Министерство Российской Федерации по связи и информатизации

Московский технический университет связи и информатики

Кафедра Радиотехнических систем

Учебное пособие

Быстрые алгоритмы обработки радиосигналов и их вычислительная сложность

Москва 2001

План УМД 2000 - 2001 уч.г.

Учебное пособие

Быстрые алгоритмы обработки радиосигналов и их вычислительная сложность

Авторы: доц. Крейнделин В.Б., проф. Шлома А.М.

Настоящее учебное пособие предназначено для студентов радиотехнического факультета по курсу "Радиотехнические системы".

Краткое содержание: В учебном пособии рассмотрены быстрые алгоритмы цифровой обработки сигналов: быстрое преобразование Фурье и быстрое преобразование Адамара. Приводятся известные оценки вычислительной сложности этих алгоритмов. Рассмотрен также алгоритм Штрассена для быстрого умножения матриц. Вводятся понятия порядка сложности алгоритма, алгоритмов полиномиальной и экспоненциальной сложности. В приложении приводятся примеры программ, реализующих алгоритмы быстрого преобразования Фурье и быстрого преобразования Адамара.

Рецензент: проф. Бонч - Бруевич А.М.

Издание утверждено Советом РТФ. Протокол №1 заседания кафедры РТС от 30 августа 2001 г.

Содержание.

Введение.

В связи с достижениями в области микроэлектроники и микропроцессорной техники в последние годы широкое применение находят цифровые системы связи, в которых приём сообщений осуществляется с помощью обработки сигналов на основе цифровых вычислительных устройств. Это обусловлено тем, что цифровые устройства позволяют реализовать практически любые сложные алгоритмы обработки сигналов, в том числе такие, которые трудно реализовать с помощью аналоговых систем. Ввиду простоты перестройки цифровых устройств, на их базе можно строить адаптивные системы.

Цель данного методического пособия состоит в том, чтобы дать студенту общее представление о наиболее часто встречающихся на практике алгоритмах обработки сигналов.

В цифровых системах обработке подвергаются сигналы с дискретным временем. Их можно получить из аналоговых сигналов взятием отсчётов в дискретные моменты времени в соответствии с теоремой Котельникова. Математически дискретный сигнал описывается функцией дискретного времени *s(i), i=0,1,…,N-1*, где *i* – номер отсчёта; N- общее число отсчетов.

Помимо описания дискретного сигнала совокупностью его отсчётов существуют и другие способы его представления. В частности, при анализе и синтезе дискретных фильтров целесообразно использовать спектральную форму, т. е. представить сигнал в виде совокупности элементарных функций. Для этой цели используются системы дискретных базисных функций *{$\varphi_i(j)$}, j=0,1,…,N-1; i=0,1,…,N-1*.

Прямое и обратное обобщенное преобразования Фурье сигнала *s(i)* определяются следующим образом

$$c(k) = \frac{1}{P_{\varphi_k} N} \sum_{i=1}^{N-1} s(i)\varphi_k^*(i), \quad k = 0,1,..., N-1;$$

$$s(i) = \sum_{k=0}^{N-1} c(k)\varphi_k(i), \quad i = 0,1,..., N-1, \tag{1}$$

Здесь c(k) - называется дискретным спектром

Выражения (1) удобно представлять в матричной форме:

$$C_{N\times 1} = \frac{1}{P_\varphi N} \Phi_N S_{N\times 1}; \quad S_{N\times 1} = \Phi'_N C_{N\times 1}, \tag{2}$$

где C_{N-1} и S_{N-1} – матрицы-столбцы размера $(N\text{x}1)$, составленные из коэффициентов c_k и отсчётов сигнала $s(i)$ соответственно; Φ_N – квадратная матрица преобразования порядка N, k-й строкой которой является базисная функция $\varphi^*_k(i)$; Φ'_N – квадратная матрица порядка N, получаемая из матрицы Φ_N её транспонированием и заменой элементов на комплексно-сопряжённые.

Среди различных дискретных спектральных преобразований в современной теории и технике цифровой радиосвязи нашли широкое применение дискретные преобразования Фурье, Уолша и Хаара. Дискретные преобразования применимы как для периодических сигналов, так и сигналов, заданных на конечном интервале. Конкретный вид дискретного преобразования определяется видом матрицы преобразования Φ_N.

1. Базовые алгоритмы обработки сигналов.

1.1 Дискретное преобразование Фурье.

В случае дискретного преобразования Фурье (ДПФ) в качестве системы базисных функций берётся система дискретных экспоненциальных функций

$$\varphi_k(i)= W^{ki}_N,\ i=0,1,\ldots,N-1;\ k=0,1,\ldots,N-1,$$

где $j=\sqrt{-1}$; $W_N=exp(-j2\pi/N)$.

Функции W^{ki}_N являются ортогональными, т.е. удовлетворяют условию

$$\sum_{i=0}^{N-1} W_N^{ki} W_N^{-li} = \begin{cases} N, & \text{если } (k-l)=mN, \text{ где } m-\text{целое число,} \\ 0 & \text{в других случаях.} \end{cases} \quad (3)$$

Действительно, если $k-l=mN$, то

$$\sum_{i=0}^{N-1} W_N^{ki} W_N^{-li} = \sum_{i=0}^{N-1} W_N^{mNi} = \sum_{i=0}^{N-1} \exp\{-\frac{j2\pi mN}{N} i\} = \sum_{i=0}^{N-1} 1^{mi} = N$$

В противном случае

$$\sum_{i=0}^{N-1} W_N^{ki} W_N^{-li} = \sum_{i=0}^{N-1} W_N^{(k-l)i} = \frac{1-W_N^{(k-l)N}}{1-W_N^{k-l}} = \frac{1-1}{1-W_N^{k-l}} = 0.$$

Функции W^{ki}_N являются периодическими с периодом N как по переменной k, так и по переменной i, т. е.

$$W_N^{ki} = W_N^{(k+N)i} = W_N^{k(i+N)}. \quad (4)$$

С помощью (1), прямое и обратное ДПФ можно записать как

$$c(k)=\frac{1}{N}\sum_{i=0}^{N-1}s(i)W_N^{ki},\quad k=0,1,...,N-1;$$
$$s(i)=\sum_{i=0}^{N-1}c(k)W_N^{-ki},\quad i=0,1,...,N-1. \tag{5}$$

Комплекснозначная числовая последовательность c(k) представляет собой спектр исходного сигнала s(i) в ортогональном базисе W_N^{-ki}.

Выражения (5) устанавливают взаимно однозначное соответствие между последовательностью отсчётов сигнала *s(i)* и её спектром *c(k).* Так как функции W_N^{ki} являются периодическими, то спектр дискретного сигнала также является периодическим: $c(k)=c(k+mN)$, $m=\pm1,\pm2,\ldots$ Свойство периодичности спектра является характерной особенностью ДПФ.

Рассмотрим некоторые свойства ДПФ.

1.Свойство линейности:

если $F[s_1(i)]=c_1(k)$ и $F[s_2(i)]=c_2(k)$, $i=k=0,1,\ldots,N-1$,

то $F[s_1(i)+s_2(i)]=c_1(k)+c_2(k)$, т. е. спектр суммы дискретных сигналов равен сумме их спектров.

2.Свойство комплексной сопряжённости:

если $F[s(i)]=c(k)$, $i=k=0,1,\ldots,N-1$, причём N – чётное число, *s(i)* – действительное, то

$c(N/2+l)=c^*(N/2-l)$, $l=0,1,\ldots,N-1$,

где $c^*(k)$ – величина, комплексно-сопряжённая *c(k).*

3.Теорема запаздывания :

если $F[s(i)]=c(k)$, $i=k=0,1,\ldots,N-1$, то $F[s(i+m)]=c(k)W^{km}$, где $m=1,\ldots,N-1$.

4.Теорема о свертке:

ДПФ от свертки двух сигналов равно произведению ДПФ этих сигналов, т. е.

$$F\left[s_{1\times2}(m)=\frac{1}{N}\sum_{i=0}^{N-1}s_1(i)s_2(m-i)\right]=F\left(s_1(i)\right)\cdot F\left(s_2(i)\right),\quad m=0,1,...,N-1. \tag{6}$$

Рассмотрим связь между ДПФ и непрерывным преобразованием Фурье. Пусть *s(t)* – непрерывный сигнал с числом степеней свободы $N=2F_{max}T_c$, где F_{max} – максимальная частота в спектре *s(t);* T_c – длительность сигнала *s(t).*. Представим этот сигнал в виде ряда Котельникова

$$s(t)=\sum_{i=0}^{N-1}s(i\Delta t)\frac{\sin 2\pi F_{\max}(t+i\Delta t)}{2\pi F_{\max}(t+i\Delta t)}\exp(-j\omega t)dt=$$
$$=\sum_{i=0}^{N-1}s(i\Delta t)\int_{-\infty}^{\infty}\frac{\sin 2\pi F_{\max}(t+i\Delta t)}{2\pi F_{\max}(t+i\Delta t)}\exp(-j\omega t)dt. \tag{7}$$

Интеграл в (7) представляет собой спектральную плотность $F_\varphi(j\omega)$ сдвинутой функции отсчётов .

Поэтому

$$F(j\omega) = \Delta t \sum_{i=0}^{N-1} s(i\Delta t)\exp(-ji\Delta t\omega), \quad (-2\pi F_{\max} \le \omega \le 2\pi F_{\max}).$$

Определим значения спектральной плотности в точках $\omega=k\Omega$, $k=0,\pm1,\ldots,\pm N/2$, $\Omega=2\pi/T_c$:

$$F(jk\Omega) = \Delta t \sum_{i=0}^{N-1} s(i\Delta t)\exp(-ji\Delta tk\Omega) = \Delta t \sum_{i=0}^{N-1} s(i\Delta t)\exp\frac{-jik2\pi}{N}. \quad (8)$$

Сравнивая (5) и (8), получим

$$c(k) = \begin{cases} (1/T_c)F(jk\Omega), & k = 0,1,2,\ldots,N/2, \\ (1/T_c)F(j(-N/2+l)\Omega), & k = N/2+l, \quad l = 1,2,\ldots,N. \end{cases}$$

Установим связь между ДПФ и рядом Фурье. Замечая, что коэффициенты ряда Фурье и дискретные значения спектральной плотности сигнала удовлетворяют соотношению $c_k=F(jk\Omega)/T_c$, находим

$$c(k) = \begin{cases} c_k, & k = 0,1,2,\ldots,N/2, \\ c_{N/2+l}, & k = N/2+l, \quad l = 1,2,\ldots,N/2-1. \end{cases} \quad (9)$$

Таким образом, в случае сигнала с конечным числом степеней свободы N коэффициенты $c(k)$, $k=0,1,\ldots,N/2$, совпадают с коэффициентами ряда Фурье.

Дискретное преобразование Фурье удобно представить в матричной форме:

$C_{N\times1}=F_N S_{N\times1}/N$; $S_{N\times1}=F'_N C_{N\times1}$,

где F_N – квадратная матрица порядка N, элементы которой $f_{ki}=W_N^{ki}$; F'_N – квадратная матрица порядка N, получаемая из матрицы F_N с помощью её транспонирования и замены элементов на комплексно-сопряжённые[*].

Дискретное преобразование Фурье применяется для вычисления спектров функций, заданных таблицами или графиками, обработки экспериментальных данных, а также нахождения сигнала на выходе дискретного фильтра .

---

[*] Математически это означает, что матрица F'_N является Эрмитово сопряженной по отношению к исходной матрице F_N.

1.2. Дискретное преобразование Уолша-Адамара.

В последние годы наряду с ДПФ в системах цифровой связи получило широкое распространение дискретное преобразование Уолша-Адамара (ДПУ).

В случае ДПУ в качестве системы базисных функций используется система дискретных функций Уолша $\{\mathrm{wal}_k(i)\}$ i=0, 1, …, N–1, k=0, 1, …, N–1, причем $N=2^n$, где n – целое число. Рисунок 1 наглядно иллюстрирует построение первых восьми дискретных функций Уолша.

Функции $\mathrm{wal}_k(i)$, i=0, 1, …, N–1, k=0, 1, …, N–1, являются ортогональными

$$\sum_{i=0}^{N-1}\mathrm{wal}_k(i)\mathrm{wal}_j(i)=\begin{cases}N & \text{при } k=j,\\ 0 & \text{в других случаях.}\end{cases} \qquad (10)$$

и удовлетворяют условию мультипликативности $\mathrm{wal}_k(i)\mathrm{wal}_j(i) = \mathrm{wal}_{k\oplus j}(i)$.

Прямое и обратное ДПУ в соответствии с (1) определяются как

$$c(k) = \frac{1}{N}\sum_{i=0}^{\mathrm{N-1}} s(i)\mathrm{wal}_k(i),$$
$$k = 0, 1, \ldots, \mathrm{N}-1, \qquad (11)$$
$$s(i) = \sum_{k=0}^{\mathrm{N}-1} c(k)\mathrm{wal}_k(i),$$
$$i = 0, 1, \ldots, \mathrm{N}-1.$$

Так же, как и в случае ДПФ, спектр оказывается периодическим: $c(k)=c(k+mN)$, где m–целое число.

Однако ДПУ имеет существенные отличия от ДПФ. Это касается, в частности, теорем запаздывания и свертки. В случае ДПФ умножение спектральных коэффициентов $c(k)$ на функцию W^{-km} приводит к сдвигу номеров отсчетов сигнала $s(i)$ на m единиц (к временному сдвигу последовательности отсчетов на m интервалов).

В случае ДПУ умножение спектральных коэффициентов $c(k)$ на $\mathrm{wal}_k(m)$ приводит к так называемому д и а д н о м у с д в и г у последовательности отсчетов сигнала:

$$c(k)\mathrm{wal}_k(m) = \frac{1}{N}\sum_{i=0}^{\mathrm{N}-1} s(i)\mathrm{wal}_k(i)\mathrm{wal}_k(m) = \qquad (12)$$
$$= \frac{1}{N}\sum_{i=0}^{N-1} s(i)\mathrm{wal}_k(i\oplus m) = \frac{1}{N}\sum_{i=0}^{\mathrm{N}-1} s(i\oplus m)\mathrm{wal}_k(i)$$

Запись $s(i\oplus m)$, i= 0, 1, …, 2^n–1, означает диадный сдвиг последовательности отсчетов сигнала $s(i)$, i= 0, 1, … , 2^n–1, на m интервалов; $i\oplus m$ – это десятичное число, двоичное представление которого определяется поразрядным сложением по модулю два двоичных представлений чисел i и m.

Заметим, что значение $(i \oplus m)$ при любых i и m не превышает число $N-1=2^n-1$. Поэтому при диадном сдвиге происходят перестановки отсчетов в пределах исходной совокупности. В качестве примера укажем, что если $N=16$, то отсчет с номером 7 при диадном сдвиге на 10 интервалов получает номер 13, а отсчет с номером 8 при том же диадном сдвиге получает номер 2.

Теорема о произведении спектра (или теорема о свертке) в случае ДПУ математически записывается как

$$c_1(k)c_2(k) = F[s_{1*2}(i)] = F\left[\frac{1}{N}\sum_{j=0}^{N-1} s_1(j)\, s_2(i \oplus j)\right], \qquad (13)$$

где $c_1(k)$, $c_2(k)$, $k=0, 1, \ldots, N-1$ – спектральные коэффициенты сигналов $s_1(i)$ и $s_2(i)$, $i=0, 1, \ldots, N-1$, в базисе функций Уолша.

ДПУ, как и ДПФ, удобно представить в матричной форме

$$\mathbf{C}_{N\times 1} = \mathbf{W}_N \mathbf{S}_{N\times 1}/N; \qquad \mathbf{S}_{N\times 1} = \mathbf{W}'_N \mathbf{C}_{N\times 1}, \qquad (14)$$

где $\mathbf{W}_N$ – матрица преобразования Уолша порядка N, k-я строка которой совпадает с функцией $\mathrm{wal}_k(i)$, $i=0, 1, \ldots, N-1$.

При N = 8

$$W_N = \begin{Vmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{Vmatrix}$$

Матрица преобразований Уолша $\mathbf{W}_N$ является симметрической. Поэтому $\mathbf{W}'_N = \mathbf{W}_N$ и (14) можно записать как $\mathbf{S}_{N\times 1} = \mathbf{W}_N \mathbf{C}_{N\times 1}$. Элементы матрицы преобразования Уолша принимают только значения ±1. Поэтому при вычислении как прямого, так и обратного ДПУ приходится выполнять лишь операции сложения и вычитания, т. е. отсутствует более трудоемкая операция умножения. В этом отношении ДПУ имеет большое преимущество по сравнению с ДПФ, при вычислении которого требуется производить умножение на комплексные числа $\exp(\pm j2\pi ki/N)$, причем действительную и комплексную части этих чисел приходится представлять большим числом разрядов для уменьшения ошибки вычисления. Поэтому, с точки зрения сложности реализации, алгоритм ДПУ является более простым, чем алгоритм ДПФ. Подробнее вопросы вычислительной сложности реализации различных алгоритмов обработки сигналов рассмотрены в разделах 2.1 и 2.5.

2. Быстрые алгоритмы обработки сигналов.

2.1.Вычислительная сложность алгоритмов.

Для оценки алгоритмов существует много критериев. Чаще всего нас будет интересовать порядок роста необходимых для решения задачи времени и емкости памяти при увеличении объема входных данных. Нам хотелось бы связать с каждой конкретной задачей некоторое число, называемое ее размером, которое выражало бы меру количества входных данных. Например, размером задачи умножения матриц может быть наибольший размер матриц-сомножителей. Размером задачи о графах может быть число ребер данного графа.

Время, затрачиваемое алгоритмом, как функция размера задачи, называется асимптотической временной сложностью. Аналогично можно определить *емкостную сложность* и *асимптотическую емкостную сложность*.

Именно асимптотическая сложность алгоритма определяет в итоге размер задач, которые можно решить с помощью этого алгоритма. Если алгоритм обрабатывает входные данные размера n за время cn^2 , где c - некоторая постоянная, то говорят, что временная сложность этого алгоритма есть $O(n^2)$ (читается «порядка n^2»). Точнее, говорят, что (неотрицательная) функция $g(n)$ есть $O(f(n))$, если существует такая постоянная c, что $g(n) \leq cf(n)$ для всех, кроме конечного (возможно, пустого) множества, неотрицательных значений n.

Можно было бы подумать, что колоссальный рост скорости вычислений, вызванный появлением нынешнего поколения цифровых вычислительных устройств, особенно устройств цифровой обработки сигналов, уменьшит значение эффективных алгоритмов. Однако происходит в точности противоположное. Так как вычислительные устройства работают все быстрее и мы можем решать более сложные задачи, именно сложность алгоритма определяет то увеличение размера задачи, которое можно достичь с увеличением скорости работы вычислительного устройства.

Здесь следует подчеркнуть, что при разработке современных систем цифровой радиосвязи имеет место тот факт, что сложность решаемых задач растет значительно быстрее производительности вычислительных устройств. Понятие вычислительной сложности вводится для того, чтобы классифицировать различные алгоритмы по степени сложности их практической реализации. Обычно в качестве критерия вычислительной сложности берут число операций (сложений, умножений, делений, сравнений и др.), необходимое для реализации данного алгоритма. Отметим, что вычислительная сложность зависит от типа процессора (вычислительного устройства), на котором реализован алгоритм.

Рассмотрим теперь некоторые способы уменьшения вычислительной сложности алгоритмов обработки сигналов. В качестве иллюстрации мы будем использовать рассмотренные в разделе 1.2 дискретные преобразования Фурье.

Непосредственное вычисление спектра сигнала по (5) требует выполнения большого числа операций и соответственно предъявляет высокие требования к быстродействию сигнального процессора. Так, в случае ДПФ необходимо произвести N^2 операций умножения и $(N–1)N$ операций сложения с комплексными числами. При больших значениях N число операций может оказаться недопустимо большим. Поэтому проблема разработки более рациональных алгоритмов вычисления ДПФ является весьма актуальной с практической точки зрения. Такие алгоритмы были найдены и получили название алгоритмов б ы с т р ы х п р е о б р а з о в а н и й. Их применение значительно сокращает число арифметических операций, необходимых для вычисления спектра, а также объем памяти вычислительного устройства.

Известны различные варианты алгоритмов быстрых преобразований Фурье (БПФ), а также Уолша - Адамара (БПУ), Адамара (БПА), и др.. Несмотря на многообразие этих алгоритмов, обусловленное различными подходами при их выводе, все они могут быть получены либо представлением одномерного массива отсчетов сигнала двумерным, либо факторизацией* матриц преобразований.

Первый метод состоит в разбиении исходной последовательности отсчетов на две вспомогательные более короткие последовательности, нахождении спектров этих вспомогательных последовательностей и определении по ним спектра исходной последовательности.

Можно показать, что суммарное число операций, необходимое для вычисления спектров вспомогательных последовательностей и определения по ним спектра исходной последовательности, оказывается меньше числа операций, которое требуется для вычисления спектра в соответствии (5).

Действительно, пусть исходная последовательность состоит из N отсчетов, причем N – четное число. Для вычисления ее спектра по (5) требуется выполнить N^2 умножений и $N(N–1)$ сложений.

Разобьем ее на две последовательности одинаковой длины. Для нахождения их спектров по (5) требуется выполнить $N^2/2$ умножений и $N(N/2–1)/2$ сложений. Так как для пересчета спектров вспомогательных последовательностей в спектр исходной требуется небольшое число операций, то однократное применение процедуры позволяет сократить объем вычислений практически вдвое.

Для дальнейшего уменьшения числа операций описанную процедуру применяют многократно для нахождения спектров вспомогательных последовательностей.

Метод факторизации заключается в представлении матрицы преобразований Φ_N в виде произведения более простых, состоящих в основном из нулевых элементов, матриц*: $\Phi_N = \Phi_N^{(1)}, \Phi_N^{(2)} \ldots \Phi_N^{(k)}$, где k – число сомножителей числа N.

---

* Факторизация матрицы - это представление ее в виде произведения двух и более других матриц.

* Матрицы, содержащие большое количество нулевых элементов и малое количество ненулевых элементов, называемых разреженными матрицами.

Соответственно коэффициенты $c(k)$ определяются матричным уравнением

$$\mathrm{C}_{N\times1}= \Phi_N^{(1)}\,\Phi_N^{(2)}\ldots\,\Phi_N^{(k)}\mathrm{S}_{N\times1}/N. \qquad (15)$$

Из (15) следует, что коэффициенты $c(k)$ определяются в несколько этапов: сначала находится матрица-столбец $\mathrm{S}_{N\times1}^{(1)} = \Phi_N^{(k)}\mathrm{S}_{N\times1}$, затем матрица-столбец $\mathrm{S}_{N\times1}^{(2)} =\Phi_N^{(k-1)}\mathrm{S}_{N\times1}^{(1)}$ и т. д., наконец, матрица-столбец $\mathrm{S}_{N\times1}^{(k)} = \mathrm{C}_{N\times1} =1/N[\Phi_N^{(1)}\,\mathrm{S}_N^{(k-1)}]$.

Экономия в числе операций достигается тем, что на каждом этапе производится сравнительно небольшое число операций, так как матрицы $\Phi_N^{(i)}$, $i= 1, 2, \ldots, k$, являются разреженными.

2.2. Быстрое преобразование Фурье (БПФ).

При изучении алгоритмов БПФ можно было бы исходить из общих позиций, указанных в предыдущем параграфе. Однако для облегчения понимания сущности БПФ рассмотрим частные случаи – алгоритмы БПФ с прореживанием по времени и по частоте. Оба алгоритма основаны на представлении одномерного массива отсчетов сигнала двумерным. Рассмотрим БПФ с прореживанием по времени. Пусть $\{s(i)\}$, $i= 0, 1, 2, \ldots, N-1$, – последовательность отсчетов сигнала, причем $N=2^n$, где n – целое число. Разобьем ее на две последовательности $\{s_1(i)\}$ и $\{s_2(i)\}$ таким образом, что $s_1(i)=s(2i)$, $i=0, 1, 2, \ldots, N/2-1$; $s_2(i)=s(2i+1)$, $i=0, 1, 2, \ldots, N/2-1$, т. е. $\{s_1(i)\}$ и $\{s_2(i)\}$ образуют соответственно из четных и нечетных членов последовательности $\{s(i)\}$.

С учетом (5) ДПФ сигнала $\{s(i)\}$ теперь можно представить в виде

$$c(k) = \frac{2}{N}\sum_{i=0}^{N-1} s(i)W_N^{ki} + \frac{2}{N}\sum_{i=0}^{N-1} s(i)W_N^{ki} =$$

i-четные $\qquad\qquad$ i-нечетные

$$= \frac{2}{N}\sum_{i=0}^{\frac{N}{2}-1} s(2i)W_N^{2ki} + \frac{2}{N}\sum_{i=0}^{\frac{N}{2}-1} s(2i+1)W_N^{(2i+1)k}$$

или, обращая внимание на то, что

$$W_N^2 = [\exp(-j2\pi/N)]^2 = \exp\left(-j\frac{2\pi}{H/2}\right),$$

в виде

$$c(k) = \frac{2}{N}\sum_{i=0}^{N/2-1} s_1(i)W_{N/2}^{ki} + \frac{2}{N}W_N^k\sum_{i=0}^{N/2-1} s_2(i)W_{N/2}^{ki} \qquad (16)$$

Первое слагаемое в (16) есть спектр последовательности $\{s_1(i)\}$, а второе – спектр последовательности $\{s_2(i)\}$, умноженный на W_N^k. Таким образом,

$$c(k) = c_1(k) + c_2(k)W_N^{\,k}. \qquad (17)$$

Так как $c_1(k)$ и $c_2(k)$ определены лишь для $0 = k = N/2-1$, то для нахождения коэффициентов $c(k)$, $k=N/2, N/2+1, \ldots, N-1$, формулу (17) надо соответствующим образом доопределить. Учитывая периодичность функции $W_{N/2}^{\,k}$, т. е. $W_{N/2}^{\,m+N/2} = W_{N/2}^{\,m}$, а также то, что $W_N^{\,m+N/2} = \exp\left(-j2\pi\dfrac{m+N/2}{N}\right) = \exp\left(-j2\pi\dfrac{m}{N}\right)\exp(-j\pi) = -\exp\left(-j2\pi\dfrac{m}{N}\right) = -W_N^{\,m}$, окончательно находим

$$c(k) = \begin{cases} c_1(k) + c_2(k)W_N^{\,k}, & 0 \le k \le N/2 - 1, \\ c_1(k - N/2) - c_2(k - N/2)W_N^{\,k-N/2}, & N/2 \le k \le N - 1. \end{cases} \qquad (18)$$

Рассмотренная методика может быть применена для нахождения спектров $c_1(k)$ и $c_2(k)$, $k= 0, 1, \ldots, N/2-1$. При этом каждая из последовательностей $\{s_j(i)\}$, $j= 1, 2$, разбивается на две под последовательности $\{s_{j1}(i)\}$ и $\{s_{j2}(i)\}$, $j= 0, 1, \ldots, N/4-1$, по тому же принципу, что и последовательность $s(i)$, $i= 0, 1, \ldots, N-1$.

Выражение для спектра $c_j(k)$, $j= 1, 2$, запишется в виде $c_j(k)=c_{j1}(k)+W_{N12}^{\,k}c_{j2}(k)= =c_{j1}(k)+W_N^{\,2k}c_{j2}(k)$, $0 = k = N/2-1$, $j= 1, 2$, где $c_{j1}(k)$ и $c_{j2}(k)$ – спектры последовательностей $\{s_{j1}(i)\}$ и $\{s_{j2}(i)\}$ соответственно. Процесс разбиения последовательностей продолжается до тех пор, пока каждая из них не будет содержать только два отсчета.

На рис. 2 с помощью направленного графа представлена последовательность выполнения операций в случае описанного алгоритма для $N=8$. В нижней части рис.2 показана элементарная операция, на использовании которой основан описываемый алгоритм БПФ. Эта операция получила название 'бабочка'. В настоящее время существуют специализированные процессоры БПФ, в которых 'бабочка' выполняется за один такт.

Порядок следования отсчетов входной последовательности выбран таким образом, чтобы выходная последовательность $c(k)$ имела естественный порядок следования, т. е. $k= 0, 1, \ldots, N-1$. Можно показать, что если $N=2^n$, n – целое число, то входная последовательность должна быть расположена в двоично-инверсном порядке, который определяется следующим образом. Порядковый номер отсчета входной последовательности записывают в двоичном коде, используя n разрядов. Затем порядок следования разрядов инвертируют (старший разряд становится младшим и т. д.). Десятичное число, соответствующее полученной двоичной комбинации, и является номером элемента входной последовательности после перестановки. Такая перестановка входных данных называется двоично - инверсной.

Для вычисления обратного ДПФ также можно использовать описанный алгоритм. Для этого составим выражение, комплексно-сопряженное со вторым уравнением в (5):

$$s^*(i) = \sum_{k=0}^{N-1} c^*(k)W^{ki}, \quad i= 0, 1, \ldots, N-1. \qquad (19)$$

Правая часть (19) с точностью до сомножителя 1/N представляет собой ДПФ последовательности $\{c^*(k)\}$, а следовательно, $s^*(i)$ можно вычислить, используя вышеописанный быстрый алгоритм. Зная $\{s^*(i)\}$ нетрудно осуществить переход к искомой последовательности $\{ s(i)\}$.

Рассмотрим БПФ с прореживанием по частоте. В этом случае исходная последовательность разбивается на короткие последовательности следующим образом: $s_1(i)=s(i)$, $i= 0, 1, \ldots, N/2-1$; $s_2(i)=s(i+N/2)$, $i= 0, 1, \ldots, N/2-1$, т. е. $\{s_1(i)\}$ содержит первые $N/2$ членов последовательности $\{s(i)\}$, а $\{s_2(i)\}$ – остальные.

Преобразование Фурье последовательности $\{s(i)\}$ с учетом ее разбиения запишется как

$$c(k) = \frac{2}{N}\sum_{i=0}^{N/2-1} s(i)W_N^{ik} + \frac{2}{N}\sum_{i=N/2}^{N-1} s(i)W_N^{ik} = \frac{2}{N}\sum_{i=0}^{N/2-1} s_1(i)W_N^{ik} +$$

$$+\frac{2}{N}\sum_{i=0}^{N/2-1} s_2(i)W_N^{(i+N/2)k} = \frac{2}{N}\sum_{i=0}^{N/2-1} [s_1(i)+s_2(i)\exp(-j\pi k)]W_N^{ik}. \qquad (20)$$

Из (20) получаем для четных коэффициентов

$$c(2k)=\frac{2}{N}\sum_{i=0}^{N/2-1} [s_1(i)+s_2(i)](W_N^2)^{ik}=\frac{2}{N}\sum_{i=0}^{N/2-1} [s_1(i)+s_2(i)]W_{N/2}^{ik} \qquad (21)$$

и для нечетных коэффициентов

$$c(2k+1)=\frac{2}{N}\sum_{i=0}^{N/2-1} \{[s_1(i)-s_2(i)](W_N^i\}W_{N/2}^{ik}. \qquad (22)$$

Таким образом, коэффициенты $c(2k)$ и $c(2k+1)$ можно получить, зная спектры последовательностей $\{f(i)\}=\{s_1(i)+s_2(i)\}$, $i= 0, 1, \ldots, N/2-1$, и $\{h(i)\}=\{s_1(i)-s_2(i)\}$, $i= 0, 1, \ldots, N/2-1$. В свою очередь, спектры $c(2k)$ и $c(2k+1)$, $k= 0, 1, \ldots, N/2-1$, можно вычислить по описанной выше методике, разбивая каждую из последовательностей $\{f(i)\}$ и $\{h(i)\}$ на две короткие по тому же принципу, что и последовательность $\{s(i)\}$. Процесс продолжается до тех пор, пока каждая из последовательностей не будет содержать только два отсчета.

Общее число операций, проводимых при вычислении спектра, в обоих рассмотренных алгоритмах одинаково и равно приблизительно $N \log N$. Таким образом, применение БПФ позволяет уменьшить число производимых операций приблизительно в $N/\log N$ раз. При больших значениях N число операций уменьшается весьма значительно. Например, при N = 1024 достигается ускорение вычислителей в 100 раз. В заключение отметим, что алгоритмы БПФ могут быть получены также методом факторизации матрицы преобразований.

2.3. Быстрое преобразование Уолша-Адамара.

Быстрые алгоритмы преобразований Уолша существуют для любого способа упорядочения функций Уолша. Ниже рассматривается лишь быстрое преобразование Адамара (БПА). Алгоритм такого преобразования можно получить либо факторизацией матрицы Адамара $\mathbf{A}_N$, либо ее разбиением. Будем использовать второй метод как более наглядный и простой для понимания сущности алгоритма.

Напомним, что матрицей Адамара A_N порядка N называется квадраткая матрица, получаемая с помощью рекуррентрого соотношения:

$A_1 = 1$

$$A_N = \begin{Vmatrix} A_{N/2} & A_{N/2} \\ A_{N/2} & -A_{N/2} \end{Vmatrix}$$

Приведем вывод алгоритма БПА на конкретном примере. Пусть $s(i)$, i= 0, 1, …, N–1, – последовательность отсчетов сигнала, причем N=8. Тогда

$$\mathbf{C}_{8\times1} = \frac{1}{8}\mathbf{A}_8\mathbf{S}_{8\times1}. \qquad (24)$$

Воспользовавшись (23), преобразование (24) перепишем в виде

$$\begin{Vmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \\ c(4) \\ c(5) \\ c(6) \\ c(7) \end{Vmatrix} = \frac{1}{8}\begin{Vmatrix} A_4 & A_4 \\ A_4 & -A_4 \end{Vmatrix} \times \begin{Vmatrix} s(0) \\ s(1) \\ s(2) \\ s(3) \\ s(4) \\ s(5) \\ s(6) \\ s(7) \end{Vmatrix}. \qquad (25)$$

Следует отметить, сто строки матрицы Адамара $\mathbf{A}_N$ представляет собой функции Уолша. Матрица Адамара $\mathbf{A}_N$ отличается от матрицы $\mathbf{W}_N$,приведенной в разделе 1.2., только порядком следования строк.

Выражение (25) можно разбить на два:

$$\begin{Vmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \end{Vmatrix} = \frac{1}{8}\mathbf{A}_4 \begin{Vmatrix} s_1(0) \\ s_1(1) \\ s_1(2) \\ s_1(3) \end{Vmatrix}; \qquad \begin{Vmatrix} c(4) \\ c(5) \\ c(6) \\ c(7) \end{Vmatrix} = \frac{1}{8}\mathbf{A}_4 \begin{Vmatrix} s_1(4) \\ s_1(5) \\ s_1(6) \\ s_1(7) \end{Vmatrix}, \qquad (26)$$

где $s_1(i) = s(i)+s(i+4)$, i= 0, 1, 2, 3; $s_1(i) = s(i–4)–s(i)$, i= 4, 5, 6, 7.

Описанную методику применяем к каждому из преобразований (26). В результате имеем:

$$\begin{Vmatrix} c(0) \\ c(1) \\ c(2) \\ c(3) \end{Vmatrix} = \frac{1}{8} \begin{Vmatrix} \mathbf{A}_2 & \mathbf{A}_2 \\ \mathbf{A}_2 & -\mathbf{A}_2 \end{Vmatrix} \begin{Vmatrix} s_1(0) \\ s_1(1) \\ s_1(2) \\ s_1(3) \end{Vmatrix}; \quad \begin{Vmatrix} c(4) \\ c(5) \\ c(6) \\ c(7) \end{Vmatrix} = \frac{1}{8} \begin{Vmatrix} \mathbf{A}_2 & \mathbf{A}_2 \\ \mathbf{A}_2 & -\mathbf{A}_2 \end{Vmatrix} \begin{Vmatrix} s_1(4) \\ s_1(5) \\ s_1(6) \\ s_1(7) \end{Vmatrix}. \quad (27)$$

Каждое из преобразований (27) вновь представляем в виде двух выражений

$$\begin{Vmatrix} c(0) \\ c(1) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_1(0) + s_1(2) \\ s_1(1) + s_1(3) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_2(0) \\ s_2(1) \end{Vmatrix};$$

$$\begin{Vmatrix} c(4) \\ c(5) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_1(4) + s_1(6) \\ s_1(5) + s_1(7) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_2(4) \\ s_2(5) \end{Vmatrix}; \quad (28)$$

$$\begin{Vmatrix} c(2) \\ c(3) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_1(0) - s_1(2) \\ s_1(1) - s_1(3) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_2(2) \\ s_2(3) \end{Vmatrix};$$

$$\begin{Vmatrix} c(6) \\ c(7) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_1(4) - s_1(6) \\ s_1(5) - s_1(5) \end{Vmatrix} = \frac{1}{8} \mathbf{A}_2 \begin{Vmatrix} s_2(6) \\ s_2(7) \end{Vmatrix}.$$

Наконец, учитывая, что

$$\mathbf{A}_2 = \begin{Vmatrix} 1 & 1 \\ 1 & -1 \end{Vmatrix},$$

получаем $c(0)=[s_2(0)+s_2(1)]/8=s_3(0)/8$; $c(1)=[s_2(0)-s_2(1)]/8=s_3(1)/8$;
$c(2)=[s_2(2)+s_2(3)]/8=s_3(2)/8$; $c(3)=[s_2(2)-s_2(3)]/8=s_3(3)/8$;
$c(4)=[s_2(4)+s_2(5)]/8=s_3(4)/8$; $c(5)=[s_2(4)-s_2(5)]/8=s_3(5)/8$;
$c(6)=[s_2(6)+s_2(7)]/8=s_3(6)/8$; $c(7)=[s_2(6)-s_2(7)]/8=s_3(7)/8$.

На рис.3 с помощью направленного графа представлена последовательность операций для описанного алгоритма.

Для реализации алгоритма требуются только операции сложения и вычитания. Общее их число равно 24.

В общем случае для любого $N=2^n$ последовательность операций такая же, как в рассмотренном примере. Общее число этапов вычисления коэффициентов $c(k)$, $k= 0, 1, \ldots, 2^n-1$, равно $\log_2 2^n=n$. На каждом i-м этапе участвует 2^{i-1} групп по $N/2^{i-1}$ элементов. Над половиной элементов каждой группы осуществляется операция сложения, над другой половиной – вычитания. Общее число производимых операций при вычислении всех коэффициентов спектра равно $N\log_2 N$. Выигрыш в числе операций по сравнению с прямым вычислением коэффициентов по формуле типа (11) составляет приблизительно $N/\log_2 N$ раз. При больших значениях N выигрыш может достигать сотен и тысяч раз. Алгоритм применим для вычисления обратного преобразования Уолша-Адамара.

Быстрые преобразования Уолша-Адамара могут быть получены также методом факторизации матриц.

2.4. Алгоритм Штрассена для быстрого умножения матриц.

В этом разделе мы рассмотрим один алгоритм вычисления произведения 2-х матриц, имеющий меньшее количество операций умножения, чем традиционные алгоритмы. Рассмотрим сначала традиционные алгоритмы.

Пусть *A* и *B* – две *(N*N)* – матрицы, причем N – степень числа 2. Можно разбить каждую матрицу *A* и *B* на четыре *((N/2)*(N/2))* –матрицы и через них выразить произведение матриц *A* и *B*:

$$\begin{Vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{Vmatrix} \begin{Vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{Vmatrix} = \begin{Vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{Vmatrix} \qquad (29)$$

где

$$C_{11}=A_{11}B_{11}+A_{12}B_{21},$$
$$C_{12}=A_{11}B_{12}+A_{12}B_{22},$$
$$C_{21}=A_{21}B_{11}+A_{22}B_{21},$$
$$C_{22}=A_{21}B_{12}+A_{22}B_{22}.$$

Если рассматривать *A* и *B* как *(2*2)* – матрицы, элементами каждой из которых служат *((N/2)*(N/2))* – матрицы, то их произведение можно выразить через суммы и произведения *((N/2)*(N/2))* – матриц. Допустим, что C_{ij} вычисляются с помощью *m* умножений и *a* сложений (или вычитаний) *((N/2)*(N/2))* – матриц. Рекурсивно применяя этот алгоритм, можно вычислить произведение двух *(N*N)* – матриц за время *T(N)*[*], удовлетворяющее неравенству

$$T(N) \le mT\left(\frac{N}{2}\right) + \frac{aN^2}{4}, N > 2 \qquad (30)$$

если *N* – степень числа 2. Первое слагаемое в правой части неравенства – это сложность умножения *m* пар *((N/2)*(N/2))* – матриц, а второе – сложность выполнения *a* сложений при условии, что каждое сложение или вычитание двух *((N/2)*(N/2))* – матриц требует $N^2/4$ времени.

Рассуждения показывают, что при $m > 4$ решение неравенства ограничено сверху величиной $kN^{\log(m)}$, где *k* – некоторая постоянная. Вид этого решения не зависит от *a*,

[*] Здесь имеется в виду, что функция T(N) есть вычислительная сложность алгоритма умножения двух матриц размерности N.

т.е. от числа сложений. Таким образом, если $m < 8$, то мы получаем метод, асимптотически лучший обычного метода сложности $O(n^3)$.

Штрассен изобрел искусный метод умножения двух (2*2) –матриц с произвольными элементами, в котором достаточно семи умножений. Рекурсивно применяя свой метод, он смог умножить две *(N*N)* – матрицы за время $O(N^{log7})$, что по порядку примерно равно $N^{2,81}$.

Можно показать, что произведение двух (2*2) – матриц с произвольными элементами можно вычислить, выполнив семь умножений и 18 сложений (вычитаний). Поясним это более подробно.

Пусть требуется вычислить произведение матриц

$$\begin{Vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{Vmatrix} = \begin{Vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{Vmatrix} \begin{Vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{Vmatrix} \qquad (31)$$

Для этого сначала вычислим произведения

$$\begin{aligned} M_1&=(A_{12}-A_{22})(B_{21}+B_{22}),\\ M_2&=(A_{11+}A_{22})(B_{11}+B_{22}),\\ M_3&=(A_{11}-A_{21})(B_{11}+B_{12}),\\ M_4&=(A_{11+}A_{12})B_{22},\\ M_5&=A_{11}(B_{12}-B_{22}),\\ M_6&=A_{22}(B_{21}-B_{11}),\\ M_7&=(A_{21+}A_{22})B_{11}. \end{aligned} \qquad (32)$$

Затем вычисляем C_{ij} по формулам

$$\begin{aligned} C_{11}&=M_1+M_2-M_4+M_6,\\ C_{12}&=M_4+M_5,\\ C_{21}&=M_6+M_7,\\ C_{22}&=M_2-M_3+M_5-M_7. \end{aligned}$$

Число операций подсчитывается тривиально. Доказательство того, что в результате получаются требуемые величины C_{ij}, представляет собой простое алгебраическое упражнение.

Вопрос о числе операций решается следующим образом.

Покажем, что две *(N*N)* – матрицы с произвольными элементами можно перемножить за $O(N^{log7})$ арифметических операций.

Для доказательства этого утверждения сначала рассмотрим случай $N=2^k$. Пусть *T(N)* – число арифметических операций, необходимых для умножения двух *(N*N)* – матриц. Тогда по доказанному выше утверждению имеем:

$$T(N)=7T(\frac{N}{2})+18(\frac{N}{2})^2 \quad \text{для} \quad N \geq 2. \qquad (33)$$

Следовательно, *T(N)* составляет $O(7^{logN})$, или $O(N^{log7})$.

Если *n* не является степенью числа 2, то нетрудно показать, что и в этом случае $T(N) = O(N^{log7})$.

Алгоритм Штрассена (31),(32) имеет смысл использовать, если N велико (N > 40) и время выполнения операции умножения значительно превышает время сложения (вычитания).

2.5. Порядок сложности алгоритма. Алгоритмы полиномиальной и экспоненциальной сложности.

Как было уже сказано в разделе 2.1, *вычислительная сложность алгоритма* – это количество элементарных операций, которое необходимо для реализации данного алгоритма. Отметим, что вычислительная сложность может существенным образом зависеть от порядка выполнения операций в данном алгоритме. Для иллюстрации этого факта приведем следующий пример: Пусть небоходимо вычислить следующее произведение:

$$\vec{Y} = \overline{A} * \overline{B} * \vec{x} \qquad (34)$$

где $\overline{A}, \overline{B}$ - квадратные матрицы размерности N×N, а $\vec{x}$ - вектор столбец размерности $N \times 1$.

Вычисление величины Y в (34) можно вести вдумя очевидными способами. Можно, например, сначала перемножить матрицы A и B, а затем их произведение C умножить на вектор X. Это составляет содержание Способа 1, (35):

Способ1. $Y = \left(\overline{A} * \overline{B}\right) * \vec{x} = C * x$:

Первый способ требует для вычисления элементов матрицы C = A * B

$n_{умн} = N^2 * N = N^3$ операций умножения

$n_{сложс} = N^2 * (N - 1)$ операций сложения

Теперь матрицу C необходимо умножить на вектор x. Для получения каждого элемента Y_j требуется сделать N умножений и (N-1) сложений. Для $\vec{Y} = \overline{C} * \vec{x}$ нужно: N*N умножений , (N-1)*N сложений.

Итак, для $\vec{Y} = \left(\overline{A} * \overline{B}\right) * \vec{x}$ необходимо выполнить

$n_{умн} = N^3 + N^2 = N^2 * (N + 1)$ операций умножения

$n_{сложс} = N^3 - N^2 + N^2 - N = N^3 - N$ операций сложения.

При таком порядке вычисления число умножений и сложений пропорционально третьей степени размерности матрицы, т.. сложность алгоритма вычисления вектораY по способу 1 есть $O(N^3)$.

Рассмотрим теперь другой способ.

Способ 2. $\vec{Y} = \overline{A} * \left(\overline{B} * \vec{x}\right)$ (36)

Согласно (35) сначала вычисляется произведение C=B*x, а затем матрица A умножается на вектор C. Подсчитаем число операций при данном порядке проведения вычислений.

Для вычисления C= $\overline{B} * \vec{x}$ требуется N^2 умножений и $N*(N-1)$ сложений.

Для вычисления произведения A*C также требуется N^2 умножений и $N*(N-1)$ сложений. Таким образом, при использовании Способа 2 (36) для вычисления вектора Y всего требуется

$n_{умн} = 2N^2$ умножений

$n_{слож} = 2N(N-1)$ сложений.

При использовании(36) сложность алгоритма вычисления вектора Y есть величина, пропорциональная второй степени размерности матрицы, т.е. $O(N^2)$.
Отсюда можно сделать вывод о предпочтительности использования Способа 2 (36) для вычисления вектора Y. Введем теперь понятие порядка сложности.

Порядок сложности характеризует зависимость между числом операций и некоторыми параметрами алгоритма (в данном случае порядком матрицы).

Говорят, что алгоритм имеет полиномиальную сложность, если число операций в нём равно полиному некоторой степени от параметра алгоритма.

Степень этого полинома называют порядком сложности.

Если число операций, необходимое для реализации алгоритма, пропорционально первой степени некоторого параметра, то такой алгоритм называется алгоритмом с линейной сложностью. На практике такие алгоритмы являются предпочтительными.

Одна из наиболее актуальных и важных задач обработки сигналов - снижение порядка сложности алгоритмов. Следует отметить, что множество алгоритмов обработки сигналов далеко не исчерпывается алгоритмами полиномиальной сложности.

Часто бывает так, что число операций увеличивается пропорционально экспоненте от параметра алгоритма. В этом случае говорят, что алгоритм имеет экспоненциальную вычислительную сложность.

Отметим, что число операций не всегда однозначно характеризует сложность алгоритма. Это связано с тем, что различные операции имеют различное время выполнения, к тому же зависящие от типа используемого сигнального процессора.

Например, некоторые процессоры не имеют аппаратного умножителя и поэтому операция умножения реализуется в них значительно медленнее операции сложения. В этом случае путь сокращения времени вычислений состоит в замене операций умножения операциями сложения, как это было с успехом сделано в алгоритме Штрассена.

В то же время использование алгоритма Штрассена для обработки сигналов с помощью сигнальных процессоров с одинаковым временем выполнения умножения и сложения, совершенно неэффективно.

Таким образом, вычислительная сложность алгоритма является величиной относительной, зависящей от типа сигнального процессора.

Заключение.

В данном методическом пособии рассмотрены лишь некоторые вопросы современной теории обработки сигналов. Вообще, проблема построения быстрых алгоритмов является важнейшей проблемой, возникающей при разработке систем цифровой связи. Актуальность этой проблемы еще более возрастает именно в наше время, когда при разработке систем связи требуется достигать их конкурентоспособности, т.е. наряду с высокими качественными характеристиками система должна иметь низкую стоимость. В свою очередь, низкая стоимость достигается использованием алгоритмов обработки сигналов с низкой вычислительной сложностью, позволяющих их реализовать на дешевых сигнальных процессорах.

Список литературы.

1. Ахмед Н., Рао К.Р. Ортогональные преобразования при обработке цифровых сигналов: Пер. С анг. /под ред. И.Б. Фоменко - М.: Связь, 1980
2. Дядюнов Н.Г., Сенин А.И. Ортогональные и квазиортогональные сигналы. М.: Связь, 1977
3. Акимов П.С., Сенин А.И., Соленов В. И. Сигналы и их обработка в информационных системах. - М.: Радио и связь, 1994.
4. Залманзон Л.А. Преобразования Фурье, Уолша, Хаара и их применение в управлении, связи и других областях. - М.: Наука, 1989.
5. Рабинер Л, Гоулд Б. Теория и применение цифровой обработки сигналов: Пер. С англ./ под ред. Ю.Н. Александрова. - М.: Мир 1978.
6. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. С англ./ под ред. Ю. В. Матиясевича, М. Мир, 1979.
7. Куприянов М.С., Матюшкин Б. Д. Цифровая обработка сигналов: процессоры, алгоритмы, средства проектирования. СПб. Политехника, 1999.
8. Гольденберг Л.М., Матюшкин Б.Д, Поляк М. Н. Цифровая обработка сигналов: Справочник - М.: Радио и связь, 1985.

Приложение

Рассмотренные в данном методическом пособии алгоритмы легко можно реализовать средствами различных языков программирования. В качестве примера ниже приводится текст программы 'fft.c', реализующей алгоритм БПФ с прореживанием во времени, описанный в разделе 2.2. Данная программа написана на языке СИ и является стандартной программой в известной системе MATLAB.

```
/* ------------------------------------------- */
/*    A COOLEY-TUKEY RADIX-2, DIF  FFT PROGRAM    */
/*      COMPLEX INPUT DATA IN ARRAYS X AND Y      */
/*                                                */
/*    C. S. BURRUS, RICE UNIVERSITY, SEPT 1983    */
/*                                                */
/* ------------------------------------------- */
```

```
/*  Copyright (c) 1995-98 by The MathWorks, Inc.       */
/*  $Revision: 1.6 $  $Date: 1997/11/26 20:24:41 $  */

#include <math.h>

#include "tmwtypes.h" /* real_T, etc */

void fft(int_T n, real_T *x, real_T *y)
{
    real_T a, e, xt, yt;
    int_T  n1, n2, i, j, k, m;

    /*
     * Calculate m such that n=2^m
     *
     * NOTE: If frexp() == 1, then frexp does not conform
     * to the ANSI C spec of [0.5, 1)
     */
    if ( frexp((real_T)n, &m) != 1.0 ) {
        m--;
    }

    /* --------------MAIN FFT LOOPS-------------------------
--- */

    /* Parameter adjustments */
    --y;
    --x;

    /* Function Body */
    n2 = n;
    for (k = 1; k <= m; ++k) {
     n1 = n2;
     n2 /= 2;
     e = (real_T)6.28318530717958647692528676655900576839 /
n1;
     a = 0.0;
     for (j = 1; j <= n2; ++j) {
         real_T c = cos(a);
         real_T s = sin(a);

         a = j * e;
         for (i = j; n1 < 0 ? i >= n : i <= n; i += n1) {
          int_T q = i + n2;

          xt = x[i] - x[q];
```

```
            x[i] += x[q];
            yt = y[i] - y[q];
            y[i] += y[q];
            x[q] = c * xt + s * yt;
            y[q] = c * yt - s * xt;
           }
      }
     }

     /* ------------DIGIT REVERSE COUNTER----------------- */

     j = 1;
     n1 = n - 1;
     for (i = 1; i <= n1; ++i) {
      if (i < j) {
           xt = x[j]; x[j] = x[i]; x[i] = xt;
           xt = y[j]; y[j] = y[i]; y[i] = xt;
      }
      k = n / 2;
      while (k<j) {
           j -= k;
           k /= 2;
      }
      j += k;
     }
}

/* [EOF] fft.c */
```

Приведем также в качестве примера программу, реализующую алгоритм быстрого преобразования Уолша-Адамара, который был описан в разделе 2.3. . Данная программа написана на языке MATLAB, версия 5.3.

```
function [x_p_y,x_m_y]=butterfly(x,y)


x_p_y=x+y;
x_m_y=x-y;

return;

clc; clear;
% -- the program for Fast Hadamard Transfrorm for N=8 --
% - S - is input vector --
% - S3 - is output vector --
```

```
S=[1 2 3 4 5 6 7 8];



% -- stage 1 --

[S1(1),S1(5)]=butterfly(S(1),S(5));
[S1(2),S1(6)]=butterfly(S(2),S(6));
[S1(3),S1(7)]=butterfly(S(3),S(7));
[S1(4),S1(8)]=butterfly(S(4),S(8));

% -- stage 2 --

[S2(1),S2(3)]=butterfly(S1(1),S1(3));
[S2(2),S2(4)]=butterfly(S1(2),S1(4));
[S2(5),S2(7)]=butterfly(S1(5),S1(7));
[S2(6),S2(8)]=butterfly(S1(6),S1(8));

% -- stage 3 --

[S3(1),S3(2)]=butterfly(S2(1),S2(2));
[S3(3),S3(4)]=butterfly(S2(3),S2(4));
[S3(5),S3(6)]=butterfly(S2(5),S2(6));
[S3(7),S3(8)]=butterfly(S2(7),S2(8));
```

Рис.1 Дискретные функции

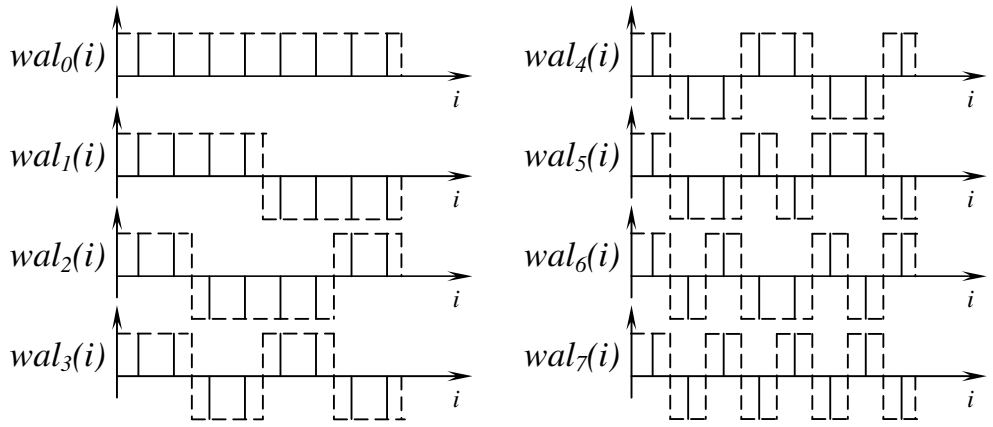
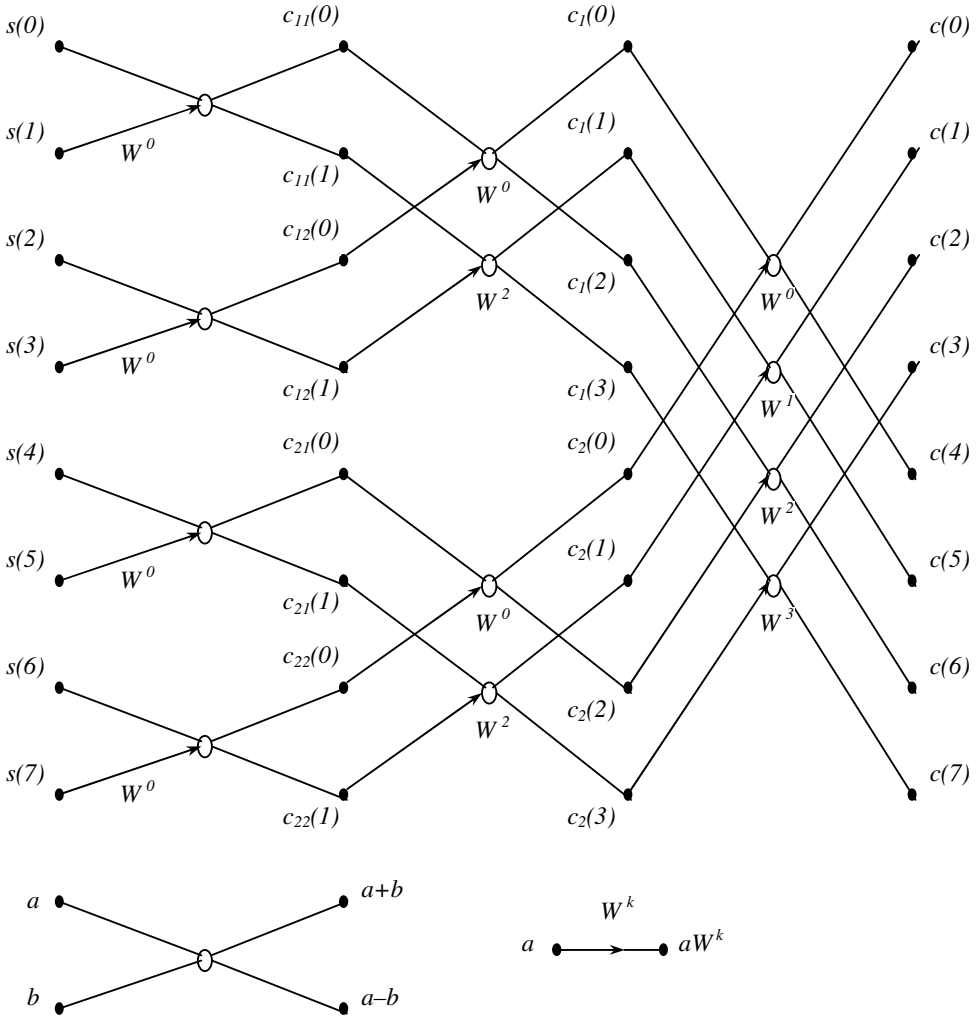


Рис.2 Граф алгоритма БПФ



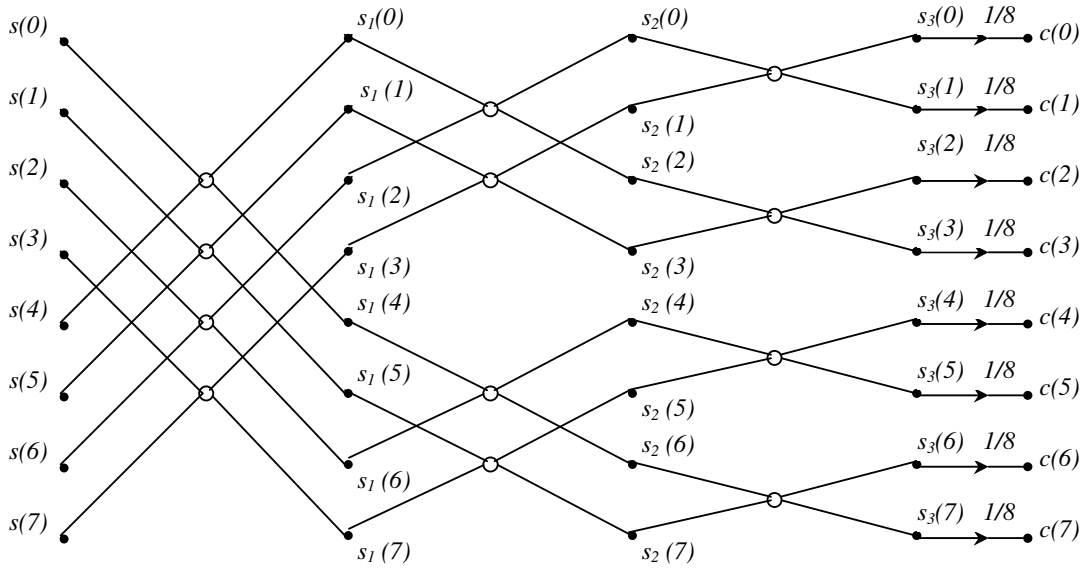


Рис.3. Граф алгоритма БПА..