Быстрые алгоритмы в цифровой обработке изображений

Преобразования и медианные фильтры

Под редакцией
Т.С. Хуанга

Перевод с английского
под редакцией
Л.П. Ярославского

МОСКВА „РАДИО И СВЯЗЬ" 1984

Two-Dimensional Digital Signal Processing II

Transforms and Median Filters

Edited by T. S. Huang

With Contributions by
J.-O. Eklundh T. S. Huang B. I. Justusson
H. J. Nussbaumer S. G. Tyan S. Zohar

Springer-Verlag Berlin Heidelberg New York 1981

УДК 621.391:681.301

Т. С. Хуанг, Дж.-О. Эклунд, Г. Дж. Нуссбаумер, Ш. Зохар, Б. И. Юстуссон, Ш.-Г. Тян.

Быстрые алгоритмы в цифровой обработке изображений/Т. С. Хуанг, Дж.-О. Эклунд, Г. Дж. Нуссбаумер и др.; Под ред. Т. С. Хуанга: Пер. с англ. — М.: Радио и связь, 1984. — 224 с., ил.

Изложены основы теории и применения новых эффективных в вычислительном отношении алгоритмов цифровой обработки изображений. Рассмотрены алгоритмы быстрого транспонирования двумерных массивов, хранящихся во внешних запоминающих устройствах, принципы организации вычислений при реализации алгоритма Винограда дискретного преобразования Фурье, позволяющего выполнить его с уменьшенным, по сравнению с известными алгоритмами быстрого преобразования Фурье, числом умножений; раскрыты основные идеи, лежащие в основе полиномиальных преобразований и их использования для вычисления свертки и спектрального анализа двумерных сигналов; с позиций детерминистического и статистического подходов обсуждаются свойства медианных фильтров и их возможные применения в обработке изображений.

Для научных работников, занимающихся цифровой обработкой изображений и других двумерных сигналов и вопросами ее математического и аппаратурного обеспечения.

Табл. 26. Ил. 49. Библиограф. 168 назв.

Редакция переводной литературы

Перевод с английского **А. А. Гурова, В. Б. Макулова, С. Л. Ярославского**

Б $\frac{1502000000-130}{046(01)-84}$ **52—84**

Предисловие редактора перевода

Читателю, взявшему в руки эту книгу, нет надобности объяснять, насколько актуальными сейчас являются задачи цифровой обработки изображений. Но, может быть, не всякий читатель ощутил на себе, как велика дистанция от первых опытов, показывающих, что ту или иную задачу обработки изображений можно решить на ЭВМ в принципе, до реальной регулярной работы. И это — не только и не столько технические трудности создания или «добывания» достаточно мощной ЭВМ и устройств ввода-вывода изображений, а трудности принципиальные.

Они связаны с тем, что изображение — это двумерный и, вообще говоря, векторный сигнал, несущий огромное количество информации. Кроме того, изображение часто выступает как особый сигнал, предназначенный для визуального восприятия. В этих случаях требуется, чтобы обработка выполнялась в реальном масштабе времени пользователя, т. е. за секунды или доли секунды на один кадр. Цветное изображение, содержащее $500\times$ $\times 500$ элементов (примерно соответствует телевизионному вещательному стандарту), по современным представлениям — весьма скромное по объему изображение, а ведь это массив в 750 000 байтов, который необходимо обработать за секунды, причем только пересылка этого массива со скоростью 1 Мбайт/с занимает почти секунду.

Преодолеть это «проклятие размерности» можно лишь правильной, рациональной организацией вычислений, такой организацией, которая не только могла бы «бороться» с двумерностью, но и использовала бы двумерность для эффективной работы вычислительных устройств последовательного действия, которыми мы в настоящее время располагаем.

Если известна конкретная задача обработки изображений, можно искать эффективные способы обработки, учитывая особенности и ограничения этой конкретной узкой задачи. Но если, как требуется сейчас, говорить о регулярной обработке изображений для решения достаточно широкого класса задач в этой области и создавать для этого аппаратурно-программно-алгоритмическое обеспечение, то необходимо выделить набор определенных стандартных операций, из которых, как из готовых блоков, можно строить алгоритм (и соответственно программу или цифровой процессор) для решения произвольной задачи. Такими операциями являются двумерная свертка, двумерное дискретное преоб-

разование Фурье и другие линейные ортогональные преобразования.

Хорошо известно, как стимулировало развитие цифровой обработки изображений изобретение алгоритма быстрого преобразования Фурье и других быстрых алгоритмов ортогональных преобразований. В последние годы в этом отношении наметился существенный прогресс, связанный с теоретико-числовыми преобразованиями, алгоритмом Винограда вычисления дискретного преобразования Фурье и, наконец, полиномиальными преобразованиями, где удается воспользоваться двумерностью данных для ускорения вычислений. Предлагаемая читателю книга поможет специалистам освоить эти достижения.

Но ДПФ и свертка — это только линейные преобразования. В арсенале средств обработки изображений необходимы и нелинейные преобразования. Конечно, произвольное преобразование цифрового сигнала можно реализовать из линейных и нелинейных преобразований отдельных отсчетов сигнала (поэлементных нелинейных преобразований). Однако все-таки желательно иметь более крупные блоки, чем поэлементные преобразования.

Особенность изображений как двумерных сигналов состоит в том, что отдельные элементы изображения меняются обычно не независимо друг от друга, а находятся в определенной связи с соседними элементами. Поэтому большинство алгоритмов преобразования изображений носит локальный характер, т. е. обрабатывают изображения сразу по группам элементов, располагающихся в некоторой окрестности вокруг данного. Линейные преобразования легко и естественно удовлетворяют такому свойству локальности и при этом допускают построение алгоритмов, вычислительная сложность которых мало зависит от размеров охватываемой окрестности. Такие же свойства требуются и от нелинейных преобразований изображений.

В настоящее время намечается формирование одного весьма полезного класса нелинейных преобразований, обладающего свойствами локальности и простоты вычислений. Его составляют алгоритмы, которые можно было бы назвать алгоритмами ранговой фильтрации, так как они основаны на измерении локальных ранговых статистик изображений. Поскольку любые ранговые статистики можно найти из локальных гистограмм, а локальные гистограммы можно вычислить рекурсивно, в принципе вычислительная сложность алгоритмов ранговой фильтрации почти не зависит от размеров окрестности. При вычислении конкретных ранговых статистик и производных от них возможны дальнейшие упрощения, связанные, в частности, с информационной избыточностью изображений.

Наиболее известный сейчас алгоритм этого класса — алгоритм медианной фильтрации. Ему посвящены две главы предлагаемой книги, которые, безусловно, привлекут внимание специалистов, так как в них впервые подробно и систематически разбираются

свойства медианных фильтров. Как известно, медиана является робастной (устойчивой к распределению) оценкой среднего значения выборки, и именно свойство робастности определяет преимущества медианного фильтра перед фильтрами, вычисляющими локальное среднее и использовавшимися до сих пор для сглаживания изображений. Эта связь медианной фильтрации с робастными алгоритмами оценки параметров, развиваемыми в настоящее время в математической статистике, не показана в книге, и поэтому на нее целесообразно указать особо. Она дает повод для далеко идущих обобщений медианных фильтров, например, обобщений в сторону создания медианных согласованных двумерных фильтров как робастных аналогов хорошо известных линейных согласованных фильтров.

Другими примерами ранговых алгоритмов могут служить алгоритмы экстремальной фильтрации, которые заменяют анализируемый элемент изображения максимумом или минимумом по окрестности. А если заменять значение элемента изображения его рангом по окрестности, т. е. номером, который данный элемент займет в ряду расположенных по возрастанию значений элементов в заданной окрестности, очевидно, мы получим не что иное, как хорошо известный алгоритм скользящей эквализации. Таким образом, к ранговым алгоритмам можно отнести и скользящую эквализацию, и другие алгоритмы адаптивных амплитудных преобразований, основанные на анализе локальных гистограмм. Эта связь подчеркивает еще одно свойство ранговых алгоритмов, их локальную адаптацию к характеристикам обрабатываемого изображения и потенциальные возможности использования их не только для робастного сглаживания, но и для выделения признаков при препарировании и автоматическом распознавании изображений. Эти соображения также стоит учитывать при чтении последних двух глав книги.

При переводе книги и переводчики, и редактор столкнулись с рядом трудностей, притом не только терминологического порядка. Книга невелика по объему, но очень насыщена новыми фактами, излагаемыми, особенно в гл. 2—4, в достаточно, а иногда, может быть, чрезмерно общей форме. Поэтому мы постарались по возможности облегчить ее чтение с помощью примечаний. Надеемся, что читателю окажется полезным также предлагаемый редактором дополнительный список литературы на русском языке по вопросам цифровой обработки изображений и быстрым алгоритмам обработки.

Перевод книги выполнен В. Б. Макуловым (предисловие, гл. 1, 2, 4), А. А. Гуровым (гл. 3) и С. Л. Ярославским (гл. 5, 6).

Д-р физ.-мат. наук
Л. П. Ярославский

Предисловие

В последнее десятилетие размах исследований в области цифровой обработки изображений стремительно возрос. И это не удивительно, если учесть, что в широком смысле обработка изображений — это обработка многомерных сигналов, а большинство сигналов в реальном мире является многомерными. Фактически и одномерные сигналы, с которыми мы работаем, часто представляют собой упрощенную копию многомерных сигналов. Например, речь часто рассматривают как одномерный сигнал, т. е. функцию одной переменной (времени). Однако в действительности речевой сигнал существует в пространстве и поэтому является функций четырех переменных (трех пространственных переменных и времени).

Существуют аналоговые (оптические и электронно-оптические) и цифровые методы обработки изображений. Из-за присущих цифровым методам преимуществ (гибкости, точности), быстрого развития вычислительной техники и таких связанных с ней отраслей техники, как производство схем высокой и сверхвысокой степени интеграции, можно смело сказать, что, за исключением некоторых специальных задач, предпочтение отдается обычно цифровым методам.

Цель данной книги, как и предыдущего тома (Two-Dimensional Digital Signal Processing I: Linear Filters), — дать глубокий анализ трех важнейших классов цифровых методов решения задач обработки изображений: линейной фильтрации, преобразований и медианной фильтрации. Эти книги взаимосвязаны, но пользоваться ими можно и независимо друг от друга.

В шестом томе серии Picture Processing and Digital Filtering (первое издание, 1975) [1] были подробно рассмотрены избранные задачи цифровой обработки двумерных сигналов, включая преобразования, проектирование фильтров и реставрацию изображений. С тех пор в этих направлениях был достигнут значительный прогресс. В 1978 г., когда планировалось второе издание этого тома (опубликовано в 1979 г.), мы решили не вносить существенных изменений, а лишь добавить новую главу, содержащую краткий обзор наиболее современных достижений. Мы рассчитывали, что подробное рассмотрение некоторых важных новых результатов

[1] Имеется перевод: Обработка изображений и цифровая фильтрация/Под ред. Т. Хуанга: Пер. с англ. — М.: Мир, 1979. — 318 с. — *Прим. ред.*

появится в последующих томах физической серии издательства Шпрингер.

Две книги, о которых здесь идет речь, стали первыми из задуманных томов. Материал в них разделен на три части. В первой, посвященной линейным фильтрам, представлены полученные недавно основные результаты в области синтеза рекурсивных и нерекурсивных фильтров, исследования их устойчивости, калмановской фильтрации (с приложением к улучшению визуального качества и реставрации изображений). Среди наиболее важных вопросов здесь рассмотрены вопросы проектирования и исследования устойчивости рекурсивных по полуплоскости фильтров, представляющих в настоящее время большой интерес.

Вторая и третья части вошли в настоящий том. В разделах, посвященных преобразованиям, обсуждаются две задачи: алгоритмы транспонирования больших матриц и теоретико-числовые методы вычисления преобразований и свертки. Приведен детальный вывод алгоритма Винограда преобразования Фурье.

В первой и второй частях основное внимание уделено линейной обработке. В третьей части, посвященной медианной фильтрации, изучается особый метод нелинейной обработки. Медианная фильтрация стала довольно популярной в обработке изображений и речевых сигналов. Однако по этим вопросам опубликованы лишь скудные материалы. Две главы третьей части содержат новые результаты, большинство из которых приводятся здесь впервые.

Главы в этом томе носят дидактический характер. В то же время они выводят читателя на самый передний край современных исследований. Книга будет полезна как справочник научным работникам и инженерам, может служить дополнительным учебным пособием при чтении курсов различной сложности по цифровой обработке сигналов, обработке изображений и цифровой фильтрации.

Урбана, Иллинойс, сентябрь 1980 *Томас С. Хуанг*

ГЛАВА 1

ВВЕДЕНИЕ

(Т. С. Хуанг) [1]

Цель этого тома заключается в подробном рассмотрении двумерных цифровых преобразований и медианной фильтрации. Такой выбор продиктован приложениями этих математических операций к обработке изображений.

Существуют три основные области обработки изображений [1.1]: эффективное кодирование, реставрация и улучшение визуального качества изображений, распознавание образов. Многие методы реставрации и улучшения визуального качества изображений основаны на использовании линейных пространственно-инвариантных (ЛПИ) фильтров. Такие фильтры подробно рассмотрены в [1.1, 1.2]. Преобразования и связанные с ними методы позволяют построить эффективную с вычислительной точки зрения реализацию ЛПИ-фильтров. Многие преобразования полезны также при эффективном кодировании и выделении признаков в распознавании образов.

Для успешной реставрации изображений и улучшения их визуального качества часто требуются нелинейные методы. Одним из таких методов, популярных в последнее время не только в обработке изображений, но и вообще в обработке сигналов, является медианная фильтрация. Этот метод может применяться также при решении некоторых задач, связанных с распознаванием, например, утоньшения линий и выделения небольших изолированных объектов на изображении.

В следующих разделах более подробно обсуждается содержание глав этой книги.

1.1. Преобразования

Различные двумерные преобразования рассматривались с единых позиций (на основе понятия внешнего произведения) в [1.2, гл. 2]. Более детальное обсуждение некоторых из них можно найти в [1.3]. Среди этих преобразований наиболее широко применяется, несомненно, преобразование Фурье. Другие полезны главным образом в кодировании изображений и отчасти — в задачах распознавания. Опыт последних лет показал, что среди независимых от изображений преобразований (в эту категорию не входит преобразование Карунена — Лоэва) наилучшие результаты в ко-

[1] Department of Electrical Engineering and Coordinated Science Lab., University of Illinois, Urbana, IL61801, USA.

дировании изображений дают дискретное косинусное преобразование (ДКП) и слэнт-преобразование[1]. Как показано в [1.4], с теоретической точки зрения ДКП асимптотически оптимально для всех марковских сигналов конечного порядка[2]. Джейн недавно ввел новое семейство унитарных преобразований, которое включает многие известные преобразования, такие, как ДКП [1.5].

Разложение по сингулярным значениям (РСЗ) и его приложения достаточно глубоко рассмотрены в [1.2, гл. 1, 2]. Недавно было проведено сопоставление поведения сингулярных значений с автокорреляционной функцией изображения [1.6]. Разработан новый метод подавления шума при реставрации изображений посредством РСЗ [1.7].

Обычный метод вычисления ДКП состоит в использовании быстрого преобразования Фурье [1.8, 1.9]. Однако новейшие быстрые алгоритмы обещают увеличение скорости в шесть раз [1.10].

Для выполнения двумерных преобразований, таких, как преобразование Фурье или Адамара, на ЭВМ, чья память на магнитных сердечниках не позволяет хранить изображение целиком, необходима дополнительная память. При этом обычно используется транспонирование матриц. Эффективный алгоритм транспонирования матриц был предложен в 1972 г. Эклундом [1.11]. Несколько позже он разработал два новых алгоритма и получил некоторые результаты по оптимальной стратегии [1.12] (см. также [1.13]). С другой стороны, двумерное дискретное преобразование Фурье (ДПФ) и аналогичные ему преобразования можно выполнять и без транспонирования матриц [1.14—1.18].

Интересной новой областью исследований является использование теоретико-числовых методов в обработке сигналов. Первым, кто предложил применить теоретико-числовые преобразования (ТЧП) (например, преобразования по числам Ферма) для вычисления быстрой двумерной свертки, был Рейдер. Блестящим введением в эту идею является раздел, написанный им в [1.24]. Детальное изложение можно найти в [1.25, 1.26].

Используя ТЧП, заметную экономию вычислений можно получить в тех случаях, когда: 1) одномерные последовательности, подлежащие обработке, относительно коротки; 2) необходима значительная точность и 3) операции умножения дороже операций сложения.

---

[1] В оригинале slant transform. Этот термин иногда переводится как преобразование по наклонному базису или преобразование по пилообразному базису; смысл состоит в том, что базисные функции имеют кусочно-пилообразную форму. — *Прим. ред.*

[2] Преимущество дискретного косинусного преобразования связано с более фундаментальным фактом — ДКП соответствует преобразованию Фурье четным образом продолженных последовательностей. Благодаря четному продолжению устраняются разрывы на краях последовательности и ее ряд Фурье сходится намного быстрее [1]. — *Прим. ред.*

Виноград применил теоретико-числовые методы к вычислению ДПФ [1.27—1.30]. При этом число умножений значительно снижается по сравнению с БПФ, а число сложений почти не меняется. Например, для 1024-точечной последовательности БПФ требует 12 288 умножений и 26 624 сложений, тогда как алгоритм Винограда преобразования Фурье (АВПФ) для 1008 точек требует 4212 умножений и 25 224 сложений. Дополнительные сведения и соображения по программной реализации можно найти в [1.31—1.34].

АВПФ реализован в быстродействующем процессоре обработки сигналов [1.35]. Ошибки квантования (обусловленные округлением и квантованием коэффициентов) при АВПФ для случая вычислений с фиксированной запятой изучались в [1.36]. Было обнаружено, что в общем случае при той же ошибке вычислений АВПФ требует для представления данных на 1—2 бита больше, чем алгоритм БПФ.

Материал по преобразованиям в книге разделен на две части: алгоритмы транспонирования матриц и теоретико-числовые методы. В гл. 2 обсуждаются некоторые эффективные алгоритмы транспонирования матриц, а также прямой метод преобразования Андерсона [1.14]. В гл. 3 описываются быстрые алгоритмы цифровой свертки и преобразования Фурье, основанные на полиномиальных преобразованиях. В некоторых алгоритмах используется комбинация полиномиальных преобразований и алгоритма Винограда. Подробный вывод АВПФ приведен в гл. 4.

1.2. Медианные фильтры

Линейные пространственно-инвариантные (ЛПИ) фильтры полезны для реставрации и улучшения визуального качества изображений. Их можно применять, например, при реализации винеровских фильтров для снижения уровня шума на изображениях. Однако, чтобы подавить шум и при этом сохранить контурную часть изображений, приходится применять нелинейные или линейные пространственно-неинвариантные (ЛПНИ) фильтры. Ограничения на использование ЛПИ-фильтров в задачах реставрации изображений обсуждаются в [1.2, гл. 1, 5].

Многие нелинейные и ЛПНИ-фильтры для реставрации изображений описаны в [1.2, гл. 1, 5]. В гл. 5 предыдущего тома, посвященного линейным фильтрам [1.1], были описаны калмановские ЛПНИ-фильтры, используемые для подавления шума при реставрации изображений. В гл. 5 и 6 этого тома рассмотрена особая нелинейная процедура — медианная фильтрация. Обнаружено, что применение медианных фильтров эффективно для подавления некоторых видов шума и периодических помех без одновременного искажения сигнала [1.37—1.39]. Такие фильтры стали весьма популярны в обработке изображений и речевых сигналов.

Поскольку теоретический анализ поведения медианных фильтров очень труден, опубликовано очень мало результатов по этому вопросу. Две главы нашей книги содержат в основном новые результаты, не освещенные до сих пор в открытой литературе. В гл. 5 рассматриваются статистические свойства медианных фильтров. В частности, излагаются различные свойства выходного сигнала медианного фильтра при гауссовском шуме или сумме ступенчатой функции и гауссовского шума на входе.

Глава 6 посвящена детерминированным свойствам медианных фильтров. Особенно интересными представляются результаты, относящиеся к так называемым стабильным точкам медианных фильтров. Стабильной точкой является последовательность (в одномерном случае) или массив (в двумерном случае), которые не изменяются при медианной фильтрации. В гл. 6 Тян показал, что в одномерном случае стабильными точками медианных фильтров являются «локально-монотонные» последовательности. Исключение составляют некоторые периодические двоичные последовательности. В последнее время Галлагер и Вайс [1.40] сумели устранить это исключение, ограничив длину последовательностей.

В гл. 6 кратко описан эффективный алгоритм [1.39] медианной фильтрации, основанный на модификации гистограмм. В [1.41, 1.42] обсуждается аппаратурная реализация медианной фильтрации в реальном масштабе времени на основе цифровых избирательных схем. Метод нахождения медианы, основанный на двоичном представлении элементов изображения в апертуре фильтра, предложен в [1.43], где сравниваются аппаратурная реализация этого метода, алгоритм преобразования гистограмм и метод цифровых избирательных схем по сложности и скорости. Реализация медианных фильтров на двоичном матричном процессоре рассмотрена в [1.41]. Разработан метод реализации медианных фильтров в конвейерном процессоре, работающем синхронно с видеосигналом [1.45].

В гл. 5 и 6 изложен материал главным образом теоретического характера. В качестве дополнения представим здесь некоторые экспериментальные результаты. На рис. 1.1 показаны примеры стабильных точек медианных фильтров. Даны исходное изображение (*а*) и результаты шестикратного применения трех различных медианных фильтров (*б*). Дальнейшее применение фильтров не вносит существенных изменений в результаты. Таким образом, изображения на рис. 1.1, *б*—*г* являются стабильными точками трех медианных фильтров.

Медианные фильтры особенно удобны для борьбы с импульсным (точечным) шумом. Этот факт иллюстрируется на рис. 1.2. На рис. 1.2, *а* показан результат передачи изображения 1.1, *а* по двоичному симметричному каналу с шумом при использовании импульсно-кодовой модуляции. В этом случае на изображении появляется импульсный шум. Применение медианного фильтра позволяет подавить бо́льшую часть шумовых выбросов (рис. 1.2, *б*),

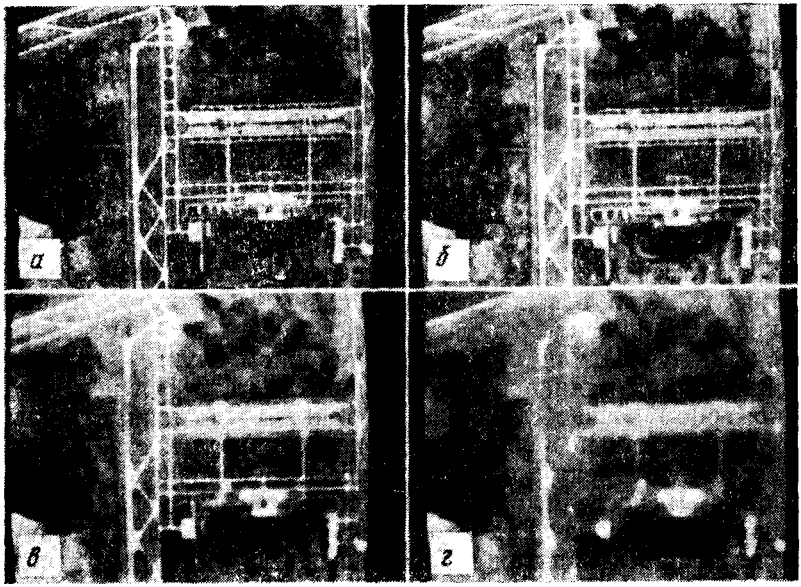


Рис. 1.1. Стабильные точки медианных фильтров:
а — аэрофотоснимок части аэропорта, дискретизированный на 256×256 элементов по 8 бит/отсчет; *б* — изображение, подвергнутое шестикратной медианной фильтрации с крестообразной апертурой 3×3; *в* — то же, при квадратной апертуре 3×3; *г* — то же, при квадратной апертуре 5×5

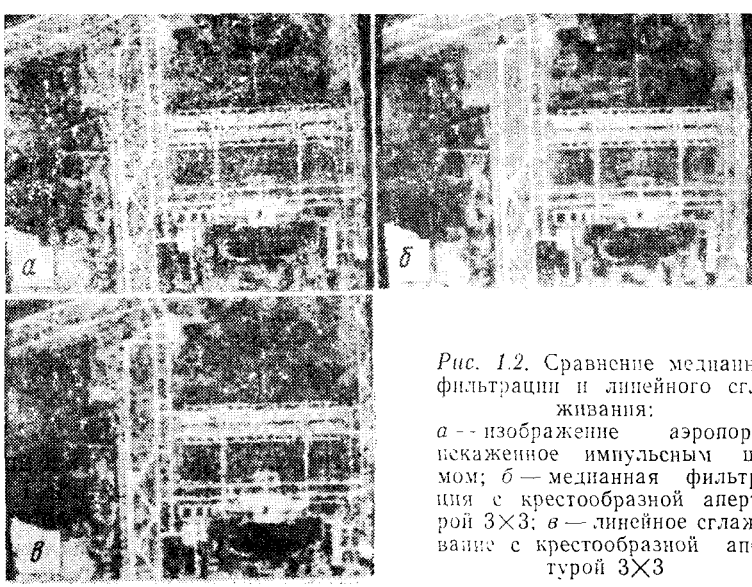


Рис. 1.2. Сравнение медианной фильтрации и линейного сглаживания:
а — изображение аэропорта, искаженное импульсным шумом; *б* — медианная фильтрация с крестообразной апертурой 3×3; *в* — линейное сглаживание с крестообразной апертурой 3×3

в то время как линейное сглаживание оказывается совершенно неэффективным (рис. 1.2, *в*).

Хотя в гл. 5 и 6 обсуждаются двумерные (пространственные) фильтры, очевидно, что к движущимся изображениям, таким, как телевизионные, могут применяться трехмерные медианные фильтры (пространственно-временные), т. е. апертура фильтра может быть трехмерной. Медианная временна́я фильтрация особенно удобна для подавления пачек шумовых выбросов, включая выпадение строк. Кроме того, она намного лучше, чем временно́е усреднение (линейное сглаживание), сохраняет движение. В [1.46] описано несколько экспериментов по временной фильтрации (включая фильтрацию с компенсацией движения). В одном из экспериментов по фильтрации последовательность кадров панорамирования, содержащая белый гауссовский шум и случайные выпадения строк, подвергалась медианной фильтрации и линейному сглаживанию. Кадровая частота последовательности составляла 30 кадров/с, каждый кадр содержал примерно 200 строк по 256 элементов в каждой с 8 бит/отсчет. Панорамирование проводилось горизонтально со скоростью примерно 5 элементов изображения на кадр. Результаты по одному кадру показаны на рис. 1.3: зашумленный исходный кадр (*а*), тот же кадр после линейного сглаживания (*б*) и кадр, обработанный медианным фильтром (*в*). Необходимо отметить, что медианный фильтр дает

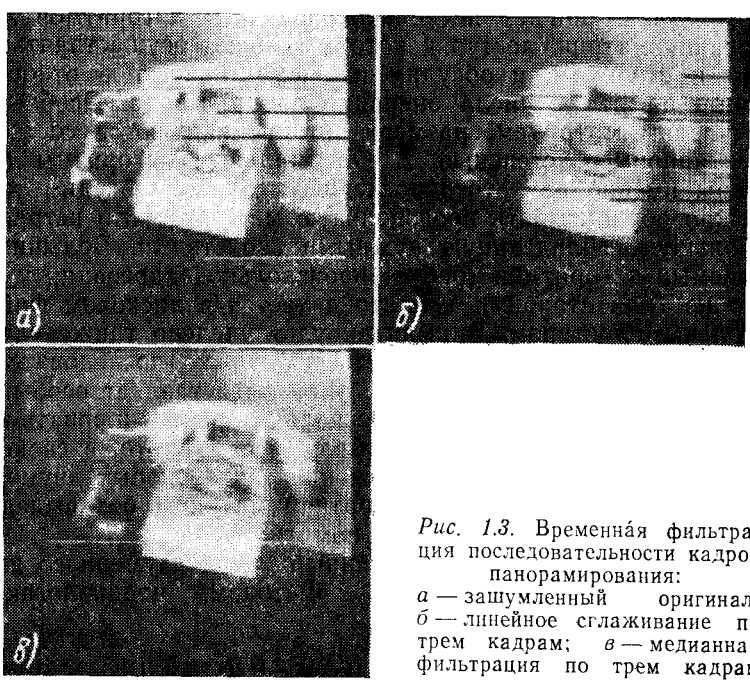


Рис. 1.3. Временна́я фильтрация последовательности кадров панорамирования:
а — зашумленный оригинал; *б* — линейное сглаживание по трем кадрам; *в* — медианная фильтрация по трем кадрам

намного лучшие результаты в отношении снижения числа выпадений строк и сохранения резкости контуров. Однако для подавления гауссовского шума более эффективно линейное сглаживание. Приведенные данные согласуются с теоретическими (см. гл. 5 и 6).

Хотя и медианная фильтрация и линейное сглаживание используются для улучшения субъективного качества изображения, пока не ясно, способствуют ли они дальнейшему машинному анализу изображений — распознаванию образов или измерениям на изображении. Были проведены тщательные исследования влияния линейной и медианной фильтрации на эффективность выделения контуров, анализ формы и текстурный анализ. Некоторые результаты приведены в [1.47].

ГЛАВА 2

ЭФФЕКТИВНЫЕ МЕТОДЫ ТРАНСПОНИРОВАНИЯ МАТРИЦ

(Дж.-О. Эклунд) [1]

В этой главе опишем несколько различных алгоритмов транспонирования матриц, доступ к которым может осуществляться по строкам и столбцам, и обсудим их эффективность в отношении требуемой памяти и числа операций ввода-вывода. Особое внимание уделим матрицам, размеры которых являются составным числом, например степенью 2. Оптимальные алгоритмы будут построены именно для этого последнего случая. Обсудим также, как можно обрабатывать произвольную матрицу, если встраивать ее в бóльшую, обладающую нужными свойствами. Большинство описываемых алгоритмов требует произвольной адресации к каждой строке (или столбцу), как это делается в дисковом файле с произвольным доступом. Внимание будет уделено также случаю, когда доступ к строкам осуществляется только последовательно, как в ленточном файле. Первый случай представляет особый интерес, так как некоторые методы позволяют транспонировать квадратную матрицу, оставляя ее на своем месте. А это, в свою очередь, означает возможность выполнения также любых двумерных разделимых преобразований квадратной матрицы без выделения дополнительного места во внешней памяти.

Рассмотрим также непосредственный метод двумерного преобразования Фурье. Сравнение его с методом, предполагающим

[1] National Defense Research Institute (FOA), P. O. Box 1165 S-581 11 Linköping, Sweden.

транспонирование, показывает, что в обычных приложениях они по существу равноценны по эффективности. Однако имеются и важные различия.

2.1. Транспонирование матриц в обработке сигналов

Интегральные преобразования (например, преобразование Фурье) являются важным вычислительным инструментом в цифровой обработке сигналов. В двумерном случае их дискретные представления являются двойными суммами, вычисляемыми последовательно по двум координатам. Эти двойные суммы можно вычислять непосредственно, когда преобразуемый массив целиком размещается в оперативном запоминающем устройстве (ОЗУ). Задача усложняется, когда это невозможно или невыгодно. Часто матрицы хранятся в блок-ориентированных запоминающих устройствах, например на дисках, когда наименьшей доступной записью является полная строка или полный столбец. В этом случае непосредственное вычисление преобразования становится дорогим вследствие затрат времени на доступ к матрице, хранящейся во внешнем запоминающем устройстве (ВЗУ).

Один из способов решения этой проблемы заключается в транспонировании матрицы после преобразования по строкам. Другой способ, предложенный в [2.1], вытекает непосредственно из определения быстрого преобразования Фурье (БПФ); при этом необходимо помнить, что БПФ строится как последовательность преобразований над подмассивами и все эти преобразования выполняются с оставлением массива на месте.

Оба подхода основаны на разделимости преобразования Фурье. Двумерное дискретное преобразование Фурье (ДПФ) комплексного массива $x\ (m,\ n)$, $m=0,\ 1,\ \ldots,\ M-1$, $n=0,\ 1,\ \ldots,\ N-1$, можно определить следующим образом:

$$X(k,\ l)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x(m,\ n)\,W_M^{km}\,W_N^{ln},\quad k=0,\ 1,\ldots,\ M-1,$$

$$l=0,\ 1,\ldots,\ N-1, \tag{2.1}$$

где для удобства обозначено $W_p=\exp(-2\pi j/P)$. Разделимость означает, что ядро преобразования $W_M^{km}\,W_N^{ln}$ является произведением двух функций, одна из которых зависит только от m, а другая — от n. Это позволяет вычислять преобразование за два шага: сначала

$$Y(m,\ l)=\sum_{n=0}^{N-1} x(m,\ n)W_N^{ln}, \tag{2.2а}$$

а затем

$$X(k,\ l)=\sum_{m=0}^{M-1} Y(m,\ l)\,W_M^{km}, \tag{2.2б}$$

где первая сумма является N-точечным одномерным ДПФ по *строкам* матрицы $[x(m, n)]$, а вторая — M-точечным одномерным ДПФ по *столбцам* результирующей матрицы $[Y(m, l)]$. Следовательно, двумерное преобразование можно вычислить с помощью одномерного преобразования и транспонирования матриц, что представляет особенный интерес, поскольку существует быстрый алгоритм для одномерного ДПФ и, кроме того, этот алгоритм можно реализовать аппаратурно.

С другой стороны, факт, что сумма в (2.2б) также является ДПФ, наводит на мысль о возможности непосредственного вычисления $[X(k, l)]$ с использованием методов итеративного суммирования для БПФ в сочетании с вводом и выводом преобразованных строк.

Подходы, основанные на транспонировании, применимы к любым разделимым преобразованиям, а прямые методы работают, если существует быстрый алгоритм преобразования, как например, в случае преобразования Адамара. Оба метода обобщаются также для более высоких размерностей. (О вычислении ДПФ с использованием траспонирования см. [2.2].)

Важнее, однако, что и одномерные преобразования больших массивов могут быть разбиты на преобразования меньших массивов и транспонирование матриц. В случае ДПФ это можно сделать в соответствии с [2.2].

ДПФ массива $x(m)$, $m=0, 1, \ldots, M-1$, можно определить следующим образом:

$$X(k)=\sum_{m=0}^{M-1} x(m) W_M^{km}, \quad k=0, 1, \ldots, M-1. \tag{2.3}$$

Теперь, если $M=M_0M_1$, запишем

$$m=m_1 M_0+m_0, \quad 0\leqslant m_0<M_0, \quad 0\leqslant m_1<M_1; \tag{2.4}$$

$$k=k_0 M_1+k_1, \quad 0\leqslant k_0<M_0, \quad 0\leqslant k_1<M_1, \tag{2.5}$$

и рассмотрим x и X как матрицы, положив

$$x(m)\Leftrightarrow y(m_0, m_1); \tag{2.6}$$

$$X(k)\Leftrightarrow Y(k_1, k_0). \tag{2.7}$$

Так как $W_M^{M_0M_1}=1$ и $W_M^{M_1}=W_{M_0}$, то

$$Y(k_1, k_0)=\sum_{m_0=0}^{M_0-1}\sum_{m_1=0}^{M_1-1} y(m_0, m_1) W_M^{(m_0+m_1 M_0)(k_1+k_0 M_1)}=$$

$$=\sum_{m_0=0}^{M_0-1}\sum_{m_1=0}^{M_1-1} y(m_0, m_1) W_M^{(m_0+m_1 M_0) k_1} W_{M_0}^{m_0 k_0}. \tag{2.8}$$

Вычислив сумму за два шага, получим

$$Z(m_0, k_1)=\sum_{m_1=0}^{M_1-1} y(m_0, m_1) W_M^{(m_0+m_1 M_0) k_1},$$

$$m_0=0, 1, \ldots, M_0-1, \quad k_1=0, 1, \ldots, M_1-1; \tag{2.9}$$

$$Y(k_1, k_0) = \sum_{m_0=0}^{M_0-1} Z(m_0, k_1) W_{M_0}^{m_0 k_0},$$

$$k_0 = 0, 1, \ldots, M_0 - 1, \; k_1 = 0, 1, \ldots, M_1 - 1. \qquad (2.10)$$

Здесь для данной строки y сначала вычисляем соответствующую *строку* Z путем перемножения соответствующих фазовых множителей и суммирования. Затем для данного *столбца* Z вычисляем его ДПФ, являющееся соответствующим столбцом Y. Для этого необходимо применить транспонирование Z.

Были предложены также другие методы вычисления одномерных и многомерных преобразований в ОЗУ ограниченного объема. Одни методы требуют транспортирования матриц (см., например, [2.2, 2.3]), другие не требуют (см., например, [2.4, 2.5]). Последние, в общем, менее эффективны.

2.2. Методы транспонирования матриц, хранящихся во внешних запоминающих устройствах

2.2.1. Определение критериев эффективности

В следующих разделах будет описан ряд методов транспонирования матриц, хранящихся в ОЗУ так, что каждая запись содержит одну строку (или столбец). В дальнейшем, однако, будем всегда считать, что матрицы записаны построчно. Работа всех алгоритмов, в общем, состоит в том, что они считывают ограниченное число записей в ОЗУ, переупорядочивают их, формируют новые выходные записи и заносят их в выходной файл, совпадающий или не совпадающий с входным файлом. Эффективность алгоритмов будет определяться с точки зрения требуемого числа ячеек RM ОЗУ и числа операций ввода-вывода IO (т. е. числа записей, которые нужно считать или записать). Эти показатели не учитывают всех вычислительных затрат, однако неучтенной частью можно пренебречь или считать ее неизменной для разных методов, если они применяются к одной и той же матрице. Точнее, существуют два основных неучтенных вида затрат.

К первому виду относятся затраты на переупорядочение данных в памяти для формирования новых выходных записей. Однако новые записи *на выходе* можно создавать без фактического перемещения данных в памяти. Например, можно использовать неявные циклы в операторах вывода в Фортране (см. разд. 2.7). Следовательно, затраты процессорного времени невелики и, кроме того, они пропорциональны произведению числа элементов матрицы на число просмотров данных, которое уже учитывается в числе операций ввода-вывода. Хотя мы считаем все алгоритмы построенными именно таким способом, для простоты будем говорить о транспонировании матриц и переупорядочении массивов в памяти.

Неточность оценки второго вида возникает вследствие того, что при учете числа пересылаемых записей мы пренебрегаем возможным различием длин записей для того или иного метода. Поэтому для большей точности необходимо учитывать также число слов, подлежащих пересылке. Однако затратами на это можно пренебречь по двум причинам. Во-первых, время доступа к одной записи обычно намного больше, чем время пересылки одного слова. Для современных быстродействующих накопителей на дисках отношение этих времен обычно превышает 1000. Во-вторых, тщательное рассмотрение различных методов показывает, что возможные изменения длины записи обычно малы по сравнению с этим отношением.

2.2.2. Простой метод блочного транспонирования

Простой и очевидный метод транспонирования матрицы $M\times N$ приведен в [2.1]. При транспонировании этим методом исходные записи длиной N слов разбиваются на части длиной K слов, где K выбирается так, чтобы KM элементов могли быть размещены в ОЗУ. Затем для каждого $i=1, \ldots, N/K$ в ОЗУ считываются укороченные записи с номером $j=i, i+N/K, \ldots, i+(M-1)N/K$, образуя там матрицу $M\times K$, которая затем подвергается транспонированию. При этом создаются K последовательных строк матрицы-результата, которые могут быть переписаны во внешнее запоминающее устройство.

Число операций считывания-записи, необходимых для этого алгоритма, пропорционально $M\times N/K$ [1]. Для обеспечения высокой эффективности необходимо стремиться к большим значениям K, следовательно, требуется большой объем памяти ОЗУ. Фактически этот метод сопоставим с другими предлагаемыми методами только в тех случаях, когда матрица может быть размещена в ОЗУ почти целиком. Аналогична эффективность и другого метода блочного транспонирования [2.3].

2.2.3. Транспонирование с использованием разбиений на квадраты

Второй алгоритм был впервые предложен Эклундом [2.6]. Он основан на последовательном транспонировании квадратных фрагментов и является естественным обобщением другого метода, описанного [2.7]. Он также сходен с методом, используемым в [2.8].

Пусть A — матрица $M\times N$ и $\bar{M}=m_1\cdot\ldots\cdot m_p\geqslant M$, где $m_i>1$. Положим $m_0=1$, $M_0=M$, $N_0=N$ и определим для $i=1, \ldots, p$

$$P_i=m_1\cdot\ldots\cdot m_i, \tag{2.11}$$

[1] Указанное автором выражение определяет только число операций считывания. Для записи матрицы-результата необходимо еще N операций записи. Таким образом, общее число операций ввода-вывода, необходимых для блочного алгоритма, определяется выражением $M\times N/K+N$. — *Прим. перев.*

$$M_i = \left\lceil \frac{M_{i-1}}{m_i} \right\rceil, \tag{2.12}$$

$$N_i = \left\lceil \frac{N_{i-1}}{m_i} \right\rceil, \tag{2.13}$$

где $\lceil\ \rceil$ определено следующим образом:

$$\lceil x \rceil = \begin{cases} x, & x \text{ — целое} \\ [x]+1 & \text{в противном случае,} \end{cases} \tag{2.14}$$

$[\ ]$ — обозначение целой части x. Тогда транспонирование матрицы A можно выполнить за p шагов следующим образом.

Шаг 1. Сформируем M_1 матриц размерами $m_1 \times N$ из последовательных строк A. Каждую такую матрицу можно разбить на N_1 матриц $m_1 \times m_1$, возможно, дополнив ее предварительно нулями [не более $m_1(m_1-1)$ нулей]. Эти квадратные матрицы можно теперь транспонировать на месте. В результате получим матрицу $A^{(1)}$, элементами которой будем считать подматрицы $1 \times P_1$ — фрагменты строк. При таком рассмотрении матрица $A^{(1)}$ имеет размеры $P_1 M_1 \times N_1$.

Определим индуктивно i-й шаг, считая первые i-1 шагов выполненными.

Шаг i. На шаге i начнем с матрицы $A^{(i-1)}$ размерами $P_{i-1} M_{i-1} \times N_{i-1}$. Элементами этой матрицы являются подматрицы — фрагменты строк размерами $1 \times P_{i-1}$. Затем из строк матрицы $A^{(i-1)}$ строим M_i подматриц $m_i \times N_{i-1}$. Каждая из этих подматриц формируется из тех строк матрицы $A^{(i-1)}$, которые находятся в ней на расстоянии P_{i-1} строк друг от друга. Выбор строк для каждой подматрицы выполняется следующим образом. Если обработано λP_i строк ($\lambda = 1, \ldots, M_i - 1$), то выбираются m_i строк, начиная с первой необработанной строки, т. е. строки $\lambda P_i + \mu$, $\mu = 1, 2, \ldots, P_{i-1}$, так, что каждая следующая выбираемая строка отстоит от предыдущей (уже выбранной) на P_{i-1} строк [1]. Следовательно, типичная подматрица состоит из строк, номера которых в матрице A^{i-1} определяются выражением $\lambda P_i + \mu + \nu P_{i-1}$, где $\nu = 0, 1, \ldots m_i - 1$, а значения λ и μ из указанных выше диапазонов зафиксированы для каждой подматрицы.

Сформированные подматрицы, при необходимости дополненные нулями, можно разбить на N_i матриц $m_i \times m_i$. Элементами этих квадратных матриц будем считать фрагменты строк размерами $1 \times P_{i-1}$. Матрицы транспонируются, и получается матрица $A^{(i)}$ с размерами $P_i M_i \times N_i$ с элементами-подматрицами $1 \times P_i$. Эти элементы являются фрагментами строк матрицы-результата $\tilde{A}$.

---

[1] Представленный здесь алгоритм основан на разбиении исходной матрицы на квадратные фрагменты. Транспонирование выполняется сначала внутри каждого фрагмента, а затем фрагменты переставляются должным образом. Перестановка фрагментов должна осуществляться построчно. Этот процесс и описан здесь. Некоторая сложность описания объясняется стремлением автора рассмотреть наиболее общий случай. — *Прим. перев.*

Приведенное описание алгоритма дает интуитивное ощущение того, каким образом формируются строки матрицы $\tilde{A}$ как части записей матрицы $A^{(p)}$. Это описание полезно также при анализе эффективности алгоритма. Однако из него не следует непосредственно, каким образом каждый элемент оказывается в конце концов на правильном месте. Поэтому опишем алгоритм по-другому, безотносительно к разбиениям, так, чтобы иметь возможность доказать работоспособность алгоритма и одновременно показать, как он может быть реализован.

Вначале рассмотрим случай, когда $M \geqslant N$. Пусть $[k^{(i)}, l^{(i)}]$ обозначает положение в матрице $A^{(i)}$ (не рассматриваемой больше как фрагментированная) элемента $a(k, l)$ матрицы A, $i=1, \ldots, p$; $k=0, 1, \ldots M-1$; $l=0, 1, \ldots N-1$. Отметим, что k и l можно единственным способом представить в виде

$$k = k_{p-1}P_{p-1} + \ldots + k_1P_1 + k_0, \quad l = l_{p-1}P_{p-1} + \ldots + l_1P_1 + l_0, \quad (2.15)$$

где $0 \leqslant k_i < m_{i+1}$; $0 \leqslant l_i < m_{i+1}$, а P_i определяется выражением (2.11). Представление (2.15) очевидным образом получается из главных остатков при последовательном делении на $m_1, m_2, \ldots, m_p$ (подобно тому, как при обычной системе счисления). Для упрощения будем использовать обозначения

$$k = n(k_{p-1}, \ldots, k_1, k_0), \quad l = n(l_{p-1}, \ldots, l_1, l_0). \quad (2.16)$$

Теперь шаг 1 означает, что матрица A разбита на подматрицы $m_1 \times m_1$, которые затем транспонируются. Следовательно,

$$k^{(1)} = n(k_{p-1}, \ldots, k_1, l_0), \quad l^{(1)} = n(l_{p-1}, \ldots, l_1, k_0), \quad (2.17)$$

поскольку $k_1, \ldots, k_{p-1}$ и $l_1, \ldots, l_{p-1}$ определяют, к какой именно подматрице $m_1 \times m_1$ принадлежит данный элемент.

По индукции предположим, что после i-1 шагов имеем

$$k^{(i-1)} = n(k_{p-1}, \ldots, k_i, k_{i-1}, l_{i-2}, \ldots, l_0),$$
$$l^{(i-1)} = n(l_{p-1}, \ldots, l_i, l_{i-1}, k_{i-2}, \ldots, k_0). \quad (2.18)$$

Шаг i можно теперь описать следующим образом. Будем считать, что матрица разбита на подматрицы размерами $P_i \times P_i$. Затем внутри каждой подматрицы сформируем матрицы $m_i \times m_i$, выбирая всевозможные комбинации из m_i строк и m_i столбцов, разделенных P_{i-1} строками (столбцами). Это означает, что $k_i, \ldots, k_{p-1}$ и $l_i, \ldots, l_{p-1}$ остаются неизменными, так как они определяют конкретную матрицу $P_i \times P_i$, с которой мы работаем. То же относится и к $l_{i-2}, \ldots, l_0$ и $k_{i-2}, \ldots, k_0$, поскольку эти значения определяют матрицу $m_i \times m_i$, к которой принадлежит элемент. В этой матрице элемент $a(k, l)$ перед транспонированием находится в строке k_{i-1} и столбце l_{i-1}. Поэтому после транспонирования его положение в матрице $A^{(i)}$ определяется:

$$k^{(i)} = n(k_{p-1}, \ldots, k_i, l_{i-1}\ l_{i-2}, \ldots, l_0),$$
$$l^{(i)} = n(l_{p-1}, \ldots, l_i, k_{i-1}, k_{i-2}, \ldots, k_0). \quad (2.19)$$

Следовательно, $[k^{(p)}, l^{(p)}] = (l, k)$. Это и доказывает, что $a(k, l)$ занимает правильное положение в $A^{(p)}$. Отсюда видно, что транспонирование выполняется как последовательный обмен разрядами в выражении (2.15), которое является представлением номеров строки и столбца в позиционно-численном представлении по смешанному основанию. В частности, если $m_i = 2$ для всех i, то получаем алгоритм, описанный в [2.7, 2.9].

Остается только установить соотношение между $A^{(p)}$ и $\tilde{A}$. Если $\bar{M} = M$, первые N строк $A^{(p)}$, очевидно, являются строками A, тогда как последние $\bar{M}-N$ строк содержат только нули. В сущности, никогда нет необходимости создавать эти строки. С другой стороны, если $\bar{M} > M$, то строки матрицы $A^{(p)}$ также содержат $\bar{M}-M$ нулей в конце каждой строки. Эти нулевые части строк, как и нулевые строки, никогда не следует записывать в выходную матрицу.

Случай $M < N$ может быть рассмотрен аналогично, если в (2.15) записать

$$l = l_p P_p + l_{p-1} P_{p-1} + \ldots + l_0 \tag{2.20}$$

при $0 \leqslant l_i < m_{i+1}$ для $i = 0, 1, \ldots, p-1$ и $0 < l_p$. Конечное положение $a(k, l)$ теперь будет $[k^{(p)}, l^{(p)}]$, где

$$k^{(p)} = l_{p-1} P_{p-1} + \ldots + l_0, \tag{2.21}$$

$$l^{(p)} = l_p P_p + k_{p-1} P_{p-1} + \ldots + k_0. \tag{2.22}$$

В этом случае строки матрицы $A^{(p)}$ содержат одну ($\bar{M} \geqslant N$) или несколько ($\bar{M} < N$) строк матрицы $\tilde{A}$. Записывая $A^{(p)}$ как расчлененную на матрицы $N_p \bar{M} \times \bar{M}$, где N_p определяется (2.13),

$$A^{(p)} = (B_1 | B_2 | \ldots B_{N_p}), \tag{2.23}$$

получим $\tilde{A}$ как

$$\left(\begin{array}{c} B_1 \\ \hline B_2 \\ \hline \vdots \\ \hline B_{N_p} \end{array}\right) = \left(\begin{array}{c|c} \tilde{A} & 0 \\ \hline 0 & 0 \end{array}\right). \tag{2.24}$$

Следует заметить, что эти операции, так же как и отбрасывание нулей, выполняются при формировании выходных записей на шаге p без дополнительной обработки данных.

Если $M \neq N$, то, по крайней мере, для некоторых произведений $\bar{M}$ алгоритм сначала будет транспонировать матрицы $K \times K$, где $K = \min(M, N)$, а затем отсекать ($M < N$) или расщеплять ($M > N$) строки.

Если строки матриц $A^{(i)}$ создаются при выводе на внешнее ЗУ, эффективность алгоритма по требуемому числу ячеек ОЗУ на i-м шаге характеризуется

$$\mathrm{RM}_s = \max_{1 \leqslant i \leqslant p} \{m_i N_{i-1} P_{i-1}\}. \tag{2.25}$$

M дополнительных ячеек памяти позволили бы действительно сформировать строки матрицы A в ОЗУ, что представляет интерес при выполнении двумерных преобразований. Однако это легко можно сделать посредством транспонирования в основную память, не используя дополнительные ячейки, если $m_p \leqslant N_p$, или применяя $m_p N_p$ дополнительных ячеек, если $m_p > N_p$ (см. разд. 2.7). Оказывается, что для оптимального разбиения обычно $m_p N_p \ll M$ [1].

Можно заметить, что транспонирование в основной памяти на каждом шаге, вместо формирования выходной записи в заданном порядке, оказывается не только более медленным, но и требует большего объема памяти. Точнее говоря, имеем

$$\mathrm{RM}_{s'} = \max_{1 \leqslant i < p} \{m_i N_i P_i\}. \tag{2.26}$$

В дальнейшем покажем (см. подраздел 2.3.1), что значение $N_i P_i$ увеличивается с ростом i, откуда и следует, что $\mathrm{RM}_{s'} \geqslant \mathrm{RM}_s$.

Если строки матрицы $A^{(p)}$, состоящие из нулей, не записываются, то необходимое число операций ввода-вывода будет определяться выражением:

$$\mathrm{IO}_s = M + 2 \sum_{i=1}^{p-1} M_i P_i + N. \tag{2.27}$$

Если $\bar{M} = M$, то представленный здесь алгоритм обладает тем свойством, что все матрицы $A = A^{(0)}, A^{(1)}, \ldots, A^{(p-1)}$ содержат по M строк, в то время как $A^{(p)}$ содержит N строк, если нули в ней отброшены. Отсюда вытекает, что число операций ввода-вывода зависит только от числа множителей M. Если, как упоминалось ранее, пренебречь изменением (увеличением) длины строк, то можно оптимизировать алгоритм для заданных M и p, минимизируя требуемое число ячеек ОЗУ. Далее будет показано, как при некоторых условиях можно определить оптимальную в этом смысле факторизацию M. Если же эти условия не выполняются, оптимум можно легко найти в каждом конкретном случае. Кроме того, будет продемонстрирована зависимость решения от p. Если $\bar{M} > M$, то факторизация, минимизирующая требования к памяти, не будет минимизировать требуемое число операций ввода-вывода, однако число операций ввода-вывода для разных факторизаций с одним и тем же числом сомножителей меняется незначительно. В связи с этим можно предложить простой алгоритм, который итеративно с увеличением $\bar{M}$ быстро находит факторизацию, минимизирующую требования к памяти для каждого p. Поскольку $p \leqslant [\log_2 M]$, легко определить, какое из этих решений является самым дешевым с точки зрения объема памяти и операций ввода-вывода. В общем, таким способом можно определить стратегию, близкую к глобальному оптимуму. Этот подход будет описан в разд. 2.4.

---

[1] Это доказано в разд. 2.3. — *Прим. перев.*

2.2.4. Алгоритм Флойда

Алгоритм разбиения на квадраты обладает таким свойством, что если все сомножители M равны 2 или степеням 2, то он может быть реализован с помощью операций сдвига и маскирования. Это следует из описания, приведенного в подразд. 2.2.3 (см. также [2.7]). Другой алгоритм, предложенный Флойдом, также обладает этим свойством. Он совпадает с алгоритмом разбиения на квадраты, если $M=N=2^m$ и может быть применен к любой квадратной матрице непосредственно (без дополнения нулями).

Матрица $M\times M$ может быть транспонирована за $p=\log_2 M$ проходов, используя $2M$ ячеек основной памяти, следующим образом:

1) формируем матрицу $A^{(1)}$, у которой $a^{(1)}(k, l)=a(-k, l)$;
2) формируем матрицу $B^{(0)}$, у которой $b^{(0)}(k, k+l)=a^{(1)}(k, l)$;
3) рекурсивно, для $i=1, \ldots, p$ формируем $B^{(i)}$ из $B^{(i-1)}$, сдвигая столбцы, в номерах которых i-й двоичный разряд [см. l_{i-1} в (2.15)] равен 1, на 2^i мест;
4) формируем матрицу $\tilde{A}$ из $B^{(p)}$, полагая $\tilde{a}(k, l)=b^{(p)}(k, k+l)$.

Заметим, что ни $A^{(1)}$, ни $B^{(0)}$ или $B^{(p)}$ на самом деле создавать не нужно. Доказательство алгоритма приведено в [2.10].

Используя, скажем, $2^m M$ ячеек ОЗУ, можно выполнить m шагов алгоритма за один проход, как в алгоритме разбиения на квадраты. Однако этот алгоритм затем требует некоторой дополнительной «бухгалтерии», так как циклический сдвиг элементов охватывает *все* строки и не может быть выполнен поэтому внутри данной части из 2^m строк.

Таким образом, видим, что если используются

$$\mathrm{RM_F}=2^m M \tag{2.28}$$

ячеек памяти, число операций ввода-вывода

$$\mathrm{IO_F}=2M\lceil\log_2 M/m\rceil. \tag{2.29}$$

В частности, в [2.10] показано, что если M является степенью 2, то правая часть (2.29) является также нижней границей числа операций ввода-вывода, необходимых для транспонирования матрицы $M\times M$ при данном объеме памяти. Следовательно, этот алгоритм, так же как алгоритм разбиения на квадраты и алгоритм, представленный в следующем разделе, минимизирует IO для данного значения RM. Можно отметить, что ту же самую нижнюю границу получил Стоун [2.11], который также разработал алгоритм, достигающий этой границы. Однако его алгоритм не применим непосредственно к решению этой задачи и не может быть реализован с помощью только операций сдвига и маскирования.

Равенство (2.29) дает только верхнюю границу требуемого числа операций ввода-вывода. В этом отношении оптимальный алгоритм фактически может быть много лучше. Например, опти-

мальный алгоритм для матрицы 5×5 с $m=1$ даст IO=28 по сравнению с верхней границей, равной 30 [2.10]. Не существует, однако, алгоритма, который обеспечивал бы минимальное значение IO для *всех* квадратных матриц при заданном значении RM. Алгоритм Флойда в том виде, как он описан здесь, в общем случае требует затраты большего числа операций ввода-вывода, чем другие алгоритмы, которые здесь рассматриваются. Однако эти алгоритмы обычно требуют несколько большего объема оперативного или внешнего ЗУ. Пример будет приведен в разд. 2.5.

Алгоритм Флойда можно очевидным образом обобщить на случай прямоугольной матрицы, используя подход, описанный в подразд. 2.2.3.

2.2.5. Транспонирование методом «ввод строки-вывод столбца»

Этот алгоритм является обобщением предложенного Кнутом [2.12] алгоритма транспонирования матриц, хранящихся в ЗУ последовательного типа. В первоначальном виде алгоритм транспонирует матрицу $M\times M$, хранящуюся последовательно (в форме одна запись — одна строка) за $\lceil\log_2 M\rceil$ проходов, используя $2^{\lceil\log_2 M\rceil+1}$ ячеек ОЗУ и четыре дополнительных последовательных файла. Рассмотрим работу алгоритма. На каждом шаге создаются два файла.

Шаг 1.

Файл 1: $a_{11}a_{21}a_{12}a_{22}\ldots a_{1,\,M/2}a_{2,\,M/2}\ldots a_{1M}a_{2M}a_{51}a_{61}a_{52}a_{62}\ldots$
$\ldots a_{M-3,M}a_{M-2,\,M}$.

Файл 2: $a_{31}a_{41}a_{32}a_{42}\ldots a_{3,\,M/2}a_{4,\,M/2}\ldots a_{3M}a_{4M}a_{71}a_{81}a_{72}a_{82}\ldots$
$a_{M-1,\,M}a_{M,M}$.

Шаг 2.

Файл 3: $a_{11}a_{21}a_{31}a_{41}a_{12}a_{22}a_{32}a_{42}\ldots a_{1,\,M/4}a_{2,\,M/4}a_{3,\,M/4}\,a_{4,\,M/4}\ldots$
$\ldots a_{4,\,M}a_{91}\ldots a_{M-5,\,M}$.

Файл 4: $a_{51}a_{61}a_{71}a_{81}a_{52}a_{62}a_{72}a_{82}\ldots a_{5,\,M/4}a_{6,\,M/4}a_{7,\,M/4}a_{8,\,M/4}\ldots$
$\ldots a_{8,\,M}a_{13,\,1}\ldots a_{M,\,M}$.

Каждые два файла на следующем шаге обрабатываются синхронно посредством считывания одной записи из каждого входного файла и последующей записи данных в выходные файлы в новом порядке, по две записи в каждый файл попеременно. Очевидно, транспонирование будет получено после $\lceil\log_2 M\rceil$ проходов или после $M\lceil\log_2 M\rceil$ операций считывания и записи.

В процессе работы возникает необходимость в незначительном изменении размеров записей. Фактически требуется увеличивать некоторые записи таким образом, чтобы на i-м шаге, $i=1,\ldots,$ $\lceil\log_2 M\rceil-1$, они содержали по $k2^i$ элементов, где k — четное число. Это означает, что некоторые входные записи на последнем шаге содержат $2^{\lceil\log_2 M\rceil}$ элементов.

Тщательное рассмотрение алгоритма показывает, что i-й шаг в действительности соответствует последовательности полных пе-

рестановок [2.11] пар массивов, элементами которых являются подмассивы из 2^{i-1} элементов. В этом смысле алгоритм Кнута, конечно, подобен алгоритму Стоуна [2.11], но перестановки выполняются при выводе на внешнее ЗУ и переупорядоченные массивы на каждом шаге i не формируются в ОЗУ на этом шаге. Такое обстоятельство наводит на мысль о небольшой модификации алгоритма, при которой на последнем шаге транспонируются матрицы 2×2, элементами которых являются матрицы-столбцы из $2^{\lceil \log_2 M\rceil - 1} \times 1$ элементов. Таким образом, получаем строки транспонированной матрицы в ОЗУ *перед* записью их в выходной файл. Это свойство полезно, если алгоритм применяется в сочетании, например, с вычислением БПФ.

Рассмотренный алгоритм может быть естественным образом обобщен на случай прямоугольных матриц с помощью метода, описанного в подразд. 2.2.3. Он эффективен для транспонирования матриц, хранящихся в ЗУ последовательного типа. Например, он, вообще говоря, более эффективен, чем подход, предложенный в [2.14, 2.15] (доказательство см. в [2.6]).

Пусть матрица A хранится в ЗУ с произвольным доступом. Тогда алгоритм имеет обобщение, которое работает в точности аналогично алгоритму разбиения на квадраты, за исключением только одного аспекта. Имея матрицу $A^{(i-1)}$ размерами $1\times P_{i-1}$, которая построена из фрагментов строк матрицы $\tilde{A}$, формируем строки $A^{(i)}$, записывая m_i последовательных столбцов подматриц в оперативную память. Тогда строки матриц $A^{(i)}$ будут состоять из частей строк матрицы $\tilde{A}$ размера P_i. Обычно нет необходимости дополнять строки на выходе фиктивными данными для выравнивания размеров матрицы.

Отметим, что для двоичного случая этот метод не идентичен методу разбиения на квадраты. Однако они равноценны в отношении требований к ресурсам, т. е.

$$\mathrm{RM_{RC}} = \mathrm{RM_S} = \max_{1\leqslant i\leqslant p} \{m_i N_{i-1} P_{i-1}\}, \tag{2.30}$$

$$\mathrm{IO_{RC}} = \mathrm{IO_S} = M + 2\sum_{i=1}^{p-1} M_i P_i + N. \tag{2.31}$$

Имеющиеся отличия иногда могут оказаться важными. Так, перестановки на месте не столь просты, как транспонирование квадратных матриц. Но обычно это нас не интересует до последнего шага, и поэтому можно считать, что оба метода будут работать одинаково (см. примеры в разд. 2.5). Более важны особенности, которые касаются требований к внешней памяти. Применительно к квадратным матрицам, хранящимся в файлах с произвольным доступом, оба алгоритма могут работать с оставлением на месте с одним единственным файлом. Это справедливо также для метода прямоугольных разбиений, приведенного ниже, который можно заставить работать с квадратами ($m_i = n_i$). Для неквадратных матриц эти три метода уже не подобны друг другу.

Первоначальный выходной файл может всегда быть последовательным и неизменным. Независимо от этого метод разбиения на квадраты может работать с оставлением на месте вплоть до шага p, на котором матрица $\tilde{A}$ запоминается во второй области произвольного доступа. Метод разбиения на прямоугольники требует двух областей произвольного доступа для размещения $A^{(2i)}$ и $A^{(2i+1)}$, $i=0, 1, \ldots$. Кроме того, размеры файлов будут меняться. Наконец, метод «ввод строки-вывод столбца» может быть реализован подобно методу разбиения на квадраты. Если, однако, на последнем шаге порядок, в котором обрабатываются группы строк $A^{(p-1)}$, изменяется (см. пример в разд. 2.6), строки $\tilde{A}$ могут формироваться последовательно. К тому же метод работает с последовательными файлами, если число дополнительных файлов составляет $\max\limits_{1\leqslant i\leqslant p}\{m_i\}$.

Конечно, существуют также прямоугольные варианты приведенного в этом разделе алгоритма. Мы советуем читателю самому разобраться в этих деталях.

2.2.6. Алгоритм прямоугольных разбиений

Последний алгоритм, который мы рассмотрим, впервые был предложен в [2.13]. Этот алгоритм состоит в последовательном транспонировании подматриц порядка $m_i \times n_i$, где m_i и n_i — множители M и N соответственно. Позже Рамапрайян [2.8] обобщил алгоритмы на случай, когда $\bar{M}=m_1 \cdot \ldots \cdot m_p \geqslant M$ и $\bar{N}=n_1 \cdot \ldots \cdot n_p \geqslant N$, где p, $m_1, \ldots, m_p$ и $n_1, \ldots, n$ определяются численной оптимизацией требуемого времени вычислений.

Алгоритм работает следующим образом. Пусть A — матрица $M \times N$, и предположим, что $\bar{M}=m_1 \cdot \ldots \cdot m_p \geqslant M$ и $\bar{N}=n_1 \cdot \ldots \cdot n_p \geqslant N$, где p — произвольное целое число, $p \geqslant 2$. Положим $m_0=1$, $n_0=1$, $M_0=M$, $N_0=N$ и определим набор целых чисел для $i=0, 1, \ldots, p$:

$$P_i=m_0 \cdot m_1 \cdot \ldots \cdot m_i; \quad Q_i=n_0 \cdot n_1 \cdot \ldots \cdot n_i;$$
$$M_i=\lceil M_{i-1}/m_i \rceil; \quad N_i=\lceil N_{i-1}/n_i \rceil. \tag{2.32}$$

Тогда на i-м шаге, $i=1, 2, \ldots, p$, входная матрица $A^{(i-1)}$, где $A^{(0)}=A$, рассматривается как матрица $M_{i-1} \times N_{i-1}$, элементами которой являются матрицы $Q_{i-1} \times P_{i-1}$, представляющие собой фактически подматрицы $\tilde{A}$. Эта секционированная матрица рассматривается как матрица $M_i \times N_i$, из которой селективно считываются в оперативную память матрицы $m_i \times n_i$, где они транспонируются и записываются в $A^{(i)}$. Строки в считываемые матрицы выбираются как в [2.13], т. е. с шагом Q_{i-1}. Таким образом, матрица $A^{(i)}$ будет построена из блоков размерами $Q_i \times P_i$ и на шаге p будет сформирована матрица $\bar{N} \times \bar{M}$, содержащая $\tilde{A}$.

Можно посмотреть на $A^{(i-1)}$ и по-другому, считая ее построенной из фрагментов $1 \times P_{i-1}$, так как подматрицы $Q_{i-1} \times P_{i-1}$ на самом деле никогда не создаются. Тогда строки $A^{(i)}$ будут содержать фрагменты $1 \times P_i$, которые образуют части строк матрицы $\tilde{A}$.

Если $\bar{M}=M$ и $\bar{N}=N$, алгоритм совпадает с алгоритмом, приведенным в [2.13]. В противном случае матрица $A^{(i-1)}$ на i-м шаге превращается в минимальную для данных $m_i, \ldots, m_p$ и $n_1, \ldots, n_p$ матрицу, с которой работает алгоритм [2.13].

Требуемый объем памяти определяется выражением

$$\mathrm{RM_R} = \max_{1 \leqslant i \leqslant p} \{m_i N_{i-1} P_{i-1} + P_i N_i\}, \tag{2.33}$$

где при $i \neq p$ второе слагаемое может быть опущено.

Число операций ввода-вывода

$$\mathrm{IO_R} = M + 2\sum_{i=1}^{p-1} Q_i M_i + N. \tag{2.34}$$

2.3. Оптимизация эффективности алгоритма

2.3.1. Две леммы

Все представленные в предыдущем разделе алгоритмы, за исключением двоично-ориентированного алгоритма Флойда, включают преобразование исходной матрицы в матрицу больших размеров. Обсудим теперь, каким образом выбирать это преобразование для оптимизации эффективности алгоритмов. Прежде всего нам нужны две основополагающие леммы.

Пусть $\bar{M}$ и N — произвольные целые числа, $\bar{M}>1$, $N>1$, и предположим, что для некоторого $p>1$ $\bar{M}=m_1, \ldots, m_p$, где $m_i>1$, $i=1, \ldots, p$. Определим, как и прежде, $m_0=1$, $p_i=m_0 \cdot \ldots \cdot m_i$, $i=0, 1, \ldots, p$ и $N_0=N$, $N_i=\lceil N_{i-1}/m_i \rceil$, $i=1, \ldots, p$. Тогда имеем следующее.

Лемма 1. Последовательность $(N_i P_i)_{i=0}^{p}$ является возрастающей.

Доказательство. По определению, $N_i=\lceil N_{i-1}/m_i \rceil \geqslant N_{i-1}/m_i$, т. е. $N_i m_i \geqslant N_{i-1}$. Следовательно, умножение на P_{i-1} дает $N_i P_i \geqslant N_{i-1} P_{i-1}$.

Эта лемма просто устанавливает, что в алгоритме разбиения на квадраты длина строк увеличивается.

Лемма 2. $N/P_i \leqslant N_i < N/P_i + 1$, $i=1, \ldots, p$.

Доказательство. Используем индукцию по числу сомножителей P_i. Имеем

$$N_1 = \begin{cases} N/m_1, & \text{если } m_1 \text{ является делителем } N; \\ \lceil N/m_1 \rceil + 1 & \text{в противном случае.} \end{cases}$$

В первом случае справедливость леммы очевидна, во втором — надо только заметить, что $\lceil N/m_1 \rceil < N/m_1$.

Предположим, что неравенство справедливо для $i-1$ сомножителей, т. е.

$$N/P_{i-1} \leqslant N_{i-1} < N/P_{i-1} + 1. \tag{2.35}$$

Запишем $N_{i-1}=q_i m_i+r_i$, где $0\leqslant r_i<m_i$. Тогда $q_i=[N_{i-1}/m_i]$ и

$$N_i=\begin{cases}q_i, & \text{если } r_i=0;\\ q_i+1, & \text{если } 0<r_i<m_i.\end{cases} \tag{2.36}$$

Используя левое неравенство из (2.35), находим

$$N_i\geqslant(N_{i-1}/m_i)\geqslant(N/P_{i-1})/m_i=N/P_i. \tag{2.37}$$

С другой стороны, если $r_i=0$, то

$$N_i=N_{i-1}/m_i<(N/P_{i-1}+1)/m_i=N/P_i+1/m_i<N/P_i+1. \tag{2.38}$$

Далее, если $1\leqslant r_i<m_i$, можно записать

$$N_i=q_i+1=N_{i-1}/m_i+(m_i-r_i)/m_i<(N_{i-1}/P_{i-1}+1)/m_i+$$
$$+(m_i-r_i)/m_i=N/P_i+[m_i-(r_i-1)]/m_i\leqslant N/P_i+1\ \blacklozenge \tag{2.39}$$

Отсюда непосредственно вытекает следующее.

Следствие. Так как N_i — целое, N_i и N_iP_i инвариантны относительно изменения факторизации $P_i=m_1\cdot\ldots\cdot m_i$. В частности, $m_1, \ldots, m_i$ можно переставить.

Введем несколько новых, более общих обозначений. Будем обозначать конечные массивы положительных целых чисел через $[\alpha]$, $[\beta]$, ... или через перечисление их элементов $(m_1, m_2, \ldots, m_p)$. Для данного массива $[\alpha]=(m_1, m_2, \ldots, m_p)$ запишем подмассивы $[\alpha_j]=(m_1, \ldots, m_j)$ и положим $N_{[\alpha_0]}=N, \ldots, N_{[\alpha_j]}=[N_{[\alpha_{j-1}]}/m_j]$ и $P_{[\alpha_0]}=1$, $P_{[\alpha_j]}=m_1\cdot\ldots\cdot m_j$, $j=1, \ldots, p$.

2.3.2. Алгоритм разбиения на квадраты

Положим $R_{[\alpha_j]}=m_jN_{[\alpha_{j-1}]}P_{[\alpha_{j-1}]}$ и $R[\alpha]=\max\limits_{1\leqslant j\leqslant p}\{R_{[\alpha_j]}\}$. Тогда имеем следующее.

Теорема 1. Пусть $\bar{M}$ и N — заданные целые числа, $\bar{M}>1$, $N>1$, и предположим $\bar{M}=m_1\cdot\ldots\cdot m_p$, где $m_1\geqslant m_2\geqslant\ldots\geqslant m_p>1$. Пусть $[\alpha]=(m_1, \ldots, m_p)$, и предположим, что $\pi[\alpha]$ — произвольная перестановка $[\alpha]$. Тогда $R[\alpha]\leqslant R[\pi[\alpha]]$.

Перед тем как доказывать теорему, сделаем несколько замечаний. Вернувшись к выражениям (2.25), (2.27), обнаружим, что в соответствии с утверждением теоремы алгоритм разбиений на квадраты при заданной факторизации требует минимального объема памяти, если сомножители упорядочены по их убыванию. Тогда получим, что строки матрицы $\bar{A}$ никогда не формируются (как последовательные массивы) в ОЗУ. Но это требует выделения на шаге p не более m_pN_p дополнительных ячеек памяти (см. [2.6]), и мы увидим, что и в этом случае упорядочение сомножителей по убыванию обеспечивает минимум. Теорема также справедлива, если $R'_{[\alpha_j]}=m_jN_jP_{[\alpha_j]}$, что получается, если на каждом шаге формируются строки; в этом виде ее можно доказать подобным образом.

То, что действительно существует зависимость от порядка сомножителей, доказывают следующие примеры.

Примеры. Если $N=22$ и $\bar{M}=5\cdot3=15$, имеем $R_{(5)}=110$ и $R_{(5,3)}=75$, в то время как $R_{(3)}=66$ и $R_{(3,5)}=120$, т. е. $R(3,5)=120>110=R(5,3)$. Задавая $\bar{M}=6\cdot5=30$, видим, что $R(6,5)=132$, в то время как $R(5,6)=150$, т. е. это свойство не зависит от того, меньше $\bar{M}$ чем N или нет. В этих двух примерах максимум достигается для максимального сомножителя. Это не всегда так при $\bar{M}>N$. Например, если $N=22$ и $\bar{M}=7\cdot6$, то $R_{(7)}=154$ и $R_{(7,6)}=168$.

Однако, если $\bar{M}\leqslant N$, то $R_{\max}$ всегда достигается для последнего по порядку максимального сомножителя (см. ниже). Это утверждение, однако, неверно для массива $m_jN_jP_j$, $j=1,\ldots,p$, получаемого, когда обмен выполняется на каждом шаге.

Доказательство. Предположим, что m_i — сомножитель, для которого достигается максимальное значение $R[\alpha]$. Пусть $\pi[\alpha]$ — произвольная перестановка $[\alpha]$, и рассмотрим самый короткий массив в начале $\pi[\alpha]$, содержащий все элементы $m_1,\ldots,m_i$. Обозначим его $[\bar{\beta}]$. Если $[\bar{\beta}]$ содержит j элементов, то $i\leqslant j\leqslant p$, и его последним элементом является m_k, где $1\leqslant k\leqslant i$. Пусть $[\beta]$ — подмассив, полученный из $[\bar{\beta}]$ удалением m_k. Теперь переупорядочим $[\bar{\beta}]$ в $[\bar{\gamma}]$ следующим образом. Если $k=i$, положим $[\bar{\gamma}]=[\bar{\beta}]$, но если $k<i$, массив $[\bar{\gamma}]$ получается из $[\bar{\beta}]$ переменой мест m_i и m_k. Пусть, как и прежде, $[\gamma]$ — подмассив в начале $[\bar{\gamma}]$, содержащий $j-1$ элементов.

Теперь можно увидеть, что, так как $P_{[\gamma]}\geqslant P_{[\beta]}$, из леммы 2 и ее следствия вытекает, что $N_{[\gamma]}\leqslant N_{[\beta]}$ (заметим, что лемма 2 ограничивает N_i интервалом, содержащим точно одно целое число). Кроме того, $m_iP_{[\gamma]}=m_kP_{[\beta]}$, так что $m_kP_{[\beta]}N_{[\beta]}\geqslant m_iP_{[\gamma]}N_{[\gamma]}$.

Пусть $[\delta]$ — массив, получаемый из $[\gamma]$ помещением $m_1,\ldots,m_{i-1}$ в начало. Тогда снова, используя следствие леммы 2, получаем $P_{[\delta]}N_{[\delta]}=P_{[\gamma]}N_{[\gamma]}$. Применяя лемму 1 к $[\delta]$, имеем

$$R[\pi[\alpha]]\geqslant m_kP_{[\beta]}N_{[\beta]}\geqslant m_iP_{[\gamma]}N_{[\gamma]}=m_iP_{[\delta]}N_{[\delta]}\geqslant$$
$$\geqslant m_iP_{i-1}N_{i-1}=R[\alpha]\ \blacklozenge$$

Эта теорема не учитывает необходимость в дополнительных ячейках памяти для формирования строк матрицы $\tilde{A}$ на шаге p. Для этого достаточно выделить на p-м шаге m_pN_p дополнительных слов (см. [2.6]). Тогда имеем:

Следствие. Теорема 1 справедлива также для массива $(T_i)^p{}_{i=1}$, $T_i=m_iP_{i-1}N_{i-1}$, $i=1,\ldots,p-1$; $T_p=m_pP_{p-1}N_{p-1}+m_pN_p$.

Доказательство. Пусть $m'_1,\ldots,m'_p$ — переупорядоченная последовательность $m_1,\ldots,m_p$, и обозначим через $(T'_i)^p{}_{i-1}$ соответствующий массив, для которого отыскивается максимум. Предположим также, что $\max T'_i=T'_j$. Тогда можно увидеть, что в соответствии с предположениями $m'_p\geqslant m_p$, так что $P'_{p-1}\leqslant P_{p-1}$ и, следовательно, $N'_{p-1}\geqslant N_{p-1}$. Итак, поскольку $P'_p=P_p=\bar{M}$ и $N'_p=N_p$, получаем $T'_p=P'_pN'_{p-1}+m'_pN'_p\geqslant P_pN_{p-1}+m_pN_p=T_p$. Этим и доказывается следствие, так как согласно теореме

$$\max_{1\leqslant i\leqslant p-1}T_i\leqslant\max_{1\leqslant i\leqslant p}T'_i\ \blacklozenge$$

Теперь докажем тот, впрочем, достаточно очевидный факт, что наилучшая факторизация на $p+1$ сомножитель требует меньшего объема оперативной памяти, чем произвольная факторизация на p сомножителей. Определим для данного $\overline{M}$

$$\Omega_p(\overline{M}) = \{[\alpha] = (m_1, \ldots, m_p);\quad m_i > 1,\quad m_1 \cdot \ldots \cdot m_p = \overline{M}\} \tag{2.40}$$

и положим

$$S_p(\overline{M}) = \min\{R_{[\alpha]};\quad [\alpha] \in \Omega_p(\overline{M})\}. \tag{2.41}$$

Теорема 2. Если $\overline{M}$ можно представить в виде произведения $p+1$ нетривиальных сомножителей, то $S_{p+1}(\overline{M}) \leqslant S_p(\overline{M})$.

Доказательство. Пусть $[\alpha] = (m_1, \ldots, m_p)$ — оптимальная факторизация $\overline{M}$, дающая $S_p(\overline{M})$. В соответствии с предположением, по крайней мере один сомножитель, например, m_j, может быть записан как $m_j = st$, где $s > 1$, $t > 1$, s и t — целые. Положим $[\beta] = (m_1, \ldots, m_{j-1}, s, t, m_{j+1}, \ldots, m_p)$, $[\sigma] = (m_1, \ldots, m_{j-1}, s)$ и $[\tau] = (m_1, \ldots, m_{j-1}, s, t)$. Из следствия леммы 2 известно, что $N_{[\tau]} = N_j$, а кроме того, из этой же леммы вытекает, что $N_{[\sigma]} \leqslant N_{j-1}$. Так как $sP_{j-1} < P_j$, получим

$$S_{p+1}(\overline{M}) \leqslant R_{[\beta]} = \max\{P_1 N_0, \ldots, P_{j-1} N_{j-2}\cdot$$
$$s P_{j-1} N_{j-1},\quad P_j N_{[\sigma]},\quad P_{j+1} N_j, \ldots,\ P_p N_{p-1}\} \leqslant$$
$$\leqslant \max\{P_1 N_0, \ldots, P_{j-1} N_{j-2},\ P_j N_{j-1},\ P_{j+1} N_j, \ldots,\ P_p N_{p-1}\} = S_p(\overline{M}) \blacklozenge \tag{2.42}$$

Теорема остается справедливой и в случае, когда требования к памяти на шаге i определяются значением $m_i P_i N_i$, но доказательство ее в этом случае несколько сложнее.

Теперь нам нужна теорема, характеризующая оптимальное разложение на p сомножителей. Один результат в этом направлении дает следующая теорема.

Введем обозначение

$$r_p(\overline{M}) = \min\{\|[\alpha]\|_\infty;\quad [\alpha] \in \Omega_p(\overline{M})\}, \tag{2.43}$$

где $\|(m_1, \ldots, m_p)\|_\infty = \max\limits_{1 \leqslant i \leqslant p}\{m_i\}$.

Теорема 3. Предположим, что $\overline{M} \leqslant N$. Тогда $R[\alpha] = S_p(\overline{M})$, только если $\|[\alpha]\|_\infty = r_p(\overline{M})$. Фактически, если в $\Omega_p(\overline{M})$ имеется несколько элементов с минимальной нормой $r_p(\overline{M})$, то величину $S_p(\overline{M})$ будут минимизировать те из них, которые встречаются реже всех.

Доказательство. В соответствии с теоремой 1 мы должны рассматривать только массивы, расположенные по убыванию. Пусть $[\alpha] = (m_1, \ldots, m_p) \in \Omega_p$. Тогда, так как $\overline{M} \leqslant N$, лемма 2 дает

$$m_i P_{i-1} N_{i-1} < P_i(N/P_{i-1} + 1) = m_i N + P_i \leqslant (m_i + 1) N. \tag{2.44}$$

Отсюда вытекает, что в точке максимума должно быть $m_i=m_1$. Если $[\alpha]$ и $[\alpha']$ — два убывающих массива в $\Omega_p(\bar{M})$, такие, что $r_p(\bar{M})=m_1<m'_1$, то, очевидно,

$$m_i P_{i-1} N_{i-1} < (m_1+1)N \leqslant m'_1 N \leqslant R[\alpha'], \qquad (2.45)$$

так что $R[\alpha]<R[\alpha']$. Последняя часть теоремы вытекает из (2.25) и леммы 1, если используется расположение по убыванию ◆

Из этой теоремы следует, что требования к памяти (для фиксированного p) являются минимальными, когда значения сомножителей изменяются как можно меньше. Однако это не справедливо при $\bar{M}>N$. Если, например, $\bar{M}=9\cdot9\cdot4\cdot4=6\cdot6\cdot6\cdot6$ и $N=243$, то $R(9,9,4,4)=2187$, тогда как $R(6,6,6,6)=2592$. Тем не менее, как будет показано в разд. 2.4, теорему можно использовать для облегчения поиска оптимального разбиения. Теорема не справедлива, если рассматривать l^1-нормы или массив $m_iP_iN_i$ (см. [2.6]). Можно также отметить, что даже если $M<N$, лучшая факторизация $\bar{M}$ могла бы удовлетворять неравенству $\bar{M}>N$.

К сожалению, требуемое число операций ввода-вывода [см. (2.27)] не будет минимально, если сомножители упорядочены в убывающую последовательность. Если, например, $M=52$, то $\bar{M}=5\cdot4\cdot3$, $3\cdot4\cdot5$ и $4\cdot5\cdot3$ дают $IO_s=52+230+N$, $52+228+N$ и $52+224+N$ соответственно. Если $\bar{M}=M$, то $IO_s=(2p+1)M+N$ независимо от порядка сомножителей. Вообще, из леммы 2 известно, что $\bar{M}\leqslant M_iP_i\leqslant\bar{M}+P_i-1$, т. е.

$$M+N+2p\,\bar{M} \leqslant IO_s^{(p)} \leqslant M+N+2p\,\bar{M}+2\sum_{i=1}^{p-1}(P_i-1). \qquad (2.46)$$

Необходимое дополнительное число операций ввода-вывода, вызванное увеличением M до $\bar{M}$, оказывается при $m_i>1$ не больше, чем

$$2\sum_{i=1}^{p-1}(P_i-1)=2\left[\frac{\bar{M}}{m_p}\left(\frac{1}{m_2\cdot\ldots\cdot m_{p-1}}+\ldots+\frac{1}{m_{p-1}}+1\right)-(p-1)\right]<$$
$$<2\left[2\,\frac{\bar{M}}{m_p}-(p-1)\right]. \qquad (2.47)$$

Это соответствует менее чем $2\bar{M}/m_p$ просмотрам данных.

Используя тот факт, что, начиная с шага 2, некоторые строки содержат фиктивные элементы, можно, конечно, дополнительно снизить это значение. Такая процедура, однако, усложнит реализацию алгоритма.

Заслуживает упоминания частный случай, когда $\bar{M}$ и N могут быть представлены в виде $\bar{M}=m_1\cdot\ldots\cdot m_p$, $N=m_1\cdot\ldots\cdot m_{p-1}\cdot k$. Такое разбиение даст в результате число операций ввода-вывода, точно равное $2(p-1)\bar{M}+N$, независимо от значения сомножителей. Если $\geqslant m_p$, т. е. если $N\geqslant M$, не требуется дополнительного объема памяти на шаге p, даже если формируются строки матрицы $\bar{A}$. Из теоремы 3 следует, что такое упорядочение требует оптимального числа ячеек памяти, равного $\max\limits_{1\leqslant i\leqslant p} m_iN$. В то же время необходимое число операций ввода-вывода будет меньше, чем требу-

ется при убывающей последовательности. Последнее утверждение еще остается в силе для случая $k<m_p$, но необходимый объем памяти слегка возрастает.

Примеры. Пусть $M=60$ и $N=72$. Тогда факторизация $M=5\cdot4\cdot3=3\cdot4\cdot5$ дает в обоих случаях объем памяти, равный 360. При этом число операций ввода-вывода равно 442 и 372 соответственно. Если, с другой стороны, $N=24$, получаем RM=126 и 130, тогда как IO=214 и 180 соответственно.

2.3.3. Алгоритм прямоугольных разбиений

Итак, мы готовы к рассмотрению алгоритма прямоугольных разбиений. Это означает, что теперь и далее $\bar{N}=n_1\cdot\ldots\cdot n_p$, $N_0=N$ и $N_i=\lceil N_{i-1}/n_i\rceil$. Безотносительно к числу дополнительных ячеек, необходимых для формирования строк на каждом шаге (так же, как на шаге p), имеем следующую теорему.

Теорема 4. Если $(m_i)^p{}_{i=1}$ возрастает, а $(n_i)^p{}_{i=1}$ убывает, то $(P_jN_{i-1})^p{}_{i-1}$ не может иметь строгого внутреннего максимума. **В частности, такая упорядоченность будет минимизировать требования к объему памяти и удовлетворять условию**

$$\max_{1\leqslant i\leqslant p}\{P_iN_{i-1}\}\leqslant\max\{\min_{1\leqslant i\leqslant p}\{m_iN\},\ \min_{1\leqslant i\leqslant p}\{n_i+1\}\bar{M}\}.$$

Доказательство. Предположим, что строгий максимум получен **для P_iN_{i-1}, где $1<i<p$. Тогда $P_{i+1}N_i<P_iN_{i-1}$, так что в соответствии с определением N_i**

$$0<P_iN_{i-1}-P_{i+1}N_i=P_i(N_{i-1}-m_{i+1}N_i)\leqslant P_i(N_{i-1}-m_{i+1}N_{i-1}/n_i)=$$
$$=P_iN_{i-1}(1-m_{i+1}/n_i),\qquad(2.48)$$

т. е. $m_{i+1}<n_i$. Из условий теоремы вытекает, что $m_i\leqslant m_{i+1}<n_i\leqslant n_{i-1}$ и, значит, $m_i+1\leqslant n_{i-1}$ (так как имеем дело с целыми числами). Имеем также $P_iN_{i-1}>P_{i-1}N_{i-2}$. Из леммы 2 вытекает, что $N_{i-1}<N_{i-2}/n_{i-1}+1$, т. е. $(N_{i-1}-1)n_{i-1}<N_{i-2}$. Это означает, что

$$0<P_iN_{i-1}-P_{i-1}N_{i-2}=P_{i-1}(m_iN_{i-1}-N_{i-2})<$$
$$<P_{i-1}\{m_iN_{i-1}-(N_{i-1}-1)n_{i-1}\}\leqslant P_{i-1}\{m_{i-1}-(N_{i-1}-1)(m_i+1)\}=$$
$$=P_{i-1}(m_iN_{i-1}-N_{i-1}+1).\qquad(2.49)$$

Поскольку рассматриваются только целые числа, то $N_{i-1}\leqslant m_i\leqslant n_i$. Но тогда $N_i=1$, так что $P_{i+1}N_i=P_{i+1}=m_{i+1}P_i$ и, **поскольку $N_{i-1}\leqslant m_i$, $P_iN_{i-1}\leqslant P_im_i\leqslant P_im_{i+1}=P_{i+1}N_i$,** что противоречит предположению.

Заметим, что если $P_{i+1}N_i<P_iN_{i-1}$, лемма 2 дает строгое неравенство в (2.49), даже если $P_iN_{i-1}=P_{i-1}N_{i-2}$. Это означает, что в массиве не может быть внутренних неизменяющихся участков, поскольку тогда должно выполняться неравенство $P_{i+1}N_i<P_iN_i$ для некоторых i, $1<i<p$.

Очевидно, всегда справедливо соотношение $\max\limits_{1\leqslant i\leqslant p}\{P_iN_{i-1}\}\geqslant$ $\geqslant\max\{(m_1N,\ \bar{M}N_{p-1})\}$, причем равенство имеет место, когда используется предложенное упорядочение. Кроме того, такая гра

ница является минимальной, если m_1 и N_{p-1} минимальны. Но раньше отмечалось, что N_{p-1} является минимумом, если Q_{p-1} является максимумом, т. е. если $n_p=\min\limits_{1\leqslant i\leqslant p}\{n_i\}$. Окончательно из леммы 2 выводим, что

$$N_{p-1}<N/Q_{p-1}+1\leqslant \overline{N}/Q_{p-1}+1=n_p+1 \quad \blacklozenge$$

Примеры. Приводимые здесь примеры показывают, что массив может иметь внутренний минимум. Кроме того, при обратном упорядочении может встретиться внутренний максимум. Следовательно, не существует обратной теоремы, дающей верхнюю границу для необходимого объема памяти.

(m_1, m_2, m_3)	(n_1, n_2, n_3)	N	(P_1N, P_2N_1, P_3N_2)	
(2, 4, 6)	(4, 3, 2)	19	(38, 40, 96)	возрастание
(2, 3, 5)	(6, 4, 2)	47	(94, 48, 60)	внутренний минимум
(2, 3, 4)	(8, 5, 3)	115	(230, 90, 72)	убывание
(10, 9, 8)	(2, 3, 4)	19	(190, 900, 2160)	возрастание
(6, 4, 2)	(2, 3, 4)	19	(114, 240, 144)	внутренний максимум
(2, 3, 4)	(4, 5, 6)	119	(238, 180, 144)	убывание

Для того чтобы создать строки матрицы $\tilde{A}$ на шаге p, необходимы m_pN_{p-1} дополнительных ячеек памяти, если $m_p>N_{p-1}$ (см. [2.6]).

Способы упорядочения по теореме 4 в этом случае уже не гарантируют минимума. Если, например, $N=19$, $(n_1, n_2, n_3)=(4,3,2)$, то необходимое число ячеек памяти равно $96+12=108$ при $(m_1, m_2, m_3)=(2,4,6)$, но если $(m_1, m_2, m_3)=(2,6,4)$, то это число равно $96+8=104$. Поскольку $N_{p-1}\leqslant n_p$, из теоремы 4 можно заключить, что для данных $\overline{M}$ и N

$$\min\{RM_R\}\leqslant \max\{\underline{m}N,\ (\underline{n}+1)\overline{M}+\overline{m}\underline{n}\}, \qquad (2.50)$$

где

$$\underline{m}=\min_{1\leqslant i\leqslant p}\{m_i\};\quad \overline{m}=\max_{1\leqslant i\leqslant p}\{m_i\}\quad \text{и}\quad \underline{n}=\min_{1\leqslant i\leqslant p}\{n_i\}.$$

Метод прямоугольных разбиений обладает важным свойством, заключающимся в том, что существует непосредственная взаимосвязь между объемом памяти и требуемым числом операций ввода-вывода. Это видно из следующей теоремы.

Теорема 5. Сумма $M+2\sum\limits_{i=1}^{p-1}M_iQ_i+N$ является минимальной, если $(m_i)_{i=1}^p$ убывает, а $(n_i)_{i=1}^p$ возрастает, и максимальной если $(m_i)_{i=1}^p$ возрастает, а $(n_i)_{i=1}^p$ убывает.

Доказательство. Докажем сначала первую часть посредством индукции по p. Пусть $p=2$ и $m_1\geqslant m_2$, $n_1\leqslant n_2$. Введем $M^{(1)}_1=[M/m_1]$ и $M^{(2)}_1=[M/m_2]$. Очевидно, $M^{(1)}_1\leqslant M^{(2)}_1$, тогда получаем $M^{(1)}_1n_1\leqslant M^{(1)}_1n_2$ $M^{(1)}_1n_1\leqslant M^{(2)}_1n_1\geqslant M^{(2)}_1n_2$, что доказывает утверждение для $p=2$.

Теперь предположим, что утверждение истинно для $p-1$ сомножителей и рассмотрим произвольные факторизации $\overline{M}=m_1\cdot\ldots\cdot m$ и $\overline{N}=n_1\cdot\ldots\cdot n_p$. Во-первых, переупорядочим $m_1, \ldots, m_{p-1}$ в $m'_1\geqslant\ldots\geqslant m'_{p-1}$ и $n_1, \ldots, n_{p-1}$ в $n'_1\leqslant\ldots\leqslant n'_{p-1}$ и положим $m'_p=$

$=m_p$ и $n'_p=n_p$. Тогда в соответствии с леммой 2 $M'_{p-1}=M_{p-1}$. Очевидно, $Q'_{p-1}=Q_{p-1}$, так что по индукции имеем $\sum\limits_{i=1}^{p-1} M'_i Q'_i \leqslant \sum\limits_{i=1}^{p-1} M_i Q_i$.

Если $m'_p \leqslant m'_{p-1}$, последовательность m'_i является теперь убывающей. В противном случае положим $m''_p=m'_{p-1}$ и $m''_{p-1}=m'_p$, и $m''_i=m'_i$, $i=1, \ldots p-2$. Это дает $M''_{p-1} \leqslant M'_{p-1}$, в то время как $M''_i=M'_i$, $i=1, \ldots p-2$, поэтому $\sum\limits_{i=1}^{p-1} M''_i Q'_i \leqslant \sum\limits_{i=1}^{p-1} M'_i Q'_i$. Окончательно, если $(m''_i)_{i=1}^{p-1}$ не убывает, можно применить теорему к $(p-1)$ сомножителям, чтобы получить $m'''_1 \geqslant \ldots \geqslant m'''_{p-1}$, $m'''_p \geqslant m''_p$, где $\sum\limits_{i=1}^{p-1} M'''_i Q'_i \leqslant \sum\limits_{i=1}^{p-1} M''_i Q'_i$.

Соответственно, если $n'_p < n'_{p-1}$, положим $n''_i=n'_i$, $i=1, \ldots, p-2$, $n''_{p-1}=n'_p$ и $n''_p=n'_{p-1}$. Тогда $Q''_i=Q'_i$, $i=1, \ldots, p-2$, и $Q''_{p-1} < Q'_{p-1}$, поэтому $\sum\limits_{i=1}^{p-1} M'''_i Q'_i < \sum\limits_{i=1}^{p-1} M'''_i Q'_i$. Используя утверждение для $(p-1)$ сомножителей еще раз, получим $n'''_1 \leqslant \ldots \leqslant n'''_{p-1}$ и $n'''_p=n''_p$, где $\sum\limits_{i=1}^{p-1} M'''_i Q'''_i \leqslant \sum\limits_{i=1}^{p-1} M'''_i Q''_i \leqslant \sum\limits_{i=1}^{p-1} M_i Q_i$, и $(m'''_i)_{i=1}^{p}$ убывает, а $(n'''_i)_{i=1}^{p}$ возрастает.

Вторая часть теоремы следует из аналогичных рассуждений ◆

2.3.4. О преимуществах введения единичного сомножителя

В рассмотренных теоремах, относящихся к алгоритму разбиения на квадраты, предполагалось, что все сомножители $m_1, \ldots, m_p$ больше 1. На самом деле это предположение не является необходимым. Легко увидеть, что при удалении любого единичного сомножителя требуемый объем памяти будет таким же, а число операций ввода-вывода уменьшится.

Алгоритм разбиения на прямоугольники не обладает этим свойством, если только единичный сомножитель не встречается внутри массива. Более точно имеем следующее.

Утверждение 2. Пусть $\bar{M}=m_1 \cdot \ldots \cdot m_p$ и $\bar{N}=n_1 \cdot \ldots \cdot n_p$. Тогда, если $m_k=1$ для некоторого k, $1<k \leqslant p$, существуют разбиения $m'_1 \cdot \ldots \cdot m'_{p-1}=\bar{M}$ и $n'_1 \cdot \ldots \cdot n'_{p-1}=\bar{N}$, для которых

$$\max_{1 \leqslant i \leqslant p-1} \{P'_i N'_{i-1}\} \leqslant \max_{1 \leqslant i \leqslant p} \{P_i N_{i-1}\} \text{ и } \sum_{i=1}^{p-2} M'_1 Q'_i \leqslant \sum_{i=1}^{p-1} M_i Q_i.$$

Такое же утверждение можно сделать, если $n_k=1$ и $1 \leqslant k \leqslant p$.

Доказательство. Если $m_k=1$, то новые разбиения задаются значениями $m'_i=m_i$, $i=1, \ldots, k-1$; $m'_i=m_{i+1}$, $i=k, \ldots, p-1$; $n'_i=n_i$, $i=1, \ldots, k-2$; $n'_{k-1}=n_k n_{k-1}$, $n'_i=n_{i+1}$, $i=k, \ldots, p-1$. Поскольку $N'_{k-1}=N_k$, требования к объему памяти будут определяться

величиной $\max_{1\leqslant i\leqslant p}\{p_iN_{i-1}\}$. В сумме для определения числа операций ввода-вывода слагаемое M_kQ_k может быть удалено.

Если $n_k=1$, положим $m'_k=m_km_{k+1}$ и сохраним остальные сомножители, чтобы получить тот же самый результат ◆

2.4. Оптимизация алгоритма разбиение на квадраты и алгоритма «ввод строки-вывод столбца»

Результаты подразд. 2.3.2 можно использовать теперь для поиска оптимального способа транспонирования произвольной матрицы с помощью алгоритма разбиения на квадраты или алгоритма «ввод строки-вывод столбца».

Более точно, будем минимизировать требования к объему памяти для заданного числа просмотров данных. Это не всегда обеспечивает минимум числа операций ввода-вывода, но отклонения от минимума малы (см. подразд. 2.3.2). Более того, если оптимум с точки зрения требуемого объема памяти достигнут без введения нулевых строк ($\bar{M}=M$), то требования к числу операций ввода-вывода также минимальны (см. подразд. 2.2.3). И наконец, так как число просмотров данных, необходимое для матрицы $M\times N$, ограничено сверху $\lceil\log_2 M\rceil$ ($M\leqslant N$) или $\lceil\log_2 N\rceil+1$ ($M>N$) — можно найти (приблизительный) оптимум и с точки зрения объема памяти, и с точки зрения числа операций ввода-вывода при заданных относительных стоимостях памяти и операций обмена.

Сначала остановимся на случае $M\leqslant N$. Тогда для данного p можно действовать следующим образом.

1. Положить $m=\lfloor M^{1/p}\rfloor$. (Тогда $m=\min\{k;\ k^p\geqslant M\}$.)

2. Найти наименьшее произведение $\bar{M}=m_1\cdot\ldots\cdot m_p\geqslant M$ при $m_1\geqslant\ldots\geqslant m_p$, используя $m_1=m$, поскольку $(m-1)^p<M$ согласно п. 1.

3. Вычислить $\mathrm{RM_s}(\bar{M})$ — необходимое число ячеек оперативной памяти в соответствии с (2.25).

4. Найти лучшее произведение $\bar{M}'=m'_1\cdot\ldots\cdot m'_p\geqslant M$, $m'_1\geqslant\ldots\geqslant m'_p$, для которого $m'_1<\mathrm{RM_s}(\bar{M})/N$. (Мы должны рассматривать только эти произведения, так как $\mathrm{RM_s}(\bar{M}')\geqslant m'_1N$.) Заметим, что $m'_1\geqslant m_1$, потому что $\bar{M}'\geqslant\bar{M}$.

5. Если такое $\bar{M}'$ найдено, заменить $\bar{M}$ на $\bar{M}'$ и повторить п. 4.

Множество произведений в п. 4 предпочтительно исследовать в (возрастающем) лексикографическом порядке. Кроме того, из указанного способа выбора $\bar{M}$ и теоремы 3 следует, что надо рассматривать только те произведения, для которых $\bar{M}'>N$, т. е. поиск в п. 4 ведется по множеству

$$\Omega=\{M'=m'_1\cdot\ldots\cdot m'_p;\quad m'_1\geqslant\ldots\geqslant m'_p,$$
$$\bar{M}>N,\quad m'_1\leqslant\mathrm{RM_s}(\bar{M})/N\}. \tag{2.51}$$

Случай $M>N$ несколько сложнее. Во-первых, рассматривается матрица, разбитая на матрицы $N\times N$, и вычисляется оптимальное произведение $\bar{M}$ из p сомножителей для них. Если $\bar{M}\geqslant M$, то получена наилучшая p-шаговая факторизация. Если $\bar{M}<M$, то

для объединения строк в строки, состоящие из M элементов, требуется дополнительный шаг. Если пренебречь всеми нулевыми элементами, понадобится $\max\{M, RM_s(\bar{M})\}$ ячеек памяти для транспонирования за $(p+1)$ шагов. Однако есть возможность получить лучший результат, если непосредственно применить транспонирование за $(p+1)$ шагов.

Например, если $M=27$, $N=25$ и $p=2$, то $\bar{M}=5\cdot5$ и потребуется 125 ячеек памяти с использованием трех шагов. Но матрица 27×25 может быть транспонирована за 3 шага с помощью 81 ячейки ($M=3\cdot3\cdot3$). Заметим, что по теореме 2 вслед за транспонированием матриц $N\times N$ за $(p-1)$ шагов надо выполнить объединение строк которое при $\bar{M}\geqslant M$ никогда не может дать лучшего результата.

В табл. 2.1 приведен пример работы этого метода для матрицы 620×1000.

Таблица 2.1

Требования к объему памяти для транспонирования матрицы 620×1000 при оптимальных $RM_s(\bar{M})$

p	$\bar{M}$	$RM_s(\bar{M})$	Ω
2	25^2	25000	$\varnothing$
3	$9^2\cdot8$	9072	$\varnothing$
4	5^4	5000	$\varnothing$
5	$4^4\cdot3$	4096	$\{4^5\}$
6	3^6	3645	$\varnothing$
7	$3^4\cdot2^3$	3078	$\{3^{7-k}2^k;\ k=0, \ldots, 2\}$
8	$3^3\cdot2^5$	3024	$\{3^{8-k}2^k;\ k=0, \ldots, 4\}$
9	$3\cdot2^8$	3000	$\{3^{9-k}2^k;\ k=0, \ldots, 7\}$
10	2^{10}	2048	$\varnothing$

2.5. Примеры

Для иллюстрации работы четырех алгоритмов, представленных в подразд. 2.2.3 — 2.2.6, приведены результаты их применения к матрице 6×6.

1. Метод Флойда. RM=12, IO=24. Заметим, что матрицы $A^{(1)}$, $B^{(0)}$ и $B^{(6)}$ не создаются.

$$\begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 \\ 10 & 11 & 12 & 13 & 14 & 15 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 30 & 31 & 32 & 33 & 34 & 35 \\ 40 & 41 & 42 & 43 & 44 & 45 \\ 50 & 51 & 52 & 53 & 54 & 55 \end{bmatrix} \qquad \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 \\ 50 & 51 & 52 & 53 & 54 & 55 \\ 40 & 41 & 42 & 43 & 44 & 45 \\ 30 & 31 & 32 & 33 & 34 & 35 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 10 & 11 & 12 & 13 & 14 & 15 \end{bmatrix} \qquad \begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 \\ 51 & 52 & 53 & 54 & 55 & 50 \\ 42 & 43 & 44 & 45 & 40 & 41 \\ 33 & 34 & 35 & 30 & 31 & 32 \\ 24 & 25 & 20 & 21 & 22 & 23 \\ 15 & 10 & 11 & 12 & 13 & 14 \end{bmatrix}$$

а) $A=A^{(0)}$ б) $A^{(1)}$, где $a^{(1)}_{i,j}=a_{-i,j}$ в) $B^{(0)}$, где $b^{(0)}_{i,i+j}=a^{(1)}_{i,j}$

$$\begin{bmatrix} 00 & 10 & 02 & 12 & 04 & 14 \\ 51 & 01 & 53 & 03 & 55 & 05 \\ 42 & 52 & 44 & 54 & 40 & 50 \\ 33 & 43 & 35 & 45 & 31 & 41 \\ 24 & 34 & 20 & 30 & 22 & 32 \\ 15 & 25 & 11 & 21 & 13 & 23 \end{bmatrix}$$

г) $B^{(1)}$, столбцы с $j_0=1$ сдвинуты на 1 шаг

$$\begin{bmatrix} 00 & 10 & 20 & 30 & 04 & 14 \\ 51 & 01 & 11 & 21 & 55 & 05 \\ 42 & 52 & 02 & 12 & 40 & 50 \\ 33 & 43 & 53 & 03 & 31 & 41 \\ 24 & 34 & 44 & 54 & 22 & 32 \\ 15 & 25 & 35 & 45 & 13 & 23 \end{bmatrix}$$

д) $B^{(2)}$, столбцы с $j_1=1$ сдвинуты на 2 шага

$$\begin{bmatrix} 00 & 10 & 20 & 30 & 40 & 50 \\ 51 & 01 & 11 & 21 & 31 & 41 \\ 42 & 52 & 02 & 12 & 22 & 32 \\ 33 & 43 & 53 & 03 & 13 & 23 \\ 24 & 34 & 44 & 54 & 04 & 14 \\ 15 & 25 & 35 & 45 & 55 & 05 \end{bmatrix}$$

е) $B^{(3)}$, столбцы с $j_2=1$ сдвинуты на 4 шага

$$\begin{bmatrix} 00 & 10 & 20 & 30 & 40 & 50 \\ 01 & 11 & 21 & 31 & 41 & 51 \\ 02 & 12 & 22 & 32 & 42 & 52 \\ 03 & 13 & 23 & 33 & 43 & 53 \\ 04 & 14 & 24 & 34 & 44 & 54 \\ 05 & 15 & 25 & 35 & 45 & 55 \end{bmatrix}$$

ж) $A^{(1)}$, где $a^1_{i,j}=b^{(3)}_{i,i+1}$

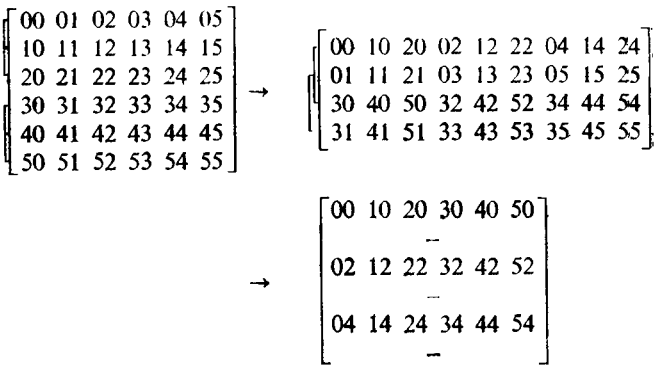
2. Алгоритм разбиения на квадраты: $\bar{M}=3\cdot 2$, RM=18, IO=24. Показано, как группируются строки и как отображаются группы строк на шаге 2.

$$\begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 \\ 10 & 11 & 12 & 13 & 14 & 15 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 30 & 31 & 32 & 33 & 34 & 35 \\ 40 & 41 & 42 & 43 & 44 & 45 \\ 50 & 51 & 52 & 53 & 54 & 55 \end{bmatrix} \rightarrow \begin{bmatrix} 00 & 10 & 20 & 03 & 13 & 23 \\ 01 & 11 & 21 & 04 & 14 & 24 \\ 02 & 12 & 22 & 05 & 15 & 25 \\ 30 & 40 & 50 & 33 & 43 & 53 \\ 31 & 41 & 51 & 34 & 44 & 54 \\ 32 & 42 & 52 & 35 & 45 & 55 \end{bmatrix} \rightarrow \begin{bmatrix} 00 & 10 & 20 & 30 & 40 & 50 \\ - & - & - & - & - & - \\ - & - & - & - & - & - \\ 03 & 13 & 23 & 33 & 43 & 53 \\ - & - & - & - & - & - \\ - & - & - & - & - & - \end{bmatrix}$$

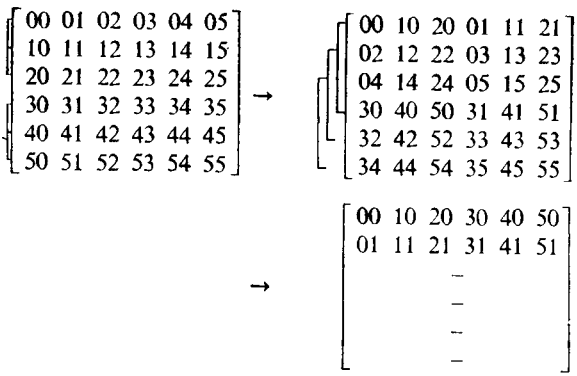
3. Метод прямоугольных разбиений: $\bar{M}=2\cdot 3$, $\bar{N}=3\cdot 2$, RM=18 (возможно также RM=12, см. [2.6]), IO=30.

$$\begin{bmatrix} 00 & 01 & 02 & 03 & 04 & 05 \\ 10 & 11 & 12 & 13 & 14 & 15 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 30 & 31 & 32 & 33 & 34 & 35 \\ 40 & 41 & 42 & 43 & 44 & 45 \\ 50 & 51 & 52 & 53 & 54 & 55 \end{bmatrix} \rightarrow \begin{bmatrix} 00 & 10 & 03 & 13 \\ 01 & 11 & 04 & 14 \\ 02 & 12 & 05 & 15 \\ 20 & 30 & 23 & 33 \\ 21 & 31 & 24 & 34 \\ 22 & 32 & 25 & 35 \\ 40 & 50 & 43 & 53 \\ 41 & 51 & 44 & 54 \\ 42 & 52 & 45 & 55 \end{bmatrix} \rightarrow \begin{bmatrix} 00 & 10 & 20 & 30 & 40 & 50 \\ & & & & & \\ 03 & 13 & 23 & 33 & 43 & 53 \\ & & & & & \end{bmatrix}$$

4. Прямоугольный метод: $\bar{M}=3\cdot 2$, $\bar{N}=2\cdot 3$, IO=20, RM=18.



5) Метод «ввод строк-вывод столбца»: $\bar{M}=3\cdot 2$; RM=18, IO=24.



2.6. Метод Андерсона для непосредственного вычисления многомерного ДПФ

Андерсон [2.1] предложил альтернативный метод решения задачи ДПФ матрицы, записанной построчно в ЗУ с произвольным доступом. Поскольку внешняя сумма (2.2б) является ДПФ для фиксированного l, очевидно, что для ее вычисления можно использовать БПФ. Основная идея использования БПФ заключается в том, что ДПФ массива может быть вычислено итеративным суммированием, при котором каждая сумма в свою очередь является ДПФ. Если, в частности, размер массива M является степенью 2, то каждая частичная сумма может рассматриваться как двухточечные ДПФ. Тогда (см., например, [2.16] или [2.17]) на i-м шаге, $i=1, \ldots, \log_2 M$, преобразуются две точки с индексами m и m', двоичные представления которых отличаются точно в позиции i. При этом результирующий массив оказывается представленным в двоично-инвертированном порядке, но можно так

переупорядочить исходный массив, что результирующие значения будут следовать в естественном порядке (см. также [2.18]).

Обнаружив, что сумма в (2.2б) вычисляется для всех l, можно действовать следующим образом. На шаге i в оперативную память считываются пары строк, для которых индексы отличаются точно в позиции i их двоичного представления. Затем на месте выполняется двухэлементное ДПФ для всех столбцов. Эти модифицированные строки записываются снова на свои старые места. После $\log_2 M$ шагов получаем нужное преобразование, но расположенное в двоично-инвертированном порядке. Однако это не составляет никакой проблемы, поскольку известен ключ для перестановки, который можно использовать либо при вводе, либо при дальнейшей обработке. Безотносительно к тому, как учитывается двоичная инверсия, можно заметить, что адресация строк полностью аналогична адресации в методе разбиения на квадраты (и в алгоритме Флойда) для транспонирования матриц.

Эта аналогия между двумя методами распространяется на любые возможные обобщения. В обоих случаях можно снизить время ввода-вывода, рассматривая более двух строк одновременно. Действительно, если матрица содержит $M \times M$ элементов и есть возможность хранить в оперативной памяти 2^m строк, то число операций ввода-вывода

$$IO_A = 2 \cdot M \cdot \lceil \log_2 M/m \rceil. \qquad (2.52)$$

Если M и N являются степенями 2, то IO_A совпадает с IO_S и IO_F. Кроме того, метод обобщается на произвольную факторизацию M, так же как и БПФ, и сохраняется аналогия с методами транспонирования, описанными в разд. 2.2. Следовательно, эффективность, измеренная с точки зрения операций ввода-вывода, остается той же самой и в этом случае.

Непосредственный метод принципиально прост за счет того, что он сам основан на использовании свойств быстрых алгоритмов (например, БПФ). Его достоинство проявляется также в возможности обрабатывать на месте произвольную матрицу. Однако метод обладает рядом недостатков, о которых будет сказано в следующем разделе.

2.7. Обсуждение результатов

Были представлены два подхода к организации вычисления разделимых двумерных преобразований для матриц с построчным доступом (или с доступом по столбцам). Один из методов основан на транспонировании матрицы, а другой — на свойствах быстрых алгоритмов, таких, как БПФ. Было показано, что эти методы по существу эквивалентны с точки зрения эффективности в случае, когда размеры матрицы являются составными числами. (Предполагается, что на самом деле транспонирование в оперативной памяти не выполняется, так что время обработки одинаково для

обоих подходов). В таких случаях непосредственный метод может применяться для преобразования любой матрицы с оставлением на месте. Однако этому методу присуща меньшая модульность при проектировании, и поэтому более общий метод с использованием транспонирования матриц дает бо́льшую эффективность в других случаях, как будет показано. Модульность методов транспонирования подразумевает также, что их можно применять в комбинации с аппаратурной реализацией быстрых преобразований.

Преимущество модульности особенно очевидно в следующем случае. Предположим, что необходимо вычислить БПФ матрицы $M \times N$ и что M содержит большой простой сомножитель. Суммирования в БПФ будут одинаковыми для обоих методов, но требуемый объем памяти может быть резко снижен, если преобразование выполняется с помощью удобного разбиения. Заполнение массива нулями при вычислении БПФ может оказаться нежелательным, поскольку при этом изменяются результаты преобразования. Рассмотрим случай, приведенный, в табл. 2.1. При непосредственном методе потребуются два просмотра данных и 31 000 ячеек памяти. Этот объем памяти сравним с 25 000 ячеек, необходимых при транспонировании за два просмотра. (Одномерное ДПФ может быть совмещено с транспонированием, так что двух просмотров достаточно.) Кроме того, в табл. 2.1 показано, что транспонирование может быть реализовано со значительно меньшими затратами объема памяти за счет увеличения времени обмена. Наоборот, есть случаи, когда можно сэкономить время ввода-вывода за счет объема памяти.

Упомянем некоторые менее важные различия между методами. Предположим, что надо выполнить прямое и обратное преобразование матрицы $M \times N$ и что M и N выбраны достаточно сложными составными числами, чтобы не приходилось увеличивать матрицы, дополняя их нулями. Предположим, что KN элементов заполняют оперативную память, и что KN/M — целое, тогда

$$IO_S = 2\left(\lceil \log_2 M/\log_2 K \rceil M + \lceil \log_2 N/\log_2 (KN/M) \rceil N, \right. \quad (2.53)$$

$$IO_A = 4 \lceil \log_2 M/\log_2 K \rceil M. \quad (2.54)$$

На основе этих выражений можно показать, что $IO_S < IO_A$, если $M < N$, и $IO_S > IO_A$, если $M > N$. Например, если $M = 2^{10}$, $N = 2^8$ и $K = 2^3$, то $IO_S = 48 \cdot 2^8$, а $IO_A = 64 \cdot 2^8$, но если $M = 2^8$, $N = 2^{10}$, то $IO_S = 22 \cdot 2^8$ и $IO_A = 12 \cdot 2^8$. Можно заметить, также, что если $M > N$, минимальный объем памяти для выполнения пары преобразований, независимо от времени ввода-вывода, меньше для непосредственного метода.

Все рассмотренные алгоритмы транспонирования матриц являются оптимальными для случая, когда размеры матрицы являются степенями 2. При этом они могут быть реализованы с помощью только операций сдвига и маскирования. Однако когда эти алгоритмы применяются для транспонирования произвольных прямоугольных матриц, между ними есть небольшое различие.

Алгоритм Флойда может работать с любой квадратной матрицей, используя только операции сдвига и маскирования, и распространяется на прямоугольные матрицы так, как описано в подразд. 2.2.3. Однако три других алгоритма не обладают этим «двоичным» свойством, поскольку они используют расширение в форме (2.15). С другой стороны, они могут выполнять транспонирование с меньшими затратами операций ввода-вывода при чуть большем объеме памяти. Эти три алгоритма требуют также специального этапа оптимизации, который достаточно прост для метода разбиения на квадраты, и алгоритма «ввод строки-вывод столбца», но несколько более сложен для алгоритма разбиения на прямоугольники.

Последний алгоритм также несколько сложнее по своей сущности, так как число и длины строк изменяются во время транспонирования, даже если размеры матриц являются составными числами.

В общем, более простые алгоритмы разбиения на квадраты и «ввод строки-вывод столбца» почти всегда эффективны. Поэтому они предпочтительнее, если не требуется реализации только посредством сдвигов и маскирования, когда можно использовать алгоритм Флойда.

ГЛАВА 3

ВЫЧИСЛЕНИЕ ДВУМЕРНЫХ СВЕРТОК И ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

(Г. Дж. Нуссбаумер) [1]

Главная цель главы — вывод быстрых алгоритмов вычисления двумерных сверток и дискретных преобразований Фурье (ДПФ). Эти алгоритмы базируются на преобразовании двумерных последовательностей в одномерные с использованием основных свойств колец полиномов. По сравнению с более традиционными методами, основанными на быстром преобразовании Фурье (БПФ), они значительно снижают число арифметических операций, требуемых для вычисления сверток и ДПФ.

В начале главы дан краткий обзор тех разделов алгебры полиномов, которые имеют отношение к вычислению сверток и ДПФ. Затем вводится новый вид преобразований, называемых *полиномиальными преобразованиями*, и показано, что эти преобразования, которые можно рассматривать как ДПФ, определенные на полях полиномов, являются эффективным средством вычисления двумерной фильтрации. Обсуждается также применение полиномиальных преоб-

[1] IBM Centre d'Etudes et Recherches F-06610 LaGaude, France.

разований для вычисления двумерных ДПФ и показано, что эти преобразования могут быть использованы в комбинации с алгоритмом Винограда или БПФ для дальнейшего снижения числа арифметических операций.

3.1. Свертки и алгебра полиномов

Основой многих новейших методов быстрого вычисления сверток и ДПФ, таких, как алгоритм Винограда преобразования Фурье [3.1], алгоритм быстрой свертки Агарвала — Кули [3.2] и полиномиальные преобразования [3.3—3.5], является алгебра полиномов. Поскольку алгебра полиномов не нашла еще широкого применения в радиоэлектронике, за исключением теории кодирования, приведем здесь краткие сведения, касающиеся наиболее важных аспектов, связанных с вычислением сверток. Для более детального изучения читатель может воспользоваться любым учебником по современной алгебре [3.6].

Рассмотрим сначала дискретную свертку

$$y_m = \sum_{l=0}^{L-1} h_l x_{m-l}. \tag{3.1}$$

При прямом вычислении этой конечной дискретной свертки требуется выполнить L умножений на каждый входной отсчет. В большинстве быстрых алгоритмов бесконечная входная последовательность $\{x_n\}$ делится на последовательные блоки по N отсчетов и y_m вычисляется как сумма апериодических сверток каждого из этих блоков с L-точечной последовательностью $\{h_l\}$. Вычисление этих апериодических сверток эквивалентно определению коэффициентов произведения $Y(Z)$ двух полиномов $H(Z)$ и $X(Z)$:

$$H(Z) = \sum_{l=0}^{L-1} h_l Z^l, \tag{3.2}$$

$$X(Z) = \sum_{n=0}^{N-1} x_n Z^n, \tag{3.3}$$

$$Y(Z) = H(Z) X(Z) = \sum_{m=0}^{L+N-2} y_m Z^m, \tag{3.4}$$

где h_l и x_n — элементы некоторого поля F обычно вещественных или комплексных чисел.

Очевидно, что полиномиальное представление свертки является более сложным, чем стандартное представление. Мы, однако, увидим, что использование полиномов обеспечивает эффективное снижение сложности вычисления сверток и полиномиальных произведений. Для этого необходимо сначала ввести понятия об остаточных полиномах и китайской теореме об остатках.

3.1.1. Остаточные полиномы

Рассматривая полиномы с коэффициентами, заданными на поле, определим что полином $P(Z)$ является делителем полинома $H(Z)$, если можно найти такой полином $D(Z)$, для которого выполняется соотношение $H(Z) = P(Z)D(Z)$. Полином $H(Z)$ называется неприводимым, если только его единственными делителя-

ми являются полиномы $P(Z)$, степени которых равны 0. Если $P(Z)$ не является делителем $H(Z)$, то при делении $H(Z)$ на $P(Z)$ образуется остаток $R(Z)$:

$$H(Z) = P(Z)\,D(Z) + R(Z). \tag{3.5}$$

Легко показать, что это представление единственно. Все полиномы, которые при делении на $P(Z)$ дают один и тот же остаток $R(Z)$, называются конгруэнтными по модулю $P(Z)$. Соотношение конгруэнтности обозначается

$$R(Z) \equiv H(Z) \bmod P(Z). \tag{3.6}$$

Вычисление $R(Z)$ значительно упрощается, если $P(Z)$ — произведение d полиномов, не имеющих общих множителей (обычно такие полиномы называются взаимно-простыми по аналогии с взаимно-простыми числами):

$$P(Z) = \prod_{i=1}^{d} P_i(Z). \tag{3.7}$$

В этом случае $R(Z)$ может быть однозначно выражен как функция полиномов $H_i(Z)$, полученных приведением $H(Z)$ по модулю различных полиномов $P_i(Z)$. Это выражение, являющееся обобщением на кольцах полиномов известной в теории чисел китайской теоремы об остатках [3.7], задается как:

$$R(Z) \equiv H(Z) \bmod P(Z) \equiv \sum_{i=1}^{d} S_i(Z)\,H_i(Z) \bmod P(Z), \tag{3.8}$$

где для каждого значения u индекса i

$$\begin{aligned} S_u(Z) &\equiv 0 \bmod P_i(Z),\ i \neq u, \\ &\equiv 1 \bmod P_u(Z); \end{aligned} \tag{3.9}$$

$$S_u(Z) \equiv \left\{\left[\prod_{\substack{i=1\\ i\neq u}}^{d} P_i(Z)\right] \Big/ \left\{\left[\prod_{\substack{i=1\\ i\neq u}}^{d} P_i(Z)\right] \bmod P_u(Z)\right\}\right\} \bmod P(Z). \tag{3.10}$$

Соотношения (3.8),(3.9) можно легко проверить приведением (3.8) по модулю различных полиномов $P_u(Z)$. Поскольку $[H(Z) \bmod P(Z)] \bmod P_i(Z) \equiv H_i(Z)$, выражение (3.8) выполняется для каждого $P_u(Z)$. Точно так же приведение $S_u(Z)$, определенного в (3.10), по модулю $P_u(Z)$ дает (3.9) при условии, что $\prod_{i=1}^{d} P_i(Z) \not\equiv 0 \bmod P_u(Z)$. Выполнение последнего условия гарантируется тем, что полиномы $P_u(Z)$ являются взаимно-простыми. Полиномы $S_u(Z)$ можно легко построить, пользуясь полиномиальным вариантом алгоритма Евклида.

3.1.2. Алгоритмы свертки и произведений полиномов в алгебре полиномов

Китайская теорема об остатках является центральной при вычислении сверток, так как она позволяет заменить вычисление произведения $H(Z)X(Z)$ двух больших полиномов по модулю $P(Z)$ на d произведений $H_i(Z)X_i(Z)$ по модулю $P_i(Z)$ значительно меньших полиномов. Здесь $X_i(Z) = X(Z) \bmod P_i(Z)$. Покажем теперь, как китайскую теорему об остатках можно использовать для получения минимального числа умножений, требуемых для вычисления свертки или полиномиальных произведений, и как достичь на практике этой нижней границы.

Теорема 3.1. (Алгоритм Кука — Тоома [3.2].) Апериодическая свертка (3.4) может быть вычислена за $L+N-1$ общих умножений.

Конструктивное доказательство теоремы можно получить, если заметить, что $Y(Z)$ имеет степень $L+N-2$. Следовательно, $Y(Z)$ не изменится, если оно определено по модулю любого полинома $P(Z)$ степени, равной $L+N-1$:

$$Y(Z) \equiv H(Z)\,X(Z) \bmod P(Z). \tag{3.11}$$

Предположим, что $P(Z)$ выбран так, что

$$P(Z) = \prod_{i=1}^{L+N-1} (Z - a_i), \tag{3.12}$$

где a_i есть $L+N-1$ различных чисел в поле F коэффициентов. $Y(Z)$ может быть вычислен путем приведения $H(Z)$ и $X(Z)$ по модулю $(Z-a_i)$, что эквивалентно подстановке a_i вместо Z в $H(Z)$ и $X(Z)$. Тогда $Y(Z)$ вычисляется за $L+N-1$ общих умножений с помощью $L+N-1$ произведений $H(a_i)X(a_i)$ и восстановления $Y(Z)$ в соответствии с китайской теоремой об остатках. ◆

Заметим, что приведение по модулю $(Z-a_i)$ и восстановление в соответствии с китайской теоремой об остатках подразумевают умножения на скаляры, которые являются степенями a_i. Для коротких сверток в качестве $(L+N-1)$ различных a_i можно выбрать просто целые числа, например, 0, +1, —1, так что эти умножения становятся либо тривиальными, либо сводятся к нескольким сложениям. Для бо́льших сверток, однако, требование, чтобы все a_i были различными, приводит к необходимости выбора бо́льших чисел, так что эти скалярные произведения должны либо трактоваться как общие умножения, либо будут требовать большого числа сложений. Таким образом, алгоритм Кука — Тоома практически используется только для коротких сверток. Для бо́льших сверток лучшего баланса между минимизацией числа умножений и минимизацией числа сложений можно достичь с помощью методов преобразования, которые, как мы покажем, тесно связаны с алгоритмом Кука — Тоома.

В алгоритме Кука — Тоома результирующий полином $Y(Z)$ степени $L+N-2$ восстановлен из $L+N-1$ числовых значений $Y(a_i)$, полученных заменой Z на a_i в $H(Z)$ и $X(Z)$. Таким образом, алгоритм Кука — Тоома может рассматриваться как реализация интерполяционного метода Лагранжа [3.2]. Выбор $L+N-1$ различных a_i и поля коэффициентов произволен. Если различные a_i выбраны в качестве последовательных степеней числа W и если поле F таково, что $1, W, W^2, \ldots, W^{L+N-2}$ все различны, алгоритм Кука — Тоома приводит к вычислению апериодической свертки с помощью преобразований, имеющих структуру ДПФ. Например, если $W=2$ и F есть кольцо чисел по модулю числа Мерсенна $(2^q-1,\ q$ — простое) или числа Ферма $(2^q+1,\ q=2^t)$, то такие преобразования будут преобразованиями Мерсенна или Ферма [3.8, 3.9].

Когда F является полем комплексных чисел и $W=\exp[-j2\pi/(L+N-1)]$, $j=\sqrt{-1}$, использование алгоритма Кука — Тоома эквивалентно вычислению апериодической свертки с помощью ДПФ. В этом случае $P(Z)$ преобразуется к виду

$$P(Z) = \prod_{i=0}^{L+N-2} (Z - W^i) = Z^{L+N-1} - 1, \tag{3.13}$$

а апериодическая свертка L-точечной последовательности $\{h_l\}$ и N-точечной $\{x_n\}$ вычисляется как циклическая свертка двух расширенных последовательно-

стей из $L+N-1$ точек, полученных добавлением $N-1$ нулей к последовательности $\{h_l\}$ и $L-1$ нулей к последовательности $\{x_n\}$. Таким образом, традиционный метод вычисления апериодических сверток с помощью ДПФ и метод секционирования с суммированием [3.10] близко связаны с алгоритмом Кука — Тоома.

Из теоремы 3.1 видно, что апериодическая $(L+N-2)$-точечная свертка может быть вычислена с помощью $L+N-1$ умножений посредством полиномиального умножения по модулю полинома $P(Z)$ степени $D=L+N-1$, причем этот полином выбирается как произведение D полиномов первой степени в поле коэффициентов F. Полученные результаты на случай полиномиальных произведений по модулю любого полинома $P(Z)$ обобщает следующая важная теорема, принадлежащая Винограду [3.11].

Теорема 3.2. Минимальное число общих умножений, необходимых для вычисления произведения полиномов $Y(Z) \equiv H(Z)X(Z) \bmod P(Z)$, равно $(2D-d)$, где D — степень $P(Z)$ и d — число неприводимых множителей $P_i(Z)$ полинома $P(Z)$ поля F (включая 1).

Конструктивное доказательство этой теоремы строится с помощью китайской теоремы об остатках. Поскольку $P(Z)$ является произведением d неприводимых полиномов $P_i(Z)$, $Y(Z)$ можно найти приведением $H(Z)$ и $X(Z)$ по модулю $P_i(Z)$ вычислением d полиномиальных произведений $Y_i(Z) \equiv H_i(Z)X_i(Z) \bmod P_i(Z)$, соответствующих приведенным полиномам $H_i(Z)$, $X_i(Z)$, и восстановлением $Y(Z)$ из $Y_i(Z)$ с помощью китайской теоремы об остатках. Как и в теореме 3.1, умножения на скаляры, используемые при приведении и восстановлении по китайской теореме об остатках, не учитываются, и единственными общими умножениями являются те, которые входят в произведение $H_i(Z)X_i(Z)$. Пусть D_i — степень $P_i(Z)$ и $D=\sum_{i=1}^{d} D_i$. Поскольку $P_i(Z)$ имеет степень D_i, $H_i(Z)$ и $X_i(Z)$ имеют степень D_i-1, то их произведение можно вычислить за $2D_i-1$ умножений по теореме 3.1:

$$Y_i(Z) \equiv [H_i(Z)\,X_i(Z) \bmod Q_i(Z)] \bmod P_i(Z), \qquad (3.14)$$

где $Q_i(Z)$ — полином степени $2D_i-1$, выбирается в виде произведения $(2D_i-1)$ полиномов первой степени в поле F. Таким образом, число общих умножений соответствует $\sum_{i=1}^{d} (2D_i-1) = 2D-d$. ◆

Важное следствие из теоремы 3.2 относится к вычислению циклических сверток. Циклическую свертку двух D-точечных последовательностей можно представлять в полиномиальных обозначениях как полиномиальное произведение по модулю (Z^D-1). Было показано, что, если F — поле комплексных чисел, (Z^D-1) разлагается на D полиномов $(Z-W^i)$ первой степени и что циклическая свертка вычисляется с помощью ДПФ посредством D общих умножений. К сожалению, W^i — иррациональные и комплексные числа, поэтому умножения на скаляры, используемые при вычислении ДПФ, должны рассматриваться как общие умножения.

Если F — поле рациональных чисел, (Z^D-1) разлагается на произведения полиномов с рациональными коэффициентами. Такие полиномы называются *циклотомическими* [3.7]. Циклотомические полиномы являются неприводимыми в поле рациональных чисел. Их корни комплексны и образуют подмножество множества W^0, W^1, ..., W^{D-1}, где $W=\exp(-j\,2\pi/D)$, $j=\sqrt{-1}$.

Можно показать, что для заданного D число d различных циклотомических полиномиальных множителей (Z^D-1) равно числу делителей D, включая 1 и D. Кроме того, степень каждого циклотомического полинома, соответствующая данному делителю D_i числа D, равна функции Эйлера $\varphi(D_i)$ [3.7]. Таким образом, для циклической свертки теорема 3.2 сводится к следующей теореме.

Теорема 3.3. Минимальное число общих умножений, необходимое для вычисления циклической D-точечной свертки, равно $(2D-d)$, где d — число делителей D, включая 1 и D.

Вычисление циклической свертки по теореме 3.2 значительно упрощается вследствие того, что коэффициенты циклотомических полиномов являются просто целыми числами. Более того, эти целые числа могут принимать значения только 0, +1 или —1, за исключением случая очень больших циклотомических полиномов [3.2]. При этом приведение и восстановление по китайской теореме об остатках могут быть выполнены с малым числом сложений, что можно проиллюстрировать на простом примере 5-точечной циклической свертки. В этом случае имеются только два делителя числа 5: 1 и 5. Таким образом, (Z^5-1) разлагается на два циклотомических полинома: $Z^5-1=(Z-1)(Z^4+Z^3+Z^2+Z+1)$. Приведение входных полиномов $H(Z)$ и $X(Z)$ по модулю $Z-1$ равносильно простой замене Z на 1 в $H(Z)$ и $X(Z)$ и поэтому выполняется за 4 сложения. Приведения по модулю $Z^4+Z^3+Z^2+Z+1$ также выполняются за 4 сложения путем вычитания члена четвертой степени из всех остальных членов входных полиномов, так как $Z^4\equiv -1-Z-Z^2-Z^3$. Умножение по модулю $(Z-1)$ есть просто скалярное умножение, а умножение по модулю $(Z^4+Z^3+Z^2+Z+1)$ выполняется за 7 умножений при использовании теоремы 3.1. Восстановление по китайской теореме об остатках может рассматриваться как обращение приведений и выполняется за 8 сложений. Таким образом, 5-точечная циклическая свертка с учетом приведений и восстановления по китайской теореме об остатках вычисляется только за 8 умножений и ограниченное число сложений.

3.2. Использование полиномиальных преобразований для вычисления двумерной свертки

Непосредственное выполнение двумерной нерекурсивной фильтрации массива $N\times N$ требует вычисления числа операций, пропорционального N^2M^2 (где $M\times M$ — размеры апертуры свертки) и, таким образом, большой вычислительной мощности даже для относительно малых сверток. Это обстоятельство стимулировало ряд попыток найти более эффективные методы реализации двумерных нерекурсивных цифровых фильтров.

В большинстве этих методов цифровая нерекурсивная фильтрация осуществляется путем вычисления серии циклических сверток фрагментов, полученных секционированием входных последовательностей и дополнением их соответствующим числом нулей. Окончательный результат цифровой фильтрации при этом получается методом перекрытия с суммированием или перекрытия с накоплением [3.10]. При таком подходе объем вычислений определяется объемом вычислений циклических сверток, которые обычно сильно упрощаются при использовании алгоритмов быстрых преобразований Фурье (БПФ) [3.12], теоретико-числовых преобразований (ТЧП) [3.8, 3.9] или метода Агарвала — Кули [3.2]. По сравнению с прямым вычислением эти методы позволяют существенно снизить число операций, но, вообще говоря, для вычисления двумерных сверток

они не являются оптимальными. Вычисление с помощью БПФ связано с операциями над комплексными числами, значительными ошибками округления, требует определенных средств для вычисления тригонометрических функций и по-прежнему большого числа умножений. ТЧП могут быть вычислены без умножений и без ошибок округления при вычислении свертки. Однако эти преобразования строго ограничены по длине слова, длине преобразуемой последовательности и используют модулярную арифметику, которая обычно в универсальных ЭВМ реализуется недостаточно эффективно. Метод Агарвала — Кули, основанный на вложении нескольких коротких сверток, привлекателен для одномерных сверток, так как он не использует модулярной арифметики и не требует каких-либо манипуляций с тригонометрическими функциями. К сожалению, этот метод не очень эффективен для двумерных сверток.

Покажем, что с помощью полиномиальных преобразований [3.3, 3.5] для отображения двумерных сверток в одномерные свертки и полиномиальные произведения можно значительно упростить вычисление двумерных сверток. Полиномиальные преобразования определены на полях полиномов и обладают свойством циклической свертки. Эти преобразования вычисляются в обычной арифметике без умножений и в сочетании с эффективными алгоритмами короткой свертки и полиномиального произведения позволяют минимизировать объем вычислений двумерных сверток.

3.2.1. Полиномиальные преобразования

Пусть $y_{u,l}$ — двумерная $N \times N$-точечная циклическая свертка

$$y_{u,l} = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} h_{n,m}\, x_{u-n,\,l-m},\quad u, l = 0, \ldots, N-1. \tag{3.15}$$

Чтобы сделать основные упрощения при вычислении этой свертки, необходимо представить ее в терминах полиномиальной алгебры. Такое представление выполнено с учетом того, что (3.15) можно рассматривать как одномерную полиномиальную свертку:

$$Y_l(Z) \equiv \sum_{m=0}^{N-1} H_m(Z)\, X_{l-m}(Z) \bmod (Z^N - 1), \tag{3.16}$$

$$H_m(Z) = \sum_{n=0}^{N-1} h_{n,m} Z^n, \quad m = 0, \ldots, N-1, \tag{3.17}$$

$$X_r(Z) = \sum_{s=0}^{N-1} x_{s,r} Z^s, \quad r = 0, \ldots, N-1, \tag{3.18}$$

где $y_{u,l}$ получено из N полиномов $Y_l(Z)$ путем выбора коэффициентов при Z^u в $Y_l(Z)$:

$$Y_l(Z) = \sum_{u=0}^{N-1} y_{u,l} Z^u, \quad l = 0, \ldots, N-1. \tag{3.19}$$

Предположим сперва, что N — нечетное простое число, $N = q$. В этом случае, как показано в разд. 3.1, $(Z^q - 1)$ является произведением двух циклотомических полиномов:

$$Z^q - 1 = (Z-1) P(Z), \tag{3.20}$$

$$P(Z) = Z^{q-1} + Z^{q-2} + \ldots + 1. \tag{3.21}$$

Поскольку $Y_l(Z)$ определен по модулю (Z^q-1), он может быть вычислен посредством приведения $H_m(Z)$ и $X_r(Z)$ по модулю $(Z-1)$ и $P(Z)$, множителей (Z^q-1), вычисления полиномиальных сверток $Y_{1,l}(Z)\equiv Y_l(Z) \bmod P(Z)$ и $Y_{2,l}\equiv Y_l(Z) \bmod (Z-1)$ над приведенными полиномами и последующего восстановления $Y_l(Z)$ в соответствии с китайской теоремой об остатках:

$$Y_l(Z) \equiv S_1(Z)\, Y_{1,l}(Z) + S_2(Z)\, Y_{2,l} \bmod (Z^q-1), \tag{3.22}$$

$$\begin{cases} S_1(Z) \equiv 1, & S_2(Z) \equiv 0 \bmod P(Z); \\ S_1(Z) \equiv 0, & S_2(Z) \equiv 1 \bmod (Z-1), \end{cases} \tag{3.23}$$

где

$$S_1(Z) = [q - P(Z)]/q, \tag{3.24}$$

$$S_2(Z) = P(Z)/q. \tag{3.25}$$

Задача вычисления $Y_l(Z)$, следовательно, сводится к более простой задаче вычисления $Y_{1,l}(Z)$ и $Y_{2,l}$. Вычисление $Y_{2,l}$ является чрезвычайно простым, так как $Y_{2,l}$ определено по модулю $(Z-1)$. Таким образом, $Y_{2,l}$ есть сверточное произведение скаляров $H_{2,m}$ и $X_{2,r}$, полученных подстановкой 1 вместо Z в $H_m(Z)$ и $X_r(Z)$:

$$Y_{2,l} = \sum_{m=0}^{q-1} H_{2,m} X_{2,l-m}, \quad l = 0, \ldots, q-1 \tag{3.26}$$

$$H_{2,m} = \sum_{n=0}^{q-1} h_{n,m} \quad X_{2,r} = \sum_{s=0}^{q-1} x_{s,r}. \tag{3.27}$$

Наиболее трудной частью вычисления $Y_l(Z)$ является определение $Y_{1,l}(Z)$. Для его упрощения введем преобразование $\bar{H}_k(Z)$, определенное по модулю $P(Z)$:

$$\bar{H}_k(Z) \equiv \sum_{m=0}^{q-1} H_{1,m}(Z)\, Z^{mk} \bmod P(Z), \tag{3.28}$$

где $H_{1,m}(Z) \equiv H_m(Z) \bmod P(Z)$, $k = 0, \ldots, q-1$. Будем называть это преобразование (которое имеет ту же структуру, что и ДПФ, но с заменой комплексных экспонент на Z) *полиномиальным*. Аналогично определим обратное преобразование

$$H_{1,l}(Z) \equiv \frac{1}{q} \sum_{k=0}^{q-1} \bar{H}_k(Z)\, Z^{-lk} \bmod P(Z), \; l = 0, \ldots, q-1. \tag{3.29}$$

Полиномиальные преобразования вычисляются с помощью умножений на степени Z и сложений. Полиномиальные сложения выполняются путем сложения отдельных чисел, соответствующих каждому коэффициенту Z. Умножения на степени Z можно упростить, если иметь в виду, что $Z^q \equiv 1 \bmod P(Z)$. Тогда $H_{1,m}(Z) Z^{mk} \bmod P(Z) \equiv [H_{1,m}(Z) Z^{mk} \bmod (Z^q-1)] \bmod P(Z)$. Умножение $H_{1m}(Z)$ на $Z^{mk} \bmod (Z^q-1)$ является, следовательно, простым циклическим сдвигом на $[(mk) \bmod q]$ точек элементов $(h_{0,m} - h_{q-1,m})$, $(h_{1,m} - h_{q-1,m}), \ldots, (h_{q-2,m} - h_{q-1,m})$, 0 от $H_{1,m}(Z)$, составляющих q полиномиальных элементов. Поскольку $Z^{q-1} \equiv -Z^{q-2} - Z^{q-3} \ldots - 1 \bmod P(Z)$, приведение по модулю $P(Z)$ выполняется вычитанием коэффициента при Z^{q-1} из всех других коэффициентов Z. Таким образом, полиномиальные преобразования вычисляются с помощью простых

сложений, без каких-либо ограничений на арифметику, поскольку поле F коэффициентов может быть выбрано произвольным.

Покажем теперь, что полиномиальные преобразования, наряду с ДПФ и ТЧП, обладают свойством циклической свертки, и, следовательно, могут быть использованы для упрощения вычисления свертки. Это можно сделать, найдя преобразования $\bar{H}_k(Z)$ и $\bar{X}_k(Z)$ от $H_{l,m}(Z)$ и $X_{l,r}(Z)$ с помощью (3.28), умножив $\bar{H}_k(Z)$ на $\bar{X}_k(Z)$ по модулю $P(Z)$ и вычислив обратное преобразование $Q_l(Z)$ от $\bar{H}_k(Z)\bar{X}_k(Z)$:

$$Q_l(Z) \equiv \sum_{m=0}^{q-1} \sum_{r=0}^{q-1} H_{l,m}(Z)\, X_{l,r}(Z)\, \frac{1}{q} \sum_{k=0}^{q-1} Z^{pk} \bmod P(Z), \tag{3.30}$$

где $p=m+r-l$. Пусть $S=\sum_{k=0}^{q-1} Z^{pk}$. При $p \not\equiv 0 \bmod q$ последовательность экспонент от $pk \bmod q$ представляет собой простую перестановку целых чисел 0, 1, ..., $q-1$. Следовательно, $S \equiv \sum_{k=0}^{q-1} Z^k \equiv P(Z) \equiv 0 \bmod P(Z)$. Для $p \equiv 0 \bmod q$ $S=q$. Поэтому единственный ненулевой случай соответствует $p=0$ или $r \equiv l-m \bmod q$ и $Q_l(Z)$ приводится к циклической полиномиальной свертке

$$Q_l(Z) \equiv Y_{l,l}(Z) \equiv \sum_{m=0}^{q-1} H_{l,m}(Z)\, X_{l,l-m}(Z) \bmod P(Z). \tag{3.31}$$

При этих условиях $Y_{l,l}(Z)$ вычисляется с помощью трех полиномиальных преобразований и q полиномиальных умножений $\bar{H}_k(Z)\bar{X}_k(Z)$, определенных по модулю $P(Z)$. В большинстве применений цифровой фильтрации одна из входных последовательностей является постоянной. Ее преобразование можно вычислить заранее, и для этого потребуется только два полиномиальных преобразования. В этом случае можно также упростить восстановление по китайской теореме об остатках, учитывая, что согласно (3.10):

$$S_1(Z) \equiv (Z-1)/[(Z-1) \bmod P(Z)] \equiv (Z-1)\, T_1(Z),$$
$$T_1(Z) = [-Z^{q-2} - 2Z^{q-3} - \ldots - (q-3)Z^2 - (q-2)Z + 1 - q]/q. \tag{3.32}$$

Поскольку $T_1(Z)$ определено по модулю $P(Z)$, то предварительное умножение на $T_1(Z)$ может быть выполнено перед восстановлением по китайской теореме об остатках и объединено с предварительным вычислением преобразования постоянной последовательности. Точно так же предварительное умножение на $1/q$ можно объединить с вычислением скалярной свертки $Y_{2,l}$ в (3.26), в силу чего восстановление по китайской теореме об остатках, заданное в (3.22), сводится к

$$Y_l(Z) \equiv (Z-1)\, Y_{1,l}(Z) + (Z^{q-1} + Z^{q-2} + \ldots + 1)\, Y_{2,l} \bmod (Z^q - 1). \tag{3.33}$$

При этих условиях $(q \times q)$-точечная свертка вычисляется по схеме, показанной на рис. 3.1. Можно видеть, что единственные умножения, которые требуются для вычисления $y_{u,l}$, необходимы при вычислении одной q-точечной свертки и q полиномиальных произведений по модулю $P(Z)$. Это означает, что если свертка и полиномиальные произведения вычисляются по алгоритму с минимальным числом умножений, определенным согласно теоремам 3.2 и 3.3, то $(q \times q)$-точечная свертка, q — простое, вычисляется только за $2q^2-q-2$ умно-

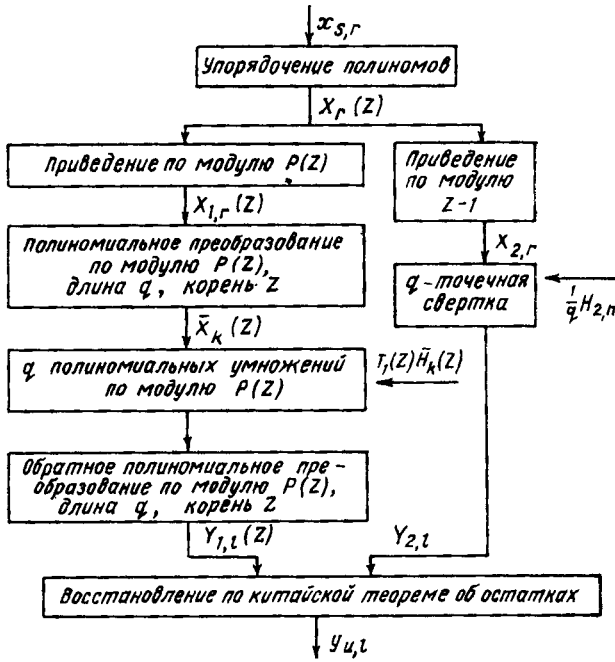


Рис. 3.1. Вычисление двумерной $(N\times N)$-точечной свертки с помощью полиномиальных преобразоваиий, N — простое

жений. Можно в действительности показать, что это является теоретически минимальным числом умножений для $(q\times q)$-точечной свертки при простом q [3.13].

3.2.2. Составные полиномиальные преобразования

Для простоты мы до сих пор ограничивались $(N\times N)$-точечными свертками, $N=q$, q — простое. На практике метод полиномиального преобразования не ограничен случаем простого N. Из (3.28) — (3.31) ясно, что любое полиномиальное преобразование, имеющее корень aZ^b и определенное по модулю полинома $P_{e_i}(Z)$, будет обладать свойством циклической свертки при условии

$$(aZ^b)^N \equiv 1 \bmod P_{e_i}(Z)\,, \tag{3.34}$$

$$S \equiv \begin{cases} 0 \bmod P_{e_i}(Z) \text{ для } p \not\equiv 0 \bmod N, \\ N \bmod P_{e_i}(Z) \text{ для } p \equiv 0 \bmod N, \end{cases} \tag{3.35}$$

когда $S=\sum_{k=0}^{N-1}(aZ^d)^{pk}$. Поскольку для вычисления двумерных сверток здесь используются полиномиальные преобразования, $P_{e_i}(Z)$ является циклотомическим полиномом, множителем (Z^N-1), т. е.

$$Z^N - 1 = \prod_{i=1}^{d} P_{e_i}(Z), \tag{3.36}$$

где степень каждого полинома $P_{e_i}(Z)$ есть $\varphi(e_i)$ — фуикция Эйлера [3.7]. Покажем сначала, что всегда существует полиномиальное N-точечное преобразование с корием Z, обладающее свойством циклической свертки, если оно определено по модулю $P_{e_d}(Z)$, являющемуся наибольшим циклотомическим полиномом, множителем (Z^N-1). Это можно увидеть, если учесть, что, поскольку $Z^N \equiv 1 \bmod (Z^N-1)$ и P_{e_d} является множителем (Z^N-1), то $Z^N \equiv 1 \bmod P_{e_d}(Z)$. Отсюда вытекает (3.34). В условии (3.35) $p \equiv 0 \bmod N$, так как $S \equiv \sum_{k=0}^{N-1} Z^{pk} = N$.

Для $p \not\equiv 0 \bmod N$, N — взаимно-простое, $S \equiv \sum_{k=0}^{N-1} Z^k \bmod (Z^N-1)$, откуда вытекает, что $S = \prod_{i=2}^{d} P_{e_i}(Z) \equiv 0 \bmod P_{e_d}(Z)$. Для $p \not\equiv 0 \bmod N$ и невзаимно-простого с N p всегда можно рассматривать, без потери общности, как множитель N. Тогда $S \equiv p \sum_{k=0}^{N/p-1} Z^{pk} = p(Z^N-1)/(Z^p-1)$. Наибольший полиномиальный множитель (Z^N-1) есть полином $P_{e_d}(Z)$ степени $\varphi(N)$. Наибольший полиномиальный множитель (Z^p-1) является цнклотомическим полиномом $Q(Z)$ степени $\varphi(p)$. $Q(Z)$ отличается от $P_{e_d}(Z)$, поскольку $\varphi(p) < \varphi(N)$, и он не может быть множителем для $P_{e_d}(Z)$, так как $P_{e_d}(Z)$ — неприводимый полином. Таким образом, $Z^p-1 \not\equiv 0 \bmod P_{e_d}(Z)$, а $S \equiv 0 \bmod P_{e_d}(Z)$. Тем самым соотношение (3.35) доказано.

При этих условиях $(N \times N)$-точечная свертка $y_{u,l}$ вычисляется при помощи преобразования входной последовательности N полиномов по N членов, приведенных по модулю $P_{e_d}(Z)$ и по модулю $\prod_{i=1}^{d-1} P_{e_i}(Z)$. Выходные отсчеты $y_{u,l}$ получаются путем восстановления по китайской теореме об остатках из полиномиальной свертки $Y_{1,l}(Z) \bmod P_{e_d}(Z)$ и полиномиальной свертки $Y_{2,l}(Z) \bmod \prod_{i=1}^{d-1} P_{e_i}(Z)$. $Y_{1,l}(Z)$ вычисляется с помощью N-точечных полиномиальных преобразований. Поскольку степень $P_{e_d}(Z)$ есть $\varphi(N)$, степень $\prod_{i=1}^{d-1} P_{e_i}(Z)$ равна $N-\varphi(N)$ и $Y_{2,l}(Z)$ можно рассматривать как обобщеиную скаляриую $[N \times (N-\varphi(N))]$-точечную свертку. Такой же метод можно использовать рекурсивно для вычисления $Y_{2,l}(Z)$ с помощью полиномиальных преобразований.

Необходимо также заметить, что когда N_1-точечное полиномиальное преобразование (корень aZ^b, N_1 — нечетное) обладает свойством свертки, всегда можно увеличить длину преобразуемой последовательности до N_1N_2, где $N_2 = 2^t$. В этом случае корнем нового преобразования будет aWZ^b, где $W = \exp(-j2\pi/N_2)$, $j = \sqrt{-1}$. Это свойство является прямым следствием того, что W — корень из единицы порядка N_2 и aZ^b — кореиь из единицы порядка N_1, где N_1 и N_2 — взаимио-простые. Оно чрезвычайно полезио для вычисления $(2N_1 \times 2N_1)$- и $(4N_1 \times 4N_1)$-точечных сверток, так как в этих случаях $W = -1$ или $-j$ и полиномиальные преобразования по-прежнему вычисляются без умножений.

Для составного N наиболее интересны случаи, когда N — степень простого числа. На рис. 3.2 показан алгоритм вычисления $(q^2 \times q^2)$-точечной свертки с помощью полиномиальных преобразований, q — простое нечетное. В этом примере $(q^2 \times q^2)$-точечная свертка отображена без умноженнй в q^2+q полиномиальных

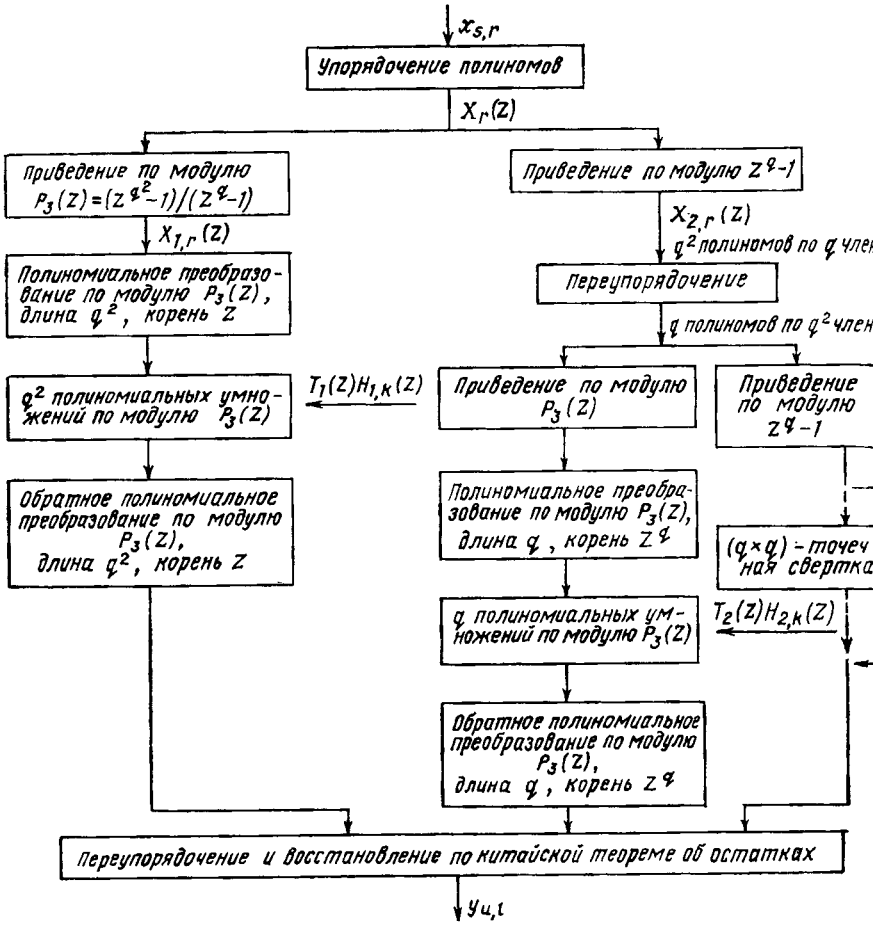


Рис. 3.2. Вычисление двумерной $(N\times N)$-точечной свертки с помощью полиномиальных преобразований $N=q^2$, q — простое

произведений по модулю $P_3(Z)=(Z^{q^2}-1)/Z^q-1=Z^{q(q-1)}+Z^{q(q-2)}+\ \ldots\ +1$ и в $(q\times q)$-точечную свертку. Последнюю свертку саму можно отобразить без умножений в q полиномиальных произведений по модулю $P_2(Z)=Z^{q-1}+Z^{q-2}+\ \ldots\ \ldots\ +1$ и одну q-точечную свертку с помощью метода, представленного на рис. 3.1. Первый шаг вычисления $(2^t\times 2^t)$-точечной свертки с помощью полиномиальных преобразований показан также на рис. 3.3. В этом случае вычисление выполняется за $(t-1)$ шагов с полиномиальными преобразованиями, определенными по модулю $P_{t+1}(Z)=Z^{2t-1}+1$, $P_t(Z)=Z^{2t-2}+1, \ldots, P_3(Z)=Z^2+1$. Эти полиномиальные преобразования особенно интересны потому, что длины преобразуемых последовательностей являются степенями двойки и поэтому они могут быть вычислены с уменьшенным числом сложений с помощью алгоритма типа БПФ по основанию 2. В табл. 3.1 сведены основные условия для полиноми-

Таблица 3.1

Полиномиальные преобразования для вычисления сверток
(q, q_1, q_2 — простые)

Преобразование по кольцу $P(Z)$	Преобразование: длина	Преобразование: корень	Размеры массива $N \times N$	Число сложений для приведений полиномиальных преобразований и восстановления по китайской теореме об остатках	Полиномиальные произведения и свертки
$(Z^q-1)/(Z-1)$	q	Z	$q \times q$	$2(q^3+q^2-5q+4)$	q произведений $P(Z)$, свертка $2q$
	$2q$	$-Z$	$2q \times q$	$4(q^3+2q^2-6q+4)$	$2q$ произведений $P(Z)$, свертка $2q$
$(Z^{2q}-1)/(Z^2-1)$	$2q$	$-Z^{q+1}$	$2q \times 2q$	$8(q^3+2q^2-6q+4)$	$2q$ произведений $P(Z)$, свертка $2 \times 2q$
$(Z^{q^2}-1)/(Z^q-1)$	q	Z^q	$q \times q^2$	$2(q^4+2q^3-4q^2-q+4)$	q произведений $P(Z)$, q произведений $(Z^q-1)/(Z-1)$, свертка q
	q^2	Z	$q^2 \times q^2$	$2(2q^5+q^4-5q^3+q^2+6)$	$q(q+1)$ произведений $P(Z)$, q произведений $(Z^q-1)/(Z-1)$, свертка q
$(Z^{2q^2}-1)/(Z^{2q}-1)$	$2q^2$	$-Z^{q^2+1}$	$2q^2 \times 2q^2$	$8(2q^5+2q^4-6q^3-q^2+5q+2)$	$2q(q+1)$ произведений $P(Z)$, $2q$ произведений $(Z^{2q}-1)/(Z^2-1)$, свертка $2 \times 2q$
$(Z^{q_1q_2}-1)/(Z^{q_2}-1)$	q_1	Z^{q_2}	$q_1 \times q_1q_2$	$2q_2(q^3_1+q^2_1-5q_1+4)$	q_1 произведений $P(Z)$, свертка $q_1 \times q_2$
$Z^{2^{t-1}}+1$	$2t$	Z	$2^t \times 2^t$	$2^{2t-1}(3t+5)$	$3 \cdot 2^{t-1}$ произведений $P(Z)$, свертка $2^{t-1} \times 2^{t-1}$

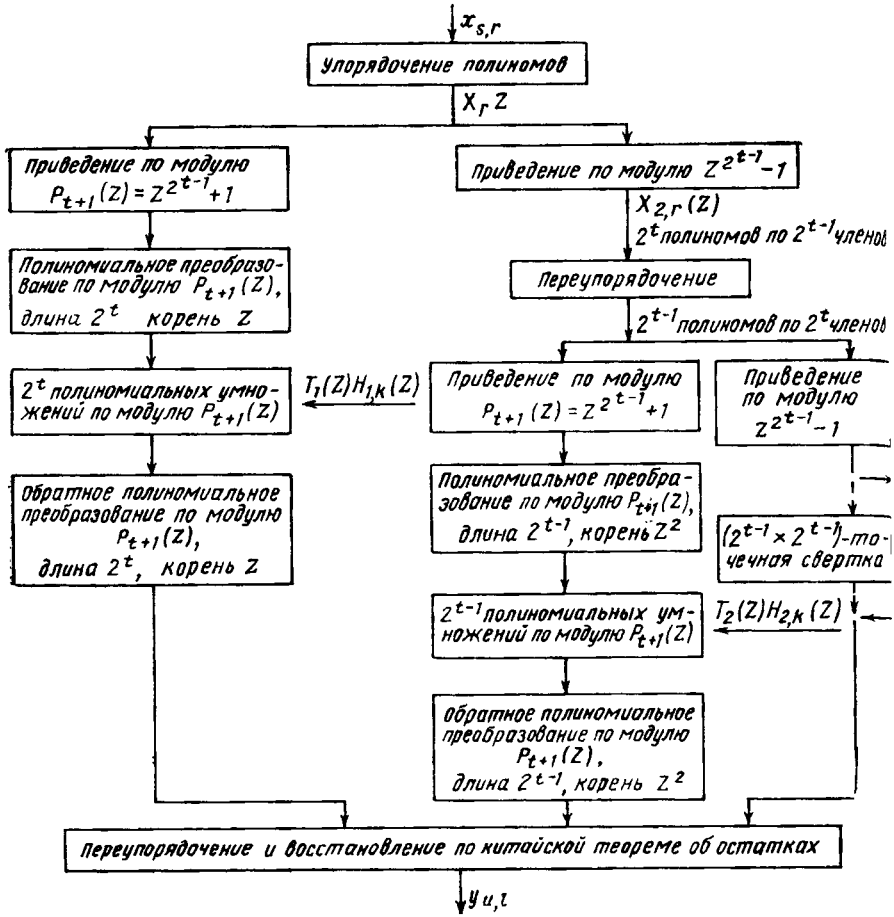


Рис. 3.3. Вычисление двумерной $(N \times N)$-точечной свертки с помощью полиномиальных преобразований, $N = 2^t$

альных преобразований, которые можно вычислить без умножений и которые соответствуют таким двумерным массивам, размеры которых имеют общий множитель.

3.2.3. Вычисление полиномиальных преобразований и приведений полиномов

Вычисление двумерных сверток с помощью полиномиальных преобразований включает вычисление приведений, восстановление по китайской теореме об остатках и полиномиальные преобразования. Для наиболее интересных случаев, собранных в табл. 3.1, определим число сложений, требуемых при выполнении этих операций.

Когда $N = q$, q — простое, входные последовательности должны быть приведены по модулю $Z-1$ и по модулю $P(Z) = Z^{q-1} + Z^{q-2} + \ldots + 1$. Каждая опе-

рация выполняется очень просто за $q(q-1)$ сложений, путем суммирования всех членов входной последовательности для приведения по модулю $(Z-1)$, как показано в (3.27) и с учетом того, что в результате приведений по модулю $P(Z)$ образуются N полиномов $H_{1,m}(Z)$:

$$H_{1,m}(Z) = \sum_{n=0}^{q-2} (h_{n,m} - h_{q-1,m}) Z^n, \quad m = 0, \ldots, q-1. \tag{3.37}$$

Восстановление по китайской теореме об остатках выполняется за $2q(q-1)$ сложений с использованием (3.33). Поскольку $Y_{1,l}(z)$ представляет собой полином из $q-1$ членов, $Y_{1,l}(Z)$ может быть определен как

$$Y_{1,l}(Z) = \sum_{u=0}^{q-2} y_{1,u,l} Z^u. \tag{3.38}$$

Таким образом, (3.33) переходит в

$$Y_l(Z) = Y_{2,l} - y_{1,0,l} + \sum_{u=1}^{q-2} (y_{1,u-1,l} - y_{1,u,l} + Y_{2,l}) Z^u + \\ + (y_{1,q-2,l} + Y_{2,l}) Z^{q-1}, \quad l = 0, \ldots, q-1. \tag{3.39}$$

Для $N=q^2$, q — простое, или для $N=2^t$ приведения и восстановление по китайской теореме об остатках выполняются аналогичным образом. В табл. 3.2 в третьей колонке приведено соответствующее число сложений.

Для $N=q$, q — простое, полиномиальное преобразование $H_k(Z)$, определенное с помощью (3.28), вычисляется следующим образом:

$$\overline{H}_k(Z) \equiv H_{1,0}(Z) + \sum_{m=1}^{q-1} H_{1,m}(Z) Z^{mk} \bmod P(Z), \quad k = 0, \ldots, q-2; \tag{3.40}$$

$$\overline{H}_{q-1}(Z) \equiv H_{1,0}(Z) + \sum_{m=1}^{q-1} H_{1,m}(Z) Z^{m(q-1)} \bmod P(Z), \tag{3.41}$$

Таблица 3.2

Число сложений для вычисления приведений, операций восстановления по китайской теореме об остатках и полиномиальных преобразований (q, q_1, q_2 — нечетные простые)

Длина последовательности	Кольцо	Число сложений	
		для приведений и восстановления по китайской теореме об остатках	для полиномиальных преобразований
q	$(Z^q-1)/(Z-1)$	$4q(q-1)$	q^3-q^2-3q+4
$2q$	$(Z^q-1)/(Z-1)$	$8q(q-1)$	$2(q^3-4q+4)$
$2q$	$(Z^{2q}-1)/(Z^2-1)$	$16q(q-1)$	$4(q^3-4q+4)$
q	$(Z^{q^2}-1)/(Z^q-1)$	$4q^2(q-1)$	$q(q^3-q^2-3q+4)$
q^2	$(Z^{q^2}-1)/(Z^q-1)$	$4q^3(q-1)$	$2q^5-2q^4-5q^3+5q^2+q+2$
$2q$	$(Z^{2q^2}-1)/(Z^{2q}-1)$	$16q^2(q-1)$	$4q(q^3-4q+4)$
$2q^2$	$(Z^{2q^2}-1)/(Z^{2q}-1)$	$16q^3(q-1)$	$4(2q^5-q^4-6q^3+5q^2+q+2)$
2^t	$Z^{2^{t-1}}+1$	2^{2t+1}	$t2^{2t-1}$

где $H_{1,m}(Z)$ — полиномы из $q-1$ членов. Пусть $R_k(Z) \equiv \sum_{m=1}^{q-1} H_{1,m}(Z) Z^{mk}$. Так как $R_0(Z) = \sum_{m=1}^{q-1} H_{1,m}(Z)$, то $R_0(Z)$ вычисляется за $(q-2)(q-1)$ сложений. При $k \neq 0$ $R_k(Z)$ представляет собой сумму $q-1$ полиномов, каждый из которых умножен на степень Z. $R_k(Z)$ сначала вычисляется по модулю Z^q-1. Поскольку $Z^q \equiv 1$, каждый полином $H_{1,m}(Z)$ после умножения на степень Z становится полиномом из q членов, где один член равен нулю, т. е. вычисление $R_k(Z) \bmod (Z^q-1)$ требует q^2-3q+1 сложений. Затем $R_k(Z)$ приводится по модулю $P(Z)$ с помощью $q-1$ сложений. Таким образом, вычисление каждого $R_k(Z)$ при $k \neq 0$ требует q^2-2q сложений. Поскольку $\sum_{k=0}^{q-2} Z^{mk} \equiv -Z^{m(q-1)} \bmod P(Z)$, для $m \neq 0$,

$$\sum_{k=0}^{q-2} \sum_{m=1}^{q-1} H_{1,m}(Z) Z^{mk} \equiv -\sum_{m=1}^{q-1} H_{1,m}(Z) Z^{m(q-1)} \bmod P(Z) \tag{3.42}$$

и $\sum_{m=1}^{q-1} H_{1,m}(Z) Z^{m(q-1)}$ вычисляется за $(q-1)(q-2)$ сложений при помощи соотношения:

$$\sum_{m=1}^{q-1} H_{1,m}(Z) Z^{m(q-1)} \equiv -\sum_{k=0}^{q-2} R_k(Z) \bmod P(Z). \tag{3.43}$$

Окончательно $H_k(Z)$ получается сложением $H_{1,0}(Z)$ с $R_k(Z)$ и $\sum_{m=1}^{q-1} H_{1,m}(Z) Z^{m(q-1)}$ за $q(q-1)$ сложений. Таким образом, для вычисления полиномиального преобразования из q членов, q — простое, требуется общее число сложений q^3-q^2-3q+4.

Полиномиальные преобразования для составного N вычисляются с уменьшенным числом сложений с помощью алгоритма типа БПФ. Если, например, $N=N_1N_2$, полиномиальное преобразование $H_k(Z)$ последовательности N элементов вычисляется с помощью N_1 N_2-точечных преобразований, N_2 N_1-точечных преобразований и умножения на фазовые множители $Z^{m_2k_2}$:

$$\overline{H}_{N_2k_1+k_2}(Z) \equiv \sum_{m_2=0}^{N_1-1} Z^{N_2m_2k_1} \times$$

$$\times Z^{m_2k_2} \sum_{m_1=0}^{N_2-1} H_{1,N_1m_1+m_2}(Z) Z^{N_1m_1k_2}, \; k_1 = 0, \ldots, N_1-1; \; k_2 = 0, \ldots, N_2-1. \tag{3.44}$$

В четвертой колонке табл. 3.2 приведено число сложений, требуемых для различных полиномиальных преобразований. В табл. 3.1 в четвертой колонке также дано общее число сложений для приведений, полиномиальных преобразований и восстановления по китайской теореме об остатках для различных двумерных сверток, вычисляемых с помощью полиномиальных преобразований.

3.2.4. Вычисление полиномиальных произведений и одномерных сверток

В предыдущих разделах показано, что двумерные свертки можно эффективно вычислять, отображая с помощью полиномиальных преобразований двумерные свертки в одномерные полиномиальные произведения и свертки. Если полиномиальные преобразования выбраны правильно, отображение выполняется без умножений и с ограниченным числом сложений. Таким образом, при вычислении двумерных сверток с помощью полиномиальных преобразований сложность вычислений сильно зависит от эффективности алгоритмов, использованных для вычисления полиномиальных произведений и одномерных сверток.

Первый метод вычисления одномерных сверток и полиномиальных произведений состоит в применении ДПФ или ТЧП. Так как эти преобразования обладают свойством свертки, с их помощью можно непосредственно вычислять одномерные свертки. Эти преобразования можно использовать также для вычисления полиномиальных произведений по модулю $P_{e_i}(Z)$ с учетом того, что, поскольку $P_{e_i}(Z)$, определенный выражением (3.36), является множителем для Z^N-1, все вычисления можно выполнять по модулю Z^N-1 с приведением на последнем этапе по модулю $P_{e_i}(Z)$. В соответствии с этим методом вычисление полиномиального произведения по модулю $P_{e_i}(Z)$ заменяется вычислением полиномиального произведения по модулю Z^N-1, которое является N-точечной сверткой. Тогда вычисление полиномиальных произведений по модулю $P_{e_i}(Z)$, $P_{e_i}(Z)$ — множитель (Z^N-1), можно выполнить с помощью N-точечной ДПФ или ТЧП.

Этот метод можно пояснить на следующем простом примере. Предположим, что два входных полинома $X(Z)=x_0+x_1Z$ и $H(Z)=h_0+h_1Z$ нужно перемножить по модулю $P(Z)=(Z^3-1)/(Z-1)=Z^2+Z+1$. Для этого сначала нужно найти 3-точечную циклическую свертку $Y(Z)$ двух входных последовательностей при

$$Y(Z)=y_0+y_1Z+y_2Z^2, \tag{3.45}$$

$$\begin{cases} y_0=h_0x_0, \\ y_1=h_0x_1+h_1x_0, \\ y_2=h_1x_1. \end{cases} \tag{3.46}$$

Полиномиальное произведение $Y_1(Z)$, определенное по модулю $P(Z)$, задается тогда выражением $Y_1(Z)\equiv\{[H(Z)X(Z)]\bmod(Z^N-1)\}\bmod P(Z)$, которое соответствует простому приведению $Y(Z)\bmod P(Z)$. Поскольку $Z^2\equiv -Z-1$, то это приведение достигается простым вычитанием y_2 из y_0 и y_1, причем

$$Y_1(Z)=y_{1,0}+y_{1,1}Z\equiv Y(Z)\bmod P(Z), \tag{3.47}$$

$$\begin{cases} y_{1,0}=h_0x_0-h_1x_1, \\ y_{1,1}=h_0x_1+h_1x_0-h_1x_1. \end{cases} \tag{3.48}$$

Схема вычислений по этому методу приведена на рис. 3.4 для $(q\times q)$-точечной свертки, q — простое число. В этом случае с помощью метода, приведенного на рис. 3.1, двумерная свертка вместо одной q-точечной свертки плюс q полиномиальных произведений по модулю $(Z^q-1)/(Z-1)$ отображается в $(q+1)$ q-точечные свертки.

Таким образом, одномерные свертки и полиномиальные произведения могут быть вычислены с помощью ДПФ или ТЧП. Трудности, связанные с использо-

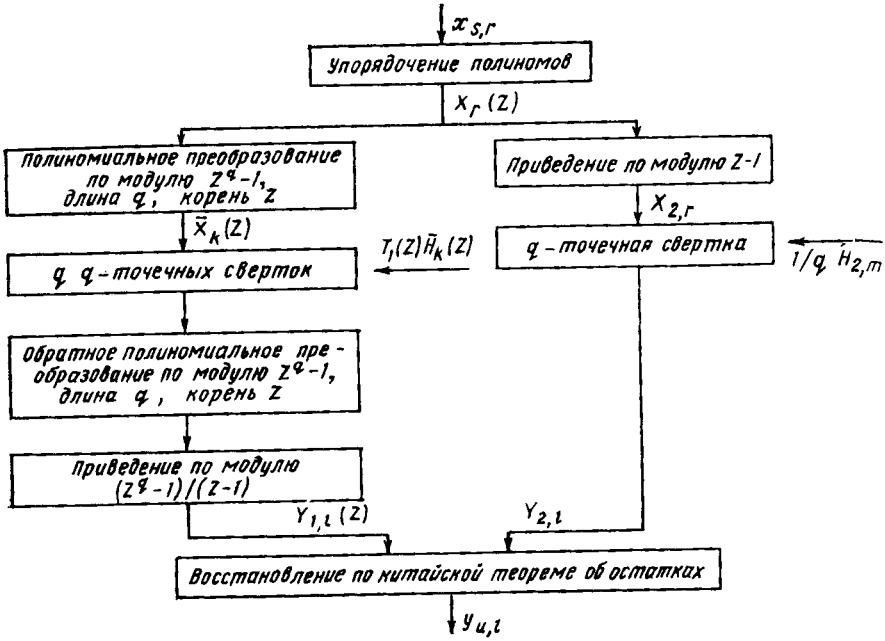


Рис. 3.4. Вычисление $(q\times q)$-точечной свертки с помощью полиномиальных преобразований, q — простое. Полиномиальные произведения по модулю $(Z^N-1)/(Z-1)$ заменяются N-точечными свертками

ванием этих преобразований, такие, как ошибки округления для ДПФ или модулярная арифметика для ТЧП, относятся теперь только к части общего вычислительного процесса. Однако в разд. 3.1 мы видели, что методы, основанные на интерполяции и китайской теореме об остатках, порождают для коротких сверток и полиномиальных произведений более эффективные алгоритмы, чем алгоритмы их вычислений с помощью ДПФ или ТЧП. Следовательно, такие алгоритмы часто выгодно применять в комбинации с полиномиальными преобразованиями. Мы увидим, что эти алгоритмы нельзя получать систематически из общего метода, как в случае, например, алгоритмов БПФ. Таким образом, каждый алгоритм короткой свертки или полиномиального произведения в данном применении нужно программировать специально и для сокращения размеров программы желательно использовать только ограниченное число различных алгоритмов. Одним из таких способов является вычисление коротких сверток как полиномиальных произведений с помощью китайской теоремы об остатках. В случае двумерной $(q\times q)$-точечной свертки, q — простое, метод полиномиального преобразования требует вычисления q полиномиальных произведений по модулю $P(Z)=(Z^q-1)/(Z-1)$ и одной q-точечной циклической свертки. Однако, если q-точечная циклическая свертка находится с помощью китайской теоремы об остатках за одно умножение и одно полиномиальное произведение по модулю $P(Z)$, вычисление можно выполнить с помощью только одного алгоритма полиномиального произведения по модулю $P(Z)$. В этом случае $(q\times q)$-точечная свертка вычисляется с помощью $q+1$ полиномиальных произведений по модулю

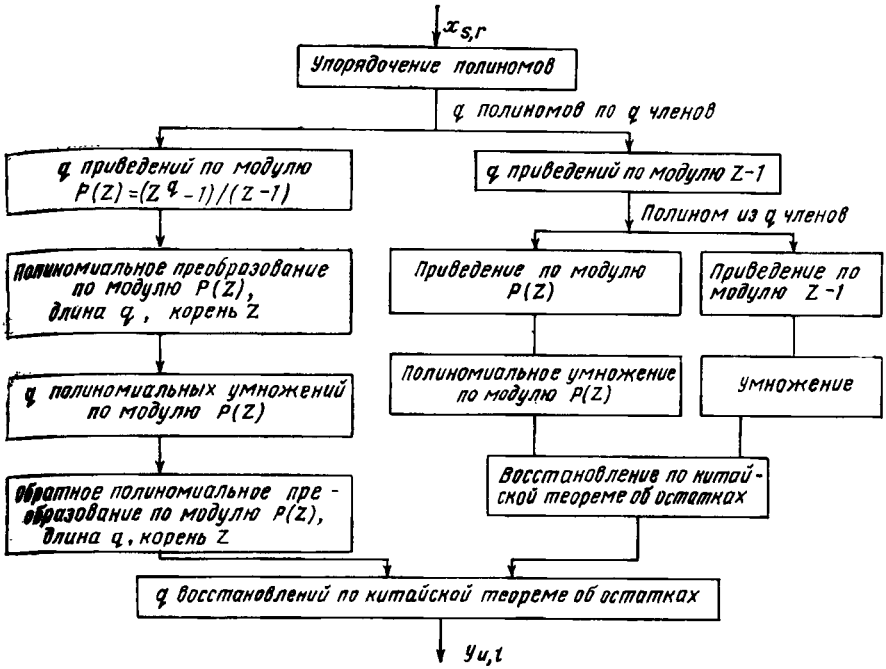


Рис. 3.5. Вычисление $(q \times q)$-точечной свертки с помощью полиномиальных преобразований, q — простое; q-точечная свертка заменяется одним умножением и одним полиномиальным произведением по модулю $P(Z)$

$P(Z)$ и одного умножения вместо q полиномиальных произведений по модулю $P(Z)$ и одной q-точечной свертки. Схема вычислений по этому методу приведена на рис. 3.5.

В дальнейшем будем предполагать, что все двумерные свертки вычисляются аналогично, так что займемся только алгоритмами полиномиального произведения. Кроме того, увидим, что в большинстве случаев свертки можно найти с помощью ограниченного набора алгоритмов полиномиальных произведений, главным образом тех, которые определены по модулю: $P(Z)=(Z^q-1)/(Z-1)$, $q=3, 5, 7$, и $P(Z)=(Z^9-1)/(Z^3-1)$ или $Z^{2^t}+1$. Поскольку эти полиномы $P(Z)$ неприводимы, их всегда можно найти с помощью интерполяции в соответствии с теоремой 3.2 за $(D-1)$ умножений, где D — степень $P(Z)$. Как показано в подразд. 3.5.1, 3.5.2, использование этого метода для умножений по модулю Z^2+1 и модулю $(Z^3-1)/(Z-1)$ порождает алгоритмы с тремя умножениями и тремя сложениями.

Для полиномиальных произведений большей размерности этот метод, обеспечивающий минимальное число умножений, приводит к чрезмерному увеличению числа сложений. Это происходит потому, что в этом случае требуется $(2D-1)$ различных полиномов $Z-a_i$. Четыре наиболее простых интерполяционных полинома имеют вид: Z, $1/Z$, $Z-1$ и $Z+1$. Таким образом, когда степень D полинома $P(Z)$ больше двух, необходимо использовать целые a_i, отличные от 0 или ± 1, а соответствующие приведения по модулю $Z-a_i$ и восстановление по китайской теореме об остатках будут включать умножения на степени a_i.

Такие операции можно выполнить либо посредством скалярных произведений, либо с помощью большого числа последовательных сложений. Это приводит к построению компромиссных алгоритмов, в которых сохранен разумный баланс между числом сложений и числом умножений. Хотя не существует общих методов построения таких алгоритмов, полезны некоторые общие принципы.

Первый подход к снижению числа сложений за счет небольшого увеличения числа умножений состоит в применении комплексных интерполяционных полиномов. Если используем, например, $Z^2+1=(Z+j)(Z-j)$, $j=\sqrt{-1}$, нам нужно 3 действительных умножения вместо двух для полинома, имеющего действительные целые корни. Однако приведения и восстановление по китайской теореме об остатках остаются простыми, так как требуют только простых сложений и умножений на ± 1 или $\pm j$. Другой подход состоит в преобразовании одномерных полиномиальных произведений в многомерные. Например, умножение по модулю $P(Z)=(Z^9-1)/(Z^3-1)=Z^6+Z^3+1$ можно заменить на умножение по модулю (Z^3-Z_1), $(Z^2_1+Z_1+1)$, так как подстановка $Z_1\equiv Z^3$ в $Z^2_1+Z_1+1$ дает $P(Z)$. В соответствии с этим методом входные полиномы $X(Z)=\sum_{n=0}^{5} x_n Z^n$ и $H(Z)=\sum_{m=0}^{5} h_m Z^m$ сначала приводятся по модулю (Z^3-Z_1) посредством подстановки $Z^3\equiv Z_1$ в $X(Z)$ и $H(Z)$. Таким образом,

$$X(Z)=X_0(Z)+ZX_1(Z)+Z^2X_2(Z), \tag{3.49}$$

$$\begin{cases} X_0(Z)=x_0+x_3Z_1, \\ X_1(Z)=x_1+x_4Z_1, \\ X_2(Z)=x_2+x_5Z_1 \end{cases} \tag{3.50}$$

и аналогичные соотношения выполняются для $H(Z)$. Теперь полиномиальное произведение по модулю Z^3-Z_1 можно вычислить по методу интерполяции за 5 умножений по модулю $(Z^2_1+Z_1+1)$. Поскольку, как показано в подразд. 3.5.2, каждое из этих полиномиальных умножений требует трех скалярных умножений, полиномиальное произведение по модулю $(Z^9-1)/(Z^3-1)$ выполняется за 15 умножений. Подробный алгоритм приведен в подразд. 3.5.5.

Аналогичный метод можно использовать для полиномиальных произведений по модулю $Z^{2^t}+1$. Полиномиальное умножение по модулю Z^4+1 выполняется, например, по модулю Z^2-Z_1, Z^2_1+1 за 9 умножений и 15 сложений (см. подразд. 3.5.3), в то время как традиционный интерполяционный метод потребовал бы 7 умножений и 41 сложение.

Все алгоритмы, построенные по этим методам, можно рассматривать как вычисления с помощью прямоугольных преобразований. Предположим, общая структура алгоритма полиномиального произведения по модулю полинома $P(Z)$ степени N, требующего M умножений, такова:

$$\mathbf{H}=\mathbf{E}\,\mathbf{h}, \tag{3.51}$$

$$X=\mathbf{F}\,\mathbf{x}, \tag{3.52}$$

$$Y=\mathbf{H}\times\mathbf{X}, \tag{3.53}$$

$$\mathbf{y}=\mathbf{G}\,\mathbf{Y}, \tag{3.54}$$

где $\mathbf{h}$ и $\mathbf{x}$ — вектор-столбцы входных последовательностей $\{h_m\}$ и $\{x_n\}$; $\mathbf{E}$ и $\mathbf{F}$ — входные матрицы $M\times N$ (знак $\times$ означает поэлементное произведение); $\mathbf{G}$ — выходная матрица $N\times M$; $\mathbf{y}$ — вектор-столбец выходной последовательности

$\{y_l\}$. Здесь Eh, Fx и GY вычисляются без умножений, тогда как H×X находится посредством M умножений. Если алгоритм построен с помощью интерполяционного метода, матрицы E и F соответствуют различным приведениям по модулю $Z-a_i$, а матрица G — восстановлению по китайской теореме об остатках. Так как восстановление по китайской теореме об остатках можно рассматривать как обратную операцию по отношению к приведению, она приблизительно эквивалентна по сложности двум приведениям. Поэтому матрица G обычно примерно в два раза сложнее матриц E и F. В большинстве практических приложений фильтрации одна из входных последовательностей $\{h_m\}$ постоянна, поэтому H можно вычислить и запомнить заранее. Таким образом, число сложений для алгоритма преимущественно зависит от сложности матриц F и G, а не матрицы E. При этих условиях для снижения числа сложений было бы чрезвычайно желательно осуществить перестановку матриц E и G. Заметим, что в соответствии с (3.50)—(3.54) y_l задается следующим образом:

$$y_l = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} h_m x_n \sum_{k=0}^{M-1} e_{m,k} f_{n,k} g_{k,l}, \tag{3.55}$$

где $e_{m,k}$, $f_{n,k}$ и $g_{k,l}$ — элементы матрицы E, F и G. Для алгоритма по модулю $P(Z)=Z^N-1$, соответствующему циклической свертке, находим обычное условие, определяющее циклическую свертку:

$$S = \sum_{k=0}^{M-1} e_{m,k} f_{n,k} g_{k,l} = \begin{cases} 1, & \text{если } m+n-l \equiv 0 \bmod N; \\ 0, & \text{если } m+n-l \not\equiv 0 \bmod N. \end{cases} \tag{3.56}$$

Аналогично алгоритмы полиномиального произведения по модулю Z^N+1 или Z^N-Z_1 удовлетворяют условиям:

$$S = \begin{cases} 1, & \text{если } m+n-l=0; \\ -1 \text{ или } Z_1, & \text{если } m+n-l=N; \\ 0, & \text{если } m+n-l \not\equiv 0 \bmod N. \end{cases} \tag{3.57}$$

Теперь заменим матрицы E и G матрицами E^1 и G^1 таким образом, чтобы их элементами были $e_{N-l,k}$ и $g_{k,N-m}$. При этих условиях S переходит в S^1 и из (3.56), очевидно, вытекает

$$S^1 = \sum_{k=0}^{M-1} e_{N-l,k} f_{n,k} g_{k,N-m} = \begin{cases} 1, & \text{если } m+n-l \equiv 0 \bmod N; \\ 0, & \text{если } m+n-l \not\equiv 0 \bmod N. \end{cases} \tag{3.58}$$

Следовательно, как указано в [3.14], свойство свертки по-прежнему сохраняется, если матрицы E и G меняются местами с помощью простого транспонирования и перегруппировки столбцов и строк. Такой же общий подход можно применить для полиномиальных произведений по модулю Z^N+1 или Z^N-Z_1. Однако в этих случаях условия для $S=1$ и $S=-1$ или Z_1 в (3.57) заменяются на $m=0$ или $l=0$. Таким образом, элементами матриц E^1 и G^1 должны быть: $g_{k,N-m}$ для $m=0$, $-g_{k,N-m}$ или $(1/Z_1)g_{k,N-m}$ для $m=0$ и $e_{N-l,k}$ для $l\neq 0$, $-e_{N-l,k}$ или $Z_1 e_{N-l,k}$ для $l=0$. Этот подход применен для описанного в подразд. 3.5.5 алгоритма полиномиального произведения по модулю $(Z^9-1)/(Z^3-1)$.

В приложении 3.5 приведены подробные алгоритмы для наиболее часто используемых полиномиальных произведений, а в табл. 3.3 — соответствующее число операций. Число арифметических операций для двумерных сверток, вычис-

Таблица 3.3

Число операций для алгоритмов полиномиальных произведений

Кольцо $P(Z)$	Степень полинома $P(Z)$	Число умножений	Число сложений
Z^2+1	2	3	3
$(Z^3-1)/(Z-1)$	2	3	3
Z^4+1	4	9	15
Z^4+1	4	7	41
$(Z^5-1)/(Z-1)$	4	9	16
$(Z^5-1)/(Z-1)$	4	7	46
$(Z^7-1)/(Z-1)$	6	15	53
$(Z^9-1)/(Z^3-1)$	6	15	39
Z^8+1	8	21	77
$(Z^2_1+Z_1+1)(Z^5_2-1)/(Z_2-1)$	8	21	83
$(Z^4_1+1)(Z^2_2+Z_2+1)$	8	21	76
$(Z^2_1+1)(Z^5_2-1)/(Z-1)$	8	21	76
$Z^{16}+1$	16	63	205
$(Z^{27}-1)/(Z^9-1)$	18	75	267
$Z^{32}+1$	32	147	739
$Z^{64}+1$	64	315	1891

Таблица 3.4

Число операций для двумерных сверток, вычисляемых с помощью полиномиальных преобразований

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
3×3(9)	13	70	1,44	7,78
4×4(16)	22	122	1,37	7,62
5×5(25)	55	369	2,20	14,76
6×6(36)	52	424	1,44	11,78
7×7(49)	121	1163	2,47	23,73
8×8(64)	130	750	2,03	11,72
9×9(81)	193	1382	2,38	17,06
10×10(100)	220	1876	2,20	18,76
14×14(196)	484	5436	2,47	27,73
16×16(256)	634	4774	2,48	18,65
18×18(324)	772	6576	2,38	20,30
24×24(576)	1402	12 954	2,43	22,49
27×27(729)	2893	21 266	3,97	29,17
32×32(1024)	3658	24 854	3,57	24,27
64×64(4096)	17 770	142 902	4,34	34,89
128×128(16 384)	78 250	718 966	4,78	43,88

ляемых в соответствии с этими алгоритмами, дано в табл. 3.4. Следовало бы заметить, что мы оптимизировали алгоритмы с точки зрения снижения числа умножений. Если вычисления осуществляются на ЭВМ, у которой время выполнения умножения сравнимо с временем сложения, предпочтительно использовать другие алгоритмы полиномиальных произведений, в которых еще больше уменьшается число сложений за счет некоторого увеличения числа умножений.

3.2.5. Гнездовые алгоритмы

Систематическое применение методов, рассмотренных в предыдущем разделе, позволяет построить большое число алгоритмов полиномиального произведения, требующих относительно малого числа сложений и умножений, и, следовательно, вычислять большие двумерные свертки с помощью составных полиномиальных преобразований. В качестве альтернативного подхода может служить конструирование больших двумерных сверток из ограниченного ряда коротких двумерных сверток методами, подобными тому, который использовался для одномерных сверток [3.2]. В этом методе двумерная $(N_1N_2\times N_1N_2)$-точечная свертка $y_{u,l}$, N_1 и N_2 — взаимно-простые, превращается в четырехмерную $[(N_1\times N_1)\times(N_2\times N_2)]$-точечную свертку путем простого изменения индексации. Пусть задано

$$y_{u,l}=\sum_{m=0}^{N_1N_2-1}\sum_{n=0}^{N_1N_2-1} h_{n,m}\,x_{u-n,\,l-m}. \tag{3.59}$$

Поскольку N_1 и N_2 — взаимно-простые числа и индексы l, m, n и u определены по модулю N_1N_2, прямым следствием китайской теоремы об остатках [3.7] является то, что l, m, n и u можно отобразить на два ряда индексов l_1, m_1, n_1, u_1 и l_2, m_2, n_2, u_2, где

$$\begin{cases} l \equiv N_1 l_2 + N_2 l_1 \bmod N_1N_2; \\ m \equiv N_1 m_2 + N_2 m_1 \bmod N_1N_2; \\ n \equiv N_1 n_2 + N_2 n_1 \bmod N_1N_2,\ l_1 m_1,\ n_1,\ u_1 = 0,\ \ldots,\ N_1-1; \\ u \equiv N_1 u_2 + N_2 u_1 \bmod N_1N_2;\ l_2,\ m_2,\ n_2,\ u_2 = 0,\ \ldots,\ N_2-1. \end{cases} \tag{3.60}$$

Следовательно, $y_{u,l}$ преобразуется в четырехмерную $(N_1\times N_1)\times(N_2\times N_2)$-точечную свертку

$$y_{N_1u_2+N_2u_1,\,N_1l_2+N_2l_1}=\sum_{m_1=0}^{N_1-1}\sum_{n_1=0}^{N_1-1}\sum_{m_2=0}^{N_2-1}\sum_{n_2=0}^{N_2-1} h_{N_1n_2+N_2n_1,\,N_1m_2+N_2m_1}\times$$

$$\times\, x_{N_1(u_2-n_2)+N_2(u_1-n_1),\,N_1(l_2-m_2)+N_2(l_1-m_1)}. \tag{3.61}$$

Эту четырехмерную свертку можно рассматривать как двумерную $(N_1\times N_1)$-точечную свертку, в которой все скаляры заменены двумерными $(N_2\times N_2)$-точечными полиномами. Точнее, представляя входные и выходные последовательности в виде двумерных полиномов, получим

$$H_{n,m_1}(Z_1,Z_2)=\sum_{m_2=0}^{N_2-1}\sum_{n_2=0}^{N_2-1} h_{N_1n_2+N_2n_1,\,N_1m_2+N_2m_1}\,Z_1^{m_2}\,Z_2^{n_2}; \tag{3.62}$$

$$X_{r_1,\,s_1}(Z_1,Z_2)=\sum_{r_2=0}^{N_2-1}\sum_{s_2=0}^{N_2-1} x_{N_1r_2+N_2r_1,\,N_1s_2+N_2s_1}\times$$

$$\times\, Z_1^{r_2}\,Z_2^{s_2},\ r_1s_1=0,\ \ldots,\ N_1-1; \tag{3.63}$$

$$Y_{u_1,\,l_1}(Z_1, Z_2) = \sum_{u_2=0}^{N_2-1} \sum_{l_2=0}^{N_2-1} y_{N_1u_2+N_2u_1,\,N_1l_2+N_2l_1}\, Z_1^{u_2}\, Z_2^{l_2}; \qquad (3.64)$$

$$Y_{u_1,l_1}(Z_1, Z_2) = \sum_{m_1=0}^{N_1-1} \sum_{n_1=0}^{N_1-1} H_{n_1,m_1}(Z_1, Z_2) \times$$

$$\times X_{u_1-n_1,\,l_1-m_1}(Z_1, Z_2) \bmod (Z_1^{N_2}-1), (Z_2^{N_2}-1). \qquad (3.65)$$

Каждое полиномиальное умножение $H_{n_1m_1}(Z_1, Z_2)X_{u_1-n_1,\,l_1-m_1}(Z_1, Z_2)$ по модулю $Z_1^{N_2}-1$, $Z_2^{N_2}-1$ соответствует $(N_2\times N_2)$-точечной свертке, которая вычисляется с M_2 скалярными умножениями и A_2 скалярными сложениями. В $(N_1\times N_1)$-точечной свертке все скаляры заменяются полиномами $N_2\times N_2$. Таким образом, если M_1 и A_1 — число умножений и число сложений, требуемых для вычисления скалярной $(N_1\times N_1)$-точечной свертки, то число умножений M и число сложений A, требуемых для вычисления двумерной $(N_1N_2\times N_1N_2)$-точечной свертки, сводится к

$$M = M_1 M_2, \qquad (3.66)$$

$$A = N_2^2 A_1 + M_1 A_2. \qquad (3.67)$$

Если N_1 и N_2 поменять местами, число умножений останется таким же, число сложений станет равным $N_1^2A_2+M_2A_1$. В случае более двух множителей этот способ вычисления можно использовать рекурсивно, при условии, что все множители взаимно-простые. На практике $(N_1\times N_1)$- и $(N_2\times N_2)$-точечные свертки вычисляются с помощью полиномиальных преобразований, а большие двумерные свертки — посредством небольшого набора алгоритмов полиномиальных преобразований. Например, (15×15)-точечную свертку можно было бы вычислить с помощью (3×3)- и (5×5)-точечных сверток. Поскольку эти свертки вычисляются соответственно за 13 умножений, 70 сложений и 55 умножений, 369 сложений (см. табл. 3.4), вложение двух алгоритмов позволяет вычислить (15×15)-точечную свертку за 715 умножений и 6547 сложений. В табл. 3.5 приведено число арифметических операций для различных двумерных сверток, вычисленных с помощью полиномиальных преобразований и вложения. Можно видеть, что этот метод дает очень малое число умножений на отсчет, но число

Таблица 3.5

Число операций для двумерных сверток, вычисляемых с помощью полиномиальных преобразований и вложения

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
12×12 (144)	286	2638	1,99	18,32
20×20 (400)	946	14030	2,36	35,07
30×30 (900)	2236	35404	2,48	39,34
36×36 (1296)	4246	40286	3,28	31,08
40×40 (1600)	4558	80802	2,85	50,50
60×60 (3600)	12 298	195 414	3,42	54,28
72×72 (5184)	20 458	232 514	3,95	44,85
80×80 (6400)	27 262	345 826	4,26	54,03
120×120 (14 400)	59 254	1 081 756	4,11	75,12

Таблица 3.6

Число операций для двумерных сверток, вычисляемых с помощью полиномиальных преобразований и послойного вложения

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
12×12 (144)	286	2 290	1,99	15,90
20×20 (400)	946	10 826	2,36	27,06
30×30 (900)	2236	28 348	2,48	31,50
36×36 (1296)	4246	34 010	3,28	26,24
40×40 (1600)	4558	69 534	2,85	43,46
60×60 (3600)	12 298	129 106	3,42	35,86
72×72 (5184)	20 458	192 470	3,95	37,13
80×80 (6400)	26 254	308 494	4,10	48,20
120×120 (14 400)	59 254	686 398	4,11	47,67

сложений на отсчет, в общем, выше, чем при использовании составных полиномиальных преобразований. Число сложений можно уменьшить посредством небольшого усовершенствования метода вложения Агарвала — Кули. Это осуществляется с помощью метода послойного вложения [3.4], в котором первый шаг вычислений такой же, как в методе Агарвала — Кули, причем $(N_1N_2 \times N_1N_2)$-точечная свертка вычисляется как $(N_1 \times N_1)$-точечная, в которой скаляры заменяются полиномами $N_2 \times N_2$. Таким образом, A_1 сложений, соответствующих алгоритму $N_1 \times N_1$, дают согласно (3.67) $N^2_2 A_1$ сложений. В методе Агарвала — Кули M_1 умножений, соответствующих алгоритму $N_1 \times N_1$, затем заменяют на M_1 $(N_2 \times N_2)$-точечных сверток, что дает, таким образом, $M_1 A_2$ сложений. В послойно-гнездовом алгоритме, однако, роль N_1 и N_2 на этом шаге меняется. Если предположить, что N_1 и N_2 — простые, то M_1 умножений, связанных с алгоритмом $N_1 \times N_1$, соответствуют N_1+1 произведениям полиномов из N_1-1 членов плюс одно умножение. Поэтому M_1 $(N_2 \times N_2)$-точечных сверток можно рассматривать как $(N_2 \times N_2)$-точечную свертку, в которой скаляры заменены на N_1+1 полиномов из N_1-1 элементов плюс один скаляр. Таким образом, каждое сложение в алгоритме $N_2 \times N_2$ повторяется N^2_1 раз вместо M_1 и, поскольку $N^2_1 < M_1$, число сложений уменьшается, в то время как число умножений остается неизменным.

В табл. 3.6 приведено число арифметических операций для различных двумерных сверток, вычисляемых с помощью полиномиальных преобразований и метода послойного вложения. Можно видеть, что число сложений значительно меньше, чем при обычном вложении, и сравнимо с числом сложений для метода, использующего большие составные полиномиальные преобразования.

3.2.6. Сравнение с традиционными вычислительными методами

Подробное сравнение метода полиномиального преобразования с традиционными вычислительными методами затруднено тем, что все зависит от конкретного выбора алгоритмов, относительной стоимости умножений и сложений, а также стоимости вспомогательных операций.

Для сверток вещественных последовательностей метод полиномиального преобразования требует только вещественных арифметических операций и не

Таблица 3.7

Число операций для двумерных сверток, вычисляемых методом вложения Агарвала — Кули

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
3×3 (9)	16	77	1,78	8,56
4×4 (16)	25	135	1,56	8,44
5×5 (25)	100	465	4,00	18,60
7×7 (49)	256	1610	5,22	32,86
8×8 (64)	196	1012	3,06	15,81
9×9 (81)	361	2072	4,46	25,58
12×12 (144)	400	3140	2,78	21,81
16×16 (256)	1225	7905	4,79	30,88
20×20 (400)	2500	16 100	6,25	40,25
30×30 (900)	6400	41 060	7,11	45,62
36×36 (1296)	9025	62 735	6,96	48,41
40×40 (1600)	19 600	116 440	12,25	72,77
60×60 (3600)	40 000	264 500	11,11	73,47
72×72 (5184)	70 756	488 084	13,65	94,15
80×80 (6400)	122 500	799 800	19,14	124,97
120×120 (14 400)	313 600	1 986 240	21,78	137,93

использует тригонометрических функций. Таким образом, представляется логичным сравнивать его с методом вложения Агарвала — Кули, который также обладает этими свойствами и относительно эффективен для одномерных сверток. В табл. 3.7 приведено число арифметических операций для двумерных сверток, вычисленных по этому методу, как описано в подразд. 3.2.5, но без полиномиальных преобразований. Из сравнения табл. 3.7 и 3.6 можно видеть, что число операций всегда больше, чем в методе полиномиального преобразования, для которого относительная эффективность повышается с увеличением размерности векторов в операции свертки. Например, для (120×120)-точечной свертки метод полиномиального преобразования требует примерно в 5 раз меньше умножений и в 2,5 раза меньше сложений, чем метод Агарвала — Кули.

Сравнение с вычислением при помощи БПФ является до некоторой степени более сложным, поскольку в этом случае вещественные свертки требуют комплексной арифметики и включают операции с тригонометрическими функциями. В табл. 3.8 приведено число вещественных операций для вещественных двумерных сверток, вычисляемых с помощью БПФ-алгоритма Рейдера — Бреннера [3.15]. Предполагается, что две вещественные свертки вычисляются отдельно, а тригонометрические функции вычисляются и запоминаются заранее. При этих условиях, довольно благоприятных для алгоритма БПФ, число сложений для него немного больше, а число умножений примерно в два раза больше по сравнению с методом полиномиального преобразования. Применение обычного алгоритма БПФ по основанию 4 или алгоритма Винограда преобразования Фурье [3.1] приведет к сравнимым результатам.

Воспользовавшись тем, что для некоторых полей $j = \sqrt{-1}$ является вещественным, метод полиномиального преобразования для комплексных сверток можно реализовать, затрачивая два действительных умножения на комплексное умножение. Например, для случая полей полиномов по модулю Z^q+1 при четном

Таблица 3.8

Число вещественных операций для вещественных сверток при вычислении с помощью БПФ. (Алгоритм Рейдера — Бреннера, 2 вещественные свертки на ДПФ)

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
8×8 (64)	160	928	2,50	14,50
16×16 (256)	1024	5120	4,00	20,00
32×32 (1024)	5888	28672	5,75	28,00
64×64 (4096)	31 232	147 456	7,62	36,00
128×128 (16 384)	156 672	720 896	9,56	44,00

q $j \equiv Z^{q/2}$. Таким образом, комплексная свертка $Y_l(Z) + j\,\bar{Y}_l(Z)$, определенная по модулю Z^q+1, вычисляется с помощью двух действительных сверток

$$Q_{1,l}(Z) \equiv \sum_{m=0}^{N-1} [H_m(Z) + Z^{q/2}\,\hat{H}_m(Z)]\,[X_{l-m}(Z) + Z^{q/2}\,\hat{X}_{l-m}(Z)] \bmod (Z^q+1); \tag{3.68}$$

$$Q_{2,l}(Z) \equiv \sum_{m=0}^{N-1} [H_m(Z) - Z^{q/2}\,\hat{H}_m(Z)]\,[X_{l-m}(Z) - Z^{q/2}\,\hat{X}_{l-m}(Z)] \bmod (Z^q+1); \tag{3.69}$$

$$Y_l(Z) = [Q_{1,l}(Z) + Q_{2,l}(Z)]/2; \tag{3.70}$$

$$\hat{Y}_l(Z) \equiv -Z^{q/2}\,[Q_{1,l}(Z) - Q_{2,l}(Z)]/2 \bmod (Z^q+1). \tag{3.71}$$

При этих условиях для вычисления комплексных сверток с помощью полиномиальных преобразований требуется вдвое больше арифметических операций, чем для вещественных сверток. Относительная эффективность метода полиномиального преобразования по сравнению с методом БПФ примерно такая же, как и при вычислении вещественных сверток.

Необходимо также заметить, что, когда алгоритмы реализуются путем микропрограммирования, может оказаться выгодным вычислять короткие полиномиальные произведения с помощью распределенной арифметики [3.16]. Для этого метода, который предполагает битовые операции над отдельными разрядами чисел, число операций в алгоритмах полиномиального произведения резко уменьшается, что существенно повышает, таким образом, эффективность метода полиномиального преобразования за счет увеличения сложности программирования.

3.3. Вычисление двумерных ДПФ с помощью полиномиальных преобразований

Показано, что число арифметических операций, требуемых для вычисления двумерных циклических сверток, существенно снижается при замене традиционных методов, использующих ДПФ, полиномиальными преобразованиями. Частично экономия обусловлена тем, что короткие одномерные свертки и полино-

миальные произведения вычисляются с помощью интерполяции более эффективно, чем методами ДПФ. Однако наиболее существенный фактор понижения сложности вычислительного процесса связан с эффективным отображением с помощью полиномиальных преобразований двумерных последовательностей в одномерные. Покажем теперь, что эти методы можно распространить на вычисление ДПФ и что полиномиальные преобразования можно применить для преобразования двумерных ДПФ либо в одномерные ДПФ, либо в одномерные свертки и полиномиальные произведения [3.17]. На практике оба метода обеспечивают существенное снижение числа арифметических операций и могут применяться совместно для достижения оптимальной эффективности вычисления больших ДПФ.

3.3.1. Алгоритм редуцированного ДПФ

Рассмотрим двумерное $N \times N$-точечное ДПФ

$$\bar{X}_{k_1, k_2} = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} x_{n_1, n_2} W^{n_1 k_1} W^{n_2 k_2}, \quad W = \exp(-j 2\pi/N);$$

$$k_1, k_2 = 0, 1, \ldots, N-1; \quad j = \sqrt{-1}. \tag{3.72}$$

Традиционный построчно-столбцовый метод [3.18] состоит в вычислении $\bar{X}_{k_1,k_2}$ как N одномерных ДПФ по индексу k_2 и N одномерных ДПФ по индексу k_1. В этом методе $\bar{X}_{k_1,k_2}$ отображается в $2N$ N-точечных ДПФ. Если M_1 – число комплексных умножений, требуемых для вычисления этого ДПФ, то общее число M умножений, соответствующих $\bar{X}_{k_1,k_2}$, есть $M = 2NM_1$. Когда $M_1 < 2N$, более эффективно вычислять $\bar{X}_{k_1,k_2}$ с помощью гнездового алгоритма Винограда [3.1]. В этом случае $\bar{X}_{k_1,k_2}$ вычисляется как N-точечное ДПФ, в котором каждый скаляр заменяется вектором из N точек и каждое умножение заменяется N-точечным ДПФ, так что $M = M_1^2$.

Чтобы вычислить $\bar{X}_{k_1,k_2}$ с помощью полиномиальных преобразований, необходимо сначала представить ДПФ в терминах алгебры полиномов. Это осуществляется заменой (3.72) на

$$\bar{X}_{k_1}(Z) = \sum_{n_1=0}^{N-1} X_{n_1}(Z) W^{n_1 k_1} \bmod (Z^N - 1); \tag{3.73}$$

$$X_{n_1}(Z) = \sum_{n_2=0}^{N-1} x_{n_1, n_2} Z^{n_2}, \quad k_1 = 0, \ldots, N-1; \tag{3.74}$$

$$\bar{X}_{k_1, k_2} = \bar{X}_{k_1}(Z) \bmod (Z - W^{k_2}), \quad k_2 = 0, \ldots, N-1. \tag{3.75}$$

В (3.75) $\bar{X}_{k_1,k_2}$ получено заменой Z на W^{k_2} в (3.74). Таким образом, (3.73) – (3.75) эквивалентны (3.72). Заметим, что на этом шаге нет необходимости задавать $\bar{X}_{k_1}(Z)$ по модулю $Z^N - 1$, хотя такое задание справедливо, поскольку $Z^N \equiv W^{Nk_2} = 1$.

Предположим теперь, что N-простое. Тогда $Z^N - 1$ разлагается в произведение двух циклотомических полиномов:

$$Z^N - 1 = (Z - 1) P(Z), \tag{3.76}$$

$$P(Z) = Z^{N-1} + Z^{N-2} + \ldots + 1. \tag{3.77}$$

При $k_2=0$, что соответствует $Z\equiv 1$, $\bar{X}_{k_1,\,0}$ представляет собой простое N-точечное ДПФ

$$\bar{X}_{k_1,\,0}=\sum_{n_1=0}^{N-1}\left(\sum_{n_2=0}^{N-1}x_{n_1,n_2}\right)W^{n_1k_1},\quad k_1=0,\ldots,N-1. \tag{3.78}$$

При $k_2\neq 0$ W^{k_2} всегда является корнем полинома $P(Z)$:

$$P(Z)=\prod_{k_2=1}^{N-1}(Z-W^{k_2}). \tag{3.79}$$

Поскольку $(Z-W^{k_2})$ — множитель $P(Z)$, а $P(Z)$ — множитель (Z^N-1), $X_{k_1}(Z)$ можно определить по модулю $P(Z)$ вместо (Z^N-1) и (3.73)—(3.75) сводятся к

$$\bar{X}_{1,\,k_1}(Z)\equiv\sum_{n_1=0}^{N-1}X_{1,\,n_1}(Z)\,W^{n_1k_1}\bmod P(Z),\quad k_1=0,\ldots,N-1; \tag{3.80}$$

$$X_{1,\,n_1}(Z)=\sum_{n_2=0}^{N-2}(x_{n_1,n_2}-x_{n_1,\,N-1})\,Z^{n_2}\equiv X_{n_1}\bmod P(Z); \tag{3.81}$$

$$\bar{X}_{k_1,k_2}\equiv\bar{X}_{1,\,k_1}(Z)\bmod(Z-W^{k_2}),\quad k_2=1,\ldots,N-1. \tag{3.82}$$

Если N простое и $k_2\neq 0$, то перестановка k_1k_2 по модулю N отображает все значения k_1. Таким образом, порядок следования индексов k_1 можно изменить посредством умножения их на k_2, что приведет к

$$\bar{X}_{1,k_1k_2}(Z)\equiv\sum_{n_1=0}^{N-1}X_{1,\,n_1}(Z)\,W^{n_1k_1k_2}\bmod P(Z),$$
$$k_1=0,\ldots,N-1; \tag{3.83}$$

$$\bar{X}_{k_1k_2,\,k_2}\equiv\bar{X}_{1,\,k_1k_2}(Z)\bmod(Z-W^{k_2}),$$
$$k_2=1,\ldots,N-1. \tag{3.84}$$

Поскольку $\bar{X}_{k_1k_2,k_2}$ определен по модулю $(Z-W^{k_2})$, $W^{k_2}\equiv Z$. Подстановка Z вместо W^{k_2} в (3.83) дает

$$\bar{X}_{1,\,k_1k_2}(Z)\equiv\sum_{n_1=0}^{N-1}X_{1,\,n_1}(Z)\,Z^{n_1k_1}\bmod P(Z),\quad k_1=0,\ldots,N-1, \tag{3.85}$$

где правая часть равенства не зависит от k_2, $\bar{X}_{1,k_1k_2}(Z)$ — полиномиальное преобразование последовательности длиной N, которое вычисляется без умножений с помощью только N^3-N^2-3N+4 операций сложения. При этих условиях единственными умножениями, требуемыми для вычисления $(N\times N)$-точечного ДПФ, являются те, которые соответствуют (3.84) и заданному (3.78) N-точечному ДПФ.

Покажем теперь, что вычисление (3.84) эквивалентно вычислению N дискретных преобразований Фурье последовательности длиной N, в которой последний входной отсчет равен 0, а первый выходной отсчет не вычисляется. Это можно видеть из того, что выходные отсчеты $X_{1,k_1k_2}(Z)$ полиномиального преобразования (3.85) представляют собой N полиномов по $N-1$ членов. Полагая, что эти полиномы заданы выражением

$$\bar{X}_{1,\,k_1k_2}(Z)=\sum_{l=0}^{N-2}y_{k_1,l}Z^l, \tag{3.86}$$

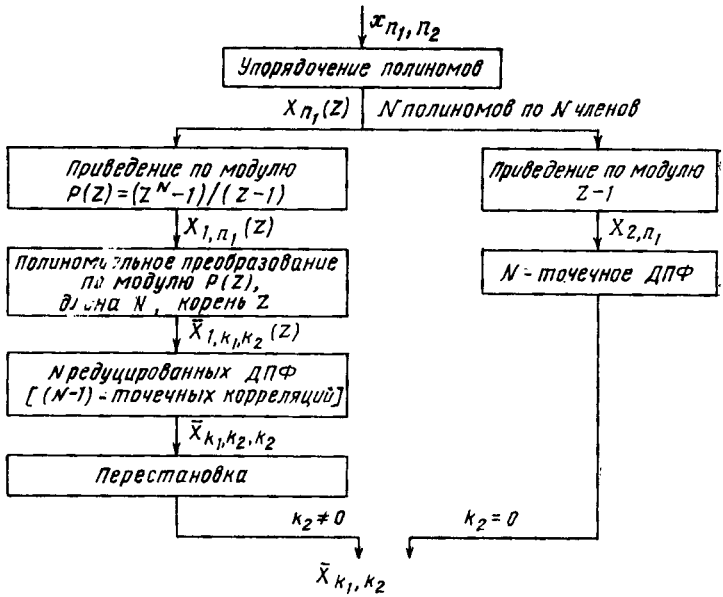


Рис. 3.6. Вычисление $(N\times N)$-точечного ДПФ с помощью полиномиальных преобразований, N — простое. Алгоритм редуцированного ДПФ

и подставляя их в (3.84), получим

$$\bar{X}_{k_1k_2,\,k_2}=\sum_{l=0}^{N-2} y_{k_1,\,l}W^{k_2l},\quad k_2=1,\ldots,N-1. \tag{3.87}$$

Таким образом, для простого N $(N\times N)$-точечное ДПФ отображается с помощью полиномиальных преобразований в $N+1$ N-точечных ДПФ вместо $2N$ N-точечных ДПФ для построчно-столбцового метода вычислений. Схема вычислений по этому методу полиномиального преобразования показана на рис. 3.6. Воспользовавшись тем, что в N ДПФ, определенных (3.87), последний входной член равен нулю, а первый выходной член не вычисляется, можно получить небольшое дополнительное сокращение числа умножений. Это можно реализовать, если заметить, что, поскольку $\sum_{l=1}^{N-1} W^{k_2l}=-1$ при $k_2\neq 0$ и простом N, то (3.87) можно переписать как

$$\bar{X}_{k_1k_2,\,k_2}=\sum_{l=1}^{N-1} y^1_{k_1,\,l}W^{k_2l},\quad k_2=1,\ldots,N-1, \tag{3.88}$$

где

$$y^1_{k_1,\,l}=(y_{k_1,\,l}-y_{k_1,\,0});\quad y^1_{k_1,\,N-1}=-y_{k_1,\,0};\quad l=1,\ldots,N-2. \tag{3.89}$$

Тогда *редуцированные* ДПФ (3.88) можно вычислить как $(N-1)$-точечные корреляции, используя алгоритм Рейдера [3.19]. Действительно, если g — примитивный корень по модулю N [3.7], то

$$l\equiv g^u \bmod N,\quad k_2\equiv g^v \bmod N. \tag{3.90}$$

В этом случае редуцированное ДПФ (3.88) превращается в корреляцию

$$\bar{X}_{k_1 g^v,\, g^{\sigma}} = \sum_{u=0}^{N-2} y^1_{k_1,\, g^u} W^{g^{u+v}}. \qquad (3.91)$$

Подстановка последовательности $y^1_{k_1,l}$ в $y_{k,l}$ эквивалентна умножению $X_{k_1 k_2}(Z)$ на $Z^{-1} \bmod (Z^N - 1)$ с последующим приведением по модулю $P(Z)$ и умножением на Z. На практике умножение на Z^{-1} можно объединить с упорядочением входных полиномов. Тогда приведение по модулю $P(Z)$ выполняется без сложений, как часть вычисления полиномиального преобразования.

Предположим, что N-точечное ДПФ вычисляется за M_1 комплексных умножений с помощью алгоритмов коротких корреляций, например, таких, как алгоритм Винограда преобразования Фурье. Тогда соответствующие корреляции потребуют $M_1 - 1$ комплексных умножений, а общее число M умножений, необходимых для вычисления $(N \times N)$-точечного ДПФ с помощью полиномиальных преобразований, понизится до

$$M = (N+1) M_1 - N. \qquad (3.92)$$

Это составляет примерно половину того, что необходимо для построчно-столбцового метода вычислений ($2NM_1$ умножений) и всегда, за исключением $M_1 = N$, ниже соответствующего числа умножений для гнездового алгоритма Винограда (M^2_1). Более того, когда ДПФ и редуцированные N-точечные ДПФ вычисляются с помощью алгоритмов коротких корреляций, все комплексные умножения сводятся к умножениям на чисто вещественные или чисто мнимые числа и могут выполняться только за 2 вещественных умножения.

Аналогичный метод полиномиального преобразования для вычисления ДПФ можно так же просто определить и для составного N. Условия выполнения полиномиального преобразования без умножений устанавливаются с помощью метода, очень похожего на тот, который использовался в подразд. 3.2.1 для двумерных сверток (табл. 3.9). Особенно интересен случай $(N \times N)$-точечного ДПФ при $N = 2^t$. Эти ДПФ вычисляются, как показано на рис. 3.7, посредством одного 2^t-точечного полиномиального преобразования по модулю $P_{t+1}(Z) = Z^{2^t - 1} + 1$, одного 2^{t-1}-точечного полиномиального преобразования по модулю $P_{t+1}(Z)$, $2^t + 2^{t-1}$ 2^t-точечных редуцированных ДПФ, половина входных и выходных членов которых равна нулю, и одного $(N/2 \times N/2)$-точечного ДПФ, которое, в свою очередь, можно вычислить аналогичным образом. Для случая $N = 2^t$ наиболее интересно то, что полиномиальные преобразования вычисляются с помощью алгоритма типа БПФ, так что все вычисления организуются способом, очень похожим на традиционное вычисление БПФ.

Более того, редуцированные N-точечные ДПФ можно тогда рассматривать как обычные $(N/2)$-точечные ДПФ, в которых входная последовательность умножается поточечно на последовательность $1, W, W^2$, и поэтому они идентичны редуцированным ДПФ, появляющимся на первом шаге прореживания по частоте в алгоритме БПФ с основанием 2. Точнее, редуцированное N-точечное ДПФ, $N = 2^t$, задается следующим образом:

$$\bar{X}_{(2v+1)k_1,\, 2v+1} = \sum_{l=0}^{N/2-1} y_{k_1,l} W^l W^{2vl}, \quad v = 0, \ldots, N/2 - 1. \qquad (3.93)$$

В этом редуцированном ДПФ умножения на W^l обычно являются комплексными. Однако, используя алгоритм Рейдера — Бреннера [3.15] или один из его производных, эти умножения можно преобразовать в умножения чисто вещест-

Таблица 3.9

Вычисление двумерных ДПФ с помощью полиномиальных преобразований. Алгоритм редуцированного ДПФ

Размеры массива	Полиномиальные преобразования	Число сложений для полиномиальных преобразований и приведений	Дискретные преобразования Фурье и редуцированные ДПФ
N — простое	N-точечное полиномиальное преобразование по модулю $(Z^N-1)/(Z-1)$	N^3+N^2-5N+4	N-точечное ДПФ, N редуцированных N-точечных ДПФ [N ($N-1$)-точечных корреляций]
$N=q^2$, q — простое	q^2-точечное полиномиальное преобразование по модулю $(Z^{q^2}-1)/(Z^q-1)$ q-точечное полиномиальное преобразование по модулю $(Z^{q^2}-1)/(Z^q-1)$	$2q^5+q^4-5q^3+q^2+6$	(q^2+q) редуцированных q^2-точечных ДПФ q редуцированных q-точечных ДПФ [q ($q-1$)-точечных корреляций] q-точечное ДПФ
$N=2^t$	2^t-точечное полиномиальное преобразование по модулю $(Z^{2^{t-1}}+1)$ 2^{t-1}-точечное полиномиальное преобразование по модулю $(Z^{2^{t-1}}+1)$	$(3t+5)2^{2(t-1)}$	$3,2^{t-1}$ редуцированных N-точечных ДПФ $(N/2\times N/2)$-точечное ДПФ
$N=q_1q_2$, q_1q_2 — простые	q_1q_2-точечное полиномиальное преобразование q_2 q_1-точечных полиномиальных преобразований q_1 q_2-точечных полиномиальных преобразований	$q^2_1q^2_2(q_1+q_2+2)-5q_1q_2(q_1+q_2)+4(q^2_1+q^2_2)$	$(q_1q_2+q_1+q_2)$ редуцированных q_1q_2-точечных ДПФ q_1 редуцированных q_1-точечных ДПФ q_2 редуцированных q_2-точечных ДПФ q_1q_2-точечное ДПФ

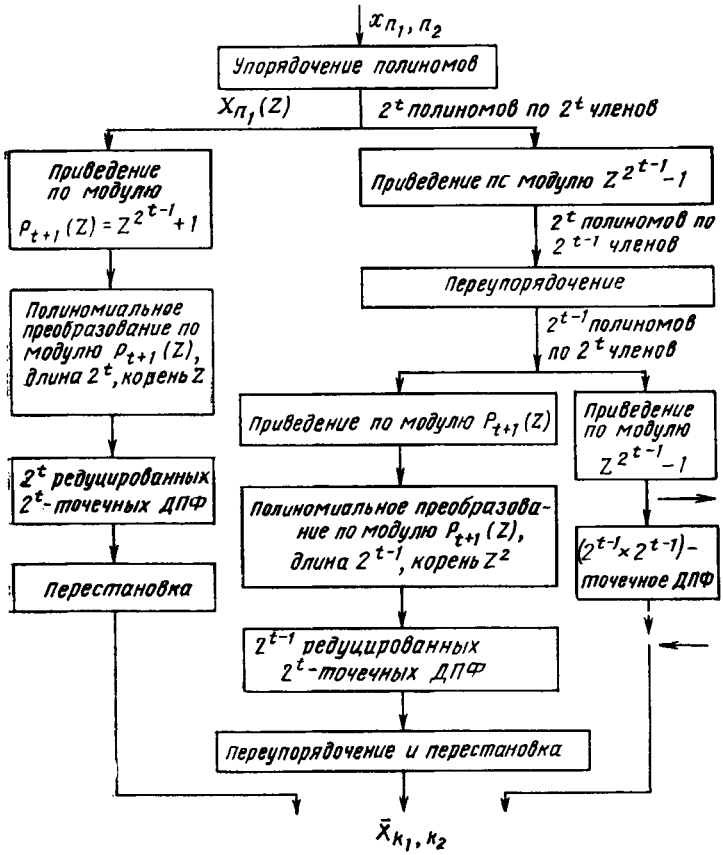


Рис. 3.7. Вычисление $(2^t \times 2)^t$-точечного ДПФ с помощью полиномиальных преобразований. Алгоритм редуцированного ДПФ

вениых или чисто мнимых чисел. Это осуществляется заменой входной последовательности $y_{k_1, l}$ последовательностью $y^1_{k_1, l}$, определяемой как

$$\begin{cases} y^1_{k_1, 0} = 0, \quad y^1_{k_1, N/4} = 0, \\ y^1_{k_1, l} = y_{k_1, l} / 2\cos(2\pi l/N), \quad l \neq 0, \ N/4. \end{cases} \tag{3.94}$$

Тогда $(N/2)$-точечное ДПФ $\bar{X}^1_{(2v+1)k_1, 2v+1}$ от $y^1_{k_1, l}$ вычисляется при

$$\bar{X}^1_{(2v+1)k_1, 2v+1} = \sum_{l=0}^{N/2-1} y^1_{k_1, l} W^{2vl}. \tag{3.95}$$

Поскольку $W^{2vl} + W^{2(v+1)l} = W^l W^{2vl}(W^l + W^{-l}) = 2W^l W^{2vl}\cos(2\pi l/N)$, $\bar{X}_{(2v+1)k_1, 2v+1}$ вычисляются посредством простых сложений:

$$\begin{aligned} \bar{X}_{(2v+1)k_1, 2v+1} &= \bar{X}^1_{(2v+1)k_1, 2v+1} + \bar{X}^1_{(2v+3)k_1, 2v+3} + e_0, \ v \text{—четное}; \\ \bar{X}_{(2v+1)k_1, 2v+1} &= \bar{X}^1_{(2v+1)k_1, 2v+1} + \bar{X}^1_{(2v+3)k_1, 2v+3} + e_1, \ v \text{—нечетное}, \end{aligned} \tag{3.96}$$

где $e_0 = y_{k_1, 0} - \mathrm{j} y_{k_1, N/4}$; $e_1 = y_{k_1, 0} + \mathrm{j} y_{k_1, N/4}$.

Таблица 3.10

Число вещественных операций для коротких и редуцированных ДПФ

Длина последовательности	Число умножений	Число сложений	
2	4(4)	4	ДПФ
3	6(2)	12	
4	8(8)	16	
5	12(2)	34	
7	18(2)	72	
8	16(12)	52	
9	22(2)	88	
16	36(16)	148	
32	104(36)	424	
64	272(76)	1104	
128	672(156)	2720	
256	1600(316)	6464	
512	3712(636)	14 976	
1024	8448(1276)	34 048	
3	4(0)	8	Редуцированные ДПФ, вычисляемые как корреляции
5	10(0)	30	
7	16(0)	68	
4	4(4)	4	Редуцированные ДПФ
8	8(4)	20	
9	16(0)	56	
16	20(4)	64	
32	68(20)	212	
64	168(40)	552	
128	400(80)	1360	
256	928(160)	3232	
512	2112(320)	7488	
1024	4736(640)	17 024	

Примечание. В скобках указаны тривиальные операции умножения на ± 1, $\pm j$

При использовании этого метода редуцированное ДПФ, определенное в (3.93), вычисляется за M_1+N-4 вещественных умножений и A_1+2N вещественных сложений, где M_1 и A_1 — число умножений и число сложений, требуемых для вычисления $(N/2)$-точечных ДПФ.

В разд. 3.6 приведены различные алгоритмы коротких редуцированных ДПФ, а в табл. 3.10 собраны данные о числе вещественных операций для наиболее часто употребляемых алгоритмов. На основании этих данных и алгоритмов коротких ДПФ, предназначенных для использования в алгоритмах Винограда преобразования Фурье [3.1, 3.2] и Рейдера — Бреннера, в табл. 3.11 показано число вещественных операций для различных двумерных ДПФ.

Подробное сравнение с обычными вычислительными методами будет дано в подразд. 3.3.3, но пока можно заметить, что число умножений равно примерно половине того числа, которое соответствует традиционному вычислению БПФ с помощью алгоритма Рейдера — Бреннера, а число сложений незначительно уменьшается. Число операций также меньше, чем в обычных алгоритмах Винограда преобразования Фурье.

Таблица 3.11

Число вещественных операций для комплексных двумерных ДПФ, вычисляемых с помощью полиномиальных преобразований по алгоритму редуцированного ДПФ

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
2×2	0	16	0	4,00
3×3	16	86	1,78	9,56
4×4	0	128	0	8,00
5×5	60	442	2,40	17,68
7×7	128	1270	2,61	25,92
8×8	48	816	0,75	12,75
9×9	208	1570	2,57	19,38
16×16	432	4528	1,69	17,69
32×32	2736	24 944	2,67	24,36
64×64	15 024	125 040	3,67	30,53
128×128	76 464	599 152	4,67	36,57
256×256	371 376	2 790 512	5,67	42,58
512×512	1 747 632	12 735 600	6,67	48,58
1024×1024	8 039 088	57 234 544	7,67	54,58

Примечание. Тривиальные умножения на ± 1, $\pm j$ не учитываются.

3.3.2. Гнездовые алгоритмы и алгоритмы простых множителей

Метод полиномиального преобразования, рассмотренный в предыдущем разделе, позволяет вычислять большие двумерные $(N \times N)$-точечные ДПФ при условии, что имеются алгоритмы больших редуцированных ДПФ. Для $N = 2^t$ эти алгоритмы редуцированных ДПФ строятся достаточно просто с помощью метода Рейдера — Бреннера. Когда N-составное и не является степенью 2, алгоритмы больших редуцированных ДПФ можно построить путем вложения алгоритмов коротких редуцированных ДПФ. Однако при составном N часто гораздо удобнее конструировать большие преобразования, используя небольшой набор коротких преобразований в гнездовых алгоритмах [3.1] или алгоритме простых множителей [3.20, 3.21]. Эти методы особенно удобны, когда ДПФ имеет в обеих размерностях взаимно-простые множители. В последующем ограничимся рассмотрением $(N_1N_2 \times N_1N_2)$-точечных преобразований Фурье, где N_1 и N_2 — взаимно-простые, поскольку при более чем двух взаимно-простых множителях бо́льшие ДПФ можно вычислять рекурсивно, следуя алгоритму для двух множителей.

В гнездовых алгоритмах двумерное $(N_1N_2 \times N_1N_2)$-точечное ДПФ сначала превращается в четырехмерное $[(N_1 \times N_1) \times (N_2 \times N_2)]$-точечное путем простой индексации, на основе китайской теоремы об остатках, что аналогично методу, описанному в подразд. 3.2.5 для двумерных сверток. Четырехмерное ДПФ, в свою очередь, вычисляется по методу вложения Винограда [3.1] с помощью полиномиальных преобразований для $(N_1 \times N_1)$-точечного ДПФ, в котором каждый скаляр заменяется массивом $N_2 \times N_2$ и каждое умножение заменяется $(N_2 \times N_2)$-точечным ДПФ. Таким образом, если M_1, M_2, M и A_1, A_2, A — соответ-

ственно число комплексных умножений и сложений, требуемых для оценки $(N_1\times N_1)$-, $(N_2\times N_2)$- и $(N_1N_2\times N_1N_2)$-точечных ДПФ, то

$$M = M_1M_2, \tag{3.97}$$

$$A = N^2{}_2A_1 + M_1A_2. \tag{3.98}$$

Четырехмерное $[(N_1\times N_1)\times(N_2\times N_2)]$-точечное ДПФ можно также выполнить с помощью алгоритма простых множителей или построчно-столбцового метода как $N^2{}_1$ $(N_2\times N_2)$-точечных ДПФ и $N^2{}_2$ $(N_1\times N_1)$-точечных ДПФ. В этом случае число операций становится равным

$$M = N^2{}_1M_2 + N^2{}_2M_1, \tag{3.99}$$

$$A = N^2{}_1A_2 + N^2{}_2A_1. \tag{3.100}$$

Поскольку $M_1 \geqslant N^2{}_1$, $M_2 \geqslant N^2{}_2$, то метод вложения всегда требует больше сложений, чем алгоритм простых множителей. Однако, если M_1 и M_2 не намного больше $N^2{}_1$ и $N^2{}_2$, что характерно для случая относительно коротких ДПФ, метод вложения требует меньше умножений, чем алгоритм простых множителей. При этом число сложений возрастает незначительно. Таким образом, гнездовой алгоритм больше пригоден для ДПФ массивов средних размеров, в то время как метод простых множителей — для ДПФ больших массивов.

Вычислительную эффективность этих алгоритмов можно улучшить, применяя метод вложения [3.17]. При этом, если воспользоваться способом, аналогичным описанному в подразд. 3.2.5 для сверток, понижается число сложений. Для алгоритма простых множителей использование метода вложения уменьшает число умножений. Для упрощения обозначений опишем этот последний алгоритм для $(N_1\times N_2)$-точечной ДПФ, результаты которого легко обобщить на $[(N_1\times N_1)\times(N_2\times N_2)]$-точечное ДПФ посредством замены скаляров векторами, состоящими из N_1 и N_2 элементов соответственно. В последующем будем предполагать, что N_1 и N_2 — простые. $(N_1\times N_2)$-точечное ДПФ $\bar{X}_{k_1,k_2}$ задано следующим образом:

$$\bar{X}_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} W_1^{n_1k_1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} W_2^{n_2k_2}, \tag{3.101}$$

$$W_1 = \exp(-\mathrm{j}2\pi/N_1); \quad W_2 = \exp(-\mathrm{j}2\pi/N_2);$$

$$k_1 = 0,\ldots,N_1-1; \quad k_2 = 0,\ldots,N_2-1.$$

Используя алгоритм Рейдера [3.19], это ДПФ можно найти как одно N_1-точечное ДПФ, одну $(N_1-1)\times(N_2-1)$-точечную и одну (N_2-1)-точечную корреляцию

$$\bar{X}_{k_1,0} = \sum_{n_1=0}^{N_1-1}\left(\sum_{n_2=0}^{N_2-1} x_{n_1,n_2}\right) W_1^{n_1k_1}, \quad k_1 = 0,\ldots,N_1-1; \tag{3.102}$$

$$\bar{X}_{0,g^{v_2}} = \sum_{u_2=0}^{N_2-2}\left[\sum_{n_1=0}^{N_1-1}(x_{n_1,g^{u_2}} - x_{n_1,0})\right] W_2^{g^{t_2-v_2}}; \tag{3.103}$$

$$\bar{X}_{h^{v_1},g^{v_2}} = \sum_{u_2=0}^{N_2-2} W_2^{g^{u_2}+v_2} \sum_{u_1=0}^{N_1-2}(x_{h^{u_1},g^{u_2}} - x_{h^{u_1},0} - x_{0,g^{u_2}} + x_{0,0}) W_1^{h^{u_1}+v_1}, \tag{3.104}$$

где h и g — примитивные корни по модулю N_1 и N_2 и

$$k_1 \equiv h^{v_1} \bmod N_1; \quad n_1 \equiv h^{u_1} \bmod N_1; \quad u_1, v_1 = 0, \ldots, N_1 - 2; \tag{3.105}$$

$$k_2 \equiv g^{v_2} \bmod N_2; \quad n_2 \equiv g^{u_2} \bmod N_2; \quad u_2, v_2 = 0, \ldots, N_2 - 2. \tag{3.106}$$

Двумерная корреляция (3.104) является полуразделимой и поэтому может быть вычислена как N_2-1 (N_1-1)-точечных корреляций плюс N_1-1 (N_2-1)-точечных корреляций. При этих условиях, если M_1 и M_2 — число умножений, соответствующих N_1- и N_2-точечным ДПФ, общее число умножений снижается до $N_1M_2+N_2M_1-N_1-N_2+1$ вместо $N_1M_2+N_2M_1$ для обычного алгоритма простых множителей. Таким образом, метод вложения сберегает N_1+N_2-1 умножений. Большей экономии можно достичь, преобразуя корреляции в циклотомические полиномы.

3.3.3. Вычисление преобразования Фурье методом Винограда с помощью полиномиальных преобразований

В подразд. 3.3.1 обсуждался эффективный метод вычисления ДПФ с помощью полиномиальных преобразований. Этот метод редуцированного ДПФ применим главным образом к многомерным ДПФ, когда имеется общий множитель в двух или более измерениях, и основан на отображении многомерных ДПФ в одномерные и редуцированные ДПФ с помощью полиномиальных преобразований. В этом разделе предлагается другой метод вычисления ДПФ с помощью полиномиальных преобразований [3.1, 3.17]. Покажем, что если ДПФ вычислять с помощью алгоритма Винограда, полиномиальные преобразования можно использовать для отображения ДПФ в одномерные свертки и полиномиальные произведения. По сравнению с алгоритмом Винограда эти полиномиальные преобразования позволяют получить значительную экономию вычислений.

В данном методе ДПФ сначала преобразуется в многомерные корреляции с помощью алгоритма Винограда [3.1, 3.21]. Для $N_1 \times N_2$-точечного ДПФ такое преобразование выполняется путем вычисления N_1-точечного ДПФ, в котором каждый скаляр заменяется N_2-точечным ДПФ. Если N_1- и N_2-точечные ДПФ вычисляются как корреляции, в соответствии с алгоритмом Рейдера, эту вычислительную процедуру можно рассматривать как преобразование ДПФ ($N_1 \times N_2$)-точечного ДПФ в одномерные ДПФ и корреляции и двумерные корреляции. Например, для простого случая, соответствующего простым N_1 и N_2, ($N_1 \times N_2$)-точечное ДПФ вычисляется, как следует из (3.102) — (3.104), посредством одного N_1-точечного ДПФ, одной (N_2-1)- и одной (N_1-1) $\times$ (N_2-1)-точечных корреляций. Такой же метод используется для вычисления одномерного N_1N_2-точечного ДПФ, если N_1 и N_2 — взаимно-простые, поскольку это ДПФ можно превратить в двумерное ($N_1 \times N_2$)-точечное ДПФ простой переиндексацией с помощью алгоритма Гуда [3.20]. Если N имеет более двух множителей, тот же метод можно применить рекурсивно. В этом случае ДПФ превращается в многомерные корреляции.

В обычном алгоритме Винограда многомерные корреляции вычисляются простым вложением коротких одномерных корреляций, что эквивалентно вычислению корреляций с помощью алгоритма Агарвала — Кули [3.2]. Однако в разд. 3.2 было показано, что многомерные корреляции вычисляются более эффективно с помощью полиномиальных преобразований, чем методом Агарвала — Кули, если коррелируемые массивы имеют общие множители в нескольких

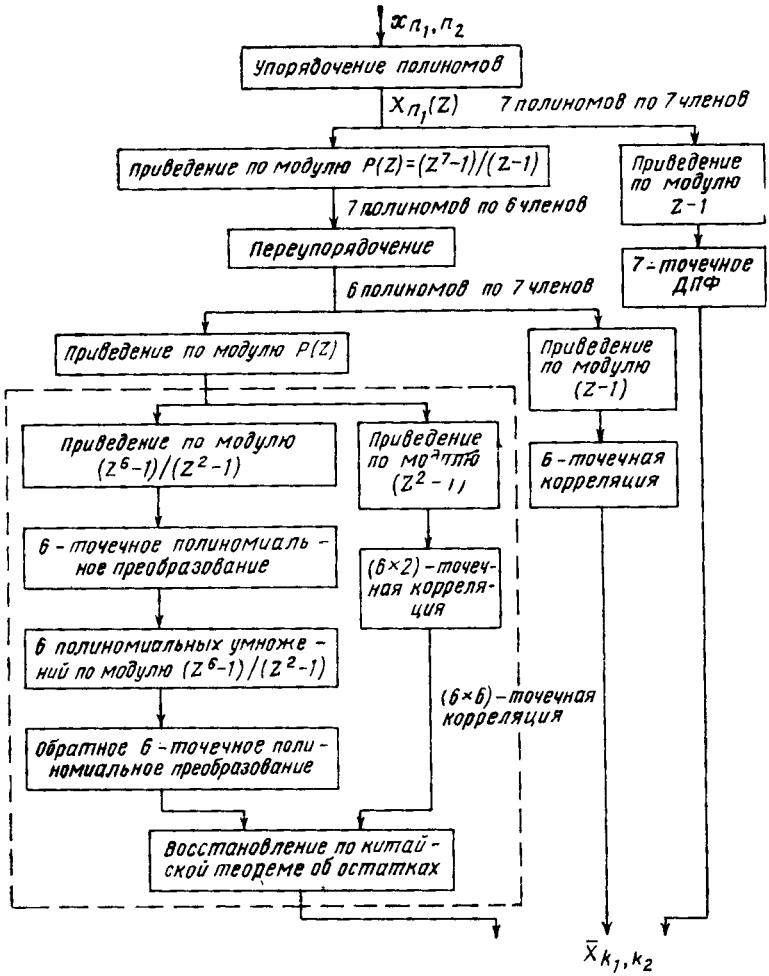


Рис. 3.8. Вычисление (7×7)-точечного ДПФ с помощью алгоритма Винограда и полиномиальных преобразований

размерностях. Таким образом, вычисление многомерных сверток с помощью полиномиальных преобразований снижает число сложений и умножений в алгоритме Винограда преобразования Фурье.

Этот метод иллюстрируется на рис. 3.8 для (7×7)-точечного ДПФ. Так как 7-точечное ДПФ вычисляется посредством одного умножения и одной 6-точечной корреляции, то (7×7)-точечное ДПФ отображается в одно 7-точечное ДПФ, одну 6-точечную корреляцию и одну (6×6)-точечную корреляцию. Поэтому, если используется алгоритм, требующий 8 комплексных умножений для 6-точечной корреляции, то (7×7)-точечное ДПФ вычисляется в соответствии с обычным алгоритмом Винограда за 81 комплексное умножение. Вычисление (6×6)-точеч-

ной корреляции с помощью полиномиальных преобразований требует только 52 умножения вместо 64 для обычного метода вложения. Поэтому, если метод полиномиального преобразования сочетать с алгоритмом Винограда, число комплексных умножений, требуемых для вычисления (7×7)-точечного ДПФ, снижается с 81 до 69. Необходимо заметить, однако, что первый метод полиномиального преобразования, который базируется на редуцированных ДПФ, потребовал бы в этом случае только 65 комплексных умножений. Фактически такой результат носит достаточно общий характер, и первый метод полиномиального преобразования, оперирующий с бо́льшими полями, где бы он ни использовался, обычно предпочтительнее.

Наиболее интересным применением второго полиномиального преобразования, базирующегося на полях меньшей протяженности, является вычисление ДПФ, для которого первый метод неприменим. Это относится к случаям двумерных ДПФ, когда обе размерности не содержат общих множителей, в то время как при соответствующей двумерной корреляции общие множители для обеих размерностей имеются. Например, (7×9)-точечное ДПФ нельзя вычислить эффективно первым методом, так как 7 и 9 не имеют общих множителей. При использовании алгоритма Винограда это ДПФ вычисляется методом вложения 7- и 9-точечных ДПФ. При использовании алгоритма Рейдера 7-точечное ДПФ превращается в одно умножение и одну 6-точечную корреляцию, а 9-точечное ДПФ — в 5 умножений и одну 6-точечную корреляцию. Таким образом, (7×9)-точечное ДПФ вычисляется посредством пяти 7-точечных ДПФ, одной 6- и одной (6×6)-точечных корреляций. Использование полиномиальных преобразований для вычисления (6×6)-точечной корреляции порождает алгоритм, требующий только 174 действительных умножений вместо 198 для обычного алгоритма Винограда.

Чтобы достичь максимальной эффективности при вычислении больших многомерных ДПФ, целесообразно сочетать оба метода полиномиальных преобразований. Если это сделать, например, при (63×63)-точечном ДПФ, то ДПФ вычисляется методом вложения (7×7)- и (9×9)-точечных ДПФ с помощью первого метода полиномиального преобразования, который отображает (7×7)-точечное ДПФ за одно умножение плюс 8 6-точечных корреляций, а (9×9)-точечное ДПФ — за 33 умножения плюс 12 6-точечных корреляций. Таким образом, (63×63)-точечное ДПФ вычисляется за 33 умножения, 276 6-точечных корреляций и 96 (6×6)-точечных корреляций. Если (6×6)-точечные корреляции вычисляются с помощью полиномиальных преобразований, (63×63)-точечное ДПФ вычисляется за 11 344 действительных умножений вместо 13 650 умножений только для первого метода и 19 602 умножений для традиционного алгоритма Винограда. Необходимо заметить, что при совместном применении двух методов полиномиального преобразования число сложений также уменьшается по сравнению с применением только первого метода.

В табл. 3.12 дано число вещественных операций для комплексных двумерных ДПФ, вычисляемых с помощью двух методов полиномиального преобразования. Можно видеть, что даже для преобразований массивов больших размерностей число умножений и сложений может быть очень небольшим [например, (1008×1008)-точечное ДПФ вычисляется посредством 3,39 действительных умножений на отсчет или около одного комплексного умножения на отсчет].

На практике экономия вычислений, полученная с помощью двух методов полиномиального преобразования, может быть весьма значительной, что видно

Таблица 3.12

Число вещественных операций для комплексных двумерных ДПФ, вычисляемых с помощью комбинации двух методов полиномиального преобразования и метода послойного вложения

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
63×63	11 344	193 480	2,86	48,75
80×80	16 944	231 344	2,65	36,15
120×120	35 632	553 392	2,47	38,43
240×240	153 904	2 542 896	2,67	44,15
504×504	726 064	15 621 424	2,86	61,50
1008×1008	3 449 024	71 455 456	3,39	70,33

при сравнении данных табл. 3.11 и 3.12 с данными табл. 3.13. Последние соответствуют двумерным ДПФ, вычисляемым с помощью алгоритма БПФ Рейдера — Бреннера и построчно — столбцового метода. При использовании только первого метода полиномиального преобразования число сложений всегда меньше, чем при использовании алгоритма БПФ, а число умножений для ДПФ массивов больших размерностей уменьшается примерно в два раза. При объединении двух методов полиномиального преобразования число сложений остается таким же, как у алгоритма БПФ, а число умножений уменьшается примерно в четыре раза для ДПФ массивов бо́льших размерностей. Метод полиномиального преобразования также значительно эффективнее, чем алгоритм Винограда преобразования Фурье. Это можно видеть из того, что (1008×1008)-точечное ДПФ, вычисляемое по этому алгоритму, требует 6,25 умножений и 91,61 сложений на отсчет, тогда как первый метод полиномиального преобразования требует 7,67 умножений и 54,58 сложений на отсчет для (1024×1024)-точечного ДПФ, а сочетание двух методов полиномиального преобразования порождает алгоритм только с 3,39 умножениями и 70,33 сложениями на отсчет для (1008×1008)-точечного ДПФ.

Таблица 3.13

Число вещественных операций для комплексных двумерных ДПФ, вычисляемых по алгоритму БПФ Рейдера — Бреннера и построчно-столбцовому методу

Размеры массива	Число умножений	Число сложений	Число умножений на отсчет	Число сложений на отсчет
8×8	64	832	1,00	13,00
16×16	640	4736	2,50	18,50
32×32	4352	27136	4,25	26,50
64×64	25 088	141 312	6,12	34,50
128×128	132 096	696 320	8,06	42,50
256×256	657 408	3 309 568	10,03	50,50
512×512	3 149 824	15 335 424	12,02	58,50
1024×1024	14 688 256	69 730 304	14,01	66,50

3.3.4. Связь между полиномиальными преобразованиями и ДПФ

Из (3.28) -- (3.31) следует, что существование N-точечного полиномиального преобразования, обладающего свойством циклической свертки, зависит только от существования корня N-й степени из единицы и элемента N^{-1} в поле коэффициентов. Эти условия являются общими для всех N-точечных преобразований, обладающих свойством циклической свертки [3.22], и N-точечное полиномиальное преобразование, имеющее корень Z и определенное по модулю $P(Z)$, можно рассматривать как ДПФ, определенное в кольце полиномов по модулю $P(Z)$.

Это свойство аналогично теоретико-числовым преобразованиям, которые можно рассматривать как ДПФ, определенные на кольце целых чисел. Фактически ТЧП являются частными случаями полиномиальных преобразований, в которых N-битовые элементы должны рассматриваться как полиномы. В наибольшей степени это очевидно для 2^{t+1}-точечного полиномиального преобразования, определенного по модулю $Z^{2^t}+1$. Такое преобразование вычисляет 2^{t+1}-точечную циклическую свертку с помощью полиномов по 2^t членов. Если 2^{t+1} входных полиномов определены как элементы по 2^t бит, то полиномиальное преобразование сводится к преобразованию последовательности длиной 2^{t+1} по числам Ферма, с корнем 2 и определенному по модулю $2^{2^t}+1$.

Поскольку полиномиальные преобразования являются ДПФ, определенными на кольцах полиномов, они по существу эквивалентны многомерным ДПФ и поэтому применяются преимущественно для решения многомерных задач. Их главное достоинство по сравнению с ДПФ и ТЧП обусловлено тем, что они систематически используют удобство операций на кольце полиномов для определения простых корней из единицы, что позволяет обойтись без умножений при вычислении преобразования. Для некоторых приложений, таких, как вычисление $(N\times N)$-точечных сверток, полиномиальные преобразования оптимальны в том смысле, что могут порождать алгоритмы с таким числом общих умножений, которое нельзя больше уменьшить чисто алгебраическими методами.

3.4. Заключительные замечания

Полиномиальные преобразования обеспечивают эффективные средства отображения двумерных сверток и ДПФ в одномерные свертки и ДПФ. Эти преобразования представляют особый интерес, когда преобразуемые последовательности имеют общий множитель в обоих измерениях. В этих случаях полиномиальные преобразования обеспечивают существенную экономию в числе арифметических операций по сравнению с более традиционными вычислительными методами.

При вычислении с помощью полиномиальных преобразований двумерные свертки и ДПФ преобразуются в одномерные свертки и ДПФ, которые, в свою очередь, можно вычислить с помощью различных методов, таких, как БПФ, ТЧП, алгоритмы распределенной арифметики или короткой свертки. Тем самым порождается большое разнообразие алгоритмов, среди которых всегда можно найти компромисс между числом умножений и сложений, чтобы использовать их в конкретных приложениях.

По-видимому, наиболее интересны те полиномиальные преобразования, размерность которых равна степени 2. Эти преобразования имеют структуру, по-

добную БПФ, и поэтому могут программироваться аналогичным образом. Комбинируя эти преобразования с традиционными алгоритмами БПФ для вычисления одномерных сверток, полиномиальных произведений и ДПФ, можно получить двумерные алгоритмы, которые основаны на простоте метода БПФ и обеспечивают возможность существенного уменьшения числа операций.

3.5. Приложение. Алгоритмы коротких полиномиальных произведений

Ниже приведены алгоритмы коротких полиномиальных произведений, предназначенные для минимизации числа умножений при сохранении разумно небольшого числа сложений. Здесь $\{x_n\}$, $\{h_l\}$ — входные последовательности, $\{h_l\}$ предполагается заданной и выражения, включающие $\{h_l\}$, вычисляются и запоминаются заранее; $\{y_m\}$ — выходная последовательность. Выражения в скобках обозначают группировку сложений. Входные и выходные сложения должны выполняться в заданной индексно-числовой последовательности.

3.5.1. Полиномиальное произведение по модулю (Z^2+1)

3 умножения, 3 сложения

$$\begin{aligned} a_0&=x_0+x_1 & b_0&=h_0\\ a_1&=x_1 & b_1&=h_0+h_1\\ a_2&=x_0 & b_2&=h_1-h_0 \end{aligned}$$

$$m_k=a_kb_k, \quad k=0, \ldots, 2$$

$$\begin{aligned} y_0&=m_0-m_1\\ y_1&=m_0+m_2 \end{aligned}$$

3.5.2. Полиномиальное произведение по модулю $(Z^3-1)/(Z-1)$

3 умножения, 3 сложения

$$\begin{aligned} a_0&=x_1 & b_0&=h_0-h_1\\ a_1&=x_0-x_1 & b_1&=h_0\\ a_2&=x_0 & b_2&=h_1 \end{aligned}$$

$$m_k=a_kb_k, \quad k=0, 1, 2$$

$$\begin{aligned} y_0&=m_0+m_1\\ y_1&=m_0+m_2 \end{aligned}$$

3.5.3. Полиномиальное произведение по модулю (Z^4+1)

9 умножений, 15 сложений

$$\begin{aligned} a_0&=x_1+x_3 & b_0&=h_0-h_3\\ a_1&=(x_0+x_2)-(x_1+x_3) & b_1&=h_0\\ a_2&=(x_0+x_2) & b_2&=h_0+h_1\\ a_3&=x_3 & b_3&=h_0+h_2+h_1-h_3\\ a_4&=x_2-x_3 & b_4&=h_0+h_2\\ a_5&=x_2 & b_5&=h_0+h_2+h_1+h_3\\ a_6&=x_1 & b_6&=-h_0+h_2+h_1+h_3\\ a_7&=x_0-x_1 & b_7&=-h_0+h_2\\ a_8&=x_0 & b_8&=-h_0+h_2-h_1+h_3 \end{aligned}$$

$$m_k = a_k b_k, \quad k = 0, \ldots, 8$$

$$y_0 = (m_0 + m_1) - (m_3 + m_4) \qquad y_2 = (m_0 + m_1) - (m_6 + m_7)$$
$$y_1 = (m_2 - m_1) + (m_4 - m_5) \qquad y_3 = (m_2 - m_1) + (m_8 - m_7)$$

3.5.4. Полиномиальное произведение по модулю $(Z^5-1)/(Z-1)$

9 умножений, 16 сложений

$$\begin{aligned}
a_0 &= x_0 & b_0 &= h_0 \\
a_1 &= x_1 & b_1 &= h_1 \\
a_2 &= x_0 - x_1 & b_2 &= -h_0 + h_1 \\
a_3 &= x_2 & b_3 &= h_2 \\
a_4 &= x_3 & b_4 &= h_3 \\
a_5 &= x_2 - x_3 & b_5 &= -h_2 + h_3 \\
a_6 &= x_0 - x_2 & b_6 &= h_2 - h_0 \\
a_7 &= x_1 - x_3 & b_7 &= h_3 - h_1 \\
a_8 &= -a_6 + a_7 & b_8 &= b_6 - b_7
\end{aligned}$$

$$m_k = a_k b_k, \quad k = 0, \ldots, 8$$

$$\begin{aligned}
u_0 &= m_0 - m_7 & u_1 &= m_2 + m_0 \\
y_0 &= u_0 - m_1 + m_5 & y_2 &= u_0 - m_4 + m_6 \\
y_1 &= u_1 - m_3 - m_7 & y_3 &= u_1 + m_5 + m_6 + m_8
\end{aligned}$$

3.5.5. Полиномиальное произведение по модулю $(Z^9-1)/(Z^3-1)$

15 умножений, 39 сложений

$$\begin{aligned}
a_0 &= x_0 + x_2 & & \\
a_1 &= x_3 + x_5 & & \\
a_2 &= a_1 + x_4 & b_2 &= (h_0 + 3h_1 + 2h_2 - 2h_3 - 3h_4 - h_5)/6 \\
a_3 &= a_0 + x_1 & b_3 &= (h_0 - h_2 + h_3 + 3h_4 + 2h_5)/6 \\
a_4 &= a_3 - a_2 & b_4 &= b_2 + b_3 \\
a_5 &= a_1 - x_4 & b_5 &= (-h_0 + h_1 - h_4 + h_5)/2 \\
a_6 &= a_0 - x_1 & b_6 &= (h_0 - h_2 - h_3 + h_4)/2 \\
a_7 &= a_6 - a_5 & b_7 &= b_5 + b_6 \\
a_8 &= x_3 & b_8 &= 2h_0 + h_1 - h_2 - 2h_3 + h_5 \\
a_9 &= x_0 - x_3 & b_9 &= 2h_0 - h_2 + h_4 \\
a_{10} &= x_0 & b_{10} &= b_9 - b_8 \\
a_{11} &= x_5 & b_{11} &= h_0 - h_1 - 2h_2 + h_4 \\
a_{12} &= x_2 - x_5 & b_{12} &= -h_1 + h_3 - 2h_5 \\
a_{13} &= x_2 & b_{13} &= b_{12} - b_{11} \\
a_{14} &= -a_{12} + x_0 - x_4 & b_{14} &= (h_0 - h_2 - 2h_3 + 2h_5)/3 \\
a_{15} &= a_9 + x_5 - x_1 & b_{15} &= (-h_0 + h_2 - h_3 + h_5)/3 \\
a_{16} &= -a_{15} + a_{14} & b_{16} &= b_{15} - b_{14}
\end{aligned}$$

$$m_k = a_{k+2} b_{k+2}, \quad k = 0, \ldots, 14$$

$$\begin{aligned}
u_0 &= m_0 + m_1 & u_5 &= m_3 + m_5 \\
u_1 &= m_3 + m_4 & u_6 &= m_{12} + m_{14} \\
u_2 &= m_{13} + m_{14} & u_7 &= -u_3 + m_6 \\
u_3 &= u_0 + u_1 & u_8 &= u_4 + u_5 \\
u_4 &= m_0 + m_2 & u_9 &= m_9 - u_6
\end{aligned}$$

$$\begin{aligned}
y_0 &= m_7 + u_2 + u_7 & y_3 &= u_7 + u_8 + m_8 + u_6 \\
y_1 &= u_8 + m_{10} + u_9 & y_4 &= u_3 + m_{11} + u_9 + u_2 \\
y_2 &= u_4 - u_5 + u_2 & y_5 &= u_0 - u_1 + u_6
\end{aligned}$$

3.5.6. Полиномиальное произведение по модулю $(Z^7-1)/(Z-1)$

15 умножений, 53 сложения

$$
\begin{aligned}
a_0&=x_0+x_2\\
a_1&=a_0+x_1\\
a_2&=a_1+x_2\\
a_3&=x_3+x_5\\
a_4&=a_3+x_4\\
a_5&=a_4+x_5\\
a_6&=x_0\\
a_7&=a_1\\
a_8&=a_0-x_1\\
a_9&=a_2+a_2-x_0\\
a_{10}&=x_2\\
a_{11}&=x_3\\
a_{12}&=a_4\\
a_{13}&=a_3-x_4\\
a_{14}&=a_5+a_5-x_3\\
a_{15}&=x_5\\
a_{16}&=a_{11}-a_6\\
a_{17}&=a_{12}-a_7\\
a_{18}&=a_{13}-a_8\\
a_{19}&=a_{14}-a_9\\
a_{20}&=a_{15}-a_{10}
\end{aligned}
$$

$$
\begin{aligned}
b_0&=(-2h_5+3h_4-h_3-2h_2+h_1+2h_0)/2\\
b_1&=(3h_5-11h_4+10h_3+3h_2-11h_1-4h_0)/14\\
b_2&=(3h_5-h_4-2h_3+3h_2-h_1)/6\\
b_3&=(h_4-h_3+h_1)/6\\
b_4&=-h_5-2h_4+3h_3-h_2-2h_1+h_0\\
b_5&=(h_5+2h_4-h_3-2h_2+3h_1-h_0)/2\\
b_6&=(-11h_5-4h_4+10h_3+3h_2-11h_1+10h_0)/14\\
b_7&=(-h_5-2h_3+3h_2-h_1-2h_0)/6\\
b_8&=(h_5-h_3+h_1-h_0)/6\\
b_9&=-2h_5+h_4+2h_3-h_2-2h_1+3h_0\\
b_{10}&=(2h_4-h_3-2h_2+h_1)/2\\
b_{11}&=(-2h_5-2h_4+12h_3+5h_2-9h_1-2h_0)/14\\
b_{12}&=(-2h_3+3h_2-h_1)/6\\
b_{13}&=(-h_3+h_1)/6\\
b_{14}&=2h_3-h_2-2h_1+h_0
\end{aligned}
$$

$$m_k=a_{k+6}b_k,\quad k=0,\ \ldots,\ 14$$

$$
\begin{aligned}
u_0&=m_5+m_0 & u_{12}&=u_0+u_{11}\\
u_1&=m_6+m_1 & u_{13}&=u_{10}+u_3\\
u_2&=m_7+m_2 & u_{14}&=u_{13}-u_2\\
u_3&=m_8+m_3 & u_{15}&=u_{13}+u_3+u_3+u_4+u_2\\
u_4&=m_9+m_4 & u_{16}&=-u_{12}-u_{15}-u_{14}\\
u_5&=m_{10}+m_0 & u_{17}&=u_7-u_8\\
u_6&=m_{11}+m_1 & u_{18}&=u_5+u_{17}+u_7\\
u_7&=m_{12}+m_2 & u_{19}&=u_{17}-u_6-u_8\\
u_8&=m_{13}+m_3 & u_{20}&=(u_7+u_8)+(u_7+u_8)+u_9\\
u_9&=m_{14}+m_4 & u_{21}&=u_{19}+u_{19}\\
u_{10}&=u_1+u_3 & u_{22}&=u_{21}-u_{18}\\
u_{11}&=u_{10}+u_2 & u_{23}&=u_{21}-u_{20}
\end{aligned}
$$

$$
\begin{aligned}
y_0&=u_{18} & y_3&=u_{14}+u_{21}\\
y_1&=u_{16}+u_{19} & y_4&=u_{12}+u_{22}\\
y_2&=u_{15}+u_{23} & y_5&=u_{20}
\end{aligned}
$$

3.5.7. Полиномиальное произведение по модулю (Z^8+1)

21 умножение, 77 сложений

$$
\begin{aligned}
a_0&=x_0+x_2\\
a_1&=x_1+x_3\\
a_2&=x_0-x_2\\
a_3&=x_7-x_5\\
a_4&=x_4+x_6\\
a_5&=x_5+x_7\\
a_6&=x_4-x_6\\
a_7&=x_1-x_3\\
a_8&=a_0+a_1\\
a_9&=a_4+a_5\\
a_{10}&=a_8+a_9\\
a_{11}&=a_0-a_1
\end{aligned}
$$

$$
\begin{aligned}
b_0&=(h_0+h_1+h_2-h_3+h_4+h_5+h_6+h_7)/4\\
b_1&=(-h_0-h_1-h_2-h_3+h_4+h_5+h_6-h_7)/4\\
b_2&=(h_3-h_4-h_5-h_6)/4\\
b_3&=(5h_0-5h_1+5h_2-7h_3+5h_4-5h_5+5h_6-h_7)
\end{aligned}
$$

$a_{12}=a_4-a_5$
$a_{13}=a_{11}+a_{12}$
$a_{14}=a_2+a_3$
$a_{15}=a_6+a_7$
$a_{16}=a_{14}+a_{15}$
$a_{17}=a_2-a_3$
$a_{18}=a_6-a_7$
$a_{19}=a_{17}+a_{18}$
$a_{20}=x_0+x_4$
$a_{21}=x_0$
$a_{22}=x_4$
$a_{23}=x_3+x_7$
$a_{24}=x_3$
$a_{25}=x_7$
$a_{26}=a_{15}-a_9+x_0-a_{23}$
$a_{27}=a_8-a_{14}+x_3+x_4-x_7$
$a_{28}=a_{26}+a_{27}$

$b_4=(-5h_0+5h_1-5h_2+h_3+5h_4-5h_5+5h_6-7h_7)/20$
$b_5=(3h_3-5h_4+5h_5-5h_6+4h_7)/20$
$b_6=(h_0+h_1-h_2-h_3+h_4-h_5-h_6+3h_7)/4$
$b_7=(-h_0+h_1+h_2-3h_3+h_4+h_5-h_6-h_7)/4$
$b_8=(-h_1+2h_3-h_4+h_6-h_7)/4$
$b_9=(5h_0-5h_1-5h_2+h_3+5h_4+5h_5-5h_6-3h_7)/20$
$b_{10}=(-5h_0-5h_1+5h_2+3h_3+5h_4-5h_5-5h_6+h_7)/20$
$b_{11}=(5h_1-2h_3-5h_4+5h_6+h_7)/20$
$b_{12}=h_0-h_3+h_4$
$b_{13}=-2h_0+h_3-h_7$
$b_{14}=h_3-2h_4+h_7$
$b_{15}=-h_2+h_6-2h_7$
$b_{16}=2h_3-2h_6+2h_7$
$b_{17}=2h_2-2h_3+2h_7$
$b_{18}=(-h_3+h_7)/5$
$b_{19}=(-h_3-h_7)/5$
$b_{20}=h_3/5$

$$m_k=a_{k+8}b_k, \quad k=0, \ldots, 20$$

$u_0=m_0+m_2$
$u_1=m_1+m_2$
$u_2=m_3+m_5$
$u_3=m_4+m_5$
$u_4=m_6+m_8$
$u_5=m_7+m_8$
$u_6=m_{11}+m_9$
$u_7=m_{10}+m_{11}$
$u_8=m_{20}+m_{19}$
$u_9=m_{12}+u_8$
$u_{10}=u_9+m_{13}$
$u_{11}=u_0+u_2$
$u_{12}=u_0-u_2$
$u_{13}=u_1+u_3$
$u_{14}=u_1-u_3$
$u_{15}=u_4+u_6$
$u_{16}=u_4-u_6$
$u_{17}=u_5+u_7$
$u_{18}=u_5-u_7$
$u_{19}=m_{18}-m_{19}$
$u_{20}=u_9+m_{14}$
$u_{21}=u_{19}+u_{19}$
$u_{22}=m_{15}-u_{21}$
$u_{23}=u_{22}+m_{16}$
$u_{24}=-u_{22}-m_{17}$
$u_{25}=u_8+u_8$
$u_{26}=u_{25}+u_{25}$
$u_{27}=u_{19}+u_{25}$

$y_0=u_{13}+u_{17}+u_{20}$
$y_1=u_{12}-u_{18}+u_{23}$
$y_2=u_{11}-u_{15}+u_{25}$
$y_3=u_{12}+u_{18}+u_{27}$
$y_4=u_{11}+u_{15}+u_{10}+u_{19}$
$y_5=-u_{14}-u_{16}+u_{24}+u_{26}$
$y_6=-u_{13}+u_{17}+u_{21}+u_{25}$
$y_7=-u_{14}+u_{16}+u_{19}$

3.6. Приложение. Алгоритмы редуцированного ДПФ для $N=4, 8, 9, 16$

Эти алгоритмы вычисляют $q^{t-1}(q-1)$ выходных отсчетов N-точечных ДПФ $N=q^t$, q — простое, где последние q^{t-1} входных отсчетов $\{x_n\}$ равны нулю:

$$\overline{X}_k=\sum_{n=0}^{q^{t-1}(q-1)-1} x_n W^{nk}, \quad 1 \leqq k \leqq N-1, \quad k \not\equiv 0 \mod q,$$

$$W=\exp -\mathrm{j}\,2\pi/N, \quad \mathrm{j}=\sqrt{-1}.$$

Алгоритмы располагаются в таком же порядке, что и в разд. 3.5. В скобках указано число тривиальных умножений на ± 1, $\pm \mathrm{j}$.

3.6.1. $N=4$

2 умножения (2), 2 сложения

$m_0=1\cdot x_0$
$m_1=-\mathrm{j}\cdot x_1$
$\overline{X}_1=m_0+m_1$
$\overline{X}_3=m_0-m_1$

3.6.2. $N=8$, $u=\pi/4$

4 умножения (2), 10 сложений

$$m_0=1\cdot x_0 \qquad m_1=(x_1-x_3)\cos u$$
$$m_2=-\mathrm{j}\,x_2 \qquad m_3=-\mathrm{j}\,(x_1+x_3)\sin u$$

$$s_1=m_0+m_1 \qquad s_2=m_0-m_1$$
$$s_3=m_2+m_3 \qquad s_4=m_2-m_3$$

$$\bar{X}_1=s_1+s_3, \quad \bar{X}_3=s_2-s_4, \quad \bar{X}_5=s_2+s_4, \quad \bar{X}_7=s_1-s_3.$$

3.6.3. $N=16$, $u=2\pi/16$

10 умножений (2), 32 сложения

$$t_1=x_1+x_7 \qquad t_2=x_1-x_7 \qquad t_3=x_3+x_5 \qquad t_4=x_5-x_3$$
$$m_0=1\cdot x_0 \qquad m_1=(x_2-x_6)\cos 2u \qquad m_2=(t_2+t_4)\cos 3u$$
$$m_3=(\cos u+\cos 3u)\,t_2 \qquad m_4=(\cos 3u-\cos u)\,t_4$$
$$m_5=-\mathrm{j}\cdot x_4 \qquad m_6=-\mathrm{j}\,(x_2+x_6)\sin 2u \qquad m_7=-\mathrm{j}\,(t_1+t_3)\sin 3u$$
$$m_8=\mathrm{j}\,(\sin 3u-\sin u)-t_1 \qquad m_9=-\mathrm{j}\,(\sin u+\sin 3u)\,t_3$$

$$s_1=m_0+m_1 \qquad s_2=m_0-m_1 \qquad s_3=m_3-m_2$$
$$s_4=m_4-m_2 \qquad s_5=s_1+s_3 \qquad s_6=s_1-s_3$$
$$s_7=s_2+s_4 \qquad s_8=s_2-s_4 \qquad s_9=m_5+m_6$$
$$s_{10}=m_5-m_6 \qquad s_{11}=m_7+m_8 \qquad s_{12}=m_7-m_9$$
$$s_{13}=s_9+s_{11} \qquad s_{14}=s_9-s_{11} \qquad s_{15}=s_{10}+s_{12}$$
$$s_{16}=s_{10}-s_{12}$$

$$\bar{X}_1=s_5+s_{13} \qquad \bar{X}_3=s_8-s_{16} \qquad \bar{X}_5=s_7+s_{15}$$
$$\bar{X}_7=s_6+s_{14} \qquad \bar{X}_9=s_6+s_{14} \qquad \bar{X}_{11}=s_7-s_{15}$$
$$\bar{X}_{13}=s_8+s_{16} \qquad \bar{X}_{15}=s_5-s_{13}$$

3.6.4. $N=9$, $u=2\pi/9$

8 умножений, 28 сложений

$$t_1=x_4+x_5 \qquad t_2=x_4-x_5$$

$$m_0=\frac{1}{2}\,(x_0+x_0-x_3) \qquad m_4=-\mathrm{j}\,(\sin 3u)\,x_3$$

$$m_1=\left(\frac{2\cos u-\cos 2u-\cos 4u}{3}\right)(x_1-x_2) \qquad m_5=-\mathrm{j}\,(\sin u)\,(x_1+x_2)$$

$$m_2=\left(\frac{\cos u+\cos 2u-2\cos 4u}{3}\right)(x_2-t_1) \qquad m_6=-\mathrm{j}\,(\sin 4u)\,(x_2+t_2)$$

$$m_3=\left(\frac{\cos u-2\cos 2u+\cos 4u}{3}\right)(t_1-x_1) \qquad m_7=\mathrm{j}\,(\sin 2u)\,(x_1-t_2)$$

$$s_2=m_1+m_2+m_0 \qquad s_3=-m_2+m_3+m_0$$
$$s_4=-m_1-m_3+m_0 \qquad s_5=m_4+m_5+m_6$$
$$s_6=-m_6+m_7+m_4 \qquad s_7=-m_5-m_7+m_4$$

$$\bar{X}_1=s_2+s_5 \qquad \bar{X}_2=s_3-s_6 \qquad \bar{X}_4=s_4+s_7$$
$$\bar{X}_5=s_4-s_7 \qquad \bar{X}_7=s_3+s_6 \qquad \bar{X}_8=s_2-s_5$$

ГЛАВА 4

АЛГОРИТМ ВИНОГРАДА ДЛЯ ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

(Ш. Зохар) [1]

Здесь детально рассматривается новый алгоритм ДПФ, предложенный Ш. Виноградом. Этот алгоритм требует примерно в пять раз меньше умножений, чем алгоритм Кули — Тьюки, и применим к любым порядкам, которые являются произведением взаимно-простых множителей из следующего списка: 2, 3, 4, 5, 7, 8, 9, 16. Алгоритм описан с помощью схем специальной формы — по одной для каждого члена списка. Такие схемы являются удобным, компактным графическим представлением последовательностей арифметических операций в соответствующих частях алгоритма. Использование этих схем вместе с табл. 4.5, 4.6 позволяет относительно легко реализовать алгоритм и оценить его эффективность.

Материал изложен так, что при первом чтении можно опускать значительные части текста.

4.1. Обзор

Со времен открытия алгоритма БПФ [4.1] многих мучил вопрос: «Является ли алгоритм БПФ единственным способом быстрого вычисления ДПФ или все-таки существует более быстрый алгоритм, который еще предстоит найти?» В 1968 г. один ответ на этот вопрос дал Явн [4.2], показав, что число умножений можно уменьшить в два раза при неизменном числе сложений. В 1969 г. значительный шаг на этом пути сделал Виноград [4.3], который разработал алгоритм, снижающий число умножений в алгоритме БПФ по основанию 2 [4.1] примерно в 5 раз. Такое уменьшение числа умножений сопровождается небольшим увеличением или уменьшением числа сложений. В большинстве случаев рост не превышает 20%.

В качестве основы для сравнения алгоритмов здесь и далее будем использовать «номинальную» эффективность алгоритма Кули — Тьюки (БПФ), а именно, число $\mathscr{M}_{\text{КТ}}$ умножений и $\mathscr{A}_{\text{КТ}}$ сложений вещественных чисел при вычислении ДПФ порядка N с комплексными данными [2]:

$$\mathscr{M}_{\text{КТ}} = 2N \log_2 N; \quad \mathscr{A}_{\text{КТ}} = 1{,}5\, \mathscr{M}_{\text{КТ}}. \tag{4.1}$$

[1] Jet Propulsion Lab California Institute of Technology, 4800 Oak Grove Drive Pasadena, CA 9113, USA.

[2] Равенство (4.1) принято как удобная точка отсчета. Следует иметь в виду, что кроме алгоритма Явна имеются и другие варианты БПФ, более эффективные, чем (4.1).

Итак, выбираем (4.1) в качестве основы сравнения при всех N.

Алгоритм Винограда решает упомянутую задачу, используя $\mathscr{M}_{КТ}/5$ умножений вещественных чисел. Рассмотрим возможности и ограничения алгоритма с тем, чтобы читатель, прежде чем приступить к более глубокому изучению, смог оценить пригодность его для своих нужд.

На нынешней стадии разработки алгоритм применим для любых N, удовлетворяющих соотношению

$$N = \prod_{k=1}^{\varkappa} N_k, \tag{4.2}$$

в котором N_k — взаимно-простые множители, взятые из списка

$$N_k = 2, 3, 4, 5, 7, 8, 9, 16. \tag{4.3}$$

Следовательно, максимальное значение N равно $16 \cdot 9 \cdot 7 \cdot 5 = 5040$. Все значения N, удовлетворяющие указанному условию, перечислены в сводной табл. 4.6. В графе G_∞ для каждого значения N приводится фактическое значение выигрыша в числе умножений. В среднем для всех $N > 140$ G_∞ равно примерно 5,5. При таком значительном снижении числа умножений есть основания ожидать, что в большинстве случаев алгоритм будет работать быстрее, чем алгоритм Кули—Тьюки. Чтобы быть более уверенными в этом, надо знать основной системный параметр μ, который является отношением времени выполнения одного вещественного умножения к времени выполнения одного вещественного сложения. (Термин «вещественный» используется как противоположность термина «комплексный», а не «целое», если иметь в виду Фортран). Для очень больших μ (например, в микропроцессорах, программных умножителях) выигрыш в скорости приближается к G_∞ асимптотически. Для меньших значений μ выигрыш будет меньше. Обозначая выигрыш в скорости через G, можно показать, что

$$G(\mu) = G_\infty (\mu + 1{,}5)/(\mu + R), \tag{4.4}$$

где R — параметр, указанный в табл. 4.6 в одной колонке с G_∞. Очевидно, теперь легко вычислить выигрыш в скорости для любой системы и любого допустимого N. Будет показано, что (4.4) основывается только на времени выполнения арифметических операций и что структурная сложность алгоритма Винограда обусловливает замедление его программной реализации.

Основным недостатком нового алгоритма является необходимость в большом объеме памяти. В табл. 4.6 это отражено параметром $\mathscr{M}$. Для обработки комплексных данных надо иметь в памяти массив длиной $1{,}5\mathscr{M}$ вещественных слов. $\mathscr{M}$ изменяется примерно от $2N$ в начальных строках таблицы до $4N$ в последних ее строках. Так, для больших значений N требуется массив $6N$, что на $4N$ больше, чем минимально необходимые $2N$ для запоминания входного вектора. Из полного объема памяти в $1{,}5\mathscr{M}$ веще-

ственных слов $0{,}5\mathscr{M}$ необходимы для запоминания предварительно вычисленных констант. Поскольку не все константы отличаются друг от друга, вероятно, есть возможность снизить эту часть затрат памяти за счет использования более сложных схем адресации.

Другим возможным недостатком нового алгоритма является то, что он может потребовать больше бит на одно слово для сохранения заданной точности по сравнению с алгоритмом Кули — Тьюки. Это обстоятельство более детально рассматривается в разд. 4.9, но в целом проблема заслуживает дальнейшего изучения.

Вывод алгоритма разбит на две части. В первой рассматриваются быстрые алгоритмы ДПФ для малых порядков, значения которых перечислены в (4.3). Алгоритмы образуют набор конструктивных блоков, которые во второй части объединяются в искомые полные алгоритмы ДПФ для порядков N, описываемых выражением (4.2).

Алгоритмы малых порядков из (4.3) являются производными от алгоритмов порядков 2, 4, 6 преобразований другого типа. Эти преобразования называются лево-циркулянтными (ЛЦП) (чаще называются циклической корреляцией; см. разд. 4.2). Итак, излагаемый материал имеет следующую структуру. Раздел 4.2 посвящен взаимосвязи ДПФ — ЛЦП. Далее следует описание трех алгоритмов ЛЦП (разд. 4.3) и построенных на их основе семи алгоритмов ДПФ (разд. 4.4.—4.6). В разд. 4.7 рассматривается объединение этих алгоритмов малого порядка в искомый алгоритм порядка N, удовлетворяющий (4.2). Раздел 4.8 посвящен оценке эффективности. В разд. 4.9 подводятся итоги и даются комментарии по поводу преобразования, выполняемого с оставлением на месте.

Мы постарались сделать рассмотрение достаточно детальным и полным, желая создать необходимую базу для дальнейшей разработки проблемы. Однако следует указать, что приемлемый уровень понимания основных идей и их приложений может быть достигнут без детального вывода алгоритмов малых порядков. При этом достаточно изучить разд. 4.2, первую часть разд. 4.3 (вплоть до рассмотрения лево-циркулянтного преобразования порядка 4), введение вектора η, обобщение схемы в разд. 4.6 [см. фрагмент между (4.126) и (4.127)], разд. 4.7, последнюю часть разд. 4.8 [следующую за (4.181)] и разд. 4.9.

4.2. Основная идея алгоритма

Краеугольным камнем алгоритма Винограда является теорема [4.4], дающая решение, казалось бы, отвлеченной задачи: для заданных полиномов $A(x)$, $B(x)$ определить минимальное число умножений, необходимых для вычисления,

$$\{A(x)\,B(x)\} \bmod C(x), \tag{4.5}$$

где $C(x)$ — заданный монический полином. С ДПФ эту задачу связывают два обстоятельства. Во-первых, можно показать, что матрица ДПФ родственна другому специальному преобразованию, матрица которого является лево-циркулянтной (точное определение приведено ниже). Во-вторых, можно показать, что вычисление этого преобразования идентично вычислению (4.5). Поэтому минимизация числа умножений в (4.5) приводит к минимизации числа умножений при вычислении ДПФ.

Начнем с нескольких необходимых определений. Матрицей Ханкеля называется матрица, в которой значение элемента a_{ij} зависит только от $(i+j)$. В такой матрице элементы, расположенные по любой диагонали, идущей налево вниз, одинаковы. Очевидно, такая матрица полностью определяется своими первой строкой и последним столбцом.

Здесь будем рассматривать матрицу, которая является частным случаем матрицы Ханкеля, а именно, матрицу Ханкеля (порядка n с индексами, лежащими в диапазоне 0,1, ..., n—1), у которой

$$a_{\rho,\, n-1} = a_{0,\, \rho-1} \quad (1 \leqslant \rho \leqslant n-1), \tag{4.6}$$

т. е. последний столбец является тривиальной перестановкой элементов первой строки. Следовательно, такая матрица полностью определяется своей первой строкой. Вторая строка получается из первой циклическим сдвигом влево (элемент, выдвинутый за пределы строки влево, появляется справа), третья строка порождается таким же способом из второй и т. д. Назовем такую матрицу *лево-циркулянтной, или ЛЦ-матрицей*[1]. Подобным образом линейное преобразование, задаваемое такой матрицей, будем называть лево-циркулярным преобразованием (ЛЦП).

В общем виде ЛЦП третьего порядка записывается выражением

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_2 & a_1 & a_0 \\ a_1 & a_0 & a_2 \\ a_0 & a_2 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}. \tag{4.7}$$

Заметим, что (4.7) можно рассматривать как соотношение для получения последовательности (c_0, c_1, c_2) из последовательностей (a_0, a_1, a_2) и (b_0, b_1, b_2). С этой точки зрения $\{c_i\}$ часто называют циклической корреляцией последовательностей $\{a_i\}$ и $\{b_i\}$. С другой стороны, инвертированную последовательность (c_2, c_1, c_0) рассматривают как циклическую свертку последовательностей $\{a_i\}$, $\{b_i\}$.

В большинство последующих математических преобразований входят матрицы, которые не будучи сами ЛЦ-матрицами, содер-

[1] За основу взято понятие циркулянтной матрицы, которое обычно используется для описания матрицы, порождаемой ее первой строкой, циклически сдвигаемой вправо.

жат ЛЦ-подматрицы. Будем называть такие матрицы *квази-левоциркулянтными матрицами* (КЛЦ).

Обратимся к связи между ЛЦП и ДПФ. Как станет очевидным позже, нам нужно рассматривать некоторую тривиальную модификацию ДПФ[1]:

$$W = \exp\left(-\mathrm{i}\frac{2\pi}{N}\right), \tag{4.8}$$

$$F_u = \Omega \sum_{v=0}^{N-1} W^{uv} f_v \quad (u = 0,\ 1,\ \ldots,\ N-1). \tag{4.9}$$

где Ω — произвольная комплексная константа. Когда $\Omega = 1$, то (4.9) превращается в стандартное ДПФ.

Пусть N — порядок ДПФ — удовлетворяет

$$\left.\begin{aligned} &N = p^k \quad (p \text{ — простое, } k \text{ — целое}),\\ &k < \begin{cases} \infty & (p \text{ — нечетное}),\\ 3 & (p = 2). \end{cases} \end{aligned}\right\} \tag{4.10}$$

Покажем теперь, что для таких N (4.9) может быть представлено в виде КЛЦ-матрицы[2]. Переход к этому виду, приведенный в [4.5], основывается на теоретико-числовом понятии примитивного корня. Примитивным корнем g для числа N, удовлетворяющего (4.10), является целое число, целые степени которого, взятые по mod N, порождают все целые числа из интервала $(1, N)$, за исключением чисел, кратных p.

Количество чисел, лежащих в упомянутом интервале и кратных p,

$$N/p = p^{k-1}. \tag{4.11}$$

поэтому количество целых чисел, порождаемых корнем g,

$$n = p^k - p^{k-1} = (p-1)\,p^{k-1}, \tag{4.12}$$

и можно сказать, что последовательность

$$\{g^\rho \bmod N\} \quad (\rho = 0,\ 1,\ \ldots,\ n-1) \tag{4.13}$$

является перестановкой тех целых чисел из интервала $(1, N)$, которые не кратны p.

Используем теперь эти идеи для переобозначения индексов в (4.9). Все индексы, не кратные p, будут представлены в виде (4.13). А именно, обозначая

$$\left.\begin{aligned} r &= g^\rho \bmod N\\ s &= g^\sigma \bmod N \end{aligned}\right\} \ (\rho,\ \sigma = 0,\ 1,\ \ldots,\ n-1), \tag{4.14}$$

определим

$$B_\rho = F_r; \quad b_\sigma = f_s \quad (\rho,\ \sigma = 0,\ 1,\ \ldots,\ n-1). \tag{4.15}$$

---

[1] В этой главе для обозначения $\sqrt{-1}$ используется i, что соответствует обозначению j в других главах.

[2] Это справедливо в более широком диапазоне, чем задает выражение (4.10), однако для наших целей достаточно (4.10).

Теперь с помощью индексов, кратных p, определим

$$B_{(i)} = F_i; \quad b_{(i)} = f_i \quad (i \bmod p = 0). \tag{4.16}$$

Введение заключенного в скобки индекса (i) заслуживает некоторого пояснения. То, что здесь рассматривается, представляет собой некоторое перемешивание (перенумерацию) величин, обозначенных F_r (и f_s). Из предыдущего следует, что N членов, образующих множество $\{F_r\}$, можно разбить на два подмножества: n-элементное множество, в котором r выражается через (4.14), и оставшееся множество $N-n$ элементов, в котором r кратно p. Используя определение (4.15), можно обозначить элементы первого подмножества $B_0, B_1, \ldots, B_{n-1}$. Подобным образом можно было бы определить $B_{n+m} = F_{mp}$, m — целое, в результате чего получили бы удобную нумерацию элементов второго подмножества $B_n, B_{n+1}, \ldots, B_{N-1}$. Однако делать это нет необходимости, и такой подход вызвал бы трудности в разд. 4.6, где множество $\{F_r\}$ должно быть разбито на три подмножества. Выбранный здесь подход основывается на том факте, что индексация элементов второго подмножества может быть совершенно произвольной, лишь бы она обеспечивала надежное опознавание множества, на которое мы ссылаемся. Из (4.16) явно видно, что для индексации элементов второго подмножества выбрано простое копирование индексов соответствующих членов F_r. Индексные скобки в $B_{(i)}$ являются просто признаком того, что этот член является элементом второго подмножества. Заметим, что нулевой индекс появляется в обоих подмножествах. При этом

$$B_0 = F_1; \quad B_{(0)} = F_0. \tag{4.17}$$

Введя необходимые понятия, перейдем к исключению F_u, f_v из (4.9). Для этого разделим сумму (4.9) на две части. В первой части $v = mp$ (m — целое), т. е. $v \bmod p = 0$. Во второй части $v = s$. В результате получим[1]

$$B_{(tp)} = F_{tp} = \Omega \sum_{m=0}^{N-n-1} W^{mtp^2} b_{(mp)} +$$

$$+ \Omega \sum_{\sigma=0}^{n-1} W^{tpg^\sigma} b_\sigma \quad (t = 0, 1, \ldots, N-n-1), \tag{4.18}$$

$$B_\rho = \hat{B}_\rho + B'_\rho \quad (\rho = 0, 1, \ldots, n-1), \tag{4.19}$$

где

$$\hat{B}_\rho = \Omega \sum_{m=0}^{N-n-1} W^{mpg^\rho} b_{(mp)} \quad (\rho = 0, 1, \ldots, n-1), \tag{4.20}$$

$$B'_\rho = \Omega \sum_{\sigma=0}^{n-1} W^{(g^\rho+\sigma)} b_\sigma \quad (\rho = 0, 1, \ldots, n-1). \tag{4.21}$$

---

[1] Здесь используется тот факт, что из (4.8) следует $W^{(m \bmod N)} = W^m$.

Зависимость показателя степени W в (4.21) от $(\rho+\sigma)$ показывает, что (4.21) является преобразованием Ханкеля. Это значит, что если записать (4.21) в матричной форме, используя ρ и σ в качестве индексов строки и столбца соответственно, то матрица, преобразующая b в B', является матрицей Ханкеля. Более того, она является тем частным случаем матриц Ханкеля, которые определены как лево-циркулянтные. Чтобы показать это, заметим, что по условиям (4.6) наша матрица Ханкеля будет лево-циркулянтной, если

$$g^{\rho+n-1} = g^{\rho-1} \bmod N. \tag{4.22}$$

Но поскольку примитивный корень N всегда удовлетворяет[1]

$$g^n = 1 \bmod N, \tag{4.23}$$

очевидно, что (4.22) действительно справедливо и (4.21) является ЛЦП. Итак, перестановки (4.14)—(4.16) преобразуют любую матрицу ДПФ порядка N [N удовлетворяет (4.10)] в матрицу КЛЦ, ЛЦ-часть которой имеет порядок n.

Особый интерес представляет частный случай, когда N — простое число, т. е. $k=1$, $N=p$, так что (4.12) дает

$$n = N-1, \tag{4.24}$$

а последовательность (4.13) является результатом перестановки целых чисел 1, 2, ..., N—1. В этом случае (4.18)—(4.20) упрощаются следующим образом:

$$B_{(0)} = \Omega\left(b_{(0)} + \sum_{\sigma=0}^{n-1} b_\sigma\right), \tag{4.25}$$

$$\hat{B}_\rho = \Omega\, b_{(0)} \quad (\rho = 0,\ 1,\ \ldots,\ n-1). \tag{4.26}$$

Чтобы проиллюстрировать эти идеи, рассмотрим случай $N=7$. Наименьшим положительным примитивным корнем 7 является 3 [4.6]. Непосредственное вычисление дает

ρ	0	1	2	3	4	5
$3^\rho \bmod 7$	1	3	2	6	4	5

(4.27)

Применяя (4.14)—(4.16), приходим к следующему виду:

---

[1] В стандартной трактовке примитивных корней (4.13) обычно заменяется на

$$\{g^\rho \bmod N \quad (\rho = 1,\ 2,\ \ldots,\ N). \tag{4.13'}$$

Сопоставление с (4.23) показывает, что две формулы эквивалентны. Для доказательства (4.23) предположим, что $g^m = 1 \bmod N$ для $m<n$. Отсюда следует, что $g^{m+1} = g^1 \bmod N$, и поэтому два члена последовательности (4.13′) одинаковы. Это противоречит тому, что g — примитивный корень. Поэтому $m=n$.

$$\begin{bmatrix} F_0 \\ F_1 \\ F_3 \\ F_2 \\ F_6 \\ F_4 \\ F_5 \end{bmatrix} = \begin{bmatrix} B_{(0)} \\ B_0 \\ B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \end{bmatrix} = \Omega \left[\begin{array}{c|cccccc} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ \hline W^0 & W^1 & W^3 & W^2 & W^6 & W^4 & W^5 \\ W^0 & W^3 & W^2 & W^6 & W^4 & W^5 & W^1 \\ W^0 & W^2 & W^6 & W^4 & W^5 & W^1 & W^3 \\ W^0 & W^6 & W^4 & W^5 & W^1 & W^3 & W^2 \\ W^0 & W^4 & W^5 & W^1 & W^3 & W^2 & W^6 \\ W^0 & W^5 & W^1 & W^3 & W^2 & W^6 & W^4 \end{array}\right] \begin{bmatrix} b_{(0)} \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} \quad \begin{bmatrix} b_{(0)} \\ b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_3 \\ f_2 \\ f_6 \\ f_4 \\ f_5 \end{bmatrix} \tag{4.28}$$

ЛЦ-структура здесь совершенно очевидна.

Итак, установлена первая связь с задачей (4.5) для N, удовлетворяющих (4.10). Это ограничение на N далее будет ослаблено. Обратимся теперь к второй связи, а именно, к демонстрации того, что вычисление ЛЦП эквивалентно вычислению (4.5). Мы намереваемся установить эту эквивалентность для того, чтобы опираясь на нее, построить алгоритмы некоторых ЛЦ-преобразований общего вида малых порядков. Следует заметить, что ЛЦ-матрица, с которой мы имеем дело, является матрицей, полученной перестановкой подматрицы ДПФ и, будучи таковой, останется функцией одной переменной (W), тогда как ЛЦ-матрица общего вида порядка n является функцией n переменных. Это обстоятельство предполагает дальнейшие упрощения, которые и будут реализованы позже.

Рассматриваемое умножение матриц показано в следующем выражении, где явно просматривается ЛЦ-структура:

$$\begin{bmatrix} t_m \\ t_{m-1} \\ t_{m-2} \\ \vdots \\ t_2 \\ t_1 \\ t_0 \end{bmatrix} \begin{bmatrix} a_m & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 & a_0 \\ a_{m-1} & a_{m-2} & & & & a_0 & a_m \\ a_{m-2} & & & & & & a_{m-1} \\ \vdots & & & & & & \vdots \\ a_2 & & & & & & a_3 \\ a_1 & & & & & a_3 & a_2 \\ a_0 & a_m & a_{m-1} & \cdots & a_3 & a_2 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-2} \\ b_{m-1} \\ b_m \end{bmatrix} \tag{4.29}$$

Введем теперь вспомогательные полиномы (индексы полиномов указывают на их степень).

$$A_m(x) = \sum_{i=0}^{m} a_i x^i, \tag{4.30}$$

$$B_m(x) = \sum_{i=0}^{m} b_i x^i, \tag{4.31}$$

$$T_m(x) = \sum_{i=0}^{m} t_i x^i. \tag{4.32}$$

Рассмотрим произведение полиномов

$$V_{2m}(x) = A_m(x)\, B_m(x) = \sum_{i=0}^{2m} v_i\, x^i. \tag{4.33}$$

Легко увидеть, что коэффициенты этого полинома можно получить перемножением матриц (4.34).

$$\begin{bmatrix} v_{2m} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ v_{m+1} \\ v_m \\ v_{m-1} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ v_0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & a_m \\ 0 & 0 & & & & a_m & a_{m-1} \\ 0 & & & & & & a_{m-2} \\ \vdots & & & & & & \vdots \\ 0 & & & & & & a_2 \\ 0 & a_m & & & & a_2 & a_1 \\ a_m & a_{m-1} & a_{m-2} & \dots & a_2 & a_1 & a_0 \\ a_{m-1} & a_{m-2} & & & & a_0 & 0 \\ a_{m-2} & & & & & & 0 \\ \vdots & & & & & & \vdots \\ a_2 & & & & & & 0 \\ a_1 & a_0 & & & & 0 & 0 \\ a_0 & 0 & 0 & \dots & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-2} \\ b_{m-1} \\ b_m \end{bmatrix} \tag{4.34}$$

Сравнивая (4.34) с (4.29), видим, что перестановка двух отмеченных треугольных секций преобразует матрицу (4.34) в тривиальное расширение матрицы (4.29). Этот факт можно отобразить следующими соотношениями между полиномами $T_m(x)$ и $V_{2m}(x)$:

$$\begin{aligned} T_m(x) = V_{2m}(x) &- (a_m b_1 + a_{m-1} b_2 + \dots + a_1 b_m)(x^{m+1} - 1) - \\ &- (a_m b_2 + \dots + a_2 b_m)\, x\, (x^{m+1} - 1) - \\ &\cdot \\ &\cdot \\ &\cdot \\ &- (a_m b_{m-1} + a_{m-1} b_m)\, x^{m-2}\, (x^{m+1} - 1) - \\ &- a_m b_m\, x^{m-1}\, (x^{m+1} - 1), \end{aligned} \tag{4.35}$$

$$T_m(x) = V_{2m}(x) - (x^{m+1} - 1)\, F_{m-1}(x), \tag{4.36}$$

где $F_{m-1}(x)$ оказывается полиномом степени $(m-1)$. Следовательно,

$$V_{2m}(x)/(x^{m+1} - 1) = F_{m-1}(x) + T_m(x)/(x^{m+1} - 1). \tag{4.37}$$

Заметим, что в стоящей справа дроби степень числителя меньше степени знаменателя. Это означает, что $T_m(x)$ является остатком от деления V_{2m} на $(x^{m+1}-1)$. Другими словами,

$$T_m(x) = \{A_m(x)\, B_m(x)\} \bmod (x^n - 1). \tag{4.38}$$

Здесь используется тот факт, что порядок ЛЦ-матрицы в (4.29)

$$n = m + 1. \tag{4.39}$$

Равенство (4.38) показывает, что ЛЦП из (4.29) можно выполнить путем перемножения полиномов, идентичного перемножению в (4.5). Это определяет вторую связь.

Теперь рассмотрим число умножений, необходимых для вычисления (4.29). Прямое умножение матриц требует n^2 скалярных умножений. Значительно меньшее значение следует из теоремы Винограда [4.4]. В более узкой трактовке применительно к данному случаю теорема устанавливает, что, если $(x^n - 1)$ представим в виде

$$x^n - 1 = \prod_{i=1}^{k(n)} m_i(x), \tag{4.40}$$

где $m_i(x)$ — различные полиномы, несократимые над полем рациональных чисел, то минимальное число умножений равно $(2n - k)$, вследствие того, что умножения на рациональные числа не учитываются.

Возможность исключения умножений на рациональные числа заслуживает пояснения. Предположим, имеем минимальную реализацию $T_m(x)$ в следующем виде:

$$T_m(x) = \sum_{r=1}^{R} \left(\frac{J_r}{K_r} \right) F_r(x), \tag{4.41}$$

где J_r, K_r — целые числа и $F_r(x)$ не содержит никаких рациональных умножений. В соответствии с теоремой Винограда полиномы F_r потребуют в общей сложности $(2n - k)$ умножений и дополнительные R рациональные умножения, появившиеся в (4.41), не учитываются. Чтобы увидеть, что же включается в число умножений, поясним дробь в (4.41). Пусть K — наименьший общий знаменатель и

$$J_r/K_r = J'_r/K. \tag{4.42}$$

Тогда

$$K T_m(x) = \sum_{r=1}^{R} J'_r F_r(x). \tag{4.43}$$

Каждое умножение на J'_r может быть реализовано как $(J'_r - 1)$ сложений, так что $KT_m(x)$ не требует умножений сверх $(2n - k)$, используемых при вычислении F_r. Наконец, компенсируем умножение на K слева предварительным масштабированием матрицы, a, т. е. заменяем a_i в $A_m(x)$ на

$$\hat{a}_i = a_i/K, \quad \hat{A}_m(x) = \sum_{i=0}^{m} \hat{a}_i x^i. \tag{4.44}$$

Таким образом, вместо (4.38) теперь имеем

$$T_m(x) = K\{\hat{A}_m(x)\, B_m(x)\} \bmod (x^n - 1), \qquad (4.45)$$

и отсюда видно, что умножения на рациональные числа могут быть устранены без увеличения числа иррациональных умножений.

Стоит отметить, что, хотя с чисто теоретической точки зрения все верно, практически следует учитывать затраты в виде дополнительных сложений, вводимых для исключения рациональных умножений. Очевидно, что, когда эти затраты превышают затраты на умножения, которые они заменили, лучше оставить умножения. Такая ситуация действительно возникает при больших n. Поэтому алгоритм, использующий достоинства теоремы Винограда, должен быть сконструирован так, чтобы ДПФ большого порядка разбивалось на ЛЦ-преобразования малых порядков. Фактически все порядки ДПФ, появляющиеся в табл. 4.6 ($N_{\max}=5040$), реализуются посредством трех ЛЦ-преобразований порядков 2, 4, 6.

Факторизация (x^n-1) для этих трех случаев показана в табл. 4.1, в последнем столбце которой указано минимальное число умножений по теореме Винограда. В следующем разделе разберем конкретные алгоритмы, обеспечивающие достижение такого минимума. Эти три алгоритма служат основой при последующем построении алгоритмов для всех порядков, перечисленных в (4.3).

Таблица 4.1

Рациональная факторизация (x^n-1)

n	x^n-1	k	$2n-k$
2	$(x-1)(x+1)$	2	2
4	$(x-1)(x+1)(x^2+1)$	3	5
6	$(x-1)(x+1)(x^2+x+1)(x^2-x+1)$	4	8

4.3. Базовые алгоритмы лево-циркулянтного преобразования

Попытаемся прежде всего достаточно детально представить общий метод, с тем чтобы алгоритмы для трех указанных значений n получились просто, как его очевидное следствие.

Отправной точкой является выражение (4.45), в котором K остается неопределенным до самого конца вывода. Факторизация (x^n-1) приведена в табл. 4.1, где указаны множители $m_i(x)$ из (4.40). Зная m_i, находим $T_m(x)$ по двухфазной схеме, основанной на полиномиальном варианте китайской теоремы об остатках [4.7]. На фазе 1 вычисляем для всех i из (4.40)

$$u_i(x) = \{\hat{A}_m(x)\, B_m(x)\} \bmod m_i(x). \qquad (4.46)$$

Эта часть основана целиком на результатах, вывод которых приведен в приложении и которые представлены в табл. П.1, П.2. На

фазе 2 для формирования $T_m(x)$ из u_i используется полиномиальный вариант алгоритма Гарнера [4.7]. Для этого требуются вспомогательные функции $c_{ij}(x)$, $v_i(x)$, введенные ниже. Их использование при формировании $T_m(x)$ будет видно из (4.50):

$$[m_i(x)\,c_{ij}(x)] \bmod m_j(x) = 1 \quad [\text{определение } c_{ij}(x)], \tag{4.47}$$

$$v_1(x) = u_1(x), \tag{4.48}$$

$$v_i(x) = \{[\ldots[[[u_i(x) - v_1(x)]\,c_{1,\,i}(x) - v_2(x)]\,c_{2,\,i}(x) - v_3(x)]\,c_{3,\,i}(x) - \ldots, - v_{i-1}(x)]\,c_{i-1,\,i}(x)\} \bmod m_i(x), \tag{4.49}$$

$$T_m(x) = K\left[v_1(x) + \sum_{i=2}^{k(m+1)} v_i(x) \prod_{j=1}^{i-1} m_j(x)\right] (k \text{ из табл. 4.1}). \tag{4.50}$$

Если степень $m_j(x)$ равна 1, т. е. $m_j(x) = x - x_j$, вычисление $c_{ij}(x)$ тривиально. Из (П.5) видно, что в этом случае

$$m_i(x_j)\,c_{ij}(x_j) = 1. \tag{4.51}$$

Теперь можно вычислить любые $c_{ij}(x)$, удовлетворяющие (4.47) и, следовательно, (4.51). Выбирая наименьшую степень, получаем

$$c_{ij}(x) = c_{ij} = [m_i(x_j)]^{-1}. \tag{4.52}$$

Разобрав принципы вывода алгоритмов, обратимся теперь к частным случаям. Чтобы установить общую картину, будем придерживаться сформулированных принципов даже при рассмотрении случая самого низкого порядка ($n=2$), когда вывод мог быть много проще.

4.3.1. Лево-циркулянтное преобразование порядка 2

$$\left.\begin{aligned} &\begin{bmatrix} t_1 \\ t_0 \end{bmatrix} = \begin{bmatrix} a_1 & a_0 \\ a_0 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \\ &\hat{a}_i = a_i/K; \quad \hat{A}_1(x) = \sum_{i=0}^{1} \hat{a}_i x^i; \quad B_1(x) = \sum_{i=0}^{1} b_i x^i \end{aligned}\right\} \tag{4.53}$$

Фаза 1

$$T_1(x) = K\{\hat{A}_1(x)\,B_1(x)\} \bmod [\underbrace{(x-1)}_{m_1}\,\underbrace{(x+1)}_{m_2}], \tag{4.54}$$

$$u_1(x) = \{\hat{A}_1(x)\,B_1(x)\} \bmod (x-1) =$$

$$= \underbrace{(\hat{a}_0 + \hat{a}_1)}_{\alpha_1}\,\underbrace{(b_0 + b_1)}_{\beta_1} \quad [\text{см. (П. 5)}],$$

$$\delta_1 = u_1 = \alpha_1 \beta_1,$$

$$u_2(x) = \{\hat{A}_1(x)\,B_1(x)\} \bmod (x+1) = \underbrace{(\hat{a}_0 - \hat{a}_1)}_{\alpha_2}\,\underbrace{(b_0 - b_1)}_{\beta_2},$$

$$\delta_2 = u_2 = \alpha_2 \beta_2.$$

Фаза 2

$$c_{12}(x)=\frac{1}{m_1(x_2)}=\frac{1}{m_1(-1)}=-\frac{1}{2},\quad v_1=\delta_1,$$

$$v_2=\left|(\delta_2-\delta_1)\left(-\frac{1}{2}\right)\right| \bmod (x+1)=\frac{1}{2}(\delta_1-\delta_2),$$

$$T_1(x)=K\left[\delta_1+\frac{1}{2}(\delta_1-\delta_2)(x-1)\right]=$$

$$=\underbrace{\left[\frac{K}{2}(\delta_1-\delta_2)\right]}_{t_1}x+\underbrace{\left[\frac{K}{2}(\delta_1+\delta_2)\right]}_{t_0}. \qquad (4.55)$$

Выбирая $K=2$, получаем $t_0=\delta_1+\delta_2$, $t_1=\delta_1-\delta_2$. Окончательный вид алгоритма представлен графически на рис. 4.1. Способ изображения, которого мы придерживаемся, достаточно прост, поэтому будем использовать его и в более сложных таблицах. β_i обозначает линейную комбинацию b_j; строка β_i содержит (ненулевые) коэффициенты этой линейной комбинации. Аналогично, t_i — линейные комбинации δ_j; в столбце t_i приведены коэффициенты этих линейных комбинаций. Стрелка в правом углу показывает, что β_i порождаются только значениями b_j. Обычно нет какой-либо неопределенности в отношении этого направления, так что стрелки можно опустить. Равенства $\delta_i=\beta_i\alpha_i$ указаны на изображении с использованием символики для умножения, принятой в Фортране.

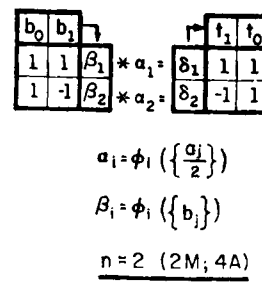


Рис. 4.1. Алгоритм ЛЦП порядка 2

Напомним, наконец, что основное выражение (4.45) полностью симметрично относительно $\{\hat{a}_j\}$ и $\{b_j\}$. Это удобное свойство использовано в принятых здесь обозначениях. Итак, наряду с каждым β_i, которое является функцией $\{b_j\}$ (скажем, $\varphi_i(\{b_j\})$), определяемой таблицей, имеется переменная α_i, которая является точно такой же функцией $\{\hat{a}_j\}$:

$$\alpha_i=\varphi_i(\{a_j/K\}).$$

Это обстоятельство будет учитываться во всей главе. Практически это означает, что через левую часть таблицы необходимо пройти дважды. Во-первых, подставляем $\{a_j/K\}$ вместо $\{b_j\}$ и получаем таким образом $\{\alpha_i\}$ вместо $\{\beta_i\}$; затем проходим всю таблицу, задавая на входе $\{b_j\}$. Заметим, однако, что несмотря на математическую симметрию между $\{a_j/K\}$ и $\{b_j\}$, на практике между ними имеется важное различие: a рассматривается как матрица констант, преобразующая ряд различных векторов данных $\mathbf{b}$. Поэтому значения α_i можно предварительно вычислить раз и навсегда, а затраты на их вычисление при подсчете стоимости преобразования одного вектора данных полностью игнорировать.

Имея это в виду, учитываем только те арифметические операции, которые явно указаны в схеме алгоритма. В нашем случае это 2 умножения и 4 сложения, для которых выбраны обозначения (2М и 4А), приведенные на рис. 4.1. Заметим, что 2 умножения — это тот минимум, который определяется теоремой Винограда (см. правую колонку в табл. 4.1).

4.3.2. Лево-циркулянтное преобразование порядка 4

Относительно всех таблиц алгоритмов, разрабатываемых в этой главе, предполагаем, что читатель будет прослеживать таблицу алгоритма и отмечать графическое представление каждого математического утверждения по ходу построения алгоритма (рис. 4.2)

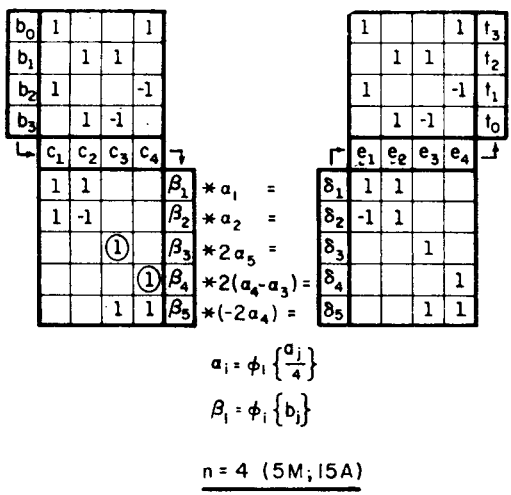


Рис. 4.2. Алгоритм ЛЦП порядка 4

$$\begin{bmatrix} t_3 \\ t_2 \\ t_1 \\ t_0 \end{bmatrix} = \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \\ a_2 & a_1 & a_0 & a_3 \\ a_1 & a_0 & a_3 & a_2 \\ a_0 & a_3 & a_2 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad \left| \begin{array}{l} \hat{a}_i = \dfrac{a_i}{K} \\ \hat{A}_3(x) = \sum\limits_{i=0}^{3} \hat{a}_i x^i \\ B_3(x) = \sum\limits_{i=0}^{3} b_i x^i \end{array} \right. . \qquad (4.56)$$

Фаза 1

$$T_3(x) = K\{\hat{A}_3(x) B_3(x)\} \bmod [\underbrace{(x^2+1)}_{m_1} \underbrace{(x+1)}_{m_2} \underbrace{(x-1)}_{m_3}], \qquad (4.57)$$

$$u_3(x) = \hat{A}_3(1)\,B_3(1) = \underbrace{(\hat{a}_0+\hat{a}_1+\hat{a}_2+\hat{a}_3)}_{\alpha_1}\,\underbrace{(b_0+b_1+b_2+b_3)}_{\beta_1},$$

$$c_1 = b_0+b_2, \quad c_2 = b_1+b_3,$$

$$\therefore \beta_1 = c_1+c_2, \quad \delta_1 = u_3 = \alpha_1\beta_1,$$

$$u_2(x) = \hat{A}_3(-1)\,B_3(-1) = \underbrace{(\hat{a}_0-\hat{a}_1+\hat{a}_2-\hat{a}_3)}_{\alpha_2}\,\underbrace{(b_0-b_1+b_2-b_3)}_{\beta_2},$$

$$\therefore \beta_2 = c_1-c_2, \quad \delta_2 = u_2 = \alpha_2\beta_2.$$

Величина $u_1(x)$ находится за два шага:

$$\left.\begin{aligned} \hat{\boldsymbol{A}}_3(x) \bmod (x^2+1) &= \underbrace{(\hat{a}_1-\hat{a}_3)}_{\alpha_3}\,x + \underbrace{(\hat{a}_0-\hat{a}_2)}_{\alpha_4}, \\ \boldsymbol{B}_3(x) \bmod (x^2+1) &= \underbrace{(b_1-b_3)}_{\beta_3}\,x + \underbrace{(b_0-b_1)}_{\beta_4}. \end{aligned}\right\} \text{(из табл. П.1)} \qquad (4.58)$$

Заметим, что схема алгоритма на рис. 4.2 определяет β_3, β_4 косвенно через $c_3=b_1-b_3$, $c_4=b_0-b_2$. Поэтому $\beta_3=c_3$, $\beta_4=c_4$, так что какие-либо арифметические операции здесь отсутствуют. Везде, где такая ситуация повторится, мы, чтобы обратить на нее внимание, будем отмечать кружочком соответствующий член в таблице.

Теперь объединим оба результата из (4.58), используя табл. П.2 ($\alpha_3=p_1$, $\beta_3=q_1$ и т. д.):

$$u_1(x) = [-(\alpha_4-\alpha_3)\beta_4 + \alpha_1(\beta_3+\beta_4)]\,x + [\alpha_4(\beta_3+\beta_4) - (\alpha_3+\alpha_4)\beta_3].$$

Обозначим

$$\alpha_5 = \alpha_3+\alpha_4, \quad \beta_5 = \beta_3+\beta_4 = c_3+c_4, \quad \delta_3 = 2\alpha_5\beta_5, \quad \delta_4 = 2(\alpha_4-\alpha_3)\beta_4,$$

$$\delta_5 = -2\alpha_4\beta_5.$$

Следовательно, $$u_1(x) = -\frac{1}{2}\underbrace{(\delta_4+\delta_5)}_{e_4}\,x - \frac{1}{2}\underbrace{(\delta_5+\delta_3)}_{e_3}.$$

Фаза 2

$$c_{12} = \frac{1}{m_1(x_2)} = \frac{1}{m_1(-1)} = \frac{1}{2}, \quad c_{13} = \frac{1}{m_1(x_3)} = \frac{1}{m_1(1)} = \frac{1}{2},$$

$$c_{23} = \frac{1}{m_2(x_3)} = \frac{1}{m_2(1)} = \frac{1}{2};$$

$$v_1(x) = u_1(x) = -\frac{1}{2}(e_4x+e_3),$$

$$v_2 = \left\{\left(\delta_2 + \frac{1}{2}e_4x + \frac{1}{2}e_3\right)\frac{1}{2}\right\} \bmod (x+1) = \frac{1}{4}(2\delta_2 - e_4 + e_3),$$

$$v_3 = \left\{\left[\left(\delta_1 + \frac{1}{2}e_4x + \frac{1}{2}e_3\right)\frac{1}{2} - \frac{1}{4}(2\delta_2 - e_4 + e_3)\right]\frac{1}{2}\right\} \bmod (x-1) =$$

$$= \frac{1}{4}(\delta_1 - \delta_2 + e_4),$$

$$T_3(x) = -\frac{K}{2}(e_4 x + e_3) + \frac{K}{4}(2\delta_2 + e_4 + e_3)(x^2+1) +$$

$$+\frac{K}{4}(\delta_1 - \delta_2 + e_4)(x^3 + x^2 + x + 1).$$

При $K=4$

$$T_3(x) = (\delta_1 - \delta_2 + e_4)x^3 + (\delta_1 + \delta_2 + e_3)x^2 +$$
$$+(\delta_1 - \delta_2 - e_4)x + (\delta_1 + \delta_2 - e_3).$$

Введя теперь $e_1 = \delta_1 - \delta_2$, $e_2 = \delta_1 + \delta_2$, окончательно получим

$$T_3(x) = (e_1 + e_4)x^3 + (e_2 + e_3)x^2 + (e_1 - e_4)x + (e_2 - e_3). \qquad (4.59)$$

Подсчет по изображению алгоритма числа арифметических операций (исключая манипуляции с α_i) дает 5 умножений и 15 сложений[1], что снова соответствует минимальному числу умножений, указанному в табл. 4.1.

4.3.3. Лево-циркулянтное преобразование порядка 6

$$\begin{bmatrix} t_5 \\ t_4 \\ t_3 \\ t_2 \\ t_1 \\ t_0 \end{bmatrix} = \begin{bmatrix} a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ a_4 & a_3 & a_2 & a_1 & a_0 & a_5 \\ a_3 & a_2 & a_1 & a_0 & a_5 & a_4 \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 \\ a_1 & a_0 & a_5 & a_4 & a_3 & a_2 \\ a_0 & a_5 & a_4 & a_3 & a_2 & a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix} \quad \begin{array}{l} \hat{a}_i = a_i/K \\ \hat{A}_5(x) = \sum_{i=0}^{5} \hat{a}_i x^i \\ B_5(x) = \sum_{i=0}^{5} b_i x^i \end{array} \Bigg\}. \qquad (4.60)$$

Фаза 1

$$T_5(x) = K\{\hat{A}_5(x) B_5(x)\} \bmod \{\underbrace{(x^2-x+1)}_{m_1}\underbrace{(x^2+x+1)}_{m_2}\underbrace{(x+1)}_{m_3}\underbrace{(x-1)}_{m_4}\}, \qquad (4.61)$$

$$u_1 = \hat{A}_5(1) B_5(1) = \left(\sum_{i=0}^{5} \hat{a}_i\right)\left(\sum_{i=0}^{5} b_i\right),$$

$$c_1 = b_3 + b_0, \quad c_2 = b_5 + b_2, \quad c_3 = b_1 + b_1,$$

$$u_1 = \underbrace{\left(\sum_{i=0}^{5} \hat{a}_i\right)}_{\alpha_1}\underbrace{(c_1 + c_2 + c_3)}_{\beta_1}, \quad \delta_1 = u_1 = \alpha_1 \beta_1,$$

$$u_3 = \hat{A}_5(-1) B_5(-1) = (\underbrace{-\hat{a}_0 + \hat{a}_1 - \hat{a}_2 + \hat{a}_3 - \hat{a}_4 + \hat{a}_5}_{\alpha_6}) \times$$

$$\times (\underbrace{-b_0 + b_1 - b_2 + b_3 - b_4 + b_5}_{\beta_6}),$$

---

[1] Автор выражает признательность д-ру Р. Дж. Липсу, работающему в Лаборатории реактивного движения, за помощь в уменьшении числа сложений с 16 до 15.

$$c_6 = b_3 - b_0, \quad c_5 = b_5 - b_2, \quad c_4 = b_4 - b_1,$$
$$u_3 = \alpha_6 \underbrace{(c_6 + c_5 - c_4)}_{\beta_6}, \quad \delta_6 = u_3 = \alpha_6 \beta_6.$$

Выражения для $u_1(x_1)$, $u_2(x)$ получаем за два шага:

$$B_5(x) \bmod (x^2 + x + 1) = [\underbrace{(b_4 + b_1)}_{c_3} - \underbrace{(b_5 + b_2)}_{c_2}]\, x + [\underbrace{(b_3 + b_0)}_{c_1} - \underbrace{(b_5 + b_2)}_{c_2}] =$$
$$= \underbrace{(c_3 - c_2)}_{\beta_2} x + \underbrace{(c_1 - c_2)}_{\beta_3} = \beta_2 x + \beta_3 \quad \text{(из табл. П.1)},$$
$$\therefore \hat{A}_5(x) \bmod (x^2 + x + 1) = \alpha_2 x + \alpha_3,$$
$$\therefore u_2(x) = \{\hat{A}_5(x)\, B_5(x)\} \bmod (x^2 + x + 1) =$$
$$= [\alpha_3 b_3 - (\alpha_3 - \alpha_2)(\beta_3 - \beta_2)]^{1}_{0} x + (\alpha_3 \beta_3 - \alpha_2 \beta_2) \text{ (из табл. П.2)}. \quad (4.62)$$

Теперь введем

$$\left.\begin{aligned} \beta_7 &= \beta_3 - \beta_2 = c_1 - c_2 + c_2 - c_3 = c_1 - c_3, \\ \alpha_7 &= \alpha_3 - \alpha_2 \end{aligned}\right\} \quad (4.63)$$

и используем эти выражения для преобразования (4.62)[1]:

$$u_2(x) = \underbrace{(\alpha_2 \beta_3 + \alpha_7 \beta_2)}_{-e_3} x + \underbrace{(\alpha_7 \beta_2 + \alpha_3 \beta_7)}_{e_2}; \quad (4.64)$$

$$B_5(x) \bmod (x^2 - x + 1) = -[\underbrace{(b_5 - b_2)}_{c_5} + \underbrace{(b_4 - b_1)}_{c_4}]\, x -$$
$$- [\underbrace{(b_3 - b_0)}_{c_6} - \underbrace{(b_5 - b_2)}_{c_5}] = -\underbrace{(c_5 + c_4)}_{\beta_5} x - \underbrace{(c_6 - c_5)}_{\beta_4} =$$
$$= -\beta_5 x - \beta_4 \quad \text{(из табл. П.1)},$$
$$\therefore \hat{A}_5(x) \bmod (x^2 - x + 1) = -\alpha_5 x - \alpha_4,$$
$$\therefore u_1(x) = \{\hat{A}_5(x)\, B_5(x)\} \bmod (x^2 - x + 1) =$$
$$= [(\alpha_4 + \alpha_5)(\beta_4 + \beta_5) - \alpha_4 \beta_4]\, x + (\alpha_4 \beta_4 - \alpha_5 \beta_5) \quad \text{(из табл. П.2)}. \quad (4.65)$$

Введем

$$\left.\begin{aligned} \beta_8 &= \beta_5 + \beta_4 = c_5 + c_4 + c_6 - c_5 = c_4 + c_6, \\ \alpha_8 &= \alpha_5 + \alpha_4 \end{aligned}\right\} \quad (4.66)$$

и преобразуем (4.65):

$$u_1(x) = \underbrace{(\alpha_5 \beta_4 + \alpha_8 \beta_5)}_{e_4} x + (\underbrace{\alpha_4 \beta_8 - \alpha_8 \beta_5}_{}). \quad (4.67)$$

---

[1] Введенные здесь e_i не появляются в окончательных изображениях алгоритмов :: используются только на промежуточных шагах.

Фаза 2

$$\{m_1(x)\,c_{12}(x)\} \bmod m_2(x) = 1.$$

Пусть $c_{12}(x) = \gamma_1 x + \gamma_0$,

$$\therefore m_1(x)\,c_{12}(x) = (x^2 - x + 1)(\gamma_1 x + \gamma_0) =$$

$$= \gamma_1 x^3 + (\gamma_0 - \gamma_1) x^2 + (\gamma_1 - \gamma_0) x + \gamma_0,$$

$$\therefore (\gamma_1 - \gamma_0 - \gamma_0 + \gamma_1) x + (\gamma_0 - \gamma_0 + \gamma_1 + \gamma_1) = 1 \quad \text{(из табл. П.1)}$$

$$\therefore \gamma_1 = \gamma_0 = 1/2, \quad c_{12}(x) = (x+1)/2,$$

$$c_{13} = \frac{1}{m_1(x_3)} = \frac{1}{m_1(-1)} = \frac{1}{3}, \quad c_{23} = \frac{1}{m_2(x_3)} = \frac{1}{m_2(-1)} = 1,$$

$$c_{14} = \frac{1}{m_1(x_4)} = \frac{1}{m_1(1)} = 1, \quad c_{24} = \frac{1}{m_2(x_4)} = \frac{1}{m_2(1)} = \frac{1}{3},$$

$$c_{34} = \frac{1}{m_3(x_4)} = \frac{1}{m_3(1)} = \frac{1}{2},$$

$$v_1 = e_4 x + e_5,$$

$$v_2 = \left[\left(-e_3 x + e_2 - e_4 x - e_5\right)\frac{1}{2}(x+1)\right] \bmod (x^2 + x + 1),$$

$$v_2 = \frac{1}{2}\left[-(e_4 + e_3) x^2 + (-e_4 - e_3 + e_2 - e_5) x + \right.$$

$$\left. + (e_2 - e_3)\right] \bmod (x^2 + x + 1),$$

$$v_2 = \frac{1}{2}\left[(e_2 - e_5) x + (e_2 + e_3 + e_4 - e_5)\right] \text{ (из табл. П.1)},$$

$$v_3 = \left\{\frac{1}{3}(\delta_6 - e_4 x - e_5) - \frac{1}{2}\left[(e_2 - e_5) x + (e_2 + e_3 + e_4 - e_5)\right]\right\} \bmod (x+1) =$$

$$= \frac{1}{6}(-3e_3 - e_4 - 2e_5 + 2\delta_6),$$

$$v_4 = \frac{1}{2}\left\{\left\langle(\delta_1 - e_4 x - e_5) - \frac{1}{2}\left[(e_2 - e_5) x + (e_2 + e_3 + e_4 - e_5)\right]\right\rangle \frac{1}{3} - \right.$$

$$\left. - \frac{1}{6}(-3e_3 - e_4 - 2e_5 + 2\delta_6)\right\} \bmod (x-1) =$$

$$= \frac{1}{6}(\delta_1 - e_2 + e_3 - e_1 + e_5 - \delta_6),$$

$$T_5(x) = K(e_4 x + e_5) + \frac{K}{2}\left[(e_2 - e_5) x + (e_2 + e_3 + e_4 - e_5)\right](x^2 - x + 1) +$$

$$+ \frac{K}{6}(-3e_3 - e_4 - 2e_5 + 2\delta_6)(x^4 + x^2 + 1) +$$

$$+ \frac{K}{6}(\delta_1 - e_2 + e_3 - e_4 + e_5 - \delta_6)(x^5 + x^4 + x^3 + x^2 + x + 1).$$

Собрав члены с одинаковыми степенями, получим искомые значения для t_i. Подставим теперь $K=6$ и представим зависимость t_i от e_i и δ_i в виде таблицы на рис. 4.3*а*.

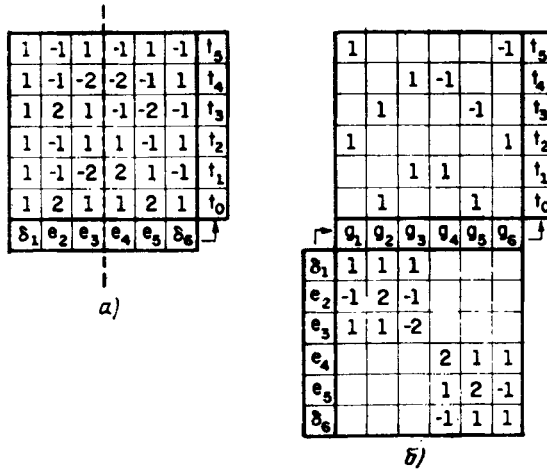


Рис. 4.3. Этапы разработки алгоритма ЛЦП порядка 6

До сих пор вывод алгоритма представлял собой практически непосредственное применение основной идеи, высказанной в начале раздела. Однако полный алгоритм, основанный на рис. 4.3*а*, требует слишком много сложений. Чтобы избавиться от некоторых из них, придется прибегнуть к менее очевидным преобразованиям. Изучая внимательно рис. 4.3*а*, можно заметить, что коэффициенты у t_0 и t_3 имеют одинаковые абсолютные значения. Слева от линии симметрии коэффициенты полностью идентичны, а справа имеют противоположные знаки. Это справедливо также и для пар (t_1, t_4) и (t_2, t_5). Используя эту симметрию, можно снизить число сложений, как показано на рис. 4.3*б*.

Окончательные преобразования связаны с исключением e_i на рис. 4.3*б*. В приведенных ниже выкладках мы опираемся на определения (4.64), (4.67) и соответствующее использование (4.63), (4.66):

$$g_1-\delta_1=e_3-e_2=-2\alpha_7\beta_2-\alpha_2(\beta_7+\beta_2)-\alpha_3\beta_7=$$
$$=\beta_2\underbrace{(-\alpha_3-\alpha_7)}_{\delta_2}+\beta_7\underbrace{(-\alpha_2-\alpha_3)}_{\delta_7}, \tag{4.68}$$

$$g_6-\delta_6=e_4-e_5=\alpha_5(\beta_8-\beta_5)+2\alpha_8\beta_5-$$
$$-\alpha_4\beta_8=\beta_5\underbrace{(\alpha_8+\alpha_4)}_{\delta_5}+\beta_8\underbrace{(\alpha_5-\alpha_4)}_{\delta_8}, \tag{4.69}$$

$$g_2-\delta_1=2e_2+e_3=\alpha_7(\beta_3-\beta_7)-\alpha_2\beta_3+2\alpha_3\beta_7=$$
$$=\beta_3\underbrace{(\alpha_7-\alpha_2)}_{\delta_3}-\beta_7\underbrace{(-\alpha_2-\alpha_3)}_{\delta_7}, \tag{4.70}$$

$$g_5-\delta_6=2e_5+e_4=-\alpha_8(\beta_8-\beta_4)+\alpha_5\beta_4+2\alpha_4\beta_8=$$
$$=\beta_4\underbrace{(\alpha_8+\alpha_5)}_{\delta_4}-\beta_8\underbrace{(\alpha_5-\alpha_4)}_{\delta_8}; \tag{4.71}$$

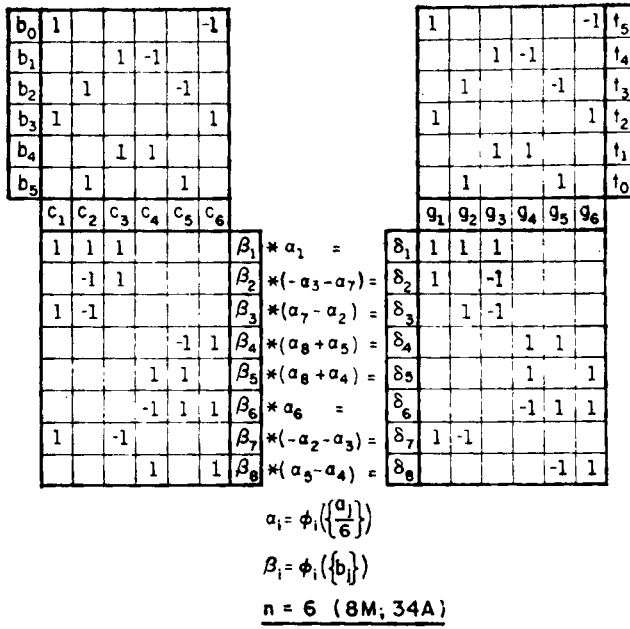


Рис. 4.4. Алгоритм ЛЦП порядка 6

$$g_3 - \delta_1 = -e_2 - 2e_3 = \alpha_7\beta_2 + 2\alpha_2\beta_3 - \alpha_3(\beta_3 - \beta_2) = \\ = -\beta_2(\underbrace{-\alpha_3 - \alpha_7}_{\delta_2}) - \beta_3\underbrace{(\alpha_7 - \alpha_2)}_{\delta_3}, \qquad (4.72)$$

$$g_4 + \delta_6 = e_5 + 2e_4 = \alpha_8\beta_5 + 2\alpha_5\beta_4 + \alpha_4(\beta_5 + \beta_4) = \\ = \beta_4\underbrace{(\alpha_8 + \alpha_5)}_{\delta_4} + \beta_5\underbrace{(\alpha_8 + \alpha_4)}_{\delta_5}. \qquad (4.73)$$

На этом вывод алгоритма завершается (рис. 4.4).

4.4. Базовые алгоритмы ДПФ для простых N

В этом разделе построенные выше схемы используются для получения алгоритмов ДПФ для простых нечетных элементов списка (4.3), а именно, 3, 5, 7. Как указывалось в разд. 4.1, они послужат конструктивными блоками для алгоритмов ДПФ более высоких порядков. В разд. 4.2 показано, что при соответствующей перестановке индексов матрица ДПФ порядка N отображает ЛЦ-подматрицу порядка n. Основная часть вклада этой подматрицы в выполнение преобразования в целом раскрыта в (4.21), которое повторим еще раз:

$$B'_\rho = \Omega \sum_{\rho=0}^{n-1} W^{g^\rho + \sigma} b_\sigma \quad (\rho = 0,\ 1,\ldots,\ n-1). \qquad (4.74)$$

С другой стороны, ЛЦТ-схемы в последнем разделе основываются на представлении ЛЦ-преобразования порядка n, определяемом (4.29), (4.39). Поэтому при применении ЛЦП-схем к ЛЦ-преобразованиям, выражаемым (4.74), нужно принять следующие определения:

$$B'_\rho = t_{n-1-\rho} \quad (\rho = 0,\ 1,\ \ldots,\ n-1), \tag{4.75}$$

$$a_\rho = \Omega W^{g^{n-1-\rho}} \quad (\rho = 0,\ 1,\ \ldots,\ n-1). \tag{4.76}$$

Обратим внимание на роль (4.76). ЛЦП-схемы как обобщенные задают лишь правила вычисления α_i по значениям a_i. Однако, поскольку (4.76) содержит явную формулу для a_i, можно на самом деле вычислить конкретные значения α_i:

$$\alpha_i = \Omega \varepsilon_i, \tag{4.77}$$

где ε_i — функции i и N, поэтому их можно вычислить заранее.

Перепишем оставшиеся выражения ДПФ с переставленными индексами (4.19), (4.25), (4.26):

$$B_\rho = \hat{B}_\rho + B'_\rho \quad (\rho = 0,\ 1,\ \ldots,\ n-1), \tag{4.78}$$

$$\hat{B}_\rho = \Omega b_0 \quad (\rho = 0,\ 1,\ \ldots,\ n-1), \tag{4.79}$$

$$B_{(0)} = \Omega\left(b_{(0)} + \sum_{\sigma=0}^{n-1} b_\sigma\right). \tag{4.80}$$

Заметим, что (4.79) и (4.80) справедливы только для случая простого N и основаны на равенстве

$$n = N - 1. \tag{4.81}$$

С другой стороны, равенства (4.74)—(4.78) являются совершенно общими и могут использоваться также в следующих двух разделах, где N — не простое.

Первый шаг в конструировании ДПФ-схемы для простых N заключается в вычислении констант $\varepsilon_i(N)$. Для этого с помощью (4.76) вычисляются a_ρ, которые применяются затем в ЛЦП-схемах порядка $(N-1)$ для вычисления α_i и, следовательно, ε_i (4.77).

Следующим шагом является преобразование ЛЦП-схем порядка $(N-1)$ в ДПФ-схемы порядка N. Это эквивалентно реализации (4.74) и дает схему преобразования b_i в t_j. Теперь с помощью (4.14)—(4.16), (4.75) заменим эти переменные переменными f_i и F_j. С этого момента индексация на входе и выходе будет, как правило, немонотонной. Поэтому, чтобы сохранить монотонность, переставим строки и столбцы во входных и выходных квадратах.

Следует отметить, что замена переменных (4.14)—(4.16) была введена для того, чтобы выявить ЛЦ-структуру и сделать, таким образом, возможным использование (4.45). Однако после этого нам хотелось бы иметь результирующую ДПФ-схему в таком виде, чтобы в ней фигурировали переменные F_u, f_v со стандартной монотонной последовательностью индексов, поскольку это облегчает объединение отдельных схем в алгоритм для больших N.

Обратимся теперь к реализации (4.80). Поскольку все три ЛЦП-схемы удовлетворяют соотношению

$$\beta_1 = \sum_{i=0}^{n-1} b_i, \tag{4.82}$$

равенство (4.80) эквивалентно

$$F_0 = B_{(0)} = \Omega(b_{(0)} + \beta_1) = \Omega(f_0 + \beta_1). \tag{4.83}$$

Наконец, чтобы эффективно реализовать (4.78), во всех трех ЛЦП-схемах выделяем член, который появляется с коэффициентом $+1$ у всех t_i. Оказывается, что этот член есть δ_1.

Следовательно, замена $\delta_1 = \alpha_1\beta_1$ на

$$\delta_1 = \Omega\, b_{(0)} + \alpha_1\beta_1 \tag{4.84}$$

преобразует на выходе B'_ρ в B_ρ. Заметим, однако, что слагаемое $\Omega b_{(0)}$ уже входит в $B_{(0)}$, вычисленное в (4.83). Это дает возможность исключить одно или два умножения в (4.84). Действительно, объединяем (4.83) с (4.84) и получаем [используя (4.77)]:

$$\delta_1 = B_{(0)} + (\varepsilon_1 - 1)\,\Omega\beta_1 = F_0 + (\varepsilon_1 - 1)\,\Omega\beta_1. \tag{4.85}$$

Отсюда видно, что множитель у β_1 является произведением Ω и некоторой функции от N. Это оказывается общим для всех β_i во всех ДПФ-схемах, которые предстоит построить. Поэтому введем обозначение

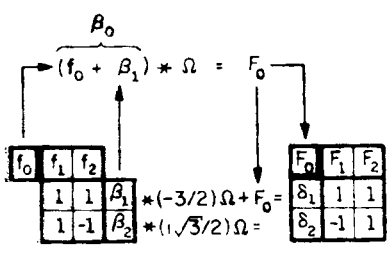
$$\xi_i = (\omega_i\,\Omega)\,\beta_i, \tag{4.86}$$

где ω_i — функция только i, N. β_i всегда первоначально преобразуется в соответствии с (4.86). Следовательно, одной из задач при конструировании ДПФ-схем будет определение значений констант ω_i. В ходе разработки будет видно, что лежащее в основе ЛЦП-схемы вместе с выражениями (4.77), (4.86) всегда однозначно определяют ω_i. В случае (4.85)

$$\omega_1 = \varepsilon_1 - 1, \quad \delta_1 = B_{(0)} + (\omega_1\,\Omega)\,\beta_1 = F_0 + (\omega_1\,\Omega)\,\beta_1. \tag{4.87}$$

Итак, мы рассмотрели с общих позиций связь ЛЦП с ДПФ. Обратимся теперь к частным случаям.

4.4.1. ДПФ порядка 3 (рис. 4.5)



Равенство (4.83) явно указано на схеме. Из (4.76), учитывая, что $W = \exp\left(-i\frac{2\pi}{3}\right)$; $g = 2$, получаем

$$\begin{bmatrix} a_0/2 \\ a_1/2 \end{bmatrix} = \Omega/2 \begin{bmatrix} W^2 \\ W^1 \end{bmatrix} + \Omega/2 \begin{bmatrix} \overline{W}^1 \\ W^1 \end{bmatrix}, \tag{4.88}$$

Рис. 4.5. Алгоритм ДПФ порядка 3

где $\overline{W}$ — комплексно сопряженное значение W. Следовательно, применяя ЛЦП-схему второго порядка (см. рис. 4.1), находим

$\frac{\Omega}{2}\overline{W}^1$	$\frac{\Omega}{2}W^1$	
1	1	α_1
1	−1	α_2

$$= -\frac{\Omega}{2}, \quad \varepsilon_1 = -\frac{1}{2}, \quad \omega_1 = \varepsilon_1 - 1 = -\frac{3}{2};$$

$$= i\frac{\sqrt{3}}{2}\Omega, \quad \varepsilon_2 = i\frac{\sqrt{3}}{2}, \quad \omega_2 = \varepsilon_2 = i\frac{\sqrt{3}}{2},$$

4.4.2. ДПФ порядка 5 (рис. 4.6)

$W = \exp\left(-i\frac{2\pi}{5}\right)$, $g = 2$. Следовательно, из (4.76)

$$\frac{1}{4}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \frac{\Omega}{4}\begin{bmatrix} W^3 \\ W^4 \\ W^2 \\ W^1 \end{bmatrix} = \frac{\Omega}{4}\begin{bmatrix} \overline{W}^2 \\ \overline{W}^1 \\ W^2 \\ W^1 \end{bmatrix} \tag{4.89}$$

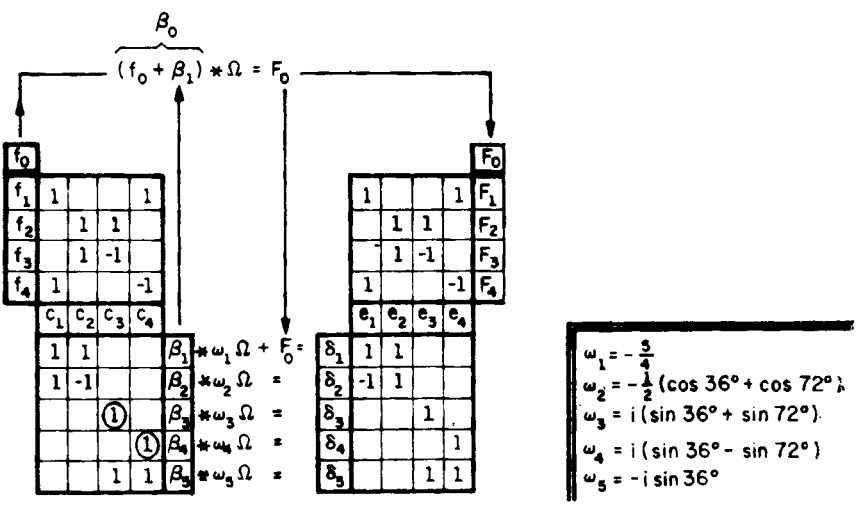
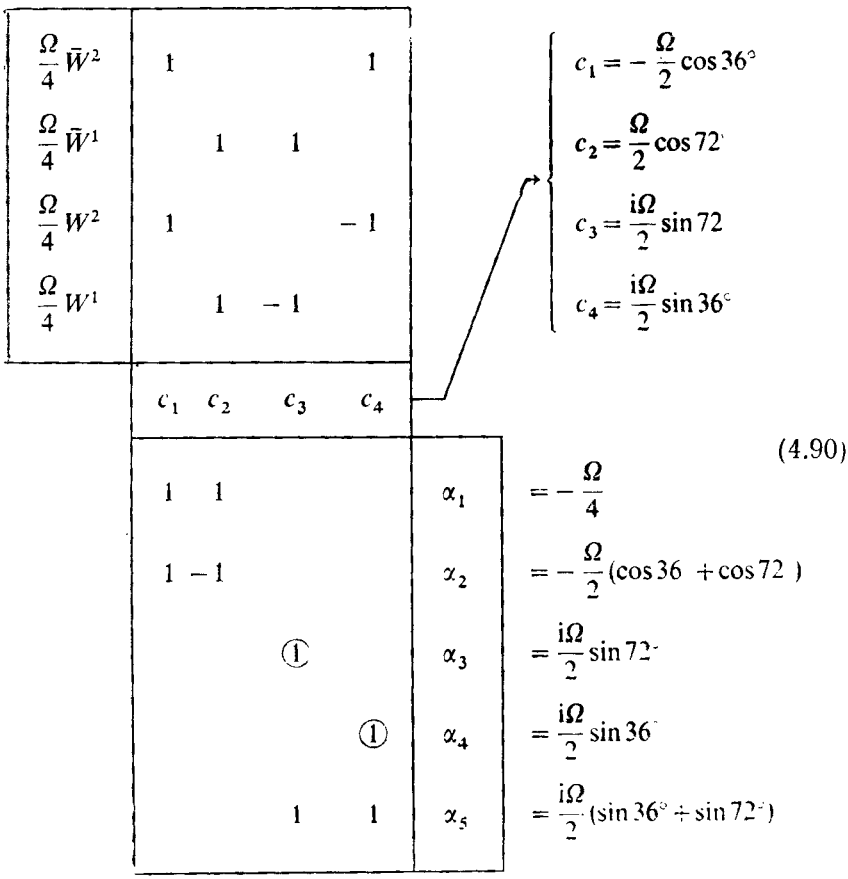


Рис. 4.6. Алгоритм ДПФ порядка 5

находятся по следующим правилам из рис. 4.2:



$$
\begin{array}{|cc|cccc|}
\hline
\frac{\Omega}{4}\bar{W}^2 & & 1 & & & 1 \\
\frac{\Omega}{4}\bar{W}^1 & & & 1 & 1 & \\
\frac{\Omega}{4}W^2 & & 1 & & & -1 \\
\frac{\Omega}{4}W^1 & & & 1 & -1 & \\
\hline
 & & c_1 & c_2 & c_3 & c_4 \\
\hline
1 & 1 & & & & \alpha_1 = -\frac{\Omega}{4} \\
1 & -1 & & & & \alpha_2 = -\frac{\Omega}{2}(\cos 36 + \cos 72) \\
 & & & \textcircled{1} & & \alpha_3 = \frac{i\Omega}{2}\sin 72 \\
 & & & & \textcircled{1} & \alpha_4 = \frac{i\Omega}{2}\sin 36 \\
 & & & 1 & 1 & \alpha_5 = \frac{i\Omega}{2}(\sin 36 + \sin 72) \\
\hline
\end{array}
$$

$$
\left\{
\begin{aligned}
c_1 &= -\frac{\Omega}{2}\cos 36 \\
c_2 &= \frac{\Omega}{2}\cos 72 \\
c_3 &= \frac{i\Omega}{2}\sin 72 \\
c_4 &= \frac{i\Omega}{2}\sin 36
\end{aligned}
\right.
\qquad (4.90)
$$

Равенство (4.77), (4.90) определяют ε_i. Наконец, из (4.87) с использованием коэффициентов при β_i на рис. 4.2 получаем

$$
\left.
\begin{aligned}
\omega_1 &= \varepsilon_1 - 1 = -5/4, \\
\omega_2 &= \varepsilon_2 = -\frac{1}{2}(\cos 36^\circ + \cos 72^\circ), \\
\omega_3 &= 2\varepsilon_5 = i(\sin 36^\circ + \sin 72^\circ), \\
\omega_4 &= 2(\varepsilon_4 - \varepsilon_3) = i(\sin 36^\circ - \sin 72^\circ), \\
\omega_5 &= -2\varepsilon_4 = -\sin 36^\circ.
\end{aligned}
\right\} \qquad (4.91)
$$

4.4.3. ДПФ порядка 7

$W = \exp\left(-\mathrm{i}\,\frac{2\pi}{7}\right)$, $g = 3$. Тогда из (4.76)

$$\frac{1}{6}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \frac{\Omega}{6}\begin{bmatrix} W^5 \\ W^4 \\ W^6 \\ W^2 \\ W^3 \\ W^1 \end{bmatrix} = \frac{\Omega}{6}\begin{bmatrix} \overline{W}^2 \\ \overline{W}^3 \\ \overline{W}^1 \\ \overline{W}^2 \\ \overline{W}^3 \\ \overline{W}^1 \end{bmatrix}. \tag{4.92}$$

Все α_i выражаются через угол

$$\theta = 90°/7 \tag{4.93}$$

и кратные ему углы. Правило рис. 4.4 для вычисления α_i показано на рис. 4.7, а окончательная схема, основанная на нем, — на рис. 4.8.

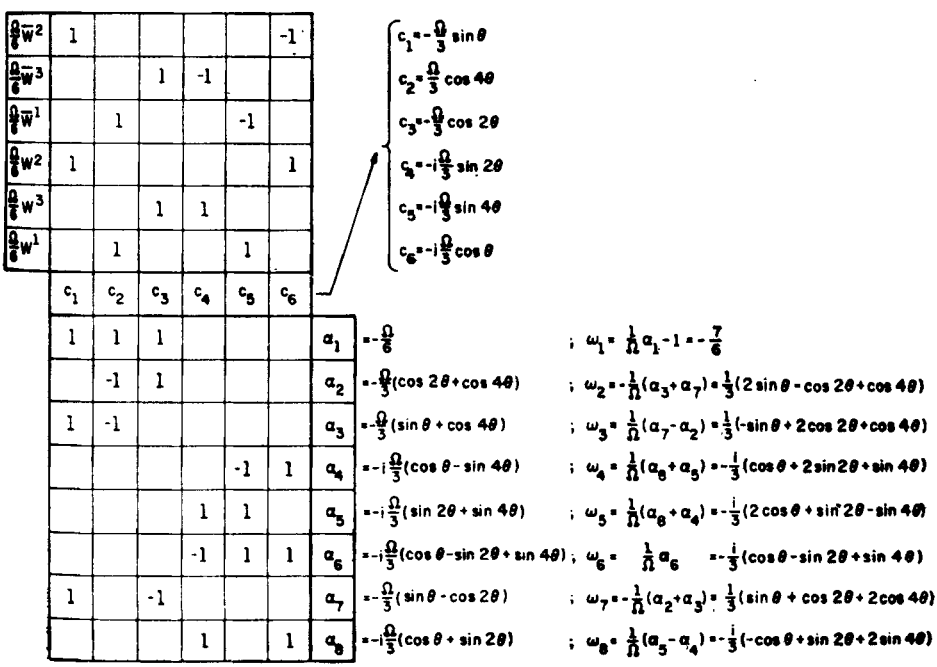


Рис. 4.7. Вычисление ω_i для алгоритма ДПФ порядка 7

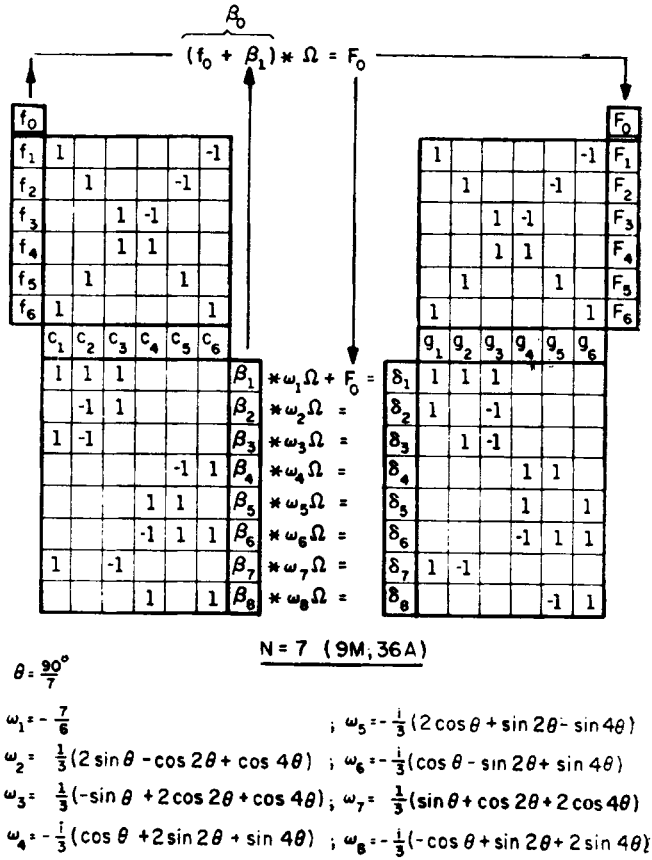


Рис. 4.8. Алгоритм ДПФ порядка 7

4.5. Базовые алгоритмы ДПФ для $N=4$, 9

Из (4.2) можно увидеть, что при использовании схем, полученных в последнем разделе, наивысший реализуемый порядок $N=105$ $(3\cdot5\cdot7)$. Учитывая это, добавим тривиальный алгоритм для $N=2$ (рис. 4.9), увеличив таким образом максимальное значение N до 210.

Если бы мы захотели получить еще бо́льшие N, нам нужно было бы или построить новые схемы для следующих по порядку простых чисел (11, 13, 17, ...), или придумать новые схемы для $N=p^k$ (p — простое). Выберем последнее.

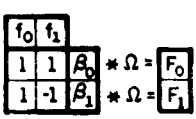


Рис. 4.9. Алгоритм ДПФ порядка 2

4.5.1. ДПФ порядка 4

$N=2^2$, т. е. в соответствии с (4.12)

$$n=2. \tag{4.94}$$

Примитивный корень из 4 равен 3. Отсюда

ρ	0	1
$r=3^{\rho} \bmod 4$	1	3

(4.95)

Число строк, исключаемых из ЛЦ-таблицы, равно 2. Оно значительно увеличивается в последующих таблицах, достигая 8 при $N=16$. Это обстоятельство требует новой и слегка модифицированной терминологии для облегчения работы с частью матрицы, не являющейся левым циркулянтом.

Во-первых, расширим определение r так, чтобы (4.15) включало теперь (4.16):

ρ	0	1	(0)	(2)
r	1	3	0	2

(4.96)

Подобным образом для индексов столбца σ, s теперь имеем

σ	0	1	(0)	(2)
s	1	3	0	2

(4.97)

В этом и в следующем разделах будем придерживаться такого расширенного определения r, s без явного упоминания об этом.

Дополним теперь (4.19) — (4.21):

$$F'_r = B'_{\rho}, \quad \hat{F}_r = \hat{B}_{\rho}, \quad F_r = \hat{F}_r + F'_r. \tag{4.98}$$

И, наконец, зная, что элемент (r, s) матрицы ДПФ (4.9) — это ΩW^{rs}, введем «экспоненциальную» матрицу E:

$$E_{rs} = (rs) \bmod N, \tag{4.99}$$

которая оказывается очень удобной для учета вклада части, не являющейся лево-циркулянтной.

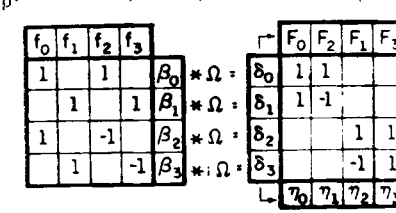


Рис. 4.10. Алгоритм ДПФ порядка 4

В данном случае

$$E=\begin{array}{c} \begin{matrix} s\to 0 & 2 & 1 & 3 \\ \sigma\to(0) & (2) & 0 & 1 \end{matrix} \\ \left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ \hline 0 & 2 & 1 & 3 \\ 0 & 2 & 3 & 1 \end{array}\right] \end{array} \begin{array}{cc} (0) & 0 \\ (2) & 2 \\ & \\ 0 & 1 \\ 1 & 3 \\ \uparrow & \uparrow \\ \rho & r \end{array} \qquad (4.100)$$

Здесь добавлены оба типа индексов строки и столбца. Проще всего получить E непосредственно из определения (4.99). Выписывая эту матрицу, нужно следить за тем, чтобы незаключенные в скобки индексы ρ, σ следовали в порядке 0, 1, 2, ... Этим матрице придается ЛЦ-структура. Последовательность других индексов безразлична.

Видно, что (4.100), как и следовало ожидать, представляет ЛЦ-подматрицу второго порядка. Используем действие этой подматрицы вместе со схемой ЛЦП на рис. 4.1, начиная с вычисления

$$\frac{1}{2}\begin{bmatrix} a_0 \\ a_1 \end{bmatrix}=\frac{\Omega}{2}\begin{bmatrix} W^3 \\ W^1 \end{bmatrix}=\frac{\Omega}{2}\begin{bmatrix} \mathrm{i} \\ -\mathrm{i} \end{bmatrix}. \qquad (4.101)$$

Следовательно (см. рис. 4.1),

$$\alpha_1=0,\ \alpha_2=\mathrm{i}\,\Omega. \qquad (4.102)$$

Отсюда следует, что только строка β_2 дает вклад в формирование F'_i. Перепишем ее в строку β_3 на рис. 4.10. Выходной вектор F (см. рис. 4.10) оказывается неупорядоченным по своим составляющим. В разд. 4.9 будут показаны преимущества некоторых схемных структур, которые обеспечивают экономию объема памяти при реализации алгоритма. Схемы, построенные до этого, имеют и такого рода структуру, и упорядоченную индексацию выходного вектора F. Начиная с этого момента, по-видимому, уже невозможно иметь оба желаемых свойства одновременно. Мы выбрали структуру, экономящую объем памяти, что представляется более важным, поэтому имеем неупорядоченный выходной вектор. В данном случае закон перемешивания индексов достаточно прост, но при $N=16$ он становится сложным. Для облегчения выкладок введем дополнительный вектор η в формируемые схемы. Неупорядоченный вектор F идентичен упорядоченному η (см. рис. 4.10).

Теперь вернемся к реализации оставшейся части матрицы E (4.100)

$$\left.\begin{aligned} F_r-F'_r&=\Omega\underbrace{(f_0-f_2)}_{\beta_2}=\underbrace{\Omega\beta_2}_{\delta_2}, \\ F_r&=F'_r+\delta_2 \end{aligned}\right\}(r=1,\ 3). \qquad (4.103)$$

Здесь использован тот факт, что $N=4$, $W^2=-1$. Равенства, подобные (4.100), могут быть выведены непосредственно из матрицы E.

Такие равенства будут даваться в дальнейшем без каких-либо комментариев:

$$F_2 = \Omega[\underbrace{(f_0+f_2)}_{\beta_0} - \underbrace{(f_1+f_3)}_{\beta_1}] = \underbrace{\Omega\beta_0}_{\delta_0} - \underbrace{\Omega\beta_1}_{\delta_1} = \delta_0 - \delta_1,$$

$$F_0 = \Omega[\underbrace{(f_0+f_2)}_{\beta_0} + \underbrace{(f_1+f_3)}_{\beta_1}] = \underbrace{\Omega\beta_0}_{\delta_0} + \underbrace{\Omega\beta_1}_{\delta_1} = \delta_0 + \delta_1.$$

Этим завершается вывод.

4.5.2. ДПФ порядка 9

$N=3^2$. Следовательно, из (4.12) имеем

$$n=6. \tag{4.104}$$

Примитивным корнем из 9 является примитивный корень из 3, а именно 2. Это приводит к

ρ	0	1	2	3	4	5
$r=2^\rho \bmod 9$	1	2	4	8	7	5

. (4.105)

Отсюда матрица E имеет вид

$$E = \begin{array}{c} s\to \\ \sigma\to \\ \end{array} \begin{array}{cccccccccc} 0 & 3 & 6 & 1 & 2 & 4 & 8 & 7 & 5 & \\ (0) & (3) & (6) & 0 & 1 & 2 & 3 & 4 & 5 & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0)\ 0 \\ 0 & 0 & 0 & 3 & 6 & 3 & 6 & 3 & 6 & (3)\ 3 \\ 0 & 0 & 0 & 6 & 3 & 6 & 3 & 6 & 3 & (6)\ 6 \\ \hline 0 & 3 & 6 & 1 & 2 & 4 & 8 & 7 & 5 & 0\ 1 \\ 0 & 6 & 3 & 2 & 4 & 8 & 7 & 5 & 1 & 1\ 2 \\ 0 & 3 & 6 & 4 & 8 & 7 & 5 & 1 & 2 & 2\ 4 \\ 0 & 6 & 3 & 8 & 7 & 5 & 1 & 2 & 4 & 3\ 8 \\ 0 & 3 & 6 & 7 & 5 & 1 & 2 & 4 & 8 & 4\ 7 \\ 0 & 6 & 3 & 5 & 1 & 2 & 4 & 8 & 7 & 5\ 5 \\ & & & & & & & & & \uparrow\ \uparrow \\ & & & & & & & & & \rho\ r \end{array} \tag{4.106}$$

Поскольку $n=6$, здесь применима схема рис. 4.4. Рассмотрим сначала α_i. Матрица E (4.106) отображает последовательность a_i [это согласуется также с (4.76)]. Тогда

$$\frac{1}{6}\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \frac{\Omega}{6}\begin{bmatrix} W^5 \\ W^7 \\ W^8 \\ W^4 \\ W^2 \\ W^1 \end{bmatrix} = \frac{\Omega}{6}\begin{bmatrix} \overline{W}^4 \\ \overline{W}^2 \\ \overline{W}^1 \\ W^4 \\ W^2 \\ W^1 \end{bmatrix}. \tag{4.107}$$

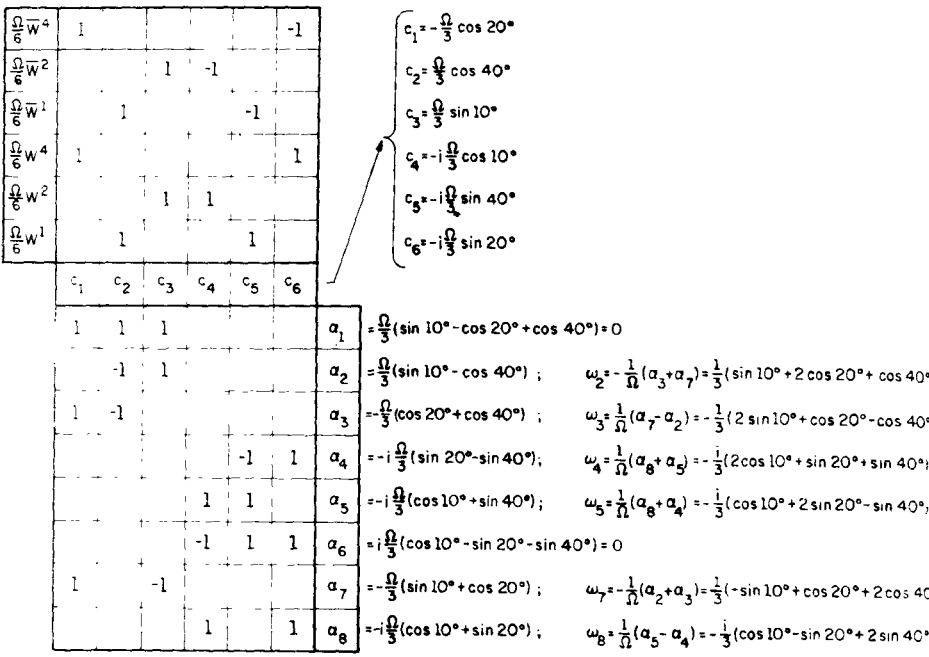


Рис. 4.11. Вычисление ω_i для алгоритма ДПФ порядка 9

Вычисление α_i и ω_i показано на рис. 4.11. Заметим, что в терминологии рис. 4.4, 4.11

$$\alpha_1 = \alpha_6 = 0. \tag{4.108}$$

Это означает, что в реализации F'_i, которая выполняется копированием рис. 4.4, 4.11 в рис. 4.13, члены, связанные с α_1 и α_6, должны быть отброшены[1]. При копировании этих рисунков используем некоторые перестановки, кроме очевидных перестановок входных и выходных переменных. Эти перестановки расшифрованы на рис. 4.12. Заметим, что перестановка $g_i \to g_j$ показана за два последовательных шага: $g_i \to e_j$, $e_j \to g_j$. (Например, $g_1 \to e_9 \to g_3$.)[2]

Обратимся к учету оставшейся части матрицы, используя в качестве руководства (4.106). При этом нам встретятся следующие константы:

$$\Omega W^3 = \Omega(-\sin 30° - i\cos 30°) = -\Omega\left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right),$$

$$\Omega W^6 = \Omega \overline{W}^3 = -\Omega\left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right),$$

[1] Рис. 4.13 может создать впечатление, что эти члены не были отброшены. Это заблуждение, поскольку члены, создающие такое впечатление, добавлены позже.

[2] В случае $g_3 \to e_2 \to g_2$ отброшены два изменения знака.

Индексы (i) рис. 4.4; 4.11	0	1	2	3	4	5	6	7	8
$b_i \to f_j$	1	2	4	8	7	5			
$c_i \to c_j$		1	4	2	7	5	8		
$\omega_i \to \omega_j;\ \beta_i \to \beta_j;\ \delta_i \to \delta_j$			4	2	7	5		9	10
$g_i \to e_j$		9	4	2	7	5	10		
$e_j \to g_j$		3	4	2	7	5	6		
$t_i \to F_j$	5	7	8	4	2	1			

Индексы (j) к рис. 4.13

Рис. 4.12. Перестановка индексов для формирования рис. 4.13

которые будут применяться в дальнейшем.

$r = \mathbf{1;\ 4;\ 7}$

$$F_r - F'_r = \Omega\left[f_0 - \left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right) f_3 - \left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right) f_6\right] =$$

$$= \Omega\left[\underbrace{f_0}_{\beta_0} - \frac{1}{2}\underbrace{(f_6 + f_3)}_{\beta_3} + i\frac{\sqrt{3}}{2}\underbrace{(f_6 - f_3)}_{\beta_6}\right] =$$

$$= \underbrace{\Omega\beta_0}_{\delta_0} - \underbrace{\frac{1}{2}\Omega\beta_3}_{\delta_3} - \underbrace{\left(-i\frac{\sqrt{3}}{2}\Omega\beta_6\right)}_{\delta_6} = \underbrace{\delta_0 - \delta_3}_{e_3} - \underbrace{\delta_6}_{e_6}.$$

$$\therefore F_r = F'_r + e_3 - e_6.$$

Воздержимся пока от дальнейшей разработки этого случая и перейдем к следующей группе.

$r = \mathbf{2;\ 8;\ 5}$.

Этот случай по сравнению с предыдущим требует замены W^3 на W^6 $(= \overline{W}^3)$. Отсюда $F_r = F'_r + e_3 + e_6$. С этих позиций обе группы можно реализовать так, как показано на рис. 4.13.

Теперь реализуем F_3, F_6. Из (4.106)

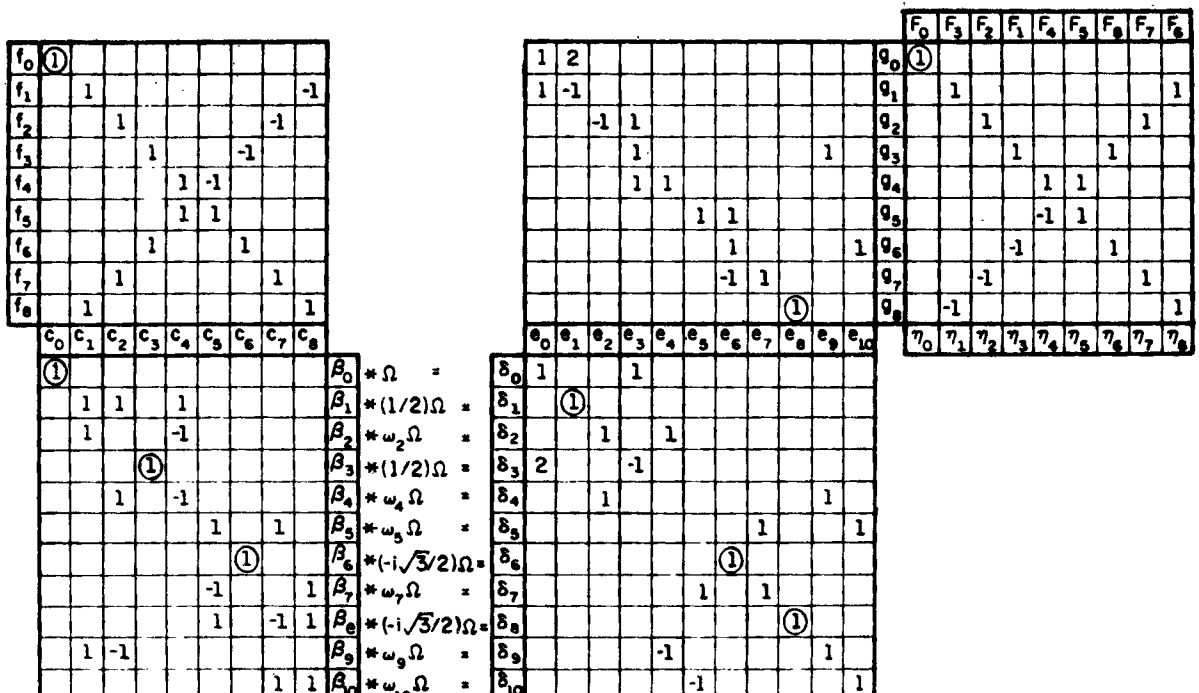
$$F_3 = \Omega\left[(f_0 + f_3 + f_6) - \left(\frac{1}{2} + i\frac{\sqrt{3}}{2}\right)(f_1 + f_4 + f_7) - \right.$$

$$\left. - \left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right)(f_2 + f_8 + f_5)\right] =$$

$$= \Omega\left\{\underbrace{f_0}_{\beta_0} + \underbrace{(f_6 + f_3)}_{\beta_3} - \frac{1}{2}\left[\underbrace{(f_8 + f_1)}_{c_1} + \underbrace{(f_7 + f_2)}_{c_2} + \underbrace{(f_5 + f_4)}_{c_4}\right] + \right.$$

$$\left. + i\frac{\sqrt{3}}{2}\left[\underbrace{(f_8 - f_1)}_{c_8} - \underbrace{(f_7 - f_2)}_{c_7} + \underbrace{(f_5 - f_4)}_{c_5}\right]\right\} =$$

$$= \underbrace{\Omega\beta_0}_{\delta_0} - \underbrace{\Omega\beta_3}_{\delta_3} - \frac{1}{2}\Omega\underbrace{(c_1 + c_2 + c_4)}_{\beta_1} + i\frac{\sqrt{3}}{2}\Omega\underbrace{(c_5 - c_7 + c_8)}_{\beta_8} =$$



N = 9 (11M; 44A)

Рис. 4.13. Алгоритм ДПФ порядка 9

$$= \underbrace{(\delta_0 + 2\delta_3)}_{e_0} - \underbrace{\left(\frac{1}{2}\Omega\beta_1\right)}_{\delta_1} - \underbrace{\left(-i\frac{\sqrt{3}}{2}\Omega\beta_8\right)}_{\delta_8} =$$

$$= e_0 - \underbrace{\delta_1}_{e_1} - \underbrace{\delta_8}_{e_8} = \underbrace{e_0 - e_1}_{g_1} - \underbrace{e_8}_{g_8} = g_1 - g_8.$$

Обратимся теперь к F_6. Единственное отличие здесь состоит в замене W^3 на W^6 $(= \overline{W}^3)$. Отсюда $F_6 = g_1 + g_8$.

Наконец,

$$F_0 = \Omega(f_0 + f_3 + f_6) + \Omega\beta_1 = \underbrace{\Omega\beta_0}_{\delta_0} + \underbrace{\Omega\beta_3}_{2\delta_3} + \underbrace{\Omega\beta_1}_{2\delta_1} =$$

$$= \underbrace{(\delta_0 + 2\delta_3)}_{e_0} + \underbrace{2\delta_1}_{e_1} = e_0 + 2e_1 = g_0.$$

Этим завершается вывод.

С двумя схемами, разработанными в этом разделе, максимальное реализуемое значение N достигает $1260 = (4 \cdot 5 \cdot 7 \cdot 9)$. Мы увеличим его до 5040 с помощью схем, рассматриваемых в следующем разделе.

4.6. Базовые алгоритмы ДПФ для $N = 8, 16$

Разрабатывая схемы для $N = 8, 16$, мы сталкиваемся с затруднением, обусловленным тем, что

$$N = 2^k \quad (k > 2) \tag{4.109}$$

не имеет примитивных корней. Другими словами, не существует целое число, степени которого по модулю N порождали бы все нечетные числа из интервала $(1, N)$. Однако одну половину этих чисел можно сгенерировать, используя степени числа 3 по модулю N, а другую половину — используя отрицательные значения этих степеней по модулю N [4.9]. Таким образом, возникает необходимость разбиения $\{F_r\}$ на 3 подмножества:

$$F_r = \begin{cases} B_{(r)} & (r \text{ — четное}), \\ B_\rho & (r = 3^\rho \bmod N), \\ B_{\overline{\rho}} & (r = -3^\rho \bmod N). \end{cases} \tag{4.110}$$

Как и индексы, заключенные в скобки (i) (см. разд. 4.2), индекс i служит для идентификации подмножества, к которому принадлежит $B_{\overline{i}}$. Заметим, что нулевой индекс появляется во всех трех подмножествах:

$$B_{(0)} = F_0, \quad B_0 = F_1, \quad B_{\overline{0}} = F_{N-1}. \tag{4.111}$$

Теперь можно заменить (4.14) на

$$\left.\begin{aligned} r &= 3^\rho \bmod N \\ s &= 3^\sigma \bmod N \end{aligned}\right\} (\rho, \sigma = 0, 1, \ldots, N/4 - 1), \tag{4.112}$$

$$\left.\begin{aligned} r &= -3^{\rho} \bmod N \\ s &= -3^{\sigma} \bmod N \end{aligned}\right\} (\rho,\ \sigma = \bar{0},\ \bar{1},\ \ldots,\ \overline{N/4-1}). \tag{4.113}$$

Проиллюстрируем это для $N=8$

ρ	$\bar{0}$	$\bar{1}$	0	1
$r = 3^{\rho} \bmod 8$	—	—	1	3
$r = -3^{\rho} \bmod 8$	7	5	—	—

(4.114)

Равенства (4.15), (4.16) должны теперь читаться как

$$\left.\begin{aligned} F_r &= B_{\rho}, \quad f_s = b_{\sigma} \quad (r,\ s \text{ — нечетные}), \\ F_r &= B_{(r)}, \quad f_s = b_{(s)} \quad (r,\ s \text{ — четные}), \end{aligned}\right\} \tag{4.115}$$

так что общее функциональное соотношение между r и ρ для $N=8$ может быть подытожено следующим образом:

ρ	$\bar{0}$	$\bar{1}$	0	1	(0)	(2)	(4)	(6)
r	7	5	1	3	0	2	4	6

(4.116)

Аналогичная таблица устанавливает соотношение между s и σ.

Равенства (4.112)—(4.115) применяются теперь, чтобы исключить F_u, f_v из (4.9) (см. анализ аналогичной ситуации в разд. 4.2, учитывая, что здесь $p=2$):

$$B_{(2t)} = F_{2t} = \Omega \sum_{m=0}^{N/2-1} W^{4mt} b_{(2m)} +$$

$$+ \Omega \sum_{\sigma=0}^{N/4-1} (W^{-2t3^{\sigma}} b_{\bar{\sigma}} + W^{2t3^{\sigma}} b_{\sigma}) \quad (t = 0, 1, \ldots, N/2-1), \tag{4.117}$$

$$B_{\rho} = \hat{B}_{\rho} + B'_{\rho},\ \hat{F}_r = \hat{B}_{\rho},\ F'_r = B'_{\rho} \tag{4.118}$$

(r — нечетное),

$$\left.\begin{aligned} \hat{B}_{\bar{\rho}} &= \Omega \sum_{m=0}^{N/2-1} W^{-2m3^{\rho}} b_{(2m)}, \\ \hat{B}_{\rho} &= \Omega \sum_{m=0}^{N/2-1} W^{2m3^{\rho}} b_{(2m)} \end{aligned}\right\} (\rho = 0, 1, \ldots, N/4-1), \tag{4.119}$$

$$\left.\begin{aligned} B'_{\bar{\rho}} &= \Omega \sum_{\sigma=0}^{N/4-1} (W^{3^{\rho+\sigma}} b_{\bar{\sigma}} + W^{-3^{\rho+\sigma}} b_{\sigma}), \\ B'_{\rho} &= \Omega \sum_{\sigma=0}^{N/4-1} (W^{-3^{\rho+\sigma}} b_{\bar{\sigma}} + W^{3^{\rho+\sigma}} b_{\sigma}) \end{aligned}\right\} (\rho = 0, 1, \ldots$$

$$\ldots, N/4-1). \tag{4.120}$$

Очевидно, что все четыре матричные произведения, появляющиеся в (4.120), являются преобразованиями Ханкеля $(\rho+\sigma)$ порядка $N/4$. Докажем теперь, что они являются также ЛЦ-преобразованиями. Для этого надо показать, что [см. (4.6), (4.22)]

$$3^{\rho+N/4-1} = 3^{\rho-1} \bmod N \,(N=2^k, k>2) \tag{4.121}$$

или, что эквивалентно,

$$3^{N/4} = 1 \bmod N \,(N=2^k, k>2). \tag{4.122}$$

Докажем (4.122) индукцией по k. Предположим, что оно истинно для $N=n$. Тогда $3^{n/4} \equiv mn+1$ (m — целое). Возведение в квадрат дает $3^{n/2} = (m^2 n/2)(2n) + m(2n) + 1$, так что $3^{(2n)/4} = 1 \bmod (2n)$ и (4.122) истинно для $N=2n$. Наконец, очевидно, что (4.122) истинно для $k=3$.

Таким образом, (4.120) включает в себя четыре произведения ЛЦ-матриц. Кроме того, эти четыре матрицы содержат в себе составную матрицу второго порядка, которая сама является ЛЦ-матрицей. Все эти свойства выявляются совершенно отчетливо в двух случаях, разбираемых ниже.

4.6.1. ДПФ порядка 8 (рис. 4.14)

Применяя перестановку (4.116), получим матрицу:

$$E = \left[\begin{array}{cccc|cccc|cc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (0) & 0 \\
0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 & (4) & 4 \\
0 & 0 & 4 & 4 & 6 & 2 & 2 & 6 & (2) & 2 \\
0 & 0 & 4 & 4 & 2 & 6 & 6 & 2 & (6) & 6 \\
\hline
0 & 4 & 6 & 2 & 1 & 3 & 7 & 5 & \bar{0} & 7 \\
0 & 4 & 2 & 6 & 3 & 1 & 5 & 7 & \bar{1} & 5 \\
0 & 4 & 2 & 6 & 7 & 5 & 1 & 3 & 0 & 1 \\
0 & 4 & 6 & 2 & 5 & 7 & 3 & 1 & 1 & 3
\end{array}\right] \tag{4.123}$$

$s \to$ 0 4 2 6 7 5 1 3

$\sigma \to$ (0) (4) (2) (6) $\bar{0}$ $\bar{1}$ 0 1

ρ r

Обратимся теперь к явному представлению подматрицы, соответствующей нижнему правому квадрату E. Обозначая

$$\omega_k = \Omega W^k, \tag{4.124}$$

получаем:

$$\begin{array}{c} \hat{t}_1 \\ \hat{t}_0 \end{array}\left\{\left[\begin{array}{c} F_7' \\ F_5' \\ \hline F_1' \\ F_3' \end{array}\right]\right. \overbrace{\left[\begin{array}{cc|cc} \omega_1 & \omega_3 & \omega_7 & \omega_5 \\ \omega_3 & \omega_1 & \omega_5 & \omega_7 \\ \hline \omega_7 & \omega_5 & \omega_1 & \omega_3 \\ \omega_5 & \omega_7 & \omega_3 & \omega_1 \end{array}\right]}^{\hat{a}_1 \qquad \hat{a}_0} \left[\begin{array}{c} f_7 \\ f_5 \\ \hline f_1 \\ f_3 \end{array}\right] \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\\a\\a\end{array}}\right\}\hat{b}_0 \\ \left.\vphantom{\begin{array}{c}a\\a\\a\\a\end{array}}\right\}\hat{b}_1 \end{array} \tag{4.125}$$

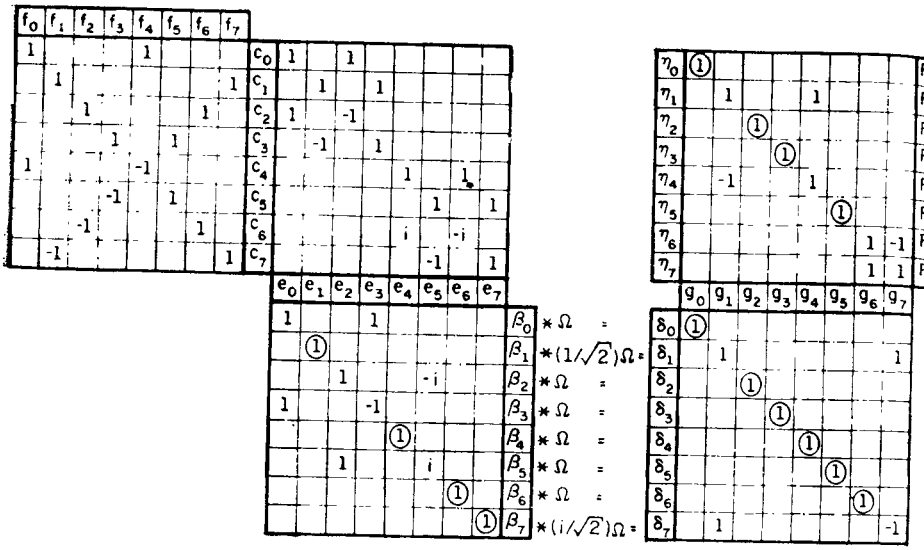


Рис. 4.14. Алгоритм ДПФ порядка 8

Мы видим здесь четыре ЛЦ-подматрицы второго порядка, по две матрицы каждого типа: $\hat{a}_0$ и $\hat{a}_1$. Используя введенную терминологию, выпишем эквивалентную блочную матрицу второго порядка

$$\begin{bmatrix} \hat{t}_1 \\ \hat{t}_0 \end{bmatrix} = \begin{bmatrix} \hat{a}_1 & \hat{a}_0 \\ \hat{a}_0 & \hat{a}_1 \end{bmatrix} \begin{bmatrix} \hat{b}_0 \\ \hat{b}_1 \end{bmatrix}. \tag{4.126}$$

Заметим, что (4.126) тоже является ЛЦ-матрицей. Оценим (4.126) непосредственно с помощью схемы на рис. 4.1. Здесь приходится прибегнуть к некоторым уточнениям. Все схемы в этой главе разрабатывались с неявным предположением о скалярном характере всех появляющихся в них переменных. На самом деле это не является необходимым. Таким образом, можно сделать вывод, что все схемы справедливы и при такой обобщающей интерпретации: 1) строки (столбцы) переменных (f_i, c_i, β_i, δ_i, e_i, F_i, ...) являются подматрицами-столбцами; 2) α_i, $a_i\Omega$ являются квадратными матрицами; 3) ω_i остается скалярной константой; 4) входы в схемы $\beta_i * \alpha_i = \delta_i$, $\beta_i * \omega_i\Omega = \delta_i$ нужно интерпретировать как матричные произведения[1] $\alpha_i\beta_i = \delta_i$, $\omega_i\Omega\beta_i = \delta_i$.

Непосредственной целью введения такого обобщения является вычисление (4.126). Однако важность его выходит за рамки непосредственного применения. Это обобщение означает, что наше

[1] «Модифицированной интерпретации» можно избежать, если использовать матрицы-строки вместо матриц-столбцов. Заметим в связи с этим, что мы будем рассматривать матрицу Ω как симметричную.

основное выражение для ДПФ, а именно (4.9), можно теперь интерпретировать как содержащее блочную матрицу ДПФ, т. е. такую матрицу, у которой блок (u, v) является произвольной постоянной матрицей Ω, умноженной на скаляр W^{uv}. К такой блочной матрице ДПФ применимы все уже разработанные схемы, а также две последние, разрабатываемые в этом разделе. Именно это обобщение переносит схемы из сферы теоретических рассуждений по поводу быстрых алгоритмов ДПФ очень низких порядков в область быстродействующих алгоритмов ДПФ высоких порядков. Конкретный способ применения этого обобщения приведен в следующем разделе.

Вернемся к вычислению (4.126). Из рис. 4.1 получаем

$$\left|\begin{array}{cc|c} \hat{b}_0 & \hat{b}_1 & \\ \hline 1 & 1 & \hat{\beta}_1 \\ 1 & -1 & \hat{\beta}_2 \end{array}\right| * \hat{\alpha}_1 = \left|\begin{array}{cc|c} \hat{t}_1 & \hat{t}_0 & \\ \hline 1 & 1 & \hat{\delta}_1 \\ -1 & 1 & \hat{\delta}_2 \end{array}\right|, \quad (4.127)$$

$$\left|\begin{array}{cc|c} \frac{1}{2}\hat{a}_0 & \frac{1}{2}\hat{a}_1 & \\ \hline 1 & 1 & \hat{\alpha}_1 \\ 1 & -1 & \hat{\alpha}_2 \end{array}\right|. \quad (4.128)$$

Отсюда (обозначая $\gamma = 1/\sqrt{2}$)

$$\hat{\alpha}_1 = \frac{\Omega}{2}\begin{bmatrix} W^7+W^1 & W^5+W^3 \\ W^5+W^3 & W^7+W^1 \end{bmatrix} = \gamma\Omega\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \hat{\beta}_1 = \begin{bmatrix} f_7+f_1 \\ f_5+f_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_3 \end{bmatrix},$$

$$\hat{\alpha}_2 = \frac{\Omega}{2}\begin{bmatrix} W^7-W^1 & W^5-W^3 \\ W^5-W^3 & W^7-W^1 \end{bmatrix} = \mathrm{i}\gamma\Omega\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \hat{\beta}_2 = \begin{bmatrix} f_7-f_1 \\ f_5-f_3 \end{bmatrix} = \begin{bmatrix} c_7 \\ c_5 \end{bmatrix},$$

$$\hat{\delta}_1 = \hat{\alpha}_1\hat{\beta}_1 = \gamma\Omega\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} c_1 \\ c_3 \end{bmatrix} = \gamma\Omega\begin{bmatrix} c_1-c_3 \\ c_3-c_1 \end{bmatrix} = \gamma\Omega\begin{bmatrix} \beta_1 \\ -\beta_1 \end{bmatrix} = \begin{bmatrix} \delta_1 \\ -\delta_1 \end{bmatrix},$$

$$\hat{\delta}_2 = \hat{\alpha}_2\hat{\beta}_2 = \mathrm{i}\gamma\Omega\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} c_7 \\ c_5 \end{bmatrix} = \mathrm{i}\gamma\Omega\begin{bmatrix} c_5+c_7 \\ c_5+c_7 \end{bmatrix} = \mathrm{i}\gamma\Omega\begin{bmatrix} \beta_7 \\ \beta_7 \end{bmatrix} = \begin{bmatrix} \delta_7 \\ \delta_7 \end{bmatrix},$$

$$\begin{bmatrix} F'_7 \\ F' \end{bmatrix} = \hat{t}_1 = \hat{\delta}_1 - \hat{\delta}_2 = \begin{bmatrix} \delta_1-\delta_7 \\ -(\delta_1+\delta_7) \end{bmatrix} =$$

$$= \begin{bmatrix} g_7 \\ -g_1 \end{bmatrix}, \quad \begin{bmatrix} F'_1 \\ F'_3 \end{bmatrix} = \hat{t}_0 = \hat{\delta}_1 + \hat{\delta}_2 = \begin{bmatrix} \delta_1+\delta_7 \\ -(\delta_1-\delta_7) \end{bmatrix} = \begin{bmatrix} g_1 \\ -g_7 \end{bmatrix}.$$

Обратимся теперь к оставшимся частям (4.123).

$r = 1; 5$

$$F_r = F'_r + \Omega[\underbrace{(f_0-f_4)}_{c_4} + \mathrm{i}\underbrace{(f_6-f_2)}_{c_6}] = F'_r + \underbrace{\Omega(\underbrace{c_4-\mathrm{i}c_6}_{\beta_4})}_{g_4} = F'_r + g_4.$$

$r = 3; 7$

$$F_r = F'_r + \underbrace{\Omega(\underbrace{c_4+\mathrm{i}c_6}_{\beta_6})}_{g_6} = F'_r + g_6,$$

$$F_6 = \Omega\,[\underbrace{(f_0+f_4)}_{c_0} - \underbrace{(f_2+f_6)}_{c_2} - \mathrm{i}\,\underbrace{(f_7-f_1)}_{c_7} + \mathrm{i}\,\underbrace{(f_5-f_3)}_{c_5}] =$$

$$= \Omega\,[\underbrace{(c_0-c_2)}_{e_2} + \mathrm{i}\,\underbrace{(c_5-c_7)}_{e_5}],$$

$$F_7 = \Omega\,\underbrace{(e_2+\mathrm{i}\,e_5)}_{\beta_5} = \Omega\beta_5 = g_5,\quad F_2 = \Omega\,\underbrace{(e_2-\mathrm{i}\,e_5)}_{\beta_2} = \Omega\beta_2 = g_2,$$

$$F_4 = \Omega\,[\underbrace{(f_0+f_4)}_{c_0} + \underbrace{(f_6+f_2)}_{c_2} - \underbrace{(f_7+f_1)}_{c_1} - \underbrace{(f_5+f_3)}_{c_3}] =$$

$$= \Omega\,[\underbrace{(c_0+c_2)}_{e_0} - \underbrace{(c_1+c_3)}_{e_3}] = \Omega\,\underbrace{(e_0-e_3)}_{\beta_3} = \Omega\beta_3 = g_3,$$

$$F_0 = \Omega\,\underbrace{(e_0+e_3)}_{\beta_0} = \Omega\beta_0 = g_0.$$

Этим завершается вывод.

4.6.2. ДПФ порядка 16

Перестановка определяется таблицей

ρ	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	0	1	2	3
$r = 3^\rho \bmod 16$	—	—	—	—	1	3	9	11
$r = -3^\rho \bmod 16$	15	13	7	5	—	—	—	—

. (4.129)

Отсюда получаем следующую матрицу E:

	$s\to$ 0	4	8	12	2	6	10	14	15	13	7	5	1	3	9	11		
	$\sigma\to$ (0)	(4)	(8)	(12)	(2)	(6)	(10)	(14)	$\bar{0}$	$\bar{1}$	$\bar{2}$	$\bar{3}$	0	1	2	3		
$E=$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(0)	0
	0	0	0	0	8	8	8	8	12	4	12	4	4	12	4	12	(4)	4
	0	0	0	0	0	0	0	0	8	8	8	8	8	8	8	8	(8)	8
	0	0	0	0	8	8	8	8	4	12	4	12	12	4	12	4	(12)	12
	0	8	0	8	4	12	4	12	14	10	14	10	2	6	2	6	(2)	2
	0	8	0	8	12	4	12	4	10	14	10	14	6	2	6	2	(6)	6
	0	8	0	8	4	12	4	12	6	2	6	2	10	14	10	14	(10)	10
	0	8	0	8	12	4	12	4	2	6	2	6	14	10	14	10	(14)	14
	0	12	8	4	14	10	6	2	1	3	9	11	15	13	7	5	$\bar{0}$	15
	0	4	8	12	10	14	2	6	3	9	11	1	13	7	5	15	$\bar{1}$	13
	0	12	8	4	14	10	6	2	9	11	1	3	7	5	15	13	$\bar{2}$	7
	0	4	8	12	10	14	2	6	11	1	3	9	5	15	13	7	$\bar{3}$	5
	0	4	8	12	2	6	10	14	15	13	7	5	1	3	9	11	0	1
	0	12	8	4	6	2	14	10	13	7	5	15	3	9	11	1	1	3
	0	4	8	12	2	6	10	14	7	5	15	13	9	11	1	3	2	9
	0	12	8	4	6	2	14	10	5	15	13	7	11	1	3	9	3	11
																	↑ ϱ	↑ r

(4.130)

Учитывая сложность случая, будем строить схему в два приема. Составляющая ЛЦ-подматриц F'_i рассматривается сначала отдельно в промежуточной схеме (рис. 4.15), а затем будет перенесена в окончательную схему (рис. 4.17).

Как и в предыдущем случае, выпишем ЛЦ-часть явно

$$\begin{array}{c} \hat{t}_1 \\ \hat{t}_0 \end{array} \left\{ \begin{array}{|c|} \hline F'_{15} \\ F'_{13} \\ F'_{7} \\ F'_{5} \\ \hline F'_{1} \\ F'_{3} \\ F'_{9} \\ F'_{11} \\ \hline \end{array} \right. = \left[\begin{array}{cccc|cccc} \multicolumn{4}{c|}{\hat{a}_1} & \multicolumn{4}{c}{\hat{a}_0} \\ \hline w_1 & w_3 & w_9 & w_{11} & w_{15} & w_{13} & w_7 & w_5 \\ w_3 & w_9 & w_{11} & w_1 & w_{13} & w_7 & w_5 & w_{15} \\ w_9 & w_{11} & w_1 & w_3 & w_7 & w_5 & w_{15} & w_{13} \\ w_{11} & w_1 & w_3 & w_9 & w_5 & w_{15} & w_{13} & w_7 \\ \hline w_{15} & w_{13} & w_7 & w_5 & w_1 & w_3 & w_9 & w_{11} \\ w_{13} & w_7 & w_5 & w_{15} & w_3 & w_9 & w_{11} & w_1 \\ w_7 & w_5 & w_{15} & w_{13} & w_9 & w_{11} & w_1 & w_3 \\ w_5 & w_{15} & w_{13} & w_7 & w_{11} & w_1 & w_3 & w_9 \end{array} \right] \left\{ \begin{array}{|c|} \hline f_{15} \\ f_{13} \\ f_{7} \\ f_{5} \\ \hline f_{1} \\ f_{3} \\ f_{9} \\ f_{11} \\ \hline \end{array} \right. \begin{array}{c} \hat{b}_0 \\ \hat{b}_1 \end{array} , \qquad (4.131)$$

т. е.

$$\left[\begin{array}{c} \hat{t}_1 \\ \hat{t}_0 \end{array} \right] = \left[\begin{array}{cc} \hat{a}_1 & \hat{a}_0 \\ \hat{a}_0 & \hat{a}_1 \end{array} \right] \left[\begin{array}{c} \hat{b}_0 \\ \hat{b}_1 \end{array} \right] . \qquad (4.132)$$

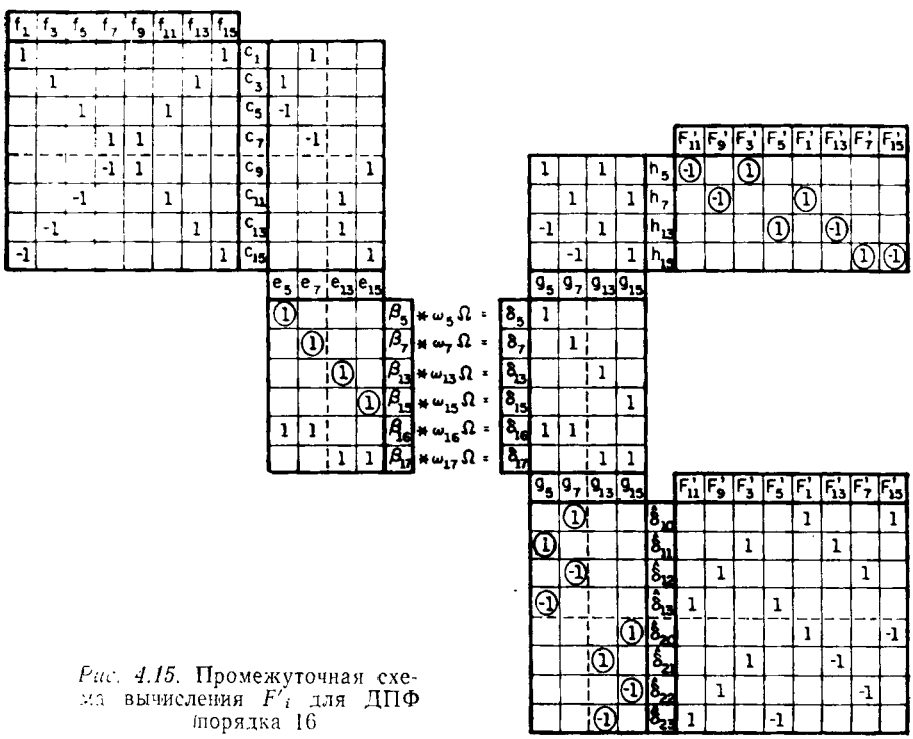


Рис. 4.15. Промежуточная схема вычисления F'_i для ДПФ порядка 16

Обработаем (4.132) по схемам (4.127), (4.128). Из (4.128) очевидно, что $\hat{\alpha}_1$, $\hat{\alpha}_2$ будут ЛЦ-матрицами. Следовательно, для их вычисления достаточно иметь первые строки. Выразим их через базовый угол

$$\theta = 360°/16 = 22{,}5°. \tag{4.133}$$

$$[\text{Первая строка } \hat{\alpha}_1] = \Omega[\cos\theta \sin\theta — \cos\theta — \sin\theta], \tag{4.134}$$

$$[\text{Первая строка } \hat{\alpha}_2] = i\Omega[\sin\theta \cos\theta — \sin\theta — \cos\theta]. \tag{4.135}$$

Обращаясь к (4.127), получаем:

$$\hat{\beta}_1 = \begin{bmatrix} f_{15}+f_1 \\ f_{13}+f_3 \\ f_7+f_9 \\ f_5+f_{11} \end{bmatrix} = \begin{bmatrix} c_1 \\ c_3 \\ c_7 \\ c_5 \end{bmatrix}, \quad \hat{\beta}_2 = \begin{bmatrix} f_{15}-f_1 \\ f_{13}-f_3 \\ f_7-f_9 \\ f_5-f_{11} \end{bmatrix} = \begin{bmatrix} c_{15} \\ c_{13} \\ -c_9 \\ -c_{11} \end{bmatrix}. \tag{4.136}$$

Следующий шаг в (4.127) требует вычисления двух ЛЦ-преобразований порядка 4, а именно:

$$\hat{\delta}_i = \hat{\alpha}_i \hat{\beta}_i \quad (i = 1, 2). \tag{4.137}$$

Выберем следующие обозначения для компонент $\hat{\delta}_i$:

$$\delta_i = \begin{bmatrix} \hat{\delta}_{i0} \\ \hat{\delta}_{i1} \\ \hat{\delta}_{i2} \\ \hat{\delta}_{i3} \end{bmatrix}. \tag{4.138}$$

Теперь их можно определить с помощью схемы на рис. 4.2.

Реализация $\hat{\delta}_1 = \hat{\alpha}_1\hat{\beta}_1$

Исчезновение первых двух множителей в (4.139) означает, что необходима только часть схемы рис. 4.2, включающая строки β_3, β_4, β_5. Эта часть переписывается в строки β_5, β_7, β_{16} на рис. 4.15 (этим объясняются индексы у ω). Заметим, что на рис. 4.15 указаны два альтернативных пути, ведущих от g_i к F'_r. Верхний путь сейчас можно игнорировать. Предыдущие относились к $\hat{\delta}_{1j}$. Обратимся теперь к $\hat{\delta}_{2j}$.

Реализация $\hat{\delta}_2 = \hat{\alpha}_2\hat{\beta}_2$

В (4.140) это очень напоминает случай $\hat{\delta}_1$. Строки β_3, β_4, β_5 на рис. 4.2 переписываем теперь в строки β_{13}, β_{15}, β_{17} на рис. 4.15.

$$
\begin{array}{|cccc|}
\hline
-\frac{\Omega}{4}\sin\theta & 1 & & 1 \\
-\frac{\Omega}{4}\cos\theta & & 1 & 1 \\
\frac{\Omega}{4}\sin\theta & 1 & & -1 \\
\frac{\Omega}{4}\cos\theta & & 1 & -1 \\
\hline
\end{array}
$$

$$
\begin{array}{|cccc|}
\hline
0 & 0 & -\frac{\Omega}{2}\cos\theta & -\frac{\Omega}{2}\sin\theta \\
\hline
\end{array}
$$

$$
\begin{array}{|cccc|c|l|}
\hline
1 & 1 & & & 0 & \\
1 & -1 & & & 0 & \\
 & & (1) & & -\frac{\Omega}{2}\cos\theta & = \varepsilon_5\Omega;\ \omega_5 = 2\varepsilon_{16} \\
 & & & & & \quad = -(\cos\theta + \sin\theta) \\
 & & & (1) & -\frac{\Omega}{2}\sin\theta & = \varepsilon_7\Omega;\ \omega_7 = 2(\varepsilon_7 - \varepsilon_5) \\
 & & & & & \quad = (\cos\theta - \sin\theta) \\
 & & 1 & 1 & -\frac{\Omega}{2}(\cos\theta + \sin\theta) & = \varepsilon_{16}\Omega;\ \omega_{16} = -2\varepsilon_7 \\
 & & & & & \quad = \sin\theta \\
\hline
\end{array}
\quad \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} (4.139)
$$

$$
\begin{array}{|cccc|}
\hline
-\frac{i\Omega}{4}\cos\theta & 1 & & 1 \\
-\frac{i\Omega}{4}\sin\theta & & 1 & 1 \\
\frac{i\Omega}{4}\cos\theta & 1 & & -1 \\
\frac{i\Omega}{4}\sin\theta & & 1 & -1 \\
\hline
\end{array}
$$

$$
\begin{array}{|cccc|}
\hline
0 & 0 & -\frac{i\Omega}{2}\sin\theta & -\frac{i\Omega}{2}\cos\theta \\
\hline
\end{array}
$$

$$
\begin{array}{|cccc|c|l|}
\hline
1 & 1 & & & 0 & \\
1 & -1 & & & 0 & \\
 & & (1) & & -\frac{i\Omega}{2}\sin\theta & = \varepsilon_{13}\Omega;\ \omega_{13} = 2\varepsilon_{17} \\
 & & & & & \quad = -i(\sin\theta + \cos\theta) \\
 & & & (1) & -\frac{i\Omega}{2}\cos\theta & = \varepsilon_{15}\Omega;\ \omega_{15} = 2(\varepsilon_{15} - \varepsilon_{13}) \\
 & & & & & \quad = i(\sin\theta - \cos\theta) \\
 & & 1 & 1 & -\frac{i\Omega}{2}(\sin\theta + \cos\theta) & = \varepsilon_{17}\Omega;\ \omega_{17} = -2\varepsilon_{15} \\
 & & & & & \quad = i\cos\theta. \\
\hline
\end{array}
\quad \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} (4.140)
$$

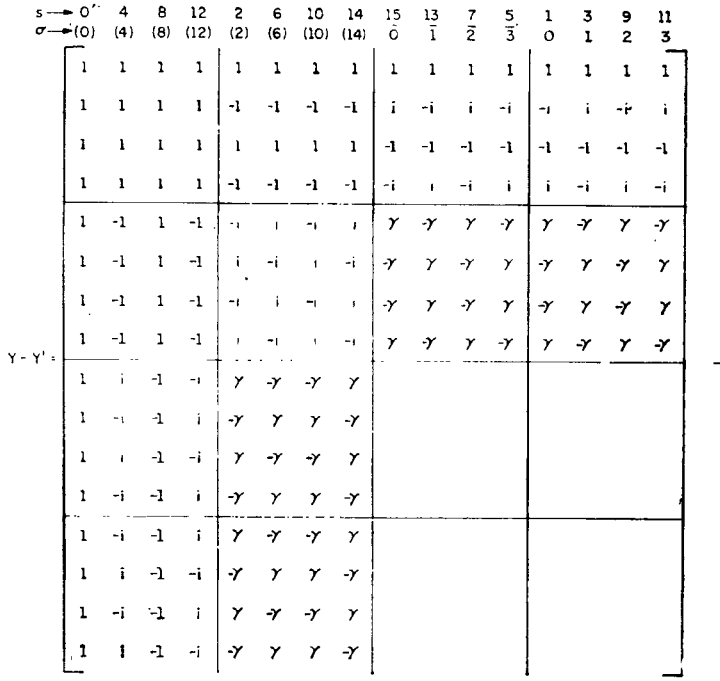


Рис. 4.16. Матрица E для вычисления F_i—F'_i

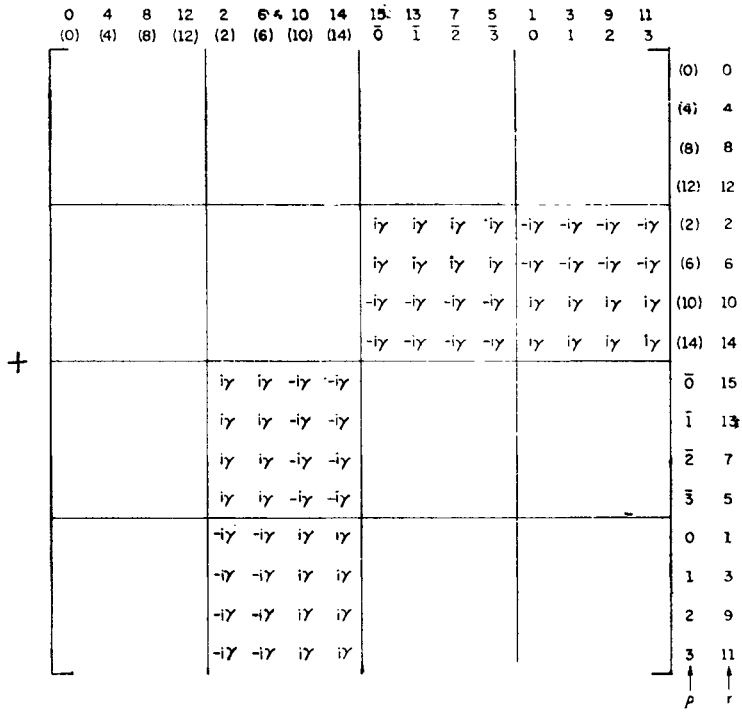
Имея $\hat{\delta}_1$, $\hat{\delta}_2$, применим (4.127) для получения $\hat{t}_1$, $\hat{t}_0$, как показано в нижней части рис. 4.15. Переход от g_i к F'_τ (см. нижнюю часть рисунка) требует 8 сложений. Верхняя часть реализует то же самое преобразование, только за 4 сложения. Именно этот вариант перенесен на рис. 4.17. В идентичности обоих путей можно легко убедиться. Например, в верхней части $F'_1 = g_7 + g_{15}$, но то же самое имеет место и в нижней части. Прослеживание таких соответствий по всем восьми выходам позволяет установить идентичность обоих вариантов.

Рассмотрим теперь реализацию оставшихся частей (4.130). Для того чтобы подчеркнуть использованные здесь симметрии, на рис. 4.16 в явной форме показана оставшаяся часть матрицы ДПФ с выполненными перестановками. Для удобства эта матрица представлена здесь в виде суммы двух матриц и выражена через константу

$$\gamma = 1/\sqrt{2}. \tag{4.141}$$

Теперь определим

$$c_{14} = f_6 - f_{14},\ c_{12} = f_4 - f_{12},\ c_{10} = f_2 - f_{10},\ c_8 = f_0 - f_8$$



для ДПФ порядка 16

и проведем вывод следующим образом.

$r=1;\ 9$

$$F_r-F'_r=\Omega\{\underbrace{(c_8-ic_{12})}_{\beta_{12}}-\gamma[\underbrace{(c_{14}-c_{10})}_{e_{10}}+i\underbrace{(c_{14}+c_{10})}_{e_{14}}]\}=$$

$$=\underbrace{\Omega\beta_{12}}_{\delta_{12}}-i\gamma\Omega\underbrace{(e_{14}-ie_{10})}_{\beta_{10}}=\underbrace{\delta_{12}}_{g_{12}}-i\underbrace{\gamma\Omega\beta_{10}}_{\delta_{10}}=$$

$$=g_{12}-\underbrace{\delta_{10}}_{g_{10}}=g_{12}-g_{10}=h_{12},\ \therefore\ F_r=F'_r+h_{12}.$$

$r=5;\ 13$

$$F_r=F'_r+\underbrace{(g_{12}+g_{10})}_{h_{10}}=F'_r+h_{10}.$$

$r=3;\ 11$

$$F_r-F'_r=\Omega\{\underbrace{(c_8+ic_{12})}_{\beta_8}+\gamma[\underbrace{(c_{14}-c_{10})}_{e_{10}}-i\underbrace{(c_{14}+c_{10})}_{e_{14}}]\}=$$

$$=\underbrace{\Omega\beta_8}_{\delta_8}-i\gamma\Omega\underbrace{(e_{14}+ie_{10})}_{\beta_{14}}=\underbrace{\delta_8}_{g_8}-i\underbrace{\gamma\Omega\beta_{14}}_{\delta_{14}}=g_8-\underbrace{\delta_{14}}_{g_{14}}=$$

$$=g_8-g_{14}=h_8,\ F_r=F'_r+h_8.$$

$r=7;\ 15$

$$F_r = F'_r + \underbrace{(g_8 + g_{14})}_{h_{14}} = F^e_r + h_{14}.$$

Теперь определим

$$c_0 = f_0 + f_8,\ c_2 = f_2 + f_{10},\ c_4 = f_4 + f_{12},\ c_6 = f_6 + f_{14},$$
$$g_4 = \delta_4 = \Omega\beta_4 = \Omega\,(c_0 - c_4)$$

и обратимся к группе строк (F_2, F_6, F_{10}, F_{14})

$$F_2 = g_4 + \Omega\{-\underbrace{\mathrm{i}\,(c_2 - c_6)}_{\beta_6} +$$

$$+\gamma\,[\underbrace{(c_7 + c_1)}_{e_1} - \underbrace{(c_5 + c_3)}_{e_3} + \mathrm{i}\,\underbrace{(c_{15} - c_9)}_{e_9} + \mathrm{i}\,\underbrace{(c_{13} - c_{11})}_{e_{11}}]\} =$$

$$= g_4 - \underbrace{\mathrm{i}\,\Omega\beta_6}_{\delta_6} + \gamma\Omega\,\underbrace{(e_1 - e_3)}_{\beta_3} + \mathrm{i}\,\gamma\Omega\,\underbrace{(e_9 + e_{11})}_{\beta_9} = g_4 - \underbrace{\delta_6}_{g_6} + \underbrace{\gamma\Omega\beta_3}_{\delta_3} +$$

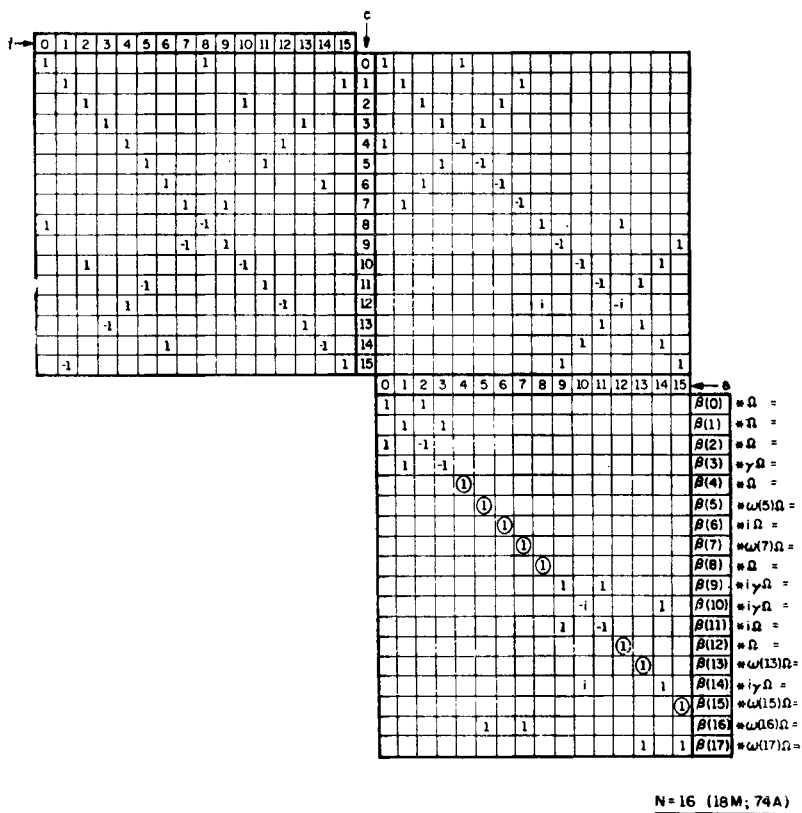


Рис. 4.17. Алгоритм ДПФ

$$+\underbrace{i\gamma\Omega\beta_0}_{\delta_0} = g_4 - g_6 + \underbrace{\delta_3}_{g_3} + \underbrace{\delta_9}_{g_9} = \underbrace{(g_4 - g_6)}_{h_4} + \underbrace{(g_9 + g_3)}_{h_3} = h_4 + h_3.$$

Отмечая на рис. 4.16 столбцы, связанные с каждым из четырех g_i, входящих в F_2, видим, что те же самые g_i входят в F_6, F_{10}, F_{14}, но с другими знаками:

$$F_{10} = (g_4 - g_6) - (g_9 + g_3) = h_4 - h_3,$$

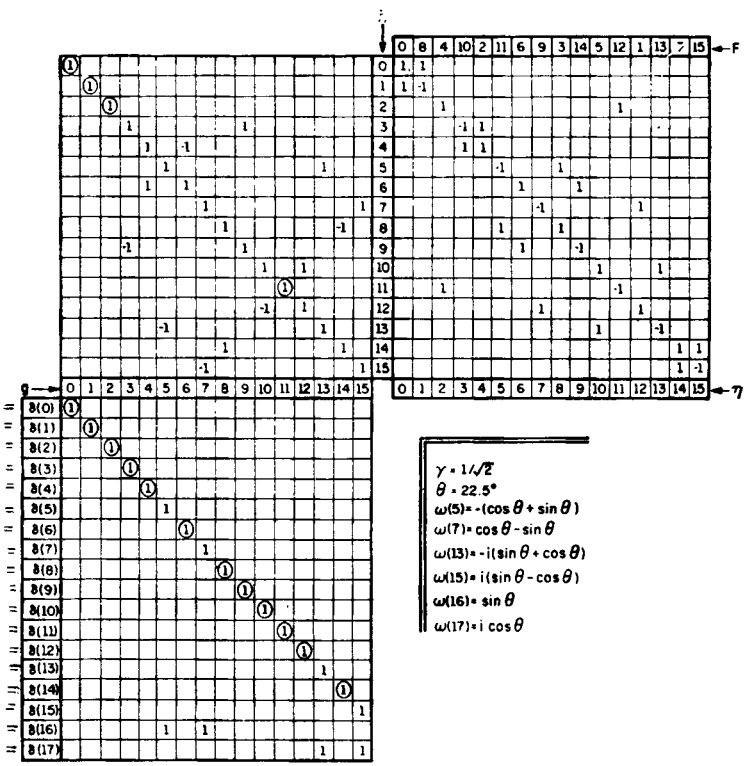
$$F_6 = \underbrace{(g_4 + g_6)}_{h_6} + \underbrace{(g_9 - g_3)}_{h_9} = h_6 + h_9,$$

$$F_{14} = (g_4 + g_6) - (g_9 - g_3) = h_6 - h_9.$$

Наконец, рассмотрим четыре верхние строки

$$F_0 = \Omega[\underbrace{(c_0 + c_4)}_{e_0} + \underbrace{(c_2 + c_6)}_{e_2} + \underbrace{(c_7 + c_1)}_{e_1} +$$

$$+ \underbrace{(c_5 + c_3)}_{e_3}] = \Omega[\underbrace{(e_0 + e_2)}_{\beta_0} + \underbrace{(e_1 + e_3)}_{\beta_1}] =$$



порядка 16

$$= \underbrace{\Omega\beta_0}_{\delta_0} + \underbrace{\Omega\beta_1}_{\delta_1} = \underbrace{\delta_0}_{g_0} + \underbrace{\delta_1}_{g_1} = \underbrace{g_0}_{h_0} + \underbrace{g_1}_{h_1} = h_0 + h_1,$$

$$\therefore F_8 = h_0 - h_1,$$

$$F_4 = \Omega\{\underbrace{(e_0 - e_2)}_{\beta_2} + i[\underbrace{(c_{15} - c_9)}_{e_9} - \underbrace{(c_{13} - c_{11})}_{e_{11}}]\} =$$

$$= \underbrace{\Omega\beta_2}_{\delta_2} + i\,\Omega\underbrace{(e_9 - e_{11})}_{\beta_{11}} = \underbrace{\delta_2}_{g_2} + i\,\underbrace{\Omega\beta_{11}}_{\delta_{11}} = \underbrace{g_2}_{h_2} + \underbrace{\delta_{11}}_{g_{11}} =$$

$$= h_2 + \underbrace{g_{11}}_{h_{11}} = h_2 + h_{11},$$

$$\therefore F_{12} = h_2 - h_{11}.$$

На этом вывод завершается. (Заметим, что размеры схемы на рис. 4.17 накладывают определенные ограничения на обозначения, в частности, $\omega(i) = \omega_i$, $\beta(i) = \beta_i$ и т. д.)

4.7. Общий алгоритм [1]

Теперь у нас, наконец, есть схемы ДПФ всех порядков, перечисленных в (4.3). Объединим их в алгоритм ДПФ порядка

$$N = \prod_{k=1}^{\varkappa} N_k \quad (N_k \text{ — взаимно-простые числа}). \qquad (4.142)$$

Наш метод опирается на идею блочной матрицы ДПФ, введенной в последнем разделе. Проиллюстрируем его простым примером. Предположим, мы хотим реализовать ДПФ порядка $N = 120 = 8 \cdot 15$. Можно ли использовать схему восьмого порядка для решения этой задачи? Можно было бы, конечно, рассмотреть исходную матрицу ДПФ как блочную матрицу порядка 8 с блоками порядка 15. Однако применение схемы порядка 8 оправдано только при условии, что эта блочная матрица отображает структуру блочной матрицы ДПФ порядка 8. Простая проверка показывает, что здесь это определенно не так. Тем не менее это обстоятельство не отменяет выбранный подход. Мысленно можно переставить элементы во входном и выходном векторах таким образом, чтобы матрица преобразования превратилась в блочную матрицу ДПФ порядка 8. В этом случае, несомненно, можно было бы применить схему порядка 8 (см. рис. 4.14). Однако на этом пути мы очень скоро столкнулись бы с другим препятствием: схема требует вычисления $\Omega\beta_i$, где Ω является теперь матрицей порядка 15, входящей в качестве блока $(0, 0)$ в блочную матрицу ДПФ с переставленными элементами. В схеме необходимо реализовать 8 таких преобразований, при этом для обеспечения высокой вычислительной эффективности алгоритма нам надо избегать прямого умно-

[1] Идея, положенная в основу выводов этого раздела, обычно приписывается Гуду [4.10]. Мы сочли более удобным избежать использования в явном виде кронекеровского произведения матриц.

жения матриц. Здесь появляется привлекательная возможность: можно предложить дополнительные ограничения на начальную перестановку, так чтобы Ω превратилась в блочную матрицу ДПФ, скажем, третьего порядка. Если бы такая возможность была, это позволило бы нам использовать преимущества высокой эффективности схемы ДПФ порядка 3 и, таким образом, реализовать ДПФ порядка 120 с помощью схем порядков 8, 3, 5.

Обратимся теперь к более точной формулировке упомянутой идеи, начиная с разработки необходимой терминологии. Будем считать, что индексы в (4.142) отражают порядок применения схем. Так, N_1 — порядок первой реализуемой схемы, N_2 соответствует второй схеме и т. д. В нашем примере

$$\varkappa = 3, N_1 = 8, N_2 = 3, N_3 = 5, \ldots N = 120. \tag{4.143}$$

Пусть Ω_0 будет исходной матрицей ДПФ порядка N с переставленными элементами. Если рассматривать ее как блочную матрицу порядка N_1, то каждый ее блок будет иметь порядок $\nu_1 = N/N_1$. Обозначим блок (0,0) через Ω_1. Следующий шаг в реализации основывается на рассмотрении Ω_1 как блочной матрицы ДПФ порядка N_2. Ее блок (0,0), обозначенный Ω_2, имеет порядок $\nu_2 = \nu_1/N_2$ и т. д. Это приводит к следующим определениям:

$$\nu_i = \prod_{k=i+1}^{\varkappa} N_k \, (0 \leqslant i < \varkappa), \nu_\varkappa = 1. \tag{4.144}$$

Пусть $\Omega_0 (= Y)$ будет матрицей ДПФ порядка N с переставленными элементами. Тогда Ω — верхняя левая подматрица порядка ν_i. Заметим, что (4.144) подразумевает существование последовательности таких подматриц: Ω_0, Ω_1, ..., $\Omega_\varkappa$. Порядок подматриц понижается слева направо. Ω_0 слева имеет порядок N и поэтому идентична полной матрице. $\Omega_\varkappa$ справа имеет порядок 1 и поэтому является верхним левым элементом полной матрицы.

Теперь можно сформулировать набор достаточных условий для перестановок элементов, которые позволили бы реализовать упомянутую схему:

1. Ω_0 должна быть блочной матрицей порядка N_1, т. е. блочной матрицей, блок (m, n) которой является матрицей Ω_1, умноженной на W^{mn}, где $W = \exp(-i 2\pi/N_1)$ [см. (4.9)]. Это обеспечит реализацию фазы 1 схемы порядка N_1. Фаза 2 требует определения $(\omega_i \Omega_1) \beta_i$. Следовательно,

2. Ω_1 должна быть блочной матрицей ДПФ порядка N_2. Мы используем фазу 1 схемы порядка N_2 и на фазе 2 приходим к преобразованию, включающему Ω_2. Следовательно,

3. Ω_2 должна быть блочной матрицей ДПФ порядка N_3.

. .
. .
. .

k. $\Omega_{\varkappa-1}$ должна быть блочной матрицей ДПФ порядка $\varkappa$.

Подведем итоги сказанному. Приемлемой схемой изменения индексов была бы схема, удовлетворяющая следующим требованиям:

Подматрица Ω_{k-1} матрицы Y должна быть блочной матрицей ДПФ порядка N_k. Это должно выполняться для всех $1 \leqslant k \leqslant \varkappa$. (4.145)

Хотя основанный на предыдущих рассуждениях алгоритм был бы вполне удобен с точки зрения его реализации, однако не ясно, существует ли на самом деле схема изменения индексации, удовлетворяющая одновременно всем $\varkappa$ требованиям (4.145).

Приступим теперь к разработке схемы изменения индексации, которая быстро приводит к (4.145), лишь незначительно усложняя приведенный выше эскиз алгоритма. Начнем со стандартной матрицы ДПФ [см. (4.9) при $\Omega = 1$] [1]:

$$\hat{F}_u = \sum_{v=0}^{N-1} W^{uv} \hat{f}_v \, (u = 0, 1, \ldots, N-1). \tag{4.146}$$

Пусть

$$Q_m = \hat{F}_u, \; q_n = \hat{f}_v, \tag{4.147}$$

где

$$u = \lambda(m), \; v = \lambda(n) \; (m, n = 0, 1, \ldots, N-1), \tag{4.148}$$

а функцию λ еще предстоит определить. Тогда (4.146) преобразуется в

$$Q_m = \sum_{n=0}^{N-1} W^{\lambda(m)\lambda(n)} q_n = \sum_{n=0}^{N-1} Y_{mn} q_n \, (m = 0, 1, \ldots, q-1). \tag{4.149}$$

Функция λ будет определена в терминах модульного представления [4.7] u и v. Другими словами, изменение индексации зависит от величин

$$\left.\begin{aligned} u_k &= u \bmod N_k \\ v_k &= v \bmod N_k \end{aligned}\right\} \; (k = 1, 2, \ldots, \varkappa). \tag{4.150}$$

В соответствии с китайской теоремой об остатках [4.7] эти остатки единственным образом определяют любые $0 \leqslant (u, v) < N$. Следовательно, можно выбрать следующее представление для u, v:

$$\left.\begin{aligned} u &= (u_1, u_2, \ldots, u_\varkappa) \\ v &= (v_1, v_2, \ldots, v_\varkappa) \end{aligned}\right\}. \tag{4.151}$$

---

[1] Здесь применены обозначения $\hat{F}_u$, $\hat{f}_v$ с тем, чтобы отличить их от используемых в схемах переменных F_u, f_v.

Теперь функция λ определяется с помощью объединения (4.148), (4.151):

$$v=(v_1, v_2, \ldots, v_\varkappa)=\lambda(n) \tag{4.152}$$

следующим образом:

> $\lambda(n)$ должна быть такова, что, когда n пробегает значения 0, 1, ..., $N-1$, v_k изменяются, периодически принимая значения из последовательности 0, 1, ..., N_k-1, начиная с $v_k=0$. v_k должно принимать новое значение при каждом увеличении v_k на n [см. (4.144)]. Это означает, что v_1 изменяется очень медленно, v_2 — быстрее и т. д., вплоть до $v_\varkappa$, которое изменяется синхронно с n.

Чтобы проиллюстрировать такую перестановку, рассмотрим пример (4.143), для которого часть индексной последовательности будет выглядеть так:

n	v
.	.
.	.
.	.
3	48 = (0, 0, 3)
4	24 = (0, 0, 4)
5	40 = (0, 1, 0)
6	16 = (0, 1, 1)
.	.
.	.
.	.
62	12 = (4, 0, 2)
63	108 = (4, 0, 3)
64	84 = (4, 0, 4)
65	100 = (4, 1, 0)
66	76 = (4, 1, 1)
.	.
.	.
.	.

Для определения этой последовательности с помощью ЭВМ можно воспользоваться подпрограммами модулярной арифметики. С другой стороны, ее можно построить по схеме, не применяя ни ЭВМ, ни какие-либо вычисления. Все, что потребуется, — это утомительное переписывание повторяющихся последовательностей. Этот метод для примера (4.143) проиллюстрирован в табл. 4.2, 4.3. Сначала выпишем последовательность 0, 1, ..., $N-1$, как показано в табл. 4.2 в колонке для v. Затем из колонки для v_k выпишем повторение последовательности 0, 1, ..., N_k-1 и повторим это для всех $1\leqslant k\leqslant\varkappa$. Так получаем представление (4.151) для всех v. Частичное упорядочение табл. 4.2 экономит усилия и удобно для выполнения следующего шага, который является просто переупорядочением табл. 4.2 в требуемую последовательность.

Чтобы упростить процесс, используем табл. 4.2 для определения порядка, в котором заполняется табл. 4.3. Например, первыми значениями v, копируемыми табл. 4.2 в табл. 4.3, являются 0,40,80,8,48,88,16,56, ... Табл. 4.3 теперь задает последовательность индексов у входного вектора с переставленными элементами для (4.143), а именно:

$$\hat{f}_0, \hat{f}_{96}, \hat{f}_{72}, \ldots, \hat{f}_{104}, \hat{f}_{105}, \ldots, \hat{f}_{89}, \hat{f}_{90}, \ldots, \hat{f}_{119}.$$

Чтобы облегчить анализ выбранного способа перестановки, введем E — экспоненциальную матрицу Y. Вспомним, что из (4.149) вытекает

$$Y_{mn}=W^{\lambda(m)\lambda(n)}. \tag{4.153}$$

Таблица 4.2

Модулярное представление v в примере (4.143)
$N_1=8$, $N_2=3$, $N_3=5$

v_1	r	v_2	v_3	r	v_2	v_3	r	v_2	v_3	r	v_2	v_3	r	v_2	v_3
0	0	0	0	8	2	3	16	1	1	24	0	4	32	2	2
1	1	1	1	9	0	4	17	2	2	25	1	0	33	0	3
2	2	2	2	10	1	0	18	0	3	26	2	1	34	1	4
3	3	0	3	11	2	1	19	1	4	27	0	2	35	2	0
4	4	1	4	12	0	2	20	2	0	28	1	3	36	0	1
5	5	2	0	13	1	3	21	0	1	29	2	4	37	1	2
6	6	0	1	14	2	4	22	1	2	30	0	0	38	2	3
7	7	1	2	15	0	0	23	2	3	31	1	1	39	0	4
0	40	1	0	48	0	3	56	2	1	64	1	4	72	0	2
1	41	2	1	49	1	4	57	0	2	65	2	0	73	1	3
2	42	0	2	50	2	0	58	1	3	66	0	1	74	2	4
3	43	1	3	51	0	1	59	2	4	67	1	2	75	0	0
4	44	2	4	52	1	2	60	0	0	68	2	3	76	1	1
5	45	0	0	53	2	3	61	1	1	69	0	4	77	2	2
6	46	1	1	54	0	4	62	2	2	70	1	0	78	0	3
7	47	2	2	55	1	0	63	0	3	71	2	1	79	1	4
0	80	2	0	88	1	3	96	0	1	104	2	4	112	1	2
1	81	0	1	89	2	4	97	1	2	105	0	0	113	2	3
2	82	1	2	90	0	0	98	2	3	106	1	1	114	0	4
3	83	2	3	91	1	1	99	0	4	107	2	2	115	1	0
4	84	0	4	92	2	2	100	1	0	108	0	3	116	2	1
5	85	1	0	93	0	3	101	2	1	109	1	4	117	0	2
6	86	2	1	94	1	4	102	0	2	110	2	0	118	1	3
7	87	0	2	95	2	0	103	1	3	111	0	1	119	2	4

Отсюда определяем экспоненциальную матрицу E следующим образом:

$$E_{mn}=\lambda(m)\,\lambda(n)=uv. \tag{4.154}$$

Обозначим ε_k как верхние левые подматрицы порядка ν_k матрицы E, подобно тому как выделялись подматрицы Ω_k из Y.

Рассмотрим теперь распределение аргументов u_k, v_k в ε_{k-1}. Перестановка определяет, что u_k должно увеличиваться на единицу в каждых ν_k строках, начиная с нулевого значения в верхнем левом углу. Подобным образом ν_k должно увеличиваться на единицу в каждых ν_k столбцах. Это, в свою очередь, определяет разбиение ε_{k-1} на подматрицы порядка ν_k. Матрица ε_{k-1} может рассматриваться теперь как блочная матрица порядка $N_k(=\nu_{k-1}/\nu_k)$, блок (r, s) которой характеризуется

$$u_k=r,\ v_k=s. \tag{4.155}$$

Очевидно, что блок (0,0) этой блочной матрицы идентичен ε_k. Выберем элемент в произвольной позиции в ε_k. Его индексы будут иметь вид

Таблица 4.3

Изменение индексации для примера (4.143)
n **через** $v=(v_1, v_2, v_3)$

v_1								v_2	t
0	1	2	3	4	5	6	7		
0	105	90	75	60	45	30	15	0	0
96	81	66	51	36	21	6	111	0	1
72	57	42	27	12	117	102	87	0	2
48	33	18	3	108	93	78	63	0	3
24	9	114	99	84	69	54	39	0	4
40	25	10	115	100	85	70	55	1	0
16	1	106	91	76	61	46	31	1	1
112	97	82	67	52	37	22	7	1	2
88	73	58	43	28	13	118	103	1	3
64	49	34	19	4	109	94	79	1	4
80	65	50	35	20	5	110	95	2	0
56	41	26	11	116	101	86	71	2	1
32	17	2	107	92	77	62	47	2	2
8	113	98	83	68	53	38	23	2	3
104	89	74	59	44	29	14	119	2	4

$$u=(0, \dots, 0, u_{k+1}, u_{k+2}, \dots, u_{\varkappa}),$$
$$v=(0, \dots, 0, v_{k+1}, v_{k+2}, \dots, v_{\varkappa}). \tag{4.156}$$

Выбранная схема перестановки гарантирует, что все скалярные элементы матрицы ε_{k-1}, занимающие те же самые позиции в других блоках ε_{k-1}, имеют u_i, v_i, которые отличаются от (4.156) только в u_k, v_k и удовлетворяют (4.155). Это означает, что разность между элементами в блоке (r, s) матрицы ε_{k-1} и соответствующим элементом в ε_k [блок (0,0)] равна {[4.7], (4.154)}

$$\Delta E(r, s)=(0, \dots, 0, \underset{k\text{-я позиция}}{rs \bmod N_k}, 0, \dots, 0). \tag{4.157}$$

Важно отметить, что эта разность не зависит от конкретного общего расположения двух парных элементов в их соответствующих блоках. Следовательно, она постоянна для всего блока (r, s). Другими словами, блок (r, s) из ε_{k-1} может быть получен из ε_k просто добавлением константы $\Delta E(r, s)$ ко всем его элементам.

Нам нужно явное выражение для этой важной константы. Учитывая неявную ее формулировку (4.157), приходим к выводу, что константа должна быть целым числом, кратным n_k, где

$$n_i=\prod_{\substack{k=1\\k\neq i}}^{\varkappa} N_k=\frac{N}{N_i}. \tag{4.158}$$

А именно, мы должны иметь

$$\Delta E(r, s) = \eta(r, s)\, n_k, \tag{4.159}$$

где целое η удовлетворяет

$$\eta < N_k, n_k\, \eta - rs \equiv 0 \pmod{N_k}. \tag{4.160}\ (4.161)$$

Соотношение (4.161) является линейным однородным относительно неизвестного η и имеет решение [4.8], а именно:

$$\eta = (rs') \bmod N_k, \tag{4.162}$$

где [1]

$$s' = (s\,\zeta_k) \bmod N_k,\ \zeta_k = n_k^{\varphi(N_k)-1} \bmod N_k. \tag{4.163}\ (4.164)$$

Результат, установленный для подматриц ε_{k-1}, распространяется на подматрицы Ω_{k-1} следующим образом: блок (r, s) матрицы Ω_{k-1} представляет собой $W^{\Delta E(r,s)}\Omega_k$. Используя (4.159), находим:

$$W^{\Delta E(r,s)} = \exp\left[-\mathrm{i}\,\frac{2\pi}{N_k}\,\eta(r, s)\right]. \tag{4.165}$$

Обозначим

$$W_k = \exp\left(-\mathrm{i}\,\frac{2\pi}{N_k}\right), \tag{4.166}$$

тогда

$$W^{\Delta E(r,s)} = W_k^{\eta(r,s)}, \tag{4.167}$$

так что блок (r, s) блочной матрицы Ω_{k-1} порядка N_k представляет собой

$$W_k^{\eta(r,s)}\,\Omega_k = W^{(rs')\bmod N_k}\,\Omega_k = W_k^{rs'}\,\Omega_k.$$

Это очень напоминает блок (r, s) блочной матрицы ДПФ (4.9) порядка N_k с $\Omega = \Omega_k$. Единственная разница состоит в том, что s заменено на s'. Эта, однако, тривиальная разница сводится только в перестановке столбцов. Для доказательства достаточно показать, что существует однозначное соответствие между s и s', так что когда s пробегает значения 0, 1, ..., $N_k - 1$, s' пробегает те же значения в переставленном порядке. А это, несомненно, так, если ζ_k в (4.163) и N_k — взаимно-простые. Но это гарантируется (4.164), поскольку n_k и N_k — взаимно-простые [см. (4.158)].

Отсюда можно сделать вывод, что тривиальная модификация схемы ДПФ порядка N_k позволит найти преобразование, определяемое матрицей Ω_{k-1}. А именно, мы модифицируем входной квадрат схемы N_k перестановкой строк/столбцов f_i так, чтобы последовательность i была идентична последовательности s' (вместо последовательности натуральных чисел s в немодифицированной схеме). Назовем такую схему «модифицированной схемой» и

[1] $\varphi(n)$ — это тотиентная функция Эйлера, которая определяется как количество целых чисел, взаимно-простых с n и меньших, чем n.

будем с этого момента использовать этот термин только в этом ограниченном точном значении.

Проиллюстрируем модификацию схемы для $k=1$ в примере (4.143):

$$n_1 = N_2 N_3 = 15, \varphi(N_1) = \varphi(8) = 4,$$
$$\zeta_1 = 15^{4-1} \bmod 8 = (-1)^3 \bmod 8 = 7.$$

Следовательно,

s	0	1	2	3	4	5	6	7
$s' = (7s) \bmod 8$	0	7	6	5	4	3	2	1

. (4.168)

Модифицированный входной квадрат, который требуется для (4.168), показан на рис. 4.18. Заметим, что в этом случае модификация заключается только в изменениях знака. Установленный результат можно сформулировать так:

Подматрица Ω_{k-1} из Y является результатом перестановки столбцов в блочной матрице ДПФ порядка N_k (4.9) с $\Omega = \Omega_k$. Это справедливо для всех $1 \leqslant k \leqslant \varkappa$. (4.169)

Отличие от (4.145) состоит в перестановке столбцов. Но, как уже отмечалось, это означает только то, что в алгоритме порядка N (4.142) стандартные схемы должны быть заменены на модифицированные. В этом и заключаются незначительные усложнения, о которых упоминалось выше.

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	
1				1				c_0
	1						1	c_1
		1				1		c_2
			1		1			c_3
1				-1				c_4
			1		-1			c_5
		1				-1		c_6
	1						-1	c_7

Рис. 4.18. Входной квадрат модифицированной схемы восьмого порядка для примера (4.143)

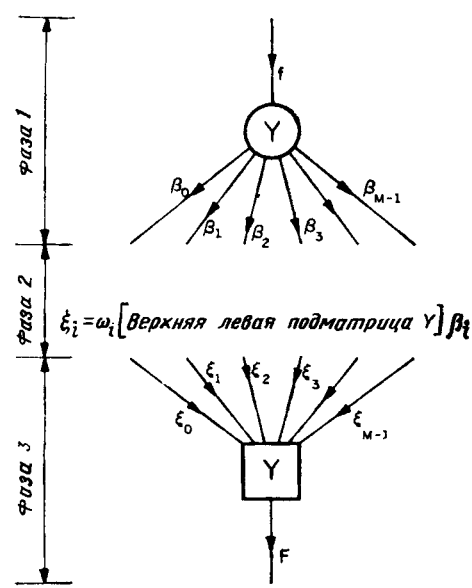


Рис. 4.19. Схематическое представление базовых схем ДПФ

В заключение приведем важную графическую схему общего алгоритма. Она основывается на символическом представлении схем, показанном на рис. 4.19. Для словесной формулировки будем использовать понятия обобщенных блочных матриц. Пусть схемные переменные f_i, β_i, δ_i, F_i, ... представляют m-мерные векторы. Тогда схема порядка N применима к любому матричному преобразованию порядка mN, в котором матрица преобразования Y, рассматриваемая как блочная матрица порядка N, является матрицей ДПФ порядка N (4.9). В этом случае параметр схемы Ω — верхняя левая подматрица матрицы Y, имеющая порядок m.

Изучение схем показывает, что все они состоят из трех частей, соответствующих трем различным фазам алгоритмов, которые ими описываются. На первой фазе обрабатывается mN-мерный входной вектор f, чтобы получить m-мерные векторы β_i. На второй фазе матрица Ω, умноженная на скаляр, преобразует β_i в ξ_i в соответствии с (4.86) $[\xi_i = (\omega_i\Omega)\beta_i,\ i=0, 1, ..., M-1]$[1]. M — это число умножений, определяемое назначением схемы. Очевидно, это также число векторов β_i, построенных на фазе *1*. И, наконец, на фазе *3* для получения mN-мерного выходного вектора F обрабатываются m-мерные векторы ξ_i. Три фазы представлены схематично на рис. 4.19 со следующими обозначениями. Все линии отображают векторы. С каждой линией связано целое число, показывающее размерность вектора, представляемого этой линией (рис. 4.20). Фаза *1* обозначается кружком, фаза *3* — квадратом. В каждом случае символ внутри фигуры обозначает матрицу, определяющую преобразования очерченного кругом входа в очерченный квадратом выход. Заметим, что единственными арифметическими операциями, включенными внутрь кружка или квадрата, являются сложения или вычитания.

Общий алгоритм приведен на рис. 4.20. Здесь показан конкретный пример для

$$\varkappa = 3,\ N_1 = 3,\ N_2 = 2,\ N_3 = 5,\ \ldots,\ N = 30. \qquad (4.170)$$

Модификация для любого другого случая выполняется совсем просто, как это будет показано позже. Первый шаг алгоритма — перестановка элементов входного вектора f, порождающая вектор q. Это действие и соответствующая перестановка вектора F преобразуют матрицу ДПФ в матрицу Y из (4.153). Теперь матрица Y разбивается так, как показано на рис. 4.21 (указанные «размеры» относятся к числу строк и столбцов). На следующем шаге $Y(=\Omega_0)$ рассматривается как блочная матрица третьего порядка ($N_1=3$) и используется модифицированная схема ДПФ третьего порядка со значением Ω на схеме, тождественно равным Ω_1 на рис. 4.21. Фаза *1* показана в верхней части рис. 4.20 (Ω_0 в кружке), фаза *3* — в нижней части (Ω_0 в квадрате), а фаза *2* — на остальной части рисунка между ними. На фазе *1* требуется

[1] Для большинства значений i, но не для всех, $\xi_i = \delta_i$. Например, из схемы для $N=5$ следует, что $\xi_0 = F_0$, $\delta_1 = \xi_1 + \xi_0$.

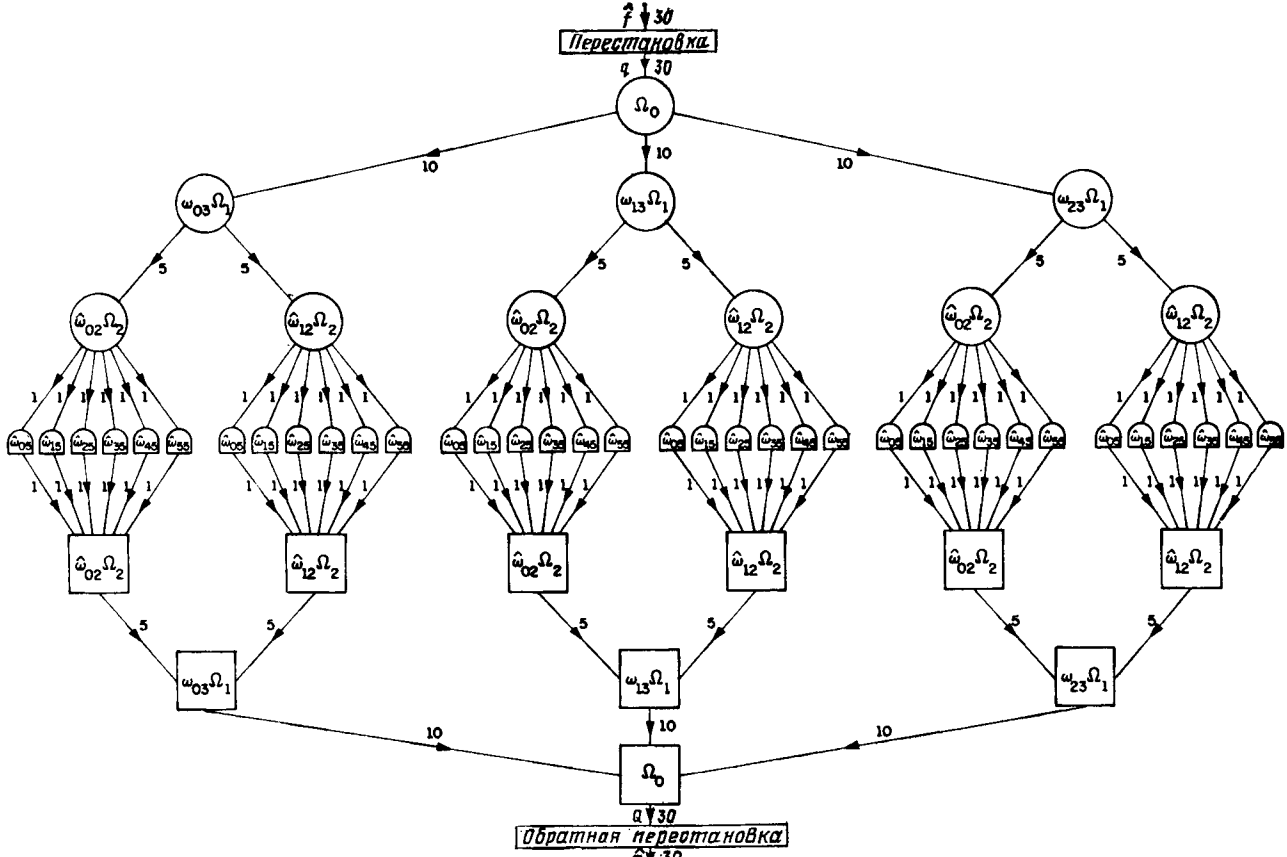


Рис. 4.20. Алгоритм ДПФ порядка 30 [пример (4.170)]

Таблица 4.4

Множители схемы ДПФ ($\omega_{k,N}$)

k \ N	2	3	4	5	7	8	9	16
0	1	1	1	1	1	1	1	1
1	1	−1,5	1	−1,25	−1,1666667E−0	7,0710678E−1	0,5	1
2		i8,6602540E−1	1	−5,5901699E−1	5,5854267E−2	1	−1,7364818E−1	1
3			1	i1,5388418E−0	7,3430220E−1	1	0,5	7,0710678E−1
4				−i3,6327126E−1	−i8,7484229E−1	1	9,3969262E−1	1
5				−i5,8778525E−1	−i5,3396936E−1	1	−i3,4202014E−1	−1,3065630E−0
6					−i4,4095855E−1	1	−i8,6602540E−1	i
7					7,9015647E−1	−i7,0710678E−1	−i9,8480775E−1	5,4119610E−1
8					−i3,4087293E−1		−i8,6602540E−1	1
9							7,6604444E−1	i7,0710678E−1
10							−i6,4278761E−1	i7,0710678E−1
11								i
12								1
13								−i1,3065630E−0
14								i7,0710678E−1
15								−i5,4119610E−1
16								3,8268343E−1
17								i9,2387953E−1

Примечание. Числа, представленные тремя или менее значащими цифрами, являются точными. Все остальные числа представлены в формате-E Фортрана (3,4E—1=3,4×10⁻¹).

разделение входного вектора на три его «компоненты». Это делается следующим очевидным образом:

$$f_0=\begin{bmatrix} q_0 \\ \vdots \\ q_9 \end{bmatrix}, \quad f_1\begin{bmatrix} q_{10} \\ \vdots \\ q_{19} \end{bmatrix}, \quad f_2=\begin{bmatrix} q_{20} \\ \vdots \\ q_{29} \end{bmatrix}. \qquad (4.171)$$

Выходом фазы *1* являются три вектора β_i размерности 10. Перед тем как перейти к рассмотрению фазы *2*, необходимо ввести обобщенные обозначения, которые используются для ω_i. Величины ω_i в схеме ДПФ порядка N будут обозначаться как $\omega_{i,N}$ ($\hat{\omega}_{i,N}$, которые также появляются на рис. 4.20, будут определены позже). Эти константы или явно указаны на схемах, или выводятся из условия, что каждое β_i умножается на $(\omega_i\Omega)$. Например, на схеме ДПФ порядка 5 явно указано, что $\omega_{1,5}=-5/4$. Очевидно, что $\omega_{0,5}=1$. Для удобства все значения ω_{ij} сведены в табл. 4.4. Эти значения вычислены на 10-разрядном калькуляторе. Более точные значения можно получить, используя явные выражения, указанные в схемах.

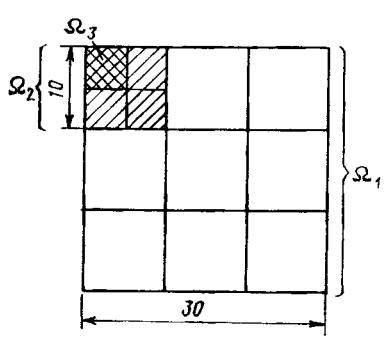


Рис. 4.21. Разбиение матрицы Y для примера (4.170)

Возвращаясь к рис. 4.20, обнаруживаем, что на фазе *2* схемы

третьего порядка требуется вычислить $\xi_i=(\omega_{i3}\Omega_1)\beta_i$. Здесь мы снова применяем результат (4.169), из которого вытекает, что для $k=2$ преобразование $(\omega_{i3}\Omega_1)\beta_i$ может быть вычислено с помощью модифицированной схемы ДПФ второго порядка [со схемным значением Ω, идентифицируемым как $(\omega_{i3}\Omega_2)$, см. рис. 4.21]. Поэтому каждый из трех 10-мерных векторов β_i показан на рис. 4.20 как вход первой фазы схемы второго порядка. В каждом из этих трех применений схемы второго порядка фаза *1* порождает пару векторов размерности 5. Рассмотрим теперь самый левый пятимерный вектор β на рис. 4.20. Он порожден фазой *1* схемы, реализующей преобразование, в основе которого лежит матрица $(\omega_{03}\Omega_1)$. Поэтому фаза *2* требует его преобразования с помощью матрицы $\omega_{02}(\omega_{03}\Omega_2)$. На рис. 4.20 вместо этого указана матрица $\hat{\omega}_{02}\Omega_2$. Здесь мы придерживаемся несколько необычной терминологии:

$$\hat{\omega}_{ij}=\omega_{ij}*(\text{первый множитель } \Omega_k, \text{ встречающийся при движении против стрелок в верхней половине рис. 4.20}). \tag{4.172}$$

Таким образом, $\hat{\omega}_{ij}$ определена только по отношению к конкретной диаграмме. Кроме того, тот же самый символ может иметь разные численные значения при разных расположениях на диаграмме. Например, мы только что видели (см. рис. 4.20), что на крайнем слева месте $\hat{\omega}_{02}=\omega_{02}\omega_{03}$. Этот же символ появляется еще два раза правее. В первом случае он равен $\omega_{02}\omega_{13}$, во втором — $\omega_{02}\omega_{23}$.

Вернемся к основному предмету наших рассуждений. Каждый из пятимерных векторов β теперь должен быть входом модифицированной схемы ДПФ порядка 5. На этот раз «векторы» β, полученные на фазе *1*, имеют размерность 1, т. е. фактически являются скалярами. Они должны умножаться на $\hat{\omega}_{i5}\Omega_3=\hat{\omega}_{i5}$ (Ω_3 — это скалярная 1). В этой точке фактически и выполняются умножения, и необходимо иметь численные значения множителей. Эти значения очевидны. Рассмотрим, например, самый левый множитель на рис. 4.20:

$$\hat{\omega}_{05}=\omega_{05}\,\hat{\omega}_{02}=\omega_{05}\,\omega_{02}\,\omega_{03}.$$

Подобным образом самый правый множитель на рис. 4.20 равен

$$\hat{\omega}_{55}=\omega_{55}\,\hat{\omega}_{12}=\omega_{55}\,\omega_{12}\,\omega_{23},$$

и вообще значение множителя в конкретном узле является произведением всех членов ω_{ij} (без значка «$\wedge$»), которые встречаются при движении по древовидной структуре вверх от рассматриваемого узла.

На рис. 4.20 эти умножения показаны в форме полукруга и полуквадрата, расположенных вдоль средней линии схемы. (Их

можно рассматривать как объединенные три фазы алгоритма ДПФ порядка 1 при $\Omega = \hat{\omega}_{i5}$.)

Получаемые после умножения 36 членов образуют 6 независимых групп, являющихся результатом шести отдельных применений модифицированной схемы порядка 5. Члены каждой группы объединяются теперь в соответствии с предписаниями фазы *3* схемы пятого порядка для получения шести векторов размерности 5. Каждый из них является членом вида $\hat{\omega}_{i2}\Omega_2\beta_i$, требующим для завершения вычислений в трех применениях схемы второго порядка. Эти вычисления порождают теперь в качестве выходов схемы три вектора размерности 10, которые идентичны $\omega_{i3}\Omega_1\beta_i$ для схемы третьего порядка. Объединение их в соответствии с предписаниями фазы *3* этой схемы порождают 30-мерный вектор Q, который с точностью до перестановки является нужным нам вектором F.

Хотя рис. 4.20 непосредственно применим только к примеру (4.170), он достаточно характерен и для всех значений N. Для больших N может появиться еще один уровень ветвления в каждой половине схемы, а число ответвлений от каждого узла может увеличиться. Но общая структура идентична показанной на рис. 4.20.

4.8 Оценка быстродействия

В разд. 4.4—4.6 мы построили базовые схемы ДПФ; в разд. 4.7 показали, как их использовать в эффективном алгоритме для некоторых значений N. В этом разделе нашей целью являются определение того, насколько быстрым является результирующий алгоритм, и получение сводной таблицы 4.6 соответствующих параметров для всех порядков, реализуемых с помощью сконструированных схем. В качестве основы будем использовать табл. 4.5, в которую сведены параметры разработанных схем. Все параметры ранее были полностью объяснены и указаны на изображении каждой схемы.

Таблица 4.5

Сводка параметров основных схем ДПФ

Порядок схемы N	Общее число умножений M	Число умножений на 1 или на i m	Число сложений A	Рисунок схемы
2	2	2	2	4.9
3	3	1	6	4.5
4	4	4	8	4.10
5	6	1	17	4.6
7	9	1	36	4.8
8	8	6	26	4.14
9	11	1	44	4.13
16	18	8	74	4.17

Рассмотрим алгоритм порядка $N=\prod_{k=1}^{\varkappa} N_k$, применяемый к комплексным данным[1]. Для каждого N_k из табл. 4.5 выбираются соответствующее число комплексных умножений M_k и комплексных сложений A_k. Нас интересуют две функции этих переменных — общее число вещественных умножений $\mathscr{M}$ и общее число вещественных сложений $\mathscr{A}$ для всего алгоритма, реализованного в порядке, предусмотренном (4.142) (фаза *1* схемы N_1 реализуется первой; фаза *1* схемы $N_\varkappa$ — последней). Теперь обратимся к математической формулировке некоторых характеристик алгоритма, структура которого совершенно очевидна из рис. 4.20.

Заметим, что выходом фазы *1* схемы N_1 является множество из M_1 векторов (β_i) размерности ν_1. Каждый из них порождает (на выходе фазы *1* схемы N_2) M_2 векторов размерности ν_2 и т. д. Очевидно поэтому, что

Общий выход фазы *1* всех схем порядка N_k состоит из $\prod_{i=1}^{k} M_i$ векторов размерности ν_k (4.144). (4.173)

Поскольку каждый из этих векторов поступает на вход фазы *1* схемы N_{k+1}, указанное в (4.173) число определяет также число схем порядка N_{k+1}, или, что эквивалентно,

$$\text{Число схем порядка } N_k \text{ равно } \prod_{i=1}^{k-1} M_i. \tag{4.174}$$

Простейшим применением полученных результатов является определение числа умножений. Пусть $\mathscr{M}_\varkappa$ — общее число (комплексных) скалярных умножений на фазе *2* алгоритма, реализованного в виде каскада из $\varkappa$ ступеней. Перемножаемые переменные являются одномерными векторами β_i, полученными на последней ступени каскада $(N_\varkappa)$. Из (4.173) известно, что существует $\prod_{i=1}^{\varkappa} M_i$ таких членов. Следовательно,

$$\mathscr{M}_\varkappa = \prod_{i=1}^{\varkappa} M_i. \tag{4.175}$$

Чтобы получить общее число вещественных умножений $(\mathscr{M})$ на весь каскад, отметим, что в каждом из учитываемых умножений только один из двух сомножителей является комплексным, а другой — вещественным или мнимым (см. табл. 4.4). Поэтому

$$\mathscr{M} = 2\,\mathscr{M}_\varkappa. \tag{4.176}$$

В разд. 4.7 показано, что каждый из полных множителей каскада является произведением $\varkappa$ схемных множителей, по одному из каждой схемы каскада. Поскольку каждая схема имеет по крайней мере один множитель, значение которого равно 1 или i

[1] Определение числа арифметических операций для вещественных данных является более сложным и будет лишь кратко рассмотрено позже.

(см. $m \geqslant 1$ в табл. 4.5), некоторые из $\mathscr{M}_\varkappa$ полных множителей будут 1 или i.

Рассмотрение структуры рис. 4.20 показывает, что число таких множителей [1]

$$P_\varkappa = \prod_{i=1}^{\varkappa} m_i . \tag{4.177}$$

Поэтому, если пойти на более сложное программирование, позволяющее выделить эти $P_\varkappa$ тривиальных множителей, ДПФ можно будет вычислить с меньшим числом вещественных умножений

$$\hat{\mathscr{M}} = 2(\mathscr{M}_\varkappa - P_\varkappa). \tag{4.178}$$

В итоговую табл. 4.6 включены оба случая [(4.176), (4.178)] [2]. Обратимся теперь к подсчету числа сложений. Пусть $\mathscr{A}_\varkappa$ будет общим числом (комплексных) скалярных сложений в последовательности из $\varkappa$ ступеней. Тогда соответствующее число вещественных сложений

$$\mathscr{A} = 2\mathscr{A}_\varkappa. \tag{4.179}$$

Простейший способ определения $\mathscr{A}$ состоит в использовании рекурсивных рассуждений. Предположим, что известен результат для последовательности длиной $(\varkappa-1)$, и мы добавляем еще одну $\varkappa$-ю ступень. Отсюда вытекают два обстоятельства. Во-первых, сложения на первых $(\varkappa-1)$ ступенях относятся теперь к векторам, размерность которых в $N_\varkappa$ раз больше их предшествующих значений. Следовательно, предыдущие $\mathscr{A}_{\varkappa-1}$ сложений превращаются в $N_\varkappa \mathscr{A}_{\varkappa-1}$ сложений. К этому необходимо прибавить число сложений на последней ступени. Из (4.174) число схем на последней ступени равно $\prod_{i=1}^{\varkappa-1} M_i = \mathscr{M}_{\varkappa-1}$ (4.175). Каждая такая схема требует $A_\varkappa$ сложений скалярных величин. Следовательно, число скалярных сложений на последней ступени равно $\mathscr{M}_{\varkappa-1}\mathscr{A}_\varkappa$, а общее число

$$\left.\begin{aligned} \mathscr{A}_\varkappa &= \mathscr{A}_{\varkappa-1} N_\varkappa + \mathscr{M}_{\varkappa-1} A_\varkappa, \\ \mathscr{M}_\varkappa &= \mathscr{M}_{\varkappa-1} M_\varkappa, \\ \mathscr{A}_1 &= A_1,\ \mathscr{M}_1 = M_1. \end{aligned}\right\} \tag{4.180}$$

Здесь для получения полной системы правил последовательного вычисления $\mathscr{A}_\varkappa$ и $\mathscr{M}_\varkappa$ добавлено выражение (4.176) в рекурсив-

---

[1] Все допустимые значения N выражаются как $N = H \cdot 2^r$ (H — нечетное, $0 \leqslant r \leqslant 4$). Используя это и значения m_i, перечисленные в табл. 4.5, получаем $P_\varkappa = 2r + \delta_{0,r}$.

[2] Могут возразить, что в двоичной машине умножение на 1/2 также является тривиальным и должно исключаться из подсчета числа умножений. Если придерживаться такой точки зрения, то схема порядка 9 будет иметь 3 тривиальных умножения ($m=3$). Это обстоятельство будет достаточно существенным для $N=144$, снижая $\mathscr{M}$ с 380 (см. табл. 4.6) до 348.

Сводка характеристик алгоритма ДПФ

Порядок ДПФ	С учетом умножений на 1 и i			Без учета умножений на 1 и i			Число вещественных сложений	Последовательность реализации схем $N_k(\zeta_k)$							
	Выигрыш в скорости как функция параметров $G(\mu)=G_\infty\frac{\mu+1.5}{\mu+R}$		Число вещественных умножений	Выигрыш в скорости как функция параметров $\hat{G}(\mu)=\hat{G}_\infty\frac{\mu+1.5}{\mu+\hat{R}}$		Число вещественных умножений		k				k			
N	G_∞	R	M	$\hat{G}_\infty$	$\hat{R}$	$\hat{M}$	A	1	2	3	4	1	2	3	4
2	1,000	1,000	4	–	–	0	4	2(1)							
3	1,585	2,000	6	2,377	3,000	4	12	3(1)							
4	2,000	2,000	8	–	–	0	16	4(1)							
5	1,935	2,833	12	2,322	3,400	10	34	5(1)							
6	2,585	3,000	12	3,877	4,500	8	36	3(2)	2(1)			2(1)	3(2)		
7	2,183	4,000	18	2,456	4,500	16	72	7(1)							
8	3,000	3,250	16	12,000	13,000	4	52	8(1)							
9	2,594	4,000	22	2,853	4,400	20	88	9(1)							
10	2,768	3,667	24	3,322	4,400	20	88	2(1)	5(3)						
12	3,585	4,000	24	5,377	6,000	16	96	3(1)	4(3)			4(3)	3(1)		
14	2,961	4,778	36	3,331	5,375	32	172	2(1)	7(4)						
15	3,256	4,500	36	3,447	4,765	34	162	3(2)	5(2)						
16	3,556	4,111	36	6,400	7,400	20	148	16(1)							
18	3,412	4,818	44	3,753	5,300	40	212	2(1)	9(5)						
20	3,602	4,500	48	4,322	5,400	40	216	4(1)	5(4)						
21	3,416	5,556	54	3,548	5,769	52	300	3(1)	7(5)						
24	4,585	5,250	48	6,113	7,000	36	252	3(2)	8(3)			8(3)	3(2)		
28	3,739	5,556	72	4,206	6,250	64	400	4(3)	7(2)						
30	4,089	5,333	72	4,330	5,647	68	384	3(1)	2(1)	5(1)		2(1)	3(1)	5(1)	
35	3,325	6,167	108	3,387	6,283	106	666	7(3)	5(3)						
36	4,230	5,636	88	4,653	6,200	80	496	4(1)	9(7)						
40	4,435	5,542	96	5,069	6,333	84	532	8(5)	5(2)						
42	4,194	6,333	108	4,355	6,577	104	684	3(2)	2(1)	7(6)		2(1)	3(2)	7(6)	
45	3,744	6,167	132	3,802	6,262	130	814	9(2)	5(4)						
48	4,964	5,889	108	5,828	6,913	92	636	3(1)	16(11)						
56	4,517	6,528	144	4,927	7,121	132	940	8(7)	7(1)						
60	4,922	6,167	144	5,212	6,529	136	888	3(2)	4(3)	5(3)		4(3)	3(2)	5(3)	
63	3,804	7,111	198	3,843	7,184	196	1 408	9(4)	7(4)						
70	3,973	6,815	216	4,048	6,943	212	1 472	2(1)	7(5)	5(4)					
72	5,048	6,659	176	5,417	7,146	164	1 172	8(1)	9(8)						
80	4,683	6,259	216	5,058	6,760	200	1 352	16(13)	5(1)						
84	4,972	7,111	216	5,163	7,385	208	1 536	3(1)	4(1)	7(3)		4(1)	3(1)	7(3)	
90	4,426	6,848	264	4,494	6,954	260	1 808	2(1)	9(1)	5(2)					
105	4,352	7,463	324	4,379	7,509	322	2 418	3(2)	7(1)	5(1)					
112	4,706	7,198	324	4,951	7,571	308	2 332	16(7)	7(4)						
120	5,756	7,208	288	6,006	7,522	276	2 076	3(1)	8(7)	5(4)		8(7)	3(1)	5(4)	
126	4,440	7,747	396	4,485	7,827	392	3 068	2(1)	9(2)	7(2)					
140	4,621	7,463	432	4,708	7,604	424	3 224	4(3)	7(6)	5(2)					
144	5,214	7,364	396	5,434	7,674	380	2 916	16(9)	9(4)						
168	5,750	8,083	432	5,914	8,314	420	3 492	3(2)	8(5)	7(5)		8(5)	3(2)	7(5)	
180	5,108	7,530	528	5,187	7,646	520	3 976	4(1)	9(5)	5(1)					
210	5,000	8,111	648	5,031	8,161	644	5 256	3(1)	2(1)	7(4)	5(3)	2(1)	3(1)	7(4)	5(3)
240	5,857	7,741	648	6,005	7,937	632	5 016	3(2)	16(15)	5(2)					
252	5,076	8,384	792	5,128	8,469	784	6 640	4(3)	9(1)	7(1)					
280	5,269	8,273	864	5,343	8,390	852	7 148	8(3)	7(3)	5(1)					
315	4,401	8,759	1,188	4,409	8,774	1 186	10 406	9(8)	7(5)	5(2)					
336	5,802	8,580	972	5,899	8,724	956	8 340	3(1)	16(13)	7(6)					
360	5,790	8,383	1,056	5,856	8,479	1 044	8 852	8(5)	9(7)	5(3)					
420	5,648	8,759	1,296	5,683	8,814	1 288	11 352	3(2)	4(1)	7(2)	5(4)	4(1)	3(2)	7(2)	5(4)
504	5,713	9,179	1,584	5,756	9,249	1 572	14 540	8(7)	9(5)	7(4)					
560	5,260	8,831	1,944	5,303	8,905	1 928	17 168	16(11)	7(5)	5(3)					
630	4,931	9,290	2,376	4,940	9,305	2 372	22 072	2(1)	9(4)	7(6)	5(1)				
720	5,753	8,970	2,376	5,792	9,031	2 360	21 312	16(5)	9(8)	5(4)					
840	6,296	9,569	2,592	6,326	9,614	2 580	24 804	3(1)	8(1)	7(1)	5(2)	8(1)	3(1)	7(1)	5(2)
1008	5,644	9,727	3,564	5,669	9,771	3 548	34 668	16(15)	9(7)	7(2)					
1260	5,462	9,820	4,752	5,471	9,836	4 744	46 664	4(3)	9(2)	7(3)	5(3)				
1680	6,173	9,984	5,832	6,190	10,011	5 816	58 224	3(2)	16(9)	7(4)	5(1)				
2520	5,992	10,483	9,504	6,000	10,496	9 492	99 628	8(3)	9(1)	7(5)	5(4)				
5040	5,798	10,939	21,384	5,802	10,948	21 368	233 928	16(3)	9(5)	7(6)	5(2)				

ном представлении и определены начальные условия. Выражение (4.180) дает также явную формулу для $\mathscr{A}_\varkappa$. Например,

$$\mathscr{A}_4 = A_1 N_2 N_3 N_4 + M_1 A_2 N_3 N_4 + \\ + M_1 M_2 A_3 N_4 + M_1 M_2 M_3 A_4. \qquad (4.181)$$

Важным и очевидным следствием (4.181) является то, что $\mathscr{A}_\varkappa$ в отличие от $\mathscr{M}_\varkappa$ является также функцией порядка, в котором $N_\varkappa$ входят в N. Поэтому важно найти порядок следования каскадов, минимизирующий $\mathscr{A}_\varkappa$.

Имея в своем распоряжении (4.175)—(4.180) и табл. 4.5, можем вычислить $\mathscr{A}$ и $\mathscr{M}_\varkappa$ для всех N, удовлетворяющих (4.142). В табл. 4.6 представлены результаты таких вычислений, выполненных с помощью простой программы, которая позволяет также получить данные для выбора наиболее эффективного варианта порядка следования каскадов. Эта последовательность N_k показана в последних столбцах табл. 4.6. Здесь в таблице оставлено место для двух последовательностей, так как в некоторых случаях одни и те же значения $\mathscr{M}$ и $\mathscr{A}$ получаются для двух разных последовательностей N_k. В таких случаях окончательный выбор определяется не эффективностью, а другими соображениями.

Справа от каждого значения N_k стоит число в скобках — значение ζ_k из (4.163). Таким образом, табл. 4.6 дает и последовательность схем, которая должна быть реализована, и перестановку, которую надо выполнить во входном квадрате каждой схемы [см. рассуждения, предшествующие (4.168)].

Заметим, что очень важно придерживаться порядка, предписанного табл. 4.6. Например, для $N=240$ таблица определяет значение $\mathscr{A}=5016$ с предписываемым порядком 3; 16; 5. Если вместо этого выбрать порядок 5; 16; 3, то число вещественных сложений возрастает до 5592, т. е. на 11%.

Обратимся теперь к двум оставшимся столбцам табл. 4.6, а именно, G_∞ и R. Оба параметра уже упоминались в разд. 4.1. Они требуются для определения выигрыша в скорости в конкретной системе.

Пусть

$$\mu = \frac{\text{время одного вещественного умножения}}{\text{время одного вещественного сложения}}$$

в рассматриваемой системе. Определим выигрыш G данного алгоритма по сравнению с алгоритмом Кули—Тьюки:

$$G = \frac{\mu \mathscr{M}_{\text{КТ}} + \mathscr{A}_{\text{КТ}}}{\mu \mathscr{M} + \mathscr{A}}, \qquad (4.182)$$

где $\mathscr{M}_{\text{КТ}}$, $\mathscr{A}_{\text{КТ}}$ — параметры Кули—Тьюки, введенные в (4.1). Очевидно, что G — это отношение времени, требующегося для алгоритма Кули—Тьюки, к времени, которое необходимо для алгоритма Винограда. Будем называть его выигрышем в скорости, хотя

следует принимать во внимание, что он отражает только время, занимаемое арифметическими вычислениями. Поэтому этот показатель не отражает большей сложности алгоритма Винограда, которая замедляет выполнение его на универсальной ЭВМ. Однако, когда рассматривается специализированный ДПФ-процессор, эта оценка выигрыша в скорости весьма правдоподобна.

G — функция μ и еще четырех параметров, появляющихся в (4.182). Более удобной формулой, содержащей только два параметра [см. (4.1)], является выражение

$$G(\mu) = G_\infty (\mu + 1{,}5)/(\mu + R), \tag{4.183}$$

где

$$G_\infty = \mathscr{M}_{\text{КТ}}/\mathscr{M}, \tag{4.184}$$

$$R = \mathscr{A}/\mathscr{M}, \tag{4.185}$$

G_∞ — это асимптотический выигрыш в скорости, который достигается при больших значениях μ; R является полюсом функции $G(\mu)$ при $(\mu = -R)$ и поэтому определяет вторую асимптоту. Добавление к этим двум параметрам нуля функций $G(\mu)$ при $\mu = -1{,}5$ позволяет легко представить себе вид функции и достаточно точно оценить ее. И, наконец, если заменить в (4.182)—(4.185) $\mathscr{M}$ на $\hat{\mathscr{M}}$, получим значения, приведенные в столбцах G, G_∞, R табл. 4.6.

В заключение сделаем несколько замечаний по поводу случая вещественных данных. Число арифметических операций вовсе не составляет половины соответствующего числа в случае комплексных данных[1]. Основная причина этого заключается в том, что, поскольку результат ДПФ вещественного вектора является в общем случае комплексным, некоторые из промежуточных результатов будут комплексными. Например, согласно схеме ДПФ восьмого порядка

$$\beta_5 = e_2 + \mathrm{i}\, e_5 \tag{4.186}$$

и поэтому β_5 является комплексным даже для вещественных данных. Это значит, что схема, реализующая $\xi_5 = \Omega\beta_5$, имеет на входе комплексные данные.

Другим фактором, который надо учитывать, является то, что представление комплексного числа в виде суммы двух его компонент не означает на самом деле никакого сложения, несмотря на наличие знака плюс. В упомянутом выше примере и e_2, и e_5 являются вещественными, когда вещественны f_i. Следовательно, «сложение», появившееся в (4.186), становится фиктивным.

---

[1] При использовании так называемых совмещенных алгоритмов, осуществляющих преобразования пар последовательностей вещественных чисел, число дополнительных операций сложения минимально и равно длине последовательности. — *Прим. ред.*

4.9. Заключительные замечания

Мы попытались здесь вывести алгоритм Винограда дискретного преобразования Фурье, начиная с общей концепции, формирования необходимых конструктивных блоков и кончая детальным описанием объединения этих блоков в общий алгоритм.

Стартовой точкой в применении алгоритма является табл. 4.6, поскольку она представляет список допустимых значений N и соответствующих параметров, характеризующих производительность алгоритма. После выбора нужного N следует обратиться к табл. 4.5 и отыскать конкретные схемы, требующиеся в соответствии с табл. 4.6. При реализации алгоритма сначала осуществляется перестановка элементов входного вектора, а затем применяется фаза *1* схем (в том порядке, который предписан табл. 4.6) к все меньшим и меньшим сегментам вектора данных, которые оказываются в различных частично преобразованных состояниях. Этот процесс завершается, когда, в конце концов, однокомпонентные «сегменты» умножаются на константы в фазе *2*. С этого момента процесс развивается в обратном направлении на фазе *3*. Скаляры, появившиеся как результат фазы *2*, комбинируются в векторы все большей и большей размерности, превращаясь, наконец, в N-мерный вектор, который отличается от результата преобразования лишь перестановкой своих элементов.

Эта глава содержит подробную информацию, достаточную для непосредственной аппаратурной или программной реализации упомянутой выше схемы. Хотя мы не рассмотрели конкретной реализации, стоит обратить внимание на некоторые особенности схем, специально внесенные в них для облегчения их реализации. Мы имеем в виду преобразование с оставлением на месте. Рассмотрим схему седьмого порядка (см. рис. 4.8). Заметим, что исходные компоненты f_2, f_5 используются только для вычисления c_2, c_5. Следовательно, нет необходимости выделять дополнительную память для c_2, c_5. Они могут храниться в массиве f на месте f_2, f_5. Единственно, что требуется, — это временно сохранить, скажем, f_2, так чтобы, запомнив c_2, можно было бы использовать f_2 для вычисления c_5. Даже если f_2 является вектором, необходимо лишь одно слово памяти для временного хранения, так как вычисления выполняются покомпонентно.

Это свойство, которое мы проиллюстрировали здесь на примере преобразования $(f_2, f_5) \rightarrow (c_2, c_5)$, является общим для всех переменных в схемах порядков 2, 3, 5, 7. Оно остается в силе и для остальных схем, если рассматривать η как выходной вектор. Единственное отклонение заключается в том, что в некоторых случаях должны рассматриваться группы из трех, а не из двух компонент. В выбранной схеме таким случаем является вычисление β_1, β_2, β_3. Однако даже в этом случае достаточно иметь временную память на одно слово.

Надо особо отметить, что в тех применениях, где свойство оставления на месте не используется, схемы порядков 4, 9, 8, 16 мо-

гут быть несколько упрощены за счет перестановки F_i строк/столбцов в выходном квадрате, чтобы получить выходной вектор F без перестановки элементов. В этом случае, конечно, можно обойтись без вектора η.

Обратимся теперь к краткому рассмотрению вопроса потери точности, упомянутому в разд. 4.1. В приложении (см. последний раздел) некоторые преобразования, применявшиеся при построении схем, отрицательно сказываются на точности. Подобная ситуация имеет место и при вычислении δ_1 в некоторых схемах. Анализ (4.84), (4.85) показывает, что выбранный вид выражения (4.85) включает в себя сложение и вычитание $\Omega\beta_1$. Следовательно, если $|\Omega\beta_1| \gg |\delta_1|$, мы непременно столкнемся с проблемой потери значащих битов в арифметике с плавающей запятой и возможностью переполнения в арифметике с фиксированной запятой. Подобные же манипуляции с операциями сложения — вычитания имеют место в том или ином виде при выводе всех схем.

Вследствие этих особенностей схем для гарантии определенной степени точности в алгоритме Винограда необходимо, вероятно, больше битов на слово, чем в алгоритме Кули—Тьюки. Мы не анализировали здесь эту проблему точности, но надо отметить, что структура алгоритма в том виде, в котором она представлена на рис. 4.20, делает такой анализ относительно простым.

В заключение упомянем еще один важный аспект алгоритма, представленного на рис. 4.20, а именно, удобство его структуры для реализации различных схем параллельной обработки и конвейерной организации. Действительно, он создает благодатную основу для всевозможных проектных решений и различных вариантов реализации. По мере расширения известности алгоритма, несомненно, будет материализовано все больше и больше таких вариантов.

Признательность. Представленная здесь работа выполнена в Лаборатории реактивного движения (ЛРД) Калифорнийского технологического института, по контракту NASA № NAS7-100. Автор хотел бы также выразить благодарность д-ру Л. Д. Баумерту, бывшему сотруднику ЛРД, за полезные комментарии, относящиеся к некоторым теоретико-числовым аспектам этой работы.

Приложение. Конгруентность полиномов.

Выкладки в разд. 4.3 требуют численного решения уравнения

$$R(x) = S_n(x) \bmod m(x), \tag{П.1}$$

где

$$S_n(x) = \sum_{k=0}^{n} s_k x^k, \tag{П.2}$$

а $m(x)$ — моический полином степени 1 или 2, корни которого лежат на единичной окружности. Здесь мы получаем все необходимые для этого результаты.

1) $m(x) = x - x_0 (x_0 = \pm 1)$; $R(x)$ должен иметь степень нуль

$$R(x) = r_0 \tag{П.3}$$

и (П.1) в этом случае эквивалентно

$$S_n(x) = (x - x_0)\, Q_{n-1}(x) + r_0, \qquad (П.4)$$

где Q_{n-1} является полиномом степени $(n-1)$. Подставляя $x = x_0$ в (П.4), получаем

$$R(x) = r_0 = S_n(x_0). \qquad (П.5)$$

Заметим, что при $x_0 = \pm 1$ r_0 является алгебраической суммой коэффициентов $S_n(x)$, не содержащей умножений.

2) $m(x) = x^2 - 2x\cos\theta + 1 = (x - x_0)(x - \bar{x}_0)$; $R(x)$ должен быть полиномом первой степени

$$R(x) = r_0 + r_1 x \qquad (П.6)$$

и (П.1) эквивалентно

$$S_n(x) = (x - x_0)(x - \bar{x}_0)\, Q_{n-2}(x) + (r_0 + r_1 x). \qquad (П.7)$$

Отсюда

$$S_n(x_0) = r_0 + r_1 x_0, \qquad (П.8)$$

$$S_n(\bar{x}_0) = r_0 + r_1 \bar{x}_0. \qquad (П.9)$$

Заметим, что

$$x_0 = \cos\theta + i\sin\theta = e^{i\theta}. \qquad (П.10)$$

Следовательно, вычитая (П.9) из (П.8), получаем

$$2i r_1 \sin\theta = S_n(x_0) - S_n(\bar{x}_0) = \sum_{k=0}^{n} s_k \left(e^{i k\theta} - e^{-i k\theta}\right),$$

$$\therefore r_1 = \frac{1}{\sin\theta} \sum_{k=0}^{n} s_k \sin k\theta,$$

$$r_1 = \sum_{k=1}^{n} S_k\, U_{k-1}(\cos\theta), \qquad (П.11)$$

где $U_m(x)$ — полином Чебышева второго рода.

Для получения r_0 умножаем (П.8) на $\bar{x}_0$ и (П.9) на x_0, после чего находим их разность

$$2i r_0 \sin\theta = x_0 S_n(\bar{x}_0) - \bar{x}_0 S_n(x_0) = \sum_{k=1}^{n} s_k \left[e^{-i(k-1)\theta} - e^{i(k-1)\theta}\right] =$$

$$= 2i s_0 \sin\theta - 2i \sum_{k=2}^{n} s_k \sin(k-1)\theta,$$

$$\therefore r_0 = s_0 - \sum_{k=2}^{n} s_k\, U_{k-2}(\cos\theta). \qquad (П.12)$$

В табл. П.1 перечислены конкретные полиномы $m(x)$ и соответствующие значения $R(x)$, для которых требуется θ. Заметим, что для этих значений θ $U_k(\cos\theta)$ принимают только значения 0, ± 1. Следовательно, и r_0, и r_1 являются алгебраической суммой коэффициентов $S_n(x)$, не содержащей умножений. Результаты для всех требующихся степеней $S_n(x)$ сведены в табл. П.1.

Таблица П.1

$S_n(x) \bmod m(x)$ для $S_n(x) = \sum_{k=0}^{n} s_k x^k$

$m(x)$	θ^o	$S_2(x) \bmod m(x)$	$S_3(x) \bmod m(x)$	$S_5(x) \bmod m(x)$
x^2-x+1	60	$(s_1+s_2)x+(s_0-s_2)$	–	$(s_1+s_2-s_4-s_5)x+(s_0-s_2-s_3+s_5)$
x^2+1	90	$s_1 x+(s_0-s_2)$	$(s_1-s_3)x+(s_0-s_2)$	–
x^2+x+1	120	$(s_1-s_2)x+(s_0-s_2)$	$(s_1-s_2)x+(s_0-s_2+s_3)$	$(s_1-s_2+s_4-s_5)x+(s_0-s_2+s_3-s_5)$

Конкретное применение табл. П.1 заключается в следующем. Дано

$$P(x) \bmod m(x) = p_1 x + p_0, \qquad (П.13)$$

$$Q(x) \bmod m(x) = q_1 x + q_0. \qquad (П.14)$$

Найти

$$G(x) = g_1 x + g_0 = [P(x)\,Q(x)] \bmod m(x). \qquad (П.15)$$

$G(x)$ может быть выражена с помощью (П.13), (П.14) следующим образом:

$$G(x) = \{[P(x) \bmod m(x)]\,[Q(x) \bmod m(x)]\} \bmod m(x) =$$
$$= [(p_1 q_1)x^2 + (p_1 q_0 + p_0 q_1)x + p_0 q_0] \bmod m(x). \qquad (П.16)$$

Отождествляя заключенный в скобки полином с $S_2(x)$ в табл. П.1, получаем:

$$G(x) = [p_1(q_0+q_1)+p_0 q_1]x + (p_0 q_0 - p_1 q_1) \quad [m(x) = x^2-x+1], \qquad (П.17)$$

$$G(x) = (p_1 q_0 + p_0 q_1)x + (p_0 q_0 - p_1 q_1) \quad [m(x) = x^2+1], \qquad (П.18)$$

$$G(x) = [p_1(q_0-q_1)+p_0 q_1]x + (p_0 q_0 - p_1 q_1) \quad [m(x) = x^2+x+1]. \qquad (П.19)$$

Начиная с p_i, q_i, каждая из этих формул требует 4 умножения. Однако при соответствующем переупорядочении членов можно заменить одно из этих умножений дополнительным сложением, тем самым ускорив вычисления. Завершается это добавлением и вычитанием $p_0 q_0$ из члена g_1. Этого достаточно для (П.17), (П.19). В (П.18) мы также модифицируем член g_0, добавляя и вычитая $p_0 q_1$. Результаты сведены в табл. П.2, и мы видим, что трех умножений теперь достаточно. Однако случай (x^2+1), по-видимому, показывает, что затраты оказываются выше, чем утверждалось ранее, а именно, они составляют 3 дополнительных сложения. В общем случае это, несомненно, верно. Но мы намереваемся применять эти результаты в ситуации, когда арифметические операции, включающие только p_0, p_1, не принимаются в расчет. (Соответствующие констан-

Таблица П.2

$[P(x)Q(x)] \bmod m(x)$ для $P(x) \bmod m(x) = p_1 x + p_0$
$Q(x) \bmod m(x) = q_1 x + q_0$

$m(x)$	$[P(x)Q(x)] \bmod m(x)$
x^2-x+1	$[(p_1+p_0)(q_1+q_0)-p_0 q_0]x+(p_0 q_0 - p_1 q_1)$
x^2+1	$[(p_1-p_0)q_0+(q_1+q_0)p_0]x+(q_1+q_0)p_0-(p_1+p_0)q_1$
x^2+x+1	$[(p_1-p_0)(q_0-q_1)+p_0 q_0]x+(p_0 q_0 - p_1 q_1)$

ты вычисляются заранее.) При этих условиях затраты составляют одно дополнительное сложение.

Надо заметить, что более высокая скорость, достигаемая за счет применения формул из табл. П.2, сопровождается необходимостью иметь больше разрядов в слове. В арифметике с фиксированной запятой эти дополнительные разряды должны предотвращать переполнение на промежуточных этапах. В арифметике с плавающей запятой эти разряды нужны, чтобы предотвратить потерю точности. Рассмотрим крайний случай, когда $p_0 q_0 \gg g_1$. На выражение (П.17) это не влияет, но в арифметике с плавающей запятой (табл. П.2) может дать значение g_1, которое будет чисто случайным.

ГЛАВА 5

МЕДИАННАЯ ФИЛЬТРАЦИЯ: СТАТИСТИЧЕСКИЕ СВОЙСТВА

(Б. И. Юстуссон) [1]

Медианная фильтрация является методом нелинейной обработки сигналов, который может быть полезен при подавлении шумов. Она была предложена в качестве инструмента анализа временных рядов Тьюки [5.1], в 1971 г. и позже ее стали применять также при обработке изображений. Медианная фильтрация осуществляется посредством движения некоторой апертуры вдоль дискретизированного изображения (последовательности) и замены значения элемента изображения в центре апертуры медианой исходных значений отсчетов внутри апертуры. При этом обычно получается более гладкое, по сравнению с исходным, результирующее изображение (последовательность отсчетов).

Классическая процедура сглаживания состоит в использовании линейной фильтрации нижних частот и во многих случаях является наиболее приемлемой процедурой. Тем не менее в определенных ситуациях медианная фильтрация предпочтительней. Она имеет следующие основные преимущества: 1) медианная фильтрация сохраняет резкие перепады, тогда как линейная низкочастотная фильтрация смазывает такие перепады; 2) медианные фильтры очень эффективны при сглаживании импульсного шума. Эти свойства пояснены на рис. 5.1.

Основная цель главы — представить различные теоретические результаты, касающиеся медианной фильтрации. Автор надеется, что эти результаты помогут составить правильное суждение о практической применимости медианных фильтров.

[1] Department of Mathematics, Royal Institute of Technology S-100 44 Stockholm 70, Sweden.

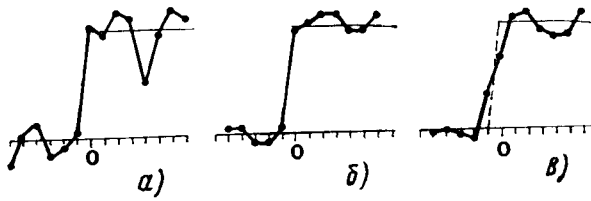


Рис. 5.1. Последовательности типа граница плюс шум (*а*) после медианной фильтрации (*б*), после фильтрации с помощью скользящего среднего (*в*); $n=3$

Основные определения, касающиеся медианных фильтров, даны в разд. 5.1. В разд. 5.2 исследуется способность медианных фильтров подавлять шум, а также приведены формулы, которые дают количественные представления о степени подавления шума. Рассматриваются белый, небелый, импульсный и точечный шумы. В разд. 5.3 сравнивается качество фильтрации посредством вычисления скользящего среднего и медианных фильтров на изображениях вида «перепад плюс шум». Влияние медианных фильтров на статистику второго порядка случайного шума обсуждается в разд. 5.4. Для входного сигнала с белым шумом даны точные результаты; для небелого шума с помощью предельных теорем получены приближенные результаты. Частотная характеристика рассматривается посредством оценки отклика фильтра на простую косинусоиду, а также на сигналы более общего вида. В разд. 5.5 представлены некоторые модификации медианных фильтров, которые также обладают свойством сохранения перепадов, но отличаются от простых медианных фильтров другими свойствами. Некоторые применения медиан и других порядковых статистик рассматриваются в разд. 5.6.

В заключение приведен небольшой обзор более ранних работ, касающихся медиан и медианной фильтрации.

Медианы давно использовались и изучались в статистике как альтернатива средним арифметическим значениям отсчетов в оценке выборочных средних значений популяций. Большинство исследований касались медиан и других порядковых статистик последовательностей независимых случайных величин (см. хорошо известные монографии [5.2, 5.3]). Однако медианы зависимых случайных величин также изучались в литературе (см. [5.4], где даны дополнительные ссылки).

Как упоминалось выше, скользящая оценка медианы была предложена Тьюки, который применил ее для сглаживания временны́х рядов, встречающихся в экономических исследованиях. Тьюки также рассматривал итеративную медианную фильтрацию и указывал, что она сохраняет во временны́х рядах большие резкие изменения их уровня (т. е. перепады). В [5.5] и [5.6] применена скользящая медиана при обработке речи для очистки высоких тонов от помех [5.7]. Разработан метод обработки сигналов для подчеркивания краев, в котором медианный фильтр предназначен для уничтожения ложных колебаний после линейной фильтрации.

Позже медианные фильтры были применены несколькими авторами в обработке изображений. В 1975 г. Прэтт исследовал эффективность медианной фильтрации изображений с нормальным белым и импульсным шумами, а также влияние различных форм апертуры фильтра. Его результаты были опубликованы в [5.8, разд. 12.6]. Медианные фильтры были использованы для коррекции шума сканирующих устройств [5.9].

5.1. Определение медианных фильтров

5.1.1. Одномерные медианные фильтры

Медианой последовательности $x_1, \ldots, x_n$, n — нечетное, является средний по значению член ряда, получающегося при упорядочении последовательности по возрастанию. Для четного n определим медиану как среднее арифметическое двух средних членов. В литературе можно найти другие определения, но поскольку они мало отличаются друг от друга и n в большинстве интересующих нас случаев нечетное, мы не будем возвращаться к этому в дальнейшем. Обозначим медиану следующим образом:

$$\text{медиана } (x_1, \ldots, x_n). \tag{5.1}$$

Например: медиана $(0, 3, 4, 0, 7) = 3$.

Медианный фильтр последовательности длиной $n\{x_i, i \in \mathbb{Z}\}$ для нечетных n определяется как

$$y_i = \text{медиана } x_i \triangleq \underset{n}{\text{медиана}} (x_{i-v}, \ldots, x_i, \ldots, x_{i+v}), \quad i \in \mathbb{Z}, \tag{5.2}$$

где $v = (n-1)/2$ и $\mathbb{Z}$ обозначает множество всех натуральных чисел. Будем использовать также термины *скользящая медиана* и *текущая медиана*.

Легко видеть, что медианный фильтр сохраняет перепады, тогда как соответствующая фильтрация путем вычисления скользящего среднего

$$z_i = (x_{i-v} + \ldots + x_i + \ldots + x_{i+v})/n, \; i \in \mathbb{Z}, \tag{5.3}$$

превращает перепад в пологий скат шириной n (см. гл. 6).

5.1.2. Двумерные медианные фильтры

Будем считать, что цифровые изображения представляются набором чисел на квадратной решетке $\{x_{i,j}\}$, где (i, j) изменяются по $\mathbb{Z}^2$ или некоторому подмножеству $\mathbb{Z}^2$.

Двумерный медианный фильтр с апертурой A для изображения $\{x_{ij}, (i, j) \in \mathbb{Z}^2\}$ определяется как

$$y_{ij} = \underset{A}{\text{медиана}}\, x_{ij} \triangleq \text{медиана}\,[x_{i+r,j+s}; \; (rs) \in A], \quad (i, j) \in \mathbb{Z}^2. \tag{5.4}$$

Можно использовать различные формы апертур A фильтра, например, линейные сегменты, квадраты, круги, кресты, квадратные рамки, кольца. Некоторые из них показаны на рис. 5.2. Форма «колец» на рис. 5.2,*е* была выбрана так, чтобы число точек в каж-

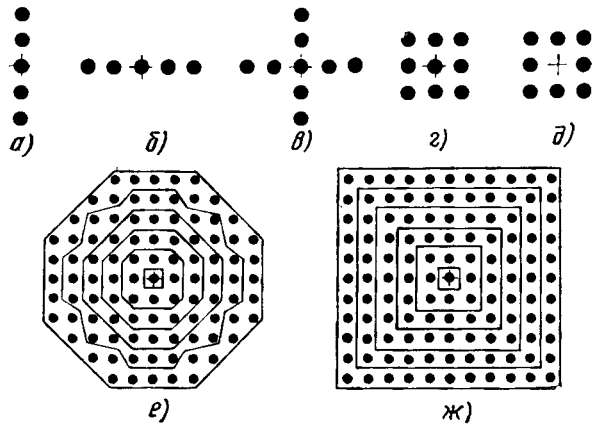


Рис. 5.2. Апертуры фильтра:
а, б — линейные сегменты, *в* — крест, *г* — квадрат, *д* — квадратная рамка, *е* — круги и кольца, *ж* — квадраты и квадратные рамки

дом кольце было приблизительно пропорционально площади соответствующего правильного кольца.

Приведенные определения медианных фильтров не объясняют способа нахождения выходного сигнала вблизи конечных и пограничных точек в конечных последовательностях и изображениях. Один из простых приемов, которые использовались в рассматриваемых ниже экспериментах, состоит в том, что нужно находить медиану только тех точек внутри изображения, которые попадают в пределы апертуры. Поэтому для точек, расположенных рядом с границами, медианы будут определены, исходя из меньшего, чем в A, числа точек.

5.1.3. Сохранение перепадов

Под *изображением перепада* понимаем изображение, в котором точки по одну сторону от некоторой линии имеют одинаковое значение a, а все точки по другую сторону от этой линии — значение b, $b \neq a$.

Следующие результаты представляют фундаментальное свойство медианных фильтров. Если апертура A симметрична относительно начала координат и содержит его в себе, т. е. если

$$(r, s) \in A \Rightarrow (-r, -s) \in A, \quad (0, 0) \in A, \qquad (5.5), \ (5.6)$$

то тогда медианный фильтр (5.4) сохраняет любое изображение перепада. Подробное обсуждение эффектов медианной фильтрации других детерминированных сигналов, отличных от простого перепада, читатель может найти в гл. 6. Условия (5.5), (5.6) выполняются для всех апертур рис. 5.2, кроме квадратной рамки и колец, которые не содержат начала координат. Тем не менее квадратные рамки и кольца будут лишь незначительно изменять перепад. Точ-

но так же будут вести себя и фильтры с другой формой апертуры, для которых выполняются условия (5.5), (5.6). Эти условия означают, что число точек n в A является четным.

5.2. Подавление шумов с помощью медианной фильтрации

Как утверждалось выше, медианные фильтры могут использоваться для подавления шумов. Прэтт [5.8] на качественном уровне рассмотрел их действие на белый и импульсный шумы. Здесь приведем некоторые соотношения для дисперсии, которые в количественной форме оценивают степень подавления шума.

Медианные фильтры нелинейны, и это усложняет математический анализ их характеристик. Нельзя разграничить влияние этих фильтров на сигнал и шум, что для линейных фильтров делается очень просто. Ограничимся рассмотрением простейшего случая постоянного сигнала.

5.2.1. Белый шум

Модель белого шума. Значения элементов изображения $\{x_{ij}\}$ или последовательность чисел $\{x_i\}$ являются независимыми одинаково распределенными (НОР) случайными величинами со средним значением m

$$x = m + z, \tag{5.7}$$

где $E(z) = 0$ и, следовательно, $E(x) = m$.

Пусть $F(x)$ и $f(x) = F'(x)$ обозначают функции распределения и плотности вероятностей величин x. Теперь запишем два известных результата из теории вероятностей, касающихся медиан независимых, одинаково распределенных случайных величин (см. [5.3], гл. 2.9).

Плотность распределения y = медиана $(x_1, \ldots, x_n)$ для нечетных n

$$g(y) = n \binom{n-1}{(n-1)/2} f(y) F(y)^{(n-1)/2} [1 - F(y)]^{(n-1)/2}. \tag{5.8}$$

Распределение y = медиана $(x_1, \ldots, x_n)$ для больших n является приблизительно нормальным $N(\tilde{m}, \sigma_n)$, где $\tilde{m}$ — теоретическая медиана, определяемая из условия $F(\tilde{m}) = 0{,}5$ и

$$\sigma_n^2 = \frac{1}{n\, 4 f^2(\tilde{m})} = \mathrm{Var}\ [\text{медиана } (x_1, \ldots, x_n)]. \tag{5.9}$$

При малых n обычно можно получить лучшее приближение для дисперсии заменой члена $1/n$ в (5.9) членом $1/(n+b)$, где

$$b = 1/[4 f^2(\tilde{m})\, \sigma_x^2] - 1.$$

Эта модификация получена вследствие выбора b таким, что при $n = 1$ формула (5.9) становится точной.

Приведенные результаты справедливы как для одномерной, так и двумерной фильтрации, если n выбирать равным числу точек в апертуре фильтра. Если $f(x)$ симметрична относительно m, то (5.8) также будет симметрична относительно m и, таким образом, справедлива следующая простая формула:

$$E\,[\text{медиана}\ (x_1, \ldots, x_n)] = E(x_i) = m. \tag{5.10}$$

Пример 5.1. Равномерное распределение. Если случайные величины x являются НОР и равномерно распределены на отрезке $[0, 1]$, то можно найти точное значение дисперсии медианы, используя (5.8):

$$\mathrm{Var}\,[\text{медиана}\ (x_1, \ldots, x_n)] = \frac{1}{(n+2)\,4} = \frac{\sigma_x^2}{n+2} \cdot 3. \tag{5.11}$$

Формула (5.9) после небольшой модификации дает тот же результат.

Пример 5.2. Нормальное распределение. Если случайные величины x являются независимыми, одинаково распределенными с нормальным распределением $N(m, \sigma)$, то $\tilde{m} = m$ и дисперсию можно найти только при помощи численного интегрирования, используя (5.8). Значения дисперсии медианы случайных величин $N(0, 1)$ сведены в табл. 5.4 в строках $(n, m, k) = (n, n, n)$. Формула (5.9), модифицированная для малых n, дает:

$$\mathrm{Var}\,[\text{медиана}\ (x_1, \ldots, x_n)] \approx \frac{\sigma^2}{n + \pi/2 - 1}\,\frac{\pi}{2}, \quad n = 1, 3, 5, \ldots \tag{5.12}$$

Эта формула обеспечивает хорошую точность для всех нечетных n.

Среднее значение $\bar{x}$ для n НОР случайных величин имеет дисперсию σ^2/n. Равенство (5.12) показывает также, что для нормального белого шума дисперсия медианы приблизительно на $(\pi/2-1) = 57\%$ больше, чем дисперсия для среднего. Следовательно, скользящее усреднение подавляет нормальный белый шум несколько лучше, чем медианный фильтр с такой же апертурой. Иначе говоря, чтобы медианный фильтр давал ту же дисперсию шума, что и скользящее усреднение, в его апертуре должно быть на 57% больше точек. Результаты медианной фильтрации и фильтрации скользящим усреднением с апертурой 3×3 показаны на рис. 5.3. Каждое изображение состоит из 45×30 элементов, размеры каждого элемента 1×1 мм; a_1 — исходное тестовое изображение; b_1, v_1, g_1 были получены изменением шкалы значений a_1 и добавлением нормального белого шума со значениями стандартного отклонения $h/5$, $h/3$, h, где h — наибольшая высота перепада (рис. 5.3 будет рассмотрен также в разд. 5.3).

Пример 5.3. Двойное экспоненциальное распределение. Пусть случайные величины x имеют двойное экспоненциальное распределение со средним m и дисперсией σ^2, т. е. имеют плотность распределения

$$f(x) = \frac{\sqrt{2}}{\sigma} \exp\left(-\sqrt{2}\,|x-m|/\sigma\right), \quad x \in R. \tag{5.13}$$

Тогда согласно (5.9) асимптотическая дисперсия медианы $(x_1, \ldots, x_n)$

$$\sigma_n^2 = \frac{\sigma^2}{(n-1/2)}\,\frac{1}{2} \approx \mathrm{Var}\ (\text{медиана}), \tag{5.14}$$

что на 50% меньше, чем дисперсия σ^2/n среднего арифметического $\bar{x}$. Таким образом, для этого типа шума медиана является лучшей оценкой m, чем среднее

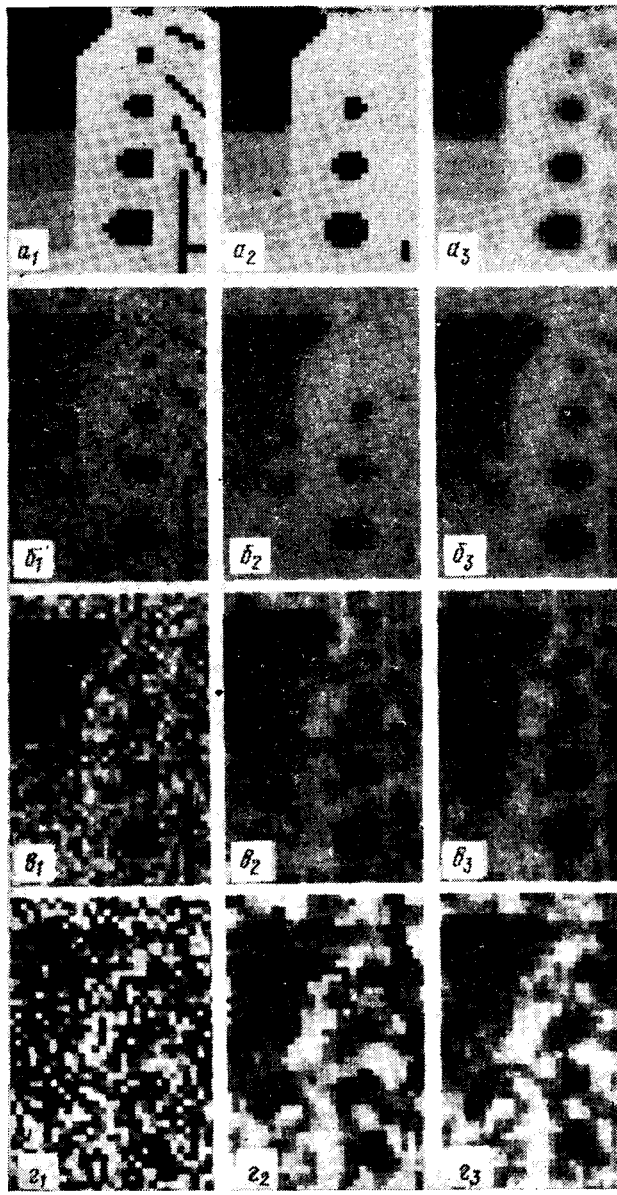


Рис. 5.3. Фильтрация изображений с нормальным шумом $N(0, \sigma)$:
a_1—$г_1$ — изображения на входе при $\sigma = 0,\ h/5,\ h/3,\ h$;
a_2—$г_2$ — изображения, подвергнутые медианной фильтрации; a_3—$г_3$ — изображения, подвергнутые фильтрации с помощью скользящего усреднения

арифметическое $\bar{x}$. Медиана является наиболее правдоподобной оценкой и, следовательно, оптимальной оценкой m по критерию минимума среднеквадратичной погрешности (по крайней мере асимптотически). То, что медиана является здесь наиболее правдоподобной оценкой, с очевидностью вытекает из общего результата, показывающего, что медиана представляет собой наилучшую по минимуму абсолютного отклонения оценку центра распределения, т. е.

$$\min_{a} \sum_{i=1}^{n} |x_i - a| \tag{5.15}$$

достигается для a = медиана $(x_1, \ldots, x_n)$. Из полученных результатов вытекает более общий вывод о том, что медиана лучше, чем арифметическое среднее, служит для подавления шумов с распределениями с тяжелыми хвостами. Предельным случаем шума с таким распределением является импульсный шум, который будет рассмотрен в подразд. 5.2.3.

5.2.2. Небелый шум

Для входных последовательностей (изображений), которые являются случайными процессами (случайными полями) общего вида, т. е. полями с не независимыми значениями отсчетов, нельзя получить простые точные формулы для распределения медиан. Тем не менее существуют предельные теоремы, аналогичные (5.9) (см. [5.4, 5.10], где можно найти также дополнительные ссылки на литературу). Условия, необходимые для предельных теорем, состоят в том, что процессы $\{x_i\}$, $\{x_{ij}\}$ стационарны и перемешаны. Согласно условиям перемешивания отсчеты процесса, расположенные далеко друг от друга, должны быть практически независимы (подробности см. в [5.4, 5.10]). Для стационарного перемешанного нормального процесса с ковариационной функцией

$$\mathrm{Cov}\,(x_i, x_{i+\tau}) = r_x(\tau) = \sigma_x^2 \rho_x(\tau), \tau = 0, \pm 1, \ldots, \tag{5.16}$$

имеем приближенное выражение для дисперсии медианы

$$\mathrm{Var}\,[\text{медиана}\,(x_1, \ldots, x_n)] \approx$$

$$\approx \frac{\sigma_x^2}{n + \pi/2 - 1} \sum_{j=-(n-1)}^{n-1} \left(1 - \frac{|j|}{n}\right) \arcsin[\rho_x(j)]. \tag{5.17}$$

Для случая двумерной фильтрации получаем аналогичный результат. В разд. 5.4 эти виды приближений и предельные теоремы будут рассмотрены дополнительно.

Интересно сравнить (5.17) с дисперсией арифметического среднего $\bar{x} = (\Sigma x_i)/n$ n случайных величин:

$$\mathrm{Var}\,(\bar{x}) = \frac{\sigma_x^2}{n} \sum_{j=-(n-1)}^{n-1} \left(1 - \frac{|j|}{n}\right) \rho_x(j). \tag{5.18}$$

Сходство (5.17) и (5.18) бросается в глаза. Для нормальных процессов с неотрицательными значениями корреляции

$$\rho_x(\tau) \geqslant 0, \tau = 0, \pm 1, \ldots, \tag{5.19}$$

Таблица 5.1

Относительные значения дисперсии для нормальных авторегрессионных процессов AR(1)

a / Var (медиана) / Var $(\bar{x})$	0,9	0,5	0,0	—0,5	—0,9
	1,10	1,21	1,57	2,59	6,59

получаем при больших n, используя (5.17), (5.18) и тот факт, что $\rho_x(\tau) \leqslant \arcsin \rho_x(\tau) \leqslant \rho_x(\tau)\pi/2$:

$$1 \leqslant \text{Var (медиана)}/\text{Var}(\bar{x}) \leqslant \pi/2. \qquad (5.20)$$

(Этот результат справедлив также для двумерной фильтрации.) Таким образом, для нормальных процессов с неотрицательными значениями корреляции дисперсия медианы почти на 57% больше дисперсии арифметического среднего. Для процессов с отрицательными и положительными значениями корреляции значения отношений дисперсий (5.20) могут быть намного больше $\pi/2$. Это иллюстрируется в табл. 5.1, где представлены значения отношений дисперсий для нормальных авторегрессионных процессов AR (1) первого порядка с

$$\rho_x(\tau) = a^{|\tau|}, \tau = 0, \pm 1, \ldots \qquad (5.21)$$

В работе [5.10] сообщается о результатах, полученных путем моделирования на ЭВМ нормальных процессов AR(1), которые показывают, что значения отношений, приведенные в табл. 5.1, приблизительно верны и для малых n, кроме случая $a = -0{,}9$. Для $a = -0{,}9$ и $n = 9$ значение этого отношения оказалось равным 14,9.

5.2.3. Импульсный и точечный шумы

Под *импульсным шумом* понимаем искажение сигнала импульсами, т. е. выбросами с очень большими положительными или отрицательными значениями и малой длительностью. Медианная фильтрация хорошо приспособлена для подавления такого шума [5.5, 5.8] при условии, что размер апертуры фильтра должен быть выбран по крайней мере в два раза больше ширины импульса. В этом случае импульсы шума, которые достаточно удалены друг от друга, будут полностью убраны медианным фильтром. Однако импульсы, расположенные близко друг к другу, могут сохраняться.

При обработке изображений импульсный шум возникает, например, вследствие ошибок декодирования, которые приводят к появлению черных и белых точек на изображении. Поэтому его часто называют точечным шумом[1]. Выбросы шума особенно заметны на очень темных или очень светлых участках изображений. Для та-

[1] В отечественной литературе применяют также термин *импульсный шум*. — *Прим. ред.*

ких участков можно вывести несколько несложных формул для вероятности правильного воспроизведения. Рассмотрим две модели. В первой модели все выбросы шума имеют одинаковое значение, во второй шум принимает значения, выбранные случайно из всего диапазона от черного до белого.

Импульсный шум. Модель 1. Появление выброса шума в каждой точке (i, j) изображения имеет вероятность p и не зависит ни от наличия шума в других точках изображения, ни от исходного изображения. Искаженная точка приобретает стабильное значение d (т. е. значение черного). Пусть $\{x_{ij}\}$ — искаженное изображение. Тогда

$$x_{ij} = \begin{cases} d & \text{с вероятностью } p, \\ s_{ij} & \text{с вероятностью } (1-p), \end{cases} \tag{5.22}$$

где s_{ij} — значения неискаженного изображения.

Предположим теперь, что точка (i', j') расположена на участке с постоянным значением $\{s_{ij}\}$ исходного изображения [по крайней мере в окрестности A с центром в (i', j')], т. е.

$$s_{i'+r,\, j'+s} = s_{i'j'} = c \neq d, (r, s) \in A. \tag{5.23}$$

Применим к $\{x_{ij}\}$ медианный фильтр с апертурой A

$$y_{ij} = \text{медиана } (x_{ij}). \tag{5.24}$$

Тогда значение выходной величины $y_{i'j'}$ будет верным, т. е. $y_{i'j'} = s_{i'j'} = c$, в том и только в том случае, если число выбросов шума в пределах апертуры A с центром в (i', j') меньше половины числа точек в A, т. е. меньше или равно $(n-1)/2$, где n — размер апертуры A. Из того, что число искаженных точек в апертуре имеет биномиальное распределение, вытекает следующий результат:

$$P[\text{правильного воспроизведения в } (i', j')] = P(y_{i'j'} = s_{i'j'}) = \sum_{k=0}^{(n-1)/2} \binom{n}{k} p^k (1-p)^{n-k} \triangleq Q(n, p). \tag{5.25}$$

Значения $1-Q(n, p)$ для различных значений n и p приведены в табл. 5.2. Видно, что если вероятность ошибки p не очень велика

Таблица 5.2

Вероятность ошибки при фильтрации импульсного шума, $1-Q(n, p)$

Вероятность ошибки p	Размер апертуры n				
	3	5	9	25	49
0,01	0,00030	0,0000099	0,0000000	0	0
0,05	0,00725	0,00116	0,000033	0,0000000	0
0,1	0,028	0,0086	0,00089	0,0000002	0
0,15	0,0608	0,0266	0,00563	0,000017	0
0,2	0,104	0,058	0,0196	0,00037	0,000013
0,3	0,216	0,163	0,099	0,017	0,00165
0,4	0,352	0,317	0,267	0,154	0,0776
0,5	0,500	0,500	0,500	0,500	0,500

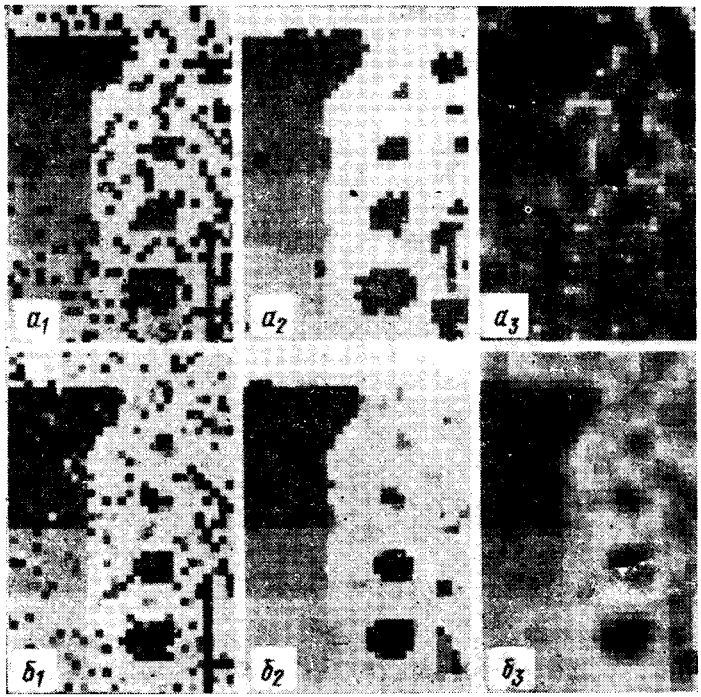


Рис. 5.4. Фильтрация изображений с импульсным шумом: a_1, $б_1$ — изображения на входе с шумами модели 1 и модели 2 при вероятности ошибки $p=0.2$; a_2, $б_2$ — изображения, подвергнутые медианной фильтрации; a_3, $б_3$ — изображения, подвергнутые фильтрации с помощью скользящего усреднения

(скажем, не более 0,3), то медианная фильтрация с достаточно малой апертурой значительно снизит число ошибок. Фильтр с большой апертурой подавит шум в еще большей степени, но он также исказит и сигнал. Результаты медианной фильтрации импульсного шума иллюстрируются на рис. 5.4*а*.

Импульсный шум. Модель 2. Эта модель отличается от модели 1 только тем, что искаженные точки приобретают случайные, а не фиксированные, значения z_{ij}. Предполагается, что они являются независимыми случайными величинами с равномерным распределением на непрерывном интервале $[0, d]$. Итак,

$$x_{ij} = \begin{cases} z_{ij} \text{ с вероятностью } p, \\ s_{ij} \text{ с вероятностью } (1-p). \end{cases} \tag{5.26}$$

Для получения простой формулы предположим, что неискаженное изображение является полностью белым (или полностью черным) в окрестностях (i', j'), т. е. в (5.23) $c=0$ (или $c=d$). Это по сути наиболее сложный случай для медианного фильтра, так как все ошибочные значения попадают по одну и ту же сторону от пра-

вильного значения. Вероятность правильного воспроизведения совпадает с $Q(n, p)$ в (5.25), но кроме того, значения неисправленных ошибок уменьшаются. Математическое ожидание выходных величин и оставшихся выбросов шума определяется формулами:

$$E[\text{медиана } (x_{i'j'})] = \\ = d \sum_{k=(n+1)/2}^{n} \frac{k-(n-1)/2}{k+1} \binom{n}{k} p^k (1-p)^{n-k}, \tag{5.27}$$

$$E[\text{медиана } (x_{i'j'}) \mid \text{ошибочное воспроизведение в точке } (i', j')] = \\ = E[\text{медиана } (x_{i', j'})] / [1 - Q(n, p)]. \tag{5.28}$$

Доказательство. Пусть $N_{i'j'}$ — число ошибок в апертуре A с центром в (i', j'), распределенное по биномиальному закону. Условное распределение медианы в (i, j) при $N_{i'j'}$, равном k, такое же, как и распределение медианы $(0, \ldots, 0, z_1, \ldots, z_k)$, где число нулей равняется $(n-k)$, а $z_1, \ldots, z_k$ — независимые и равномерно распределенные на интервале $(0, d)$ числа. Далее имеем

$$\text{медиана } (0, \ldots, 0, z_1, \ldots, z_k) = \\ = \begin{cases} 0, & \text{если } k \leqslant (n-1)/2, \\ z_{(r, k)}, & \text{если } k \geqslant (n+1)/2, \end{cases} \tag{5.29}$$

где $z_{(r,k)}$ означает r-ю порядковую статистику $z_1, \ldots, z_k$ и $r = k - (n-1)/2$. Окончательно имеем

$$E[\underset{A}{\text{медиана}} (x_{i'j'})] = \\ = \sum_{k=0}^{n} E[\text{медиана } (x_{i'j'}) \mid N_{i'j'} = k] P(N_{i'j'} = k) = \\ = \sum_{k=(n+1)/2}^{n} E\{z_{[k-(n-1)/2, k]}\} \binom{n}{k} p^k (1-p)^{n-k} = \\ = \sum_{k=(n+1)/2}^{n} d \frac{k-(n-1)/2}{k+1} \binom{n}{k} p^k (1-p)^{n-k}. \tag{5.30}$$

На последнем шаге мы использовали значение математического ожидания порядковых статистик равномерного распределения (см., например, [5.3, гл. 3]).

Случай $n=9$ и $p=0{,}2$ проиллюстрирован на рис. 5.4*б*. Согласно приведенным формулам доля искаженных точек должна уменьшиться с $p=0{,}2$ до $1-Q(n, p) = 0{,}0196$ и математическое ожидание ошибки — с $E(z_{ij}) = 0{,}5\,d$ до [согласно (5.28)]

$$\frac{0{,}00366}{0{,}0196} d = 0{,}187\, d\,. \tag{5.31}$$

Результат фильтрации хорошо согласуется с этими оценками. Как видно из рис. 5.4, скользящее усреднение плохо приспособлено к фильтрации импульсного и точечного шумов. Некоторые фильтры, предназначенные для подавления точечного шума, были

предложены в [5.11]. Мы не сравнивали эти фильтры с медианным фильтром[1].

Одним из шумов, похожих на импульсный, является шум пропадания строк, который появляется при пропадании или искажении целых строк в процессе сканирования изображений [5.9]. Медианный фильтр с прямоугольной апертурой $n \times m$ (m точек на n строках) будет, как и в случае импульсного шума, уменьшать число ошибок. Предположим, что искажение строк происходит независимо для разных строк с вероятностью p и что все элементы искаженной строки приобретают одинаковое значение d, а $s_{ij} = c$. Тогда вероятность правильного воспроизведения будет равна $Q(n, p)$. Заметим, что здесь n означает число строк в прямоугольной апертуре. С точки зрения вычислений простейшей апертурой является апертура с $m = 1$, но иногда могут оказаться более выгодными бо́льшие значения m.

5.3. Перепад плюс шум

Мы уже видели, что медианные фильтры сохраняют перепады (в отсутствие шума), тогда как скользящее усреднение смазывает такие перепады. Кроме того, для случая нормального белого шума (на постоянном фоне) скользящее усреднение уменьшает такой шум несколько эффективнее, чем медианные фильтры с тем же размером апертуры. В этом разделе рассмотрим фильтрацию перепадов при наличии аддитивного белого шума, т. е. фильтрацию последовательностей, или изображений, с

$$x = s + z, \tag{5.32}$$

где s — детерминированный сигнал, равный 0 по одну сторону от перепада и h — по другую, а z — случайные значения белого шума. В подразд. 5.3.1 сравнивается действие на такие последовательности (изображения) медианной фильтрации и скользящего усреднения. В подразд. 5.3.2 содержится математический вывод распределения порядковых статистик таких последовательностей. Этот вывод мы включили по той причине, что он является примером рассуждений, которые можно использовать при выводе других результатов, касающихся порядковых статистик независимых случайных величин, как, например, (5.8) и результаты в подразд. 5.4.1.

5.3.1. Сравнение медианной фильтрации и скользящего усреднения

Предположим, что случайные значения шума z распределены по нормальному закону $N(0, \sigma)$. Для начала рассмотрим одномерную фильтрацию и будем считать, что перепад происходит в

[1] Эффективные итеративный и рекурсивный алгоритмы фильтрации импульсного шума описаны также в [7]. Сравнение этих алгоритмов с алгоритмом медианной фильтрации при квадратной апертуре 3×3 показывает, что при вероятности ошибки до 0,4 алгоритм медианной фильтрации значительно уступает итеративному алгоритму, примерно эквивалентен рекурсивному алгоритму по значению среднеквадратической погрешности фильтрации и намного уступает обоим алгоритмам по критерию вероятности ложного исправления. — *Прим. ред.*

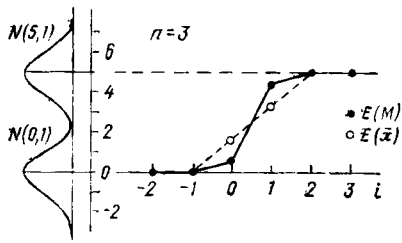


Рис. 5.5 Граница плюс шум. Математические ожидания для скользящей медианы (M), скользящего среднего (x) при $n=3$, $h=5$, $\sigma=1$

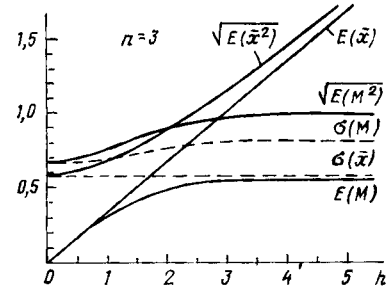


Рис. 5.6. Граница плюс шум. Моменты для скользящей медианы (M) и скользящего среднего (x) при $n=3$, $h=5$, $\sigma=1$

точке $i=1$ (рис. 5.5). Таким образом, для $i\leqslant 0$ величина x_i есть $N(0, \sigma)$, а для $i\geqslant 1$ величина x_i есть $N(h, \sigma)$. Плотность вероятностей $g(x)$ n-точечной медианы с k величинами x_i из $N(h, \sigma)$ и $(n-k)$ величинами x_i из $N(0, \sigma)$ может быть получена из (5.37)—(5.39), приведенных в следующем разделе. Хотя формула для $g(x)$ достаточно сложна, путем численного интегрирования можно очень просто найти средние значения и стандартные отклонения. Это было сделано для $\sigma=1$ и некоторых значений n и k. Результаты представлены в табл. 5.3. Значения для $\sigma\neq 1$ можно получить путем соответствующего изменения масштаба, а для $k\geqslant (n+1)/2$ — зная симметричные значения аргументов.

Распределение скользящего среднего, как легко понять, является $N(hk/n, \sigma/\sqrt{n})$, где k — число точек в пределах апертуры, имеющих значение $s=h$.

Таблица 5.3

Среднее значение и стандартное отклонение медиан на последовательностях типа граница плюс шум: z_1+h, ..., z_k+h, z_{k+1}, ..., z_n, где z_i — независимые одинаково распределенные величины $N(0, 1)$

n	k	h					
		0,0	1,0	2,0	3,0	4,0	5,0
3	1 E(медиана)	0,000	0,305	0,486	0,549	0,562	0,564
	σ(медиана)	0,670	0,697	0,760	0,806	0,822	0,826
9	3 E(медиана)	0,000	0,318	0,540	0,626	0,641	0,642
	σ(медиана)	0,408	0,424	0,471	0,513	0,527	0,529
5	1 E(медиана)	0,000	0,179	0,270	0,294	0,297	0,297
	σ(медиана)	0,536	0,551	0,580	0,596	0,600	0,600
5	2 E(медиана)	0,000	0,386	0,676	0,808	0,841	0,846
	σ(медиана)	0,536	0,560	0,631	0,705	0,740	0,748
25	5 E(медиана)	0,000	0,184	0,286	0,312	0,315	0,315
	σ(медиана)	0,248	0,256	0,271	0,280	0,282	0,282
25	10 E(медиана)	0,000	0,391	0,719	0,900	0,944	0,948
	σ(медиана)	0,248	0,260	0,295	0,346	0,371	0,375

На рис. 5.5 показана последовательность значений математического ожидания медиан и скользящего среднего вблизи перепада высотой $h=5$ при $n=3$. Значения скользящего среднего следуют по наклонной линии, и это свидетельствует о существенном смазывании перепада. Поведение математического ожидания значений медианы также свидетельствует о некотором смазывании, хотя и гораздо меньше, чем для скользящего среднего.

Чтобы иметь возможность сравнить эффективность фильтров на последовательностях типа перепад плюс шум, нужны меры точности передачи перепада. Воспользуемся мерой среднеквадратичной ошибки (СКО), усредненной по N точкам вблизи перепада

$$\frac{1}{N}\sum_{i} E(y_i - s_i)^2, \tag{5.33}$$

где y_i — значения на выходе фильтра. Для случая, показанного на рис. 5.5, т. е. для $n=3$ и $N=2$, выражение (5.33) равно $E(y^2_0)$. На рис. 5.6 представлены графики значений $E(y^2_0)^{1/2}$ как функции перепада h для медианного фильтра и для скользящего усреднения. Видно, что при $h<2(h<2\sigma)$ СКО для скользящего среднего немного меньше, чем для медианы, а при $h>3$ $(h>3\sigma)$ СКО медианы значительно меньше, чем СКО среднего. Этот результат показывает, что скользящая медиана значительно лучше, чем скользящее среднее, для перепадов большой высоты $(h>3\sigma)$, а для перепадов меньшей высоты различие между двумя фильтрами очень незначительно. Очень похожие результаты получены для больших апертур, $n=5$, и для двумерной фильтрации с апертурами 3×3 и 5×5. Эти выводы также подтверждаются результатами фильтрации изображений на рис. 5.3, где h фиксировано и $\sigma=0$, $h/5$, $h/3$ и h. Мера точности, которую мы использовали, может характеризовать только резкость *поперек* перепада. Она ничего не говорит о гладкости фильтрованного изображения *вдоль* перепада. На рис. 5.3 показано, что скользящее усреднение дает сигналы гладкие вдоль перепада, тогда как при обработке с помощью медианных фильтров протяженные перепады оказываются слегка изрезанными.

Сделаем теперь несколько дополнительных замечаний относительно поведения медиан при изменении h. Мы предполагаем, что k — число переменных со средним h — меньше $(n+1)/2$. По рис. 5.6 и из табл. 5.3 видно, что стандартные отклонения увеличиваются с увеличением h и имеют асимптотически ограниченные значения. Математические ожидания медианы для малых h близки к математическим ожиданиям для соответствующих средних

$$E(\text{медиана}) \approx E(\bar{x}) = h\,\frac{k}{n}, \tag{5.34}$$

но для больших h они асимптотически ограничены и, таким образом, ведут себя совершенно иначе, чем математические ожидания средних. Объясняется это тем, что при больших h (скажем, $h>4$) переменные x со средним значением 0 (скажем, здесь x_1, ..., x_{n-k}

будут резко отделены от переменных x со средним h (скажем, здесь $x_{n-k+1}, \ldots, x_n$) и тогда

$$\text{медиана}\,(x_1, \ldots, x_{n-k}, \ldots, x_n) \approx x_{(n+1)/2,\, n-k}, \tag{5.35}$$

где индекс в правой части означает $(n+1)/2$-ю порядковую статистику последовательности $x_1, \ldots, x_{n-k}$. Математические ожидания и дисперсии нормальных порядковых статистик можно найти в [5.12]. Приближенная формула для математического ожидания $x_{r,n}$:

$$E[x_{(r,n)}] \approx F^{-1}(r/(n+1)), \tag{5.36}$$

где $F(x)$ — функция распределения x. В заключение отметим, что полученные результаты можно, конечно, использовать при анализе не только перепадов, но и других объектов. Модель $x=s+z$ может быть пригодна для описания, например, импульсов с аддитивным шумом.

5.3.2. Распределение порядковых статистик в выборках из двух распределений

Пусть $x_1, \ldots, x_n$ — независимые случайные величины, причем $x_1, \ldots, x_k$ имеют функцию распределения $F_1(x)$ и плотность распределения $f_1(x)=F'_1(x)$, а $x_{k+1}, \ldots, x_n$ имеют функцию распределения $F_2(x)$ и плотность распределения $f_2(x)=F'_2(x)$. Тогда $x(r, n)$, r-я порядковая статистика из $x_1, \ldots, x_n$ [$r=(n+1)/2$ дает медиану] имеет плотность распределения

$$g(x)=g_1(x)+g_2(x), \tag{5.37}$$

где

$$g_1(x)=\sum_j k \binom{k-1}{j}\binom{n-k}{r-j-1} f_1(x) F_1^j(x) F_2^{r-j-1}(x) \times$$
$$\times [1-F_1(x)]^{k-j-1} [1-F_2(x)]^{n-k-r+j+1}, \tag{5.38}$$

$$g_2(x)=\sum_j (n-k)\binom{k}{j}\binom{n-k-1}{r-j-1} \times$$
$$\times f_2(x) F_1^j(x) F_2^{r-j-1}(x) [1-F_1(x)]^{k-j} [1-F_2(x)]^{n-k-r+j}. \tag{5.39}$$

Суммирование выполняется по всем натуральным числам j, для которых соответствующие биномиальные коэффициенты $\binom{p}{q}$ удовлетворяют условию $p \geqslant q \geqslant 0$.

Доказательство. Плотность распределения $g(x)x_{(r,n)}$ может быть найдена по формуле

$$g(x)=\lim_{dx \to 0+} \frac{1}{dx} P[x \leqslant x_{(r,n)} \leqslant x+dx]. \tag{5.40}$$

Метод доказательства заключается в том, что события в (5.40) делятся на подмножества, вероятности которых можно определить. Число различных подмножеств подсчитывается методами комби-

наторики. Мы также используем тот факт, что в бесконечно малый интервал $[x, x+dx]$ может попасть не более чем одна переменная x_i. Применены следующие подмножества:

A_1 — в интервал $[x, x+dx]$ попадает один член из последовательности $x_1, \ldots, x_k$;

A_2 — в интервал $[x, x+dx]$ попадает один член из последовательности $x_{k+1}, \ldots, x_n$;

B_j — точно j членов из последовательности $x_1, \ldots, x_k$ попадают в интервал $(-\infty, x)$, $j=0, \ldots, k$.

По закону полной вероятности

$$\frac{1}{dx} P[x \leqslant x_{(r,n)} < x+dx] =$$

$$= \sum_j \frac{1}{dx} P[\{x \leqslant x_{(r,n)} \leqslant x+dx\} \cap A_1 \cap B_j] +$$

$$+ \sum_j \frac{1}{dx} P[\{x \leqslant x_{(r,n)} \leqslant x+dx\} \cap A_2 \cap B_j]. \qquad (5.41)$$

Рассмотрим событие в первой сумме. Оно происходит тогда и только тогда, когда одна из величин $x_1, \ldots, x_k$ попадает в интервал $[x, x+dx]$; j величин из оставшихся в $x_1, \ldots, x_k$ попадают в интервал $(-\infty, x)$; $(r-j-1)$ величин из $x_{k+1}, \ldots, x_n$ попадают в интервал $(-\infty, x)$ и все оставшиеся величины — в интервал $(x+dx, +\infty)$. Вероятность этого события будет равна

$$kf_1(x)\,dx \binom{k-1}{j} F_1(x)^j \binom{n-k}{r-j-k} \times$$

$$\times F_2(x)^{r-j-1} [1-F_1(x+dx)]^{k-j-1} [1-F_2(x+dx)]^{n-k-r+j+1}. \qquad (5.42)$$

Подставляя (5.42) в (5.41) и устремляя dx к 0 справа, получим $g_1(x)$. С помощью аналогичных рассуждений получим $g_2(x)$ из второй суммы в (5.41). В этом случае один элемент последовательности $x_{k+1}, \ldots, x_n$ попадает в интервал $[x, x+dx]$.

5.4. Другие свойства медианных фильтров

В этом разделе рассматриваются ковариационные и спектральные свойства медианных фильтров. Полученные результаты показывают, что статистические характеристики второго порядка медиан очень похожи на соответствующие характеристики скользящих средних. Последняя часть раздела содержит некоторые результаты, касающиеся поведения выборочных функций на выходе медианных фильтров.

5.4.1. Ковариационные функции при белом шуме на входе

Ковариационные функции реализаций белого шума, отфильтрованных с помощью медианных фильтров, были рассчитаны в [5.10]. Сначала были выведены формулы для распределения пар

порядковых статистик частично перекрывающихся выборок, после чего путем численного интегрирования по этим формулам были найдены ковариации. Формулы распределения были получены с помощью развития идей, использованных в подразд. 5.3.2. Поскольку эти формулы достаточно сложны, они здесь не воспроизводятся, а даются результирующие ковариационные функции для случая нормального белого шума на входе медианного фильтра.

Пусть $\{x_i\}$ — независимые случайные величины $N(0, 1)$ и

$$C(n, m, k) = \mathrm{Cov}[\text{медиана}(x_1, \ldots, x_n), \text{медиана}(x_{n-k+1}, \ldots, x_{n-m+k})], \quad (5.43)$$

т. е. $C(n, m, k)$ — ковариация медиан выборок длиной n и m с интервалом перекрытия k. Численные значения $C(n, m, k)$ представлены в табл. 5.4 наряду с вероятностями $P(n, m, k)$ равенства значений двух медиан в (5.43). Для перекрывающихся выборок, т. е. при $k=0$, мы, конечно, имеем $C(n, m, k)=0$. Заметим также, что $C(n, n, n)=\mathrm{Var}$ (медиана). Ковариация нормальных случайных величин $N(m_x, \sigma)$ равна $\sigma^2 C(n, m, k)$. Автоковариационная функция для выходной последовательности n-точечного медианного фильтра, $y_i=$медиана (x_i), при нормальном белом шуме $N(m_x\sigma)$ на входе равна

$$r_y(\tau) = \mathrm{Cov}(y_{i+\tau}, y_i) = \sigma^2 C[n, n, (n-|\tau|)^+], \quad (5.44)$$

где $(a)^+=\max(a, 0)$. Ковариационная функция r_y для $n=3$, $\sigma=1$ и ковариационная функция r_z трехточечного скользящего среднего показаны на рис. 5.7. Для произвольного n и для белого шума на входе имеем

$$r_z(\tau) = \sigma^2 \left(1 - \frac{|\tau|}{n}\right)^+. \quad (5.45)$$

На рис. 5.7 видно, что r_y и r_z похожи по форме, или, что то же самое, нормализованные функции корреляции подобны друг другу, но r_y имеет большие значения, т. е. $r_y(0) > r_z(0)$.

Ковариационные функции для двумерных медианных фильтров можно выразить также через $C(n, m, k)$. Для квадратной апертуры $n \times n$

$$r_y(\tau_1, \tau_2) = \sigma^2 C[n^2, n^2, (n-|\tau_1|)^+ (n-|\tau_2|)^+]. \quad (5.46)$$

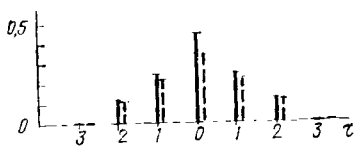


Рис. 5.7. Ковариационные функции для 3-точечной скользящей медианы (жирные линии) и 3-точечного скользящего среднего (пунктирные линии) при нормальном белом шуме $N(0, 1)$

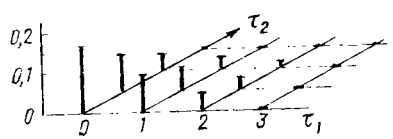


Рис. 5.8. Ковариационная функция для двумерного медианного фильтра с квадратной апертурой 3×3 при нормальном белом шуме $N(0, 1)$

Таблица 5

Ковариации $C(n, m, k)$ и вероятности равенства $P(n, m, k)$ значений двух медиан $N(0, 1)$-процесса для выборок длиной n и m с размером перекрытия k[$C(n, n, n)$=Var (медиана)]

n	m	k	$C(n, m, k)$	$P(n, m, k)$	n	m	k	$C(n, m, k)$	$P(n, m, k)$
1	1	1	1,0000	1,0000	25	25	5	0,0105	0,0253
					25	25	10	0,0214	0,0581
3	1	1	0,3333	0,3333	25	25	15	0,0329	0,1061
3	3	1	0,1177	0,1333	25	25	20	0,0453	0,1972
3	3	2	0,2480	0,3333	25	25	23	0,0538	0,3452
3	3	3	0,4487	1,0000	25	25	24	0,0572	0,4800
					25	25	25	0,0617	1,0000
5	1	1	0,2000	0,2000					
					49	49	7	0,0040	0,0125
5	3	1	0,0721	0,0857	49	49	14	0,0081	0,0272
5	3	2	0,1494	0,2000	49	49	21	0,0123	0,0456
5	3	3	0,2398	0,4000	49	49	28	0,0166	0,0701
					49	49	35	0,0209	0,1069
5	5	1	0,0445	0,0571	49	49	42	0,0258	0,1797
5	5	2	0,0914	0,1286	49	49	47	0,0296	0,3597
5	5	3	0,1422	0,2286	49	49	48	0,0305	0,4898
5	5	4	0,2003	0,4000	49	49	49	0,0318	1,0000
5	5	5	0,2868	1,0000					
					81	81	9	0,0018	0,0074
7	1	1	0,1428	0,1429	81	81	18	0,0039	0,0158
7	3	1	0,0520	0,0635	81	81	27	0,0059	0,0256
7	3	2	0,1069	0,1429	81	81	36	0,0079	0,0373
7	3	3	0,1672	0,2571	81	81	45	0,0099	0,0521
					81	81	54	0,0120	0,0721
7	5	1	0,0323	0,0433	81	81	63	0,0142	0,1028
7	5	2	0,0660	0,0952	81	81	72	0,0165	0,1649
7	5	3	0,1016	0,1619	81	81	79	0,0185	0,3658
7	5	4	0,1402	0,2571	81	81	80	0,0188	0,4938
7	5	5	0,1848	0,4286	81	81	81	0,0193	1,0000
7	7	1	0,0235	0,0333	1	1	1	1,0000	1,0000
7	7	2	0,0478	0,0722	9	1	1	0,1111	0,1111
7	7	3	0,0732	0,1195	9	9	9	0,1661	1,0000
7	7	4	0,1000	0,1810	25	1	1	0,0400	0,0400
7	7	5	0,1290	0,2698	25	9	9	0,0519	0,1199
7	7	6	0,1621	0,4286	25	25	25	0,0617	1,0000
7	7	7	0,2104	1,000	49	1	1	0,0206	0,0204
					49	9	9	0,0261	0,0550
9	1	1	0,1111	0,1111	49	25	25	0,0286	0,1157
9	3	1	0,0407	0,0505	49	49	49	0,0318	1,0000
9	3	2	0,0832	0,1111	81	1	1	0,0123	0,0123
9	3	3	0,1286	0,1905	81	9	9	0,0157	0,0320
					81	25	25	0,0171	0,0594
9	5	1	0,0253	0,0350	81	49	49	0,0180	0,1091
9	5	2	0,0516	0,0775	81	81	81	0,0193	1,0000
9	5	3	0,0791	0,1255					
9	5	4	0,1081	0,1905	121	1	1	0,0083	0,0083
9	5	5	0,1398	0,2857	121	9	9	0,0105	0,0210
					121	25	25	0,0114	0,0372
9	7	1	0,0156	0,0272	121	49	49	0,0119	0,0598
9	7	2	0,0376	0,0583	121	81	81	0,0123	0,1027
9	7	3	0,0573	0,0949	121	121	121	0,0129	1,0000
9	7	4	0,0778	0,1400					
9	7	5	0,0995	0,1991	2	2	2	0,5000	1,0000
9	7	6	0,1230	0,2857	4	4	4	0,2982	1,0000
9	7	7	0,1499	0,4444	6	6	6	0,2147	1,0000
					8	8	8	0,1682	1,0000
9	9	1	0,0146	0,0224	12	12	12	0,1175	1,0000
9	9	2	0,0296	0,0476	16	16	16	0,0904	1,0000
9	9	3	0,0449	0,0766	20	20	20	0,0734	1,0000
9	9	4	0,0608	0,1110	24	24	24	0,0619	1,0000
9	9	5	0,0774	0,1536	28	28	28	0,0535	1,0000
9	9	6	0,0949	0,2100	32	32	32	0,0471	1,0000
9	9	7	0,1138	0,2929	36	36	36	0,0420	1,0000
9	9	8	0,1352	0,4444	40	40	40	0,0380	1,0000
9	9	9	0,1661	1,0000					

Функция r_y в первом квадранте для апертуры 3×3 элемента показана на рис. 5.8. Значения функции в других квадрантах можно получить с учетом симметрии.

Сходство функций корреляции скользящей медианы и скользящего среднего до некоторой степени объясняется относительно высокой корреляцией между медианой и средним. Дополнительные объяснения даны в следующем подразделе. Если $\{x_i\}$ — независимые случайные величины с распределением $N(m, \sigma)$, то

$$\operatorname{Cov}\left[\text{медиана}(x_1, \ldots, x_n), \frac{1}{n}\sum_{1}^{n} x_i\right] = \operatorname{Var}(\bar{x}) = \frac{\sigma^2}{n} \quad (5.47)$$

(см. [5.3, с. 31]). Используя (5.12), для больших n получаем коэффициент корреляции

$$\rho(\text{медиана}, \bar{x}) \approx \sqrt{2/\pi} = 0{,}8. \quad (5.48)$$

5.4.2. Ковариационные функции при небелом шуме на входе

Дать общие точные формулы для автоковариационной функции отфильтрованного медианным фильтром небелого шума не представляется возможным. Мы приведем здесь некоторые приближенные формулы, которые были получены в [5.10] при рассмотрении предельных результатов, когда размер апертуры стремится к бесконечности. Эти формулы удивительно хорошо работают и для апертуры с малыми размерами. Подробный вывод читатель может найти в [5.10].

Допустим, что $\{x_i\}$ — стационарная перемешанная последовательность с маргинальной функцией распределения $F(x)$ и плотностью распределения $f(x)$. Имеем

$$P[\text{медиана}(x_{i-v}, \ldots, x_i, \ldots, x_{i+v}) \leqslant x] = P\left[\sum_{j=-v}^{v} \operatorname{sign}(x_{i+j} - x)] \leqslant 0\right]. \quad (5.49)$$

«Обращением» (5.49) можно получить следующую приближенную формулу представления (представления Бахадура) для больших n:

$$y_i = \text{медиана}(x_{i-v}, \ldots, x_i, \ldots, x_{i+v}) \approx$$
$$\approx \tilde{m} + \frac{1}{2f(\tilde{m})\, n} \sum_{j=-v}^{v} \operatorname{sign}(x_{i+j} - \tilde{m}), \quad (5.50)$$

где $v = (n-1)/2$ и $F(\tilde{m}) = 0{,}5$. Таким образом, скользящая медиана ведет себя как скользящее среднее знаковой функции элементов последовательности (результата их жесткого ограничения). Теперь, вычислив функцию ковариации скользящего среднего в правой части (5.50), можно получить приближенную формулу функции ковариации для последовательности, подвергнутой медианной фильтрации:

$$r_y(\tau) \approx \frac{1}{nf^2(\tilde{m})} \sum_{j=-(n-1)}^{n-1} \left(1 - \frac{|j|}{n}\right) c_{j+\tau}, \quad (5.51)$$

где $c_k = P(x_0 \leqslant \tilde{m},\ x_k \leqslant \tilde{m}) - 1/4$. Для нормального шума с ковариационной функцией $r_x(\tau) = \sigma^2 \rho(\tau)$ значения c_k могут быть подсчитаны точно. Пользуясь этим и произведя небольшую модификацию, упомянутую в подразд. 5.2.1, получаем

$$r_y(\tau) \approx \frac{\sigma^2}{n + \pi/2 - 1} \sum_{j=-(n-1)}^{n-1} \left(1 - \frac{|j|}{n}\right) \arcsin[\rho(j+\tau)]. \qquad (5.52)$$

В [5.10] мы проверили точность приближения (5.52) для нормального белого шума и нормальных процессов AR(1). Для входных процессов с нулевыми положительными (или не очень большими отрицательными корреляциями) точность будет хорошей даже при очень малых значениях n. С другой стороны, для процесса с функцией корреляции $\rho(\tau) = (-0{,}9)^{|\tau|}$ погрешность велика.

Скользящая медиана почти не сглаживает процессы, ведущие себя на больших интервалах, как функции вида $x_i \approx (-1)^i y$. В самом деле, форма входной последовательности $x_i = (-1)^i y$, $i \in \mathbb{Z}$, будет оставлена медианным фильтром без изменений, хотя для некоторых значений n она сдвинется на один шаг (см. гл. 6). В противоположность этому скользящее усреднение оказывает большое сглаживающее действие на подобный процесс, так как регулярные флуктуации значений x полностью уничтожаются. В целом можно ожидать, что приближенные формулы ковариационных функций скользящих медиан будут полезны только для последовательностей, на которые медианные фильтры действуют так же, как и скользящее усреднение. Так, в случае с сильно осциллирующими последовательностями и последовательностями перепадов большой пользы от них ждать не следует. Теперь можно объяснить сходство между корреляционными свойствами медианных фильтров и скользящего усреднения. Для больших n (5.51) можно аппроксимировать формулой

$$r(\tau) \approx \text{const}\,[1 - |\tau|/n]^{+}, \qquad (5.53)$$

которая асимптотически верна для всех скользящих средних, хотя и с разными константами. Таким образом, они имеют одинаковую нормализованную корреляционную функцию, но могут иметь разные асимптотические значения дисперсий.

В подразд. 5.2.2 упоминалось, что медианы по большим апертурам имеют приблизительно нормальное распределение. Это можно доказать, используя представления Бахадура и применяя центральную предельную теорему для стационарных перемешанных процессов в правой части (5.50). Описанные идеи можно также применить к двумерной медианной фильтрации. В этом случае мы получаем следующее представление Бахадура:

$$y_{ij} = \text{медиана}\,(x_{ij}) \approx$$

$$\approx \tilde{m} + \frac{1}{2f(\tilde{m})\,n} \sum_{(r,s)\in A} \text{sign}\,(x_{i+r,\,j+s} - \tilde{m}), \qquad (5.54)$$

где n — размер апертуры A. Для нормального шума $\{x_{ij}\}$ соответствующее приближение ковариационной функции будет иметь вид

$$r_y(\tau_1, \tau_2) \approx \frac{\sigma^2}{n + \pi/2 - 1} \times$$

$$\times \sum_{(r,s)\in A} \sum_{(r',s')\in A} \arcsin[\rho(\tau_1 + r - r', \tau_2 + s - s')]. \qquad (5.55)$$

Для некоторых апертур (5.55) можно еще упростить.

5.4.3. Отклик на косинусоидальные функции [1]

Для описания фильтров часто используют их импульсную реакцию, отклик на ступенчатую функцию и частотные характеристики. Так как медианный фильтр стирает импульсы и сохраняет перепады, импульсная реакция равна 0, а отклик на ступенчатую функцию равен 1. В данном подразделе мы найдем энергетический спектр результата фильтрации косинусоидальных колебаний медианного фильтра при $n=3$ и $n=5$. Сначала рассмотрим фильтрацию в непрерывном времени. Пусть для $0 \leqslant \omega_0 \leqslant \pi$

$$x(t) = \cos(\omega_0 t), \quad t \in R, \qquad (5.56)$$

$$y(t) = \text{медиана}\,[x(t-1), x(t), x(t+1)], \; t \in R. \qquad (5.57)$$

Легко увидеть (см. рис. 5.9a), что

$$y(t) = \begin{cases} \cos[\omega_0(t-1)], & 0 \leqslant t \leqslant 1/2; \\ \cos(\omega_0 t), & 1/2 \leqslant t \leqslant T/2 - 1/2; \\ \cos[\omega_0(t+1)], & T/2 - 1/2 \leqslant t \leqslant T/2, \end{cases} \qquad (5.58)$$

где $T = 2\pi/\omega_0$. Так как $y(t)$ — четная периодическая функция с периодом T, выражение (5.58) определяет $y(t)$ для всех t. Дисперсию $y(t)$ можно найти прямым интегрированием

$$\sigma_y^2 = \frac{1}{T} \int_0^T y^2(t)\, dt = \qquad (5.59а)$$

$$= \frac{1}{2}\left[1 + \frac{1}{\pi}(\sin 2\omega_0 - 2\sin\omega_0)\right], \; 0 \leqslant \omega_0 \leqslant \pi. \qquad (5.59б)$$

Можно разложить $y(t)$ в ряд Фурье с коэффициентами

$$c_k = \frac{1}{T} \int_0^T \exp(\mathrm{i}\, tk\omega_0)\, y(t)\, dt, \; k = 0, \pm 1, \pm 2, \ldots \qquad (5.60)$$

Простой подсчет дает

$$c_1 = \frac{1}{2}\left(1 - \frac{\omega_0}{\pi} - \frac{\omega_0}{\pi}\cos\omega_0\right), \; 0 \leqslant \omega_0 \leqslant \pi. \qquad (5.61)$$

---

[1] В оригинале дан заголовок «Частотная характеристика». Однако термин «частотная характеристика» используется для описания линейных фильтров, тогда как медианный фильтр нелинеен. Предлагаемый нами заголовок более точно отражает смысл характеристики, рассмотренной в этом подразделе. — *Прим. ред.*

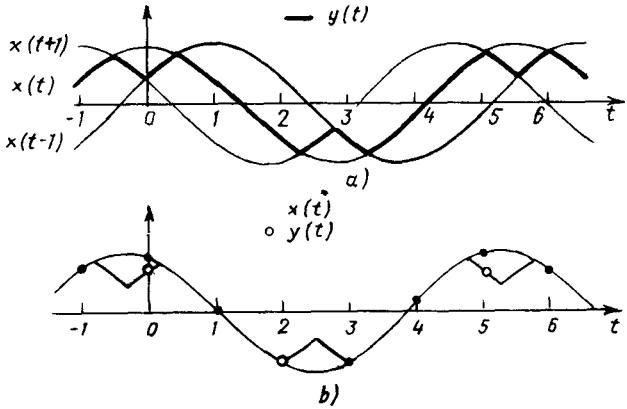


Рис. 5.9. Медианная фильтрация косинусоиды при $n=3$: *а* — непрерывное время, $x(t)=\cos(\omega_0 t)$; *б* — дискретное время, $x(t)=\cos(\omega_0 t+\theta)$, $\omega_0=9/8$, период $T=2\pi/\omega_0=5{,}6$, $y(t)=$медиана $[x(t-1),\ x(t),\ x(t+1)]$

Заметим, что $2c^2_1$ — спектральная плотность на частотах $\pm\omega_0$, а σ^2_y — полная энергия спектра. На рис. 5.10*а* показаны σ^2_y и $2c^2_1$ как функции ω_0 для $0\leqslant\omega_0\leqslant\pi$, а также зависимости σ^2_y для соответствующего 3-точечного скользящего среднего (для линейных фильтров $2c^2_1=\sigma^2_y$).

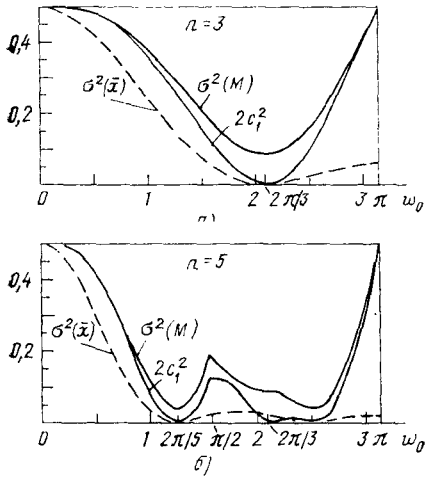
Видно, что для низкочастотной косинусоидальной функции на входе ($\omega_0\leqslant 2\pi/3$ или $T\geqslant n$) скользящая медиана и скользящее среднее имеют сходные характеристики, тогда как при $\omega_0>2\pi/3$ для медианы σ^2_y и $2c^2_1$ увеличиваются, а при $\omega_0=\pi$ они достигают того же значения, что и при $\omega_0=0$. Это объясняется тем, что 3-точечный медианный фильтр сохраняет форму последовательности $x_i=(-1)^i x$, но сдвигает ее на один шаг.

В дискретном времени выбор нуля в качестве фазового сдвига в (5.56) достаточно произволен. Если же сдвиг фазы θ считать случайным и имеющим равномерное распределение на интервале $[0, 2\pi]$, получаем стационарный процесс

$$x(t)=\cos(\omega_0 t+\theta),\ t=0,\pm 1,\pm 2,\ldots,\ldots, \tag{5.62}$$

и выход медианного фильтра (см. рис. 5.9*б*) будет иметь те же ковариационные свойства, что и в непрерывном времени (5.57). Энергетический спектр последовательности на входе медианного фильтра получается из (5.60) путем сложения спектральных плотностей c^2_k для $k\omega$ на интервале $[-\pi, \pi]$. Для значений ω_0, которые укладываются в π целое число раз, это может привести к перекрытию спектральных плотностей, но независимо от этого можно рассматривать $2c^2_1$ как спектральную плотность на частоте $\pm\omega_0$.

Мы провели аналогичный анализ для медианных фильтров при $n=5$. Результаты представлены на рис. 5.10,*б*. Вывод формул был достаточно громоздким. Результаты согласуются с эксперимента-



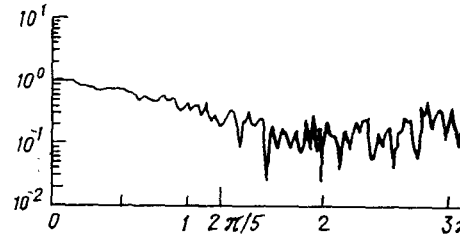


Рис. 5.11. Сечение эмпирической «передаточной функции» для изображения 5×5, подвергнутого медианной фильтрации (воспроизведено из [5.14, рис. 3.6] с любезного разрешения автора)

←

Рис. 5.10. Фильтрация косинусоиды. Дисперсия $\sigma^2(M)$, спектральная плотность $2c_1^2$ на частоте ω_0 для косинусоиды, подвергнутой медианной фильтрации, и дисперсия $\sigma^2(x)$ для косинусоиды, подвергнутой фильтрации с помощью скользящего усреднения

ми на моделях, проведенными в [5.13]. При $\omega_0 \leqslant 2\pi/5$, т. е. $T \geqslant n=5$, спектральные характеристики скользящих медиан и скользящих средних одинаковы. В случае произвольного n это будет верно при $T \geqslant n$. Для больших значений n результаты могут быть получены численным интегрированием (5.59а) и (5.60), в которых $y(t)$ определяется (5.57).

Двумерная медианная фильтрация с квадратными апертурами 3×3 и 5×5 косинусоидального сигнала вдоль оси абсцисс точечного растра дает такие же значения дисперсии и спектральные компоненты, как показано выше.

Медианные фильтры нелинейны, поэтому описанные частотные отклики для одиночных косинусоидальных функций не соответствуют передаточным характеристикам для произвольных сигналов, являющихся суммой косинусоидальных функций. Хейгстер [5.14] подсчитал эмпирические частотные характеристики как отношение преобразований Фурье выходных и входных изображений. Пример для медианного фильтра с апертурой 5×5 приведен на рис. 5.11. На нем показано сечение частотной характеристики, являющейся функцией двух переменных, вдоль одной из координатных осей. Заметим, что для частот $\omega_1 \leqslant 2\pi/5$ кривая является очень гладкой и ее можно интерпретировать как частотную характеристику, тогда как для $\omega_1 \geqslant 2\pi/5$ кривая становится резко нерегулярной из-за интерференции различных частот. Разумеется, такие частотные характеристики зависят от выбираемых входных изображений. Результаты Хейгстера подтверждают отмеченное выше сходство частотных характеристик скользящих медиан и скользящих средних для частот $\omega \leqslant 2\pi/n$, или $T \geqslant n$.

5.4.4. Свойства выборочных функций

Мы уже обсуждали действие медианной фильтрации на перепады и сильно осциллирующие последовательности. За исключением таких последовательностей, общая форма последовательностей, подвергнутых медианной фильтрации, близка к форме последовательностей, подвергнутых фильтрации с помощью скользящего среднего. Этот вывод следует из представления Бахадура (5.50).

Хотя общий вид последовательностей скользящей медианы и скользящего среднего в целом одинаков, в деталях имеются различия. Несколько авторов отмечали, что последовательности, подвергнутые медианной фильтрации, включают в себя много элементов с одинаковыми значениями. Простую оценку вероятности равенства следующих друг за другом величин дает такой результат. Пусть последовательность на входе $\{x_i\}$ — стационарный перемешанный случайный процесс. Тогда

$$\lim_{n\to\infty} P[\underset{n}{\text{медиана}}\ (x_i) = \text{медиана}\ (x_{i+1})] = 0{,}5. \qquad (5.63)$$

Таким образом, примерно 50% всех членов в последовательности, подвергнутой медианной фильтрации, равны соседним справа при больших n. (Заметим, что это не то же самое, что равенство любому из двух соседних элементов слева и справа.)

Доказательство. При движении фильтра от i к $i+1$ в апертуре фильтра x_{i-v} заменяется на x_{i+1+v}. Если эти две случайные величины лежат по одну сторону от $y = \text{медиана}\ (x_i)$, то медиана $(x_{i+1}) =$ $=\text{медиана}\ (x_i)$. Из условия перемешивания вытекает, что x_{i-v} и x_{i+1+v} асимптотически независимы. Наряду с соотношением медиана $(x_i) \approx m$ это дает

$$P[\text{медиана}\ (x_i) = \text{медиана}\ (x_{i+1})] \approx P(x_{i-v} < \hat{m}, x_{i+1+v} < \hat{m}) +$$

$$+ P(x_{i-v} > \hat{m}, x_{i+1+v} > \hat{m}) = 0{,}5 \cdot 0{,}5 + 0{,}5 \cdot 0{,}5 = 0{,}5, \qquad (5.64)$$

что доказывает (5.63).

Для входного сигнала в виде белого шума вероятность в (5.63) может быть подсчитана точно: она равняется $0{,}5(1-1/n)$ [5.10]. Для двумерной фильтрации такой простой результат, как (5.63), отсутствует, но для белого шума на входе и квадратных апертур при подсчете вероятностей равенства значений

$$P[\text{медиана}\ (x_{ij}) = \text{медиана}\ (x_{i,\,j+1})] = P(n^2,\ n^2, n^2-n) \qquad (5.65)$$

можно использовать табл. 5.4, что для апертуры 3×3 дает 0,2100, а для апертуры 9×9 — 0,1649.

Другой особенностью последовательностей, подвергнутых медианной фильтрации, является наличие небольших случайных перепадов, что отличает их от последовательностей, подвергнутых фильтрации скользящим усреднением, которые характеризуются очень медленными изменениями. Для удаления таких случайных перепадов Рабинер [5.5] применял после медианной фильтрации 3-точечный линейный фильтр.

5.5. Некоторые другие фильтры, сохраняющие перепады

Рассмотрим фильтры, которые имеют такие же основные свойства, как и медианные: свойства сохранения перепадов и подавления шума. Все фильтры, за исключением тех, которым посвящена последняя часть раздела, являются вариантами медианных фильтров. Описанные выше медианные фильтры здесь будут называться *простыми медианными фильтрами*. Формулы дисперсии для выхода фильтров даны для белого шума на входе.

5.5.1. Линейная комбинация медиан

Пусть A_k $(k=1, \ldots, K)$ — различные апертуры. Тогда *фильтр, полученный с помощью линейной комбинации медиан* [5.15, 5.16], определяется следующим образом:

$$y_{ij} = \sum_{k=1}^{K} a_k \underset{A_k}{\text{медиана}} (x_{ij}), \tag{5.66}$$

где a_k — вещественные коэффициенты. Апертуры могут быть, например, квадратами со стороной $1, 3, \ldots, 2k-1$ или окружностями с диаметром $1, 3, \ldots, 2k-1$. Можно, разумеется, также выбрать наборы апертур, которые не включают начало координат, например, квадратные рамки или кольца (см. рис. 5.2).

Если все апертуры A_k симметричны относительно центра координат и содержат его по условиям (5.5), (5.6), тогда форма изображения перепада $\{x^0_{ij}\}$ сохранится, а высота перепада изменится в $\Sigma\, a_k$ раз:

$$y_{ij} = \left(\sum_{k=1}^{K} a_k \right) x^0_{ij}. \tag{5.67}$$

Это следует из того, что каждая медиана в комбинации сохраняет перепад. Заметим, что если $\Sigma a_k = 0$, то $y_{ij} = 0$.

Для нормального белого шума $\{x_{ij}\}$ на входе можно вычислить дисперсию y_{ij}, используя результаты подразд. 5.4.1:

$$\mathrm{Var}\,(y_{ij}) = \sum_{k=1}^{K} \sum_{m=1}^{K} a_k\, a_m\, C(n_k, n_m, n_{km}), \tag{5.68}$$

где $C(n_k, n_m, n_{km})$ обозначает то же, что и в подразд. 5.4.1, а n_k, n_m, n_{km} — число точек в апертурах A_k, A_m и $A_k \cap A_m$ соответственно. В табл. 5.4 приведены необходимые значения $C(n_k, n_m, n_{km})$ для случаев, когда апертуры имеют форму квадратов, колец и квадратных рамок. В двух последних случаях апертуры не перекрываются и для них формула дисперсии упрощается:

$$\mathrm{Var}\,(y_{ij}) = \sum_{k=1}^{K} a_k^2 C(n_k, n_k, n_k). \tag{5.69}$$

Другим способом выяснить поведение линейной комбинации медианных фильтров является использование сходства многих свойств простых медианных фильтров и скользящего усреднения.

Если заменить каждую медиану в (5.66) средним арифметическим тех же величин x, получим линейный фильтр

$$z_{ij} = \sum_{k=1}^{K} a_k \underset{A_k}{\text{среднее}} (x_{ij}). \tag{5.70}$$

Легко подсчитать моменты второго порядка для этого фильтра. Они дают по крайней мере некоторую информацию о свойствах соответствующего медианного фильтра. Можно также вернуться назад, начать с линейного фильтра с требуемыми свойствами и построить соответствующую линейную комбинацию медианных фильтров. Это особенно просто, если апертуры взаимно не перекрываются (как например, квадратные рамки или кольца).

5.5.2. Взвешенно-медианные фильтры

В простых медианных фильтрах все величины x в пределах апертуры влияют на результат фильтрации. Но иногда желательно придать бо́льший вес центральным точкам. Эта возможность реализована в фильтре следующего вида. Для простоты рассмотрим только одномерные фильтры.

Основная идея состоит в изменении числа элементов в каждом наборе $(x_{i-v}, x_i, \ldots, x_{i+r})$ путем повторения k_r раз элементов x_{i+r} и нахождения медианы по такой растянутой последовательности чисел. Назовем полученный фильтр *взвешенно-медианным*. Поясним это с помощью простого примера. Для $n=3$, $k_{-1}=k_1=2$, $k_0=3$ имеем

$$\begin{aligned} y_i = &\text{взвешенная медиана } (x_{i-1}, x_i, x_{i+1}) = \\ &= \text{медиана } (x_{i-1}, x_{i-1}, x_i, x_i, x_i, x_{i+1}, x_{i+1}). \end{aligned} \tag{5.71}$$

Если веса симметричны, $k_{-r}=k_r$ и если k_0 нечетно, то взвешенно-медианный фильтр сохраняет перепады.

Если входной сигнал $\{x_i\}$ является белым шумом с плотностью вероятностей $f(x)$, то взвешенная медиана с весами $\{k_r\}$ для больших n распределена приблизительно по нормальному закону $N(m, \sigma_n)$, где m — теоретическая медиана и

$$\sigma_n^2 = \frac{\sum k_r^2}{\left(\sum k_r\right)^2} \cdot \frac{1}{4 f^2 (\tilde{m})}. \tag{5.72}$$

Так как этот результат нигде больше не приводился, дадим краткое указание для его доказательства. Используя обобщенное представление Бахадура [см. (5.50)], получаем:

$$\begin{aligned} y_i = &\text{взвешенная медиана } (x_i) \approx \tilde{m} + \\ &+ \frac{1}{2f(\tilde{m}) \sum k_r} \sum_{r=-v}^{v} k_r \operatorname{sign} (x_{i+r} - \tilde{m}). \end{aligned} \tag{5.73}$$

По центральной предельной теореме выражение в правой части имеет приблизительно нормальное распределение при выполнении условия Линдберга, которое для весов последовательности $k_r = k_r^{(n)}$ имеет вид:

$$\max_r \frac{[k_r^{(n)}]^2}{\sum_i [k_i^{(n)}]^2} \to 0 \text{ при } n \to \infty. \tag{5.74}$$

Это условие по существу требует, чтобы ни один из весов k_r не был значительно больше остальных.

5.5.3. Итерационные медианные фильтры

Поскольку при медианной фильтрации сохраняются перепады, то они сохраняются и при *итеративном применении медианных фильтров* [5.1, 5.8]. Тьюки предложил замечательную методику сглаживания, заключающуюся в том, чтобы повторять медианную фильтрацию до тех пор, пока не прекратятся изменения, т. е. пока не получится последовательность, инвариантная к медианной фильтрации (стабильная точка, ср. гл. 6). Заметим, что такая инвариантная последовательность не обязательно должна состоять из одинаковых элементов (в отличие от последовательностей, получаемых при итерации скользящего усреднения стационарных последовательностей $\{x_i\}$, которые в конце концов становятся последовательностями с постоянными значениями).

Статистические свойства итерационных медианных фильтров проанализировать трудно. Можно поделиться здесь некоторым опытом, полученным при моделировании одномерных процессов AR(1). При неотрицательно коррелированных AR(1)-последовательностях с параметром $a \geqslant 0$ для получения инвариантной последовательности было нужно только несколько итераций и после первой медианной фильтрации наблюдались лишь небольшие изменения. Поэтому представляется вероятным, что формулы дисперсии для простых медиан приближенно справедливы также и для итерационных медианных фильтров при обработке процессов с неотрицательными ковариационными функциями. Для процессов AR(1) с чередующимися положительными и отрицательными корреляциями, $a < 0$, для получения инвариантной последовательности потребовалось большое число итераций, и во время процесса фильтрации происходили большие изменения. Результирующие последовательности были гораздо более гладкими и ближе к среднему уровню, чем последовательности после одного шага фильтрации.

При использовании итерационных медианных фильтров можно, конечно, менять апертуры от шага к шагу итерации. Прэтт [5.8] и Нарендра [5.17] исследовали метод двумерной фильтрации, в котором сначала к каждой строке изображения применяется одномерный медианный фильтр, а затем к каждому столбцу обработан-

ного изображения также применяется одномерный медианный фильтр, т. е. сначала

$$z_{ij} = \text{медиана}(x_{i,j-v}, \ldots, x_{i,j}, \ldots, x_{i,j+v}), \tag{5.75}$$

после чего

$$y_{ij} = \text{медиана}(z_{i-v,j}, \ldots, z_{i,j}, \ldots, z_{i+v,j}). \tag{5.76}$$

Такой фильтр называется *разделимым медианным фильтром*. Его статистические свойства можно проанализировать теоретически, если $\{x_{ij}\}$ — белый шум с плотностью вероятностей $f(x)$ (см. [5.17]). Главное заключается в том, что переменные z в (5.76) независимы, так как они определяются переменными x в различных строках. Нарендра нашел точный вид плотности вероятностей f_z переменных z с помощью (5.8) и подставил f_z в приближенную формулу дисперсии (5.9) для y_{ij}. Мы приведем несколько более простую формулу, полученную при использовании того факта, что согласно (5.9) f_z есть приблизительно нормальная плотность $N(\tilde{m}, \sigma_n)$. Подставив эту величину в (5.12), получим

$$\operatorname{Var}(y_{ij}) \approx \frac{1}{n + \pi/2 - 1} \frac{\pi}{2} \frac{1}{4f^2(\tilde{m})n}. \tag{5.77}$$

Для нормальных $N(m, \sigma)$ величин $\{x_{ij}\}$ с помощью (5.77) и небольшой модификации, описанной в подразд. 5.2.1, получим

$$\operatorname{Var}(y_{ij}) \approx \frac{\sigma^2}{(n + \pi/2 - 1)^2} \left(\frac{\pi}{2}\right)^2. \tag{5.78}$$

При больших n это дает дисперсию выходного сигнала, равную $2{,}47\,(\sigma/n)^2$, тогда как дисперсия для простой медианы по апертуре $n \times n$ равна $1{,}57\,(\sigma/n)^2$, а дисперсия скользящего среднего по апертуре $n \times n$ равна $(\sigma/n)^2$. Дисперсия последовательности на выходе разделимого медианного фильтра примерно на 57% больше, чем дисперсия медианы по апертуре $n \times n$, и примерно на 147% превышает дисперсию скользящего среднего.

5.5.4. Сглаживание остатка

Предположим, что переменная x удовлетворяет модели типа «сигнал плюс шум»:

$$x_i = s_i + n_i, \tag{5.79}$$

где сигнал $\{s_i\}$ изменяется медленно по сравнению с шумом $\{n_i\}$. Медианная фильтрация дает оценку s_i:

$$y_i = \text{медиана}(x_i) \approx s_i. \tag{5.80}$$

Таким образом, остаток

$$\hat{n} = x_i - \text{медиана}(x_i) \approx n_i \tag{5.81}$$

является оценкой шума. Дальнейшая медианная фильтрация остатка может еще больше уменьшить шум

$$z_i = \text{медиана}(\hat{n}_i) \approx 0. \qquad (5.82)$$

Можно надеяться, что суммирование y_i и z_i даст хорошую оценку s_i:

$$s_i = y_i + z_i = \text{медиана}(x_i) + \text{медиана}[x_i - \text{медиана}(x_i)]. \qquad (5.83)$$

Такая методика сглаживания остатка (или двойного сглаживания) была предложена и изучена в [5.5, 5.13 и др.]. Легко понять, что такая методика сохраняет перепады. Некоторые другие методы комбинированного сглаживания можно найти в [5.18].

5.5.5. Адаптивные фильтры, сохраняющие перепады

Фильтры, описанные выше, являются сглаживающими фильтрами общего назначения. Для ограниченных классов изображений могут быть разработаны специальные фильтры. Один важный класс изображений встречается в дистанционном зондировании при классификации точек изображения на ограниченное число классов. Изображения могут быть описаны как состоящие из компактных участков с постоянным значением сигнала. Другими словами, можно считать, что идеальное изображение состоит из участков с постоянным значением сигнала, но наблюдаемые изображения искажены добавлением шума. Для этого типа изображений было предложено несколько алгоритмов сглаживания (см., например, [5.19, 5.20], где можно найти дополнительные ссылки). В предложенных алгоритмах были использованы следующие основные принципы.

Апертура имеет центр в точке (i, j). Если в пределах апертуры наблюдается малое различие между значениями, то (i, j) считается внутренней точкой и ее значение заменяется средним значением по апертуре. Если же, наоборот, различие между значениями внутри апертуры велико, то (i, j) считается пограничной точкой и ей приписывается среднее значение по точкам, лежащим с той же стороны от границы, что и (i, j).

В качестве меры различия можно использовать локальные дисперсии, значения Лапласиана или градиента. С увеличением размеров апертуры объем вычислений резко возрастает, поэтому было предложено итерировать эти алгоритмы, вместо того чтобы использовать большие апертуры[1]. Такие алгоритмы не только сохраняют резкие перепады, но и обостряют смазанные. Например, наклонные плоскости превращаются в ступеньки. Таким образом, алгоритмы не сохраняют наклонные перепады на самом деле. Некоторые из алгоритмов предназначены для сохранения не только перепадов, но и острых углов граничных участков. Более детальное изучение материала, изложенного в этой части, читатель может найти в [5.19, 5.20].

[1] Имеется в виду последовательное применение алгоритмов с малой апертурой. — *Прим. ред.*

5.6. Использование медиан и других порядковых статистик в обработке изображений

В предыдущих разделах мы исследовали способность медианных фильтров осуществлять сглаживание. В этом разделе приведем некоторые примеры того, как медианные фильтры можно комбинировать с другими способами обработки изображений. Примеры взяты из задач обнаружения границ, выделения объектов и классификации. Заключительная часть содержит краткий перечень общих порядковых статистик и их использования в обработке изображений.

5.6.1. Обнаружение границ

Свойство медианных фильтров сохранять перепад делает возможным их применение в качестве *предварительных фильтров* при обнаружении границ, т. е. для сглаживания изображений перед применением обнаружителя границ. Многие обнаружители границ принадлежат к типу

$$w_{ij}=\begin{cases}1, & \text{если } |g_1|>\Delta \text{ или } |g_2|>\Delta,\\ 0 & \text{в противном случае,}\end{cases} \tag{5.84}$$

где g_1, g_2 — градиенты в точке (i, j), а Δ — пороговое значение. После того как было выполнено предварительное сглаживание, полезно использовать градиенты, являющиеся симметричными функциями, найденными по окрестности, большей единицы. Например,

$$g_1=y_{i+u,j}-y_{i-u,j},\ g_2=y_{i,j+u}-y_{i,j-u}, \tag{5.85}$$

где $1/2\leqslant u\leqslant n/2$, а n — размер боковой стороны сглаживающей апертуры. Заметим, что использование медианных фильтров и медианных обнаружителей границ равнозначно использованию блочного обнаружителя границ, в котором градиенты находятся по медианам в блоках:

$$\begin{aligned}g_1&=\underset{A_1}{\text{медиана}}\,(x_{ij})-\underset{A_2}{\text{медиана}}\,(x_{ij}),\\ g_2&=\underset{B_1}{\text{медиана}}\,(x_{ij})-\underset{B_2}{\text{медиана}}\,(x_{ij}).\end{aligned} \tag{5.86}$$

Квадратные блоки с размерами 3×3 показаны на рис. 5.12. Размер окрестности, по которой находится градиент, равен 2 ($u=1$).

Скользящие медианы могут быть также использованы для *повторной фильтрации* после усиления границ [5.7]. Резкая граница, смазанная низкочастотным линейным фильтром, может быть обострена посредством либо инверсной фильтрации, либо ее модификации. Так как импульсные характеристики линейных фильтров инверсного типа часто имеют большие боковые лепестки, то они

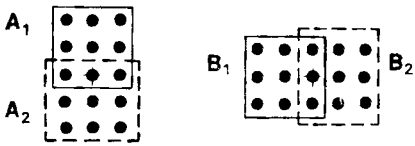


Рис. 5.12. Обнаружение границ. Предварительная фильтрация медианным фильтром (с апертурой 3×3) соответствует обнаружителю границ блочного типа. Центральная точка помечена крестиком

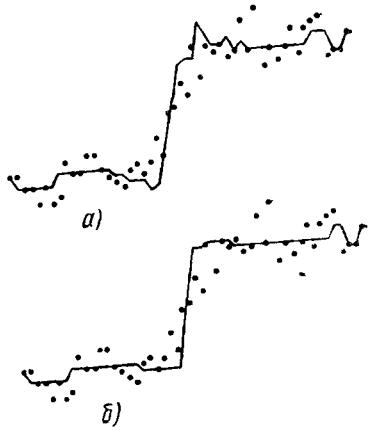


Рис. 5.13. Восстановление ступенчатого перепада с помощью линейного фильтра (*а*), пяти итераций линейного и медианного фильтров (*б*); точки показывают данные на входе, выходные точки соединены между собой линиями. (Воспроизведено из [5.7, рис. 2.6] с любезного разрешения Американского оптического общества © и Б. Р. Фридена)

обычно приводят к ложным колебаниям на границах (повторам) (ср. с рис. 5.13,*а*). Фриден [5.7] использовал медианный фильтр для уничтожения таких ложных колебаний (на рис. 5.13,*б* линейное восстановление и медианная фильтрация были повторены пять раз). Используемый размер апертуры $n=11$ был равен периоду колебаний. Заметим, что граница на рис. 5.13,*б* не только имеет меньше ложных колебаний, но также острее, чем граница на рис. 5.13,*а*.

5.6.2. Выделение объектов

Выделение объектов на изображении — это одна из характерных задач обработки изображений. Будем считать, что объекты разбросаны на плавно изменяющемся фоне и что они достаточно хорошо разделены, как например, на аэрофотоснимке камней, разбросанных на холмах. Можно предложить следующий метод выделения объектов: сначала изображение сглаживается, с тем чтобы получить оценку переменного фона, после чего эта оценка вычитается из исходного изображения. В идеальном случае после этой обработки объекты остаются на постоянном фоне (с нулевым уровнем) и их можно обнаружить путем сравнения с порогом. В качестве оператора сглаживания можно использовать, например, скользящую медиану или скользящее среднее, т. е.

$$y_{ij}=x_{ij}-\underset{A}{\text{медиана}}\,(x_{ij}) \tag{5.87}$$

или

$$z_{ij}=x_{ij}-\underset{A}{\text{среднее}}\,(x_{ij}). \tag{5.88}$$

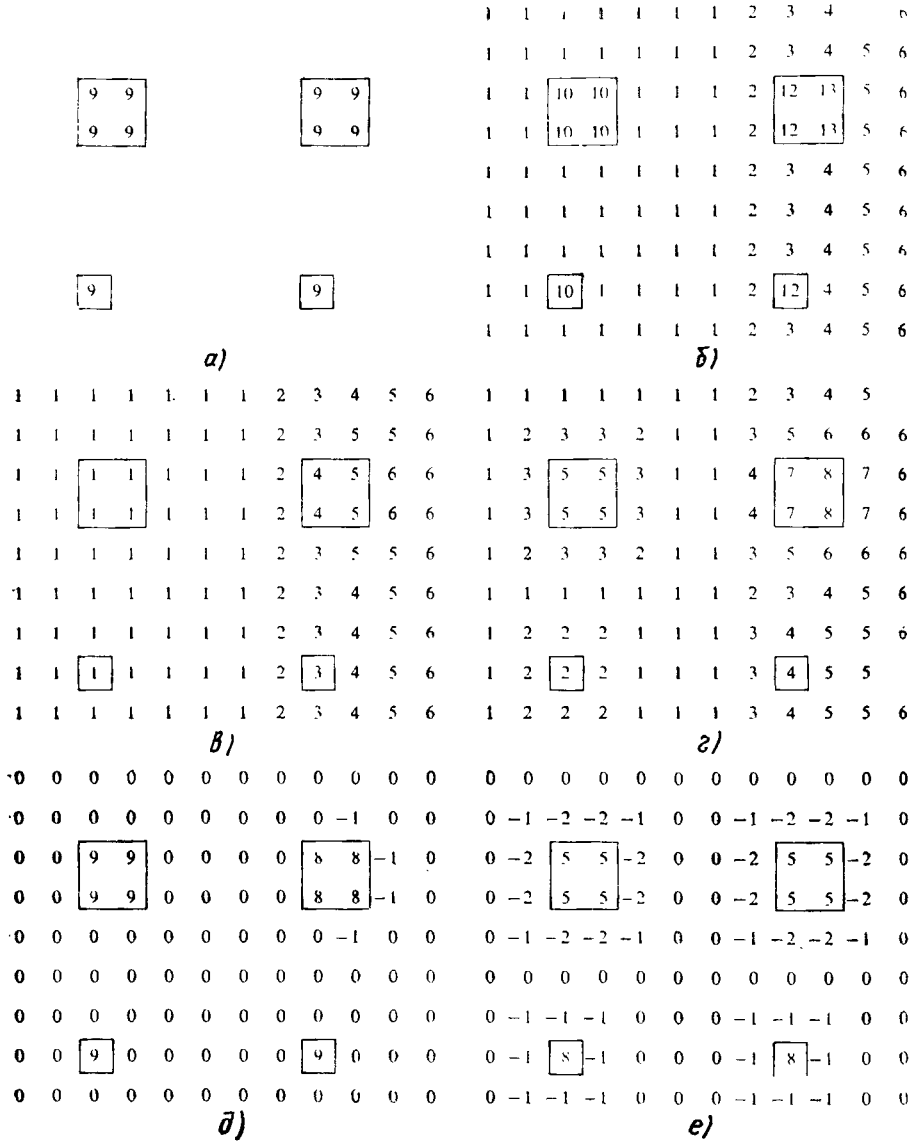


Рис. 5.14. Выделение объектов:
а — исходные объекты; *б* — изображение на входе фильтра, состоящее из объектов на постоянном фоне; *в* — медианная фильтрация входного изображения с апертурой 3×3; *г* — фильтрация с помощью скользящего среднего по окрестности 3×3; *д* — разность *в* и входного изображения; *е* — разность *г* и входного изображения

Апертуры могут иметь вид квадратов или крестов. Если размеры апертуры, по крайней мере, вдвое больше размеров объектов, то скользящая медиана стирает объекты и, следовательно, вычи-

тание в (5.87) дает полное их выделение. Это верно только для частей изображения с постоянным фоном. В других частях возникают некоторые искажения объектов. Однако фильтр, основанный на скользящем среднем (5.88), дает значительно больше искажений во всех точках (рис. 5.14). Наконец, заметим, что (5.87) — это линейная комбинация с медианным фильтром при $\Sigma a_k = 1 - 1 = 0$, и, таким образом, обработка по (5.87) не затрагивает границ объектов.

5.6.3. Классификация

Многоспектральная классификация изображений при дистанционном зондировании обычно проводится как поэлементная классификация, т. е. при классификации точки используются значения спектров только в этой точке. Однако для зашумленных изображений нужен тот или иной вид сглаживания, в связи с чем было предложено несколько методов.

Наиболее легко и часто применяемый метод — это сглаживание результата поэлементной классификации посредством решения по локальному большинству. Другой простой метод состоит в предварительном сглаживании каждой спектральной компоненты с помощью скользящих средних или скользящих медиан. Все методы имеют тот недостаток, что элемент изображения, который однозначно классифицируется по своим спектральным значениям, после операции сглаживания может быть отнесен к другому классу вследствие влияния соседних элементов. Комбинируя спектральные значения в каждой точке с медианой соседних точек и используя эти комбинированные значения для классификации, получим алгоритм, лишенный упомянутого недостатка.

Пусть $x^k{}_{ij}$ $(k = 1, \ldots, K)$ обозначают спектральные величины в точке (ij) и пусть

$$y_{ij}^{(k)} = \underset{A}{\text{медиана}} \, [x_{ij}^{(k)}]. \tag{5.89}$$

Тогда при классификации по обычным правилам для многомерных нормальных распределений можно использовать расширенный вектор признаков:

$$(x_{ij}^{(k)}, y_{ij}^{(k)}, \ k = 1, \ldots, K).$$

Использование медианных фильтров для сглаживания обосновывается их свойством сохранять перепады, что в данном случае приводит к сохранению границ классов.

5.6.4. Порядковые статистики общего вида

Для n чисел $x_1, \ldots, x_n$ r-я порядковая статистика $x_{(r,n)}$ определяется как r-е по значению число из данных n. В частности, $x_{(1,n)} = \min x_i$, $x_{(n,n)} = \max x_i$, а $x_{[(n+1)/2,n]} = \text{медиана } (x_i)$. Фильтрация со *скользящими порядковыми статистиками* осуществляется анало-

гично медианной фильтрации: апертура скользит по изображению и подсчитывается r-я порядковая статистика значений внутри апертуры. Такие фильтры также называются *процентильными фильтрами.*

Скользящие порядковые статистики могут быть использованы для сжатия, сужения и растяжения (расширения) объектов в изображении (эрозия и расширение). Они применялись, например, к клеточным изображениям [5.14]. Для сжатия используется $r<(n+1)/2$, например, $r=1$, а для расширения — $r>(n+1)/2$, например, $r=n$. Для $r=(n+1)/2$, т. е. для медиан, средний уровень изображения не меняется. Сжатие и расширение часто осуществляются путем сравнения исходного изображения с порогом и последующего применения скользящих решающих правил типа r- из n. Заметим, что фильтрация с помощью скользящих порядковых статистик при последующем сравнении с порогом дает точно такие же результаты. Промежуточное изображение может содержать значимую полезную информацию. На рис. 5.15 показана фильтрация с апертурой 3×3 и $(r, n)=(2,9)$.

Статистическая литература по порядковым статистикам огромна. Большинство представленных выше результатов по медианам имеют аналоги для порядковых статистик общего вида [5.3, 5.10]. Для входного сигнала с белым шумом могут быть получены как точное, так и асимптотическое распределения, тогда как для небелого шума имеются только асимптотические. Однако предельное поведение экстремальных порядковых статистик [для $r/n\to0$ или 1, например, $x_{(1,n)}$ и $x_{(n,n)}$] различно по своей природе. В частности, предельные распределения не являются нормальными.

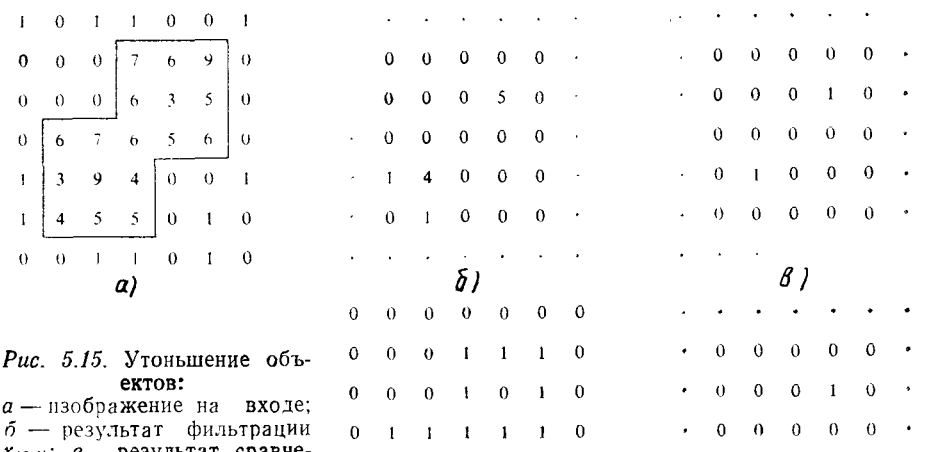


Рис. 5.15. Утоньшение объектов:
а — изображение на входе; *б* — результат фильтрации $x_{r,9}$; *в* — результат сравнения *б* с порогом, равным 4; *г* — результат сравнения исходного изображения с порогом, равным 4; *д* — результат фильтрации 2-из-9 изображения *г*

Для скользящих порядковых статистик — стационарных, удовлетворяющих условию перемешивания процессов $\{x_i\}$ при r/n, достаточно отличающихся от 0 и 1, — среднее выходного сигнала приблизительно равно λ-процентилю маргинальной функции распределения $F(x)$ величин x_i, где $\lambda = r(n+1)$, т. е.

$$E[x_{(r,n)}] \approx x_\lambda = F^{-1}(\lambda), \tag{5.90}$$

а автоковариационная функция асимптотически равна

$$r(\tau) \approx \text{const}\left(1 - \frac{|\tau|}{n}\right)^+, \tag{5.91}$$

т. е. сходна с (5.53) для медиан. С помощью (5.90) можно оценить, насколько изменится средний уровень последовательности или изображения. Формула (5.91) показывает, что скользящие порядковые статистики дают сглаживающий эффект, если значение r/n не слишком близко к 0 или 1.

Хейгстер [5.14] исследовал эмпирические передаточные функции (ср. подразд. 5.4.3) для скользящих порядковых статистик. Если значение r/n не слишком близко к 0 или 1, эти функции похожи на передаточные функции медианных и частотные характеристики низкочастотных линейных фильтров, но для r/n, близких к 0 или 1, они не являются гладкими даже на низких частотах.

Признательность. Автор хотел бы поблагодарить проф. Б. Розе и д-ра Т. Орхауга за полезные обсуждения в процессе этой работы, а также Гуниллу Джонсон за подготовку иллюстраций.

ГЛАВА 6

МЕДИАННАЯ ФИЛЬТРАЦИЯ: ДЕТЕРМИНИРОВАННЫЕ СВОЙСТВА

(Ш.-Г. Тян) [1]

В данной главе на основе детерминированного подхода излагаются некоторые новые результаты, касающиеся свойств медианных фильтров. Наша цель — найти для каждого медианного фильтра специальный класс последовательностей, которые не изменяются под действием этого фильтра. Такие последовательности называются *стабильными точками* фильтра. Необходимость решения поставленной задачи может вначале показаться непонятной, поэтому следует сказать несколько слов об истории развития медианной фильтрации. Медианная фильтрация в том виде, в каком она была введена в [6.1], используется главным образом вследствие ее вычислительной простоты и нечувствительности к помехам,

[1] M/A-COM Labs, 11717 Exploration Lane Germantown, MD 20767, USA.

имеющим распределение с тяжелыми хвостами. Затем обратили внимание на другое свойство медианных фильтров, а именно, на то, что действуя как сглаживающие фильтры, они в то же время могут сохранять крутые перепады, или контуры в сигналах. Эта способность сохранять края делает медианные фильтры удобными сглаживающими фильтрами, если в сигнале часто встречаются крутые перепады и их нельзя смазывать при сглаживании. То, что такие фильтры сохраняют перепады, равносильно утверждению, что перепады инвариантны к медианной фильтрации. Это — одна из причин, которая заставляет заниматься изучением стабильных точек медианных фильтров. Если стабильная точка имеет не удовлетворяющие нас характеристики, то для того чтобы она не появлялась на выходе медианного фильтра, может оказаться необходимым так модифицировать фильтр, что данная частная последовательность не будет стабильной точкой или что эти нежелательные свойства будут подавляться. Может также случиться, как в рассмотренных ниже примерах, что произвольный входной сигнал при повторной многократной медианной фильтрации будет сходиться к одной из стабильных точек фильтра. Следовательно, изучая стабильные точки, можно достичь хотя бы на качественном, если не количественном, уровне понимания работы медианного фильтра на входных последовательностях общего вида.

Характеристики стабильных точек одномерных медианных фильтров рассматриваются в разд. 6.1. Здесь показано, что эти точки можно разделить на две категории. Стабильные точки первой категории можно рассматривать как некоторые обобщенные монотонные последовательности. Ко второй категории относятся точки более специального вида. Затем результаты, полученные в одномерном случае, переносятся на двумерный, однако эта часть теории еще в значительной степени не завершена. В заключение обсуждается интересный алгоритм Хуанга и других [6.4] двумерной медианной фильтрации.

6.1. Стабильные точки одномерных медианных фильтров

Обозначим одномерный медианный фильтр с шириной апертуры $(2k+1)$ через МФ_{2k+1}, т. е. определим соотношение вход-выход как $\{y_n\} = \text{МФ}_{2k+1}\{x_n\}$, где

$$y_n = \text{медиана}(x_{n-k}, \ldots, x_n, \ldots, x_{n+k}) \text{ для всех } n. \qquad (6.1)$$

Предполагается, что последовательности продолжаются бесконечно в обе стороны. Имеется относительно немного хорошо известных свойств медиан, которые могут быть непосредственно полезны нам. Начнем с рассмотрения двух из них.

Свойство 1. Если $x_{-k} \leqslant \ldots \leqslant x_0 \leqslant \ldots \leqslant x_k$, то

$$\text{медиана}(x_{-k}, \ldots, x_0, \ldots, x_k) = x_0.$$

Свойство 2. Если $g(x)$ монотонна, то

$$\text{медиана}[g(x_1), \ldots, g(x_{2k+1})] = g[\text{медиана}(x_1, \ldots, x_{2k+1})].$$

Для медианного фильтра, определенного по формуле (6.1), из свойства *1* можно сделать вывод, что монотонная последовательность $\{x_n\}$, т. е. последовательность с $x_n \leqslant x_m$ (или $x_n \geqslant x_m$) для всех $n<m$, *инвариантна к медианной фильтрации* с произвольной шириной апертуры. Например, ступенчатая функция является монотонной последовательностью, поэтому она инвариант к медианной фильтрации. Если считать перепад в идеальном случае ступенчатой функцией, то очень просто объяснить, почему после медианной фильтрации перепады сохраняются. Монотонные последовательности являются всего лишь простейшими стабильными точками медианных фильтров. Их обобщение дает более важный класс стабильных точек, который будет обсуждаться в этом разделе. Из свойства *2* вытекает, что, поскольку рассматриваются только медианы, масштаб значений последовательностей не играет роли. В самом деле, исходную последовательность данных $\{x_n\}$ можно свести к бинарной последовательности, без потери какой-либо информации о медианах, путем введения последовательности $\{g_a(x_n)\}$, такой, что для всех $-\infty<a<\infty$, где

$$g_a(x)=\begin{cases}1, & x \geqslant a,\\ 0, & x<a.\end{cases} \tag{6.2}$$

Очевидно, что $\{x_n\}$ является стабильной точкой медианного фильтра МФ_{2n+1} тогда и только тогда, когда $\{g_a(x_n)\}$ является его стабильной точкой для всех a. Это свойство позволяет значительно упростить входную последовательность данных и будет особенно полезно при рассмотрении двумерных последовательностей.

Как указывалось выше, монотонные последовательности — это стабильные точки медианных фильтров с произвольной шириной окна. Однако требование абсолютной монотонности является необязательным. Поскольку медианный фильтр имеет фиксированные и конечные размеры апертуры, интуитивно ясно, что монотонность должна сохраняться только в пределах каждого сегмента, совпадающего по размерам с апертурой. На самом деле это требование можно ослабить еще сильнее.

Определение. Последовательность $\{x_n\}$ является *локально-монотонной* на отрезке m [ЛОМО (m)], если $(x_n, \ldots, x_{n+m-1}$ монотонна для каждого n.

Очевидно, что ЛОМО (m) является также ЛОМО (p), если $p<m$. Допустим, что $\{x_n\}$ есть ЛОМО (m). Тогда последовательности $(x_n, \ldots, x_{n+m-1})$ и $(x_{n+1}, \ldots, x_{n+m})$ монотонны. Если $x_n<x_{n+m-1}$ и $x_{n+1}>x_{n+m}$, то имеем $x_i<x_j$ и $x_i \geqslant x_j$ для всех $n+1 \leqslant i \leqslant n+m-1$, откуда вытекает, что $x_{n+1}=\ldots=x_{n+m-1}$. Следовательно, локально-монотонные последовательности могут быть иначе определены при помощи леммы.

Лемма 6.1. Если имеется какое-нибудь изменение в общем направлении, то ЛОМО (m)-последовательность должна оставаться постоянной хотя бы для $(m-1)$ членов.

Используя функцию уровня $g_a(x)$, определенную ранее, можно легко показать, что последовательность является ЛОМО (m),

если при всех a выходной сигнал остается равным 1 или 0 хотя бы для $(m-1)$ членов.

Получаем теорему, касающуюся локально-монотонных последовательностей и медианной фильтрации.

Теорема 6.1. ЛОМО (m)-последовательность инвариантна к $МФ_{2k+1}$ для всех k, $k \leqslant m-2$.

Доказательство. Рассмотрим сегмент $(x_{n-k}, \ldots, x_{n+k})$, $k=m-2$, который либо монотонен, либо внутри него общее направление меняется, оставаясь постоянным на субсегментах $m-1=k+1$ (см. лемму 6.1). В первом случае медиана элементов этого сегмента, очевидно, равна x_n, во втором субсегмент, имея длину не менее $(k+1)$, должен содержать x. Следовательно, найдется по крайней мере $(k+1)$ элементов в сегменте, равных x_n, и, таким образом, медиана также должна быть равна x_n.

Хотя последовательности ЛОМО $(k+2)$ инвариантны к $МФ_{2k+1}$, обычно они не исчерпывают собой набор стабильных точек $МФ_{2k+1}$. Было установлено, что эти стабильные точки относятся к двум совершенно разным категориям. Последовательности ЛОМО$(k+2)$, рассмотренные выше, образуют первую категорию. Теоремы 6.2 и 6.3 характеризуют эти две категории и показывают различия между ними. Доказательства теорем приведены в приложении 6.А.

Теорема 6.2. Если $\{x_n\}$ является стабильной точкой $МФ_{2k+1}$ и имеется монотонный сегмент $(x_p, x_{p+1}, \ldots, x_{p+k})$ длиной $(k+1)$, тогда $\{x_n\}$ является ЛОМО$(k+2)$.

Эта теорема говорит о том, что если стабильная точка $МФ_{2k+1}$ является достаточно гладкой [т. е. монотонной для сегмента длиной $(k+1)$], то она является гладкой по всей длине [т. е. является ЛОМО$(k+2)$].

Определение. Последовательность *нигде* не является гладкой [назовем такую последовательность НЕЛОМО(k)], если она не содержит монотонного сегмента длиной k.

Теорема 6.3. Если $\{x_n\}$ является стабильной точкой $МФ_{2k+1}$ и если она является НЕЛОМО$(k+1)$, то $\{x_n\}$ — бинарная последовательность, т. е. x_n может принимать только два значения.

Теорема 6.3 представляется несколько неожиданной в теории медианных фильтров, и следствия из нее будут обсуждаться позднее. Для каждого медианного фильтра $МФ_{2k+1}$ будем относить стабильные точки, описываемые теоремой 6.2, к точкам первого типа и точки, описываемые теоремой 6.3, — ко второму типу.

Поскольку любой сегмент, состоящий из двух отсчетов, является монотонным, то по теореме 6.2 любая стабильная точка $МФ_3$ является ЛОМО(3). Для $МФ_3$ не имеется стабильных точек второго типа. Однако для медианных фильтров с размером апертуры больше 3 стабильные точки второго типа существуют. Ниже описан метод получения некоторых из них.

Рассмотрим периодическое продолжение последовательности

$$a_0, a_1, a_2, \ldots, a_k; \qquad -a_0, -a_1, \ldots, -a_k, \tag{6.3}$$

где $a_i = 1$ или -1. Очевидно, что эта последовательность является стабильной точкой $МФ_{2k+1}$. Если $(a_0, \ldots, a_k)$ монотонна, то эта последовательность есть ЛОМО$(k+2)$ и, следовательно, она принадлежит к стабильным точкам первого типа. Если $(a_0, \ldots, a_k)$ не монотонна, то легко увидеть, что эта последовательность есть НЕЛОМО$(k+1)$ и, следовательно, она принадлежит к стабильным точкам второго типа. Этот метод позволяет получить стабильные точки для $МФ_5$ и $МФ_7$, но для $МФ_9$ он не справедлив. Например, периодическое чередование 1, 1, -1, -1 является стабильной точкой второго типа для $МФ_9$, но ее период равен 4, а не 10, как требуется в (6.3). Остается невыясненным, все ли стабильные точки второго типа периодические. Так как стабильные точки второго типа для $МФ_{2k+1}$ двузначны и имеют тенденцию колебаться быстрее [будучи НЕЛОМО$(k+1)$], чем стабильные точки первого типа [ЛОМО$(k+2)$], их можно считать нежелательными с точки зрения сглаживания последовательностей. Действительно, если отрезок последовательности данных является бинарной и быстро колеблющейся последовательностью, нахождение медианы в конечном счете мало что дает. Поэтому интересно рассмотреть те обобщенные медианные фильтры, которые являются сглаживающими, построенными на простых медианах, но, если это нужно, ведут себя при наличии стабильных точек второго типа или бинарных быстро колеблющихся последовательностей как линейные фильтры. Прежде чем мы остановимся на получении таких обобщенных медианных фильтров, приведем теорему, обратную теореме 6.1.

Теорема 6.4. Если последовательность $\{x_n\}$ инвариантна к $МФ_{2p+1}$ для всех $p = 1, 2, \ldots, k$, то она является ЛОМО $(k+2)$.

Доказательство. По теореме 6.2 и с учетом того факта, что любая последовательность является ЛОМО(2), стабильная точка для $МФ_3$, т. е. для $k=1$, должна быть ЛОМО(3). Предположим, что теорема верна для $k-1$. Тогда последовательность $\{x_n\}$, которая инвариантна к $МФ_{2p+1}$ для всех $p = 1, 2, \ldots, k$, должна быть ЛОМО$(k+1)$, следовательно, каждый сегмент длиной $(k+1)$ монотонен. Согласно теореме 6.2 последовательность $\{x_n\}$ является ЛОМО$(k+2)$.

В [6.2] отмечено в частном случае, каковы возможные соотношения между k и m в теореме 6.1. В работе [6.5] также описано соотношение между k и m и, кроме того, показано, что $МФ_{2k+1}$ стремится создать плоские верхушки (или впадины) длиной $(k+1)$, что очень характерно для последовательностей ЛОМО$(k+2)$ (см. лемму 6.1).

До сих пор мы рассматривали только двусторонние бесконечные последовательности; для последовательностей с конечной длиной в [6.6, 6.7] были использованы несколько определений концевых точек. Например, для $МФ_{2k+1}$ мы можем уменьшать размер апертуры на 2 для каждого шага по направлению к концу последовательности, как только центр апертуры окажется только на k отсчетов от ее конца [6.7]. При таком определении теоремы 6.1 и 6.2 остаются справедливыми, однако, теорема 6.3 нуждается в не-

значительном изменении для концевых точек. Но так как это ничего не дает для понимания проблемы, мы не будем на нем останавливаться.

Была замечена очень тесная связь между медианными фильтрами и их стабильными точками. Начнем с рассмотрения этой связи для стабильных точек первого типа. Локально-монотонные последовательности обладают некоторым типом гладкости, выражающейся в монотонности. Возьмем в качестве примера последовательности ЛОМО(m); в пределах сегмента из последовательных m отсчетов не допускается никаких изменений, или, что то же самое, для любого изменения сигнала он должен оставаться постоянным в течение следующих (m—1) отсчетов, а между плоскими участками сигнал является монотонным. Это исключает возможность появления изолированных пиков или выбросов с длительностью, меньшей или равной (m—2). Согласно работе [6.7] они не имеют должной поддержки и медианные фильтры с размером апертуры, большим или равным ($2m$—3), способны устранять их. С другой стороны, разрывы в виде перепадов допускаются независимо от перепада, поскольку сигнал локально-монотонен, и следующий перепад, который, по предположению, будет иметь противоположное направление, не может встретиться в пределах (m—1) отсчетов. Точно так же медианные фильтры с размером апертуры, меньшим или равным ($2m$—3), способны сохранить их. Конечно, не все свойства ЛОМО-последовательностей полезны. Например, плоские участки, которые неизбежны для любого изменения в ЛОМО-последовательностях, могут оказаться нежелательными, а медианная фильтрация имеет тенденцию создавать большое их число [6.6, 6.8]. Так как требуется только локальная монотонность, сигналы такого рода локально не должны быть полиномами малого порядка. Это непосредственно подтверждается медианной фильтрацией зашумленного пилообразного сигнала. Выходной сигнал фильтра будет больше напоминать лестницу с регулярно расположенными ступеньками, чем пилообразную кривую. Интуитивно понятно, что медианной фильтрацией можно восстановить только монотонность, но не линейность или другие свойства полинома низкого порядка, присущие сигналу [6.7, 6.9]. В этом случае можно рекомендовать после медианной фильтрации использовать симметричные линейные сглаживающие фильтры с малой апертурой. Малые апертуры выбираются во избежание искажений на перепадах. Что касается плоских участков, образованных при медианной фильтрации, то искаженный сигнал может быть частично восстановлен с помощью специальной процедуры огрубления [6.6—6.8].

Мы обсудили преимущества и недостатки медианной фильтрации, исходя из характеристик локально-монотонных последовательностей, которые являются также стабильными точками первого типа. Теперь рассмотрим стабильные точки второго типа. Представляется, что стабильные точки второго типа — это последовательности фиксированной формы, которые редко встречаются

в отрезках реальных сигналов. Однако возможно, что части последовательности данных быстро флуктуируют, принимая два значения. Если требуется восстановить взвешенное среднее каждой части, то медианные фильтры вряд ли будут пригодны. Здесь опять после медианной фильтрации можно использовать симметричные линейные сглаживающие фильтры с малой апертурой. Однако нас больше интересуют нелинейные сглаживающие фильтры, которые строятся на простых медианных фильтрах и не имеют таких стабильных точек второго типа. Основная причина интереса к нелинейным сглаживающим фильтрам состоит в том, что они сохраняют перепады. Этому посвящен разд. 6.2.

6.2. Некоторые обобщенные медианные фильтры

Существует другой класс последовательностей, связанных с медианными фильтрами, которые нам следует изучить. Эти последовательности можно назвать *рекуррентными точками (последовательностями)*. В общем, для нелинейного сглаживающего фильтра T последовательность *рекуррентна*, если она является стабильной точкой T^m при некотором $m \geqslant 2$, но не для T. Здесь T^m означает m-кратное применение T. Например, знакопеременная (осциллирующая) последовательность 1, —1, 1, —1, ... не является стабильной точкой МФ_3, тем не менее это — стабильная точка МФ^2_3. В целом нам известно очень немного о рекуррентных последовательностях медианных фильтров. Несколько найденных нами примеров дают возможность предположить, что они должны быть бинарными, флуктуирующими и иметь фиксированную форму. Однако у нас нет математического доказательства правильности этих наблюдений. Если существование стабильных точек второго типа делает медианные фильтры бесполезными в качестве сглаживающих, то следует искать альтернативные пути, свободные от этих точек. Точно так же хотелось бы избежать и рекуррентных точек. В этом направлении получено не очень много результатов и некоторые из них, касающиеся МФ_3, даны ниже без доказательства. Детали можно найти в приложении 6.Б.

Теорема 6.5. Пусть $a_0 \neq 1$, $a_k \geqslant 0$ и $\sum_{k=0}^{n} a_k = 1$. Тогда стабильными точками сглаживающего фильтра

$$T = \sum_{k=0}^{n} a_k \text{МФ}_3^k \tag{6.4}$$

являются только ЛОМО(3)-последовательности при $a_k \neq 0$ для некоторых нечетных k. Если $a_k = 0$ для всех нечетных k, то стабильными точками T будут и последовательности ЛОМО(3), и знакопеременные последовательности. Для $m \geqslant 2$ стабильной точкой T^m также должна быть ЛОМО(3) или знакопеременные последовательности. Последние не могут быть инвариантны к T^m, если только не справедливо, что: а) $a_k = 0$ для всех четных k и m или 2) $a_k = 0$ для всех нечетных k.

С учетом приведенной теоремы сделать, чтобы знакопеременная последовательность не была ни стабильной, ни рекуррентной точкой, можно только, если рассматривать сглаживающий фильтр T в (6.4) для, по крайней мере, четных k и нечетных j, причем a_k и a_j должны быть положительны. Например, вместо простого $МФ_3$ можно использовать $T_1=\beta I+(1-\beta)МФ_3$, где $0<\beta<1$, I — тождественный оператор, или $T_2=МФ_3+(1-\beta)МФ^2{}_3=МФ_3*T_1$. В самом деле, оба они свободны от стабильных точек второго типа и рекуррентных точек. Далее можно показать, что верны следующие теоремы.

Теорема 6.6. Для $T=\beta I+(1-\beta)МФ_3$, $0<\beta<1$, последовательность $T^m\{x_n\}$ сходится поточечно к ЛОМО(3)-последовательностям при $m\to\infty$.

Знакопеременные последовательности являются рекуррентными точками $МФ_3$ и стабильными точками второго типа для $МФ_5$, однако с помощью линейной комбинации $МФ_3$ и $МФ_5$ можно от них избавиться.

Теорема 6.7. Пусть $T=\beta МФ_3+(1-\beta)МФ_5$, $0<\beta<1$. Тогда $\{x_n\}$ является стабильной точкой T, если и только если она есть ЛОМО(4).

Нетрудно показать, что стабильными точками сглаживающих фильтров, построенных посредством повторения выпуклых комбинаций или соединения [т. е. $\beta T_1+(1-\beta)T_2$ или T_1*T_2] медианных фильтров с апертурой, меньшей или равной $(2k+1)$, являются последовательности ЛОМО$(k+2)$. Но установить, имеют ли они другие стабильные точки, в особенности второго типа, достаточно сложно.

Для более полного понимания детерминированных свойств медианных фильтров или их обобщений, очевидно, необходимо пойти дальше простой теории стабильных точек. В [6.5] при изучении устойчивости медианных и связанных с ними нелинейных фильтров применялся иной подход к изучению сглаживающих свойств этих фильтров. Исследованные сигналы имели вид чистой синусоиды. Сначала брались отсчеты чистой синусоиды с нулевой фазой. Затем после выполнения медианной или другой нелинейной фильтрации вычислялись: мощность или амплитуда основной гармоники (которая имела частоту входного сигнала), а также нескольких первых гармоник (или их смеси). Тем самым можно определить мощность, пропускаемую фильтром на частоте входного сигнала, а также мощность, перешедшую к каждой из ее гармоник. Подобно передаточной функции по мощности линейной системы, для рассматриваемого нелинейного фильтра можно также построить график, отражающий часть мощности, переданной на входной частоте при чистой синусоиде с нулевой фазой на входе.

Хотя принцип суперпозиции к нелинейным фильтрам не применим, определение переданной мощности и мощности, перешедшей в гармоники, все-таки дает важную информацию о поведении и свойствах каждого нелинейного фильтра. Например, численные результаты в [6.5] показывают, что передаточные функции меди-

анных фильтров с апертурой, размеры которой — нечетные числа, имеют довольно большие боковые лепестки. Так, при частоте дискретизации 128 отсчет/с для $МФ_5$ возникают большие боковые лепестки на частотах 32 и 64 Гц. Интересно, что это явление тесно связано с стабильными и рекуррентными точками $МФ_5$. Рассмотрим это подробнее.

Пусть выходной сигнал x_n — дискретная синусоида с частотой f и фазой φ. Частота дискретизации равна 128 отсчет/с. Имеем

$$x_n = \sin\left(\frac{2\pi fn}{128} + \varphi\right).$$

При $f=32$ Гц

$$x_n = \begin{cases} (-1)^{n/2}\sin\varphi, & n \text{ — четное}, \\ (-1)^{(n-1)/2}\cos\varphi, & n \text{ — нечетное}. \end{cases}$$

Без потери общности предположим, что $|\varphi| \leqslant \pi/4$. Выходной сигнал y_n $МФ_5$

$$y_n = \begin{cases} -\sqrt{2}\sin\varphi\sin(\pi n/2 + \pi/4), & 0 \leqslant \varphi \leqslant \pi/4, \\ \sqrt{2}\sin\varphi\sin(\pi n/2 - \pi/4), & -\pi/4 \leqslant \varphi \leqslant 0. \end{cases}$$

Таким образом, амплитуда, переданная на частоте $f=32$ Гц, равна $|\sqrt{2}\sin\varphi|$ при $|\varphi| \leqslant \pi/4$. Введя случайную фазу с равномерным распределением, получим, что средняя переданная мощность равна

$$\frac{4}{\pi}\int_0^{\pi/4} 2(\sin\varphi)^2\, d\varphi = 0{,}363,$$

что дает значительный (—4,4 дБ) боковой лепесток в передаточной функции по мощности.

Внимательное рассмотрение выходного сигнала $\{y_n\}$ показывает, что это рекуррентная последовательность $\ldots, a, a, -a, -a, \ldots$ $\ldots, МФ_5$, **или стабильная точка** $МФ^2_5$. **Действительно, входной** сигнал $\{x_n\}$ сам по себе является рекуррентной последовательностью при условии, что $\varphi = \pm\pi/4$. Поэтому совершенно ясно, почему на этой частоте будет возникать большой боковой лепесток.

Аналогично, при $f=64$ Гц входной сигнал $x_n = \sin(\pi n + \varphi) = (-1)^n\sin\varphi$, т. е. он является знакопеременной последовательностью и потому будет стабильной точкой второго типа для $МФ_5$. Ясно, что $\{y_n\} = \{x_n\}$ и передаточная функция имеет на этой частоте пик, равный 1.

Для устранения нежелательных боковых лепестков было предложено использовать $МФ_4$ после $МФ_2$, т. е. $МФ_4 * МФ_2$. Здесь **выходной** сигнал медианного фильтра с апертурой, размеры которой являются четными числами, определяется выражениями:

$$\{y_{n+1/2}\} = МФ_{2k}\{x_n\}, \quad y_{n+1/2} = \text{медиана}\,(x_{n-k+1}, \ldots, x_{n+1}),$$

где медиана четного числа отсчетов есть среднее двух центральных отсчетов. Следовательно, выходной сигнал y_n $МФ_4 * МФ_2$ равен

$$y_n = \frac{1}{2} \text{медиана } (x_{n-1}, x_n, x_{n+1}, x_{n+2}) + \\ + \frac{1}{2} \text{медиана } (x_{n-2}, x_{n-1}, x_n, x_{n+1}),$$

что также покрывает пять соседних отсчетов. В [6.5] показано, что передаточная функция по мощности такого составного фильтра имеет лепесток, равный —13,3 дБ вблизи $f=43$ Гц. С точки зрения сглаживания он работает гораздо лучше, чем простой фильтр $МФ_4$, Однако при этом ступенчатые функции уже не сохраняются. Фактически $МФ_4$ в значительной мере лишен свойства медианных фильтров сохранять перепады; как предложено в [6.5], $МФ_4$ следует рассматривать, скорее, как на 25% сглаживающий усреднением фильтр, а не как представитель обобщенных медианных фильтров. Кроме того, $МФ_2$ — это фильтр, усредняющий каждые два соседних отсчета.

С другой стороны, в соответствии с нашей теорией стабильных точек, можно рассматривать среднее от $МФ_3$ и $МФ_5$, т. е. $0{,}5МФ_3 + 0{,}5МФ_5$.

Для синусоиды с частотой $f=32$ Гц можно легко проверить, что $МФ_3\{x_n\} = -МФ_5\{x_n\}$ для произвольного φ, т. е. при $f=32$ Гц передаточная функция этого фильтра проходит через нуль. Аналогично, при $f=64$ Гц входной сигнал, который является знакопеременной последовательностью, есть стабильная точка $МФ_5$ и рекуррентная точка $МФ_3$; находя среднее между ними, мы опять-таки получаем нуль. Так как фильтр $0{,}5МФ_3 + 0{,}5МФ_5$ не имеет других стабильных точек, кроме последовательности ЛОМО(4), для нас неважно, что он имеет некоторую рекуррентную точку, поэтому можно предположить, что боковые лепестки его передаточной функции будут гораздо меньше, чем у $МФ_3$ и $МФ_5$. Действительно, наш численный эксперимент это подтверждает. Он показывает, что максимальное значение бокового лепестка составляет 13 дБ при $f=51$ Гц, а следующее наибольшее значение равно —23,3 дБ при $f=37$ Гц.

6.3. Стабильные точки двумерных медианных фильтров

Чтобы распространить приведенные результаты, полученные для стабильных точек одномерных медианных фильтров, на их двумерные аналоги, представляющие большой практический интерес в обработке изображений, естественно, следует попытаться отыскать те характеристики, которые отличают один тип стабильных точек от другого. Такая задача гораздо сложнее, чем та, которая решалась выше. Для понимания этого достаточно сравнить используемые апертуры фильтров. В одномерном случае, при смещении апертуры на один шаг, вводится только один новый отсчет

и пропадает один старый. Следовательно, многообразие структур стабильных точек одномерных медианных фильтров сильно ограничено. Хотя мы не знаем всех свойств стабильных точек второго типа и не знаем, как их получить и еще меньше нам известно о рекуррентных точках обобщенных медианных фильтров, мы высказали весьма полезные соображения относительно их общих характеристик. Кроме того, когда апертура двумерного медианного фильтра) которая не вырождается в линейный сегмент) сдвигается на один шаг, то вводится или пропадает более одного отсчета. Интуитивно ясно, что это расширяет наши возможности и стабильные точки двумерных медианных фильтров будут значительно сложнее или менее структурированы, чем их одномерные аналоги. Примеры стабильных точек показывают, что они могут сильно напоминать стабильные точки второго типа в одной области, оставаясь локально-монотонными в другой; для одномерного случая это неверно. Мы еще вернемся к этому.

В гл. 5 показано, что изображения с перепадами сохраняются после двумерной медианной фильтрации, если апертура симметрична и имеет центр. Благодаря своей практической важности ранее рассматривались только апертуры этого типа, и именно такие апертуры будут подразумеваться в этой главе. Изображение перепада напоминает ступенчатую функцию в одномерном случае; в обоих случаях это простейшие монотонные функции. Как указано в начале разд. 6.1, локальной монотонности достаточно, чтобы сделать последовательность стабильной точкой. По аналогии можно ожидать, что изображение будет инвариантно медианной фильтрации с апертурой A до тех пор, пока оно остается монотонным в пределах апертуры, когда ее центр передвигается от одного элемента изображения к другому. Необходимо точное определение *монотонности*, которое будет дано позднее. Следуя гипотезе, изложенной в [6.10], мы сначала разложим двумерную апертуру на строки и предположим, что то, что требуется от всей апертуры, истинно также для каждой строки. Будем считать, что апертура содержит начало координат (0,0) и симметрична относительно него.

Лемма 6.2. Пусть A — апертура и L — произвольная линия в R^2. Если

$$\text{медиана}\,(x_{i,j} | (i,j) \in L \cap A) = x_{0,0} \tag{6.5}$$

для всех L, проходящих через начало координат, то медиана $(x_{i,j} | (i,j) \in A) = x_{0,0}$.

Доказательство. Из предположения, что A включает начало координат $(0,0)$ и симметрична относительно него, имеем, что $L \cap A$ должна содержать нечетное число точек и, чтобы выполнялось (6.5), половина отсчетов, не считая $x_{0,0}$, должна быть больше или равна $x_{0,0}$, а другая половина — меньше или равна $x_{0,0}$. Так как все линии, проходящие через начало координат, не пересекаются нигде, кроме начала координат, то, исключая точку $(0,0)$, полови-

на отсчетов в A должна быть больше или равна $x_{0,0}$, а другая — меньше или равна $x_{0,0}$.

Будем говорить, что для апертуры A изображение $\{x_{i,j}\}$ *локально-монотонно по отношению к A*, если для всех сдвигов (r, s) изображения относительно апертуры и для всех линий L, которые проходят через точку $(0, 0)$, центр апертуры, $\{x_{i+r, j+s}\}$ монотонна в пределах отрезков линий, попадающих в апертуру.

Лемма 6.3. Если изображение $\{x_{i,j}\}$ является локально-монотонным по отношению к апертуре A, то оно является стабильной точкой медианного фильтра с апертурой, равной A или являющейся подмножеством A.

Доказательство. Предположим, что для любого (r, s) $\{x_{i+r, j+s}\}$ монотонно на $L \cap A$ для всех L, проходящих через начало координат; следовательно,

$$\text{медиана } (x_{i+r, j+s} \mid (i, j) \in A \cap L) = x_{r,s}.$$

Тогда доказательство следует просто из леммы 6.2.

Для ослабления требования локальной монотонности в приведенной лемме мы можем рассмотреть следующий класс апертур.

Определение. Апертура A является p-симметричной, если в дополнение к тому, что она содержит начало координат $(0, 0)$ и симметрична относительно него, она содержит все точки (r, s) пересечения $\mathbb{Z}^2$ и конечного линейного сегмента $\theta(i, j)$, $0<\theta<1$, для всех (i, j) в A, где θ (i, j) — линейный сегмент, соединяющий точки $(0, 0)$ и (i, j).

Теперь допустим, что L является произвольной линией в $\mathbb{Z}^2$, а A—p-симметричная апертура. Тогда точки, содержащиеся в L, будут расположены периодически. Для точки (r, s) на L обозначим через $N_{L,A}$ число точек, содержащихся в $L \cap \{A + (r, s)\}$, где $\{A + (r, s)\}$ означает апертуру A, смещенную так, что ее центр находится в точке (r, s). В результате периодичности $\mathbb{Z}^2$ число $N_{L,A}$ не зависит от выбранного (r, s) и является нечетным вследствие симметричности апертуры A. На самом деле, все линии в $\mathbb{Z}^2$, параллельные L, имеют одно и то же $N_{L,A}$. Теорема которая распространяет теорему 6.1 на двумерный случай, является обобщением результата, полученного в [6.10] путем использования идеи локально-монотонных последовательностей.

Теорема 6.8. Пусть A является p-симметричной апертурой, а $\{x_{i,j}\}$ — изображение. Если для каждой линии L отсчеты $x_{i,j}$ на ней локально-монотонны на длине $(N_{L,A}+3)/2$, то $\{x_{i,j}\}$ инвариантно к медианной фильтрации с p-симметричной апертурой, равной A или являющейся подмножеством A.

Доказательство. Так как отсчеты на линии L, проходящей через произвольную точку (r, s), локально-монотонны на длине $(N_{L,A}+3)/2$, то по теореме 6.1 они инвариантны к одномерной медианной фильтрации с апертурой, размер которой меньше или равен $N_{L,A}$. Для любой p-симметричной апертуры B, аналогичной апертуре A или являющейся ее подмножеством, $N_{L,B} \leqslant N_{L,A}$, а пе-

ресечение $L \cap \{B+(r, s)\}$ будет подсегментом из последовательно расположенных точек на L с центром в (r, s). Следовательно,

$$\text{медиана } (x_{i+r, j+s} | (i, j) \in B), \qquad (i+r, j+s) \in L) = x_{r,s}.$$

В силу леммы 6.2 теорема доказана.

Аналогично теореме 6.4 для данной теоремы также существует обратная. Доказательство ее не отличается от доказательства теоремы 6.4 и потому опускается.

Теорема 6.9. Если изображение $\{x_{i,j}\}$ инвариантно к любой медианной фильтрации с p-симметричной апертурой, которая аналогична A или является ее подмножеством, где A p-симметрична, то отсчеты $x_{i,j}$ на любой линии локально-монотонны на длине $(N_{i,A}+3)/2$.

Если изображение инвариантно к любой медианной фильтрации с p-симметричной апертурой, то согласно теореме 6.9 ее пересечение с любой линией L из $\mathbb{Z}^2$ должно быть монотонным. Изображения, обладающие этим свойством, называются *монотонными во всех направлениях* [6.10] и, как оказывается, имеют простую структуру. Юстуссоном был обнаружен следующий результат [6.10].

Если $\{x_{i,j}\}$ монотонно во всех направлениях, то существует наклон b, такой, что $x_{i,j} \geqslant x_{r,s}$ для всех $j-bi > s-br$ или $x_{i,j} \leqslant x_{r,s}$ для всех $j-bi > s-br$. В случае $b = \pm\infty$ неравенство $j-bi > s-br$ заменяется более простым неравенством $i > r$. Наклон b единствен, если только $\{x_{i,j}\}$ не является константой.

Схема доказательства приведена ниже. Рассмотрим функцию уровня $g_c(x)$ из (6.2) и изображение $\{g_c(x_{i,j})\}$. Пусть $S_{1,c} = \{(i, j) \mid g_c(x_{i,j}) = 1\}$, а $S_{0,i}$ является дополнением $S_{1,c}$ к $\mathbb{Z}^2$. Очевидно, что $S_{1,c} \supset S_{1,d}$ для $c < d$. Допустим, что $S_{1,c}$ и $S_{0,c}$ — непустые множества. Тогда, предполагая, что $\{x_{i,j}\}$ монотонно во всех направлениях, можно показать, что оба они выпуклы, т. е. любая точка в $\mathbb{Z}^2$, которая является выпуклой комбинацией точек из $S_{1,c}$, находится также в $S_{1,c}$. Далее, они являются исключением в том смысле, что минимальные выпуклые множества в $\mathbb{R}^2$, обозначенные $C(S_{1,c})$ и $C(S_{0,c})$, от $S_{1,c}$ и $S_{0,c}$, соответственно, являются исключением. Объединение $C(S_{1,c}) \cup C(S_{0,c})$ может не быть равным $\mathbb{R}^2$. Тогда по принципу разделения гиперплоскостей имеется прямая $y = a+bx$, которая разделяет $C(S_{1,c})$ и $C(S_{0,c})$. [$C(S_{1,c})$ и $C(S_{0,c})$ оба могут иметь точки на прямой $y = a+bx$]. Наклон единствен, даже если разделяющая прямая не существует. Ясно, что $x_{i,j} \geqslant c$ для всех $j-bi > a$ и $x_{i,j} < c$ для всех $j-bi < a$, или наоборот. Так как $S_{1,c} \supset S_{1,d}$ при условии, что $c < d$, можно показать, что b — единственно для всех c, если только $S_{1,c}$ или $S_{0,c}$ не пусты. Доказательство завершается рассмотрением всех $g_c(x)$. В соответствии с полученным результатом, если линия L не параллельна $y = bx$ и если $x_{ik,jk} = c$ для всех $k = 1, 2, 3$, где (i_k, j_k) — три последовательные точки на L, то $x_{i,j} = c$ для всех (i, j) на линии L^*, которая параллельна прямой $y = bx$ и пересекает L в точке (i_2, j_2).

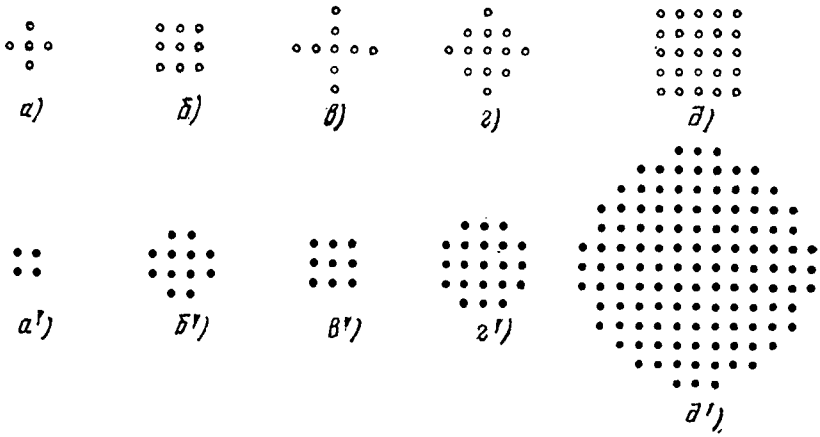


Рис. 6.1. Некоторые двумерные апертуры ($а$--$д$) и соответствующие им стабильные точки [($а'$)—($д'$)]

Достаточные условия, приведенные в лемме 6.3 или в теореме 6.8, обычно трудно выполнить, особенно для больших апертур. На рис. 6.1 показаны некоторые двоичные изображения, которые для нескольких p-симметричных апертур удовлетворяют достаточному условию теоремы 6.8 (оно является менее жестким, чем условие леммы 6.3). Эти двоичные изображения могут рассматриваться либо как однородные объекты на однородном белом фоне, либо как выходные сигналы функций уровня $g_a(\cdot)$. Они строятся следующим образом.

Для каждой апертуры изучим те направленные линейные сегменты (спицы), которые начинаются в центре (основании) и кончаются в точке на границе (верхушке). Каждый из этих линейных сегментов можно определить, задав его угол θ, $0° \leqslant \theta \leqslant 360°$, с сегментом, соединяющим центр, допустим, точку $(0, 0)$ и точку $(1, 0)$. Тогда можно легко увидеть, что каждый объект имеет границу, которая является кусочно-линейной и образована этими линейными сегментами, расположенными в порядке возрастания или уменьшения θ, причем основание каждого связано с верхушкой предыдущего. Поэтому объект является выпуклой группой в $\mathbb{Z}^2$ с указанной границей. Не все двоичные изображения, полученные этим способом, являются локально-монотонными в смысле теоремы 6.8. Например, если используемая апертура имеет форму креста, состоящего из пяти точек $(0,0)$ и (i, j), где $|i| = |j| = 1$, то объект, построенный таким образом, имеет граничные точки $(1,0)$, $(0,1)$, $(—1.0)$, $(0—1)$ и локальная монотонность в смысле теоремы 6.8 в центре объекта не сохраняется. Однако она, по-видимому, сохраняется, если апертура A удовлетворяет следующему условию:

$$\lim_{n \leftarrow \infty} A_n = \mathbb{Z}^2,$$

где $A_{n+1}=A_n+A=\{x+y|x\in A_n y\in A\}$ и $A_1=A$. Мы можем считать, что апертура A *вырождается*, если она не удовлетворяет этому условию, поскольку двумерная область $\mathbb{Z}^2$ может быть разложена на несвязные участки, которые в совокупности составляют A^*, где $A^*=\lim_{n\to\infty} A_n$, и медианная фильтрация с вырожденной апертурой A может быть выполнена также путем раздельной фильтрации по каждой из составных частей A^*. Взаимосвязь между составными частями A^* полностью отсутствует.

Возвращаясь к рис. 6.1, можно сделать следующие наблюдения. Во-первых, ни одна из апертур не является вырожденной. Во-вторых, каждый объект является наименьшим *конечным* выпуклым объектом, который сохраняется при медианной фильтрации с соответствующей апертурой. Даже несмотря на поставленное ограничение, что каждая линия L должна быть локально-монотонной и инвариантной к $\text{МФ}_{N_{L,A}}$, оказывается, что в случае *конечного* выпуклого объекта достаточное условие теоремы 6.8 является также необходимым для того, чтобы объект сохранялся при медианной фильтрации. Наконец, по аналогии с тем, как мы построили границу каждого выпуклого объекта, можно было бы предположить, что, в общем, для того, чтобы двоичное изображение гладких объектов на равномерном фоне было инвариантным к медианной фильтрации с невырожденной p-симметричной апертурой, будет необходимым и достаточным следующее условие: *выпуклая* часть границ объектов или фона должна складываться из хорд длиной, большей или равной размерам апертуры, и соответствующие хорды любых двух связанных сегментов контуров должны также являться смежными хордами апертуры. Это наложило бы некоторые ограничения на контуры сохраняемых изображений. К сожалению, такое условие не является ни необходимым, ни достаточным. В самом деле, определение таких понятий, как граница или линейные сегменты границы для двоичных изображений общего вида, является достаточно широким.

В начале этой гравы было упомянуто, что в отличие от стабильных точек МФ_{2k+1}, которые являются либо ЛОМО$(k+2)$, либо НЕЛОМО$(k+1)$, стабильные точки двумерных медианных фильтров могут быть смешанными. Некоторые примеры показаны на рис. 6.2. Рассмотрим периодическое продолжение изображений, показанных на рис. 6.2,*а*—*в*. Очевидно, они аналогичны во всем,

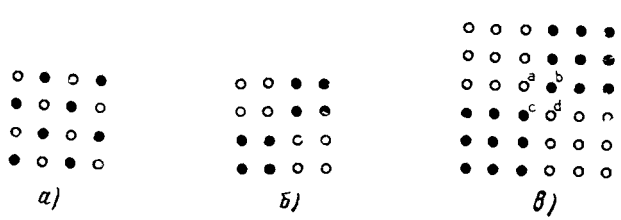


Рис. 6.2. Примеры стабильных точек для квадратной апертуры 3×3

кроме масштаба, и все три инвариантны к медианному фильтру с квадратной апертурой 3×3, обозначенной через A. Если о гладкости судить по локальной монотонности в квадратной апертуре 3×3 (см. определение после леммы 6.2), то a и $б$ нигде не являются локально-монотонными по отношению к A. Изображение b локально-монотонно по отношению к A везде, кроме тех седловых точек типа a, b, c и d, где это неверно. При еще меньшем масштабе изображение, которое остается стабильной точкой медианного фильтра с апертурой A, становится более гладким; однако указанные седловые точки также сохраняются. Следовательно, изображение на рис. 6.2,$в$ можно рассматривать как стабильную точку смешанного типа.

6.4. Алгоритм быстрой медианной фильтрации

Интересный и эффективный алгоритм двумерной медианной фильтрации с произвольной апертурой был предложен Хуангом и другими [6.4]. Используется тот факт, что при смещении на один отсчет убирается только часть отсчетов, содержащихся в апертуре, и столько же отсчетов добавляется. Этот раздел основывается на [6.4].

Хотя алгоритм работает для произвольных апертур, мы используем в качестве примера прямоугольную апертуру $m \times n$, где m — число столбцов. Предполагается, что апертура перемещается слева направо по горизонтали и возвращается назад при переходе к следующей строке. Пограничные точки не рассматриваются; их можно обрабатывать, уменьшая размеры апертуры, как и в одномерном случае. Изображение квантовано, скажем, на 256 уровней. Тогда для элементов изображения первого (начального) положения апертуры вычисляются и с каждым следующим шагом вправо корректируются: 1) ГИСТ — гистограмма распределения значений; 2) МДН — медиана, 3) МЧМ — число элементов изображения, значения которых меньше МДН.

При движении апертуры направо на один шаг каждый элемент изображения левого крайнего столбца g предыдущей апертуры убирается; гистограмма и счетчик, МЧМ, корректируются следующим образом:

$$\begin{aligned} &\text{ГИСТ}[g] \leftarrow \text{ГИСТ}[g] - 1, \\ &\text{МЧМ} \leftarrow \text{МЧМ} - 1, \text{ если } g < \text{МДН}. \end{aligned} \tag{6.6}$$

Аналогично при сдвиге на один шаг добавляется каждый элемент изображения g крайнего правого столбца в апертуре. ГИСТ и МЧМ должны быть заменены соответственно на

$$\begin{aligned} &\text{ГИСТ}[g] \leftarrow \text{ГИСТ}[g] + 1, \\ &\text{МЧМ} \leftarrow \text{МЧМ} + 1, \text{ если } g < \text{МДН}. \end{aligned} \tag{6.7}$$

После этого ГИСТ дает гистограмму для текущего положения апертуры, а счетчик МЧМ будет содержать число элементов в дан-

ной апертуре, имеющих значения, меньшие, чем медиана при предыдущем положении апертуры.

Затем медиана в данном положении апертуры находится путем уменьшения (или увеличения) МДН в зависимости от того, превышает МЧМ отношение $(mn-1)/2$, которое мы обозначим через П, или нет. Сначала

$$\text{МЧМ сравнивается с П.} \tag{6.8}$$

Возможны такие случаи.

Случай 1. МЧМ$>$П, что указывает на то, что МДН больше медианы в данном положении апертуры. Тогда вводим коррекцию

$$\text{МДН} \leftarrow \text{МДН} - 1$$

$$\text{МЧМ} \leftarrow \text{МЧМ} - \text{ГИСТ [МДН]},$$

пока не получим

$$\text{МЧМ} \leqslant \text{П}. \tag{6.9}$$

Случай 2. МЧМ$\leqslant$*П*, что указывает на то, что МДН меньше или равна медиане в данном положении апертуры. Проверим неравенство

$$\text{МЧМ} + \text{ГИСТ [МДН]} \leqslant \text{П}. \tag{6.10}$$

Если оно не справедливо, то МДН точно является требуемой медианой. Если оно справедливо, что свидетельствует о том, что МДН$+1$ все еще меньше или равна требуемой медиане, то вводим коррекцию

$$\text{МЧМ} \leftarrow \text{МЧМ} + \text{ГИСТ [МДН]}$$

$$\text{МДН} \leftarrow \text{МДН} + 1$$

и возвращаемся к (6.10).

Очевидно, каждая из операций (6.6) и (6.7) требует n сравнений. Пусть d равна медиане на предыдущей апертуре. Для случая *1* $d<0$ (6.8) и (6.9) требуют $(1+|d|)$ сравнений; для случая *2* $d\geqslant 0$ (6.8) и (6.10) требуют $(2+d)$ сравнений. Пусть $P_i = \text{Prob}(d=i)$. Тогда математическое ожидание будет равно

$$\bar{c} = 2n + \sum_{i<0} (1+|i|) P_i + \sum_{i\geqslant 0} (2+i) P_i = 2n +$$

$$+ |\bar{d}| + 1 + \text{Prob}(d \geqslant 0),$$

где $|d|$ — среднее значение $|d|$. Считая, что $\text{Prob}(d>0) = \text{Prob}(d<0)$, имеем $\bar{c} = 2n + |\bar{d}| + 1{,}5 + 0{,}5P_0$. В [6.4] отмечено, что $|\bar{d}|$ обычно мало, и экспериментальные результаты показывают, что оно меньше 10. Поскольку $|\bar{d}|$ мало, возможна значительная экономия времени вычислений даже при малых $n \times m$. При увеличении размеров апертуры преимущества этого алгоритма по сравнению с теми, которые не используют общности $n(m-2)$ элементов в рассматриваемой апертуре с предшествующим элементом, становится все более заметными.

6.5. Выводы

В этой главе были представлены некоторые новые результаты, касающиеся свойств медианных фильтров и прежде всего их стабильных точек и соотношений между стабильными точками и экспериментальными наблюдениями, о которых ранее сообщали другие исследователи. В заключение приведен интересный алгоритм Хуанга и других.

По определению, стабильные точки медианных фильтров — это последовательности, не меняющиеся под их действием. Для последовательностей, которые не являются ни стабильными, ни рекуррентными точками, эффект медианной фильтрации без надлежащего статистического исследования может быть определен только качественно. Если судить о свойствах медианного фильтра или обобщенного медианного фильтра только по стабильным или рекуррентным точкам, можно впасть в заблуждение. Дополнительная информация по этому вопросу приведена в гл. 5.

Стабильные точки второго типа, а также рекуррентные точки двумерных медианных фильтров не изучены; на рис. 6.2 даны только несколько примеров. Неясно, могут ли они создать какие-либо трудности при обработке изображений реального мира. Если да, то их можно преодолеть с помощью взвешенной медианной фильтрации, которая, как было показано в разд. 6.2, эффективна в одномерном случае.

Приложение 6.А.

Докажем теоремы 6.2 и 6.3. Нам нужна лемма.

Лемма 6.А.1. Пусть $n<m$ и пусть $x_n<x_i<x_m$ для всех $n<i<m$. Если медиана $(x_{n-p}, \ldots, x_{n+p})=x_n$ и медиана $(x_{m-q}, \ldots, x_{m+q})=x_m$, где $n-p\leqslant m-q$ и $n+p\leqslant m+q$, то $x_j\geqslant x_m$ или $x_j\leqslant x_n$ для каждого j, где $n-p\leqslant j<n$ или $m<j\leqslant m+q$. Далее, $x_j\leqslant x_n$ для всех $n-p\leqslant j<\min(m-q, n)$ и $x_j\geqslant x_m$ для всех $\max(m, n+p)<j\leqslant m+q$.

Доказательство. Если медиана $(x_{n-p}, \ldots, x_{n+p})=x$, то среди отсчетов сегмента имеется, по крайней мере, p отсчетов, исключая x_n, которые меньше или равны x_n. Аналогично среди отсчетов $(x_{m-q}, \ldots, x_{m+q})$ существуют, по крайней мере, q отсчетов, отличных от x_m, которые будут больше или равны x_m. Так как $x_n<x_i<x_m$ для всех $n<i<m$ и так как допускается, что $n-p\leqslant m-q$ и $n+p\leqslant m+q$, мы имеем, по крайней мере, q отсчетов, больших или равных x_m, и, по крайней мере, p отсчетов, меньших или равных x_n, в сегментах $(x_{n-p}, \ldots, x_{n-1})$ и $(x_{m+1}, \ldots, x_{m+q})$, которые при объединении содержат только $(p+q)$ отсчетов. Поэтому среди $(p+q)$ отсчетов найдется точно p отсчетов, меньших и равных x_n, и q отсчетов, больших и равных x_m. Это доказывает первую часть леммы. Чтобы доказать ее вторую часть, достаточно заметить, что если существует j, где $n-p\leqslant j<\min(m-q, n)$ такое, что $x_j\geqslant x_m$, тогда, как вытекает из первой части леммы, исключая x_m среди отсчетов $(x_{m-q}, \ldots, x_{m+q})$, имеется максимум $(q-1)$ отсчетов, которые больше и равны x_m. Это противоречит допущению, что $x_m=$медиана $(x_{m-q}, \ldots, x_{m+q})$.

Доказательство теоремы 6.2. Допустим, что сегмент $(x_{n-k}, \ldots, x_n)$, который содержит $(k+1)$ отсчетов, является монотонно-возрастающим, т. е. $x_{n-k}<x_n$; в противном случае $(x_{n-k}, \ldots, x_{n+1})$ монотонен независимо от x_{n+1}, если $x_{n-k}=x_n$. Если $x_{n+1}\geqslant x_n$, то сегмент $(x_{n-k}, \ldots, x_{n+1})$, который содержит $(k+2)$ отсчетов, монотонен и мы можем перейти к сегменту $(x_{n-k+1}, \ldots, x_{n+1})$, который также монотонен, и проверить x_{n+2}. Если $x_{n+1}<x_n$, то согласно лемме 6.А.1 должно быть $x_{n-k}\geqslant x_n$, что противоречит допущению $x_{n-k}<x_n$; следовательно, должно быть $x_{n-k+1}\geqslant x_n$. Те же соображения можно высказать и относительно x_{n-k-1}, чтобы показать, что $x_{n-k-1}\leqslant x_{n-k}$, и, следовательно, сегмент $(x_{n-k-1}, \ldots, x_n)$ также монотонен. Доказательство завершается применением этих же рассуждений к x_{n+2}, x_{n+3}, ... и x_{n-k-2}, x_{n-k-3}, ...

Доказательство теоремы 6.3. Предположим, что последовательность $\{x_n\}$ — стабильная точка $\mathrm{M}\Phi_{2k+1}$ и является НЕЛОМО$(k+1)$. Мы исследуем все возможности изменения состояния последовательности, скажем, в начале. Без потери общности допустим, что: 1) $x_0>x_1$. Тогда по лемме 6.А.1 имеем: 2) $x_i\geqslant x_0$ или $x_i\leqslant x_1$ для каждого i, где $-k\leqslant i<0$, или $1<i\leqslant k+1$.

Кроме того, имеем: 3) $x_{-k}\geqslant x_0$ и $x_{1+k}\leqslant x_1$.

Рассмотрим следующий случай: 4) $x_i\geqslant x_0$ для всех $-k\leqslant j<0$. Пусть j будет наименьшим целым, удовлетворяющим условию

$$x_j > x_{j+1}\geqslant \ldots \geqslant x_0 > x_1, \quad -k\leqslant j\leqslant -1. \tag{6.А.1}$$

Если такого j не существует, то должно выполняться $x_{-k}=x_{k+1}= \ldots =x_0$. Если такое j существует, то согласно лемме 6.А.1 и допущению $x_i\geqslant x_0$ имеем, что $x_i\geqslant x_j$ для всех $-k\leqslant i\leqslant j$; на самом деле $x_i=x_j$, так как j — наименьшее число, удовлетворяющее (6.А.1). Поэтому сегмент $(x_{-k}, x_{-k+1}, \ldots, x_0)$ монотонен на длине $(k+1)$; это нарушает условие — x_n есть НЕЛОМО$(k+1)$. Отсюда следует, что: а) в сегменте существует, по крайней мере, один отсчет, который меньше или равен x_1, и аналогично, б) в $(x_2, \ldots, x_k)$ существует, по крайней мере, один отсчет, который больше или равен x_0.

Достаточно рассмотреть только 4а);

5) существует i, $-k+1\leqslant i\leqslant -1$, такое, что $x_i\leqslant x_1$. Если существует такое i, что $x_i<x_1$, то пусть p будет наибольшим целым, таким, что $-k+1\leqslant p\leqslant -1$ и $x_p<x_1$. Кроме того, пусть j будет наименьшим, таким, что $p<j\leqslant 0$ и $x_j\geqslant x_0$. Очевидно, $x_p<x_{p+1}= \ldots =x_{j-1}<x_j$ по 2). По лемме 6.А.1 имеем $x_n\geqslant x_j$ или $x_n\leqslant x_p$ для каждого n, $j<n\leqslant j+k$. Так как $j<1<j+k$, то или $x_1\geqslant x_j>x_0$, или $x_1\geqslant x_p>x_1$. Это противоречит нашим допущениям. Отсюда вытекает, что: а) $x_i=x_1$, если $x_i\leqslant x_1$, где $-k+1\leqslant i\leqslant -1$, и по аналогии, б) $x_n=x_0$, если $x_n\geqslant x_0$, где $2\leqslant n\leqslant k$.

Теперь рассмотрим: 6) пусть q будет наибольшим целым, таким, что $-k\leqslant q\leqslant -1$, $x_q\neq x_0$ и $x_q\neq x_1$.

По 2) и 5а) имеем, что $x_q>x_0$. Пусть m будет наименьшим целым, таким, что $q<m\leqslant 1$ и $x_m=x_1$. Тогда $x_q>x_{q+1}= \ldots =x_{m-1}>x_m$. По лемме 6.А.1 имеем $x_n\leqslant x_q$ или $x_n\leqslant x_m$ для каждого n, $m<n\leqslant m+k$. Рассмотрим две возможности: а) если $m<0$, то вышесказанное означает, что $x_0\geqslant x_q>x_0$ или $x_0\leqslant x_m=x_1$; ни то, ни другое не верно; б) если $m=1$, то имеем $x_n\geqslant x_q>x_0$ или $x_n\leqslant x_m=x_1$ для каждого n, $1<n\leqslant 1+k$; по 4б) и 5б) должен существовать, по крайней мере, один отсчет x_i, $1<i\leqslant k$, такой, что $x_i=x_0$, что противоречит сделанному выше выводу.

Отсюда следует, что такое q не существует или, что то же самое, $x_i = x_1$ или $x_i = x_0$ для каждого i, $-k \leqslant i \leqslant -1$. По аналогии, сказанное также справедливо для всех отсчетов x_i, $2 \leqslant i \leqslant k+1$. По 4а) в сегменте $(x_{-k}, \ldots, x_0)$ существует, по крайней мере, один перепад и, аналогично, по 4б) — один перепад в сегменте $(x_1, \ldots, x_{1+k})$. Доказательство завершается применением этих же рассуждений к перепадам в двух упомянутых сегментах и вне их.

Приложение 6.Б

В приложении доказывается теорема 6.5, а также показано, что эта теорема является частным случаем некоторых свойств класса более общих сглаживающих операторов, аналогичных $МФ_3$. Сначала рассмотрим этот класс в общем виде.

Если положить, что $\{z_n\} = T\{x_n\}$, где T — оператор, и y_n = медиана (x_{n-1}, x_n, x_{n+1}), то существуют три условия возрастающей строгости, которые можно наложить на T:

I) $(z_n - x_n)(z_n - y_n) \leqslant 0$ для всех n;

II) в добавление к I): $y_n \leqslant z_n < x_n$ или $y_n \geqslant z_n > x_n$, если $x_n \neq y_n$;

III) в добавление к I): $(z_n - x_n)(z_n - y_n) < 0$, если $x_n \neq y_n$.

Другими словами, если $x_n = y_n$, то z_n содержится соответственно в I) закрытом интервале между x_n и y_n, во II) полуоткрытом интервале, исключая x_n, или в III) открытом интервале. Например, $МФ^2_3$ удовлетворяет только I), $МФ_3$ — II), но не III), и $T = 1/2I + 1/2МФ_3$ удовлетворяет III). I обозначает здесь только тождественный оператор. Имеем простую лемму.

Лемма 6.Б.1. Предположим, что T удовлетворяет I). Если (x_{n-1}, x_n, x_{n+1}) монотонна, то $z_n = x_n$ и (z_{n-1}, z_n, z_{n+1}) также монотонна.

Доказательство очевидно и по этой причине опущено. Ясно, что последовательности ЛОМО(3) являются стабильными точками любого T, удовлетворяющего I). Следующая лемма показывает, что некоторые свойства сохраняются для линейной комбинации или последовательного включения двух сглаживающих фильтров.

Лемма 6.Б.2. Пусть $0 < \alpha < 1$ и пусть $T_1 * T_2$ — составной сглаживающий фильтр, $T_2\{T_1\{x_n\}\}$.

Тогда: а) если T_1 и T_2 удовлетворяют I), то и $\alpha T_1 + (1-\alpha)T_2$, и $T_1 * T_2$ удовлетворяют I); б) если T_1 удовлетворяет II) и T_2 удовлетворяет I), то $\alpha T_1 + (1-\alpha)T_2$ удовлетворяет II); в) если T_1 удовлетворяет III) и T_2 удовлетворяет I), то $\alpha T_1 + (1+\alpha)T_2$ удовлетворяет III) и $T_1 * T_2$ удовлетворяет II); кроме того, если T_2 также удовлетворяет III), то $T_1 * T_2$ удовлетворяет III).

Замечание: даже если и T_1 и T_2 удовлетворяют II), то составной фильтр $T_1 * T_2$ может не удовлетворять II).

Доказательство. Доказательство для случая выпуклой комбинации тривиально, поэтому рассмотрим только случай составного фильтра. Пусть

$$T_1\{x_n\} = \{t_n\}, \quad T_2\{t_n\} = \{s_n\} \quad \text{и} \quad МФ_3\{x_n\} = \{y_n\}.$$

Если $y_n = x_n$, то $t_n = x_n$ и (t_{n-1}, t_n, t_{n+1}) монотонна, согласно лемме 6.Б.1; отсюда $s_n = t_n$. Если $y_n \neq x_n$ скажем, $x_n < y_n = x_{n-1}$, то рассмотрим а) и в) отдельно.

а) имеем $x_n \leqslant y_{n-1} \leqslant t_{n-1} \leqslant x_{n-1} = y_n$. Так как медиана (t_{n-1}, t_n, t_{n+1})

находится между t_{n-1} и t_n, которые, в свою очередь, находятся между x_n и y_n, то s_n содержится в сегменте между x_n и y_n.

в) имеем или $y_{n-1}=x_{n-1}$, или $x_n \leqslant y_{n-1} < x_{n-1}$. Следовательно, $t_{n-1}=x_{n-1}$ или $x_n < t_{n-1} < x_{n-1}$, а медиана (t_{n-1}, t_n, t_{n+1}), которая находится между t_{n-1} и t_n, расположена в полуинтервале $(x_n, y_n]$. Таким образом, $x_n < s_n \leqslant y_n$. Если T_2 удовлетворяет также III), то $x_n < s_n < y_n$.

Выше было указано, что $МФ_3$ удовлетворяет II). В самом деле, для любого сглаживающего фильтра, удовлетворяющего II), единственными стабильными точками являются последовательности ЛОМО (3).

Лемма 6.Б.3. Если T удовлетворяет II) и если $\{x_n\}$ — стабильная точка для T, то она является последовательностью ЛОМО (3).

Доказательство. Если $\{x_n\}$ не есть ЛОМО (3), то существует n, такое, что $y_n \neq x_n$, скажем, $y_n > x_n$. По предположению, T удовлетворяет II), таким образом $z_n > x_n$, где $T\{x_n\} = \{z_n\}$. Это противоречит допущению, что $\{x_n\}$ — стабильная точка для T. Для составных сглаживающих фильтров имеем лемму.

Лемма 6.Б.4. Пусть $T = T_1 * T_2$ и пусть $\{x_n\}$ — стабильная точка T. Тогда:

а) если T_1 удовлетворяет II) и T_2 удовлетворяет I), то $\{x_n\}$ является или последовательностью ЛОМО (3), или осциллирующей последовательностью (т. е. ..., 0, 1, 0, 1, ...);

б) $\{x_n\}$ является последовательностью ЛОМО (3), если T_1 удовлетворяет III). Вышеизложенные результаты справедливы для $T = T_2 * T_1$.

Доказательство. Используем обозначения, введенные при доказательстве леммы 6.Б.2. Предположим, что $\{x_n\}$ — стабильная точка T, но не ЛОМО (3). Тогда существует n, такое, что $y_n \neq x_n$. Предположим, что $x_n < y_n$.

а) Если T удовлетворяет II), то $x_n < t_n \leqslant y_n$, $x_n \leqslant t_{n-1} \leqslant x_{n-1}$ и $x_n \leqslant t_{n+1} \leqslant x_{n+1}$. Для того чтобы выполнялось $s_n = x_n$, мы должны иметь $x_n =$ медиана (t_{n-1}, t_n, t_{n+1}) или $x_n = t_{n-1} = t_{n+1}$.

Отсюда следует, что $x_n \leqslant s_{n+1} \leqslant t_n$ и $x_n \leqslant s_{n-1} \leqslant t_n$. Поэтому $t_n = y_n = x_{n-1} = x_{n+1}$. Так как $t_{n-1} = x_n < x_{n-1}$, то последовательность (x_{n-2}, x_{n-1}, x_n) не может быть монотонной, что справедливо также и для (x_n, x_{n+1}, x_{n+2}). Согласно вышеизложенному $x_{n-2} = x_n = x_{n+2}$; таким образом, $\{x_n\}$ должна быть осциллирующей последовательностью. Если вместо этого T_1 удовлетворяет I) и T_2 удовлетворяет II), тогда последовательность (t_{n-1}, t_n, t_{n+1}) не может быть монотонной; в противном случае (s_{n-1}, s_n, s_{n+1}) также монотонна, что противоречит допущению $y_n \neq x_n$. Если медиана $(t_{n-1}, t_n, t_{n+1}) > t_n$, то $s_n > t_n \geqslant x_n$, что не может быть, так как $\{x_n\}$ — стабильная точка для T. Поэтому имеем медиану $(t_{n-1}, t_n, t_{n+1}) < t_n$ и отсюда $s_n = t_{n-1} = t_{n+1} = x_n$. Это означает, что $\{x_n\}$ является осциллирующей последовательностью.

б) Если T_1 удовлетворяет III) и T_2 удовлетворяет I), то, согласно лемме 6.Б.2 в), $T = T_1 * T_2$ удовлетворяет II) и, согласно лемме 6.5.3, ее стабильной точкой является только последовательность ЛОМО (3). Если же вместо этого T_1 удовлетворяет I) и T_2 удовлетворяет III), то последовательность (t_{n-1}, t_n, t_{n+1}) не может быть монотонной, иначе (s_{n-1}, s_n, s_{n+1}) монотонна, что противоречит предположению о том, что $y_n \neq x_n$. Если медиана $(t_{n-1}, t_n, t_{n+1}) < t_n$, тогда $s_n > t_n \geqslant x_n$, что невозможно, так как $\{x_n\}$ — стабильная точка. Наконец, если медиана $(t_{n-1}, t_n, t_{n+1}) < t_n$, то имеем $x_n \leqslant$ медиана $(t_{n-1}, t_n, t_{n+1}) < s_n < t_n$, что снова противоречит гипотезе о стабильных точках. Поэтому можно сделать вывод, что $\{x_n\}$ должна быть последовательностью ЛОМО (3).

Замечание. Как отмечалось выше, стабильными точками последовательностей $МФ_3$ являются только последовательности ЛОМО (3).

Однако непосредственное применение леммы 6.Б.4 а) показывает, что стабильными точками $МФ^k{}_3$, который можно представить как $МФ_3 * МФ_3{}^{k-1}$, могут быть как осциллирующие последовательности, так и последовательности ЛОМО (3). Можно легко проверить, что осциллирующие последовательности являются стабильными точками $МФ_3{}^k$, если k — четное. При нечетных k стабильными точками являются только последовательности ЛОМО (3). Таким образом, осциллирующие последовательности являются единственными *рекуррентными* точками $МФ_3$.

Так как осциллирующая последовательность обычно считается достаточно негладкой, бессмысленно использовать $МФ_3$ или $МФ_3{}^k$ в качестве единственного сглаживающего фильтра при сглаживании. Ниже мы рассмотрим простой способ, который может избавить нас от этих нежелательных последовательностей. Сначала нам понадобится лемма.

Лемма 6.Б.5. Пусть $0<\alpha<1$ и пусть T_1 и T_2 удовлетворяют 1). Тогда $\{x_n\}$ — стабильная точка для $T=\alpha T_1+(1-\alpha)T_2$, если она является стабильной точкой для T_1 и T_2 одновременно.

Доказательство. Тривиально.

Объединяя лемму с замечанием к лемме 6.Б.4, имеем.

Лемма 6.Б.6. Пусть $a_0 \neq 1$, $a_k \geqslant 0$, $\sum\limits_{k=0}^{n} a_k = 1$. Тогда последовательности ЛОМО (3) являются стабильными точками сглаживающего фильтра

$$T = \sum_{k=0}^{n} a_k \, МФ_3^k$$

только для некоторых нечетных k, $a_k \neq 0$. Если же $a_k = 0$ при всех нечетных k, то единственными стабильными точками для T являются как осциллирующие последовательности, так и последовательности ЛОМО (3). Чтобы получить стабильные точки для T^m, достаточно представить T^m как

$$T^m = T^{m-1} * \left(\sum_{k=0}^{n} a_k \, МФ_3^k \right) = a_0 \, T^{m-1} + a_1 \, T^{m-1} * МФ_3 +$$

$$+ \sum_{k=2}^{n} a_k \, T^{m-1} * МФ_3^{k-1} * МФ_3 .$$

Если $\{x_n\}$ — стабильная точка для T^m, то согласно леммам 6.Б.4, 6.Б.5 и предположению, что $a_0 \neq 1$, она должна быть либо последовательностью ЛОМО (3), либо осциллирующей последовательностью. Можно проверить, что осциллирующая последовательность не может быть инвариантна к T^m, пока $a_k = 0$ для всех четных k и m или $a_k = 0$ для всех нечетных k. Во втором случае она является стабильной точкой для T.

СПИСОК ЛИТЕРАТУРЫ

1.1 T.S.Huang (ed.): *Two-Dimensional Digital Signal Processing I: Linear Filters*, Topics in Applied Physics, Vol. 42 (Springer, Berlin, Heidelberg, New York 1981)
1.2 T.S.Huang (ed.): *Picture Processing and Digital Filtering*. 2nd ed., (Springer, Berlin, Heidelberg, New York 1979)
1.3 N.Ahmed, K.R.Rao: *Orthogonal Transforms for Digital Signal Processing* (Springer, Berlin, Heidelberg, New York 1975)
1.4 Y.Yemini, J.Pearl: IEEE Trans. PAMI-**1**, 366–371 (1979)
1.5 A.K.Jain: IEEE Trans. PAMI-**1**, 356–365 (1979)
1.6 S.C.Sahasrabudhe, P.M.Vaidya: IEEE Trans. ASSP-**27**, 434–436 (1979)
1.7 S.C.Sahasrabudhe, A.D.Kulkarni: Comp. Graphics and Image Proc. **9**, 203–212 (1979)
1.8 N.Ahmed, T.Natarjan, K.R.Rao: IEEE Trans. C-**23**, 90–93 (1974)
1.9 B.D.Tseng, W.C.Miller: IEEE Trans. C-**27**, 966–968 (1978)
1.10 W.H.Chen, C.H.Smith, S.C.Fralick: A fast computational algorithm for the discrete cosine transform, IEEE Trans. COM-**25**, 1004–1009 (1977)
1.11 J.O.Eklundh: IEEE Trans. C-**21**, 801 (1972)
1.12 J.O.Eklundh: Efficient matrix transposition with limited high-speed storage, FOA Reports, Vol. 12, No. 1, pp. 1–19, 1978, National Defense Research Institute, Linkoping, Sweden
1.13 M.B.Arl: IEEE Trans. C-**27**, 72–75 (1979)
1.14 G.L.Anderson: IEEE Trans. ASSP-**28**, 280–284 (1980)
1.15 M.Onoe: IEEE Proc. **63**, 196–197 (1975)
1.16 I.DeLotto, D.Dotti: Comp. Graphics and Image Proc. **4**, 271–278 (1975)
1.17 G.E.Rivard: IEEE Trans. ASSP-**25**, 250–252 (1977)
1.18 D.H.Harris, J.H.McClellan, D.S.K.Chan, H.W.Schuessler: Vector radix fast Fourier transform, 1977 IEEE Int. Conf. on ASSP Record, (1977) pp. 548–551
1.19 C.M.Rader: IEEE Trans. CS-**22**, 575 (1975)
1.20 R.C.Agarwal, C.S.Burrus: Proc. IEEE **63**, 550–560 (1975)
1.21 I.S.Reed, T.K.Truong, Y.S.Kwoh, E.L.Hall: IEEE Trans. C-**26**, 874–881 (1977)
1.22 P.R.Chevillat: IEEE Trans. ASSP-**26**, 284–290 (1978)
1.23 B.Rice: IEEE Trans. ASSP-**27**, 432–433 (1979)
1.24 L.R.Rabiner, B.Gold: *Theory and Application of Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ 1975) pp. 419–434
1.25 J.McClellan, C.M.Rader: *Number Theory in Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ 1979)
1.26 H.Nussbaumer: *Fast Algorithms for the Computation of Convolutions and DFTs*, Springer Series in Information Sciences, Vol. 2 (Springer, Berlin, Heidelberg, New York 1981)
1.27 H.F.Silverman: IEEE Trans. ASSP-**25**, 152–165 (1977)
1.28 H.F.Silverman: IEEE Trans. ASSP-**26**, 268 und 482 (1978)
1.29 B.D.Tseng, W.C.Miller: IEEE Trans. ASSP-**26**, 268–269 (1978)
1.30 S.Zohar: IEEE Trans. ASSP-**27**, 409–421 (1979)
1.31 T.W.Parsons: IEEE Trans. ASSP-**27**, 398–402 (1979)
1.32 I.S.Reed, T.K.Truong: IEEE Trans. C-**28**, 487–492 (1979)
1.33 H.Nawab, J.H.McClellan: IEEE Trans. ASSP-**27**, 394–398 (1979)
1.34 L.R.Morris: IEEE Trans. ASSP-**26**, 141–150 (1978)
1.35 A.Peled: A Low-Cost Image Processing Facility Employing a New Hardware Realization of High-Speed Signal Processors, in *Advances in Digital Image Processing*, ed. by P. Stucki (Plenum, New York 1979)
1.36 R.W.Patterson, J.H.McClellan: IEEE Trans. ASSP-**26**, 447–455 (1978)
1.37 W.K.Pratt: "Median Filtering"; semiannual report, Image Processing Inst., Univ. of Southern California, Sept. (1975) pp. 116–123
1.38 B.R.Frieden: J. Opt. Soc. Am. **66**, 280–283 (1976)
1.39 T.S.Huang, G.J.Yang, G.Y.Tang: IEEE Trans. ASSP-**27**, 13–18 (1979). A shorter version

appeared in Proc. 1978 IEEE Conf. on Pattern Recognition and Image Processing, May 31-June 2, 1978, pp. 128–131
1.40 N.C.Gallagher, G.L.Wise: IEEE Trans. ASSP (to be published)
1.41 W.L.Eversole, D.J.Mayer, F.B.Frazee. T.F.Cheek, Jr.: "Investigation of VLSI Technologies for Image Processing", Proc. Image Understanding Workshop, Pittsburgh, Penn., pp. 191–195, Nov. (1978)
1.42 P.M.Narendra: IEEE Trans. PAMI (to be published). A short version appeared in Proc. 1978 IEEE Conf. on Pattern Recognition and Image Processing Chicago, ILL., pp. 137–141; May (1978)
1.43 E.Ataman, V.K.Aatre. K.M.Wong: IEEE Trans ASSP-**28**, 415–421 (1980)
1.44 A.P.Reeves, A.Rostampour: IEEE Trans. PAMI (to be published)
1.45 G.J.Wolfe, J.L.Mannos: "A Fast Median Filter Implementation", Proc. SPIE Seminar on Image Processing, Sept. 1979, San Diego, Ca.
1.46 T.S.Huang: "Noise filtering of moving images", Proc. Workshop on Automatic Tracking, Redstone Arsenal, Ala Nov. 19–20 (1979)
1.47 G.Yang, T.S.Huang: 'Median Filters and Their Applications to Image Processing"; Tech. Rpt., School of Electrical Engineering, Purdue University (1980)

2.1 G.L.Anderson: IEEE ASSP-**28**, 280–284 (1980)
2.2 N.M.Brenner: IEEE AU-**17**, 128–132 (1969)
2.3 B.R.Hunt: Proc. IEEE **60**, 884–887 (1972)
2.4 R.C.Singleton: IEEE AU-**15**, 91–98 (1967)
2.5 H.L.Buijs: Appl. Opt. **8**, 211–212 (1969)
2.6 J.O.Eklundh: "Efficient Matrix Transposition with Limited High-Speed Storage"; FOA Reports. Vol. 12, No. 1, National Defence Research Institute, Stockholm, Sweden (1978)
2.7 J.O.Eklundh: IEEE C-**21**, 801–803 (1972)
2.8 H.K.Ramapriyan: IEEE C-**24**, 1221–1226 (1976)
2.9 R.E.Twogood, M.P.Ekstrom: IEEE C-**24**, 950–952 (1976)
2.10 R.W.Floyd: "Permuting Information in Idealized Two-Level Storage", in *Complexity of Computer Computations*, ed. by R.E.Miller, J.W.Thatcher (Plenum Press, New York 1972) pp. 105–109
2.11 H.S.Stone: IEEE C-**20**, 153–161 (1971)
2.12 D.E.Knuth: *The Art of Computer Programming*, Vol. 3 (Addison-Wesley, Reading, MA 1973) pp. 7, 573
2.13 L.G.Delcaro, G.L.Sicuranza: IEEE C-**23**, 967–970 (1974)
2.14 U.Schumann: Angew. Informatik **14**, 213–216 (1972)
2.15 U.Schumann: IEEE C-**22**, 542–543 (1973)
2.16 J.W.Cooley, J.W.Tukey: Math. Comput. **19**, 297–301 (1965)
2.17 W.T.Cochran et al.: IEEE AU-**15**, 45–55 (1967)
2.18 G.E.Rivard: IEEE ASSP-**25**, 250–252 (1977)

3.1 S.Winograd: Math. Comput. **32**, 175–199 (1978)
3.2 R.C.Agarwal, J.W.Cooley: IEEE Trans. ASSP-**25**, 392–410 (1977)
3.3 H.J.Nussbaumer: Electron. Lett. **13**, 386–387 (1977)
3.4 H.J.Nussbaumer: "New Algorithms for Convolution and DFT Based on Polynomial Transforms", in IEEE 1978 Intern. Conf. Acoust., Speech, Signal Processing Proc., pp. 638–641
3.5 H.J.Nussbaumer, P.Quandalle: IBM J. Res. Dev. **22**, 134–144 (1978)
3.6 D.J.Winter: *The Structure of Fields* (Springer, Berlin, Heidelberg, New York 1974)
3.7 T.Nagell: *Introduction to Number Theory* (Chelsea, New York 1964)
3.8 C.M.Rader: IEEE Trans. C-**21**, 1269–1273 (1972)
3.9 R.C.Agarwal, C.S.Burrus: Proc. IEEE **63**, 550–560 (1975)
3.10 B.Gold, C.M.Rader, A.V.Oppenheim, T.G.Stockham: *Digital Processing of Signals*, (McGraw Hill, New York 1969) Ch. 7, pp. 203–213

3.11 S.Winograd. "Some Bilinear Forms Whose Multiplicative Complexity Depends on the Field of Constants"; IBM Res. Rpt. RC5669, IBM Watson Research Center, Yorktown Heights, N.Y. (1975)
3.12 J.W.Cooley, J.W.Tukey: Math. Comput. **19**, 297–301 (1965)
3.13 P.Quandalle: "Filtrage numérique rapide par transformées de Fourier et transformées polynômiales. Etude de l'implantation des algorithmes sur microprocesseurs"; Ph. D. Thesis, University of Nice, France (1979)
3.14 R.C.Agarwal, J.W.Cooley: "New Algorithms for Digital Convolution", in 1977 Intern. Conf., Acoust., Speech and Signal Processing Proc., p. 360
3.15 C.M.Rader, N.M.Brenner: IEEE Trans. ASSP-**24**, 264–266 (1976)
3.16 C.S.Burrus: "Digital Filter Realization by Distributed Arithmetic", in Proc. 1976 IEEE Intern. Symp. Circuits and Systems, Munich (1976) pp. 106–109
3.17 H.J.Nussbaumer, P.Quandalle: IEEE Trans. ASSP-**27**, 169–181 (1979)
3.18 A.V.Oppenheim, R.W.Schafer: *Digital Signal Processing*, (Prentice Hall, Englewood Cliffs N.J. 1975) pp. 320–321
3.19 C.M.Rader: Proc. IEEE **56**, 1107–1108 (1968)
3.20 I.J.Good: IEEE Trans. C-**20**, 310–317 (1971)
3.21 D.P.Kolba, T.W.Parks: IEEE Trans. ASSP-**25**, 281–294 (1977)
3.22 R.C.Agarwal, C.S.Burrus: IEEE Trans. ASSP-**22**, 87–97 (1974)

4.1 J.W.Cooley, J.W.Tukey: Math. Comp. **19**, 297–301 (1965)
4.2 R.Yavne: "An Economical Method for Calculating the Discrete Fourier Transform", *AFIPS Conference Proceedings*, Vol. 33, Part 1 (Thompson Book Co., Washington, D.C, 1968) pp. 115–125
4.3 S.Winograd: Proc. Nat. Acad. Sci. USA **73**, No. 4, 1005–1006 (1976)
4.4 S.Winograd: "The Effect of the Field of Constants on the Number of Multiplications", *Proceedings of the 16th Annual Symposium on Foundations of Computer Science* (The Institute of Electrical and Electronics Engineers, New York 1975) pp. 1–2
4.5 C.M.Rader: Proc. IEEE **56**, 1007–1008 (1968)
4.6 M.Abramowitz, I.Stegun: *Handbook of Mathematical Functions* (Dover, New York 1965) p. 864
4.7 D.E.Knuth: *The Art of Computer Programming*, Vol. 2 (Addison-Wesley, Reading, Mass. 1969) Sect. 4.3.2
4.8 T.Nagell: *Introduction to Number Theory* (Wiley, New York 1951)
4.9 W.J.LeVeque: *Topics in Number Theory*, Vol. 1 (Addison-Wesley, Reading, Mass. 1958)
4.10 I.J.Good: J. R. Stat. Soc. **B20**, 361–372 (1958)

5.1 J.W.Tukey: *Exploratory Data Analysis* (Addison-Wesley, Reading, Mass., 1977, preliminary ed. 1971)
5.2 A.E.Sarhan, B.G.Greenberg (eds.): *Contributions to Order Statistics* (Wiley, New York 1962)
5.3 H.A.David: *Order Statistics* (Wiley, New York 1970)
5.4 J.L.Gastwirth, H.Rubin: Ann. Stat. **3**, 1070–1100 (1975)
5.5 L.R.Rabiner, M.R.Sambur, C.E.Schmidt: IEEE Trans. ASSP-**23**, 552–557 (1975)
5.6 N.S.Jayant: IEEE Trans. COM-**24**, 1043–1045 (1976)
5.7 B.R.Frieden: J. Opt. Soc. Am. **66**, 280–283 (1976)
5.8 W.K.Pratt: *Digital Image Processing* (Wiley, New York 1978)
5.9 G.W.Wecksung, K.Campbell: Computer **7**, 63–71 (1974)
5.10 B.Justusson: "Order Statistics on Stationary Random Processes, with Applications to Moving Medians"; Tech. Rpt. TRITA-MAT-1, Royal Institute of Technology, Stockholm, Sweden (1979)
5.11 A.Rosenfeld, A.C.Kak: *Digital Picture Processing* (Academic Press, New York 1976)
5.12 D.B.Owen: *Handbook of Statistical Tables* (Addison-Wesley, Reading, Mass. 1962)
5.13 P.F.Velleman: "Definition and Comparison of Robust Nonlinear Data Smoothing Algorithms"; Tech. Rpt. Economic and Social Statistics Dept., Cornell University (1978)

5.14 G.Heygster: "Untersuchung von Zweidimensionalen Rangordnungsoperatoren im Orts- und Frequenzbereich", in *Bildverarbeitung und Mustererkennung*, ed. by E. Triendl, Informatik Fachberichte, Vol. 17 (Springer, Berlin, Heidelberg, New York 1978) pp. 204–208
5.15 B.Justusson: "Noise Reduction by Median Filtering" in Proc. 4th Intern. Joint Conf. on Pattern Recognition (1978) pp. 502–504
5.16 S.G.Tyan: "Fixed Points of Running Medians"; Tech. Rpt. Dept. of Electrical Engineering and Electrophysics, Polytechnic Institute of New York (1977)
5.17 P.M.Narendra: "A Separable Median Filter for Image Noise Smoothing" in Proc. IEEE Conf. on Pattern Recognition and Image Processing (1978) pp. 137–141
5.18 P.F.Velleman: Proc. Nat. Acad. Sci. USA **74**, 434–436 (1977)
5.19 L.S.Davis, A.Rosenfeld: IEEE Trans. SMC-**8**, 704–710 (1978)
5.20 M.Nagao, T.Matsuyama: "Edge Preserving Smoothing" in Proc. 4th Intern. Joint Conf. on Pattern Recognition (1978) pp. 518–520

6.1 J.W.Tukey: *Exploratory Data Analysis* (Addison-Wesley, Reading, MA 1971)
6.2 L.R.Rabiner, M.R.Sambur, C.E.Schmidt: IEEE Trans. ASSP-**23**, 552–557 (1975)
6.3 B.k.Frieden: J. Opt. Soc. Am. **66**, 280–283 (1976)
6.4 T.S.Huang, G.J.Yang, G.Y.Tang: "A Fast Two-Dimensional Median Filtering Algorithm"; School of Electrical Engineering, Purdue University (1977)
6.5 P.F.Velleman: "Robust Non-Linear Data Smoothing"; Tech. Rpt. 89, Ser. 2, Dept. of Statistics, Princeton University (1975)
6.6 J.W.Tukey: *Exploratory Data Analysis* (Addison-Wesley, Reading, MA 1977)
6.7 P.F.Velleman: Proc. Nat. Acad. Sci. USA **74**, 434–436 (1977)
6.8 J.W.Tukey.: Class Notes on Special Topics in Statistics, Statistics 411, Dept. of Statistics, Princeton University (1974)
6.9 I.J.Schoenberg: "Some Analytical Aspects of the Problem of Smoothing", in *Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948* (Interscience, New York 1948) pp. 351–370
6.10 B.Justusson: "Median Filters on Deterministic Signals"; report in preparation, Math. Inst., Royal Institute of Technology, Stockholm, Sweden (1979)

Дополнительный список литературы

Преобразования

B. Arambepola: General discrete Fourier transform and the fast Fourier algorithm. Proc. EUSIPCO-80, Sept. 16–18, 1980, Lausanne, Switzerland, pp. 583–588
B. Arambepola, P. J. W. Rayner: Discrete transforms over residue class polynomial rings with applications in computing multidimensional convolutions. IEEE Trans. ASSP-**28** (4), 407–414 (1980)
J. W. Cooley, S. Winograd: A limited range DFT algorithm, Proc. ICASSP-80, April 9–11, 1980, Denver, Colorado, pp. 213–217
F. A. Kamagar, K. R. Rao: Fast algorithms for the 2-D discrete cosine transformation. Proc. IEEE Intern. Symp. on Circuits and Systems, April 28–30, 1980, pp. 206–209
J. Makhoul: A fast cosine transform in one and two dimensions. IEEE Trans. ASSP-**28** (1), 27–34 (1980)
D. C. Munson, B. Lin: Floating point error bound in the prime factor FFT. Proc. ICASSP-80, April 9–11, 1980, Denver, Colorado, pp. 69–72
H. Nawab: "Parallelism and code optimization issues in FFT and WFTA algorithms"; S. M. thesis, Massachusetts Inst. Technol., Cambridge (1978)
H. Nawab, J. H. McClellan: Corrections to "Bounds on the minimum number of data transfers in WFTA and FFT programs". IEEE Trans., ASSP-**28**, 480–481 (1980)
H. J. Nussbaumer: Fast polynomial transform algorithms for digital convolution. IEEE Trans. ASSP-**28** (2), 205–215 (1980)

H. J. Nussbaumer: Fast polynomial transform methods for multidimensional DFTs. Proc. ICASSP-80, April 9–11, 1980, Denver, Colorado, pp. 235–237
R. W. Patterson: "Fixed point error analysis of the Winograd Fourier transform algorithm"; M. S. thesis, Massachusetts Inst. Technol. Cambridge (1977)
B. Rice: Some good fields and rings for computing number theoretic transforms. IEEE Trans. ASSP-27 (4), 432–433 (1979)
H. F. Silverman: A method for programming the complex, general-*N* Winograd Fourier transform algorithm. Proc. ICASSP-77, Hartford, CT, May 1977, pp. 369–372
S. Zohar: "Outline of a fast hardware implementation of Winograd's DFT algorithm", in IEEE 1980 Intern. Conf. ASSP-3, 796–799

Медианные фильтры

E. Ataman, V. K. Aatre, K. M. Wong. Some statistical properties of median filters, Tech. Rpt. Dept. of Electrical Engineering, Nova Scotia Technical College, Halifax, Canada (1979)
E. Ataman, V. K. Aatre, K. M. Wong: A fast method for real-time median filtering. IEEE Trans. ASSP-28 (4), 415–421 (1980)
G. Heygster: „Rangordnungsoperatoren in der digitalen Bildverarbeitung", Ph. D. Thesis, Math. Nat. Fakultät, Georg-August-Universität, Göttingen, (1979)
G. Heygester: Rank filters in digital image processing, Proc. 5th Intern. Conf. on Pattern Recognition, Dec. 1–4, 1980, Miami Beach, Florida, pp. 1165–1167
T. Kitahashi, O. Saito, L. Abele, F. Wahl, H. Marko: An extension of median filtering and its deterministic properties, Proc. 5th Intern. Conf. on Pattern Recognition, Dec. 1–4, 1980, Miami Beach, Florida, pp. 1177–1179
C. L. Mallows: "Some theoretical results on Tukey's 3 R smoothers", in *Smooting Techniques for Curve Estimation*, ed. by Th. Gasser, M. Rosenblatt, Springer Lecture Notes in Mathematics 757, (Springer, Berlin, Heidelberg, New York, 1979) pp. 77–90

Список литературы, переведенной на русский язык

1.3. **Ахмед Н., Рао К. Р.** Ортогональные преобразования для цифровой обработки сигналов: Пер. с англ. — М.: Связь, 1980. — 248 с.
1.15. **Оноэ М.** Метод двумерного преобразования без транспонирования большой матрицы данных. — ТИИЭР, т. 63, № 1, 1975, с. 198—199.
1.20. **Агарвал, Баррас.** Теоретико-числовые преобразования для быстрого вычисления цифровой свертки. — ТИИЭР, т. 63, № 4, 1975, с. 6—20.
1.24. **Рабинер Л. Р., Голд Б.** Теория и применение цифровой обработки сигналов: Пер. с англ. — М.: Мир, 1978. — 848 с.
1.25. **Макклеллан Дж. Х., Рейдер Ч. М.** Применение теории чисел в цифровой обработке сигналов: Пер. с англ. — М.: Радио и связь, 1983. — 264 с.
1.26. **Нуссбаумер Г. Дж.** Быстрое преобразование Фурье и алгоритмы вычисления сверток. — М.: Радио и связь, 1984. — 256 с.
2.3. **Хант.** Структура данных и организация вычислений при цифровом улучшении качества изображений. — ТИИЭР, т. 60, № 7, 1972, с. 160—164.
2.12. **Кнут Д.** Искусство программирования для ЭВМ. Сортировка и поиск: Пер. с англ. — М.: Мир, 1978. Т. 3. — 844 с.
3.9. **Агарвал, Баррас.** Теоретико-числовые преобразования для быстрого вычисления цифровой свертки. — ТИИЭР, т. 63, № 4, 1975, с. 6—20.
3.10. **Голд Б., Рейдер Ч.** Цифровая обработка сигналов: Пер. с англ. — М.: Сов. радио, 1973, ч. 7, с. 234—269.
3.18. **Оппенхейм А., Шафер Р.** Цифровая обработка сигналов: Пер. с англ. — М.: Связь, 1979. — 416 с.

3.19. **Рейдер К.** Дискретные преобразования Фурье для случая, когда число выборочных величин является простым числом. — ТИИЭР, 1968, т. 56, № 7, с. 95—96.
4.7. **Кнут Д.** Искусство программирования для ЭВМ. Т. 2. Получисленные алгоритмы: Пер. с англ. — М.: Мир, 1978. — 724 с.
5.8 **Прэтт У.** Цифровая обработка изображений: Пер. с англ. — М.: Мир, 1982. Кн. 1 и 2. — 312 и 480 с.

Дополнительный список литературы, предложенный редактором перевода

1. **Ершов А. А.** Стабильные методы оценки параметров. (Обзор). — Автоматика и телемеханика, 1978, № 8, с. 66.
2. **Иконика.** Теория и методы обработки изображений. Сб. статей под ред. Д. С. Лебедева, Н. Р. Поповой. — М.: Наука, 1983. — 156 с.
3. **Красненкер В. М.** Стабильные методы обнаружения сигналов на фоне помех. (Обзор). — Автоматика и телемеханика, № 15, 1980. — 65 с.
4. **Стогов Г. В., Макшанов А. В., Мусаев А. А.** Устойчивые методы обработки результатов измерений. — Зарубежная радиоэлектроника, № 9, 1982, с. 3.
5. **Цифровая** обработка сигналов и ее применения. — Сб. статей под ред. Л. П. Ярославского. — М.: Наука, 1981. — 207 с.
6. **Ярославский Л. П.** Введение в цифровую обработку изображений. — М.: Сов. радио, 1979. — 312 с.
7. **Ярославский Л. П., Мерзляков Н. С.** Цифровая голография. — М.: Наука, 1982. — 220 с.

ОГЛАВЛЕНИЕ

Т. С. Хуанг, Дж.-О. Эклунд, Г. Дж. Нуссбаумер и др.

БЫСТРЫЕ АЛГОРИТМЫ В ЦИФРОВОЙ ОБРАБОТКЕ ИЗОБРАЖЕНИЙ

Редактор Е. А Засядько
Обложка художника Г. С. Студеникиной
Художественный редактор Л. Н. Сильянов
Технический редактор Г. И. Колосова
Корректор Т. Л. Кускова

ИБ № 659

Сдано в набор 28.12.83 **Подписано в печать 29.03.84**
Формат 60×90¹/₁₆ Бумага тип. № 1 Гарнитура литературная Печать высокая
Усл. печ. л. 14.0 Усл. кр.-отт. 14,25 Уч.-изд. л. 14,47 Тираж 8000 экз. Изд. № 20450
Зак. № 2 **Цена 1 р. 70 к.**
Издательство «Радио и связь». 101000 Москва, Почтамт, а/я 693

Московская типография № 5 ВГО «Союзучетиздат»
101000 Москва, ул. Кирова, д. 40