

**Министерство науки и высшего образования Российской Федерации
ФГАОУ ВО «Севастопольский государственный университет»**

**Институт информационных технологий
и управления в технических системах**

ВВОДНОЕ ЛАБОРАТОРНОЕ ЗАНЯТИЕ

«Освоение интерфейса MATLAB»

по дисциплине «Теория вероятностей, вероятностные процессы
и математическая статистика»

для студентов всех форм обучения направления подготовки:

09.03.02 – «Информационные системы и технологии»,

09.03.03 – «Прикладная информатика»

Севастополь

2022

СОДЕРЖАНИЕ

1	Цель работы	3
2	Краткие теоретические сведения	3
2.1	Что такое Matlab?	3
2.2	Интерфейс Matlab	5
2.3	Справочная система Matlab	7
2.4	Переменные и операторы Matlab	9
2.5	Векторы и матрицы.....	11
2.6	Графика в Matlab	16
2.7	Создание функций в Matlab	23
3	Порядок выполнения работы	26
4	Содержание отчета.....	27
5	Контрольные вопросы	27

ВВОДНОЕ ЛАБОРАТОРНОЕ ЗАНЯТИЕ «ОСВОЕНИЕ ИНТЕРФЕЙСА MATLAB»

1 Цель работы

Получение общего представления о математическом пакете Matlab – особенностей интерфейса, функциональных основных возможностей, формирования навыков практической работы в среде Matlab, математических вычислений, моделирования, разработки приложений и анализа данных.

2 Краткие теоретические сведения

2.1 Что такое Matlab?

Matlab – это высокопроизводительный язык для технических расчетов, выполняемых на ЭВМ. Он включает в себя вычисления, визуализацию и программирование в удобной среде, где задачи и решения выражаются в форме, близкой к математической.

Система Matlab (сокращение от MATrix LABoratory – МАТричная ЛАБОратория) разработана фирмой The MathWorks, Inc, США, г. Нейтик, шт. Массачусетс). Она распространяется более 10 лет и постоянно совершенствуется. Современные версии Matlab допускают возможность обращения к программам, написанным на языках C и C++.

Наиболее часто Matlab используется для:

- математических вычислений;
- создания алгоритмов;
- моделирования;
- анализа данных, исследования и визуализация;
- научной и инженерной графики.

Система Matlab состоит из пяти основных частей:

1) *язык Matlab*. Это язык матриц и массивов высокого уровня с управлением потоками, функциями, структурами данных, вводом-выводом, обладающий особенностями объектно-ориентированного программирования. Это позволяет программировать как в "небольшом" масштабе для быстрого создания черновых программ, так и в "большом" для создания больших и сложных приложений;

2) *среда Matlab*. Это набор инструментов и приспособлений, с которыми работает пользователь или программист Matlab. Она включает в себя средства для управления переменными в рабочем пространстве Matlab, вводом и выводом данных, а также создания, контроля и отладки м-файлов и приложений Matlab;

3) *управляемая графика*. Это – графическая система Matlab, которая включает в себя команды высокого уровня для визуализации двух- и трехмерных данных, обработки изображений, анимации и иллюстрированной графики. В нее входят также команды низкого уровня, позволяющие полностью редактировать внешний вид графики, так же, как при создании Графического Пользовательского Интерфейса (GUI) для Matlab-приложений;

4) *библиотека математических функций*. Это обширная коллекция эффективных вычислительных алгоритмов, начиная от элементарных функций (сумма, тригонометрические функции, комплексная арифметика) и кончая сложными (обращение матриц, нахождение собственных значений и собственных векторов, функции Бесселя, преобразование Фурье и т.д.);

5) *программный интерфейс*. Это библиотека, которая позволяет писать программы на C и FORTRAN, которые взаимодействуют с Matlab. Она включает средства для вызова программ из Matlab (динамическая связь), вызывая Matlab как вычислительный инструмент и для чтения-записи MAT-файлов;

В системе по умолчанию реализована комплексная арифметика, вычисления производятся с двойной точностью, базовый элемент – массив.

2.2 Интерфейс Matlab

При запуске системы открывается рабочий стол (desktop) системы (рисунок 1).

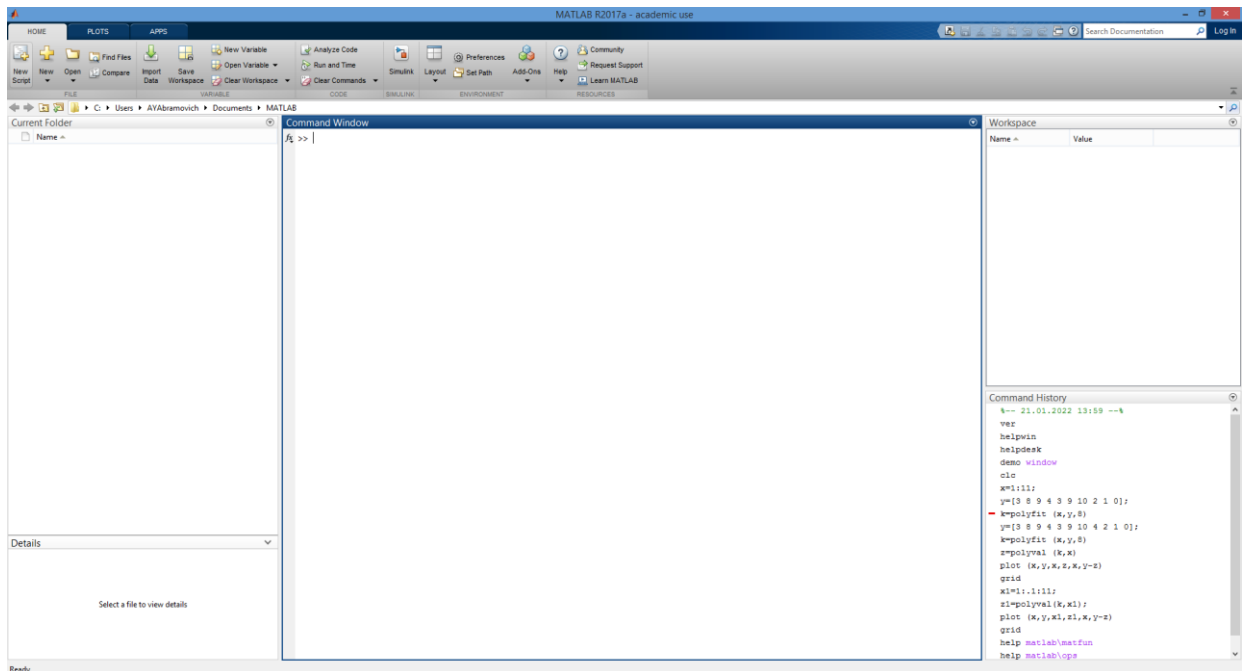


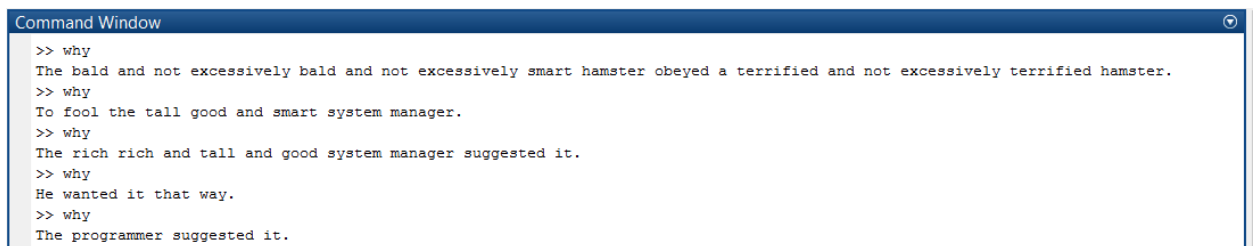
Рисунок 1 – Рабочий стол системы

Интерфейс рабочего стола содержит:

- меню, расположенное в верхней части окна – содержит вкладки HOME, PLOTS, APPS;
- *адресную строку* – указатель рабочего каталога, позволяющую выбрать рабочую директорию;
- окно в центре – *Command Window*, которое предназначено для ввода чисел, переменных, выражений и команд, для просмотра результатов вычислений, для отображения текстов выполняемых программ, а также для вывода сообщений об ошибках;
- окно слева – *Current Folder* в котором отображаются файлы текущего каталога и определенные данные описания их свойств;
- окно справа – *Workspace*, представляющие все глобальные переменные, использованные в командах или при выполнении программ;
- окно справа внизу – *Command History*, в котором все команды, выполненные в командной строке.

При интерактивной работе в командном окне все команды и их последовательности помещаются в строку ввода, она начинается символом `>>`. Исполняются команды после нажатия клавиши *Enter*. А отделяются команды друг от друга *запятой или точкой запятой*. Если использовать разделитель точку с запятой, то результат выполнения команды не отображается.

На рисунке 2, представлены результаты исполнения команды `why` в командном окне.



```

>> why
The bald and not excessively bald and not excessively smart hamster obeyed a terrified and not excessively terrified hamster.
>> why
To fool the tall good and smart system manager.
>> why
The rich rich and tall and good system manager suggested it.
>> why
He wanted it that way.
>> why
The programmer suggested it.
  
```

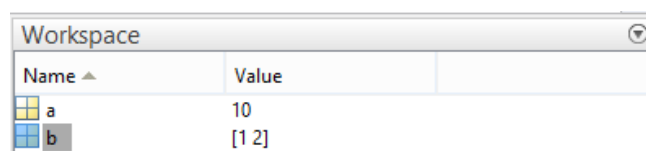
Рисунок 2 – Пример выполнения программы

Выполненные команды помещаются в стек и могут быть извлечены в строку ввода перебором исполненных команд с помощью стрелок $\uparrow\downarrow$ и при необходимости редактируются при повторном исполнении. Строка вывода не доступна для редактирования.

Для очистки рабочего окна используется команда `clc`.

Все переменные среды – *глобальные*. Это может стать причиной ошибок, если какие-то переменные уже ранее были определены и их используют повторно. Поэтому необходимо *внимательно контролировать процесс идентификации и использования переменных*.

Workspace – рабочее пространство; окно, содержащее информацию обо всех переменных, типе, значениях (рисунок 3).



Name	Value
a	10
b	[1 2]

Рисунок 3 – Окно Workspace

Щелчком по пиктограмме переменной активируется редактор переменных (VE), позволяющий изменять их значения в интерактивном режиме. Этот прием работы отображен на рисунке 4.

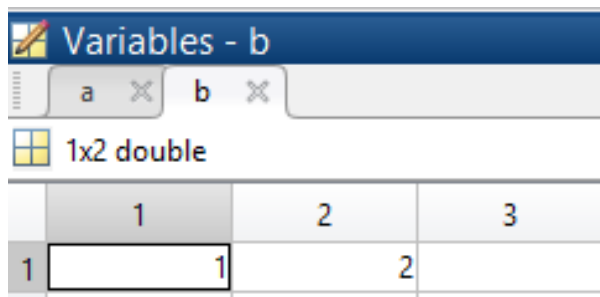


Рисунок 4 – Окно редактора переменных

Для очистки рабочего пространства используется команда `clear`. Для очистки конкретной переменной необходимо указать ее имя `clear <имя переменной>`.

2.3 Справочная система Matlab

Существуют следующие способы получить информацию о функциях системы Matlab в процессе сеанса работы:

- команда `help`;
- команда `doc`;
- меню Help;
- обращение к Web-серверу фирмы The MathWorks.

Для обращения к справочной системе необходимо в командном окне набрать одну из команд: `help` (выводит на экран список каталогов, как показано на рисунке 5) или же `doc` (открывает окно документации (рисунок 6)).

Чтобы отобразить справку для функции, метода, класса, тулбокса или переменной, необходимо задать имя функции (метода, класса и т.д.) периодом. Например, на рисунке 7 отображена справка для функции `plot`.

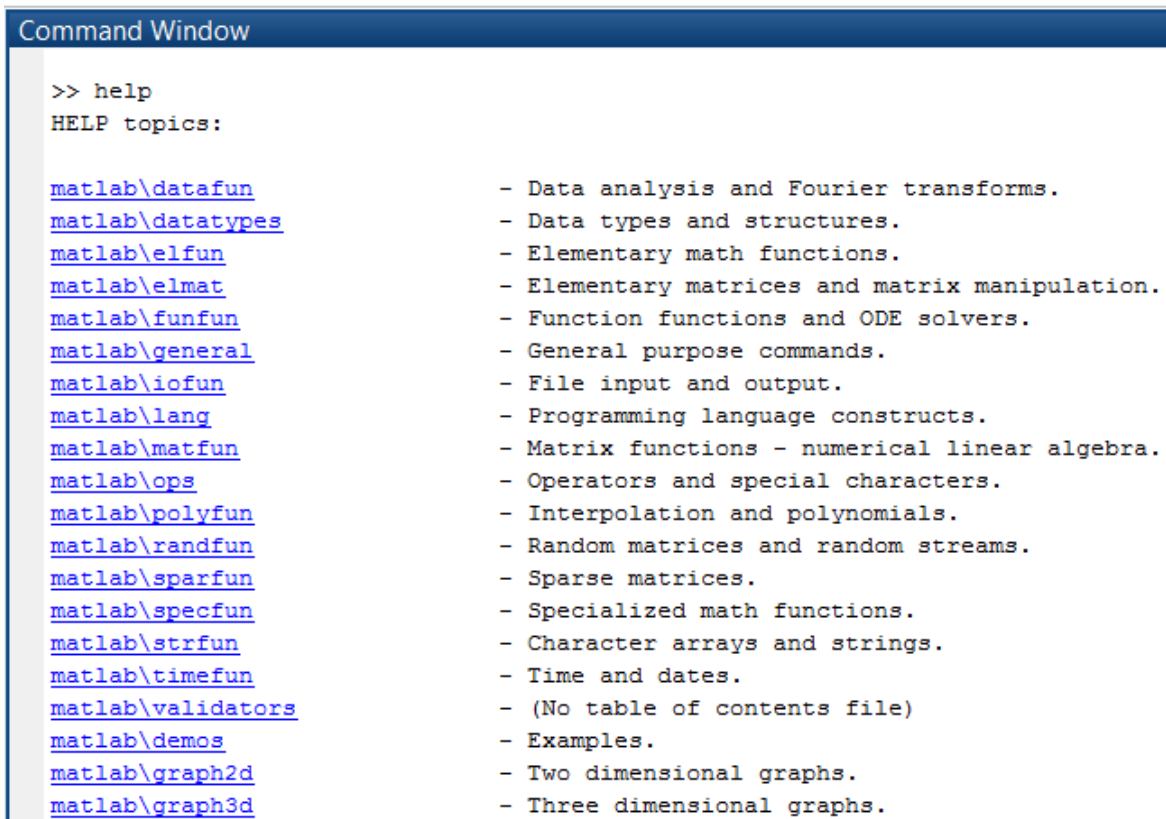


Рисунок 5 – Выполнение команды help

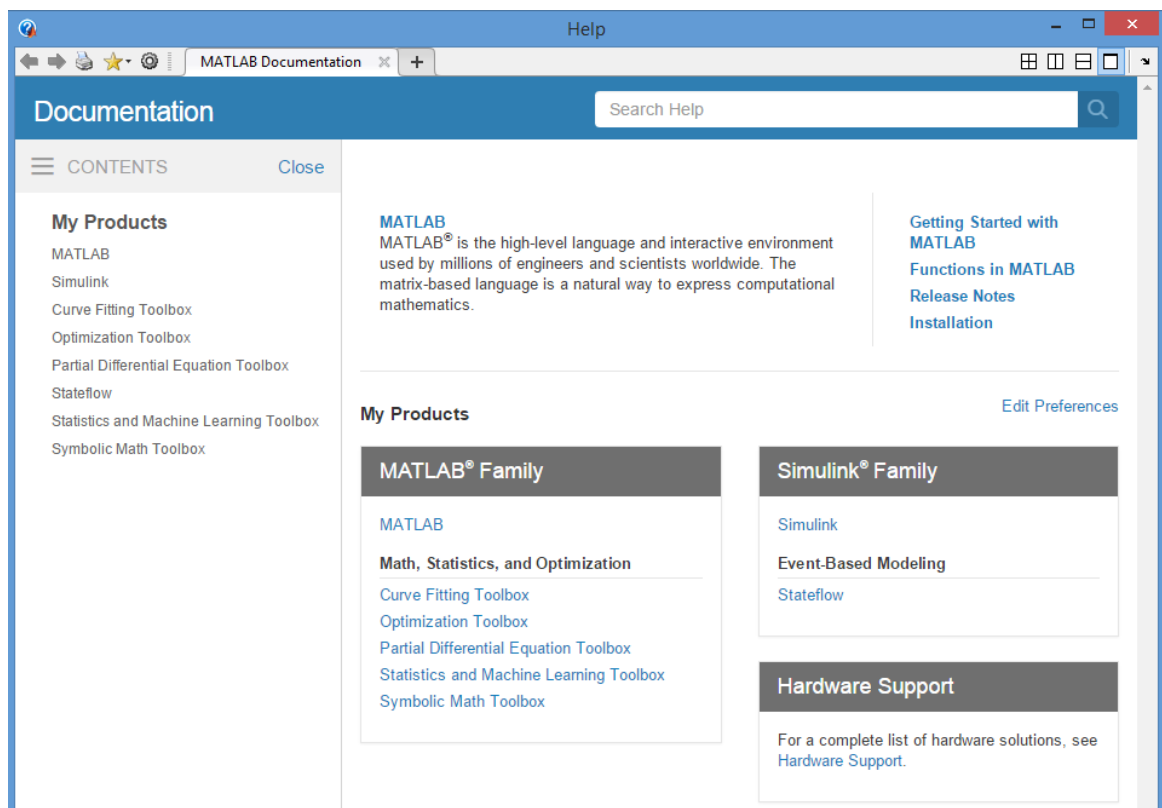


Рисунок 6 – Окно документации


```

Command Window

>> help plot
plot    Linear plot.
plot(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
then the vector is plotted versus the rows or columns of the matrix,
whichever line up. If X is a scalar and Y is a vector, disconnected
line objects are created and plotted as discrete points vertically at
X.

plot(Y) plots the columns of Y versus their index.
If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)).
In all other uses of plot, the imaginary part is ignored.

Various line types, plot symbols and colors may be obtained with
plot(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:

      b    blue      .    point      -    solid
      g    green     o    circle     :    dotted
      r    red       x    x-mark     -.   dashdot
      c    cyan      +    plus       --   dashed
      m    magenta   *    star       (none) no line
      y    yellow    s    square
      k    black     d    diamond
      w    white     v    triangle (down)
                        ^    triangle (up)
                        <    triangle (left)
                        >    triangle (right)
                        p    pentagram
                        h    hexagram

```

Рисунок 7 – Справочная информация

Аналогично вызывается справка для отдельной функции (метода, класса и т.д.) с использованием команды `doc` (`doc plot`, `doc sum` и пр.).

Как и во многих других приложениях, примеры справки доступны для копирования с последующим выполнением в рабочей среде.

2.4 Переменные и операторы Matlab

Переменная в Matlab – это массив (матрица), вектор или скаляр. Matlab не требует какого-либо описания типа переменной или размерности массива.

Имена переменных, констант и функций могут быть составлены из любых символов латинского алфавита, кроме специальных и цифр, начинаются с буквы.

Системные константы:

- `pi` = 3.14159265358979;
- `i` = мнимая единица, то же `j`;
- `Inf` = бесконечность, результат деления на 0;

- NaN = неопределенное значение (Not a Number), 0/0, inf-inf;
- $\text{eps} = 2^{-52}$ или $2.2204\text{e-}016$ – характеризует точность вычислений с плавающей точкой.

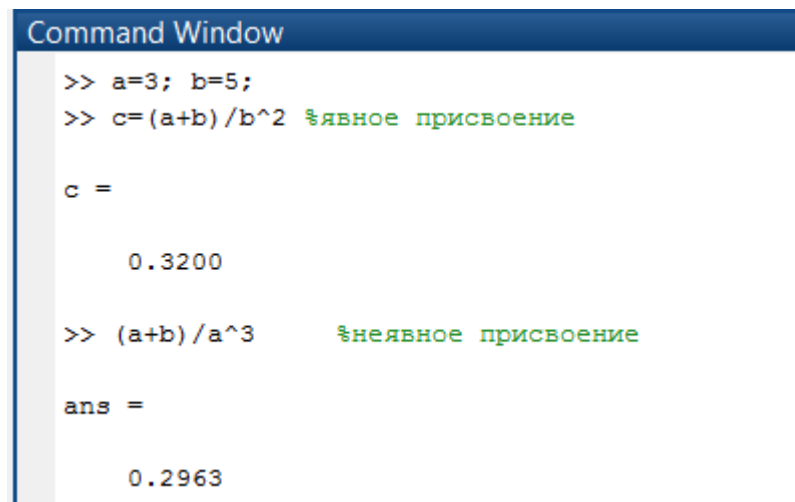
Язык Matlab это язык операторов. Операторы задаются по одному в командной строке для исполнения в интерактивном режиме или в виде списка в m-файле или в script-файле. Важно, что признаком начала командной строки является указатель `>>`, выводимый системой автоматически, при готовности к приему команды. *В тексте программ этот символ не нужен!*

Две формы записи операторов:

- с явным присвоением: *переменная = выражение*;
- с неявным присвоением: *выражение*.

При записи оператора с неявным присвоением, результат вычислений присваивается автоматически внутренней переменной `ans` (answer), может быть вызван и использован по этому имени и сохраняется до выполнения следующего оператора с неявным присвоением.

Примеры записи с явным и неявным присвоением отображены на рисунке 8.



```

Command Window
>> a=3; b=5;
>> c=(a+b)/b^2 %явное присвоение

c =

    0.3200

>> (a+b)/a^3 %неявное присвоение

ans =

    0.2963
  
```

Рисунок 8 – Запись операторов

Все встроенные элементарные функции должны *записываться в программах малыми буквами!*

2.5 Векторы и матрицы

Ввод векторов и матриц.

Основными объектами, с которыми начинает работать пользователь, знакомящийся с Matlab, являются матрицы. Если проверить с помощью команды `size` размер числа 5, или символа 'A', то в результате получится два числа – количество строк и количество столбцов, в данном случае – это две единицы.

Для ввода векторов и матриц используются квадратные скобки []. Для разделения данных в векторах и матрицах в строке служат пробел и запятая, и точка с запятой – в столбце.

Пример 1– Задание векторов

```
>> % Вектор-строка
>> a1=[1 2 3]

a1 =

     1     2     3
>> % Вектор-столбец
>> a3=[1;2;3]

a3 =

     1
     2
     3
```

Пример 2 – Задание матриц

```
>> % Матрица, размера 2x3
>> b1=[1 2 3; 4 5 6]

b1 =

     1     2     3
     4     5     6

>> % Матрица, размера 3x2
>> b2=[1 2; 3 4; 5 6]

b2 =

     1     2
     3     4
     5     6
```

Значения вектора можно задать с помощью следующей конструкции:

[начальное значение : шаг : конечное значение] ИЛИ [начальное значение : : конечное значение] тогда шаг по умолчанию равен единице.

Квадратные скобки в этом выражении можно опустить.

Пример 3 – Задание вектора и вычисление вектора

```
>> % Вектор x
>> x=0:0.75:6

x =
    0    0.7500    1.5000    2.2500    3.0000    3.7500
4.5000    5.2500    6.0000

>> % Вычисление вектора y
>> y=cos(x)

y =
    1.0000    0.7317    0.0707   -0.6282   -0.9900   -0.8206
-0.2108    0.5121    0.9602
```

Помимо этого, значения вектора можно задать функцией:

```
x = linspace(начальное значение, конечное значение),
```

(можно пользоваться при создании линейного массива). При этом вектор x по умолчанию будет содержать сто компонент. Возможен и другой способ вызова функции `linspace` – с тремя входными параметрами, последним из которых является количество компонент вектора.

В таблице 1 приведены функции, позволяющие создать некоторые специальные и часто используемые матрицы.

Таблица 1 – Функции создания специальных матриц

Функция	Описание
<code>ones(n)</code> <code>ones(n,m)</code>	Создание матрицы n -го порядка (или же размера $n \times m$) заполненной единицами
<code>zeros(n)</code> <code>zeros(n,m)</code>	Создание матрицы n -го порядка (или же размера $n \times m$) заполненной нулями. Обычно команду <code>zeros</code> используют для инициализации матриц

Продолжение таблицы 1

Функция	Описание
eye (n) eye (n, m)	Создание матрицы n-го порядка (или же размера n x m) с единицами на главной диагонали и нулях на остальных позициях
magic (n)	Формирует матрицу Альбрехта Дюрера (магический квадрат) – матрица знаменита тем, что суммы элементов в строках, столбцах и диагоналях одинаковы.

Создание матриц, заполненных случайными числами.

Существует несколько функций, позволяющих заполнять матрицы случайными числами:

1) функция `rand(n)` формирует массив размера $n \times n$, элементами которого являются случайные величины, распределенные по равномерному закону в интервале (0, 1);

2) функция `randi([imin, imax], n)` формирует массив n-го порядка с элементами в диапазоне от `imin` до `imax`.

Пример 4 – Использование функции `rand`

```
>> % Матрица 3-го порядка,
>> % заполненная вещественными случайными числами
>> % с равномерным распределением из открытого интервала (0,1)
>> A=rand(3)

A =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
>> % Матрица 2x3,
>> A=rand([2 3])

A =
    0.9649    0.9706    0.4854
    0.1576    0.9572    0.8003
```

Пример 5 – Использование функции `randi`

```
>>% Матрица 7-го порядка с элементами в диапазоне от -15 до 3
>> A=randi([-15 3],7)

A =
```

14

-13	1	-12	-2	0	-10	-9
-7	2	-2	-9	-12	-3	-4
2	-3	-15	3	-6	-3	-11
0	-1	-10	-15	-7	-12	-1
3	-1	-15	-7	-3	-13	-11
-3	-8	-14	-8	-2	-6	-6
-15	-3	0	-1	-1	3	-2

```
>> % Матрица 2x4 с элементами в диапазоне от -3 до 3
```

```
>> B=randi([-3 3],2,4)
```

B =

-1	2	0	1
-2	-1	-2	-2

Обращение к элементам матрицы.

Для обращения к элементам матрицы используют круглые скобки (). Первый индекс – номер строки, второй – номер столбца. Возможно задавать диапазон строк-столбцов, используя для этого двоеточие :.

Пример 6 – Задание матрицы и обращение к ее элементам

```
>> % Задание матрицы
```

```
>> A=[1 2 3; 4 5 6; 7 8 9]
```

A =

1	2	3
4	5	6
7	8	9

```
>> % Изменение 1-го элемента матрицы
```

```
>> A(1,1)=100
```

A =

100	2	3
4	5	6
7	8	9

```
>> % Изменение 3-й строки матрицы
```

```
>> A(3,:)=50
```

A =

100	2	3
4	5	6
50	50	50

```
>> % Изменение 2-го столбца матрицы
>> A(:,2)=33

A =
```

```
100    33     3
   4    33     6
   50    33    50
```

В примере, представленном выше, знак двоеточие обозначает, что в рассмотрение берутся все элементы.

Пример 7 – Изменение фрагмента матрицы

```
>> % Очистка переменной A
>> clear A
>> % Задание квадратной матрицы 3-го порядка из единиц
>> A=ones(3)
```

```
A =

     1     1     1
     1     1     1
     1     1     1
```

```
>> % Изменение фрагмента матрицы
>> A(1:2,1:2)=55
```

```
A =

    55    55     1
    55    55     1
     1     1     1
```

В примере 7 значения элементов строк и столбцов с 1 по 2 заменяются на 55.

Удаление элементов матрицы.

Удалить из матрицы можно строку или столбец целиком. Для удаления строки или столбца необходимо присвоить удаляемому элементу пустой массив, используя для этого пустые квадратные скобки [].

Матричные операции.

В Matlab определены матричные операции *по правилам линейной алгебры*: при сложении и вычитании должны совпадать размерности, при

умножении и делении число столбцов первого матричного сомножителя и число строк второго должны совпадать.

В таблице 2 приведены часто применяемые матричные функции.

Таблица 2 – Матричные функции

Функция	Описание
sum (A) sum (A, 2)	Сумма всех элементов вектора A. Если A – матрица, то суммирование элементов произойдет по столбцам. Для суммирования по строкам, необходимо указать второй параметр в функции sum, а именно 2
sum (sum (A))	Сумма всех элементов матрицы
prod (A) prod (A, 2) prod (prod (A))	Произведение элементов вектора (матрицы). Аналогично функции sum
diag (A)	Позволяет выделить диагонали матрицы A
size (A)	Возвращающая количество строк и количество столбцов матрицы A
numel (A)	Возвращает общее количество элементов матрицы A
fliplr (A)	Отражение матрицы A
rot90 (A)	Поворот матрицы A

При проведении операций с матрицами нужно помнить приоритет операций: сначала выполняется операция транспонирования, затем возведения в степень, потом умножение и деление, а в последнюю очередь – сложение.

2.6 Графика в Matlab

Система Matlab предоставляет огромное количество графических средств. К ним относятся команды построения простых графиков функций, комбинированные и презентационные графики, элементы анимации и средства проектирования графического пользовательского интерфейса.

Построение графиков функций.

Элементарные графические функции системы Matlab позволяют построить и вывести следующие типы графиков: линейный, логарифмический, полулогарифмический, полярный.

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

Двумерные графики:

- `plot` – график в линейном масштабе;
- `loglog` – график в логарифмическом масштабе;
- `semilogx`, `semilogy` – график в полулогарифмическом масштабе;
- `polar` – график в полярных координатах.

Для построения графика функции $y(x)$ в простейшем случае используют команду `plot`.

Пример 8 – Построение графика с использованием функции `plot`

```
>> % задание вектора x
>> x = [0:0.005:5];
>> % расчет значений функции
>> y = exp(-x).*sin(10*x);
>> plot (y)
```

После выполнения скрипта из примера 8 на экране возникнет графическое окно (рисунок 9).

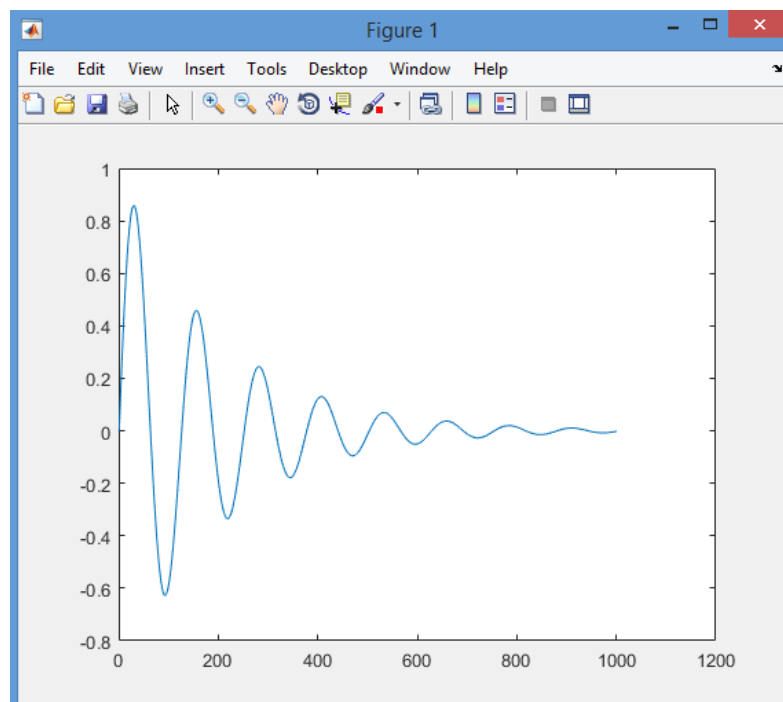


Рисунок 9 – Графическое окно

Выполняя команды меню этого графического окна File→Save as, File→Export Setup, можно сохранить построенный график в файл Matlab с расширением fig, либо экспортировать в графические форматы – png, eps, gif.

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. График функции из примера 8 представлен на рисунке 10.

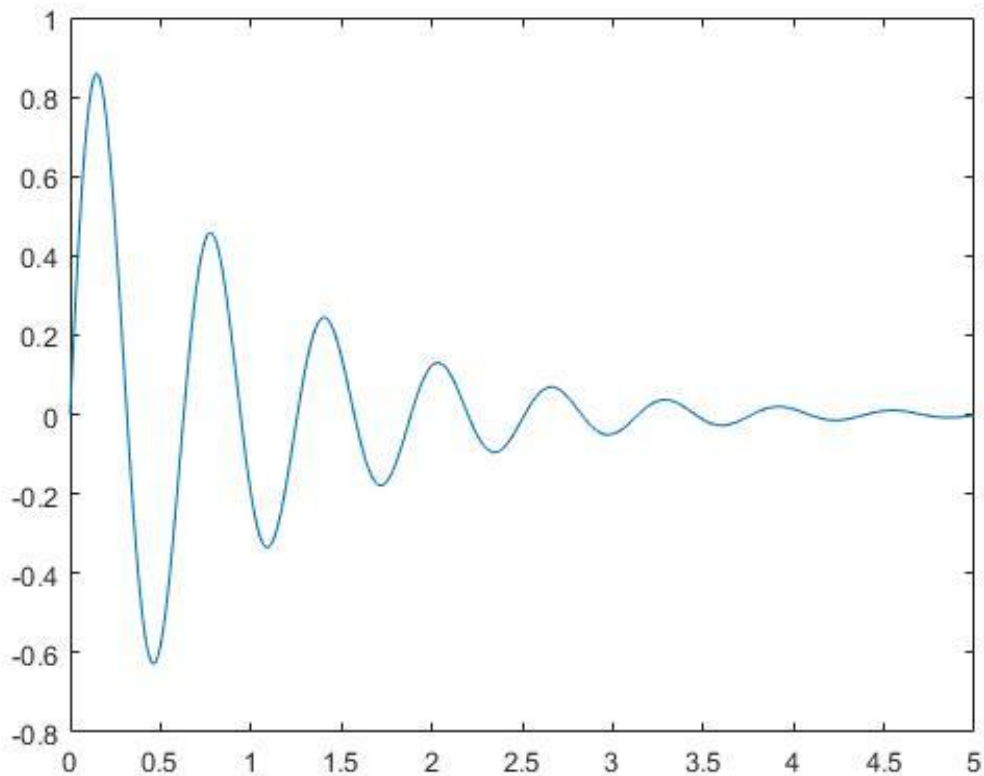


Рисунок 10 – График функции

Несколько графиков в одном графическом окне.

Для того, чтобы построить еще один график в этом же графическом окне, то можно действовать двумя путями – воспользоваться командой `hold on`, либо добавить еще два аргумента в команду `plot`.

Пример 9 – Два графика функции в одних осях с помощью `hold on` и `plot`

```
>> x = [0:0.5:5];
>> y1 = exp(-x).*sin(10*x);
>> y2 = exp(-x).*cos(10*x);
>> % построение первого графика функции
>> plot(x, y1)
>> % продолжать построение в этом же окне
>> hold on
>> % построение второго графика функции
>> plot(x, y2)
>> % построение сразу двух графиков функций
>> plot(x, y1, x, y2)
```

В обоих случаях результат будет одинаковым (рисунок 11).

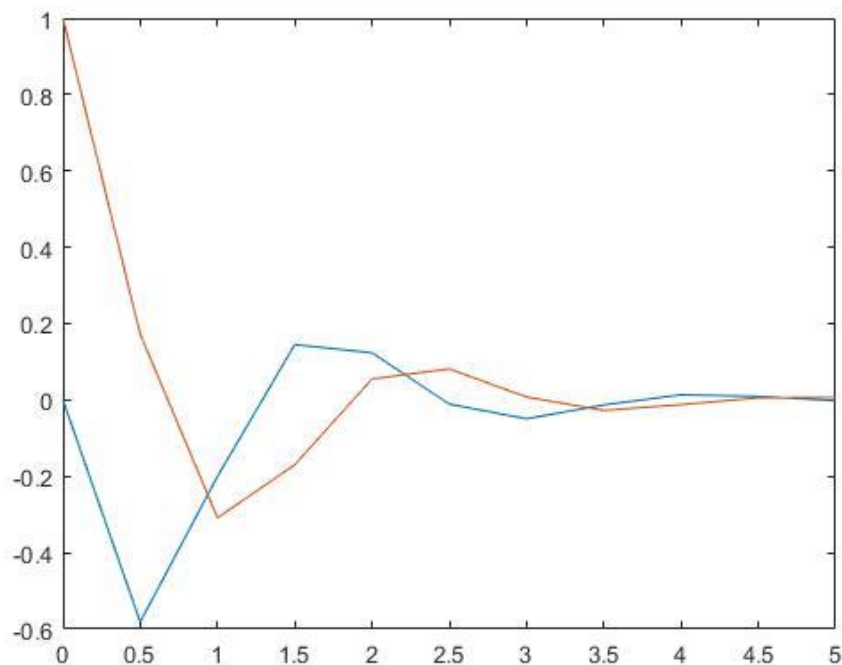


Рисунок 11 – Два графика в одних осях

С помощью функции `delete` производят удаление графика.

Установка параметров графиков.

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`.

Некоторые обозначения для типа линий, цветов и маркеров приведены в таблице 3. Больше информации приведено в библиотеке команды `plot`.

Таблица 3 – Обозначение типа линий, цветов и маркеров

Цвет	
Y	Желтый
M	Розовый
C	Голубой
R	Красный
Линия	
-	Сплошная
:	Пунктирная
-.	Штрих-пунктирная
--	Штриховая
Маркер	
.	Точка
x	Крестик
s	Квадрат
d	Ромб
*	Звездочка

Пример 10 – Задание цвета и типа линии для графика

```
>> x = [0:0.5:5];
>> y1 = exp(-x).*sin(10*x);
>> y2 = exp(-x).*cos(10*x);
>> plot(x, y1, 'm--*')
>> plot(x, y2, 'r-.d')
```

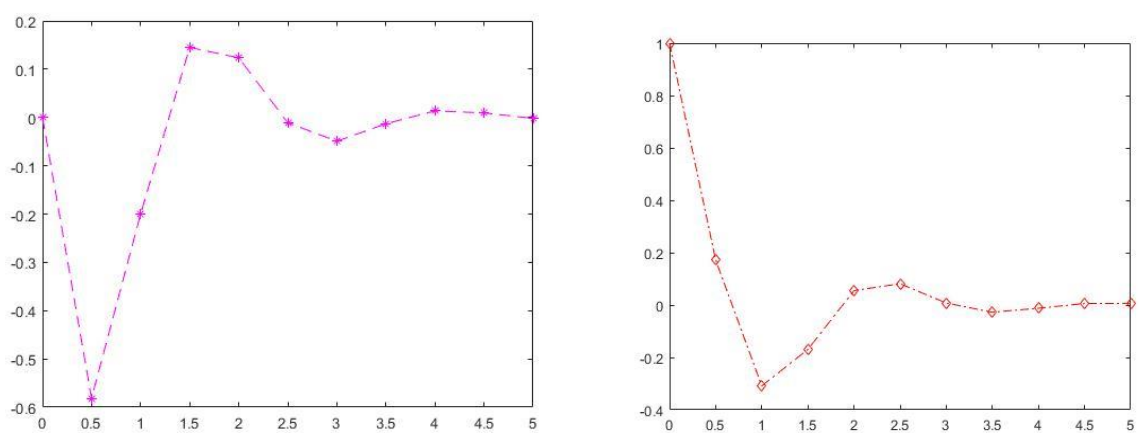


Рисунок 12 – Результат построения графиков: задании типа цвета, линий и маркеров

Поименование осей осуществляется посредством функций `xlabel` и `ylabel`. Важно то, что дополнительные параметры для графических осей устанавливаются, когда оси уже созданы, то есть после команды `plot`.

Команда `grid on` отображает главные линии сетки для текущей системы координат.

Заголовок для графического окна задается с помощью функции `title`, ее параметром является текстовая строка. Если заголовок используется не так часто, то использование легенды – `legend` всегда целесообразно, когда в одних осях отображается более одного графика. Для каждого из графиков в качестве параметров функции `legend` используют текстовую строку, описывающую соответствующий график.

Пример 11 – Использование легенды и подписи осей

```
>> x = [0:0.5:5];
>> y1 = exp(-x).*sin(10*x);
>> y2 = exp(-x).*cos(10*x);
>> % построение графиков двух функций
>> plot(x, y1, 'k-', x, y2, 'k:')
>> % создание легенды
>> legend('y1=exp(-x)*sin(10*x)', 'y2=exp(-x)*cos(10*x)')
>> % создание подписей к осям
>> xlabel('x')
>> ylabel('y')
>> % отображение линий сетки
>> grid on
```

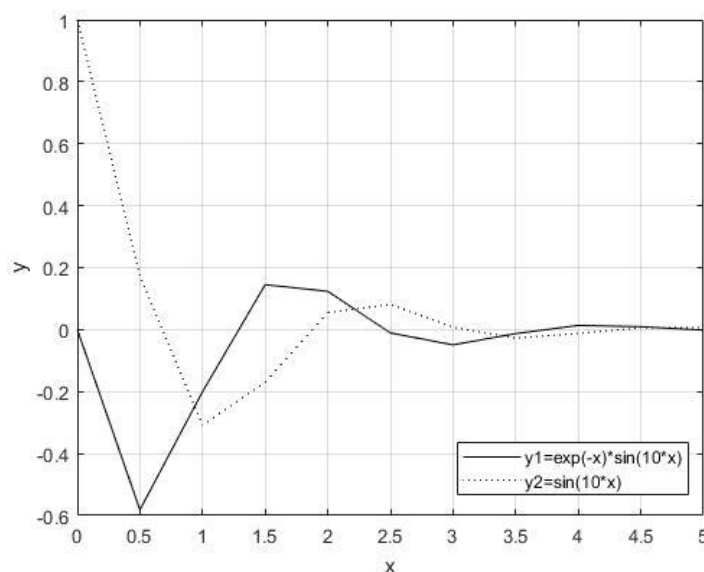


Рисунок 13 – Результат построения графика

Положение легенды можно менять уже после построения графика с помощью перетаскивания.

Создание нескольких графических окон.

Для визуализации графиков функций в одном окне `figure` размещается несколько графических окон. Структура таких окон и выбор активного окна осуществляются процедурой `subplot(m, n, p)`, где **m**, **n**, **p** – 3 цифры, производит разбивку графического окна на несколько подокон, создавая при этом новые объекты; значение **m** указывает, на сколько частей разбивается окно по горизонтали, **n** – по вертикали, а **p** – номер подокна, куда будет выводиться очередной график. Эти же команды могут использоваться для перехода от одного подокна к другому.

Пример 12 – Использование легенды и подписи осей

```
>> % Задание вектора-параметра t
>> t=0:0.001:8*pi;
>> % Данные и вектора для 1-го окна subplot
>> k=5;
>> x11=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
>> y11=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
>> % Данные и вектора для 2-го окна subplot
>> k=5.5;
>> x21=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
>> y21=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
>> % Данные и вектора для 3-го окна subplot
>> k=6;
>> x31=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
>> y31=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
>> % Данные и вектора для 4-го окна subplot
>> k=3.6;
>> x41=(k-1)*(cos(t)+cos((k-1)*t)/(k-1));
>> y41=(k-1)*(sin(t)-sin((k-1)*t)/(k-1));
>> % Построение графиков в 1-м окне subplot
>> subplot(2,2,1)
>> plot(x11,y11)
>> legend('k=5')
>> % Построение графиков во 2-м окне subplot
>> subplot(2,2,2)
>> plot(x21,y21)
>> legend('k=5.5')
>> % Построение графиков в 3-м окне subplot
>> subplot(2,2,3)
```

```
>> plot(x31,y31)
>> legend('k=6')
>> % Построение графиков в 4-м окне subplot
>> subplot(2,2,4)
>> plot(x41,y41)
>> legend('k=3.6')
```

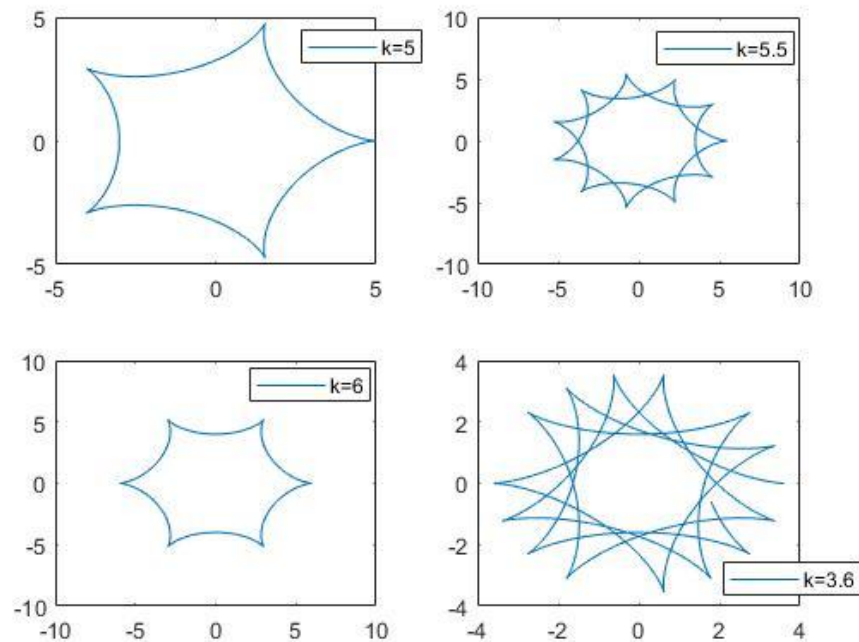


Рисунок 14 – Результат использования процедуры subplot

Для того, чтобы вернуть графическое окно в обычное состояние необходимо использовать процедуру `subplot(1, 1, 1)`, которая удаляет все подокна.

2.7 Создание функций в Matlab

В Matlab для эффективного программирования используются процедуры и функции. Каждая функция записывается в отдельном файле с расширением `*.m` (m-файл) и ее имя должно совпадать с именем этого файла.

Для создания m-файла необходимо из меню File командного окна выбрать New Script или же New → Script. После чего появляется окно редактора, как показано на рисунке 15.

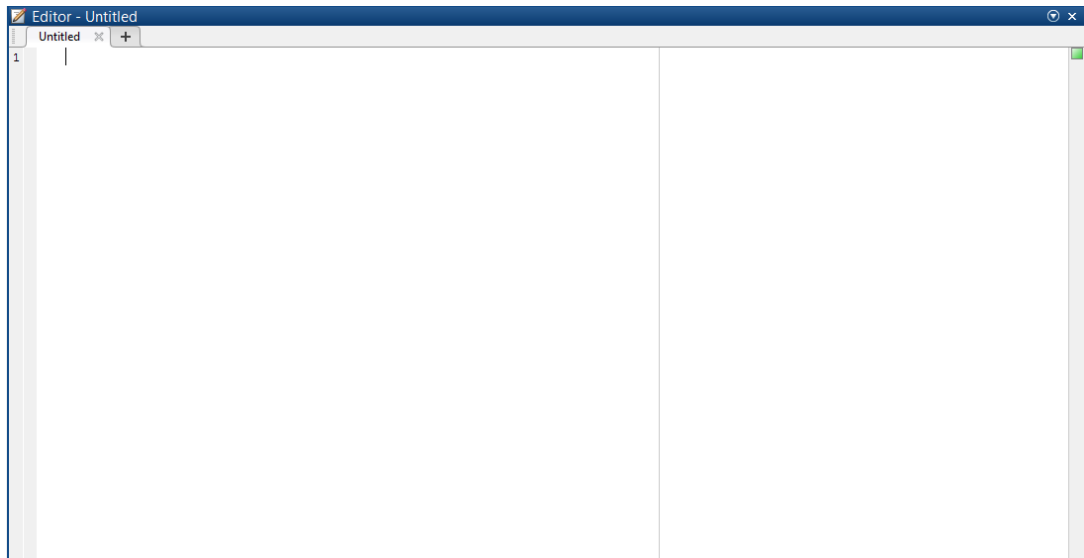


Рисунок 15 – Окно редактора

Для создания функций и процедур используется одинаковый заголовок, но в процедуре может быть один или несколько выходных параметров:

```
function [out1,out2] = myproc(in1,in2,in3),
```

а в функции только один, который вычисляется в последнем исполняемом операторе процедуры:

```
function resfunc = myfun(in1,in2,in3)
resfunc=sin(in1)*in2^in3 % некоторое выражение
```

Во избежание вывода на экран нежелательных промежуточных результатов, необходимо в теле функции все операторы завершать символом «;».

Пример 13 – Создание и вызов процедуры

```
function [x1,x2] = myfun(a,b,c)
%описание процедуры проводится в новом окне редактора
x1 = sin(a);
x2 = (c - b) / (2*a);
>> %вызов процедуры производится в командном окне
>> [r1,r2]=myfun(1,2,3)
r1 =
    0.8415

r2 =
    0.5000
```


Важно! Перед тем как вызывать функцию, необходимо сохранить m-файл (один из способов – используя комбинацию клавиш Ctrl+S). *Имя функции и имя файла должны совпадать!*

Пример 14 – Построение графика функции

```
function y = fun(x)
% Вычисление двух функций
% y(1) = 200 sin(x)/x, y(2) = x^2.
y(:,1) = 200*sin(x)./ x;
y(:,2) = x.^2;
>> %ВЫЗОВ функции
>> fun(0.5)

ans =

    191.7702    0.2500

>> fplot(@fun, [-20 20])
```

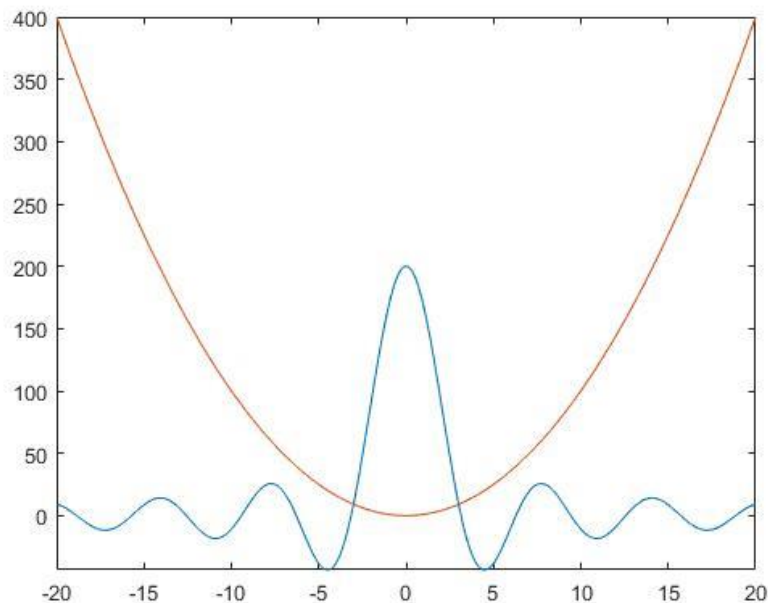


Рисунок 16 – Результат построения

Использованная в примере функция `fplot`, которая позволяет строить функцию, заданную в символьном виде, в интервале изменения аргумента x от **xmin** до **xmax** без фиксированного шага изменения x . Одним из входных параметров `fplot` является функция, перед которой необходимо указывать указатель «@».

В написании функций могут использоваться операторы управления вычислительным процессом: операторы условного перехода и операторы организации циклических процессов.

Все операторы цикла и условного перехода построены в Matlab в виде составного оператора, который начинается одним из служебных слов **if**, **while**, **switch** или **for** и заканчивается служебным словом **end**.

3 Порядок выполнения работы

1. Проработать необходимый теоретический материал, опираясь на сведения и указания, представленные в предыдущем пункте. Все разработанные примеры разобрать на практике, проводя аналогичные операции в среде Matlab (для графиков изменить оформление).

2. Изучить документацию по следующим библиотекам Matlab: **ops**, **relop**, **slash**, **elfun**, **specfun**, **plot**. В отчете привести краткое описание каждой из них.

3. Выполнить задание согласно варианту.

3.1 Вычислить значения функции $f(x)$ на отрезке $[a; b]$ с шагом h .

Таблица 4 – Варианты заданий

Вариант	$f(x)$	a	b	h
1	$\frac{x^2}{1 + 0,25\sqrt{x}}$	0.7	7.5	0.45
2	$\frac{x^3 - 0,3x}{\sqrt{1 + 2x}}$	2.06	8.1	1.2
3	$\frac{2e^{-x}}{2\pi + x^3}$	0	1.5	0.15
4	$\frac{\cos \pi x^2}{\sqrt{1 - 3x}}$	-1	0	0.1
5	$\frac{e^{\frac{x}{2}}}{1 + x^2}$	1.4	5	0.5
6	$e^{-2x} + x^4 + 1$	3.2	6.6	0.7

Продолжение таблицы 4.

Вариант	$f(x)$	a	b	h
7	$(e + x)\sin(\pi\sqrt{x - 1})$	0.25	2.25	0.2
8	$x^3 - 3x + \frac{x^3 - 0,3x}{\sqrt{1 + 2x}}$	0	4.5	0.5
9	$\sin(x) + \frac{2e^{-2x}}{3\pi + x^3}$	0	1.6	0.25
10	$\frac{\sin\pi x^4}{\sqrt{1 - 14x}}$	-1	0	0.1

Важно! Если в процессе вычислений требуется поэлементно умножить, разделить или возвести в степень элементы вектора или матрицы, то для этого используются операторы:

- `.*` – поэлементное умножение;
- `./` и `.\` – поэлементные деления;
- `.^` – поэлементное возведение в степень.

3.2 Построить график функции: настроить его оформление, добавить подпись осей и название графика.

4. Оформить отчет.

4 Содержание отчета

Цель работы; разобранные примеры из пункта 2; краткое описание изученных библиотек Matlab; решенное задание по варианту; выводы по работе.

5 Контрольные вопросы

- 1) Что такое Matlab?
- 2) Каким образом можно получить справку о той или иной команде?
- 3) Как вводятся матрицы и векторы в языке Matlab? Какими функциями можно формировать специальные матрицы и векторы в языке Matlab?
- 4) Какие функции Matlab осуществляют вывод графиков на экран?

5) Какими функциями обеспечивается снабжение графика координатными линиями и надписями?

6) Можно ли построить несколько графиков в одной системе координат и в одном графическом окне?

7) Как построить несколько отдельных графиков в одном графическом окне в разных графических полях?