МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению практических занятий
по дисциплине
«Цифровые сигнальные процессоры (DSP)»
для студентов направления подготовки
11.04.04 «Электроника и наноэлектроника»
Направленность (профиль)
«Интеллектуальные программируемые системы и устройства»

Ставрополь
2026

Содержание

ВВЕДЕНИЕ

1. Цель и задачи освоения дисциплины (модуля)

Цель освоения дисциплины: формирование у студентов профессиональных компетенций направленных на расчет, проектирование и конструирование интеллектуальных программируемых систем и устройств с использованием цифровых сигнальных процессоров.

Задачи освоения дисциплины:

1. Систематизация фундаментальных знаний по теории и практике цифровой обработки сигналов

2. Формирование системы знаний, умений и навыков по проведению расчетов схем и устройств различного функционального назначения на основе цифровых сигнальных процессоров

3. Формирование представлений по конструкции устройств аналоговой и цифровой электроники бытового и промышленного назначения на основе цифровых сигнальных процессоров, а также программирует основные алгоритмы цифровой обработки одномерных и двумерных сигналов

2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ПК-1. Способен проводить исследование, расчет и проектирование устройств, приборов и систем электронной техники в соответствии с техническим заданием	ИД-1$_{ПК-1}$: Выполняет расчет электронных приборов, схем и устройств различного функционального назначения в области электроники и наноэлектроники.	Выполняет расчет схем и устройств различного функционального назначения на основе цифровых сигнальных процессоров
	ИД-3$_{ПК-1}$: Разрабатывает конструкцию отдельных модулей, электронных приборов и устройств различного функционального назначения	Разрабатывает конструкцию устройств аналоговой и цифровой электроники бытового и промышленного назначения на основе цифровых сигнальных процессоров, а также программирует основные алгоритмы цифровой обработки одномерных и двумерных сигналов
ПК-2. Способен разрабатывать специализированное программное обеспечение	ИД-1$_{ПК-2}$ Интересуется современными тенденциями развития программных	Способен к самостоятельному поиску и анализу информации по тенденциям

интеллектуальных программируемых систем и устройств	средств и языков программирования	развития программных средств и языков программирования, используемых для цифровой обработки сигналов
	ИД-2$_{ПК-2}$ Использует теоретические знания и практические навыки, позволяющие строить алгоритмы, анализировать их работу при разных входных данных и реализовать их при помощи современных языков программирования	Способен применять теоретические знания и практические навыки для разработки алгоритмов цифровой обработки сигналов и их реализации с использованием цифровых сигнальных процессоров.
	ИД-3$_{ПК-2}$ Использует языки программирования высокого уровня и профессиональные системы программирования, инструментальные средства при решении профессионально- прикладных задач в профессиональной сфере	Анализирует техническую документация и выбирает языки программирования и архитектуру микропроцессорной системы для разработки интеллектуальных программируемых систем и устройств
ПК-6 Способен использовать знания о системах интернета вещей	ИД-2$_{ПК-6}$ Определяет требования к системам интернета вещей в зависимости от поставленной задачи по их применению	Успешно определяет требования к системам интернета вещей в зависимости от поставленной задачи по их применению
	ИД-3$_{ПК-6}$ Демонстрирует навыки моделирования и расчета в системах интернета вещей	Успешно демонстрирует навыки моделирования и расчета в системах интернета вещей

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1
РЕШЕНИЕ РАЗНОСТНЫХ УРАВНЕНИЙ С ПРИМЕНЕНИЕМ Z-ПРЕОБРАЗОВАНИЯ

Цель занятия

Выработать практические умения и приобрести навыки решения разностных уравнений с применением Z-преобразования

Задание

1. Повторить материал лекционного занятия
2. Решить разностные уравнения первого порядка
3. Решить разностные уравнения второго и выше порядков

Указания по выполнению задания

1. Повторение материала лекционного занятия

Повторение производится на самостоятельной подготовке каждым студентом самостоятельно путем изучения текстов лекций темы №1 и литературы Л1 с.4-43. В результате повторения обучаемые должны знать ответы на вопросы контрольно-тренажной работы (собеседования).

ВОПРОСЫ КОНТРОЛЬНО-ТРЕНАЖНОЙ РАБОТЫ

1. Классификация сигналов
2. Дать определение линейной системы
3. Линейная система с постоянными параметрами
4. Необходимое и достаточное условие физической реализуемости и устойчивости дискретных систем
5. Разностные уравнения
6. Общие сведения о частотных характеристиках цифровых устройств
7. Соотношение между частотными характеристиками непрерывных и дискретных систем
8. Свойства Z-преобразования

2. Решение разностных уравнений первого порядка

В качестве примера рассмотрим разностное уравнение первого порядка

$$y(n) = x(n) + ay(n-1) \quad . \qquad (1)$$

Пусть на вход поступает последовательность $x(n) = e^{j\omega n} u_{-1}(n)$. Чтобы найти одностороннее Z-преобразование $y(n)$, умножим обе части равенства (1) на z^{-n} и просуммируем от 0 до ∞:

$$\sum_{n=0}^{\infty} y(n) z^{-n} = \sum_{n=0}^{\infty} x(n) z^{-n} + a \sum_{n=0}^{\infty} y(n-1) z^{-n} .$$

Воспользуемся свойством Z-преобразования, связанным с задержкой последовательности, рассмотренным в предыдущей лекции. Имеем

$$Y(z) = X(z) + az^{-1} Y(z)$$

откуда

$$Y(z) = \frac{X(z)}{1 - az^{-1}} .$$

Так как $H(z) = {Y(z)}/{X(z)}$ получим $H(z) = {1}/{1 - az^{-1}}$.

Путем замены вида $z = e^{j\omega}$ получим комплексную частотную функцию линейной дискретной системы первого порядка:

$$H(e^{j\omega}) = \frac{1}{1 - ae^{-j\omega}} . \qquad (2)$$

В соответствии с формулой Эйлера $e^{j\omega} = \cos\omega + j\sin\omega$, поэтому выражение (2) можно переписать следующим образом

$$H(e^{j\omega}) = \frac{1}{1 - a\cos\omega + ja\sin\omega} ,$$

откуда $\left|H(e^{j\omega})\right| = \dfrac{1}{\sqrt{1 - 2a\cos\omega + a^2}}$,

$$\arg\left(H(e^{j\omega})\right) = arctg \frac{a\sin\omega}{1 - a\cos\omega} .$$

Задание № 1. Выполняется преподавателем или одним из наиболее подготовленных студентов у доски.

Решить разностное уравнение $y(n) = x(n) + 0.8y(n-1)$

Далее обучаемые самостоятельно выполняют задания №2 и № 3

Задание № 2

Решить разностное уравнение $y(n) = 0.5x(n) + y(n-1)$

Задание №3

Решить разностное уравнение $y(n) = 0.6x(n) + 0.3y(n-1)$

3. Решение разностных уравнений второго и выше порядков

В общем случае разностное уравнение $L-$го порядка имеет вид

$$y(n) = \sum_{i=0}^{L} a_i x(n-i) - \sum_{i=1}^{L} b_i y(n-i). \qquad (3)$$

Вычисляя односторонние Z-преобразования от обеих частей уравнения (3), получим

$$Y(z) = \sum_{i=0}^{L} a_i z^{-i} X(z) - \sum_{i=1}^{L} b_i z^{-i} Y(z).$$

Теперь можно получить выражение для $Y(z)$ через $X(z)$ и определить $H(z)$, а взяв обратное Z-преобразование, найти отклик $y(n)$.

Задание № 4.(Выполняется преподавателем)

Решить разностное уравнение

$y(n) = x(n) - 3x(n-1) + 0.4y(n-1) - 0.1y(n-2)$.

Чтобы найти одностороннее Z-преобразование $y(n)$, умножим обе части равенства (1) на z^{-n} и просуммируем от 0 до ∞, в результате получим

$$Y(z) = X(z) - 3z^{-1}X(z) + 0.4z^{-1}Y(z) - 0.1z^{-2}Y(z),$$

откуда

$Y(z) = \dfrac{X(z)\left(1 - 3z^{-1}\right)}{1 - 0.4z^{-1} + 0.1z^{-2}}$, так как $H(z) = \dfrac{Y(z)}{X(z)}$ получим

$$H(z) = \frac{1 - 3z^{-1}}{1 - 0.4z^{-1} + 0.1z^{-2}}.$$

Заменим параметр Z на $e^{j\omega}$:

$$H(j\omega) = \frac{1 - 3e^{-j\omega}}{1 - 0.4e^{-j\omega} + 0.1e^{-j2\omega}} =$$

$$= \frac{1 - 3\cos\omega + j3\sin\omega}{1 - 0.4\cos\omega + 0.1\cos 2\omega + j0.4\sin\omega - j0.1\sin 2\omega}.$$

Отсюда

$$|H(j\omega)| = \frac{\sqrt{1 - 6\cos\omega + 9}}{\sqrt{(1 - 0.4\cos\omega + 0.1\cos 2\omega)^2 + (0.4\sin\omega - 0.1\sin 2\omega)^2}},$$

$$\arg(H(j\omega)) = arctg\frac{3\sin\omega}{1 - 3\cos\omega} - arctg\frac{0.4\sin\omega - 0.1\sin 2\omega}{1 - 0.4\cos\omega + 0.1\cos 2\omega}.$$

Задание №5

Решить разностное уравнение
$y(n) = 2x(n) - x(n-1) + 0.8y(n-1) + y(n-2).$

Задание №6

Решить разностное уравнение $y(n) = x(n) + 0.8y(n-1) + 2y(n-3).$

Для выставления оценки за работу на практическом занятии и для выявления глубины усвоения материала, кроме проверки отчета, обучаемому могут быть заданы один или два вопроса из перечня вопросов коллоквиума.

Вопросы коллоквиума

1. Назначение z-преобразования
2. Определение устойчивости системы по её коэффициентам в z-плоскости
3. Дать определение амплитудно-частотной характеристики ЛПП-системы
4. Дать определение фазо-частотной характеристики ЛПП-системы
5. Изобразить АЧХ ЛПП-системы первого и второго порядков
6. Изобразить ФЧХ ЛПП-системы первого и второго порядков

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2
СТРУКТУРНЫЙ СИНТЕЗ ЦИФРОВЫХ ФИЛЬТРОВ

Цель занятия

Выработать практические умения и приобрести навыки решения задач структурного синтеза цифровых фильтров и анализа их характеристик

Задание

1.Повторить материал лекционного занятия
2.Осуществить структурный синтез цифрового фильтра по аналоговому образцу
3.Выполнить анализ временных характеристик цифрового фильтра
4.Выполнить анализ частотных характеристик цифрового фильтра

Указания по выполнению задания

1.Повторение материала лекционного занятия

Повторение производится на самостоятельной подготовке каждым студентом самостоятельно путем изучения текстов лекций темы №2 и литературы Л1 с.44-53. В результате повторения обучаемые должны знать ответы на вопросы контрольно-тренажной работы.

ВОПРОСЫ КОНТРОЛЬНО-ТРЕНАЖНОЙ РАБОТЫ

1. Цифровой фильтр. Определение, достоинства и недостатки
2. Виды полиномов, используемых для аппроксимации АЧХ фильтров
3. Каким образом происходит пересчет аналоговых частот в цифровые
4. Частотные характеристики ЦФ , формулы вычисления
5. Суть билинейного Z- преобразования
6. Временные и шумовые характеристики фильтров

2. Структурный синтез цифрового фильтра по аналоговому образцу.

Задача структурного синтеза цифрового фильтра включает: вычисление коэффициентов фильтра, определение способа соединения звеньев фильтра и построение структурной схемы фильтра.

Расчет коэффициентов фильтра можно вести прямым методом или по аналоговому прототипу. Наибольшее распространение получил метод расчета по аналоговому прототипу. Синтез по аналоговому прототипу основан на конформном отображении точек частотной характеристики аналогового фильтра из *p*- плоскости в z- плоскость. При этом может быть использовано билинейное z-преобразование, инвариантное преобразование импульсной характеристики, отображение дифференциалов и т.д. Наибольшее распростра-

нение нашел метод билинейного z-преобразования, так как кроме основного достоинства связанного с высокой степенью соответствия результатов преобразования, для этого метода создано много программных продуктов, облегчающих задачу синтеза.

Одним из таких программных продуктов является пакет Signal processing toolbox интегрированной системы инженерных и научных расчетов MATLAB.

Порядок выполнения задания практического занятия с использованием MATLAB показан на рис.1, а текст программы приведен в ПРИЛОЖЕНИИ.

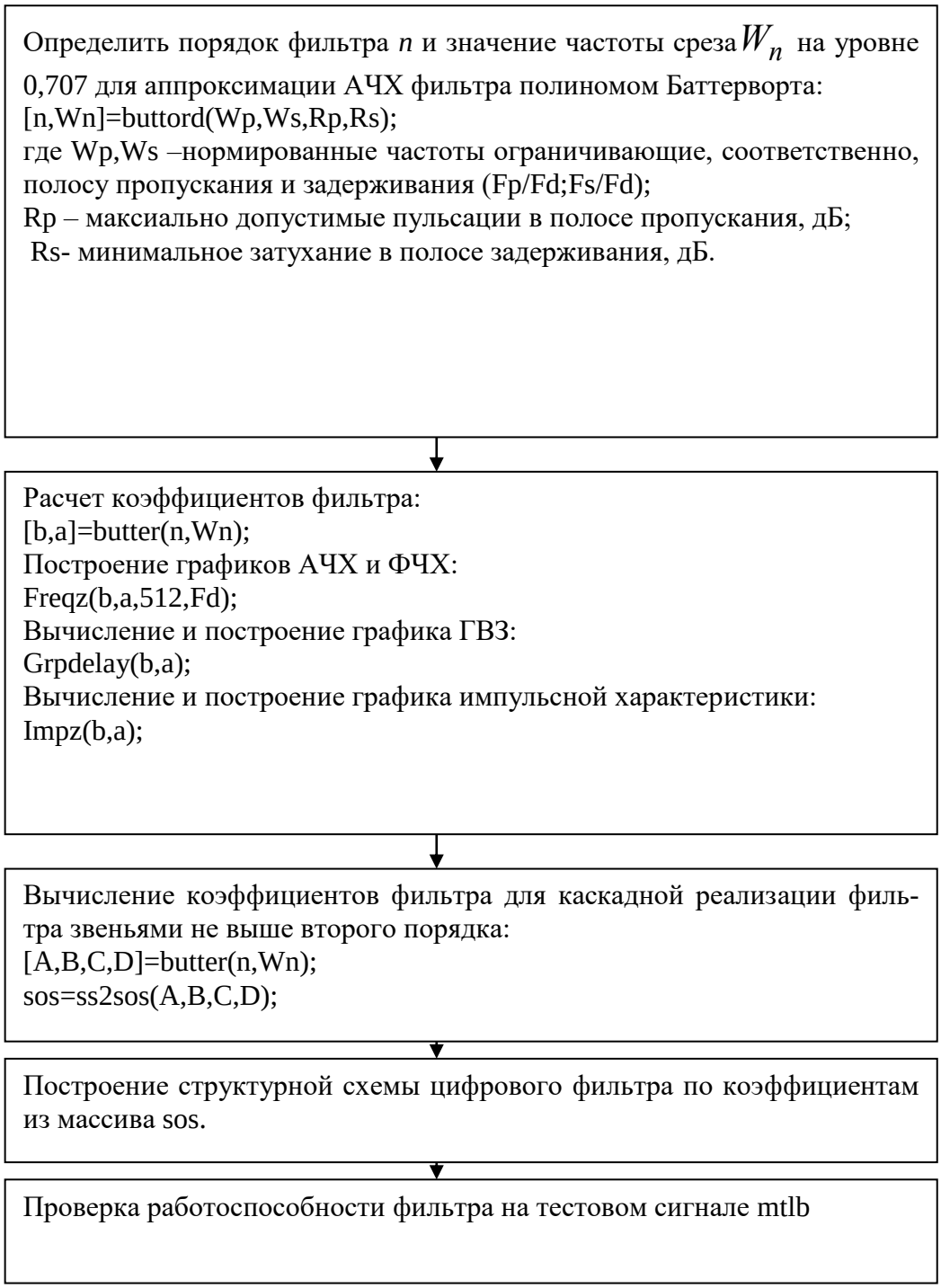


Рис.1.

Варианты заданий для самостоятельной работы приведены в табл.1. Задание выбирается по указанию преподавателя.

Таблица 1.

Параметры фильтра	Варианты заданий				
	1	2	3	4	5
Частота среза Fp, кГц	1	2	3	4	2
Частота режекции Fs, кГц	1,5	2,5	4,5	4,8	2,5
Затухания в полосе пропускания Rp, дБ	3	2,5	2	1,5	2,5
Затухания в полосе режекции Rs, дБ	12	10	10	10	10
Частота дискретизации Fd, кГц	10	20	30	40	20
Вид полинома	Баттерворта	Баттерворта	Чебышевский 1	Чебышевский 1	Эллиптический

Функции, необходимые для выполнения задания приведены в табл.2.

Таблица 2

Назначение функции	Полином Баттерворта	Полином Чебышева 1	Эллиптический полином
Определение порядка ЦФ	[n,Wn]= buttord(Wp,Ws,Rp,Rs);	[n,Wn]= cheb1ord(Wp,Ws,Rp,Rs)	[n,Wn]= ellipord(Wp,Ws,Rp,Rs)
Вычисление коэффициентов ЦФ	[b,a]=butter(n,Wn);	[b,a]=cheby1(n,Rp,Wn)	[b,a]=ellip(n,Wn)
Вычисление параметров каскадной формы	[A,B,C,D]= butter(n,Wn);	[A,B,C,D]= cheby1(n,Rp,Wn);	[A,B,C,D]=ellip(n,Wn);

Для выставления оценки за работу на практическом занятии и для выявления глубины усвоения материала, кроме проверки отчета, обучаемому могут быть заданы один или два вопроса из перечня вопросов коллоквиума.

ПРИЛОЖЕНИЕ

```
Wp=1000/10000;Ws=1500/10000;%определение входных
Rp=3;Rs=15;                %параметров
[n,Wn]=buttord(Wp,Ws,Rp,Rs);%вычисление порядка ЦФ
[b,a]=butter(n,Wn);        %определение коэффициентов
figure(1);
freqz(b,a,1024,10000);    %построение графика АЧХ и ФЧХ
[h,f]=freqz(b,a,1024,10000);
figure(2);plot(1:10000/1024:10000-1,abs(h));grid on;
ylabel('Нормированная АЧХ фильтра');
xlabel('Частота, Гц ');
figure(3);
subplot(2,1,1);grpdelay(b,a);%определение ГВЗ
subplot(2,1,2);impz(b,a);    %определение отчетов ИХ
ylabel('Импульсная характеристика фильтра');
xlabel('Номер отсчета ');
[A,B,C,D]=butter(n,Wn);%вычисление коэффициентов
sos=ss2sos(A,B,C,D);   %ЦФ для каскадной реализации
figure(4);
load mtlb;subplot(2,1,1);%проверка фильтра
plot([1:2000],mtlb(1:2000));xlabel('Входной зашумленный сигнал');
y=filter(b,a,mtlb);subplot(2,1,2);
plot([1:2000],y(1:2000));
xlabel('Результат фильтрации');
```

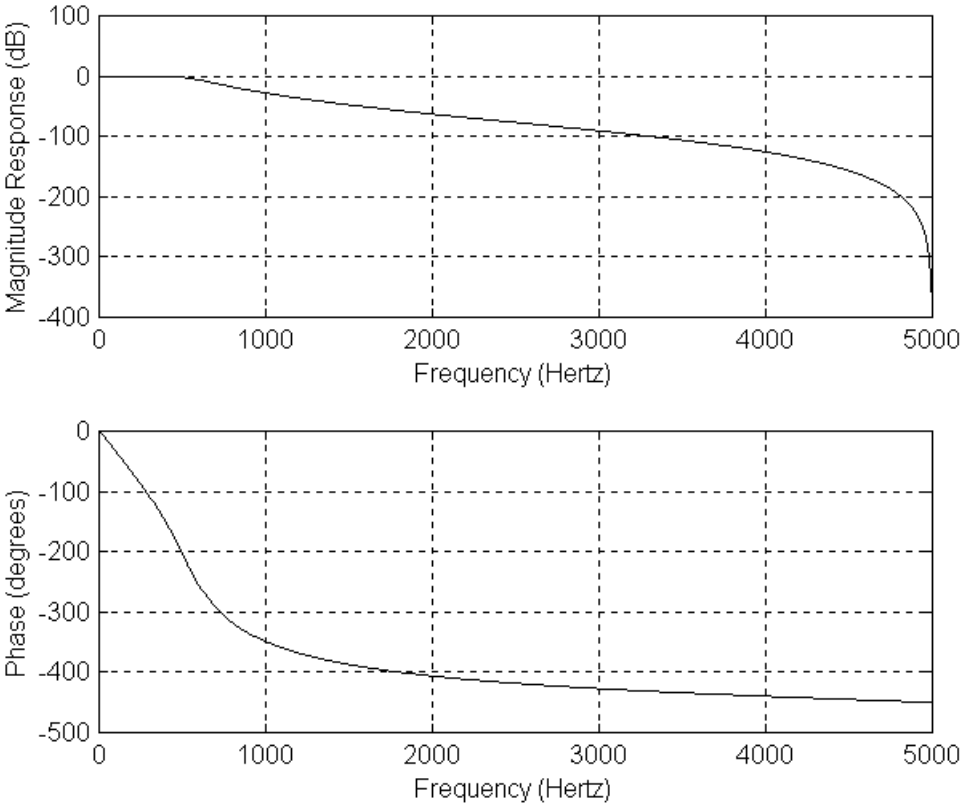

Magnitude Response (dB)
100
0
-100
-200
-300
-400
0
1000
2000
3000
4000
5000
Frequency (Hertz)
Phase (degrees)
0
-100
-200
-300
-400
-500
0
1000
2000
3000
4000
5000
Frequency (Hertz)

Рис.2

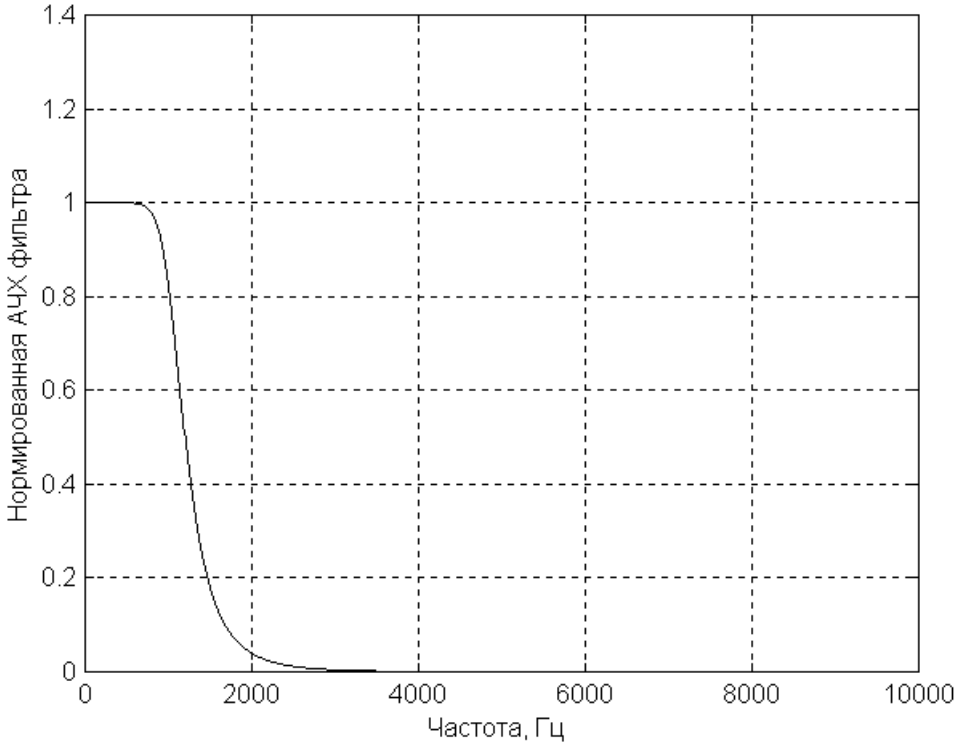

Нормированная АЧХ фильтра
Частота, Гц
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
2000
4000
6000
8000
10000

Рис.3

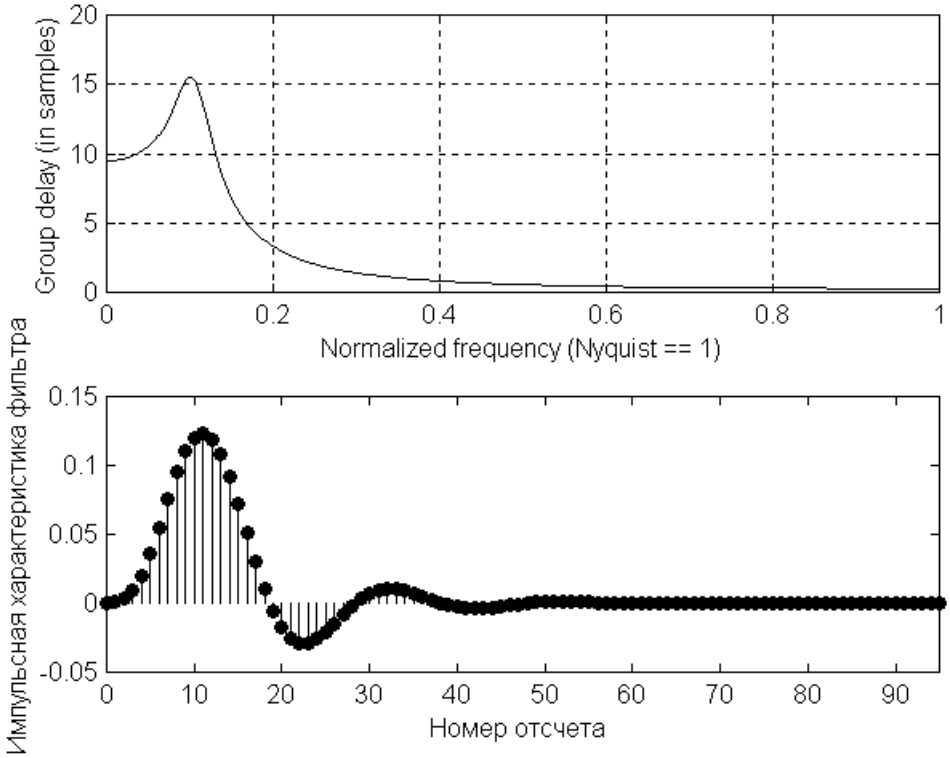

Group delay (in samples)
Normalized frequency (Nyquist == 1)
Импульсная характеристика фильтра
Номер отсчета
0
5
10
15
20
0
0.2
0.4
0.6
0.8
1
-0.05
0
0.05
0.1
0.15
0
10
20
30
40
50
60
70
80
90

Рис. 4

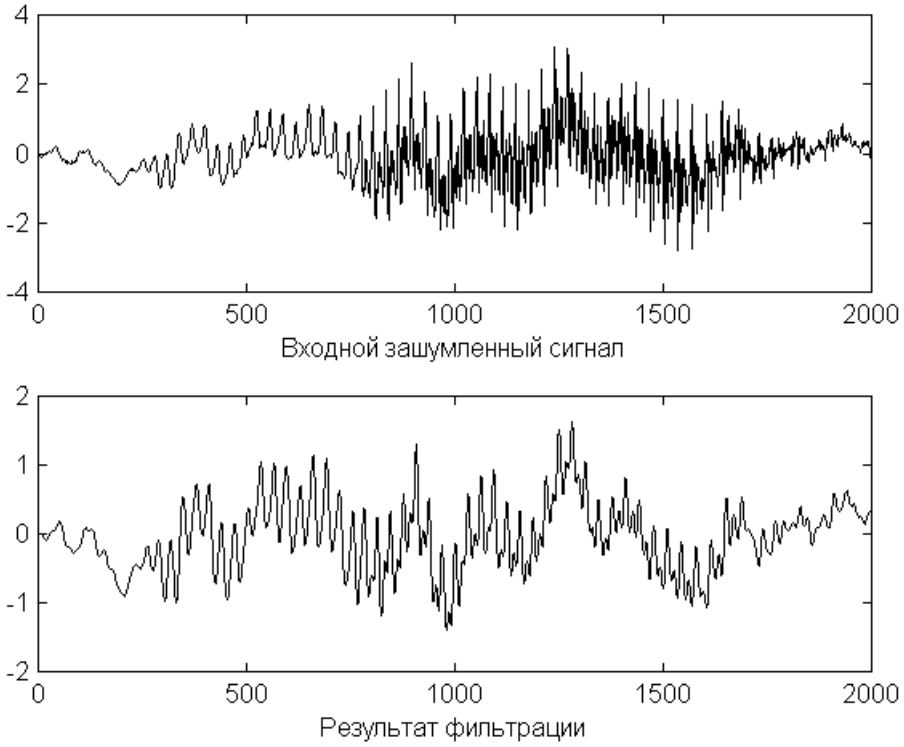

4
2
0
-2
-4
0
500
1000
1500
2000
Входной зашумленный сигнал
2
1
0
-1
-2
0
500
1000
1500
2000
Результат фильтрации

Рис.5

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3
ОПРЕДЕЛЕНИЕ ШУМОВЫХ ХАРАКТЕРИСТИК ЦИФРОВОГО УСТРОЙСТВА

Цель занятия

Выработать практические умения и приобрести навыки определения шумовых характеристик цифрового устройства и оценки влияния вычислительных ресурсов на ошибки цифровой обработки

Задание

1.Повторить материал лекционного занятия
2.Определить ошибки дискретизации
3.Определить ошибки квантования
4.Оценить влияние вычислительных ресурсов на ошибки цифровой обработки

Указания по выполнению задания

1. Повторение материала лекционного занятия

Повторение производится на самостоятельной подготовке каждым студентом самостоятельно путем изучения текстов лекций темы №3 и литературы Л1 с.103-114. В результате повторения обучаемые должны знать ответы на вопросы контрольно-тренажной работы.

ВОПРОСЫ КОНТРОЛЬНО-ТРЕНАЖНОЙ РАБОТЫ

1. Причины возникновения ошибок при цифровой обработке сигналов
2. Описание сигнала на выходе АЦП. Квантование входного сигнала с усечением
3. Описание сигнала на выходе АЦП. Квантование входного сигнала с округлением
4. Квантование результатов арифметических операций. Дисперсия шумов округления и усечения результатов операций
5. Обобщенная линейная модель цифрового фильтра для учета ошибок
7. Ошибки предельных циклов
8. Ошибки пульсаций, вызванных переполнением

1. Определение ошибок дискретизации

На этапе дискретизации входного сигнала формируется последовательность $s(n) = s(t)|_{t=nT}$, в которой отсчеты $s(n)$ представлены с неограниченной точностью.

Используя интегральную систему инженерных и научных расчетов MATLAB смоделируем входной сигнал, являющийся суммой трех гармонических колебаний с частотами 50, 150 и 250 Гц и случайной аддитивной компоненты с нулевым средним значением (рис.1)

```
x=[];xd=[];
n=0;hag=7;
for t=0:0.001:0.1;
n=n+1;
x(n)=sin(2*pi*50*t)+sin(2*pi*150*t)+sin(2*pi*250*t)*2*randn;
end;
figure(1);
plot(x(1:100));xlabel('Входной зашумленный сигнал x(nT)');
```

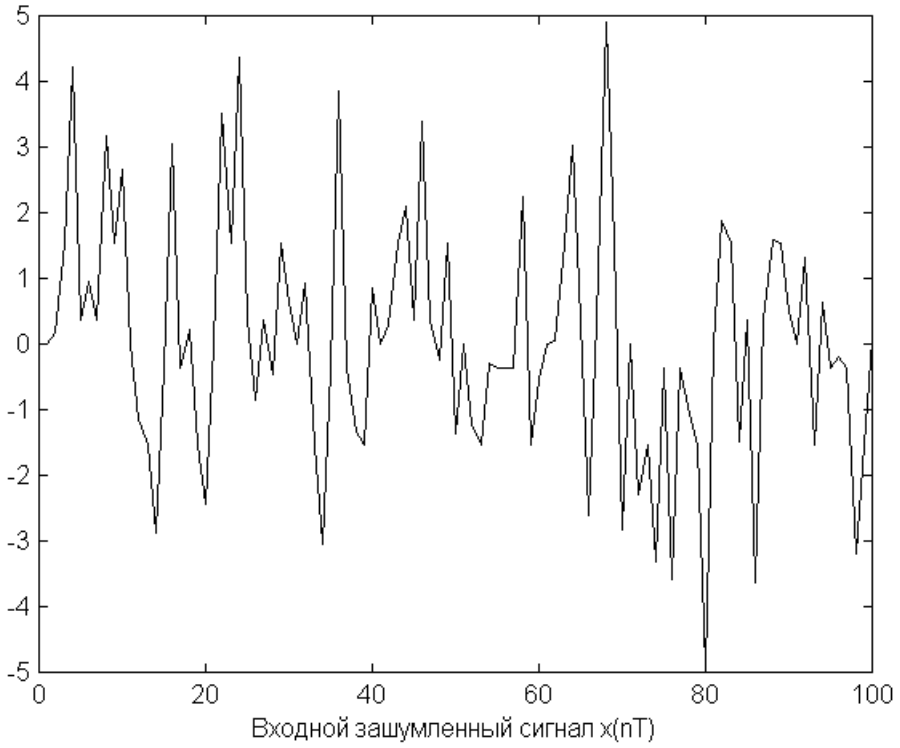


Рис.1

Период формирования входных отсчетов в четыре раза превышает период максимальной частоты гармонических колебаний, составляющих входной сигнал и равен $\Delta t = 0{,}001$ с.

Изменяя шаг дискретизации $hag = Td / \Delta t = \{1,2,3,4,5,6,7,8,9\}$ определить значение ошибки дискретизации и построить график зависимости дисперсии ошибки дискретизации от периода дискретизации.

Для двух значений периода дискретизации ($hag = 1,2$) изобразить в отчетах графики входного сигнала, дискретных отсчетов входного сигнала и результат восстановления входного сигнала при использовании интерполятора нулевого порядка.

Текст программы расчетов приведен ниже. Набор данного текста осуществляется вслед за текстом модели входного сигнала.

```
figure(2);
k1=1;k2=0;
for k=1:n;
   if k1==1; xd(k)=x(k);k2=k2+1;k1=0;else xd(k)=xd(k-
1);k2=k2+1;end;
   if k2==hag; k1=1;k2=0;end;
end;
subplot(3,1,1);plot(x(1:20));
subplot(3,1,2);plot(1:20,xd(1:20),'bo',1:20,x(1:20),'b--');
subplot(3,1,3);stairs(xd(1:20));
sum=0;
for k=1:n;
   sum=sum+(xd(k)-x(k))^2;
end;
sigmd=sqrt(sum/n)
```

Пример графиков входного сигнала, дискретных отсчетов входного сигнала и результат восстановления входного сигнала при использовании интерполятора нулевого порядка для $hag = 2$ прведен на рис.2.

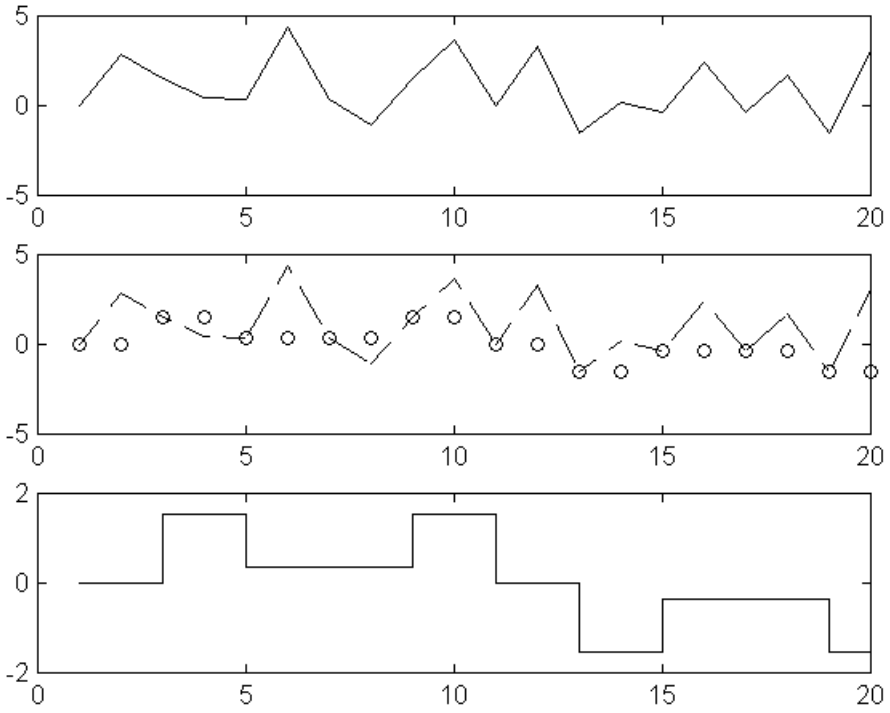


Рис.2

По результатам расчетов заполнить таблицу 1, построить график.

Таблица 1

hag	1	2	3	4	5	6	7	8	9
σ_{∂}									

2. Определение ошибок квантования.

Ошибкой квантования называется разность дискретизированного и квантованного по уровню входного сигнала $e(n) = s(n) - s_{кв}(n)$.

Дисперсия ошибок квантования зависит от шага квантования и равна

$$\sigma^2_{кв.вх} = \frac{Q^2}{12}.$$

Значение шага квантования определяется диапазоном изменения входного сигнала и разрядностью АЦП

$$Q = \frac{U_{\max} - U_{\min}}{2^n - 1},$$

где $U_{\max}, U_{\min}$ соответственно, максимальное и минимальное значения входного сигнала; n разрядность АЦП.

Выполнить расчет дисперсии ошибки квантования и построить график зависимости $\sigma^2_{кв.вх}(n)$ с использованием программы приведенной ниже по вариантам заданным преподавателем из таблицы 2.

```
Umax=10;Umin=0.01;
for ii=4:16;
   nb=ii;
   Q=((Umax-Umin)/(2^nb-1));
   sigmk(ii)=Q^2/12;
end;
plot(4:16,sigmk(4:16));grid on;
xlabel('разрядность АЦП');
ylabel('дисперсия шума квантования');
```

Таблица 2

Номер варианта	1	2	3	4
$U_{\max}, U_{\min}$	100, 0.5	25, 0.05	42.5, 0.8	34.1, 0.01

3. Оценка влияния вычислительных ресурсов на ошибки цифровой обработки

Выполним расчет дисперсии шума квантования результатов арифметических операций для различных значений разрядности регистров арифметического устройства (умножителей и сумматоров), а также для различного количества операций умножения (порядка фильтра).

Дисперсия шума округления результатов арифметических операций вычисляется по формуле

$$\sigma_o^2 = \frac{2^{(-2b)}}{12},$$

где b - число разрядов регистров вычислителя.

Выполнить расчет дисперсии ошибки квантования и построить график зависимости $\sigma^2_{кв.о}(n)$ с использованием программы приведенной ниже.

```
for ii=4:16;
   nb=ii;
   sigmo(ii)=(2^(-nb*2))/12;
end;
plot(4:16,sigmo(4:16));grid on;
xlabel('разрядность АЦП');
ylabel('дисперсия шума округления результатов');
```

Суммарное значение дисперсии шумов округления результатов вычислений цифрового устройства обработки сигналов определяется выражением

$$\sigma^2_{\Sigma} = \sum_{n=0}^{N-1} \sigma^2_{кв.о} h^2(n),$$

где N -порядок фильтра, $h(n)$ - отчеты импульсной характеристики.

Программа расчета суммарной дисперсии приведена ниже.

```
Wp=1000/10000;Ws=1500/10000;so=[];sp=[];
Rp=3;Rs=15;
[n,Wn]=buttord(Wp,Ws,Rp,Rs);
[b,a]=butter(n,Wn);
nn=impz(b,a);
for N=2:4:16;
   sum=0;
   for ii=4:16;
   nb=ii;
   sig0=(2^(-nb*2))/12;
   for is=1:N;
      sum=sum+nn(is,1)^2;
   end;
   sp(N)=sum;
   s0(ii)=sig0*sum;
end;
plot(4:16,s0(4:16));grid on;hold on;
ylabel('Дисперсия ошибок');
xlabel('Разрядность вычислителя');
end;
hold off;
```

Выполнить расчеты и построить график зависимости суммарной дисперсии ошибок округления результатов арифметических операций для варианта цифрового фильтра синтезированного на практическом занятии №2.

Выполнить анализ полученных зависимостей и записать в отчет выводы по практическому занятию

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4
ВЫПОЛНЕНИЕ ДИСКРЕТНОЙ СВЕРТКИ И ДИСКРЕТНОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

Цель занятия

Выработать практические умения и приобрести навыки решения задач с применением дискретной свертки сигналов и дискретного преобразования Фурье

Задание

1.Повторить материал лекционного занятия
2.Решить задачи с применением дискретной свертки.
3.Решить задачи с применением ДПФ.

Указания по выполнению задания

1. Повторение материала лекционного занятия

Повторение производится на самостоятельной подготовке каждым студентом самостоятельно путем изучения текстов лекций темы №4 и литературы Л1 с.46-69. В результате повторения обучаемые должны знать ответы на вопросы контрольно-тренажной работы.

ВОПРОСЫ КОНТРОЛЬНО-ТРЕНАЖНОЙ РАБОТЫ

1. Дискретная кольцевая свертка сигналов.
2. Дискретная линейная свертка сигналов.
3. Алгоритм выполнения дискретной секционированной свертки методом перекрытия с накоплением.
4. Алгоритм выполнения дискретной секционированной свертки методом перекрытия с суммированием.
5. Дискретное преобразование Фурье. Определение. Назначение.
6. Свойства линейности и комплексной сопряженности ДПФ.
7. Свойства сдвига, круговой свертки и корреляции ДПФ.

2. Решение задач с применением дискретной свертки.

Если заданы одномерные массивы x и y длины, которых соответственно равны $m = length(x)$ и $n = length(y)$, то свертка z - это одномерный массив длиной $m + n - 1$, k-й элемент которого определяется по формуле

$$z(k) = \sum_{j=\max(1,k+1-n)}^{\min(k,m)} x(j)y(k+1-j).$$

Синтаксис оператора свертки:

z=conv(x,y)

Пример 1. Выполнить свертку двух последовательностей $x = \{-2;1;2;1\}$ и $h = \{2;1;0;0\}$.

Программа свертки:

```
x=[-2 1 2 1];
h=[2 1 0 0];
z=conv(x,h);
subplot(3,1,1);
stem(x);grid on;
xlabel('Входной сигнал');
subplot(3,1,2);
stem(h);grid on;
xlabel('Отсчеты импульсной характеристики');
subplot(3,1,3);
stem(z);grid on;
xlabel('Свертка сигналов');
```

Графики исходных сигналов и результатов свертки приведены на рис.1.

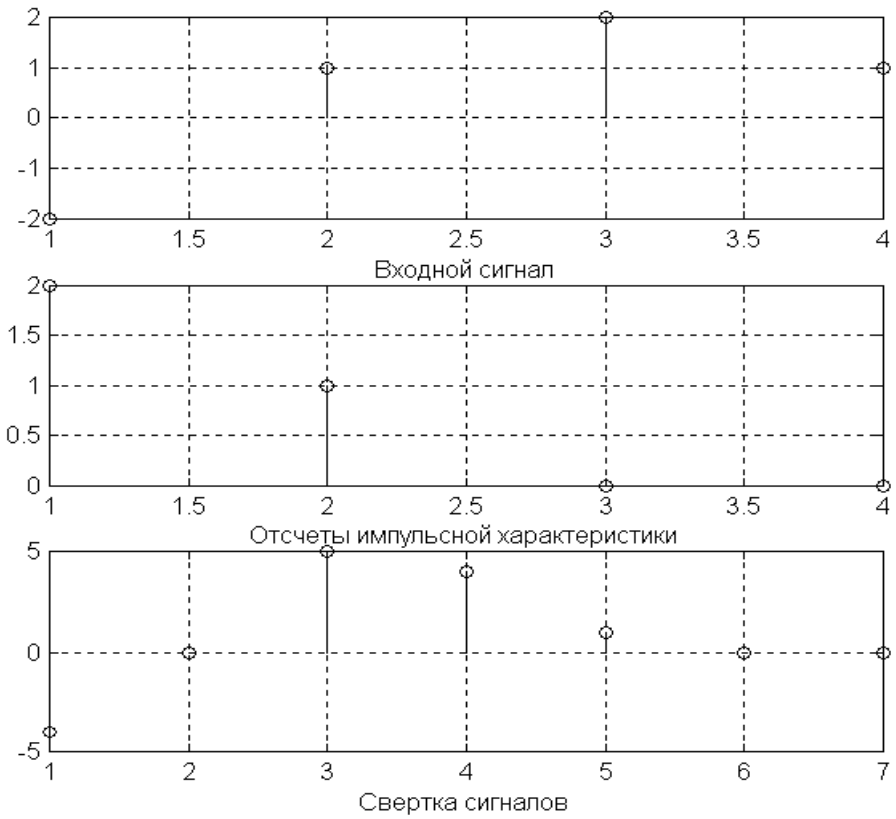


Рис.1.

Задание №1. Выполнить свертку следующих последовательностей $x = \{1;2;3;4;3;2;1\}$ и $y = \{3;2;1;0\}$. Построить графики.

Задание №2. Выполнить свертку следующих последовательностей $x = \{3;2;1;0;1;2;3\}$ и $y = \{3;2;1;0\}$. Построить графики.

Задание №3. Выполнить свертку следующих последовательностей $x = \{3;2;1;0;-1;-2;-3\}$ и $y = \{3;2;1;0\}$. Построить графики.

Пример 2. Выполнить свертку сигналов $x = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$ при $f_1 = 50; f_2 = 120; t = 0 : 0.001 : 0.5$ и $h = \sin(m)/m$ при $m = 0.1 : 0.01 : 2\pi$.

Программа выполнения свертки:

```
x=[];h=[];z=[];
t=0:0.001:0.5;
x=sin(2*pi*50*t)+sin(2*pi*120*t);
m=0.1:0.01:2*pi;
for k=1:600;
   h(k)=sin(m(k))/m(k);
end;
z=conv(x,h);
subplot(3,1,1);
stem(x(1:20));grid on;
xlabel('Входной сигнал');
subplot(3,1,2);
plot(h);grid on;
xlabel('Отсчеты импульсной характеристики');
subplot(3,1,3);
stem(z(1:20));grid on;
xlabel('Свертка сигналов');
```

Полученные графики представлены на рис.2

Задание №4. Выполнить свертку следующих последовательностей $x = \sin(2\pi f_1 t) + \cos(2\pi f_2 t)$ при $f_1 = 50; f_2 = 120; t = 0 : 0.001 : 0.5$ и $h = abs(\sin(m)/m)$ при $m = 0.1 : 0.01 : 2\pi$.

Задание №5. Выполнить свертку последовательности прямоугольных импульсов с последовательностью $h = \sin(m)/m$ при $m = 0.1 : 0.01 : 2\pi$.

Для задания входных последовательностей используйте следующие команды:

```
t=0:0.01:4*pi;
x=square(t,5);
m=0.1:0.01:2*pi;
h=sinc(m);
```

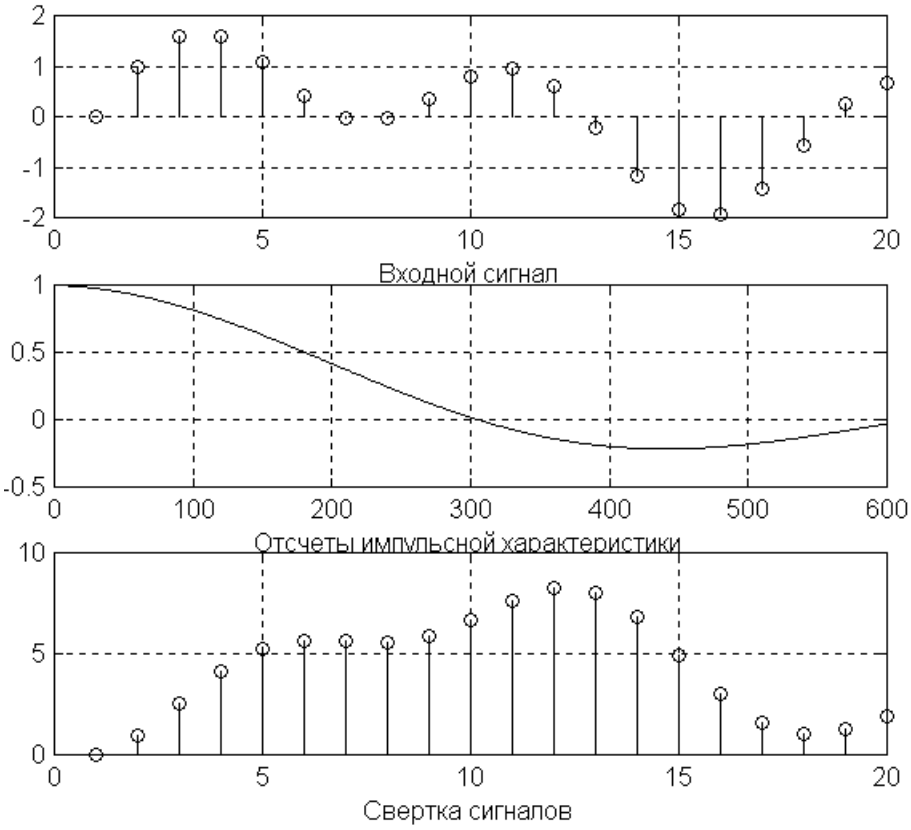


Рис.2

3. Решение задач с применением ДПФ

Дискретное преобразование Фурье используется для перевода сигнала из временной области в частотную. В составе системы Matlab для определения матрицы коэффициентов преобразования Фурье используется функция

a=dftmx(n) ;

где $n \times n$- размер матрицы.

Для вычисления ДПФ входной последовательности *x* необходимо домножить её отсчеты на коэффициенты матрицы *a.*

Пример. Выполнить ДПФ входной последовательности вида: $x = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + 2 * randn$, где $randn$ -функция генератора случайных чисел, распределенных по нормальному закону относительно 0 , при $A_1 = 2; A_2 = 5; f_1 = 70; f_2 = 170; t = 0 : 0.001 : 0.6$

Текст программы приведен ниже:

```
x=[];a=[];z=[];X=[];
t=0:0.001:0.6;
x=2*sin(2*pi*70*t)+5*sin(2*pi*170*t)+2*randn(size(t));
X=x(1:512);
a=dftmtx(512);
z=X*a/256;
subplot(2,1,1);
plot(x(1:100));grid on;
```

```
xlabel('Входной сигнал');
f=1000*(0:255)/512;
subplot(2,1,2);
plot(f,abs(z(1:256)));grid on;
xlabel('ДПФ сигналов');
```

Результаты расчетов приведены на рис.3.

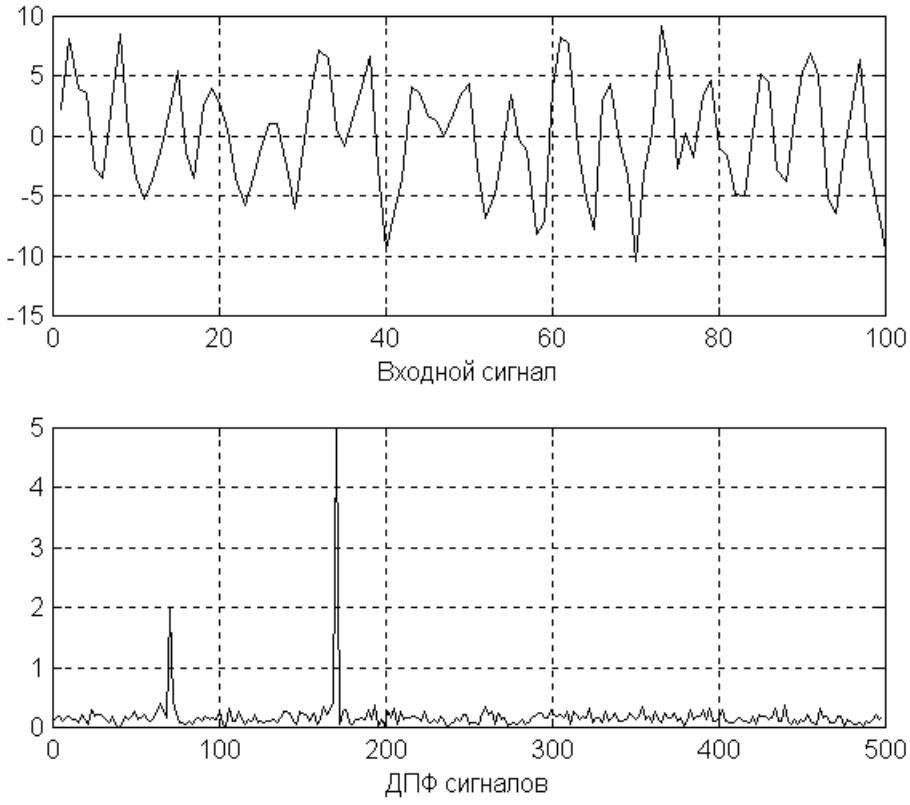


Рис. 3

Задание №6. Выполнить ДПФ последовательности $x = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + A_3 * randn$, где $randn$ -функция генератора случайных чисел, распределенных по нормальному закону относительно 0 , при $t = 0 : 0.001 : 0.6$. Значения исходных данных взять из таблицы в соответствии с вариантом, указанным преподавателем.

Таблица

Вариант	A_1	A_2	A_3	f_1	f_2
1	1	2	3	50	100
2	2	2	2	50	120
3	2	3	1	60	170
4	5	5	1	70	120
5	2	1	1	50	110

Построить графики результатов преобразования.

Задание №7. Выполнить ДПФ последовательности прямоугольных импульсов

```
t=0:0.01:4*pi;
x=square(t,5);
```

Построить графики результатов преобразования.

Задание №8. Выполнить ДПФ последовательности пилообразных импульсов

```
t=0:0.1:20*pi;
x=sawtooth(t,1);
```

Построить графики результатов преобразования.

Выполнить анализ полученных зависимостей и записать в отчет выводы по практическому занятию .

На занятии иметь:

- Конспект лекций;
- Задание на практическое занятие №4;
- Выполненное задание на самоподготовку.

При подготовке к занятию:

- Повторить лекционный материал;
- Найти ответы на вопросы контрольно-тренажной работы;
- Повторить порядок работы с программой Matlab.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5
ВЫПОЛНЕНИЕ ОРТОГОНАЛЬНЫХ ПРЕОБРАЗОВАНИЙ

Цель занятия

Выработать практические умения и приобрести навыки решения задач с применением ортогональных преобразований сигналов .

Задание

1. Повторить материал лекционного занятия
2. Решить задачи с применением ДПУ.
3. Решить задачи с применением дискретного преобразования Хаара.
4. Решить задачи с применением дискретного преобразования Гильберта.

Указания по выполнению задания

1.Повторение материала лекционного занятия

Повторение производится самостоятельно каждым студентом путем изучения текстов лекций темы №5 и литературы Л1 с.53-98. В результате повторения обучаемые должны знать ответы на вопросы контрольно-тренажной работы.

ВОПРОСЫ КОНТРОЛЬНО-ТРЕНАЖНОЙ РАБОТЫ

1. Дать определение частости. Привести пример использования.
2. Дискретное преобразование Хаара, выражение в матричной форме.
3. Функции Уолша упорядоченные по Уолшу.
4. Функции Уолша, упорядоченные по Пэли.
5. Функции Уолша, упорядоченные по Адамару.
6. Преобразование Гильберта. Назначение, запись в матричной форме.
7. Преобразование Гильберта синусоидального сигнала.

2. Решение задач с применением ДПУ

Наибольшее применение для реализации на вычислительных машинах нашло дискретное преобразование Уолша в котором функции Уолша упорядочены по Адамару. В литературе это преобразование называется дискретное преобразование Уолша-Адамара (ДПУА) или преобразование Адамара.

В составе системы инженерных и научных расчетов Matlab реализована функция H=hadamard(n) , возвращающая матрицу дискретных отсчетов ДПУА (матрицу Адамара) порядка n. Матрица Адамара порядка n>2 существует только тогда, когда n кратно 4. Реализованный в системе Matlab алго-

ритм вычисляет матрицы Адамара для тех случаев, когда величины $n, n/12, n/20$ являются степенями по основанию 2.

Например,

» H=hadamard(8)

H =

```
   1    1    1    1    1    1    1    1
   1   -1    1   -1    1   -1    1   -1
   1    1   -1   -1    1    1   -1   -1
   1   -1   -1    1    1   -1   -1    1
   1    1    1    1   -1   -1   -1   -1
   1   -1    1   -1   -1    1   -1    1
   1    1   -1   -1   -1   -1    1    1
   1   -1   -1    1   -1    1    1   -1
```

Картина линий уровня для этой матрицы напоминает ковер(рис.1):

» contour(hadamard(8))

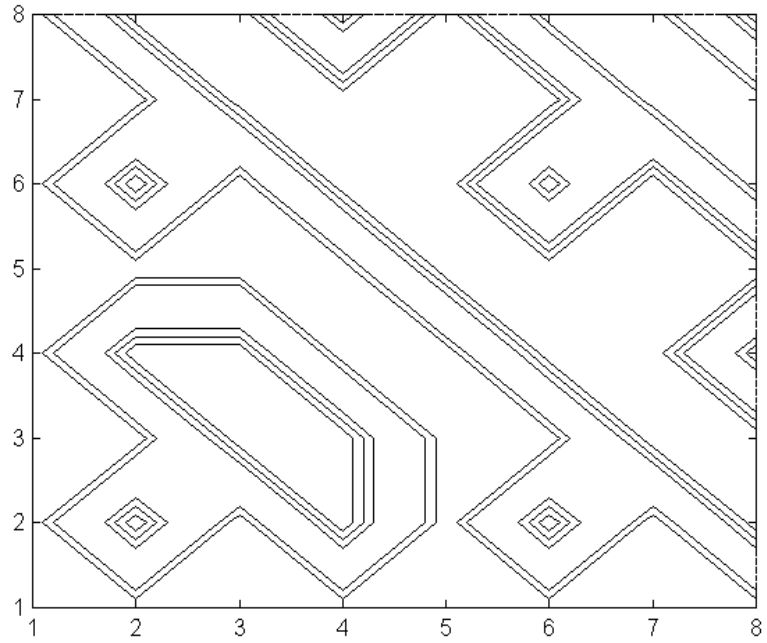


Рис.1

Использование ортогональных преобразований позволяет осуществить сжатие сигналов так как в разном базисе (системе координат) отчеты сигнала располагаются с разной дисперсией (степенью компактности). Рассмотрим пример использования ДПУА для тестового сигнала mtlb .Дисперсия входного сигнала (std(a)=0.6352). Дисперсия результата преобразования (std(y)=0.0203) . В преобразованном сигнале обнулены все отсчеты, значения которых меньше 10 процентов от максимального. Результат восстановления показывает практически полное соответствие входного сигнала и восстановленного по сжатому сигналу.

Программа модели приведена ниже. Результаты моделирования показаны на рис.2, 3.

```
load mtlb;
figure(1);
N=1024;
subplot(2,1,1);
plot(mtlb(1:N));grid on;
ylabel('Входной сигнал');
H=hadamard(N);
a=mtlb(1:N);
y=H*a;
y=y/N;
subplot(2,1,2);
plot(y);grid;
ylabel('Результат преобразования');
xlabel('Номер отсчета');
figure(2);
for ik=1:N;
   if abs(y(ik))<max(y)/10; y(ik)=0; end;
end;
b=H'*y;
subplot(2,1,1);
plot(y);grid;
ylabel('ДПУА без части отсчетов ');
subplot(2,1,2);
plot(b);grid on;
ylabel('Выходной сигнал');
xlabel('Номер отсчета');
```

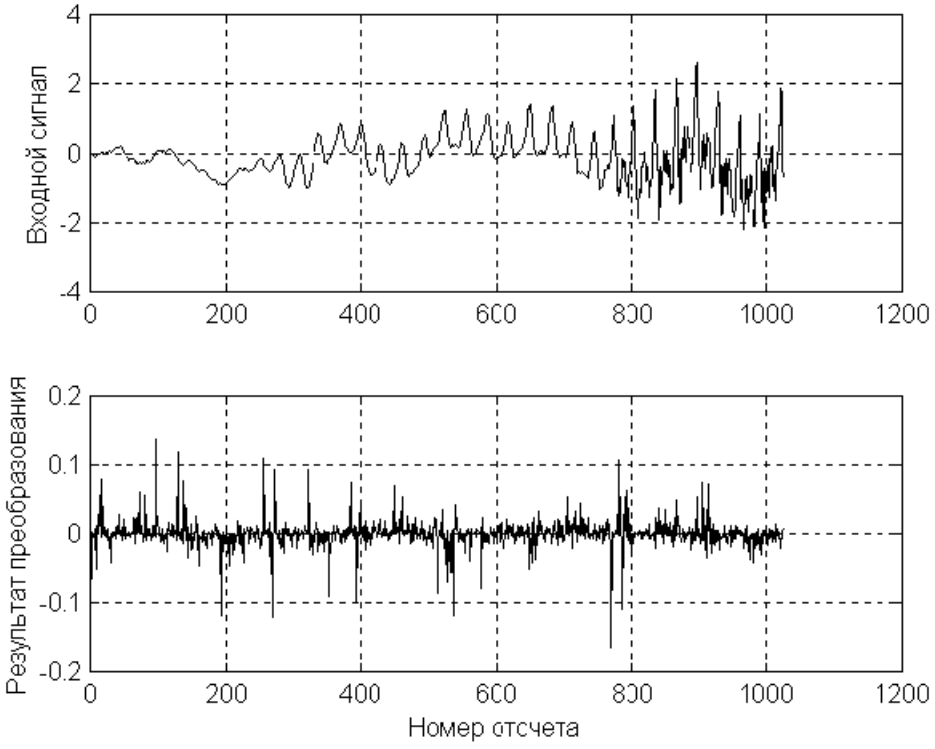


Рис. 2 Входной сигнал и результат его ДПУА

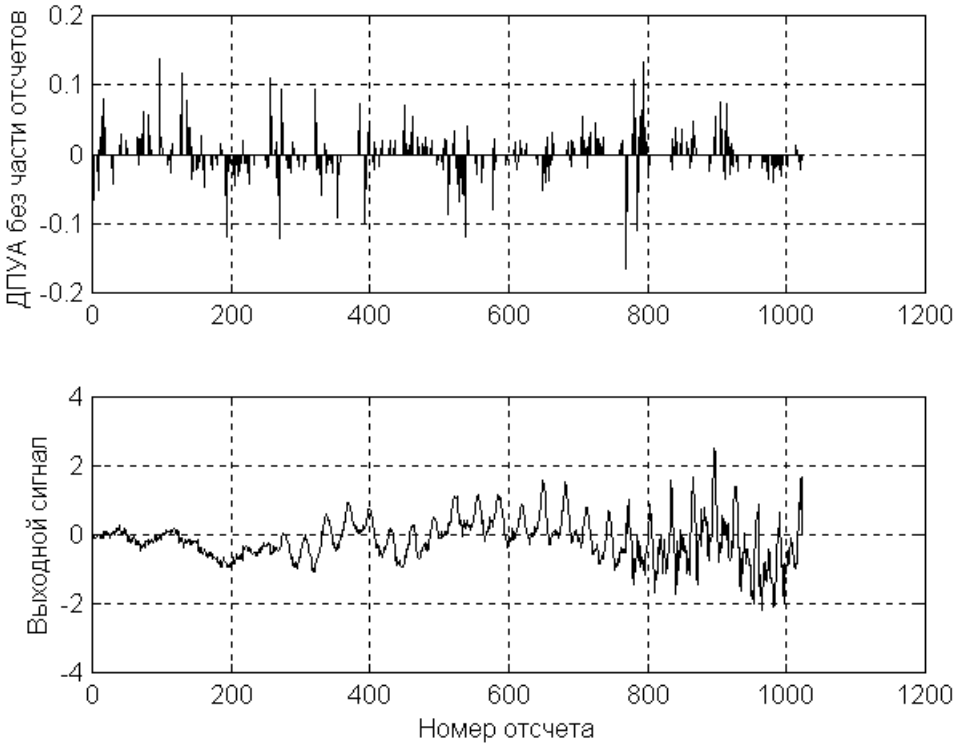


Рис. 3. ДПУА входного сигнала без отсчетов значения, которых меньше 10 процентов от максимального и результат обратного преобразования

Самостоятельно выполнить ДПУА для N=128,256,512. Сделать выводы о точности преобразования. При N=1024 изменить величину порога до 20 процентов, сделать выводы о точности восстановления входного сигнала.

3. Решение задач с применением дискретного преобразования Хаара

Система Matlab не содержит функции преобразования Хаара поэтому для выполнения преобразования необходимо вручную задать матрицу дискретных значений функции Хаара.

Ниже приведена программа прямого и обратного дискретного преобразования Хаара для последовательностей содержащих 8 отсчетов. На рис.4. показаны результирующие графики.

```
har=[1 1 1 1 1 1 1 1
   1 1 1 1 -1 -1 -1 -1
   1.4142 1.4142 -1.4142 -1.4142 0 0 0 0
   0 0 0 0 1.4142 1.4142 -1.4142 -1.4142
   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];
t=0:0.1:1;
x=r*randn(size(t));
a=x(1:8);
y=a*har;
y=y/8;
subplot(3,1,1);plot(a);grid;ylabel('Входной сигнал');
```

```
subplot(3,1,2);plot(y);grid;ylabel('Прямое ДПХ');
b=y*har';
subplot(3,1,3);plot(b);grid;ylabel('Обратное ДПХ');
```

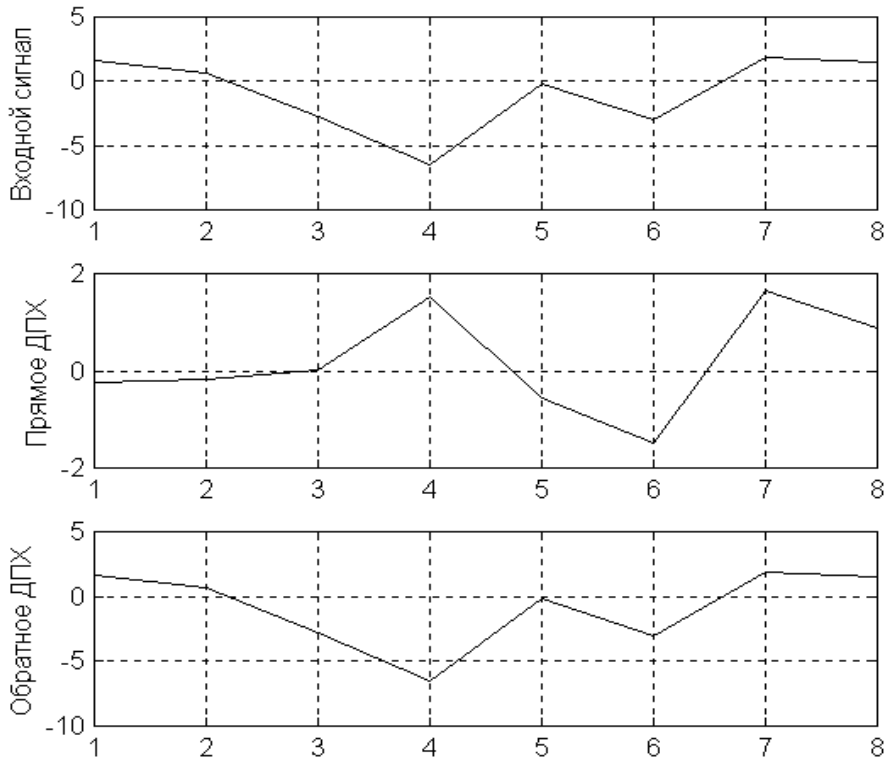


Рис.4

Ещё одним примером ортогонального преобразования является преобразование Хартли. Элементы матрицы Хартли рассчитываются следующим образом

$$O(i,j)=\sin\left(\frac{2\pi(i-1)(j-1)}{n}\right)+\cos\left(\frac{2\pi(i-1)(j-1)}{n}\right).$$

В системе Matlab создано несколько моделирующих ортогональные матрицы программ. Эти программы собраны в пакет ORTHOG . Для вызова матрицы Хартли необходимо указать

O=galley(‘orthog’,n,5);

Где n- число отсчетов в матрице, 5- номер соответствующий матрице Хартли в пакете ORTHOG .

По аналогии с преобразованием Уолша-Адамара выполним преобразование Хартли испытательного сигнала mtlb. Программа прямого и обратного преобразований приведена ниже, а результаты преобразований на рис.5 и 6. Необходимо обратить внимание на симметричное расположение отсчетов прямого преобразования Хартли. Самостоятельно выполнить ДПХ для N=128,256,512. Сделать выводы о точности преобразования. При N=1024 изменить величину порога до 20 процентов, сделать выводы о точности восстановления входного сигнала.

```
load mtlb;
figure(1);
N=1024;
subplot(2,1,1);
plot(mtlb(1:N));grid on;
ylabel('Входной сигнал');
O=gallery('orthog',N,5);
a=mtlb(1:N);
y=O*a;
subplot(2,1,2);
plot(y);grid;
ylabel('Преобразование Хартли');
xlabel('Номер отсчета');
figure(2);
for ik=1:N;
   if abs(y(ik))<max(y)/10; y(ik)=0; end;
end;
b=O'*y;
subplot(2,1,1);
plot(y);grid;
ylabel('ДПХ без части отсчетов ');
subplot(2,1,2);
plot(b);grid on;
ylabel('Выходной сигнал');
xlabel('Номер отсчета');
```

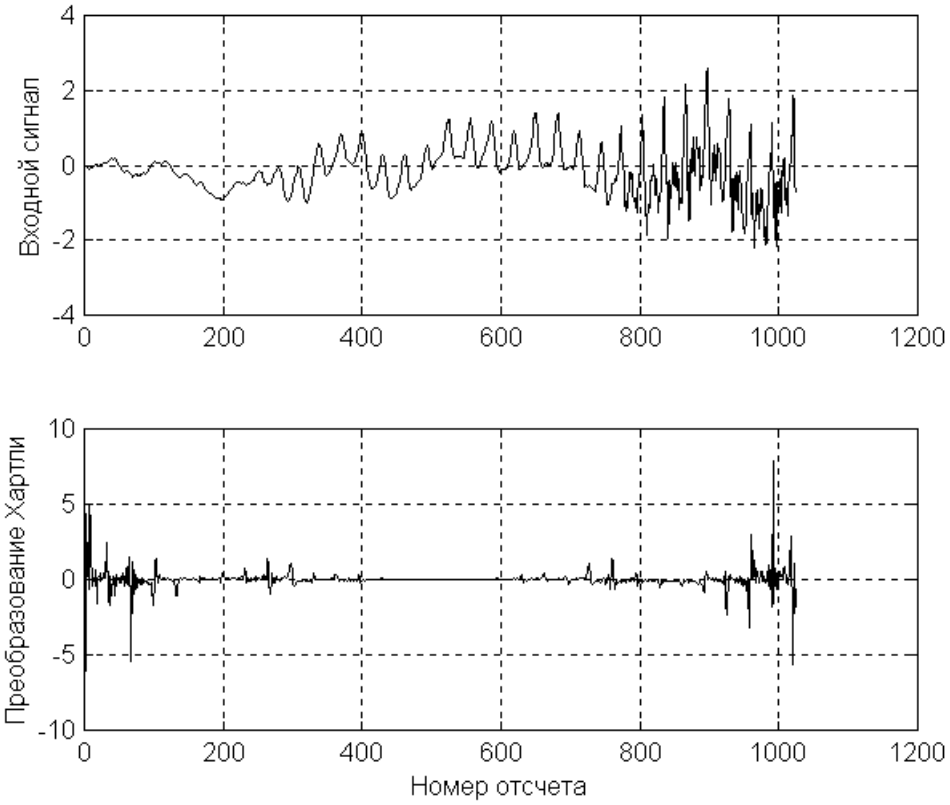


Рис.5

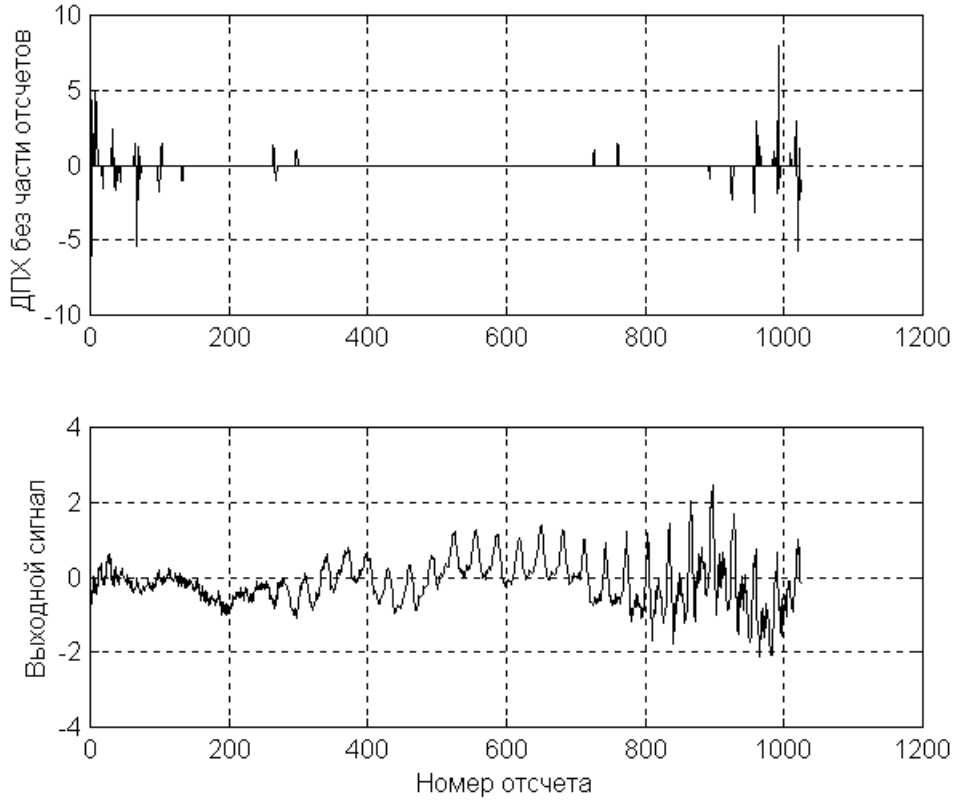


Рис.6

4. Решение задач с применением дискретного преобразования Гильберта.

Преобразование Гильберта в системе Matlab осуществляется специальной функцией

Y=hilbert(x);

Функция Y=hilbert(x) по исходной действительной последовательности x вычисляет комплексную последовательность y, которая является аналитическим сигналом. Действительная часть аналитического сигнала совпадает с исходной последовательностью, а комплексная – представляет собой её преобразование Гильберта $\widetilde{x}$. Таким образом, $y = x + j\widetilde{x}$. Преобразование Гильберта используется для вычисления мгновенных параметров временных последовательностей. Сигнал y можно представить в виде $y(t) = A(t)e^{j\Theta(t)}$. В этом случае $A(t)$ называется мгновенной амплитудой сигнала $x(t)$, а $\Theta(t)$ - мгновенной фазой. Мгновенная частота f_0 определяется из выражения $f_0 = \left(\frac{1}{2\pi}\right)\frac{d\Theta(t)}{dt}$. Для синусоиды огибающая и мгновенная частота являются константами. Мгновенная фаза изменяется по пилообразному закону, а в пределах одного периода – линейно. Ниже приведен пример программы

выполнения преобразования Гильберта испытательного сигнала mtlb. Самостоятельно выполнить преобразование Гильберта для синусоидального сигнала с амплитудой 5 вольт и частотой 50 герц дискретизированной на интервале 0:0.6 с шагом 0.001.

```
load mtlb;
subplot(3,1,1);
plot(mtlb(1:256));grid on;
ylabel('Входной сигнал');
a=mtlb(1:256);
y=hilbert(a);
subplot(3,1,2);
plot(abs(y(1:256)));
ylabel('Мгн. амплитуда');
subplot(3,1,3);
plot(angle(y(1:256)));
ylabel('Мгн. фаза');
xlabel('Номер отсчета');
```

Выполнить анализ полученных зависимостей и записать в отчет выводы по практическому занятию .

На занятии иметь:

- Конспект лекций;
- Задание на практическое занятие №5;
- Выполненное задание на самоподготовку.

При подготовке к занятию:

- Повторить лекционный материал;
- Найти ответы на вопросы контрольно-тренажной работы;
- Повторить порядок работы с программой Matlab.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6
ВЫПОЛНЕНИЕ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ

Цель занятия

Выработать практические умения и приобрести навыки решения задач с применением быстрого преобразования Фурье.

Задание

1. Повторить материал лекционного занятия
2. Решить задачи спектрального анализа с использованием алгоритма БПФ.
3. Вычислить корреляционную последовательность с использованием алгоритма БПФ.

Указания по выполнению задания

1. Повторение материала лекционного занятия

Повторение производится каждым студентам самостоятельно путем изучения текстов лекций темы №6 и литературы Л1 с.80-92. В результате повторения обучаемые должны знать ответы на вопросы контрольно-тренажной работы.

ВОПРОСЫ КОНТРОЛЬНО-ТРЕНАЖНОЙ РАБОТЫ

1. Алгоритм БПФ с прореживанием по времени, базовая операция, графическое обозначение .
2. Алгоритм БПФ с прореживанием по частоте, базовая операция, графическое обозначение.
3. Операция двоичной инверсии отсчетов, назначение, пример выполнения.
4. Определение потребных вычислительных ресурсов для реализации БПФ.
5. Алгоритмы БПФ с произвольным основанием.

2. Решение задач спектрального анализа с использованием алгоритма БПФ.

Вычисление амплитудного и фазового спектра.

Рассмотрим результат преобразования Фурье синусоидального колебания с частотой 50 герц.

```
t=0:0.001:0.6;
x=sin(2*pi*50*t);
subplot(3,1,1);
plot(x(1:500)); grid on;ylabel('signal');
Y=fft(x,512);
Ay=abs(Y)/512;
Fy=angle(Y);
```

```
subplot(3,1,2);
plot(Ay);grid on;ylabel('АЧХ');
subplot(3,1,3);
plot(Fy);grid on;ylabel('ФЧХ');
```

Результат выполнения программы приведен на рис.1.

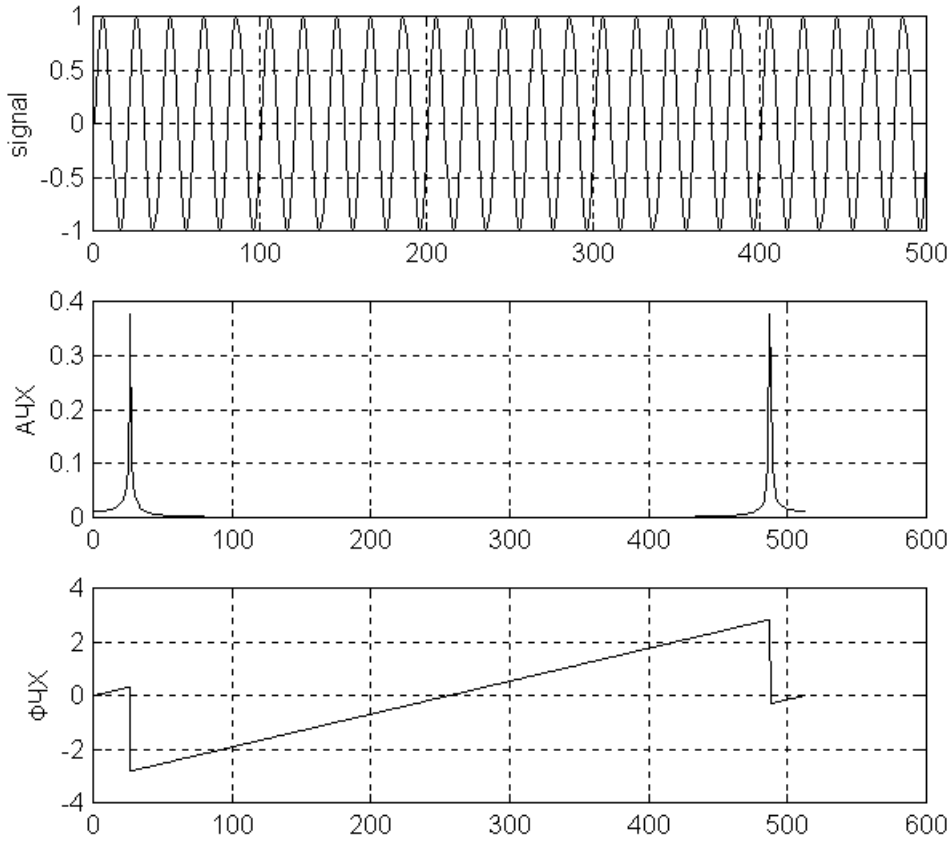


Рис.1

Из рисунка видно, что амплитудный спектр сигнала симметричен относительно середины отсчетов. Выполним преобразование выводимых графиков АЧХ и ФЧХ для вывода только первой половины отсчетов, кроме этого введем переменную f для привязки полученных отсчетов к оси частот.

```
t=0:0.001:0.6;
x=sin(2*pi*50*t);
subplot(3,1,1);
plot(x(1:500)); grid on;ylabel('signal');
Y=fft(x,512);
Ay=abs(Y)/512;
Fy=angle(Y);
f=1000*(0:255)/512;
subplot(3,1,2);
plot(f,Ay(1:256));grid on;ylabel('АЧХ');
subplot(3,1,3);
plot(f,Fy(1:256));grid on;ylabel('ФЧХ');
```

Изобразить в отчетах полученные графики.

Рассмотрим задачу определения амплитудного спектра аддитивной смеси двух гармонических колебаний с гауссовским шумом.

```
t=0:0.001:0.6;
x=sin(2*pi*50*t)+sin(2*pi*120*t);
y=x+2*randn(size(t));
subplot(2,1,1);
plot(y(1:500)); grid on;ylabel('signal');xlabel('n');
Y=fft(y,512);
Ay=abs(Y)/512;
Fy=angle(Y);
f=1000*(0:255)/512;
subplot(2,1,2);
plot(f,Ay(1:256));grid on;ylabel('АЧХ');xlabel('частота, Гц');
```

Результаты вычислений приведены на рис.2

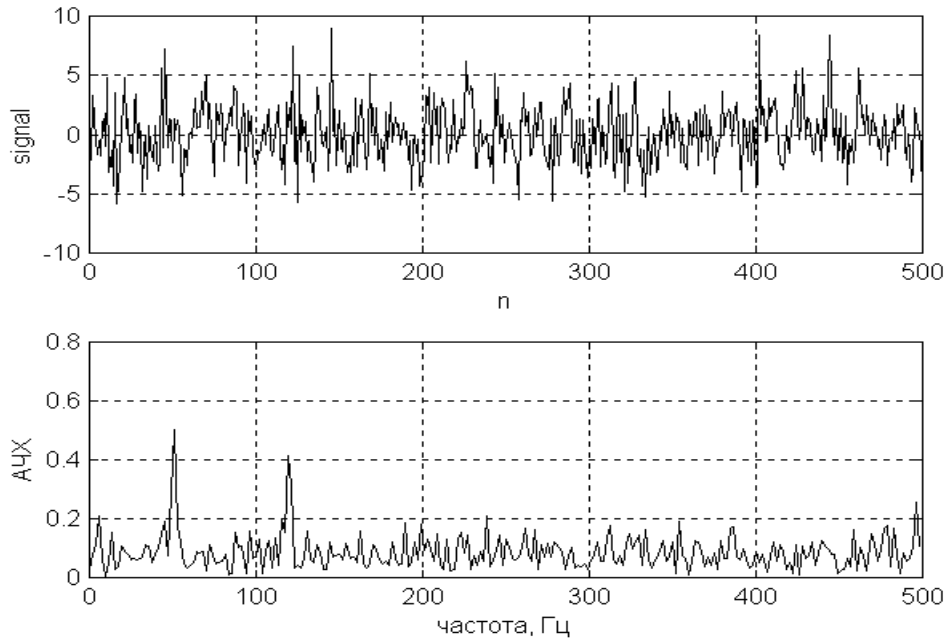


Рис.2

Задание №1. Выполнить БПФ последовательности , получить амплитудный и фазовый спектр последовательности

$x = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + A_3 * randn$, где $randn$ -функция генератора случайных чисел, распределенных по нормальному закону относительно 0 , при $t = 0 : 0.001 : 0.6$. Значения исходных данных взять из таблицы в соответствии с вариантом, указанным преподавателем.

Таблица

Вариант	A_1	A_2	A_3	f_1	f_2
1	1	2	3	50	100
2	2	2	2	50	120
3	2	3	1	60	170
4	5	5	1	70	120
5	2	1	1	50	110

Построить графики результатов преобразования.

Задание №2. Выполнить БПФ последовательности прямоугольных импульсов

```
t=0:0.01:4*pi;
x=square(t,5);
```

Построить графики АЧХ и ФЧХ.

Задание №3. Выполнить БПФ последовательности пилообразных импульсов

```
t=0:0.1:20*pi;
x=sawtooth(t,1);
```

Построить графики АЧХ и ФЧХ .

Задание №4. Выполнить БПФ последовательности пилообразных импульсов

```
t=0:0.001:0.6;
x=exp(-100*t);
```

Построить графики АЧХ и ФЧХ .

Задание №5 Используя приведенную ниже программу выполнить исследование точности и скорости вычисления БПФ для $N = 100,110,128,150,200,256,300,500,512,800,1024$.

```
Ay=[];Fy=[];
t=0:0.001:1;
x=sin(50*2*pi*t);N=256;
tic,Y=fft(x,N);toc
Ay=abs(Y)/N;
Fy=angle(Y);
subplot(3,1,1);
plot(x(1:100)); grid on;ylabel('signal');xlabel('n');
subplot(3,1,2);
plot(Ay(1:100));grid on;ylabel('АЧХ');xlabel('частота, Гц');
subplot(3,1,3);
plot(Fy(1:100));grid on;ylabel('ФЧХ');xlabel('частота, Гц');
```

Построить график зависимости времени вычисления БПФ от числа отсчетов.

3. Вычисление корреляционной последовательности с использованием алгоритма БПФ

Последовательность вычислений корреляционной последовательности с помощью прямого и обратного БПФ приведена на рис.3.

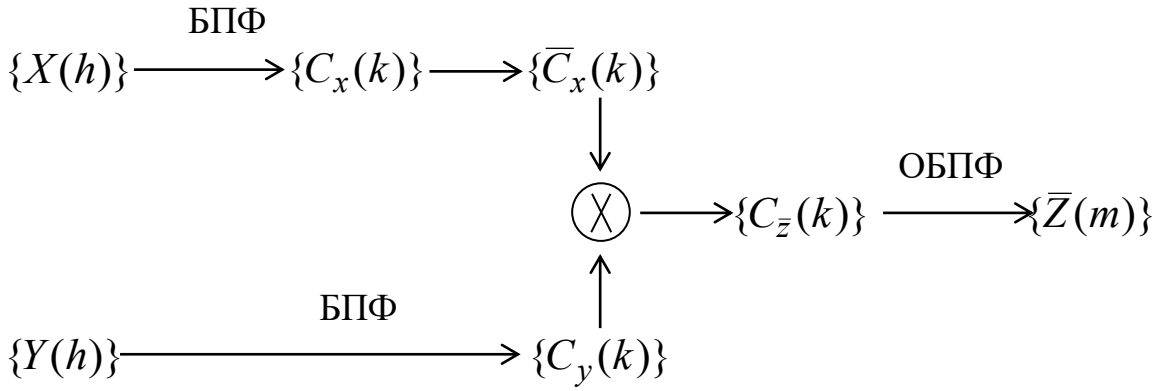


Рис.3

Программа вычисления имеет следующий вид

```
Ay=[];Fy=[];
t=0:0.001:1;
N=size(t)
for k=1:N(2);
   if k<0.25*N(2); h(k)=1;else h(k)=0; end;
end;
x=sin(50*2*pi*t);
X=fft(x,512);
H=fft(h,512);
XC=conj(X);
for k=1:512;
   Z(k)=XC(k)*H(k);
end;
z=ifft(Z);
subplot(3,1,1);
plot(x(1:500)); grid on;ylabel('signal 1');
subplot(3,1,2);
plot(h(1:500));grid on;ylabel('signal 2');
subplot(3,1,3);
plot(abs(z(1:500)));grid on;
ylabel('Корреляционная последовательность');
```

Изобразить в конспекте полученные графики.

Выполнить расчет корреляционной последовательности для последовательностей приведенных в заданиях 2,3,4. Выполнить расчет автокорреляционной последовательности любой из последовательностей для нескольких значений величины задержки.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7
ФУНКЦИОНАЛЬНЫЕ УСТРОЙСТВА ЦСП

Цель занятия

Изучить принципы построения и функциональных устройств цифровых сигнальных процессоров

Учебные вопросы

1. Интерфейсные устройства
2. Архитектура арифметических устройств
3. Структуры данных и переменные

1.Интерфейсные устройства

Интерфейс с памятью. Семейство цифровых сигнальных процессоров «ПКК Миландр» использует модифицированную Гарвардскую архитектуру, в которой память данных содержит данные, а память инструкций содержит как инструкции, так и данные. Процессор содержит ОЗУ и/или ПЗУ на кристалле, так что часть адресного пространства памяти данных и памяти инструкций находится на чипе. Эти процессоры также имеют адресное пространство загрузочной памяти вдобавок к адресным пространствам памяти данных и инструкций.

В процессорах, которые имеют внутреннюю память, внутренняя шина адреса памяти инструкций (PMA) и внутренняя шина адреса памяти данных (DMA) мультиплексированы в единую шину адреса, которая выведена наружу чипа. Также, внутренняя шина данных памяти инструкций (PMD) и внутренняя шина данных памяти данных (DMD) мультиплексированы в единую шину адреса, которая выведена наружу чипа. 16 старших разрядов внешней шины данных используются как шина DMD. Другими словами, D23-8 используется как DMD15-0.

Есть три различных адресных пространства: память данных, память инструкций и загрузочная память. Сигналы **PMS, DMS и** BMS показывают, какая из них используется. Так как шина данных и шина инструкций используют одни и те же физические линии, то если требуется более одной пересылки данных в/из внешней памяти за одну инструкцию, будет использован дополнительный процессорный цикл на исполнение команды.

Интерфейс с загрузочной памятью. Вся внутренняя память программ, или любая ее часть, может быть загружена из внешнего источника с использованием процедуры загрузки. Для взаимодействия с недорогими микросхемами EPROM, процессор загружает инструкции побайтно.

Автоматическая загрузка после сброса зависит от состояния сигнала MMAP (Memory Map Control Signal/сигнал управления картой памяти). Начальная загрузка происходит, если сигнал MMAP находится в состоянии логического нуля во время сброса процессора. Начальная загрузка может быть также инициирована программно после сброса.

Процессоры семейства, имеющие порт интерфейса с хост-процессором (HIP), могут быть загружены как с использованием интерфейса с загрузочной памятью, так и с использованием HIP (с хост-компьютера). Состояние линии BMODE определяет используемый метод (загрузка из памяти если BMODE - 0, загрузка с использовании HIP, если BMODE - 1).

Сигнал BR распознается во время начальной загрузки. Шина предоставляется после загрузки текущего байта.

Интерфейс с памятью программ. Процессор адресует 16К 24-битных слов памяти программ и до 2К слов непосредственно на кристалле. Процессор записывает 14- битный адрес на 14-битную шину адреса инструкций (PMA), которая выведена наружу кристалла для доступа к внешней памяти. Инструкции или данные передаются по 24-битной шине памяти инструкций (PMD), которая также выведена наружу. Для исполнения инструкций , которые требуют одновременно доступа к внешней памяти инструкций и к внешней памяти данных , данные из памяти инструкций читаются первыми, затем данные из памяти данных. Выход PMS (выбора памяти программ) указывает на то, что на шину адреса поступил адрес памяти инструкций.

Две управляющие линии устанавливают направление передачи данных. Сигнал RD (чтение данных) , активное состояние которого низкое, указывает на процесс чтения из памяти. Сигнал WR (запись данных), активное состояние которого тоже низкое, указывает на процесс записи в память.

Интерфейс с памятью данных. Процессор адресует 16К 16-битных слов памяти данных. Данные передаются по старшим 16 битам 24-битной шины данных. Выход DMS (выбора памяти данных) указывает на то, что на шину адреса поступил адрес памяти данных.

2.Архитектура арифметических устройств

Все вычислительные устройства: ALU (арифметическо-логическое устройство), MAC (устройство умножителя/аккумулятора) и SHIFTER (устройства сдвига) являются 16-битными устройствами с фиксированной точкой. Почти все операции подразумевают представление знаковых чисел в форме дополнения до двух. Остальные же используют беззнаковые числа или просто строки битов. Специальная поддержка имеется для многословных вычислений и блочной плавающей арифметики.

В процессорах ППС знаковые числа всегда представлены в форме дополнения до двух. Никакие другие форматы не поддерживаются.

Арифметика и типы данных. Строки битов- это простейшая двоичная форма записи из 16 бит данных. Примерами операций, в которых использу-

ется этот формат, являются логические операции NOT, AND, OR, XOR. Эти операции, исполняемые ALU,, считают, что их аргументы - строки битов и не заботятся о знаке или положении десятичной точки.

Числа без знака. Беззнаковые двоичные числа могут принимать только положительные значения и потому имеют почти вдвое больший диапазон, чем знаковые числа той же длины. Младшие слова чисел с увеличенной точностью используются как беззнаковые числа.

Числа со знаком в форме дополнения до двух. Для арифметики процессоров семейства термин "знаковый" всегда обозначает что отрицательные числа записываются в форме дополнения до двух (в дополнительном коде). Многие инструкции процессоров ППС подразумевают или поддерживают арифметику по модулю 2.

Дробь 1.15. Арифметические инструкции процессоров ППС оптимизированы , для операций с числами в дробном двоичном формате 1.15. В этом формате левый бит числа обозначает его знак, а 15 оставшихся бит представляют числа от -1 до почти 1 (из-за несимметричности представления знаковых чисел). Примеры чисел в формате 1.15 и их десятичные эквиваленты:

Число в формате 1.15	***Десятинное значение***
0x0001	**0.000031**
Qx7FFF	**0.999969**
OxFFFF	**-0.000031**
0x8000	**-1.000000**

Все арифметико-логические операции трактуют свои операнды и получают результаты как 16-разрядные битовые строки, за исключением примитивов знакового деления (DIVS). Различные флаги трактуют результаты как числа со знаком: флаг переполнения (AV) и флаг отрицательного числа (AN). Логика флага переполнения (AV) основана на арифметике по модулю 2. Он устанавливается, если знаковый бит изменяется непредсказуемым образом. Например, при сложении 2-ух положительных чисел, результат также должен быть положителен. Если же происходит переполнение (перенос в знаковый бит, устанавливающий его в единицу, так что результат получается отрицательным, устанавливается бит AV.

Логика флага переноса (AC) основана на беззнаковой арифметике. Этот флаг устанавливается в том случае, если генерируется перенос из старшего разряда числа, который не может быть записан в результат. Этот флаг очень полезен при операциях с многословными представлениями чисел для младших слов.

Результаты умножения представляют собой битовые строки. Операнды же обрабатываются так, как это указано в самой инструкции (умножение знаковых, умножение беззнаковых, умножение знакового на беззнаковое или операция округления). 32-битный результат из умножителя считается знаковым, так как происходит знаковое расширение на все 40 бит набора регистров умножителя (MR).

Процессоры ППС обычно поддерживают 2 формата коррекции результата умножения: дробный (1.15) и целый (также называется 16.0). Когда процессор умножает два 1.15 операнда, результат является числом в формате 2.30 (два знаковых бита и 30 дробных бит). В дробном режиме MAC автоматически сдвигает результат умножения влево на 1 бит перед переносом его в регистр результата (MR). После этого сдвига формат результата становится 1.31, что позволяет его округлить до формата 1.15.

В целочисленном режиме сдвиг влево не происходит. Например, если операнды формата 16.6, 32-битный результат умножения будет в формате 32.0. Более того, сдвиг влево здесь не нужен, ибо он изменит значение результата.

Многие сдвиговые операции созданы специально для знаковых или беззнаковых чисел: логические сдвиги предполагают беззнаковые операнды, тогда как арифметические сдвиги предполагают знаковые операнды.

Экспоненциальная логика предполагает знаковые операнды и поддерживает блочную плавающую точку, которая также базируется на формате дополнения до двух.

Структура ALU. Структурная схема ALU показана на рис.1.

ALU имеет три 16-битных регистра, доступных для программиста: X,У-регистры операндов и R - регистр результата. ALU использует входной сигнал переноса (CI), который соответствует биту переноса в регистре арифметического состояния (ASTAT). ALU генерирует шесть статусных сигналов:

результат ноль (AZ),
результат отрицательный (AN),
перенос (AC),
переполнение результата (AV),
знак (AS),
состояние частного (AQ).

В конце цикла все сигналы арифметического статуса изменяют состояния соответствующих битов в регистре арифметического статуса (ASTAT). Входной порт X может принимать данные из двух источников:

блока регистров AX или с шины результата (R). Шина результатов соединяет выходные регистры всех вычислительных устройств, позволяя им быть непосредственно операндами инструкций. Блок регистров AX состоит из 2-х регистров: AXO и AX1. Информацию в эти регистры можно записать, а также прочитать информацию из регистра через шину DMD. Выход блока регистров AX организован таким образом, что один из них может обеспечивать операнд для ALU, в то время как другой может передавать своё содержимое в память через шину DMD.

Входной порт Y также может принимать данные из двух источников: из набора регистров AY или из регистра обратной связи AF. Блок регистров AY состоит из 2-ух регистров AY0 и AY1. Эти регистры доступны через шину DMD.

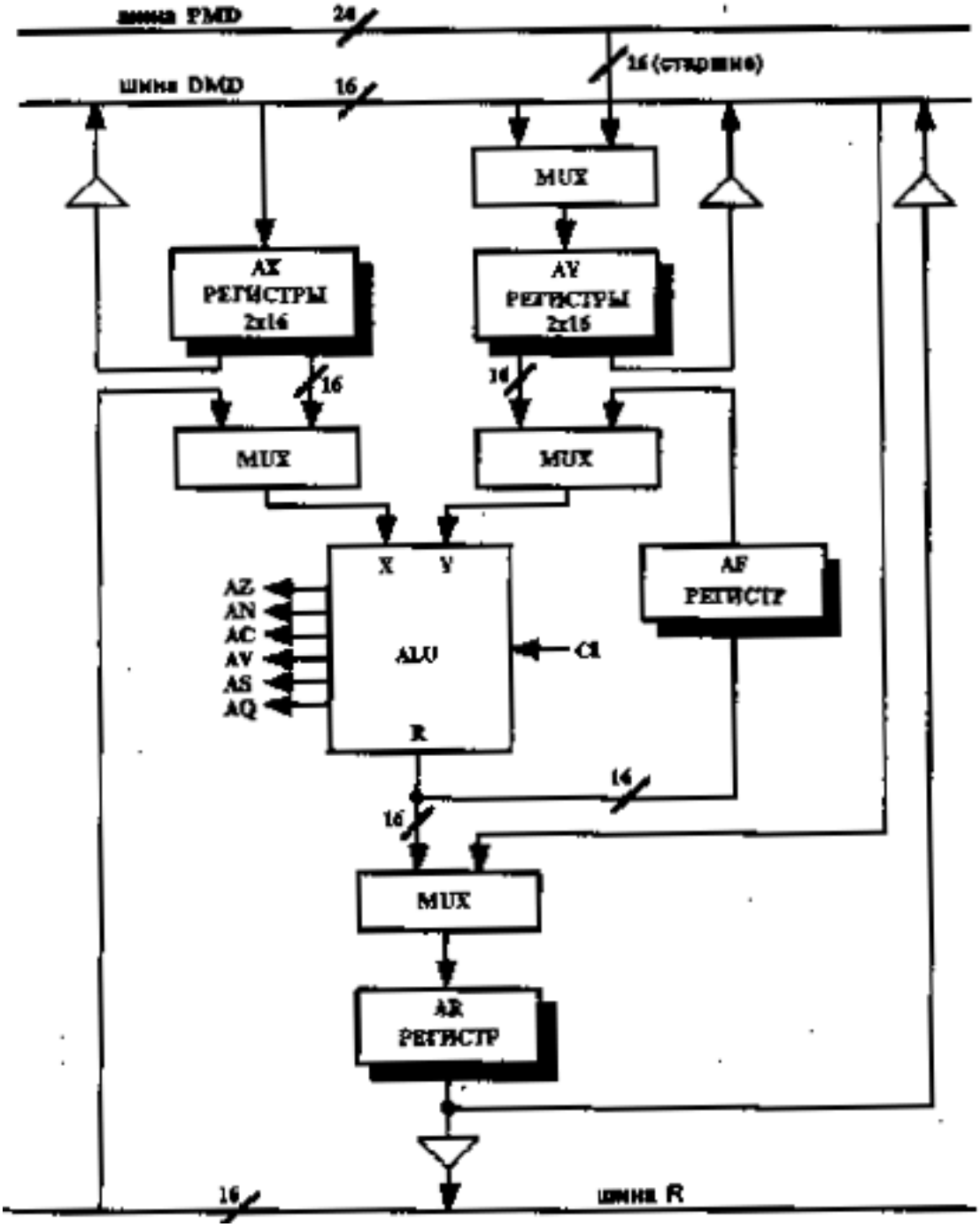


Рис.1 Структура ALU

Результат работы ALU загружается либо в регистр обратной связи AF, либо в регистр результата AR. Регистр обратной связи AF — внутренний регистр ALU, позволяет использовать результат непосредственно, как операнд Y. Регистр результата AR может записываться как на шину DMD, так и на шину результатов R. Он также непосредственно загружаем с шины DMD.

Набор инструкций позволяет осуществлять чтение этих регистров с шины PMD, но при этом нужно использовать устройство обмена между DMD-PMD шинами.

Любые регистры, связанные с ALU, доступны как для записи, так и для чтения в одном цикле. Регистры читаются в начале цикла и записываются в конце. Новое значение, записанное в регистр, не может быть считано до начала следующего цикла.

ALU содержит два набора регистров AR, AF, AXO, AX1, AYO, AY1. В каждый момент времени доступен лишь один набор. Дополнительный набор регистров может быть сделан активным (например, при обработке прерывания) для очень быстрого переключения контекстов. Новая задача, такая, как обработка прерывания, может быть выполнена без запоминания текущего состояния регистров ALU.

Выбор первичного или вторичного набора регистров контролируется битом 0 в регистре режима и статуса процессора (MSTAT). Если этот бит нулевой, используется первичный набор, если же он равен единице то используется вторичный набор регистров.

Список стандартных функций ALU:

R=X+Y	сложение X и Y
R=X+Y+CI	сложение X и Y с переносом
R=X-Y	вычесть Y из X
R=X-Y-CI-1	вычесть Y из X с заемом
R=Y-X	вычесть X из Y
R=Y-X-CI-l	вычесть X из Y с заемом
R=-X	арифметическое отрицание X
R=-Y	арифметическое отрицание Y
R=Y+1	инкремент Y
R=Y-1	декремент Y
R=PASS X	результат равен операнду X
R=PASS Y	результат равен операнду Y
R=0 (PASS 0)	очистить результат
R=ABS X	результат равен абсолютному значению X
R=X AND Y	логическое и (AND) X HY
R=X OR Y	логическое или (OR) X и Y
R=X XOR Y	исключающее логическое или (XOP) X и Y
R=NOT X	логическое отрицание X
R=NOT Y	логическое отрицание Y

Для обработки чисел с повышенной точностью предусмотрен сигнал переноса и флаг переноса (AC). Операция сложения с переносом (+CI) предназначена для сложения старших разрядов чисел двойной длины. Вычитание с заемом (+CI-1) предназначено для вычитания старших разрядов чисел двойной длины.

3.Структуры данных и переменные

Программное обеспечение поддерживает описание и использование одномерных массивов. Массив может содержать одно значение (переменную) или несколько значений (массив). Вдобавок, массив может использоваться как кольцевой буфер.

Массивы. Термины массив, буфер данных и переменная являются взаимозаменяемыми. Массивы объявляются ассемблерными директивами и могут быть косвенно адресованы или доступны непосредственно по имени, могут инициализироваться в директиве или из внешних файлов данных, могут быть линейными или с кольцевой адресацией.

Массив объявляется следующей директивой:

```
.VAR/DMmy_array[128];
```

Эта директива объявляет массив my_aггay из 128 16-битных элементов. расположенный в памяти данных (DM). Специальные операторы и % обозначают адрес и длину массива, соответственно. Эти операторы могут использоваться как показано ниже:

```
IO=^my_aггay  {указатель на адрес массива}
MXO=DM(IO,MO); {загрузка MXO из массива}
```

Эти инструкции загружают в регистр MXO значение первого элемента массива my_aггay. Используя возможность автоматической модификации адресов можно исполнять вторую из этих инструкций в цикле и таким образом поочередно адресовать все элементы массива.

Если же вам необходимо адресовать только первую ячейку массива, можно непосредственно использовать имя массива как метку, например, в следующих обстоятельствах:

```
MXO=DM(my_array);
```

Массив или буфер данных с длиной в один элемент является простой переменной размером в слово, и объявляется так:

```
.VAR/DM my_param;
```

Кольцевые массивы. Достаточно частой задачей в алгоритмах цифровой обработки является организация кольцевого буфера. Это непосредственно достигается использованием генераторов адресов данных (DAG) с использованием L-регистров. Сначала вы должны объявить буфер кольцевым:

```
.VAR/DM/CIRC my_array[128];
```

Это позволяет компоновщику поместить этот массив по допустимому адресу. Следующим шагом вы должны инициализировать L-регистр, обычно с использованием ассемблерного оператора % (или с использованием константы), а также I-регистр и M-регистр:

```
LO=%my_array; {длинна кольцевого буфера}
              {при этом оператор % автоматически}
              {представит длину массива}
IO=^my_array; {указатель на  адрес буфера}
MO=1 ;        {значение инкремента = 1}
```

Далее, инструкция:

```
MXO=DM(IO,MO); {загрузка MXO из буфера}
```

выполняемая в цикле, выбирает поочередно все элементы массива my__aггay, причем производится автоматический сдвиг к началу массива при достижении последнего элемента. Для нециклических массивов любой длины L-регистр (соответствующий I-регистру) необходимо устанавливать в 0

Косвенная линейная (не кольцевая) адресация. Косвенная адресация производится загрузкой адреса в индексный (I) регистр и указанием одного из доступных регистров-модификаторов (M).

Соответствующие L-регистры используются для циклического перехода указателя для кольцевых буферов данных. Работа с буфером как с кольцевым происходит только если соответствующий L-регистр ненулевой. Для линейной адресации, L-регистр, соответствующий I-регистру, должен быть установлен в нуль

Время исполнения инструкций. Все инструкции исполняются за один цикл, за исключением нижеследующих случаев:

- Одновременный доступ к внешней памяти более одного раза за инструкцию;
- Реализуются циклы ожидания;
- Происходит автобуферизация SPORT.

3.Система программирования ППС

С точки зрения программиста, процессор состоит из трех вычислительных устройств, двух генераторов адресов данных, и генератора адресов инструкций, а также периферии на кристалле и/или память. Почти все команды с использованием этих архитектурных компонент оперируют с одним или с большим количеством регистров — для сохранения данных, для слежения за такими значениями, как указатели, или, например, для указания режимов работы.

Система программирования процессора обычно включает в себя программно доступные регистры, совокупность реализованных команд и специальное программное обеспечение.

Внутренние регистры содержат данные, адреса, управляющую или статусную информацию. Например, AXO содержит операнд АЛУ (данные);
I4 содержит указатель (адрес) генератора адресов данных; ASTAT содержит флаги статуса арифметических операций; поля, содержащиеся в регистре DWAIT, управляют количеством циклов ожидания, используемых при доступе к различным зонам памяти данных.

Есть 2 типа доступа к регистрам. Некоторые регистры, такие, как MXO и IMASK, могут быть непосредственно записаны или считаны ассемблерной инструкцией, например :

```
MXO=1234;
IMASK=OxF.
```

Другой способ доступа – обращение к регистрам по адресам памяти данных, закрепленных за этими регистрами. Например, следующий код очистит регистр DWAIT, который находится по адресу памяти данных Ox3FFE**:**

```
AXO=0;
DM(Ox3FFE)=AXO;
```

(AXO используется для хранения константы 0 так как в системе команд ППС нет инструкции для записи константы по непосредственному адресу.)

Все регистры, доступные в процессорах семейства ADSP-21x показаны на рисунке 2. Регистры сгруппированы по выполняемой ими функции: генераторы адресов данных (DAGs), генератор адресов инструкций, вычислительные устройства (АЛУ, MAC и SHIFTER), устройство обмена между шинами, интерфейс с памятью, последовательные порты, таймер, порт интерфейса с хост-процессором (HIP) и аналоговый интерфейс.

Генераторы адресов данных. DAG1 и DAG2 каждый имеют по 12 14-битных регистров: 4 индексных (I) регистра для хранения указателей, 4 регистра-модификатора (M) для модификации указателей и 4 регистра длины (L) для реализации кольцевых буферов. DAG1 адресует только память данных и имеет возможность битового обращения (реверс бит) генерируемого адреса. DAG2 адресует как память программ, так и память данных и может генери-

ровать адреса для безусловных переходов (переходы и вызовы процедур),как и адреса для доступа к данным.
Например, инструкция:

AXO=DM(IO.MO);

производит косвенное чтение памяти данных (DM — Data Memory) по адресу, указываемому IO. Инструкция

PM(I4,M5)-MRI;

производит косвенную запись памяти инструкций (PM — Program Memory) по адресу, указываемому I4 с пост-модификацией регистра I4 регистром M5. Инструкция

JUMP (I4);

показывает пример косвенного перехода.

Генератор адресов инструкций . Регистры, относящиеся к генератору адресов инструкций, управляют подпрограммами, циклами и прерываниями. Они также содержат его статус и выбирают режим операции.

• Прерывания

Регистр 1CNTL управляет вложением прерываний и их чувствительностью ко внешним прерываниям; регистр IFC позволяет программно очищать и генерировать прерывания; регистр IMASK маскирует (запрещает) отдельные прерывания.

• Счетчики циклов

Регистр CNTR содержит значение счетчика текущего цикла. Стек счетчика циклов позволяет вложение циклов до 4 уровней с использованием аппаратного счетчика циклов. Запись в этот регистр сохраняет его текущее содержимое в стеке счетчика циклов перед записью нового значения. Например, инструкция

CNTR=1O;

сохраняет текущее содержимое CNTR в стеке счетчика циклов и затем загружает в него значение 10.

Инструкция OWRCNTR позволяет изменить значение счетчика текущего цикла без его сохранения в стеке счетчика циклов.

• Биты статуса и режима

Регистр статуса стека (SSTAT) содержит флаги пустоты и полноты стеков. Регистр арифметического статуса (ASTAT) содержит статусные флаги вычислительных устройств. Регистр режима и статуса (MSTAT) содержит различные управляющие флаги. Он содержит 7 бит, которые управляют выбором дополнительного набора регистров, режимом реверса DAG1, режимом переполнения и насыщения АЛУ, расположением результата MAC, разрешением таймера и GO-режимом.

• Стеки

Генератор адресов инструкций содержит четыре стека, которые позво-

ляют проводить вложение циклов, подпрограмм и прерываний.

Стек счетчика инструкций имеет ширину 14 бит и глубиной в 16 позиций. Он содержит адреса возврата из подпрограмм и программ обработки прерываний , а также адреса вершины цикла для циклов.

3

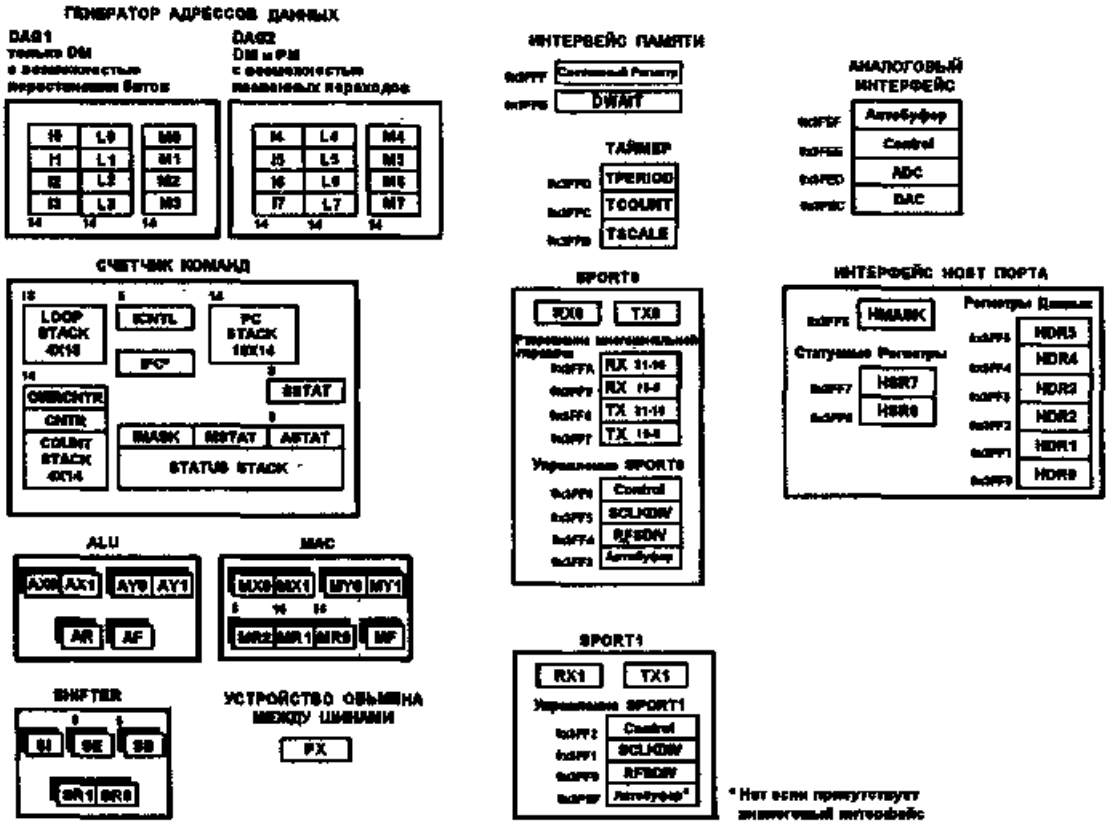


Рис.2 Регистры процессоров семейства ADSP-21x

Работа с этим стеком происходит |автоматически при вызове подпрограмм и прерываний. Вдобавок, существуют инструкций (PUSH, POP) для ручного помещения и извлечения данных из стека счетчика инструкций.

Стек циклов имеет ширину 18 бит, 14 бит для хранения адреса конча цикла и 4 бита для хранения условия завершения цикла. Этот стек содержит 4 элемента. Значение в нем автоматически сохраняется при исполнении инструкции DO UNTIL**.** Это значение автоматически извлекается из него во время выхода из цикла, если цикл был вложенным. Значение из стека циклов может быть вручную извлечено инструкцией POP LOOP.

Стек статуса, значение в котором автоматически сохраняется при входе в процедуру обработки прерывания, вмещает содержимое регистров маскирования прерываний (IMASK), режима и статуса (MSTAT),и арифметического статуса (ASTAT). Глубина и ширина этого стека зависит от модели процессора, ибо различные модели поддерживают различное количество прерываний. Содержимое этого стека автоматически восстанавливается при ис-

полнении инструкции RTI (возврат из прерывания). В/из стека статуса можно сохранять/восстанавливать значения вручную с помощью инструкций PUSH STS и POP STS.

Стек счетчика циклов имеет ширину 14 бит и содержит значение счетчиков (CNTR) дня вложенных циклов, использующих аппаратный счетчик. Текущее значение в него автоматически помещается при записи нового значения в регистр CNTR. Значение из стека счетчика циклов может быть вручную извлечено инструкцией POP CNTR.

Вычислительные устройства. Регистры вычислительных устройств содержат данные.

ALU и MAC требуют двух операндов для большинства операций. Регистры AXO,AX1,MXO,MX1 содержат операнды X, а регистры AYO.A MYO, MY1 содержат операнды Y.

Регистры AR и AF содержат результаты ALU; AF может служить Y-операндом для следующей операции ALU, а AR может может служить X-операндом для любого вычислительного устройства. Также, регистры M MR1,MR2 и MF содержат результаты MAC-а и могут использоваться как операнды для дальнейших вычислений. 16-битные регистры MRO и N вместе с 8-бигным MR2 могут содержать 40-битный результат MAC.

SHIFTER может получать исходные данные из ALU или MAC, из своих собственных выходных регистров, или из специального входного регистра сдвигового массива (SI). Он хранит 32-битный результат в регистрах SR0 и SR1. Регистр SB содержит экспоненту для блочных операций с плавающей точкой. Регистр SE содержит значение сдвига для операций нормализации и денормализации.

Набор команд. Набор инструкций обеспечивает полный контроль над всеми тремя вычислительными устройствами процессора. Арифметические инструкции работают с16-битными операндами непосредственно.

Также приняты необходимые меры для работы с числами повышенной точности.

Высокоуровневый синтаксис ассемблера процессоров ППС одновременно хорошо читаем и эффективен. В отличие от многих ассемблеров, он использует алгебраическую нотацию для записи арифметических операций и пересылок данных, что приводит к высокой читаемости исходного кода. В то же время не происходит потери производительности: каждый оператор ассемблерной программы транслируется в одну 24-битную инструкцию, которая исполняется за один цикл. В наборе инструкций нет инструкций**,** выполняющихся дольше одного цикла (если доступ к памяти не требует циклов ожидания.

Во всех процессорах семейства инструкция IDLE введена для помещения процессора в состояние ожидания прерывания. Эта инструкция переводит процессор в состояние пониженного энергопотребления во время ожидания прерывания.

Поддерживаются 2 режима адресации для доступа к памяти. Прямая адресация использует непосредственное выражение в качестве адреса памяти;

косвенная адресация использует индексные (I) регистры генераторов адресов данных.

24-битное слово инструкции позволяет распараллелить операции. Набор инструкций позволяет исполнить за один цикл любую из следующих комбинаций инструкций:

любую операцию ALU, MAC или SHIFTER,условную или безусловную;
любую пересылку регистр-регистр;
любое чтение или запись памяти данных;
вычисление с одновременной пересылкой регистр-регистр;
вычисление с одновременной записью или чтением памяти;
вычисление с одновременным чтением 2 операндов из памяти.

Набор инструкций позволяет программировать ППС с максимальной гибкостью. Он обеспечивает пересылку из любого регистра в любой, и пересылку в/из большинства регистров в память. В дополнение почти любая инструкция ALU, MAC и SHIFTER может быть скомбинирована с любой пересылкой регистр-регистр или с пересылкой регистр-память.

Набор инструкций процессоров семейства ADSP-21xx разделен на следующие категории:

Вычислительные: ALU, MAC и SHIFTER.
Пересылки данных.
Управление программой.
Многофункциональные инструкции
Различные инструкции.

4 Подробнее с системой команд можно ознакомиться по специальной литературе.

Семейство процессоров ADSP-21xx имеет полный набор программных и аппаратных средств поддержки разработчика. Средства разработчика для процессоров семейства ADSP-21xx включают программное обеспечение для программирования и аппаратные эмуляторы для отладки программного и аппаратного обеспечения.

Программное обеспечение разработчика включает:

• *Построитель систем* - определяет архитектуру конструируемой системы. Это включает в себя наличие и размер внешней RAM/ROM памяти, отображение портов ввода-вывода для конструируемого устройства, а также расположение памяти инструкций и данных.

• *Ассемблер* — ассемблирует модули с инструкциями и данными и поддерживает высокоуровневый набор инструкций. В дополнении к тому, что он поддерживает полный диапазон системных диагностик, он имеет гибкий макроязык, включаемые файлы, и поддерживает модульное программирование.

• *Линкер* — объединяет отдельно ассемблированные модули. Он располагает данные и код программы по аппаратным компонентам конструируемой системы, так, как это указано в выводе Построителя Систем.

• *Эмулятор* — позволяет проводить интерактивную покомандную эмуляцию аппаратной конфигурации, описанной Построителем Систем. Он отме-

чает некорректные операции и поддерживает полностью символическое ассемблирование и дизассемблирование.

• *Программатор ППЗУ*— читает вывод линкера и генерирует файлы, совместимые с программаторами ППЗУ.

• *Компилятор языка С* - читает ANSI-C совместимые исходные коды и выводит команды ADSP-21xx ассемблера, готовые к ассемблированию. Также поддерживается использование "ин-лайновых" ассемблерных кодов.

5 • *Эмуляторы EZ-ICE* — обеспечивают аппаратную отладку ADSP-2100 систем. Они обеспечивают внешнюю внутрисхемную эмуляцию с небольшим или вообще нулевым замедлением. Платы EZ-ICE представляют из себя дешевые базовые аппаратные платформы для запуска примеров приложений.

6 Производительность для задач DSP. DSP приложения налагают специальные требования на производительность процессора, что отличает архитектуру DSP от других архитектур и процессоров. Должно обеспечиваться не только быстрое исполнение инструкций, но и достаточно быстрая работа в следующих областях:

• *Быстрая и гибкая арифметика* — базовая архитектура обеспечивает выполнение умножения, умножения с накоплением, сравнительно большого объема сдвигов, и стандартных арифметико-логических операций в течение одного цикла. Кроме того, арифметические устройства допускают произвольную последовательность вычислений, так что данный DSP алгоритм может исполняться без изменения последовательности инструкций.

• *Расширенный диапазон* — расширенные суммы произведений, достаточно часто встречающиеся в DSP алгоритмах, поддерживаются в устройстве умножения/аккумулирования (MAC) процессоров семейства. 40-битовый аккумулятор обеспечивает 8-битовыю защиту от переполнения при последовательности сложений, чтобы не допустить потерю данных или диапазона; прежде чем произойдет потеря данных, должно случиться 256 переполнений. Существуют специальные инструкции для масштабирования чисел с плавающей точкой.

• *Загрузка 2-ух операндов в одном цикле* — при расширенном суммировании произведений, в каждом цикле требуется 2 операнда, чтобы производить вычисления. Процессоры могут обеспечить производительность 2 операнда/цикл, где бы ни находились данные — в памяти на чипе или во внешней памяти.

• *Аппаратно реализованные кольцевые буфера* — большой класс DSP приложений, включая фильтры, нуждается в кольцевых буферах. Процессоры включают аппаратуру указателя для обработки циклического перехода, что упрощает реализацию кольцевых буферов и уменьшает нагрузку, что в свою очередь увеличивает производительность.

• *Циклы и переходы без дополнительных циклов ожидания* — многие DSP алгоритмы повторяемы и семантически представляют из себя циклы. Генератор адресов инструкций процессоров семейства поддерживает циклы без дополнительных затрат времени, сочетая отличную производительность с

ясностью структуры программы. Также, условные переходы не вызывают дополнительных затрат времени.

• *И другие специализированные операции,* в том числе примитивы операций деления и арифметики с плавающей запятой, режим реверса бит адреса для реализации алгоритма БПФ и многое другое.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8
ИЗУЧЕНИЕ ПРОГРАММНОЙ СРЕДЫ MPLAD IDE.

Цель занятия

Изучить основы программирования цифровых сигнальных процессоров (ЦСП) фирмы Microchip. Научиться создавать новый проект, настраивать параметры проекта, компилировать проект.

Учебные вопросы

1. Введение в язык ассемблер ЦСП
2. Создание нового проекта
3. Работа с документацией по микропроцессору.

Для программирования ЦСП фирма Microchip предоставляет разработчику программную среду MPLAB IDE. Данная программная среда позволяет выполнить весь спектр операций при создании работоспособного кода для ЦСП. Разработчик имеет возможность создать программный код как на языке низкого уровня - ассемблер, так и на языке высокого уровня – СИ.

Выбор языка программирования выполняется на этапе создания проекта. Также, в особых случаях, существует возможность использования двух языков программирования. После написания программы, ее компиляции (создания машинного кода по написанному разработчиком ассемблерному или СИ коду), имеется возможность пошаговой отладки программного кода.

Процесс отладки представляет собой важный этап в разработке программного кода, при котором выявляются ошибки невидимые компилятором (ошибки, не являющиеся синтаксическими), например взаимодействие отдельных блоков кода программы, взаимные противоречия и т.п. Уменьшить возможные логические ошибки помогает детальная схема алгоритма программного кода. Последним этапом является программирование ЦСП откомпилированным машинным кодом.

Для программирования ЦСП в работе используется внутрисхемный отладчик ICD2. Он может работать как программатор, так и внутрисхемный отладчик. В режиме работы ICD2 как внутрисхемного отладчика пользователь может отлаживать (прогонять) программу непосредственно в готовом устройстве. Существуют ошибки программного кода, которые возможно выявить только при использовании внутрисхемного отладчика.

Создание нового проекта

1. Запустите MPLAB IDE.

2. Выберите меню .

3. На первом шаге необходимо выбрать процессор dsPIC33FJ256GP710

4. На втором шаге выберите язык программирования – Microchip ASM30 Toolsuite.

5. На третьем шаге необходимо определить каталог нового проекта и

название. Следует отметить, что путь должен быть только из латинских букв и не слишком длинным.

6. На четвертом шаге в проект включаются уже готовые модули. Пока пропустите данный этап. После выполнения вышеприведенных шагов экран будет выглядеть, как показано на рисунке 1.

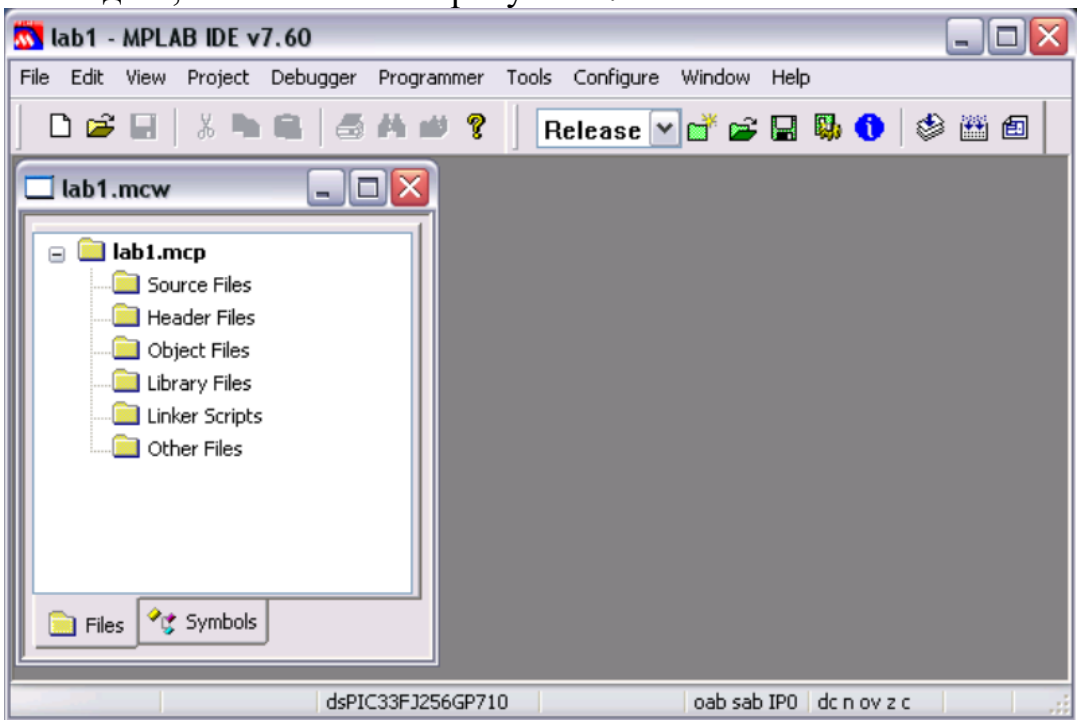


Рисунок 1 – Вид программы после создания нового проекта

В данном случае в окне с названием проекта нет ни одного рабочего файла.

Подключение библиотек и упаковочных файлов процессора

1. Нажмите правой клавишей по полю с надписью .
2. Выберите .
3. Необходимо подключить файл , расположенный в каталоге
4. Нажмите правой клавишей по полю с надписью .
5. Выберите .
6. Необходимо подключить файл <p33FJ256GP710.inc>, расположенный в каталоге <C:\Program Files\Microchip\MPLAB ASM30 Suite\Support\dsPIC33F \inc>

После подключения необходимых библиотек окно программы будет выглядеть подобно рисунку 2.

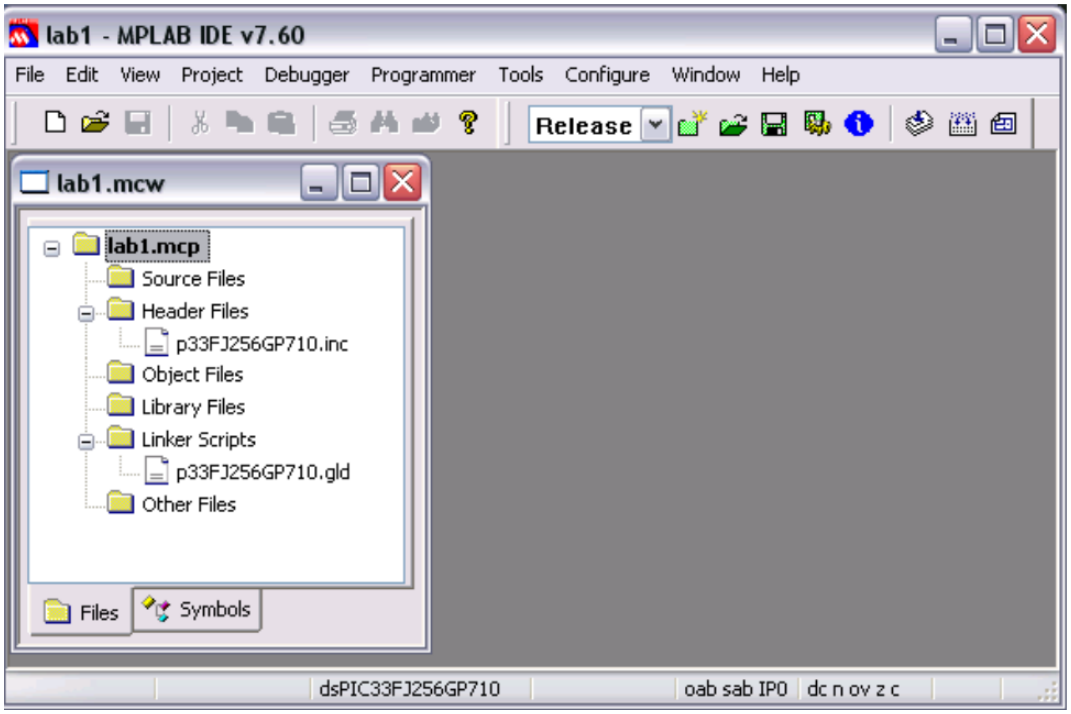


Рисунок 2- Вид программы после подключения библиотек и упаковочных файлов процессора

Создание файла с исполняемым кодом

1. Выберите меню <File\New>
2. Запишите его с именем <main.s> в каталог с программой, т.е. D:\brig1\lab1\lab1
3. Нажмите правой клавишей по полю с надписью <Source Files>.
4. Выберите <Add Files…>.
5. Необходимо подключить файл <main.s>.
6. Введите в файл <main.s> следующий код:

```
      .include "p33FJ256GP710.inc"
            .global __reset
            .text
__reset:
            nop
            mov         #0, W1
            mov         #100, W2
main:
            inc         W1, W1
            inc         W2, W2
            add         W1, W2, W3
            goto        main
.end
```

В первой строке подключается заголовочный файл процессора <p33fj256gp710.inc>. Во второй, метке <__reset> назначается глобальный статус, по этой метке код привязывается к стартовому адресу. Далее директива <.text>, которая определяет следующий за ним текст как исполняемый код. Далее идет ассемблерный код.

Заканчивается ассемблерный код директивой <.end>.

При правильном выполнении операций вид программы будет соответствовать рисунку 3.

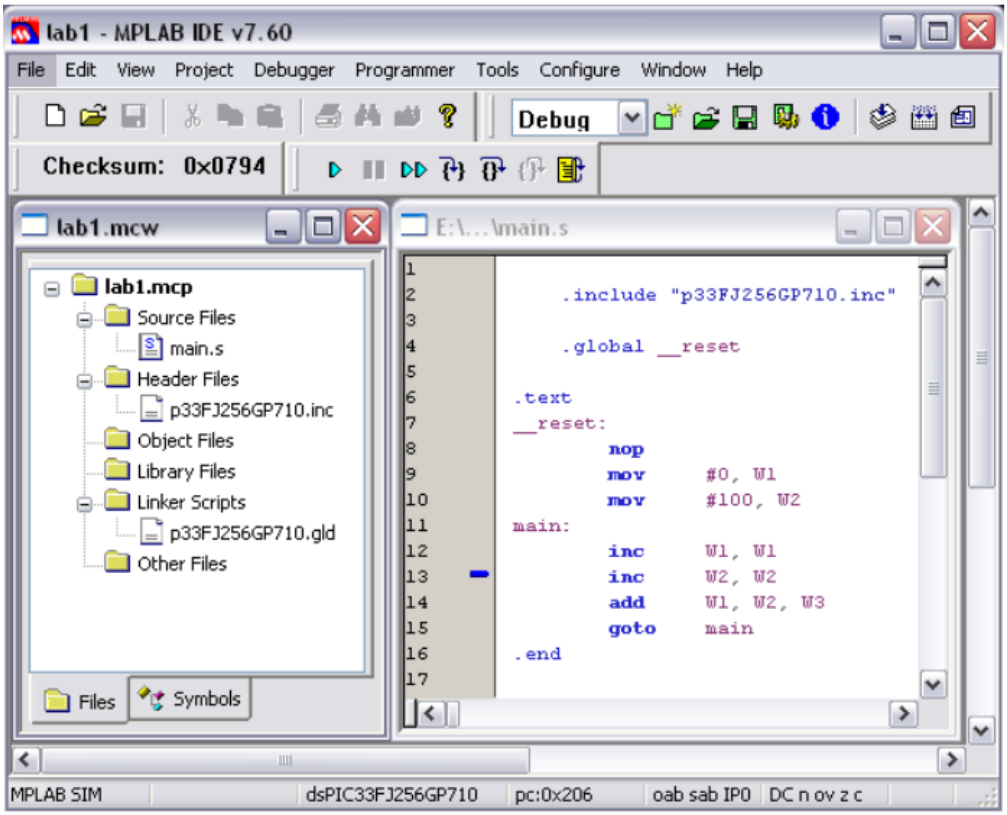


Рисунок 3- Создание файла с исполняемым кодом

Задание для самостоятельного выполнения

1. Создайте новый проект. Процессор – dsPIC33FJ256GP710.

2. Подключите необходимые библиотеки и перепишите программу, приведенную выше.

3. Необходимо подключить отладчик (симулятор), встроенный в среду MPLAB IDE. Выберите в меню <Debugger\Select Tool\MPLAB SIM>.

4. Скомпилируйте проект, для этого выберите в меню <Project\Build All>. Если код написан правильно и ошибок при компиляции нет, то можно открыть окно памяти программ, в котором написанная программа предстаиться в машинных кодах по жестко прописанным ячейкам памяти. Нажмите в меню <View\Program Memory>.

5. Откройте окно Watch, которое предназначено для отображения состояния необходимых переменных в режиме отладки программы. Для этого выберите в меню <View\Watch>. Добавьте в окно Watch регистры, которые мы хотим наблюдать в ходе выполнения программы. Для этого в левом

окошке найдите WREG1 и нажмите кнопку <Add SFR>. Далее добавьте регистры WREG2, WREG3. Результат представлен на рисунке 4.

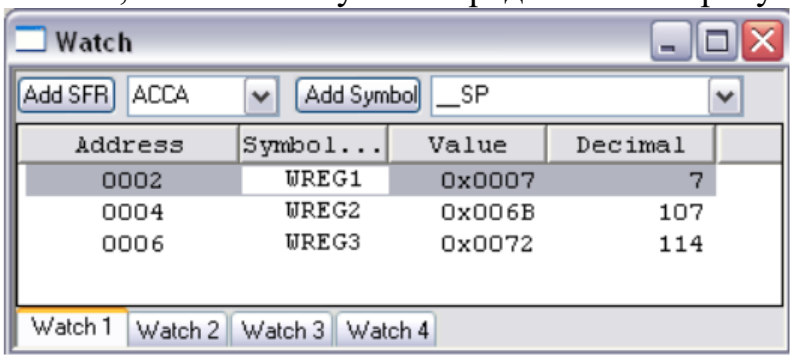


Рисунок 4- Окно Watch

6. Далее приступите к отладке кода. Перейдите в окно <Program Memory>, причем окна должны так располагаться, чтобы вы могли наблюдать за изменением состояния регистров в окне <Watch>. При каждом нажатии на клавишу <F7 > ассемблерный код будет выполняться по одной команде. Исследуйте изменение регистров согласно пошаговому выполнению программы.

7. Для того чтобы программа выполнялась не в пошаговом режиме (для отладки), а в обычном необходимо нажать клавишу <F9>. Так же, если необходимо прервать программу нажмите <F5>, а если установить начальный (стартовый) адрес, следует нажать клавишу <F6>.

8. Скорректируйте программу, согласно варианту

Вариант 1. Регистр WREG1 инициируется значением 1, регистр WREG2 – значением 10. Регистр WREG1 инкрементируется на 2. Содержимое регистра WREG2 декрементируется на 1, а результат помещается в регистр WREG3.

Вариант 2. Регистр WREG1 инициируется значением 2, регистр WREG2 – значением 9. Содержимое регистра WREG1 декрементируется на 1, а результат помещается в регистр WREG4. Регистр WREG2 инкрементируется на 2.

Вариант 3. Регистр WREG1 инициируется значением 3, регистр WREG2 – значением 5. Регистр WREG1 декрементируется на 1. Содержимое регистра WREG2 инкрементируется на 1, а результат помещается в регистр WREG5.

Вариант 4. Регистр WREG1 инициируется значением 4, регистр WREG2 – значением 6. Регистр WREG2 декрементируется на 1. Содержимое регистра WREG2 инкрементируется на 2, а результат помещается в регистр WREG4.

Вариант 5. Регистр WREG1 инициируется значением 10, регистр WREG2 – значением 0. Регистр WREG1 декрементируется на 2. Содержимое регистра WREG2 инкрементируется на 2, а результат помещается в регистр WREG3.

Вариант 6. Регистр WREG1 инициируется значением 20, регистр WREG2 – значением 2. Содержимое регистра WREG1 инкрементируется на 1, а результат помещается в регистр WREG6. Регистр WREG2 декрементируется на 2.

Вариант 7. Регистр WREG1 инициируется значением 11, регистр WREG2 – значением 10. Содержимое регистра WREG1 инкрементируется на 2, а результат помещается в регистр WREG3. Регистр WREG2 декрементируется на 1.

Вариант 8. Регистр WREG1 инициируется значением 5, регистр WREG2 – значением 6. Регистр WREG1 декрементируется на 1. Содержимое регистра WREG2 инкрементируется на 2, а результат помещается в регистр WREG4

Вариант 9. Регистр WREG1 инициируется значением 15, регистр WREG2 – значением 1. Содержимое регистра WREG1 декрементируется на 1, а результат помещается в регистр WREG3. Регистр WREG2 инкрементируется на 2.

Вариант 10. Регистр WREG1 инициируется значением 100, регистр WREG2 – значением 0. Регистр WREG1 инкрементируется на 2. Содержимое регистра WREG2 декрементируется на 1, а результат помещается в регистр WREG4.

10.Подготовьте отчет в соответствии с требованиями, изложенными на стр. 21.

Контрольные вопросы

1. На каком языке программирования можно написать код для процессоров, рассматриваемых в работе?
2. Назначение MPLAB SIM.
3. Назначение MPLAD IDE.
4. Объясните процесс компиляции проекта.
5. Назначение окна Watch.
6. Назначение окна Program Memory.
7. С помощью какой инструкции осуществляется инкремент содержимого регистра?
8. С помощью какой инструкции осуществляется декремент содержимого регистра?

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
по выполнению лабораторных работ
по дисциплине
«Цифровые сигнальные процессоры (DSP)»
для студентов направления подготовки
11.04.04 «Электроника и наноэлектроника»
Направленность (профиль)
«Интеллектуальные программируемые системы и устройства»

Ставрополь
2026

Содержание

ЛАБОРАТОРНАЯ РАБОТА №1.Базовые сигналы в ЦОС

ЦЕЛЬ РАБОТЫ – изучение пакета Matlab,программирование базовых сигналов цифровой обработки (ЦОС) в пакете Matlab.

1.1 Теоретические сведения

Пакет Matlab

Для выполнения лабораторных работ используется система инженерных и научных расчетов Matlab (сокращение от MATrixLABoratatory – матричная лаборатория). Основным объектом системы Matlab является матрица. Все вычисления система осуществляет в арифметике с плавающей точкой.

Система Matlab работает в режиме интерпретации команд и операторов, которые вводятся в ходе сеанса в командной строке, а Matlab выполняет их немедленную обработку и выдает вычисленный результат. Однако в Matlab можно запустить на выполнение заранее подготовленную последовательность команд и операторов, записанную в виде файла с расширением .m (M-файл).

M-файлы разделяются на два вида: файлы-сценарии и процедуры-функции.

Все M-файлы должны располагаться в рабочем каталоге или каталоге, зарегистрированном в списке путей системы Matlab. Для изменения рабочего каталога необходимо выбрать в меню File пункт SetPath.... При проведении лабораторной работы система разрешает изменять только рабочий каталог и не разрешает добавлять и удалять каталоги из списка путей. При выходе из системы сведения о рабочем каталоге не сохраняются, поэтому при новом запуске Matlab его необходимо ввести заново.

Файлы-сценарии

Файл-сценарий – это текстовый файл, содержащий последовательности команд и операторов. В языке нет специальных операндов для обозначения начала и конца файла-сценария, а также точки входа. Началом сценария является начало файла, а концом соответственно конец файла.

Именем сценария является имя его файла, отсюда следуют ограничения, накладываемые на имена файлов:

1) имя файла сценария может содержать только символы латинского алфавита от `A` до `Z` и от `a` до `z`, цифры от `0` до `9` и символ подчеркивания;

2) имя файла должно начинаться с символов`A...Z` или `a...z`либо с символа подчеркивания.

Результат выполнения каждого оператора отображается в окне командной строки. Для того чтобы система не выводила результат, необходимо в конце оператора ставить точку с запятой.

Файлы процедуры-функции

Файл процедуры-функции также является текстовым файлом, содержащим последовательности команд и операторов. Отличия файла процедуры-функции от файла-сценария заключаются в следующем:

1. Первой строкой должен быть заголовок функции, имеющий следующий формат:

```
function [<список выходных параметров>] = <имя функции>(<список входных параметров>)
```

Имя функции должно совпадать с именем файла.

2. При выполнении функции она сначала компилируется во внутренний формат, после чего исполняется. В связи с этим функция выполняется на порядок быстрее файла-сценария, который работает в режиме интерпретации команд.

По соглашению, принятому в Matlab, начиная со второй строки функции может располагаться несколько строк комментария. Этот комментарий считается справкой по использованию функции и может быть вызван с помощью команды `help` и имени функции. Такую справку содержат абсолютно все функции пакета Matlab. Например:

```
help filter   % справка по функции filter
```

Рассмотрим функцию, вычисляющую минимальное и максимальное значения массива (файл `minmax.m`):

```
function [minim, maxim] = minmax(a)
minim = min(a)
maxim = max(a)
```

Как видно из данного примера, функция может иметь несколько возвращаемых значений. Представленную в примере функцию можно использовать следующим образом:

```
[a,b] = minmax(x)    % В a будет минимум, а в b - максимум
    a = minmax(x)    % В a будет минимум, а максимум теряется
```

Данные в системе Matlab

В системе Matlab присутствует только один тип данных – это прямоугольный массив комплексных чисел. Для создания переменной ей просто необходимо присвоить некоторое значение. Существует несколько вариантов записи массива:

1. *Объявление пустого массива*. Объявление пустого массива, т.е. массива нулевого размера, имеет следующий вид:

```
a = []
```

2. *Объявление скаляра*. Скаляр является частным случаем матрицы, т.е. это матрица размером 1×1. Пример:

```
a = 5
b = -4.8
c = 0.25 + 18i % Комплексное число
```

3. *Объявление вектора*. Для объявления вектора необходимо записать его элементы, разделенные пробелами, в квадратных скобках. Пример:

```
a = [0 1]
b = [3.1 -6]
c = [a 4 b]    % Результат c = [0 1 4 3.1 -6]
```

4. *Объявление монотонно возрастающего или убывающего вектора*. Формат объявления монотонно возрастающего или убывающего вектора следующий: *первый_элемент* : [*шаг* :] *конечный_элемент*. Если шаг равен 1, то его разрешается опустить. Пример:

```
a = 1:10      % Результат b = [1 2 3 4 5 6 7 8 9 10]
b = 4:-2:-4   % Результат b = [4 2 0 -2 -4]
c = -1:3:10   % Результат b = [-1 2 5 8]
```

5. *Объявление массива*. Для объявления вектора необходимо записать его элементы в квадратных скобках, притом элементы строки разделяются пробелами, а строки между собой – точкой с запятой. Пример:

```
a = [1 2 3; 4 5 6; 7 8 9]    % Результат
```
$$a = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Bmatrix}$$

Для объявления массивов также используются специальные функции. Вот некоторые наиболее часто используемые:

`zeros(m, n)` – возвращает массив размером $m \times n$, заполненный нулями;

`ones(m, n)` – возвращает массив размером $m \times n$, заполненный единицами;

`rand(m, n)` – возвращает массив размером $m \times n$, заполненный псевдослучайными величинами, распределенными по равномерному закону;

`randn(m, n)` – возвращает массив размером $m \times n$, заполненный псевдослучайными величинами, распределенными по нормальному закону.

В данных функциях может опускаться первый операнд. В этом случае возвращается квадратная матрица *n*×*n*. Пример:

```
a = zeros(1, 100)% Вектор из 100 нулей
b = ones(20)     % Матрица 20×20 единиц
c = rand(2, 50)  % Матрица 2×50 псевдослучайных чисел
```

Доступ к массивам и его элементам

Доступ к массивам осуществляется по их именам. Все операции с массивами осуществляются по значению, т.е. например, при выполнении операции присваивания будет копироваться все содержимое массива. Пример:

```
a = [2 4 5; 4 6 1]
b = a
```

В результате выполнения данной программы переменная `b` будет иметь размер 2×3 и содержать значение [2 4 5; 4 6 1], притом `a` и `b` будут размещаться в разных участках памяти.

Для доступа к элементу массива необходимо после имени задать в круглых скобках индекс элемента, состоящий из двух чисел или векторов, разделенных запятой. Нумерация начинается с единицы. Первое число или вектор определяет номер строки (номера строк), второе – соответственно столбца. Если номер строки пропущен (обычно при обращении к элементам вектора), то он считается равным единице. Пример:

```
a = [11 12 13 14 15;         % Исходный массив
     21 22 23 24 25;
     31 32 33 34 35;
     41 42 43 44 45]
b = a(2, 3)              % Результат: b = 23
a(3) = 6                 % Результат: a(1,3) = 0
c = a(3, 2:4)            % Результат: c = [32 33 34]
d = a([1 3], 1:3)        % Результат: d = [11 12 13;
                         %                 31 32 33]
a([1 4], [1 5]) = zeros(2,2); % Результат:
                         % a = [0  12 13 14 0;
                         %      21 22 23 24 25;
                         %      31 32 33 34 35;
                         %      0  42 43 44 0]
```

Арифметические операторы

В системе реализовано два типа арифметических операций. Операции над матрицами определены в соответствии с правилами линейной алгебры, а операции над массивами выполняются поэлементно. Для обозначения операций выполняемыми над элементами массива используется знак точки«**.**» (табл. 1.1).

Таблица 1.1

Операции Matlab

Знак	Операция
+	**Сложение: C=A+B** Если A – скаляр, B – массив, то $C_{ij} = A + B_{ij}$. Если A – массив, B – скаляр, то $C_{ij} = A_{ij} + B$. Если A – массив, B – массив, то $C_{ij} = A_{ij} + B_{ij}$. Для выполнения операции сложения массивов они должны иметь одинаковый размер.
–	**Вычитание: C=A–B** Если A – скаляр, B – массив, то $C_{ij} = A - B_{ij}$. Если A – массив, B – скаляр, то $C_{ij} = A_{ij} - B$. Если A – массив, B – массив, то $C_{ij} = A_{ij} - B_{ij}$. Для выполнения операции вычитания массивов они должны иметь одинаковый размер.
*	**Умножение матриц: C=A*B** Если A – скаляр, B – массив, то $C_{ij} = A \cdot B_{ij}$. Если A – массив, B – скаляр, то $C_{ij} = A_{ij} \cdot B$. Если A – массив, B – массив, то $C_{ik} = \sum_{j=1}^{n} A_{ij} \cdot B_{jk}$. Для выполнения операции число столбцов первого массива должно быть равно числу строк второго массива.
.*	**Поэлементное умножение**:**C=A.*B** Если A – скаляр, B – массив, то $C_{ij} = A \cdot B_{ij}$. Если A – массив, B – скаляр, то $C_{ij} = A_{ij} \cdot B$. Если A – массив, B – массив, то $C_{ij} = A_{ij} \cdot B_{ij}$. Для выполнения операции поэлементного умножения массивов они должны иметь одинаковый размер.
`	**Транспонирование матрицы: A`** Для действительных массивов результатом является транспонированная матрица. Для комплексных массивов транспонирование дополняется комплексным сопряжением.
.`	**Транспонирование массива: A.`** Для действительных и комплексных массивов строки просто заменяются столбцами. Комплексное сопряжение не выполняется.

Логические операторы. Операции отношения.

В языке Matlab используются следующие логические операторы (табл.1.2):

Таблица 1.2

Операции отношения в Matlab

Операция	Отношение
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
==	Равно
~=	Не равно

Все логические операторы осуществляют операцию поэлементного сравнения двух массивов. Если один из операндов является скаляром, то он поэлементно сравнивается со всеми элементами другого операнда. Логические операторы возвращают в качестве результата массив того же размера, элементы которого равны единице, если результат сравнения соответствующих элементов равен ИСТИНА, и нулю – в противоположном случае.

Операторы <, >, <=, >= используются для сравнения только действительных частей комплексных элементов, а операции == и ~= осуществляют сравнение как действительных, так и мнимых частей. Пример:

```
X = [2 4 2.5;
     12i 6 3]
Y = [3 4 12;
     2 3 6]
Z = X >= Y

% Результат

Z = [0 1 0;
     0 0 1]
```

Логические операции

В языке Matlab есть три логические операции (табл. 1.3):

Таблица 1.3

Логические операции в Matlab

Операция	И	ИЛИ	НЕ
Обозначение	&	|	~

При выполнении логических операций массив рассматривается как совокупность булевых переменных, так что значение 0 соответствует булеву значению FALSE, а любое другое значение – булеву значению TRUE.Функция ИСКЛЮЧАЮЩЕЕ ИЛИ реализована в виде функции `xor(A, B)`.

Логические операции имеют низший приоритет по отношению к операциям отношения и арифметическим операциям.

Оператор цикла с определенным числом операций

Оператор цикла с определенным числом операций имеет следующий вид:

```
for v = <выражение-массив>
<операторы>
end
```

В отличие от универсальных языков программирования переменная цикла в языке Matlab является массивом. Поэтому исполнение цикла состоит в том, что переменной цикла присваиваются значения столбцов массива и затем выполняются операторы, которые должны зависеть от переменной `v`. Тем не менее на практике чаще всего применяют линейные конструкции вида `m : n` или `a : <шаг> : b`.

Для прерывания выполнения цикла используется оператор `break`. Пример:

```
for i = 1:10
   x(i) = i .^ 2
end                % Результат x = [1 4 9 16 ... 100]

y = []
for k = [0 3 1 2]
   y = [y 2.^k]
end                % Результат y = [1 8 2 4]
```

Оператор цикла с неопределенным числом операций

Оператор цикла с неопределенным числом операций имеет следующий вид:

```
while <логическое выражение>
<операторы>
end
```

Цикл `while ... end` выполняется до тех пор, пока массив логического выражения не станет нулевым.

Логическое выражение имеет форму

`выражение <оператор отношения> выражение,`

где допустимы следующие операторы отношений : ==, ~=, <=, >=, <, >. Пример:

```
i = 1
s = 0
while x(i) ~= 0 & i < 4
  s = s + x(i)
i = i + 1
end
```

Условное выражение

Как и во всех языках программирования, в языке Matlab есть конструкции для организации условного выполнения операторов. Одна из конструкций имеет следующий формат:

```
if <логическое выражение>
<операторы>
elseif <логическое выражение>
<операторы>
else
<операторы>
end
```

В условном выражении может присутствовать несколько блоков `elseif` или данный блок может отсутствовать. Также может отсутствовать блок `else`.

Логическое выражение имеет форму

```
выражение <оператор отношения> выражение,
```

где допустимы следующие операторы отношений : `==`, `~=`, `<=`, `>=`, `<`, `>`. Пример:

```
if a > 0
   b = 2
elseif a < 0
b = 0
else
   b = 1
end
```

В Matlab имеется еще одна конструкция для организации условного выполнения операторов:

```
switch <анализируемое выражение>
  case <выражение 1>
<операторы>
  case <выражение 2>
<операторы>
  ...
  otherwise
<операторы>
end
```

Оператор `switch` последовательно сравнивает анализируемое выражение с выражениями, записанными после `case`, и если выражения равны, то выполняются соответствующие операторы. Если не было найдено ни одного равенства, то выполняются операторы, записанные после `otherwise`.

Графические возможности языка Matlab

Среда обладает богатыми возможностями для графического представления массивов как в двухмерном, так и в трехмерном виде. В пакет Matlab включены демонстрационные программы для показа возможностей системы, в том числе и графических. Для вызова демонстрационных примеров необходимо в командном окне ввести команду `demo`.

Далее будут рассмотрены графические функции, которые необходимы для выполнения лабораторных работ.

Функция plot

Функция `plot` имеет следующий синтаксис:

```
plot(y)
plot(x,y)
plot(x,y,s)
plot(x1,y1,s1,x1,y1,s1)
```

Функция `plot(y)` строит график элементов одномерного массива *y* в зависимости от номера элемента. Если элементы массива *y* комплексные, то строится график, каждая точка которого определяется соответствующей действительной и мнимой частями числа.

Функция `plot(x,y)` соответствует построению обычной функции, когда одномерный массив `x` соответствует значениям аргумента, а одномерный массив `y` – значениям функции (табл. 1.4).

Таблица 1.4

Управление функцией `plot`

Тип линии	
Непрерывная	`-`
Штриховая	`--`
Двойной пунктир	`:`
Штрихпунктирная	`-.`
Тип точки	
Точка	`.`
Плюс	`+`
Звездочка	`*`
Кружок	`o`
Цвет	
Фиолетовый	`m`
Голубой	`c`
Красный	`r`
Зеленый	`g`
Синий	`b`
Черный	`k`

Функция `plot(x,y,s)` аналогична `plot(x,y)`, с той разницей, что текстовая строка `s` определяет цвет и стиль линии, а также вид точек графика. Строка`s` может содержать до трех символов из табл. 1.4.

Если цвет линии не указан, он выбирается по умолчанию из шести первых цветов, с желтого до синего, повторяясь циклически.

Функция `plot(x1,y1,s1,x2,y2,s2)` позволяет объединить несколько функций `plot(x, y, s)`. Пример:

```
x = -pi:pi/100:pi
y = sin(x)
plot(y)         % Результат показан на рис. 1.1,а
plot(x, y)      % Результат показан на рис.1.1,б
```

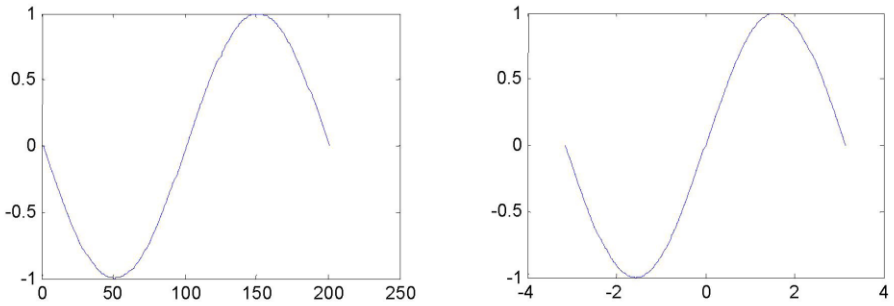


Рис. 1.1. Результат работы примера, демонстрирующего работу функции `plot`

Функция stem

Функция `stem` выводит график элементов одномерного массива в виде вертикальных линий, которые заканчиваются в точках графика, помечаемых кружочком.

Функция `stem` имеет следующий синтаксис:

```
stem(y)
stem(x,y)
stem(x,y,s)
```

Первый вариант функции строит зависимость значений элементов массива от номера элемента, вторая зависимость – `y(x)`, а третья – аналогична второй, за исключением того, что позволяет задавать цвет и стиль линий с помощью строки `s`. Правила задания стиля линий аналогичны функции `plot`. Пример:

```
x = -pi:pi/25:pi
y = sin(x)
stem(y)         % Результат показан на рис. 1.2,а
stem(x, y)      % Результат показан на рис.1.2,б
```

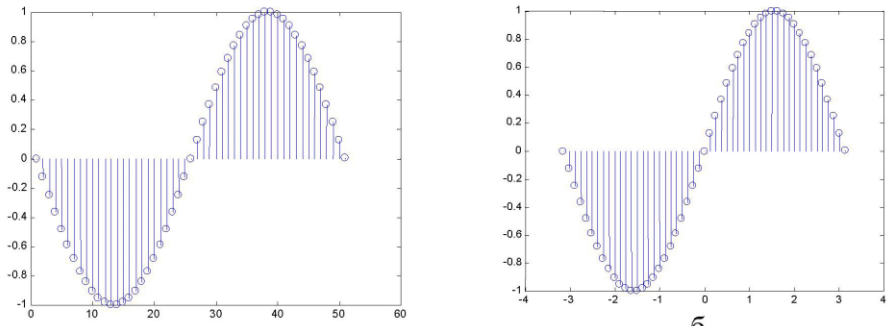


Рис. 1.2. Результат работы примера, демонстрирующего работу функции `stem`

Функция figure

Данная функция создает окно для вывода графика, т.е. дальнейший вывод графической информации будет осуществляться в данном окне, пока не встретится функция `figure`.

Функция subplot

Функция `subplot` предназначена для разбиения активного окна на области и выбора активной области для вывода графических данных. Функция `subplot` имеет следующий формат:

```
subplot(mnp);
subplot(m,n,p);
```

где `m` указывает, на сколько частей разбивается окно по вертикали, `n` – по горизонтали, `p` определяет номер области, в которую будут выводиться графические данные. Если числа `n`, `m` и `p` находятся в диапазоне от 1 до 9, то они обычно пишутся слитно (первый вариант записи функции), в противном случае они разделяются запятыми (второй вариант). Пример:

```
n = 0:99
x = sin(0.3*n)
y = cos(0.4*n)
subplot(211)
plot(x)
subplot(212)
plot(y)
```

Данная программа разбивает окно вывода на две части по вертикали и выводит график *x* в верхней части, а график *y*– в нижней. Результат работы программы показан на рис.1..

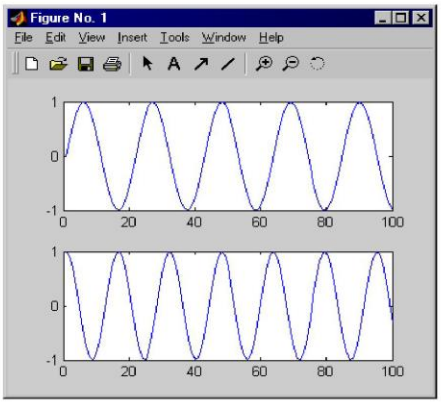


Рис. 1.3. Результат работы программы, демонстрирующей работу функции `subplot`

1.2 Базовые сигналы ЦОС

Наиболее важными последовательностями, которые часто используются в цифровой обработке сигналов, являются:

а)**единичный импульс**(рис. 1.2):

$$\delta(n) = \begin{cases} 1, & n = 0, \\ 0 & \text{иначе;} \end{cases}$$

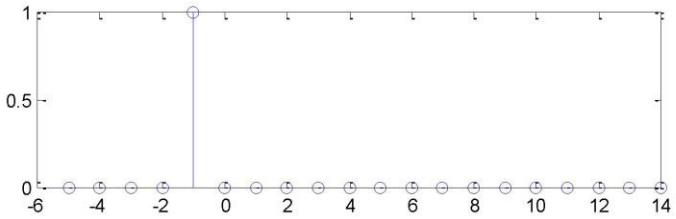


Рис. 1.2. Единичный импульс

б)**единичный скачок** (рис. 1.3):

$$u(n) = \begin{cases} 1, & n > 0, \\ 0 & \text{иначе;} \end{cases}$$

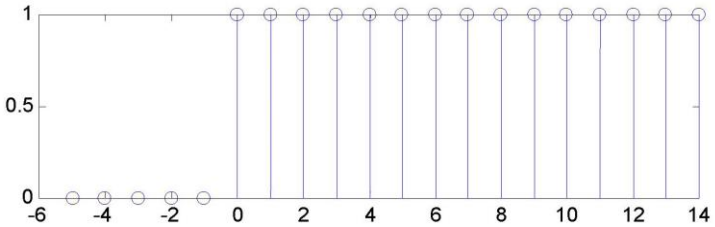


Рис. 1.3. Единичный скачок

в)**убывающая экспонента** (рис. 1.4):

$$g(n) = a^n u(n), \quad |a| < 1;$$

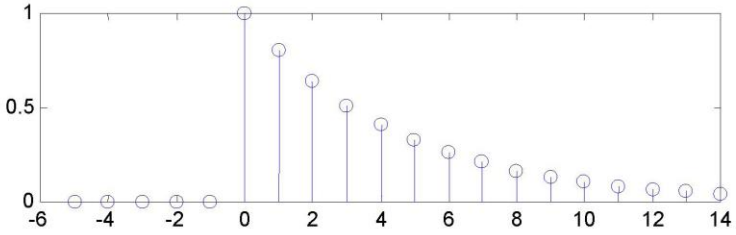


Рис. 1.4. Убывающая экспонента (a = 0,8)

г)**косинусоида** (рис.1.5):

$$h(n) = A \cdot \cos\left(\frac{2\pi n}{n_0} + \varphi\right) = A \cdot \cos\left(\omega n + \varphi\right);$$

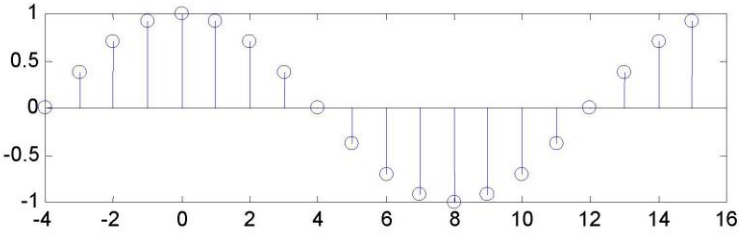


Рис. 1.5. Косинусоида $\cos\left(\frac{2\pi n}{16}\right)$

д)**комплексная экспонента**:

$$e^{j\omega n} = \cos(\omega n) + j\sin(\omega n).$$

1.3 Порядок выполнения работы

1.3.1 Сформируйте и постройте графики следующих последовательностей. Используйте для этого возможности вычисления синуса или косинуса от векторного аргумента:

$$\begin{aligned} x_1(n) &= \sin(\omega_1 n), && n = 0 \ldots 25, \\ x_2(n) &= \sin(\omega_1 n), && n = -15 \ldots 25, \\ x_3(n) &= \sin\left(\omega_2 n + \frac{\pi}{2}\right), && n = -10 \ldots 10, \\ x_4(n) &= \cos(\omega_3 n), && n = 0 \ldots 50. \end{aligned}$$

Значения $\omega_1 \dots \omega_3$ выберитеиз табл. 1.5.

Таблица 1.5

Значения $\omega_1 \dots \omega_3$

Номер варианта	1	2	3	4	5	6
ω_1	$\pi/13$	$\pi/14$	$\pi/15$	$\pi/16$	$\pi/17$	$\pi/18$
ω_2	π	$2\,\pi$	3π	π	2π	3π
ω_3	$\pi/\sqrt{19}$	$\pi/\sqrt{21}$	$\pi/\sqrt{23}$	$\pi/\sqrt{20}$	$\pi/\sqrt{22}$	$\pi/\sqrt{18}$

Упростите $x_3(n)$, чтобы не использовать тригонометрические функции.

1.3.2. Сформируйте и постройте график следующих последовательностей:

$$\begin{aligned} x_1(n) &= a_1\delta(n-b_1), & n &= 1 \dots 20, \\ x_2(n) &= a_2\delta(n), & n &= -15 \dots 15, \\ x_3(n) &= a_3u(n-b_3), & n &= 300 \dots 350, \\ x_4(n) &= a_4u(n+b_4), & n &= -10 \dots 0. \end{aligned}$$

Для выполнения задания напишите в Matlab функции `delta(n)`.и `unit_step(n)` для расчета дельта-функции и функции единичного скачка соответственно. Значения $a_1 \dots a_4$ и $b_1 \dots b_4$ выберитеиз табл. 1.6.

Таблица 1.6

Числовые значения $a_1 \dots a_4$ и $b_1 \dots b_4$

Номер варианта	1	2	3	4	5	6
a_1	1,2	1,3	1,4	1,5	1,6	1,7
b_1	4	5	6	7	8	9
a_2	0,6	0,7	0,8	0,9	1,1	1,2
a_3	1,8	1,7	1,6	1,5	1,4	1,3
b_3	310	312	318	321	328	333
a_4	3,8	4,0	4,2	4,4	4,6	4,8
b_4	3	4	5	6	7	8

1.3.3. Напишите функцию для формирования синусоиды, получаемой в результате дискретизации с частотой F_sнепрерывной синусоиды:

$$s(t) = A\sin(2\pi ft + \varphi_0),$$

где $t = \dots -3\Delta T, -2\Delta T, -\Delta T, 0, \Delta T, 2\Delta T, \dots \Delta T = 1/F_s$;$A$ – амплитуда;f – частота синусоиды;φ_0 – начальная фаза.

Функция должна иметь шесть входных аргументов: A, f, φ_0, F_s, t_0 – начальное время, t_1 – конечное. Выходными параметрами функции должны быть временные отметки (моменты времени в которые выполняется дискретизация синусоиды) и значения синусоиды в эти моменты.

Сформируйте синусоиду дискретного времени дискретизацией синусоиды непрерывного времени со следующими параметрами (табл. 1.7):

Таблица 1.7

Числовые значения параметров для формирования синусоид дискретного времени

Номер варианта	1	2	3	4	5	6
Частота сигнала	900 Гц	1 кГц	1,1 кГц	1,3 кГц	1,4 кГц	1,5 кГц
Начальная фаза	45°	60°	30°	45°	60°	30°
Нормализованная амплитуда	40	45	50	55	60	65
Частота дискретизации	8 кГц	8 кГц	8 кГц	10 кГц	10 кГц	10 кГц
Начальное время	0 с	0 с	0 с	0 с	0 с	0 с
Конечное время	6 мс	7 мс	8 мс	4 мс	5 мс	6 мс

1.3.4. Напишите функцию для формирования затухающей экспоненты $g(n) = a^n u(n)$. Постройте график функции, выбрав параметр a согласно варианту (табл. 1.8).

Таблица 1.8

Значения параметра a

Номер варианта	1	2	3	4	5	6
a	0,7	0,75	0,8	0,85	0,9	0,95

1.3.5. Для формирования комплексной экспоненты может быть использована формула Эйлера:

$$x(n) = z^n = r^n e^{j\theta n} = r^n(\cos(\theta n) + jsin(\theta n)),$$

где $z = re^{j\theta}$, $j^2 = -1$.

Используйте данное выражение со следующими параметрами (табл. 1.9).

Таблица 1.9

Параметры комплексных экспонент (угол θ задан в радианах)

Номер варианта	1	2	3	4	5	6
r	0,89	0,9	0,94	0,95	0,97	0,99
θ	30°	45°	60°	30°	45°	60°

Постройте график действительной и мнимой части для $n = 0 \ldots 100$. Как сказывается изменение θ? Постройте график, откладывая по оси ординат действительную часть, а по оси абсцисс – мнимую (должна получиться спираль). Поэкспериментируйте с углом θ для получения спиралей различного вида.

1.3.6. При помощи формулы Эйлера из комплексных экспонент можно получать функции косинуса и синуса:

$$\sin(\varphi) = \frac{1}{2j}\left(e^{j\varphi} - e^{-j\varphi}\right),$$
$$\cos(\varphi) = \frac{1}{2}\left(e^{j\varphi} + e^{-j\varphi}\right).$$

Сформируйте и постройте графики следующих последовательностей используя приведенные формулы (комплексная единица в Matlab задается как `1j` или `1i`):

$$\begin{aligned} x_1(n) &= \sin(\omega_1 n), & n &= 0 \ldots 25, \\ x_2(n) &= \cos(\omega_1 n), & n &= -15 \ldots 25, \\ x_3(n) &= a^n \sin(\omega_2 n), & n &= -20 \ldots 0, \\ x_4(n) &= b^n \cos(\omega_2 n), & n &= 0 \ldots 50. \end{aligned}$$

Значения параметров ω_1, ω_2,a, bвыберитеиз табл. 1.10.

Таблица 1.10

Значения параметров ω_1, ω_2, a, b

Номер варианта	1	2	3	4	5	6
ω_1	$\pi/3$	$\pi/6$	$\pi/5$	$\pi/\sqrt{7}$	$\pi/8$	$\pi/13$
ω_2	$\pi/4$	$\pi/9$	$\pi/16$	$\pi/6$	$\pi/\sqrt{11}$	$\pi/3$
a	1,1	1,15	1,4	1,3	1,2	1,05
b	0,89	0,9	0,94	0,95	0,97	0,99

ЛАБОРАТОРНАЯ РАБОТА №2. Разностные уравнения

ЦЕЛЬ РАБОТЫ – изучение разностных уравнений, программирование разностных уравнений в пакете Matlab.

2.1. Теоретические сведения

Разностное уравнение

В практическом применении ЦОС особую роль играет класс систем, которые описываются разностными уравнениями с фиксированными коэффициентами:

$$a_0 y(n) = \sum_{i=0}^{M} b_i x(n-i) - \sum_{i=1}^{N} a_i y(n-i), \qquad (2.1)$$

где $\{b_i\}$ – *коэффициенты прямой связи*, которые применяются к поступающему в систему сигналу; $x(n)$ и $\{a_i\}$ – *коэффициенты обратной связи*, которые применяются к выходному сигналу $y(n)$.

В зависимости от значения коэффициентов $\{b_i\}$ и $\{a_i\}$выражение (2.1)может описывать работу различных устройств (например интегратора, дифференциатора, полосового фильтра, фильтра нижних частот и т.д.).В Matlabразностные уравнения задаются двумя векторами коэффициентов: $\{b_i\}$ и $\{a_i\}$.

На рис. 2.1показана схема реализации разностного уравнения (2.1). Заметим, что чаще всего коэффициент a_0имеет единичное значение и поэтому не показан на схеме. Через z^{-1}на рисунке обозначен блок задержки (регистр), который хранит соответствующий задержанный отсчет. В Matlabлинейную систему реализует функция`y=filter(b,a,x)`, где `b` – коэффициенты прямой связи, `a` – коэффициенты обратной связи, `x` – входной сигнал. Ниже приводится пример использования функции `filter` для реализации линейной системы:

$$y(n) = 0{,}3x(n) + 0{,}6x(n-1) + 0{,}3x(n-2) - 0{,}9y(n-2). \qquad (2.2)$$

```
N = 64;
x = [1 zeros(1,N-1)]; % дельта-импульс
b = [0.3 0.6 0.3];
a = [1 0 0.9];
y = filter(b,a,x);
```

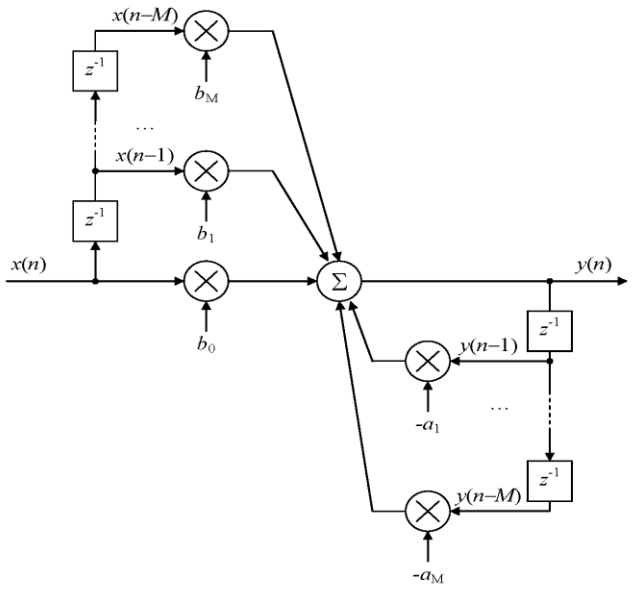


Рис. 2.1.Схема реализации разностного уравнения

На рис. 2.2показаны графики входного и выходного сигналов линейной системы из приведенного примера.

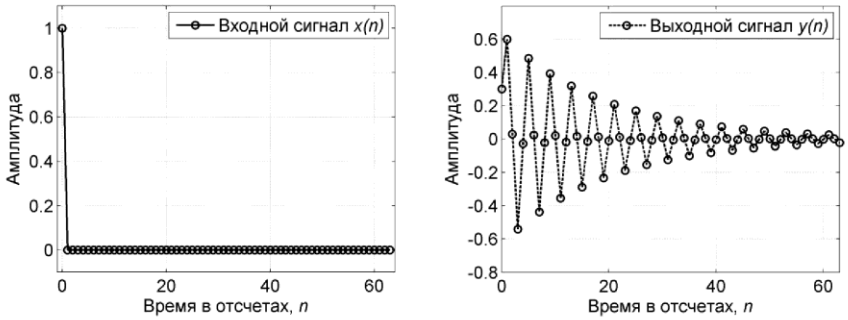


Рис. 2.2. Работа линейной системы

Импульсная характеристика

Системы, описываемые уравнением(2.1), делят на два класса:

а) рекурсивные;

б) нерекурсивные.

В *рекурсивных* системах выход $y[n]$зависит как от входного сигнала $x[n]$ и его предыстории $x[n-i]$, так и от предыдущих выходных значений$y[n-i]$. В *нерекурсивных* системах выходной сигнал зависит только от входного сигнала и его предыстории. Другими словами, у нерекурсивных систем все коэффициенты a_i(кроме a_0)равны нулю.

Помимо задания коэффициентов $\{b_i\}$и$\{a_i\}$ в выражении (2.1) линейную систему можно описать посредством ее *импульсной характеристики*. **Импульсная характеристика** – это временной сигнал, который генерирует система при подаче на ее вход дельта-импульса$\delta(n)$. Пример получения импульсной характеристики показан на рис. 2.2. Очевидно, что при подаче дельта-импульса на вход нерекурсивной системы ее выходной сигнал будет иметь конечную длительность, отчего такие системы называются системами с конечной импульсной характеристикой (КИХ). Для нерекурсивных систем вследствие наличия обратной связи характерна бесконечная импульсная характеристика (БИХ).

Собственная частота

Известно, что импульсная характеристика линейной системы может содержать колебания нескольких *собственных частот*. Эти собственные частоты определяются коэффициентами обратной связи $\{a_i\}$. Каждый корень характеристического полинома (p_i)

$$A(z) = 1 + \sum_{i=1}^{N} a_i z^{-i} \tag{2.3}$$

дает свой вклад в импульсную характеристику вида $p_i^n u(n)$.

В Matlabдля определения корней полинома имеется функция `root`. Для получения полной информации о работе данной функции `root`наберите к консольном окне системы Matlab`help root`.

Для разностных уравнений второго порядка (как например в (2.2)) характерно наличие двух различных собственных частот. Вследствие чего их импульсная характеристика описывается выражением

$$h(n) = (\alpha p_1^n + \beta p_2^n)u(n). \tag{2.4}$$

Линейная система

Линейная система, описываемая уравнением(2.1),входит в более общий класс *систем дискретного времени*:

$$y[n] = L[x[n]].$$

В этом выражении оператор $L[\cdot]$задает алгоритм, согласно которому из последовательности $x[n]$получается$y[n]$. Поскольку большинство систем,рассматриваемых в курсе ТиПЦОС,являются*линейными* (КИХ, БИХ-фильтры и дискретное преобразование Фурье), то необходимо знать их отличительные свойства.

Как же отличить линейную систему от любой другой? Существенным признаком линейной системы является выполнение***принципа суперпозиции***:

$$L[a_1x_1[n] + a_2x_2[n]] = a_1L[x_1[n]] + a_2L[x_2[n]].$$

Другое важное свойство линейных систем – это их ***инвариантность*** к сдвигу. Инвариантность означает, чтоесли сигнал $x[n]$ вызывает отклик $y[n]$, то задержанный сигнал $x[n-k]$будетвызывать задержанный отклик $y[n-k]$.Рассмотрим пример, поясняющий введенные понятия.

Пример 2.1. Нелинейная система

Пусть дискретная система описывается уравнением

$$y[n] = [x[n]]^2. \tag{2.5}$$

*Чтобы определить, является ли система**линейной,**проверим выполнение принципа суперпозиции для двух тестовых сигналов* $x_1[n]$ *и* $x_2[n]$*(рис. 2.3).*

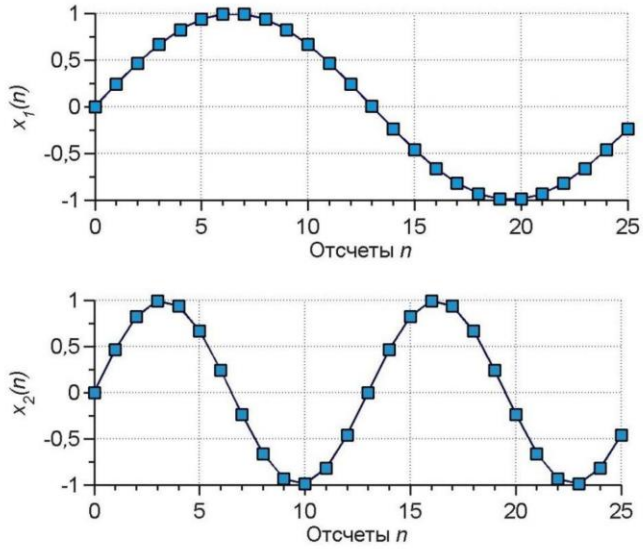


Рис. 2.3. Входные сигналы

Сложим эти два сигнала и пропустим через дискретную систему (2.5). Результат показан на рис.2.4.

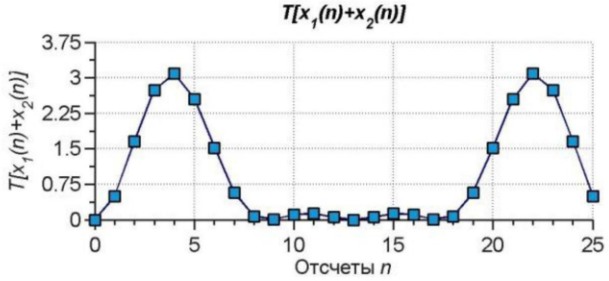


Рис. 2.4. Выходной сигнал после подачи на вход суммы сигналов $x_1[n]$и $x_2[n]$

Теперь пропустим по отдельности сигналы $x_1[n]$ *и*$x_2[n]$ *через рассматриваемую дискретную систему и затем сложим их, в результате чего получим сигнал, изображенный на рис2.5.*

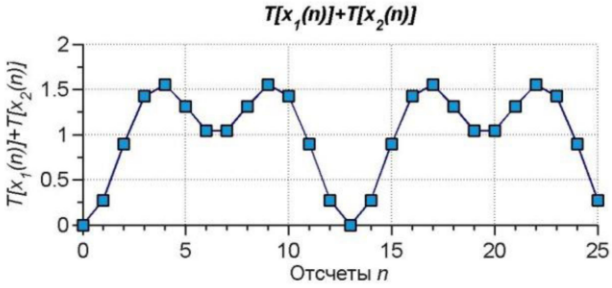


Рис. 2.5. Суммавыходных сигналов системы после подачи на вход сигналов $x_1[n]$и $x_2[n]$

Легко видеть, что графики на рис. 2.4 и 2.5различны. Из этого можно заключить, что (2.5) описывает ***нелинейную*** *систему.*

Z-преобразование

Z-преобразование дискретного сигнала $x(n)$имеет вид

$$X(z) = \sum_{n=0}^{\infty} x(n)z^{-n}.$$

Одной изособенностейz-преобразования является свойство*задержкиилисмещения*, которое имеет следующий смысл:если z-образ последовательности $x(n)$равен$X(z)$, то z-образ задержанной последовательности $x(n-m)$равен $z^{-m}X(z)$:

$$x(n) \to X(z),$$
$$x(n-m) \to z^{-m}X(z).$$

Z-преобразование бывает весьма полезно, когда необходимо определить стабильность линейной системы (везде далее под термином«линейная система» мы будем иметь в виду линейную систему, инвариантную к сдвигу). Система называется *стабильной*, если на любое ограниченное воздействие $x[n]$ее отклик также ограничен, т.е.

$$|x(nT)| \leq M_x < \infty$$

длявсех $n = 0, 1, 2 \ldots$, и

$$|y(n)| \leq M_y < \infty$$

для всех $n = 0, 1, 2 \ldots$.

Пример 2.2. Нестабильность линейной системы

В качестве тестового сигнала будем использовать единичный скачок длительностью 5 отсчетов (рис.2.6).

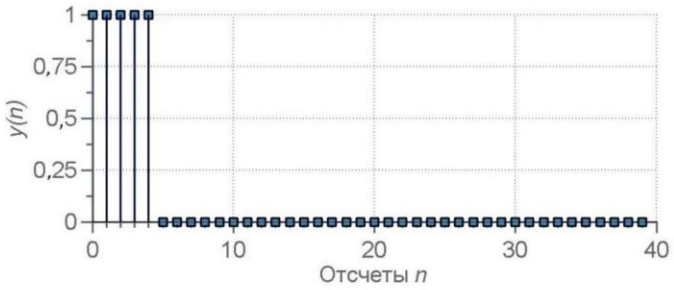


Рис. 2.6. Единичный отсчет длительностью пять отсчетов

Приведем отклик на единичный скачок нестабильной линейной системы (рис. 2.7).

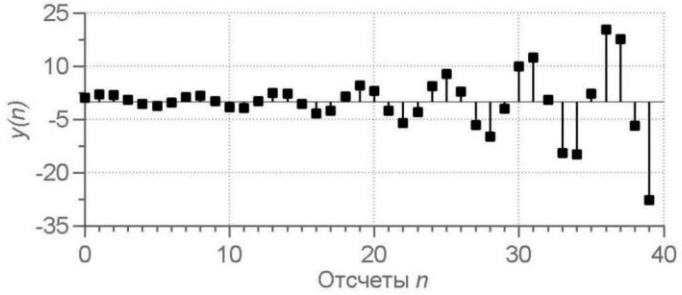


Рис. 2.7. Нестабильная система

Система нестабильна, поскольку отклик на сигнал ограниченной длительности не сходится к нулю и лишь увеличивается с увеличением n.

Приведем отклик на тот же единичный скачок стабильной системы (рис.2.8).

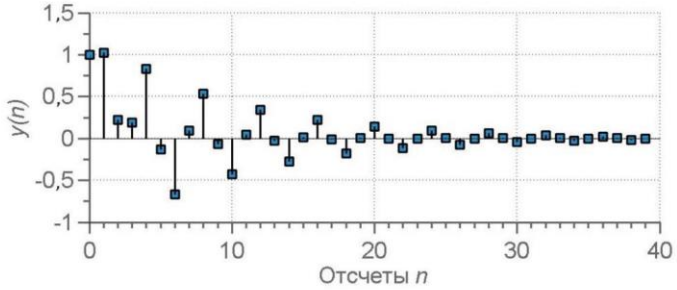


Рис. 2.8. Стабильная система

Легко увидеть, что отклик системы сходится к нулю – значит, система стабильна.

Часто необходимо определить стабильность линейной системы, зная лишь ее коэффициенты $\{b_i\}$и$\{a_i\}$из уравнения (2.1).Для этого необходимо обратиться к математическому аппаратуz-преобразования.

Вначале рассмотрим уравнение *свертки* во временной области:

$$y(n) = \sum_{i=0}^{M} h_i x(n-i), \tag{2.6}$$

где$x(n)$– это входной сигнал;$y(n)$ – выходной сигнал;$\{h_i\}$ –фиксированные коэффициенты.

Заметим, что сверткой описывается работа линейных нерекурсивных систем. Учитывая свойство задержки

$$h_0 x(n) \to h_0 X(z),$$
$$h_i x(n-i) \to h_i z^{-i} X(z),$$

z-преобразование для (2.6) принимает вид

$$Y(z) = \sum_{i=0}^{M} h_i z^{-i} X(z). \tag{2.7}$$

Вынося $X(z)$за знак суммы, получаем

$$Y(z) = X(z) \sum_{i=0}^{M} h_i z^{-i} = X(z) H(z). \tag{2.8}$$

Таким образом, получено еще одно свойство z-преобразования, которое показывает, что свертка двух сигналов во временной области эквивалента перемножению их образов в z-области.

Аналогичным образом можно поступить с разностным уравнением (2.1).Применяя свойство задержки z-преобразования

$$a_i x(n-i) \to a_i z^{-i} X(z),$$
$$b_i y(n-i) \to b_i z^{-i} Y(z),$$

получим

$$Y(z) = \sum_{i=0}^{M} b_i z^{-i} X(z) - \sum_{i=1}^{M} a_i z^{-i} Y(z).$$

Преобразуем выражение к виду

$$Y(z)\left(1 + \sum_{i=1}^{M} a_i z^{-i}\right) = X(z) \sum_{i=0}^{M} b_i z^{-i},$$

из которого легко получить *передаточную функцию* линейной дискретной системы:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{(1 + \sum_{i=1}^{M} a_i z^{-i})}. \quad (2.9)$$

Передаточнаяфункция$H(z)$полностью определяет линейную систему. Зная коэффициенты $\{b_i\}$и$\{a_i\}$, из выражения (2.9) можно записать разностное уравнение (2.1), которое описывает работу линейной системы.

Рассмотрим передаточную функцию в общем виде

$$H(z) = \frac{N(z)}{D(z)},$$

где $N(z)$и $D(z)$ – полиномы от z^{-1}порядка M.

Поскольку полином степени Mимеет ровно Mкорней, функцию $H(z)$можно разложить намножители и представить в виде:

$$H(z) = \frac{K(z-z_1)(z-z_2)\dots(z-z_M)}{(z-p_1)(z-p_2)\dots(z-p_M)}, \quad (2.10)$$

где z_i– i-й нуль;p_i–i-й полюс;K–коэффициент усиления.

Информацию, содержащуюся в уравнении (2.10), удобно и полезно изображать в виде диаграммы *нулей и полюсов*в z-плоскости (см. пример 2.3). На диаграмме крестиком обозначается положение полюсов, а кружком – положение нулей. Важной особенностью диаграммы нулей и полюсов является *единичная окружность*, которая задается уравнением $|z| = 1$. Для определения стабильности линейной системы существует правило: ***устойчивых (стабильных) систем все полюса должны лежать внутри единичной окружности в z-плоскости***.

Пример 2.3. Определение стабильности системы

Определить стабильность системы с коэффициентами $b_0 = 1$, $b_1 = 1$*и* $a_1 = -0{,}5$ *и изобразить схему, реализующую систему.*

Вначале построим передаточную функцию согласно формуле (2.9):

$$H(z) = \frac{1 + z^{-1}}{1 - 0{,}5z^{-1}}. \tag{2.11}$$

Затем преобразуем к виду (2.10):

$$H(z) = \frac{z + 1}{z - 0{,}5}. \tag{2.12}$$

Диаграмма нулей и полюсов для (2.12)показана на рис.2.9, из которой видно, что единственный полюс рассматриваемой системы $p_1 = 0{,}5$ *лежит внутри единичной окружности и, следовательно, линейная система стабильна.*

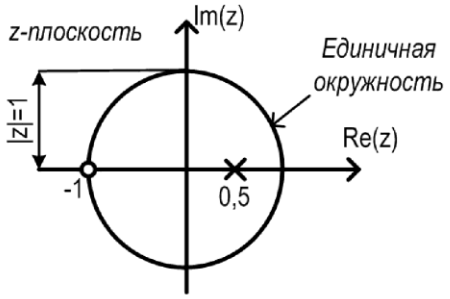


Рис. 2.9. Диаграмма нулей и полюсов

Чтобы изобразить схему, реализующую линейную систему, необходимо преобразовать передаточную функцию (2.11) к разностному уравнению:

$$y(n) = x(n) + x(n-1) + 0{,}5y(n-1). \tag{2.13}$$

На рис.2.10приведена схема, реализующая выражение (2.13): *через* z^{-1} *изображаются блоки задержки сигнала на один такт. При аппаратной реализации задержка реализуется в виде регистра, который хранит предыдущее значение входного или выходного отсчета.*

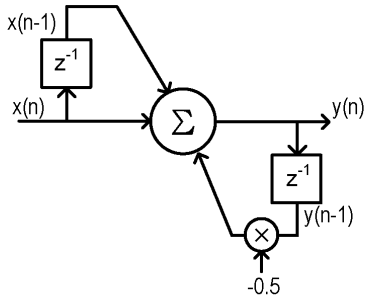


Рис. 2.10. Блок-схема фильтра

Частотная характеристика линейной системы

Для описания линейных систем в частотной области используется специальный входнойсигнал:

$$x(n) = e^{j\omega n}, -\infty < n < \infty. \tag{2.14}$$

Если такая последовательность поступает на вход линейной системы с *импульсной характеристикой*$h(n)$, то на выходе появится последовательность

$$y(n) = \sum_{m=0}^{M} h(m) \cdot e^{j\omega(n-m)} = e^{j\omega n} \sum_{m=0}^{M} h(m) \cdot e^{-j\omega m} = x(n)H\left(e^{j\omega}\right).$$

Таким образом, при подаче на вход сигналов вида (2.14)выходной сигнал совпадает со входным с точностью до комплексного множителя$H\left(e^{j\omega}\right)$. Этот комплексный коэффициент называется *частотной характеристикой системы* и выражается через ее импульсную характеристику следующим образом:

$$H\left(e^{j\omega}\right) = \sum_{m=0}^{M} h(m) \cdot e^{-j\omega m}.$$

Частотная характеристика является периодической функциейω, причем ее период равен2π. Эта периодичность связана со спецификой дискретного колебания: входная последовательность с частотой$(\omega + 2m\pi)$ $(m = \pm 1, \pm 2, \dots)$ не отличается от входной последовательности с частотойω, т.е.

$$\tilde{x}(n) = e^{j(\omega+2m\pi)n} = e^{j\omega n} = x(n).$$

Поскольку$H(e^{j\omega})$ – периодическая функция, то для полного описания достаточно задать ее на любом интервале длиной2π. Обычно для этой цели используют интервал$0 \le \omega \le 2\pi$.

2.2. Порядок выполнения работы

2.2.1. Для вариантов 1–3: найдите коэффициенты $\{b_i\}$ и$\{a_i\}$фильтра, который задан своей передаточной функцией.

$$H(z) = \frac{k(z - z_1)(z - z_2)}{(z - p_1)(z - p_2)}.$$

Таблица2.1

Числовые значения параметров k, z_1, z_2, p_1 и p_2

Номер варианта	k	z_1	z_2	p_1	p_2
1	0,09316	–0,99868 +0,05141j	–0,99868 –0,05141j	+0,51743 +0,40197j	+0,51743 –0,40197j
2	0,44405	–0,99992 +0,01297j	–0,99992 –0,01297j	–0,28370 +0,48321j	–0,28370 –0,48321j
3	0,17534	–0,99946 +0,03280j	–0,99946 –0,03280j	+0,28602 +0,48265j	+0,28602 –0,48265j

Для вариантов4–6: запишите передаточную функцию и найдите нули и полюса фильтра, которыйзадан в виде блок-схемы (рис. 2.11). Значения a_1, a_2, b_0, b_1 и b_2 выберите из табл. 2.2.

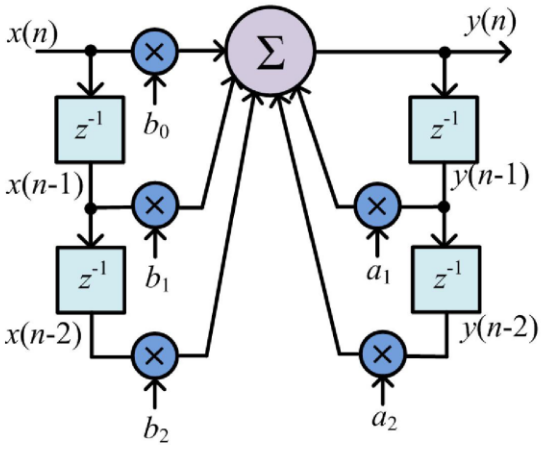


Рис. 2.11. Блок-схема линейной системы второго порядка

Таблица2.2

Числовые значения параметров a_1, a_2, b_0, b_1 и b_2

Номер варианта	a_1	a_2	b_0	b_1	b_2
4	0,22888	2,06144	0,86232	0,19501	1,73566
5	0,06797	1,53449	0,67549	0,05196	1,35097
6	0,09241	1,63060	0,70587	0,07190	1,41295

2.2.2. Напишите функцию, реализующую разностное уравнение (2.1). На вход функции поступают коэффициенты $\{b_i\}$,$\{a_i\}$ и входной сигнал $x[n]$. С помощью написанной функции постройте импульсную характеристику для фильтра из задания 2.2.1. Сравните результат с работой функции `filter`.

2.2.3. Определитесобственные частоты импульсной характеристики фильтра из задания 2.2.1 (см. формулу (2.3) и пояснение к ней). Найдите коэффициенты αи βиз уравнения (2.4) с которыми колебания собственных частот входят в импульсную характеристику. Для этого вычислите $h(n)$для любых двух значений n(например для $n = 0$и 1) и составьте систему линейных уравнений относительно неизвестных αи β. Решите эту систему с использованием Matlab-оператора «\» (backslash). Используя найденные значения αи β,постройте график $h(n)$согласно уравнению (2.4) и сравните его с результатом из задания 2.2.2.

2.2.4. Найдите отклик фильтра из задания 2.2.1 при воздействии на вход единичного скачка$x(n) = u(n - n_0),\ n = 1 \ldots N$. Выбрав достаточное значение N, определите, к какому значению сходится выходной сигнал фильтра. Это значение называют устоявшимся режимом фильтра, а переменную часть характеристики – переходной характеристикой. В качестве n_0 возьмите номер своего варианта (т.е. если выполняете вариант №2, то $n_0 = 2$).

2.2.5. Вычислите отклик фильтра на следующие сигналы (табл.ица 2.3)*.

Таблица 2.3

Сигналы для воздействия на линейную систему

Номер варианта	Задание
1–2	Периодический прямоугольный сигнал с частотой 10 Гц и сигнал такой же формы с частотой 20 Гц. Частота дискретизации сигналовf_s=80 Гц
3–4	Периодический треугольный сигнал с частотой 15 Гц и сигнал такой же формы с частотой 80 Гц. Частота дискретизации сигналовf_s=320 Гц
5	Периодический пилообразный сигнал с частотой 20 Гц и сигнал такой же формы с частотой 110 Гц. Частота дискретизации сигналовf_s=440 Гц
6	Периодический трапециевидный сигнал с частотой 18 Гц и сигнал такой же формы с частотой 130 Гц. Частота дискретизации сигналовf_s=520 Гц

* Функции для генерирования сигналов заданной формы можно найти в сетевом каталоге: "\\1-311-kafevs\Docs\ТиПЦОС\Функции для ЛР№2\".

2.2.6. Вычислите частотную характеристику линейной системы из задания 2.2.1по формуле

$$H\left(e^{j\omega}\right) = \frac{\sum_{k=0}^{M} b_k e^{j\omega k}}{\left(1 + \sum_{k=1}^{M} a_k e^{j\omega k}\right)}, \qquad \omega \in [0\ \ \pi].$$

Постройте график $A(\omega) = |H(e^{j\omega})|$ и $\varphi(\omega) = \arg H(e^{j\omega})$. Функция$A(\omega)$ – это амплитудно-частотная характеристика (АЧХ), а $\varphi(\omega)$– фазо-частотная характеристика (ФЧХ).

2.2.7. Постройте АЧХ и ФЧХ линейной системы из задания 2.2.1при помощи функции `freqz`.

2.2.8*. Алгоритм Карплуса–Стронга для имитации звука гитарной струны. Рассмотрим разностное уравнение:

$$y[n] = \alpha y[n - M] + x[n], \tag{2.15}$$

где $x[n]$ – входной сигнал;M – задержка;α – коэффициент затухания.

Предполагается, что значение задержкиM равно длине входного сигнала. Число генерируемых выходных значений $y[n]$ должно быть кратно M. Согласно алгоритму на вход необходимо подать случайную последовательность длинойM. Такую последовательность можно получить следующим образом:

```
x=(1/3)*randn(M,1);
```

Напишите в Matlabфункцию `y=ks_synthesis(x, alpha, P)`,которая генерирует выходную последовательность согласно разностному уравнению (2.15), длина выходной последовательности равна $M \times P$, где M – длина последовательности x.

Для получения «реальных аккордов» значение M должно быть выбрано исходя из частоты дискретизации сигнала F_s и «опорной» частоты F_0. Например A4 имеет «опорную» частоту 440 Гц, другие ноты могут быть получены из нее по формуле $F_0 = 440 \times 2^{n/12}$, где n – число полутонов между A4 и необходимой нотой. Например, открытый аккорд из песни Битлз «Hardday'snight» состоит из следующих нот D3, F3, G3, F4, A4, C5, G5. Ниже приведена заготовка Matlab-кода, необходимая для генерирования данного аккорда:

* Выполнение данного задания не является обязательным.

```
clear all; close all; clc;

% Параметры:
%
% - Fs       : частотадискретизации
% - F0       : частотаноты, формирующейаккорд
% - gain     : усилениеотдельныхнотваккорде
% - duration : длительностьаккордавсекундах
% - alpha    : ослабление в алгоритме Карплуса-Стронга

Fs = 48000;

% D2, A2, D3, F3, G3, F4, A4, C5, G5
F0 = 440*[2^-(31/12); 2^-(19/12); 2^-(16/12); 2^(-14/12); 2^-
(4/12); 1; 2^(3/12); 2^(10/12)];
gain = [1.2 3.0 1.0 2.2 1.0 1.0 1.0 3.5];
duration = 4;
alpha = 0.9785;

% Количествоотсчетовваккорде
nbsample_chord = Fs*duration;

first_duration = ceil(nbsample_chord / round(Fs/F0(1)));

% Инициализация
chord = zeros(nbsample_chord, 1);

for i = 1:length(F0)

  current_M = round(Fs/F0(i));
  current_duration = ceil(nbsample_chord/current_M);

  current_alpha = alpha^(first_duration/current_duration);

% Формированиевходногоивыходного сигналовалгоритма Карплуса-
% Стронга
x = (1/3)*rand(current_M, 1);
  y = ks_synthesis (x, current_alpha, current_duration);
y = y(1:nbsample_chord);

    % Добавление ноты к аккорду
chord = chord + gain(i) * y;
end

% Проигрывание аккорда
sound(chord, Fs);
```

ЛАБОРАТОРНАЯ РАБОТА №3.Дискретное преобразование Фурье

ЦЕЛЬ РАБОТЫ –Изучение дискретного преобразования Фурье (ДПФ) и его свойств,получение практических навыков использования ДПФ в пакете Matlab.

3.1. Теоретические сведения

Частотно-временной анализ. Преобразование Фурье

Преобразование Фурье является одним из мощнейших и эффективных средств для решения многих связанных с обработкой сигналов. *Жан Батист Жозеф Фурье* (1768–1830) предложил концепцию представления заданногосигнала в виде тригонометрического ряда из косинусов и синусов:

$$x(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos \omega_k t + b_k \sin \omega_k t), \qquad (3.1)$$

где $\omega_k = \frac{2\pi k}{T_p}$;$T_p$ – период сигнала.

Особенность разложения (3.1) в том, что оно применимо только для периодического сигнала (рис. 3.1).

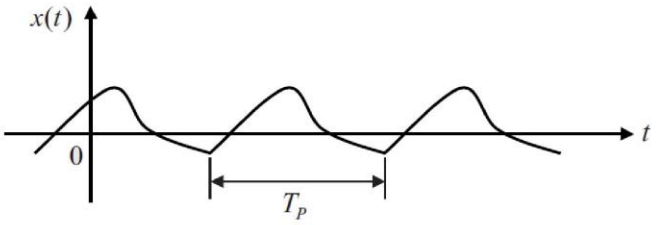


Рис. 3.1. Периодический сигнал

Самая низкочастотная составляющая сигнала, которая входит в (3.1),называется*основной частотой,*или *частотой основного тона,*$f_1 = 1/T_P$, поскольку все остальные частоты кратны ей.Частотные компоненты сигнала называются ***гармониками***. Например,частотная компонента $(a_2 \cos \omega_2 t + b_2 \sin \omega_2 t)$ является второй гармоникой. Коэффициент $a_0/2$являетсяпостоянной составляющей и представляет собой*среднее значение* сигнала за период. Причина, по которой стремятся представить сигнал в форме(3.1), заключается в том, что часто полезно разложить «сложный» сигнал в сумму «простых» – в данном случае косинусов и синусов. Не следует считать, что периодические сигналы редкость в природе. Например, на рис. 3.2 показан участок речевого сигнала (звук И), форма которого близка к периодической. Похожей периодичностью обладают почти все гласные (вокализованные) звуки.

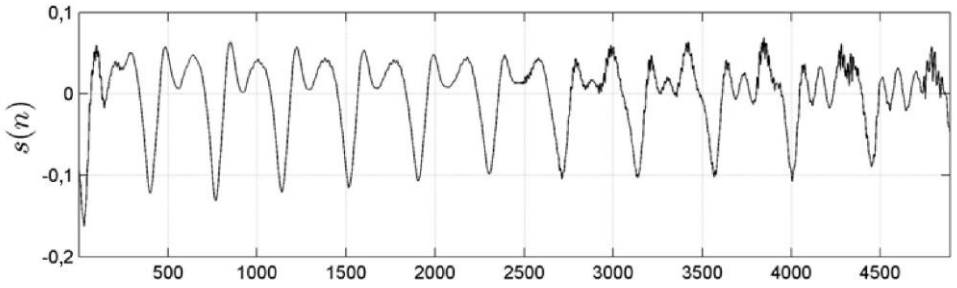


Рис. 3.2.Пример речевого сигнала

Ортогональность функций sin *и* cos

Часто возникает вопрос, почему для разложения функции в ряд (3.1) выбраны именно функции sin и cos? Так произошло благодаря тому, что эти функции обладают особым свойством *ортогональности*, известным из курса геометрии, где оно относилось к векторам. Оказывается, между функциями и векторами существует аналогия: как произвольный вектор можно представить в виде суммы ортогональных векторов (составляющих базис), так и произвольную функцию можно представить в виде суммы ортогональных функций (рис. 3.3).

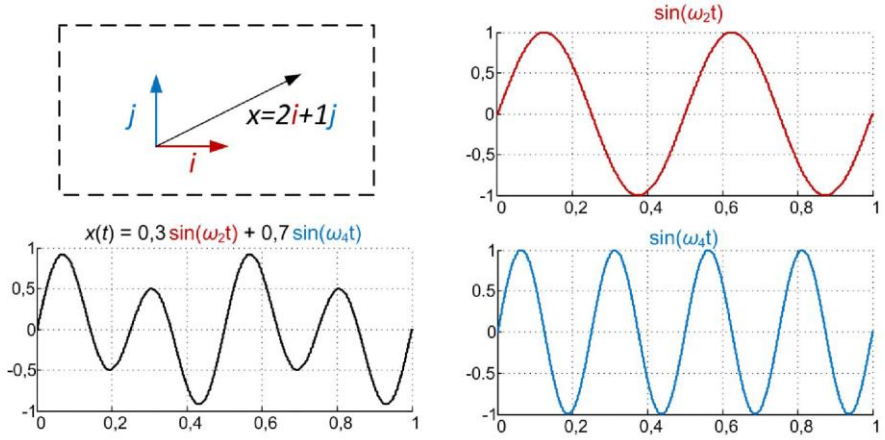


Рис. 3.3.Разложение в ортогональный базис вектора и функции

Различие при разложении вектора и функции в том, что количество базисных векторов конечно, а число базисных функций бесконечно*.

* Иногда говорят, что функция – это *бесконечномерный* вектор.

Как определить, является ли одна функция ортогональной по отношению к другой? Известно, что ортогональные векторы имеют нулевую проекцию друг на друга. С понятием проекции связано понятие скалярного произведения векторов

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = \sum_{i=1}^{n} a_i b_i,$$

которое равно нулю для ортогональных векторов. Для функций также есть аналог скалярного произведения:

$$\int_a^b f_1(t) \cdot f_2(t) dt.$$

Это выражение равно нулю, если функции $f_1(t)$и $f_2(t)$ортогональны на интервале $[a,\ b]$, иначе получаемое число показывает *проекцию* одной функции на другую. Исходя из того, что геометрический смысл интеграла – площадь под кривой, на рис. 3.4 иллюстрируется понятие скалярного произведения функций.

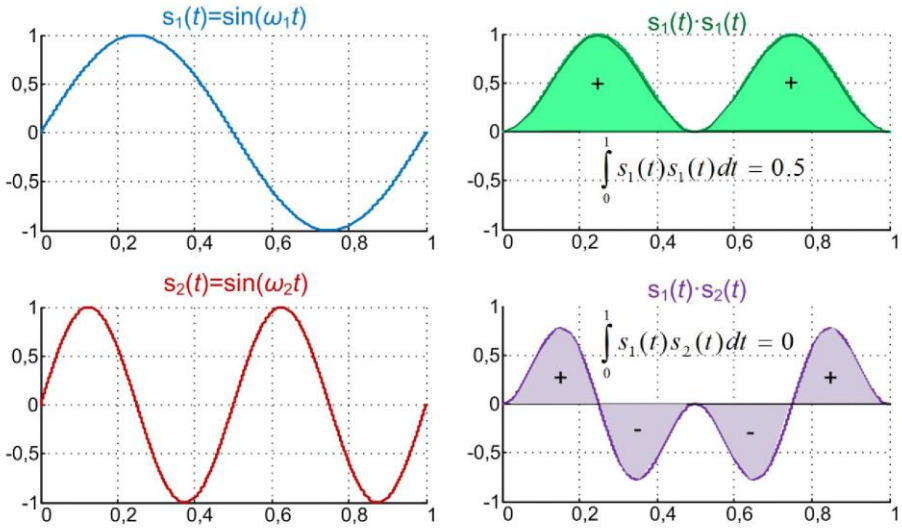


Рис. 3.4.Скалярное произведение функций

Учитывая введенные вначале обозначения, свойство ортогональности функций sin и cos записываются следующим образом:

$$\int_{-T/2}^{T/2} \cos \omega_k t \sin \omega_m t \, dt = 0, \quad \forall k, m, \tag{3.2}$$

$$\int_{-T/2}^{T/2} \cos\omega_k t \cos\omega_m t\, dt = \begin{cases} 0, & k \neq m \\ T/2, & k = m, \end{cases}$$

$$\int_{-T/2}^{T/2} \sin\omega_k t \sin\omega_m t\, dt = \begin{cases} 0, & k \neq m \\ T/2, & k = m. \end{cases}$$

Фурье-анализ

Основная задача Фурье-анализа найти коэффициенты a_k и b_k в выражении(3.1). Для этого необходимо умножить левую и правую части (3.1)на $\cos\omega_m t$:

$$x(t)\cos\omega_m t = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos\omega_k t \cos\omega_m t + b_k \sin\omega_k t \cos\omega_m t).$$

Интегрируя обе части по переменной$t \in [-T/2,\ T/2]$,получаем

$$\int_{-T/2}^{T/2} x(t)\cos\omega_m t\, dt = \frac{a_0}{2}\int_{-T/2}^{T/2} \cos\omega_m t\, dt +$$

$$+ \sum_{k=1}^{\infty}\left(a_k \int_{-T/2}^{T/2} \cos\omega_k t \cos\omega_m t\, dt + b_k \int_{-T/2}^{T/2} \sin\omega_k t \cos\omega_m t\, dt\right).$$

Используя свойство ортогональности (3.2), легко можно найти выражения для коэффициентов a_k и b_k:

$$a_k = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\cos\omega_k t\, dt,$$

$$b_k = \frac{2}{T}\int_{-T/2}^{T/2} x(t)\sin\omega_k t\, dt.$$

Часто бывает полезным следующее разложение функции $x(t)$, которое может быть получено из (3.1):

$$x(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} M_k \cos(\omega_k t + \varphi_k),$$

где $\omega_1 = 2\pi/T$ –основная частота (частота основного тона); $M_k = \sqrt{a_k^2 + b_k^2}$ – амплитуда компоненты сигнала на частоте $k\omega_1$; $\varphi_k = \text{arctg}(-b_k/a_k)$ – фаза компоненты сигнала на частоте $k\omega_1$.

Пример 3.1. Сигналмеандр

В качестве примера рассмотрим ряд Фурье для периодическогосигналах$x(t+nT) = x(t)$,*который определен следующим образом (рис. 3.5):*

$$x(t) = \begin{cases} -1, & -T/2 < t < 0, \\ 1, & 0 \le t < T/2. \end{cases} \tag{3.3}$$

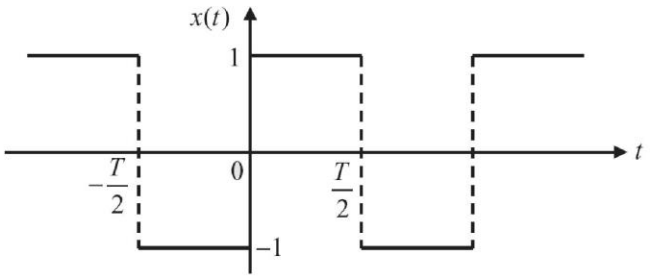


Рис. 3.5.Меандр

Легко заметить, что среднее значение сигнала равно нулю:

$$\frac{a_0}{2} = \frac{1}{T}\int_{-T/2}^{T/2} x(t)dt = 0,$$

а коэффициенты a_k *и* b_k *равны*

$$a_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)\cos(\omega_k t)\,dt = \frac{2}{T}\left[\int_{-\frac{T}{2}}^{0} -\cos(\omega_k t)\,dt + \int_{0}^{\frac{T}{2}} \cos(\omega_k t)\,dt\right] = 0,$$

$$b_k = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)\cos(\omega_k t)\,dt = \frac{2}{T}\left[\int_{-\frac{T}{2}}^{0} -\cos(\omega_k t)\,dt + \int_{0}^{\frac{T}{2}} \sin(\omega_k t)\,dt\right] =$$

$$= \frac{2}{k\pi}(1 - \cos k\pi).$$

Подставляя полученные значения в (3.1),получим выражение для сигнала:

$$x(t) = \frac{4}{\pi}\left[sin(\omega_1 t) + \frac{1}{3}sin(3\omega_1 t) + \frac{1}{5}sin(5\omega_1 t) + \cdots\right]. \quad (3.4)$$

Необходимо отметить, что в разложении участвуют только функции sin. *Это происходит из-за того, что функция меандра является нечетной и для ее представления не нужнычетные функции(*cos*).*

Полезно рассмотреть,как частичные суммы(3.4)аппроксимируют исходный сигналx(t).Обозначим через $S_n(t)$ сумму первых nчленов в (3.4). Графики первых трех частичных сумм приведены на рис. 3.6.

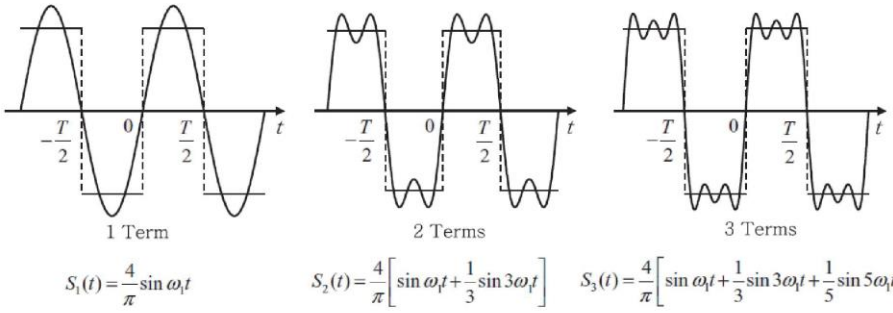


Рис. 3.6.Частичные суммы, аппроксимирующие сигнал меандр

Эффект Гиббса

Поведение частичных сумм ряда Фурье в точке разрыва функции называют *эффектом Гиббса*. Создается впечатление, что колебания в точках разрыва исчезнут, если просуммировать больше членов ряда, однако этого не происходит (рис. 3.7). Заметьте, что от количества слагаемых в ряде Фурье амплитуда выброса не изменяется.

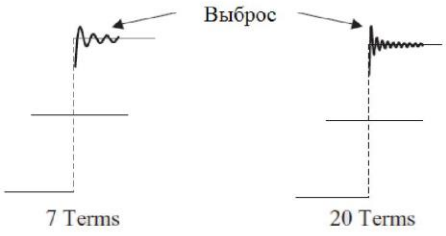


Рис. 3.7.Эффект Гиббса

Комплексная запись ряда Фурье

Наиболее часто ряд Фурье записывается в терминах комплексных экспонент $e^{j\omega_k t}$,где $\omega_k = 2\pi k/T_P$. Переход к комплексной форме записи ряда Фурье можно выполнить, если учесть, что

$$\cos\theta = \frac{1}{2}\left(e^{j\theta} + e^{-j\theta}\right) \quad \text{и} \quad \sin\theta = \frac{1}{2j}\left(e^{j\theta} - e^{-j\theta}\right). \tag{3.5}$$

Используя эти соотношения, выражение (3.1) можно преобразовать к виду

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{-j\omega_k t}, \tag{3.6}$$

$$c_k = \frac{1}{T_P}\int_0^{T_P} x(t) e^{-j\omega_k t} dt,$$

где комплексные коэффициенты c_k связаны с a_k и b_k следующим образом:

$$c_k = (a_k - jb_k)/2.$$

Коэффициент c_k несет в себе информацию об амплитуде синусной и косинусной волны k-й компоненты разложения Фурье (илиk-й гармоники). Форма (3.6)связана сразложением

$$x(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} M_k \cos(\omega_k t + \varphi_k) \tag{3.7}$$

следующим образом*:

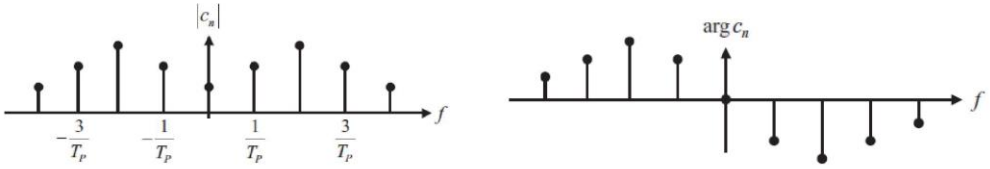
$$|c_k| = \sqrt{a_k^2 + b_k^2} = M_k, \quad \arg(c_k) = \operatorname{arctg}\left(\frac{-b_k}{a_k}\right) = \varphi_k.$$

Заметим, что в (3.6)«кирпичиками», из которых складывается сигнал, являются *комплексные экспоненты* $e^{-j\omega_k t}$. Коэффициенты c_k также имеют комплексные значения и несут в себе информацию не только об амплитуде, но и о фазеk-й гармоники. Кроме того, в (3.6)возникают компонентыс«отрицательной частотой». Понятие отрицательной частотыимеет чисто математическую природу и берет свои истоки в (3.5).

Спектр сигнала

Для пояснения понятия спектра сигнала обратимся к комплексной форме ряда Фурье (3.6). График зависимости$|c_k|$ от частоты ω_kназывается *амплитудным спектром* сигнала $x(t)$. График зависимости фаз частотных компонент $\arg c_k$ в от частоты ω_kназывается *фазовым спектром* сигнала $x(t)$. Примеры амплитудного и фазового спектров показаны на рис. 3.8.

* В Matlab для вычисления модуля комплексного числа используется функция `abs`, а для вычисления аргумента комплексного числа – функция `angle`.



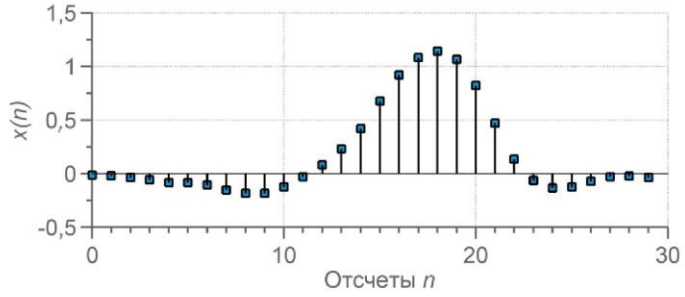
а б

Рис.3.8.Амплитудный (а) и фазовый (б) спектры

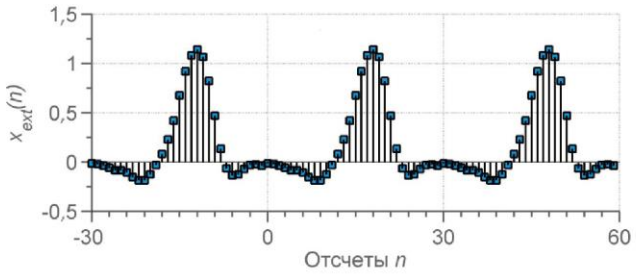
Альтернативным способом изобразить спектр сигнала является построение графиков для M_kи φ_k из выражения (3.7).

Дискретное преобразование Фурье

ДПФ является аналогом ряда Фурье для сигнала, определенного в дискретные моменты времени. В ДПФ «предполагается», что вне заданного интервала сигнал имеет периодическое продолжение, как и в случае ряда Фурье (сравнитерис. 3.9 и 3.1).



а



б

Рис. 3.9.Дискретный сигнал (а) и его ДПФ-расширение (б)

По этой причине полный период функциисin с точки зрения ДПФ представляется, как показано на рис. 3.10 (без нулевого отсчета в конце). Так проис-

ходит потому, что именно в этом случае ДПФ-расширение сигнала тоже является синусом (рис. 3.11).

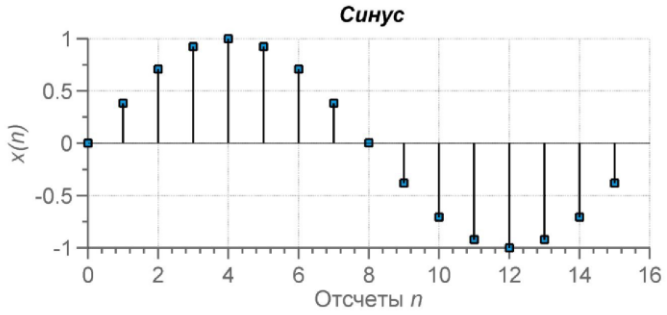


Рис. 3.10.Один период дискретного синуса

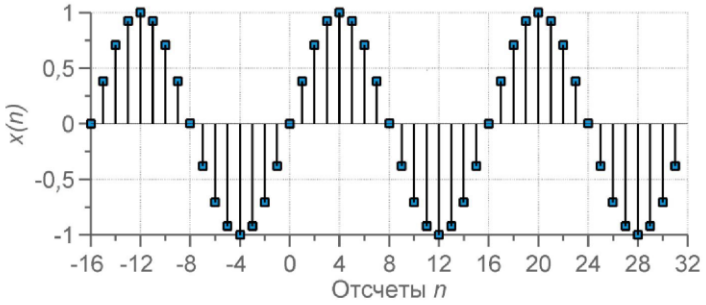


Рис.3.11.Дискретный синус: ДПФ-расширение

Что делает ДПФ?

Как показано на рис.3.12, ДПФпереводитN точек входного сигнала в два выходных сигнала из $N/2+1$ точек. Входной сигнал – это сигнал, который подвергается разложению, а два выходных сигнала содержат *амплитуды* синусов и косинусов. Как правило, входной сигнал определен во *временной области*. Это вызвано тем, что большинство сигналов, встречающихся в ЦОС, состоят из отсчетов, полученных через равные интервалы *времени*. Термин *частотная область* используется для описания амплитуд синусов и косинусов (включая специальное масштабирование, которое будет объяснено позже).

Частотная область содержит точно такую же информацию, как и временная, но в другой форме. Если известно представление сигнала в одной из областей, всегда можноего представить в другой. Если известен сигнал во временной области, то процесс вычисления сигнала в частотной области называется *разложением, анализом*, *прямым ДПФ* или просто *ДПФ*. Если известен сигнал в частотной области, то вычисление сигнала во временной области называется *синтезом,* или *обратным ДПФ*. Как синтез, так и анализ могут быть представлены в виде формул и компьютерных алгоритмов.

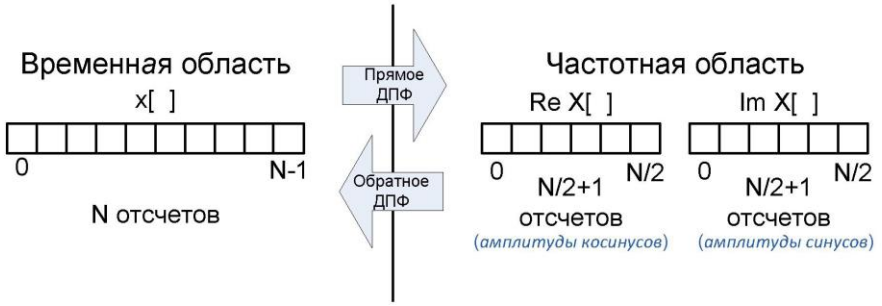


Рис.3.12.ДПФ:представление сигнала во временной и частотной областях

Число отсчетов во временной области обычно обозначается *переменной*N. Значение N может быть любой положительной величиной, но чаще всего его выбирают кратным степени числа 2, т. е. 128, 256, 512 и т. д. Это происходит по следующим двум причинам: 1) при хранении цифровых данных адресное пространство памяти кратно степени двойки; 2) большинство эффективных алгоритмов для вычисления ДПФ используют N, кратное степени двойки.

На рис. 3.13 показан пример ДПФ для $N = 32$. Сигнал во временной области представляется в виде массива $x[0],\ldots x[31]$, сигнал в частотной области – в виде двух массивов $Re\,X[0]\ldots Re\,X[16]$ и $Im\,X[0]\ldots Im\,X[16]$. Заметим, что 32 точки временной области соответствуют 17 точкам в каждом массиве частотной области с номерами частот от 0 до 16, т. е. N точек временной области соответствуют $N/2 + 1$ точек частотной области (не $N/2$ точек). Опускание этой дополнительной точки является типичной ошибкой при программировании ДПФ.

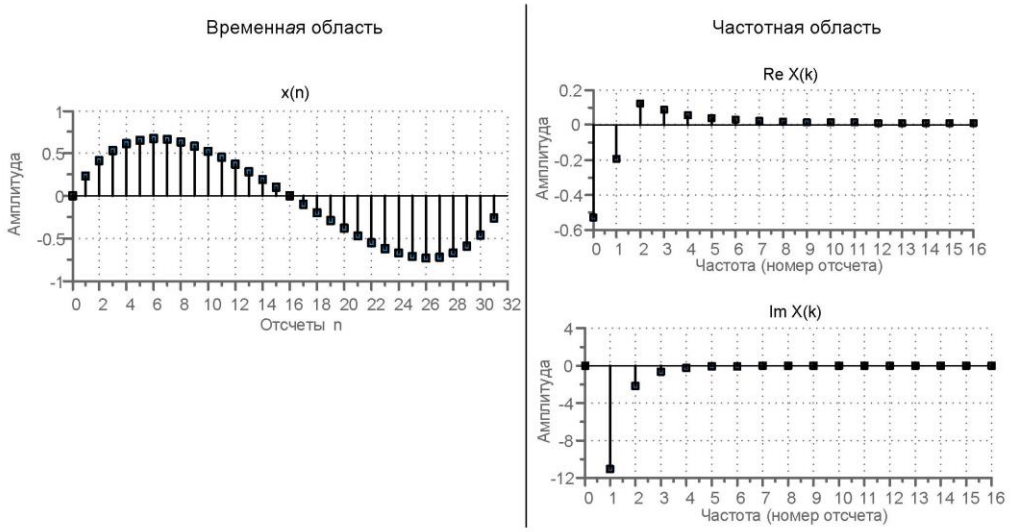


Рис. 3.13.Пример ДПФ для 32 отсчетов: представление сигнала во временной и частотной областях

Масштаб частотной оси

Горизонтальная ось в частотной области может быть обозначена четырьмя различными способами, все они общеприняты в ЦОС. При *первом способе* горизонтальная ось размечена номерами отсчетов от 0 до $N/2$. В этом случае индексы в частотной области есть целые числа, например, $Re\,X[k]$ и $Im\,X[k]$, где k меняется от 0 до $N/2$ с шагом единица. Этот способ часто применяют программисты, потому что он соответствует коду, который используется для доступа к массиву(см. рис. 3.13).

Во*втором способе* горизонтальная ось размечена *в долях частоты дискретизации*. Это означает, что величина вдоль горизонтальной оси всегда изменяется от 0 до 0,5, т. к. дискретные данные могут содержать частоты только от постояннойсоставляющей до половины частоты дискретизации. Для этого обозначения используется переменная f. Реальная и мнимая части записываются как $Re\,X[f]$ и $Im\,X[f]$, где f принимает значения $N/2+1$различных значений, равномерно расположенных в интервале от0до0,5. Связь между первым и вторым способом масштабирования частотной оси выражается следующим образом:

$$f = \frac{k}{N}.$$

Третий способ по сути совпадает со вторым, за исключением того, что горизонтальная ось умножена на 2π. Индекс, используемый для разметки, – ω. При этом обозначении реальная и мнимая части записываются как$Re\,X[\omega]$ и $Im\,X[\omega]$, где ω принимает значения $N/2+1$ величин, равномерно расположенных между 0 иπ. Параметр ω называется *нормированной круговой частотой*. Этот способ обозначений удобен тем, что позволяетсократить математические выкладки. Например, рассмотрим запись косинуса,используя три различных способа задания частотной оси:

k:c$(n) = \cos(2\pi kn/N)$,

f:$c(n) = \cos(2\pi fn)$,

ω:$c(n) = \cos(\omega n)$.

Четвертый способ заключается в разметке горизонтальной оси в величинах аналоговых частот, используемых в *практических* приложениях. Например, если в системе применяется частота дискретизации 10 кГц (т.е. 10 000 отсчетов в секунду), то на графике частота будет меняться от 0 до 5 кГц. Этот способ хорош для представления частотных данных в обозначениях, используемых на практике. Недостаток заключается в том, что он привязан к конкретному значению частоты дискретизации входного сигнала, и поэтому неприменим,когда ведется речь от абстрактном сигнале.

Базисные функции ДПФ

Синусы и косинусы, используемые в ДПФ, обычно называются *базисными функциями*. Эти базисные функции имеют *единичную* амплитуду. Выходом ДПФ является набор чисел, которые представляют амплитуды базисных функций. Если определить каждую амплитуду (в частотной области) соответствующего синуса или косинуса, то в результате будут получены значения, которые отличаются от реальных на*масштабирующий коэффициент*.

Базисные функции ДПФ определяются следующими выражениями:

$$c_k[n] = \cos\left(\frac{2\pi kn}{N}\right), \qquad s_k[n] = \sin\left(\frac{2\pi kn}{N}\right),$$

где$n = 0,1 \dots N-1; k = 0,1 \dots N/2; c_k[n]$ – косинусная волна с амплитудой, содержащейся в $Re\,X[k]; s_k[n]$ – синусная волна с амплитудой, содержащейся в $Im\,X[k]$. Например, рис. 3.14 показывает некоторые из 17 синусов и 17 косинусов, используемых в32-точечном ДПФ.

Рассмотрим некоторые из этих базисных функций. Косинусная волна $c_0[\,]$ имеет нулевую частоту, т.е. это постоянныйсигнал сединичной амплитудой. Это означает, что $Re\,X[0]$ содержит величину, усредненную по всем точкам сигнала во временной области. В электронике можно было бы сказать, что $Re\,X[0]$ содержит *постоянную составляющую*. Синусная волна с нулевой частотой$s_0[\,]$ состоит из одних нулей. Она не воздействует на формирование сигнала.

Синусоиды $c_2[n]$ и $s_2[n]$ совершают 2 полных колебания на отрезке из N точек. Им соответствуют $Re\,X[2]$ и $Im\,X[2]$. Подобным образом$c_{10}[n]$ и $s_{10}[n]$ совершают 10 полных колебаний на N точках. Этим синусоидам соответствуют амплитуды $Re\,X[10]$ и $Im\,X[10]$. Проблема заключается в том, что отсчеты $c_{10}[n]$ и $s_{10}[n]$ не отражают *наглядно* характер синусных и косинусных волн. Если бы не было непрерывной кривой на графике, было бы трудно определить вид сигнала.

Самые высокие частоты в базисных функциях – это $c_{N/2}$и$s_{N/2}$ (для рассматриваемого примера $c_{16}[n]$ и $s_{16}[n]$). Дискретная косинусная волна чередуется между 1 и -1, что может быть интерпретировано как отсчеты по пикам непрерывной синусоиды. В противоположность этому дискретная синусная волна состоит из всех нулей в результате попадания отсчетов на *нулевые пересечения*. Это делает величину $Im\,X[N/2]$ такой же, как и $Im\,X[0]$: всегда равной нулю и не влияющей на синтез сигнала во временной области,что дает ответ на вопрос, почему из N отсчетов на входе ДПФ образуется $N+2$ отсчетов на выходе. Ответ в том, чтодва выходных отсчета не содержат информацию, позволяя другим N быть полностью независимыми.

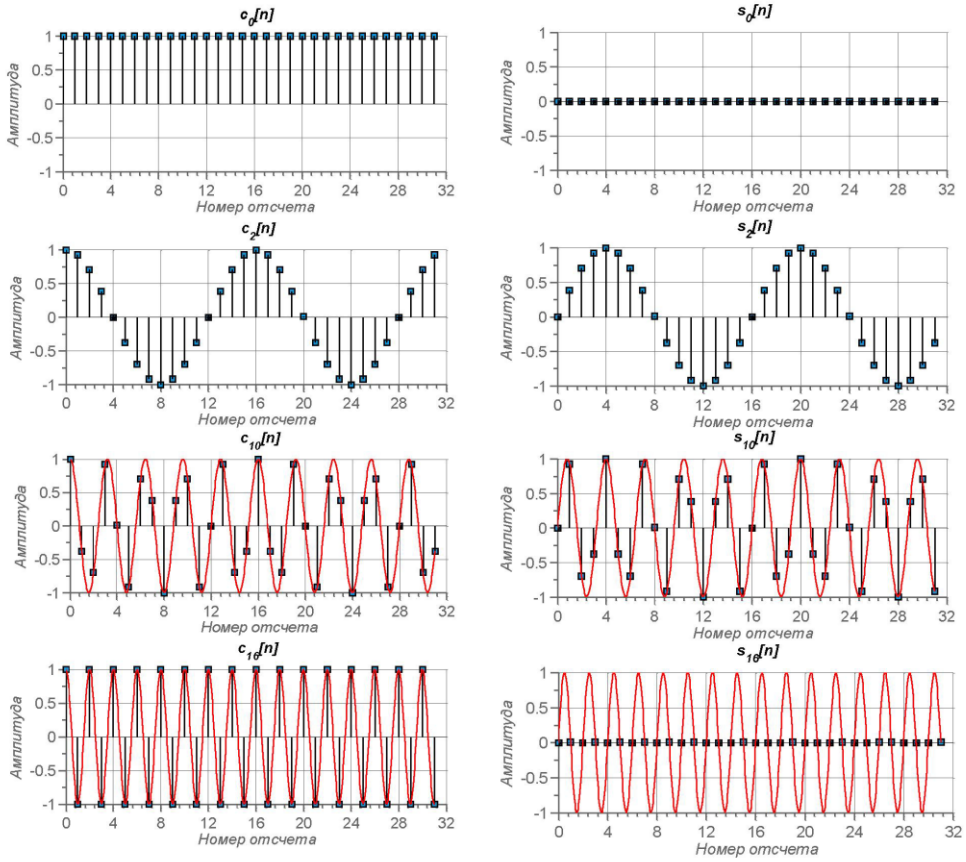


Рис.3.14.Базовые функции 32-точечного ДПФ

Вычисление ДПФ

Выражения для вычисления прямого ДПФ:

$$ReX[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi kn}{N}\right), ImX(k) = -\sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi kn}{N}\right),$$

где $x[n]$ – анализируемый сигнал во временной области;$Re\ X(k)$и $Im\ X(k)$– вычисляемые сигналы в частотной области; индекс k изменяется от 0 до $N/2$.

Выражение для ДПФ можно записать и в комплексной форме, используя формулу Эйлера $e^{j\varphi} = \cos\varphi + j\sin\varphi$:

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}, \qquad k = 0, 1 \ldots N/2.$$

Уравнение синтеза или обратного ДПФ можно записать как

$$x[n] = \sum_{k=0}^{N/2} Re\,\hat{X}(k) \cos\left(\frac{2\pi nk}{N}\right) + \sum_{k=0}^{N/2} Im\,\hat{X}(k) \sin\left(\frac{2\pi nk}{N}\right). \quad (3.8)$$

Иными словами, любой сигнал из Nточек $x[n]$ может быть создан суммированием $N/2+1$ косинусных волн и $N/2+1$ синусных волн. Амплитуды косинусных и синусных волн содержатся в массивах $Re\hat{X}(k)$ и$Im\hat{X}(k)$ соответственно.

В выражении (3.8) массивы обозначаются$Re\,\hat{X}(k)$ и$Im\,\hat{X}(k)$ вместо $Re\,X(k)$ и $Im\,X(k)$. Это вызвано тем, что амплитуды, необходимые для синтеза, слегка отличаются от значений сигнала в частотной области. Это влияние коэффициента масштабирования.Для получения правильного результата требуется нормализация.Незнание этого факта часто приводит к ошибкам при программировании формул ДПФ. Формально масштабирование выглядит следующим образом:

$$Re\,\hat{X}(k) = \frac{Re\,X(k)}{N/2}\ ,\ \ Im\,\hat{X}(k) = -\frac{Im\,X(k)}{N/2}\ ,$$

за исключением двух специальных случаев:

$$Re\,\hat{X}(0) = \frac{Re\,X(0)}{N}\ ,\ \ Re\,\hat{X}(N/2) = \frac{Re\,X(N/2)}{N}\ .$$

По аналогии с выражением (3.7) формулу для обратного преобразования Фурье можно записать в следующем виде:

$$x[n] = \sum_{k=0}^{N/2} M_k \cos\left(\frac{2\pi nk}{N} + \varphi_k\right), \quad (3.9)$$

где

$$M_k = \sqrt{\left(Re\,\hat{X}(k)\right)^2 + \left(Im\,\hat{X}(k)\right)^2},\ \ \varphi_k = \mathrm{arctg}\frac{-Im\,\hat{X}(k)}{Re\,\hat{X}(k)}.$$

Выражение (3.9) показывает, что с помощью ДПФ можно представить сигнал в виде суммы гармонических колебаний. При этом каждая гармоника имеет свою амплитуду M_k и фазу φ_k. Для правильного вычисления фазы в среде Matlabпредусмотрена функция `atan2(imX, reX)`, которая возвращает значение угла в диапазоне $[-\pi,\ \pi]$.

Существует и комплексная форма записи обратного ДПФ:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{\frac{j2\pi kn}{N}}, \qquad n = 0,1,\dots N-1.$$

Вычисление ДПФ в пакете Matlab

В пакете Matlab для вычисления дискретного преобразования Фурье используется функция `FFT(X)`. В случае матричного аргумента ДПФ рассчитывается для каждого столбца матрицы. Может оказаться полезным явное указание размера преобразования: `FFT(X,N)`. Если реальная длина вектора X меньше заданной размерности N, он дополняется нулями. При этом можно получить больше частотных отчетов и, следовательно, улучшить условия различения синусоидальных компонент сигнала. Если длина входного вектора больше N, лишние отсчеты отсекаются.

Обратное дискретное преобразование Фурье вычисляется с помощью функции `IFFT(X)`, которая используется аналогично предыдущей.

1.4 Порядок выполнения работы

3.2.1. Разработайте функцию `DFT`, вычисляющую ДПФ от входного вектора, не используя функцию Matlab`FFT`, и рисующую графики действительной и мнимой частей результата преобразования. Сравните результаты работы своей функции с функцией Matlab `FFT`.

3.2.2. Предположим, что задан входной сигнал $x[n]$значения ДПФ сигнала $X(k)$. Разработайте в среде Matlab функцию `[cA, sA]=SinCosAmps(X)`, которая из комплексных значений $X(k)$ вычисляет амплитуды косинусов и синусов, на которые раскладывается сигнал$x[n]$. Если входной сигнал имеет размерность N, то выходные массивы `cA` и `sA` должны иметь размерность $N/2 + 1$.

3.2.3.Напишите Matlab-функцию, которая выполняет синтез сигнала $x[n]$ из амплитуд косинусов и синусов, полученных функцией `SinCosAmps`. Проверьте работу функции.

3.2.4. Напишите Matlab-функцию которая преобразует комплексные значения ДПФ сигнала $X(k)$ в гармонические параметры M_k и φ_k(см. формулу (3.9)). Если $X(k)$ имеет размерность N, то размерность массивов M_kи φ_k должна быть $N/2 + 1$. Используя разработанную функцию произвольного сигнала $x[n]$, постройте амплитудный и фазовый спектры сигнала.

3.2.5. Напишите Matlab-функцию которая выполняет синтез сигнала из гармонических параметров M_k и φ_k (см. формулу(3.9)). Проверьте работу функции.

3.2.6. Используя функцию из задания 3.2.5, выполните Фурье-анализ ЭКГ-сигнала**. Постройте график сигнала во временной и частотной областях. По оси абсцисс в частотной области отложите аналоговые частоты.

3.2.7. Используйте функции из задания 3.2.5 и 3.2.6 для изменения тембра речевого сигнала. Для это запрограммируйте в Matlab следующий алгоритм:

- загрузите wav-файл при помощи функции `[x, fs]=wavread('путь_к_файлу');`
- входной сигнал разбейте на последовательные секции по 512 отсчетов;
- для каждой секции сигнала выполните ДПФ и найдите гармонические параметры M_k иφ_k;
- преобразуйте амплитуды гармоник M_k при помощи функции `ChangeTimbre(M_k,alpha)`, оставляя фазы гармоник неизменными. Параметр `alpha` влияет на степень изменения тембра:$\alpha = 1$ – не изменит тембра, $\alpha > 1$– тембр становится более высоким, $\alpha < 1$– более низким. При $\alpha > 2$ и$\alpha < 0{,}5$ в речевой сигнал может быть значительно искажен;
- выполните синтез секции речевого сигнала из параметров M_k иφ_kпри помощи функции из задания 3.2.6;
- запишите синтезированный речевой сигнал `y` в wav-файл при помощи функции `wavwrite(y,fs,'имя_файла')`.

3.2.8. Выполните моделирование работы полупроводникового диода при прохождении через него синусоидального сигнала (рис. 3.15).

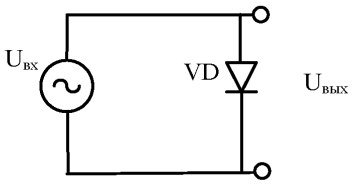


Рис.3.15. Схема для задания 3.2.8

Выходное напряжение определяется следующим образом

$$U_{\text{вых}} = \begin{cases} U_{\text{вх}} \times \operatorname{acrtg}\left(\dfrac{1}{r_{\text{пр}}}\right), & U_{\text{вх}} > U_{\text{отс}} \\ U_{\text{отс}}\left(\dfrac{U_{\text{вх}}}{U_{\text{отс}}}\right)^2, & 0 < U_{\text{вх}} < U_{\text{отс}} \\ 0, & \text{U}_{\text{вх}} < 0 \end{cases}.$$

** Файл `ecg_data`хранится в папке "\\1-311-kafevs\Docs\ТиПЦОС\Материалы для ЛР№3\". Для загрузки необходимо скопировать файл в текущий каталог Matlabи выполнить команду `load('ecg_data');`частота дискретизации сигнала равна 360 Гц.

Постройте график выходного напряжения, если на входе действует сигнал $x(t) = 0{,}5\sin(2\pi 10t),\ 0 < t < 0{,}5\ c$. Для получения дискретного сигнала $x[n]$ выполните дискретизацию сигнала с частотой 100 Гц. Постройте спектры входного ($x[n]$) и выходного сигнала $y[n]$. Какие выводы можно сделать по внешнему виду полученных спектров?

ЛАБОРАТОРНАЯ РАБОТА №4. Спектральный анализ

ЦЕЛЬ РАБОТЫ –изучение методов определения основных свойств случайных сигналов, таких, как мощность, спектральная плотность мощности (СПМ) и автокорреляционная функция (АКФ),получение практических навыков анализа случайных сигналов в среде Matlab.

4.1. Теоретические сведения

Случайные сигналы

Наряду с детерминированными (аналитическими*) сигналами существуют стохастические, или *случайные сигналы*. Отличительная черта случайного сигнала состоит в том, что его мгновенные значения заранее непредсказуемы. Характеристики таких сигналов довольно точно можно описать в вероятностном (статистическом) смысле. Важнейшими характеристиками случайного сигнала является его *автокорреляционная функция* и *спектральная плотность мощности*, которые тесно связаны между собой.

Анализ случайных сигналов выполняется в предположении, что рассматриваемый сигнал является *стационарным*. Свойство *стационарности* означает, что статистические характеристики сигнала, такие как среднее значение μи дисперсия (разброс) σ^2,не зависят от времени. Так, периодический сигнал является стационарным, а кратковременный переходной**– нестационарным.

Пример транзиентного сигнала показан на рис. 4.1,а. Чтобы убедиться в его нестационарности, вычислим *кратковременное* среднее значение. Для этого выберем значения сигнала для моментов времени с 0 по 99 и посчитаем среднее, которое отложим на графике в момент времени 0, затем окно наблюдения сместим на один отсчет влево и посчитаем среднее значение сигнала в промежутке от 1 до 100, а результат отложим на графике в момент времени 1 и т.д. Ниже приведен Matlab-код, выполняющий описанное действие:

```
x   = wavread('letter_T_female_low_8k');
N   = length(x);
Npt = 100;
y   = zeros(1, N);
for n = 1:N-Npt
    y(n) = sum(x(n:n+Npt-1))/Npt;
end
plot(1:N, y)
```

* Аналитические сигналы имеют однозначное математическое описание, например $\sin x$ или x^2.

** Часто употребляют название «транзиентный» сигнал.

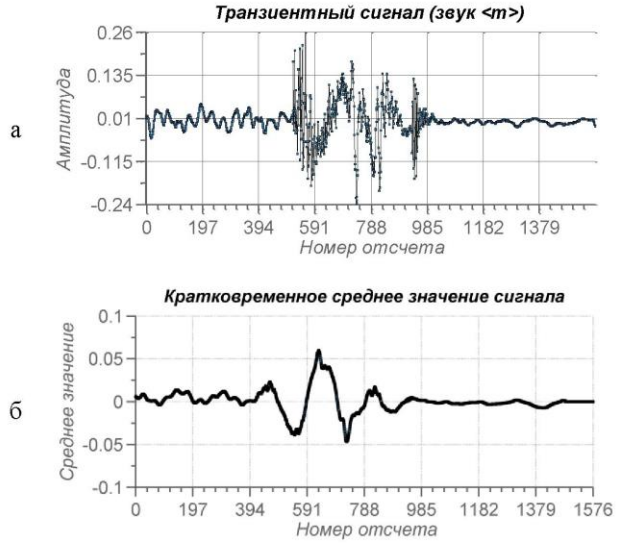


Рис. 4.1.Пример нестационарного сигнала (а) и график изменения его среднего значения во времени (б)

На рис. 4.2 показан пример стационарного сигнала и график изменения его среднего значения. Можно сделать вывод, что среднее значение сигнала практически не меняется со временем, это позволяет утверждать, что представленный сигнал отвечает условию стационарности.

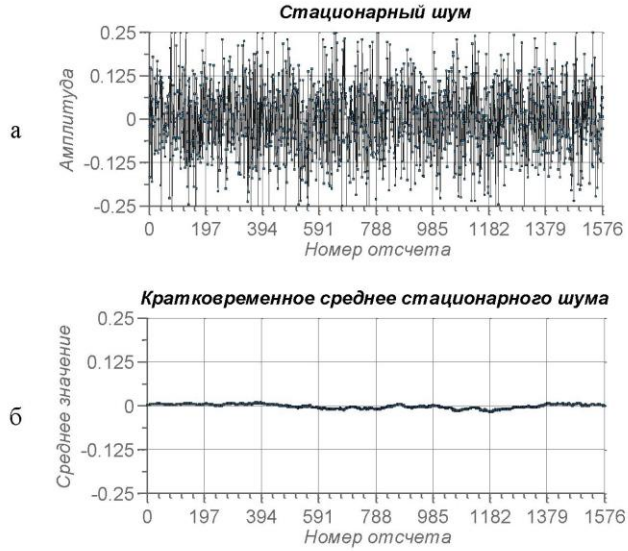


Рис. 4.2.Пример стационарного сигнала (а) и график изменения его среднего значения во времени (б)

Мощность и спектральная плотность мощности

Мгновенная мощность сигнала $x(t)$равна квадрату его амплитуды $|x(t)|^2$. Это утверждение верно, даже когда $x(t)$ принимает комплексные значения. Часто более полезным является понятие *средней мощности*. Пусть дискретный сигнал $x(n)$состоит из отсчетов непрерывного сигнала $x(t)$, тогдасредняя мощность такого сигнала вычисляется как

$$P_a = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n). \tag{4.1}$$

Также мы можем найти среднюю мощность сигнала, если известно его ДПФ:

$$P_a = \frac{1}{N^2}\sum_{k=0}^{N-1} |X(k)|^2. \tag{4.2}$$

Из выражений (4.1) и (4.2) можно сделать вывод, что

$$\sum_{n=0}^{N-1} x^2(n) = \frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2.$$

Этот результат известен как теорема *Парсеваля*, которая показывает, что энергия сигнала может быть найдена как из временных отсчетов сигнала, так и из компонент преобразования Фурье. Теорема Парсеваля позволяет ввести понятие *периодограммы*:

Периодограмма: $$P_{xx}(k) = \frac{1}{N}|X(k)|^2, \qquad k = 0,1,\dots,N-1. \tag{4.3}$$

Таким образом, периодограмма – это функция, принимающая действительные значения и обладающая свойством симметрии $P_{xx}(k) = P_{xx}(N-k)$вследствие симметричности $X(k)$*.Зная периодограмму можно записать выражение для средней мощности:

Средняя мощность: $$P_a = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n) = \frac{1}{N}\sum_{k=0}^{N-1} P_{xx}(k). \tag{4.4}$$

---

* ДПФ $X(k)$ действительного сигнала $x(n)$обладает свойствомсимметрии $X(k) = X^*(N-k)$, где «*» знак комплексного сопряжения.

Выражение(4.4) дает нам понять, что также как $x^2(n)$показывает мощность сигнала в отдельной момент времени n, так $P_{xx}(k)$ показывает мощность отдельной k-й частотной компоненты сигнала. Говорят, что периодограмма является *оценкой спектральной плотности мощности* сигнала $x(n)$.

Зачем знать СПМ сигнала?

Знание спектральной плотности мощности сигнала позволяет ответить на вопрос, какие частотные составляющие сигнала имеют наибольшую энергию. Часто записанный сигнал представляет собой смесь нескольких источников. В этом случае если каждый источник имеет свою «собственную» частоту, то по СПМ можно судить о количестве имеющихся источников: оно будет равно числу максимумов (пиков) функции СПМ. Ниже приведен Matlab-код, моделирующий описанную ситуацию. Имеется два источника сигнала: первый – частотой 500 Гц, второй – 1125 Гц. Записанный сигнал представляет собой смесь сигналов, поступающих от двух источников на фоне шума*.

```
Npt = 256;
Fs  = 8000; % частота дискретизации 8 кГц
n   = 0:Npt-1;
f1  = 500;  % частота первого источника (в Гц)
f2  = 1125; % частота первого источника (в Гц)
x       =     0.4*cos(2*pi*f1/Fs*n)   +   0.2*cos(2*pi*f2/Fs*n)   +
0.2*randn(1,Npt);
plot(n, x)
```

По временной форме сигнала (рис. 4.3) трудно судить о количестве источников. Однако это возможно сделать, если построить периодограмму сигнала (рис. 4.4) и по оси абсцисс отложить аналоговые частоты. Периодограмма показывает, что исходный сигнал состоял из двух синусоидальных компонент с разными амплитудами и частотами (в нашем случае каждая из компонент генерировалась своим источником). Также периодограмма позволяет оценить частоту синусоидальных компонент. Наличие в периодограмме компонент с малыми амплитудами объясняется присутствием шума во входном сигнале.

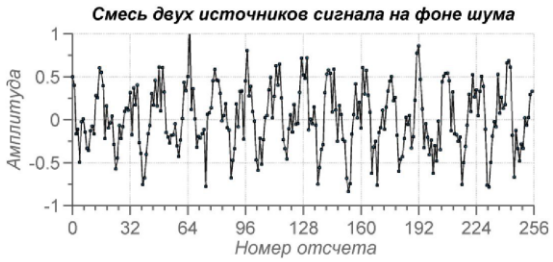


Рис. 4.3. Временная форма смеси сигналов

* Гауссовский шум с нулевым средним значением и СКО, равным единице, генерируется в Matlab-командой `randn(1,N)`.

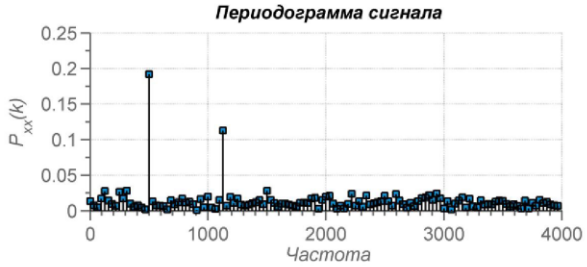


Рис. 4.4. Периодограмма смеси сигналов

Автокорреляционная функция сигнала

Объясним понятие автокорреляционной функции сигнала $x(n)$, $n = 0,1,\dots,N-1$. Предположим, что сигнал $x(n)$имеет за пределами своего определения периодическое продолжение. Это предположение весьма обоснованно, особенно для стационарных сигналов. АКФ сигнала $x(n)$определяется следующим образом:

$$\varphi_{xx}(m) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n+m), \qquad m = 0,1,\dots,N-1. \tag{4.5}$$

Можно заметить, что мощность сигнала сконцентрирована в первом отсчете АКФ:

$$P_a = \frac{1}{N}\sum_{n=0}^{N-1} x^2(n) = \varphi_{xx}(0).$$

Однако более важным является другой результат: *дискретное преобразование Фурье, взятое от автокорреляционной функции сигнала, равно периодограмме этого сигнала*:

$$\text{ДПФ}\{\varphi_{xx}(m)\} = \frac{1}{N}|X(k)|^2 = P_{xx}(k), \qquad k = 0,1,\dots,N-1.$$

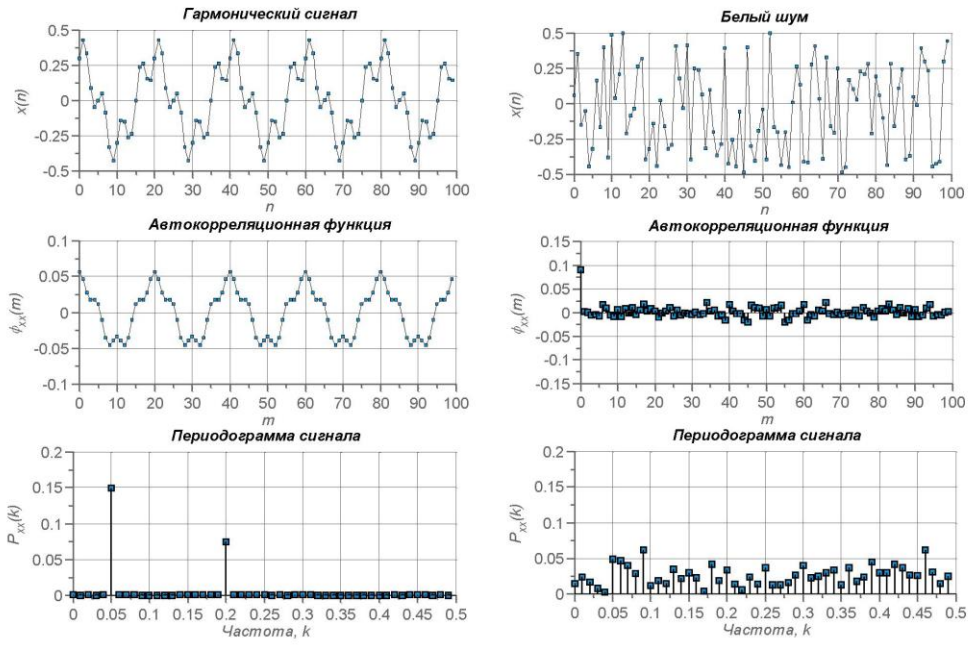
Повторим, что последнее утверждение справедливо *только* в том случае, когда сигнал $x(n)$имеет периодическое продолжение,т.е. когда $x(N) = x(0)$, $x(N+1) = x(1)$и т. д.* Автокорреляционная функция φ_{xx} показывает, насколько сильна связь между отсчетами сигнала. Например $\varphi_{xx}(1)$показывает, насколько сильно зависимы соседние отсчеты в сигнале, $\varphi_{xx}(2)$обнаруживает связь между отсчетами $x(n)$и $x(n \pm 2)$, т. е. между отсчетами, отстоящими друг

* В Matlabможно легко получить периодическое продолжение сигнала, используя операцию конкатенации. Например, если вектор-строка `sig`содержит отсчеты сигнала, то периодическое продолжение сигнала можно получить командой `periodic_sig = [sigsig];`

от друга на два интервала дискретизации. Таким образом, с помощью АКФ можно выявить скрытую в сигнале периодичность. Допустим, что сигнал содержит гармоническое колебание с периодом в 50 отсчетов. Можно уверенно сказать, что автокорреляционная функция $\varphi_{xx}(m)$такого сигнала будет иметь локальный максимум в точке $m = 50$.

Отметим, что ни $\varphi_{xx}(m)$,ни $P_{xx}(k)$не содержат никакой информации о фазе сигнала $x(n)$. По этой причине сдвиг $x(n)$во временной области не влияет на автокорреляционную функцию и периодограмму.

В качестве характерных примеров рассмотрим АКФ и периодограмму гармонического сигнала и белого шума (рис. 4.5).



а б

Рис. 4.5.Временная форма, АКФ и периодограмма гармонического сигнала $x(n) = 0{,}3\cos(2\pi \cdot 0{,}05n) + 0{,}15\sin(2\pi \cdot 0{,}2n), n = 0, 1, \ldots, 99$(а) и белого шума (б)

В обоих случаях частота дискретизации сигнала равна 1 Гц. Периодический сигнал содержит две синусоидальные компоненты: одна с периодом $T_1 = 20$ отсчетов (частота $f_1 = 1/T_1 = 0{,}05$), другая с периодом $T_2 = 5$ отсчетов (частота $f_2 = 1/T_2 = 0{,}20$). Таким образом, в сигнале имеется частота основного тона f_1и ее 4-я гармоника, поскольку $f_2 = 4f_1$. С помощью АКФ можно обнаружить период частоты основного тона, как правило, он соответствует первому максимуму$\varphi_{xx}(m)$для $m > 0$. В нашем случае мы наблюдаем максимум при $m = 20$, что в точности соответствует периоду основного колебания. Периодо-

грамма гармонического сигнала имеет два спектральных пика, показывая тем самым, что вся энергия сигнала сосредоточена в двух гармонических компонентах. *Белый шум* (рис. 4.5, б) представляет собой случайный сигнал. Анализ его периодограммы показывает, что мощность белого шума равномерно распределена во всем частотном диапазоне. Каждый последующий отсчет белого шума не зависит от предыдущего. Именно этот факт отражает АКФ$\varphi_{xx}(m)$ белого шума, в которой имеется пиктолько для $m = 0$. При любом другом сдвиге mАКФ близка к нулю, что говорит о статистической независимости отсчетов белого шума.

Оценка СПМ сигнала

Представим, что дана запись длительного стационарного сигнала и необходимо определить его СПМ. Прямой метод определения СПМ основан на вычислении квадрата модуля преобразования Фурье этого сигнала (см. формулу (4.3)). Полученная таким образом периодограмма, которую иногда называют *выборочным спектром*, представляет собой *оценку* СПМ. На практике такой метод практически не применяется, поскольку полученная СПМ является *статистически несостоятельной*. Это означает, что наличие в СПМ пика на какой-либо частоте не гарантирует нам наличие гармонической компоненты с такой частотой в сигнале. Например, на рис. 4.5 периодограмма белого шума содержит максимум на частоте 0,09 Гц, однако это не значит, что в сигнале присутствует стационарный гармонический сигнал с такой частой и амплитудой.

Для получения *устойчивой* (достоверной) оценки СПМ применяют метод усреднения периодограмм. Метод заключается в выполнении следующих действий:

1) разбить исходный сигнал $x(n)$ на K перекрывающихся или неперекрывающихся сегментов (рис. 4.6);

2) для каждого сегмента $x_i(n)$выполнить расчет периодограммы $P_{xx}^{(i)}(k)$;

3) усреднить полученные периодограммы $P_{xx}(k) = \frac{1}{K}\sum_{i=1}^{K} P_{xx}^{(i)}(k)$.

Данный метод имеет регулируемые параметры $NSAMP$ – число отсчетов на сегмент – и $NSHIFT$ – число отсчетов, на которое необходимо сдвинуть начало следующего сегмента. Ниже приводится каркас Matlab-кода, который необходим для реализации метода усреднения периодограмм.

```
Npt   = length(x);
Nsamp = 50;
Nshift= 10;
K     = floor((Npt-Nsamp)/Nshift)+1;
for i=1:K
    x_i = x(1+(i-1)*Nshift:Nsamp+(i-1)*Nshift);
% Вычисления с сегментом x_i
    % ...
end
```

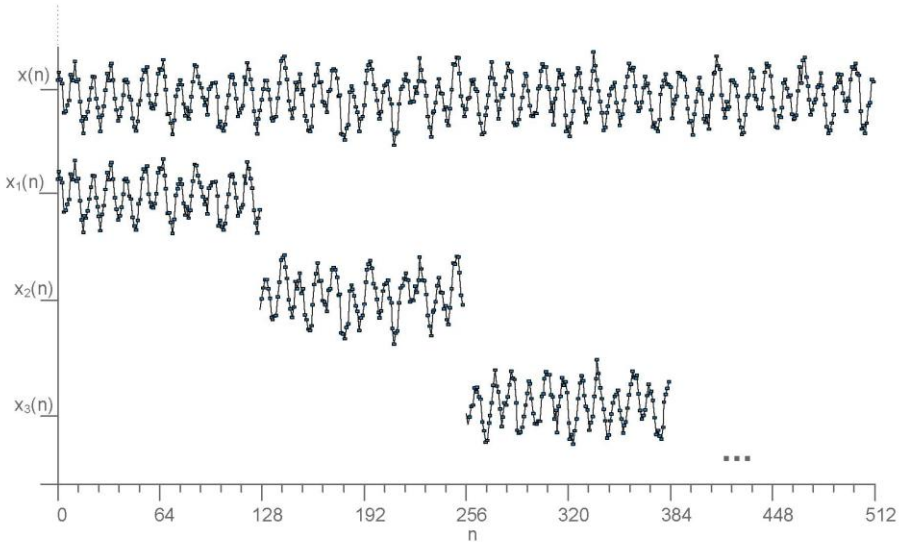


Рис. 4.6. Сегментация сигнала (без перекрытия)

Количество сегментов выбирается в зависимости от необходимой точности (надежности) спектральной оценки. При малом значении параметра $NSAMP$ получается больше сегментов, по которым производится усреднение, и, следовательно, ошибка уменьшается, но падает частотное разрешение. Увеличение параметра $NSAMP$ повышает частотное разрешение (спектр становится более детализированным), но, естественно, за счет увеличения ошибки, возникающей из-за уменьшения числа усредняемых сегментов.

Кратковременный анализ речевых сигналов

На рис.4.7показанавременная форма речевого сигнала. Из графика видно, что свойства речевого сигнала, например, характер возбуждения на вокализованных и невокализованных участках, пиковая амплитуда и др., изменяются во времени. Вокализованными называются те сегменты речи, где задействованы голосовые связки диктора. Такие звуки как 'а','о','ж',являются вокализованными, в то время как звуки 'с','ш','щ' являются невокализованными.

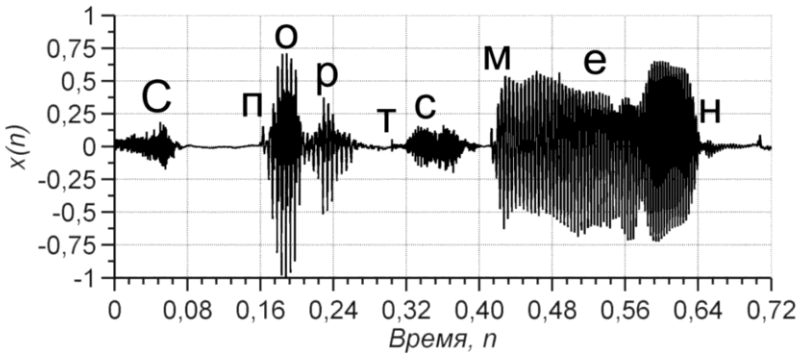


Рис. 4.7. Речевой сигнал (слово «спортсмен»)

В основе большинства методов обработки речи лежит предположение о том, что свойства речевого сигнала с течением времени медленно изменяются. Это предположение приводит к методам *кратковременного анализа*, в которых сегменты речевого сигнала выделяются и обрабатываются так, как если бы они были короткими участками отдельных звуков с отличающимися свойствами. *Сегменты*, которые иногда называют *кадрами анализа*, или *фреймами*, обычно пересекаются. Результатом обработки на каждом интервале является число или совокупность чисел. Следовательно, подобная обработка приводит к новой, зависящей от времени последовательности, которая может служить характеристикой речевого сигнала.

В общем виде большинство методов кратковременного анализа могут быть описаны выражением

$$Q(n) = \sum_{m=-\infty}^{\infty} T[x(m)]w(n-m). \tag{4.6}$$

Смысл этого выражения в следующем: сигнал $x(m)$подвергается преобразованию $T[\cdot]$, затемрезультирующая последовательность умножается на последовательность значений временного окна (весовая функция) с центром на отсчете n. Результаты суммируются по всем ненулевым значениям.

Примером такого анализа может служить измерение кратковременной мощности сигнала, которая определяется как

$$P(n) = \frac{1}{N+1} \sum_{m=n-N/2}^{n+N/2} x^2(m). \tag{4.7}$$

Таким образом, кратковременная мощность в момент времени nесть просто средняя сумма квадратов отсчетов в диапазоне от $n - N/2$до $n + N/2$. Из (4.6) видно, что в (4.7)$T[\cdot]$представляет собой возведение в квадрати деление на $(N + 1)$, а

$$w(n) = \begin{cases} 1, & -N/2 \le n \le N/2, \\ 0 & \text{в противном случае.} \end{cases} \quad (4.8)$$

Вычисление кратковременной мощности иллюстрирует рис. 4.8. Отметим, что окно «скользит» вдоль последовательности квадратов значений сигнала, ограничивая длительность интервала, используемого в вычислениях.

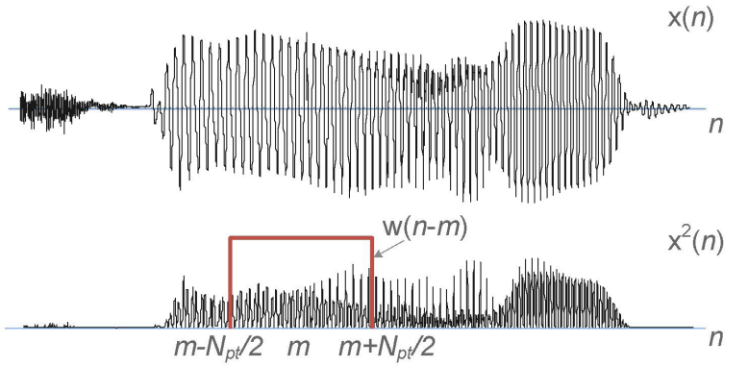


Рис. 4.8. Иллюстрация вычисления функции кратковременной мощности

Основное значение функции $P(n)$в применении к обработке речи состоит в том, что эта величина позволяет отличить вокализованные речевые сегменты от невокализованных.

Часто в методах кратковременного анализа вместо прямоугольного окна (см. формулу (4.8) и рис. 4.8) используют окна, имеющие колоколообразный вид. Эти окна имеют значения, близкие к единице в центре сегмента и близкие к нулю по краям. Применением этих окон стараются ослабить краевые эффекты, которые возникают, когда на границах сегмента сигнал имеет большую амплитуду.

4.2. Порядок выполнения работы

4.2.1. Разработайте в Matlabфункцию [`P, P_dB]=calc_power(x)`для вычисления мощности сигнала по формуле (4.1). Функция также должна выдавать значение мощности в децибелах $P_{dB} = 10\log_{10} P_a$. Посчитайте мощность следующих сигналов при$n = 0, 1 \ldots, 127$:

$$x_1(n) = 0{,}5\sin\left(2\pi\frac{3}{128}n\right),$$
$$x_2(n) = 0{,}5\sin\left(2\pi\frac{30}{128}n\right),$$
$$x_3(n) = 0{,}3\cos\left(2\pi\frac{6}{128}n\right) + 0{,}4\cos\left(2\pi\frac{10}{128}n\right).$$

4.2.2. Разработайте в Matlabфункцию длявычисление периодограммы сигнала (см. формулу(4.3)). Постройте периодограммы сигналов из зада-

ния 4.2.1. Вычислите мощность сигналов из полученных периодограмм по формуле(4.2).

4.2.3. Предположим, что зарегистрированный сигнал представляет собой сумму полезного сигнала и белого шума($n = 0, 1, \ldots, 511$):

$$x(n) = 0{,}4\cos\left(2\pi\frac{9}{128}n\right) + A \cdot r(n).$$

Определите максимальное отношение сигнал/шум (ОСШ), при котором по периодограмме $x(n)$можно обнаружить в нем наличие полезного сигнала. Отношение сигнал/шум (англ. *signal-to-noise ratio*, *SNR*) – безразмерная величина, равная отношению мощности полезного сигнала к мощности шума:

$$\mathrm{SNR(dB)} = 10\log_{10}\frac{P_{signal}}{P_{noise}}.$$

Для определения требуемого ОСШ необходимо выполнить несколько экспериментов, плавно изменяя параметр A и наблюдая изменение периодограммы сигнала. Будем считать, что сигнал можно обнаружить, если уровень полезного сигнала на 10 дБ больше уровня компонент спектра, образуемых шумовым сигналом. Белый шум $r(n)$можно получить в Matlabпри помощи функции

```
rand(1, 512)
```

4.2.4. Оцените СПМ сигнала на выходе фильтра при помощи метода усреднения периодограмм. Параметры $NSAMP$ и $NSHIFT$ подберите самостоятельно. На вход фильтра поступает 4096 отсчетов белого шума. Коэффициенты фильтра:

```
b = [0.00482434 0 -0.0192973 0 0.02894606 0 -0.0192973 0
0.00482434];
a = [1 -2.06958023 3.99771255 -4.3894077 4.45285533 -2.9060422 1.75168470
- 0.5862147 0.18737949];
```

4.2.5. Разработайте в Matlabфункцию вычисления АКФ сигнала. Примените функцию для определения периода основного тона речевого сигнала (wav-файл для анализа хранится в папке "\\1-311-kafevs\Docs\ТиПЦОС\Материалы для ЛР№4\007.wav"). Wav-файл можно загрузить командой`[x, fs] =wavread('путь_к_файлу')`. Входной сигнал разбивается на перекрывающиеся секции по 320 отсчетов. Перекрытие секций составляет 160 отсчетов. Для каждой секции сигнала вычисляется АКФ и находится номер отсчета, содержащий первый максимум функции. Номер этого отсчета и является *оценкой* периода основного тона. Полученные значения отобразите на графике:

по оси абсцисс – номер сегмента, по оси ординат – значение периода основного тона.

4.2.6. Разработайте в Matlabфункцию для отделения вокализованных участков речи от невокализованных:

```
[x_v, x_u] = vu_separate(x, N, threshold)
```

Сепарация (разделение) происходит на основе анализа кратковременной мощности сигнала(см.(4.7)). На вход функции поступает речевой сигнал `x`, длина окна анализа на котором считается мощность `N` и значение порога `threshold`. Если мощность сигнала в момент времени n больше порога, то считается, что отсчет $x(n)$ является вокализованным, в противном случае – невокализованным. Сепарация выполняется следующим образом:

а) для входного сигнала $x(n)$ рассчитывается кратковременная мощность $P(n)$ по выражению (4.7);

б) рассчитывается признак вокализованности:

$$v(n) = \begin{cases} 1, & P(n) > \text{thershold}, \\ 0, & P(n) \le \text{thershold}; \end{cases}$$

в) вычисляется сепарация сигнала:

$$x_v(n) = x(n) \cdot v(n),$$
$$x_u(n) = x(n) \cdot \big(1 - v(n)\big).$$

Удобно мощность сигнала посчитать в децибелах:

$$P_{dB}(n) = 10 \log_{10} P(n)$$

и порог для сепарации также подбирать в децибелах.

ЛАБОРАТОРНАЯ РАБОТА №5.Расчет цифровых фильтров с бесконечными импульсными характеристиками

ЦЕЛЬ РАБОТЫ –расчет БИХ-фильтров с различными аппроксимациями идеальной АЧХ в пакете Matlab и изучение их свойств.

5.1. Теоретические сведения

Основные свойства БИХ-фильтров

Цифровой фильтр с бесконечной импульсной характеристикой (БИХ-фильтр) является физически реализуемым и устойчивым, если его импульсная характеристика $h(n)$ удовлетворяет следующим условиям:

$$h(n) = 0, \quad n < 0,$$

$$\sum_{n=0}^{\infty} |h(n)| < \infty .$$

Наиболее общая форма записи z-преобразования импульсной характеристики БИХ-фильтров имеет вид

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}} .$$

Здесь по крайней мере один из коэффициентов a_iотличен от нуля, причем сразу все корни знаменателя (полюса передаточной функции) не могут в точности компенсироваться корнями числителя (нулями передаточной функции). Нули $H(z)$ могут располагаться на всей z-плоскости, но полюсы в соответствии с условием устойчивости фильтра обязательно должны размещаться внутри круга единичного радиуса. Как правило, число нулей M не превышает числа полюсов N. Системы, удовлетворяющие этому условию, называются системами N-го порядка. При $M>N$ порядок системы становится неопределенным. В этом случае можно считать, что передаточная функция $H(z)$ соответствует последовательному соединению системы N-го порядка и фильтра с конечной импульсной характеристикой $(M - N)$-го порядка.

Основными функциями, характеризующими фильтр, являются квадрат амплитудной характеристики, фазовая характеристика и характеристика групповой задержки.

При расчете БИХ-фильтра с использованием аппроксимации только амплитудной характеристики (т.е. без учета фазовой характеристики) удобнее всего оперировать квадратом амплитудной характеристики, определяемым следующим образом:

$$\left| H\left(e^{j\omega}\right) \right|^2 = \left| H(z) H\left(z^{-1}\right) \right|_{z=e^{j\omega}} .$$

Расположению нулей и полюсов этой функции в z-плоскости свойственна симметрия с зеркальным отображением относительно единичной окружности. Полюсы $H(z)$ располагаются внутри единичной окружности, поэтому они полностью определяются квадратом амплитудной характеристики фильтра. Нули передаточной функции $H(z)$ чаще всего выбираются таким способом, чтобы соответствующие им нули квадрата амплитудной характеристики располагались на единичной окружности или внутри ее в z-плоскости. Фильтры с такими нулями являются минимально-фазовыми фильтрами.

Так как передаточная функция БИХ-фильтра в общем случае представляет собой комплексную функцию от ω, можно рассматривать и фазовую характеристику фильтра, которая равна

$$\beta\left(e^{j\omega}\right) = \operatorname{arctg}\left\{\frac{Im\left[H(z)\right]}{Re\left[H(z)\right]}\right\}_{z=e^{j\omega}}.$$

Фазовая характеристика БИХ-фильтра, как правило, существенно нелинейна, поэтому для оценки дисперсионного воздействия фильтра на типовой обрабатываемый сигнал часто используется характеристика групповой задержки, записываемая как

$$\tau_g\left(e^{j\omega}\right) = -\frac{d\beta\left(e^{j\omega}\right)}{d\omega} = -jz\frac{d\beta}{dz}\Big|_{z=e^{j\omega}}.$$

Предпочтительна приблизительно постоянная характеристика групповой задержки во всей полосе (или полосах) пропускания фильтра.

Методы расчета коэффициентов БИХ-фильтра

Расчет фильтра сводится к нахождению значений его коэффициентов b_i и a_i, обеспечивающих аппроксимацию заданных характеристик (таких как импульсная, частотная, характеристика групповой задержки и др.) в том или ином смысле (например, в среднеквадратичном или минимаксном). Таким образом, задача расчета фильтра в значительной мере сводится к задаче аппроксимации и может быть решена чисто математическими методами.

Расчет цифровых БИХ-фильтров практически не связан с фильтрами непрерывного времени. Однако вместо того, чтобы заново создавать теорию расчета цифровых фильтров, можно воспользоваться простыми методами отображения, позволяющими преобразовывать аналоговые фильтры в цифровые. Именно эти методы чаще всего применяются при расчете стандартных БИХ-фильтров: нижних и верхних частот, полосовых, режекторных. В таких случаях последовательность расчета должна быть следующей:

– расчет аналогового фильтра-прототипа;

– преобразование полосы частот;

– дискретизация аналогового фильтра.

Другую группу методов расчета цифровых БИХ-фильтров образуют прямые методы расчета в z-плоскости. Часто удается найти такое расположение

полюсов и нулей фильтра, при котором обеспечивается некоторая аппроксимация непосредственно заданной характеристики фильтра.

Третий, также часто встречающийся подход к расчету БИХ-фильтров, заключается в использовании процедур оптимизации для нахождения положения нулей и полюсов в *z*-плоскости. В этом случае нельзя получить формулы, связывающие коэффициенты фильтра с параметрами заданной характеристики. Расчет производится методом последовательных приближений.

Расчет аналоговых фильтров-прототипов

В качестве аналогового фильтра-прототипа используется нормированный фильтр нижних частот с частотой среза, равной 1 рад/с. Передаточная функция такого фильтра представляет собой рациональную функцию следующего вида:

$$H(s) = \frac{\sum_{i=0}^{m} b_i s^i}{1 + \sum_{i=1}^{n} a_i s^i}.$$

Аппроксимируемой функцией является квадрат амплитудной характеристики. Чаще всего применяются фильтры Баттерворта, Чебышева и эллиптические.

Фильтры Баттерворта имеют максимально гладкую амплитудную характеристику в начале координат в *s*-плоскости. Ее квадрат равен

$$\left|H(\Omega)\right|^2 = \frac{1}{1 + \left(\Omega^2\right)^n},$$

где n – порядок фильтра.

Данные фильтры имеют только полюсы (все нули расположены на бесконечности). На частоте Ω=1 рад/с коэффициент передачи равен $1/\sqrt{2}$ (т.е. на частоте среза амплитудная характеристика спадает на 3 дБ). Порядок фильтра полностью определяет весь фильтр.

Отличительной чертой фильтров Чебышева является наименьшая величина максимальной ошибки аппроксимации в заданной полосе частот. Ошибка аппроксимации представляется равновеликими пульсациями, флуктуирует между максимумами и минимумами равной величины. В зависимости от того, где минимизируется ошибка – в полосе пропускания или подавления, различают фильтры Чебышева типа I и II.

Фильтры Чебышева типа I имеют только полюсы и обеспечивают равновеликие пульсации амплитудной характеристики в полосе пропускания и монотонное изменение ослабления в полосе подавления. Квадрат амплитудной характеристики фильтра n-го порядка описывается выражением

$$\left|H(\Omega)\right|^2 = \frac{1}{1 + \varepsilon^2 T_n^2(\Omega)},$$

где ε – параметр, характеризующий пульсации в полосе пропускания; $T_n(\Omega)$ – полином Чебышева n-го порядка, равный

$$T_n(\Omega)=\begin{cases}\cos(n\arccos\Omega), & |\Omega|\le 1,\\ \operatorname{ch}(n\operatorname{Arch}\Omega), & |\Omega|>1.\end{cases}$$

Свойство оптимальности фильтров Чебышева типа I заключается в том, что не существует какого-либо другого фильтра n-го порядка, содержащего только полюсы, которые имели бы такие же или лучшие характеристики и в полосе пропускания, и в полосе подавления.

Фильтры Чебышева типа II (иногда их называют обратными фильтрами Чебышева) обеспечивают монотонное изменение ослабления в полосе пропускания (максимально гладкое при Ω=0) и равновеликие пульсации в полосе подавления. Нули фильтров располагаются на мнимой оси в s-плоскости, а полюсы – в левой полуплоскости. Квадрат амплитудной характеристики фильтра n-го порядка можно представить следующим образом:

$$|H(\Omega)|^2=\frac{1}{1+\varepsilon^2\left[T_n(\Omega_r)/T_n(\Omega_r/\Omega)\right]^2},$$

где Ω_r – наименьшая частота, на которой в полосе подавления достигается заданный уровень ослабления.

Эллиптические фильтры (называемые также фильтрами Кауэра или Золотарева) характеризуются тем, что их амплитудная характеристика имеет равновеликие пульсации и в полосе пропускания, и в полосе подавления. Данные фильтры оптимальны с точки зрения минимальной ширины переходной полосы. Квадрат амплитудной характеристики записывается в виде

$$|H(\Omega)|^2=\frac{1}{1+\varepsilon^2 R_n^2(\Omega,L)},$$

где $R_n(\Omega,L)$ – рациональная функция Чебышева; L – параметр, характеризующий ее пульсации.

Преобразование полосы частот для аналоговых фильтров

Для преобразования фильтра нижних частот (ФНЧ) с частотой среза 1 рад/с в другой фильтр нижних частот (имеющий другую частоту среза), а также в фильтр верхних частот (ФВЧ), полосовой (ПФ) или режекторный (РФ) применяют следующие преобразования:

$$\text{ФНЧ} \rightarrow \text{ФНЧ}: \quad s\rightarrow\frac{s}{\Omega_u},$$

$$\text{ФНЧ} \rightarrow \text{ФВЧ}: \quad s\rightarrow\frac{\Omega_u}{s},$$

$$\text{ФНЧ} \rightarrow \text{ПФ}: \quad s\rightarrow\frac{s^2+\Omega_u\Omega_l}{s(\Omega_u-\Omega_l)},$$

$$\text{ФНЧ} \rightarrow \text{РФ:} \qquad s \rightarrow \frac{s\left(\Omega_u - \Omega_l\right)}{s^2 + \Omega_u \Omega_l}.$$

Здесь Ω_l – нижняя частота среза; Ω_u – верхняя частота среза.

Данные соотношения имеют нелинейный характер, однако это не создает никаких трудностей, поскольку частотные характеристики преобразуемых фильтров аппроксимируются ступенчатой функцией. Так, нелинейность отображения приводит к изменению взаимного расположения максимумов и минимумов пульсаций характеристик эллиптических фильтров, но не влияет на амплитуду этих пульсаций. Поэтому фильтры, рассчитанные методами преобразования полосы, сохраняют равновеликий характер пульсаций фильтра-прототипа.

Дискретизация аналогового фильтра

Наиболее распространенным методом дискретизации аналоговых фильтров является метод билинейного z-преобразования. Он основан на простом конформном отображении s-плоскости в z-плоскости, получающемся в результате следующей замены:

$$s \rightarrow \frac{2}{T}\frac{\left(1 - z^{-1}\right)}{\left(1 + z^{-1}\right)}.$$

При таком преобразовании ось $j\Omega$ из s-плоскости отображается в единичную окружность на z-плоскости; левая полуплоскость s отображается в единичный круг, а правая полуплоскость s – в область вне единичного круга на z-плоскости.

Определенным недостатком билинейного z-преобразования является нелинейность соотношения между частотами аналогового фильтра Ω и цифрового фильтра ω:

$$\Omega \rightarrow \frac{2}{T} tg\left(\frac{\omega T}{2}\right).$$

Данный недостаток не позволяет, например, использовать билинейное z-преобразование для получения цифрового дифференцирующего фильтра. Существует, правда, большой класс фильтров, для которых частотная деформация может быть скомпенсирована. К ним относятся фильтры нижних и верхних частот, полосовые и режекторные. Метод компенсации прост. Пусть известна совокупность частот среза цифрового фильтра. Используя приведенное соотношение между частотными шкалами, пересчитаем все частоты среза цифрового фильтра в частоты среза аналогового. Теперь рассчитаем аналоговый фильтр, все характерные частоты которого совпадали бы с этими пересчитанными частотами. Выполнив билинейное z-преобразование, получим требуемый цифровой фильтр.

Расчет цифровых фильтров в пакете Matlab

Расчет БИХ-фильтров с аппроксимацией Баттерворта в пакете Matlab осуществляется с помощью функции `butter`. Ее формат следующий:

```
[B,A] = butter(N,Wn)
```

Рассчитывается фильтр нижних частот `N`-го порядка с частотой среза `Wn`, заданной в долях частоты Найквиста (`0 < Wn < 1`). В векторах `B` и `A` длиной `N+1` возвращаются коэффициенты числителя и знаменателя передаточной функции, расположенные в порядке уменьшения степеней z. Команда

```
[B,A] = butter(N,Wn,'high')
```

позволяет получить фильтр нижних частот. Если `Wn` – двухэлементный вектор `Wn = [W1 W2]`, то будет рассчитан полосовой фильтр порядка `2N` с полосой пропускания `W1 < W < W2`. Для получения режекторного фильтра следует использовать команду

```
[B,A] = BUTTER(N,Wn,'stop')
```

При расчете фильтров в качестве исходных данных обычно задаются граничные частоты полос пропускания `Wp` и подавления `Ws`, а также наибольшее допустимое отклонение в полосе пропускания `Rp` и наименьшее допустимое затухание в полосе подавления `Rs`. Получить необходимые для функции `butter` порядок фильтра и частоту среза можно с помощью функции `butterord`:

```
[N, Wn] = butterord(Wp, Ws, Rp, Rs)
```

Например, `Wp = 0.2`, `Ws = 0.1` соответствует фильтру верхних частот, а `Wp = [0.1 0.8]`, `Ws = [0.2 0.7]` – полосовому.

Для расчета фильтров Чебышева типов I и II используются функции `cheby1` и `cheby2`:

```
[B,A] = cheby1(N,R,Wn)
[B,A] = cheby2(N,R,Wn)
```

Для `cheby1` параметр `R` является размахом колебаний в полосе пропускания (в децибелах), а для `cheby2` – в полосе подавления. В остальном данные функции аналогичны `butter`.

Определить порядок фильтров Чебышева типов I и II и их частоту среза можно с помощью функций `cheby1ord` и `cheby2ord`, которые аналогичныфункции `butter`.

Эллиптические фильтры рассчитываются функцией`ellip`:

```
[B,A] = ellip(N,Rp,Rs,Wn)
```

Параметры `Rp` и `Rs` обозначают размах колебаний в полосах пропускания и подавления соответственно, выраженные в децибелах. Остальные параметры соответствуют функции `butter`. Для нахождения `N` и `Wn` используется функция `ellipord`.

При анализе цифровых БИХ-фильтров часто бывает необходимо построить диаграмму расположения нулей и полюсов передаточной функции на z-плоскости. Для этого служит функция Matlab `zplane`. В качестве параметров ей передаются вектора `B` и `A` – коэффициенты числителя и знаменателя передаточной функции. Функция `zplane`строит на графике единичную окружность и отмечает положение нулей с помощью символа `'o'` и полюсов с помощью символа `'x'`. При кратности нулей и полюсов, большей единицы, она показывается справа вверху.

5.2. Порядок выполнения работы

5.2.1. В соответствии с номером варианта рассчитайте ФНЧ с аппроксимацией Баттерворта (табл. 5.1).

Таблица5.1

Варианты для задания 5.2.1

Номер варианта	1	2	3	4	5	6
Порядок фильтра	7	6	5	8	9	5
Частота дискретизации, Гц	200	2000	16000	8000	10000	20
Частота среза, Гц	60	400	5000	1900	2000	5

Постройте АЧХ и ФЧХ, диаграмму расположения нулей и полюсов передаточной функции, значимую часть импульсной характеристики. Каким образом можно получить каждые два графика из третьего?

5.2.2. Выполните задание 5.2.1, воспользовавшись функцией `cheby1` и приняв допустимую неравномерность АЧХ в полосе пропускания 0,5 дБ.

5.2.3. Выполните задание 5.2.2, воспользовавшись функцией `ellip` и приняв минимальное затухание АЧХ в полосе подавления 30 дБ.

5.2.4. Требуется цифровой ФВЧ со следующими параметрами (табл. 5.2):

Таблица5.2

Варианты для задания 5.2.4(частоты определены в долях от частоты Найквиста)

Номер варианта	1	2	3	4	5	6
Граничная частота подавления	0,64	0,28	0,4	0,1	0,86	0,72
Граничная частота пропускания	0,7	0,32	0,6	0,15	0,9	0,73
Допустимая неравномерность в полосе пропускания, дБ	0,05	0,1	$1\cdot10^{-5}$	0,15	0,11	1
Минимальное затухание в полосе подавления, дБ	40	30	100	85	57	30

Какой порядок будет иметь такой фильтр с аппроксимациями Баттерворта, Чебышева типа I, Чебышева типаII, эллиптической? Сравните эффективность различных аппроксимаций при более жестких и более мягких требованиях к АЧХ.

5.2.5. Исследуйте, как скажется на АЧХ и ФЧХ фильтров из заданий 5.2.1–5.2.3 усечение коэффициентов передаточной функции до четырех десятичных разрядов,до двух разрядов? Сделайте выводы.

ЛАБОРАТОРНАЯ РАБОТА №6. Расчет цифровых КИХ-фильтров с линейной фазовой характеристикой методом взвешивания

ЦЕЛЬ РАБОТЫ –расчет КИХ-фильтров с линейной фазовой характеристикой в пакете Matlab и изучение их свойств,изучение различных типов временных окон.

6.1. Теоретические сведения

Представление о КИХ-фильтре

Представление о цифровом фильтре с конечной импульсной характеристикой можно получить, если рассмотреть задачу вычисления скользящего значения в массиве данных. Предположим, есть массив данных $x(n), n = 0, 1, \dots$. Необходимо сформировать массив выходных (сглаженных) данных

$$y(n) = \frac{1}{5}\sum_{m=-2}^{2} x(n-m). \tag{6.1}$$

В данном случае сглаживание состоит в том, что для получения выходного значения в каждый момент времени nберется среднее значение от пяти точек входных данных $x(n-2)$, $x(n-1)$, $x(n)$, $x(n+1)$и $x(n+2)$. Заметим, что с помощью этой формулы невозможно сгладить по два значения в начале и в конце потока данных. На рис.Рис. 6.16 поясняется работа формулы (6.1).

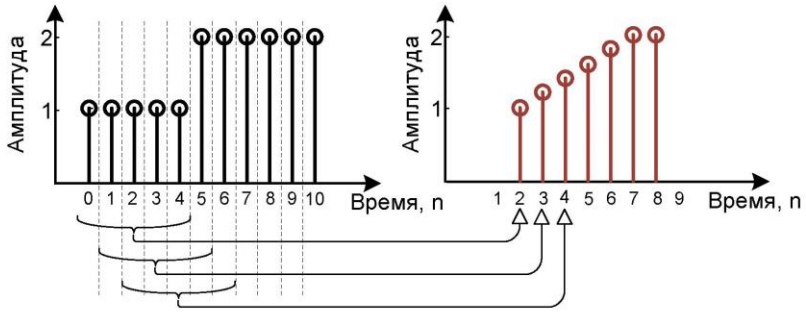


Рис. 6.1.Процесс вычисления скользящего среднего значения массива данных

Формула (6.1) представляет собой цифровой КИХ-фильтр ($N = 4$)

$$y(n) = \sum_{m=0}^{N} x(n-m)h(m), \tag{6.2}$$

у которого все коэффициенты $h(m)$имеют значения $1/5$.

Как выглядит формула (6.1) с частотной точки зрения? Предположим, что входная функция представляет собой комплексную синусоиду $e^{j\omega n}$. Поскольку формула линейна по отношению к данным, то такая же функция будет всегда на выходе, за исключением того, что она умножается на множитель $H(\omega)$, который называется частотной характеристикой фильтра (см. лабораторную работу №2). Подставив комплексную экспоненту $e^{j\omega n}$в формулу сглаживания пятерками, получим

$$H(\omega) = \frac{1}{5}(e^{-2j\omega} + e^{-j\omega} + 1 + e^{j\omega} + e^{2j\omega}).$$

Отметим, что отрицательные и положительные частоты имеют одинаковые коэффициенты и, следовательно, попарно могут быть заменены соответствующими косинусными функциями:

$$H(\omega) = \frac{1}{5}(1 + 2\cos\omega + 2\cos 2\omega).$$

Вид этой функции показан на рис. 6.2.Частотная характеристика $H(\omega)$ показывает, что происходит, когда усредняются данные с частотой ω. Очевидно, что сглаживание можно выполнять и по 7, и по 9, и т.д. точкам.

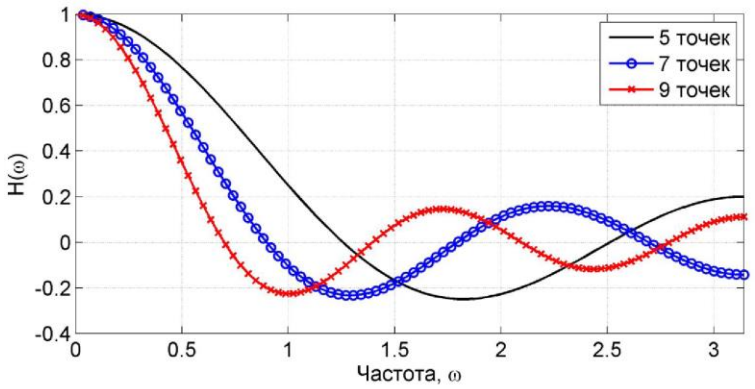


Рис. 6.2. КИХ-фильтр для сглаживания данных в частотной области

На рис. 6.3 и 6.4 показано вычисление скользящего среднего для данных, содержащих колебания различных частот. Из приведенных иллюстраций можно сделать вывод, что увеличение частоты колебаний во входных данных привело к уменьшению амплитуды колебаний в выходном массиве данных, что находится в полном соответствии с графиком частотной характеристики.

Таким образом, мы рассмотрели частный случай КИХ-фильтра(6.2), у которого все коэффициенты являются постоянными. В общем случае проектирование КИХ-фильтра состоит в расчете таких коэффициентов $h(m)$, которые обеспечивали бы требуемый вид частотной характеристики.

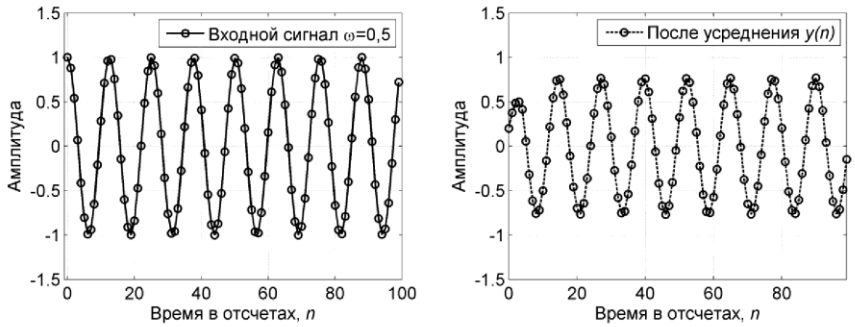


Рис. 6.3. Усреднение по 5точкам данных, содержащих колебание частотой $\omega = 0{,}5$рад/с

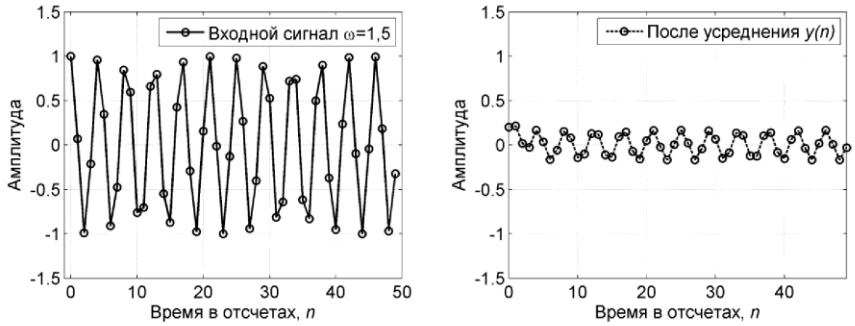


Рис. 6.4.Усреднение по 5 точкам данных, содержащих колебание частотой $\omega = 1{,}5$рад/с

Основные свойства КИХ-фильтров

Пусть импульсная характеристика КИХ-фильтра $h(n)$ задана на интервале $0 \le n \le N-1$. Ее преобразование Фурье равно

$$H\left(e^{j\omega}\right) = \sum_{n=0}^{N-1} h(n) e^{-j\omega n}.$$

Рассматривая только действительные $h(n)$, получим дополнительные ограничения на функцию $H(e^{j\omega})$, представив ее через амплитуду и фазу:

$$H\left(e^{j\omega}\right) = \pm\left| H\left(e^{j\omega}\right)\right| e^{j\theta(\omega)}.$$

Модуль преобразования Фурье является симметричной функцией, а фаза – антисимметричной функцией частоты, т. е.

$$\left| H\left(e^{j\omega}\right)\right| = \left| H\left(e^{-j\omega}\right)\right|, \quad 0 \le \omega \le \pi, \theta(\omega) = -\theta(-\omega).$$

На практике при расчете КИХ-фильтров часто требуется строго линейная фазовая характеристика. Линейность фазы является еще одним ограничением; она означает, что фазовая характеристика $\theta(\omega)$ имеет вид

$$\theta(\omega) = -\alpha\omega, \quad -\pi \le \omega \le \pi,$$

где α – постоянная фазовая задержка, выраженная через число интервалов дискретизации.

Выражение для преобразования Фурье от импульсной характеристики теперь можно переписать следующим образом:

$$H\left(e^{j\omega}\right)=\sum_{n=0}^{N-1}h(n)e^{-j\omega n}=\pm\left|H\left(e^{j\omega}\right)\right|e^{-j\alpha\omega}=H^{*}\left(e^{j\omega}\right)e^{-j\alpha\omega},$$

где $H^{*}(e^{j\omega})$ – действительная функция.

Для каждого N существует только одна фазовая задержка α, при которой достигается строгая линейность фазовой характеристики фильтра:

$$\alpha=\frac{N-1}{2}.$$

При этом импульсная характеристика должна обладать вполне определенной симметрией:

$$h(n)=h(N-1-n),\quad 0\le n\le N-1.$$

Если N– нечетное, то α – целое, т.е. задержка в фильтре равна целому числу интервалов дискретизации. Центр симметрии импульсной характеристики приходится на центральный отсчет, для которого нет пары. Функция $H^{*}(e^{j\omega})$ в этом случае симметрична относительно π.

При четном N задержка становится дробной. Центр симметрии импульсной характеристики располагается между двумя центральными отсчетами, и каждый отсчет имеет парный. Функция $H^{*}(e^{j\omega})$ асимметрична относительно π и обращается в 0 при $\omega=\pi$. Отсюда следует, что такие фильтры нельзя использовать, например, для проектирования фильтров верхних частот.

Согласно условию линейности фазовой характеристики, требуется, чтобы фильтр имел постоянную как групповую, так и фазовую задержку. Если достаточно, чтобы только групповая задержка была постоянной (как это часто бывает), можно определить еще один тип фильтра с линейной фазой, фазовая характеристика которого является кусочно-линейной функцией частоты:

$$H\left(e^{j\omega}\right)=\pm\left|H\left(e^{j\omega}\right)\right|e^{j(\beta-\alpha\omega)}.$$

Линейность фазы в этом случае достигается при следующих условиях:

$$\alpha=\frac{N-1}{2},$$

$$\beta=\pm\frac{\pi}{2},$$

$$h(n)=-h(N-1-n),\quad 0\le n\le N-1.$$

Импульсные характеристики фильтров данного типа, в отличие от предыдущих, антисимметричны относительно центра.

Для нечетных значений N средний отсчет импульсной характеристики всегда равен нулю. Функция $H^{*}(e^{j\omega})$ антисимметрична относительно π и обращается в 0 при $\omega=0$ и $\omega=\pi$. Без учета множителя с линейным изменением фазы частотная характеристика является чисто мнимой, поэтому такие фильтры

наиболее пригодны для проектирования преобразователей Гильберта и дифференциаторов.

В случае, если N четно, $H^*(e^{j\omega})$ симметрична относительно π и обращается в 0 при ω=0.

Расчет КИХ-фильтров с линейной фазой методом взвешивания

Суть метода взвешивания (метод окна)сводится к получению конечной импульсной характеристики путем усечения последовательности бесконечной импульсной характеристики длины. Поскольку частотная характеристика требуемого идеального цифрового фильтра $H_d(e^{j\omega})$является периодической функцией частоты, ее можно представить рядом Фурье:

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d(n) e^{-j\omega n}, \qquad (6.3)$$

где $h_d(n)$ – соответствующая последовательность отсчетов импульсной характеристики, т.е.

$$h_d(n) = (1/2\pi) \int_{-\pi}^{\pi} H_d(e^{j\omega}) \cdot e^{j\omega n} d\omega. \qquad (6.4)$$

Использование соотношений(6.3) и (6.4)для проектирования КИХ-фильтров связано с двумя трудностями. Во-первых, импульсная характеристика фильтра в общем случае имеет бесконечную длину, поскольку суммирование в (6.3) производится в бесконечных пределах. Во-вторых, такой фильтр физически нереализуем, т. к. импульсная характеристика начинается в – ∞,т.е. никакая конечная задержка не сделает фильтр физически реализуемым. Итак, фильтр, рассчитываемый на основе представления функции $H_d(e^{j\omega})$рядом Фурье, оказывается физически нереализуемым БИХ-фильтром.

Один из возможных методов получения КИХ-фильтра, аппроксимирующего заданную функцию $H_d(e^{j\omega})$, заключается в усечении бесконечного ряда Фурье (6.3) за пределами $n = \pm(N-1)/2$.Однако простое усечение ряда приводит к хорошо известному явлению Гиббса, которое проявляется в виде выбросов и пульсаций определенного уровня до и после точки разрыва в аппроксимируемой частотной характеристике. Так, например, при аппроксимации стандартных фильтров типа идеального фильтра нижних частот или полосового фильтра максимальная амплитуда пульсаций частотной характеристики составляет около 9% и *не* уменьшается с увеличением длины импульсной характеристики, т.е. учет все большего числа членов ряда Фурье не приводит к уменьшению максимальной амплитуды пульсаций. Вместо этого по мере увеличения Nуменьшается ширина выброса.

Лучшие результаты дает метод проектирования КИХ-фильтров, основанный на использовании весовой последовательности конечной длины $w(n)$, называемой окном, для модификации коэффициентов Фурье $h_d(n)$в формуле (6.3),с тем чтобы управлять сходимостью ряда Фурье.

Метод взвешивания иллюстрируется на рисРис. 6.5: заданная периодическая частотная характеристика $H_d(e^{j\omega})$и ее коэффициенты Фурье $h_d(n)$ – на рис. 6.5, а, весовая последовательность конечной длины $w(n)$и ее преобразование Фурье $W(e^{j\omega})$ – на рис. 6.5, б. Для большинства приемлемых окон функция $W(e^{j\omega})$ имеет главный лепесток, содержащий почти всю энергию окна, и боковые лепестки, которые обычно быстро затухают. Чтобы получить аппроксимацию функции $H_d(e^{j\omega})$, формируется последовательность $h(n) = h_d(n)w(n)$, в точности равная нулю за пределами интервала $-(N-1)/2 \le n \le (N-1)/2$.

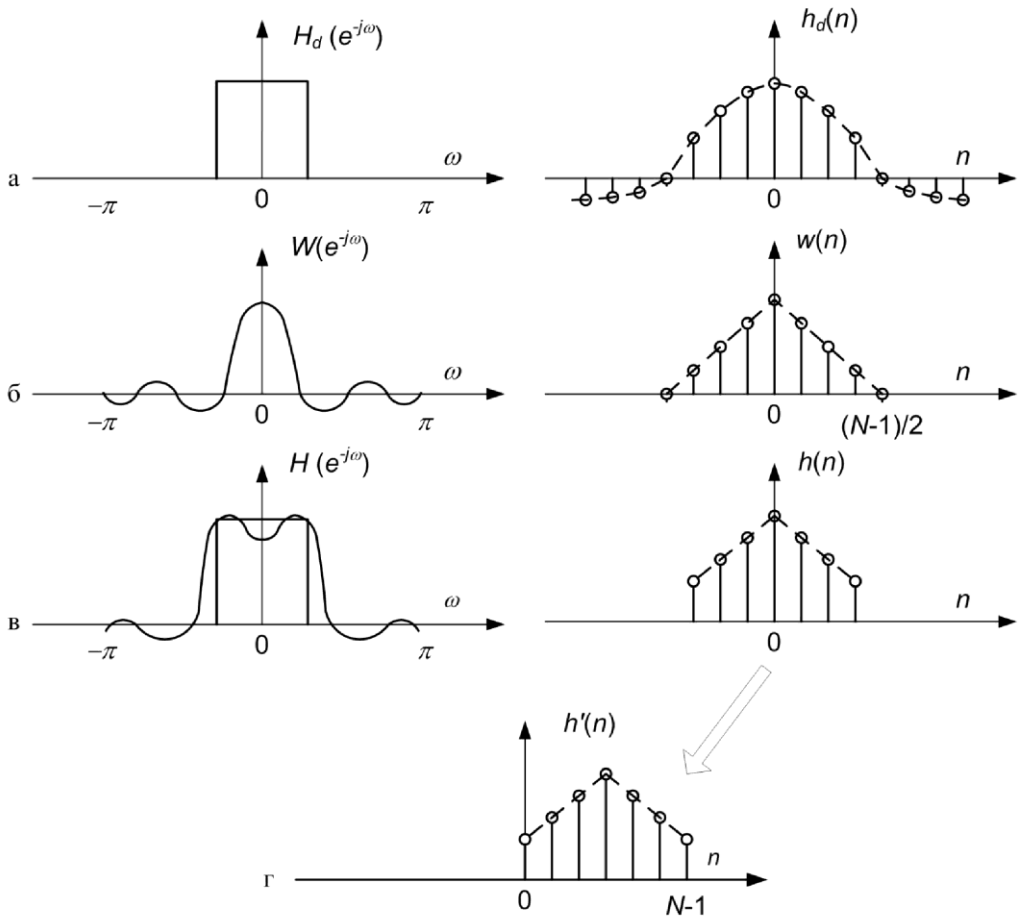


Рис. 6.5. Иллюстрация метода взвешивания

На рис. 6.5,впредставлена последовательность $h(n)$и ее преобразование Фурье $H(e^{j\omega})$, равное, очевидно, круговой свертке функций $H_d(e^{j\omega})$и $W(e^{j\omega})$, поскольку $h(n)$является произведением $h_d(n)w(n)$:

$$H(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta. \qquad (6.5)$$

Физически реализуемая последовательность $h'(n) = h(n-(N-1)/2)$, которая равна задержанной последовательности $h(n)$и может быть использована в качестве искомой импульсной характеристики фильтра показана на рис. 6.5, г. При этом частотная характеристика физически реализуемого фильтра будет равна$H(e^{j\omega})e^{j\omega(N-1)/2}$ и при соблюдении условий симметрии импульсной характеристики: $h'(n) = \pm h'(N-1-n)$ будет иметь линейную ФЧХ.

На примере, представленном на рис. 6.5, можно проследить влияние операции взвешивания коэффициентов Фурье фильтра на его частотную характеристику.

Прежде всего, по обе стороны от точек разрыва заданной функции $H_d(e^{j\omega})$появляются переходные полосы. Ясно, что поскольку результирующая частотная характеристика фильтра равна круговой свертке идеальной частотной характеристики и частотной характеристики окна, то ширина переходных полос зависит от ширины главного лепестка функции $W(e^{j\omega})$. Кроме того, на всех частотах ω возникают ошибки аппроксимации, имеющие вид пульсаций частотной характеристики, которые обусловлены боковыми лепестками функции $W(e^{j\omega})$. Ясно, наконец, и то, что получаемые фильтры ни в каком смысле *не являются оптимальными* (даже если окна и удовлетворяют тому или иному критерию оптимальности), поскольку их частотные характеристики рассчитываются через свертку.

После общего рассмотрения метода взвешивания возникают два вопроса: какими свойствами должны обладать окна и насколько точно они могут быть реализованы на практике? Ответ на первый вопрос относительно прост. Желательно, чтобы окно обладало следующими свойствами:

1) ширина главного лепестка частотной характеристики окна, содержащего по возможности большую часть общей энергии, должна быть малой;

2) энергия в боковых лепестках частотной характеристики окна должна быстро уменьшаться при приближении ωк π.

Было предложено много окон, аппроксимирующих заданные характеристики. В качестве примеров мы рассмотрим три окна, а именно: прямоугольное окно, «обобщенное» окно Хэмминга и окно Кайзера. Эти окна обладают свойствами всех возможных видов окон и позволяют достаточно хорошо понять преимущества и недостатки метода взвешивания.

Прямоугольное окно

N-точечное *прямоугольное окно*, соответствующее простому усечению (без модификации) ряда Фурье, описывается весовой функцией

$$w_R(n) = \begin{cases} 1, & -\frac{N-1}{2} \le n \le \frac{N-1}{2}, \\ 0, & \text{при других } n. \end{cases}$$

Здесь и далее предполагается, что N –нечетноечисло. С помощью простой модификации аналогичные результаты могут быть получены и для четного N. Предполагается также, что последовательность окна имеет нулевую задержку.Частотная характеристика прямоугольного окна описывается соотношением

$$W(e^{j\omega}) = \sum_{n=-(N-1)/2}^{(N-1)/2} e^{-j\omega n} = \frac{e^{j\omega(N-1)/2}(1-e^{-j\omega N})}{(1-e^{-j\omega})} =$$

$$= \frac{e^{j\omega N/2} - e^{-j\omega N/2}}{e^{j\omega/2} - e^{-j\omega/2}} = \frac{\sin(\omega N/2)}{\sin(\omega/2)}.$$

Для вычисления N-точечного прямоугольного окна в пакете Matlabможно использовать функцию `boxcar(N)`. График прямоугольного окна и его АЧХдля N =23 в диапазоне от $-\pi$ до πпредставлен на рис. 6.6.

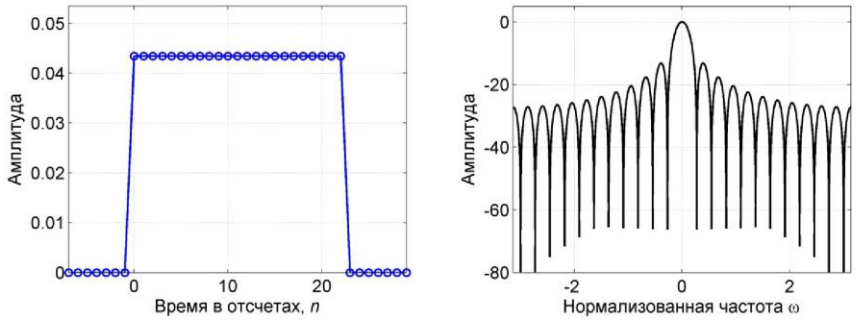


Рис. 6.6.Прямоугольное окно: представление во временной (а) и частотной (б) областях

Окно Бартлетта

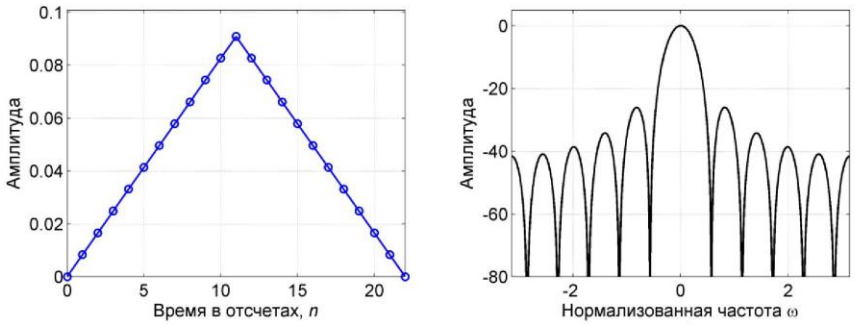
N-точечное *окно Бартлетта*–треугольное окно. Коэффициенты окна Бартлетта вычисляются по следующим формулам:

$$w(k)=\begin{cases}\dfrac{2k}{n-1}, & 0\le k\le\dfrac{n-1}{2},\\ 2-\dfrac{2k}{n-1}, & \dfrac{n-1}{2}\le k\le n-1,\end{cases}$$ (для нечетного n);

$$w(k)=\begin{cases}\dfrac{2k}{n-1}, & 0\le k\le\dfrac{n}{2}-1,\\ \dfrac{2(n-k-1)}{n-1}, & \dfrac{n}{2}\le k\le n-1,\end{cases}$$ (для четного n).

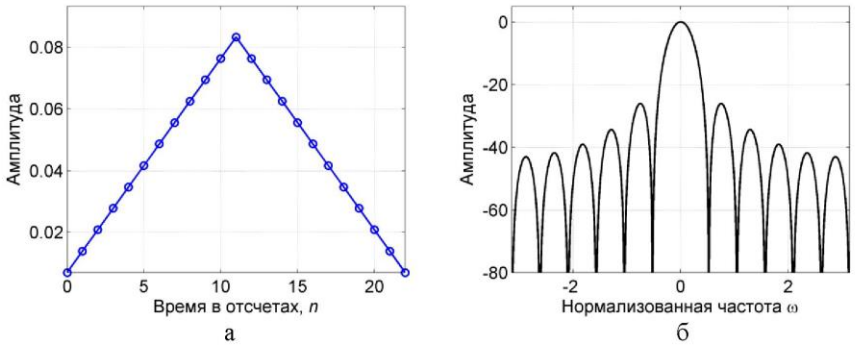
Окно Бартлетта почти полностью совпадает с треугольным окном.Отличие от треугольного окна заключается лишь в том, что значение окна Бартлеттапри $k = 1$ и $k = n$ равно нулю.

Для вычисления N-точечных окон Бартлетта и треугольного в Matlabможно использовать функции `bartlett(N)`и `triang(N)`соответственно.Графикии АЧХ окна Бартлетта и треугольного окнадля $N = 23$ в диапазоне от $-\pi$ до πпредставлены нарис. 6.7и 6.8 соответственно.



а б

Рис. 6.7.Окно Бартлетта: представление во временной (а) и частотной (б) областях



а б

Рис. 6.8.Треугольное окна: представление во временной (а) и частотной (б) областях

Обобщенное окно Хэмминга

Окно, называемое *обобщенным окном Хэмминга*, имеет вид

$$w_H(n) = \begin{cases} \alpha + (1-\alpha)\cos\left(\dfrac{2\pi n}{N-1}\right), & -\dfrac{N-1}{2} \le n \le \dfrac{N-1}{2}, \\ 0, \text{ при других } n, \end{cases} \tag{6.6}$$

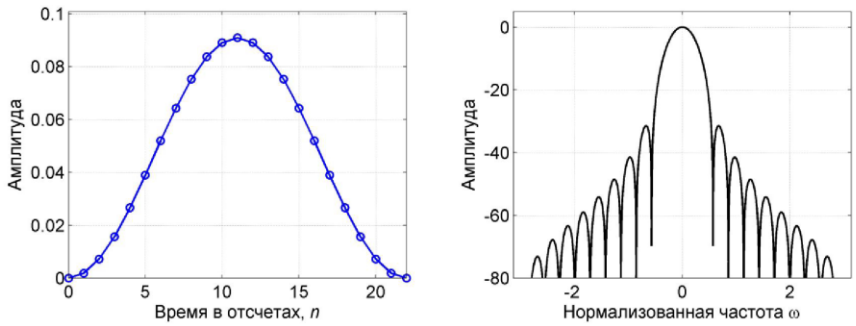
причем α лежит в пределах $0 \le \alpha \le 1$.Случай, когда$\alpha = 0{,}5$соответствует окну Ханна, когда$\alpha = 0{,}54$– окну Хэмминга. Частотную характеристику рассматриваемого окна легко получить, если учесть, что оно может быть представлено в виде произведения прямоугольного окна и окна, определяемого формулой(6.6), но для всех n, т. е.

$$w_H(n) = w_R(n)\left[\alpha + (1-\alpha)\cos\left(\frac{2\pi n}{N-1}\right)\right].$$

Следовательно, частотная характеристика обобщенного окна Хэмминга может быть записана в следующем виде:

$$W_H(e^{j\omega}) = \alpha W_R(e^{j\omega}) + \frac{1-\alpha}{2}W_R\left(e^{j[\omega-(2\pi/(N-1))]}\right) + \frac{1-\alpha}{2}W_R\left(e^{j[\omega+(2\pi/(N-1))]}\right).$$

Для вычисления N-точечного окна Ханна в Matlabможно использовать функцию `hann(N)`, окна Хэмминга – функцию `hamming(N)`, окна Блэкмана – функцию `blackman(N)`. Эти окна являются частным случаем обобщенного окна Хэмминга. Графики и АЧХ окон Ханна, Хэмминга и Блэкмана для $N = 23$ в диапазоне от $-\pi$ до πпредставлены на рис. 6.9, 6.10и 6.11 соответственно.



а б

Рис. 6.9.Окно Ханна: представление во временной (а) и частотной (б) областях

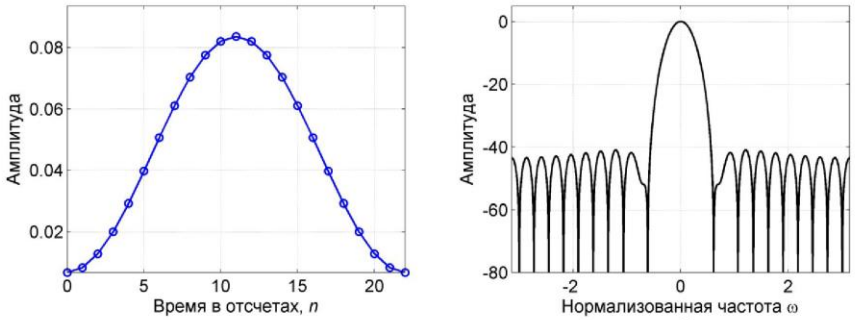


Рис. 6.10.Окно Хэмминга: представление во временной (а) и частотной (б) областях

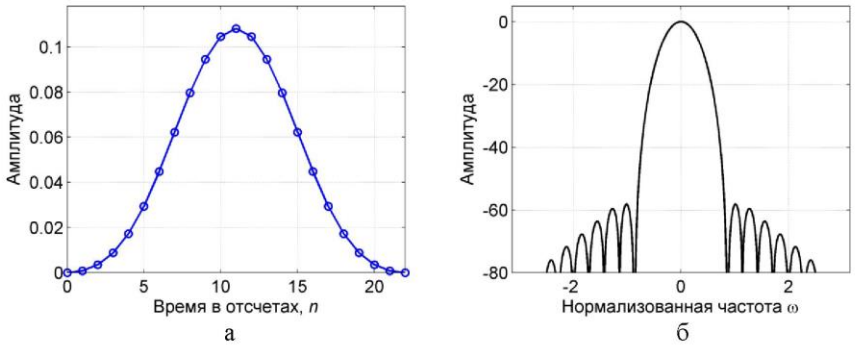


Рис. 6.11.Окно Блэкмана: представление во временной (а) и частотной (б) областях

Из рисунков видно, что ширина главного лепестка частотной характеристики окна Хэмминга в два раза больше, чем для прямоугольного окна. Однако уровень боковых лепестков в случае окна Хэмминга значительно ниже, чем у прямоугольного окна. При$\alpha = 0{,}54$, т. е. для обычного окна Хэмминга, 99,96 % общей энергии спектра содержится в главном лепестке, а максимумы боковых лепестков намного ниже главного максимума. В отличие от окнаХэмминга максимум боковых лепестков в спектре прямоугольного окна ниже главного максимума всего на 10–20 дБ.

При расчете фильтра нижних частот расширение главного лепестка соответствует расширению переходной полосы между полосами пропускания и непропускания, тогда как уменьшение уровня боковых лепестков соответствует меньшим пульсациям в полосе пропускания и лучшему подавлению в полосе непропускания фильтра.

Окно Кайзера

Задача расчета хороших окон фактически сводится к математической задаче отыскания ограниченных во времени функций, преобразования Фурье которых наилучшим образом аппроксимируют функции, ограниченные по часто-

Лабораторная работа №7 Исследование инструментальной среды программирования ЦСП

Цель работы: Приобрести навыки программирования типовых задач ЦОС с использованием инструментальной среды программирования

1 Структура окон рабочего пространства среды разработки

На рис.1 представлен вид рабочего пространства среды VisualDSP++ с открытым в ней проектом. Верхняя часть окна программы состоит из строки меню (File, Edit, Session, View, Project, Register, Memory, Debug, Settings, Tools, Window, Help) и панелей инструментов, обеспечивающих быстрый вызов требуемых функций и выполнение основных действий по созданию кода и его отладке.

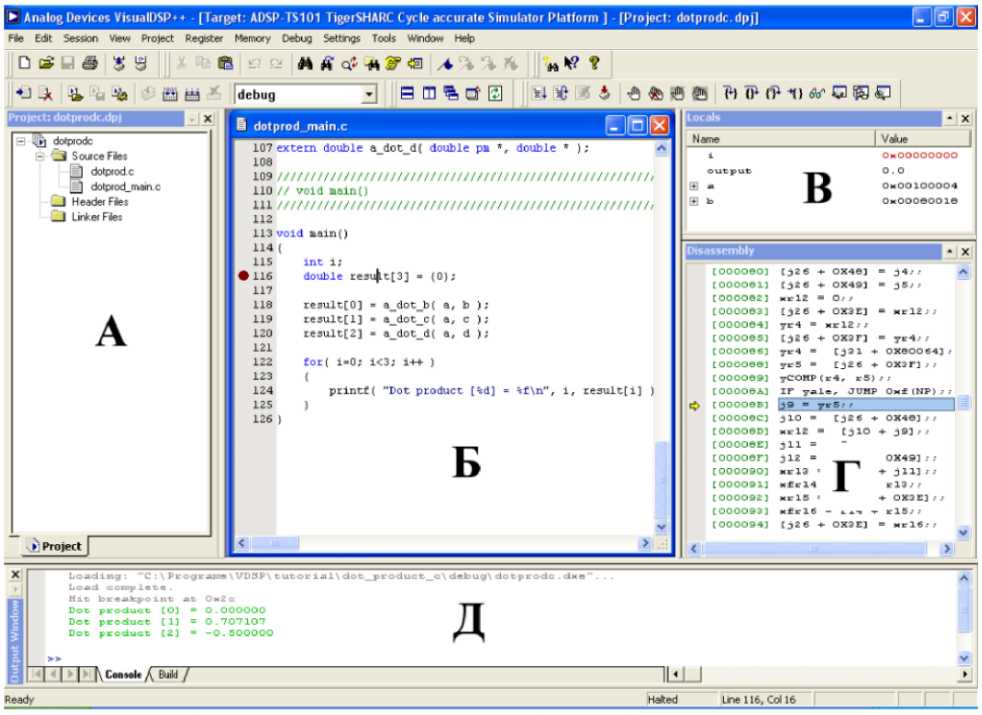


Рис. 1 Общий вид окна среды разработки VisualDSP++ с открытым проектом

Представленная на рис.1 раскладка рабочего пространства включает 5 открытых окон.

А – окно проекта. Здесь представлено дерево каталогов и файлов, составляющих проект;

Б – окно редактора исходного кода одного их файлов проекта;

В – окно вывода значений локальных переменных в процессе отладки программы;

Г – окно дизассемблера. Здесь отображается исполняемый код программы на языке ассемблера того сигнального процессора, для которого создается данный проект.

Д – одна из вкладок окна сообщений среды VisualDSP, в данном случае -вкладка консольных сообщений с результатами вывода отлаживаемой программы. В это окно выводится также информация о ходе компиляции и компоновки программы и сообщения об ошибках построения исполняемого кода.

2 Начало работы с VisualDSP ++

До начала работы с VisualDSP ++ необходимо установить это программное обеспечение на компьютер. Этот момент пропустим, т.к. установка VisualDSP ++ на компьютер не должна доставлять особых трудностей. Рекомендуется производить ее при отсутствии других активных приложений, в особенности других инсталляторов драйверов. Режим работы VisualDSP++ можно оперативно задать без переустановки программы через пункт меню Session -> New Session. Если у вас нет под рукой отладочной платы ADSP-BF533 EZ-KIT Lite, создаем режим симуляции. Уже в этом режиме мы сможем работать с инструментальной системой и изучать особенности архитектуры Blackfin.

Для этого достаточно открыть демонстрационные файл проект *.dpj, находящиеся в директории Example. Поскольку у нас речь идет о ADSP-BF533 EZ-KIT Lite, мы устанавливаем режим “EZ-KIT Lite”. для более подробной работы и изучения сигнального процессора. Но прежде чем задать этот режим, мы должны подключиться к отладочной плате через USB. USB обладает благоприятной для пользователя особенностью — поддержкой технологии plug and play (подключи и работай), что упрощает работу с драйверами. Убедившись, что подключение организовано между компьютером и отладочной платой, мы можем задать режим “EZ-KIT Lite”.

Таким образом, создав файл проект, либо выбрав в директории Example, запускаем проект на построение командой меню Project->Build Project. При этом модули проекта компилируются, затем связываются, размещаются и загружаются в память устройства ADSP-BF533 EZ-KIT Lite. Остается запустить программу на выполнение: выбираем пункт меню Debug -> Run.

Таким образом, мы сможем пронаблюдать результат, тем самым можно сделать вывод, что ничего сложного в освоении VisualDSP++ нет. Для более подробного освоения Visual DSP++ необходимо обратиться к Руководству по Visual DSP++.

3. Решение задач по программированию ЦСП

Рассмотрим технологию разработки программ для цифровых сигнальных процессоров фирмы Analog Devices ADSP-21262-EZ lite и ADSPˇ BF-533-EZ lite. Программы написаны на языке Си в среде разработки VisualDSP++ 5.0.

3.1 Реализация программы на мигание светодиодом для отладочного модуля ADSP-BF-533-EZ lite

Программа на мигание диодов не имеет ничего общего с цифровой обработкой сигналов, а является скорее простым способом познакомится с платой. Данная программа вызывает попеременное мигание диодов LED4, LED6, LED8 и LED5, LED7, LED9. Сам пример программы с комментариями представлен ниже:

```
#include <cdefBF533.h> // подключаем необходимые библиотеки
#include <stdio.h>
// addresses in Flash for the LED's
#define pFlashA_PortB_Dir (volatile unsigned char *)0x20270007
#define pFlashA_PortB_Data (volatile unsigned char *)0x20270005
int main(void)
{
// установка интерфейса внешней шины данных, для порта flash A
*pEBIU_AMBCTL0 = 0x7bb07bb0;
*pEBIU_AMBCTL1 = 0x7bb07bb0;
*pEBIU_AMGCTL = 0x000f;
// устанавливаем flash порт на выход, b 111111
*pFlashA_PortB_Dir = 0x3f;
unsigned int delay = 0;
while (1) {
*pFlashA_PortB_Data = 0x15; // b 010101
for (delay = 0x00FFFFFF; delay > 0; delay--);
*pFlashA_PortB_Data = 0x2A; // b 101010
for (delay = 0x00FFFFFF; delay > 0; delay--);
}
}
```

Командой *pFlashA_PortB_Dir = 0x3f; мы зажигаем все диоды, а участком программы

```
*pFlashA_PortB_Data = 0x15; // b 010101
for (delay = 0x00FFFFFF; delay > 0; delay--);
*pFlashA_PortB_Data = 0x2A; // b 101010
for (delay = 0x00FFFFFF; delay > 0; delay--);
```

вызываем попеременное прерывание определенных диодов.

Рис. 2 Процесс чередования загорания диодов

3.2 Реализация КИХ-фильтра на отладочной отладочном модуле ADSP˘ BF-533-EZ lite

Как говорилось ранее на лекциях фильтры делятся на два типа с конечной (КИХ) и бесконечной (БИХ) импульсной характеристикой. В данной лабораторной работе мы будем проектировать фильтр с конечной импульсной характеристикой.

Для начала посмотрим принцип его работы на основе фильтра низких частот представленного на рисунке 3.

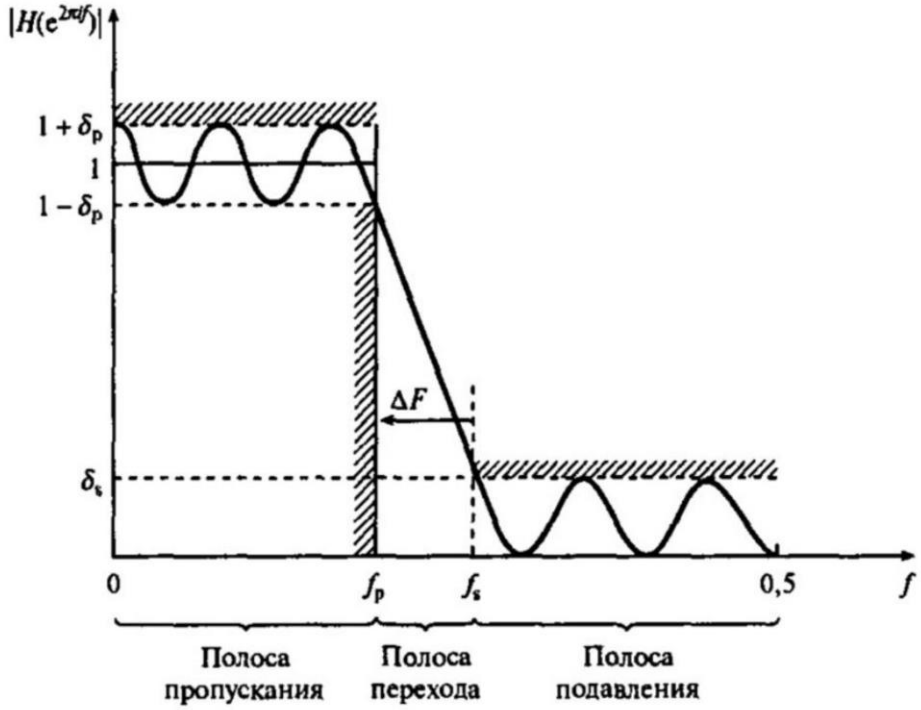


Рис. 3. Амплитудно-частотная характеристика фильтра низких частот.

Как видно по содержимому рисунка 3, этот фильтр пропускает нижние частоты, а все остальные старается отсечь (подавить), или хотя бы ослабить. Пульсации в полосе пропускания и полосе подавления (режекции) выбираются в зависимости от принимаемого сигнала, но при использовании различных весовых функций, на них могут накладываться определенные ограничения.

Принцип работы фильтра довольно прост, он получает последовательность входных отчетов и, с помощью умножения на коэффициенты (отчеты импульсной характеристики) преобразует их, затем выдает конечную последовательность. Отсюда определяются цикличность действий реализуемого алгоритма:

$$y(n) = \sum_{i=0}^{P} b_i x(n-i)$$

Исходя из АЧХ (рисунок 3) идеальная частотная характеристикой имеет вид прямоугольного окна. Обратное преобразование Фурье идеальной АЧХ даст нам отчеты импульсной характеристики. Но можно пойти и по более простому пути – есть уже заранее вычисленные идеальные импульсные характеристики, например для фильтра нижних частот формула выглядит следующим образом:

$$h_D(n) = 2f_c \times \frac{\sin(nw_c)}{nw_c} \text{ для } n \neq 0$$

$$h_D(n) = 2f_c \text{ для } n = 0$$

где wc и fc – частота среза.

Реальная характеристика является результатом аппроксимации идеальной АЧХ с использованием известных полиномов, например, с использованием весовых коэффициентов Хэмминг [11]:

$$w(n) = 0.53836 - 0.46164 \cos\left(\frac{2\pi n}{N-1}\right)$$

где N – длина фильтра, т.е количество коэффициентов.

Таким образом реализуемый алгоритм будет иметь вид свертки отчетов входного сигнала с весовыми коэффициентами Хэмминга:

$$h(n) = h_D(n) \times w(n)$$

Перед началом работы, необходимо установить переключатели отладочной платы SW9 и SW3 в правильное положение:

SW3-> OFF,OFF,OFF,OFF,OFF,ON

SW9-> ON,OFF,ON,ON,ON,ON.

За основу программы был взят пример, предложенный разработчиками, который можно найти по следующей директории: C:\Program Files (x86)\Analog Devices\ VisualDSP 5.0\Blackfin\ Examples\ ADSP-BF533 EZ-KIT

Lite\Audio Codec Talkthrough - I2S (C). Далее будет представлен фрагмент программы непосредственно отвечающий за преобразование сигнала:

```
#include "Talkthrough.h" //подключаем основные библиотеки
#include "math.h"
//--------------------------------------------------------------------------//
// Function: Process_Data()
// Описание: Эта функция вызывается изнутри SPORT0 ISR при каждом
//получении нового кадра. Новые данные получаются из переменных:
//iChannel0LeftIn, iChannel0RightIn, iChannel1LeftIn and iChannel1RightIn.
//Обработанные данные передаются в переменные: iChannel0LeftOut,
//iChannel0RightOut, iChannel1LeftOut, iChannel1RightOut, iChannel2LeftOut
//and iChannel2RightOut.
//--------------------------------------------------------------------------//
volatile long unsigned Sum_ = 0;
#define W 14 //задаем размерность фильтра
void Process_Data(void)
{
unsigned char i;
float S[W]; //переменная для обработки входного сигнала
float H[W] = {0.0175, 0.0072, 0.0119, 0.0187, 0.0261, 0.0372, 0.0387, 0.0369,
0.0322, 0.0256, 0.0181, 0.0109, 0.0049, 0.0008}; //заранее посчитанные
коэффициенты фильтра
volatile long unsigned Sum_ = 0;
for (i=1;i<W;i++)
{
 S[i-1] = S[i] ;
}
37
S[W-1] = iChannel0LeftIn/2 ; //обработка входного сигнала
//Нормировка импульсной характеристики
double SUM=0;
for (i=0; i<W; i++) SUM +=H[i];
for (i=0; i<W; i++) H[i]/=SUM; //сумма коэффициентов равна 1
for(i=0;i<W;i++)
{
 Sum_ += S[i]*H[i]; //перемножение входного сигнала с коэффициентами
}
iChannel0LeftOut = Sum_; // вывод обработанного сигнала
}
```

С генератора, встроенного в осциллограф, на вход платы ЦСП подается прямоугольный сигнал с частотой 1кГц. Изображение входного и выходного сигналов (после низкочастотной фильтрации) представлены на рисунке 4.

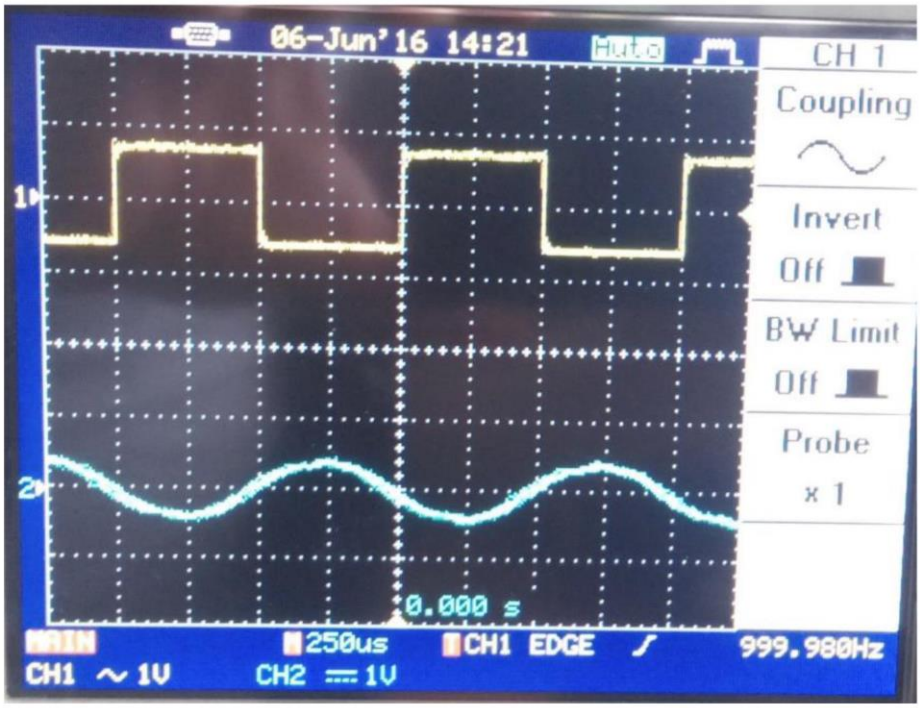


Рис. 4. Результат обработки прямоугольного сигнала.

На рисунке 5 изображена отладочная плата и схема подключения аудио контактов.

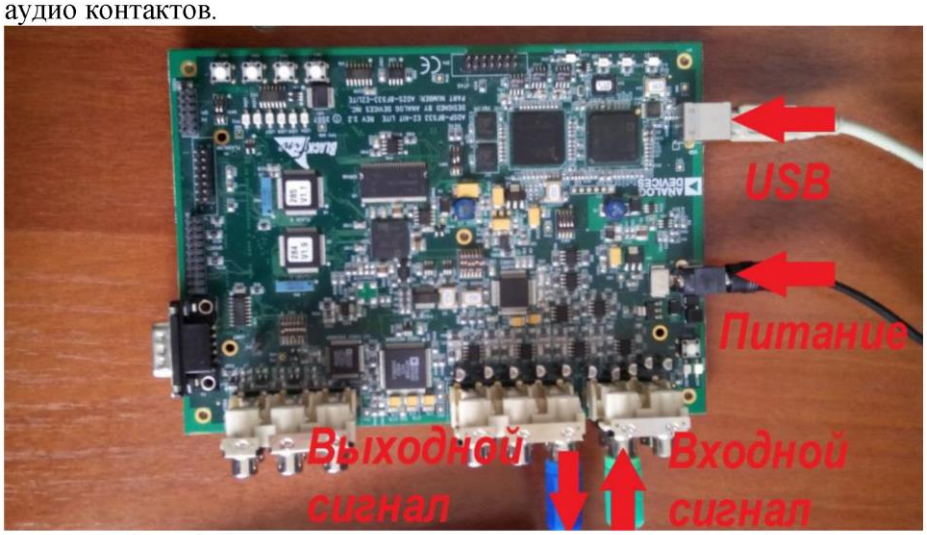


Рис. 5. Схема подключения контактов к отладочной плате.

3.3 Реализация сглаживающего фильтра на отладочной отладочном модуле ADSP-BF-533-EZ lite используя каналы Background Telemetry Channel (BTC).

BTC представляет собой способ обмена данными между встроенным процессором (The Blackfin в данном случае) и хостом (вашим компьютером). BTC хост и целевое приложение имеют доступ для чтения / записи общих регистров, используемых BTC в реальном времени. Все это позволяет просматривать входные и выходные данные через интерфейс отладчика в реальном времени.

В данном упражнении для преобразования сигнала мы будем использовать метод экспоненциального скользящего. Это простейший сглаживающий цифровой фильтр. Alpha - параметр фильтра определяющий степень сглаживания. Он может принимать значения от 0 до 1. Чем он меньше, тем сильнее будет сглаживаться входной сигнал. В частных случаях при Alpha=0 фильтр перестает реагировать на изменения входного сигнала, а при Alpha=1 он просто повторяет входной сигнал.

Запишем формулу, экспоненциального скользящего:

$$Y(n) = alpha*Y(n)+(1-alpha)*X(n)$$

Перед началом работы, необходимо установить переключатели SW9 и SW3 в правильное положение:

SW3-> OFF,OFF,OFF,OFF,OFF,ON

SW9-> ON,OFF,ON,ON,OFF,OFF

За основу программы был взят пример, предложенный разработчиками, который можно найти по следующей директории: C:\Program Files (x86)\Analog Devices\VisualDSP 5.0\Blackfin\Examples\ADSP-BF533 EZ-KIT Lite\Background_Telemetry\AudioDemo. Далее будет представлен фрагмент программы непосредственно отвечающий за преобразование сигнала:

```
#include "Talkthrough.h" //запись основных библиотек
#include <btc.h>
#define ALPHA 0.8 //запись коэффициента сглаживания
void Process_Data(void)
{
//запись данных в каналы BTC
int nValue;
extern int BTCLeftVolume;
extern int BTCRightVolume;
//запись данных входного сигнала
nValue = (iChannel0LeftIn >> 8); //upper 24-bits
nValue = (nValue >> BTCLeftVolume); //амплитуда
btc_write_value(0, (unsigned int*)&nValue, sizeof(nValue));
iChannel0LeftOut = (ALPHA*iChannel0LeftOut + (1-
ALPHA)*((nValue<<8)/2)); //преобразование входного сигнала
```

```
//запись данных выходного сигнала
nValue = (iChannel0LeftOut >> 8); //upper 24-bits
nValue = (nValue >> BTCRightVolume);//амплитуда
btc_write_value(1, (unsigned int*)&nValue, sizeof(nValue));
iChannel0LeftOut = (nValue << 8);
}
```

Для того, чтобы пронаблюдать входной и выходной сигналы на экране компьютера необходимо выполнить следующие действия:

Открыть окно каналов BTC для входного и выходного сигналов (View>>Debug Windows-Plot-Restore) рисунок 6.

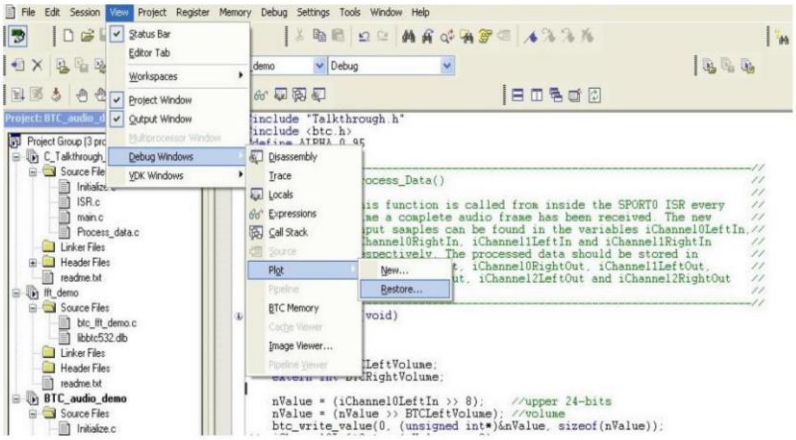


Рис. 6. Открытие окна канала BTC.

Далее дважды кликнуть на файл left.vps и right.vps, для открытия окна входного и выходного сигналов соответственно, рисунок 7.

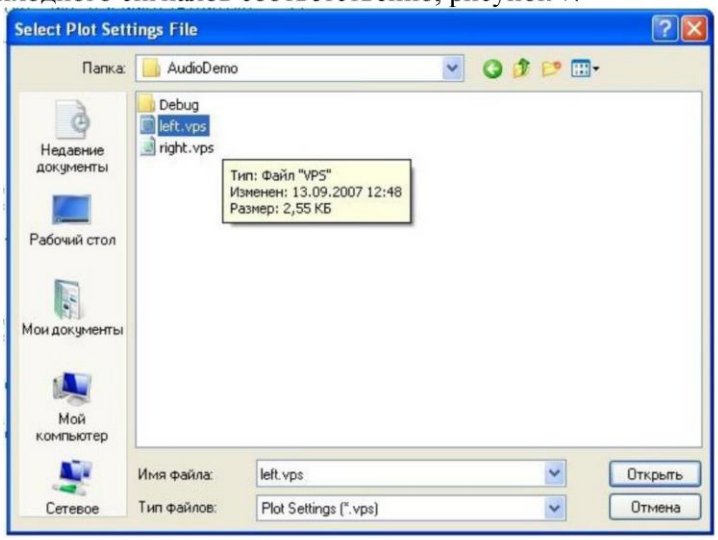


Рис. 7. Файлы для входного и выходного сигнала.

В результате нам откроется окно BTC канала, рисунок 8.

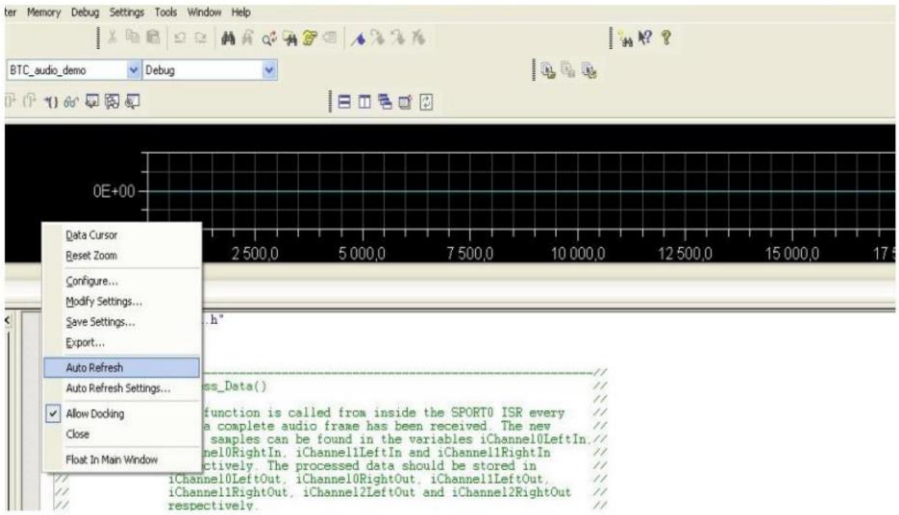


Рис. 8. Открытое окно канала BTC.

Для того, чтобы увидеть входной и выходной сигналы необходимо запустить программу и установить авто-обновление графиков. Для этого необходимо кликнуть правой кнопкой мыши на окне BTC и выбрать функцию “Auto Refresh".

В результате мы получим визуализацию входного и выходного (преобразованного) сигнала. Изменяя значение коэффициента сглаживания Alpha мы будем получать разный выходной сигнал. На рисунках 9 и 10 представлены графики входного и выходного сигнала при значении Alpha 0.95 и 0.5 соответственно.

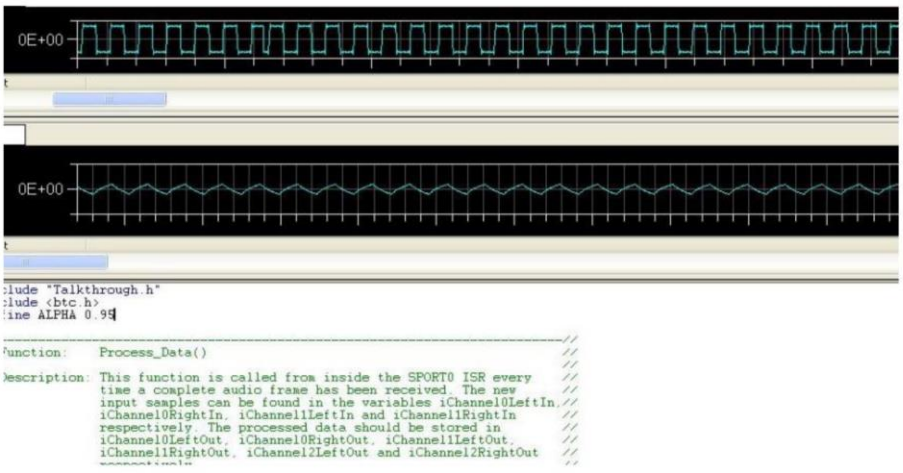


Рис. 9 Входной и выходной сигнал при значении Alpha = 0.95.

Рис. 10. Входной и выходной сигнал при значении Alpha = 0.5.

3.4 Реализация программы на включение светодиодов для отладочного модуля ADSP-21262-EZ lite.

Заключительное задание представляет собой управляющий алгоритм и реализует программу включения встроенных светодиодов отладочного модуля ADSP-21262-EZ lite. Данное упражнение показывает, как нажимая на пользовательские кнопки добиться включения определенных диодов.

Пользовательские кнопки SW1, SW2, SW3 и SW4 соединены с регистрами IRQ1, IRQ2, DAI_P19, и DAI_P20 соответственно. Нажатие на определенную кнопку вызывает прерывание определенного флага, тем самым зажигая светодиод.

За основу программы был взят пример, предложенный разработчиками, который можно найти по следующей директории: C:\Program Files (x86)\Analog Devices\VisualDSP 5.0\212xx\examples\ADSP-21262 EZ-KIT Lite\EZkit Push Button (C). Далее будет представлен фрагмент программы отвечающий за зажигание светодиодов:

```
#include <def21262.h> //подключение основных библиотек
#include <cdef21262.h>
#define SRUDEBUG // Check SRU Routings for errors.
#include <SRU.h>
#include <signal.h>
#define LED1 1 //ввод переменных для диодов
#define LED2 2
#define LED3 4
```

```
#define LED
void DAIroutine(int); //ввод переменных для прерываний
void IRQ1_routine(int);
void IRQ2_routine(int);
void handle_LED(int);
main(){ //начало основной программы
//Установки приоритета прерывания
 *pDAI_IRPTL_PRI = SRU_EXTMISCB1_INT |
SRU_EXTMISCB2_INT; //включение прерываний
 *pDAI_IRPTL_RE = SRU_EXTMISCB1_INT |
SRU_EXTMISCB2_INT;
 *pSYSCTL |= IRQ1EN|IRQ2EN;
 asm("#include <def21262.h>");
 asm("bit set mode2 IRQ1E|IRQ2E;");
 interrupt(SIG_DAIH,DAIroutine);
interrupt(SIG_SPIH,DAIroutine);
 interrupt(SIG_IRQ1,IRQ1_routine);
 interrupt(SIG_IRQ2,IRQ2_routine);
//Pin Assignments in SRU_PIN3 (Group D)
 //assign pin buffer 19 low so it is an input
 SRU(LOW,DAI_PB19_I);
 //assign pin buffer 20 low so it is an input
 SRU(LOW,DAI_PB20_I);
//Route MISCB singnals in SRU_EXT_MISCB (Group E)
 //route so that DAI pin buffer 19 connects to MISCB1
 SRU(DAI_PB19_O,MISCB1_I);
 //route so that DAI pin buffer 20 connects to MISCB2
 SRU(DAI_PB20_O,MISCB2_I);
//Pin Buffer Disable in SRU_PINEN0 (Group F)
 //assign pin 19 low so it is an input
 SRU(LOW,PBEN19_I);
 //assign pin 20 low so it is an input
 SRU(LOW,PBEN20_I);
 for(;;)
 {}
}
void IRQ1_routine(int sig_int){
 handle_LED(LED1);
}
void IRQ2_routine(int sig_int){
 handle_LED(LED2);
}
```

```
//Set up interrupt service routines to toggle LEDs 3 and 4.
void DAIroutine(int sig_int){
 static int interrupt_reg;
 interrupt_reg = *pDAI_IRPTL_H;
 //test for SRU_EXTMISCB1_INT
 if ((interrupt_reg & SRU_EXTMISCB1_INT) != 0)
 handle_LED(LED3);
 //test for SRU_EXTMISCB2_INT
 if ((interrupt_reg & SRU_EXTMISCB2_INT) != 0)
 handle_LED(LED4);
}
void handle_LED(int led_value){
//lights as described at the top of the file
 *pPPCTL=0;
 *pIIPP=(int) &led_value;
 *pIMPP=1;
*pICPP=1;
 *pEMPP=1;
 *pECPP=1;
 *pEIPP=0x400000;
 *pPPCTL=PPTRAN|PPBHC|PPDUR20|PPDEN|PPEN;
}
```

Запустив программу мы можем убедиться в том, что нажатие на кнопки SW1, SW2, SW3 и SW4 вызывает загорание диодов LED1, LED2, LED3 и LED4 соответственно. Процесс загорания диодов представлен на рисунке 11.

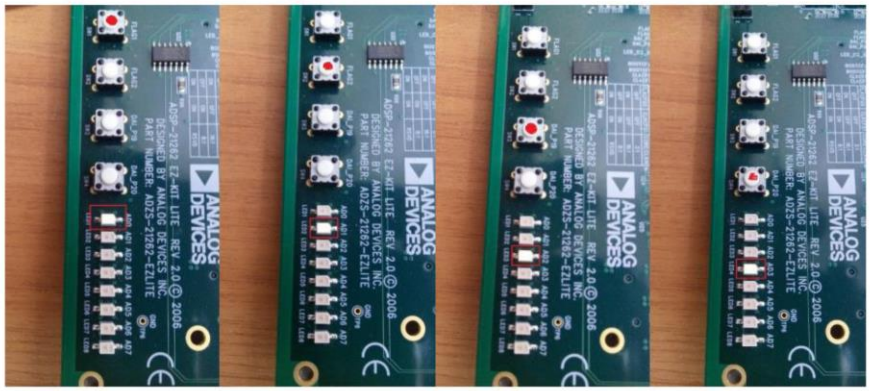


Рис. 1. Процесс управления состоянием светодиодов LED1, LED2, LED3 и LED4.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Баженов А.В., Линец Г.И. Цифровая обработка сигналов: учебное пособие./А.В. Баженов, Г.И. Линец.- Ставрополь:Издательско-информационный центр «Фабула», 2022.-178 с. ISBN 978-5-91903-278-6

2. Пасечников, И. И.
    Цифровая обработка сигналов Электронный ресурс / Пасечников И. И. : учебное пособие. - Тамбов : ТГУ им. Г.Р.Державина, 2019. - 156 с. - ISBN 978-5-00078-261-3, экземпляров неограничено

3. Бастракова М.И. Схемотехника телекоммуникационных устройств: практикум / М.И. Бастракова, В.В. Павлов; Поволжский государственный технологический университет. - Йошкар-Ола: ПГТУ, 2019. - 52 с.: схем., ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-5-8158-2073-9, экземпляров неограниченно.

4. Дуркин В.В. Схемотехника аналоговых электронных устройств Электронный ресурс / Дуркин В. В., Тырыкин С. В., Белоруцкий Р. Ю.: учебно-методическое пособие. - Новосибирск: НГТУ, 2019. - 88 с. - ISBN 978-5-7782-3937-1, экземпляров неограниченно.

5.Водовозов, А. М. Микроконтроллеры для систем автоматики [Электронный ресурс] : учебное пособие / А. М. Водовозов. — Электрон. текстовые данные. — М. : Инфра-Инженерия, 2016. — 164 c. — 978-5-9729-0138-8. — Режим доступа: http://www.iprbookshop.ru/51727.html

6.Основы программирования микропроцессорных контроллеров в цифровых системах управления технологическими процессами [Электронный ресурс] : учебное пособие / В. С. Кудряшов, А. В. Иванов, М. В. Алексеев [и др.]. — Электрон. текстовые данные. — Воронеж : Воронежский государственный университет инженерных технологий, 2014. — 144 c. — 978-5-00032-054-9. — Режим доступа: http://www.iprbookshop.ru/47437.html

7.Шегал, А. А. Применение программного комплекса Multisim для проектирования устройств на микроконтроллерах [Электронный ресурс] : лабораторный практикум / А. А. Шегал ; под ред. В. И. Иевлев. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2014. — 116 c. — 978-5-7996-1117-0. — Режим доступа: http://www.iprbookshop.ru/65968.html

8. Алиев, М. Т.
    Микропроцессоры и микропроцессорные системы управления. 8-разрядные процессоры семейства AVR : лабораторный практикум / М.Т. Алиев, Т.С. Буканова ; Поволжский государственный технологический университет. - Йошкар-Ола : ПГТУ, 2016. - 64 с. : схем., табл., ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-5-8158-1775-3, экземпляров неограничено

9. Безуглов, Д. А. Цифровые устройства и микропроцессоры : учеб. пособие / Д. А. Безуглов, И. В. Калиенко. - Изд. 2-е. - Ростов н/Д : Феникс, 2008. - 468 с. : ил. - (Высшее образование). - Библиогр.: с. 464-465. - ISBN 978-5-222-13917-2

10.Хартов, В. Я. Микроконтроллеры AVR : практикум для начинаю-

щих : учеб. пособие / В. Я. Хартов. - М. : МГТУ, 2007. - 240 с. : ил. - Прил.: с. 231-238. - Библиогр.: с. 239. - ISBN 978-5-7038-3051-2

11.Руководство пользователя по программированию ПЛК в CoDeSys 2.3//https://owen.ru/product/codesys_v2/documentation

12. Баженов, А.В. Цифровые методы реализации пространственно-временной обработки сигналов: Учебное пособие / А.В. Баженов, Н.В. Гривенная, М.Ф. Горяинов, С.В. Малыгин – Ставрополь: ТИС, 2016. – 219 с., ил.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по организации и проведению самостоятельной работы
по дисциплине «Цифровые сигнальные процессоры (DSP)»
для студентов направления подготовки
11.04.04 Электроника и наноэлектроника
Направленность (профиль):
Интеллектуальные программируемые системы и устройства

Ставрополь
2026

Содержание

ВВЕДЕНИЕ

Современные стандарты подготовки магистрантов предусматривают значительный объем внеаудиторной работы. Освоение программы курса предполагает получение как теоретических знаний по цифровой обработки сигналов, так и формирование навыков практической работы с цифровыми сигнальными процессорами. Поэтому самостоятельная работа в рамках курса ориентирована как на теоретический, так и на практический аспект.

Самостоятельная работа студентов – это выполнение теоретических и практических заданий студентами по усвоению изучаемой дисциплины «Цифровые сигнальные процессоры (DSP)»

Целью самостоятельной работы является закрепление и углубление знаний, полученных студентами на лекциях, подготовке к текущим практическим занятиям, промежуточным формам контроля знаний и к итоговому контролю.

Дидактические цели самостоятельных занятий:

- формирование профессиональных умений;
- формирование умений и навыков самостоятельного умственного труда;
- мотивирование регулярной целенаправленной работы по освоению специальности;
- развитие самостоятельности мышления;
- формирование убежденности, волевых черт характера, способности к самоорганизации;
- овладение технологическим учебным инструментом.

Самостоятельная работа включает те разделы курса, которые не получили достаточного освещения на лекциях по причине ограниченности лекционного времени и большого объема изучаемого материала.

Методическое обеспечение самостоятельной работы по дисциплине состоит из:

1) Определения учебных вопросов, которые студенты должны изучить самостоятельно;

2) Подбора необходимой учебной литературы, обязательной для проработки и изучения;

3) Поиска дополнительной научной литературы, к которой студенты могут обращаться по желанию, если у них возникает интерес в данной теме;

4) Определения контрольных вопросов, позволяющих студентам самостоятельно проверить качество полученных знаний;

5) Организации консультаций преподавателя со студентами для разъяснения вопросов, вызвавших у обучающихся затруднения при самостоятельном освоении учебного материала.

Самостоятельная работа студента – это особым образом организованная деятельность, включающая в свою структуру такие компоненты, как:

- уяснение цели и поставленной учебной задачи;
- четкое и системное планирование самостоятельной работы;
- поиск необходимой учебной и научной информации;
- освоение собственной информации и ее логическая переработка;
- использование методов исследовательской, научно-исследовательской работы для решения поставленных задач;
- выработка собственной позиции по поводу полученной задачи;
- представление, обоснование и защита полученного решения;
- проведение самоанализа и самоконтроля

Виды самостоятельных работ по учебной дисциплине:

- работа с понятийным аппаратом;
- конспектирование первоисточников;
- проектирование цифровых устройств на ЦСП;
- анализ научных исследований;

– подготовка научного проекта по теме.

Контроль знаний студентов включает формы текущего и итогового контроля. Текущий контроль осуществляется в процессе изучения курса и включает в себя оценку работы студентов на практических занятиях (собеседование, групповые научные проекты, тестирование), а также подготовку реферата.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ТЕОРЕТИЧЕСКОГО МАТЕРИАЛА

Комплексное изучение предлагаемой студентам учебной дисциплины «Цифровые сигнальные процессоры (DSP)» предполагает овладение материалами лекций, учебников, программы, творческую работу студентов в ходе проведения практических занятий, а также систематическое выполнение заданий для самостоятельной работы.

В ходе лекций раскрываются основные вопросы в рамках рассматриваемых тем, делаются акценты на наиболее сложные и интересные положения изучаемого материала, которые должны быть приняты студентами во внимание. Материалы лекций являются основой для подготовки студентов к практическим занятиям.

Работа с конспектом лекций

Просмотрите конспект сразу после занятий. Отметьте материал конспекта лекций, который вызывает затруднения для понимания. Попытайтесь найти ответы на затруднительные вопросы, используя предлагаемую литературу. Если самостоятельно не удалось разобраться в материале, сформулируйте вопросы и обратитесь на текущей консультации или на ближайшей лекции за помощью к преподавателю.

Каждую неделю отводите время для повторения пройденного материала, проверяя свои знания, умения и навыки по контрольным вопросам и тестам.

Выполнение практических занятий по дисциплине «Цифровые сигнальные процессоры (DSP)»

На первом занятии получите у преподавателя задания по курсу, планы подготовки к практическим занятиям. Обзаведитесь всем необходимым методическим обеспечением.

Задачи исследований, проводимых на практических занятиях:

- освоение инструментальных средств разработки программ для ЦСП,
- изучение языков описания схем,
- ознакомление с современными устройствами на базе ЦСП,
- разработка и реализация устройств бытовой и промышленной электроники на ЦСП,
- разработка и реализация алгоритмов цифровой обработки,
- организация интерфейса между основной ЭВМ и устройством на базе ЦСП,

Выполнение лабораторных работ по дисциплине «Цифровые сигнальные процессоры (DSP)»

На первом занятии получите у преподавателя задания по курсу, планы подготовки к лабораторным работам и согласуйте форму отчета по лабораторным работам. Обзаведитесь всем необходимым методическим обеспечением.

Задачи исследований, проводимых на лабораторных работах:

- исследование возможностей инструментальных средств разработки программ для ЦСП,
- исследование языков описания схем,
- исследование архитектурных особенностей современных ЦСП (DSP),
- исследование типовых технических решений, интеллектуальных программируемых системы и устройств на ЦСП,
- разработка и исследование алгоритмов цифровой обработки.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО КОНСПЕКТИРОВАНИЮ ПЕРВОИСТОЧНИКОВ

Изучение и анализ литературных источников является обязательным видом самостоятельной работы студентов. Изучение литературы по избранной теме имеет своей задачей проследить характер постановки и решения определенной проблемы различными авторами, аргументацию их выводов и обобщений, провести анализ и систематизировать полученный материал на основе собственного осмысления с целью выяснения современного состояния вопроса.

Проработка отобранного материала обязательно должна идти с одновременным ведением записей прочитанного и своих замечаний. Запись может иметь как форму конспекта, так и выписок, а также картотеку положений, тезисов, идей, методик, что в дальнейшем облегчит классификацию и систематизацию полученного материала. Такого рода записи являются лучшим способом накопления и первичной обработки материал, одной из обязательных форм организации умственного труда.

Планом удобно пользоваться при подготовке к устному выступлению по выбранной теме. Каждый пункт плана должен раскрывать одну из сторон избранной темы, а весь план должен охватывать ее целиком.

Тезисы предполагают сжатое изложение основных положений текста в форме утверждения или отрицания. Они являются более совершенной формой записей и представляют основу для дискуссии (собеседования). К тому же их легко запомнить.

Конспектирование – процесс мысленной переработки и письменной фиксации информации, в виде краткого изложения основного содержания, смысла какого-либо текста.

Результат конспектирования – запись, позволяющая конспектирующему немедленно или через некоторый срок с нужной полнотой восстановить полученную информацию. Конспект в переводе с латыни означает «обзор». По существу, его и составлять надо как обзор, содержащий основные мысли текста без подробностей и второстепенных деталей. Конспект носит индивидуализированный характер: он рассчитан на самого автора и поэтому может оказаться малопонятным для других.

Для того чтобы осуществлять этот вид работы, в каждом конкретном случае необходимо грамотно решить следующие задачи:

1. Сориентироваться в общей композиции текста (уметь определить вступление, основную часть, заключение).
2. Увидеть логико-смысловую канву сообщения, понять систему изложения автором информации в целом, а также ход развития каждой отдельной мысли.
3. Выявить «ключевые» мысли, т. е. основные смысловые вехи, на которые «нанизано» все содержание текста.
4. Определить детализирующую информацию.
5. Лаконично сформулировать основную информацию, не перенося на письмо все целиком и дословно.

Во всяком научном тексте содержится информация 2-х видов: основная и вспомогательная. Основной является информация, имеющая наиболее существенное значение для раскрытия содержания темы или вопроса. К ней относятся: определения научных по-

нятий, формулировки законов, теоретических принципов и т. д. Назначение вспомогательной информации – помочь читателю лучше усвоить предлагаемый материал. К этому типу информации относятся разного рода комментарии. Основную – записываем как можно полнее, вспомогательную, как правило, опускаем. Содержание конспектирования составляет переработка основной информации в целях ее обобщения и сокращения. Обобщить – значит представить ее в более общей, схематической форме, в виде тезисов, выводов, отдельных заголовков, изложения основных результатов и т. п. Читая, мы интуитивно используем некоторые слова и фразы в качестве опорных. Такие опорные слова и фразы называются ключевыми. Ключевые слова и фразы несут основную смысловую и эмоциональную нагрузку содержания текста.

Выбор ключевых слов – это первый этап смыслового свертывания, смыслового сжатия материала.

Важными требованиями к конспекту являются наглядность и обозримость записей и такое их расположение, которое давало бы возможность уяснить логические связи и иерархию понятий.

По форме *конспекты подразделяются* на формализованные и графические.

1. Формализованные (все записи вносятся в заранее подготовленные таблицы).

Это удобно, во-первых, при конспектировании материалов, когда перечень характеристик описываемых предметов или явлений более, или менее постоянен, во-вторых, при подготовке единого конспекта по нескольким источникам. Особенно если есть необходимость сравнения отдельных данных. Разновидностью формализованного конспекта является запись, составленная в форме ответов на заранее подготовленные вопросы, обеспечивающие исчерпывающие характеристики однотипных предметов или явлений.

2. Графические (элементы конспектируемой работы располагаются в таком виде, при котором видна иерархия понятий и взаимосвязь между ними).

По каждой работе может быть не один, а несколько графических конспектов, отображающих книгу в целом и отдельные ее части. Ведение графического конспекта - наиболее совершенный способ изображения внутренней структуры книги, а сам этот процесс помогает усвоению ее содержания.

Можно выделить следующие основные **типы конспектов: плановый, текстуальный, сводный, тематический.**

Плановый – легко получить с помощью предварительно сделанного плана произведения, каждому вопросу плана отвечает определенная часть конспекта:

а) вопросно-ответный (на пункты плана, выраженные в вопросительной форме, конспект дает точные ответы);

б) схематичный плановый конспект (отражает логическую структуру и взаимосвязь отдельных положений).

Текстуальный – это конспект, созданный в основном из цитат.

Сводный конспект – сочетает выписки, цитаты, иногда тезисы; часть его текста может быть снабжена планом.

Тематический – дает более или менее исчерпывающий ответ (в зависимости из числа привлеченных источников и другого материала, например, своих же записей) на поставленный вопрос – тему: обзорный; хронологический.

Роль конспекта – чисто учебная: он помогает зафиксировать основные понятия и положения первичного текста и в нужный момент их воспроизвести, например, при написании реферата или подготовке к экзамену.

Способы конспектирования.

- *Тезисы* – это кратко сформулированные основные мысли, положения изучаемого материала. Тезисы лаконично выражают суть читаемого, дают возможность раскрыть содержание. Приступая к освоению записи в виде тезисов, полезно в самом тексте отмечать места, наиболее четко формулирующие основную мысль, которую автор доказывает (ес-

ли, конечно, это не библиотечная книга). Часто такой отбор облегчается шрифтовым выделением, сделанным в самом тексте.

- ***Линейно-последовательная запись текста.*** При конспектировании линейно – последовательным способом целесообразно использование плакатно-оформительских средств, которые включают в себя следующие:

· сдвиг текста конспекта по горизонтали, по вертикали;
· выделение жирным (или другим) шрифтом особо значимых слов;
· использование различных цветов;
· подчеркивание;
· заключение в рамку главной информации.

- С***пособ «вопросов – ответов».*** Он заключается в том, что, поделив страницу тетради пополам вертикальной чертой, конспектирующий в левой части страницы самостоятельно формулирует вопросы или проблемы, затронутые в данном тексте, а в правой части дает ответы на них.

Одна из модификаций способа «вопросов – ответов» - таблица, где место вопроса занимает формулировка проблемы, поднятой автором (лектором), а место ответа – решение данной проблемы. Иногда в таблице могут появиться и дополнительные графы: например, «мое мнение» и т. п.

- С***хема с фрагментами –*** способ конспектирования, позволяющий ярче выявить структуру текста, - при этом фрагменты текста (опорные слова, словосочетания, пояснения всякого рода) в сочетании с графикой помогают созданию рационально - лаконичного конспекта.

- ***Простая схема –*** способ конспектирования, близкий к схеме с фрагментами, объяснений к которой конспектирующий не пишет, но должен уметь давать их устно. Этот способ требует высокой квалификации конспектирующего. В противном случае такой конспект нельзя будет использовать.

- ***Параллельный способ*** конспектирования. Конспект оформляется на двух листах параллельно или один лист делится вертикальной чертой пополам и записи делаются в правой и в левой части листа.

Однако лучше использовать разные способы конспектирования для записи одного и того же материала.

Комбинированный конспект – вершина овладения рациональным конспектированием. При этом умело используются все перечисленные способы, сочетая их в одном конспекте (один из видов конспекта свободно перетекает в другой в зависимости от конспектируемого текста, от желания и умения конспектирующего). Именно при комбинированном конспекте более всего проявляется уровень подготовки и индивидуальность студента.

Принципы составления конспекта прочитанного.

1. Записать все выходные данные источника: автор, название, год и место издания. Если текст взят из периодического издания (газеты или журнала), то записать его название, год, месяц, номер, число, место издания.

2. Выделить поля слева или справа, можно с обеих сторон. Слева на полях отмечаются страницы оригинала, структурные разделы статьи или книги (названия параграфов, подзаголовки и т. п.), формулируются основные проблемы. Справа – способы фиксации прочитанной информации.

Один из видов чтения – углубленное – предполагает глубокое усвоение прочитанного и часто сохранение информации в целях последующего обращения к ней. Эффективность

такого чтения повышается, если прочитанное зафиксировано не только в памяти, но и на бумаге. Психологи утверждают, что записанное лучше и полнее усваивается, прочнее откладывается в памяти. Установлено, что если прочитать 1000 слов и затем записать 50, подытоживающих прочитанное, то коэффициент усвоения будет выше, чем, если прочитать 10000 слов, не записав ни одного. Кроме того, при записи прочитанного формируется навык свертывания информации. И, наконец, чередование чтения и записывания уменьшает усталость, повышает работоспособность и производительность умственного труда.

1. Резюмирование.

Резюме – краткий итог прочитанного, содержащий его оценку. Резюме характеризует основные выводы книги, главные итоги.

Выбор языковых средств для построения резюме-выводов подчинен основной задаче свертывания информации: минимум языковых средств – максимум информации. Это обычно одно – три четких, кратких, выразительных предложения, раскрывающих, по мнению автора, самую суть описываемого объекта.

2. Аннотация.

Аннотация – краткая обобщенная характеристика печатной работы (книги, статьи), включающая иногда и его оценку. Это наикратчайшее изложение содержания первичного документа, дающее общее представление о теме.

Основное ее назначение – дать некоторое представление о книге (статье, научной работе) с тем, чтобы рекомендовать ее определенному кругу читателей или воспользоваться своими записями при выполнении работы исследовательского, реферативного характера. Поэтому аннотации не требуется изложения содержания произведения, в ней лишь перечисляются вопросы, которые освещены в первоисточнике (содержание этих вопросов не раскрывается). Аннотация отвечает на вопрос: «О чем говорится в первичном тексте?», дает представление только о главной теме и перечне вопросов, затрагиваемых в тексте первоисточника.

Текст аннотации не стандартизирован. В научной литературе можно встретить различные требования к составлению аннотаций. Например, текст справочной аннотации может включать следующие сведения:

· тип и название аннотируемого документа (монография, диссертация, сборник, статья и т. п.)
· задачи, поставленные автором аннотируемого документа
· метод, которым пользовался автор (эксперимент, сравнительный анализ, компиляция других источников)
· принадлежность автора к определенной научной школе или направлению
· структуру аннотируемого документа
· предмет и тему произведения, основные положения и выводы автора
· характеристику вспомогательных иллюстративных материалов, дополнений, приложений, справочного аппарата, включая указатели и библиографию.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПОДГОТОВКЕ К ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Контроль и оценка знаний студентов является неотъемлемой составной частью учебно-воспитательного процесса в вузе. Экзаменационная сессия – самая ответственная пора в студенческой жизни: она является итогом большой работы студентов в семестре.

Промежуточная аттестация по дисциплине проводится в виде зачета с оценкой. Отдельное время на проведение зачета учебным планом не предусмотрено, поэтому важно своевременно выполнять все контрольные мероприятия и получать максимально возможные баллы. Для этого необходим0 готовиться к каждому собеседованию, проводимому преподавателем во время защиты отчетов по лабораторным работам и результатов выполнения индивидуальных практических заданий. Таким образом подготовка к зачету с оценкой представляет собой в систематический труд на протяжении семестра, учебного года, охватывающий все формы учебного процесса: лекции, изучение и конспектирование рекомендованной литературы, активное участие в практических и лабораторных занятиях.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Перечень основной литературы:

1. Баженов А.В., Линец Г.И. Цифровая обработка сигналов: учебное пособие./А.В. Баженов, Г.И. Линец.- Ставрополь:Издательско-информационный центр «Фабула», 2022.- 178 с. ISBN 978-5-91903-278-6

2. Пасечников, И. И.
    Цифровая обработка сигналов Электронный ресурс / Пасечников И. И. : учебное пособие. - Тамбов : ТГУ им. Г.Р.Державина, 2019. - 156 с. - ISBN 978-5-00078-261-3, экземпляров неограничено

3. Бастракова М.И. Схемотехника телекоммуникационных устройств: практикум / М.И. Бастракова, В.В. Павлов; Поволжский государственный технологический университет. - Йошкар-Ола: ПГТУ, 2019. - 52 с.: схем., ил. - http://biblioclub.ru/. - Библиогр. в кн. - ISBN 978-5-8158-2073-9, экземпляров неограниченно.

4. Дуркин В.В. Схемотехника аналоговых электронных устройств Электронный ресурс / Дуркин В. В., Тырыкин С. В., Белоруцкий Р. Ю.: учебно-методическое пособие. - Новосибирск: НГТУ, 2019. - 88 с. - ISBN 978-5-7782-3937-1, экземпляров неограниченно.

Перечень дополнительной литературы:

1. Шефер, Е. А.
    Цифровая обработка изображений : учебное пособие / Е. А. Шефер. - Цифровая обработка изображений,2031-02-04. - Электрон. дан. (1 файл). - Санкт-Петербург : Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2019. - 100 с. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 2227-8397, экземпляров неограничено

Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине (модулю)

1. Методические рекомендации к практическим занятиям по дисциплине «Цифровые сигнальные процессоры»: Направление подготовки 11.04.04. «Электроника и наноэлектроника/ Сост. А. В. Баженов. – Электронный ресурс. – Ставрополь: СКФУ, 2022. экземпляров неограниченно.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)

1 http://elibrary.rsl.ru
2 http://www.edu.ru
3 http://www.ict.edu.ru
4 http://www.openet.edu.ru
5 www.ncfu.ru/nauchnaya-biblioteka-skfu.html
6 http://elibrary.rsl.ru