Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»

Факультет вечернего, заочного и дистанционного обучения
Кафедра ЭВМ

Контрольная работа
по дисциплине «Цифровая обработка сигналов и изображений»

студента 5 курса 500503 учебной группы
Авсеева С.П.

Минск 2009

Содержание

1 Практическая часть

1.1 Реализация алгоритма БПФ

Исходные данные: функция $y = cos(x) + sin(3x)$, $N = 32$

Алгоритм был реализован на языке ruby с использованием gnuplot для построения графиков.

```
#!/usr/bin/ruby
# encoding: utf-8

require 'plot'
require 'complex'

def fft(a, sign = 1)
  return a if a.size == 1
  even = fft(a[0...a.size/2], sign)
  odd = fft(a[a.size/2..-1], sign)

  n = a.size
  wn = Complex(Math.cos(2 * Math::PI / n), sign * Math.sin(2 * Math::PI / n))
  y = []
  w = Complex(1)
  (0...a.size/2).each do |j|
    y[j] = even[j] + w*odd[j]
    y[j + a.size/2] = even[j] - w*odd[j]
    w *= wn
  end
  y
end

def sort(a)
  return a if a.size == 1
  r = [[],[]]
  a.each_with_index { |p, i|  r[i%2] << p }
  sort(r[0]) + sort(r[1])
end

plot("cos(x) + sin(3*x)", "[0:2*pi]", "images/ruby-source.eps")

N = 31
x, a = 0, []
step = 2*Math::PI / N
until x > 2*Math::PI
  a << Complex(Math.cos(x) + Math.sin(3*x))
  x += step
end

y1 = fft(sort(a))
data = [(0..N).to_a, y1.map{|i| i.abs}]
plot(data, "[0:#{N}]", "images/ruby-fft.eps")
plot(data, "[0:#{N}]", "images/ruby-fft-stem.eps", "stem")

y2 = fft(sort(y1), -1)
data = [(0..N).to_a, y2.map{|i| i/y1.size}]
plot(data, "[0:#{N}]", "images/ruby-ifft.eps")
plot(data, "[0:#{N}]", "images/ruby-ifft-stem.eps", "stem")
```

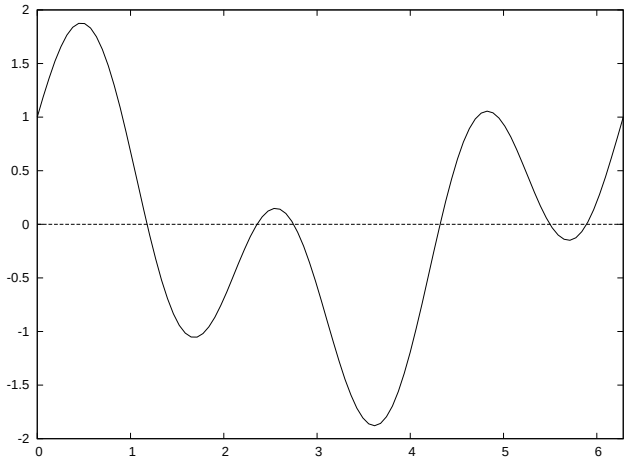


График исходной функции

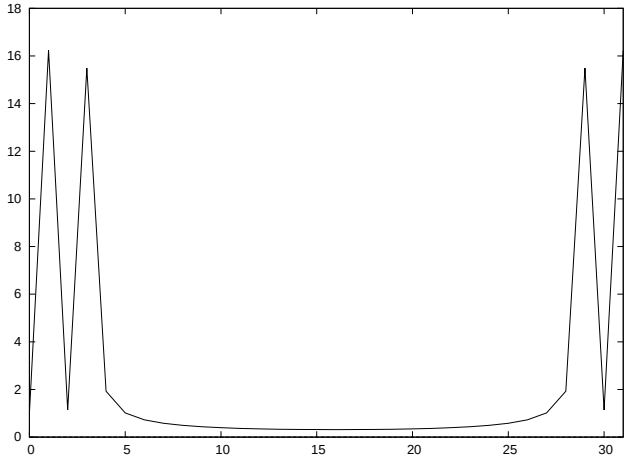
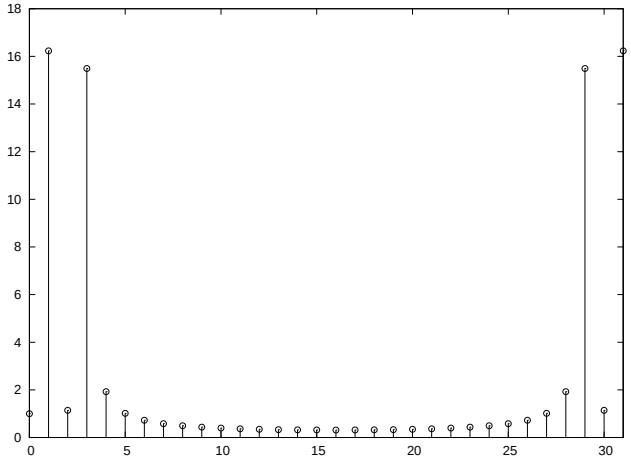


График по результатам прямого БПФ



Дискретное представление прямого БПФ

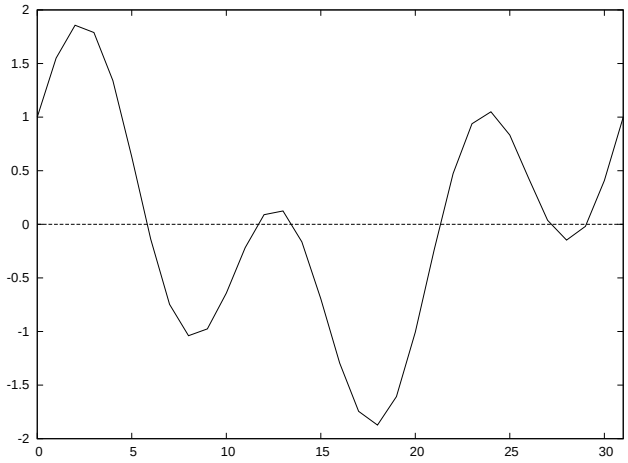
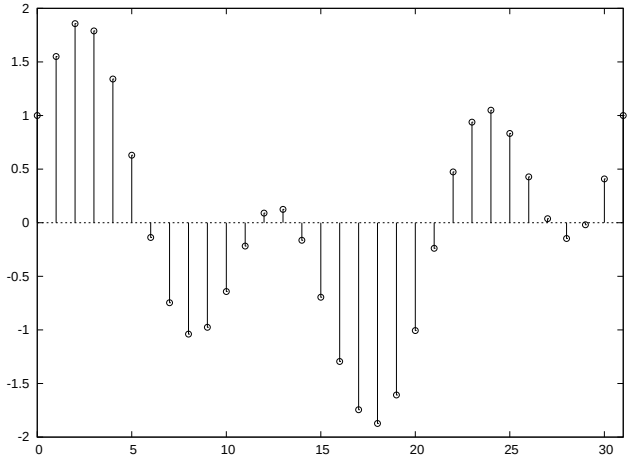


График по результатам обратного БПФ



Дискретное представление обратного БПФ

1.1.1 Модель на Octave

Octave — это свободная альтернатива MathLab, поддерживающий большую часть его синтаксиса, функций и библиотек.

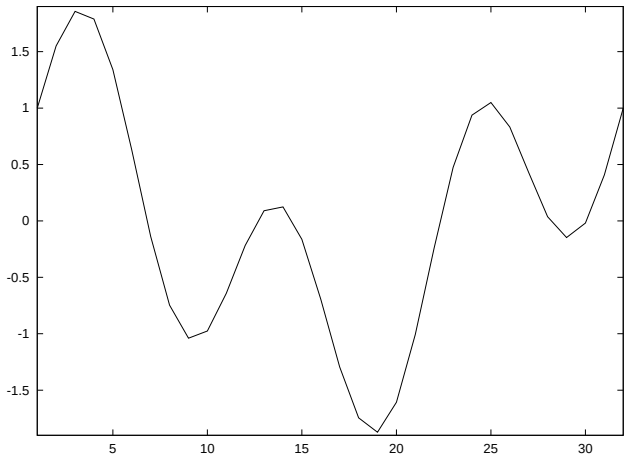


График исходной функции

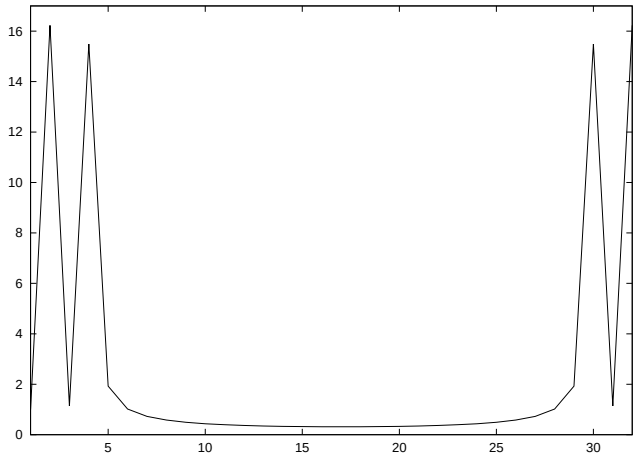
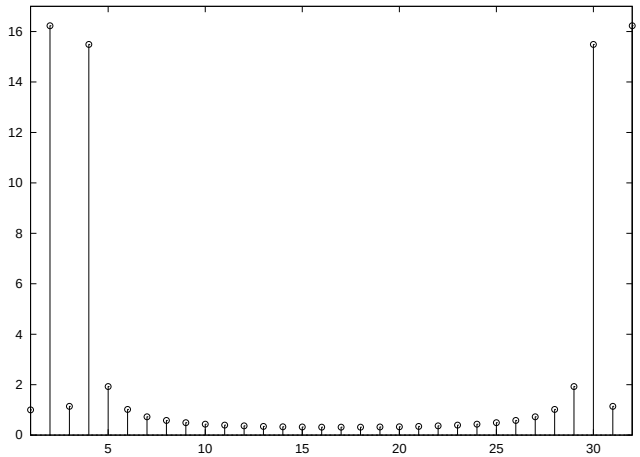


График по результатам прямого БПФ



Дискретное представление прямого БПФ

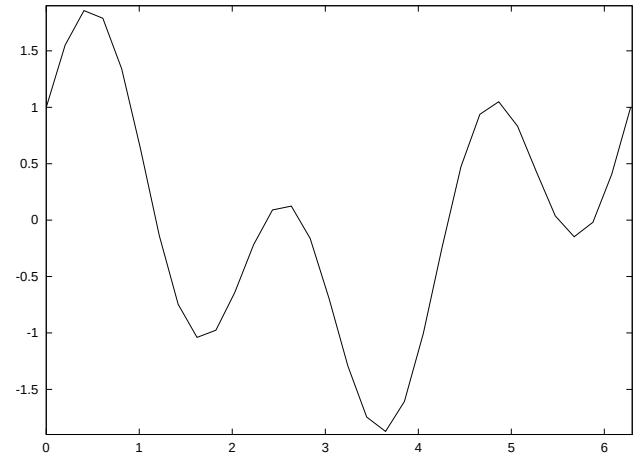


График по результатам обратного БПФ

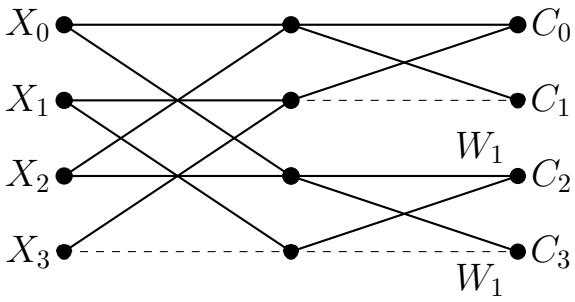
1.2 Граф-схема алгоритма БПФ для $N = 4$

Определим степени W, т.к. $N = 4$, то $l = 0, 1, ..., N/2 - 1 = 0, 1$. Тогда получим множество $S_1 = 0, 1$, после инверсии и перевода в десятичную систему это множество останется прежним, значит имеем W_0, W_1.

Т.к. $N = 4$, то число итераций для БПФ $r = \overline{1,2}$, и r итерация состоит из 2^{r-1} групп.

Вычислим $C_x(k)$, $X(m) = 0, 1, 2, 3$; для этого выразим последовательность 0,1,2,3 в двоичной форме 00, 01, 10, 11; сделаем двоичную инверсию каждому члену последовательности: 00, 10, 01, 11 и запишем в десятичной форме: 0, 2, 1, 3. Таким образом, получим $C_x(0)$, $C_x(2)$, $C_x(1)$, $C_x(3)$.

Следовательно граф-схема алгоритма БПФ будет иметь вид:



1.3 Аналитический вывод множества функций Уолша

Вывести множество функций Уолша для $N = 8$.

1.3.1 Множество функций Уолша, упорядоченное по Уолшу (по частоте)

Используя символ $\oplus$ для обозначения операции поразрядного сложения по модулю 2, способ построения функций Уолша можно выразить аналитически для любого $N = 2^r$ в виде следующего соотношения:

$$wal(n,t) = \prod_{k=1}^{r} [r_k(t)]^{n_{r-k+1} \oplus n_{r-k}} \tag{1.1}$$

Для $N = 2^r = 2^3 = 8$ произведение состоит из трёх сомножителей:

при $k = 1$, $[r_1(t)]^{n_3 \oplus n_2}$,
при $k = 2$, $[r_2(t)]^{n_2 \oplus n_1}$,
при $k = 3$, $[r_3(t)]^{n_1 \oplus n_0}$.

Для перемножения функций Радемахера можно использовать код Грея. Представим номера функций в двоичном виде и коде Грея

N	Двоичный код	Код Грея
0	000	000
1	001	001
2	010	011
3	011	010

N	Двоичный код	Код Грея
4	100	110
5	101	111
6	110	101
7	111	100

Тогда функции Уолша будут иметь вид вид:

$$
\begin{aligned}
wal(0,t) &= r_1^0(t) \cdot r_2^0(t) \cdot r_3^0(t) = 1 \\
wal(1,t) &= r_1^1(t) \cdot r_2^0(t) \cdot r_3^0(t) = r_1(t) \\
wal(2,t) &= r_1^1(t) \cdot r_2^1(t) \cdot r_3^0(t) = r_1(t) \cdot r_2(t) \\
wal(3,t) &= r_1^0(t) \cdot r_2^1(t) \cdot r_3^0(t) = r_2(t) \\
wal(4,t) &= r_1^0(t) \cdot r_2^1(t) \cdot r_3^1(t) = r_2(t) \cdot r_1(t) \\
wal(5,t) &= r_1^1(t) \cdot r_2^1(t) \cdot r_3^1(t) = r_1(t) \cdot r_2(t) \cdot r_3(t) \\
wal(6,t) &= r_1^1(t) \cdot r_2^0(t) \cdot r_3^1(t) = r_1(t) \cdot r_3(t) \\
wal(7,t) &= r_1^0(t) \cdot r_2^0(t) \cdot r_3^1(t) = r_3(t)
\end{aligned}
$$

Если обозначить функции Радемахера с помощью $\{1, -1\}$, получим

$$
R_8 = \begin{bmatrix}
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1
\end{bmatrix}
$$

Тогда функции Уолша будут выглядеть так

$$
W_8 = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1
\end{bmatrix}
$$

1.3.2 Множество функций Уолша, упорядоченное по Адомару

При $N = 2^n$ матрица Адамара может быть получеа с помощью соотношения

$$
H_{2^n} = \begin{bmatrix}
H_{2^{n-1}} & H_{2^{n-1}} \\
H_{2^{n-1}} & -H_{2^{n-1}}
\end{bmatrix}; H_0 = [1]
$$

Для $N = 8$ матрица Адамара будет иметь вид:

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

Каждая строка этой матрицы является функцией Уолша.

1.4 Преобразование двоичного кода в код Грея и наоборот

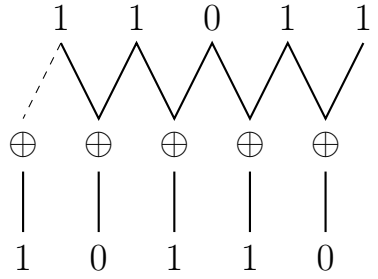
Пусть $g_{n-1}g_{n-2}\dots g_2g_1g_0$ кодовое слово в n-разрядном двоичном коде Грея, соответствующее двоичному числу $b_{n-1}b_{n-2}\dots b_2b_1b_0$. Тогда g_i может быть получена как

$$g_i = b_i \oplus b_{i+1}, \quad 0 \le i \le n-2 g_{n-1} = b_{n-1}$$

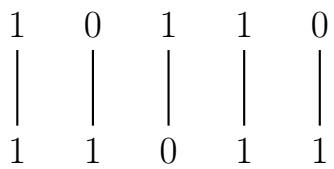
Преобразование кода Грея в двоичный код начинается с цифры самого левого разряда и движения вправо, принимая $b_i = g_i$, если число единиц, предшествующих g_i, четно и $b_i = \overline{g_i}$, если число единиц, предшествующих g_i , нечетно. При этом нулевое число единиц считается четным.

По заданию необходимо перевести число 11011 в код Грея и обратно.

Преобразование в код Грея:



Преобразование из кода Грея:



2 Теоретическая часть

2.1 Двумерное ДПФ

Дискретное преобразование Фурье можно обобщить на случай многих измерений, причем наиболее полезным оказывается обобщение на случай двух измерений, поскольку оно широко пименяется при обработке изображений. Двумерное ДПФ определяется следующим образом:

$$C_{xx}(k_1, k_2) = \frac{1}{N_1 N_2} \sum_{m_2=0}^{N_2-1} \sum_{m_1=0}^{N_1-1} X(m_1, m_2) W_1^{k_1 m_1} W_2^{k_2 m_2}, \tag{2.1}$$

где $W = e^{-i2\pi/N}$ и m_i, k_i изменяются в пределах от 0 до $(N_i - 1)$, $i = 1, 2$. Массив данных образует матрицу $[X(m_1, m_2)]$ размером $(N_1 \times N_2)$, т.е.

$$[X(m_1, m_2)] = \begin{bmatrix} X(0,0) & X(0,1) & \dots & X(0, N_2-1) \\ X(1,0) & X(1,1) & \dots & X(1, N_2-1) \\ \dots & \dots & \dots & \dots \\ X(N_1-1,0) & X(N_1-1,1) & \dots & X(N_1-1, N_2-1) \end{bmatrix} \tag{2.2}$$

Рассмотрим в выражении (2.1) внутреннюю сумму, которая определяется как

$$\frac{1}{N_1} \sum_{m_1=0}^{N_1-1} X(m_1, m_2) W_1^{k_1 m_1} = \frac{1}{N_1} [X(0, m_2) + X(1, m_2) W_1^{k_1} + \dots$$
$$\dots + X(N_1 - 1, m_2) W_1^{k_1 (N_1 - 1)}]. \tag{2.3}$$

Из выражения (2.3) следует, что правая часть представляет собой ДПФ каждлшл столбца матрицы данных $[X(m_1, m_2)]$. Поэтому введем обозначения

$$\frac{1}{N_1} \sum_{m_1=0}^{N_1-1} X(m_1, m_2) W_1^{k_1 m_1} = C_x(k_1, m_2). \tag{2.4}$$

Коэффициенты $C_x(k_1, m_2)$ в выражении (2.4) можно записать в форме матрицы $[C_x(k_1, m_2]$ размером $(N_1 \times N_2)$:

$$[C_x(k_1, m_2)] = \begin{bmatrix} C_x(0,0) & C_x(0,1) & \dots & C_x(0, N_2-1) \\ C_x(1,0) & C_x(1,1) & \dots & C_x(1, N_2-1) \\ \dots & \dots & \dots & \dots \\ C_x(N_1-1,0) & C_x(N_1-1,1) & \dots & C_x(N_1-1, N_2-1) \end{bmatrix} \tag{2.5}$$

В результате подстановки (2.3) в (2.1) получаем

$$C_{xx}(k_1, k_2) = \frac{1}{N_2} \sum_{m_2}^{N_2-1} C_x(k_1, m_2) W_2^{k_2 m_2} \tag{2.6}$$

или

$$C_{xx}(k_1, k_2) = \frac{1}{N_2}[C_x(k_1, 0) + C_x(k_1, 1) W_2^{k_2} + \ldots \\ \ldots + C_x(k_1, N_2 - 1) W_2^{k_2(N_2-1)}].$$

Это означает, что коэффициенты $C_{xx}(k_1, k_2)$ получаются путём вычисления ДПФ каждой строки матрицы $[C_x(k_1, m_2)]$, определённой выражением (2.5). В результате получается множество из $N_1 N_2$ коэффициентов, которые могут быть также записаны в виде матрицы

$$[C_{xx}(k_1, m_2)] = \begin{bmatrix} C_{xx}(0,0) & C_{xx}(0,1) & \ldots & C_{xx}(0, N_2-1) \\ C_{xx}(1,0) & C_{xx}(1,1) & \ldots & C_{xx}(1, N_2-1) \\ \ldots & \ldots & \ldots & \ldots \\ C_{xx}(N_1-1,0) & C_{xx}(N_1-1,1) & \ldots & C_{xx}(N_1-1, N_2-1) \end{bmatrix} \tag{2.7}$$

2.2 Кратномасштабный анализ

Кратномасштабный анализ (multiresolution analysis), который представляет собой процесс декомпозиции сигнала на различных частотах и различном разрешении. КМА позволяет получить хорошее разрешение по времени (плохое по частоте) на высоких частотах и хорошее разрешение по частоте (плохое по времени) на низких частотах.Этот подход становится особенно эффективным, когда сигнал имеет высокочастотные компоненты короткой длительности и протяженные низкочастотные компоненты.