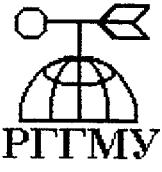
Министерство образования и науки Российской Федерации

Федеральное агентство по образованию
Государственное образовательное учреждение
высшего профессионального образования
Российский государственный гидрометеорологический университет

Е.А. Чернецова

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

«Введение в MATLAB»



Санкт-Петербург

2006

УДК 681.3.06(075.8)

Чернецова Е.А. Лабораторный практикум «Введение в MATLAB». – СПб.: изд. РГГМУ, 2006. – 88 с.

Рецензент: А.И.Яшин, проф.ГЭТУ

В лабораторный практикум «Введение в MATLAB» включены тринадцать практических работ, которые охватывают разделы вычислительной математики, линейной алгебры, обработки массивов и векторов, статистической обработки данных, моделирования и исследования сигналов различной физической природы. Практикум предназначен для студентов, обучающихся по специальностям «Морские информационные системы и оборудование» и «Информационная безопасность в телекоммуникационных системах».

Содержащиеся в практикуме задания, методические указания и рекомендации по их выполнению позволяют использовать MATLAB в качестве базового пособия при выполнении студентами лабораторных , курсовых и дипломных работ.

Практикум предназначен для студентов гидрометеорологического профиля, а также может быть полезным для всех желающих ознакомиться с приемами работы в пакете MATLAB.

Предисловие

Система MATLAB (сокращенное от MATrix LABoratory – МАТричная ЛАБоратория) разработана фирмой The MathWorks, Inc. (США, г.Нейтик, шт.Массачусетс) и является интерактивной системой для выполнения инженерных и научных расчетов, ориентированной на работу с массивами данных. Система использует математический сопроцессор и допускает возможность обращения к программам, написанным на языках FORTRAN, C, C++.

Система поддерживает выполнение операций с векторами, матрицами и массивами данных, реализует сингулярное и спектральное разложения, вычисление ранга и чисел обусловленности матриц, поддерживает работу с алгебраическими полиномами, решение нелинейных уравнений и задач оптимизации, интегрирование в квадратурах, решение дифференциальных и разностных уравнений, построение различных видов графиков, трехмерных поверхностей и линий уровня. В системе реализована удобная операционная среда, которая позволяет формулировать проблемы и получать решения в привычной математической форме, не прибегая к рутинному программированию.

Основным объектом системы MATLAB является прямоугольный числовой массив, допускающий комплексные элементы и ввод матриц без явного указания их размеров. Система позволяет решать многие вычислительные проблемы за значительно более короткое время, чем то, которое потребовалось бы для написания соответствующих программ на языках FORTRAN, BASIC или C.

Пакет MATLAB применяется для численных расчетов, макетирования алгоритмов, исследований в области автоматического управления, статистической обработки сигналов и процессов.

Для запуска пакета необходимо кнопкой «Пуск» выбрать в меню «Программы For student» программу MATLAB 6.5. Откроется рабочее окно,

в котором осуществляется ввод команд и файлов после приглашения «>>». При вводе данных в рабочем окне пакет MATLAB работает в режиме интерпретации вводимых команд и операторов, которые вводятся в ходе сеанса в командной строке, а MATLAB выполняет их немедленную обработку и выдает вычисленный результат.

Однако в MATLAB есть возможность обработки заранее подготовленной последовательности команд и операторов, записанной в виде файла. Для этого необходимо в меню «Файл» выбрать опцию «Открытие файла» и в открывшемся окне записать файлы-сценарии или процедуры-функции в виде mat-файла, пригодного для дальнейшей обработки в пакете MATLAB. При этом редактирование, копирование и другие операции с файлом производятся аналогично редактору WORD.

Для сохранения и последующего редактирования произведенных вычислений необходимо создать рабочую папку, в которую должны быть скопированы все произведенные вычисления.

Автор благодарит А.Д. Шишкина за полезные советы и рекомендации при составлении практикума.

Работа №1.

Решение уравнений и обработка матриц

Цель работы: приобретение навыков работы в рабочем окне пакета MATLAB: изучение способов формирования векторов и подматриц, вычисление элементарных математических и тригонометрических функций, вывод графиков функций в двумерной и трехмерной графике.

Формы задания переменных

В рабочем окне MATLAB можно задавать переменные, представляющие собой числа, векторы-строки (фактически представляющие собой матрицу размером $[1\times n]$), векторы-столбцы (фактически представляющие собой матрицу размером $[n\times 1]$) и матрицы размерностью $[n\times n]$.

Например, вводим переменную A=2. В рабочем окне выводится:

» A=2

A = 2

Если на конце выражения поставить знак «;» например:

A=2;

то выражение будет сразу записано в память без вывода на экран. Это позволяет сокращать время выполнения вычислительных программ и операций.

Числа в обозначении вектора-столбца отделяются друг от друга точкой с запятой, числа в обозначении вектора-строки отделяются друг от друга запятой, запятая в обозначении десятичных дробей обозначается точкой. При вводе векторов и матриц используются квадратные скобки.

Пример ввода в рабочем окне MATLAB вектора-столбца размерностью [5×1]: *A*=[1;12;34;4.768;5.13]

```
A =
    1.0000
   12.0000
   34.0000
    4.7680
    5.1300
```

Пример ввода в рабочем окне MATLAB вектора-строки размерностью [1×3]:

```
» A=[1.25,3.567,8.74]
A =
    1.2500    3.5670    8.7400
```

Пример ввода в рабочем окне MATLAB матрицы размерностью [3 x 3]:

```
» A=[1,2.3,5.2;2,4,8;9.3,3.7,2.56]
A =
    1.0000    2.3000    5.2000
    2.0000    4.0000    8.0000
    9.3000    3.7000    2.5600
```

Формат представления чисел на экране дисплея по желанию пользователя можно регулировать с помощью команды format. В системе MATLAB возможно задание нескольких выходных форматов. В данных примерах используется format short вывода чисел на экран (5 значащих десятичных цифр), можно установить format long (16 значащих десятичных цифр).

Формирование векторов и подматриц

Оператор «:» является очень полезным оператором языка MATLAB. Он применяется для формирования векторов и матриц или для выделения из них подвекторов, подматриц, подблоков массива.

Например, зададим вектор *A*, состоящий из цифр от 1 до 10 следующим образом:

```
A=1:10
A =
   1   2   3   4   5   6   7   8   9   10
```

Зададим вектор A, состоящий из цифр от 1 до 10 с шагом 2 следующим образом:

```
A=1:2:10
A =
   1   3   5   7   9
```

Зададим матрицу *A* следующим образом:

```
A =
   1   2   3   4   5
   2   3   4   6   7
   4   6   8   9   0
   1   3   6   9   9
   3   5   6   7   8
   1   2   3   4   6
```

и выделим из него подблок, включающий в себя строки со второй по четвертую столбцов с третьего по пятый следующим образом:

```
A(2:4,3:5)
ans =
   4   6   7
   8   9   0
   6   9   9
```

Оператор *A*(5, :) позволяет выделить все столбцы пятой строки массива *A*;

A(5,:)

ans =

3 5 6 7 8

Оператор *A*(:, 5) позволяет выделить все строки пятого столбца массива *A*;

A(:,5)

ans = 5 7 0 9 8 6

Операторы умножения «.*» и правого деления «./» с точкой используются при перемножении или делении массивов (каждое число первого массива умножается/делится на соответствующее число второго массива).

Например:

A =

1 2 3

4 5 6

1 4 8

B =

1 5 7

5 7 9

5 3 2

A.**B*

ans =

1 10 21

20 35 54

5 12 16

Операторы умножения и правого деления без точки применяются при перемножении или делении матриц по правилам линейной алгебры.

Например:

*A*B*

```
ans =  26   28   31
       59   73   85
       61   57   59
```

При применении оператора левого деления с точкой «.\» и оператора без точки «\» выполняется решение систем линейных уравнений вида *AX=B* по методу наименьших квадратов для матриц и векторов. Например:

A\B

```
ans =
   6.3333  -14.3333  -23.3333
  -9.6667   25.6667   41.6667
   4.6667  -10.6667  -17.6667
```

» *A.\B*

```
ans =
   1.0000   2.5000   2.3333
   1.2500   1.4000   1.5000
   5.0000   0.7500   0.2500
```

При транспонировании массива (операция «.'») строки просто заменяются столбцами:

A.'

```
ans =
   1   4   1
   2   5   4
   3   6   8
```

При транспонировании матрицы (операция «‘» без точки) результатом является транспонированная матрица, для комплексных чисел выполняется операция комплексного сопряжения.

Над матрицами можно производить разнообразные операции.

Например, с помощью команды *diag* возможно формировать или извлекать диагонали матрицы.

Функция *X=diag(v)* формирует квадратную матрицу *X* с вектором *v* на главной диагонали;

Функция *X=diag(v,k)* формирует квадратную матрицу *X* порядка *length(v)+abs(k)* с вектором *v* на *k*-той диагонали.

Функция *v=diag(X,k)* извлекает из матрицы *X* диагональ с номером *k;* при *k*>0 - это номер *k-й* верхней диагонали, при *k*<0 - это номер *k-й* нижней диагонали.

Функция *B=reshape(A,m,n)* возвращает матрицу размером [*m x n*], сформированную из элементов матрицы *A* путем их последовательной выборки по столбцам.

Элементарные графические функции системы MATLAB позволяют построить на экране и вывести на печать следующие типы графиков: линейный, логарифмический, полулогарифмический, полярный.

Для каждого графика можно задать заголовок, нанести обозначение осей и масштабную сетку.

Рассмотрим некоторые операторы для построения графиков в системе MATLAB.

Команда *plot(y)* строит график элементов одномерного массива *y* в зависимости от номера элемента; команда *plot(x,y)* соответствует построению обычной функции, когда одномерный массив *x* соответствует значениям аргумента, а одномерный массив *y* – значениям функции. Команда *plot(x,y,s)* позволяет выделить график функции, указав способ отображения линии, способ отображения точек, цвет линий и точек с помощью строковой переменной *s*, которая может включать до трех символов из следующей таблицы:

y	желтый	.	точка	-	непрерывная
m	фиолетовый	o	кружок	:	пунктирная
c	голубой	x	крестик	-.	штрихпунктирная

r	красный	+	плюс	--	штриховая
g	зеленый	*	звездочка		
b	синий	*s*	квадрат		
w	белый	*d*	ромбик		
k	черный	*v*	треугольник (нижний)		
		^	треугольник (верхний)		
		<	треугольник (левый)		
		>	треугольник (правый)		
		p	пентаграмма		
		h	гексаграмма.		

Команда *plot(x1,y1,s1,x2,y2,s2,.....)* позволяет объединить на одном графике несколько функций *y1(x1), y2(x2),.....*, определив для каждой из них свой способ отображения.

В системе MATLAB предусмотрено несколько команд и функций для построения трехмерных графиков. Значения элементов числового массива рассматриваются как z-координаты точек над плоскостью, определяемой координатами *x* и *y*. Возможно несколько способов соединения этих точек.

Первый из них – это соединение точек в сечении (функция *plot3*). Например, команда *plot3(x1,y1,z1,s1,x2,y2,z2,s2,...)* позволяет объединить на одном графике несколько функций *z1(x1,y1),z2(x2,y2),....*, определив для каждой из них свой способ изображения.

Команда *mesh(X,Y,Z)* выводит на экран сетчатую поверхность для значений массива *Z*, определенных на множестве значений массивов *X* и *Y*. Цвета узлов поверхности задаются командой *colormap(C),* где *C* - палитра.

Цвета:

hsv - выделение оттенками

hot – черно-красно-желто-белые цвета

gray – линейная шкала оттенков серого цвета

bone – шкала серых цветов с оттенками белого

copper – линейный медный цвет

pink – пастельные оттенки розового цвета

white – белый цвет

flag - попеременно красный, белый, синий и черный цвета

lines – линейное чередование цветов

colorcube – чередование серого, красного, зеленого и белого

vga - 16 цветов

jet – вариант *hsv*

prism – использование 6 цветов

cool – оттенки синего и фиолетового

autumn – оттенки красного и желтого

spring - оттенки фиолетового и желтого

winter - оттенки синего и зеленого

summer - оттенки зеленого и желтого

Команда *surf(X,Y,Z)* выводит на экран сетчатую поверхность для значений массива *Z*, определенных на множестве значений массивов *X* и *Y*.

Для вращения трехмерного изображения используется команда *rotate3d*.

Для того чтобы построить несколько графиков и разместить каждый график в своем окне *figure*, нужно после построения первого графика выполнить команды: *file – New – Figure*, тогда будет создано новое окно, в котором будет изображен новый график после введения в рабочем окне соответствующих команд.

Для сохранения рабочей области MATLAB используется команда *save*. Команда *save имя файла* выгружает все переменные из рабочей области в

двоичный файл с именем *имя файла.mat*. Если имя файла отсутствует, то ему присваивается специальное имя *matlab.mat*. Команда *save имя файла A B C* выгружает переменные *A, B, C*. Для очистки рабочей области используется команда *clear all*. Для очистки экрана используется команда *clc*. Для загрузки переменных в рабочую область используется команда *load*. Команда *load* без параметров считывает данные из файла *matlab.mat*, если он создан командой *save*. Команда *load имя файла* загружает переменные из mat-файла имя файла.mat. После загрузки переменных в рабочую область бывает необходимо вывести список переменных текущей рабочей области. Эта операция выполняется с помощью команды *who*.

Задание для выполнения работы

1. В рабочем окне MATLAB ввести матрицу *A*.

2. Выделить из матрицы *A* подматрицу *B*.

3. Выделить из матрицы *A* подматрицу *C*.

4. Извлечь из матрицы *A* диагональ *D*.

5. Умножить матрицу *B* на матрицу *C* с точкой и без точки.

6. Разделить матрицу *B* на матрицу *C* левым делением с точкой и без точки.

7. Разделить матрицу *B* на матрицу *C* правым делением с точкой и без точки.

8. Транспонировать матрицу *A*.

9. Построить с помощью функции *plot* вектор, состоящий из всех строк третьего столбца матрицы A.

10. Построить трехмерный график матрицы A в зависимости от номера элемента по строкам и столбцам с использованием команд *mesh* и *surf* с использованием различной цветовой палитры и с возможностью поворачивать изображение под разными ракурсами.

11. Сохранить переменные *A,B,C,D* в mat-файле.

12. Очистить рабочую область. Очистить экран. Загрузить сохраненные переменные в рабочую область. Вывести список переменных.

Варианты заданий для работы

Вариант 1:

A =

0,0063	0,038	0,67	0,0054	0,0065	0,0049
0,57	0,05	0,0784	0,0046	0,29	0,003
0,0044	0,575	0,0064	0,698	0,543	0,0001
0,997	0,4567	0,002	0,578	0,445	0,0254
0,02	0,0446	0,268	0,077	0,0057	0,0054
0,0089	0,0987	0,0057	0,798	0,0965	0,0361

B =

0,05	0,0784	0,0046	0,29
0,575	0,0064	0,698	0,543
0,4567	0,002	0,578	0,445
0,0446	0,268	0,077	0,0057

C=

0,0063	0,038	0,67	0,0054
0,57	0,05	0,0784	0,0046
0,0044	0,575	0,0064	0,698
0,997	0,4567	0,002	0,578

D=0,997 0,0446 0,0057

Вариант 2:

$A=$

0.8762	0.7726	0.7582	0.8002	0.7962	0.6665	0.9865
0.3556	0.6205	0.9317	0.8679	0.8710	0.7559	0.9688
0.9060	0.9906	0.9514	0.9894	0.9640	0.8660	0.8546
0.9792	0.9989	0.9916	0.9457	0.9376	0.9376	0.9182
0.9972	0.9634	0.9278	0.8655	0.9988	0.8666	0.8898
0.9929	0.9995	1.0000	0.9859	0.9325	0.9996	0.4691
0.7826	0.6121	0.9962	0.7631	0.8936	0.9744	0.5783

$B=$

0.9514	0.9894	0.9640	0.8660	0.8546
0.9916	0.9457	0.9376	0.9376	0.9182
0.9278	0.8655	0.9988	0.8666	0.8898

$C=$

0.8762	0.7726	0.7582	0.8002	0.7962
0.3556	0.6205	0.9317	0.8679	0.8710
0.9060	0.9906	0.9514	0.9894	0.9640

$D=$

0.7726	0.9317	0.9894	0.9376	0.8666	0.4691

Задание 3:

$A=$

9.5272	2.1815	4.5297	5.2380	2.3439	1.4659	11.2004

2.8767 4.7603 4.2683 2.6028 2.6997 0.5291 5.0222

2.0590 4.9244 8.7242 2.4919 2.7565 8.6705 4.8267

1.6392 3.0574 7.7370 3.2249 2.4287 4.3578 8.9045

3.3267 10.1745 3.2031 9.7044 3.3384 22.8564 7.6963

1.9371 3.2575 6.9681 7.3445 5.6299 19.9665 1.1599

3.0202 3.6819 5.4846 1.1845 1.9250 4.5214 9.1030

B=

3.0574 7.7370 3.2249 2.4287 4.3578

10.1745 3.2031 9.7044 3.3384 22.8564

3.2575 6.9681 7.3445 5.6299 19.9665

3.6819 5.4846 1.1845 1.9250 4.5214

C=

2.1815 4.5297 5.2380 2.3439 1.4659

4.7603 4.2683 2.6028 2.6997 0.5291

4.9244 8.7242 2.4919 2.7565 8.6705

3.0574 7.7370 3.2249 2.4287 4.3578

D=

3.3267 3.2575 5.4846

Вариант 4:

A=

0.6934 3.4302 3.2723 0.0546 12.2107 8.9556 8.7879

1.6332 1.5901 2.4343 4.1119 9.1181 0.5733 5.2615

0.5854 1.4918 2.3426 2.7809 13.4954 29.4956 7.2479

1.6579 4.3528 0.3552 1.3945 11.7359 3.1786 2.3811

5.7940 1.3223 1.5670 4.8091 4.6988 3.6736 0.3824

6.6318 4.5172 1.8277 3.8735 1.4285 2.5523 6.4176

4.1101 9.5744 0.8614 10.5800 7.6150 6.9610 3.4473

B=

0.0546 12.2107 8.9556 8.7879

4.1119 9.1181 0.5733 5.2615

2.7809 13.4954 29.4956 7.2479

1.3945 11.7359 3.1786 2.3811

C=

1.6579 4.3528 0.3552 1.3945

5.7940 1.3223 1.5670 4.8091

6.6318 4.5172 1.8277 3.8735

4.1101 9.5744 0.8614 10.5800

D=

0.6934 1.5901 2.3426 1.3945 4.6988 2.5523 3.4473

Вариант 5:

A=

0.0081 7.6399 1.3506 3.7951 0.2055 0.3593

1.0536 22.3808 0.0565 29.7027 0.2646 0.2551

1.3466 304.4029 24.9269 0.7303 0.5680 2.1895

1.9071 0.0532 3.9344 0.0059 0.1992 16.7611

4.4069 39.8499 1.0821 0.7376 0.1972 0.0173

7.8419 0.5524 9.0982 0.2388 0.0941 2.0778

1.1999 0.0007 3.4331 0.0017 1.9596 0.0136

B=

1.3466 304.4029 24.9269 0.7303

1.9071 0.0532 3.9344 0.0059

4.4069 39.8499 1.0821 0.7376

C=

3.9344 0.0059 0.1992 16.7611

1.0821 0.7376 0.1972 0.0173

9.0982 0.2388 0.0941 2.0778

D=

1.3506 29.7027 0.5680 16.7611

Вариант 6:

A=

5.5577 3.1121 8.5322 1.2649 0.3179 2.3806 0.6724

2.6287 0.8913 1.8364 30.3882 3.4777 3.3664 6.0280

7.0923 1.2885 2.3576 6.0922 4.1388 2.9021 8.1857

2.3959 0.1863 0.4417 2.1104 2.5111 0.3615 2.9413

0.4207 0.5489 0.4693 9.7843 0.5146 0.8183 2.9543

2.8124 2.2652 2.2507 4.4427 5.0705 0.3707 2.1769

4.1257 1.3867 0.7189 3.3453 2.0308 2.3465 0.0931

B=

0.4207 0.5489 0.4693

2.8124 2.2652 2.2507

4.1257 1.3867 0.7189

C=

0.3179 2.3806 0.6724

3.4777 3.3664 6.0280

4.1388 2.9021 8.1857

D=

0.4207 2.2652 0.7189

Задание 7:

A=

-1.5250 0.8125 0.4701 0.8586 -0.7312 -0.3821 -0.1118

0.6926 0.2615 -0.9401 -0.4834 -0.5043 -0.5334 -0.7713

0.5721 0.3278 0.2788 1.3160 0.2772 0.5896 -1.2083

1.2352 -0.1324 -2.6244 -2.2499 -0.9274 0.1167 -0.5862

0.2040 -0.7154 0.9599 0.1525 -0.0132 1.5805 -0.0282

-0.7866 0.8675 -1.1172 -0.4035 0.1472 1.6554 0.1948

-0.4514 1.8440 -0.2479 -0.0799 0.6316 0.3078 -1.1212

B=

-1.5250 0.8125 0.4701

0.6926 0.2615 -0.9401

0.5721 0.3278 0.2788

C=

0.9599 0.1525 -0.0132

-1.1172 -0.4035 0.1472

-0.2479 -0.0799 0.6316

D=

0.8586 -0.5043 0.5896 -0.5862

Вариант 8:

A=

7.8372 9.8118 9.3180 3.3191 2.7952 13.3430 4.7176

1.6721 11.6365 10.5697 13.5716 12.8557 15.9542 17.0757

10.6267 10.8732 15.3338 18.1178 8.0006 3.9877 5.9745

11.4384 9.0665 10.2964 6.5411 13.4500 9.9011 12.6437

4.2676 13.6290 9.5218 14.2900 14.0781 9.2164 11.0966

15.9546 7.0584 5.8383 16.2700 13.5595 1.9796 5.3905

15.9458 20.9159 11.4721 2.0314 16.4512 11.2865 -0.8534

B=

2.7952 13.3430 4.7176

12.8557 15.9542 17.0757

8.0006 3.9877 5.9745

13.4500 9.9011 12.6437

D=

2.7952 15.9542 5.9745

C=

9.8118 9.3180 3.3191

11.6365 10.5697 13.5716

10.8732 15.3338 18.1178

9.0665 10.2964 6.5411

Вариант 9:

A=

2.4715 1.3398 7.1498 3.3309 4.0694 4.0688 1.0263

3.1349 3.0280 1.2954 0.8586 5.1534 1.1241 3.1686

4.0119 3.0124 4.5339 2.2718 1.0535 5.3689 2.2468

6.7803 2.8451 2.1964 3.6142 2.6910 4.0757 4.6596

5.8766 1.1233 3.8031 4.6361 3.4043 3.8412 3.2876

2.3595 3.6832 4.0496 1.0548 2.9069 3.8958 3.4562

2.1662 4.5635 2.1771 3.6568 4.2041 1.1899 1.8600

B=

2.4715 1.3398 7.1498 3.3309

3.1349 3.0280 1.2954 0.8586

4.0119 3.0124 4.5339 2.2718

C=

4.6361 3.4043 3.8412 3.2876

1.0548 2.9069 3.8958 3.4562

3.6568 4.2041 1.1899 1.8600

D=

6.7803 1.1233 4.0496 3.6568

Вариант 10:

A=

0.5656	0.5770	0.7293	0.3464	1.8473	1.3967	1.0284
0.8998	1.0439	0.3127	0.7245	0.7060	0.9616	0.5749
1.8133	0.3670	0.5402	0.7032	1.0038	1.5598	1.2747
0.5723	0.7815	1.6952	0.6011	0.6761	3.1729	0.9975
1.3739	1.2599	0.9450	0.9130	1.3411	1.2999	0.8710
0.7403	0.8517	1.2088	2.1394	0.8820	0.9941	1.8931
1.3173	1.8557	1.6034	0.9810	1.2713	1.5787	2.5388

D=

0.8998	0.3670	1.6952	0.9130	0.8820	1.5787

B=

1.8133	0.3670	0.5402	0.7032
0.5723	0.7815	1.6952	0.6011
1.3739	1.2599	0.9450	0.9130
0.7403	0.8517	1.2088	2.1394

C=

0.3127	0.7245	0.7060	0.9616
0.5402	0.7032	1.0038	1.5598
1.6952	0.6011	0.6761	3.1729
0.9450	0.9130	1.3411	1.2999

Работа № 2

Элементы программирования в MATLAB

Цель работы: приобретение навыков работы с *m*-файлами; создание программ-функций, работа с циклами и условными операторами.

При использовании ввода данных в рабочем окне MATLAB работает в режиме интерпретации команд и операторов: они вводятся в ходе сеанса в командной строке, а MATLAB выполняет их немедленную обработку и выдает вычисленный результат. Примером такого режима работы являются команды решения системы уравнений, нахождения пределов функций и обработки матриц.

Команда *solve* позволяет решать уравнения и системы уравнений. Так, например, функция *g=solve('eq')* решает уравнение, заданное в виде символьного, либо строкового выражения с указанием или без указания знака равенства. Если знак равенства не указан, то предполагается уравнение вида *eq=0*. Переменная, относительно которой ищется решение, если она не указана явно, определяется автоматически с помощью функции *findsym*. Функция *g=solve('eq',var)* решает уравнение относительно переменной *var*. Функции *g=solve('eq1','eq2',....,'eqn')* и *g=solve('eq1','eq2',...'eqn',var1,var2,...varn)* решают системы уравнений относительно *n* переменных. Причем для систем уравнений с одним выходным аргументом решение возвращается в виде массива записей. Для того, чтобы решения вернулись упорядоченными по именам переменных, нужно задать число выходных аргументов в квадратных скобках, равное числу переменных.

Фундаментальная идея нахождения предела функции заключается в достижении переменной определенного значения. Вспомните, что определение производной дается пределом,

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

при условии, что он существует. Symbolic Math Toolbox позволяет вычислить пределы функции впрямую. Команды

```
syms h n x
limit( (cos(x+h) - cos(x))/h,h,0 ),
```

которые возвращают

```
ans =
-sin(x)
```

и

```
limit( (1 + x/n)^n,n,inf ),
```

которые возвращают

```
ans =
exp(x),
```

иллюстрируют два наиболее важных предела в математике: производную (случай с *cos x*) и экспоненциальную функцию. Поскольку многие пределы являются двусторонними

$$\lim_{x \to a} f(x)$$

,

т.е. приближениями слева и справа к *a*, пределы в особенных точках $f(x)$ таковыми не являются. Поэтому три предела

$$\lim_{x \to 0} \frac{1}{x}, \lim_{x \to 0-} \frac{1}{x}, \text{and } \lim_{x \to 0+} \frac{1}{x}$$

соответствуют трем достижимым результатам: неопределенному, $-\infty$ и ∞ соответственно.

В случае неопределенного результата, Symbolic Math Toolbox возвращает *NaN* (не число). Команда

```
limit(1/x,x,0)
```

или

limit(1/x)

возвращают

ans =

NaN.

Команда

limit(1/x,x,0,'left')

возвращает

ans =

-inf

тогда как команда

limit(1/x,x,0,'right')

возвращает

ans =

inf.

Примечание: Заметьте, что по умолчанию *limit(f)* является тем же самым, что *limit(f,x,0).*

Опции команды *limit* представлены в таблице. Заметьте, что функция *f* является функцией символьного объекта *x.*

Mathematical Operation	MATLAB Command
$\lim_{x \to 0} f(x)$	`limit(f)`
$\lim_{x \to a} f(x)$	`limit(f,x,a)` or `limit(f,a)`
$\lim_{x \to a-} f(x)$	`limit(f,x,a,'left')`
$\lim_{x \to a+} f(x)$	`limit(f,x,a,'right')`

Система MATLAB работает с данными, представленными в различных форматах. Наиболее часто используются данные, представленные в так называемых ASCII-кодах (American Standard Code for Information Interchange) и в двоичных кодах. Файлы данных,

представленных в ASCII-кодах, должны иметь расширение *.yyy* и не содержать никаких других символов, кроме цифр и точки. Они создаются в текстовом редакторе (например, в Norton Commander с помощью команды *Shift F4*) и загружаются в командное окно MATLAB с помощью команды *load *.yyy* (* обозначает имя файла).

Иногда бывает необходимо преобразовать размеры матрицы, например, заменить строки столбцами, а столбцы строками. Это можно сделать с помощью команды *B=reshape(A,m,n)*, которая возвращает матрицу размером *m x n* , сформированную из элементов матрицы *A* путем их последовательной выборки по столбцам.

Однако в пакете MATLAB есть возможность обработки заранее подготовленной последовательности команд и операторов, записанной в виде файла.

M-файлы, содержащие команды и операторы MATLAB, разделяются на файлы-сценарии и процедуры-функции. Программирование осуществляется в окне *MATLAB Editor/Debugger,* которое вызывается последовательностью команд меню: *File-New-M-File*

Если первая строка *M*-файла начинается с определения *function[список выходных переменных]=имя функции(список входных переменных),*то этот файл представляет собой *M*-функцию. Она может быть сохранена в текущем каталоге с помощью последовательности команд меню: *File-Save* или *File-Save As*. Для предотвращения путаницы рекомендуется сохранять такой *M*-файл под именем функции, но в принципе имена *M*-файла и функции могут отличаться.

Например, *function[G]=max1(A,k,s).*

Стандартная структура *M*-функции должна быть следующей:

- Первая строка объявляет имя функции, ее входные и выходные аргументы.
- Для обозначения строк комментариев служит символ «%».

- Переменные в теле функции являются локальными, т.е. значения таких переменных можно изменять и переприсваивать, так как они изолированы от переменных других функций.

В отличие от универсальных языков программирования переменная цикла в языке MATLAB является массивом.

Рассмотрим операторы циклов, применяемые в MATLAB.

А) оператор цикла с определенным числом операций

for v=<выражение-массив>

<операторы>

end.

На практике в качестве выражения наиболее часто применяются линейные конструкции вида *m:n* или *a:<шаг>:b*.

Б) оператор цикла с неопределенным числом операций

while<логическое выражение>

<операторы>

end.

Условное выражение вида: *if....else....end*

if<логическое выражение>
<операторы>
end

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

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

Логическое выражение имеет форму:

выражение<оператор отношения>выражение,

где оператор отношения: ==, <=, >=, <, >, ~.

Задание для выполнения работы

1. С помощью команды *solve,* сформированной в командной строке, решить систему уравнений (вариант задается преподавателем).

1. $\begin{cases} 3x+y=2(x-y) \\ (3x+y)^2+2(x-y)^2=96 \end{cases}$

2. $\begin{cases} x-3y+2=(x+y)^2 \\ (x+y)^2+(x-3y)^2=8 \end{cases}$

3. $\begin{cases} x^3-y^3=124 \\ x^2+xy+y^2=31 \end{cases}$

4. $\begin{cases} x^2+y^2=5 \\ y^6+y^4x^2=80 \end{cases}$

5. $\begin{cases} x^3+y=1 \\ y^3-4y^2+4y+x^6=1 \end{cases}$

6. $\begin{cases} x^2+2y^2-3x-5y=-4 \\ -2x^2-6y^2+2x+15y=6 \end{cases}$

7. $\begin{cases} 2x^2-5xy+3x-2y=2 \\ 5xy-2x^2+7x-8y=-22 \end{cases}$

8. $\begin{cases} 12x^2+2y^2-6x+5y=3 \\ 18x^2+3y^2-6x+8y=7 \end{cases}$

9. $\begin{cases} 9y^2+6xy-4x-9y+2=0 \\ 27y^2+3xy-2x-42y+16=0 \end{cases}$

10. $\begin{cases} x^3+y^3-xy+2x+2y=5 \\ x^3+y^3+2xy-x-y=2 \end{cases}$

Корни уравнений могут быть как действительными, так и комплексными.

2. Найти пределы функций (вариант задается преподавателем).

Задачи
Найти пределы:

1. $\lim\limits_{n\to\infty}\left(1+\frac{2}{n}\right)^n$.
2. $\lim\limits_{x\to 0}(1-3x)^{\frac{1}{x}}$.
3. $\lim\limits_{x\to 0}\frac{\ln(1+4x)}{x}$.
4. $\lim\limits_{x\to\infty} x\,[\ln(x+1)-\ln x]$.
5. $\lim\limits_{x\to\infty}\left(\frac{x+1}{x-3}\right)^x$.
6. $\lim\limits_{x\to\infty}\left(\frac{x+n}{x+m}\right)^x$.
7. $\lim\limits_{x\to\infty}\frac{\sin\frac{x}{5}}{x}$.
8. $\lim\limits_{x\to 0} x\sin\frac{1}{x}$.
9. $\lim\limits_{x\to 0}\frac{\operatorname{arctg} x}{x}$.
10. $\lim\limits_{x\to 0}\frac{\sin ax}{\operatorname{tg} bx}$.
11. $\lim\limits_{x\to 0}\frac{\sin^3\frac{x}{2}}{x^3}$.
12. $\lim\limits_{x\to 0}\frac{\sin x}{\sqrt{x+4}-2}$.
13. $\lim\limits_{x\to 1}\frac{\sin(x-1)}{x^3-1}$.
14. $\lim\limits_{x\to 0}\frac{1-\cos x-\operatorname{tg}^2 x}{x\sin x}$.
15. $\lim\limits_{x\to 0}\left(\frac{\sin\frac{x}{2}}{x}\right)^{x+3}$.
16. $\lim\limits_{x\to\infty}\left(\frac{2x+1}{4x-3}\right)^x$.

17. $\lim_{x\to 0}(\cos x)^{\frac{1}{x^2}}$. 18. $\lim_{x\to\frac{\pi}{2}}(\sin x)^{\operatorname{tg}^2 x}$.

Задачи

Найти пределы:

1. $\lim_{x\to+\infty}(\sqrt{x^2+5x+4}-\sqrt{x^2+x})$.
2. $\lim_{x\to -4}\left(\frac{1}{x+4}-\frac{8}{16-x^2}\right)$.
3. $\lim_{x\to 0}\left(\frac{\cos x}{\sin^2 x}-\operatorname{ctg}^2 x\right)$.
4. $\lim_{x\to 0}(\operatorname{ctg} x-\operatorname{cosec} x)$.
5. $\lim_{x\to 0} x\operatorname{ctg}\frac{\pi}{2}x$.
6. $\lim_{x\to\frac{\pi}{2}}\left(x-\frac{\pi}{2}\right)\operatorname{tg} x$.
7. $\lim_{x\to 0}\frac{e^{-x}-1}{3x}$.
8. $\lim_{m\to\infty} m[\ln(m+4)-\ln m]$.
9. $\lim_{x\to 0}\frac{\sqrt{x+9}-3}{\sin 6x}$.
10. $\lim_{n\to\infty}\frac{5-2^n}{5+2^{n+1}}$.
11. $\lim_{x\to 0}(\cos 2x)^{\operatorname{ctg}^2 2x}$.
12. $\lim_{x\to 0}\frac{1+x^2-\cos x}{\sin^2 x}$.

3. Используя конструкции условных операторов и операторов цикла, сформировать программу-функцию обработки матрицы. Для формирования процедур-функций в пакете MATLAB необходимо открыть окно *MATLAB Editor\Debugger*. В этом окне пишется последовательность команд, затем созданному *m*-файлу присваивается имя и он сохраняется в текущем каталоге командой *File Save* или в каком-либо другом каталоге командой *File Save As*, который выбирается в полосе прокрутки *Current Directory* (см. рис. 1).

Внимание! Имя файла и имя функции должны быть одинаковыми.

Для запуска созданной функции необходимо вызвать ее в рабочее окно MATLAB, изображенное на рис. 1.1

Если файл-функция была создана в рабочем каталоге MATLAB, то формат команды:

[список выходных переменных]=имя функции (список входных переменных),например, [*G*]=max1(*A*,*k*,*s*).

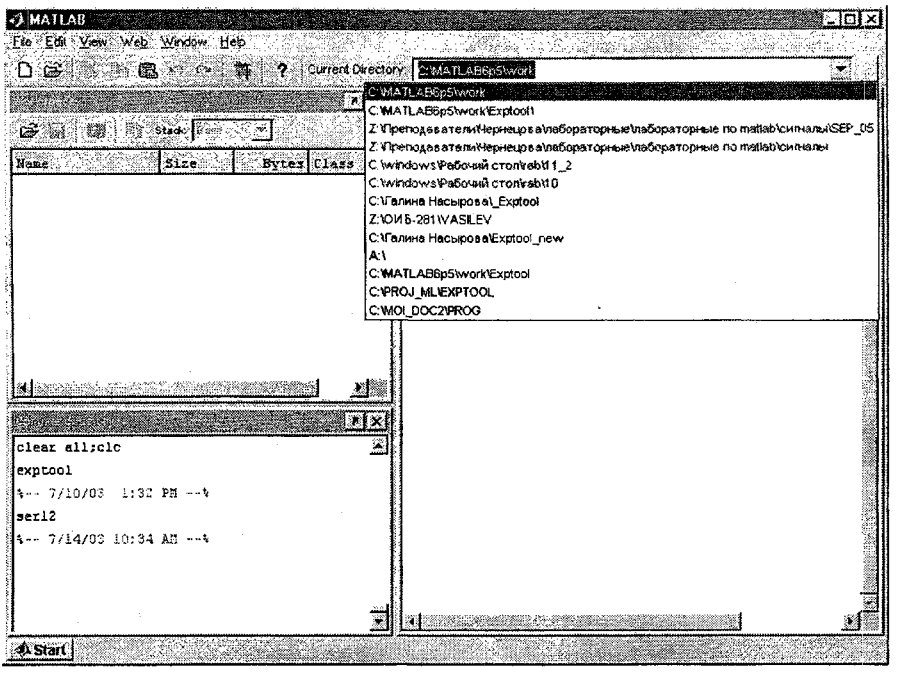


Рис. 1.1. Выбор рабочего каталога в окне MATLAB

4. Записать соответствующую по номеру задания матрицу A (см. работу № 1) в ASCII-файл, загрузить его в рабочую область и сформировать из матрицы A такую матрицу, в которой на главной диагонали стоят единицы, на соседней верхней диагонали стоят двойки, на соседней нижней диагонали стоят тройки, а несколько оставшихся элементов матрицы A заменить полученными значениями корней уравнений системы. Преобразовать размеры матрицы с помощью команды *reshape*.

Работа № 3

Графическая обработка экспериментальных данных

Цель работы: приобретение навыков обработки и графического представления файлов данных, записанных в ASCII – кодах.

Рассмотрим экспериментально полученные с помощью различных датчиков сигналы, записанные в ASCII-кодах.

Загрузим сигнал в рабочую область с помощью команды *load,* например, *load sig10.yyy.* (Вид сигнала и вариант файла задаются преподавателем). При загрузке файла необходимо указать путь доступа к файлу в окне «Текущий каталог» (см. рис.1.1).

Так как сигналы довольно продолжительны во времени, при построении графиков имеет смысл нанести координатную сетку командой *grid on*, а также включить масштабирование командой *zoom on,* чтобы иметь возможность выделять для просмотра различные участки сигнала. Так как запись сигнала может быть довольно большой (например, 80 000 отсчетов), то для анализа удобно выделить некоторое временное окно. Для того чтобы ось отсчетов времени была промаркирована в секундах, воспользуемся формулой перевода вектора отсчетов во временной вектор по формуле: $t = N/f$, где t –временной вектор в секундах; N – вектор отсчетов в записи; f – частота дискретизации (1/с);

Примечание. В файле vm19_2 частота дискретизации $f_д = 256\ c^{-1}$, во всех остальных файлах частота дискретизации $f_д = 2048\ c^{-1}$.

Для того чтобы определить длину записи, воспользуемся командами *length(x)* , возвращающей количество элементов вектора *x*, или *[m,n]=size(x),* возвращающей число строк и столбцов массива *x* в виде двух переменных.

При выводе графиков существует необходимость буквенных обозначений на них. Так, например, для вывода заголовка графика используется команда *title('text')*. Для подписей осей используем команды *xlabel(str1), ylabel(str2),* где *str1* и *str2* – строковые переменные, получаемые с использованием команды *sprintf*. Например, команда

Str=sprintf('The array is %dx%d.',2,3)

выдает сообщение: The array is 2x3., а команда

str=sprintf('%0.5g',(1+sqrt(5))/2)

выдает сообщение: 1.618.

Рассмотрим некоторые процедуры, применяемые при обработке сигналов.

Выделение огибающей сигнала производится с помощью последовательного выполнения команд `hilbert` и *abs*. Команда `hilbert` возвращает комплексные значения сигнала (амплитудную и фазовую составляющую), а команда *abs* выделяет его реальную часть (амплитудную).

Вычисление спектра сигнала производится выделением абсолютного значения дискретного преобразования Фурье (команда *fft*) от ковариационной матрицы массива (команда *xcov*). Знание спектра сигнала позволяет выделить в нем частоты, на которые приходится большая часть энергии сигнала (выделение максимума спектральной плотности мощности).

Вычисление автокорреляционной функции сигнала s производится с помощью команды *xcov(s)* и служит для определения линейной зависимости между отсчетами сигнала в выбранные моменты времени.

Вычисление взаимнокорреляционной функции сигналов s1 и s2 производится с помощью команды *xcov(s1,s2)* и позволяет получить представление о линейной зависимости между значениями двух случайных функций в выбранные моменты времени.

Вычисление взаимного спектра сигналов s1 и s2 производится выделением абсолютного значения дискретного преобразования Фурье

(команда *fft*) от взаимнокорреляционной функции *xcov(s1,s2)*. Знание взаимного спектра помогает выделить одинаковые частоты в двух сигналах.

Фильтрация сигнала применяется для очистки сигнала от нежелательных частот.

Рассмотрим фильтр нижних частот (ФНЧ). Он пропускает все частоты, лежащие ниже частоты среза $f_{среза}$ и отфильтровывает (убирает из сигнала) все частоты, лежащие выше $f_{среза}$. Порядок фильтра N указывает на количество используемых аппаратных фильтрующих цепочек. Команда *[B,A] = butter(N,Wn)* вычисляет коэффициенты фильтра. Параметр $Wn = f_{среза}/(f_д/2)$. Команда *y=filtfilt(B,A,X)* выполняет фильтрацию заданных в массиве X данных с минимальными фазовыми искажениями.

Для работы указанных процедур и выводов графиков целесообразно выбрать небольшой, наиболее информативный интервал по оси времени.

В MATLAB существует способ записи данных в файл, который затем может быть открыт в пакете Excel. Синтаксис команды:

wk1write('filename',M)

записывает матрицу M в формате Lotus 1-2-3 в файл с именем *filename.wk1*.

Задание для выполнения работы

Загрузить ASCII-файл в рабочую область, вывести график сигнала, нанести сетку, включить масштаб и выбрать окно для анализа в отсчетах. Вывести график выбранного сигнала с использованием временной шкалы (секунды) и шкалы амплитуды в вольтах или делениях размерности АЦП (в соответствии с сигналом), с подписями заголовка и названий осей, нанести сетку и включить масштаб. Оформить в виде процедуры-функции вычисление огибающей сигнала и спектра для выбранного отрезка сигнала. Вывести графики огибающей и спектра сигнала в matlab. Сохранить данные о спектре в файле формата Lotus, открыть файл в Excel и построить график спектра с использованием инструментария Excel.

Работа № 4

Обработка статистических данных

Цель работы: Обработка временных рядов и графическое представление статистических характеристик.

При обработке временных рядов или других данных для их анализа используют различные статистические характеристики. Наиболее известные из них: гистограммы, плотности распределения вероятностей (ПРВ), корреляционные и ковариационные функции.

Рассмотрим ASCII-файл *vm19_2*, в котором записан вектор $\vec{Y}_n = (y_1, y_2, ..., y_n)^T$. Составляющие вектора y_i, $i=1,...,n$ представляют собой измеренные значения сигнала на выходе какого-либо датчика в отдельные (i-е) моменты времени и являются случайными величинами. Исследуем некоторые статистические характеристики экспериментально полученного временного ряда. Для этого нужно получить плотности распределения вероятностей (или построить гистограммы) для различных параметров сигнала, например, для мгновенных значений сигнала, для его огибающей и т.д.

Для построения гистограмм в MATLAB существует функция *N=hist(y,x),* возвращающая распределение вектора y по интервалам разбиения, количество которых определяет переменная x. Пример построения гистограммы приведен на рис. 4.1

Для вывода гистограммы необходимо использовать рисующую функцию *bar(N)*.

```
ans =
  1
>> bar(mn)
>> hold on
>> t=20;
>> Y = NORMPDF(1:20,11,2.5);
>> plot(Y,'r')
```

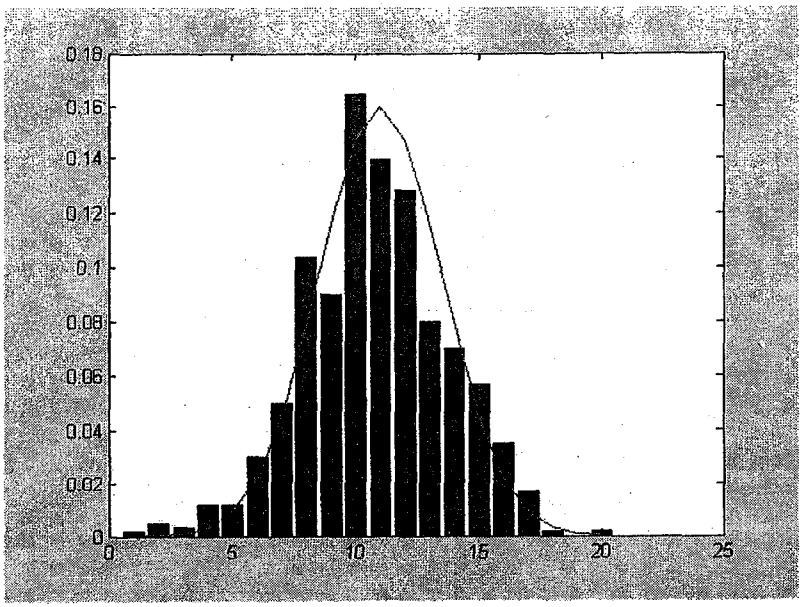


Рис.4.1.Пример построения гистограммы в MATLAB

Большой интерес представляет аппроксимация полученной гистограммы известными законами распределения. Так как гистограмма представляет собой плотность распределения вероятности, воспользуемся командой из *matlab Statistic Toolbox*, которая позволяет генерировать ПРВ разных законов распределения при заданных параметрах. Например, гауссовское (нормальное) распределение вероятностей генерируется командой

Y=normpdf(Y,MU,SIGMA).

Команда возвращает нормальную ПРВ значений, задаваемых вектором *Y*, со средним значением, задаваемым переменной MU, и стандартным отклонением, задаваемым параметром SIGMA.

Точность аппроксимации определяется по критерию *x*- квадрат:

$$\chi^2(A) = \sum_{m=1}^{M} \frac{[\eta_m - q_m]^2}{q_m},$$

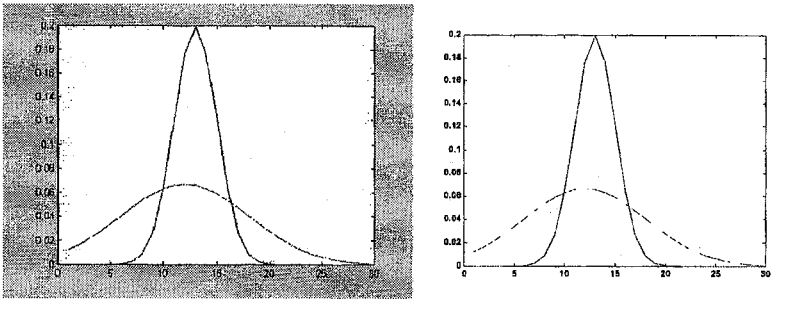
где A - исследуемый параметр;

η_m - сравниваемое распределение вероятностей;

q_m - распределение вероятностей образца.

Для признания аппроксимации удовлетворительная точность аппроксимации (выражающая отклонение сравниваемой ПРВ от образцовой) не должна превышать 20-30%, т.е. должно выполняться условие $\chi^2(A) \le 0{,}3$.

При изображении нескольких графиков в одном окне часто бывает удобно пользоваться командой *hold on*, которая позволяет удержать первый график в графическом окне без замены его на другой, таким образом, первое изображение просто наслаивается на другое. Например, команда *plot(b,'r');hold on; plot(a,'b')* строит график, представленный на рис. 4.2.



а) б)

Рис.4.2.Формы представления графиков

Примечание. Для экспортирования фигур из MATLAB в Word используйте последовательность команд (панель инструментов в *figure)*: *Edit – Copy Figure*, предварительно, установив последовательностью команд: *File – Preferences – Copying Options* (панель инструментов командного окна) формат графического вывода: либо *Windows Bitmap* (рис.4.2,а), либо *Windows Metafile* (рис. 4.2,б).

Задание для выполнения работы

Загрузить в рабочую область файл экспериментальных данных и построить гистограмму мгновенных значений для отрезка записи, на котором зафиксирован сигнал с учетом правила 10 точек. Преобразовать гистограмму в ПРВ. Построить ПРВ известного закона распределения с использованием команд из *matlab Statistic Toolbox* и аппроксимировать ею экспериментальную ПРВ по критерию х-квадрат, добившись приемлемой точности аппроксимации. Вывести график исследуемой и образцовой ПРВ с использованием команды *hold on*. Экспортировать график в Word с использованием разных форматов графического вывода.

Работа № 5

Назначение и область применения вейвлет-анализа

Цель работы: исследование сигналов с помощью вейвлет-преобразования

Термин "вейвлеты" используется в математике для обозначения некоторого ортонормированного базиса, обеспечивающего хорошие свойства приближения.

В настоящее время вейвлет-анализ является одним из наиболее мощных и при этом гибких средств исследования данных. Помимо возможностей сжатия и фильтрации данных анализ в базисе вейвлет-функций позволяет решать задачи идентификации, моделирования, аппроксимации стационарных и нестационарных процессов, исследовать вопросы наличия разрывов в производных, осуществлять поиск точек склеивания данных, удалять в данных тренд, отыскивать признаки фрактальности информации. В основе подобных возможностей вейвлет-анализа лежит природа его многомасштабности.

Вейвлет-функции обладают свойством частотно-временной локализации сигналов и быстрыми вычислительными алгоритмами. Это является достоинством вейвлет-анализа, так как среди основных требований, предъявляемых к системам обнаружения и идентификации, особое место занимает ограничение на время анализа.

Основное различие, существующее между привычной синусоидой и вейвлетом, на основании которых построены соответственно гармонический анализ и вейвлет- анализ, состоит в их различной способности к временной локализации. В то время как синусоида позволяет локализовать составляющие данных лишь в частотной области, вейвлеты обладают

способностью к время-частотной локализации. Поясним это различие с помощью рис. 5.1. В верхней его части показаны две синусоиды : $\sin(8\pi x)$ и $\sin(16\pi x)$, заданные на интервале (0,1). Нижняя половина рисунка содержит изображение вейвлета db10 (вейвлета Добеши 10-го порядка), частота осцилляций которого может быть задана наперед.Таким образом, свойство вейвлета локализовать информацию во временной областине означает наличие у него компактного носителя: здесь является вполне достаточной концентрация его осцилляций на некотором конечном интервале, тогда как синусоида компактный носитель иметь не может.

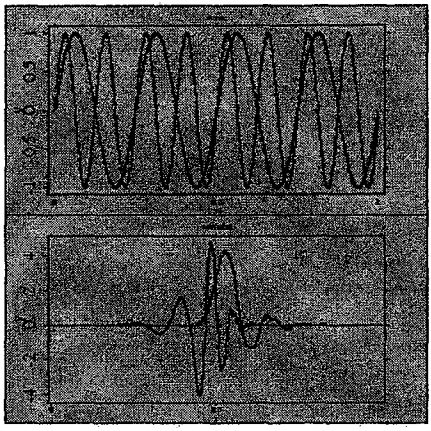
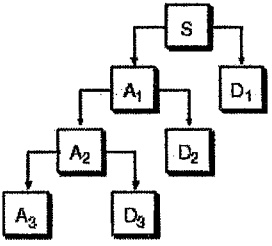


Рис 5.1.Вейвлет-анализ

Приведенный пример свидетельствует о том, что применение вейвлет-базисов может помочь в определении характерных точек сигнала – разрывов, скачков фазы, изменении производной.

Структура вейвлет-анализа имеет древовидную форму, представленную на рис. 5.2. На первом этапе сигнал раскладывается с использованием заданного базиса функций на аппроксимирующую и детализирующую составляющую. Затем, аппроксимирующая составляющая раскладывается, в свою очередь, на аппроксимирующую и детальную

составляющие второго уровня. Для декомпозиции n-го уровня существуют $n+1$ возможных способов.



$$S = A_1 + D_1$$
$$= A_2 + D_2 + D_1$$
$$= A_3 + D_3 + D_2 + D_1$$

Рис. 5.2.Структура вейвлета

При использовании пакетного вейвлет-анализа раскладываться по базисным функциям могут не только аппроксимирующие, но и детальные составляющие сигнала. Это соответствует более чем 2^{2m-1} способам декомпозиции сигнала. Дерево декомпозиции представлено на рис. 5.3.

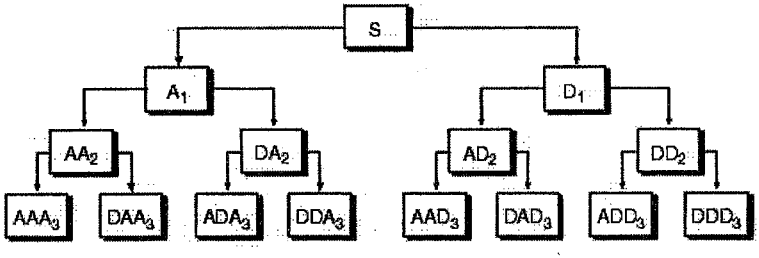


Рис. 5.3.Дерево декомпозиции

Это дерево представляет собой часть полного бинарного дерева декомпозиции сигнала. Пакетный вейвлет-анализ позволяет представить сигнал S как сумму A1 + AAD3 + DAD3 + DD2. Для достижения поставленной цели с использованием пакетного вейвлет-преобразования необходимо выбрать наилучшую структуру дерева разложения. Эта структура выбирается исходя из минимума значений критериев:

- энтропии Шеннона,
- логарифма энергии сигнала,
- порогового значения энергии сигнала.

В качестве примера практического использования вейвлет-анализа рассмотрим анализ сигнала шагов человека, записанного на выходе сейсмоакустического датчика. Цель анализа - выявить характерные точки во временной области. Для анализа используем пакет *MatLab Wavelet Toolbox*, позволяющий синтезировать всевозможные алгоритмы обработки информации (данных, сигналов и изображений) с использованием вейвлет-функций в пакетном режиме. На рис. 5.4. показана декомпозиция (разложение) сигнала шага с использованием обратного биортогонального ('rbio') пакетного вейвлет-преобразования (порядок вейвлета внутри семейства 'rbio'- 6.8, уровень декомпозиции – 2, дерево декомпозиции построено с использованием критерия минимума энтропии Шеннона). Правый верхний рисунок показывает временную реализацию сигнала. Левый рисунок показывает дерево разложения на уровни. Правый нижний рисунок показывает четыре (нижних) уровня разложения. Коэффициенты разложения сигнала представлены различными оттенками. Из рис. 5.4 видно, что наиболее информативным является четвертый уровень. Коэффициенты разложения позволяют выделить характерную точку сигнала, соответствующую отрыву носка ноги. Для более наглядного представления изменения коэффициентов разложения при изменении сигнала нужно воспользоваться левой кнопкой мыши для выделения нужного уровня коэффициентов и масштабирующей кнопкой Y+ интерфейса (см. рис. 5.9).

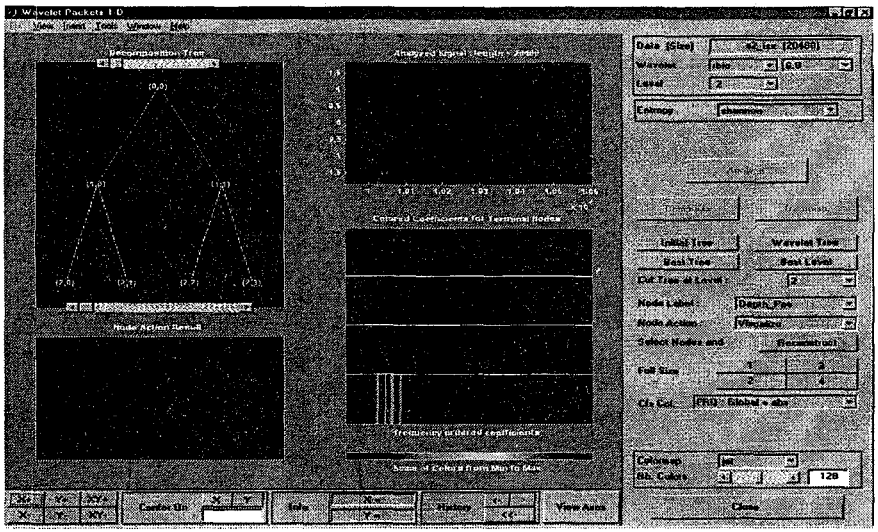


Рис. 5.4.Декомпозиция (разложение) анализа с использованием пакетного вейвлет-преобразования

Режимы работы и порядок выполнения задания

1. **Интерфейсный режим.** Пакет MATLAB предоставляет удобный графический интерфейс для работы с вейвлетами. Он вызывается с помощью команды wavemenu, набранной в командной строке MATLAB.

Появляется диалоговое окно (рис. 5.5).

Одномерные вейвлет-преобразования используются для обработки аудио-сигналов, двумерные – для обработки сигналов изображения.

В данной работе для предварительно сохраненных в виде mat-файлов одномерных сигналов будем использовать Wavelet 1-D и Wavelet Packet 1-D.

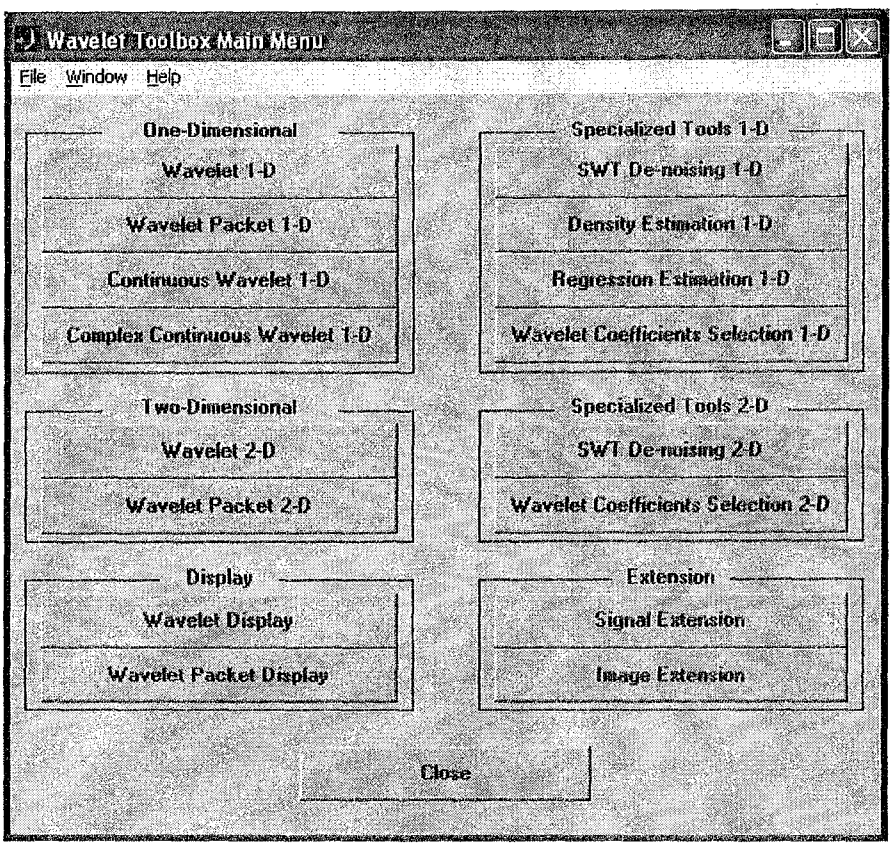
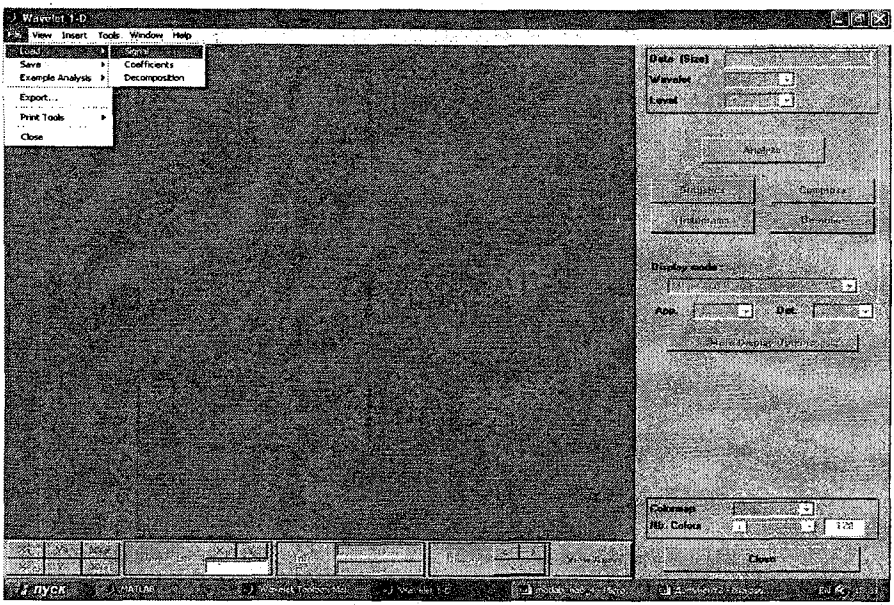


Рис. 5.5.Диалоговое окно графического интерфейса для работы с вейвлетами.

При выборе Wavelet 1-D появляется окно, представленное на рис.5.6.



Рису. 5.6 Диалоговое окно Wavelet 1D

Загружаем в него предварительно сохраненный в виде mat-файла исследуемый сигнал, выбираем вид вейвлета и степень детализации (рис. 5.7), нажимаем кнопку Analyze.

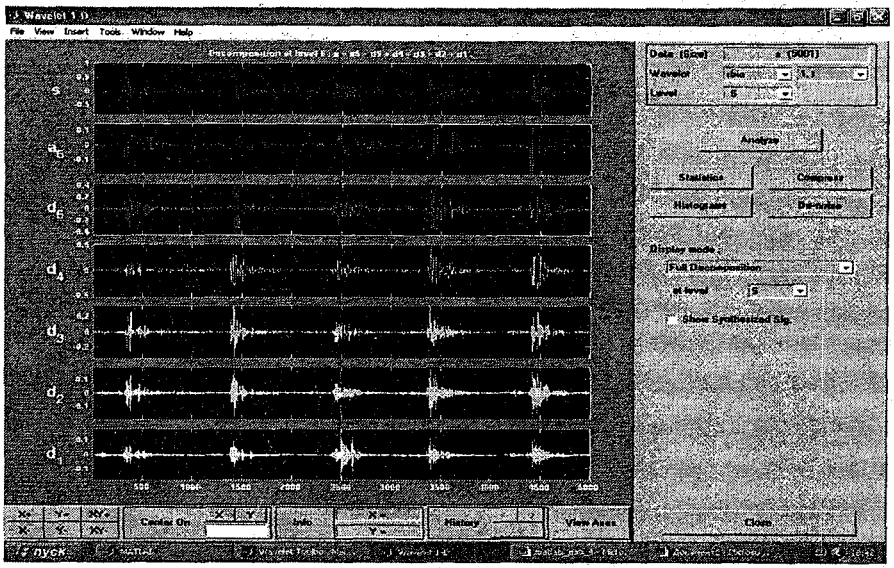
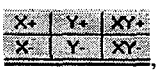


Рис 5. 7. Вид вейвлета и степень детализации

Получаем *Full Decomposition* или Полное разложение сигнала. В полосе прокрутки выбираем *Show and Scroll (Stem CFs)* (Показ и Прокрутка). Получаем абсолютные значения коэффициентов разложения. С помощью левой кнопки мыши выбираем заданный коэффициент разложения (уровень детализации). С помощью кнопки *Y+* (см. рисунок 5.8) выводим его на весь экран.



, для отмены действия используем кнопку 

.

а) *в)*

Рис.5.8.Рабочие кнопки *а* и *в*.

С помощью левой кнопки мыши и кнопки X+ выделяем нужный фрагмент сигнала (полезный сигнал). Получаем рисунок, похожий на рис. 5.9.

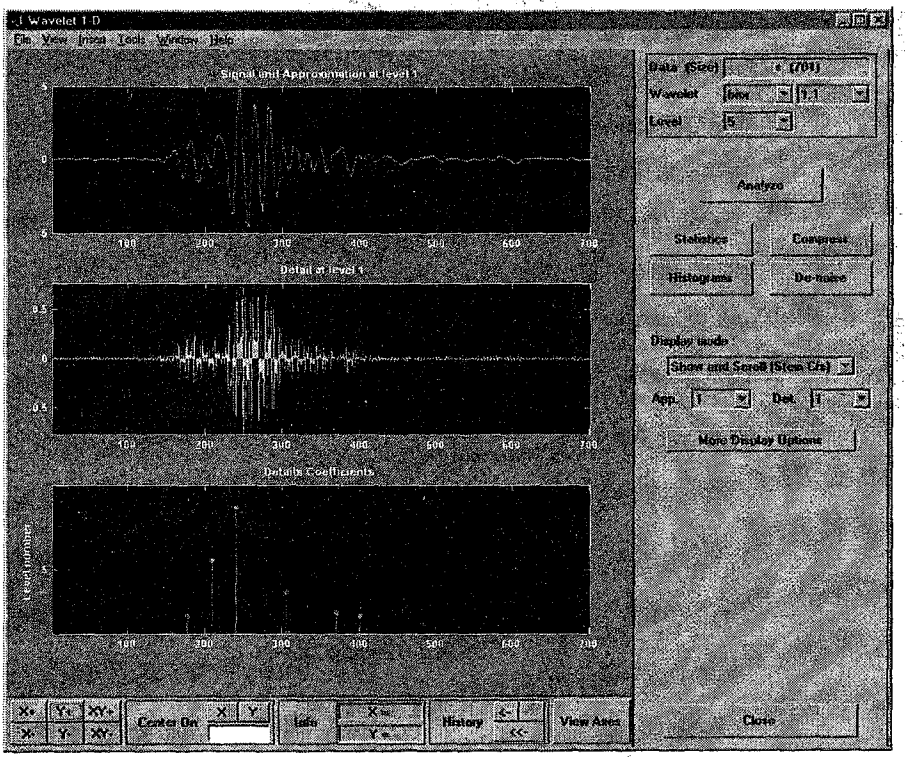


Рис. 5.9.Выделение нужного фрагмента

Для выбора нужного коэффициента и временного интервала используем кнопки.

2. **Командный режим.** Нужные коэффициенты детализации также можно получить, написав несколько команд.

Так, например,

Load s - загрузка сигнала;

[c,l]=wavedec(s,3,'rbio1.1')- вспомогательные коэффициенты разложения;

cA3=appcoef(c,l,'rbio1.1')- аппроксимирующий коэффициент;

cB3=detcoef(c,l,3)- детализирующий коэффициент третьего уровня;

Поскольку массив полученных коэффициентов имеет иную размерность, чем исходный сигнал, то для синхронизации значений сигнала значениям вейвлет-коэффициентов при выводе на экран графиков нужно использовать масштабирование.

Масштабирование графика исходного сигнала для соответствия графику вейвлета

Команда: *a=length(s2_isx)* определяет длину исходного сигнала;

Результат:

a =

20480

>> *b=length(cd2)* - определяет длину вейвлет-сигнала;

b =

5120

k=5120/20480 - определяет коэффициент масштаба;

l=1:20480 - вектор исходных отсчетов сигнала;

>> *l1=l*k*- вектор промасштабированных отсчетов сигнала;

>> *subplot(2,1,1);plot(l1,s2_isx);subplot(2,1,2);plot(cd2)*- вывод на экран графиков рис. 5.10

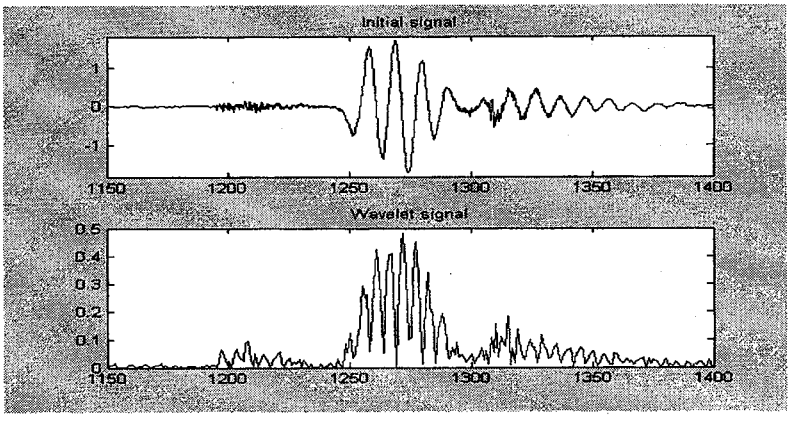


Рис. 5.10.Результат масштабирования графиков

При выполнении пакетного вейвлет-анализа (кнопка меню Wavelet Packet 1-D.) необходимо получить окно, аналогичное рис. 5.11.

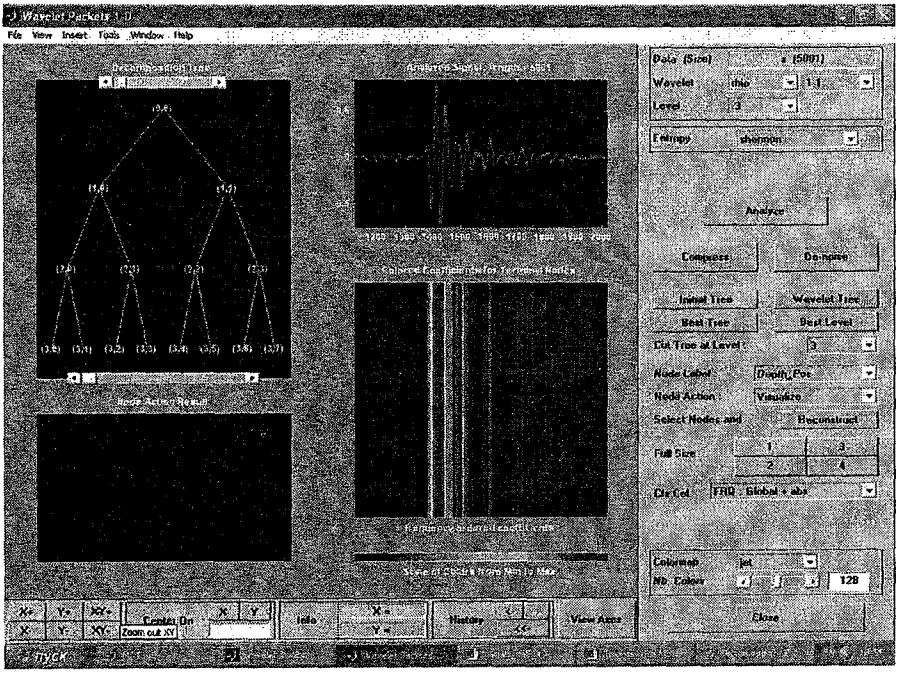


Рис. 5.11.Окно,полученное при выполнении вейвлет-анализа

Коэффициенты нужного уровня можно получить без использования графического интерфейса с помощью команд пакетного вейвлет-преобразования:

*x = sin(8*pi*[0:0.005:1]);* - пример исходного сигнала;

t = wpdec(x,4,'db1'); - пакетное вейвлет преобразование вейвлета Добеши 1, уровня 4;

wpviewcf(t,1);- графическое представление в цвете коэффициентов пакетного вейвлет-преобразования.

Задание для выполнения работы

Исследовать заданный сигнал с помощью графического интерфейса 1-D и Packet 1-D вейвлет-преобразования и написать программу для извлечения заданных коэффициентов детализации.

Варианты вейвлетов:

1. haar, level 3
2. db3, level 2
3. haar, level 5
4. db4, level 2
5. db2, level 4
6. db5, level 7
7. bior1.1, level 3
8. bior1.3, level 4
9. bior2.2, level 3
10. bior3.5, level 2
11. bior5.5, level 2
12. rbio1.1, level 2
13. rbio1.3, level 2
14. rbio2.2, level 2
15. rbio3.5, level 3
16. rbio2.8, level 5
17. dmey , level 2
18. dmey, level 3
19. dmey, level 7
20. coif 2, level 3

Работа № 6

Работа с бинарным файлом

Цель работы: приобретение навыков работы с файлами данных, записанных в различных форматах.

1. Открытие бинарного файла

Для работы с реальными сигналами, поступающими от датчиков на систему ввода информации в компьютер необходимо уметь работать с данными, записанными в бинарных кодах, так как именно в таком виде представляется сигнал на выходе АЦП.

Сначала необходимо открыть бинарный файл с помощью команды, создающей идентификатор файла *fid=fopen(filename,permission)*, которая открывает файл *filename* с определенным разрешением *permission*. Разрешение может быть:

'rb' – для чтения бинарного файла;

'w' – для записи (с созданием, если необходимо);

'r+' – для записи и чтения без создания;

'w+' - для записи и чтения с созданием или усечением;

и другие.

Если открытие файла произошло успешно, то идентификатор файла *fid* выдает скалярное целое: *fid=1* (стандартный выход) или *fid=2* (стандартная ошибка). Если открытие файла не произошло успешно, то *fid= -1*.

Затем необходимо считать бинарные данные из файла с помощью команды *fread*. Синтаксис команды:

[a, count]=fread(fid,size,precision), или *a=fread(fid,size,precision)*,

где *a* – выходная матрица,

count – возвращает количество успешно считанных элементов,

fid- идентификатор файла, полученный с помощью команды *fopen,*

size - размер считываемого фрагмента. Если он не определен, то считывается весь файл,

precision - представление считываемых битов как символьных, числовых значений или чисел с плавающей точкой.

Например,

'*char*' – символьная переменная, 8 бит, со знаком или без;

'*int16*' – целое, 16 бит;

'*float32*' – число с плавающей точкой, 32 бита;

и другие

В экспериментально полученном файле существует следующая структура:

Первый байт занят служебной информацией, состоящей из:

8-ми чисел символьных переменных, представляющих номера восьми каналов;

8-ми целых 16-битовых чисел, представляющих коэффициенты усиления в каналах;

1–го целого 16-битового числа, определяющего частоту дискретизации АЦП.

Со второго байта идет строка сигналов, не разделенных по каналам, поэтому после успешного открытия файла и считывания первого байта информации возникает необходимость установить индикатор позиции на второй байт в файле. Это производится с помощью команды *fseek.* Команда *fseek* устанавливает индикатор позиции в файле с данным идентификатором *fid* на данную позицию *offset* относительно момента отсчета *origin.* Синтаксис команды: *fseek(fid,offset,origin).* Например, команда *fseek(fid,4,0)* устанавливает индикатор позиции на четвертый байт относительно текущей позиции.

После установления индикатора позиции на второй байт нужно считать данные как 16-битовые целые числа и определить длину записи. Разделив длину записи на сумму каналов, получим длину записи в одном

канале. Воспользовавшись командой *reshape,* разделим всю запись на столбцы по числу каналов.

Необходимо отметить, что для записи экспериментальных данных использовалось АЦП, поэтому при чтении бинарного файла значения амплитуды сигнала получаются не в вольтах, а в единицах, в которых учтена разрядность АЦП. Для перевода этих единиц в вольты необходимо полученные значения амплитуды сигнала умножить на коэффициент

$$K = \frac{10}{65535},$$

где 10 – это диапазон изменения напряжения от +5В до –5В;

65535 – количество разрядов АЦП.

2. Обработка сигнала по частям, вывод графиков характеристик в каждой части сигнала и сохранение последнего результата в excel-файле wk1 с выдачей запроса на сохранение

Поскольку временная реализация сигнала довольно длительная, имеет смысл обработать сигнал по частям и просмотреть графики полученных характеристик для выбора наиболее интересных отрезков временной реализации. Сделать это можно с использованием цикла *for....end* с переменной, заданной пошагово (см. работу №2). Шаг определяет размер исследуемого фрагмента сигнала. На каждом шаге можно построить график полученной характеристики (см. работу №3). Для того чтобы он задержался на экране для просмотра, прежде чем программа продолжит выполняться, нужно воспользоваться командой *pause.*

Сохранение последнего полученного результата с выдачей запроса на сохранение происходит с использованием команд *questdlg*, а также с помощью команд *switch* и *case.*

Пример использования команды *questdlg:*

ButtonName=questdlg('What is your wish?', ...

```
        'Genie Question', ...
        'Food','Clothing','Money','Money');
switch ButtonName,
 case 'Food',
  disp('Food is delivered');
 case 'Clothing',
  disp('The Emperor''s  new clothes have arrived.')
 case 'Money',
  disp('A ton of money falls out the sky.');
end % switch
```

3. Открытие файла, записанного с помощью DAQ Card

Компания National Instruments разработала устройство DAQ Card (Data Acquisition Card – Карта сбора данных), сопрягаемое с компьютером, которая позволяет вести запись сигналов от различных датчиков. Открытие файла, записанного DAQ Card включает в себя:

1.Создание идентификатора файла с такими же свойствами, как для бинарного файла.

2.Считывание из файла строки состояния каналов channels *(size=8; precision= 'single')*.

3.Считывание из файла количества отсчетов на канал N_ch *(size=1; precision= 'single')*.

4.Считывание из файла частоты дискретизации Rate *(size=1; precision= 'single')*.

5.Считывание из файла двумерного массива данных, в котором столбец соответствует каналу Signals *(size=[N_ch sum(Channels)]; precision= 'single')*.

Задание для выполнения работы

1. Написать программу-функцию по открытию экспериментально полученного бинарного файла. Перевести значения амплитуды сигнала в вольты и вывести график сигнала.

2. Обработать полученный сигнал (бинарный файл) по частям в соответствии с вариантом, вывести график заданной характеристики в каждой части сигнала и сохранить последний результат в excel-файле wk1 с выдачей запроса на сохранение.

3. Написать программу-функцию по открытию экспериментально полученного файла DAQ Card.

Варианты для задания

Вариант 1. Длина исследуемого фрагмента 1000 отсчетов,исследуемая характеристика – автокорреляционная функция сигнала первого канала.

Вариант 2. Длина исследуемого фрагмента 1000 отсчетов, исследуемая характеристика – огибающая сигнала первого канала.

Вариант 3. Длина исследуемого фрагмента 1000 отсчетов, исследуемая характеристика – спектр сигнала второго канала.

Вариант 4. Длина исследуемого фрагмента 1000 отсчетов, исследуемая характеристика – отфильтрованный сигнал первого канала (ФНЧ 4 порядок, fсреза=100 Гц).

Вариант 5. Длина исследуемых фрагментов сигналов 1000 отсчетов каждый, исследуемая характеристика – взаимнокорреляционная функция сигналов первого и второго каналов.

Вариант 6. Длина исследуемых фрагментов сигналов 1000 отсчетов каждый,
исследуемая характеристика – взаимный спектр сигналов первого и второго каналов.

Вариант 7. Длина исследуемого фрагмента 2000 отсчетов, исследуемая характеристика – автокорреляционная функция сигнала третьего канала.

Вариант 8. Длина исследуемого фрагмента 2000 отсчетов, исследуемая характеристика – огибающая сигнала пятого канала.

Вариант 9. Длина исследуемого фрагмента 2000 отсчетов, исследуемая характеристика – спектр сигнала четвертого канала.

Вариант 10. Длина исследуемого фрагмента 2000 отсчетов, исследуемая характеристика – отфильтрованный сигнал второго канала (ФНЧ 4 порядок, fсреза=200 Гц).

Вариант 11. Длина исследуемых фрагментов сигналов 2000 отсчетов каждый, исследуемая характеристика – взаимнокорреляционная функция сигналов третьего и второго каналов.

Вариант 12. Длина исследуемых фрагментов сигналов 2000 отсчетов каждый, исследуемая характеристика – взаимный спектр сигналов четвертого и второго каналов.

Вариант 13. Длина исследуемого фрагмента 4000 отсчетов, исследуемая характеристика – автокорреляционная функция сигнала пятого канала.

Вариант 14. Длина исследуемого фрагмента 4000 отсчетов, исследуемая характеристика – огибающая сигнала второго канала.

Вариант 15. Длина исследуемого фрагмента 4000 отсчетов, исследуемая характеристика – спектр сигнала третьего канала.

Вариант 16. Длина исследуемого фрагмента 4000 отсчетов, исследуемая характеристика – отфильтрованный сигнал первого канала (ФНЧ 4 порядок, fсреза=150 Гц).

Вариант 17. Длина исследуемых фрагментов сигналов 4000 отсчетов каждый, исследуемая характеристика – взаимнокорреляционная функция сигналов третьего и пятого каналов.

Вариант 18. Длина исследуемых фрагментов сигналов 4000 отсчетов каждый, исследуемая характеристика – взаимный спектр сигналов четвертого и второго каналов.

Вариант 19. Длина исследуемого фрагмента 3000 отсчетов, исследуемая характеристика – автокорреляционная функция сигнала второго канала.

Вариант 20. Длина исследуемого фрагмента 3000 отсчетов, исследуемая характеристика – огибающая сигнала третьего канала.

Вариант 21. Длина исследуемого фрагмента 3000 отсчетов, исследуемая характеристика – спектр сигнала четвертого канала.

Вариант 22. Длина исследуемого фрагмента 3000 отсчетов, исследуемая характеристика – отфильтрованный сигнал третьего канала (ФНЧ 3 порядок, fсреза=150 Гц).

Вариант 23. Длина исследуемых фрагментов сигналов 3000 отсчетов каждый, исследуемая характеристика – взаимнокорреляционная функция сигналов третьего и четвертого каналов.

Вариант 24. Длина исследуемых фрагментов сигналов 3000 отсчетов каждый, исследуемая характеристика – взаимный спектр сигналов четвертого и первого каналов.

Вариант 25. Длина исследуемого фрагмента 1500 отсчетов, исследуемая характеристика – автокорреляционная функция сигнала первого канала.

Вариант 26. Длина исследуемого фрагмента 1500 отсчетов, исследуемая характеристика – огибающая сигнала второго канала.

Вариант 27. Длина исследуемого фрагмента 1500 отсчетов, исследуемая характеристика – спектр сигнала третьего канала.

Вариант 28. Длина исследуемого фрагмента 1500 отсчетов, исследуемая характеристика – отфильтрованный сигнал четвертого канала (ФНЧ 4 порядок, fсреза=300 Гц).

Вариант 29. Длина исследуемых фрагментов сигналов 1500 отсчетов каждый,

исследуемая характеристика – взаимнокорреляционная функция сигналов третьего и первого каналов.

Вариант 30. Длина исследуемых фрагментов сигналов 1500 отсчетов каждый, исследуемая характеристика – взаимный спектр сигналов второго и первого каналов.

Работа № 7

Создание файлов-сценариев

Цель работы: Знакомство со *script*-файлами и организация диалогового режима с пользователем

Кроме процедур-функций в пакете MATLAB можно создавать файлы-сценарии с использованием окна *MATLAB Editor\Debugger* . В этом окне пишется последовательность команд, причем составной частью сценария могут быть и функции. Затем, созданному m-файлу присваивается имя. Файл сохраняется и затем может быть запущен из рабочего окна (если он сохранен в рабочем каталоге MATLAB) набором в командной строке имени файла или последовательностью команд, запускающей браузер *(file-run script),* если он помещен в какой-либо другой каталог. Рабочий каталог MATLAB устанавливается с помощью браузера пути (иконка на панели инструментов, показанная на рис. 7.1) последовательностью команд *path – add to path* - и сохраняется в списке путей с помощью команд *file-save Path.*

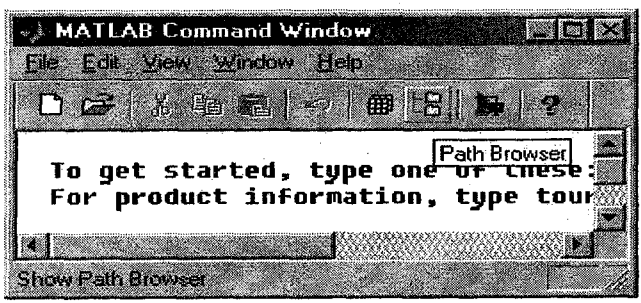


Рис. 7.1.Иконка рабочего каталога MATLAB

Файлы-сценарии удобно оформлять в диалоговом режиме с пользователем. Наиболее употребительной в диалоговом режиме является команда *inputdlg*.

Синтаксис команды: *Answer=inputdlg(Prompt,Title,LineNo, DefAns).*

В ней *Prompt* представляет собой строку подписей над линейками (количество их определяется командой *LineNo*), в которые вводятся данные. *Title* представляет собой заголовок диалогового окна, а *DefAns* представляет собой строку вводимых по умолчанию данных в линейки.

После введения данных в строки диалогового окна необходимо присвоить значения строк каким-либо переменным. Это делается, например, с помощью команд *var1=answer{1}; var2=answer{2},* присваивающих переменным *var1* и *var2* значения 1 и 2 строк диалогового окна *answer*. Поскольку вводимые в линейки диалогового окна данные могут быть не только строковыми, но и представлять собой числа, то считывание введенных в диалоговом режиме числовых данных в память matlab для дальнейшей работы производится с помощью команды *x=str2num(s),* преобразующую матрицу строк в массив чисел. Для организации вывода значений переменных на экран дисплея используется команда *disp*.

Очень удобно вводить с помощью диалогового окна путь к открываемому файлу, который можно прописать в линейке по умолчанию. Однако в некоторых ситуациях этот путь является ошибочным и требуется проверить факт, что переменная, которой присвоено значение пути к файлу, определена. Эта проверка осуществляется с помощью команды *var= exist('A')*. Если переменная *var* равна нулю, то переменная *A* не определена.

Еще одним распространенным диалоговым режимом является выдача сообщения об ошибке, организуемое с помощью команды *errordlg('Errorstring','Dlgname'),* в которой *Errorstring* представляет собой сообщение об ошибке, а *Dlgname* – заголовок диалогового окна.

Для того, чтобы лучше понять работу файлов-сценариев и диалогового окна, создайте файл-сценарий по образцу примера 1, в котором

«закомментирована» переменная var1. Программа должна проверить ее существование и выдать сообщение об ошибке. Если затем знак комментария снять, то на дисплее должны распечататься значения переменных var1 и var2, значения которых вводятся в виде строк в диалоговом окне.

Пример 1.Example.m:

```
  prompt={'Enter the matrix size for x^2:','Enter the colormap name:'};
  def={'20','hsv'};
  Title='Input for Peaks function';
  lineNo=1;
  answer=inputdlg(prompt,Title,lineNo,def);
  var11=answer{1};
  % var1=str2num(var11);
f=exist('var1');
if (f==0)
errordlg('Значение не введено','Ошибка');
else
  var2=answer{2};
. disp('var1=');
  disp(var1);
  disp('var2=');
  disp(var2);
end
```

В пакете MATLAB можно также задавать путь к файлу через поиск в проводнике.

Для этого используется команда открытия файла через диалоговое окно *UIGETFILE.* Синтаксис: *[FILENAME, PATHNAME] = UIGETFILE('filterSpec', 'dialogTitle').*

Команда показывает диалоговое окно для заполнения пользователем и возвращает строки имени файла и пути к нему. Успешное выполнение

команды происходит только в том случае, если файл существует. Если выбранный файл не существует, показывается сообщение об ошибке и контроль возвращает диалоговое окно. Пользователь может ввести другое имя файла или нажать «выход». Параметр *filterSpec* определяет начальный показ файлов в диалоговом окне. Например, '*.m' показывает все m-файлы MATLAB.

Параметр *'dialogTitle'* является строкой, содержащей название диалогового окна.

Затем полученный путь к файлу и его имя необходимо соединить в цепочку, чтобы получить доступ к файлу. Для соединения строк в цепочку используется команда *STRCAT*. Синтаксис: *T = STRCAT(S1,S2,S3,...)*. Команда составляет горизонтальную цепочку из символов *S1, S2, S3* и т.д., многоточие игнорируется. Например, *strcat({'Red','Yellow'},{'Green','Blue'})* возвращает *'RedGreen' 'YellowBlue'*.

Задание для выполнения работы

Задание № 1.

Создать файл-сценарий открытия файла, записанного в ASCII-кодах и файл-сценарий открытия экспериментального файла, записанного в бинарных кодах с использованием диалогового режима ввода пути к файлу и выдачей сообщения об ошибке в случае ввода неправильного пути.

Задание № 2.

Создать файл-сценарий открытия файла, записанного в ASCII-кодах и файл-сценарий открытия экспериментального файла, записанного в бинарных кодах с использованием поиска файла через проводник.

Работа № 8

Создание графического интерфейса

***Цель работы*:** создание удобного интерфейса пользователя для выполнения файла-сценария.

Как правило, программы в пакете matlab составляются для того, чтобы графически представить результаты расчетов. Для выполнения этого необходимо, чтобы окна *figure* при выполнении программы не закрывались, а постоянно находились в рабочей области. Кроме того, окна *figure* могут содержать различные «кнопки», которые делают работу с программой более удобной. Поэтому часто решение задачи оформляют в виде двух скрипт-файлов: первый задает удобный пользовательский интерфейс, а второй представляет собой так называемую CallBack-программу, представляющую собой фактически набор подпрограмм для выполнения основной программы, которая обращается к CallBack-программе с помощью цифровых меток.

Рассмотрим написание первой программы для создания интерфейса пользователя.

Все переменные в ней должны быть объявлены глобальными.

Окно *figure* может иметь название, задаваемое с помощью параметра *'Name'*. Окно устанавливается командой *set*, имеющей несколько параметров. Размеры окна задаются командой *'Position'* , имеющей четыре параметра: [лево, низ, ширина, высота]. Параметр *'Resize'* позволяет свертывать окно, параметр *'Numbertitle'* включает и выключает нумерацию окон *figure*.

Как правило, нужно вывести на экран сигналы нескольких (например, двух, каналов). Для них в главной фигуре надо образовать ячейки с осями при помощи команды *axes*, с параметром *'Position'*, в котором (0,0) соответствует нижнему левому углу, а (1.0,1.0) соответствует верхнему правому углу. В команде *set* параметр *'Fontsize'* соответствует размеру шрифта, а параметр *'Box'* нанесению меток по всему окну.

Например, программа

```
        global H1 H1_1 H1_2 H1_3;
  H1=figure('Name','Пример');
  set(H1,'Position',[1 29 800
534],'Resize','On','NumberTitle','Off');
H1_1=axes('Position',[0.3 0.7 0.65 0.2]);
set(H1_1,'FontSize',10,'Box','On');
H1_2=axes('Position',[0.3 0.4 0.65 0.2]);
set(H1_2,'FontSize',10,'Box','On');
H1_3=axes('Position',[0.3 0.1 0.65 0.2]);
set(H1_3,'FontSize',10,'Box','On');
```

выводит на экран дисплея окно с тремя ячейками для последующего вывода в них сигнальных реализаций.

В оставшемся пространстве главной фигуры можно разместить клавиши управления с помощью команды *uicontrol* с параметрами *'Style'*- задает вид кнопки, которая может быть 'Pushbutton', 'Radiobutton', *'Checkbox' ,'Edit', 'Text'; 'Position'; 'String'* – выводит надпись на кнопке; *'Enable'* – делает кнопку рабочей или выключенной; *'CallBack'* – задает обращение к подпрограмме при нажатии кнопки; *'NumOfCBFun=5; Podprcb'* – задает метку обращения (*Number of Callback Function*) к подпрограмме *Podprcb* .

Пример:

```
H5=uicontrol('Style','checkbox','Position',[20 130 135
5],'String','Полосовой
фильтр','Enable','Off','CallBack','NumOFCBFun=5;
Podprcb');
```

Кроме управляющих кнопок, для создания удобного интерфейса можно воспользоваться командой *makemenu(fig,Labels,Calls),* создающей структуру меню в фигуре *fig* в соответствии с порядком матрицы строк *Labels*. Каскадность меню указывается начальным знаком '>' в матрице *Labels*. *Calls* является матрицей строк, содержащей коллбэки. Она должна иметь то же самое количество рядов как и *Labels*.

Например, программа

```
gcf=figure('Name','Пример');
  set(gcf,'Units','pixel','Position',[204 8 593
520],'Resize','On','NumberTitle','Off','WindowStyle','n
ormal','Color','White');

labels = str2mat( ...
     '&File', ...
     '>&New^n', ...
     '>&Open', ...
     '>>Open &document^d', ...
     '>>Open &graph^g', ...
     '>-------', ...
     '>&Save^s', ...
     '&Edit', ...
     '&View', ...
     '>&Axis^a', ...
     '>&Selection region^r' ...
     );
        calls = str2mat( ...
     '', ...
     'NumOFCBFun=8; examplelcb;',...
     '', ...
     'NumOFCBFun=9; examplelcb ;',...
```

```
    'NumOFCBFun=10; examplelcb;',...
    '', ...
    'NumOFCBFun=11; examplelcb ;',...
    '', ...
    '', ...
    'NumOFCBFun=12; examplelcb;',...
    'NumOFCBFun=15; examplelcb;'...
    );
  handles = makemenu(gcf, labels, calls);
```

открывает окно «Пример», изображенное на рис. 8.1, в котором открыто меню '*File Edit View*' с разными подменю, обращаясь к которым можно запустить на выполнение различные задания, помеченные метками в подпрограмме *examplecb*.

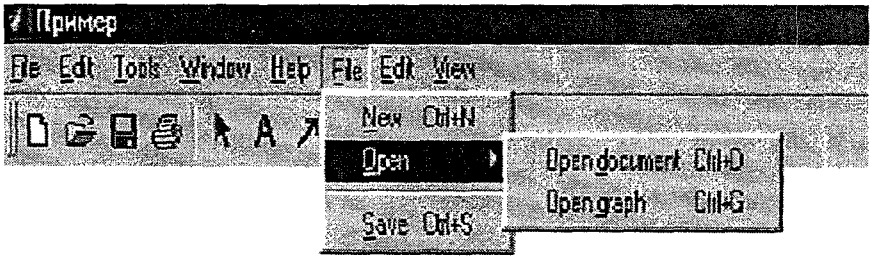


Рис. 8.1. Пример меню, при помощи которого можно запустить на выполнение различные задания

Задание для выполнения работы

Написать программу создания многооконного интерфейса, выводящего на экран фигуру, изображенную на рис. 8.2 с управляющими кнопками.

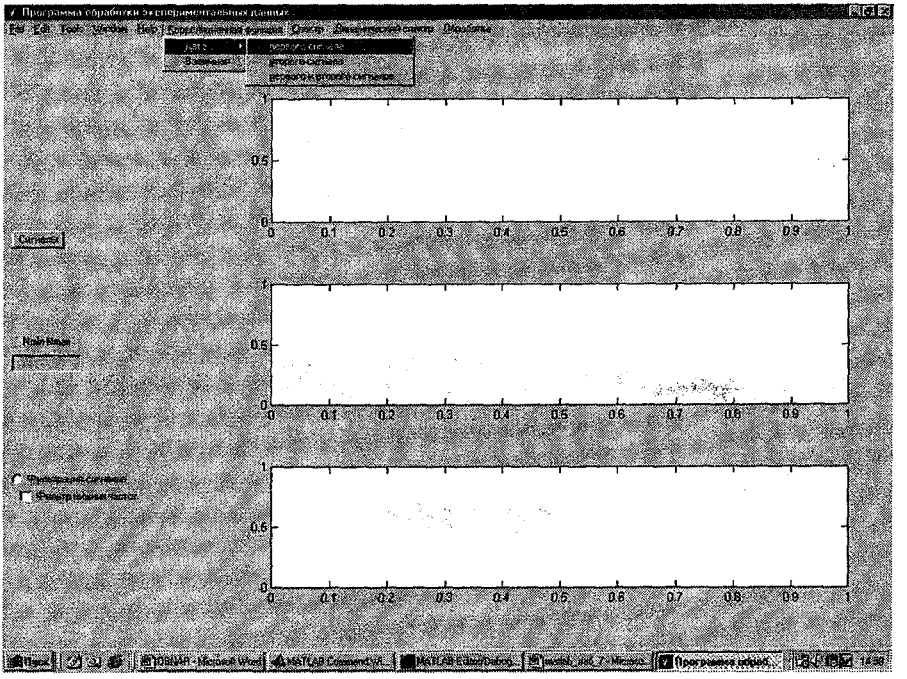


Рис. 8.2.Многооконный интерфейс

Создание callback-подпрограмы

Цель работы: написание callback-подпрограммы для программы, создающей интерфейс пользователя.

Как указывалось в работе № 7, для программы, задающей интерфейс, нужно написать callback-подпрограммы, которые будут осуществляться, например, при нажатии каких-либо кнопок.

Для примера рассмотрим callback-подпрограмму, позволяющую выводить общее количество отсчетов временного ряда и изменять его, когда необходимо выделить для исследования какой-либо участок сигнала. Рассмотрим программу *primer* следующего содержания:

```
   global H1 H1_1  H10 H14 H15;
    H1=figure('Name','Программа обработки
экспериментальных данных ');
    set(H1,'Position',[300 300 500
134],'Resize','On','NumberTitle','Off');
   H1_1=axes('Position',[0.25 0.2 0.7 0.7]);
set(H1_1,'FontSize',10,'Box','On');

H10=uicontrol('Style','PushButton','Position',[10 91 60
20],'String','Сигналы','FontSize',6,'CallBack',
'NumOFCBFun=10; primercb');
    H14=uicontrol('Style','Edit','Position',[10 41 80
20],'Enable','Off','CallBack','NumOFCBFun=14;
studycb');
  H15=uicontrol('Style','Text','Position',[10 63 80
20],'String','Nmin Nmax','Enable','On');
```

Программа позволяет организовать панель figure, изображенную на рис. 9.1 для вывода на экран временного ряда.

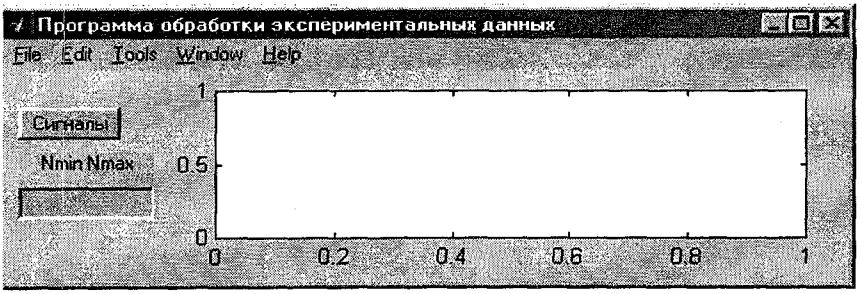


Рис. 9.1. Панель figure для вывода на экран временного ряда

Callback-программу к программе *primer* удобно назвать *primercb*. Обращение к рабочим кнопкам интерфейса организуется с помощью операторов *«if», «elseif», «end».*

Щелчок «мышью» на кнопку «Сигналы» соответствует оператору в *call-back* программе *if(NumOFCBFun==10);* следом идут операторы (или диалоговый режим), например, открытия сигнального файла.

Иногда бывает необходимо при нажатии на кнопку «Сигналы» получить меню, позволяющее выбрать открытие либо бинарного файла, либо файла в ASCII – кодах. В этом случае можно воспользоваться оператором *menu*, который имеет синтаксис

Choice=menu(header,item1,item2,....), и показывает на экране дисплея заголовок меню *header* и темы меню *items*.

Например, команда *K = MENU('Choose a color','Red','Blue','Green')* высвечивает на экране окно, показанное на рис/ 9.2. Нажатие на кнопки *menu* описывается также с помощью операторов *«if», «elseif», «end».* Так, например, для рисунка 9.2 указание в call-back программе *if k=2* обозначает

выбор кнопки «*blue*»; далее должны следовать команды, которые нужно выполнить при нажатии этой кнопки.

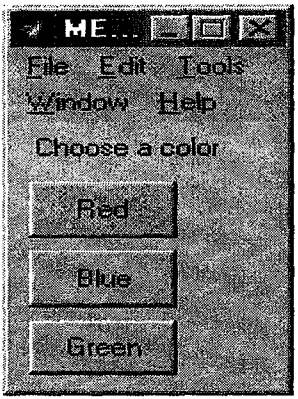


Рисунок 9.2. Окно команды "MENU"

Допустим, Вы успешно открыли файл *h1* и знаете длину его числового вектора *lh1*. С помощью команды *sprintf* получим строковую переменную числа отсчетов с 1 до *lh1*.

lhs=sprintf('1 %d',lh1);

С помощью команды

set(H,'PropertyName',PropertyValue),

устанавливающей значение *PropertyValue* с именем *'PropertyName'* в графический объект *H*, установим масштаб выводимого в окне сигнала командой

set(H14,'String',lhs).

Считаем значения сигнала в память с помощью команды *v1=sscanf(lhs,'%g');* Определим анализируемый фрагмент сигнала как *s11=h1(v1(1):v1(2));* теперь можно вывести его в графическом окне и определить более короткий отрезок на нем (введя новые значения начала и конца анализируемого фрагмента), который подлежит исследованию. С помощью команды

v=get(H,'PropertyName'),

которая возвращает значение специфицированного *'PropertyName'* для графического объекта *H*, получим их для считывания в память. Установим новые пределы осей в окне и новый анализируемый фрагмент сигнала:

```
            elseif(NumOFCBFun==14);
        str=get(H14,'String'); v1=sscanf(str,'%g');
        axes(H1_1);
        xlim([v1(1) v1(2)]);
    %Анализируемый фрагмент сигнала
        s11=h1(v1(1):v1(2));
```

Задание для выполнения работы

Создать окно интерфейса, показанное на рис. 9.1, с возможностью выбора:

1. Открытия бинарного файла или файла в ASCII-кодах.

2. Написать call-back подпрограмму для этого интерфейса с возможностью выбора анализируемого фрагмента сигнала путем задания величины масштаба в окне.

Работа №10

Обработка файлов

Цель работы: создание в среде MATLAB исполняемых файлов

Компилятор MATLAB (*mcc*) может трансформировать MATLAB-файлы (расширение **.m*) в файлы языка *C* (но не обратно). Это очень удобно, когда нужно запустить программы на компьютере, где не установлен пакет MATLAB или когда разработчику программы не хочется, чтобы кто-нибудь мог посмотреть коды программы.

При создании исполняемых MEX-файлов существуют следующие ограничения:

1. Нельзя компилировать *Script*-файлы, все компилируемые файлы должны быть функциями.
2. Нельзя компилировать файлы, использующие объекты.
3. Нельзя компилировать файлы, содержащие ASCII-переменные для загрузки в рабочую область и для сохранения.
4. Нельзя компилировать файлы, содержащие русскоязычные комментарии.
5. Для запуска MEX-файла на компьютере, где MATLAB не установлен, необходимо наличие всех библиотек (*dll*-файлы из каталога *bin)*, а это занимает около 40 Мб.

Таким образом, *mex*-файлы можно создавать для обработки данных, полученных от какой-либо записывающей аппаратуры (бинарные файлы, файлы, полученные с помощью записей DAQ-карты).

Главная трудность при написании компилируемых программ состоит в создании удобного графического интерфейса, ведь MEX-файлы не работают с объектами и с глобальными переменными. Поэтому надо позаботиться о

том, чтобы созданный в главной функции *function proba()* интерфейс, заданный, как показано на рис. 10.1, и оси, заданные с помощью команды *A1=axes('Position',[0.25 0.55 0.7 0.35],...*

'Box','On',...

'XGrid','On',...

'YGrid','On',...

'UserData','A1'...

);

различались *call-back* функцией *function proba_cb(action).* Для этого в *call-back* функции используются команды :

A1=findobj('UserData','A1');

set(A1,'UserData','A1','XGrid','On','YGrid','On');

и после каждого изменения графического объекта (например, вывода графика) команда *set(A1,'UserData','A1','XGrid','On','YGrid','On');* повторяется опять.

Для открытия либо DAQ, либо бинарного файла, используется переход

swich

case

end

Для открытия файла, записанного DAQ-картой, используются следующие команды:

fid=fopen(NameOFfile,'rb','b');

Channels=fread(fid,8,'single'); - количество каналов,

N_ch=fread(fid,1,'single'); - количество отсчетов, соответствующих одному каналу,

Rate=fread(fid,1,'single'); - частота дискретизации,

Signals=fread(fid,[N_ch sum(Channels)],'single'); - двумерный массив данных, в которых столбец соответствует каналу.

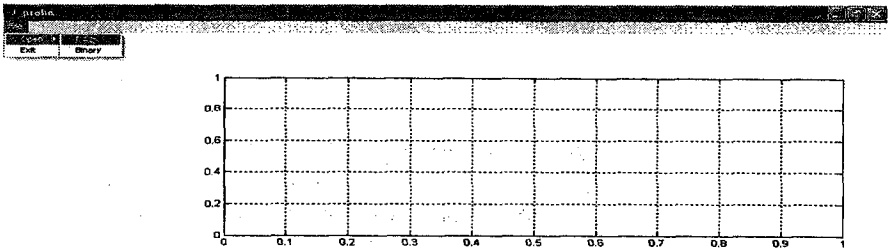


Рисунок 10.1 Интерфейс исполняемой программы

После написания и сохранения функций в рабочем окне MATLAB, в командной строке набирается команда

mcc -m -B sgl proba.m proba_cb.m;

в которой через пробел перечисляются все функции проекта. Эта команда создает MEX-файл в выбранной заранее директории MATLAB, который можно затем запустить как EXE-файл.

Задание для выполнения работы

Создать в среде MATLAB исполняемую программу, позволяющую открывать бинарные файлы и файлы, записанные с помощью DAQ-карты. Пример реализации открытия исполняемой программой файлов представлены на рисунках 10.2 и. 10.3.

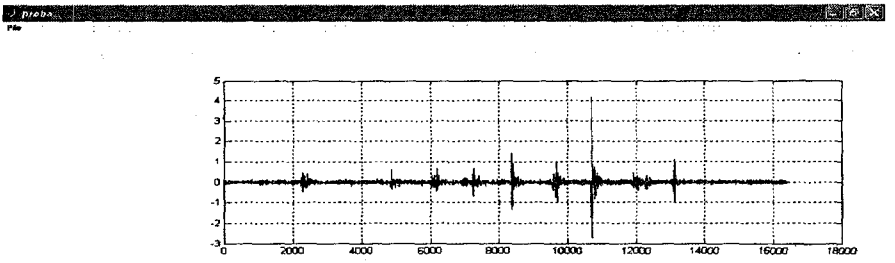


Рис. 10.2 Файл DAQ-карты

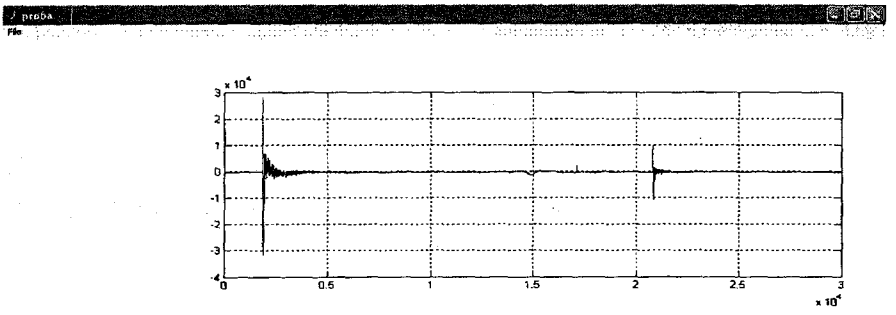


Рис. 10.3 Бинарный файл

Работа № 11

Исследование временных характеристик сигналов

Цель работы: исследование временных характеристик сигнала с использованием графического интерфейса.

Задание для выполнения работы

Создать графический интерфейс, показанный на рис. 11.1, и CallBack–программу к нему для расчета временных характеристик заданного сигнала

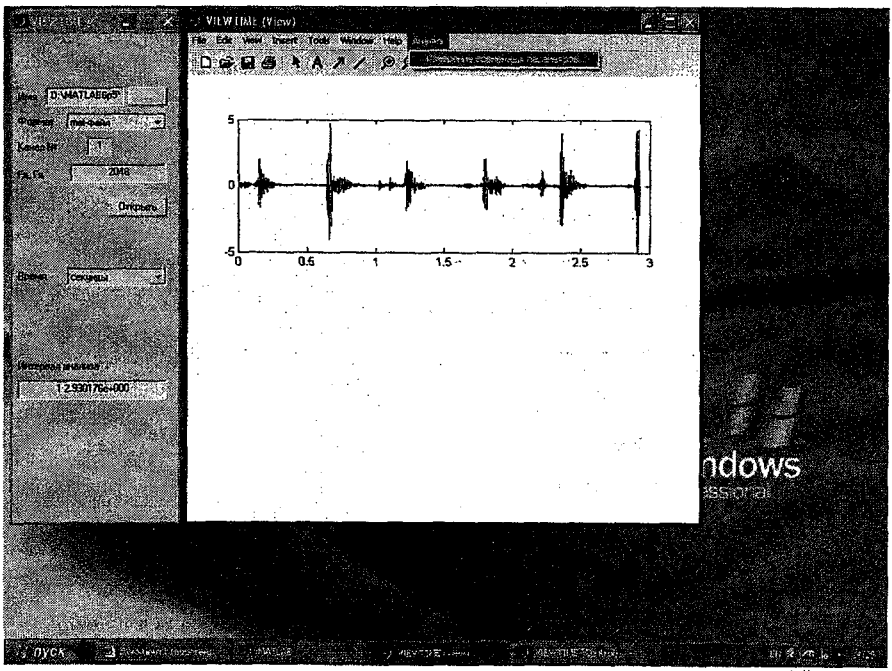
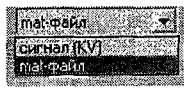


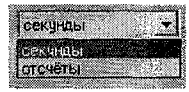
Рис. 11.1.Графический интерфейс

Рекомендации по выполнению работы

Графический интерфейс состоит из двух панелей: контрольной, изображенной на рис. 11.2, и панели просмотра, изображенной на рис. 10.3.

На контрольной панели необходимо создать кнопку  , при нажатии на которую в CallBack-функции вызывается команда *uigetfile*, выводящая в текстовом окне путь к файлу.

Полосы прокрутки 



можно осуществить с помощью команды

H0_5=uicontrol('Style','PopUpmenu','Units','normalized','Position',
[0.33 0.8 0.6 0.035],'String','сигнал (KV)|mat-файл','Value',1);

В CallBack-функции команды, соответствующие первому названию, будут выполняться по командам

TT=get(var, 'Value');
If (TT= =1);.....

На панели просмотра необходимо предусмотреть меню «Измерение временных параметров» с помощью команды *makemenu.*

При обращении к CallBack- функции, соответствующей этой операции, можно запустить другую программу со своим графическим интерфейсом и своими callback-функциями.

Например, команды

elseif (NumOFCBFun==5);
timelim;
end

запускают программу *timelim* (рис. 11. 4) *rco* своими callback-функциями *timelimcb.*

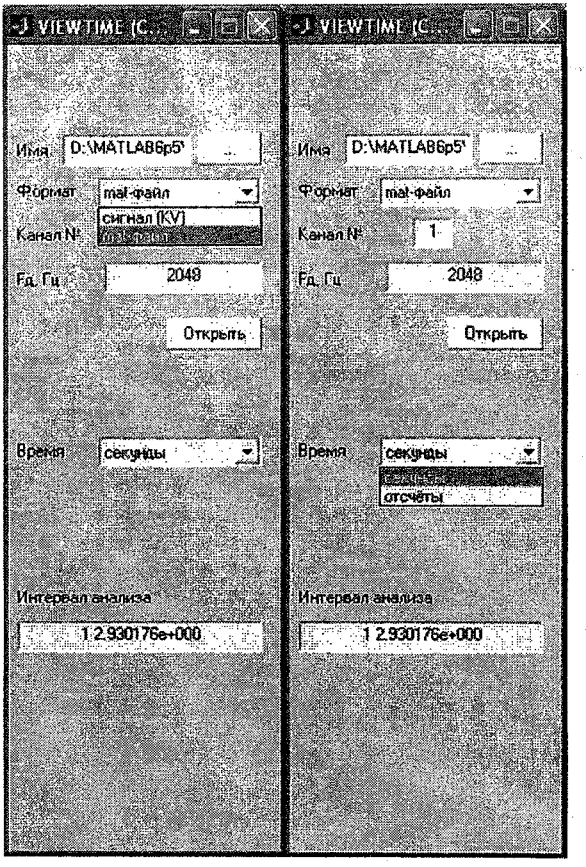


Рис. 11.2. Графический интерфейс контрольной панели

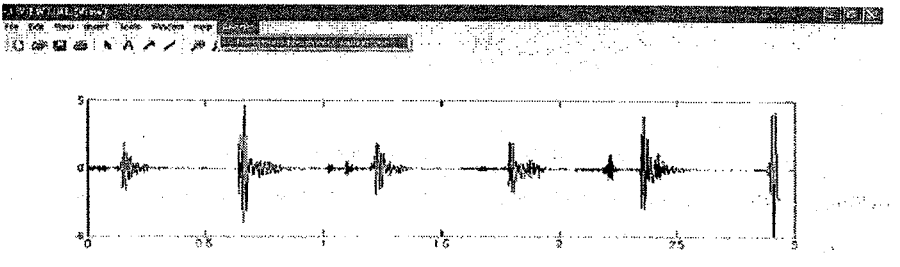


Рис. 11.3.Графический интерфейс панели просмотра

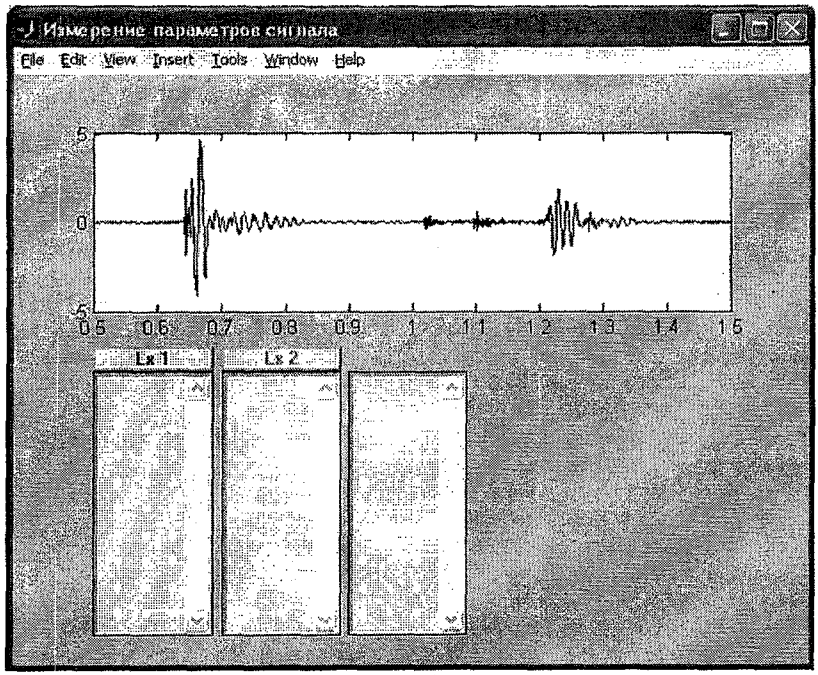


Рис. 11.4.Программа timelim

Кнопки *Lx*1 и *Lx*2 имеют стиль *Pushbutton* и вызывают callback-функции. Например,

B1=uicontrol('Style','PushButton','Units','normalized','Position',[0.1 0.5 0.15 0.04],'String',

'Lx 1','FontWeight','Bold','Foregroundcolor','Red','Callback','NumCB=1; timelimcb;');

В соответствующей callback-функции NumCB=1 идет обращение к команде *ginput* – графическому вводу с помощью мыши. Синтаксис команды

[X,Y] = GINPUT(N).

Она рассчитывает N точек отсчетов и выдает их X и Y координаты. Курсор устанавливается с помощью мыши.

Координаты *X* начальной характерной точки, конечной характерной точки и разность между ними (время в секундах) показываются в трех полосах прокрутки, имеющих стиль *ListBox* , например, *E*3=uicontrol('Style','ListBox','Units','normalized','Position',[0.1 0.05 0.15 0.45]); поскольку стиль *ListBox* требует данные типа *String*, то можно организовать в главной программе *timelim* массивы ячеек:

Mas_lx1=cell([128,1]); массив ячеек для первой временной точки (количество точек =128)

Mas_lx2=cell([128,1]); массив ячеек для второй временной точки (количество точек =128)

Mas_dt=cell([128,1]); массив ячеек для подсчета разности времени

NumOFLx1=1; номер ячейки в *Lx*1

NumOFLx2=1; номер ячейки в *Lx*2

NumOFdt=1; номер ячейки разности времен,

а callback-функцию написать так: например, для *Lx1*

```
if(NumCB==1)
  [lx1,ly1]=ginput(1);
  str_lx1=num2str(lx1);
  str_ly1=num2str(ly1);
  Mas_lx1{NumOFLx1}=str_lx1;
  set(E3,'String',Mas_lx1);
  axes(A1_1);
  yl=ylim;
  l1=line([lx1 lx1],[yl(1) yl(2)]);
  set(l1,'Color','Red','LineWidth',1,'EraseMode','xor');
  i1=get(E3,'Value');
  i1=i1+1;
  set(E3,'Value',i1);
```

NumOFLx1=NumOFLx1+1;

В полосах прокрутки должны появиться числа, соответствующие началу и концу временного интервала и его длительность, как это показано на рис. 11.5 .

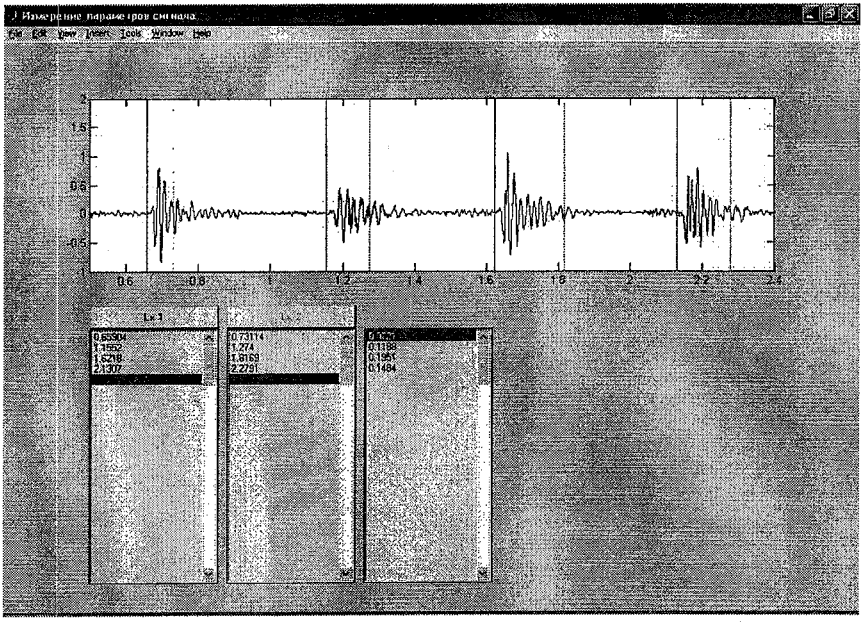


Рис. 11.5.Полоса прокрутки

Методика построения двумерной гистограммы

***Цель работы*:** построение гистограммы двух сигналов в трехмерном пространстве.

При исследовании сигналов часто бывает необходимо получить условные плотности распределения вероятностей двух сигналов, т.е. построить двумерную гистограмму этих сигналов в трехмерном пространстве. Для сигналов, изображенных на рисунке 12.1, такая гистограмма приведена на рис. 12.2.

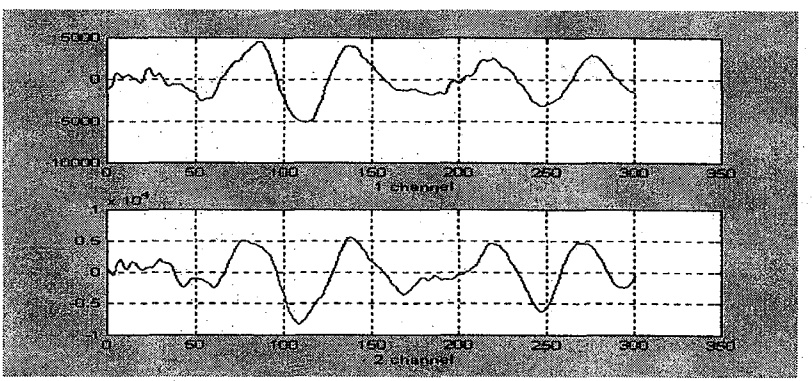


Рис. 12.1.Сигналы для построения гистограммы

Для построения гистограммы мгновенных значений исследуемых отрезков двух сигналов необходимо:

- определить максимальное значение амплитуды, встречающееся на исследуемых отрезках сигналов;
- определить минимальное значение амплитуды, встречающееся на исследуемых отрезках сигналов;

- весь диапазон значений амплитуд от минимального до максимального значения разбить на интервалы группирования, число интервалов задается в диалоговом режиме;
- для каждого интервала группирования произвести подсчет количества попаданий в него мгновенных значений амплитуд первого и второго сигналов. Таким образом заполняется некоторая матрица (назовем ее, для примера *M*). Если значение первого сигнала попало в первый интервал, и значение второго сигнала попало в первый интервал, то к значению *M*(1,1) добавляется 1. Если, например, значение первого сигнала попало во второй интервал, а значение второго сигнала попало в пятый интервал, то 1 добавляется к значению *M*(2,5);

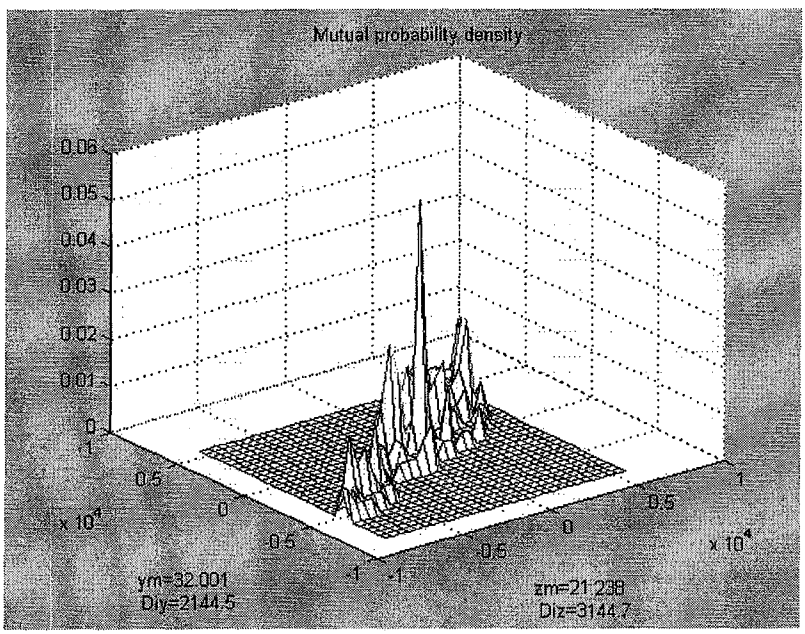


Рис. 12.2.Двумерная гистограмма

- для получения плотности вероятности нужно нормировать число попаданий к общему числу исследуемых отсчетов сигнальной реализации в каждой ячейке полученной матрицы;
- построить полученную плотность вероятности в трехмерном пространстве с помощью команды *mesh* или *surf*, привязав координатные оси к значениям амплитуд в интервалах группирования;
- определить параметры полученного распределения вероятностей. Математическое ожидание считается по формуле: $E(x) = \sum xf(x)dx$, где x – среднее значение амплитуды сигнала в данном интервале группирования, $f(x)$ – значение ПРВ; дисперсия расчитывается отдельно для осей по формуле: $Д(x) = (x - \mu)^2$, где μ - математическое ожидание. Ассиметрия распределения рассчитывается для каждой оси с помощью команды *skewness(x)*, эксцесс распределения также расчитывается для каждой оси с помощью команды *kurtosis(x)*.

Задание для выполнения работы

Построить гистограмму двух сигналов в трехмерном пространстве и определить параметры полученного распределения вероятностей.

Работа № 13

Определение связности двух сигналов

Цель работы: разработка программы определения связности двух сигналов

Если получена условная плотность вероятности $W(x,y)$, то есть возможность рассчитать ряд статистических расстояний, показывающих, насколько совместная ПРВ двух сигналов удалена от произведения независимых ПРВ $W(x)W(y)$, и установить, насколько сигналы x и y связаны друг с другом. Для независимых сигналов $W(x,y)=W(x)W(y)$, для зависимых сигналов $W(x,y)\neq W(x)W(y)$. Наиболее часто для определения статистической связи двух сигналов используют расстояние Кульбака, в котором используется двоичный логарифм:

$$\rho_1(x,y)=\int\log\frac{w(x,y)}{w(x)w(y)}w(x,y)dxdy, \qquad (13.1)$$

Методика определения расстояния Кульбака в пакете MATLAB такова:

1. Получение матрицы ПРВ двух сигналов.

2.Получение одномерных ПРВ отдельных сигналов

3.Перемножение одномерных ПРВ и получение матрицы произведения одномерных ПРВ; поскольку это произведение стоит в знаменателе формулы, а делить на 0 нельзя, то для расчета все нули в матрице должны быть заменены на очень малую величину, например, 0,00000000001.

4.Деление и умножение в формуле (13.1) производится почленно (операторы с точкой)

5.Интеграл заменяется суммированием.

Полученное в результате вычислений число представляет собой количество битов информации и оно тем больше, чем больше связность двух сигналов. Обычно считают, что если полученное число $a \geq 1$, то сигналы между собой связаны.

Таким образом, если один из сигналов, например, x считать опорным, а второй сигнал y считать тестируемым, то расстояние Кульбака можно применить для построения автоматического обнаружителя сигналов.

Пусть у нас есть запись шагов человека. Выберем некоторый отрезок записи, зафиксировавший шаг, в качестве опорного сигнала. Характерным признаком этого сигнала, который улавливает человеческий глаз, является его форма. Используем этот признак для построения автоматического обнаружителя шагов человека. Построим огибающую опорного сигнала, нормируем ее к своему максимальному значению, получив таким образом параметр «форма огибающей». Разбив второй тестируемый сигнал на отрезки такой же длины и строя на каждом отрезке нормированную огибающую, можно найти расстояние Кульбака для каждого отрезка по параметру «форма огибающей» и, произведя его сравнение с порогом 1, принять решение о том, есть шаг человека в исследуемом сигнале или нет.

Визуализация результатов обычно производится на графике в окне *figure*. Для установки постоянного окна используется команда *set*. Пример использования команды:

```
o1=figure(1);
set(o1,'Position',[5,50,560,442]);
mesh(a,a,S);
```

Задание для выполнения работы

1.Для двух сигналов написать программу для вычислению расстояния Кульбака.

2.Написать программу, моделирующую работу обнаружителя шагов человека с использованием критерия Кульбака.

Список рекомендованной литературы

1 Лазарев Ю.Ф. MatLAB 5.x. Издательская группа BHV. Серия «Библиотека студента».- СПб.,2000.-384 с.

2 Мартынов Н.Н., Иванов А.П. MatLAB 5.x. Вычисления, визуализация, программирование.-М.: Изд-во «КУДИЦ-ОБРАЗ», 2000.-336 с.

3 Яковлев В. MATLAB 6/6.5+Simulink 4/5 в математике и моделировании.-СПб.: Изд-во «Солон-Пресс»,2003.- 576 с.

4 Потемкин В.Г. MatLAB 5 для студентов. Диалог МИФИ Справ. пособие.- М.: 1998.-314 с.

5 Шишкин А.Д. Программирование на языке Си. Учебное пособие.- СПб.: изд. РГГМУ, 2003.-104 с

Содержание

Учебное издание

Лабораторный практикум
«Введение в MATLAB»

Чернецова Елена Анатольевна

Редактор – О.С. Крайнова

Отпечатано с готового оригинал-макета в ЦНИТ «АСТЕРИОН»
Заказ № 32. Подписано в печать 14.02.2006 г. Бумага офсетная.
Формат 60x84 1/16. Объем 5,5 п. л. Тираж 100 экз.
Санкт-Петербург, 191015, а/я 83,
тел. /факс (812) 275-73-00, 275-53-92, тел. 970-35-70
www.asterion.ru E-mail: asterion@ asterion.ru