

```
hd = dfilt.structure(...)
```

Входными параметрами функции являются коэффициенты фильтра, способ задания которых зависит от выбранной формы реализации. Возможные структуры фильтра, имена соответствующих конструкторов и способы задания входных параметров для них сведены в табл. 4.2.

**Таблица 4.2.** Возможные формы реализации объектов дискретных фильтров и способы задания коэффициентов для них

Имя конструктора <i>structure</i>	Форма реализации фильтра	Список параметров
delay	Фильтр задержки сигнала на целое число отсчетов	( <i>latency</i> ) — число отсчетов, на которое задерживается сигнал. По умолчанию данный параметр равен 1
df1	Direct Form I (прямая форма, см. рис. 4.11)	( <i>b</i> , <i>a</i> ) — два вектора, задающие коэффициенты полиномов числителя и знаменателя функции передачи фильтра
df1t	Direct Form I Transposed (транспонированная форма, см. рис. 4.16)	То же, что для df1
df2	Direct Form II (каноническая форма, см. рис. 4.13)	То же, что для df1
df2t	Direct Form II Transposed (транспонированная форма, см. рис. 4.15)	То же, что для df1
df1sos, df1tsos, df2sos, df2tsos	Четыре предыдущие структуры, реализованные в виде каскадно включенных секций второго порядка	( <i>S</i> , <i>g</i> ), где <i>S</i> — матрица описания секций второго порядка, а <i>g</i> — дополнительный коэффициент усиления (см. разд. "Функции, работающие с секциями второго порядка" ранее в этой главе)
dffir	FIR (прямая форма нерекурсивного фильтра, см. рис. 4.10)	( <i>b</i> ) — вектор отсчетов импульсной характеристики фильтра
dffirt	FIR Transposed (транспонированная форма нерекурсивного фильтра)	То же, что для dffir
dfsymfir	Symmetric FIR (нерекурсивный фильтр с четной симметрией, см. табл. 4.1, типы I и II)	То же, что для dffir, но вектор <i>b</i> должен удовлетворять условиям четной симметрии: $b(k) = b(\text{length}(b)+1-k)$
dfasymfir	Antisymmetric FIR (нерекурсивный фильтр с нечетной симметрией, см. табл. 4.1, типы III и IV)	То же, что для dffir, но вектор <i>b</i> должен удовлетворять условиям нечетной симметрии: $b(k) = -b(\text{length}(b)+1-k)$ . Если длина вектора нечетная, то средний элемент должен быть нулевым: $b((\text{length}(b)+1)/2) = 0$

Таблица 4.2 (окончание)

Имя конструктора <i>structure</i>	Форма реализации фильтра	Список параметров
fftfir	Реализация на основе быстрого преобразования Фурье (она будет рассмотрена в главе 5) методом <i>перекрытия с суммированием</i> ( <i>overlap-add</i> )	( <i>b</i> , <i>len</i> ), где <i>b</i> — вектор отсчетов импульсной характеристики фильтра, а <i>len</i> — длина блоков, на которые делится обрабатываемый сигнал. Параметр <i>len</i> можно опустить, по умолчанию его значение равно 100
statespace	Реализация в пространстве состояний	( <i>A</i> , <i>B</i> , <i>C</i> , <i>D</i> ) — четыре матрицы с параметрами описания фильтра в пространстве состояний (см. разд. "Пространство состояний" этой главы)
scalar	Фильтр нулевого порядка, реализующий умножение входного сигнала на фиксированный коэффициент	( <i>g</i> ) — скалярный коэффициент усиления. По умолчанию этот параметр равен 1
cascade	Последовательное (каскадное) соединение нескольких фильтров	( <i>hd1</i> , <i>hd2</i> , ...) — объекты соединяемых дискретных фильтров
parallel	Параллельное соединение нескольких фильтров	То же, что для cascade

### ЗАМЕЧАНИЕ

Возможны также следующие варианты для параметра 'structure', соответствующие структурам, не рассматриваемым в данной книге: allpass, cascadeallpass, cascadeddfallpass, farrowfd, farrowlinearfd, wdfallpass, latticeallpass, latticear, latticearma, latticemax, latticemin, calattice и calatticepc. Ознакомиться со способами задания коэффициентов фильтра для этих форм реализации можно по документации пакетов Signal Processing и Filter Design.

Для задания свойств объекта (их список будет приведен далее) можно указывать при вызове функции dfilt дополнительные параметры в виде пар "имя свойства — значение свойства":

```
hd = dfilt.structure(..., 'name1', value1, 'name2', value2, ...)
```

Здесь 'name1', 'name2' и т. д. — имена свойств, а value1, value2 и т. д. — соответствующие значения.

Список свойств дискретного фильтра приведен в табл. 4.3. Считывать и задавать значения этих свойств можно с помощью функций *get* и *set* соответственно.

Обработка сигнала дискретным фильтром осуществляется, как уже было сказано, с помощью функции *filter*:

```
y = filter(hd, x)
```

Здесь hd — объект дискретного фильтра, x — входной сигнал, y — выходной сигнал. Доступ к начальному и конечному внутренним состояниям фильтра при необходимости осуществляется через свойство States.

Таблица 4.3. Свойства объекта дискретного фильтра

Имя свойства	Описание
Arithmetric	Формат представления чисел при вычислениях, осуществляемых фильтром. По умолчанию данное свойство имеет значение <code>double</code> , об остальных вариантах речь пойдет в главе 7 при описании квантованных фильтров
FilterStructure	Форма реализации фильтра (см. табл. 4.2; только для чтения)
NumSamplesProcessed	Число обработанных фильтром отсчетов сигнала (только для чтения)
PersistentMemory	Значение <code>false</code> , используемое по умолчанию, заставляет фильтр обнулять внутреннее состояние перед каждым выполнением операции фильтрации. Если присвоить этому свойству значение <code>true</code> , обнуление производиться не будет
States	Внутреннее состояние фильтра (содержимое его элементов памяти). Формат этого свойства зависит от формы реализации фильтра
	<b>Наличие следующих свойств зависит от формы реализации фильтра</b>
Denominator	Коэффициенты полинома знаменателя функции передачи фильтра. Используется в структурах <code>df1</code> , <code>df1t</code> , <code>df2</code> , <code>df2t</code>
Numerator	Коэффициенты полинома числителя функции передачи фильтра. Используется в структурах <code>df1</code> , <code>df1t</code> , <code>df2</code> , <code>df2t</code> , <code>dffir</code> , <code>dffirt</code> , <code>dfsymfir</code> , <code>dfasymfir</code> , <code>fftfir</code>
sosMatrix	Матрица коэффициентов секций второго порядка. Используется в структурах <code>df1sos</code> , <code>df1tsos</code> , <code>df2sos</code> , <code>df2tsos</code>
ScaleValues	Дополнительный множитель в функции передачи фильтра. Используется в структурах <code>df1sos</code> , <code>df1tsos</code> , <code>df2sos</code> , <code>df2tsos</code>
BlockLength	Длина блока быстрого преобразования Фурье. Используется в структуре <code>fftfir</code>
NonProcessedSamples	Число отсчетов, хранящихся в буфере, но еще не обработанных. Используется в структуре <code>fftfir</code>
A, B, C, D	Параметры пространства состояния. Используется в структуре <code>statespace</code>
Gain	Скалярный коэффициент усиления. Используется в структуре <code>scalar</code>
Section	Массив объектов класса <code>dfilt</code> , представляющих отдельные секции фильтра. Используется в структурах <code>cascade</code> и <code>parallel</code>

Помимо функции `filter` пакет Signal Processing расширяет реализации функций анализа фильтров `freqz`, `phasez`, `phasedelay`, `grpdelay`, `impz`, `impzlength`, `stepz`, `zerophase` и `zplane`, так что они тоже могут вместо двух векторов `a` и `b` принимать на входе объект дискретного фильтра для расчета или отображения его соответствующих характеристик: `freqz(hd, ...)` или `hd.freqz(...)` и т. д.

Помимо уже перечисленных, с объектами дискретных фильтров могут работать следующие функции:

- `addstage(hd, hd1)` — добавляет новую секцию `hd1` к фильтру `hd` (он должен иметь структуру `cascade` или `parallel`);
- `block(hd)` — создание блока Simulink, соответствующего по структуре и параметрам объекту `hd`. У этой функции есть целый ряд дополнительных параметров, управляющих созданием блока. Ознакомиться с ними можно по документации пакета Signal Processing;
- `c = coefficients(hd)` — возвращает коэффициенты фильтра в виде массива ячеек `c`. Структура и содержимое массива зависят от формы реализации фильтра;
- `c = coeffs(hd)` — то же, что `coefficients`, но результат представлен в виде структуры, а не массива ячеек;
- `hd1 = convert(hd, 'newstruct')` — производит преобразование формы реализации дискретного фильтра. Новая форма задается параметром '`'newstruct'`', который должен иметь одно из значений, приведенных в табл. 4.2;
- `hd1 = copy(hd)` — создает *независимую копию* объекта дискретного фильтра с теми же значениями свойств, что у исходного объекта `hd`;

#### **ВНИМАНИЕ!**

Программная реализация обычной операции присваивания объектов в MATLAB (`hd1 = hd`) сводится к копированию *указателя*, так что в результате оба идентификатора ссылаются на одну и ту же область памяти. Функция же `copy` создает *новый* объект и копирует в него все значения свойств оригинала.

- `c = cost(hd)` — возвращает структуру с информацией о вычислительных затратах, необходимых для реализации фильтра `hd`;
- `disp(hd)` — выводит на экран свойства объекта;
- `fcfwrite(hd, filename, fmt)` — запись коэффициентов фильтра в текстовый файл с именем `filename` в формате, определяемом параметром `fmt`: шестнадцатеричном ('`hex`'), десятичном ('`dec`'), двоичном ('`bin`');
- `c = fftcoeffs(hd)` — возвращает частотные коэффициенты для фильтров класса `fftfir`;
- `type = firtype(hd)` — определяет тип симметрии фильтра `hd` (он должен быть нерекурсивным фильтром с линейной ФЧХ). Функция возвращает целочисленные значения от 1 до 4 (см. табл. 4.1 ранее в этой главе);
- `S = info(hd)` — возвращает матрицу строк, содержащих информацию об объекте дискретного фильтра `hd`. Если добавить второй входной параметр в виде строки '`long`', будет возвращена более подробная информация;
- `flag = isallpass(hd)` — возвращает 1, если фильтр `hd` является всепропускающим, и 0 в противном случае;
- `flag = iscasacade(hd)` — возвращает 1, если фильтр `hd` имеет каскадную структуру (см. конструктор `cascade` в табл. 4.2), и 0 в противном случае;

- `flag = isfir(hd)` — возвращает 1, если фильтр `hd` является нерекурсивным, и 0 в противном случае;
- `flag = islinphase(hd)` — возвращает 1, если фильтр `hd` имеет линейную ФЧХ, и 0 в противном случае;
- `flag = ismaxphase(hd)` — возвращает 1, если фильтр `hd` является максимально-фазовым (т. е. все нули его функции передачи лежат на комплексной плоскости *вне* единичной окружности или *на* ней), и 0 в противном случае;
- `flag = isminphase(hd)` — возвращает 1, если фильтр `hd` является минимально-фазовым (т. е. все нули его функции передачи лежат на комплексной плоскости *внутри* единичной окружности или *на* ней), и 0 в противном случае;
- `flag = isparallel(hd)` — возвращает 1, если фильтр `hd` имеет параллельную структуру (см. конструктор `parallel` в табл. 4.2), и 0 в противном случае;
- `flag = isreal(hd)` — возвращает 1, если все коэффициенты фильтра `hd` являются вещественными, и 0 в противном случае;
- `flag = isscalar(hd)` — возвращает 1, если фильтр `hd` является скалярным (см. конструктор `scalar` в табл. 4.2), и 0 в противном случае;
- `flag = issos(hd)` — возвращает 1, если фильтр `hd` составлен из секций второго порядка, т. е. если порядок любой его секции не превосходит двух, и 0 в противном случае;
- `flag = isstable(hd)` — возвращает 1, если фильтр `hd` является устойчивым, и 0 в противном случае;
- `N = nsections(hd)` — возвращает число каскадно или параллельно включенных секций дискретного фильтра `hd`;
- `N = nstages(hd)` — возвращает число каскадно или параллельно включенных секций дискретного фильтра `hd`. В отличие от функции `nsections`, данная функция работает *только* с объектами классов `cascade` и `parallel`;
- `N = nstates(hd)` — возвращает число элементов памяти, использованных в фильтре `hd`. Это число зависит как от порядка фильтра, так и от формы его реализации;
- `N = order(hd)` — возвращает порядок фильтра `hd`;
- `realizemdl(hd)` — реализует модель Simulink, эквивалентную дискретному фильтру `hd`. В отличие от функции `block` (см. выше), реализация осуществляется не с помощью готового блока, а путем построения структуры фильтра из отдельных элементов — сумматоров, элементов памяти и усилителей;
- `removestage(hd, k)` — удаляет `k`-ю секцию фильтра `hd` (он должен иметь структуру `cascade` или `parallel`);
- `reset(hd)` — обнуление внутреннего состояния фильтра `hd`;
- `setstage(hd, hd1, k)` — заменяет `k`-ю секцию фильтра `hd` (он должен иметь структуру `cascade` или `parallel`) фильтром `hd1`;

- `hd1 = sos(hd, 'order')` — преобразует объект `hd`, имеющий структуру `df1`, `df1t`, `df2` или `df2t`, в каскадно включенные секции второго порядка, представленные в виде объекта `hd1`. Назначение необязательного параметра '`order`' рассмотрено ранее в разд. "Функции, работающие с секциями второго порядка" этой главы;
- `[A, B, C, D] = ss(hd)` — рассчитывает и возвращает параметры пространства состояния, соответствующие дискретному фильтру `hd`;
- `[b, a] = tf(hd)` — рассчитывает и возвращает коэффициенты полиномов числителя (`b`) и знаменателя (`a`) функции передачи дискретного фильтра `hd`;
- `[z, p, k] = zpk(hd)` — рассчитывает и возвращает векторы нулей (`z`) и полюсов (`p`), а также коэффициент усиления (`k`) для дискретного фильтра `hd`.

В качестве примера сравним количество элементов памяти (функция `nstates`), необходимое для реализации фильтра в различных формах. Пусть это будет тот же фильтр, что использовался при демонстрации работы с секциями второго порядка ранее в этой главе (фильтр Баттерворта 3-го порядка с частотой среза, равной 0,4 частоты Найквиста, синтезированный методом билинейного  $z$ -преобразования):

```
>> [b, a] = butter(3, 0.4);
```

Создаем объект дискретного фильтра, реализующего эту функцию передачи в прямой форме (структура `df1`), и узнаем количество используемых фильтром элементов памяти:

```
>> hd_df1 = dfilt.df1(b, a);
>> hd_df1.nstates
ans =
6
```

Как следует из структурной схемы фильтра (см. рис. 4.11), число элементов памяти в данном случае равно сумме порядков числителя и знаменателя функции передачи.

Преобразуем фильтр в каноническую форму с помощью функции `convert` и снова проверим число элементов памяти:

```
>> hd_df2 = hd_df1.convert('df2');
>> hd_df2.nstates
ans =
3
```

В данном случае число элементов памяти равно порядку фильтра, т. е. максимальному из порядков числителя и знаменателя функции передачи (см. структурную схему данной формы реализации на рис. 4.13).

В заключение раздела посмотрим, каковы вычислительные затраты, необходимые для реализации фильтра в прямой форме:

```
>> hd_df1.cost
ans =
Number of Multipliers : 7
```

```

Number of Adders : 6
Number of States : 6
Multiplications per Input Sample : 7
Additions per Input Sample : 6

```

Если вызвать функцию `cost` для объекта, соответствующего канонической форме, можно убедиться, что требуемое количество математических операций для него точно такое же, отличается лишь число элементов памяти (Number of States).

## Функции расчета резонаторов второго порядка

Для расчета фильтров, описанных ранее в разд. "Резонатор второго порядка" и "Режектор второго порядка" этой главы, в пакете расширения Filter Design имеются две функции: `iirpeak` (расчет резонатора) и `iirnotch` (расчет режектора). Эти функции имеют идентичный синтаксис вызова:

```
[b, a] = iirpeak(w0, bw)
[b, a] = iirnotch(w0, bw)
```

Входной параметр `w0` определяет частоту настройки фильтра, нормированную к частоте Найквиста. На этой частоте коэффициент передачи резонатора равен единице, а режектора — нулю. Второй входной параметр `bw` задает ширину полосы пропускания (для режекторного фильтра — задерживания) по уровню  $-3$  дБ, также нормированную к частоте Найквиста.

Результатами работы функций являются векторы коэффициентов полиномов числителя (`b`) и знаменателя (`a`) функции передачи рассчитанного фильтра.

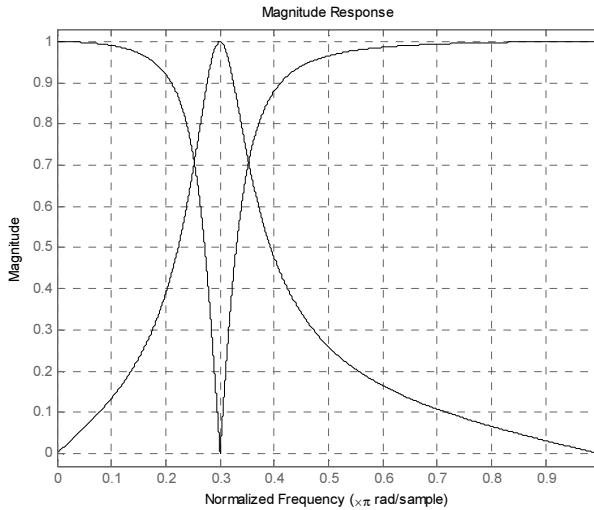
Возможен еще один вариант синтаксиса, при использовании которого с помощью дополнительного входного параметра `Ab` задается уровень (в децибелах), по которому измеряется полоса пропускания или задерживания:

```
[b, a] = iirpeak(w0, bw, Ab)
[b, a] = iirnotch(w0, bw, Ab)
```

В качестве примера использования функций `iirpeak` и `iirnotch` рассчитаем полосовой и режекторный фильтры, настроенные на частоту, равную  $0,3$  частоты Найквиста, и имеющие нормированную ширину полосы пропускания (задерживания) по уровню  $-3$  дБ, равную  $0,1$ :

```
>> [b1, a1] = iirpeak(0.3, 0.1);
>> [b2, a2] = iirnotch(0.3, 0.1);
>> fvtool(b1, a1, b2, a2)
```

АЧХ рассчитанных фильтров, графики которых получены с помощью функции `fvtool`, приведены на рис. 4.28. Видно, что кривые пересекаются именно на уровне  $-3$  дБ (примерно  $0,7$ ), по которому определялась ширина полосы пропускания и задерживания.



**Рис. 4.28.** Частотные характеристики фильтров второго порядка, рассчитанных с помощью функций `iirpeak` и `iirnotch`

## Некоторые идеализированные фильтры

В данном разделе рассмотрены характеристики некоторых идеализированных фильтров, выполняющих часто используемые в обработке сигналов операции. Эти фильтры являются нереализуемыми, поскольку их функции передачи не выражаются в виде рациональных дробей, а импульсная характеристика является бесконечной в обоих направлениях. Однако знание того, как выглядят идеализированные характеристики, полезно для понимания принципов синтеза фильтров, аппроксимирующих данные преобразования (этот вопрос рассмотрен в главе 6).

Расчет характеристик основан на том факте, что, согласно формуле (4.7), частотная характеристика любого дискретного фильтра может быть представлена в виде ряда Фурье, коэффициентами которого являются отсчеты импульсной характеристики:

$$\dot{K}(\omega) = \sum_{k=-\infty}^{\infty} h(k) e^{-j\omega kT}. \quad (4.33)$$

Нижний предел суммирования в этой формуле, в отличие от (4.7), равен минус бесконечности, т. к. в данном разделе рассматриваются идеализированные фильтры, не удовлетворяющие принципу причинности и не являющиеся, следовательно, физически реализуемыми.

Таким образом, расчет любого идеализированного фильтра может быть выполнен следующим образом:

1. Задается идеализированная частотная характеристика, которая должна быть периодической с периодом, равным частоте дискретизации.

2. Вычисляются коэффициенты разложения этой характеристики в ряд Фурье в частотной области, которые и являются отсчетами искомой импульсной характеристики.

Сравнение формулы (4.33) с формулами (1.10) и (1.12) (см. разд. "Ряд Фурье" главы 1) позволяет получить формулу для расчета коэффициентов импульсной характеристики в следующей форме:

$$h(k) = T \int_{-1/(2T)}^{1/(2T)} \dot{K}(f) e^{j2\pi fkT} df. \quad (4.34)$$

## Дискретное преобразование Гильберта

Все сказанное в разд. "Комплексная огибающая" главы 1 о преобразовании Гильберта относилось к аналоговым сигналам. В дискретном случае указанные спектральные соотношения должны выполняться в рабочей полосе частот (т. е. от нуля до частоты Найквиста). Таким образом, частотная характеристика дискретного преобразования Гильберта имеет следующий вид:

$$\dot{K}(\omega) = \begin{cases} j, & \left(k - \frac{1}{2}\right)\omega_d < \omega < k\omega_d, \\ 0, & \omega = \frac{k}{2}\omega_d, \\ -j, & k\omega_d < \omega < \left(k + \frac{1}{2}\right)\omega_d, \end{cases}$$

для всех целочисленных  $k$ . Графики АЧХ и ФЧХ дискретного преобразования Гильберта показаны на рис. 4.29.

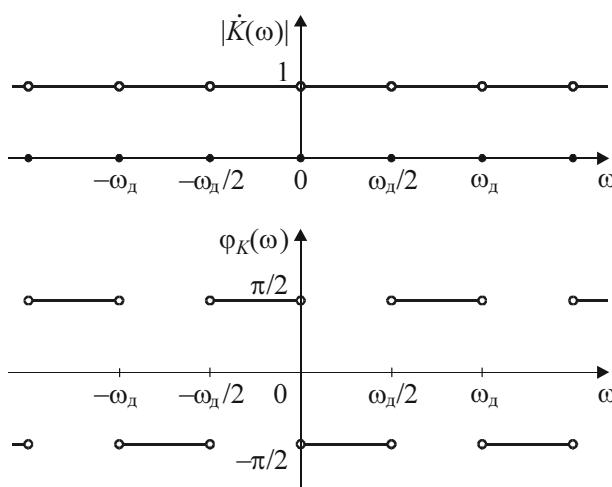


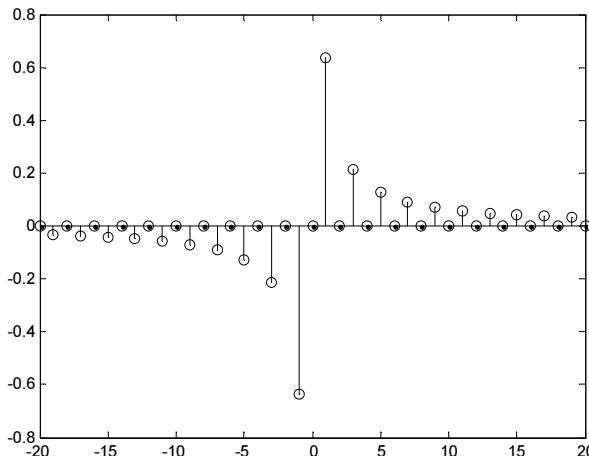
Рис. 4.29. АЧХ (сверху) и ФЧХ (снизу) дискретного преобразования Гильберта

Воспользовавшись формулой (4.34), получаем следующее:

$$h(k) = \begin{cases} 0, & k \text{ четно}, \\ \frac{2}{\pi k}, & k \text{ нечетно}. \end{cases}$$

Как и для аналогового случая, импульсная характеристика получилась физически нереализуемой — она является бесконечно протяженной в обоих направлениях. Ее график показан на рис. 4.30:

```
>> k = -20:20;
>> h = 2 / pi ./ k .* mod(k, 2);
>> h(~k) = 0; % убираем Inf из-за деления на ноль
>> stem(k, h)
```

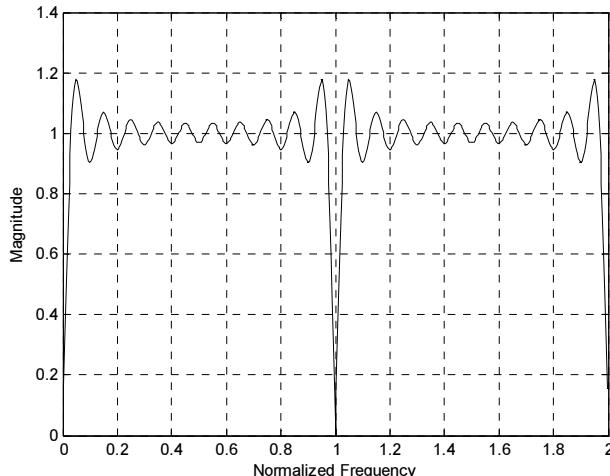


**Рис. 4.30.** Импульсная характеристика дискретного преобразования Гильберта

Поскольку фильтр оказывается физически нереализуемым, осуществить дискретное преобразование Гильберта можно лишь приближенно. Простое усечение импульсной характеристики во времени приводит к нежелательным эффектам, связанным с явлением Гиббса. Поэтому фильтры, приближенно реализующие преобразование Гильберта, приходится аппроксимировать с применением более сложных алгоритмов. Речь о них пойдет в разд. "Реализация метода Ремеза" главы 6.

Для иллюстрации искажений частотных характеристик, возникающих при усечении импульсной характеристики во времени, построим график АЧХ фрагмента импульсной характеристики, приведенного выше на рис. 4.30 (симметрия используемого фрагмента импульсной характеристики является принципиальной для сохранения линейной ФЧХ фильтра). Результаты показаны на рис. 4.31 (на горизонтальной оси графика отложены нормированные значения частоты, такие, что частота Найквиста равна единице):

```
>> [hh, f] = freqz(h, 1, 'whole');
>> plot(f/pi, abs(hh)), grid
>> xlabel('Normalized Frequency')
>> ylabel('Magnitude')
```



**Рис. 4.31.** Искажения, возникающие при усечении импульсной характеристики преобразования Гильберта

На рис. 4.31 хорошо заметны выбросы АЧХ вблизи частот, кратных частоте Найквиста. Величина максимального выброса составляет примерно 18% от номинального коэффициента передачи (это согласуется с тем, что говорилось в главе 1 применительно к разложению меандра в ряд Фурье — в данном случае коэффициент передачи меняется скачком от  $-j$  до  $j$ , так что величина скачка равна двум, и выброс составляет 9% от величины этого скачка).

Для уменьшения пульсаций АЧХ при синтезе фильтров, выполняющих преобразование Гильберта, задают переходные полосы вблизи нулевой частоты и частоты Найквиста. Коэффициент передачи фильтра в этих полосах считается неопределенным, и оптимизация производится только в оставшейся (рабочей) полосе частот.

### Функция *hilbert*

В MATLAB для формирования аналитического сигнала служит функция *hilbert*. Синтаксис ее вызова более чем прост:

```
y = hilbert(x)
```

Здесь *x* — входной вещественный сигнал (если *x* — матрица, отдельные столбцы обрабатываются независимо), *y* — сформированный комплексный аналитический сигнал.

**ВНИМАНИЕ!**

Вопреки своему названию, функция `hilbert` осуществляет не преобразование Гильберта, а именно формирование аналитического сигнала. Преобразование Гильберта с помощью этой функции можно реализовать, взяв мнимую часть результата: `imag(hilbert(x))`.

Формирование аналитического сигнала функцией `hilbert` производится в частотной области. Вычисляется дискретное преобразование Фурье (оно рассмотрено в главе 5) входного сигнала, далее обнуляется половина спектра (т. е. создается односторонний спектр аналитического сигнала) и, наконец, производится обратное преобразование Фурье.

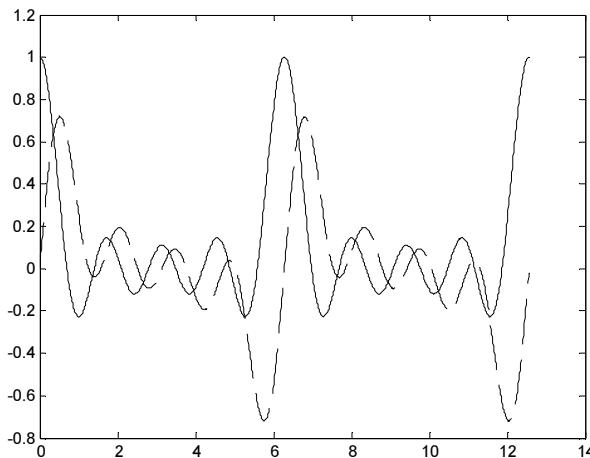
В качестве второго аргумента можно указать желаемую размерность дискретного преобразования Фурье:

```
y = hilbert(x, n)
```

При этом входные данные `x` усекаются или дополняются нулями до заданного размера.

**ЗАМЕЧАНИЕ**

Как указывалось ранее, преобразование Гильберта физически нереализуемо. Однако это ограничение можно обойти, если известны (точнее, считаются известными) будущие значения сигнала. Например, мы можем считать, что сигнал *закончился* (т. е. все его будущие значения равны нулю) или что он *периодически продолжается* за пределами области определения. Так как функция `hilbert` использует дискретное преобразование Фурье, в ней неявно подразумевается последний из перечисленных вариантов — периодическое продолжение сигнала.



**Рис. 4.32.** Функция Дирихле (сплошная линия) и ее преобразование Гильберта (пунктирная линия)

В качестве примера вычислим аналитический сигнал для функции Дирихле 9-го порядка (рис. 4.32):

```
>> t = 0:0.01:4*pi; % рассматриваем два периода функции
>> x = diric(t, 9);
>> y = hilbert(x);
>> plot(t, real(y), t, imag(y), '--')
```

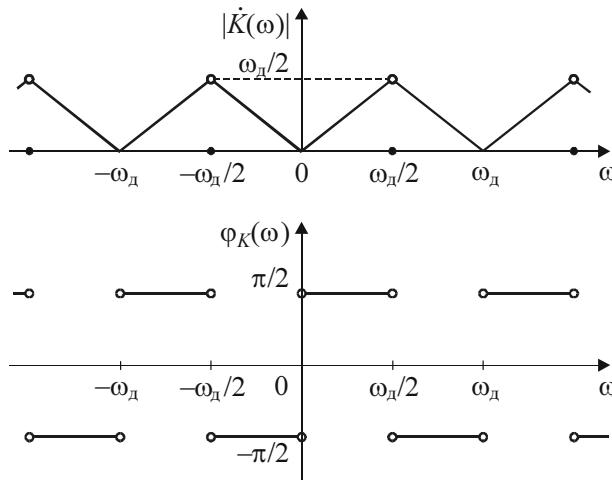
На графике видно, что вблизи тех мест, где исходный сигнал имеет экстремумы, его преобразование Гильберта пересекает ось абсцисс, и наоборот.

## Идеальный дифференцирующий фильтр

В разд. "Свойства преобразования Фурье" главы 1 было показано, что при дифференцировании сигнала по времени его спектральная функция умножается на  $j\omega$ . В дискретной дифференцирующей системе это соотношение должно выполняться в полосе частот от нуля до частоты Найквиста. Таким образом, частотная характеристика идеального дискретного дифференцирующего фильтра имеет вид

$$\dot{K}(\omega) = \begin{cases} j(\omega - k\omega_d), & |\omega - k\omega_d| < \frac{\omega_d}{2}, \\ 0, & |\omega - k\omega_d| = \frac{\omega_d}{2}. \end{cases}$$

для всех целочисленных  $k$ . Графики АЧХ и ФЧХ идеального дискретного дифференцирующего фильтра показаны на рис. 4.33.



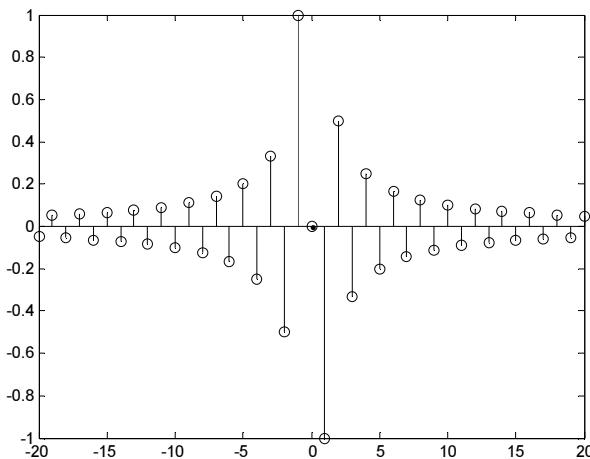
**Рис. 4.33.** АЧХ (сверху) и ФЧХ (снизу) идеального дискретного дифференцирующего фильтра

Применение к этой частотной характеристике формулы (4.34) дает следующее:

$$h(k) = \begin{cases} 0, & k = 0, \\ \frac{(-1)^k}{kT}, & k \neq 0. \end{cases}$$

Разумеется, импульсная характеристика получилась физически нереализуемой — она является бесконечно протяженной в обоих направлениях. Кроме того, импульсная характеристика не является безразмерной — из-за деления на  $T$  она имеет размерность частоты. Причиной является тот факт, что дифференцирование аналоговой функции меняет ее размерность. Построим график импульсной характеристики идеального дискретного дифференцирующего фильтра, считая период дискретизации равным единице (рис. 4.34):

```
>> k = -20:20;
>> h = (-1).^k ./ k;
>> h(~k) = 0; % убираем Inf из-за деления на ноль
>> stem(k, h)
```



**Рис. 4.34.** Импульсная характеристика идеального дискретного дифференцирующего фильтра

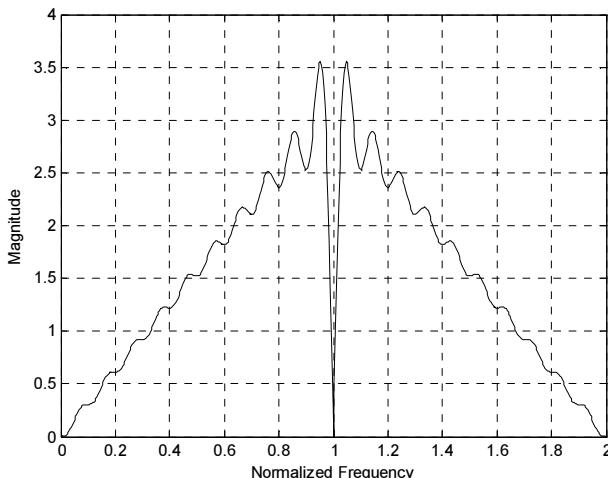
Поскольку фильтр оказывается физически нереализуемым, осуществить дифференцирование дискретного сигнала можно лишь приближенно. Простое усечение импульсной характеристики во времени приводит к нежелательным эффектам, связанным с явлением Гиббса. Поэтому фильтры, приближенно реализующие дифференцирование, приходится аппроксимировать с применением более сложных алгоритмов. Они рассмотрены в разд. "Реализация метода Ремеза" главы 6.

Для иллюстрации искажений частотных характеристик, возникающих при простом усечении импульсной характеристики во времени, построим график АЧХ фрагмента импульсной характеристики, приведенного выше на рис. 4.34 (симметрия используемого фрагмента импульсной характеристики является принципиальной для сохранения линейной ФЧХ фильтра). Результаты показаны на рис. 4.35:

```

>> [hh, f] = freqz(h, 1, 'whole');
>> plot(f/pi, abs(hh)), grid
>> xlabel('Normalized Frequency')
>> ylabel('Magnitude')

```



**Рис. 4.35.** Искажения АЧХ, возникающие при усечении импульсной характеристики идеального дифференцирующего фильтра

На рис. 4.35 хорошо заметны пульсации АЧХ, усиливающиеся при приближении к частоте Найквиста. Для уменьшения этих пульсаций при синтезе дифференцирующих фильтров применяют те же подходы, что и для фильтров, выполняющих преобразование Гильберта — задают переходные полосы вблизи нулевой частоты и частоты Найквиста, считают коэффициент передачи фильтра в этих полосах неопределенным и производят оптимизацию только в оставшейся (рабочей) полосе частот.

## Идеальный фильтр задержки

Записать выражение для импульсной характеристики фильтра, задерживающего дискретный сигнал на целое число отсчетов, не составляет труда:

$$h(k) = \begin{cases} 1, & k = k_0, \\ 0, & k \neq k_0, \end{cases}$$

где  $k_0$  — целочисленная величина задержки. Однако во многих приложениях возникает необходимость реализовывать задержку сигнала не только на целое, но и на *дробное* число отсчетов. Эту задачу мы сейчас и обсудим.

В соответствии со свойствами преобразования Фурье (см. разд. "Свойства преобразования Фурье" главы 1), частотная характеристика идеального устройства, задерживающего сигнал на  $k_0$  отсчетов, имеет вид

$$\dot{K}(\omega) = \exp(-j\omega k_0 T).$$

Используя эту частотную характеристику в формуле (4.34), можно получить импульсную характеристику идеального задерживающего фильтра в рамках общего подхода, используемого в данном разделе. Однако в представленном случае несложно получить результат и непосредственно во временной области.

Задержку дискретного сигнала на произвольный временной интервал можно представить как совокупность следующих трех операций:

- восстановление аналогового сигнала согласно теореме Котельникова (см. формулу (3.12) в главе 3);
- задержка аналогового сигнала на время  $\tau = k_0 T$ ;
- дискретизация полученного сигнала в моменты времени  $kT$ .

Математическая запись перечисленных действий приводит к следующей формуле, связывающей входной и выходной сигналы:

$$y(k) = \sum_{n=-\infty}^{\infty} x(n) \frac{\sin(\pi(k - n - k_0))}{\pi(k - n - k_0)}. \quad (4.35)$$

Сравнивая эту формулу с формулой (4.3), описывающей преобразование сигнала дискретным фильтром, можно сделать вывод о том, что импульсная характеристика идеального задерживающего фильтра имеет вид

$$h(k) = \frac{\sin(\pi(k - k_0))}{\pi(k - k_0)}. \quad (4.36)$$

Подчеркнем еще раз, что величина  $k_0$  в данной формуле не обязательно является целочисленной. Построим график этой функции для случая  $k_0 = 0,25$  (рис. 4.36; пунктиром показан график дискретизируемой функции sinc):

```
>> k = -5:5;
>> k0 = 0.25;
>> h = sinc(k - k0);
>> stem(k, h, 'filled')
>> t = -5:0.1:5;
>> hold on
>> plot(t, sinc(t - k0), '--')
>> hold off
```

Таким образом, импульсная характеристика задерживающего фильтра представляет собой отсчеты функции  $\sin(x)/x$ , взятые с соответствующим сдвигом по времени.

Из рис. 4.36 видно, что получить физически реализуемую аппроксимацию дробной задержки можно только если целая часть величины  $k_0$  достаточно велика — в этом случае отбрасываемый слева "хвост" имеет малый уровень. А вот если попытаться осуществить задержку, скажем, на те же 0,25 отсчета, погрешности окажутся значительными, т. к. в формуле (4.35) в этом случае велик вклад от будущих, еще не пришедших на вход отсчетов с номерами  $n > k$ .

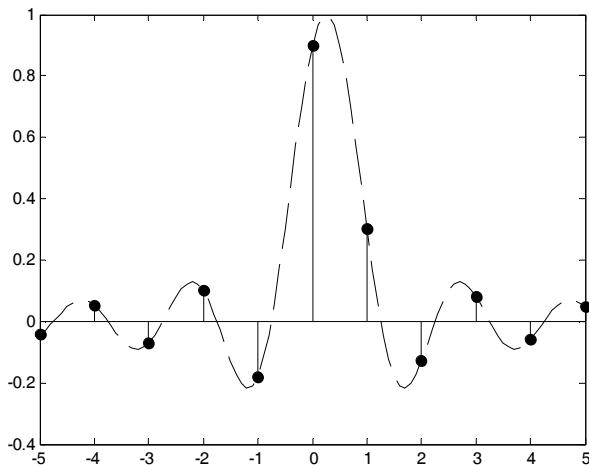
Построим графики АЧХ физически реализуемых фильтров 20-го порядка, полученных согласно (4.36) для  $k_0 = 0,25$  и  $k_0 = 10,25$  (рис. 4.37):

```
>> N = 20;
>> k = 0:N;
>> h025 = sinc(k - 0.25);
```

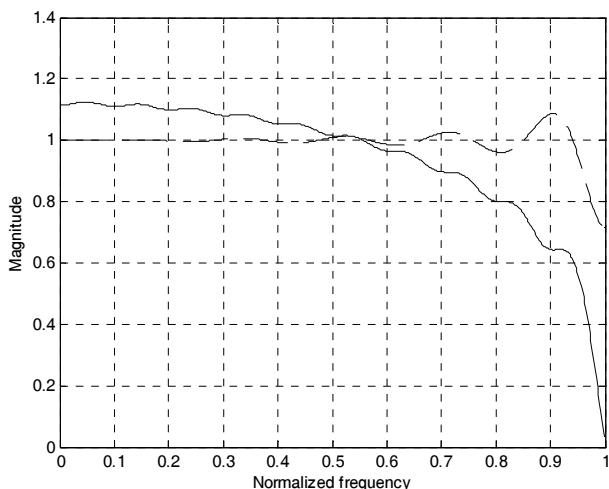
```

>> h1025 = sinc(k - 10.25);
>> [H025, f] = freqz(h025);
>> H1025 = freqz(h1025);
>> plot(f/pi, abs(H025));
>> hold on
>> plot(f/pi, abs(H1025), '--'); grid on
>> hold off
>> xlabel('Normalized frequency')
>> ylabel('Magnitude')

```



**Рис. 4.36.** Импульсная характеристика идеального фильтра задержки сигнала на 0,25 отсчета



**Рис. 4.37.** АЧХ фильтров 20-го порядка, получаемых при усечении идеализированного фильтра задержки на 0,25 отсчета (сплошная линия) и 10,25 отсчетов (пунктирная линия)

Из графиков видно, что при задержке на 10,25 отсчетов неравномерность АЧХ существенно меньше, чем при задержке на 0,25 отсчета.

Проблема реализации малых задержек обычно решается за счет того, что сигнал, как правило, необходимо задержать *относительно другого сигнала*. Таким образом, имеют значение не абсолютные, а *относительные* задержки. Реализовать же относительную задержку на 0,25 отсчета можно, например, задержав один сигнал на 10 отсчетов, а другой — на 10,25 отсчетов.

С искажениями АЧХ, имеющими место даже при большой целой части реализуемой задержки, борются так же, как в прочих случаях — ограничивая рабочую полосу частот фильтра и задавая переходную полосу в окрестностях частоты Найквиста, где частотная характеристика идеализированного фильтра при нецелочисленной задержке имеет разрыв.

## Визуализатор фильтров

Рассмотренные ранее функции `freqz`, `impz` и т. д. позволяют получать графики отдельных характеристик дискретного фильтра. Если же необходимо иметь возможность просмотреть все характеристики фильтра, можно воспользоваться графическим пользовательским интерфейсом — визуализатором фильтров. Он реализован в виде функции `fvtool` (*Filter Visualization Tool*). Простейший вариант синтаксиса вызова этой функции следующий:

```
fvtool(b, a)
```

Здесь `b` и `a` — векторы коэффициентов полиномов числителя и знаменателя функции передачи соответственно.

Вместо коэффициентов полиномов можно передать функции объект дискретного фильтра:

```
fvtool(hd)
```

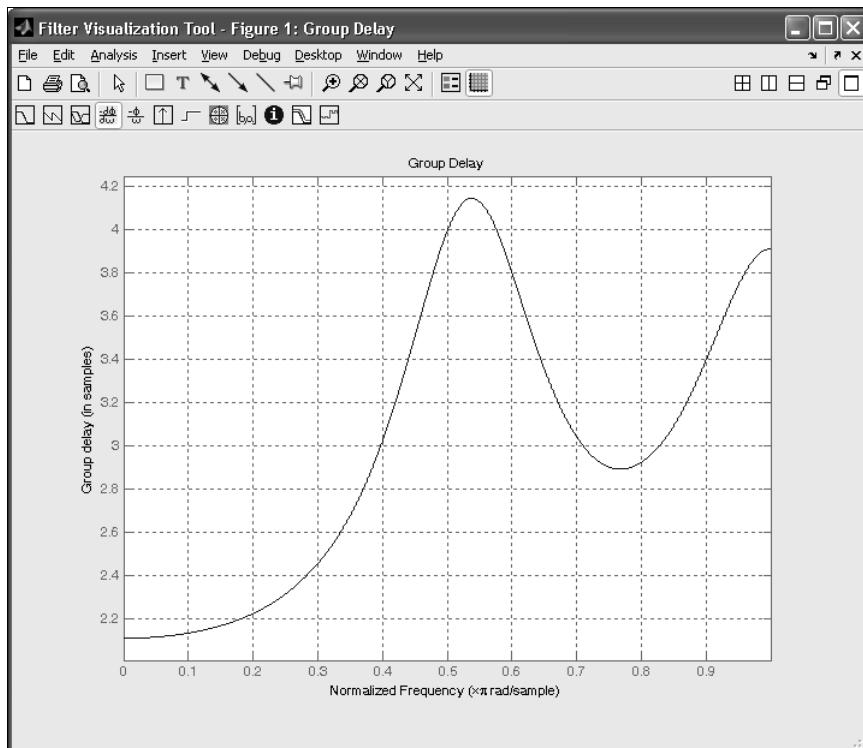
Ценным свойством является возможность вызвать функцию `fvtool` для одновременного просмотра характеристик нескольких фильтров. В этом случае следует указать в списке входных параметров несколько пар векторов или несколько объектов:

```
fvtool(b1, a1, b2, a2, ...)  
fvtool(hd1, hd2, ...)
```

В качестве примера вызовем функцию `fvtool` для просмотра характеристик того же фильтра, что использовался ранее, при описании функции `freqz` (см. рис. 4.22):

```
>> b = [1 2 3 4];  
>> a = [1 -0.1];  
>> fvtool(b, a)
```

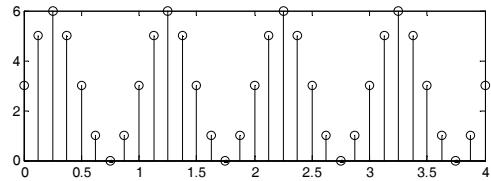
Появится окно, показанное на рис. 4.38. Выбор отображаемой характеристики производится с помощью кнопок панели инструментов. На рис. 4.38 представлен график частотной зависимости групповой задержки анализируемого фильтра.



**Рис. 4.38.** Просмотр характеристик дискретного фильтра с помощью функции `fvtool`

Интерфейс окна просмотра характеристик фильтра аналогичен интерфейсу программы расчета и анализа фильтров FDATool, которая рассмотрена в главе 6.

# ГЛАВА 5



## Спектральный анализ

В главе 3 было показано, что при дискретизации аналогового сигнала его спектр становится периодическим с периодом повторения, равным частоте дискретизации. Однако одного только этого соотношения оказывается недостаточно для решения всех практических задач спектрального анализа. Во-первых, в качестве исходных данных выступает именно последовательность дискретных отсчетов, а не аналоговый сигнал. Во-вторых, в большинстве случаев анализируемые сигналы являются случайными процессами, что требует выполнения какого-либо усреднения при расчете их спектров. Кроме того, в ряде случаев нам известна некоторая дополнительная информация об анализируемом сигнале, и эту информацию желательно учесть в спектральном анализе.

Обо всех этих аспектах спектрального анализа и пойдет речь в данной главе. Прежде всего мы рассмотрим *дискретное преобразование Фурье* (ДПФ) — разновидность преобразования Фурье, специально предназначенную для работы с дискретными сигналами. Далее обсудим идеи, лежащие в основе алгоритмов *быстрого преобразования Фурье*, позволяющих значительно ускорить вычисления.

Дискретное преобразование Фурье, по возможности вычисляемое быстрыми методами, лежит в основе различных технологий спектрального анализа, предназначенных для исследования случайных процессов. Дело в том, что если анализируемый сигнал представляет собой случайный процесс, то простое вычисление его ДПФ обычно не представляет большого интереса, т. к. в результате получается лишь спектр единственной реализации процесса. Поэтому для спектрального анализа случайных сигналов необходимо использовать усреднение спектра. Такие методы, в которых используется только информация, извлеченная из самого входного сигнала, называются *непараметрическими* (*nonparametric*).

Другой класс методов предполагает наличие некоторой статистической модели случайного сигнала. Процесс спектрального анализа в данном случае включает в себя определение параметров этой модели, и потому такие методы называются *параметрическими* (*parametric*). Используется также термин "модельный спектральный анализ" (*Model-Based Spectrum Analysis, MBSA*).

В MATLAB имеются функции, реализующие разнообразные методы спектрального анализа — как параметрические, так и непараметрические. Все они рассмотрены в этой главе.

## Дискретное преобразование Фурье

В разд. "Спектр дискретного сигнала" главы 3 мы проанализировали явления, происходящие со спектром при дискретизации сигнала. Рассмотрим теперь, что представляет собой спектр дискретного *периодического* сигнала.

Итак, пусть последовательность отсчетов  $\{x(k)\}$  является периодической с периодом  $N$ :

$$x(k + N) = x(k) \quad \text{для любого } k.$$

Такая последовательность полностью описывается *конечным* набором чисел, в качестве которого можно взять произвольный фрагмент длиной  $N$ , например  $\{x(k), k = 0, 1, \dots, N - 1\}$ . Поставленный в соответствие этой последовательности сигнал из смещенных по времени дельта-функций:

$$s(t) = \sum_{k=-\infty}^{\infty} x(k) \delta(t - kT) \quad (5.1)$$

также, разумеется, будет периодическим с минимальным периодом  $NT$ .

Так как сигнал (5.1) является дискретным, его спектр должен быть *периодическим* с периодом  $2\pi/T$ . Так как этот сигнал является также и периодическим, его спектр должен быть *дискретным* с расстоянием между гармониками, равным  $2\pi/(NT)$ .

Итак, периодический дискретный сигнал имеет периодический дискретный спектр, который также описывается конечным набором из  $N$  чисел (один период спектра содержит  $\frac{2\pi}{T} / \frac{2\pi}{NT} = N$  гармоник).

Рассмотрим процедуру вычисления спектра периодического дискретного сигнала. Так как сигнал периодический, будем раскладывать его в ряд Фурье. Коэффициенты  $\dot{X}(n)$  этого ряда, согласно общей формуле (1.12), равны

$$\begin{aligned} \dot{X}(n) &= \frac{1}{NT} \int_0^{NT} s(t) e^{-j\omega_n t} dt = \frac{1}{NT} \int_0^{NT} \sum_{k=0}^{N-1} x(k) \delta(t - kT) e^{-j\omega_n t} dt = \\ &= \frac{1}{NT} \sum_{k=0}^{N-1} x(k) \int_0^{NT} \delta(t - kT) e^{-j\omega_n t} dt = \frac{1}{NT} \sum_{k=0}^{N-1} x(k) e^{-j\omega_n kT} = \\ &= \frac{1}{NT} \sum_{k=0}^{N-1} x(k) \exp\left(-j \frac{2\pi n k}{N}\right). \end{aligned} \quad (5.2)$$

Таким образом, формула для вычисления комплексных амплитуд гармоник представляет собой линейную комбинацию отсчетов сигнала.

В выражении (5.2) реальный масштаб времени фигурирует только в множителе  $1/T$  перед оператором суммирования. При рассмотрении дискретных последовательностей обычно оперируют номерами отсчетов и спектральных гармоник без привязки к действительному масштабу времени и частоты. Поэтому множитель  $1/T$  из (5.2) удаляют, т. е. считают частоту дискретизации равной единице. Удаляют обычно и множитель  $1/N$  (об этом см. замечание далее). Получившееся выражение называет-

ся дискретным преобразованием Фурье (ДПФ; английский термин — *Discrete Fourier Transform*, DFT):

$$\dot{X}(n) = \sum_{k=0}^{N-1} x(k) \exp\left(-j \frac{2\pi n k}{N}\right). \quad (5.3)$$

Существует и *обратное* дискретное преобразование Фурье. Переход от дискретного спектра к временным отсчетам сигнала выражается следующей формулой:

$$x(k) = \frac{1}{N} \sum_{n=0}^{N-1} \dot{X}(n) \exp\left(j \frac{2\pi n k}{N}\right). \quad (5.4)$$

Это выражение отличается от формулы прямого ДПФ (5.3) лишь знаком в показателе комплексной экспоненты и наличием множителя  $1/N$  перед оператором суммирования.

### ЗАМЕЧАНИЕ

В размещении множителя  $1/N$  в формулах (5.3) и (5.4) нет полного единства. В большинстве источников, среди которых [1, 4, 8], а также в математических пакетах компьютерных программ (в том числе и в MATLAB), этот множитель фигурирует в формуле *обратного* ДПФ (5.4) (этот вариант принят и в данной книге). В то же время в учебнике [2] этот множитель включен в формулу *прямого* ДПФ (5.3). Встречается (особенно в "чисто математических" источниках) также симметричный вариант, когда в формулах прямого и обратного ДПФ фигурируют одинаковые множители, равные  $1/\sqrt{N}$ .

## Свойства дискретного преобразования Фурье

В целом свойства ДПФ аналогичны свойствам непрерывного преобразования Фурье (см. главу 1), однако дискретный характер анализируемого сигнала привносит некоторую специфику.

### Линейность

Из формулы (5.3) очевидно, что ДПФ является линейным, т. е. если последовательностям  $\{x(k)\}$  и  $\{y(k)\}$  с одним и тем же периодом  $N$  соответствуют наборы гармоник  $\{\dot{X}(n)\}$  и  $\{\dot{Y}(n)\}$ , то последовательности  $\{ax(k) + by(k)\}$  будет соответствовать спектр  $a\dot{X}(n) + b\dot{Y}(n)$ .

### Задержка

Если задержать исходную последовательность на один такт ( $y(k) = x(k - 1)$ ), то, согласно (5.3), спектр необходимо умножить на  $\exp\left(-j \frac{2\pi n}{N}\right)$ :

$$\dot{Y}(n) = \dot{X}(n) \exp\left(-j \frac{2\pi n}{N}\right).$$

Поскольку мы считаем последовательность  $\{x(k)\}$  периодической, рассматриваемый здесь сдвиг является циклическим:  $y(0) = x(-1) = x(N-1)$ .

## Симметрия

Как уже отмечалось, спектр дискретного периодического сигнала является периодическим. Кроме того, сохраняется и свойство симметрии, которым обладает спектр непрерывного вещественного сигнала ( $\dot{S}(-\omega) = \dot{S}^*(\omega)$ ). Поэтому

$$\dot{X}(N-n) = \dot{X}(-n) = \dot{X}^*(n). \quad (5.5)$$

Гармоника с нулевым номером (постоянная составляющая), как видно из (5.3), представляет собой сумму отсчетов последовательности на одном периоде:

$$\dot{X}(0) = \sum_{k=0}^{N-1} x(k). \quad (5.6)$$

Если  $N$  четно, то амплитуда гармоники с номером  $N/2$  является суммой отсчетов с чередующимися знаками:

$$\dot{X}\left(\frac{N}{2}\right) = x(0) - x(1) + \dots + x(N-2) - x(N-1) = \sum_{k=0}^{N-1} (-1)^k x(k).$$

Согласно (5.5), спектр является "сопряженно-симметричным" относительно  $N/2$ , т. е. содержит ровно такое же количество информации, что и сам сигнал. В самом деле, исходная последовательность представляется набором из  $N$  вещественных чисел. Спектр же представляется набором из  $N/2$  (вторая половина взаимно-однозначно связана с первой) комплексных чисел, каждое из которых с информационной точки зрения эквивалентно двум вещественным. Если же исходная последовательность  $\{x(k)\}$  не является вещественной, симметрия спектра отсутствует и  $N$  комплексным отсчетам во временной области соответствует  $N$  комплексных отсчетов в спектральной области.

## ДПФ произведения последовательностей

Возьмем две последовательности отсчетов  $\{x_1(k)\}$  и  $\{x_2(k)\}$  одинаковой длины  $N$  и вычислим результат их поэлементного умножения:

$$y(k) = x_1(k)x_2(k).$$

Если применить к этой формуле прямое ДПФ, получится следующее выражение:

$$\dot{Y}(m) = \frac{1}{N} \sum_{n=0}^{N-1} \dot{X}_1(n) \dot{X}_2(m-n). \quad (5.7)$$

### ЗАМЕЧАНИЕ

При выполнении вычислений по формуле (5.7) могут понадобиться значения  $\dot{X}_2(n)$  с номерами  $m$ , выходящими за рамки диапазона  $0 \dots N-1$ . В этом случае следует воспользоваться свойством периодичности спектра:  $\dot{X}_2(n) = \dot{X}_2(n \pm N)$ .

Это выражение представляет собой *круговую свертку* спектров  $\dot{X}_1(n)$  и  $\dot{X}_2(n)$ . Итак, как и для непрерывного преобразования Фурье, спектр произведения является сверткой спектров.

При  $n = 0$  из (5.7) получается дискретный аналог *теоремы Рэлея* (см. формулу (1.31) в разд. "Энергетические расчеты в спектральной области" главы 1):

$$\dot{Y}(0) = \sum_{k=0}^{N-1} x_1(k)x_2(k) = \frac{1}{N} \sum_{n=0}^{N-1} \dot{X}_1(n)\dot{X}_2(-n) = \frac{1}{N} \sum_{n=0}^{N-1} \dot{X}_1(n)\dot{X}_2^*(n). \quad (5.8)$$

При выводе формулы (5.8) были использованы соотношения (5.5) и (5.6).

Если, кроме того, последовательности  $\{x_1(k)\}$  и  $\{x_2(k)\}$  совпадают, т. е.  $x_1(k) = x_2(k) = x(k)$  для всех  $k = 0 \dots N - 1$ , из (5.8) получается дискретный аналог равенства Парсеваля (см. разд. "Энергетические расчеты в спектральной области" главы 1 и формулу (1.32)):

$$\sum_{k=0}^{N-1} x^2(k) = \frac{1}{N} \sum_{n=0}^{N-1} |\dot{X}(n)|^2.$$

## Круговая свертка

Так как мы рассматриваем периодические последовательности, то и суммирование при вычислении свертки таких последовательностей следует производить по одному периоду. Такую операцию называют *круговой сверткой*:

$$y(k) = \sum_{m=0}^{N-1} x_1(m)x_2(k-m) = \sum_{m=0}^{N-1} x_1(m)x_2((k-m) \bmod N) \quad (5.9)$$

### ЗАМЕЧАНИЕ

В этой формуле выражение  $(k - m) \bmod N$  означает взятие  $(k - m)$  по модулю  $N$ , т. е. вычисление остатка от деления  $(k - m)$  на  $N$ .

Подставив выражение (5.9) в (5.3), легко убедиться, что круговая свертка периодических временных последовательностей соответствует перемножению их спектров:

$$\dot{Y}(n) = \dot{X}_1(n)\dot{X}_2(n). \quad (5.10)$$

### ЗАМЕЧАНИЕ

*Круговую свертку* периодических последовательностей не следует путать с *линейной сверткой* (4.3), являющейся основой алгоритма дискретной фильтрации.

## Восстановление непрерывного сигнала с помощью ДПФ

Являясь по своей сути спектром *дискретного* периодического сигнала, ДПФ позволяет легко восстановить *непрерывный* периодический сигнал, занимающий ограни-

ченную полосу частот. Для этого в формуле обратного ДПФ (5.4) необходимо заменить дискретный параметр (номер отсчета  $k$ ) на непрерывный — нормированное время  $t/T$ , где  $T$  — период дискретизации:

$$x(t) = \frac{1}{N} \sum_{n=-N/2}^{N/2-1} \dot{X}(n) \exp\left(j \frac{2\pi n t}{NT}\right). \quad (5.11)$$

Следует обратить внимание на еще одно отличие этого соотношения от формулы (5.4): диапазон индексов суммирования смещен вниз на  $N/2$  (при четном  $N$ ; при нечетном  $N$  суммирование производится от  $n = -(N-1)/2$  до  $(N-1)/2$ ). Это необходимо, чтобы получить аналоговый сигнал, занимающий полосу частот от 0 до  $\pi/T$ . Коэффициенты  $\dot{X}(n)$  с отрицательными номерами могут быть получены из соотношения симметрии (5.5).

Результат восстановления непрерывного периодического сигнала с помощью ДПФ, разумеется, совпадает с результатами, получаемыми при использовании ряда Котельникова (3.12). Однако использование ДПФ в данном случае оказывается более предпочтительным, т. к. ряд Котельникова для периодического сигнала содержит бесконечное число слагаемых, а формула (5.11) — конечное.

## Матрица ДПФ

ДПФ является линейным преобразованием, трансформирующим вектор временных отсчетов в вектор такой же длины, содержащий отсчеты спектральные. Такое преобразование может быть реализовано [5] как умножение некоторой квадратной матрицы на входной вектор-столбец:

$$\mathbf{y} = \mathbf{A} \mathbf{x}, \quad (5.12)$$

где  $\mathbf{A}$  — матрица преобразования. В случае ДПФ эта матрица имеет вид

$$\mathbf{A}_{\text{DFT}} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-j\frac{2\pi}{N}} & e^{-j\frac{4\pi}{N}} & e^{-j\frac{6\pi}{N}} & \dots & e^{-j\frac{2\pi}{N}(N-1)} \\ 1 & e^{-j\frac{4\pi}{N}} & e^{-j\frac{8\pi}{N}} & e^{-j\frac{12\pi}{N}} & \dots & e^{-j\frac{2\pi}{N}2(N-1)} \\ 1 & e^{-j\frac{6\pi}{N}} & e^{-j\frac{12\pi}{N}} & e^{-j\frac{18\pi}{N}} & \dots & e^{-j\frac{2\pi}{N}3(N-1)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & e^{-j\frac{2\pi}{N}(N-1)} & e^{-j\frac{2\pi}{N}2(N-1)} & e^{-j\frac{2\pi}{N}3(N-1)} & \dots & e^{-j\frac{2\pi}{N}(N-1)^2} \end{bmatrix}. \quad (5.13)$$

Общая формула для элемента матрицы, расположенного в  $n$ -м столбце  $m$ -й строки, выглядит так:

$$A_{\text{DFT}}(m, n) = \exp\left(-j2\pi \frac{(m-1)(n-1)}{N}\right), \quad 1 \leq m \leq N, \quad 1 \leq n \leq N.$$

Вычисление ДПФ путем умножения матрицы на вектор полностью соответствует формуле (5.3). Этот метод требует большого количества вычислительных опера-

ций, поэтому на практике вместо него применяются быстрые алгоритмы, рассматриваемые далее.

В MATLAB матрица ДПФ вычисляется с помощью функции `dftmtx`.

## Связь ДПФ и спектра дискретного сигнала

Имея один и тот же конечный набор чисел, можно рассчитать либо *спектральную функцию* этого дискретного сигнала по формуле (3.2), либо его ДПФ по формуле (5.3). Разумеется, возникает вопрос о том, как связаны друг с другом эти два спектральных представления, полученные на основе одних и тех же отсчетов сигнала.

Сравнение формул (3.2) и (5.3) показывает, что ДПФ представляет собой просто *дискретные отсчеты* спектральной функции дискретного сигнала, соответствующие частотам  $\omega_n = \omega_d n / N$ :

$$\dot{X}(n) = \dot{S}\left(\frac{2\pi n}{NT}\right) = \dot{S}\left(\omega_d \frac{n}{N}\right). \quad (5.14)$$

По этой причине значения ДПФ иногда называют *спектральными отсчетами*.

Формула (5.14) определяет *частотную шкалу* ДПФ: спектральным отсчетам с номерами  $0, \dots, N-1$  соответствует сетка частот с шагом  $f_d/N$ , простирающаяся от нуля *почти* до частоты дискретизации (рис. 5.1).

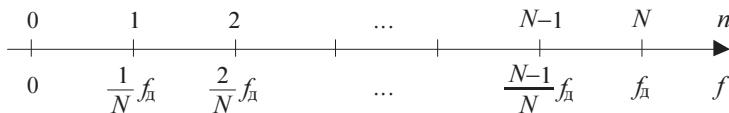


Рис. 5.1. Частотная шкала ДПФ

Из соотношения (5.14) следует еще один важный вывод: если добавить к конечному набору отсчетов некоторое количество нулей, спектральная функция дискретного сигнала, естественно, не изменится, но ДПФ даст большее число спектральных отсчетов, соответствующих частотам, более тесно расположенным в интервале от нуля до частоты дискретизации.

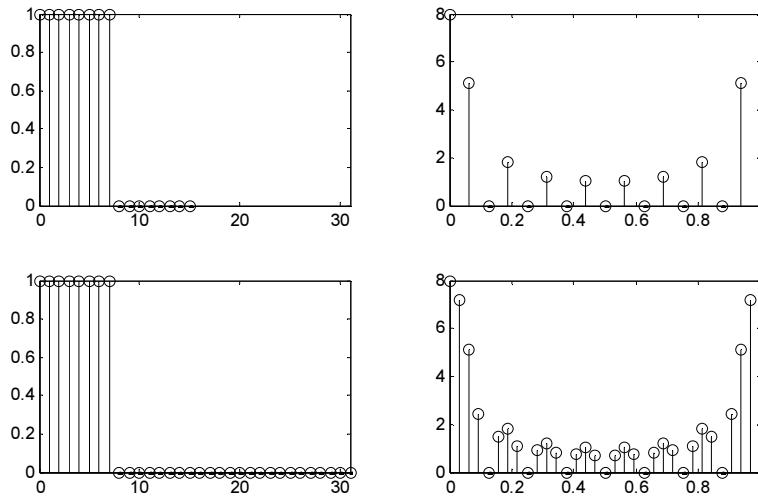
Поясним сказанное на простом примере, вычислив ДПФ для отсчетов прямоугольного импульса при разном количестве концевых нулей (рис. 5.2):

```
>> x1 = [ones(8,1); zeros(8,1)]; % 16 отсчетов
>> y1 = fft(x1); % ДПФ сигнала x1
>> x2 = [x1; zeros(16,1)]; % добавляем 16 нулей
>> y2 = fft(x2); % ДПФ сигнала x2
>> subplot(2, 2, 1)
>> stem(0:15, x1) % график сигнала x1
>> xlim([0 31])
>> subplot(2, 2, 2)
>> stem((0:15)/16, abs(y1)) % модуль ДПФ сигнала x1
```

```

>> subplot(2, 2, 3)
>> stem(0:31, x2) % график сигнала x2
>> xlim([0 31])
>> subplot(2, 2, 4)
>> stem((0:31)/32, abs(y2)) % модуль ДПФ сигнала x2

```



**Рис. 5.2.** Повышение спектрального разрешения ДПФ при дополнении сигнала нулями: сверху — исходный сигнал и модуль его ДПФ, снизу — сигнал, дополненный 16 нулями, и модуль его ДПФ

Горизонтальные оси на спектральных графиках проградуированы в долях частоты дискретизации. Из рис. 5.2 видно, что после увеличения длины сигнала вдвое за счет добавления нулевых отсчетов результат ДПФ стал содержать вдвое больше значений, соответствующих в два раза чаще расположенным частотам. Таким образом, дополнение сигнала нулями позволяет более подробно рассмотреть структуру спектра непериодической последовательности отсчетов, но не увеличивает реальное *спектральное разрешение*.

## Алгоритм быстрого преобразования Фурье

Для вычисления одного коэффициента ДПФ по формуле (5.3) необходимо выполнить  $N$  комплексных умножений и сложений. Таким образом, расчет всего ДПФ, содержащего  $N$  коэффициентов, потребует  $N^2$  пар операций "умножение-сложение" (то же самое можно увидеть и из формулы (5.12), представляющей ДПФ как умножение матрицы на вектор). Число операций возрастает пропорционально квадрату размерности ДПФ. Однако, если  $N$  не является простым числом и может быть разложено на множители, процесс вычислений можно ускорить, разделив анализируемый набор отсчетов на части, вычислив их ДПФ и объединив результаты. Такие способы вычисления ДПФ называются *быстрым преобразованием Фурье*.

(БПФ; английский термин — *Fast Fourier Transform, FFT*) и повсеместно используются на практике.

При реализации БПФ возможно несколько вариантов организации вычислений в зависимости от способа деления последовательности отсчетов на части (*прореживание по времени либо по частоте*) и от того, на сколько фрагментов производится разбиение последовательности на каждом шаге (*основание БПФ*).

## БПФ с прореживанием по времени

Рассмотрим идею БПФ с *прореживанием по времени* (*decimation in time, DIT*) на примере деления набора отсчетов пополам.

Итак, пусть  $N$  — четное число. Выделим в формуле (5.3) два слагаемых, соответствующих элементам исходной последовательности с четными и нечетными номерами:

$$\dot{X}(n) = \sum_{m=0}^{N/2-1} x(2m)e^{-j\frac{2\pi mn}{N}} + \sum_{m=0}^{N/2-1} x(2m+1)e^{-j\frac{2\pi(2m+1)n}{N}}.$$

Введем обозначения  $y(m) = x(2m)$  и  $z(m) = x(2m+1)$ , а также вынесем из второй суммы общий множитель  $e^{-j2\pi n/N}$ :

$$\dot{X}(n) = \sum_{m=0}^{N/2-1} y(m)e^{-j\frac{2\pi mn}{N/2}} + e^{-j\frac{2\pi n}{N}} \sum_{m=0}^{N/2-1} z(m)e^{-j\frac{2\pi mn}{N/2}}. \quad (5.15)$$

Две суммы в (5.15) представляют собой ДПФ последовательностей  $\{y(m)\}$  (отсчеты с четными номерами) и  $\{z(m)\}$  (отсчеты с нечетными номерами). Каждое из этих ДПФ имеет размерность  $N/2$ . Таким образом,

$$\dot{X}(n) = \dot{Y}(n) + e^{-j\frac{2\pi n}{N}} \dot{Z}(n), \quad (5.16)$$

где  $\dot{Y}(n)$  и  $\dot{Z}(n)$  — ДПФ, соответственно, последовательностей отсчетов с четными и нечетными номерами:

$$\dot{Y}(n) = \sum_{m=0}^{N/2-1} y(m)e^{-j\frac{2\pi mn}{N/2}},$$

$$\dot{Z}(n) = \sum_{m=0}^{N/2-1} z(m)e^{-j\frac{2\pi mn}{N/2}}.$$

Так как ДПФ размерности  $N/2$  дает лишь  $N/2$  спектральных коэффициентов, непосредственно использовать формулу (5.16) можно только при  $0 \leq n < N/2$ . Для остальных  $n$  ( $N/2 \leq n < N$ ) следует воспользоваться периодичностью спектра дискретного сигнала (и, соответственно, периодичностью результатов ДПФ):

$$\dot{Y}\left(n + \frac{N}{2}\right) = \dot{Y}(n), \quad \dot{Z}\left(n + \frac{N}{2}\right) = \dot{Z}(n).$$

С учетом этого при  $n \geq N/2$  формула (5.16) представляется в виде

$$\begin{aligned}\dot{X}(n) &= \dot{Y}\left(n - \frac{N}{2}\right) + e^{-j\frac{2\pi n}{N}} \dot{Z}\left(n - \frac{N}{2}\right) = \\ &= \dot{Y}\left(n - \frac{N}{2}\right) - e^{-j\frac{2\pi}{N}\left(n - \frac{N}{2}\right)} \dot{Z}\left(n - \frac{N}{2}\right).\end{aligned}\quad (5.17)$$

Процесс вычисления 8-точечного ДПФ путем разбиения его на два 4-точечных ДПФ иллюстрируется на рис. 5.3. На этом рисунке использовано следующее обозначение для комплексных экспонент:

$$\dot{w}_N^n = \exp\left(-j \frac{2\pi n}{N}\right).$$

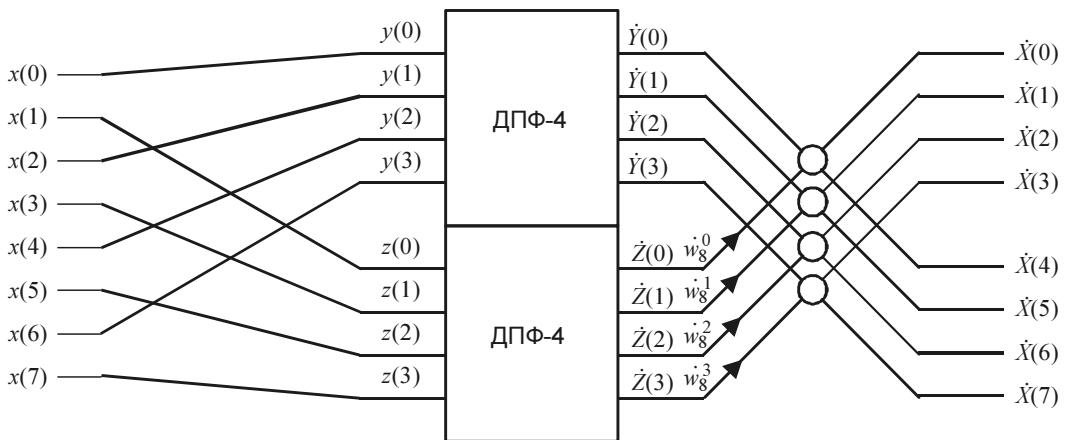
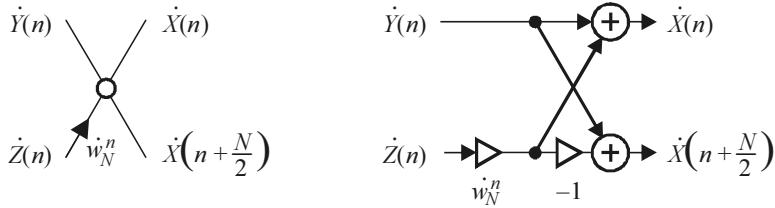


Рис. 5.3. Вычисление 8-точечного ДПФ с помощью двух 4-точечных ДПФ

Блоки, выполняющие на рис. 5.3 объединение результатов двух ДПФ, требуют дополнительных комментариев. Каждый такой блок имеет два входных и два выходных сигнала. Один из входных сигналов умножается на комплексную экспоненту  $\dot{w}_N^n$ , после чего суммируется со вторым входным сигналом и вычитается из него, формируя таким образом два выходных сигнала. Это соответствует реализации формул (5.16) и (5.17). Данная операция получила название "бабочки" (*butterfly*). Расшифровка ее структуры представлена на рис. 5.4.

Оценим количество операций, необходимое для вычисления ДПФ указанным способом. Каждое из двух ДПФ половинной размерности требует  $N^2/4$  операций. Кроме того, при вычислении окончательных результатов каждый спектральный коэффициент  $\dot{Z}(n)$  умножается на экспоненциальный комплексный множитель.

Это добавляет еще  $N/2$  операций. Итого получается  $2 \frac{N^2}{4} + \frac{N}{2} = \frac{N(N+1)}{2}$ , что почти вдвое меньше, чем при вычислении ДПФ прямым способом.



**Рис. 5.4.** Условное обозначение "бабочки" БПФ с прореживанием по времени (слева) и ее структурная схема (справа)

Если  $N/2$  тоже является четным числом (т. е. если  $N$  делится на 4), можно продолжить описанную процедуру, выразив результат через четыре ДПФ размерности  $N/4$ . Это позволяет еще больше сократить число требуемых вычислительных операций.

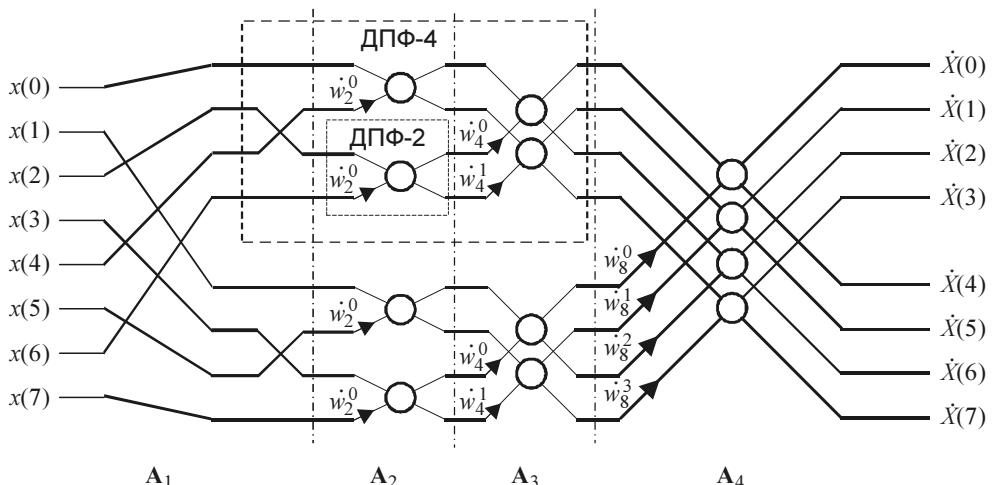
### ЗАМЕЧАНИЕ

Делить исходную последовательность можно на любое количество частей. Таким образом, приведенный алгоритм позволяет уменьшить число операций в случае любого  $N$ , не являющегося простым числом. Степень ускорения вычислений зависит от числа фрагментов последовательности и является максимальной при делении на две части, как в рассмотренном примере.

Наибольшая степень ускорения вычислений может быть достигнута при  $N = 2^k$ , в этом случае деление последовательностей на две части можно продолжать до тех пор, пока не получатся двухэлементные последовательности, ДПФ которых рассчитывается вообще без использования операций умножения (достаточно вычислить сумму и разность двух отсчетов).

### Полная схема 8-точечного БПФ

Если в схеме на рис. 5.3 блоки 4-точечного ДПФ представить как комбинацию двух двухточечных ДПФ и учесть, что двухточечное ДПФ сводится к единственной "бабочке", получится схема, показанная на рис. 5.5.



**Рис. 5.5.** Полная схема 8-точечного БПФ с прореживанием по времени

Используя приведенную схему, оценим количество арифметических операций, необходимых для вычисления БПФ с размерностью, равной степени двойки. Выделим в приведенной структуре отдельные каскады преобразования (на рис. 5.5 они отделены друг от друга вертикальными штрихпунктирными линиями). Размерность ДПФ в смежных каскадах отличается в два раза, поэтому общее число каскадов составляет  $\log_2(N)$  (самый первый каскад осуществляет только перестановку отсчетов и не требует арифметических операций). Каждый каскад содержит  $N/2$  "бабочек", для реализации каждой "бабочки" необходимо два комплексных сложения и одно комплексное умножение (см. рис. 5.4). Считая затраты на одну "бабочку" равными двум парам комплексных операций "умножение-сложение", получаем общее число таких операций, равное

$$\log_2(N) \cdot \frac{N}{2} \cdot 2 = N \log_2(N).$$

Таким образом, вычислительные затраты по сравнению с непосредственным использованием формулы (5.3) уменьшаются в  $N/\log_2(N)$  раз. При больших  $N$  это отношение становится весьма велико (например,  $1024/\log_2(1024) = 102,4$ , т. е. при  $N = 1024$  достигается более чем стократное ускорение).

## Математическая основа БПФ

Ранее уже было сказано, что приведенную на рис. 5.5 структуру можно представить как каскадное выполнение нескольких элементарных операций. Каждый каскад реализует некоторое линейное преобразование последовательности чисел; математически оно может быть представлено как умножение вектора-столбца слева на матрицу преобразования. Обозначения этих матриц указаны на рис. 5.5 снизу, под соответствующими каскадами структуры.

В результате последовательного выполнения указанных операций общая матрица ДПФ (см. приведенную ранее формулу (5.13)) формируется как произведение матриц, соответствующих отдельным каскадам:

$$\mathbf{A}_{\text{DFT}} = \mathbf{A}_4 \mathbf{A}_3 \mathbf{A}_2 \mathbf{A}_1.$$

Рассмотрим структуру матриц  $\mathbf{A}_1, \dots, \mathbf{A}_4$  подробнее. Первый каскад — это просто перестановка входных отсчетов в другом порядке. Матрица соответствующего преобразования содержит 8 единиц в позициях, определяемых правилом перестановки (чтобы не загромождать формулы, нулевые элементы матриц в формулах этого раздела не показаны):

$$\mathbf{A}_1 = \begin{bmatrix} 1 & & & & & & & \\ - & & & & & & & \\ - & + & + & - & - & + & + & \\ & 1 & & & & & & \\ - & - & + & + & - & - & + & \\ & & 1 & & & & & \\ - & + & - & + & + & - & - & \\ & & & 1 & & & & \\ - & - & + & - & - & + & + & \\ & & & & 1 & & & \\ - & + & + & - & - & - & + & \\ & & & & & 1 & & \\ - & - & - & + & + & + & - & \\ & & & & & & 1 & \end{bmatrix}.$$

Следующие три каскада однотипны, каждый из них реализует четыре "бабочки", показанные на рис. 5.4, отличия состоят только в том, какие пары входных/выходных отсчетов используются для "бабочек" и чему равны множители  $\dot{w}_N^n$ :

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{A}_3 = \begin{bmatrix} 1 & 1 & 1 & \dot{w}_4^1 \\ 1 & 1 & -\dot{w}_4^1 & \\ 1 & -1 & 1 & \\ 1 & -1 & -\dot{w}_4^1 & \\ 1 & 1 & 1 & \\ 1 & 1 & -\dot{w}_4^1 & \\ 1 & -1 & 1 & \\ 1 & -1 & -\dot{w}_4^1 & \end{bmatrix},$$

$$\mathbf{A}_4 = \begin{bmatrix} 1 & 1 & 1 & \dot{w}_8^1 \\ 1 & 1 & \dot{w}_8^2 & \\ 1 & 1 & \dot{w}_8^3 & \\ 1 & -1 & 1 & \\ 1 & -1 & -\dot{w}_8^1 & \\ 1 & 1 & -\dot{w}_8^2 & \\ 1 & 1 & -\dot{w}_8^3 & \\ 1 & -1 & 1 & \\ 1 & -1 & -\dot{w}_8^1 & \end{bmatrix}.$$

Приведенные матрицы показывают следующее: при реализации БПФ производится факторизация матрицы преобразования, в результате чего она оказывается представленной в виде произведения *разреженных матриц*, лишь малая доля элементов которых отлична от нуля. За счет этого и происходит ускорение вычислений.

## Бит-реверсная адресация

Как уже было сказано, первый каскад схемы рис. 5.5, описываемый матрицей  $\mathbf{A}_1$ , осуществляет лишь перестановку элементов вектора входных отсчетов. Рассмотрим закон этой перестановки подробнее, для чего составим применительно к данному примеру таблицу соответствия номеров (табл. 5.1).

Сравнение столбцов двоичных представлений в табл. 5.1 показывает, что правило перестановки получается путем зеркальной перестановки битов в номерах отсчетов. Это называется *бит-реверсной перестановкой* (*bit-reversed order*), такой способ адресации элементов массивов поддерживается в цифровых сигнальных процессорах на аппаратном уровне — специально для удобства реализации алгоритмов БПФ.

В MATLAB перестановка массива в бит-реверсном порядке реализуется с помощью функции `bitrevorder` (в более общем случае произвольной системы счисления — `digitrevorder`).

Таблица 5.1. Правило перестановки отсчетов при 8-точечном БПФ

Номер отсчета на входе		Номер отсчета на выходе	
Десятичный	Двоичный	Десятичный	Двоичный
0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111

## БПФ с прореживанием по частоте

Формулы прямого и обратного ДПФ (5.3) и (5.4) отличаются только знаком в показателе экспоненты и множителем перед суммой. Поэтому можно получить еще один вариант алгоритма БПФ, выполнив преобразования, показанные на схеме рис. 5.3, в обратном порядке. Этот способ вычислений называется *прореживанием по частоте* (*decimation in frequency*, DIF). Покажем, как получить описание этого метода на основе формулы прямого ДПФ (5.3).

Разделим исходную последовательность  $\{x(k)\}$  на две следующие друг за другом половины (как и в предыдущем случае,  $N$  должно быть четным числом):

$$\dot{X}(n) = \sum_{m=0}^{N/2-1} x(m)e^{-j\frac{2\pi mn}{N}} + \sum_{m=0}^{N/2-1} x(m + \frac{N}{2})e^{-j\frac{2\pi(m+N/2)n}{N}}.$$

Из второй суммы можно выделить множитель  $e^{-j\frac{2\pi(N/2)n}{N}} = e^{-j\pi n} = (-1)^n$ . Этот множитель равен 1 или  $-1$  в зависимости от четности номера вычисляемого спектрального отсчета  $n$ , поэтому дальше рассматриваем четные и нечетные  $n$  по отдельности. После выделения множителя  $\pm 1$  комплексные экспоненты в обеих суммах становятся одинаковыми, поэтому выносим их за скобки, объединяя две суммы:

$$\dot{X}(2k) = \sum_{m=0}^{N/2-1} \left( x(m) + x(m + \frac{N}{2}) \right) e^{-j\frac{2\pi mk}{N/2}},$$

$$\dot{X}(2k+1) = \sum_{m=0}^{N/2-1} \left( x(m) - x(m + \frac{N}{2}) \right) e^{-j\frac{2\pi mk}{N/2}} e^{-j\frac{2\pi m}{N}}.$$

Фигурирующие здесь суммы представляют собой ДПФ суммы и разности половин исходной последовательности, при этом разность перед вычислением ДПФ умно-

жается на комплексные экспоненты  $\exp(-j2\pi m/N)$ . Каждое из двух используемых здесь ДПФ имеет размерность  $N/2$ .

Итак, при прореживании по частоте вычисления организуются следующим образом:

1. Из исходной последовательности  $\{x(k)\}$  длиной  $N$  получаются две последовательности  $\{y(m)\}$  и  $\{z(m)\}$  длиной  $N/2$  согласно следующим формулам:

$$y(m) = x(m) + x(m + \frac{N}{2}),$$

$$z(m) = \left( x(m) - x(m + \frac{N}{2}) \right) e^{-j\frac{2\pi m}{N}}.$$

2. ДПФ последовательности  $\{y(m)\}$  дает спектральные отсчеты с четными номерами, ДПФ последовательности  $\{z(m)\}$  — с нечетными:

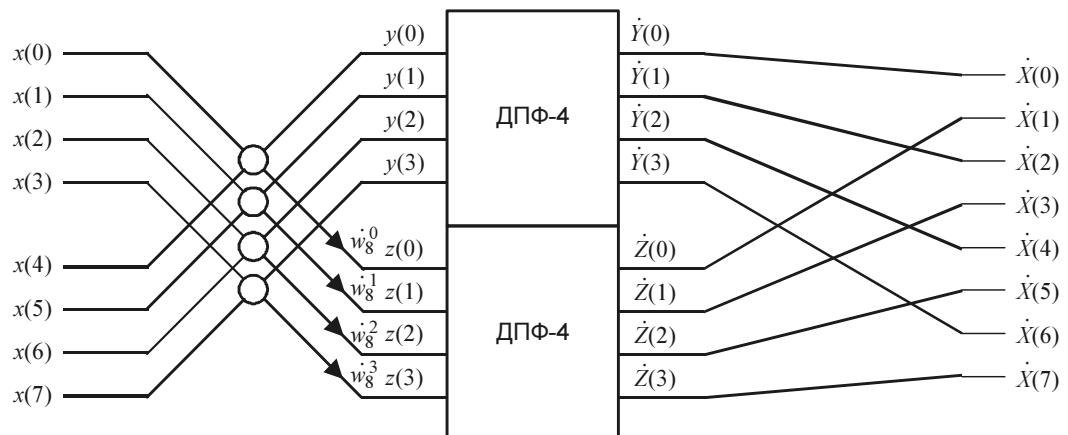
$$\dot{X}(2k) = \dot{Y}(k) = \sum_{m=0}^{N/2-1} y(m) e^{-j\frac{2\pi km}{N/2}},$$

$$\dot{X}(2k+1) = \dot{Z}(k) = \sum_{m=0}^{N/2-1} z(m) e^{-j\frac{2\pi km}{N/2}}.$$

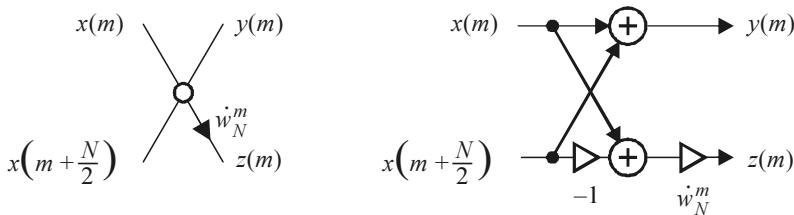
Все сказанное в предыдущем разделе о возможности деления последовательности на иное, отличное от двух, число частей и об уменьшении числа операций, требуемых для расчетов, относится и к алгоритму с прореживанием по частоте.

Процесс вычисления 8-точечного ДПФ путем разбиения его на два 4-точечных ДПФ с прореживанием по частоте показан на рис. 5.6.

Поскольку комплексный экспоненциальный множитель в данном алгоритме применяется к результату вычитания двух сигналов, "бабочка" БПФ с прореживанием по частоте имеет несколько иную структурную схему (рис. 5.7).



**Рис. 5.6.** Вычисление 8-точечного ДПФ с помощью двух 4-точечных ДПФ путем прореживания по частоте



**Рис. 5.7.** Условное обозначение "бабочки" БПФ с прореживанием по частоте (слева) и ее структурная схема (справа)

### ЗАМЕЧАНИЕ 1

Для получения алгоритма *обратного БПФ* достаточно поменять в приведенных формулах знак в показателях комплексных экспонент и добавить на выходе (или на входе) деление на два (в более общем случае — на используемый коэффициент прореживания).

### ЗАМЕЧАНИЕ 2

Возможен также обобщенный подход к рассмотрению алгоритмов БПФ с прореживанием по времени и по частоте. Его описание можно найти в [7, 8].

## Основание алгоритма БПФ

В названиях алгоритмов БПФ можно встретить слово "RADIX" ("основание" — в математическом смысле). Следующее после него число обозначает число фрагментов, на которое разбивается сигнал на каждом этапе прореживания (а также минимальный размер "кусочков" входного вектора, который достигается в результате его последовательных разбиений).

В алгоритмах "RADIX-2" размер анализируемой последовательности должен быть равен степени двойки, а ее половинное деление производится вплоть до получения двухэлементных последовательностей. Вычисление их ДПФ не требует операций умножения — два спектральных отсчета представляют собой сумму и разность отсчетов временных:

$$\dot{X}(0) = x(0) + x(1),$$

$$\dot{X}(1) = x(0) - x(1).$$

В алгоритмах "RADIX-4" количество отсчетов сигнала должно быть равно степени четверки, при каждом прореживании сигнал делится на четыре фрагмента, а последней стадией деления являются четырехэлементные последовательности. При вычислении их ДПФ умножение производится только на  $\pm j$ , а такое умножение сводится к взаимной перестановке вещественной и мнимой частей комплексного числа с изменением знака у одной из них:

$$\dot{X}(0) = x(0) + x(1) + x(2) + x(3),$$

$$\dot{X}(1) = x(0) - j x(1) - x(2) + j x(3),$$

$$\dot{X}(2) = x(0) - x(1) + x(2) - x(3),$$

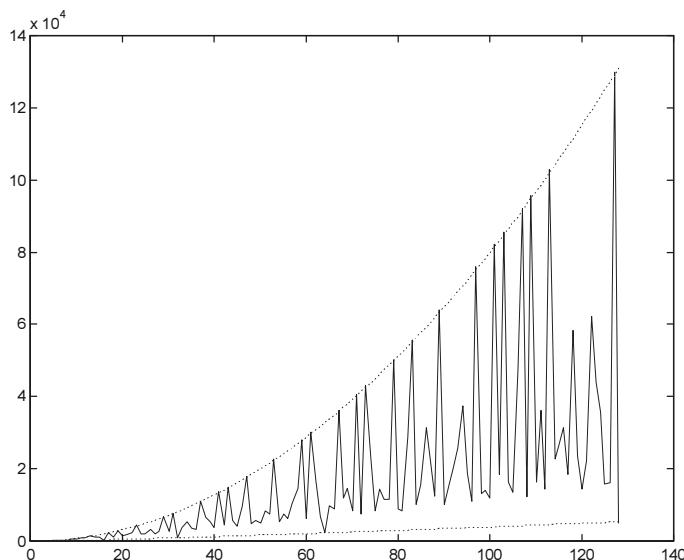
$$\dot{X}(3) = x(0) + j x(1) - x(2) - j x(3).$$

В [8] показано, что использование основания 4 позволяет ощутимо уменьшить число выполняемых умножений.

## Вычислительные затраты при произвольной размерности БПФ

Как уже говорилось, некоторое ускорение вычислений за счет применения быстрых алгоритмов возможно, если размерность преобразования не является простым числом. Проиллюстрируем этот факт количественно.

На рис. 5.8 показан график зависимости числа вычислительных операций от размерности БПФ. Этот график получен с помощью существовавшей в старых версиях MATLAB (до 5.x включительно) функции `flops`, подсчитывавшей число выполненных арифметических операций.



**Рис. 5.8.** Зависимость числа арифметических операций от размерности БПФ

Кроме собственно зависимости числа операций от размерности БПФ на графике показаны также верхняя и нижняя границы. Верхняя линия — это число операций, соответствующее прямой формуле ДПФ и равное  $8N^2$  (множитель 8 возникает из-за того, что на графике показано количество вещественных операций). Нижняя линия — число операций, соответствующее максимальному ускорению вычислений и равное  $6N \log_2(N)$  (множитель 6 также возникает из-за подсчета вещественных операций). Хорошо видно, что график числа операций при БПФ касается верхней

границы, если  $N$  — простое число, и касается нижней границы, если  $N$  — степень двойки.

## Выводы

Наибольшее ускорение вычислений благодаря алгоритму БПФ достигается при длине анализируемого вектора, равной степени двойки. При разложении длины вектора на иные множители ускорение также возможно, хотя и не столь значительное. Если длина вектора — простое число, вычисление спектра может быть выполнено только по прямой формуле ДПФ.

Пример, демонстрирующий зависимость числа вычислительных операций от размерности БПФ, будет приведен далее, при описании функции `fft` в разд. "Функции спектрального анализа в MATLAB" этой главы.

Завершая краткое рассмотрение идеи БПФ, необходимо отметить следующее:

- БПФ не является приближенным алгоритмом; при отсутствии вычислительных погрешностей он даст в точности тот же результат, что и исходная формула ДПФ (5.3). Ускорение достигается исключительно за счет оптимальной организации вычислений;
- применение БПФ имеет смысл, если число элементов в анализируемой последовательности является степенью числа 2. Как уже отмечалось, некоторое ускорение вычислений возможно и при разложении  $N$  на другие множители, однако это ускорение не столь велико, как при  $N = 2^k$ ;
- алгоритм БПФ предназначен для одновременного расчета *всех* спектральных отсчетов  $\hat{X}(n)$ . Если же необходимо получить эти отсчеты лишь для некоторых  $n$ , может оказаться предпочтительнее прямая формула ДПФ или рассматриваемый далее алгоритм Герцеля.

## Взаимосвязь ДПФ и фильтрации

К данному моменту мы уже познакомились с понятиями дискретной фильтрации и дискретного преобразования Фурье, однако пока что эти понятия обсуждались по отдельности. В данном разделе рассматривается, как они связаны друг с другом, причем связь эта оказывается двусторонней — ДПФ можно представить как обработку сигнала набором фильтров, а дискретную фильтрацию можно организовать с помощью ДПФ, что позволяет значительно уменьшить число вычислительных операций за счет использования алгоритмов БПФ.

## ДПФ как дискретная фильтрация

Рассмотрев принципы дискретной фильтрации и познакомившись с дискретным преобразованием Фурье, можно заметить, что формулы, описывающие эти два процессы, весьма схожи — в обоих случаях они представляют собой линейную комбинацию отсчетов входного сигнала. Это говорит о том, что ДПФ можно трак-

товать как обработку сигнала фильтром с соответствующей импульсной характеристикой.

Эту импульсную характеристику можно получить, если заметить, что  $e^{j2\pi n} = 1$  при целочисленном  $n$ , и с учетом этого записать формулу прямого ДПФ (5.3) в виде

$$\dot{X}(n) = \sum_{k=0}^{N-1} x(k) \exp\left(j2\pi \frac{n}{N}(N-k)\right).$$

Преобразованная таким способом формула ДПФ представляет собой дискретную свертку (4.3), т. е.  $N$ -й отсчет результата обработки входного сигнала  $x(k)$  фильтром, импульсная характеристика которого равна

$$h_n(k) = \exp\left(j2\pi \frac{n}{N} k\right), \quad k = 0, 1, \dots, N-1.$$

Разумеется, импульсная характеристика для каждого частотного отсчета ДПФ своя; чтобы подчеркнуть это, в ее обозначении использован индекс  $n$ .

Определим частотную характеристику такого фильтра. Для этого, как было показано в разд. "Способы описания дискретных систем" главы 4, сначала необходимо получить функцию передачи:

$$\begin{aligned} H_n(z) &= 1 + \exp\left(j2\pi \frac{n}{N}\right) z^{-1} + \exp\left(j2\pi \frac{2n}{N}\right) z^{-2} + \dots + \exp\left(j2\pi \frac{(N-1)n}{N}\right) z^{-N+1} = \\ &= \frac{\exp\left(j2\pi \frac{Nn}{N}\right) z^{-N} - 1}{\exp\left(j2\pi \frac{n}{N}\right) z^{-1} - 1} = \frac{1 - z^{-N}}{1 - \exp\left(j2\pi \frac{n}{N}\right) z^{-1}}. \end{aligned} \quad (5.18)$$

### ЗАМЕЧАНИЕ

Для расчета функции передачи использована формула суммы конечной геометрической прогрессии:

$$\sum_{k=0}^N b_0 q^k = b_0 \frac{q^{N+1} - 1}{q - 1}.$$

Для расчета частотной характеристики используем подстановку  $z = e^{j\omega T}$ :

$$\dot{K}_n(\omega) = H_n(e^{j\omega T}) = \frac{1 - e^{-j\omega NT}}{1 - e^{j2\pi \frac{n}{N}} e^{-j\omega T}}.$$

АЧХ такого фильтра после несложных тригонометрических преобразований можно записать следующим образом:

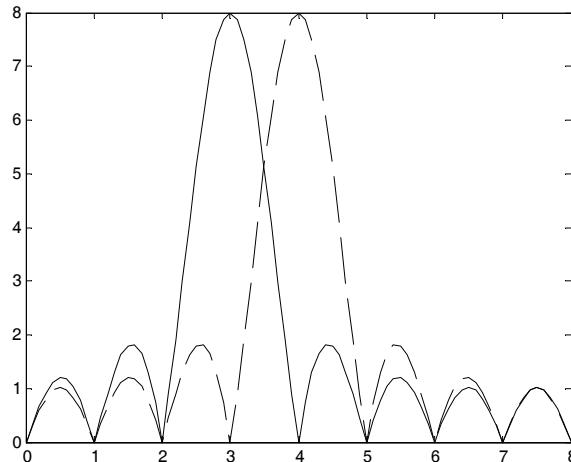
$$K_n(\omega) = |\dot{K}_n(\omega)| = \frac{\left| \sin \frac{\omega NT}{2} \right|}{\left| \sin \frac{1}{2} \left( \omega T - \frac{2\pi n}{N} \right) \right|} = \frac{\left| \sin \left( \pi N \frac{\omega - \omega_n}{\omega_d} \right) \right|}{\left| \sin \left( \pi \frac{\omega - \omega_n}{\omega_d} \right) \right|}, \text{ где } \omega_n = \frac{n}{N} \omega_d. \quad (5.19)$$

Результат показывает, что АЧХ фильтра описывается рассмотренной в разд. "Функции генерации периодических сигналов" главы 3 функцией Дирихле:

$$K(\omega) = N \left| \text{diric}_N \left( 2\pi \frac{\omega - \omega_n}{\omega_d} \right) \right|.$$

На рис. 5.9 показан график АЧХ одного из каналов ДПФ при  $N=8$  (градуировка частотной оси выполнена в номерах каналов). Пунктирной линией на этом рисунке изображена АЧХ соседнего частотного канала:

```
>> N = 8;
>> w = 0:0.1:N;
>> plot(w, N*abs(diric((w-3)/N*2*pi, N)))
>> hold on
>> plot(w, N*abs(diric((w-4)/N*2*pi, N)), '--')
>> hold off
```



**Рис. 5.9.** АЧХ частотных каналов ДПФ

Как видно из графиков, АЧХ фильтра, реализуемого при вычислении ДПФ, имеет большой уровень боковых лепестков — примерно  $1/(N \sin(3\pi/(2N)))$ . При  $N \gg 1$  это выражение стремится к  $2/(3\pi) \approx 0,2 \approx -13,5$  дБ. Для уменьшения уровня боковых лепестков используют *весовые функции*, или *окна* (см. далее в этой главе).

## Алгоритм Герцеля

Функция передачи отдельного канала ДПФ, записанная в форме (5.18), соответствует рекурсивному фильтру  $N$ -го порядка. Поэтому вычисление отдельного спектрального отсчета можно реализовать с помощью такого фильтра. Если необходимо вычислить лишь некоторые отсчеты ДПФ, данный алгоритм может оказаться предпочтительнее, чем использование БПФ, принципиально ориентированного на получение *всех* спектральных отсчетов.

Однако для удобства практического использования формулу (5.18) можно модифицировать. Прежде всего заметим, что слагаемое  $-z^{-N}$  в числителе выполняет лишь роль "ограничителя", заставляя импульсную характеристику закончиться, достигнув длины  $N$  отсчетов. Такой фильтр вычисляет отсчет *мгновенного спектра* сигнала, все время используя *последние  $N$  отсчетов* входного сигнала. Если же мы будем обрабатывать сигнал порциями по  $N$  отсчетов, перед каждой порцией обнуляя состояние фильтра, то конечность длины импульсной характеристики не будет иметь значения. В этом случае можно удалить слагаемое  $-z^{-N}$  из числителя функции передачи, получив таким образом рекурсивный фильтр первого порядка:

$$H_n(z) = \frac{1}{1 - \exp\left(j2\pi \frac{n}{N}\right)z^{-1}}.$$

Следующее изменение связано с тем, что коэффициент обратной связи в единственной рекурсивной ветви данного фильтра является комплексным. Поэтому при обработке каждого входного отсчета необходимо выполнить четыре вещественных умножения и столько же вещественных сложений. Число операций можно сократить, умножив числитель и знаменатель функции передачи на  $(1 - e^{-j2\pi n/N} z^{-1})$ , перейдя за счет этого к фильтру второго порядка:

$$H_n(z) = \frac{1 - \exp\left(-j2\pi \frac{n}{N}\right)z^{-1}}{\left(1 - \exp\left(j2\pi \frac{n}{N}\right)z^{-1}\right)\left(1 - \exp\left(-j2\pi \frac{n}{N}\right)z^{-1}\right)} = \frac{1 - \exp\left(-j2\pi \frac{n}{N}\right)z^{-1}}{1 - 2\cos\left(2\pi \frac{n}{N}\right) + z^{-2}}.$$

Рекурсивная часть получившегося фильтра имеет вещественные коэффициенты, к тому же коэффициент при  $z^{-2}$  равен единице. Поэтому при обработке каждого входного отсчета в рекурсивной ветви фильтра необходимо выполнить всего одно вещественное умножение и два вещественных сложения.

В рекурсивной ветви фильтра комплексные коэффициенты исчезли, но они, естественно, появились в числите функции передачи, т. е. в нерекурсивной части фильтра. Однако если реализовать фильтр в *канонической форме* (см. рис. 4.13), то расчеты будут выполняться сначала в рекурсивной ветви, а затем в нерекурсивной. Поскольку нас интересует только конечный результат вычисления спектрального отсчета, комплексное умножение и сложение в нерекурсивной ветви должны быть выполнены только один раз — для получения окончательного результата.

Полученная структура фильтра показана на рис. 5.10. Данный способ вычисления отдельного спектрального отсчета называется *алгоритмом Герцеля* (*Hoertzel*) [7].

Для вычисления спектрального отсчета по алгоритму Герцеля требуется  $N + 4$  вещественных умножений и  $2N + 3$  вещественных сложений. Если число вычисляемых спектральных отсчетов невелико, этот алгоритм оказывается эффективнее, чем расчет всех спектральных отсчетов с помощью БПФ.

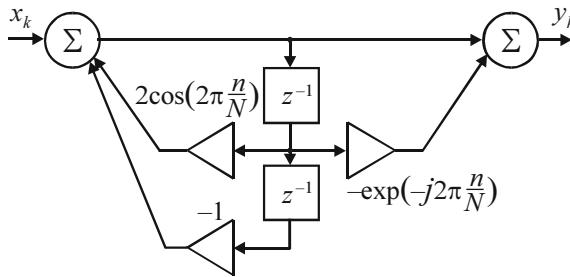


Рис. 5.10. Алгоритм Герцеля

## Дискретная фильтрация с помощью ДПФ

Ранее в этой главе мы представили ДПФ как дискретную фильтрацию. Теперь же займемся обратной задачей — реализацией фильтрации с помощью ДПФ.

Рассматривая свойства ДПФ, мы видели, что перемножение дискретных спектров соответствует *круговой свертке* периодических последовательностей, в то время как выходной сигнал дискретного фильтра представляет собой *линейную свертку* последовательностей конечной длины — входного сигнала и импульсной характеристики фильтра (см. разд. "Способы описания дискретных систем" главы 4). Однако с помощью ДПФ можно получить и линейную свертку, для этого необходимо добавить к каждой из сворачиваемых последовательностей нулевые отсчеты в количестве, не меньшем чем длина второй последовательности минус единица. Длины двух последовательностей после этого должны стать одинаковыми и равными как минимум длине линейной свертки исходных последовательностей.

Круговая свертка таких дополненных нулями последовательностей совпадает с линейной сверткой исходных последовательностей.

Поясним сказанное на простом примере. Пусть сворачиваемые последовательности состоят из четырех элементов:

$$x_1 = \{1, 2, 4, 8\} \text{ и } x_2 = \{2, 3, 4, 5\}.$$

Линейная свертка этих последовательностей рассчитывается согласно формуле (4.3). Процесс расчета показан на рис. 5.11, сверху.

Круговая свертка этих же последовательностей рассчитывается по формуле (5.9). Процесс расчета показан на рис. 5.11, снизу. Результат, как видите, совсем другой. Но если мы дополним каждую из последовательностей тремя нулями, чтобы их длины стали равны семи (это длина их линейной свертки), и вычислим круговую свертку еще раз, то получим результат линейной свертки, приведенный выше (рис. 5.12).

Поскольку ДПФ линейной свертки равно произведению ДПФ сворачиваемых последовательностей, мы получаем следующий алгоритм фильтрации в частотной области.

1. Последовательность отсчетов входного сигнала и импульсная характеристика фильтра дополняются нулями так, чтобы длины последовательностей стали

## Исходные сигналы

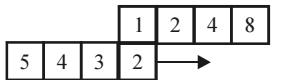
 $x_1:$ 

1	2	4	8
---	---	---	---

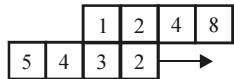
 $x_2:$ 

2	3	4	5
---	---	---	---

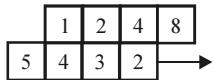
## Вычисление линейной свертки



$y(0) = 1 \cdot 2 = 2$



$y(1) = 1 \cdot 3 + 2 \cdot 2 = 7$



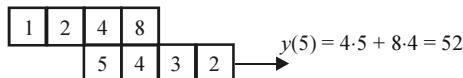
$y(2) = 1 \cdot 4 + 2 \cdot 3 + 4 \cdot 2 = 18$



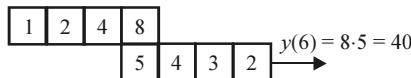
$y(3) = 1 \cdot 5 + 2 \cdot 4 + 4 \cdot 3 + 8 \cdot 2 = 41$



$y(4) = 2 \cdot 5 + 4 \cdot 4 + 8 \cdot 3 = 50$



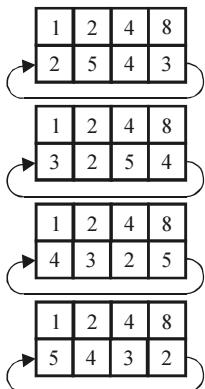
$y(5) = 4 \cdot 5 + 8 \cdot 4 = 52$



Результат

2	7	18	41	50	52	40
---	---	----	----	----	----	----

## Вычисление круговой свертки



$y(0) = 1 \cdot 2 + 2 \cdot 5 + 4 \cdot 4 + 8 \cdot 3 = 52$

$y(1) = 1 \cdot 3 + 2 \cdot 2 + 4 \cdot 5 + 8 \cdot 4 = 59$

$y(2) = 1 \cdot 4 + 2 \cdot 3 + 4 \cdot 2 + 8 \cdot 5 = 58$

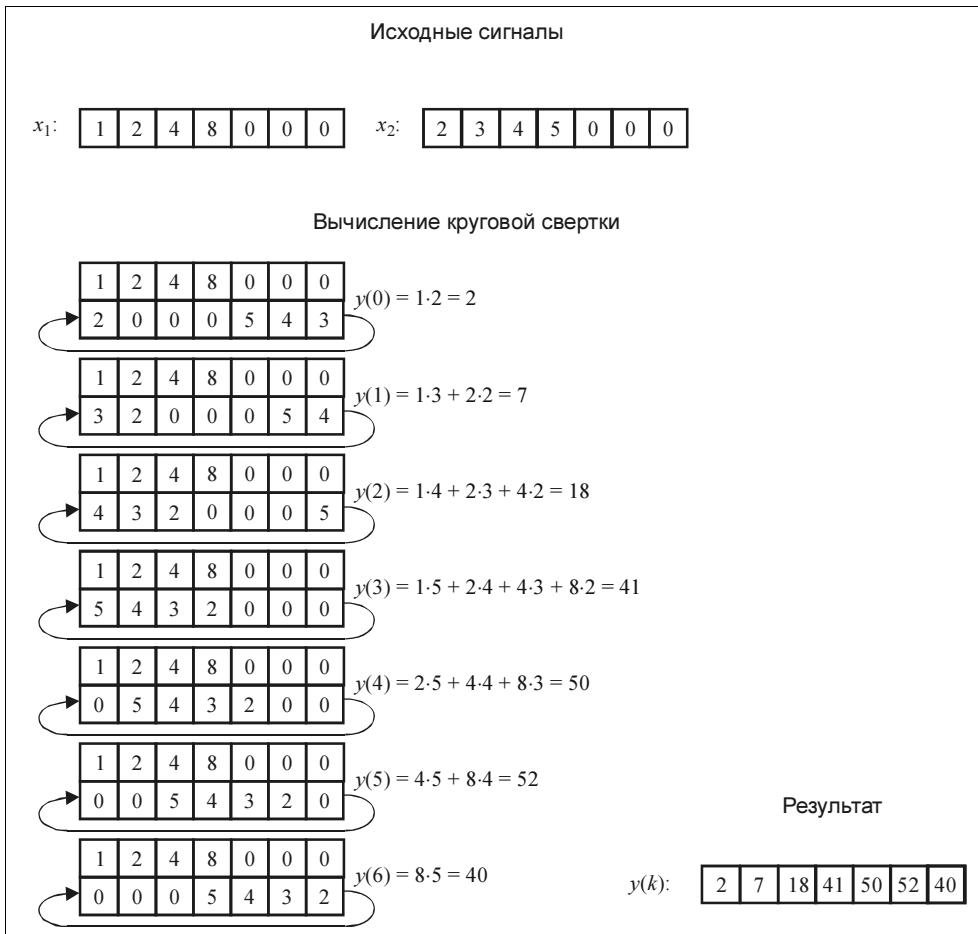
$y(3) = 1 \cdot 5 + 2 \cdot 4 + 4 \cdot 3 + 8 \cdot 2 = 41$

Результат

$y(k):$ 

52	59	58	41
----	----	----	----

**Рис. 5.11.** Расчет линейной (сверху) и круговой (снизу) свертки дискретных последовательностей



**Рис. 5.12.** Круговая свертка последовательностей, дополненных нулями, совпадает с их линейной сверткой

равными и не меньшими, чем сумма длин исходных последовательностей минус единица.

2. Вычисляются ДПФ дополненных нулями последовательностей.
3. Вычисленные ДПФ поэлементно перемножаются.
4. Вычисляется обратное ДПФ от результата перемножения.

Реализуем обычную линейную свертку и приведенный алгоритм средствами MATLAB. Сначала задаем исходные последовательности:

```
>> x1 = [1 2 4 8];
>> x2 = [2 3 4 5];
```

Теперь вычислим линейную свертку, чтобы затем было с чем сравнивать результаты фильтрации с помощью ДПФ:

```
>> y = conv(x1, x2)
y =
    2     7    18    41    50    52    40
```

Далее реализуем рассмотренный алгоритм (забегая вперед, скажем, что прямое и обратное ДПФ в MATLAB вычисляются по быстрому алгоритму с помощью функций `fft` и `ifft` соответственно; описание данных функций будет приведено в этой главе далее). Дополнять векторы нулями в явном виде не обязательно — это сделает функция `fft` при принудительном задании размерности преобразования:

```
>> X1 = fft(x1, 8);
>> X2 = fft(x2, 8);
>> Y = X1 .* X2;
>> y_fft = ifft(Y)
y_fft =
    Columns 1 through 6
    2.0000    7.0000   18.0000   41.0000   50.0000   52.0000
    Columns 7 through 8
    40.0000    0.0000
```

Как видите, результат в точности соответствует полученному ранее.

Количество нулей, добавляемых к последовательностям, может быть больше минимально необходимого — при этом лишь появится соответствующее число краевых нулей в вычисленной свертке (именно это было проделано в только что приведенном примере). Таким образом, мы всегда можем выбрать количество нулей таким, чтобы длина получившейся последовательности была равна степени двойки, и использовать эффективный алгоритм БПФ. Это дает возможность существенно сократить число операций по сравнению с вычислением свертки обычным методом.

Недостатком описанного метода фильтрации является то, что входной сигнал обрабатывается *сразу целиком*, т. е. расчеты можно начать только *по окончании* входного сигнала. Однако на практике в большинстве случаев необходимо обрабатывать отсчеты входного сигнала по мере их поступления.

Возможным решением проблемы является *блоковая*, или *секционная* обработка входного сигнала. При этом последовательность входных отсчетов делится на блоки (секции) заданного размера, и эти блоки подвергаются фильтрации в частотной области по отдельности. При выборе размера блока необходимо реализовать компромисс между числом операций (чем больше размер блока, тем эффективнее использование алгоритма БПФ) и задержкой получения выходного сигнала (чем меньше размер блока, тем меньше эта задержка).

При использовании блоковой фильтрации необходимо разобраться в идее объединения выходных сигналов, полученных при обработке отдельных блоков. Рассмотрим ее на конкретном примере.

Пусть входной сигнал  $x$  и импульсная характеристика фильтра  $h$  имеют следующий вид:

```
>> h = [1 4 2];
>> x = [1 2 3 4 5 4 3 3 2 2 1 1];
```

Прежде всего вычислим линейную свертку, чтобы знать, что должно получиться:

```
>> y = conv(x, h)
y =
Columns 1 through 10
    1      6     13     20     27     32     29     23     20     16
Columns 11 through 14
    13      9      6      2
```

Теперь разобьем входной сигнал на два блока по 6 отсчетов и обработаем их по отдельности:

```
>> x1 = x(1:6);
>> x2 = x(7:end);
>> y1 = conv(x1, h)
y1 =
    1      6     13     20     27     32     26      8
>> y2 = conv(x2, h)
y2 =
    3     15     20     16     13      9      6      2
```

Анализируя полученные результаты, мы заметим, что первые шесть отсчетов вектора  $y_1$  совпадают с первыми шестью элементами полного выходного сигнала  $y$ . Последние шесть отсчетов вектора  $y_2$ , в свою очередь, совпадают с шестью последними элементами полного выходного сигнала  $y$ . Оставшиеся два "средних" элемента (7-й и 8-й) вектора  $y$  могут быть получены суммированием двух последних элементов вектора  $y_1$  с двумя первыми элементами вектора  $y_2$ .

```
>> y_add = [y1(1:6) y1(7:8)+y2(1:2) y2(3:end)]
y_add =
Columns 1 through 10
    1      6     13     20     27     32     29     23     20     16
Columns 11 through 14
    13      9      6      2
```

Результат совпадает с полученным ранее. Итак, в общем случае фильтрация организуется следующим образом.

1. Входной сигнал разбивается на блоки длиной  $N$  отсчетов.
2. Каждый блок фильтруется независимо. Длина выходного сигнала составляет  $N + M - 1$  отсчетов, где  $M$  — длина импульсной характеристики фильтра. Для повышения эффективности фильтрация осуществляется в частотной области с использованием БПФ.
3. Блоки выходного сигнала объединяются, при этом их крайние  $M - 1$  отсчетов перекрываются и суммируются.

Данный алгоритм называется методом *перекрытия с суммированием* (*overlap-add*). Другой возможный вариант — *перекрытие с накоплением* (*overlap-save*). При этом входной сигнал разбивается на блоки, перекрывающиеся по краям на  $M - 1$  отсчетов, а у выходного сигнала для каждого блока отбрасываются крайние "хвосты" по

$M - 1$  отсчетов с каждой стороны. После этого выходные блоки объединяются без перекрытия.

## "Идеальная" фильтрация путем обнуления части спектральных отсчетов

У читателя, не слишком глубоко знакомого с теорией дискретных сигналов и систем, может возникнуть впечатление, что, рассчитав ДПФ сигнала, обнулив часть спектра и вычислив обратное ДПФ, можно реализовать идеальный фильтр, пропускающий некоторую заданную полосу частот. В данном разделе мы поговорим о том, что происходит при такой обработке сигнала и насколько "идеальным" является получаемый результат фильтрации.

Первое, что следует отметить, касается того факта, что у вещественного фильтра частотная характеристика является симметричной относительно нулевой частоты, поэтому, обнуляя часть спектра, следует выбирать обнуляемую часть правильно, т. е. с учетом этого факта.

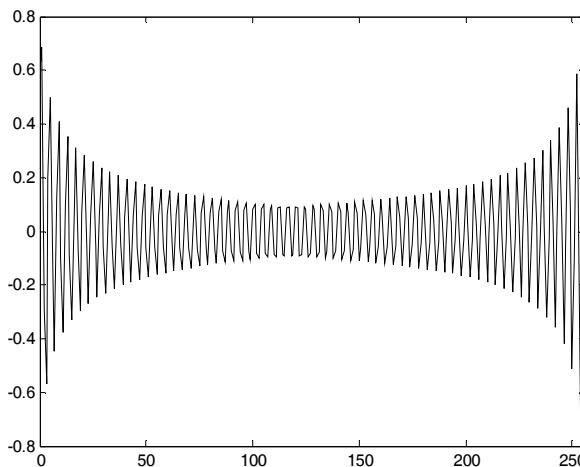
Кроме того, следует помнить, что при ДПФ подразумевается периодическое продолжение сигнала, поэтому при недостаточном количестве нулевых отсчетов в конце обрабатываемого сигнала возможно циклическое наложение переходных процессов в выходном сигнале, искажающее ожидаемый результат.

Наконец, необходимо понимать, что "идеальной" такая фильтрация будет только применительно к *периодически продолженному* сигналу. Действительно, после обнуления части спектральных коэффициентов и выполнения обратного ДПФ в дискретном спектре, соответствующем периодически продолженному выходному сигналу, не будет содержаться гармоник на обнуленных частотах. Но если рассматривать входной и выходной сигналы как последовательности конечной длины, обладающие непрерывными спектральными функциями, фильтрация будет отнюдь не идеальной. Строго говоря, результат вообще не будет соответствовать обработке сигнала линейной системой с постоянными параметрами — из-за того, что свертка при его получении выполнялась не линейная, а круговая.

В качестве примера продемонстрируем, что произойдет, если для синусоиды с частотой, составляющей 0,51 от частоты Найквиста, вычислить ДПФ, а потом обнулить спектральные коэффициенты, соответствующие частотам, превышающим половину частоты Найквиста. В результате получается "идеальный ФНЧ", который не должен пропускать частоту этой синусоиды. Результат, полученный после вычисления обратного ДПФ, показан на рис. 5.13.

```
>> N = 256; % размер ДПФ
>> N0 = N/4; % номер коэффициента, соответствующего частоте среза
>> f = 0.51; % нормированная частота синусоиды
>> t = 0:N-1; % вектор моментов времени
>> s = sin(pi*f*t); % обрабатываемый сигнал
>> sf = fft(s); % расчет ДПФ
>> sf(N0+2:N-N0) = 0; % обнуление части спектра
```

```
>> s2 = ifft(sf); % обратное ДПФ
>> plot(t, s2) % график сигнала
>> xlim([0 N])
```



**Рис. 5.13.** Результат "идеальной фильтрации" синусоиды, частота которой превышает частоту среза реализованного ФНЧ

Из графика видно, что уровень колебаний, полученных после "идеального фильтра", довольно высок; кроме того, их амплитуда увеличивается на краях сигнала — это связано с тем, что, как уже было не раз сказано, при использовании ДПФ подразумевается периодическое продолжение сигнала.

Несколько ослабить проблему можно, дополнив обрабатываемый сигнал нулями и соответствующим образом увеличив размерность ДПФ, но в любом случае данный подход *не эквивалентен* линейной фильтрации непериодического сигнала, а получаемые при его использовании результаты далеки от "идеальных". Поэтому при реализации фильтрации в частотной области следует использовать методы, описанные в этой главе ранее.

## Растекание спектра

Как уже говорилось, при ДПФ предполагается, что последовательность отсчетов анализируемого сигнала является периодически продолженной вперед и назад во времени. При этом, если значения начальных и конечных отсчетов сигнала сильно различаются, при периодическом повторении на стыках сегментов возникают скачки, из-за которых спектр сигнала расширяется.

Это явление, называемое *растеканием спектра* (*spectrum leakage*), можно наглядно проиллюстрировать на простейшем примере вычисления спектра дискретного гармонического сигнала

$$s(k) = A \cos(\omega kT + \varphi), \quad k = 0, 1, \dots, N-1. \quad (5.20)$$

Если анализируемая последовательность содержит *целое* число периодов гармонического сигнала (т. е. если отношение  $N\omega T/(2\pi)$  является целым числом), то периодически продолженный сигнал представляет собой гармонические колебания (без скачков), а подстановка (5.20) в формулу ДПФ (5.3) показывает, что вычисленное ДПФ содержит лишь два спектральных отсчета, отличных от нуля:

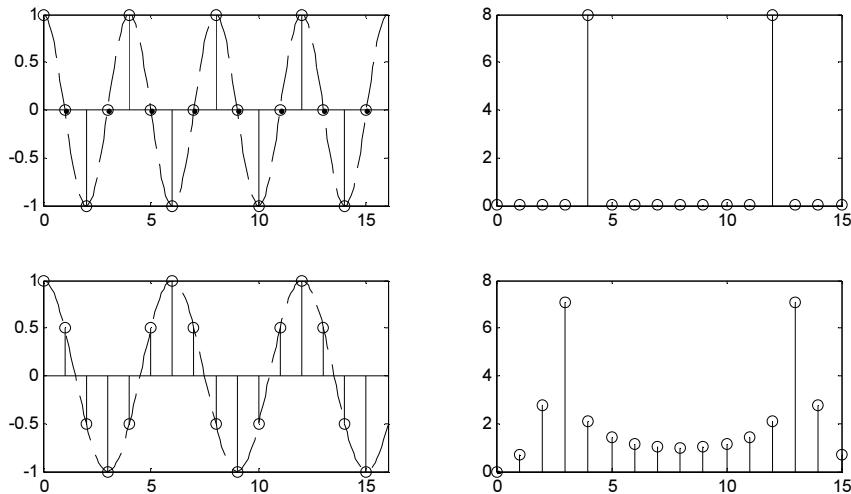
$$\dot{X}(n) = \begin{cases} \frac{AN}{2} e^{j\varphi}, & n = \frac{\omega T}{2\pi} N, \\ \frac{AN}{2} e^{-j\varphi}, & n = \left(1 - \frac{\omega T}{2\pi}\right) N, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Таким образом, аналогично спектру непрерывного гармонического сигнала, ДПФ отличается от нуля всего для двух значений  $n$ . Однако если отношение  $N\omega T/(2\pi)$  не является целым числом, спектр оказывается значительно более богатым. Этому можно дать простое объяснение: ведь в данном случае периодически продолженная последовательность уже не будет являться набором отсчетов непрерывной синусоиды. Поэтому, в полном соответствии со свойствами преобразования Фурье, в спектре появляются дополнительные составляющие.

Построим графики дискретных сигналов, содержащих по 16 отсчетов гармонических сигналов с периодами, равными 4 отсчетам (периодически продолженный сигнал является периодическим, рис. 5.14 сверху) и 6 отсчетам (периодически продолженный сигнал содержит скачки, рис. 5.14 снизу), и покажем вид модуля их ДПФ:

```
>> td = 0:15; % время для дискретных сигналов
>> ta = 0:0.1:16; % время для аналоговых сигналов
>> T1 = 4; % период первого сигнала — 4 отсчета
>> T2 = 6; % период второго сигнала — 6 отсчетов
>> x1d = cos(2*pi*td/T1); % дискретный сигнал
>> x1a = cos(2*pi*ta/T1); % аналоговый сигнал
>> y1 = fft(x1d); % ДПФ
>> x2d = cos(2*pi*td/T2); % дискретный сигнал
>> x2a = cos(2*pi*ta/T2); % аналоговый сигнал
>> y2 = fft(x2d); % ДПФ
>> subplot(2, 2, 1)
>> stem(td, x1d) % график дискретного сигнала
>> hold on
>> plot(ta, x1a, '--') % график аналогового сигнала
>> hold off
>> xlim([0 16])
>> subplot(2, 2, 2)
>> stem(td, abs(y1)) % график спектра
>> subplot(2, 2, 3)
>> stem(td, x2d) % график дискретного сигнала
>> hold on
```

```
>> plot(td, x2a, '--')      % график аналогового сигнала
>> hold off
>> xlim([0 16])
>> subplot(2, 2, 4)
>> stem(td, abs(y2))       % график спектра
```



**Рис. 5.14.** Дискретное преобразование Фурье для целого (сверху) и нецелого (снизу) числа периодов гармонического сигнала (слева — исходные последовательности, справа — модули их ДПФ)

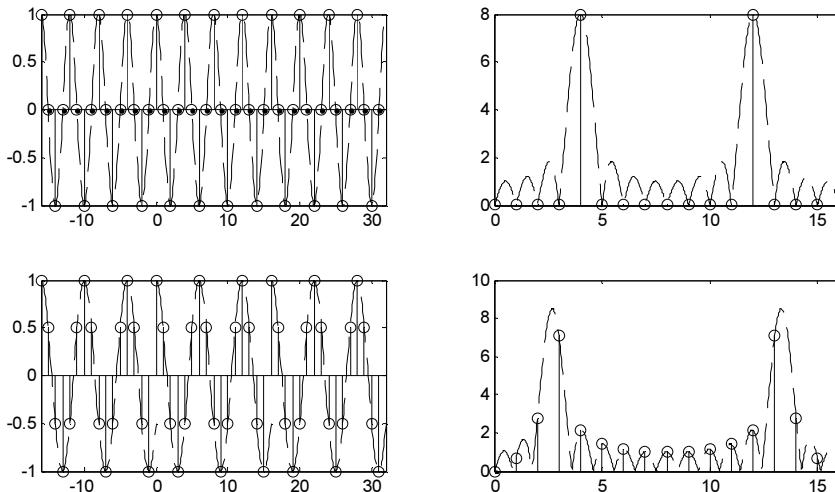
Необходимо подчеркнуть, что причиной растекания спектра является именно периодическое продолжение анализируемого сигнала. Спектр *одиночного* фрагмента дискретной синусоиды, в соответствии с тем, что говорилось в главе 3, является периодической непрерывной функцией частоты. Эта функция имеет лепестковую структуру независимо от того, целое или нецелое число периодов укладывается в анализируемом сегменте. Однако дискретный ряд частот, на которых вычисляется ДПФ, может быть по-разному расположен относительно лепестков спектральной функции. В случае целого числа периодов все анализируемые частоты (кроме двух) попадают как раз на границы между лепестками. При нецелом числе периодов такого не происходит. Для иллюстрации этого факта построим графики периодически продолженных сигналов, а также модулей их ДПФ и непрерывных амплитудных спектров одиночных фрагментов синусоиды (рис. 5.15):

```
>> td_m = [[td-16, td, td+16]];
>> x1d_m = [x1d, x1d, x1d]; % продолженный сигнал x1d
>> x2d_m = [x2d, x2d, x2d]; % продолженный сигнал x2d
>> % спектры одиночных сигналов x1d и x2d
>> [h1, w1] = freqz(x1d, 1, [], 16, 'whole');
>> [h2, w2] = freqz(x2d, 1, [], 16, 'whole');
```

```

>> subplot(2, 2, 1)
>> stem(td_m, x1d_m)      % график продолженного сигнала x1d
>> hold on
>> % график продолженного сигнала x1a
>> plot(ta-16, x1a, 'b--', ta, x1a, 'b--', ta+16, x1a, 'b--')
>> hold off
>> xlim([-16 32])
>> subplot(2, 2, 2)
>> stem(td, abs(y1))      % график модуля ДПФ сигнала x1d
>> hold on
>> plot(w1, abs(h1), '--') % спектр одиночного сигнала x1d
>> xlim([0 16])
>> hold off
>> subplot(2, 2, 3)
>> stem(td_m, x2d_m)      % график продолженного сигнала x2d
>> hold on
>> % график продолженного сигнала x2a
>> plot(ta-16, x2a, 'b--', ta, x2a, 'b--', ta+16, x2a, 'b--')
>> hold off
>> xlim([-16 32])
>> subplot(2, 2, 4)
>> stem(td, abs(y2))      % график модуля ДПФ сигнала x2d
>> hold on
>> plot(w2, abs(h2), '--') % спектр одиночного сигнала x2d
>> xlim([0 16])
>> hold off

```



**Рис. 5.15.** Происхождение растекания спектра: слева — периодически продолженные сигналы, справа — амплитудные спектры одиночных сигналов (пунктирные линии) и модули ДПФ (кружочки)

Возможна еще одна трактовка причины растекания спектра. В этом случае мы не считаем анализируемый сигнал периодически продолженным, а предполагаем, что все содержащиеся в нем гармонические компоненты существуют и за пределами анализируемого фрагмента. Спектр такого сигнала представляет собой набор дельта-функций, а растекание спектра при ДПФ можно объяснить наличием боковых лепестков у АЧХ фильтров, соответствующих отдельным частотным каналам ДПФ (см. ранее разд. "ДПФ как дискретная фильтрация" этой главы, формулу (5.19) и рис. 5.9).

### **ЗАМЕЧАНИЕ**

Аналогичные проблемы, связанные с наличием боковых лепестков у АЧХ фильтров, соответствующих отдельным частотным каналам ДПФ, возникают при использовании ДПФ для синтеза дискретных фильтров (см. разд. "Синтез с использованием окон" главы 6).

## **Весовые функции**

Для уменьшения растекания спектра при ДПФ применяются *весовые функции* (*weighting functions*), которые также называют *окнами* (*window*). В этом случае перед расчетом ДПФ сигнал умножается на вещественную неотрицательную весовую функцию  $w(k)$ , которая должна спадать к краям сегмента. Формула прямого ДПФ (5.3) при использовании весовой функции принимает следующий вид:

$$\dot{X}_w(n) = \sum_{k=0}^{N-1} x(k)w(k)e^{-j\frac{2\pi nk}{N}}.$$

Роль весовой функции в этой формуле можно рассматривать с различных точек зрения. Сначала проанализируем ситуацию во временной области. Если мы используем весовую функцию, которая имеет максимум в середине (при  $k = N/2$ ) и плавно спадает к краям ( $k = 0$  и  $k = N - 1$ ), то это приведет к ослаблению эффектов, связанных с возникновением скачков сигнала при периодическом повторении анализируемой конечной последовательности, и, таким образом, к уменьшению растекания спектра.

Аналогичный вывод можно сделать, рассмотрев влияние весовой функции в частотной области. Умножение сигнала на весовую функцию соответствует *свертке* спектров сигнала и весовой функции (см. ранее разд. "Свойства дискретного преобразования Фурье" этой главы). Это приводит к тому, что пики, содержащиеся в спектре сигнала, несколько расширяются. Однако при этом становится возможно уменьшить уровень боковых лепестков спектральной функции, что и является целью применения весовых функций.

Если трактовать ДПФ как фильтрацию, при использовании весовой функции  $w(k)$  получаются частотные характеристики фильтров следующего вида:

$$\dot{K}_{n,w}(\omega) = \sum_{k=0}^{N-1} w(N-k)e^{j2\pi\frac{nk}{N}}e^{-j\omega kT}.$$

Выбирая весовую функцию  $w(k)$  определенным образом, можно уменьшить уровень боковых лепестков частотной характеристики фильтров, соответствующих отдельным каналам ДПФ. Естественно, платой за это является расширение центрального лепестка частотной характеристики.

## Спектр дискретного случайного процесса

Все сказанное в этой главе до сих пор относилось к детерминированным сигналам, теперь же мы переходим к рассмотрению спектров дискретных случайных процессов.

Для определения спектральных характеристик дискретного стационарного случайного процесса используем тот же подход, что и в аналоговом случае (см. разд. "Спектральные характеристики случайных процессов" главы 1 и формулу (1.73)) — будем усреднять спектр мощности:

$$W(\omega) = \lim_{n \rightarrow \infty} \frac{1}{2n+1} \overline{\left| \sum_{k=-n}^n x(k)e^{-j\omega kT} \right|^2}. \quad (5.21)$$

Черта сверху обозначает здесь усреднение по ансамблю реализаций. Если процесс эргодический, спектр мощности для всех реализаций является одинаковым и выполнять усреднение по ансамблю не обязательно.

Выполнять вычисления непосредственно по формуле (5.21) неудобно, поэтому попробуем привести ее к более приемлемому виду. Для этого раскроем выражение для квадрата модуля:

$$\begin{aligned} W(\omega) &= \lim_{n \rightarrow \infty} \frac{1}{2n+1} \left[ \overline{\sum_{k=-n}^n x(k)e^{-j\omega kT}} \sum_{m=-n}^n x^*(m)e^{j\omega mT} \right] = \\ &= \lim_{n \rightarrow \infty} \frac{1}{2n+1} \sum_{k=-n}^n \sum_{m=-n}^m \overline{x(k)x^*(m)} e^{-j\omega(k-m)T} = \\ &= \lim_{n \rightarrow \infty} \frac{1}{2n+1} \sum_{k=-n}^n \sum_{m=-n}^m R_x(k-m) e^{-j\omega(k-m)T}. \end{aligned}$$

Суммируемые слагаемые зависят от *разности* индексов суммирования  $k$  и  $m$ , поэтому можно преобразовать двойную сумму в одиночную:

$$W(\omega) = \lim_{n \rightarrow \infty} \frac{1}{2n+1} \sum_{l=-2n}^{2n} (2n+1-|l|) R_x(l) e^{-j\omega lT} = \lim_{n \rightarrow \infty} \sum_{l=-2n}^{2n} \left( 1 - \frac{|l|}{2n+1} \right) R_x(l) e^{-j\omega lT}.$$

Поскольку при любом  $l$

$$\lim_{n \rightarrow \infty} \left( 1 - \frac{|l|}{2n+1} \right) = 1,$$

окончательно получаем

$$W(\omega) = \sum_{k=-\infty}^{\infty} R_x(k) e^{-j\omega kT}. \quad (5.22)$$

Выражение (5.22) представляет собой дискретный аналог теоремы Винера — Хинчина: *спектр дискретного случайного процесса является преобразованием Фурье от его корреляционной функции.*

## Непараметрические методы

При использовании непараметрических методов расчета спектра случайного процесса используется только информация, заключенная в отсчетах сигнала, без каких-либо дополнительных предположений. Мы кратко рассмотрим два таких метода — периодограмму и метод Уэлча (Welch). Примеры реализации данных методов спектрального анализа будут приведены далее, при описании соответствующих функций MATLAB.

### Периодограмма

Этим довольно странно звучащим термином — *периодограмма (periodogram)* — называется оценка спектральной плотности мощности, полученная по  $N$  отсчетам одной реализации случайного процесса согласно определению (5.21) (естественно, не путем взятия предела, а усреднением конечного числа слагаемых). Таким образом, периодограмма рассчитывается по следующей формуле:

$$\hat{W}(\omega) = \frac{1}{Nf_d} \left| \sum_{k=0}^{N-1} x(k) e^{-j\omega kT} \right|^2. \quad (5.23)$$

Деление на частоту дискретизации  $f_d$  необходимо для получения оценки спектральной плотности мощности *аналогового* случайного процесса, восстановленного по отсчетам  $x(k)$  (ведь в формуле (3.7), связывающей спектры аналогового и дискретизированного сигналов, имеется множитель  $1/T = f_d$ ).

Если при расчете спектра используется весовая функция (окно) с коэффициентами  $w(k)$ , формула (5.23) слегка модифицируется — вместо числа отсчетов  $N$  в знаменателе должна стоять сумма квадратов модулей коэффициентов окна. Полученная оценка спектра мощности называется *модифицированной периодограммой (modified periodogram)*:

$$\hat{W}(\omega) = \frac{1}{f_d} \frac{\left| \sum_{k=0}^{N-1} x(k) w(k) e^{-j\omega kT} \right|^2}{\sum_{k=0}^{N-1} |w(k)|^2}. \quad (5.24)$$

В книге [12] показано, что периодограмма не является состоятельной оценкой спектральной плотности мощности, поскольку дисперсия такой оценки сравнима с квадратом ее математического ожидания независимо от числа отсчетов  $N$  в обрабатываемом фрагменте сигнала. С ростом числа используемых отсчетов значения периодограммы лишь начинают все быстрее флюктуировать, так что ее график становится все более изрезанным.

## Метод Уэлча

При вычислении периодограммы по длинному фрагменту случайного сигнала она оказывается весьма изрезанной (см. далее пример вычисления периодограммы в разделе, посвященном описанию функции `periodogram` — рис. 5.30). Для уменьшения этой изрезанности необходимо применить какое-либо усреднение. Даньелл (Daniell) предложил сглаживать быстрые флюктуации выборочного спектра путем усреднения по соседним частотам спектра. Данный метод, называемый *периодограммой Даньелла*, сводится к вычислению свертки периодограммы со сглаживающей функцией. В методе Бартлетта (*Bartlett*) анализируемый сигнал делится на неперекрывающиеся фрагменты, для каждого фрагмента вычисляется периодограмма и затем эти периодограммы усредняются. Если корреляционная функция сигнала на длительности фрагмента затухает до пренебрежимо малых значений, то периодограммы отдельных фрагментов можно считать независимыми. В этом случае дисперсия *периодограммы Бартлетта* обратно пропорциональна числу используемых фрагментов, однако с ростом числа фрагментов при фиксированном общем числе отсчетов сигнала падает спектральное разрешение (за счет того, что фрагменты становятся короче).

Уэлч (*Welch*) внес в метод Бартлетта два усовершенствования: использование весовой функции и разбиение сигнала на *перекрывающиеся* фрагменты. Применение весовой функции позволяет ослабить растекание спектра и уменьшить смещение получаемой оценки спектра плотности мощности ценой незначительного ухудшения разрешающей способности. Перекрытие фрагментов введено для того, чтобы увеличить их число и уменьшить дисперсию оценки.

Итак, вычисления при использовании *метода Уэлча* (он называется еще методом усреднения модифицированных периодограмм — *averaged modified periodogram method*) организуются следующим образом:

1. Вектор отсчетов сигнала делится на перекрывающиеся фрагменты. Как правило, на практике используется перекрытие на 50%. Строго говоря, оптимальная степень перекрытия зависит от используемой весовой функции. В [12] приводятся данные о том, что для гауссовых случайных процессов при использовании окна Ханна минимальная дисперсия оценки спектра плотности мощности получается при перекрытии фрагментов на 65%.
2. Каждый фрагмент умножается на используемую весовую функцию.
3. Для взвешенных фрагментов вычисляются модифицированные периодограммы.
4. Периодограммы всех фрагментов усредняются.

Так же как и для периодограммы Бартлетта, дисперсия оценки, получаемой методом Уэлча, уменьшается примерно пропорционально числу фрагментов. Благодаря перекрытию в методе Уэлча используется больше фрагментов, поэтому дисперсия оценки спектра плотности мощности оказывается меньше, чем для метода Бартлетта.

Метод Уэлча, согласно [12], является наиболее популярным периодограммным методом спектрального анализа.

## Параметрические методы

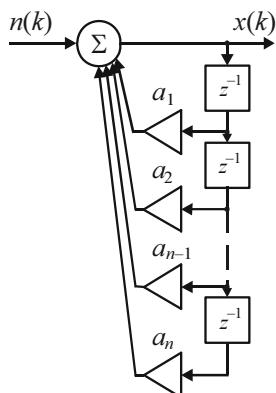
Использование параметрических методов подразумевает наличие некоторой математической модели анализируемого случайного процесса. Спектральный анализ сводится в данном случае к решению оптимизационной задачи, т. е. поиску таких параметров модели, при которых она наиболее близка к реально наблюдаемому сигналу.

Мы рассмотрим два параметрических метода: авторегрессионный и MUSIC (MUltiple SIgnal Classification).

## Авторегрессионная модель

Среди возможных параметрических методов спектрального анализа наибольшее распространение получили методы, основанные на *авторегрессионной* (*Autoregressive, AR*) модели формирования сигнала. Это обусловлено простотой модели, удобством расчетов на ее основе и тем, что данная модель хорошо соответствует многим реальным задачам.

Согласно авторегрессионной модели, сигнал  $\{x(k)\}$  формируется путем пропускания дискретного белого шума  $\{n(k)\}$  через "чисто рекурсивный" фильтр  $N$ -го порядка (рис. 5.16).



**Рис. 5.16.** Авторегрессионная модель формирования сигнала

### ЗАМЕЧАНИЕ

Помимо авторегрессионной существуют и другие модели формирования сигнала путем пропускания белого шума через формирующий фильтр. Так, в MA-модели

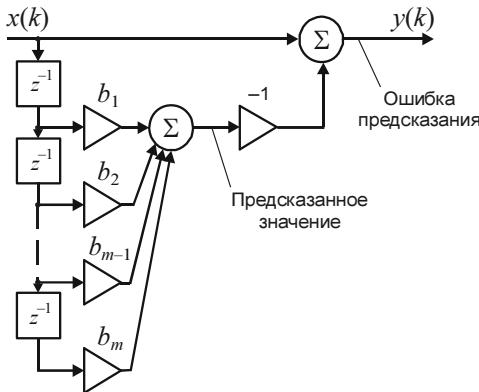
(*Moving Average*, "скользящее среднее") для этого используется нерекурсивный фильтр, а в *ARMA*-модели (*Autoregressive Moving Average*) — фильтр общего вида, содержащий рекурсивную и нерекурсивную ветви.

Спектральная плотность мощности такого сигнала пропорциональна квадрату модуля коэффициента функции передачи фильтра:

$$W(\omega) = \frac{\sigma_n^2}{f_d} \frac{1}{|1 - a_1 e^{-j\omega T} - a_2 e^{-j2\omega T} - \dots - a_N e^{-jN\omega T}|^2}. \quad (5.25)$$

Таким образом, данный метод спектрального анализа сводится к определению коэффициентов модели  $\{a_i\}$  заданного порядка  $N$ , оценке мощности белого шума  $\sigma_n^2$  и расчету спектральной плотности мощности по формуле (5.25).

Для определения коэффициентов модели производится минимизация ошибки линейного предсказания сигнала. Сущность этого метода состоит в следующем. Сигнал  $\{x(k)\}$  пропускается через нерекурсивный фильтр с коэффициентами  $\{1, -b_1, -b_2, \dots, -b_N\}$  (рис. 5.17).



**Рис. 5.17.** Линейное предсказание сигнала

Взвешенную сумму предыдущих отсчетов входного сигнала называют *линейным предсказанием* (*linear prediction*) следующего входного отсчета, а выходной сигнал рассматриваемого фильтра, т. е. разность между истинным и предсказанным значением отсчета — *ошибкой предсказания* (*prediction error*).

$Z$ -преобразование ошибки предсказания с учетом модели формирования сигнала  $\{x(k)\}$  можно записать следующим образом:

$$Y(z) = N(z) \frac{1 - b_1 z^{-1} - b_2 z^{-2} - \dots - b_N z^{-N}}{1 - a_1 z^{-1} - a_2 z^{-2} - \dots - a_N z^{-N}}.$$

Отношение двух полиномов можно в общем случае записать в виде  $z$ -преобразования бесконечной импульсной характеристики соответствующего рекурсивного фильтра,  $\{h(k)\}$ , причем, поскольку  $b_0 = 1$ , первый элемент этой импульсной характеристики в данном случае равен единице:

$$Y(z) = N(z)(1 + h_1 z^{-1} + h_2 z^{-2} + \dots).$$

Сам выходной сигнал будет представлять собой результат преобразования белого шума  $\{n(k)\}$  эквивалентным фильтром  $\{h(k)\}$ :

$$y(k) = n(k) + h_1 n(k-1) + h_2 n(k-2) + \dots$$

Поскольку отсчеты белого шума являются независимыми, дисперсия сигнала  $\{y(k)\}$  рассчитывается согласно следующей формуле:

$$\sigma_y^2 = \sigma_n^2 (1 + h_1^2 + h_2^2 + \dots)$$

Отсюда видно, что  $\sigma_y^2 \geq \sigma_n^2$  и равенство достигается, только когда  $h_i = 0$  для всех  $i$ .

Это возможно, если наборы коэффициентов  $\{a_i\}$  и  $\{b_i\}$  совпадают.

Таким образом, коэффициенты фильтра, минимизирующего дисперсию (или среднеквадратическое значение) ошибки предсказания, совпадают с коэффициентами авторегрессионной модели формирования сигнала.

Теперь обратимся к собственно процедуре расчета коэффициентов предсказывающего фильтра по отсчетам сигнала  $\{x(k)\}$ . Ошибка предсказания определяется как

$$\begin{aligned} y(k) &= x(k) - b_1 x(k-1) - b_2 x(k-2) - \dots - b_N x(k-N) = \\ &= x(k) - \sum_{m=1}^N b_m x(k-m). \end{aligned}$$

Нам нужно подобрать коэффициенты  $b_m$ , обеспечивающие минимальное среднеквадратическое значение  $y(k)$ . Возведем  $y(k)$  в квадрат:

$$\begin{aligned} y^2(k) &= x^2(k) - 2x(k) \sum_{m=1}^N b_m x(k-m) + \left( \sum_{m=1}^N b_m x(k-m) \right)^2 = \\ &= x^2(k) - 2x(k) \sum_{m=1}^N b_m x(k-m) + \sum_{m=1}^N \sum_{n=1}^N b_m b_n x(k-m) x(k-n). \end{aligned}$$

Теперь усредняем это по ансамблю реализаций, чтобы получить средний квадрат (считаем случайный процесс  $x(k)$  центрированным, поэтому математическое ожидание  $y(k)$  также равно нулю, а средний квадрат равен дисперсии):

$$\sigma_y^2 = \overline{y^2(k)} = \overline{x^2(k)} - 2 \sum_{m=1}^N b_m \overline{x(k)x(k-m)} + \sum_{m=1}^N \sum_{n=1}^N b_m b_n \overline{x(k-m)x(k-n)}.$$

Усредненные попарные произведения, входящие в это выражение, дают значения корреляционной функции входного сигнала:

$$\begin{aligned} \overline{x^2(k)} &= R_x(0) = \sigma_x^2, \\ \overline{x(k)x(k-m)} &= R_x(m), \\ \overline{x(k-m)x(k-n)} &= R_x(m-n). \end{aligned}$$

С учетом этих обозначений средний квадрат сигнала  $\{y(k)\}$  можно записать следующим образом:

$$\sigma_y^2 = \sigma_x^2 - 2 \sum_{m=1}^N b_m R_x(m) + \sum_{m=1}^N \sum_{n=1}^N b_m b_n R_x(m-n). \quad (5.26)$$

Дифференцирование этого выражения по  $b_k$  дает

$$\frac{\partial \sigma_y^2}{\partial b_k} = -2R_x(k) + 2 \sum_{m=1}^N b_m R_x(m-k).$$

Таким образом, для нахождения коэффициентов линейного предсказания необходимо решить систему линейных уравнений

$$\sum_{m=1}^N b_m R_x(m-k) = R_x(k), \quad k = 1, 2, \dots, N.$$

Можно записать эту систему в матричном виде:

$$\mathbf{R}_x \mathbf{b} = \mathbf{p}. \quad (5.27)$$

Здесь  $\mathbf{R}_x$  — корреляционная матрица сигнала  $\{x(k)\}$  (см. разд. "Дискретные случайные сигналы" главы 3),  $\mathbf{b}$  — столбец коэффициентов  $b_m$ ,  $\mathbf{p}$  — столбец значений корреляционной функции  $R_x(k)$  для  $k$  от 1 до  $N$ .

Теперь можно записать искомое решение:

$$\mathbf{b} = \mathbf{R}_x^{-1} \mathbf{p}. \quad (5.28)$$

### **ЗАМЕЧАНИЕ**

Формула (5.28) имеет очень важное значение, и сфера ее использования далеко не ограничивается расчетом коэффициентов линейного предсказания. Рассматривая нашу частную задачу, мы получили формулу *оптимального фильтра Винера*, который более подробно рассмотрен далее в главе 9.

Подстановка найденного вектора  $\mathbf{b}$  в формулу (5.26) дает минимально возможное значение среднего квадрата ошибки предсказания:

$$(\sigma_y^2)_{\min} = \sigma_x^2 - \mathbf{p}^T \mathbf{R}_x^{-1} \mathbf{p}. \quad (5.29)$$

### **ЗАМЕЧАНИЕ**

В приведенных выкладках предполагалось, что сигнал  $\{x(k)\}$  вещественный. Это было сделано лишь для упрощения изложения материала и не является принципиальным. В комплексном случае формула (5.28) не меняется, а операцию транспонирования в (5.29) следует заменить эрмитовым сопряжением (сочетанием транспонирования с комплексным сопряжением). Кроме того, при вычислении корреляционной функции комплексного сигнала один из перемножаемых сигналов подвергается комплексному сопряжению.

На практике мы не знаем истинной корреляционной функции исследуемого сигнала, поэтому для минимизации ошибки предсказания используются *оценки КФ*, полученные путем временного усреднения.

Итак, задача параметрического спектрального анализа распадается на два шага: сначала необходимо получить оценку корреляционной матрицы  $\mathbf{R}_x$ , а затем решить систему линейных уравнений (5.27). Рассмотрим эти два этапа более подробно.

## Оценка корреляционной матрицы

Элементы корреляционной матрицы эргодического дискретного случайного процесса определяются как усредненные по времени произведения сдвинутых копий бесконечно длинной реализации:

$$R_{x m+1, n+1} = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{k=-N}^N x^*(k-m)x(k-n) = \langle \mathbf{x}(m)^H \mathbf{x}(n) \rangle. \quad (5.30)$$

Здесь векторное обозначение  $\mathbf{x}(m)$  использовано для бесконечного столбца отсчетов реализации случайного процесса, задержанной на  $m$  тактов, а угловые скобки обозначают усреднение по времени. Для большей общности считается, что случайный процесс может быть комплексным, поэтому вместо транспонирования к одному из векторов применена операция эрмитова сопряжения (сочетания комплексного сопряжения и транспонирования).

Однако на практике нам доступен для наблюдения только фрагмент реализации, ограниченный по времени. Из-за этого при формировании сдвинутых копий возникает проблема, связанная с нехваткой отсчетов на краях анализируемого интервала. Поясним проблему краевых эффектов и возможные подходы к ее решению на простом примере. Пусть имеется последовательность из шести отсчетов,  $\{x(0), x(1), x(2), x(3), x(4), x(5)\}$ , и нам необходимо оценить ее корреляционную матрицу размером  $3 \times 3$ , т. е. при сдвигах на 0, 1 и 2 отсчета. Для этого, согласно (5.30), придется сформировать три вектора:  $\mathbf{x}(0)$ ,  $\mathbf{x}(1)$  и  $\mathbf{x}(2)$ . Если выписать друг под другом сдвинутые копии последовательности для требуемых сдвигов, становится очевидным, что для некоторых отсчетов сдвинутых пар не имеется:

$$\begin{aligned} \mathbf{x}(0): & \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \\ \mathbf{x}(1): & \quad \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \\ \mathbf{x}(2): & \quad \quad \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \end{aligned}$$

Данную проблему можно решать различными способами, перечисленными и про-комментированными в следующем списке. Следует сразу же отметить, что лишь в одном из этих случаев получающаяся оценка корреляционной матрицы будет являться матрицей Теплица (этим свойством, как указывалось в разд. "Дискретные случайные сигналы" главы 3, обладает истинная корреляционная матрица стационарного случайного процесса). Соответственно, и другие свойства истинной корреляционной матрицы, такие как положительная определенность, для получаемых оценок тоже могут не выполняться.

**ЗАМЕЧАНИЕ**

К сожалению, названия большинства методов оценки КФ, используемые в литературе по авторегрессионному спектральному анализу (см., например, [1]) и в документации соответствующих функций MATLAB, не отражают сущности выполняемых операций.

- *Автокорреляционный (autocorrelation) метод.* Недостающие отсчеты в начале и конце сигнала считаются равными нулю:

$$\begin{aligned}\mathbf{x}(0): & \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \quad 0 \quad 0 \\ \mathbf{x}(1): & \quad 0 \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \quad 0 \\ \mathbf{x}(2): & \quad 0 \quad 0 \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5)\end{aligned}$$

При усреднении по времени для нормировки используется длина *исходной* реализации (в нашем примере — 6). Этот метод всегда дает теплицеву матрицу.

- *Метод предзвешивания (prewindowing).* Недостающие отсчеты в начале сигнала считаются равными нулю, в конце сигнал обрезается, чтобы при всех сдвигах отсчетов хватало:

$$\begin{aligned}\mathbf{x}(0): & \quad x(0) \quad x(1) \quad X(2) \quad x(3) \quad x(4) \quad x(5) \\ \mathbf{x}(1): & \quad 0 \quad x(0) \quad X(1) \quad x(2) \quad x(3) \quad x(4) \\ \mathbf{x}(2): & \quad 0 \quad 0 \quad X(0) \quad x(1) \quad x(2) \quad x(3)\end{aligned}$$

При усреднении по времени для нормировки используется длина *исходной* реализации (в нашем примере — 6). Получаемая оценка не является теплицевой матрицей.

- *Метод постзвешивания (postwindowing).* Недостающие отсчеты в конце сигнала считаются равными нулю, в начале сигнал обрезается, чтобы при всех сдвигах отсчетов хватало:

$$\begin{aligned}\mathbf{x}(0): & \quad x(2) \quad x(3) \quad x(4) \quad x(5) \quad 0 \quad 0 \\ \mathbf{x}(1): & \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5) \quad 0 \\ \mathbf{x}(2): & \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x(5)\end{aligned}$$

При усреднении по времени для нормировки используется длина *исходной* реализации (в нашем примере — 6). Получаемая оценка не является теплицевой матрицей.

- *Ковариационный (covariance) метод.* Сигнал обрезается в начале и в конце, чтобы при всех сдвигах отсчетов хватало:

$$\begin{aligned}\mathbf{x}(0): & \quad x(2) \quad x(3) \quad x(4) \quad x(5) \\ \mathbf{x}(1): & \quad x(1) \quad x(2) \quad x(3) \quad x(4) \\ \mathbf{x}(2): & \quad x(0) \quad x(1) \quad x(2) \quad x(3)\end{aligned}$$

При усреднении по времени для нормировки используется длина *обрезанной* реализации (в нашем примере — 4). Получаемая оценка не является теплицевой матрицей.

- *Модифицированный ковариационный (modified covariance)* метод. Сигнал обрезается в начале и в конце, а затем полученная матрица отсчетов расширяется: к ней добавляется ее комплексно-сопряженная копия, зеркально перевернутая вдоль координаты временных сдвигов:

$$\begin{aligned} \mathbf{x}(0): & \quad x(2) \quad x(3) \quad x(4) \quad x(5) \quad x^*(0) \quad x^*(1) \quad x^*(2) \quad x^*(3) \\ \mathbf{x}(1): & \quad x(1) \quad x(2) \quad x(3) \quad x(4) \quad x^*(1) \quad x^*(2) \quad x^*(3) \quad x^*(4) \\ \mathbf{x}(2): & \quad x(0) \quad x(1) \quad x(2) \quad x(3) \quad x^*(2) \quad x^*(3) \quad x^*(4) \quad x^*(5) \end{aligned}$$

При усреднении по времени для нормировки используется длина *расширенной* реализации (в нашем примере — 8). Получаемая оценка не является теплицевой матрицей.

Подробнее узнать о свойствах перечисленных методов оценивания корреляционной матрицы можно из книги [12].

### Алгоритм Левинсона — Дарбина

В общем случае система (5.27) может решаться обычными методами решения систем линейных уравнений. Однако если корреляционная матрица  $\mathbf{R}_x$  имеет теплицеву структуру (это имеет место, если используется аналитически полученная корреляционная матрица стационарного случайного процесса либо ее оценка, рассчитанная автокорреляционным методом, рассмотренным ранее), можно сократить число вычислений, воспользовавшись *алгоритмом Левинсона — Дарбина (Levinson — Durbin algorithm)*.

Перепишем систему уравнений (5.27), раскрыв в ней структуру матрицы  $\mathbf{R}_x$  и вектора  $\mathbf{p}$ :

$$\left[ \begin{array}{ccccc} R_x(0) & R_x(1) & R_x(2) & \dots & R_x(N-1) \\ R_x(1) & R_x(0) & R_x(1) & \dots & R_x(N-2) \\ R_x(2) & R_x(1) & R_x(0) & \dots & R_x(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_x(N-1) & R_x(N-2) & R_x(N-3) & \dots & R_x(0) \end{array} \right] \left[ \begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{array} \right] = \left[ \begin{array}{c} R_x(1) \\ R_x(2) \\ R_x(3) \\ \vdots \\ R_x(N) \end{array} \right]$$

В такой форме записи становится наглядно видно, что, помимо уже отмеченной теплицевой структуры матрицы, данная система обладает еще одной особенностью: вектор-столбец правой части составлен из тех же значений корреляционной функции, которые формируют матрицу системы. Исключение составляет лишь последний элемент  $R_x(N)$ , отсутствующий в корреляционной матрице.

Сущность алгоритма Левинсона — Дарбина состоит в итерационном решении данной системы уравнений с постепенным увеличением ее порядка. В процессе решения мы получаем решение системы (5.27) для порядка предсказания, равного 1, 2, 3, ...,  $N$ . Таким образом, при переходе к следующей итерации необходимо выполнить две задачи: рассчитать значение нового коэффициента предсказания и обновить величины уже имеющихся.

Для использования в дальнейших выкладках введем следующие обозначения:

- $\mathbf{R}_k$  — корреляционная матрица  $\mathbf{R}_x$  размером  $k \times k$ :

$$\mathbf{R}_k = \begin{bmatrix} R_x(0) & R_x(1) & R_x(2) & \dots & R_x(k-1) \\ R_x(1) & R_x(0) & R_x(1) & \dots & R_x(k-2) \\ R_x(2) & R_x(1) & R_x(0) & \dots & R_x(k-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_x(k-1) & R_x(k-2) & R_x(k-3) & \dots & R_x(0) \end{bmatrix};$$

- $\mathbf{r}_k$  — вектор-столбец значений корреляционной функции  $R_x(1), R_x(2), \dots, R_x(k)$  (правая часть системы порядка  $k$ ):

$$\mathbf{r}_k = \begin{bmatrix} R_x(1) \\ R_x(2) \\ R_x(3) \\ \vdots \\ R_x(k) \end{bmatrix};$$

- $\mathbf{p}_k$  — *перевернутый* вектор-столбец значений корреляционной функции:

$$\mathbf{p}_k = \begin{bmatrix} R_x(k) \\ R_x(k-1) \\ R_x(k-2) \\ \vdots \\ R_x(1) \end{bmatrix};$$

- $\mathbf{b}_k^{(k)}$  — вектор-столбец коэффициентов предсказания  $k$ -го порядка, полученный на  $k$ -й итерации (в данном случае нижний индекс показывает размер вектора, а верхний индекс в скобках — номер итерации):

$$\mathbf{b}_k^{(k)} = \begin{bmatrix} b_1^{(k)} \\ b_2^{(k)} \\ b_3^{(k)} \\ \vdots \\ b_k^{(k)} \end{bmatrix},$$

где  $b_i^{(k)}$  —  $i$ -й элемент вектора коэффициентов предсказания, полученного на  $k$ -й итерации (для скалярных величин нижний индекс обозначает номер элемента вектора);

- $\mathbf{p}_k^{(k)}$  — *перевернутый* вектор-столбец коэффициентов предсказания  $k$ -го порядка, полученный на  $k$ -й итерации:

$$\mathbf{b}_k^{(k)} = \begin{bmatrix} b_k^{(k)} \\ b_{k-1}^{(k)} \\ b_{k-2}^{(k)} \\ \vdots \\ b_1^{(k)} \end{bmatrix};$$

- $\mathbf{b}_k^{(k+1)}$  — вектор-столбец коэффициентов предсказания  $k$ -го порядка, обновленный в результате выполнения  $(k + 1)$ -й итерации;
- $E_k$  — средний квадрат ошибки предсказания при использовании предсказывающего фильтра  $k$ -го порядка. Согласно (5.29) и введенным обозначениям,

$$E_k = R_x(0) - \mathbf{r}_k^T \mathbf{R}_k^{-1} \mathbf{r}_k. \quad (5.31)$$

Итак, сначала рассматриваем предсказание первого порядка. В этом случае в системе остается лишь одно уравнение:

$$R_x(0)b_1^{(1)} = R_x(1).$$

Это уравнение имеет очевидное решение:

$$b_1^{(1)} = \frac{R_x(1)}{R_x(0)}. \quad (5.32)$$

### ЗАМЕЧАНИЕ

Вектор коэффициентов предсказания в данном случае представляет собой скаляр, поэтому векторные обозначения в последних двух формулах не использованы.

Теперь предположим, что нам известно решение системы (5.27) для некоторого порядка  $k$ , т. е. в нашем распоряжении имеется вектор  $\mathbf{b}_k^{(k)}$ , для которого

$$\mathbf{R}_k \mathbf{b}_k^{(k)} = \mathbf{r}_k. \quad (5.33)$$

Кроме того, сразу же заметим, что благодаря теплицевой структуре корреляционной матрицы данное равенство не нарушится при вертикальном перевороте векторов-столбцов  $\mathbf{b}_k^{(k)}$  и  $\mathbf{r}_k$ :

$$\mathbf{R}_k \mathbf{b}_k^{(k)} = \mathbf{r}_k. \quad (5.34)$$

Наша задача состоит в том, чтобы выразить решение для  $(k + 1)$ -й итерации,  $\mathbf{b}_{k+1}^{(k+1)}$ , через уже имеющееся решение  $\mathbf{b}_k^{(k)}$ . Для этого нужно решить систему уравнений

$$\mathbf{R}_{k+1} \mathbf{b}_{k+1}^{(k+1)} = \mathbf{r}_{k+1}. \quad (5.35)$$

Прежде всего заметим, что по сравнению с предыдущей итерацией (5.33) к корреляционной матрице  $\mathbf{R}_k$  добавились новые строка и столбец, а к корреляционному

вектору  $\mathbf{r}_k$  — новый элемент  $R_x(k+1)$ . Покажем это в явном виде, переписав формулу (5.35) следующим образом:

$$\left[ \begin{array}{c|c} \mathbf{R}_k & \boldsymbol{\rho}_k \\ \hline \boldsymbol{\rho}_k^T & R_x(0) \end{array} \right] \left[ \begin{array}{c} \mathbf{b}_k^{(k+1)} \\ b_{k+1}^{(k+1)} \end{array} \right] = \left[ \begin{array}{c} \mathbf{r}_k \\ R_x(k+1) \end{array} \right].$$

Выполнив в этой формуле блоковое умножение матриц, получаем следующую систему из двух уравнений:

$$\begin{cases} \mathbf{R}_k \mathbf{b}_k^{(k+1)} + \boldsymbol{\rho}_k b_{k+1}^{(k+1)} = \mathbf{r}_k, \\ \boldsymbol{\rho}_k^T \mathbf{b}_k^{(k+1)} + R_x(0) b_{k+1}^{(k+1)} = R_x(k+1). \end{cases} \quad (5.36)$$

Подставим в первое уравнение этой системы выражение для  $\mathbf{r}_k$  из (5.33) и для  $\boldsymbol{\rho}_k$  из (5.34):

$$\mathbf{R}_k \mathbf{b}_k^{(k+1)} + \mathbf{R}_k \boldsymbol{\beta}_k^{(k)} b_{k+1}^{(k+1)} = \mathbf{R}_k \mathbf{b}_k^{(k)}.$$

При невырожденной матрице  $\mathbf{R}_k$  отсюда сразу же следует, что

$$\mathbf{b}_k^{(k+1)} = \mathbf{b}_k^{(k)} - \boldsymbol{\beta}_k^{(k)} b_{k+1}^{(k+1)}. \quad (5.37)$$

Таким образом, мы выразили первые  $k$  элементов нового вектора коэффициентов предсказания,  $\mathbf{b}_k^{(k+1)}$ , через их старые значения  $\mathbf{b}_k^{(k)}$ , полученные на предыдущей итерации. Однако в формулу (5.37) входит еще не рассчитанный элемент нового вектора —  $b_{k+1}^{(k+1)}$ . Чтобы определить его, подставим (5.37) во второе уравнение системы (5.36):

$$\boldsymbol{\rho}_k^T (\mathbf{b}_k^{(k)} - \boldsymbol{\beta}_k^{(k)} b_{k+1}^{(k+1)}) + R_x(0) b_{k+1}^{(k+1)} = R_x(k+1).$$

Это скалярное уравнение относительно  $b_{k+1}^{(k+1)}$ , решение которого находится элементарно:

$$b_{k+1}^{(k+1)} = \frac{R_x(k+1) - \boldsymbol{\rho}_k^T \mathbf{b}_k^{(k)}}{R_x(0) - \boldsymbol{\rho}_k^T \boldsymbol{\beta}_k^{(k)}}. \quad (5.38)$$

Выполнив вычисления сначала по формуле (5.38), а затем по формуле (5.37), можно получить окончательное значение нового вектора коэффициентов предсказания:

$$\mathbf{b}_{k+1}^{(k+1)} = \left[ \begin{array}{c} \mathbf{b}_k^{(k+1)} \\ b_{k+1}^{(k+1)} \end{array} \right]. \quad (5.39)$$

Дополнительно упростить расчеты можно, вычисляя знаменатель формулы (5.38) рекурсивно. Чтобы показать это, прежде всего заметим, что он равен среднему квадрату ошибки предсказания (5.31). Действительно,

$$R_x(0) - \boldsymbol{\rho}_k^T \boldsymbol{\beta}_k^{(k)} = R_x(0) - \mathbf{r}_k^T \mathbf{b}_k^{(k)} = R_x(0) - \mathbf{r}_k^T \mathbf{R}_k^{-1} \mathbf{r}_k = E_k. \quad (5.40)$$

Теперь выразим  $E_k$  через  $E_{k-1}$  — средний квадрат ошибки, полученный на предыдущей итерации. Для этого выделим в векторе  $\mathbf{r}_k$  добавляемый элемент, а вектор  $\mathbf{b}_k^{(k)}$  представим в виде (5.39):

$$E_k = R_x(0) - \mathbf{r}_k^T \mathbf{b}_k^{(k)} = R_x(0) - \left[ \begin{array}{c|c} \mathbf{r}_{k-1}^T & R_x(k) \end{array} \right] \begin{bmatrix} \mathbf{b}_{k-1}^{(k-1)} \\ b_k^{(k)} \end{bmatrix}.$$

Подставляем сюда (5.37) и выполняем умножение строки на столбец:

$$\begin{aligned} E_k &= R_x(0) - \left[ \begin{array}{c|c} \mathbf{r}_{k-1}^T & R_x(k) \end{array} \right] \begin{bmatrix} \mathbf{b}_{k-1}^{(k-1)} - \mathbf{P}_{k-1}^{(k-1)} b_k^{(k)} \\ b_k^{(k)} \end{bmatrix} = \\ &= R_x(0) - \mathbf{r}_{k-1}^T \mathbf{b}_{k-1}^{(k-1)} + \mathbf{r}_{k-1}^T \mathbf{P}_{k-1}^{(k-1)} b_k^{(k)} - R_x(k) b_k^{(k)} = \\ &= R_x(0) - \mathbf{r}_{k-1}^T \mathbf{b}_{k-1}^{(k-1)} - (R_x(k) - \mathbf{r}_{k-1}^T \mathbf{P}_{k-1}^{(k-1)}) b_k^{(k)}. \end{aligned}$$

В получившейся формуле первые два слагаемых, согласно (5.40), дают  $E_{k-1}$  — средний квадрат ошибки, полученный на предыдущей итерации:

$$R_x(0) - \mathbf{r}_{k-1}^T \mathbf{b}_{k-1}^{(k-1)} = E_{k-1},$$

а выражение в скобках, согласно (5.38) и (5.40), равно

$$R_x(k) - \mathbf{r}_{k-1}^T \mathbf{P}_{k-1}^{(k-1)} = b_k^{(k)} (R_x(0) - \mathbf{r}_{k-1}^T \mathbf{b}_{k-1}^{(k-1)}) = b_k^{(k)} E_{k-1}.$$

Таким образом, окончательно получаем для среднего квадрата ошибки предсказания очень простое рекурсивное выражение:

$$E_k = E_{k-1} - E_{k-1} (b_k^{(k)})^2 = E_{k-1} \left( 1 - (b_k^{(k)})^2 \right).$$

В заключение данного раздела укажем последовательность операций, выполняемых при реализации алгоритма Левинсона — Дарбина.

**Инициализация.** Вычисляются параметры предсказания первого порядка:  $b_1^{(1)}$  и средний квадрат ошибки предсказания  $E_1$

$$b_1^{(1)} = \frac{R_x(1)}{R_x(0)}, \quad E_1 = R_x(0) \left( 1 - (b_1^{(1)})^2 \right).$$

**Рекурсия.** Для  $k = 2, 3, \dots, N$  выполняются следующие действия:

□ вычисляется добавляемый элемент вектора коэффициентов предсказания:

$$b_k^{(k)} = \frac{R_x(k) - \mathbf{P}_{k-1}^T \mathbf{b}_{k-1}^{(k-1)}}{E_{k-1}};$$

### ЗАМЕЧАНИЕ

Величины  $b_k^{(k)}$ , получаемые в процессе расчета для  $k = 1, \dots, N$ , называются *коэффициентами отражения* (*reflection coefficients*). Эти значения фигурируют в схемах *решетчатых фильтров* (*lattice filter*), не рассматриваемых в данной книге.

- производится обновление остальных коэффициентов предсказания:

$$\mathbf{b}_{k-1}^{(k)} = \mathbf{b}_{k-1}^{(k-1)} - \boldsymbol{\beta}_{k-1}^{(k-1)} b_k^{(k)};$$

- пересчитывается средний квадрат ошибки предсказания:

$$E_k = E_{k-1} \left( 1 - \left( b_k^{(k)} \right)^2 \right).$$

## Методы авторегрессионного спектрального анализа

Существует целый ряд методов авторегрессионного анализа, отличающихся в основном способом получения оценки корреляционной матрицы. В большинстве методов используется один из вариантов оценки, рассмотренных в этой главе ранее. После получения оценки корреляционной матрицы производится решение системы линейных уравнений (5.27). Если оценка корреляционной матрицы имеет теплицеву структуру, для решения может использоваться рассмотренный в предыдущем разделе алгоритм Левинсона — Дарбина.

В книге [12] рассматриваются следующие методы авторегрессионного спектрального анализа (этот же набор методов реализован в пакете расширения MATLAB Signal Processing Toolbox): *метод Берга (Burg)*, *ковариационный метод*, *модифицированный ковариационный метод*, *автокорреляционный метод Юла — Уолкера (Yule — Walker)*. За исключением метода Берга, в их названиях содержится указание на способ получения оценки автокорреляционной матрицы (см. разд. "Оценка корреляционной матрицы" ранее в этой главе). Подробные теоретические сведения об этих методах можно найти в [12], здесь же отметим лишь то, что при анализе длинных последовательностей отсчетов все методы дают практически одинаковые результаты, различия начинают проявляться в случае обработки коротких фрагментов.

Достоинства и недостатки методов авторегрессионного спектрального анализа, реализованных в MATLAB, кратко перечислены в табл. 5.2. Приведенные сведения взяты из [12] и документации пакета Signal Processing.

**Таблица 5.2.** Характеристики авторегрессионных методов спектрального анализа

Метод	Достоинства	Недостатки
Берга	Высокая разрешающая способность при анализе коротких сигналов Гарантированная стабильность рассчитанного формирующего фильтра	Положения спектральных пиков сильно зависят от начальных фаз синусоид При большом порядке модели может наблюдаться расщепление спектральных пиков При анализе суммы синусоид с шумом получаются смещенные спектральные пики

Таблица 5.2 (окончание)

Метод	Достоинства	Недостатки
Ковариационный	Большая (по сравнению с методом Юла — Уолкера) разрешающая способность при анализе коротких сигналов  Возможность оценки частот для сигнала, представляющего собой сумму "чистых" синусоид	Рассчитанный формирующий фильтр может оказаться нестабильным  При анализе суммы синусоид с шумом получаются смещенные спектральные пики
Модифицированный ковариационный	Высокая разрешающая способность при анализе коротких сигналов  Возможность оценки частот для сигнала, представляющего собой сумму "чистых" синусоид  Отсутствие расщепления спектральных пики	Положения спектральных пики в некоторой степени зависят от начальных фаз синусоид  Рассчитанный формирующий фильтр может оказаться нестабильным  При анализе суммы синусоид с шумом получаются слегка смещенные спектральные пики
Юла — Уолкера	Хорошие результаты при анализе длинных сигналов  Гарантируемая стабильность рассчитанного формирующего фильтра	Плохие результаты при анализе коротких сигналов  При анализе суммы синусоид с шумом получаются смещенные спектральные пики

Авторегрессионные методы анализа спектра больше всего подходят для сигналов, действительно являющихся авторегрессионными процессами. Вообще, хорошие результаты эти методы дают тогда, когда спектр анализируемого сигнала имеет четко выраженные пики. В частности, к таким сигналам относится сумма нескольких синусоид с шумом (хотя авторегрессионным процессом такой сигнал не является).

При использовании авторегрессионных методов важно правильно выбрать порядок авторегрессионной модели — он должен быть в два раза больше числа синусоидальных колебаний, которые предположительно содержатся в анализируемом сигнале.

Пример, демонстрирующий влияние порядка модели на результаты расчетов, приведен далее, при описании функций авторегрессионного спектрального анализа, имеющихся в MATLAB.

Помимо спектрального анализа техника линейного предсказания широко используется для кодирования звуковых сигналов. При этом звук делится на фрагменты (кадры) и вместо самих отсчетов звука для каждого кадра передаются только коэффициенты авторегрессионной модели формирования звука, что позволяет во много раз уменьшить требуемую скорость передачи данных. Такая технология используется, например, в системах сотовой связи.

## Метод MUSIC

Метод *MUSIC* (*M*ULTiple *S*ignal *C*lassification) предназначен для спектрального анализа сигналов, представляющих собой сумму нескольких синусоид (точнее, в общем случае — нескольких комплексных экспонент) с белым шумом. Целью спектрального анализа подобных сигналов, как правило, является не расчет спектра как такового, а определение частот и уровней (амплитуд или мощностей) гармонических составляющих. Метод MUSIC предназначен именно для этого, поэтому получаемая с его помощью зависимость уровня сигнала от частоты называется *псевдоспектром* (*pseudospectrum*).

### ЗАМЕЧАНИЕ

Метод MUSIC опирается на модель обрабатываемого сигнала, поэтому в данной книге он отнесен к параметрическим. В то же время в нем не производится оптимизация параметров модели по какому-либо критерию. Поэтому в большинстве источников его и родственные ему методы выделяют в отдельную категорию — методы, основанные на анализе собственного пространства корреляционной матрицы сигнала.

В основе метода лежит анализ собственных чисел и собственных векторов корреляционной матрицы сигнала. Подробное математическое обоснование можно найти в [12], далее приводятся лишь базовые идеи.

Итак, пусть сигнал представляет собой сумму  $M$  комплексных экспонент, начальные фазы которых случайны, с белым шумом:

$$x(k) = n(k) + \sum_{m=1}^M A_m \exp(j\omega_m kT + j\phi_m). \quad (5.41)$$

Здесь  $n(k)$  — отсчеты дискретного белого шума;  $A_m$ ,  $\omega_m$  и  $\phi_m$  — соответственно амплитуды, частоты и начальные фазы комплексных экспонент, содержащихся в сигнале;  $T$  — период дискретизации. Начальные фазы  $\phi_m$  — независимые случайные величины, равномерно распределенные на интервале  $0\dots 2\pi$ .

### ЗАМЕЧАНИЕ

Каждая вещественная синусоидальная составляющая сигнала может быть представлена как сумма двух комплексных экспонент с помощью формулы Эйлера.

Корреляционная функция такого сигнала будет иметь следующий вид:

$$R_x(k) = \sigma_n^2 x_0(k) + \sum_{m=1}^M A_m^2 \exp(-j\omega_m kT). \quad (5.42)$$

Здесь  $x_0(k)$  — единичная импульсная функция (3.19), равная 1 при  $k = 0$  и 0 в остальных случаях;  $\sigma_n^2$  — дисперсия шума  $n(k)$ .

Далее из отсчетов корреляционной функции формируется корреляционная матрица размера  $N \times N$ , причем размер матрицы должен превышать число комплексных экспонент:  $N > M$ .

Анализируя собственные числа и собственные векторы такой корреляционной матрицы, можно показать, что наименьшее собственное число равно  $\sigma_n^2$ , и оно имеет кратность  $N - M$ . Остальные  $M$  собственных чисел превышают  $\sigma_n^2$ , а их конкретные значения зависят от амплитуд и частот комплексных экспонент.

Собственные векторы, соответствующие  $M$  наибольшим собственным числам, представляют собой линейные комбинации комплексных экспонент, содержащихся в сигнале. Множество всех таких линейных комбинаций называют *сигнальным подпространством* (*signal subspace*), а упомянутые собственные векторы образуют базис в этом подпространстве.

Оставшиеся  $N - M$  собственных векторов ортогональны всем комплексным экспонентам, содержащимся в сигнале. Эти векторы образуют базис *шумового подпространства* (*noise subspace*).

Спектр комплексной экспоненты, рассчитанный аналитически, представляет собой дельта-функцию, расположенную на соответствующей частоте. Чтобы получить бесконечные выбросы на частотах  $\omega_m$ , псевдоспектр при использовании метода MUSIC рассчитывается следующим образом:

$$W(\omega) = \frac{1}{\sum_{k=M+1}^N \left| \sum_{n=0}^{N-1} v_k(n) e^{-j\omega n T} \right|^2}. \quad (5.43)$$

Здесь  $v_k(n)$  —  $n$ -й элемент  $k$ -го собственного вектора корреляционной матрицы. Векторы пронумерованы по убыванию соответствующих им собственных чисел. Таким образом, внешняя сумма по  $k$  от  $M + 1$  до  $N$  задает участие в вычислениях только *шумовых* собственных векторов. Внутренняя сумма представляет собой вычисление спектра собственного вектора на частоте  $\omega$ . Шумовые собственные векторы ортогональны всем комплексным экспонентам с частотами  $\omega_m$ , содержащимся в сигнале. Поэтому при  $\omega = \omega_m$  все внутренние суммы равны нулю, и  $W(\omega) \rightarrow \infty$ .

Сами частоты  $\omega_m$  можно получить, приравняв к нулю знаменатель формулы (5.43):

$$\sum_{k=M+1}^N \left| \sum_{n=0}^{N-1} v_k(n) e^{-j\omega n T} \right|^2 = 0.$$

Раскроем квадрат модуля, превратив его в двойную сумму:

$$\sum_{k=M+1}^N \sum_{n=0}^{N-1} \sum_{l=0}^{N-1} v_k(n) v_k^*(l) e^{-j\omega(n-l)T} = 0. \quad (5.44)$$

Теперь видно, что левая часть полученного выражения в конечном счете представляет собой полином относительно  $e^{-j\omega T}$  степени  $2N - 2$ . Те корни этого полинома, которые лежат на единичной окружности, равны  $\exp(-j\omega_m T)$ , и для расчета частот остается вычислить аргументы этих комплексных чисел и разделить их на  $T$ .

Все сказанное до сих пор относилось к идеализированной модели сигнала (5.41) и аналитически полученной корреляционной функции (5.42). В реальных ситуациях, когда сигнал не вполне соответствует модели (5.41), а его корреляционная функция оценивается по результатам измерений (к тому же зачастую без возможности усреднения по нескольким наблюдениям), результаты оказываются несколько иными — пики в (5.43) будут иметь конечную высоту, а корни полинома (5.44) не будут лежать строго на единичной окружности. Однако если сигнал близок к предполагаемой модели, метод MUSIC дает хорошие результаты. Для расчета частот выбираются корни полинома (5.44), ближайшие к единичной окружности.

## Метод EV

Близким родственником MUSIC является метод анализа собственных векторов (*eigenvectors*, *EV*). Его отличие состоит лишь в том, что в формулах (5.43) и (5.44) при суммировании по  $k$  используются весовые коэффициенты, обратно пропорциональные соответствующим собственным числам. Таким образом, формула для псевдоспектра принимает вид:

$$W(\omega) = \frac{1}{\sum_{k=M+1}^N \frac{1}{\lambda_k} \left| \sum_{n=0}^{N-1} v_k(n) e^{-j\omega nT} \right|^2}, \quad (5.45)$$

а уравнение, которое необходимо решить для оценки значений частот, оказывается следующим:

$$\sum_{k=M+1}^N \frac{1}{\lambda_k} \sum_{n=0}^{N-1} \sum_{l=0}^{N-1} v_k(n) v_k^*(l) e^{-j\omega(n-l)T} = 0. \quad (5.46)$$

В этих формулах  $\lambda_k$  — собственное число, соответствующее  $k$ -му собственному вектору  $v_k$ .

В [12] приводятся сведения о том, что при заданном значении порядка  $M$  метод EV порождает меньше ложных спектральных пиков, чем MUSIC, и, как правило, лучше передает форму спектра шума.

Еще раз подчеркнем, что псевдоспектры (5.43) и (5.45) не являются оценками истинного спектра плотности мощности, а представляют собой лишь спектральные псевдооценки, позволяющие оценивать частоты синусоидальных или узкополосных составляющих сигнала с разрешением, несколько превосходящим разрешение авторегрессионных методов.

## Функции спектрального анализа в MATLAB

Для выполнения спектрального анализа в MATLAB имеется большое число функций, реализующих разнообразные методы — от обычного ДПФ до MUSIC.

## Прямое и обратное ДПФ

Для выполнения прямого и обратного ДПФ в MATLAB служат функции `fft` и `ifft`:

- $y = \text{fft}(x)$  — вычисляет прямое ДПФ для вектора  $x$ ; если  $x$  — матрица, преобразование производится для каждого ее столбца по отдельности;
- $y = \text{fft}(x, N)$  — предварительно приводит исходные данные к размеру  $N$ , урезая их или дополняя нулями;
- $x = \text{ifft}(y)$  и  $x = \text{ifft}(y, N)$  — аналогичные варианты вызова для функции обратного ДПФ.

Функции `fft` и `ifft` входят в ядро системы MATLAB. Вычисления реализованы с использованием библиотеки FFTW ([www.fftw.org](http://www.fftw.org)), включающей в себя большое количество алгоритмов вычисления ДПФ и позволяющей оптимизировать вычислительные затраты при любой длине обрабатываемого сигнала.

## Функция `fftshift`

Элементы вектора, возвращаемого функцией `fft`, соответствуют частотам, равномерно распределенным в диапазоне от нуля и почти до частоты дискретизации. Первый элемент, таким образом, соответствует нулевой частоте, а последний — частоте, меньшей частоты дискретизации на  $f_d/N$ , где  $N$  — размер входного и выходного векторов. При выводе спектральных графиков иногда желательно, чтобы нулевая частота находилась в центре, а диапазон отображаемых частот простирался от  $-f_d/2$  до  $f_d/2$ . Сделать это позволяет функция `fftshift`, которая меняет местами половины переданного ей вектора:

```
y = fftshift(x)
```

Продемонстрируем действие функции `fftshift` на примере двух коротких векторов четной и нечетной длины:

```
>> fftshift([1 2 3 4])
ans =
    3     4     1     2
>> fftshift([1 2 3 4 5])
ans =
    4     5     1     2     3
```

Как видите, в случае четной длины действительно происходит перестановка половины входного вектора (при этом первый элемент результирующего вектора соответствует частоте Найквиста). В случае нечетной длины перестановка выполняется так, чтобы первый элемент, соответствующий нулевой частоте, стал *средним* элементом результирующего вектора.

Разумеется, функцию `fftshift` можно использовать не только для вывода спектральных графиков, но и в других случаях, когда требуется поменять местами половиной вектора.

## Блоковая фильтрация в частотной области

Для реализации блоковой фильтрации с помощью БПФ в MATLAB используется функция `fftfilt`, имеющая следующий синтаксис:

```
y = fftfilt(h, x, n)
```

Здесь `h` — импульсная характеристика фильтра, `x` — входной сигнал, `n` — размерность БПФ (точнее, размерность БПФ при задании этого параметра определяется так: `Nfft = 2^nextpow2(n)`, т. е. `n` округляется вверх до степени двойки).

Выходной параметр `y` — результат фильтрации.

Третий входной параметр `n` при вызове можно опускать, тогда размерность БПФ будет выбираться автоматически, исходя из максимальной эффективности вычислений (минимального числа операций).

## Окна

MATLAB содержит (в пакете Signal Processing) целый ряд стандартных *весовых функций*. Они возвращают векторы отсчетов, которые могут использоваться в качестве одного из параметров разнообразных функций непараметрического спектрального анализа.

Все рассматриваемые ниже функции принимают в качестве параметра требуемую длину вектора (`n`), которая должна быть целым положительным числом, и возвращают вектор-столбец `w`. При `n=1` все функции возвращают значение 1.

В приводимых далее формулах для отсчетов окон  $w(k)$  предполагается, что  $k$  лежит в диапазоне от единицы до  $n$ .

Амплитудный спектр весовой функции соответствует частотной характеристике нулевого канала ДПФ при использовании данной весовой функции.

Графики окон и их амплитудных спектров будут строиться для  $n = 32$  с помощью специальной функции визуализатора окон `wvtool`. Эта функция будет рассмотрена при описании первого окна. Фазовый спектр для всех окон линейно зависит от частоты, поэтому его графики не представляют интереса.

Форма большинства окон и их амплитудных спектров примерно одинакова, отличия состоят лишь в количественных параметрах. Поэтому графики будут приведены лишь для некоторых окон.

### Прямоугольное окно

Функция `rectwin`, реализующая "прямоугольное окно", введена в MATLAB лишь для полноты набора весовых функций, поскольку она соответствует отсутствию взвешивания:

```
w = rectwin(n)
```

Возвращаемый вектор заполнен единицами: `w = ones(n,1)`.

Приводить график данной весовой функции, в принципе, не имеет смысла, а ее амплитудный спектр (смещенный по горизонтали) для  $n = 8$  был показан ранее на рис. 5.9. Однако на этом примере мы продемонстрируем вызов визуализатора окон, который будет многократно использован на последующих страницах.

## Визуализатор окон

Для одновременного просмотра оконных функций во временной и частотной областях предназначен *визуализатор окон*, реализованный в виде функции `wvtool` (*Window Visualization Tool*). При ее вызове необходимо указать в списке входных параметров вектор отсчетов окна или несколько таких векторов:

```
wvtool(w)
wvtool(w1, w2, ...)
```

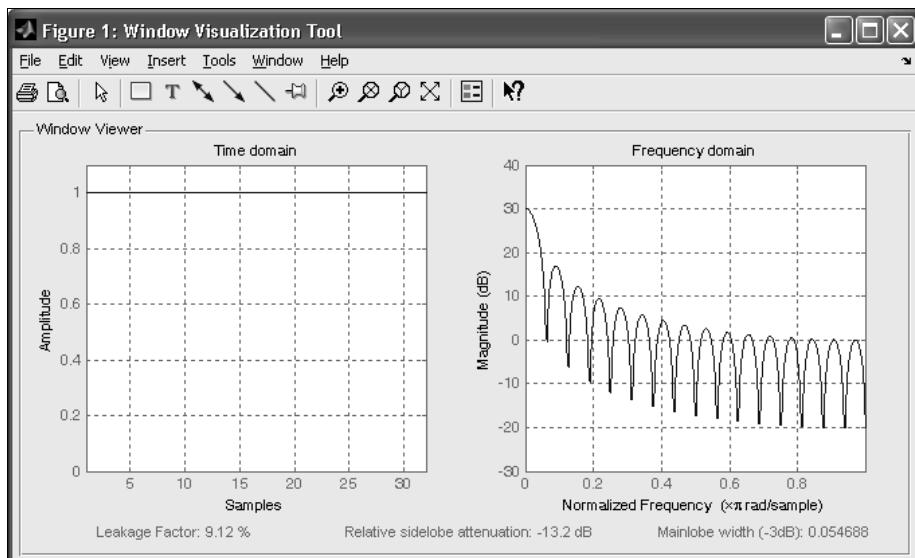
В качестве примера вызовем функцию `wvtool` для просмотра прямоугольного окна 32-го порядка:

```
>> wvtool(rectwin(32))
```

Появится окно, показанное на рис. 5.18.

В левой части выводится график окна во временной области, а в правой — график его амплитудного спектра. Под графиками отображается количественная информация об окне:

- Leakage Factor** — коэффициент утечки, который показывает, какая доля общей энергии окна сосредоточена в боковых лепестках его спектра;
- Relative sidelobe attenuation** — уровень максимального из боковых лепестков спектра относительно спектральной функции на нулевой частоте (т. е. относительно главного лепестка);



**Рис. 5.18.** Просмотр характеристик окна с помощью функции `wvtool`

□ **Mainlobe width (-3dB)** — ширина главного лепестка по уровню  $-3$  дБ. Здесь имеется в виду полная ширина главного лепестка (с учетом отрицательных частот), так что значение нормированной частоты, которое на графике соответствует уровню  $-3$  дБ от максимума, будет в два раза меньше. Для наглядности можно сравнить эту величину с нормированным расстоянием между частотами соседних каналов ДПФ, равным  $2/n$  ( $n$  — длина окна). В приводимых в этом разделе примерах это составляет  $2/32 = 0,0625$ .

Итак, прямоугольное окно обеспечивает уровень боковых лепестков  $-13,2$  дБ, в боковых лепестках спектра сосредоточено  $9,12\%$  общей мощности окна, а нормированная ширина главного лепестка составляет  $0,055$ .

Следует иметь в виду, что оценка параметров окна производится функцией `wvtool` по дискретной сетке частот анализа, число точек которой по умолчанию равно  $512$ . Иногда этого может быть слишком мало для получения достаточно точных результатов. В таком случае следует воспользоваться командой меню **View | Analysis Parameters** и задать большее число точек в поле ввода **Number of Points** появившегося окна диалога. Назначение остальных элементов интерфейса окна функции `wvtool` и упомянутого окна параметров анализа самоочевидно и в пояснениях не нуждается.

### ЗАМЕЧАНИЕ

В дальнейшем изображения окна WVTool будут приводиться в черно-белом виде и без элементов интерфейса.

## Треугольное окно

Функция `triang` реализует треугольное окно:

`w = triang(n)`

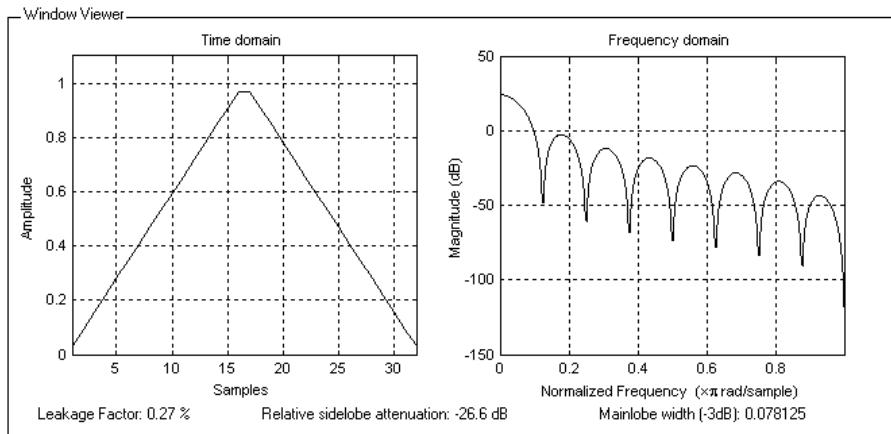
Отсчеты треугольного окна рассчитываются по следующей формуле:

$$w(k) = 1 - \frac{|2k - n - 1|}{n + 1}, \quad k = 1, 2, \dots, n \text{ — для нечетного } n,$$

$$w(k) = 1 - \frac{|2k - n - 1|}{n}, \quad k = 1, 2, \dots, n \text{ — для четного } n.$$

При любом  $n$  треугольное окно является симметричным. При нечетном  $n$  крайние значения треугольного окна (при  $k = 1$  и  $k = n$ ) равны  $2/(n + 1)$ , а в середине окна (при  $k = (n + 1)/2$ ) достигается единичное значение. При четном  $n$  крайние значения окна равны  $1/n$ , а максимальное значение, достигаемое в середине окна (при  $k = n/2$  и  $k = n/2 + 1$ ), равно  $(n - 1)/n$ .

На рис. 5.19 приведены графики треугольного окна и его амплитудного спектра при  $n = 32$ .



**Рис. 5.19.** Треугольное окно (слева) и его амплитудный спектр (справа)

## Окно Бартлетта

Функция `bartlett(n)` реализует окно Бартлетта:

```
w = bartlett(n)
```

Окно Бартлетта, по сути, тоже является треугольным, но рассчитывается несколько иначе:

$$w(k) = 1 - \frac{|2k - n - 1|}{n - 1}, \quad k = 1, 2, \dots, n. \quad (5.47)$$

В отличие от треугольного окна, значения окна Бартлетта по краям (при  $k = 1$  и  $k = n$ ) равны нулю. Окно Бартлетта представляет собой отсчеты симметричного треугольного импульса, который начинается при  $k = 1$ , заканчивается при  $k = n$  и имеет единичную амплитуду. Максимум значения этого импульса достигается при  $k = n/2$ , поэтому при нечетном  $n$  окно Бартлетта не достигает единичного значения в середине.

При нечетном  $n$  ненулевые отсчеты окна Бартлетта совпадают с отсчетами треугольного окна длины  $n - 2$ .

## Косинусные окна

Данный термин обозначает окна, которые представляют собой сумму нескольких гармонических составляющих, периоды которых укладываются на длине окна цеплюе число раз. К этой категории относятся следующие весовые функции, реализованные в пакете Signal Processing:

- **окно Ханна** — функция `hann` (по аналогии с окном Хэмминга его часто ошибочно называют *окном Хэннинга*; даже соответствующая функция в ранних версиях пакета Signal Processing имела имя `hanning`):

```
w = hann(n, 'sflag')
```

- окно Хэмминга — функция `hamming`:

```
w = hamming(n, 'sflag')
```

- окно Блэкмена — функция `blackman`:

```
w = blackman(n, 'sflag')
```

- окно Блэкмена — Харриса — функция `blackmanharris`:

```
w = blackmanharris(n)
```

- окно Наттолла (версия окна Блэкмена — Харриса, предложенная Наттоллом) — функция `nuttallwin`:

```
w = nuttallwin(n)
```

- окно с плоской вершиной (*flat top window*) — функция `flattopwin`:

```
w = flattopwin(n, 'sflag')
```

Строковый параметр '`sflag`' позволяет выбрать режим расчета окна. При значении '`'symmetric'`', принятом по умолчанию, генерируется симметричное окно, для которого  $w(k) = w(n+1-k)$ . При значении '`'periodic'`' создается слегка несимметричное окно, синусоидальные компоненты которого будут аккуратно стыковаться при соединении нескольких экземпляров окна.

### **ЗАМЕЧАНИЕ**

Как видно из приведенных образцов синтаксиса, не все косинусные окна поддерживают параметр '`sflag`'. В чем причина такого отсутствия единобразия, неясно — возможно, дело всего лишь в том, что разные функции реализовывали разные программисты. Собственно расчет перечисленных окон производится с помощью двух разных функций, расположенных в папке **private** пакета Signal Processing — функция `gencoswin` получает в качестве параметра имя окна и поддерживает параметр '`sflag`', а функция `min4termwin` берет вектор весовых коэффициентов при косинусоидальных слагаемых и не поддерживает дополнительных параметров.

В симметричном случае отсчеты косинусных окон рассчитываются по формуле:

$$w(k) = a_0 + a_1 \cos\left(2\pi \frac{k-1}{n-1}\right) + a_2 \cos\left(4\pi \frac{k-1}{n-1}\right) + a_3 \cos\left(6\pi \frac{k-1}{n-1}\right) + a_4 \cos\left(8\pi \frac{k-1}{n-1}\right).$$

Для периодического варианта  $n-1$  в знаменателях формулы заменяется на  $n$  (возможна и иная трактовка: выполняется расчет по приведенной формуле для окна длиной  $n+1$ , а затем последний элемент отбрасывается).

Значения весовых коэффициентов  $a_k$  при косинусных слагаемых различных окон приведены в табл. 5.3, а графики некоторых из этих окон и их амплитудных спектров при  $n=32$  представлены на рис. 5.20—5.22.

Из рис. 5.22 видно, что для окна с плоской вершиной встроенный в функцию `wvtool` алгоритм оказался не в состоянии оценить ни уровень боковых лепестков, ни долю сосредоточенной в них мощности. Оценка, произведенная вручную, показывает,

Таблица 5.3. Коэффициенты косинусных окон

Окно	$a_0$	$a_1$	$a_2$	$a_3$	$a_4$
Ханна	0,5	-0,5	0	0	0
Хэмминга	0,54	-0,46	0	0	0
Блэкмена	0,42	-0,5	0,08	0	0
Блэкмена — Харриса	0,35875	-0,48829	0,14128	-0,01168	0
Наттолла	0,3635819	-0,4891775	0,1365995	-0,0106411	0
С плоской вершиной	0,2156	-0,416	0,2781	-0,0836	0,0069

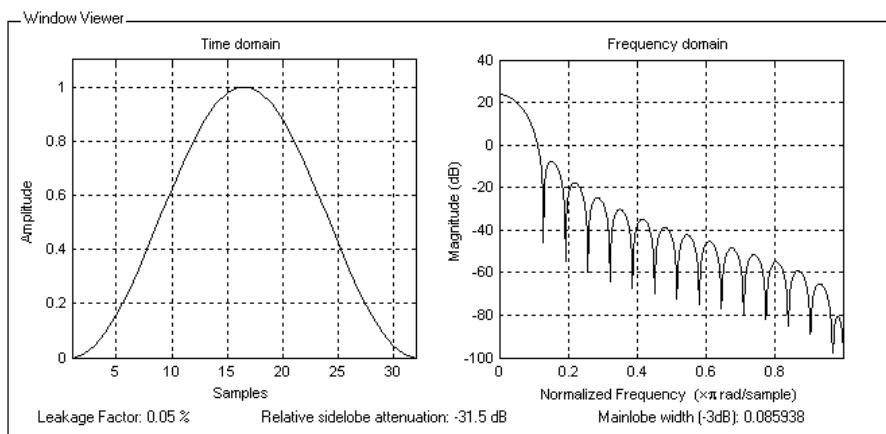


Рис. 5.20. Окно Ханна (слева) и его амплитудный спектр (справа)

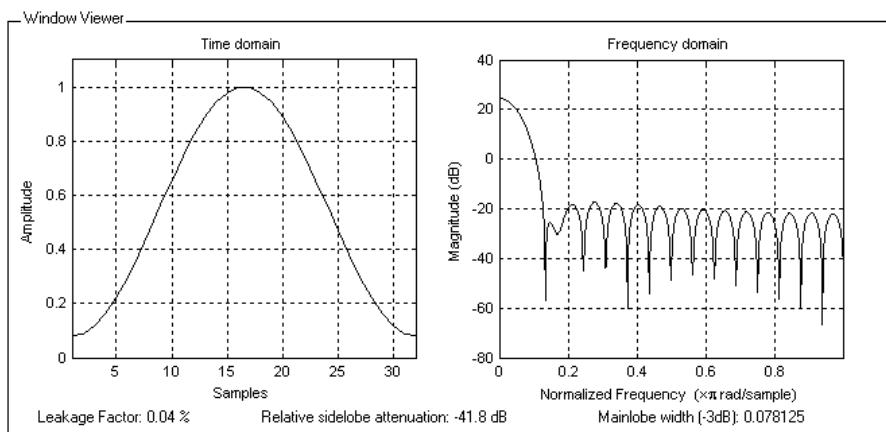
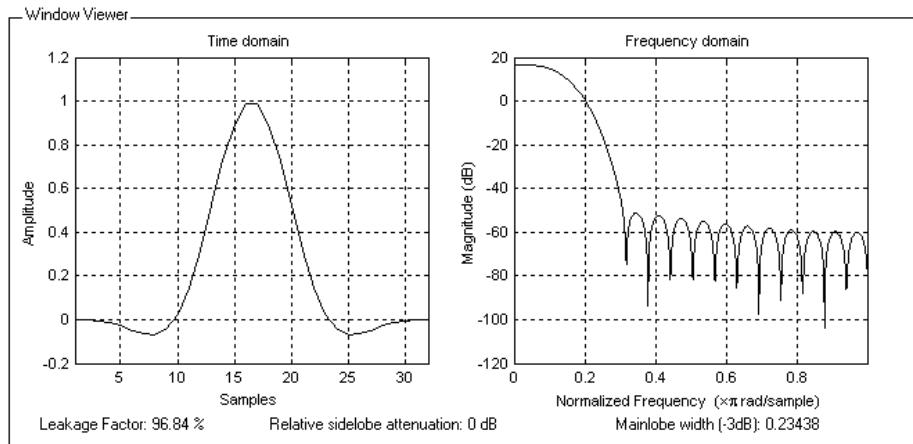


Рис. 5.21. Окно Хэмминга (слева) и его амплитудный спектр (справа)



**Рис. 5.22.** Окно с плоской вершиной (слева) и его амплитудный спектр (справа)

что в боковых лепестках сосредоточено менее чем  $2 \cdot 10^{-5}$  общей мощности окна, а их относительный уровень равен  $-67,7$  дБ. Данное окно во многих отношениях является исключением из общего правила — во-первых, оно является знакопеременным, а во-вторых, по соотношению "ширина главного лепестка" (параметру, наиболее важному при использовании окон для спектрального анализа) оно безнадежно уступает другим (см. далее сравнение параметров в табл. 5.4). Зато вершина главного лепестка этого окна имеет практически плоский вид (это можно увидеть, увеличив соответствующий фрагмент спектра в окне WVTool), что и обусловило его название.

## Окно Бартлетта — Ханна

Функция `barthannwin` реализует окно Бартлетта — Ханна:

```
w = barthannwin(n)
```

Это окно представляет собой линейную комбинацию окон Бартлетта (5.47) и Ханна (см. ранее разд. "Косинусные окна" в этой главе) и рассчитывается по следующей формуле:

$$w(k) = 0,62 - 0,48 \left| \frac{k-1}{n-1} - 0,5 \right| - 0,38 \cos \left( 2\pi \frac{k-1}{n-1} \right).$$

## Окно Бомена

Функция `bohmanwin` реализует окно Бомена:

```
w = bohmanwin(n)
```

Окно Бомена представляет собой свертку двух одинаковых косинусоидальных импульсов. После упрощений эта формула сводится к произведению косинусоидаль-

ной и треугольной функций с дополнительным синусоидальным слагаемым, обеспечивающим нулевое значение первой производной по краям окна:

$$w(k) = \left(1 - \left|\frac{k-1-n/2}{n/2}\right|\right) \cos\left(\pi \frac{k-1-n/2}{n/2}\right) + \frac{1}{\pi} \sin\left(\pi \left|\frac{k-1-n/2}{n/2}\right|\right).$$

## Окно Тьюки

Функция `tukeywin` реализует окно Тьюки:

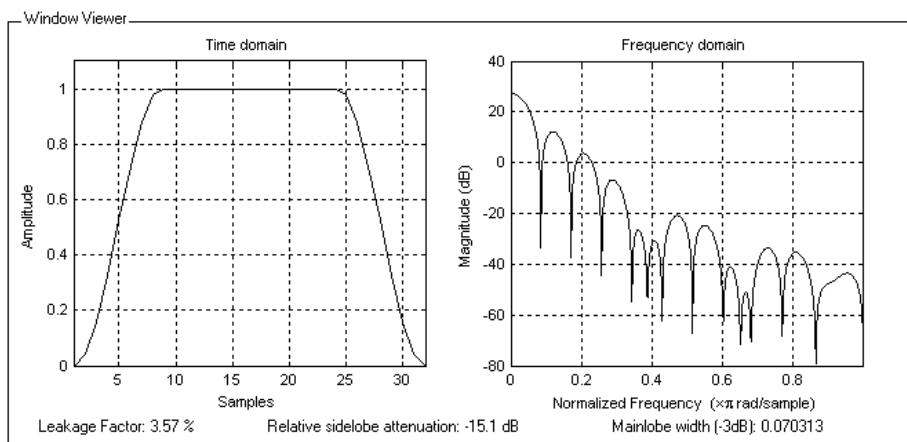
```
w = tukeywin(n, r)
```

Данное окно представляет собой прямоугольник с косинусоидально сглаженными краями. Второй входной параметр `r` задает ширину зоны сглаживания. Расчет коэффициентов окна производится по следующей формуле:

$$w(k) = \begin{cases} \frac{1}{2} \left(1 - \cos\left(\frac{2\pi(k-1)}{(n-1)r}\right)\right), & 1 \leq k \leq \frac{(n-1)r}{2} + 1, \\ 1, & \frac{(n-1)r}{2} + 1 \leq k \leq n - \frac{(n-1)r}{2}, \\ \frac{1}{2} \left(1 - \cos\left(\frac{2\pi(k-n)}{(n-1)r}\right)\right), & n - \frac{(n-1)r}{2} \leq k \leq n. \end{cases}$$

При  $r = 0$  окно Тьюки превращается в прямоугольное окно (см. ранее описание функции `rectwin`), а при  $r = 1$  — в окно Ханна (см. ранее описание функции `hann` в разд. "Косинусные окна" этой главы). При вызове функции параметр `r` можно опустить; его значение по умолчанию равно 0,5.

На рис. 5.23 приведены графики окна Тьюки и его амплитудного спектра при  $n = 32$  и  $r = 0,5$ .



**Рис. 5.23.** Окно Тьюки (слева) и его амплитудный спектр (справа) при  $r = 0,5$

## Окно Кайзера

Функция `kaiser` реализует окно Кайзера:

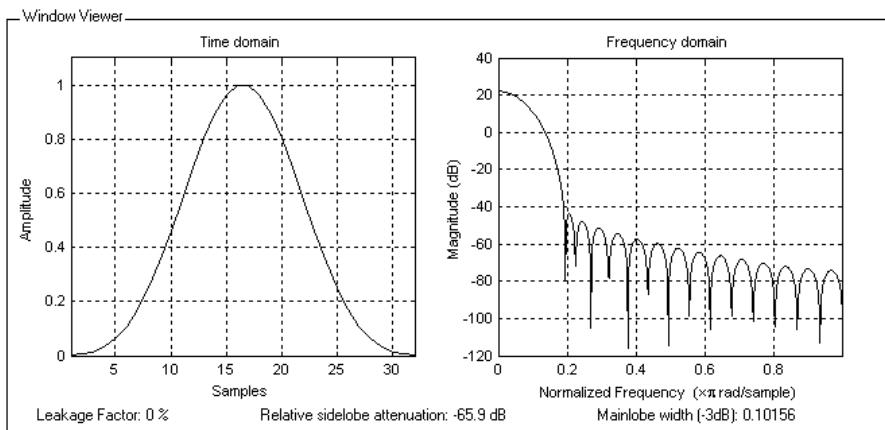
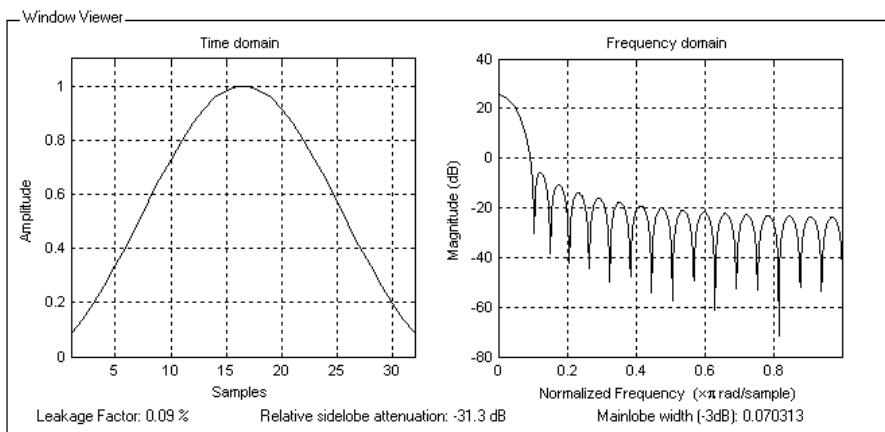
```
w = kaiser(n, beta)
```

Здесь `beta` — параметр окна (см. формулу далее).

Отсчеты окна Кайзера рассчитываются по формуле

$$w(k) = \frac{I_0\left(\beta \sqrt{1 - \left(\frac{2k-n-1}{n-1}\right)^2}\right)}{|I_0(\beta)|}, \quad k = 1, 2, \dots, n.$$

Здесь  $I_0(x)$  — модифицированная функция Бесселя первого рода нулевого порядка. Чем больше  $\beta$ , тем больше доля энергии, сосредоточенная в главном лепестке



**Рис. 5.24.** Окно Кайзера (слева) и его амплитудный спектр (справа) для  $\beta = 4$  (сверху) и  $\beta = 9$  (снизу)

спектра (и тем шире этот главный лепесток), и тем меньше уровень боковых лепестков. На практике чаще всего используются значения  $\beta$  от 4 до 9.

На рис. 5.24 приведены графики окна Кайзера и его амплитудного спектра при  $n = 32$  для двух указанных крайних значений  $\beta$ .

## Окно Чебышева

Функция `chebwin` реализует окно Чебышева:

```
w = chebwin(n, beta)
```

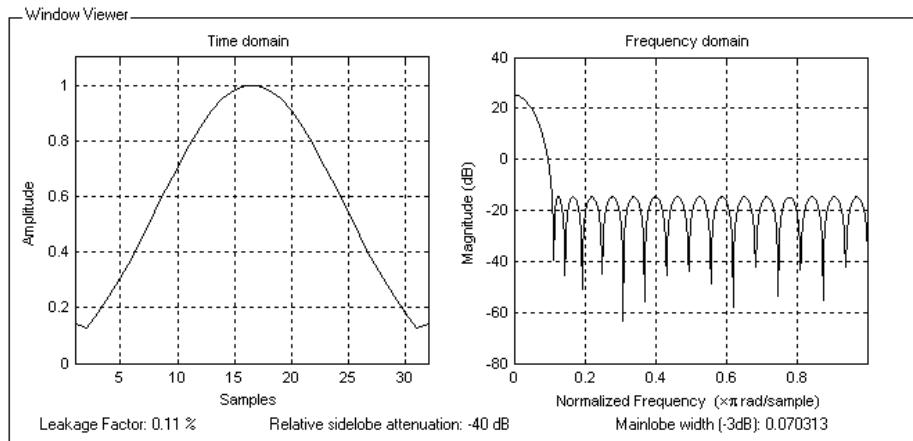
Здесь `beta` — степень подавления боковых лепестков в децибелах (см. формулу далее). При вызове функции этот параметр можно опустить; его значение по умолчанию равно 100 дБ.

Для окна Чебышева все боковые лепестки имеют одинаковый, заданный при расчете окна уровень. Отсчеты окна Чебышева рассчитываются путем вычисления обратного преобразования Фурье от его частотной характеристики:

$$\hat{S}(\omega) = \frac{\cos\left((n-1)\arccos\left(\alpha \cos\left(\pi \frac{\omega}{\omega_d}\right)\right)\right)}{\operatorname{ch}\left((n-1)\operatorname{arch}(\alpha)\right)}, \text{ где } \alpha = \operatorname{ch}\left(\frac{\operatorname{arch}(10^{\beta/20})}{n-1}\right).$$

Здесь  $\beta$  — степень подавления боковых лепестков в децибелах,  $n$  — требуемое количество отсчетов окна.

На рис. 5.25 приведены графики окна Чебышева и его амплитудного спектра при  $n = 32$  для уровня боковых лепестков  $-40$  дБ.



**Рис. 5.25.** Окно Чебышева (слева) и его амплитудный спектр (справа) для уровня боковых лепестков  $-40$  дБ

## Гауссово окно

Функция `gausswin` реализует гауссово окно:

`w = gausswin(n, alpha)`

Гауссово окно, как явствует из его названия, имеет вид функции  $\exp(-x^2)$ . Расчет коэффициентов окна производится по следующей формуле:

$$w(k) = \exp\left(-\frac{1}{2}\left(\alpha \frac{k - 1 - n/2}{n/2}\right)^2\right).$$

Параметр `alpha` (в формуле —  $\alpha$ ) регулирует эффективную ширину окна — чем меньше его величина, тем шире окно. При вызове функции данный параметр можно опустить; его значение по умолчанию равно 2,5.

## Окно Парзена

Функция `parzenwin` реализует окно Парзена:

`w = parzenwin(n)`

Данное окно представляет собой кусочно-кубическую аппроксимацию гауссова окна (при  $\alpha \approx 3,3$ ). Уровень боковых лепестков окна Парзена уменьшается пропорционально четвертой степени частоты. Расчет коэффициентов окна производится по следующей формуле:

$$w(k) = \begin{cases} 2\left(1 - \frac{|k - (n+1)/2|}{n/2}\right)^3, & 1 \leq k \leq \frac{n+2}{4}, \quad \frac{3n+2}{4} \leq k \leq n, \\ 1 - 6\left(\frac{k - (n+1)/2}{n/2}\right)^2\left(1 - \frac{|k - (n+1)/2|}{n/2}\right), & \frac{n+2}{4} \leq k \leq \frac{3n+2}{4}. \end{cases}$$

## Сводка параметров окон

В заключение рассмотрения оконных функций приведем сводку их параметров (для  $n = 32$ ), отображаемых в нижней части окна функции `wvtool`. Эти результаты представлены в виде табл. 5.4 и рис. 5.26.

**Таблица 5.4.** Параметры оконных функций

Окно	Коэффициент утечки, %	Уровень боковых лепестков, дБ	Нормированная ширина главного лепестка по уровню -3 дБ
Прямоугольное	9,12	-13,2	0,055176
Треугольное	0,27	-26,6	0,079346
Бартлетта	0,28	-26,5	0,082031
Хана	0,05	-31,5	0,092773

Таблица 5.4 (окончание)

Окно	Коэффициент утечки, %	Уровень боковых лепестков, дБ	Нормированная ширина главного лепестка по уровню $-3$ дБ
Хэмминга	0,04	-41,8	0,082764
Блэкмена	0	-58,2	0,105710
Блэкмена — Харриса	0	-92,0	0,122310
Наттолла	0	-88,8	0,120360
С плоской вершиной	0	-67,7	0,240480
Бартлетта — Хана	0,03	-35,8	0,089844
Бомена	0	-46,0	0,109620
Тьюки при $\alpha = 0,5$	3,57	-15,1	0,073975
Кайзера при $\beta = 4$	0,09	-31,3	0,076416
Кайзера при $\beta = 9$	0	-65,9	0,107420
Чебышева при $\beta = 40$ дБ	0,11	-40	0,076904
Гауссово при $\alpha = 2,5$	0,01	-43,1	0,085449
Парзена	0	-53,1	0,113530

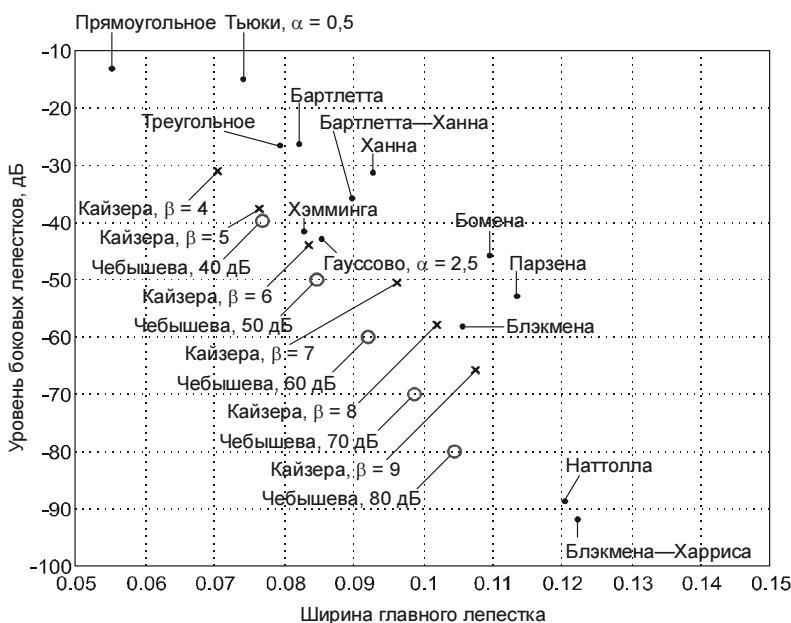


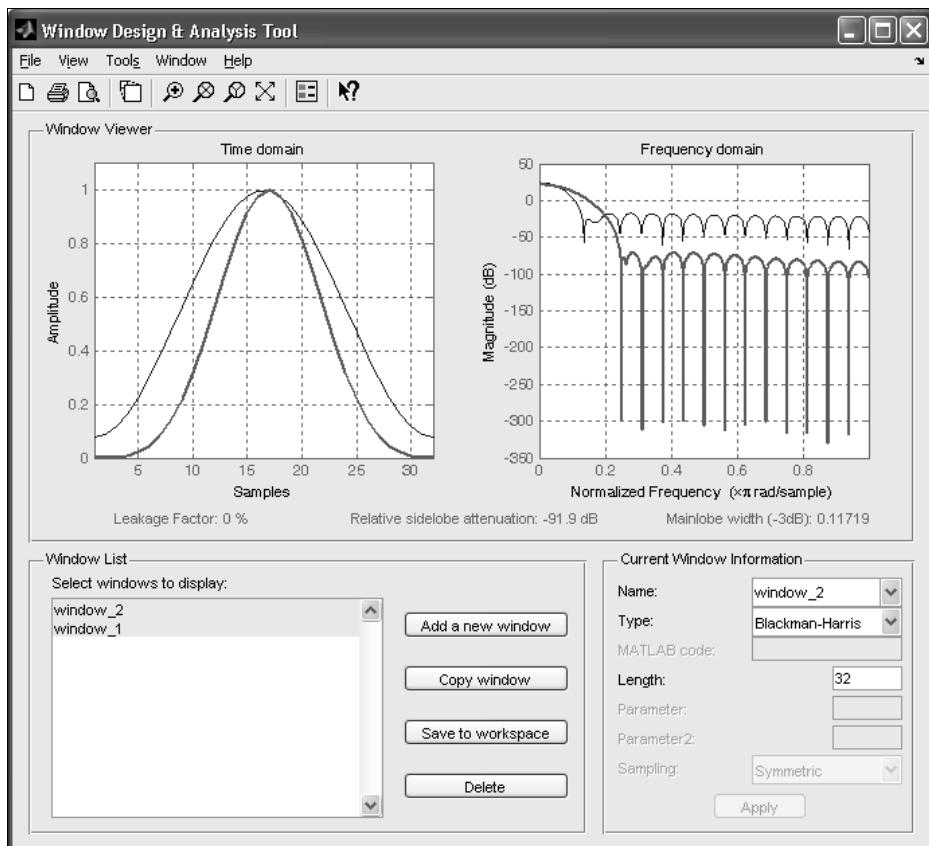
Рис. 5.26. Соотношение между шириной главного лепестка и уровнем боковых лепестков для различных окон

Нулевой коэффициент утечки в табл. 5.4 означает, что его значение не превышает 0,005%. Нормированную ширину главного лепестка полезно сравнить с нормированным расстоянием между частотными каналами ДПФ, которое при размерности 32 составляет  $2/32 = 0,0625$ . Приведенные данные получены с помощью функции `wvtool` при числе точек частотного анализа, равном 16384.

Следует иметь в виду, что указанные параметры зависят от длины рассматриваемого окна — этим, как правило, и объясняется различие между характеристиками окон, приводимых в разных источниках. Например, для окна Наттолла при переходе от  $n = 32$  к  $n = 256$  уровень боковых лепестков уменьшается более чем на 9 дБ — от  $-88,8$  до  $-97,9$  дБ.

## Среда анализа и расчета окон

Помимо функции `wvtool`, предназначеннной лишь для просмотра окон, в пакете Signal Processing имеется еще функция `wintool`, которая не только отображает временные и частотные характеристики, но и реализует полноценную среду анализа и расчета окон (*Window Design & Analysis Tool*) с графическим пользовательским интерфейсом.



**Рис. 5.27.** Пользовательский интерфейс среды анализа и расчета окон

Для вызова среды необходимо набрать имя соответствующей функции в командной строке:

```
>> wintool
```

Появится окно, показанное на рис. 5.27. Его верхняя половина выглядит так же, как показанное ранее на рис. 5.18 окно функции `wvtool`, а в нижней части расположены элементы управления, позволяющие рассчитывать окна и управлять их отображением.

Назначение элементов интерфейса не нуждается в особых комментариях, скажем лишь, что для одновременного просмотра характеристик нескольких окон нужно выделить их обычными средствами Windows (щелчок мышью при нажатой клавише `<Shift>` или `<Ctrl>`) в списке **Window List**. В качестве примера на рис. 5.27 сравниваются характеристики окна Хэмминга 32-го порядка и окна Блэкмена — Харриса, также 32-го порядка.

## Функции непараметрического спектрального анализа

В данном разделе будут рассмотрены три функции, осуществляющие спектральный анализ сигнала без использования дополнительной информации:

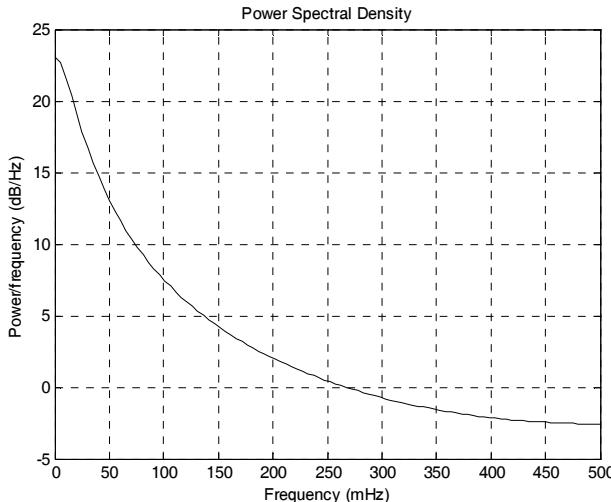
- `spectrogram` — вычисление мгновенного спектра сигнала;
- `periodogram` — вычисление *спектральной плотности мощности* (СПМ) одной реализации случайного сигнала;
- `pwelch` — оценка спектральной плотности мощности случайного процесса методом усреднения модифицированных периодограмм.

Для демонстрации работы многих функций спектрального анализа будет использоваться дискретный случайный процесс с экспоненциальной функцией корреляции. Формирование такого процесса было рассмотрено в разд. "Формирование случайных сигналов" главы 3, поэтому соответствующие строки листингов далее будут приводиться без комментариев.

Чтобы в дальнейшем было с чем сравнивать получаемые результаты, построим теоретический спектр плотности мощности данного сигнала. Поскольку он является авторегрессионным процессом первого порядка, его спектр можно рассчитать по формуле (5.25), положив в ней  $N=1$ . Коэффициент  $a_1$ , как и в примере главы 3, принимаем равным 0,9, а частоту дискретизации  $f_d$  и дисперсию белого шума  $\sigma_n^2$  считаем равными единице. Функции спектрального анализа пакета Signal Processing для вещественного сигнала рассчитывают *одностороннюю* СПМ, поэтому умножим результат вычислений по формуле (5.25) на два. Строим график СПМ (рис. 5.28):

```
>> T = 1; % период дискретизации
>> var_n = 1; % дисперсия белого шума
>> a1 = 0.9; % параметр экспоненциальной корреляции
>> f = linspace(0, 1/(2*T), 100); % вектор значений частот
```

```
>> % рассчитываем спектр плотности мощности
>> P = 2 * var_n * T ./ abs(1 - a1 * exp(-2i*pi*f*T)).^2;
>> Hpsd = dspdata.psd(P, f, 'Fs', 1/T); % объект для хранения спектра
>> plot(Hpsd) % вывод графика СПМ
```



**Рис. 5.28.** СПМ экспоненциально коррелированного дискретного случайного процесса

## Спектrogramма

Спектrogramмой (*spectrogram*) сигнала называется его *мгновенный спектр*, зависящий от времени. Для вычисления спектrogramмы вектор сигнала разбивается на фрагменты (возможно, с перекрытием). Для каждого фрагмента вычисляется спектр с помощью функции *fft*. Набор спектров всех фрагментов и образует спектrogramму.

Для вычисления спектrogramмы служит функция *spectrogram*:

```
[S, F, T, P] = spectrogram(x, w, nooverlap, nfft, fs)
```

Здесь *x* — вектор сигнала (сигнал не может быть многоканальным), *w* — длина используемого окна или вектор его отсчетов (длина этого вектора определяет размер фрагментов, на которые разбивается анализируемый сигнал), *noverlap* — величина перекрытия соседних фрагментов (в отсчетах), *nfft* — используемая размерность дискретного преобразования Фурье, *fs* — частота дискретизации в герцах.

Обязательным входным параметром является только вектор значений сигнала *x*, остальные параметры имеют значения по умолчанию, которые используются, если в качестве параметра указана пустая матрица [] или если несколько последних параметров при вызове опущены:

- *w* = *hamming*(*nw*, 'periodic') — длина окна *nw* выбирается так, чтобы при делении сигнала на фрагменты число фрагментов оказалось равным 8;

- nooverlap = length(w)/2;
- nfft = max(256, 2^nextpow2(length(w)));
- fs = 2;

Выходной параметр  $S$  — матрица, столбцы которой являются комплексными спектрами отдельных фрагментов сигнала. Если входной вектор  $x$  является вещественным, спектр вычисляется только для положительных частот, и возвращаемая матрица  $S$  содержит  $nfft/2+1$  (для четного  $nfft$ ) или  $(nfft+1)/2$  (для нечетного  $nfft$ ) строк. В случае комплексного анализируемого вектора  $x$  спектр не обладает симметрией, и возвращаемая матрица  $S$  содержит  $nfft$  строк.

Следующие два выходных параметра,  $F$  и  $T$ , являются векторами, содержащими соответственно частотную и временную шкалы спектrogramмы. Длина вектора  $F$  равна числу строк матрицы  $S$ , а длина вектора  $T$  — числу ее строк.

Последний выходной параметр — матрица  $P$  — содержит спектры *мощности* фрагментов сигнала. Она связана с матрицей комплексных спектров  $S$  следующим образом:

```
P = abs(S).^2/sum(w.^2);
```

При отсутствии выходных параметров функция `spectrogram` строит графическое изображение спектrogramмы в координатах "время — частота", отображая значения спектра мощности с помощью цвета. В этом случае можно использовать дополнительный входной строковый параметр '`freqloc`', определяющий направление частотной оси на графике:

```
spectrogram(..., 'freqloc')
```

Возможные значения данного параметра следующие:

- '`xaxis`' — частотная ось направлена по горизонтали;
- '`yaxis`' — частотная ось направлена по вертикали.

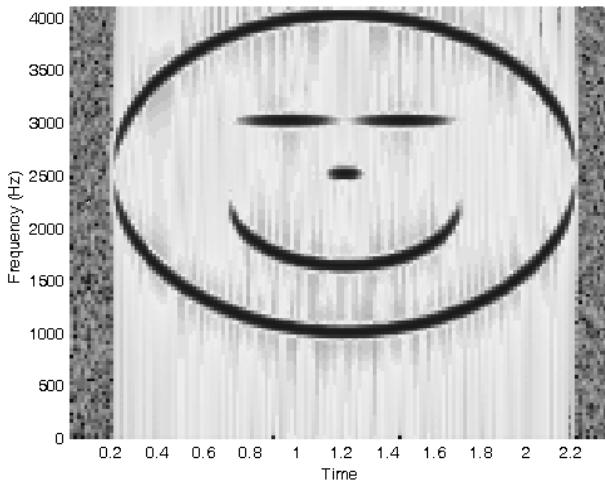
Примеры использования функции `spectrogram` уже приводились в главе 3 (см. рис. 3.29—3.31). Поэтому здесь мы лишь покажем, какой забавный вид может иметь спектrogramма сигнала, сформированного из нескольких импульсов с меняющейся частотой. Запись такого сигнала содержится в файле `vcosig.mat`, входящем в состав пакета Signal Processing. В этом же файле хранится и значение частоты дискретизации этого сигнала (переменная `fs`). Формируем спектrogramму (рис. 5.29):

```
>> load vcosig
>> spectrogram(vcosig, 256, [], [], fs, 'yaxis')
```

### **ВНИМАНИЕ!**

Для отображения спектrogramмы используется цветовая палитра `jet`, так что минимальному уровню амплитудного спектра соответствует темно-синий цвет, максимальному — темно-красный. Переход от синего к красному происходит через оттенки голубого, зеленого и желтого. В результате при черно-белой печати зависимость оттенка графика от уровня спектра оказывается немонотонной. Для правильной пе-

передачи уровня спектра в черно-белом виде следует установить для рисунка палитру оттенков серого командой `colormap gray`. В данном примере этого не сделано только потому, что " рожица" выглядит выразительнее именно после преобразования исходных цветов рисунка в оттенки серого, а не при использовании серой палитры для вывода спектограммы.



**Рис. 5.29.** Пример использования функции `spectrogram`

## Расчет периодограммы

Для вычисления периодограммы в MATLAB предназначена функция `periodogram`. Синтаксис ее вызова следующий:

```
[Pxx, f] = periodogram(x, w, nfft, fs, 'range')
```

Единственным обязательным входным параметром является `x` — вектор отсчетов сигнала. Остальные параметры имеют значения по умолчанию, которые используются, если в качестве параметра указана пустая матрица `[]` или если некоторое количество параметров (начиная с последнего) опущены при вызове.

Вектор `w` должен содержать коэффициенты используемого окна (при этом вычисляется модифицированная периодограмма по формуле (5.24)). По умолчанию используется прямоугольное окно, т. е. периодограмма вычисляется по формуле (5.23).

Параметр `nfft` задает размерность БПФ, используемого для вычисления периодограммы. По умолчанию этот параметр равен `max(256, 2^nextpow2(length(x)))`. Входной сигнал, умноженный на окно, приводится к размеру `nfft` (обрезается либо дополняется нулями).

Параметр `fs` — частота дискретизации в герцах. Эта величина используется в формулах (5.23) и (5.24), а также для оцифровки графика и расчета возвращаемого вектора частот `f`. Значение по умолчанию равно  $2\pi$ .

Строковый параметр 'range' определяет частотный диапазон для возвращаемого вектора  $P_{xx}$ . Возможны два значения:

- 'twosided' — векторы  $P_{xx}$  и  $f$  имеют длину  $nfft$  и соответствуют полному диапазону частот  $0 \dots fs$ . Этот вариант используется по умолчанию, если  $x$  содержит комплексные отсчеты;
- 'onesided' — векторы  $P_{xx}$  и  $f$  имеют длину  $\text{ceil}((nfft + 1)/2)$  и соответствуют половинному диапазону частот  $0 \dots fs/2$ . Этот вариант используется по умолчанию в случае вещественного вектора  $x$ .

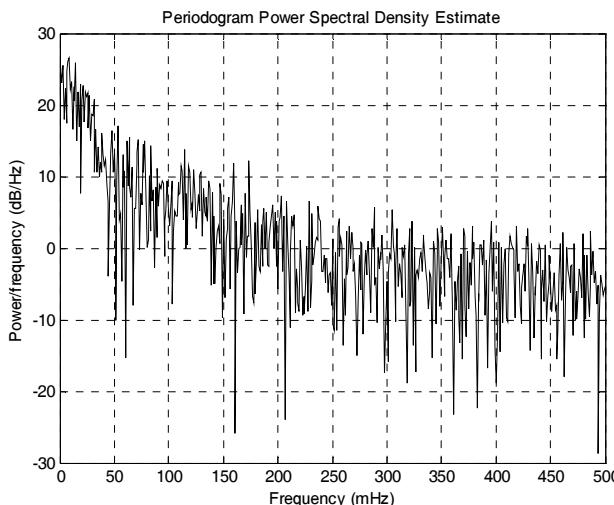
Параметр 'range' может быть указан в списке параметров в любом месте после  $w$ .

Возвращаемые параметры:  $P_{xx}$  — вектор значений спектральной плотности мощности,  $f$  — вектор значений частот, использованных для расчета. Шаг между соседними элементами этого вектора равен  $fs/nfft$ , первый элемент равен нулю.

Если выходные параметры при вызове не указаны, функция строит график спектральной плотности с помощью метода `plot` объекта класса `dspdata.psd`.

В качестве примера оценим спектр плотности мощности экспоненциально коррелированного случайного процесса (рис. 5.30):

```
>> % формирование случайного сигнала
>> x0 = randn(1, 1000);
>> a = 0.9;
>> x = filter(1, [1 -a], x0);
>> % расчет периодограммы и вывод графика
>> periodogram(x, [], [], 1)
```



**Рис. 5.30.** Периодограмма экспоненциально коррелированного случайного процесса, полученная с помощью функции `periodogram`

Как видите, хотя общий ход графика совпадает с аналитически построенным спектром данного случайного процесса (см. рис. 5.28), периодограмма оказывается

сильно изрезанной, что полностью соответствует теоретическим сведениям, приведенным ранее в разд. "Непараметрические методы" этой главы.

## Реализация метода Уэлча

Функция `pwelch` предназначена для определения спектра мощности случайного сигнала методом Уэлча (методом усреднения модифицированных периодограмм — *averaged modified periodogram method*). Синтаксис вызова функции следующий:

```
[Pxx, f] = pwelch(x, w, noverlap, nfft, fs, 'range')
```

Единственным обязательным входным параметром является `x` — вектор отсчетов анализируемого сигнала. Все остальные параметры имеют значения по умолчанию, которые используются, если при вызове в качестве параметра указана пустая матрица `[]` или если несколько последних параметров опущено.

Параметр `w` управляет выбором окна, используемого для анализа. Если `w` — число, используется окно Хэмминга указанной длины, если вектор, то данный вектор используется в качестве окна. По умолчанию используется окно Хэмминга, длина которого выбирается так, чтобы с учетом заданного перекрытия (см. ниже) сигнал оказался разделенным на 8 фрагментов.

Параметр `noverlap` задает (в отсчетах) перекрытие соседних фрагментов сигнала, для которых вычисляются периодограммы. По умолчанию перекрытие равно половине длины окна.

Параметр `nfft` задает размерность БПФ, используемого для вычисления периодограммы. По умолчанию `nfft = max(256, 2^nextpow2(nw))`, где `nw` — длина фрагмента сигнала (длина используемого окна).

Параметр `fs` указывает частоту дискретизации сигнала. Это значение используется для нормировки рассчитанного спектра мощности, а также при расчете возвращающегося вектора `f` и для оцифровки графика. По умолчанию значение этого параметра равно  $2\pi$ .

Строковый параметр `'range'` определяет частотный диапазон для возвращаемого вектора `Pxx`. Он используется так же, как для рассмотренной ранее функции `periodogram`, и может быть указан в списке параметров в любом месте после `noverlap`.

Возвращаемые параметры: `Pxx` — вектор значений спектральной плотности мощности, `f` — вектор значений частот, использованных для расчета. Шаг между соседними элементами этого вектора равен `fs/nfft`, первый элемент равен нулю.

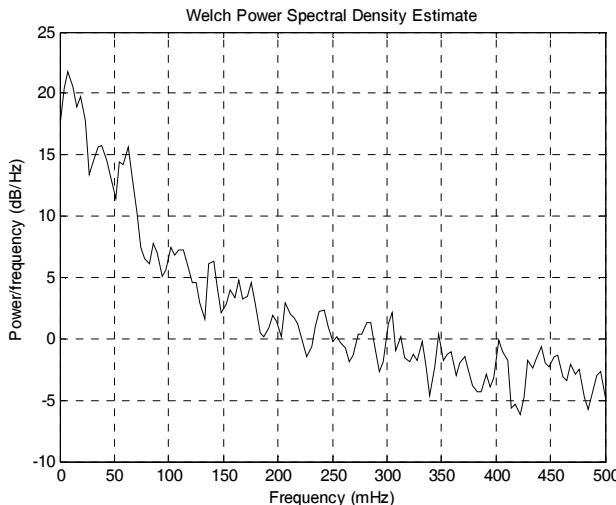
Если выходные параметры при вызове не указаны, функция строит график спектральной плотности с помощью метода `plot` объекта класса `dspdata.psd`.

Расчет спектра производится следующим образом. Анализируемый сигнал `x` делится на перекрывающиеся фрагменты согласно параметрам `w` и `noverlap`. Для каждого фрагмента вычисляется модифицированная периодограмма с использованием заданных окна и размерности БПФ (см. разд. "Расчет периодограммы" этой главы).

Полученный набор модифицированных периодограмм усредняется, и результат делится на  $f_s$ .

В качестве примера оценим спектральную плотность мощности экспоненциально коррелированного случайного процесса (рис. 5.31):

```
>> % формирование случайного сигнала
>> x0 = randn(1, 1000);
>> a = 0.9;
>> x = filter(1, [1 -a], x0);
>> % оценка СПМ методом Уэлча и вывод графика
>> pwelch(x, [], [], [], 1)
```



**Рис. 5.31.** Оценка спектра плотности мощности, полученная с помощью функции `pwelch`

Как видите, по сравнению с периодограммой, показанной ранее на рис. 5.30, спектр значительно меньше изрезан и приближается к теоретическому спектру данного процесса (см. рис. 5.28).

### ЗАМЕЧАНИЕ 1

На самых краях графика рис. 5.31 можно видеть резкий спад оценки СПМ примерно на 3 дБ. Это связано с тем, что для вещественного сигнала функции спектрального анализа возвращают оценку одностороннего спектра, уровень которого в два раза (на 3 дБ) выше, чем у двустороннего. Нулевая частота и частота Найквиста не имеют "зеркальной" пары, поэтому для них удвоения уровня спектра не происходит, что и вызывает скачки на графике. Эти скачки еще сильнее выражены на более гладких графиках, получаемых при использовании авторегрессионных методов (см. рис. 5.32).

### ЗАМЕЧАНИЕ 2

На методе Уэлча основано еще несколько функций пакета Signal Processing: `cpsd` (оценка взаимного спектра случайных сигналов), `mscoh` (оценка квадрата модуля

функции взаимной когерентности случайных сигналов) и `tfeestimate` (оценка комплексного коэффициента передачи линейной стационарной системы по реализациям входного и выходного случайных сигналов).

## Параметрический спектральный анализ в MATLAB

В данном разделе будет рассмотрено семейство функций, осуществляющих спектральный анализ с использованием авторегрессионных моделей, а также несколько функций, реализующих методы MUSIC и EV.

### Работа с авторегрессионными моделями

В MATLAB реализовано несколько методов анализа спектра на основе авторегрессионной модели формирования сигнала. Эти методы отличаются друг от друга способом оценивания коэффициентов авторегрессионной модели. Каждому методу соответствуют две функции — функция вычисления коэффициентов модели и функция собственно спектрального анализа. Функция спектрального анализа вызывает функцию расчета коэффициентов модели, а затем производит вычисление спектра по формуле (5.25). Имена функций сведены в табл. 5.5.

**Таблица 5.5.** Функции авторегрессионных моделей

Название метода	Функция расчета коэффициентов модели	Функция спектрального анализа
Ковариационный метод	<code>arcov</code>	<code>pcov</code>
Модифицированный ковариационный метод	<code>armcov</code>	<code>pmcov</code>
Метод Берга	<code>arburg</code>	<code>pburg</code>
Авторегрессионный метод Юла — Уолкера	<code>aryule</code>	<code>pyulear</code>

Синтаксис вызова функций для разных методов анализа совершенно одинаков, поэтому рассматривать функции по отдельности не имеет смысла.

Помимо перечисленных, в пакете Signal Processing Toolbox имеются функция `levinson`, реализующая алгоритм Левинсона — Дарбина, и функция `rlevinson`, осуществляющая обратное преобразование, т. е. рассчитывающая корреляционную функцию сигнала по коэффициентам линейного предсказания.

### Функции расчета коэффициентов авторегрессионных моделей

Синтаксис вызова функций, выполняющих оценивание коэффициентов авторегрессионных моделей формирования сигнала, приведен ниже:

`[a, e, r] = arxxxx(x, p)`

Здесь `arxxxx` — имя одной из функций, перечисленных во втором столбце табл. 5.5.

Входные параметры:  $x$  — вектор отсчетов сигнала,  $p$  — порядок авторегрессионной модели.

Выходные параметры:  $a$  — вектор коэффициентов модели (это полный вектор коэффициентов знаменателя функции передачи формирующего фильтра, его длина равна  $p + 1$ , а первый элемент равен 1);  $e$  — оценка дисперсии белого шума, возбуждающего фильтр;  $r$  — вектор коэффициентов отражения (коэффициенты  $b_k^{(k)}$  из формул разд. "Алгоритм Левинсона—Дарбина" этой главы).

В качестве примера оценим параметры авторегрессионной модели для экспоненциально коррелированного шума (такой шум является авторегрессионным процессом первого порядка, поэтому задаем  $p = 1$ ):

```
>> % формирование случайного сигнала
>> x0 = randn(1, 1000);
>> a = 0.9;
>> x = filter(1, [1 -a], x0);
>> % оценка параметров авторегрессионной модели
>> [aa, e] = arburg(x, 1)
aa =
    1.0000   -0.9030
e =
    1.0201
```

Как видите, оценка коэффициента обратной связи весьма близка к истинной. Оценка дисперсии белого шума, возбуждающего авторегрессионный формирователь, для данной реализации процесса также оказалась очень близка к теоретическому значению, равному единице.

## Функции авторегрессионного спектрального анализа

Синтаксис вызова функций, производящих расчет спектра мощности на основе авторегрессионной модели формирования сигнала, приведен ниже:

```
[Pxx, f] = pxxx(x, p, nfft, fs, 'range')
```

Здесь  $\text{pxxx}$  — имя одной из функций, перечисленных в последнем столбце табл. 5.5.

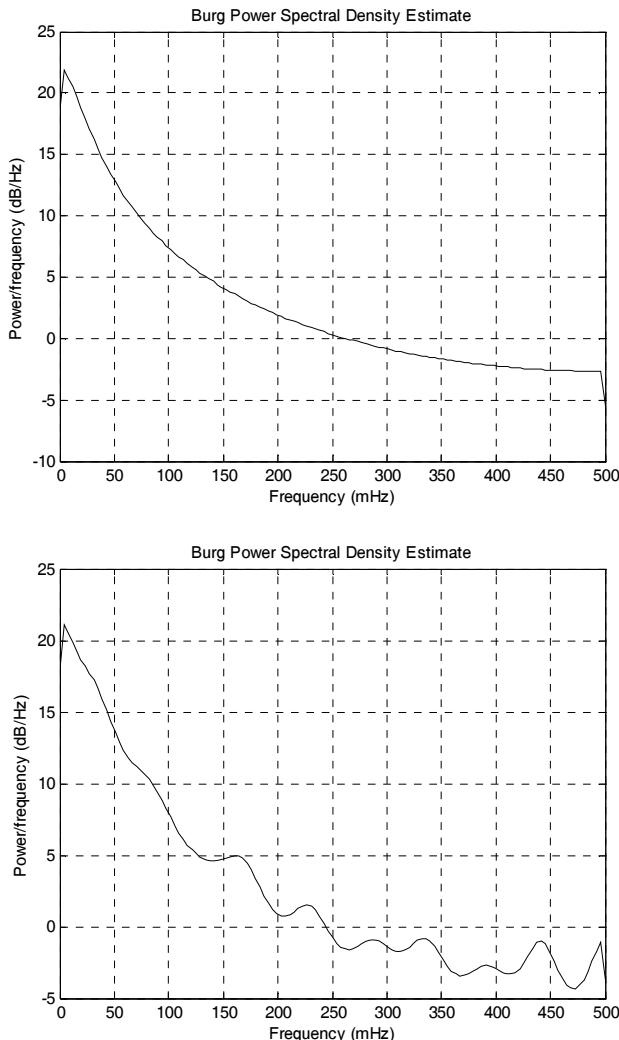
Обязательные входные параметры:  $x$  — вектор отсчетов сигнала,  $p$  — порядок авторегрессионной модели. Остальные входные параметры имеют значения по умолчанию, которые используются, если параметр представляет собой пустую матрицу  $[]$  или отсутствует.

Параметр  $nfft$  задает число частотных точек для расчета спектра. По умолчанию его значение равно 256. Параметр  $fs$  — частота дискретизации входного сигнала (используется при расчете возвращаемого вектора частот  $f$  и для оцифровки горизонтальной оси графика). Значение по умолчанию —  $2\pi$  Гц.

Строковый параметр  $'range'$  определяет частотный диапазон для возвращаемого вектора  $Pxx$ . Он используется так же, как для рассмотренной ранее функции  $\text{periodogram}$ , и может быть указан в списке параметров в любом месте после  $p$ .

Возвращаемые параметры:  $\text{Pxx}$  — вектор значений спектральной плотности мощности,  $f$  — вектор значений частот, использованных для расчета. Шаг между соседними элементами этого вектора равен  $fs/nfft$ , первый элемент равен нулю.

Если выходные параметры при вызове не указаны, функция строит график спектральной плотности с помощью метода `plot` объекта класса `dspdata.psd`.



**Рис. 5.32.** Результаты авторегрессионного спектрального анализа при оптимальном (сверху) и избыточном (снизу) порядке модели

В качестве примера выполним спектральный анализ экспоненциально коррелированного случайного сигнала. Такой сигнал является авторегрессионным процессом первого порядка, поэтому мы продемонстрируем влияние порядка авторегрессионной модели на получаемые результаты, построив оценки спектра (рис. 5.32) для  $p = 1$  (оптимальный порядок модели) и  $p = 20$  (избыточный порядок):

```
>> % формирование случайного сигнала
>> x0 = randn(1, 1000);
>> a = 0.9;
>> x = filter(1, [1 -a], x0);
>> % оценка СПМ методом Берга и вывод графика
>> pburg(x, 1, [], 1) % порядок модели оптимальен
>> figure
>> pburg(x, 20, [], 1) % порядок модели избыточен
```

Из графиков видно, что при оптимальном порядке модели (рис. 5.32, сверху) спектр практически совпадает с теоретическими результатами (см. рис. 5.28), а при избыточном (рис. 5.32, снизу) в спектре начинают возникать ложные флуктуации.

### **Функции *levinson* и *rlevinson***

Алгоритм Левинсона — Дарбина реализован в пакете расширения Signal Processing Toolbox в виде функции *levinson*. В простейшем случае она имеет следующий синтаксис вызова:

```
[a, e] = levinson(r)
```

Входной параметр *r* задает вектор отсчетов корреляционной функции сигнала. Длина этого вектора определяет порядок предсказания.

Возвращаемый результат *a* представляет собой вектор коэффициентов фильтра, формирующего сигнал ошибки предсказания, а скаляр *e* — это средний квадрат этой ошибки. Так же, как и для функций расчета коэффициентов авторегрессионных моделей (см. соответствующий раздел ранее в этой главе), *a* — это *полный* вектор знаменателя функции передачи формирующего фильтра, его длина на единицу превышает порядок предсказания, первый элемент равен 1, а остальные элементы имеют противоположный знак по сравнению с элементами вектора *b*, вычислявшегося ранее в разд. "Алгоритм Левинсона — Дарбина" этой главы.

Если необходимо рассчитать вектор коэффициентов предсказания меньшего порядка, чем длина корреляционной функции, можно задать порядок предсказания с помощью второго входного параметра *p*:

```
[a, e] = levinson(r, p)
```

Кроме того, можно указать третий выходной параметр, чтобы получить в результате расчета дополнительную информацию — вектор коэффициентов отражения *k*:

```
[a, e, k] = ...
```

С помощью функции *rlevinson* можно осуществить обратное преобразование, т. е. по известным коэффициентам линейного предсказания рассчитать корреляционную функцию сигнала. В простейшем случае данная функция имеет следующий синтаксис вызова:

```
r = rlevinson(a, e)
```

Смысл параметров *r*, *a* и *e* здесь такой же, как и для функции *levinson*. Следует отметить, что величина среднего квадрата ошибки предсказания *e* влияет только на

масштаб получаемой корреляционной функции, но не на ее форму. Однако, к сожалению, значения по умолчанию (которое обеспечивало бы, скажем, расчет нормированной КФ) для этого параметра не предусмотрено.

При вызове функции `rlevinson` можно использовать до четырех выходных параметров, чтобы получить дополнительную информацию:

```
[r, u, k, e_all] = rlevinson(a, e)
```

Дополнительные выходные параметры имеют следующий смысл:

□  $u$  — квадратная матрица, размер которой на единицу больше порядка предсказания. В этой матрице содержатся комплексно-сопряженные и перевернутые по вертикали коэффициенты векторов авторегрессионных моделей всех порядков, начиная с нулевого. Эти векторы являются столбцами матрицы  $u$ , так что первый столбец соответствует нулевому порядку предсказания. Таким образом, на главной диагонали матрицы стоят единицы, а ниже главной диагонали — нули. Выделить из матрицы  $u$  вектор коэффициентов авторегрессионной модели порядка  $n$  можно так:

```
a_n = u((n+1):-1:1, n+1)'
```

□  $k$  — вектор-столбец коэффициентов отражения. Его также можно получить из матрицы  $u$  — ведь последний элемент вектора коэффициентов предсказания является именно соответствующим коэффициентом отражения. Поэтому  $k = u(1, 2:end)'$ ;

□  $e_{all}$  — вектор-строка значений среднего квадрата ошибки предсказания всех порядков, начиная с первого.

В качестве примера посмотрим, как с ростом порядка предсказания меняются вектор коэффициентов предсказания и средний квадрат ошибки для использованного в предыдущем разделе экспоненциально коррелированного случайного процесса. Для этого после формирования модели такого сигнала мы вычислим оценку его КФ с помощью функции `xcorr`, затем используем функцию `levinson`, чтобы получить параметры предсказания максимального из интересующих нас порядков, и, наконец, вызовем функцию `rlevinson`, чтобы рассчитать параметры предсказания для всех порядков, начиная с первого. В заключение матрица коэффициентов авторегрессионных моделей и вектор средних квадратов ошибки выводятся на экран:

```
>> % формирование случайного сигнала
>> x0 = randn(1, 1000);
>> a0 = 0.9;
>> x = filter(1, [1 -a0], x0);
>> N = 8; % максимальный порядок предсказания
>> r = xcorr(x, N, 'biased'); % оценка КФ
>> r = r(N+1:end); % берем нужную половину вектора
>> [a, e] = levinson(r); % расчет предсказания порядка N
>> [rl, u, k, e1] = rlevinson(a, e); % параметры всех порядков
>> u % выводим матрицу коэффициентов предсказания всех порядков
>> e1 % выводим средний квадрат ошибки всех порядков
```

В результате работы этого кода будет выведено следующее:

```
u =
1.0000 -0.9164 0.0499 0.0386 0.0196 -0.0119 -0.0338 0.0217 -0.0001
0 1.0000 -0.9621 0.0128 0.0198 0.0310 0.0206 -0.0546 0.0217
0 0 1.0000 -0.9602 0.0130 0.0196 0.0306 0.0209 -0.0546
0 0 0 1.0000 -0.9594 0.0128 0.0190 0.0310 0.0209
0 0 0 0 1.0000 -0.9597 0.0117 0.0196 0.0310
0 0 0 0 0 1.0000 -0.9593 0.0122 0.0196
0 0 0 0 0 0 1.0000 -0.9600 0.0122
0 0 0 0 0 0 0 1.0000 -0.9600
0 0 0 0 0 0 0 0 1.0000
```

e1 =

```
0.9387 0.9364 0.9350 0.9346 0.9345 0.9334 0.9330 0.9330
```

Полученные результаты показывают, что средний квадрат ошибки с ростом порядка предсказания практически не уменьшился (при переходе от первого порядка предсказания к восьмому его значение упало менее чем на 1%). Так и должно было быть, поскольку анализируемый случайный процесс является авторегрессионным процессом первого порядка, и первый порядок предсказания является в данном случае оптимальным. Что касается вектора коэффициентов предсказания, то из матрицы  $u$  видно, что при избыточном порядке предсказания "лишние" элементы вектора коэффициентов имеют абсолютные величины порядка нескольких сотых (примерно до 0,05). "Лишние" степени свободы дают модели возможность подстраиваться под особенности конкретной реализации случайного процесса, что и приводит к появлению флюктуаций на графике оценки СПМ (см. рис. 5.32, снизу).

## Расчет корреляционной матрицы

Как отмечалось в этой главе ранее, оценка корреляционной матрицы сигнала является первым этапом авторегрессионного спектрального анализа, причем эта оценка может быть получена различными способами. В MATLAB для оценки корреляционной матрицы случайного сигнала по вектору его отсчетов с использованием временного усреднения служит функция `corrmtx`, имеющая следующий синтаксис:

```
[X, R] = corrmtx(x, m, 'method')
```

Здесь  $x$  — вектор отсчетов сигнала,  $m$  — размерность рассчитываемой корреляционной матрицы. Необязательный строковый параметр '`method`' управляет тем, как обрабатываются крайние отсчеты сигнала. Возможные значения этого параметра соответствуют методам, перечисленным ранее в разд. "*Оценка корреляционной матрицы*" этой главы:

- '`autocorrelation`' — автокорреляционный метод (этот вариант используется по умолчанию);
- '`prewindowed`' — метод предзвешивания;
- '`postwindowed`' — метод постзвешивания;

- 'covariance' — ковариационный метод;
- 'modified' — модифицированный ковариационный метод.

Результатами работы функции являются матрица промежуточных данных  $\mathbf{x}$  и собственно корреляционная матрица  $\mathbf{R}$ , которая рассчитывается как  $\mathbf{x}' * \mathbf{x}$  (апостроф в MATLAB обозначает эрмитово сопряжение).

Столбцы выходной матрицы  $\mathbf{x}$  представляют собой сдвинутые копии вектора отсчетов сигнала, в начале и в конце дополненные нулями либо обрезанные (в зависимости от использованного метода), а также деленные на корень квадратный из нормировочного коэффициента (он также зависит от выбранного метода). Если не учитывать упомянутое деление, то матрица  $\mathbf{x}$  имеет вид, транспонированный по отношению к таблицам, приводившимся ранее в качестве примера в разд. "Оценка корреляционной матрицы" этой главы.

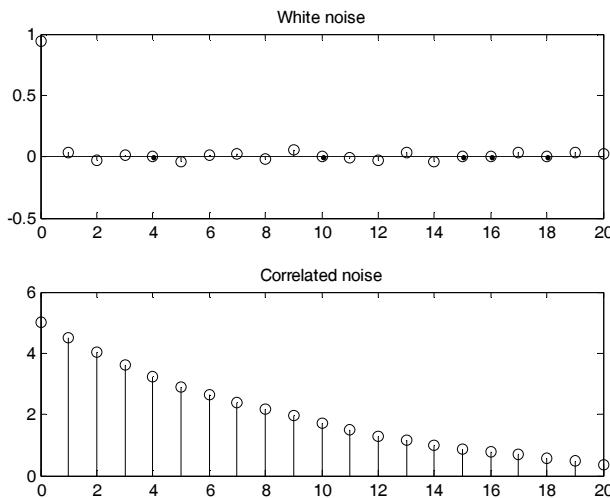


Рис. 5.33. Оценка корреляционной функции белого (сверху) и экспоненциально коррелированного (снизу) шума

В качестве примера вычислим корреляционные матрицы и построим графики их первых строк (т. е. корреляционных функций) для использовавшихся в предыдущих разделах экспоненциально коррелированного шума и белого шума, на основе которого он был получен. При расчете использован выбираемый по умолчанию автокорреляционный метод, а результаты показаны на рис. 5.33:

```
>> % формирование случайного сигнала
>> x0 = randn(1, 1000); % нормальный белый шум
>> a0 = 0.9; % параметр экспоненциальной корреляции
>> x = filter(1, [1 -a0], x0); % фильтрация белого шума
>> N = 20; % размер корреляционной матрицы
>> [tmp, R0] = corrmtx(x0, N);
>> [tmp, R] = corrmtx(x, N);
```

```

>> k = 0:N;                                % индексы КФ
>> subplot(2, 1, 1)
>> stem(k, R0(1,:))                      % график КФ белого шума
>> title('White noise')
>> subplot(2, 1, 2)
>> stem(k, R(1,:))                        % график КФ коррелированного шума
>> title('Correlated noise')

```

Из графиков видно, что отсчеты белого шума действительно некоррелированы, а отсчеты фильтрованного шума на самом деле дают требуемую экспоненциальную функцию корреляции.

## Реализация метода MUSIC

Функция `pmusic` реализует расчет псевдоспектра путем анализа собственных чисел и собственных векторов корреляционной матрицы сигнала (метод MUSIC). Синтаксис вызова функции `pmusic` следующий:

```
[s, f, v, e] = pmusic(x, p, nfft, fs, nwin, noverlap, 'range')
```

У функции много входных параметров, однако обязательными являются только `x` и `p`. В простейшем случае `x` — это вектор отсчетов сигнала, а `p` — целое число, задающее размерность сигнального подпространства. Однако смысл этих параметров может быть и несколько иным, о чем речь пойдет чуть ниже.

## Формирование корреляционной матрицы

Данный метод расчета спектра основан на анализе корреляционной матрицы сигнала, поэтому прежде всего необходимо определить эту матрицу. Если `x` — вектор отсчетов сигнала, формированием корреляционной матрицы управляют параметры `nwin` и `noverlap`. Параметр `nwin` может быть числом, обозначающим размер прямоугольного окна, или вектором, содержащим коэффициенты окна. Сигнал делится на фрагменты в соответствии с длиной окна (равной `nwin` или `length(nwin)`), при этом перекрытие соседних фрагментов составляет `noverlap` отсчетов.

По умолчанию используется прямоугольное окно, размер которого в два раза превышает заданную размерность сигнального подпространства. Перекрытие фрагментов по умолчанию на единицу меньше размера окна (т. е. соседние фрагменты сигнала сдвинуты друг относительно друга на один отсчет).

Кроме вектора, входной параметр `x` может быть матрицей, содержащей несколько реализаций (наблюдений) случайного процесса. При этом каждая строка матрицы соответствует отдельной реализации. Корреляционная матрица в этом случае рассчитывается как  $\mathbf{x}^* \mathbf{x}$ . Параметр `noverlap` игнорируется, так же как и числовой параметр `nwin`. Если `nwin` задан в виде вектора, его длина должна быть равна числу столбцов в матрице `x` (при этом каждая реализация сигнала поэлементно умножается на вектор коэффициентов окна перед вычислением корреляционной матрицы).

Наконец, в качестве `x` можно подставить уже готовую корреляционную матрицу (в этом случае параметр `x` должен удовлетворять свойствам корреляционной мат-

рицы — быть квадратной эрмитовой, т. е. обладающей комплексно-сопряженной симметрией, матрицей Теплица). Чтобы использовать такой вариант, в число входных параметров необходимо включить строку 'corr' (в любом месте списка параметров после размерности сигнального подпространства  $p$ ). Параметры  $nwin$  и  $noverlap$  в данном случае игнорируются.

## Управление размером сигнального подпространства

Как уже говорилось, в простейшем случае размерность сигнального подпространства задается числом  $p$ . Однако функция `pmusic` позволяет использовать более гибкий способ разделения собственного пространства корреляционной матрицы на сигнальное и шумовое подпространства. Для этого параметр  $p$  должен быть двухэлементным вектором. Значение  $p(1)$  задает максимально допустимый предел размерности сигнального подпространства, а значение  $p(2)$  — пороговый уровень для собственных чисел, соответствующих сигнальному подпространству. Если  $\lambda_{min}$  — минимальное собственное число корреляционной матрицы, то собственные числа, меньшие чем  $\lambda_{min} p(2)$ , будут отнесены к шумовым компонентам, а собственные числа, превышающие этот порог, — к сигнальным компонентам.

Чтобы использовать этот режим, длина окна должна быть не меньше, чем  $p(1)$ .

Значение  $p(1)$  может быть равно бесконечности (`Inf`), при этом необходимо явное задание параметра  $nwin$ , поскольку использование значения по умолчанию оказывается невозможным.

## Управление расчетом псевдоспектра

Параметр  $nfft$  задает число точек для расчета псевдоспектра. Значение по умолчанию — 256. Частота дискретизации  $fs$  используется для расчета возвращаемого вектора  $f$  и для вывода графика. Ее значение по умолчанию равно  $2\pi$ . Наконец, строковый параметр 'range' задает диапазон частот анализа. Этот параметр может принимать одно из двух значений:

- 'whole' — длина возвращаемых векторов  $s$  и  $f$  равна  $nfft$ , расчет проводится для всего частотного диапазона  $0...fs$ . Этот вариант принят по умолчанию для случая, когда  $x$  содержит комплексные элементы;
- 'half' — длина возвращаемых векторов  $s$  и  $f$  равна  $\text{ceil}((nfft+1)/2)$ , расчет проводится для половинного частотного диапазона  $0...fs/2$ . Этот вариант принят по умолчанию для вещественного сигнала  $x$ .

## Выходные параметры

Возвращаемые функцией `pmusic` результаты имеют следующий смысл:

- $s$  — вектор рассчитанных значений псевдоспектра;
- $f$  — вектор частот, которым соответствуют значения в векторе  $s$ ;
- $v$  — собственные векторы корреляционной матрицы, соответствующие шумовому подпространству;

- $e$  — собственные числа корреляционной матрицы, соответствующие шумовому подпространству.

## Функция *rootmusic*

Если функция *pmusic* дает вектор рассчитанных значений псевдоспектра, то функция *rootmusic*, используя тот же метод анализа MUSIC, извлекает из корреляционной матрицы сигнала несколько иную информацию — частоты содержащихся в сигнале комплексных экспонент и соответствующие им мощности.

Синтаксис вызова функции следующий:

```
[f, pow] = rootmusic(x, p, fs, 'corr')
```

Все входные параметры имеют тот же смысл, что и для функции *pmusic*.

Результатами работы функции являются векторы частот  $f$  и соответствующих им мощностей  $pow$ . Длина вектора  $f$  равна размерности сигнального подпространства. Размер вектора  $pow$  зависит от того, вещественным или комплексным является анализируемый сигнал  $x$ . Если сигнал  $x$  комплексный, длина вектора  $pow$  равна длине вектора  $f$ , а если вещественный, то она вдвое меньше.

## Пример использования

В качестве примера использования функций *pmusic* и *rootmusic* сформируем сигнал, состоящий из трех синусоид разного уровня, смешанных с белым шумом, а затем выведем график его псевдоспектра (рис. 5.34) и оценим частоты гармонических составляющих:

```
>> Fs = 1000; % частота дискретизации
>> f1 = 50; % частоты, содержащиеся в сигнале
>> f2 = 120;
>> f3 = 300;
>> t = 0:1/Fs:1; % дискретное время
>> % анализируемый сигнал
>> s = 2*cos(2*pi*f1*t) + 5*sin(2*pi*f2*t) - 0.5*cos(2*pi*f3*t);
>> s = awgn(s, 10, 'measured'); % отношение С/Ш = 10 dB
>> pmusic(s, 6, [], Fs) % график псевдоспектра
>> rootmusic(s, 6, Fs) % оценка частот
ans =
120.0089
-120.0089
50.2420
-50.2420
299.1724
-299.1724
```

Как видите, частоты оценены довольно точно, а график демонстрирует четкие пики.

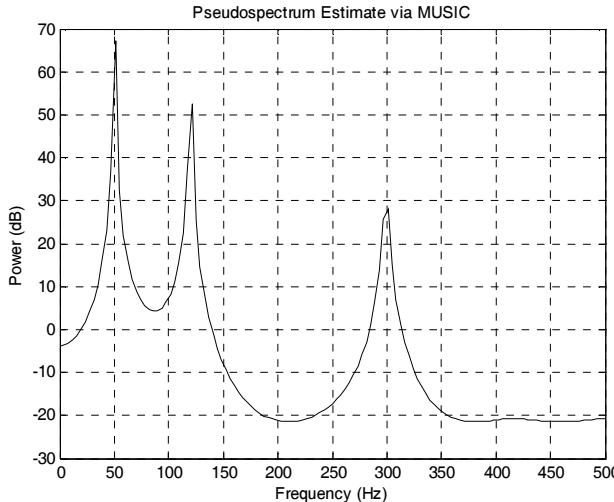


Рис. 5.34. Оценка псевдоспектра, полученная с помощью функции `pmusic`

## Реализация метода EV

Функции `peig` и `rooteig` аналогичны функциям `pmusic` и `rootmusic` соответственно. Отличие состоит в том, что с их помощью реализуется метод EV и расчеты производятся по формулам (5.45) и (5.46), где при суммировании используются весовые коэффициенты, обратно пропорциональные "шумовым" собственным числам корреляционной матрицы сигнала. Синтаксис вызова функций следующий:

```
[s, f, v, e] = peig(x, p, nfft, fs, nwin, noverlap, 'range')
[f, pow] = rooteig(x, p, fs, 'corr')
```

Все входные и выходные параметры соответствуют описаниям, приведенным ранее для функций `pmusic` и `rootmusic`.

## Хранение и отображение спектральных данных

Для хранения результатов спектральных расчетов и выполнения операций над ними предназначены объекты `dspdata`. Имеется три варианта конструкторов этих объектов:

- `dspdata.msspectrum` — объект для хранения дискретных спектров, элементы которых представляют собой *мощности* спектральных составляющих;
- `dspdata.psd` — объект для хранения спектров, значения которых представляют собой *спектральную плотность мощности*;
- `dspdata.pseudospectrum` — объект для хранения псевдоспектров, полученных в результате применения методов MUSIC или EV.

Рассмотрим работу со спектрами на примере объекта `dspdata.psd`. Чтобы создать объект, записать в него частоту дискретизации `fs`, вектор частот `f` и вектор соответ-

ствующих этим частотам значений СПМ  $P_{xx}$ , необходимо выполнить следующий код:

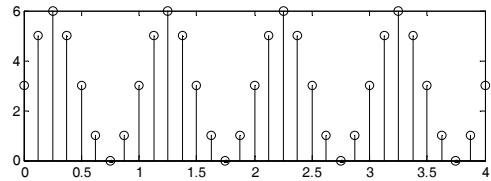
```
>> Hpsd = dspdata.psd(Pxx, f, 'Fs', Fs);
```

Теперь над сохраненными в объекте данными можно выполнять различные операции, например:

- `plot(Hpsd)` — выводит график спектра в текущем графическом окне;
- `P = avgpower(Hpsd, [f1 f2])` — возвращает среднюю мощность в частотном диапазоне между  $f_1$  и  $f_2$ . Второй входной параметр можно опустить, в этом случае будет рассчитана полная средняя мощность сигнала.

Пример использования объекта `dspdata` был приведен при построении рис. 5.28. За информацией о других свойствах и методах этих объектов обратитесь к справочной системе MATLAB.

# ГЛАВА 6



## Проектирование дискретных фильтров

Под *проектированием* (или *синтезом*) цифрового фильтра понимается выбор таких наборов коэффициентов  $\{a_i\}$  и  $\{b_i\}$ , при которых характеристики получающегося фильтра удовлетворяют заданным требованиям. Строго говоря, в задачу проектирования входит и выбор подходящей структуры фильтра (см. разд. "Формы реализации цифровых фильтров" главы 4) с учетом конечной точности вычислений. Это особенно актуально при реализации фильтров "в железе" — с использованием специализированных БИС или цифровых сигнальных процессоров. Эффекты, связанные с конечной точностью вычислений, рассмотрены в главе 7, здесь же речь пойдет только о методах синтеза цифровых фильтров, т. е. способах получения значений  $\{a_i\}$  и  $\{b_i\}$ .

Методы синтеза цифровых фильтров можно классифицировать по различным признакам:

- по типу получаемого фильтра:
  - методы синтеза *рекурсивных* фильтров;
  - методы синтеза *нерекурсивных* фильтров;
- по наличию аналогового прототипа:
  - методы синтеза *с использованием* аналогового прототипа;
  - *прямые* (*без использования* аналогового прототипа) методы синтеза.

### Синтез рекурсивных фильтров по аналоговому прототипу

При синтезе дискретного фильтра по аналоговому прототипу необходимо реализовать переход из  $p$ -области в  $z$ -область, т. е. преобразовать функцию передачи аналогового фильтра  $H(p)$  в функцию передачи дискретного фильтра  $H(z)$ . Получающийся дискретный фильтр не может быть полностью идентичен аналоговому по своим характеристикам — хотя бы потому, что частотные характеристики дискретного фильтра являются периодическими. Можно говорить только об определенном *соответствии* характеристик аналогового и дискретного фильтров.

В данном разделе мы рассмотрим два метода синтеза рекурсивных дискретных фильтров по аналоговым прототипам:

- метод билинейного  $z$ -преобразования;
- метод инвариантной импульсной характеристики.

Поскольку теория аппроксимации идеальных АЧХ аналоговыми средствами хорошо развита (см. разд. "Расчет аналоговых фильтров-прототипов" главы 2), методы синтеза дискретных фильтров по аналоговым прототипам получили широкое распространение.

## Метод билинейного $z$ -преобразования

Данный метод позволяет синтезировать рекурсивный дискретный фильтр по частотной характеристике аналогового прототипа.

Функция передачи аналоговой цепи с сосредоточенными параметрами представляет собой дробно-рациональную функцию переменной  $p$ . Чтобы получить функцию передачи дискретного фильтра, необходимо перейти из  $p$ -области в  $z$ -область, причем дробно-рациональный характер функции должен сохраняться (см. разд. "Функция передачи" главы 4). Поэтому замена для переменной  $p$  должна представлять собой также дробно-рациональную функцию переменной  $z$ . Чтобы частотные характеристики аналогового и дискретного фильтров были связаны простой зависимостью, искомая замена переменной должна отображать мнимую ось в  $p$ -области на единичную окружность в  $z$ -области. В этом случае частотные характеристики аналогового и дискретного фильтров будут связаны лишь трансформацией частотной оси и никаких искажений "по вертикали" не будет.

Простейшей из функций, которая удовлетворяет перечисленным требованиям, является *билинейное  $z$ -преобразование* (*bilinear transformation*):

$$p = \frac{2}{T} \frac{z-1}{z+1} = 2f_d \frac{1-z^{-1}}{1+z^{-1}}. \quad (6.1)$$

Частотные характеристики аналогового  $\dot{K}_a(\omega)$  и дискретного  $\dot{K}_d(\omega)$  фильтров, как уже было сказано, связаны друг с другом лишь трансформацией частотной оси:

$$\begin{aligned} \dot{K}_d(\omega) &= \dot{K}_a\left(\frac{2}{T} \operatorname{tg}\left(\frac{\omega T}{2}\right)\right), \\ \dot{K}_a(\omega) &= \dot{K}_d\left(\frac{2}{T} \operatorname{arctg}\left(\frac{\omega T}{2}\right)\right). \end{aligned} \quad (6.2)$$

На низких частотах, когда  $\omega T \ll 1$ , тангенс примерно равен своему аргументу, так что

$$\frac{2}{T} \operatorname{tg}\left(\frac{\omega T}{2}\right) \approx \omega, \text{ если } \omega T \ll 1.$$

Поэтому в области низких частот частотные характеристики аналогового и дискретного фильтров почти совпадают. Далее, по мере ускорения роста функции тангенса, частотная характеристика дискретного фильтра все сильнее сжимается по горизонтали (по сравнению с аналоговым прототипом) и на частоте Найквиста, равной  $\pi/T$ , достигает значения, которое частотная характеристика аналогового фильтра имела бы на бесконечной частоте (рис. 6.1).

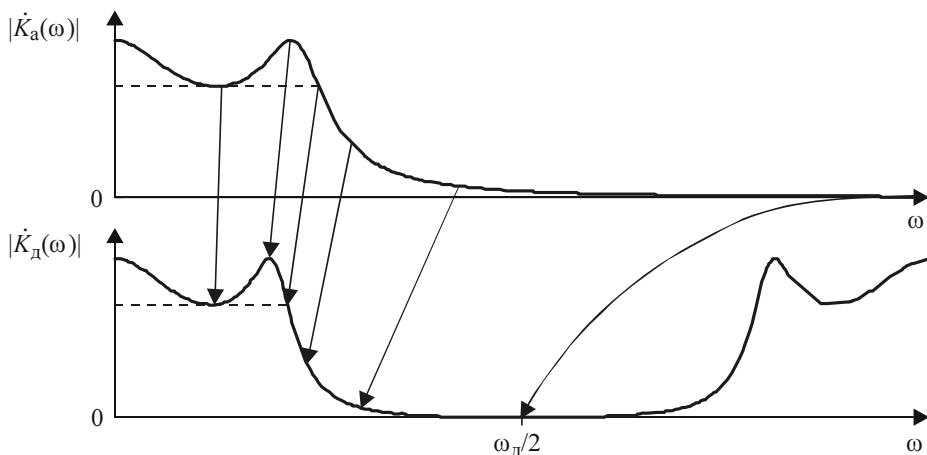


Рис. 6.1. Трансформация частотной оси при билинейном  $z$ -преобразовании

При билинейном  $z$ -преобразовании левая половина  $p$ -плоскости отображается внутрь единичной окружности на  $z$ -плоскости, поэтому синтез по устойчивому аналоговому прототипу даст гарантированно устойчивый дискретный фильтр (см. разд. "Полюсы и вычеты" в главах 2 и 4, где шла речь об условиях устойчивости соответственно аналоговых и дискретных систем).

Для получения дискретного фильтра с заданными частотами среза необходимо скорректировать частоты среза аналогового прототипа, чтобы компенсировать искажения частотной оси. Так, для синтеза дискретного ФНЧ с частотой среза  $\omega_{0d}$  аналоговый фильтр-прототип должен иметь частоту среза  $\omega_{0a}$ , связанную с  $\omega_{0d}$  следующим образом:

$$\omega_{0a} = \frac{2}{T} \operatorname{tg} \left( \frac{\omega_{0d} T}{2} \right). \quad (6.3)$$

## Метод инвариантной импульсной характеристики

Метод *инвариантной импульсной характеристики* (*impulse invariance*) позволяет синтезировать рекурсивный дискретный фильтр путем дискретизации импульсной характеристики аналогового прототипа.

Частотная характеристика получаемого фильтра связана с частотной характеристикой аналогового прототипа точно так же, как спектр дискретизированного сигнала

связан со спектром сигнала аналогового — периодическим повторением (см. формулу (3.7) в разд. "Спектр дискретного сигнала" главы 3). Поэтому для получения хороших результатов при таком методе синтеза коэффициент передачи аналогового прототипа должен быть пренебрежимо малым на частотах, превышающих частоту Найквиста. Отсюда следует также, что этот метод подходит для создания ФНЧ и полосовых фильтров, но непригоден для синтеза ФВЧ и режекторных фильтров. Кроме того, чтобы избежать появления дополнительного множителя, равного  $1/T$  (см. формулу (3.7)), дискретизированную импульсную характеристику аналогового фильтра нужно еще умножить на период дискретизации  $T$ . В итоге получаем, что связь между импульсными характеристиками аналогового прототипа и результирующего дискретного фильтра должна выражаться следующей формулой:

$$h_{\text{д}}(k) = Th_a(kT). \quad (6.4)$$

Как говорилось в разд. "Полюсы и вычеты" главы 2, импульсная характеристика аналоговой цепи с сосредоточенными параметрами может быть представлена в виде суммы экспоненциальных слагаемых

$$h_{1a}(t) = Ae^{pt}, \quad t \geq 0,$$

и (при наличии кратных полюсов) слагаемых вида

$$h_{2a}(t) = At^n e^{pt}, \quad t \geq 0.$$

Каждое из этих слагаемых может быть дискретизовано и воспроизведено в дискретном виде. Экспоненциальные слагаемые вида  $h_{1a}(t)$  после дискретизации в соответствии с формулой (6.4) дают последовательность отсчетов

$$h_{1\text{д}}(k) = ATe^{kpT}, \quad k \geq 0. \quad (6.5)$$

$Z$ -преобразование этой последовательности равно (см. разд. "Примеры вычисления  $Z$ -преобразования" главы 3)

$$H_1(z) = AT(1 + e^{pT}z^{-1} + e^{2pT}z^{-2} + e^{3pT}z^{-3} + \dots) = \frac{AT}{1 - e^{pT}z^{-1}}.$$

Такая функция передачи соответствует дискретному рекурсивному фильтру 1-го порядка.

Теперь займемся слагаемыми вида  $h_{2a}(t)$ , соответствующими кратным полюсам. Такое слагаемое дает после дискретизации последовательность отсчетов

$$h_{2\text{д}}(k) = AT(kT)^n e^{kpT}, \quad k \geq 0.$$

$Z$ -преобразование этой последовательности можно легко рассчитать, если заметить, что она представляет собой  $n$ -ю производную по  $p$  от последовательности (6.5):

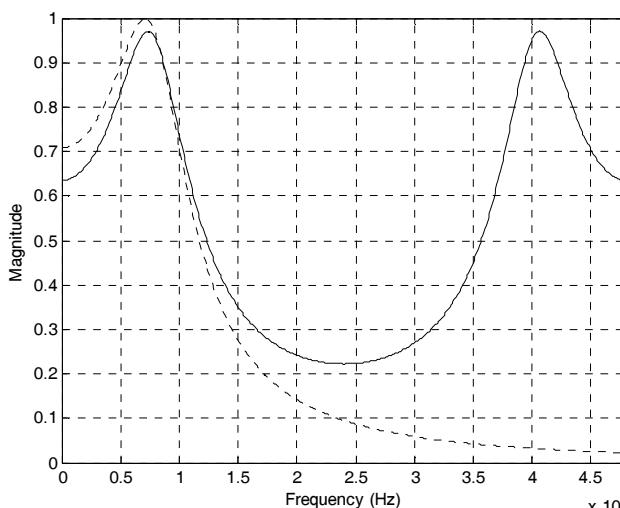
$$\begin{aligned} H_2(z) &= AT \left( T^n e^{pT} z^{-1} + (2T)^n e^{2pT} z^{-2} + (3T)^n e^{3pT} z^{-3} + \dots \right) = \\ &= AT \frac{d^n}{dp^n} \left( \frac{1}{1 - e^{pT} z^{-1}} \right). \end{aligned}$$

Такая функция передачи, получаемая после вычисления  $n$ -й производной, соответствует дискретному рекурсивному фильтру  $n$ -го порядка.

Выполнив указанные преобразования для всех слагаемых, мы получим дискретный фильтр, импульсная характеристика которого представляет собой дискретизированную версию импульсной характеристики аналогового прототипа.

В качестве примера синтезируем методом инвариантной импульсной характеристики ФНЧ Чебышева 2-го порядка с частотой среза 10 кГц, причем специально выберем недостаточно высокую частоту дискретизации (48 кГц), чтобы хорошо видеть эффекты, связанные с наложением сдвинутых копий спектра (рис. 6.2):

```
>> f0 = 10e3; % частота среза
>> [b, a] = cheby1(2, 3, f0*2*pi, 's'); % аналоговый фильтр
>> Fs = 48e3; % частота дискретизации
>> [bz, az] = impinvar(b, a, Fs); % дискретный фильтр
>> % ЧХ дискретного фильтра
>> [hz, wz] = freqz(bz, az, [], Fs, 'whole');
>> % ЧХ аналогового фильтра
>> h = freqs(b, a, 2*pi*wz);
>> plot(wz, abs(h), ':')
>> hold on
>> plot(wz, abs(hz))
>> hold off
>> xlabel('Frequency (Hz)')
>> ylabel('Magnitude')
>> xlim([0 Fs])
>> grid on
```



**Рис. 6.2.** АЧХ аналогового прототипа (пунктир) и дискретного фильтра (сплошная линия), синтезированного методом инвариантной импульсной характеристики

На рисунке хорошо видно, что из-за недостаточно высокой частоты дискретизации коэффициент передачи аналогового фильтра на частоте Найквиста недостаточно мал, что обуславливает заметные искажения формы АЧХ синтезированного дискретного фильтра. Повышение частоты дискретизации позволяет сделать эти искажения пренебрежимо малыми.

## Прямые методы синтеза

Название "прямые методы" означает, что в данном случае не используется аналоговый прототип. Исходными данными для синтеза служат какие-либо параметры фильтра (чаще всего — его АЧХ), которые могут задаваться, вообще говоря, произвольно.

Прямые методы синтеза можно разделить на две категории:

- *оптимальные* методы, в которых численными итерационными методами ищется минимум заданной функции качества;
- *субоптимальные* методы, не дающие в точности оптимального решения, но позволяющие значительно упростить вычисления по сравнению с оптимальными методами. Как правило, эти методы используют специфику решаемой задачи, например, дробно-рациональный вид функции передачи рекурсивного фильтра или экспоненциальный вид отдельных слагаемых его импульсной характеристики.

### Оптимальные методы

Чаще всего отправной точкой при расчете фильтра служит его желаемая частотная характеристика — либо АЧХ (когда фазовые характеристики не важны), либо комплексный коэффициент передачи. В качестве минимизируемой меры отклонения характеристики фильтра от заданной в общем случае используется  $p$ -норма ошибки (см. формулу (1.39) в разд. "Пространство сигналов" главы 1). Чаще всего используются два значения  $p$ :  $p = 2$  (квадратическая ошибка, см. формулу (1.40)) и  $p = \infty$  (максимальное абсолютное значение ошибки).

Поскольку корень  $p$ -й степени, входящий в (1.39), при любом  $p$  является монотонно возрастающей функцией, при расчете минимизируемой величины его можно не вычислять.

Если при синтезе фильтра нас интересует только его АЧХ, минимизируемая функция при использовании  $p$ -нормы ошибки рассчитывается следующим образом:

$$L_p = \int_0^{\omega_d} w(\omega) |D(\omega) - |\dot{K}(\omega)||^p d\omega, \quad (6.6)$$

где  $D(\omega)$  — желаемая АЧХ,  $|\dot{K}(\omega)|$  — АЧХ фильтра,  $w(\omega)$  — неотрицательная вещественная весовая функция. Использование весовой функции позволяет придать разную значимость различным участкам частотной оси. В частности, это дает воз-

можность задать переходные зоны, поведение АЧХ в которых не имеет значения. В этих зонах значение весовой функции должно быть нулевым.

Если необходимо аппроксимировать заданную частотную зависимость комплексного коэффициента передачи, норма ошибки аппроксимации рассчитывается так:

$$L_p = \int_0^{\omega_d} w(\omega) |\dot{D}(\omega) - \dot{K}(\omega)|^p d\omega. \quad (6.7)$$

Здесь  $\dot{D}(\omega)$  — желаемая комплексная частотная характеристика,  $\dot{K}(\omega)$  — частотная характеристика реального фильтра.

Для рекурсивных фильтров минимизация критерия (6.6) или (6.7) требует применения методов численной оптимизации (см., например, [34]). В случае же нерекурсивных фильтров, коэффициенты которых связаны с частотной характеристикой линейно, возможны более простые подходы. Рассмотрим подробнее процедуры синтеза оптимальных нерекурсивных фильтров по критерию (6.7) при  $p = 2$  и  $p = \infty$ .

## Минимизация квадратической ошибки ( $p = 2$ )

Подставив в (6.7) формулу (4.7), связывающую импульсную и частотную характеристики дискретного фильтра, и приняв  $p = 2$ , для нерекурсивного фильтра порядка  $N$  получим следующее:

$$L_2 = \int_0^{\omega_d} w(\omega) \left| \dot{D}(\omega) - \sum_{k=0}^N b_k e^{-j\omega kT} \right|^2 d\omega \rightarrow \min.$$

Элементарные преобразования этой формулы позволяют переписать ее так:

$$\begin{aligned} L_2 = & \int_0^{\omega_d} w(\omega) |\dot{D}(\omega)|^2 d\omega + \sum_{k_1=0}^N \sum_{k_2=0}^N b_{k_1} b_{k_2}^* \int_0^{\omega_d} w(\omega) e^{-j\omega(k_1-k_2)T} d\omega - \\ & - \sum_{k=0}^N b_k \int_0^{\omega_d} w(\omega) \dot{D}^*(\omega) e^{-j\omega kT} d\omega - \sum_{k=0}^N b_k^* \int_0^{\omega_d} w(\omega) \dot{D}(\omega) e^{j\omega kT} d\omega \rightarrow \min. \end{aligned} \quad (6.8)$$

Первое слагаемое не зависит от коэффициентов фильтра и не влияет на результат оптимизации. Для интегралов, входящих в остальные слагаемые, введем следующие обозначения:

$$\dot{v}_k = \int_0^{\omega_d} w(\omega) e^{j\omega kT} d\omega, \quad \dot{u}_k = \int_0^{\omega_d} w(\omega) \dot{D}(\omega) e^{j\omega kT} d\omega. \quad (6.9)$$

Значения  $\dot{v}_k$  с точностью до постоянного множителя представляют собой коэффициенты разложения в ряд Фурье весовой функции  $w(\omega)$ , а значения  $\dot{u}_k$  — аналогичные коэффициенты для взвешенной желаемой частотной характеристики  $w(\omega)\dot{D}(\omega)$  (см. разд. "Некоторые идеализированные фильтры" главы 4).

С использованием обозначений (6.9) формула (6.8) принимает вид

$$L_2 = \text{const} + \sum_{k_1=0}^N \sum_{k_2=0}^N b_{k_1} b_{k_2}^* \dot{v}_{k_2-k_1} - \sum_{k=0}^N b_k \dot{u}_k^* - \sum_{k=0}^N b_k^* \dot{u}_k \rightarrow \min, \quad (6.10)$$

или, в матричном виде,

$$L_2 = \text{const} + \mathbf{b}^H \mathbf{V} \mathbf{b} - \mathbf{u}^H \mathbf{b} - \mathbf{b}^H \mathbf{u} \rightarrow \min, \quad (6.11)$$

где  $\mathbf{V}$  — теплицева квадратная матрица, составленная из элементов  $\dot{v}_k$ , а  $\mathbf{u}$  и  $\mathbf{b}$  — векторы-столбцы из элементов  $\dot{u}_k$  и  $b_k$  соответственно:

$$\mathbf{V} = \begin{bmatrix} \dot{v}_0 & \dot{v}_{-1} & \dots & \dot{v}_{-N} \\ \dot{v}_1 & \dot{v}_0 & \dots & \dot{v}_{-N+1} \\ \vdots & \vdots & \ddots & \vdots \\ \dot{v}_N & \dot{v}_{N-1} & \dots & \dot{v}_0 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \dot{u}_0 \\ \dot{u}_1 \\ \vdots \\ \dot{u}_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix}.$$

Дифференцирование формулы (6.10) по  $b_i$  и приравнивание этих частных производных к нулю дает систему линейных уравнений относительно искомых коэффициентов фильтра. В матричном виде эта система выглядит следующим образом:

$$\mathbf{V} \mathbf{b} = \mathbf{u},$$

а вектор оптимальных коэффициентов фильтра при невырожденной матрице  $\mathbf{V}$  определяется по формуле

$$\mathbf{b} = \mathbf{V}^{-1} \mathbf{u}.$$

Следует отметить, что при  $w(\omega) = 1$  матрица  $\mathbf{V}$  становится диагональной ( $\mathbf{V} = (2\pi/T)\mathbf{I}$ ), следовательно,

$$\mathbf{b} = \frac{T}{2\pi} \mathbf{u}.$$

Этот результат вполне предсказуем, ведь усеченный ряд Фурье минимизирует именно квадратическую ошибку представления сигнала (см. разд. "Пространство сигналов" главы 1). Таким образом, если весовая функция  $w(\omega)$  постоянна на всех частотах, искомые коэффициенты фильтра представляют собой просто усеченную до конечной длины импульсную характеристику соответствующего идеального фильтра (см. формулу (4.34) в главе 4).

В заключение следует отметить, что минимизация квадратической ошибки приводит к неравномерным пульсациям АЧХ, уровень которых возрастает вблизи частот, где желаемая частотная характеристика изменяется скачкообразно. Это связано с эффектом Гиббса, который рассматривался в главе 1 при обсуждении рядов Фурье. Введение переходных полос, для которых  $w(\omega) = 0$ , позволяет уменьшить уровень пульсаций на краях полос пропускания и задерживания, но эти пульсации все равно остаются неравномерными. Пример этого можно увидеть в настоящей главе далее, при описании функции `firls` (см. рис. 6.13).

## Минимаксная оптимизация ( $p = \infty$ )

При  $p = \infty$  норма ошибки (6.6) или (6.7) равна максимальному абсолютному отклонению характеристики от заданной. Минимизация этой нормы, т. е. решение *минимаксной* (*minimax*) оптимизационной задачи, приводит к тому, что упомянутое максимальное абсолютное отклонение достигается на *нескольких* частотах. Таким образом, данный подход приводит к фильтрам с равномерными пульсациями АЧХ (*equiripple filter*).

Для нерекурсивных фильтров эта задача может быть решена с помощью элегантного итерационного метода, основанного на *чебышевской аппроксимации с использованием алгоритма многократной замены Ремеза*. Соответствующий алгоритм расчета применительно к синтезу дискретных фильтров разработан Парксом (*T. W. Parks*) и Макклелланом (*J. H. McClellan*). Подробно его теоретические основы рассмотрены, например, в [8, 16, 19, 21], а мы в данном разделе продемонстрируем основные идеи на примере синтеза ФНЧ с линейной ФЧХ (тип симметрии I, см. табл. 4.1 в главе 4).

Алгоритм базируется на двух основных положениях:

- максимальное абсолютное отклонение частотной характеристики от заданной наблюдается в экстремумах АЧХ, а также на граничных частотах полос пропускания и задерживания, причем знаки отклонения чередуются;
- значение частотной характеристики на фиксированной частоте линейно зависит от коэффициентов фильтра.

Таким образом, если мы знаем значения частот (их называют *экстремальными*), на которых имеет место максимальное отклонение частотной характеристики от заданной, то для нахождения коэффициентов фильтра достаточно решить систему линейных уравнений.

Сущность рассматриваемого алгоритма заключается в итерационном поиске экстремальных частот. После задания начального приближения циклически выполняются следующие действия:

- составляется и решается система линейных уравнений для нахождения коэффициентов фильтра, дающих частотную характеристику, отклонения которой от заданной имеют на экстремальных частотах одинаковый модуль и чередующиеся знаки;
- рассчитывается частотная характеристика получившегося фильтра и определяются положения ее локальных экстремумов. В результате формируется новый набор экстремальных частот.

Перечисленные шаги выполняются до тех пор, пока изменение экстремальных частот не станет меньше заданного порога.

Порядок решаемой системы уравнений должен быть равен  $N/2 + 2$  ( $N$  — порядок фильтра), т. к. в число неизвестных входят  $N/2 + 1$  коэффициентов симметричного фильтра и сама величина абсолютного отклонения от заданной характеристики.

Перейдем к обещанному примеру. Пусть требуется синтезировать ФНЧ 32-го порядка с верхней границей полосы пропускания 0,2 и нижней границей полосы задерживания 0,25 (частоты нормированы к частоте Найквиста). Сформируем начальное приближение для набора из  $32/2 + 2 = 18$  экстремальных частот, включив в их число, как уже упоминалось, границы полос пропускания и задерживания, а остальные 16 частот выбрав произвольно:

```
>> fp = 0.2; % граница полосы пропускания
>> fs = 0.25; % граница полосы задерживания
>> N = 32; % порядок фильтра
>> ff = [0.04:0.04:0.2 0.25:0.06:0.97]'; % начальный набор частот
```

Теперь займемся системой уравнений. Для удобства будем считать, что коэффициенты искомого фильтра пронумерованы не от 0 до  $N$ , а от  $-N/2$  до  $N/2$ , тогда его частотная характеристика благодаря симметрии фильтра становится вещественной и может быть записана так:

$$K(\omega) = \sum_{k=-N/2}^{N/2} b_k e^{-j\omega kT} = b_0 + 2 \sum_{k=1}^{N/2} b_k \cos(\omega kT). \quad (6.12)$$

На экстремальных частотах  $\omega_n$  эта формула должна давать результат, отличающийся от желаемой характеристики  $D(\omega)$  на некоторую неизвестную, но одинаковую для всех  $n$  (после умножения упомянутого расхождения на весовую функцию  $w(\omega_n)$ ) величину  $d$ , причем знаки отклонений должны чередоваться. Поэтому решаемая система уравнений будет иметь вид

$$b_0 + 2 \sum_{k=1}^{N/2} b_k \cos(\omega_n kT) + \frac{(-1)^n}{w(\omega_n)} d = D(\omega_n), \quad n = 1, \dots, \frac{N}{2} + 2.$$

Множитель  $(-1)^n$  отвечает в этой формуле за чередование знаков отклонения (предполагается, что набор частот  $\{\omega_n\}$  является упорядоченным). Реализуем решение этой системы в MATLAB (весовая функция  $w(\omega)$  в нашем примере на всех экстремальных частотах равна единице):

```
>> % формируем матрицу системы
>> A = 2*cos(pi*ff*(0:N/2)); % набор необходимых косинусов
>> A(:, 1) = 1; % коррекция для первого столбца
>> A(1:2:end, N/2+2) = 1; % формируем последний столбец
>> A(2:2:end, N/2+2) = -1; % с чередованием знаков
>> D = ff <= fp; % желаемая частотная характеристика
>> b0 = A\D; % решение системы уравнений
>> d = b0(end); % найденное равномерное отклонение
>> b0 = b0(1:end-1); % половина симметричной импульсной x-ки фильтра
```

### ЗАМЕЧАНИЕ

Здесь система линейных уравнений решена общим методом. На самом деле алгоритм Паркса — Макклеллана включает в себя и особый метод решения этой системы, учитывающий ее специфику.

Теперь выведем график частотной характеристики получившегося фильтра и покажем, что на использованном для расчета наборе частот ее отклонения от желаемой действительно знакопеременные и одинаковые по абсолютной величине (рис. 6.3):

```
>> b = [flipud(b0)' b0(2:end)']; % полный набор коэффициентов фильтра
>> [hh, f] = zerophase(b, 1, [], 2); % частотная характеристика фильтра
>> plot(f, hh) % график частотной характеристики
>> hold on
>> plot(ff, D'-d*(-1).^(0:N/2+1), 'o') % знакопеременные отклонения
>> grid on
>> hold off
```

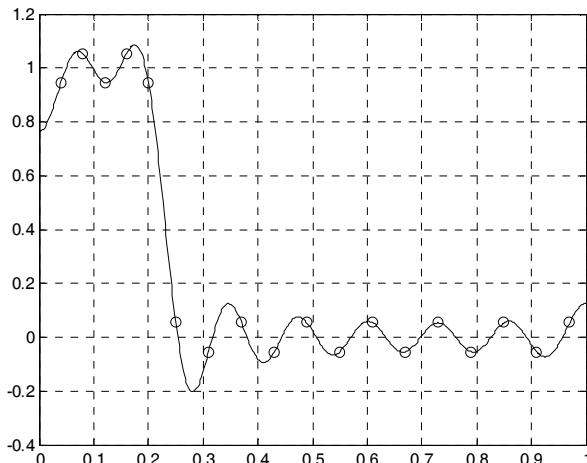


Рис. 6.3. Минимаксная оптимизация ФНЧ — первая итерация

Из графика видно, что на заданных частотах отклонение характеристики от желаемой ведет себя как нужно, но эти частоты не являются экстремумами. Определим положения экстремумов и используем их в качестве нового набора экстремальных частот:

```
>> [~, indmax] = findpeaks(hh); % индексы максимумов
>> [~, indmin] = findpeaks(-hh); % индексы минимумов
>> ff = sort([f(indmax indmin)]; fp; fs]); % сортировка частот
```

Мы нашли положения локальных максимумов и минимумов частотной характеристики с помощью функции `findpeaks` и добавили в набор частот границы полос пропускания и задерживания. Однако легко убедиться, что получившийся вектор `ff` содержит лишь 16 элементов, поэтому придется добавить в него нулевую частоту и частоту Найквиста, т. к. на этих частотах частотная характеристика тоже имеет экстремумы (см. рис. 6.3):

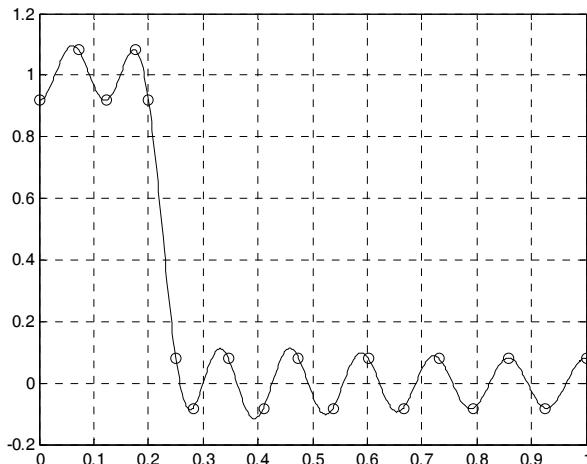
```
>> ff = [0; ff; 1]; % добавляем нулевую частоту и частоту Найквиста
```

### ЗАМЕЧАНИЕ

Добавление нулевой частоты и/или частоты Найквиста в набор экстремальных частот является неизбежным, т. к. частотная характеристика (6.12) может быть после тригонометрических преобразований представлена в виде полинома степени  $N/2$  относительно  $\cos(\omega T)$ , а полином такой степени может иметь максимум  $N/2 - 1$  экстремум. Поэтому, чтобы получить  $N/2 + 2$  экстремальных частот, помимо краев полос пропускания и задерживания обязательно придется задействовать нулевую частоту и/или частоту Найквиста. Экстремумы функции (6.12) на этих двух частотах имеют место всегда, т. к. они обусловлены симметрией частотной характеристики вещественного фильтра.

Новое приближение для экстремальных частот сформировано, теперь снова решаем систему линейных уравнений и выводим графики (рис. 6.4). Соответствующий код приводится уже без комментариев и в компактном формате:

```
>> A = 2*cos(pi*ff*(0:N/2)); A(:, 1) = 1;
>> A(1:2:end, N/2+2) = 1; A(2:2:end, N/2+2) = -1;
>> D = ff <= fp; b0 = A\D;
>> d = b0(end); b0 = b0(1:end-1); b = [flipud(b0)' b0(2:end)'];
>> [hh, f] = zerophase(b, 1, [], 2);
>> plot(f, hh); hold on; plot(ff, D'-d*(-1).^(0:N/2+1), 'o'); hold off
>> grid on
```



**Рис. 6.4.** Минимаксная оптимизация ФНЧ — вторая итерация

График показывает, что уже на второй итерации погрешности нахождения экстремумов оказались невелики. При дальнейших итерациях они еще больше уменьшаются, а частотная характеристика приближается к окончательной, вид которой показан в этой главе далее, при описании функции `firpm` (см. рис. 6.15).

### ЗАМЕЧАНИЕ

С помощью другой функции MATLAB, `firgr`, можно, помимо прочего, получить информацию о числе итераций, потребовавшихся для синтеза фильтра данным мето-

дом. При синтезе фильтра, рассматриваемого в данном примере, алгоритм завершает работу после пяти итераций.

## Субоптимальные методы

Как уже говорилось, субоптимальные методы позволяют упростить вычисления за счет учета специфики решаемой задачи. При этом, в частности, может использоваться тот факт, что коэффициенты числителя функции передачи фильтра *линейно* связаны с его комплексной частотной характеристикой и потому влияют на ее форму значительно слабее, чем коэффициенты знаменателя.

Кроме того, для поиска коэффициентов знаменателя часто используются методы авторегрессионного анализа (см. разд. "Авторегрессионная модель" главы 5).

Вот перечень субоптимальных методов синтеза, реализованных в MATLAB (пакет Signal Processing):

- метод, основанный на решении системы уравнений Юла — Уолкера (Yule — Walker) для поиска коэффициентов знаменателя функции передачи фильтра;
- метод идентификации частотной характеристики, в котором минимизируется норма разности между *числителем* функции передачи и *произведением* желающей частотной характеристики и знаменателя функции передачи фильтра;
- аппроксимация заданной импульсной характеристики с помощью метода экспоненциального оценивания Прони (данный алгоритм был разработан Гаспаром Риие (бароном де Прони) в 1795 г. для подгонки экспоненциальной модели под экспериментальные данные при исследовании физических свойств газов);
- аппроксимация заданной импульсной характеристики с помощью итерационного метода Штейглица — МакБрайда;
- метод, основанный на использовании весовых функций (окон). Данный алгоритм более подробно рассмотрен далее, в разд. "Синтез с использованием окон" этой главы;
- синтез ФНЧ с косинусоидальным сглаживанием АЧХ является частным случаем использования окон; данный метод также подробно рассмотрен далее (см. разд. "Фильтры с косинусоидальным сглаживанием АЧХ" этой главы).

Дополнительные сведения о конкретных субоптимальных методах приведены далее, при описании соответствующих функций MATLAB.

## Субоптимальный синтез нерекурсивных фильтров

В данном разделе подробно рассмотрена общая идея синтеза нерекурсивных фильтров с использованием окон, а также частный случай, связанный с применением конкретного окна и приводящий к фильтрам с косинусоидальным сглаживанием АЧХ.

## Синтез с использованием окон

Данный метод предназначен для синтеза нерекурсивных фильтров. Идея его очень проста. Прежде всего мы задаем желаемый комплексный коэффициент передачи в виде непрерывной функции, определенной в диапазоне частот от нуля до частоты Найквиста (если синтезируется вещественный фильтр) или до частоты дискретизации (если проектируется комплексный фильтр). Обратное преобразование Фурье этой характеристики, вычисленное с учетом ее периодического характера, даст бесконечную в обе стороны последовательность отсчетов импульсной характеристики. Для получения реализуемого нерекурсивного фильтра заданного порядка эта последовательность усекается — из нее выбирается центральный фрагмент нужной длины.

Простое усечение последовательности отсчетов импульсной характеристики соответствует использованию *прямоугольного окна*. Из-за усечения первоначально заданная частотная характеристика искажается — она сворачивается со спектром окна. В результате появляются переходные полосы между областями пропускания и задерживания, наблюдаются колебания коэффициента передачи в полосах пропускания, а в полосах задерживания АЧХ, как правило, приобретает лепестковый характер.

Для ослабления перечисленных эффектов, и прежде всего для уменьшения уровня лепестков в полосах задерживания, усеченная импульсная характеристика умножается на весовую функцию (окно), плавно спадающую к краям.

Мы уже говорили о весовых функциях применительно к спектральному анализу (см. разд. "Весовые функции" главы 5) и видели, что различные используемые на практике окна имеют разный уровень боковых лепестков. Однако величина боковых лепестков собственного спектра окна не совпадает с величиной лепестков АЧХ фильтра, синтезированного с применением данного окна. Убедимся в этом на простом примере, используя различные окна для синтеза нерекурсивного ФНЧ 32-го порядка с частотой среза, равной  $1/4$  от частоты Найквиста.

Прежде всего рассчитываем бесконечную импульсную характеристику. Обратное преобразование Фурье от интересующей нас АЧХ (ФЧХ считаем нулевой) дает следующее:

$$h(k) = \frac{1}{4} \frac{\sin(\pi k / 4)}{\pi k / 4}, \quad -\infty < k < \infty. \quad (6.13)$$

Для получения фильтра 32-го порядка выбираем 33 отсчета этой импульсной характеристики, расположенных симметрично относительно нуля, т. е. используем  $k = -16 \dots 16$ . Выбор симметричного фрагмента бесконечной импульсной характеристики позволяет получить фильтр с линейной ФЧХ и, следовательно, постоянной групповой задержкой, равной (в отсчетах) половине порядка фильтра.

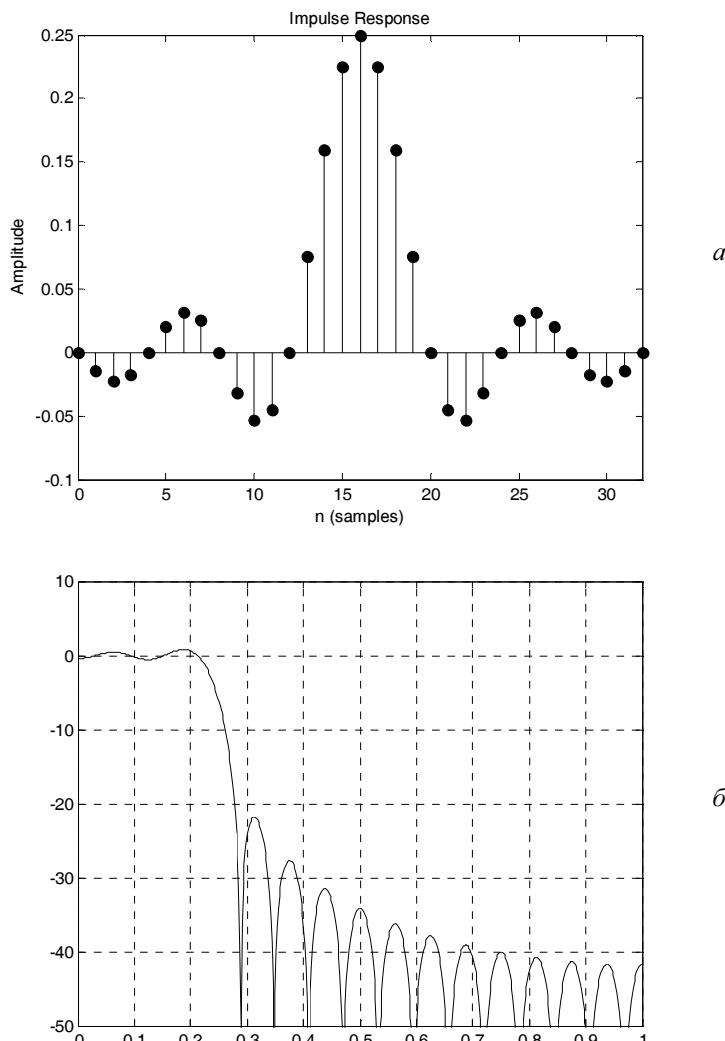
Урезав импульсную характеристику, мы фактически использовали прямоугольное окно. Построим графики импульсной характеристики полученного фильтра и его АЧХ (рис. 6.5):

```

>> k = (-16:16)';
>> b = sinc(k/4)/4;
>> impz(b) % график импульсной характеристики
>> figure
>> [h, f] = freqz(b, 1, [], 2);
>> plot(f, 20*log10(abs(h))) % график АЧХ в децибелах
>> ylim([-50 10])
>> grid on

```

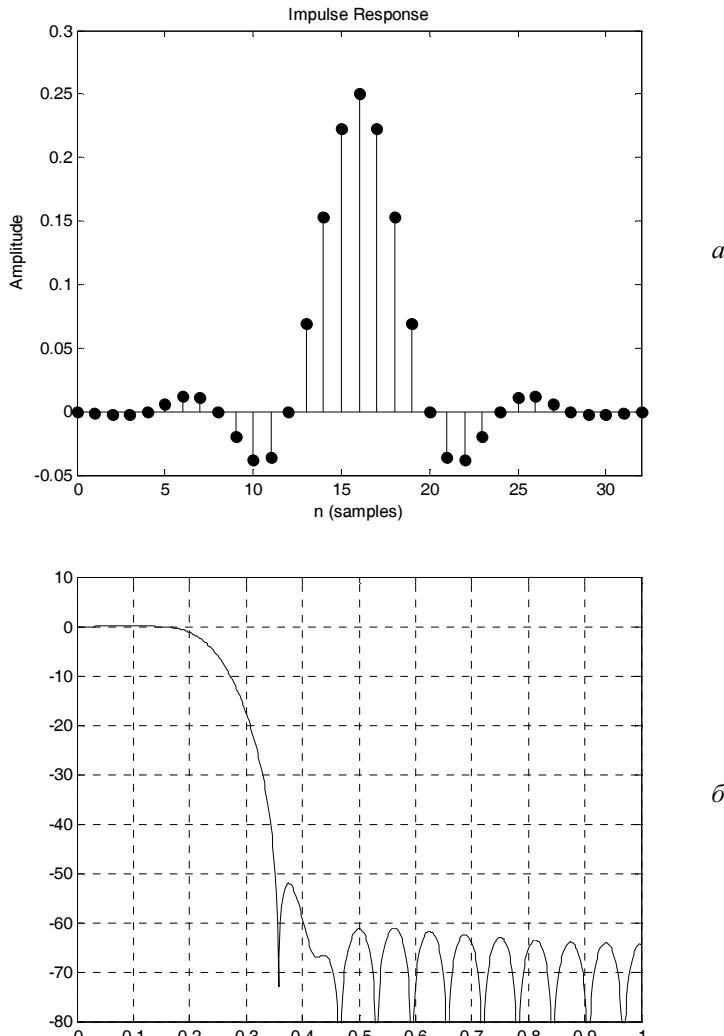
Из графика АЧХ видно, что уровень боковых лепестков составляет  $-21$  дБ, а их ширина равна  $1/32$  от частоты дискретизации.



**Рис. 6.5.** Импульсная характеристика (*а*) и АЧХ (*б*) ФНЧ, синтезированного с использованием прямоугольного окна

Теперь воспользуемся какой-либо весовой функцией, например окном Хэмминга, которое используется по умолчанию при синтезе фильтров данным методом в MATLAB (функции `firl` и `fir2`). Результаты показаны на рис. 6.6:

```
>> b1 = b .* hamming(33);
>> impz(b1) % график импульсной характеристики
>> figure
>> [h1, f] = freqz(b1, 1, [], 2);
>> plot(f, 20*log10(abs(h1))) % график АЧХ в децибелах
>> ylim([-80 10])
>> grid on
```



**Рис. 6.6.** Импульсная характеристика (а) и АЧХ (б) ФНЧ, синтезированного с использованием окна Хэмминга

Как видите, уровень лепестков в полосе задерживания упал примерно до  $-52$  дБ за счет некоторого расширения полосы пропускания.

Сведения об уровне лепестков, достигаемом при синтезе ФНЧ с использованием разных окон, сведены в табл. 6.1. Результаты получены с помощью MATLAB при синтезе фильтра 256-го порядка с частотой среза, равной  $1/4$  от частоты Найквиста. Уровень лепестков в некоторой степени зависит от порядка фильтра и его частоты среза, поэтому данные, приводимые в различных литературных источниках, могут несколько различаться. Этот эффект мы только что наблюдали: АЧХ, приведенная на рис. 6.6, имеет уровень боковых лепестков  $-52$  дБ (фильтр 32-го порядка), а в табл. 6.1 для окна Хэмминга указана величина  $-53,8$  дБ (фильтр 256-го порядка).

Кроме уровня боковых лепестков, в той же табл. 6.1 показано, как в зависимости от типа окна меняется ширина переходной зоны между полосами пропускания и задерживания. Для этого приведены нормированные значения частоты среза фильтра (по уровню  $-3$  дБ) и частоты, на которой коэффициент передачи первый раз принимает значение, соответствующее уровню боковых лепестков. Ширина переходной зоны вычисляется как разность между этими значениями.

**Таблица 6.1.** Уровень боковых лепестков и границы переходной зоны при синтезе ФНЧ с окнами разного типа

Тип окна	Уровень боковых лепестков, дБ	Конец полосы пропускания по уровню $-3$ дБ	Начало полосы задерживания по уровню боковых лепестков	Ширина переходной зоны
Прямоугольное	-21,1	0,2483	0,2536	0,0053
Тьюки	-22,0	0,2478	0,2550	0,0072
Треугольное*	-26,5	0,2466	0,2653	0,0187
Бартлетта*	-26,5	0,2466	0,2655	0,0189
Ханна	-43,9	0,2466	0,2622	0,0156
Бартлетта — Ханна	-39,8	0,2466	0,2634	0,0168
Бомена*	-51,95	0,2458	0,2732	0,0274
Хэмминга	-53,8	0,2469	0,2630	0,0161
Парзена*	-56,6	0,2455	0,2809	0,0354
Гауссово	-59,2	0,2466	0,2703	0,0237
Блэкмена	-75,3	0,2460	0,2718	0,0258
Блэкмена — Харриса	-109,3	0,2453	0,2806	0,0353
Наттолла	-113,9	0,2453	0,2803	0,0350
Кайзера при $\beta = 4$	-45,2	0,2472	0,2601	0,0129
Кайзера при $\beta = 9$	-90,6	0,2459	0,2724	0,0265

Таблица 6.1 (окончание)

Тип окна	Уровень боковых лепестков, дБ	Конец полосы пропускания по уровню -3 дБ	Начало полосы задерживания по уровню боковых лепестков	Ширина переходной зоны
Чебышева при $\beta = 40$ дБ	-51,1	0,2471	0,2614	0,0143
Чебышева при $\beta = 60$ дБ	-71,6	0,2465	0,2680	0,0215
Чебышева при $\beta = 80$ дБ	-92,4	0,2459	0,2736	0,0277
Окно с плоской вершиной	-90,7	0,2425	0,2857	0,0432

**ЗАМЕЧАНИЕ**

В случае треугольного окна, а также окон Бартлетта, Бомена и Парзена АЧХ фильтра не имеет боковых лепестков — она спадает монотонно, но при этом имеет вид скругленных "ступенек". Для этих окон уровень лепестков и начало полосы задерживания измерены в точке перегиба АЧХ, соответствующей первой "ступеньке".

В случае окон Кайзера и Чебышева результаты зависят от дополнительных параметров этих окон. Для окна Кайзера есть эмпирическая формула, связывающая параметр  $\beta$  с уровнем подавления боковых лепестков  $\alpha$  (в дБ) при синтезе ФНЧ (см., например, [16]):

$$\beta = \begin{cases} 0, & \alpha < 21, \\ 0,5842(\alpha - 21)^{0,4} + 0,07886(\alpha - 21), & 21 \leq \alpha \leq 50, \\ 0,1102(\alpha - 8,7), & \alpha > 50. \end{cases} \quad (6.14)$$

Результаты из табл. 6.1 графически представлены на рис. 6.7. Рисунок показывает, что по соотношению "Уровень боковых лепестков/ширина переходной зоны" вне конкуренции окна Кайзера (на графике — кружочки) и Чебышева (на графике — крестики), обеспечивающие практически идентичные показатели. К тому же у этих окон имеется настраиваемый параметр, позволяющий регулировать соотношение между указанными величинами. Основным же недостатком этих окон является трудоемкость их расчета. Близкие характеристики обеспечивает окно Хэмминга (соответствующая ему точка лежит практически на линии, образованной точками для окон Кайзера и Чебышева), но оно дает единственное сочетание уровня боковых лепестков и ширины переходной зоны.

Однако следует помнить, что уровень боковых лепестков и ширина переходной зоны — не единственные параметры, характеризующие качество фильтра. Помимо этого, разным окнам соответствует разный уровень пульсаций АЧХ в полосе пропускания. Так, например, окно с плоской вершиной, выглядящее безнадежно плохим в табл. 6.1 и на рис. 6.7, может обеспечивать, в отличие от многих других окон,

очень малые колебания АЧХ в полосе пропускания (сотые или тысячные доли дБибела).

В заключение данного раздела следует отметить, что при смене используемого окна одновременно меняются уровень боковых лепестков, ширина переходной зоны и интенсивность пульсаций АЧХ фильтра в полосе пропускания. Поэтому выбирать окно следует исходя из полного комплекса требований, предъявляемых к параметрам фильтра.

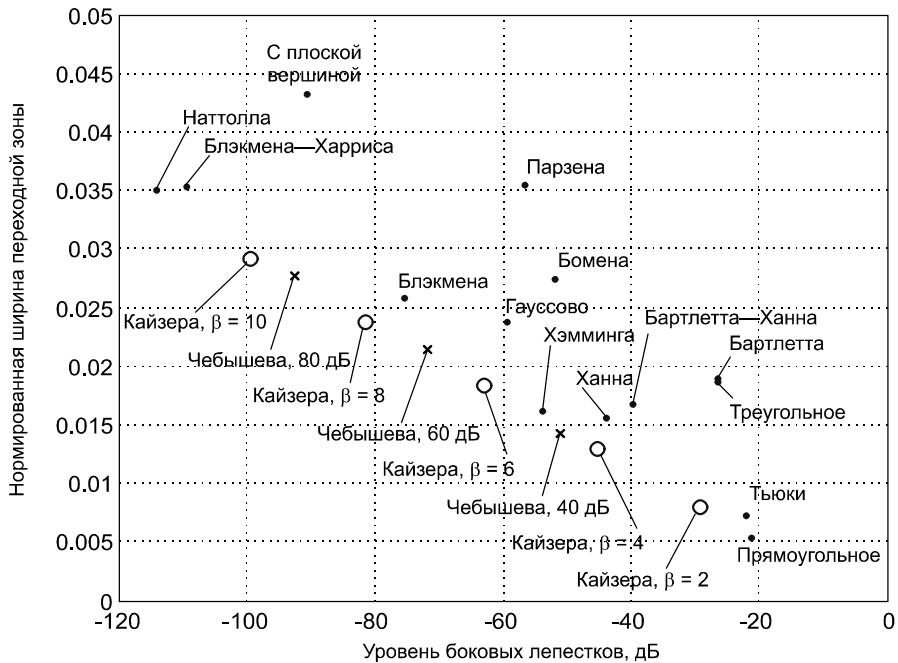


Рис. 6.7. Параметры ФНЧ, синтезированных с использованием окон разного типа

## Фильтры с косинусоидальным сглаживанием АЧХ

Строго говоря, расчет фильтров с косинусоидальным сглаживанием АЧХ следует отнести к категории методов, основанных на весовых функциях. Однако мы рассматриваем эти фильтры отдельно по следующим двум причинам. Во-первых, они получили широкое распространение в телекоммуникационных системах (в частности, для формирования спектра при квадратурной манипуляции). Во-вторых, и это главное, данный частный метод удобно рассматривать с несколько иной точки зрения, чем использование окон вообще, а умение видеть различные трактовки одной и той же проблемы является важнейшим качеством современного специалиста в любой области.

Итак, пусть нам необходимо синтезировать ФНЧ с заданной частотой среза. Обсуждая использование окон, мы видели, что обратное преобразование Фурье от идеальной прямоугольной АЧХ дает отсчеты бесконечной в обе стороны импульсной

характеристики вида  $\sin(ak)/(ak)$  (см. формулу (6.13)). Далее было показано, что простое усечение этой бесконечной импульсной характеристики плохо сказывается на АЧХ, поэтому необходимо использовать весовые функции. На сей раз задумаемся, от чего может зависеть степень проявления нежелательных эффектов при усечении бесконечных характеристик. Очевидный ответ — от скорости убывания отсчетов характеристики. Исходный вариант (6.13), соответствующий прямоугольной АЧХ, убывает пропорционально  $k$ . Это связано с наличием разрывов (скачков) у АЧХ (см. разд. "Примеры разложения сигналов в ряд Фурье" главы 1). Там идет речь о связи скорости убывания спектра с количеством непрерывных производных у сигнала, но благодаря дуальности прямого и обратного преобразований Фурье те же соотношения выполняются и в другую сторону).

Итак, увеличив скорость затухания импульсной характеристики, можно надеяться, что ее усечение скажется на виде АЧХ не столь катастрофически. Для ускорения затухания необходимо изменить синтезируемую АЧХ так, чтобы она не содержала разрывов. В этом разделе мы рассматриваем *косинусоидальное сглаживание*, при котором в переходной зоне от полосы пропускания к полосе задерживания АЧХ представляет собой половину периода косинуса. Такой фильтр называется *фильтром с косинусоидальным сглаживанием АЧХ (raised cosine filter)* и для аналогового случая описывается в частотной области следующим образом (рис. 6.8):

$$\dot{K}(\omega) = \begin{cases} 1, & |\omega| \leq \omega_0(1 - \alpha), \\ \frac{1}{2} - \frac{1}{2} \sin\left(\frac{\pi |\omega| - \omega_0}{\alpha \omega_0}\right), & \omega_0(1 - \alpha) < |\omega| \leq \omega_0(1 + \alpha), \\ 0, & |\omega| > \omega_0(1 + \alpha). \end{cases} \quad (6.15)$$

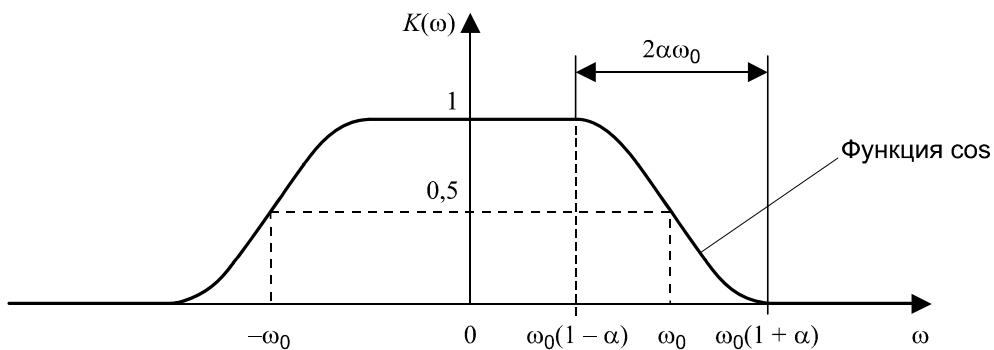


Рис. 6.8. АЧХ фильтра с косинусоидальным сглаживанием

Параметр  $\alpha$  называется *коэффициентом сглаживания (rolloff factor)*, он равен половине ширины переходной зоны, нормированной к частоте среза. При  $\alpha = 0$  фильтр превращается в идеальный ФНЧ с прямоугольной АЧХ, при  $\alpha = 1$  АЧХ перестает содержать плоский участок в полосе пропускания.

Обратное преобразование Фурье от формулы (6.15) дает выражение для импульсной характеристики ФНЧ с косинусоидальным сглаживанием АЧХ:

$$h(t) = \frac{\omega_0}{\pi} \frac{\cos(\alpha\omega_0 t)}{1 - \left(\frac{2\alpha\omega_0 t}{\pi}\right)^2} \frac{\sin(\omega_0 t)}{\omega_0 t}. \quad (6.16)$$

Использованный способ сглаживания делает непрерывной не только саму АЧХ, но и ее первую производную, поэтому полученная импульсная характеристика фильтра убывает пропорционально  $t^3$ .

### ЗАМЕЧАНИЕ

Неопределенность 0/0, имеющая место в (6.16) при  $\alpha\omega_0 t = \pm\pi/2$ , после вычисления пределов согласно правилу Лопитала раскрывается следующим образом:

$$h\left(\pm\frac{\pi}{2\alpha\omega_0}\right) = \frac{\omega_0\alpha}{2\pi} \sin\left(\frac{\pi}{2\alpha}\right).$$

Проверим, как будет выглядеть АЧХ дискретного фильтра с косинусоидальным сглаживанием после усечения импульсной характеристики. Пусть частота среза составляет 1/8 от частоты дискретизации (1/4 от частоты Найквиста). Тогда в (6.16) нужно подставить дискретные значения

$$t = kT = \frac{2\pi}{\omega_d} k = \frac{2\pi}{8\omega_0} k = \frac{\pi k}{4\omega_0}.$$

С учетом этого

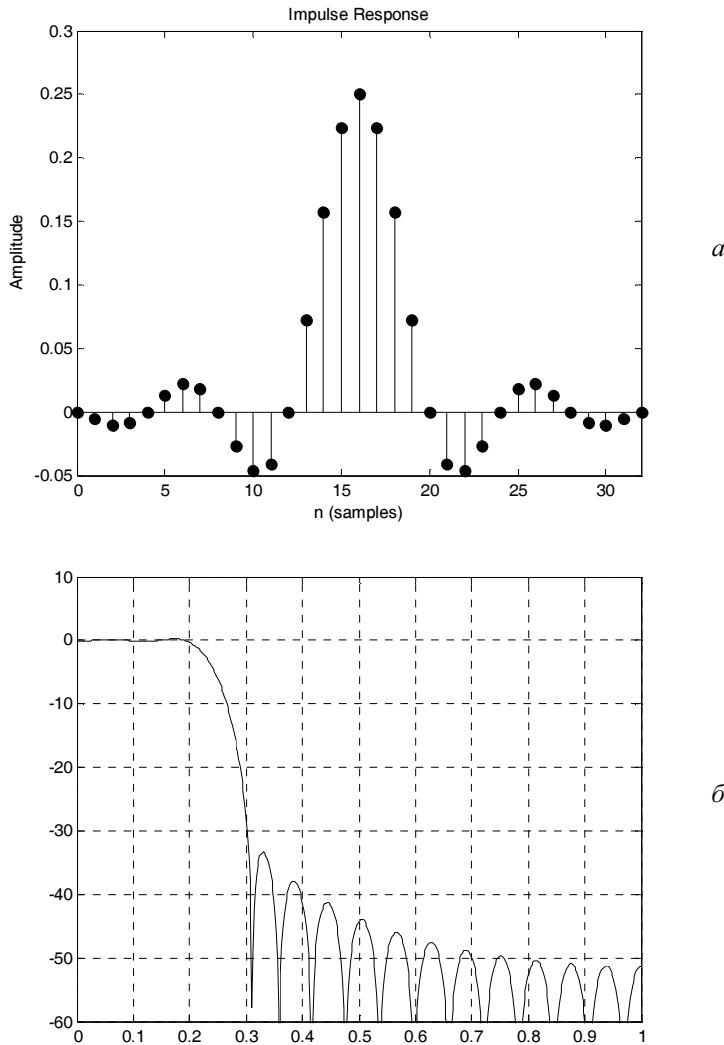
$$h(k) = T \frac{\omega_0}{\pi} \frac{\cos\left(\frac{\alpha\pi k}{4}\right)}{1 - \left(\frac{\alpha k}{2}\right)^2} \frac{\sin\left(\frac{\pi k}{4}\right)}{\frac{\pi k}{4}} = \frac{1}{4} \frac{\cos\left(\frac{\alpha\pi k}{4}\right)}{1 - \left(\frac{\alpha k}{2}\right)^2} \frac{\sin\left(\frac{\pi k}{4}\right)}{\frac{\pi k}{4}}.$$

Будем использовать коэффициент сглаживания  $\alpha = 0,25$  и рассчитаем фильтр 32-го порядка, взяв  $k$  в диапазоне  $-16 \dots 16$  (рис. 6.9):

```
>> k = -16:16;
>> alpha = 0.25; % коэффициент сглаживания
>> w = cos(alpha*pi*k/4) ./ (1 - (alpha*k/2).^2);
>> w(isinf(w)) = pi/4; % устранение неопределенностей
>> b = w .* sinc(k/4) / 4;
>> impz(b) % график импульсной характеристики
>> figure
>> [h, f] = freqz(b, 1, [], 2);
>> plot(f, 20*log10(abs(h))) % график АЧХ в децибелах
>> ylim([-60 10])
>> grid on
```

Сравнение рис. 6.9 и 6.5, где были приведены результаты простого усечения импульсной характеристики идеального ФНЧ, показывает, что импульсная характеристи-

стика фильтра с косинусоидальным сглаживанием затухает быстрее, а уровень боковых лепестков АЧХ уменьшился более чем на 10 дБ (до -33,3 дБ) без заметного расширения полосы пропускания. Пульсации коэффициента передачи в полосе пропускания в данном случае тоже оказываются слабее.



**Рис. 6.9.** Импульсная (сверху) и частотная (снизу) характеристики дискретного фильтра с косинусоидальным сглаживанием АЧХ

### SQRT-вариант

Фильтры с косинусоидальным сглаживанием АЧХ, как уже говорилось, часто используются в системах связи для формирования спектра сигнала. При этом используется "распределение" этого фильтра между передатчиком и приемником для реа-

лизации *согласованной фильтрации*. Тогда на передающей и приемной сторонах используются одинаковые фильтры, АЧХ которых показана на рис. 6.10 и представляет собой квадратный корень из формулы (6.15):

$$\dot{K}(\omega) = \begin{cases} 1, & |\omega| \leq \omega_0(1-\alpha), \\ \cos\left(\frac{\pi}{4\alpha\omega_0}(|\omega| - \omega_0(1-\alpha))\right), & \omega_0(1-\alpha) < |\omega| \leq \omega_0(1+\alpha), \\ 0, & |\omega| > \omega_0(1+\alpha). \end{cases}$$

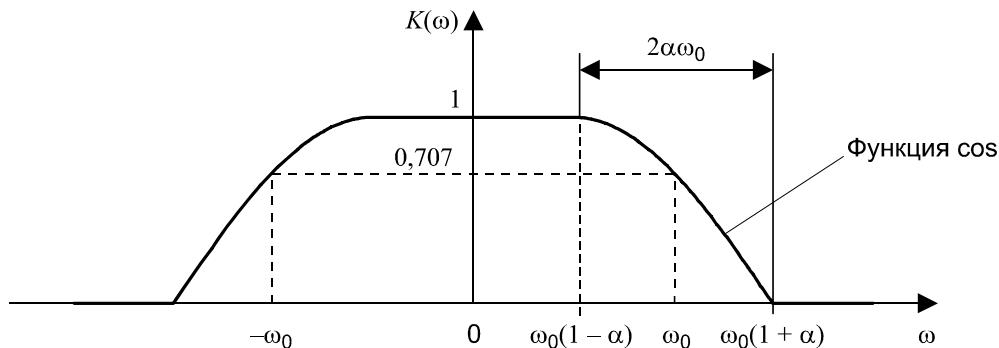


Рис. 6.10. АЧХ SQRT-варианта ФНЧ с косинусоидальным сглаживанием

Импульсная характеристика такого фильтра описывается следующей формулой:

$$h(t) = \frac{4\alpha\omega_0}{\pi^2 - (4\alpha\omega_0 t)^2} \left( \cos((1+\alpha)\omega_0 t) + \frac{\sin((1-\alpha)\omega_0 t)}{4\alpha\omega_0 t / \pi} \right). \quad (6.17)$$

Фильтр такого типа называется *SQRT-вариантом* фильтра с косинусоидальным сглаживанием АЧХ (*square root raised cosine filter*).

## Синтез дискретных фильтров в MATLAB

В MATLAB имеется несколько десятков функций синтеза дискретных фильтров. Большая их часть сосредоточена в пакете Signal Processing, три функции расчета конкретных фильтров имеются в пакете Communications, наконец, несколько редко используемых на практике методов добавляет специализированный пакет Filter Design.

Общие сведения о функциях расчета фильтров представлены в табл. 6.2. В столбце "Пакет" используются следующие обозначения:

- SP** — Signal Processing;
- Comm** — Communications;
- FD** — Filter Design.

Таблица 6.2. Функции синтеза дискретных фильтров

Функция	Пакет	Тип фильтра	АЧХ	Метод синтеза
butter	SP	Рекурсивный	Баттерворт	Билинейное $z$ -преобразование
cheby1	SP	Рекурсивный	Чебышева первого рода	Билинейное $z$ -преобразование
cheby2	SP	Рекурсивный	Чебышева второго рода	Билинейное $z$ -преобразование
ellip	SP	Рекурсивный	Золотарева — Кауэра (эллиптическая)	Билинейное $z$ -преобразование
bilinear	SP	Рекурсивный	Произвольный аналоговый прототип	Билинейное $z$ -преобразование
maxflat	SP	Рекурсивный или нерекурсивный	Обобщенный ФНЧ Баттерворт	Расчет положения нулей и полюсов
impinvar	SP	Рекурсивный	Произвольный аналоговый прототип	Инвариантное преобразование импульсной характеристики
yulewalk	SP	Рекурсивный	Кусочно-линейная	Авторегрессионный метод
invfreqz	SP	Рекурсивный	Произвольная	Минимизация разности между числителем функции передачи и произведением ее знаменателя и желаемой ЧХ
prony	SP	Рекурсивный	Синтез по заданной импульсной характеристике	Экспоненциальная аппроксимация Прони
stmcb	SP	Рекурсивный	Синтез по заданной импульсной характеристике	Минимизация среднеквадратической ошибки
firl	SP	Нерекурсивный	Многополосная	Обратное преобразование Фурье с использованием окон
fir2	SP	Нерекурсивный	Кусочно-линейная	Обратное преобразование Фурье с использованием окон
firls	SP	Нерекурсивный	Кусочно-линейная с переходными полосами	Минимизация среднеквадратической ошибки
fircls	SP	Нерекурсивный	Кусочно-постоянная	Минимизация среднеквадратической ошибки с ограничением максимального отклонения

Таблица 6.2 (продолжение)

Функция	Пакет	Тип фильтра	АЧХ	Метод синтеза
fircls1	SP	Нерекурсивный	ФНЧ, ФВЧ	Минимизация среднеквадратической ошибки с ограничением максимального отклонения
firrcos	SP	Нерекурсивный	ФНЧ	Косинусоидальное сглаживание
gaussfir	SP	Нерекурсивный	ФНЧ с гауссовой формой ИХ и АЧХ	Прямой расчет гауссовой импульсной характеристики
intfilt	SP	Нерекурсивный	Интерполятор (ФНЧ)	Минимаксная аппроксимация
firpm	SP	Нерекурсивный	Кусочно-линейная с переходными полосами	Минимаксная аппроксимация
cfirpm	SP	Нерекурсивный (в том числе с нелинейной ФЧХ и комплексными коэффициентами)	Кусочно-линейная с переходными полосами	Минимаксная аппроксимация
firgr	FD	Нерекурсивный	Кусочно-линейная с переходными полосами	Минимаксная аппроксимация
rcosfir	Comm	Нерекурсивный	ФНЧ	Косинусоидальное сглаживание
rcosiir	Comm	Рекурсивный	ФНЧ	Разложение по сингулярным числам матрицы Ганкеля, заполненной отсчетами импульсной характеристики
hilbiir	Comm	Рекурсивный	Фильтр Гильберта	Разложение по сингулярным числам матрицы Ганкеля, заполненной отсчетами импульсной характеристики
firlpnorm	FD	Нерекурсивный	Кусочно-линейная с переходными полосами	Минимизация $p$ -нормы ошибки
iirgrpdelay	FD	Рекурсивный	Синтез по заданной групповой задержке	Минимизация $p$ -нормы ошибки
iirlpnorm	FD	Рекурсивный	Кусочно-линейная с переходными полосами	Минимизация $p$ -нормы ошибки

Таблица 6.2 (окончание)

Функция	Пакет	Тип фильтра	АЧХ	Метод синтеза
iirlpnormc	FD	Рекурсивный	Кусочно-линейная с переходными полосами	Минимизация $p$ -нормы ошибки с ограничением модулей полюсов фильтра
fircband	FD	Нерекурсивный	Кусочно-линейная с переходными полосами	Минимаксная аппроксимация с ограничением величины пульсаций для отдельных частотных полос
firceqrip	FD	Нерекурсивный	ФНЧ, ФВЧ, в том числе с формой АЧХ вида $1/\text{sinc}$ в полосе пропускания	Минимаксная аппроксимация с дополнительными ограничениями
fireqint	FD	Нерекурсивный	Интерполятор (ФНЧ)	Минимаксная аппроксимация
firhalfband	FD	Нерекурсивный	ФНЧ или ФВЧ с частотой среза, равной половине частоты Найквиста	Минимаксная аппроксимация или обратное преобразование Фурье с использованием окон
firnyquist	FD	Нерекурсивный	ФНЧ с частотой среза, в целое число раз меньшей частоты Найквиста	Минимаксная аппроксимация или обратное преобразование Фурье с использованием окон

Кроме перечисленных функций в пакете Signal Processing имеется программа FDATool (Filter Design and Analysis Tool), реализующая графический интерфейс для расчета фильтров и просмотра их характеристик.

## Функции, использующие билинейное $z$ -преобразование

В пакете Signal Processing имеется функция `bilinear`, позволяющая синтезировать дискретный фильтр методом билинейного  $z$ -преобразования по произвольному аналоговому прототипу. Кроме того, имеются готовые функции расчета фильтров Баттервортса, Чебышева первого и второго рода, а также эллиптических фильтров (фильтров Золотарева — Кауэра). Свойства этих аналоговых прототипов были описаны в главе 2, там же были рассмотрены функции MATLAB для расчета аналоговых вариантов данных фильтров. При расчете дискретных фильтров используются те же самые функции, только без последнего параметра '`s`'; кроме того, частоты среза задаются нормированными к частоте Найквиста. Аналогичное "двойное значение" имеют и функции выбора порядка фильтра (см. разд. "Выбор порядка фильтра" главы 2).

В принципе, сказанного должно быть достаточно, чтобы читатель смог воспользоваться функциями синтеза дискретных фильтров, обратившись к материалу главы 2. Однако, чтобы сосредоточить информацию о синтезе дискретных фильтров в одном месте, кратко повторим синтаксис вызова функций и описание их параметров.

## Функции синтеза стандартных фильтров

Функции, о которых здесь пойдет речь, выполняют расчет дискретных ФНЧ, ФВЧ, полосовых и режекторных фильтров по аналоговым прототипам Баттервортса, Чебышева (первого и второго рода) и Золотарева — Кауэра методом билинейного  $z$ -преобразования. Требуемая при этом последовательность действий оформлена в виде следующих функций MATLAB:

- `butter(n, w0, type)` — расчет фильтров Баттервортса;
- `cheby1(n, Rp, w0, type)` — расчет фильтров Чебышева первого рода;
- `cheby2(n, Rs, w0, type)` — расчет фильтров Чебышева второго рода;
- `ellip(n, Rp, Rs, w0, type)` — расчет эллиптических фильтров (фильтров Золотарева — Кауэра).

Параметры всех функций задаются одинаково, поэтому обсуждать функции по отдельности не будем.

Перечисленные функции позволяют рассчитывать как дискретные, так и аналоговые фильтры. Признаком дискретного расчета служит отсутствие строки '`s`' в конце списка входных параметров. Использование этих функций для расчета аналоговых фильтров было рассмотрено в разд. "Расчет аналоговых фильтров" главы 2.

Параметры `n`, `Rp`, `Rs` (их состав зависит от типа фильтра) — это параметры фильтрапрототипа: `n` — порядок фильтра, `Rp` — уровень пульсаций АЧХ в полосе пропускания (в децибелах), `Rs` — уровень пульсаций АЧХ в полосе задерживания (в децибелах). Более подробное описание этих параметров было приведено в разделах главы 2, посвященных фильтрам-прототипам.

Параметры `w0` и `type` используются совместно для задания типа фильтра и значений его частот среза (нормированных к частоте Найквиста):

- ФНЧ: `w0` — скаляр, параметр `type='low'` или отсутствует;
- ФВЧ: `w0` — скаляр, `type='high'`;
- полосовой фильтр: `w0` — двухэлементный вектор частот среза `[w1 w2]`, параметр `type='bandpass'` или отсутствует;
- режекторный фильтр: `w0` — двухэлементный вектор частот среза `[w1 w2]`, `type='stop'`.

В зависимости от того, сколько выходных параметров указано при вызове, функции могут возвращать результаты расчета в виде коэффициентов полиномов числителя и знаменателя функции передачи (два выходных параметра), нулей и полюсов (три выходных параметра) либо параметров пространства состояний (четыре выходных параметра):

```
[a, b] = ...
[z, p, k] = ...
[A, B, C, D] = ...
```

С учетом всего сказанного перечислим действия, выполняемые функциями при расчете дискретных фильтров:

1. Производится расчет фильтра-прототипа с заданными параметрами АЧХ.
2. Полученные нули и полюсы преобразуются в параметры пространства состояний.
3. Производится преобразование фильтра-прототипа к требуемому типу с заданными частотами среза (частоты среза при этом корректируются по формуле (6.3)).
4. С помощью функции `bilinear` аналоговый фильтр преобразуется в дискретный.
5. Выполняется преобразование описания фильтра к виду, заданному при вызове функции.

## Функции выбора порядка фильтров

Рассмотренные выше функции расчета фильтров требуют задания в качестве входных параметров порядка фильтра и его частоты среза. При этом понятие частоты среза для фильтров разных типов определяется по-разному. Однако исходными данными при разработке фильтров, как правило, являются другие параметры: частотные границы полос пропускания и задерживания, а также допустимая неравномерность АЧХ в полосе пропускания и минимально необходимое затухание в полосе задерживания (см. рис. 2.19 в разд. "Выбор порядка фильтра" главы 2).

Выбрать минимально необходимый порядок фильтра позволяют следующие однотипные функции пакета Signal Processing:

```
[n, Wn] = buttord(Wp, Ws, Rp, Rs)
[n, Wn] = cheblord(Wp, Ws, Rp, Rs)
[n, Wn] = cheb2ord(Wp, Ws, Rp, Rs)
[n, Wn] = ellipord(Wp, Ws, Rp, Rs)
```

### **ЗАМЕЧАНИЕ**

Данные функции позволяют выбирать порядок как для дискретных, так и для аналоговых фильтров. Признаком дискретного расчета является отсутствие строки '`s`' в конце списка входных параметров. Использование этих функций для определения требуемого порядка аналоговых фильтров обсуждалось в разд. "Выбор порядка фильтра" главы 2.

Входной параметр `Rp` — допустимый уровень пульсаций в полосе пропускания (в децибелах), `Rs` — минимально необходимое затухание в полосе задерживания (в децибелах). Параметры `Wp` и `Ws` задают границы полос пропускания и задерживания (нормированные к частоте Найквиста), способ задания этих параметров зависит от типа проектируемого фильтра:

- ФНЧ:  $W_p$  и  $W_s$  — числа, при этом должно выполняться неравенство  $W_p < W_s$ ;
- ФВЧ:  $W_p$  и  $W_s$  — числа, при этом должно выполняться неравенство  $W_p > W_s$ ;
- полосовой фильтр:  $W_p$  и  $W_s$  — двухэлементные векторы, при этом должны выполняться неравенства  $W_s(1) < W_p(1) < W_p(2) < W_s(2)$ ;
- режекторный фильтр:  $W_p$  и  $W_s$  — двухэлементные векторы, при этом должны выполняться неравенства  $W_p(1) < W_s(1) < W_s(2) < W_p(2)$ .

Выходными параметрами являются минимально необходимый для выполнения заданных требований порядок фильтра  $n$  и частота среза фильтра  $W_n$  (нормированная к частоте Найквиста). Эти параметры должны затем использоваться при вызове функции расчета фильтра. Возврат значения  $W_n$  избавляет пользователя от забот, связанных с тем, что при расчете разных фильтров понятие частоты среза имеет разный смысл.

Поскольку порядок фильтра — величина целочисленная, то обычно оказывается, что фильтр минимально необходимого порядка обеспечивает некоторый запас по исходным параметрам. Функции выбора порядка фильтра при дискретном варианте расчета используют этот запас точно так же, как в аналоговом случае (см. разд. "Выбор порядка фильтра" главы 2) — для фильтров Баттерворта и Чебышева первого рода будет увеличиваться затухание в полосе задерживания, для фильтров Чебышева второго рода — уменьшаться пульсации в полосе пропускания, а для эллиптических фильтров — расширяться полоса задерживания.

В качестве примера зададим очень жесткие требования к АЧХ фильтра низких частот — узкую переходную полосу, малые пульсации в полосе пропускания и большое затухание в полосе задерживания — и посмотрим, каким окажется минимальный порядок для фильтров четырех стандартных типов:

```
>> Wp = 0.2; % конец полосы пропускания
>> Ws = 0.21; % начало полосы задерживания
>> Rp = 1;      % пульсации АЧХ в полосе пропускания (дБ)
>> Rs = 60;     % затухание в полосе задерживания (дБ)
>> [n, Wn] = buttord(Wp, Ws, Rp, Rs)
n =
    145
Wn =
    0.2009
>> [n, Wn] = cheb1ord(Wp, Ws, Rp, Rs)
n =
    26
Wn =
    0.2000
>> [n, Wn] = cheb2ord(Wp, Ws, Rp, Rs)
n =
    26
Wn =
    0.2095
```

```
>> [n, Wn] = ellipord(Wp, Ws, Rp, Rs)
n =
10
Wn =
0.2000
```

Результаты показывают, что для фильтра Баттервортса требуется очень большой порядок, порядок фильтров Чебышева обоих типов более чем в пять раз меньше и, наконец, минимальный порядок требуется при синтезе эллиптического фильтра.

## Функция *bilinear*

Функция *bilinear* предназначена для синтеза дискретных фильтров по произвольным аналоговым прототипам методом билинейного *z*-преобразования. Она может преобразовывать заданные различными способами аналоговые описания систем в дискретные и поэтому имеет три варианта синтаксиса:

```
[bz, az] = bilinear(b, a, Fs, Fp)
[zz, pz, kz] = bilinear(z, p, k, Fs, Fp)
[Az, Bz, Cz, Dz] = bilinear(A, B, C, D, Fs, Fp)
```

Здесь *b* и *a* — коэффициенты полиномов числителя и знаменателя функции передачи аналогового прототипа, *z*, *p* и *k* — нули, полюсы и коэффициент усиления аналогового прототипа, *A*, *B*, *C* и *D* — параметры пространства состояний для аналогового прототипа. Аналогичные идентификаторы с буквой *z* в конце обозначают соответствующие параметры синтезированной дискретной системы.

Функция различает способы задания описания аналогового прототипа по размеру переданных параметров. Если два первых входных параметра — векторы-строки, они считаются коэффициентами полиномов числителя и знаменателя функции передачи. Если два первых входных параметра — векторы-столбцы, они считаются нулями и полюсами функции передачи. Если первый входной параметр — матрица, значит, аналоговый прототип задан в пространстве состояний.

### **ВНИМАНИЕ!**

Аналоговый прототип при использовании функции *bilinear* должен быть физически реализуемым, т. е. степень полинома числителя функции передачи не должна превышать степень полинома ее знаменателя.

Параметр *Fs* задает частоту дискретизации в герцах. Последний входной параметр — *Fp* — является необязательным. При его отсутствии билинейное *z*-преобразование выполняется по формуле (6.1), а трансформация частотной оси описывается соотношением (6.2) (см. ранее разд. "Метод билинейного *z*-преобразования" этой главы).

Если при вызове использован параметр *Fp*, то он задает частоту (в герцах), на которой комплексные коэффициенты передачи аналоговой и дискретной систем будут совпадать. Для этого производится предварительное масштабирование частотной оси в *p*-области, в результате чего используемая при билинейном *z*-преобразовании подстановка принимает вид

$$p = \frac{\pi \frac{F_p}{F_s}}{\operatorname{tg} \left( \pi \frac{F_p}{F_s} \right)} 2F_s \frac{1 - z^{-1}}{1 + z^{-1}}.$$

Трансформация частотной оси при этом выглядит следующим образом:

$$f_{\text{а.ф.}} = F_p \frac{\operatorname{tg} \left( \pi \frac{f_{\text{д.ф.}}}{F_s} \right)}{\operatorname{tg} \left( \pi \frac{F_p}{F_s} \right)}, \quad f_{\text{д.ф.}} = \frac{F_s}{\pi} \operatorname{arctg} \left( \frac{f_{\text{а.ф.}}}{F_p} \operatorname{tg} \left( \pi \frac{F_p}{F_s} \right) \right),$$

то есть комплексный коэффициент передачи аналоговой системы на частоте  $f_{\text{а.ф.}}$  будет совпадать с комплексным коэффициентом передачи дискретной системы на частоте  $f_{\text{д.ф.}}$ . Из формул видно, что если  $f_{\text{а.ф.}} = F_p$ , то и  $f_{\text{д.ф.}} = F_p$ . На низких частотах  $f_{\text{а.ф.}}$  и  $f_{\text{д.ф.}}$  связаны приблизительно линейно:

$$f_{\text{а.ф.}} \approx f_{\text{д.ф.}} \frac{\pi \frac{F_p}{F_s}}{\operatorname{tg} \left( \pi \frac{F_p}{F_s} \right)} \quad \text{при } f_{\text{д.ф.}} \ll F_p.$$

## Функция *impinvar*

Функция *impinvar* предназначена для синтеза дискретных фильтров по произвольным аналоговым прототипам методом инвариантной импульсной характеристики. Синтаксис вызова функции следующий:

```
[bz, az] = impinvar(b, a, Fs, tol)
```

Входные параметры *b* и *a* — коэффициенты числителя и знаменателя функции передачи аналогового прототипа, *Fs* — частота дискретизации (по умолчанию ее значение равно 1 Гц).

Параметр *tol* задает относительный порог обнаружения кратных полюсов. Два близко расположенных полюса считаются совпадающими, если расстояние между ними, деленное на больший из их модулей, меньше *tol*.

Выходные параметры *bz* и *az* — коэффициенты числителя и знаменателя функции передачи для синтезированного дискретного фильтра.

В соответствии с теорией, изложенной ранее в разд. "Метод инвариантной импульсной характеристики" этой главы, функция *impinvar* при синтезе фильтра выполняет следующие действия:

1. Функция передачи аналогового прототипа представляется в виде суммы простых дробей.

2. Найденные полюсы  $p$  трансформируются в  $\exp(p/F_s)$ .
3. Функция передачи из суммы простейших дробей преобразуется обратно в дробно-рациональный вид.

## Функции прямого синтеза рекурсивных фильтров

В этом разделе будут рассмотрены функции `yulewalk`, `invfreqz` и `prony`, реализующие алгоритмы прямого синтеза рекурсивных фильтров, перечисленные ранее в разд. "Субоптимальные методы" этой главы.

### Функция `yulewalk`

Функция `yulewalk` предназначена для синтеза рекурсивных фильтров по заданной кусочно-линейной АЧХ. При этом производится минимизация среднеквадратической ошибки во временной области. Синтаксис вызова функции следующий:

```
[b, a] = yulewalk(n, f, m)
```

Здесь  $n$  — порядок рассчитываемого фильтра (возвращаемые векторы  $b$  и  $a$  будут иметь длину  $n + 1$ ).

Параметры  $f$  и  $m$  должны быть векторами одинаковой длины, они совместно определяют желаемую АЧХ синтезируемого фильтра. Вектор  $f$  содержит значения частот, нормированные к частоте Найквиста, а вектор  $m$  — соответствующие этим частотам значения АЧХ. В промежутках между заданными точками АЧХ интерполируется по линейному закону. Частоты в векторе  $f$  должны образовывать неубывающую последовательность, кроме того, должны выполняться равенства  $f(1) = 0$  и  $f(\text{end}) = 1$ . Вывести график синтезируемой АЧХ можно командой `plot(f, m)`.

#### **ВНИМАНИЕ!**

Частоты в векторе  $f$  могут дублироваться, что дает возможность задать скачкообразное изменение АЧХ. Однако задавать такие скачки при синтезе рекурсивных фильтров не рекомендуется.

Результатами работы функции являются векторы  $b$  и  $a$  коэффициентов полиномов числителя и знаменателя функции передачи рекурсивного фильтра. Длина этих векторов равна  $n + 1$ .

Расчет фильтра выполняется следующим образом. Сначала определяется полином знаменателя, при этом числитель функции передачи считается равным единице и задача оптимизации сводится к решению модифицированной системы уравнений Юла — Уолкера (отсчеты корреляционной функции определяются путем обратного ДПФ от квадрата модуля АЧХ). Затем оптимизируется полином числителя, чтобы улучшить приближение к заданным характеристикам.

### Функция `invfreqz`

Функция `invfreqz`, как следует из ее названия, является обратной по отношению к функции `freqz` (см. разд. "Расчет частотной характеристики" главы 4). Это

значит, что она позволяет получить коэффициенты полиномов числителя и знаменателя функции передачи рекурсивного дискретного фильтра по значениям его комплексного коэффициента передачи. Таким образом, функция `invfreqz` может использоваться для синтеза рекурсивных фильтров, причем, в отличие от многих других функций синтеза, здесь в качестве желаемого результата задается не АЧХ, а комплексная частотная характеристика (т. е. АЧХ вместе с ФЧХ). Синтаксис вызова функции следующий:

```
[b, a] = invfreqz(h, w, nb, na, wt, iter, tol, 'trace')
```

Здесь `w` — вектор частот, а `h` — вектор соответствующих им значений комплексного коэффициента передачи. Параметры `w` и `h` должны быть векторами одинаковой длины. Частоты в векторе `w` должны лежать в диапазоне от 0 до  $\pi$ , значение  $\pi$  соответствует частоте Найквиста.

### **ВНИМАНИЕ!**

Нормировка частот, используемая в функции `invfreqz`, отличается от варианта, принятого во всех других функциях обработки сигналов, поэтому стоит повторить еще раз: частота Найквиста в данном случае равна  $\pi$ , а не единице.

Частотная характеристика синтезируемого фильтра сравнивается с заданной только в указанных дискретных частотных точках; никакой интерполяции не производится.

При указанном выше способе вызова частотная характеристика считается симметричной и рассчитывается фильтр с вещественными коэффициентами. Чтобы синтезировать фильтр с комплексными коэффициентами, необходимо после параметра `w` поставить дополнительный параметр в виде строки '`complex`':

```
[b, a] = invfreqz(h, w, 'complex', ...)
```

В этом случае элементы вектора `w` могут лежать в диапазоне от  $-\pi$  до  $\pi$ .

Параметры `nb` и `na` задают степени полиномов числителя и знаменателя функции передачи фильтра.

Остальные параметры функции являются необязательными и имеют значения по умолчанию. Вектор `wt` должен иметь такой же размер, как векторы `h` и `w`. Он задает весовые коэффициенты для расчета ошибки. По умолчанию все частотные точки имеют одинаковый вес.

Оставшиеся три входных параметра имеют отношение лишь к одному из двух возможных алгоритмов расчета, так что само их наличие или отсутствие управляет выбором алгоритма.

Если параметры `iter`, `tol` и '`trace`' отсутствуют, то функция минимизирует ошибку, рассчитываемую следующим образом:

$$e = \sum_{k=1}^N wt(k) |h(k) \dot{A}(w(k)) - \dot{B}(w(k))|^2. \quad (6.18)$$

Здесь  $N$  — количество элементов в векторах  $w$ ,  $h$  и  $wt$ , а  $\dot{A}(w(k))$  и  $\dot{B}(w(k))$  — спектры векторов, составленных из коэффициентов полиномов числителя и знаменателя функции передачи фильтра, вычисленные на  $k$ -й частоте из вектора  $w$ :

$$\dot{A}(w(k)) = \sum_{n=1}^{na+1} a(n)e^{-j(n-1)w(k)}, \quad \dot{B}(w(k)) = \sum_{n=1}^{nb+1} b(n)e^{-j(n-1)w(k)}. \quad (6.19)$$

Подстановка (6.19) в (6.18) и приравнивание к нулю частных производных по  $b(n)$  и  $a(n)$  дает систему линейных уравнений относительно коэффициентов фильтра. Эта система решается соответствующими средствами MATLAB (оператор \).

Разумеется, критерий (6.18) не минимизирует отклонение частотной характеристики фильтра от заданной. Однако он приводит к несложной задаче решения системы линейных уравнений и дает точное решение, если векторы  $h$  и  $w$  действительно описывают фильтр, имеющий  $nb$  нулей и  $na$  полюсов.

Другой алгоритм — это "честный" поиск фильтра, дающего минимальную взвешенную квадратическую ошибку воспроизведения заданной частотной характеристики. Минимизируемая ошибка рассчитывается при этом следующим образом:

$$e = \sum_{k=1}^N wt(k) \left| \dot{h}(k) - \frac{\dot{B}(w(k))}{\dot{A}(w(k))} \right|^2. \quad (6.20)$$

В данном случае дифференцирование критерия  $e$  по коэффициентам фильтра дает систему нелинейных уравнений, поэтому найти аналитическое решение не удается и оптимизация производится численным итерационным методом.

Признаком расчета по второму алгоритму является наличие входного параметра  $iter$ , задающего число итераций. Необходимое количество итераций зависит от параметров синтезируемого фильтра. Можно начать со значения 20, увеличивая его, если заданная точность не будет достигнута.

Параметр  $tol$  задает критерий остановки итерационного алгоритма. Если величина нормы градиента целевой функции (6.20) становится меньше  $tol$ , решение считается найденным. По умолчанию значение  $tol$  равно 0,01.

Наконец, при указании параметра 'trace' функция выводит информацию о ходе итерационной оптимизации.

Результатом работы функции являются векторы  $b$  и  $a$  коэффициентов полиномов числителя и знаменателя функции передачи синтезированного рекурсивного фильтра. Размеры этих векторов составляют  $nb + 1$  и  $na + 1$  соответственно.

## Функция *prony*

Функция *prony* позволяет синтезировать рекурсивный фильтр по заданной импульсной характеристике. Синтаксис вызова функции следующий:

```
[b, a] = prony(h, nb, na)
```

Здесь  $h$  — вектор отсчетов синтезируемой импульсной характеристики,  $n_b$  и  $n_a$  — порядки полиномов числителя и знаменателя функции передачи рекурсивного фильтра соответственно.

Возвращаемые результаты — векторы  $b$  и  $a$  коэффициентов полиномов числителя и знаменателя функции передачи синтезированного рекурсивного фильтра. Длина вектора  $b$  равна  $n_b+1$ , длина вектора  $a$  —  $n_a+1$ .

Расчет фильтра производится следующим образом. Сначала определяется знаменатель функции передачи с помощью ковариационного метода анализа авторегрессионных моделей. Затем находится такой полином числителя, чтобы первые  $n_b+1$  отсчетов импульсной характеристики получившегося фильтра в точности совпадали с соответствующим фрагментом вектора  $h$ . Устойчивость рассчитанного фильтра не гарантируется, однако данный метод может дать точные результаты, если отсчеты в векторе  $h$  действительно представляют собой начало импульсной характеристики рекурсивного фильтра с числителем и знаменателем порядков  $n_b$  и  $n_a$ .

В качестве примера попытаемся синтезировать рекурсивный фильтр 5-го порядка с *конечной* импульсной характеристикой треугольного вида. Чтобы уменьшить "хвост" после окончания заданного фрагмента, дополним задаваемую импульсную характеристику некоторым количеством нулей. Результат синтеза (импульсная характеристика полученного фильтра) показан на рис. 6.11:

```
>> h = [1 2 3 4 5 4 3 2 1 zeros(1, 9)];
>> [b, a] = prony(h, 5, 5);
>> impz(b, a)
```

Из графика видно, что результаты получены весьма неплохие — затухающий "хвост" оказался небольшим.

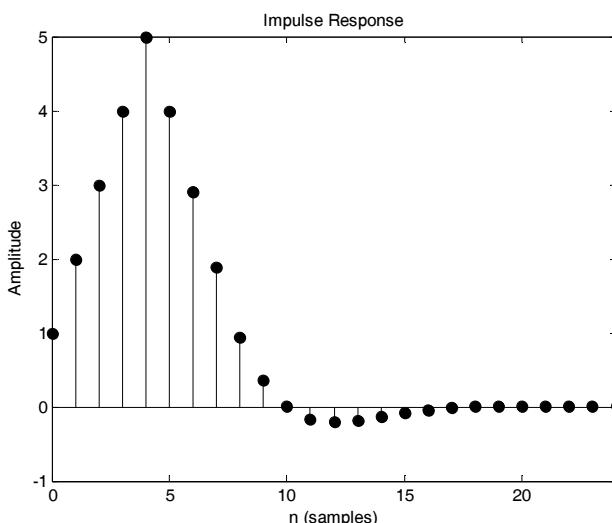


Рис. 6.11. Результаты аппроксимации треугольной импульсной характеристики в рекурсивном виде с помощью функции `prony`

### ЗАМЕЧАНИЕ

Если при вызове функции `prony` указать порядок числителя, на единицу меньший числа ненулевых отсчетов заданной импульсной характеристики (в нашем примере — 8) или превышающий это значение, будет синтезирован *нерекурсивный* фильтр, *точно* воспроизводящий заданную импульсную характеристику.

### Функция `stmcb`

Функция `stmcb`, аналогично функции `prony`, позволяет синтезировать рекурсивный фильтр по заданной импульсной характеристике. Однако она не стремится обеспечить точное совпадение начальных фрагментов импульсных характеристик — вместо этого функция минимизирует *квадратичное отклонение* полученной характеристики от заданной, т. е. сумму квадратов модулей разностей отсчетов полученной и желаемой импульсных характеристик.

Синтаксис вызова функции `stmcb` следующий:

```
[b, a] = stmcb(h, nb, na, niter, ai)
```

Здесь  $h$  — вектор отсчетов синтезируемой импульсной характеристики,  $nb$  и  $na$  — порядки полиномов числителя и знаменателя функции передачи рекурсивного фильтра соответственно. Остальные входные параметры являются необязательными и имеют значения по умолчанию:

- $niter$  — число итераций, выполняемых в процессе численного решения оптимизационной задачи (по умолчанию 5);
- $ai$  — начальное приближение для вектора коэффициентов знаменателя функции передачи синтезируемого фильтра (по умолчанию он находится с помощью функции `prony` следующим образом: `[tmp, ai] = prony(h, 0, na)`).

Возвращаемые результаты — векторы  $b$  и  $a$  коэффициентов полиномов числителя и знаменателя функции передачи синтезированного рекурсивного фильтра. Длина вектора  $b$  равна  $nb+1$ , длина вектора  $a$  —  $na+1$ .

Функция, как уже было сказано, минимизирует сумму квадратов модулей разностей между заданной импульсной характеристикой  $\{h(k)\}$  и импульсной характеристикой  $\{\hat{h}(k)\}$  рекурсивного фильтра с заданными порядками полиномов числителя и знаменателя функции передачи:

$$\sum_{k=0}^N |h(k) - \hat{h}(k)|^2 \rightarrow \min, \quad (6.21)$$

где  $N$  — порядок заданной импульсной характеристики. В  $z$ -области стоящее под знаком модуля выражение имеет вид

$$H(z) - \hat{H}(z) = H(z) - \frac{B(z)}{A(z)}, \quad (6.22)$$

где  $B(z)$  и  $A(z)$  — полиномы соответственно числителя и знаменателя функции передачи синтезируемого фильтра. Поиск коэффициентов этих полиномов произво-

дится с помощью итерационного метода Штейглица — МакБрайда (K. Steiglitz и L. E. McBride), суть которого состоит в том, что (6.22) записывается в форме

$$H(z) - \hat{H}(z) = \frac{H(z)}{A(z)} A(z) - \frac{1}{A(z)} B(z)$$

и в знаменателях дробей используется оценка  $A(z)$ , полученная на предыдущей итерации. Получившееся выражение соответствует нерекурсивному фильтру, так что для нахождения коэффициентов полиномов  $B(z)$  и  $A(z)$ , минимизирующих (6.21), необходимо решить систему линейных уравнений (в MATLAB это выполняется с помощью оператора \).

В процессе решения всегда выполняется заданное число итераций; какой-либо проверки сходимости процесса не производится.

Как видно из (6.21), при суммировании учитывается конечное число слагаемых, равное длине целевой импульсной характеристики. Поэтому при аппроксимации характеристики конечной длины ее следует дополнить некоторым количеством нулей — это уменьшит амплитуду затухающего "хвоста".

В качестве примера попытаемся синтезировать данным методом рекурсивный фильтр 5-го порядка с такой же треугольной импульсной характеристикой, что использовалась при демонстрации метода Прони. Результат синтеза (импульсная характеристика полученного фильтра) показан на рис. 6.12:

```
>> h = [1 2 3 4 5 4 3 2 1 zeros(1, 9)];
>> [b, a] = stmcbs(h, 5, 5);
>> impz(b, a)
```

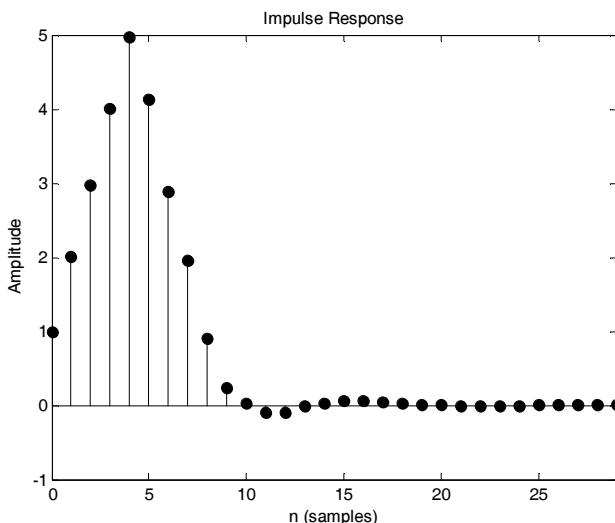


Рис. 6.12. Результаты аппроксимации треугольной импульсной характеристики в рекурсивном виде с помощью функции stmcbs

Сравнение графиков рис. 6.11 и 6.12 показывает, что функция `stmcb` дала несколько большую (по сравнению с функцией `prony`) ошибку на ненулевом участке целевой импульсной характеристики (асимметрия треугольника оказалась чуть сильнее), но зато обеспечила меньший выброс после завершения треугольного фрагмента.

### **ЗАМЕЧАНИЕ**

Функция `stmcb` может также найти коэффициенты полиномов функции передачи рекурсивной системы по ее входному и выходному сигналам. В этом случае вместо вектора `h` в списке параметров указываются два вектора одинаковой длины, задающие эти сигналы (`x` — входной сигнал, `y` — выходной):

```
[b, a] = stmcb(x, y, nb, na, niter, ai)
```

## **ФУНКЦИИ СИНТЕЗА С ИСПОЛЬЗОВАНИЕМ ОКОН**

Общая идея синтеза нерекурсивных фильтров с использованием окон была рассмотрена ранее, в разд. "Субоптимальный синтез нерекурсивных фильтров" этой главы. В пакете Signal Processing имеются две функции, реализующие данный метод. Различия между ними состоят в типе аппроксимируемой АЧХ — функция `fir1` предназначена для синтеза многополосных фильтров, АЧХ которых в заданных частотных точках скачкообразно меняется, принимая значения 0 или 1, а функция `fir2` допускает задание произвольной кусочно-линейной АЧХ. Кроме того, в том же пакете Signal Processing имеется функция `kaiserord`, предназначенная для выбора порядка фильтра, который при синтезе с применением окна Кайзера будет удовлетворять заданным требованиям.

### **Функция `fir1`**

Функция `fir1` позволяет рассчитывать нерекурсивные полосно-пропускающие фильтры с линейной ФЧХ, в том числе многополосные (*multiband filter*), методом обратного преобразования Фурье с использованием окон. Синтаксис вызова функции следующий:

```
b = fir1(n, Wn, 'ftype', window, 'normalization')
```

Здесь `n` — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать  $n + 1$  ненулевых отсчетов).

Параметры `Wn` и `'ftype'` (который может отсутствовать) совместно определяют тип синтезируемого фильтра и его частоту (частоты) среза. Частоты среза задаются нормированными к частоте Найквиста (т. е. их значения должны лежать в диапазоне 0...1, величина 1 соответствует половине частоты дискретизации). Значения этих параметров зависят от типа фильтра:

- ФНЧ: `Wn` — частота среза, параметр `'ftype'` отсутствует;
- ФВЧ: `Wn` — частота среза, `'ftype' = 'high'`;
- полосовой фильтр: `Wn = [w1 w2]`, где `w1` и `w2` — границы полосы пропускания, параметр `'ftype'` отсутствует;

- режекторный фильтр:  $Wn = [w1 \ w2]$ , где  $w1$  и  $w2$  — границы полосы задерживания, 'ftype' = 'stop';
- многополосный фильтр, пропускающий низкие частоты:  $Wn = [w1 \ w2 \ \dots \ wn]$ , где  $wi$  — границы полос пропускания, упорядоченные по возрастанию, 'ftype' = 'DC-1'. Такой фильтр будет иметь полосы пропускания  $0 \dots w1, w2 \dots w3$  и т. д.;
- многополосный фильтр, задерживающий низкие частоты:  $Wn = [w1 \ w2 \ \dots \ wn]$ , где  $wi$  — границы полос пропускания, упорядоченные по возрастанию, 'ftype' = 'DC-0'. Такой фильтр будет иметь полосы пропускания  $w1 \dots w2, w3 \dots w4$  и т. д.

### **ЗАМЕЧАНИЕ**

Обозначения 'DC-0' и 'DC-1' расшифровываются следующим образом: DC обозначает постоянный ток (direct current), т. е. нулевую частоту, а 0 или 1 — коэффициент передачи синтезируемого фильтра на нулевой частоте.

Для ФВЧ и режекторных фильтров, а также многополосных фильтров с ненулевым коэффициентом передачи на частоте Найквиста возможен только четный порядок. Если задать нечетное  $n$ , будет рассчитан фильтр на единицу большего порядка.

Параметр `window` задает используемое при синтезе окно. Это должен быть вектор-столбец, содержащий  $n + 1$  элементов. По умолчанию используется окно Хэмминга, рассчитываемое путем вызова функции `hamming(n+1)`.

Строковый параметр 'normalization' управляет нормировкой (масштабированием) рассчитанной импульсной характеристики фильтра. По умолчанию используется значение 'scale', при котором импульсная характеристика нормируется так, чтобы обеспечить единичное значение (0 дБ) коэффициента передачи в центре полосы пропускания (для многополосных фильтров — в центре самой низкочастотной полосы пропускания). Второе возможное значение параметра — 'noscale', в этом случае нормировка не производится.

Результатом работы функции является вектор  $b$  коэффициентов нерекурсивного фильтра. Рассчитанный фильтр имеет линейную ФЧХ и вносит групповую задержку, независимо от частоты равную  $n/2$  отсчетам. Расчет фильтра выполняется следующим образом: обратное преобразование Фурье от заданной идеальной АЧХ дает бесконечный набор отсчетов импульсной характеристики. Этот набор симметрично усекается до  $n + 1$  отсчетов и для ослабления эффектов усечения умножается на используемое окно. Если задана нормировка, определяется модуль коэффициента передачи получившегося фильтра в центре полосы пропускания, и коэффициенты фильтра делятся на это значение.

### **Функция `fir2`**

Функция `fir2`, так же как и `fir1`, предназначена для синтеза нерекурсивных фильтров с линейной ФЧХ методом обратного преобразования Фурье с использованием окон. Отличие заключается в том, что при синтезе можно задать произвольную кусочно-линейную АЧХ.

Синтаксис вызова функции следующий:

```
b = fir2(n, f, m, Npt, lap, window)
```

Здесь  $n$  — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать  $n + 1$  ненулевых отсчетов).

Параметры  $f$  и  $m$  должны быть векторами одинаковой длины, они совместно определяют желаемую АЧХ синтезируемого фильтра. Вектор  $f$  содержит значения частот, нормированные к частоте Найквиста, а вектор  $m$  — соответствующие этим частотам значения АЧХ. В промежутках между заданными точками АЧХ интерполируется по линейному закону. Частоты в векторе  $f$  должны образовывать неубывающую последовательность, кроме того, должны выполняться равенства  $f(1) = 0$  и  $f(\text{end}) = 1$ . Частоты могут дублироваться, что дает возможность задать скачкообразное изменение АЧХ. Вывести график синтезируемой АЧХ можно командой `plot(f, m)`.

Для фильтров, коэффициент передачи которых на частоте Найквиста не равен нулю, возможен только четный порядок. Если задать нечетное  $n$ , будет рассчитан фильтр на единицу большего порядка.

Остальные параметры функции являются необязательными. Параметр  $Npt$  задает число точек, используемое при интерполяции заданной АЧХ. По умолчанию его значение равно 512.

Параметр  $lap$  задает ширину переходных зон вокруг скачков. При интерполяции АЧХ в этих зонах вместо скачка будет сформирован линейный переход. Ширина зон измеряется в точках интерполяции и по умолчанию равна 25.

Параметр  $window$  задает используемое при синтезе окно. Это должен быть вектор-столбец, содержащий  $n + 1$  элементов. По умолчанию используется окно Хэмминга, рассчитываемое путем вызова функции `hamming(n+1)`.

Результатом работы функции является вектор  $b$  коэффициентов нерекурсивного фильтра. Рассчитанный фильтр имеет линейную ФЧХ и вносит групповую задержку, независимо от частоты равную  $n/2$  отсчетам. Расчет фильтра выполняется следующим образом: заданная АЧХ кусочно-линейно интерполируется на равномерную сетку из  $Npt$  точек, обратное ДПФ дает импульсную характеристику, она симметрично усекается до  $n + 1$  отсчетов и для ослабления эффектов усечения умножается на используемое окно.

### **ЗАМЕЧАНИЕ**

Число точек интерполяции должно быть не меньше длины синтезируемого фильтра. Поэтому при расчете фильтров порядка 512 и более значение параметра  $Npt$ , принятого по умолчанию, оказывается недостаточным и его необходимо задать явно.

## **Функция *kaiserord***

Функция `kaiserord` предназначена для определения порядка фильтра, удовлетворяющего заданным требованиям, при синтезе функцией `firl` с использованием окна Кайзера.

Функция имеет два варианта синтаксиса:

```
[n, wn, beta, ftype] = kaiserord(f, a, dev, Fs)
c = kaiserord(f, a, dev, Fs, 'cell')
```

Во втором варианте синтаксиса возвращаемый массив ячеек *c* содержит полный набор параметров для вызова функции *fir1*.

Параметры *f* и *a* совместно задают кусочно-постоянную АЧХ. В векторе *f* должна содержаться возрастающая последовательность значений частот. Вектор *a* задает постоянные значения АЧХ для полос, ограниченных парами частот из вектора *f*, при этом первая полоса начинается от нулевой частоты, а последняя заканчивается на частоте Найквиста:

- в полосе от нуля до  $f(1)$  АЧХ равна  $a(1)$ ;
- в полосе от  $f(1)$  до  $f(2)$  АЧХ не определена;
- в полосе от  $f(2)$  до  $f(3)$  АЧХ равна  $a(2)$ ;
- в полосе от  $f(3)$  до  $f(4)$  АЧХ не определена;
- так продолжается до конца векторов *f* и *a*;
- в полосе от  $f(end-1)$  до  $f(end)$  АЧХ не определена;
- в полосе от  $f(end)$  до  $Fs/2$  АЧХ равна  $a(end)$ .

Таким образом, длина вектора *f* должна быть равна  $\text{length}(a) * 2 - 2$ .

Поскольку функция формирует исходные данные для функции *fir1*, в векторе *a* могут содержаться только значения 0 и 1.

Параметр *dev* должен быть вектором той же длины, что и *a*, содержащим вещественные положительные числа. Он задает максимально допустимую ошибку воспроизведения кусочно-постоянной АЧХ для отдельных частотных полос. Ошибка задается как *абсолютная* погрешность АЧХ.

Параметр *Fs* задает частоту дискретизации и является необязательным. При его использовании следует указывать в векторе *f* не нормированные, а исходные значения частот в тех же единицах, что и *Fs*. По умолчанию частота дискретизации считается равной двум, так что частота Найквиста равна единице.

Результатами работы функции являются порядок фильтра *n*, вектор нормированных частот среза *wn*, параметр окна Кайзера *beta* и указатель типа фильтра *ftype*. Эти параметры затем должны использоваться при вызове функции *fir1* следующим образом:

```
b = fir1(n, wn, kaiser(n+1, beta), ftype, 'noscale')
```

Если использован второй вариант вызова функции *kaiserord*, вызывать функцию *fir1* следует так:

```
b = fir1(c{1}, c{2}, c{3}, c{4}, c{5})
```

Функция *kaiserord* использует эмпирические формулы для расчета параметров. Так, параметр окна Кайзера *beta* определяется по формуле (6.14) (см. ранее

разд. "Синтез с использованием окон" этой главы). Порядок фильтра  $n$  оценивается по следующей формуле [16]:

$$n = \frac{\alpha - 7,95}{7,1785 \Delta f}. \quad (6.23)$$

Здесь  $\alpha$  — степень подавления боковых лепестков АЧХ в децибелах, а  $\Delta f$  — нормированная к частоте Найквиста ширина самой узкой переходной полосы с неопределенной АЧХ.

Функция основана на простых прикидочных формулах и поэтому может давать неточные результаты, особенно если в заданной АЧХ имеются частоты среза, близкие к нулю или частоте Найквиста, или если задано слишком слабое подавление боковых лепестков (в векторе `dev` есть элементы, превышающие 0,1). Поэтому после синтеза фильтра функцией `firl` следует проверить его АЧХ с помощью функции `freqz` и при необходимости попробовать точнее подобрать порядок фильтра вручную.

## Функции расчета ФНЧ с косинусоидальным сглаживанием

Функции для расчета ФНЧ с косинусоидальным сглаживанием АЧХ имеются в двух пакетах расширения MATLAB — Signal Processing (функция `firrcos`) и Communications (функция `rcosine` и две вызываемые ею функции более низкого уровня `rcosfir` и `rcosiir`). Сравнивая их, следует отметить, что функция `firrcos` из пакета Signal Processing имеет более общий характер, тогда как функции пакета Communications приспособлены для решения конкретной задачи — интерполяции сигнала с целью формирования спектра при квадратурной манипуляции. Этим обусловлен ряд ограничений на параметры фильтра (на каждый лепесток импульсной характеристики должно приходиться целое число интервалов дискретизации, на длине нерекурсивного фильтра должно укладываться целое число лепестков и т. п.). В функции `firrcos` ограничения такого рода отсутствуют. Зато пакет Communications дает возможность аппроксимировать данный фильтр в рекурсивном виде (функция `rcosiir`).

Еще одно различие между функцией `firrcos` и функциями `rcosfir` и `rcosiir` заключается в используемой нормировке. Функция `firrcos` рассчитывает фильтр, коэффициент передачи которого в полосе пропускания равен единице, а функции пакета Communications делают равной единице *высоту пика импульсной характеристики*.

Подводя итог, скажем, что выбор функции для расчета фильтра в конкретной ситуации следует производить из соображений удобства задания параметров.

А теперь перейдем к описанию конкретных функций.

### Функция `firrcos`

Функция `firrcos` (пакет Signal Processing) предназначена для расчета нерекурсивного ФНЧ с косинусоидальным сглаживанием АЧХ.

Синтаксис ее вызова следующий:

```
b = firrcos(n, F0, df, Fs, 'transition', 'type', delay, window)
```

Здесь  $n$  — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать  $n + 1$  ненулевых отсчетов),  $F_0$  — частота среза,  $F_s$  — частота дискретизации ( $F_0$  и  $F_s$  должны быть заданы в одних и тех же единицах).

Параметр  $df$  задает ширину переходной зоны между полосами пропускания и задерживания. Это может делаться двумя способами в зависимости от значения параметра 'transition':

- при 'transition' = 'bandwidth' параметр  $df$  задает ширину переходной зоны в единицах частоты (при этом должны выполняться неравенства  $df \leq 2*F_0$  и  $F_0+df/2 \leq F_s/2$ ). Этот вариант принят по умолчанию;
- при 'transition' = 'rolloff' параметр  $df$  задает коэффициент сглаживания АЧХ, обозначенный в формулах (6.15) и (6.16) как  $\alpha$ . Значение параметра  $df$  в этом случае должно лежать в диапазоне 0...1.

Входные параметры, начиная с  $F_s$ , являются необязательными и имеют значения по умолчанию, которые используются, если при вызове в качестве параметра указана пустая матрица [] или если несколько последних параметров опущено. Частота дискретизации  $F_s$  по умолчанию считается равной двум, так что частота Найквиста равна единице.

Строковый параметр 'type' задает тип синтезируемого фильтра. При принятом по умолчанию варианте 'normal' рассчитывается обычный ФНЧ с косинусоидальным сглаживанием по формуле (6.16). Если 'type' = 'sqrt', рассчитывается SQRT-вариант данного фильтра по формуле (6.17).

Параметр  $delay$  задает вносимую фильтром задержку (т. е. место расположения пика импульсной характеристики). Задержка измеряется в отсчетах и должна быть целым числом в диапазоне от нуля до  $n + 1$ . По умолчанию принято значение  $floor((n + 1) / 2)$ .

Параметр  $window$  позволяет использовать при синтезе фильтра дополнительную весовую функцию (см. ранее разд. "Синтез с использованием окон" этой главы). Это должен быть вектор-столбец длиной  $n + 1$ . По умолчанию весовая функция не используется (т. е. применяется прямоугольное окно).

Результатом работы функции является вектор  $b$  коэффициентов нерекурсивного фильтра.

## Функция *rcosine*

Функция *rcosine* пакета Communications позволяет рассчитывать ФНЧ с косинусоидальным сглаживанием АЧХ в нерекурсивном и рекурсивном виде. Синтаксис вызова функции следующий:

```
[b, a] = rcosine(Fd, Fs, 'type', r, delay, tol)
```

Обязательными параметрами здесь являются только  $F_s$  и  $F_d$ . Использованная в этой функции система задания параметров отражает основное назначение данных

фильтров, которые в системах связи широко используются для интерполяции сигнала при цифровой модуляции. Параметр  $F_d$  — это символьная скорость (т. е. частота дискретизации до интерполяции), а  $F_s$  — частота дискретизации выходного сигнала. Отношение  $F_s/F_d$  должно быть положительным целым числом. Оно равно числу отсчетов, приходящихся на один лепесток импульсной характеристики фильтра.

Остальные входные параметры являются необязательными и имеют значения по умолчанию, которые используются, если при вызове в качестве параметра указана пустая матрица [] или если несколько последних параметров опущено.

Строковый параметр 'type' задает тип рассчитываемого фильтра. Возможны следующие варианты:

- 'default' или 'fir/normal' — рассчитывается нерекурсивный ФНЧ с косинусоидальным сглаживанием. Этот вариант принят по умолчанию;
- 'iir' или 'iir/normal' — рассчитывается рекурсивный ФНЧ, аппроксимирующий АЧХ с косинусоидальным сглаживанием;
- 'sqrt' или 'fir/sqrt' — рассчитывается SQRT-вариант нерекурсивного ФНЧ с косинусоидальным сглаживанием;
- 'iir/sqrt' — рассчитывается SQRT-вариант рекурсивного ФНЧ с косинусоидальным сглаживанием.

Параметр  $r$  задает коэффициент сглаживания АЧХ. По умолчанию используется значение 0,5.

Параметр  $delay$  задает вносимую фильтром задержку, т. е. положение пика импульсной характеристики. Величина задержки измеряется в символьных тактах (т. е. в лепестках импульсной характеристики) и должна быть целым положительным числом. По умолчанию задержка равна трем тактам (в секундах —  $3/F_d$ ).

Порядок нерекурсивного фильтра определяется требованием линейности ФЧХ и, следовательно, симметрии импульсной характеристики. Таким образом, пик импульсной характеристики должен находиться в ее середине, и длительность импульсной характеристики составляет  $2*delay$  символьных тактов. Порядок фильтра получается равным  $2*delay*Fs/F_d$ .

В случае рекурсивного фильтра его порядок можно задать явно с помощью параметра  $tol$ , указав для него целое положительное значение. Если  $tol$  меньше единицы, этот параметр задает относительную погрешность синтеза рекурсивного фильтра, а порядок фильтра при этом выбирается автоматически. По умолчанию используется значение 0,01. При расчете нерекурсивных фильтров параметр  $tol$  игнорируется.

Возвращаемыми результатами являются векторы коэффициентов полиномов числителя  $b$  и знаменателя  $a$  функции передачи рассчитанного фильтра. Если рассчитывался нерекурсивный фильтр, знаменатель  $a$  содержит один элемент, равный единице.

Расчет фильтра производится с помощью двух функций более низкого уровня — `rcosfir` (нерекурсивный фильтр) и `rcosiir` (рекурсивный фильтр).

## Функция расчета рекурсивного фильтра Гильберта

Расчет рекурсивного фильтра, аппроксимирующего преобразование Гильберта, можно выполнить с помощью входящей в состав пакета Communications функции `hilbiir`, имеющей следующий синтаксис вызова:

```
[num, den] = hilbiir(ts, dly, bandwidth, tol);
```

Все входные параметры функции являются необязательными и имеют значения по умолчанию. Параметр `ts` задает интервал дискретизации входного сигнала фильтра. По умолчанию используется значение  $2/7$  с.

Параметр `dly` задает вносимую фильтром групповую задержку. Ее значение должно как минимум в несколько раз превышать величину периода дискретизации `ts`. Рекомендуется также выбирать соотношение между этими двумя параметрами так, чтобы значение `dly` было равно *полуцелому* числу интервалов `ts` (т. е. `rem(dly, ts) = ts/2`). Например, можно взять `ts` равным  $2*dly/N$ , где `N` — положительное нечетное целое число. По умолчанию групповая задержка равна  $ts*7/2$ .

Параметр `bandwidth` задает предполагаемую полосу частот входного сигнала, позволяя функции использовать дополнительный фильтр-компенсатор, чтобы в полосе частот сигнала приблизить коэффициент передачи синтезированного фильтра к идеальному. Если значение `bandwidth` равно нулю или превосходит  $1/(2*ts)$ , то компенсатор не используется.

Если параметр `tol` превышает единицу, он задает порядок рассчитываемого фильтра (по умолчанию — 4). Если  $tol < 1$ , значение этого параметра задает допуск, используемый при синтезе фильтра методом разложения по сингулярным числам. По умолчанию этот допуск равен 0,05.

Результатами работы являются векторы коэффициентов числителя `num` и знаменателя `den` функции передачи рекурсивного фильтра. Функцию можно вызвать с четырьмя выходными параметрами, тогда будут возвращены параметры фильтра в пространстве состояний:

```
[A, B, C, D] = hilbiir(...);
```

При отсутствии выходных параметров функция строит графики импульсных характеристик синтезированного и идеального фильтров Гильберта.

Расчет фильтра производится следующим образом. Сначала рассчитывается импульсная характеристика идеального фильтра Гильберта с заданной групповой задержкой. Затем выполняется аппроксимация этой импульсной характеристики методом разложения соответствующей матрицы Ганкеля по сингулярным числам.

## Функции минимизации среднеквадратической ошибки

В пакете Signal Processing имеется три функции, реализующие синтез нерекурсивных фильтров по критерию минимального квадратического отклонения АЧХ от заданной. Функция `firls` аппроксимирует произвольную кусочно-линейную АЧХ с переходными (незданными) полосами. Остальные две функции выполняют оп-

тимизацию с ограничением максимального абсолютного отклонения АЧХ от заданной. Функция `fircls` реализует произвольную кусочно-постоянную АЧХ, а функция `fircls1` предназначена для синтеза ФНЧ и ФВЧ.

## Функция `firls`

Функция `firls` предназначена для синтеза нерекурсивных фильтров с линейной ФЧХ по критерию минимальной среднеквадратической ошибки воспроизведения заданной АЧХ в заданном наборе частотных полос. При этом ошибка в разных частотных полосах может учитываться с разными весовыми коэффициентами. Синтаксис вызова функции следующий:

```
b = firls(n, f, a, w, 'ftype')
```

Здесь  $n$  — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать  $n + 1$  ненулевых отсчетов).

Параметры  $f$  и  $a$  должны быть векторами одинаковой четной длины, они совместно определяют желаемую АЧХ синтезируемого фильтра. Вектор  $f$  содержит значения частот, нормированные к частоте Найквиста, а вектор  $a$  — соответствующие этим частотам значения АЧХ. В промежутках между заданными точками АЧХ интерполируется по линейному закону. Частоты в векторе  $f$  должны образовывать неубывающую последовательность, однако, в отличие от функций `fir2` и `yulewalk`, ограничений на первый и последний элементы вектора  $f$  не накладывается.

Синтезируемая АЧХ определяется с помощью векторов  $f$  и  $a$  следующим образом:

- в полосах частот от  $f(2k - 1)$  до  $f(2k)$  АЧХ линейно меняется от  $a(2k - 1)$  до  $a(2k)$ . Здесь  $k$  — целые числа от единицы до половины длины векторов  $f$  и  $a$ ;
- в полосах частот от  $f(2k)$  до  $f(2k + 1)$  АЧХ считается не заданной и в процессе синтеза может оказаться произвольной.

Вывести график синтезируемой АЧХ можно следующей командой:

```
>> plot(reshape(f, 2, length(f)/2), reshape(a, 2, length(a)/2), 'b')
```

### ЗАМЕЧАНИЕ

Если не задать цвет линий принудительно (в данном случае он сделан синим с помощью строки '`b`'), отдельные линейные фрагменты будут выводиться разными цветами.

Для фильтров, коэффициент передачи которых на частоте Найквиста не равен нулю, возможен только четный порядок. Если задать нечетное  $n$ , будет рассчитан фильтр на единицу большего порядка.

Остальные параметры функции являются необязательными. Параметр  $w$  задает весовые коэффициенты для оптимизируемых частотных полос. Этот параметр должен быть вектором, длина которого вдвое меньше, чем длина векторов  $f$  и  $a$ , и содержать неотрицательные вещественные числа. В тех частотных полосах, которым приписаны большие весовые коэффициенты, будет обеспечена меньшая ошибка

воспроизведения заданной АЧХ. По умолчанию все весовые коэффициенты считаются одинаковыми.

При отсутствии параметра 'ftype' производится расчет фильтров с симметричной импульсной характеристикой, для которой  $b(k) = b(n + 2 - k)$ . Если использовать параметр 'ftype', становится возможным синтезировать фильтры с антисимметричной импульсной характеристикой, когда  $b(k) = -b(n + 2 - k)$ . Данный параметр может принимать одно из двух строковых значений:

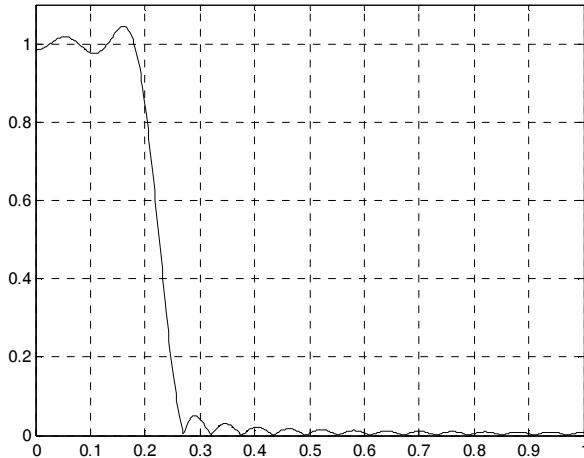
- 'hilbert' — синтезируется фильтр с антисимметричной импульсной характеристикой. Название данного режима объясняется тем, что такую характеристику, в частности, должен иметь фильтр, реализующий преобразование Гильберта (см. разд. "Дискретное преобразование Гильберта" главы 4);
- 'differentiator' — синтезируется фильтр с антисимметричной импульсной характеристикой и, кроме того, при расчете взвешенной ошибки дополнительно используется весовой множитель  $1/f^2$ . В результате ошибка воспроизведения АЧХ на низких частотах будет намного меньше, чем на высоких. При синтезе дифференцирующего фильтра, АЧХ которого пропорциональна частоте (см. разд. "Свойства преобразования Фурье" главы 1), это означает минимизацию относительной ошибки воспроизведения заданной АЧХ.

Результатом работы функции является вектор  $\mathbf{b}$  коэффициентов нерекурсивного фильтра. Рассчитанный фильтр имеет линейную ФЧХ и вносит групповую задержку, независимо от частоты равную  $n/2$  отсчетам.

Расчет фильтра производится следующим образом. Минимизация взвешенной среднеквадратической ошибки воспроизведения заданной АЧХ приводит к системе линейных уравнений относительно коэффициентов фильтра. Порядок системы с учетом симметрии коэффициентов составляет примерно  $n/2$ . Составленная система решается с помощью оператора левого матричного деления \. В процессе решения возможна выдача предупреждения "Matrix is close to singular or badly scaled" (матрица близка к сингулярной или плохо масштабирована). Это, как правило, означает неудачное сочетание порядка фильтра и заданной АЧХ. Возвращаемые результаты могут быть ненадежными, поэтому необходимо проверить их, рассчитав частотную характеристику с помощью функции freqz (впрочем, это полезно сделать в любом случае).

В качестве примера синтезируем данным методом ФНЧ 32-го порядка с нормированной частотой среза, равной 0,2, и началом полосы задерживания на нормированной частоте 0,25. График АЧХ полученного фильтра приведен на рис. 6.13:

```
>> f = [0 0.2 0.25 1];
>> a = [1 1 0 0];
>> b= fir1(32, f, a);
>> [h, w] = freqz(b);
>> plot(w/pi, abs(h)) % график АЧХ
>> grid on
>> ylim([0 1.1])
```



**Рис. 6.13.** АЧХ ФНЧ, синтезированного путем минимизации среднеквадратической ошибки с помощью функции *firls*

На рисунке хорошо видны проявления эффекта Гиббса, неизбежно возникающего при минимизации среднеквадратической ошибки — амплитуда максимальных выбросов АЧХ составляет примерно 0,05. Далее мы синтезируем фильтр с такими же исходными данными еще несколькими методами, чтобы оценить разницу результатов синтеза.

### Функция *fircls*

Функция *fircls* предназначена для синтеза нерекурсивных фильтров с линейной ФЧХ путем минимизации среднеквадратической ошибки воспроизведения заданной кусочно-постоянной АЧХ с ограничением предельно допустимых отклонений от нее (*constrained least square*). Синтаксис вызова функции следующий:

```
b = fircls(n, f, amp, up, 'design_flag')
```

Здесь *n* — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать *n* + 1 ненулевых отсчетов).

Векторы *f* и *amp* совместно определяют желаемую АЧХ синтезируемого фильтра. Вектор *f* содержит значения частот, нормированные к частоте Найквиста. Эти частоты должны образовывать возрастающую последовательность, кроме того, должны выполняться равенства *f*(1) = 0 и *f*(end) = 1. Длина вектора *amp* должна быть на единицу меньше, чем длина вектора *f*. Элементы вектора *amp* задают кусочно-постоянную АЧХ (в диапазоне частот от *f*(*k*) до *f*(*k* + 1) постоянное значение АЧХ равно *amp*(*k*)). Вывести график синтезируемой АЧХ можно следующей командой:

```
>> stairs(f, [amp(:);amp(end)])
```

Для фильтров, коэффициент передачи которых на частоте Найквиста не равен нулю, возможен только четный порядок. Если задать нечетное *n*, будет рассчитан фильтр на единицу большего порядка.

Векторы `up` и `lo` задают верхнюю и нижнюю допустимые границы для значений АЧХ синтезируемого фильтра. Эти векторы должны иметь такой же размер, как и вектор `amp`. Используемый критерий оптимизации можно сформулировать следующим образом: ищутся коэффициенты фильтра заданного порядка, которые дают АЧХ, лежащую между `lo` и `up` и имеющую при этом минимальное среднеквадратическое отклонение от `amp`.

Строковый параметр '`design_flag`' является необязательным, он позволяет управлять выводом информации о ходе поиска решения (по умолчанию эта информация не выводится). Возможны три значения данного параметра:

- '`trace`' — текстовый вывод величины ошибки для каждой итерации;
- '`plots`' — на каждой итерации выводится набор графиков: общий вид АЧХ и крупным планом все отдельные полосы частот;
- '`both`' — одновременный вывод текста и графиков.

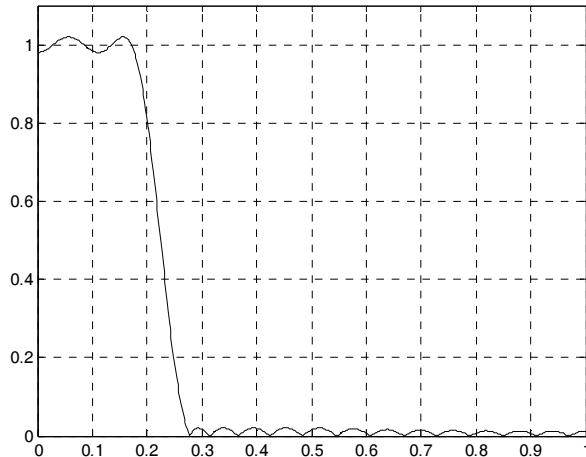
Результатом работы функции является вектор `b` коэффициентов нерекурсивного фильтра. Рассчитанный фильтр имеет линейную ФЧХ и вносит групповую задержку, независимо от частоты равную  $n/2$  отсчетам.

Расчет фильтра осуществляется численным итерационным методом оптимизации с ограничениями.

В качестве примера синтезируем ФНЧ с теми же параметрами, что и в примере для функции `firls` (см. рис. 6.13), задав ограничение отклонения АЧХ от желаемой, равное 0,02 (напомним, что амплитуда максимальных выбросов АЧХ при минимизации среднеквадратической ошибки без ограничений составила примерно 0,05). Функция `fircls` осуществляет аппроксимацию кусочно-постоянной АЧХ, не позволяя задавать переходные полосы, поэтому зададим для границы между полосами пропускания и задерживания значение 0,225, равное середине переходной полосы из примера для функции `firls`. График АЧХ полученного фильтра приведен на рис. 6.14:

```
>> f = [0 0.225 1];
>> a = [1 0];
>> lo = [0.98 -0.02]; % минимальный коэффициент передачи
>> up = [1.02 0.02]; % максимальный коэффициент передачи
>> b = fircls(32, f, a, up, lo);
>> [h, w] = freqz(b);
>> plot(w/pi, abs(h)) % график АЧХ
>> grid on
>> ylim([0 1.1])
```

Сравнение АЧХ на рис. 6.13 и 6.14 показывает, что во втором случае амплитуда пульсаций действительно уменьшилась до заданного уровня 0,02, но при этом расширилась переходная полоса, ширину которой функция `fircls` выбирает автоматически.



**Рис. 6.14.** АЧХ ФНЧ, синтезированного путем минимизации среднеквадратической ошибки с ограничением минимального отклонения с помощью функции `fircls`

### Функция `fircls1`

Функция `fircls1` — это специализированная версия функции `fircls`, предназначенная для синтеза ФНЧ и ФВЧ с линейной ФЧХ путем минимизации среднеквадратической ошибки воспроизведения идеальной частотной характеристики с ограничением предельно допустимых отклонений от нее (*constrained least square*). Кроме того, функция дает возможность более гибкого задания допустимых отклонений АЧХ в полосах пропускания и задерживания, а также учета ошибки в этих полосах с разным весом. Синтаксис вызова функции следующий:

```
b = fircls1(n, w0, dp, ds, 'ftype')
```

Здесь  $n$  — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать  $n + 1$  ненулевых отсчетов).

Параметр  $w_0$  задает частоту среза, как обычно, нормированную к частоте Найквиста.

Параметры  $dp$  и  $ds$  задают допустимые абсолютные уровни пульсаций в полосах пропускания (passband) и задерживания (stopband) соответственно. В полосе пропускания фильтр должен иметь коэффициент передачи, лежащий в пределах  $1 \pm dp$ , а в полосе задерживания — в пределах  $\pm ds$  (здесь имеется в виду не АЧХ, а вещественная частотная характеристика, получающаяся после компенсации вносимой фильтром задержки, поэтому возможны и отрицательные значения).

При отсутствии параметра 'ftype' синтезируется ФНЧ, при 'ftype' = 'high' — ФВЧ. Для ФВЧ, поскольку их коэффициент передачи на частоте Найквиста не равен нулю, возможен только четный порядок. Если задать нечетное  $n$ , будет рассчитан фильтр на единицу большего порядка.

Между  $ds$  и 'ftype' могут находиться дополнительные параметры, причем их значение меняется в зависимости от их количества, поэтому данные варианты син-

таксиса мы рассматриваем отдельно. Первый такой вариант выглядит следующим образом:

```
b = fircls1(n, w0, dp, ds, wt, 'ftype')
```

Появившийся здесь параметр `wt` указывает нормированную частоту, за которой должно удовлетворяться ограничение `dp` или `ds`. Это облегчает работу функции, позволяя образовать переходную зону между полосами пропускания и задерживания. Как именно формируется переходная зона, зависит от соотношения между `wt` и `w0` и от типа синтезируемого фильтра:

□ если синтезируется ФНЧ:

- $wt < w0$ : коэффициент передачи лежит в пределах  $1 \pm dp$  при  $0 \leq \omega \leq wt$ ;
- $wt > w0$ : коэффициент передачи лежит в пределах  $\pm ds$  при  $wt \leq \omega \leq 1$ ;

□ если синтезируется ФВЧ:

- $wt < w0$ : коэффициент передачи лежит в пределах  $\pm ds$  при  $0 \leq \omega \leq wt$ ;
- $wt > w0$ : коэффициент передачи лежит в пределах  $1 \pm dp$  при  $wt \leq \omega \leq 1$ .

Следующий вариант синтаксиса содержит три дополнительных параметра:

```
b = fircls1(n, w0, dp, ds, wp, ws, k, 'ftype')
```

В этом случае можно задать соотношение между ошибками воспроизведения АЧХ в полосах пропускания и задерживания. Это соотношение задается параметром `k`, а параметры `wp` и `ws` указывают границы полос пропускания и задерживания для расчета соотношения ошибок:

□ при синтезе ФНЧ должны выполняться неравенства  $wp < w0 < ws$ , а `k` рассчитывается по формуле

$$k = \frac{\int_0^{\omega_p} |A(\omega) - 1|^2 d\omega}{\int_{\omega_s}^1 |A(\omega)|^2 d\omega};$$

□ при синтезе ФВЧ должны выполняться неравенства  $ws < w0 < wp$ , а `k` рассчитывается по формуле

$$k = \frac{\int_{\omega_p}^1 |A(\omega) - 1|^2 d\omega}{\int_0^{\omega_s} |A(\omega)|^2 d\omega}.$$

Здесь  $A(\omega)$  — АЧХ синтезированного фильтра, а частоты считаются нормированными к частоте Найквиста.

Наконец, в любом варианте синтаксиса в конец списка параметров можно добавить строковый параметр `'design_flag'`:

```
b = fircls1(..., 'design_flag')
```

Этот параметр позволяет управлять выводом информации о ходе поиска решения (по умолчанию эта информация не выводится).

Возможны три значения данного параметра:

- 'trace' — текстовый вывод величины ошибки для каждой итерации;
- 'plots' — на каждой итерации выводится набор графиков: общий вид АЧХ и крупным планом все отдельные полосы частот;
- 'both' — одновременный вывод текста и графиков.

Результатом работы функции является вектор  $b$  коэффициентов нерекурсивного фильтра. Рассчитанный фильтр имеет линейную ФЧХ и вносит групповую задержку, независимо от частоты равную  $n/2$  отсчетам.

Расчет фильтра осуществляется численным итерационным методом оптимизации с ограничениями.

## Реализация метода Ремеза

Как говорилось ранее в разд. "Прямые методы синтеза" этой главы, метод Ремеза предназначен для синтеза нерекурсивных фильтров путем минимаксной аппроксимации заданной АЧХ. В настоящее время в MATLAB имеется три функции, реализующие данный метод и несколько отличающиеся по своим возможностям. Две из них, `firpm` и `cfirpm`, находятся в пакете Signal Processing, а третья, `firgr` (она является наиболее обобщенным вариантом) — в пакете Filter Design. Кроме того, в пакете Signal Processing имеется функция `firpmord`, предназначенная для оценки порядка фильтра, который при синтезе методом Ремеза будет удовлетворять заданным требованиям.

### ЗАМЕЧАНИЕ

В старых версиях пакетов расширения Signal Processing и Filter Design, входящих в состав MATLAB версий до 6.x включительно, рассматриваемые в этом разделе функции имели имена `remez` (вместо `firpm`), `cremez` (вместо `cfirpm`), `gremez` (вместо `firgr`), `remezord` (вместо `firpmord`). Функции со старыми именами по-прежнему существуют, но объявлены устаревшими.

## Функция `firpm`

Функция `firpm` предназначена для синтеза нерекурсивных фильтров с линейной ФЧХ и кусочно-линейной АЧХ с возможными зонами неопределенности. Возможно также задание произвольной формы АЧХ с помощью функции пользователя (см. далее). Синтаксис вызова функции следующий:

```
[b, delta, opt] = firpm(n, f, a, w, 'ftype', {lgrid})
```

Здесь  $n$  — порядок рассчитываемого фильтра (его импульсная характеристика будет содержать  $n + 1$  ненулевых отсчетов).

Параметры  $f$  и  $a$  должны быть векторами одинаковой четной длины, они совместно определяют желаемую АЧХ синтезируемого фильтра. Вектор  $f$  содержит значения частот, нормированные к частоте Найквиста, а вектор  $a$  — соответствующие этим

частотам значения АЧХ. В промежутках между заданными точками АЧХ интерполируется по линейному закону. Частоты в векторе  $f$  должны образовывать строго возрастающую последовательность (повторяющиеся элементы не допускаются).

Синтезируемая АЧХ определяется с помощью векторов  $f$  и  $a$  следующим образом:

- в полосах частот от  $f(2k - 1)$  до  $f(2k)$  АЧХ линейно меняется от  $a(2k - 1)$  до  $a(2k)$ . Здесь  $k$  — целые числа от единицы до половины длины векторов  $f$  и  $a$ ;
- в полосах частот от  $f(2k)$  до  $f(2k + 1)$  АЧХ считается не заданной и в процессе синтеза может оказаться произвольной.

Вывести график синтезируемой АЧХ можно следующей командой:

```
>> plot(reshape(f, 2, length(f)/2), reshape(a, 2, length(a)/2), 'b')
```

### **ЗАМЕЧАНИЕ**

Если не задать синий цвет линий принудительно с помощью строки '`b`', отдельные линейные фрагменты будут выводиться разными цветами.

Для фильтров, коэффициент передачи которых на частоте Найквиста не равен нулю, возможен только четный порядок. Если задать нечетное  $n$ , будет рассчитан фильтр на единицу большего порядка.

Остальные параметры функции являются необязательными. Параметр  $w$  задает весовые коэффициенты для оптимизируемых частотных полос. Этот параметр должен быть вектором, длина которого вдвое меньше, чем длина векторов  $f$  и  $a$ , и содержать неотрицательные вещественные числа. В тех частотных полосах, которым приписаны большие весовые коэффициенты, будет обеспечена меньшая ошибка воспроизведения заданной АЧХ. По умолчанию все весовые коэффициенты считаются одинаковыми.

При отсутствии параметра '`fotype`' производится расчет фильтров с симметричной импульсной характеристикой, для которой  $b(k) = b(n + 2 - k)$ . Если использовать параметр '`fotype`', становится возможным синтезировать фильтры с антисимметричной импульсной характеристикой, когда  $b(k) = -b(n + 2 - k)$ . Данный параметр может принимать одно из двух строковых значений:

- '`hilbert`' — синтезируется фильтр с антисимметричной импульсной характеристикой. Название данного режима объясняется тем, что такую характеристику, в частности, должен иметь фильтр, реализующий преобразование Гильберта (см. разд. "Дискретное преобразование Гильберта" главы 4);
- '`differentiator`' — синтезируется фильтр с антисимметричной импульсной характеристикой и, кроме того, при расчете взвешенной ошибки дополнительно используется весовой множитель  $1/f$ . В результате ошибка воспроизведения АЧХ на низких частотах будет намного меньше, чем на высоких. При синтезе дифференцирующего фильтра, АЧХ которого пропорциональна частоте (см. разд. "Свойства преобразования Фурье" главы 1), это означает минимизацию относительной ошибки воспроизведения заданной АЧХ.

### ЗАМЕЧАНИЕ

Обратите внимание на то, что в режиме 'differentiator' функциями `firpm` и `firls` используются разные дополнительные весовые функции —  $1/f$  и  $1/f^2$  соответственно. Это объясняется различием используемых критериев оптимизации — функция `firpm` минимизирует максимальное отклонение АЧХ от заданной (ошибка линейно связана с АЧХ), а функция `firls` минимизирует ошибку, связь которой с АЧХ является квадратичной.

Параметр `{lgrid}`, хотя и является целым положительным числом, должен быть задан в виде одноэлементного массива ячеек (т. е. в фигурных скобках). Этот параметр регулирует плотность сетки частот, которая используется при расчете АЧХ фильтра с целью поиска экстремумов. Число точек сетки приблизительно равно  $lgrid \cdot n / (2 \cdot bw)$ , где `bw` — суммарная полоса пропускания фильтра, нормированная к частоте Найквиста. Увеличение параметра `lgrid` может сделать пульсации более равномерными, но приведет к увеличению времени расчета. По умолчанию значение параметра `lgrid` равно 16.

Главным результатом работы функции является вектор `b` коэффициентов нерекурсивного фильтра. Рассчитанный фильтр имеет линейную ФЧХ и вносит групповую задержку, независимо от частоты равную  $n/2$  отсчетам. Кроме того, можно получить дополнительную информацию о результатах синтеза с помощью параметров `delta` и `opt`.

Выходной параметр `delta` содержит величину пульсаций АЧХ полученного фильтра (т. е. значение максимального отклонения АЧХ от заданной).

Выходной параметр `opt` представляет собой структуру со следующими полями:

- `opt.fgrid` — вектор сетки частот, использованной при синтезе;
- `opt.des` — значения заданной АЧХ для частот из `opt.fgrid`;
- `opt.wt` — весовые коэффициенты для частот из `opt.fgrid`;
- `opt.H` — получившаяся комплексная частотная характеристика фильтра для частот из `opt.fgrid`;
- `opt.error` — ошибка воспроизведения заданной частотной характеристики для частот из `opt.fgrid`;
- `opt.iextr` — номера элементов вектора `opt.fgrid`, соответствующих экстремальным частотам (частотам, на которых отклонение АЧХ от заданной максимально);
- `opt.fextr` — вектор значений экстремальных частот.

Для ускорения расчетов основная часть алгоритма синтеза реализована в машинных кодах (в виде DLL-файла).

В процессе расчета может быть выдано сообщение о том, что алгоритм не сходится (возможно, из-за ошибок округления):

```
-- Failure to Converge --
Probable cause is machine rounding error.
```

В этом случае следует обязательно проверить результаты синтеза, рассчитав АЧХ полученного фильтра с помощью функции `freqz`. Впрочем, такой контроль полезен всегда.

### **Задание пользовательской функции расчета АЧХ**

Если АЧХ, которую необходимо синтезировать, не является кусочно-линейной, можно воспользоваться другим вариантом синтаксиса функции `firpm`:

```
[b, delta, opt] = firpm(n, f, @fresp, w, 'ftype', {lgrid})
```

Здесь `fresp` — имя функции, рассчитывающей желаемую АЧХ. Остальные параметры имеют то же назначение, что и ранее. Функция `fresp` должна вызываться следующим образом:

```
[dh, dw] = fresp(n, f, gf, w)
```

Входные параметры `n`, `f` и `w` здесь те же, что используются при вызове функции `firpm`, а вектор `gf` — сетка частот, сформированная функцией `firpm` исходя из значений параметров `n`, `f` и `lgrid`.

Функция `fresp` должна вычислить значения АЧХ `dh` и весовых коэффициентов `dw` для частот из вектора `gf`. Остальные входные параметры (`n`, `f` и `w`), в принципе, при расчете могут не использоваться. Их передача в функцию `fresp` позволяет реализовать дополнительную гибкость.

При необходимости можно организовать передачу в функцию `fresp` дополнительных параметров. Для этого при вызове функции `firpm` необходимо взять указатель на функцию расчета АЧХ вместе с ее дополнительными параметрами в фигурные скобки:

```
... = firpm(n, f, {@fresp, p1, p2 ...}, w, 'ftype', {lgrid})
```

Функция `fresp` при этом будет вызвана так:

```
[dh, dw] = fresp(n, f, gf, w, p1, p2 ...)
```

В каталоге `toolbox\signal\signal\private` (содержимое данного каталога доступно только функциям пакета Signal Processing) имеется несколько готовых функций расчета АЧХ, которые можно использовать в качестве параметров при вызове `firpm` и `cfirpm`. Так, функция `firpmfrf` реализует кусочно-линейную интерполяцию АЧХ и используется по умолчанию функцией `firpm`. Данная функция вызывается следующим образом: `{'firpmfrf', a}`. Здесь `a` — вектор значений АЧХ, соответствующих частотам из вектора `f`. Таким образом, приведенный ранее первый вариант синтаксиса вызова функции `firpm` на самом деле трансформируется так:

```
... = firpm(n, f, {@firpmfrf, a}, w, 'ftype', {lgrid})
```

Другие функции этого семейства имеют следующие имена: `allpass`, `lowpass`, `highpass`, `bandpass`, `bandstop`, `multiband`, `inv sinc`, `hilbfilt`, `differentiator`. Справку об их использовании можно получить в командной строке MATLAB, введя команду

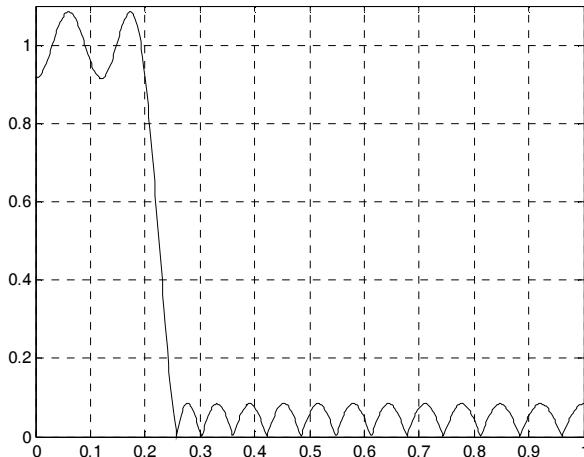
```
help private/имя_функции
```

Текст перечисленных функций можно использовать в качестве образца при написании собственных функций расчета АЧХ.

### Пример использования

В качестве примера синтезируем методом Ремеза ФНЧ с теми же параметрами, что и при демонстрации использования функций `firls` и `fircls` (см. рис. 6.13 и 6.14). График АЧХ полученного фильтра представлен на рис. 6.15:

```
>> f = [0 0.2 0.25 1];
>> a = [1 1 0 0];
>> b = firpm(32, f, a);
>> [h, w] = freqz(b);
>> plot(w/pi, abs(h)) % график АЧХ
>> grid on
>> ylim([0 1.1])
```



**Рис. 6.15.** АЧХ ФНЧ, синтезированного путем минимаксной аппроксимации с помощью функции `firpm`

Сравнение АЧХ на рис. 6.13, 6.14 и 6.15 показывает, что пульсации АЧХ получились больше, чем при минимизации среднеквадратической ошибки (их амплитуда составляет примерно 0,085). Однако фильтр, синтезированный методом Ремеза, дает минимальную ширину переходной полосы (равномерный характер пульсаций в полосах пропускания и задерживания строго выдержан, тогда как при минимизации среднеквадратической ошибки АЧХ далеко отходит от заданных значений при приближении к переходной зоне). Если задать границы переходной зоны такими, какими они оказались в действительности при использовании функции `fircls` (примерно от 0,175 до 0,275, см. рис. 6.14), синтез методом Ремеза даст такие же или меньшие пульсации. Убедимся в этом, не выводя график АЧХ, а проверив значение второго выходного параметра функции `firpm`:

```
>> f = [0 0.175 0.275 1];
>> a = [1 1 0 0];
>> [b, d] = firpm(32, f, a);
>> disp(d) % вывод амплитуды пульсаций
    0.0176
```

Как видите, в данном случае пульсации действительно оказались несколько меньше, чем при минимизации среднеквадратической ошибки с ограничением максимального отклонения.

## Функция *cfirpm*

Функция *cfirpm* является расширенным вариантом функции *firpm*, имеющим следующие отличия:

- ФЧХ фильтра не обязательно должна быть линейной;
- можно синтезировать фильтры с комплексными коэффициентами;
- тип симметрии импульсной характеристики задается в явном виде, а не косвенно;
- в алгоритм расчета добавлена вторая стадия, уточняющая решение, найденное методом Ремеза;
- имеется некоторое количество предопределенных функций расчета АЧХ.

Синтаксис вызова функции следующий:

```
[b, delta, opt] = cfirpm(n, f, a, w, {lgrid}, ...
    'sym', 'skip_stage2', 'debug')
```

или

```
[b, delta, opt] = cfirpm(n, f, {@fresp, p1, p2 ...}, ...
    w, {lgrid}, 'sym', 'skip_stage2', 'debug')
```

Выходные и почти все входные параметры совпадают с параметрами функции *firpm*, рассмотренной ранее. Поэтому остановимся только на имеющихся различиях.

Частоты в векторе *f* должны лежать в диапазоне  $-1 \dots 1$ . Возможность задавать поведение АЧХ в области отрицательных частот позволяет рассчитывать фильтры с комплексными коэффициентами.

Строковый параметр '*sym*' позволяет управлять симметрией импульсной характеристики. Возможны следующие четыре значения:

- '*none*' — ограничений симметрии не накладывается;
- '*even*' — четная симметрия импульсной характеристики:  $b(k) = b(n + 2 - k)$ ;
- '*odd*' — нечетная симметрия импульсной характеристики:  $b(k) = -b(n + 2 - k)$ ;
- '*real*' — комплексно-сопряженная симметрия импульсной характеристики:  $b(k) = b^*(n + 2 - k)$  (этот вариант позволяет получить фильтр с линейной ФЧХ и несимметричной относительно нулевой частоты АЧХ).

Использование отрицательных частот в векторе  $f$  возможно только при '`sym`' = 'none' или 'real'.

При указании в списке параметров строки '`skip_stage2`' отключается вторая стадия алгоритма (она выполняется, если оптимальное решение не найдено стандартным методом Ремеза). Это может повысить скорость расчетов за счет уменьшения точности.

Последний входной параметр '`debug`' управляет выдачей сообщений о ходе оптимизации. Возможны следующие строковые значения:

- 'off' — сообщения не выводятся (этот вариант используется по умолчанию);
- 'trace' — выводятся текстовые сообщения;
- 'plots' — выводятся графики;
- 'both' — выводятся текстовые сообщения и графики.

### Функция `firpmord`

Функция `firpmord` предназначена для оценки требуемого при синтезе методом Ремеза порядка многополосного нерекурсивного фильтра с заданной величиной пульсаций АЧХ. Синтаксис вызова функции следующий:

```
[n, fo, ao, w] = firpmord(f, a, dev, Fs)
```

Параметры  $f$  и  $a$  совместно задают кусочно-постоянную АЧХ. В векторе  $f$  должна содержаться возрастающая последовательность значений частот. Вектор  $a$  задает постоянные значения АЧХ для полос, ограниченных парами частот из вектора  $f$ , при этом первая полоса начинается от нулевой частоты, а последняя заканчивается на частоте Найквиста:

- в полосе частот от нуля до  $f(1)$  АЧХ равна  $a(1)$ ;
- в полосе частот от  $f(1)$  до  $f(2)$  АЧХ не определена;
- в полосе частот от  $f(2)$  до  $f(3)$  АЧХ равна  $a(2)$ ;
- в полосе частот от  $f(3)$  до  $f(4)$  АЧХ не определена;
- так продолжается до конца векторов  $f$  и  $a$ ;
- в полосе частот от  $f(end-1)$  до  $f(end)$  АЧХ не определена;
- в полосе частот от  $f(end)$  до  $Fs/2$  АЧХ равна  $a(end)$ .

Таким образом, длина вектора  $f$  должна быть равна `length(a)*2-2`.

Параметр `dev` должен быть вектором той же длины, что и  $a$ , содержащим вещественные положительные числа. Он задает максимально допустимую ошибку воспроизведения кусочно-постоянной АЧХ для отдельных частотных полос. Ошибка задается как *абсолютная* погрешность АЧХ.

Параметр `Fs` задает частоту дискретизации и является необязательным. При его использовании следует указывать в векторе  $f$  не нормированные, а исходные значения частот в тех же единицах, что и `Fs`. По умолчанию частота дискретизации считается равной двум, так что частота Найквиста равна единице.

Результатами работы функции являются порядок фильтра  $n$  и векторы  $fo$ ,  $ao$  и  $w$ , которые предназначены для использования при вызове функции `firpm`:

```
b = firpm(n, fo, ao, w)
```

Результаты работы функции могут быть неточными, если у заданной АЧХ имеются частоты среза, близкие к нулю или частоте Найквиста. Кроме того, в любом случае оценка порядка фильтра, производимая функцией `firpmord`, является приближенной, поэтому если синтезированный фильтр не удовлетворяет заданным требованиям, необходимо более точно подобрать порядок фильтра вручную.

#### **ЗАМЕЧАНИЕ**

Механизм автоматического подбора минимально необходимого порядка фильтра встроен в функцию `firgr` пакета Filter Design (см. далее).

## **Функции пакета Filter Design**

Пакет Filter Design содержит около десятка функций, предназначенных для синтеза дискретных фильтров. Мы рассмотрим функцию `firgr`, представляющую собой еще один расширенный вариант метода Ремеза, и четыре функции, минимизирующие  $p$ -норму ошибки при произвольном  $p$ .

### **Функция `firgr`**

Функция `firgr`, входящая в пакет Filter Design, обладает расширенными возможностями по сравнению с функциями `firpm` и `cfirpm`. Основными из этих расширений являются следующие:

- возможность автоматического подбора порядка фильтра;
- возможность описания *особых точек* АЧХ;
- большее количество информации, возвращаемой в структуре `opt`;
- возможность группировки частотных полос для расчета нескольких отдельных ошибок аппроксимации.

#### **ЗАМЕЧАНИЕ**

Ранее в данной функции существовала также возможность задания ограничений величины пульсаций для отдельных частотных полос. В настоящее время этот вариант алгоритма реализован в виде отдельной функции `fircband`.

### **Новые поля структуры `opt`**

В простейшем варианте синтаксис вызова функции `firgr` такой же, как у функции `firpm`:

```
[b, error, opt] = firgr(n, f, a, w)
[b, error, opt] = firgr(n, f, a, 'hilbert')
[b, error, opt] = firgr(n, f, a, 'differentiator')
```

При таком способе вызова работа функции ничем не отличается от того, что делает функция `firpm`, разве что в информационной структуре `opt` присутствует ряд дополнительных полей:

- `opt.order` — порядок синтезированного фильтра;
- `opt.edgeCheck` — результаты проверки аномалий в переходных зонах (см. далее). Это вектор, содержащий по одному элементу на каждую частотную полосу. Возможны следующие значения элементов вектора: 1 — все в порядке, 0 — обнаружена аномалия, -1 — проверка не выполнялась;
- `opt.iterations` — число выполненных итераций;
- `opt.evals` — число вычислений целевой функции.

Поддерживается и вариант синтаксиса с задаваемой пользователем функцией расчета АЧХ:

```
[b, error, opt] = firgr(n, f, {@fresp, p1, p2 ...}, w)
```

По умолчанию в качестве `fresp` используется функция `firpmfrf2`, расположенная в каталоге `toolbox\filterdesign\filterdesign\private` и реализующая расчет кусочно-линейной АЧХ.

## Подбор порядка фильтра

Для реализации автоматического подбора порядка фильтра при заданной величине пульсаций АЧХ функция `firgr` вызывается следующим образом:

```
[b, error, opt] = firgr('m', f, a, r)
```

Вместо порядка фильтра указывается строковый параметр '`m`', который может принимать одно из трех следующих значений:

- '`minorder`' — подбирается минимальный порядок фильтра, при котором отклонения АЧХ от заданной лежат в допустимых пределах;
- '`mineven`' — подбирается минимально возможный четный порядок фильтра, при котором отклонения АЧХ от заданной лежат в допустимых пределах;
- '`minodd`' — подбирается минимально возможный нечетный порядок фильтра, при котором отклонения АЧХ от заданной лежат в допустимых пределах.

Параметр `r` — вектор допустимых абсолютных значений пульсаций АЧХ для отдельных частотных полос. Его длина должна быть в два раза меньше длины вектора `f`.

Начальное приближение `ni` для подбора порядка фильтра можно указать следующим образом:

```
[b, error, opt] = firgr({'m', ni}, f, a, r)
```

Такое указание начального значения порядка фильтра является обязательным в тех случаях, когда нельзя использовать функцию `firpmord` (например, при расчете дифференцирующих фильтров и преобразователей Гильберта).

## Использование независимых ошибок аппроксимации

Чтобы независимо минимизировать величину пульсаций АЧХ в разных полосах частот, функция `firgr` вызывается следующим образом:

```
[b, error, opt] = firgr(n, f, a, w, e)
```

Параметр `e` определяет, как группируются частотные полосы для расчета независимых ошибок аппроксимации. Этот параметр представляет собой массив ячеек с числом элементов, равным длине вектора `w`. Каждый элемент `e{k}` должен быть строкой вида '`e#`', где `#` — целое положительное число. Это означает, что  $k$ -я полоса частот принимает участие в расчете ошибки аппроксимации с номером `#`.

В данном режиме функция `firgr` пытается независимо минимизировать отклонение АЧХ от заданной в нескольких полосах частот. В результате получается фильтр, пульсации АЧХ которого равномерны в каждой группе частотных полос, формирующих общую ошибку аппроксимации, но для разных групп полос пульсации будут разными. Таким образом можно, например, синтезировать фильтр, в полосе пропускания которого пульсации АЧХ будут меньше, чем в полосе задерживания. В результате иногда могут получаться фильтры, имеющие очень узкие переходные зоны между полосами пропускания и задерживания.

По умолчанию параметр `e` имеет значение `{'e1', 'e1', 'e1', ...}`, т. е. все частотные полосы формируют одну общую ошибку аппроксимации АЧХ.

## Описание особых точек АЧХ

Для описания особых точек АЧХ фильтра используется массив ячеек `s`, число элементов в котором должно быть равно длине векторов `f` и `a`:

```
[b, error, opt] = firgr(n, f, a, s)
```

Элементами массива `s` являются односимвольные строки, имеющие следующее значение:

- '`n`' — обычная точка АЧХ (normal point);
- '`s`' — "точечная" полоса (single-point band);
- '`f`' — коэффициент передачи синтезированного фильтра на данной частоте должен быть точно равен заданному (forced frequency point);
- '`i`' — коэффициент передачи на данной частоте не определен (indeterminate frequency point; такую точку можно использовать на границе вплотную примыкающих друг к другу частотных полос).

## Указание дополнительных свойств синтезируемого фильтра

Для синтеза фильтра, обладающего некоторыми дополнительными свойствами, в конце списка параметров функции `firgr` могут использоваться следующие строевые значения:

- '`1`', '`2`', '`3`' или '`4`' — синтезируется фильтр соответствующего типа (см. табл. 4.1 в разд. "Симметричные фильтры" главы 4);

- 'minphase' или 'maxphase' — синтезируется соответственно *минимально-фазовый* фильтр (все нули функции передачи лежат на  $z$ -плоскости *внутри* единичной окружности или на ней) или *максимально-фазовый* фильтр (все нули функции передачи лежат на  $z$ -плоскости *снаружи* единичной окружности или на ней);
- 'check' — выполняется проверка наличия аномалий в переходных зонах и при их обнаружении выдается соответствующее предупреждение. Информация об обнаруженных аномалиях возвращается также в поле `edgeCheck` структуры `opt` (см. ранее).

## Функции, использующие $p$ -норму ошибки

Перечисленные далее функции выполняют прямой синтез фильтров путем минимизации  $p$ -нормы ошибки воспроизведения заданной характеристики (АЧХ или групповой задержки):

- `firlpnorm` — расчет нерекурсивного фильтра по заданной АЧХ:

```
[b, err] = firlpnorm(n, f, edges, a, w, p, dens, b0)
```

- `iirlpnorm` — расчет рекурсивного фильтра по заданной АЧХ:

```
[b, a, err] = iirlpnorm(n, d, f, edges, a, w, p, dens, b0, a0)
```

- `iirlpnormc` — расчет рекурсивного фильтра по заданной АЧХ с ограничением модуля полюсов:

```
[b, a, err] = iirlpnormc(n, d, f, edges, a, w, radius, p, dens, b0, a0)
```

- `iirgrpdelay` — расчет *всепропускающего* (см. разд. "Нули и полюсы" главы 4) рекурсивного фильтра по заданной зависимости групповой задержки от частоты:

```
[b, a, tau] = iirgrpdelay(n, f, edges, gd, w, radius, p, dens, a0, tau)
```

Параметры всех этих функций задаются одинаково и имеют следующий смысл:

- `n` — порядок числителя функции передачи фильтра;
- `d` — порядок знаменателя функции передачи фильтра;
- `f` — вектор частот для задания частотной характеристики фильтра (вектор должен содержать возрастающую последовательность значений, нормированных к частоте Найквиста, причем должны выполняться равенства  $f(1) = 0$  и  $f(\text{end}) = 1$ );
- `a` — вектор значений АЧХ для частот из вектора `f` (используется кусочно-линейная интерполяция);
- `gd` — вектор значений групповой задержки, измеряемой в отсчетах, для частот из вектора `f` (используется кусочно-линейная интерполяция);
- `edges` — вектор четной длины, задающий границы частотных полос, в промежутках между которыми АЧХ считается незаданной. Таким образом, АЧХ считается *заданной* в полосах `edges(1)...edges(2)`, `edges(3)...edges(4)` и т. д., и *неопределенной* в полосах `edges(2)...edges(3)`, `edges(4)...edges(1)` и т. д.

ределенной в полосах  $0 \dots edges(1)$ ,  $edges(2) \dots edges(3)$  и т. д. Вектор `edges` должен содержать значения частот, имеющиеся в векторе `f`.

Приведенные далее параметры являются необязательными и имеют значения по умолчанию:

- `w` — вектор весовых коэффициентов. Он должен иметь такой же размер, как векторы `f`, `a` и `gd`, и содержать неотрицательные вещественные числа. По умолчанию все элементы этого вектора равны единице;
- `radius` — предельно допустимая величина модулей полюсов синтезируемого фильтра. По умолчанию используется значение 0,999999;
- `p` — двухэлементный вектор вида `[pmin pmax]`, указывающий границы для параметра `p` используемой при расчете нормы ошибки. Элементы вектора должны быть целыми положительными четными числами. По умолчанию используется значение `[2 128]`, что дает результаты, близкие к минимаксной оптимизации. Возможно также значение '`inspect`', при этом оптимизация не производится и функция выдает используемое ею начальное приближение;
- `dens` — плотность частотной сетки, используемой функцией для расчета ошибок. Число точек сетки равно `dens * (n+1)`. По умолчанию используется значение `dens = 20`;
- `b0` — начальное приближение для числителя функции передачи фильтра;
- `a0` — начальное приближение для знаменателя функции передачи фильтра;
- `tau` — дополнительное смещение групповой задержки, используемое функцией `iirgrpdelay` для того, чтобы сделать заданную групповую задержку неотрицательной на всех частотах (всепропускающие фильтры могут иметь только положительную групповую задержку). По умолчанию `tau = max(gd)`.

Результатами работы функций являются векторы `b` и `a` коэффициентов полиномов числителя и знаменателя функции передачи фильтра, а также достигнутое минимальное значение  $p$ -нормы ошибки `err`. Функция `iirgrpdelay` может также возвращать дополнительное смещение групповой задержки `tau`, использованное при синтезе.

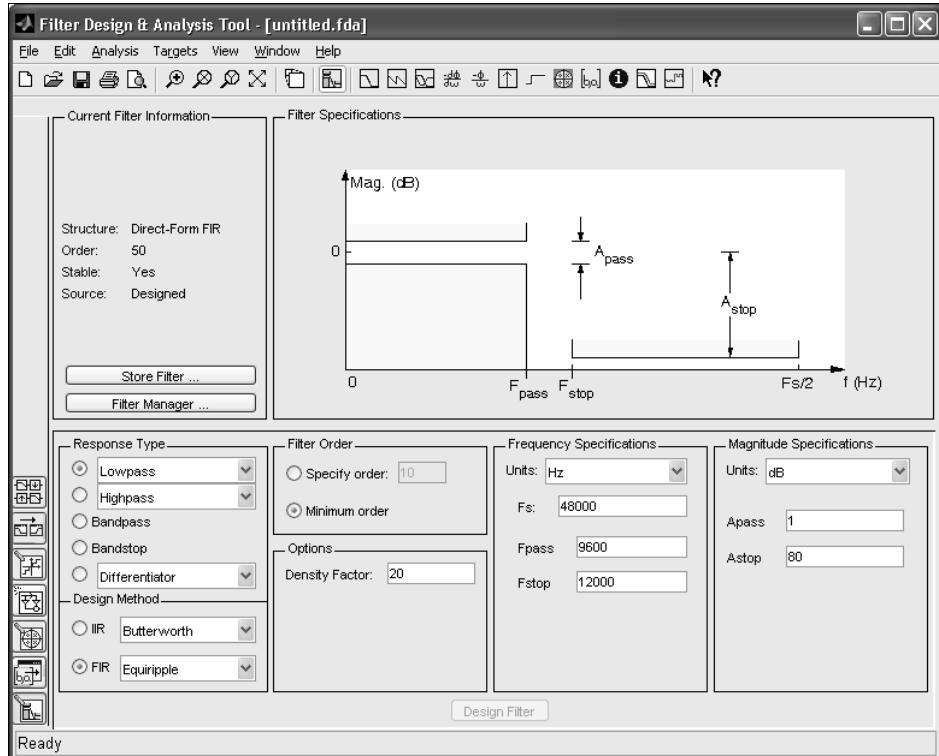
## Графическая среда для синтеза и анализа фильтров

В пакете Signal Processing имеется две графических среды, позволяющие рассчитывать и анализировать дискретные фильтры. Это среда *FDATool* (Filter Design & Analysis Tool) и "построитель фильтров" *FilterBuilder*. Первая из них существует уже давно, вторая появилась лишь в последних версиях пакетов Signal Processing и Filter Design. С какой целью разработчиками созданы два разных пользовательских интерфейса для выполнения одних и тех же операций, на данный момент не вполне понятно. Поэтому далее в этой главе описываются возможности графического интерфейса FDATool, а затем приводятся лишь краткие сведения о программе FilterBuilder.

Для запуска программы расчета фильтров необходимо набрать ее имя в командной строке MATLAB:

```
>> fdatool
```

После этого появится окно программы, показанное на рис. 6.16.



**Рис. 6.16.** Интерфейс программы FDATool

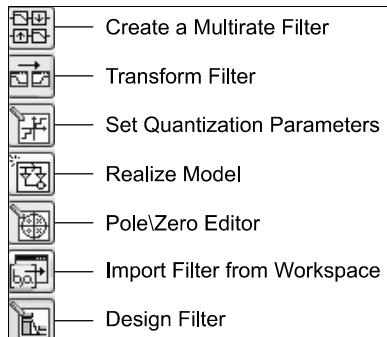
Большая часть команд меню программы FDATool дублирует действия, выполняемые с помощью элементов графического интерфейса, поэтому отдельно обсуждать структуру меню мы не будем. Однако следует отметить, что многим операциям сопоставлены клавиатурные комбинации, узнать которые можно, заглянув именно в меню.

## Переключение вкладок

В левом нижнем углу окна среды FDATool расположены семь кнопок (рис. 6.17), позволяющих переключаться между различными панелями ввода данных (условимся называть их вкладками):

- **Design Filter** (см. рис. 6.16) — вкладка ввода данных для синтеза фильтра;
- **Import Filter from Workspace** (см. далее рис. 6.20) — вкладка, позволяющая импортировать фильтр для анализа его параметров в среде FDATool;

- Pole\Zero Editor** (см. далее рис. 6.21) — вкладка, позволяющая редактировать расположение нулей и полюсов функции передачи фильтра;
- Realize Model** — вкладка, позволяющая создать блок фильтра для модели Simulink (имеется, если установлен набор блоков DSP Blockset). Рассмотрение взаимодействия с Simulink выходит за рамки тематики данной книги; дополнительную информацию можно найти в справочной системе MATLAB;
- Set Quantization Parameters** — вкладка ввода параметров квантования фильтра (имеется, если установлен пакет Filter Design). Об эффектах квантования и об использовании этой вкладки речь пойдет далее, в *главе 7*;
- Transform Filter** (см. далее рис. 6.22) — вкладка трансформации частотной оси (имеется, если установлен пакет Filter Design);
- Create a Multirate Filter** — вкладка расчета многоскоростных фильтров, изменяющих частоту дискретизации сигнала. О многоскоростной обработке сигналов речь пойдет далее, в *главе 10*.



**Рис. 6.17.** Кнопки  
переключения вкладок  
FDATool

## Расчет фильтра

Расчет фильтра начинается с задания требуемых параметров на вкладке **Design Filter** (см. рис. 6.16). Тип синтезируемой АЧХ выбирается с помощью переключателя **Response Type**. Возможны следующие варианты: **Lowpass** (ФНЧ; раскрывающийся список позволяет выбрать еще несколько разновидностей), **Highpass** (ФВЧ, здесь также имеется список дополнительных вариантов), **Bandpass** (полосовой фильтр), **Bandstop** (режекторный фильтр). Выбор пятого, самого нижнего, положения переключателя позволяет использовать раскрывающийся список, в котором перечислены более сложные варианты: **Differentiator** (дифференцирующий фильтр), **Hilbert Transformer** (преобразователь Гильберта), **Multiband** (многополосный фильтр), **Arbitrary Magnitude** (произвольная АЧХ), **Arbitrary Group Delay** (произвольная групповая задержка), **Peaking** (одиночный или гребенчатый резонатор) и **Notching** (одиночный или гребенчатый режектор).

Выбрав категорию синтезируемой АЧХ, следует выбрать тип синтезируемого фильтра, установив переключатель, расположенный в разделе **Design Method**, в положение **IIR** (рекурсивный) или **FIR** (нерекурсивный). Каждому положению

переключателя соответствует список возможных методов синтеза. Состав этого списка меняется в зависимости от выбранного типа АЧХ; кроме того, набор доступных методов синтеза зависит от того, установлен ли пакет Filter Design.

Далее необходимо выбрать порядок фильтра в разделе **Filter Order**. В ряде случаев, помимо явного указания порядка в поле ввода **Specify order**, возможен автоматический выбор порядка путем установки переключателя в положение **Minimum order**.

Наконец, необходимо задать числовые параметры этой АЧХ в разделах **Frequency Specifications** и **Magnitude Specifications** (при выборе типа АЧХ из дополнительного раскрывающегося списка эти два раздела объединяются под общим названием **Frequency and Magnitude Specifications**). Содержимое этих областей окна меняется в зависимости от выбранного типа АЧХ. На рис. 6.16 показаны поля ввода параметров, соответствующие синтезу ФНЧ. При заполнении полей ввода можно ссылаться на переменные, существующие в данный момент в рабочей области памяти MATLAB.

### **ЗАМЕЧАНИЕ**

Подробное описание способов задания разнообразных АЧХ вы можете найти в разделах этой главы, посвященных функциям синтеза фильтров. Параметры, задаваемые в FDATool, как правило, передаются в функцию, реализующую выбранный метод синтеза, без дополнительных преобразований.

Числовые параметры, которые необходимо задать, иллюстрируются графиком, выводимым в разделе **Filter Specifications**. Вид этого графика также меняется в зависимости от выбранного типа АЧХ, на рис. 6.16 он соответствует синтезу ФНЧ.

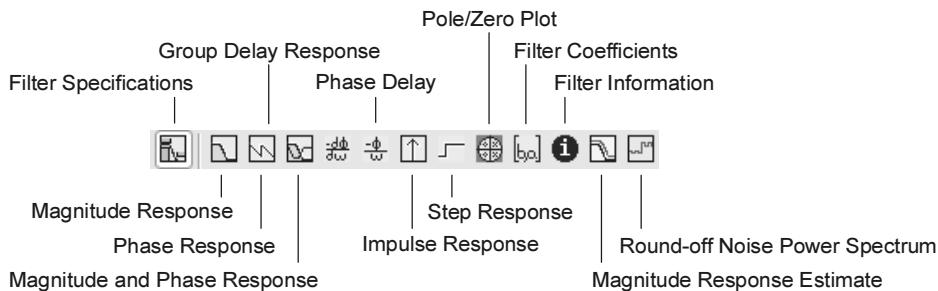
Выбрав метод синтеза и задав характеристики фильтра, щелкните на расположенной в нижней части окна кнопке **Design Filter**. Программа FDATool вызовет нужную функцию синтеза, передав ей указанные вами спецификации фильтра.

## **Просмотр характеристик фильтра**

После выполнения расчета фильтра в разделе **Current Filter Information** окна программы FDATool появится информация о завершении вычислений (**Source: Designed**). Теперь можно просматривать характеристики получившегося фильтра, чтобы проверить, соответствует ли он нашим требованиям.

Выбор графика для просмотра производится с помощью кнопок панели инструментов, показанных на рис. 6.18, или следующих команд меню **Analysis**:

- Filter Specifications** — вывод графика, поясняющего задание параметров АЧХ;
- Magnitude Response** — вывод графика АЧХ;
- Phase Response** — вывод графика ФЧХ;
- Magnitude and Phase Response** — одновременный вывод графиков АЧХ и ФЧХ;
- Group Delay Response** — вывод графика частотной зависимости групповой задержки;
- Phase Delay** — вывод графика частотной зависимости фазовой задержки;



**Рис. 6.18.** Панель инструментов программы FDATool

- Impulse Response** — вывод графика импульсной характеристики;
  - Step Response** — вывод графика переходной характеристики;
  - Pole/Zero Plot** — вывод графика расположения нулей и полюсов на  $z$ -плоскости;
  - Filter Coefficients** — просмотр коэффициентов фильтра. Выводимое здесь представление фильтра зависит от выбранной формы реализации (см. далее);
  - Filter Information** — просмотр общей информации о фильтре (выводятся сведения о структуре фильтра, его порядке, стабильности, линейности ФЧХ);
  - Magnitude Response Estimate** — оценка АЧХ квантованного фильтра с учетом эффектов квантования, производимая путем пропускания через фильтр широкополосного сигнала (это делается с помощью функции `freqrespest`);
  - Round-off Noise Power Spectrum** — оценка спектральной плотности мощности шума округления, возникающего в квантованном фильтре (для этого используется функция `noisepsd`).

## **ЗАМЕЧАНИЕ**

Кнопки **Magnitude Response Estimate**, **Round-off Noise Power Spectrum** и **Turn Quantization on**, связанные с анализом эффектов квантования, будут присутствовать на панели инструментов только при установленном пакете расширения Filter Design Toolbox. Об эффектах квантования речь пойдет далее, в главе 7.

Когда на экран выведен какой-либо из графиков, можно использовать кнопки **Zoom In**, **Zoom X-Axis**, **Zoom Y-Axis** и **Restore Default View** панели инструментов для управления масштабом отображения.

## **Сохранение результатов работы**

Чтобы сохранить результаты текущего сеанса работы с программой FDATool, щелкните на кнопке **Save session** панели инструментов или воспользуйтесь одноименной командой меню **File**. В этом меню есть также команда **Save session As**, позволяющая сохранить сеанс под новым именем. Файлы сохраненных сеансов работы имеют расширение **fda**.

Загрузить сохраненный сеанс можно с помощью кнопки **Open Session** панели инструментов или одноименной команды меню **File**.

Разумеется, ценность программы FDATool была бы невелика, если бы в ней не было средств экспорта коэффициентов рассчитанного фильтра для использования в MATLAB или других программах. Экспорт описания фильтра производится с помощью команды **Export** меню **File** или комбинации клавиш <Ctrl>+<E>. После выбора команды появляется окно экспорта, показанное на рис. 6.19.



**Рис. 6.19.** Окно экспорта  
описания фильтра

Раскрывающийся список **Export To** позволяет выбрать способ экспорта:

- **Workspace** — данные передаются непосредственно в рабочую область памяти MATLAB;
- **Coefficient File (ASCII)** — коэффициенты фильтра записываются в текстовый файл. Формат этого файла совпадает с форматом представления информации о коэффициентах фильтра в окне программы FDATool;
- **MAT-file** — информация о фильтре сохраняется в виде МАТ-файла, который затем можно будет загрузить в MATLAB командой `load`;
- **SPTool** — информация о фильтре экспортируется в среду SPTool, рассматриваемую в *приложении 4*.

Раскрывающийся список **Export As** позволяет выбрать способ представления экспортируемой информации о фильтре:

- **Coefficients** — экспортируются векторы или матрицы, описывающие фильтр;
- **Objects** — экспортируются объекты фильтров (см. разд. "Объекты дискретных фильтров" главы 4).

В разделе **Variable Names** задаются имена переменных для хранения векторов и матриц, описывающих фильтр. Состав полей ввода зависит от формы реализации фильтра (см. далее). При записи информации в текстовый файл или экспорте в SPTool эти поля ввода недоступны.

При экспорте в рабочую область памяти MATLAB доступен также флажок **Overwrite Variables**. Если он установлен, то при наличии в памяти MATLAB переменных, имена которых совпадают с теми, что указаны в окне экспорта, их значения будут заменены новыми. Если флажок снят, то попытка экспортовать данные в уже существующую переменную вызовет появление сообщения об ошибке.

Задав все необходимые параметры, щелкните на кнопке **OK** для выполнения экспорта данных. При экспорте в файл будет запрошено имя создаваемого файла.

### ЗАМЕЧАНИЕ

В меню **File** имеется также команда **Export to Simulink Model**, позволяющая экспортить синтезированный фильтр в виде модели Simulink. Кроме того, с помощью команд меню **Targets** можно экспортить описание фильтра в системы разработки устройств на цифровых сигнальных процессорах или программируемых логических интегральных схемах.

## Импорт описания фильтра

Программу FDATool можно использовать не только для расчета фильтров с заданными параметрами, но и для анализа характеристик уже рассчитанных фильтров. Для этого используется режим *импорта* описания фильтра, выбираемый с помощью команды **Import Filter** меню **Filter**. При включении этого режима вкладка **Design Filter** заменяется на вкладку **Import Filter**, показанную на рис. 6.20.

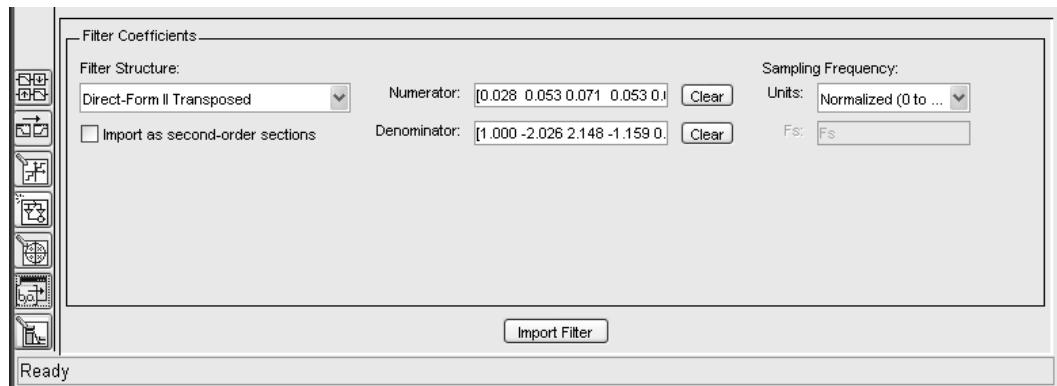


Рис. 6.20. Вкладка Import Filter

Под импортом в программе FDATool подразумевается получение готовых векторов и матриц, описывающих фильтр. Можно ввести эти векторы и матрицы вручную (используя синтаксис MATLAB) или сослаться на идентификаторы переменных, существующих в данный момент в рабочей области памяти MATLAB и хранящих описание фильтра. Эти векторы или имена переменных задаются в полях ввода, состав которых зависит от выбранной формы описания фильтра (форма выбирается с помощью раскрывающегося списка **Filter Structure**). На рис. 6.20 выбрана прямая форма реализации фильтра, поэтому предлагается ввести векторы коэффициентов полиномов числителя (поле **Numerator**) и знаменателя (поле **Denominator**) функции передачи.

В разделе **Sampling Frequency** задается частота дискретизации — она используется для оцифровки частотных осей графиков. Единица измерения частоты выбирается из списка **Units**, а само значение вводится в поле **Fs**. По умолчанию выбраны нормированные частоты, при этом частота Найквиста равна единице.

Задав каким-либо способом описание фильтра, щелкните на кнопке **Import Filter**, расположенной в нижней части окна программы, и выводимый в данный момент график будет перерисован в соответствии с результатами анализа импортированного фильтра.

Теперь можно просматривать графики характеристик импортированного фильтра, как было описано ранее. Для возврата в режим *расчета* фильтров используйте кнопку **Design Filter** (см. рис. 6.17).

## Редактирование расположения нулей и полюсов функции передачи фильтра

Вкладка **Pole\Zero Editor** появляется на экране после щелчка на одноименной кнопке (см. рис. 6.17). Она позволяет редактировать параметры рассчитанного или импортированного фильтра, представленного в виде набора нулей и полюсов (см. разд. "Нули и полюсы" главы 4). На этой вкладке, показанной на рис. 6.21, имеются следующие элементы управления:

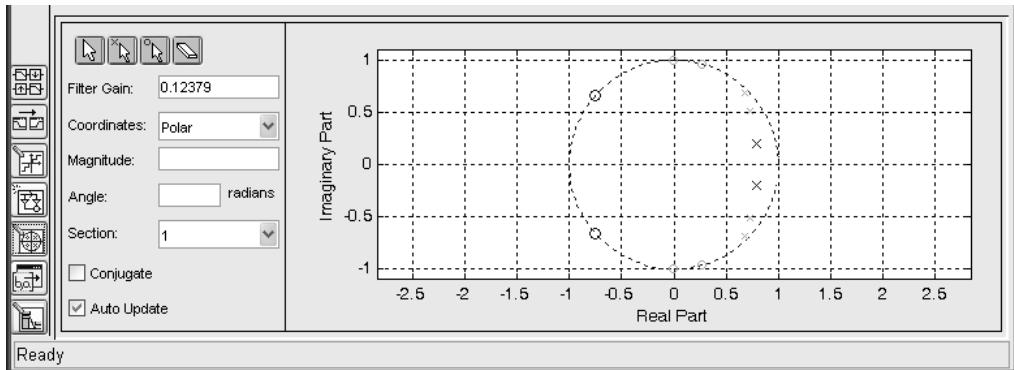


Рис. 6.21. Вкладка Pole\Zero Editor

- четыре кнопки, расположенные в левом верхнем углу вкладки, переключают режимы графического редактирования. Перечислим их слева направо:
  - **Move Pole/Zero** — режим перемещения мышью имеющихся нулей и полюсов;
  - **Add Pole** — режим добавления (щелчком мыши) новых полюсов функции передачи;
  - **Add Zero** — режим добавления (щелчком мыши) новых нулей функции передачи;
  - **Delete Pole/Zero** — режим удаления имеющихся нулей и полюсов;
- поле ввода **Filter Gain** — общий коэффициент усиления фильтра (коэффициент  $k$  в формуле (4.8) в главе 4);
- раскрывающийся список **Coordinates** определяет, в каких координатах будут отображаться положения выделенных нулей и полюсов — полярных (**Polar**) или

декартовых (**Rectangular**). В зависимости от выбранного варианта меняется название следующих двух полей ввода;

- два поля ввода — **Magnitude** и **Angle** (для полярных координат) либо **Real** и **Imaginary** (для декартовых координат) — отображают и позволяют редактировать координаты выбранного в данный момент нуля или полюса функции передачи;
- раскрывающийся список **Section** доступен, если редактируемый фильтр представляет собой каскад секций второго порядка. В нем выбирается редактируемая секция фильтра. Нули и полюсы этой секции отображаются на вкладке значками большего размера (см. рис. 6.21). Нули и полюсы остальных секций редактировать нельзя;
- установленный флажок **Conjugate** блокирует пары комплексно-сопряженных нулей и полюсов, позволяя редактировать их только совместно и с сохранением зеркального расположения относительно горизонтальной оси. Это позволяет сохранить вещественность фильтра в процессе редактирования. Если флажок снять, появится возможность создавать комплексные фильтры с несимметричными частотными характеристиками;
- при установленном флажке **Auto Update** отображаемый график выбранной характеристики фильтра непрерывно обновляется в процессе перетаскивания нуля или полюса. Если этот флажок снят, график будет перерисован только после завершения перетаскивания.

Помимо возможности редактирования нулей и полюсов перетаскиванием и прямым вводом координат, имеется еще ряд команд подменю **Edit | Pole/Zero Editor**, позволяющих выделять группы нулей и/или полюсов, выбирая их по ряду признаков, а также осуществлять геометрические преобразования над выделенными группами.

## Преобразование частотной оси

Имея синтезированный либо импортированный фильтр, в среде FDATool можно произвести его частотное преобразование — превратить ФНЧ в полосовой фильтр и т. п. Для этого служит вкладка **Frequency Transformations**, появляющаяся на экране после щелчка на кнопке **Transform Filter** (см. рис. 6.17). На этой вкладке, показанной на рис. 6.22, имеются следующие элементы управления:

- раскрывающийся список **Original filter type** позволяет указать тип АЧХ исходного фильтра. Возможны всего четыре варианта: **Lowpass** (ФНЧ), **Highpass** (ФВЧ), **Bandpass** (полосовой фильтр) и **Bandstop** (режекторный фильтр);
- раскрывающийся список **Transformed filter type** позволяет указать тип АЧХ преобразованного фильтра. Здесь вариантов больше — если исходный фильтр является ФНЧ или ФВЧ, к уже перечисленным типам добавляется **Multiband** (многополосный фильтр), а также **Bandpass (complex)**, **Bandstop (complex)** и **Multiband (complex)**, позволяющие получать фильтры соответствующих типов с АЧХ, несимметричной относительно нулевой частоты (т. е. фильтры с комплексными коэффициентами). Кроме того, если исходный фильтр является не-

рекурсивным, в данном списке появятся ФНЧ и ФВЧ с пометкой (**FIR**), это означает сохранение нерекурсивной структуры фильтра (в остальных случаях получится рекурсивный фильтр). Если же исходный фильтр был полосовым или режекторным, в списке **Transformed filter type** будет присутствовать только один вариант — совпадающий с исходным;

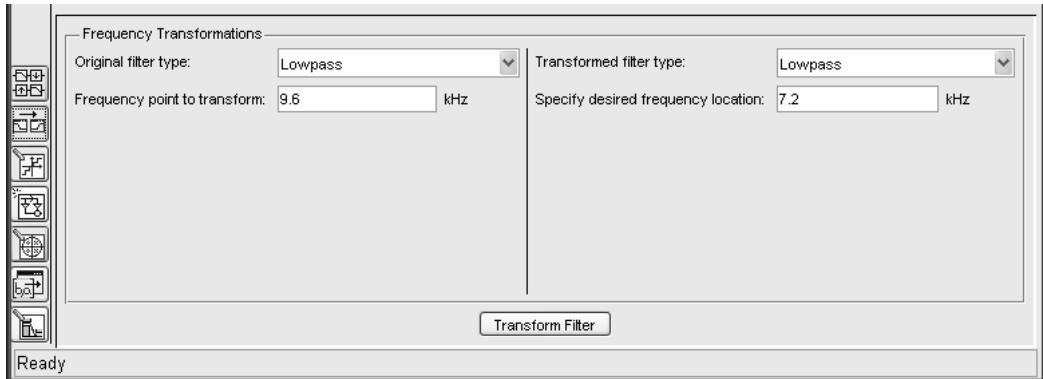


Рис. 6.22. Вкладка Frequency Transformations

- поле ввода **Frequency point to transform** задает частоту привязки — точку частотной характеристики исходного фильтра, которая после трансформации частотной оси окажется расположена на одной или нескольких частотах, задаваемых в правой части вкладки;
- поля с именами **Specify ... frequency location(s)** задают одну или несколько частот, на которых АЧХ трансформированного фильтра будет совпадать с АЧХ исходного фильтра, взятой на частоте, указанной в поле **Frequency point to transform**. Если преобразованный фильтр является ФНЧ или ФВЧ, будет присутствовать одно поле ввода **Specify desired frequency location**; в случае полосового или режекторного фильтра полей ввода будет два: **Specify desired low frequency location** и **Specify desired high frequency location** (указанные для исходного фильтра точка частотной привязки отображается в две указанных частоты); наконец, в случае многополосного фильтра будет присутствовать поле ввода **Specify a vector of desired frequency locations**, куда необходимо ввести вектор, задающий множество частот, в которое будет отображаться точка частотной привязки. Все сказанное относится к случаю, когда исходный фильтр является ФНЧ или ФВЧ; для полосовых и режекторных фильтров возможно только монотонное нелинейное масштабирование частотной оси, при котором частота привязки из поля **Frequency point to transform** оказывается на частоте, заданной в поле **Specify desired frequency location**.

Основное использование вкладки частотных преобразований — трансформация ФНЧ в фильтры других типов. В этом случае в поле **Frequency point to transform** вводится частота среза исходного фильтра, а в полях **Specify ... frequency location(s)** задаются частоты среза (одна или несколько) для трансформированного фильтра.

Идея трансформации частотной оси была изложена в главе 2 применительно к аналоговым фильтрам. Среда FDATool производит аналогичные преобразования непосредственно над дискретными фильтрами, используя для этого многочисленные функции, имеющиеся в пакете Filter Design.

## Объекты спецификаций фильтров и среда FilterBuilder

В последних версиях Signal Processing Toolbox начала развиваться новая концепция программной реализации синтеза дискретных и цифровых фильтров. Ее основой являются объекты спецификаций фильтров (класс объектов `fdesign`). Основные идеи новой концепции состоят в том, что сначала создается объект, хранящий все требования к фильтру, а затем для этого объекта вызывается метод `design` с указанием конкретного алгоритма расчета. Результатом работы этого метода является объект дискретного фильтра.

В качестве примера покажем, как реализуется синтез одного и того же фильтра "традиционными" и "новыми" средствами. Пусть нам необходимо рассчитать эллиптический ФНЧ со следующими параметрами:

- граница полосы пропускания: 0,2 от частоты Найквиста;
- граница полосы задерживания: 0,25 от частоты Найквиста;
- неравномерность АЧХ в полосе пропускания: 1 дБ;
- подавление сигнала в полосе задерживания: 40 дБ;
- фильтр должен быть представлен в виде секций второго порядка.

Описанные в этой главе ранее функции позволяют сделать это следующим образом:

```
>> [n, w0] = ellipord(0.2, 0.25, 1, 40); % определение порядка фильтра  
>> [b, a] = ellip(n, 1, 40, w0);           % расчет фильтра  
>> sos = tf2sos(b, a); % преобразование к секциям второго порядка
```

А вот как будет выглядеть синтез этого же фильтра с использованием объекта `fdesign`:

```
>> f = fdesign.lowpass('Fp,Fst,Ap,Ast', 0.2, 0.25, 1, 40);  
>> h = f.design('ellip');
```

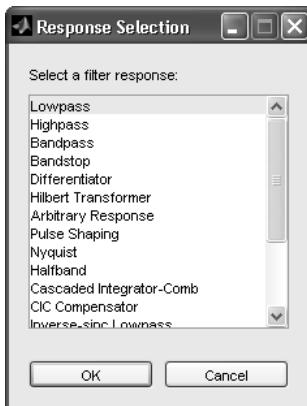
Первая строка кода создает объект спецификаций ФНЧ (конструктор `lowpass`), в параметрах конструктора указывается строка, задающая один из возможных наборов параметров, после чего перечисляются значения этих параметров. Вторая строка выполняет собственно синтез фильтра. Для рекурсивных фильтров результат всегда представлен в виде секций второго порядка.

Сравнив значения матрицы `sos` и поля `sosMatrix` объекта `h`, можно убедиться, что они отличаются только порядком следования секций.

Графической оболочкой для объектов спецификаций фильтров является программа FilterBuilder. Для ее запуска необходимо набрать имя программы в командной строке MATLAB:

```
>> filterbuilder
```

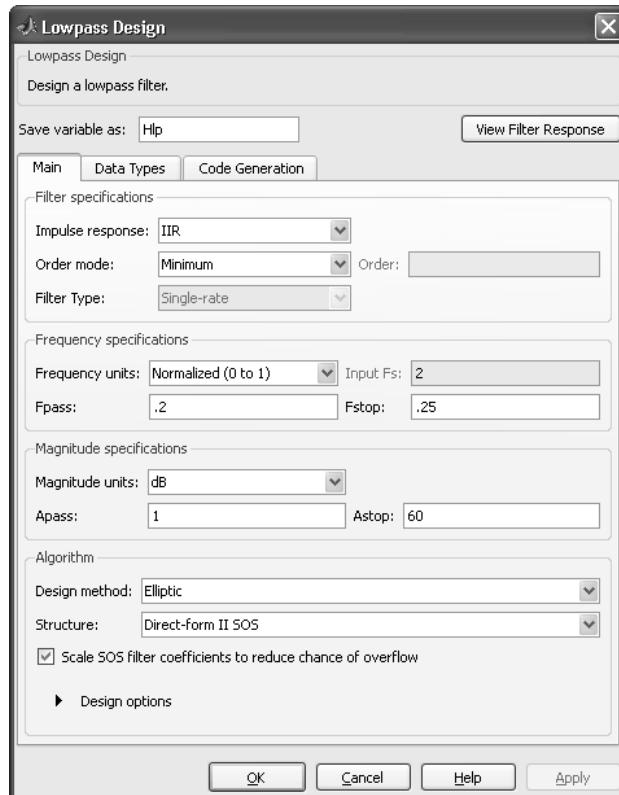
Появится окно для выбора типа частотной характеристики фильтра, показанное на рис. 6.23.



**Рис. 6.23.** Окно выбора типа фильтра в среде FilterBuilder

В этом окне необходимо выбрать категорию, к которой относится синтезируемый фильтр, и щелкнуть на кнопке **OK**. Появится основное окно среды FilterBuilder, показанное на рис. 6.24.

На вкладке **Main** этого окна задаются различные параметры, относящиеся как к спецификациям фильтра, так и к настройкам метода синтеза. Вкладка **Data Types** позволяет выбрать тип арифметики, используемой фильтром: с плавающей (форматы `double` и `single`) или с фиксированной запятой. Наконец, вкладка **Code Generation** позволяет экспорттировать M-файл, содержащий код для синтеза фильтра, создать блок или подсистему для среды моделирования Simulink, сгенерировать код на языке VHDL или Verilog.



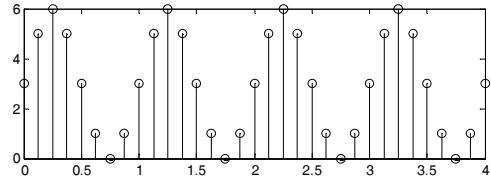
**Рис. 6.24.** Окно задания параметров фильтра низких частот в среде FilterBuilder

Задав необходимые параметры, необходимо щелкнуть на кнопке **Apply**, расположенной в правом нижнем углу окна. Будет выполнен синтез фильтра и соответствующий объект (имя переменной для него задается в поле ввода **Save variable as**) появится в рабочей области памяти MATLAB. Сообщение об этом будет выведено в командной строке:

```
The variable 'Hlp' has been exported to the command window.
```

Для просмотра характеристик синтезированного фильтра нужно щелкнуть на кнопке **View Filter Response**, расположенной в правом верхнем углу окна. Откроется окно визуализатора фильтров FVTool (см. главу 4). Единственной особенностью работы визуализатора в данном случае является то, что на график АЧХ фильтра будут наложены красные линии, демонстрирующие заданные при синтезе спецификации.

# ГЛАВА 7



## Эффекты квантования в цифровых системах

В разд. "Аналоговые, дискретные и цифровые сигналы" главы 3 мы обсуждали разницу между *дискретными* и *цифровыми* сигналами. До сих пор, строго говоря, речь шла о *дискретных* сигналах и системах, поскольку отсчеты сигналов и коэффициенты фильтров считались представленными *точно* (без погрешностей). В данной главе мы займемся именно *цифровыми* сигналами и системами, рассмотрев эффекты, возникающие вследствие конечной точности представления отсчетов сигналов и параметров систем в вычислительных устройствах.

Теоретический анализ вопросов, связанных с конечной точностью представления чисел в цифровых системах, весьма непрост. В данной главе мы лишь рассмотрим основные источники погрешностей и познакомимся со средствами MATLAB, позволяющими производить квантование сигналов и анализировать эффекты квантования в алгоритмах цифровой обработки сигналов. Читатель, которого заинтересуют более подробные теоретические построения, может обратиться, например, к фундаментальному труду [8].

Эффекты, связанные с конечной разрядностью представления чисел, можно разделить на следующие категории:

- шум квантования, возникающий при аналого-цифровом преобразовании;
- искажения характеристик, происходящие при квантовании коэффициентов цифровых фильтров;
- переполнение разрядной сетки в процессе вычислений;
- округление промежуточных результатов вычислений.

Далее мы рассмотрим эти категории подробнее. Но сначала поговорим о способах представления чисел в вычислительных устройствах.

## Форматы представления чисел

Форматы представления дробных чисел в вычислительных устройствах можно разделить на две группы: форматы с *фиксированной запятой* (*fixed point*) и форматы с *плавающей запятой* (*floating point*). В дополнение к этому существует несколько способов представления отрицательных чисел.

## Представление отрицательных чисел

Для представления знака числа обычно используется старший двоичный разряд, при этом возможно несколько способов:

- **прямой код:** 0 в старшем разряде соответствует положительным числам, 1 — отрицательным. Остальные разряды представляют модуль числа. В таком коде удобно осуществлять операции умножения (модули чисел перемножаются, а знаковые разряды складываются по модулю два), но неудобно реализовывать сложение. Кроме того, некоторые проблемы создает наличие двух представлений нуля — "положительного" и "отрицательного";
- **дополнительный код:** 0 в старшем разряде соответствует положительным числам, 1 — отрицательным. Неотрицательные числа представляются без каких-либо особенностей, а для превращения положительного числа в равное ему по модулю отрицательное необходимо инвертировать все разряды двоичного представления (включая знаковый) и к получившемуся двоичному числу прибавить единицу. В дополнительном коде удобно выполнять операции сложения (числа со знаком складываются точно так же, как беззнаковые);
- **обратный код:** 0 в старшем разряде соответствует положительным числам, 1 — отрицательным. Неотрицательные числа представляются без каких-либо особенностей, а превращение положительного числа в равное ему по модулю отрицательное производится путем простой инверсии всех разрядов двоичного представления (включая знаковый);
- **смещенный код:** трактовка знакового разряда здесь противоположна предыдущим вариантам — 1 означает положительное число, а 0 — отрицательное. Представления чисел получаются путем прибавления к ним константы  $2^N - 1$ , где  $N$  — число двоичных разрядов (не считая знакового).

В табл. 7.1 описанные варианты поясняются на примере 8-разрядных целых чисел.

Таблица 7.1. Способы представления чисел со знаком

Беззнаковое значение	Двоичное представление	Значение со знаком			
		Прямой код	Дополнительный код	Обратный код	Смещенный код
0	00000000	0	0	0	-127
1	00000001	1	1	1	-126
...	...	...	...	...	...
126	01111110	126	126	126	-1
127	01111111	127	127	127	0
128	10000000	-0	-128	-127	1
129	10000001	-1	-127	-126	2
...	...	...	...	...	...

Таблица 7.1 (окончание)

Беззнаковое значение	Двоичное представление	Значение со знаком			
		Прямой код	Дополнительный код	Обратный код	Смещенный код
254	11111110	-126	-2	-1	127
255	11111111	-127	-1	-0	128

Наибольшее распространение для представления целых чисел и чисел с фиксированной запятой получил дополнительный код. Прямой код используется для представления мантиссы, а смещенный код — для представления порядка чисел в формате с плавающей запятой (см. далее).

## Формат с фиксированной запятой

Название "формат с фиксированной запятой" означает, что в двоичном представлении дробного числа для хранения его целой и дробной частей отведено фиксированное число разрядов. Иными словами, запятая, разделяющая целую и дробную части в двоичном представлении числа, находится на фиксированном месте.

Часто формат с фиксированной запятой обозначают парой целых чисел:  $M.N$ . В большинстве случаев (об одном из исключений будет сказано далее)  $M$  обозначает число разрядов целой части числа (включая знак), а  $N$  — число разрядов дробной части. Этот способ обозначения широко распространен в литературе по цифровой обработке сигналов.

В качестве примера рассмотрим формат 1.15, часто применяемый для представления чисел в цифровых сигнальных процессорах. Как следует из обозначения, целая часть числа содержит только один разряд — знаковый. Поэтому числа, которые можно представить в этом формате, по модулю не превосходят единицы. Пятнадцать разрядов после запятой обеспечивают дискретность представления, равную  $2^{-15} = 1/32768 \approx 3 \cdot 10^{-5}$ . Для представления отрицательных чисел используется дополнительный код (см. ранее разд. "Представление отрицательных чисел" этой главы). Более детально данный формат иллюстрируется в табл. 7.2.

Приведенная таблица наглядно демонстрирует два важных факта, характеризующих форматы с фиксированной запятой. Во-первых, сравнение столбцов "Целое число в дополнительном коде" и "Число в формате 1.15" показывает, что формат с фиксированной запятой по сути дела означает просто договоренность делить целые числа на постоянный коэффициент, равный  $2^N$  ( $N$  — число разрядов дробной части). В нашем примере этот коэффициент равен 32 768.

Во-вторых, из таблицы видно, что значение  $-1$  в данном формате представить можно, а  $+1$  — нет (максимальное положительное число равно  $\frac{32767}{32768} \approx 0,9999695$ ). Это общее свойство форматов с фиксированной запятой — максимальное по модулю

отрицательное число равно  $-2^{M-1}$ , а максимальное положительное "не дотягивает" до  $2^{M-1}$  на  $2^{-N}$ .

Таблица 7.2. Формат с фиксированной запятой 1.15

Двоичное представление	Шестнадцатеричное представление	Целое число в дополнительном коде	Число в формате 1.15
0000 0000 0000 0000	0000	0	0
0000 0000 0000 0001	0001	1	$\frac{1}{32768}$
0000 0000 0000 0010	0002	2	$\frac{2}{32768}$
...	...	...	...
0111 1111 1111 1110	7FFE	32 766	$\frac{32766}{32768}$
0111 1111 1111 1111	7FFF	32 767	$\frac{32767}{32768}$
1000 0000 0000 0000	8000	-32 768	-1
1000 0000 0000 0001	8001	-32 767	$-\frac{32767}{32768}$
...	...	...	...
1111 1111 1111 1110	FFFE	-2	$-\frac{2}{32768}$
1111 1111 1111 1111	FFFF	-1	$-\frac{1}{32768}$

### ВНИМАНИЕ!

В языках программирования высокого уровня часто применяется другая трактовка первого из пары чисел  $M.N$ , обозначающих формат с фиксированной запятой —  $M$  обозначает не число разрядов целой части, а общее число двоичных разрядов, используемое для представления чисел. Такой подход используется и в MATLAB. В частности, в объектах-квантователях, создаваемых средствами пакета Fixed-Point (о них речь пойдет далее в этой главе), рассмотренный формат 1.15 задается с помощью двухэлементного вектора [16 15].

Достоинствами формата с фиксированной запятой являются равномерность квантования и простота реализации арифметических операций; главный недостаток — ограниченный динамический диапазон. *Динамическим диапазоном* называют от-

ношение между самым большим и самым малым по модулю (но отличным от нуля) числами, которые можно представить с помощью данного формата. Для формата с фиксированной запятой это отношение равно  $2^{M-1}$ .

## Формат с плавающей запятой

Числа в формате с плавающей запятой (иногда его называют еще *экспоненциальным* форматом или *научной нотацией*) представляются в виде

$$x = f \cdot 2^e,$$

где  $f$  — *мантиssa (mantissa)*, а  $e$  — *порядок (exponent)*. Обычно мантисса представляется в формате с фиксированной запятой, а порядок является целым числом.

Поскольку в экспоненциальной форме записи числа присутствуют два параметра, такое представление оказывается неоднозначным. Приведем простой пример:

$$2 = 2 \cdot 2^0 = 1 \cdot 2^1 = \frac{1}{2} \cdot 2^2 = \dots$$

Чтобы устранить эту неоднозначность, принято ограничивать диапазон допустимых значений мантиссы, например, или так:  $0,5 \leq |f| < 1$ , или так:  $1 \leq |f| < 2$  (этот вариант используется наиболее часто). Процедура приведения мантиссы к допустимому диапазону называется *нормализацией*, а экспоненциальная запись числа, удовлетворяющая указанным ограничениям, — *нормализованной экспоненциальной формой*.

Нормализация мантиссы позволяет сэкономить один разряд в ее двоичном представлении. Действительно, если нормализованная мантисса лежит в диапазоне  $[0,5, 1[$ , то ее двоичная запись имеет вид "0,1...", а если для нормализации используется диапазон  $[1, 2[$ , двоичная запись будет выглядеть как "1,...". В обоих случаях заранее известно, что первый значащий разряд равен единице, поэтому его можно не хранить. Иногда это называется использованием *неявного старшего бита*.

Для представления отрицательных чисел необходимо обеспечить возможность хранения отрицательных значений мантиссы. Как правило, для этого используется *прямой код* (см. ранее разд. "Представление отрицательных чисел" этой главы), т. е. модуль мантиссы и знаковый бит хранятся независимо.

Использование строго нормализованной мантиссы делает невозможным представление нулевого значения в формате с плавающей запятой. Поэтому используется специальное соглашение о том, что число, содержащее нули во всех разрядах мантиссы и порядка, считается нулем. Знаковый разряд при этом может иметь любое значение. Таким образом, имеется два представления нуля — "положительный нуль" и "отрицательный нуль". Об этом уже говорилось ранее в разд. "Представление отрицательных чисел" этой главы.

Чтобы использование данного соглашения для представления нуля не привело к возникновению "дырки" в наборе представимых чисел, нулевое представление порядка должно соответствовать не нулевому, а минимально возможному (т. е. отри-

цательному) его значению. Поэтому для представления порядка чисел с плавающей запятой используется *смещенный код* (см. ранее разд. "Представление отрицательных чисел" этой главы).

### **ЗАМЕЧАНИЕ**

Если бы порядок хранился, например, в дополнительном коде, число, имеющее нули во всех разрядах мантиссы и порядка, формально равнялось бы единице, и замена его на нуль привела бы к невозможности представления единичного значения, что явно неприемлемо.

Итак, для представления нулевого значения в формате с плавающей запятой мантисса неминуемо должна быть *денормализованной*. Однако если использовать денормализованную мантиссу только для представления нуля, малые по модулю числа, представимые в данном формате, оказываются расположены неравномерно — вокруг нуля появляется "мертвая зона", размер которой существенно превышает расстояние между ближайшими к нулю представимыми числами.

Чтобы решить эту проблему и сделать расположение представимых чисел вблизи нуля равномерным, применяют следующее *соглашение о денормализации*: если все разряды порядка имеют нулевые значения, то величина порядка увеличивается на единицу, а мантисса считается *денормализованной*, т. е. содержащей в неявном старшем бите *ноль*, а не единицу. Это позволяет расширить возможности представления малых по модулю чисел. Фактически в данном случае мы увеличиваем диапазон возможных отрицательных порядков за счет сокращения числа значащих цифр мантиссы.

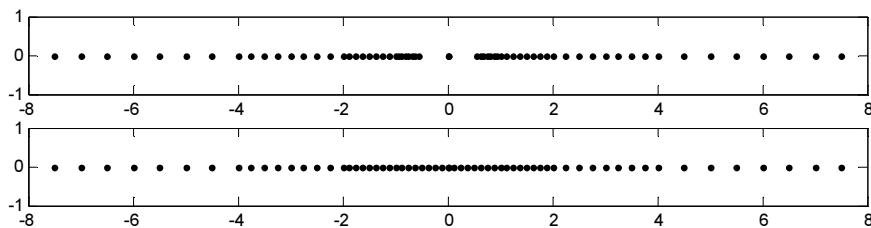
Поясним сказанное на примере, рассмотрев 6-битовый формат с плавающей запятой, в котором отведено 2 бита для хранения порядка и 3 — для хранения мантиссы. Биты двоичного представления числа, таким образом, имеют следующее назначение:

seemmm

Здесь *s* — знаковый бит, *e* — биты порядка, *m* — биты мантиссы. Рассчитаем и покажем на графике (рис. 7.1) все числа, которые можно представить в данном формате без использования и с использованием денормализации мантиссы для малых чисел. Соответствующие вычисления можно осуществить с помощью следующей MATLAB-программы:

```
% первый вариант — денормализация только для нуля
for k = 0:63          % перебор всех возможных представлений
    s = 1 - 2 * bitget(k, 6);      % знаковый множитель
    e = bi2de(bitget(k, 4:5))-1;  % порядок
    m = 1 + bitand(k, 7) / 8;     % мантисса
    if ~bitand(k, 31), f(k+1)=0;   % нуль — особый случай
    else f(k+1)=s*m*(2^e);
    end
end
subplot(2, 1, 1)
plot(f, f*0, '.')
```

```
% второй вариант — денормализация для малых по модулю чисел
for k = 0:63 % перебор всех возможных представлений
    s = 1 - 2 * bitget(k, 6); % знаковый множитель
    e = bi2de(bitget(k, 4:5)); % смещенный порядок
    if e % обычный случай — нормализуем мантиссу
        e = e-1; % истинный порядок
        m = 1 + bitand(k, 7) / 8; % нормализованная мантисса
    else % нули в поле порядка — не нормализуем мантиссу
        m = bitand(k, 7) / 8; % денормализованная мантисса
    end
    f(k+1)=s*m*(2^e);
end
subplot(2, 1, 2)
plot(f, f*0, '.')
```



**Рис. 7.1.** Числа, представимые в 6-битовом формате с плавающей запятой без использования (сверху) и с использованием (снизу) соглашения о денормализации мантиссы для малых чисел

Сравнение двух графиков рис. 7.1 наглядно демонстрирует, как денормализация мантиссы влияет на возможности представления малых по модулю чисел.

### ЗАМЕЧАНИЕ

В приведенном листинге использованы следующие функции MATLAB: `bitget` — возвращает биты с заданными номерами из двоичного представления целого неотрицательного числа; `bitand` — выполняет побитовую логическую операцию "И" над неотрицательными целыми числами; `bi2de` — преобразует вектор, состоящий из нулей и единиц, трактуя его как запись числа в двоичной системе счисления.

В заключение данного раздела приведем сведения о форматах с плавающей запятой, поддерживаемых процессорами персональных компьютеров:

- формат с *одинарной* точностью (`single` в языке Pascal, `float` в языке C) — 32 бита, включающие знаковый разряд, 8 бит порядка и 23 бита мантиссы;
- формат с *двойной* точностью (`double` в языках Pascal и C) — 64 бита, включающие знаковый разряд, 11 бит порядка и 52 бита мантиссы. Это основной формат хранения данных в MATLAB;
- формат с *расширенной* точностью (`extended` в языке Pascal, `long double` в языке C) — 80 бит, включающих знаковый разряд, 15 бит порядка и 64 бита ман-

тиссы. В отличие от двух предыдущих форматов, здесь не используется неявный старший бит мантиссы.

Формат с плавающей запятой за счет экспоненциального представления чисел обладает существенно большим динамическим диапазоном, чем формат с фиксированной запятой при той же длине слова. Однако платой за это является неравномерность квантования и повышенная сложность реализации арифметических операций.

## Процесс квантования

Как уже отмечалось в главе 3, квантованием называется процесс преобразования истинных значений отсчетов сигнала в двоичные числа, имеющие конечное число разрядов. Там же было введено понятие *шума квантования*. В данном разделе мы обсудим шум квантования несколько подробнее, а также рассмотрим идею *неравномерного квантования*.

## Шум квантования

Как было отмечено в разд. "Аналоговые, дискретные и цифровые сигналы" главы 3, при представлении отсчетов дискретного сигнала в виде чисел с ограниченной разрядностью происходит их округление. Разность между исходным и округленным значениями называется *шумом квантования*.

Анализ вопросов, связанных с шумами квантования и ошибками округления в цифровых системах обработки сигналов, весьма сложен (см., например, [8]). В данном разделе будет представлено лишь несколько положений общего характера.

В качестве иллюстрации процесса квантования на рис. 7.2 показаны (без дискретизации по времени) гармонический сигнал  $s(t)$ , результат его квантования  $s_k(t)$  и возникающий при этом шум  $e(t) = s(t) - s_k(t)$ . Очевидно, что значения шума квантования лежат в следующих пределах:

$$-\frac{\Delta}{2} \leq e(t) \leq \frac{\Delta}{2},$$

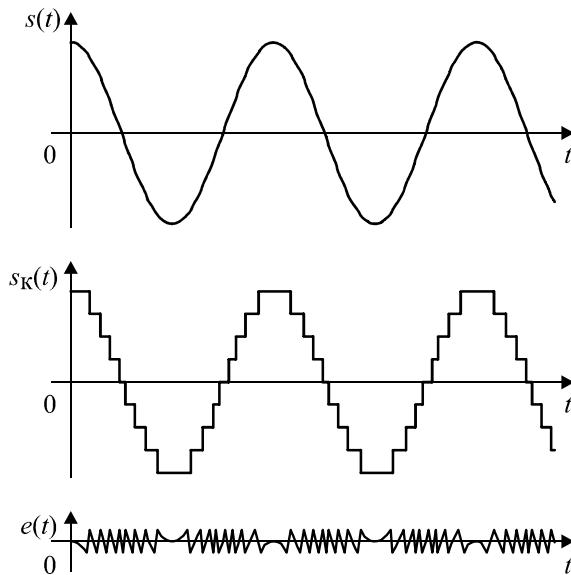
где  $\Delta$  — расстояние между соседними уровнями квантования, т. е. разность между ближайшими возможными значениями квантованного сигнала.

В большинстве случаев можно считать  $e(t)$  случайным процессом, имеющим равномерное распределение вероятности в указанных пределах. Такой случайный процесс имеет нулевое среднее значение и дисперсию, равную  $\Delta^2/12$  (см. разд. "Равномерное распределение" главы 1).

### ЗАМЕЧАНИЕ

На рис. 7.2 предполагалось, что при квантовании производится *округление* значений уровня сигнала. В реальных АЦП вместо этого может использоваться *усечение*, т. е. округление в сторону меньшего значения. В этом случае шум квантования лежит

в диапазоне  $0 \dots \Delta$ , его среднее значение равно  $\Delta/2$ , а дисперсия, как и в случае округления, составляет  $\Delta^2/12$ .



**Рис. 7.2.** Процесс квантования гармонического сигнала

После дискретизации шум квантования представляет собой последовательность чисел  $e(kT)$ , образующую *дискретный случайный процесс* (см. разд. "Дискретные случайные сигналы" главы 3). Во многих случаях отсчеты этой последовательности можно считать некоррелированными друг с другом и статистически независимыми от квантуемого сигнала. Таким образом, шум квантования моделируется в виде аддитивного белого шума (см. разд. "Дискретный белый шум" главы 3) с равномерным распределением.

### ЗАМЕЧАНИЕ

Указанные предположения о свойствах шума квантования близки к реальности, если шаг уровней квантования  $\Delta$  достаточно мал — настолько, что квантованные значения соседних отсчетов сигнала в большинстве случаев различаются на *несколько* уровней квантования. Если же сигнал надолго "зависает" между двумя уровнями, отсчеты шума квантования перестают быть некоррелированными, теряется при этом и их статистическая независимость от сигнала.

В качестве примера рассчитаем отношение сигнал/шум (*signal-to-noise ratio, SNR*) при квантовании гармонического сигнала. Пусть квантованию подвергается гармонический сигнал с амплитудой  $A$ . Определим отношение сигнал/шум, разделив эту амплитуду на среднеквадратическое значение шума квантования:

$$\text{С/Ш} = \frac{A}{\sqrt{\Delta^2/12}} = \frac{2A\sqrt{3}}{\Delta} = N\sqrt{3},$$

где  $N = 2A/\Delta$  — число уровней квантования, укладывающихся в размахе сигнала.

АЦП, имеющий  $q$  двоичных разрядов, обеспечивает  $N = 2^q$  уровней квантования. Если размах сигнала соответствует полному рабочему диапазону АЦП, то отношение сигнал/шум равно

$$\text{С/Ш} = 2^q \sqrt{3} .$$

Если выразить этот результат в децибелах, получится простая формула, показывающая связь между числом двоичных разрядов, используемых для представления отсчетов сигналов, и максимально достижимым в этом случае отношением сигнал/шум:

$$\text{С/Ш}_{\text{дБ}} = 20 \lg(2^q \sqrt{3}) = 20q \lg 2 + 10 \lg 3 \approx 6q + 4,77 \text{ дБ} .$$

## Неравномерное квантование

Равномерное квантование, о котором шла речь до сих пор, гарантирует, что размах шума квантования не будет превосходить величины шага квантования (за исключением тех случаев, когда значение входного сигнала выходит за допустимые пределы). Однако если потребовать минимизации среднеквадратического значения шума квантования, оптимальный набор уровней квантования будет зависеть от статистических свойств сигнала, а именно от плотности вероятности его мгновенных значений.

В этом случае интуитивно ясно, что уровни квантования должны располагаться плотнее друг к другу в областях тех значений, которые сигнал принимает с большей вероятностью.

Идея неравномерного квантования в общем случае формулируется следующим образом: диапазон возможных значений сигнала делится на  $N$  зон квантования  $a_0 \dots a_1$ ,  $a_1 \dots a_2$ , ...,  $a_{N-1} \dots a_N$  (часто можно считать  $a_0 = -\infty$  и  $a_N = +\infty$ ). Зонам квантования сопоставлены квантованные значения  $b_k \in [a_{k-1}, a_k]$ . Если входной сигнал попадает в диапазон  $a_{k-1} \dots a_k$ , его квантованное значение принимается равным  $b_k$ .

Итак, пусть сигнал имеет плотность вероятности  $p(x)$  и мы хотим осуществить его  $N$ -уровневое квантование так, чтобы минимизировать средний квадрат шума квантования. Этот средний квадрат рассчитывается как

$$\overline{e^2} = \sum_{k=1}^N \int_{a_{k-1}}^{a_k} (x - b_k)^2 p(x) dx = \overline{x^2} - 2 \sum_{k=1}^N b_k \int_{a_{k-1}}^{a_k} x p(x) dx + \sum_{k=1}^N b_k^2 \int_{a_{k-1}}^{a_k} p(x) dx ,$$

где  $\overline{x^2}$  — средний квадрат сигнала  $x$ . Приравнивание к нулю частных производных этого выражения по  $a_k$  и  $b_k$  дает следующие соотношения для оптимальных параметров квантования:

$$b_k = \frac{\int_{a_{k-1}}^{a_k} xp(x)dx}{\int_{a_{k-1}}^{a_k} p(x)dx}, \quad a_k = \frac{b_{k-1} + b_k}{2}. \quad (7.1)$$

Данные формулы при известной плотности вероятности  $p(x)$  дают систему нелинейных уравнений относительно  $a_k$  и  $b_k$ . Аналитическое решение этой системы даже для несложных функций  $p(x)$  оказывается весьма непростым, и его в большинстве случаев приходится искать численными методами.

### **ЗАМЕЧАНИЕ**

При выполнении условий (7.1) автоматически обеспечивается нулевое среднее значение шума квантования.

Если формула для плотности вероятности сигнала неизвестна, но имеется "типичный" набор его отсчетов, можно произвести оптимизацию параметров квантования по этому тестовому набору. Поиск оптимальных значений  $a_k$  и  $b_k$  в этом случае производится численным итерационным методом (см. далее в этой главе описание функций quantiz и lloyds).

Неравномерное квантование применяется, например, в современных цифровых телефонных сетях. Малые значения речевого сигнала более вероятны, чем большие, поэтому используется нелинейное преобразование сигнала, когда диапазон значений, при равномерном квантовании представляемый 12 двоичными разрядами (4096 уровней), преобразуется в 256 (8 двоичных разрядов) *неравномерно* расположенных уровней согласно Рекомендации Международного союза электросвязи (ITU-T) G.711. Зависимость уровня квантования от его номера представляет собой кусочно-линейную аппроксимацию экспоненциального закона. В цифровых каналах связи передаются 8-разрядные номера уровней квантования, а при цифроаналоговом преобразовании они конвертируются в 12-разрядные значения соответствующих им уровней сигнала.

## **Эффекты квантования в цифровых фильтрах**

Шум квантования — не единственная проблема, связанная с конечной разрядностью используемых чисел. Так, неизбежное округление разнообразных коэффициентов, используемых в алгоритмах цифровой обработки сигналов, приводит к тому, что параметры фильтров и других устройств отличаются от желаемых, причем возможны ситуации, когда эти отличия весьма существенны. Кроме того, из-за округления промежуточных результатов может происходить накопление вычислительных погрешностей, также искажающих конечный результат. Эти эффекты будут рассмотрены в данном разделе.

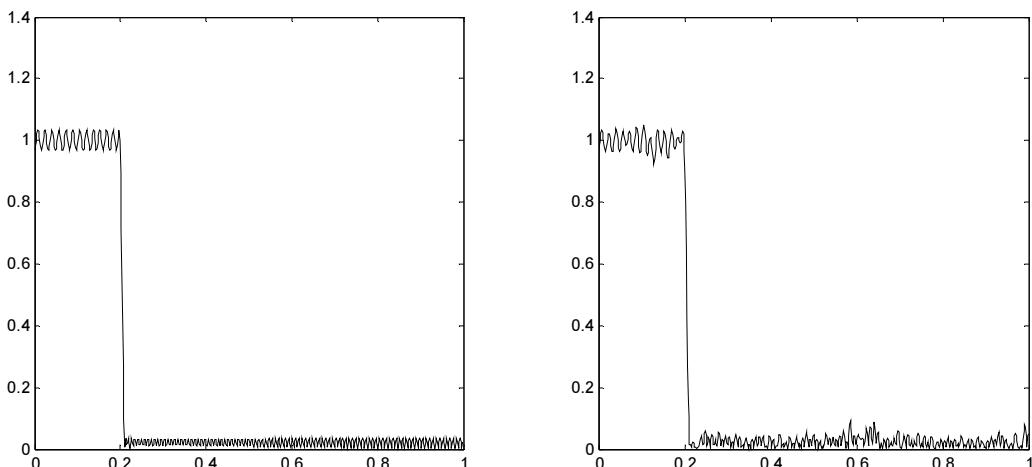
## **Квантование коэффициентов цифровых фильтров**

До сих пор, рассматривая характеристики дискретных фильтров, мы получали коэффициенты фильтров некоторыми расчетными методами и считали, что они пред-

ставлены точно. Однако при практической реализации фильтров почти неизбежно возникает необходимость округления их коэффициентов. При использовании цифровых сигнальных процессоров это связано с поддерживаемыми ими форматами представления чисел, при создании программ обработки сигналов для персональных компьютеров — со стремлением повысить быстродействие.

Из-за округления коэффициентов характеристики фильтра претерпевают искажения, величина которых зависит не только от погрешности представления коэффициентов, но и от исходных параметров фильтра и формы его построения (см. разд. "Формы реализации дискретных фильтров" главы 4).

В нерекурсивных фильтрах коэффициенты равны отсчетам импульсной характеристики и линейно связаны с комплексным коэффициентом передачи. Поэтому малые искажения коэффициентов приводят к малым искажениям частотных характеристик и проблемы, связанные с округлением коэффициентов, проявляются редко. Однако, если фильтр должен иметь очень крутой спад АЧХ между полосами пропускания и задерживания, округление коэффициентов все же может привести к заметным искажениям частотных характеристик.



**Рис. 7.3.** АЧХ нерекурсивного ФНЧ до (слева) и после (справа) округления коэффициентов

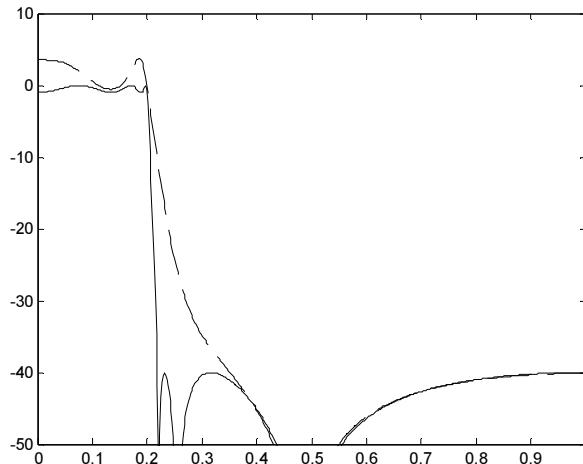
Убедимся в этом на простом примере. Синтезируем методом Ремеза ФНЧ 256-го порядка с полосой пропускания, простирающейся до 0,2, и полосой задерживания, начинающейся от 0,21 (указаны частоты, нормированные к частоте Найквиста). Затем округлим коэффициенты фильтра с точностью до 1/256, оставив в них 8 двоичных разрядов после запятой, и построим графики АЧХ до и после округления (рис. 7.3):

```
>> b = firpm(256, [0 0.2 0.21 1], [1 1 0 0]);
>> bq = round(b*256)/256; % округление
>> [h, f] = freqz(b);
```

```
>> hq = freqz(bq);
>> subplot(1, 2, 1)
>> plot(f/pi, abs(h))
>> subplot(1, 2, 2)
>> plot(f/pi, abs(hq))
```

Как видите, ничего особенно страшного не произошло — лишь увеличился размах пульсаций АЧХ. Естественно, в любом случае следует обязательно проконтролировать параметры фильтра после округления коэффициентов, чтобы проверить, удовлетворяет ли квантованный фильтр предъявляемым к нему требованиям.

Значительно серьезнее сказывается округление коэффициентов на характеристиках рекурсивных фильтров, поскольку коэффициенты знаменателя функции передачи связаны с импульсной и частотными характеристиками *нелинейно*. Как правило, наибольшие искажения происходят в тех случаях, когда АЧХ фильтра имеет крутое скаты в переходных зонах между полосами пропускания и задерживания.



**Рис. 7.4.** АЧХ рекурсивного фильтра до (сплошная линия) и после (пунктирная линия) округления коэффициентов

Приведем пример, иллюстрирующий сказанное. Рассчитаем эллиптический ФНЧ 6-го порядка, имеющий частоту среза, составляющую 0,2 от частоты Найквиста, пульсации в полосе пропускания, равные 1 дБ, и уровень пульсаций в полосе задерживания, равный -40 дБ. Напомним, что из всех фильтров, синтезируемых по аналоговым прототипам, именно эллиптические фильтры дают максимальную крутизну спада АЧХ при переходе от полосы пропускания к полосе задерживания. Далее округляем коэффициенты фильтра точно так же, как это делалось в предыдущем примере, и строим графики АЧХ фильтра (в децибелах) до и после округления (рис. 7.4):

```
>> [b, a] = ellip(6, 1, 40, 0.2); % исходный фильтр
>> bq = round(b*256)/256;           % округление числителя
>> aq = round(a*256)/256;           % округление знаменателя
```

```

>> [h, f] = freqz(b, a);           % ЧХ исходного фильтра
>> hq = freqz(bq, aq);          % ЧХ округленного фильтра
>> plot(f/pi, 20*log10(abs(h)), f/pi, 20*log10(abs(hq)), '--')
>> ylim([-50 10])

```

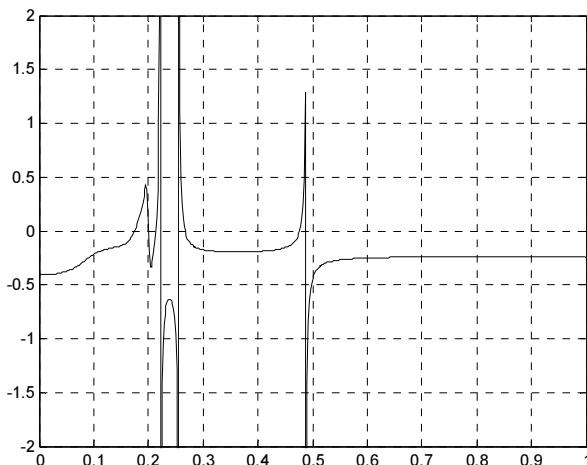
В данном случае АЧХ изменилась просто катастрофически: размах пульсаций коэффициента передачи в полосе пропускания вместо 1 дБ составляет 4 дБ, скат АЧХ стал заметно более пологим, в полосе задерживания исчезли две точки с нулевым коэффициентом передачи.

Теперь посмотрим, что изменится, если мы представим исходный фильтр по-другому — в виде трех последовательно включенных секций второго порядка. В данном случае разница между АЧХ исходного и квантованного фильтра мала и визуально плохо проявляется. Поэтому вместо графиков самих АЧХ построим график их *отношения* в децибелах (рис. 7.5):

```

>> [sos, g] = tf2sos(b, a);      % секции 2-го порядка
>> sos_q = round(sos*256) / 256; % округление коэффициентов
>> g_q = round(g*256) / 256;
>> [b_sos_q, a_sos_q] = sos2tf(sos_q, g_q);
>> [h, f] = freqz(b, a);
>> h_sos_q = freqz(b_sos_q, a_sos_q);
>> plot(f/pi, 20*log10(abs(h_sos_q) ./abs(h)));
>> ylim([-2 2]);
>> grid on

```



**Рис. 7.5.** Отношение АЧХ (в децибелах), полученных до и после округления коэффициентов рекурсивного фильтра, представленного в виде секций второго порядка

Рисунок 7.5 показывает, что разница между АЧХ в полосе пропускания не превосходит 0,5 дБ. Большие относительные расхождения наблюдаются лишь в окрестностях нулей функции передачи, где это не имеет большого значения. Таким образом, при представлении фильтра в виде последовательно включенных секций второго

порядка округление коэффициентов влияет на характеристики фильтра значительно слабее, чем при прямой форме реализации.

Данный пример демонстрирует общую закономерность, впервые установленную Кайзером [8]: параметры любого фильтра с резким изменением частотной характеристики в переходной полосе, реализованного в прямой форме, *крайне* чувствительны к значениям коэффициентов фильтра.

Существует два основных подхода к анализу и синтезу фильтров с квантованными коэффициентами. Можно рассматривать погрешности представления коэффициентов как случайные величины и, задав статистическое описание этих погрешностей, оценить среднеквадратическое отклонение частотной характеристики от исходной. Второй подход заключается в использовании прямой оптимизации квантованных коэффициентов с целью достижения минимального (в смысле среднеквадратической ошибки или максимального модуля разности) отклонения АЧХ от заданной. Более подробную информацию об этом можно найти в [8].

## Масштабирование коэффициентов цифровых фильтров

При разработке систем, работающих в реальном масштабе времени, для ускорения вычислений часто используется формат с фиксированной запятой. В этом случае может оказаться, что значения некоторых коэффициентов фильтров выходят за пределы диапазона, представимого в выбранном формате. Для решения данной проблемы прибегают к *масштабированию* фильтров. Наиболее часто коэффициенты приводят к диапазону  $[-1, 1]$ .

Рассмотрим сущность масштабирования на несложном примере. Рассчитаем эллиптический ФНЧ 4-го порядка с частотой среза, равной 20% от частоты Найквиста, пульсациями в полосе пропускания 1 дБ и подавлением сигнала в полосе задерживания 60 дБ:

```
>> [b, a] = ellip(4, 1, 60, 0.2)
b =
    0.0059    0.0053    0.0096    0.0053    0.0059
a =
    1.0000   -3.0477    3.8240   -2.2926    0.5523
```

Как видите, максимальное (по модулю) значение коэффициентов фильтра составляет 3,824. Запишем для данного фильтра алгоритм фильтрации (см. формулу (4.1) в разд. "Сущность линейной дискретной обработки" главы 4), а затем разделим и умножим правую часть формулы на 4 (в качестве коэффициента масштабирования удобно выбирать степень двойки):

$$\begin{aligned} y(k) &= 0,0059x(k) + 0,0053x(k-1) + 0,0096x(k-2) + 0,0053x(k-3) + 0,0059x(k-4) + \\ &\quad + 3,0477y(k-1) - 3,8240y(k-2) + 2,2926y(k-3) - 0,5523y(k-4) = \\ &= 4(0,0015x(k) + 0,0013x(k-1) + 0,0024x(k-2) + 0,0013x(k-3) + 0,0015x(k-4) + \\ &\quad + 0,7619y(k-1) - 0,9560y(k-2) + 0,5731y(k-3) - 0,1381y(k-4)). \end{aligned}$$

В полученной формуле алгоритма фильтрации значения коэффициентов по модулю не превышают единицы, а для сохранения результата неизменным понадобилось умножение выходного сигнала на коэффициент масштабирования. Исходный и масштабированный варианты структурной схемы показаны на рис. 7.6.

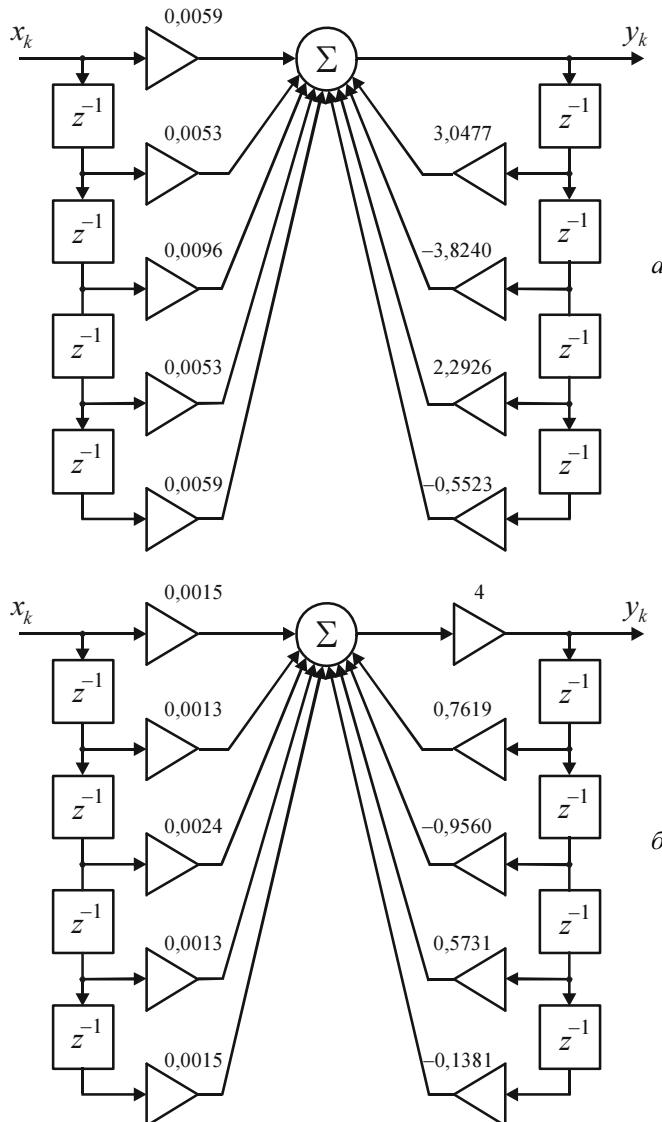


Рис. 7.6. Фильтр 4-го порядка (а) подвергается масштабированию с коэффициентом 4 (б)

Итак, при масштабировании цифрового фильтра все его коэффициенты *делятся* на одну и ту же константу, и на нее же *умножается* рассчитанный выходной сигнал. В качестве масштабирующего множителя удобно выбирать степень двойки, по-

скольку умножение на степень двойки в формате с фиксированной запятой сводится к поразрядному сдвигу двоичного представления числа влево.

Если фильтр высокого порядка реализуется "целиком" (без разделения его на последовательно включенные секции), диапазон абсолютных значений коэффициентов фильтра может быть весьма широк. При представлении коэффициентов в формате с фиксированной запятой оказывается, что самые маленькие (по модулю) коэффициенты могут из-за округления потерять несколько значащих цифр. Это приводит к искажению характеристик фильтра, порой весьма существенному, особенно в рекурсивных фильтрах. Данный эффект мы уже обсуждали в предыдущем разделе. Там же мы видели, что реализация фильтра в виде последовательно включенных секций второго порядка значительно ослабляет эти искажения. Это происходит из-за того, что в каждой отдельно взятой секции второго порядка разброс абсолютных значений коэффициентов оказывается значительно меньше, чем у фильтра в целом, особенно если при разбиении на секции группировать ближайшие друг к другу пары нулей и полюсов.

## Переполнение разрядной сетки в процессе вычислений

В процессе вычисления выходного сигнала фильтра производится множество операций умножения и сложения. При этом промежуточные результаты вычислений могут значительно превосходить окончательное значение и вызвать переполнение разрядной сетки вычислительного устройства.

В качестве примера посмотрим, каких значений достигают промежуточные результаты вычислений при обработке синусоидального сигнала фильтром низких частот Баттерворт 4-го порядка, реализованным в прямой форме (см. рис. 4.11). Чтобы получить доступ к промежуточным результатам вычислений, нам придется вместо использования функции `filter` реализовать фильтр вручную. Приведенная далее MATLAB-программа рассчитывает фильтр методом билинейного  $z$ -преобразования, а затем пропускает через него синусоидальный сигнал, сохраняя при этом в массиве `state` всю историю работы единственного сумматора, имеющегося в данной схеме фильтра:

```

Fs = 8e3; % частота дискретизации
t = 0:1/Fs:0.01; % дискретное время
F0 = 1e3; % частота сигнала
s = sin(2*pi*F0*t); % входной сигнал
N = 4; % порядок фильтра
[b, a] = butter(N, F0/Fs*2); % частота среза равна частоте сигнала
dx = zeros(1, N+1); % линия задержки для входного сигнала
dy = zeros(1, N+1); % линия задержки для выходного сигнала
m = 1;
state = zeros(1, 2*N*length(s));
y = zeros(1, length(t));
for k = 1:length(t) % цикл по отсчетам входного сигнала
    y(k) = 0; % начальное состояние сумматора
    dx = [s(k) dx(1:N)]; % новое состояние линии задержки

```

```

for n = 1:N+1           % цикл по нерекурсивной ветви
    y(k) = y(k) + b(n) * dx(n);
    state(m) = y(k);      % текущее состояние сумматора
    m = m + 1;
end
for n = 2:N+1           % цикл по рекурсивной ветви
    y(k) = y(k) - a(n) * dy(n);
    state(m) = y(k);      % текущее состояние сумматора
    m = m + 1;
end
dy(2:N+1) = [y(k) dy(2:N)]; % новое состояние линии задержки
end
plot(t, s, t, y, '--')    % входной и выходной сигналы
figure
plot(state)               % промежуточные состояния сумматора

```

Входной и выходной сигналы фильтра показаны на рис. 7.7, *a*, а промежуточные результаты суммирования — на рис. 7.7, *б*. Из графиков видно, что промежуточные значения достигают  $\pm 1,3$  и превосходят амплитуду как входного, так и выходного сигналов. Это может привести к переполнениям в арифметическом устройстве и искажениям в выходном сигнале.

При реализации фильтра в канонической форме (см. рис. 4.13) вычисления выполняются сначала в рекурсивной ветви, а затем в нерекурсивной. При этом уровень сигнала, отсчеты которого хранятся в единственной линии задержки фильтра, может значительно превышать амплитуды входного и выходного сигналов.

Продемонстрируем это, рассмотрев тот же фильтр Баттерворт 4-го порядка, который использовался в предыдущем примере. Сигнал, хранящийся в линии задержки при канонической реализации фильтра, получается в результате обработки входного сигнала "чисто рекурсивным" фильтром, в нерекурсивной ветви которого не используется задержанных отсчетов. Для оценки уровня этого сигнала достаточно рассчитать АЧХ рекурсивной части фильтра (рис. 7.8):

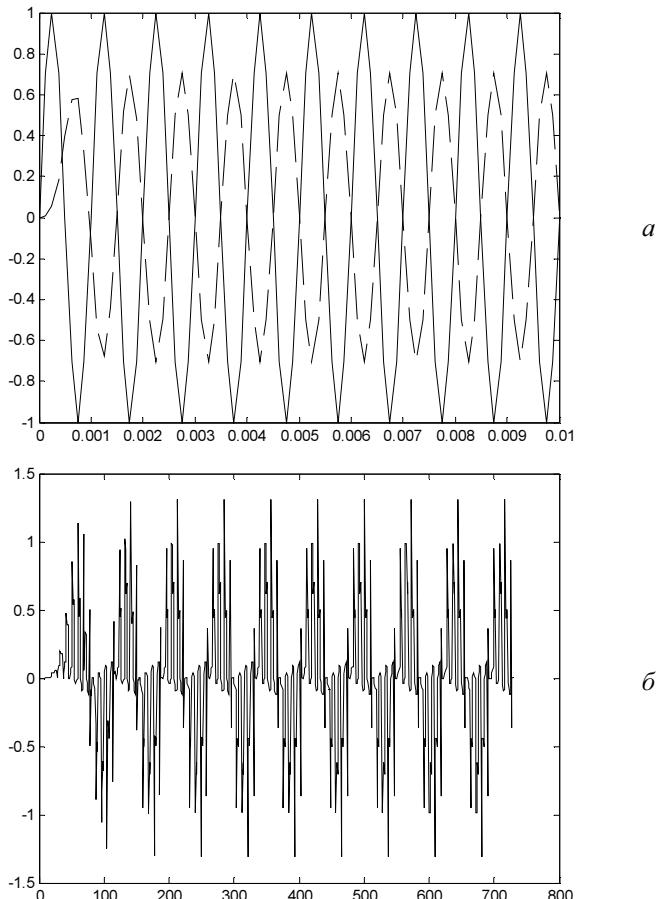
```

>> [b, a] = butter(4, 1/4);      % расчет фильтра
>> [h, f] = freqz(1, a, [], 2); % АЧХ рекурсивной ветви
>> plot(f, abs(h))            % график АЧХ
>> grid on

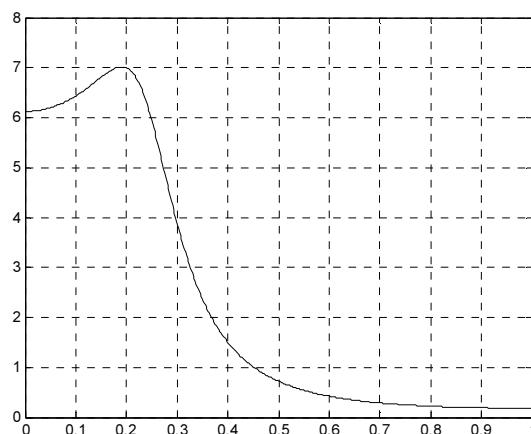
```

В данном случае для возникновения переполнений оказывается еще больше возможностей, чем в предыдущем — как видно из графика, коэффициент передачи рекурсивной части фильтра может достигать семи. Это означает, что промежуточные результаты вычислений для канонической формы реализации фильтра могут в 7 раз превосходить амплитуду входного сигнала!

Приведенные примеры показывают, что переполнения в процессе вычислений могут стать серьезной проблемой при реализации алгоритмов цифровой обработки сигналов. Уменьшить вероятность переполнений можно, используя для хранения чисел формат с плавающей запятой, обладающий большим динамическим диапазоном.



**Рис. 7.7.** Входной и выходной сигналы ФНЧ Баттервортса (*а*) и промежуточные результаты суммирования (*б*)



**Рис. 7.8.** АЧХ рекурсивной части ФНЧ Баттервортса 4-го порядка

ном, однако в этом случае возрастает сложность реализации арифметических операций. В любом случае разработанная система обработки сигналов должна тщательно тестироваться для обнаружения потенциальных проблем, связанных с конечной точностью вычислений. Такое тестирование возможно реализовать, например, с помощью средств пакета Filter Design, рассматриваемых далее в этой главе.

## Округление промежуточных результатов вычислений

При реализации различных алгоритмов обработки сигналов в процессе вычислений формируется множество промежуточных результатов. Формат хранения этих результатов зачастую вынуждает округлять их, что приводит к появлению дополнительных погрешностей. Операции, при которых появляются эти погрешности, зависят от способа представления чисел, используемого в вычислительном устройстве.

При использовании арифметики с фиксированной запятой операции сложения и вычитания не приводят к необходимости округления результатов — они могут вызвать лишь переполнение. Действительно, ведь количество знаков после запятой у результата сложения двух чисел, представленных в формате с фиксированной запятой, такое же, как было у слагаемых (слева суммируемые числа записаны в двоичной системе счисления, справа — в десятичной):

$$\begin{array}{r} 10,1100 \\ + \quad \quad \quad 2,75 \\ \hline 01,0101 \quad \quad 1,3125 \\ \hline \hline 100,0001 \quad \quad 4,0625 \end{array}$$

В данном примере суммируемые числа имели по 4 двоичных разряда после запятой, и столько же разрядов после запятой необходимо для представления их суммы.

А вот для представления *целой части* суммы нужно три разряда, тогда как для слагаемых было достаточно двух. Поэтому, если для представления целой части чисел используется два разряда, в данном случае возникнет *переполнение*.

В отличие от сложения, умножение чисел с фиксированной запятой приводит к увеличению числа значащих цифр результата (по сравнению с сомножителями) и, следовательно, к необходимости округления. Перемножим те же два числа, которые использовались в примере сложения:

$$\begin{array}{r} 10,1100 \quad \quad \quad 2,75 \\ \times \quad \quad \quad \times \\ 01,0101 \quad \quad 1,3125 \\ \hline \hline 10\ 1100 \quad \quad 3,609375 \\ 1011\ 00 \\ \hline \hline 11,1001\ 1100 \end{array}$$

Результат умножения в данном примере имеет 6 значащих цифр после запятой. Если для представления результата используется тот же формат, что и для сомножителей, придется производить округление:

$$11,10011100 \approx 11,1010 \quad 3.609375 \approx 3,625$$

Если для представления чисел используется формат с плавающей запятой, все сказанное об умножении сохраняет силу. Впрочем, если результат умножения по модулю не превосходит единицы, использование формата с плавающей запятой даст большую точность, поскольку не будут зря тратиться разряды для представления незначащих нулей слева.

Однако операции сложения в формате с плавающей запятой тоже могут приводить к потере точности. Пусть рассматриваемые нами два числа представлены в формате с плавающей запятой, использующем 6 разрядов для хранения дробной части нормализованной мантиссы (мантиssa и порядок записаны в двоичной системе счисления):

$$10,1100 = 0,101100 \cdot 2^{10}$$

$$01,0101 = 0,101010 \cdot 2^{01}$$

Представим результат сложения тоже в экспоненциальном формате:

$$0,101100 \cdot 2^{10} + 0,101010 \cdot 2^{01} = 0,1000001 \cdot 2^{11}$$

Как видите, для точного представления результата шести значащих цифр мантиссы оказалось мало, а значит, придется выполнять округление. В данном случае, когда погрешность округления в обе стороны одинакова, возможно использование различных подходов. Выберем, например, округление в большую сторону:

$$0,1000001 \cdot 2^{11} \approx 0,100001 \cdot 2^{11} \quad 4,0625 \approx 4,125$$

В результате при сложении чисел с плавающей запятой может не выполняться свойство ассоциативности — результат суммирования нескольких чисел может зависеть от последовательности выполнения сложений. При обучении программированию обычно дается рекомендация складывать числа, начиная с меньших (по модулю) и заканчивая самыми большими. Однако при реализации алгоритмов цифровой обработки сигналов мы, как правило, лишены возможности произвольно менять последовательность суммирования (хотя бы потому, что для сортировки слагаемых потребуется дополнительное время).

## Аналитическая модель собственного шума в фильтрах с фиксированной запятой

Для статистического анализа ошибок округления, возникающих при выполнении вычислений с фиксированной запятой, в структурную схему фильтра вводятся эквивалентные источники шума, добавляемого к результатам умножения (ведь, как уже было сказано ранее, при сложении чисел с фиксированной запятой погрешностей не возникает) [8].

Предполагается, что эти источники обладают следующими свойствами:

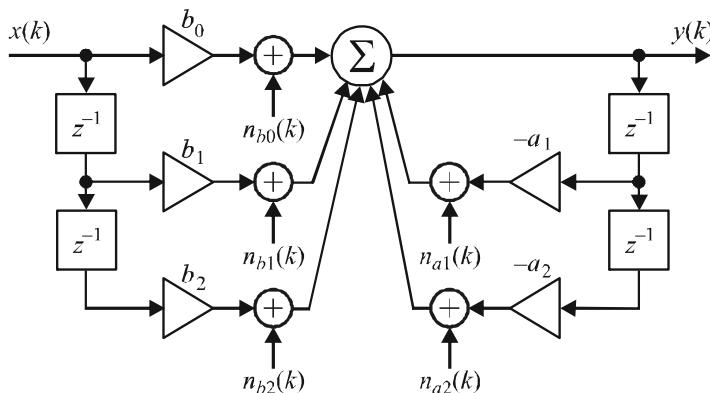
- распределение вероятности генерируемых ими шумов является равномерным, ширина диапазона равномерного распределения равна единице младшего разряда используемого формата представления чисел;
- шумовые отсчеты, генерируемые источниками, являются некоррелированными, т. е. генерируется белый шум;
- шумы, генерируемые разными источниками, являются статистически независимыми;
- собственные шумы фильтра и обрабатываемый полезный сигнал также являются статистически независимыми.

Условия, при которых эти предположения близки к реальности, такие же, как и для выполнения аналогичных предположений о шумах квантования сигнала — сетка уровней равномерного квантования должна быть настолько частой, чтобы значение сигнала между соседними отсчетами изменялось на *несколько* уровней.

Способ организации вычислений в фильтре определяется его структурной схемой, поэтому для фильтров, реализованных в разных формах, статистические свойства собственных шумов тоже будут разными. Проанализируем спектральные свойства собственных шумов для фильтров второго порядка, реализованных в соответствии со структурными схемами, рассмотренными в главе 4 (см. рис. 4.11, 4.13, 4.15 и 4.16).

## Прямая форма

Структурная схема фильтра второго порядка, реализованного в прямой форме, включающая источники собственных шумов, показана на рис. 7.9.



**Рис. 7.9.** Прямая форма реализации цифрового фильтра с источниками шума округления

Из рис. 7.9 видно, что в данном случае все пять собственных шумов являются слагаемыми общей суммы, формирующей выходной сигнал фильтра. Таким образом,

шум от любого источника проходит на выход только через рекурсивную часть фильтра, так что функция передачи для любого источника шума оказывается равна

$$H_n(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}.$$

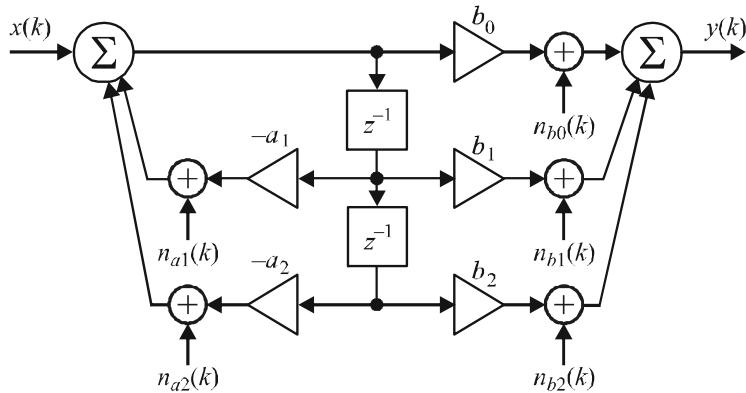
Соответствующий коэффициент передачи по мощности равен  $|H_n(e^{-j\omega T})|^2$ , вследствие независимости источников шумов их мощности суммируются. В итоге получаем СПМ собственного шума на выходе фильтра, равную

$$W_n(\omega) = \frac{5\sigma_n^2}{f_d} \frac{1}{|1 + a_1 e^{-j\omega T} + a_2 e^{-j2\omega T}|^2}, \quad (7.2)$$

где  $\sigma_n^2$  — дисперсия каждого источника шума (входящее в формулу отношение  $\sigma_n^2/f_d$  дает двустороннюю СПМ этого источника).

## Каноническая форма

Структурная схема фильтра второго порядка, реализованного в канонической форме, включающая источники собственных шумов, показана на рис. 7.10.



**Рис. 7.10.** Каноническая форма реализации цифрового фильтра с источниками шума окружения

Из рис. 7.10 видно, что шумы  $n_{bi}(k)$  приложены непосредственно к выходу схемы и не проходят через какие-либо рекурсивные ветви. Что касается шумов  $n_{ai}(k)$ , то они приложены к входу схемы и, таким образом, преобразуются в соответствии с функцией передачи фильтра.

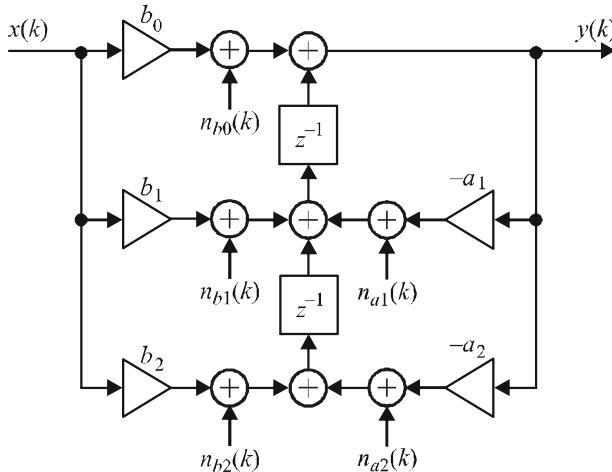
В результате СПМ собственного шума на выходе данной схемы можно записать в виде:

$$W_n(\omega) = \frac{\sigma_n^2}{f_d} \left( 3 + 2 \left| \frac{b_0 + b_1 e^{-j\omega T} + b_2 e^{-j2\omega T}}{1 + a_1 e^{-j\omega T} + a_2 e^{-j2\omega T}} \right|^2 \right) = \frac{\sigma_n^2}{f_d} \left( 3 + 2 |\dot{K}(\omega)|^2 \right), \quad (7.3)$$

где  $\dot{K}(\omega)$  — комплексный коэффициент передачи фильтра.

## Транспонированная форма

Структурная схема фильтра второго порядка, реализованного в транспонированной форме, включающая источники собственных шумов, показана на рис. 7.11.



**Рис. 7.11.** Транспонированная форма реализации цифрового фильтра с источниками шума округления

Из рисунка видно, что все шумы приложены к выходу фильтра внутри петель обратных связей и, следовательно, преобразуются рекурсивной частью фильтра. При этом шумы разных источников приобретают различные дополнительные задержки, но вследствие независимости и стационарности шумов это не имеет значения. В итоге СПМ собственного шума в данном случае описывается той же формулой (7.2), что и для прямой формы.

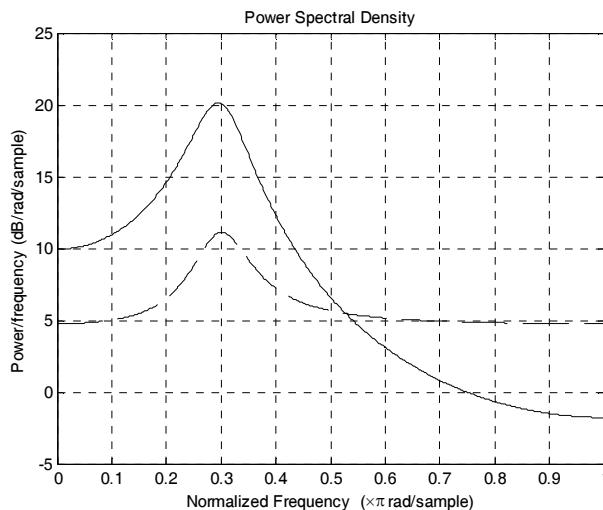
Аналогичным образом можно показать, что для варианта транспонированной формы, показанного на рис. 4.16, СПМ собственного шума определяется той же формулой (7.3), что и для канонической формы.

## Пример расчета собственных шумов

В качестве примера построим графики СПМ собственных шумов для резонатора второго порядка, АЧХ которого была показана на рис. 4.28, согласно формулам (7.2) и (7.3). СПМ шумовых источников примем равной единице:

```
>> [b, a] = iirpeak(0.3, 0.1); % расчет фильтра
>> w_direct = 5*abs(freqz(1, a)).^2; % прямая форма
```

```
>> w_canon = 3 + 2*5*abs(freqz(b, a)).^2; % каноническая форма
>> hpsd = dspdata.psd([w_direct w_canon]);
>> plot(hpsd)
```



**Рис. 7.12.** СПМ собственных шумов резонатора второго порядка, реализованного в прямой (сплошная линия) и канонической (пунктирная линия) формах

Результат работы данного кода показан на рис. 7.12. Кривые показывают, что СПМ собственного шума цифрового фильтра существенным образом зависит от формы его реализации.

Что касается ошибок округления в цифровых фильтрах с плавающей запятой, при их аналитическом описании возникает ряд проблем. Во-первых, как уже было сказано, погрешности в этом случае возникают не только при умножении, но и при сложении; во-вторых, из-за экспоненциального формата представления чисел масштаб погрешностей оказывается пропорциональным результату арифметической операции, при которой эти погрешности возникают. Иными словами, шумы в таких схемах являются не аддитивными, а *мультипликативными* [8]. Поэтому статистический анализ собственных шумов фильтров с плавающей запятой оказывается крайне сложным и на практике чаще всего заменяется компьютерным моделированием.

Примеры моделирования ошибок округления в цифровом фильтре и при расчете БПФ будут приведены далее в этой главе, при описании объектов квантованных фильтров и квантованного БПФ.

## Предельные циклы

Ошибки округления при вычислениях в цифровых фильтрах могут приводить к еще одной очень серьезной неприятности — появлению так называемых *предельных*

циклов (*limit cycle*), когда вроде бы устойчивый фильтр начинает демонстрировать неустойчивое поведение.

Поясним это явление на простейшем примере, взятом из [8]. Пусть имеется рекурсивный фильтр первого порядка, описываемый разностным уравнением

$$y(k) = x(k) + 0,95 y(k - 1).$$

Единственный полюс функции передачи данного фильтра, равный 0,95, находится внутри единичной окружности, что свидетельствует об устойчивости фильтра. Пусть входной сигнал отсутствует ( $x(k) = 0$ ), а внутреннее состояние фильтра, которое в данном случае представляется единственным числом  $y(0)$ , равно 13. При точных вычислениях сигнал на выходе фильтра экспоненциально затухает:

$$y(1) = 0,95 y(0) = 0,95 \cdot 13 = 12,35,$$

$$y(2) = 0,95 y(1) = 0,95 \cdot 12,35 = 11,7325,$$

$$y(3) = 0,95 y(2) = 0,95 \cdot 11,7325 = 11,145875,$$

$$y(4) = 0,95 y(3) = 0,95 \cdot 11,145875 = 10,58858125,$$

$$y(5) = 0,95 y(4) = 0,95 \cdot 10,58858125 = 10,0591521875,$$

...

А теперь будем считать, что в ячейке памяти фильтра значение хранится в *целочисленном формате* и после выполнения умножения используется *округление* (будем обозначать его квадратными скобками). Выходной сигнал радикально изменится:

$$y(1) = [0,95 y(0)] = [0,95 \cdot 13] = [12,35] = 12,$$

$$y(2) = [0,95 y(1)] = [0,95 \cdot 12] = [11,4] = 11,$$

$$y(3) = [0,95 y(2)] = [0,95 \cdot 11] = [10,45] = 10,$$

$$y(4) = [0,95 y(3)] = [0,95 \cdot 10] = [9,5] = 10,$$

$$y(5) = [0,95 y(4)] = [0,95 \cdot 10] = [9,5] = 10,$$

...

Итак, выходной сигнал фильтра "залипает" на значении 10 и далее не меняется. Это и есть простейший случай предельного цикла (с периодом, равным единице).

Если в разностном уравнении фильтра изменить знак у коэффициента 0,95, мы получим предельный цикл с периодом, равным двум, в виде чередования значений 10 и -10. В фильтрах более высокого порядка могут возникать предельные циклы с разным периодом.

Имеются две *разновидности* предельных циклов:

- предельные циклы низкого уровня (*granular limit cycle*) возникают, когда значения внутреннего состояния фильтра при отсутствии входного сигнала затухают, но из-за ошибок округления не доходят до нуля. Именно такой цикл наблюдается в приведенном примере;

- предельные циклы высокого уровня (*overflow limit cycle*) имеют место в том случае, когда из-за вычислительных погрешностей значения внутреннего состояния фильтра при отсутствии входного сигнала не затухают, а возрастают, вызывая переполнение.

Для анализа возможностей возникновения предельных циклов используется понятие **эффективных значений коэффициентов знаменателя** функции передачи фильтра. Под эффективным значением коэффициента понимается отношение округленного результата умножения к использованному в качестве множителя значению внутреннего состояния фильтра. Эффективное значение коэффициента будет разным для разных значений состояния. Если для какого-то внутреннего состояния фильтра набор эффективных значений коэффициентов дает полюс функции передачи, расположенный *на единичной окружности*, это означает, что данное внутреннее состояние при отсутствии входного сигнала дает предельный цикл.

В нашем примере эффективное значение единственного коэффициента фильтра равно

$$a_{\text{эфф}} = \frac{[0,95y(k-1)]}{y(k-1)}.$$

Построим график значений  $a_{\text{эфф}}$  для целочисленных значений  $y(k-1)$ , лежащих в диапазоне от  $-20$  до  $20$  (рис. 7.13):

```
>> y = -20:20;
>> y(find(~y))=[]; % удаляем нулевое значение
>> a_eff = round(0.95*y). /y;
>> plot(y, a_eff, '.')
>> ylim([0.9 1.1])
>> xlabel('y(k-1)')
>> ylabel('a_{eff}')
```

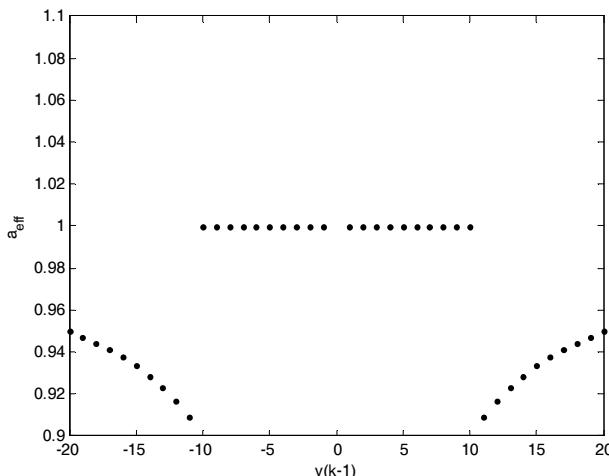


Рис. 7.13. Зависимость эффективного значения коэффициента рекурсивного фильтра первого порядка от внутреннего состояния фильтра

Как видите, если  $y(k - 1)$  лежит в диапазоне от  $-10$  до  $10$ , значение  $a_{\text{эфф}}$  становится равным единице, а фильтр оказывается *на границе устойчивости* (полюс функции передачи лежит на единичной окружности), что и приводит к появлению предельного цикла.

В приведенном примере анализ предельных циклов оказывается очень простым, поскольку внутреннее состояние фильтра описывается единственным числом. Для фильтров, степень знаменателя функции передачи которых превышает единицу, анализ базируется на тех же принципах, но оказывается более сложным. Для выполнения такого анализа в пакете Filter Design имеется специальная функция `limitcycle`, которую мы рассмотрим далее в этой главе.

## Учет эффектов конечной точности вычислений в MATLAB

В пакетах расширения MATLAB имеется ряд средств для учета эффектов, связанных с конечной точностью представления чисел в вычислительных системах. К таким средствам относятся функции, реализующие равномерное и неравномерное квантование, а также объекты квантователей, квантованных фильтров и квантованных БПФ.

### Функции квантования

Функции квантования имеются в двух пакетах расширения MATLAB. В пакете Signal Processing содержатся функции `uencode` и `udecode`, осуществляющие равномерное (буква "u" в именах функций означает "uniform" — равномерный) квантование и восстановление сигнала. Пакет Communications предоставляет в распоряжение пользователя более гибкую функцию неравномерного квантования `quantiz`, а также функцию `lloyds`, предназначенную для оптимизации параметров неравномерного квантования по тестовому набору данных.

#### Функция `uencode`

Функция `uencode` осуществляет равномерное квантование входного сигнала. Синтаксис вызова функции следующий:

```
у = uencode(х, n, v, 'signflag')
```

Здесь  $x$  — массив неквантованных отсчетов входного сигнала (числа с плавающей запятой),  $n$  — число двоичных разрядов, используемое для представления квантованных значений, которое может лежать в диапазоне от 2 до 32.

Остальные входные параметры являются необязательными. Параметр  $v$  задает предельное абсолютное значение квантуемого сигнала. По умолчанию этот параметр равен 1, так что уровни квантования перекрывают диапазон  $-1\dots1$ .

Результатом работы функции является массив  $y$ , содержащий целые числа — номера уровней квантования, соответствующих отсчетам из массива  $x$ . Размеры массивов  $x$  и  $y$  совпадают.

Способ нумерации уровней квантования определяется последним входным параметром 'signflag', который может принимать одно из двух строковых значений:

- 'unsigned' — в массиве  $y$  содержатся целые числа от 0 до  $2^n - 1$  (этот вариант принят по умолчанию);
- 'signed' — в массиве  $y$  содержатся целые числа от  $-2^{n-1}$  до  $2^{n-1} - 1$ .

MATLAB хранит выходной массив  $y$  в одном из целочисленных форматов данных согласно табл. 7.3.

**Таблица 7.3.** Типы данных, используемые функцией `uencode`

Число разрядов $n$	Тип выходных данных	
	'signflag' = 'unsigned'	'signflag' = 'signed'
2...8	uint8	int8
9...16	uint16	int16
17...32	uint32	int32

Если значения входного сигнала выходят за пределы диапазона  $-v \dots v$ , происходит *насыщение (saturation)*: значениям, меньшим  $-v$ , соответствует минимально возможное значение  $y$  (0 или  $-2^{n-1}$ ), а значениям, превышающим  $v$  — максимально возможное ( $2^n - 1$  или  $2^{n-1} - 1$ ).

Обратное преобразование номеров уровней квантования в квантованные значения производится рассматриваемой далее функцией `udecode`.

### Функция `udecode`

Функция `udecode` является обратной по отношению к `uencode` — она превращает номера уровней квантования в соответствующие квантованные значения. Синтаксис вызова функции следующий:

```
x = udecode(y, n, v, 'saturatemode')
```

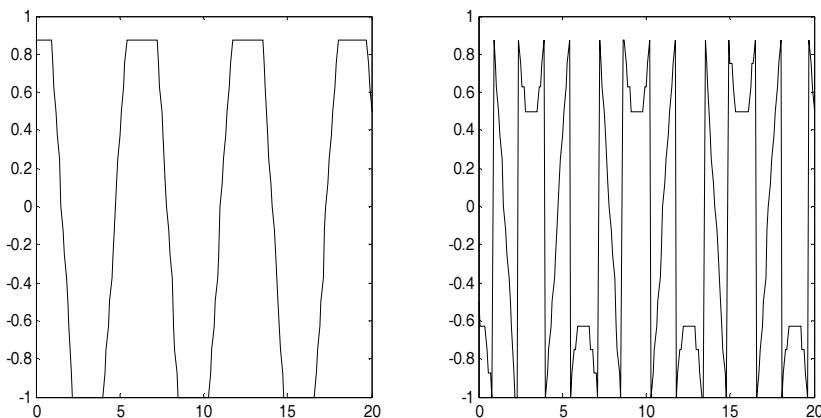
Смысл параметров  $x$ ,  $y$ ,  $n$  и  $v$  здесь тот же, что и для функции `uencode`.

Строчный параметр 'saturatemode' определяет поведение функции в случае, если в массиве  $y$  обнаруживаются значения, представленные большим числом двоичных разрядов, чем указано в параметре  $n$ :

- 'saturate' (насыщение; этот вариант принят по умолчанию) — при выходе значений  $y$  из допустимого диапазона для их представления в массиве  $x$  используются крайние (минимальный и максимальный) уровни квантования;
- 'wrap' (перенос) — при выходе значений  $y$  из допустимого диапазона "лишние" старшие разряды просто игнорируются, что эквивалентно смещению номеров уровней квантования на  $2^n$  (возможно, неоднократному).

В качестве примера использования функций `uencode` и `udecode` произведем квантование синусоидального сигнала, а затем при восстановлении квантованных значений уменьшим число двоичных разрядов, чтобы продемонстрировать влияние параметра '`saturatemode`' (рис. 7.14):

```
>> t = 0:0.1:20;
>> s = 0.75 * cos(t);
>> y = uencode(s, 5, 1, 'signed');      % 32 уровня квантования
>> % при восстановлении используем 16 уровней квантования
>> x1 = udecode(y, 4, 1, 'saturate');   % режим насыщения
>> x2 = udecode(y, 4, 1, 'wrap');       % режим переноса
>> subplot(1, 2, 1)
>> plot(t, x1)
>> subplot(1, 2, 2)
>> plot(t, x2)
```



**Рис. 7.14.** Квантованный сигнал, восстановленный при недостаточном числе уровней квантования: слева — режим насыщения, справа — режим переноса

## Функция `quantiz`

Функция `quantiz` реализует неравномерное квантование сигнала, описанное в этой главе ранее. Синтаксис вызова функции следующий:

```
[index, quants, distor] = quantiz(sig, partition, codebook)
```

Здесь `sig` — вектор отсчетов сигнала, `partition` — вектор границ зон квантования (в разд. "Неравномерное квантование" этой главы для этих границ использовалось обозначение  $a_k$ ), `codebook` — вектор уровней квантования (в упомянутом разделе —  $b_k$ ). Длина этого вектора должна на единицу превосходить длину вектора `partition`.

Выходные параметры `index` и `quants` имеют такие же размеры, как и вектор сигнала, и содержат соответственно индексы и значения квантованного сигнала.

Результат `index` определяется следующим образом:

- если  $\text{sig}(k) \leq \text{partition}(1)$ , то  $\text{index}(k) = 0$ ;
- если  $\text{partition}(m) < \text{sig}(k) \leq \text{partition}(m+1)$ , то  $\text{index}(k) = m$ ;
- если  $\text{sig}(k) > \text{partition}(\text{end})$ , то  $\text{index}(k) = \text{length}(\text{partition})$ .

Результат `quants` содержит квантованные значения входного сигнала и рассчитывается как `codebook(index+1)`. Таким образом, величина `codebook(1)` задает уровень квантования для всех значений сигнала, меньших, чем `partition(1)`, величина `codebook(m)` соответствует зоне квантования `partition(m-1)...partition(m)`, а последнее значение `codebook(end)` будет использоваться в качестве уровня квантования для всех значений сигнала, превышающих `partition(end)`.

Третий выходной параметр `distor` содержит значение среднего квадрата ошибки квантования.

Если выходные параметры `quants` и `distor` не используются, входной параметр `codebook` при вызове функции можно опустить.

## Функция *lloyds*

Функция `lloyds` осуществляет оптимизацию параметров неравномерного квантования, основываясь на тестовом сигнале, являющемся "типичным представителем" тех данных, которые будут затем подвергаться квантованию. Синтаксис вызова функции следующий:

```
[partition, codebook, distor] = lloyds(x, initcodebook, tol, plotflag)
```

Здесь `x` — тестовый сигнал, который должен быть достаточно длинным, чтобы по нему можно было оценить плотность вероятности отсчетов, `initcodebook` — начальное приближение для набора уровней квантования. Этот параметр может также быть не вектором, а числом, в таком случае он задает лишь количество уровней квантования, а начальное приближение выбирается автоматически.

Оставшиеся два входных параметра являются необязательными и при вызове могут быть опущены.

Параметр `tol` позволяет управлять критерием завершения итерационного алгоритма поиска оптимального решения. Если относительное изменение среднего квадрата ошибки на очередной итерации оказывается меньше `tol`, решение считается найденным. По умолчанию значение параметра `tol` равно  $10^{-7}$ .

При использовании параметра `plotflag` функция выводит график тестового сигнала, показывая горизонтальными линиями зеленого цвета подобранные уровни квантования, а линиями красного цвета — границы зон квантования. Значение параметра `plotflag` может быть любым — важен лишь факт его наличия или отсутствия.

Результаты работы функции — выходные параметры `partition`, `codebook` и `distor` — имеют тот же смысл, что и для функции `quantiz`.

## Объекты квантователей

Пакет Fixed-Point, как следует из его названия, предназначен для реализации вычислений с фиксированной запятой. В частности, в его состав входят квантователи, реализованные в виде *объектов MATLAB* (см. разд. "Общая информация об объектах MATLAB" главы 4).

Для создания объекта квантователя служит функция `quantizer`. У данного объекта немного свойств, доступных для записи, причем числовым является только одно из них, а все строковые свойства имеют уникальные значения. Это позволяет вызывать конструктор объекта, указывая только значения свойств и опуская их имена:

```
q = quantizer(value1, value2, ...)
```

При этом не обязательно перечислять значения всех свойств — те из них, которые не заданы при вызове функции, сохранят значения, принятые по умолчанию.

Разумеется, возможен и полный вариант вызова, когда указываются и имена, и значения свойств:

```
q = quantizer('name1', value1, 'name2', value2, ...)
```

Здесь '`name1`', '`name2`' и т. д. — имена свойств, а `value1`, `value2` и т. д. — соответствующие значения.

Список свойств квантователя фильтра приведен в табл. 7.4. Считывать и задавать значения этих свойств можно с помощью функций `get` и `set` соответственно (см. разд. "Общая информация об объектах MATLAB" главы 4).

Значения свойств `Max`, `Min`, `Noperations`, `Noverflows` и `Nunderflows`, предоставляющих статистическую информацию и доступных только для чтения, обновляются при каждом выполнении операции квантования (функция `quantize`, см. далее) и обнуляются при выполнении функции `reset`.

Операция квантования выполняется с помощью функции `quantize`. Синтаксис ее вызова следующий:

```
y = quantize(q, x)
```

Здесь `q` — объект квантователя, а `x` — массив значений входного сигнала. Результатом работы является массив квантованных значений `y`. Размеры массивов `x` и `y` одинаковы.

Можно вызывать функцию `quantize` для квантования нескольких переменных одновременно:

```
[y1, y2, ...] = quantize(q, x1, x2, ...)
```

Здесь `y1` — результат квантования `x1`, `y2` — результат квантования `x2` и т. д.

Как отмечалось в разд. "Формат с фиксированной запятой" этой главы, диапазон представимых в таком формате чисел является несимметричным. В частности, если целая часть содержит единственный знаковый разряд, оказывается невозможным точно представить значение +1. Для решения этой проблемы в пакете Filter Design имеются функция `unitquantize` и объект `unitquantizer`.

Таблица 7.4. Свойства объекта квантователя

Имя свойства	Описание
<b>Свойства, доступные для чтения и записи</b>	
Mode	<p><i>Тип квантователя:</i></p> <p>'double' — квантователь с плавающей запятой, стандартный 64-битовый формат double. Все остальные параметры в данном случае игнорируются;</p> <p>'float' — квантователь с плавающей запятой;</p> <p>'fixed' — квантователь с фиксированной запятой (этот вариант принят по умолчанию);</p> <p>'single' — квантователь с плавающей запятой, стандартный 32-битовый формат single. Все остальные параметры в данном случае игнорируются;</p> <p>'ufixed' — беззнаковый квантователь с фиксированной запятой</p>
RoundMode	<p><i>Режим округления:</i></p> <p>'ceil' — округление вверх;</p> <p>'convergent' — округление к ближайшему целому значению, в случае равенства расстояний выбирается <i>четное</i> значение;</p> <p>'fix' — округление к нулю;</p> <p>'floor' — округление вниз (этот вариант принят по умолчанию);</p> <p>'round' — округление к ближайшему целому значению, в случае равенства расстояний выбирается значение, <i>большее по модулю</i></p>
OverflowMode	<p>Реакция на переполнение (только для формата с фиксированной запятой):</p> <p>'saturate' — при переполнении происходит насыщение (этот вариант принят по умолчанию);</p> <p>'wrap' — при переполнении игнорируются "лишние" старшие разряды</p>
Format	Двухэлементный вектор, первый элемент которого задает общее число двоичных разрядов, используемое для представления квантованных чисел. Второй элемент вектора для квантователей с фиксированной запятой (Mode = 'fixed' или 'ufixed') указывает число разрядов дробной части, а для квантователей с плавающей запятой (Mode = 'float') — число разрядов, используемое для представления порядка. Значение по умолчанию равно [16 15]
<b>Свойства, доступные только для чтения</b>	
Max	Максимальное значение входного (неквантованного) сигнала, зафиксированное за время работы квантователя
Min	Минимальное значение входного (неквантованного) сигнала, зафиксированное за время работы квантователя
Noverflows	Число переполнений, произошедших за время работы квантователя
Nunderflows	Число случаев потери значимости (когда ненулевой входной сигнал в результате квантования округляется до нуля; используется также термин "антипереполнение"), произошедших за время работы квантователя
Noperations	Количество выполненных операций квантования

Функция `unitquantize` вызывается точно так же, как и функция `quantize`:

```
y = unitquantize(q, x)
[y1, y2, ...] = unitquantize(q, x1, x2, ...)
```

Работа этой функции отличается от функции `quantize` лишь в одном аспекте: значения, близкие к единице (точнее, отстоящие от нее не более чем на `eps`, где `eps` — точность представления чисел в квантователе, см. далее), при квантовании становятся равными единице.

Чтобы пояснить разницу между функциями `quantize` и `unitquantize`, приведем простой пример. Создадим квантователь с фиксированной запятой, использующий один разряд для целой части и 7 — для дробной:

```
>> q = quantizer('fixed', [8 7]);
```

Теперь произведем квантование единичного значения с помощью данного квантователя и функций `quantize` и `unitquantize`:

```
>> quantize(q, 1)
Warning: 1 overflow.
> In embedded.quantizer.quantize at 74
ans =
    0.9922

>> unitquantize(q, 1)
ans =
    1
```

При использовании функции `quantize` было выдано предупреждение о произошедшем переполнении и указана соответствующая строка кода в M-файле функции `quantize`, а квантованное значение оказалось меньше единицы (оно равно  $1 - 2^{-7}$ , т. е. максимальному числу, представимому в используемом нами формате 1.7). При использовании же функции `unitquantize` переполнения не происходит, а результат квантования оказывается равен единице.

Той же цели — обеспечению точного представления единичных значений — служит и объект `unitquantizer`. Он полностью аналогичен объекту `quantizer`, но для него не нужно явно вызывать функцию `unitquantize` — это будет сделано автоматически при обычном квантовании.

Для иллюстрации сказанного создадим объект `unitquantizer` с теми же параметрами, как у квантователя в предыдущем примере, и выполним квантование единичного значения:

```
>> uq = unitquantizer('fixed', [8 7]);
>> quantize(uq, 1)
ans =
    1
```

Как видите, переполнения не произошло, а результат равен единице.

### **ВНИМАНИЕ!**

Перечисленные трюки с квантованием единичных значений становятся возможны исключительно благодаря тому, что MATLAB использует для представления всех дробных чисел (как неквантованных, так и квантованных) формат с плавающей запятой `double`. При реализации алгоритмов обработки сигнала "в железе" или на языках высокого уровня с использованием целочисленной арифметики точно представить непредставимые значения, естественно, не удастся. Зачем же тогда пакет Fixed-Point содержит такие средства? Дело в том, что объекты квантователей предназначены прежде всего для моделирования различных алгоритмов цифровой обработки сигналов. При реальной обработке сигналов аппаратными средствами проблему с единичными значениями можно обойти, проигнорировав операции умножения отсчетов сигнала на коэффициенты, равные единице. В модели же можно, не модифицируя общий алгоритм вычислений, просто использовать квантователи типа `unitquantizer`. Например, это делается при включении режима оптимизации умножений в объектах квантованного БПФ (свойство `OptimizeUnityGains`, см. далее).

Помимо функций `quantize` и `unitquantize` с объектами квантователей могут работать следующие функции:

- `x = bin2num(q, b)` — преобразование строки `b`, трактуемой как двоичная запись внутреннего представления квантованного числа в квантователе `q`, в число `x`;
- `q1 = copy(q)` — создает *независимую копию* объекта квантователя с теми же значениями свойств, что у исходного объекта `q` (см. приведенный в главе 4 комментарий к одноименной функции для объектов дискретных фильтров);
- `x = denormalmax(q)` — для квантователей с плавающей запятой возвращает значение наибольшего *денормализованного* положительного квантованного числа. Для квантователей с фиксированной запятой возвращает `eps(q)`;
- `x = denormalmin(q)` — для квантователей с плавающей запятой возвращает значение наименьшего *денормализованного* положительного квантованного числа. Для квантователей с фиксированной запятой возвращает `eps(q)`;
- `disp(q)` — выводит на экран свойства объекта;
- `e = eps(q)` — возвращает относительную точность представления чисел в квантователе `q`. Под точностью представления подразумевается минимальное число, при добавлении которого к единице результат квантования будет отличаться от единицы;
- `m = errmean(q)` — возвращает математическое ожидание ошибки квантования;
- `[f, x] = errpdf(q)` или `f = errpdf(q, x)` — возвращает плотность вероятности `f` равномерно распределенной ошибки квантования, вычисленную для значений аргумента из вектора `x`. Эти значения могут выбираться автоматически (первый вариант синтаксиса) или задаваться при вызове функции (второй вариант синтаксиса);
- `v = errvar(q)` — возвращает дисперсию ошибки квантования;
- `b = exponentbias(q)` — возвращает смещение, используемое в квантователе с плавающей запятой при представлении порядка квантованных чисел. В случае формата с фиксированной запятой результат всегда равен нулю;

- $e = \text{exponentlength}(q)$  — возвращает число двоичных разрядов, используемое в квантователе с плавающей запятой для представления порядка квантованных чисел. В случае формата с фиксированной запятой результат всегда равен нулю;
- $e_{\max} = \text{exponentmax}(q)$  — если в квантователе используется формат с плавающей запятой, возвращает максимальное значение порядка квантованных чисел. В случае формата с фиксированной запятой результат всегда равен нулю;
- $e_{\min} = \text{exponentmin}(q)$  — если в квантователе используется формат с плавающей запятой, возвращает минимальное значение порядка квантованных чисел. В случае формата с фиксированной запятой результат всегда равен нулю;
- $f = \text{fractionlength}(q)$  — возвращает количество разрядов дробной части квантованных чисел;
- $x = \text{hex2num}(q, h)$  — преобразование строки  $h$ , трактуемой как шестнадцатеричная запись внутреннего представления квантованного числа в квантователе  $q$ , в число  $x$ ;
- $\text{flag} = \text{isequal}(q_1, q_2, \dots)$  — возвращает 1, если значения свойств всех объектов квантователей  $q_1, q_2, \dots$  совпадают;
- $\text{flag} = \text{isfixed}(q)$  — возвращает 1, если объект  $q$  является квантователем с фиксированной запятой;
- $\text{flag} = \text{isfloat}(q)$  — возвращает 1, если объект  $q$  является квантователем с плавающей запятой;
- $\text{flag} = \text{isquantizer}(q)$  — возвращает 1, если объект  $q$  относится к классу **quantizer** или **unitquantizer**;
- $e = \text{lsb}(q)$  — возвращает относительную точность представления чисел в квантователе  $q$ . Под точностью представления подразумевается минимальное число, при добавлении которого к единице результат квантования будет отличаться от единицы;
- $x = \text{max}(q)$  — возвращает значение свойства **Max**;
- $x = \text{min}(q)$  — возвращает значение свойства **Min**;
- $n = \text{noperations}(q)$  — возвращает значение свойства **Noperations**;
- $n = \text{noverflows}(q)$  — возвращает значение свойства **Noverflows**;
- $b = \text{num2bin}(q, x)$  — преобразование числа  $x$  в строку  $b$ , представляющую собой двоичную запись квантованного числа, в соответствии с форматом, используемым квантователем  $q$ ;
- $h = \text{num2hex}(q, x)$  — преобразование числа  $x$  в строку  $h$ , представляющую собой шестнадцатеричную запись квантованного числа, в соответствии с форматом, используемым квантователем  $q$ ;
- $n = \text{num2int}(q, x)$  — преобразование числа  $x$  в целое число  $n$ , являющееся внутренним представлением квантованного значения, в соответствии с форматом, используемым квантователем с фиксированной запятой  $q$ . Для квантователей с плавающей запятой  $n = x$ ;

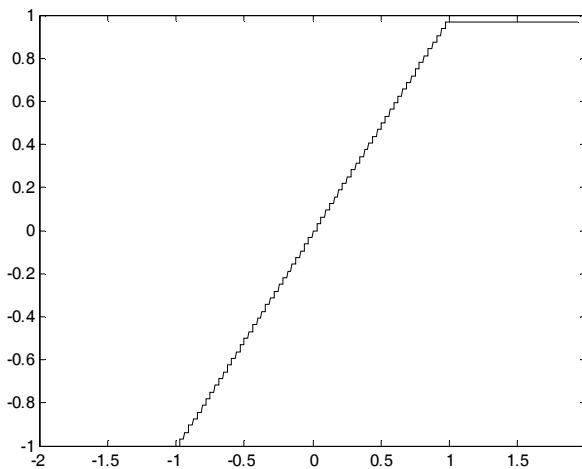
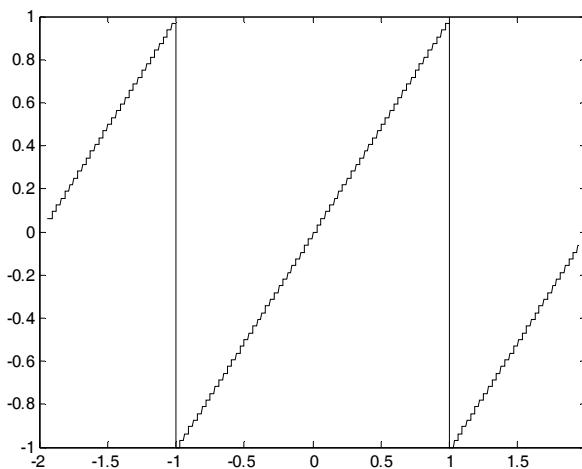
- `n = nunderflows(q)` — возвращает значение свойства `Nunderflows`;
- `s = qreport(q)` — возвращает структуру, содержащую статистическую информацию о работе квантователя (максимальное и минимальное значения входного сигнала, число переполнений и случаев потери значимости, а также количество выполненных квантований). При отсутствии выходного параметра информация выводится на экран;
- `x = randquant(q, m, n)` — создает матрицу случайных чисел, содержащую `m` строк и `n` столбцов и квантованных с помощью квантователя `q`. Для квантователей с фиксированной запятой генерируемые числа равномерно распределены в полном динамическом диапазоне квантователя. Для квантователей с плавающей запятой генерируемые числа равномерно распределены в диапазоне `-sqrt(realmax(q)) ... sqrt(realmax(q))`;
- `r = range(q)` — возвращает двухэлементный вектор, содержащий минимальное и максимальное значения выходного квантованного сигнала для квантователя `q`;
- `[a, b] = range(q)` — возвращает минимальное и максимальное значения выходного квантованного сигнала для квантователя `q` в виде двух отдельных переменных `a` (минимальное значение) и `b` (максимальное значение);
- `x = realmax(q)` — возвращает наибольшее положительное квантованное число;
- `x = realmin(q)` — возвращает наименьшее положительное нормализованное квантованное число;
- `reset(q)` — сбрасывает в исходное состояние значения свойств квантователя `q`, связанных со сбором статистической информации (`Max`, `Min`, `Noperations`, `Noverflows` и `Nunderflows`);
- `y = round(q, x)` — производит *округление* числа `x` с помощью квантователя `q`. В отличие от операции квантования, при этом не производится проверка и обработка переполнений;
- `s = tostring(q)` — возвращает строку вызова функции `quantizer`, в результате выполнения которой будет создан объект, совпадающий с объектом `q`;
- `w = wordlength(q)` — возвращает общее число двоичных разрядов, используемое квантователем для представления квантованных чисел.

В качестве примера использования функций `quantizer` и `quantize` создадим объекты квантователей с фиксированной и плавающей запятой, а затем построим для них графики характеристик квантования, т. е. зависимости выходного (квантованного) сигнала от входного (неквантованного).

Чтобы на этих графиках был хорошо виден ступенчатый характер зависимостей, число уровней квантования должно быть небольшим, поэтому будем использовать длину слова, равную шести двоичным разрядам.

В квантователе с фиксированной запятой дробная часть будет содержать пять разрядов. Для обработки переполнений зададим режим насыщения, а всем остальным свойствам оставим значения по умолчанию:

```
>> q_fixed = quantizer('fixed', [6 5], 'saturate')
q_fixed =
    Mode = fixed
    RoundMode = floor
    OverflowMode = saturate
    Format = [6 5]
```

*a**б*

**Рис. 7.15.** Характеристика квантования для квантователя с фиксированной запятой: при переполнении происходит насыщение (a), при переполнении отбрасываются старшие разряды (б)

Построим график характеристики квантования (рис. 7.15, a), выбрав такой диапазон значений входного сигнала, чтобы вызвать переполнение квантователя. Для автоматического определения этого диапазона используем функцию `realmax`:

```

>> x_max=2*realmax(q_fixed);
>> x = linspace(-x_max, x_max, 1000);
>> y = quantize(q_fixed, x);
Warning: 484 overflows.
> In embedded.quantizer.quantize at 74
>> plot(x, y)

```

Теперь изменим режим обработки переполнения, чтобы вместо насыщения происходило отбрасывание "лишних" старших разрядов. Затем снова построим характеристику квантования (рис. 7.15, б):

```

>> q_fixed.overflowmode = 'wrap';
>> y = quantize(q_fixed, x);
Warning: 484 overflows.
> In embedded.quantizer.quantize at 74
>> plot(x, y)

```

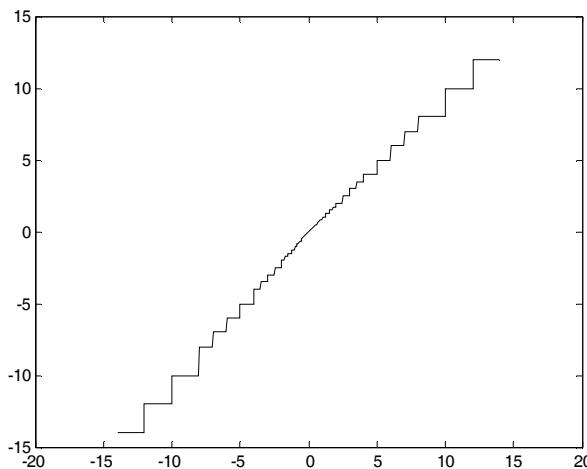
Графики на рис. 7.15 наглядно иллюстрируют сущность различий между режимами обработки переполнений 'saturate' и 'wrap'.

В квантователе с плавающей запятой используем три разряда для хранения порядка. Значения остальных свойств оставим установленными по умолчанию:

```

>> q_float = quantizer('float', [6 3])
q_float =
    Mode = float
    RoundMode = floor
    Format = [6 3]

```



**Рис. 7.16.** Характеристика квантования для квантователя с плавающей запятой

Теперь строим характеристику квантования так же, как и раньше (рис. 7.16):

```

>> x_max=1.2*realmax(q_float);
>> x = linspace(-x_max, x_max, 1000);

```

```
>> y = quantize(q_float, x);
Warning: 168 overflows.
> In embedded.quantizer.quantize at 74
>> plot(x, y)
```

Сравнение характеристик квантования на рис. 7.15 и 7.16 наглядно демонстрирует особенности квантователей с фиксированной и плавающей запятой.

## Квантованные фильтры

Моделирование цифровых фильтров с учетом шумов квантования и округления в настоящее время осуществляется с помощью объектов дискретных фильтров класса `dfilt`, рассмотренных в главе 4. При установленном пакете расширения Filter Design у этих объектов появляется строковое свойство `Arithmetic`, которое может принимать значения '`double`' (вариант по умолчанию; вычисления производятся с использованием 64-битового формата с плавающей запятой) и '`single`' (32-битовый формат с плавающей запятой).

Если установлен также пакет Fixed-Point, это свойство может принимать еще одно значение — '`fixed`', позволяя моделировать вычисления с фиксированной запятой. При этом у объектов появляется большое количество новых свойств, описывающих форматы представления чисел в различных точках структурной схемы. Полное описание этих свойств заняло бы слишком много места, к тому же их конкретный набор определяется формой реализации фильтра, а доступность некоторых свойств зависит от значений, присвоенных другим свойствам. Поэтому ограничимся лишь кратким обзором важнейших из них, а за полной информацией следует обращаться к документации MATLAB.

- свойства, имена которых заканчиваются на `WordLength` и `FracLength`, задают соответственно общее количество битов и количество битов дробной части для разных точек схемы. Например, пара свойств `InputWordLength` и `InputFracLength` относится к представлению входного сигнала, `CoeffWordLength` задает число битов в представлении коэффициентов фильтра, а `NumFracLength` — размер дробной части в коэффициентах числителя функции передачи и т. д.;
- свойства, имена которых заканчиваются на `AutoScale`, принимают логические значения и при значении `true` (1) включают автоматическое распределение общей длины слова между целой и дробной частями. Свойство `CoeffAutoScale` делает это для коэффициентов фильтра, `StateAutoScale` — для ячеек памяти фильтра и т. д. При значении `false` (0) становятся доступны свойства ...`FracLength` для соответствующих элементов схемы;
- `ProductMode` — режим обработки результатов умножений. Возможные значения: `FullPrecision` (сохраняются все разряды результата), `KeepLSB` (результат усекается с сохранением младших битов), `KeepMSB` (результат усекается с сохранением старших битов), `SpecifyPrecision` (формат представления результатов умножений задается вручную);

- RoundMode — используемый способ округления. Возможные значения такие же, что и для объектов квантователей (см. одноименное свойство в табл. 7.4);
- OverflowMode — используемый режим обработки переполнений. Возможные значения такие же, что и для объектов квантователей (см. одноименное свойство в табл. 7.4).

Обработка сигнала квантованным фильтром осуществляется, как обычно, с помощью функции `filter`:

```
у = filter(hd, x)
```

Здесь `hd` — объект квантованного фильтра, `x` — входной сигнал, `y` — выходной сигнал.

Квантованные фильтры имеют следующие специфичные для них методы (приведены лишь основные варианты синтаксиса их вызова, за полным описанием обратитесь к справочной системе):

- `hd1 = autoscale(hd, x)` — производит автоматическую настройку длины дробной части чисел в различных точках схемы квантованного фильтра `hd`, чтобы при обработке сигнала `x` не возникало переполнений и обеспечивалась максимально возможная точность вычислений. Полный размер машинных слов не меняется, регулируется только размер дробной части. Если не указать выходной параметр `hd1`, настройки будут произведены в исходном объекте `hd`;
- `hd1 = double(hd)` — преобразует квантованный фильтр в неквантованный, т. е. выполняющий вычисления в формате `double`. В отличие от функции `reffilter`, коэффициенты полученного фильтра будут иметь значения *квантованных* коэффициентов фильтра `hd`;
- `[h, w] = freqrespest(hd, L)` — функция оценки частотной характеристики квантованного фильтра `hd`. Измерение производится путем пропускания через фильтр сигнала, представляющего собой сумму синусоид с одинаковыми амплитудами, равномерно расположеннымми частотами и случайными начальными фазами. Второй входной параметр задает число таких испытаний (их результаты усредняются). Выходные параметры `h` и `w` содержат соответственно вектор полученных оценок частотной характеристики и вектор частот, для которых эти оценки получены. При отсутствии выходных параметров выводится график АЧХ в окне среды FVTool;
- `limitcycle` — функция анализа предельных циклов в квантованных фильтрах. Она подробно рассмотрена далее;
- `hpsd = noisepsd(hd, L)` — функция оценки СПМ собственного шума квантованного фильтра `hd`. Она пропускает через фильтр тот же сигнал, что использует функция `freqrespest`, и вычисляет разность между получившейся СПМ и СПМ, рассчитанной исходя из частотной характеристики фильтра. Второй входной параметр задает число таких испытаний (их результаты усредняются). Результатом работы функции является объект класса `dspdata.psd` (см. разд. "Хранение и отображение спектральных данных" главы 5). При отсутствии выходных параметров выводится график СПМ шума в окне среды FVTool;

- `g = normalize(hd)` — приведение коэффициентов нерекурсивной части (т. е. числителя функции передачи) квантованного фильтра `hd` к диапазону  $\pm 1$ . Результатом работы функции является общий множитель `g`, "вынесенный за скобки" в результате указанной нормировки;
- `denormalize(hd)` — обращает действие функции `normalize`, возвращая коэффициенты фильтра `hd` к прежним значениям;
- `s = qreport(hd)` — возвращает структуру, содержащую статистическую информацию о работе квантователей, входящих в состав фильтра (максимальное и минимальное значения входного сигнала, число переполнений, а также количество выполненных квантований). При отсутствии выходного параметра информация выводится на экран;

### **ВНИМАНИЕ!**

По умолчанию сбор статистической информации не ведется, поэтому для использования функции `qreport` необходимо до начала работы фильтра включить *протоколирование (logging) работы квантователей* с помощью функции `fipref` пакета Fixed-Point: `fipref('LoggingMode', 'on')`.

- `hd1 = reffilter(hd)` — возвращает объект исходного (неквантованного) фильтра, из которого был получен квантованный фильтр `hd`. В отличие от функции `double`, коэффициенты полученного фильтра будут иметь значения, которые были у коэффициентов фильтра `hd` *перед выполнением их квантования*;
- `g = set2int(hd)` — преобразует коэффициенты *нерекурсивного* квантованного фильтра `hd` к целочисленным значениям, задавая для них нулевую длину дробной части. Возвращаемый результат `g` представляет собой использованный при этом коэффициент масштабирования, всегда равный некоторой степени двойки.

В качестве примера рассчитаем эллиптический ФНЧ 5-го порядка, создадим для него объект квантованного фильтра и произведем с помощью этого фильтра обработку последовательности прямоугольных импульсов.

Пусть ФНЧ имеет частоту среза, равную 20% от частоты Найквиста, пульсации АЧХ в полосе пропускания 1 дБ и ослабление в полосе задерживания 40 дБ:

```
>> [b, a] = ellip(5, 1, 40, 0.2)
b =
    0.0153   -0.0221    0.0154    0.0154   -0.0221    0.0153
a =
    1.0000   -3.9043    6.5819   -5.8940    2.7935   -0.5599
```

Создаем объект дискретного фильтра:

```
>> hd = dfilt.df2t(b, a)
hd =
    FilterStructure: 'Direct-Form II Transposed'
    Arithmetic: 'double'
    Numerator: [1x6 double]
    Denominator: [1x6 double]
    PersistentMemory: false
```

### ЗАМЕЧАНИЕ

Мы использовали стандартную для MATLAB форму реализации фильтра — транспонированную (Direct Form II Transposed, см. рис. 4.15).

Как видите, по умолчанию фильтр работает в формате `double`. Превратим его в фильтр с фиксированной запятой, задав соответствующее значение для свойства `Arithmetic`:

```
>> hd.Arithmetic = 'fixed'  
hd =  
    FilterStructure: 'Direct-Form II Transposed'  
    Arithmetic: 'fixed'  
    Numerator: [1x6 double]  
    Denominator: [1x6 double]  
    PersistentMemory: false  
  
    CoeffWordLength: 16  
    CoeffAutoScale: true  
        Signed: true  
  
    InputWordLength: 16  
    InputFracLength: 15  
  
    OutputWordLength: 16  
    OutputFracLength: 15  
  
    StateWordLength: 16  
    StateAutoScale: true  
  
    ProductMode: 'FullPrecision'  
  
    AccumMode: 'KeepMSB'  
    AccumWordLength: 40  
    CastBeforeSum: true  
  
    RoundMode: 'convergent'  
    OverflowMode: 'wrap'
```

Как видите, у фильтра появилось большое количество новых свойств, относящихся к форматам представления различных чисел, участвующих в вычислениях.

Теперь построим графики АЧХ и ФЧХ квантованного фильтра. Для этого воспользуемся функцией `freqz` (рис. 7.17):

```
>> freqz(hd)
```

На графиках, которые для объектов класса `dfilt` выводятся в окне визуализатора фильтров FVTool (см. главу 4), штрихпунктирными линиями отображаются характеристики исходного фильтра, а сплошными — квантованного. Однако в данном

случае благодаря довольно высокой (16 двоичных разрядов) точности представления коэффициентов и оптимизации, по умолчанию выполняемой при их квантовании, разница характеристик практически незаметна.

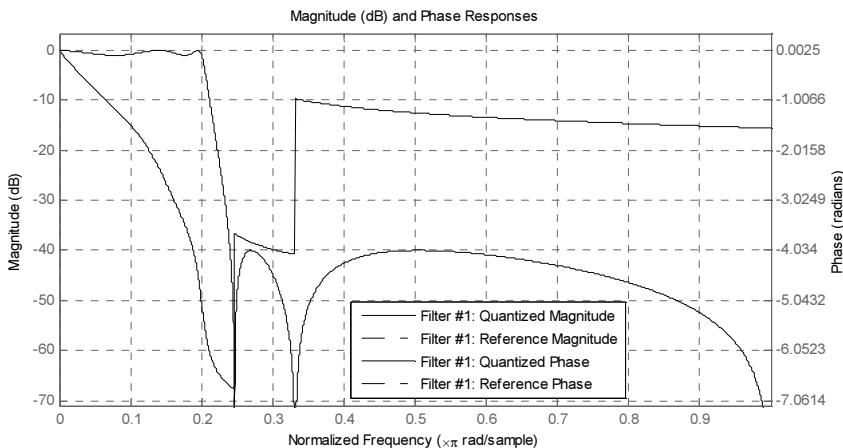


Рис. 7.17. АЧХ и ФЧХ квантованного эллиптического ФНЧ 5-го порядка

Далее создаем входной сигнал в виде последовательности прямоугольных импульсов с амплитудой 0,8:

```
>> x = 0.8 * [ones(1,100) -ones(1,100) ones(1,100)];
```

Чтобы пропустить этот сигнал через исходный (неквантованный) фильтр, воспользуемся функцией `filter`:

```
>> y = filter(b, a, x);
```

Та же функция используется и для обработки сигнала квантованным фильтром:

```
>> y_q = hd.filter(x);
```

Наконец, строим графики выходного сигнала для исходного (рис. 7.18, *a*) и квантованного (рис. 7.18, *б*) фильтров:

```
>> plot(y)
>> figure
>> plot(y_q)
```

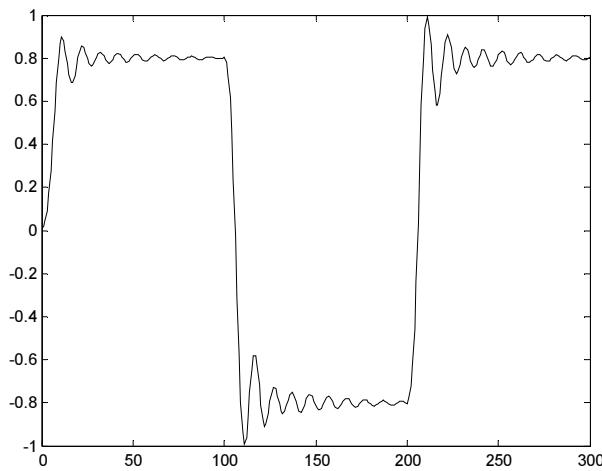
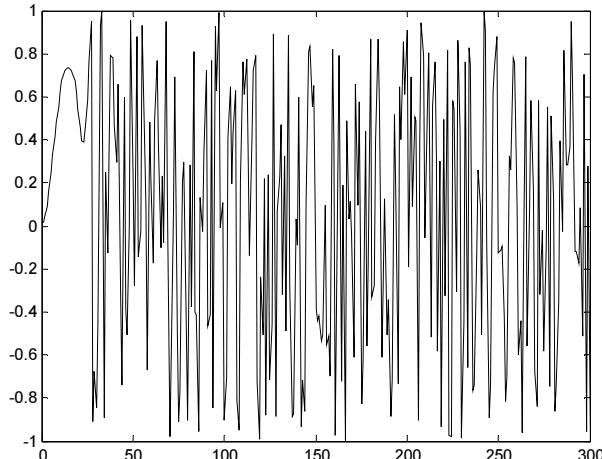
Сравнение верхнего и нижнего графиков на рис. 7.18 показывает, что, хотя абсолютные значения сигнала на выходе исходного фильтра не превосходят единицы, выходной сигнал квантованного фильтра похож на исходный только в течение нескольких первых отсчетов, а после этого квантованный фильтр, как говорится, "идет вразнос". Чтобы понять причины этого, повторим обработку сигнала, предварительно включив протоколирование (logging) работы квантователей с помощью функции `fipref` пакета Fixed-Point:

```
>> fipref('LoggingMode', 'on');
>> y_q = hd.filter(x);
```

Теперь вызовем отчет о работе квантователей фильтра с помощью функции `qreport`:

```
>> qreport(hd)
ans =
```

Fixed-Point Report					
	Min	Max		Range	N of Overflows
Input:	-0.79998779	0.79998779		-1 0.99996948	0/300 (0%)
Output:	-0.99664307	0.99664307		-1 0.99996948	227/300 (76%)
States:	-13.492188	14.617188		-256 255.99219	0/1500 (0%)
Num Prod:	-0.017662542	0.017662542		-0.0625 0.0625	0/1800 (0%)
Den Prod:	-6.5599358	6.5599358		-16 16	0/1500 (0%)
Num Acc:	-13.4786	14.629442		-256 256	0/1500 (0%)
Den Acc:	-13.493492	14.614632		-256 256	0/1500 (0%)

*a**б*

**Рис. 7.18.** Выходной сигнал неквантованного (*а*) и квантованного (*б*) фильтров

Отчет показывает, что при квантовании выходного сигнала (а ведь он попадает в цепи обратной связи рекурсивного фильтра и используется при дальнейших вычислениях!) произошло большое количество переполнений. Это связано с тем, что оптимизация при преобразовании фильтра к формату с фиксированной запятой производится только в отношении представления коэффициентов фильтра. Так как свойства обрабатываемого сигнала заранее неизвестны, невозможно и провести оптимизацию форматов представления промежуточных результатов. Произведем автоматическое масштабирование фильтра, предъявив в качестве образца наш входной сигнал  $x$ :

```
>> hd1 = autoscale(hd, x)
hd1 =
    FilterStructure: 'Direct-Form II Transposed'
    Arithmetic: 'fixed'
    Numerator: [1x6 double]
    Denominator: [1x6 double]
    PersistentMemory: false

    CoeffWordLength: 16
    CoeffAutoScale: false
    NumFracLength: 20
    DenFracLength: 12
    Signed: true

    InputWordLength: 16
    InputFracLength: 15

    OutputWordLength: 16
    OutputFracLength: 15

    StateWordLength: 16
    StateAutoScale: false
    StateFracLength: 13

    ProductMode: 'SpecifyPrecision'
    ProductWordLength: 32
    NumProdFracLength: 35
    DenProdFracLength: 27

    AccumMode: 'SpecifyPrecision'
    AccumWordLength: 40
    NumAccumFracLength: 35
    DenAccumFracLength: 27
    CastBeforeSum: true

    RoundMode: 'convergent'
    OverflowMode: 'wrap'
```

Как видите, часть свойств нового фильтра имеет другие значения, выбранные исходя из свойств сигнала  $x$ . Обработаем сигнал новым фильтром и выведем отчет:

```
>> y_q1 = hd1.filter(x);
>> qreport(hd1)
```

ans =

Fixed-Point Report

	Min	Max		Range		N of Overflows
Input:	-0.79998779	0.79998779		-1 0.99996948		0/300 (0%)
Output:	-0.99078369	0.99835205		-1 0.99996948		0/300 (0%)
States:	-3.6672363	3.6939697		-4 3.9998779		0/1500 (0%)
Num Prod:	-0.017662542	0.017662542		-0.0625 0.0625		0/1800 (0%)
Den Prod:	-6.5213692	6.5711844		-16 16		0/1500 (0%)
Num Acc:	-3.6795477	3.7062811		-16 16		0/1500 (0%)
Den Acc:	-6.5213692	6.5711844		-4096 4096		0/1500 (0%)

Как видите, сообщения о переполнениях исчезли. Выведя график сигнала  $y_{q1}$ , можно убедиться, что он не имеет заметных искажений. Измерим максимальную разницу между "образцовым" выходным сигналом  $y$  и новым выходным сигналом квантованного фильтра  $y_{q1}$ :

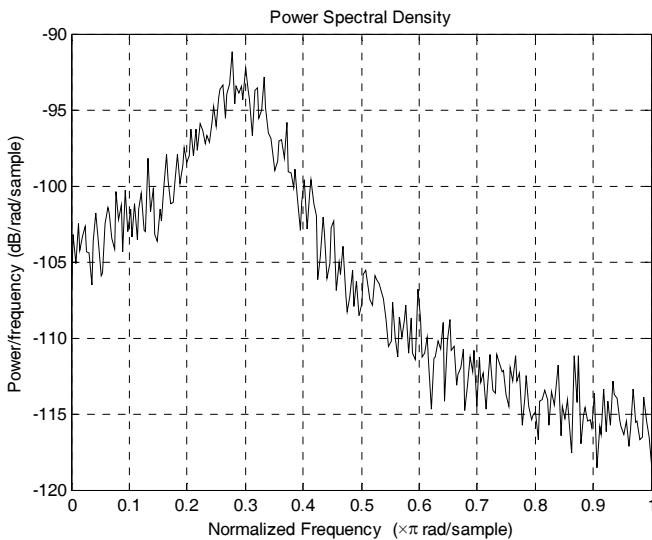
```
>> max(abs(y - double(y_q1)))
ans =
0.0143
```

Таким образом, погрешность вычислений в данном случае составила примерно 1,5% от амплитуды выходного сигнала. Дополнительно уменьшить ее можно, преобразовав фильтр в набор каскадов второго порядка.

В заключение данного раздела приведем прием использования функции `noisepsd`, оценив с ее помощью СПМ собственного шума резонатора второго порядка, реализованного в прямой форме (ее график, полученный аналитически, был показан на рис. 7.12):

```
>> [b, a] = iirpeak(0.3, 0.1); % расчет фильтра
>> hd = dfilt.df1(b, a); % создание объекта
>> hd.Arithmetic = 'fixed'; % переход к фиксированной запятой
>> hpsd = hd.noisepsd; % оценка СПМ собственного шума
>> plot(hpsd) % вывод графика
```

Результат работы приведенного кода показан на рис. 7.19. Из графика видно, что форма кривой очень хорошо совпадает с аналитически предсказанной (см. рис. 7.12).



**Рис. 7.19.** Оценка СПМ собственных шумов резонатора второго порядка, реализованного в прямой форме

## Анализ предельных циклов

Для проверки возможности возникновения предельных циклов в квантованном фильтре служит функция `limitcycle`. Синтаксис ее вызова следующий:

```
report = limitcycle(hd, N, L, Stop)
```

Единственным обязательным входным параметром здесь является `hd` — объект квантованного фильтра. Остальные входные параметры имеют значения по умолчанию.

Сущность работы функции `limitcycle` состоит в задании случайных начальных состояний фильтра и анализе его поведения при нулевом входном сигнале. Число испытаний задается параметром `N` и по умолчанию равно 20.

Параметр `L` задает продолжительность анализа: длительность нулевого входного сигнала в `L` раз превышает длину импульсной характеристики фильтра, рассчитываемую функцией `impzlength`. По умолчанию значение `L` равно 2.

Строковый параметр `Stop` позволяет управлять критерием остановки работы функции. Возможны следующие значения:

- 'granular' — остановка работы производится при обнаружении предельного цикла низкого уровня;
- 'overflow' — остановка работы производится при обнаружении предельного цикла высокого уровня;
- 'either' — остановка работы производится при обнаружении предельного цикла любого типа (этот вариант используется по умолчанию).

Функция возвращает структуру `report`, содержащую следующие поля с информацией о результатах анализа:

- `LimitCycle` — строка, показывающая тип обнаруженного предельного цикла и равная '`granular`', '`overflow`' или '`none`';
- `Zi` — вектор начального состояния фильтра, вызвавшего предельный цикл;
- `Output` — отсчеты выходного сигнала фильтра в установившемся состоянии предельного цикла;
- `Trial` — номер испытания, на котором обнаружен предельный цикл.

Если в процессе работы функции были найдены предельные циклы как низкого, так и высокого уровня, возвращаются сведения о цикле высокого уровня.

## Расширение программы FDATool

Пакет Filter Design расширяет графическую среду расчета и анализа фильтров FDATool, добавляя в нее возможность описания и анализа квантованных фильтров.

Для выполнения квантования фильтра необходимо перейти на вкладку **Set Quantization Parameters**, щелкнув на соответствующей кнопке в левой части окна (см. рис. 6.16 в главе 6). На вкладке имеется раскрывающийся список **Filter arithmetic** (рис. 7.20), предлагающий три варианта: **Double-precision floating-point**, **Single-precision floating-point** и **Fixed-point**. В первых двух случаях, задающих вычисления с плавающей запятой в форматах `double` (64 бита) и `single` (32 бита), дополнительных элементов управления на вкладке не имеется. При выборе третьего варианта, соответствующего вычислениям с фиксированной запятой, вкладка приобретает вид, показанный на рис. 7.20.

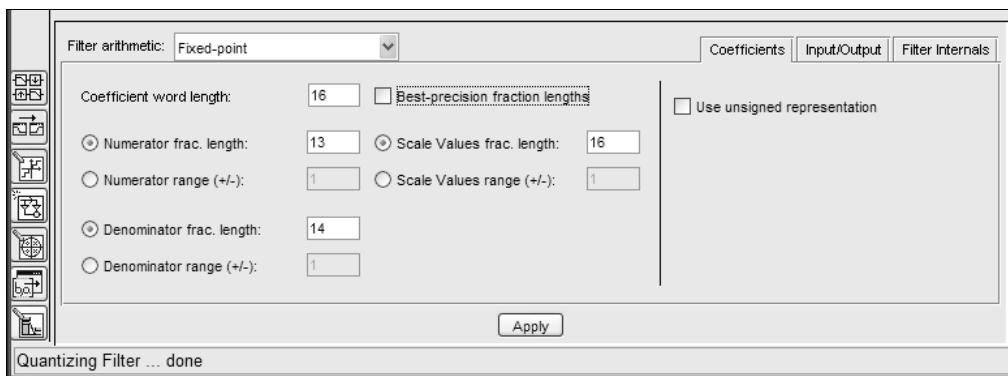
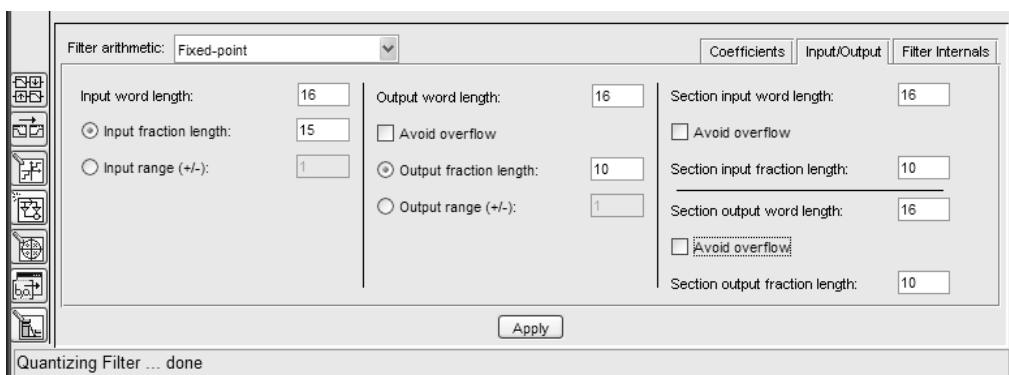


Рис. 7.20. Вкладка задания параметров квантования коэффициентов фильтра

На вкладке появляются три дополнительных вкладки: **Coefficients**, **Input/Output** и **Filter Internals**. Их вид зависит от формы реализации фильтра; рисунки, приведенные в данном разделе, соответствуют каскадной реализации фильтра в виде набора секций второго порядка.

Вкладка **Coefficients** (см. рис. 7.20) отвечает за формат представления коэффициентов фильтра. В поле **Coefficient word length** задается общее число битов для представления коэффициентов. Если установлен флагок **Best-precision fraction lengths**, число битов для дробной части коэффициентов выбирается автоматически. Если этот флагок снят, становятся доступны остальные элементы управления. Для коэффициентов числителя (**Numerator**), знаменателя (**Denominator**) и коэффициентов усиления секций (**Scale Values**) можно задавать либо число битов дробной части (**frac. length**), либо диапазон представимых чисел (**range (+/-)**). Флагок **Use unsigned representation** включает режим беззнакового представления коэффициентов. Коэффициенты, имевшие отрицательные значения, при этом становятся равными нулю.

Вкладка **Input/Output** (рис. 7.21) определяет формат представления входного и выходного сигналов фильтра, а также входных и выходных сигналов отдельных секций фильтра, если таковые имеются.

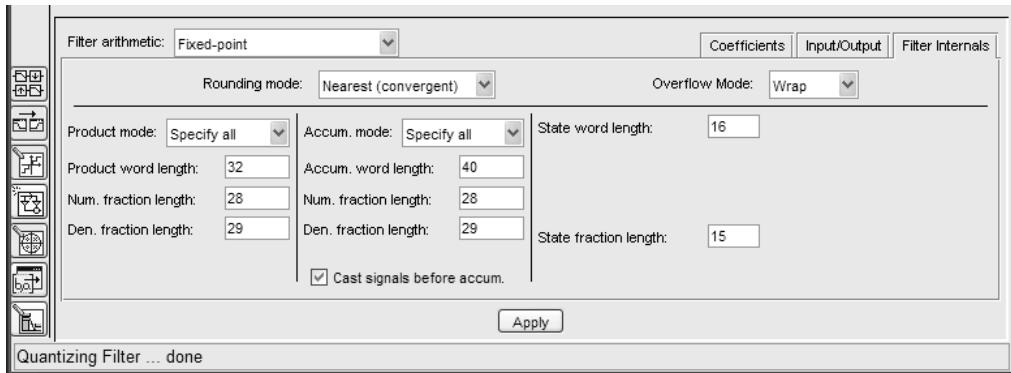


**Рис. 7.21.** Вкладка задания параметров квантования входного и выходного сигналов фильтра

Поля **Input word length** и **Output word length** задают общее число битов для представления отсчетов соответственно входного и выходного сигналов, а поля **Section input word length** и **Section output word length** отвечают за входные и выходные сигналы отдельных секций. Способ управления размером дробной части аналогичен тому, как это делается для коэффициентов фильтра, однако режим автоматического выбора включается с помощью флагков **Avoid overflow**, индивидуальных для каждого представленного на вкладке сигнала (кроме входного, для которого размер дробной части нужно обязательно указывать вручную).

Вкладка **Filter Internals** (рис. 7.22) определяет формат представления промежуточных результатов вычислений в различных точках схемы фильтра.

Два раскрывающихся списка в верхней части вкладки задают режимы округления (список **Rounding mode**) и обработки переполнений (список **Overflow mode**). Возможные варианты для этих двух режимов такие же, как для свойств **RoundMode** и **OverflowMode** объектов квантователей (см. табл. 7.4).



**Рис. 7.22.** Вкладка задания параметров квантования промежуточных результатов

Группа элементов управления в левой части вкладки задает формат представления результатов умножений. Раскрывающийся список **Product mode** соответствует свойству **ProductMode** объектов квантованных фильтров (см. ранее в этой главе). При выборе варианта **Full precision** формат произведений задается автоматически. Если выбрать вариант **Keep LSB** или **Keep MSB**, станет доступным поле **Product word length**, определяющее число битов для представления результатов умножения. Наконец, если выбрать вариант **Specify all**, можно будет задать число битов дробной части, причем отдельно для числителя (**Num. fraction length**) и знаменателя (**Den. fraction length**).

Следующая группа элементов устроена аналогично, но относится к формату регистра-аккумулятора, предназначенного для накопления результатов при суммировании чисел. Единственное отличие — флажок **Cast signals before accum.**. Если он установлен, суммируемые сигналы будут предварительно преобразовываться в формат, заданный для аккумулятора.

Наконец, поля ввода в правой части вкладки задают формат хранения данных в элементах памяти фильтра: **State word length** — общее количество битов, **State fraction length** — количество битов в дробной части.

Как уже говорилось в этой главе, результаты квантования дискретного фильтра сильно зависят от формы его реализации (см. табл. 4.2). Чтобы сменить форму реализации фильтра, необходимо щелкнуть правой кнопкой мыши в области **Current Filter Information** основного окна программы FDA Tool и выбрать в появившемся контекстном меню команду **Convert structure**. Кроме того, в этом контекстном меню имеется команда **Show filter structure**, при выборе которой открывается окно справочной системы, содержащее информацию о конструкторе объекта дискретного фильтра, соответствующем текущей форме реализации, где приведена в том числе и структурная схема.

После задания всех параметров квантования необходимо щелкнуть на кнопке **Apply**, расположенной в нижней части вкладки. В результате будут рассчитаны характеристики квантованного фильтра. На графиках характеристики исходного фильтра выводятся пунктирными линиями, а квантованного — сплошными (см. рис. 7.23).

Для квантованных фильтров становятся доступными две крайние справа кнопки панели инструментов среды FDATool — **Magnitude Response Estimate** и **Round-off Noise Power Spectrum** (см. рис. 6.17), с помощью которых можно вывести на экран графики оценки частотной характеристики квантованного фильтра и СПМ его собственного шума (соответствующие расчеты выполняются с помощью рассмотренных ранее в этой главе функций `freqrespest` и `noisepsd`).

Квантованный фильтр можно экспорттировать из программы FDATool обычным способом — с помощью команды **Export** меню **File**. Данные экспортируются в виде объекта квантованного фильтра, описанного ранее. Таким образом, среда FDATool предоставляет удобный графический интерфейс для настройки свойств объекта квантованного фильтра.

## Квантованное БПФ

Для создания объекта квантованного БПФ служит функция `qfft`. В простейшем случае она может вызываться без входных параметров, но поскольку обычно требуется выполнять БПФ заданной размерности, удобнее всего задавать размерность БПФ сразу же при создании объекта. Для этого нужно использовать в качестве параметров функции-конструктора строку '`length`' и значение размерности:

```
f = qfft('length', N)
```

Здесь `N` — размерность БПФ.

Для задания свойств объекта (их список будет приведен далее) можно указывать при вызове функции `qfft` параметры в виде пар "имя свойства — значение свойства":

```
f = qfft('name1', value1, 'name2', value2, ...)
```

Здесь '`name1`', '`name2`' и т. д. — имена свойств, а `value1`, `value2` и т. д. — соответствующие значения.

Особо отметим еще один вариант вызова, позволяющий задать общий формат квантования для всех величин при вычислении БПФ. Для этого в списке параметров нужно задать значение свойства '`quantizer`':

```
f = qfft(..., 'quantizer', [M N], ...)
```

В результате для всех величин будет использован формат с фиксированной запятой с длиной слова `M` бит и `N` битами после запятой.

Список свойств квантованного БПФ приведен в табл. 7.5. Считывать и задавать значения этих свойств можно с помощью функций `get` и `set` соответственно (см. разд. "Общая информация об объектах MATLAB" главы 4).

**Таблица 7.5.** Свойства объекта квантованного БПФ

Имя свойства	Описание
<code>CoefficientFormat</code>	Объект <code>quantizer</code> , задающий формат квантования коэффициентов (twiddle factors), используемых при вычислении БПФ

Таблица 7.5 (окончание)

Имя свойства	Описание
InputFormat	Объект <code>quantizer</code> , задающий формат квантования входного сигнала
Length	Размерность БПФ. Она должна являться степенью основания БПФ (см. далее описание свойства <code>Radix</code> ). Значение по умолчанию равно 16
NumberOfSections	Число ступеней БПФ, равное $\log_2(\text{Length}) / \text{Log}_2(\text{Radix})$ (только для чтения)
MultiplicandFormat	Объект <code>quantizer</code> , задающий формат квантования чисел, умножаемых на коэффициенты БПФ
OptimizeUnityGains	Это свойство не отображается в явном виде при выводе информации об объекте на экран, поскольку ему не соответствует поле структуры объекта. Однако значение этого свойства можно считывать и задавать (возможные варианты — 'on' и принятое по умолчанию 'off'). Если установлено состояние 'on', при вычислениях будут исключены умножения на коэффициенты, равные единице. Кроме того, при этом для квантования коэффициентов вместо объекта <code>quantizer</code> будет использован объект <code>unitquantizer</code> , что позволяет <i>точно</i> представлять единичные значения
OutputFormat	Объект <code>quantizer</code> , задающий формат квантования выходного сигнала
ProductFormat	Объект <code>quantizer</code> , задающий формат квантования результатов операций умножения
Radix	Основание БПФ, которое может быть равно 2 (по умолчанию) или 4
ScaleValues	Коэффициенты масштабирования сигналов. Значение этого свойства может быть либо числом (масштабируется только входной сигнал), либо вектором длиной <code>NumberOfSections</code> (масштабируется входной сигнал каждой ступени БПФ)
SumFormat	Объект <code>quantizer</code> , задающий формат квантования результатов операций сложения

Вычисление квантованного БПФ осуществляется с помощью функций `fft` (прямое БПФ) и `ifft` (обратное БПФ). Отличие от обычного варианта использования этих функций состоит в том, что в начало списка параметров добавляется объект квантованного БПФ:

```
y = fft(f, x)
x = ifft(f, y)
```

Здесь `f` — объект квантованного БПФ, `x` — сигнал во временной области, `y` — сигнал в частотной области.

Если размерность неквантованного БПФ в MATLAB подстраивается под длину сигнала, то в случае квантованного БПФ все наоборот — сигнал дополняется нулями или усекается, чтобы его длина стала равна размерности БПФ.

Помимо функций `fft` и `ifft` с объектами квантованных БПФ могут работать следующие функции:

- `f1 = copyobj(f)` — создает *независимую копию* объекта квантованного БПФ с теми же значениями свойств, что у исходного объекта `f` (см. приведенный в главе 4 комментарий к аналогичной по смыслу функции `copy` для объектов дискретных фильтров);
- `disp(f)` — выводит на экран свойства объекта;
- `eps(f)` — выводит на экран относительную точность представления чисел в объекте квантованного БПФ;
- `n = length(f)` — возвращает размерность БПФ (значение свойства `Length`);
- `noperations(f)` — выводит на экран сведения о числе операций квантования, выполненных различными квантователями объекта квантованного БПФ `f`;
- `noverflows(f)` — выводит на экран сведения о числе переполнений, возникших при выполнении операций квантования различными квантователями объекта квантованного БПФ `f`;
- `opt = optimizeunitygains(f)` — возвращает значение свойства `OptimizeUnityGains`;
- `qreport(f)` — выводит на экран статистическую информацию о работе квантователей, входящих в состав объекта квантованного БПФ (максимальное и минимальное значения входного сигнала, число переполнений и случаев потери значимости, а также количество выполненных квантований);
- `s = qreport(f)` — возвращает ту же информацию в полях структуры `s`;
- `r = radix(f)` — возвращает основание БПФ (значение свойства `Radix`);
- `range(f)` — отображает сведения о минимальных и максимальных значениях выходного квантованного сигнала для квантователей, входящих в состав объекта квантованного БПФ `f`;
- `[R1, R2, ...] = range(f, T1, T2, ...)` — возвращает двухэлементные векторы `R1`, `R2` и т. д., содержащие минимальные и максимальные значения выходного квантованного сигнала для квантователей, входящих в состав объекта квантованного БПФ `f` и указанных входными строковыми параметрами `T1`, `T2` и т. д. Возможные значения этих параметров — `'coefficient'`, `'input'`, `'output'`, `'multiplicand'`, `'product'` и `'sum'`;
- `reset(f)` — сбрасывает в исходное состояние значения свойств, связанных со сбором статистической информации (`Max`, `Min`, `Noperations`, `Noverflows` и `Nunderflows`), для всех квантователей, входящих в состав объекта квантованного БПФ `f`;
- `s = tostring(f)` — возвращает строку вызова функции `qfft`, в результате выполнения которой будет создан объект, совпадающий с объектом `f`;

- `w = twiddles(f)` — возвращает квантованные значения коэффициентов (комплексных экспонент), используемых объектом `f` при вычислении БПФ.

В качестве примера создадим объект квантованного БПФ размерности 256 и затем вычислим спектр синусоидального сигнала посредством обычной функции `fft` и с помощью квантованного объекта.

При создании объекта необходимо задать только размерность БПФ, для всех остальных свойств мы оставляем значения по умолчанию:

```
>> f = qfft('length', 256)
f =
    Radix = 2
    Length = 256
CoefficientFormat = quantizer('fixed', 'round', 'saturate', [16 15])
    InputFormat = quantizer('fixed', 'floor', 'saturate', [16 15])
    OutputFormat = quantizer('fixed', 'floor', 'saturate', [16 15])
MultiplicandFormat = quantizer('fixed', 'floor', 'saturate', [16 15])
    ProductFormat = quantizer('fixed', 'floor', 'saturate', [32 30])
    SumFormat = quantizer('fixed', 'floor', 'saturate', [32 30])
NumberOfSections = 8
ScaleValues = [1]
```

Пусть обрабатываемый сигнал имеет такую частоту, что анализируемый фрагмент (256 отсчетов) содержит нецелое число периодов:

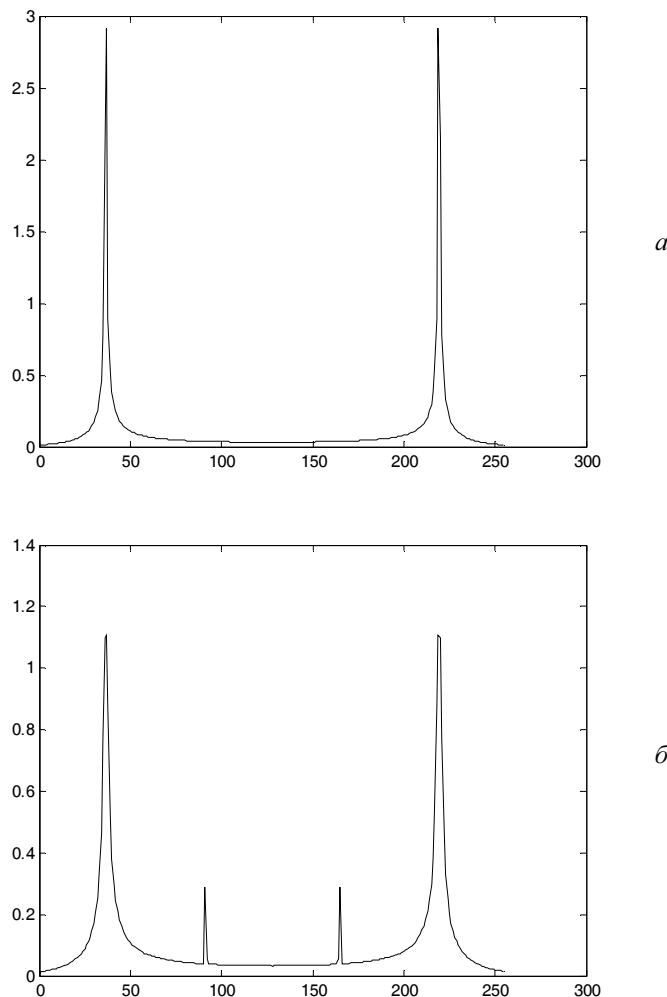
```
>> t = 0:255; % дискретное время
>> s = 1/32*cos(2*pi*t/7); % период сигнала равен 7 отсчетам
```

Теперь рассчитываем два варианта спектра: с помощью обычного (вектор `sp`) и квантованного (вектор `sp_q`) БПФ и выводим графики их модулей (рис. 7.23):

```
>> sp = fft(s); % обычное БПФ
>> sp_q = fft(f, s); % квантованное БПФ
Warning: 2062 overflows in quantized fft.
> In @qfft\private\privfft at 118
In qfft.fft at 35
      Max      Min      NOverflows      NUnderflows      NOperations
Coefficient      1      -1          8              7             510
      Input  0.03125 -0.02816          0              0             256
      Output      2      -2          4              0             512
Multiplicand     1.406   -1.406        2054             42            8192
      Product     0.9999      -1          0              0            8192
      Sum          2      -2          4              0            10240
>> plot(t, abs(sp));
>> figure
>> plot(t, abs(sp_q))
```

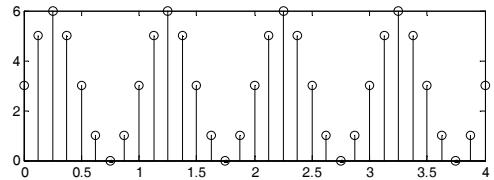
Сравнение верхнего и нижнего графиков на рис. 7.23 показывает, что переполнения, о которых сообщил при вычислениях объект `f`, весьма существенно исказили

результаты расчета — в спектре появились ложные составляющие. Это свидетельствует о необходимости либо уменьшить уровень входного сигнала (в данном случае, чтобы избавиться от ложных пиков, необходимо уменьшить амплитуду гармонического сигнала с  $1/32$  до  $1/46$ , хотя сообщения о переполнении множителей при этом не исчезают), либо использовать в объекте  $f$  квантователи с большей верхней границей диапазона представимых чисел.



**Рис. 7.23.** Результаты вычисления спектра синусоидального сигнала с помощью обычного (a) и квантованного (б) БПФ

# ГЛАВА 8



## Модуляция и демодуляция

При создании систем передачи информации в большинстве случаев оказывается, что спектр исходного сигнала, подлежащего передаче, сосредоточен отнюдь не на тех частотах, которые эффективно пропускает имеющийся канал связи. Кроме того, очень часто необходимо в одном и том же канале связи передавать несколько сигналов одновременно. Одним из способов решения этой задачи является использование *частотного разделения каналов*, при котором разные сигналы занимают неперекрывающиеся полосы частот.

Далее, во многих случаях требуется, чтобы передаваемый сигнал был *узкополосным*. Это означает, что эффективная ширина спектра намного меньше его центральной частоты:

$$\Delta f \ll f_0.$$

Перечисленные причины приводят к необходимости такой трансформации исходного сигнала, чтобы требования, предъявляемые к занимаемой сигналом полосе частот, были выполнены, а сам исходный сигнал можно было восстановить.

Решение указанной проблемы достигается при использовании *модуляции* (*modulation*), сущность которой заключается в следующем. Формируется некоторое колебание (чаще всего гармоническое), называемое *несущим колебанием* или просто *несущей* (*carrier*), и какой-либо из параметров этого колебания изменяется во времени пропорционально исходному сигналу. Исходный сигнал называют *модулирующим* (*modulating signal*), а результирующее колебание с изменяющимися во времени параметрами — *модулированным сигналом* (*modulated signal*). Обратный процесс — выделение модулирующего сигнала из модулированного колебания — называется *демодуляцией* (*demodulation*).

Запишем (в очередной раз) гармонический сигнал общего вида:

$$s(t) = A \cos(\omega_0 t + \phi_0).$$

У данного сигнала есть три параметра: амплитуда  $A$ , частота  $\omega_0$  и начальная фаза  $\phi_0$ . Каждый из них можно связать с модулирующим сигналом, получив таким образом три основных вида модуляции: амплитудную, частотную и фазовую. Как мы

увидим далее, частотная и фазовая модуляция очень тесно взаимосвязаны, поскольку обе они влияют на аргумент функции  $\cos$ . Поэтому эти два вида модуляции имеют общее название — *угловая модуляция*.

В современных системах передачи цифровой информации также получила распространение *квадратурная модуляция*, при которой одновременно изменяются амплитуда и фаза сигнала.

Все упомянутые виды модуляции будут более подробно рассмотрены в следующих разделах.

В MATLAB функции, реализующие процедуры модуляции и демодуляции, имеются в двух пакетах расширения — *Signal Processing* и *Communications*. Возможности этих пакетов несколько различаются — как по набору реализованных методов модуляции/демодуляции, так и по степени контроля над параметрами модулированного сигнала.

В процессе изложения теоретического материала демонстрационные примеры будут реализовываться с помощью базовых функций MATLAB, без использования специальных средств модуляции/демодуляции из упомянутых пакетов. Это позволит читателю лучше понять сущность рассматриваемых методов и алгоритмов.

## Амплитудная модуляция

Как явствует из названия, при амплитудной модуляции (*AM*; английский термин — *amplitude modulation, AM*) в соответствии с модулирующим сигналом изменяется амплитуда несущего колебания:

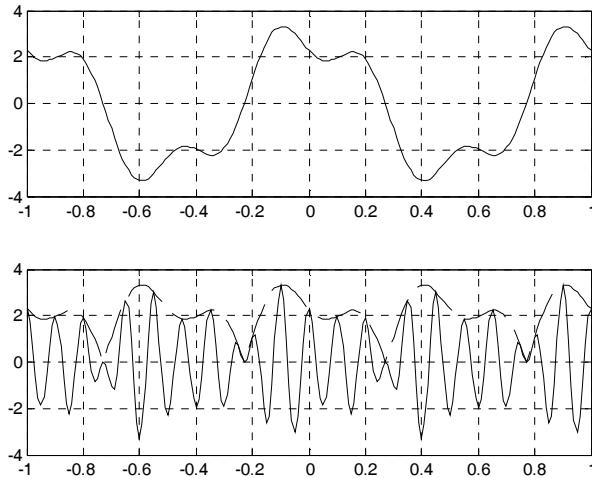
$$s_{\text{AM}}(t) = A(t) \cos(\omega_0 t + \phi_0).$$

Однако если амплитуду  $A(t)$  просто сделать прямо пропорциональной модулирующему сигналу, возможно возникновение следующей проблемы. Как правило, модулирующий сигнал является двуполярным (знакопеременным). Рассмотрим, например, такой сигнал (рис. 8.1, сверху):

$$s_M(t) = 3 \cos(2\pi t) - \sin(6\pi t + \pi/4).$$

Если мы непосредственно используем его в качестве амплитудной функции  $A(t)$ , получится следующее (рис. 8.1, снизу):

```
>> t = -1:0.01:1;
>> s_M = 3 * cos(2*pi*t) - sin(6*pi*t+pi/4);
>> Fc = 10; % несущая частота
>> s_AM = s_M .* cos(2*pi*Fc*t);
>> subplot(2, 1, 1)
>> plot(t, s_M)
>> grid on
>> subplot(2, 1, 2)
>> plot(t, s_AM, t, abs(s_M), '--')
>> grid on
```

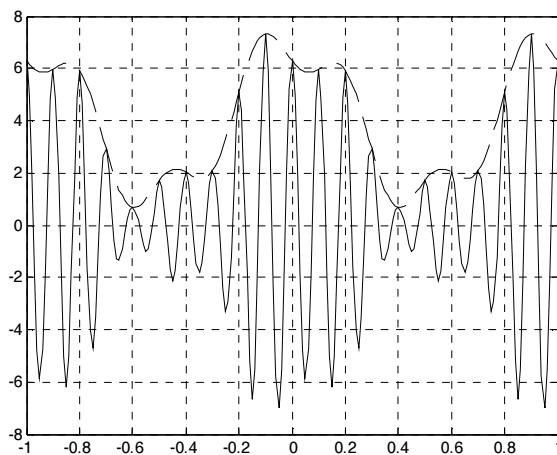


**Рис. 8.1.** Умножение двуполярного модулирующего сигнала (сверху) на несущее колебание дает неправильную амплитудную огибающую (снизу)

Из нижнего графика рис. 8.1 видно, что амплитудная огибающая, которая будет выделена в процессе демодуляции, в данном случае оказывается неправильной — она соответствует *модулю* исходного сигнала.

Поэтому при реализации АМ к модулирующему сигналу предварительно добавляют постоянную составляющую, чтобы сделать его однополярным:

$$A(t) = A_0 + k s_M(t).$$



**Рис. 8.2.** Добавление постоянной составляющей делает модулирующий сигнал однополярным

Для нашего примера достаточно будет постоянной составляющей, равной четырем (рис. 8.2):

```
>> s_AM = (4 + s_M) .* cos(2*pi*Fc*t);
>> plot(t, s_AM, t, 4+s_M, '--')
>> grid on
```

Теперь форма амплитудной огибающей соответствует модулирующему сигналу с точностью до постоянной составляющей, которая легко может быть удалена после демодуляции.

Итак, окончательно можно записать АМ-сигнал в следующем виде:

$$s_{AM}(t) = (A_0 + k s_M(t)) \cos(\omega_0 t + \phi_0). \quad (8.1)$$

## Однотональная АМ

Для понимания сути амплитудной модуляции и спектральной структуры АМ-сигнала полезно подробнее рассмотреть частный случай, когда модулирующий сигнал является гармоническим:

$$s_M(t) = A_M \cos(\Omega t + \Phi_0).$$

$$s_{AM}(t) = (A_0 + A_M \cos(\Omega t + \Phi_0)) \cos(\omega_0 t + \phi_0). \quad (8.2)$$

Отношение между амплитудами модулирующего сигнала  $A_M$  и несущего колебания  $A_0$  называется *коэффициентом модуляции* или *глубиной модуляции*:

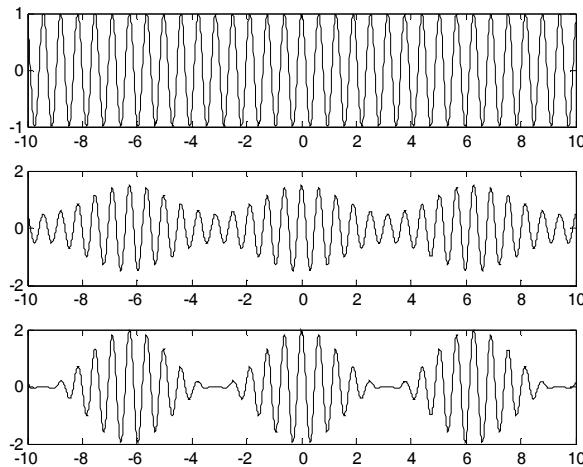
$$m = \frac{A_M}{A_0}.$$

С учетом этого можно записать

$$s_{AM}(t) = A_0 (1 + m \cos(\Omega t + \Phi_0)) \cos(\omega_0 t + \phi_0).$$

На рис. 8.3 показан вид однотонального АМ-сигнала при разных значениях коэффициента модуляции:

```
>> Fs = 100; % частота дискретизации
>> t = -10:1/Fs:10; % дискретное время
>> omega0 = 10; % несущая частота
>> OMEGA = 1; % частота модулирующего сигнала
>> s_AM_0 = cos(omega0 * t);
>> s_AM_50 = (1+0.5*cos(OMEGA*t)) .* cos(omega0 * t);
>> s_AM_100 = (1+cos(OMEGA*t)) .* cos(omega0 * t);
>> subplot(3, 1, 1)
>> plot(t, s_AM_0) % модуляция отсутствует
>> subplot(3, 1, 2)
>> plot(t, s_AM_50) % глубина модуляции 50%
>> subplot(3, 1, 3)
>> plot(t, s_AM_100) % глубина модуляции 100%
```



**Рис. 8.3.** Однотональный АМ-сигнал: сверху —  $m = 0$  (немодулированная несущая), в центре —  $m = 0,5$ , снизу —  $m = 1$

Очевидно, что максимальное значение огибающей однотонального АМ-сигнала достигается тогда, когда оба косинуса равны 1:

$$A_{\max} = A(1 + m).$$

Минимальное значение огибающей соответствует тем моментам, когда косинус модулирующего сигнала равен  $-1$ :

$$A_{\min} = A(1 - m).$$

Отсюда следует формула, позволяющая вычислить коэффициент модуляции  $m$  по результатам измерения (например, с помощью осциллографа) максимальной и минимальной амплитуд сигнала:

$$m = \frac{A_{\max} - A_{\min}}{A_{\max} + A_{\min}}.$$

Обычно коэффициент модуляции должен лежать в диапазоне  $0\dots1$ . При  $m > 1$  имеет место *перемодуляция*; подстановка таких значений в приведенную формулу дает результат, показанный на рис. 8.4:

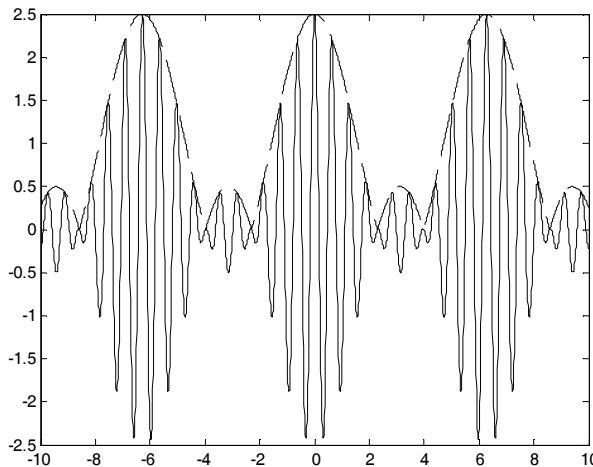
```
>> A_AM_150 = 1 + 1.5 * cos(OMEGA*t);
>> s_AM_150 = A_AM_150 .* cos(omega0 * t); % глубина модуляции 150%
>> plot(t, s_AM_150, t, abs(A_AM_150), '--')
```

Как уже указывалось, амплитудная огибающая при перемодуляции искажается. Однако, как мы увидим далее, и этот режим может быть полезен на практике.

Теперь займемся анализом спектрального состава такого колебания. Для этого сначала раскроем скобки в выражении для однотонального АМ-сигнала, а затем выполним тригонометрические преобразования:

$$\begin{aligned}
 s_{\text{AM}}(t) &= A_0 \cos(\omega_0 t + \varphi_0) + A_0 m \cos(\Omega t + \Phi_0) \cos(\omega_0 t + \varphi_0) = \\
 &= A_0 \cos(\omega_0 t + \varphi_0) + \frac{A_0 m}{2} \cos((\omega_0 + \Omega)t + \varphi_0 + \Phi_0) + \\
 &\quad + \frac{A_0 m}{2} \cos((\omega_0 - \Omega)t + \varphi_0 - \Phi_0).
 \end{aligned} \tag{8.3}$$

Результат преобразования показывает, что однотональный АМ-сигнал состоит из трех гармонических составляющих, одна из которых представляет собой несущее колебание с частотой  $\omega_0$ , а две оставшихся (их называют *боковыми частотами*) отстоят от него вверх и вниз по частоте на величину  $\Omega$ . Амплитуда несущего колебания равна  $A_0$  и не зависит от уровня модулирующего сигнала. Амплитуды боковых частот, равные  $A_0 m/2$ , напротив, пропорциональны коэффициенту модуляции.



**Рис. 8.4.** Однотональный АМ-сигнал в случае перемодуляции ( $m = 1,5$ )

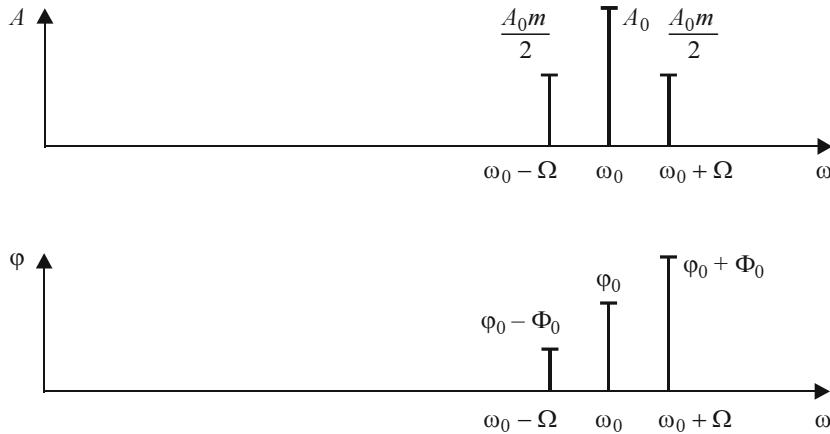
Для *верхней* боковой частоты начальные фазы несущей и модулирующего сигнала складываются, а для *нижней* — вычитаются.

Амплитудный и фазовый спектры однотонального АМ-сигнала показаны на рис. 8.5.

### ЗАМЕЧАНИЕ

Следует подчеркнуть, что в общем случае полученное представление однотонального АМ-сигнала *не является* его разложением в ряд Фурье. Такое представление в виде суммы гармонических функций может рассматриваться как ряд Фурье только в том случае, когда для всех трех частот спектральных составляющих существует общий делитель. Это возможно, если отношение  $\omega_0/\Omega$  является рациональной дробью.

Из графиков видно, что *ширина спектра* однотонального АМ-сигнала в два раза превышает частоту модулирующего сигнала:  $\Delta\omega = 2\Omega$ .

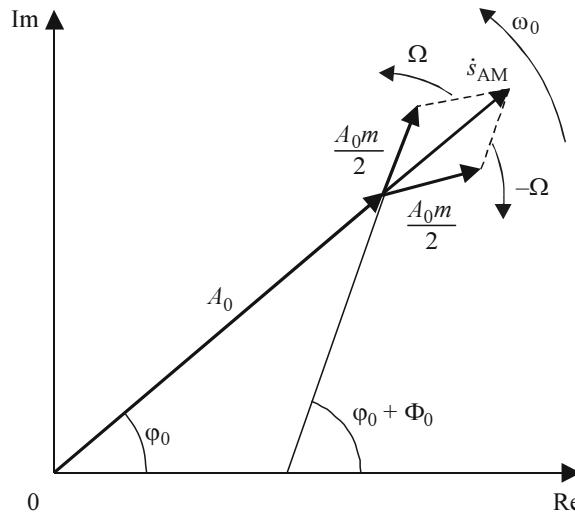


**Рис. 8.5.** Амплитудный (сверху) и фазовый (снизу) спектры однотонального АМ-сигнала

Чтобы понять, как из трех гармонических составляющих с *постоянной* амплитудой складывается сигнал с *меняющейся* амплитудой, создадим векторную диаграмму. Для этого представим каждое из трех гармонических колебаний как вещественную часть комплексной экспоненты:

$$s_{\text{AM}}(t) = \operatorname{Re} \left[ e^{j\omega_0 t} \left( A_0 e^{j\varphi_0} + \frac{A_0 m}{2} e^{j(\Omega t + \varphi_0 + \Phi_0)} + \frac{A_0 m}{2} e^{j(-\Omega t + \varphi_0 - \Phi_0)} \right) \right].$$

Построим векторную диаграмму, демонстрирующую суммирование этих составляющих (рис. 8.6).



**Рис. 8.6.** Векторная диаграмма формирования однотонального АМ-сигнала

При сложении колебаний с разными частотами постоянно меняются их взаимные фазовые соотношения. Векторы боковых частот врачаются в разные стороны с уг-

ловой скоростью  $\Omega$  относительно вектора несущего колебания. В результате колебания боковых частот оказываются то синфазными с несущим колебанием, увеличивая его амплитуду, то противофазными с ним (тогда амплитуда сигнала уменьшается). Наконец, вектор несущей частоты вращается с угловой скоростью  $\omega_0$ . Значение АМ-сигнала определяется проекцией результирующего вектора на вещественную (горизонтальную) ось.

## АМ-сигнал в общем случае

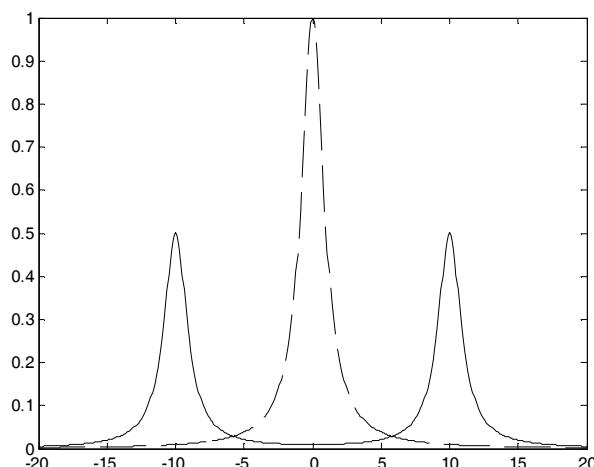
Формула, связывающая спектр АМ-сигнала со спектром модулирующего сигнала, нами уже была получена (см. в главе 1 свойство преобразования Фурье (1.24), касающееся умножения сигнала на гармоническую функцию). Действительно, ведь АМ-сигнал — это и есть результат умножения модулирующего сигнала (с добавленной постоянной составляющей) на гармоническое несущее колебание.

Повторим, слегка изменив обозначения, полученную ранее формулу (1.24):

$$\dot{S}_{\text{AM}}(\omega) = \frac{1}{2} e^{j\phi_0} \dot{S}_A(\omega + \omega_0) + \frac{1}{2} e^{-j\phi_0} \dot{S}_A(\omega - \omega_0). \quad (8.4)$$

Спектр огибающей  $A(t)$  при амплитудной модуляции сдвигается в область несущей частоты  $\pm\omega_0$ , "раздваиваясь" и уменьшаясь в два раза по уровню. Покажем это на графике, задав какую-нибудь функцию для спектра огибающей  $\dot{S}_A(\omega)$  (рис. 8.7):

```
>> w = -20:0.1:20; % значения частот для расчета
>> w0 = 10; % несущая частота
>> S_A = 1./(1+w.^2); % спектр модулирующего сигнала
>> % спектр модулированного сигнала
>> S_AM = 0.5./ (1+ (w+w0).^2) + 0.5./ (1+ (w-w0).^2);
>> plot(w, S_A, '--', w, S_AM)
```



**Рис. 8.7.** Спектры огибающей (пунктирная линия) и АМ-сигнала (сплошная линия)

Итак, спектр АМ-сигнала в общем случае содержит несущую частоту (уровень которой определяется постоянной составляющей огибающей), а также *верхнюю и нижнюю боковые полосы*.

Из графиков видно, что ширина спектра АМ-сигнала вдвое больше максимальной (граничной) частоты модулирующего сигнала:  $\Delta\omega = 2\Omega_{\max}$ .

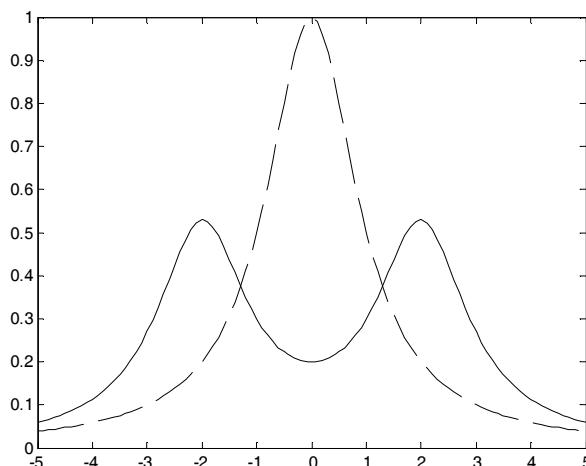
Вычислим значение спектральной функции АМ-сигнала на несущей частоте:

$$\dot{S}_{\text{AM}}(\omega_0) = \frac{1}{2}e^{-j\varphi_0}\dot{S}_A(0) + \frac{1}{2}e^{j\varphi_0}\dot{S}_A(2\omega_0).$$

Первое слагаемое результата — как и положено, деленная пополам постоянная составляющая модулирующего сигнала. А вот второе слагаемое представляет собой "хвост" от второй "половинки" спектра, сконцентрированной в области отрицательных частот, в окрестностях частоты  $-\omega_0$ . Следует иметь в виду, что, поскольку все реальные сигналы имеют конечную длительность (и, следовательно, бесконечно протяженный спектр), данное явление наложения "хвостов" всегда будет иметь место. В большинстве практических ситуаций, однако, несущая частота значительно превышает эффективную граничную частоту спектра огибающей, так что влияние данного эффекта пренебрежимо мало.

Графически проиллюстрируем наложение "хвостов" сдвинутых копий спектра, уменьшив в рассмотренном ранее примере несущую частоту (рис. 8.8):

```
>> w = -5:0.1:5; % значения частот для расчета
>> w0 = 2; % несущая частота
>> S_A = 1./(1+w.^2); % спектр модулирующего сигнала
>> % спектр модулированного сигнала
>> S_AM = 0.5./((1+(w+w0).^2) + 0.5./((1+(w-w0).^2));
>> plot(w, S_A, '--', w, S_AM)
```



**Рис. 8.8.** При недостаточно высокой несущей частоте спектр АМ-сигнала (сплошная линия) может быть существенно несимметричным относительно несущей частоты из-за наложения "хвостов"

## Энергетические соотношения в АМ-сигнале

В этом разделе нам вновь придется вернуться к рассмотрению однотонального АМ-сигнала (8.2), чтобы выяснить, как распределяется мощность в его спектре.

Определим среднюю мощность однотонального АМ-сигнала. Как говорилось ранее, АМ-сигнал в общем случае не является периодическим, поэтому для расчета средней мощности необходимо применить предельный переход согласно формуле (1.5):

$$\begin{aligned} P_{\text{ср}} &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} s^2(t) dt = \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \left( A_0 (1 + m \cos(\Omega t + \Phi_0)) \cos(\omega_0 t + \varphi_0) \right)^2 dt = \\ &= \frac{A_0^2}{2} + \frac{A_0^2 m^2}{4}. \end{aligned}$$

### ЗАМЕЧАНИЕ

Тот же результат можно получить и без вычисления интеграла. Достаточно вспомнить, что мощность гармонического колебания с амплитудой  $A$  равна  $A^2/2$  и что гармонические колебания разных частот являются некоррелированными, а потому их мощности можно складывать. Далее остается только применить все сказанное к представлению однотонального АМ-сигнала в виде суммы гармонических составляющих (см. формулу (8.3) и рис. 8.5).

Первое слагаемое не зависит от коэффициента модуляции и представляет собой мощность немодулированной несущей. Полезная мощность, заключенная в боковых частотах, представлена вторым слагаемым.

Введем в рассмотрение *коэффициент полезного действия* (КПД) амплитудной модуляции, определив его как отношение мощности боковых частот к общей средней мощности сигнала:

$$\eta_{\text{AM}} = \frac{A_0^2 \frac{m^2}{4}}{A_0^2 \left( \frac{1}{2} + \frac{m^2}{4} \right)} = \frac{m^2}{m^2 + 2}.$$

Построим график зависимости КПД от коэффициента модуляции  $m$  (рис. 8.9):

```
>> m = 0:0.01:1;
>> eta = m.^2./(m.^2+2);
>> plot(m, eta)
>> xlabel('m')
>> ylabel('\eta_{AM}')
```

Результаты неутешительные — даже при максимально допустимом значении коэффициента модуляции ( $m = 1$ ) КПД составляет лишь 33%, т. е. две трети мощности тратится на передачу бесполезной в информационном отношении несущей.

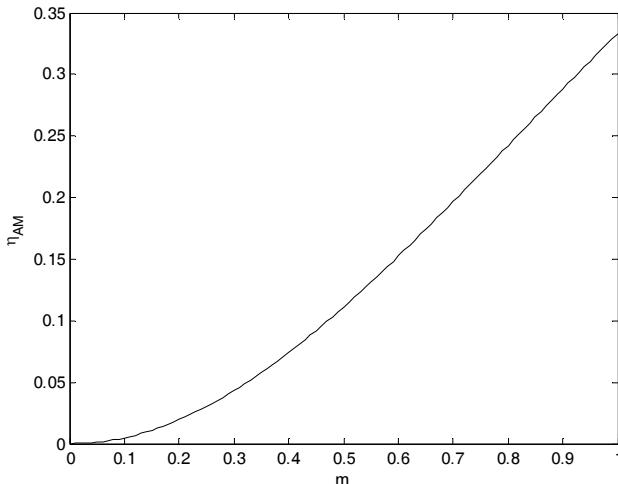


Рис. 8.9. Зависимость КПД от коэффициента амплитудной модуляции

Исторически АМ была первым практически освоенным видом модуляции. Однако низкий КПД и ширина спектра, вдвое превышающая ширину спектра модулирующего сигнала, привели к тому, что сфера применения АМ стала довольно узкой. В настоящее время АМ применяется для радиовещания на сравнительно низких частотах (в диапазонах длинных, средних и коротких волн) и для передачи изображения в телевизионном вещании.

## Демодуляция АМ

Демодуляция АМ-сигнала может быть выполнена несколькими способами. Простейший путь — имитировать работу аналогового двухполупериодного детектора. Мы вычисляем модуль входного АМ-сигнала, а затем сглаживаем получившиеся однополярные косинусоидальные импульсы, пропуская их через ФНЧ (рис. 8.10):

```
>> y = abs(s_AM_50); % модуль АМ-сигнала
>> [b, a] = butter(5, 2*OMEGA/pi/Fs); % сглаживающий ФНЧ
>> z = filtfilt(b, a, y); % фильтрация
>> plot(t(1:1000), y(1:1000), '--', t(1:1000), z(1:1000))
```

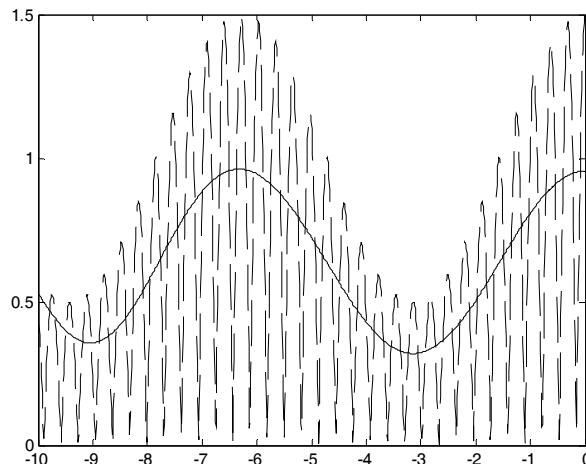
Данный способ, очевидно, не будет работать правильно в случае перемодуляции.

Следующий метод — так называемое *синхронное детектирование*, суть которого состоит в умножении частоты сигнала на *опорное колебание* с несущей частотой:

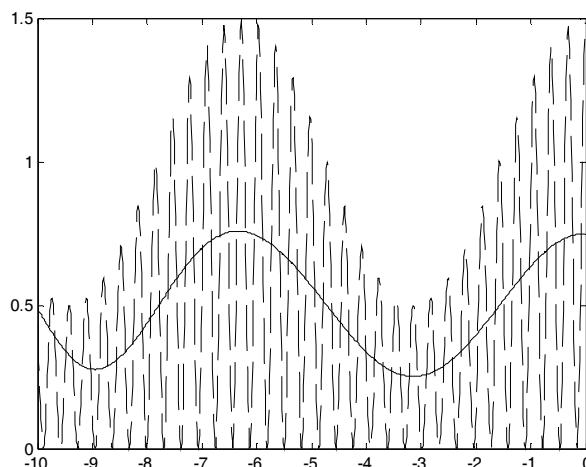
$$\begin{aligned} y(t) &= s_{AM}(t) \cos(\omega_0 t + \phi_0) = A(t) \cos^2(\omega_0 t + \phi_0) = \\ &= \frac{1}{2} A(t) + \frac{1}{2} A(t) \cos(2\omega_0 t + 2\phi_0). \end{aligned} \tag{8.5}$$

Результат умножения содержит два слагаемых. Первое — это искомая амплитудная функция, второе — АМ-сигнал с несущей частотой  $2\omega_0$ . Этот высокочастотный сигнал легко удаляется путем пропускания сигнала через ФНЧ (рис. 8.11):

```
>> % умножение на опорное колебание
>> y = s_AM_50 .* cos(omega0*t);
>> [b, a] = butter(5, 2*OMEGA/pi/Fs); % сглаживающий ФНЧ
>> z = filtfilt(b, a, y); % фильтрация
>> plot(t(1:1000), y(1:1000), '--', t(1:1000), z(1:1000))
```



**Рис. 8.10.** Двухполупериодное детектирование АМ-сигнала: однополярные импульсы (пунктирная линия) и результат их сглаживания (сплошная линия)



**Рис. 8.11.** Синхронное детектирование АМ-сигнала: результат умножения на опорное колебание (пунктирная линия) и выделенный низкочастотный сигнал (сплошная линия)

Однако в данном случае необходимо очень точное совпадение начальных фаз и частот опорного колебания демодулятора и несущего колебания АМ-сигнала. При совпадении частот, но несовпадении начальных фаз выходной низкочастотный сигнал оказывается умноженным на косинус фазовой ошибки:

$$\begin{aligned}y(t) &= s_{\text{AM}}(t) \cos(\omega_0 t + \varphi) = A(t) \cos(\omega_0 t + \varphi_0) \cos(\omega_0 t + \varphi) = \\&= \frac{1}{2} A(t) \cos(\varphi - \varphi_0) + \frac{1}{2} A(t) \cos(2\omega_0 t + \varphi_0 + \varphi).\end{aligned}$$

Таким образом, при наличии фазовой ошибки уровень полезного сигнала на выходе демодулятора падает, а при ошибке, равной  $90^\circ$ , становится равным нулю.

При наличии частотного сдвига между несущим и опорным колебаниями ситуация становится еще хуже — выходной низкочастотный сигнал оказывается умноженным на гармоническое колебание с разностной частотой:

$$\begin{aligned}y(t) &= s_{\text{AM}}(t) \cos((\omega_0 + \Delta\omega)t) = A(t) \cos(\omega_0 t + \varphi_0) \cos((\omega_0 + \Delta\omega)t) = \\&= \frac{1}{2} A(t) \cos(\Delta\omega t - \varphi_0) + \frac{1}{2} A(t) \cos((2\omega_0 + \Delta\omega)t + \varphi_0).\end{aligned}$$

В результате выходной сигнал будет пульсировать с частотой  $\Delta\omega$ . Это явление называется *биениями* (*beat*), а разность частот  $\Delta\omega$  — *частотой биений* (*beat frequency*).

Для поддержания частотной и фазовой синхронизации между несущим и опорным колебаниями используются следящие системы фазовой автоподстройки частоты (ФАПЧ), рассмотрение которых выходит за рамки тематики данной книги.

Достоинством синхронного детектирования является то, что оно позволяет правильно демодулировать сигнал даже в случае перемодуляции (ведь формула (8.5) не перестает быть верной в случае знакопеременной функции  $A(t)$ ).

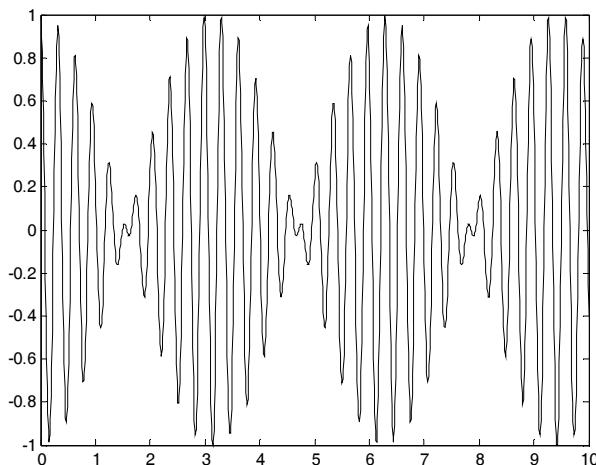
## Разновидности амплитудной модуляции

Попытки улучшить характеристики АМ привели к разработке нескольких ее модификаций, которые и будут рассмотрены в данном разделе.

### АМ с подавленной несущей

Первое, что приходит в голову при размышлении на тему повышения КПД амплитудной модуляции, — это идея удалить бесполезное несущее колебание, все-таки отказавшись от добавления постоянной составляющей к модулирующему сигналу. Такой способ называется *AM с подавленной несущей* (AM-ПН, английский термин — *amplitude modulation with suppressed carrier*, *AM-SC*):

$$s(t) = s_{\text{M}}(t) \cos(\omega_0 t + \varphi_0).$$



**Рис. 8.12.** Однотональная АМ с подавленной несущей

Построим график такого сигнала при однотональной модуляции (рис. 8.12):

```
>> t = 0:0.01:10; % дискретное время
>> omega0 = 20; % несущая частота
>> OMEGA = 1; % частота модулирующего сигнала
>> s_M = cos(OMEGA*t); % модулирующий сигнал
>> s_AM_SC = s_M .* cos(omega0*t); % АМ-ПН сигнал
>> plot(t, s_AM_SC)
```

Энергетический выигрыш при этом, конечно, велик (согласно введенному нами определению, КПД становится равным 100%).

Ширина спектра АМ-сигнала с подавленной несущей такая же, как в случае обычной АМ (ведь подавлена средняя (несущая) частота, а боковые частоты остались на месте).

Таким образом, АМ с подавленной несущей обладает определенными преимуществами по сравнению с обычной АМ. Однако этот способ модуляции не получил широкого распространения, и связано это с проблемами, возникающими при демодуляции сигнала.

## Демодуляция АМ с подавленной несущей

Как уже было указано ранее в разд. "Демодуляция АМ" этой главы, формула (8.5) справедлива как для однополярной, так и для знакопеременной амплитудной функции  $A(t)$ . Поэтому демодуляция АМ с подавленной несущей может выполняться путем синхронного детектирования. При этом сохраняет силу все сказанное о необходимости точного соответствия частот и начальных фаз несущего и опорного колебаний.

Для облегчения правильного восстановления несущей иногда применяют следующий прием. На передающей стороне несущее колебание подавляется не полностью.

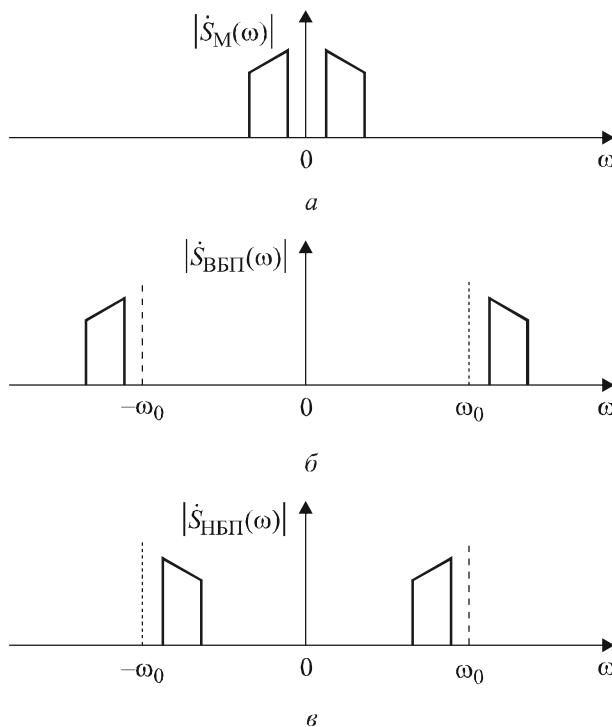
Его "остаток" с небольшой амплитудой (он называется *пилот-сигналом*) используют для синхронизации частоты и фазы несущего колебания на приемной стороне.

## Однополосная модуляция

Рассмотренная в предыдущем разделе двухполосная АМ с подавленной несущей имеет преимущества перед обычной АМ только в энергетическом смысле — за счет устранения несущего колебания. Ширина же спектра при этом остается равной удвоенной частоте модулирующего сигнала.

Однако можно легко заметить, что спектры двух боковых полос АМ-сигнала являются зеркальным отражением друг друга, т. е. они несут одну и ту же информацию. Поэтому одну из боковых полос можно удалить. Получающаяся модуляция называется *однополосной* (английский термин — *single side band, SSB*).

В зависимости от того, какая боковая полоса сохраняется, говорят об однополосной модуляции с использованием *верхней* или *нижней боковой полосы*. Формирование однополосного сигнала проще всего пояснить, приведя несколько спектральных графиков (рис. 8.13).



**Рис. 8.13.** Однополосная модуляция: *a* — спектр модулирующего сигнала, *б* — спектр однополосного сигнала с верхней боковой полосой, *в* — то же с нижней боковой полосой

По сути дела, при однополосной модуляции происходит просто сдвиг спектра сигнала в окрестности частоты несущего колебания. В отличие от АМ, каждая "половинка" спектра смещается в своем направлении: область положительных частот — к  $\omega_0$ , а область отрицательных частот — к  $-\omega_0$ .

Очевидно, что ширина спектра однополосного сигнала равна ширине спектра модулирующего сигнала. Таким образом, спектр однополосного сигнала оказывается в два раза уже, чем при обычной АМ.

В отличие от предыдущих случаев, здесь нам не удастся простыми средствами выразить связь между модулированным и модулирующим сигналами. Чтобы сделать это, придется воспользоваться преобразованием Гильберта и понятием аналитического сигнала (см. разд. "Комплексная огибающая" главы 1).

Итак, прежде всего мы формируем из модулирующего сигнала аналитический сигнал, имеющий односторонний спектр. Умножение этого сигнала на  $\exp(j\omega_0 t)$  сдвигает его односторонний спектр на  $\omega_0$  вправо (вверх по частоте), формируя односторонний спектр однополосного сигнала с верхней боковой полосой. Наконец, чтобы перейти от аналитического сигнала обратно к вещественному, нужно взять вещественную часть. Формирование сигнала с нижней боковой полосой описывается аналогично, только умножать аналитический сигнал нужно на  $\exp(-j\omega_0 t)$  (тогда его спектр сдвигается влево, в область отрицательных частот, и займет положение нижней боковой полосы). Теперь осталось записать все сказанное математически:

$$\begin{aligned} s_{SSB}(t) &= \operatorname{Re}\left(\left(x(t) + jx_{\perp}(t)\right) \exp(\pm j\omega_0 t)\right) = \\ &= x(t) \cos \omega_0 t \mp x_{\perp}(t) \sin \omega_0 t. \end{aligned} \quad (8.6)$$

Знак "минус" в окончательной формуле соответствует выделению верхней боковой полосы, "плюс" — нижней.

Итак, однополосный сигнал можно представить как сумму двух АМ-сигналов, несущие колебания которых имеют одну и ту же частоту, но сдвинуты по фазе друг относительно друга на  $90^\circ$ . Амплитудными функциями этих АМ-сигналов являются модулирующий сигнал и его квадратурное дополнение. В зависимости от того, складываются эти два АМ-сигнала или вычитаются (а точнее, какая из двух несущих опережает другую по фазе), формируется однополосный сигнал с верхней или нижней боковой полосой.

Амплитудная огибающая однополосного сигнала нисколько не похожа на модулирующий низкочастотный сигнал. Проверим это на конкретном примере. Случай гармонической модуляции на сей раз рассматривать не будем — он слишком прост (синусоида с частотой  $\Omega$  при однополосной модуляции превратится в синусоиду с частотой  $\omega_0 \pm \Omega$ ; выбор знака определяется тем, какая используется боковая полоса). Возьмем модулирующий сигнал, состоящий из двух гармоник (рис. 8.14):

```
>> Fs = 1000; % частота дискретизации
>> t = 0:1/Fs:1; % дискретное время
```

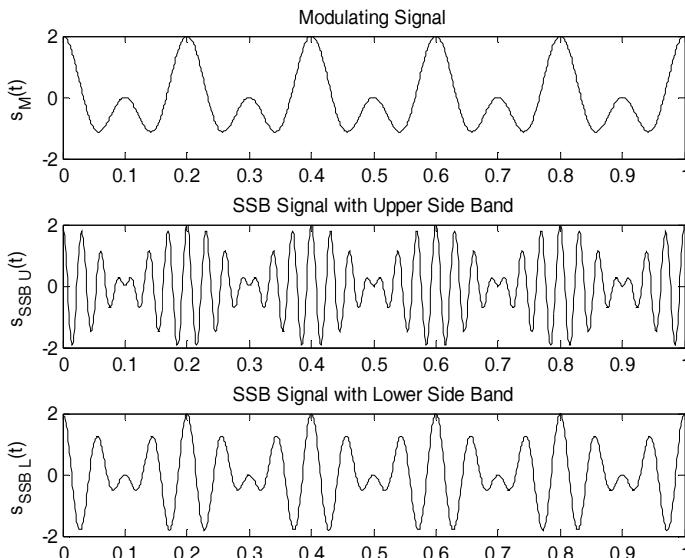
```

>> Fc = 25;           % несущая частота
>> f1 = 5;            % первая частота модуляции
>> f2 = 10;           % вторая частота модуляции
>> % модулирующий сигнал
>> s_M = cos(2*pi*f1*t) + cos(2*pi*f2*t);
>> % SSB-сигнал с верхней боковой полосой
>> s_SSB_U = cos(2*pi*(Fc+f1)*t) + cos(2*pi*(Fc+f2)*t);
>> % SSB-сигнал с нижней боковой полосой
>> s_SSB_L = cos(2*pi*(Fc-f1)*t) + cos(2*pi*(Fc-f2)*t);
>> subplot(3, 1, 1)
>> plot(t, s_M)
>> ylabel('s_{M}(t)')
>> title('Modulating Signal')
>> subplot(3, 1, 2)
>> plot(t, s_SSB_U)
>> ylabel('s_{SSB U}(t)')
>> title('SSB Signal with Upper Side Band')
>> subplot(3, 1, 3)
>> plot(t, s_SSB_L)
>> ylabel('s_{SSB L}(t)')
>> title('SSB Signal with Lower Side Band')

```

### ЗАМЕЧАНИЕ

Из (8.6) видно, что амплитудная огибающая сигнала с однополосной модуляцией представляет собой модуль аналитического сигнала  $x(t) + jx_{\perp}(t)$ .



**Рис. 8.14.** Однополосная модуляция: сверху — модулирующий сигнал, в центре — сигнал с верхней боковой полосой, снизу — сигнал с нижней боковой полосой

Однополосная модуляция с различным уровнем несущего колебания — от полного подавления до полного сохранения — нашла широкое применение в технике профессиональной радиосвязи.

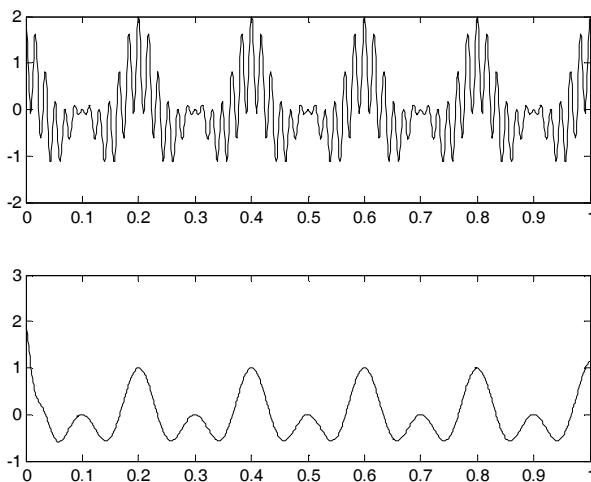
## Демодуляция однополосного сигнала

Несмотря на то что визуально (на графике) однополосный сигнал сильно отличается от обычного АМ-сигнала, его демодуляция возможна тем же методом синхронного детектирования — путем умножения на опорное колебание:

$$\begin{aligned} y(t) &= s_{SSB}(t) \cos \omega_0 t = (x(t) \cos \omega_0 t \pm x_{\perp}(t) \sin \omega_0 t) \cos \omega_0 t = \\ &= \frac{1}{2} x(t) + \frac{1}{2} x(t) \cos 2\omega_0 t \pm \frac{1}{2} x_{\perp}(t) \sin 2\omega_0 t. \end{aligned}$$

Результат умножения (рис. 8.15, сверху) содержит три слагаемых. Первое — это модулирующий сигнал, а второе и третье образуют однополосный сигнал на удвоенной несущей  $2\omega_0$ . Остается лишь выделить модулирующий сигнал с помощью ФНЧ (рис. 8.15, снизу):

```
>> % умножение на опорное колебание
>> y = s_SSB_U .* cos(2*pi*Fc*t);
>> [b, a] = butter(5, Fc/Fs^2); % сглаживающий ФНЧ
>> z = filtfilt(b, a, y); % фильтрация
>> subplot(2, 1, 1)
>> plot(t, y)
>> subplot(2, 1, 2)
>> plot(t, z)
```



**Рис. 8.15.** Демодуляция однополосного сигнала, показанного на рис. 8.14:  
сверху — результат умножения на несущее колебание,  
снизу — отфильтрованный демодулированный сигнал

Эффекты, возникающие при наличии частотного или фазового сдвига у опорного колебания, в случае однополосного сигнала проявляются не так, как при демодуляции АМ. Сначала рассмотрим ситуацию, когда есть только фазовый сдвиг, а ошибка по частоте отсутствует:

$$\begin{aligned} y(t) &= s_{\text{SSB}}(t) \cos(\omega_0 t + \varphi) = (x(t) \cos \omega_0 t \pm x_{\perp}(t) \sin \omega_0 t) \cos(\omega_0 t + \varphi) = \\ &= x(t) \cos \omega_0 t \cos(\omega_0 t + \varphi) \pm x_{\perp}(t) \sin \omega_0 t \cos(\omega_0 t + \varphi) = \\ &= \frac{1}{2} (x(t) \cos \varphi \pm x_{\perp}(t) \sin \varphi) + \frac{1}{2} (x(t) \cos(2\omega_0 t + \varphi) \pm x_{\perp}(t) \sin(2\omega_0 t + \varphi)). \end{aligned}$$

Низкочастотная составляющая в этом случае представляет собой линейную комбинацию модулирующего сигнала  $x(t)$  и его квадратурного дополнения  $x_{\perp}(t)$ . Со спектральной точки зрения это означает фазовый сдвиг всех частотных составляющих сигнала на  $\varphi$ . Форма сигнала, разумеется, при этом искажается. Приемлемы ли такие искажения, зависит от характера передаваемого сигнала. Если однополосная модуляция используется для передачи звукового сигнала, фазовые искажения несущественны, поскольку ухо человека нечувствительно к фазовым соотношениям в акустическом сигнале.

Пусть теперь опорное колебание имеет частоту  $\omega_0 + \Delta\omega$ . Чтобы получить результат умножения, достаточно подставить в предыдущую формулу значение  $\varphi = \Delta\omega t$ :

$$\begin{aligned} y(t) &= s_{\text{SSB}}(t) \cos((\omega_0 + \Delta\omega)t) = \frac{1}{2} (x(t) \cos(\Delta\omega t) \pm x_{\perp}(t) \sin(\Delta\omega t)) + \\ &+ \frac{1}{2} (x(t) \cos((2\omega_0 + \Delta\omega)t) \pm x_{\perp}(t) \sin((2\omega_0 + \Delta\omega)t)). \end{aligned}$$

Низкочастотная составляющая здесь фактически представляет собой однополосный сигнал с несущей частотой  $\Delta\omega$ . С частотной точки зрения это означает сдвиг спектра модулирующего сигнала на  $\Delta\omega$ . Это, разумеется, значительно более серьезные искажения, чем в предыдущем случае. Скажем, при передаче музыки искажения такого рода совершенно неприемлемы. При передаче же речевого сигнала (а профессиональная радиосвязь — это основная сфера применения однополосной модуляции) сдвиг спектра приводит к искажению тембра голоса, но разборчивость речи сохраняется при величине сдвига до нескольких десятков, а для опытного оператора — до нескольких сотен герц.

## Полярная модуляция

То, что мы рассмотрим в этом разделе, не является отдельным видом модуляции. Это скорее демонстрация применения амплитудной модуляции для решения конкретной технической задачи. Речь пойдет о том, как реализуется стереофоническое радиовещание в УКВ-диапазоне.

Для осуществления *стереовещания* необходимо передавать два сигнала  $s_L(t)$  и  $s_R(t)$  (левого и правого каналов) одновременно. В то же время при разработке сис-

темы такого вещания накладывается еще и требование совместимости с уже имеющимися монофоническими приемниками. Поэтому для стереовещания модифицируется низкочастотный *модулирующий* сигнал, который поступает на вход модулятора передатчика (в передатчике используется угловая модуляция, которая будет рассмотрена далее). Низкочастотная составляющая модулирующего сигнала, лежащая в звуковом диапазоне, для совместимости с монофоническими приемниками должна представлять собой монофонический сигнал, т. е. сумму сигналов левого и правого каналов:

$$s_{\text{MONO}}(t) = s_L(t) + s_R(t).$$

В области более высоких (ультразвуковых) частот модулирующего сигнала с помощью амплитудной модуляции передается дополнительный сигнал, позволяющий впоследствии выделить из полученной смеси сигналы  $s_L(t)$  и  $s_R(t)$  по отдельности. Поскольку монофонический сигнал — это сумма двух каналов, этим дополнительным компонентом, позволяющим восстановить исходные сигналы двух каналов, естественно выбрать их разность:

$$s_{\text{DIFF}}(t) = s_L(t) - s_R(t).$$

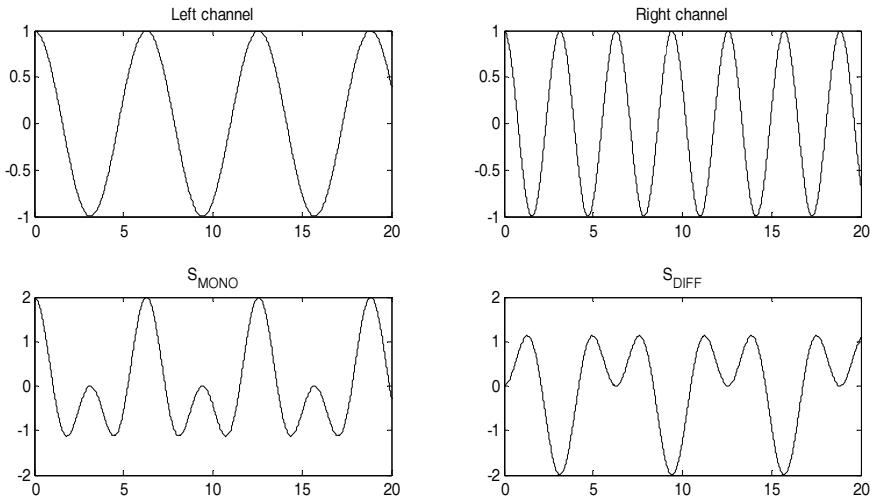
Рассмотрим конкретный пример, задав для правого и левого каналов гармонические сигналы разных частот:

$$s_L(t) = A_L \cos(\Omega_L t),$$

$$s_R(t) = A_R \cos(\Omega_R t).$$

Графики сигналов  $s_L(t)$ ,  $s_R(t)$ ,  $s_{\text{MONO}}(t)$  и  $s_{\text{DIFF}}(t)$  для данного случая показаны на рис. 8.16:

```
>> t = 0:0.01:20;          % дискретное время
>> w_L = 1;                % частота сигнала левого канала
>> w_R = 2;                % частота сигнала правого канала
>> s_L = cos(w_L*t);      % сигнал левого канала
>> s_R = cos(w_R*t);      % сигнал правого канала
>> s_mono = s_L + s_R;    % монофонический сигнал
>> s_diff = s_L - s_R;    % разностный сигнал
>> subplot(2, 2, 1)
>> plot(t, s_L)
>> title('Left channel')
>> subplot(2, 2, 2)
>> plot(t, s_R)
>> title('Right channel')
>> subplot(2, 2, 3)
>> plot(t, s_mono)
>> title('S_{MONO}')
>> subplot(2, 2, 4)
>> plot(t, s_diff)
>> title('S_{DIFF}')
```



**Рис. 8.16.** Графики сигналов: сверху — сигналы левого и правого каналов, снизу — суммарный и разностный сигналы

Складывая и вычитая сигналы  $s_{\text{MONO}}(t)$  и  $s_{\text{DIFF}}(t)$ , можно восстановить сигналы левого и правого каналов:

$$s_L(t) = \frac{s_{\text{MONO}}(t) + s_{\text{DIFF}}(t)}{2},$$

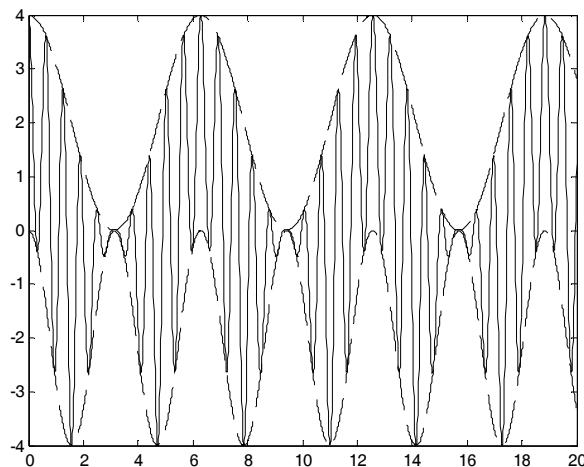
$$s_R(t) = \frac{s_{\text{MONO}}(t) - s_{\text{DIFF}}(t)}{2}.$$

Монофонический сигнал, как уже было сказано, для совместимости должен передаваться как есть, а разностный сигнал модулирует несущую частоту, расположенную несколько выше звукового диапазона частот (ее называют *поднесущей* (*subcarrier*), поскольку весь сигнал о формировании которого идет речь, затем используется для угловой модуляции несущего колебания радиопередатчика):

$$s(t) = s_{\text{MONO}}(t) + (A_0 + s_{\text{DIFF}}(t)) \cos(\omega_0 t).$$

Построим график получающегося сигнала (рис. 8.17):

```
>> w0 = 10; % поднесущая частота
>> A0 = 2; % смещение для разностного сигнала
>> % композитный стереосигнал
>> s = s_mono + (A0 + s_diff) .* cos(w0*t);
>> plot(t, s)
>> hold on
>> plot(t, s_mono+A0+s_diff, '--') % верхняя огибающая
>> plot(t, s_mono-A0-s_diff, '--') % нижняя огибающая
>> hold off
```



**Рис. 8.17.** Композитный стереосигнал

Получающийся сигнал (его называют *композитным стереосигналом*), как видно из рисунка, благодаря наличию низкочастотной составляющей обладает интересным свойством: его верхняя и нижняя (положительная и отрицательная) огибающие *не совпадают*. Более того, каждая из них с точностью до постоянной составляющей соответствует сигналу одного из стереоканалов: верхняя (положительная) огибающая — это левый канал, а нижняя (отрицательная) — правый. Такая структура сигнала позволяет легко выделить из него сигналы стереоканалов.

Однако в показанном на рис. 8.17 виде сигнал для передачи не используется. Дело в том, что из-за наличия несущего колебания размах сигнала (разность между его максимальным и минимальным значениями) сильно увеличивается по сравнению с монофоническим сигналом, что нежелательно. Поэтому несущее колебание частично или полностью подавляют, так что разностный сигнал передается в режиме *перемодуляции*.

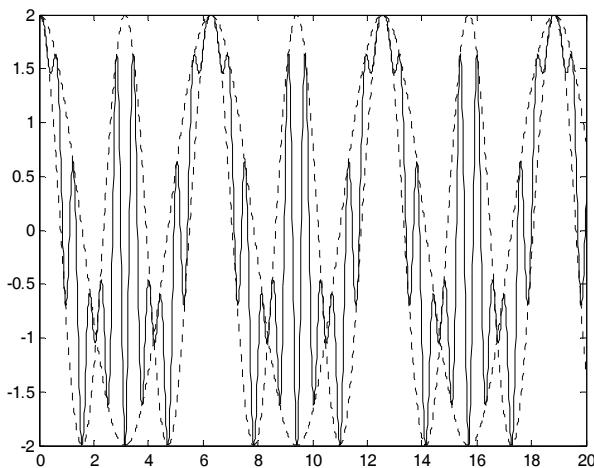
### ЗАМЕЧАНИЕ

Отечественный стандарт, согласно которому ведется радиовещание в диапазоне 65—74 МГц, предусматривает частичное подавление поднесущей частоты, равной 31,25 кГц. Зарубежный стандарт, используемый в диапазоне 88—108 МГц, специфицирует полное подавление поднесущей частоты, равной 38 кГц. При этом для обеспечения возможности правильного восстановления поднесущего колебания в стереодекодере приемника в состав сигнала вводится *пилот-сигнал* на половинной частоте поднесущей (19 кГц).

Изобразим композитный стереосигнал для случая полного подавления несущей (пилот-сигнал добавлять не будем) (рис. 8.18):

```
>> s = s_mono + s_diff .* cos(w0*t);
>> plot(t, s)
>> hold on
>> plot(t, s_mono+s_diff, ':') % верхняя огибающая
```

```
>> plot(t, s_mono-s_diff, ':') % нижняя огибающая
>> hold off
```



**Рис. 8.18.** Композитный стереосигнал с подавленной поднесущей

Из графика видно, что благодаря подавлению поднесущей размах сигнала существенно (в два раза) уменьшился.

## Угловая модуляция

В начале главы уже было сказано, что фазовая и частотная модуляция тесно связаны друг с другом, благодаря чему и получили общее название "угловая модуляция" (УМ; английский термин — *angle modulation*). Поэтому прежде всего в этом разделе нужно рассмотреть сходство и различие этих двух видов модуляции.

### Фазовая и частотная модуляция

Названия двух рассматриваемых видов модуляции, как и в случае с АМ, указывают параметр несущего колебания, который *линейно связан* с модулирующим сигналом.

#### Фазовая модуляция

Пусть модулирующий сигнал определяет начальную фазу несущего колебания:

$$\phi(t) = ks_M(t).$$

Тогда мы получаем сигнал с *фазовой модуляцией* (ФМ; английский термин — *phase modulation, PM*):

$$s_{\text{ФМ}}(t) = A \cos(\omega_0 t + ks_M(t)). \quad (8.7)$$

Весь аргумент функции  $\cos$ , взятый целиком, называется *полной фазой* колебания:

$$\Psi(t) = \omega_0 t + ks_M(t).$$

Круговая частота колебания по определению представляет собой скорость изменения начальной фазы. Подобно тому как в случае неравномерного движения вводится понятие мгновенной скорости (равной производной от координаты по времени), для колебаний с угловой модуляцией вводится понятие *мгновенной частоты* (*instantaneous frequency*), равной производной от полной фазы по времени:

$$\omega(t) = \frac{d\Psi}{dt} = \omega_0 + k \frac{ds_M}{dt}.$$

Итак, в случае фазовой модуляции изменяется не только начальная фаза, но и мгновенная частота колебания.

Соответственно, полная фаза может быть найдена путем интегрирования мгновенной частоты:

$$\Psi(t) = \int \omega(t) dt.$$

## Частотная модуляция

Теперь мы можем ввести понятие *частотной модуляции* (ЧМ; английский термин — *frequency modulation, FM*), при которой модулирующий сигнал линейно связан с мгновенной частотой колебания:

$$\omega(t) = \omega_0 + ks_M(t).$$

Добавка в виде константы  $\omega_0$  необходима для того, чтобы сделать колебание высокочастотным.

Полная фаза находится, как уже говорилось, путем интегрирования:

$$\Psi(t) = \omega_0 t + k \int s_M(t) dt + \phi_0. \quad (8.8)$$

Здесь  $\phi_0$  — произвольная постоянная интегрирования.

Наконец, сам ЧМ-сигнал имеет следующий вид:

$$s_{\text{ЧМ}}(t) = A \cos \left( \omega_0 t + k \int s_M(t) dt + \phi_0 \right).$$

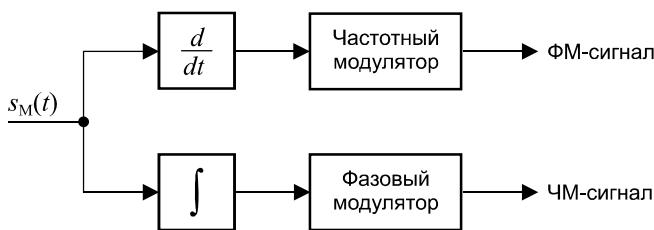
Как видим, начальная фаза колебания при частотной модуляции претерпевает изменения, пропорциональные интегралу от модулирующего сигнала:

$$\phi(t) = k \int s_M(t) dt + \phi_0.$$

Таким образом, частотная и фазовая модуляции оказываются взаимосвязанными: если изменяется начальная фаза колебания, изменяется и его мгновенная частота, и наоборот. По данной причине два этих вида модуляции и объединяют под общим названием "угловая модуляция".

Из сказанного можно сделать несколько выводов:

- По форме колебания с угловой модуляцией нельзя определить, ФМ это или ЧМ. Для этого необходимо знать еще и модулирующий сигнал.
- Если пропустить модулирующий сигнал через идеальное *дифференцирующее* устройство, а затем подать его на *частотный* модулятор, получится *фазовая* модуляция (верхняя ветвь на рис. 8.19).
- Если пропустить модулирующий сигнал через идеальное *интегрирующее* устройство, а затем подать его на *фазовый* модулятор, получится *частотная* модуляция (нижняя ветвь на рис. 8.19).



**Рис. 8.19.** Взаимосвязь фазовой и частотной модуляции

В заключение данного раздела приведем сводную таблицу (табл. 8.1), показывающую, как связаны с модулирующим сигналом различные характеристики модулированного колебания при фазовой и частотной модуляции.

**Таблица 8.1.** Связь характеристик модулированного колебания с модулирующим сигналом при фазовой и частотной модуляции

Параметр	ФМ	ЧМ
Начальная фаза	$\phi(t) = ks_M(t)$	$\phi(t) = k \int s_M(t) dt + \phi_0$
Полная фаза	$\Psi(t) = \omega_0 t + ks_M(t)$	$\Psi(t) = \omega_0 t + k \int s_M(t) dt + \phi_0$
Мгновенная частота	$\omega(t) = \omega_0 + k \frac{ds_M}{dt}$	$\omega(t) = \omega_0 + ks_M(t)$

## Гармоническая угловая модуляция

Аналогично тому, как мы это делали для амплитудной модуляции, рассмотрим случай гармонического модулирующего сигнала. Начальная фаза колебания изменяется при этом по гармоническому закону:

$$\phi(t) = \beta \sin(\Omega t).$$

Коэффициент  $\beta$  называется *индексом угловой модуляции* (*modulation index*). Он определяет интенсивность колебаний начальной фазы.

Полная фаза получится путем добавления линейного слагаемого  $\omega_0 t$ :

$$\Psi(t) = \omega_0 t + \beta \sin(\Omega t).$$

Наконец, сам сигнал с гармонической УМ:

$$s(t) = A \cos(\omega_0 t + \beta \sin(\Omega t)).$$

Как уже говорилось, при изменении начальной фазы изменяется и мгновенная частота:

$$\omega(t) = \frac{d\Psi(t)}{dt} = \omega_0 + \beta \Omega \cos(\Omega t).$$

В данном случае мгновенная частота меняется также по гармоническому закону. Как видно из полученной формулы, ее максимальное отклонение от среднего значения  $\omega_0$  составляет  $\beta\Omega$ . Эта величина называется *девиацией частоты* (*frequency deviation*) и обозначается  $\omega_d$ . Таким образом, мы получили важную формулу, показывающую, что индекс угловой модуляции равен отношению девиации частоты к частоте модулирующего сигнала:

$$\beta = \frac{\omega_d}{\Omega}.$$

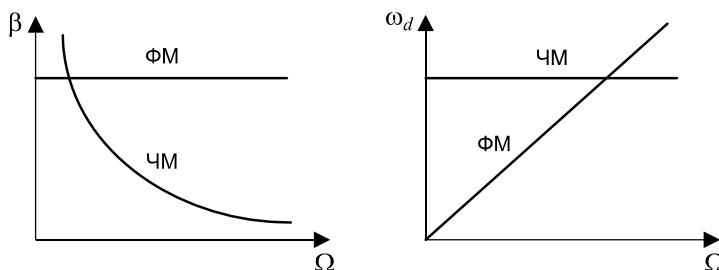
Итак, при гармонической УМ и начальная фаза, и мгновенная частота меняются по гармоническому закону. Различия между частотной и фазовой модуляцией начинают проявляться при изменении частоты модулирующего сигнала  $\Omega$ .

При ФМ индекс  $\beta$  является характеристическим параметром модуляции и от частоты модулирующего сигнала не зависит. Девиация частоты оказывается прямо пропорциональной  $\Omega$ :

$$\beta = \text{const}, \quad \omega_d = \beta \Omega.$$

При ЧМ характеристическим параметром, не зависящим от частоты модулирующего сигнала, является девиация частоты  $\omega_d$ . Индекс модуляции в этом случае оказывается обратно пропорциональным  $\Omega$ :

$$\omega_d = \text{const}, \quad \beta = \frac{\omega_d}{\Omega}.$$



**Рис. 8.20.** Зависимость индекса модуляции (слева) и девиации частоты (справа) от частоты гармонического модулирующего сигнала при ФМ и ЧМ

Зависимости индекса модуляции и девиации частоты от частоты модулирующего сигнала в случае ФМ и ЧМ представлены на рис. 8.20.

## Спектр сигнала с гармонической угловой модуляцией

В данном разделе мы сначала получим точное выражение для спектра радиосигнала с гармонической УМ, а затем рассмотрим его приближенный вариант, справедливый для малых индексов модуляции ( $\beta \ll 1$ ).

Для расчета спектрального представления сигнала с гармонической УМ мы прежде всего представим сигнал в виде вещественной части комплексной экспоненты:

$$\begin{aligned}s_{\text{УМ}}(t) &= \operatorname{Re}\left(A \exp\left(j\omega_0 t + j\varphi_0 + j\beta \sin(\Omega t + \Phi_0)\right)\right) = \\ &= \operatorname{Re}\left(A \exp\left(j\omega_0 t + j\varphi_0\right) \exp\left(j\beta \sin(\Omega t + \Phi_0)\right)\right).\end{aligned}$$

Теперь воспользуемся представлением выражения  $\exp(j\beta \sin x)$  в виде ряда Фурье:

$$\exp(j\beta \sin x) = \sum_{k=-\infty}^{\infty} J_k(\beta) \exp(jkx). \quad (8.9)$$

Здесь  $J_k(\beta)$  — функция Бесселя 1-го рода порядка  $k$  от аргумента  $\beta$ .

Используя представление (8.9), можно записать

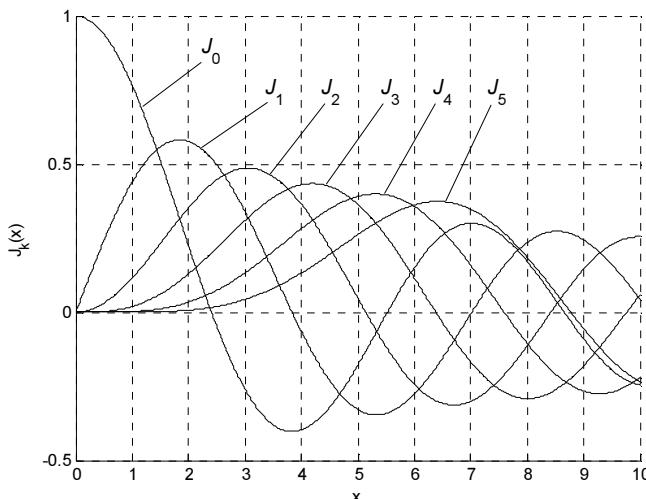
$$\begin{aligned}s_{\text{УМ}}(t) &= \operatorname{Re}\left(A \exp(j\omega_0 t + j\varphi_0) \sum_{k=-\infty}^{\infty} J_k(\beta) \exp(jk(\Omega t + \Phi_0))\right) = \\ &= A \sum_{k=-\infty}^{\infty} J_k(\beta) \cos((\omega_0 + k\Omega)t + \varphi_0 + k\Phi_0).\end{aligned} \quad (8.10)$$

Как видите, спектр сигнала содержит бесконечное количество составляющих с частотами  $\omega_0 + k\Omega$ ,  $k = 0, \pm 1, \pm 2, \dots$ . Амплитуда  $k$ -й составляющей равна  $AJ_k(\beta)$ , т. е. пропорциональна функции Бесселя  $k$ -го порядка, аргументом которой является индекс модуляции  $\beta$ .

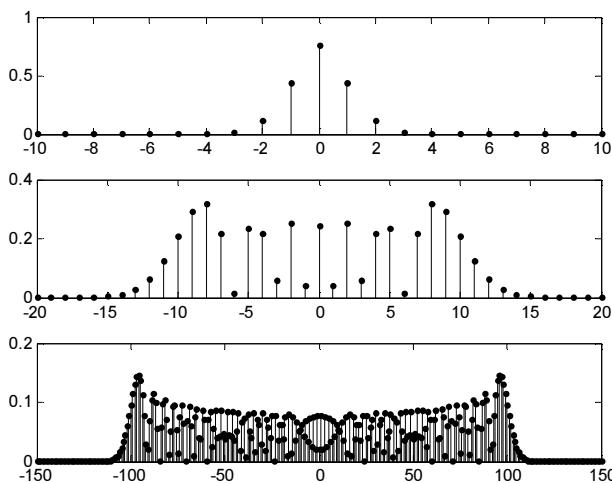
Функции Бесселя имеют колебательный характер (графики нескольких из них приведены на рис. 8.21), поэтому спектр при удалении от несущей частоты  $\omega_0$  спадает немонотонно. Для примера на рис. 8.22 приведены спектрограммы, соответствующие  $\beta = 1, 10$  и  $100$ . На рисунке предполагается, что  $\omega_0 \gg \Omega$ , так что наличием "хвостов", заползающих из области отрицательных частот, можно пренебречь:

```
>> % построение графиков функций Бесселя
>> x = 0:0.01:10;
>> figure
>> hold on
>> for k = 0:5, plot(x, besselj(k, x)), end
>> hold off
>> grid on
```

```
>> xlabel('x')
>> ylabel('J_{k}(x)')
```



**Рис. 8.21.** Функции Бесселя



**Рис. 8.22.** Амплитудный спектр сигнала с гармонической УМ при индексе модуляции, равном 1 (сверху), 10 (в центре) и 100 (снизу)

```
>> % спектрограммы УМ-сигналов с разным индексом модуляции
>> subplot(3, 1, 1)
>> k = -10:10;
>> stem(k, abs(besselj(k, 1)), '.')
>> subplot(3, 1, 2)
```

```

>> k = -20:20;
>> stem(k, abs(besselj(k, 10)), '.')
>> subplot(3, 1, 3)
>> k = -150:150;
>> stem(k, abs(besselj(k, 100)), '.')

```

Теперь приближенно рассмотрим частный случай малого индекса модуляции ( $\beta \ll 1$ ). Начнем с того, что применим к сигналу с гармонической УМ тригонометрические преобразования, чтобы раскрыть косинус суммы:

$$s_{\text{УМ}}(t) = A \cos(\omega_0 t + \phi_0) \cos(\beta \sin(\Omega t + \Phi_0)) - \\ - A \sin(\omega_0 t + \phi_0) \sin(\beta \sin(\Omega t + \Phi_0)).$$

Поскольку мы считаем, что  $\beta \ll 1$ , можно приближенно принять, что

$$\cos(\beta \sin(\Omega t + \Phi_0)) \approx 1,$$

$$\sin(\beta \sin(\Omega t + \Phi_0)) \approx \beta \sin(\Omega t + \Phi_0).$$

С учетом этого

$$s_{\text{УМ}}(t) \approx A \cos(\omega_0 t + \phi_0) - A \sin(\omega_0 t + \phi_0) \beta \sin(\Omega t + \Phi_0).$$

Остается представить последнее слагаемое в виде полуразности косинусов:

$$s_{\text{УМ}}(t) \approx A \cos(\omega_0 t + \phi_0) - \frac{A\beta}{2} \cos((\omega_0 + \Omega)t + \phi_0 + \Phi_0) + \\ + \frac{A\beta}{2} \cos((\omega_0 - \Omega)t + \phi_0 - \Phi_0).$$

Полученный результат сильно напоминает полученное ранее представление (8.3) для АМ-сигнала с гармонической модуляцией — тоже три составляющих с теми же частотами, да и амплитуды их рассчитываются аналогично (только вместо коэффициента амплитудной модуляции  $m$  в формуле фигурирует индекс угловой модуляции  $\beta$ ). Однако есть не слишком бросающееся в глаза, но тем не менее принципиальное отличие, превращающее амплитудную модуляцию в угловую — знак "минус" перед одним из слагаемых, соответствующих боковым частотам. Почему этот знак оказывается столь важным, мы увидим далее, при создании векторной диаграммы, а пока отметим главное: чтобы превратить сигнал с гармонической АМ в сигнал с гармонической УМ, достаточно изменить на  $180^\circ$  начальную фазу одной из боковых частот.

### **ЗАМЕЧАНИЕ**

Еще один способ превратить АМ в УМ — изменить фазу составляющей с несущей частотой на  $90^\circ$ . В том, почему это именно так, читателю предлагается разобраться самостоятельно.

Амплитудный и фазовый спектры сигнала с УМ при малом индексе модуляции (ее еще называют узкополосной УМ) показаны на рис. 8.23.

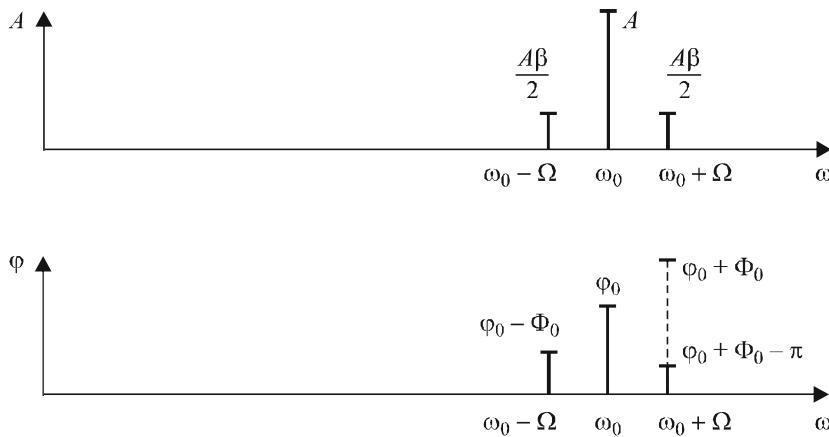


Рис. 8.23. Амплитудный (сверху) и фазовый (снизу) спектры сигнала с гармонической УМ при  $\beta \ll 1$

Теперь построим векторную диаграмму аналогично тому, как мы делали это для гармонической АМ. Поскольку спектральные представления сигналов с гармонической АМ и УМ различаются лишь знаком перед одним из слагаемых, подробно комментировать построение иллюстрации, показанной на рис. 8.24, нет необходимости. Скажем только, что для большей наглядности при построении выбрано довольно большое значение индекса модуляции ( $\beta = 0,5$ ).

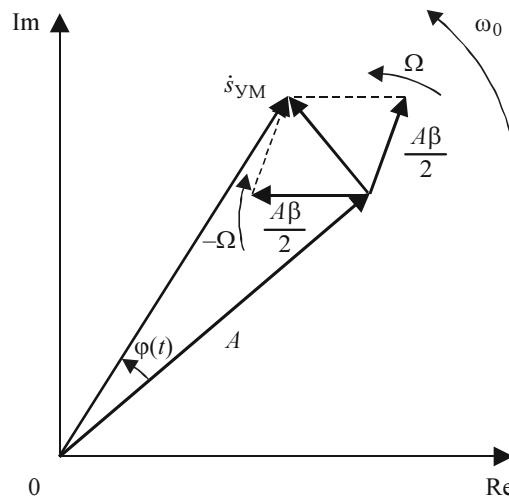


Рис. 8.24. Векторная диаграмма сигнала с гармонической УМ

### ЗАМЕЧАНИЕ

Из векторной диаграммы видно, что при сложении векторов меняется не только начальная фаза, но и амплитуда результирующего вектора. Это связано с тем, что формула, по которой производится построение диаграммы, является приближенной. Заинтересованный читатель может выполнить точный расчет амплитудной огибающей

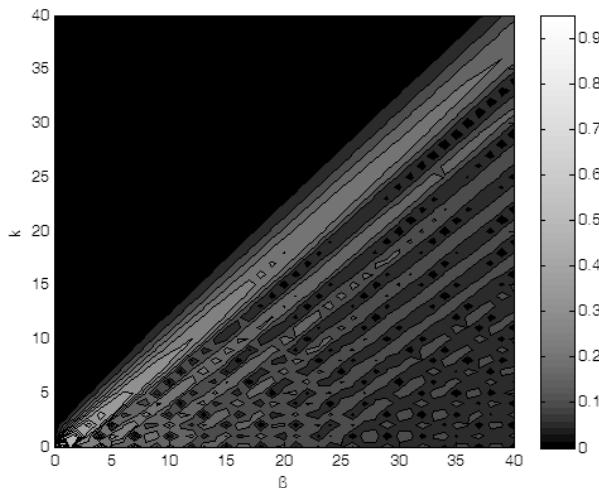
и фазовой функции данного сигнала, воспользовавшись преобразованием Гильберта (см. разд. "Комплексная огибающая" главы 1).

## Ширина спектра сигнала с гармонической УМ

При малом индексе модуляции, когда возможно использовать только что рассмотренное приближенное спектральное представление, ширину спектра можно, очевидно, принять равной  $2\Omega$  (то же имело место в случае АМ-сигнала). Сложнее оценить эффективную ширину спектра в случае большого индекса модуляции  $\beta$ . (В случае УМ речь идет именно об *эффективной* ширине спектра, поскольку спектр, строго говоря, содержит бесконечное число составляющих.)

При фиксированном аргументе  $\beta$  функции Бесселя затухают с ростом (по модулю) их порядка  $k$ . Попробуем на глаз оценить, как соотносятся индекс модуляции  $\beta$  и номер функции Бесселя  $k$ , начиная с которого абсолютные величины функций Бесселя становятся пренебрежимо малыми. Для этого построим линии равного уровня для взятых по модулю функций Бесселя в координатах  $(\beta, k)$  при изменении  $\beta$  и  $k$  от 0 до 40 (рис. 8.25):

```
>> N = 40; % предельное значение beta и k
>> for k = 0:N, x(k+1,:) = besselj(k, 0:N); end
>> contourf(0:N, 0:N, abs(x), [0:0.05:1])
>> xlabel('\beta')
>> ylabel('k')
>> colorbar
>> colormap gray
```



**Рис. 8.25.** Зависимость абсолютных значений функций Бесселя от аргумента  $\beta$  и порядка  $k$

Из графика видно, что с ростом порядка  $k$  при фиксированном аргументе  $\beta$  функции Бесселя затухают. Обычно считают, и рисунок это подтверждает, что при

оценке эффективной ширины спектра в случае  $\beta \gg 1$  можно пренебречь составляющими с номерами, по абсолютной величине превосходящими индекс угловой модуляции  $\beta$ . Таким образом, остаются составляющие, для которых  $|k| \leq \beta$ , и эффективная ширина спектра оказывается равной

$$\Delta\omega_{\text{эфф}} = 2\beta\Omega. \quad (8.11)$$

С учетом приведенного ранее соотношения, определяющего индекс угловой модуляции как отношение между девиацией частоты и частотой модулирующего сигнала, эффективная ширина спектра оказывается равной удвоенной девиации частоты:

$$\Delta\omega_{\text{эфф}} = 2\omega_d. \quad (8.12)$$

Итак, в зависимости от значения индекса модуляции  $\beta$  можно привести две формулы для расчета эффективной ширины спектра УМ-сигнала:

- при  $\beta \ll 1$  ширина спектра равна удвоенной частоте модулирующего сигнала:  $\Delta\omega_{\text{эфф}} = 2\Omega$ ;
- при  $\beta \gg 1$  ширина спектра равна удвоенной девиации частоты:  $\Delta\omega_{\text{эфф}} = 2\omega_d$ .

Критерии (8.11) и (8.12) часто встречаются в литературе. Подвернем их количественной проверке, построив два графика. Первый график будет показывать, какая доля средней мощности УМ-сигнала содержится в гармонических составляющих его спектра, ограниченных частотным диапазоном  $\omega_0 - \beta\Omega \dots \omega_0 + \beta\Omega$ . Для этого прежде всего заметим, что для УМ-сигнала средняя мощность рассчитывается так же, как и для обычной синусоиды — она равна  $A^2/2$ . Действительно, рассчитывая среднюю мощность по формуле (1.5), мы получаем

$$\begin{aligned} P_{\text{ср}} &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} (A \cos(\omega_0 t + \beta \sin(\Omega t)))^2 dt = \\ &= \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \frac{A^2}{2} dt + \lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \frac{A^2}{2} \cos(2\omega_0 t + 2\beta \sin(\Omega t)) dt. \end{aligned}$$

Подынтегральное выражение в первом интеграле не зависит от переменной интегрирования, поэтому предел оказывается равным

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \frac{A^2}{2} dt = \lim_{T \rightarrow \infty} \frac{1}{T} \frac{A^2 T}{2} = \frac{A^2}{2}.$$

Под вторым интегралом стоит быстроосциллирующая знакопеременная функция, поэтому с ростом  $T$  результат усреднения по интервалу  $-T \dots T$  стремится к нулю:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \int_{-T/2}^{T/2} \frac{A^2}{2} \cos(2\omega_0 t + 2\beta \sin(\Omega t)) dt = 0.$$

Таким образом, окончательно имеем  $P_{\text{ср}} = A^2/2$ .

Если в спектральном представлении (8.10) мы ограничим набор рассматриваемых гармонических слагаемых номерами  $k = -\beta \dots \beta$  (считаем здесь, что  $\beta$  — целое число), средняя мощность этого сигнала будет равна

$$P_{cp}(\beta) = \sum_{k=-\beta}^{\beta} \frac{A^2}{2} |J_k(\beta)|^2,$$

а доля полной средней мощности, соответственно, может быть рассчитана как

$$\frac{P_{cp}(\beta)}{P_{cp}} = \sum_{k=-\beta}^{\beta} |J_k(\beta)|^2.$$

Составим код MATLAB, позволяющий построить соответствующий график (рис. 8.26).

```
>> beta = 0:100; % диапазон значений индекса модуляции
>> for b = 1:length(beta)
>>     p(b) = sum(besselj(-b:b, b).^2);
>> end
>> plot(beta, p)
>> grid on
>> xlabel('\beta')
>> ylabel('P(\beta)/P_{cp}')
```

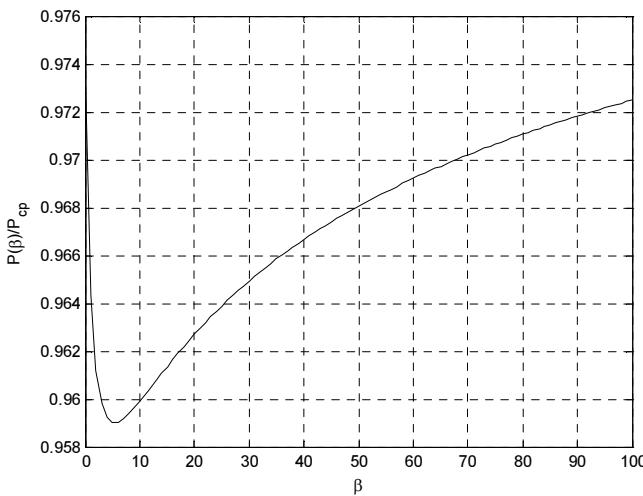


Рис. 8.26. Доля мощности УМ-сигнала, попадающая в диапазон частот, определенный согласно (8.11)

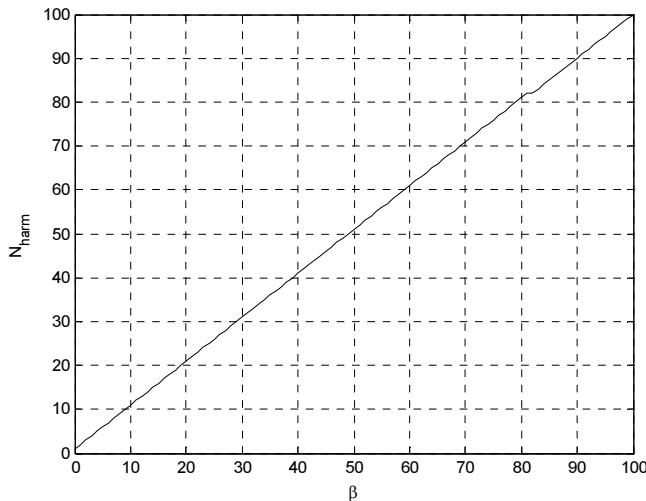
Как видно из рис. 8.26, для рассматриваемого диапазона значений индекса угловой модуляции  $\beta$  ширина спектра, определенная согласно (8.11) или (8.12), охватывает примерно 96 ... 97,5% полной мощности сигнала.

Второй график покажет, сколько гармонических составляющих нужно учитывать в (8.10), чтобы доля мощности получаемого при этом сигнала была не меньше заданного уровня. Для этого нужно найти минимальное значение  $N_{\text{harm}}$ , при котором выполняется неравенство

$$\sum_{k=-N_{\text{harm}}}^{N_{\text{harm}}} |J_k(\beta)|^2 \geq p_0,$$

где  $p_0$  — заданная доля мощности. Построим такой график с помощью MATLAB. Результат работы этого кода приведен на рис. 8.27.

```
>> p0 = 0.95; % требуемая доля полной мощности
>> beta = 0:100; % диапазон значений индекса модуляции
>> for b = 1:length(beta)
>>   k = 0;
>>   while (sum(besselj(-k:k,b).^2) < p0); k = k+1; end
>>   N_harm(b) = k;
>> end
>> plot(beta, N_harm)
>> grid on
>> xlabel('\beta')
>> ylabel('N_{\text{harm}}')
```



**Рис. 8.27.** Зависимость числа гармонических составляющих (в одну сторону от несущей), необходимого для перекрытия не менее 95% полной средней мощности сигнала с гармонической УМ

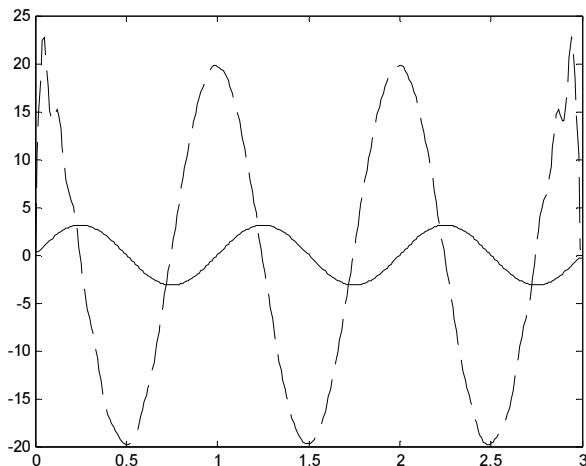
Из графика видно, что критерии (8.11), (8.12) вполне разумны — построенная кривая не имеет заметных отклонений от линейного закона  $N_{\text{harm}} = \beta$ . Заинтересован-

ный читатель может самостоятельно убедиться в том, что изменение требуемой доли мощности  $p_0$  лишь незначительно влияет на поведение графика.

Что же касается случая произвольного модулирующего сигнала, то, в отличие от АМ, для угловой модуляции получить аналитическое выражение для спектра при этом не удается. Даже попытка рассмотреть двухтональный модулирующий сигнал делает аналитические выкладки намного сложнее, чем при гармонической модуляции.

## Демодуляция УМ

Как и в случае АМ, демодуляция УМ-сигнала может выполняться различными способами. Наиболее радикальный подход — вычислить аналитический сигнал (см. разд. "Комплексная огибающая" главы 1) и выделить его фазовую функцию. Дальнейшие действия зависят от вида угловой модуляции. Для демодуляции ФМ из фазовой функции вычитается линейное слагаемое  $\omega_0 t$ , соответствующее немодулированной несущей. В случае ЧМ фазовая функция дифференцируется, а из результата вычитается константа  $\omega_0$  (рис. 8.28).



**Рис. 8.28.** Результаты фазовой (сплошная линия) и частотной (пунктирная линия) демодуляции сигнала с гармонической УМ

```
>> Fs = 100; % частота дискретизации
>> t = 0:1/Fs:3; % дискретное время
>> % сигнал с гармонической УМ
>> beta = pi; % индекс модуляции
>> Fc = 10; % несущая частота
>> F = 1; % частота модуляции
>> s = cos(2*pi*Fc*t + beta*sin(2*pi*F*t));;
>> y = hilbert(s); % аналитический сигнал
```

```
>> phi = unwrap(angle(y)); % фазовая функция
>> z_pm = phi - 2*pi*Fc*t; % демодуляция ФМ
>> z_fm = diff(phi)*Fs - 2*pi*Fc; % демодуляция ЧМ
>> plot(t, z_pm, t(1:end-1), z_fm, '--')
```

### ЗАМЕЧАНИЕ

В приведенном коде вычисление производной от полной фазы по времени аппроксимируется конечными разностями — `diff(phi)*Fs`.

Недостатком данного метода является обработка *всего входного сигнала сразу* с использованием преобразования Гильберта. Для реализации демодуляции в реальном масштабе времени необходима последовательная обработка поступающих отсчетов входного сигнала. Это можно реализовать, используя для получения аналитического сигнала приближенную гильбертовскую фильтрацию во временной области (синтез нерекурсивных и рекурсивных фильтров Гильберта был рассмотрен в *главе 6*).

Еще одной альтернативой, пригодной для реализации в реальном масштабе времени, является *квадратурная обработка*. При этом входной сигнал умножается на два опорных колебания, сдвиг по фазе между которыми составляет  $90^\circ$ :

$$\begin{aligned} y_I(t) &= s_{\text{УМ}}(t) \cos \omega_0 t = A \cos(\omega_0 t + \phi(t)) \cos \omega_0 t = \\ &= \frac{A}{2} \cos \phi(t) + \frac{A}{2} \cos(2\omega_0 t + \phi(t)), \\ y_Q(t) &= s_{\text{УМ}}(t) \sin \omega_0 t = A \cos(\omega_0 t + \phi(t)) \sin \omega_0 t = \\ &= -\frac{A}{2} \sin \phi(t) + \frac{A}{2} \sin(2\omega_0 t + \phi(t)). \end{aligned}$$

Каждый из результатов умножения содержит два слагаемых. Одно из них — низкочастотное (косинус или синус начальной фазы), другое — высокочастотное (УМ-сигнал с несущей частотой  $2\omega_0$ ). Низкочастотные составляющие выделяются с помощью ФНЧ:

$$y'_I(t) = \frac{A}{2} \cos \phi(t), \quad y'_Q(t) = -\frac{A}{2} \sin \phi(t).$$

Дальнейшие действия, так же как и раньше, зависят от вида угловой модуляции. Для демодуляции ФМ нам необходимо вычислить фазу полученной пары квадратурных составляющих:

$$\begin{aligned} x_{\Phi\text{М}}(t) &= -\arg(y'_I(t) + j y'_Q(t)) = \\ &= -\arg\left(\frac{A}{2} \cos \phi(t) - j \frac{A}{2} \sin \phi(t)\right) = \\ &= -\arg\left(\frac{A}{2} \exp(-j\phi(t))\right) = \phi(t). \end{aligned}$$

Для демодуляции ЧМ полученную фазовую функцию необходимо продифференцировать:

$$\begin{aligned} x_{\text{ЧМ}}(t) &= \frac{dx_{\Phi M}}{dt} = -\frac{d}{dt} \arg(y'_I(t) + jy'_Q(t)) = \\ &= -\frac{d}{dt} \arctg \frac{y'_Q(t)}{y'_I(t)} = \frac{\frac{dy'_I}{dt} y'_Q(t) - \frac{dy'_Q}{dt} y'_I(t)}{y'^2_I(t) + y'^2_Q(t)}. \end{aligned}$$

Структурная схема получившегося демодулятора показана на рис. 8.29.

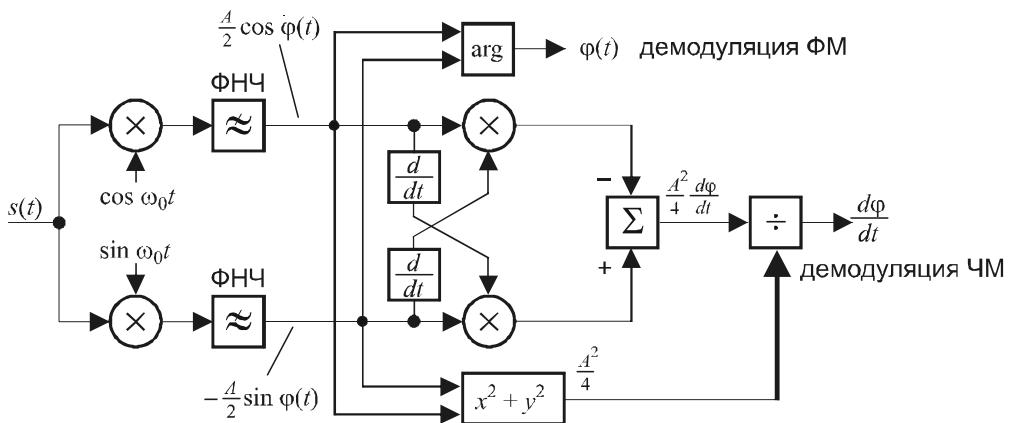


Рис. 8.29. Квадратурная обработка сигнала с угловой модуляцией

Достоинством данной схемы является то, что при высокой несущей частоте входные блоки (генератор опорного колебания и умножители) могут быть выполнены в аналоговом виде (см. разд. "Квадратурная дискретизация узкополосных сигналов" главы 3).

Еще одним способом демодуляции сигналов с УМ является использование следящих систем ФАПЧ. Рассмотрение таких систем выходит за рамки тематики данной книги.

## Квадратурная модуляция

В предыдущих разделах мы рассмотрели случаи, когда амплитуда и начальная фаза гармонического колебания подвергались модуляции по отдельности. Однако можно изменять эти два параметра одновременно, получив за счет этого возможность передавать два сигнала сразу:

$$s(t) = A(t) \cos(\omega_0 t + \phi(t)).$$

Такую модуляцию можно было бы назвать *амплитудно-фазовой*. Однако два модулирующих сигнала оказываются в данном случае "неравноправными", т. к. они модулируют существенно разные параметры несущего колебания. Можно сделать си-

туацию более "симметричной", слегка преобразовав форму представления рассматриваемого сигнала. Для начала раскроем косинус суммы:

$$s(t) = A(t) \cos(\omega_0 t) \cos(\phi(t)) - A(t) \sin(\omega_0 t) \sin(\phi(t)).$$

Теперь сигнал оказался представленным в виде суммы двух АМ-колебаний. Их несущие —  $\cos(\omega_0 t)$  и  $\sin(\omega_0 t)$  — сдвинуты по фазе на  $90^\circ$  друг относительно друга, а амплитудные функции равны  $A(t) \cos \phi(t)$  и  $-A(t) \sin \phi(t)$ . Обозначим эти амплитудные функции как  $a(t)$  и  $b(t)$  и используем их в качестве новой пары модулирующих сигналов (вместо амплитуды и начальной фазы):

$$s(t) = a(t) \cos(\omega_0 t) + b(t) \sin(\omega_0 t). \quad (8.13)$$

Такое представление рассматриваемого сигнала называется *квадратурным* (quadrature), а данный способ модуляции — *квадратурной модуляцией* (КАМ, английский термин — *quadrature amplitude modulation*, QAM). Модулирующие сигналы при этом оказываются совершенно равноправными.

#### **ЗАМЕЧАНИЕ**

Следует подчеркнуть, что квадратурная модуляция и амплитудно-фазовая модуляция — это разные представления одного и того же сигнала. Различие между ними состоит только в том, как пара модулирующих сигналов управляет параметрами результирующего колебания.

## **Спектр сигнала с квадратурной модуляцией**

В случае амплитудно-фазового представления сигнала записать аналитическое выражение для спектра не представляется возможным. А вот для квадратурного представления получить спектральную функцию не составляет труда.

Поскольку КАМ-сигнал представляет собой сумму двух АМ-сигналов, мы можем воспользоваться формулой (8.4) и сразу же записать

$$\dot{S}(\omega) = \frac{1}{2} \dot{A}(\omega + \omega_0) + \frac{1}{2} \dot{A}(\omega - \omega_0) - \frac{1}{2} j \dot{B}(\omega + \omega_0) + \frac{1}{2} j \dot{B}(\omega - \omega_0).$$

Итак, аналогично тому, что происходит при амплитудной модуляции, спектры модулирующих сигналов "раздваиваются" и "переезжают" в окрестности частоты несущей  $\pm \omega_0$ . Если спектры модулирующих сигналов  $a(t)$  и  $b(t)$  занимают одну и ту же полосу частот (как обычно и бывает), то они будут перекрываться и после сдвига в область несущей частоты. Однако при этом спектр, соответствующий синусной несущей, дополнительно умножается на  $\pm j$ . Именно это дает возможность разделить квадратурные составляющие при приеме сигнала.

## **Демодуляция сигнала с квадратурной модуляцией**

Как и другие разновидности АМ, квадратурно-модулированный сигнал может быть демодулирован путем умножения на опорное колебание. Однако поскольку КАМ-

сигнал представляет собой сумму двух АМ-сигналов, то и опорных колебаний должно быть два — со сдвигом фаз на  $90^\circ$ :

$$\begin{aligned} y_1(t) &= s_{\text{KAM}}(t) \cos \omega_0 t = (a(t) \cos \omega_0 t + b(t) \sin \omega_0 t) \cos \omega_0 t = \\ &= \frac{1}{2} a(t) + \frac{1}{2} a(t) \cos 2\omega_0 t + \frac{1}{2} b(t) \sin 2\omega_0 t, \\ y_Q(t) &= s_{\text{KAM}}(t) \sin \omega_0 t = (a(t) \cos \omega_0 t + b(t) \sin \omega_0 t) \sin \omega_0 t = \\ &= \frac{1}{2} b(t) + \frac{1}{2} a(t) \sin 2\omega_0 t - \frac{1}{2} b(t) \cos 2\omega_0 t. \end{aligned} \quad (8.14)$$

Результат каждого умножения содержит три слагаемых. Одно из них является низкочастотным и представляет собой модулирующую функцию  $a(t)$  или  $b(t)$  с уменьшенным вдвое уровнем. Остальные два слагаемых образуют КАМ-сигнал с несущей частотой  $2\omega_0$ . Поэтому полезные составляющие легко выделяются путем пропускания результатов умножения через ФНЧ. Структура демодулятора показана на рис. 8.30, а.

Если не ограничиваться вещественными сигналами, можно представить демодуляцию КАМ несколько проще. При этом требуется всего один канал, а опорным колебанием служит комплексная экспонента  $\exp(j\omega_0 t)$ :

$$\begin{aligned} y(t) &= s_{\text{KAM}}(t) \exp(j\omega_0 t) = (a(t) \cos \omega_0 t + b(t) \sin \omega_0 t) \exp(j\omega_0 t) = \\ &= \frac{1}{2} (a(t) + jb(t)) + \frac{1}{2} (a(t) - jb(t)) \exp(j2\omega_0 t). \end{aligned}$$

Результат комплексного умножения содержит низкочастотное слагаемое, представляющее обе модулирующие функции, и высокочастотное комплексное слагаемое, содержащее множитель  $\exp(j2\omega_0 t)$ . Низкочастотное слагаемое, как обычно, выделяется с помощью ФНЧ. Структура комплексного варианта демодулятора показана на рис. 8.30, б. Вещественная и мнимая части комплексного выходного сигнала соответствуют двум выходным сигналам вещественного демодулятора.

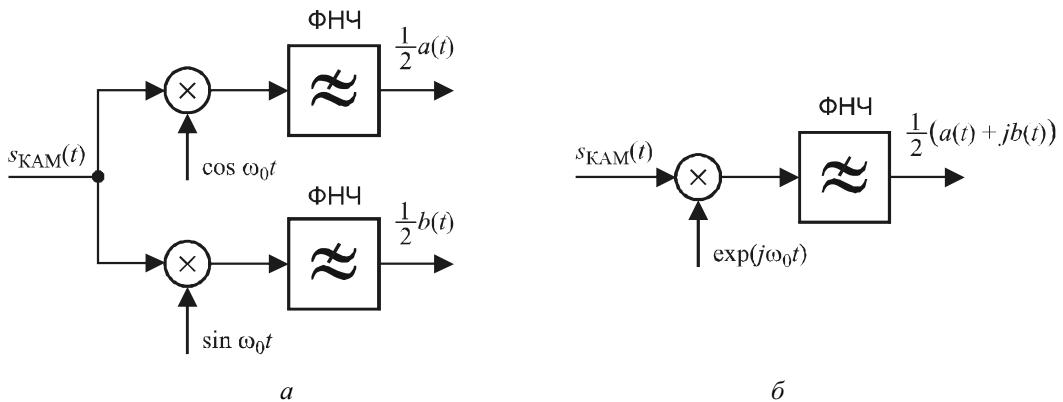


Рис. 8.30. Демодуляция КАМ-сигнала в вещественном (а) и комплексном (б) представлении

При демодуляции очень важно точное соблюдение частоты и начальной фазы опорного колебания. При наличии фазовой ошибки  $\Delta\phi$  на выходах демодулятора будут получены линейные комбинации модулирующих функций:

$$\operatorname{Re}((a(t) + jb(t))\exp(j\Delta\phi)) = a(t)\cos\Delta\phi - b(t)\sin\Delta\phi,$$

$$\operatorname{Im}((a(t) + jb(t))\exp(j\Delta\phi)) = a(t)\sin\Delta\phi + b(t)\cos\Delta\phi.$$

При наличии ошибки по частоте фазовый сдвиг линейно меняется во времени, так что достаточно подставить в предыдущую формулу  $\Delta\phi = \Delta\omega t$ :

$$\operatorname{Re}((a(t) + jb(t))\exp(j\Delta\omega t)) = a(t)\cos\Delta\omega t - b(t)\sin\Delta\omega t,$$

$$\operatorname{Im}((a(t) + jb(t))\exp(j\Delta\omega t)) = a(t)\sin\Delta\omega t + b(t)\cos\Delta\omega t.$$

Таким образом, выходные сигналы демодулятора оказываются промодулированы по амплитуде частотой *биений*.

## Способы модуляции, используемые при передаче цифровой информации

В настоящее время все большая часть информации, передаваемой по разнообразным каналам связи, существует в цифровом виде. Это означает, что передаче подлежит не непрерывный (аналоговый) модулирующий сигнал, а последовательность целых чисел  $n_0, n_1, n_2, \dots$ , которые могут принимать значения из некоторого фиксированного конечного множества. Эти числа, называемые *символами* (*symbol*), поступают от источника информации с периодом  $T$ , а частота, соответствующая этому периоду, называется *символьной скоростью* (*symbol rate*):  $f_{\text{sym}} = 1/T$  (для ее обозначения в приводимых далее MATLAB-примерах будет использоваться идентификатор `Fd`).

### ЗАМЕЧАНИЕ

Часто используемым на практике вариантом является *двоичная* (*binary*) последовательность символов, когда каждое из чисел  $n_i$  может принимать одно из двух значений — 0 или 1.

Последовательность передаваемых символов является, очевидно, дискретным сигналом. Поскольку символы принимают значения из конечного множества, этот сигнал фактически является и *квантованным*, т. е., согласно определениям, введенным в разд. "Аналоговые, дискретные и цифровые сигналы" главы 3, его можно назвать *цифровым* сигналом. Далее в этой главе будут рассматриваться вопросы, связанные с преобразованием этого цифрового сигнала в аналоговый модулированный сигнал.

Типичный подход при осуществлении передачи дискретной последовательности символов состоит в следующем. Каждому из возможных значений символа сопос-

тавляется некоторый набор параметров несущего колебания. В простейшем варианте эти параметры поддерживаются постоянными в течение интервала  $T$ , т. е. до прихода следующего символа. Фактически это означает преобразование последовательности чисел  $\{n_k\}$  в ступенчатый сигнал  $s_n(t)$  с использованием кусочно-постоянной интерполяции:

$$s_n(t) = f(n_k), \quad kT \leq t < (k+1)T.$$

Здесь  $f$  — некоторая функция преобразования. Полученный сигнал  $s_n(t)$  далее используется в качестве модулирующего сигнала обычным способом.

Такой способ модуляции, когда параметры несущего колебания меняются скачкообразно, называется *манипуляцией (keying)*. В зависимости от того, какие именно параметры изменяются, различают амплитудную (АМн), фазовую (ФМн), частотную (ЧМн) и квадратурную (КАМ) манипуляцию. Кроме того, при передаче цифровой информации может использоваться несущее колебание, отличное по форме от гармонического. Так, при использовании в качестве несущего колебания последовательности прямоугольных импульсов возможны амплитудно-импульсная (АИМ), широтно-импульсная (ШИМ) и времяимпульсная (ВИМ) модуляция.

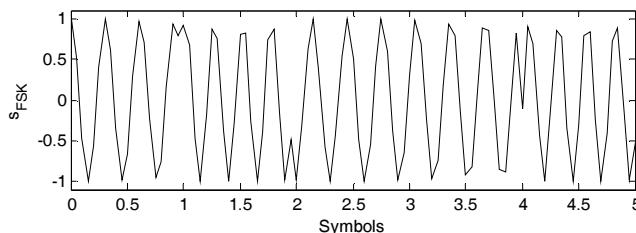
## Частотная манипуляция

При частотной манипуляции (ЧМн; английский термин — *frequency shift keying, FSK*) каждому возможному значению передаваемого символа сопоставляется своя частота. В течение каждого символьного интервала передается гармоническое колебание с частотой, соответствующей текущему символу.

В качестве примера сформируем 2-позиционный (бинарный) ЧМн-сигнал, параметры которого соответствуют старому модемному стандарту V.21. Для передачи битов со значениями 0 и 1 используются частоты 980 и 1180 Гц, символьная скорость равна 300 символам в секунду, а частота дискретизации составляет 6 кГц (рис. 8.31):

```
>> bits = [0 1 0 0 1 randint(1, 95)]; % цифровое сообщение
>> N = length(bits); % длина сообщения
>> Fd = 300; % символьная скорость
>> FsFd = 20; % отношение Fs/Fd
>> Fs = Fd * FsFd; % частота дискретизации
>> f = [980 1180]; % частоты манипуляции
>> t = (0:N*FsFd-1)/Fs; % дискретное время
>> carriers = cos(2*pi*t'*f); % столбцы - колебания для "0" и "1"
>> mask = repmat(bits, FsFd, 1); % маска переключения колебаний
>> mask = mask(:); % "растягиваем" маску в один столбец
>> % Формируем ЧМн-сигнал, переключая колебания
>> s_fsk = carriers(:,1) .* (1-mask) + carriers(:,2) .* mask;
>> td = (0:N*FsFd-1)/FsFd; % время для графика - в символах
>> Ng = 5; % число символов, выводимых на график
>> plot(td(1:Ng*FsFd+1), s_fsk(1:Ng*FsFd+1))
```

```
>> xlabel('Symbols')
>> ylabel('s_{FSK}')
>> ylim([-1.1 1.1])
```



**Рис. 8.31.** Частотно-манипулированный сигнал

### ЗАМЕЧАНИЕ

Первые пять бит сообщения задаются принудительно, чтобы на выводимом фрагменте графика обязательно были представлены и нулевые, и единичные биты. Полный же сигнал должен быть достаточно длинным и случайным — это необходимо для последующего спектрального анализа. Поэтому остаток 100-битового сообщения формируется с помощью функции `randint` из пакета Communications, генерирующей случайные целые числа.

Поскольку мы формировали посылки, коммутируя два независимых колебания, получившийся сигнал содержит скачки — это хорошо видно на графике. В разд. "Примеры разложения сигналов в ряд Фурье" главы I было показано, что чем более гладким является сигнал, тем быстрее убывает его спектр. Поэтому можно попытаться сделать спектр ЧМн-сигнала компактнее, изменив способ формирования так, чтобы устраниТЬ скачки.

Такой способ формирования сигнала называется *частотной манипуляцией с непрерывной фазой* (*continuous phase frequency shift keying, CPFSK*). При этом формируется линейно меняющаяся полная фаза колебания, а передаваемые символы управляют скоростью ее изменения. (Можно сказать и так: передаваемые символы переключают значение мгновенной частоты; эта частота интегрируется, давая непрерывную фазовую функцию; косинус такой полной фазы тоже будет непрерывной функцией.) Сформируем указанным способом ЧМн-сигнал с теми же параметрами, что и в предыдущем примере (рис. 8.32):

```
>> pps = 2*pi*f/Fs; % сдвиг фазы на один отсчет
>> s1 = repmat(pps, FsFd, 1); % столбцы - сдвиги для 0 и 1
>> d_cpfesk = s1(:, bits+1); % каждый столбец - один символ
>> d_cpfesk = d_cpfesk(:); % "растягиваем" в один столбец
>> phi_cpfesk = cumsum(d_cpfesk); % интегрируем фазовые сдвиги
>> s_cpfesk = cos(phi_cpfesk); % ЧМн-сигнал
>> plot(td(1:Ng*FsFd-1), s_cpfesk(1:Ng*FsFd-1))
>> xlabel('Symbols')
>> ylabel('s_{CPFSK}')
>> ylim([-1.1 1.1])
```

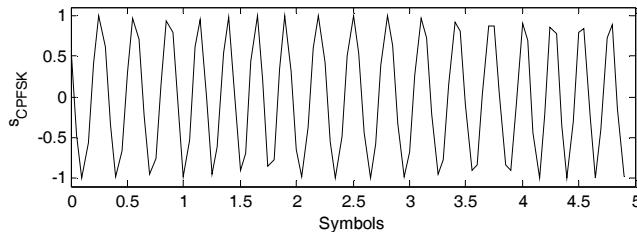


Рис. 8.32. ЧМн-сигнал с непрерывной фазовой функцией

Как видите, скачки исчезли. Теперь сравним спектры мощности при двух вариантах формирования ЧМн-сигнала (рис. 8.33):

```
>> pwelch(s_fsk, [], [], [], Fs)
>> ylim([-80 -20])
>> figure
>> pwelch(s_cpfk, [], [], [], Fs)
>> ylim([-80 -20])
```

Сравнение верхнего и нижнего графиков рис. 8.33 показывает, что при наличии скачков фазы спектр сигнала затухает существенно медленнее, чем в случае непрерывной фазы. Кроме того, в последнем случае в спектре отсутствуют пики на частотах манипуляции (980 и 1180 Гц).

Прием ЧМн-сигнала, как правило, осуществляется корреляционным методом. Сущность его состоит в вычислении взаимной корреляции (см. разд. "Корреляционный анализ" главы 1) между принимаемым сигналом и колебаниями-образцами (опорными сигналами), представляющими собой гармонические колебания с используемыми для манипуляции частотами. В качестве выходного символа выбирается тот, частота которого оказывается максимально коррелирована с входным сигналом.

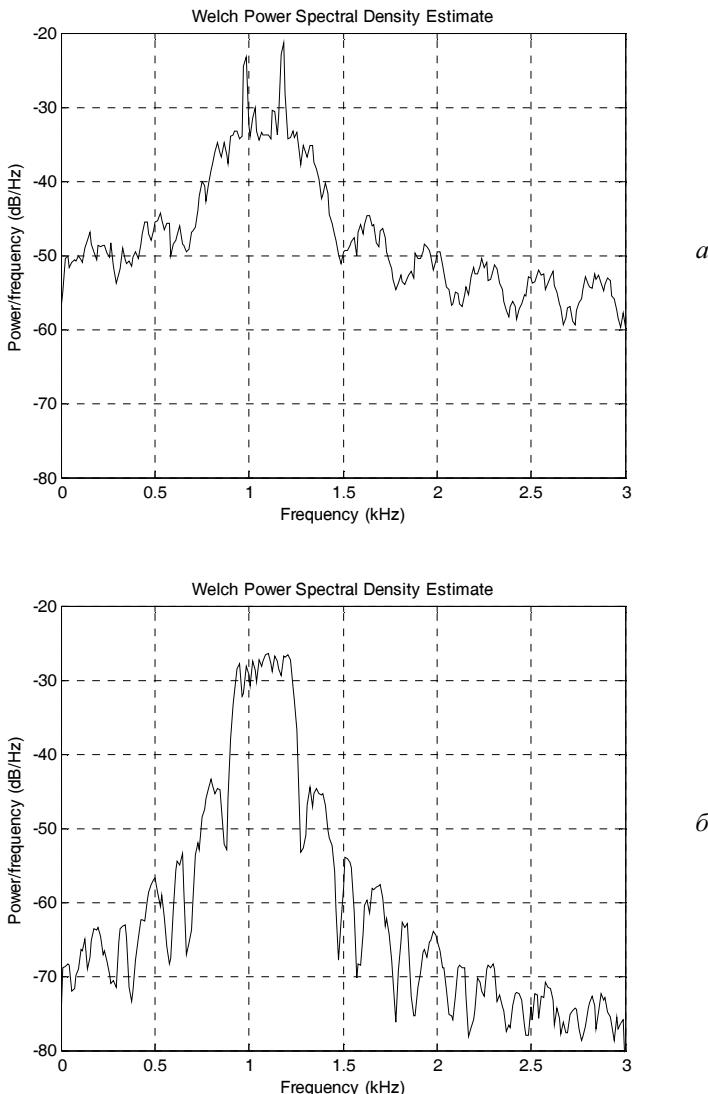
Корреляционный прием может быть *когерентным* или *некогерентным*. Когерентный алгоритм может использоваться, если известна начальная фаза колебания. Опорными сигналами при этом служат вещественные синусоиды с нужными частотами и начальными фазами. Для принятия решения о принятом символе сравниваются вещественные результаты вычисления корреляционных сумм.

На практике, однако, начальная фаза, как правило, неизвестна. В таких случаях применяют некогерентный корреляционный прием, при котором опорные сигналы представляют собой комплексные экспоненты с нужными частотами. Для принятия решения о принятом символе сравниваются *модули* комплексных результатов вычисления корреляционных сумм. Модуль комплексной ВКФ не зависит от начальных фаз сигналов, однако помехоустойчивость такого алгоритма несколько хуже, чем в случае когерентного приема.

Реализуем когерентную демодуляцию для первого варианта формирования ЧМн-сигнала (с разрывами начальной фазы):

```
>> z0 = sum(buffer(s_fsk.*carriers(:,1), FsFd)); % корреляция с "0"
>> z1 = sum(buffer(s_fsk.*carriers(:,2), FsFd)); % корреляция с "1"
```

```
>> x = double(z1 > z0); % решения о принятых символах
>> symerr(bits, x)      % число ошибок приема
ans =
0
```



**Рис. 8.33.** Оценки СПМ ЧМн-сигнала при наличии (a) и отсутствии (б) разрывов у фазовой функции

В этом коде сигнал умножается на опорные колебания из матрицы `carriers`, затем производится "нарезка" результата на символы функцией `buffer`, после чего реализуется суммирование значений в пределах длительности каждого символа с помощью функции `sum`. Как видите, сигнал принят без ошибок.

Теперь демодулируем сигнал с непрерывной начальной фазой некогерентным методом:

```
>> y = buffer(s_cpfsk, FsFd); % каждый столбец - символ
>> t = (0:FsFd-1)/Fs; % дискретное время для одного символа
>> s0_exp = exp(i*2*pi*t'*f); % столбцы - опорные сигналы
>> z = s0_exp.' * y; % корреляции с "0" (строка 1) и с "1" (2)
>> x = double(abs(z(2,:)) > abs(z(1,:))); % решения о принятых символах
>> symerr(bits, x) % число ошибок приема
ans =
0
```

Ошибок в этом случае нет.

## Минимальная частотная манипуляция

Для повышения помехоустойчивости ЧМн желательно, чтобы посылки, соответствующие разным символам, были некоррелированы (см. разд. "Корреляционный анализ" главы 1). Считая начальные фазы посылок нулевыми, ЧМн-сигналы для символов 0 и 1 можно записать так:

$$s_0(t) = A \cos \omega_0 t, \quad 0 \leq t \leq T,$$

$$s_1(t) = A \cos \omega_1 t, \quad 0 \leq t \leq T.$$

Их ВКФ при нулевом временном сдвиге равна

$$\begin{aligned} B_{01}(0) &= \int_0^T s_0(t)s_1(t)dt = A^2 \int_0^T \cos \omega_0 t \cos \omega_1 t dt = \\ &= \frac{A^2 \sin(\omega_1 + \omega_0)T}{2(\omega_1 + \omega_0)} + \frac{A^2 \sin(\omega_1 - \omega_0)T}{2(\omega_1 - \omega_0)}. \end{aligned}$$

Если  $(\omega_1 + \omega_0)T \gg 1$ , первое слагаемое значительно меньше второго и им можно пренебречь:

$$B_{01}(0) \approx \frac{A^2 \sin(\omega_1 - \omega_0)T}{2(\omega_1 - \omega_0)}.$$

Это значение равно нулю при  $(\omega_1 - \omega_0)T = \pi k$ , где  $k$  — целое число, не равное нулю. Таким образом, минимальное значение расстояния между частотами манипуляции, при котором посылки, соответствующие разным символам, оказываются некоррелированными, составляет

$$\Delta\omega_{\min} = \frac{\pi}{T}, \quad \Delta f_{\min} = \frac{1}{2T} = \frac{f_T}{2}, \quad (8.15)$$

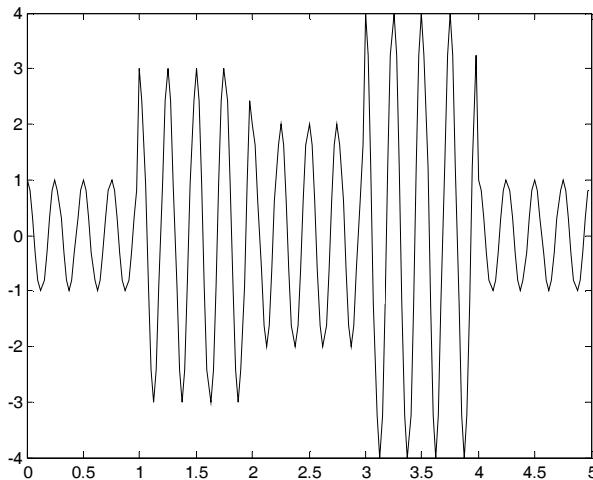
где  $f_T = 1/T$  — символьная скорость.

Двухпозиционная (бинарная) ЧМн с непрерывной фазой, частоты которой выбраны согласно (8.15), получила название *минимальной частотной манипуляции* (*MЧМн*, английский термин — *minimum shift keying, MSK*).

## Амплитудная манипуляция

Как будет показано далее, *амплитудная манипуляция* (*AMн*; английский термин — *amplitude shift keying, ASK*), при которой скачкообразно меняется амплитуда несущего колебания, является частным случаем цифровой квадратурной модуляции. Поэтому здесь мы только построим в качестве примера график АМн-сигнала (рис. 8.34):

```
>> sy = [1 3 2 4 1]; % передаваемые символы
>> Fd = 1; % символьная скорость
>> Fc = 4; % несущая частота
>> FsFd = 40; % отношение Fs/Fd
>> Fs = Fd * FsFd; % частота дискретизации
>> t = (0:length(sy)*FsFd-1)/Fs; % дискретное время
>> % формируем АМн-сигнал
>> s_ask = sy(floor(Fd*t)+1) .* cos(2*pi*Fc*t);
>> plot(t, s_ask)
```



**Рис. 8.34.** Сигнал с амплитудной манипуляцией

## Фазовая манипуляция

*Фазовая манипуляция* (*ФМн*; английский термин — *phase shift keying, PSK*), при которой скачкообразно меняется фаза несущего колебания, тоже является частным случаем цифровой квадратурной модуляции.

На практике фазовая манипуляция используется при небольшом числе возможных значений начальной фазы — как правило, 2, 4 или 8. Кроме того, при приеме сигнала сложно измерить *абсолютное* значение начальной фазы; значительно проще определить *относительный* фазовый сдвиг между двумя соседними символами.

Поэтому обычно используется *фазоразностная манипуляция* (синонимы — *дифференциальная фазовая манипуляция*, *относительная фазовая манипуляция*; английский термин — *differential phase shift keying, DPSK*).

Построим график сигнала с 4-позиционной фазовой манипуляцией (рис. 8.35):

```
>> sy = [0 2 1 3 0]; % передаваемые символы
>> Fd = 1; % символьная скорость
>> Fc = 4; % несущая частота
>> FsFd = 40; % отношение Fs/Fd
>> Fs = Fd * FsFd; % частота дискретизации
>> t = (0:length(sy)*FsFd-1)/Fs; % дискретное время
>> % формируем ФМн-сигнал
>> s_psk = cos(2*pi*Fc*t + pi/2*sy(floor(Fd*t)+1));
>> plot(t, s_psk)
>> xlabel('Symbols')
>> ylabel('s_{PSK}')
>> ylim([-1.1 1.1])
```

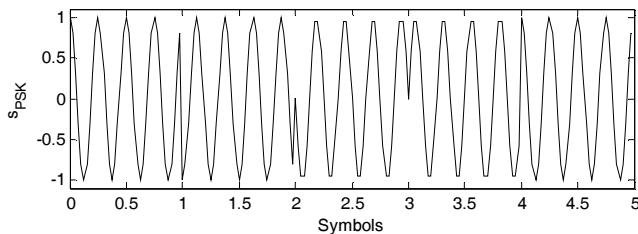


Рис. 8.35. Сигнал с 4-позиционной фазовой манипуляцией

## Квадратурная модуляция

При цифровой *квадратурной модуляции* (*KAM*; английский термин — *quadrature amplitude modulation, QAM*) каждому из возможных значений дискретного символа  $C_k$  ставится в соответствие *пара* величин — амплитуды синфазной и квадратурной составляющих либо, что эквивалентно, амплитуда и начальная фаза несущего колебания:

$$C_k \rightarrow (a_k, b_k), \quad s(t) = a_k \cos \omega_0 t + b_k \sin \omega_0 t, \quad kT \leq t < (k+1)T$$

или

$$C_k \rightarrow (A_k, \varphi_k), \quad s(t) = A_k \cos(\omega_0 t + \varphi_k), \quad kT \leq t < (k+1)T.$$

Параметры аналогового колебания, сопоставленные дискретному символу  $C_k$ , удобно представлять в виде комплексного числа в алгебраической ( $a_k + jb_k$ ) или экспоненциальной ( $A_k \exp(j\varphi_k)$ ) форме. Совокупность этих комплексных чисел для всех возможных значений дискретного символа называется *сигнальным созвездием* (*constellation*).

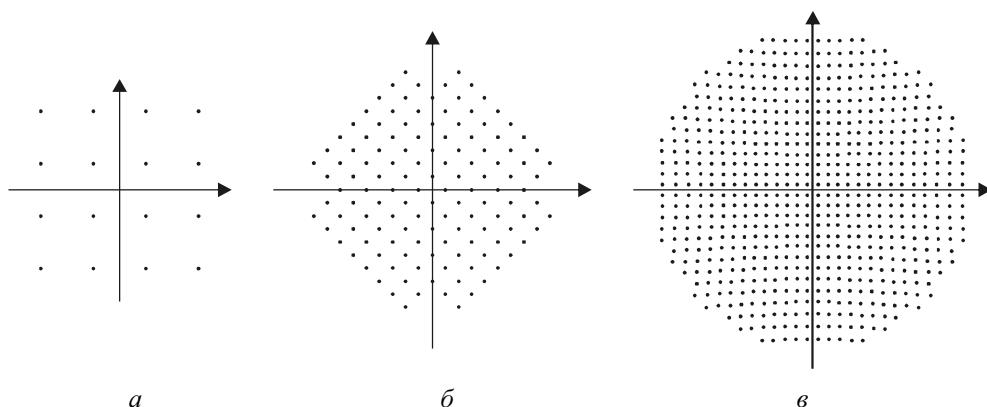
### ЗАМЕЧАНИЕ

В литературных источниках также используются термины "сигнальная конструкция" (*signal structure*) и "пространственная диаграмма" (*signal space diagram*). В данной книге сделан выбор в пользу более выразительного и запоминающегося слова — "созвездие".

При представлении дискретного символа комплексным числом  $\dot{C}_k$  сигнал с квадратурной модуляцией можно записать следующим образом:

$$s(t) = \operatorname{Re}(\dot{C}_k \exp(-j\omega_0 t)), \quad kT \leq t < (k+1)T.$$

На практике используются созвездия, содержащие от четырех точек до нескольких тысяч. На рис. 8.36 показаны некоторые созвездия, используемые modemами, предназначенными для передачи данных по телефонным линиям.



**Рис. 8.36.** Примеры созвездий, используемых при квадратурной модуляции:  
а — 16 точек (протокол V.32, скорость 9600 бит/с), б — 128 точек (протокол V.32bis, скорость 14 400 бит/с), в — 640 точек (протокол V.34, скорость 28 800 бит/с)

График сигнала с квадратурной модуляцией оказывается не очень наглядным из-за смешанного (амплитудно-фазового) характера модуляции. Изменения амплитуды и фазы при переходе от символа к символу могут быть небольшими и плохо заметными на графике.

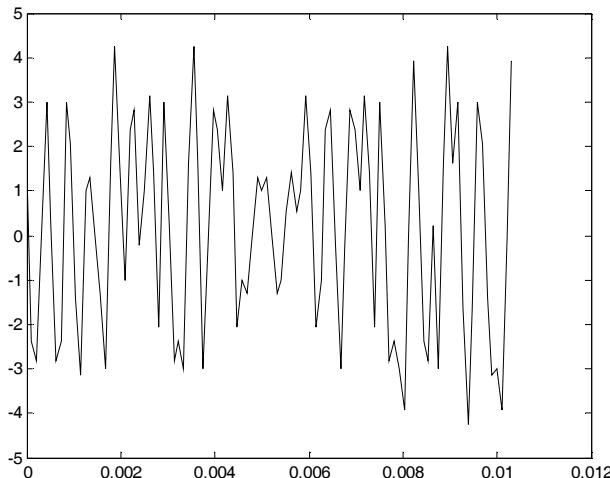
Построим тем не менее график сигнала, сформированного с использованием 16-точечного "квадратного" созвездия, показанного на рис. 8.36, а. Поскольку нас сейчас не интересует конкретный способ связи дискретных символов и точек созвездия, мы просто создадим векторы амплитуд синфазной и квадратурной составляющих, значения которых случайно выбраны из набора  $\{-3, -1, 1, 3\}$  (рис. 8.37):

```
>> N = 1000; % число символов
>> aa = randint(1, N, 4); % случайные целые числа 0...3
>> bb = randint(1, N, 4);
>> a1 = 2*aa-3; % преобразуем к требуемому набору
>> b1 = 2*bb-3;
```

```

>> Fd = 2400; % символьная скорость
>> Fc = 1800; % несущая частота
>> FsFd = 4; % число отсчетов на один символ
>> Fs = Fd * FsFd; % частота дискретизации
>> % дублируем каждый отсчет FsFd раз
>> a1 = repmat(a1, FsFd, 1);
>> a1 = a1(:);
>> b1 = repmat(b1, FsFd, 1);
>> b1 = b1(:);
>> % формируем аналоговый сигнал
>> t = (0:N*FsFd-1)/Fs; % дискретное время
>> t = t'; % превращаем строку в столбец
>> s_qam16 = a1 .* cos(2*pi*Fc*t) + b1 .* sin(2*pi*Fc*t);
>> plot(t(1:100), s_qam16(1:100))

```



**Рис. 8.37.** Сигнал с 16-позиционной квадратурной модуляцией

Параметры сформированного сигнала (структура созвездия, значения символьной скорости и несущей частоты) соответствуют модему, передающему данные со скоростью 9600 бит/с в соответствии с Рекомендацией ITU-T V.32. Прослушаем сигнал, используя для этого функцию `soundsc`, чтобы не заботиться о приведении сигнала к диапазону уровней  $-1 \dots 1$ :

```
>> soundsc(repmat(s_qam16, 10, 1), Fs)
```

### ЗАМЕЧАНИЕ

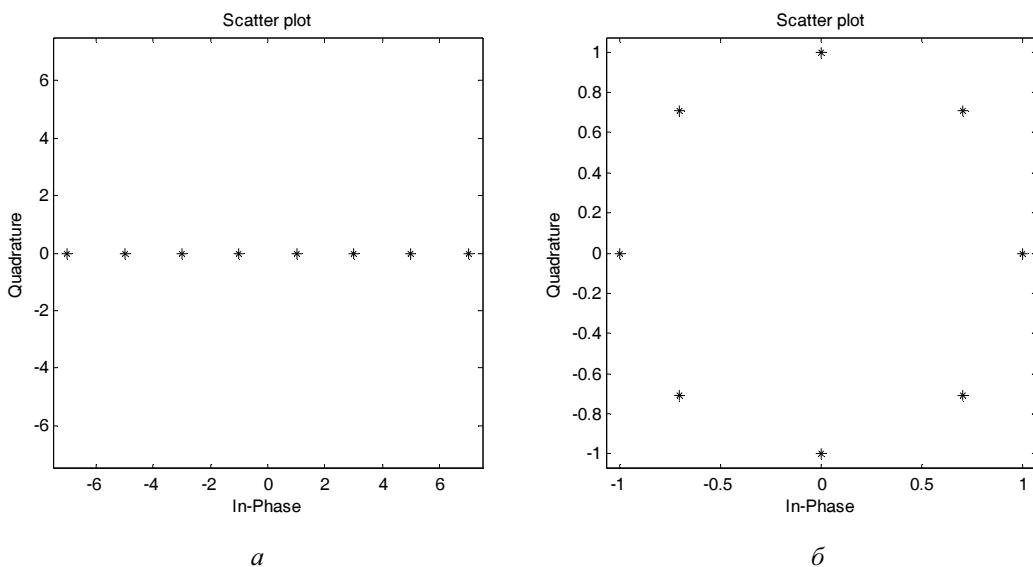
Функция `repmat` использована здесь, чтобы повторить сформированный сигнал десять раз — иначе звук окажется слишком коротким.

Если вы когда-нибудь слышали шуршащий звук модема, то должны заметить, что в сформированном нами сигнале что-то не так. Действительно, на практике при осу-

ществлении квадратурной модуляции выполняется еще одна операция, которую мы пока пропустили. Речь о ней пойдет далее, в разд. "Формирование спектра" этой главы.

При квадратурной модуляции могут меняться и амплитуда, и начальная фаза несущего колебания, поэтому амплитудная и фазовая манипуляция являются частными случаями квадратурной — нужно лишь использовать соответствующие созвездия. Выведем графики этих созвездий с помощью функции `scatterplot` (она будет рассмотрена далее, в разд. "Построение диаграммы рассеяния" этой главы). Результат показан на рис. 8.38:

```
>> scatterplot(pammod(0:7, 8), 1, 0, '*')
>> scatterplot(pskmod(0:7, 8), 1, 0, '*')
```



**Рис. 8.38.** Созвездия, соответствующие 8-позиционной амплитудной (а) и фазовой (б) манипуляции

### ЗАМЕЧАНИЕ

Функции цифровой модуляции пакета Communications при реализации амплитудной манипуляции используют как положительные, так и *отрицательные* амплитудные множители, допуская, таким образом, скачки фазы несущей на  $180^\circ$ . Это видно из графика созвездия амплитудной манипуляции, приведенного на рис. 8.38, а.

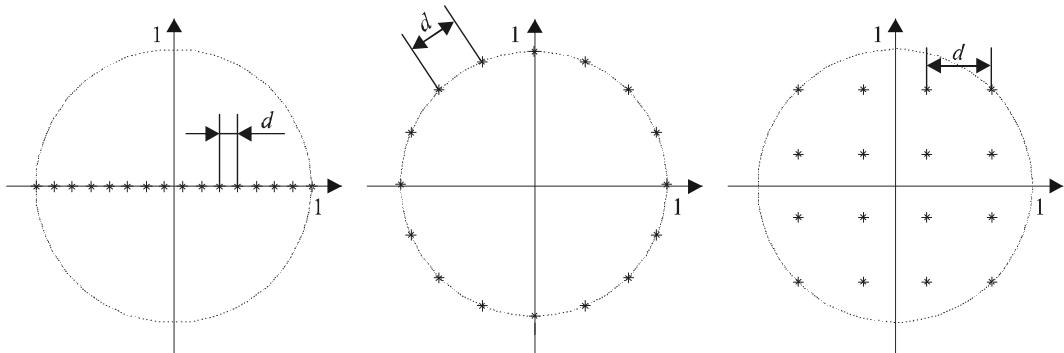
## Помехоустойчивость

За счет использования двумерного характера гармонического несущего колебания (под двумерностью здесь понимается наличие двух параметров, которые можно независимо изменять) квадратурная модуляция обеспечивает большую помехоустойчивость (т. е. меньшую вероятность ошибки), чем АМн и ФМн. Не вдаваясь

в подробности, скажем, что помехоустойчивость тем выше, чем больше расстояние  $d$  между ближайшими точками созвездия на комплексной плоскости. При этом для корректности сравнения разных созвездий у них должны быть одинаковыми, помимо числа точек, среднеквадратические амплитуды:

$$\sigma = \sqrt{\frac{1}{N} \sum_{k=0}^{N-1} |\dot{C}_k|^2}.$$

Сравним для примера помехоустойчивость 16-позиционной амплитудной, фазовой и квадратурной модуляции (рис. 8.39).



**Рис. 8.39.** Сравнение помехоустойчивости 16-позиционной амплитудной (слева), фазовой (в центре) и квадратурной (справа) модуляции

Результаты расчетов минимального расстояния между точками, среднеквадратического уровня сигналов, а также межточечного расстояния, нормированного к этому уровню, сведены в табл. 8.2.

**Таблица 8.2.** Сравнение помехоустойчивости разных видов манипуляции

Вид манипуляции	Минимальное межточечное расстояние, $d_{\min}$	Среднеквадратический уровень сигнала, $\sigma$	Нормированное расстояние, $d_{\min}/\sigma$
АМн	$\frac{2}{15} = 0,1333$	$\frac{\sqrt{85}}{15} \approx 0,6146$	$\frac{2}{\sqrt{85}} \approx 0,2169$
ФМн	$2\sin\left(\frac{\pi}{16}\right) \approx 0,39$	1	$2\sin\left(\frac{\pi}{16}\right) \approx 0,39$
Квадратурная	$\frac{\sqrt{2}}{3} \approx 0,47$	$\frac{\sqrt{5}}{3} \approx 0,745$	$\frac{\sqrt{2}}{\sqrt{5}} \approx 0,63$

Анализ данных из таблицы показывает, что квадратурная модуляция обеспечивает максимальное межточечное расстояние, нормированное к среднеквадратическому уровню сигнала.

## Демодуляция

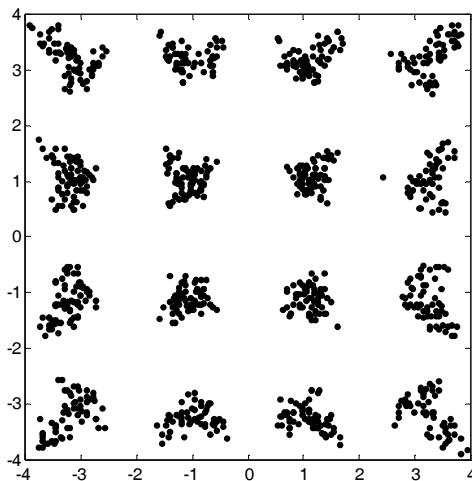
Демодулируется сигнал с квадратурной модуляцией так же, как и в случае аналого-вой квадратурной модуляции — сигнал умножается на два несущих колебания, сдвинутых по фазе друг относительно друга на  $90^\circ$ , а результаты умножения пропускаются через ФНЧ (см. ранее рис. 8.30 в разд. "Демодуляция сигнала с квадратурной модуляцией" этой главы). На выходе этих ФНЧ будут получены аналоговые сигналы синфазной и квадратурной составляющих. Далее эти сигналы дискретизируются с частотой, равной символьной скорости. Пары отсчетов синфазной и квадратурной составляющих образуют комплексное число, и ближайшая к этому числу точка используемого созвездия (а точнее — соответствующий этой точке информационный символ) выдается в качестве выходного результата.

Реализуем описанный алгоритм демодуляции для сформированного ранее сигнала `s_qam16`. Приведенный далее код реализует квадратурную демодуляцию, дискретизацию полученного сигнала с символьной частотой (для этого из вектора `y` выбираются элементы с шагом  $F_s/F_d$ ), вывод графика расположения принятых точек на комплексной плоскости (такой график называется *диаграммой рассеяния* — *scatter plot*, рис. 8.40), выбор ближайших координат точек из использованного созвездия и сравнение полученных синфазных `a2` и квадратурных `b2` амплитуд с исходными амплитудами `aa` и `bb`:

```
>> % умножение на комплексное опорное колебание
>> y = s_qam16 .* exp(2i*pi*Fc*t) * 2;
>> [b, a] = butter(2, Fd*2/Fs); % слаживающий ФНЧ
>> y = filtfilt(b, a, y); % фильтрация
>> z = y(3:FsFd:end); % дискретизация с символьной частотой
>> plot(z, '.')
>> axis square % вывод диаграммы рассеяния
>> a2 = round((real(z)+3)/2); % оценка синфазной амплитуды
>> a2(find(a2<0)) = 0;
>> a2(find(a2>3)) = 3;
>> b2 = round((imag(z)+3)/2); % оценка квадратурной амплитуды
>> b2(find(b2<0)) = 0;
>> b2(find(b2>3)) = 3;
>> symerr(aa', a2) % число ошибок по синфазной амплитуде
ans =
0
>> symerr(bb', b2) % число ошибок по квадратурной амплитуде
ans =
0
```

Функция округления `round` в этом коде применена для поиска точки используемого созвездия, наиболее близкой к принятому отсчету сигнала. Такое решение возможно благодаря простой структуре использованного нами созвездия. В более общем случае для этого необходимо воспользоваться объектом демодулятора `modem.genqamdemod` (см. далее разд. "Функции модуляции и демодуляции пакета *Communications*" этой главы). Функция `symerr` подсчитывает число неправильно

принятых символов (она возвращает число различающихся элементов двух векторов одинаковой длины). Как видите, сигнал принят без ошибок.



**Рис. 8.40.** Диаграмма рассеяния при приеме сигнала с квадратурной модуляцией

#### ЗАМЕЧАНИЕ

На рис. 8.40 видно, что пятна вокруг точек созвездия имеют довольно большой размер, хотя обрабатывался незашумленный сигнал. Это объясняется тем, что приемный ФНЧ был выбран произвольно, без какого-либо согласования его свойств со свойствами обрабатываемого сигнала. Результат, получаемый при таком согласовании, приведен в следующем разделе (см. далее рис. 8.43, б).

## Формирование спектра

Если параметры модуляции аналогового сигнала поддерживаются постоянными в течение символического такта и в начале нового такта изменяются скачкообразно, это приводит к появлению скачков и в сформированном сигнале. Как было показано при обсуждении свойств спектров сигналов (см. разд. "Примеры разложения сигналов в ряд Фурье" главы I), спектр сигнала, содержащего скачки, затухает с ростом частоты медленно — пропорционально  $1/\omega$ . Чтобы сделать спектр более компактным, необходимо обеспечить гладкость сигнала, а это, в свою очередь, означает гладкость модулирующей функции. Следовательно, вместо скачкообразного изменения параметров модуляции нам необходимо выполнить *интерполяцию* между точками созвездия, соответствующими последовательным символам.

Согласно теореме Котельникова, мы можем соединить отсчеты, следующие с символической скоростью  $F_d$ , плавной функцией, занимающей полосу частот от нуля до  $F_d/2$ . В этом случае квадратурно-модулированный сигнал будет занимать полосу частот шириной  $F_d$ . Однако медленное затухание функций sinc, составляющих базис Котельникова, делает неудобной интерполяцию на их основе. Наибольшее рас-

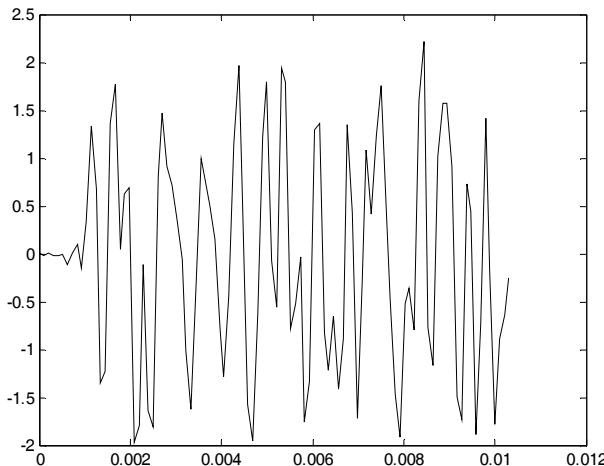
пространение при интерполяции отсчетов для цифровой модуляции получил SQRT-вариант фильтра с косинусоидальным сглаживанием АЧХ (*square root raised-cosine filter*; расчет таких фильтров обсуждался в главе 6).

Фильтр, используемый для интерполяции, определяет форму спектра КАМ-сигнала, поэтому его называют *формирующим фильтром (shaping filter)*, а сам процесс интерполяции — *формированием спектра (spectral shaping)*.

Скачкообразное изменение параметров модуляции можно рассматривать как использование формирующего фильтра с прямоугольной импульсной характеристической, длительность которой равна символьному интервалу.

Повторим формирование 16-позиционного КАМ-сигнала (см. рис. 8.37), используя на этот раз формирующий фильтр с косинусоидальным сглаживанием АЧХ (рис. 8.41):

```
>> % используем сформированные ранее векторы амплитуд aa и bb
>> a1s = 2*aa-3; % преобразуем к требуемому набору
>> b1s = 2*bb-3;
>> a1s = rcosflt(a1s, Fd, Fs, 'sqrt');
>> b1s = rcosflt(b1s, Fd, Fs, 'sqrt');
>> % формируем сигнал с квадратурной модуляцией
>> t = (0:length(a1s)-1)/Fs; % дискретное время
>> t = t'; % превращаем строку в столбец
>> s_qam16s = a1s .* cos(2*pi*Fc*t) + b1s .* sin(2*pi*Fc*t);
>> plot(t(1:100), s_qam16s(1:100))
```



**Рис. 8.41.** Сигнал с 16-позиционной квадратурной модуляцией при использовании формирования спектра

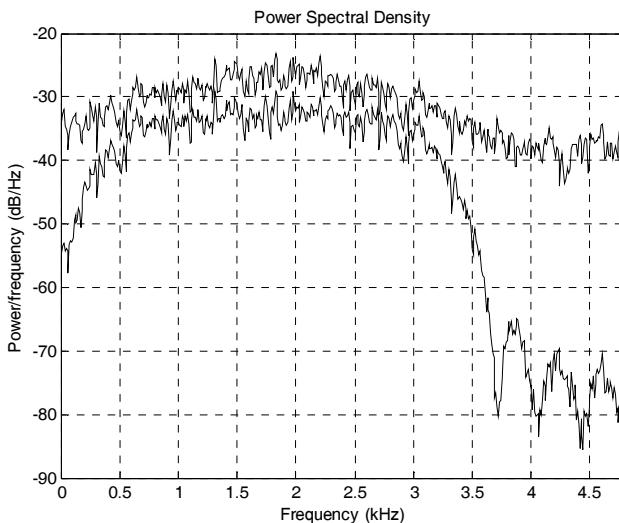
Прослушаем и этот сигнал, снова используя функцию soundsc:

```
>> soundsc(repmat(s_qam16s, 10, 1), Fs)
```

Звук стал больше похож на тот, что производится модемом — все дело именно в формировании спектра.

Сравним спектры мощности сигналов  $s_{\text{qam16}}$  и  $s_{\text{qam16s}}$ , чтобы наглядно показать влияние формирующего фильтра (рис. 8.42):

```
>> [P1, f] = pwelch(s_qam16, [], [], [], Fs);
>> P2 = pwelch(s_qam16s, [], [], [], [], Fs);
>> Hpsd1 = dspdata.psd(P1, f, 'Fs', Fs);
>> Hpsd2 = dspdata.psd(P2, f, 'Fs', Fs);
>> plot(Hpsd1)
>> hold on
>> plot(Hpsd2)
>> hold off
```



**Рис. 8.42.** Спектры плотности мощности сигналов с квадратурной модуляцией при отсутствии (верхний график) и наличии (нижний график) формирующего фильтра

Из графиков видно, что при использовании формирующего фильтра спектр сигнала оказывается значительно компактнее.

При приеме такого сигнала в качестве ФНЧ необходимо использовать такой же фильтр, как для формирования спектра. Последовательное использование двух SQRT-фильтров с косинусоидальным сглаживанием дает результирующую импульсную характеристику вида (6.16) (см. рис. 6.9), равную нулю в точках, сдвинутых на целое число символов относительно пика. Это позволяет при правильном выборе моментов взятия отсчетов устраниТЬ помехи от соседних символов (так называемую *межсимвольную интерференцию*, *MSI*; английский термин — *intersymbol interference, ISI*):

```
>> % умножение на комплексное опорное колебание
>> y = s_qam16s .* exp(-2i*pi*Fc*t) * 2;
```

```

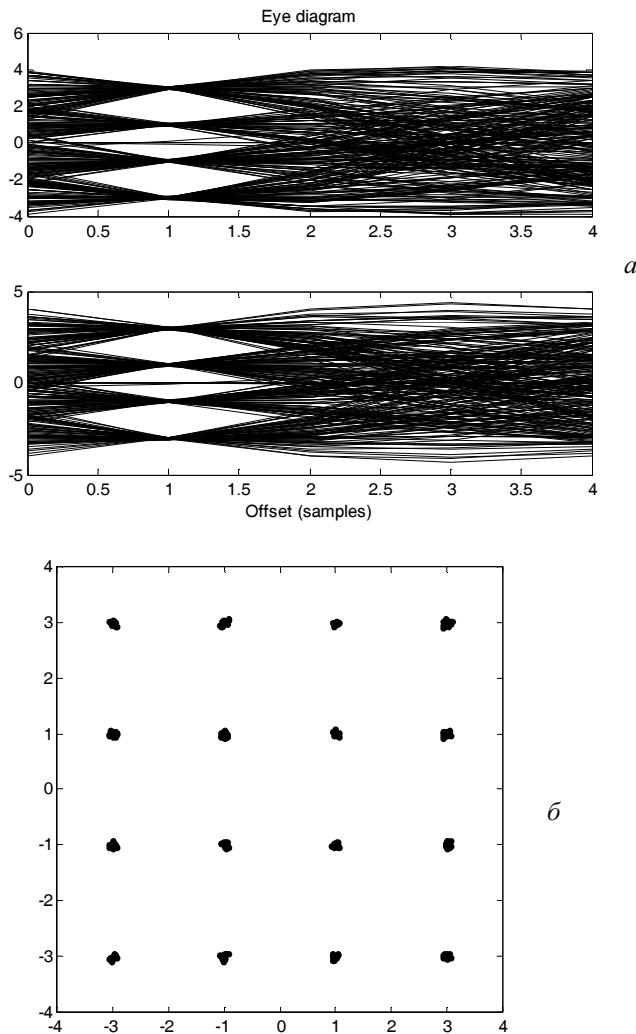
>> b = rcosine(Fd, Fs, 'sqrt'); % сглаживающий ФНЧ
>> y = filter(b, 1, y); % фильтрация
>> % подготовка данных для глазковой диаграммы
>> y_eye = reshape(y, FsFd, length(y)/FsFd);
>> y_eye = [0 y_eye(end,1:end-1); y_eye];
>> % вывод глазковой диаграммы
>> subplot(2, 1, 1)
>> plot(0:FsFd, real(y_eye), 'b')
>> title ('Eye diagram')
>> subplot(2, 1, 2)
>> plot(0:FsFd, imag(y_eye), 'b')
>> xlabel('Offset (samples)')
>> % выбор моментов взятия отсчетов
>> offset = 1;
>> z = y(offset:FsFd:end); % дискретизация с символьной частотой
>> z(1:6) = []; % удаление начального "хвоста"
>> figure
>> plot(z, '.') % вывод диаграммы рассеяния
>> axis square
>> a2 = round((real(z)+3)/2); % оценка синфазной амплитуды
>> a2(find(a2<0)) = 0;
>> a2(find(a2>3)) = 3;
>> b2 = round((imag(z)+3)/2); % оценка квадратурной амплитуды
>> b2(find(b2<0)) = 0;
>> b2(find(b2>3)) = 3;
>> symerr(aa', a2) % число ошибок по синфазной амплитуде
ans =
    0
>> symerr(bb', b2) % число ошибок по квадратурной амплитуде
ans =
    0

```

На рис. 8.43, *a* показана глазковая диаграмма (*eye diagram*) для данного сигнала. Глазковая диаграмма представляет собой "осциллограмму" сигнала, построенную при длительности "прямого хода развертки", равной одному символьному такту, и бесконечном "времени послесвещения экрана". В точках оптимальной дискретизации линии на такой диаграмме образуют узкие пучки, свободное пространство между которыми по форме напоминает раскрытий глаз. В данном случае видно, что выбирать элементы из вектора *y* нужно начиная с первого. Поскольку сигнал является комплексным, приведены отдельные графики для его вещественной и мнимой частей. На рис. 8.43, *b* приведена диаграмма рассеяния, полученная при приеме данного сигнала. Благодаря использованию согласованных друг с другом фильтров на передающей и приемной сторонах разброс точек оказывается значительно меньше, чем на приведенном ранее рис. 8.40.

При прохождении сигнала через канал связи, обладающий *частотной дисперсией*, т. е. вносящий разную групповую задержку на разных частотах, символы оказыва-

ются "размазанными" во времени и "наползают" друг на друга. В этом случае устранить межсимвольную интерференцию полностью не удается. Чтобы минимизировать ее, используют *адаптивные фильтры*, параметры которых автоматически подстраиваются под характеристики обрабатываемого сигнала. Такие фильтры рассмотрены в главе 9.



**Рис. 8.43.** Глазковая диаграмма (a) и диаграмма рассеяния (б), полученные при приеме сигнала с квадратурной модуляцией

## Широтно-импульсная модуляция

При широтно-импульсной модуляции (ШИМ; английский термин — *pulse width modulation*, PWM) в качестве несущего колебания используется периодическая по-

следовательность прямоугольных импульсов, а информационным параметром, связанным с дискретным модулирующим сигналом, является длительность этих импульсов (рис. 8.44).

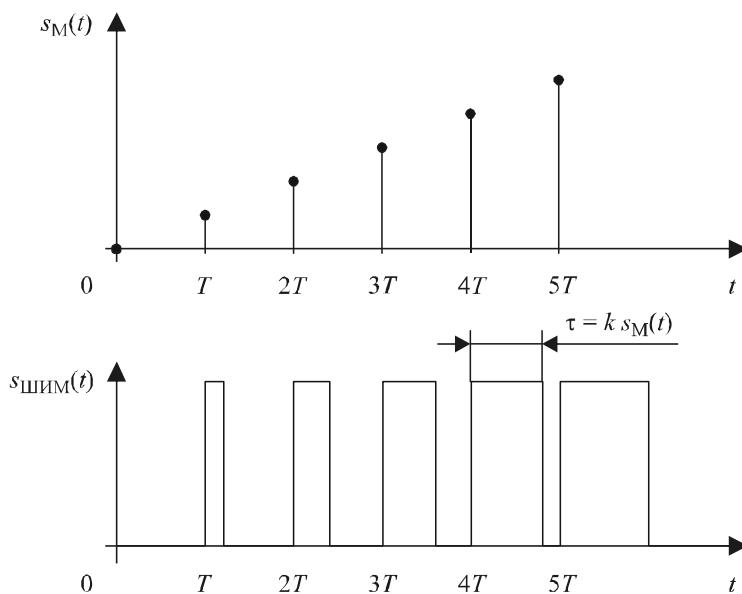


Рис. 8.44. Широтно-импульсная модуляция

Периодическая последовательность прямоугольных импульсов одинаковой длительности имеет постоянную составляющую, обратно пропорциональную скважности импульсов, т. е. прямо пропорциональную их длительности (см. разд. "Примеры разложения сигналов в ряд Фурье" главы 1). Пропустив импульсы через ФНЧ с частотой среза, значительно меньшей, чем частота следования импульсов, эту постоянную составляющую можно легко выделить, получив постоянное напряжение. Если длительность импульсов будет различной, ФНЧ выделит медленно меняющееся напряжение, отслеживающее закон изменения длительности импульсов. Таким образом с помощью ШИМ можно создать несложный ЦАП: значения отсчетов сигнала кодируются длительностью импульсов, а ФНЧ преобразует импульсную последовательность в плавно меняющийся сигнал.

### **ЗАМЕЧАНИЕ**

Именно такие ЦАП используются, например, для реализации настройки на каналы, регулировок громкости, яркости и т. п. в большинстве современных телевизоров: команды от кнопок управления меняют скважность импульсов, вырабатываемых управляющим процессором, а слаживающие цепи формируют из этих импульсов постоянные напряжения, управляющие аналоговыми цепями.

Число уровней напряжения, формируемых таким ЦАП, определяется числом возможных длительностей импульсов. Если импульсы формируются в цифровом виде, шаг изменения длительности определяется частотой дискретизации импульсной

последовательности. Отношение этой частоты к частоте дискретизации формируемого сигнала даст возможное число уровней.

Например, если мы захотим поэкспериментировать с ШИМ, используя звуковую карту компьютера, нам окажется доступно всего шесть уровней (максимальная частота дискретизации, обеспечиваемая звуковой картой, — 48 кГц; минимально приемлемая частота дискретизации звукового сигнала — 8 кГц; получаем  $48/8 = 6$  уровней). Для реальной работы, конечно, маловато, а для экспериментов подойдет.

Попробуем сформировать таким путем синусоиду:

```
>> N = 6; % число уровней
>> Fs = 8e3; % частота дискретизации сигнала
>> Fs0 = Fs * N; % частота дискретизации импульсов
>> f0 = 1e3; % частота синусоиды
>> t = 0:1/Fs:1; % одна секунда дискретного времени
>> t0 = (0:length(t)*N-1)/Fs0; % дискретное время импульсов
>> s = sin(2*pi*f0*t); % отсчеты дискретной синусоиды
>> sq = round((s+1)*N/2); % квантованные отсчеты
>> s_pwm = zeros(N, length(t)); % заготовка для ШИМ-сигнала
>> for k=1:N, s_pwm(k, sq>=k) = 1; end
>> s_pwm = s_pwm(:); % "растягиваем" матрицу в один столбец
```

В приведенном коде следует обратить внимание на цикл `for`, с помощью которого формируется ШИМ-сигнал. Мы сначала создаем шестистрочную матрицу, заполненную нулями, а затем построчно формируем ее содержимое, в цикле сравнивая отсчеты сигнала с постепенно увеличивающимся порогом и записывая единицы в те столбцы, где отсчеты сигнала превышают порог или равны ему.

Посмотрим, как выглядят исходные отсчеты синусоиды и соответствующий ШИМ-сигнал (рис. 8.45):

```
>> subplot(2, 1, 1)
>> stem(t(1:33), sq(1:33))
>> subplot(2, 1, 2)
>> stairs(t0(1:193), s_pwm(1:193))
>> ylim([-0.1 1.1])
```

Теперь взглянем на спектр ШИМ-сигнала (рис. 8.46):

```
>> periodogram(s_pwm, [], [], Fs0);
>> ylim([-50 0])
```

Как видите, наиболее интенсивная спектральная составляющая (не считая составляющей с нулевой частотой) действительно имеет частоту исходной синусоиды — 1 кГц. Выделим ее, пропустив ШИМ-сигнал через ФНЧ с частотой среза, равной частоте Найквиста, т. е. 4 кГц (рис. 8.47):

```
>> [b, a] = butter(5, Fs/Fs0);
>> s_filtered = filter(b, a, s_pwm);
>> plot(t0(1:193), s_filtered(1:193))
>> figure
>> periodogram(s_filtered, [], [], Fs0)
>> ylim([-50 0])
```

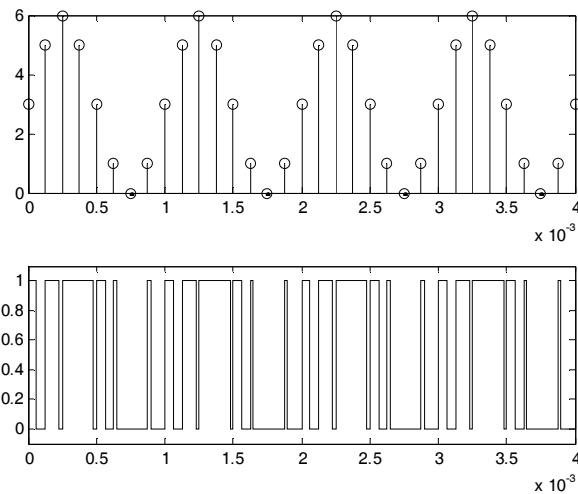


Рис. 8.45. Отсчеты синусоиды (сверху) и соответствующий им ШИМ-сигнал (снизу)

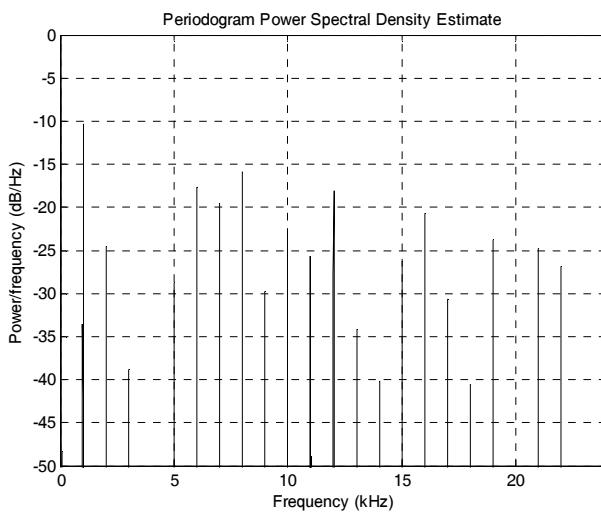
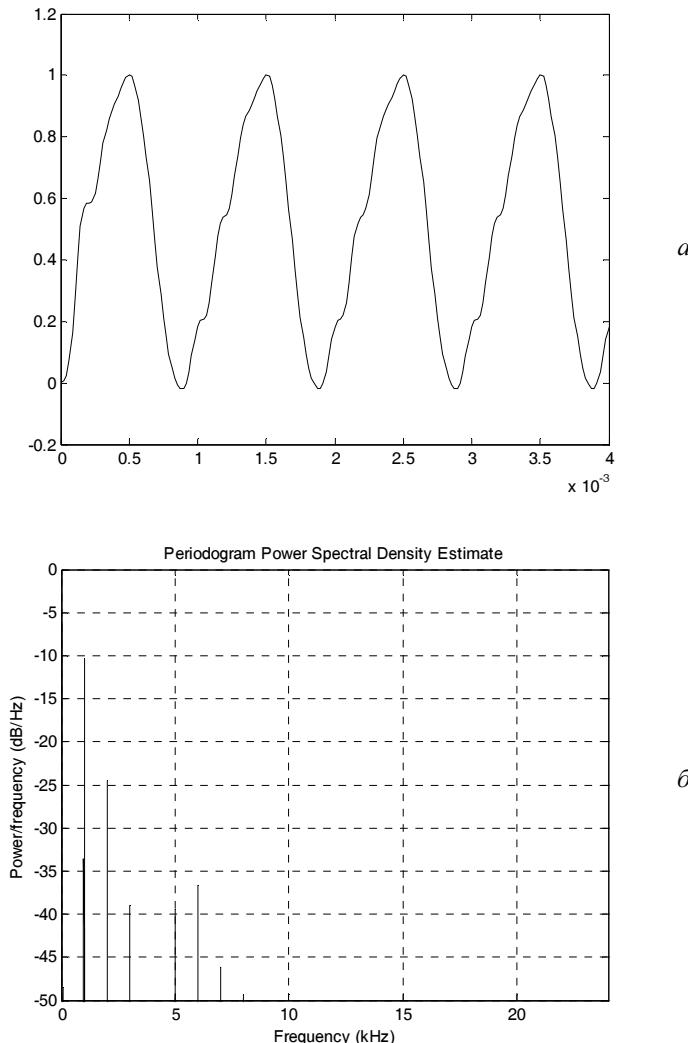


Рис. 8.46. Спектр мощности синусоиды, представленной с помощью ШИМ

Результат, как видите, действительно напоминает синусоиду, хотя оставшиеся искажения хорошо заметны на глаз — шести уровней квантования все-таки явно мало. Желающие могут послушать, как звучат ШИМ-сигнал и выделенная из него низкочастотная составляющая:

```
>> wavplay(s_pwm, Fs0)      % ШИМ-сигнал
>> wavplay(s_filtered, Fs0) % восстановленная синусоида
```

Тональный сигнал хорошо прослушивается в обоих случаях, но в фильтрованном сигнале он, естественно, звучит значительно чище.



**Рис. 8.47.** Синусоида, полученная путем низкочастотной фильтрации ШИМ-сигнала (а) и ее спектр мощности (б)

## Функции модуляции и демодуляции пакета Signal Processing

В пакете Signal Processing модуляция и демодуляция реализуются с помощью всего двух функций — `modulate` и `demod` соответственно. Используемый вид модуляции указывается в качестве одного из входных параметров этих функций.

В общем виде синтаксис вызова функций `modulate` и `demod` выглядит так:

```
[y, t] = modulate(x, Fc, Fs, 'method', opt)
x = demod(y, Fc, Fs, 'method', opt)
```

Здесь  $x$  — вектор отсчетов модулирующего сигнала,  $y$  — вектор отсчетов модулированного сигнала,  $F_c$  — несущая частота,  $F_s$  — частота дискретизации, 'method' — вид модуляции. Назначение дополнительного параметра  $opt$  зависит от вида модуляции и для конкретных случаев будет рассмотрено далее.

Второй выходной параметр  $t$ , возвращаемый функцией `modulate`, — это вектор моментов времени, использованный для расчетов. Он определяется следующим образом:

```
t = ((0:length(y)-1)/Fs)';
```

Обе функции могут обрабатывать многоканальные сигналы, в этом случае  $x$  и  $y$  являются матрицами, обработка которых производится, как обычно, по столбцам.

Параметры  $opt$  и 'method' при вызове функций можно опускать. В качестве вида модуляции в случае отсутствия параметра 'method' подразумевается АМ с подавленной несущей. Используемое по умолчанию значение параметра  $opt$  зависит от вида модуляции.

Для получения хороших результатов следует выбирать достаточно высокое отношение  $F_s/F_c$ . Для аналоговых видов модуляции (все разновидности АМ и УМ, а также квадратурная модуляция) достаточным является  $F_s/F_c = 4$ . Для ШИМ и ВИМ это отношение определяет число представимых уровней сигнала и потому должно быть существенно больше.

### **ВНИМАНИЕ!**

Следует отметить, что функция демодуляции `demod`, входящая в пакет Signal Processing, носит демонстрационный характер и будет хорошо работать только с "чистым" сигналом (желательно — сформированным именно с помощью функции `modulate`). Причиной является упрощенный характер реализованных алгоритмов и невозможность управления их параметрами (например, начальной фазой опорного колебания при демодуляции различных видов АМ). В этом смысле более приспособленными к реальному миру являются функции демодуляции из пакета Communications.

А теперь перейдем к рассмотрению использования функций `modulate` и `demod` для реализации конкретных видов модуляции.

## **Амплитудная модуляция**

Для реализации амплитудной модуляции параметр 'method' должен иметь значение 'amdsb-tc'. Расшифровка аббревиатуры следующая:

- am** — амплитудная модуляция (amplitude modulation);
- dsb** — две боковые полосы (double side band);
- tc** — передача несущей (transmitted carrier).

Параметр  $opt$  задает уровень немодулированной несущей, поэтому дадим ему более осмысленное обозначение  $A_0$ :

```
[y, t] = modulate(x, Fc, Fs, 'amdsb-tc', A0)
x = demod(y, Fc, Fs, 'amdsb-tc', A0)
```

Модуляция реализуется почти по формуле (8.1):

$$y = (x - A_0) \cdot \cos(2\pi F_c t)$$

Значение по умолчанию для параметра  $A_0$  равно  $\min(\min(x))$ . Это обеспечивает однополярность амплитудного множителя  $(x-A_0)$ . Двукратное применение функции  $\min$  обеспечивает правильный выбор постоянного смещения в случае многоканального сигнала.

### **ВНИМАНИЕ!**

Обратите внимание на то, что величина  $A_0$  не прибавляется к модулирующему сигналу  $x$ , а вычитается из него. Поэтому при задании параметра  $A_0$  вручную его значение должно быть отрицательным, если модулирующий сигнал является двуполярным.

При демодуляции входной сигнал прежде всего умножается на опорное колебание:

$$x = y \cdot \cos(2\pi F_c t);$$

Низкочастотная составляющая выделяется с помощью ФНЧ с частотой среза, равной несущей частоте. Чтобы фильтрация не вносила сдвига по времени, она осуществляется с помощью функции `filtfilt` (см. разд. "Компенсация фазового сдвига" главы 4). В качестве ФНЧ используется фильтр Баттерворта 5-го порядка:

```
[b, a] = butter(5, Fc^2/Fs);
x = filtfilt(b, a, x);
```

Наконец, из демодулированного сигнала вычитается уровень несущей  $A_0$ :

$$x = x - A_0;$$

По умолчанию значение параметра  $A_0$  в функции `demod` равно нулю.

### **ВНИМАНИЕ!**

Поскольку и при модуляции, и при демодуляции параметр  $A_0$  фигурирует со знаком "минус", для правильного удаления постоянного смещения из демодулированного сигнала уровни несущей, задаваемые при модуляции и демодуляции, должны иметь разные знаки и соотноситься друг с другом как, соответственно,  $A_0$  и  $-A_0/2$ . Деление на 2 необходимо из-за того, что после умножения на опорное колебание и низкочастотной фильтрации выделенный полезный сигнал имеет вдвое меньший уровень, чем исходный модулирующий сигнал (см. формулу (8.5)).

Если значение параметра  $A_0$  при модуляции слишком велико (больше, чем  $\min(\min(x))$ ), амплитудный множитель перестает быть однополярным, т. е. возникает *перемодуляция* (см. рис. 8.4). Демодуляция такого сигнала осуществляется правильно.

Векторы (матрицы)  $x$  и  $y$  в случае АМ имеют одинаковые размеры.

## АМ с подавленной несущей

Для реализации АМ с подавленной несущей параметр 'method' должен иметь значение 'amdsb-sc' или просто 'am'. Это значение параметра 'method' принято по умолчанию. Первая часть аббревиатуры вида модуляции расшифровывается так же, как в случае обычной АМ, а sc означает подавленную несущую (suppressed carrier):

```
[y, t] = modulate(x, Fc, Fs, 'amdsb-sc')
x = demod(y, Fc, Fs, 'amdsb-sc')
```

АМ с подавленной несущей получается из обычной АМ, реализация которой была рассмотрена в предыдущем разделе, при  $A_0 = 0$ . Входной параметр opt в случае АМ с подавленной несущей не используется. Демодуляция осуществляется так же, как для обычной АМ (с учетом нулевого значения параметра  $A_0$ ).

Векторы (матрицы) x и y в случае АМ с подавленной несущей имеют одинаковые размеры.

## Однополосная модуляция

Для реализации однополосной модуляции параметр 'method' должен иметь значение 'amssb'. В этой аббревиатуре ssb означает одну боковую полосу (single side band). Параметр opt для данного вида модуляции не используется:

```
[y, t] = modulate(x, Fc, Fs, 'amssb')
x = demod(y, Fc, Fs, 'amssb')
```

Формирование однополосного сигнала производится по формуле (8.6):

```
y = x .* cos(2*pi*Fc*t) + imag(hilbert(x)) .* sin(2*pi*Fc*t)
```

При этом результирующий сигнал содержит *нижнюю* боковую полосу. Формирование сигнала с *верхней* боковой полосой официально не предусмотрено, но функцию можно легко "обмануть", задав отрицательное значение для несущей частоты Fc. Согласно формуле (8.6), благодаря нечетности функции sin это приведет к смене выделяемой боковой полосы.

Демодуляция однополосного сигнала производится функцией demod точно так же, как и демодуляция других разновидностей АМ — модулированный сигнал умножается на опорное колебание, а затем пропускается через ФНЧ.

Векторы (матрицы) x и y в случае однополосной модуляции имеют одинаковые размеры.

## Фазовая модуляция

Для реализации фазовой модуляции параметр 'method' должен иметь значение 'pm'. Параметр opt задает интенсивность флуктуаций начальной фазы, поэтому дадим ему обозначение beta (буква β использовалась ранее для обозначения индекса угловой модуляции):

```
[y, t] = modulate(x, Fc, Fs, 'pm', beta)
x = demod(y, Fc, Fs, 'pm', beta)
```

Формирование сигнала с фазовой модуляцией производится по формуле (8.7):

```
y = cos(2*pi*Fc*t + beta*x)
```

По умолчанию величина параметра `beta` выбирается так, чтобы максимальное *положительное* отклонение фазы было равно  $\pi$ :

```
beta = pi/max(max(x))
```

Отклонение фазы в отрицательную сторону зависит от минимального значения модулирующего сигнала и может быть любым. Двукратное применение функции `max` обеспечивает правильный выбор интенсивности флюктуаций фазы в случае многоканального сигнала.

Демодуляция ФМ-сигнала производится путем формирования аналитического сигнала с помощью функции `hilbert`, сдвига его спектра в область нулевых частот путем умножения на  $\exp(-j2\pi F_c t)$  и выделения фазового угла. В завершение результат делится на параметр `beta`:

```
x = angle(hilbert(y) .* exp(-2i*pi*Fc*t)) / beta;
```

При демодуляции параметр `beta` по умолчанию равен единице.

Векторы (матрицы) `x` и `y` в случае фазовой модуляции имеют одинаковые размеры.

## Частотная модуляция

Для реализации частотной модуляции параметр `'method'` должен иметь значение `'fm'`. Параметр `opt` задает интенсивность флюктуаций мгновенной частоты, поэтому дадим ему обозначение `dw`:

```
[y, t] = modulate(x, Fc, Fs, 'fm', dw)
x = demod(y, Fc, Fs, 'fm', dw)
```

Начальная фаза при ЧМ пропорциональна интегралу от модулирующего сигнала (см. разд. "Фазовая и частотная модуляция" этой главы и формулу (8.8)). Интегрирование производится с помощью функции `cumsum` (данная функция вычисляет частичные суммы элементов вектора; это соответствует численному интегрированию методом прямоугольников):

```
y = cos(2*pi*Fc*t + dw*cumsum(x))
```

По умолчанию величина параметра `dw` выбирается так, чтобы максимальное *положительное* отклонение мгновенной частоты было равно `Fc`:

```
dw = Fc/Fs * 2*pi/max(max(x))
```

Отклонение мгновенной частоты в отрицательную сторону зависит от минимального значения модулирующего сигнала и может быть любым. Двукратное применение функции `max` обеспечивает правильный выбор интенсивности флюктуаций мгновенной частоты в случае многоканального сигнала.

Демодуляция ЧМ-сигнала производится путем формирования аналитического сигнала с помощью функции `hilbert`, сдвига его спектра в область нулевых частот путем умножения на  $\exp(-j2\pi F_c t)$  и выделения фазового угла. Далее фаза дифференцируется с помощью функции `diff`, а результат делится на параметр `dw`:

```
x = diff(unwrap(angle(hilbert(y) .* exp(-2i*pi*Fc*t)))) / dw;
```

При демодуляции параметр `dw` по умолчанию равен единице.

Векторы (матрицы) `x` и `y` в случае частотной модуляции имеют одинаковые размеры.

## Квадратурная модуляция

Для реализации квадратурной модуляции параметр 'method' должен иметь значение 'qam'. Параметр `x` в данном случае является синфазным модулирующим сигналом, а параметр `opt` — квадратурным. Дадим этим параметрам обозначения `x_i` (синфазная составляющая, in-phase) и `x_q` (квадратурная составляющая, quadrature):

```
[y, t] = modulate(x_i, Fc, Fs, 'qam', x_q)
[x_i, x_q] = demod(y, Fc, Fs, 'qam')
```

Синтаксис вызова функции демодуляции является исключением из общего правила — используются *два* выходных параметра.

Формирование сигнала с квадратурной модуляцией производится по формуле (8.13):

$$y = x_i \cdot \cos(2\pi Fc \cdot t) + x_q \cdot \sin(2\pi Fc \cdot t)$$

Векторы (матрицы) `x_i` и `x_q` должны иметь одинаковые размеры. Тот же размер будет иметь и сформированный сигнал `y`.

Демодуляция сигнала с квадратурной модуляцией производится так же, как в случае АМ, только используются два опорных колебания, сдвинутых по фазе на  $90^\circ$  друг относительно друга:

```
x_i = 2 * y .* cos(2*pi*Fc*t);
x_q = 2 * y .* sin(2*pi*Fc*t);
[b, a] = butter(5, Fc^2/Fs);
x_i = filtfilt(b, a, x_i);
x_q = filtfilt(b, a, x_q);
```

Дополнительный параметр `opt` при демодуляции не используется.

## Широтно-импульсная модуляция

Для реализации широтно-импульсной модуляции параметр 'method' должен иметь значение 'pwm'. При отсутствии дополнительного параметра `opt` формируемые импульсы выравниваются влево, т. е. начало импульса совпадает с началом временного такта.

Под несущей частотой `Fc` в данном случае понимается частота следования импульсов, а параметр `Fs` представляет частоту дискретизации сформированного сигнала.

Это отношение определяет возможное число уровней сигнала, представимое с помощью ШИМ (см. ранее разд. "Способы модуляции, используемые при передаче цифровой информации" этой главы):

```
[y, t] = modulate(x, Fc, Fs, 'pwm')
x = demod(y, Fc, Fs, 'pwm')
```

Выходной сигнал  $y$  содержит в  $Fs/Fc$  раз больше отсчетов, чем модулирующий сигнал  $x$ .

При использовании в качестве параметра  $opt$  текстовой строки 'centered' импульсы центрируются относительно начала временных тактов. Для демодуляции ШИМ-сигнала, сформированного в таком режиме, тоже нужно указать данный параметр:

```
[y, t] = modulate(x, Fc, Fs, 'pwm', 'centered')
x = demod(y, Fc, Fs, 'pwm', 'centered')
```

Отсчеты модулирующего сигнала в векторе  $x$  должны лежать в диапазоне 0...1, они задают длину формируемых импульсов волях периода следования импульсов.

## Времяимпульсная модуляция

Времяимпульсная модуляция (ВИМ; английский термин — *pulse position modulation, PPM*) не рассматривалась в теоретическом разделе данной главы. Модулированный сигнал при использовании ВИМ представляет собой неравномерную последовательность импульсов фиксированной длительности. Смещения импульсов относительно моментов начала тактовых интервалов пропорциональны модулирующему сигналу. Таким образом, информационным параметром ВИМ-сигнала является временное положение импульсов в пределах тактовых интервалов.

Для реализации времяимпульсной модуляции параметр 'method' должен иметь значение 'ppm'. Под несущей частотой  $Fc$  в данном случае понимается величина, обратная длительности тактового интервала, а параметр  $Fs$  представляет частоту дискретизации сформированного сигнала. Это отношение определяет возможное число уровней сигнала, представимое с помощью ВИМ.

Параметр  $opt$  в данном случае задает длительность формируемых импульсов, нормированную к длительности тактового интервала. Дадим этому параметру обозначение  $\tau_{au}$ :

```
[y, t] = modulate(x, Fc, Fs, 'ppm', tau)
x = demod(y, Fc, Fs, 'ppm')
```

Выходной сигнал  $y$  содержит в  $Fs/Fc$  раз больше отсчетов, чем модулирующий сигнал  $x$ .

По умолчанию значение параметра  $\tau_{au}$  равно 0,1.

Отсчеты модулирующего сигнала в векторе  $x$  должны лежать в диапазоне 0...1, они задают относительное положение формируемых импульсов в пределах тактовых интервалов.

Для получения хороших результатов должно выполняться соотношение  $\tau\alpha * F_s / F_c \gg 1$ , т. е. длительность формируемых импульсов, измеренная в отсчетах, должна быть существенно больше единицы. В противном случае погрешности из-за округления положения импульсов оказываются слишком велики.

При демодуляции дополнительный параметр  $\tau\alpha$  не используется. Для правильной демодуляции необходимо, чтобы соседние импульсы не перекрывались (перекрытие может возникнуть при модуляции, если за отсчетом модулирующего сигнала, близким к 1, следует отсчет, близкий к 0).

## Функции модуляции и демодуляции пакета Communications

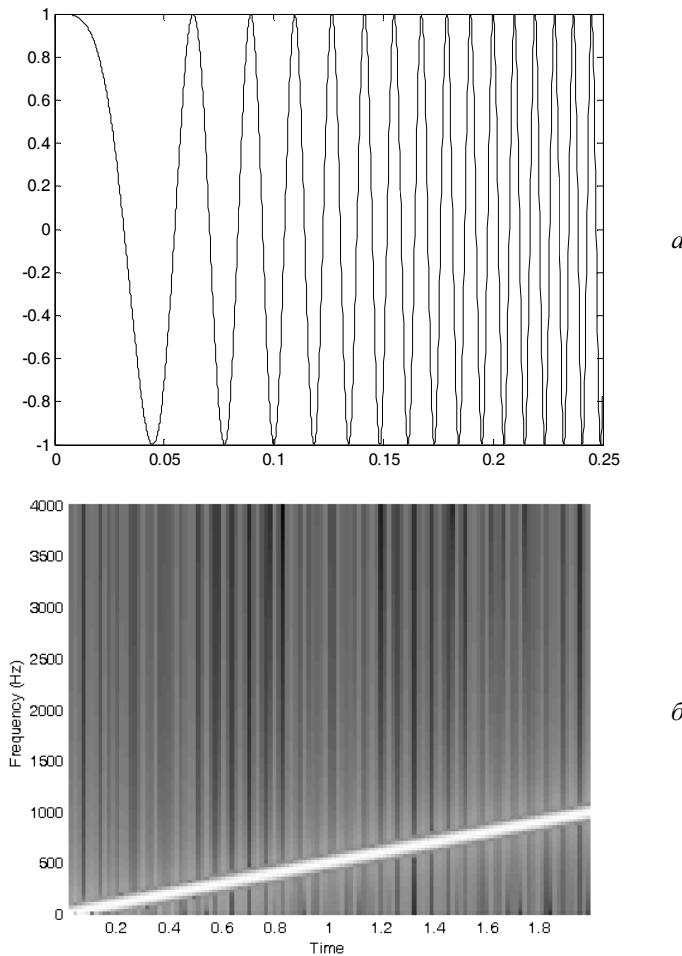
В пакете расширения Communications содержится большое количество функций, относящихся к задачам модуляции и демодуляции — как аналоговой, так и цифровой. В настоящее время для аналоговой модуляции имеются только функции, формирующие и обрабатывающие *вещественный* сигнал на *несущей частоте* (passband modulation), а для цифровой, наоборот, только функции, работающие с *комплексной огибающей* сигнала (baseband modulation; о понятии комплексной огибающей см. соответствующий раздел главы 1).

### Аналоговая модуляция

При описании синтаксиса вызова рассматриваемых далее функций аналоговой модуляции и демодуляции используются следующие общие параметры:

- $x$  — модулирующий сигнал. Если  $x$  — вектор, он может быть как строкой, так и столбцом. Если  $x$  — матрица, каждый ее столбец обрабатывается независимо, что позволяет осуществлять многоканальную обработку сигнала;
- $y$  — модулированный сигнал;
- $z$  — демодулированный сигнал;
- $F_s$  — частота дискретизации модулированного сигнала. При аналоговой модуляции она же является и частотой дискретизации модулирующего сигнала;
- $F_c$  — несущая частота модулированного сигнала;
- $ini\_phase$  — начальная фаза несущего колебания в радианах. По умолчанию значение этого параметра равно нулю;
- $num$  и  $den$  — соответственно числитель и знаменатель передаточной функции фильтра, используемого для фильтрации выходного сигнала в функциях демодуляции. Если не указывать эти параметры, по умолчанию используется фильтр Баттервортса 5-го порядка с частотой среза, равной несущей частоте.

При рассмотрении теоретических основ различных видов модуляции было приведено достаточное количество временных зависимостей, поэтому при рассмотрении функций MATLAB мы будем приводить в основном спектрограммы сигналов, хорошо демонстрирующие разницу между различными способами модуляции.



**Рис. 8.48.** Модулирующий ЛЧМ-сигнал (a) и его спектрограмма (б)

Для рассматриваемых далее примеров будем использовать частоту дискретизации 8 кГц и несущую частоту 2 кГц:

```
>> Fs = 8e3;
>> Fc = 2e3;
```

В качестве модулирующего используем сигнал длительностью 2 с, мгновенная частота которого линейно изменяется от нуля до 1 кГц. Такой сигнал генерируется функцией chirp:

```
>> t = 0:1/Fs:2; % вектор времени
>> s_M = chirp(t, 0, 2, 1e3); % ЛЧМ-сигнал
```

Построим график начального фрагмента этого сигнала (при попытке изобразить весь сигнал точки на графике просто сольются — в сигнале содержится слишком много колебаний) (рис. 8.48, а):

```
>> plot(t(1:2000), s_M(1:2000))
```

Мгновенная частота такого сигнала меняется достаточно медленно, что позволяет хорошо отслеживать эти изменения на спектrogramме (рис. 8.48, б):

```
>> spectrogram(s_M, 256, [], [], Fs, 'yaxis')
>> colormap gray
```

## Амплитудная модуляция

Амплитудная модуляция производится с помощью функции `ammod`. Синтаксис вызова функции имеет следующий вид:

```
y = ammod(x, Fc, Fs, ini_phase, carramp);
```

Не упомянутый в общем списке параметр `carramp` задает амплитуду несущего колебания, т. е. постоянную составляющую, добавляемую к модулирующему сигналу перед умножением его на несущее колебание. По умолчанию значение этого параметра равно нулю, так что формируется АМ-сигнал с подавленной несущей. Для отсутствия перемодуляции значение этого параметра должно быть не меньше чем `-min(min(x))` (это выражение дает минимально возможную одинаковую для всех каналов постоянную составляющую, обеспечивающую однополярность амплитудного множителя).

Реализуем амплитудную модуляцию для сформированного ранее модулирующего ЛЧМ-сигнала. Сразу же построим график начального фрагмента полученного АМ-сигнала и спектrogramму этого сигнала (рис. 8.49). Сначала реализуем принятый по умолчанию вариант с подавленной несущей:

```
>> s_AM = ammod(s_M, Fc, Fs);
>> plot(t(1:2000), s_AM(1:2000))
>> figure
>> spectrogram(s_AM, 256, [], [], Fs, 'yaxis')
>> colormap gray
```

На спектrogramме АМ-сигнала с подавленной несущей хорошо видны две боковые полосы.

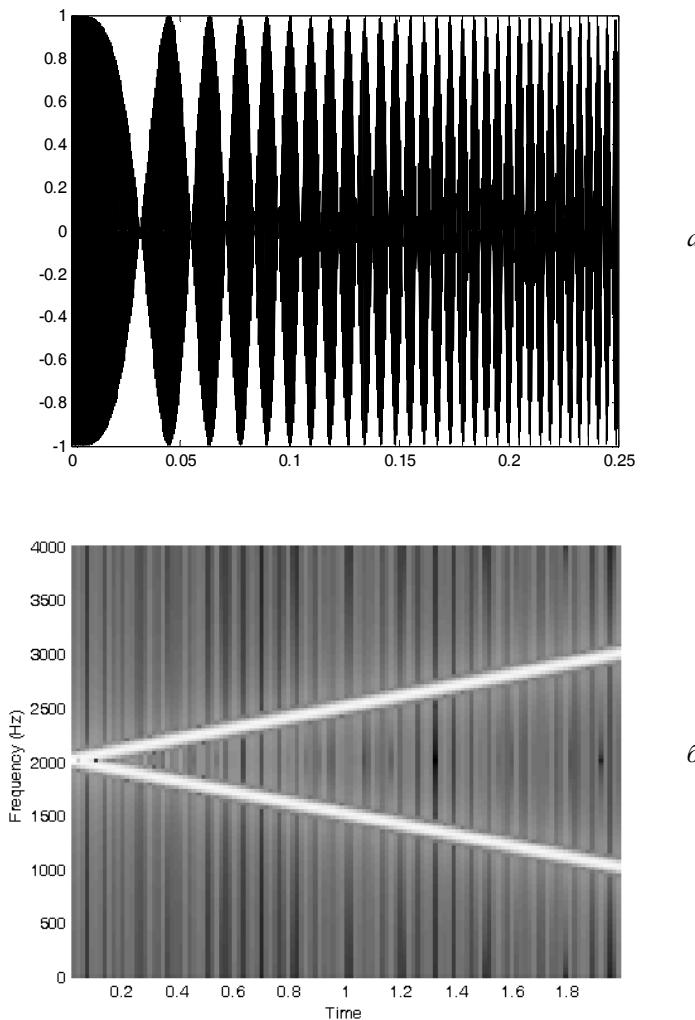
Теперь воспользуемся параметром `carramp`, чтобы реализовать классическую АМ, содержащую несущее колебание:

```
>> s_AM_SC = ammod(s_M, Fc, Fs, [], 1);
>> plot(t(1:2000), s_AM_SC(1:2000))
>> figure
>> spectrogram(s_AM_SC, 256, [], [], Fs, 'yaxis')
>> colormap gray
```

Из рис. 8.49 и 8.50 хорошо видно, что спектrogramма АМ-сигнала с подавленной несущей отличается от спектrogramмы обычного АМ-сигнала только отсутствием несущего колебания.

Демодуляция АМ-сигнала производится с помощью функции `amdemod`:

```
z = amdemod(y, Fc, Fs, ini_phase, carramp, num, den);
```



**Рис. 8.49.** АМ-сигнал с подавленной несущей (а) и его спектрограмма (б)

Входной сигнал у умножается на несущее колебание, при этом начальная фаза этого колебания должна быть такой же, как использованная при модуляции. Результат умножения пропускается через ФНЧ, заданный параметрами `num` и `den`, при этом используется двунаправленная фильтрация с помощью функции `filtfilt`.

Параметр `carramp` — это константа, вычитаемая из полученных значений амплитудной огибающей. По умолчанию он равен нулю.

Данный способ демодуляции позволяет правильно обрабатывать в том числе и сигнал с перемодуляцией.

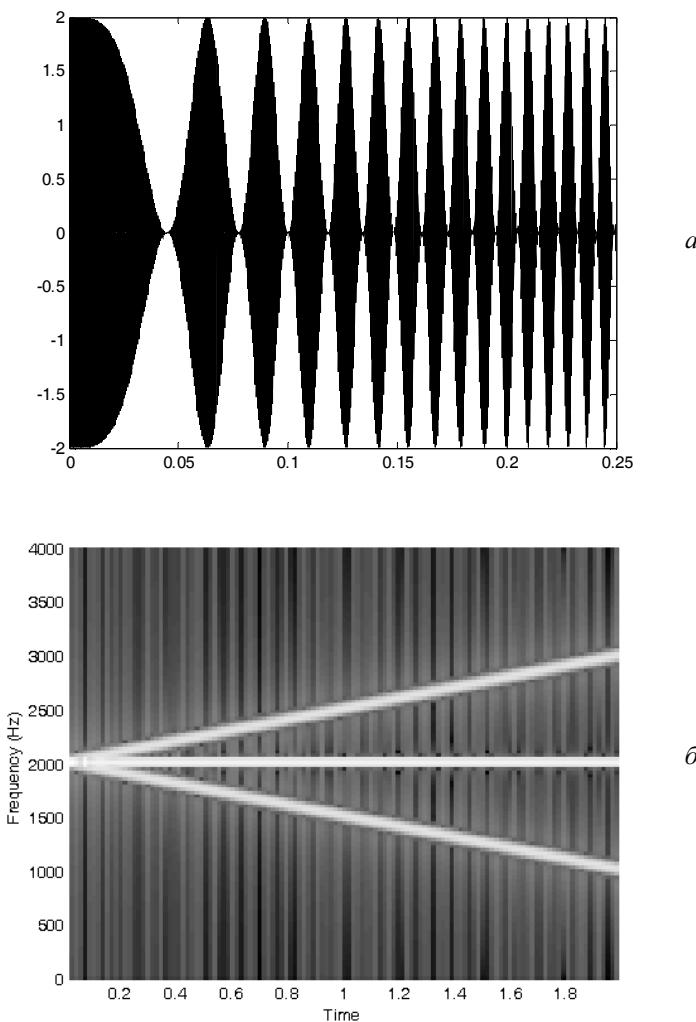


Рис. 8.50. АМ-сигнал (а) и его спектрограмма (б)

## Однополосная модуляция

Однополосная модуляция производится с помощью функции `ssbmod`. Аббревиатура `ssb` означает наличие только одной боковой полосы (single side band). Синтаксис вызова функции имеет следующий вид:

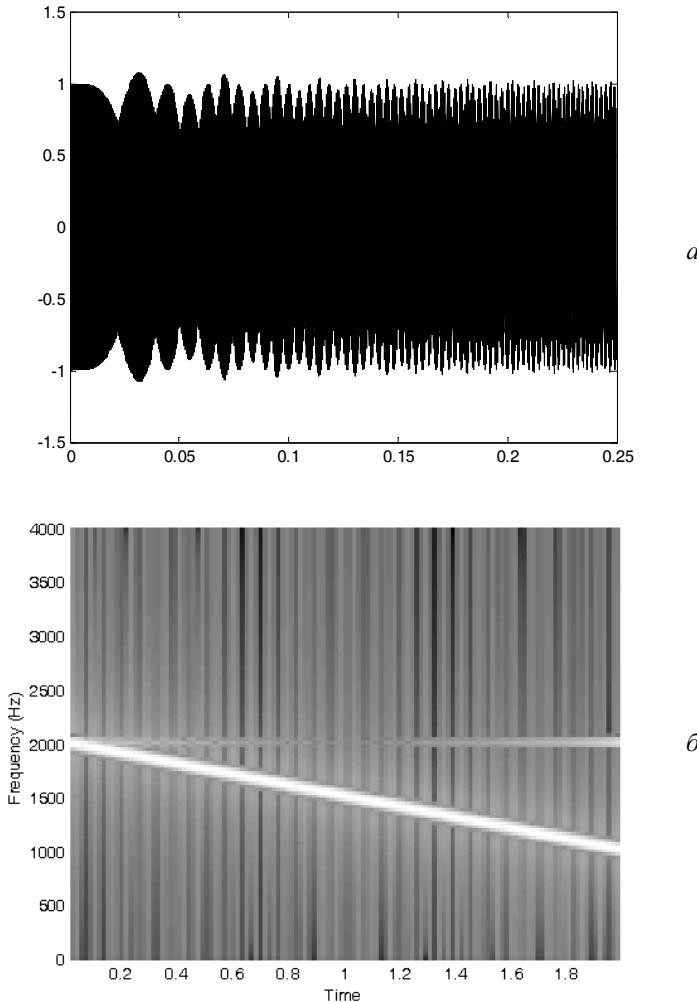
```
y = ssbmod(x, Fc, Fs, ini_phase);
```

Формирование однополосного сигнала производится по формуле (8.6):

```
y = x .* cos(2 * pi * Fc * t + ini_phase) + ...
    imag(hilbert(x)) .* sin(2 * pi * Fc * t + ini_phase);
```

При этом результирующий сигнал содержит *нижнюю* боковую полосу. Для формирования сигнала с *верхней* боковой полосой необходимо добавить дополнительный входной параметр в виде строки 'upper':

```
y = ssbmod(x, Fc, Fs, ini_phase, 'upper');
```



**Рис. 8.51.** Однополосный сигнал с нижней боковой полосой (а) и его спектрограмма (б)

Реализуем однополосную модуляцию для сформированного ранее модулирующего ЛЧМ-сигнала. Сразу же построим график начального фрагмента полученного сигнала и спектрограмму этого сигнала (рис. 8.51):

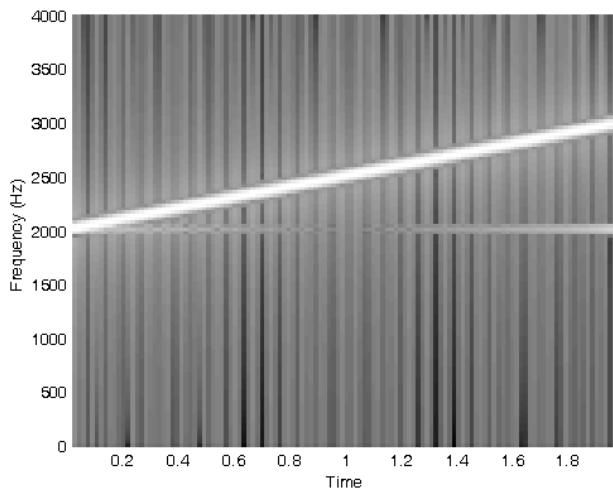
```
>> s_SSB = ssbmod(s_M, Fc, Fs);
>> plot(t(1:2000), s_SSB(1:2000))
>> figure
```

```
>> spectrogram(s_SSB, 256, [], [], Fs, 'yaxis')
>> colormap gray
```

На спектрограмме однополосного сигнала хорошо видно отсутствие верхней боковой полосы; правда, наблюдаются и неподавленные остатки несущего колебания.

Теперь сформируем сигнал с верхней боковой полосой (рис. 8.52):

```
>> s_SSB_UP = ssbmod(s_M, Fc, Fs, [], 'upper');
>> spectrogram(s_SSB_UP, 256, [], [], Fs, 'yaxis')
>> colormap gray
```



**Рис. 8.52.** Спектрограмма однополосного сигнала с верхней боковой полосой

Спектрограмма наглядно показывает, что в данном случае в сигнале действительно сохранена только *верхняя* боковая полоса.

Демодуляция однополосного сигнала производится с помощью функции `ssbdemod`:

```
z = ssbdemod(y, Fc, Fs, ini_phase, num, den)
```

При демодуляции входной сигнал `y` умножается на несущее колебание, а затем подвергается двунаправленной фильтрации (функция `filtfilt`) с помощью ФНЧ, задаваемого параметрами `num` и `den`.

## Фазовая модуляция

Фазовая модуляция производится с помощью функции `pmmod`. Синтаксис вызова функции имеет следующий вид:

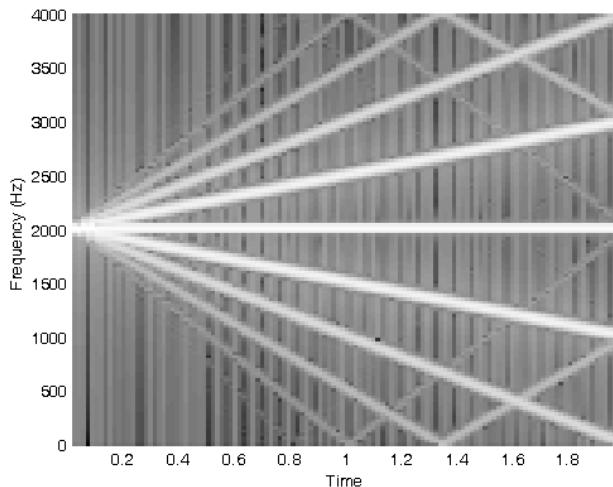
```
y = pmmod(x, Fc, Fs, phasedev, ini_phase);
```

Обязательный входной параметр `phasedev` — константа, на которую умножается модулирующий сигнал перед использованием его в качестве начальной фазы несущего колебания.

щего колебания. Эта константа равна отклонению начальной фазы (в радианах) сформированного сигнала от значения `ini_phase` при уровне модулирующего сигнала `x`, равном единице.

Реализуем фазовую модуляцию для сформированного ранее модулирующего ЛЧМ-сигнала. Сразу же построим спектрограмму этого сигнала (рис. 8.53):

```
>> s_PM = pmmod(s_M, Fc, Fs, 1);
>> spectrogram(s_PM, 256, [], [], Fs, 'yaxis')
>> colormap gray
```



**Рис. 8.53.** Спектрограмма ФМ-сигнала

На спектрограмме ФМ-сигнала хорошо видны несущее колебание и множество боковых частот. Согласно формулам, приведенным ранее в разд. "Спектр сигнала с гармонической угловой модуляцией" этой главы, амплитуды спектральных составляющих пропорциональны функциям Бесселя. "Отражения" спектральных линий от верхнего и нижнего краев спектрограммы связаны с появлением *ложных частот* (aliasing, подробнее см. разд. "Аналоговые, дискретные и цифровые сигналы" и "Спектр дискретного сигнала" главы 3).

Демодуляция ФМ-сигнала производится с помощью функции `pmdemod`:

```
z = pmdemod(y, Fc, Fs, phasedev, ini_phase);
```

Демодуляция ФМ-сигнала производится путем формирования аналитического сигнала с помощью функции `hilbert`, сдвига его спектра в область нулевых частот путем умножения на  $\exp(-j2\pi F_c t - j\phi_0)$  и выделения фазового угла. В завершение результат делится на параметр `phasedev`.

## Частотная модуляция

Частотная модуляция производится с помощью функции `fmod` (аббревиатура `fm` означает частотную модуляцию — frequency modulation). Синтаксис вызова функции имеет следующий вид:

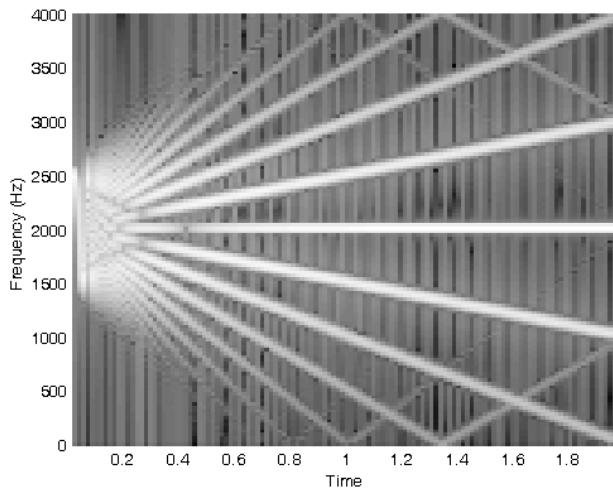
```
y = fmod(x, Fc, Fs, freqdev, ini_phase);
```

Обязательный входной параметр `freqdev` — константа, на которую умножается модулирующий сигнал перед использованием его в качестве отклонения мгновенной частоты от несущей частоты. Эта константа равна отклонению мгновенной частоты (в тех же единицах, в которых заданы параметры `Fc` и `Fs`) сформированного сигнала при уровне модулирующего сигнала `x`, равном единице.

При формировании ЧМ-сигнала модулирующий сигнал приближенно интегрируется с помощью функции `cumsum`, а затем используется в качестве начальной фазы несущего колебания.

Реализуем частотную модуляцию для сформированного ранее модулирующего ЛЧМ-сигнала, задав максимальное отклонение мгновенной частоты равным 500 Гц (поскольку амплитуда модулирующего сигнала равна единице, параметр `freqdev` следует задать равным 500). Сразу же построим спектrogramму полученного сигнала (рис. 8.54):

```
>> s_FM = fmod(s_M, Fc, Fs, 500);
>> spectrogram(s_FM, 256, [], [], Fs, 'yaxis')
>> colormap gray
```



**Рис. 8.54.** Спектrogramма ЧМ-сигнала

На спектrogramме ЧМ-сигнала, как и в случае ФМ (см. рис. 8.53), хорошо видны несущее колебание и множество боковых частот. Однако имеются и отличия: сначала, пока частота модулирующего сигнала низкая, индекс модуляции велик (на-

помним, что при ЧМ индекс угловой модуляции меняется обратно пропорционально частоте модулирующего сигнала) и мы наблюдаем большое количество боковых частот — белое пятно в левой части графика имеет значительно большую высоту, чем в случае ФМ. По мере роста частоты модулирующего сигнала индекс модуляции падает и количество значимых боковых частот уменьшается. В результате в правой части графика спектр визуально не отличается от случая ФМ.

Демодуляция ЧМ-сигнала производится с помощью функции `fmdemod`:

```
z = fmdemod(y, Fc, Fs, freqdev, ini_phase)
```

Демодуляция ЧМ-сигнала производится путем формирования аналитического сигнала с помощью функции `hilbert`, сдвига его спектра в область нулевых частот путем умножения на  $\exp(-j2\pi F_c t - j\phi_0)$  и выделения фазового угла. У фазы устраняются незначащие скачки на  $2\pi$  (функция `unwrap`), после чего она дифференцируется с помощью функции `diff`, а результат делится на  $2\pi$  и на параметр `freqdev`.

## Цифровая модуляция

В настоящее время цифровая модуляция и демодуляция (за исключением частотной) реализованы в пакете Communications на базе объектов класса `modem`. Существовавшие ранее функции модуляции и демодуляции пока что доступны, но объявлены устаревшими и от их использования рекомендовано отказаться.

Общая информация об использовании объектов в MATLAB была приведена в *главе 4* применительно к объектам дискретных фильтров. В данном вводном разделе будут рассмотрены общие свойства и методы объектов модуляторов и демодуляторов, а в последующих разделах приведена информация, специфичная для отдельных видов модуляции.

Для создания объекта модулятора или демодулятора служит функция `modem`. При ее вызове необходимо указать конкретный вид модуляции или демодуляции — для этого после имени функции ставится точка и указывается соответствующий идентификатор метода (конструктора):

```
hmod = modem.modtype(....)
hdemod = modem.demodtype(....)
```

Входными параметрами функции являются пары "`'имя_свойства'`, `значение_свойства'`", при этом абсолютно все свойства имеют значения по умолчанию, поэтому в простейшем случае список входных параметров может полностью отсутствовать:

```
hmod = modem.modtype
hdemod = modem.demodtype
```

Главными методами объектов `modem` являются `modulate` для модуляторов и `demodulate` для демодуляторов:

```
y = hmod.modulate(x);
z = hdemod.demodulate(y);
```

Здесь  $x$  — входное сообщение, элементами которого должны являться либо целые числа в диапазоне, определяемом размером сигнального созвездия, либо отдельные биты, принимающие значения 0 и 1 (в зависимости от значения свойства `InputType`, см. далее). Параметр  $x$  может быть матрицей, тогда ее столбцы обрабатываются независимо.

Результат работы модулятора — вектор или матрица  $y$ , содержащая отсчеты комплексной огибающей сигнала. Никаких процедур, связанных с формированием спектра сигнала, модуляторами не выполняется.

Результат работы демодулятора — вектор или матрица  $z$ , содержащая оценки элементов принятого сообщения (тоже в виде целых чисел или отдельных битов, в зависимости от значения свойства `OutputType`, см. далее).

Возможен и другой вариант синтаксиса вызова функций модуляции и демодуляции, когда объект передается в качестве входного параметра:

```
y = modulate(hmod, x);
z = demodulate(hdemod, y);
```

Ряд свойств объектов модуляторов и демодуляторов являются общими и используются для всех или почти для всех видов модуляции. Эти свойства приведены в табл. 8.3.

**Таблица 8.3.** Общие свойства объектов модуляторов и демодуляторов

Имя свойства	Описание
Type	Строка, указывающая тип модулятора или демодулятора (только для чтения)
M	Число уровней манипуляции (размер сигнального созвездия). Значение этого свойства должно быть степенью двойки. По умолчанию используется значение 2 для фазовой манипуляции, 16 — для квадратурной
Constellation	Вектор точек сигнального созвездия (только для чтения, кроме объектов <code>genqammod</code> и <code>genqamdemod</code> )
SymbolOrder	Способ сопоставления битовых комбинаций точкам созвездия: 'binary' (принято по умолчанию) — точки созвездия нумеруются последовательно; 'gray' — используется код Грея, при котором соседние точки созвездия отличаются друг от друга лишь одним битом; 'user-defined' — последовательность нумерации точек созвездия задается пользователем
SymbolMapping	Вектор соответствия битовых комбинаций точкам созвездия. Это свойство доступно для записи, только если для свойства <code>SymbolOrder</code> задано значение 'user-defined', в остальных случаях этот вектор рассчитывается автоматически
PhaseOffset	Угол поворота (в радианах) сигнального созвездия относительно его стандартного положения. По умолчанию 0
InputType	Тип входных данных для объектов модуляторов: 'integer' (принято по умолчанию) — элементы передаваемого сообщения должны быть целыми числами в диапазоне от 0 до $M - 1$ ; 'bit' — элементы передаваемого сообщения должны быть битами, т. е. принимать только значения 0 и 1. При модуляции биты будут собраны в группы (символы) по $\log_2(M)$ элементов

Таблица 8.3 (окончание)

Имя свойства	Описание
OutputType	Это свойство аналогично свойству InputType, но оно определяет форму представления результата работы демодуляторов, причем только если свойство DecisionType имеет значение 'hard decision'
DecisionType	Вид решений, выдаваемых демодуляторами: 'hard decision' (принято по умолчанию) — выдаются <i>жесткие решения</i> , т. е. предположения демодулятора о том, какие элементы сообщения были переданы; 'llr' — для каждого бита выдается <i>мягкое решение</i> , т. е. <i>логарифм отношения правдоподобия</i> , характеризующий "степень уверенности" демодулятора в значении этого бита. Мягкие решения могут использоваться в тех системах цифровой связи, где применяется помехоустойчивое кодирование (см., например, [35, 36]); 'approximate llr' — приближенный вариант мягких решений
NoiseVariance	Дисперсия шума, содержащегося во входном сигнале демодулятора. Эта величина необходима только при расчете мягких решений, т. е. при значении свойства DecisionType, равном 'llr' или 'approximate llr'

Что касается методов, имеющихся у данных объектов, то, помимо уже упоминавшихся методов модуляции и демодуляции, имеется всего три метода:

- `h1 = copy(h)` — создает *независимую копию* объекта с теми же значениями свойств, что у исходного объекта `h`;
- `disp(h)` — выводит на экран свойства объекта;
- `reset(h, nc)` — производит сброс внутреннего состояния для объектов модуляторов и демодуляторов, обладающих памятью (это относится к модуляторам `dpskmod`, `oqpskmod`, `mskmod` и соответствующим демодуляторам).

### ЗАМЕЧАНИЕ

Два из трех упомянутых ранее видов цифровой модуляции в книге не рассматриваются: фазоразностная манипуляция (DPSK; объекты `dpskmod` и `dpskdemod`) и квадратурная фазовая манипуляция со сдвигом (Offset QPSK; объекты `oqpskmod` и `oqpskdemod`).

Далее рассмотрим особенности объектов, реализующих конкретные виды цифровой модуляции.

## Амплитудная манипуляция

Для реализации амплитудной манипуляции используются объекты `pammod` и `pamdemod` (аббревиатура `pam` означает амплитудно-импульсную модуляцию, *Pulse Amplitude Modulation*):

```
hmod = modem.pammod(....)
hdemod = modem.pamdemod(....)
```

Единственной особенностью объектов этих классов является отсутствие возможности задавать фазовый сдвиг — сигнальное созвездие всегда расположено вдоль ве-

щественной оси, а свойство `PhaseOffset` не поддерживается. Вероятно, это связано с тем, что на практике многоуровневая ( $M > 2$ ) амплитудная манипуляция используется практически исключительно в низкочастотных системах связи без несущего колебания, а единственный распространенный в радиотехнических системах вариант с  $M = 2$  эквивалентен рассматриваемой далее фазовой манипуляции.

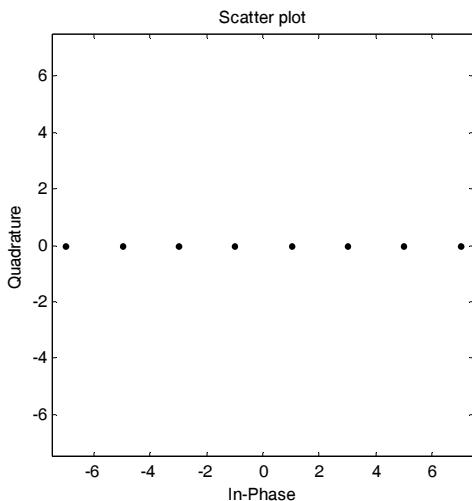
Процесс модуляции сводится к отображению элементов входного сообщения в точки сигнального созвездия, а демодуляции (при использовании жестких решений) — к поиску точки сигнального созвездия, ближайшей к полученному на выходе значению.

Точки сигнального созвездия имеют вещественные целочисленные координаты, рассчитываемые как

$$-(M-1) : 2 : (M-1)$$

В качестве примера на рис. 8.55 показано сигнальное созвездие для 8-позиционной АМН:

```
>> hmod = modem.pammod('M', 8);
>> scatterplot(hmod.Constellation)
```



**Рис. 8.55.** Сигнальное созвездие 8-позиционной амплитудной манипуляции

## Квадратурная модуляция со стандартными созвездиями

Для реализации квадратурной модуляции со стандартными созвездиями, точки которых расположены на равномерной квадратной сетке, используются объекты `qammod` и `qamdemod`:

```
hmod = modem.qammod(....)
hdemod = modem.qamdemod(....)
```

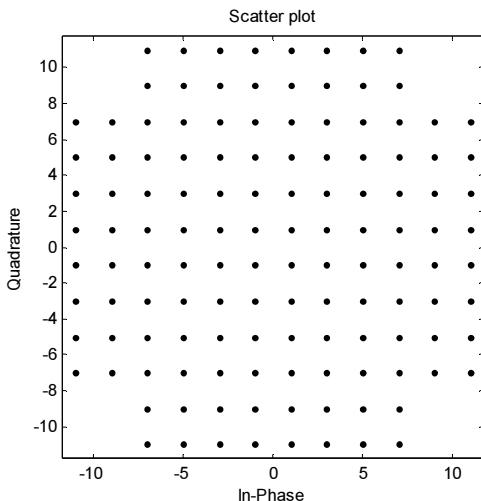
Процесс модуляции сводится к отображению элементов входного сообщения в точки сигнального созвездия, а демодуляции (при использовании жестких реше-

ний) — к поиску точки сигнального созвездия, ближайшей к полученному на входе значению.

Точки сигнального созвездия имеют целочисленные координаты на комплексной плоскости, расстояние между ближайшими точками равно двум.

В качестве примера на рис. 8.56 показано сигнальное созвездие для 128-позиционной КАМ:

```
>> hmod = modem.qammod('M', 128);
>> scatterplot(hmod Constellation)
```



**Рис. 8.56.** Сигнальное созвездие 128-позиционной квадратурной модуляции

## Квадратурная модуляция с произвольным созвездием

Для реализации квадратурной модуляции с произвольным созвездием, задаваемым пользователем, используются объекты `genqammod` и `genqamdemod`:

```
hmod = modem.genqammod(....)
hdemod = modem.genqamdemod(....)
```

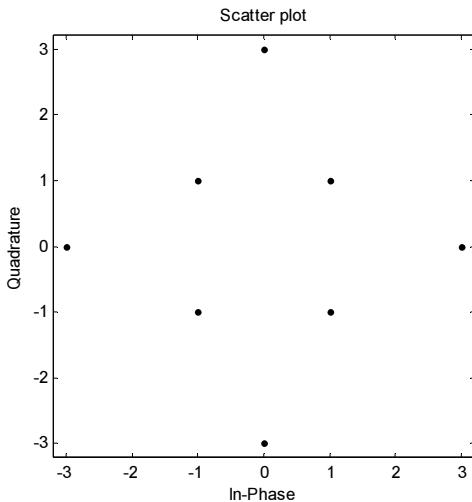
В отличие от прочих видов модуляции, здесь вместо числа точек созвездия (свойство `M`) задается вектор, содержащий координаты точек созвездия на комплексной плоскости (свойство `Constellation`). Свойство `M` при этом обновляется автоматически. Кроме того, из-за произвольности созвездия не имеют смысла свойства `SymbolOrder` и `SymbolMapping`. Не поддерживается и свойство `PhaseOffset` — фазовый сдвиг легко реализуется умножением вектора точек созвездия на соответствующую комплексную экспоненту.

Процесс модуляции сводится к отображению элементов входного сообщения в точки сигнального созвездия, а демодуляции (при использовании жестких реше-

ний) — к поиску точки сигнального созвездия, ближайшей к полученному на входе значению.

В качестве примера создадим объект модулятора 8-позиционной КАМ с созвездием, не относящимся к "стандартным" (рис. 8.57):

```
>> hmod = modem.genqammod;
>> hmod.Constellation = [-1+1i 1+1i -1-1i 1-1i 3i 3 -3 -3i];
>> scatterplot(hmod.Constellation)
```



**Рис. 8.57.** "Нестандартное" созвездие 8-позиционной квадратурной модуляции

## Фазовая манипуляция

Для реализации фазовой манипуляции используются объекты `pskmod` и `pskdemod`:

```
hmod = modem.pskmod(...)  
hdemod = modem.pskdemod(...)
```

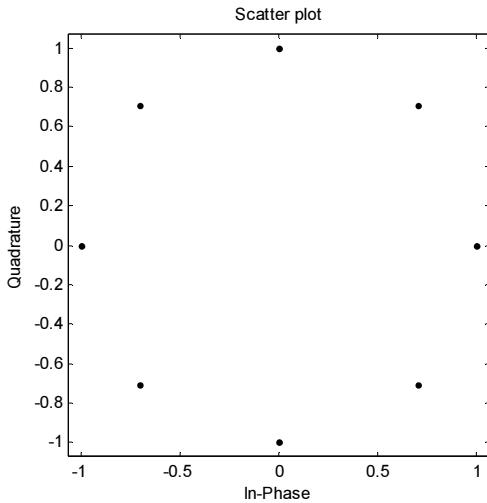
Процесс модуляции сводится к отображению элементов входного сообщения в точки сигнального созвездия, а демодуляции (при использовании жестких решений) — к поиску точки сигнального созвездия, ближайшей к полученному на входе значению.

Точки сигнального созвездия равномерно распределены по единичной окружности и при нулевом фазовом сдвиге могут быть рассчитаны как

$$\exp(2i\pi(0:M-1)/M)$$

В качестве примера на рис. 8.58 показано сигнальное созвездие для 8-позиционной ФМн:

```
>> hmod = modem.pskmod('M', 8);  
>> scatterplot(hmod.Constellation)
```



**Рис. 8.58.** Сигнальное созвездие 8-позиционной фазовой манипуляции

## Минимальная частотная манипуляция

Для реализации минимальной частотной манипуляции (МЧМ) используются объекты `mskmod` и `mskdemod`:

```
hmod = modem.mskmod(....)
hdemod = modem.mskdemod(....)
```

Напомним, что минимальная частотная манипуляция — это ЧМн с непрерывной фазой и разносом частот нуля и единицы, равным половине символьной скорости ( $F_d/2$ ). Как говорилось в разд. "Способы модуляции, используемые при передаче цифровой информации" этой главы, сигналы, соответствующие нулю и единице, при этом оказываются некоррелированными. Входному биту "0" соответствует частота комплексной огибающей, равная  $-F_d/4$ , биту "1" —  $+F_d/4$ .

Объекты модулятора и демодулятора для данного вида модуляции не имеют свойств `Constellation`, `SymbolOrder`, `SymbolMapping`, `PhaseOffset` и `NoiseVariance`, свойство `M` доступно только для чтения и равно 2, а значение свойства демодулятора `DecisionType` автоматически устанавливается равным '`hard decision`'. Кроме того, эти объекты имеют два специфических для них свойства:

- `Precoding` — способ кодирования при минимальной частотной манипуляции. По умолчанию используется значение '`off`', при котором передаваемые биты задают знак отклонения частоты сигнала. При установке значения '`on`' результирующую модуляцию можно трактовать либо как частотную с дифференциальным кодированием данных, либо как четырехпозиционную фазовую модуляцию со сдвигом. Подробнее о взаимосвязи МЧМ и фазовой манипуляции см., например, в [35, 36];
- `SamplesPerSymbol` — число отсчетов комплексной огибающей, формируемых для каждого символа входного сигнала.

## Частотная манипуляция

Как уже отмечалось ранее, частотная манипуляция в настоящее время остается единственным видом цифровой модуляции, не реализованным в виде объектов класса `modem`. Функция частотной манипуляции `fskmod` вызывается следующим образом:

```
y = fskmod(x, M, freq_sep, nsamp, Fs, phase_cont, symbol_order)
```

Здесь  $M$  — число позиций манипуляции;  $x$  — входной сигнал, значения которого должны представлять собой целые числа, лежащие в диапазоне от 0 до  $M - 1$  включительно; `freq_sep` — расстояние между соседними частотами манипуляции; `nsamp` — число отсчетов комплексной огибающей, формируемых для каждого символа входного сигнала; `Fs` — частота дискретизации (по умолчанию 1 Гц); `phase_cont` — строковый параметр, задающий формирование сигнала с непрерывной фазой (значение '`cont`', используется по умолчанию) или с возможными разрывами фазы (значение '`discont`'); `symbol_order` — способ сопоставления частот манипуляции и передаваемых битовых комбинаций: '`bin`' (частоты нумеруются подряд; этот вариант используется по умолчанию) или '`gray`' (при назначении битовых комбинаций частотам используется код Грэя).

Параметры, для которых указаны значения по умолчанию, являются необязательными.

Результат  $y$  представляет собой отсчеты сформированной комплексной огибающей. Ее демодуляция выполняется с помощью функции `fskdemod`:

```
z = fskdemod(y, M, freq_sep, nsamp, Fs, symbol_order)
```

Здесь  $y$  — входные отсчеты комплексной огибающей,  $z$  — демодулированные символы. Смысл остальных параметров был описан ранее применительно к функции модуляции.

Функция `fskdemod` использует для приема ЧМн-сигнала некогерентный корреляционный алгоритм. Возможность осуществления когерентного приема таких сигналов в пакете `Communications` в настоящее время отсутствует.

## Построение диаграммы рассеяния

Для построения диаграммы рассеяния (см. ранее рис. 8.40 и 8.43, а также комментарии к ним) предназначена функция `scatterplot`, имеющая следующий синтаксис вызова:

```
h = scatterplot(x, n, offset, plotstring, h);
```

Обязательным параметром является только входной сигнал  $x$ . Интерпретация массива  $x$  зависит от размера массива и наличия у него мнимой части:

- если  $x$  — вещественная двухстолбцевая матрица, функция `scatterplot` интерпретирует первый столбец как *синфазную*, а второй — как *квадратурную* составляющую;

- если  $x$  — комплексный вектор, функция `scatterplot` интерпретирует его вещественную часть как *синфазную*, а мнимую — как *квадратурную* составляющую;
- если  $x$  — вещественный вектор, функция `scatterplot` интерпретирует его как вещественный сигнал (т. е. в данном случае квадратурная составляющая равна нулю).

Остальные входные параметры являются необязательными и имеют значения по умолчанию. Параметр  $n$  задает число отсчетов на символный такт (применительно к обозначениям, использованным ранее, оно равно  $F_s/F_d$ ). При построении диаграммы используется только каждый  $n$ -й отсчет сигнала, т. е. предварительно выполняется *прореживание* сигнала  $x$  с коэффициентом  $n$ . По умолчанию значение параметра  $n$  равно единице.

Параметр `offset` задает смещение, используемое при прореживании. При построении диаграммы используется каждый  $n$ -й отсчет сигнала  $x$ , начиная с отсчета с номером (`offset+1`). Значение параметра `offset` должно быть целым числом, лежащим в диапазоне от 0 до  $n-1$ . По умолчанию значение параметра `offset` равно нулю.

Строковый параметр `plotstring` задает символы точек, тип линии и цвет для диаграммы. Формат и назначение элементов строки те же самые, что используются в функции `plot` (см. далее разд. "Настройка внешнего вида графиков" приложения 1). По умолчанию диаграмма выводится точками синего цвета.

При указании входного параметра `h` вместо создания нового графического окна диаграмма выводится в существующем окне с дескриптором `h`. Параметр `h` должен быть дескриптором графического окна, ранее созданного функцией `scatterplot`. Имеющаяся в этом окне диаграмма стирается. Для вывода нескольких сигналов в одном окне можно использовать команду `hold on`.

Выходным параметром `h` является дескриптор графического окна, содержащего диаграмму.

В случае вещественной двухстолбцовой матрицы  $x$  диаграмма рассеяния формируется функцией `scatterplot` следующим образом:

```
plot(x(offset+1:n:end,1), x(offset+1:n:end,2), plotstring)
```

Если  $x$  — комплексный вектор, формирование диаграммы рассеяния описывается так:

```
plot(real(x(offset+1:n:end)), imag(x(offset+1:n:end)), plotstring)
```

Наконец, в случае вещественного вектора  $x$  диаграмма строится так:

```
plot(x(offset+1:n:end), 0*x(offset+1:n:end), plotstring)
```

Использование диаграмм рассеяния иллюстрируется демонстрационной программой `scattereyedemo`, имеющейся в составе пакета Communications.

### **ЗАМЕЧАНИЕ**

Помимо функции `scatterplot`, вывод диаграммы рассеяния в настоящее время реализован также с помощью объекта класса `commscope.ScatterPlot`.

## Построение глазковой диаграммы

Для построения глазковой диаграммы (см. ранее рис. 8.43 и комментарии к нему) предназначена функция `eyediagram`, имеющая следующий синтаксис вызова:

```
h = eyediagram(x, n, period, offset, plotstring, h);
```

Обязательными параметрами являются входной сигнал `x` и число отсчетов на символный такт `n` (применительно к обозначениям, использованным ранее, оно равно  $F_s/F_d$ ). Интерпретация массива `x` и число выводимых диаграмм зависят от размера массива и наличия у него мнимой части:

- если `x` — вещественная двухстолбовая матрица, функция `eyediagram` интерпретирует первый столбец как *синфазную*, а второй — как *квадратурную* составляющую. Диаграммы для двух составляющих выводятся в отдельных системах координат в общем графическом окне;
- если `x` — комплексный вектор, функция `eyediagram` интерпретирует его вещественную часть как *синфазную*, а мнимую — как *квадратурную* составляющую. Диаграммы для двух составляющих выводятся в отдельных системах координат в общем графическом окне;
- если `x` — вещественный вектор, функция `eyediagram` интерпретирует его как вещественный сигнал. Графическое окно в данном случае содержит единственную диаграмму.

Остальные входные параметры являются необязательными и имеют значения по умолчанию. Параметр `period` задает длительность символного такта. Это значение используется для оцифровки горизонтальной оси, крайние значения которой считаются равными `-period/2` и `period/2`. По умолчанию значение параметра `period` равно единице.

Параметр `offset` задает горизонтальное смещение графика. Функция предполагает, что значениям времени, кратным длительности символного такта `period`, соответствуют отсчет сигнала с номером `offset+1` и следующие за ним с шагом `n`. Значение параметра `offset` должно быть неотрицательным целым числом, лежащим в диапазоне от 0 до `n-1`. По умолчанию значение параметра `offset` равно нулю.

Строковый параметр `plotstring` задает символы точек, тип линии и цвет для графика. Формат и назначение элементов строки те же самые, что используются в функции `plot` (см. далее разд. "Настройка внешнего вида графиков" приложения 1). По умолчанию, как обычно, графики выводятся сплошными линиями синего цвета.

При указании входного параметра `h` вместо создания нового графического окна график выводится в существующем окне с дескриптором `h`. Параметр `h` должен быть дескриптором графического окна, ранее созданного функцией `eyediagram`.

### **ВНИМАНИЕ!**

Вывод диаграмм для нескольких сигналов в одном окне *нельзя* реализовать с помощью команды `hold on`.

Выходным параметром `h` является дескриптор графического окна, содержащего диаграмму.

Использование глазковых диаграмм иллюстрируется демонстрационной программой `scattereyedemo`, имеющейся в составе пакета Communications.

### **ЗАМЕЧАНИЕ**

Помимо функции `eyediagram`, вывод глазковой диаграммы в настоящее время реализован также с помощью объекта класса `commscope.eyediagram`.

## **Нормировка сигнала по мощности**

Как уже было сказано при описании объектов модуляторов и демодуляторов, используемые ими сигнальные созвездия имеют строго определенный масштаб — для АМн и КАМ точки лежат на целочисленной сетке, а при ФМн — на единичной окружности. Функция `modnorm` позволяет рассчитать нормировочный коэффициент для приведения АМн, ФМн или КАМ-сигнала к заданной средней или пиковой мощности. Функция имеет следующий синтаксис вызова:

```
scale = modnorm(const, pow_type, pow)
```

Здесь `const` — вектор комплексных точек используемого созвездия (свойство `Constellation` объектов модуляторов и демодуляторов), `pow_type` — строковый параметр, задающий тип нормировки ('`avpow`' — приведение к заданной средней мощности, '`peakpow`' — приведение к заданной пиковой мощности), `pow` — желаемая мощность сигнала (средняя или пиковая, в зависимости от значения параметра `pow_type`).

Результатом работы функции является нормировочный коэффициент `scale` — после умножения на него сигнал будет иметь заданную мощность.

## **Пример формирования и обработки сигнала с цифровой модуляцией**

Как уже было сказано, объекты цифровой модуляции и демодуляции для большинства видов модуляции просто осуществляют взаимное отображение элементов сообщения и точек на комплексной плоскости. Во многих случаях этого оказывается достаточно для моделирования исследуемых аспектов систем цифровой связи. Если же, тем не менее, необходимо сформировать вещественный модулированный сигнал на некоторой несущей частоте, для этого необходимо выполнить следующие действия:

1. Сформировать набор комплексных амплитуд, соответствующих символам цифрового сигнала, с помощью соответствующего объекта модулятора.
2. Выполнить интерполяцию сигнала с использованием желаемого формирующего фильтра. Для этого в составе Communications Toolbox имеются функции `rectpulse` (она осуществляет кусочно-постоянную интерполяцию, задавая тем самым сигнальные посылки в виде прямоугольных импульсов) и `rcosflt` (интерполяция производится с помощью фильтра с косинусоидальным сглаживанием АЧХ; такие фильтры рассматривались в главе 6).

3. Полученные значения интерполированной комплексной огибающей нужно использовать для осуществления аналоговой квадратурной модуляции. Готовой функции для этого в Communications Toolbox нет, но можно воспользоваться функцией `ammod` дважды, чтобы отдельно сформировать синфазную и квадратурную составляющие сигнала (в приводимом далее коде предполагается, что отсчеты комплексной огибающей сохранены в виде вектора `xc`):

```
y = ammod(real(xc), Fc, Fs) + ammod(imag(xc), Fc, Fs, pi/2);
```

Еще один вариант — использовать функцию `modulate` пакета Signal Processing:

```
y = modulate(real(xc), Fc, Fs, 'qam', imag(xc));
```

Наконец, можно сформировать сигнал вручную:

```
t = (0:length(xc)-1)'/Fs; % вектор-столбец моментов времени
carrier = exp(2i*pi*Fc*t); % несущее колебание
s = real(xc .* carrier); % сформированный вещественный сигнал
```

Для реализации демодуляции вещественного сигнала выполняются обратные операции:

- Сигнал умножается на несущее колебание в виде комплексной экспоненты. Для этого можно использовать те же три строки кода, что были приведены чуть раньше применительно к "ручному" формированию вещественного сигнала, в них необходимо только изменить знак в показателе комплексной экспоненты.
- Полученный комплексный сигнал обрабатывается приемным фильтром, после чего результат прореживается — из него выбирается каждый  $N$ -й отсчет, где  $N$  — число отсчетов на символ (в соответствии с используемыми в пакете Communications обозначениями это число рассчитывается как  $F_s/F_d$ ). Если при формировании сигнала использовалась функция `rectpulse`, эти две операции можно выполнить с помощью функции `intdump` (интегратор со сбросом). В остальных случаях фильтрацию и прореживание сигнала придется выполнять по отдельности. При этом важно правильно определить номер первого выбираемого отсчета.
- Отсчеты, полученные после прореживания, обрабатываются с помощью одного из объектов демодуляторов.

### **ЗАМЕЧАНИЕ**

В случае частотной и минимальной частотной манипуляции для получения и демодуляции вещественного сигнала необходимо выполнить только первый и третий пункты из приведенных списков.

В качестве примера произведем формирование и прием сигнала с 16-позиционной КАМ, аналогичного использованному в примерах, приведенных в теоретической части главы (см. рис. 8.41—8.43). Заодно продемонстрируем и использование функции `modnorm`, приведя с ее помощью сигнал к единичной средней мощности:

```
>> % задание параметров
>> m = 4; % число битов в символе
>> M = 2^m; % размер созвездия
```

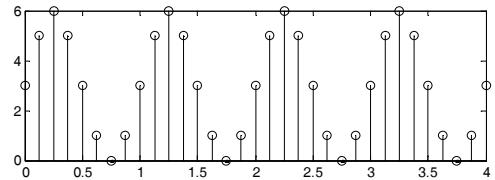
```

>> Nsym = 1000; % число символов
>> Nbit = Nsym*m; % число битов
>> Fd = 2400; % символьная скорость
>> Fc = 1800; % несущая частота
>> FsFd = 4; % число отсчетов на один символ
>> Fs = Fd * FsFd; % частота дискретизации
>> % формирование КАМ-сигнала
>> x = randint(Nbit, 1); % случайный набор битов
>> hmod = modem.qammod('M', M); % объект модулятора
>> hmod.InputType = 'bit'; % на входе – биты
>> xm = hmod.modulate(x); % преобразуем в точки на комплексной плоскости
>> A = modnorm(hmod.Constellation, 'avpow', 1); % расчет масштаба
>> xm = xm * A; % приведение к единичной мощности
>> xc = rcosflt(xm, Fd, Fs, 'sqrt'); % интерполяция
>> t = (0:length(xc)-1)'/Fs; % вектор-столбец моментов времени
>> carrier = exp(2i*pi*Fc*t); % несущее колебание
>> s = real(xc .* carrier); % сформированный вещественный сигнал
>> % прием КАМ-сигнала
>> yc = s .* conj(carrier) * 2; % умножение на опорное колебание
>> b = rcosine(Fd, Fs, 'sqrt'); % приемный согласованный фильтр
>> yc = filter(b, 1, yc); % фильтрация
>> offset = 1; % выбор моментов взятия отсчетов
>> ym = yc(offset:FsFd:end); % дискретизация с символьной частотой
>> ym(1:6) = []; % удаление начального "хвоста"
>> ym = ym/A; % обратное масштабирование
>> hdemod = modem.qamdemod('M', M); % объект демодулятора
>> hdemod.OutputType = 'bit'; % на выходе – биты
>> z = hdemod.demodulate(ym); % собственно демодуляция КАМ-сигнала
>> biterr(x, z) % число битовых ошибок
ans =
    0

```

Как видим, сигнал демодулирован правильно.

# ГЛАВА 9



## Адаптивные фильтры

При поиске оптимальных алгоритмов обработки сигнала неизбежно приходится опираться на некоторые *статистические модели* сигналов и шумов. Чаще всего при формировании этих моделей используются концепции линейности, стационарности и нормальности (гауссовости). Однако перечисленные принципы далеко не всегда выполняются на практике, а от адекватности выбранной модели в значительной мере зависит качество приема сигнала. Адаптивные фильтры позволяют системе подстраиваться под статистические параметры входного сигнала, не требуя при этом задания каких-либо моделей.

Появившись в конце 1950-х гг., адаптивные фильтры прошли большой путь, превратившись из экзотической технологии, применявшейся преимущественно в военных целях, в "ширпотреб", без которого сейчас была бы немыслима работа модемов, сотовых телефонов и многое другое.

Существует большое количество адаптивных алгоритмов, различающихся вычислительной сложностью, особенностями поведения, используемыми исходными данными и структурами самих адаптирующихся систем. В данной главе будут рассмотрены лишь несколько основных алгоритмов, но сначала следует остановиться на классификации адаптивных систем.

### ЗАМЕЧАНИЕ

К сожалению, число книг на русском языке, посвященных адаптивной фильтрации, крайне ограничено. В списке литературы приведены книги [13] и [14], некоторая информация имеется также в классическом труде по цифровой связи [35]. Однако эти публикации не затрагивают целого ряда важных вопросов. В качестве дополнительных, но, к сожалению, менее доступных источников информации можно рекомендовать замечательную книгу Саймона Хайкина (S. Haykin. Adaptive Filter Theory. Prentice Hall, четыре издания вышли в 1986, 1991, 1996 и 2002 гг.), а также обзорную статью, содержащую сведения о большом числе адаптивных алгоритмов (Glentis G.-O., Berberidis K., Theodoridis S. Efficient Least Squares Adaptive Algorithms for FIR Transversal Filtering // IEEE Signal Processing Magazine, 1999, Vol. 16, No. 4, pp. 13–41).

## Основные понятия адаптивной обработки сигналов

Важнейшим признаком классификации является наличие или отсутствие *образцового*, или *опорного* сигнала (*desired signal, reference signal*). При наличии образцового сигнала процесс адаптации называется *обучением с учителем* (*supervised learning*). В данном случае адаптивный фильтр стремится сделать свой выходной сигнал *максимально близким* к образцовому сигналу. О том, что именно подразумевается под близостью сигналов, сказано далее при описании конкретных алгоритмов. Адаптация без использования образцового сигнала называется *слепой адаптацией* (*blind adaptation*) или *обучением без учителя* (*unsupervised learning*). Разумеется, в этом случае требуется некоторая информация о структуре полезного входного сигнала (например, знание типа и параметров используемой модуляции). Очевидно, что слепая адаптация является более сложной вычислительной задачей, чем адаптация с использованием образцового сигнала.

### ЗАМЕЧАНИЕ

У внимательного читателя должен возникнуть вопрос — какой практический смысл могут иметь алгоритмы с использованием образцового сигнала, если при этом выходной сигнал должен быть заранее известен? Однако есть целый ряд практических задач, при решении которых образцовый сигнал оказывается доступен. Подробнее об этом далее в разд. *"Применение адаптивных фильтров"* этой главы. Забегая вперед, заметим, что в ряде случаев при этом полезным сигналом является не выходной сигнал фильтра, а *сигнал ошибки*, т. е. разность между образцовым сигналом и выходным сигналом адаптивного фильтра.

Следующим признаком классификации является тип системы, осуществляющей обработку сигнала. Различают *линейные* и *нелинейные* адаптивные системы, при этом имеется в виду линейность по отношению не ко входному сигналу, а к *параметрам системы, настраиваемым в процессе адаптации*.

Наиболее распространены линейные адаптивные системы, в которых обработка сигнала производится нерекурсивным дискретным фильтром. Одним из главных достоинств такого варианта является то, что нерекурсивный фильтр является устойчивым при любых значениях его коэффициентов. Однако следует отметить, что алгоритм адаптации в любом случае вносит в систему обратную связь, вследствие чего адаптивная система в целом может стать неустойчивой.

К нелинейным адаптивным системам относятся, в частности, *нейронные сети* (*neural networks*), в определенной степени моделирующие работу нервной системы живых организмов. Рассмотрение таких систем выходит за рамки тематики данной книги. За дополнительной информацией можно обратиться, например, к книгам [43, 44].

Еще одна разновидность нелинейных адаптивных систем — рекурсивные адаптивные фильтры. Однако их создание связано с серьезными проблемами, прежде всего связанными с устойчивостью, поэтому такие фильтры не получили широкого распространения. Некоторые теоретические сведения на эту тему содержатся в [13].

Итак, в данной главе речь пойдет об адаптивных фильтрах, использующих образцовый сигнал. Общая структура такой адаптивной системы показана на рис. 9.1.

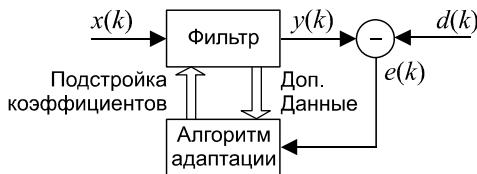


Рис. 9.1. Общая структура адаптивного фильтра

Входной дискретный сигнал  $x(k)$  обрабатывается дискретным фильтром, в результате чего получается выходной сигнал  $y(k)$ . Этот выходной сигнал сравнивается с *образцовым* сигналом  $d(k)$ , разность между ними образует сигнал ошибки  $e(k)$ . Задача адаптивного фильтра — минимизировать ошибку воспроизведения образцового сигнала. С этой целью блок адаптации после обработки каждого отсчета анализирует сигнал ошибки и дополнительные данные, поступающие из фильтра, используя результаты этого анализа для подстройки параметров (коэффициентов) фильтра.

В данной главе рассмотрены два адаптивных алгоритма, часто применяемых на практике в различных радиотехнических системах. Это алгоритмы *LMS* (Least Mean Square, метод наименьшего квадрата, в отечественных источниках иногда используется аббревиатура *MНК*), и *RLS* (Recursive Least Squares, рекурсивный метод наименьших квадратов, в отечественных источниках иногда используется аббревиатура *RНК*).

Выход формул, описывающих данные алгоритмы, производится на основе уравнений оптимальной фильтрации сигнала. Возможны различные подходы к решению задачи оптимальной фильтрации. Статистический подход в сочетании с методом градиентной оптимизации дает алгоритм LMS, а детерминированный подход приводит к алгоритму RLS.

Далее мы рассмотрим перечисленные подходы к решению задачи оптимальной фильтрации и идеи, лежащие в основе адаптивных алгоритмов.

Все перечисленное относится к алгоритмам, использующим образцовый сигнал (т. е. к обучению с учителем).

## Оптимальный фильтр Винера

Говоря об оптимальной фильтрации сигнала, следует помнить, что данная задача становится осмысленной лишь после задания двух вещей — математической модели входного сигнала и оптимизируемого критерия качества. Тогда задача оптимальной фильтрации сводится к математической оптимизационной задаче, которая может быть решена аналитически либо численно.

Итак, пусть входной дискретный случайный сигнал  $\{x(k)\}$  обрабатывается дискретным фильтром порядка  $N$  с коэффициентами  $\{w_n\}$ ,  $n = 0, 1, \dots, N$  (рис. 9.2).

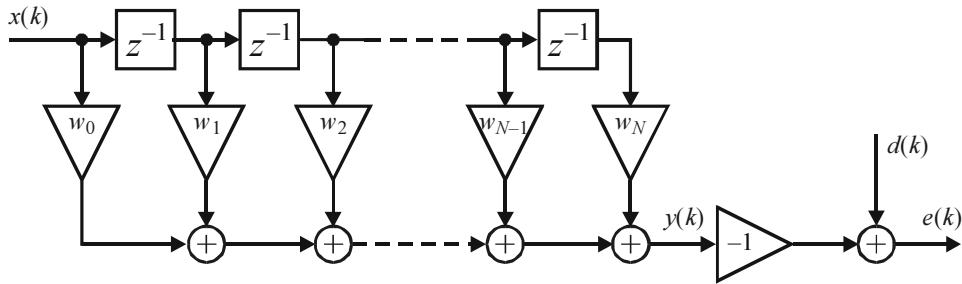


Рис. 9.2. Формирование сигнала ошибки

Выходной сигнал фильтра равен

$$y(k) = \sum_{n=0}^N w_n x(k-n). \quad (9.1)$$

Кроме того, имеется образцовый (также случайный) сигнал  $d(k)$ . Ошибка воспроизведения образцового сигнала равна

$$e(k) = d(k) - y(k) = d(k) - \sum_{n=0}^N w_n x(k-n). \quad (9.2)$$

Наша задача — найти такие коэффициенты фильтра  $\{w_n\}$ , которые обеспечивают максимальную близость выходного сигнала фильтра к образцовому, т. е. минимизируют ошибку  $e(k)$ . Но поскольку  $e(k)$  также является случайным процессом, в качестве меры ее величины разумно принять средний квадрат. Таким образом, оптимизируемая функция выглядит так:

$$J(\{w_n\}) = \overline{e^2(k)} \rightarrow \min.$$

Для решения поставленной задачи прежде всего перепишем (9.2) в матричном виде. Для этого обозначим вектор-столбец коэффициентов фильтра как  $\mathbf{w}$ , а вектор-столбец содержимого линии задержки фильтра на  $k$ -м шаге как  $\mathbf{x}(k)$ :

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_N \end{bmatrix}, \quad \mathbf{x}(k) = \begin{bmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-N) \end{bmatrix}.$$

С учетом этих обозначений (9.2) принимает следующий вид:

$$e(k) = d(k) - \mathbf{x}^T(k)\mathbf{w}. \quad (9.3)$$

Квадрат ошибки будет равен

$$\begin{aligned} e^2(k) &= (d(k) - \mathbf{x}^T(k)\mathbf{w})^2 = d^2(k) - 2d(k)\mathbf{x}^T(k)\mathbf{w} + (\mathbf{x}^T(k)\mathbf{w})^2 = \\ &= d^2(k) - 2d(k)\mathbf{x}^T(k)\mathbf{w} + \mathbf{w}^T \mathbf{x}(k) \mathbf{x}^T(k) \mathbf{w}. \end{aligned}$$

Статистически усредняя это выражение, получаем следующее:

$$J(\mathbf{w}) = \overline{e^2(k)} = \overline{d^2(k)} - 2\left(\overline{d(k)x(k)}\right)^T \mathbf{w} + \mathbf{w}^T \overline{x(k)x^T(k)} \mathbf{w}. \quad (9.4)$$

Рассмотрим подробнее входящие в полученную формулу усредненные величины:

- $\overline{d^2(k)}$  — это средний квадрат образцового сигнала. Он представляет собой отдельное слагаемое, которое не зависит от коэффициентов фильтра и потому может быть отброшено (однако оно влияет на величину среднего квадрата ошибки, получаемую при оптимальных значениях коэффициентов фильтра). Обозначим данную величину как  $\sigma_d^2$ ;
- $\overline{d(k)x(k)}$  — это вектор-столбец взаимных корреляций между  $k$ -м отсчетом образцового сигнала и содержимым линии задержки фильтра на  $k$ -м шаге. Будем считать случайные процессы  $x(k)$  и  $d(k)$  совместно стационарными, тогда вектор взаимных корреляций не зависит от номера шага  $k$ . В дальнейших выкладках этот вектор будет обозначен как  $\mathbf{p}$ :

$$\mathbf{p} = \begin{bmatrix} \overline{d(k)x(k)} \\ \overline{d(k)x(k-1)} \\ \vdots \\ \overline{d(k)x(k-N)} \end{bmatrix}.$$

- $\overline{x(k)x^T(k)}$  — это квадратная матрица размером  $(N+1) \times (N+1)$ , являющаяся корреляционной матрицей сигнала (см. разд. "Дискретные случайные процессы" главы 3). Как отмечалось в указанном разделе, для стационарного случайного процесса корреляционная матрица имеет вид матрицы Тэплица, вдоль диагоналей которой стоят значения корреляционной функции:

$$\mathbf{R} = \begin{bmatrix} R_x(0) & R_x(1) & R_x(2) & \dots & R_x(N) \\ R_x(1) & R_x(0) & R_x(1) & \dots & R_x(N-1) \\ R_x(2) & R_x(1) & R_x(0) & \dots & R_x(N-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_x(N) & R_x(N-1) & R_x(N-2) & \dots & R_x(0) \end{bmatrix},$$

где  $R_x(\Delta k) = \overline{x(k)x(k-\Delta k)}$  — КФ входного сигнала.

С учетом введенных обозначений (9.4) принимает следующий вид:

$$J(\mathbf{w}) = \sigma_d^2 - 2\mathbf{p}^T \mathbf{w} + \mathbf{w}^T \mathbf{R} \mathbf{w}. \quad (9.5)$$

Данное выражение представляет собой квадратичную форму относительно  $\mathbf{w}$  и потому при невырожденной матрице  $\mathbf{R}$  имеет единственный минимум, для нахождения которого необходимо приравнять к нулю вектор градиента:

$$\nabla J(\mathbf{w}) = -2\mathbf{p} + 2\mathbf{R} \mathbf{w} = \mathbf{0}. \quad (9.6)$$

Отсюда получаем равенство, называемое *уравнением Винера — Хопфа (Wiener — Hopf equation)*:

$$\mathbf{R} \mathbf{w} = \mathbf{p}. \quad (9.7)$$

Умножив слева обе части равенства на обратную корреляционную матрицу  $\mathbf{R}^{-1}$ , получаем искомое решение для оптимальных коэффициентов фильтра:

$$\mathbf{w} = \mathbf{R}^{-1} \mathbf{p}. \quad (9.8)$$

Такой фильтр называется *фильтром Винера (Wiener filter)*. Подстановка (9.8) в (9.5) дает минимально достижимую дисперсию сигнала ошибки:

$$\overline{e^2(k)}_{\min} = \sigma_d^2 - \mathbf{p}^T \mathbf{R}^{-1} \mathbf{p}. \quad (9.9)$$

Несложно также показать, что  $\overline{e(k)y(k)} = 0$  и  $\overline{e(k)x(k)} = 0$ , т. е. что сигнал ошибки для фильтра Винера некоррелирован с входным и выходным сигналами фильтра.

В качестве примера с помощью MATLAB рассчитаем оптимальный фильтр для коррекции искажений, вносимых в передаваемый сигнал  $x_0(k)$  каналом связи, имеющим четырехэлементную импульсную характеристику следующего вида:

$$\{h_k\} = \{-2, -4, 6, 3\}, \quad k = 0, 1, 2, 3.$$

Отсчеты передаваемого сигнала будем считать независимыми случайными величинами с нулевым средним значением и единичной дисперсией. В этом случае корреляционная функция сигнала, прошедшего через канал связи, будет совпадать с корреляционной функцией импульсной характеристики канала:

$$\{R(\Delta k)\} = \left\{ \sum_k h_k h_{k-\Delta k} \right\} = \{65, 2, -24, -6\}, \quad \Delta k = 0, 1, 2, 3.$$

Корреляционная матрица входного сигнала строится как *матрица Теллица* на основе данной корреляционной функции:

$$\mathbf{R} = \begin{bmatrix} 65 & 2 & -24 & -6 & 0 & \dots & 0 \\ 2 & 65 & 2 & -24 & -6 & \dots & 0 \\ -24 & 2 & 65 & 2 & -24 & \dots & 0 \\ -6 & -24 & 2 & 65 & 2 & \dots & 0 \\ 0 & -6 & -24 & 2 & 65 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 65 \end{bmatrix}.$$

Восстановление переданного сигнала неизбежно требует внесения некоторой временной задержки, поэтому образцовый сигнал должен представлять собой задержанную копию переданного:

$$d(k) = x_0(k - \Delta k).$$

В линии задержки фильтра на  $k$ -м шаге находятся отсчеты искаженного сигнала с номерами  $k, k-1, k-2, \dots, k-N$ , где  $N$  — порядок фильтра. Каждый из этих отсчетов представляет собой линейную комбинацию отсчетов переданного сигнала:

$$x(k-n) = \sum_{m=-\infty}^{\infty} x_0(m)h_{k-n-m}. \quad (9.10)$$

Поскольку отсчеты исходного сигнала считаются статистически независимыми, при вычислении  $n$ -го элемента вектора  $\mathbf{p}$  результат усреднения будет отличен от нуля только для одного слагаемого (9.10). Сразу же учтем тот факт, что средний квадрат сигнала  $x_0(k)$  равен единице:

$$\begin{aligned} p_n &= \overline{x(k-n)d(k)} = \overline{\sum_{m=-\infty}^{\infty} x_0(m)h_{k-n-m} x_0(k-\Delta k)} = \\ &= \sum_{m=-\infty}^{\infty} h_{k-n-m} \overline{x_0(m)x_0(k-\Delta k)} = h_{\Delta k-m}. \end{aligned}$$

Таким образом, вектор  $\mathbf{p}$  содержит *перевернутую* импульсную характеристику канала, при необходимости обрезанную или дополненную нулями с одной или двух сторон:

$$\mathbf{p} = \begin{bmatrix} h_{\Delta k} \\ h_{\Delta k-1} \\ \vdots \\ h_1 \\ h_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

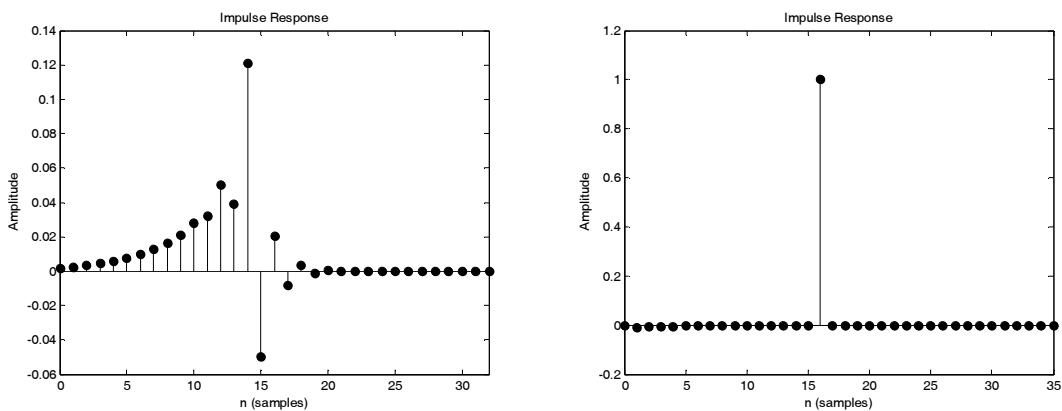
Далее приводится код MATLAB-программы, реализующей расчет оптимального фильтра. Код составлен так, чтобы можно было легко варьировать вид импульсной характеристики канала и порядок рассчитываемого фильтра (единственное ограничение — предполагается, что импульсная характеристика канала связи целиком укладывается в пределах вектора  $\mathbf{p}$ ):

```

h = [-2 -4 6 3]; % импульсная характеристика канала связи
N = 32;           % порядок рассчитываемого фильтра
r = xcorr(h, N); % двусторонняя КФ импульсной характеристики канала
r = r(N+1:end); % односторонняя КФ импульсной характеристики канала
R = toeplitz(r); % корреляционная матрица искаженного сигнала
p = zeros(N+1, 1);
delay = 16;        % задержка фильтрации
p(delay-length(h)+2:delay+1) = fliplr(h); % вектор взаимных корреляций
w = R\p;           % коэффициенты оптимального фильтра
subplot(1,2,1)

```

```
impz(w) % график имп. х-ки рассчитанного фильтра
subplot(1,2,2)
impz(conv(h, w)) % график имп. х-ки скорректированного канала
```



**Рис. 9.3.** Импульсные характеристики корректирующего фильтра (слева) и скорректированного канала связи (справа)

#### ЗАМЕЧАНИЕ

В приведенном коде следует обратить внимание на использование оператора матричного деления (\) для решения системы линейных уравнений (9.7).

На рис. 9.3 приведены графики импульсной характеристики корректирующего фильтра и сквозной импульсной характеристики скорректированного канала связи. Видно, что сквозная характеристика близка к единичному импульсу, что говорит о хорошем качестве коррекции. Столь качественная коррекция оказалась возможна потому, что частотная характеристика канала не имеет глубоких провалов (с помощью функции freqz легко убедиться в том, что неравномерность АЧХ канала составляет около 11 дБ).

## Градиентный поиск оптимального решения

Один из наиболее распространенных адаптивных алгоритмов основан на поиске минимума целевой функции (9.4) методом наискорейшего спуска. При использовании данного способа оптимизации вектор коэффициентов фильтра зависит от номера итерации  $k$ :  $\mathbf{w}(k)$ . На каждой итерации вектор коэффициентов смещается на величину, пропорциональную градиенту целевой функции в данной точке (см. формулу (9.6)):

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{\mu}{2} \nabla J(\mathbf{w}(k)) = \mathbf{w}(k) + \mu \mathbf{p} - \mu \mathbf{R} \mathbf{w}(k), \quad (9.11)$$

где  $\mu$  — положительный коэффициент, называемый *размером шага*.

Анализ сходимости данного процесса приведен, например, в [13]. Показано, что алгоритм сходится, если

$$0 < \mu < 2/\lambda_{\max}, \quad (9.12)$$

где  $\lambda_{\max}$  — максимальное *собственное число* корреляционной матрицы  $\mathbf{R}$ . Скорость сходимости при этом зависит от разброса собственных чисел корреляционной матрицы  $\mathbf{R}$  — чем меньше отношение  $\lambda_{\max}/\lambda_{\min}$ , тем быстрее сходится итерационный процесс.

## Адаптивный алгоритм LMS

При реализации наискорейшего спуска по формуле (9.11) нужно вычислять значения градиента, а для их расчета, в свою очередь, необходимо знать значения матрицы  $\mathbf{R}$  и вектора  $\mathbf{p}$ . На практике могут быть доступны лишь *оценки* этих параметров, получаемые по входным данным. Простейшими такими оценками являются *мгновенные* значения корреляционной матрицы и вектора взаимных корреляций, получаемые без какого-либо усреднения:

$$\hat{\mathbf{R}}(k) = \mathbf{x}(k)\mathbf{x}^T(k),$$

$$\hat{\mathbf{p}}(k) = d(k)\mathbf{x}(k).$$

При использовании данных оценок формула (9.11) принимает следующий вид:

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + \mu d(k)\mathbf{x}(k) - \mu \mathbf{x}(k)\mathbf{x}^T(k) \mathbf{w}(k) = \\ &= \mathbf{w}(k) + \mu \mathbf{x}(k) \left( d(k) - \mathbf{x}^T(k) \mathbf{w}(k) \right). \end{aligned} \quad (9.13)$$

Выражение, стоящее в скобках, согласно (9.3), представляет собой разность между образцовым сигналом и выходным сигналом фильтра на  $k$ -м шаге, т. е. ошибку фильтрации  $e(k)$ . С учетом этого выражение для рекурсивного обновления коэффициентов фильтра оказывается очень простым:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k). \quad (9.14)$$

Алгоритм адаптивной фильтрации, основанный на формуле (9.14), получил название *LMS* (Least Mean Square, метод наименьшего квадрата). Можно получить ту же формулу и несколько иным образом: использовав вместо градиента *статистически усредненного* квадрата ошибки  $\overline{e^2(k)}$  градиент его *мгновенного* значения  $e^2(k)$ .

Несмотря на простоту алгоритма LMS, анализ его поведения оказывается чрезвычайно сложной задачей, для которой не существует точного аналитического решения. Частные результаты, полученные при использовании различных приближений, можно найти в большом количестве книг и статей. Достаточно подробный анализ выполнен в упоминавшейся в начале главы книге С. Хайкина, при этом получены следующие результаты. Алгоритм LMS *сходится в среднем* (*converges in the mean values*; это означает, что математические ожидания коэффициентов фильтра при

$k \rightarrow \infty$  стремится к оптимальному решению (9.8)), если коэффициент  $\mu$  удовлетворяет тому же условию (9.12), которое требуется для сходимости детерминированного градиентного спуска. Однако это еще не гарантирует того, что *средние квадраты* коэффициентов будут стремиться к фиксированным значениям. Алгоритм *сходится в среднем квадрате* (*converges in the mean square*) в более узком диапазоне значений  $\mu$ :

$$\mu_{\max} \approx \frac{2}{\sum_{n=0}^N \lambda_n} = \frac{2}{\text{trace}(\mathbf{R})} = \frac{2}{(N+1)\sigma_x^2}, \quad (9.15)$$

где  $\lambda_k$  — собственные числа корреляционной матрицы  $\mathbf{R}$ , а  $\sigma_x^2$  — средний квадрат входного сигнала фильтра.

На формуле (9.15) основан *нормированный* (normalized) LMS-алгоритм, в котором коэффициент  $\mu$  на каждом шаге рассчитывается исходя из энергии сигнала, содержащегося в линии задержки:

$$\mu(k) = \frac{\mu_0}{\mathbf{x}^T \mathbf{x} + \varepsilon}, \quad (9.16)$$

где  $\mu_0$  — нормированное значение  $\mu$ , лежащее в диапазоне от 0 до 2, а  $\varepsilon$  — малая положительная константа, назначение которой — ограничить рост  $\mu$  при нулевом сигнале на входе фильтра (из формулы (9.16) видно, что максимальное значение  $\mu$  составляет  $\mu_0/\varepsilon$ ).

Даже если LMS-алгоритм сходится, дисперсии коэффициентов фильтра при  $k \rightarrow \infty$  не стремятся к нулю — коэффициенты флюктуируют вокруг оптимальных значений. Из-за этого ошибка фильтрации в установившемся режиме оказывается больше, чем ошибка для винеровского фильтра  $\overline{e^2(k)}_{\min}$  (см. ранее формулу (9.9)):

$$\overline{e^2(k)}_{\text{LMS}} = \overline{e^2(k)}_{\min} + E_{\text{ex}},$$

где  $E_{\text{ex}}$  — средний квадрат *избыточной ошибки* (*excess error*) алгоритма LMS. В той же книге С. Хайкина приводится следующая приближенная формула для так называемого *коэффициента расстройки* (*misalignment*), равного отношению средних квадратов избыточной и винеровской ошибок:

$$M = \frac{\overline{e^2(k)}_{\text{LMS}}}{\overline{e^2(k)}_{\min}} - 1 = \frac{E_{\text{ex}}}{\overline{e^2(k)}_{\min}} \approx \frac{\mu \sum_{n=0}^N \lambda_n}{2 - \mu \sum_{n=0}^N \lambda_n} = \frac{\mu(N+1)\sigma_{\text{bx}}^2}{2 - \mu(N+1)\sigma_{\text{bx}}^2} = \frac{\mu}{\mu_{\max} - \mu}. \quad (9.17)$$

Значение коэффициента  $\mu$  влияет на два главных параметра LMS-фильтра — скорость сходимости и коэффициент расстройки. Чем больше  $\mu$ , тем быстрее сходится алгоритм, но тем больше, согласно (9.17), становится коэффициент расстройки, и наоборот.

Основным достоинством алгоритма LMS является предельная вычислительная простота — для подстройки коэффициентов фильтра на каждом шаге нужно выполнить  $N + 1$  пар операций "умножение-сложение". Платой за простоту являются медленная сходимость и повышенная (по сравнению с минимально достижимым значением (9.9)) дисперсия ошибки в установившемся режиме.

Существует большое число модификаций алгоритма LMS (см., например, обзорную статью, упомянутую в начале этой главы), направленных на ускорение сходимости либо на уменьшение числа арифметических операций. Ускорение сходимости может быть достигнуто за счет улучшения используемой оценки градиента, а также за счет преобразования входного сигнала с целью сделать его отсчеты некоррелированными. Уменьшение вычислительной сложности может быть достигнуто, в частности, за счет использования в (9.14) не самих сигнала ошибки и содержимого линии задержки фильтра, а лишь их знаков. Это позволяет полностью избавиться от операций умножения при обновлении коэффициентов фильтра. В целом следует отметить, что требования ускорения сходимости и сокращения вычислительных затрат являются взаимоисключающими.

## Детерминированная задача оптимальной фильтрации

Рассматривая статистическую задачу оптимизации, мы считали входной сигнал *случайным процессом* и минимизировали *дисперсию* ошибки воспроизведения образцового сигнала. Однако возможен и иной подход, не использующий статистические методы.

Итак, пусть обработке подвергается последовательность отсчетов  $\{x(k)\}$ , коэффициенты нерекурсивного фильтра порядка  $N$  образуют набор  $\{w_n\}$ ,  $n = 0, 1, \dots, N$ , а отсчеты образцового сигнала равны  $\{d(k)\}$ . Выходной сигнал фильтра определяется формулой (9.1), а ошибка воспроизведения образцового сигнала — формулой (9.2) или, в векторном виде, (9.3).

Теперь сформулируем детерминированную оптимизационную задачу: мы хотим отыскать такие коэффициенты фильтра  $\{w_n\}$ , чтобы суммарная квадратичная ошибка воспроизведения образцового сигнала была минимальной:

$$J(\{w_n\}) = \sum_{k=0}^{K-1} |e(k)|^2 \rightarrow \min. \quad (9.18)$$

Для решения задачи необходимо перейти в формуле (9.3) к матричной записи, так сказать, вдоль координаты  $k$ , получив формулы для векторов-столбцов выходного сигнала  $\mathbf{y}$  и ошибки воспроизведения входного сигнала  $\mathbf{e}$ :

$$\mathbf{y} = \mathbf{X}^T \mathbf{w}, \quad \mathbf{e} = \mathbf{d} - \mathbf{X}^T \mathbf{w}. \quad (9.19)$$

Здесь  $\mathbf{d}$  — вектор-столбец отсчетов образцового сигнала, а  $\mathbf{X}$  — матрица, столбцы которой представляют собой содержимое линии задержки фильтра на разных тактах:

$$\mathbf{d} = \begin{bmatrix} d(0) \\ d(1) \\ \vdots \\ d(K-1) \end{bmatrix}, \quad \mathbf{X} = [\mathbf{x}(0) \ \mathbf{x}(1) \ \dots \ \mathbf{x}(N-1)].$$

Выражение (9.18) для суммарной квадратичной ошибки в матричном виде можно записать следующим образом:

$$J(\mathbf{w}) = \mathbf{e}^T \mathbf{e} \rightarrow \min. \quad (9.20)$$

Подставив (9.19) в (9.20), имеем

$$\begin{aligned} J(\mathbf{w}) &= (\mathbf{d} - \mathbf{X}^T \mathbf{w})^T (\mathbf{d} - \mathbf{X}^T \mathbf{w}) = \\ &= \mathbf{d}^T \mathbf{d} - (\mathbf{X}^T \mathbf{w})^T \mathbf{d} - \mathbf{d}^T (\mathbf{X}^T \mathbf{w}) + (\mathbf{X}^T \mathbf{w})^T (\mathbf{X}^T \mathbf{w}) = \\ &= \mathbf{d}^T \mathbf{d} - \mathbf{w}^T \mathbf{X} \mathbf{d} - \mathbf{d}^T \mathbf{X}^T \mathbf{w} + \mathbf{w}^T \mathbf{X} \mathbf{X}^T \mathbf{w}. \end{aligned}$$

Для нахождения минимума необходимо вычислить градиент данного функционала и приравнять его к нулю:

$$\nabla J(\mathbf{w}) = -2\mathbf{X}\mathbf{d} + 2\mathbf{X}\mathbf{X}^T \mathbf{w} = \mathbf{0}.$$

Отсюда легко получается искомое оптимальное решение:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^T)^{-1} \mathbf{X}\mathbf{d}. \quad (9.21)$$

В формуле (9.21) прослеживается близкое родство с формулой (9.8), описывающей оптимальный в статистическом смысле фильтр Винера. Действительно, если учесть, что  $(\mathbf{X}\mathbf{X}^T)^{-1}/K$  дает оценку корреляционной матрицы сигнала, полученную по одной реализации эргодического случайного процесса путем временного усреднения, а  $\mathbf{X}\mathbf{d}/K$  является аналогичной оценкой взаимных корреляций между образцовым сигналом и содержимым линии задержки фильтра, то формулы (9.8) и (9.21) совпадут.

## Адаптивный алгоритм RLS

В принципе, в процессе приема сигнала можно на каждом очередном шаге пересчитывать коэффициенты фильтра непосредственно по формуле (9.21), однако это связано с неоправданно большими вычислительными затратами. Действительно, на каждом шаге увеличивается размер матрицы  $\mathbf{X}$ ; кроме того, необходимо каждый раз заново вычислять обратную матрицу  $(\mathbf{X}\mathbf{X}^T)^{-1}$ .

Сократить вычислительные затраты можно, если заметить, что на каждом шаге к матрице  $\mathbf{X}$  добавляется лишь один новый столбец, а к вектору  $\mathbf{d}$  — один новый элемент. Это дает возможность организовать вычисления *рекурсивно*. Соответствующий алгоритм получил название *рекурсивного метода наименьших квадратов* (*Recursive Least Squares, RLS*).

На  $k$ -м шаге оптимальный вектор коэффициентов фильтра, согласно (9.21), равен

$$\mathbf{w}(k) = \left( \mathbf{X}(k) \mathbf{X}^T(k) \right)^{-1} \mathbf{X}(k) \mathbf{d}(k). \quad (9.22)$$

Посмотрим, какие новые данные добавляются к  $\mathbf{X}(k)$  и  $\mathbf{d}(k)$  на следующем шаге и как мы можем использовать их для обновления рассчитанного ранее решения — ведь именно в таком постоянном обновлении состоит суть работы адаптивного фильтра.

К матрице  $\mathbf{X}$  на каждом шаге добавляется новый столбец  $\mathbf{x}(k+1)$ , а к вектору  $\mathbf{d}$  — новый элемент  $d(k+1)$ :

$$\mathbf{X}(k+1) = [\mathbf{X}(k) \mid \mathbf{x}(k+1)], \quad \mathbf{d}(k+1) = \begin{bmatrix} \mathbf{d}(k) \\ d(k+1) \end{bmatrix}. \quad (9.23)$$

В алгоритме RLS производится рекурсивное обновление оценки обратной корреляционной матрицы

$$\mathbf{P}(k) = \left( \mathbf{X}(k) \mathbf{X}^T(k) \right)^{-1}.$$

При переходе к следующему шагу мы имеем

$$\begin{aligned} \mathbf{P}(k+1) &= \left( \mathbf{X}(k+1) \mathbf{X}^T(k+1) \right)^{-1} = \left( [\mathbf{X}(k) \mid \mathbf{x}(k+1)] \begin{bmatrix} \mathbf{X}^T(k) \\ \mathbf{x}^T(k+1) \end{bmatrix} \right)^{-1} = \\ &= \left( \mathbf{X}(k) \mathbf{X}^T(k) + \mathbf{x}(k+1) \mathbf{x}^T(k+1) \right)^{-1}. \end{aligned} \quad (9.24)$$

Теперь воспользуемся матричным тождеством

$$(\mathbf{A} + \mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} (\mathbf{C}^{-1} + \mathbf{D} \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{D} \mathbf{A}^{-1}, \quad (9.25)$$

которое является справедливым для произвольных квадратных невырожденных матриц  $\mathbf{A}$  и  $\mathbf{C}$  и произвольных матриц  $\mathbf{B}$  и  $\mathbf{D}$  совместимых размеров.

Установим между (9.24) и (9.25) следующее соответствие:

- $\mathbf{A} = \mathbf{X}(k) \mathbf{X}^T(k) = \mathbf{P}^{-1}(k)$ ,  $\mathbf{A}^{-1} = \mathbf{P}(k)$  (квадратная матрица);
- $\mathbf{B} = \mathbf{x}(k+1)$  (вектор-столбец);
- $\mathbf{C} = 1$  (скаляр);
- $\mathbf{D} = \mathbf{x}^T(k+1)$  (вектор-строка).

В результате равенство (9.24) можно записать следующим образом:

$$\mathbf{P}(k+1) = \mathbf{P}(k) - \mathbf{P}(k) \mathbf{x}(k+1) \left( 1 + \mathbf{x}^T(k+1) \mathbf{P}(k) \mathbf{x}(k+1) \right)^{-1} \mathbf{x}^T(k+1) \mathbf{P}(k). \quad (9.26)$$

Здесь следует заметить, что фрагмент выражения в круглых скобках, возводимый в минус первую степень, представляет собой скаляр. Далее используем (9.23) и (9.26) в выражении (9.22) для коэффициентов фильтра:

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{P}(k+1)\mathbf{X}(k+1)\mathbf{d}(k+1) = P(k+1)[\mathbf{X}(k) \mid \mathbf{x}(k+1)] \left[ \frac{\mathbf{d}(k)}{d(k+1)} \right] = \\
&= P(k+1)(\mathbf{X}(k)\mathbf{d}(k) + \mathbf{x}(k+1)d(k+1)) = \\
&= \left( \mathbf{P}(k) - \frac{\mathbf{P}(k)\mathbf{x}(k+1)\mathbf{x}^T(k+1)\mathbf{P}(k)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} \right) (\mathbf{X}(k)\mathbf{d}(k) + \mathbf{x}(k+1)d(k+1)).
\end{aligned}$$

Раскроем скобки в этом выражении:

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{P}(k)\mathbf{X}(k)\mathbf{d}(k) - \frac{\mathbf{P}(k)\mathbf{x}(k+1)\mathbf{x}^T(k+1)\mathbf{P}(k)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} \mathbf{X}(k)\mathbf{d}(k) + \\
&\quad + \mathbf{P}(k)\mathbf{x}(k+1)d(k+1) - \frac{\mathbf{P}(k)\mathbf{x}(k+1)\mathbf{x}^T(k+1)\mathbf{P}(k)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} \mathbf{x}(k+1)d(k+1).
\end{aligned}$$

Первое слагаемое в полученной формуле, согласно (9.22), представляет собой коэффициенты оптимального фильтра для  $k$ -го шага —  $\mathbf{w}(k)$ . Этот же вектор можно выделить в качестве множителя во втором слагаемом. Что касается третьего и четвертого слагаемых, в них можно выделить общий множитель  $\mathbf{P}(k) \mathbf{x}(k+1) d(k+1)$ :

$$\begin{aligned}
\mathbf{w}(k+1) &= \mathbf{w}(k) - \frac{\mathbf{P}(k)\mathbf{x}(k+1)\mathbf{x}^T(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} \mathbf{w}(k) + \\
&\quad + \mathbf{P}(k)\mathbf{x}(k+1)d(k+1) \left( 1 - \frac{\mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} \right).
\end{aligned}$$

Выполнив вычитание в круглых скобках, получаем

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{\mathbf{P}(k)\mathbf{x}(k+1)\mathbf{x}^T(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} \mathbf{w}(k) + \frac{\mathbf{P}(k)\mathbf{x}(k+1)d(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)}.$$

Теперь видно, что второе и третье слагаемые имеют общий множитель, который можно вынести за скобки:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mathbf{P}(k)\mathbf{x}(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} (d(k+1) - \mathbf{x}^T(k+1)\mathbf{w}(k)).$$

Далее заметим, что произведение  $\mathbf{x}^T(k+1) \mathbf{w}(k)$  есть результат обработки нового (поступившего на  $(k+1)$ -м шаге) входного сигнала фильтром со старыми (т. е. имеющимися на данный момент) коэффициентами  $\mathbf{w}(k)$ . Иными словами, это произведение представляет собой выходной сигнал адаптивного фильтра  $y(k+1)$ . Соответственно, получается, что разность, стоящая в скобках, — не что иное, как ошибка фильтрации  $e(k+1)$ :

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mathbf{P}(k)\mathbf{x}(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)} e(k+1).$$

Наконец, в качестве последнего штриха введем обозначение для вынесенного за скобки векторного множителя:

$$\mathbf{K}(k+1) = \frac{\mathbf{P}(k)\mathbf{x}(k+1)}{1 + \mathbf{x}^T(k+1)\mathbf{P}(k)\mathbf{x}(k+1)}.$$

С учетом этого формула для коэффициентов фильтра примет вид

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{K}(k+1) e(k+1). \quad (9.27)$$

Вектор  $\mathbf{K}(k+1)$  называют *вектором коэффициентов усиления*.

### ЗАМЕЧАНИЕ

Структура формулы (9.27) совпадает со структурой формулы (9.14), описывающей обновление коэффициентов фильтра для LMS-алгоритма — к текущему вектору коэффициентов фильтра добавляется слагаемое, рассчитываемое как произведение сигнала ошибки на некий вектор коэффициентов усиления. Различие состоит лишь в способе получения этого вектора — в алгоритме LMS он пропорционален содержимому линии задержки фильтра, а в алгоритме RLS рассчитывается более сложным образом.

Итак, при использовании адаптивного алгоритма RLS необходимо на каждом временном такте выполнить следующие шаги:

- При поступлении новых входных данных  $\mathbf{x}(k)$  производится фильтрация сигнала с использованием текущих коэффициентов фильтра  $\mathbf{w}(k-1)$  и вычисление величины ошибки воспроизведения образцового сигнала:

$$y(k) = \mathbf{x}(k)^T \mathbf{w}(k-1), \quad e(k) = d(k) - y(k).$$

- Рассчитывается вектор-столбец *коэффициентов усиления* (обратите внимание на то, что при вычислениях не используются предыдущие значения — вектор  $\mathbf{K}$  всякий раз рассчитывается заново, т. е. вычисления не являются рекурсивными; кроме того, еще раз отметим, что знаменатель дроби в следующих двух формулах является скаляром, а не матрицей):

$$\mathbf{K}(k) = \frac{\mathbf{P}(k-1)\mathbf{x}(k)}{1 + \mathbf{x}^T(k)\mathbf{P}(k-1)\mathbf{x}(k)}. \quad (9.28)$$

- Выполняется обновление оценки обратной корреляционной матрицы сигнала:

$$\mathbf{P}(k) = \mathbf{P}(k-1) - \frac{\mathbf{P}(k-1)\mathbf{x}(k)\mathbf{x}(k)^T\mathbf{P}(k-1)}{1 + \mathbf{x}(k)^T\mathbf{P}(k-1)\mathbf{x}(k)}. \quad (9.29)$$

- Наконец, производится обновление коэффициентов фильтра:

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{K}(k)e(k).$$

Осталось разобраться с начальными значениями рекурсивно обновляемых матрицы  $\mathbf{P}$  и вектора  $\mathbf{w}$ . Вектор коэффициентов фильтра  $\mathbf{w}$  перед началом работы алгоритма обычно заполняется нулями. Что касается матрицы  $\mathbf{P}$ , то строгий анализ показыва-

ет, что после заполнения линии задержки фильтра отсчетами сигнала результат вычислений не будет зависеть от начальных условий, если

$$\mathbf{P}(-1) = \begin{bmatrix} \infty & 0 & 0 & \dots & 0 \\ 0 & \infty & 0 & \dots & 0 \\ 0 & 0 & \infty & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \infty \end{bmatrix}.$$

На практике диагональ матрицы заполняется большими положительными значениями. Например, в [13] рекомендуется величина  $100/\sigma_x^2$ .

По сравнению с LMS-алгоритмом алгоритм RLS требует значительно большего числа вычислительных операций. При оптимальной организации вычислений для обновления коэффициентов фильтра необходимо  $2,5N^2 + 4N$  пар операций "умножение—сложение" [13]. Под оптимальной организацией здесь, в частности, подразумевается учет симметрии матрицы  $\mathbf{P}$ . Таким образом, число операций в алгоритме RLS квадратично возрастает с увеличением порядка фильтра.

Зато алгоритм RLS сходится значительно быстрее, чем алгоритм LMS. Строго говоря, само понятие сходимости применимо здесь лишь условно, поскольку данный алгоритм не является алгоритмом последовательного приближения. Алгоритм RLS на каждом шаге дает оптимальные коэффициенты фильтра, соответствующие формуле (9.21), и переходный процесс в начале работы связан лишь с рекурсивным расчетом оценки матрицы  $\mathbf{P}$  и постепенным заполнением линии задержки фильтра отсчетами входного сигнала.

## Экспоненциальное забывание

В формулах (9.18) и (9.20) значениям ошибки на всех временных тактах придается одинаковый вес. В результате, если статистические свойства входного сигнала со временем изменяются, это приведет к ухудшению качества фильтрации. Чтобы дать фильтру возможность отслеживать нестационарный входной сигнал, можно применить в (9.18) экспоненциальное забывание (*exponential forgetting*), при котором вес прошлых значений сигнала ошибки экспоненциально уменьшается:

$$J(\mathbf{w}) = \sum_{k=0}^{K-1} \lambda^{K-1-k} |e(k)|^2,$$

где  $\lambda$  — коэффициент забывания (forgetting factor),  $0 < \lambda \leq 1$ .

При использовании экспоненциального забывания формулы (9.28) и (9.29) принимают следующий вид:

$$\mathbf{K}(k) = \frac{\mathbf{P}(k-1)\mathbf{x}(k)}{\lambda + \mathbf{x}^T(k)\mathbf{P}(k-1)\mathbf{x}(k)},$$

$$\mathbf{P}(k) = \frac{1}{\lambda} \left( \mathbf{P}(k-1) - \mathbf{K}(k)\mathbf{x}^T(k)\mathbf{P}(k-1) \right).$$

## Применение адаптивных фильтров

Адаптивные фильтры в настоящее время нашли применение во многих системах обработки сигналов. В данном разделе мы рассмотрим лишь некоторые из возможных областей их использования.

### Идентификация систем

Все способы применения адаптивных фильтров так или иначе сводятся к решению задачи *идентификации*, т. е. определения характеристик, некоторой системы. Возможны два варианта идентификации — *прямая* и *обратная*. В первом случае адаптивный фильтр включается параллельно с исследуемой системой (рис. 9.4, а).

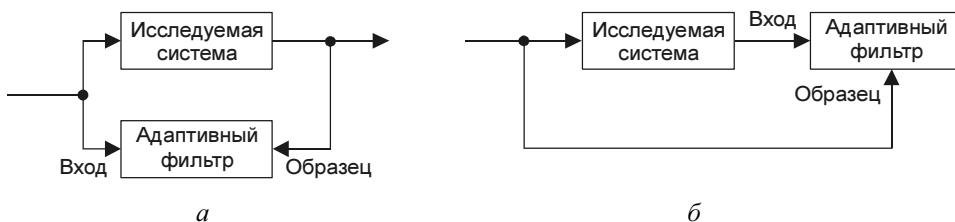


Рис. 9.4. Идентификация систем с помощью адаптивного фильтра:  
а — прямая, б — обратная

Входной сигнал является общим для исследуемой системы и адаптивного фильтра, а выходной сигнал системы служит для адаптивного фильтра образцовым сигналом. В процессе адаптации временные и частотные характеристики фильтра будут стремиться к соответствующим характеристикам исследуемой системы.

При обратной идентификации адаптивный фильтр включается последовательно с исследуемой системой (см. рис. 9.4, б). Выходной сигнал системы поступает на вход адаптивного фильтра, а входной сигнал системы является для адаптивного фильтра образцом. Таким образом, фильтр стремится компенсировать влияние системы и восстановить исходный сигнал, устранив внесенные системой искажения.

Теперь перейдем от этих обобщенных схем к рассмотрению более конкретных вариантов.

### Линейное предсказание

Согласно определению, данному в разд. "Авторегрессионная модель" главы 5 для предсказывающего фильтра, он минимизирует средний квадрат ошибки предсказания сигнала по его предыдущим отсчетам. Однако такую же задачу будет решать и рассмотренный в этой главе ранее фильтр Винера, если в качестве образца использовать текущий отсчет сигнала, а на вход фильтра подать сигнал, задержанный на один такт. Адаптивные алгоритмы в процессе работы сходятся к оптимальному винеровскому решению, поэтому для решения задачи линейного предсказания можно использовать адаптивный фильтр, включенный по схеме, показанной на рис. 9.5.

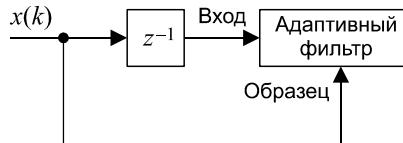


Рис. 9.5. Линейное предсказание с помощью адаптивного фильтра

В процессе адаптации коэффициенты фильтра будут стремиться к коэффициентам авторегрессионной модели (см. рис. 5.16 в главе 5), а сигнал ошибки дает оценку белого шума, возбуждающего эту модель.

## Подавление шума

Пусть нам необходимо обеспечить системой речевой связи пилота самолета или, скажем, водителя трактора. При этом воспринимаемый микрофоном речевой сигнал неизбежно окажется сильно зашумленным звуками работающего двигателя и т. п. Мы не можем избавиться от этих шумов, но можем получить *образец* шумового сигнала, установив второй микрофон в непосредственной близости от двигателя или иного источника шумов. Разумеется, этот шум нельзя просто вычесть из речевого сигнала, поскольку по дороге до двух микрофонов шум следует *разными путями* и, следовательно, претерпевает *разные искажения* (рис. 9.6). Однако шумовые случайные процессы, воспринимаемые двумя микрофонами, будут *коррелированными*, т. к. они происходят из общего источника. В то же время, очевидно, что шумовой сигнал не коррелирован с полезным речевым сигналом.

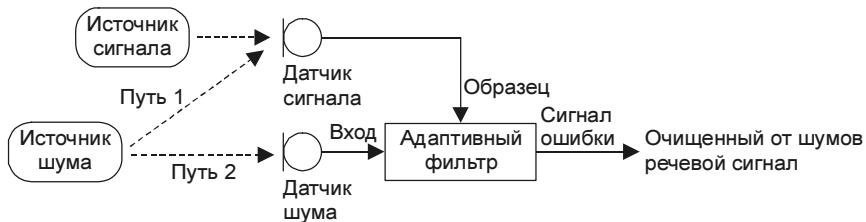


Рис. 9.6. Подавление шума с помощью адаптивного фильтра

С помощью адаптивного фильтра в данном случае решается задача прямой идентификации преобразований шума на пути к сигнальному микрофону. Входным сигналом адаптивного фильтра является шумовой сигнал от дополнительного микрофона (на рис. 9.6 он обозначен как *датчик шума*), а в качестве образцового сигнала используется сигнально-шумовая смесь, воспринимаемая основным микрофоном (на рис. 9.6 — *датчик сигнала*).

Адаптивный фильтр стремится преобразовать входной сигнал так, чтобы сделать его как можно ближе (в смысле среднеквадратической ошибки) к образцовому. Поскольку с входным сигналом фильтра коррелирована лишь шумовая составляющая образцового сигнала, после завершения процесса адаптации на выходе фильтра будет получена оценка шума, присутствующего в образцовом сигнале. Сигнал ошибки

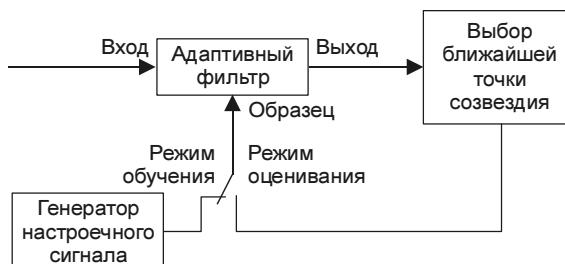
ки, рассчитываемый как разность между образцовым сигналом и выходным сигналом адаптивного фильтра, будет в этом случае представлять собой очищенный от шума речевой сигнал.

Система речевой связи не является единственным возможным объектом применения рассматриваемой системы шумоподавления. Возможны и иные области использования данной идеи.

## Выравнивание частотной характеристики канала связи

При передаче по каналу связи информационный сигнал неизбежно претерпевает некоторые искажения. В системах цифровой связи эти искажения могут привести к возникновению ошибок при приеме данных. Для устранения этих ошибок (или, во всяком случае, уменьшения их числа) необходимо компенсировать влияние канала связи, т. е. решить задачу обратной идентификации (см. рис. 9.4, б). В частотной области компенсация вносимых каналом искажений означает *выравнивание* (*equalization*) его частотной характеристики, поэтому фильтры, выполняющие такое выравнивание, получили название *эквалайзеров* (*equalizer*).

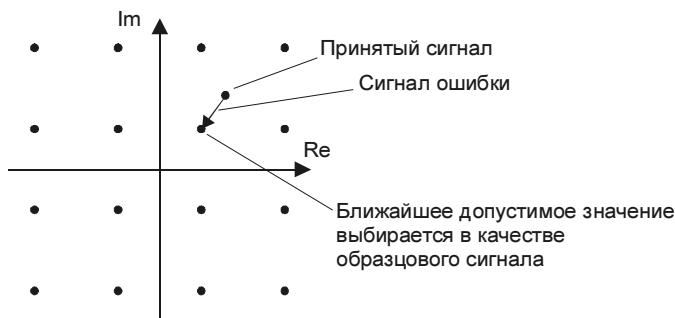
При использовании адаптивного фильтра в качестве эквалайзера возникает проблема получения образцового сигнала. Эта проблема решается путем передачи специального настроичного сигнала перед началом передачи данных. В качестве такого настроичного сигнала обычно используется псевдослучайная последовательность символов. Алгоритм формирования этого сигнала известен приемной стороне, поэтому образцовый сигнал может быть сгенерирован автономно и использован для обучения адаптивного фильтра. Этот режим работы называется *режимом обучения* (*training mode*) (рис. 9.7).



**Рис. 9.7.** Выравнивание частотной характеристики канала связи с помощью адаптивного фильтра

После окончания настроичного сигнала начинается собственно передача данных. Приемник при этом переключается в другой режим, называемый *режимом оценивания* (*decision-directed mode*). В этом режиме для получения образцового сигнала используется тот факт, что множество возможных значений сигнала в системе цифровой связи является конечным. После приема очередного отсчета (или целого фрагмента сигнала) ищется ближайшее к принятому сигналу допустимое значение.

Оно используется в качестве образцового сигнала, а разность между этим значением и принятым сигналом дает сигнал ошибки, используемый для адаптации. На рис. 9.8 это иллюстрируется применительно к 16-позиционной квадратурной модуляции.



**Рис. 9.8.** Формирование образцового сигнала и сигнала ошибки в режиме оценивания

Если после настройки эквалайзера, произведенной в режиме обучения, уровень шумов на выходе фильтра оказывается таким, что ближайшая допустимая точка в большинстве случаев оказывается правильной (т. е. если вероятность ошибки мала), адаптивный алгоритм сохраняет стабильность.

#### **ЗАМЕЧАНИЕ**

В некоторых, в частности многопользовательских, системах связи передавать настроенный сигнал не представляется возможным. В этом случае может использоваться *слепая* (blind) адаптация, рассмотрение которой выходит за рамки данной книги. Некоторые алгоритмы слепой адаптации приведены в [35].

## **Эхоподавление**

Данная технология, так же как и выравнивание канала связи, широко используется в современных модемах. Скоростные модемы для телефонных линий связи работают в *дуплексном режиме*, т. е. передают и принимают данные одновременно, при этом для передачи и приема используется одна и та же полоса частот. Однако сигнал собственного передатчика в данном случае неизбежно просачивается в приемник, мешая работе последнего. Просачивающийся сигнал может распространяться разными путями, приобретая при этом не известные заранее искажения (рис. 9.9).

Подавить эхо-сигнал можно с помощью адаптивного фильтра. При этом решается задача прямой идентификации тракта распространения эха. На вход адаптивного фильтра поступает сигнал передатчика модема, а в качестве образцового сигнала используется принимаемый сигнал, содержащий эхо (рис. 9.10). Адаптивный фильтр формирует оценку эхо-сигнала, а сигнал ошибки представляет собой принимаемый сигнал, очищенный от эха.

Для правильной работы системы эхоподавления необходимо, чтобы передаваемый и принимаемый сигналы были некоррелированы. Для обеспечения некоррелиро-

ванными входные данные, поступающие в модем для передачи, прежде всего подвергаются *скремблированию (scrambling)*, т. е. преобразуются в псевдослучайный битовый поток. При этом два взаимодействующих модема используют *разные* скремблеры, что и обеспечивает некоррелированность передаваемых ими сигналов. Эхоподавление, осуществляющее согласно схеме рис. 9.10, используется во всех современных модемах.

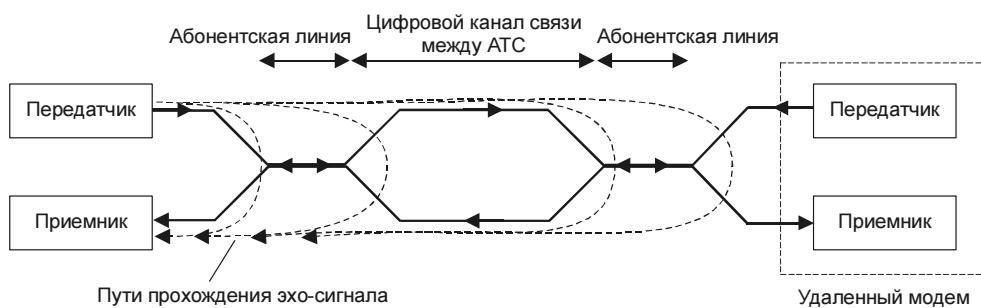


Рис. 9.9. Формирование эхо-сигнала

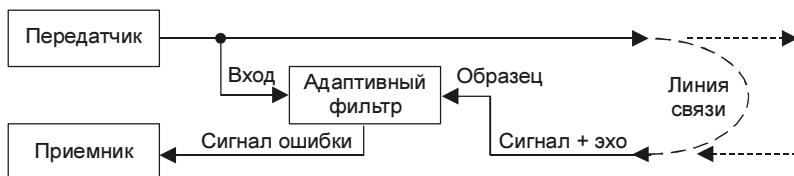


Рис. 9.10. Эхоподавление, осуществляющее с помощью адаптивного фильтра

## Объекты адаптивной фильтрации пакета Filter Design

В версии 4.7 пакета Filter Design (R2010a) адаптивные фильтры реализованы *только* в виде объектов MATLAB, а существовавшие ранее функции адаптивной фильтрации не только объявлены устаревшими, но и удалены.

Общие сведения об использовании объектов в MATLAB были приведены в главе 4. Для создания объекта адаптивного фильтра служит функция `adaptfilt`. При ее вызове необходимо указать требуемый алгоритм адаптивной фильтрации — для этого после имени функции ставится точка и указывается соответствующий идентификатор метода (конструктора):

```
ha = adaptfilt.algorithm(...)
```

В настоящее время в пакете Filter Design реализовано около тридцати разновидностей алгоритмов адаптивной фильтрации. К сожалению, автору неизвестен источник на русском языке, который содержал бы обзор этих алгоритмов. При наличии доступа к зарубежным книгам и журналам можно рекомендовать статьи, ссылки на

которые имеются в справочной системе MATLAB, а также литературу, упомянутую в начале этой главы.

Полный список реализованных алгоритмов адаптивной фильтрации и имена соответствующих конструкторов объектов `adaptfilt` приведены в табл. 9.1.

**Таблица 9.1.** Свойства объектов адаптивных фильтров

Конструктор	Алгоритм
	<b>Семейство алгоритмов LMS</b>
<code>adaptfilt.adjlms</code>	Сопряженный (adjoint) LMS-алгоритм
<code>adaptfilt.blms</code>	Блоковый LMS-алгоритм
<code>adaptfilt.blmsfft</code>	Блоковый LMS-алгоритм, реализованный в частотной области с применением БПФ
<code>adaptfilt.dlms</code>	LMS-алгоритм с задержкой обновления коэффициентов фильтра
<code>adaptfilt.filtxlms</code>	LMS-алгоритм с дополнительной фильтрацией выходного сигнала фильтра перед формированием сигнала ошибки (filtered-x LMS)
<code>adaptfilt.lms</code>	"Классический" LMS-алгоритм (см. формулу (9.14))
<code>adaptfilt.nlms</code>	Нормированный LMS-алгоритм (расчет шага $\mu$ на каждом такте осуществляется автоматически по формуле (9.16) исходя из энергии фрагмента сигнала, содержащегося в линии задержки фильтра)
<code>adaptfilt.sd</code>	LMS-алгоритм с использованием для адаптации только знака данных, содержащихся в линии задержки фильтра (sign-data; вместо $x(k)$ в формуле (9.14) используется $\text{sign}(x(k))$ )
<code>adaptfilt.se</code>	LMS-алгоритм с использованием для адаптации только знака ошибки (sign-error; вместо $e(k)$ в формуле (9.14) используется $\text{sign}(e(k))$ )
<code>adaptfilt.ss</code>	LMS-алгоритм с использованием для адаптации только знаков сигнала ошибки и данных, содержащихся в линии задержки фильтра (sign-sign; комбинация двух предыдущих вариантов)
	<b>Семейство алгоритмов RLS</b>
<code>adaptfilt.ftf</code>	Быстрый RLS-алгоритм
<code>adaptfilt.hrls</code>	RLS-алгоритм с использованием преобразования Хаусхолдера
<code>adaptfilt.hswrls</code>	RLS-алгоритм со скользящим окном и использованием преобразования Хаусхолдера
<code>adaptfilt.qrdrls</code>	RLS-алгоритм с использованием QR-разложения матрицы
<code>adaptfilt.rls</code>	"Классический" RLS-алгоритм, в том числе с экспоненциальным забыванием данных
<code>adaptfilt.swftf</code>	Быстрый RLS-алгоритм со скользящим окном
<code>adaptfilt.swrls</code>	RLS-алгоритм со скользящим окном

Таблица 9.1 (окончание)

Конструктор	Алгоритм
<b>Алгоритмы, основанные на методе аффинных проекций</b>	
adaptfilt.ap	Метод аффинных проекций с прямым обращением матрицы
adaptfilt.apru	Метод аффинных проекций с рекурсивным обновлением обратной матрицы
adaptfilt.bap	Блоковый алгоритм аффинных проекций
<b>Алгоритмы адаптации, работающие в частотной области</b>	
adaptfilt.fdaf	LMS-алгоритм в частотной области с индивидуальной нормировкой размера шага в каждом частотном канале (frequency domain adaptive filter, FDAF)
adaptfilt.pbfdaf	Нормированный LMS-алгоритм в частотной области, использующий деление сигнала на блоки малого размера во временной области (partitioned block frequency domain adaptive filter, PBFDAF)
adaptfilt.pbufdaf	Нормированный LMS-алгоритм в частотной области, использующий деление сигнала на блоки малого размера во временной области, без ограничения длины эквивалентной импульсной характеристики (partitioned block unconstrained frequency domain adaptive filter, PBUFDASF)
adaptfilt.tdafdct	Адаптивный фильтр, использующий LMS-алгоритм после предварительного применения к входному сигналу дискретного косинусного преобразования
adaptfilt.tdafdft	Адаптивный фильтр, использующий LMS-алгоритм после предварительного применения к входному сигналу дискретного преобразования Фурье
adaptfilt.ufdaf	Нормированный LMS-алгоритм в частотной области без ограничения длины эквивалентной импульсной характеристики (unconstrained frequency domain adaptive filter, UFDAF)
<b>Адаптивные алгоритмы для решетчатых фильтров</b>	
adaptfilt.gal	Алгоритм адаптации для решетчатого фильтра, основанный на градиентном спуске (gradient-adaptive lattice, GAL)
adaptfilt.lsl	Алгоритм адаптации для решетчатого фильтра, основанный на рекурсивном методе наименьших квадратов (least squares lattice, LSL)
adaptfilt.qrdlsl	Вариант алгоритма LSL, использующий QR-разложение матрицы

Набор входных параметров конструктора зависит от реализуемого алгоритма. Рассмотрим более подробно использование объектов адаптивных фильтров для тех алгоритмов, которые были ранее описаны в этой главе. Вызов конструкторов для них имеет следующий вид:

```

ha = adaptfilt.lms(L, step, leakage, coeffs, states)
ha = adaptfilt.nlms(L, step, leakage, offset, coeffs, states)
ha = adaptfilt.sd(L, step, leakage, coeffs, states)
ha = adaptfilt.se(L, step, leakage, coeffs, states)
ha = adaptfilt.ss(L, step, leakage, coeffs, states)
ha = adaptfilt.rls(L, lambda, invcov, coeffs, states)

```

Назначение входных параметров расшифровано в табл. 9.2. Там же приведены и значения по умолчанию, которые имеются для всех параметров конструкторов.

**Таблица 9.2.** Параметры конструкторов рассматриваемых адаптивных фильтров

Параметр	Назначение	Имеет смысл для
L	Длина фильтра (напомним, что это значение на единицу превышает порядок фильтра). По умолчанию $L = 10$	всех алгоритмов
coeffs	Вектор (строка длиной L) начальных значений коэффициентов фильтра. По умолчанию этот вектор заполняется нулями	всех алгоритмов
states	Вектор (столбец высотой L-1) начального содержимого линии задержки фильтра. По умолчанию этот вектор заполняется нулями	всех алгоритмов
step	Размер шага $\mu$ . По умолчанию $step = 0.1$ (в случае нормированного LMS $step = 1$ )	всех разновидностей LMS
leakage	Коэффициент утечки (в диапазоне 0...1). По умолчанию $leakage = 1$	всех разновидностей LMS
offset	Константа $\epsilon$ (см. формулу (9.16)). По умолчанию $offset = 0$	нормированного LMS
lambda	Коэффициент экспоненциального забывания $\lambda$ (в диапазоне 0...1). По умолчанию $lambda = 1$	RLS
invcov	Начальное значение оценки обратной корреляционной матрицы входного сигнала. По умолчанию используется матрица, главная диагональ которой заполнена значением, равным 1000: $invcov = 1000 * eye(L)$	RLS

### ЗАМЕЧАНИЕ

В табл. 9.2 и далее в табл. 9.3 упоминается не рассматривавшийся в теоретической части главы параметр LMS-алгоритма — *коэффициент утечки (leakage)*. Это коэффициент, лежащий в диапазоне от 0 до 1, на который может умножаться старый вектор коэффициентов  $w(k)$  в формуле (9.14) для улучшения стабильности алгоритма.

Список свойств объектов адаптивных фильтров для рассматриваемых алгоритмов приведен в табл. 9.3. Считывать и задавать значения этих свойств можно с помощью функций *get* и *set* соответственно (см. разд. "Общая информация об объектах MATLAB" главы 4).

Таблица 9.3. Свойства объектов адаптивных фильтров

Имя свойства	Описание
Algorithm	Строка, содержащая название используемого алгоритма адаптации (только для чтения)
FilterLength	Длина вектора коэффициентов фильтра (напомним, что это значение на единицу превышает порядок фильтра)
Coefficients	Вектор коэффициентов фильтра
States	Содержимое линии задержки фильтра
PersistentMemory	Данное свойство может принимать логические значения <code>true</code> (включено) и <code>false</code> (выключено). При включении данного режима (принято по умолчанию) перед каждой операцией фильтрации (метод <code>filter</code> ) производится сброс фильтра в исходное состояние. При выключении данного режима текущее состояние фильтра сохраняется, что позволяет организовать блоковую обработку сигнала
NumSamplesProcessed	Число обработанных отсчетов сигнала (только для чтения)
StepSize (для всех вариантов алгоритма LMS)	Размер шага $\mu$
Leakage (для всех вариантов алгоритма LMS)	Коэффициент утечки (в диапазоне 0...1, по умолчанию равен 1)
Offset (для алгоритма nlms)	Константа $\epsilon$ (см. формулу (9.16))
ForgettingFactor (для алгоритма RLS)	Коэффициент забывания $\lambda$ (в диапазоне 0...1, по умолчанию 1)
KalmanGain (для алгоритма RLS)	Вектор коэффициентов усиления $\mathbf{K}$ (см. формулу (9.28); только для чтения)
InvCov (для алгоритма RLS)	Оценка обратной корреляционной матрицы $\mathbf{P}$

Обработка сигнала адаптивным фильтром осуществляется с помощью функции `filter`:

```
[y, e] = filter(ha, x, d)
```

Здесь `ha` — объект адаптивного фильтра, `x` — вектор отсчетов входного сигнала, `d` — вектор отсчетов образцового сигнала. Результатами работы функции являются векторы отсчетов выходного сигнала `y` и сигнала ошибки фильтрации `e`.

Как показывает синтаксис вызова функции фильтрации, образцовый сигнал должен быть известен заранее, что делает невозможным применение данных функций в режиме адаптации по оценке сигнала (*decision-directed mode*). Кроме того, реализация данной функции подразумевает на каждом временном такте сдвиг данных в линии задержки фильтра на один отсчет. Это затрудняет применение объектов в тех ситуациях, когда данные обновляются иным образом (это имеет место, например, в так называемых *дробных* (*fractionally-spaced equalizer*) эквалайзерах).

Требуемое обновление можно реализовать путем осуществления поотсчетной обработки с ручной коррекцией состояния линии задержки, но такой код будет громоздким, медленным и надуманным.

### **ЗАМЕЧАНИЕ**

Возможность адаптации по оценке сигнала реализована в пакете расширения Communications Toolbox, где для этого имеются специальные объекты эквалайзеров.

С объектами адаптивных фильтров можно использовать многие методы, описанные в главе 4 применительно к объектам класса `dfilt`. Поскольку адаптивный фильтр является системой с переменными параметрами, эти методы дают результаты, соответствующие *текущему состоянию* фильтра.

Аналогичным образом объекты адаптивных фильтров поддерживаются и визуализатором фильтров FVTool:

```
fvtool(ha)
```

К методам, специфическим именно для адаптивных фильтров, относятся три функции, перечисленные далее:

- `[mumax, mumaxmse] = maxstep(ha, x)` — данная функция производит оценку максимально допустимой величины размера шага  $\mu$  для объекта адаптивного фильтра `ha` (функция применима для следующих вариантов алгоритма LMS: `lms`, `nlms`, `blms`, `blmsfft`, `se`). Входной параметр `x` — пример возможного входного сигнала фильтра (для алгоритма `nlms` задавать этот параметр не нужно). Этот параметр может быть матрицей, тогда ее столбцы рассматриваются как различные реализации входного случайного сигнала. Выходной параметр `mumax` — предельная величина  $\mu$ , при которой обеспечивается сходимость алгоритма в *среднем* (см. формулу (9.12)), выходной параметр `mumaxmse` — предельная величина  $\mu$ , при которой обеспечивается сходимость алгоритма в *среднем квадрате* (см. формулу (9.15));
- `[mmse, emse, meanw, mse, tracek] = msepred(ha, x, d, m)` — данная функция производит оценку среднего квадрата ошибки фильтрации для объекта адаптивного фильтра `ha` (функция применима для следующих вариантов алгоритма LMS: `lms`, `nlms`, `blms`, `blmsfft`, `se`). Входные параметры `x` и `d` — векторы отсчетов входного и образцового сигнала соответственно. Выходные параметры: `mmse` — средний квадрат ошибки фильтрации в установленном режиме, `emse` — избыточная ошибка фильтрации (см. формулу (9.17)), `meanw` — матрица предсказанных математических ожиданий коэффициентов фильтра для всех моментов времени, `mse` — вектор зависимости среднего квадрата ошибки от времени, `tracek` — вектор зависимости от времени для суммарного квадратичного отклонения коэффициентов фильтра от оптимальных. При расчете трех последних выходных параметров может производиться прореживание по времени с коэффициентом `m` (по умолчанию `m = 1`);
- `[mse, meanw, w, tracek] = msesim(ha, x, d, m)` — данная функция по назначению аналогична предыдущей, но измерение среднего квадрата ошибки производится путем *моделирования*, без использования теоретических оценок. Поэтому

функция применима к любым объектам адаптивных фильтров, работающих во временной области, независимо от используемых ими алгоритмов адаптации. Производимое моделирование предполагает усреднение по ансамблю реализаций, поэтому входные параметры  $x$  и  $d$  должны быть матрицами, столбцы которых содержат отдельные реализации входного и образцового сигналов соответственно. Выходной параметр  $w$  представляет собой вектор финальных значений коэффициентов фильтра. Остальные входные и выходные параметры совпадают по назначению с одноименными параметрами функции `msepred`.

## Примеры реализации адаптивной фильтрации

Приведем несколько примеров, показывающих, как различные задачи, рассмотренные ранее в разд. "Применение адаптивных фильтров" этой главы, могут быть смоделированы с использованием объектов адаптивных фильтров из пакета Filter Design.

### Идентификация системы

В данном примере мы реализуем решение задачи прямой идентификации системы согласно рис. 9.4, а. Входным сигналом системы будет служить дискретный белый гауссов шум, а сама система будет представлять собой нерекурсивный фильтр 31-го порядка с импульсной характеристикой в виде экспоненциально затухающей синусоиды, на период колебаний которой приходится 4 отсчета. Кроме того, после обработки половины входного сигнала параметры системы скачкообразно изменятся — у ее импульсной характеристики поменяется знак. Это позволит нам посмотреть на то, как реагируют алгоритмы LMS и RLS на резкое изменение статистических свойств обрабатываемого сигнала. Итак, формируем входной и выходной сигналы идентифицируемой системы:

```
>> x = randn(2000, 1); % дискретный белый гауссов шум
>> t = 0:31;
>> b = exp(-t/5) .* cos(t*pi/2); % импульсная характеристика системы
>> % генерируем первую половину выходного сигнала
>> [y(1:1000), state] = filter(b, 1, x(1:1000));
>> % генерируем вторую половину выходного сигнала
>> y(1001:2000) = filter(-b, 1, x(1001:2000), state);
```

Далее создадим объекты LMS- и RLS-адаптивных фильтров с помощью соответствующих конструкторов. Для LMS-алгоритма предварительно рассчитаем значение коэффициента  $\mu$ , выбрав его в два раза меньшим предельного значения, определяемого формулой (9.15), для всех остальных параметров используем значения по умолчанию:

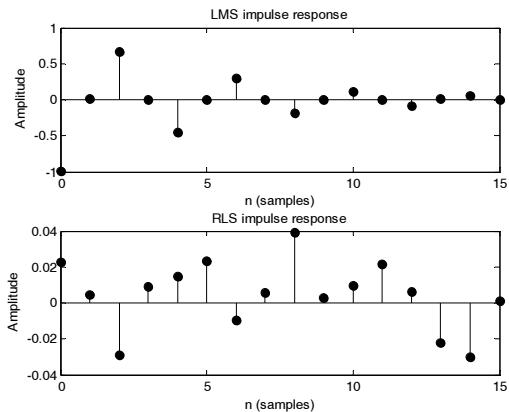
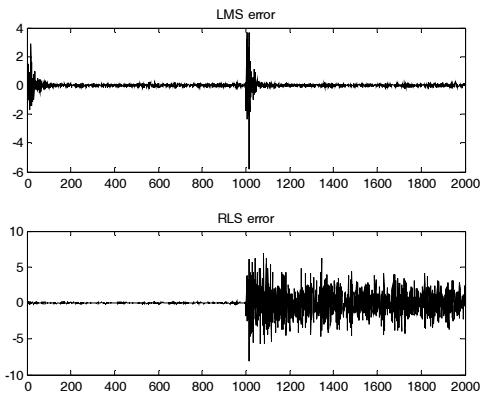
```
>> N = 16; % длина фильтров
>> mu = 1/N/var(y); % расчет размера шага для алгоритма LMS
>> ha_lms = adaptfilt.lms(N, mu); % создание объекта LMS-фильтра
>> ha_rls = adaptfilt.rls(N); % создание объекта RLS-фильтра
```

Теперь реализуем собственно фильтрацию, воспользовавшись функцией `filter`. В соответствии с рис. 9.4, *a* входной сигнал адаптивного фильтра совпадает с входным сигналом исследуемой системы ( $x$ ), а образцовый сигнал — это выходной сигнал системы ( $y$ ):

```
>> [y_lms, e_lms] = filter(ha_lms, x, y); % фильтрация LMS-фильтром
>> [y_rls, e_rls] = filter(ha_rls, x, y); % фильтрация RLS-фильтром
```

Построим для обоих фильтров графики зависимости сигнала ошибки от времени, а также выведем на экран их импульсные характеристики, полученные на момент завершения обработки сигнала (рис. 9.11):

```
>> subplot(2,2,1)
>> plot(e_lms)
>> title('LMS error')
>> subplot(2,2,2)
>> impz(ha_lms.coefficients)
>> title('LMS impulse response')
>> subplot(2,2,3)
>> plot(e_rls)
>> title('RLS error')
>> subplot(2,2,4)
>> impz(ha_rls.coefficients)
>> title('RLS impulse response')
```



**Рис. 9.11.** Результаты идентификации системы с помощью адаптивных алгоритмов LMS (сверху) и RLS (снизу): слева — зависимость ошибки от времени, справа — итоговые импульсные характеристики фильтров

Полученные результаты наглядно демонстрируют основные особенности LMS- и RLS-алгоритмов. На начальном этапе RLS-алгоритм показал отсутствие сколько-нибудь заметного переходного процесса и меньшую (по сравнению с LMS-алгоритмом) ошибку в установившемся режиме. Однако после скачкообразного изменения характеристик системы RLS-алгоритм так и не смог приспособиться

к новым условиям, тогда как LMS-фильтр после некоторого переходного процесса свел ошибку фильтрации к тому же уровню, который она имела раньше.

Сказанному соответствуют и итоговые импульсные характеристики фильтров (см. рис. 9.11, справа) — характеристика LMS-фильтра хорошо совпадает с экспоненциально затухающими колебаниями импульсной характеристики анализируемой системы, а характеристика RLS-фильтра выглядит абсолютно случайным набором значений.

### **ЗАМЕЧАНИЕ**

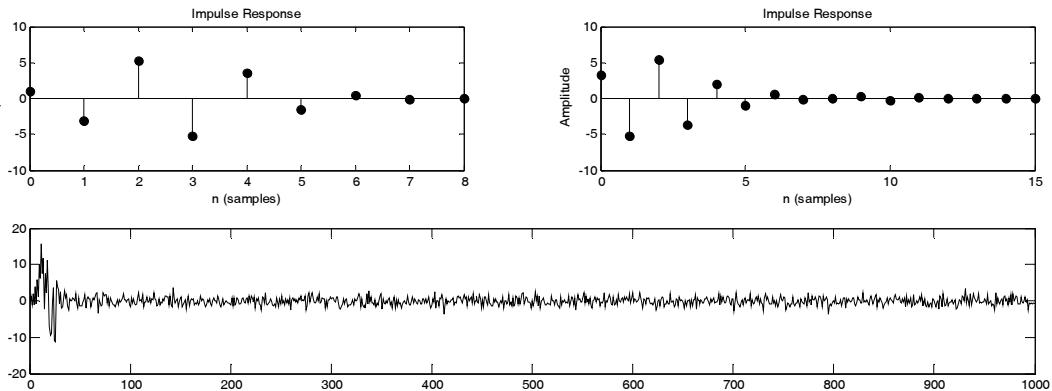
В данном примере не использовалось экспоненциальное забывание, дающее RLS-алгоритму возможность отслеживать изменения статистических свойств обрабатываемого сигнала. Чтобы использовать забывание, полю `ForgettingFactor` объекта RLS-фильтра в данном примере необходимо присвоить значение, меньшее единицы.

## **Линейное предсказание**

Реализуем с помощью RLS-фильтра линейное предсказание сигнала, сформированного с помощью авторегрессионной модели. В качестве коэффициентов модели используем знаменатель фильтра Баттервортса 8-го порядка с частотой среза, равной 0,3 от частоты Найквиста:

```
>> [b, a] = butter(8, 0.3); % фильтр Баттервортса
>> b = 1; % оставляем только знаменатель
>> x = randn(1000, 1); % входной сигнал AR-модели
>> y = filter(b, a, x); % формирование сигнала
>> ha = adaptfilt.rls(16); % создание RLS-фильтра
>> % линейное предсказание с помощью адаптивного фильтра
>> [y1, e] = filter(ha, y(1:end-1), y(2:end));
>> subplot(2, 2, 1)
>> impz(a) % коэффициенты AR-модели
>> subplot(2, 2, 2)
>> impz(ha.coefficients) % имп. х-ка адаптивного фильтра
>> subplot(2, 1, 2)
>> plot(e) % ошибка фильтрации
```

Результаты работы программы показаны на рис. 9.12. Видно, что коэффициенты адаптивного фильтра совпадают с коэффициентами использованной авторегрессионной модели (при сравнении верхних графиков на рис. 9.12 следует учесть, что первый элемент вектора *a*, равный единице, не входит в число коэффициентов модели; кроме того, коэффициенты модели и коэффициенты предсказывающего фильтра имеют разные знаки (см. рис. 5.13 в главе 5)). На нижнем графике рис. 9.12, демонстрирующем поведение ошибки фильтрации во времени, хорошо видно, что после начального переходного процесса устанавливается стационарный режим работы. Согласно идеи линейного предсказания (см. разд. "Авторегрессионная модель" главы 5), сигнал ошибки фильтрации при этом является оценкой белого шума, возбуждающего авторегрессионную модель.



**Рис. 9.12.** Линейное предсказание с помощью адаптивного фильтра:  
сверху — коэффициенты авторегрессионной модели (слева) и итоговая импульсная  
характеристика адаптивного фильтра (справа),  
снизу — зависимость ошибки фильтрации от времени

## Шумоподавление

Для демонстрации шумоподавления используем входящий в состав поставки MATLAB файл `mtlb.mat`, в котором содержатся запись произнесенного слова "MATLAB" (переменная `mtlb`) и частота дискретизации данного сигнала (переменная `Fs`). Речевой сигнал является очень коротким, поэтому повторим его 10 раз с помощью функции `repmat` (график одного периода исходного речевого сигнала показан на рис. 9.13, сверху):

```
>> load mtlb % загрузка данных
>> s = repmat(mtlb, 10, 1); % 10-кратное повторение сигнала
>> subplot(3, 1, 1)
>> plot(s(1:length(mtlb))) % график сигнала
```

Можно также прослушать исходный звук с помощью функции `soundsc`:

```
>> soundsc(s, Fs)
```

Теперь сформируем дискретный белый шум и пропустим его через два фильтра, чтобы получить коррелированные случайные сигналы. В данном эксперименте мы используем нерекурсивные фильтры первого порядка, один из которых будет вычислять сумму, а другой — разность пары соседних отсчетов. Эти фильтры являются простейшими ФНЧ и ФВЧ соответственно:

```
>> noise = randn(length(s), 1); % дискретный белый шум
>> noisel = filter([1 1], 1, noise); % шум, пропущенный через ФНЧ
>> noise2 = filter([1 -1], 1, noise); % шум, пропущенный через ФВЧ
```

Шум `noisel` мы добавим к речевому сигналу. На графике (см. рис. 9.13, в центре) видно, что полезный сигнал визуально не выделяется над шумом:

```
>> sn = s + noisel; % добавляем шум к сигналу
>> subplot(3, 1, 2)
>> plot(sn(1:length(mtlb))) % график зашумленного сигнала
```

Воспроизведение зашумленного звука показывает, что и на слух полезный сигнал разобрать нельзя:

```
>> soundsc(sn, Fs)
```

Теперь создадим объект адаптивного фильтра длины 16, использующего алгоритм RLS, и в соответствии с рис. 9.6 подадим на вход фильтра шумовой сигнал noise2, использовав в качестве образцового зашумленный полезный сигнал sn:

```
>> ha1 = adaptfilt.rls(16);
>> [y, e] = filter(ha1, noise2, sn);
```

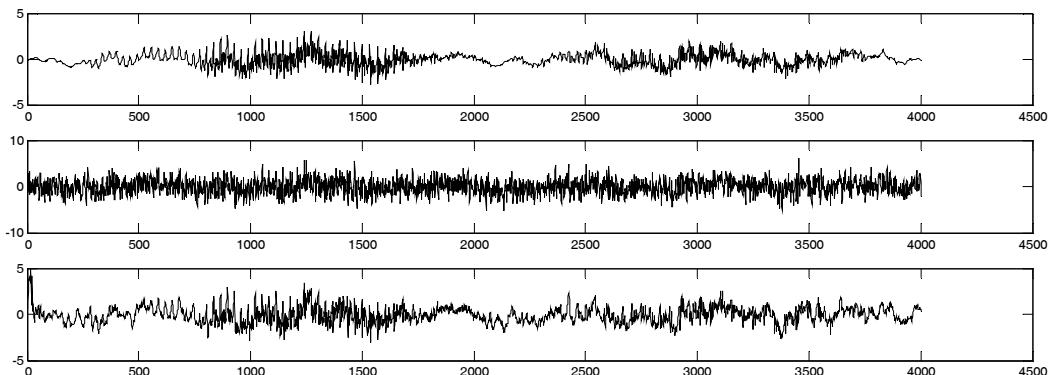
Полученный сигнал ошибки e представляет собой очищенный от шума полезный сигнал (график одного его периода показан на рис. 9.13, снизу):

```
>> subplot(3, 1, 3)
>> plot(e(1:length(mtlb)))
>> ylim([-5 5])
```

На рисунке хорошо видно, что сигнал ошибки визуально весьма близок к полезному сигналу, показанному на верхнем графике. Воспроизведем получившийся звук:

```
>> soundsc(e, Fs)
```

Прослушивание показывает, что остаточный шум все равно присутствует, но слова разбираются без труда.



**Рис. 9.13.** Исходный сигнал (сверху), зашумленный сигнал (в центре) и сигнал, очищенный от шума с помощью адаптивного фильтра (снизу)

Завершая рассмотрение данного примера, следует отметить, что сигнал был более чем в четыре раза (более чем на 6 дБ) слабее по мощности, чем шум:

```
>> 10*log10(var(s)/var(noise1)) % отношение С/Ш в децибелах
ans =
-6.6026
```

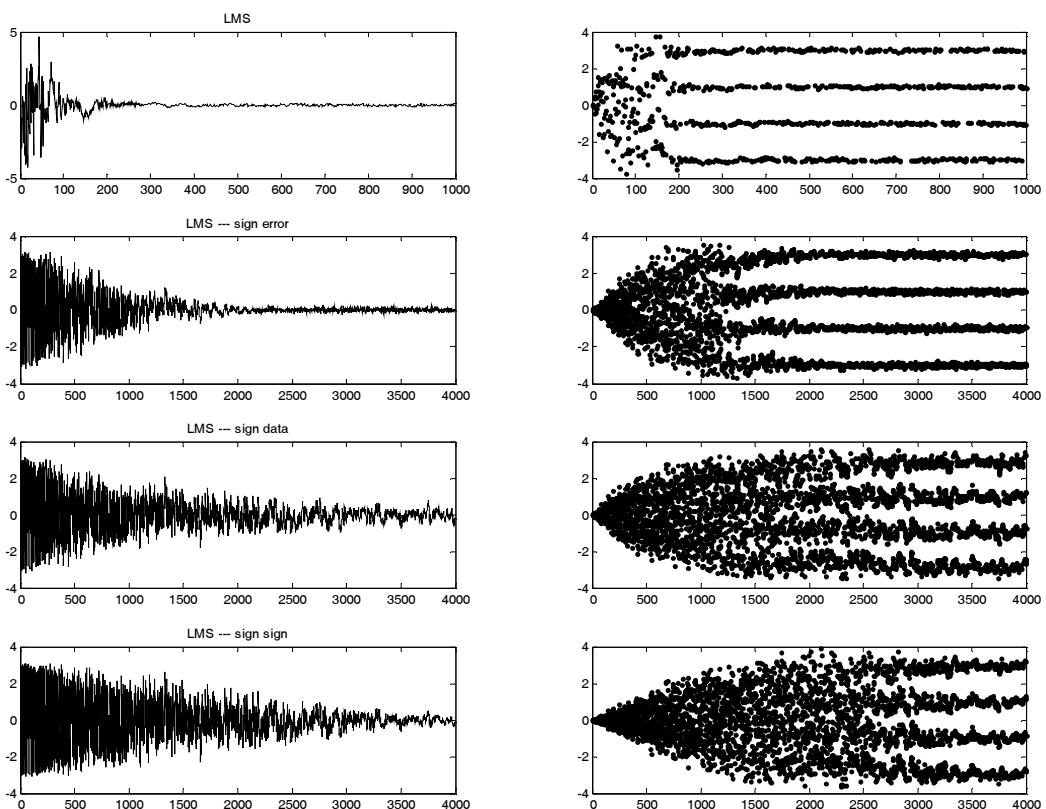
## Компенсация искажений, вносимых каналом связи

В качестве примера реализуем адаптивную коррекцию искажений, вносимых в сигнал тем же каналом связи, что использовался в примере расчета фильтра Винера. При этом сравним качество работы четырех версий алгоритма LMS — классического варианта и трех вариантов с использованием знаковых преобразований. Чтобы более наглядно показать результат компенсации, в качестве входного используем четырехуровневый цифровой сигнал, равновероятно и независимо принимающий значения  $-3, -1, 1$  и  $3$ . Код соответствующей MATLAB-программы приведен далее.

```
h = [-2 -4 6 3]; % импульсная характеристика канала связи
% генерация сигнала
NX = 4000; % число отсчетов
levels = [-3 -1 1 3]; % возможные уровни исходного сигнала
x = randsrc(NX, 1, levels); % случайный сигнал
y = conv(x, h); % искаженный сигнал
N = 32; % длина адаптивного фильтра
mu = 1/var(levels, 1)/sum(h.^2)/N; % размер шага для LMS
% создание объектов
ha_lms = adaptfilt.lms(N, mu);
ha_lms_se = adaptfilt.se(N, mu/10);
ha_lms_sd = adaptfilt.sd(N, mu);
ha_lms_ss = adaptfilt.ss(N, mu);
y = y(round(N/2)+1:end); % задержка фильтрации
[x, y] = eqtflength(x, y); % выравнивание длин векторов x и y
% адаптивная фильтрация
[x1_lms, e_lms] = filter(ha_lms, y(1:1000), x(1:1000));
[x1_lms_se, e_lms_se] = filter(ha_lms_se, y, x);
[x1_lms_sd, e_lms_sd] = filter(ha_lms_sd, y, x);
[x1_lms_ss, e_lms_ss] = filter(ha_lms_ss, y, x);
% вывод графиков
subplot(4,2,1), plot(e_lms), title('LMS')
subplot(4,2,2), plot(x1_lms,'.')
subplot(4,2,3), plot(e_lms_se), title('LMS --- sign error')
subplot(4,2,4), plot(x1_lms_se,'.')
subplot(4,2,5), plot(e_lms_sd), title('LMS --- sign data')
subplot(4,2,6), plot(x1_lms_sd,'.')
subplot(4,2,7), plot(e_lms_ss), title('LMS --- sign sign')
subplot(4,2,8), plot(x1_lms_ss,'.)
```

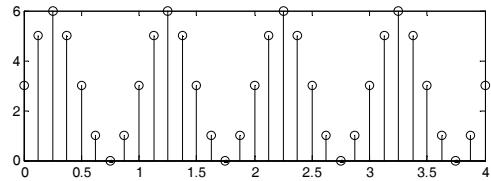
На рис. 9.14 показаны результаты работы программы. В левом столбце — зависимость сигнала ошибки от номера шага, в правом — график выходного сигнала

фильтра. Видно, что использование знаковых преобразований замедляет сходимость LMS-алгоритма (пришлось даже увеличить число отображаемых итераций с 1000 до 4000). В наименьшей степени сходимость замедляется при знаковом преобразовании сигнала ошибки (*sign error*), в наибольшей — при одновременном знаковом преобразовании сигнала ошибки и содержимого линии задержки фильтра (*sign sign*). Вариант со знаковым преобразованием только содержимого линии задержки (*sign data*) занимает промежуточное положение. Кроме того, предельное значение  $\mu$  для знаковых вариантов сильно зависит от масштаба уровней образцового и входного сигналов. Так, в данном примере пришлось в 10 раз уменьшить значение  $\mu$  для варианта со знаковым преобразованием сигнала ошибки.



**Рис. 9.14.** Компенсация искажений, вносимых каналом связи, с помощью различных вариантов LMS-алгоритма: слева — ошибка фильтрации, справа — выходной сигнал

# ГЛАВА 10



## Многоскоростная обработка сигналов

Термин "многоскоростная (multirate) обработка" означает, что частота дискретизации сигнала в различных точках системы является разной. При этом изменение частоты дискретизации может быть как основной задачей, решаемой системой, так и вспомогательным приемом.

В данной главе рассмотрены некоторые примеры многоскоростных систем, предназначенные для изменения частоты дискретизации сигнала и реализации его многоканальной фильтрации.

Экономная в вычислительном плане реализация многоскоростных систем может быть осуществлена с помощью так называемых *полифазных структур*, которые также рассмотрены в данной главе.

### Изменение частоты дискретизации

При решении различных задач обработки сигналов приходится увеличивать или уменьшать частоту дискретизации сигналов. Это необходимо, например, для согласования различных стандартов хранения и передачи дискретной информации. Классическим примером является преобразование аудиозаписей между форматами компакт-дисков (частота дискретизации 44,1 кГц) и цифровой магнитной записи R-DAT (частота дискретизации 48 кГц).

Приведенный пример не относится к самым простым, поскольку коэффициент изменения частоты дискретизации не является целым числом. В зависимости от значения этого коэффициента выделяют следующие варианты:

- *интерполяция (interpolation)* — повышение частоты дискретизации в целое число раз;
- *прореживание (decimation)* — понижение частоты дискретизации в целое число раз;
- *передискретизация (resampling)* — изменение частоты дискретизации в произвольное (в общем случае дробное) число раз.

Рассмотрим алгоритмы реализации этих вариантов изменения частоты дискретизации более подробно.

## Прореживание

Казалось бы, ломать — не строить, и для понижения частоты дискретизации в  $N$  раз достаточно взять из исходной последовательности каждый  $N$ -й отсчет. Данная операция называется *понижением частоты дискретизации (downsampling)*, а выполняющий ее блок на структурных схемах обозначается символом, показанным на рис. 10.1. Однако одного только блока понижения частоты дискретизации оказывается недостаточно — для предупреждения возможных побочных эффектов следует принять некоторые меры.

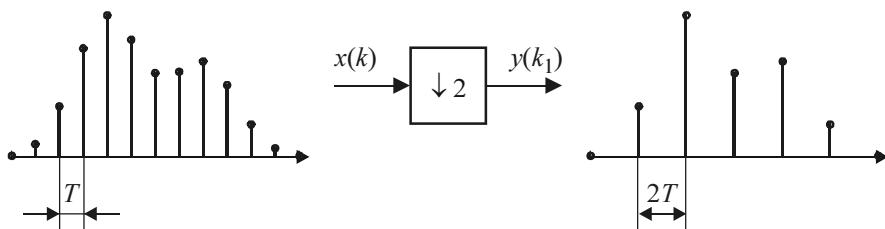


Рис. 10.1. Понижение частоты дискретизации в два раза

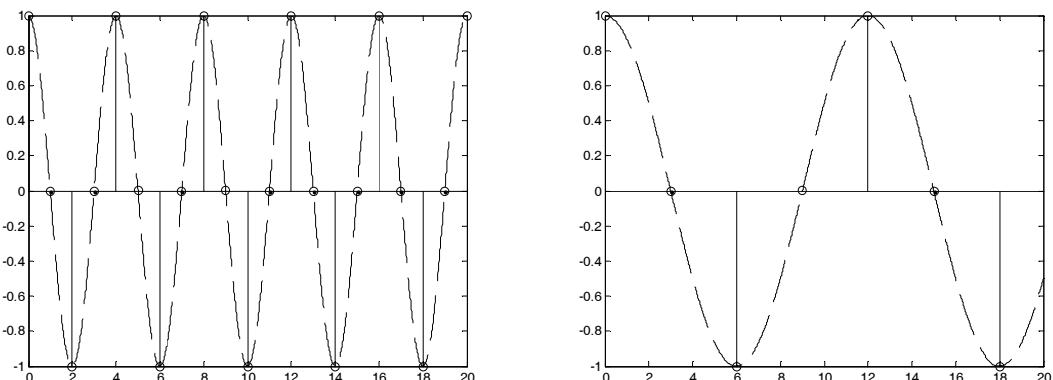


Рис. 10.2. Появление ложных частот при прореживании дискретного сигнала

Дело в том, что процесс прореживания по сути сводится к *дискретизации дискретного сигнала*. При этом имеют место все эффекты, упомянутые в разд. "Спектр дискретного сигнала" главы 3, в том числе и появление *ложных частот (aliasing, рис. 10.2)*:

```
>> k = 0:20; % дискретное время
>> t = 0:0.01:20; % аналоговое время
>> s1 = cos(2*pi*t/4); % аналоговый сигнал
>> sd1 = cos(2*pi*k/4); % дискретный сигнал
>> Dec = 3; % коэффициент прореживания
>> sd2 = sd1(1:Dec:end); % прореженный сигнал
>> s2 = cos(2*pi*t/12); % ложный сигнал
```

```
>> subplot(1, 2, 1)
>> stem(k, sd1); hold on; plot(t, s1, '--'); hold off
>> subplot(1, 2, 2)
>> stem(k(1:Dec:end), sd2); hold on; plot(t, s2, '--'); hold off
```

Таким образом, если в спектре исходного (прореживаемого) сигнала содержатся частоты, превышающие половину новой частоты дискретизации (т. е. новую частоту Найквиста), это приведет к появлению в спектре выходного сигнала ложных частотных составляющих.

Для устранения этого нежелательного эффекта следует, как и при дискретизации аналогового сигнала, предварительно пропустить сигнал через ФНЧ с частотой среды, равной новой частоте Найквиста.

В результате полная структура устройства, выполняющего прореживание, включает в себя каскадно соединенные ФНЧ и блок понижения частоты дискретизации (рис. 10.3).

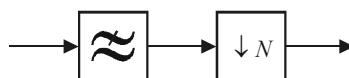


Рис. 10.3. Реализация прореживания сигнала

Чтобы сохранить фазовые соотношения во входном сигнале, следует использовать нерекурсивный фильтр с линейной ФЧХ. Использование нерекурсивного фильтра позволяет весьма эффективно организовать вычисления — поскольку нас интересует только каждый  $N$ -й отсчет выходного сигнала, остальные отсчеты можно и не вычислять. При этом можно считать, что сигнал "заталкивается" в линию задержки фильтра порциями по  $N$  отсчетов. Если использовать рекурсивный фильтр, такой экономии добиться не удастся — из-за наличия обратных связей придется вычислять *весь* выходной сигнал фильтра и только потом отбрасывать лишние отсчеты.

## Интерполяция

При интерполяции нам необходимо повысить частоту дискретизации в  $N$  раз, т. е. "растянуть" входной сигнал, а образовавшиеся промежутки между отсчетами чем-то заполнить.

Подобно тому как прореживание сводится к дискретизации дискретного сигнала, процесс интерполяции оказывается подобным процессу восстановления непрерывного сигнала, только происходящему в дискретной области.

Исходный сигнал имеет периодический спектр, повторяющийся с частотой дискретизации  $f_d$  (рис. 10.4, *a*). Прежде всего мы "растягиваем" этот сигнал, добавляя между его отсчетами по  $N - 1$  нулей. Эта операция называется *повышением частоты дискретизации (upsampling)*, а выполняющий ее блок на структурных схемах изображается символом, показанным на рис. 10.5.

После вставки нулей частота дискретизации сигнала станет равна  $N f_d$ , но период повторения спектра останется прежним —  $f_d$  (см. рис. 10.4, *б*). Теперь необходимо

пропустить полученный сигнал через ФНЧ с частотой среза  $f_d/2$  (его идеализированная АЧХ показана на рис. 10.4, б пунктиром). В результате фильтрации получится интерполированный сигнал, частота дискретизации которого равна  $Nf_d$ , а спектр в полосе частот от нуля до  $f_d/2$  остался прежним (см. рис. 10.4, в).

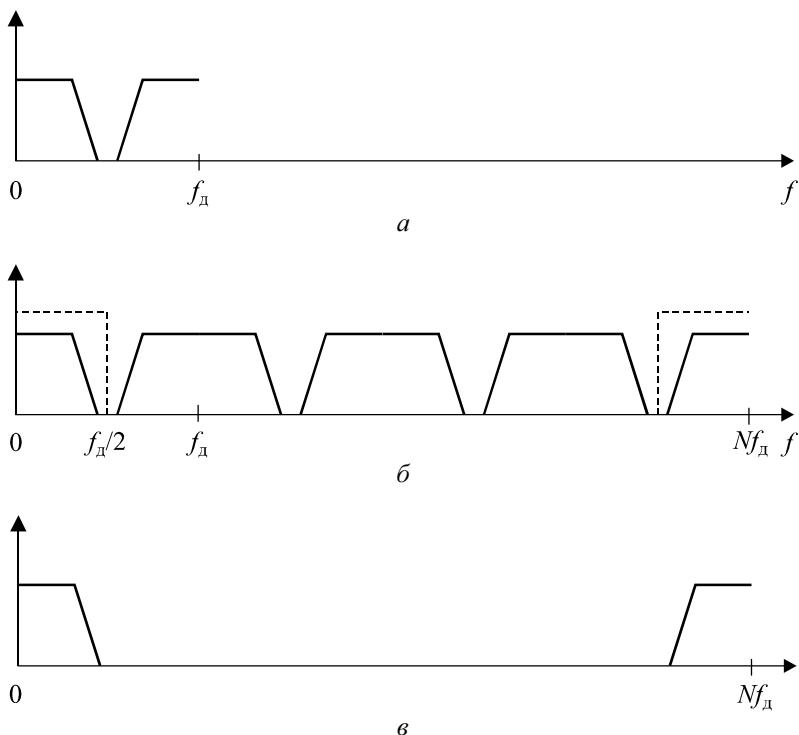


Рис. 10.4. Спектры сигналов в процессе интерполяции

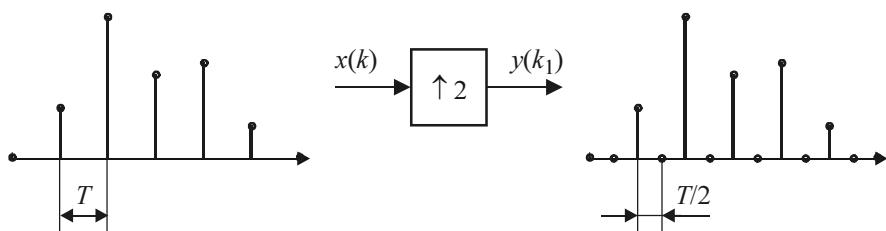


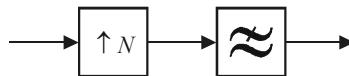
Рис. 10.5. Повышение частоты дискретизации в два раза

### ЗАМЕЧАНИЕ

В формуле (3.7), описывающей связь между спектрами аналогового и дискретизированного сигналов, присутствует множитель  $1/T$ , равный частоте дискретизации  $f_d$ . При повышении частоты дискретизации в  $N$  раз величина этого множителя пропор-

ционально возрастает. Отсюда следует, что в процессе интерполяции необходимо повысить уровень обрабатываемого дискретного сигнала в  $N$  раз. Иными словами, интерполирующий ФНЧ должен иметь в полосе пропускания коэффициент передачи, равный  $N$ .

Таким образом, полная структура устройства, выполняющего интерполяцию, включает в себя каскадно соединенные блок повышения частоты дискретизации и ФНЧ (рис. 10.6).



**Рис. 10.6.** Реализация интерполяции сигнала

Для сохранения фазовых соотношений во входном сигнале следует использовать нерекурсивный фильтр с линейной ФЧХ. Поскольку в линии задержки фильтра в каждый момент содержится большое количество нулевых отсчетов и положения этих отсчетов на каждом шаге заранее известны, при реализации алгоритма можно сэкономить время вычислений, игнорируя арифметические операции с этими отсчетами.

## Передискретизация

Если коэффициент изменения частоты дискретизации не является целым числом, но может быть представлен в виде рациональной дроби, используется процедура *передискретизации* (*resampling*).

Передискретизация является сочетанием интерполяции и прореживания. Если исходную частоту дискретизации необходимо умножить на  $p/q$ , сначала выполняется интерполяция с коэффициентом  $p$ , а затем прореживание с коэффициентом  $q$ . Поэтому в принципе данную операцию можно осуществить средствами MATLAB следующим образом (функции `decimate` и `interp`, реализующие соответственно прореживание и интерполяцию, описываются в *этой главе* далее):

```
y = decimate(interp(x, p), q)
```

Однако в этом случае будет выполняться много лишних операций, которые могут быть устранены при оптимальной организации вычислений.

Рассмотрим последовательность действий, выполняемых при последовательном осуществлении интерполяции и прореживания.

1. Между отсчетами входного сигнала вставляется по  $(p - 1)$  нулей.
2. Полученный сигнал пропускается через ФНЧ с частотой среза  $f_d/2$ .
3. Полученный сигнал пропускается через ФНЧ с частотой среза  $(f_d p/q)/2$ .
4. Из полученного сигнала выбирается каждый  $q$ -й отсчет.

Шаги 1 и 2 относятся к процессу интерполяции, 3 и 4 — к процессу прореживания.

Первое, что можно заметить, — это последовательное пропускание сигнала через два ФНЧ с разными частотами среза. Очевидно, что фильтр с большей частотой среза является избыточным. Поэтому можно оставить только один фильтр с меньшей частотой среза. При повышении частоты дискретизации эта частота среза равна частоте Найквиста входного сигнала, при понижении — частоте Найквиста выходного сигнала.

Вторая модификация та же, что рассматривалась ранее при обсуждении процесса прореживания. Поскольку нас интересует лишь каждый  $q$ -й отсчет фильтрованного сигнала, нет смысла выполнять фильтрующие вычисления для всех отсчетов. Наконец, как и в случае интерполяции, в линии задержки фильтра в каждый момент содержится большое количество нулевых отсчетов, которые можно игнорировать при вычислениях.

С учетом сказанного последовательность действий передискретизации будет выглядеть следующим образом.

1. Между отсчетами входного сигнала вставляется по  $(p - 1)$  нулей.
2. Полученный сигнал порциями по  $q$  отсчетов "заталкивается" в ФНЧ с частотой среза, равной  $f_d/2 \cdot \min(1, p/q)$ . Результаты фильтрации для этих порций являются отсчетами сигнала с частотой дискретизации  $f_d p/q$ . При фильтрации игнорируются арифметические операции с отсчетами, имеющими нулевые значения.

#### **ЗАМЕЧАНИЕ**

Следует помнить, что ФНЧ в данной схеме работает на повышенной частоте дискретизации, равной  $p f_d$ .

## **Многокаскадная реализация прореживания и интерполяции**

Из схем рис. 10.3 и 10.6 видно, что в обоих случаях ФНЧ работает на более высокой из двух частот дискретизации, а его частота среза определяется более низкой из двух частот Найквиста. Поэтому, если необходимо реализовать изменение частоты дискретизации с большим коэффициентом, нормированная частота среза ФНЧ окажется очень низкой, а требуемый порядок фильтра и вычислительные затраты — соответственно, очень большими (ведь порядок нерекурсивного фильтра пропорционален отношению частоты дискретизации к ширине переходной полосы, см., например, формулу (6.23) в главе 6).

Однако эти затраты можно уменьшить, если разложить коэффициент изменения частоты дискретизации на множители и реализовать требуемое преобразование по-этапно, соединив несколько каскадов. Экономия достигается за счет того, что в данном случае имеется несколько фильтров, работающих на разных частотах дискретизации и имеющих разные нормированные частоты среза.

При разделении системы на два каскада часто можно выбрать множители несколькими способами, например,  $100 = 25 \times 4 = 20 \times 5 = 10 \times 10$ . В книге [23] приводится формула, позволяющая определить оптимальный вариант, минимизирующий вычислительные затраты. Согласно ей, один из множителей, составляющих общий коэффициент преобразования ( $N = N_1 N_2$ ) определяется следующим образом:

$$N_{1\text{ opt}} \approx \frac{2N \left( \sqrt{N \frac{F}{2-F}} - 1 \right)}{F(N+1)-2}, \quad (10.1)$$

где  $F = \Delta f / f_{N\text{ min}}$  — ширина переходной полосы ФНЧ, нормированная к минимальной из двух частот Найквиста  $f_{N\text{ min}}$ .

Коэффициент преобразования второго каскада должен быть равен, соответственно,  $N_{2\text{ opt}} = N/N_{1\text{ opt}}$ . Каскад с коэффициентом преобразования  $N_1$  должен работать на более высокой частоте, т. е. являться первым каскадом при прореживании и вторым — при интерполяции.

В качестве примера рассчитаем оптимальное распределение коэффициентов преобразования между каскадами для случая, когда необходимо понизить частоту дискретизации с 400 кГц до 4 кГц ( $N = 100$ ), а максимальная частота в спектре входного сигнала составляет 1,8 кГц (данний пример также взят из книги [23]). Более низкая из двух частот Найквиста в данном случае равна  $f_{N\text{ min}} = 4 \text{ кГц}/2 = 2 \text{ кГц}$ , нормированная ширина переходной полосы составляет  $F = (2 - 1,8)/2 = 0,1$ . Подставив это в (10.1), получаем

$$N_{1\text{ opt}} \approx \frac{2 \cdot 100 \left( \sqrt{100 \cdot \frac{0,1}{2-0,1}} - 1 \right)}{0,1 \cdot (100+1) - 2} \approx 32.$$

Выбираем из приведенных ранее возможных вариантов разложения числа 100 на множители ближайший к полученному значению:  $N_1 = 25$ ,  $N_2 = 4$ . ФНЧ первого каскада работает на частоте дискретизации 400 кГц, выходная частота Найквиста для него составляет  $400/25/2 = 8$  кГц, нормированная ширина переходной полосы  $(8 - 1,8)/200 = 0,031$ . ФНЧ второго каскада работает на частоте дискретизации  $400/25 = 16$  кГц, выходная частота Найквиста 2 кГц, нормированная ширина переходной полосы  $(2 - 1,8)/8 = 0,025$ . Таким образом, нормированная ширина переходных полос обоих фильтров оказывается существенно больше, чем при однокаскадной реализации прореживания ( $(2 - 1,8)/200 = 0,001$ ), за счет этого существенно снижается необходимый порядок фильтров, что и приводит к снижению вычислительных затрат.

## Структуры "интегратор — гребенчатый фильтр"

Если в качестве ФНЧ в схемах рис. 10.3 и 10.6 воспользоваться устройством, вычисляющим скользящую сумму  $N$  отсчетов сигнала, можно получить весьма эффективную в вычислительном отношении реализацию фильтра:

$$H(z) = 1 + z^{-1} + \dots + z^{-(N-1)} = \frac{1 - z^{-N}}{1 - z^{-1}}.$$

Структура этого фильтра в прямой и канонической (но без объединения элементов задержки) формах показана на рис. 10.7.

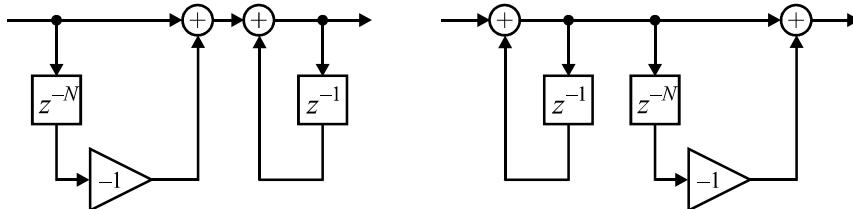


Рис. 10.7. CIC-фильтр в прямой (слева) и канонической (справа) формах

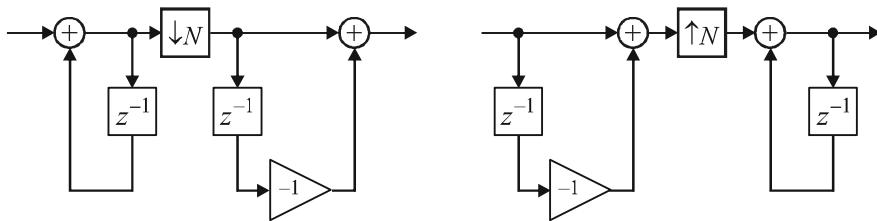
Частотная характеристика данного фильтра имеет вид

$$\dot{K}(\omega) = H(e^{j\omega T}) = \frac{1 - e^{-j\omega NT}}{1 - e^{-j\omega T}} = \exp\left(-j\omega \frac{N-1}{2}T\right) \frac{\sin\left(\frac{\omega NT}{2}\right)}{\sin\left(\frac{\omega T}{2}\right)},$$

т. е. его АЧХ описывается функцией Дирихле (см. главу 3). С точки зрения качества фильтрации данный ФНЧ оставляет желать много лучшего, главную его практическую ценность составляет вычислительная простота — как видно из рис. 10.7, для его реализации не требуется ни одной операции умножения.

Рекурсивная часть фильтра представляет собой дискретный интегратор (она осуществляет суммирование с накоплением), а нерекурсивная — *гребенчатый фильтр* (*comb filter*, его АЧХ имеет вид "гребенки" из равномерно расположенных пиков), поэтому такие структуры обозначаются английской аббревиатурой *CIC* (*Cascaded Integrator — Comb filter*, каскадированные интегратор и гребенчатый фильтр).

При использовании данного фильтра при реализации прореживания и интерполяции можно еще больше сократить требуемые вычислительные ресурсы, поменяв местами нерекурсивную "половинку" фильтра и блок изменения частоты дискретизации (для этого при прореживании необходимо использовать каноническую, а при интерполяции — прямую форму реализации). В результате линия задержки в нерекурсивной части фильтра сокращается в  $N$  раз и становится равной одному отсчету, а структуры прореживающего и интерполирующего устройств принимают вид, показанный на рис. 10.8.



**Рис. 10.8.** Реализация прореживания (слева) и интерполяции (справа) на базе CIC-фильтра

Из-за недостаточного качества фильтрации в таких структурах имеют место неравномерность частотной характеристики в полосе пропускания и заметные помехи от сдвинутых копий спектра. С первой проблемой борются, используя дополнительные корректирующие фильтры, со второй — применяя многокаскадные структуры. Более подробную информацию о CIC-структуре можно найти, например, в [23].

#### ЗАМЕЧАНИЕ

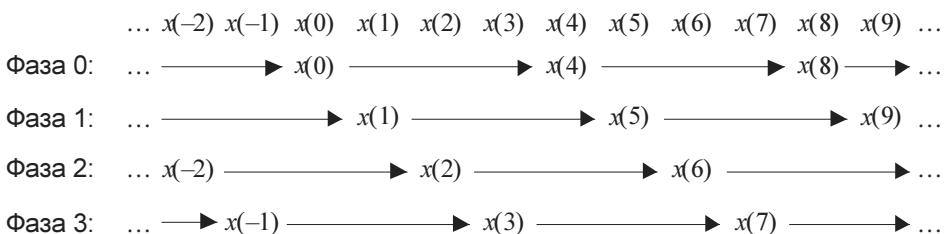
Рекурсивная ветвь CIC-фильтра имеет полюс на единичной окружности, т. е. находится *на грани устойчивости*. Поэтому, чтобы избежать переполнения при рекурсивном суммировании, следует обеспечить отсутствие постоянной составляющей в обрабатываемом сигнале.

## Полифазные структуры

Обсуждая алгоритмы изменения частоты дискретизации, мы увидели, что они могут быть эффективно организованы, если при выполнении операций фильтрации отказаться от вычисления ненужных выходных отсчетов и игнорировать входные отсчеты, равные нулю. Подвести теоретическую базу под эти рассуждения позволяет *полифазное (polyphase) представление* дискретных сигналов, речь о котором пойдет в *данном разделе*.

## Идея полифазного представления сигналов

Смысл полифазного представления сигналов заключается в том, что последовательность отсчетов разделяется на  $N$  фаз (*phase*), сдвинутых друг относительно друга и имеющих пониженную в  $N$  раз частоту дискретизации (рис. 10.9).



**Рис. 10.9.** Разделение сигнала на четыре фазы

Из рис. 10.9 видно, что последовательность отсчетов  $n$ -й фазы можно записать как

$$x_n(k) = \{ \dots, x(n-N), x(n), x(n+N), \dots \} = \{ x(kN+n) \}.$$

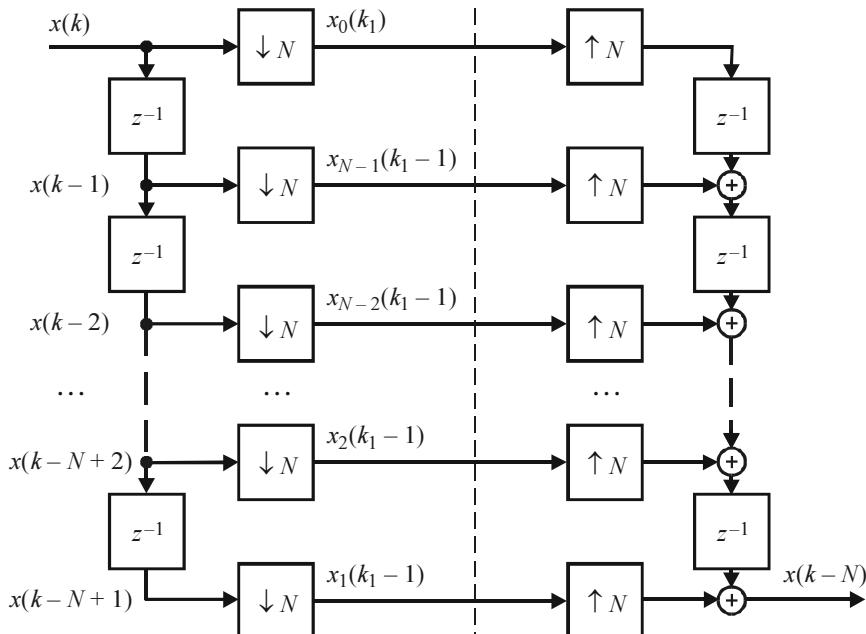
Выясним, как связаны друг с другом  $z$ -преобразования исходного сигнала и отдельных его фаз. Это несложно сделать, воспользовавшись свойствами  $z$ -преобразования (см. соответствующий раздел главы 3). Пусть  $X_n(z)$  —  $z$ -преобразование последовательности, соответствующей  $n$ -й фазе:

$$X_n(z) = \sum_{k=-\infty}^{\infty} x_n(k) z^{-k} = \sum_{k=-\infty}^{\infty} x(kN+n) z^{-k}.$$

Чтобы получить из отдельных фаз исходную последовательность, необходимо вставить между их отсчетами по  $N - 1$  нулей, задержать каждую фазу на число отсчетов, соответствующее ее номеру ( $0, 1, \dots, N - 1$ ), и суммировать результаты. Согласно свойствам  $z$ -преобразования, указанная вставка нулей соответствует замене аргумента с  $z$  на  $z^N$ , а задержка приведет к умножению  $z$ -преобразования на  $z^{-n}$ . В итоге получаем искомую формулу связи:

$$X(z) = \sum_{n=0}^{N-1} X_n(z^N) z^{-n}. \quad (10.2)$$

В следующих разделах мы воспользуемся этой формулой для математического описания процессов, происходящих в многоскоростных системах.



**Рис. 10.10.** Разделение сигнала на фазы (слева) и сложение сигнала из отдельных фаз (справа)

На рис. 10.10 показана структура системы, "разбирающей" сигнал на отдельные фазы, а затем "собирающей" его обратно. На этом рисунке следует обратить внимание на нумерацию фаз внутри схемы, а также на их задержки. Кроме того, необходимо отметить, что последовательное выполнение операций разделения сигнала на фазы и его обратной сборки приводит к результирующей задержке, равной  $N$  отсчетам.

## Полифазная реализация процесса интерполяции

Сигнал, подвергаемый фильтрации в схеме интерполяции, показанной ранее на рис. 10.6, содержит большое количество нулевых значений. Умножение этих нулей на коэффициенты импульсной характеристики фильтра в процессе расчета свертки — совершенно непроизводительная траты вычислительных ресурсов. Попробуем ликвидировать эти "лишние" операции, реорганизовав процесс вычислений. Для этого прежде всего установим связь между  $z$ -преобразованиями входного и интерполированного сигналов.

Вывод соответствующего соотношения несложен — достаточно воспользоваться свойствами  $z$ -преобразования (см. главу 3), касающимися вставки нулей и дискретной свертки (формула (3.25)). В результате получаем следующее:

$$Y(z) = X(z^N)H(z). \quad (10.3)$$

Теперь представим функцию передачи ФНЧ  $H(z)$  в полифазном виде согласно формуле (10.2):

$$H(z) = \sum_{n=0}^{N-1} H_n(z^N)z^{-n}. \quad (10.4)$$

Здесь

$$H_0(z) = h_0 + h_N z^{-1} + h_{2N} z^{-2} + \dots,$$

$$H_1(z) = h_1 + h_{N+1} z^{-1} + h_{2N+1} z^{-2} + \dots$$

и т. д. — функции передачи *частичных фильтров*, полученных путем разделения исходной импульсной характеристики на отдельные фазы. Импульсная характеристика  $n$ -го частичного фильтра получается путем выборки из исходной импульсной характеристики каждого  $N$ -го отсчета, начиная с  $n$ -го.

Переписав (10.3) с учетом (10.4), получаем

$$Y(z) = X(z^N) \sum_{n=0}^{N-1} H_n(z^N)z^{-n} = \sum_{n=0}^{N-1} X(z^N)H_n(z^N)z^{-n} = \sum_{n=0}^{N-1} Y_n(z^N)z^{-n}, \quad (10.5)$$

где  $Y_n(z) = X(z)H_n(z)$  — результат обработки входного сигнала  $n$ -м частичным фильтром.

Формула (10.5) показывает, что обработка сигнала частичными фильтрами может производиться на исходной частоте дискретизации, а повышение частоты дискретизации путем вставки нулей осуществляться на выходах частичных фильтров.

В результате получаем следующую схему (рис. 10.11), называемую *полифазной реализацией* процесса интерполяции. В данном случае мы заменяем один фильтр, работающий на повышенной частоте дискретизации, набором из  $N$  фильтров, работающих на исходной частоте дискретизации.

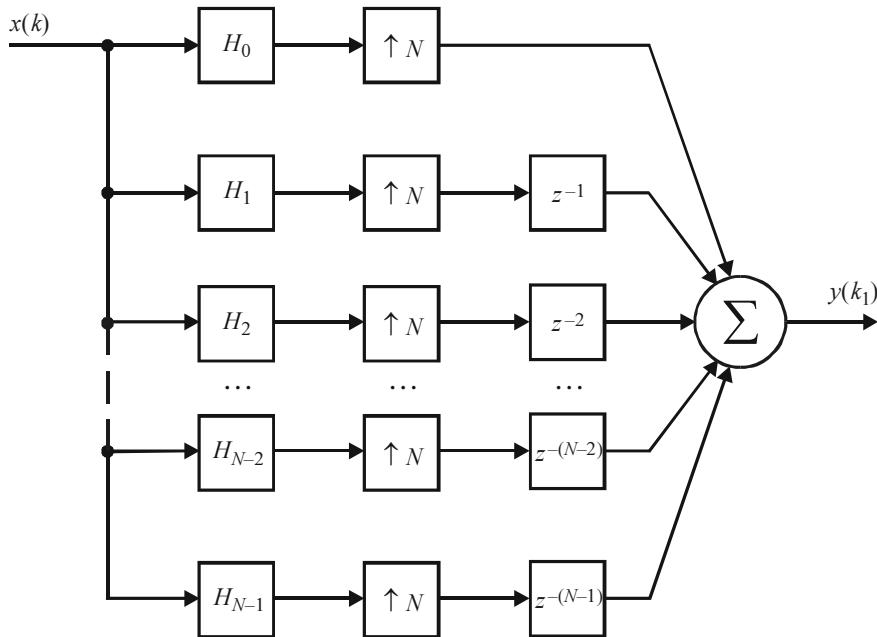


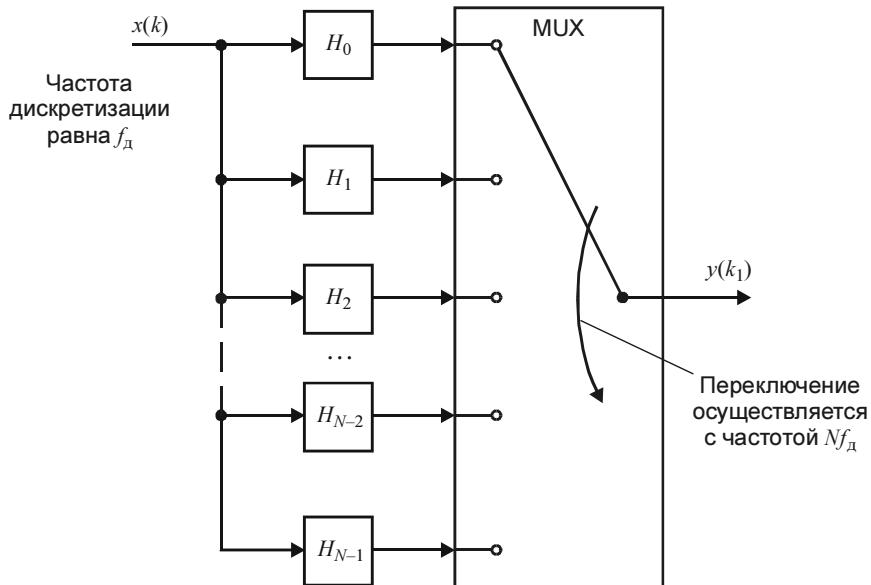
Рис. 10.11. Полифазная реализация интерполяции сигнала

Вставка нулей теперь производится после фильтрации, так что от незначащих операций при вычислении свертки мы избавились. Однако при реализации системы непосредственно по схеме рис. 10.11 избыточными оказываются операции сложения — ведь при вычислении каждого отсчета выходного сигнала ненулевое значение присутствует только на одном из входов сумматора.

По сути дела, блоки вставки нулей и элементы задержки на рис. 10.11 образуют *мультиплексор* — на выходах частичных фильтров одновременно появляются  $N$  значений интерполированного сигнала, которые *поочередно* подаются на выход, что и обеспечивает повышение частоты дискретизации. Соответствующий вариант схемы, который обычно и используется при практической реализации интерполяции, показан на рис. 10.12.

## Полифазная реализация процесса прореживания

Из выходного сигнала фильтра в схеме прореживания, показанной ранее на рис. 10.3, выбрасывается часть отсчетов, что делает ненужными вычислительные операции, затраченные на их расчет. Ликвидируем эти непроизводительные затраты, реорганизовав процесс вычислений.



**Рис. 10.12.** Интерполяция с мультиплексированием выходов частичных фильтров

Для этого, как и при рассмотрении интерполяции, выявим связь между  $z$ -преобразованиями входного и выходного сигналов. Однако, поскольку процесс выборки каждого  $N$ -го отсчета не имеет простого описания в  $z$ -области, начнем с того, что представим  $z$ -преобразование входного сигнала в полифазном виде согласно формуле (10.2):

$$X(z) = \sum_{n=0}^{N-1} X_n(z^N) z^{-n},$$

где

$$X_0(z) = x_0 + x_N z^{-1} + x_{2N} z^{-2} + \dots,$$

$$X_1(z) = x_1 + x_{N+1} z^{-1} + x_{2N+1} z^{-2} + \dots$$

и т. д. —  $z$ -преобразования отдельных фаз входного сигнала.

Кроме того, импульсную характеристику ФНЧ также представим в полифазном виде согласно приведенной ранее формуле (10.4). В результате  $z$ -преобразование входного сигнала, пропущенного через ФНЧ, можно представить в следующей форме:

$$X'(z) = X(z)H(z) = \left( \sum_{n=0}^{N-1} X_n(z^N) z^{-n} \right) \left( \sum_{n=0}^{N-1} H_n(z^N) z^{-n} \right).$$

Преобразуем произведение сумм в двойную сумму:

$$X'(z) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} X_{n_1}(z^N) z^{-n_1} H_{n_2}(z^N) z^{-n_2} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} X_{n_1}(z^N) H_{n_2}(z^N) z^{-(n_1+n_2)}. \quad (10.6)$$

Стоящий после ФНЧ блок понижения частоты дискретизации, согласно рис. 10.10, выделяет из сигнала его фазу с нулевым номером. Эту фазу образуют отсчеты с номерами, кратными  $N$ , поэтому для получения  $z$ -преобразования выходного сигнала необходимо выделить в (10.6) слагаемые, у которых степень  $z$  кратна  $N$ . Это имеет место в следующих ситуациях:

- $n_1 = n_2 = 0$  ;
- $n_2 = N - n_1$  для  $n_1 = 1, 2, \dots, N - 1$  .

В итоге, после замены  $z^N$  на  $z$ , отражающей понижение частоты дискретизации в  $N$  раз, мы получаем следующее выражение для  $z$ -преобразования выходного сигнала:

$$Y(z) = X_0(z)H_0(z) + \sum_{n=1}^{N-1} X_n(z)H_{N-n}(z)z^{-1}.$$

Сопоставляя эту формулу с рис. 10.10, мы получаем схему, показанную на рис. 10.13 и называемую *полифазной реализацией* процесса прореживания. Как и при интерполяции, мы заменяем один фильтр, работающий на исходной (высокой) частоте дискретизации, набором из  $N$  фильтров, работающих на выходной (пониженной) частоте дискретизации.

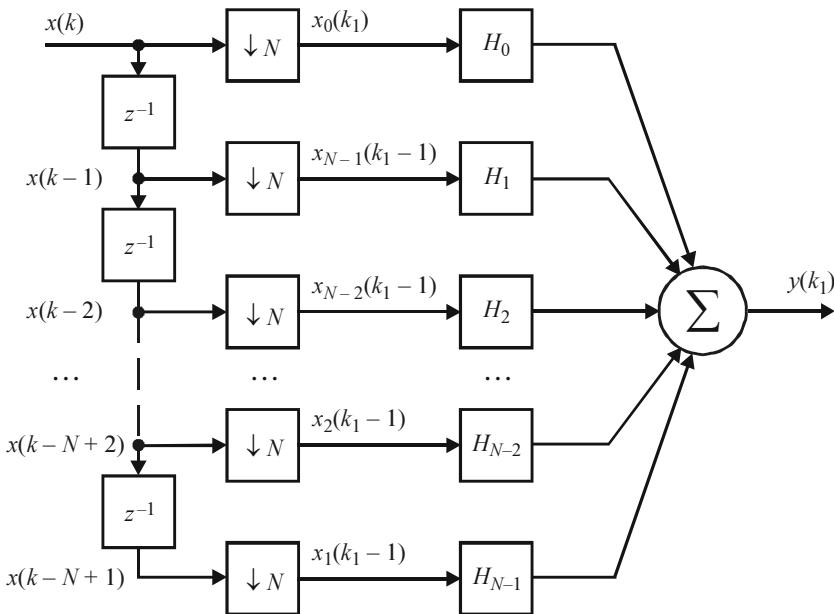


Рис. 10.13. Полифазная реализация прореживания сигнала

Отбрасывание лишних отсчетов теперь производится до выполнения фильтрации, так что от операций по их расчету мы избавились. Осталось заметить, что роль элементов задержки и блоков понижения частоты дискретизации в данной схеме состоит в том, чтобы обеспечить одновременную подачу "кадра" из  $N$  отсчетов

входного сигнала на входы частичных фильтров. Это можно реализовать с помощью демультиплексора, осуществляющего необходимую коммутацию входного сигнала. Соответствующий вариант схемы, который обычно и используется при практической реализации прореживания, показан на рис. 10.14. В данной структуре имеется в виду, что результаты фильтрации сохраняются на выходах частичных фильтров в течение всего цикла коммутации, чтобы обеспечить правильное сложение выходных сигналов.

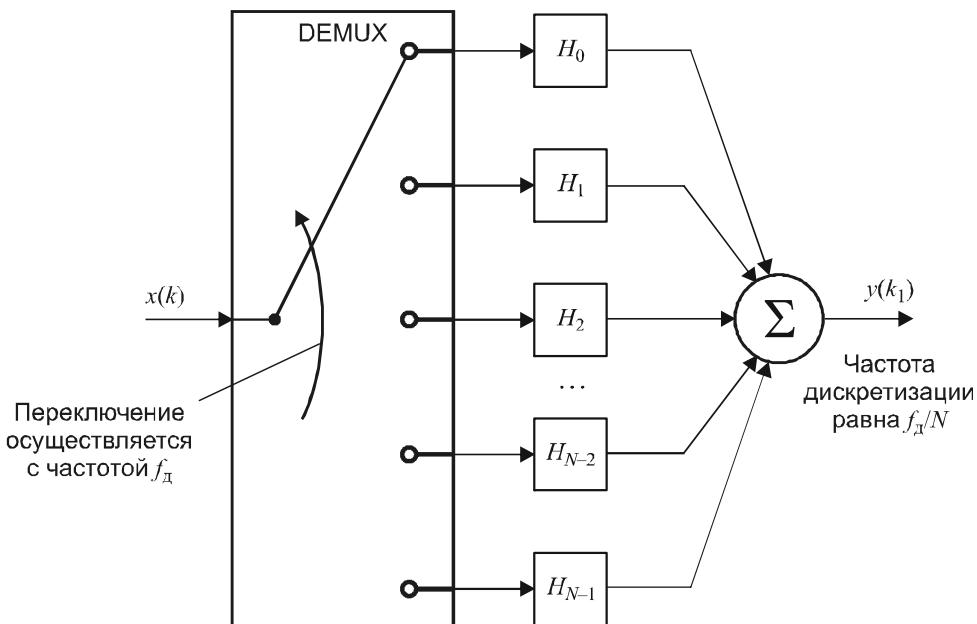


Рис. 10.14. Прореживание с демультиплексированием входного сигнала

## Банки фильтров

Задачи, решаемые многоскоростными системами, не ограничиваются простым изменением частоты дискретизации сигнала. Большое распространение в современных системах получили так называемые *банки фильтров* (*filter bank*), позволяющие сэкономить число вычислительных операций за счет одновременной и взаимосвязанной реализации нескольких фильтров. Экономия в данном случае, так же как и в описанных ранее устройствах, достигается за счет полифазного представления сигналов и фильтров.

Различают *банки анализа* (*analysis bank*) и *банки синтеза* (*synthesis bank*), их обобщенная структура показана на рис. 10.15. Банк анализа состоит из  $N$  фильтров с функциями передачи  $H^{(n)}(z)$  и общим входным сигналом  $x(k)$ . Выходные сигналы фильтров  $u_n(k)$  являются более узкополосными, чем входной сигнал, поэтому их частоту дискретизации можно понизить. Часто коэффициент понижения частоты

дискретизации может быть равен числу каналов  $N$ . В итоге получаются  $N$  выходных сигналов  $y_n(k_1)$ .

Банк синтеза решает обратную задачу —  $N$  входных сигналов  $x_n(k)$  проходят через блоки повышения частоты дискретизации, обрабатываются фильтрами с функциями передачи  $F^{(n)}(z)$  и затем суммируются, формируя выходной сигнал  $y(k_1)$ .

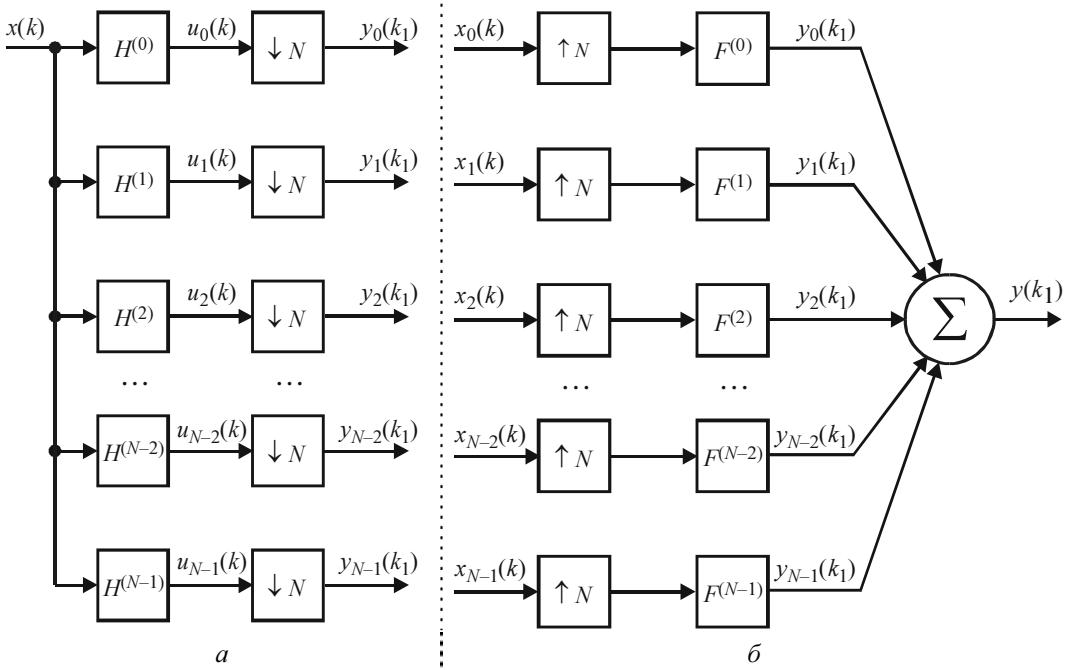


Рис. 10.15. Банки фильтров анализа (a) и синтеза (б)

Рисунок 10.15 наглядно демонстрирует, что банк анализа и банк синтеза можно соединить друг с другом, получив систему, внутри которой обрабатываемый сигнал разделен на несколько каналов с разными частотными свойствами. В этих каналах сигнал может подвергаться дополнительной обработке, какой именно — зависит от решаемой задачи. Такую обработку сигнала называют *субполосной обработкой* (*subband processing*).

Наибольшее распространение получили банки фильтров, основанные на *ДПФ* (*DFT filter bank*), которые и будут рассмотрены в *данном разделе*.

## Банк анализа

Пусть импульсные характеристики фильтров, использованных в банке анализа, показанном на рис. 10.15 слева, связаны друг с другом следующим образом:

$$h^{(n)}(k) = h(k) \exp(-j\omega_n k), \quad (10.7)$$

где  $h(k)$  — импульсная характеристика общего для всех фильтров *прототипа*, а  $\omega_n$  — частотный сдвиг. Согласно свойствам преобразования Фурье (см. главу 1), частотные характеристики семейства фильтров, полученного таким образом, будут отличаться друг от друга только сдвигом по частоте:

$$\dot{K}^{(n)}(\omega) = \dot{K}(\omega + \omega_n). \quad (10.8)$$

### ЗАМЕЧАНИЕ

Прототип в данном случае должен являться фильтром низких частот. Остальные фильтры, полученные на его основе при  $n \neq 0$ , будут полосовыми фильтрами.

Кроме того, будем считать, что частоты  $\omega_n$  образуют равномерную сетку, аналогичную той, что используется в ДПФ (см. главу 5 и формулу (5.14)):

$$\omega_n = \frac{2\pi}{N}n, \quad n = 0, 1, 2, \dots, N-1. \quad (10.9)$$

Выходной сигнал фильтра  $n$ -го канала представляет собой свертку входного сигнала с импульсной характеристикой  $n$ -го фильтра:

$$u_n(k) = \sum_{m=-\infty}^{\infty} h^{(n)}(m)x(k-m) = \sum_{m=-\infty}^{\infty} h(m)x(k-m)\exp\left(-j\frac{2\pi n}{N}m\right). \quad (10.10)$$

Представим формулу (10.10) в полифазном виде, разделив сумму на  $N$  отдельных секций:

$$u_n(k) = \sum_{p=-\infty}^{\infty} \sum_{m=0}^{N-1} h(pN+m)x(k-pN-m)\exp\left(-j\frac{2\pi n}{N}(pN+m)\right). \quad (10.11)$$

В (10.11)  $m$  обозначает номер фазы, а  $p$  — номер отсчета в пределах фазы.

Теперь заметим, что в полученном выражении имеется экспоненциальный множитель, равный единице:

$$\exp\left(-j\frac{2\pi n}{N}pN\right) = \exp(-j2\pi np) = 1,$$

поскольку в этой формуле  $n$  и  $p$  — целые числа. С учетом этого (10.11) принимает вид

$$u_n(k) = \sum_{p=-\infty}^{\infty} \sum_{m=0}^{N-1} h(pN+m)x(k-pN-m)\exp\left(-j\frac{2\pi n}{N}m\right). \quad (10.12)$$

При понижении частоты дискретизации производится выборка каждого  $N$ -го отсчета, так что выходной сигнал  $n$ -го канала можно записать следующим образом:

$$y_n(k_1) = u_n(k_1N) = \sum_{p=\infty}^{\infty} \sum_{m=0}^{N-1} h(pN+m)x(k_1N-pN-m)\exp\left(-j\frac{2\pi n}{N}m\right). \quad (10.13)$$

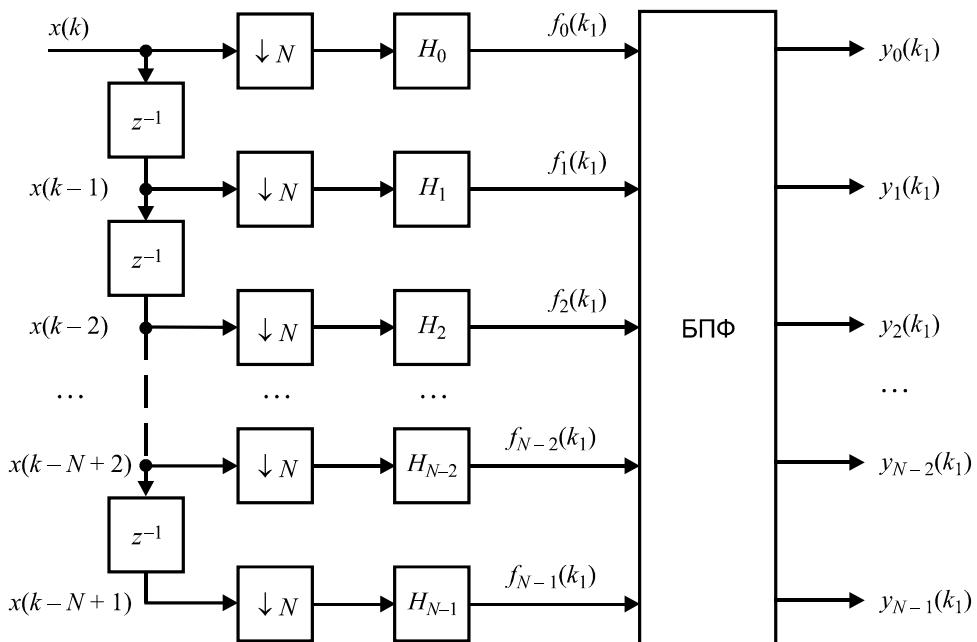
Сумма по  $t$  в этом выражении представляет собой прямое ДПФ (см. формулу (5.3) в главе 5), которое применяется к набору отсчетов

$$f_m(k_1) = \sum_{p=-\infty}^{\infty} h(pN+m)x((k_1-p)N-m), \quad m = 0, 1, \dots, N-1.$$

Зависимость  $f_m$  от  $k_1$  при фиксированном  $m$  представляет собой свертку последовательностей  $\{h(pN + m)\}$  и  $\{x(pN - m)\}$ , имеющих следующий смысл:

- $\{h(pN + m)\}$  — отсчеты  $m$ -й фазы импульсной характеристики фильтра;
  - $\{x(pN - m)\}$  — отсчеты  $(N - m)$ -й фазы входного сигнала.

Реализация вычислений согласно формуле (10.13) приводит к структурной схеме банка фильтров анализа, показанной на рис. 10.16. Фазы входного сигнала в данной схеме формируются так же, как было показано ранее на рис. 10.10, а обозначения  $H_0, H_1, \dots, H_{N-1}$  использованы для функций передачи частичных фильтров (см. ранее формулу (10.4)). На рисунке также отражен тот факт, что при практической реализации банков фильтров вычисление ДПФ обычно выполняется с использованием быстрого алгоритма (БПФ; см. соответствующий раздел главы 5).



**Рис. 10.16.** Полифазная реализация банка фильтров анализа на основе ДПФ

## ЗАМЕЧАНИЕ

На входе схемы рис. 10.16 вместо элементов задержки и блоков понижения частоты дискретизации можно использовать демультиплексор, подобно тому как это было сделано в схеме устройства прореживания, показанной ранее на рис. 10.14.

В качестве примера реализуем 16-канальный банк фильтров и пропустим через него частотно-модулированный сигнал. В MATLAB не существует готовых средств для моделирования банков фильтров, поэтому все придется сделать вручную. Прежде всего зададим важные количественные параметры  $N$  (число частотных каналов) и  $P$  (длина частичных фильтров), после чего рассчитаем ФНЧ, на базе которого будет построен наш банк. Воспользуемся для этого функцией `firl` (см. главу 6) и выберем порядок фильтра равным  $N*P-1$  (вычитание единицы необходимо, поскольку число коэффициентов нерекурсивного фильтра на единицу превышает его порядок). Нормированная частота среза синтезируемого ФНЧ должна быть равна  $1/N$ , т. к. мы хотим разделить общую рабочую полосу системы на  $N$  каналов. Далее выделим отдельные фазы импульсной характеристики фильтра. Для этого удобнее всего воспользоваться функцией `buffer`, которая делит сигнал на блоки указанного размера, располагая их по столбцам. Строки полученной матрицы в соответствуют частичным фильтрам, необходимым для реализации банка:

```
>> N = 16; % число частотных каналов
>> P = 8; % длина каждого частичного фильтра
>> b = firl(N*P-1, 1/N); % расчет ФНЧ
>> B = buffer(b, N); % строки — фазы импульсной характеристики
```

Теперь сформируем входной сигнал для данного примера. Это будет комплексная экспонента с мгновенной частотой, за одну секунду меняющейся по линейному закону от нуля до частоты дискретизации, выбранной равной 10 кГц. Для деления входного сигнала на фазы снова воспользуемся функцией `buffer`, сформировав матрицу  $x$ , строки которой соответствуют отдельным фазам сигнала. Кроме того, заметим, что, в соответствии с формулой (10.13) и рис. 10.16, первая фаза входного сигнала обрабатывается последней фазой импульсной характеристики фильтра, вторая фаза — предпоследней и т. д. Поэтому для удобства дальнейших вычислений перевернем матрицу  $x$  по вертикали с помощью функции `flipud`, чтобы номера фаз сигнала и фильтра совпали:

```
>> Fs = 10e3; % частота дискретизации
>> t = 0:1/Fs:1; % одна секунда дискретного времени
>> x = exp(ji*2*pi*Fs*t.^2/2); % комплексный ЛЧМ-сигнал
>> X = buffer(x, N); % строки — фазы сигнала
>> X = flipud(X);
```

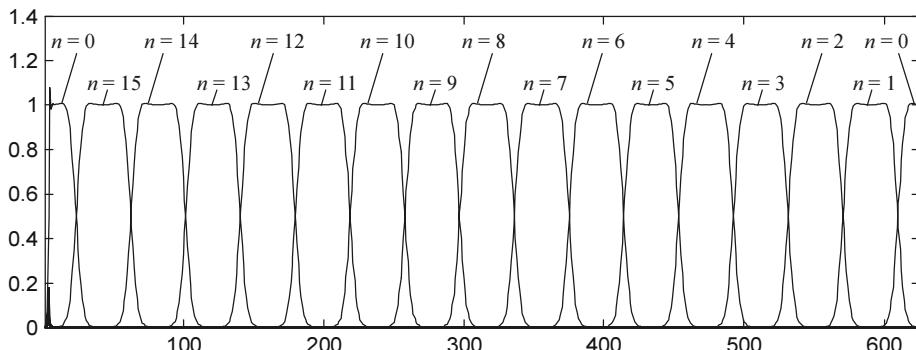
Подготовительные работы закончены, теперь настала очередь реализовывать собственно фильтрацию. Обрабатываем в цикле отдельные фазы сигнала (строки матрицы  $x$ ) частичными фильтрами (импульсные характеристики которых хранятся в строках матрицы  $B$ ), сохраняя результаты в строках промежуточной матрицы  $F$ . Столбцы этой матрицы, согласно рис. 10.16, будут содержать входные данные для дискретного преобразования Фурье. Функция `fft` обрабатывает матричный входной сигнал именно по столбцам, так что дополнительных преобразований сигнала не требуется:

```
>> for k=1:N, F(k,:) = filter(B(k,:), 1, X(k,:)); end
>> Y = fft(F);
```

Строки полученной матрицы  $Y$  содержат выходные сигналы отдельных каналов банка фильтров. Эти сигналы являются комплексными, поэтому для визуализации мы используем их модуль. Кроме того, функция `plot` строит график матрицы по столбцам, поэтому матрицу  $Y$  необходимо транспонировать:

```
>> plot(abs(Y'), 'b')
>> xlim([1 size(Y, 2)]) % точные пределы оси времени
```

Полученный результат показан на рис. 10.17 (подписи с номерами каналов добавлены вручную). На графиках хорошо видно, как по мере изменения мгновенной частоты сигнала он поочередно попадает на выходы разных каналов банка фильтров.



**Рис. 10.17.** Модули выходных сигналов банка фильтров анализа

Последовательность появления сигнала на выходах разных каналов нуждается в дополнительных комментариях. Прежде всего заметим, что начиная с середины сигнала, использованного в данном примере, частота его колебаний превышает частоту Найквиста. В соответствии с законом появления ложных частот при дискретизации (см. главу 3) это означает, что частота дискретизированного сигнала, начиная с данного момента, становится отрицательной. При этом она продолжает линейно возрастать со временем и к концу сигнала достигает нулевого значения. Сказанное относится и к полосам пропускания отдельных каналов, средние частоты которых, согласно (10.8), уменьшаются с ростом  $n$ . В результате в канал 0 попадают положительные и отрицательные частоты, близкие к нулю (этим объясняется тот факт, что в конце сигнал снова появляется на выходе нулевого канала), а канал 8 пропускает положительные и отрицательные частоты, близкие по модулю к частоте Найквиста. Полосы пропускания каналов 1—7 целиком лежат в области отрицательных частот, а каналов 9—15 — в области положительных частот.

## Банк синтеза

Теперь рассмотрим процессы, происходящие в правой части схемы, показанной ранее на рис. 10.15. Входной сигнал  $n$ -го канала,  $x_n(k)$ , поступает на вход блока увеличения частоты дискретизации в  $N$  раз, после чего подвергается обработке

фильтром с функцией передачи  $F^{(n)}(z)$  и, соответственно, импульсной характеристикой  $f^{(n)}(k)$ . Выходной сигнал фильтра  $n$ -го канала, таким образом, равен

$$y_n(k_1) = \sum_{k=-\infty}^{\infty} x_n(k) f^{(n)}(k_1 - kN).$$

Полный выходной сигнал банка фильтров представляет собой сумму выходных сигналов всех каналов:

$$y(k_1) = \sum_{n=0}^{N-1} y_n(k_1) = \sum_{n=0}^{N-1} \sum_{k=-\infty}^{\infty} x_n(k) f^{(n)}(k_1 - kN).$$

Теперь учтем приведенное ранее соотношение (10.7) между импульсными характеристиками фильтров разных каналов (импульсная характеристика прототипа в данном случае обозначена как  $f(k_1)$ ):

$$\begin{aligned} y(k_1) &= \sum_{n=0}^{N-1} \sum_{k=-\infty}^{\infty} x_n(k) f(k_1 - kN) \exp\left(-j \frac{2\pi}{N} n(k_1 - kN)\right) = \\ &= \sum_{n=0}^{N-1} \sum_{k=-\infty}^{\infty} x_n(k) f(k_1 - kN) \exp\left(-j \frac{2\pi}{N} nk_1\right). \end{aligned}$$

Здесь было учтено, что  $\exp\left(-j \frac{2\pi}{N} nkN\right) = \exp(-j2\pi nk) = 1$ .

Изменим порядок суммирования, сделав сумму по  $n$  внутренней:

$$y(k_1) = \sum_{k=-\infty}^{\infty} f(k_1 - kN) \left[ \sum_{n=0}^{N-1} x_n(k) \exp\left(-j \frac{2\pi}{N} nk_1\right) \right]. \quad (10.14)$$

Сумма, стоящая в квадратных скобках, представляет собой  $k_1$ -й спектральный отсчет прямого ДПФ от набора отсчетов  $\{x_n(k)\}$ ,  $n = 0, 1, \dots, N-1$ , т. е. от набора входных сигналов всех каналов на такте  $k$ . Поскольку  $k_1$  в (10.14) может принимать любые значения, следует воспользоваться свойством периодичности результатов ДПФ. Обозначим результаты вычисления данного ДПФ как  $\{\dot{X}_{k_1}(k)\}$ :

$$\dot{X}_{k_1}(k) = \sum_{n=0}^{N-1} x_n(k) \exp\left(-j \frac{2\pi}{N} nk_1\right).$$

С учетом этого (10.14) принимает вид

$$y(k_1) = \sum_{k=-\infty}^{\infty} \dot{X}_{k_1}(k) h(k_1 - kN). \quad (10.15)$$

Данная сумма представляет собой свертку выходного сигнала  $k_1$ -го частотного канала ДПФ и  $k_1$ -й фазы импульсной характеристики фильтра. Таким образом, каж-

дый спектральный отсчет ДПФ фильтруется соответствующим частичным фильтром. При разных  $k_1$ , согласно (10.15), на выход схемы должны попадать сигналы разных фаз. Поскольку набор результатов ДПФ является периодическим с периодом  $N$  (иными словами,  $\dot{X}_{k_1+N}(k) = \dot{X}_{k_1}(k)$ ), а  $(k_1 + N)$ -я фаза импульсной характеристики фильтра отличается от  $k_1$ -й только задержкой на  $N$  отсчетов, перебор фаз должен производиться циклически с периодом  $N$ . Для этого можно использовать мультиплексор, подобно тому как это было сделано в схеме интерполятора (см. рис. 10.12).

Сказанное приводит к схеме реализации банка фильтров синтеза с использованием ДПФ, показанной на рис. 10.18.

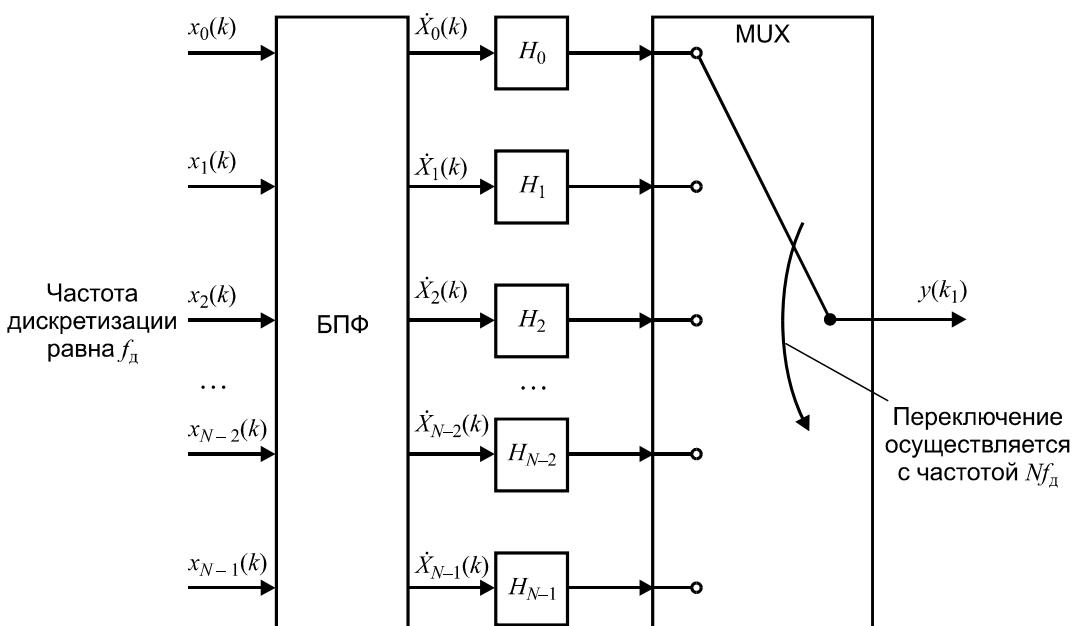


Рис. 10.18. Полифазная реализация банка фильтров синтеза на основе ДПФ

### ЗАМЕЧАНИЕ

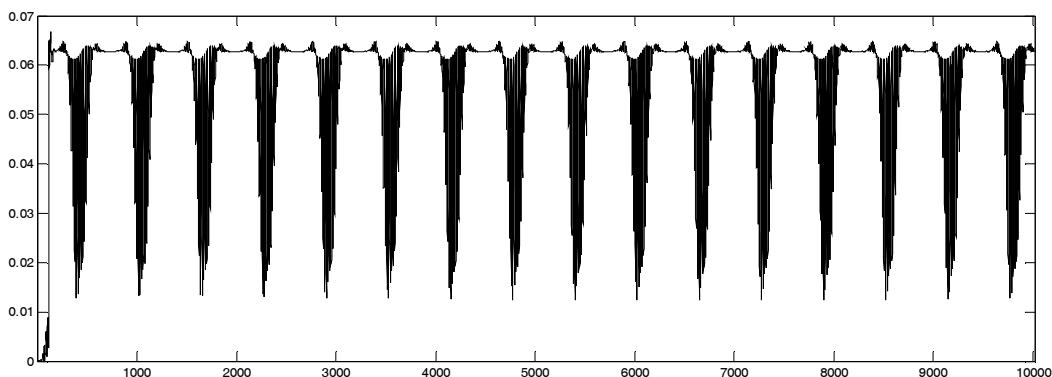
На выходе схемы вместо мультиплексора можно использовать элементы задержки и сумматор, подобно тому как это было сделано в схеме интерполятора, показанной ранее на рис. 10.11.

В качестве примера посмотрим, что получится, если смоделировать банк синтеза, используя выходные сигналы и фильтр из предыдущего примера:

```
>> X1 = fft(Y); % выполняем ДПФ
>> for k=1:N      % цикл по каналам
>>     y(k,:) = filter(B(k,:), 1, X1(k,:));
>> end
```

```
>> y = y(:); % вытягиваем в один столбец
>> plot(abs(y))
>> xlim([1 length(y)]) % точные пределы оси времени
```

На рис. 10.19 показан график модуля комплексного выходного сигнала банка фильтров синтеза. Видно, что выходной сигнал имеет в  $N$  раз меньший уровень, чем входной сигнал банка фильтров анализа (это произошло из-за того, что интерполирующие фильтры в данном примере имели в полосе пропускания коэффициент передачи, равный единице, а не  $N$ , как это требуется для выполнения операции интерполяции — см. замечание к рис. 10.4). Кроме того, сигнал претерпевает сильные искажения в моменты перехода из канала в канал (см. ранее рис. 10.17). Однако в данном примере не ставилась задача обеспечить точное восстановление исходного сигнала, поэтому фильтр был выбран произвольно.



**Рис. 10.19.** Модуль выходного сигнала банка фильтров синтеза

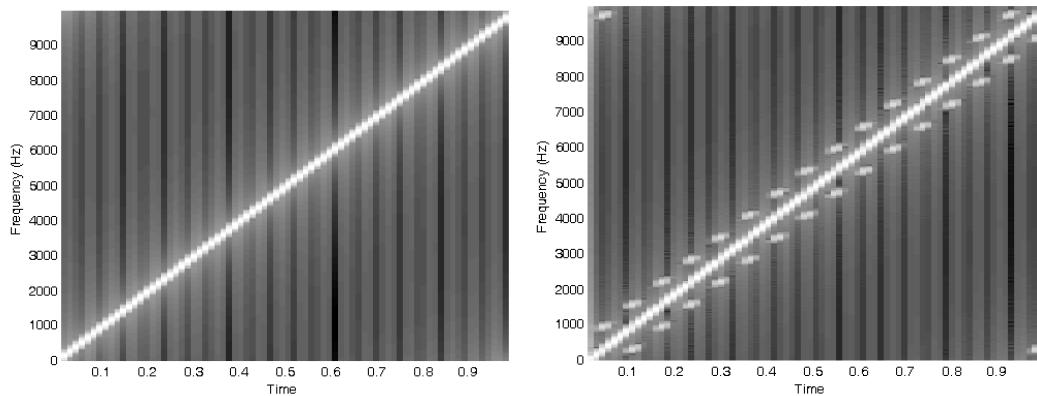
### ЗАМЕЧАНИЕ

Пары банков анализа и синтеза, обеспечивающие совпадение (с точностью до задержки) входного и выходного сигналов, называются *банками точного восстановления* (*perfect reconstruction, PR*).

В заключение данного примера приведем спектрограммы входного сигнала банка фильтров анализа ( $x$ ) и выходного сигнала банка фильтров синтеза ( $y$ ), чтобы проверить, корректно ли восстановлены частотные свойства сигнала (по временным зависимостям сделать это для комплексного сигнала довольно сложно):

```
>> spectrogram(x, 256, [], [], Fs, 'yaxis'); colormap gray
>> figure
>> spectrogram(y, 256, [], [], Fs, 'yaxis'); colormap gray
```

Полученные спектрограммы показаны на рис. 10.20. На графиках наглядно видно, что общая форма зависимости мгновенной частоты сигнала от времени восстановлена правильно. Также хорошо заметны искажения, возникающие при переходе сигнала из одного частотного канала в другой.



**Рис. 10.20.** Спектрограммы входного сигнала банка фильтров анализа (слева) и выходного сигнала банка фильтров синтеза (справа)

## Функции изменения частоты дискретизации в MATLAB

В пакете расширения Signal Processing Toolbox имеется несколько функций, предназначенных для изменения частоты дискретизации сигнала и реализующих алгоритмы, описанные в начале данной главы. К таким функциям относятся:

- `downsample` — понижение частоты дискретизации сигнала;
- `upsample` — повышение частоты дискретизации сигнала;
- `decimate` — прореживание сигнала;
- `interp` — интерполяция сигнала;
- `resample` — изменение частоты дискретизации с рациональным коэффициентом (ядром данной процедуры является функция `upfirdn`).

Далее использование этих функций рассматривается более подробно.

## Функции изменения частоты дискретизации

Функции `downsample` и `upsample` реализуют операции выделения каждого  $N$ -го отсчета и вставки нулей соответственно. Иллюстрации, поясняющие выполняемые ими действия, были приведены в этой главе ранее на рис. 10.1 и 10.5. Функции имеют одинаковый синтаксис вызова:

```
y = downsample(x, N, phase)
y = upsample(x, N, phase)
```

Здесь  $x$  — входной сигнал,  $N$  — целочисленный коэффициент изменения частоты дискретизации,  $y$  — выходной сигнал. Если входной сигнал  $x$  является матрицей, она обрабатывается по столбцам.

Третий входной параметр `phase` является необязательным (его значение по умолчанию равно нулю), он управляет "начальной фазой" выборки отсчетов и вставки нулей. Это происходит следующим образом:

- при понижении частоты дискретизации из входного сигнала `x` выбирается каждый  $N$ -й отсчет начиная с `x(phase+1)`;
- при повышении частоты дискретизации между отсчетами входного сигнала `x` вставляется по  $N-1$  нулевых значений, при этом перед первым отсчетом `x(1)` добавляется `phase` нулевых значений.

## Функция прореживания

Прореживание в MATLAB выполняется с помощью функции `decimate`. Синтаксис ее вызова следующий:

```
y = decimate(x, r)
```

Здесь `x` — входной сигнал, `r` — целочисленный коэффициент понижения частоты дискретизации. Выходной параметр `y` — прореженный сигнал.

В качестве предварительного фильтра по умолчанию используется ФНЧ Чебышева первого рода 8-го порядка с уровнем пульсаций в полосе пропускания 0,05 дБ и частотой среза, равной 0,8 новой (после прореживания) частоты Найквиста. Для устранения фазовых искажений применяется двунаправленная обработка входного сигнала с помощью функции `filtfilt` (см. разд. "Компенсация фазового сдвига" главы 4). Поэтому в конечном счете порядок фильтра удваивается.

Можно задать порядок фильтра Чебышева, используя для этого третий числовой входной параметр:

```
y = decimate(x, r, n)
```

При этом не рекомендуется использовать  $n > 13$ .

Если в качестве третьего входного параметра использовать строку '`fir`', будет использован нерекурсивный фильтр 30-го порядка, рассчитанный с помощью функции `firl` (см. разд. "Функции синтеза с использованием окон" главы 6):

```
y = decimate(x, r, 'fir')
```

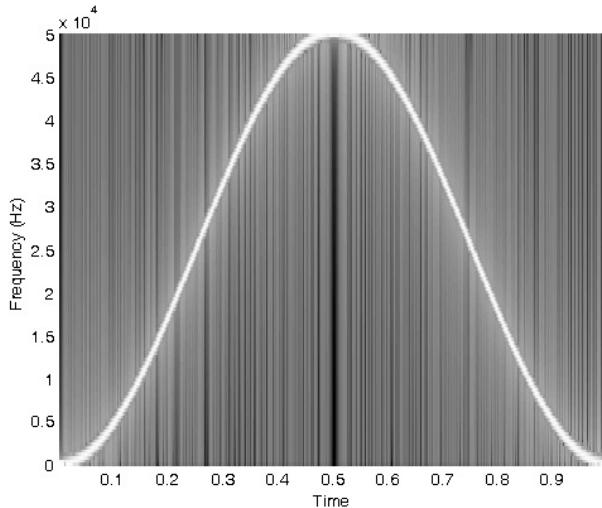
Данный нерекурсивный фильтр имеет линейную ФЧХ, поэтому производить двунаправленную фильтрацию нет необходимости. Это позволяет уменьшить расход памяти при выполнении операции прореживания.

Наконец, при использовании четырех входных параметров можно задать порядок нерекурсивного фильтра:

```
y = decimate(x, r, n, 'fir')
```

Для демонстрации прореживания сформируем сигнал, мгновенная частота которого меняется по гармоническому закону от нуля до частоты Найквиста и обратно в течение одной секунды (рис. 10.21):

```
>> Fs = 100e3;
>> t = 0:1/Fs:1;
>> s = cos(2*pi*t*Fs/4-Fs/4*sin(2*pi*t));
>> spectrogram(s, 256, [], [], Fs, 'yaxis')
>> colormap gray
```



**Рис. 10.21.** Спектрограмма прореживаемого сигнала

### ЗАМЕЧАНИЕ

О понятии мгновенной частоты речь шла в разд. "Угловая модуляция" главы 8. Функция `spectrogram`, которая показывает цветом зависимость мгновенного амплитудно-спектра от времени, была рассмотрена в главе 5.

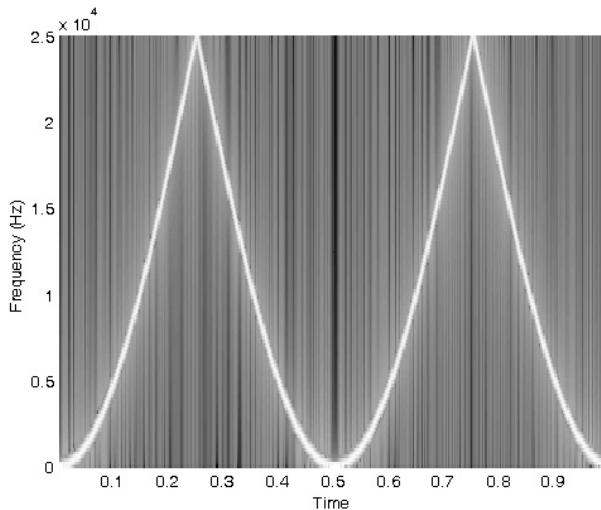
На спектрограмме сигнала видно, что в его спектре содержатся частоты, которые после прореживания с любым коэффициентом будут превышать новую частоту Найквиста, что неминуемо должно вызывать появление ложных частот.

Убедимся в этом, понизив частоту дискретизации в два раза путем выборки каждого второго отсчета с помощью рассмотренной ранее функции `downsample` (рис. 10.22):

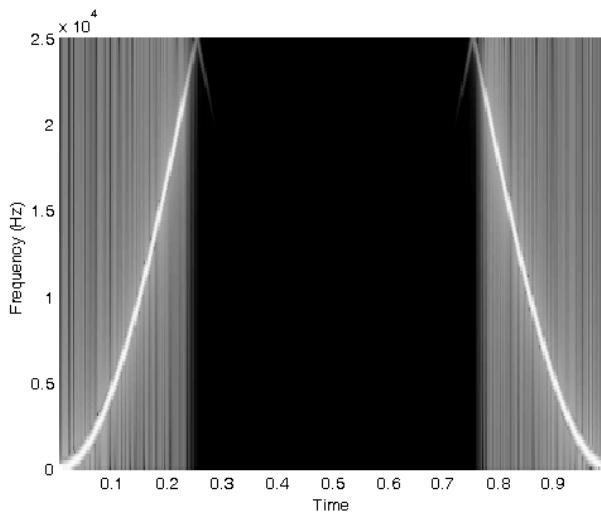
```
>> s1 = downsample(s, 2); % прореженный сигнал
>> spectrogram(s1, 256, [], [], Fs/2, 'yaxis')
>> colormap gray
```

Теперь выполним прореживание с предварительной фильтрацией, воспользовавшись для этого функцией `decimate` (рис. 10.23):

```
>> s2 = decimate(s, 2); % прореженный сигнал
>> spectrogram(s2, 256, [], [], Fs/2, 'yaxis')
>> colormap gray
```



**Рис. 10.22.** Прореживание без предварительной фильтрации создает ложные частоты



**Рис. 10.23.** Предварительная фильтрация позволяет устраниить ложные частоты при прореживании сигнала

Графики показывают, что при прореживании с предварительной фильтрацией оказалась удалена середина сигнала, содержащая высокие частоты. Однако в данном случае не произошло появления ложных частот.

## Функция интерполяции

Интерполяция в MATLAB выполняется с помощью функции `interp`. Синтаксис ее вызова следующий:

```
y = interp(x, r)
```

Здесь  $x$  — входной сигнал,  $r$  — целочисленный коэффициент повышения частоты дискретизации. Выходной параметр  $y$  — интерполированный сигнал.

Для интерполяции используется нерекурсивный фильтр с линейной ФЧХ, перекрывающий целое число межотсчетных интервалов. По умолчанию порядок фильтра равен  $8 \times r$ , т. е. фильтр в каждый момент использует 8 отсчетов входного сигнала. Частота среза фильтра по умолчанию равна половине исходной частоты Найквиста.

### **ВНИМАНИЕ!**

Обратите внимание на то, что по умолчанию функция `interp` считает, что полоса частот входного сигнала ограничена *половиной* частоты Найквиста.

Используя третий и четвертый входной параметры, можно управлять длиной фильтра и его частотой среза:

```
y = interp(x, r, 1, alpha)
```

Здесь 1 — половина количества отсчетов, используемых для интерполяции (длина фильтра равна  $2 \times r \times 1 + 1$ );  $\alpha$  — нормированная частота среза фильтра (значение 1 соответствует частоте Найквиста). По умолчанию  $1 = 4$  и  $\alpha = 0.5$ .

При указании двух выходных параметров во втором из них функция возвращает коэффициенты фильтра, использованного для интерполяции:

```
[y, b] = interp(...)
```

Рассмотрим пример интерполяции, сформировав для этого 13 отсчетов, принимающих значения  $\pm 1$ , и дополнив их с каждой стороны пятью нулями (рис. 10.24):

```
>> x=[0 0 0 0 0 1 1 1 1 -1 -1 1 1 -1 1 -1 1 0 0 0 0 0];
>> % повышаем частоту дискретизации в пять раз
>> y = interp(x, 5);
>> % график интерполированного сигнала - сплошная линия
>> plot((0:length(y)-1), y)
>> % график исходного сигнала - кружочки
>> hold on
>> plot((0:length(x)-1)*5, x, 'o')
>> hold off
```

Обратите внимание на большие выбросы, сформированные между отсчетами с одинаковыми значениями в процессе интерполяции. Эти выбросы обусловлены именно ограничением полосы частот выходного сигнала. Если соединить исходные отсчеты "более плавной" кривой, спектр полученного сигнала окажется шире. Таким образом, мы получили еще одно подтверждение факта, упоминавшегося при обсуждении теоремы Котельникова в главе 3: ограниченность полосы частот, занимаемой сигналом, вовсе не означает малости изменения значений сигнала между отсчетами.

### **ЗАМЕЧАНИЕ**

Данный сигнал называется 13-элементным *кодом Баркера*. Главным свойством кодов Баркера является вид их корреляционной функции, представляющей собой узкий вы-

сокий пик, окруженный треугольными лепестками малого уровня. Коды Баркера имеют максимально возможное для бинарных (т. е. принимающих два значения) сигналов отношение высоты центрального пика корреляционной функции к уровню ее боковых лепестков. Это отношение равно числу отсчетов сигнала и в данном примере составляет 13.

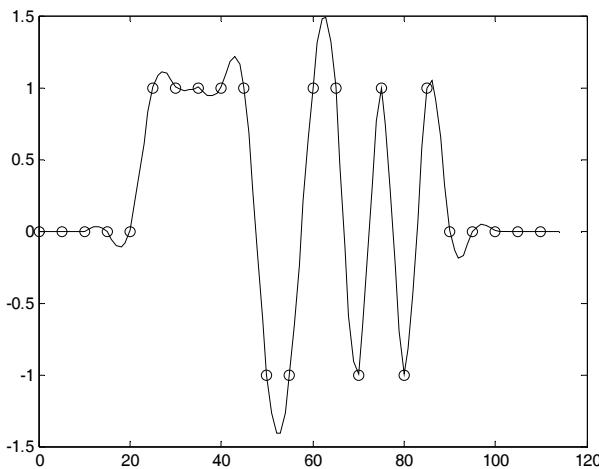


Рис. 10.24. Исходный (кружочки) и интерполированный (сплошная линия) сигналы при пятикратном повышении частоты дискретизации

## Функции передискретизации

Для выполнения передискретизации в MATLAB используется функция `resample`. Синтаксис ее вызова следующий:

```
у = resample(х, p, q)
```

Здесь `х` — входной сигнал, `p` и `q` — числитель и знаменатель дробного коэффициента изменения частоты дискретизации. Выходной параметр `у` — передискретизированный сигнал.

В качестве фильтра используется нерекурсивный ФНЧ, рассчитанный с помощью функции `firls` с использованием окна Кайзера.

### ЗАМЕЧАНИЕ

Подробнее об использовании функции `firls` для синтеза фильтров с линейной ФЧХ см. в главе 6; окно Кайзера, наряду с другими весовыми функциями, было рассмотрено в главе 5.

Ядром функции `resample` является функция `upfirdn` (*upsampling, FIR filtering and downsampling*), которая и выполняет приведенную ранее последовательность действий. Эта функция реализована в машинных кодах (в виде DLL-файла) для достижения максимальной производительности. Функцию `upfirdn` можно использовать

в тех случаях, когда при передискретизации необходимо задать конкретный ФНЧ. Синтаксис вызова функции `upfirdn` следующий:

```
y = upfirdn(x, h, p, q)
```

Здесь  $x$  — входной сигнал,  $p/q$  — коэффициент изменения частоты дискретизации,  $h$  — импульсная характеристика нерекурсивного ФНЧ. Выходной параметр  $y$  — передискретизированный сигнал.

Для осуществления блоковой обработки сигнала имеется вариант синтаксиса с доступом к внутреннему состоянию фильтра:

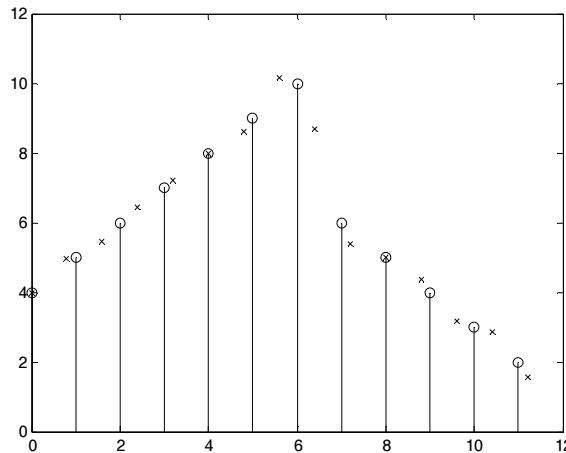
```
[y, z2] = upfirdn(x, h, p, q, z1)
```

Здесь  $z1$  — начальное состояние фильтра,  $z2$  — его конечное состояние. Длина вектора  $z1$  должна быть на единицу меньше длины импульсной характеристики фильтра  $h$ . Использование начального и конечного состояний фильтра для организации блоковой фильтрации обсуждалось в разд. "Доступ к внутреннему состоянию фильтра" главы 4.

Если один из параметров  $x$  и  $h$  является матрицей, эта матрица рассматривается как многоканальный сигнал или фильтр (каждый столбец матрицы соответствует одному каналу). Если оба параметра  $x$  и  $h$  являются матрицами с одинаковым числом столбцов, при обработке каждого сигнала используется свой фильтр. Выходной сигнал  $y$  во всех этих случаях является матрицей с тем же числом столбцов.

В качестве примера увеличим в  $5/4$  раз частоту кусочно-линейного сигнала, представляющего собой набор отсчетов со значениями

$$\{4, 5, 6, 7, 8, 9, 10, 6, 5, 4, 3, 2\}.$$



**Рис. 10.25.** Исходный (стебельки) и передискретизированный (крестики) сигналы при повышении частоты дискретизации в  $5/4$  раз

Результат показан на рис. 10.25:

```
>> x = [4:10 6:-1:2];
>> y = resample(x, 5, 4);
```

```
>> % график исходного сигнала – стебельки
>> stem(0:length(x)-1, x)
>> % график передискретизированного сигнала – крестики
>> hold on
>> plot((0:length(y)-1)*4/5, y, 'x')
>> hold off
```

На рис. 10.25 видно, что новые отсчеты действительно интерполируют исходный сигнал, однако наблюдается и эффект Гиббса (см. разд. "Примеры разложения сигналов в ряд Фурье" главы 1), хотя в данном случае он выражен слабо, поскольку частота дискретизации меняется незначительно.

# ПРИЛОЖЕНИЕ 1

## Основы работы с MATLAB

Система MATLAB (сокращение от MATrix LABoratory — матричная лаборатория) появилась в 1984 г. и за прошедшие двадцать лет стала мировым стандартом в области научных и технических расчетов. Основная причина этой популярности, вероятно, кроется в том, что MATLAB дал инженерам и ученым именно то, что им было нужно — возможность с непревзойденной легкостью применять к произвольным данным, представленным в виде векторов и матриц, разнообразнейшие численные алгоритмы. Удобный язык программирования, в котором благодаря матричной ориентации системы значительно уменьшилась необходимость в циклах, еще больше расширил сферу применения MATLAB.

В данном приложении приведены основные сведения о работе с MATLAB. Этой информации достаточно для того, чтобы познакомиться с системой и затем продолжить осваивать ее самостоятельно. За более подробными сведениями обращайтесь к книгам, целиком посвященным MATLAB (таким как [24]), а также, разумеется, к справочной системе и документации. Надеюсь также, что некоторую помощь в освоении MATLAB окажут многочисленные примеры расчетов, имеющиеся во всех главах книги.

Информация, приводимая в данном приложении, соответствует версии MATLAB 7.10 (Release 2010a).

### Установка

Прежде всего следует сказать несколько слов о системных требованиях. В целом можно сказать, что если на вашем компьютере работает система Windows XP/Vista/Windows 7, то должны работать и современные версии MATLAB. Однако некоторые ограничения все-таки имеются. Вот официальная информация о системных требованиях для Windows-платформ, взятая с сайта фирмы MathWorks:

- **Операционная система:** Windows XP SP3, Windows XP x64 SP2, Windows Server 2003 R2 SP2, Windows Vista SP1 или SP2, Windows Server 2008 SP2 или R2, Windows 7;
- **Процессор:** любой процессор семейства x86 производства Intel или AMD, поддерживающий дополнительный набор инструкций SSE2;

- **Оперативная память:** 1 Гбайт, рекомендуется не менее 2 Гбайт;
- **Дисковое пространство:** 1 Гбайт для ядра MATLAB, 3—4 Гбайт для "типичной установки" некоторого набора пакетов расширения.

Если вам приходилось устанавливать программное обеспечение в системе Windows, установка MATLAB не должна вызвать сложностей. Для работы с рассматриваемыми в книге примерами необходимо установить собственно MATLAB, а также четыре пакета расширения — Signal Processing Toolbox, Communications Toolbox, Filter Design Toolbox и Fixed-Point Toolbox. Остальные компоненты при работе с книгой не понадобятся, так что можете выбирать их, руководствуясь собственными профессиональными интересами и объемом свободного места на диске. Список компонентов с очень краткими описаниями их назначения приведен в *приложении 3*.

## Работа в интерактивном режиме

После запуска MATLAB на экране появится окно, показанное на рис. П1.1. MATLAB — интерактивная диалоговая система, поэтому большая часть ее главного окна предназначена для ввода команд и вывода результатов. Эта область называется *командным окном* — **Command Window**.

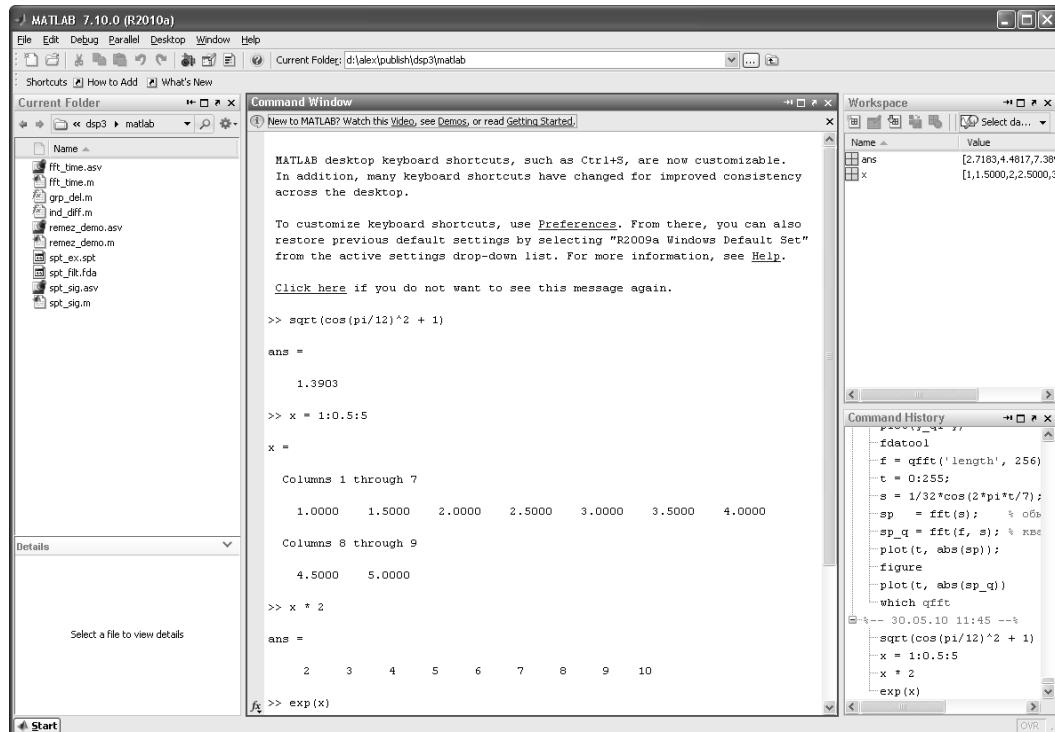


Рис. П1.1. Главное окно MATLAB

Попробуем вычислить значение какого-нибудь выражения, введя его в командном окне и нажав клавишу <Enter>:

```
>> sqrt(cos(pi/12)^2 + 1)
ans =
1.3903
```

Итак, MATLAB может работать как очень большой и мощный калькулятор, вычисляя значения математических выражений. При этом арифметические операции записываются традиционно, для возведения в степень используется знак ^, порядком вычислений можно управлять с помощью круглых скобок, и в круглых же скобках записываются аргументы вызываемых функций (в данном примере используются функции косинуса cos и квадратного корня sqrt).

На рис. П1.1 показано, как выглядят вводимые команды и выводимые результаты в окне MATLAB. При представлении результатов в книге пустые строки удалены для экономии места.

Поскольку мы не присвоили результат вычислений никакой переменной, MATLAB автоматически создал для него переменную с именем ans и показал ее значение.

Однако главная ценность MATLAB состоит в возможности одновременно оперировать несколькими значениями. Введем следующую команду:

```
>> x = 1:0.5:5
x =
Columns 1 through 7
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
Columns 8 through 9
    4.5000    5.0000
```

Мы создали переменную x, которая представляет собой *вектор-строку*, содержащую 9 элементов. Стоящее справа от знака равенства выражение генерирует арифметическую прогрессию, первый член которой равен 1, разность между соседними элементами равна 0,5, а последний член не превосходит 5.

### ЗАМЕЧАНИЕ

Имена *переменных* MATLAB чувствительны к регистру символов, так что x и X — это две разные переменные.

Над полученным вектором можно выполнять математические операции:

```
>> x * 2
ans =
     2      3      4      5      6      7      8      9      10
```

Можно применять к векторам математические функции:

```
>> exp(x)
ans =
Columns 1 through 7
    2.7183    4.4817    7.3891   12.1825   20.0855   33.1155   54.5982
```

```
Columns 8 through 9
90.0171 148.4132
```

Но вычисление следующего выражения дает неожиданный результат:

```
>> exp(x)/x
ans =
16.6289
```

А этот пример вообще вызовет появление сообщения об ошибке:

```
>> x * x
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Дело в том, что операции умножения, деления и возведения в степень в MATLAB выполняются не поэлементно, а по правилам матричных операций. Матричные правила запрещают умножение строки на строку, поэтому выдано сообщение о несовместимости размеров перемножаемых матриц.

### **ЗАМЕЧАНИЕ**

Несовместимость размеров матриц — вероятно, одна из наиболее часто встречающихся ошибок. Есть две главные причины ее возникновения. Во-первых, может оказаться неправильной *ориентация* какого-нибудь из используемых в выражении массивов (пример — попытка сложить вектор-строку и вектор-столбец). Во-вторых, какой-нибудь из промежуточных результатов, с вашей точки зрения являющийся *числом* (скаляром), может на самом деле оказаться *матрицей* — скажем, из-за ускользнувших от вашего внимания тонкостей работы вызываемой функции или из-за каких-то особенностей обрабатываемых данных.

Чтобы произвести *поэлементные* действия, к знаку операции необходимо добавить точку спереди (*.\**, *./* или *.^*). Теперь мы можем получить желаемые результаты, переписав два последних примера:

```
>> exp(x) ./ x
ans =
Columns 1 through 7
2.7183    2.9878    3.6945    4.8730    6.6952    9.4616   13.6495
Columns 8 through 9
20.0038   29.6826
>> x .* x
ans =
Columns 1 through 7
1.0000    2.2500    4.0000    6.2500    9.0000   12.2500   16.0000
Columns 8 through 9
20.2500   25.0000
```

Чтобы продемонстрировать именно *матричное* умножение, создадим второй вектор, причем не строку, а столбец:

```
>> y = x'
y =
1.0000
```

```
1.5000  
2.0000  
2.5000  
3.0000  
3.5000  
4.0000  
4.5000  
5.0000
```

Апостроф в MATLAB обозначает эрмитово сопряжение — сочетание транспонирования с комплексным сопряжением. В данном случае вектор  $x$  вещественный, поэтому комплексное сопряжение ничего не меняет. Для выполнения транспонирования без комплексного сопряжения к апострофу необходимо добавить точку (. $'$ ).

Теперь у нас есть вектор-строка  $x$  и вектор-столбец  $y$ , размерности которых позволяют перемножать их, причем в любом порядке:

```
>> x * y  
ans =  
    96  
>> y * x  
ans =  
Columns 1 through 7  
 1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000  
 1.5000    2.2500    3.0000    3.7500    4.5000    5.2500    6.0000  
 2.0000    3.0000    4.0000    5.0000    6.0000    7.0000    8.0000  
 2.5000    3.7500    5.0000    6.2500    7.5000    8.7500   10.0000  
 3.0000    4.5000    6.0000    7.5000    9.0000   10.5000   12.0000  
 3.5000    5.2500    7.0000    8.7500   10.5000   12.2500   14.0000  
 4.0000    6.0000    8.0000   10.0000   12.0000   14.0000   16.0000  
 4.5000    6.7500    9.0000   11.2500   13.5000   15.7500   18.0000  
 5.0000    7.5000   10.0000   12.5000   15.0000   17.5000   20.0000  
Columns 8 through 9  
 4.5000    5.0000  
 6.7500    7.5000  
 9.0000   10.0000  
11.2500   12.5000  
13.5000   15.0000  
15.7500   17.5000  
18.0000   20.0000  
20.2500   22.5000  
22.5000   25.0000
```

В соответствии с правилами матричных операций при умножении строки на столбец получилось число, а при умножении столбца на строку — квадратная матрица.

Векторы и матрицы можно формировать и из отдельных элементов, при этом элементы в строке разделяются пробелами или запятыми, между строками ставится точка с запятой, а все в целом ограничивается квадратными скобками:

```
>> M = [1 2;3,4]
M =
    1     2
    3     4
```

Точка с запятой имеет и еще одно важное применение — размещенная в конце строки, она подавляет вывод результата вычислений на экран:

```
>> M2 = M * M;
```

Узнать значение переменной можно, просто введя ее имя:

```
>> M2
M2 =
    7     10
   15     22
```

И в заключение первого знакомства приведем несложный пример, связанный с тематикой книги — обработкой сигналов. Для начала зададим частоту дискретизации  $F_s$  равной 8 кГц:

```
>> Fs = 8e3;
```

Теперь сформируем вектор значений времени, охватывающих с этой частотой дискретизации интервал длиной в одну секунду:

```
>> t = 0:1/Fs:1;
```

Рассчитаем для этих моментов времени значения периодической последовательности двуполярных прямоугольных импульсов с частотой следования  $f_0$ , равной 50 Гц. Для этого вычислим значения синусоиды нужной частоты и применим к ним знаковую функцию `sign`:

```
>> f0 = 50;
>> s = sign(sin(2*pi*f0*t));
```

Теперь подготовим фильтр для обработки нашего сигнала. Пусть это будет ФНЧ Баттерворта 4-го порядка с частотой среза  $f_1$ , равной 200 Гц. Такой фильтр рассчитывается с помощью функции `butter` из пакета расширения Signal Processing:

```
>> f1 = 200;
>> [b, a] = butter(4, f1*2/Fs)
b =
    1.0e-003 *
    0.0312    0.1250    0.1874    0.1250    0.0312
a =
    1.0000   -3.5897    4.8513   -2.9241    0.6630
```

Здесь мы впервые сталкиваемся с интересной особенностью MATLAB: в отличие от большинства языков программирования, функции MATLAB могут возвращать несколько результатов (в данном примере это векторы `b` и `a` — коэффициенты полиномов числителя и знаменателя функции передачи рассчитанного фильтра).

### ЗАМЕЧАНИЕ

Выходные параметры функции перечисляются через запятую в квадратных скобках. При использовании такой функции в составе математического выражения значением функции является *первый* из списка возвращаемых ею результатов.

Теперь можно обработать сигнал  $s$  фильтром, описываемым векторами  $b$  и  $a$ . Для этого предназначена функция `filter`:

```
>> s2 = filter(b, a, s);
```

Наконец, взглянем на полученные результаты, выведя графики первых 200 значений входного и выходного сигналов с помощью функции `plot`:

```
>> plot(t(1:200), s(1:200), t(1:200), s2(1:200), ':')
```

Результирующий график, который выводится в отдельном окне, называемом *графическим окном* (*figure window*), показан на рис. П1.2. Подробнее о функции `plot` рассказано далее в разд. "Графика" данного приложения.

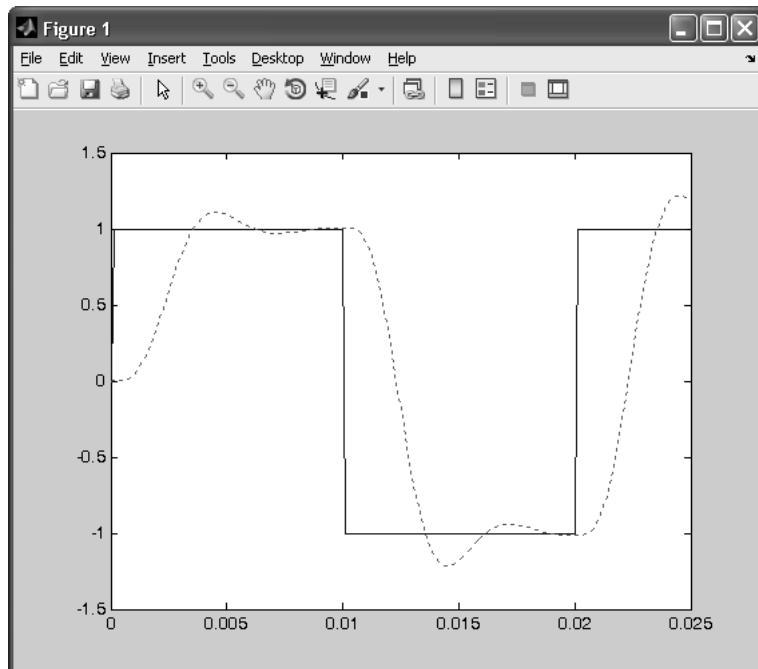


Рис. П1.2. Графики входного (сплошная линия) и выходного (пунктирная линия) сигналов дискретного фильтра

## Справочная система

В MATLAB имеется два уровня справочной системы: встроенная справка и HTML-документация.

Встроенная справка доступна всегда, даже в том случае, если вы отказались от установки документации, поскольку ее текст расположен непосредственно в фай-

лах функций MATLAB. Для получения справки о конкретной функции необходимо ввести команду `help имя_функции`, например:

```
>> help log2
LOG2    Base 2 logarithm and dissect floating point number.
Y = LOG2(X) is the base 2 logarithm of the elements of X.
```

`[F,E] = LOG2(X)` for each element of the real array `X`, returns an array `F` of real numbers, usually in the range  $0.5 \leq \text{abs}(F) < 1$ , and an array `E` of integers, so that  $X = F .* 2.^E$ . Any zeros in `X` produce `F = 0` and `E = 0`. This corresponds to the ANSI C function `frexp()` and the IEEE floating point standard function `logb()`.

See also [log](#), [log10](#), [pow2](#), [nextpow2](#), [realmax](#), [realmin](#).

Overloaded methods:

[codistributed/log2](#)  
[fints/log2](#)

Reference page in Help browser

[doc log2](#)

Для просмотра HTML-документации выберите одну из первых четырех команд меню **Help (Product Help, Function Browser, Using the Desktop или Using the Command Window)** или введите команду `helpdesk`. Появится окно, показанное на рис. П1.3.

В левой части окна имеется раздел **Help Navigator** с полем ввода поискового запроса и двумя вкладками — **Contents** (содержание) и **Search Results** (результаты поиска). На рис. П1.3 показано, как с помощью поискового запроса найдена справка по той же функции `log2`. Как видите, HTML-документация содержит больше информации, чем встроенная справка.

### ЗАМЕЧАНИЕ

Открыть страницу HTML-справки по конкретной функции можно, набрав в командной строке `doc имя_функции`.

На сайте [www.mathworks.com](http://www.mathworks.com) имеется также документация в формате PDF. Ее текст полностью совпадает с HTML-версией справки (к сожалению, дублируются и иногда встречающиеся в документации ошибки). Основное назначение PDF-варианта документации — обеспечить удобство распечатывания.

## Интерфейс главного окна

Говоря об интерфейсе главного окна MATLAB, следует сразу отметить главное — MATLAB был и остается системой командной строки, и все элементы интерфейса, которыми он обзавелся в современных версиях, являются лишь средством "мышиного" доступа к некоторым командам. Однако по-прежнему абсолютно все необходимые операции можно выполнять из командной строки.

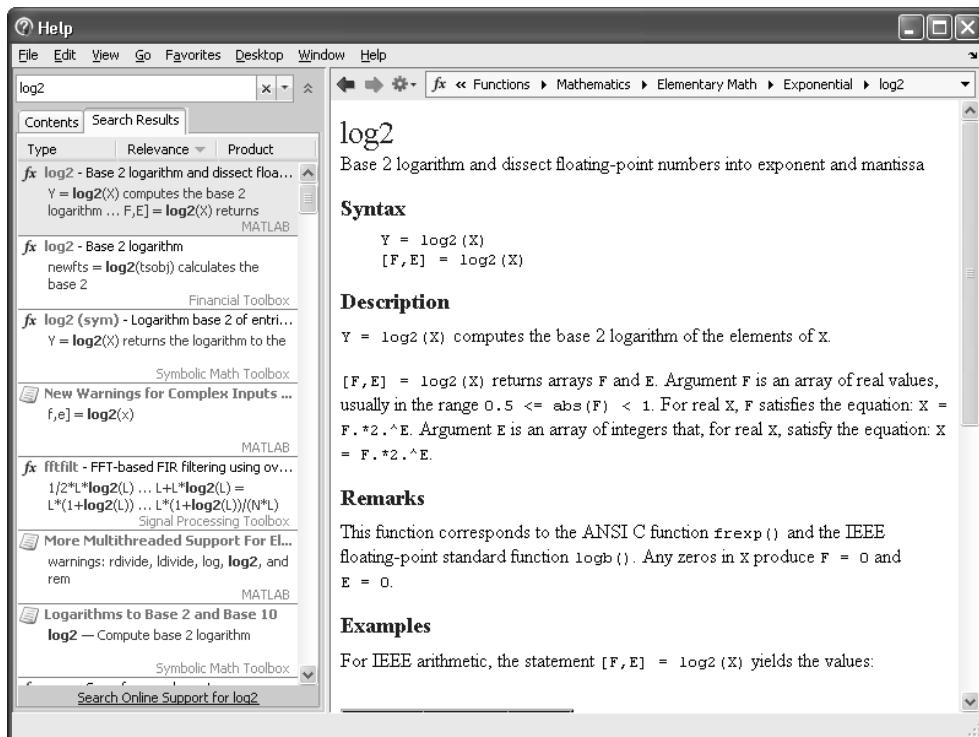


Рис. П1.3. Просмотр HTML-документации

На приведенном ранее рис. П1.1, кроме командного окна, видны три дополнительных окна — **Current Folder** (слева), **Workspace** и **Command History** (справа).

Очень удобным инструментом является окно **Command History** (см. рис. П1.1). Здесь представлен список вводившихся команд, причем этот список охватывает не только текущий сеанс работы, но и предыдущие. Команды из списка можно повторно выполнять (двойным щелчком мыши или нажатием клавиши <F9>) либо перетаскивать в командную строку для редактирования.

Еще один способ доступа к предыдущим использованным командам — их "перелистывание" в командной строке с помощью клавиш <↑> и <↓>.

Окно **Workspace** (см. рис. П1.1) демонстрирует содержимое рабочей области памяти MATLAB. По умолчанию здесь представлены имена переменных (**Name**), их значение (**Value**; для матричных переменных с большим числом элементов указывается только размер), а также значение минимального и максимального элементов (**Min**, **Max**). Состав отображаемых столбцов можно менять с помощью контекстного меню, вызываемого щелчком правой кнопкой мыши на строке заголовков таблицы. Двойной щелчок на имени переменной вызывает *визуальный редактор переменных* — **Variable Editor**.

Окно **Current Folder** (см. рис. П1.1) позволяет работать с файлами текущей папки — открывать их для редактирования, переименовывать, удалять, запускать MATLAB-программы и т. п.

Что касается панели инструментов главного окна MATLAB (см. рис. П1.1), отметим лишь раскрывающийся список **Current Folder**, в котором перечислены использовавшиеся рабочие папки. При работе с несколькими MATLAB-проектами этот список дает возможность удобного переключения между ними. Выбрать новую рабочую папку позволяет кнопка с многоточием, расположенная справа от списка.

## Массивы

В этом разделе мы познакомимся с массивами — основными типами данных MATLAB. Как уже говорилось, название системы MATLAB расшифровывается как "MATrix LABoratory". Для простоты можно считать, что все, с чем оперирует MATLAB, являются массивами, состоящими из комплексных чисел, вещественная и мнимая части которых представлены в 8-байтовом формате `double`. Даже обычное число (скаляр) с точки зрения MATLAB представляет собой матрицу размера  $1 \times 1$ .

### ЗАМЕЧАНИЕ

В настоящее время в MATLAB поддерживаются также целочисленные типы данных и формат с плавающей запятой одинарной точности (single precision, 4 байта).

MATLAB — интерпретирующая система, поэтому массивы не нужно как-то специально описывать. Размер массива может меняться в ходе работы — если вы присвоите значение элементу массива, указав индекс, превышающий текущий размер, MATLAB увеличит массив, дополнив его нулями. Уменьшить текущий размер массива можно только принудительно, удалив часть его элементов.

### ВНИМАНИЕ!

При каждом изменении размера массива MATLAB заново выделяет для него блок памяти. О том, к каким неприятностям это может привести, рассказано далее в разд. "Оптимизация MATLAB-программ" этого приложения.

Посмотрим, как меняется размер массива в процессе работы, набрав в главном окне MATLAB несколько команд.

Для начала создадим числовую переменную:

```
>> x = 5  
x =  
5
```

Как указывалось ранее, даже такая переменная `x` трактуется системой как массив. Убедимся в этом с помощью функции `size`, возвращающей вектор, содержащий количество строк и столбцов массива:

```
>> size(x)  
ans =  
1 1
```

### ЗАМЕЧАНИЕ

Кроме функции `size`, возвращающей вектор размеров массива по всем измерениям, имеется функция `length`, позволяющая определить *максимальный* из размеров мас-

сива. Особенно она удобна при определении длины одномерных векторов. Имеется также функция `numel`, возвращающая общее число элементов массива.

Итак, с точки зрения MATLAB `x` — это массив  $1 \times 1$ . Сделаем эту переменную "настоящим" массивом, присвоив значение, скажем, элементу, расположенному в пятом столбце четвертой строки:

```
>> x(4, 5) = 6
x =
    5     0     0     0     0
    0     0     0     0     0
    0     0     0     0     0
    0     0     0     0     6
```

### ЗАМЕЧАНИЕ

Нумерация элементов массивов в MATLAB начинается с единицы, как это принято при записи матриц в математике, а не с нуля, как, скажем, в языках программирования C/C++. При обращении к элементу массива индексы указываются через запятую в круглых скобках.

Как видите, переменная `x` стала массивом размером  $4 \times 5$ :

```
>> size(x)
ans =
    4      5
```

При попытке обратиться к элементу с индексами, выходящими за текущий размер массива, разумеется, выдается сообщение об ошибке:

```
>> x(2, 7)
??? Index exceeds matrix dimensions.
```

Чтобы уменьшить размер массива, нужно удалить из него некоторое количество строк или столбцов. Для этого нужно просто присвоить соответствующему фрагменту массива "пустое" значение `[]`:

```
>> x(2, :) = []
x =
    5     0     0     0     0
    0     0     0     0     0
    0     0     0     0     6
```

Здесь мы использовали в качестве второго индекса символ двоеточия `(:)`, что означает "все элементы массива вдоль данной размерности". Таким образом, мы получили доступ не к одному элементу массива, а к целому фрагменту. Возможность работы с фрагментами массивов, имеющаяся в MATLAB, отсутствует в большинстве языков программирования, поэтому ее стоит обсудить подробнее.

Для использования в дальнейших примерах сформируем квадратную матрицу `A` размером  $5 \times 5$  — "магический квадрат":

```
>> A = magic(5)
A =
 17    24     1     8    15
 23     5     7    14    16
  4     6    13    20    22
 10    12    19    21     3
 11    18    25     2     9
```

Для начала выделим из этого массива прямоугольный блок, простирающийся от второй до третьей строки и от второго до четвертого столбца:

```
>> A(2:3, 2:4)
ans =
  5     7    14
  6    13    20
```

Таким образом можно выделить из массива произвольный прямоугольный блок. Если выделяемый блок по одному из измерений совпадает с размером исходного массива, вместо задания диапазона индексов можно указать просто двоеточие (мы уже сталкивались с этим при удалении фрагмента массива):

```
>> A(3:4, :)
ans =
  4     6    13    20    22
 10    12    19    21     3
```

Некоторые отступления от общего синтаксиса возможны и тогда, когда выделяемый фрагмент "упирается" в границу массива справа или снизу. В этом случае для обозначения последнего (максимального) возможного значения индекса по какой-либо размерности можно использовать ключевое слово `end`. Это значение можно использовать и при вычислении индексов. Например, выделим из массива `A` блок размером  $2 \times 3$ , расположенный в правом нижнем углу:

```
>> A(end-1:end, end-2:end)
ans =
  19    21     3
  25     2     9
```

Выделяемый фрагмент не обязательно должен быть сплошным. Выделим из массива `A` второй и четвертый элементы первой и последней строк:

```
>> A([1 end], [2 4])
ans =
  24     8
  18     2
```

Наконец, для выделения из массива абсолютно произвольного набора элементов придется прибегнуть к обращению по линейному (т. е. одномерному) индексу. Элементы двумерных массивов при этом нумеруются по столбцам. Выделим из массива `A` элементы, лежащие на кросс-диагонали:

```
>> A([5 9 13 17 21])
ans =
    11    12    13    14    15
```

Для автоматического формирования одномерных индексов из двумерных предназначена функция `sub2ind`. Применим ее для получения одномерных индексов элементов массива `A`, лежащих на его кросс-диагонали:

```
>> x = 1:5;      % номера столбцов
>> y = 5:-1:1;  % номера строк
>> ind = sub2ind(size(A), y, x)
ind =
    5    9   13   17   21
```

Полученные значения фрагментов массивов можно не только сохранять в виде новых переменных, с ними можно производить вычисления, так сказать, "на месте". В качестве примера увеличим в два раза значения элементов первого столбца массива `A`:

```
>> A(:, 1) = A(:, 1)*2
A =
    34    24     1     8    15
    46     5     7    14    16
     8     6    13    20    22
    20    12    19    21     3
    22    18    25     2     9
```

Мы рассмотрели лишь основные приемы работы с массивами в MATLAB. Имеется еще множество функций, реализующих преобразования массивов, матричные операции и т. д. Часть этих функций перечислена в *приложении 2*, а более подробная информация содержится в справочной системе.

## Другие типы данных

Помимо одномерных векторов и двумерных матриц MATLAB поддерживает ряд других типов данных. К ним относятся многомерные массивы, строки, структуры, массивы ячеек, а также *объекты*. Подробное рассмотрение реализации объектно-ориентированных концепций в MATLAB выходит за рамки этого краткого описания, хотя минимальные сведения о работе с объектами приведены в *главе 4* при обсуждении объектов дискретных фильтров, реализованных в пакете Signal Processing. Остальные перечисленные типы данных рассматриваются далее.

## Многомерные массивы

Многомерные массивы являются естественным обобщением двумерных массивов. Для обращения к их элементам используется необходимое число индексов, однако заполнять многомерный массив числами и выражениями приходится по частям,

т. к. синтаксис MATLAB позволяет записывать поэлементное содержимое только для одно- и двумерных массивов. В качестве примера создадим трехмерный массив размером  $2 \times 2 \times 2$ :

```
>> x(:,:,:,1) = [1 2;3 4]; % первый "слой"
>> x(:,:,:,2) = [5 6;7 8]; % второй "слой"
>> x
x(:,:,:,1) =
    1     2
    3     4
x(:,:,:,2) =
    5     6
    7     8
```

Как видите, вывод значений многомерного массива MATLAB тоже осуществляет по "слоям".

Многие функции MATLAB, ориентированные на работу с *векторами*, при получении входного параметра в виде *двумерного массива* обрабатывают его столбцы независимо. При использовании в качестве параметра *многомерного массива* такая обработка будет вестись вдоль первого измерения, размер массива вдоль которого не равен единице. Например, для созданного в приведенном примере трехмерного массива были бы независимо обработаны двухэлементные векторы  $x(:,:,1)$ ,  $x(:,:,2)$ ,  $x(:,:,1,2)$  и  $x(:,:,2,2)$ . Кроме того, многие такие функции позволяют явно указать, вдоль какой размерности следует выбирать векторы из многомерного массива. Соответствующий параметр в документации обычно обозначается идентификатором *dim*. В качестве примера можно привести функции быстрого преобразования Фурье *fft* и *ifft*, имеющие следующий вариант синтаксиса:  $y = fft(x, n, dim)$  и  $x = ifft(y, n, dim)$ .

Имеются и функции, специально предназначенные для обработки многомерных данных. Обычно идентификаторы таких функций отличаются от имен их одномерных родственников добавленной в конце буквой "n". Примером опять-таки могут служить функции быстрого преобразования Фурье, у которых имеются многомерные варианты *fftn* и *ifftn*.

## Строки

Для задания строковых констант используются апострофы:

```
>> s1 = 'Digital';
>> s2 = 'Signal';
>> s3 = 'Processing';
```

С точки зрения MATLAB строки представляют собой *массивы символов*. Поэтому, например, операция соединения (конкатенации) строк выполняется не с помощью оператора сложения, а с использованием синтаксиса горизонтального соединения матриц:

```
>> s4 = [s1 ' ' s2 ' ' s3]
s4 =
Digital Signal Processing
```

Основные функции работы со строками перечислены в разд. "Strfun" приложения 2.

## Структуры

Структуры (*structure*) позволяют объединить в одной переменной разнородные данные (*поля* — *field*) и осуществлять доступ к ним по именам. В языке C соответствующий тип данных тоже называется структурой, а в языке Pascal — записью (*record*).

В MATLAB для создания структуры не требуется специальных объявлений — достаточно присвоить значение какому-либо полю. Имена полей указываются после имени переменной через точку. Например, для хранения информации, извлеченной из WAV-файла, можно использовать следующую структуру:

```
>> w.name = 'c:\windows\media\tada.wav'; % имя файла
>> w.Fs = 22050; % частота дискретизации
>> w.bits = 16; % число бит на отсчет
>> w.channels = 2; % число каналов
>> w.samples = 42752; % число отсчетов
>> w.data = wavread(w.name); % двумерный массив отсчетов
```

MATLAB показывает значения переменных-структур следующим образом:

```
>> w
w =
    name: 'c:\windows\media\tada.wav'
    Fs: 22050
    bits: 16
    channels: 2
    samples: 42752
    data: [42752x2 double]
```

Значения полей, имеющих короткие представления, выводятся полностью, а для больших массивов указываются только размеры и тип данных.

Можно создавать и массивы структур, для этого, как обычно, используются индексы. Превратим переменную *w* в двухэлементный массив структур:

```
>> w(2).name = 'c:\windows\media\chord.wav'
w =
1x2 struct array with fields:
  name
  Fs
  bits
  channels
  samples
  data
```

Для массивов структур MATLAB показывает только размеры массива и список полей.

## Массивы ячеек

Массивы ячеек (*cell array*), так же как и структуры, позволяют объединить в одной переменной разнородные данные. Однако обращение к этим данным производится не по именам полей, а по числовым индексам. Чтобы отличить массив ячеек от обычного массива, его индексы записываются не в круглых скобках, а в фигурных. В качестве примера представим ту же информацию из WAV-файла, которую мы использовали при рассмотрении структур, в виде массива ячеек:

```
>> w{1} = 'c:\windows\media\tada.wav'; % имя файла
>> w{2} = 22050; % частота дискретизации
>> w{3} = 16; % число бит на отсчет
>> w{4} = 2; % число каналов
>> w{5} = 42752; % число отсчетов
>> w{6} = wavread(w{1}); % двумерный массив отсчетов
```

Приведенный пример не имеет особого практического смысла — в данном случае значительно удобнее был бы доступ по именам полей. Однако массивы ячеек удобны в тех случаях, когда нужно создать массив из векторов разной длины или матриц разного размера. Примером может служить хранение в массиве ячеек информации о фильтре, составленном из последовательно включенных секций разного порядка.

MATLAB показывает содержимое массивов ячеек следующим образом:

```
>> w
w =
[1x25 char] [22050] [16] [2] [42752] [42752x2 double]
```

Как и в случае структур, значения элементов, имеющих короткие представления, выводятся полностью, а для больших массивов указываются только размеры и тип данных. В данном случае "слишком длинной" для вывода оказалась и текстовая строка.

## Программирование

Интерактивная работа с MATLAB в режиме командной строки очень удобна при освоении системы и при необходимости поэкспериментировать, чтобы испытать разные подходы к анализу и обработке имеющихся данных. Однако по мере приобретения опыта, а также при решении конкретной задачи вы заметите, что очень часто набираете одни и те же последовательности команд. Это означает, что настало время превратить такие последовательности в программы (сценарии) и функции MATLAB.

Эти программы и функции представляют собой текстовые (ASCII-) файлы с расширением .m, в которых записаны команды и операторы MATLAB. В принципе такие

файлы могут создаваться и модифицироваться с помощью любого ASCII-редактора, но, начиная с версии 5, в MATLAB появился свой редактор, обеспечивающий, в частности, цветовое выделение синтаксических элементов языка, а также интегрированный с отладчиком. О работе с редактором речь пойдет далее, а сначала следует рассмотреть различия между программами и функциями MATLAB.

## Программы и функции

Как уже говорилось, текстовые файлы, содержащие операторы MATLAB, могут быть двух типов: *программы* (*script*; иногда используется дословный перевод — "сценарий") и *функции* (*function*). Между ними имеются следующие различия:

- функции имеют заголовок `function`, а программы — заголовок `script` или вовсе не имеют заголовка;
- функции могут принимать входные параметры и возвращать результаты вычислений, а программы — нет;
- программы используют рабочую область памяти среды MATLAB, а каждая функция при вызове создает свою собственную рабочую область памяти и общается с внешним миром только через параметры и глобальные переменные (см. далее).

Итак, как уже было сказано, программы и функции MATLAB представляют собой текстовые файлы с записью команд, операторов и вызовов функций MATLAB. В этих файлах могут содержаться любые команды MATLAB, в том числе те, которые обычно не применяются в командной строке (операторы циклов, условные операторы и т. п.); использование этих средств в командной строке возможно, но приводит к необходимости набирать много текста, так что удобнее оказывается хранить такие фрагменты в виде файлов программ).

В одной строке можно записывать несколько операторов. Для их разделения следует использовать точку с запятой (если необходимо подавить вывод результатов вычислений в командное окно) или запятую (если этот вывод нужно разрешить).

Длинные строки для улучшения читаемости программы можно делить на части. Признаком того, что запись оператора продолжается на следующей строке, являются три точки в конце строки. Удобно использовать эту возможность, например, для более наглядной записи поэлементно заданной матрицы:

```
M = [a b c d; ...
      b b c d; ...
      c c c d; ...
      d d d d];
```

Хорошая программа невозможна без комментариев. Для обозначения комментариев в MATLAB используется знак `%` — комментарием считается часть строки справа от него. Можно также создавать многострочные комментарии, ограничивая их строками `%{` и `%}`. При необходимости закомментировать большой фрагмент текста или, наоборот, снять с него знаки комментария можно воспользовавшись соот-

ветственно командами **Comment** и **Uncomment** меню **Text** редактора/отладчика (см. далее).

## Редактор/отладчик М-файлов

Для запуска редактора/отладчика М-файлов необходимо создать новый М-файл или открыть существующий. Для этого используются команды **New | Script**, **New | Function** и **Open** меню **File** главного окна MATLAB или соответствующие кнопки панели инструментов. Окно редактора/отладчика показано на рис. П1.4.

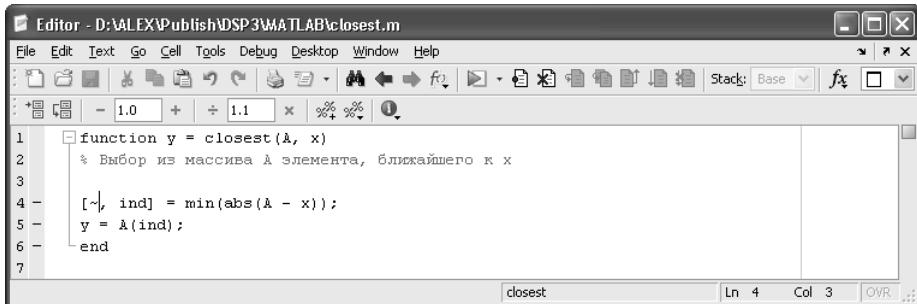


Рис. П1.4. Редактор/отладчик М-файлов

Функции редактирования и работы с файлами, обеспечиваемые редактором/отладчиком М-файлов, являются типичными для текстовых редакторов, поэтому мы не будем обсуждать их. Из специфических возможностей следует отметить цветовое выделение синтаксических элементов языка программирования MATLAB, а также команды меню **Text**, позволяющие закомментировать выделенный фрагмент текста (**Comment**), убрать знаки комментария (**Uncomment**), а также автоматически отформатировать левый край выделенного фрагмента текста (**Smart Indent**).

Правая часть панели инструментов содержит кнопки управления отладчиком. Эти функции редактора/отладчика будут рассмотрены далее в разд. "Работа с отладчиком" этого приложения.

## Создание функций

Текст М-файла функции должен начинаться с заголовка **function**, имеющего следующий вид:

```
function [y1, y2, ...] = fname(x1, x2, ...)
```

Здесь **fname** — имя функции, **x1**, **x2** и т. д. — входные параметры, **y1**, **y2** и т. д. — выходные параметры. Входные и/или выходные параметры могут отсутствовать.

### **ВНИМАНИЕ!**

На самом деле имя функции определяется не строкой **fname**, а именем, под которым сохранен М-файл. По этой же причине имена функций, в отличие от имен перемен-

ных, в системе Windows не чувствительны к регистру символов (но при несовпадении регистра выдается предупреждение). Однако это относится только к функциям, существующим в виде М-файлов — имена функций, встроенных в ядро MATLAB, должны записываться только строчными буквами.

В качестве примера создадим функцию `closest`, которая будет принимать два входных параметра, вектор и число, и выбирать из вектора значение, максимально близкое ко второму (числовому) параметру. Для этого откроем окно редактора/отладчика командой меню **File | New | Function**. В окне появится шаблон функции:

```
function [ output_args ] = Untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

end
```

Отредактируйте шаблон, чтобы код принял следующий вид:

```
function y = closest(A, x)
% Выбор из массива A элемента, ближайшего к x

[~, ind] = min(abs(A - x));
y = A(ind);
end
```

Введя текст, сохраните файл под именем `closest.m`.

Теперь кратко прокомментируем этот программный код. Первая строка — это заголовок функции, составленный согласно приведенному ранее шаблону. Вторая строка, являющаяся комментарием, — это встроенная справка об использовании нашей функции. По команде `help fname` в командное окно выводятся строки комментария, расположенные в файле `fname.m` непосредственно после заголовка функции. Проверим это:

```
>> help closest
Выбор из массива A элемента, ближайшего к x
```

Таким образом, создавать встроенную справку для собственных функций совсем несложно.

Далее с помощью функции `min` определяется номер интересующего нас элемента, и в последней строке выходному параметру `y` присваивается значение этого элемента.

### ЗАМЕЧАНИЕ

Символ `~` (тильда), использованный в качестве первого выходного параметра функции `min`, означает, что этот первый результат (собственно минимальное значение разности) нам не нужен. Возможность не создавать в таких случаях переменные, не нужные в дальнейшем, появилась, начиная с выпуска R2009b. Также следует отметить, что слово `end` в конце текста функции является обязательным только при размещении в том же файле дополнительных субфункций (см. далее).

Переменная `ind`, создаваемая внутри функции, является *локальной*. Такие переменные создаются внутри функции и уничтожаются при завершении ее работы. Если переменные с такими же именами существовали в рабочей области памяти MATLAB до вызова функции, их значения не изменятся.

Функция общается с внешним миром только через свои параметры и *глобальные переменные*. Чтобы объявить переменную `x` глобальной, необходимо использовать в головной программе и во *всех* функциях, которые должны иметь доступ к данной переменной, следующую директиву:

```
global x
```

Возврат из функции производится по достижении конца ее файла. Для досрочного возврата можно использовать оператор `return`. В любом случае результатами работы функции являются значения, которые выходные параметры имели на момент возврата.

### **ЗАМЕЧАНИЕ**

Присвоить некоторое значение локальной переменной вызывающей функции или переменной, находящейся в рабочей области памяти MATLAB, позволяет функция `assignin`.

## **Субфункции и вложенные функции**

В конце М-файла, после завершения кода функции, можно добавить описания других функций. Они называются *субфункциями* (*subfunction*) и доступны только для функций из этого файла. Обмен информацией с субфункциями производится обычными методами — через входные и выходные параметры и глобальные переменные. Начиная с версии MATLAB 7.0 появилась также возможность создавать *вложенные функции* (*nested function*). Код вложенной функции располагается *внутри* кода другой функции, при этом вложенная функция имеет доступ к локальным переменным, созданным во внешней функции.

## **Путь поиска**

Чтобы программа или функция была доступна из среды MATLAB, система должна быть способна найти соответствующий М-файл. Поиск файлов осуществляется следующим образом: сначала просматривается текущая рабочая папка (ее имя показано в панели инструментов главного окна, см. рис. П1.1), а затем папки, входящие в *путь поиска* (MATLAB *search path*). Когда пользователь начинает создавать собственные М-файлы, встает вопрос о том, где их целесообразно размещать. На этот счет можно дать следующую рекомендацию: файлы, относящиеся к разным проектам, следует размещать в отдельных папках, *не добавляя* эти папки в путь поиска. Выпадающий список **Current Folder** панели инструментов главного окна обеспечивает удобство переключения между папками разных проектов. Если же в процессе работы у вас начинает формироваться собственная библиотека функций, используемых сразу в *нескольких* проектах, имеет смысл собрать их в одну папку и

добавить ее имя в путь поиска (для вызова редактора пути поиска используется команда **Set Path** меню **File** главного окна MATLAB). Тогда эти функции будут доступны в любой момент, вне зависимости от того, какая папка является текущей. Следует, однако, помнить о том, что при наличии (в разных папках) нескольких функций с одинаковым именем будет вызвана та из них, которая будет найдена раньше. Поэтому такой ситуации следует избегать, проверяя при выборе имен для собственных функций, нет ли функций с такими именами в самом MATLAB или его пакетах расширения.

## Логические условия

В качестве логических условий в условных операторах и циклах `while` (см. далее) могут использоваться числовые скалярные значения. При этом нулевое значение трактуется как "ложь" (`false`), а любое ненулевое — как "истина" (`true`).

Кроме того, для формирования условий часто используются операторы сравнения `==` (равно), `~=` (не равно), `<` (меньше), `>` (больше), `<=` (меньше или равно), `>=` (больше или равно). Следует иметь в виду, что для массивов они производят *поэлементное* сравнение, возвращая массив такого же размера, как сравниваемые аргументы. Результатирующий массив содержит единицы там, где сравнение элементов дало выполнение условия, и нули там, где условие сравнения выполнено не было. Эти единицы и нули имеют тип данных `logical`, и для использования полученных результатов в математических операциях следует преобразовать их к числовому типу функцией `double`.

Для формирования логических условий полезны также следующие функции:

- `all(x)` — возвращает 1, если все элементы `x` отличны от нуля;
- `any(x)` — возвращает 1, если хотя бы один элемент `x` отличен от нуля;
- `isequal(x, y)` — возвращает 1, если значения `x` и `y` совпадают. В отличие от конструкции `all(x==y)`, использование данной функции не приведет к ошибке, если размеры `x` и `y` не совпадают;
- `isempty(x)` — возвращает 1, если `x` является пустой матрицей (т. е. имеет размер `0x0`).

Имеется еще несколько десятков функций проверки разнообразных условий, имена которых имеют вид `is...`. За более подробной информацией обратитесь к справочной системе.

## Условный оператор

Условный оператор в MATLAB реализуется с помощью ключевых слов `if`, `else` и `end`:

```
if cond
    % ветвь, выполняемая, если cond истинно
    ...

```

```

else
    % ветвь, выполняемая, если cond должно
    ...
end

```

Здесь `cond` — проверяемое значение (см. ранее разд. "Логические условия" этого приложения).

Для оператора `if`, вложенного в ветвь `else` другого (внешнего) оператора `if`, есть сокращенная форма записи с использованием ключевого слова `elseif`:

```

if cond1
    % ветвь, выполняемая, если cond1 истинно
    ...
elseif cond2
    % ветвь, выполняемая, если cond1 ложно, а cond2 истинно
    ...
else
    % ветвь, выполняемая, если cond1 и cond2 ложны
    ...
end

```

Аналогичным образом можно использовать `elseif` несколько раз.

## Оператор выбора

Если в условном операторе выбирается одна из двух возможных альтернатив, то оператор выбора реализует разветвление программы на несколько путей. Выбор конкретного пути зависит от значения заданной переменной. Запись оператора выбора в MATLAB производится с помощью ключевых слов `switch`, `case`, `otherwise` и `end`:

```

switch x
    case x1
        % ветвь, выполняемая, если x равно x1
        ...
    case x2
        % ветвь, выполняемая, если x равно x2
        ...
    otherwise
        % ветвь, выполняемая, если все условия case не соблюдаются
        ...
end

```

При выполнении оператора `switch` значение `x` поочередно сравнивается с `x1`, `x2` и т. д. При обнаружении первого совпадения выполняются операторы соответствующей ветви, после чего производится выход из оператора выбора. Таким образом, в отличие от аналогичного оператора языка C, "проваливания" на следующую ветвь не происходит и операторы `break` в конце каждой ветви в MATLAB не нужны.

## Циклы

Для реализации циклов в MATLAB имеется два оператора — `for` и `while`. В операторе `for` переменная-счетчик цикла поочередно принимает значения элементов некоторого вектора:

```
for k = x  
    % тело цикла  
    ...  
end
```

Операторы, входящие в тело цикла, будут выполняться при значении переменной `k`, равном `x(1)`, затем `x(2)` и т. д. до `x(end)`. После завершения цикла значение `k` остается равным `x(end)`.

Разумеется, чаще всего в цикле `for` используется последовательный перебор целочисленных значений счетчика цикла:

```
for k = 1:N
```

Однако при необходимости можно использовать дробное значение шага

```
for k = -1:0.01:1
```

или перебор произвольных значений:

```
for k = [1 10 pi 9 ln(3)]
```

### ВНИМАНИЕ!

Изучая программирование, люди, как правило, приобретают привычку использовать в качестве счетчиков циклов переменные с именами `i` и `j`. В MATLAB это чревато неприятностями, поскольку по умолчанию эти переменные имеют значение мнимой единицы. Использование переменной `i` или `j` одновременно в качестве счетчика цикла и в качестве мнимой единицы — это те "грабли", на которые рано или поздно наступают очень многие пользователи MATLAB. Чтобы избежать неприятностей, рекомендуется в качестве мнимой единицы использовать не *переменную i (j)*, а *константу 1i (1j)*.

Второй тип циклов реализуется с помощью оператора `while`. Тело цикла выполняется, пока условие `cond` остается истинным (т. е. значение `cond` отлично от нуля):

```
while cond  
    % тело цикла  
    ...  
end
```

Если `cond` изначально имеет нулевое значение, тело цикла не будет выполнено ни разу.

Чтобы реализовать досрочное завершение цикла, используется оператор `break`. Разумеется, он должен применяться в сочетании с условным оператором, например, так:

```

for k = 1:N
    % первая половина тела цикла
    ...
    % проверка дополнительного условия завершения
    if cond
        break
    end
    % вторая половина тела цикла
    ...
end

```

Оператор `break` можно использовать для реализации циклов с проверкой условия завершения не в начале (как делает оператор `while`), а в произвольном месте цикла. Для этого с помощью оператора `while` создается "вечный" цикл, а его завершение производится оператором `break`:

```

while 1 % "вечный" цикл
    % первая половина тела цикла
    ...
    % проверка условия завершения
    if cond
        break
    end
    % вторая половина тела цикла
    ...
end

```

Кроме оператора `break` для управления выполнением программы внутри цикла имеется оператор `continue`. Размещенный внутри цикла `for` или `while`, этот оператор передает управление на проверку условия завершения цикла, пропуская при этом оставшийся фрагмент тела цикла. Это позволяет сделать код более наглядным, избежав использования длинного оператора `if`.

## Функции с переменным числом параметров

При вызове функции можно использовать меньшее количество входных и выходных параметров, чем указано в заголовке функции. При этом рассчитанные функцией "лишние" выходные параметры просто игнорируются. Что касается неуказанных входных параметров, то функция выдаст сообщение об ошибке, когда попытается обратиться к одному из них. Однако внутри функции доступны сведения об использованном при вызове числе входных и выходных параметров, что позволяет в зависимости от этого управлять алгоритмом работы функции.

Число переданных входных параметров можно узнать с помощью функции `nargin`, а число ожидаемых выходных параметров — с помощью функции `nargout`. В простейшем случае функция `nargin` может использоваться для задания входным параметрам значений по умолчанию:

```
function y = nargin_demo(x, N, mu, fid)
if nargin < 4
    fid = 1;
end
if nargin < 3
    mu = 0.2;
end
if nargin < 2
    N = 256;
end
...
...
```

При вызове такой функции можно опускать один, два или три последних входных параметра, аналогично тому, как это позволяют делать язык C++ и последние версии языка Object Pascal:

```
y = nargin_demo(x, N, mu, fid) % полный вариант
y = nargin_demo(x, N, mu)
y = nargin_demo(x, N)
y = nargin_demo(x)
```

Вместо опущенных параметров будут использованы заданные в тексте функции значения по умолчанию.

Аналогичным образом можно использовать и функцию `nargout` — например, чтобы сэкономить время, не вычисляя незатребованные выходные величины. Можно использовать функции `nargin` и `nargout` и для более серьезного управления логикой работы — скажем, менять используемый алгоритм в зависимости от числа входных и выходных параметров.

## Ввод и вывод данных

Как мы уже знаем, простейшим способом вывода значения переменной в командное окно MATLAB является указание ее имени без точки с запятой. Аналогичное действие производит функция `disp(name)`, отличие заключается только в том, что на экран не будет выводиться идентификатор переменной `name`.

## Форматированный вывод

Чтобы выводить в командное окно информацию в более аккуратном виде, следует воспользоваться функциями форматированного вывода. Это функции `sprintf` (формирование строковой переменной с форматированным текстом) и `fprintf` (вывод форматированного текста в поток вывода). Синтаксис обеих функций очень схож и практически не отличается от синтаксиса соответствующих функций языка C. Поэтому кратко рассмотрим лишь функцию `fprintf`:

```
fprintf(fid, 'format', x1, x2, ...)
```

Здесь `fid` — числовой идентификатор потока вывода. При нулевом значении вывод не производится, значение 1 соответствует стандартному потоку вывода (вывод в командное окно), значение 2 — стандартному потоку вывода сообщений об ошиб-

ках (тоже вывод в командное окно, но красным цветом). Для записи информации в файл в качестве `fid` необходимо использовать значение, полученное от функции `fopen` при открытии файла (работа с файловыми потоками в данном приложении не рассматривается). Данный параметр можно опустить, тогда по умолчанию вывод будет производиться в командное окно.

Строковый параметр '`format`' задает формат вывода. Эта строка должна содержать фиксированный выводимый текст и *спецификаторы формата*, на место которых будут подставлены значения переменных `x1`, `x2` и т. д. Спецификаторы формата начинаются со знака `%`, за которым следуют цифры, задающие число символов для вывода, и буква, определяющая тип формата. Полная информация о возможных спецификаторах формата имеется в HTML-справке, здесь же приведем лишь несколько конкретных примеров:

- `%d` — вывод целого числа, количество позиций выбирается автоматически;
- `%.3f` — вывод числа в формате с фиксированной запятой, после запятой выводится три десятичных знака, количество позиций для целой части выбирается автоматически;
- `%-15.3e` — вывод числа в экспоненциальном формате, мантисса имеет три десятичных знака после запятой, поле вывода занимает 15 символов, текст выравнивается по левому краю поля.

Для вывода специальных символов можно использовать комбинации `\n` (перевод строки), `\r` (возврат каретки), `\t` (табуляция), `\b` (забой), `\\"` (обратная косая черта), `%%` (символ процента), `'` (одиночный апостроф).

Теперь используем перечисленные спецификаторы и некоторые из символов:

```
>> n = 15;
>> err = 128.3567;
>> value = 1234567890;
>> fprintf(1, 'Шаг %d, ошибка %.3f \nx' '= %-15.3e м\n', n, err, value)
Шаг 15, ошибка 128.357
x'= 1.235e+009      м
```

## Сохранение и загрузка значений переменных

MATLAB позволяет сохранять переменные в виде МАТ-файлов (они имеют расширение `.mat`), которые затем могут быть снова загружены. Для сохранения всего содержимого рабочей области памяти используется команда

```
>> save filename
```

Все существующие в данный момент переменные будут записаны в файл `filename.mat`, расположенный в текущей папке.

Чтобы сохранить значения только некоторых переменных, следует указать их имена после имени файла:

```
>> save filename x y asd ...
```

В файл `filename.mat` будут записаны только значения переменных `x`, `y`, `asd` и т. д.

Загрузка сохраненных переменных производится командой `load`:

```
>> load filename
```

Файл `filename.mat` должен быть доступен, т. е. он должен находиться в текущей папке или в одной из папок, внесенных в список для поиска файлов (см. ранее разд. "Путь поиска" этого приложения).

## Работа с отладчиком

Отладочные средства собраны в правой части панели инструментов окна редактора/отладчика (рис. П1.5), а также в меню **Debug** и **Breakpoints**.

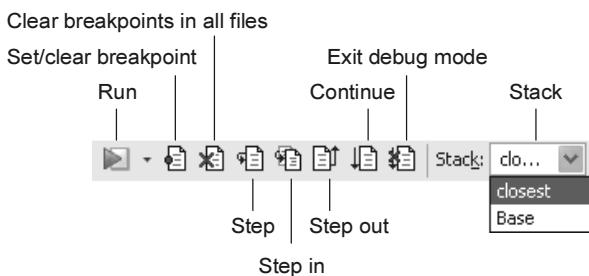


Рис. П1.5. Отладочные средства панели инструментов окна редактора/отладчика

Кнопки панели инструментов имеют следующее назначение:

- **Run** — запуск текущего файла на выполнение;
- **Set/clear breakpoint** — установка или сброс точки прерывания в текущей строке;
- **Clear breakpoints in all files** — сброс всех точек прерывания во всех открытых в редакторе/отладчике файлах;
- **Step** — выполнение одной строки программы, при этом вызов функции считается одним оператором и заход внутрь вызываемой функции не производится;
- **Step in** — выполнение одной строки программы с заходом внутрь вызываемых функций;
- **Step out** — выполнение оставшейся части текущей функции, остановка программы производится после возврата из функции;
- **Continue** — продолжение выполнения программы;
- **Exit debug mode** — завершение отладки;
- **Stack** — этот раскрывающийся список позволяет переключаться между рабочими пространствами MATLAB и вызванных функций.

При остановке программы в отладочном режиме (в установленной точке прерывания или при пошаговом выполнении) в главном окне появляется отладочная подсказка `K>>`. При этом становятся доступными рабочие области памяти всех вызванных в данный момент функций. По умолчанию текущей является рабочая область

памяти последней вызванной функции. Переключаться между рабочими областями памяти всего стека функций можно с помощью раскрывающегося списка **Stack** панели отладочных инструментов (см. рис. П1.5). Такой же список имеется в панели инструментов окна **Workspace** (см. рис. П1.1). Область памяти среды MATLAB фигурирует в списке под именем **Base**.

Кроме того, узнать значение переменной можно, задержав указатель мыши на ее идентификаторе в окне редактора/отладчика. Появится всплывающая подсказка, отображающая значение переменной. Этот способ имеет смысл применять только для скалярных переменных и массивов небольшого размера, поскольку для больших массивов отображается только часть содержимого либо только размер и тип данных.

## Оптимизация MATLAB-программ

Под оптимизацией в данном случае подразумевается повышение скорости работы функций и программ. В MATLAB применимы все или почти все общие приемы оптимизации программ (типа вынесения из цикла наружу всех вычислений, не зависящих от счетчика цикла). Однако у MATLAB имеется и некоторая специфика, которую следует учитывать при оптимизации кода. Об этой специфике и пойдет речь в данном разделе.

При необходимости повысить быстродействие MATLAB-программы следует учитывать два аспекта:

- MATLAB — *интерпретирующая*, а не компилирующая система;
- функции векторно-матричных операций встроены в ядро системы и выполняются предельно быстро.

Самая общая рекомендация, которая дается практически в любой книге, посвященной MATLAB, — стараться избегать использования циклов, заменяя их векторно-матричными операциями и поэлементными операциями над массивами. Приведем простейший пример, вычислив миллион значений функции  $\sin(x)\exp(-x/10^4)$  двумя способами — с помощью цикла и путем выполнения поэлементных операций над векторами. Хронометраж будем вести с помощью функций `tic` и `toc`. Первая из них фиксирует начало отсчета времени, а вторая выводит на экран время (в секундах), прошедшее после `tic`. Вот соответствующий программный код:

```
>> % вычисления в цикле
>> % для чистоты эксперимента уничтожаем все переменные
>> clear all
>> N = 1e6;           % число вычисляемых значений
>> x = zeros(1, N); % выделяем память для результата
>> tic; for k = 1:N, x(k) = sin(k)*exp(-k/1e4); end; toc
Elapsed time is 0.155766 seconds.
>> % векторная операция
>> % для чистоты эксперимента уничтожаем все переменные
>> clear all
```

```
>> N = 1e6; % число вычисляемых значений
>> x = zeros(1, N); % выделяем память для результата
>> tic; k=1:N; x = sin(k).*exp(-k/1e4); toc
Elapsed time is 0.045408 seconds.
```

Как видите, отказ от использования цикла даже в этом простейшем примере ускорил вычисления почти в 3,5 раза. В более сложных ситуациях выигрыш может быть существенно значительнее.

Однако не следует воспринимать данный совет как абсолютную догму. Не все алгоритмы хорошо поддаются векторизации, и иногда попытка искусственно свести алгоритм к последовательности матричных операций может потребовать создания промежуточных матриц большого размера. Необходимость выделения памяти для хранения этих матриц может свести на нет преимущества отказа от циклов.

Итак, без циклов все-таки обойтись не удается, но при необходимости их использования следует обратить внимание на еще один аспект — выделение памяти для хранения переменных. Язык программирования MATLAB, являясь интерпретируемым языком сверхвысокого уровня, скрывает от пользователя операции, связанные с выделением и освобождением памяти. В результате мы чаще всего даже не задумываемся над тем, что в действительности происходит при выполнении простейшего (с точки зрения пользователя) оператора присваивания. Рассмотрим небольшой пример:

```
for k = 1:N
    % какие-нибудь вычисления
    ...
    x(k) = ...
end
```

Если переменная  $x$  не существовала до начала цикла, то при каждом проходе цикла происходит увеличение числа элементов в ней. В результате при каждом выполнении оператора присваивания система должна проделать следующее:

- выделить память под переменную  $x$  нового размера;
- скопировать туда старое содержимое переменной;
- освободить память, отведенную ранее под хранение старого значения  $x$ ;
- наконец, выполнить собственно оператор присваивания для  $x(k)$ .

Операции выделения и освобождения памяти выполняются довольно медленно, поэтому необходимость проделывать их при каждом проходе цикла может сильно замедлить работу программы, особенно если наращиваемые массивы имеют большой размер и если в цикле их используется несколько.

В качестве примера будем в цикле заполнять два вектора случайными гауссовыми числами, выводя на экран время, потребовавшееся для добавления каждого 2000 элементов:

```
>> clear all % для чистоты эксперимента
>> tic
```

```
>> for k = 1:2e4
>>     x(k) = randn;
>>     y(k) = randn;
>>     if rem(k, 2e3)==0
>>         t=toc;
>>         fprintf(1, '%d: %.3f seconds\n', k, t)
>>         tic
>>     end
>> end
2000: 0.035 seconds
4000: 0.080 seconds
6000: 0.118 seconds
8000: 0.152 seconds
10000: 0.189 seconds
12000: 0.221 seconds
14000: 0.255 seconds
16000: 0.291 seconds
18000: 0.325 seconds
20000: 0.359 seconds
```

Как видите, по мере роста размера массивов скорость работы сильно падает — для добавления последней пары тысяч элементов понадобилось примерно в 10 раз больше времени, чем для первой!

Решением этой проблемы является предварительное выделение памяти под заполняемые в цикле массивы. Для этого нужно создать переменную нужного размера, скажем, с помощью функции zeros, формирующую массив, заполненный нулями. Модифицируем предыдущий пример:

```
>> clear all          % для чистоты эксперимента
>> x = zeros(1, 2e4); % создаем массивы заранее
>> y = zeros(1, 2e4);
>> tic
>> for k = 1:2e4
>>     x(k) = randn;
>>     y(k) = randn;
>>     if rem(k, 2e3)==0
>>         t=toc;
>>         fprintf(1, '%d: %.3f seconds\n', k, t)
>>         tic
>>     end
>> end
2000: 0.010 seconds
4000: 0.011 seconds
6000: 0.010 seconds
8000: 0.010 seconds
10000: 0.010 seconds
12000: 0.010 seconds
14000: 0.010 seconds
```

```
16000: 0.010 seconds
18000: 0.010 seconds
20000: 0.010 seconds
```

Результат налицо: работа цикла проходит равномерно, не замедляясь.

При использовании оператора `while` число проходов цикла заранее не известно. Поэтому не известен и итоговый размер наращиваемых в таком цикле массивов. В этой ситуации можно применить компромиссный подход, увеличивая массивы внутри цикла, но не при каждом проходе, а изредка, зато большими кусками. Получается код наподобие следующего:

```
size_incr = 10000;           % дискретность приращения
x = zeros(1, size_incr);    % создание массивов
y = zeros(1, size_incr);
k = 1;                      % счетчик элементов
while cond
    % проверяем, исчерпана ли текущая длина массивов
    if k>length(x)
        % выделяем дополнительную память
        x = [x zeros(1, size_incr)];
        y = [y zeros(1, size_incr)];
    end
    % какие-нибудь вычисления
    ...
    x(k) = ...
    y(k) = ...
    k = k+1;      % увеличиваем значение счетчика
end
% после завершения цикла удаляем нулевые "хвосты"
x(k:end) = [];
y(k:end) = [];
```

Данный подход позволяет существенно ускорить вычисления, если число проходов цикла заранее не известно.

До сих пор мы измеряли время выполнения фрагментов программы с помощью функций `tic` и `toc`. Они весьма полезны, но в MATLAB имеется и более серьезный хронометрирующий инструмент — утилита профилирования, вызываемая командой

```
profile on
```

В качестве примера создадим несколько вариантов функции `ind_diff`, вычисляющей квадратную матрицу, значения элементов которой равны разности их индексов:  $A_{ij} = i - j$ . Единственным входным параметром является размер `N` создаваемой матрицы.

Первая версия кода будет наиболее прямолинейной — с использованием двух вложенных циклов:

```

function y = ind_diff(N)
for k = 1:N
    for l = 1:N
        y(k,l) = k - l;
    end
end

```

Сохраняем текст функции в виде файла `ind_diff.m`, включаем профилирование и вызываем функцию:

```

>> profile on
>> y = ind_diff(500);

```

Для вывода результатов профилирования используется следующая команда:

```
>> profile report
```

MATLAB сгенерирует отчет в HTML-формате и покажет его в собственном браузере (рис. П1.6).

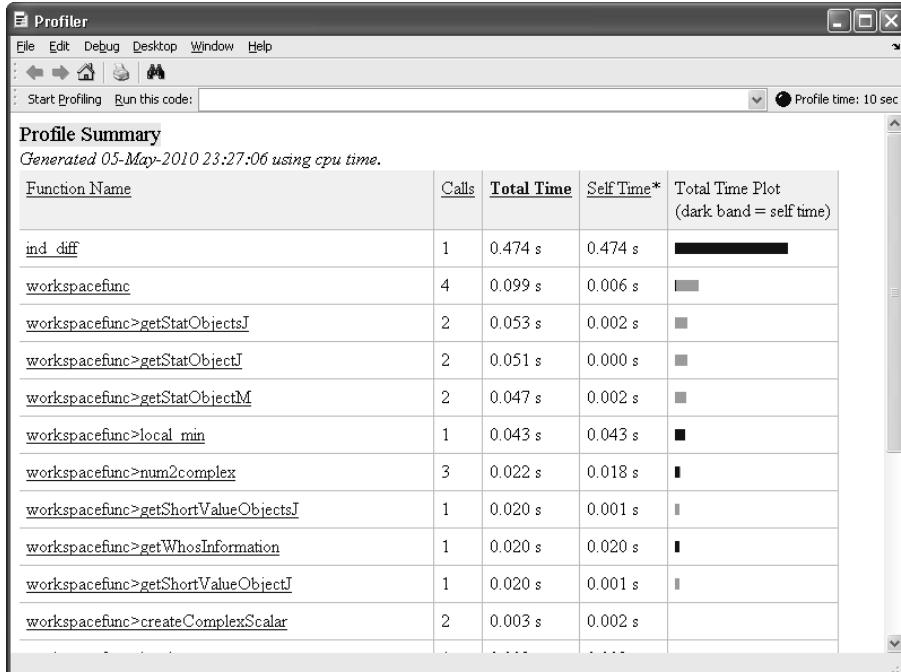
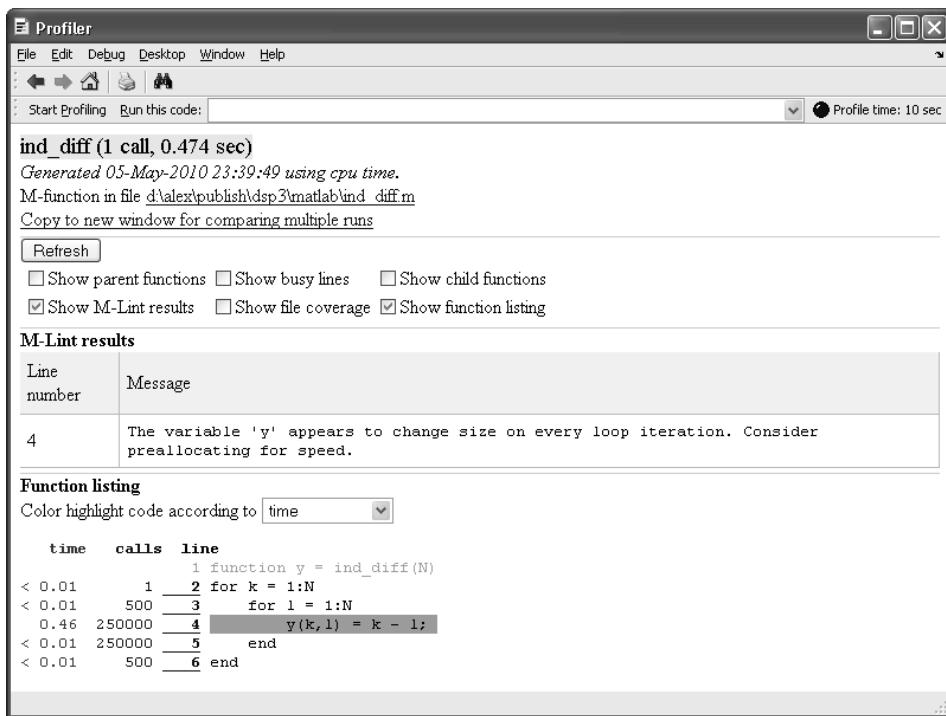


Рис. П1.6. Итоговый отчет о профилировании

В итоговом отчете показан список вызванных функций, для каждой из них в таблице приведено число вызовов (**Calls**), общее время, затраченное на ее выполнение (**Total Time**), и собственное время выполнения (**Self Time**), в котором не учитывается время работы других функций, вызываемых из данной. В данном случае, кроме служебных функций среды MATLAB, вызывалась только наша функция `ind_diff`. Чтобы получить подробную информацию о результатах профилирования

этой функции, необходимо щелкнуть на ссылке с ее именем в столбце **Function Name**. Откроется страница детализированной информации (рис. П1.7).



**Рис. П1.7.** Отчет о результатах профилирования — детальная информация о вызываемых функциях

В верхней части окна находится группа флажков, которые позволяют выбирать состав отображаемой информации. По умолчанию они все установлены, но в данном примере выбраны только два раздела, которые представляют наибольший интерес.

Установка флажка **Show M-Lint results** позволяет получить рекомендации анализатора кода M-Lint о возможных модификациях программы. В таблице выводятся номера строк кода и соответствующие советы. В данном случае в сообщении для строки 4 говорится, что размер массива у увеличивается при каждом проходе цикла и предлагается выделить под него память заранее.

Установка флажка **Show function listing** позволяет получить наиболее интересную часть отчета — листинг функции, рядом со строками которого показано число их выполнений и затраченное на это время (в секундах). Строки, вызвавшие наибольшие временные затраты, выделены цветом.

Проанализируем полученные результаты. Некоторое время потрачено на обслуживание циклов (строки 2, 3, 5 и 6), но основные затраты пришлись на оператор присваивания (строка 4).

Воспользуемся данным ранее советом (тем более что его повторяет и автоматический анализатор) и заранее выделим память под создаваемый массив:

```

function y = ind_diff(N)
y = zeros(N, N);
for k = 1:N
    for l = 1:N
        y(k,l) = k - l;
    end
end

```

Сохраняем новую версию и снова включаем профилирование (команда `profile report` выключает режим профилирования):

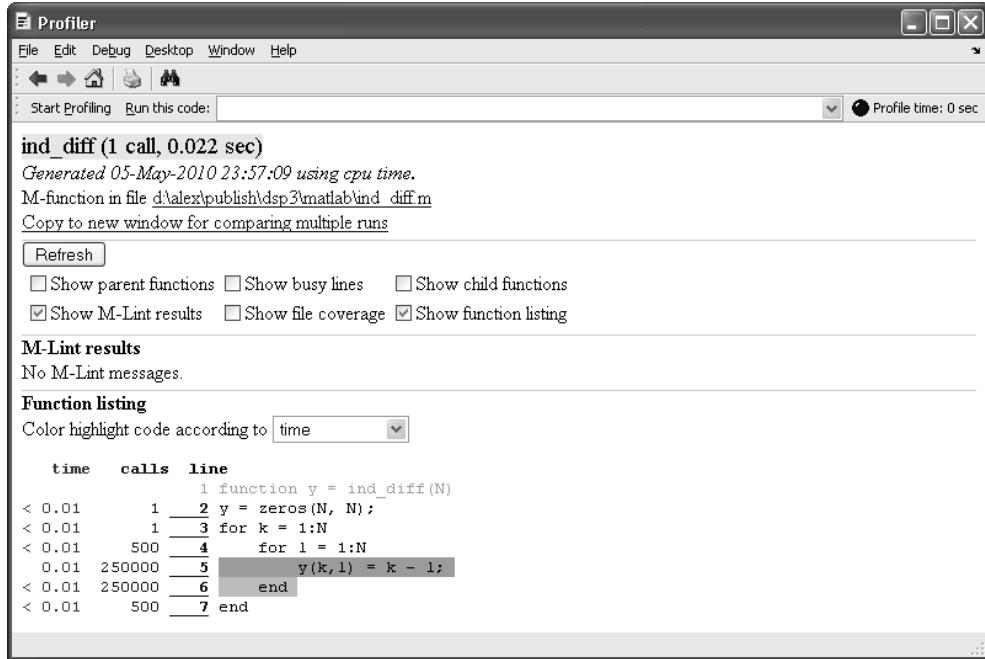
```

>> profile on
>> y = ind_diff(500);
>> profile report

```

Сгенерированный детальный отчет показан на рис. П1.8. Как видите, время, затрачиваемое на выполнение присваивания внутри цикла, уменьшилось во много раз. Завершаем работу с утилитой профилирования:

```
>> profile off
```



**Рис. П1.8.** Результаты профилирования оптимизированного кода

### ЗАМЕЧАНИЕ

Приведенные варианты кода были рассмотрены лишь с целью продемонстрировать профилирование. Время выполнения данной функции можно еще сильнее уменьшить, отказавшись от циклов вовсе. Если вам действительно нужно создать такую

матрицу, следует заметить, что вдоль ее диагоналей, параллельных главной, стоят одинаковые числа, и воспользоваться функцией `toeplitz`:

```
y = toeplitz(0:N-1, -(0:N-1));
```

Это, пожалуй, наиболее быстрый способ.

## Графика

Говоря о графических средствах MATLAB, прежде всего нужно отметить следующее. MATLAB — матричная программа, и ее графические команды могут лишь разнообразными способами визуализировать векторы и матрицы. На практике это означает, что точки, соответствующие элементам векторов и матриц, могут соединяться лишь прямыми линиями — никакого сглаживания или интерполяции производиться не будет. Если вы хотите, чтобы график выглядел более плавным, позаботьтесь о том, чтобы в визуализируемом массиве было больше точек. Если строится график функции, уменьшите шаг между соседними значениями ее аргумента. Если необходимо сгладить экспериментально полученные данные, для которых получить дополнительные точки затруднительно, следует воспользоваться функциями интерполяции (`interp*`, `spline`, `pchip`, `griddata*`) или аппроксимации (`polyfit`; кроме того, имеется графический интерфейс аппроксимации, вызываемый командой **Basic Fitting** из меню **Tools** графических окон). За более подробными сведениями обратитесь к документации MATLAB.

## Двумерная графика

Основным средством двумерной графики является функция `plot`. Наиболее типичный вариант ее использования выглядит следующим образом:

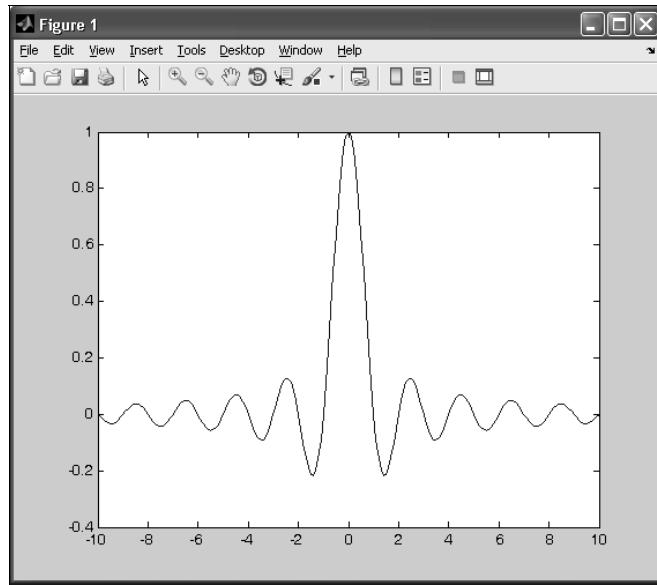
```
plot(x, y)
```

Здесь `x` и `y` — векторы одинаковой длины, задающие соответствующие координаты точек, выводимых на график. По умолчанию точки соединяются сплошными линиями синего цвета. Тип и цвет линий и символов точек можно изменить, об этом будет рассказано далее в разд. *"Настройка внешнего вида графиков"* этого приложения.

В качестве примера построим график функции  $y = \sin(\pi x)/(\pi x)$ , которая в MATLAB имеет имя `sinc` (рис. П1.9):

```
>> x = -10:0.1:10; % значения координаты x
>> y = sinc(x);    % значения координаты y
>> plot(x, y)
```

Если на экране еще не было графиков, функция `plot` создаст графическое окно. Если на экране уже имелись графические окна, график будет выведен в *текущее* окно — то, которое последним выводилось на передний план. О том, как можно создать несколько графических окон, рассказано далее в разд. *"Одновременный вывод нескольких графиков"* этого приложения.



**Рис. П1.9.** График, построенный функцией `plot`

### ЗАМЕЧАНИЕ 1

Можно использовать функцию `plot` с одним аргументом — `plot(y)`. В этом случае вектор `y` задает вертикальные координаты точек, а по горизонтали откладываются номера элементов вектора.

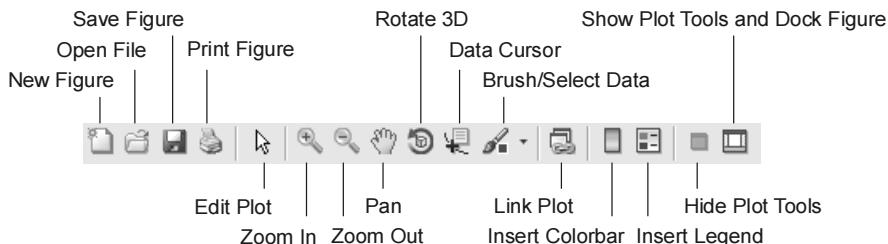
### ЗАМЕЧАНИЕ 2

С помощью кнопок с маленькими стрелками, расположенных в правом верхнем углу под кнопкой закрытия окна, графические окна можно пристыковывать (dock) к командному окну и отделять (undock) от него.

## Панель управления графического окна

В верхней части графического окна расположена панель инструментов (рис. П1.10), кнопки которой выполняют следующие функции:

- **New Figure** — создает новое графическое окно;
- **Open File** — позволяет создать новое графическое окно, загрузив в него ранее сохраненный график;



**Рис. П1.10.** Панель инструментов графического окна

- **Save Figure** — позволяет сохранить графическое окно в виде файла с расширением .fig. Сохраняемый файл представляет собой МАТ-файл (см. ранее разд. "Ввод и вывод данных" этого приложения), в котором записаны массивы данных, представленных на графике, а также все параметры настройки графического окна. Сохраненный FIG-файл впоследствии можно загрузить с помощью кнопки **Open File**, воссоздав записанный график;
- **Print Figure** — позволяет распечатать содержимое графического окна;
- **Edit Plot** — включает режим редактирования графика. В этом режиме можно выделять элементы графика мышью и настраивать их свойства;
- **Zoom In** — включает режим увеличения масштаба. В этом режиме щелчок левой кнопкой мыши в области графика увеличивает масштаб отображения, правой — уменьшает. Можно также, удерживая левую кнопку мыши нажатой, обозначить прямоугольную область, которую необходимо рассмотреть в увеличенном масштабе;
- **Zoom Out** — включает режим уменьшения масштаба. В этом режиме щелчок левой кнопкой мыши в области графика уменьшает масштаб отображения, правой — увеличивает;
- **Pan** — включает режим панорамирования. В этом режиме можно перетаскивать видимую часть графика мышью, не меняя ее масштаба;
- **Rotate 3D** — включает режим настройки точки обзора. В этом режиме перемещение указателя мыши в области графика при нажатой левой кнопке мыши приводит к повороту габаритного параллелепипеда трехмерного графика. Использование данного режима для двумерных графиков не имеет особого смысла;
- **Data Cursor** — включает режим добавления подписей к точкам графика. В подписях отображаются значения координат помеченных точек;
- **Brush>Select Data** — включает режим интерактивного выделения фрагментов графика и редактирования тех данных, по которым этот фрагмент построен;
- **Link Plot** — включает режим синхронизации графика со значениями переменных, задающих векторы координат точек графика. В этом режиме изменения значений переменных вызывают автоматическую перерисовку графика;
- **Insert Colorbar** — добавляет на график цветовую шкалу;
- **Insert Legend** — добавляет на график легенду;
- **Hide Plot Tools** — закрывает окно инспектора свойств графических объектов, если оно открыто;
- **Show Plot Tools and Dock Figure** — открывает окно инспектора свойств графических объектов и пристыковывает к нему окно графика.

Команды меню графического окна в основном совпадают с кнопками панели инструментов. Отметим лишь несколько команд, представляющих особый интерес:

- **File | Save As** — позволяет сохранить график в виде файла одного из поддерживаемых графических форматов;

- Tools | Basic Fitting** — выводит окно настройки аппроксимации, с помощью которого можно подобрать подходящую аппроксимирующую кривую для представленных на графике данных;
- Tools | Data Statistics** — выводит окно просмотра статистической информации о представленных на графике данных.

### **ВНИМАНИЕ!**

График MATLAB является *векторным графическим объектом*, для превращения которого в экранные пиксели может понадобиться проделать значительный объем вычислений. При перетаскивании окна размер графика не меняется, так что нет необходимости пересчитывать его заново — Windows лишь перемещает растровый образ окна. Если же вы меняете *размер* окна, разворачивая его, перетаскивая его границы или используя команду **Размер** системного меню окна, все вычисления, необходимые для построения растрового образа графика, системе придется проделать заново. Поэтому перерисовка графика после изменения размеров его окна потребует столько же времени, сколько и первоначальное построение. Имейте это в виду, если ваш график содержит большое количество элементов.

Ситуация усугубляется, если в Windows установлен режим отображения содержимого окна при перетаскивании. При этом система попытается отследить перетаскивание границы окна, непрерывно перерисовывая график. В случае сложного графика это может привести к весьма существенному торможению работы.

## **Другие разновидности двумерных графиков**

Кроме функции `plot` есть еще целый ряд функций с аналогичным синтаксисом, позволяющих получить другие разновидности двумерных графиков. Ниже перечислены только некоторые из этих функций, наиболее полезные в инженерных и научных приложениях:

- `semilogx(x, y)` — график с логарифмическим масштабом по оси *x*;
- `semilogy(x, y)` — график с логарифмическим масштабом по оси *y*;
- `loglog(x, y)` — график с логарифмическим масштабом по обеим осям;
- `plotyy(x1, y1, x2, y2)` — вывод зависимостей *y1* от *x1* и *y2* от *x2* с раздельной оцифровкой вертикальных осей (для первого графика оцифровка вертикальной оси наносится слева, для второго — справа);
- `stem(x, y)` — график в виде "стебельков";
- `stairs(x, y)` — график в виде ступенчатой линии;
- `polar(phi, r)` — график в полярных координатах (*phi* — угловые координаты точек в радианах, *r* — соответствующие им радиусы).

## **Трехмерная графика**

В MATLAB имеется функция с именем `plot3`, но, вопреки ожиданиям, основным средством трехмерной графики является вовсе не она. Дело в том, что функция `plot3(x, y, z)` является просто трехмерным аналогом функции `plot`, т. е. она по-

зволяет построить *линию* в трехмерном пространстве по координатам точек, задаваемым векторами *x*, *y* и *z*. На практике гораздо чаще бывает необходимо строить графики функций двух переменных в виде поверхностей или линий равного уровня.

Для построения трехмерных поверхностей служит функция *surf*, имеющая следующий синтаксис:

```
surf(X, Y, Z)
```

Здесь *X*, *Y* и *Z* — двумерные массивы одинакового размера. Функция строит поверхность, состоящую из четырехугольных ячеек. Координаты углов каждой ячейки задаются значениями четырех соседних элементов массивов *X*, *Y* и *Z* с индексами  $(i, j)$ ,  $(i, j+1)$ ,  $(i+1, j)$  и  $(i+1, j+1)$ . Цвет ячеек (точнее, индекс цвета в текущей палитре) по умолчанию меняется пропорционально координате *Z*.

Из способа задания параметров для функции *surf* видно, что ее возможности не ограничиваются построением двухкоординатных функциональных зависимостей. Действительно, функция *surf* позволяет строить произвольные трехмерные поверхности. Однако эта гибкость несколько усложняет построение обычных графиков функций двух переменных.

Чтобы построить график функционально заданной поверхности  $z = f(x, y)$ , массивы *X* и *Y*, передаваемые функции *surf*, должны задавать сетку (как правило, равномерную). Значения массива *Z* рассчитываются по формуле функциональной зависимости с использованием поэлементных операций над массивами *X* и *Y*.

Массивы *X* и *Y* в данном случае удобнее всего формировать с помощью специальной функции *meshgrid*, имеющей следующий синтаксис:

```
[X, Y] = meshgrid(x, y)
```

Здесь *x* и *y* — векторы, задающие наборы значений *x*- и *y*-координат. Формируемые массивы *X* и *Y* имеют *length(y)* строк и *length(x)* столбцов. Строки массива *X* являются копиями вектора *x*, а столбцы массива *Y* — копиями вектора *y*. Чтобы пояснить работу функции *meshgrid*, передадим ей два целочисленных вектора:

```
>> [X, Y] = meshgrid(1:5, -3:3)
```

```
X =
```

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

```
Y =
```

-3	-3	-3	-3	-3
-2	-2	-2	-2	-2
-1	-1	-1	-1	-1
0	0	0	0	0

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

В качестве примера построим график функции

$$z(x, y) = \frac{\sin(\pi\sqrt{x^2 + y^2})}{\pi\sqrt{x^2 + y^2}}$$

при изменении  $x$  и  $y$  от  $-5$  до  $5$  с шагом  $0,2$  (рис. П1.11):

```
>> x = -5:0.2:5;
>> [X, Y] = meshgrid(x);
>> Z = sinc(sqrt(X.^2+Y.^2));
>> surf(X, Y, Z)
>> colormap gray
```

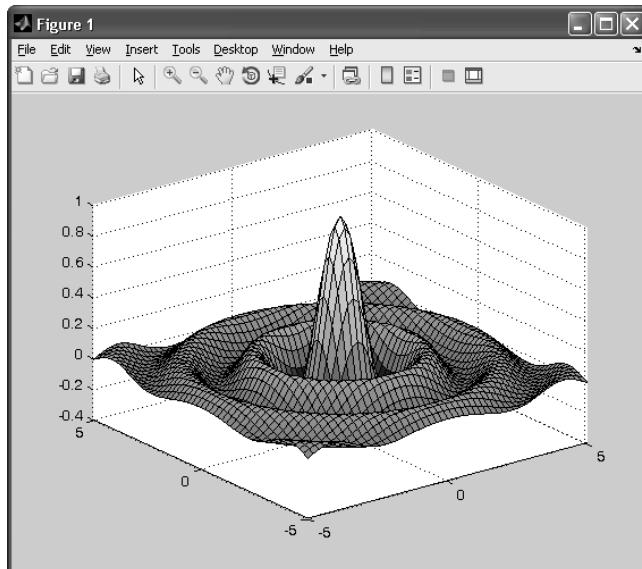


Рис. П1.11. Трехмерная поверхность, построенная с помощью функции `surf`

### ЗАМЕЧАНИЕ

В приведенном фрагменте кода функция `meshgrid` используется с одним входным параметром. Это сокращенный вариант вызова, эквивалентный `meshgrid(x, x)`.

Для построения графика функции двух переменных в виде линий равного уровня служит функция `contour`, имеющая следующий синтаксис:

```
[c, h] = contour(X, Y, Z, V)
```

Здесь  $Z$  — двумерный массив, содержащий значения функции, а параметры  $X$  и  $Y$  могут быть как массивами, полученными от функции `meshgrid` (см. ранее), так и векторами координат, передаваемыми функции `meshgrid`. При отсутствии этих

параметров в качестве координат используются номера строк и столбцов матрицы  $Z$ .

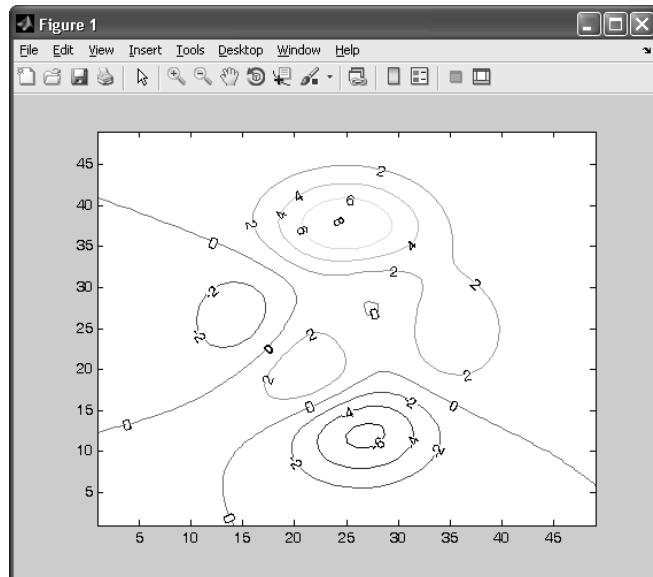
Параметр  $V$  может быть числом (тогда он задает количество выводимых линий уровня) или вектором (тогда этот параметр трактуется как набор значений функции для построения линий равного уровня). При отсутствии данного параметра значения уровня выбираются автоматически.

Возвращаемыми результатами  $c$  и  $h$  являются структуры данных, которые необходимо передать другой функции, `clabel`, для нанесения оцифровки на линии уровня:

```
clabel(c, h)
```

В качестве примера построим линии уровня для функции `peaks`. Это демонстрационная функция трехмерной графики MATLAB, генерирующая поверхность с несколькими пиками и впадинами. Нанесем также на линии уровня оцифровку с помощью функции `clabel` (рис. П1.12):

```
>> [c, h] = contour(peaks);
>> clabel(c, h)
```



**Рис. П1.12.** Линии равного уровня, построенные с помощью функции `contour`, с оцифровкой, нанесенной функцией `clabel`

## Другие разновидности трехмерных графиков

Кроме функций `surf` и `contour` есть еще целый ряд функций с аналогичным синтаксисом, позволяющих получить другие разновидности трехмерных графиков. Ниже перечислены только некоторые из этих функций, наиболее полезные в инженерных и научных приложениях:

- `surf(x, y, z)` — комбинация функций `surf` и `contour`. Линии равного уровня выводятся на нижней координатной плоскости;
- `mesh(x, y, z)` — построение поверхности в виде сетки с закрашенными ребрами и незакрашенными четырехугольными ячейками;
- `meshc(x, y, z)` — комбинация функций `mesh` и `contour`. Линии равного уровня выводятся на нижней координатной плоскости;
- `meshz(x, y, z)` — то же, что `mesh`, но с краев построенной сетчатой поверхности вниз спадает "занавес";
- `waterfall(x, y, z)` — то же, что `mesh`, но ребра, разделяющие ячейки, проводятся только вдоль оси  $x$ . В результате объемное тело выглядит "нарезанным на ломтики";
- `stem3(x, y, z)` — вывод трехмерного графика в виде "стебельков", начинающихся при  $z = 0$  в точках, задаваемых массивами  $x$  и  $y$ . Высота "стебельков" определяется массивом  $z$ ;
- `contourf(x, y, z)` — то же, что `contour`, но пространство между линиями равного уровня окрашено в различные цвета в зависимости от значений  $z$ ;
- `contour3(x, y, z)` — то же, что `contour`, но линии равного уровня рисуются не в одной плоскости, а при соответствующих значениях  $z$ -координаты;
- `pcolor(x, y, z)` — строит двумерный график, представляющий собой сетку из четырехугольных ячеек, координаты вершин которых задаются массивами  $x$  и  $y$ . Ячейки закрашены цветами, определяемыми значениями элементов массива  $z$ . Тот же результат можно получить, если для поверхности, построенной функцией `surf`, установить точку обзора точно сверху.

## Настройка внешнего вида графиков

В большинстве случаев параметры оформления графиков, выбираемые по умолчанию, оказываются вполне приемлемыми. Однако иногда для повышения выразительности изображения возникает необходимость изменить тип и цвет линий, нанести на график поясняющие надписи и т. п. В данном разделе кратко рассматриваются соответствующие средства MATLAB.

### Настройка линий и точек

Соединение точек графика отрезками синего цвета — не единственный вариант отображения. MATLAB позволяет управлять цветом графика, типом линии и способом отображения точек данных. Для этого используется дополнительный параметр команд `plot`, `stem` и т. п. Этот параметр представляет собой текстовую строку `'LineSpec'`, символы которой и указывают нужные режимы:

```
plot(x, y, 'LineSpec')
```

Список возможных символов и их значения приведены в табл. П1.1.

**Таблица П1.1.** Символы управления линиями и точками графиков

Символ	Назначение
<b>Управление цветом</b>	
b	Синий (по умолчанию) — Blue
c	Голубой — Cyan
g	Зеленый — Green
k	Черный — black
m	Фиолетовый — Magenta
r	Красный — Red
w	Белый — White
y	Желтый — Yellow
<b>Управление типом линии</b>	
-	Непрерывная (по умолчанию)
--	Пунктирная, длинный штрих
:	Пунктирная, короткий штрих
-. .	Штрихпунктирная
<b>Управление маркерами точек данных</b>	
.	Точка (по умолчанию)
+	Знак "плюс"
*	Звездочка
o	Кружок
x	Крестик
s	Квадрат — Square
d	Ромб — Diamond
p	Пятиугольник — Pentagon
h	Шестиугольник — Hexagon
v	Треугольник вершиной вниз
^	Треугольник вершиной вверх
<	Треугольник вершиной влево
>	Треугольник вершиной вправо

Перечисленные в таблице символы можно комбинировать. К примеру, строка ' -.dr' задаст вывод точек графика ромбиками (d), а соединяющих точки линий — штрихпунктиром (-.). И точки, и линии при этом будут красного цвета (r).

## Функции оформления графиков

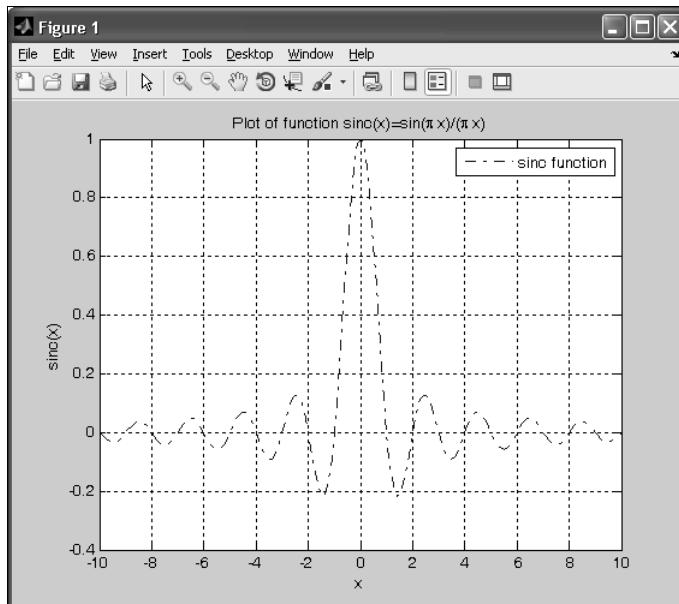
Только что рассмотренные параметры управления точками и линиями задаются непосредственно при выводе графика функцией `plot` или какой-либо подобной ей. Кроме того, существует еще целый ряд функций, которые вызываются уже после построения графика и меняют его внешний вид. Эти функции воздействуют только на текущий график (т. е. на график в текущем графическим окне и, если использовалась команда `subplot`, в текущей клетке этого окна). Ниже приведен список наиболее важных из этих функций:

- `axis` — настройка координатных осей. Вот некоторые способы использования этой функции ( обратите внимание на то, что не все перечисленные режимы являются взаимоисключающими):
  - `axis([xmin xmax ymin ymax])` — настройка предельных значений осей для двумерного графика;
  - `axis([xmin xmax ymin ymax zmin zmax])` — настройка предельных значений осей для трехмерного графика;
  - `axis auto` — автоматический выбор пределов (этот режим установлен по умолчанию);
  - `axis tight` — предельные значения осей в точности соответствуют диапазонам изменения данных;
  - `axis equal` — для всех осей графика устанавливается одинаковый масштаб;
  - `axis image` — то же, что комбинация `axis equal` и `axis tight`;
  - `axis square` — делает длину осей одинаковой (а область графика — квадратной);
  - `axis normal` — отменяет действие режимов `axis square` и `axis equal`;
  - `axis off` — отключает отображение осей вместе с оцифровкой, подписями и фоном графика;
  - `axis on` — включает отображение осей, их оцифровки, подписей, а также фона графика;
- `grid` — включение (`grid on`) и выключение (`grid off`) отображения линий сетки;
- `box` — включение (`box on`) и выключение (`box off`) отображения обрамляющего график прямоугольника или параллелепипеда;
- `legend('string1', 'string2', 'string3', ...)` — добавление аннотации ("легенды") к графику. Каждая из строк-параметров задает аннотацию для одной из выведенных на график зависимостей;
- `title('text')` — добавление заголовка к графику;
- `xlabel('text')` — добавление подписи к оси  $x$ ;
- `ylabel('text')` — добавление подписи к оси  $y$ ;
- `zlabel('text')` — добавление подписи к оси  $z$ ;

- `text(x, y, 'string')` или `text(x, y, z, 'string')` — размещение строки `'string'` в точке двумерного или трехмерного графика с указанными координатами;
- `xlim([xmin xmax])` — задание пределов по оси *x*;
- `ylim([ymin ymax])` — задание пределов по оси *y*;
- `zlim([zmin zmax])` — задание пределов по оси *z*.

В качестве примера повторим вывод графика, показанного ранее на рис. П1.9, штрихпунктирной линией. Кроме того, добавим сетку, подписи к осям, "легенду" и заголовок. Результат показан на рис. П1.13:

```
>> x = -10:0.1:10; % значения координаты x
>> y = sinc(x);    % значения координаты y
>> plot(x, y, '-.') % добавляем сетку
>> xlabel('x')      % подпись оси x
>> ylabel('sinc(x)') % подпись оси y
>> legend('sinc function') % аннотация
>> % заголовок графика
>> title('Plot of function sinc(x)=sin(\pi x)/(\pi x)')
```



**Рис. П1.13.** График с настроенными параметрами отображения линий, подписями осей, "легендой" и заголовком

Обратите внимание на то, что последовательность символов `\pi`, использованная в аргументе функции `title`, на рис. П1.13 превратилась в греческую букву  $\pi$ . Аналогичным образом при использовании функций `title`, `xlabel`, `ylabel`, `legend` и т. п.

можно получить и другие специальные символы. Дело в том, что MATLAB распознает в выводимых на график текстовых строках некоторые команды языка TeX. Это команды вывода греческих букв (`\alpha`, `\beta` и т. д. для строчных букв и `\Gamma`, `\Delta` и т. д. для прописных, не совпадающих по начертанию с английскими), вывода математических символов (`\infty` для  $\infty$ , `\leq` для  $\leq$ , `\geq` для  $\geq$ , `\pm` для  $\pm$  и т. п.), форматирования нижних и верхних индексов (соответственно `_...` и `^...`), а также управления начертанием шрифта (`\bf` — полужирный, `\it` — курсив, `\sl` — наклонный, `\rm` — обычный). За более подробной информацией о форматировании выводимого на график текста обратитесь к документации MATLAB.

### **ЗАМЕЧАНИЕ**

*TeX* — это непревзойденная по своим возможностям система подготовки текстов, содержащих математические формулы. *TeX* не является визуальной системой — это скорее язык программирования. Чтобы продемонстрировать идею, приведем маленький пример — вы набираете во входном файле, скажем, `$\gamma=\sqrt{\alpha^2+\beta^2}` и получаете на печати  $\gamma = \sqrt{\alpha^2 + \beta^2}$ . Подумайте — может быть, это лучше, чем щелкать мышью в Equation Editor? Подробнее узнать о *TeX* можно, например, из книг [37, 38].

## **Одновременный вывод нескольких графиков**

Вывести на экран несколько графиков одновременно можно по-разному — в различных графических окнах, в разных областях одного окна или же в общих осьх координат. В зависимости от этого используются различные средства MATLAB.

Чтобы вывести несколько графиков в разных графических окнах, нужно создать эти графические окна с помощью функции `figure` или команды меню **File | New | Figure** любого окна MATLAB. После создания нового окна оно становится текущим, и последующий графический вывод направляется именно в него.

### **ЗАМЕЧАНИЕ**

Сделать конкретное графическое окно текущим можно, выведя его на передний план и затем переключившись в командное окно MATLAB.

Приведенные далее команды создадут два графика в разных окнах:

```
>> plot(x1, y1)
>> figure
>> plot(x2, y2)
```

Для вывода нескольких графиков в различных областях одного окна используется функция `subplot`, имеющая следующий синтаксис:

```
subplot(Rows, Cols, N)
```

В результате графическое окно будет разбито на клетки в виде матрицы, имеющей `Rows` строк и `Cols` столбцов, и `N`-я клетка (в отличие от нумерации элементов в матрицах, нумерация клеток ведется *по строкам*) становится текущей.

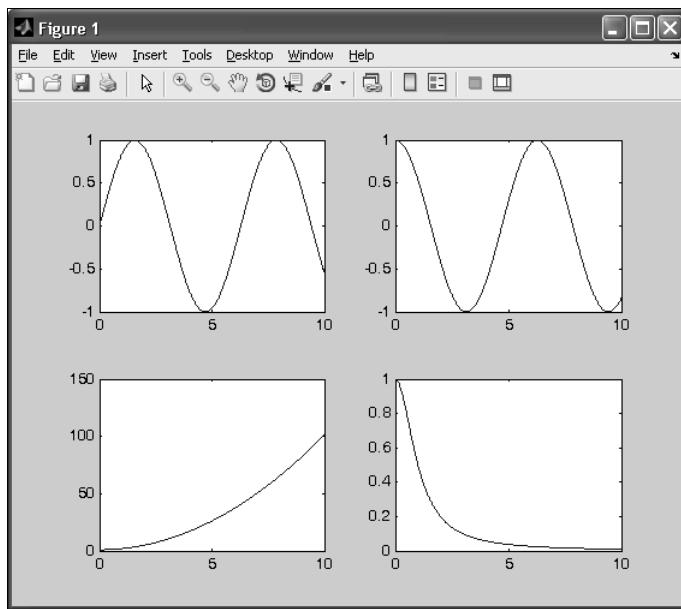


Рис. П1.14. Результат использования команды `subplot`

Приведенные далее команды создают четыре графика, располагая их в одном графическом окне в две строки и два столбца (рис. П1.14):

```
>> t = 0:0.01:10;
>> subplot(2, 2, 1)
>> plot(t, sin(t))
>> subplot(2, 2, 2)
>> plot(t, cos(t))
>> subplot(2, 2, 3)
>> plot(t, t.^2+1)
>> subplot(2, 2, 4)
>> plot(t, 1./(t.^2+1))
```

Наконец, вывести несколько графиков в общих осях тоже можно разными способами. Во-первых, если параметр функции `plot`, задающий значения координат  $y$ , является матрицей, то строятся отдельные графики для каждого столбца матрицы, например:

```
>> x = (0:0.1:10)';
>> plot(x, [sin(x) cos(x)])
```

Во-вторых, при вызове функции `plot` можно перечислить несколько пар значений  $x$ - и  $y$ -координат:

```
plot(x1, y1, x2, y2, ...)
```

При этом для каждой пары  $(x_i, y_i)$  будет построен свой график. Поэтому тот же результат, что и в предыдущем примере, можно получить и таким образом:

```
>> plot(x, sin(x), x, cos(x))
```

В-третьих, можно использовать команду `hold on`, включающую режим сохранения содержимого текущих координатных осей при выводе в них новых графиков. Тогда новые команды `plot` будут строить графики поверх имеющихся, не затирая их. Для выключения этого режима используется команда `hold off`.

Перепишем наш пример еще раз, используя команды `hold`:

```
>> plot(x, sin(x))
>> hold on
>> plot(x, cos(x))
>> hold off
```

## Дальнейшее использование графиков

Содержимое графического окна можно распечатать, используя следующие две команды меню **File: Print Preview** и **Print**. Печать в MATLAB выполняется так же, как в других Windows-приложениях, поэтому подробно рассматривать этот процесс мы не будем.

Печатать графики непосредственно из MATLAB приходится редко, гораздо чаще необходимо использовать их в документах, создаваемых, например, в Microsoft Word. Для этого необходимо скопировать изображение в буфер обмена командой **Edit | Copy Figure** меню графического окна.

Однако для получения желаемого результата необходимо настроить параметры копирования с помощью команды **Edit | Copy Options** меню графического окна. После ее вызова открывается соответствующий раздел окна **Preferences**. В нем имеется два трехпозиционных переключателя и один флажок. Переключатель **Clipboard format** позволяет выбирать следующие варианты:

- Metafile (may lose information)** — в буфер помещается векторный образ окна;
- Preserve information (metafile if possible)** — MATLAB выбирает растровый или векторный вариант по своему усмотрению, в зависимости от сложности изображения;
- Bitmap** — в буфер помещается растровый образ окна.

Векторное изображение можно масштабировать без потери качества и редактировать в соответствующих программах (CorelDRAW, Adobe Illustrator и т. п.). При копировании растрового образа неизбежно происходит потеря информации, однако прорисовка и печать растрового рисунка может оказаться быстрее, чем в случае сложного векторного изображения. В большинстве случаев целесообразно выбирать вариант **Preserve information (metafile if possible)**.

Переключатель **Figure background color** управляет копированием фона графического окна:

- Use figure color** — копируется фоновый прямоугольник того же цвета, что на экране (по умолчанию — серый);
- Force white background** — копируется фоновый прямоугольник белого цвета;

- Transparent background** — изображение копируется без фонового прямоугольника.

Наиболее целесообразный вариант — **Transparent background**.

Наконец, флажок **Match figure screen size** управляет размером копируемого изображения. Если флажок установлен, копируемое изображение будет иметь такие же размеры, как на экране. Если флажок снят, размер копируемого изображения определяется настройками, сделанными в окне, вызываемом командой **File | Export Setup**.

#### **ЗАМЕЧАНИЕ**

Помимо копирования в буфер обмена изображения могут выводиться на печать и записываться в файлы различных графических форматов с помощью функций `print` и `saveas`. Они имеют большое количество дополнительных ключей, ознакомиться с назначением которых можно по документации MATLAB.

## **Дополнительные источники информации**

В этом приложении были представлены лишь основные приемы использования MATLAB. Далее, в *приложении 2* будет приведен краткий обзор функций базовой библиотеки MATLAB, а в *приложении 3* — список компонентов системы. За дополнительной информацией следует обращаться к справочной системе, документации и книгам о MATLAB, которых издано уже немало. Кроме того, на сайте фирмы-производителя [www.mathworks.com](http://www.mathworks.com) можно найти информацию о последних версиях программных продуктов, документацию, а также коллекцию M-файлов, разработанных пользователями MATLAB (ее адрес — <http://www.mathworks.com/matlabcentral/fileexchange/>). Возможно, вам удастся найти в этой коллекции что-нибудь полезное для решения стоящих перед вами задач.

Англоязычный форум по MATLAB находится по адресу <http://www.mathworks.com/matlabcentral/newsreader/>.

Компания Softline ([www.softline.ru](http://www.softline.ru)) — официальный дилер фирмы The MathWorks в России — поддерживает сайт [matlab.exponenta.ru](http://matlab.exponenta.ru), на котором имеются информационные материалы (на русском языке) по MATLAB и его пакетам расширения. Там же организован и форум отечественных пользователей MATLAB.

## ПРИЛОЖЕНИЕ 2

# Обзор функций MATLAB

В данном приложении будет дан краткий обзор функций, входящих в базовую библиотеку MATLAB. Этих функций слишком много, чтобы можно было привести хотя бы краткое описание для каждой из них. Поэтому далее приводится список наиболее важных тематических разделов с краткой их характеристикой. Для некоторых разделов перечислены также важнейшие функции. За более подробным описанием следует обращаться к справочной системе MATLAB.

Разделы перечислены в алфавитном порядке имен их папок, а не в порядке значимости. Получить справку по конкретному разделу можно, набрав в командной строке следующий текст:

```
>> help имя_раздела
```

В результате будет выведен список функций данного раздела с краткими комментариями по их назначению.

## Audiovideo

В данном разделе содержатся функции работы с мультимедиафайлами, а также функции взаимодействия со звуковой картой компьютера. Важнейшие из последних были рассмотрены в разд. "Получение данных из внешних источников" главы 3.

## Datafun

В этом разделе сосредоточены разнообразные функции анализа данных. Большинство из них ориентировано на работу с векторным аргументом; матрицы обрабатываются по столбцам. Вот список важнейших функций:

### □ Базовые операции:

- `max, min` — вычисление наибольшего и наименьшего элемента, а также индекса этого элемента;
- `mean, median, var, std` — оценка среднего значения, медианы, дисперсии и среднеквадратического отклонения;
- `sort, sortrows` — сортировка (по возрастанию) элементов и строк матрицы;

- `sum`, `prod` — сумма и произведение элементов;
- `hist`, `histc` — построение гистограммы и расчет данных для нее;
- `trapz` — численное интегрирование методом трапеций;
- `cumsum` — расчет частичных сумм элементов (численное интегрирование методом прямоугольников);
- `cumprod` — расчет частичных произведений элементов.

□ *Конечные разности:*

- `diff` — вычисление разностей между соседними элементами;
- `gradient` — аппроксимация градиента;
- `del2` — дискретный лапласиан.

□ *Корреляция:*

- `corrcoef` — коэффициент взаимной корреляции;
- `cov` — ковариационная матрица;
- `subspace` — расчет угла между векторами.

□ *Фильтрация и свертка:*

- `filter`, `filter2` — одномерная и двумерная дискретная фильтрация;
- `conv`, `deconv` — свертка (умножение полиномов) и обращение свертки (деление полиномов);
- `conv2`, `convn` — двумерная и многомерная свертка;
- `detrend` — удаление линейного тренда.

□ *Преобразование Фурье:*

- `fft`, `fft2`, `fftn` — одномерное, двумерное и многомерное прямое дискретное преобразование Фурье;
- `ifft`, `ifft2`, `ifftn` — одномерное, двумерное и многомерное обратное дискретное преобразование Фурье;
- `fftshift`, `ifftshift` — два зеркальных варианта перестановки половин вектора или квадрантов матрицы.

## Datatypes

В этом разделе сосредоточены средства поддержки типов данных MATLAB.

## Demos

В данном разделе собраны демонстрационные примеры.

## Elfun

Как явствует из названия раздела, в нем содержатся элементарные математические функции. В большинстве случаев эти функции встроены в ядро системы, так что в М-файлах хранится только текст справки.

### □ Тригонометрические функции:

- `sin, sinh` — синус и гиперболический синус;
- `asin, asinh` — арксинус и гиперболический арксинус;
- `cos, cosh` — косинус и гиперболический косинус;
- `acos, acosh` — арккосинус и гиперболический арккосинус;
- `tan, tanh` — тангенс и гиперболический тангенс;
- `atan, atanh` — арктангенс и гиперболический арктангенс;
- `atan2` — вычисление угловой координаты точки по паре ее декартовых координат;
- `sec, sech` — секанс и гиперболический секанс;
- `asec, asech` — аркsecанс и гиперболический аркsecанс;
- `csc, csch` — косеканс и гиперболический косеканс;
- `acsc, acsch` — арккосеканс и гиперболический арккосеканс;
- `cot, coth` — котангенс и гиперболический котангенс;
- `acot, acoth` — арkkотангенс и гиперболический арkkотангенс.

### □ Экспоненциальные функции:

- `exp, log` — экспонента и натуральный логарифм;
- `log10` — десятичный логарифм;
- `log2` — двоичный логарифм, а также выделение порядка и мантиссы числа;
- `pow2` — функция  $2^x$ , а также "сборка" числа из мантиссы и порядка;
- `realpow, reallog, realsqrt` — версии оператора возведения в степень, а также функций `log` и `sqrt`, возвращающие только вещественные результаты (если аргументы функции таковы, что результат должен оказаться комплексным, выдается сообщение об ошибке);
- `sqrt` — квадратный корень;
- `nextpow2` — ближайшая "сверху" степень двойки.

### □ Функции для работы с комплексными числами:

- `abs, angle` — модуль и аргумент (фаза) комплексного числа;
- `complex` — "сборка" комплексного числа из вещественной и мнимой частей;

- `conj` — комплексное сопряжение;
- `real, imag` — вещественная и мнимая части комплексного числа;
- `unwrap` — устранение скачков фазы (подробнее см. разд. "Построение графиков фазочастотных характеристик" главы 2);
- `isreal` — проверка вещественности массива;
- `cplxpair` — сортировка чисел по комплексно-сопряженным парам.

#### □ Округление и остаток:

- `fix, floor, ceil, round` — округление в сторону нуля, вниз, вверх, к ближайшему целому числу;
- `rem, mod` — остаток от деления и взятие числа по заданному модулю (остаток со знаком);
- `sign` — знаковая функция.

## Elmat

В данном разделе собраны функции элементарных матричных операций.

#### □ Создание элементарных матриц:

- `zeros, ones` — матрицы, заполненные нулями и единицами;
- `eye` — единичная матрица;
- `repmat` — дублирование матрицы "мозаикой";
- `rand, randn` — генерация случайных чисел с равномерным и нормальным распределением;
- `linspace, logspace` — генерация арифметической и геометрической прогрессии;
- `freqspace` — генерация вектора частот для анализа частотных характеристик;
- `meshgrid` — генерация массивов координат для построения трехмерных графиков.

#### □ Информационные функции:

- `size, length, numel, ndims` — размер массива, длина вектора, число элементов массива, число размерностей массива;
- `isempty, isnumeric, islogical` — проверка, является ли массив пустым, числовым, логическим;
- `isequal, isequalwithequalnans` — проверка пары массивов на равенство (первая функция считает элементы, равные NaN, несовпадающими, вторая — сопадающими);
- `logical` — преобразование числовых величин в логические.

□ *Преобразования матриц:*

- `cat` — соединение массивов;
- `reshape` — изменение размеров матрицы при сохранении числа элементов;
- `diag` — создание диагональных матриц и выделение вектора диагональных элементов матрицы;
- `blkdiag` — создание блочно-диагональной матрицы;
- `tril, triu` — выделение нижнего и верхнего треугольных блоков матрицы;
- `fliplr, flipud, flipdim` — зеркальное отражение матрицы относительно вертикальной и горизонтальной осей, отражение многомерного массива относительно указанной размерности;
- `rot90` — поворот матрицы на 90°;
- `find` — индексы ненулевых элементов;
- `end` — последний индекс вдоль данной размерности;
- `sub2ind, ind2sub` — преобразование набора индексов в линейный индекс и обратно.

□ *Специальные переменные и константы:*

- `ans` — результат последней операции;
- `eps` — относительная точность вычислений с плавающей запятой;
- `realmin, realmax` — минимально и максимально возможные числа с плавающей запятой;
- `pi` — число  $\pi$ ;
- `i, j` — мнимая единица;
- `inf` — бесконечность;
- `NaN` — нечисловое значение (Not-a-Number);
- `isnan, isinf, isfinite` — проверка на `NaN`, бесконечность и конечность значения.

□ *Специальные матрицы.* К этой категории относятся функции, генерирующие матрицы Адамара (`hadamard`), Ганкеля (`hankel`), Гильберта (`hilb`), Паскаля (`pascal`) и др. Для задач, связанных с обработкой сигналов, особый интерес представляет функция `toeplitz`, предназначенная для построения матриц с теплоизцевой структурой (у таких матриц вдоль диагоналей расположены одинаковые значения). Поскольку именно такой вид имеют корреляционные матрицы стационарных случайных сигналов, функцию `toeplitz` можно использовать для построения корреляционной матрицы по отсчетам корреляционной функции.

## Funfun

В данном разделе собраны функции поиска минимумов и нулей функций, численного интегрирования, а также численного решения дифференциальных уравнений. Все эти операции требуют указания функции, с которой будет вестись работа, поэтому данный раздел и получил такое имя — "Function functions". Вот важнейшие функции этого раздела:

- `fzero` — решение нелинейных уравнений;
- `fminsearch` — многомерная нелинейная минимизация без ограничений;
- `fminbnd` — скалярная нелинейная минимизация с ограничениями;
- `quad`, `quadl`, `dblquad`, `triplequad` — численное интегрирование, в том числе двойное и тройное;
- `ode45`, `ode23`, `ode113`, `ode23t`, `ode15s`, `ode23s`, `ode23tb` — различные методы численного решения дифференциальных уравнений.

## General

Данный раздел содержит функции общего назначения. Сюда относятся команды работы со справочной системой, управления рабочей областью памяти, открытия и сохранения файлов, настройки пути поиска файлов, управления командным окном, средства вызова команд операционной системы, а также отладочные средства.

## Graph2d, Graph3d

В данных разделах содержатся средства двумерной и трехмерной графики. Важнейшие из них были рассмотрены в соответствующих разд. *приложения 1*.

## Graphics

В этом разделе собраны средства поддержки *дескрипторной графики* (Handle Graphics®) MATLAB. Вкратце ее идею можно описать следующим образом. Все графические функции, такие как `figure`, `plot` и пр., могут возвращать результат — число или несколько чисел, являющихся *дескрипторами* (handle) созданных функцией графических объектов — графических окон, координатных осей, графиков и т. д. Эти дескрипторы используются для получения программного доступа к графическим объектам и их свойствам. Более подробное описание дескрипторной графики выходит за рамки тематики данной книги. За необходимой информацией обратитесь к справочной системе MATLAB.

## Guide

MATLAB позволяет создавать программы, имеющие графический пользовательский интерфейс. В данном разделе библиотеки находится визуальный конструктор форм — функция `guide`.

## lofun

К данной категории относятся средства файлового ввода/вывода. Важнейшие функции работы с файлами были рассмотрены в разд. "Ввод и вывод данных" приложения 1.

## Lang

В данном разделе собраны конструкции языка программирования MATLAB. Основные элементы языка были описаны в разд. "Программирование" приложения 1. Из того, что не вошло в этот раздел, отметим следующее:

- `try` и `catch` — операторы для обработки исключительных ситуаций (exception);
- `eval(S)` — вычисление MATLAB-выражения, заданного в виде строкового входного параметра `S`;
- `feval(F, x1, x2, ...)` — вычисление функции, имя которой задано строковым входным параметром `F`, с передачей ей входных параметров `x1, x2` и т. д.;
- `persistent` — определение локальной переменной функции, значение которой сохраняется при следующем вызове функции;
- `exist` — проверка существования переменной или функции;
- `isglobal` — проверка глобальности переменной;
- `disp` — вывод значения переменной в командное окно MATLAB;
- `fprintf, sprintf` — запись форматированных данных в поток вывода и в строковую переменную;
- `input` — запрос на ввод значения переменной пользователем;
- `pause` — приостановка работы программы.

## Matfun

В данном разделе собраны матричные функции линейной алгебры. Отметим лишь важнейшие из них:

- `norm` — норма матрицы или вектора;
- `rank` — ранг матрицы;
- `det, trace` — определитель и след (сумма диагональных элементов) матрицы;
- `\ и /` — решение систем линейных уравнений (матричное деление);
- `inv` — обращение матрицы;
- `eig` — расчет собственных чисел и собственных векторов матрицы;
- `svd` — разложение по сингулярным числам;
- `poly` — характеристический полином матрицы;

- `expm`, `logm`, `sqrtm` — матричная экспонента, матричный логарифм, матричный квадратный корень.

## Ops

Данный раздел посвящен операторам и символам, имеющим для MATLAB специальное значение. Поскольку поддержка этих базовых возможностей встроена в ядро системы, в каталоге **Ops** в основном содержится лишь справочная информация. Исключением являются, в частности, функции работы с битовыми представлениями неотрицательных целых чисел и функции, реализующие операции с множествами:

- `bitand`, `bitor`, `bitcmp`, `bitxor` — побитовые логические операции "И", "ИЛИ", "НЕ", "исключающее ИЛИ";
- `bitmax` — максимальное целое число, которое можно точно представить в используемом MATLAB формате с плавающей запятой. Для платформы PC это значение равно  $2^{53} - 1$ ;
- `bitget`, `bitset` — считывание значения бита с заданным номером и его установка в заданное значение;
- `bitshift` — побитовый сдвиг на заданное число разрядов;
- `unique` — удаление из массива повторяющихся элементов;
- `union`, `intersect`, `setdiff`, `setxor` — операции объединения, пересечения, вычитания и "исключающего ИЛИ" над множествами;
- `ismember` — проверка принадлежности элемента к множеству.

## Polyfun

В данном разделе собраны функции интерполяции и работы с полиномами. Перечислим лишь наиболее важные для задач обработки сигналов функции этой категории:

- `roots`, `poly` — поиск корней полинома и расчет коэффициентов полинома по его корням;
- `polyval`, `polyvalm` — вычисление значения полинома в заданной точке от числового и матричного аргумента;
- `residue` — представление дробно-рациональной функции в виде суммы простых дробей (эта функция была рассмотрена в разд. "Преобразование способов описания линейных цепей" главы 2);
- `polyfit` — полиномиальная аппроксимация данных;
- `polyder` — дифференцирование полиномов, их произведений и отношений;
- `polyint` — интегрирование полиномов;
- `conv`, `deconv` — умножение и деление полиномов.

## Sparfun

В некоторых предметных областях приходится иметь дело с матрицами очень большого размера, у которых отлична от нуля лишь малая часть элементов. Такие матрицы называются *разреженными* (sparse). На хранение нулевых элементов и выполнение математических операций с ними впустую тратится много ресурсов компьютера. В данном разделе собраны функции, позволяющие хранить матрицы и выполнять матричные операции непосредственно в разреженном виде, т. е. хранить только значения и индексы ненулевых элементов матриц. В задачах обработки сигналов такие матрицы практически не встречаются, поэтому конкретные функции данного раздела здесь не перечисляются.

## Specfun

Как явствует из названия, в этом разделе находятся специальные математические функции. Вот важнейшие из них:

- airy — функции Эйри;
- besselj, bessely, besselh, besseli, besselk — функции Бесселя;
- beta, betainc, betaln — бета-функция, неполная бета-функция и логарифм бета-функции;
- ellipj — эллиптические функции Якоби;
- ellipke — полный эллиптический интеграл;
- erf, erfc, erfcx, erfinv — функция ошибок: обычная, дополнительная, масштабированная дополнительная, обратная;
- expint — интегральная экспонента;
- gamma, gammainc, gammaln — гамма-функция, неполная гамма-функция и логарифм гамма-функции;
- psi — пси-функция (дигамма-функция);
- legendre — присоединенная функция Лежандра;
- cross — векторное произведение;
- dot — скалярное произведение векторов;
- factor — разложение натурального числа на простые множители;
- primes, isprime — генерация простых чисел и проверка, является ли натуральное число простым;
- gcd, lcm — наибольший общий делитель и наименьшее общее кратное;
- rat — рациональная аппроксимация;
- perms — генерация всех возможных перестановок элементов вектора;
- nchoosek — расчет биномиальных коэффициентов;
- factorial — факториал.

## Specgraph

В данном разделе собраны специализированные графические функции. Сюда относятся разнообразные диаграммы (`area`, `bar`, `barh`, `pie` и др.), рассмотренные нами ранее функции `stem`, `stairs`, `contour` и `contourf`, а также многие другие. Особый интерес представляют функции, имена которых начинаются с букв `ez` (от "easy" — "легкий") — `ezplot`, `ezpolar`, `ezcontour`, `ezcontourf`, `ezgraph3`, `ezmesh`, `ezmeshc`, `ezplot3`, `ezsrf` и `ezsurf`. Они позволяют строить графики соответствующих типов (см. разд. "Графика" приложения 1), задавая не массивы данных, а строки, в которых записаны формулы для вычислений. Также в данном разделе имеются функции чтения и записи графических файлов, трехмерной визуализации и анимации.

## Strfun

К данному разделу относятся функции работы со строками (см. разд. "Строки" приложения 1). Важнейшими из них являются следующие:

- `blanks`, `deblank` — создание строки, заполненной пробелами, и удаление концевых пробелов из строки;
- `strcat` — соединение строк;
- `strcmp`, `strcmpi` — сравнение строк с учетом и без учета регистра символов;
- `strncmp`, `strncmpi` — сравнение первых  $N$  символов строк с учетом и без учета регистра символов;
- `findstr`, `regexp` — поиск одной строки внутри другой и поиск с использованием регулярных выражений;
- `upper`, `lower` — преобразование строки к верхнему и нижнему регистрам;
- `num2str`, `str2num` — преобразование числа в строку и обратно;
- `int2str` — преобразование целого числа в строку;
- `str2double` — преобразование строки в число в формате `double`;
- `sprintf` — запись форматированных данных в строку;
- `hex2dec`, `dec2hex` — преобразование шестнадцатеричной строки в целое число и обратно;
- `bin2dec`, `dec2bin` — преобразование двоичной строки в целое число и обратно;
- `base2dec`, `dec2base` — преобразование строки, записанной в произвольной системе счисления, в целое число и обратно.

## Timefun

В данном разделе собраны функции работы с датами и временем. Важнейшими из них являются следующие:

- `now` — текущие дата и время в виде числа;
- `date` — текущая дата;

- `clock` — текущие дата и время в виде вектора;
- `tic, toc` — запуск и остановка таймера;
- `pause` — приостановка работы программы на заданное время.

## Timeseries

В данном разделе собраны функции, реализующие объекты временных рядов (*time series*) и средства работы с ними.

## Uitools

MATLAB позволяет создавать программы, имеющие графический пользовательский интерфейс. В данном разделе библиотеки собраны средства, обеспечивающие соответствующие функции. Вопросы создания пользовательского интерфейса выходят далеко за рамки тематики данной книги, скажем лишь, что запуск конструктора форм производится функцией `guide`. За дополнительной информацией обратитесь к справочной системе MATLAB.

## Winfun

В данном разделе собраны функции взаимодействия MATLAB с операционной системой Windows через интерфейсы DDE и ActiveX. Работу с ActiveX демонстрируют два примера: `mwsamp` (создание элемента управления ActiveX) и `sampev` (создание обработчика события для сервера ActiveX; эта функция используется примером `mwsamp`).

## **ПРИЛОЖЕНИЕ 3**

### **Компоненты MATLAB**

MATLAB — сложная система, и она постоянно развивается, приобретая все новые возможности. Главная сила MATLAB — это многочисленные пакеты расширения, ориентированные на решение задач в различных предметных областях. В данном приложении перечислены компоненты MATLAB и приведены их краткие описания.

Набор поставляемых фирмой Mathworks компонентов постоянно пополняется. Приводимый далее список соответствует состоянию на май 2010 г. Разделение компонентов на категории в целом соответствует предлагаемому фирмой Mathworks.

Адрес списка компонентов —

[http://www.mathworks.com/products/product\\_listing/index.html](http://www.mathworks.com/products/product_listing/index.html).

#### **MATLAB**

Помимо собственно ядра MATLAB в состав системы входит целый ряд компонентов, предоставляющих дополнительные возможности:

- *MATLAB* — ядро системы MATLAB ("язык технических вычислений", как называют свой продукт разработчики);
- *Parallel Computing Toolbox* и *MATLAB Distributed Computing Server* — средства поддержки распределенных вычислений.

#### **Пакеты расширения MATLAB**

Пакеты расширения MATLAB (toolbox) представляют собой наборы функций, объединенные общей тематикой и ориентированные на решение задач конкретной предметной области.

#### **Математика и оптимизация**

К данной группе относятся пакеты общематематического характера, реализующие ряд численных алгоритмов, а также позволяющие выполнять символьные (аналитические) вычисления:

- *Optimization Toolbox* — функции, реализующие разнообразные численные методы оптимизации, в том числе при наличии линейных и нелинейных ограничений [25];
- *Symbolic Math Toolbox* — реализация аналитических расчетов с использованием ядра MuPAD®;
- *Partial Differential Equation Toolbox* — решение дифференциальных уравнений в частных производных [25];
- *Global Optimization Toolbox* — реализация разнообразных алгоритмов поиска глобальных экстремумов в многоэкстремальных задачах. В частности, здесь реализованы генетические алгоритмы и алгоритмы прямого поиска.

## Статистика и анализ данных

В данную группу входят пакеты, реализующие анализ разнообразных наборов данных:

- *Statistics Toolbox* — функции статистического моделирования и проверки гипотез [25];
- *Neural Network Toolbox* — функции моделирования и анализа нейронных сетей;
- *Curve Fitting Toolbox* — функции аппроксимации;
- *Spline Toolbox* — функции кусочно-полиномиальной (сплайновой) аппроксимации и интерполяции;
- *Model-Based Calibration Toolbox* — средства калибровки сложных моделей для проверки их соответствия экспериментальным данным.

## Анализ и синтез систем управления

Системы с обратной связью (системы управления), как и обработка сигналов, являются одной из старейших областей применения MATLAB. В настоящее время для анализа и проектирования таких систем имеется целый ряд пакетов расширения:

- *Control System Toolbox* — функции моделирования и анализа систем автоматического управления (систем с обратной связью) [29];
- *System Identification Toolbox* — функции идентификации линейных динамических систем, т. е. создания их математических моделей на основе измеренных входных и выходных сигналов;
- *Fuzzy Logic Toolbox* — функции моделирования и анализа систем с нечеткой логикой;
- *Robust Control Toolbox* — функции моделирования и анализа систем управления, устойчивых по отношению к случайным воздействиям;
- *Model Predictive Control Toolbox* — функции моделирования и анализа систем управления с большим числом входных и выходных параметров при наличии ограничений;

- *Aerospace Toolbox* — функции расчета и моделирования аэрокосмических систем.

## Обработка сигналов и телекоммуникации

Пакет Signal Processing входил в число первых пакетов расширения MATLAB. В настоящее время функции для решения задач обработки сигналов и родственные средства, связанные с системами связи, собраны в следующих пакетах расширения, многие из которых были рассмотрены в этой книге:

- *Signal Processing Toolbox* — функции анализа и обработки сигналов;
- *Communications Toolbox* — функции анализа и моделирования систем связи;
- *Filter Design Toolbox* — функции анализа и синтеза фильтров, в том числе с учетом эффектов квантования и арифметики с фиксированной запятой;
- *Filter Design HDL Coder* — генератор кода VHDL или Verilog для реализации фильтров с фиксированной запятой, разработанных с помощью пакета Filter Design Toolbox;
- *Wavelet Toolbox* — функции анализа, очистки от шума и компрессии сигналов и изображений с использованием вейвлет-алгоритмов;
- *Fixed-Point Toolbox* — поддержка вычислений с фиксированной запятой;
- *RF Toolbox* — функции и графический интерфейс для разработки и анализа систем, составленных из высокочастотных (ВЧ) компонентов (фильтров, линий передачи, усилителей, смесителей и т. п.), на уровне параметров многополюсников и физических свойств.

## Обработка изображений

Важной современной областью обработки сигналов является обработка многомерных сигналов, к которым относятся, в частности, изображения (с теоретической точки зрения они представляют собой двумерные сигналы). Для работы с изображениями предназначены следующие пакеты расширения:

- *Image Processing Toolbox* — функции анализа и обработки изображений;
- *Image Acquisition Toolbox* — функции взаимодействия с устройствами ввода изображений, такими как видеокамеры и платы видеозахвата;
- *Mapping Toolbox* — пакет расширения для визуализации географических карт и работы с другой географической информацией.

## Тесты и измерения

К данной группе относятся пакеты, реализующие взаимодействие с различными системами тестирования и сбора данных:

- *Data Acquisition Toolbox* — средство взаимодействия с оборудованием аналогового и цифрового ввода/вывода данных;

- *Instrument Control Toolbox* — средство взаимодействия MATLAB с современной измерительной аппаратурой через протоколы GPIB и VISA;
- *SystemTest* — средство разработки тестовых процедур для MATLAB-алгоритмов и Simulink-моделей;
- *OPC Toolbox* — данный пакет расширения позволяет быстро прототипировать аналитические, мониторинговые, оптимизационные и управляющие приложения, использующие производственные данные в реальном масштабе времени. Пакет позволяет подключаться к серверам OPC, а также считывать и записывать OPC-данные с помощью MATLAB;
- *Vehicle Network Toolbox* — данный пакет реализует средства взаимодействия с бортовыми сетями передачи данных современных автомобилей (стандарт CAN).

## Вычислительная биология

Недавнее появление данной группы, состоящей в настоящий момент из двух пакетов, во многом было обусловлено интенсивными научными работами по расшифровке геномов растений, животных и человека:

- *Bioinformatics Toolbox* — функции для решения задач вычислительной молекулярной биологии (анализ генетической информации и т. п.);
- *SimBiology* — графические и программные средства для решения вычислительных задач биологии и фармакокинетики.

## Финансовое моделирование и анализ

С математической точки зрения финансовые расчеты сводятся к статистическому анализу и анализу временных рядов (последний термин фактически является синонимом обработки сигналов). Тем не менее, для финансового анализа имеется ряд специфических средств:

- *Financial Toolbox* — функции для моделирования финансовых данных и разработки алгоритмов финансового анализа;
- *Financial Derivatives Toolbox* — функции анализа и моделирования процентных ставок и рисков;
- *Datafeed Toolbox* — средство получения финансовых данных от поставщиков информации в реальном времени;
- *Fixed-Income Toolbox* — функции для моделирования и анализа финансовых инструментов с фиксированным доходом;
- *Econometrics Toolbox* — функции для моделирования экономических данных.

## Разработка приложений

Хотя MATLAB по своей сути является интерпретирующей системой, в настоящее время в нем имеется и возможность компиляции кода, т. е. создания автономных программ:

- *MATLAB Compiler* — средство создания независимых приложений;
- *Spreadsheet Link EX* — средство интеграции MATLAB и Microsoft Excel.

## Целевые платформы для разработки приложений

Помимо автономных приложений, MATLAB позволяет создавать и программные модули различных типов:

- *MATLAB Builder EX* — средство создания внешних функций и макросов для Microsoft Excel;
- *MATLAB Builder NE* — средство создания компонентов .NET и COM-объектов;
- *MATLAB Builder JA* — средство создания Java-классов.

## Связь с базами данных и генерирование отчетов

Пакеты этой группы реализуют интерфейс с базами данных и позволяют генерировать отчеты:

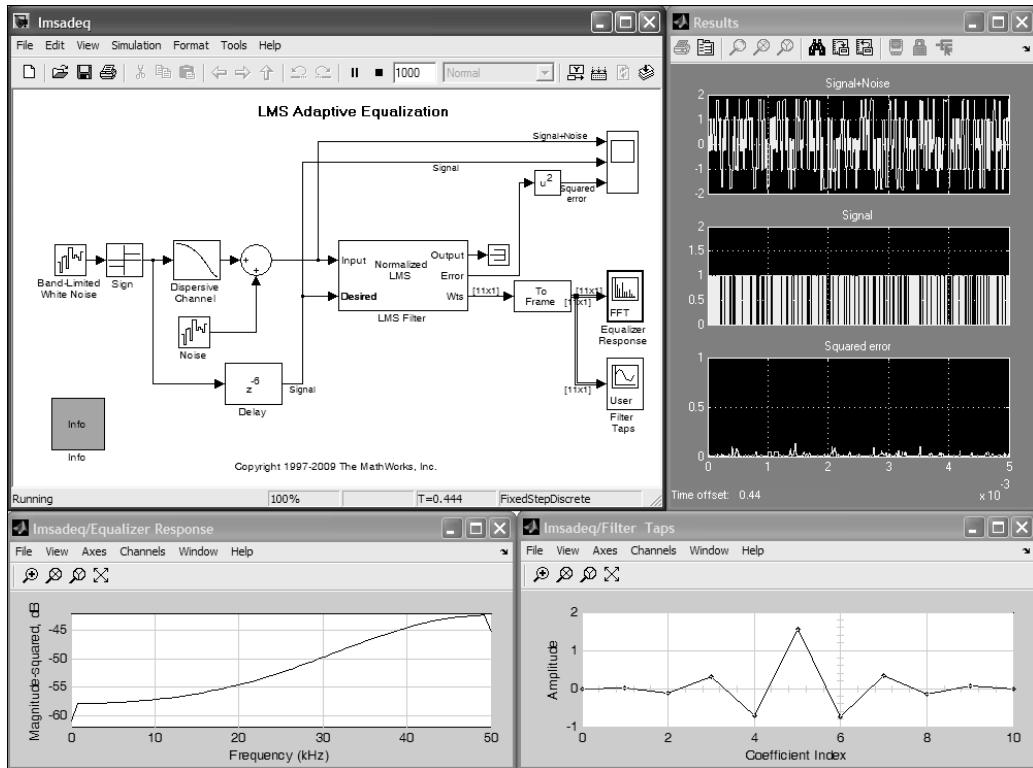
- *Database Toolbox* — средство обмена данными с реляционными базами данных;
- *MATLAB Report Generator* — средство создания документации для MATLAB-приложений и данных.

## Simulink

С технической точки зрения Simulink® является расширением MATLAB, однако он предоставляет столько новых возможностей, что его следует рассматривать как отдельный программный продукт. Simulink — это графическая среда моделирования аналоговых и дискретных систем. Моделирование производится путем перетаскивания блоков из окон библиотек в окно создаваемой модели и настройки связей между ними. Большое число имеющихся библиотек позволяет моделировать самые разнообразные электрические, механические, гидравлические и другие системы. После создания модели можно запустить процесс моделирования. Simulink создаст систему дифференциальных уравнений, описывающих модель, и начнет выполнять ее решение численным методом. В качестве примера на рис. П3.1 показана модель адаптивного эквалайзера, использующего LMS-алгоритм, рассмотренный в главе 9 (это один из демонстрационных примеров набора блоков Signal Processing Blockset). Кроме окна модели на рисунке видны окна трехканального осциллографа и графиков частотной и импульсной характеристик фильтра. В процессе моделирования графики постоянно обновляются. Подробнее о системе Simulink можно прочитать в книге [29].

Simulink в настоящее время представляет собой весьма сложную систему, состоящую из нескольких отдельно устанавливаемых компонентов:

- *Simulink*® — ядро системы Simulink, среда моделирования систем непрерывного и дискретного времени;



**Рис. П3.1.** Simulink-модель и окна графиков, динамически обновляемых в процессе моделирования

- *Simulink Report Generator* — средство создания документации для моделей Simulink и Stateflow;
- *Simulink Fixed Point* — средства реализации арифметики с фиксированной запятой в моделях Simulink;
- *Stateflow®* — средство моделирования диаграмм состояний и потоков данных;
- *SimEvents* — средство моделирования систем, управляемых событиями.

## Наборы блоков Simulink

Наборы блоков Simulink (*blockset*) представляют собой тематические библиотеки, ориентированные на конкретные предметные области.

### Физическое моделирование

В эту группу объединены наборы блоков, реализующие моделирование физических процессов различной природы:

- *Simscape* — средства моделирования систем смешанной физической природы;
- *SimMechanics* — средства моделирования механических систем [29];

- *SimPowerSystems* — средства моделирования энергетических систем [29, 30];
- *SimDriveline* — средства моделирования механических трансмиссий;
- *SimHydraulics* — средства моделирования гидравлических систем;
- *SimElectronics* — средства моделирования электронных и электромеханических систем.

## Имитационная графика

Визуализация результатов является важным аспектом моделирования, и наборы блоков данной группы позволяют реализовать более сложные варианты, нежели простое отображение графиков и числовых значений:

- *Simulink 3D Animation* — средства создания виртуальных трехмерных моделей на основе языка VRML;
- *Gauges Blockset* — набор графических блоков измерительных приборов для отображения сигналов и параметров моделирования.

## Анализ и синтез систем управления

Как и аналогичные по назначению пакеты расширения, эти наборы блоков представляют одну из важнейших прикладных областей использования MATLAB/Simulink:

- *Simulink Control Design* — средства выполнения линейного анализа нелинейных моделей систем управления;
- *Aerospace Blockset* — блоки для моделирования авиационных, космических, а также прочих ракетных и реактивных систем [29];
- *Simulink Design Optimization* — средства оптимизации параметров моделей систем управления.

## Обработка сигналов и телекоммуникации

Наборы средств моделирования обработки сигналов и передачи данных, предоставляемые пакетами расширения MATLAB и наборами блоков Simulink, во многом совпадают:

- *Signal Processing Blockset* — блоки для моделирования систем цифровой обработки сигналов;
- *Communications Blockset* — блоки для моделирования систем связи;
- *RF Blockset* — блоки для моделирования высокочастотных устройств радиосистем методом низкочастотных эквивалентов (комплексных огибающих): усилителей, смесителей, фильтров, линий передачи и т. п.;
- *Video and Image Processing Blockset* — блоки для моделирования систем обработки подвижных и неподвижных изображений.

## Генерирование исполняемого кода

Одно из активно развивающихся направлений применения Simulink — автоматическое создание с его помощью программ для микроконтроллеров и "прошивок" для программируемой логики:

- *Real-Time Workshop®* — средство генерации С-кода на основе Simulink-моделей;
- *Real-Time Workshop Embedded Coder* — средство генерации высококачественного исходного кода для встраиваемых систем;
- *Stateflow Coder* — средство генерации С-кода на основе диаграмм Stateflow;
- *Simulink HDL Coder* — средство генерации кода Verilog и VHDL;
- *Target Support Package* — средства взаимодействия со средами разработки программ для микропроцессоров, микроконтроллеров и цифровых сигнальных процессоров различных производителей;
- *Simulink PLC Coder* — средства генерации кода, соответствующего стандарту IEC 61131, для программируемых логических контроллеров (Programmable Logic Controller, PLC) и программируемых контроллеров автоматизации (Programmable Automation Controller, PAC).

## Быстрое прототипирование и аппаратно-программное моделирование (Hardware-in-the-Loop)

Реализация взаимодействия моделей Simulink с различными аппаратными средствами в реальном масштабе времени — важнейший метод разработки прототипов разнообразных приложений:

- *xPC Target* — средство быстрого прототипирования приложений реального времени, использующих стандартные аппаратные средства персональных компьютеров;
- *xPC Target Embedded Option* — средство создания и распространения приложений реального времени, использующих стандартные аппаратные средства персональных компьютеров;
- *Real-Time Windows Target* — среда моделирования, обеспечивающая выполнение моделей Simulink и Stateflow на персональном компьютере в реальном времени.

## Верификация, проверка и тестирование моделей

К данной группе относятся компоненты Simulink, позволяющие реализовывать разнообразные тестовые процедуры:

- *Simulink Verification and Validation* — данное средство позволяет создавать в Simulink и Stateflow системы, основанные на формальных технических требованиях, разрабатывать тестовые примеры и измерять степень покрытия для этих тестов. Это позволяет отслеживать выполнение требований, проверять качество

систем, обнаруживать неадекватные требования, находить дефекты в разработанных системах;

- *Simulink Design Verifier* — генератор тестов для моделей Simulink и Stateflow;
- *Embedded IDE Link* — средство взаимодействия со средами разработки программ для встраиваемых систем различных производителей;
- *EDA Simulator Link* — средство взаимодействия со средами моделирования Mentor Graphics, Cadence и Synopsys.

## ПРИЛОЖЕНИЕ 4

### Программа SPTool

Программа SPTool (Signal Processing Tool) предоставляет в распоряжение пользователя графическую среду для просмотра графиков сигналов и их спектров, расчета и анализа фильтров, а также фильтрации сигналов. Перечисленные действия относятся к тематике разных глав данной книги, поэтому описание программы SPTool размещено в приложении.

Для запуска программы SPTool необходимо набрать ее имя в командной строке MATLAB:

```
>> sptool
```

Появится основное окно программы, показанное на рис. П4.1.

В трех списках этого окна перечислены загруженные в программу SPTool сигналы (**Signals**), фильтры (**Filters**) и спектры (**Spectra**). Расположенные под списками кнопки позволяют выполнять различные операции, большинство из которых мы рассмотрим далее.

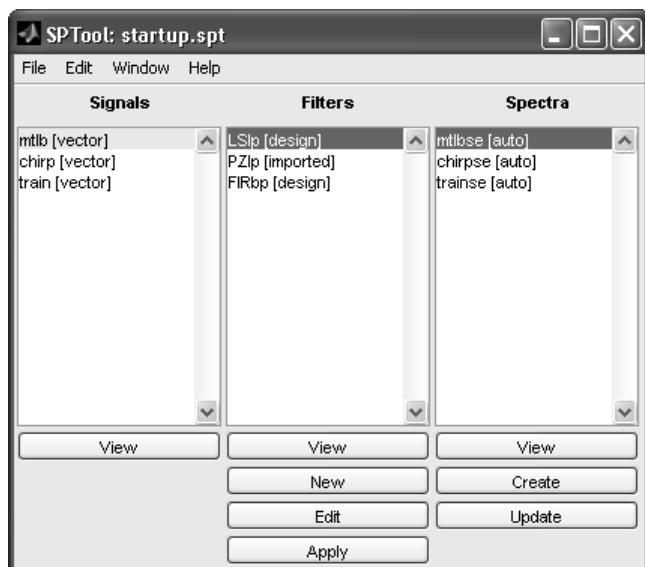


Рис. П4.1. Программа SPTool — главное окно

Типичный набор действий, выполняемых с помощью программы SPTool, включает в себя такие операции, как:

- загрузка сигнала;
- просмотр графика сигнала;
- спектральный анализ сигнала;
- расчет фильтра;
- фильтрация сигнала.

Далее мы кратко рассмотрим выполнение всех перечисленных процедур.

## Загрузка сигнала

Для загрузки сигнала в меню **File** главного окна программы SPTool необходимо выбрать команду **Import**. Появится окно **Import to SPTool**, показанное на рис. П4.2.

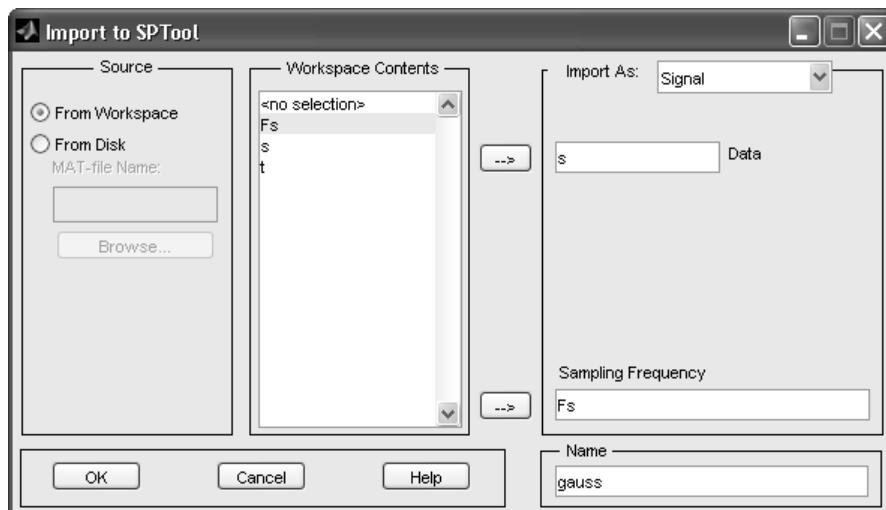


Рис. П4.2. Программа SPTool — окно импорта данных

Переключатель группы **Source** позволяет указать место хранения загружаемого сигнала — рабочую область MATLAB (**From Workspace**) или MAT-файл (**From Disk**). Во втором случае становится доступным поле ввода **MAT-file Name**, в которое нужно ввести вручную или с помощью кнопки **Browse** имя нужного MAT-файла.

В списке группы **Workspace Contents** перечислены переменные, имеющиеся в рабочей памяти MATLAB в данный момент. Если в качестве источника был выбран MAT-файл, этот список будет называться **File Contents** и перечислять переменные, сохраненные в выбранном файле.

В раскрывающемся списке группы **Import As** выберите вариант **Signal**. Остальные два варианта, **Filter** и **Spectrum**, позволяют импортировать описания фильтров и уже рассчитанных спектров для просмотра и анализа.

Выберите в списке переменную, содержащую отсчеты загружаемого сигнала, и щелкните на кнопке  $\rightarrow$ , расположенной рядом с полем ввода **Data**. Можно также вручную ввести в это поле идентификатор переменной.

В поле ввода **Sampling Frequency** по умолчанию находится значение 1. Его можно отредактировать вручную, а можно импортировать, введя идентификатор переменной или выбрав его в списке и щелкнув на нижней кнопке  $\rightarrow$ .

Наконец, в поле ввода группы **Name** можно отредактировать имя, под которым данный сигнал будет фигурировать в программе SPTool. Сделав это, щелкните на кнопке **OK**. Импортированный сигнал появится в списке **Signals** основного окна программы (см. рис. П4.1).

## Просмотр графика сигнала

Для просмотра графика сигнала выберите его в списке **Signals** (см. рис. П4.1) и щелкните на кнопке **View**, расположенной под этим списком. Появится окно **Signal Browser**, показанное на рис. П4.3.

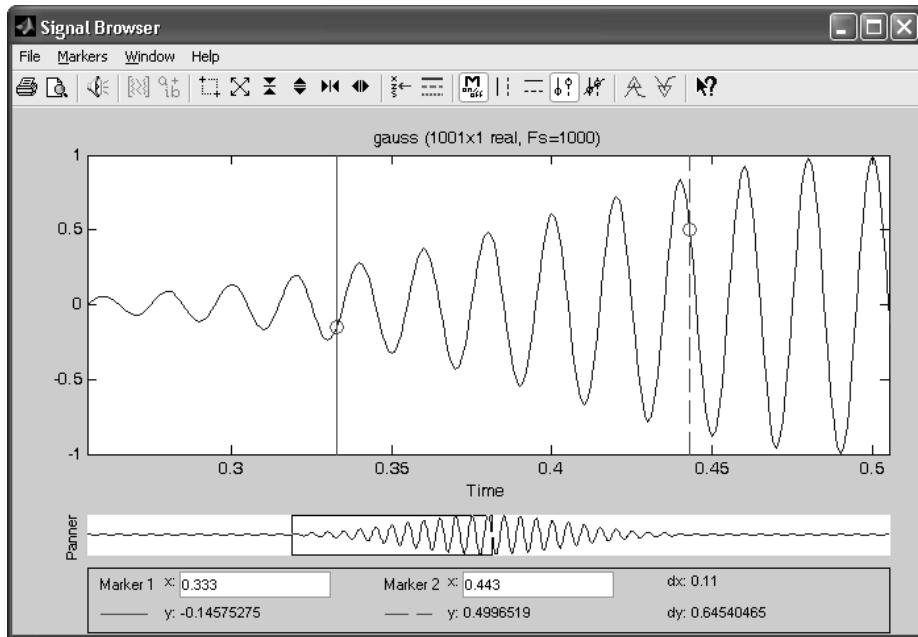


Рис. П4.3. Программа SPTool — окно просмотра сигналов

В данном окне выводятся два графика. Нижний, с надписью **Panner**, показывает панораму всего сигнала. Верхний график первоначально тоже отображает весь сигнал, но масштаб отображения можно увеличить, и тогда верхний график будет по-

казывать лишь фрагмент сигнала, а положение этого фрагмента на общей панораме демонстрируется с помощью прямоугольника на нижнем графике. Этот прямоугольник можно перетаскивать мышью, меняя тем самым участок обзора.

Кнопки панели инструментов окна **Signal Browser** позволяют распечатывать график сигнала, воспроизводить сигнал с помощью звуковой карты компьютера, управлять масштабом отображения, выбирать нужный канал в случае многоканального сигнала и задавать нужный режим отображения маркеров.

На график можно нанести два маркера, позволяющие производить над сигналом количественные измерения. Маркеры перетаскиваются с помощью мыши, а информация о помеченных ими отсчетах сигнала выводится в нижней части окна. Можно экспорттировать эту информацию в рабочую область памяти MATLAB в виде структуры — для этого служит команда **Export** меню **Markers**.

## Спектральный анализ сигнала

Для анализа спектра сигнала, загруженного в программу SPTool, выберите нужный сигнал в списке **Signals** главного окна программы (см. рис. П4.1) и щелкните на кнопке **Create**, расположенной под списком **Spectra**. Появится окно **Spectrum Viewer**, показанное на рис. П4.4.

В левой части окна выбирается метод спектрального анализа и настраиваются его параметры. За более подробной информацией о сущности различных методов и имеющихся у них параметрах обратитесь к материалу главы 5.

Раскрывающийся список, в котором первоначально выведена строка **Inherit from**, позволяет скопировать полный набор настроек анализатора спектра из другого расчета, представленного в списке **Spectra** основного окна программы.

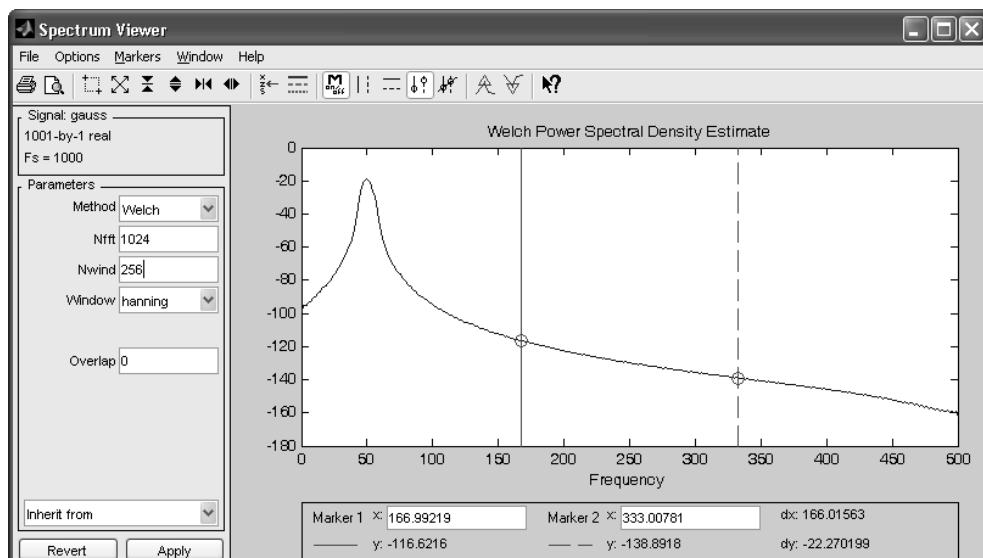


Рис. П4.4. Программа SPTool — окно спектрального анализа

Произведя настройку параметров анализа, щелкните на кнопке **Apply**. Будет рассчитана оценка спектра сигнала и выведен соответствующий график. Кнопки панели инструментов позволяют выполнять те же операции, что были перечислены ранее применительно к просмотру графиков сигналов. При просмотре спектра также возможно использование маркеров.

## Расчет фильтра

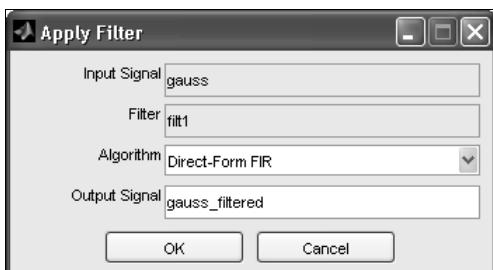
Для расчета дискретного фильтра щелкните на кнопке **New**, расположенной под списком **Filters** в главном окне программы SPTool (см. рис. П4.1). Можно также изменить параметры уже рассчитанного фильтра, выбрав его в списке **Filters** и щелкнув на кнопке **Edit**. В любом из перечисленных случаев появится окно программы FDATool, рассмотренной в главе 6.

## Просмотр характеристик фильтра

Для просмотра характеристик фильтра, загруженного в программу SPTool, выберите его в списке **Filters** основного окна программы (см. рис. П4.1) и щелкните на кнопке **View**, расположенной под этим списком. Появится окно визуализатора фильтров FVTool, рассмотренного в главе 4.

## Фильтрация сигнала

Для пропускания сигнала через фильтр необходимо выбрать сигнал и фильтр соответственно в списках **Signals** и **Filters** основного окна программы (см. рис. П4.1), а затем щелкнуть на кнопке **Apply**, расположенной под списком **Filters**. Появится окно **Apply Filter**, показанное на рис. П4.5.

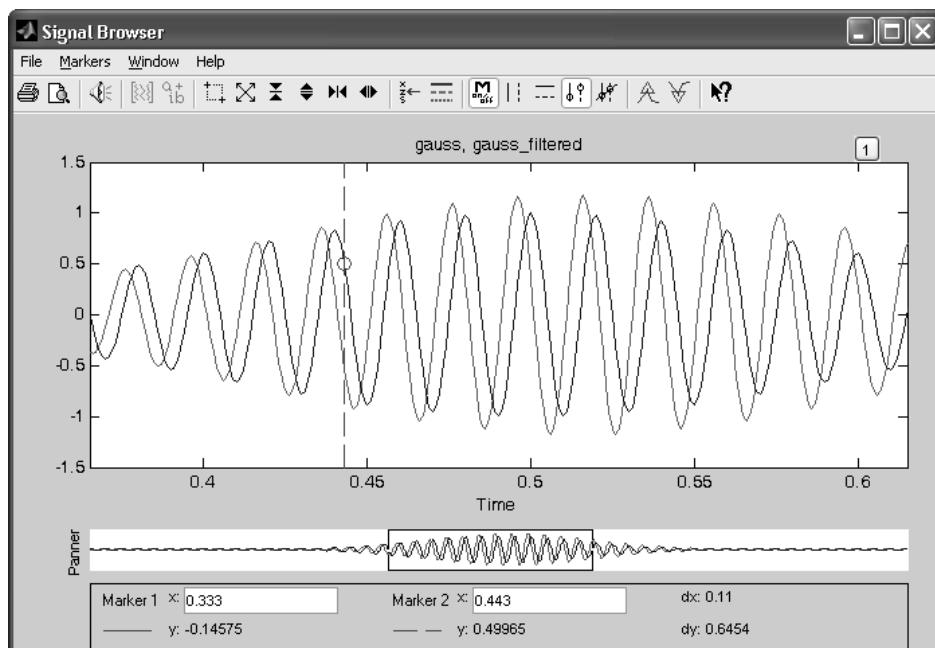


**Рис. П4.5.** Программа SPTool — окно применения фильтра

Единственным доступным полем ввода в этом окне является поле **Output Signal**, в котором нужно задать имя для выходного сигнала. В раскрывающемся списке **Algorithm** можно выбрать способ осуществления фильтрации — непосредственную реализацию разработанной в FDATool структуры фильтра (в данном примере — **Direct-Form FIR**), для нерекурсивных фильтров — фильтрацию в частотной области с помощью функции `fftfilt` (см. главу 5), для рекурсивных фильтров — фильтрацию с нулевым фазовым сдвигом с помощью функции `filtfilt` (см. главу 5).

бу 4). Выполнив эти действия, щелкните на кнопке **OK**. Будет рассчитан выходной сигнал, который появится под указанным именем в списке **Signals** основного окна программы.

Просмотреть график выходного сигнала и выполнить анализ его спектра можно описанными ранее способами. Для одновременного просмотра графиков входного и выходного сигналов нужно выбрать их оба в списке **Signals** основного окна программы (для этого при щелчке на выбранном сигнале необходимо, как это принято в Windows, нажать клавишу <Ctrl>) и щелкнуть на кнопке **View**. В открывшемся окне **Signal Browser** будут показаны графики обоих выбранных сигналов (рис. П4.6).



**Рис. П4.6.** Программа SPTool — одновременный просмотр графиков входного и выходного сигналов фильтра

#### ЗАМЕЧАНИЕ

Аналогичным образом можно просматривать несколько спектров или характеристики нескольких фильтров одновременно.

## Сохранение результатов работы

Сеанс работы с программой SPTool можно сохранить с помощью команды **Save Session** или **Save Session As** из меню **File** основного окна программы. Файлы сеансов имеют расширение **spt**. Загрузить сохраненный сеанс можно командой **Open Session** того же меню **File**.

Кроме того, можно экспортить сигналы, фильтры и спектры в виде структур данных. Для этого используется команда **Export** из меню **File** основного окна программы SPTool. После выбора данной команды появится окно **Export from SPTool**, показанное на рис. П4.7.



Рис. П4.7. Программа SPTool — окно экспорта данных

В списке, занимающем левую часть этого окна, необходимо выбрать экспортируемые объекты. Щелчок на кнопке **Export to workspace** произведет экспорт соответствующих структур данных в рабочую область памяти MATLAB, а для их записи в MAT-файл необходимо воспользоваться кнопкой **Export to disk**.

Флажок **Export filters as TF objects** появляется при установленном пакете расширения Control System, он позволяет экспортить фильтры в виде объектов класса **tf** (Transfer Function — функция передачи), поддерживаемого этим пакетом.

Информацию о составе экспортируемых структур данных можно получить из справочной системы пакета Signal Processing.

# Литература

Я решил несколько отойти от традиции, согласно которой список литературы должен выглядеть как монотонный и безликий перечень. Вместо этого я разбил список на несколько групп и дал комментарии к каждой группе, постаравшись пояснить, с какой целью включена в список та или иная книга и какую пользу сможет извлечь из нее читатель.

В список литературы помещались только источники, доступные читателю, поэтому в нем отсутствуют зарубежные издания, журнальные статьи и внутривузовские учебные пособия. Кроме того, список отражает личные предпочтения автора и ни в коей мере не претендует на абсолютную полноту.

## Радиотехника

К этой группе относятся книги, посвященные преимущественно теоретической радиотехнике. В них можно найти более глубокое теоретическое обоснование материала глав 1—2. Книги [1] и [2] — это последние издания двух классических учебников по курсу "Радиотехнические цепи и сигналы". Двухтомный курс лекций [3] представляет собой зарубежный подход к этим вопросам. Книга [4] — учебное пособие на ту же тему, в составлении которого принимал участие и автор данной книги. Монография [5] — фундаментальный курс по теории сигналов, где подробно изложены принципы построения *пространства сигналов* и решения оптимизационных задач в этом пространстве. Наконец, [6] — поистине великолепная книга, помогающая пытливо и непредвзято взглянуть на некоторые вопросы теории связи и информации, получив при этом весьма неожиданные на первый взгляд результаты. Эту книгу должен прочесть каждый, кто собирается серьезно заниматься обработкой сигналов или телекоммуникациями.

1. Гоноровский И. С. Радиотехнические цепи и сигналы: учеб. пособие для вузов. — 5-е изд., испр. и доп. — М.: Дрофа, 2006.
2. Баскаков С. И. Радиотехнические цепи и сигналы: учеб. для вузов по спец. "Радиотехника". — 4-е изд., перераб. и доп. — М.: Высшая школа, 2003.
3. Сиберт У. М. Цепи, сигналы, системы: В 2-х ч.: пер. с англ. — М.: Мир, 1988.

4. Иванов М. Т., Сергиенко А. Б., Ушаков В. Н. Теоретические основы радиотехники: учеб. пособие / под ред. В. Н. Ушакова. — 2-е изд., стер. — М.: Высш. шк., 2008.
5. Френкс Л. Теория сигналов: пер. с англ. / под ред. Д. Е. Вакмана. — М.: Сов. радио, 1974.
6. Финк Л. М. Сигналы, помехи, ошибки... Заметки о некоторых неожиданностях, парадоксах и заблуждениях в теории связи. — 2-е изд., перераб. и доп. — М.: Радио и связь, 1984.

## Цифровая обработка сигналов

К данной группе относятся книги, речь в которых идет о собственно цифровой обработке сигналов. Их, в свою очередь, можно разделить на несколько категорий. Заметное место здесь занимают изданные в 70-е—80-е годы переводы зарубежных книг [7—14], ставших классикой цифровой обработки сигналов. В книгах [7—9] подробно изложены теория дискретных сигналов, дискретной фильтрации и дискретного спектрального анализа. Несмотря на свой солидный возраст, эти книги не потеряли актуальности — разве что главы, посвященные аппаратной реализации цифровой обработки сигналов, сейчас могут вызвать лишь улыбку (ностальгическую или снисходительную — в зависимости от возраста читателя). Об их непреходящей ценности свидетельствует и тот факт, что, скажем, книги [7, 8] включены в список литературы к курсу "Discrete-Time Signal Processing", читаемому в MIT (Massachusetts Institute of Technology, Массачусетский технологический институт — одно из ведущих высших учебных заведений мира в области радиотехники, телекоммуникаций и вычислительной техники). В книге [10] обсуждаются специфические вопросы, связанные с обработкой речевых сигналов; в частности, в ней подробно рассматривается техника линейного предсказания. Предметом книги [11] являются быстрые алгоритмы, в том числе алгоритмы быстрого преобразования Фурье (см. главу 5). Книга [12] посвящена цифровому спектральному анализу, в ней можно найти подробное теоретическое обоснование вопросов, обсуждавшихся в главе 5. В книгах [13] и [14] рассматриваются *адаптивные* фильтры, параметры которых автоматически подстраиваются в соответствии со статистическими свойствами обрабатываемых сигналов (см. главу 9). Хорошим вводным курсом в предмет цифровой обработки сигналов, удачно дополняющим радиотехнические учебники [1—3], является отечественная книга [15], также относящаяся к указанному периоду.

После провальных 90-х годов ситуация с книгами по цифровой обработке сигналов начинает постепенно улучшаться. Издан перевод нового издания одной из классических книг [16]. Переведенный с японского языка вводный курс [17] можно рекомендовать читателям, не имеющим серьезной математической подготовки. В весьма объемной книге [18] подробно рассмотрены тонкости возможных способов реализации разнообразных алгоритмов. Учебное пособие [19], подготовленное сотрудниками кафедры цифровой обработки сигналов Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича, вы-

деляется детальностью теоретических рассмотрений; это же относится и к книге [20]. Книги [21, 22] сфокусированы на задачах проектирования цифровых фильтров. Наконец, изюминкой книги [23] является глава "Маленькие хитрости цифровой обработки сигналов", в которой рассматриваются несколько десятков оригинальных приемов, позволяющих повысить эффективность различных алгоритмов.

7. Голд Б., Рэйдер Ч. Цифровая обработка сигналов: пер. с англ. / под ред. А. М. Трахтмана. — М.: Сов. радио, 1973.
8. Рабинер Л., Гоулд Б. Теория и применение цифровой обработки сигналов: пер. с англ. / под ред. Ю. И. Александрова. — М.: Мир, 1978.
9. Применение цифровой обработки сигналов. Под ред. Э. Оппенгейма: пер. с англ. / под ред. А. М. Рязанцева. — М.: Мир, 1980.
10. Рабинер Л. Р., Шафер Р. В. Цифровая обработка речевых сигналов: пер. с англ. / под ред. М. В. Назарова и Ю. Н. Прохорова. — М.: Радио и связь, 1981.
11. Блейхут Р. Быстрые алгоритмы цифровой обработки сигналов: пер. с англ. — М.: Мир, 1989.
12. Марпл-мл. С. Л. Цифровой спектральный анализ и его приложения: пер. с англ. — М.: Мир, 1990.
13. Адаптивные фильтры: пер. с англ. / под ред. К. Ф. Н. Коуэна и П. М. Гранта. — М.: Мир, 1988.
14. Уидроу Б., Стирнз С. Д. Адаптивная обработка сигналов. — М.: Радио и связь, 1989.
15. Карташев В. Г. Основы теории дискретных сигналов и цифровых фильтров. — М.: Высшая школа, 1982.
16. Оппенгейм А., Шафер Р. Цифровая обработка сигналов: пер. с англ. / под ред. А. Б. Сергиенко. — 2-е изд., испр. — М.: Техносфера, 2007.
17. Юкио Сато, под ред. Есифуми Амэмия. Обработка сигналов. Первое знакомство: пер. с японского. — М.: Додэка-XXI, 2002.
18. Айфичер Э. С., Джервис Б. У. Цифровая обработка сигналов: практический подход: пер. с англ. — 2-е изд. — М.: Издательский дом "Вильямс", 2004.
19. Основы цифровой обработки сигналов: Курс лекций / А. И. Солонина, Д. А. Улахович, С. М. Арбузов, Е. Б. Соловьева. — 2-е изд., испр. и перераб. — СПб.: БХВ-Петербург, 2005.
20. Гадзиковский В. И. Теоретические основы цифровой обработки сигналов. — М.: Радио и связь, 2004.
21. Гадзиковский В. И. Методы проектирования цифровых фильтров. — М.: Горячая линия — Телеком, 2007.
22. Васильев В. П., Муро Э. Л., Смольский С. М. Основы теории и расчета цифровых фильтров: учеб. пособие для высших учеб. заведений. — М.: Издательский центр "Академия", 2007.

23. Лайонс Р. Цифровая обработка сигналов: пер. с англ. — 2-е изд. — М.: ООО "Бином-Пресс", 2006.

## MATLAB

В данную группу собраны книги, посвященные системе MATLAB. За последнее время издано множество таких книг, поэтому ограничимся лишь несколькими названиями. Книга [24] — весьма подробное описание седьмой версии MATLAB. Учебное пособие [25] посвящено общематематическим аспектам использования MATLAB. Для обработки сигналов из рассмотренного в книге наиболее интересны пакеты Optimization (Оптимизация) и Statistics (Статистика). Далее идут книги о применении MATLAB для решения различных прикладных задач — анализа и синтеза электрических цепей [26], реализации вейвлет-анализа [27], двумерной обработки сигналов [28], моделирования физических процессов и технических систем [29], создания моделей электротехнических устройств [30]. Две последние книги, помимо прочего, содержат описание Simulink — расширения MATLAB, предназначенного для моделирования динамических систем (см. *приложение 3*).

Особое место занимает книга [31], которая может быть отнесена и к предыдущей категории, т. к. содержит и теоретические справки о рассматриваемых разделах цифровой обработки сигналов, и большое количество MATLAB-примеров.

24. Ануфриев И., Смирнов А., Смирнова Е. MATLAB 7. Наиболее полное руководство. — СПб.: БХВ-Петербург, 2005.
25. Иглин С. П. Математические расчеты на базе MATLAB. — СПб.: БХВ-Петербург, 2005.
26. Новгородцев А. Б. Расчет электрических цепей в MATLAB: Учебный курс. — СПб.: Питер, 2004.
27. Смоленцев Н. К. Основы теории вейвлетов. Вейвлеты в MATLAB. — М.: ДМК Пресс, 2008.
28. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB: пер. с англ. — М.: Техносфера, 2006.
29. Лазарев Ю. Моделирование процессов и систем в MATLAB. Учебный курс. — СПб.: Питер, Издательская группа BHV, 2005.
30. Черных И. В. Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink. — СПб.: Питер, 2008.
31. Солонина А. И., Арбузов С. М. Цифровая обработка сигналов. Моделирование в MATLAB. — СПб.: БХВ-Петербург, 2008.

## Разное

К данной группе отнесены источники, не попавшие в предыдущие категории. Книги [32, 33] содержат описание упоминавшейся в *приложении 1* системы верстки текстов с математическими формулами LaTeX. Работа [34] посвящена численным

методам оптимизации, которые могут использоваться, в частности, при прямом синтезе дискретных фильтров (см. главу 6). Монографии [35, 36] посвящены не цифровой обработке сигналов как таковой, а системам цифровой связи. Однако работа таких систем основана на технологиях цифровой обработки сигналов, и в указанных книгах можно найти подробное обсуждение вопросов, связанных с процессами модуляции/демодуляции (см. главу 8), а также с применением адаптивных фильтров для компенсации искажений, вносимых каналом связи (см. главу 9). В книгах [37—40] рассматриваются вопросы, связанные с реализацией алгоритмов цифровой обработки сигналов на специализированных цифровых сигнальных процессорах. Источники [41, 42] дают читателю возможность поближе познакомиться с вейвлет-анализом — активно развивающейся областью математики, позволяющей, в частности, осуществлять частотно-временной анализ сигналов. Наконец, работы [43, 44] посвящены нелинейным обучаемым системам — нейронным сетям.

32. Беляков Н. С., Палош В. Е., Садовский П. А. ТЕХ для всех. Оформление учебных и научных работ в системе LATEX. — М.: Либроком, 2009.
33. Балдин Е. Компьютерная типография LaTeХ. — СПб.: БХВ-Петербург, 2008.
34. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация: пер. с англ. — М.: Мир, 1985.
35. Прокис Дж. Цифровая связь: пер. с англ. / под ред. Д. Д. Кловского. — М.: Радио и связь, 2000.
36. Скляр Б. Цифровая связь. Теоретические основы и практическое применение: пер. с англ. — М.: Издательский дом "Вильямс", 2003.
37. Солонина А. И., Улахович Д. А., Яковлев Л. А. Алгоритмы и процессоры обработки сигналов. — СПб.: БХВ-Петербург, 2001.
38. Солонина А. И., Улахович Д. А., Яковлев Л. А. Цифровые процессоры обработки сигналов фирмы Motorola. — СПб.: БХВ-Петербург, 2000.
39. Вальпа О. Д. Разработка устройств на основе цифровых сигнальных процессоров фирмы Analog Devices с использованием Visual DSP++. — М.: Горячая линия — Телеком, 2007.
40. Сперанский В. С. Сигнальные микропроцессоры и их применение в системах телекоммуникаций и электроники. — М.: Горячая линия — Телеком, 2008.
41. Малла С. Вэйвлеты в обработке сигналов: пер. с англ. — М.: Мир, 2005.
42. Штарк Г.-Г. Применение вейвлетов для ЦОС: пер. с англ. — М.: Техносфера, 2007.
43. Хайкин С. Нейронные сети. Полный курс. — М.: Вильямс, 2006.
44. Галушкин А. И. Нейронные сети. Основы теории. — М.: Горячая линия — Телеком, 2010.

# Предметный указатель

## A

ADC 140  
Additive white Gaussian noise channel 194  
Aerospace Blockset 721  
Aerospace Toolbox 717  
Aliasing 141, 145, 625  
All-pass filter 95, 215  
Amplitude modulation (AM) 503  
◊ with suppressed carrier 514  
Amplitude shift keying 547  
AM-SC 514  
Analog 138  
Analog-to-Digital Converter 140  
Analysis bank 638  
Analysis Parameters 341  
Angle modulation 524  
AR model 322  
Arithmetic 485  
ARMA model 323  
ASK 547  
Autocorrelation 327  
Autoregressive model 322  
Autoregressive Moving Average 323  
Averaged modified periodogram method 321, 357  
AWGN channel 194

## B

Band-pass filter 110  
Band-stop filter 110  
Bartlett 321  
Basis 54  
Beat 514  
Bessel filter 120  
Bilinear transformation 372  
Binary 541  
Bioinformatics Toolbox 718  
Bit-reversed order 299

Blockset 720  
Burg 333  
Butterfly 296  
Butterworth filter 111

## C

Canonic form 239  
Carrier 502  
Cascaded Integrator — Comb filter 631  
Cauer filter 118  
Cell array 670  
Chebyshev filter:  
◊ type I 113  
◊ type II 115  
CIC 631  
Comb filter 631  
Communications Blockset 721  
Communications Toolbox 717  
Complete orthonormal system 55  
Constellation 548  
Constrained least square 418, 420  
Continuous phase frequency shift keying 543  
Control System Toolbox 716  
Convergence 599, 600  
Convolution matrix 259  
Correlation function 40, 66  
Covariance 327  
◊ function 67  
◊ modified 328  
CPFSK 543  
Cross-correlation function 42  
Cumulative distribution function 61  
Curve Fitting Toolbox 716

## D

DAC 140  
Daniell 321

DAQFcGen 204  
 DAQScope 204  
 Data Acquisition Toolbox 203, 717  
 Database Toolbox 719  
 Datafeed Toolbox 718  
 Decimation 624  
 ◇ in frequency 300  
 ◇ in time 295  
 Decision-directed mode 609  
 Deconvolution 245  
 Demodulation 502  
 Desired signal 592  
 DFT 289  
 ◇ filter bank 639  
 DIF 300  
 Differential phase shift keying 548  
 Digital 139  
 Digital-to-Analog Converter (DAC) 140  
 Direct form I 238  
 Direct form II 239  
 Direct transposed form I 242  
 Direct transposed form II 240, 247  
 Discrete Fourier Transform 289  
 Discrete series 138  
 Discrete-time 139  
 DIT 295  
 Dot product 53  
 Double 452  
 Downsampling 625  
 DPSK 548  
 DSP Blockset 435  
 Duty cycle 16

**E**

Econometrics Toolbox 718  
 EDA Simulator Link 723  
 Eigenvectors 337  
 Elliptic filter 118  
 Embedded IDE Link 723  
 Ensemble averaging 62  
 Envelope 177  
 Equalization 609  
 Equalizer 609  
 Equiripple filter 379  
 Ergodic 71  
 Error function 65  
 EV 337  
 Even symmetry 236  
 Excess error 600  
 Exponent 450  
 Exponential forgetting 606  
 Extended 452  
 Eye diagram 557

**F**

Fast Fourier Transform 295  
 FDA (расширение файлов) 437  
 FDATool 433  
 ◇ вкладка:  
     ▫ Design Filter 435  
     ▫ Frequency Transformations 441  
     ▫ Import Filter 439  
     ▫ Set Quantization Parameters 494  
 FFT 295  
 Field 669  
 Figure window 661  
 Filter bank 638  
 Filter Design HDL Coder 717  
 Filter Design Toolbox 717  
 Filter order 234  
 Filter Visualization Tool (FVTool) 285, 486  
 ◇ для аддитивных фильтров 616  
 FilterBuilder 443, 444, 445  
 Financial Derivatives Toolbox 718  
 Financial Toolbox 718  
 Finite impulse response 236  
 FIR filter 236  
 Fixed point 446  
 Fixed-Income Toolbox 718  
 Fixed-Point Toolbox 717  
 Flat top window 343  
 Float 452  
 Floating point 446  
 Fluctuation 62  
 FM 525  
 Fourier transform 22  
 Fractionally-spaced equalizer 615  
 Frequency deviation 527  
 Frequency modulation (FM) 525  
 Frequency shift keying 542  
 FSK 542  
 Function 671  
 ◇ nested 674  
 ◇ subfunction 674  
 Fuzzy Logic Toolbox 716

**G**

Gain 93, 213, 264  
 Gauges Blockset 721  
 Gaussian distribution 64  
 Global Optimization Toolbox 716  
 Granular limit cycle 471  
 Group delay 89

**H**

Heterodyne 154  
 Heterodynning 155

High-pass filter 109

Hoertzel 307

## I

IIR filter 238

Image Acquisition Toolbox 717

Image Processing Toolbox 717

Impulse invariance 373

Infinite impulse response 238

Initial conditions 247

Inner product 53

Instantaneous frequency 525

Instrument Control Toolbox 718

Interpolation 624

Intersymbol interference 556

Inverse Chebyshev filter 117

ISI 556

## K

Kernel 56

Keying 542

## L

Leakage 314, 614

◊ factor 340

Levinson—Durbin algorithm 328

Limit cycle 471

Linear prediction 323

LMS-алгоритм 599

◊ нормированный 600

Logging 487

Long double 452

Low-pass filter 109

## M

MA model 322

Mainlobe width 341

Mantissa 450

Mapping Toolbox 717

MATLAB 655

◊ Compiler 719

◊ Report Generator 719

◊ графика 689

◊ двумерная 689

◊ копирование графиков 702

◊ настройка оформления 696

◊ панель инструментов графического окна

690

◊ трехмерная 692

◊ двоеточие 665, 666

◊ запятая 659, 671

◊ знак процента 671

◊ интерактивный режим 656

◊ интерфейс главного окна 662

◊ квадратные скобки 659

◊ команда:

◊ doc 662

◊ help 662, 673

◊ helpdesk 662

◊ load 681

◊ save 680

◊ командное окно 656

◊ компоненты 715

◊ круглые скобки 665

◊ меню Help 662

◊ окно:

◊ Command History 663

◊ Command Window 656

◊ Current Folder 663

◊ Workspace 663

◊ операторы сравнения 675

◊ переменная 657

◊ ans 657

◊ глобальная 674

◊ локальная 674

◊ подавление вывода результата 660

◊ поэлементные операции 658

◊ программирование 670

◊ ввод и вывод данных 679

◊ запись нескольких операторов в строке  
671

◊ комментарии 671

◊ логические условия 675

◊ оператор выбора 676

◊ оптимизация программ 682

◊ отладка 681

◊ программы 671

◊ профилирование 685

◊ разрыв длинных строк 671

◊ редактор/отладчик M-файлов 672

◊ сценарии 671

◊ условный оператор 675

◊ функции 671, 672

◊ циклы 677

◊ путь поиска файлов 674

◊ редактор переменных 663

◊ системные требования 655

◊ список Current Folder 664

◊ справочная система 661, 662

◊ тильда (~) 673

◊ типы данных:

◊ массивы 664

◊ массивы ячеек 670

◊ многомерные массивы 667

- нумерация элементов 665
- строки 668
- структуры 669
- ◊ точка с запятой 659, 660, 671
- ◊ три точки 671
- ◊ установка 656
- ◊ фигурные скобки 670
- MATLAB Builder 719
- MAT-файлы 202, 680
- MBSA 287
- McBride L. E. 407
- McClellan J. H. 379
- Mean value 62
- Metric 50
- Minimax 379
- Minimum shift keying 546
- Misalignment 600
- M-Lint 687
- Model Predictive Control Toolbox 716
- Model-Based Calibration Toolbox 716
- Model-Based Spectrum Analysis (MBSA) 287
- Modified covariance 328
- Modified periodogram 320
- Modulated signal 502
- Modulating signal 502
- Modulation 502
- Modulation index 526
- Moving Average 323
- MSK 546
- MSQ 12
- Multiband filter 408
- MULTiple SIgnal Classification 335
- MUSIC 335

**N**

- Nested function 674
- Neural Network Toolbox 716
- Neural networks 592
- Noise subspace 336
- Nonparametric spectrum analysis 287
- Nonrecursive filter 235
- Norm 52
- Normal distribution 64
- Number of Points 341
- Nyquist frequency 140
- Nyquist theorem 150

**O**

- Odd symmetry 236
- OPC Toolbox 718
- Optimization Toolbox 716
- Order 234

- Orthogonal 53
- Overflow limit cycle 472
- Overlap-add 269, 312
- Overlap-save 312

**P**

- Parametric spectrum analysis 287
- Parks T. W. 379
- Partial Differential Equation Toolbox 716
- Passband 111
- Perfect reconstruction 646
- Periodogram 320
- Phase 632
  - ◊ delay 88
  - ◊ filter 215
  - ◊ modulation 524
  - ◊ shift keying 547
- PM 524
- Pole 93, 213
- Polyphase 632
- Postwindowing 327
- Power spectral density 74
- PPM 568
- PR 646
- Prediction error 323
- Prewindowing 327
- Probability density function 61
- PSD 74
- Pseudospectrum 335
- PSK 547
- Pulse Amplitude Modulation 580
- Pulse position modulation 568
- Pulse train 179
- Pulse width modulation (PWM) 558
- p-норма 376
  - ◊ ошибки 432
- p-плоскость 258

**Q**

- QAM 539, 548
- Quadrature 177
  - ◊ amplitude modulation (QAM) 539, 548
- Quantization 139
- Quotient 245

**R**

- RADIX 302
- Raised cosine filter 390
  - ◊ square root 393
- Rayleigh 82
- Real-Time Windows Target 722

- Real-Time Workshop 722  
 Recursive filter 237  
 Recursive Least Squares 602  
 Reference signal 592  
 Reflection coefficients 332  
 Relative sidelobe attenuation 340  
 Remainder 245  
 Representation 53  
 Resampling 624, 628  
 RF Blockset 721  
 RF Toolbox 717  
 RLS-алгоритм 602  
 RMS 12  
 Robust Control Toolbox 716  
 Rolloff factor 390
- S**
- Sample time 138  
 Samples 138  
 Sampling 139  
 ◊ frequency 138, 168  
 ◊ theorem 150  
 Saturation 474  
 Scalar product 53  
 Scatter plot 553  
 Scrambling 611  
 Script 671  
 Search path 674  
 Second-order sections 263  
 Shannon theorem 150  
 Shaping filter 555  
 Sign LMS 623  
 Signal Processing Blockset 721  
 Signal Processing Tool 724  
 Signal Processing Toolbox 717  
 Signal space 50  
 Signal space diagram 549  
 Signal structure 549  
 Signal subspace 336  
 Signal-to-noise ratio 454  
 SimBiology 718  
 SimDriveline 721  
 SimElectronics 721  
 SimEvents 720  
 SimHydraulics 721  
 SimMechanics 720  
 SimPowerSystems 721  
 Simscape 720  
 Simulink 272, 435, 444, 719  
 ◊ 3D Animation 721  
 ◊ Control Design 721  
 ◊ Design Optimization 721  
 ◊ Design Verifier 723
- ◊ Fixed Point 720  
 ◊ HDL Coder 722  
 ◊ PLC Coder 722  
 ◊ Report Generator 720  
 ◊ Verification and Validation 722  
 ◊ ядро 719  
 Single 452  
 Single side band 516  
 SNR 454  
 Spectral shaping 555  
 Spectrogram 353  
 Spectrum leakage 314  
 Spline Toolbox 716  
 Spreadsheet Link EX 719  
 SPTool 724  
 ◊ окно:  
     ◊ Apply Filter 728  
     ◊ Export from SPTool 730  
     ◊ Import to SPTool 725  
     ◊ Signal Browser 726, 729  
     ◊ Spectrum Viewer 727  
 Square root raised cosine filter 393, 555  
 SSB 516  
 Standard deviation 63  
 State space 98  
 Stateflow 720  
 ◊ Coder 722  
 Statistics Toolbox 716  
 Steiglitz K. 407  
 Stopband 111  
 Structure 669  
 Subband processing 639  
 Subcarrier 522  
 Subfunction 674  
 Supervised learning 592  
 Symbol 541  
 Symbol rate 541  
 Symbolic Math Toolbox 716  
 Synthesis bank 638  
 System Identification Toolbox 716  
 SystemTest 718
- T**
- Target Support Package 722  
 TeX 700  
 Time averaging 71  
 Time-frequency analysis 58  
 Toeplitz matrix 167  
 Training mode 609  
 Transfer function 92, 212  
 Transition band 111  
 Transversal filter 235

**U**

Undersampling 155  
 Uniform distribution 63  
 Unsupervised learning 592  
 Upsampling 626

**V**

Variable Editor 663  
 Variance 62  
 Vehicle Network Toolbox 718  
 Verilog 444  
 VHDL 444  
 Video and Image Processing Blockset 721

**W**

Walker 333, 383  
 Wavelet Toolbox 717  
 Wavelet transform 58  
 WAV-файлы 194  
 Weighting functions 318  
 Welch 321  
 ◇ method 321

White Gaussian noise 193  
 White noise 78  
 Wiener filter 596  
 Wiener—Hopf equation 596  
 Window 318  
 Window Design & Analysis Tool 351  
 Window Visualization Tool 340

**X**

xPC Target 722

**Y**

Yule 333, 383

**Z**

Zero 93, 213  
 Z-transform 159  
 Z-плоскость 258  
 Z-преобразование 159  
 ◇ обратное 164  
 ◇ свойства 162, 163  
 ◇ связь с другими преобразованиями 161

**A**

Абсолютная интегрируемость 23  
 Автокорреляционный метод 364  
 Авторегрессионная модель 322  
 Адаптивные системы 592  
 Адаптивные фильтры 558  
 ◇ применение 607  
 Аддитивный белый гауссов шум 194  
 Алгоритм:  
 ◇ адаптации
 

- LMS 593
- RLS 593
- для решетчатых фильтров 613
- основанный на методе аффинных проекций 613
- работающий в частотной области 613

 ◇ Герцеля 307  
 ◇ дискретной фильтрации 209  
 ◇ Левинсона — Дарбина 328, 362  
 ◇ Ремеза 379  
 АМн 547  
 Амплитуда 9, 503  
 ◇ комплексная 39, 46

Амплитудная манипуляция 547, 551  
 Амплитудная модуляция (AM) 503  
 ◇ демодуляция 512  
 ◇ КПД 511  
 ◇ однотональная 505
 

- векторная диаграмма 508
- спектр 506
- ширина спектра 507

 ◇ перемодуляция 506  
 ◇ с подавленной несущей 514  
 ◇ спектр 509
 

- ширина 510

 ◇ средняя мощность 511  
 Амплитудная огибающая 46, 48, 504  
 Амплитудно-частотная характеристика  
 См. АЧХ  
 Амплитудный спектр 15, 24  
 АМ-ПН 514  
 Анализ:  
 ◇ корреляционный 40  
 ◇ сигналов 7  
 Аналитический сигнал 47, 517, 536  
 ◇ спектр 48

Аналого-цифровой преобразователь (АЦП)  
 140, 455  
 Анонимные функции 229  
 Ансамбль реализаций 58  
 Аппроксимация АЧХ 110  
 АЧХ 88, 101, 110, 191  
 ◇ крутизна ската 118  
 ◇ максимально плоская 113, 115, 117  
 ◇ неравномерность 132  
 ◇ связь с положением нулей и полюсов 93, 213

**Б**

Бабочка 296, 301  
 База сигнала 26  
 Базис 54  
 ◇ Котельникова 150  
 ◇ ортонормальный 54  
 Банк фильтров 638  
 ◇ анализа 638, 639  
 ◇ на основе ДПФ 639  
 ◇ синтеза 638, 643  
 ◇ точного восстановления 646  
 Белый шум 78  
 ◇ дискретный 167  
 ◇ преобразование в линейной системе 91  
 Биения 514, 541  
 Билинейное z-преобразование 372, 400  
 Бит-реверсная перестановка 299  
 БИХ-фильтры 238  
 Блоковая обработка 311  
 Боковые полосы 510  
 Боковые частоты 507  
 Быстрое преобразование Фурье (БПФ) 294, 295, 641  
 ◇ квантованное 497  
 ◇ обратное 302  
 ◇ основание 302  
 ◇ прореживание:  
     ▫ по времени 295  
     ▫ по частоте 300  
 ◇ число операций 298, 303

**В**

Вейвлет-преобразование 58  
 Вектор 51, 167  
 ◇ коэффициентов усиления 605  
 ◇ состояния 98, 220, 240, 241  
 Вектор-строка 657  
 Верхняя боковая полоса 516  
 Весовая функция 318, 339, 384

Вещественная система 87  
 Взаимная корреляционная функция (ВКФ) 42  
 Взаимный спектр 44  
 ◇ выходного и входного сигналов 89  
 Визуализатор окон 340  
 Вложенная функция 674  
 Восстановление:  
 ◇ непрерывного сигнала 144, 150  
 ◇ радиосигнала 153  
 Временная область 24  
 Времяимпульсная модуляция (ВИМ) 568  
 Вставка нулей 626  
 Выбор порядка фильтра 398  
 ◇ для метода Ремеза 428  
 Выравнивание частотной характеристики 609  
 Вычеты 96, 218

**Г**

Гармоники 13  
 Генерация:  
 ◇ импульсов 172  
 ◇ периодических сигналов 182  
 ◇ последовательности импульсов 179  
 ◇ программа daqfengen 204  
 ◇ сигнала с меняющейся частотой 186  
 Гетеродин 154  
 Гетеродинирование 155  
 Гистограмма 191  
 Глазковая диаграмма 557, 587  
 Глубина модуляции 505  
 Годограф 104  
 Графическое окно 661  
 Гребенчатый фильтр 631  
 Групповая задержка 89, 120, 384, 432  
 ◇ для симметричных фильтров 237  
 ◇ расчет для аналоговых систем 134

**Д**

Двоичная последовательность 541  
 Девиация частоты 527  
 Действующее значение 12  
 Дельта-функция 9, 86  
 ◇ спектр 38  
 ◇ фильтрующее свойство 9, 161  
 Демодуляция 502  
 ◇ АМ 512  
     ▫ с подавленной несущей 515  
 ◇ КАМ 539  
 ◇ квадратурной модуляции 553  
 ◇ однополосного сигнала 519  
 ◇ УМ 536

◊ ФМ 536, 537

◊ ЧМ 536, 538

◊ ЧМн 544

Демонстрационная программа:

◊ daqfcn 204

◊ daqscope 204

◊ scattereyedemo 586, 588

Демультиплексор 638, 641

Диаграмма рассеяния 553, 585

Динамический диапазон 449

Дискретизация 139

◊ дискретного сигнала 625

◊ квадратурная 154

Дискретная:

◊ синусоида 160

◊ экспоненциальная функция 160

Дискретно-аналоговые устройства 139

Дискретное преобразование Фурье (ДПФ)

287, 289, 639, 641, 644

Дискретный единичный скачок 160

Дисперсия 62, 67, 74

◊ выходного сигнала линейной системы 90

Дифференциальное уравнение линейной системы 92

Добротность 127

Дополнительный код 447, 448

## E

Единичная импульсная функция 159

Единичный дискретный скачок 160

## 3

Задержка:

◊ групповая 89, 255

◊ фазовая 88

Записи сигналов 202

Запятая:

◊ плавающая 450

◊ фиксированная 448

Затухание 132

Звук:

◊ воспроизведение 199

◊ запись 201

## И

Идентификация 607, 617

Избыточная ошибка 600

Импульс:

◊ гауссов 30

◊ конечная последовательность 179

◊ прямоугольный 25, 171, 172

◊ задержанный 25

◊ трапециевидный 174

◊ треугольный 173

◊ несимметричный 27, 171, 172

◊ симметричный 28

◊ экспоненциальный:

◊ двусторонний 30

◊ односторонний 29, 171, 172

Импульсная характеристика 373

◊ аналоговой системы 86

◊ расчет 96

◊ дискретной системы 210, 249

◊ расчет 218

Индекс угловой модуляции 526

Интеграл вероятности 64

Интегратор 631

Интервал:

◊ дискретизации 138

◊ корреляции 77

Интерполяция 181, 554, 624, 626

◊ полифазная реализация 634

Искажения мультипликативные 147

## K

Квадратурная дискретизация 154

Квадратурная амплитудная модуляция (КАМ) 503, 539, 548

◊ демодуляция 539, 553

◊ помехоустойчивость 551

◊ спектр 539

◊ мощности 556

◊ формирование 554

◊ цифровая 548

Квадратурная обработка сигнала 537

Квадратурное дополнение 47, 177

Квантование 139, 453

◊ в цифровых фильтрах 456

◊ гармонического сигнала 453

◊ отношение сигнал/шум 454

◊ зоны 455

◊ неравномерное 455, 475

◊ ошибки 139

◊ уровни 453

◊ характеристика 482

◊ шум 453

Квантованные значения 455

КИХ-фильтры 236

Классификация сигналов 8

Ключевое слово:

◊ break 677

◊ case 676

◊ continue 678

Ключевое слово (*прод.*):

- ◊ else 675
  - ◊ elseif 676
  - ◊ end 666, 675—677
  - ◊ for 677
  - ◊ function 672
  - ◊ if 675
  - ◊ otherwise 676
  - ◊ return 674
  - ◊ switch 676
  - ◊ while 677
- Ковариационная функция 67  
 Ковариационный метод 365  
 ◊ модифицированный 365  
 Код Баркера 651  
 Колебания квазигармонические 79  
 Команда *profile* 685  
 Командное окно 656  
 Комбинация линейная 54  
 Комментарии 671  
 Комплексная:
  - ◊ амплитуда 39, 46
  - ◊ огибающая 46, 48
  - ◊ частотная характеристика 88
  - ◊ экспонента:
    - спектр 39
 Комплексный коэффициент передачи 88, 212  
 Композитный стереосигнал 523  
 Константа:
  - ◊ спектр 38
 Конструктор 266  
 Корреляционная матрица 166, 595  
 ◊ белого шума 167  
 ◊ неотрицательная определенность 167  
 ◊ оценка 326  
 Корреляционная функция (КФ) 40, 66, 67  
 ◊ взаимная 42
  - выходного и входного сигналов 89
 ◊ импульсной характеристики 90  
 ◊ интервал корреляции 77  
 ◊ коэффициент корреляции 70  
 ◊ нормированная 70  
 ◊ периодического сигнала 41  
 ◊ связь со спектрами сигналов 44  
 Корреляционный анализ 40  
 Косинусоидальное сглаживание АЧХ 389, 412  
 ◊ коэффициент сглаживания 390  
 Коэффициент:
  - ◊ заполнения 16, 182
  - ◊ корреляции 69
  - ◊ модуляции 505
  - ◊ отражения 332
  - ◊ передачи 88

- ◊ расстройки 600
- ◊ сглаживания 390
- ◊ усиления 93, 213
- ◊ утечки 340, 614

**Л**

- Лепестки спектра 17, 25  
 Линейная комбинация 54  
 Линейная независимость 54  
 Линейное предсказание 323, 607, 619  
 Линейное пространство 51  
 Линейность 85, 206  
 Линия задержки 234  
 Логическая маска 171  
 Ложные частоты 141, 145, 625

**М**

- Манипуляция 542  
 ◊ амплитудная 547, 551  
 ◊ фазовая 547, 551  
 ◊ частотная 542  
 Мантисса 450  
 ◊ денормализованная 451  
 ◊ нормализация 450  
 Маска логическая 171  
 Массив ячеек 670  
 Массивы 664  
 Масштабирование коэффициентов фильтров 460  
 Математическое ожидание 62  
 ◊ функций от случайной величины 62  
 Матрица 167  
 ◊ единичная 167  
 ◊ корреляционная 166
  - белого шума 167
 ◊ разреженная 712  
 ◊ собственные числа 167  
 ◊ Теллица 167, 595, 596  
 Матричное деление:
  - ◊ оператор \ 598
 Мгновенная частота 525  
 Меандр 17  
 Межсимвольная интерференция (МСИ) 556  
 Метод:
  - ◊ EV 337
  - ◊ MUSIC 335
  - ◊ автокорреляционный 327, 333
  - ◊ Берга 333
  - ◊ инвариантной импульсной характеристики 373, 401
  - ◊ ковариационный 327, 333
    - модифицированный 328, 333, 365

- ◊ наименьшего квадрата (МНК) 593, 599
- ◊ наискорейшего спуска 598
- ◊ низкочастотного эквивалента 50
- ◊ объектов 267
- ◊ постзвешивания 327
- ◊ предзвешивания 327, 364
- ◊ Прони 383, 404
- ◊ усреднения модифицированных периодограмм 321, 357
- ◊ Уэлча 321
- ◊ Штейглица — МакБрайда 383, 407
- ◊ Юла — Уолкера 333, 383
- Метрика 50, 53, 165
- ◊ евклидова 51
- Метрическое пространство 50
- Минимаксная оптимизация 379
- Минимизация р-нормы ошибки 432
- Многополосные фильтры 408
- Моделирование сигналов 168
- Модель математическая 59
- Модельный спектральный анализ 287
- Модифицированный ковариационный метод 333, 365
- Модулированный сигнал 502
- Модулирующий сигнал 502
- Модуляция 502
  - ◊ амплитудная 154, 503
    - однотональная 505
    - перемодуляция 506
  - ◊ квадратурная 503, 539, 548
  - ◊ однополосная 516
  - ◊ полярная 520
  - ◊ угловая 154, 503, 524
  - ◊ фазовая 524
  - ◊ цифровая 541
  - ◊ частотная 525
  - ◊ широтно-импульсная 558
- Мощность 11
  - ◊ мгновенная 11
  - ◊ средняя 11, 41
    - случайного процесса 74
  - ◊ удельная 12
- Мультиплексор 635, 645
- Мультиплектические искажения 147
- МЧМн 546

## H

- Насыщение 474
- Научная нотация 450
- Начальная фаза 9, 48
- Независимость:
  - ◊ линейная 54
  - ◊ статистическая 66

- Нейронные сети 592
- Некоррелированность 65, 167
- Нелинейность 85
- Непараметрический спектральный анализ 287
- Неравенство:
  - ◊ Бесселя 55
  - ◊ Буняковского — Коши — Шварца 53
  - ◊ треугольника 52
- Нестационарность 86
- Несущая 502
- Несущее колебание 502
- Нижняя боковая полоса 516
- Низкочастотный эквивалент 50
- Норма 52, 53, 165
  - ◊ евклидова 52
- Нормированное линейное пространство 52
- Нули функции передачи 93, 213, 258

## O

- Образцовый сигнал 592
- Обратный код 447
- Обучение 592
- Объект 266, 477
  - ◊ свойства 266
- Огибающая 89, 177, 504
  - ◊ амплитудная 46, 48
  - ◊ комплексная 46, 48
- Однополосная модуляция 516
  - ◊ демодуляция 519
- Окно 318, 339, 384
  - ◊ Бартлетта 342, 388
  - ◊ Бартлетта — Ханна 345
  - ◊ Блэкмена 343
  - ◊ Блэкмена — Харриса 343
  - ◊ Бомена 345, 388
  - ◊ гауссово 346, 349
  - ◊ Кайзера 347, 388, 410, 652
  - ◊ косинусное 342
  - ◊ Наттолла 343
  - ◊ Парзена 349, 388
  - ◊ прямоугольное 339
  - ◊ сводка параметров 349
  - ◊ с плоской вершиной 343, 388
  - ◊ треугольное 341, 388
  - ◊ Ханна 342
  - ◊ Хэмминга 343, 386, 388, 409, 410
  - ◊ Чебышева 348, 388
- Округление 453
  - ◊ коэффициентов фильтров 457
  - ◊ промежуточных результатов 465
- Оператор:
  - ◊ дифференцирования 35
  - ◊ интегрирования 35

Операции поэлементные 168  
 Опорное колебание 512  
 Опорный сигнал 592  
 Ортогональная проекция 55  
 Ортогональность 53  
 Остаток 245  
 Отношение сигнал/шум 454  
 Отсчеты 138, 195  
 ◇ комплексные 154  
 ◇ спектральные 293  
 Ошибка:  
 ◇ предсказания 323  
 ◇ квантования 139

## П

Пакет расширения:  
 ◇ Data Acquisition 203  
 ◇ Statistics 190  
 Параллельные вычисления 240  
 Параметрические системы 86  
 Параметрический спектральный анализ 287  
 Передискретизация 624, 628  
 Переобласти:  
 ◇ с накоплением 312  
 ◇ с суммированием 269, 312  
 Перемодуляция 506, 523, 564  
 Переполнение 462  
 Переходная зона 111  
 Переходная характеристика 87  
 ◇ дискретной системы 250  
 Период дискретизации 138  
 Периодическое продолжение сигнала 13  
 Периодограмма 320  
 ◇ Бартлетта 321  
 ◇ Даньелла 321  
 ◇ модифицированная 320  
 Пилот-сигнал 516, 523  
 Плавающая запятая 450  
 Плотность вероятности 455  
 ◇ выходного сигнала линейной системы 91  
 ◇ двумерная 66  
 ◇ многомерная 66  
 ◇ одномерная 61  
 Повышение частоты дискретизации 626  
 Подавление шума 608, 620  
 Поднесущая 522  
 Поле 51  
 ◇ структуры 669  
 Полином Чебышева 117  
 Полиномы, деление 245  
 Полифазное представление 632  
 Полифазные структуры 624

Полная ортонормальная система 55  
 Полная фаза 46, 48, 525  
 Полоса:  
 ◇ задерживания 110, 111, 132  
 ◇ пропускания 110, 111, 132  
 Полосовой фильтр (ПФ) 110  
 ◇ второго порядка 226  
 ◇ идеальный 153  
 Получение данных из внешних источников 194  
 Полосы функции передачи 93, 96, 97, 213, 218, 258  
 ◇ кратные 96  
 Полярная модуляция 520  
 Понижение частоты дискретизации 625  
 Порядок 450  
 ◇ линейной цепи 92  
 ◇ фильтра 234, 237  
 Последовательность импульсов:  
 ◇ меандр 17  
 ◇ пилообразных 20, 205  
 ◇ прямоугольных 16, 182, 205  
 ◇ треугольных 20, 183, 205  
 Постзвешивание 327, 364  
 Постоянная составляющая 72  
 Поэлементные операции 168  
 Предзвешивание 327, 364  
 Предельный цикл 471  
 Представление:  
 ◇ приближенное 55  
 ◇ сигнала:  
 ◇ ◇ дискретное 53  
 ◇ ◇ интегральное 56  
 Преобразование Гильберта 47, 57, 79, 517  
 ◇ дискретное 276, 417, 423  
 Преобразование Лапласа 92, 98  
 ◇ связь с z-преобразованием 161  
 Преобразование Фурье 22, 57  
 ◇ быстрое 294  
 ◇ дискретное 287, 289  
 ◇ ◇ быстрые алгоритмы 294  
 ◇ ◇ восстановление непрерывного сигнала 291  
 ◇ ◇ матрица преобразования 292  
 ◇ ◇ свойства 289  
 ◇ ◇ связь со спектральной функцией 293  
 ◇ ◇ спектральное разрешение 294  
 ◇ ◇ эквивалентный фильтр 305  
 ◇ дуальность 31, 144  
 ◇ задержка 33  
 ◇ обратное 23  
 ◇ примеры расчета 24  
 ◇ прямое 23

- ◊ свойства 33
  - дифференцирование 34
  - изменение масштаба оси времени 34
  - интегрирование 35
  - линейность 33
  - произведение 36
  - свертка 35
  - умножение на гармоническую функцию 36
- ◊ связь с z-преобразованием 161
- ◊ связь с рядом Фурье 37
- Преобразование частоты на АЦП 155
- Причинность 87, 247
- Программа:
  - ◊ FDATool 244
    - квантование фильтров 494
  - ◊ SPTool 724
    - импорт данных 725
    - просмотр графика сигнала 726
    - просмотр характеристик фильтра 728
    - расчет фильтров 728
    - сохранение и загрузка сеанса 729
    - спектральный анализ 727
    - фильтрация сигнала 728
    - экспорт данных 730
- Проектирование цифровых фильтров 371
- Проекция ортогональная 55
- Прони 383
- Прореживание 295, 624, 625
  - ◊ по времени 295
  - ◊ по частоте 300
  - ◊ полифазная реализация 635
- Пространственная диаграмма 549
- Пространство:
  - ◊ сигналов 50
    - дискретных 165
    - линейное 51
    - метрическое 50
    - нормированное 52
    - со скалярным произведением 53
  - ◊ со скалярным произведением 53
  - ◊ состояний 220, 240, 241
- Простые:
  - ◊ дроби 95
  - ◊ сигналы 26
- Протоколирование работы 487
- Прототип 640
  - ◊ аналоговый 371
  - ◊ Прямой код 447, 450
  - Прямые методы синтеза цифровых фильтров 376
  - Псевдоспектр 335, 337
  - Пульсации АЧХ 113, 117, 118
    - ◊ равномерные 379

**P**

- Равенство Парсеваля 45
  - ◊ дискретное 291
- Радиоимпульс 175
  - ◊ гауссов 176
- Размер шага 598
- Разностное уравнение 209
- Распределение вероятности:
  - ◊ гауссово 64
  - ◊ нормальное 64
  - ◊ Пуассона 60
  - ◊ равномерное 63, 453
  - ◊ Рэлея 82
    - генерация случайных чисел 191
    - функция распределения 191
  - ◊ Рэлея — Райса 84
- Растекание спектра 314
- Реализация случайного процесса 58
- Режекторный фильтр второго порядка 230
- Режим:
  - ◊ дуплексный 610
  - ◊ обучения 609
  - ◊ оценивания 609
- Резонатор второго порядка 225
- Рекурсивный метод наименьших квадратов 602
- Риш Гаспар 383
- РНК 593
- Ряд:
  - ◊ Котельникова 292
  - ◊ Фурье 12, 528
    - в точках разрыва 19
    - вещественный 14
    - вычисление мощности сигнала 45
    - комплексный 14, 15
    - примеры разложения сигналов 16
    - связь с преобразованием Фурье 37
    - синусно-косинусный 13
    - частичные суммы 18
- C**
- Свертка 86, 89, 90
  - ◊ дискретная 244
    - линейная 162, 210
    - матрица свертки 259
    - представление в виде скалярного произведения 259
  - ◊ круговая 291, 308
    - периодических последовательностей 163
  - ◊ линейная 163, 308
  - ◊ обращение 245

- Свободные колебания 219
- Свойство объекта 266
- Секции второго порядка 263
- Секционная обработка 311
- Сигнал 8
  - ◊ аналитический 47, 154, 278, 517, 536
  - ◊ аналоговый 138
  - ◊ база 26
  - ◊ гармонический 9, 204
    - дискретный 314
    - со случайной начальной фазой 59
    - спектр 39
  - ◊ детерминированный 8
  - ◊ дискретизированный 142, 149
  - ◊ дискретный 139
    - случайный 165
    - спектр 142
  - ◊ комплексный 11
  - ◊ конечной длительности 9
  - ◊ многоканальный 167, 170
  - ◊ модулированный 502
  - ◊ модулирующий 502
  - ◊ образцовый 592
  - ◊ опорный 592
  - ◊ периодический 8
    - спектр 39
  - ◊ периодическое продолжение 13
  - ◊ пилообразный 20
  - ◊ постоянный во времени 38
    - спектр 38
  - ◊ простой 26
  - ◊ с большой базой 27
  - ◊ с меняющейся частотой 205
  - ◊ с ограниченной полосой частот 175
  - ◊ с ограниченной энергией 8
  - ◊ сложный 27
  - ◊ случайный 8, 58, 205
  - ◊ сопряженный 47
  - ◊ физический 138
  - ◊ финитный 9
  - ◊ цифровой 139, 541
  - ◊ частота повторения 9
  - ◊ широкополосный 27

Сигнал/шум 454

Сигнальная конструкция 549

Сигнальное подпространство 336

Символ 541

Символьная скорость 541

Синтез цифровых фильтров 371
 
  - ◊ билинейное z-преобразование 372
  - ◊ классификация методов 371
  - ◊ метод инвариантной импульсной характеристики 373

  - ◊ метод Ремеза 422
  - ◊ прямые методы 376
    - оптимальные 376
    - субоптимальные 376, 383
  - ◊ с использованием окон 384, 408
  - ◊ функции MATLAB 393

Синхронное детектирование 512, 515

Системная функция 212

Системы:

  - ◊ аналоговые 85
  - ◊ дискретные:
    - способы описания 210
    - устойчивость 219
    - физическая реализуемость 212
  - ◊ классификация 85
  - ◊ линейные 85
    - вещественные 87
    - дифференциальное уравнение (ДУ) 92
    - комплексный коэффициент передачи 92
    - нули и полюсы 93
    - операторный коэффициент передачи 92
    - полюсы и вычеты 95
    - преобразование детерминированных сигналов 86
    - преобразование случайных процессов 90
    - пространство состояний 98
    - способы описания 91
    - условие устойчивости 97
    - устойчивость 96
    - функция передачи 92
    - характеристики 86
  - ◊ нелинейные 85
  - ◊ с переменными параметрами 86
  - ◊ с постоянными параметрами 86
  - ◊ физическая реализуемость 87

Скаляр 51

Скалярное произведение 53, 54, 165

Скважность 16, 182, 559

Скорость затухания спектра 20, 21

Скремблирование 611

Слепая адаптация 592

Сложение векторов 51

Сложный сигнал 27

Случайные величины 61
 
  - ◊ генерация 189
  - ◊ двумерные 66
  - ◊ коэффициент корреляции 69
  - ◊ независимость 66, 68
  - ◊ некоррелированность 65, 69
  - ◊ статистическая связь 68
    - линейная 69

Случайный процесс 58

  - ◊ вероятностные характеристики:
    - двумерные 66

- ◊ многомерные 66
  - одномерные 61
- ◊ вещественный 166, 167
- ◊ дискретный 165, 454
  - корреляционная матрица 166
- ◊ дисперсия 62
- ◊ квазидетерминированный 60
- ◊ ковариационная функция 67
- ◊ корреляционная функция 66
- ◊ математическое ожидание 62
- ◊ модели 59
- ◊ плотность вероятности
  - одномерная 61
- ◊ реализация 58
- ◊ сечение одномерное 61
- ◊ спектр мощности 74
- ◊ спектральные характеристики 73
- ◊ среднее квадратическое отклонение 63
- ◊ стационарный 69, 166
  - коэффициент корреляции 70
- ◊ узкополосный 78
  - огибающая 82, 84
  - при наличии детерминированной составляющей 83
  - фаза 83
- ◊ функция распределения вероятности 61
- ◊ центрированный 67
- ◊ экспоненциально коррелированный 352
- ◊ эргодический 71
- Случайный сигнал:
  - ◊ интервал корреляции 77
- Смешенный код 447, 451
- Собственное число 599
- Созвездие 548
- Соотношение неопределенности 27
- Сопряженный сигнал 47
- Спектр 15
  - ◊ амплитудный 24
  - ◊ взаимный 44
  - ◊ дискретизированного сигнала 144
    - влияние формы импульсов 147
    - размерность 144
  - ◊ дискретного сигнала 142, 316
    - периодического 288
    - случайного 319
  - ◊ мгновенный 353
  - ◊ мощности 73
  - ◊ односторонний 48
  - ◊ растекание 314
  - ◊ симметрия 24
  - ◊ скорость затухания 20, 21
  - ◊ фазовый 24
- ◊ ширина эффективная 26, 77
- ◊ энергетический 44
- Спектральная плотность 23
- ◊ мощности (СПМ) 74, 352
- Спектральная функция 23
- Спектральный анализ:
  - ◊ модельный 287
  - ◊ непараметрический 287, 320
    - метод Уэлча 321
    - периодограмма 320
  - ◊ параметрический 287, 322
    - EV 337
    - MUSIC 335
    - авторегрессионные 322
- Спектрограмма 353
- Список Current Folder 674
- Среда для расчета фильтров:
  - ◊ FDATool 433
    - анализ фильтра 436
    - импорт фильтра 439
    - синтез фильтра 435
    - сохранение результатов 437
    - частотное преобразование фильтра 441
  - ◊ FilterBuilder 444
- Среднее квадратическое отклонение 63
- Среднеквадратичное значение 12
- Средняя мощность сигнала 41
- Статистическая независимость 66
- Статистическое усреднение 62
- Стационарность 86, 206
  - ◊ случайного процесса 69, 70
- Стереовещание 520
- Стереосигнал композитный 523
- Строки 668
- Структура 669
- Субдискретизация 155
- Субполосная обработка 639
- Субфункция 674
- Сходимость 599, 600

## Т

- Теорема:
- ◊ Винера — Хинчина 75
    - дискретная 320
  - ◊ дискретизации 150
  - ◊ Котельникова 149, 150
  - ◊ Найквиста 150
  - ◊ Рэлея 45
    - дискретная 291
  - ◊ Шеннона 150
  - Точное восстановление 646
  - Транспонирование 659

**У**

- Угловая модуляция 503, 524  
 ◇ гармоническая 526
  - векторная диаграмма 531
  - девиация частоты 527
  - индекс 526
  - спектр 528
  - ширина спектра 532
 ◇ демодуляция 536  
 Узкополосность 502  
 Умножение вектора на скаляр 52  
 Уравнение:  
 ◇ Винера — Хопфа 596  
 ◇ разностное 209  
 Уровни квантования 453  
 Усечение 453  
 Условие устойчивости 97  
 Условия Дирихле 12  
 Усреднение:  
 ◇ по времени 71  
 ◇ статистическое 62  
 Устойчивость 96  
 ◇ системы второго порядка 224  
 ◇ системы первого порядка 221

**Ф**

- Фаза 632  
 ◇ начальная 48  
 ◇ полная 46, 48, 525  
 Фазовая автоподстройка частоты (ФАПЧ) 514, 538  
 Фазовая задержка 88  
 Фазовая манипуляция 547, 551  
 ◇ дифференциальная 548  
 ◇ относительная 548  
 ◇ фазоразностная 548  
 Фазовая модуляция 524  
 ◇ демодуляция 536, 537  
 Фазовая функция 46  
 Фазовое звено 95, 215  
 Фазовращатель 47  
 Фазовый спектр 15, 24, 44  
 Фазовый фильтр 215  
 Фазочастотная характеристика (ФЧХ) 88, 89, 101  
 ◇ линейная 247  
 Физическая реализуемость 87  
 Фиксированная запятая 448  
 Фильтр:  
 ◇ адаптивный 206, 558, 591
  - применение 607

- антисимметричный 236
- Баттервортса 111, 130, 564
- Бесселя 120, 131
  - групповая задержка 136
- верхних частот (ФВЧ) 109
  - первого порядка 223
- Винера 325, 596
- внутреннее состояние 246
- всепропускающий 95, 215, 222, 432
- выбор порядка 132
- дискретный 206, 245
  - блочный в частотной области 311
  - внутреннее состояние 653
  - преобразование способов описания 260
  - реализация в частотной области 308
  - устойчивость 473
  - фазовая задержка 254
  - фазочастотная характеристика 253
  - форма реализации 238
  - частотная характеристика 251
- дифференцирующий 417, 423
- заграждающий 110
- Золотарева — Кауэра 118
- Кауэра 118
- максимально-фазовый 432
- минимально-фазовый 432
- многополосный 408
- начальное состояние 247
- нерекурсивный 208, 234
  - импульсная характеристика 235
- нижних частот (ФНЧ) 109, 110, 144
  - идеальный 149
  - первого порядка 222
- первого порядка 221
- полосно-задерживающий 110
- полосовой (ПФ) 110
  - второго порядка 226
  - идеальный 153
- порядок 234, 237
- пробка 110
- прототип 110
  - преобразование 123
- режекторный 110
  - второго порядка 230
- рекурсивный 208, 237
  - импульсная характеристика 237
- с бесконечной импульсной характеристикой 238
- с конечной импульсной характеристикой 235
- с косинусоидальным сглаживанием АЧХ 389, 390, 412
  - SQRT-вариант 392, 555

- ◊ с переменными параметрами 206
- ◊ симметричный 236, 247
- ◊ трансверсальный 235
- ◊ фазовый 215
- ◊ цифровой:
  - масштабирование 460
  - предельный цикл 471
- ◊ частотные преобразования 122
- ◊ частичный 634
- ◊ Чебышева:
  - второго рода 115, 130
  - групповая задержка 136
  - первого рода 113, 130
- ◊ эллиптический 118, 131
- Фильтр-прототип 371
- Флуктуации 62, 67, 72, 73
- ФМ 524
- ФМн 547
- Форма реализации:
  - ◊ каноническая 239
  - ◊ каскадная 242
  - ◊ параллельная 243, 262
  - ◊ последовательная 242, 263
  - ◊ прямая 238
  - ◊ транспонированная 240
- Форматы представления чисел 446
- ◊ отрицательных 447
- ◊ с плавающей запятой 450
- ◊ с фиксированной запятой 448
- Формирование спектра 555
- Формирующий фильтр 555
- Функция:
  - ◊ adaptfilt 611
    - свойства объекта 614
  - ◊ addstage 271
  - ◊ all 675
  - ◊ amdmod 571
  - ◊ ammod 571
  - ◊ any 675
  - ◊ arbburg 359
  - ◊ arcov 359
  - ◊ armcov 359
  - ◊ aryule 359
  - ◊ assignin 674
  - ◊ audioplayer 201
  - ◊ audiorecorder 202
  - ◊ autoscale 486
  - ◊ avgpower 370
  - ◊ awgn 194
  - ◊ axis 105, 698
  - ◊ barthannwin 345
  - ◊ bartlett 342
  - ◊ besselap 120
  - ◊ besself 131
  - ◊ bi2de 452
  - ◊ bilinear 400
  - ◊ bin2num 480
  - ◊ bitand 452
  - ◊ bitget 452
  - ◊ bitrevorder 299
  - ◊ blackman 343
  - ◊ blackmanharris 343
  - ◊ block 271
  - ◊ bohmanwin 345
  - ◊ box 698
  - ◊ buffer 642
  - ◊ buttap 113
  - ◊ butter 130, 397
  - ◊ buttord 132, 398
  - ◊ cfirpm 427
  - ◊ cheb1ap 115
  - ◊ cheb1ord 132, 398
  - ◊ cheb2ap 117
  - ◊ cheb2ord 132, 398
  - ◊ chebwin 348
  - ◊ cheby1 130, 397
  - ◊ cheby2 130, 397
  - ◊ chirp 186
  - ◊ clabel 695
  - ◊ coefficients 271
  - ◊ coeffs 271
  - ◊ colormap 188, 355
  - ◊ commsscope 586, 588
  - ◊ contour 694
  - ◊ contour3 696
  - ◊ contourf 696
  - ◊ conv 136, 244
  - ◊ convert 271, 273
  - ◊ convmtx 259
  - ◊ copy 271, 480, 580
  - ◊ copyobj 499
  - ◊ corrmtx 364
  - ◊ cost 271
  - ◊ cpsd 358
  - ◊ cremez 422
  - ◊ decimate 628, 648
  - ◊ deconv 245
  - ◊ demod 562
    - АМ с подавленной несущей 565
    - амплитудная модуляция 563
    - времязимпульсная модуляция 568
    - квадратурная модуляция 567
    - однополосная модуляция 565
    - фазовая модуляция 565
    - частотная модуляция 566
    - широтно-импульсная модуляция 567

- Функция (*прод.*):
- ◊ demodulate 578
  - ◊ denormalize 487
  - ◊ denormalmax 480
  - ◊ denormalmin 480
  - ◊ design 443
  - ◊ dfilt 266, 267, 485
    - свойства объекта 269
    - структура фильтра 268
  - ◊ dftmtx 293
  - ◊ digitrevorder 299
  - ◊ diric 185
  - ◊ disp 271, 480, 499, 580, 679
  - ◊ double 486, 675
  - ◊ downsample 647
  - ◊ dpskdemod 580
  - ◊ dpskmod 580
  - ◊ dspdata 369, 486
  - ◊ ellip 131, 397
  - ◊ ellipap 118
  - ◊ ellipord 132, 398
  - ◊ eps 480, 499
  - ◊ eqtlength 136, 260
  - ◊ erf 65
  - ◊ erfc 65
  - ◊ erfinv 65
  - ◊ errmean 480
  - ◊ errpdf 480
  - ◊ errvar 480
  - ◊ exponentbias 480
  - ◊ exponentlength 481
  - ◊ exponentmax 481
  - ◊ exponentmin 481
  - ◊ eyediagram 587
  - ◊ fcfwrite 271
  - ◊ fdesign 443
  - ◊ fft 338, 498, 642, 668
  - ◊ fftcoeffs 271
  - ◊ fftfilt 339, 728
  - ◊ fftn 668
  - ◊ fftshift 338
  - ◊ figure 700
  - ◊ filter 245, 266, 267, 269, 486, 615
  - ◊ filtfilt 248, 648, 728
  - ◊ filtic 247
  - ◊ findpeaks 381
  - ◊ fipref 487, 489
  - ◊ fir1 408, 642
  - ◊ fir2 409
  - ◊ firband 429
  - ◊ fircls 418
  - ◊ fircls1 420
  - ◊ firgr 429
  - ◊ firlpnorm 432
  - ◊ firls 416, 652
  - ◊ firpm 422
  - ◊ firpmord 428
  - ◊ firrcoefs 412
  - ◊ firtype 271
  - ◊ flattopwin 343
  - ◊ flipr 136
  - ◊ flipud 642
  - ◊ flops 303
  - ◊ fm demod 578
  - ◊ fmmod 577
  - ◊ fprintf 679
  - ◊ fractionlength 481
  - ◊ freqrespest 437, 486, 497
  - ◊ freqs 101
  - ◊ freqspace 253
  - ◊ freqz 251, 266, 270
  - ◊ fskdemod 585
  - ◊ fskmod 585
  - ◊ fvtool 229, 285, 616
  - ◊ gauspuls 176
  - ◊ gausswin 349
  - ◊ gencoswin 343
  - ◊ genqamdemod 582
  - ◊ genqammod 582
  - ◊ get 267
  - ◊ gremez 422
  - ◊ grid 105, 698
  - ◊ grpdelay 255, 270
  - ◊ hamming 343
  - ◊ hann 342
  - ◊ hanning 342
  - ◊ hex2num 481
  - ◊ hilbert 278, 566, 576
  - ◊ hilbiir 415
  - ◊ hist 191
  - ◊ ifft 338, 498, 668
  - ◊ ifftn 668
  - ◊ iirgrpdelay 432
  - ◊ iirlpnorm 432
  - ◊ iirlpnormc 432
  - ◊ iirnotch 274
  - ◊ iirpeak 274
  - ◊ imag 279
  - ◊ impinvar 401
  - ◊ impz 249, 270
  - ◊ impzlength 249, 270, 493
  - ◊ info 267, 271
  - ◊ intdump 589
  - ◊ interp 628, 650

- ◊ invfreqz 402
- ◊ isallpass 271
- ◊ iscascade 271
- ◊ isempty 675
- ◊ isequal 481, 675
- ◊ isfir 272
- ◊ isfixed 481
- ◊ isfloat 481
- ◊ islinphase 272
- ◊ ismaxphase 272
- ◊ isminphase 272
- ◊ isparallel 272
- ◊ isquantizer 481
- ◊ isreal 272
- ◊ isscalar 272
- ◊ issos 272
- ◊ isstable 272
- ◊ kaiser 347
- ◊ kaiserord 410
- ◊ legend 698
- ◊ length 499, 664
- ◊ levinson 359, 362
- ◊ limitcycle 486, 493
- ◊ lloyds 476
- ◊ loglog 692
- ◊ logspace 104
- ◊ lp2bp 127
- ◊ lp2bs 129
- ◊ lp2hp 125
- ◊ lp2lp 123
- ◊ lsb 481
- ◊ marcumq 84
- ◊ max 481
- ◊ maxstep 616
- ◊ mesh 696
- ◊ meshc 696
- ◊ meshgrid 693
- ◊ meshz 696
- ◊ min 481
- ◊ min4termwin 343
- ◊ modem 578
- ◊ modnorm 588
- ◊ modulate 562, 578, 589
  - ◊ АМ с подавленной несущей 565
  - ◊ амплитудная модуляция 563
  - ◊ времязимпульсная модуляция 568
  - ◊ квадратурная модуляция 567
  - ◊ однополосная модуляция 565
  - ◊ фазовая модуляция 565
  - ◊ частотная модуляция 566
  - ◊ широтно-импульсная модуляция 567
- ◊ mscohere 358
- ◊ msepred 616
- ◊ msesim 616
- ◊ mskdemod 584
- ◊ mskmod 584
- ◊ nargin 678
- ◊ nargout 678
- ◊ noisepsd 437, 486, 492, 497
- ◊ noperations 481, 499
- ◊ normalize 487
- ◊ normcdf 65
- ◊ norminv 65
- ◊ normpdf 65
- ◊ nooverflows 481, 499
- ◊ nsections 272
- ◊ nstages 272
- ◊ nstates 272, 273
- ◊ num2bin 481
- ◊ num2hex 481
- ◊ num2int 481
- ◊ numel 665
- ◊ nunderflows 482
- ◊ nuttallwin 343
- ◊ optimizeunitygains 499
- ◊ oqpskdemod 580
- ◊ oqpskmod 580
- ◊ order 272
- ◊ pamdemod 580
- ◊ pammod 580
- ◊ parzenwin 349
- ◊ pburg 359
- ◊ pcOLOR 696
- ◊ pcov 359
- ◊ peig 369
- ◊ periodogram 355
- ◊ phasedelay 254, 270
- ◊ phasez 253, 270
- ◊ plot 104, 168, 170, 370, 689
- ◊ plot3 692
- ◊ plotyy 692
- ◊ pmcov 359
- ◊ pmdemod 576
- ◊ pmmod 575
- ◊ pmusic 366
- ◊ polar 692
- ◊ poly 107
- ◊ polyder 136
- ◊ polyval 136
- ◊ print 703
- ◊ prony 404
- ◊ pskdemod 583
- ◊ pskmod 583
- ◊ pulstran 151, 179

- Функция (*прод.*):
- ◊ pwelch 357
  - ◊ pyulear 359
  - ◊ qamdemod 581
  - ◊ qammmod 581
  - ◊ qfft 497
  - ◊ qfunc 65
  - ◊ qreport 482, 487, 490, 499
  - ◊ quantiz 475
  - ◊ quantize 477
  - ◊ quantizer 477
  - ◊ radix 499
  - ◊ rand 189, 191
  - ◊ randint 543
  - ◊ randn 189, 193
  - ◊ randquant 482
  - ◊ range 482, 499
  - ◊ raylcdf 82
  - ◊ raylinv 82
  - ◊ raylpdf 82
  - ◊ raylrnd 82
  - ◊ raylstat 82
  - ◊ rcosfir 414
  - ◊ rcosflt 588
  - ◊ rcosiir 414
  - ◊ rcosine 413
  - ◊ realzemd1 272
  - ◊ realmax 482
  - ◊ realmin 482
  - ◊ rectpuls 172
  - ◊ rectpulse 588
  - ◊ rectwin 339
  - ◊ reffilter 487
  - ◊ remez 422
  - ◊ remezord 422
  - ◊ removestage 272
  - ◊ repmat 550
  - ◊ resample 652
  - ◊ reset 272, 482, 499, 580
  - ◊ residue 108
  - ◊ reziduez 261
  - ◊ rlevinson 359, 362
  - ◊ rooteig 369
  - ◊ rootmusic 368
  - ◊ roots 107
  - ◊ round 482, 553
  - ◊ saveas 703
  - ◊ sawtooth 183
  - ◊ scatterplot 585
  - ◊ semilogx 692
  - ◊ semilogy 692
  - ◊ set 267
  - ◊ set2int 487
  - ◊ setstage 272
  - ◊ sinc 175
  - ◊ size 664
  - ◊ softscope 204
  - ◊ sos 273
  - ◊ sos2ss 263
  - ◊ sos2tf 263
  - ◊ sos2zp 263
  - ◊ sosfilt 266
  - ◊ sound 200
  - ◊ soundsc 200, 550, 620
  - ◊ spectrogram 188, 353
  - ◊ sprintf 679
  - ◊ square 182
  - ◊ ss 273
  - ◊ ss2sos 264
  - ◊ ss2tf 107, 261
  - ◊ ss2zp 107, 261
  - ◊ ssbdemod 575
  - ◊ ssbmod 573
  - ◊ stairs 169, 692
  - ◊ stem 169, 249, 692
  - ◊ stem3 696
  - ◊ stepz 250, 270
  - ◊ stmcb 406
  - ◊ strips 197
  - ◊ sub2ind 667
  - ◊ subplot 197, 700
  - ◊ surf 693
  - ◊ surf3 696
  - ◊ symerr 553
  - ◊ text 699
  - ◊ tf 273
  - ◊ tf2sos 263
  - ◊ tf2ss 107, 261
  - ◊ tf2zp 106, 261
  - ◊ tfestimate 359
  - ◊ tic 682
  - ◊ title 698
  - ◊ toc 682
  - ◊ toeplitz 167
  - ◊ tostring 482, 499
  - ◊ triang 341
  - ◊ tripuls 173
  - ◊ tukeywin 346
  - ◊ twiddles 500
  - ◊ udecode 474
  - ◊ uencode 473
  - ◊ unitquantize 479
  - ◊ unitquantizer 479
  - ◊ unwrap 103, 253

- ◊ upfirnd 652
- ◊ upsample 647
- ◊ waterfall 696
- ◊ wavplay 201
- ◊ wavread 195
- ◊ wavrecord 201
- ◊ wavwrite 199
- ◊ wgn 193
- ◊ wintool 351
- ◊ wordlength 482
- ◊ wvtool 340
- ◊ xlabel 698
- ◊ xlim 699
- ◊ ylabel 698
- ◊ ylim 699
- ◊ yulewalk 402
- ◊ zerophase 253, 270
- ◊ zeros 684
- ◊ zlabel 698
- ◊ zlim 699
- ◊ zp2sos 263
- ◊ zp2ss 107, 261
- ◊ zp2tf 107, 261
- ◊ zpk 273
- ◊ zplane 258, 270
- ◊ анонимная 229
- ◊ Бесселя 528
- ◊ весовая 318, 339, 384
- ◊ включения 10
- ◊ вложенная 674
- ◊ Дирака, См. Дельта-функция 9
- ◊ Дирихле 184, 204
- ◊ дискретная экспоненциальная 160
- ◊ единичная импульсная 159
- ◊ единичного скачка 10, 87
  - спектр 38
- ◊ кусочно-заданная 171
- ◊ ошибок 65
- ◊ передачи 212
- ◊ распределения вероятности 61
  - преобразование 190
- ◊ с переменным числом параметров 678
- ◊ системная 212
- ◊ субфункция 674
- ◊ фазовая 46
- ◊ Хевисайда 10
- ◊ эллиптическая рациональная 118

**X**

Характеристика квантования 482

**Ц**

- Центральная предельная теорема 65, 91
- Центральная частота 48
- Цифроаналоговый преобразователь (ЦАП)
  - 140, 147, 169
  - ◊ ШИМ-реализация 559
- Цифровой процессор 140
- ЦП 140

**Ч**

- Частичный фильтр 634
- Частное 245
- Частота 9
  - ◊ биений 514
  - ◊ дискретизации 138, 168, 195
    - повышение 626
    - понижение 625
  - ◊ круговая 9
  - ◊ ложная 145
  - ◊ мгновенная 186, 525
  - ◊ Найквиста 140
  - ◊ средняя 110
  - ◊ среза 109, 110
  - ◊ центральная 48
- Частота повторения сигнала 9
- Частотная дисперсия 557
- Частотная манипуляция 542
  - ◊ демодуляция 544
    - когерентная 544
    - некогерентная 544
  - ◊ минимальная 546
  - ◊ с непрерывной фазой 543
- Частотная модуляция (ЧМ) 525
  - ◊ демодуляция 536, 538
- Частотная область 24
- Частотная характеристика 212
  - ◊ неустойчивой системы 97
- Частотно-временной анализ 58
- Частотное разделение каналов 502
- Частотные преобразования фильтров 122
- Частоты:
  - ◊ ложные 141
  - ◊ экстремальные 379
- Чебышевская аппроксимация 379
- ЧМн 542

**Ш**

- Шаг дискретизации 138
- Широкополосный сигнал 27
- Широтно-импульсная модуляция (ШИМ) 558

Шум 58  
◊ белый 78  
◊ квантования 139, 453  
◊ размах 455  
◊ коррелированный 193, 366  
Шумовое подпространство 336

## Э

Эквалайзер 609  
◊ дробный 615  
Экспоненциальное забывание 606  
Экспоненциальный формат 450  
Экстремальные частоты 379  
Энергетический спектр 44

Энергия 11, 40, 43  
◊ случайного процесса 73  
◊ удельная 12  
Эргодичность 71  
Эрмитово сопряжение (апостроф) 659  
Эффект Гиббса 20, 378  
Эффективная ширина спектра 26, 77  
Эффективные значения коэффициентов 472  
Эхоподавление 610

## Я

Ядро преобразования 56  
◊ самосопряженное 57  
Ячейка памяти 234