В.В.ЛОСЕВ

МИКРО-ПРОЦЕССОРНЫЕ УСТРОЙСТВА ОБРАБОТКИ ИНФОРМАЦИИ

●

АЛГОРИТМЫ ЦИФРОВОЙ ОБРАБОТКИ

Допущено
Министерством народного образования БССР
в качестве учебного пособия для студентов
радиотехнических специальностей вузов

Минск "Вышэйшая школа" 1990

ББК32.973.2я73
Л 79
УДК 681.325.5.068(075.8)

Рецензенты: кафедра радиосистем Ленинградского электротехнического института им. В.И. Ульянова (Ленина); доктор технических наук, профессор, заведующий кафедрой радиосистем Одесского политехнического института *М.Б. Свердлик*

Лосев В.В.

Л79 Микропроцессорные устройства обработки информации. Алгоритмы цифровой обработки: Учеб. пособие для вузов. — Мн.: Выш. шк., 1990. - 132 с: ил.

ISBN 5-339-00348-5.

Описываются наиболее употребительные алгоритмы цифровой обработки сигналов с помощью микропроцессоров и микроЭВМ. Рассматриваются вопросы повышения эффективности вычислительного процесса, быстрые алгоритмы вычислений наиболее важных выражений и функций, применение этих алгоритмов в радиотехнических системах.

Для студентов радиотехнических специальностей. Может быть полезно инженерам, занимающимся вопросами обработки цифровой информации.

2404090000 - 034
Л——————30-90 **ББК 32.973.2я73**
М304(03) -90

ISBN 5-339-00348-5

ПРЕДИСЛОВИЕ

В настоящее время цифровая обработка информации находит все более широкое применение в различных областях науки и техники. Можно с уверенностью утверждать, что она выделилась и оформилась в самостоятельное научное и техническое направление, которое уже сейчас позволило получить результаты, ранее считавшиеся недостижимыми.

Развитие указанного направления идет по двум основным путям: 1) разработки эффективных алгоритмов вычислений, минимизирующих количество элементарных операций (сложения, умножения, пересылки); 2) создания элементной базы в виде наборов микропроцессорных комплектов и микроЭВМ с соответствующим математическим обеспечением.

Первое направление до недавнего времени полностью относилось к компетенции математиков, второе — развивалось усилиями специалистов по техническим наукам. Интенсивное внедрение математических методов в технику, появление микропроцессоров и микроЭВМ, приведшее к компьютеризации многих сфер человеческой деятельности, в значительной степени стерло эти границы. Доступность и удобство вычислительных средств и^величение сложности алгоритмов обработки потребовали от инженера глубокого владения арсеналом экономных вычислительных методов и умения использовать их при проектировании радиоэлектронной аппаратуры с применением микропроцессоров и микроЭВМ. Незнание эффективных алгоритмов приводит к созданию вычислительных систем низкой производительности, что не только экономически невыгодно, но в ряде случаев (например, при обработке сигналов в реальном масштабе времени) просто неприемлемо.

В то же время прогресс в области совершенствования элементной базы и методов проектирования позволил значительно расширить возможности как программной, так и аппаратной реализации, открыть дополнительные резервы повышения точности, помехоустойчивости, быстродействия и других показателей эффективности. Все это привело к значительной ревизии представлений о радиоэлектронных устройствах и процедуре их проектирования.

Предлагаемое учебное пособие ставит перед собой цель объединить указанные направления и осветить наиболее важные проблемы цифровой обработки сигналов с единых методических позиций в простой и доступной для первоначального изучения форме. Книга содержит материал, отраженный в учебных программах для радиотехнических специальностей. Она может быть полезна также инженерам, занимающимся разработкой и эксплуатацией систем обработки дискретной информации. В основу пособия положены курсы лекций, прочитанные автором на протяжении ряда лет в Минском радиотехническом институте, материалы практических занятий и лабораторных работ. От читателя требуются математическая подготовка и знание основ цифровой техники в объеме, не превышающем вузовские программы.

Автор благодарит рецензентов — кафедру радиосистем Ленинградского электротехнического института им. В.И.Ульянова (Ленина) (зав. кафедрой доктор технических наук, профессор Ю.М. Казаринов) и М.Б.Свердлика,доктора технических наук, профессора, заведующего кафедрой радиосистем Одесского политехнического института, — за замечания, способствовавшие улучшению книги.

Все отзывы и пожелания просим присылать по адресу: 220048, Минск, проспект Машерова, 11, издательство "Вышэйшая школа".

Автор

СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ

АЦП - аналого-цифровой преобразователь
БПА - быстрое преобразование Адамара
БПФ — быстрое преобразование Фурье
ВКФ — функция Виленкина—Крестенсона
ДПФ - дискретное преобразование Фурье
ДЭФ - дискретная экспоненциальная функция
НОД - наибольший общий делитель
НП — неветвящаяся программа
ОДПФ — обратное дискретное преобразование Фурье
ПМД — последовательность максимальной длины
ТЧП - теоретико-числовое преобразование
ЦАП — цифроаналоговый преобразователь
ЦВУ — цифровое вычислительное устройство

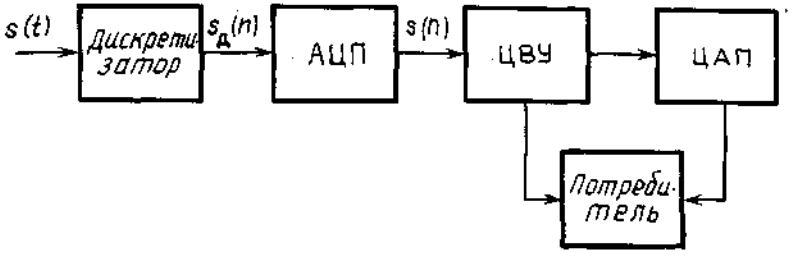


Рис. 1.2. Схема цифровой обработки.

Таким образом, общая схема обработки информации принимает вид, показанный на рис. 1.2.

1.2. ДИСКРЕТИЗАЦИЯ И КВАНТОВАНИЕ

Преобразование аналогового сигнала в цифровую форму приводит к потерям информации. Поэтому при проектировании системы цифровой обработки возникает задача оптимального выбора интервалов дискретизаций и квантования, минимизирующих эти потери и позволяющих получить устройство обработки приемлемой сложности.

Для сигналов с ограниченным спектром по теореме Котельникова интервал дискретизации T должен выбираться из условия

$$T \leqslant 1/2f_{max}, \qquad (1\text{-}1)$$

где f_{max} - верхняя граничная частота спектра исходного сигнала. Условие $O_{Л)}$ означает, что на период верхней гармоники сигнала должно приходиться не менее двух отсчетов. При этом потерь информации нет и аналоговый сигнал может быть восстановлен точно по своим дискретным отсчетам.

Для нестационарных процессов с неограниченным спектром условие (1.1) выполнить невозможно. В этом случае интервал дискретизации выбирается так, чтобы его длительность не превышала времени корреляции $г_{кор}$ (времени, за которое практически исчезает статистическая связь между соседними значениями сигнала). Ошибка в воспроизведении исходного сигнала оценивается выражением [7]

$$\sigma^2 \simeq (6a - 1)/12a^2,$$

где $a = т_{кор}/T$ - отношение времени корреляции к интервалу дискретизации.

Для квантования сигнала используется нелинейный элемент со ступенчатой амплитудной характеристикой. Входному сигналу, лежащему в интервале Ax_i, соответствует одно значение выходного сигнала x_i. *Шаг квантования* Ax_i может быть как равномерным, так и неравномерным. Исходный сигнал искажается в любом случае. Искажения характеризуются шумом квантования. Наиболее употребительным является равношаговое квантование. Шум квантования чаще всего описывается нормальным случайным процессом, корреляционная функция которого зависит от числа уровней квантования. С увеличением числа уровней квантования шум приближается к белому.

ББ

УД

1. МИКРОПРОЦЕССОРНАЯ ОБРАБОТКА ДИСКРЕТНЫХ СИГНАЛОВ

1.1. АНАЛОГОВЫЕ И ДИСКРЕТНЫЕ СИГНАЛЫ

Электрические сигналы, с помощью которых передается информация, являются *аналоговыми* и описываются непрерывными функциями, т. е. функциями, заданными на несчетном множестве точек. В наиболее распространенных случаях аргументами функций являются время t и частота f. Как аргумент, так и сама функция $s(t)$ могут принимать любые значения из некоторых интервалов $t_0 < t < t_1$, $s_0(f) < s(t) < S_1(t)$. Пример аналогового сигнала показан на рис. 1.1.

В отличие от аналогового сигнала *дискретный* сигнал $J_д(n)$ —решетчатая функция дискретной переменной n, принимающей только фиксированные значения. В большинстве практических приложений эти значения являются равноотстоящими, поэтому можно пронумеровать их целыми числами $n = 0, 1, 2, \ldots$ (см. рис. 1.1). В данном случае величина $s_д(и)$ называется *отсчетом* сигнала в точке и. Конечная последовательность отсчетов в N точках обозначается $\{s_д(n)\} = \{s_д(0), s_д(1), \ldots, s_д(N-1)\}$ или в векторной форме $S_д = [s_д(0), s_д(1), \ldots, s_д(N-1)]$.

Для обработки в цифровых вычислительных устройствах (ЦВУ) дискретные сигналы должны быть проквантованы по амплитуде, т. е. представлены в виде последовательности чисел с ограниченным количеством разрядов. Такие сигналы называются *цифровыми*. Цифровой сигнал обозначается $\{s(ri)\} = \{s(0), s(1), \ldots, s(N-1)\}$ или в векторной форме $S = [s(0), s(1), \ldots, s(N-1)]$. Преобразование сигналов в цифровую форму производится с помощью *аналого-цифрового* преобразователя (АЦП).

Полученная в вычислительном устройстве информация выдается потребителю в цифровой или аналоговой форме. В последнем случае между потребителем и вычислительным устройством ставится *цифроаналоговый преобразователь* (ЦАП).

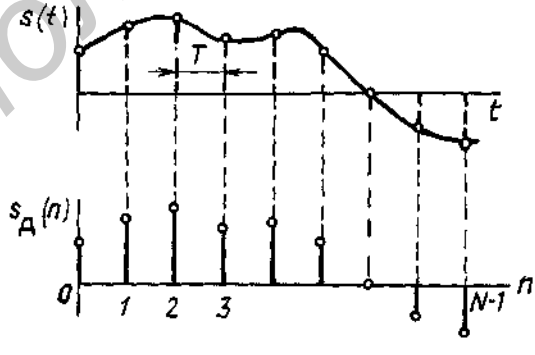


Рис. 1.1. Аналоговый и дискретный сигналы.

ками. При этом абсолютные значения складываются, а сумме присваивается код знака слагаемых. Однако, если слагаемые имеют разные знаки, то операция сложения в прямых кодах затруднена, так как в этом случае следует определить большее по модулю число, выполнить вычитание и присвоить разности знак большего по модулю числа. Поэтому для выполнения операции сложения используется обратный или дополнительный код,

Дпя образования обратного кода отрицательного числа из прямого кода в знаковом разряде следует сохранить единицу, а во всех остальных разрядах заменить единицы на нули, а нули на единицы. Например, число -125 в обратном коде запишется так: $[125]_{о} = 1.0000010$. Обратный код является дополнением модуля исходного числа до наибольшего числа без знака, помещающегося в разрядную сетку. Для нашего примера $1111101 + 0000010 = 1111111 = 2^{7} - 1$.

Дополнительный код получается из обратного, если к младшему разряду прибавить единицу. В частности, дополнительный код числа —125 равен $[-125]_{д} = 1.0000011$. Дополнительный код без знакового разряда является дополнением модуля числа до величины 2^{n}, где n — количество разрядов в формате. Если операция сложения выполняется с использованием дополнительного кода, то и в памяти удобно хранить числа в дополнительном коде.

Использование знакового разряда позволяет в и-разрядном процессоре работать с числами в диапазоне $\pm(2^{n-1} - 1)$. Приведем этот диапазон для 8-разрядного процессора (отрицательные числа записаны в дополнительном коде):

1 0 0 . . . 0 1	—127
. . . . .	
1 1 1 . . . 1 1	—1
0 0 0	0
0 0 0 . . . 0 1	1
. . . . .	
0 1 1 . . . 1 1	+127

При втором способе представления знаковый разряд отсутствует, поэтому считается, что все числа положительные и принимают значения в диапазоне $0-(2^{n}-1)$:

0 0 0 . . . 0 0	0
0 0 0 . . . 0 1	1
0 0 0 . . . 1 0	2
. . . . .	
1 1 1 . . . 1 0	254
1 1 1 . . . 1 1	255

Двоично-десятичное представление состоит из тетрад, каждая из которых представляет собой двоичное число со значениями от 0 до 9. Для 8-разрядного процессора количество тетрад равно двум. При этом диапазон представления чисел равен 0—99:

ББ

УД

Для оценки потерь можно использовать соотношения, устанавливающие связь между моментами m_i исходного непрерывного распределения и моментами μ_i этого распределения, полученными по сгруппированным на интервалах длиной Дд: выборочным значениям. Для первого и второго центрального моментов эти соотношения имеют следующий вид [7];

$$\mu_1 - m_1 = 0; \quad \mu_2 - m_2 = \Delta x^2/12 .$$

Другой подход дает использование теоремы Котельникова. Предположим, что последовательные значения амплитуд дискретизированного сигнала статистически независимы и описываются одномерной плотностью вероятности *w(x).* Преобразование Фурье функции *w(x)* называется *характеристической функцией.* Она является спектральным образом плотности вероятности. Ограничим характеристическую функцию некоторой "частотой" f_{max}, т. е. будем рассматривать плотность вероятности *w(x)* как функцию с ограниченным спектром. По теореме Котельникова такая функция однозначно представляется своими выборочными отсчетами, взятыми с интервалом $h < ^/2/_{тяк}$ • При замене сигнала его плотностью вероятности задача воспроизведения сигнала сводится к восстановлению моментов по его квантованным выборкам. Поэтому число уровней квантования обусловлено важностью восстановления или сохранения моментов. Часто (например, в задачах обнаружения) достаточно эффективным бывает грубое квантование с интервалом *h Уа* вплоть до двоичного.

1.3. МИКРОПРОЦЕССОРНАЯ ОБРАБОТКА СИГНАЛОВ

13.1. Представление чисел

Числа в микропроцессорных системах представляются в виде совокупности цифр. При этом количество двоичных разрядов называется *форматом данных.* Наиболее употребительными являются следующие форматы: 1 символ — бит; 8 символов - байт; 16 символов — полуслово; 32 символа - слово; 64 символа - двойное слово.

В большинстве микропроцессорных систем используется арифметика с фиксированной запятой (как более простая и быстродействующая при технической реализации по сравнению с арифметикой с плавающей запятой). При этом данные могут кодироваться тремя способами: 1) двоичным числом со знаком; 2) двоичным числом без знака; 3) двоично-десятичным числом со знаком или без него.

Рассмотрим эти способы на примере 8-разрядного микропроцессора. При первом способе старший разряд, называемый знаковым, содержит информацию о знаке, остальные разряды дают абсолютное значение числа. Положительные числа имеют в знаковом разряде 0, а отрицательные числа — 1. Существует три разновидности такой записи — *прямой код, обратный* и *дополнительный.*

Числа, записанные в виде абсолютного значения с добавлением знакового разряда, образуют прямой код. Например, числа 125 и —125 запишутся в прямом коде следующим образом: $[125]_{пр} = 0.1111101$, $[-125]_{пр} = 1.1111101$.

Прямые коды используются в операциях хранения, умножения и деления чисел. В прямых кодах удобно также суммировать числа с одинаковыми зна-

0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 1	1
. . . . .	
0 0 0 1 0 0 0 0	10
. . . . .	
1 0 0 1 1 0 0 1	99

Способ представления определяется программистом. Каких-либо особых сигналов, указывающих на тот или иной способ представления чисел, микропроцессор, как правило, не вырабатывает.

13.2. Арифметические и логические операции

Обработка информации в микропроцессорных системах выполняется при помощи набора одноместных или двухместных операций (сложения, вычитания, умножения, деления, левого и правого сдвигов, конъюнкции, дизъюнкции и т. д.). Эти операции либо предусмотрены системой команд микропроцессора, либо организуются программистом. В последнем случае длительность выполнения операции зависит от качества программы и на ее составление следует обратить особое внимание.

Операцию будем называть *простой,* если она имеется в системе команд микропроцессора, и *сложной ,* если она организуется программистом. Простыми являются операции сложения, вычитания, сдвига, конъюнкции, дизъюнкции. К сложным относятся, например, операции умножения и деления, образующиеся путем комбинации простых и требующие для своей реализации значительно бо́льших временны́х затрат. В ряде случаев сложная операция выполняется специализированным высокоскоростным модулем, однако включение таких модулей в систему требует дополнительных энергетических затрат, программного и аппаратного сопряжения с основным вычислителем. Поэтому в любом случае сложная операция дороже простой по временным и материальным затратам и количество таких операций в вычислительном процессе следует оптимизировать. Сказанное в первую очередь относится к операции умножения.

Существует много способов выполнения умножения. Рассмотрим наиболее употребительный и получивший название *алгоритма Бута.* Алгоритм Буга основан на одновременном анализе двух соседних разрядов множителя и может быть использован для чисел, заданных в дополнительных кодах. При этом произведение получается также в дополнительном коде сразу со знаком.

Для вывода алгоритма Бута рассмотрим множитель B , записанный в дополнительном коде в виде и-разрядного числа, старший разряд которого является знаковым:

$$[B]_{д} = b_{n-1} b_{n-2} \ldots b_0 .$$

Истинное значение числа равно $B = -2^{n-1} b_{n-1} + 2^{n-2} b_{n-2} + \ldots + 2b_1 + b_0 .$

Добавим к B величину $(b_{n-2}2^{n-2} - b_{n-2}2^{n-2}) + \ldots + (b_0 - b_0) + b_{-1}$, где

$b_{-1} = 0$. Прибавленная величина равна нулю и не меняет значения *B, но* позволяет записать:

$$B = (b_{n-2} - b_{n-1})2^{n-1} + (b_{n-3} - b_{n-2})2^{n-2} + \ldots + (b_0 - b_1)2 +$$

$$+ (b_{-1} - b_0).$$

Отсюда следует, что умножение сводится к умножению множимого на коэффициенты множителя $(b_{i-1} - bp = 0,1,-1$ и суммированию частных произведений по правилу, состоящему из трех частей: 1) если $b_{i-1} - b_i = 0$, то накапливается не множимое, а нулевая строка, т. е. суммирование как таковое отсутствует; 2) если $b_{i-1} - b_i = 1$, то множимое добавляется в накопитель; 3) если $b_{i-1} - b_i = -1$, то множимое вычитается из накопителя.

При сдвиге освобождающаяся позиция заполняется значением знакового разряда.

Пример 1.1. Перемножить числа $A = -3$ и $B = -5$. Прямые и дополнительные коды этих чисел равны:

$$[A]_{пр} = 1.011; \qquad [A]_R = 1.101;$$

$$[B]_{др} = 1.101; \qquad [B]_д = 1.011.$$

Найдем значения $Д = b_{i-1} - b_i$. Они равны

i	3	2	1	0
Д	-1	1	0	-1

Процедура умножения для 8-разрядного процессора со сдвигом частичных произведений вправо выглядит следующим образом:

Шаг 1
```
  00000000
+ 0011
----------
  00110000
  00011000
```
Начальное состояние регистра произведения. Так как Д = -1, производится вычитание числа -3, т. е. суммирование числа 3 и последующий сдвиг.

Шаг 2
```
  00001100
```
Так как Д = 0, производится только сдвиг.

Шаг 3
```
+ 1101
----------
  11011100
```
Так как Д = 1, производится суммирование числа —3, записанного в дополнительном коде, и сдвиг.

Шаг 4
```
  11101110
+ 0011
----------
  00011110
  00001111
```
Так как Д = —1, производится вычитание числа -3, т. е. прибавление числа 3 и последующий сдвиг.

Перенос из знакового разряда теряется.

13.3. Масштабирование и округление результатов счета

Использование арифметики с фиксированной запятой приводит к ограничению динамического диапазона обрабатываемых чисел. Поэтому при выпол-

нении вычислений следует своевременно производить масштабирование, т. е. исключать часть разрядов, выходящих за пределы разрядной сетки рабочих регистров. При этом возникает ошибка, равная

$$\delta = X - X',$$

ще X —исходноечисло; X' -преобразованноечисло.

Существует два метода выполнения этой операции — *усечение* и *округление.* При усечении m-разрядного числа до *п* разрядов младшие *т~п* разрядов исходного числа отбрасываются. Ошибка усечения $5_у$ удовлетворяет неравенству $2 \sim '' > 5_{\text{у}} > 0$.

При округлении те-разрядного числа до л-разрядного исходное число заменяется на ближайшее «-разрядное число. Практически это выполняется следующим образом. Если старший отбрасываемый разряд прямого кода равен нулю, то оставляются только первые «разрядов. Если же старший отбрасываемый разряд равен единице, то к оставшемуся числу добавляется единица. Ошибка округления $5_о$ удовлетворяет неравенству $-2^{\sim n \sim 1} < \S_о < 2^{\sim n \sim 1}$. Округление является более точным способом, чем усечение, поскольку при округлении максимальная ошибка равна половине шага квантования. Однако усечение имеет преимущество в скорости.

Следует отметить, что масштабирование результатов счета допустимо далеко не всегда. Например, если выполняются арифметические операции над смесью сигнала и шума при малом отношении сигнал/шум, то исключение младших разрядов приведет к потере информации о сигнале и получению неверного результата. В такой ситуации разрядная сетка процессора должна выбираться с учетом возможных значений сигнала и шума.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАЧИ

1.1. Что общего у дискретного и цифрового сигналов и в чем заключаются их различия?

1.2. Продискретизируйте и запишите в виде последовательности отсчетов в восьми точках следующие функции:

а) прямоугольный видеоимпульс с единичной амплитудой и длительностью, равной половине интервала наблюдения t'_1, т. е.

$$i(t) = \begin{cases} 1, & \text{при } 0 < t < t_1/2; \\ 0, & \text{при } t > t_1/2; \end{cases}$$

б) гармонические функции sincof, sin2cof, <J = 2rrД$_1$;

в) прямоугольный радиоимпульс с единичной амплитудой и длительностью, равной $t_1/2$, с частотой заполнения $CJ_2 = 4OJ$.

1.3. Какую частоту дискретизации следует выбрать для сигнала, прошедшего фильтр нижних частот с частотой среза 200 Гц?

1.4. Запишите в формате 1 байт числа -117, -103 в прямом, обратном и дополнительном кодах.

1.5. Перемножьте числа 7 (множимое) и -5 (множитель) по алгоритму Бута.

2. ЭФФЕКТИВНЫЕ АЛГОРИТМЫ ВЫПОЛНЕНИЯ БАЗОВЫХ ОПЕРАЦИЙ

1 1 1 ЭФФЕКТИВНОСТЬ АЛГОРИТМОВ И ОЦЕНКА ИХ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ

Вопросы сложности алгоритмов и устройств всегда были в центре внимания инженеров. Само понятие сложности эволюционировало с развитием техники и имеет много различных трактовок, отражающих как специфику устройства, так и уровень технического развития. Так, сложность первых радиоэлектронных устройств оценивали количеством содержащихся в них функциональных элементов. С развитием микроэлектроники отдельный элемент стал заменяться модулем — прибором, содержащим много элементов.

Понятие сложности алгоритма также допускает различные трактовки. Здесь могут учитываться или не учитываться такие факторы, как размер машинного слова, емкость памяти, различие в длительности выполнения отдельных команд и т. д.

К настоящему времени не существует достаточно универсального понятия сложности, которым можно пользоваться в любой ситуации. Вместе с тем создан ряд моделей вычислительного процесса, позволяющих раскрыть вычислительную сложность отдельных задач и сравнить различные алгоритмы. Наиболее употребительной является модель неветвящейся программы.

Неветвящаяся программа представляет собой программу без циклов, в которой цикл заменяется копированием повторяющейся команды соответствующее число раз. Число шагов такой программы как функция от размера входа N называется *временной сложностью,* а число переменных, участвующих в вычислениях, — *емкостной сложностью.*

Пример 2.1. Рассмотрим вычисление полинома

$$p(z) = a_N z^N + a_{N-1} z^{N-1} + \ldots + a_1 z + a_0 .$$

Можно записать:

$$a_1 z + a_0 , \; N = 1;$$

$$(a_2 z + a_1) z + a_0 , \; N = 2;$$

$$((a_3 z + a_2) z + a_1) z + a_0 , \; N = 3.$$

Неветвящаяся программа приведена в табл. 2.1.

Таблица 2.1

$N = 1$	$N = 2$	$N = 3$
$t \leftarrow a_1 z$	$t \leftarrow a_2 z$	$t \leftarrow a_3 z$
$p \leftarrow t + a_0$	$t \leftarrow t + a_1$	$t \leftarrow t + a_2$

Окончание табл.2.1

$N=1$	$N=2$	$N=3$
	$t \leftarrow tz$	$t \leftarrow tz$
	$p \leftarrow t + a_0$	$t \leftarrow t + a_1$
		$t \leftarrow tz$
		$p \leftarrow t + a_0$

Временная сложность этой программы равна *2N,* а емкостная $N+4$.

Дадим формальное определение неветвящейся программы. Пусть заданы:

1) набор входных переменных x_0 , x_1 ,..., x_{N-1} ;

2) кольцо *K* (или поле *F).* Будем считать, что кольцо и поле — это некоторое множество элементов, с которыми выполняются операции сложения и умножения (строгое определение кольца и поля дано в § 3.5);

3) множество *P* базисных операций $P = \{+ , x , /\}$ и $\{x \cdot /\}$, где + , x , / — двухместные арифметические операции сложения, умножения, деления, x *y* — одноместная операция умножения на элемент кольца или поля.

Неветвящаяся программа (НП) представляет собой последовательность строк (команд), *1-я* из которых имеет вид

$$z_l = f_l(z_{i_1}, \ldots, z_{i_u}, x_{j_1}, \ldots, x_{j_v}),$$

где I, $< /. \ldots i_u < I,$ /je *P.*

Дня любой базисной операции из множества *P* фиксируется число X(/), называемое *сложностью* этой операции. *Сложностью НПтзывается* сумма всех X *(f.)* по всем строкам этой программы.

Рассмотрим ряд модификаций этого понятия.

1. Пусть Л(/) = 1 для всякой / е P . Тогда соответствующая сложность "считает" число всех операций НП и называется *тотальной сложностью.* Обозначим ее C_t.

2. Пусть $\backslash(+) = 1$, $X(x) = X(/) = X(x\,y) = 0$, т. е. учитываются только операции сложения. Соответствующая сложность называется *аддитивной.* Обозначим ее C_a .

3. Пусть $X(x) = X(/) = 1$, а $X(+) = X(xy) = 0$, т. е. учитываются только нелинейные операции — умножение и деление. Сложность такого рода называется *мультипликативной.* Обозначим ее C .

Аддитивная сложность является хорошим критерием качества алгоритма при обработке бинарных или троичных сигналов, элементы которых кодируются в алфавитах (0,1), (1,-1), (1,0,-1). В этих случаях операции умножения и деления отсутствуют. Мультипликативная сложность обычно используется тогда, когда операция умножения существенно дороже операции сложения.

При тотальной сложности все операции оцениваются одинаково. Она используется при анализе вычислителя, построенного на матричных процессорах.

Заметим также, что сложность существенно зависит от выбора кольца K или поля F. Например, для вычисления выражения $x^2 + y^2$ в поле вещественных чисел необходимо выполнить два умножения, а в поле комплексных чисел только одно, так как $x^2 + y^2 = (x + jy)(x - jy)$.

При оценке качества алгоритма обычно оперируют асимптотической сложностью, т. е. величиной, которая получается при неограниченном увеличении размера входа. Она в итоге определяет размер задач, которые можно решить алгоритмом. Асимптотическая сложность оценивается порядком роста функции без учета мультипликативных констант. Например, если N ВХОДНЫХ переменных обрабатываются за время cN^2, где c — некоторая постоянная, то временная сложность этого алгоритма есть $O(7V^2)$ (читается: порядка $./V^2$).

Ошибочно думать, что значение эффективных алгоритмов уменьшится с ростом быстродействия вычислительных машин. В качестве примера рассмотрим пять алгоритмов A_1—A_5 различной сложности для решения одной и той же задачи:

Алгоритм	Временная сложность
A_1	N
A_2	$N \log_2 N$
A_3	N^2
A_4	N^3
A_5	2^N

В этих алгоритмах под временной сложностью будем понимать число единиц времени, требуемого для обработки входа размером N. Пусть, например, единицей времени будет одна миллисекунда. Тогда алгоритм $A^\wedge$ обработает за 1 с вход размером 1000, в то время как алгоритм A_s — вход размером не более 9. В табл. 2.2 приведены размеры задач, которые можно решить за различное время этими алгоритмами.

Таблица 2.2

Алгоритм	Временная сложность	Максимальный размер задачи		
		1 с	1 мин	1 ч
A_1	N	1000	$6\text{-}Ю^4$	$3{,}6\text{-}10^6$
A_2	$\mathrm{Mog_2}V$	140	4893	$20\text{-}10^8$
A_3	N^2	31	244	1897
A_4	N^3	10	39	153
A_S	2^N	9	15	21

Предположим, что быстродействие вычислительной машины увеличилось в 10 раз. В табл. 2.3 показано, как при этом возрастут размеры задач.

Таблица 2.?

Алгоритм	Временная сложность	Максимальный размер задачи	
		до ускорения	после ускорения
A_1	N	S_1	$10S_1$
A_2	$N\log_2 N$	S_2	$10S_2$ (при больших S_2)
A_3	N^2	S_3	$3{,}16S_3$
A_4	N^3	S_4	$2{,}15S_4$
A_5	2^N	S_5	$S_5 + 3{,}3$

Видно, что увеличение быстродействия приводит к существенному увеличению размера задачи только в алгоритмах с малой временной сложностью.

Предположим, что вместо увеличения быстродействия мы будем совершенствовать алгоритм. Рассмотрим табл. 2.2, взяв для сравнения колонку с временем решения 1 мин. Тогда, заменив алгоритм A_4 алгоритмомЛ$_3$.можно получить шестикратное увеличение размера задачи, а при замене алгоритма $A^\wedge$ на алгоритм A_2 размер задачи возрастает в 125 раз. Если выполнить сравнение за 1 ч, то различие окажется еще более существенным.

Приведенные примеры показывают, что асимптотическая сложность служит важной мерой качества алгоритма.

В практических приложениях все задачи имеют ограниченный размер. Поэтому, кроме порядка роста, следует учитывать и мультипликативную константу. Иногда больший порядок роста может иметь меньшую мультипликативную константу, и в этом случае такой алгоритм окажется предпочтительным при малых размерах задачи. Например, предположим, что временные сложности алгоритмов A_1 .A_2A_b , A_3 , A_5 равны соответственно 100Q/V, $100\text{Mog}_2\text{iV}$, ION^2 ,N^3n2. Тогда алгоритм A_5 будет наилучшим для задач размером $2 < N < 9$,A_3 - для задач размера $10^5 < N < 5\%$,A_2 для задач размером $59 < 7\text{V} < 1024$ и A_1 для задач размером $N > 1024$.

2.2. ВЫЧИСЛЕНИЯ С КОМПЛЕКСНЫМИ ЧИСЛАМИ

Рассмотрим два комплексных числа $a + jb$ и $c + jd$, где $/ = \sqrt{-1}$. Сумма этих чисел равна $(a + c) + /(b + d)$, и ее вычисление не представляет трудностей.

Для произведения обычно используется общеизвестная классическая формула

$$(a + jb)(c + jd) = (ac \sim M) + j(ad + be) .$$

Она позволяет найти действительную и мнимую части за четыре операции умножения (ac , bd , ad , be) и две операции сложения $(ac - bd , ad + be)$.

Другой способ состоит в следующем. Вычислим сначала промежуточные величины: $m_1 = (a + Ь)(c + d) = ac + ad + bc + bd$; $m_2 = ac$; $m_3 = bd$. Действительная часть произведения равна $ac - bd = m_2 - m_3$, а мнимая $ad + be = w_1 \sim m_2 - m_3$. При этом выполняются три операции умножения и пять операции сложения. Алгоритм вычислений имеет следующий вид:

$D1 \leftarrow a$, $D2 \leftarrow b$, $D3 \leftarrow c$, $D4 \leftarrow d$;

$S1 \leftarrow D1 + D2 = a + b$;

$S2 \leftarrow D3 + D4 = c + d$;

$S3 \leftarrow S1 \cdot S2 = (a + b)(c + d)$;

$S4 \leftarrow D1 \cdot D3 = ac$;

$S5 \leftarrow D2 \cdot D4 = bd$;

$S6 \leftarrow S4 - S5 = ac - bd$ (действительная часть);

$S7 \leftarrow S3 - S4 = ad + bc + bd$;

$S8 \leftarrow S7 - S5 = ad + bc$ (мнимая часть).

Пусть M и A — время, требуемое для выполнения операции умножения и сложения соответственно. Тогда второй способ лучше первого, если $3M + 5A < Ш + 2A, M/A > 3$.

Как уже указывалось, длительность операции умножения в микропроцессорах существенно больше длительности операции сложения. Поэтому применение этого способа в задачах с большим числом комплексных умножений дает заметное сокращение времени вычислений.

Рассмотрим еще один алгоритм. Его выгодно использовать, когда одно из чисел известно заранее. Вычислим:

$$m_1 = c(a + b);$$
$$m_2 = b(c + d);$$
$$m_3 = a(d - c).$$

Действительная и мнимая части равны:

$$ac - bd = m_1 - m_2;$$
$$ad + bc = m_1 + m_3.$$

Алгоритм требует выполнения трех операций умножения и пяти операций сложения. Однако если число $c + jd$ известно заранее, то можно предварительно вычислить суммы $c + d$ *ми* $- c$. При этом получается алгоритм с тремя операциями умножения и тремя операциями сложения.

2.3. ВЫЧИСЛЕНИЕ СТЕПЕНЕЙ

Рассмотрим задачу нахождения JV-й степени некоторого действительного числа x, т. е. вычисление выражения x^N, где N— целое число. Прямой путь состоит в последовательном получении степеней x^2, x^3, ..., x^N путем многократного умножения на x. Для вычисления N-*й* степени следует выполнить TV- 1 умножений.

Можно построить более экономичные алгоритмы. Предположим, что iV = $= 2^l$. Тогда для вычисления N-*й* степени достаточна..вьшодниДи/ = }о%$_2 N$ умножений. Например, x^{32} можно вычислит» заБМШУЮЩие~ пять ^игов: $x \backslash x \backslash x', x^{16}, x^{32}$.

Рассмотренный метод носит название *бинарного* (он был известен еще в древней Индии) и может быть обобщен на произвольное значение числа N. Для этого запишем /VB двоичном коде, например $N = 19 = (10011)_2$. Исключим цифру старшего разряда, которая всегда равна единице, а в остальных разрядах произведем замену нулей и единиц буквами X и SX по правилу: 1-+SX, 0 -* S. Для числа $N = 19$ получим $SSSXSX$. Эта последовательность букв дает правило вычисления x^N, если по букве S результат предыдущего шага возводить в квадрат, а по букве X умножать на x. Для нашего примера получим последовательность шагов x^2, x^4, x^5, x^9, x^{18}, x^{19}.

Оценим количество умножений. Двоичное представление числа ./Vсодержит $\lfloor \log_2 N \rfloor + 1$ бит, где $\lfloor a \rfloor$ — наименьшее целое, не превосходящее а. Пусть в этом представлении содержится v единиц. Буква S появляется при замене каждого бита, кроме первого, а буква X — при замене каждой единицы, кроме первой. Поэтому общее количество умножений равно $\lfloor \log_2 N \rfloor + v - 1$. Величина v имеет максимальное значение, равное $v_{max} = \lfloor \log_2 N \rfloor + 1$, если двоичное представление состоит из одних единиц, поэтому максимальное количество умножений равно $2\lfloor \log_2 N \rfloor$. Если N равно степени двойки, то $v = 1$, и в этом случае количество умножений минимально и равно $\log_2 N$.

Бинарный метод позволяет сократить количество умножений, однако не гарантирует минимума. Покажем это на примере. Пусть $N = 15 = (1111)_2$ ~* -> $SXSXSX$, что дает последовательность x^2, x^3, x^6, x^7, x^{14}, x^{15} »на вычисление которой затрачивается шесть операций умножения. То же самое можно сделать за пять операций следующим образом:*^{2}, x^3, x^6, x^{12}, x^{15}.

Сокращение количества умножений стало возможным благодаря тому, что число 15 имеет своими множителями числа 3 и 5. В общем случае составное число можно записать как $N = pq$. При таком представлении сначала вычисляется число $y = x^p$, а затем число $y = (x^p)^q = x^N$.

Описанный алгоритм называется *методом множителей* и формулируется следующим образом:

1) если $N = 1$, то вычислений не требуется;

2) если /V — простое, то для получения x^N сначала по методу множителей вычисляется x^{N-1}, а затем результат умножается на д:;

3) в остальных случаях число N представляется в виде $N = pq$, где p ~ наименьший простой делитель N и $q > 1$.

С помощью метода множителей сначала находится $y = x^p$, а затем $y^q = x^N$.

Пример 2.2. Пусть $N = 21 = 3 \cdot 7$. Тогда получим x^2, x^3, x^6, x^{12}, x^{15}, x^{21}.

Хотя метод множителей дает в среднем лучшие результаты по сравнению с бинарными, в некоторых случаях он может проигрывать последнему. Например, при $N = 33$ метод множителей требует семи операций умножения, а бинарный — только шести.

В ряде случаев оба рассмотренных метода оказываются неоптимальными. Например, если $N = 23$, то оба метода требуют семи операций умножения, однако x^{23} можно вычислить за шесть операций умножения следующим образом: x^2, x^3, x^5, x^{10}, x^{20}, x^{23}.

Поскольку использование любого из рассмотренных методов не гарантирует минимального количества умножений, для оценки степени их оптималь-

ности полезно определить абсолютный минимум, который может быть достигнут независимо от применяемого метода. Обозначим эту величину *l(N)*. Из предыдущих рассуждений легко получить, что $l(N) > \mathrm{riog}_2 AH$, где *Га~* - наименьшее целое, большее или равное *а* . Другими словами, в любом случае необходимо затратить не менее riog_2 ЛП умножений. Однако этого количества умножений может оказаться недостаточно. Бинарный метод гарантирует окончание вычислений за $2L\log_2 iVJ$ умножений, а метод множителей за */(7V) < 1(p) + /(<?)* умножений.

На рис. 2.1 показаны оптимальные алгоритмы с минимальным количеством умножений для $N < 100$ [6]. Эти алгоритмы изображены в виде дерева. Путь сверху вниз соответствует цепочке умножений. Например, для $N = 31$ получим путь 1-2-3-5-10-11—21-31, который соответствует последовательности операций: д: $x \cdot x = x^2$; $x^2 \cdot x = x^3$; $x^5 \cdot x^2 = x^5$; $x^5 \cdot x^5 = x^{10}$; $x^{10} \cdot x = x^{11}$; $x^{11} \cdot x^{10} = x^{21}$; $x^{21} \cdot x^{10} = x^{31}$.

Сравним рассмотренные методы с точки зрения загрузки памяти. Бинарный метод является самым экономичным, так как требует хранения в памяти только значения *х* и текущего промежуточного результата. Все остальные методы требуют хранения ряда промежуточных степеней переменной *х* .

В заключение заметим, что под *х* может пониматься не только действительное число, но и более сложный объект, например многочлен или матрица.

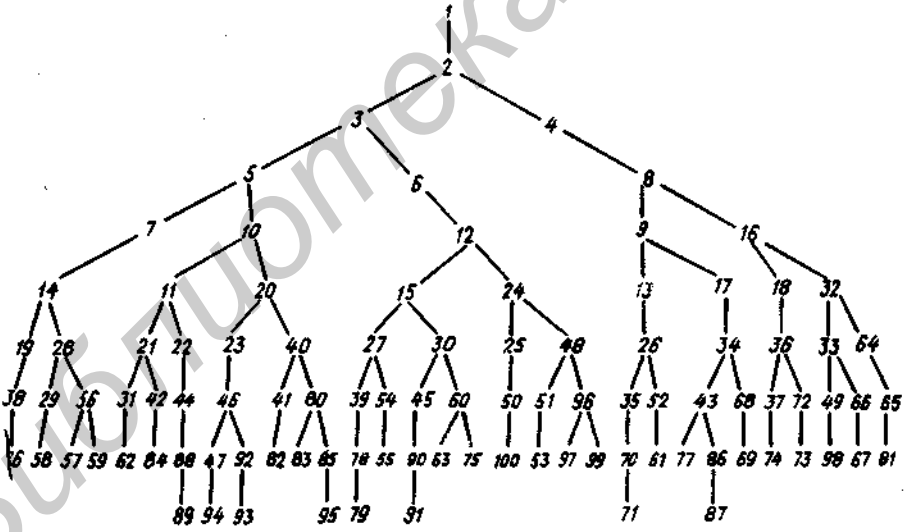


Рис- 2.1. Дерево, минимизирующее число умножений для ecexjV< 100.

2.4. ВЫЧИСЛЕНИЕ ПОЛИНОМОВ

\2.4Л. Метод Горнера

Произвольный полином степени Сможет быть записан как

$$p(z) = c_N z^N + c_{N-1} z^{N-1} + \ldots + c_1 z + c_0 . \qquad (2.1)$$

Наша цель состоит в определении величины $p(z)$ при фиксированных коэффициентах c_N, c_{N-x} ,..., c_Q и переменной z .

При непосредственном использовании формулы (2.1) следует выполнить $N-1$ умножений для вычисления всех степеней z , N умножений для определения произведений этих степеней на коэффициенты c_N, c_{N-s} , ..., c_1 RN сложений для суммирования произведений. Таким образом, вычислительная сложность равна $2N-1$ операциям умножения и /Уоперациям сложения.

Более экономичный способ дает *метод Горнера.* Метод был описан Вильямом Горнером в 1819 г., хотя фактически использовался еще Исааком Ньютоном в 1711 г. Представим полином $p(z)$ в виде $p(z) = (\ldots((c_N z + c_{N-1})z + c_{N-2})z + \ldots + c^{\wedge} z + c_Q$. Вычисления начнем с определения произведения $CflZ$, затем суммы $c_N z + c_{N-1}$, далее произведения $(c^{\wedge}z + c_{N-x})z$ и т. д. Нетрудно подсчитать, что метод Горнера требует не более N операций умножения и N операций сложения. Если полином не имеет каких-либо особенностей, то метод Горнера является оптимальным как по числу умножений, так и по числу сложений.

Полиномы специального вида можно вычислить и за меньшее количество операций. Например, рассмотрим полином

$$p(z) = z^3 + 4z^2 + 5z + 2 = (z+1)^2 (z+2).$$

Прямой метод требует четырех операций умножения и трех операций сложения. Если воспользоваться разложением на множители, то полином $p(z)$ можно вычислить только за две операции умножения и две операции сложения.

2.4.2. Вычисление полинома в точках

Рассмотрим задачу вычисления полинома в нескольких точках a, , $a_{\%}$,..., a_{κ} , $\kappa < N$. Положим сначала $z = a_x$. Можно записатьр(г) = (z -$a^{\wedge}$ $q,(,z)$ + + $r(z\backslash$ где $q(z)$ и $r(z)$ - частное и остаток от деления$p(z)$ на z - $a_{\%}$. Этот результат можно распространить на большее число точек. Рассмотрим произведение

$$m(z) = \prod_{i=1}^{k} (z - a_i)$$

и запишем $p(z) = m(z)q(z) + r(z)$. В точкеz = q. полином$m(z)$ равеннулт , ^ поэтому p Ц.) = Г(Й,.) . Теперь проблема вычисления полинома $p(z)$ свелась,/ к вычислению полинома $г$ (z), степень которого меньше.

Используем этот подход для постро быстрого алгоритма вычисление полинома степени $N-1$ в N точках, полагая для простоты $N = 2^l$. Раздеадл* /N точек на две половины и образуем полиномы

$$m_1(z) = \prod_{i=0}^{N/2} (z - a_i); \qquad m_2(z) = \prod_{i=(N/2)+1}^{N} (z - a_i) .$$

Разделим $p(z)$ на $m^{\wedge}z)$ и $m_2(z)$. При этом получим остатки ^(z) и $г_2$(§е-

пенл *N12.* Теперь проблема свелась к вычислению этих остатков в *N/2* точках, для вычисления остатков можно воспользоваться аналогичным приемом, повторяя его многократно.

Пример 2.3. Пусть требуется вычислить полином $p(z) = z^3 - 2i^2 + 3z + 1$ в точках z, равных -1, 0, 1, 2.

Образуем $m_1(z) = (z+1)(z^2 + z)$, $m_2(z) = (z-1)(z-2) = z^2 - 3z + 2$. После деления $p(z)$ на m_1, (*) и m_2 (^ получим $d_1(z) = 6z + 1$, $r_2(z) = 4z - 1$. Разделив $r_1(z)$ на $z+1$ и $z - 2$, запишем $p(-1) = -5, p(0) = 1$. Аналогично, разделив $r_2(z)$ на $z - 1$ и $z - 2$, найдем $P(1) = 3$, $p(2) = 7$.

Может показаться, что описанный алгоритм сложен из-за необходимости выполнять довольно трудоемкие операции деления и вычисления остатков. В действительности этот процесс оказывается весьма простым, если воспользоваться теорией сравнений.

2.4.3. Сравнения и вычеты

Пусть *ант*— целые числа, причем m — положительное. При делении числа a на число m получаются частное q и остаток r. Число a связано с ними соотношением $a = qm + r$, $0 \le r < m$.

Если a и b дают при делении на m один и тот же остаток, то говорят, что a сравнимо с b по модулю m. Число m называется модулем. При этом принята следующая запись: $a \equiv b \bmod m$.

Пример 2.4. Справедливы следующие сравнения: $5 \equiv 1 \bmod 2$, $8 \equiv 3 \bmod 5$, $8 \equiv -2 \bmod 5$.

Числа, сравнимые по модулю m, образуют класс сравнимых чисел. Всем числам этого класса соответствует один и тот же остаток. Этот остаток называется вычетом по модулю m и обозначается $< * >$. Если $a = baoAm$, то $<a> = <b>$.

Всего имеется m вычетов. Любой вычет, кроме нулевого, имеет два представления—положительное и отрицательное. Полная система вычетов может строиться как система неотрицательных вычетов и как система абсолютно минимальных вычетов.

Пример 2.5. Пусть $m = 5$. Вычеты равны: $<0> = <0>$, $<1> = <-4>$, $<2> = <-3>$, $<3> = <-2>$, $<4> = <-1>$. Система неотрицательных вычетов содержит числа 0, 1, 2, 3, 4, Система абсолютно минимальных вычетов представляется рядом —2, —1, 0, 1, 2.

Из определения вычета следует, что

$$<x \pm y> = <<x> \pm <y>>;$$
$$<xy> = <<x> \cdot <y>>.$$

Во всех приведенных определениях слово "число" можно заменить словом "полином". При этом получим систему полиномов, сравнимых по модулю некоторого полинома.

Теория полиномиальных сравнений во многом аналогична теории числовых сравнений. Приведем основные определения.

Полином $P(z)$ делит полином $R(z)$, если существует полином $D(z)$, та-

кой, что #(z) = *P(z)D(z).* Если *P(z)* не является делителем Я(z), то в результате деления *H(z)* на ?(z) получается частное и остаток *$$(z)* J#(z) = = Р(г)Д(г) +Л(г). Степень остатка меньше степени полинома *P*(г) (записывается degi? (z) < degP(z)).Всё полиномы, дающие при делении на Р(г) один и тот же остаток, называются сравнимыми по модулю *P(z).* Записывается *R(z)* = H(z)modP(z), а сам остаток называется полиномиальным вычетом,

(_ Вернемся теперь к задаче вычисления полиномов — остатков. Она сводится к приведению полинома p(z) по модулю полинома *m(z).* Пусть *m(z)* = $z^N + \sum_{i=0}^{N-i} m_i z^i$. Тогда по свойству сравнений m(z) =0, т. е. $z^N = -\sum_{i=1}^{7V-1} m_i z^i$

и, следовательно, z^N можно заменить полиномом — $-\sum_{i=0}^{} m_i z^i$. В частном случае, когда *m(z)* = *z -a,* имеем *p(z)* mod (z «в)= p(д) .

Пример 2. 6» Вычислим остаткиот деления полинома p(г) = z^3 - $2z^2$ + **3г+1** (из примера 2.3) на полиномы $m^$ (г) = z^2+ г, $m^$ (z) = z^3 - 3z + 2. Учитывая, что z^2 ^ -г для $т_а$ (z), получаем p(z)modm (z) = z^2 (z - 2) + 3z+ 1 = -z(z-2) + 3z + 1 = -z^2 -K2Z+ 3Z + + 1 = 6z + 1.

Аналогично для m_2 (z) записываем: p(z)modm$_2$(z)= z^2 (z ~2) + 3z + 1 = (3z - 2) X X **(z** - 2) + 3z + 1 = 3z^2 - 5z + 5 = **9JL-** 6, 5z + 5 = 4z -'*1

2.5. УМНОЖЕНИЕ ПОЛИНОМОВ

2.5.1. Алгоритм "разделяй и властвуй"

Для решения той или иной задачи ее разбивают на части и из их решения строят общее решение. Данный прием можно использовать рекурсивно, что позволяет получить довольно эффективное решение. Примером этого может служить рассмотренная в § 2.4 процедура вычисления полинома в точках. Описанный способ часто называют *алгоритмом "разделяй и властвуй".* Применим его для вычисления произведения двух полиномов.

Рассмотрим два полинома степени vV—1:

$$p(z) = \sum_{i=0}^{N-1} a_i z^i; \quad q(z) = \sum_{i=0}^{N-1} b_i z^i.$$

Их произведение равно

$$p(z)q(z) = \sum_{k=0}^{2N-2} c_k z^k,$$

где $c_k = \sum_{i=0}^{k} a_{k-i} b_i$.

Вычисление произведения *p(z)q(z)* фактически сводится к определению коэффициентов c_{fc} , каждый из которых равен сумме произведений ap_j с условием $i + j = \kappa$.

Число различных произведений вида $a_i b_j$ равно N^2 , а полином-произведение имеет *2N* — 1 коэффициентов. Поэтому сложность реализации прямого метода равна N^2 операциям умножения^2 — 27V+ 1 операциям сложения. Например, если p(z) = $a_x z + a_Q$ и *q* (z) = $b_y z + b_Q$, то произведение этих полиномов равно $a_i b_y z^2 + i.a_i b_Q + a_o bj)\ z + a_Q b_0$ • На вычисление затрачивается четыре операции умножения *(a$_1$b$_1$, a$_1$b$_Q$, a$_0$b$_e$,* $a_0 Ь_0$)иодна операция сложения $a_x Ь_0 + a_o b_1$. —"

Для построения более эффективного алгоритма рассмотрим сначала умножение полиномов первой степени *p(z)* = $a_i z + a_Q$, *q(z)* = $Ъ_x z + b_Q$. Произведение этих полиномов можно вычислить за три операции умножения (вместо четырех) по следующему алгоритму:

$$m_1 = a_0 b_0, \quad m_2 = a_1 b_1, \quad m_3 = (a_1 + a_0)(b_1 + b_0);$$
$$p(z)q(z) = m_2 z^2 + (m_3 - m_2 - m_1)z + m_1.$$

Нетрудно учидеть, что этот алгоритм совпадает с алгоритмом умножения комплексных чисел, если заменить z на / . Хотя в данном случае требуется четыре операции сложения (вместо одной), алгоритм оказывается полезным при умножении полиномов более высоких степеней. Для иллюстрации рассмотрим случай,когда*N*= 4,т. е. p(z)= $a_3 z^3 + a^{\wedge 2} + a^z + a_Q$, *q(Z)* -b^ +

" Прямой метод требует 16 операций умножения и 9 операций сложения. Разделим оба сомножителя на две части и представим их в виде:

$$p(z) = s(z)z^2 + t(z); \quad q(z) = u(z)z^2 + v(z),$$

где

$$s(z) = a_3 z + a_2; \quad u(z) = b_3 z + b_2;$$
$$t(z) = a_1 z + a_0; \quad v(z) = b_1 z + b_0.$$

Произведение этих полиномов равно *p(z)q(z)* =*j(z)u(z)z*^{4} + *(s(z)v(z)* + + f(Z)M(Z)) z^2 + f(z)y(z). Воспользуемся теперь рассмотренным способом умножения полиномов первой степени за три операции умножения. Запишем произведения:

$$m_1 = t(z)v(z), \quad m_2 = s(z)u(z); \quad m_3 = (s(z) + t(z))(u(z) + v(z)).$$

Искомый полином равен

$$p(z)q(z) = m_2 z^4 + (m_3 - m_2 - m_1)z^2 + m_1 =$$
$$= c_6 z^6 + c_5 z^5 + c_4 z^4 + c_3 z^3 + c_2 z^2 + c_1 z + c_0.$$

Полный алгоритм запишется следующим образом:

$$
\left.\begin{aligned}
M_0 &\leftarrow a_0 b_0;\\
M_2 &\leftarrow a_1 b_1;\\
d_1 &\leftarrow a_1 + a_0;\\
d_2 &\leftarrow b_1 + b_0;\\
d &\leftarrow d_1 d_2;\\
M_1 &\leftarrow d - M_2 - M_0
\end{aligned}\right\}
$$

вычисление полинома

$$m_1 = M_2 z^2 + M_1 z + M_0;$$

$$
\left.\begin{aligned}
L_0 &\leftarrow a_2 b_2;\\
L_2 &\leftarrow a_3 b_3;\\
P_1 &\leftarrow a_3 + a_2;\\
P_2 &\leftarrow b_3 + b_2;\\
P &\leftarrow P_1 P_2;\\
L_1 &\leftarrow P - L_2 - L_0
\end{aligned}\right\}
$$

вьиисление полинома

$$m_2 = L_2 z^2 + L_1 z + L_0;$$

$$
\left.\begin{aligned}
l_1 &\leftarrow a_1 + a_3;\\
l_0 &\leftarrow a_2 + a_0;\\
h_1 &\leftarrow b_3 + b_1;\\
h_0 &\leftarrow b_2 + b_0;\\
Q_0 &\leftarrow l_0 h_0;\\
Q_2 &\leftarrow l_1 h_1;\\
q_1 &\leftarrow l_1 + l_0;\\
q_2 &\leftarrow h_1 + h_0;\\
q &\leftarrow q_1 q_2;\\
Q_1 &\leftarrow q - Q_2 - Q_0
\end{aligned}\right\}
$$

вычисление полинома

$$m_3 = Q_2 z^2 + Q_1 z + Q_0;$$

$$
\left.\begin{aligned}
H_0 &\leftarrow Q_0 - L_0 - M_0;\\
H_1 &\leftarrow Q_1 - L_1 - M_1;\\
H_2 &\leftarrow Q_2 - L_2 - M_2;
\end{aligned}\right\}
$$

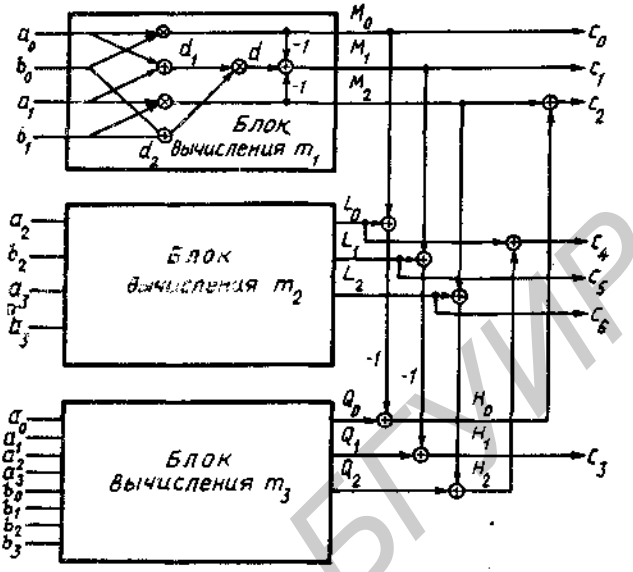
вычисление полинома

$$m_3 - m_2 - m_1 = H_2 z^2 + Hz + H_0.$$

Используя полученные выражения, можно записать:

$$
\begin{aligned}
c_6 &\leftarrow L_2 = a_3 b_3;\\
c_5 &\leftarrow L_1 = a_2 b_3 + a_3 b_2;\\
c_4 &\leftarrow L_0 + H_2 = a_3 b_1 + a_2 b_3 + a_1 b_3;\\
c_3 &\leftarrow H_1 = a_3 b_0 + a_2 b_1 + a_1 b_2 + a_0 b_3;\\
c_2 &\leftarrow H_0 + M_2 = a_2 b_0 + a_1 b_1 + a_0 b_2;\\
c_1 &\leftarrow M_1 = a_1 b_0 + a_0 b_1;\\
c_0 &\leftarrow M_0 = a_0 b_0.
\end{aligned}
$$

Рис- 2-2- Схема умножения полиномов третьей степени



На вычисление затрачивается 9 операций умножений и 24 операции сложения. Заметим, что алгоритм содержит три идентичных блока для вычисления $^{m}i \cdot {}^{m}2\ {}'{}^{m}э\ '$ Поэтому программа, написанная для одного блока с небольшими изменениями, может быть использована и для вычисления остальных блоков.

Схема, построенная по синтезированному алгоритму, приведена на рис. 2.2.

Чтобы обобщить этот метод на полиномы произвольной степени, положим для простоты, что N равно степени двух (с небольшими изменениями метод может быть использован для произвольного N). Опять разделим полиномы на две части, т. е.

$$p(z) = s(z)z^{N/2} + t(z), \quad q(z) = u(z)z^{N/2} + v(z),$$

и образуем величины:

$$m_1 = t(z)v(z);\ m_2 = s(z)u(z);\ m_3 = (s(z) + t(z))(u(z) + v(z)).$$

Для их вычисления можно повторять описанную процедуру деления вплоть до ЛГ= 1. Искомый результат имеет вид

$$p(z)q(z) = m_2 z^N + (m_3 - m_2 - m_1)z^{N/2} + m_1.$$

Таким образом, алгоритм сводит задачу большого размера к подзадачам меньшего размера с последующим объединением результатов. В свою очередь каждую из подзадач можно далее делить до самого простейшего случая.

Оценим эффективность этого алгоритма. Пусть $M(N)$ обозначает количество скалярных, умножений, необходимых для вычисления произведения двух полиномов с N коэффициентами. Используя на каждом шаге деления алгоритм с тремя умножениями, получаем

$$M(N) = 3M(N/2) = 3^2 M(N/4) = \ldots = 3^k M(N/2^k) .$$

Процесс деления прекращается, когда $\kappa = \log_2 N$. На последнем шаге используют только одну операцию умножения, т. е. $M\{1) = 1$. С учетом этого получим

$$M(N) = 3^{\log_2 N} = N^{\log_2 3} \simeq N^{1,59} .$$

Подсчитаем теперь количество операций сложений, обозначив эту величину *A (N).* В нее входят: а) операции при нахождении трех произведений полиномов с *N/2* коэффициентами. Количество их равао *3A (N/2)*; б) два сложения полиномов с *N/2* коэффициентами *(s(z) + t(z)* и *u(z) + v(z))* и два вычитания полиномов с *N—1* коэффициентами *(m₃ — m₂— m₁)*,что дает *2(N12) + + 2(N-\) =* 37V-2 операций; в) операции образования коэффициентов произведения из результатов предварительных вычислений. На это затрачивается *N—2* сложений.

Суммируя приведенные величины, получим *A (N) = 3A (N/2) + AN — 4* . При начальном условии *A* (1) = 0 последнее уравнение имеет решение *A (N) = = Ш*$^{\log_2 3}$ - 87V+ 2.

Приведенный анализ показывает, что как мультипликативная, так и аддитивная сложности операции умножения двух полиномов оцениваются величиной *O(№⁶⁹).* В то же время порядок роста сложности прямого метода умножения равен *OiN²).*

В табл. 2.4 приведены значения *M(N)* и *A (N)* для прямого метода умножения и алгоритма "разделяй и властвуй".

Таблица 24

N	Прямой метод			Алгоритм "разделяй и властвуй"		
	M(N)	*A(N)*	*M(N)+A(N)*	*M(N)*	*A(N)*	*M(N) + A(N)*
2	4	1	5	3	4	7
4	16	9	25	9	24	33
8	64	49	113	27	100	127
16	256	225	481	81	360	441
32	1024	961	1985	243	1204	1447
64	4096	3969	8065	729	3864	4593

Из табл. 2.4 видно, что мультипликативная сложность алгоритма "разделяй и властвуй" существенно меньше. При больших *N* меньше также и общая (тотальная) сложность.

25.2. Алгоритм Тоома-Кука и преобразование Фурье

Любой полином степени *N*— 1 можно задать либо коэффициентами, либо значениями в iV точках. Произведение двух поливдмов степени -/V–1 и *M—1* соответственно является полиномом степени *N+M—2* и может быть задано значе-

НЙЯМИ в $N+M-1$ точках. Это приводит к другому методу умножения полиномов. Пусть полином $p(z)$ имеет степень $N-1$, а полином $q(z)$ — степень $M-1$. Найдем значения $p(z_i)$, $q(z_i)$, полиномов в выбранных точках z_i, $i = 0, 1, ..., 0+M-2$). Затем вычислим произведения $y(z_i) = p(z_i)\,q(z_i)$. Искомый полином $y(z) = p(z)q(z)$ полностью определяется этими значениями и может быть восстановлен по величинам $y(z_i)$. Процедура восстановления называется *интерполяцией* и производится при помощи *интерполяционной формулы Лагранжа*

$$y(z) = \sum_{i=0}^{N+M-2} y(z_i)\,L_i(z).$$

Полиномы $L_i(z)$ называются *интерполяционными* и имеют вид

$$L_i(z) = \prod_{j \neq i} \frac{z - z_j}{z_i - z_j}.$$

Вычисление $y(z_i)$ требует одного умножения $y(z_i) = p(z_i)\,q(z_i)$, если известны $p(z_i)$ и $q(z_i)$. Таким образом, можно получить алгоритм, который требует только $N+M-1$ умножений, если интерполяция по формуле Лагранжа выполняется без умножений и вычисление $p(z_i)$, $q(z_i)$ также не требует умножений.

Как правило, эти условия не выполняются. Однако для полиномов малой степени количество умножений можно минимизировать путем удачного выбора точек интерполяции $z_0, z_1, ..., z_{N+M-2}$. Кроме того, данные операции умножения представляют собой умножения на фиксированные константы, поэтому выполняются просто.

Пример 2,7. Вычислим произведение полиномов $Y(z) = X(z)H(z)$, где

$$X(z) = x_2 z^2 + x_1 z + x_0;$$
$$H(z) = h_2 z^2 + h_1 z + h_0.$$

Для применения интерполяционной формулы Лагранжа выберем следующие точки интерполяции: $z_0 = 0$; $z_1 = 1$; $z_2 = -1$; $z_3 = 2$; $z_4 = -2$. Значения полинома $Y(z)$ в этих точках равны:

$$Y(0) = X(0)H(0) = x_0 h_0;$$
$$Y(1) = X(1)H(1) = (x_2 + x_1 + x_0)(h_2 + h_1 + h_0);$$
$$Y(-1) = X(-1)H(-1) = (x_2 - x_1 + x_0)(h_2 - h_1 + h_0);$$
$$Y(2) = X(2)H(2) = (4x_2 + 2x_1 + x_0)(4h_2 + 2h_1 + h_0);$$
$$Y(-2) = X(-2)H(-2) = (4x_2 - 2x_1 + x_0)(4h_2 - 2h_1 + h_0).$$

Интерполяционные полиномы имеют следующий вид:

$$L_0(z) = \frac{(z-1)(z+1)(z-2)(z+2)}{(0-1)(0+1)(0-2)(0+2)} = \frac{1}{4}(z^4 - 5z^2 + 4);$$

$$L_1(z) = \frac{(z-0)(z+1)(z-2)(z+2)}{(1-0)(1+1)(1-2)(1+2)} = -\frac{1}{6}(z^4 + z^3 - 4z^2 - 4z).$$

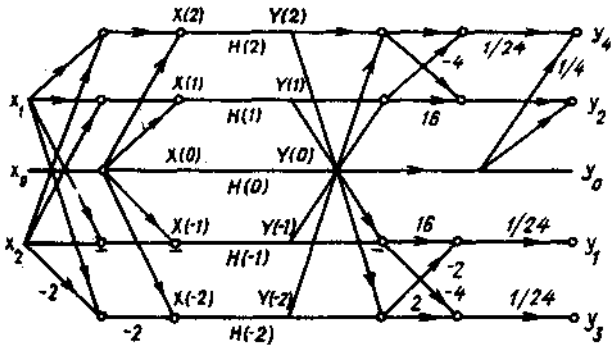


Рис. 2.3. Граф вычислительного процесса по алгоритму Тоома—Кука.

Аналогично получим:

$$L_2(z) = -\frac{1}{6}(z^4 - z^3 - 4z^2 + 4z);$$

$$L_3(z) = \frac{1}{24}(z^4 + 2z^3 - z^2 - 2z);$$

$$L_4(z) = \frac{1}{24}(z^4 - 2z^3 - z^2 + 2z).$$

После подстановки K(г$_i$) и $L_i(z)$ в интерполяционную формулу и группировки членов по степеням запишем:

$$y_0 = Y(0),\ y_1 = \frac{1}{24}(16Y(1) - 16Y(-1) - 2Y(2) + 2Y(-2));$$

$$y_2 = -\frac{5}{4}Y(0) + \frac{1}{24}(16Y(1) + 16Y(-1) - Y(2) - Y(-2));$$

$$y_3 = \frac{1}{24}(-4Y(1) + 4Y(-1) + 2Y(2) - 2Y(-2));$$

$$y_4 = \frac{1}{4}Y(0) + \frac{1}{24}(-4Y(1) - 4Y(-1) + Y(2) + Y(-2)).$$

В алгоритме используются пять операций умножения при вычислении $Y(z_i)$, умножение на масштабные коэффициенты 1/24, 5/4 и 1/4 и одна операция умножения $Y(zp$ на фиксированные константы. Эти константы удалось организовать так, что они являются степенями двойки, поэтому умножение на них сводится к сдвигу.

Во многих случаях один из сомножителей (например, H(z)) известен заранее. Это может быть, например, импульсная характеристика линейного фильтра (см. гл. 4). Тогда Я(г$_i$) вычисляется заранее и масштабные коэффициенты учитываются при этих вычислениях.

На рис. 2.3 приведенный алгоритм показан в виде графа.

Рассмотренный метод синтеза алгоритма достаточно общий, а сам алгоритм получил название *алгоритма Тоома-Кука*. Его центральной частью, определяющей эффективность, является выбор точек интерполяции. Один из возможных вариантов следующий:

$$z_i = \exp(-j\frac{2\pi}{N+M-1}i),\ i = 0,1,...,N+M-2.$$

о этом случае алгоритм Тоома—Кука эквивалентен умножению полиномов ри помощи дискретного преобразования Фурье и может быть интерпретирован следующим образом. Вычисление полиномов $p(z_i)$ и Ц.) в точках интерполяции z_i является просто преобразованием Фурье для коэффициентов этих полиномов. Далее найденные преобразования перемножаются поточечно, в результате чего получается iV+ Af — 1 произведений $p(z_i)\ q\ (z_i)$. Интерполяция по формуле Лагранжа эквивалентна выполнению обратного преобразования Фурье.

Далее покажем (см. гл. 3), что сложность вычисления преобразования фурье равна $O(N + M -1)log_2(N + M -1)$, что дает такую же асимптотическую сложность для произведения двух полиномов. Эта оценка асимптотически лучшая, но в практических приложениях следует всегда помнить о мультипликативных константах. Поэтому при не очень больших N, M предпочтительнее может оказаться алгоритм "разделяй и властвуй".

2.6. ВЕКТОРНО-МАТРИЧНОЕ И МАТРИЧНОЕ УМНОЖЕНИЕ

Пусть дана матрица А , состоящая из N_1 строк и N столбцов (матрица размером N^xN). Для вычисления произведения АХ, где $X = [x(0), x(1), ..., x(N-1)]^T$, стандартным строчно-столбцовым алгоритмом требуется затратить NN_1 операций умножения и $(N-1)N_1$ операций сложения.

В ряде случаев эту величину можно существенно уменьшить, если учесть особенности структуры матрицы А. Пусть, например, вычисляется произведение

$$Y = \begin{bmatrix} 1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix}.$$

Найдем предварительно суммы и разности:

$$a = x(0) + x(1);\quad c = x(2) + x(3);$$
$$b = x(0) - x(1);\quad d = x(2) - x(3).$$

Используя эти промежуточные результаты, получаем:

$$y(0) = a + d;\quad y(2) = b + c;$$
$$y(1) = -b + c;\quad y(3) = a - d.$$

На вычисление затрачивается 8 операций вместо 12 при обычном способе умножения.

Другим примером может служить матрица полного кода. Она имеет размер 2^N x N и содержит в качестве строк все двоичные числа от 0 до 1^N — 1. Без потери общности эти числа можно упорядочить по коду Грея, т. е. расположить так, чтобы соседние строки отличались только в одной позиции. Тогда

вычисление $y(i+1)$ по $y\{r\}$ требует не более двух операций сложения, а общее количество операций не превышает величины $1\text{-}2^N$. При прямом методе необходимо $(N-1)2^N$ операций. Несколько усложнив алгоритм, можно асимптотически довести число операций до величины $2^{N\sim 1}$ [9].

Важную роль в цифровой обработке сигналов играют матрицы Адамара порядка $N = 2^l$. Они строятся рекурсивно:

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad \mathbf{H}_{i+1} = \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_i \\ \mathbf{H}_i & -\mathbf{H}_i \end{bmatrix}.$$

В главе 3 построен алгоритм, позволяющий выполнить умножение вектора на матрицу Адамара за $\mathrm{Mog}_2 N$ операций сложения.

Перейдем к умножению матриц. Известно, что для вычисления произведения двух квадратных матриц порядка N требуется затратить N^3 операций умножения и $N^2(N-1) \sim N^3$ операций сложения. Штрассен предложил более эффективный алгоритм, позволяющий асимптотически сократить количество операций умножения и сложения до величины $O(TV^{2.81})$. В основе алгоритма Штрассена лежит способ перемножения матриц второго порядка за 7 операций умножения и 15 операций сложения. Заметим, что обычный метод требует 8 умножений и 4 сложения.

Алгоритм Штрассена состоит в следующем. Рассмотрим произведение матриц второго порядка

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}.$$

Сначала вычисляются:

$$\begin{aligned}
s_1 &= a_{21} + a_{22}; & m_1 &= s_2 s_6; & t_1 &= m_1 + m_2; \\
s_2 &= s_1 - a_{11}; & m_2 &= a_{11} b_{11}; & t_2 &= t_1 + m_4. \\
s_3 &= a_{11} - a_{21}; & m_3 &= a_{12} b_{21}; & & \\
s_4 &= a_{12} - s_2; & m_4 &= s_3 s_7; & & \\
s_5 &= b_{12} - b_{11}; & m_5 &= s_1 s_5; & & \\
s_6 &= b_{22} - b_{12}; & m_6 &= s_4 b_{22}; & & \\
s_8 &= s_6 - b_{21}; & m_7 &= a_{22} s_8; & &
\end{aligned}$$

Элементы произведения получаются по формулам:

$$\begin{aligned}
c_{11} &= m_2 + m_3; & c_{21} &= t_2 - m_7; \\
c_{12} &= t_1 + m_5 + m_6; & c_{22} &= t_2 + m_5.
\end{aligned}$$

Будем применять этот способ для вычисления произведения матриц порядка N. Для простоты рассуждений положим $N = 2^l$. Разобьем каждую из матриц на четыре равные подматрицы порядка $N/2$ и для вычисления искомого произведения воспользуемся приведенными равенствами. Аналогично матрицы порядка $N/2$ можно разбить на подматрицы порядка $N/4$ и т. д.

Пусть *M(N)* обозначает мультипликативную сложность вычисления произведения двух матриц порядка *N*. Тогда можно записать, что

$$M(N) = 7M(N/2) = 7^2 M(N/4) = \ldots = 7^k M(N/2^k) =$$
$$\ldots = 7^{\log_2 N} M(1) = N^{\log_2 7} M(1).$$

Так как $\log_2 7 = 2{,}81$ и $M(1) = 1$, то $M(N) = N^{2{,}81}$.

Рассмотрим теперь аддитивную сложность, обозначив ее *A (N)*. Для *A (N)* справедливо рекуррентное соотношение $A(N) = 7A(N/2) + 15(N/2)^2$. При начальном условии $A(1) = 0$ решение этого уравнения равно $A(N) = 5(N^{2{,}81} - 4)$.

Если интересоваться только скоростью роста, то получим

$$M(N) = A(N) = O(N^{2{,}81}).$$

Если *N* не является степенью числа два, то можно вложить исходную матрицу в матрицу, порядок которой равен наименьшей степени числа два, большей N. Это увеличит порядок матриц не более чем вдвое и отразится на мультипликативной константе, которая возрастает не более чем до семи. Однако порядок роста функций *M(N) nA(N)* сохранится прежним.

Приведенные границы сложности являются асимптотическими. Поэтому возникает вопрос, при каких значениях *N* метод Штрассена может быть рекомендован для практических расчетов. Ответ на этот вопрос можно получить лишь после конкретизации типа процессора.

Были предприняты многочисленные попытки улучшить границы Штрассена по порядку. Лучшим результатом является оценка $M(7) < O(N^{2{,}376})$, К сожалению, она представляет лишь теоретический интерес из-за большого размера матриц и мультипликативной константы.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАЧИ

2.1. Составьте неветвящуюся программу для умножения двух матриц второго порядка.

2.2. Приведите примеры вычислительных процессов, эффективность которых удобно оценивать аддитивной, мультипликативной и тотальной сложностью.

2.3. Какой выигрыш во времени можно получить, если для вычислений использовать алгоритм со сложностью $N\log_2 N$ вместо алгоритма со сложностью N^2 при *N,* равном 128 и 1024?

2.4. Когда выгодно использовать алгоритм умножения комплексных чисел с тремя операциями сложения и тремя операциями умножения?

2.5. Вычислите степени x^{39}, x^{51} бинарным методом и методом множителей. Результаты сравните с рис. 2.1.

2.6. Докажите, что метод Горнера требует не более ^умножений и -N сложений для вычисления полинома степени *N.*

2.7. Найдите вычеты: $47^{50} \bmod 3$, $3^{62} \bmod 19$, $(1010111001101)_2 \bmod 31$.

2.8. Найдите вычеты: $\sum_{i=1}^{5} a_i z^i \bmod (z^3 - 1)$, $z^8 + 2z^3 + 2z^2 + z + 1 \bmod z^4 + z + 1$.

2.9. Постройте неветвящиеся программы умножения двух полиномов второй степей^ используя прямой метод и алгоритм "разделяй и властвуй". Сравните полученные резуль таты с результатами примера 2.6.

2.10. Перемножьте два полинома третьей степени при помощи алгоритма Тоома—Кука,

2.11. Постройте алгоритм умножения вектора на матрицу полного кода длины 4, ее держащий не более 12 операций сложения (вычитания).

2.12. Используя алгоритм Штрассена, перемножьте две матрицы четвертого порядка

3. ОБРАБОТКА СИГНАЛОВ С ПОМОЩЬЮ ДИСКРЕТНЫХ ОРТОГОНАЛЬНЫХ ПРЕОБРАЗОВАНИЙ

3.1. ПРЕДСТАВЛЕНИЕ СИГНАЛОВ ФУНКЦИОНАЛЬНЫМИ РЯДАМИ

Дискретный сигнал х(и), заданный на интервале из N точек 0, 1, 2, ..., дг—1, можно записать в виде

$$s(n) = \sum_{k=0}^{\infty} c(k)\eta_k(n), \qquad (3.1)$$

где

$$c(k) = E_k^{-1} \sum_{n=0}^{N-1} s(n)\eta_k(n) . \qquad (3.2)$$

Здесь {%("")} ~ совокупность базисных функций, определенных на том же интервале;{с(£)}- коэффициенты разложения сигнала по базису /%(n)1 , называемые спектром сигнала; E^ — энергия *к-тй* базисной функции.

Из выражений (3.1), (3.2) следует, что сигнал можно задать либо его отсчетами в дискретных точках, либо набором спектральных коэффициентов, причем спектральное представление неоднозначно и зависит от выбранной системы базисных функций. Выбор системы £т?£ *(n)*} определяется соображениями практического или математического удобства.

Наибольший интерес для задач цифровой обработки представляет случай, когда число базисных функций конечно и равно размерности сигнала, *T.e.N.* Далее будем рассматривать только такие системы.

В разложении (3.1) необходимо, чтобы базисная система удовлетворяла следующим требованиям [12].

1. Функции системы должны быть линейно независимыми:

$$a_0\eta_0(n) + a_1\eta_1(n) + \ldots + a_{N-1}\eta_{N-1}(n) \neq 0$$

при любых значениях коэффициентов a_1 , кроме случая, когда $a_0 = a_1 = \ldots =$

Наиболее часто в качестве базисной системы линейно независимых функций выбирают ортогональные функции, т. е. функции, для которых выполняется условие

$$\sum_{n=0}^{N-1} \eta_k(n)\eta_l^*(n) = 0, \quad k \neq l .$$

__ В этом случае сигнал {«(и)} можно рассматривать как вектор в декартовой системе координат, осями которой являются базисные функции. Спектральные коэффициенты равны проекциям векторов на соответствующие оси (Рис. 3.1).

3 Зак. 5302

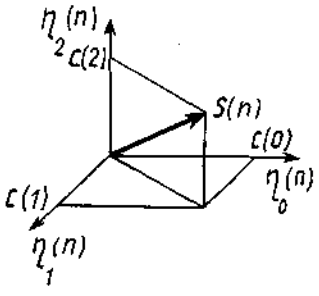
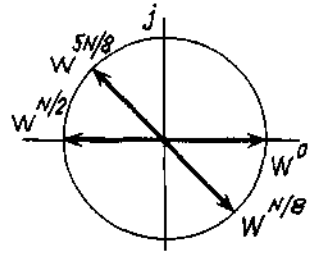


Рис. 3.1. Представление сигнала *S* (я) в пространстве ба-

Рис. j,2. Поворачивающие множители ДПФ.



2. Система {TJ^. *(n)*j- должна быть упорядоченной. Это значит, что на множестве индексов *к* введено отношение порядка, показывающее, какая из функций предыдущая и какая последующая. Например, синусоидальные и косинусоидальные функции обычно располагают в порядке возрастания аргумента (частоты) : 1, cos* , cos2x ,..., *coskx*. От введенного отношения порядка зависят форма спектра и удобство работы с ним. Например, при спектральном анализе по Фурье систем с ограниченной полосой пропускания часто отбрасываются составляющие спектра с высокими частотами.

3. Базисные функции на интервале ортогональности должны иметь конечную энергию, т. е.

$$\sum_{n=0}^{N-1} \eta_k^2(n) < \infty .$$

4. Система функций Ь?^ (и) } должна быть полной. Это означает, что к ней больше нельзя добавить ни одной функции, которая была бы ортогональна всем остальным функциям системы.

3.2. ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ

3.2.1. Дискретные экспоненциальные **функции**

Дискретное преобразование Фурье (ДПФ) устанавливает связь между временны́м и частотным представлениями сигнала при разложении его в ряд (3.1) по гармоническим функциям. Оно имеет многочисленные приложения в спектральном и корреляционном анализе, в синтезе фильтров, устройств об^наружения или оценки параметров сигналов. К настоящему времени разработаны эффективные методы вычисления ДПФ, позволяющие решать задачи, ранее считавшиеся недоступными из-за своей вычислительной сложности.

В дискретном преобразовании Фурье используется система дискретных экспоненциальных функций (ДЭФ), определяемых следующим выражением:

$$\mathrm{def}(k, n) = \exp(-j\frac{2\pi}{N}kn) = \cos\frac{2\pi}{N}kn - j\sin\frac{2\pi}{N}kn . \qquad (3.3)$$

Обе переменные *кип* принимают дискретные значения 0, 1, 2, ..., *N—1*. Переменную *к* , как правило, отождествляют с номером функции, а переменную *n* - с номером отсчета.

Обозначим $W = \exp(-j\frac{2\pi}{N})$. Тогда def(£, *n*) = $\dot{W}^{*n}$. Величина $\dot{W}^{kn}$ обычно называется *поворачивающим множителем* и представляет собой вектор на комплексной плоскости (рис. 3.2).

Всю систему ДЭФ можно записать в виде матрицы V , строки которой нумеруются переменной *к* , столбцы переменной *n,* а в пересечении *к-й* строки и и-го столбца записана величина W^{kn}:

$$\mathbf{V} = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ k \\ \vdots \\ N-1 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & \dots & n & \dots & N-1 \end{array} \\ \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & W^{kn} & & \\ & & & & & \\ & & & & & \end{bmatrix} \end{array} \qquad (3.4)$$

Например, для *N*= 8 матрица V имеет следующий вид:

$$\begin{bmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ W^0 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ W^0 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ W^0 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ W^0 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ W^0 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{bmatrix} =$$

$$= \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^1 & W^4 & W^7 & W^2 & W^5 \\ W^0 & W^4 & W^0 & W^4 & W^0 & W^4 & W^0 & W^4 \\ W^0 & W^5 & W^2 & W^7 & W^4 & W^1 & W^6 & W^3 \\ W^0 & W^6 & W^4 & W^2 & W^0 & W^6 & W^4 & W^2 \\ W^0 & W^7 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{bmatrix}. \qquad (3.5)$$

В теории и практических приложениях ДПФ одинаково важно как экспоненциальное (3.3), так и матричное представление (3.4), поэтому в дальнейшем эти две формы записи будем рассматривать совместно.

Системе ДЭФ присущи следующие свойства.

1. Ортогональность:

$$\sum_{n=0}^{N-1} W^{kn} W^{-ln} = \begin{cases} N, & \text{если } (k-l) \equiv 0 \bmod N; \\ 0, & \text{если } (k-l) \not\equiv 0 \bmod N. \end{cases} \qquad (3.6)$$

Свойство ортогональности показывает, что скалярное произведение любых двух строк матрицы V , одна из которых взята с комплексно сопряженными элементами, равно нулю, если строки различны, и равно *N,* если они совпадают. Матричная запись этого свойства имеет следующий вид:

$$\mathbf{V}(\mathbf{V}^T)^* = N\mathbf{I}, \qquad (3.7)$$

где знак * означает взятие комплексного сопряжения для всех элементов матрицы: Г — единичная матрица.

2. Периодичность. Если *кп = N1 + г* , то $W^{kn} = W^{N1+r} = W^r = W$, что позволяет записать элементы матрицы V с минимальными степенями (фазами), как это сделано в (3.5).

3. Симметричность:

$$\mathbf{V} = \mathbf{V}^T. \qquad (3.8)$$

Свойство симметричности позволяет легко найти обратную матрицу для матрицы V . С учетом (3.8) матричное выражение для ортогональности запишется так: VV* = М. Умножив обе части этого равенства справа на V^{-1} , получим

$$\mathbf{V}^{-1} = N^{-1}\mathbf{V}^*. \qquad (3.9)$$

4. Мультипликативность:

$$\begin{aligned} &\mathrm{def}(k_1, n)\,\mathrm{def}(k_2, n) = \mathrm{def}(k_1 + k_2, n); \\ &\mathrm{def}(k, n_1)\,\mathrm{def}(k, n_2) = \mathrm{def}(k, n_1 + n_2). \end{aligned} \qquad (3.10)$$

Свойство (3.8) означает, что при умножении любых двух строк (столб-

ц о в ы) матрицы V получается соответственно строка (столбец) той же матрицы. Номер строки (столбца) равен сумме номеров сомножителей.

32.2. Дискретное преобразование Фурье и его свойства

Пара дискретного преобразования Фурье последовательности $\{s(w)\} = \{s(0), s(l)', —, s(N—l)l\}$ определяется следующими равенствами:

$$f(k) = \sum_{n=0}^{N-1} s(n) W^{kn}, \quad k = 0, 1, ..., N-1; \tag{3.11}$$

$$s(n) = N^{-1} \sum_{k=0}^{N-1} f(k) W^{-kn}, \quad n = 0, 1, ..., N-1. \tag{3.12}$$

Последовательность $\{s(n)\}$ представляет собой отсчеты сигнала, а последовательность $\{/(\&)\}$ - дискретный спектр. Выражение (3.11) называется *прямым преобразованием,* а выражение (3.12) — *обратным.* Взаимная обратимость этих преобразований доказывается подстановкой *f(k)* в выражение (3.12) для х(и), а именно:

$$N^{-1} \sum_{k=0}^{N-1} (s(0) + s(1) W^{k} + ... + s(n) W^{kn} + ... + s(N-1) W^{k(N-1)}) W^{-kn} =$$

$$= N^{-1} \sum_{l=0}^{N-1} s(l) \sum_{k=0}^{N-1} W^{k(l-n)}.$$

В силу ортогональности ДЭФ (см. (3.6), (3.7)) вторая сумма отлична от нуля и равна *N,* если только / - *n* ≡ О, что дает *s* («).

Равенства, (3.11) ," (3.12) представляют собой экспоненциальную форму записи ДПФ. Соответствующее матричное представление имеет вид:

$$\mathbf{F} = \mathbf{V}\mathbf{S}; \tag{3.13}$$

$$\mathbf{S} = N^{-1} \mathbf{V}^{*} \mathbf{F}, \tag{3.14}$$

ВД S = [s(o), i(1), ..., *s(N-l))'; F* = [ДО), Д1), ..., /(iV-1)1' -векторы-столбцы отсчетов сигнала и спектральных коэффициентов соответственно. Равенство (3.14) получается из равенства (3.13), если воспользоваться выражением (3.9), а именно: $S = V^{*1} F = TT^{1} V^{*} F$.

Рассмотрим наиболее важные свойства ДПФ.

1. Периодичность. В силу периодичности ДЭФ функции *f(k)* и х(и) также периодичны, т. е.

$$f(Nl \pm k) = f(\pm k); \quad s(Nl \pm n) = s(\pm n). \tag{3.15}$$

2. Связь с коэффициентами ряда Фурье. Если частота дискретизации выбрана в соответствии с теоремой Котельникова, то при дискретизации периодической аналоговой функции *s(t)* ДПФ позволяет по выборкам *s(n)* найти спектр *f(k),* который на интервале $0 < \kappa < N — 1$ равен спектру исходной

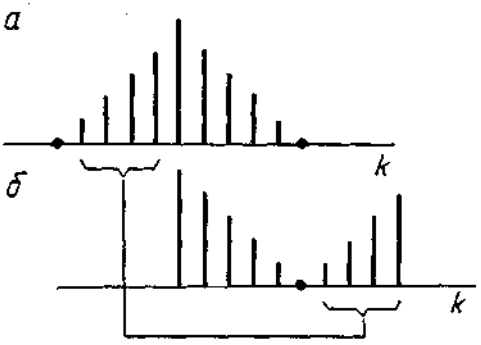


Рис. 3.3. Соотношение между коэф^ циентами ряда Фурье *(а)* и ДПФ (б).

функции *s(t)*. При этом первые *(NJ2* - 1) точек функции/(Аг) соответствуй спектральным линиям на положительных частотах (рис. 3.3), а последит *(N/2* - 1) точек *f(k)* — спектральным линиям на отрицательных частотам

В обратном преобразовании первые *(N/2* - 1) линий функции *s (ri)* СООТ ветствуют области положительных времен, а последние *(N/2* - 1) линий - области отрицательных времен.

3. Линейность. Пусть даны последовательности *x(ri) яу(n),* для которы ДПФ равны соответственно $f_x(к)$ $u/_y(к)$.

Рассмотрим взвешенную сумму этих последовательностей *z (ri) = ax(n) \ + Ьу(n).* Спектр последовательности *z* (и) равен аналогачной взвешенной сум ме спектров последовательностей *x* («) и *y (u),* т. е.

$$f_z(k) = af_x(k) + bf_y(k), \tag{3.16}$$

4. Инвариантность относительно сдвига по времени и частоте. Пусть после довательность *z (ri)* образована сдвигом по времени последовательное!! *s(ri),i.* е. *z(ri)= s(n ±h).* Тогда

$$f_z(k) = W^{\mp kh} f_x(k). \tag{3.17}$$

Свойство (3.17) показывает, что при сдвиге по времени амплитудный спектр (величина амплитуд отдельных гармоник) не меняется. Изменениям подвергаются только фазы гармонических составляющих (фазовый спектр)

Аналогичное свойство справедливо и для обратного преобразования, s именно: если *I(к) = f(k ± И)*, то

$$s_F(n) = W^{\pm nh} s_I(n). \tag{3.18}$$

Выражение (3.18) является аналогом (3.17) для временной области.

5. Теорема о свертке. Рассмотрим две последовательности (х(и)} и *{h(n)* Циклической сверткой этих последовательностей называется последователь- ность {у (и)} , значения которой определяются следующими равенствами:

$$y(n) = \sum_{l=0}^{N-1} h(l)\, s(n-l), \quad n = 0, 1, \ldots, N-1, \tag{3.19}$$

номера отсчетов берутся по модулю *N*. Последнее эквивалентно тому, что г д е ...едов ательности (s(«)} и *{h(n)j* периодические. Поэтому *s(-ri) = s(N-n),* *h* \ равенства (3.19) можно записать ив матричной форме:

$$\begin{bmatrix} y(0) \\ y(1) \\ \cdot \\ \cdot \\ \cdot \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & \dots & h(N-1) \\ h(1) & h(2) & \dots & h(0) \\ h(2) & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ h(N-1) & h(0) & \dots & h(N-2) \end{bmatrix} \begin{bmatrix} s(0) \\ s(N-1) \\ s(N-2) \\ \cdot \\ \cdot \\ s(1) \end{bmatrix}. \qquad (3.20)$$

Пример 3.1. Пусть $N = 4$. Тогда

$$y(0) = h(0)s(0) + h(1)s(-1) + h(2)s(-2) + h(3)s(-3) =$$
$$= h(0)s(0) + h(1)s(3) + h(2)s(2) + h(3)s(1);$$

$$y(1) = h(0)s(-1) + h(1)s(0) + h(2)s(-1) + h(3)s(-2) =$$
$$= h(1)s(0) + h(2)s(3) + h(3)s(2) + h(0)s(1).$$

Аналогично

$$y(2) = h(2)s(0) + h(3)s(3) + h(0)s(2) + h(1)s(1);$$

$$y(3) = h(3)s(0) + h(0)s(3) + h(1)s(2) + h(2)s(1).$$

Пусть *jf* (^)}и/^ *(Щ* -'Спектры последовательностей [S(W)}H *{h* (и)} соответственно. Теорема о свертке утверждает, что спектр свертки равен произведению спектров сворачиваемых последовательностей, т. е.

$$/,(*) = /,(*)£(*) \cdot \qquad (3\text{-}21)$$

Теорема (3.21) позволяет вычислить свертку при помощи ДПФ по формуле

$$[y(л)) = \text{ДПФ}^{-1}\ (\text{ДДф}\{з(n)\} \quad \text{ДПФ}\{\text{й } Щ\}) . \qquad (3.22)$$

Так как для вычисления ДПФ существуют быстрые алгоритмы,то этот способ часто оказывается более экономичным, чем прямое вычисление с использованием формул (3.19), (3.20).

(^Теорема о корреляции^Периодическая корреляционная функция двух поеледовательностей {s(и)j и *{h(n)* равна

$$г(и) = \sum_{/=0}^{N} h(l)s(n+l), \quad n = 0,1,\dots,N-1 , \qquad (3.23)$$

Где аналогично п.5 номера отсчетов берутся по модулю TV.

Соответствующее матричное представление имеет вид

$$\begin{bmatrix} r(0) \\ r(1) \\ \cdot \\ \cdot \\ \cdot \\ r(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & \dots & h(N-1) \\ h(N-1) & h(0) & \dots & h(N-2) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ h(1) & h(2) & \dots & h(0) \end{bmatrix} \begin{bmatrix} s(0) \\ s(1) \\ \cdot \\ \cdot \\ \cdot \\ s(N-1) \end{bmatrix}. \qquad (3.24)$$

Корреляционную функцию можно рассматривать как свертку, в которой одна из последовательностей обращена во времени, т. е. прочитана в обратном порядке, за исключением нулевого отсчета.

Спектр корреляционной функции последовательностей {r (и)} и {s(и)} равен произведению их спектров, причем один из спектров берется в комплексном сопряжении, т. е.

$$f_r(k) = f_h^*(k) f_s(k) = f_h(k) f_s^*(k) . \qquad (3.25)$$

Аналогично предыдущему для вычисления корреляционной функции на основе равенства (3.25) можно использовать ДПФ, что часто оказывается более эффективным, чем прямое вычисление по формулам (3.23), (3.24).

7. ДПФ вещественных последовательностей. Во многих случаях преобразуемая последовательность $s(n)$ вещественная. Ее ДПФ равно

$$f(k) = \sum_{n=0}^{N-1} s(n)\cos\left(\frac{2\pi}{N} kn\right) - j \sum_{n=0}^{N-1} s(n)\sin\left(\frac{2\pi}{N} kn\right).$$

Отсюда можно сделать следующие выводы:

а) спектральные коэффициенты комплексно сопряжены относительно отсчета $N/2$, т. е.

$$f(N/2 + l) = f^*(N/2 - l), \quad l = 0, 1, ..., N/2, \qquad (3.26)$$

а коэффициент $f(N/2)$ для четных N всегда действителен;

б) если последовательность $s(n)$ четная, т. е. $s(n) = s(-n)$, то ее ДПФ $f(k)$ — вещественная последовательность. Аналогично, если $s(n)$ нечетная, т.е. $s(n) = -s(-n)$, то ее ДПФ $f(k)$ есть чисто мнимая последовательность.

8. Равенство Парсеваля:

$$\sum_{n=0}^{N-1} s^2(n) = N^{-1} \sum_{k=0}^{N-1} |f(k)|^2 . \qquad (3.27)$$

Соотношение (3.27) показывает, что энергия сигнала равна суммарной энергии спектральных компонент.

Понятие ДПФ можно обобщить, если вместо последовательности $\{s(n)\}$ рассматривать многомерный массив. В частности, для случая двух измерений преобразуется массив

$$\mathbf{S} = \{s(n_1, n_2)\} = \begin{bmatrix} s(0,0) & s(0,1) & \dots & s(0, N_2-1) \\ s(1,0) & s(1,1) & \dots & s(1, N_2-1) \\ \dots & \dots & \dots & \dots \\ s(N_1-1,0) & s(N_1-1,1) & \dots & s(N_1-1, N_2-1) \end{bmatrix}.$$

Его ДПФ равно

$$f(k_1, k_2) = \sum_{n_2=0}^{N_2-1} W_2^{n_2 k_2} \sum_{n_1=0}^{N_1-1} s(n_1, n_2) W_1^{n_1 k_1} ,$$

2π 2π

$W_{N_1} = \exp(-j\frac{2\pi}{N_1})$, $W_{N_2} = \exp(-j\frac{2\pi}{N_2})$. Матричная форма этого выражения запишется так:

$$F = V_1 S V_2,$$

где V_1, V_2 — матрицы одномерных ДПФ с числом отсчетов N_1 и N_2 соответственно, а

$$F = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N_2-1) \\ f(1,0) & f(1,1) & \dots & f(1,N_2-1) \\ \dots & \dots & \dots & \dots \\ f(N_1-1,0) & f(N_1-1,1) & \dots & f(N_1-1,N_2-1) \end{bmatrix}.$$

Нетрудно видеть, что для получения двухмерного преобразования сначала выполняется преобразование V_1 столбцов массива S, а затем преобразование V_2 строк массива $V_1 S$.

3.3. БЫСТРЫЕ МЕТОДЫ ВЫЧИСЛЕНИЯ ДПФ

3.3.1. Алгоритм с прореживанием по времени

Вычисление ДПФ по формулам (3.11)—(3.14) требует выполнения N^2 операций умножения и $N(N-1)$ операций сложения комплексных чисел. Квадратичная зависимость объема вычислений от размера входной реализации является существенным препятствием при практическом использовании ДПФ. Известны более эффективные алгоритмы, чем прямой счет по указанным формулам. Они позволяют снизить вычислительную сложность до величины порядка $N\log_2 N$ и даже N. Эти алгоритмы получили название быстрого преобразования Фурье (БПФ). Существует много алгоритмов БПФ. Рассмотрим алгоритм с прореживанием по времени.

Пусть $\{s(n)\} = \{s(0), s(1), \dots s(N-1)\}$ - последовательность отсчетов, содержащая $N = 2^v$ элементов (v — целое). Разобьем ее на две части, выделив отдельно четные и нечетные отсчеты, т. е.

$$\{s(2n)\} = \{s(0),\ s(2),\ \dots,\ s(N-2)\};$$
$$\{s(2n+1)\} = \{s(1), s(3), \dots,\ s(N-1)\}.$$

Дискретное преобразование Фурье этой последовательности равно

$$f(k) = \sum_{l=0}^{N/2-1} s(2l) W^{2lk} + \sum_{l=0}^{N/2-1} s(2l+1) W^{(2l+1)k} =$$

$$= \sum_{l=0}^{N/2-1} s(2l) (W^2)^{lk} + W^k \sum_{l=0}^{N/2-1} s(2l+1) (W^2)^{lk} =$$

$$= G(k) + W^k H(k), \qquad (3.28)$$

где

$$G(k) = \sum_{l=0}^{N/2-1} s(2l)\,(W^2)^{lk},$$

$$H(k) = \sum_{l=0}^{N/2-1} s(2l+1)\,(W^2)^{lk}$$

есть дискретные преобразования Фурье четных и нечетных отсчетов.

Функция *f(k)* содержит Лоточек и имеет период *N*. В то же время функции *G(k)* и *H(к)* содержат по *N/2* точек и имеют период *N/2*. Для вычисления спектральных коэффициентов при $к < N/2$ можно непосредственно воспользоваться выражением (3.28). Для $к > N/2$ на основании периодичности ДПФ (см. (3.15)) и ДЭФ получим:

$$G(k) = G(k + \frac{N}{2}); \quad H(k) = H(k + \frac{N}{2});$$

$$W^{(k+\frac{N}{2})} = -W^k.$$

С учетом этого можно записать:

$$\begin{aligned} f(k) &= G(k) + W^k H(k), \quad 0 \leqslant k \leqslant N/2 - 1; \\ f(k) &= G(k) - W^k H(k), \quad N/2 \leqslant k \leqslant N - 1. \end{aligned} \tag{3.29}$$

Следовательно, все значения *f(k)* получаются вычислением двух *N/2* точечных преобразований с последующим их весовым сложением. Эта процедура показана на рис. *ЗА,а* для $N = 8$.

Заметим, что вычисление *G (к)* и *H(к)* прямым методом требует только $(N/2)^2$ операций умножения. Еще *N/2* операций выполняются приумножении на поворачивающие множители W^k, $к = 0, 1,..., N/2-1$. Общее количество операций уменьшается, таким образом, до величины $N^2/2 + N/2 < N^2$. Дальнейшее сокращение количества операций происходит, если для вычисления *G (к)* и *H(к)* опять "разрядить" их входные последовательности и продолжать этот процесс до тех пор, пока размер ДПФ не станет равным двум и дальнейшее прореживание будет невозможным (рис. 3.4, *б)*. Очевидно, что число шагов (итераций) этого процесса равно $\log_2 N$. На рис. 3.4, *в* показана процедура снижения размерности преобразования и общий граф вычислительного процесса для ЛГ=8.

Отметим следующие важные особенности вычислительного процесса.

1. Граф содержит $N \log_2 N$ узлов, в каждом из которых происходит суммирование или вычитание данных (см. (3.29)). Поэтому общее количество операций сложения (вычитания) равно $N \log_2 N$. Половина исходных данных на каждой итерации умножается на поворачивающие множители, поэтому общее количество комплексных умножений не превышает величины $0{,}5N \log_2 N$. Более подробное исследование показывает, что часть этих умножений тривиальна (умножения на $\pm 1, \pm j$).

2. Процедура многократного прореживания приводит к тому, что исходные данные располагаются не в естественном, а в *двоично-инверсном порядке-*

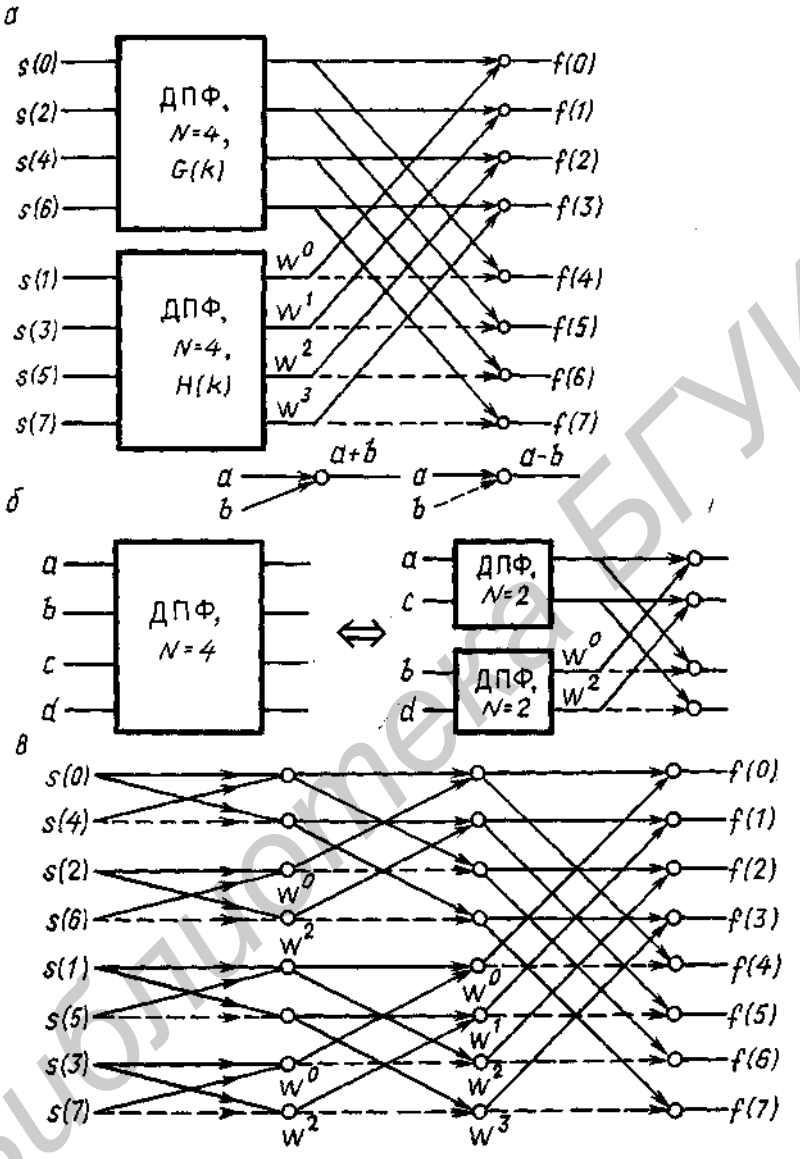


Рис. 3.4. БПФ с прореживанием по времени.

Например, для $N = 8$ естественный идвоично-инверсный порядок расположения отсчетов запишется следующим образом:

Естественный порядок	Двоично-инверсный порядок
000(0)	000(0)
001(1)	100(4)
010(2)	010(2)
011(3)	110(6)
100(4)	001(1)
101(5)	101(5)
110(6)	011(3)
111(7)	111(7)

3. Граф алгоритма имеет регулярную структуру и строится из пары базовых операций, которая получила название "бабочка". "Крылья" такой бабочки увеличиваются вдвое на каждой последующей итерации.

4. Регулярность графа позволяет производить вычисления с замещением (с оставлением на местах), т. е. записывать промежуточные результаты каждой итерации на место исходных. Действительно, на первой итерации каждая пара входных узлов воздействует только на соответствующую пару узлов, находящихся непосредственно справа за ними. Поэтому если вычисления производятся одновременно двумя узлами, то результаты могут быть записаны в те же ячейки, откуда взяты исходные данные.

На второй итерации эта ситуация сохраняется с той лишь разницей, что работающая пара узлов разделена одним промежуточным узлом. На третьей итерации работающая пара разделена тремя промежуточными узлами и т. д.

Построенный алгоритм имеет и матричную трактовку. В частности, для рассмотренного примера простой проверкой легко убедиться, что вычисляется матричное выражение

$$F = VS = \Phi_1 \Phi_2 \Phi_3 PS, \qquad (3.30)$$

где

$$P = \begin{bmatrix} 1 & & & & & & & \\ & & & 1 & & & & \\ & & 1 & & & & & \\ & & & & & & 1 & \\ & 1 & & & & & & \\ & & & & & 1 & & \\ & & & & 1 & & & \\ & & & & & & & 1 \end{bmatrix}$$

есть матрица двоичной перестановки исходных данных, а

$$\Phi_1 = \begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & W & & \\ & & 1 & & & & W^2 & \\ & & & 1 & & & & W^3 \\ 1 & & & & -1 & & & \\ & 1 & & & & -W & & \\ & & 1 & & & & -W^2 & \\ & & & 1 & & & & -W^3 \end{bmatrix};$$

$$\Phi_2 = \begin{bmatrix} 1 & & 1 & & & & & \\ & 1 & & W^2 & & & & \\ 1 & & -1 & & & & & \\ & 1 & & -W^2 & & & & \\ & & & & 1 & & 1 & \\ & & & & & 1 & & W^2 \\ & & & & 1 & & -1 & \\ & & & & & 1 & & -W^2 \end{bmatrix};$$

$$\Phi_3 = \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix}$$

есть матрицы вычислительного процесса на итерациях. Каждая из этих матриц имеет в строке и столбце только два ненулевых элемента, т. е. является слабозаполненной. Благодаря большому количеству нулевых элементов умножение на слабозаполненные матрицы требует меньших вычислительных затрат, чем на полные матрицы.

Соотношение (3.30) показывает, что матрица ДПФ разбивается на слабозаполненные сомножители. Процедура разбиения матрицы на сомножители называется *факторизацией*. Таким образом, с матричной точки зрения алгоритм БПФ является просто следствием факторизации матрицы дискретного преобразования Фурье.

3.3.2. Алгоритм с прореживанием по частоте

Разделим исходную последовательность (s(и) ⌊ на две части **по** *N/2* отсчетов в каждой. Первые *N/2* отсчетов обозначим *iq* (и) ⌋ , вторые — (й («И, т.е.

$$q(n) = s(n),\ h(n) = s(\frac{N}{2} + n),\ n = 0, 1, ..., (N/2 - 1).$$

Теперь ДПФ можно записать так:

$$f(k) = \sum_{n=0}^{N/2-1} (q(n) W^{kn} + h(n) W^{(n+\frac{N}{2})k}) =$$

$$= \sum_{n=0}^{N/2-1} (q(n) + (-1)^k h(n)) W^{kn}.$$

Рассмотрим отдельно четные и нечетные отсчеты функции *f(k) :*

$$f(2k) = \sum_{n=0}^{N/2-1} (q(n) + h(n)) (W^2)^{kn},$$

$$f(2k+1) = \sum_{n=0}^{N/2-1} (q(n) - h(n))\, W^n (W^2)^{kn}.$$

Эти выражения являются *N12*-точечными ДПФ от функций *q (n)* + *h* (и) ;, *[q* (и) - *h (ri)]W"* . Поэтому вычисление спектра можно производить по схеме, показанной на рис. 3.5, *а* (дляА^ = 8). Аналогично предыдущему процедуру последовательного деления входных отсчетов на две части можно продолжить дальше до получения двухточечных преобразований (рис. 3.5, *б).* Для этого потребуется $\log_2 N$ шагов. Результирующий граф преобразования при этом бу. дет содержать Mog_2iVузлов. Для *N*= 8 он приведен на рис. 3.5, *в* .

Как и в алгоритме с прореживанием по времени, в данном алгоритме вы. числительные затраты составляют $N\backslash og_2 N$ операций сложения и 0,5Mog^ операций умножения, а вычисления можно выполнять с замещением. Однако исходные отсчеты при этом располагаются в естественном порядке, а спект. ральные коэффициенты — в двоично-инверсном. Величина "бабочки" по мере продвижения к концу вычислений уменьшается. В соответствующем матрич. ном представлении матрицы-сомножители имеют иной вид:

$$\mathbf{F} = \mathbf{VS} = \mathbf{P'\Phi'_1\Phi'_2\Phi'_3\ddot{S}}, \qquad (3.31)$$

что дает другой вариант факторизации матрицы **V** .

3.3.3. Алгоритмы БПФ с произвольным основанием

Рассмотрим ^-точечное ДПФ.

$$f(k) = \sum_{n=0}^{N-1} s(n) W^{kn}, \quad k = 0, 1, ..., N-1. \qquad (3.32)$$

Пусть *N*— составное число. Допустим $N = N_1 N_2$. **Преобразуем** индексы *n* и *к* следующим образом:

$$n = N_1 n_2 + n_1, \quad n_1, k_1 = 0, 1, ..., N_1 - 1, \qquad (3.33)$$

$$k = N_2 k_1 + k_2, \quad n_2, k_2 = 0, 1, ..., N_2 - 1. \qquad (3.34)$$

Подставляя (3.33) и (3.34) в (3.32), получаем

$$f(N_2 k_1 + k_2) = \sum_{n_1=0}^{N_1-1} W^{N_2 n_1 k_1} W^{n_1 k_2} \sum_{n_2=0}^{N_2-1} s(N_1 n_2 + n_1) W^{N_1 n_2 k_2}.$$

Обозначим $W_1 = W^{N_2} \exp(-j\frac{2\pi}{N_1})$, $W_2 = W^{N_1} \exp(-j\frac{2\pi}{N_2})$. С учетом этого можно записать

$$f(N_2 k_1 + k_2) = \sum_{n_1=0}^{N_1-1} W_1^{n_1 k_1} W^{n_1 k_2} \sum_{n_2=0}^{N_2-1} s(N_1 n_2 + n_1) W_2^{n_2 k_2}.$$

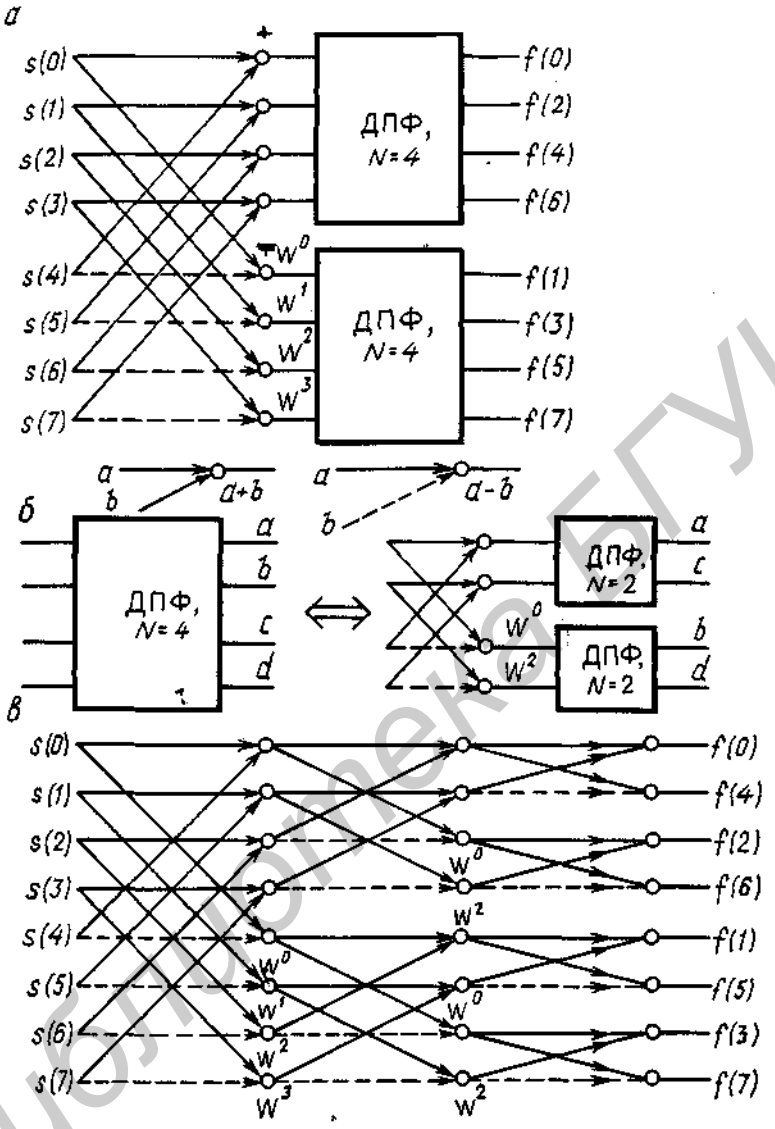


Рис. 3.5. БПФ с прореживанием по частоте.

Из этого выражения следует, что Л^Л^-точечное ДПФ можно вычислить в три этапа. Сначала вычисляются N_1 преобразований:

$$y(n_1, k_2) = \sum_{n_2=0}^{N_2-1} s(N_1 n_2 + n_1) W_2^{n_2 k_2},$$

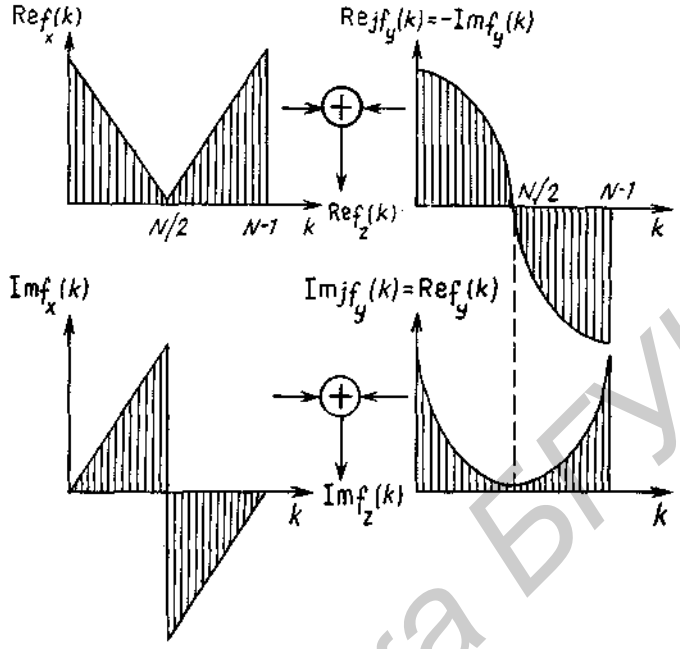


Рис. 3.6. Вычисление ДПФ действительных последовательностей.

Здесь $\kappa = 1,2,\ldots, (N/2 - 1)$ и вычисляется часть спектра, соответствующая положительным частотам.

Для $\kappa \Phi$ О, $N12$ имеем

$$\begin{aligned} &\mathrm{Re}\, f_x(0) = \mathrm{Re}\, f_z(0), \quad \mathrm{Re}\, f_x\left(\frac{N}{2}\right) = \mathrm{Re}\, f_z(N/2); \\ &\mathrm{Re}\, f_y(0) = \mathrm{Im}\, f_z(0), \quad \mathrm{Re}\, f_y(N/2) = \mathrm{Im}\, f_z(N/2). \end{aligned} \tag{3.38}$$

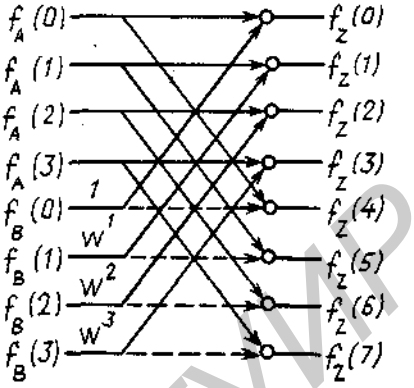
Для выполнения этих вычислений требуется затратить 4 (JV/2) = 27Voпераций сложения. Еще $\mathrm{Mog}_2./\mathrm{V}$ операций сложения и ^V/21$\log_2$^операций умножения затрачивается на вычисление преобразования функции z (и).

При раздельном преобразовании функций x (и) и y («) число операций умножения равно $Nlog_2N$, а число операций сложения $2\mathrm{Mog}_2 7\mathrm{V}$. Таким образом, выигрыш по числу операций умножения равен двум, а по числу операций сложения 21o^ $N1$ ($\log_2 7\mathrm{V} + 2$).

Рассмотрим теперь возможность сокращения длины преобразования. Пусть дана последовательность {x (И)} . Выделим в ней четные и нечетные отсчеты, т.е. произведем прореживание по времени. Далееобразуем $N/2$ точечную последовательность 2 («) · = $x(2n) + jx$ (2и+ 1) и вычислим ее ДПФ.

Из спектра $f_2(\kappa)$ можно найти спектры последовательностей^(2w) j- и {x(2«+1)} (см. § 3.3.1). Для получения спектра $f_x(\kappa)$ следует произвести

Рис. 3,7. Весовое суммирование при вычислении ДПФ действительных последовательностей.



их весовое суммирование, как это делалось в алгоритме с прореживанием по времени (см. рис. 3.4).

Пример 3.2. Вычислить спектр действительной последовательности $\{x(n)\} = \{1, 1, 1, 1, -1, -1, -1, -1\}$ при помощи ДПФ размерности $N = 4$.

Образуем последовательности четных и нечетных отсчетов:

$$\{A(n)\} = \{x(2n)\} = \{1, 1, -1, -1\}, \quad \{B(n)\} = \{x(2n+1)\} = \{1, 1, -1, -1\}$$

и комплексную последовательность

$$\{z(n)\} = \{A(n)\} + j\{B(n)\} = \{1+j,\ 1+j,\ -(1+j),\ -(1+j)\}.$$

ДПФ этой последовательности равно

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1+j \\ 1+j \\ -(1+j) \\ -(1+j) \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 0 \\ 4j \end{bmatrix} = 4 \begin{bmatrix} 0 \\ 1 \\ 0 \\ j \end{bmatrix}.$$

По формулам (3.37), (3.38) получим

$$\mathrm{Re} f_A(k) = 2\{0, 1, 0, 1\}; \quad \mathrm{Re} f_B(k) = 2\{0, 1, 0, 1\};$$
$$\mathrm{Im} f_A(k) = 2\{0, -1, 0, 1\}; \quad \mathrm{Im} f_B(k) = 2\{0, -1, 0, 1\}.$$

Объединяя действительные и мнимые части, записываем полные спектры последовательностей $\{A(n)\}$ и $\{B(n)\}$:

$$f_A(k) = f_B(k) = \{0, 2(1-j), 0, 2(1+j)\}.$$

Для получения спектра последовательности $\{x(n)\}$ выполним весовое суммирование спектров $f_A(k)$ и $f_B(k)$ (рис. 3.7). В результате получим

$$f_z(0) = 0;$$
$$f_z(1) = 2(1-j)(1+W^1);$$
$$f_z(2) = 0;$$

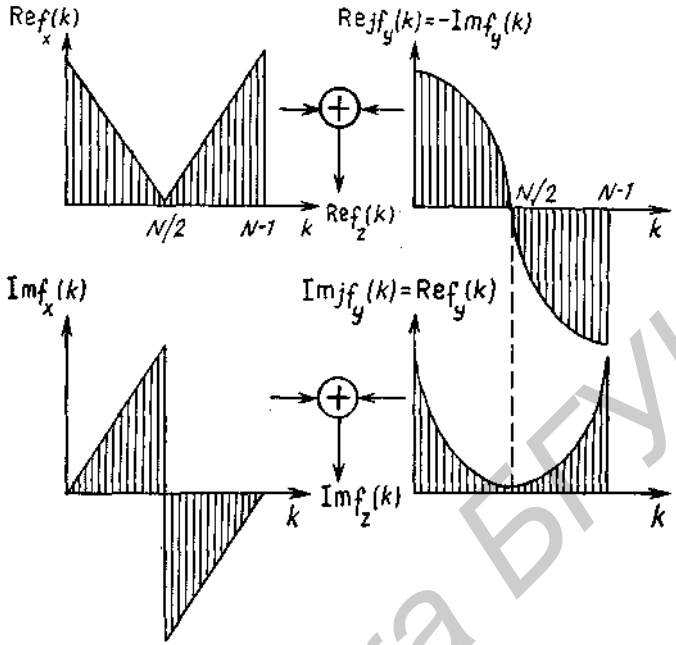


Рис. 3.6. Вычисление ДПФ действительных последовательностей.

Здесь $к = 1, 2,\ldots, (N/2 - 1)$ и вычисляется часть спектра, соответствующая положительным частотам.

Для $к Ф$ О, $7V/2$ имеем

$$\begin{aligned} &\mathrm{Re}\, f_x(0) = \mathrm{Re}\, f_z(0), \quad \mathrm{Re}\, f_x\left(\frac{N}{2}\right) = \mathrm{Re}\, f_z(N/2); \\ &\mathrm{Re}\, f_y(0) = \mathrm{Im}\, f_z(0), \quad \mathrm{Re}\, f_y(N/2) = \mathrm{Im}\, f_z(N/2). \end{aligned} \tag{3.38}$$

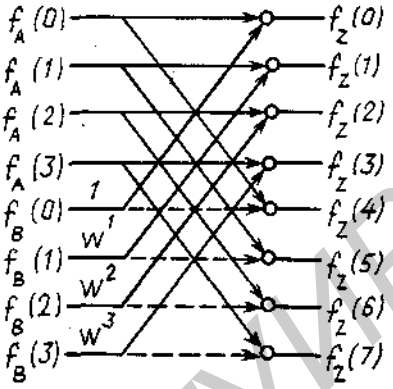
Для выполнения этих вычислений требуется затратить 4 (7V/2) = 2Wone-раций сложения. Еще $N\backslash og_2N$операций сложения и 7V/21og_2Л/операций умножения затрачивается на вычисление преобразования функции z (и).

При раздельном преобразовании функций x (и) и y (и) число операций умножения равно $N\backslash o\%_2N$, а число операций сложения 2Mo$g_2$7V. Таким образом, выигрыш по числу операций умножения равен двум, а по числу операций сложения 21o§2 Nj (log$_2$7V + 2).

Рассмотрим теперь возможность сокращения длины преобразования. Пусть дана последовательность {л: (n)\ . Выделим в ней четные и нечетные отсчеты, т.е. произведем прореживание по времени. Далее образуем $N/2$ точечную последовательность $z(ri) = x(2ri) + jx(2n + 1)$ и вычислим ее ДПФ.

Из спектра $f_z(к)$ можно найти спектры последовательностей£л:(2я) j- и $[x(2n+1)]$ (см. § 3.3.1). Для получения спектра $f_x(к)$ следует произвести

Рис. 3.7. Весовое суммирование при вычислении ДПФ действительных последовательностей.



их весовое суммирование, как это делалось в алгоритме с прореживанием по времени (см. рис. 3.4).

Пример 3.2. Вычислить спектр действительной последовательности $\{x(и)\} = \{1, 1, 1, 1, -1, -1, -1, -1\}$ при помощи ДПФ размерности $N = 4$.

Образуем последовательности четных и нечетных отсчетов:

$$\{Л(и)\} = \{x(2и)\} = \{1, 1, -1, -1\}, \quad \{B(n)\} = \{x(2л+1)\} = \{1, 1, -1, -1\}$$

и комплексную последовательность

$$\{z(и)\} = \{A(И)\} + j\{B(и)\} = (1 + j), \ (1 + j), \ -(1 + j), \ -(1 + j)j.$$

ДПФ этой последовательности равно

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 + j \\ 1 + j \\ -(1 + j) \\ -(1 + j) \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 0 \\ 4j \end{bmatrix} = 4 \begin{bmatrix} 0 \\ 1 \\ 0 \\ j \end{bmatrix}.$$

По формулам (3.37), (3.38) получим

$$\mathrm{Re} f_A(k) = 2\{0, 1, 0, 1\}; \quad \mathrm{Re} f_B(k) = 2\{0, 1, 0, 1\};$$
$$\mathrm{Im} f_A(k) = 2\{0, -1, 0, 1\}; \quad \mathrm{Im} f_B(k) = 2\{0, -1, 0, 1\}.$$

Объединяя действительные и мнимые части, записываем полные спектры последовательностей $\{A(и)\}$ и $\{B(n)\}$:

$$f_A(k) = f_B(k) = \{0, 2(1-j), 0, 2(1+j)\}.$$

Для получения спектра последовательности $\{z(и)\}$ выполним весовое суммирование спектров $f_A(k)$ и $f_B(k)$ (рис. 3.7). В результате получим

$$f_z(0) = 0;$$
$$f_z(1) = 2(1-j)(1 + W^1);$$
$$f_z(2) = 0;$$

$$f_z(3) = 2(1+j)(1+W^3);$$
$$f_z(4) = 0;$$
$$f_z(5) = 2(1-j)(1-W^1);$$
$$f_z(6) = 0;$$
$$f_z(7) = 2(1+j)(1-W^3);$$

где

$$W = \exp(-j\frac{2\pi}{8}).$$

3.4. ФУНКЦИИ УОЛША И ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ УОЛША-АДАМАРА

3.4.1. Матрицы Адамара и функции Уолша

Функции Уолша образуют полную ортонормированную систему с количеством функций 2^I, $I = 1,2,\ldots$, и значениями ± 1. Они были открыты Уолшем в 1923 г., однако в матричной форме построены еще Сильвестром в 1867 г. Адамар в 1883 г. показал, что эти матрицы принадлежат классу матриц с максимально возможным определителем. Поэтому они получили название матриц Адамара типа Сильвестра. Изучение функций Уолша удобно начать с изучения именно этих матриц.

О п р е д е л е н и е . Матрицей Адамара называется ортогональная квадратная матрица порядка N, элементами которой являются действительные числа ± 1. Матрица Адамара порядка N обозначается H_N.

Простейшей матрицей Адамара является матрица второго порядка:

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Для построения матриц порядка $N = 2^l$, $l = 2, 3,\ldots$ (матриц типа Сильвестра) используется следующая теорема.

Т е о р е м а -3.1. Если $И^\wedge$ — матрица Адамара порядка N, то матрица

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix}$$

является матрицей Адамара порядка $2N$.

Например, матрицы четвертого и восьмого порядков имеют соответственно следующий вид:

$$\mathbf{H}_4 = \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & -\mathbf{H}_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix};$$

$$H_8 = \begin{bmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

В более компактном виде $H_4 = H_2 \times H_2$, $H_5 = H_2 \times H_4$. Описанная процедура построения носит название *прямого* или *кронекерова умножения.*

Пронумеруем строки и столбцы матрицы Адамара двоичными числами от О до *N*— 1. Пусть k = *(k_{l} k_{2} ... к_{0})* - двоичный номер столбца. Тогда элемент матрицы, стоящий на пересечении строки с номером *к* и столбца с номером и, вычисляется по формуле

$$had<>,«) = (-1)^{(k\,n)}, \qquad (3.39)$$

где $(k\text{-}n) = \sum_{i=0} k_i n_i$ — скалярное произведение векторов **кип.**

Например, для матрицы H_8 с учетом (3.39) получим

$$\begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix} \begin{pmatrix} n_2 \\ n_1 \\ n_0 \end{pmatrix} = \mathbf{n}^I$$

$$\mathbf{k} = (k_2\ k_1\ k_0) \begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{matrix} \begin{bmatrix} & & \\ & & \\ - - & -(-1)^{(\mathbf{k}\cdot\mathbf{n})} & - - - - - \\ & & \\ & & \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} had\,(000, n) \\ had\,(001, n) \\ had\,(010, n) \\ had\,(011, n) \\ had\,(100, n) \\ had\,(101, n) \\ had\,(110, n) \\ had\,(111, n) \end{bmatrix}$$

Строки had(OOO,«),had(OOl ,«),...4iad(l 11 *,ri)* матрицы можно рассматривать ,как функции дискретных переменных *пик,* определенные в целочисленных точках 0, 1, 2, ..., 7V-1 интервала [О, *N).* Эти функции называются *дискретными функциями Уолша.* Первая переменная *к* обозначает номер функции, вто-

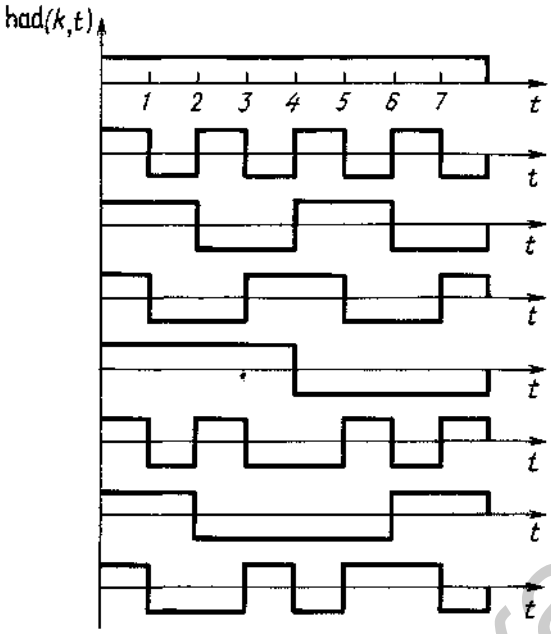


Рис. 3.8. функции Уолша в упорядочении Адамара.

рая переменная n — дискретное время. В качестве номера функции берется номер соответствующей строки матрицы Адамара. При этом говорят, что функции упорядочены по Адамару. Дискретные функции можно рассматривать как результат дискретизации непрерывных функций Уолша *had(k, i)*. Для $N = 8$ непрерывные функции Уолша показаны на рис. 3.8.

В практических приложениях используется и другая нумерация функций, что соответствует другому способу ранжирования обобщенных гармоник. Наиболее употребительными из них являются упорядочения по Пэли и по Уолшу.

Система Пэли образуется из системы Адамара двоичной инверсией номеров функций, т. е. путем записи разрядов двоичного представления номера $к$ в обратном порядке. Система Уолша располагает функцией в порядке возрастания числа смен знака на интервале (по "частотам" следования), что соответствует расположению их номеров в двоично-инверсном коде Грея.

Для $N = 8$ в рассмотренных способах упорядочения строки расположены следующим образом:

Упорядочение по Адамару	Упорядочение по Пэли	Упорядочение по Уолшу
000	000	000
001	100	100
010	010	ПО
011	110	010
100	001	011
101	101	111
ПО	011	101
111	111	001

Соответствующие матрицы при упорядочении по Пэли и по Уолшу равны:

$$\mathbf{P} = \{\mathrm{pal}(k, n)\} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

$$\mathbf{W} = \{\mathrm{wal}(k, n)\} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

/Важнейшими свойствами функций Уолша являются:

1) ортогональность

$$\sum_{n=0}^{N-1} \mathrm{had}(k_1, n)\mathrm{had}(k_2, n) = \begin{cases} N, & k_1 = k_2; \\ 0, & k_1 \neq k_2 \end{cases} \qquad (3.40)$$

или в матричной записи

$$\mathbf{H}_N \mathbf{H}_N^T = N\mathbf{I}, \qquad (3.41)$$

где I — единичная матрица;

2) симметричность

$$\mathbf{H}_N = \mathbf{H}_N^T. \qquad (3.42)$$

Используя свойство ортогональности (см. (3.40), (3.41)) и симметричности (см. (3.42)), легко получить IT' = Л^'Н;

3) мультипликативность по номеру функции

$$\mathrm{had}(k_1, n)\mathrm{had}(k_2, n) = \mathrm{had}(k_1 \oplus k_2, n) \qquad (3.43)$$

и номеру отсчета

$$\mathrm{had}(k, n_1)\mathrm{had}(k, n_2) = \mathrm{had}(k, n_1 \oplus n_2). \qquad (3\text{-}44)$$

Хотя эти свойства записаны для системы Адамара, они справедливы и для систем Пэли и Уолша.

Постоянная функция и функции меандрового типа с одной ненулевой позицией в двоичной записи номера называются функциями Радемахера. Для Л^= 8 они равны:

$$R_0 = \text{had}(000, n); \quad R_2 = \text{had}(010, n);$$
$$R_1 = \text{had}(100, n); \quad R_3 = \text{had}(001, n).$$

Из свойства мультипликативности (см. (3.43)) следует, что любая функция Уолша может быть получена перемножением функций Радемахера. Например:

$$\text{had}(011, n) = R_2 R_3;$$
$$\text{had}(110, n) = R_1 R_2.$$

Сравнивая функции Уолша с дискретными экспоненциальными функциями, можно заметить, что они совпадают при *N=2,*

3.4.2. Преобразование Уолша—Адамара

Пусть{х(и)} = (s(0), s(1), ..., *s(N-l^* - совокупность равноотстоящих отсчетов сигнала. Выражения

$$b(k) = \sum_{n=0}^{N-1} s(n)\,(-1)^{(k \cdot n)}, \quad k = 0, 1, ..., N-1; \tag{3.45}$$

$$s(n) = N^{-1} \sum_{k=0}^{N-1} b(k)(-1)^{(k \cdot n)}, \quad n = 0, 1, ..., N-1 \tag{3.46}$$

образуют пару дискретного преобразования Уолша-Адамара в показательной форме. Равенство (3.45) называется *прямым преобразованием* и дает спектр сигнала в базисе Уолша. Равенство (3.46) называют *обратным преобразованием.*

Используя матрицу Адамара порядка *N,* можно записать преобразование в матричной форме:

$$\mathbf{B} = \mathbf{HS}; \tag{3.47}$$

$$\mathbf{S} = N^{-1}\mathbf{HB}, \tag{3.48}$$

где $S = [_x(O), _s(1), \ldots, _s(^-1)]^r$, $B = [fe(O), _s b(1), \ldots, \&(7V-1)]^r$ - векторы-столбцы отсчетов сигнала и спектральных коэффициентов.

Основными свойствами преобразования являются:

1) линейность. Если {*(«)} и *{y{n)}* —две последовательности со спектрам и ^(£)} и (b{fc)} соответственно, то спектр их взвешенной суммы (z (и)} = *{Ax(n)* + *{By{n)* равен

$$[b_z(k)\} = (Ab_x(k)\} + \{Bb_y(k)j;$$

2) инвариантность к диадному сдвигу. Рассмотрим функцию *s(ri)* дискретной переменной *n* . Функция s (и © т) называется диадным сдвигом функции *s(n).* Сущность диадного сдвига заключается в перестановке отсчетов исходной функции. В частности, на место отсчета с номером *n* ставится отсчет с номером *n®m* . Например, пусть *(_s(n)}* = {00111100J и т = 4. Значения «®т ГЕЛЯ и = 0,1, , ..., 7 павны

n	$n \oplus \tau$
$000 \oplus 100$	$= 100$
$001 \oplus 100$	$= 101$
$010 \oplus 100$	$= 110$
$011 \oplus 100$	$= 111$
$100 \oplus 100$	$= 000$
$101 \oplus 100$	$= 001$
$110 \oplus 100$	$= 010$
$111 \oplus 100$	$= 011$

Из этой записи следует, что отсчет е номером нуль ставится на четвертое **место,** отсчет с номером один - на пятое место и т. д. В результате получим

$$\{s(n)\} = \{00111100\};$$
$$\{s(n \oplus \tau)\} = \{11000011\}.$$

Из свойства мультипликативности (3.44) следует, что при фиксированном т

$$\mathrm{had}(k, n \oplus \tau) = \mathrm{had}(k, n)\,\mathrm{had}(k, \tau) = \pm \mathrm{had}(k, n),$$

т. е. спектральные составляющие исходного и диадно-сдвинутого сигналов могут отличаться только знаком. Амплитудный же спектр при диадном сдвиге не меняется;

3) теорема о свертке и корреляции. Понятие диадного сдвига позволяет обобщить понятия свертки и корреляционной функции. Так как суммирование и вычитание по модулю два совпадают, то диадная свертка совпадает с диадной корреляцией и определяется выражением

$$y(n) = r(n) = \sum_{l=0}^{N-1} h(l)s(n \oplus l), \quad n = 0, 1, ..., (N-1). \tag{3.49}$$

Теорема о свертке утверждает, что спектр свертки равен **произведению** спектров сворачиваемых последовательностей:

$$b_y(k) = b_h(k)b_s(k). \tag{3.50}$$

Это позволяет для вычисления диадной свертки и **корреляционной** функции использовать преобразование Адамара:

$$\mathbf{R} = \mathbf{H}^{-1}(\mathbf{B}_h\,\mathbf{B}_s) = N^{-1}\mathbf{H}(\mathbf{B}_h\,\mathbf{B}_s). \tag{3.51}$$

3.4.3. Быстрое преобразование Уолша—Адамара

Вычисление преобразования по формулам (3.45)—(3.48) требует выполнения *N(N—1)* операций сложения. Существуют быстрые алгоритмы, которые требуют только $N\log_2 N$ операций.

Для построения быстрого алгоритма рассмотрим матричное равенство

$$\mathbf{B} = \mathbf{H}_N \mathbf{S}.$$

Запишем его **в** виде

$$\begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(\frac{N}{2}-1) \\ b(N/2) \\ \vdots \\ b(N-1) \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{N/2} & \mathbf{H}_{N/2} \\ \mathbf{H}_{N/2} & -\mathbf{H}_{N/2} \end{bmatrix} \begin{bmatrix} s(0) \\ s(1) \\ \vdots \\ s(\frac{N}{2}-1) \\ s(N/2) \\ \vdots \\ s(N-1) \end{bmatrix}$$

что позволяет отдельно вычислить первую и вторую половины спектра. Получим

$$\begin{bmatrix} b(0) \\ b(1) \\ \vdots \\ b(\frac{N}{2}-1) \end{bmatrix} = \mathbf{H}_{N/2} \begin{bmatrix} s(0)+s(N/2) \\ s(1)+s(N/2+1) \\ \dots\dots\dots \\ s(\frac{N}{2}-1)+s(N-1) \end{bmatrix}$$

$$\begin{bmatrix} b(N/2) \\ b(\frac{N}{2}-1) \\ \vdots \\ b(N-1) \end{bmatrix} = \mathbf{H}_{N/2} \begin{bmatrix} s(0)-s(N/2) \\ s(1)-s(N/2+1) \\ \dots\dots\dots \\ s(\frac{N}{2}-1)-s(N-1) \end{bmatrix}$$

Из этих выражений следует, что вычисление Л'-точечного преобразования сводится к предварительному суммированию (вычитанию) входных данных и последующему вычислению двух Ai/2-точечных преобразований (рис. 3.9, *а).*

Так как $N = 2^{l}$, то процедуру снижения.размерности преобразования можно продолжить до получения двухточечного преобразования. Для этого потребуется $\log_2 N$ шагов. На каждом шаге производится *N* сложений, поэтому общее количество операций сложения равно $N\log_2 N$. Рассмотренный алгоритм называется *быстрым преобразованием Адамара* (БПА).

На рис. 3.9, *б* приведен граф вычислительного процесса для7У= 8.

Аналогично преобразованию Фурье БПА можно трактовать как разложение (факторизацию) матрицы Адамара в произведение слабо заполненных сомножителей. Непосредственно по графу для $N = 8$ можно записать

$$\mathbf{H}_8 = \mathbf{C}_1\mathbf{C}_2\mathbf{C}_3 =$$

$$= \begin{bmatrix} 1 & 1 & & & & & & \\ 1 & -1 & & & & & & \\ & & 1 & 1 & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & 1 & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & 1 \\ & & & & & & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & & 1 & & & & & \\ & 1 & & 1 & & & & \\ 1 & & -1 & & & & & \\ & 1 & & -1 & & & & \\ & & & & 1 & & 1 & \\ & & & & & 1 & & 1 \\ & & & & 1 & & -1 & \\ & & & & & 1 & & -1 \end{bmatrix} \begin{bmatrix} 1 & & & & 1 & & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & 1 \\ 1 & & & & -1 & & & \\ & 1 & & & & -1 & & \\ & & 1 & & & & -1 & \\ & & & 1 & & & & -1 \end{bmatrix}.$$

Существуют и другие формы факторизации, в частности

$$\mathbf{H}_8 = \mathbf{C}_3\mathbf{C}_2\mathbf{C}_1; \quad \mathbf{H}_8 = \mathbf{C}^3,$$

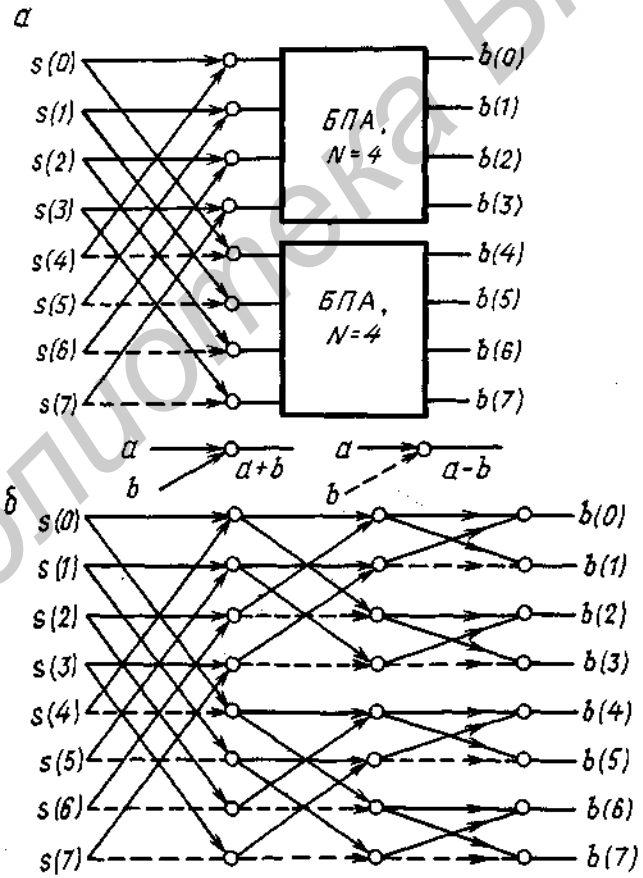


Рис. 3.9. Быстрое преобразование Уолша—Адамара.

где

$$\mathbf{C}=\begin{bmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ 1 & -1 & & & & & & \\ & & 1 & -1 & & & & \\ & & & & 1 & -1 & & \\ & & & & & & 1 & -1 \end{bmatrix}.$$

3.4.4. Двухмерное преобразование

Рассмотрим сигнал, заданный матрицей размером N_1 x N_2

$$\mathbf{S} = \begin{bmatrix} s(0,0) & s(0,1) & \dots & s(0,N_2-1) \\ s(1,0) & s(1,1) & \dots & s(1,N_2-1) \\ \dots & \dots & \dots & \dots \\ s(N_1-1,0) & s(N_1-1,1) & \dots & s(N_1-1,N_2-1) \end{bmatrix}. \qquad (3.52)$$

Пара двухмерного преобразования Уолша—Адамара этого сигнала определяется равенствами

$$b(k_1, k_2) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} s(n_1, n_2)(-1)^{(\mathbf{n_1}\cdot\mathbf{k_1})+(\mathbf{n_2}\cdot\mathbf{k_2})}; \qquad (3.53)$$

$$s(n_1, n_2) = N_1^{-1}N_2^{-1} \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} b(k_1, k_2)(-1)^{(\mathbf{n_1}\cdot\mathbf{k_1})+(\mathbf{n_2}\cdot\mathbf{k_2})}. \qquad (3.54)$$

Выражение (3.53) называется прямым преобразованием, а выражение (3.54) -обратным.

Рассмотрим в равенстве (3.53) внутреннюю сумму

$$\sum_{n_1=0}^{N_1-1} s(n_1, n_2)(-1)^{(\mathbf{n_1}\cdot\mathbf{k_1})}.$$

Легко заметить, что эта сумма является преобразованием Уолша—Адамара столбцов матрицы S (3.52). Изменяя κ_1 , получим N_1 коэффициентов преобразования, которые можно записать в виде вектора-столбца. Аналогичные векторы-столбцы получаются при изменении переменной n_2 . Поэтому все коэффициенты двухмерного преобразования можно представить в виде матрицы размером N_1 x N_2 :

$$\mathbf{B} = \begin{bmatrix} b(0,0) & b(0,1) & \dots & b(0,N_2-1) \\ b(1,0) & b(1,1) & \dots & b(1,N_2-1) \\ \dots & \dots & \dots & \dots \\ b(N_1-1,0) & b(N_1-1,1) & \dots & b(N_1-1,N_2-1) \end{bmatrix},$$

что позволяет записать пару двухмерного преобразования в матричной форме:

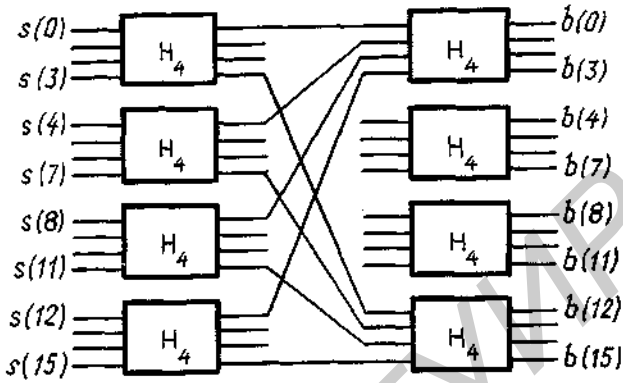


Рис. 3.10. Вычисление двухмерного преобразования при помощи одномерных.

$$\mathbf{B} = \mathbf{H}_{N_1} \mathbf{S} \mathbf{H}_{N_2}; \tag{3.55}$$

$$\mathbf{S} = N_1^{-1} N_2^{-1} \mathbf{H}_{N_1} \mathbf{B} \mathbf{H}_{N_2}. \tag{3.56}$$

Здесь (3.55) — прямое преобразование; (3.56) —обратное.

Приведенные формулы показывают, что двухмерное преобразование выполняется при помощи одномерного следующим образом:

1) выполняются одномерные преобразования размером N_1 для каждого из столбцов $\{s(«j«_2)\}^T$, $n_2 = 0,1, \ldots, N_2 - 1$, матрицы S ;

2) выполняется одномерное преобразование размером N_2 для каждой из строк матрицы $H^{\wedge}_1 \{^s(«!>^n)\}$ •

Двухмерное преобразование может быть использовано дня понижения размера процессора, если длина вектора входных данных велика. Для этого входные данные разбиваются на отрезки длины N_1. Эти отрезки записываются в виде столбцов матрицы S и далее выполняется двухмерное преобразование. Столбцы матрицы B выстраиваются в одномерный массив, который дает коэффициенты одномерного преобразования.

Пример 3.3. Вычислить преобразование длины $N = 16$ при помощи двухмерного преобразования длины $N = 4$. Так как $N = N_1 N_2 = 4.4$, то вектор S = [s(0), s(1) , ... , i (IS)] преобразуется в матрицу

$$\mathbf{S} = \begin{bmatrix} s(0) & s(4) & s(8) & s(12) \\ s(1) & s(5) & s(9) & s(13) \\ s(2) & s(6) & s(10) & s(14) \\ s(3) & s(7) & s(11) & s(15) \end{bmatrix}.$$

Для вычислений можно использовать схему, показанную на рис. 3.10.

3.4.5. Взаимосвязь спектров

Рассмотрим матричные формы **преобразования Фурье**

$$\mathbf{F} = \mathbf{V}\mathbf{S} \tag{3.57}$$

и преобразования Уолша—Адамара

$$\mathbf{B} = \mathbf{H}_N \mathbf{S} . \qquad (3.58)$$

Определим из формулы (3.57) S и подставим его в (3.58). Тогда получим

$$\mathbf{S} = \mathbf{V}^{-1}\mathbf{F}; \quad \mathbf{B} = \mathbf{H}_N \mathbf{V}^{-1}\mathbf{F}. \qquad (3.59)$$

Аналогичным образом, определив S из (3.58) и подставив его в (3.57), получим

$$\mathbf{S} = \mathbf{H}^{-1}\mathbf{B}; \quad \mathbf{F} = \mathbf{V}\mathbf{H}_N^{-1}\mathbf{B}. \qquad (3.60)$$

Выражения H^V' и VH"j^ в формулах (3.59), (3.60) позволяют пересчитать спектр по Фурье в спектр по Уолшу—Адамару и наоборот и называются *ядрами Фурье.* Учитывая равенства

$$\mathbf{V}^{-1} = N^{-1}\mathbf{V}^* ; \quad \mathbf{H}^{-1} = N^{-1}\mathbf{H},$$

можно также записать

$$\mathbf{B} = N^{-1}\mathbf{H}_N \mathbf{V}^* \mathbf{S}; \quad \mathbf{F} = \mathbf{N}^{-1}\mathbf{V}\mathbf{H}\mathbf{B} . \qquad (3.61)$$

Для умножения на ядра Фурье в формулах (3.61) существуют быстрые алгоритмы:

3.5. ТЕОРЕТИКО-ЧИСЛОВОЕ ПРЕОБРАЗОВАНИЕ

3.5.1. Кольцо и поле

Для определения *теоретико-числового преобразования* (ТЧП) нам потребуются два математических понятия — кольцо и поле.

Кольцом R называется множество элементов *(a, Ъ, с, ...)*, над которыми определены две операции — сложение и умножение (может быть, не совсем обычные) и выполняются следующие аксиомы:

1) если *a , Ъ eR* , то *a+beR* и я- ftei? . Эта аксиома называется аксиомой замкнутости;

2) в кольце существует нейтральный элемент 0, такой, что для любого элемента *а* кольца *а* + 0 = *а,* и обратный элемент *(-а)*, такой, что *а* + (-Й) = 0;

3) для элементов кольца выполняются свойства ассоциативности

$$a(be) = (ab)c$$

и дистрибутивно сти

$$a(Ь + с) = ab + ac .$$

Если в *R* существует нейтральный элемент по умножению — 1 , такой, что для любого элемента *а* кольца справедливо равенство *а* • 1 = 1 • а = а , то такое кольцо называется .уиигаркьш или *кольцом с единицей.*

Если в унитарном кольце для каждого элемента *а* существует мультипликативно обратный элемент a^{-1} , т. е. справедливо равенство $ad^{-1} = 1$, то такое кольцо называется *полем.*

В задачах цифровой обработки сигналов (в частности, в теоретико-числовых преобразованиях) важное значение имеет кольцо целых чисел по Модулю

числа M. Элементами кольца являются вычеты 0, 1,2. . . . $M-1$, а операциями — операции сложения и умножения по модулю M.

Например, если $M = 14$, то элементами кольца будут числа 0, 1, 2,..., 13. Эти числа не образуют поле, так как не для всякого элемента кольца существует обратный элемент. В частности, число 2 не имеет обратного числа. Числа же 3 и 5 являются взаимно обратными, так как $3 \cdot 5 = 15 \equiv 1 \bmod 14$.

Можно доказать, что число a имеет обратное число a^{-1}, если оно взаимно просто с модулем. Поэтому если модуль равен простому числу, то все числа кольца имеют обратные и такое кольцо будет полем. Наличие обратного элемента важно для последующего построения теоретико-числового преобразования.

Рассмотрим некоторый элемент кольца a и его степени a^2, a^3, ... Поскольку число элементов в кольце конечно и справедлива аксиома замкнутости, то на каком-то таге получим $a^{k+1} = a$, что дает $a^k = 1$. Наименьшее k, удовлетворяющее этому равенству, называется *порядком элемента* a. Для предыдущего примера легко проверить, что $3^4 = 1 \bmod 14$, т. е. число 3 имеет порядок 6.

Если a и M взаимно просты, то

$$a^{\varphi(M)} \equiv 1 \bmod M, \qquad (3.62)$$

где $\varphi(M)$ — функция Эйлера, равная числу чисел ряда 0,1...,(M—1), взаимно простых с M. Следовательно, порядок числа a должен быть делителем функции Эйлера и все порядки лежат среди всевозможных делителей. Равенство (3.62) называется *теоремой Эйлера*. С ее помощью можно доказать следующую теорему.

Т е о р е м а 3.2. Если $M = p_1^{i_1} p_2^{i_2} \dots p_l^{i_l}$, то порядок k числа a является делителем наибольшего общего делителя (НОД) чисел $P_1-1, P_2-1, \dots, P_l-1$, т. е.

$$k \mid \text{НОД}(p_1 - 1, p_2 - 1, \dots, p_l - 1).$$

Читается: k делит НОД.

Теорему Эйлера можно использовать для нахождения **обратного элемента.** Умножив обе части сравнения (3.62) на a^{-1}, получим

$$a^{-1} \equiv a^{\varphi(M)-1} \bmod M,$$

что дает

$$a^{-1} = a^{\varphi(M)-1}$$

3.5.2. Теоретико-числовое преобразование и вычисление сверток

Рассмотрим кольцо целых чисел по модулю числа $M = p_1^{i_1} \cdot p_2^{i_2} \cdot \dots \cdot p_j^{i_j}$, где p_i — простые числа. Пусть a — элемент кольца порядка N (т. е. $0 \equiv a^N \bmod M$).

Из теоремы 3.2 следует, что $N \mid \text{НОД}(p_1 - 1, p_2 - 1, \dots, p_s - 1)$.

Пусть $\{s(ri)\} = (s(0), x(1), \dots, s(N-1))$ —последовательность отсчетов

входного сигнала. Пара теоретико-числового преобразования последовательности (j (и) } определяется в экспоненциальной форме следующими равенствами:

$$a(\kappa) = \overset{N-i}{T.}\ s(n)c\overset{t}{I}^{n},\ \kappa = 0.1.\ldots.N\text{-}Л\ ; \tag{3.63}$$

$$s(n) = N'\overset{N-i}{\underset{k=0}{2}}\ a\{k)a^{\sim kn}\ ,\ n = 0,\ 1,\ \ldots,N\text{-}\backslash\ , \tag{3.64}$$

где N^{l} — число, обратное числу ./V,Т. е. $NN' \equiv 1 \bmod M$.

Соответствующие матричные выражения имеют вид

$$A = TS\ ; \tag{3.65}$$

$$S = Л/V^{l}A\ , \tag{3.66}$$

где $S = [s(0), s(1), \ldots, s(N-1)]^{T}$, $A = [a(0), a(1), \ldots, a(N-1)]^{T}$, а

$$T = \kappa \begin{bmatrix} \overset{n}{\cdot} \\ \cdot \\ \cdot \end{bmatrix} \qquad T^{-1} = k \begin{bmatrix} \overset{n}{\cdot} \\ \cdot \\ a^{-kn} \\ \cdot \end{bmatrix}$$

есть матрицы прямого и обратного преобразований соответственно.

Сравнивая формулы (3.63) —(3.66) с соответствующими формулами для ДПФ, легко заметить полную аналогию в записи этих двух преобразований. Отличие заключается в том, что поворачивающие множители W заменены элементом о порядка N. Заметим также, что ТЧП длины N существует, если только в кольце имеется элемент, обратный элементу N. Поскольку в поле наличие обратных элементов гарантируется простотой модуля, то в поле ТЧП существует всегда.

Так как ТЧП по своей структуре подобно ДПФ, то для него справедливы все свойства ДПФ с заменой W на a и выполнением арифметических операций по модулю M. В частности, строки матриц Т и $T^{"l}$ ортогональны по модулю M и справедлива теорема о свертке.

Основным применением ТЧП является вычисление сверток и корреляционных функций методом двойного преобразования. Поскольку a — целое число, а не комплексное, то операция умножения существенно упрощается и выполняется точно. Шумы округления полностью отсутствуют, что может быть важно, когда входное отношение сигнал/шум мало.

Чтобы ТЧП давало еще дополнительные преимущества, необходимо позаботиться о хорошем выборе параметров a и M. Ъ частности, умножение н? а и деление на M должны быть простыми операциями. Лучшим выбором для a является $a = 2^{l'}$. В этом случае умножение на a сводится к сдвигу числа на i позиций. Деление проще всего осуществляется, если двоичное представление модуля содержит только два ненулевых символа. В этом плане заслуживают внимания числа вида $2^{2^{t}} + 1$ — числа Ферма. Для них $Л^{r} < 2^{r+'}$ и а является степенью двойки, поэтому матрица преобразования разбивается на максимальное количество сомножителей, структура которых аналогична соответст-

вующим матрицам ДПФ. Однако существенным недостатком чисел Ферма является большая длина машинного слова. Например, если $M = 2^{64} + 1$, $TON = 128$, т. е. приходится работать с 65-разрядными двоичными числами.

При выборе меньшего модуля не всегда удается выполнить условие $a = 2^{r}$, кроме того, деление на M происходит сложнее.

Как уже указывалось, при выполнении ТЧП вычисления производятся по модулю M, поэтому величина M должна быть больше ожидаемого ответа. Иначе будет получен искаженный результат.

Приведенные рассуждения показывают, что параметры N, a и M жестко связаны между собой теоретико-числовой конструкцией кольца или поля и не могут варьироваться произвольно. Это создает определенные трудности при практическом использовании ТЧП.

Для иллюстрации свойств и возможностей ТЧП рассмотрим пример.

Пример 3.4. Вычислить свертку последовательностей $S = [2,-2,1,0]^{T}$, $H = [1,2,0,0]^{T}$.

Поскольку $N = 4$, то можно взять $M = 2^{2} + 1 = 17$. Целое число 2 имеет порядок 8, в чем можно убедиться непосредственно проверкой. Поэтому целое число 4 имеет порядок 4 и можно положить $a = 4$. Далее получим $a^{2} = 16$, $a^{3} = 13$, $a^{4} = 1$.

Матрица прямого преобразования равна

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 4^{2} & 4^{3} \\ 1 & 4^{2} & 4^{4} & 4^{6} \\ 1 & 4^{3} & 4^{6} & 4^{9} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 16 & 13 \\ 1 & 16 & 1 & 16 \\ 1 & 13 & 16 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & -1 & -4 \\ 1 & -1 & 1 & -1 \\ 1 & -4 & -1 & 4 \end{bmatrix}.$$

Как и в преобразовании Фурье, строки матрицы обратного преобразования T^{-1} расположены в обратном порядке, т. е.

$$T^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -4 & -1 & 4 \\ 1 & -1 & 1 & -1 \\ 1 & 4 & -1 & -4 \end{bmatrix}$$

Число, обратное N, равно $N' = 13$, так как $13 * 4 = 52 \equiv 1 \bmod 17$.

Прямые преобразования сворачиваемых последовательностей равны

$$A_{s} = TS = [1, 10, 5, 9]^{T};$$

$$A_{h} = TH = [3, 9, 16, 10]^{T},$$

а их произведение

$$A_{S} \cdot A_{H} = [3, 90, 80, 90]^{T} = [3, 5, 12, 5]^{T}.$$

Обратное преобразование этой последовательности дает свертку

$$Y = 13 T^{-1}(A_{S} \cdot A_{H}) = (2, 2, 14, 2)^{T} = (2, 2, -3, 2)^{T}.$$

3.6. ФУНКЦИИ ХААРА И ПРЕОБРАЗОВАНИЕ ХААРА

Функции Хаара {har(r, *к,* г)} были построены в 1910 г. Они образуют полную систему ортонормированных функций, определенных на интервале [0, *N),* $N = 2,1 = 1,2,...$, и описываются следующими рекуррентными соотношениями:

$$\text{har}(0,0,\ O\ = 1\ ,\quad \text{re}\ [0,N);\text{^'-}$$

$$hai(r,\ k,f) = \begin{cases} 2^{r/2}, & N(k-l)2^{-r} < t < N(k-O,5)2^{-r}; \\ -2^{r/2}, & N(k-0,5)\ 2^{-r} < t < Nk2^{r}; \\ 0, & \text{при остальных } te[0,N), \end{cases} \qquad (3.67)$$

где $0 < r < \log_2 N$ и $1 < к < 2^r$.

На рис. 3.11 показана система функций для Л ^ 8. При дискретизации системы (3.67) получаются дискретные функции Хаара{har(r, *к, n)* } , определенные в целочисленных точках 0, 1, 2, ...,*N*- 1. ДляЛ^ = 8 матрица этих функций имеет вид

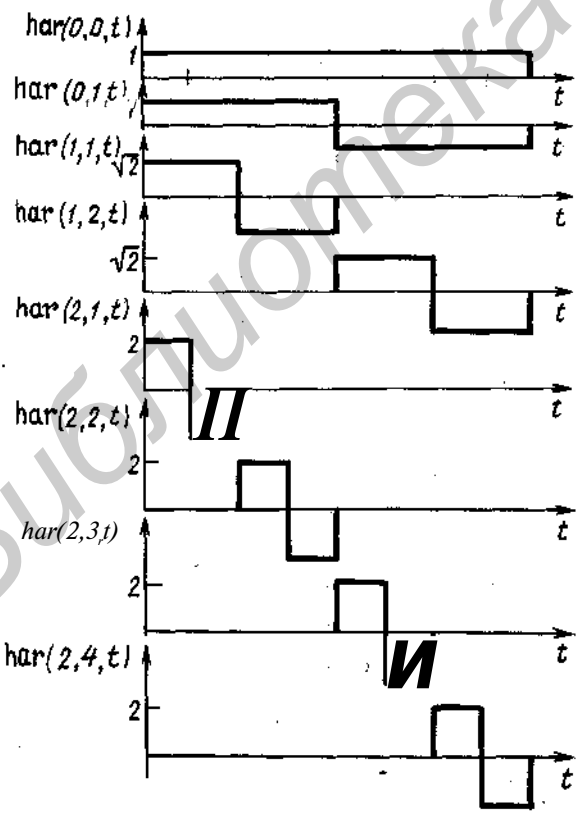


Рис. 3.11. Функции Хаара.

$$X = \{har(r, *, \cdot)\} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & 0 & 0 & 0 & 0 & & \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & & \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}.$$

Так как функции Хаара ортогональны, то $XX^T = N I$. Умножив обе части этого равенства на x^{-1}, получим $x^{-1} = W^T x^T$. Множество функций Хаара, в отличие от функций Уолша, не является мультипликативным, т. е., вообще говоря, произведение функций Хаара не приводит вновь к функции Хаара. С увеличением переменной r уменьшается интервал, на котором функция Хаара отлична от нуля. Это обстоятельство позволяет использовать разложение Хаара для получения спектральных коэффициентов, учитывающих локальное поведение функции.

Пара дискретного преобразования Хаара вектора $S = [s(0), s(1), ..., s(N-1)]^T$ определяется следующими соотношениями:

$$Y = {}_xS; \qquad (3\text{-}68)$$

$$a = {}_x^{-1}Y = {}_J v^{-1} x^r Y. \qquad (3.69)$$

Выражение (3.68) называется *прямым преобразованием,* выражение (3.69) - *обратным*.

Для вычисления этих преобразований существуют быстрые алгоритмы с числом операций сложения, равным $2(N-1)$, т. е. преобразование Хаара является самым быстрым среди всех рассмотренных ортогональных преобразований.

Граф прямого преобразования для $N = 8$ показан на рис. 3.12.

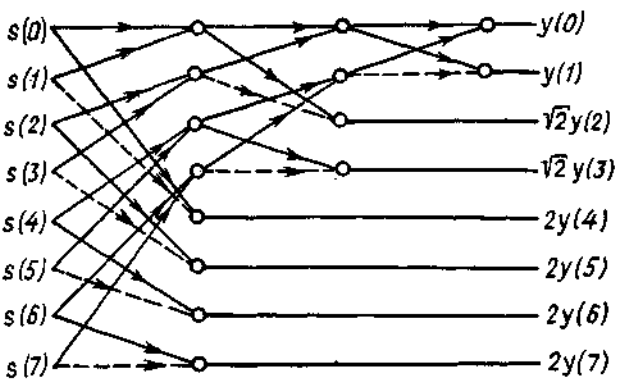


Рис. 3.12. Быстрое преобразование Хаара.

3.7. АДДИТИВНАЯ СЛОЖНОСТЬ ДИСКРЕТНЫХ ОРТОГОНАЛЬНЫХ ПРЕОБРАЗОВАНИЙ

Ранее (см. § 3.2, 3.4, 3.6) были рассмотрены различные ортогональные преобразования вектора отсчетов S . Все они с формальной точки зрения сводятся к вычислению векторно-матричного произведения AS, где A — матрица преобразования. Эти вычисления могут быть выполнены при помощи быстрых алгоритмов, позволяющих существенно сократить количество операций. Однако остается открытым вопрос, обладают ли рассмотренные алгоритмы минимальным количеством операций, и если нет, то каковы возможности их дальнейшего улучшения. Другими словами, необходимо указать нижнюю границу сложности вычисления ортогональных преобразований. Важность этой границы состоит в том, что она устанавливает предельные соотношения, к которым следует стремиться при проектировании, либо указывает на неулучшаемость алгоритма.

В настоящем параграфе ответ на поставленные вопросы дается применительно к операциям сложения, т. е. приводится нижняя граница аддитивной сложности дискретных ортогональных преобразований (вопрос о нижней границе мультипликативной сложности рассмотрен в следующей главе).

В [1] показано, что аддитивная сложность умножения вектора на матрицу A оценивается неравенством

$$C_a > (\log\hat{}\det\hat{}O/bg\hat{}c \ ,$$

где c - наибольший из модулей элементов матрицы A .

Дня матриц ортогональных преобразований Фурье, Уолша и Хаара справедливо соотношение

$$A(A^*)^T = ЛП \ ,$$

где N - порядок матрицы, а знак * обозначает комплексное сопряжение. Используя это тождество и свойства определителей, получаем

$$\det A\text{-}\det A^* = Л\hat{} \ .$$

Матрицы преобразований Уолша и Хаара действительные, а сопряженная матрица преобразования Фурье получается из исходной изменением порядка следования строк, что не меняет определителя. Поэтому

$$, detA = N^{N/2} \ .$$

Для матриц Уолша и Фурье 2c = 2, а для матриц Хаара 2c = $(2N)\hat{}^2$. Поэтому

$$C_a \geqslant \begin{cases} OJSNlo\hat{}N & \text{(для матриц Уолша и Фурье);} \\ (Nlog_2N)\ I\ (\text{I Hog}_2\text{iV}) & \text{(для матриц Хаара).} \end{cases}$$

Известны и более точные оценки. В частности, для матриц Фурье и Уолша $C_a > 0{,}63\text{Mog}_2./\text{V}$.

Рассмотренные ранее быстрые алгоритмы выполняются за Mog_2N опера-

ций для преобразований Уолша и Фурье и за $2(N-1)$ операций для матриц Хаара, т. е. вдвое сложнее.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАЧИ

3.1. Что называется обобщенным спектральным представлением сигнала?

3.2. Какие системы функций называются базисными?

3.3. Вычислите значения поворачивающих множителей W, W^2, W^3, W^1, W^2, W^3 для $N = 8$.

3.4. Докажите свойства 1-4 системы ДЭФ.

3.5. Докажите свойства 1-8 ДПФ.

3.6. Запишите матрицы прямого и обратного ДПФ для $N = 4,8$.

3.7. Вычислите спектр последовательностей (1,1, —1, -1), (1, —1, —1, 1). Убедитесь в справедливости свойств 4, 7.

3.8. Постройте графы БПФ с прореживанием по времени и частоте для $N = 16$.

3.9. Постройте графы БПФ для $N = 6$ и по ним запишите матрицы ДПФ в виде произведения двух сомножителей.

ЗЛО. Как использовать матрицы прямого ДПФ для вычисления обратного ДПФ? Вычислите обратное ДПФ для спектров,найденных в п. 3.7.

3.11. Вычислите спектр действительной последовательности (-1, —1, 1, 1, 1, 1, -1, -1) при помощи ДПФ размера $N = 4$. Сравните полученные результаты с результатами примера 3.2.

3.12. Вычислите корреляционную функцию последовательности (1,1, 1, -1) прямым методом и с помощью ДПФ. Сравните результаты.

3.13. Докажите свойства 1-3 функций Уолша.

3.14. Докажите свойства 1-3 преобразования Адамара.

3.15. Вычислите спектр по Адамару последовательности (1, 2, 1, -1, 3, 2, 1, 2) при помощи одномерного и двухмерного преобразований. Сравните результаты.

3.16.Постройте диадные сдвиги последовательности (0, 1, 2, 3, 4, 5, 6, 7) для $\Gamma = 3,5$. Вычислите спектры сдвинутых последовательностей и убедитесь в инвариантности спектров к диадному сдвигу.

3.17.Используя ядро Фурье и результаты п. 3.7, найдите спектры последовательностей (1, 1, —1, -1) и (1, —1, —1, 1) по Адамару. Убедитесь в неинвариантности спектров к циклическому сдвигу.

3.18. Чем отличается кольцо от поля? Какое понятие является более общим?

3.19. Для кольца вычетов по модулю числа 11 найдите элементы, обратные элементам 5 и 7. Является ли это кольцо полем?

3.20. Назовите основные свойства функций Хаара и преобразования Хаара. В чем существенное отличие этого преобразования от преобразований Фурье и Уолша-Адамара?

3.21. В чем заключаются преимущества и недостатки ТЧП?

3.22. Сопоставьте вычислительную сложность преобразований Фурье, Уолша—Адамара и Хаара.

4. ДИСКРЕТНАЯ СВЕРТКА И ЕЕ ВЫЧИСЛЕНИЕ

4.1. ЛИНЕЙНАЯ, ЦИКЛИЧЕСКАЯ И ДИАДНАЯ СВЕРТКИ

Многие задачи цифровой обработки сигналов с формальной точки зрения представляют собой задачи вычисления свертки двух последовательностей $\{h(n)\}$ и $\{s(n)\}$. В частности, через свертку выражается отклик линейной системы на некоторое возмущающее воздействие; при обращении одной из последовательностей по переменной n свертка превращается в корреляционную функцию, часто используемую в задачах обнаружения сигналов и определения их параметров. Список подобных примеров можно продолжить.

Рассмотрим различные виды сверток, встречающихся в технических приложениях.

Пусть даны две апериодические последовательности $\{s(и)\}$ длины L и $\{h(и)\}$ длины M. В силу апериодичности $s(n) = 0$ при $-1 < и < 0$; $h(и) = 0$ при $M - 1 < и < 0$. *Апериодическая пня линейная свертка* этих последовательностей имеет длину $L + M - 1$ и определяется как

$$y(n) = \sum_{l=0}^{L+M-2} h(l)s(n-t), \quad и = 0, 1, \ldots L+M-2. \tag{4.1}$$

Это выражение можно записать и в матричной форме. Например, для $L = M = N$ получим

$$\begin{bmatrix} 7(0) \\ 7(D \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y(27V-2) \end{bmatrix} = \begin{bmatrix} \sim^*(0) & & & \\ h(1) & A(0) & & \\ A(2) & AO) & A(0) & \\ \cdots & \cdots & \cdots & \cdots \\ h(N-1)1 & A0V-2) & hiN-3) \ldots & A(0) \\ & A(7V-1) & A(Л/-2) \ldots & A(1) \\ \cdots & \cdots & \cdots & \cdots \\ & & Л(ЛМ)й<M-2) & \\ & & & h(N-1) \end{bmatrix} \begin{bmatrix} s(0) \\ s(1) \\ s(2) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ s(N--1) \end{bmatrix} \tag{4.2}$$

Представим исходные последовательности в виде полиномов от некоторой переменной z :

$$й(г) = \sum_{/=0}^{M-1} h(l)z^{l}; \qquad s(z) = \sum_{m=o}^{1-1} x(m)z^{w}.$$

Произведение этих полиномов дает полиному (z) степени $L + M - 2$:

$$y(z) = h(z)s(z) = \sum_{n=o}^{L+M-2} y(n)z^{n}.$$

Коэффициенты $y(n)$ полинома $y(z)$ образуются суммированием всех произведений $h(l)s(m)$, для которых $l+m=n$. Следовательно, $m=n-l$ и $y(n)=\sum_{l=0}^{L+M-2} h(l)s(n-l)$, $n=0,1,\ldots,L+M-2$, что дает значения свертки. Таким образом, свертка двух последовательностей может рассматриваться как произведение двух полиномов.

Пример 4,1. Для трехточечных последовательностей $\{s(n)\}=\{s(0), s(1), s(2)\}$ и $\{h(n)\}=\{h(0), h(1), h(2)\}$ получим:

$$s(z) = s(0)+s(1)z+s(2)z^2;$$

$$h(z) = h(0)+h(1)z+h(2)z^2;$$

$$y(z) = s(z)h(z) = \sum_{n=0}^{4} y(n)z^n,$$

где коэффициенты полинома $y(z)$ равны

$$y(0) = h(0)s(0);$$

$$y(1) = h(1)s(0)+h(0)s(1);$$

$$y(2) = h(2)s(0)+h(1)s(1)+h(0)s(2);$$

$$y(3) = h(2)s(1)+h(1)s(2);$$

$$y(4) = h(2)s(2).$$

Эти коэффициенты можно получить и из матричной записи (4.2).

Циклическая свертка определяется для периодических последовательностей длины N выражением

$$y(n)=\sum_{l=0}^{N-1} h(l)s(n-l), \qquad n=0,1,\ldots,N-1. \tag{4.3}$$

В силу периодичности последовательностей номера отсчетов берутся по модулю N, поэтому $s(-n)=s(N-n)$, $h(-n)=h(N-n)$.

Матричная запись циклической свертки имеет вид:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & \ldots & h(N-1) \\ h(1) & h(2) & \ldots & h(0) \\ h(2) & h(3) & \ldots & h(1) \\ \cdots & \cdots & \cdots & \cdots \\ h(N-1) & h(0) & \ldots & h(N-2) \end{bmatrix} \begin{bmatrix} s(0) \\ s(N-1) \\ s(N-2) \\ \vdots \\ s(1) \end{bmatrix}. \tag{4.4}$$

При вычислениях по модулю N из свойств сравнений следует, что $N\equiv 0$, поэтому в полиномиальном представлении $z^N=1$ и циклическая свертка может рассматриваться как произведение двух полиномов по модулю полинома z^N-1:

$$y(z) = h(z)s(z) \bmod (z^N-1).$$

Обозначим $h(z)s(z)=w(z)=w(0)+w(1)z+\ldots+w(2N-2)z^{2N-2}$. Произведение по модулю z^N-1 просто означает, что

$$
\begin{aligned}
&y(0) = w(0) + w(N);\\
&y(1) = w(1) + w(N+1);\\
&\cdots\cdots\cdots\cdots\\
&y(N-2) = w(N-2) + w(2N-2);\\
&y(N-1) = w(N-1).
\end{aligned}
\qquad (4.5)
$$

Пример 4.2. Для трехточечной циклической свертки с учетом равенства $z^3 = 1$ получим

$$
\begin{aligned}
Y(z) &= (y(0) + y(1)z + y(2)z^2 + y(3)z^3 + y(4)z^4) \bmod (z^3 - 1) =\\
&= (y(0) + y(3)) + y(1) + y(4))z + y(2)z^2 .
\end{aligned}
$$

Используя результаты примера 4.1 и соотношения (4.5), запишем коэффициенты циклической свертки:

$$
\begin{aligned}
y(0) &= x(0)s(0) + x(2)s(1) + x(1)s(2);\\
y(1) &= x(1)s(0) + x(0)s(1) + x(2)s(2);\\
y(2) &= x(2)s(0) + x(1)s(1) + x(0)s(2).
\end{aligned}
$$

Эти равенства можно получить и с помощью матричного соотношения (4.4).

Диодная свертка определяется выражением

$$
y(n) = \sum_{l=0}^{N-1} h(l)s(n \oplus l), \quad n = 0,1,\ldots,N-1. \qquad (4.6)
$$

Здесь номера отсчетов представляются в двоичной системе счисления, арифметические операции над числами выполняются посимвольно по модулю два, а длина сворачиваемых последовательностей равна $N = 2^v$, v — целое число. При этом сдвиг функции фактически заключается в перестановке отсчетов. Таким образом, коэффициентами свертки являются суммы произведений $h(l)\,s(n \oplus l)$, полученные при различных перестановках функций $s(l)$.

Пример 4.3. Пусть $\{h(n)\} = \{1, -1, 1, -1\}$, $\{s(n)\} = \{1, 2, 3, 4\}$. Значения диадного сдвига равны 00, 01, 10, 11. При этих сдвигах аргумент $i \oplus l$ функции $s(n \oplus l)$ получает следующие значения:

00 ⊕ l	01 ⊕ l	10 ⊕ l	11 ⊕ l
00	01	10	11
01	00	11	10
10	11	00	01
11	10	01	00

Диадный сдвиг приводит к перестановке отсчетов. Например, при $i = 01$ меняются местами четные и нечетные отсчеты, при $i = 11$ отсчеты располагаются в обратном порядке. Это дает следующие сдвинутые последовательности:

$$
\begin{aligned}
&\{s(00 \oplus l)\} = \{1,2,3,4\};\\
&\{s(01 \oplus l)\} = \{2,1,4,3\};\\
&\{s(10 \oplus l)\} = \{3,4,1,2\};\\
&\{s(11 \oplus l)\} = \{4,3,2,1\}.
\end{aligned}
$$

Значения свертки равны:

$$y(00) = [1,-1,1,-1]\ [1,2,3,4]^T = -2;$$
$$y(01) = \{1,-1,1,-1]\ [2,1,4,3]^T = 2;$$
$$y(10) = [1,-1,1,-1]\ [3,4,1,2]^T = -2;$$
$$y(11) = [1,-1,1,-1]\ [4,3,2,1]^T = 2.$$

4.2. ПРЯМЫЕ МЕТОДЫ ВЫЧИСЛЕНИЯ СВЕРТОК

Для прямого вычисления сверток исяользуются выражения (4.1), (4.3), (4.6) или соответствующие им векторноматричные произведения, что требует порядка N^2 операций, где $Л^r$ - длина свертки. В ряде случаев одна из последовательностей (для определенности будем считать {h (и)} известна заранее. Такой последовательностью может быть импульсная характеристика фильтра или опорная последовательность коррелятора. Использование особенностей структуры {h («)j- позволяет часто существенно сократить число операций. В частности, сокращение почти всегда возможно, если{й (и)} — бинарная последовательность. В этом случае выполняется умножение вектора S = [s(0), s(1), ..., S(JV—1)]^T на бинарную матрицу. Оно может быть построено итеративно. Сначала вычисляются суммы, соответствующие соседним парам столбцов матрицы и элементов вектора S: 1 Н 2; 3 и 4 ит.д. Затем эти результаты используются для образования сумм соседних четырех элементов в столбцах матрицы: 1, 2, 3, 4; 5, 6, 7, 8 и т. д. Сокращение объема вычислений при такой процедуре возможно вследствие того, что на каждой итерации число различных сумм ограничено некоторой постоянной величиной, меньшей или равной числу строк матрицы. При обычном способе умножения эти ограничения не учитываются. Рассмотрим этот вопрос подробнее.

На первой итерации образуются суммы, соответствующие парам соседних столбцов матрицы и элементов вектора: 1 и 2; 3 и 4. Каждая пара столбцов содержит в своих строках только четыре различных числа 1, —1; —1, —1; —1, 1; 1,1, причем два числа являются инверсиями двух других. Этим числам соответствуют четыре различные суммы $\pm s(i) \pm s(i+1)$, на вычисление которых достаточно затратить только две операции сложения (вычитания). Общее количество операций на первой итерации равно 2 (N/2) = *N*. На второй итерации используются соседние четверки столбцов матрицы, а количество различных сумм в каждой четверке не превышает количества различных четырехразрядных чисел, т. е. 2*. Поскольку половина из них получается инвертированием другой половины, то на вычисление расходуется не более $(2^4/2)$ (7V/4) = *2N* операций. Рассуждая аналогично, получим, что на i-й итерации число операций не превышает величины 2^{i-1} (Л/2'). Заметим, кроме того, что число сложений (вычитаний) на /-й итерации не может быть больше, чем *N(N/2'),* поэтому в общем подсчете должно участвовать меньшее из этих чисел.

В табл. 4.1 показан выигрыш т? в количестве операций при рассмотренном методе умножения для квадратных матриц с $N = 2^n$.

Таблица 4.1

n	4	6	8	10	12	14	16
V	2,50	3,50	5,10	7,00	7,73	7,93	10,63

Проиллюстрируем сказанное примером.

Пример 4.4. Рассмотрим вычисление циклической свертки, матрица которой имеет следующий вид:

$$\begin{bmatrix}
1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 \\
1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & 1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 \\
1 & -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 \\
-1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\
-1 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 \\
1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 \\
-1 & 1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 \\
1 & 1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & 1 & -1
\end{bmatrix}$$

Процесс вычислений изображен на рис. 4.1 в виде графа. На первой итерации вычисляются сумма и разность соседних отсчетов вектора. Столбцы матрицы 1, 2, 3,4 и5,6,7,8 содержат в своих строках по восемь чисел, не являющихся инверсиями друг друга. Оставшиеся столбцы 9, 10, 11 содержат по четыре таких числа. Все эти числа записаны во второй итерации. Суммы элементов вектора, соответствующие этим числам, получаются путем сложения и вычитания результатов первой итерации. На третьей итерации аналогично образуются суммы, соответствующие восьмеркам соседних столбцов матрицы и элементов вектора. Поскольку столбцы с 12-го по 16-й в исходной матрице заполняются нулями, то часть вычислений на третьей итерации сводится просто к передаче ранее полученных результатов. Вычисления заканчиваются на четвертой итерации после выполнения 50 операций. Обычный метод умножения требует 110 операций.

Важным классом бинарных матриц являются матрицы Адамара. Эти матрицы (см. гл. 3) строятся итеративно:

$$\mathbf{H}_{2N} = \begin{bmatrix} \mathbf{H}_N & \mathbf{H}_N \\ \mathbf{H}_N & -\mathbf{H}_N \end{bmatrix},$$

Поэтому пары, четверки, восьмерки соседних столбцов содержат соответственно два, четыре, восемь различных чисел. На каждой итерации число различных чисел увеличивается вдвое. В то же время происходит сокращение вдвое количества промежуточных сумм за счет суммирования пар, четверок, восьмерок и т. д. В результате число узлов графа на каждой итерации остается постоянным и равным W, а вычислительный процесс заканчивается через $TV \log_2 TV$ операций. С этой точки зрения алгоритмы быстрого преобразования Уолша—Адамара можно трактовать как частный случай описанного алгоритма.

Важность матриц Адамара заключается в том, что к ним можно свести матрицы сверток ряда широкоупотребительных сигналов [9] . В качестве при-

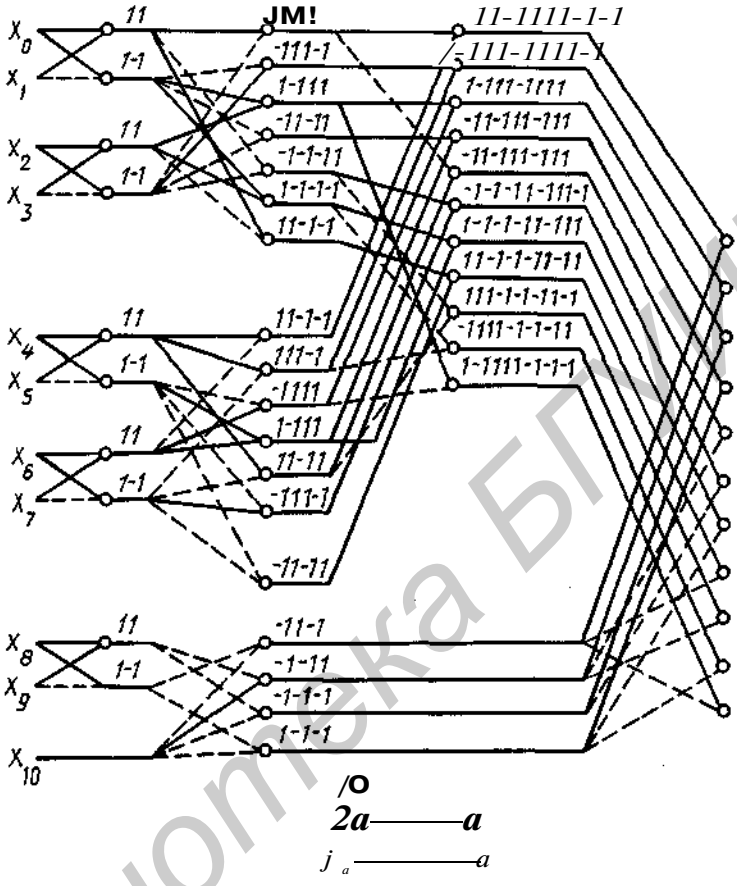


Рис. 4.1. Граф вычислительного процесса:
1 — суммирование; 2 — передача прямая; *3* - смена знака числа.

мера приведем последовательность максимальной длины. Для *N= 1* матрица свертки этой последовательности имеет вид

$$\begin{bmatrix} 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 \end{bmatrix} \cdot$$

Переставим строки матрицы следующим образом: 1 -» 1, 2 -* 2, 3 -* 5, 4->3,5-*7,6->-6,7->-4.

Запись *i-*j* означает, что строка с номером *i* ставится на место строки с номером/. Получим:

$$\begin{bmatrix} 1 & 1 & -1 & 1 & -j & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -j & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -j & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & -1 & 1 \end{bmatrix}.$$

Теперь сделаем следующую перестановку столбцов: 1 -»-4,2 ~»2, 3 → 1, 4^-6,5 -*3, 6-*7,7->5. Записываем

$$\begin{bmatrix} -1 & 1 & -1 & 1 & -2 & 1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 & -2 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Нетрудно видеть, чт последняя матрица отличается от матрицы Адамара порядка восемь отсутствием первых строки и столбца. Поэтому вычисление свертки можно выполнить следующим образом. Позиции исходной последовательности переставляются по закону перестановки столбцов, и полученный вектор дополняется слева одним нулевым отсчетом. В результате этих действий получим вектор

$$S = [0, s(3), s(2), s(5), 5(1), s(7), x(4), x(6)]^T.$$

Этот вектор умножается на матрицу Адамара порядка восемь. Результаты умножения переставляются по закону обратной перестановки строк, т. е. 1 + 1, 2 + 2, 3 + 4, 4-+7, 5 + 3, 6 + 6, 7 + 5.

Сведение матриц реальных сигналов к матрицам Адамара более подробно рассмотрено в главе 5.

4.3. ВЫЧИСЛЕНИЕ СВЕРТОК ПРИ ПОМОЩИ БЫСТРЫХ ОРТОГОНАЛЬНЫХ ПРЕОБРАЗОВАНИЙ

В основе рассматриваемых в этом параграфе методов вычислений лежат теоремы о свертке для преобразований Фурье, Уолша и ТЧП, утверждающие, что спектр свертки равен произведению спектров сворачиваемых последовательностей. Поскольку все рассмотренные преобразования обладают аналогичными свойствами, то их можно объединить понятием обобщенного преобразования Фурье. Обозначим матрицу этого преобразования Т . Тогда свертка двух векторов

$$S = [s(0), s(1) \ldots s(N-1)]^T \quad , n=[h(O), h(l). \ldots h(N-l)]^T$$

равна

$$Y = T^{-1}(TS \cdot TH) = [y(0), y(1), \ldots, y(N-1)] \quad . \qquad (4.7)$$

Таким образом, для получения свертки следует выполнить следующие действия:

1)найти преобразования Фурье (обобщенные спектры) TS и TH исходных последовательностей;

2) вычислить поточечное произведение этих последовательностей TS-TH;

3) вычислить обратное преобразование Фурье T^{-1} от произведения спектров.

Хотя на первый взгляд может показаться, что такой метод вычисления свертки довольно сложен, он тем не менее позволяет во многих случаях сократить объем вычислений. Это происходит вследствие того, что для умножения на матрицы T и T^{-1} существуют быстрые алгоритмы с числом операций, пропорциональным $N\log_2 N$. Для многих приложений один из векторов (например, H) известен заранее, что позволяет предварительно вычислить произведение TH. В этом случае вычисление свертки заключается в выполнении двух быстрых преобразований и перемножении N чисел. Заметим, что для БПФ и ТЧП это циклические свертки. Если же используется преобразование Адамара, то вычисляется диадная свертка.

Как уже отмечалось, недостатками БПФ являются необходимость работы с комплексными числами и шумы округления. ТЧП свободно от этих недостатков, но имеет свои — деление на модуль и трудность совместной оптимизации параметров *N,aяM.*

Для вычисления линейной свертки двух последовательностей длины N_1 и N_2 можно также воспользоваться соотношением (4.7), но при этом исходные данные последовательности следует дополнить нулевыми отсчетами так, чтобы их длина стала равной $N_1 + N_2 - 1$, и рассматривать как периодические.

Т і р и м е р 4.5. Вычислить линейную свертку последовательностей $S = [2, -2, 1]^T$ и $H=-[1.2]^T$.

Так как Л^ = 3 и N_2 = 2, то Л^ + Л^ - 1 = 4, и следует вычислить циклическую свертку последовательностей

$$S' = [г.\text{-}гл.о]^{3"};$$

$$H^* = " [1,2.0,0]^{г}.$$

В качестве преобразования возьмем преобразование Фурье длины 4 с матрицей

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}.$$

Прямые преобразования последовательностей равны

$$TS' = [1,1+2j, 5, 1-2j]^{г,}$$

$$Trf = [3, 1-2j, -1, 1+2j]^{T}.$$

Их произведение

$$L = TS'\text{-}TH' = [3, j, -5, 5]^{г}.$$

Обратное преобразование дает значения свертки

$$Y = 4^{-1}T^{1}L = 4^{-1}[8,8,-12,8]^{T} = [2,2,-3,2]^{T}.$$

В тех случаях, когда одна последовательность намного длиннее другой, используют разбиение длинной последовательности на короткие секции. Затем вычисляются короткие свертки и из них формируется конечный результат. Такая ситуация встречается в цифровой фильтрации, когда фильтруемые последовательности намного длиннее по сравнению с короткой импульсной характеристикой.

Существует два метода секционирования — метод перекрытия с суммированием и метод перекрытия с накоплением. Предположим, что более длинной является последовательность *[s(ri)j* . Она разбивается на блоки по Л^отсчетов. Последовательность *{h* (и)} имеет длину *L* . Линейная свертка каждого из блоков последовательности {s («)} с последовательностью *(h* (Н)J- имеет размер *N+ L — 1* и перекрывается со сверткой следующего блока в Ь-1 отсчетах. Поэтому на участке перекрытия их отсчеты следует сложить. Таким образом, на каждые *L* входных отсчетов вычисляется *N+L* — 1-точечная циклическая свертка и выполняется *L—1* сложений.

В методе перекрытия с накоплением длинная последовательность-^ *(ri)J* разбивается на секции по *N* отсчетов так, что соседние секции перекрываются в *L—1* отсчетах. Последовательность *\h (ri)$* дополняется нулевыми значениями до длины *N,* и вычисляются циклические свертки каждой секции с дополненной последовательностью {А (и)} .Первые L-1 отсчетов каждой секционной свертки отбрасываются, а остальные присоединяются к оставшимся отсчетам предыдущей, секции. Алгоритм перекрытия с накоплением дает *N - 1+* 1 отсчетов свертки без дополнительного суммирования, поэтому его реализация проще.

Если длина блока увеличивается, то общее количество блоков уменьшается. При этом число преобразований становится меньше, но время выполнения каждого преобразования возрастает. С другой стороны, сокращение длины блока потребует большего числа коротких преобразований, поэтому существует оптимум между длиной блока *N* и длиной короткой последовательности *L* . Оптимальные значения N и *L* приведены в табл. 4.2.

Таблица 4.2

L	*N*	*L*	*N*
11-17	64	311-575	4096
18-29	128	576-1050	8192
30-52	256	1051-2000	16384
53-94	512	2001-3800	32768
95-171	1024	3801-7400	65536
172-310	2048	7400	131072

tAA, ВЫЧИСЛЕНИЕ КОРОТКИХ СВЕРТОК И ПРОИЗВЕДЕНИЙ ПОЛИНОМОВ

Вычисление сверток с помощью БПФ и ТЧП является одним из наиболее

употребительных и универсальных способов сокращения объема вычислений. Однако при использовании БПФ вычисления выполняются с трансцендентными функциями — синусом и косинусом — и в комплексной арифметике. При практической реализации это приводит к ошибкам счета и двухканальному вычислителю (канал действительной части и канал мнимой части). ТЧП реализуется одним каналом, но в силу жестких теоретико-числовых связей между параметрами N, a, M зачастую не удается достичь желаемого эффекта по ускорению вычислений.

Для коротких сверток существуют более эффективные алгоритмы, основанные на специальных способах умножения полиномов. Наряду с самостоятельным применением эти алгоритмы можно использовать и для вычисления больших сверток, заменяя их на последовательность коротких. Наиболее важными из этих алгоритмов являются алгоритм Тоома—Кука и алгоритм, основанный на китайской теореме об остатках.

Алгоритм Тоома-Кука используется для вычисления линейной свертки (см. § 2.5). В основе алгоритма лежит представление свертки в виде произведения двух полиномов

$$y(z) = s(z)h(z),$$

где $h(z)$ - полином степени L; $s(n)$ - полином степени N. Значениями свертки являются коэффициенты полинома $y(z)$: $y(0), y(l), \ldots, y(N+L-2)$.

Теоретически алгоритм Тоома—Кука требует только $N + L - 2$ умножений, однако на практике к ним добавляются еще умножения на фиксированные константы.

Для вычисления циклических сверток используется китайская теорема об остатках.

4.5. КИТАЙСКАЯ ТЕОРЕМА

45.1. Полиномы над полем

В § 2.4 было введено понятие сравнения для чисел и полиномов, а также указано, что между теорией числовых и полиномиальных сравнений существует тесная связь. Остановимся теперь на этом вопросе более подробно.

Арифметические операции над полиномами сводятся к операциям над их коэффициентами, поэтому для получения соответствия между числами и полиномами операции над коэффициентами должны быть строго определены. Обычно полагают, что коэффициенты полиномов принадлежат некоторому полю, а сложение и умножение коэффициентов рассматриваются как операции в поле. Полиномы, удовлетворяющие этим требованиям, называются *полиномами над полем*.

Выбор поля коэффициентов существенно влияет на свойства полиномов. Чтобы продемонстрировать это, рассмотрим, например, полином $f(z) = z^2 + z + 1$. Если считать, что коэффициенты этого полинома принадлежат двоичному полю, состоящему из элементов 0 и 1, то в этом поле полином не имеет корней, так как $f(0) = f(1) = 1$. Поэтому он *неприводим,* т. е. не может быть разложен на множители. В то же время хорошо известно, что в поле комплексных чисел уравнение $z^2 + z + 1 = 0$ имеет два корня z_1 и z_2, и, сле-

довательно, полином раскладывается на множители: $f(z) = z^4 + z + 1 =$

Зависимость свойств полинома от выбора поля коэффициентов позволяет минимизировать вычислительную сложность многих задач цифровой обработки. Можно доказать, что множества полиномов и операций сложения и умножения по модулю полинома $P(z)$ образуют кольцо, которое будет полем, если $P(z)$ — неприводимый полином.

Пример 4.6. Рассмотрим двоичное поле, состоящее из элементов 0 и 1 и операций сложения и умножения по модулю два. Пусть над этим полем задан полином $P(z) = z^4 + z + 1$. Простой проверкой нетрудно убедиться, что этот полином неприводим в поле коэффициентов, так как $P(0) = P(1) = 1$.

Все полиномиальные вычеты по модулю этого полинома образуют поле. Его элементами будут полиномы

$a_1 = 1;$ $a_{10} = z + z^3;$

$a_2 = z;$ $a_{11} = 1 + z + z^2;$

$a_3 = z^2;$ $a_{12} = z + z^2 + z^3;$

$a_4 = z^3;$ $a_{13} = 1 + z + z^2 + z^3;$

$a_5 = 1 + z;$ $a_{14} = 1 + z^2 + z^3;$

$a_6 = z + z^2;$ $a_{15} = 1 + z^3.$

$a_7 = z^2 + z^3;$

$a_8 = 1 + z + z^3;$

$a_9 = 1 + z^2;$

В этом расширенном поле операциями будут операции сложения и умножения по модулю полинома $P(z)$. Нетрудно убедиться, что каждый элемент имеет обратный. Например, a_5 и a_{10} являются взаимнообратными, так как $a_5 a_{10} = (z + z^2)(1 + z + z^2) = z^4 + 2z^3 + 2z^2 + z = z^4 + z = z + 1 + z = 1,$

4.5.2. Китайская теорема об остатках

Рассмотрим систему сравнений с различными модулями. Требуется найти целое x, удовлетворяющее одновременно k сравнениям:

$$x \equiv r_i \bmod m_i, \quad i = \overline{1, k}.$$

Решение получается на основании теоремы, которая была известна еще в Древнем Китае и поэтому получила название китайской.

Теорема 4.1. Пусть m_i, $i = 1,2,\ldots, k$, — положительные попарно взаимно простые числа, большие единицы. Тогда система k сравнений $x \equiv r_i \bmod m_i$ имеет единственное по модулю M решение, где $M = \prod_{i=0}^{k} m_i$. Оно равно

$$x = \sum_{i=1}^{k} (M/m_i) t_i r_i \bmod M, \qquad (4.8)$$

а величины t_i должны удовлетворять условию

$$(M/m_i)\ T_j \equiv i \bmod m_i \qquad (4.9)$$

Теорема позволяет восстановить некоторое число x, если известны остатки r_i от деления этого числа на взаимно простые модули m_i. Число T_i является обратным к числу Mjm_i. Из теоремы Эйлера (3.62) следует, что $\Gamma_i \equiv (M/n_i)^{ip(m_i)-j} \bmod m_i$, поэтому равенство (4.8) можно записать по-иному:

$$x \equiv \sum_{i=1}^{k} (M/m_i)^{j} r_i \bmod M .$$

Пример 4.7. Найти число, удовлетворяющее следующей системе сравнений: $\langle x \rangle_7 = 2$, $\langle x \rangle_8 = 5$, $\langle x \rangle_9 = 5$, $\langle x \rangle_{11} = 6$.

Построим составной модуль M и величины $M_i = M/m_i$: $M = 7 \cdot 8 \cdot 9 \cdot 11 = 5544$; $M_1 = 8 \cdot 9 \cdot 11 = 792$; $M_2 = 7 \cdot 9 \cdot 11 = 693$; $M_3 = 7 \cdot 8 \cdot 11 = 616$; $M_4 = 7 \cdot 8 \cdot 9 = 504$. Теперь необходимо найти величины T_i, обратные M_i, т. е. удовлетворяющие сравнениям (4.9). Число T_2 является решением сравнения $\langle 792 \rangle_7 T_1 \equiv 1 \bmod 7$. Этому сравнению эквивалентно более простое $\langle 792 \rangle_7 T_1 \equiv 1 \bmod 7$, решение которого $T_1 = 1$. Аналогично, рассматривая сравнения

$$\langle 693 \rangle_8 \Gamma_2 \equiv 1 \bmod 8;$$

$$\langle 616 \rangle_9 \Gamma_3 \equiv 1 \bmod 9;$$

$$\langle 504 \rangle_{11} \Gamma_4 \equiv 1 \bmod 11;$$

получаем эквивалентные

$$5T_2 \equiv 1 \bmod 8;$$

$$4\Gamma_3 \equiv 1 \bmod 9;$$

$$9\Gamma_4 \equiv 1 \bmod 11.$$

Решения их можно найти простой проверкой (перебором) **или по** теореме **Эйлера. Они равны** $T_2 = 5$, $T_3 = 7$, $T_4 = 5$. Для искомого числа получим

$$x \equiv (M_1 r_1 T_1 + M_2 r_2 T_2 + M_3 r_3 T_3 + M_4 r_4 T_4) \bmod M =$$

$$= (792 \cdot 2 \cdot 1 + 693 \cdot 5 \cdot 5 + 616 \cdot 5 \cdot 7 + 504 \cdot 6 \cdot 5) \bmod 5544 = 149.$$

В силу рассмотренной аналогии между числами и полиномами китайскую теорему об остатках можно распространить и на кольцо полиномов по модулю полинома $P(z)$.

Пусть $P(z)$ есть произведение d взаимно простых полиномов (полиномов без общих множителей), т. е.

$$P(z) = \prod_{i=1}^{d} P_i(z) .$$

Кажцый полином Л (г) кольца однозначно определяется вычетами $h_i(z)$ по

модулям $p_i(z)$. Китайская теорема для полиномов позволяет непосредственно восстановить $h(z)$ по\го вычетам при помощи следующих формул:

$$h(z) \equiv \sum_{i=0}^{d} m_i(z) h_i(z) \bmod P(z), \qquad (4.10)$$

где

$$m_u(z) = t_u(z) \prod_{\substack{i=\\ /\Phi u}}^{d} p_i(z). \qquad (4.11)$$

Обратный полином $t_u(z)$ должен удовлетворять сравнению

$$t_u(z) \prod_{\substack{i=1\\ \}\Phi u}}^{d} p_i(z) = 1 \bmod p_u(z). \qquad (4.12)$$

При использовании китайской теоремы встречаются следующие трудности — определение обратных чисел 7_i\ или полиномов $t_u(z)$, удовлетворяющих сравнениям (4.9) и (4.12), и вычисление произведения двух полиномов по модулю некоторого третьего полинома.

Для числбвых полей и колец обратные элементы можно найти простым перебором или по теореме Эйлера. Другой способ нахождения обратных элементов как для чисел, так и для полиномов дает *алгоритм Евклида.*

4.5.3. Алгоритм Евклида

Наибольшее положительное целое d, делящее целые числа a и b, называется *наибольшим общим делителем* (НОД) и обозначается $d = (a, b)$. Если $d = (я, Ь) = 1$, то|2и&не имеют общих делителей, отличных от 1, и называются *взаимно простыми,* НОД можно найти с помощью алгоритма Евклида. Для описания алгоритма положим $a > b$. Разделив a на $Ь$, получим:

$$\begin{aligned} a &= bq_0 + r_0, & 0 < r_0 < b; \\ Ь &= r_0 q_{i+} \; r_{1>} & 0 < r_x < r_V, \\ r_o &= r_x \wedge^{+} {}^{2} r > \circ < {}^{2} r < r_i > \end{aligned} \qquad (4.13)$$

$$\begin{aligned} {}^{2}n\text{-}з^{=} \quad & {}^{r}{}_{n}\text{-}2 < l_n\text{-}i^{*r}n\text{-}i > \quad {}^{0 < r} n \wedge i < {}^{r} n - {}_{3} > \\ {}^{2}n\text{-}г^{=} \quad & {}^{2}n \sim x\%\text{-} \end{aligned}$$

Последний остаток $г_{n-1}$ и есть НОД.

Пример 4.8. Найти НОД чисел 525 и 231. Выполняя вычисления, получаем:

$$\begin{aligned} 525 &= 231\text{-}2 + 63; \\ 231 &= 63\text{-}3 + 42; \\ 63 &= 42\text{-}1 + 21; \\ 42 &= 21\text{-}2 \end{aligned}$$

Так как последний остаток равен 21, то (525, 231) = 21.

Заметим, что НОД является линейной комбинацией чисел *a nb* . Чтобы это увидеть, перепишем равенства (4.13) так:

$$
\begin{aligned}
r_0 &= a - bq_0;\\
r_1 &= b - r_0 q_1;\\
&\cdots\cdots\cdots\\
r_n &= r_{n-2} - r_{n-1} q_n.
\end{aligned}
\qquad (4.14)
$$

Первое число r_0 является линейной комбинацией a и b ; второе r_1 - линейной комбинацией r_0 и r_1 , следовательно, линейной комбинацией a и b и т. д. Поэтому можно записать

$$(a,b) = ka + lb \ , \qquad (4.15)$$

где k , l — целые числа. Это соотношение дает ключ к решению уравнений в целых числах.

Рассмотрим линейное целочисленное уравнение с коэффициентами a ,b ,c:

$$ax + by = c,$$

где x,y— целочисленные неизвестные. Уравнения такого вида называются*диофантовыми*.

Легко показать, что если c не делится на $d = (a, b)$, то диофантово уравнение не имеет решения. Поэтому далее будем считать, что $d \mid c$(Оделитс). Если $d \mid c$,то $c = c_1 d$ и из (4.15) следует, что существуют такие k и l , что

$$d = ka + lb,$$

поэтому

$$c = c_1 d = c_1 ka + c_1 lb \ , \qquad (4.16)$$

т. е. решениями диофантова уравнения являются $x = c_1 k$,$y = c_1 l$. Это решение не является единственным. Пусть x_0 , y_0 — частное решение, т. е. $c = ax_0 + by_0$. Можно доказать, что общее решение имеет вид:

$$
\begin{aligned}
x &= x_0 + (b/d)s \ ; \qquad (4.17)\\
y &= y_0 - (a/d)s \ ,
\end{aligned}
$$

где s — целый параметр.

Пример 4.9. Найти решение уравнения $15x + 9y = 21$. По алгоритму Евклида,

$$
\begin{aligned}
15 &= 9 \cdot 1 + 6;\\
9 &= 6 \cdot 1 + 3;\\
6 &= 3 \cdot 2 .
\end{aligned}
$$

НОД $d = (a, b) = 3$. Так как $c \mid 21$, то уравнение имеет решение. Представим d в виде линейной комбинации 15 и 9 (4.14) :

$$
\begin{aligned}
6 &= 15 - 9 \cdot 1 ;\\
3 &= 9 - 6 \cdot 1 = 9 - (15 - 9 \cdot 1) \cdot 1 = -15 + 2 \cdot 9.
\end{aligned}
$$

Получим $k = -1$, $l = 2$, $q = 7$. Это Дает частное решение $x = -7$, $y = 14$ (4.16). Общее решение (4.17) имеет вид $x = -7 + 3s$, $y = 14 - 5s$.

Введем понятие *непрерывной дроби.* Разделив a на b, получим

$$\frac{a}{b} = q_0 + \frac{r_0}{b} = q_0 + \frac{1}{b/r_0}.$$

Теперь разделим b на r_0:

$$\frac{b}{r_0} = q_1 + \frac{r_1}{r_0} = q_1 + \frac{1}{r_0/r_1}.$$

Продолжая этот процесс, запишем непрерывную дробь

$$\frac{a}{b} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{\ddots + \cfrac{1}{q_n}}} \qquad (4\text{-}18)$$

Она позволяет построить последовательность приближений к a/b, называемых подходящими дробями:

$$\frac{a_0}{b_0}, \frac{a_1}{b_1} = q_0 + \frac{1}{q_1}, \ldots \qquad (4.19)$$

Эти дроби обладают следующими свойствами:

1) $a_k = q_k a_{k-1} + a_{k-2}$,
$b_k = q_k b_{k-1} + b_{k-2}$, $k > 2$, (4.20)

2) $b_k a_{k-1} - a_k b_{k-1} = (-1)^k$, $k > 0$; (4.21)

3) $(a_k, b_k) = 1$. (4.22)

Тождества (4.20)—(4.22) позволяют упростить вычисления. Например, используя (4.21) для последнего шага алгоритма Евклида, получаем

$$a = (a - b[a/b]) \cdot (-1)^{-1}. \qquad (4.23)$$

Но $a_n/b_n = a/b$. Если $(a,b) = 1$, то $a_n = a$, $b_n = b$, т. е. (4.23) дает способ решения уравнения $ax + by = 1$, если положить

$$x = (-1)^{n-1} a_{n-1}, \quad y = (-1)^{n} b_{n-1}. \qquad (4.24)$$

Таким образом, алгоритм Евклида позволяет как найти НОД, так и решить диофантово уравнение.

Заметим, что поскольку при выводе алгоритма Евклида полагалось $a > b$,

то это условие должно выполняться и для коэффициентов диофантова уравнения. В противном случае следует переименовать неизвестные *хну* .

Вернемся теперь к задаче нахождения обратных элементов, необходимых для восстановления числа или полинома по китайской теореме. Эта задача сводится к решению сравнения вида же $\equiv 1 \bmod m$. Решением является элемент, обратный элементу а (в кольце или поле). Алгоритм Евклида позволяет найти этот элемент.

Чтобы показать это, заметим, что уравнение не изменится, если к его левой части прибавить величину, кратную модулю, т. е.

$$ax + my \equiv 1 \bmod w ,$$

где y - целое.

Полученное сравнение эквивалентно диофантовому уравнению, в котором все члены определены по модулю *m:*

$$< ax > + < my > = 1.$$

Поскольку с = 1, то из предыдущего следует, что сравнение имеет единственное решение.

Пример 4.10. Найти решение сравнения

$$693л: \equiv 1 \bmod 8 .$$

Так как $< 693 >_8 = 5$, то оно эквивалентно более простому сравнению $5x \equiv 1 \bmod 8$, которое, в свою очередь, эквивалентно диофантовому уравнению

$$5x + 8y = 1 .$$

Запишем непрерывную дробь (см. (4.18))

$$\frac{8}{5} = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{2}}},$$

что дает $n = 3$,

$$\frac{a_{л-1}}{b_{n-1}} = 1 + \cfrac{1}{1 + \frac{1}{1}} = \frac{3}{2} .$$

Так как $a < Ъ$, то, переименовав переменные с учетом (4.24), получим:

$$y = (-1)^{n+1} V, \quad = 2;$$

$$x = (-1)^{n} a_{n-1} = -3 .$$

Решением сравнения будет вычет $< -3 >_8 = < 5 >_8$.

Пример 4.11. Пусть $P(z) = z^5 - 1$. Можно доказать, что

$$z^5 - 1 = (z - 1)(z^4 + z^3 + z^2 + z + 1).$$

Найдем обратные полиномы $t_1(z)$ и $\Gamma_2(z)$, такие, что

$$t_1(z)(z-1) = 1 \bmod (z^4 + z^3 + z^2 + z + 1); \quad \textbf{(4.25)}$$

$$t_2(z)(z^4 + z^3 + z^2 + z + 1) = 1 \bmod (z - 1). \quad \textbf{(4.26)}$$

Эти полиномы являются решениями диофантова уравнения

$$(Z-1)\,f_1(Z) + (Z^4 + Z^3 + Z^2 + Z + 1)\,\Gamma_2(Z) = 1, \quad (4.27)$$

поскольку при выборе в качестве модуля полинома $z^4 + z^3 + z^2 + z + 1$ уравнение (4.27) эквивалентно сравнению (4.25), а при выборе в качестве модуля полинома $(z - 1)$ уравнение эквивалентно сравнению (4.26).

Запишем разложение в непрерывную дробь

$$\frac{z^4 + z^3 + z^2 + z + 1}{z - 1} = z^3 + 2z^2 + 3z + 4 + \frac{1}{\frac{1}{5}(z - 1)}.$$

Подходящие дроби (см. (4.19)) равны:

$$\frac{a_0(z)}{b_0(z)} = z^3 + 2z^2 + 3z + 4;$$

$$\frac{a_1(z)}{b_1(z)} = \frac{(z-1)(z^4 + z^3 + z^2 + z + 1)}{\frac{1}{5}(z - 1)},$$

т. е. $n = 1$ и, переименовав переменные в (4.23), получим

$$a_{01}(z)b_1(z) - b_0(z)a_1(z) = -1.$$

Отсюда

$$\frac{1}{5}(z-1)(z^3 + 2z^2 + 3z + 4) - \frac{1}{5}(z^4 + z^3 + z^2 + z + 1) = -1,$$

что дает

$$t_1(z) = -\frac{1}{5}(z^3 + 2z^2 + 3z + 4);$$

$$t_2(z) = 1/5.$$

В справедливости сравнений (4.25), (4.26) можно убедиться непосредственной проверкой.

ВЫЧИСЛЕНИЕ КОРОТКИХ СВЕРТОК С ПОМОЩЬЮ КИТАЙСКОЙ ТЕОРЕМЫ ОБ ОСТАТКАХ

Ранее было показано, что вычисление циклической свертки двух последо-

вательностей{л(и)} $m\{h\ \{n)j$ длины N эквивалентно задаче нахождения коэффициентов полинома

$$y(z) \equiv s(z)h\{z\ )\bmod(z^{yV} - 1) \quad . \tag{4.28}$$

Полином $z^N—l$ может быть разложен в произведение d неприводимых полиномов, называемых *круговыми* или *циклотомическими,* где d — число делителей N:

$$z^N - 1 = \prod_{i=1}^{d} p_i(z). \tag{4.29}$$

Это позволяет воспользоваться китайской теоремой (4.10), (4.11) и выполнить вычисления следующим образом:

1) полиномы $s(z)$ $nh(z)$ приводятся по модулю полиномов $P_i(z)$, ?= 1, 2. . . .d , т. е. находятся вычеты

$$s_i(z) \equiv s(z)modp\hat{}J, \qquad /z_i(z) \equiv /2(z)\bmod p_i(z);$$

2) вычисляются d произведений $s_j(z)h_j(z)$;

3) по этим произведениям с помощью китайской теоремы восстанавливается $y(z)$.

Здесь следует отметить одно очень важное обстоятельство. Поскольку, как было показано выше, разложение полинома на множители зависит от поля, из которого берутся коэффициенты, то, очевидно, и сложность алгоритма будет зависеть от поля коэффициентов. Например, полином $p(z) = z^5 — 1$ имеет следующие разложения:

над полем комплексных чисел

$$p(z) = \prod_{r=0}^{4} (z - W^r), \quad W = \exp\left(-j\frac{2\pi}{5}\right);$$

над полем действительных чисел

$$p\{z) = (z - 1)(z^2 + 2\cos\frac{2\pi}{5}z + 1)(z^2 + 2\cos\frac{2\pi}{5}z + 1);$$

над полем рациональных чисел

$$p(z) = (z - 1)(z^4 + z^3 + z^2 + z + 1).$$

Отсюда следует, что над полем комплексных чисел задача (4.28) разбивается на пять подзадач, над полем, действительных чисел — на три подзадачи, а над полем рациональных чисел - на две подзадачи. Каждая подзадача решается путем арифметических операций над коэффициентами полиномов $x_i(z)$ $nh_i(z)$. Поэтому для упрощения алгоритма поле коэффициентов следует выбирать так, чтобы операции в нем были просты в реализации. Неизвестны какие-либо универсальные способы такого выбора, но показано, что для небольших значений N использование поля рациональных чисел приводит к алгоритмам с минимальной сложностью. Это связано с тем, что в поле рациональных чисел круговые полиномы с небольшим значением индекса имеют очень простые коэффициенты: 0; 1; —1.

Для поля рациональных чисел разложение (4.29) записывается

$$z^N - 1 = \prod_{d|N} n_N C_d(z). \qquad (4.30)$$

Здесь произведение берется по всем делителям числа N. Например, если $N = 6$, то его делителями будут числа 1, 2, 3, 6 , поэтому (4.30). запишется так:

$$z^6 - 1 = C_1(z)\,C_2(z)\,C_3(z)\,C_6(z) .$$

Коэффициенты 0; 1; —1 имеют все круговые полиномы с индексом, меньшим 105. Первые десять полиномов имеют вид:

$$\begin{aligned}
&C_1(z) = z - 1;\\
&C_2(z) = z + 1;\\
&C_3(z) = z^2 + z + 1;\\
&C_4(z) = z^2 + 1;\\
&C_5(z) = z^4 + z^3 + z^2 + z + 1; \qquad (4.31)\\
&C_6(z) = z^2 - z + 1;\\
&C_7(z) = z^6 + z^5 + z^4 + z^3 + z^2 + z + 1;\\
&C_8(z) = z^4 + 1;\\
&C_9(z) = z^6 + z^3 + 1;\\
&C_{10}(z) = z^4 - z^3 + z^2 - z + 1.
\end{aligned}$$

Вернемся теперь к вычислению свертки при помощи китайской теоремы. Процедуру синтеза алгоритма рассмотрим на примере.

Пример 4.12. Синтезировать алгоритм трехточечной циклической свертки.

Для трехточечной свертки

$$y(z) = s(z)h(z) \bmod (z^3 - 1).$$

Разложение $z^3 - 1$ на множители с учетом (4.31) имеет вид

$$z^3 - 1 = C_1(z)C_3(z) = (z - 1)(z^2 + z + 1).$$

Используя алгоритм Евклида, найдем обратные полиномы, удовлетворяющие сравнениям:

$$t_1(z)C_1(z) \equiv 1 \bmod (z^2 + z + 1);$$

$$t_2(z)C_3(z) \equiv 1 \bmod (z - 1).$$

Они равны $t_1(z) = -\frac{1}{3}(z + 2)$, $t_2(z) = 1/3$.

Первый этап синтеза дает:

$$s_1(z) = s(z) \bmod C_1(z) = s(0) + s(1) + s(2) = a_1;$$

$$s_2(z) = s(z) \bmod C_3(z) = (s(0) + s(1)z + s(2)z^2) \bmod C_3(z) =$$

$$= s(0) + s(1)z + s(2)\ (-2\text{-i}) = (s(0) - s(2)) + (s(1) - s(2))z = a_2 z + a_3,$$

где Jj = s (0) + s (1) + s (2); a_2 = s (1) - s (2) ; a_3 = s (0) - s (2) .

Аналогично получим

$$h_1(z) = b_1',\ h_2(z) = b_2' z + b_3',$$

где 6j' = A(O)+A(1) + A(2); z/2 =A(0) -A (2), *'3=A(1) - A (2) .

На втором этапе подучим:

$$У_1 (г) = «j (г) fej (г) \bmod Cj (г) = «jb',\ ;$$

$$\hat{}_2 (г) = (a_2 z + aj (b'_2 z + b'_3) mod C_3(z) =$$
$$= (a_2 b'_2 z^2 + (a_3 b'_2 + a_2 b_3')z + a_3 b_3') \bmod , (г) =$$
$$= a_2 b'_2 z - a_2 b'_2 + (a_3 b'_2 + a_2 b'_3)z + a_3 b'_3 =$$
$$= (a_3 b'_2 + a_2 b'_3 - a_2 b'_2)z + a_3 b'_3 - a_2 b'_2 .$$

Для вычисления y_1 *(z)* требуется одна операция умножения,а$У_2(г)$ - четыре. Существует лучший алгоритм для вычисления. Он требует только трех операций умножения. Положим $Д_г = a_2 b'_2$; $Ц_2 = a_3 b'_3$; $д_3 = (a_3 \sim aj\ (b'_3 - b'\hat{}$. *Тогци* $y_2(z) = (/j,_2 - njz +$

На третьем этапе запишем

$$y (z) = \{C_3(z)y_1(z)t_2(,z) + C_1(z)y_2(z)t_1(z))mouz^3 - I =$$

$$= (-e_1 b_1 (z^2 + z + 1) - (/_x r + J_0)(z - 1)(z - 2)) \bmod z^3 - 1 =$$
$$= \frac{1}{3}(a_1 b'_1 - l_1 - l_0) z^2 + \frac{1}{3}(a_1 b'_1 - l_0 + 2l_1) z + \frac{1}{3}(a_1 b'_1 + 2l_0 - l_1).$$

Объединяя все соотношения, получим следующий алгоритм:

$$a_1 = s(0) + s(1) + s(2); \qquad b_1 = \frac{1}{3}(h(0) + h(1) + h(2));$$
$$a_2 = s(1) - s(2); \qquad b_2 = \frac{1}{3}(h(0) - h(2));$$
$$a_3 = s(0) - s(2); \qquad Й_3 = \frac{1}{3}i(A(1) - h(2));$$
$$^6 4 = {}^b 3 \sim {}^b 2 \ ''$$

$$\mu_0 = a_1 b_1; \qquad y(0) = \mu_0 + 2l_0 - l_1;$$
$$\mu_1 = «2*2' \qquad y(1) = \mu_0 - l_0 + 2l_1;$$
$$\mu_2 = a_3 b_3; \qquad У(2) = \mu_0 - l_1 - l_2.$$
$$\mu_3 = (a_3 - a_2) b_4;$$
$$l = \mu_2 - \mu_3;$$
$$»_0 = \mu_2 - \mu_1;$$

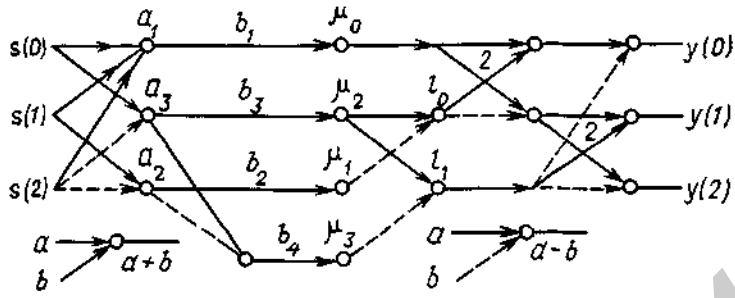


Рис. 4.2. Граф вычисления трехточечной циклической свертки.

Как уже указывалось, в реальных приложениях одна из последовательностей *{•{h (n)У*) известна и, следовательно, вычисления с переменными *h (n)* могут быть выполнены заранее, что позволяет не учитывать их в подсчете вычислительной сложности алгоритма. Процедура вычислений для такого случая показана в виде графа на рис. 4.2. Алгоритм требует 4 операции умножения и 12 операций сложения (умножения на числа 2 не учитываются).

Алгоритм можно также описать в матричной форме:

$$\mathbf{Y} = C(BH® AS), \tag{4.32}$$

где знак ® обозначает поточечное произведение векторов, а матрицы A, B, C равны:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ 1 & -1 & O \end{bmatrix}; \quad B = A/3 ; \quad \mathbf{C} = \begin{bmatrix} 1 & 1 & - & 1 & 1 \\ 1 & 1 & 1 & - & 2 \\ 1 & - & 2 & 1 & 1 \end{bmatrix}.$$

Умножение вектора $S = [s(0), s(1), x(2)]^T$ на матрицу A дает вектор *[a,* $\times^{*\cdot>}$ a_2 , $a^—a_2]^T$. Аналогично вычисляется заранее произведение $BH = [b_1, й, 6\text{-}\angle, \&_4]^T$. Их поточечное произведение дает вектор $(Д_0 , fi_2 , ji_1 , ju_3]^T$. Заключительная фаза соответствует умножению этого вектора на матрицу C . Выражение (4.32) является общей формой алгоритмов вычисления свертки с помощью китайской теоремы.

Алгоритм (4.32) иногда удается упростить, воспользовавшись принципом дуальности. В частности, можно доказать, что если существует алгоритм (4.32), то существует и дуальный алгоритм

$$Y^r = (A^R)^T [BS^r ® (C^r)^T H^r], \tag{4.33}$$

где верхний индекс *г* обозначает перестановку всех строк матрицы, кроме первой, в обратном порядке, а верхний индекс *R* - аналогичную перестановку столбцов. Подставляя в (4.33) соответствующие матрицы, получаем:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & -1 & -1 & 0 \\ 1 & 0 & 1 & -1 \end{bmatrix} \left(\begin{bmatrix} -1 & 1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} s(0) \\ x(1) \\ x(2) \end{bmatrix} ® \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -2 & 1 & 1 \\ 1 & 1 & -2 \end{bmatrix} \begin{bmatrix} h(0) \\ h(1) \\ h(2) \end{bmatrix} \right)$$

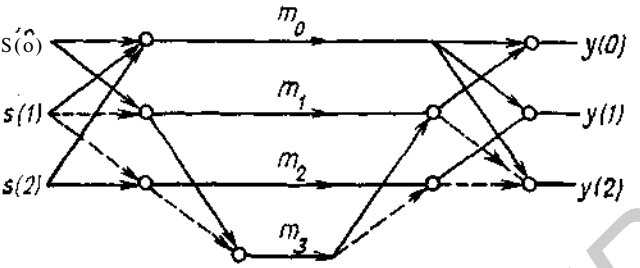


Рис. 4.3. Граф вычисления трехточечной циклической свертки для дуального алгоритма.

Граф вычислительного процесса для этого алгоритма показан на рис. 4.3.

При этом величины m_0, m_1, m_2, m_3 находятся из следующих равенств:

$$\tau_0 = (A(0) + A(1) + A(2))/3;$$

$$m_1 = (A(0) + A(1) - 2A(2))/3;$$

$$\tau_2 = (-2A(0) + A(1) + A(2))/3;$$

$$m_2 = (A(0) - 2A(1) + A(2))/3.$$

Ца вычисление свертки затрачивается 4 операции умножения и И операций сложения.

Для коротких сверток $N < 9$ по приведенной методике синтезированы алгоритмы с минимальной сложностью. Описание и структура этих алгоритмов приведены в [10, 11].

^ВЫЧИСЛЕНИЕДЛИННЫХСВЕРТОКСПОМОЩЬЮ ВЛОЖЕНИЯ КОРОТКИХ (ГНЕЗДОВОЙ АЛГОРИТМ)

Для больших сверток алгоритмы, описанные в предыдущем параграфе, становятся громоздкими и малоэффективными. Однако, если N— составное число, то матрицу свертки путем перестановки строк и столбцов можно преобразовать в блочную матрицу, каждый блок которой является матрицей циклической свертки меньшего размера, а сами блоки также образуют блочную матрицу циклической свертки. В результате такого преобразования вычисление длинной свертки можно свести к вычислению серии коротких сверток. Для того чтобы результат от перестановки не изменился, следует аналогичным образом переставить входные и выходные отсчеты.

Основой для преобразования матрицы является китайская теорема. Пусть $N = N_1 N_2$ — составное число, где N_1 и N_2 взаимно простые, т. е. $(N_1, N_2) = 1$. Любое число m из интервала 0, 1,2,..., JV—1 можно записать ввиде $((n_1, n_2))$, где $n_1 = \langle N \rangle_{N_1}$, и $n_2 = \langle N \rangle_{N_2}$. Вьиет n_1 можно рассматривать как первую координату числа N, а вычет n_2 — как вторую. Если строки исходной матрицы упорядочить по первой координате, а столбцы по второй, то получим описанную блочно-циклическую структуру. Процедура разбиения может быть повторена рекурсивно для сомножителей N_1 и N_2, если они, в свою очередь, также разбиваются на взаимно простые сомножители.

Пример 4.13. Рассмотрим вычисление шеститочечной свертки. Представим число W как 6 = 2* 3. Тогда получим следующее взаимно однозначное отображение: 0 -+ (0, 0), 1 -• (1,1), 2-* (0, 2), 3 - • (1, 0), 4 - (0,1), 5 -* (1, 2). Матричная запись исходной свертки имеет вид

$$\begin{bmatrix} y(0) \\ yil) \\ y(2) \\ y(3) \\ ;K4) \\ ;v(5) \end{bmatrix} = \begin{bmatrix} h(0) & h(1) & h(2) & h(3) & h(4) & h(5) \\ ft(1) & h(2) & h(3) & h(4) & h(5) & h(0) \\ h(2) & h(3) & h(4) & h(5) & h(0) & h(1) \\ h(3) & h(4) & h(5) & h(0) & h(1) & h(2) \\ h(4) & h(5) & h(0) & h(1) & h(2) & h(3) \\ h(5) & h(0) & h(1) & h(2) & h(3) & h(4) \end{bmatrix} \begin{bmatrix} s(0) \\ s(5) \\ s(4) \\ s(3) \\ ı(2) \\ s(1) \end{bmatrix}.$$

Переставим строки и столбцы в следующем порядке: первыми ставятся те, у которых первая координата равна 0, а вторая - в возрастающем порядке, затем те, у которых первая координата равна 1 и вторая в возрастающем порядке, и т. д. В результате получим порядок следования строк и столбцов: 0, 4, 2, 3, 1, 5 и блочно-циклическую структуру:

$$\begin{bmatrix} y\,id) \\ y(A) \\ y<2) \\ \hline \bar{y}(3 \\ y(1") \\ y(5) \end{bmatrix} = \begin{bmatrix} A(0) & A(4) & A(2) & A(3) & A(1) & A(5) \\ A(4) & A(2) & A(0) & A(1) & A(5) & A(3) \\ A(2) & A(0) & A(4) & A(5) & A(3) & A(1) \\ \hline A(3) & A(1) & A(5) & A(0) & A(4) & A(2) \\ A(1) & h(5) & A(3) & A(4) & A(2) & й(0) \\ A(5) & A(3) & A(1) & A(2) & A(0) & A(4) \end{bmatrix} \begin{bmatrix} *«>) \\ s(2) \\ s(4) \\ \hline s(3) \\ s(S) \\ s(1) \end{bmatrix}.$$

Обозначив

$$Y_0 = [y(0), y(4), y(2)]^T, \quad Y_1 = [y(3), y(1), y(5)]^T,$$

$$S_0 = [s(0), s(2), s(4)]^T, \quad S_1 = [s(3), s(5), s(1)]^T,$$

$$H_0 = \begin{bmatrix} A(0) & A(4) & A(2) \\ A(4) & A(2) & A(0) \\ A(2) & A(0) & A(4) \end{bmatrix}, \quad H_1 = \begin{bmatrix} A(3) & A(1) & A(5) \\ A(1) & A(5) & A(3) \\ A(5) & A(3) & A(1) \end{bmatrix},$$

запишем преобразованную матрицу в более компактном виде:

$$\begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} = \begin{bmatrix} H_0 & H_1 \\ H_1 & H_0 \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \end{bmatrix}.$$

Вычисление этого выражения эквивалентно вычислению получим:

$$Y_1 z + Y_0 = (H_1 z + H_0)(S_1 z + S_0) \bmod (z^2 - 1).$$

Разложение модуля на круговые многочлены равно:

$$z^2 - 1 = (z-1)(z+1).$$

Используя синтез на основе китайской теоремы (задача 4.16), получим

$$M_1 = 0{,}5(H_0 + H_1)(S_0 + S_1), \quad M_2 = 0{,}5(H_0 - H_1)(S_0 - S_1);$$

$$Y_0 = M_1 + M_2; \quad Y_1 = M_1 - M_2.$$

Так как M_1 и M_2 являются трехточечными свертками, то их вычисление можно вы-

полнить при помощи алгоритма, приведенного в примере 4.12. На вычисление M_1 и M_2 затрачивается 8 операций умножения и 22 операции сложения (умножения на 1/2 не учитываются). Еще 3-4= 12 сложений требуется на вычисление векторов $S_0 \pm S_j$, $M_1 \pm M_2$. В итоге получим 8 операций умножения и 34 операции сложения.

Можно было бы разложить число 6 как 6 = 3'2, что привело бы к трехточечной свертке блоков 2 X 2. В этом случае алгоритм потребует 8 операций умножений и 38 операций сложений.

В общем случае сложность алгоритма подсчитывается следующим образом. Пусть M_1 и M_2 - количество умножений в алгоритмах вычисления N_x и Л^-точечной сверток соответственно. Аналогично A_1 и A_2 -количество сложений. Тогда, если (iVj , N_2) = \, *то* N_1N_2 — точечная свертка, и вычисляется она за $M = M_1M_2$ операций умножения и A операций сложения. Меняя ролями N_1 и N_2, получаем, что та же свертка вычисляется за $A^N^M^A^$ сложений. Предпочтение отдается тому способу, в котором число сложений меньше.

Если JV разлагается в произведение более чем двух взаимно простых сомножителей $N^_2 \cdots ^d$ то вы Р а ж е н и я Для сложности запишутся так: $M = M_xM_2...M_d$; $A = A_xM_e ... N_d + M_1A_2N_3...N_d + M_1M_2A_3 ^...N_d + ... + M_1M_2... ...M_{d-}A_d$.

В формуле для числа операций сложений в каждом последующем слагаемом количество букв M_j увеличивается на единицу по сравнению с предыдущим, вследствие чего число операций сложения A гораздо сильнее зависит от числа операций умножения $M^$, чем от числа сложений A. Поэтому для больших сверток наиболее выгодны алгоритмы, вычисляющие короткие свертки с малым числом умножений, даже если это уменьшение достигается за счет увеличения числа сложений.

В табл. 4.3 приведено число арифметических операций на один отсчет сигнала для описанных алгоритмов.

Таблица 4.3

N	*M/N*	*A/N*
18	2,11	10,22
30	2,67	13,93
60	3,33	18,67
120	4,67	25,80
210	6,10	37,90
504	7,24	52,19
1008	9,95	70,70

Таблица 4.4

N	*M/N*	*A/N*
16	2,75	10,75
32	3,62	14,75
64	4,56	18,75
128	5,53	22,75
256	6,52	26,75
512	7,51	30,75
1024	8,50	34,75
2048	9,50	38,75

В табл. 4.4 приведены для сравнения те же величины при вычислении свертки по алгоритму двукратного БПФ. При этом предполагается, что для вычисления БПФ действительной последовательности длины N используется БПФ размера *NJ2* с комплексными входными данными, а комплексные умно-

жения выполняются тремя вещественными умножениями и тремя вещественными сложениями.

Сравнивая табл. 4.3 и 4.4, можно заметить, что гнездовой алгоритм лучше алгоритма, основанного на БПФ, для коротких и средних сверток длиной, меньшей 200-220. Преимущество гнездового алгоритма состоит в том, что он не использует тригонометрических функций и вещественные свертки используют вещественную, а не комплексную арифметику. В то же время метод, основанный на БПФ, проще в программировании и позволяет при помощи одной стандартной программы вычислить свертку для различных значений N.

4Я, МУЛЬТИПЛИКАТИВНАЯ СЛОЖНОСТЬ ВЫЧИСЛЕНИЯ СВЕРТКИ

При синтезе алгоритмов с малой вычислительной сложностью желательно знать нижние предельные значения этой величины, что позволяет судить о качестве алгоритма и возможностях его улучшения. Рассмотрим вопрос о нижних границах применительно к операциям умножения.

Т е о р е м а 4.2. Линейная свертка двух последовательностей длины N может быть вычислена за $2N-1$ умножений.

Доказательство теоремы основано на алгоритме Тоома-Кука. Значениями линейной свертки являются коэффициенты полинома

$$y(z) = s(z)h(z)$$

степени $2N-2$. По алгоритму Тоома—Кука выбираются $2N-1$ точек интерполяции z_i и находятся $2N-1$ произведений:

$$y(z_i) = s(z_i)h(z_i), \quad i = 0,1,2,\ldots,2N-2. \tag{4.34}$$

Для построения полинома $y(z)$ используются произведения (4.34) и интерполяционная формула Лагранжа. Если вычисление $s(z_i)$, $h(z_i)$ и интерполяция не требуют умножений, то для вычисления свертки используются только $2N-1$ умножений вида (4.34).

Рассмотрим теперь циклическую свертку.

Т е о р е м а 4.3. Минимальное число умножений, необходимое для вычисления циклической свертки длины N, равно $2N-\kappa$, где κ — число делителей N, включая 1 и N.

В доказательстве теоремы используется алгоритм вычисления свертки по китайской теореме, а именно: вычисляется полином

$$y(z) = s(z)h(z) \bmod (z^N - 1).$$

Для этого полином z^N-1 раскладывается в произведение круговых полиномов, число которых равно κ. Если d_j — делитель числа N, то соответствующий этому делителю круговой полином C_{d_j} имеет степень d_j, а сумма степеней всех круговых полиномов равна N. Для вычисления свертки по китайской теореме необходимо вычислить

$$y_j(z) = s_j(z)h_j(z) \bmod C_{d_j}(z), \quad j = 1,2,\ldots,*. \tag{4.35}$$

По теореме 4.2 каждое произведение типа (4.35) можно вычислить за $2n_j$ - 1 умножений. Суммируя по всем/ , получаем

$$\sum_{j=1}^{k} (2n_j - 1) = IN - k .$$

Сравним границу теоремы 4.3. с реальными значениями числа умножений для минимальных алгоритмов, описанных в [10] . Сравнение дается в табл.4.5.

Таблица 4.5

N	κ	$2N - \kappa$	Число операций	
			умножения	сложения
2	2	2	2	4
3	2	4	4	11
4	3	5	5	15
5	2	8	10	31
6	4	8	8	34
7	2	12	16	70
8	4	12	14	46
9	3	15	19	81

Из табл. 4.5 видно, что для сверток малой длины граница мультипликативной сложности достигается, и, следовательно, улучшение алгоритмов возможно только за счет сокращения числа операций сложения.

Можно доказать [10, 11] , что все алгоритмы с минимальной мультипликативной сложностью имеют структуру вида (4.32), (4.33), т. е.

$$Y = C(BH \circledR AS) , \qquad (4.36)$$

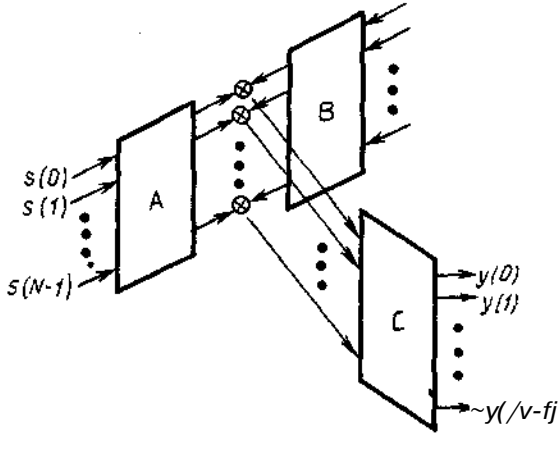


Рис. 4.4. Общая форма алгоритма вычисления свертки.

а $2N - 1$ умножений являются поточечными умножениями векторов BH и AS. Общий вид алгоритма показан на рис. 4.4. Заметим, что вычисление циклической свертки с помощью ДПФ - частный случай структуры, изображенной на рис. 4.4, когда преобразования В и А есть ДПФ, а С - ОДПФ. При этом в качестве поля коэффициентов выбрано поле комплексных чисел, в котором полином $z^N - 1$ раскладывается на N круговых полиномов:

$$z^N - 1 = \prod_{l=0}^{N-1} (z - W^l).$$

Поэтому $\kappa = N$ и N умножения являются не чем иным, как поточечными умножениями двух ДПФ. Ни одно из умножений в поле коэффициентов (умножения на поворачивающие множители) не учитывается.

4.9. АЛГОРИТМ ВИНОГРАДА ПРЕОБРАЗОВАНИЯ ФУРЬЕ

4.9.1. Гнездовой алгоритм

Рассмотренные методы вычисления сверток с помощью китайской теоремы и гнездового алгоритма, кроме самостоятельного значения, играют важную роль в построении алгоритма вычисления ДПФ, получившего название *алгоритма Винограда*. Этот *алгоритм* сокращает число умножений по сравнению с обычным методом БПФ в 2—3 раза при незначительном увеличении числа сложений. Он особенно удобен для вычисления ДПФ вещественных последовательностей. В этом случае большинство сложений и все умножения вещественные, что не только повышает скорость обработки, но и экономит память.

В основе алгоритма Винограда лежат две основные идеи — замена длинного преобразования серией коротких путем построения гнездового алгоритма, аналогичного алгоритму вычисления сверток, и вычисление коротких преобразований путем сведения их к циклическим сверткам.

Для построения гнездового алгоритма БПФ опять вернемся к китайской теореме и вспомним, что любое число может быть записано с помощью остатков от деления на взаимно простые модули $M_1, M_2, ..., M_l$ в виде $r_1, r_2, ..., r_l$. Такую запись можно рассматривать как представление числа в некоторой системе счисления по нескольким основаниям $M_1, M_2, ..., M_l$. Она называется китайской. Заметим, что в отличие от обычных систем с одним основанием (двоичной, десятичной и т. д.) китайская система является непозиционной.

Если два числа $A = (a_1, a_2, ..., a_l), B = (b_1, b_2, ..., b_l)$ записаны в китайской системе, то для суммы и произведения этих чисел справедливы равенства:

$$A + B = (\langle a_1 + b_1 \rangle_{M_1}, \langle a_2 + b_2 \rangle_{M_2}, ..., \langle a_l + b_l \rangle_{M_l}); \quad (4.37)$$

$$AB = (\langle a_1 b_1 \rangle_{M_1}, \langle a_2 b_2 \rangle_{M_2}, ..., \langle a_l b_l \rangle_{M_l}).$$

т. е. операции над числами сводятся к операциям над отдельными разрядами по соответствующим модулям. Арифметика такого рода называется *модулярной*.

Вернемся теперь к преобразованию Фурье длины $N = N_1 N_2$, $(N_1, N_2) = 1$:

$$f(k) = \sum_{n=0}^{TV-1} s(n) W^{**}; \quad k = O,l,...,N\sim l.$$

Запишем показатель *кп* в китайской системе счисления, воспользовавшись равенствами (4.37):

$$(kn)_{кит} = ((k_1, k_2)(n_1, n_2)) = (< k_1 n_1 >_{N_1}, < k_2 n_2 >_{N_2}).$$

Переход к обычному представлению осуществляется на основании выражения (4.8):

$$кп = \{ T < к\ п >_N + \Gamma < к_о п >_м \} mod N,$$
$$N_l \quad * \quad {}^{l\ l}{}^{N}i \quad N_2{}^{2} \quad 2\ з\ л_2$$

поэтому

$$f(k) = f(\quad N4 \quad \{'^{\wedge}T i^{<k}\Gamma i^{>N} i^{+} -_N{}^{<k} 2'' 2^{>} N_2 r^{x}$$
$$\wedge Д_2) = (n_1, n_2) = 0 \quad sQi\wedge nJW^{\ l}$$

Поскольку функция W^{j} периодична с периодом TV, то приведение по модулю в показателе не требуется. Обозначим

$$W_1 = W^{\frac{N}{z}} = \exp(-/-); \quad W_. = W^{\frac{N}{l}} = \exp(-/\sim).$$

Тогда

$$f(k_1, k_2) = \sum_{(n_1, n_2)=0}^{N-f} K^{n_1 - n_2} W_1^{T_1 <k_1 n_1> N_1} W_2^{T_2 <k_2 n_2> N_2}.$$

Функции И^ и Н^ периодичны с периодами JVJ nN_2 соответственно. Поэтому вьиеты в показателях можно заменить самими числами, т. е. $<к_1 n_1> = к_л n_1$ и $< £_2 «_2 >_{дг} = Д_2 «_2$. После этого получим

$$\sum_{«_2=o}^{N_2-1} {}^{2} \sum_{i=o}^{T_2 k_2 n_2 N_1 - 1} {}^{J\ 2} W_1^{T_1 k_1 n_1} \qquad (4.38)$$

$$= \sum_{1}^{iV-I} {}^{J\ J} \sum_{»}^{N'-i} {}^{2\ 2\ 2} 1(и,,и,) Й' \frac{T.к.п,}{2} \frac{ТЛ.п.}{2}.$$

Это простое двухмерное преобразование Фурье, в котором вместо W_1, W_2 следует писать $W_1^{m_1}$ и $W_2^{m_2}$ соответственно.

Таким образом, гнездовой алгоритм сводит вычисление одномерного преобразования к вычислению двухмерного, если строки и столбцы матрицы преобразования и исходных данных переставить по китайской системе счисления.

Замену W_1 на W_1^{-} и W_2 на W_2^{-} можно и не делать, если ввести различные

перестановки для строк и столбцов матрицы преобразования. Сохраним прежний (китайский) порядок следования столбцов, а для нумерации строк введем новые переменные:

$$7,1 \quad \_ \quad T \; 7. \quad . \quad 7,' \quad \_ \; _{T} \; Д. \qquad (Л\ 3Q4$$

Тогда

$$f(k'_1, k'_2) = \sum_{n_2=0}^{N_2-1} W_2^{k'_2 n_2} \sum_{n_1=0}^{N_1-1} s(n_1, n_2) W_1^{k'_1 n_1} =$$

$$= \sum_{n_1=0}^{N-\backslash} W_1^{\,\cdot\,\cdot} \sum_{n_2=0}^{N_2-1} s(n_1 n_2) W_2^{k'_2 n_2} . \qquad (4.40)$$

Это обычное двухмерное ДПФ. Его вычисление происходит в три этапа:

1) производится перестановка строк и столбцов матрицы преобразования и отсчетов последовательности (S(H)J В соответствии С китайской теоремой И равенствами (4.39);

2) выполняется внутреннее преобразование, соответствующее внутренней сумме выражений (4.38), (4.40);

3) результаты внутреннего преобразования рассматриваются как векторы, над которыми выполняется внешнее преобразование, соответствующее внешней сумме выражений (4.38), (4.40).

Ясно, что если N_1 и N_2 в свою очередь также раскладываются в произведение взаимно простых сомножителей, то дробление на малоточечные преобразования можно продолжить, увеличив при этом число измерений.

Пример 4.14. Пусть $N = N_1 N_2 = 3 \cdot 4$. Отображение в китайскую систему счисления имеет вид:

0 - (0,0); 1 - (1,1); 2 - (2,2); 3 - (0,3); 4 - (1,0);
5 - (2,1); 6 - (0,2); 7 - (1,3); 8 - (2,0); 9 - (0,1);
10 - (1,2); 11 - (2,3),

что дает следующее размещение столбцов исходной матрицы и отсчетов последовательности {«(")}: 0,9,6,3,4,1,10,7,8,5,2,11.

Обратные элементы для чисел $N/N_1 = 4$ и $N/N_2 = 3$ равны: $T^{\wedge} = \_!$, $T_{\%} = 3$, что дает $\kappa'_1 = kj$, $\kappa'_2 = 3fc$, mod4 со следующим порядком следования строк: 0, 3, 6, 9, 4, 7, 10, 1, 8, 11, 2, 5.

Обозначим

$$^{S}(o) = \begin{bmatrix} j(0) \\ s(9) \\ s(6) \\ s(3) \end{bmatrix}; \quad \mathbf{S_1} = \begin{bmatrix} J(4) \\ s(1) \\ i(10) \\ s(7) \end{bmatrix}; \quad \mathbf{S_2} = \begin{bmatrix} s(8) \\ s(5) \\ s(2) \\ s(11) \end{bmatrix}$$

для входных отсчетов и

$$F_0 = \begin{bmatrix} /(0) \\ /(3) \\ /(6) \\ /(9) \end{bmatrix}; \quad F_{1^J} = \begin{bmatrix} \sim/(4)\sim \\ /(7) \\ /(10) \\ /(1) \end{bmatrix}; \quad \mathbf{F_2} = \begin{bmatrix} /(8) \\ /(1) \\ /(2) \\ /(5) \end{bmatrix}$$

для выходных отсчетов. Введем также матрицы ДПФ порядков Л^ = 3 и Л^ = 4:

$$\mathbf{V}_1 = \begin{bmatrix} w^0_1 & w^0_1 & w^0_1 \\ W^0_1 & & w^2_1 \\ w^0_1 & {}^{2}_{w\,1} & w^1_1 \end{bmatrix}; \quad \mathbf{V}_2 = \begin{bmatrix} w^{\wedge}_2 & W^0_2 & W^0_2 & W^0_2 \\ & W^1_2 & \boldsymbol{W}^2_2 & W^3_2 \\ w4^1_2 & W^2_2 & W^{\circ}_2 & W^2_2 \\ W^0_2 & w^3_2 & w^2_2 & {}^{1}_{W2} \end{bmatrix}.$$

Используя эти матрицы и новое размещение строк и столбцов, можно записать:

$$\begin{bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} = \begin{bmatrix} W^0_1\mathbf{V}_2 & W^0_1\mathbf{V}_2 & W^0_1\mathbf{V}_2 \\ W^0_1\mathbf{V}_2 & w\backslash\nu_2 & W^2_1\mathbf{V}_2 \\ W^0_1\mathbf{V}_2 & W^2_1\mathbf{V}_2 & W^1_1\mathbf{V}_2 \end{bmatrix} \begin{bmatrix} \mathbf{S}_0 \\ \mathbf{S}_1 \\ \mathbf{S}_2 \end{bmatrix} = (\mathbf{V}_1 \times \mathbf{V}_2) \begin{bmatrix} \mathbf{S}_0 \\ \mathbf{S}_1 \\ \mathbf{S}_2 \end{bmatrix}. \qquad \textbf{(4.41)}$$

Выражение $\mathbf{V}_1 \times \mathbf{V}_2$ называется *прямым* или *кронекеровым произведением*.

Таким образом, для получения ДПФ сначала вычисляются три четырехточечных ДПФ:

$$f_0 = \mathbf{V}_2\mathbf{S}_0, \quad f_1 = \mathbf{V}_2\mathbf{S}_1, \quad f_2 = \mathbf{V}_2\mathbf{S},$$

затем тройка [f_0 f_1 f_2} преобразуется при помощи трехточечного ДПФ:

$$\begin{bmatrix} \mathbf{F}_0 \\ \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix} = \begin{bmatrix} W^0_1 & W^0_1 & W^0_1 \\ W^0_1 & W^1_1 & \boldsymbol{K} \\ {}_{w}\mathrm{o}^{\wedge}_1 & W^2_1 & \mathbf{W} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix},$$

где

$$\mathrm{w}, \quad = \quad [w_I \;, w_I . w_I, w_I \;\mathrm{i}.$$

Возможен и другой вариант. Сначала вычисляются четыре трехточечных преобразования, а затем векторное четырехточечное.

4.9.2. Вычисление коротких преобразований

Короткие ДПФ путем перестановок строк и столбцов можно свести к циклическим сверткам, после чего воспользоваться алгоритмами для вычисления коротких сверток из § 4.6.

Пример 4.15. Пусть $N = 5$. Матрица ДПФ имеет вид

$$\begin{bmatrix} W^{\circ} & W^{\circ} & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 \\ w^{\circ} & w^2 & \mathbf{w}^4 & w^{\hat{1}} & w^{\hat{3}} \\ w^0 & w^3 & w^{\hat{1}} & & w^2 \\ w^0 & \boldsymbol{w}^* & w^3 & w^2 & w^1 \end{bmatrix}.$$

Переставим строки и столбцы по закону

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 2 & 4 & 3 \end{pmatrix}.$$

т. е. нулевая строка (столбец) остается на месте, на первое место ставится вторая строка и т. д. Чтобы результат умножения не изменился, в таком же порядке следует переставить отсчеты исходного вектора и спектральные коэффициенты. Тогда получим

$$\begin{bmatrix} /(0) \\ /(2) \\ /(4) \\ /(3) \\ /d) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 \\ W^0 & W^2 & W^4 & W^1 & W^3 \\ W^0 & W^3 & W^1 & W^4 & W^2 \\ W^0 & W^4 & W^3 & W^2 & W^1 \end{bmatrix} \begin{bmatrix} s(0) \\ s(2) \\ s(4) \\ s(3) \\ s(1) \end{bmatrix}.$$

Отсюда ввдно, что

$$/(0) = \sum_{n=0}^{4} s(n),$$

а вычисление остальных коэффициентов сводится к вычислению свертки

$$\begin{bmatrix} /(2) \\ /(4) \\ /(3) \\ /(1) \end{bmatrix} - \begin{bmatrix} s(0) \\ s(0) \\ s(0) \\ s(0) \end{bmatrix} = \begin{bmatrix} W^4 & W^3 & W^1 & W^2 \\ W^3 & W^4 & W^2 & W^4 \\ W^1 & W^2 & W^4 & W^3 \\ W^2 & W^4 & W^3 & W^1 \end{bmatrix} \begin{bmatrix} s(2) \\ s(4) \\ s(3) \\ s(1) \end{bmatrix}.$$

Поскольку коэффвдиентами свертки являются поворачивающие мнонители ДПФ, обладающие специфическими свойствами, то в вычислениях возможны упрощения. В частности, можно показать, что при использовании сверточных полиномиальных алгоритмов (см. § 4,6) все умножения либо чисто действительные, либо чисто-мнимые. Это дает существенное сокращение числа умножений.

С учетом указанных упрощений построены [10,11] оптимальные алгоритмы для вычисления малоточечных ДПФ для $N = 2, 3, 4, 5, 7, 8, 9$ т^6. Характеристики алгоритмов приведены в табл. 4.6.

Таблица 4.6

N	Число операций		
	умножения	умножения на W^{p}	сложения
2	0	2	2
3	2	1	6
4	0	4	8
5	5	1	17
7	8	1	36
8	2	6	26
9	10	1	45
16	10	8	74

Гнездование этих алгоритмов позволяет вычислить ДПФ длинных последовательностей.

4.9.3. Эффективность и общая структура алгоритма Винограда

Пусть $JV = N_1 N_2 \ldots N_d$, где $(N_i, Л^r) = 1$. Вычисление ДПФ сводится к вычислению малоточечных преобразований размеров N_i, $i = 1, 2, \ldots, d$. Предположим, что на получение малоточечного преобразования размером N_i затрачивается M_i комплексных умножений и A_i комплексных сложений. Тогда общее число умножений и сложений равно соответственно

$$M = M_1 M_2 \ldots M_d$$

$$A = A_1 N_2 \ldots N_d + M_1 A_2 N_3 \ldots N_d + \ldots + M_1 M_2 \ldots M_{d-1} A_d .$$

Заметим, что число сложений зависит от порядка выполнения операций. Например, для $N = N_1 N_2$ возможны два варианта с числом сложений $N_1 A_2 + M_2 A_1$ и $N_2 A_1 + M_1 A_2$.

Выражения (4.42) позволяют определить суммарное число комплексных умножений и сложений, в предположении, что входные данные комплексные. Однако если короткие ДПФ вычисляются по оптимальным алгоритмам, то комплексные умножения сводятся к умножению числа на вещественное или чисто мнимое число и выполняются посредством двух вещественных умножений. Кроме того, часть из них тривиальна (умножения на $\pm 1, \pm j$).

Рассмотрим теперь общую структуру алгоритмов Винограда. Для двух сомножителей было получено выражение (4.41). Запишем его в более общем и компактном виде:

$$F = (V_1 \times V_2) S, \qquad (4.43)$$

где S - вектор отсчетов сигнала; V_1, V_2 — матрицы ДПФ.

Путем перестановки строк и столбцов матриц V_1 и V_2 можно преобразовать к матрицам циклических сверток. Обозначим последние D_1 и D_2. Тогда, пренебрегая порядком следования входных и выходных отсчетов, для (4.43) получаем

$$F = (D_1 \times D_2) S . \qquad (4.44)$$

Для вычисления сверток используем алгоритм (4.36):

$$Y = C(BH \circledast AX), \qquad (4.45)$$

где X - входной сигнал свертки. Выражение (4.45) можно записать и в другом виде, заменив поточечное произведение векторов BH и AX умножением вектора AS на диагональную матрицу L, по главной диагонали которой записаны элементы вектора BH:

$$Y = CLAX.$$

Отсюда следует, что матрица свертки D может быть представлена в виде

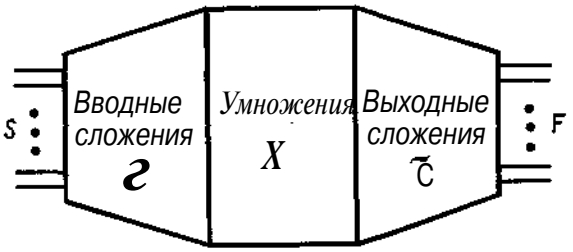


Рис. 4.5. Общая структура алгоритма Винограда.

$$D = CLA . \qquad (4.46)$$

Подставив (4.46) в (4.44), получим

$$F=((C_1L,A_1)\ X(C_2L_2A_2))S.$$

Используя свойства кронекерова произведения, преобразуем последнее выражение к виду

$$\Gamma = (C,XC_2XЦ \times L_2)(A_1 \times A_2)S = \tilde{C}\tilde{L}\tilde{A}S. \qquad (4.47)$$

Проделанные действия можно повторить для произвольного числа сомножителей, поэтому выражение (4.47) дает общую структуру алгоритма Винограда, совпадающую со структурой сверточного преобразования. В частности, из (4.47) следует, что для вычисления ДПФ алгоритмом Винограда следует выполнить сложения исходных данных по правилам, задаваемым матрицей $\tilde{A}$, затем полученный вектор поточечно умножить на диагональные элементы матрицы $\tilde{L}$. Результаты этого умножения следует сложить по правилам, задаваемым матрицей $\tilde{C}$. Структура алгоритма Винограда показана на рис. 4.5. Порядок матрицы X может быть больше величины N, что отражено на рисунке соответствующим увеличением размера средней части преобразования. Если порядок $\tilde{L}$ равен N, то алгоритм дает минимальное число умножений. Такие алгоритмы, как видно из табл. 4.6, известны для $N = 2, 3, 4, 8$.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАЧИ

4.1. В чем разница между линейной и циклической свертками? Приведите примеры использования этих понятий в технических приложениях.

4.2. Как осуществляется приведение по модулю полинома $z^N - 1$?

4.3. Чем отличаются линейный и диадный сдвиги функции?

4.4. Вычислите корреляционную функцию последовательности (1, —1, 1,1) за 8 операций сложений (вычитания).

4.5. Можно ли вычислить линейную свертку с помощью преобразования Адамара?

4.6. Какой размер должно иметь преобразование Фурье для вычисления апериодической корреляционной функции двух последовательностей длины 5 и 7?

4.7. В чем преимущества вычисления свертки с помощью ортогональных преобразований?

4.8. Сопоставьте преимущества и недостатки известных Вам ортогональных преобразований с точки зрения использования их для вычитания сверток.

4.9. Найдите НОД чисел 65 и 104, используя алгоритм Евклида.

4.10. Решите диофантово уравнение 65x+ 104y = 143.

4.11. В чем преимущества полиномиального описания свертки?

4.12. С помощью китайской теоремы восстановите число по его остаткам л^ 5mod8, r_2 = 7mod9, r_3 = 5mod11.

4.13. В чем заключаются трудности использования китайской теоремы для вычисления длинных сверток?

4.14. В чем состоит сущность гнездовых алгоритмов? Как они связаны с многомерными преобразованиями?

4.15. Запишите числа 121 и 135 в китайской системе счисления с основаниями 8, 9, 11.

4.16. Синтезируйте алгоритм четырехточечной свертки с помощью китайской теоремы.

4.17. Синтезируйте алгоритм Винограда для вычисления ДПФ длины N= 6 .

4.18. Проведите аналогию между ДПФ и вычислением полинома в точках.

5. РЕАЛИЗАЦИЯ БЫСТРЫХ АЛГОРИТМОВ ЦИФРОВОЙ ОБРАБОТКИ В РАДИОТЕХНИЧЕСКИХ СИСТЕМАХ

5.1. СТРУКТУРА СИСТЕМЫ И РЕАЛИЗАЦИЯ АЛГОРИТМА

Микропроцессорная система обработки сигналов может быть реализована различными способами. Наиболее простой в конструктивном отношении является однопроцессорная система (рис. 5.1). Она содержит центральный процессор, выполняющий арифметические и логические операции, память, устройство ввода-вывода и блок управления. Работа однопроцессорной системы заключается в последовательном выполнении определенных операций над данными, хранящимися в памяти или поступающими по каналу ввода-вывода. Команды, управляющие ходом вычислительного процесса, хранятся в памяти и расшифровываются блоком управления, который координирует работу всех узлов системы.

Независимо от конкретного назначения общим принципом описанной структуры является принцип последовательного выполнения отдельных команд, поэтому такие устройства называются *последовательными.* При их использовании алгоритм представляется в виде цепочки отдельных операций и проектировщику следует в первую очередь заботиться о минимизации числа таких операций и требуемого объема памяти. Скорость обработки определяется этими факторами и быстродействием элементной базы. Очевидно, что количество операций при решении любой задачи не может быть меньше некоторой константы, а возможности увеличения скорости переключения логических элементов ограничены существующим уровнем технологии. Поэтому классическая однопроцессорная структура во многих случаях Не обеспечивает требуемой вычислительной мощности.

Скорость обработки информации можно повысить, если выполнять трудо-

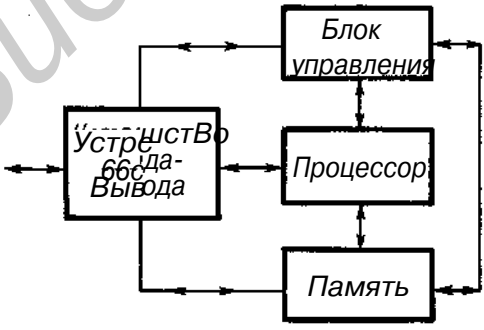


Рис. 5.1. **Однопроцессорная система.**

емкие операции отдельными специализированными процессорами или применять для решения задачи несколько процессоров, объединив их в единую систему каналами связи. Системы такого рода называются *мультипроцессорными.* Увеличение вычислительной мощности при использовании мультипроцессорной системы произойдет, однако, лишь в случае, если все процессоры или, большая их часть будут постоянно загружены. Структура такого рода называется *параллельной.* При такой структуре алгоритм решения задачи разбивается на ряд независимых частей, которые можно выполнять одновременно. Таким образом, необходимо довольно жесткое согласование численных методов с аппаратными средствами. В противном случае требуемая эффективность может быть не достигнута.

Для реализации в параллельной вычислительной структуре алгоритм представляется в виде последовательности групп операций с независимыми операциями в каждой. Такая форма представления называется *параллельной формой алгоритма.* Каждая группа операций называется *ярусом,* число групп — *высотой,* а максимальное число операций в ярусе — *шириной* параллельной формы.

Например, пусть требуется вычислить выражение

$$(x_1x_2 + x_3x_4)\ (x_5x_6 + x_7x_8)\ .$$

В параллельной форме алгоритм вычисления этого выражения можно записать так:

данные: jc_t, x_2, x_y x_4, x_s, x_6, x_v x_g;

ярус 1: XjX_2, x_3x_4, x_sx_b, $X_7{}^*{}_8$;

ярус 2: $x_yx_2 + x_3x_4$, $x_sx_6 + x_?x_8$;

ярус 3: $(x_Ix_2 + x_3x_4)(x_5x_6 + x_?x_g)$.

Высота параллельной формы равна трем, ширина — четырем. Для реализации требуется четыре процессора, способных выполнять операции сложения и умножения, причем все процессоры загружены только на первом шаге вычислений. На втором шаге работают только два процессора, а на третьем шаге — лишь один.

Оценим максимальное быстродействие параллельных алгоритмов. Для этого предположим, что каждый из процессоров параллельной структуры может выполнять бинарные или унарные операции и общее количество входных переменных и констант равно *N.* Тогда один процессор сможет выполнить все вычисления за *N—1* шагов. Если же организовать вычисления в виде дерева, как это было сделано в предыдущем примере, го время вычислений сокращается доГlo{», ЛП шагов. Древовидная структура, хотя и является самой быстродействующей, неэффективна с точки зрения количества используемых процессоров и их загрузки. В практических приложениях число процессоров, как правило, ограничено, поэтому больший интерес представляет задача максимального распараллеливания при фиксированном числе процессоров, а также задачи распараллеливания ряда частных, но широкоупотребительных формул (рекуррентных соотношений, степеней, полиномов и т. д.). Исследования по этим вопросам можно найти в [3, 13].

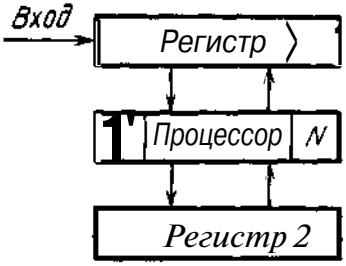


Рис. 5.2. Параллельный процессор БПА.

Разновидностью параллельной структуры является *конвейерный вычислитель.* Он состоит из последовательно соединенных специализированных процессоров, каждый из которых выполняет определенную часть некоторой сложной операции. Для того чтобы начать выполнение следующей операции, у предыдущей операции должен быть закончен только первый этап. Реализация конвейера позволяет существенно повысить скорость обработки информации, однако поток данных и последовательность их преобразований следует организовать так, чтобы стало возможным независимое выполнение ряда однотипных преобразований. Здесь снова требуется согласование численных методов с особенностями структуры вычислительной системы. Необходимость предварительного упорядочения входных данных приводит к тому, что собственно обработка отделена от процесса поиска данных в памяти, а для хранения промежуточных результатов используется сверхоперативная память.

В качестве примера для описанных структур рассмотрим реализацию быстрого преобразования Адамара (БПА).

Граф вычислительного процесса, БПА показан на рис 3.9. Последовательное устройство БПА реализуется по схеме, приведенной на рис. 5.1. Исходные данные записываются в память, затем производится их считывание, вычисление сумм, разностей и запись результатов в память. Вычисление БПА требует $\log_2 N$ шагов.

Параллельный вычислитель БПА показан на рис. 5.2. Входные данные записываются в регистр *1* , из которого поступают в параллельный процессор, состоящий изУУсумматоров-вычитателей. Вычисленныена первой итерации суммы и разности записываются в регистр *2* и являются исходными данными для второй итерации. Эти данные вновь подаются в параллельный процессор, а результаты второй итерации записываются в регистр / . Вычисления заканчиваются за *log^N* шагов.

Конвейерная схема для *N=8* показана на рис. 5.3. Она содержит три итеративные ступени. Каждая ступень состоит из двух блоков задержки *1* и *2* и арифметического устройства *3* . В качестве блоков задержки можно использовать многоразрядные регистры сдвига. В первой ступени каждый блок задержки содержит *N/2* элементов, а в каждой последующей ступени вдвое меньше, чем в предыдущей.

Рассмотрим работу процессора. С частотой тактовых импульсов компоненты сигнального вектора x (ṫ) поступают на вход первой ступени. Арифметическое устройство производит поочередно суммирование значений сигнала с выхода и входа первого блока задержки и вычитание с выхода и входа второ-

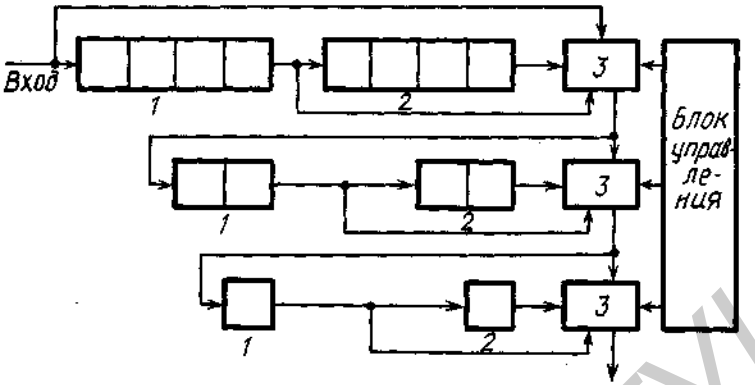


Рис. 5.3. Конвейерный процессор БПА.

го блока задержки. Таким образом, в первой ступени осуществляются вычисления, соответствующие первой итерации. Результаты вычислений поступают во вторую ступень, которая работает аналогично первой. Первый коэффициент преобразования появляется на выходе через TV тактов. В следующие *N—1* тактов на выход выдаются остальные коэффициенты.

В данном примере из-за характерных особенностей алгоритма БПА оказалось возможным использование любой аппаратной структуры. К сожалению, такая ситуация встречается на практике довольно редко. Как правило, преобразование исходного алгоритма к параллельной или конвейерной форме требует значительных интеллектуальных усилий и даже может оказаться невозможным.

Приведенные структуры, реализующие рассмотренные методы и алгоритмы, могут быть использованы при решении большого круга радиотехнических задач обработки сигналов в реальном масштабе времени. Сюда относятся: обнаружение и фильтрация сигналов, помехоустойчивое кодирование и декодирование, измерение координат путем измерения временного запаздывания и доплеровской частоты, классический и обобщенный спектральный анализ, трансформация спектров, передача и реконструкция изображений, специальные вычисления в задачах управления и диагностики и т. д.

Выбор той или иной структуры для реализации конкретного алгоритма определяется параметрами обрабатываемых сигналов и требованиями к конечным результатам. Они задают частоту дискретизации и число уровней квантования.

В радиотехнических системах указанные величины изменяются в очень широких пределах. Например, в командных и управляющих системах данные могут квантоваться с частотой от десятых долей герца до сотен герц. Доплеровский сдвиг частоты достигает десятков килогерц, а частота следования импульсов в высокоскоростных системах связи - десятков мегагерц. Число уровней квантования колеблется от двух до 2^{18}. Эти обстоятельства делают невозможным создание достаточно универсальных средств для цифровой обработки сигналов и вьшуждают инженера каждый раз приспосабливать существующую элементную базу для решения конкретной задачи.

В этой связи можно отметить три направления технической реализации рассмотренных алгоритмов - на основе универсальных микропроцессоров, на основе специализированных БИС с жесткой и полужесткой логикой и на основе однородных структур.

Среди универсальных микропроцессоров наибольшее применение для цифровой обработки сигналов получили однокристальные процессоры серий КР580, КР588, КР1810 и разрядно-секционированные процессоры К589, К1800, К1802, К1804, снабженные различными вспомогательными периферийными модулями.

Процессор КР58О имеет наименьшее быстродействие и используется для решения задач небольшой сложности (с малым количеством операций на один отсчет сигнала) и с частотой дискретизации до нескольких килогерц. Микропроцессор КР1810 является более совершенной модификацией процессора КР580 и позволяет на порядок повысить производительность устройства обработки. Секционированные процессоры используются для выполнения более трудоемких алгоритмов, например БПФ, вычисления сверток и т. д. С целью повышения скорости обработки они, как правило, совмещаются с быстродействующими аппаратными средствами: матричными умножителями, сдвигателями, буферными регистрами и т. д.

Поскольку рассмотренные алгоритмы весьма неоднородны по сложности, а диапазон обрабатываемых сигналов очень широк, то невозможно указать четкие границы применения того или иного микропроцессора.

Универсальные микропроцессоры в силу своих архитектурных особенностей часто не в состоянии обеспечить требуемого быстродействия, так как слабо учитывают такие особенности алгоритмов, как регулярность, независимость части операций, возможность конвейеризации и т. д. Поэтому в системах с жесткими требованиями к быстродействию описанные алгоритмы реализуются специализированными процессорами или вычислительными структурами с жесткой логикой. В ряду специализированных процессоров отметим микросхему для цифровой фильтрации К1815ВФ3, которая может быть использована для построения процессора БПФ с последовательным вводом-выводом данных, и периферийный процессор "Электроника МС1603". Последний работает совместно с управляющей ЭВМ и позволяет вычислять 1024-точечное БПФ с комплексными числами за 11 мс, функции корреляции и свертки той же длины — за 14 мс.

Вычислительные структуры с жесткой логикой фактически представляют собой аппаратное воплощение графа вычислительного процесса. Очевидно , что стоимость таких схем высока и их применение оправдано лишь в тех случаях, когда все другие возможности исчерпаны. Здесь широко используются высокоскоростные БИС для умножения, сдвига, хранения, аналого-цифрового ицифроаналогового преобразования сигналов.

Более подробные сведения об элементной базе для цифровой обработки сигналов и реализации ряда рассмотренных алгоритмов содержатся в [4, 13, 14].

Рассмотрим теперь третье "нетрадиционное" направление в области технических средств. Сущность этого направления состоит в том, что процессор строится из однородных, но достаточно сложных ячеек, соединенных между собой магистралями передачи информации. Система работает по принципу

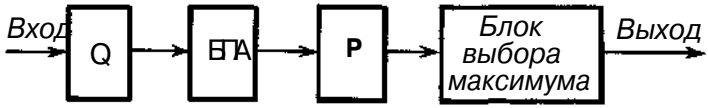


Рис. 5.4. Декодер с использованием БПА.

Перестановка столбцов (умножение на матрицу Q) эквивалентна перестановке отсчетов входного вектора. Перестановка строк (умножение на матрицу Р) эквивалентна перестановке отсчетов выходного вектора. Поэтому техническая реализация декодера содержит блоки перестановок входных и выходных данных, блок быстрого преобразования Адамара и блок выбора максимального значения (рис. 5.4).

Поскольку размерность матрицы Адамара на единицу больше размерности входного вектора, то для согласования размерностей длина входного вектора увеличивается на единицу путем введения нулевого символа на первой позиции. При анализе результатов вычислений блоком выбора максимального значения первая позиция выходного вектора не принимается во внимание. Число необходимых вычислительных операций равно $Mog_2 7V$.

Если слова кода пронумеровать в порядке возрастания первых κ символов, то необходимость в перестановке выходных данных отпадает.

Для реализации декодера необходимо определить явный вид перестановки, задаваемой матрицей Q . Вид перестановки определяется следующей теоремой.

Т е о р е м а 5.1. Пусть а — примитивный элемент поля Галуа $GF(2^k)$. Тогда подстановка

$$L: i \rightarrow a^{i+1}, \qquad (5.1)$$

примененная к столбцам матрицы S , переводит ее в матрицу Адамара порядка 2^k без первой строки и первого столбца.

С учетом теоремы 5.1 процесс декодирования сводится к следующему:

1) позиции вектора X переставляются в соответствии с выражением (5.1), после чего X дополняется слева одним нулевым символом. Обозначим результат этих операций X*;

2) выполняется умножение вектора X' на матрицу Адамара;

3) определяется максимальная компонента произведения.

П р и м е р 5.1. Пусть имеется последовательность максимальной длины 0, 0, 1, О, 1, 1, 1, описываемая полиномом $h(x) = x^3 + x + 1$. В алфавите 1, —1 она запишется так: 1, 1, —1,1, -1 , —1, —1. Матрица S кодовых слов, упорядоченных в соответствии с возрастанием первых κ символов, равна

$$S = \begin{bmatrix} 1 & 1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 & -1 & 1 \end{bmatrix}.$$

Нетрудно видеть, что первые три столбца этой матрицы образуют функции Радемахера R_1 , R_2 , R_3 без первого символа.

Элементы поля $GF(2^3)$ равны [21:a° = 100, a^1 = 010, a^2 = 001, a^3 = 110, a^4 = 011, a^5 = 111, a*= 101.

Подстановка L переставляет их следующим образом:

1) 001 -*a^0 = 100;

2) 010 -»a^1 = 010;

3) 011 -*a^2 = 001;

4) 100 -^a^3 = 110;

5) 101 ->a^4 = 011;

6) 110->-a^5 = 111;

7) 111-»a* = 101,

т.е. первая позиция переставляется на четвертую, вторая остается-на месте, третья становится на первую, четвертая на шестую и т. д. Запись в табличном виде выглядит так:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 1 & 6 & 3 & 7 & 5 & \end{pmatrix},$$

Применение этой подстановки к столбцам матрицы S дает матрицу

$$\begin{bmatrix} -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix},$$

отличающуюся от матрицы Адамара отсутствием первого столбца и первой строки.

Перестановка входных данных может быть выполнена устройством, показанным на рис. 5.5. Оно содержит запоминающее устройство (ЗУ) с произвольной выборкой, генератор поля Галуа и двоичный счетчик. При записи информации адресные входы ЗУ подключены к генератору поля Галуа, а при считывании - к счетчику. Поэтому первый символ сигнала записывается по адресу a^0 , второй - по адресу a^1 и т.д. После того как все символы будут приняты, производится считывание. Процессом считывания управляет двоичный счетчик. Первым считывается символ по адресу 1 (0...01). вторым - по адресу 2 (0...10) и т. д. При таком алгоритме входные данные переставляются в соответствии с перестановкой.

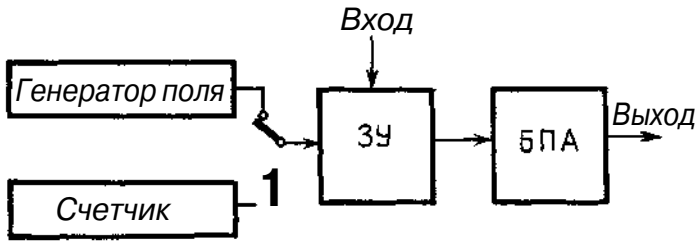


Рис. 5.5. Схема реализации перестановки.

4. Декодирование других кодов. Многие коды содержат код максимальной длины в качестве подкода и имеют кодовую матрицу следующего вида:

$$\begin{bmatrix} M \\ M\ \mathrm{diag}\, C_j \\ M\ \mathrm{diag}\, C_2 \\ \vdots \\ M\ \mathrm{diag}\, C_p \end{bmatrix}, \qquad (5.2)$$

где M - кодовая матрица кода максимальной длины; diagC$_j$ — диагональная матрица, т. е. матрица, по главной диагонали которой записана последовательность C$_j$, а остальные элементы равны нулю. Например, для кодов Голда последовательности C$_j$ также являются последовательностями максимальной длины.

Умножение вектора на матрицу (5.2) сводится к раздельному умножению на матрицы. M, MdiagC$_j$, $j = 1, 2, \ldots, p$. Умножение на diagC$_j$ заключается в смене знаков компонентов входного вектора в соответствии с последовательностью C$_j$. Умножение на *M* выполняется по рассмотренному алгоритму.

5. Синхронизация. Часто периодическая последовательность максимальной длины используется в качестве синхронизирующей для начального ввода системы в синхронизм. Задача синхронизации состоит в определении фазы приходящей последовательности, т. е. распознавания ее сдвига во времени относительно некоторого фиксированного состояния. Так как все циклические сдвиги являются словами кода максимальной длины, то задача синхронизации совпадает с задачей декодирования кода и решается с помощью описанных устройств.

5.2.2. Программная реализация

При программной реализации рассмотренных декодеров целесообразно использовать модульное программирование, при котором вся программа разбивается на отдельные функциональные модули или подпрограммы. Основными функциональными модулями являются: 1) модуль вычисления БПА; 2) модуль перестановки отсчетов в соответствии с выражением (5.1) ; 3) модуль умножения исходного вектора на последовательность C$_j$; 4) модуль выбора максимального значения.

Рассмотрим реализацию этих модулей на языке ассемблера в системе команд микропроцессора КР58ОИК8О в предположении, что длина обрабатываемого кода не превышает 255.

1. Модуль вычисления БПА. Для вычисления БПА используется граф, показанный на рис. 3.9. Его характерной особенностью является то, что результаты вычислений записываются в те же ячейки памяти, откуда берутся исходные данные. Для хранения адресов операндов выделим две регистровые пары B и D . Анализ графа показывает, что его можно разбить на каждой итерации на идентичные части. В частности, первая итерация содержит одну часть, вторая - две, третья — четыре и т. д. В общем случае число частей равно 2^{j-1}, где j — номер итерации. Основной базовой операцией каждой части является сло-

жение и вычитание пары отсчетов — операция "бабочка". "Крылья" "бабочки" уменьшаются по мере продвижения к концу вычислений. Поэтому основная часть алгоритма сводится к многократному повторению базовой операции, причем эти повторения легко организуются в циклы. Таких циклов три — итераций, групп, "бабочек" в группе. Для организации этих циклов необходимо иметь три счетчика.

Счетчик итераций (СЧ1) организуется с помощью регистра Н , в котором хранится только одна единица. Переход к следующей итерации соответствует сдвигу этой единицы вправо. Содержимое регистра Н задает также количество "бабочек" в группе. Поэтому счетчик "бабочек" (СЧ3) можно организовать с помощью команды DCR Н . При установке этого счетчика в нуль следует пе-

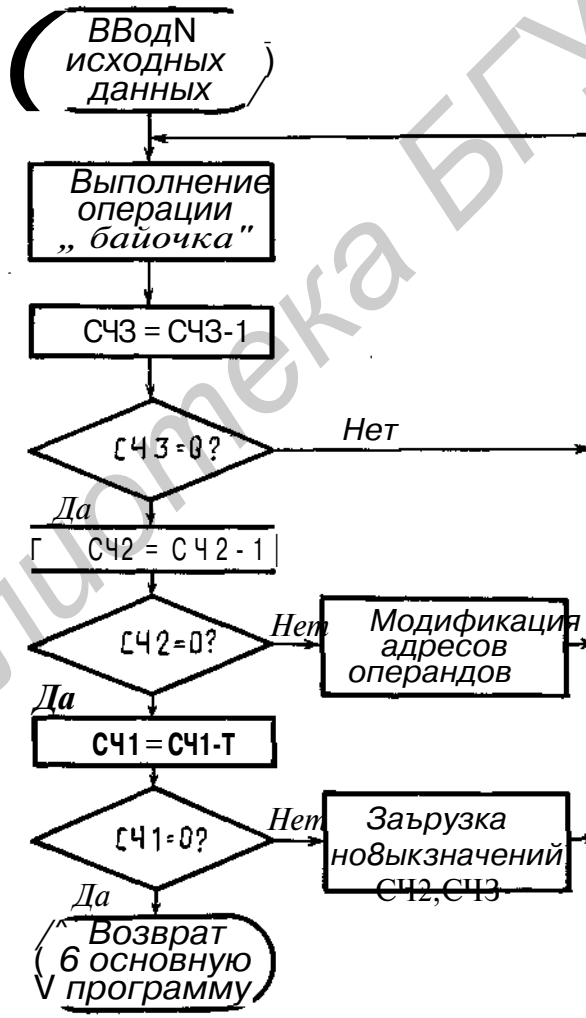


Рис. 5.6. Схема алгоритма БПА,

рейти к вьиислениям в следующей группе, для чего необходимо модифицировать адреса исходных данных. Счетчик числа групп в итерации обозначим СЧ2. Обнуление этого счетчика переводит процесс на следующую итерацию. Цикл счетчика СЧ1 является внешним. В него вложен цикл счетчика СЧ2, в который, в свою очередь, вложен цикл счетчика СЧ3. Схема алгоритма дана на рис. 5.6. Приведем текст программы:

```
M3:   ANA A          ;  Обнуление признака переноса
      LDA LENGTH     ;  Загрузка длины преобразования
      RAR            ;  Деление на два: СЧ1-= СЧ1-1
      STA LENGTH     ;  Запись в память
      JC M5          ;  Если СЧ1 = 0, то выход
      LXI B ADRES    ;  Загрузка адреса начала массива
      MOV D,B        ;
      MOV E,C        ;
      LDA LENGTH     ;
      MOV H,A        ;  } Получение адреса второго операнда
      ADD E          ;  } и организация СЧ1, СЧ3
      MOV E,A        ;
M2:   LDAX D         ;
      MOV L.A        ;
      LDAX B         ;
      ADD L          ;  } Выполнение базовой операции "бабочка"
      STAX B         ;
      SUB L          ;
      SUB L          ;
      STAX D         ;
      INX B          ;| Продвижение по массиву
      INX D          ;
      DCR H          ;  СЧ3 = СЧ3 - 1
      JNZ M2         ;  Цикл СЧ3
      LDA COUNT      ;
      DCR A          ;| СЧ2
      STA COUNT      ;
      JNZ M4         ;  Переход к блоку модификации адресов
      LDA MEM        ;
      RLC            ;  } Вычисление новых значений
      STA MEM        ;  } СЧ2, СЧ3
      STA COUNT      ;
      JMP M3         ;  Переход к новой итерации
M4:   LDA COUNT      ;
      MOV H,A        ;
      ADD C          ;
      MOV C,A        ;  } Блок модификации адреса
      MOV A, H       ;
      ADD E          ;
      MOV E, A       ;
      JMP M2         ;
M5:   RET
```

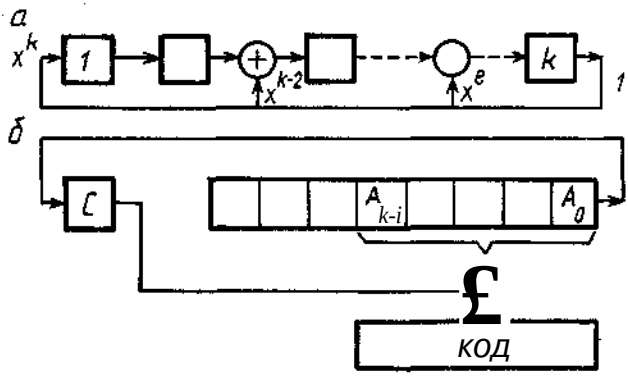


Рис. 5.7. Генератор ПМД и поля Галуа.

Вызывающая программа должна подготовить исходные данные: в ячейку памяти с номером LENGTH заносится длина преобразования *N,* а в ячейки памяти с номерами COUNT и MEM — единицы.

2. Генерирование последовательности максимальной длины и поля Галуа. Последовательность максимальной длины содержит $N \sim 2^k$ - \ символов и может быть получена с помощью регистра сдвига с *к* ячейками (рис. 5.7, *а).* Конфигурация обратных связей задается коэффициентами полинома $h'(x) = h(l/x)x^k$.

При программной реализации в качестве ячеек регистра сдвига можно использовать разряды аккумулятора (рис. 5.7, *б).* Петля обратной связи и сумматора моделируется путем суммирования по модулю два содержимого аккумулятора и константы Os , которая определяет вид обратной связи. Суммирование происходит при наличии признака переноса. Например, для полинома $h\backslash x) = x^s + x^3 + 1$ код обратной связи равен 0010100. Для организации сдвига и суммирования используются команды RAR и XRI 0s .

Вектор-состояние регистра сдвига можно рассматривать как элемент поля Галуа [2]. Таким образом, генерирование ПМД совпадает с генерированием элементов поля.

Приведем соответствующую программу:

```
      MVI  A, START   ;  Установка начального состояния
      ANA  A          ;  Сброс признака переноса
M1:   OUT  PORT       ;  Вывод состояния
      RAR             ;  Сдвиг содержимого регистра
      JNC  M1         ;  Переход к следующему состоянию, если сигнал обратной
                         связи равен нулю
      XRI  OS         ;  Вычисление состояния, если сигнал обратной связи не равен
                         нулю
      JMP  M1         ;  Переход к следующему состоянию
```

3. Модуль перестановки входных отсчетов. Пусть входные данные записаны в ячейках памяти с адресами MEM1 — (MEM1 + *N).* Для записи переставленных данных выделим ячейки с адресами MEM2 — (MEM2 + *N).* Эти массивы не должны перекрываться.

Для осуществления перестановки используем индексную адресацию. При этом исполнительный адрес определяется как сумма содержимого индексно-

го регистра и смещения. Смещением является состояние генератора поля Галуа.

Для организации индексного регистра используем регистровую пару В, в которую запишем число MEM2. Это число удобно выбрать таким, чтобы его младший байт был равен нулю. Тогда операция суммирования с индексом заменяется пересылкой смещения в регистровую пару.

Выделим регистр Н для хранения текущего состояния счетчика числа повторений, а пару В для хранения начального адреса MEM1 массива исходных данных. Для организации поля Галуа воспользуемся аккумулятором.

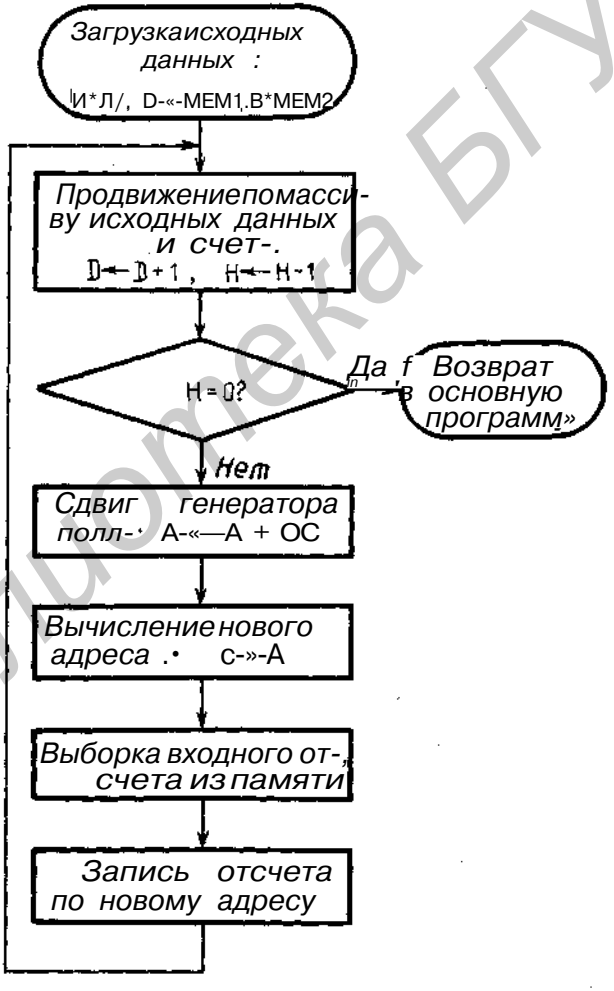


Рис. 5.8. Схема алгоритма перестановки отсчетов.

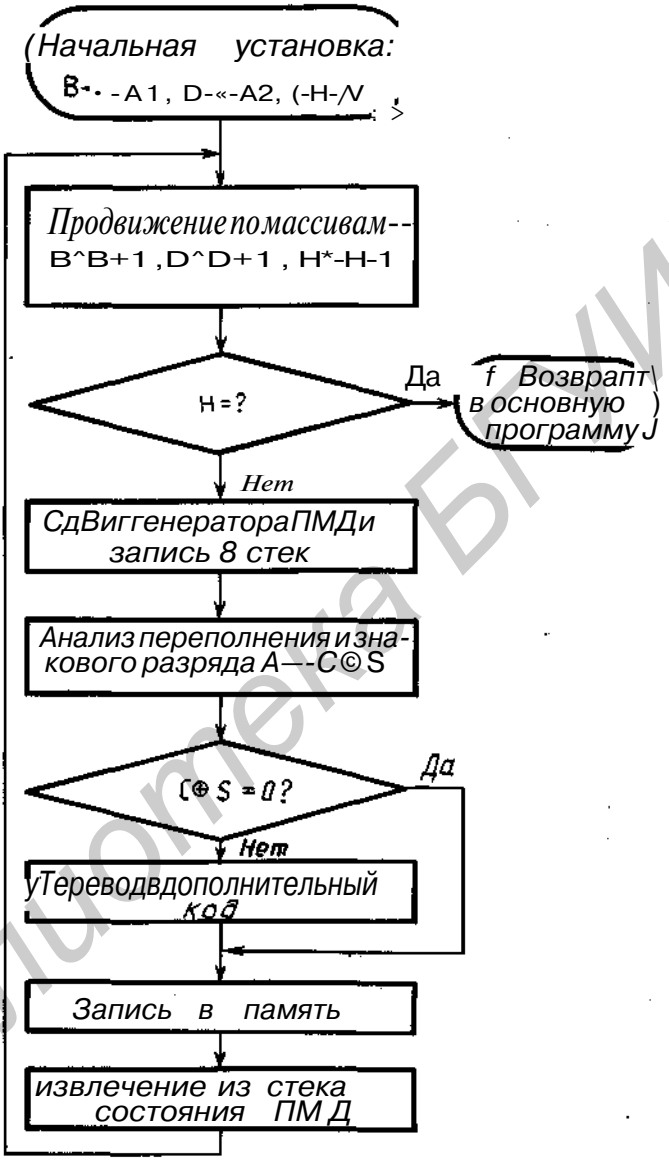


Рис. 5.9. Схема алгоритма умножения на С

Схема алгоритма перестановки показана на рис. 5.8. Текст программы имеет следующий вид:

```
MVI  H, N        ;  Загрузка счетчика
MVIA, START      ;  Загрузка аккумулятора начальным состоянием
                    генератора поля
```

```
        LXI  B, MEM2    ;  Загрузка  индексного регистра
        LXI D, MEM1     ;  Загрузка начала массива
M1:     INX-D           ;  Продвижение по массиву входных данных
        MOV C,A         ;  Получение нового адреса
        LDAX  D         ;  Извлечение входного отсчета
        STAX B          ;  Запоминание состояния поля
        DCR  H          ;  Счет
        JZ M3           ;  Возврат в основную программу, если H = O
        ANA A           ;
        RAR             ;
        JNC  Ml         ;  Генерирование поля
        XRI OS          ;
        JMP  Ml         ;
M3:     RET             ;  Возврат
```

4. Модуль умножения входного вектора на последовательность C_i . Для определенности будем считать, что выполняется декодирование кода Голда, так что C_i — последовательность максимальной длины, и программа должна генерировать эту последовательность. При каждом сдвиге генератора ПМД очередной входной отсчет извлекается из памяти. При этом изменяется его знаковый разряд, если признак переноса равен единице. В противном случае он переписывается в память без изменений. Будем считать, что входные данные содержат четыре разряда (один из которых знаковый) и представлены в прямом коде. Результат умножения будем записьшать в память в дополнительном коде. Для хранения исходного и полученного массивов выделим ячейки памяти с начальными адресами A1 и A2.

На рис. 5.9 показана схема алгоритма. Приведем текст программы:

```
        POPPSW          ;  Занесение из стека в А начального состояния генератора ПМД
        MVI H. N        ;  Регистр Н - счетчик длины слова
        LXI  B,A1       ;  В - регистр адреса входных данных
        LXI  D.A2       ;  D - регистр адреса результата перемножения
M4:     INX B           ;  } Продвижение по массивам
        INX D           ;  }
        ANA A           ;  Сброс признака переноса
        RAR             ;  Циклический сдвиг вправо содержимого генератора ПМД
        PUSH PSW        ;  Сохранение в стеке состояния генератора ПМД
        DCR  H          ;  Счет
        JZ M2           ;  Выход из подпрограммы, если Н = 0
        RAL             ;  } Четырехкратный сдвиг влево, цель которого — совместить
        RAL             ;  { бит признака переноса и бит знака входных данных
        RAL             ;  }
        RAL             ;  }
        MOV B, A        ;  A -> B
        LDAX  B         ;  Извлечение из памяти отсчета входных данных
        XRA  B          ;  A Ф B-+A
        ANI 08          ;  Выделение третьего разряда
        JZM5            ;  Если  S ® C =  0, то переход к пересылке в память
        LDAX  B         ;  Входной отсчет в А
```

```
        CMA          ;}
        INR A        ;} Перевод в дополнительный код
        JMP M3       ; Безусловный переход
M5:     LDAX B       ; Загрузка входного отсчета
M3:     STAX D       ; Запись в память результата
        POPPSW       ; Извлечение из стека состояния генератора ПМД
        JNC M4       ; Переход к новому такту, если отсутствует сигнал обратной
                     связи
        XRI OS       ; Суммирование с кодом обратной связи
        JMP M4       ; Переход к новому такту
M2:     RET          ; Возврат
```

5. Модуль определения максимума. Структуру этого модуля рассмотрим для случая, когда кодовая матрица имеет вид (5.2). Она разбивается на блоки размера N , поэтому для определения позиции максимума следует указать номер блока и номер строки в блоке.

Принцип работы программы основан на сравнении текущего элемента массива данных с текущим максимумом и фиксации текущего максимума. Величину максимума и номер его строки в блоке будем хранить в стеке и для оперативной работы вызывать в регистровую пару B. Для подсчета позиций в блоке в регистре D организуем СЧ1. Текущий номер блока будем подсчитывать в СЧ2, организованном в ячейке памяти COUNT2. Для хранения номера блока, в котором обнаружен максимум, воспользуемся ячейкой памяти с номером CYCL. Соответствующая программа имеет вид:

```
        LXI SP, STEK ; Установка указателя стека
        LXIH,DATA    ; Установка регистра адреса данных
        MVI D, N     ; Организация СЧ1
        LDA COUNT 2  ; Загрузка в аккумулятор содержимого СЧ2
        DCR A        ; Счет блоков
        JZOVT        ; Выход из подпрограммы, если (СЧ2) == Q
        STA COUNT 2  ; Запоминание номера текущего блока
        ЮРВ          ; Загрузка регистров- текущего максимума и номера
M1:     MOVA, M      ; Загрузка текущего элемента массива
        CMPB         ; Сравнение текущего элемента с текущим максимумом
        JCM2         ; Если B < A, то сохранение максимума
        MOVB,A       ; Замена текущего максимума
        MOVC, L      ; Замена номера текущего максимума
        LDA COUNT 2  ; Запоминание номера блока с максимумом
        STA CYCL     ;
M2:     INXH         ; ] Переход к
        DCRO         ; > следующей
        JNZM1        ; J позиции
        PUSH B       ; Запись в стек текущего максимума и его номера
        RET          ; Возвращение в основную программу
```

Время выполнения программ приведено в табл. 5.1.

Таблица 5.1

Длина N	Время выполнения, мс			
	вычисление БПА	перестановка	умножение на C_l	определение максимума
32	3,7	1,1	1,5	0,8
64	8,1	2,2	3	1,6
128	П,1	4,4	6	3,3
256	38,6	8,8	12	6,6

5.3. РАЗДЕЛЕНИЕ МАЖОРИТАРНО-УПЛОТНЕННЫХ СИГНАЛОВ ПРИ ПОМОЩИ ДИАДНОЙ СВЕРТКИ

Одним из способов передачи сообщений от нескольких источников по одному каналу является мажоритарное уплотнение [9]. Его структурная схема показана на рис. 5.10 и содержит генератор функций Уолша (ГФУ), коммутатор (К), перемножители и мажоритарный элемент (М). Генератор вырабатывает $N = 2^n$ дискретных функций Уолша, каждая из которых содержит $N = 2^n$ символов. С помощью коммутатора из этого множества функций выделяется только n функций. Номера выбираемых функций определяются стратегией работы системы и для дальнейшего несущественны. Выбранные n функций подаются на первые входы перемножителей. Вторые входы перемножителей подключены к источникам двоичных сообщений $b_i = \pm 1$. Таким образом, каждое из передаваемых сообщений модулируется какой-либо функцией Уолша. Выходные сигналы перемножителей поступают на мажоритарный элемент, в котором образуется единый групповой сигнал A (и). В аналитическом виде его можно записать следующим образом:

$$A(n) = \mathrm{Maj}(\hat{}_{-1}\mathrm{had}(/_0,\ll),\ b_{n-1}bzd(i_{,}\ \text{и}),\ldots.\ Z\text{»}_o\mathrm{had}(i_{\text{fl}-1}\ ,\ \text{и})).$$

Обозначим $B = (\text{й}_{3-1}, b_{n-2},\ldots, b_0)$, $I = (j_{n-x}, i_{n-2},\ldots, i_0)$. При фиксированном векторе I множество мажоритарно-уплотненных сигналов состоит из 2^n элементов. Каждому сигналу однозначно соответствует вектор B. Деко-

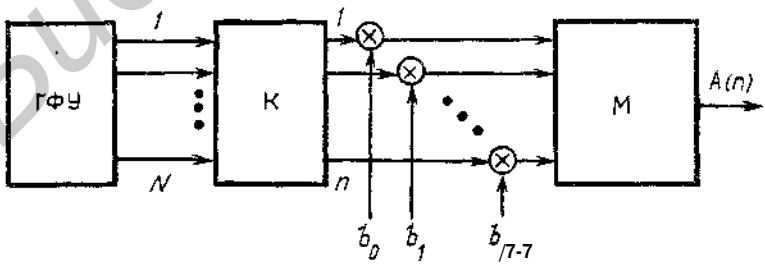


Рис. 5.10. Устройство мажоритарного уплотнения.

дарование (разделение) заключается в определении вектора В по принятому сигналу *А* (л).

Возможность использования при декодировании алгоритмов быстрого преобразования Адамара устанавливается следующей теоремой [9].

Т е о р е м а 5.2. При фиксированном векторе I множество последовательностей *А* («) инвариантно относительно диадного сдвига аргумента.

Из данной теоремы следует, что все множество мажоритарно-уплотненных сигналов получается путем диадных сдвигов единственного сигнала. Поэтому декодирование сводится к вычислению диадной| корреляционной функции (3.49), которую удобно вычислять при помощи двукратного преобразования Адамара (3.50), (3.51):

$$\mathbf{R_т = Л\hat{}ЩИХ\text{-}НА),} \qquad (5.3;$$

где **Н** — матрица Адамара; Х - вектор входного сигнала; А — вектор опорного сигнала.

П р и м е р 5.2. Пусть *N*= 8, а сигналы *А* (и) образуются мажоритарным уплотнением функций Радемахера.и их инверсий, т. е.

$$A\ (я) \quad = Uai(b_2R_1,\ \ b,R_2,\ \ b_QR,).$$

Та блица 5.2

А (я)	В	т	*А (n)*	В	т
1 1 1 -1 1 -1 -1 -1	1 1 1	0	1 -1 -1 -1 1 1 1	1-1 1 1	4
1 1 -1 1-1 1 -1 -1	1 1-1	1	-1 1 -1 -1 1 1-1	1-1 1-1	5
1 -1 1 1-1 -1 1 -1	1-1 1	2	-1 -2 1-1 1-1 1	1-1-1 1	6
-1 1 1 1-1 -1 -1 1	1-1-1	3	-1 -1 -1 1--1 1 1	1-1-1--1	7

В табл. 5.2 приведены значения *А (n)* при различных В и соответствующие им значения диадного сдвига.

Пусть в качестве опорного сигнала используется последовательность A_Q = = [1, 1, 1, -1, 1, —1, —1, —1], для которой т = 0, а на вход приемника поступает последовательность A_3 = [—1, 1, 1,1, —1, —1, —1,1], для которой т= 3. Матрица преобразования (матрица Адамара) равна

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

Умножая AJ и $\mathbf{A_в^T}$ на **Н ,** получаем **спектры опорного и приходящего сигналов:**

$$HA^{T}_{O} = [0,4,4,0,4,0,O,-4]^{\Gamma};$$

$$HA^{\wedge} = [0,-4,-4,0,4,0,0,-4]^{\wedge}.$$

Произведшие спектров равно

$$S = HA^{T}Q \cdot HA^{\wedge} = [0,-16,-16,0,16,0,0,16]^{m}.$$

Выполняя обратное преобразование, находим:

$$R_{r} = W^{l} HS = [0,0,0,8,-8,O,0,0]^{T}.$$

В последнем выражении максимальное значение имеет третий компонент, т.е. г = 3, поэтому В = [1, —1, -1].

Подсчитаем число операций, необходимых для декодирования. Из (5.3) следует, что для вычисления R_{r} надо три раза умножить вектор на матрицу Адамара и перемножить *N* чисел. Произведение НА может быть вычислено заранее, кроме того, известно [9] , что оно содержит только *NJ2* ненулевых компонент. Поэтому остается два умножения вектора на матрицу Н и умножение *N/2* чисел. Умножение вектора на матрицу Адамара выполняется за *Nlog^N* операций, поэтому окончательно получим $2Mog_{2}N + I\backslash^{r}l2$ операций. Это число можно несколько уменьшить, если учесть, что произведение спектров имеет только *Щ2* отличных от нуля компонентов.

Программная реализация состоит из модулей, описанных в § 5.2.

5.4) УСЕЧЕННЫЕ АЛГОРИТМЫ

Рассмотрим задачу вычисления векторно-матричного произведения

$$Y = AXr(y_{1},y_{2},\ldots,y_{N})^{T}.$$

Ранее было показано, что для многих матриц количество операций при вычислении Y можно уменьшить с величины $O(N^{2})$ до величины $O(Mog_{2}7V)$ и даже *O(N).* Тем не менее для больших *N* объем вычислений все еще велик. В то же время в ряде практических приложений не обязательно знать все компоненты вектора *У,* так как интерес представляет только номер максимального компонента. Таким образом, две отдельные процедуры — умножение вектора на матрицу и определение номера максимального компонента — желательно совместить в одну (см. § 5.2,5.3).

Другим примером может служить задача определения частоты синусоидального • сигнала. Пусть *s(t)* = sincof . С помощью схемы, показанной на рис. 5.11, преобразуем этот сигнал в ДЭФ на разностной частоте $£2 = ш - o_{1}$. Номер ДЭФ и, следовательно, частоту *О,* можно определить по максимальной компоненте произведения Y = VX , где X - вектор отсчетов комплексного сигнала *x(t) = sinSlt- jcosilt* ; V — матрица дискретного преобразования Фурье. Таким образом, задача определения частоты сводится к задаче нахождения максимальной компоненты ДПФ.

Совмещение векторно-матричного умножения с определением максимальной компоненты с принципиальной точки зрения возможно для любой матрицы. Проиллюстрируем это на матрице-циркулянте квадратично-вычетного ко-

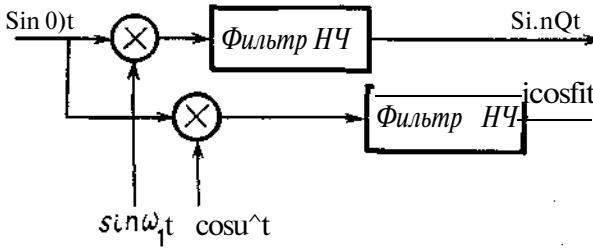


Рис. 5.11. Схема преобразования синусовдального сигнала в ДЭФ.

да из примера 4.4. Граф вычислительного процесса при умножении на эту матрицу показан на рис. 4.1.

Пусть на вход вычислителя поступает последовательность 1—111—1111—1—1—1. После выполнения первой итерации промежуточные суммы принимают значения: 0, ±2. Исключим в дальнейших вычислениях узлы первой итерации, в которых значения суммы равны 0. Аналогично на второй итерации сохраним только узлы с абсолютными значениями суммы больше 3 и т. д. Продолжив этот процесс дальше, получим граф с 19 узлами (рис. 5.12), т. е. с 19 операциями сложения-вычитания. Полное умножение вектора на матрицу по графу, показанному на рис. 4.1, требует 50 операций. Еще 10 операций необходимо затратить на поиск максимальной компоненты.

Здесь следует, однако, сделать следующие замечания. Во-первых, поиск и исключение узлов с малыми абсолютными значениями сумм требуют выполнения дополнительных операций сравнения и сортировки. Во-вторых, если сигнал принимается в смеси с шумом, то при принятии промежуточных решений возможны ошибки и, следовательно, помехоустойчивость усеченного алгоритма будет хуже, чем полного алгоритма. Ухудшение помехоустойчивости является следствием снижения вычислительных затрат.

Объем вычислительных затрат и помехоустойчивость алгоритма будут зависеть от конкретного вида матрицы A . Далее ограничимся рассмотрением матриц дискретных ортогональных преобразований, которые имеют регулярные графы вычислительного процесса.

Начнем с матриц Адамара. В этом случае задача определения максимальной компоненты произведения Y сводится к декодированию ортогонального кода (см. § 5.2). Этот код является подкодом биортотонального кода с кодовой матрицей $[H, -H]^T$, поэтому целесообразно сразу рассмотреть более общий случай декодирования биортогонального кода.

Основную идею усеченного адгоритма декодирования биортогонального кода легко понять из следующих рассуждений. В главе 3 показано, что строки матрицы Адамара порядка $N = 2^n$ можно трактовать как функции Уолша, заданные на интервале $(0, T]$. Каждая из функций Уолша, в свою очередь, является произведением не более чем n функций Радемахера $R_1(t), R_2(t), \ldots, R_n(t)$, представляющих собой меандры с периодами T, $2^{-1}\Gamma$, $2^{-2}T, \ldots, 2^{1-n}T$ соответственно. Двоичный номер функции Уолша показывает, какие именно функции Радемахера входят в произведение. Например, для $n = 3$ функция Уолша с номером 011 является произведением функций Радемахера /?,(/) и $R_3(t)$, т. е. had (Oil, t) $= R_2(t)R_3(t)$. Из этого следует, что задача деко'диро-

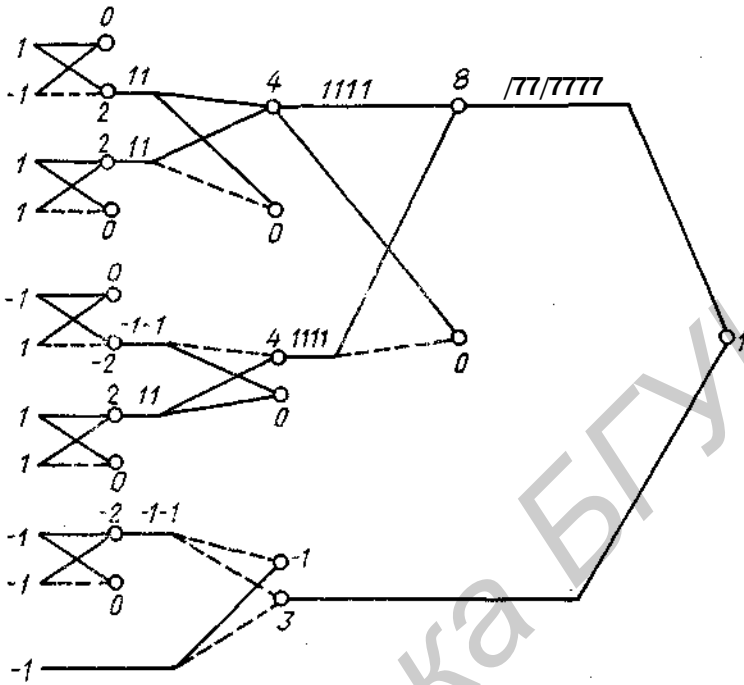


Рис. 5-12. Усеченный граф вычислительного процесса.

вания заключается в определении номера функции Уолша и знака этой функции. Последний необходим для того, чтобы указать место кодового слова в матрице $[H, -H]^m$,

Номер функции Уолша можно определить следующим образом. Если в формировании функции участвует функция Радемахера с номером и, то пары символов с номерами $2/$ и $2/ + I$ отличаются знаком, поэтому сумма символов в паре равна 0, а разность принимает значения ± 2. При отсутствии в произведении функции Радемахера с номером n сумма символов в паре равна ± 2, а разность 0. Просуммировав модули сумм и разностей всех пар и сравнив их, можно принять решение о первом индексе функции. Если этот символ равен единице, то в дальнейших вычислениях следует использовать только разности, в противном случае — только суммы.

Результаты первой итерации будем рассматривать как новый сигнал, полученный путем сжатия вдвое исходного сигнала. Поэтому для определения следующего символа можно использовать аналогичную процедуру и таким образом вычислить все символы.

Пусть $a_n a_{n-1} \cdots a_1$ — двоичный номер функции Уолша. Тогда в более строгом виде описанный алгоритм формулируется следующим образом [8]:

1) полагается $/ = N= 2'', /= 1$;

2) принятый вектор X размером i представляется в виде матрицы S раз-

мером 2 х (i'/2), первую строку которой составляют компоненты вектора с нечетными номерами, а вторую — с четными номерами;

3) вычисляется матрица S размером 2 х (//2) :

$$\tilde{S} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} S ;$$

4) элементы столбцов матрицы $\tilde{S}$, взятые по модулю, суммируются. Получается вектор-столбец S размером два;

5) в векторе S находится больший элемент. Полагается $d_j = 1$, если это первый элемент, и $a_j = 0$, если второй;

6) если / < *n* , то строка матрицы $\tilde{S}$ с номером, равным номеру максимального элемента S , переписывается на место первых г'/2 позиций вектора X . Полагается /= / + 1, *i* = //2 и следует переход к п. 2;

7) если / = *n* , то определяется номер максимального по модулю элемента S (в данном случае вектор $\tilde{S}$ имеет размер два). Знак этого элемента указывает, в какой половине матрицы [H, —H]** находится переданный вектор.

Пример 5.3. Пусть X= Г-1. 1.-1. 1,-1. 1.-1. 1] •

Первый шаг / = 1:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad \tilde{S} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -2 & -2 & 2 & 2 \end{bmatrix}^T = \begin{bmatrix} 0 \\ 8 \end{bmatrix}, \quad a_1 = 1.$$

Второй шаг / = 2:

$$S = \begin{bmatrix} -2 \\ 0 \end{bmatrix}; \quad \tilde{S} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}; \quad S = \begin{bmatrix} 2 \\ 0 \end{bmatrix}; \quad a_2 = 0.$$

Третий шаг / = 3:

$$S = \begin{bmatrix} -4 \\ 4 \end{bmatrix}; \quad \tilde{S} = \begin{bmatrix} 0 \\ -8 \end{bmatrix}^T; \quad S = \begin{bmatrix} 0 \\ 8 \end{bmatrix}; \quad a_3 = 1.$$

Отрицательный знак указывает на то, что переданный вектор находится во второй клетке матрицы [H, - H]**.

Таким образом, переданный вектор равен had(101,f) = $-R_1(t)R_3$ (f).

Алгоритм требует следующих вычислительных затрат: операций сложения — $AN - 2tog_2N + i$; операций сравнения — $\log_2 A^f + 1$; операций взятия модуля - $2N - 2$.

Описанный алгоритм можно обобщить на р-ичный случай [5] , когда матрицей кодовых слов является матрица функций Виленкина - Крестенсона (ВКФ). Эта матрица имеет порядок $N = p^n$, а ее элемент C_u(и), находящийся в *и-й* строке и *v-м* столбце (м , *v* = 0, 1,..., p^{*l} - 1), равен

$$C_u(v) = w^{i-v} ,$$

где $w = \exp(/2\text{»}г/р)$, / = $\sqrt{-1}$, "г-, и,.—коэффициентыр-ичнойзаписи чисел *и* и *v* .

Используя данное представление, запишем выражение для вычисления м-компонента вектора Y :

$$y(u) = \sum_{v=0}^{p^n-1} W^{uv} x(v) =$$

$$\sum_{v_0=0}^{p-1} \ldots \sum_{v_1=0}^{p-1} \ldots \sum_{v_{n-1}=0}^{p-1} \ldots \qquad (5.4)$$

На каждой итерации в результате последовательного вычисления сумм (5.4) цроизводится определение р-ичных разрядов м-го номера принятой ВКФ. При этом на первой итерации определяется старший разряд u_{n-1}, а на последней (и-й) — младший разряд $м_0$.

Действия на *к-й* итерации описьшаются соотношением

$$D_k = C^* B_k ,$$

где $C^* = [\, W^{lk}]$ —комплексно-сопряженная матрица ДПФ; B_k и D_k — матрицы, содержащие соответственно массивы входных и выходных данных для *к-й* итерации. В качестве B_1 используется принятый вектор X , первые p^{n-1} элементов которого служат первой строкой B_1, вторые p^{n-1} элементов — второй строкой и т.д. За оценку разряда принимается номер той строки, для которой максимальна сумма квадратов модулей стоящих в ней элементов.

При прямом умножении на матрицу С* алгоритм требует для реализации

$$\left(p^2 + \frac{p}{2}\right)\left(\frac{p^n - 1}{p-1}\right)$$

комплексных умножений и при $p \gg 1$ вьшгрывает у полного алгоритма приблизительно в *n* раз. Анализ помехоустойчивости показывает, что энергетический проигрыш усеченного алгоритма полному не превышает 3 дБ.

Построение усеченных алгоритмов БПФ рассмотрим на примере алгоритма с прореживанием по частоте. Граф вычислительного процесса этого алгоритма показан на рис. 3.5. Анализ вычислительного процесса позволяет установить следующие закономерности. Пусть $a_n a_{n-1} \ldots a_1$ — двоичный номер спектральной составляющей. Тогда в формировании величины $f(a_n, a_{n-1}, \ldots, a_1)$ участвуют на каждой итерации либо только суммарные, либо только разностные узлы графа. Суммарные узлы *i*-й итерации приводят к выходному узлу, двоичный номер которого содержит в *i*-й позиции 0, а разностные — к узлу, двоичный номер которого содержит в *i*-й позиции 1. Каждому выходному узлу соответствует свое дерево вычислений, в котором чередование суммарных и разностных узлов однозначно указывает номер спектральной составляющей.

Пусть $def(k, n) = (W^{k0}, W^{k1}, \ldots, W^{k(N-1)})$ - одна из базисных функций ДПФ. Для алгоритма с прореживанием по частоте в узлах первой итерации получим:

$$W^{kn} \pm W^{k(n+\frac{N}{2})} = W^{kn}(1 \pm W^{k\frac{N}{2}}) =$$

$$= \begin{cases} 2W^{kn} & \text{в суммарных узлах} \\ 0 & \text{в разностных узлах} \end{cases} \quad \text{% чегно е;}$$

$$\begin{cases} 0 & \text{в суммарных узлах} \\ 2W^{kn} & \text{в разностных узлах} \end{cases} \quad \text{£ нечетное,}$$

Отсюда следует, что сумма модулей выходных величин суммарных узлов равна N, а разностных 0, или наоборот.

Ненулевые значения в суммарных узлах будут только при четных & . Они равны $2W^{2kn} = 2(W^2)^{kn}$, $\kappa = 0,1,\ldots, N/2 \sim 1$. Ненулевые значения в разностных узлах будут только при нечетных κ . Умножение этих значений на поворачивающие множители дает величины $2W^{(2k+1)n}W^n = 2(W^2)^{kn}$, $\kappa = 0, 1,..., N/2 - 1$. Таким образом, на входы второй итерации поступают опять базисные функции ДПФ, но вдвое меньшего размера. Так как последующие вычисления являются ДПФ размера $N/2$, то все сказанное будет справедливо и для оставшейся части графа.

Приведенные рассуждения позволяют сформулировать следующий усеченный алгоритм вычислений: на r'-й итерации строится статистика, позволяющая определить принадлежность максимального отсчета выходного вектора множеству суммарных или разностных узлов. После этого половина узлов исключается. Выходные значения оставшихся узлов умножаются на поворачивающие множители, и следует переход к $/ + 1$ итерации.

На рис.-5.13 показан граф усеченного преобразования для $N = 8$. Аналогично можно усечь и БПФ по алгоритму Винограда. На рис. 5.14 показан пример такого усечения. Анализ этого примера показывает, что процедура принятия решения на каждой итерации в алгоритме Винограда может быть даже проще, чем в алгоритмах с прореживанием по времени или частоте.

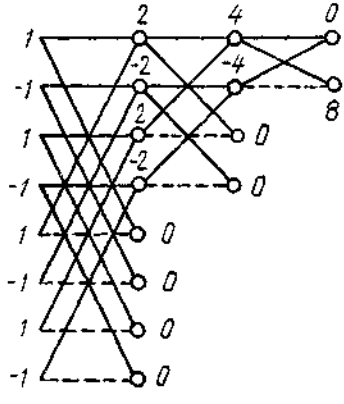


Рис. 5.13. Граф усеченного БПФ.

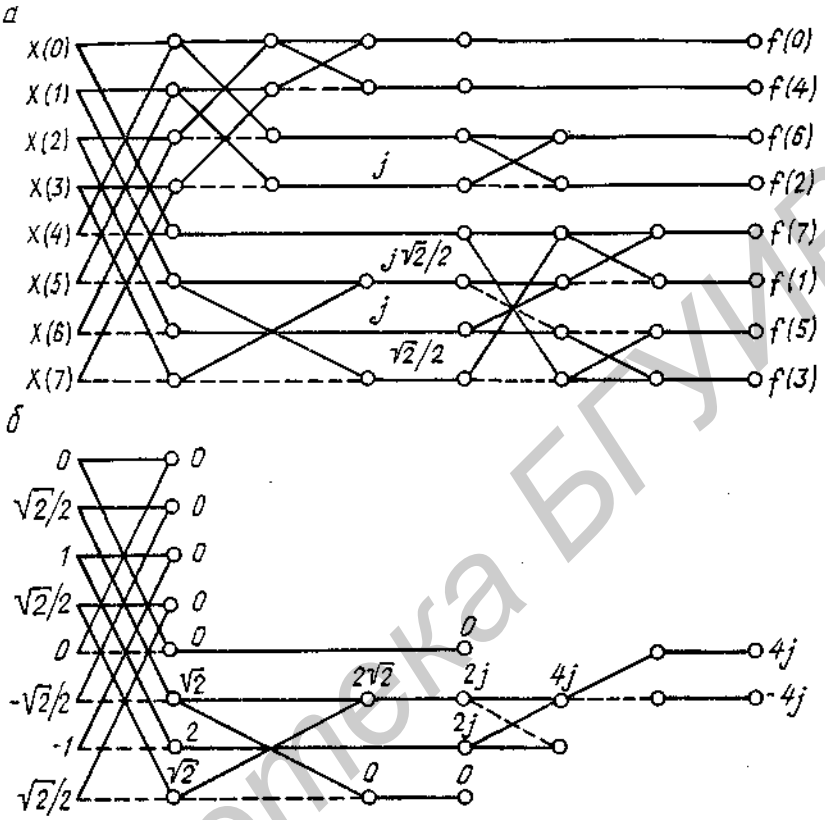


Рис. 5.14, Граф полного *(а)* и усеченного (б) БПФ по алгоритму Винограда.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАЧИ

5.1. Постройте структурную схему параллельного вычислителя, выполняющего умножение вектора на матрицу. Определите высоту и ширину параллельной формы алгоритма как функцию от размеров матрицы.

5.2. Для последовательности максимальной длины (1, —1, 1, 1, —1, —1, —l) постройie код максимальной длины. Преобразуйте матрицу кода в матрицу Адамара и запишите получившиеся при этом перестановки строк и столбцов.

5.3. При передаче сообщений кодом максимальной длины из п. 2 на приемном конце получены следующие последовательности: (-5, -5, —4, 4, 2, —4, —1), (1, 5, —4, —2, 3, 2,5). Декодируйте эти последовательности, т.е. определите номера строк в матрице кода максимальной длины, которым они соответствуют.

5.4. Выполните п. 5.3 при помощи усеченного алгоритма.

5.5. Найдите спектры функции *A* (п) для B = [- 1 , 1, 1], [1, - 1 , 1), [1, - 1 , - 1]. Проведите анализ результатов и на его основе попытайтесь найти аналитическое представление для спектра A («) при произвольном векторе B .

5.6. При передаче по каналу связи последовательностей *A* (и) на приемном конце получены следующие векторы: (—5, 4, 2, —5, 3, —4, 5,1), (5, 1, 2, 5, —3, -4, —3, 2). Декодируйте их при помощи диадной свертки, т. е. найдите соответствующие им векторы B .

5.7. Постройте граф восьмиточечного усеченного БПФ с прореживанием по частоте для синусе›идального входного сигнала с нулевой фазой и различной частотой, Проанализируйте степень усечения в зависимости от частоты.

ЛИТЕРАТУРА

\.Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. - М.: Мир, 1979. - 536 с.

2.Блейхут Р. Теория и практика кодов, контролирующих ошибки. - М.: Мир, 1986.-576 с.

3. *Воеводин В.В.* Математические модели и методы в параллельных процессах. - М.: Наука, 1986. - 296 с.

А,Казаринов Ю.М., Номоконов В.Н., Филиппов Ф.В. Применение микропроцессоров и микроЭВМ в радиотехнических системах. — М.: Высш. шк., 1988. - 207 с.

5.Канатова Л.В., Литейное В.Л., Финк Л.М. Быстрое корреляционное декодирование <?-ичных кодов максимальной длины // Проблемы передачи информации. — 1986. — Т. 22, вып. 2. - С. 98-103.

6. *Кнут Д.Е.* Искусство программирования для ЭВМ. Получисленные алгоритмы. — М,:Мир, 1978.-Т. 2. -724 с.

1.Кузьмин С.З. Основы проектирования систем цифровой обработки радиолокационной информации. — М.: Радио и связь, 1986. — 352 с.

%.Лицын С.Н., Шеховцев О.И. Быстрый алгоритм декодирования кодов Рида - Маллера первого порядка // Проблемы передачи информации. - 1983. - Т. 19, вып. 2. - С. 3-7.

9. *Лосев В.В., Бродская ЕБ., Коржик В.И.* Поиск и декодирование сложных дискретных сигналов. — М.: Радио и связь, 1988. — 224 с.

Ю.Макклеллан Дж. Х., Рейдер Ч.М. Применение теории чисел в цифровой обработке сигналов. - М.: Радио и связь, 1983. - 264 с.

11. *Нуссбаумер Г.* Быстрое преобразование Фурье и алгоритмы вычисления сверток. - М.: Радио и связь, 1985. - 248 с.

12. *Солодовников А.И., Спиваковский А.М.* Основы теории и методы спектральной обработки информации. - Л.: Изд-во Ленингр. ун-та, 1986. - 272 с.

13. Специализированные процессоры для высокопроизводительной обработки данных / О.Л. Бандман, Н.Н, Миренков, С.Г. Седухин и др. - Новосибирск: Наука. Сиб. отдние, 1988. - 208 с.

14. Цифровая обработки информации на основе быстродействующих БИС / СА.Гамкрелвдзе, А.В. Завьялов, П.П. Мальцев идр.; Под ред. В.Г. Домрачева. - М.: Энергоатомиздат, 1988. - 136 с.

ОГЛАВЛЕНИЕ

Учебное издание

Лосев Владислав Валентинович

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА ОБРАБОТКИ ИНФОРМАЦИИ.
АЛГОРИТМЫ ЦИФРОВОЙ ОБРАБОТКИ

Заведующий редакцией *А.Ф. Зиновьев.* Редактор *ЯМ Латышеве.* Младшие редакторы *С.А. Когадеева, А.М. Апель.* Художественный редактор *Ю.С Сергачев-* Технический редактор *Н.А.Лебедевич.* Корректоры *Н.Б. Кучмель, ЛА. Шлыкович.* Оператор *А. И. Маль* •

ИБ№2913

Подписано в печать с оригинала-макета 12.10.89 г. АТ 10459. Формат 60х90/16. Бумага кн.-журн. Гарнитура Пресс Роман. Печать офсетная. У^п. печ. л. 8,25. Усл. кр.-отт. 8,5 Уч.-изд.л. 8,58 .Тираж 5000 экз. Заказ 5302. Цена 35 к.

Издательство "Вышэйшая школа" Государственного комитета Белорусской ССР по печати. 220048, Минск, проспект Машерова, 11.

Типография "Победа". 222310. Молодечно.ул. Тавлая, 11.