Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования

**«Владимирский государственный университет имени
Александра Григорьевича и Николая Григорьевича Столетовых»**

Кафедра радиотехники и радиосистем

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ СИГНАЛОВ В СИСТЕМЕ MATLAB

Методические указания к лабораторным работам

Составитель
Е.К. ЛЕВИН

Владимир 2011

УДК 621.396.62
ББК 32.884.1
И85

Рецензент
Доктор технических наук, профессор
Владимирского государственного университета
П.А. Полушин

Печатается по решению редакционного совета
Владимирского государственного университета

Исследование алгоритмов обработки сигналов в системе
И85 MATLAB : метод. указания к лабораторным работам / Владим. гос. ун-т ; сост. Е. К. Левин. – Владимир : Изд-во Владим. гос. ун-та, 2011. – 78 с.

Рассмотрены исследования алгоритмов обработки сигналов, которые широко используются в системах компьютерной телефонии: эффективное кодирование сигналов, их фильтрация и спектральный анализ. Для анализа алгоритмов применяется система MATLAB, которая в настоящее время является признанным во всем мире эффективным инструментом для исследования и моделирования различных систем в разных областях науки и техники. В процессе выполнения лабораторных работ студенты знакомятся с основными функциями и возможностями системы, приобретают навыки программирования на М-языке, знакомятся с пакетом расширения Simulink, который позволяет быстро составить из типовых блоков сложную модель исследуемой системы. В конце издания проводится исследование сложного алгоритма сжатия потока данных речи, элементы которого применяются в современных системах связи.

Предназначены для магистрантов и студентов старших курсов направления «Радиотехника» специальностей 210301– «Радиофизика и электроника», 210302 – «Радиотехника» и 210405 – «Радиосвязь, радиовещание и телевидение» дневной и заочной форм обучения.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Табл. 1. Ил. 36. Библиогр.: 3 назв.

УДК 621.396.62
ББК 32.884.1

ВВЕДЕНИЕ

Современные методы обработки сигналов в системах различного назначения широко используют цифровое представление сигналов, которое позволяет применить сложные алгоритмы нелинейной обработки. К таким алгоритмам, в частности, относится сжатие потока данных речи для систем компьютерной телефонии. Данные алгоритмы настолько сложные, что получить итоговое аналитическое выражение для результата обработки не представляется возможным. Сложность задачи усугубляется и тем, что помехи и информационные сигналы являются случайными.

В силу указанных выше проблем для исследования алгоритмов обработки сигналов широко применяется имитационное моделирование исследуемых систем. Здесь неоценимую помощь исследователю может оказать система MATLAB, которая во всем мире признана как универсальный инструмент для исследования различных технических систем. Благодаря большой библиотеке функций, удобному интерфейсу она быстро завоевывает симпатии у новых пользователей.

В связи с изложенным является актуальным использование комплекса лабораторных работ, который обеспечивает получение практических навыков по применению системы MATLAB в процессе исследования алгоритмов обработки сигналов, при подготовке магистрантов, бакалавров и специалистов по направлению «Радиотехника».

Лабораторная работа 1

ИЗУЧЕНИЕ ИНТЕРФЕЙСА СИСТЕМЫ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MATLAB

Цель работы. Изучение пользовательского интерфейса и основных математических операций системы MATLAB.

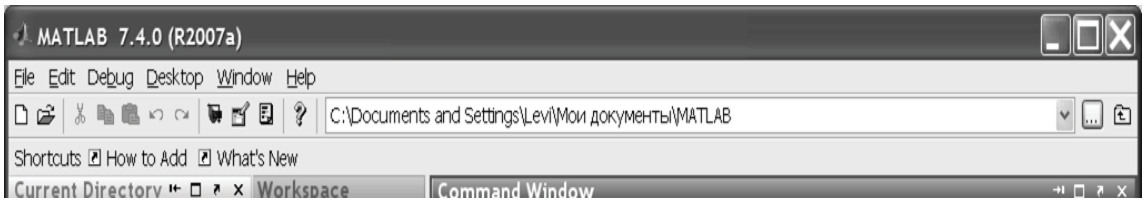
Задание 1. Изучить пользовательский интерфейс системы MATLAB с помощью информации, изложенной ниже, и выполнить все примеры, приведенные в тексте.

Основные сведения о пользовательском интерфейсе, командах и функциях системы MATLAB (на примере версии MATLAB 7.4.0)

После запуска системы MATLAB на экране компьютера появляется окно программы, которое по умолчанию разбито на три части.

Ввод данных отображается в командной части окна (**Command Window** – окно команд). В левой нижней части окна (**Command History)** отображается история введенных команд. В левой верхней части отображается либо рабочее пространство (**Workspace**) – те переменные, которые используются в данной сессии работы с системой и хранятся в памяти компьютера, либо содержание текущей (**Current Directory**) – рабочей директории.

Текущая директория системы MATLAB указывается в верхней части окна программы (см. рисунок).



Следует сразу же установить нужную рабочую директорию, в которой будут храниться все файлы, создаваемые в дальнейшем.

Имена переменных должны начинаться с буквы, причем в системе MATLAB учитывается регистр символов. В переменной **ans** хранится результат последних вычислений, если для него не указана переменная.

Знак = соответствует операции присваивания переменной некото-

рого значения. Нажатие клавиши **Enter** запускает процесс вычислений, например,

a = (3+5)*2-37
a= -21

Клавиши "Стрелка вверх" и "Стрелка вниз" позволяют вернуть в строку ввода ранее введенные с клавиатуры команды (**Command History**) и другую входную информацию. Вся видимая информация в командной части окна системы MATLAB располагается в двух зонах: просмотра и редактирования.

Зона редактирования обычно занимает одну (последнюю) строку командного окна – строка ввода, в которой находится знак приглашения ». При необходимости строку редактирования можно распространить на несколько физических строк командного окна. Для продления ввода с показом вводимой информации на следующих физических строках требуется нажать **Enter** только после трех или более точек, например,

A=5+4+…
+7+8

Командой **clc** можно стереть видимое содержимое командного окна системы MATLAB, однако это не затронет содержимого рабочего пространства. Переменные стираются из памяти компьютера командой:

clear имя1 имя2 …, удаляющей из рабочего пространства переменные с соответствующими именами. Чтобы удалить все переменные, следует использовать команду **clear.**

Если дважды щелкнуть левой клавишей мыши на строке, расположенной либо в окне истории команд, либо в окне рабочего пространства, то соответствующая команда или переменная появится в командном окне. Для просмотра значения любой переменной из текущего рабочего пространства достаточно набрать ее имя и нажать клавишу **Enter**.

После закрытия сеанса работы все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню **File | Save Workspace As...,** после чего появляется стандартное диалоговое окно операционной системы Windows для выбора каталога на диске и имени файла. Расширение имени файла должно быть **mat**. В последующих сеансах работы для загрузки в память компьютера ра-

нее сохраненного на диске “рабочего пространства” нужно выполнить команду меню: **File | Load Workspace...** либо набрать команду: **load имя_MAT-файла.**

Если имя MAT-файла не содержит полного пути к нему, то файл должен находиться в текущем каталоге системы MATLAB, который указывается в верхней части окна программы.

Чтобы сохранить историю команд для использования в последующих сеансах работы, следует сначала удалить все неправильно введенные команды в окне истории команд, затем выделить нужный фрагмент истории команд и с помощью контекстного меню сохранить выделенные строки в текстовом файле с расширением **m** (файл сценария). Для повторения сеанса работы следует либо скопировать строки файла в окно команд и нажать клавишу **Enter**, либо ввести имя файла сценария (без указания расширения) в окно команд и нажать клавишу **Enter** (файл должен находиться в текущей директории).

По любой команде и функции системы MATLAB можно получить быструю справку, выполнив команду: **help имя_команды (функции)**.

Система MATLAB осуществляет вычисления как с вещественными, так и с комплексными числами. Числа задаются в системе MATLAB мантиссой и показателем степени и записываются в одном из перечисленных вариантов:

5.2e-2; 3.44e+4; 5.7; 45;
5+8i; 7.9+0.5e3*i; 4+3j

Здесь буквой **e** обозначается основание степени, равное 10.

Для записи мнимой единицы зарезервированы (на выбор) буквы **i** или **j**. Если коэффициентом при мнимой единице является не число, а переменная, то следует использовать знак умножения: **x + *i* * y**. Под мантиссу и показатель степени отводится 8 байт памяти. Для записи комплексного числа требуется в два раза больше памяти по сравнению с вещественным числом.

Над числами и переменными производятся арифметические операции сложения, вычитания, умножения и деления, для которых в системе MATLAB используются знаки: **+ , – , *** и **/** . Кроме того, есть еще операция возведения в степень, обозначаемая знаком ^.

9 ^ (0.5)
ans=3

Приоритет в выполнении арифметических операций обычный: сначала – возведение в степень, затем – умножение и деление и потом – сложение и вычитание. Операции одинакового приоритета выполняются в порядке слева направо, но круглые скобки могут изменить этот порядок.

После введения с клавиатуры некоторого выражения обычно ставится точка с запятой. При этом результат вычислений не появляется в окне команд после выполнения вычислений – оно не «загрязняется» ненужной информацией. Особенно это важно при работе с большими матрицами. Точка с запятой используется также для разделения различных выражений, расположенных в одной строке редактирования.

A=5+8; B=6-3;

В системе MATLAB присутствуют все основные элементарные функции для вычислений с вещественными числами: степенные, показательные, тригонометрические и обратные к ним. Любая функция характеризуется своим именем, списком входных аргументов (перечисляются через запятую и стоят внутри круглых скобок, следующих за именем функции) и вычисляемым (возвращаемым) значением.

x=0.25*pi
R=sin(x); d=tan(x);

Здесь **pi** – число пи. Перечень некоторых наиболее часто используемых функций указан в таблице.

Функция	Описание функции
sqrt(x)	Функция извлечения квадратного корня
exp(x)	Возведение в степень числа **e**
log(x)	Натуральный логарифм
log10(x)	Логарифм по основанию 10
1og2(x)	Логарифм по основанию 2
round(x)	Округление до ближайшего целого
floor(x)	Округление до меньшего целого
ceil(x)	Округление до большего целого
cos(x)	Косинус
abs(x)	Абсолютное значение комплексного числа
angle(x)	Аргумент комплексного числа (в радианах)

Примеры:

X=2+2i; y1= abs(X); y2= angle(X);
B=round(5.7); B1=floor(5.7); B2=ceil(5.2)

Массивы

Именованные наборы чисел называют массивами. Всему массиву присваивается одно имя, а доступ к отдельным элементам массива осуществляется по целочисленному индексу, то есть по номеру элемента в массиве. Для создания одномерного массива можно использовать операцию конкатенации, которая обозначается с помощью квадратных скобок **[]**, либо с помощью знака **:** (двоеточие). В последнем случае значения элементов массива отличаются на фиксированную величину. При создании двумерного массива методом конкатенации используется знак **;** (точка с запятой) для разделения строк.

B=[2, 7, 9, 40, 75, 5+8i]
L= 5:2:16
M=[4,5,67,78; 33,5,6,89]

Вторая строка примера формирует массив следующим образом: число 5 – начало массива, число 2 – разность значений соседних элементов (приращение) массива, а число 16 – верхняя граница массива, которую нельзя превышать **L= 5 7 9 11 13 15**.

Третья строка формирует двумерный массив из двух строк и четырех столбцов:

M= 4 5 67 78
33 5 6 89

Для доступа к индивидуальному элементу одномерного массива нужно после его имени указать в круглых скобках индекс (номер) элемента. Например, **B(3)**. Для доступа к индивидуальному элементу двумерного массива нужно после его имени указать в круглых скобках номер строки и номер столбца элемента:

M(2,1)
ans=33

Количество элементов в одномерном массиве определяется с помощью функции **length**:

length(B)
ans =6

Размеры двумерного массива определяются функцией **size**.

```
size(M)
ans = 2   4
```

Первым показывается число строк, а вторым – число столбцов.

Выделение фрагментов матриц

Для удаления строк или столбцов соответствующему фрагменту массива присваивается пустое значение **[]**:

Здесь знак (**:**) означает “все элементы массива вдоль данной размерности” (в данном случае вдоль второй строки). Применим эту операцию для следующего массива

```
X=  5  0  0  0  0
    0  2  0  5  0
    0  1  0  0  1
    0  0  0  0  6
```

получим

```
X=  5  0  0  0  0
    0  1  0  0  1
    0  0  0  0  6
```

```
X (2,:) = [ ]
```

Операция **A(2:3 , 2:4)** означает выделение из матрицы **A** блока, простирающегося от второй до третьей строки и от второго до четвертого столбца:

```
A=  17  24   1   8  15
    23   5   7  14  16
     4   6  13  20  22
    10  12  19  21   3
    11  18  25   2   9
```

получим:

```
ans=  5   7  14
      6  13  20
```

Таким образом, можно выделить из массива произвольный прямоугольный блок. Если выделяемый блок по одному из измерений совпадает с размером исходного массива, то вместо задания диапазона индексов можно указать просто двоеточие (как и в примере с удалением фрагмента массива):

```
A(3:4,:)
ans= 4   6  13  20  22
    10  12  19  21   3
```

Некоторые отступления от общего синтаксиса возможны тогда, когда выделяемый фрагмент “упирается” в границу массива справа или снизу. В этом случае для обозначения последнего (максимального) возможного значения индекса по какой-либо размерности можно применять ключевое слово **end.** Это значение можно использовать при вычислении индексов. Например, выделим из массива **A** блок размером 2x3, расположенный в правом нижнем углу:

```
A(end-1:end, end-2:end)
ans= 19   21   3
     25    2   9
```

Выделяемый фрагмент не обязательно должен быть сплошным. Выделим из массива **A** второй и четвертый элементы первой и последней строк:

```
A([1 end], [2 4])
ans= 24   8
     18   2
```

Для выделения из массива абсолютно произвольного набора элементов используют обращение по линейному (то есть одномерному) индексу. Элементы двумерных массивов при этом нумеруются по столбцам. Выделим из массива **A** элементы, лежащие на кросс-диагонали:

```
A([5  9  13  17  21])
ans =    11  12  13  14  15
```

Вычисления с массивами

В системе MATLAB можно производить групповые вычисления над массивами, используя обычные математические функции, например,

```
x= 0 : 0.01 : pi/2; y = sin(x);
```

Над массивами одинаковых размеров допускаются операции сложения и вычитания, обозначаемые стандартными знаками + и –. **C=A+B;D=A-B.**

Если используются операнды разных размеров, то выдается сообщение об ошибке за исключением случая, когда один из операндов является скаляром:

```
A+ 5
```

В таких случаях скаляр предварительно расширяется до массива раз-

мерами с матричный операнд, а затем поэлементно складывается с матрицей **A**. Поменять местами строки матрицы с ее столбцами можно операцией транспонирования, которая обозначается знаком **'** (апостроф).

```
A=[1 1 1;  2 2 2;  3 3 3]; B = A'
B=
1  2  3
1  2  3
1  2  3
```

Знак * закреплен за перемножением матриц и векторов в смысле линейной алгебры. Операция выполнима только тогда, когда число столбцов в левом операнде равно числу строк в правом:

```
A=[1 2 3; 4 5 6; 7 8 9]; x=[1, 2, 3]'; B=A * x ;
B =
    14
    32
    50
```

Здесь вектор-столбец **x** задан не с помощью операции вертикальной конкатенации [1; 2; 3], а использована операция транспонирования, и из вектор-строки [1, 2, 3] получен нужный вектор-столбец.

Знак / (а также знак \) закреплен в системе MATLAB за решением задачи линейной алгебры – нахождением корней систем линейных уравнений.

Для поэлементного перемножения и деления массивов одинаковых размеров применяются операции, обозначаемые комбинациями двух символов: **.*** и **./**. Кроме операции **./**, есть еще операция **.**. Выражение **A./B** приводит к матрице с элементами **A(k,m)/B(k,m),** а выражение **A.\B** – к матрице с элементами **B(k,m)/A(k,m)**.

```
M=[4,5,67,78; 33,5,6,89]; N=2*[4,5,67,78; 33,5,6,89];
K=M./N;
```

Для определения суммы и среднего значения элементов столбцов матрицы используются функции соответственно: **sum(x), mean(x)**.

```
D= sum(M); D1= mean(M);
```

Задание 2

1. Выполнить вычисления с матрицами согласно заданному варианту.

2. Сохранить итоговое рабочее пространство в созданном под именем пользователя каталоге.

3. Отредактировать «Историю выполненных команд» (Command History), сохранить ее в файле и привести распечатку в отчете.

4. Подготовить таблицу с описанием изученных команд и функций.

Содержание отчета

1. Распечатка истории команд.
2. Таблица с описанием изученных команд и функций.
3. Результаты выполнения примеров и заданных вычислений.

Контрольные вопросы

1. Какими способами можно получить помощь при работе в системе MATLAB?
2. Что означает понятие «рабочее пространство»?
3. Как сохранить рабочее пространство и историю команд?
4. Продемонстрировать умение пользоваться всеми изученными командами и функциями.

Лабораторная работа 2
ОПЕРАЦИИ НАД ЗВУКОВЫМИ ФАЙЛАМИ. ГРАФИЧЕСКОЕ ОТОБРАЖЕНИЕ РЕЗУЛЬТАТОВ ВЫЧИСЛЕНИЙ

Цель работы. Изучение приемов работы с массивами и файлами, а также средств визуализации результатов работы.

Задание 1. Изучить материал, изложенный ниже, и повторить примеры и упражнения, приведенные в тексте.

Операции над звуковыми файлами

Система MATLAB позволяет читать и записывать файлы формата **wav.** Чтение осуществляется с помощью функции **wavread**, а запись – с помощью функции **wavwrite**.

Упражнение

Выбрать в одном из системных каталогов Windows стереофониче-

ский звуковой файл, переместить его в текущую директорию и запустить на исполнение функцию **wavread.**

[V,f,b] = wavread ('*.wav');

Звуковая информация из файла считывается в матрицу **V,** состоящую из двух столбцов, в скалярную переменную **f** считывается значение частоты дискретизации, а в переменную **b** – число двоичных разрядов.

Добавим к матрице **V** белый шум.

```
[m,n] = size(V);
V1 = randn( m,n );
s = 0.05;
V2 = V + s * V1;
```

Здесь в первой строке определяется число строк и столбцов матрицы **V**. Вторая строка с помощью функции **randn(m,n)** создает матрицу **V1**, которая содержит случайные числа, распределенные по закону Гаусса (по нормальному закону), и которая имеет тот же размер, что и матрица **V**. Матрица **V1** умножается на коэффициент **s** и складывается с матрицей **V**, образуя матрицу **V2** зашумленного звука.

Функция **sound** осуществляет воспроизведение звука, получая в качестве аргумента вещественный вектор или матрицу размерами **N x 2** (для стереозвука), содержащие последовательности измерений громкости звука.

```
sound(V, f, b )
sound(V2, f, b)
```

Значения элементов матрицы отсчетов должны быть ограничены диапазоном от –1.0 до +1.0. Вне этого диапазона значения матрицы ограничиваются (имеет место клиппированный звук). Аналогичная функция **soundsc** перед воспроизведением звука обеспечивает автоматическое масштабирование значений матрицы до диапазона: –1… +1 – клиппирования звука не происходит.

```
soundsc(V, f, b )
soundsc(V2, f, b)
```

Чтобы сохранить результаты экспериментов со звуком в звуковом файле, следует применить функцию **wavwrite**. В данном случае результаты эксперимента с зашумлением сохраним в файле **'Sound1.wav'**:

```
wavwrite(V2, f, b, 'Sound1.wav');
```

Здесь первым аргументом является матрица (для монофонического звука – это вектор) звуковых отсчетов, вторым – частота дискретизации, третьим – разрядность отсчетов, а последним – имя файла. Если не указывать пути к файлу, то он будет записан в текущий рабочий каталог пакета MATLAB.

Построение графиков функций

Двумерная графика

Построим график функции, соответствующей гармоническому колебанию **y=A*sin(2*pi*F*t+alfa)**, где скалярные переменные **A, F, alfa** определяют соответственно амплитуду, частоту и фазу (в радианах) колебания. Переменная **pi** определяет число **π**. Переменная **t** – одномерный массив, задающий интервал времени колебания. Определим массив **y** при следующих числовых данных:

```
A=2;
F=100;
alfa=pi/4;
Fd=1000;
t=0:1/Fd:0.05;
```

Здесь массив **t** задает интервал времени от нуля до 0,05 с с периодом дискретизации **1/Fd,** (**Fd** – частота дискретизации гармонического колебания). Вычислим массив **y** и с помощью функции **plot(t, y)** построим график гармонического колебания

```
y=A*sin(2*pi*F*t+alfa);
plot(t, y)
```

Для более удобного анализа графика нанесем на него сетку с помощью функции **grid** (**grid on –** наносит сетку на график, **grid off –** ее убирает):

```
grid on
```

Функция **plot(t,y)** строит график следующим образом: результаты вычислений – точки на графике – соединяет прямыми линиями, то есть при построении графика используется линейная интерполяция результатов вычислений. Графические объекты отображаются в специальных графических окнах, которые имеют меню и собственную панель инструментов с кнопками.

Если необходимо сохранить в файле содержимое графического окна, то выполняется команда меню графического окна **File|Save as...**, и выбирается для расширения имени файла тип **fig**, в котором сохраняются графические объекты, содержащиеся в текущем графическом окне.

Ранее полученные графические объекты могут быть восстановлены (даже если из рабочего пространства удалены соответствующие переменные), если командой **File|Open** открыть из командного окна соответствующий **fig**-файл.

Не убирая с экрана дисплея первое графическое окно, вводим с клавиатуры следующие выражения:

z =A*cos(2*pi*F*t+alfa);
plot(t, z)

и получаем график функции **z(t)** в том же окне. Если нужно второй график провести «поверх первого графика», то перед вторичным вызовом графической функции **plot** нужно выполнить команду **hold on**.

Можно построить сразу несколько графиков, используя одни и те же оси координат (в этом случае графики приобретают разный цвет):

plot(t,y,t,z)

Если графики требуется разместить в разных окнах, то используется команда **figure**, которая создает новое окно.

y=A*sin(2*pi*F*t+alfa);
plot(t, y)
figure
z=A*cos(2*pi*F*t+alfa);
plot(t, z)

Функция **subplot** позволяет разбить область вывода графической информации на несколько подобластей, в каждую из которых можно вывести графики различных функций. Функция **subplot** принимает три числовых аргумента, первый из которых равен числу рядов подобластей, второй – числу колонок подобластей, а третий аргумент – номеру подобласти (номер отсчитывается вдоль рядов с переходом на новый ряд по достижению конца ряда).

Для ранее выполненных вычислений с функциями **sin, cos** строим

график функций **sin** в первой подобласти, а график функции **cos** – во второй подобласти одного и того же графического окна:

subplot(1,2,1); plot(t,y); subplot(1,2,2); plot(t,z)

Результат построения графиков изображен на рис. 2.1.

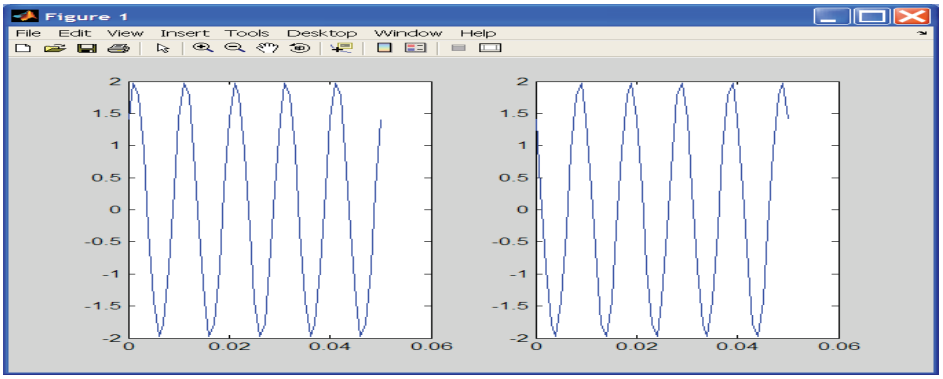


Рис. 2.1

Если для одиночного графика диапазоны изменения переменных вдоль одной или обеих осей координат слишком велики, то можно воспользоваться функциями построения графиков в логарифмическом масштабе. Для представления осей X, Y в логарифмическом масштабе по отдельности предназначены функции **semilogx, semilogy.** Функция **loglog** представляет обе оси X, Y в логарифмическом масштабе. Порядок вызова функций такой же, как для функции **plot**. Например,

semilogx(t,y)

Можно задать цвет и стиль линий, а также размещение различных надписей в пределах графического окна. Например, следующие команды:

x=0:0.1:3;
y = sin(x);
plot(x,y,'r-',x,y,'ko')

позволяют придать графику вид красной сплошной линии, на которой в дискретных вычисляемых точках проставляются черные окружности. Здесь функция **plot** дважды строит график одной и той же функции, но в двух разных стилях. Первый из этих стилей отмечен как **'r-'**, что означает проведение линии красным цветом (буква **r**), а короткая черта означает проведение сплошной линии. Второй стиль, помеченный как **'ko'**, означает проведение черным цветом (буква **k**) окружностей (буква **o)** на месте вычисляемых точек.

В общем случае функция **plot(x1,y1,s1, x2,y2,s2,...)** позволяет объединить в одном графическом окне несколько графиков функций **y1(x1), y2(x2),...,** проведя их со стилями **s1, s2,** ... . Стили **s1, s2**,... задаются в виде набора трех символьных маркеров, заключенных в одиночные кавычки (апострофы). Один из этих маркеров задает тип линии («–» – непрерывная, «--» – штриховая, «**:**» – пунктир), другой – задает цвет (**r** – красный, **g** – зеленый, **b** – синий, **k** – черный), третий – задает тип точек (**.** точка, + плюс, * звездочка, **o** – кружок, **x** – крестик).

Если указаны не все три маркера, то используются маркеры, установленные “по умолчанию”. Порядок, в котором указываются маркеры, не является существенным, то есть ‘**r**+-’ и ‘+**r**’ приводят к одинаковому результату. Если в строке стиля поставить маркер типа точки, но не проставить маркер на тип линии, то тогда отображаются только вычисляемые точки, а непрерывной линией они не соединяются. Оформить графики, вставляя текст, меняя цвет, шрифт и т.п., можно также, используя инструменты графического окна.

Если требуется указать на графике лишь точки – результаты вычислений, то удобно пользоваться командой **stem(x,y)**, с помощью которой строится «стебельковый» график. Пример (рис. 2.2):

```
A=2;
F=50;
alfa=pi/4;
Fd=1000;
t=0:1/Fd:0.05;
y=A*sin(2*pi*F*t+alfa);
stem(t, y)
```

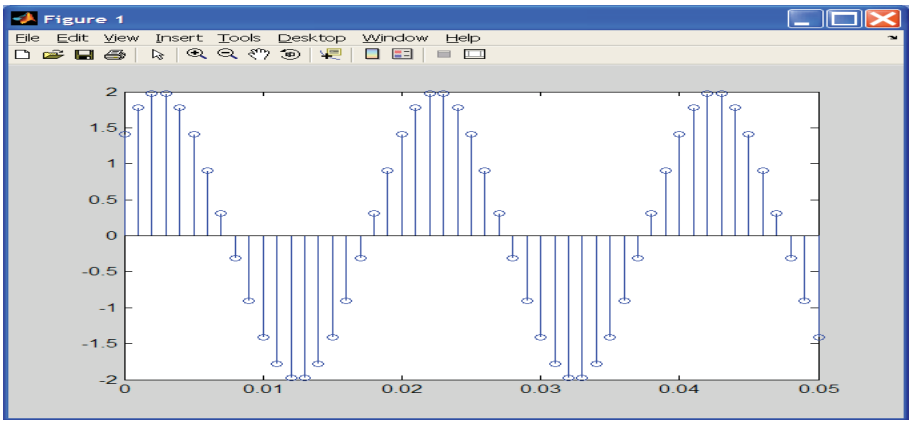


Рис. 2.2

Трехмерная графика

Каждая точка в пространстве характеризуется тремя координата-

ми. Набор точек, принадлежащих некоторой линии в пространстве, следует задать в виде трех векторов, каждый из которых задает одну из трех координат этих точек. Эти три вектора подаются на вход функции **plot3**, которая проектирует соответствующую трехмерную линию на плоскость и строит результирующее изображение. Пример построения винтовой линии (рис. 2.3):

```
t=0:pi/50:10*pi;
x=sin(t);
y=cos(t);
plot3(x,y,t);
grid on
```

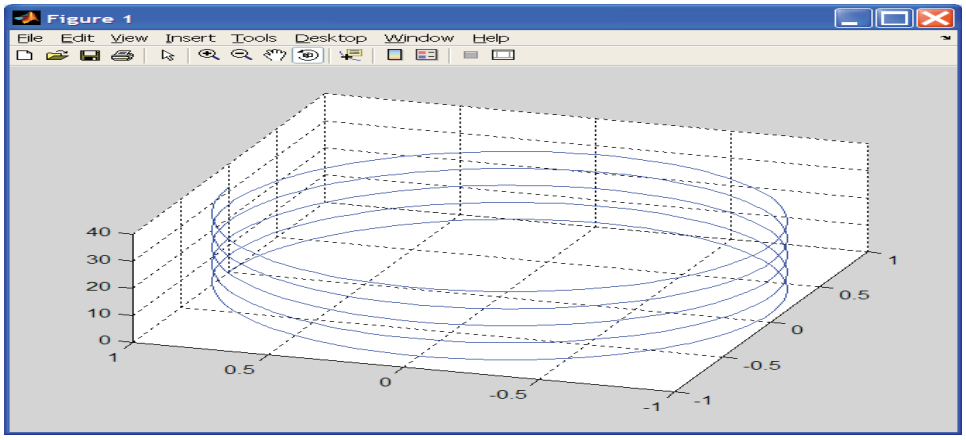


Рис. 2.3

Трехмерный график можно повернуть в удобное для просмотра положение, используя инструмент «вращение» в графическом окне.

Функцию **plot3** можно применить и для изображения поверхностей в пространстве, если провести семейство линий пересечения поверхности с набором плоскостей, которые параллельны одной из координатных плоскостей. В этом случае на вход функции **plot3** подаются не три одномерных массива, как было рассмотрено выше, а три матрицы одинакового размера.

Графики функций двух переменных представляют собой куски поверхностей, «нависающие» над областями определения функций. Пусть в точке с координатами xl, yl вычислено значение функции z=f(x,y), и оно равно zl. В некоторой другой точке (x2, y2) вычисляют значение функции z2. Продолжая этот процесс, получают массив точек (xl,yl,zl), (x2,y2,z2), … (xN,yN,zN), расположенных в трехмерном пространстве. Специальные функции системы MATLAB проводят через эти точки

гладкие поверхности и отображают их проекции на плоский дисплей компьютера.

Чаще всего точки аргументов расположены в области определения функции регулярно в виде прямоугольной сетки (матрицы точек). Такая сетка точек порождает две числовые матрицы одной и той же структуры: первая матрица содержит значения первых координат этих точек (x-координат), а вторая – значения вторых координат (y-координат). Обозначим первую матрицу как **X**, а вторую – как **Y**, а матрицу значений функции **z=f(x,y)** при этих аргументах, – буквой **Z**.

В системе MATLAB имеется специальная функция **meshgrid** для получения двумерных массивов **X** и **Y** по одномерным массивам **x, y**. Рассмотрим пример построения графика функции двух переменных $z=-x^2-y^2$ (рис. 2.4).

```
x=-2:0.1:2;
y=-1:0.1:1;
[X,Y]=meshgrid(x,y);
Z=-X.^2-Y.^2;
plot3(X,Y,Z);
grid on
```

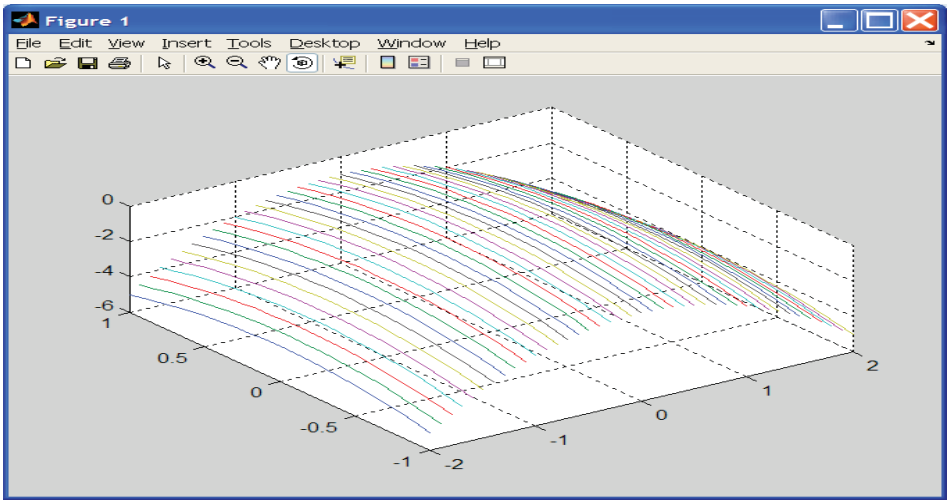


Рис. 2.4

Кроме функции **plot3**, имеется ряд функций, например, **mesh(X,Y,Z),.surf(X,Y,Z),** обеспечивающих большую наглядность трехмерных графиков. Функция **mesh** соединяет друг с другом все соседние точки поверхности графика отрезками прямых и показывает в графическом окне плоскую проекцию объемного тела.

Каркасно-ребристое тело состоит из четырехугольных граней белого цвета, а ребра граней окрашиваются в разные цвета. По умолчанию более высоким точкам графика соответствуют красные цвета, а более низким (меньшие значения третьей координаты) – темно-синие. Промежуточные области окрашиваются в светло-синие, зеленые и желтые цвета. Функция **surf** окрашивает грани в разные цвета. Чтобы ребристое тело стало прозрачным, используется команда

hidden off

Для возврата к прежнему отображению используется команда

hidden on

Задание 2

1. Построить трехмерные графики функции $z = -x^2 - y^2$, используя функции **mesh(X,Y,Z),.surf(X,Y,Z)**.

2. Для заданных значений a, b, c решить уравнение $ax^2+bx+c=0$ графическим способом путем построения графика (с размерной сеткой) функции $y=ax^2+bx+c$. Сравнить полученный результат с точным решением, полученным аналитически. Сохранить графическое окно в **fig**-файле.

3. Для заданных значений амплитуды **A**, частоты **F** колебаний, начальной фазы **alfa**, частоты дискретизации **Fd**, интервала времени от нуля до **T** построить график дискретизированной во времени гармонической функции. График представить сплошной красной линией с нанесенными на нее черными отметками, соответствующими точкам отсчета функции. Сохранить графическое окно в **fig**-файле.

4. Создать в одном окне несколько графических подокон в соответствии с вариантом. Построить в окне с номером (указан в скобках) согласно варианту график из предыдущего пункта. Сохранить графическое окно в **fig**-файле.

5. Создать звуковой файл длительностью 0,5 с для дискретизированного гармонического колебания с частотой NкГц (N – номер варианта) и амплитудой 3 В. Частота дискретизации 8 кГц.

6. Сформировать звуковой файл из фрагмента звукового файла, созданного в предыдущем пункте задания. Начало фрагмента совпадает с началом файла, а окончание определяется вариантом задания.

7. Построить таблицу изученных команд и функций с пояснениями по их использованию.

Содержание отчета

4. Распечатка истории команд.
5. Таблица с описанием изученных команд и функций.
6. Результаты выполнения примеров и заданных вычислений.

Контрольные вопросы

1. В чем заключается различие между функциями **sound** и **soundsc?**
2. Когда целесообразно использовать функцию **stem?**
3. Укажите на ошибку в следующей записи: **subplot(2,1,3)**.

Лабораторная работа 3
ПРОГРАММИРОВАНИЕ В СИСТЕМЕ MATLAB

Цель работы. Изучение основных подходов к программированию в системе MATLAB.

Задание 1. Изучить материал, изложенный ниже, и повторить примеры и упражнения, приведенные в тексте.

Основные сведения о программировании в системе MATLAB

Операторы цикла и управления

В М-языке системы MATLAB используется два вида операторов цикла. Первый из них имеет следующий вид: **while** выражение • • • **end.** Повторение участка кода, обозначенного многоточием, продолжается до тех пор, пока **выражение** «истинно» (не равно нулю). Пример:

```
S = 0; k=1; u=1;
while u >1e-8
 S= S + u;
 k=k+1; u=1/k^2;
end
```

Здесь условием останова служит требование к слагаемым быть больше некоторого заранее определенного числа 10^{-8}. Как только очередное слагаемое станет меньше этого числа, выражение, стоящее после ключевого слова **while**, станет ложным, и суммирование прекратится.

После ввода ключевого слова **while** и нажатия клавиши **Enter** текстовый курсор перемещается на следующую физическую строку, но

знак » не появляется. Это означает, что система MATLAB находится в режиме ожидания. Только после ввода ключевого слова **end** система MATLAB приступает к циклически повторяющемуся выполнению инструкций из тела цикла.

Другой вид оператора цикла имеет вид:

for varName=выражение...end, где **varName** – имя переменной цикла. Обозначенное многоточием тело цикла выполняется для всех возможных значений переменной цикла. Набор возможных значений для переменной цикла определяется выражением, которое стоит после ключевого слова **for**. Пример:

```
S=0;
for k=1:1:57
    S = S+1/k^2;
end
```

Вместо операции задания диапазона значений переменной цикла можно явно указать весь набор возможных значений в виде вектор-строки, например, **for m = [2, 5, 7, 8, 11, 23].**

К операторам управления в М-языке относятся условный оператор и оператор переключения. Условный оператор использует ключевые слова: **if**, **else, elseif** (иначе, если), **end** и может применяться в трех вариантах:

```
if условие… end
if условие … else…end
if условие1…elseif условие2…else… end
```

В последнем случае ветвей с ключевым словом **elseif** может быть несколько. Область действия условного оператора начинается ключевым словом **if**, а заканчивается ключевым словом **end**. Под условием понимается произвольное выражение, истинность или ложность которого понимается как отличие или равенство нулю.

Если условие истинно, то выполняются команды, стоящие после строки с ключевым словом **if**. Если условие ложно, то эти команды пропускаются, и осуществляется переход либо к оператору, следующему за условным (первый вариант), либо выполняются без дополнительных проверок команды, стоящие после строки с ключевым словом **else,** либо проверяют еще одно условие в строке с ключевым словом **elseif**. Пример:

```
A=[2 4 6 7;3 5 8 11;4 8 1 3];
A1=A;
I=2;
J=2;
if I > J
A(I,J) = 2;
elseif abs(I-J) == 1
A(I,J) = -1;
else
A(I,J) = 0;
end
```

После выполнения примера элемент **A(2,2)** массива **A** изменит свое значение. Это изменение удобно проследить, сравнивая массивы A и A1.

В условных выражениях, входящих в условные операторы, можно использовать массивы. Если хотя бы один элемент такого массива равен нулю, то условие считается ложным. Пример:

```
A =[12; 40];
if A
  b = 1;
else
  b =2;
```

В данном примере переменная **b** принимает значение 2, так как матрица **A** содержит один нулевой элемент, и все условие считается ложным.

Оператор переключения использует ключевые слова **switch** (переключить), **case** (случай), **otherwise** (иначе) и имеет следующий вид:

```
switch выражение
case значение1
…
case значение2
…
otherwise
…
end
```

Сначала вычисляется выражение, определяющее скалярное числовое значение, а затем полученный результат сравнивается с набором значений **значение1, значение2, …**. В случае совпадения с одним из

значений выполняется нижестоящая группа операторов. Если нет совпадения ни с одним из перечисленных значений, то выполняются операторы, стоящие после ключевого слова **otherwise**. Строк с ключевым словом **case** может быть несколько, а строка с ключевым словом **otherwise** – одна. Пример:

```
I=2;
J=2;
switch I+J
case 2,
a=10
case 4,
a=11
otherwise,
a=7
end
```

Сценарии и функции

Под сценарием понимают текстовый файл, содержащий инструкции на М-языке, подлежащие исполнению в автоматическом пакетном режиме. Файл имеет произвольное имя с расширением **m**. Любой текстовый файл, содержащий произвольный код на М-языке и имеющий оговоренное выше расширение, называется М-файлом. Создать такой файл можно в любом текстовом редакторе, но удобнее пользоваться редактором системы MATLAB, который вызывается командой меню **File | New | M-file**. М-файлы не следует путать с MAT-файлами, в которых хранятся переменные из рабочего пространства.

Переменные, определяемые в командном окне, и переменные, определяемые в сценариях, составляют единое рабочее пространство системы MATLAB. Сценарии позволяют ускорить работу пользователя за счет использования в текущем сеансе работы команд, которые использовались в предыдущих сеансах работы с системой. Однако его нельзя использовать, если обозначения переменных изменились, хотя решаемая задача, по сути, осталась прежней.

Путь к каталогу, где хранится М-файл, должен быть известен системе MATLAB. Пакет MATLAB хранит сведения обо всех известных ему каталогах. Для создания нового каталога нужно сначала выполнить команду меню (командного окна) **File | Set Path**, с помощью которой вызывается диалоговое окно с именем **Set Path.** В этом окне показы-

вается список всех зарегистрированных в системе MATLAB путей доступа. Для добавления нового каталога в список путей доступа служит кнопка **Add Folder**. После того как добавлен новый каталог, в нем можно сохранить файл с созданным в редакторе сценарием.

Чтобы использовать код, расположенный в М-файле, нужно ввести в командном окне имя М-файла, содержащего сценарий работы (указывать при этом расширение файла нельзя – возникнет ошибочная ситуация), и нажать клавишу **Enter**.

Чтобы реализовать независимый и изолированный фрагмент кода, решающий некоторую фиксированную задачу и принимающий в виде входных параметров начальную информацию, нужно оформить этот фрагмент в виде функции системы MATLAB.

Функции, как и сценарии, состоят из набора инструкций М-языка и их так же записывают в текстовые файлы с расширением **m**. Текст М-функции должен начинаться с заголовка, после которого следует тело функции. Заголовок имеет следующий общий вид:

function [Retl, Ret2,...] = FName(parl, par2,...).

С помощью ключевого слова **function** объявляется функция с именем **FName**, которая принимает входные параметры (аргументы функции) **parl, par2**,... и возвращает выходные значения **Retl, Ret2**… Указанное в заголовке имя функции должно совпадать с именем файла (без учета расширения .**m**), в который записывается текст функции. Если функция не содержит входных параметров или не возвращает выходных значений, то возможны сокращенные записи заголовка. Примеры:

function FNamel
function FName2(par1, par2, par3)
function Ret1 = FName3(par1, par2)

В первой строке представлен заголовок функции **FName1**, не имеющей ни входных параметров, ни выходных значений. Функция **FName2** принимает три входных параметра и не имеет возвращаемых значений. Функция **FName3** принимает два и возвращает одно значение.

За заголовком функции следует тело функции, которое состоит из инструкций М-языка, с помощью которых вычисляются возвращаемые значения. Заголовок и тело функции составляют определение функции. Размещать М-файл с определением функции нужно в одном из каталогов диска, входящих в список доступа пакета MATLAB.

Как входные параметры, так и возвращаемые значения могут быть

в общем случае массивами различных типов, размерностей и размеров. Пример:

```
function [A,B] = MatrProc(X1, X2, x)
%Демонстрация работы с матрицами и скалярной величиной
A = X1 .*X2*x;
B=X1 .*X2+x; %Последняя строка функции
```

Здесь входными параметрами являются два числовых массива **X1** и **X2** одинакового размера и одного скаляра **x**. Эти массивы в теле функции сначала перемножаются поэлементно, после чего результат такого перемножения дополнительно умножается на скаляр **x**. Таким образом, порождается первый из выходных массивов – массив **A**. Выходной массив **B** получается сложением матрицы со скаляром.

В тексте функции **MatrProc** помимо инструкций, вычисляющих возвращаемые функцией значения, присутствуют также комментарии, начинающиеся со знака %. Особую роль играют строки комментариев, располагающиеся сразу же за заголовком функции. Эти комментарии предназначены не только для программистов, но и для пользователей, желающих ознакомиться с краткой информацией по функции. В этих строках должна быть расположена краткая справка по функции. Справку можно получить, используя команду

```
help MatrProc
```

Используя команду type, можно ознакомиться с текстом всей функции

```
type MatrProc
```

Вызов функции осуществляется из командного окна системы MATLAB или из текста какой-либо другой функции. Синтаксис вызова функции следующий: записывается имя функции, после которого в круглых скобках через запятую перечисляются фактические входные параметры, со значениями которых производятся вычисления. Фактические параметры могут быть заданы числовыми значениями, именами переменных (уже имеющими конкретные значения), а также выражениями.

Если фактический параметр задан именем переменной, то реальные вычисления производятся не с ней самой, а с ее копией – имеет место передача параметров по значению. Пример:

```
W1=[1,2;2,2]; W2=[3,1;1,1];
[Res1,Res2]=MatrProc(W1,W2, 3);
```

Здесь имена фактических входных параметров (**W1** и **W2**) и переменных, в которых записываются результаты вычислений (**Res1** и **Res2**), не совпадают с именами аналогичных переменных в определении функции **MatrProc**. Чтобы подчеркнуть это возможное отличие, имена входных параметров и выходных значений в определении функции называют формальными. Результат выполнения примера:

```
Resl =
     9     6
     6     6
Res2 =
     6     5
     5     5
```

Функцию **MatrProc** можно использовать в составе выражений так же, как это делается с функциями, возвращающими единственное значение. В этом случае в качестве значения функции, применяемого для дальнейших вычислений, используется первое из возвращаемых функцией значений. Пример:

```
S= MatrProc(1, 2, 1)+6;
```

При вызове с параметрами 1,2,1 функция **MatrProc** возвращает два значения: 2 и 3. Для вычисления всего выражения используется первое из них, и переменная **S** становится равной 8.

В теле М-функций часто используются условные операторы совместно с инструкцией **return** для осуществления досрочного завершения функции и выхода из нее. Пример:

```
function res = MyFact(n)
   res= 1;
   if n == 1
   return
   else
    for i = 2:n
   res = res* i;
   end
end
```

Функция вычисляет факториал **n (n!)**. Если ее входной параметр равен единице, то с помощью оператора **return** осуществляется досрочный выход из функции, так как правильный результат (**res=1**;) уже сформирован. Если оператор **return** не используется, то завершение работы функции наступает после выполнения всех ее инструкций.

Если в М-файл поместить определения сразу нескольких функций, то вызывать из командного окна системы MATLAB (или из функций другого М-файла) можно будет только ту из них, имя которой совпадает с именем М-файла. Остальные функции вызываются изнутри данного М-файла. Пример:

```
function ret1=ManyFunc(xl,x2)
ret1= x1.*x2+AnotherFunc(x1);
function ret2=AnotherFunc(y)
ret2=y.*y+2*y+3;
```

Здесь определены функции **ManyFunc** и **AnotherFunc**. Однако извне можно вызывать только функцию **ManyFunc**. Функцию **AnotherFunc** может вызывать только функция **ManyFunc**.

Отладка М-функций

При возникновении ошибки в процессе выполнения М-функции в командное окно выводятся приблизительное диагностическое сообщение и номер строки, в которой произошла ошибка. Более надежным способом отладки функции являются применение точек останова **(Breakpoints**) и пошаговое выполнение тела функции. Для этого применяют встроенные возможности редактора – отладчика системы MATLAB.

Чтобы поставить точку останова на какой-либо строке кода функции, туда нужно поместить курсор и нажать клавишу **F12** (повторное нажатие этой клавиши убирает точку останова) или выполнить команду меню **Debug | Set/Clear Breakpoint.** Не закрывая окна редактора, переключаем фокус ввода с клавиатуры в командное окно системы MATLAB и запускаем обычным образом функцию на выполнение. Останов выполнения функции произойдет на строке, в которой поставлена точка останова.

Теперь можно просматривать фактические значения входных параметров функции, текущие значения переменных, а также значения выражений. Чтобы просмотреть значение переменной, достаточно подвести курсор к ее имени в тексте функции, после чего на экране появится

информация о значении переменной. Нажимая клавишу **F10**, можно выполнять функцию построчно, каждый раз проверяя результаты работы функции. Таким способом можно выявить и устранить причину ошибки.

Задание 2

1. Создать М-функцию, определяющую кратковременную среднюю энергию сегмента (фрагмента) звукового файла длительностью *N* отсчетов сигнала. Средняя энергия сегмента определяется по формуле

$$E_k = \frac{1}{N}\sum_{n=1}^{N} x_k{}^2(n),$$

где *k, N, x(n)* – соответственно номер, длительность сегмента (количество отсчетов сигнала) и значение *n*-го отсчета сигнала внутри сегмента. Имя файла, длительность *N* сегмента и номер *B* отсчета сигнала, с которого начинается сегмент, являются параметрами функции.

2. Создать М-функцию, определяющую кратковременную среднюю энергию звукового файла. Весь файл разбивается на отдельные сегменты длительностью *N* отсчетов сигнала. Каждый сегмент перекрывается с соседним сегментом. Для каждого сегмента определяется средняя энергия по формуле, приведенной выше. Имя файла, длительность *N* сегмента и перекрытие *L* сегментов являются параметрами функции. Следует учесть, что в конце файла длительность сегмента меньше *N*. Построить график *E(k)* для $N = 200$, $L = 0{,}5\ N$. Снабдить функцию комментариями.

3. Для *k*-го сегмента звукового файла создать функцию, определяющую автокорреляционную функцию сегмента

$$R_k(p) = \frac{1}{N}\sum_{n=1}^{N} x_k(n)x_k(n+p).$$

Построить график функции *R(p)* для $p = 1{:}4$. Остальные данные взять из предыдущего пункта задания.

4. Построить таблицу изученных команд и функций с пояснениями по их использованию.

Содержание отчета

1. Распечатка истории команд.
2. Таблица с описанием изученных команд и функций.
3. Результаты выполнения примеров и заданных вычислений.
4. Распечатка текста М-файла и график зависимости средней энергии звукового файла от времени.

Контрольные вопросы

1. Ответить на вопросы преподавателя по использованию изученных команд и функций.

2. Когда целесообразно использовать файл сценария, а когда – файл функции?

3. Что надо сделать, чтобы система MATLAB «увидела» файл функции, расположенный в какой-либо папке?

Лабораторная работа 4

ФОРМИРОВАНИЕ И АНАЛИЗ СИГНАЛОВ В СИСТЕМЕ MATLAB

Цель работы. Изучение возможностей MATLAB для формирования анализа сигналов**.**

Задание 1. Изучить материал, изложенный ниже, и повторить примеры и упражнения, приведенные в тексте.

Формирование сигналов

Формирование детерминированных сигналов

Для формирования дискретизированного сигнала, заданного каким-либо выражением, необходимо сначала сформировать вектор дискретных значений времени. Для этого удобно задать значение частоты дискретизации **Fs** (sampling frequency) и использовать обратную величину в качестве шага временного ряда. Пример формирования гармонических колебаний:

```
Fs = 8e3;                 % частота дискретизации 8 кГц
t = 0:1/Fs:0.025;         % 0.025 с дискретных значений времени
A = 2;                    % амплитуда – два вольта
f0 = 1e3;                 % частота 1 кГц
phi = pi/4;               % начальная фаза 45°
s1=A*cos(2*pi*f0*t+phi);  % гармоническое колебание
alpha1 = 0.1e3;           % скорость затухания
s2 = exp(-alpha1*t) .* s1; % затухающее гармоническое колебание
subplot(2,1,1);plot(t,s1); subplot(2,1,2);plot(t,s2);
```

На рис. 4.1, приведенном ниже, показаны «осциллограммы» сформированных колебаний.

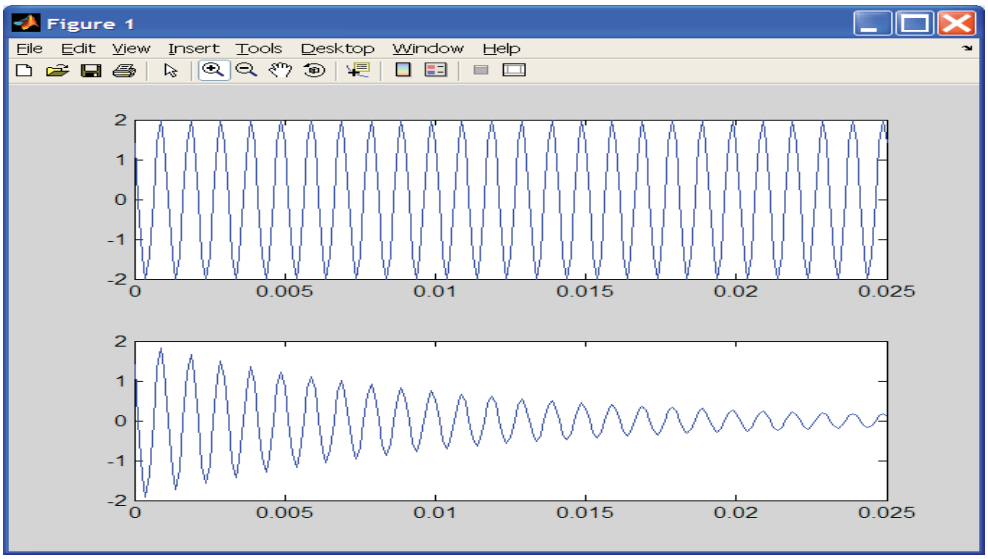


Рис. 4.1

При формировании импульсных сигналов целесообразно использовать операции сравнения, которые возвращают единицу при выполнении неравенства и ноль в противном случае, причем в случае векторного аргумента возвращается вектор результатов сравнения. Сформируем односторонний экспоненциальный импульс (рис. 4.2):

```
A = 2;
Fs = 8e3;
t = -0.005:1/Fs:0.01;
alpha = 0.5e3;
s = A * exp(-alpha * t) .* (t >= 0);
stem(t,s);
```

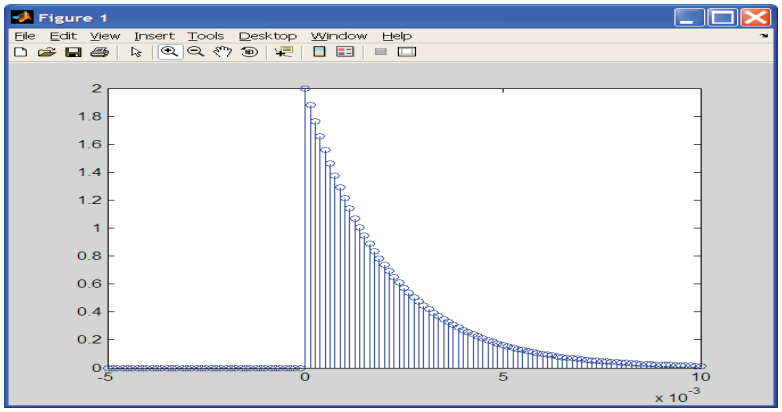


Рис. 4.2

Сформируем прямоугольный импульс, центрированный относительно начала отсчета времени (рис. 4.3):

```
A = 2;
Fs = 8e3;
T= 0.01;
t = -0.01:1/Fs:0.01;
s = A * (abs(t) <= T/2);
stem(t,s)
```

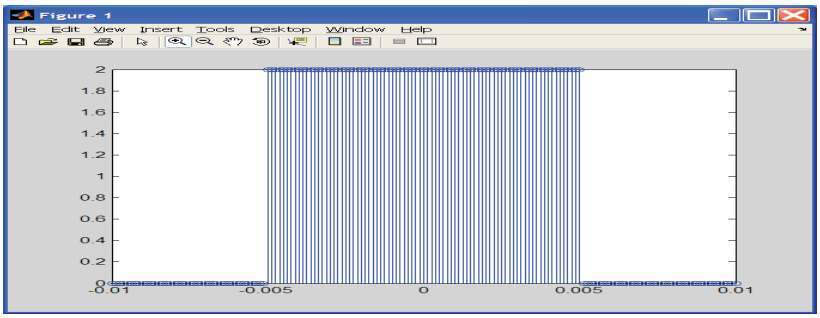


Рис. 4.3

Пример формирования несимметричного треугольного импульса (рис. 4.4):

```
A = 2;
Fs = 8e3;
T= 0.01;
t = -0.01:1/Fs:0.02;
s = A * t / T .* (t >= 0) .* (t <= T);
stem(t,s)
```

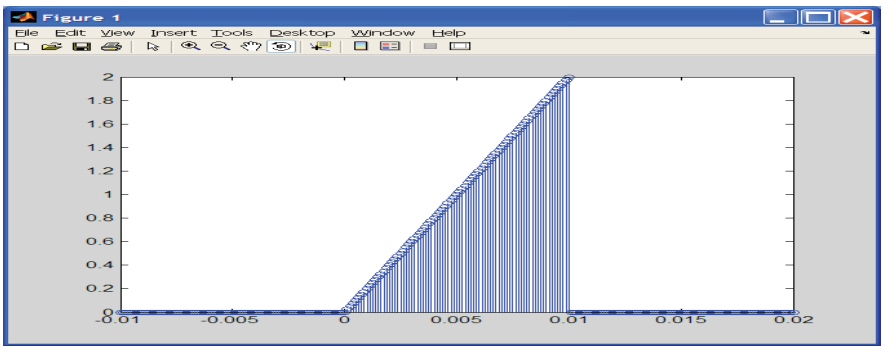


Рис. 4.4

В приведенных примерах следует обратить внимание на использование оператора **.*** для поэлементного перемножения векторов.

В составе системы MATLAB имеется ряд пакетов (**Toolboxes**), которые дополняют ядро системы другими функциями. В пакете **Signal Processing** имеется ряд функций, генерирующих часто встречающиеся на практике импульсные сигналы. В частности, имеются функции **rectpuls (**формиро-

вание одиночного прямоугольного импульса), **tripuls (**формирование одиночного треугольного импульса) и **square** (формирование периодической последовательности прямоугольных импульсов). Функция **rectpuls** формирует одиночный прямоугольный импульс с единичной амплитудой: **y = rectpuls(t, width)**. Здесь **t** – вектор значений времени, **width** – ширина (длительность) импульса. Возвращаемый результат **y** – вектор рассчитанных значений сигнала, определяемых по следующей формуле:

$$y = \begin{cases} 1, & -\frac{\text{width}}{2} \le t < \frac{\text{width}}{2}, \\ 0, & t < -\frac{\text{width}}{2}, \quad t \ge \frac{\text{width}}{2}. \end{cases}$$

Сформируем пару разнополярных прямоугольных импульсов, расположенных справа и слева от начала отсчета времени (рис. 4.5).

```
A = 2;
Fs = 8e3;
T= 0.01;
t = -0.02:1/Fs:0.02;
s = - A* rectpuls(t+T/2, T) + A * rectpuls(t-T/2, T);
stem(t, s);
ylim([-3 3])
grid on
```

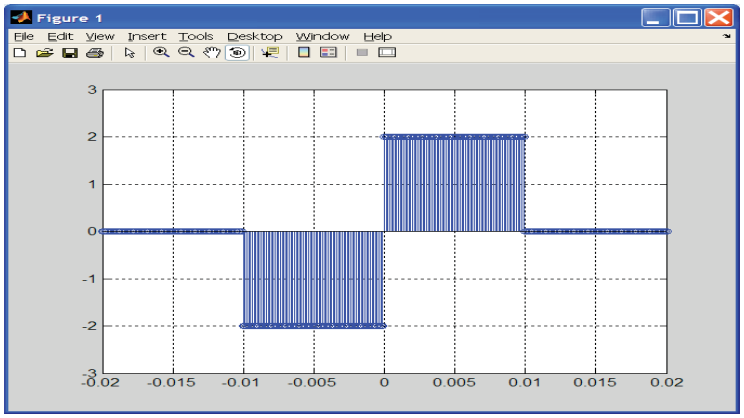


Рис. 4.5

Построим с помощью функции **tripuls** пару разнополярных треугольных импульсов (рис. 4.6).

```
s = - A* tripuls(t+T/2, T) + A * tripuls(t-T/2, T);
stem(t, s);
ylim([-3 3])
grid on
```

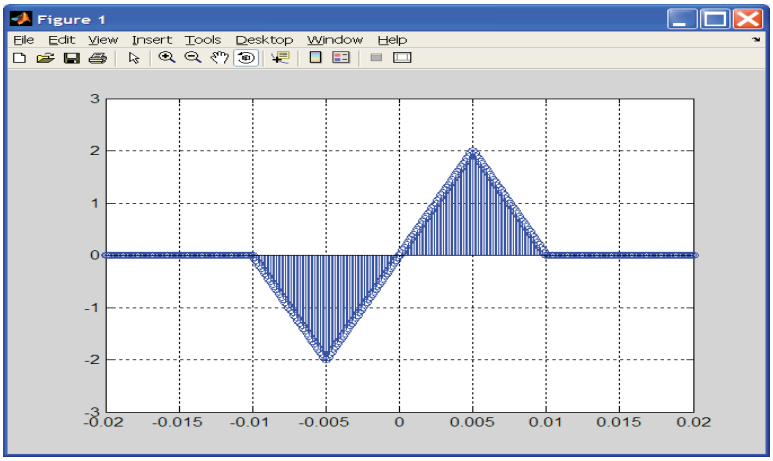


Рис. 4.6

Для формирования последовательности прямоугольных импульсов с периодом **T** служит функция **square: y=square(2*pi*t/T,duty)**. Параметр **duty** – это отношение длительности импульса к периоду (в процентах). По умолчанию значение параметра равно 50, то есть генерируется меандр, в данном случае функция аналогична функции **sin(x)**, но вместо синусоиды формируется импульсная последовательность. Сформируем последовательность однополярных прямоугольных импульсов (рис. 4.7):

```
A = 2;                  % амплитуда
Fs = 8e3;               % частота дискретизации
t = -0.02:1/Fs:0.02;    % дискретное время
f0 = 100;               % частота следования импульсов
tau = 2.5e-3;           % длительность импульсов
s2 =0.5*A*(square(2*pi*t*f0, f0*tau*100)+1);
stem(t, s2);
ylim([0 2.5]);
grid on
```

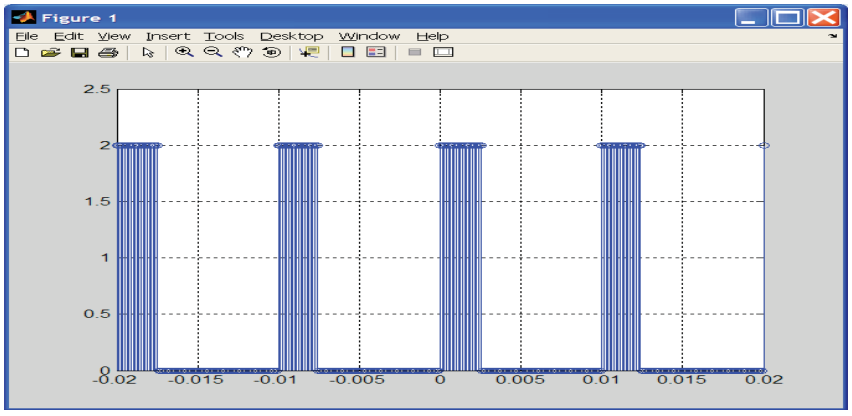


Рис. 4.7

Необходимо отметить, что частоту дискретизации следует выбирать так, чтобы количество отсчетов на интервале длительности импульса было достаточно большим, так как из-за погрешности вычислений число отсчетов на этом интервале может отличаться от требуемого на единицу.

Формирование случайных сигналов

Для генерации случайных чисел служат функции **rand(m, n)** (равномерное распределение на интервале от нуля до единицы) и **randn(m, n)** (нормальное распределение с нулевым математическим ожиданием и единичным среднеквадратическим отклонением). Здесь **m, n** – число строк и столбцов матрицы случайных чисел. Если функции заданы в виде: **rand(size(A)), randn(size(A))**, где **A** – массив, то генерируется массив случайных чисел с размерами, равными размерам массива **A**. Для генерации случайных чисел предназначены также средства пакета расширения **Statistics**, которые поддерживают множество различных законов распределения вероятностей.

Для оценки вида закона распределения вероятностей значений случайного процесса используется гистограмма. Вызов соответствующей функции имеет следующий синтаксис **N = hist(Y, M)**. Здесь **Y** – вектор сигнала; **M** – число столбцов гистограммы (по умолчанию M =10). **N** – вектор, который показывает, сколько элементов содержится в каждом столбце гистограммы. Если выходной параметр функции **hist** не указывается, то строится гистограмма. Построим гистограмму значений случайного процесса с равномерным распределением вероятностей (рис. 4.8):

```
Fs = 8e3;
t = 0:1/Fs:3;
x5=rand(size(t)); % дискрет. белый шум с равномер. распределением
hist(x5,100);
```

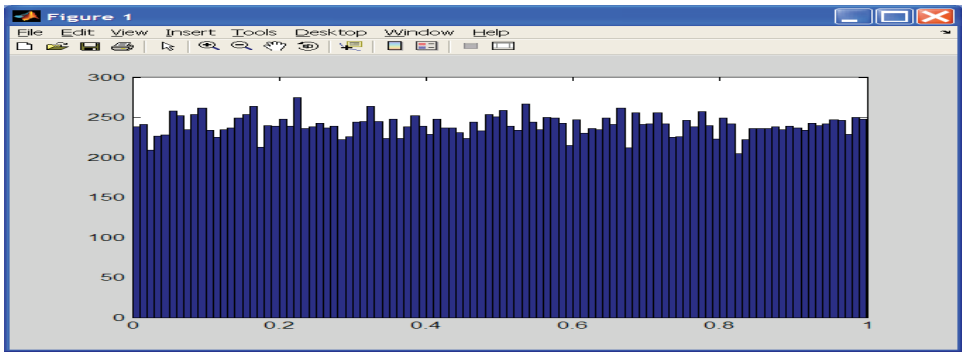


Рис. 4.8

Аналогично строим гистограмму для случайного процесса с нормальным распределением (рис. 4.9):

```
Fs = 8e3;
t = 0:1/Fs:3;
x6 = randn(size(t)); % дискрет. белый шум с норм. распределением
hist(x6,100);
```

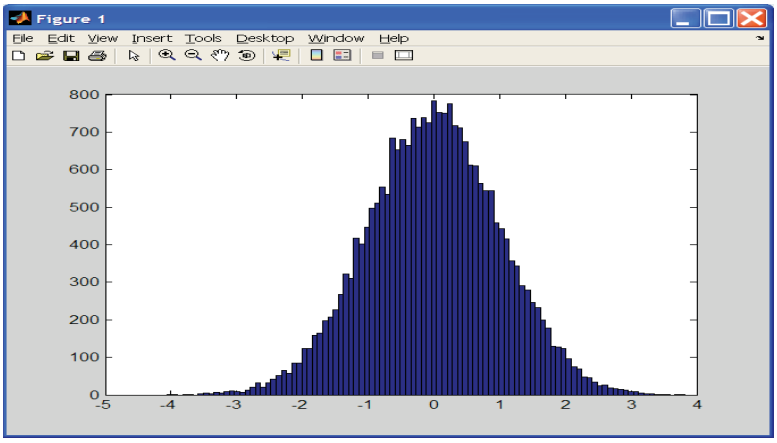


Рис. 4.9

В задачах, связанных с обработкой сигналов, для генерации дискретного нормального белого шума удобнее использовать функцию **wgn** (white Gaussian noise) пакета **Communications**, поскольку она позволяет в явном виде задавать уровень генерируемого шума. Синтаксис вызова функции следующий: **y=wgn(m,n,p,imp,state,'powertype', 'outputtype');**

Здесь **m** и **n** – как и ранее, размеры генерируемой матрицы, а **p** – мощность генерируемого шума в единицах, задаваемых параметром **'powertype'** (по умолчанию – в децибелах). Остальные параметры являются необязательными и имеют значения по умолчанию. Параметр **imp** задает импеданс нагрузки в омах (предполагается, что генерируются отсчеты случайного ***напряжения*** на этой нагрузке). По умолчанию используется импеданс нагрузки, равный 1 Ом.

Целочисленный параметр **state** позволяет принудительно задавать начальное состояние генератора гауссовских случайных чисел (функция ***randn***). По умолчанию используется текущее состояние.

Строковый параметр **'powertype'** задает единицы измерения мощности, использованные при указании параметра **p**. Возможны следующие значения:

– **'dBW'** – мощность **p** задается в децибелах, значению 0 дБ соответствует мощность 1 Вт;

– **' dBm'** – мощность p задается в децибелах, значению 0 дБ соответствует мощность, равная 10^{-3} Вт;

– **'linear'** – мощность **p** задается в ваттах, дисперсия генерируемого шума равна **p*imp**.

Строковый параметр **'outputtype'** позволяет задавать генерацию вещественного или комплексного шума. Возможны значения **'real'** (вещественный шум; генерируется по умолчанию) и **'complex'** (комплексный шум). Если генерируется комплексный шум, его вещественная и мнимая части имеют мощности p/2.

Пример генерации матрицы 1x **length(t)** вещественного шума мощностью 3 Вт на нагрузке 75 Ом:

```
Fs = 8e3;
t = 0:1/Fs:3;
y = wgn(1, length(t), 3, 75, ‘linear’, ‘real’);
plot(t,y)
figure
hist(y,100);
```

Спектральный анализ

Прямое и обратное дискретное преобразование Фурье (ДПФ)

ДПФ является основой спектрального анализа сигналов. Инструкция **y = fft(x)** – вычисляет прямое ДПФ для вектора **x**; если **x** – матрица, преобразование производится для каждого ее столбца по отдельности; **y=fft(x, N)** – предварительно приводит исходные данные к размеру **N**, урезая их или дополняя нулями. Инструкции **x = ifft(y)** и **x = = ifft(y, N)** – аналогичные варианты вызова для функции обратного ДПФ.

Для ускорения вычислений используется быстрое преобразование Фурье (БПФ). Пример определения БПФ для гармонического колебания (рис. 4.10):

```
Fs = 8e3;
f0=100;
t = 0:1/Fs:0.01;
x = sqrt(2)*sin(2*pi*f0*t);
F=fft(x);
stem(abs(F))
```

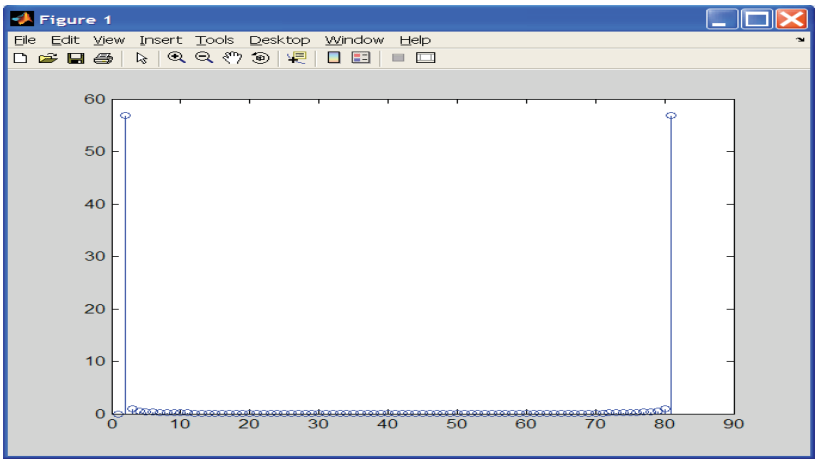


Рис. 4.10

При спектральном анализе рассматривается фрагмент (сегмент) сигнала на некотором интервале времени. Для выделения сегмента из всего сигнала используются весовые функции (временные окна). Использование весовых функций повышает точность спектрального анализа. Система MATLAB содержит (в пакете **Signal Processing**) целый ряд стандартных весовых функций.

Все весовые функции принимают в качестве параметра требуемую длину вектора (**n**), которая должна быть целым положительным числом, и возвращают вектор-столбец **w**. Рассмотрим для примера две функции.

Функция **boxcar**, реализующая «прямоугольное окно», введена в MATLAB лишь для полноты набора весовых функций, так как она соответствует отсутствию операции взвешивания: **w=boxcar(n)**. Возвращаемый вектор заполнен единицами: **w = ones(n,1)**.

Функция **hamming** реализует окно Хэмминга: **w = hamming (n,'sflag')**.

Строковый параметр **'sflag'** позволяет выбрать режим расчета окна. При значении **'symmetric'**, принятом по умолчанию, генерируется симметричное окно, для которого **w(k)=w(n +1 – k)**. Оно используется при расчете фильтров. При значении **'periodic'** создается слегка несимметричное окно, которое используется при спектральном анализе.

Спектрограмма

Если спектр сигнала меняется во времени, то для оценки спектра целесообразно использовать спектрограмму сигнала. Спектрограммой (spectrogram) сигнала называется *его мгновенный спектр,* зависящий от времени. Для вычисления спектрограммы вектор сигнала разбивается на сегменты (в общем случае с перекрытием). Для каждого сегмента вычисляется спектр с помощью функции **fft**. Набор спектров всех сегментов и образует спектрограмму. Для вычисления спектрограммы служит функция **spectrogram**.

Синтаксис вызова функции: **[S,F,T]=spectrogram (x,window, noverlap, nfft,Fs)**, где **x** – вектор сигнала; **window** – вектор весовой функции (если вместо вектора используется целое число, то используется весовая функция по умолчанию – функция Хэмминга соответствующей длины); **noverlap** – величина перекрытия соседних сегментов сигнала; **nfft** – число точек преобразования Фурье; **Fs** – частота дискретизации. **S –** матрица, каждая колонка которой содержит (**nfft/2+1**) отсчетов спектра для данного момента времени (если **nfft –** нечетное число, количество отсчетов равно (**nfft+1)/2**). Число колонок **k=fix((nx-noverlap)/(length(window)-noverlap))**, где **nx** – длина вектора сигнала. Параметр **F** – вектор частот, **T** – вектор моментов времени, его длина равна **k**.

Если выходные параметры функции не указываются (**spectrogram(x, window,noverlap, nfft, Fs)**), то строится трехмерный график спектральной плотности мощности в координатах: время, частота, уровень.

Обязательным входным параметром функции является вектор значений сигнала **x**, остальные параметры имеют значения по умолчанию, которые используются, если в качестве параметра указана пустая матрица ([]) или если несколько последних параметров при вызове опущены. Пример (рис. 4.11):

```
[V,fs,b]=wavread('9m.wav');
spectrogram(V,256,128,[],fs,'yaxis');
figure
spectrogram(V,256,128,1024,fs,'yaxis');
```

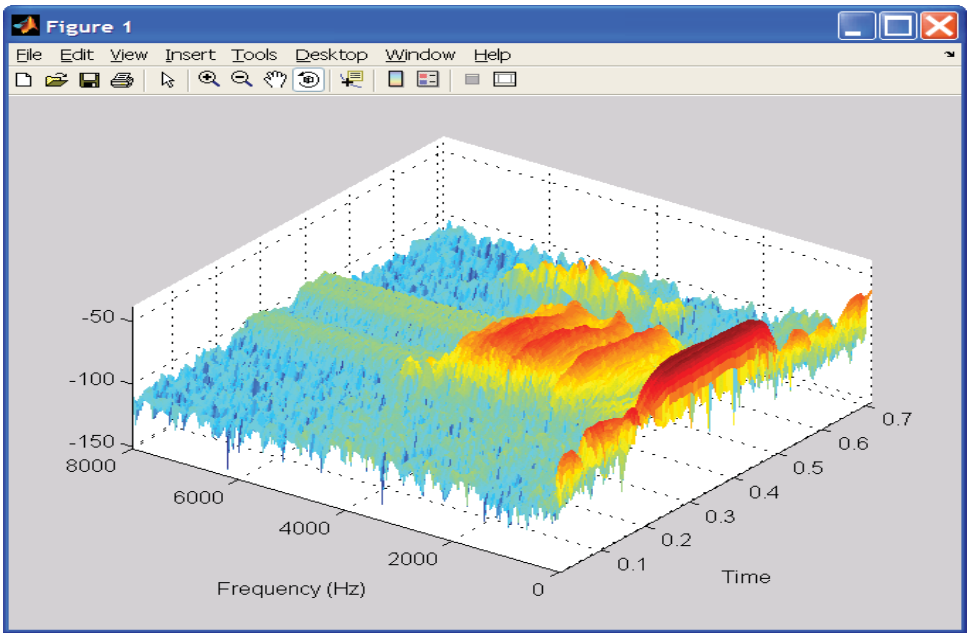


Рис. 4.11

Здесь во второй строке примера для параметра **nfft** используется

значение по умолчанию – максимальное из двух чисел: 256 и 2^k. Значение k таково, что выполняется условие $\mathbf{2^k > window.}$

Периодограмма

Периодограммой называется оценка спектральной мощности сигнала, полученная по N отсчетам одной реализации случайного процесса. Периодограмма используется для оценки спектра мощности стационарного случайного процесса. Для вычисления периодограммы предназначена функция **periodogram**. Синтаксис ее вызова следующий: **[Pxx, f] = periodogram(x, window, Nfft, Fs, 'range')**.

Обязательным входным параметром является **x** – вектор отсчетов сигнала. Остальные параметры имеют значения по умолчанию, которые используются, если в качестве параметра указана пустая матрица [] или если некоторое количество параметров (начиная с последнего) опущены при вызове.

Вектор **window** должен содержать коэффициенты используемого окна (при этом говорят о модифицированной периодограмме). По умолчанию используется прямоугольное окно.

Параметр **Nfft** задает размерность БПФ, используемого для вычисления периодограммы. По умолчанию этот параметр равен максимальному из двух чисел: 256 и 2^k. Значение k таково, что выполняется условие $\mathbf{2^k > length(x)}$. Входной сигнал, умноженный на окно, приводится к размеру **Nfft** (обрезается либо дополняется нулями).

Параметр **Fs** – частота дискретизации в герцах. Значение по умолчанию равно 2π. Строковый параметр **'range'** определяет частотный диапазон для возвращаемого вектора **Pxx**. Возможны два значения:

'twosided' – векторы **Pxx** и **f** имеют длину **Nfft** и соответствуют полному диапазону частот 0... **Fs**. Этот вариант используется по умолчанию, если **x** содержит комплексные отсчеты;

'onesided' – векторы **Pxx** и **f** имеют длину **ceil ((Nfft + 1)/2)** и соответствуют половинному диапазону частот 0... **Fs/2**. Этот вариант используется по умолчанию в случае вещественного вектора **x**.

Параметр **'range'** может быть указан в списке параметров в любом месте после **window**.

Возвращаемые параметры: **Pxx** – вектор значений спектральной плотности мощности, **f** – вектор значений частот, использованных для расчета. Шаг между соседними элементами этого вектора равен

Fs/Nfft, первый элемент равен нулю. Если выходные параметры при вызове не указаны, то функция строит график спектральной плотности мощности. Пример оценки спектральной плотности мощности зашумленного гармонического колебания (рис. 4.12):

```
Fs = 1000;
t = 0:1/Fs:3;
x = cos(2*pi*t*200)+randn(size(t));
periodogram(x,[],[],Fs);
```

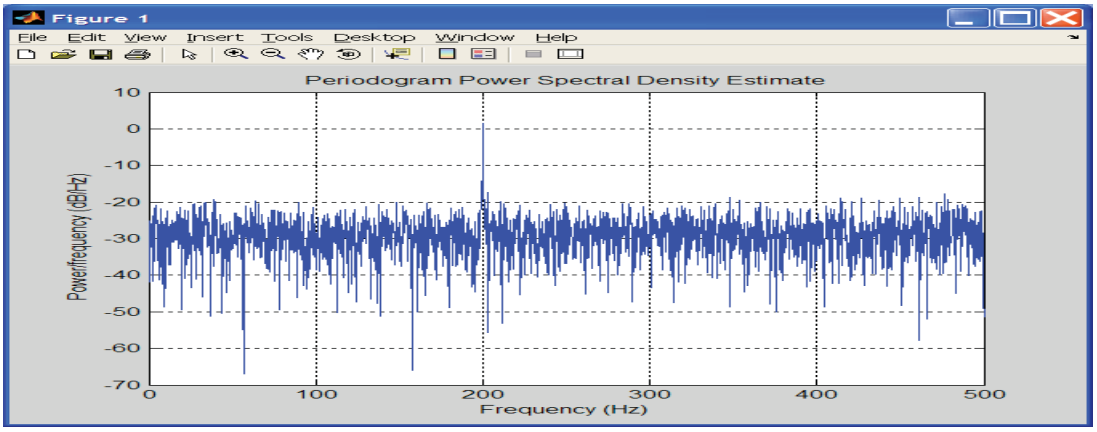


Рис. 4.12

Здесь используется значение окна по умолчанию.

Вычисление периодограммы по методу Уэлча

При вычислении периодограммы по длинному фрагменту случайного сигнала она оказывается весьма изрезанной, что затрудняет оценку формы спектра. Для уменьшения изрезанности необходимо применить усреднение, которое реализуется методом Уэлча. Вычисления при использовании метода Уэлча организуются следующим образом:

1. Вектор отсчетов сигнала делится на перекрывающиеся сегменты. Как правило, используется перекрытие на 50 %. Каждый сегмент умножается на используемую весовую функцию.

2. Для взвешенных сегментов вычисляются модифицированные периодограммы.

3. Периодограммы всех сегментов усредняются.

Следует отметить, что метод Уэлча для оценки спектра использует меньшие по размеру сегменты сигнала, поэтому разрешающая способность спектрального анализа (возможность различить две рядом располо-

женные спектральные линии) в этом случае снижается. Для реализации метода Уэлча используется функция **pwelch.** Синтаксис вызова функции следующий: **[Pxx, f] = pwelch(x, Nwin, Noverlap, Nfft, Fs, 'range')**.

Обязательным входным параметром является **x** – вектор отсчетов анализируемого сигнала. Все остальные параметры имеют значения по умолчанию, которые используются, если при вызове в качестве параметра указана пустая матрица ([]) или если несколько последних параметров опущено.

Параметр **Nwin** управляет выбором окна, используемого для анализа. Если **Nwin** – число, используется окно Хэмминга указанной длины, если вектор, то данный вектор используется в качестве окна. По умолчанию используется окно Хэмминга, длина которого выбирается так, чтобы с учетом заданного перекрытия (см. ниже) сигнал оказался разделенным на восемь фрагментов.

Параметр **Noverlap** задает (в отсчетах) перекрытие соседних фрагментов сигнала, для которых вычисляются периодограммы. По умолчанию перекрытие равно половине длины окна.

Параметр **Nfft** задает размерность БПФ, используемого для вычисления периодограммы. По умолчанию **Nfft** равно максимальному из двух чисел: 256 и 2^k. Значение **k** таково, что выполняется условие $\mathbf{2^k > Nwin,}$ где **Nwin** – длина фрагмента сигнала (длина используемого окна).

Параметр **Fs** указывает частоту дискретизации сигнала. Это значение используется для нормировки рассчитанного спектра мощности, а также при расчете возвращаемого вектора **f** и для оцифровки графика. По умолчанию значение этого параметра равно 2π.

Строковый параметр **'range'** определяет частотный диапазон для возвращаемого вектора **Pxx**. Возможны два значения:

'twosided' – векторы **Pxx** и **f** имеют длину **Nfft** и соответствуют полному диапазону частот **0...Fs**. Этот вариант используется по умолчанию, если **x** содержит комплексные отсчеты;

'onesided' – векторы **Pxx** и **f** соответствуют половинному диапазону частот **0...Fs/2**. Этот вариант используется по умолчанию в случае вещественного вектора **x**.

Параметр **'range'** может быть указан в списке параметров в любом месте после **Noverlap**.

Возвращаемые параметры: **Pxx** – вектор значений спектральной плотности мощности, **f** – вектор значений частот, использованных

для расчета. Шаг между соседними элементами этого вектора равен **Fs/Nfft**, первый элемент равен нулю. Если выходные параметры при вызове не указаны, функция строит график спектральной плотности мощности с помощью функции **psdplot**.

Расчет спектра производится следующим образом. Анализируемый сигнал **x** делится на перекрывающиеся фрагменты согласно параметрам **Nwin** и **Noverlap**. Для каждого фрагмента вычисляется модифицированная периодограмма с использованием заданных окна и размерности БПФ Полученный набор модифицированных периодограмм усредняется. Пример (рис. 4.13):

```
Fs = 1000;
t = 0:1/Fs:3;
f0=200;
x=cos(2*pi*f0*t)+randn(size(t));
pwelch(x,[],[],[],Fs,'onesided');
```

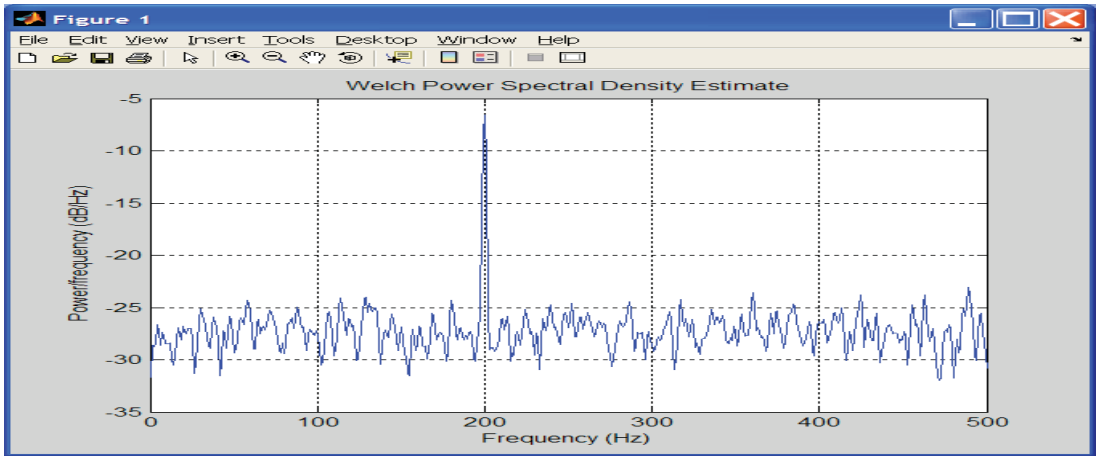


Рис. 4.13

Задание 2

1. Сформировать последовательность из трех импульсов, представляющих собой положительные полуволны синусоиды. Частота синусоиды 0,5 кГц, период следования импульсов 10 мс, частота дискретизации 10 кГц.

2. Сформировать сигнал, представляющий собой сумму гармонического сигнала с частотой 2 кГц и амплитудой 2 В и белого гауссова шума, который имеет уровень на 20 дБ меньший сигнала. Частота дискретизации 10 кГц, интервал времени (0…1) с.

3. Определить периодограмму, спектрограмму, периодограм-

му, построенную по методу Уэлча, для сигнала, представляющего собой сумму гармонического сигнала с частотой 1 кГц и амплитудой 4 В и белого гауссова шума, который имеет уровень на 10 дБ меньший сигнала. Частота дискретизации 10 кГц, интервал времени (0…1) с. Использовать окно Хэмминга длительностью 200 и с перекрытием 20 отсчетов.

4. Построить таблицу изученных команд и функций с пояснениями по их использованию.

Содержание отчета

1. Распечатка истории команд.
2. Тексты решений задач и графики по итогам решения.
3. Таблица изученных команд и функций с пояснениями по их использованию.

Контрольные вопросы

1. Ответить на вопросы преподавателя по использованию изученных команд и функций.
2. Дать сравнительную характеристику функций **pwelch** и **periodogram** по разрешающей способности спектрального анализа.
3. По спектрограмме заданного звукового файла (запись голосовой команды) определить частоты основных формант и частоту основного тона для вокализованных звуков.
4. В каких случаях при анализе сигнала целесообразно использовать спектрограмму, а в каких случаях – периодограмму?

Лабораторная работа 5
ДИСКРЕТНАЯ ФИЛЬТРАЦИЯ В СИСТЕМЕ MATLAB

Цель работы. Изучение возможностей MATLAB для фильтрации сигналов.

Задание 1. Изучить материал, изложенный ниже, и повторить примеры и упражнения, приведенные в тексте.

Краткие теоретические сведения о дискретных фильтрах

В зависимости от наличия обратной связи различают рекурсивные и нерекурсивные фильтры. Наиболее простыми по структуре являются нерекурсивные фильтры. Структура фильтра приведена на рис. 5.1.

Нерекурсивные фильтры имеют ограниченную (конечную) во времени импульсную характеристику, поэтому их часто называют КИХ (FIR)-фильтрами. Достоинства нерекурсивных фильтров: они обладают линейной ФЧХ, то есть не вносят фазовых искажений в сигнал, являются абсолютно устойчивыми.

Рекурсивные фильтры – это фильтры с обратной связью. Структура рекурсивного фильтра приведена на рис. 5.2.

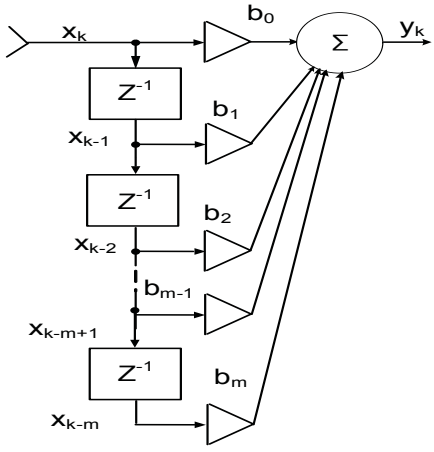


Рис. 5.1

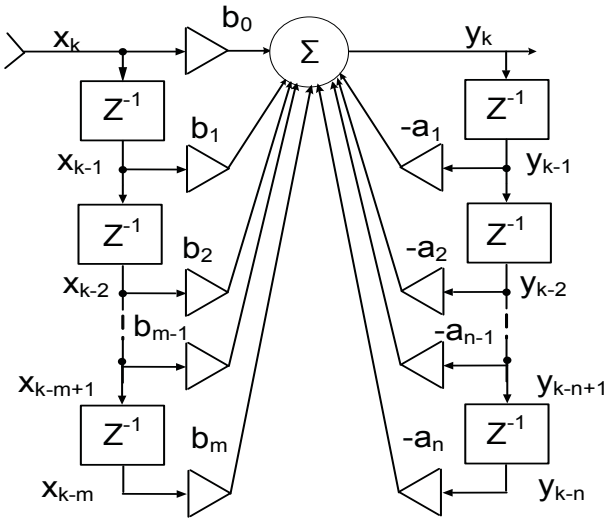


Рис. 5.2

Рекурсивные фильтры имеют бесконечную во времени импульсную характеристику, поэтому их часто называют БИХ (IIR)-фильтрами. Они имеют меньший порядок по сравнению с нерекурсивными фильтрами при одинаковых требованиях к крутизне «ската» АЧХ, но вносят фазовые искажения в сигнал и за счет наличия обратной связи могут быть неустойчивыми.

Основной характеристикой дискретного фильтра является функция передачи (системная функция):

$$H(z)= (b_0+b_1*z^{-2}+\ldots.b_m*z^{-m}) / (1 + a_1*z^{-1} +a_2*z^{-2}+...a_n*z^{-n}).$$

Числитель функции характеризуется вектором **b** коэффициентов, а знаменатель – вектором **a**.

Кроме того, часто используется частотная характеристика фильтра – зависимость комплексного коэффициента передачи фильтра от частоты. Частотная характеристика описывается амплитудно-частотной и фазочастотной характеристиками.

Проектирование (синтез) фильтров осуществляется либо по аналоговому фильтру – прототипу с помощью билинейного преобразования, либо прямым методом, то есть в соответствии с заданной частотной характеристикой. Рекурсивные фильтры обычно проектируются по первому методу, а нерекурсивные – по второму.

Аналоговые фильтры, используемые в качестве прототипов, различаются по типу аппроксимации идеальной (прямоугольной) АЧХ. Наиболее крутой спад АЧХ реализуется эллиптическим фильтром. Однако в этом случае на АЧХ появляются пульсации как в полосе пропускания, так и в полосе заграждения. Наиболее гладкой является АЧХ фильтра Баттерворта, но спад АЧХ у данного фильтра очень пологий.

Кроме указанных аппроксимаций, наиболее часто используются фильтры Чебышева первого и второго рода, которые обеспечивают более крутой спад АЧХ по сравнению с фильтром Баттерворта, но обусловливают пульсации АЧХ либо в полосе пропускания, либо в полосе заграждения.

Основной метод проектирования нерекурсивных фильтров – использование весовых (оконных) функций. Результат проектирования нерекурсивных фильтров зависит от выбора весовой функции. Весовая функция определяет уровень «лепестков» АЧХ в полосе задерживания и ширину переходной зоны АЧХ от полосы пропускания до полосы задерживания.

Следует отметить, что крутизну спада АЧХ любого фильтра можно увеличить, повышая порядок фильтра. Однако при этом увеличивается стоимость реализации фильтра.

Функции, используемые для расчета частотной характеристики фильтра и результатов фильтрации

Для расчета частотной характеристики фильтра используется функция **freqz**. Пример построения амплитудно-частотной характеристики (АЧХ):

```
b=[1 2 3 4];
a=[1 0.1];
N=1024;
```

```
Fs=1e6;
[h,f] = freqz(b,a,N,Fs);
plot(f,abs(h));
[phi,f] = phasez(b,a,N,Fs);
plot(f,phi);
grid on
```

Здесь векторы **b, a** – наборы коэффициентов числителя и знаменателя системной функции соответственно; N – число точек расчета (по умолчанию – 512); **Fs** – частота дискретизации. Выходные параметры: **h, f** – комплексный коэффициент передачи и вектор частот, для которых проведен расчет. Для построения фазочастотной характеристики (ФЧХ) используется функция **phasez**. Пример:

```
[phi,f] = phasez(b,a,N,Fs);
plot(f,phi);
grid on
```

Если использовать функцию **freqz** без указания выходных параметров, то строятся графики АЧХ и ФЧХ (рис. 5.3). Пример:

```
freqz(b,a,N,Fs);
```

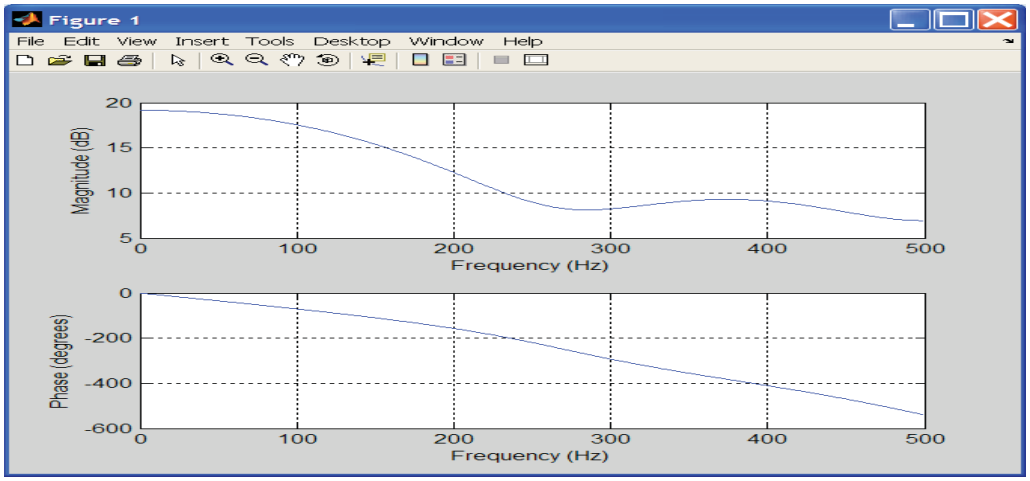


Рис. 5.3

Для расчета результатов фильтрации используется функция **filter(b,a,x)**. Здесь **x** – вектор входного сигнала. Синтаксис вызова функции следующий: **y= filter(b,a,x)**. Здесь **y** – вектор выходного сигнала. Пример:

```
Fs=1000;
```

```
t=0:1/Fs:1;
f1=10;
f2=100;
s1=sin(2*pi*f1*t);
s2=sin(2*pi*f2*t);
s=s1+s2;
plot(t,s);
B=fir1(150,0.02, 'low');
freqz(B,1,[],Fs)
y=filter(B,1,s);
plot(t,y);
```

В данном примере использована функция **fir1(N,Wn,'low')** расчета нерекурсивного фильтра нижних частот. Здесь **N –** порядок фильтра, **Wn –** частота среза (на уровне –6 дБ), нормированная к половине частоты дискретизации (в данном примере **Wn**=0,02*0,5*1000=10Гц). Если последний параметр функции принимает значения: **'high', 'bandpass', 'stop'**, то имеют место соответственно фильтр верхних частот, полосовой фильтр, режекторный фильтр. В последних двух случаях параметр **Wn = [W1 W2]**, где **W1, W2** – граничные частоты.

Функция **filter** может также использовать параметры фильтра, если последний задан в виде объекта. Синтаксис вызова функции в этом случае следующий: **y1 = filter(Hd,x)**. Здесь **Hd** – объект дискретного фильтра. Для создания объекта дискретного фильтра служит функция **dfilt**. Пример вызова функции:

```
Hd = dfilt.dffir(B)
```

После точки указывается идентификатор структуры фильтра (в данном примере используется структура прямой формы нерекурсивного фильтра), в скобках указываются параметры структуры (в данном случае – вектор коэффициентов фильтра). Для данных предыдущего примера использование фильтра в форме объекта показано следующим примером:

```
Hd = dfilt.dffir(B);
y1 = filter(Hd,s);
plot(t,y1);
```

Проектирование фильтров

Для синтеза фильтров используется графическая среда “Filter Design&Analysis Tool”, которая вызывается командой **fdatool**. На рис. 5.4 показано окно программы.

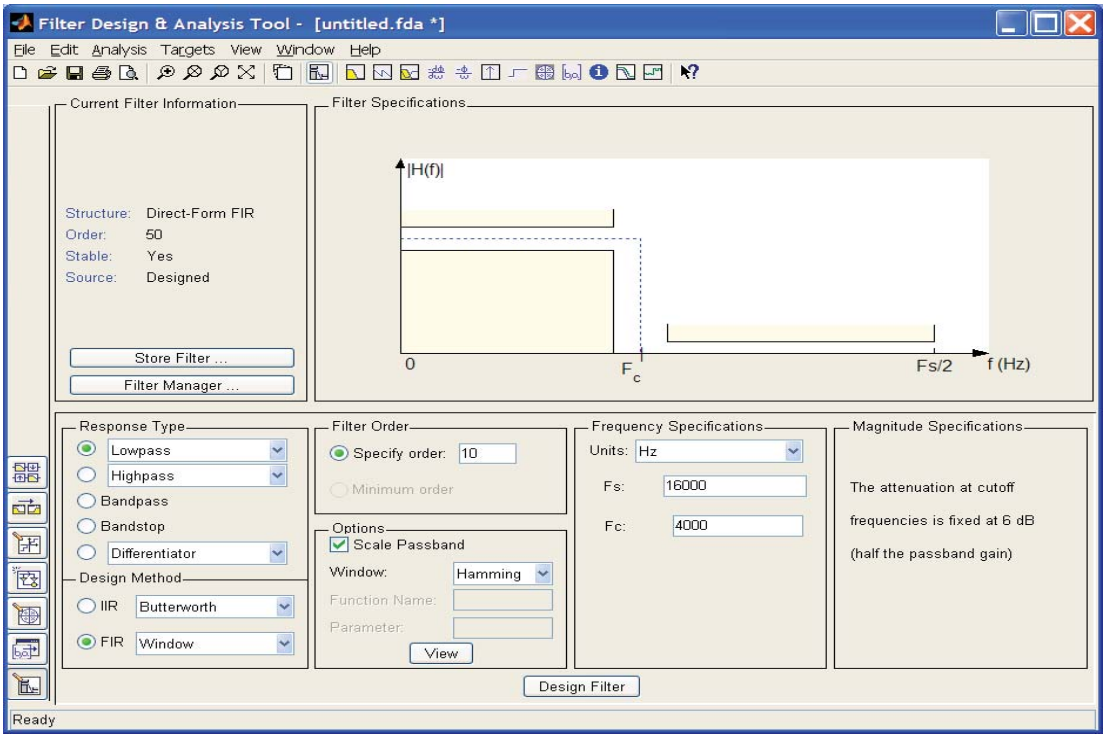


Рис. 5.4

Рассмотрим порядок расчета фильтра на конкретном примере. Пусть необходимо рассчитать нерекурсивный фильтр нижних частот со следующими параметрами: частота среза 4 кГц, частота дискретизации 16 кГц, на частоте 5 кГц необходимо обеспечить подавление не менее 40 дБ.

Открываем вкладку **Filter Design** (самая нижняя кнопка в левом углу окна программы). Выбираем соответствующий тип фильтра, метод синтеза – с помощью весовой (оконной) функции Хэмминга, устанавливаем предварительное значение порядка фильтра равным 10 и заданные значения частот дискретизации и среза. На рис. 5.4 (пункт меню **Analysis|Filter Specifications** или соответствующая кнопка на панели инструментов) видно, как указанные параметры фильтра определяют условия фильтрации.

Запускаем процесс расчета фильтра кнопкой **Filter Design** (в сере-

дине нижней части окна программы). Результаты расчета представлены на рис. 5.5.

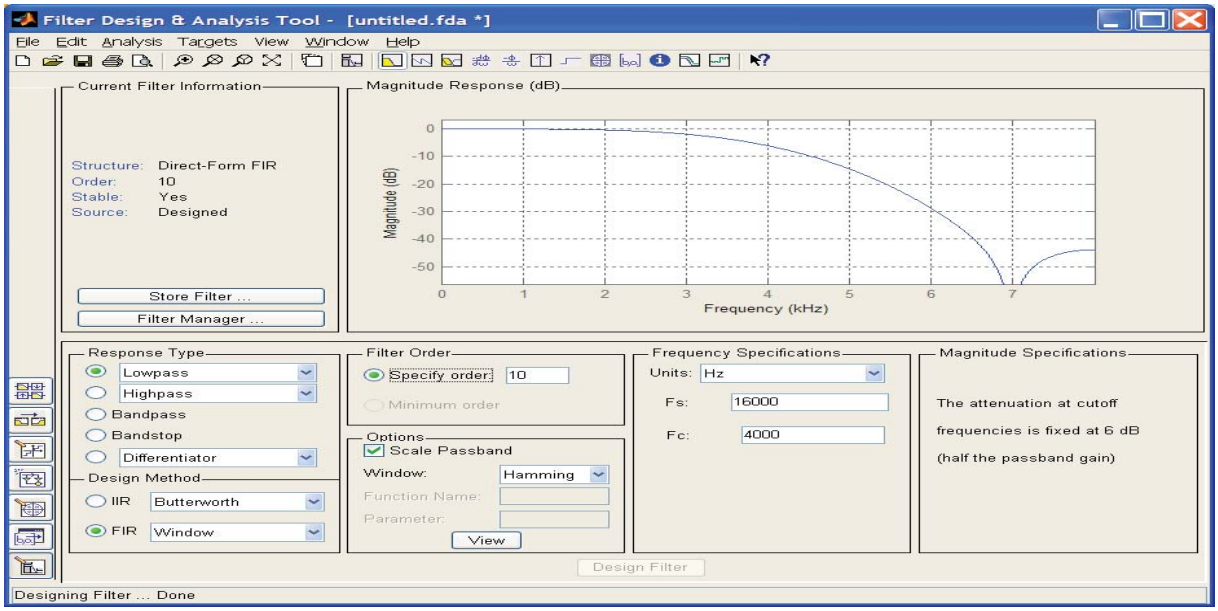


Рис. 5.5

Рис. 5.5 отображает амплитудно-частотную характеристику фильтра. В правой нижней секции окна программы указывается уровень затухания (6 дБ) на частоте среза, в верхней левой секции окна программы – результаты проектирования. Из графика видно, что уровень затухания на частоте 5 кГц составляет примерно 15 дБ, что меньше заданного затухания 40 дБ. Увеличивая порядок фильтра до 24, добиваемся нужного затухания (рис. 5.6).

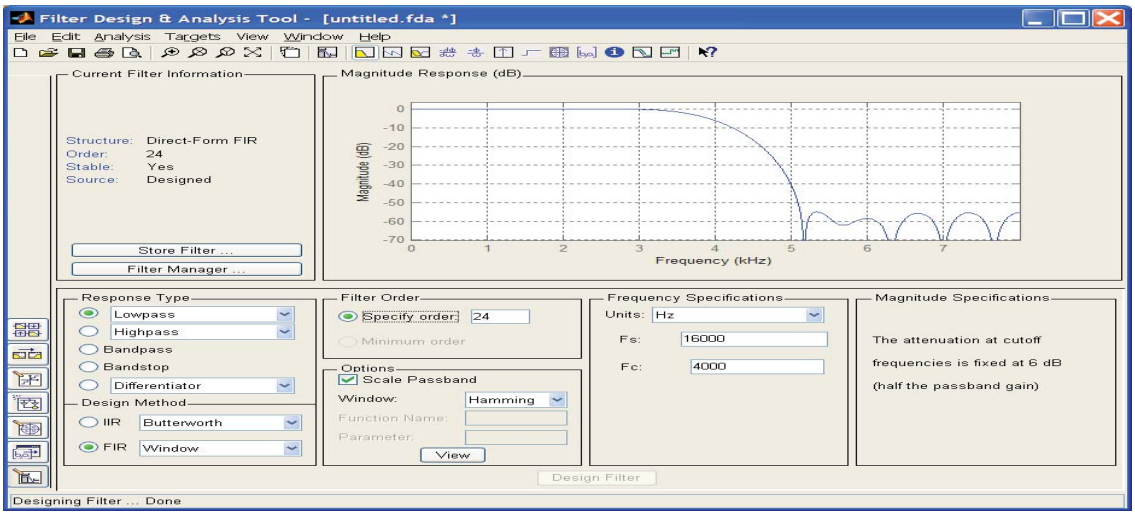


Рис. 5.6

Для сохранения результатов проектирования (в виде коэффициентов нерекурсивного фильтра) экспортируем их в рабочее пространство

MATLAB и сохраняем в переменной **LPFilter** (пункт меню **File|Export**). На рис. 5.7 отражено соответствующее диалоговое окно экспорта.

Используя интерфейс окна, показанного на рисунке, можно сохранить результаты расчета в иной форме: в виде объекта, в текстовом файле и т.д. Для проверки правильности экспорта строим график частотной характеристики фильтра и сравниваем его с аналогичным изображением в данной графической среде.

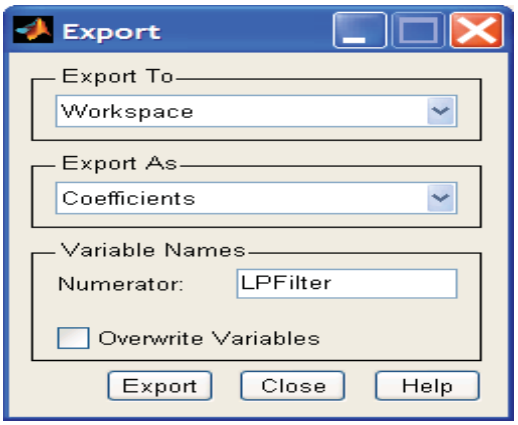


Рис. 5.7

freqz(LPFilter,1,[],16000)

Задание 2

1. Сформировать массив звуковых данных, пораженных гармонической помехой, используя заданный звуковой файл.
2. Провести спектральный анализ полученного массива заданным методом. Сформулировать необходимые требования к фильтру, устраняющему помеху.
3. Спроектировать соответствующий нерекурсивный фильтр, используя метод весовых функций.
4. Отфильтровать пораженный помехой звуковой сигнал и оценить результаты фильтрации.
5. Построить таблицу изученных команд и функций с пояснениями по их использованию.

Содержание отчета

1. Распечатка истории команд.
2. Графики по итогам выполнения задания.
3. Таблица изученных команд и функций с пояснениями по их использованию.

Контрольные вопросы

1. Дать сравнительную характеристику рекурсивных и нерекурсивных фильтров.
2. Ответить на вопросы преподавателя по использованию изученных команд и функций.

3. Ответить на вопросы преподавателя по использованию графической среды проектирования фильтров.

Лабораторная работа 6
ИЗУЧЕНИЕ ИНТЕРФЕЙСА ПАКЕТА SIMULINK

Цель работы. Изучение возможностей пакета SIMULINK для моделирования устройств обработки сигналов. Исследование алгоритма LPC-кодирования речевых сигналов.

Задание 1. Изучить материал, изложенный ниже, и повторить примеры и упражнения, приведенные в тексте.

Основные сведения о пакете SIMULINK

В состав системы MATLAB входит пакет SIMULINK. Разработка моделей средствами SIMULINK (в дальнейшем S-моделей) основана на использовании технологии *drag-and-drop* («перетащи и оставь»). Для построения модели используются модули (или блоки), хранящиеся в библиотеке SIMULINK. Библиотека содержит набор визуальных объектов, используя которые можно собирать произвольную конструкцию в окне редактирования модели. Кроме данной библиотеки, имеются отдельные библиотеки (Blocksets**)**, которые разработаны для решения специфичных задач.

Блоки, входящие в состав библиотеки, имеют параметры настройки. Чтобы открыть окно настройки параметров, нужно поместить блок в окно редактирования модели и дважды щелкнуть мышью на его изображении.

Рассмотрим задачу моделирования работы идеального ограничителя, на вход которого подается синусоидальное напряжение с амплитудой 10 В и частотой 100 Гц, частота дискретизации 2000 Гц. Допустим, что пороги ограничения составят +8 и –8 В. Моделирование проводим на интервале времени от 0 до 0,1 с.

Основными блоками являются генератор синусоидальных сигналов и нелинейность, моделирующая передаточную характеристику ограничителя. Кроме того, к этим блокам следует добавить регистрирующий блок – осциллограф.

Создание модели

Создание модели начинается с активизации кнопки SIMULINK на панели инструментов окна MATLAB (или вводом команды SIMULINK в окне команд). При этом открывается окно броузера библиотеки компонентов.

В окне броузера библиотеки нажимаем кнопку **New** (она на панели инструментов имеет пиктограмму в виде чистого листа). Появится пустое окно редактирования модели. Следующий этап – выбор источника сигнала.

Нужно в окне броузера библиотеки открыть раздел источников **Simulink/Sources**. Затем, активизировав в нем компонент **Sine Wave** (источник гармонических колебаний), мышью (при нажатой левой кнопке) переносим его в окно редактирования модели.

Для установки параметров источника синусоидальных колебаний дважды щелкаем на только что введенном блоке. Появится окно установки параметров источника. Сигнал задаем на оси времени (**Sine Type/Time Based***)*. Устанавливаем заданную амплитуду 10 В и частоту 2* pi *100 рад/с. Фазу и постоянную составляющую (bias) оставляем нулевыми. Период дискретизации (**Sample time**) устанавливаем равным величине 1/2000. Далее нажимаем кнопку **OK** (Применить), и заданные параметры сохраняются.

Выбираем раздел нелинейных элементов – **Discontinuities**. В появившемся окне этих элементов надо выбрать блок **Saturation** (Ограничение) и перенести его в нужное место окна модели, разместив справа от источника синусоидального сигнала (рис. 6.1).

Рис. 6.1

Дважды щелкнув мышью на блоке ограничения, устанавливаем параметры ограничителя: верхний порог ограничения 8 В и нижний –8 В. Поле **Sample Time** (Период дискретизации) устанавливаем равным «–1» (Унаследовано).

Аналогичным образом надо ввести блок осциллографа (раздел **Sinks**, блок **Scope**). Основные установки для данного блока: **Time Range:** 0.1; **Sampling/Sample Time:** 1/2000.

Для соединения блоков устанавливаем указатель мыши на выход источника и, нажав левую кнопку мыши (указатель должен превратиться в крестик), проводим линию до входа ограничителя. Отпустив ле-

вую кнопку мыши, получаем соединительную линию между блоками. Аналогичным образом соединяется выход блока ограничителя с входом блока осциллографа.

Размещаем над моделью пояснительную надпись. Для этого устанавливаем указатель мыши на свободное место окна модели и дважды щелкаем левой кнопкой – появляется окно ввода текста. После ввода надписи устанавливаем указатель мыши вне текстового поля и щелкаем левой кнопкой мыши. Блок с надписью выделяем и переносим в нужное место (рис. 6.2).

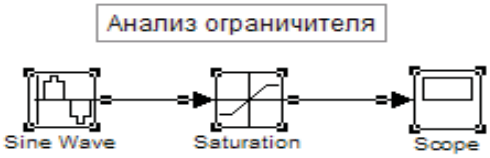


Рис. 6.2

Запуск модели

Следующий этап моделирования – запуск модели. Для настройки запуска модели следует выполнить пункт меню **Simulation/Configuraton Parameters**. В появившемся окне надо установить временной интервал моделирования (Start time 0, Stop time 0.1).

Запускаем модель, нажимая кнопку пуска (треугольник) на панели инструментов или выполняя пункт меню **Simulation/Start**. По завершении процесса моделирования активизация объекта-осциллографа выводит окно, в котором виден результат моделирования. Это окно показано на рис. 6.3.

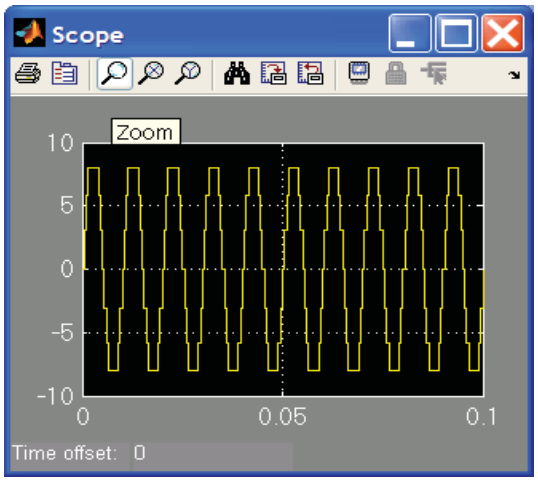


Рис. 6.3

Если на экране осциллографа изображение не соответствует ожидаемому, то целесообразно задать автоматическую настройку, нажав кнопку панели инструментов блока осциллографа с изображением бинокля. При этом автоматически выбираются масштабы просмотра по вертикали и горизонтали. Для «ручной» настройки осциллографа можно активизировать кнопку **Parameters** на панели инструментов его окна.

Для сохранения модели используется команда **Save** или **Save As** в меню **File** окна редактора моделей. Модель сохраняется в файле с расширением .**mdl** (и данном случае **– strn.mdl**).

Запуск модели можно осуществить из командного окна, используя функцию **sim.** Все переменные, используемые в модели, доступны из рабочего пространства. Поэтому можно какой-либо параметр модели обозначить через переменную и, задавая различные значения переменной, осуществлять запуск модели. Обозначим через переменную **f** значение частоты генератора и проведем запуск модели из командного окна.

```
f=100;
sim('strn.mdl');
```

Задание 2

1. Используя справочные данные Приложения, составить модель из последовательного соединения блоков генератора гармонических колебаний и осциллографа. Параметры модели установить в соответствии с заданием. Зафиксировать осциллограмму генерируемых гармонических колебаний.

2. Сформировать сигнал в соответствии с заданием и загрузить его в модель, которая составлена из блоков, указанных на рис. 6.4. Блок From Workspace обеспечивает ввод двумерного массива из рабочего пространства в модель. Первый столбец массива – вектор времени, а второй – вектор значений сигнала. Сравнить «осциллограмму» сигнала с его графиком, построенным с использованием функции **plot**.

3. Используя данные задания для п.1 и блок To Workspace, составить модель в соответствии с рис. 6.5.

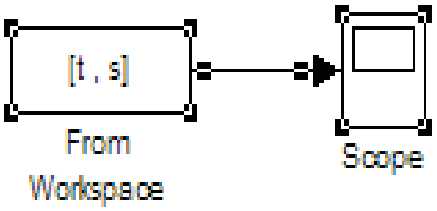


Рис. 6.4

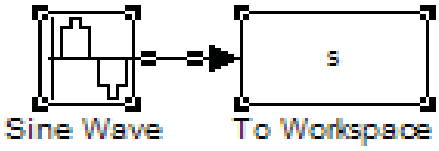


Рис. 6.5

По сформированным векторам времени и сигнала построить график с использованием функции **plot.** Перед построением графика дополнить вектор времени нулем, обеспечив тем самым одинаковую размерность векторов времени и сигнала.

Содержание отчета

1. Распечатка структурных схем исследованных моделей.

2. Графики по итогам выполнения задания.

3. Таблица изученных команд и функций с пояснениями по их использованию.

Контрольные вопросы

1. Дать сравнительную характеристику рекурсивных и нерекурсивных фильтров.

2. Ответить на вопросы преподавателя по использованию изученных команд и функций.

3. Ответить на вопросы преподавателя по использованию графической среды проектирования фильтров.

Приложение. **Справочные сведения по основным блокам SIMULINK**

Раздел «Sources (источники)»

• Chirp Signal – генератор гармонических колебаний переменной частоты;

• Constant – формирует постоянную величину (скаляр, вектор или матрицу);

• Pulse Generator – генератор импульсных сигналов;

• Ramp – создает линейно возрастающий (убывающий) сигнал;

• Random Number – источник дискретного сигнала, амплитуда которого является случайной величиной, распределенной по нормальному закону;

• Repeating Sequence – генератор периодического дискретного сигнала произвольной формы;

• Signal Generator – генератор непрерывного сигнала произвольной формы;

• Step – генерирует единичный дискретный сигнал с заданными параметрами;

• Sine Wave – генератор гармонических колебаний;

• Uniform Random Number – источник дискретного сигнала, амплитуда которого является равномерно распределенной случайной величиной;

• From Workspace – обеспечивает ввод в модель данных непосредственно из рабочей области MATLAB.

Раздел «Sinks (приемники)»

• Scope – «осциллограф»;

• XYGraph – обеспечивает создание двумерных графиков в прямоугольной системе координат;

• Display – обеспечивает отображение численных значений величин;

• To Workspace – обеспечивает сохранение результатов моделирования в рабочем пространстве.

Блок Scope

Блок позволяет в процессе моделирования наблюдать динамику изменения интересующих исследователя характеристик системы. Открыть окно Scope можно только после того, как блок помещен на поле блок-диаграммы (щелкнув дважды на его изображении мышью).

По оси ординат шкалы измерений откладываются значения наблюдаемой величины, по оси абсцисс – значения модельного времени. По умолчанию для оси ординат используется диапазон [–5; 5], а для оси модельного времени – [0; 10].

Если на вход блока Scope поступает векторная величина, то для каждого элемента вектора в окне строится отдельная кривая, отражающая динамику его изменения. Одновременно в окне Scope может отображаться до 30 кривых. Кроме того, пользователь может в одном окне создать несколько осей координат для представления того или другого параметра отдельно от других.

Для управления параметрами окна Scope в нем имеется панель инструментов, содержащая кнопки:

• *Zoom* – изменение масштаба осей графика;

• *Zoom X-axis* – изменение масштаба по оси абсцисс;

• *Zoom Y-axis* – изменение масштаба по оси ординат;

• *Autoscale* – автоматическая установка оптимального масштаба осей;

• *Save current axes settings* – сохранение установленного масштаба осей;

• *Parameters* – открытие окна настройки параметров (панели свойств) блока *Scope;*

• *Print* – печать содержимого окна *Scope.*

Некоторые настройки могут быть также выполнены с помощью команд контекстного меню, которое вызывается на экран щелчком правой кнопкой мыши (указатель мыши при этом должен находиться внутри координатной сетки). Контекстное меню содержит четыре команды:

• *Zoom out* – восстановить исходный масштаб осей;

• *Autoscale* – автошкалирование;

• *Save current axes settings* – сохранение установленного масштаба осей;

• *Axes properties* – установка дополнительных параметров графика. По этой команде открывается дополнительное диалоговое окно, с помощью которого может быть изменен диапазон значений по оси ординат и задана текстовая метка для окна графика; данную команду целесообразно использовать в тех случаях, когда в окне Scope для каждого отображаемого параметра используется отдельная координатная сетка.

Для изменения масштаба по выбранной оси координат необходимо:

1. Нажать соответствующую кнопку изменения масштаба.

2. Подвести курсор мыши к тому участку графика, который должен быть отображен в новом масштабе.

3. Щелкнуть (один раз) мышью; первый щелчок приводит к четырехкратному увеличению масштаба, каждый последующий дает увеличение масштаба в два раза.

При щелчке по кнопке *Parameters* открывается окно, содержащее две вкладки:

• *General* (Общие свойства), содержащая элементы для управления форматом вывода графиков;

• *Data history* (Сохранение данных), позволяющая устанавливать параметры записи данных, отображаемых на графиках, в рабочую область MATLAB.

Вкладка General содержит следующие элементы (рис. 6.6):

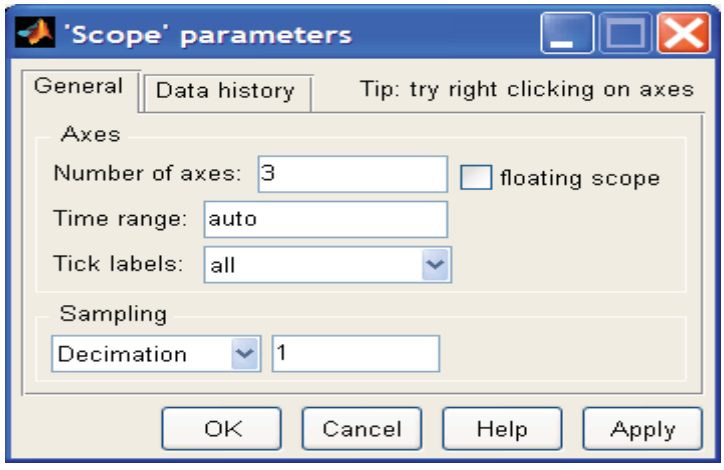


Рис. 6.6

• текстовое поле *Number of axes,* предназначенное для ввода числа подокон (графиков), создаваемых в окне Scope; по умолчанию создается

только одно подокно; оси *Y* всех создаваемых графиков являются независимыми по отношению друг к другу, однако для формирования координаты *X* используются одни и те же значения модельного времени. Значение параметра *Number of axes* определяет число входных портов блока Scope;

• текстовое поле *Time range,* в котором указывается граничное значение диапазона по оси времени (оси *X*); это значение может либо указываться явно, в единицах модельного времени, либо с помощью ключевого слова *auto,* в последнем случае граничное значение времени по оси *X* совпадает с конечным значением модельного времени, установленного для сеанса моделирования;

• раскрывающийся список *Tick labels* используется в том случае, если в окне Scope создано несколько графиков; он позволяет выбрать формат представления оси *X:*

• *bottom axes only* – значения времени по оси *X* выводятся только для нижнего графика;

• *all* – значения времени по оси *X* выводятся для всех графиков;

• *none* – значения по осям *X* и *Y* не выводятся;

• раскрывающийся список *Sampling,* предназначенный для выбора варианта управления периодичностью отрисовки графиков:

• *Decimation* – значение этого параметра интерпретируется как коэффициент «прореживания» выводимых значений; например, если *Decimition* = 3, то значения наблюдаемой характеристики отображаются в окне Scope только на каждом третьем шаге моделирования;

• *Sample time* – периодичность определяется величиной шага модельного времени, установленного для сеанса моделирования; если выполняется моделирование с постоянным шагом, то значение параметра *Sample time* должно совпадать с величиной Δt;

• флажок *floating scope* позволяет установить для блока Scope свойство «плавающий»; такой блок не имеет ни одного входного порта и обеспечивает отображение сигнала, передаваемого по выбранной в блок-диаграмме линии связи.

Если величина заданного интервала моделирования не превышает значение *Time range*, то под графиком в строке *Time offset* выводится 0. В противном случае отображается только отрезок времени, равный $T_M - n*(Time\ range)$, где T_M – длительность интервала моделирования, n – целое число. При этом в строке *Time offset* выводится величина «скрытого» интервала времени $n*(Time\ range)$.

Вкладка Data history позволяет задавать максимальный объем и способ хранения отображаемых в окне данных. Объем сохраняемых данных *(Limit rows to last)* вводится в строке редактирования. Если соответствующий флажок снят, то ограничения определяются объемом свободного пространства рабочей области MATLAB.

Способ хранения указывается с помощью флажка *Save data to workspace:* если он установлен, то отображаемые в окне Scope данные можно сохранить в рабочей области MATLAB в виде матрицы (Format/ Array). Имя матрицы указывается в строке редактирования (по умолчанию – ScopeData).

Лабораторная работа 7
ИССЛЕДОВАНИЕ LPC-РЕЧЕВОГО КОДЕРА

Цель работы. Исследование алгоритма LPC-кодирования речевых сигналов с помощью пакета SIMULINK.

Краткие теоретические сведения

Можно выделить два ключевых момента обработки речевых сигналов в системе компьютерной телефонии: организация передачи телефонной речи по вычислительным сетям и автоматическое распознавание голосовых команд. Обе эти задачи решаются на основе определения параметров речевого сигнала – эффективного кодирования (сжатия) потока данных речи.

Современные алгоритмы сжатия и распознавания голосовых команд являются очень сложными, поэтому аналитически учесть влияние помех и изменчивости произнесения команд на работу речевого кодека не представляется возможным. В данной ситуации наиболее целесообразным подходом для учета указанных факторов на этапе разработки кодека является использование имитационного моделирования его работы.

Речь состоит из последовательности звуков, которые образуются за счет возбуждения голосового тракта потоком воздуха. Звук представляется как реакция некоторого линейного фильтра на сигнал возбуждения. Причем фильтр моделирует работу голосового тракта, а сигнал возбуждения – работу голосовых связок.

Звуки речи могут быть разделены на две основные группы по типу возбуждения голосового тракта: вокализованные, невокализованные. Вокализованные звуки образуются при прохождении воздуха через вибрирующие голосовые связки. При этом возникает квазипериодическая последовательность импульсов потока воздуха, возбуждающих голосовой тракт. Примеры вокализованных звуков: “А”, ”О”, ”У”... .

Невокализованные звуки образуются при сужении голосового тракта в каком-либо месте и прохождении через него турбулентного воздушного потока. При этом формируется широкополосный шум, возбуждающий голосовой тракт. Примеры невокализованных звуков: “Ш”, ”С”, ”Х”... .

Исходя из описанного выше механизма образования звуков речи, можно предложить следующую электрическую схему, упрощенно моделирующую этот механизм. Речевой сигнал – это выход линейного фильтра, частотная характеристика (она задается, примерно, десятью коэффициентами) которого меняется во времени (примерно 100 раз в секунду) в соответствии с формируемыми (синтезируемыми) сигналами различных звуков речи.

На вход фильтра подается сигнал возбуждения: подключается либо генератор белого шума (когда моделируются невокализованные звуки), либо генератор коротких прямоугольных импульсов (случай вокализованных звуков). Частота следования импульсов равна частоте вибраций голосовых связок (частоте основного тона речевого сигнала). Уровень громкости определяется результатом перемножения сигнала возбуждения с коэффициентом усиления.

Описанная электрическая модель формирования звуков речи, по сути, является структурой речевого декодера, который стоит на приемной стороне канала связи, и по параметрам речевого сигнала, переданным по линии связи, синтезирует сигнал.

Таким образом, задача сжатия данных речевого сигнала решается заменой данных непосредственно сигнала потоком данных небольшого количества его параметров, которые меняются во времени значительно медленнее, а следовательно, и передаваться могут с меньшей скоростью.

При сжатии сигнала возникает задача определения указанных параметров – кодирования сигнала. Рассмотрим решение задачи определения параметров сигнала. Сначала на выделенном интервале стационарности

сигнала определяются параметры частотной характеристики фильтра, моделирующего работу голосового тракта на этом интервале времени.

Можно показать, что параметры частотной характеристики полностью определяются корреляционной функцией данного фрагмента (сегмента) сигнала. Можно также показать, что найденные параметры частотной характеристики однозначно соответствуют фильтру, который по нескольким предыдущим отсчетам речевого сигнала может с некоторой погрешностью вычислить (предсказать) значение последующего отсчета. Чем больше предыдущих отсчетов, тем меньше погрешность предсказания, но больше вычислительные затраты.

Найденные параметры частотной характеристики используются для построения фильтра с частотной характеристикой, обратной характеристике голосового тракта. Затем речевой сигнал пропускается через данный фильтр. При этом на выходе обратного фильтра формируется сигнал, равный погрешности предсказания. Уровень сигнала погрешности (остатка) предсказания оказывается намного меньше уровня самого сигнала. Поэтому, если сигнал остатка предсказания подвергнуть аналого-цифровому преобразованию (АЦП), то количество разрядов квантователя окажется меньше разрядности квантователя при АЦП самого сигнала в условиях сохранения равенства шумов квантования в том и другом случаях.

Из этого факта следует возможность уменьшения (сжатия) потока данных речевого сигнала, если по линии связи передавать отсчеты не самого сигнала, а отсчеты остатка предсказания и небольшое количество параметров частотной характеристики голосового тракта. По переданным данным фильтр, частотная характеристика которого формируется равной частотной характеристике голосового тракта, синтезирует речевой сигнал.

Еще большей степени сжатия можно достигнуть, если по линии связи предавать не сигнал остатка предсказания, а основные его параметры, которые позволят на приемной стороне с определенной точностью восстановить сигнал остатка предсказания.

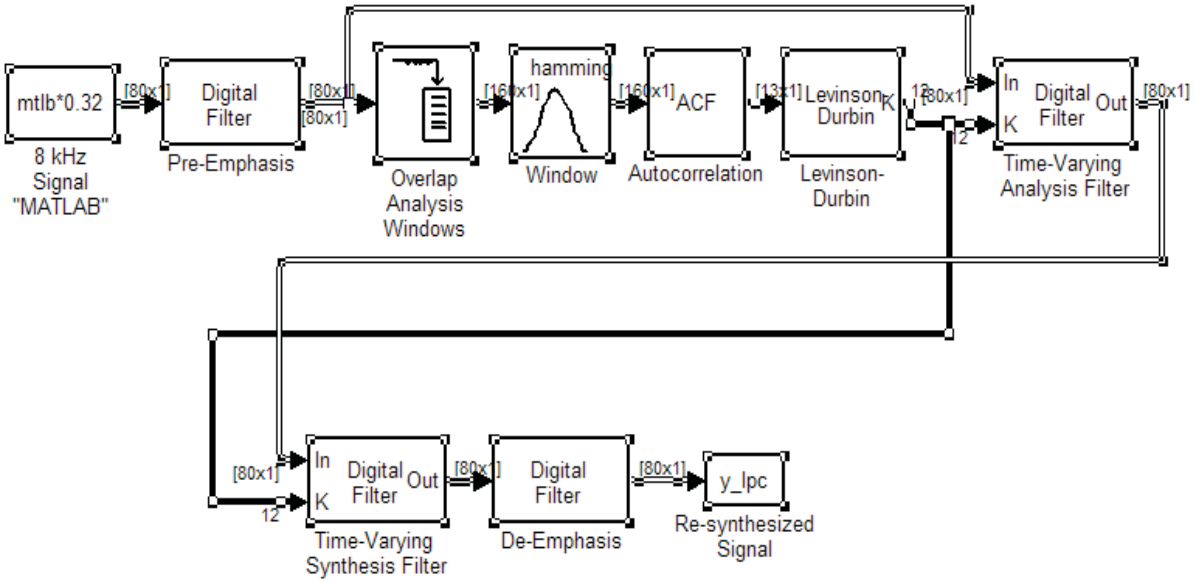
Если звук вокализованный, то сигнал остатка предсказания очень похож на последовательность коротких импульсов, и частота следования импульсов равна частоте вибрации голосовых связок. В этом случае сигнал остатка предсказания можно приближенно заменить последовательностью коротких прямоугольных импульсов. Основным пара-

метром сигнала остатка предсказания в этом случае является частота колебаний голосовых связок.

В случае невокализованного звука сигнал остатка предсказания приближенно заменяется белым шумом. Уровень громкости в обоих случаях задается коэффициентом усиления сигнала возбуждения синтезирующего фильтра. Таким образом, для обеспечения высокого коэффициента сжатия по линии связи, кроме параметров частотной характеристики голосового тракта, передаются: тип синтезируемого сигнала (вокализованный – невокализованный), частота (период) основного тона и коэффициент усиления.

Так как процедура определения параметров частотной характеристики голосового тракта, по сути, использует подход, применяемый при предсказании последующего отсчета сигнала по нескольким предыдущим, то говорят о кодировании (сжатии) речевого сигнала методом линейного предсказания (Linear Prediction Coding) – о LPC-кодеках.

Проанализируем алгоритм сжатия речевых сигналов на основе метода линейного предсказания, используя демонстрационный пример дополнительной библиотеки (Blockset) **Signal Processing** пакета SIMULINK. Схема модели соответствующего речевого кодека представлена на рисунке.



Первый блок **Signal From Workspace** модели обеспечивает загрузку переменной **mtlb**, которая хранит отсчеты речевого сигнала, из рабочего пространства в модель. Кроме того, данный блок разбивает речевой сигнал на отдельные сегменты (фреймы – frames) заданной длительности (в данном примере каждый сегмент содержит 80 отсчетов

сигнала, частота дискретизации составляет 8 кГц). Далее сигнал проходит через цифровой нерекурсивный предыскажающий фильтр (**Digital Filter**) с системной функцией $h(z)=1–0{,}95z^{-1}$, который обеспечивает необходимую точность последующей оценки спектра сигнала.

Затем сигнал проходит через буфер (**Overlap Analysis Windows**), где он разбивается на ряд перекрывающихся во времени сегментов (в данном примере длина сегмента равна 160 отсчетам сигнала, а величина перекрытия составляет 80 отсчетов). Перекрытие сегментов во времени обеспечивает более точное отслеживание изменения параметров сигнала во времени. В следующем блоке каждый сегмент взвешивается весовой функцией Хэмминга, что необходимо для более точной оценки спектра сигнала.

В блоке **Autocorrelation** вычисляется автокорреляционная функция сегмента сигнала. Число отсчетов корреляционной функции определяется заданной величиной порядка предсказания, то есть числом (в данном примере 12) предыдущих отсчетов сигнала, по которым предсказывается значение последующего отсчета.

Блок **Levinson-Durbin** обеспечивает вычисление коэффициентов отражения (частной корреляции) по алгоритму Левинсона – Дарбина, которые содержат информацию об огибающей кратковременного спектра речевого сигнала (для анализируемого сегмента) – о частотной характеристике голосового тракта.

Полученные коэффициенты поступают далее на управляющие входы нерекурсивного лестничного фильтра (блок **Time-Varying Analysis Filter**). При этом на сигнальный вход фильтра поступает кодируемый сигнал. На выходе фильтра формируется сигнал остатка предсказания.

Остаток предсказания и коэффициенты отражения поступают в канал связи, а затем в декодер речевого сигнала, где синтезируется сигнал. Процедура синтеза осуществляется в рекурсивном лестничном фильтре (блок **Time-Varying Synthesis Filter**). Системная функция этого фильтра является обратной по отношению к функции нерекурсивного лестничного фильтра.

С выхода фильтра сигнал поступает на второй рекурсивный фильтр, который устраняет искажения сигнала, введенные в него предыскажающим фильтром. Системная функция этого корректирующего фильтра является обратной по отношению к функции предыскажающего фильтра. Выход корректирующего фильтра является вы-

ходом декодера. С помощью блока **To Workspace** полученный вектор сигнала сохраняется в рабочем пространстве в переменной **y_lpc**.

Следует отметить, что рассмотренная схема речевого кодека обеспечивает относительно небольшое сжатие сигнала в основном лишь за счет уменьшения уровня остатка предсказания по сравнению с исходным сигналом. Запуск модели можно осуществить из командного окна, используя функцию **sim**:

sim('dsplpc_edit.mdl');

Все переменные, используемые в модели, доступны из рабочего пространства.

Задание

1. Загрузить модель LPC-речевого кодека.

2. Запустить процесс моделирования и зафиксировать осциллограммы и спектрограммы входного, выходного сигналов, а также сигнала остатка предсказания. Прослушать входной и синтезированный сигналы. Сформулировать выводы относительно сравнения сигналов по форме и уровню, а также по результатам прослушивания.

3. Сохранить файл модели в своей папке. Прочитать в рабочее пространство массив заданного звукового файла. Провести моделирование процесса кодирования звукового файла с заданными параметрами: размером сегмента сигнала, величиной перекрытия сегментов, формой окна и количеством коэффициентов отражения (порядком предсказания). Оценить результаты моделирования согласно п. 4. При выполнении данного пункта задания следует использовать справочные данные по блокам моделей, приведенные в приложении.

4. Для заданного номера сегмента звукового файла построить график зависимости мощности остатка предсказания от порядка предсказания. Следует воспользоваться дополнительным выходом «Мощность остатка предсказания» блока **Levinson – Durbin.** Данный выход организуется путем дополнительной настройки блока. При выполнении данного пункта задания следует обозначить порядок предсказания через переменную, использовать операторы цикла, а также осуществлять запуск модели из командного окна.

Содержание отчета

1. Структурная схема модели.

2. Распечатка результатов моделирования.

3. Файлы *.mdl.

Контрольные вопросы

1. Как изменятся сигнал остатка предсказания и синтезированный сигнал при уменьшении порядка предсказания?

2. Имеются два кодека. В одном на вход синтезирующего фильтра подается сигнал остатка предсказания, а во втором – либо белый шум, либо последовательность импульсов. В каком кодеке уровень искажений сигнала больше?

3. Что следует изменить в модели кодека, чтобы увеличить громкость синтезированного сигнала?

4. Ответить на вопросы преподавателя по использованию блоков моделей.

Приложение. **Справочные данные по блокам, используемым для моделирования LPC-кодека**

Блок **Signal From Workspace** находится в библиотеке (Blockset) **Signal Processing** в разделе **DSP Sources**. Он обеспечивает разбивку вектора звукового сигнала на сегменты. Вектор предварительно заносится в рабочее пространство MATLAB с помощью функции **Wavread**.

При задании параметров блока **Signal From Workspace** указывается имя переменной рабочего пространства, в которой хранятся данные сигнала, задается период его дискретизации (в данном случае он равен 125 мкс) и количество отсчетов сигнала в одном сегменте (в примере 160). Кроме того, указывается форма сигнала на интервале времени между моментами окончания входного сигнала и интервала моделирования (в примере – нулевой уровень). На выходе блока формируются сегменты (фреймы) длительностью 125·160 мкс.

Блок **Buffer** находится в разделе **Signal Management/Buffers**. Он обеспечивает перекрытие сегментов. Задается размер буфера – это размер сегмента на выходе блока (здесь – 160) и величина перекрытия сегментов в отсчетах сигнала (здесь – 80). Длительность сегментов на выходе буфера составит 125·(160 – 80) мкс. Начальные условия (здесь они нулевые) определяют несколько значений первых отсчетов сигнала.

Блок **Window Function** находится в разделах **DSP Sources, DSP Operations**. Он обеспечивает взвешивание сегмента окном заданного типа.

Блок **Digital Filter** находится в разделе **Filtering/Filter Designs**. Фильтр обеспечивает предыскажения сигнала. Блок **Digital Filter** реализует различные виды цифровых фильтров. Они могут задаваться как в прямой, так и в лестничной форме.

В данном примере предыскажения сигнала формируются нерекурсивным фильтром с конечной импульсной характеристикой, который позволяет сформировать нули частотной характеристики. Фильтр задан в прямой форме, поэтому его частотная характеристика, представленная в виде дробной функции *z*, определяется коэффициентами числителя.

Фильтр с обратной частотной характеристикой, устраняющий предыскажения, является рекурсивным фильтром. Его передаточная характеристика определяется коэффициентами знаменателя. Фильтр задается в диалоговом окне параметрами своей передаточной характеристики

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_1 + b_2 z^{-1} + \ldots + b_{m+1} z^{-(m-1)}}{a_1 + a_2 z^{-1} + \ldots + a_{n+1} z^{-(n-1)}},$$

где параметры числителя определяют вектор коэффициентов числителя, [b(1) b(2) ... b(m)], а параметры знаменателя – вектор коэффициентов знаменателя [a(1) a(2) ... a(n)]. Коэффициенты фильтра нормализуются по a1.

Начальные условия. По умолчанию, внутренние состояния фильтра инициализируются нулевыми значениями, что эквивалентно допущению – предыдущие входные и выходные значения равнялись нулю. Данный блок может иметь и ненулевые начальные условия для элементов задержки. Число состояний фильтра (элементов задержки) на каждый канал составляет max (m, n) – 1.

Начальные состояния параметров могут иметь четыре вида:

– Пустая матрица. Пустая матрица [] инициализирует все элементы задержки всех каналов фильтра нулями.

– Скаляр. Скалярное значение копируется для всех элементов задержки в каждом канале фильтра. Заметим, что значение ноль эквивалентно установке начальных параметров, равными пустой матрице [].

– Вектор. Вектор имеет длину, равную числу элементов задержки в каждом канале фильтра max (m, n) – 1, и определяет уникальные начальные условия для каждого элемента задержки в канале фильтра. Этот вектор начальных условий применяется к каждому каналу фильтра.

– Матрица. Матрица содержит уникальные начальные условия для

каждого элемента задержки и может задавать различные начальные условия для каждого канала фильтра. Матрица должна иметь число строк, равное количеству элементов задержки фильтра max (m, n) – 1, и должна иметь по одному столбцу на каждый канал фильтра.

Блок **Autocorrelation** находится в разделе **Statistics**. Он обеспечивает вычисление коэффициентов автокорреляции. Коэффициенты автокорреляции нужны для последующего вычисления коэффициентов частной корреляции.

Блок **Levinson-Durbin** находится в разделах **Math Functions/Matrices** и **Linear Algebra/Linear System Solvers**. Он обеспечивает вычисление коэффициентов частной корреляции.

Блок **Time-Varying Lattice Filter** – лестничный фильтр – находится в разделе Filtering/Filter Designs. Он по заданным коэффициентам частной корреляции формирует из входного сигнала сигнал остатка предсказания (фильтр при этом должен работать в режиме формирования нулей в частотной характеристике). Фильтр может также использоваться и для синтеза сигнала по заданным коэффициентам частной корреляции и по сигналу возбуждения (фильтр при этом должен работать в режиме формирования полюсов в частотной характеристике).

Блок **To Workspace** находится в основном разделе **SIMULINK** в подразделе **Sinks**. Он предназначен для сохранения полученных данных в рабочем пространстве MATLAB.

Лабораторная работа 8
МОДЕЛИРОВАНИЕ LPC-РЕЧЕВОГО ДЕКОДЕРА СРЕДСТВАМИ ПАКЕТА ВИЗУАЛЬНОГО МОДЕЛИРОВАНИЯ SIMULINK

Цель работы. Исследование алгоритма декодирования речевых сигналов.

Задание 1. Изучить материал, изложенный ниже, и повторить примеры и упражнения, приведенные в тексте.

Краткие теоретические сведения

При простейшем варианте построения речевого LPC – кодека сигнал

остатка предсказания (выходной сигнал кодера) используется в качестве сигнала, возбуждающего синтезирующий фильтр декодера. Однако в этом случае степень сжатия речевого сигнала невелика. Чтобы ее повысить, остаток предсказания моделируют и по каналу связи передают параметры модели. В простейшем случае этими параметрами являются коэффициент усиления (уровень сигнала предсказания), признак тон/шум, период основного тона в случае вокализованных сегментов речевого сигнала.

Рассмотрим состав модели сигнала возбуждения в указанном случае для ситуации с вокализованным звуком (рис. 8.1).

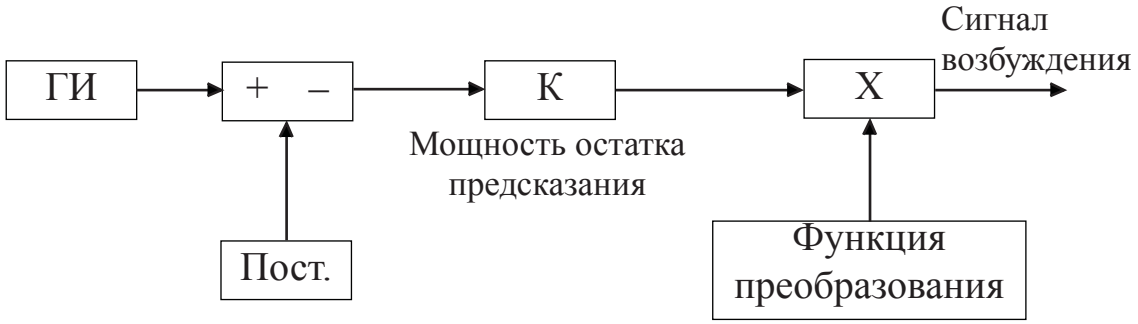


Рис. 8.1

Здесь ГИ – генератор импульсов, Пост. – формирователь постоянной величины, К – звено с постоянным коэффициентом передачи К (для регулировки уровня синтезированного сигнала), Х – перемножитель сигнала и изменяемой постоянной величины, которая формируется блоком «Функция преобразования» из информации о мощности сегмента сигнала остатка предсказания.

Блок «Функция преобразования» в данном случае реализует функцию квадратного корня для формирования значения коэффициента усиления сигнала возбуждения. Формирователь постоянной величины нужен для вычитания постоянной составляющей из сигнала генератора, что обеспечивает нормальное функционирование синтезирующего фильтра, который является рекурсивным фильтром.

Определение параметров блоков: ГИ, Пост.

Рассмотрим следующий пример. Пусть частота дискретизации f = 8000 Гц, период следования импульсов возбуждения 30/8000. Тогда параметр генератора **Puls Width** определяется как [(1/8000) /(30/8000)]100 = 100/30. Постоянная составляющая при единичной амплитуде импульсов ГИ определяется как (1/8000) /(30/8000) = 1/30.

Для упрощения реализации модели учтем, что частота основного тона при произнесении слова меняется мало, поэтому считаем параметры генератора импульсов постоянными. С учетом вышеизложенного блок-схема модели формирования сигнала возбуждения выглядит следующим образом (рис. 8.2).

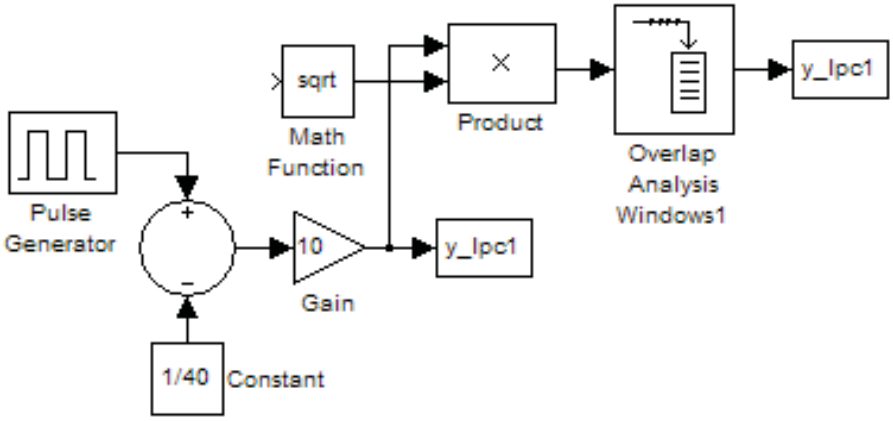


Рис. 8.2

Здесь на вход блока извлечения квадратного корня подается сигнал мощности остатка предсказания с дополнительного выхода блока Levinson-Durbin. Выход формируется путем дополнительной настройки блока. Блок буфера необходим для синхронизации потоков данных, поступающих на синтезирующий фильтр декодера: каждому вектору коэффициентов отражения должен соответствовать «свой» сегмент сигнала возбуждения синтезирующего фильтра. Величина перекрытия сегментов в буфере равна нулю, а количество отсчетов сегмента $N_1 = N - M$, где N – число отсчетов в сегменте, которое используется в буфере кодера, а M – величина перекрытия сегментов в буфере кодера.

Формирование сигнала возбуждения синтезирующего фильтра для вокализованных и невокализованных звуков

Как указывалось выше, в данном случае необходимо на стадии кодирования сигнала определить признак тон/шум, то есть произвести классификацию звуков на вокализованные и невокализованные. Одним из способов такой классификации является использование оценки $R(\tau)$ автокорреляционной функции остатка предсказания анализируемого сегмента сигнала. В случае вокализованных звуков она является периодической, и ее значения при задержках, кратных периоду

вибраций голосовых связок (периоду основного тона речевого сигнала), сравнимы со значением функции при нулевой задержке.

Если же звук невокализованный, то значение функции при нулевой задержке намного больше значений функции при ненулевых задержках. Следовательно, можно классифицировать звуки по отношению $r = R(\tau)/R(0)$ значений функции при нулевой и ненулевой задержках. Если величина r больше некоторого порогового значения, то звук считается вокализованным. В противном случае звук считается невокализованным.

Для определения корреляционной функции необходимо сигнал остатка предсказания подать на вход вычислителя корреляционной функции, который реализуется моделью, изображенной на рис. 8.3.

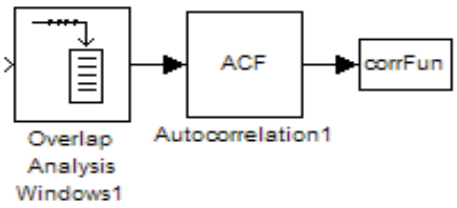


Рис. 8.3

Здесь параметры буфера устанавливаются такими же, как и в модели кодера. Параметры блока вычисления автокорреляционной функции несколько отличаются от аналогичного блока в модели кодера. Корреляционная функция рассчитывается на всем интервале длительности сегмента сигнала и нормируется относительно значения при нулевой задержке. На рис. 8.4 приведено изображение окна установки параметров блока.

Рис. 8.4

Результаты работы блока выводятся в рабочее пространство MATLAB и обрабатываются функцией **clsf** классификации, алгоритм работы которой соответствует материалу, изложенному выше. Код функции следующий:

```
function class=clsf(correl,r)
[m,n,q]=size(correl);
correl_less=zeros(m,q);
class=zeros(1,q);
```

```
for d=1:q
correl_less(:,d)=correl(:,1,d);
end
max_corr1=max(correl_less(16:150,:));
medmax=medfilt1(max_corr1,3);
for d1=1:q
if medmax(1,d1)>r
class(1,d1)=1;
end
end
```

Входными параметрами функции являются массив **correl** значений функций корреляции и значение порога **r** классификации ($0 < r < 1$). Сначала уменьшается размер массива с трехмерного до двумерного. Затем с помощью функции **max** для каждого сегмента остатка предсказания определяется максимальное значение нормированной функции автокорреляции для ненулевых значений задержки. Далее с помощью медианного фильтра уменьшаются флуктуации функции.

Работа функции заканчивается формированием массива с результатами классификации. Если сегмент сигнала соответствует вокализованному звуку, то соответствующий элемент массива равен единице. В противном случае он равен нулю. Полученный массив результатов классификации используется для формирования сигнала возбуждения синтезирующего фильтра в речевом декодере (рис. 8.5).

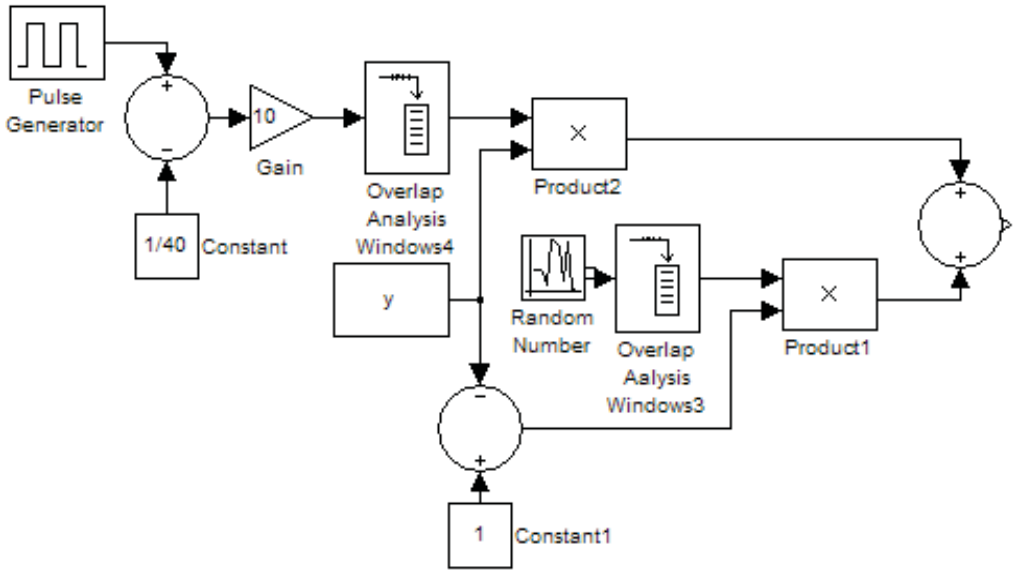


Рис. 8.5

По сравнению с ранее рассмотренным формирователем импульс-

ного сигнала возбуждения здесь используется генератор шума, который в блоке суммирования объединяется с импульсным сигналом. Перед объединением импульсный сигнал и белый шум проходят через блоки перемножения (Product1, 2), которые играют роль переключателей. Управление переключателями осуществляется массивом **y** с результатами классификации, который вводится в модель с помощью блока **Signal from Workspace** (Signal Processing Blockset, раздел Signal Processing Sourses). Диалоговое окно установки параметров блока изображено на рис. 8.6.

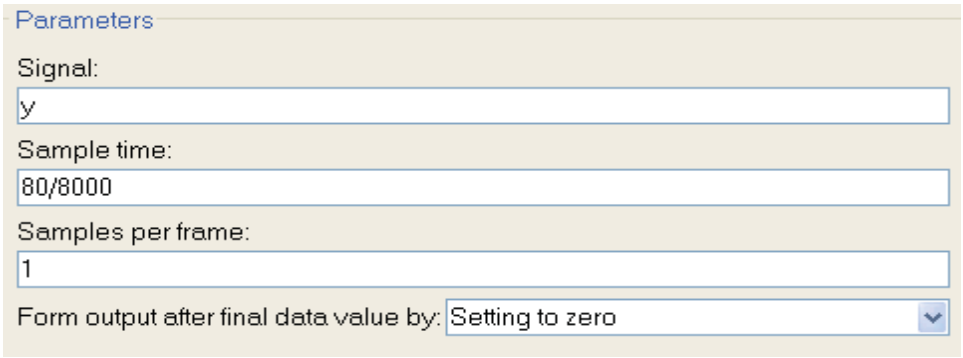


Рис. 8.6

Чтобы переключатели работали в противофазе, используется совокупность блоков ввода постоянной величины (единицы) и вычитания. Блоки буферизации данных используются для синхронизации переключателей с потоками данных от импульсного генератора и генератора шума. Диалоговое окно установки параметров блоков изображено на рис. 8.7.

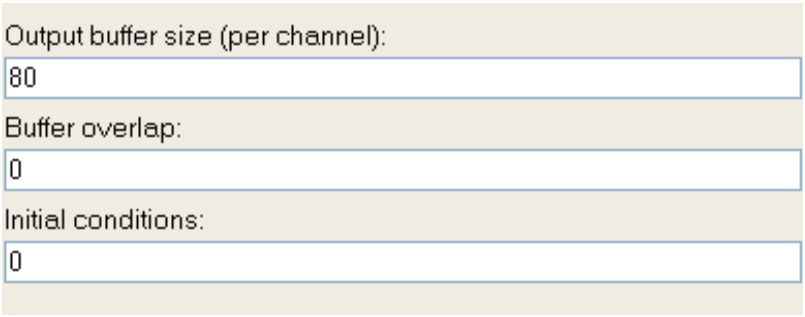


Рис. 8.7

Общая блок-схема формирователя сигнала возбуждения синтезирующего фильтра изображена на рис. 8.8.

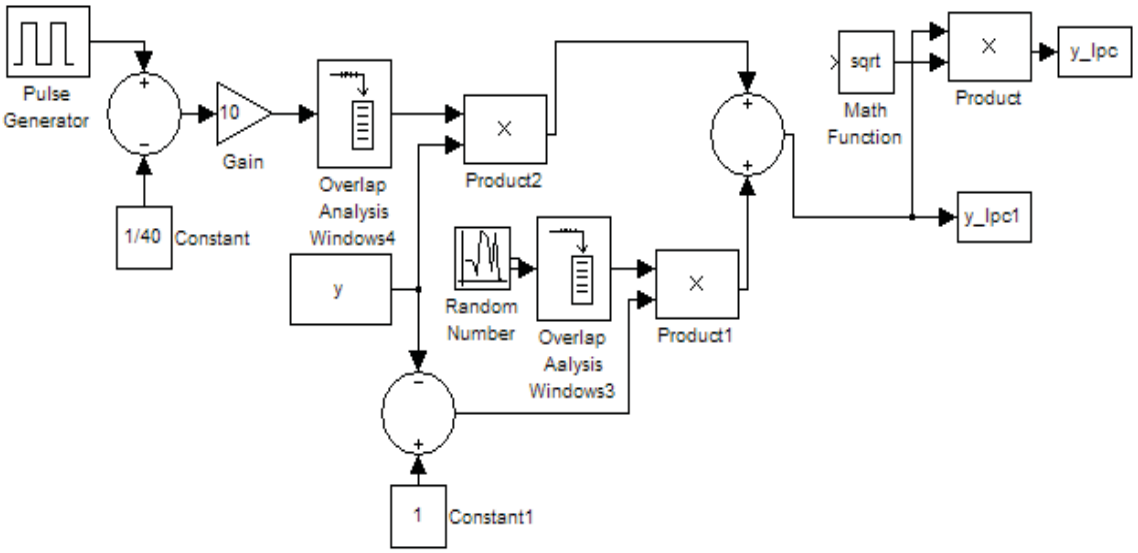


Рис. 8.8

Информацию о мощности сигнала остатка предсказания можно получить из блока **Levinson-Durbin**, установив блок в соответствующий режим.

Задание 2

1. Используя модель **LPC** кодека, в котором возбуждение синтезирующего фильтра осуществляется сигналом остатка предсказания, а также модель формирования импульсного сигнала возбуждения, создать модель кодека с импульсным сигналом возбуждения синтезирующего фильтра.

2. Прочитать в рабочее пространство заданный звуковой файл.

3. Используя модель **LPC** кодека, в котором возбуждение синтезирующего фильтра осуществляется сигналом остатка предсказания, определить с помощью функций MATLAB частоту основного тона в сигнале остатка предсказания для вокализованного сегмента речевого сигнала.

4. Установить в блоке импульсного генератора созданной модели период следования импульсов, соответствующий определенной ранее частоте основного тона. Рассчитать значение постоянной составляющей импульсной последовательности и установить полученное значение в блоке **Constant**.

5. Запустить созданную модель для заданного звукового файла.

6. Построить «осциллограммы» исходного и синтезированного речевых сигналов, обеспечив регулировками модели одинаковые уровни сигналов.

7. Меняя период следования импульсов (одновременно меняя значение постоянной составляющей и параметра **Puls Width**), проследить за изменениями формы синтезированного сигнала и прослушать соответствующий звуковой файл.

8. Сформулировать выводы о различиях исходного и синтезированного речевых сигналов.

9. Создать модель кодека с комбинированным (импульсы – шум) сигналом возбуждения синтезирующего фильтра.

10. Повторить пункты 5-, 6-, 8-й для созданной модели. Подобрать оптимальное значение порога **r** классификации, обеспечивающее минимум искажений синтезированного сигнала.

Содержание отчета

1. Файлы *.mdl созданных моделей.
2. М-файл функции классификации (с комментариями).
3. «Осциллограммы» исходного и синтезированного речевых сигналов.
4. Выводы по работе.

Контрольные вопросы

1. Почему данные о мощности остатка предсказания сначала попадают в блок извлечения квадратного корня и лишь после этого используются в блоке перемножения данных?
2. Как можно объяснить отличие синтезированного сигнала от сигнала на входе речевого кодека?
3. К чему приведет исключение процедуры устранения постоянной составляющей из сигнала возбуждения синтезирующего фильтра?
4. Каким образом осуществляется классификация сегментов речевого сигнала на вокализованные и невокализованные?

Приложение. **Справочные данные по блокам, используемым для моделирования формирователя сигнала возбуждения синтезирующего фильтра**

Для реализации модели необходимо использовать следующие блоки пакета SIMULINK.

Pulse Generator – импульсный генератор. Блок находится в библиотеке **Sources** основного пакета SIMULINK. При задании его параме-

тров следует учесть, что период и длительность импульсов должны быть кратны периоду дискретизации.

Constant находится в библиотеке **Sources**, служит для формирования постоянной величины.

Math Function – в библиотеке Math Operations.

Product – в библиотеке Math Operations. Блок обеспечивает перемножение или деление входных сигналов. При задании параметров блока необходимо определить число входов и задать операцию поэлементного перемножения входов.

Gain – в библиотеке **Math Operations**. При задании коэффициента умножения блок работает как звено с данным коэффициентом передачи.

Sum – в библиотеке Math Operations. Блок обеспечивает суммирование или вычитание входных сигналов.

РЕКОМЕНДАТЕЛЬНЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Мартынов, Н. Н. Введение в MATLAB 6 : учеб. пособие / Н.Н. Мартынов. – М. : КУДИЦ-ОБРАЗ, 2002. – 352 с. – ISBN 5-93378-039-1.

2. Сергиенко, А. Б. Цифровая обработка сигналов : учеб. для вузов / А. Б. Сергиенко. – 2-е изд., перераб. и доп. – СПб. : Питер, 2006. – 751 с. – ISBN 5-469-00816-9.

3. Рабинер, Л. Р. Цифровая обработка речевых сигналов / Л. Р. Рабинер, Р. В. Шафер ; пер. с англ. ; под ред. проф. М. В. Назарова, Ю. Н. Прохорова. – М. : Радио и связь, 1981. – 496 с.

ОГЛАВЛЕНИЕ

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ОБРАБОТКИ СИГНАЛОВ
В СИСТЕМЕ MATLAB

Методические указания к лабораторным работам

Составитель

ЛЕВИН Евгений Калманович

Ответственный за выпуск – зав. кафедрой профессор О.Р. Никитин

Подписано в печать 08.04.11.
Формат 60×84/16. Усл. печ. л. 4,53. Тираж 80 экз.
Заказ
Владимирский государственный университет.
600000, Владимир, ул. Горького, 87.