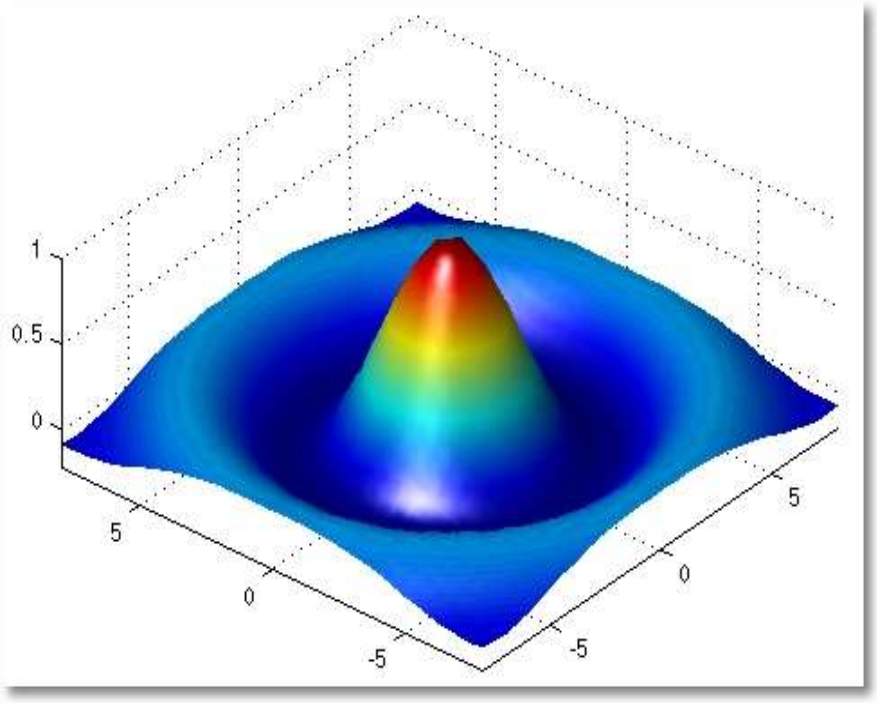
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет математики и компьютерных наук

Бейбалаев В.Д.

MAT LAB
ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Учебно-методическое пособие



Махачкала 2014

Печатается по решению редакционно-издательского совета Дагестанского государственного университета.

Бейбалаев В.Д.
MatLAB. Учебно-методическое пособие. - Махачкала: Издательство ДГУ, 2014.-61 с.

Главной целью пособия является научить студентов пользоваться простейшими методами вычислений с использованием современных информационных технологий. Наиболее подходящей для этой цели является одна из самых развитых и эффективных математических систем - MatLAB, которая занимает особое место среди множества таких систем (MathCAD, Maple, Mathematica и др.). Пособие включает в себя также и лабораторный практикум, апробированный на занятиях со студентами ДГУ, обучающимися по естественно-научным направлениям. Предназначено для бакалавров и магистров естественнонаучных направлений высших учебных заведений и для студентов средних специальных учебных заведений технического профиля.

Рецензент: доцент кафедры ИиДМ ФМиКН Якубов А.З.

ИПЦ ДГУ 2014

1. Введение в среду MATLAB

Одной из основных областей применения современных компьютеров при решении прикладных задач являются математические и научно-технические расчеты. Сложные задачи, возникающие при решении прикладных задач и моделировании различных процессов, можно разбить на ряд элементарных: решение алгебраических и дифференциальных уравнений, вычисление интегралов и т.д. Для решения таких задач на сегодняшний день разработаны методы их решения, созданы различные математические системы, доступные для изучения. Одной из таких систем является Matlab.

Matlab (MATrix LABoratory) – это:

- математические вычисления;
- создание алгоритмов;
- моделирование;
- анализ, обработка и визуализация данных;
- научная и инженерная графика;
- огромное количество прикладных пакетов.

В Matlab встроены следующие основные пакеты:

- Matlab Web Server
- Bioinformatics Toolbox
- Communications Toolbox
- Control System Toolbox
- Database Toolbox
- Distributed Computing Toolbox
- Financial Toolbox
- Fuzzy Logic Toolbox
- Genetic Algorithm and Direct Search Toolbox
- Image Processing Toolbox
- Neural Networks Toolbox
- Partial Differential Equation Toolbox
- Signal Processing Toolbox
- SimBiology
- Spline Toolbox
- Statistics Toolbox
- Symbolic Toolbox
- Virtual Reality Toolbox
- Wavelet Toolbox
- Simulink
- Aerospace Blockset
- Communications Blockset
- Video and Image Processing
- Real-Time Workshop
- Matlab Builder for .NET
- Matlab Compiler
- Интеграция в MS Office

В среде Matlab можно выделить пять основных частей:

1. Язык Matlab.
2. Среда Matlab.
3. Управляемая графика.

4. Библиотека математических функций.
5. Программный интерфейс.

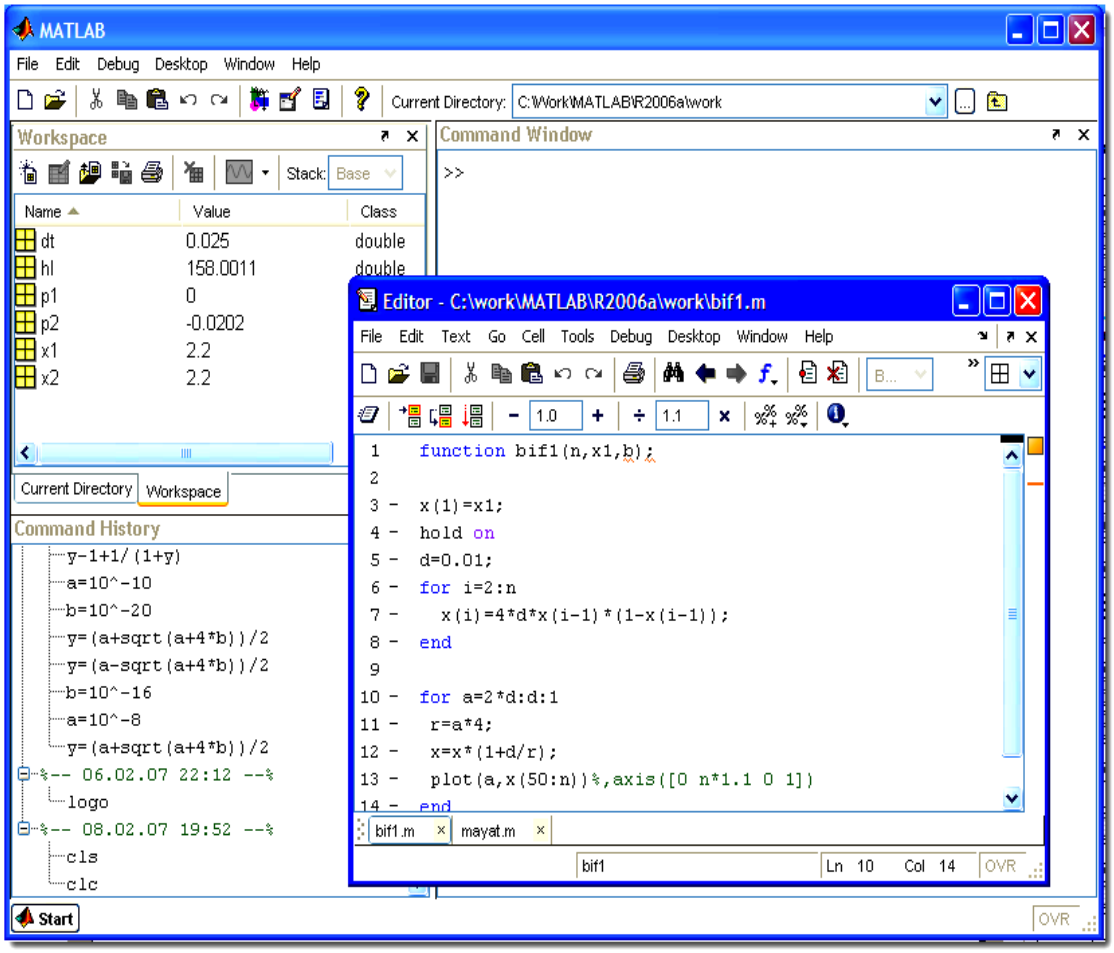
Языком Matlab является:

- Си- и Паскаль-подобный объектно-ориентированный.
- Огромный набор встроенных функций, расширяемый пользователем.

```
clear;
x1=2.2;
p1=0.0;
dt=0.025;
axis([-pi pi -pi pi]);
hl=line(x1,p1);
set(hl,'EraseMode' , 'none' , 'LineStyle' , ':' , 'Color' , 'r' );
grid on;
pause;
while 1
  x2=x1+p1*dt;
  p2=p1-sin(x2)*dt;
  if x2> pi
    x2=x2-2*pi;
  end;
  if x2< -pi
    x2=x2+2*pi;
  end;
  set(hl, 'XData' ,x2, 'YData', p2);
  x1=x2; p1=p2;
end;
```

Среда Matlab это:

- Интерактивная работа.
- Управление переменными в рабочем пространстве.
- Редактор.
- Отладчик.



Управляемая графика Matlab состоит из команд:

- высокого уровня для работы с 2D- и 3D-графикой;
- анимации;
- низкого уровня для работы с графикой.

В среде *Matlab имеется хорошая библиотека математических функций:*

- Обширная коллекция вычислительных алгоритмов от элементарных функций (*sin*, *cos* и т. П.) до более сложных
  - обращение матриц;
  - вычисление собственных значений;
  - минимизация функций;
  - дифференцирование;
  - интегрирование;
  - и пр.

В среде Matlab имеется программный интерфейс API для взаимодействия с программами на языках Си и Фортран.

Matlab – язык для работы с матричными объектами. Основной объект Matlab – матрица. Число – это матрица размера (1x1). Использование матриц существенно облегчает программирование и делает запись формул краткой и наглядной. В дальнейшем изложении предполагается знакомство с матричной алгеброй и основами программирования. Переменные в *Matlab* определяются пользователем при помощи оператора присваивания: x=5. При этом в левой части – имя переменной. В правой части оператора присваивания может стоять выражение: y=(2-x)/(x+3). Если выражение встречается вне оператора присваивания, то его значение вычисляется и помещается в системную переменную ans (от answer).Переменную ans можно использовать для задания новых выражений: z=ans*3. Если оператор присваивания завершить символом «;», то результат на экране не дублируется; в противном случае – выводится на экран:

```
>> a = 2*3
a =
     6
>> b = a/7
b =
     0.8571
```

В *Matlab* при составлении выражений могут быть использованы операторы:

+ сложение
- вычитание
* умножение
/ деление

^ возведение в степень

При этом приоритет операций обычный. Изменяется при помощи круглых скобок.

В среде *Matlab* используют следующие операции отношения:

<	меньше
<=	меньше или равно
>	больше
>=	больше или равно
==	равно
~=	не равно

Приоритет этих операций ниже чем арифметических.

```
>> a = 1; b = 2; c = 3;
>> res = (a < b) + (c ~= b) + (b == a)
res =
     2
```

А также в среде *Matlab* используют следующие логические операции:

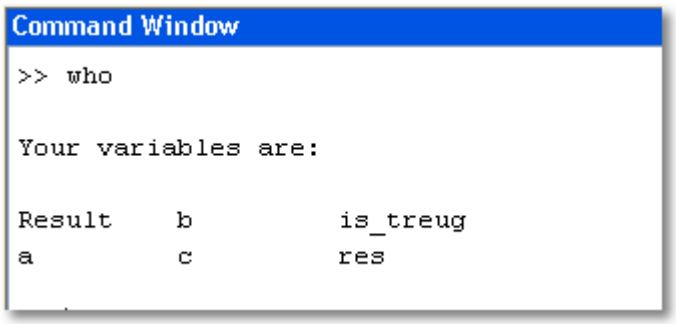
&	и
|	или
~	не
0 – ложь (false) 1 – истина (true)	

Приоритет этих операций ниже чем арифметических и операций отношения.

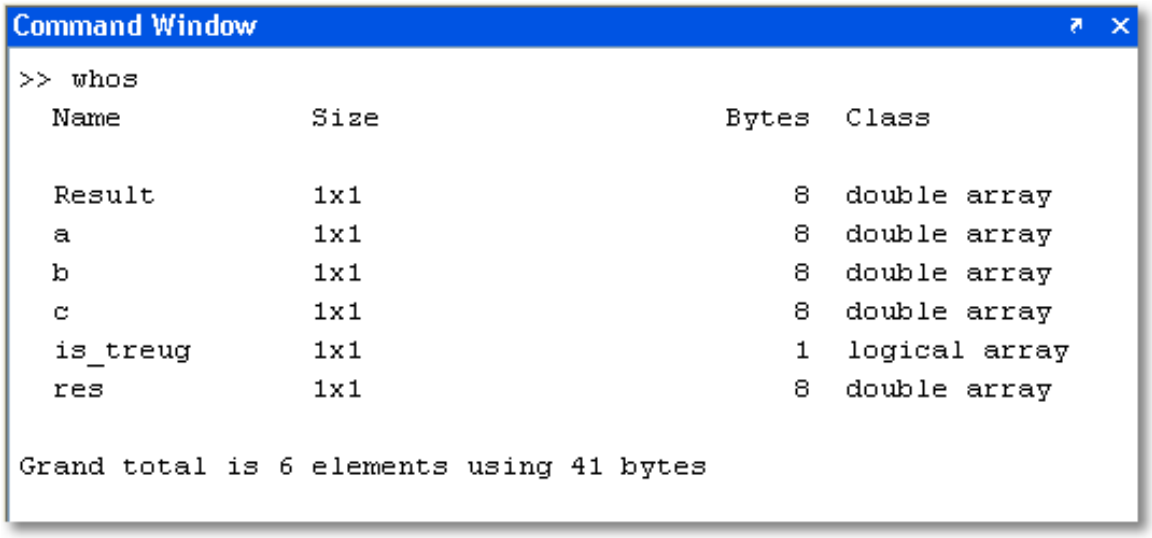
Простейший способ взаимодействия с Matlab – работа в командной строке (в режиме калькулятора). Строка начинается с приглашения: символа >>. Перемещение по стеку ранее введённых команд – клавиши ↑ и ↓. Для удобства размещения данных в КС

можно разбивать вводимое выражение знаком «…». Очистить командное окно можно командой clc.

Все переменные в Matlab хранятся в рабочем пространстве (Workspace). Порой это отнимает много места. Просмотреть список существующих в рабочем пространстве переменных можно командой who.



Более подробную информацию о переменных рабочего пространства можно вывести командой whos.



После закрытия сеанса работы Matlab все переменные, вычисленные в течение сеанса, теряются. Однако их можно сохранить для последующего использования в иных сеансах, сохранив содержимое рабочего пространства в файле на диске:

– командой меню: File \ Save Workspace As…
– командой Matlab: save.

Команды:

- save – сохраняет все переменные в файл *matlab.mat*
- save filename – сохраняет все переменные в файл *filename*

- save filename x y z – сохраняет переменные *x*, *y*, *z* в файл *filename* (можно по маске: a*)
- save filename x y z –ASCII – сохраняет переменные *x*, *y*, *z* в файл *filename* в текстовом виде
- save('filename', 'a','b','-ASCII') – процедурная форма вызова команды
  - параметры – в виде строк (в одинарных апострофах)

Подробнее про эту и любую другую команду Matlab можно узнать с помощью:

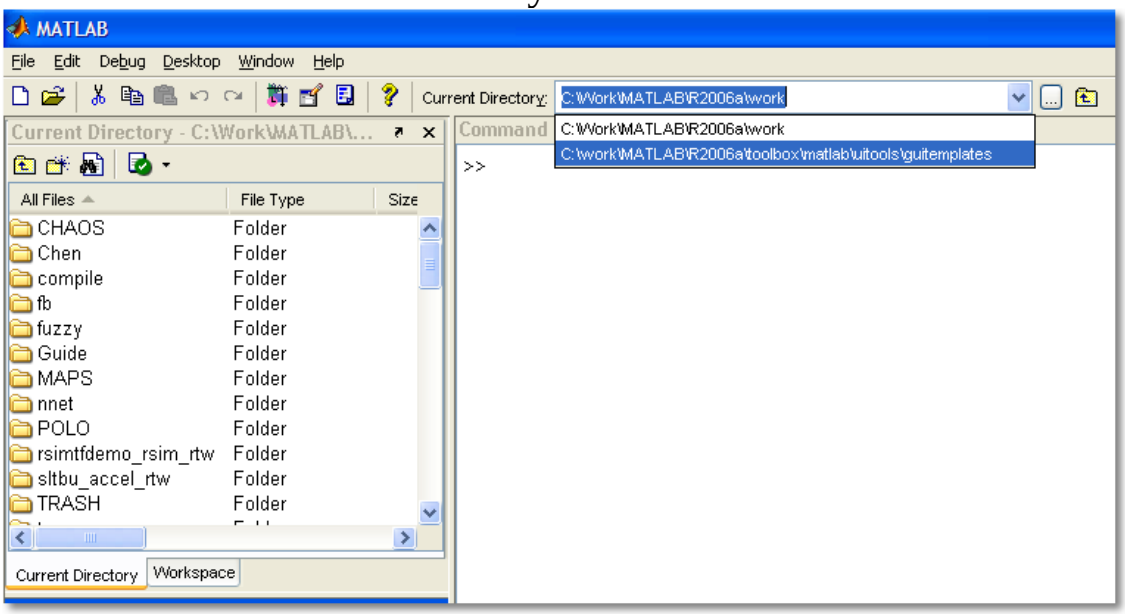
- help <имя команды>;
- или F1.

Команда clear служит для удаления переменных из рабочего пространства:

- clear – удаляет все переменные;
- clear all – удаляет всё, включая классы, функции, скомпилированные файлы и пр.;
- clear x y z – удаляет переменные *x*, *y* и *z*.

Все файлы (данные, функции и пр.), созданные пользователем сохраняются в текущем каталоге (Current Directory). Изменить текущий каталог можно:

- командой cd <путь>
- в строке ввода Current Directory на панели инструментов;
- в окне Current Directory.



Рабочую сессию в Matlab можно сохранить следующим образом:

- diary – сохраняет лог текущей сессии (весь текстовый ввод и вывод) в файл по умолчанию – в файл *diary* в текущем каталоге;

- diary filename или diary(‘filename’) – сохраняют сессию в указанном файле;
- diary off / diary on – соответственно, приостанавливают и продолжают ведение лога;

2. Математические вычисления в среде MATLAB

В среде Matlab используют следующие основные элементарные функции.

1. *Тригонометрические функции:*

- sin
- cos
- tan
- cot
- asin
- acos
- atan
- acot
- sinh
- cosh
- tanh
- coth
- asinh
- acosh
- atanh
- acoth
- sind
- cosd
- tand
- cotd

2. *Экспоненциальные:*

- exp
- log – ln
- log10
- log2
- sqrt
- nthroot(x, n).

3. *Элементарные функции округления*:

- fix – округление к нулю;
- floor – округление к минус бесконечности;
- ceil – округление к плюс бесконечности;
- round – округление к ближайшему целому;
- mod(x,y) – остаток от деления x на y без учёта знака (x - n*y, где n = floor(x/y));
- rem(x,y) – остаток от деления x на y с учётом знака (x - n*y, где n = fix(x/y)).

4. *Элементарные функции работы с комплексными числами:*

- abs(z) – модуль комплексного числа z;
- angle(z) – фаза z (в радианах);
- real(z) – действительная часть z;
- imag(z) – мнимая часть z;
- conj(z) – комплексно сопряжённое число для z;
- complex(a,b) – конструирует комплексное число a+ib;
- isreal(z) – возвращает истину, если z – действительное.

Просмотреть полный список элементарных функций можно командой

- help elfun.

Двумерные массивы в Matlab задаются в следующем виде a = [1 2; 3 4; 5 6].

```
>> a = [1   2;3  4;5  6]
a =
     1     2
     3     4
     5     6
```

Доступ к элементу массива совершается следующим образом:

```
>> a(3,1)
ans =
     5
>> a(1,3)
??? Index exceeds matrix  dimensions .
```

Любая строка и столбец матрицы – это вектор. Векторы, расположенные вдоль строк – векторы-строки (размер 1xn). Векторы, расположенные вдоль столбцов – векторы-столбцы (размер nx1). К векторам любого типа применима функция length.

```
>> c = [1;2;3]
c =
     1
     2
     3
>> c = [1   2   3]
c =
     1
     2
     3
```

Размерность массива определяется функцией ndims(A), а *размер массива* – функцией size(A):

```
>> size(a)
ans =
     2     3
>> [m n] = size(a)
m =
     2
n =
     3
```

```
>> a = [1  2  3;4  5  6]
a =
    1    2    3
    4    5    6
>> ndims(a)
ans =
     2
```

```
>> size(a)
ans =
     2     3
>> [m n] = size(a)
m =
     2
n =
     3
```

Рассмотрим две матрицы:

```
>> a = [1  2  3;4  5  6]
a =
     1    2    3
     4    5    6
>> b = [4  6  7;0  9  5]
b =
     4    6    7
     0    9    5
```

Проведём склейку «в столбик», а затем «в строку»:

```
>> c = [a; b]
c =
     1   2   3
     4   5   6
     4   6   7
     0   9   5
```

Диапазоны можно использовать как для задания значений векторов, так и для задания диапазонов индексации:

```
>> a = magic(5)

a =

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

>> a(3,:)

ans =

     4     6    13    20    22

>> a(:,1)

ans =

    17
    23
     4
    10
    11
```

```
>> a = magic(5)

a =

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

>> a(2:3,4:5)

ans =

    14    16
    20    22
```

Для работы с матрицами удобно пользоваться следующими функциями:

- ones – формирование массива из единиц;
- zeros – формирование массива из нулей;
- eye – формирование единичной матрицы;
- rand – формирование массива из чисел, случайно распределённых на отрезке [0, 1];

- randn – формирование массива из чисел, нормально распределённых на отрезке [0, 1];
- magic – формирование магического квадрата;
- pascal – формирование квадрата Паскаля;
- diag – диагональная матрица;

Рассмотрим основной синтаксис на примере функции создания единичной матрицы (eye). С помощью функции eye(m) можно создать единичную матрицу размера [m, m], а с помощью функции eye(m, n) единичную матрицу размера [m, n]. При этом «лишние» строки или столбцы дополняются нулями.

```
>> a = eye(4)

a =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

>> a = eye(4, 6)

a =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
```

Функция **zeros(m, n)** – создает матрицу размером m на n с нулевыми элементами, а команда **ones (m, n)** - создает матрицу размером m на n с единичными элементами.

```
>> z = zeros(4)

z =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> z = zeros(3, 4)

z =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
>> z = ones(5)

z =

     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1

>> z = ones(5, 3) * 7

z =

     7     7     7
     7     7     7
     7     7     7
     7     7     7
     7     7     7
```

Функция **rand (m,n)** - создает матрицу размером m на n из случайных чисел, равномерно распределенных в диапазоне от 0 до 1.

```
>> rand(4)

ans =

    0.9355    0.0579    0.1389    0.2722
    0.9169    0.3529    0.2028    0.1988
    0.4103    0.8132    0.1987    0.0153
    0.8936    0.0099    0.6038    0.7468

>> randn(4)

ans =

   -0.4326   -1.1465    0.3273   -0.5883
   -1.6656    1.1909    0.1746    2.1832
    0.1253    1.1892   -0.1867   -0.1364
    0.2877   -0.0376    0.7258    0.1139
```

```
>> magic(3)

ans =

     8     1     6
     3     5     7
     4     9     2

>> pascal(6)

ans =

     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

Функция diag используют для работы с диагональными матрицами, у которых ненулевые элементы расположены на диагоналях. Синтаксис:

- X = diag(v) – на главной диагонали матрицы X расположены элементы вектора v;
- X = diag(v,k) – на k-ой диагонали матрицы X расположены элементы вектора v (по умолчанию k=0);
- v = diag(X,k) – извлечь из матрицы X k-ую диагональ и сохранить её в векторе v.

```
>> v = 1:4

v =

     1     2     3     4

>> A = diag(v)

A =

     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4

>> B = diag(v, 2)

B =
   0     1     2
     0     0     1     0     0     0
     0     0     0     2     0     0
     0     0     0     0     3     0
     0     0     0     0     0     4
     0     0     0     0     0     0
     0     0     0     0     0     0
```

Над элементами массивов можно совершить следующие простейшие операции:

- sum- находит сумму элементов;
- prod- находит произведение элементов;
- cumsum- находит кумулятивную сумму элементов;
- cumprod: кумулятивное произведение элементов;
- max- находит максимальный элемент;
- min- находит минимальный элемент;
- sort- сортируют элементы.

Рассмотрим работу некоторых из этих функций на примерах. Для векторов функция **sum** возвращает сумму элементов. Для массивов – сумму элементов по каждому из столбцов. Результатом является вектор-строка. Остальные функции работают по этому же принципу.

```
>> A = roud(10 * rand(4,5) - 3)
A =
     3    1    2    5    5
     1   -1    3    4   -2
     2    3   -1    2    3
     0    5    1    3   -2
>> v = sum(A)
v =
     6    8    5   14    4
>> sum(v)
ans
     37
>> sum(sum(A) )
ans
     37
```

Кумулятивная сумма вычисляется так же, только происходит накопление вычисленных значений в элементах массива.

```
>> A = magic(5)

A =

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

>> cumsum(A)

ans =

    17    24     1     8    15
    40    29     8    22    31
    44    35    21    42    53
    54    47    40    63    56
    65    65    65    65    65
```

Максимальный и минимальный элементы определяют следующим образом:

```
>> A = round(100*rand(6,5)-40)

A =

    41   -23    47   -15    21
    21    43    37    -5   -33
    30    44     4   -21    -9
   -31     5    22     9    21
     2    56    55     1   -22
    -2   -25    24     6    22

>> max(A)

ans =

    41    56    55     9    22

>> [max(min(A)), min(max(A))]

ans =

     4     9
```

Вызов функций **max/min** с двумя выходными параметрами позволяет определить и индекс найденного элемента:

```
A =

   -15    -6    10   -39    38
    19     0    32    26    59
    11    -9    -9    32     7
     6     1   -29   -12    50
    14   -11     4   -14     5
    54    -1     7    31    40

>> [a,i]=max(A)

a =

    54     1    32    32    59


i =

     6     4     2     3     2
```

Функция **sort** производит сортировку элементов матрицы по столбцам:

```
>> A

A =

   -15    -6    10   -39    38
    19     0    32    26    59
    11    -9    -9    32     7
     6     1   -29   -12    50
    14   -11     4   -14     5
    54    -1     7    31    40

>> sort(A)

ans =

   -15   -11   -29   -39     5
     6    -9    -9   -14     7
    11    -6     4   -12    38
    14    -1     7    26    40
    19     0    10    31    50
    54     1    32    32    59
```

Функция **All(v)** – возвращает истину, если все элементы вектора v отличны от нуля. Для матриц выдаёт вектор-строку с аналогичным результатом для каждого столбца:

```
>> M = magic(4)-7

M =

     9    -5    -4     6
    -2     4     3     1
     2     0    -1     5
    -3     7     8    -6

>> all(M)

ans =

     1     0     1     1
```

Функция **Any(v)** – возвращает истину, если хотя бы один элемент вектора v отличен от нуля. Для матриц выдаёт вектор-строку с аналогичным результатом для каждого столбца:

```
>> M=zeros(5); M(1,1)=3; M(2,3)=5

M =

     3     0     0     0     0
     0     0     5     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> any(M)

ans =

     1     0     1     0     0
```

Функция **find** определяет индексы элементов, удовлетворяющих заданному условию:

```
>> v

v =

     5    15     2     6    -1     5     3     8

>> find(v>7)

ans =

     2     8

>> v(find(v>7))

ans =

    15     8
```

Функция **det** вычисляет определитель квадратной матрицы.

При работе с матрицами можно использовать два вида операторов:

- *матричные*- производят действия по правилам матричной алгебры;
- *поэлементные*- производят действия над соответствующими элементами матриц. При этом размеры матриц должны быть одинаковыми. От матричных операций отличаются точкой перед знаком операции.

Матричные и поэлементные операции:

- ‘ транспонирование;
- + матричное (и поэлементное) сложение;
- - матричное (и поэлементное) вычитание;
- * матричное умножение;
- / матричное деление;
- ^ матричное возведение в степень;
- \ матричное деление «слева»;
- .* поэлементное умножение;
- ./ поэлементное деление;
- .^ поэлементное возведение в степень;
- .\ поэлементное деление «слева.

Операции «*деления» слева и справа* применяются для решения систем линейных уравнений (СЛУ). Деление слева (\) для квадратных матриц реализует метод Гаусса, а для прямоугольных матриц– метод наименьших квадратов.

Пример. Найти сумму и разность матриц:

```
>>A=[1 2 3 4 5; 6 7 8 9 11]
A= 1   2   3   4   5
    6   7   8   9  11
>>B=[0 -1  -2  -3  -4;  5  6  7  8  9]
B =   0  -1   -2   -3   -4
      5   6    7    8    9
>>A+B
ans =  1    1    1    1    1
      11 13  15  17  20
>>A-B
      ans = 1  3  5  7  9
            1  1  1  1  2
```

Пример. Найти произведение матрицы на число

```
>>A*5
ans =  5   10  15  20  25
       30  35 40  45  55
```

Пример. Транспонирование матрицы

```
>>A’
ans = 1  6
      1  7
      2  8
      3  9
      4  11
```

Пример. Найти произведение двух матриц

```
>>A’*B
ans = 30  35  40  45  50
      45  40  45  50  55
      40 45  50  55  60
      45 50  55  60  65
      50 61  67  73  79
```

С помощью функции **inv (A) находят** матрицу обратную заданной матрице А. При этом исходная матрица А должна быть квадратной и её определитель должен быть отличен от нуля.

Весьма интересными в языке Matlab являются операции деления матриц слева направо и справа налево .(/ и \)

Операция **A\B** равносильна совокупности операций **inv(A)*B**, которая является решением матричного уравнения: **A*X=B.**

Для примера рассмотрим решение системы линейных алгебраических уравнений:

$x_1 + 2x_2 + 3x_3 = 14$

$2x_1 - x_2 - 5x_3 = -15$

$x_1 - x_2 - x_3 = -4$

Вводим матрицу коэффициентов A и вектор строку B:

```
>>A=[1 2 3; 2 -1 -5; 1 -1 -1]
A=   1   2   3
     2  -1  -5
     1  -1  -1
>>B= [14; -15; -4]
B  =  14
     -15
     - 4
>>x =  A \ B
x =
     1
     2
     3
```

То есть $x_1 = 1$ $x_2 = 2$ $x_3 = 3$ – корни системы уравнений.

В системе Matlab предусмотрены возможности **математического оперирования с полиномами.**

Полином (многочлен) как функция определяется следующим выражением:

$$P(x) = a_n * x^n + ... + a_2 * x^2 + a_1 * x + a_0$$

В Matlab полином задается и хранится в виде вектора, элементами которого являются коэффициенты полинома от a_n до a_0

$P = [a_n \ldots a_2\ a_1\ a_0]$

Ввод полиномов осуществляется также как и ввод вектора длиной n+1, где n – порядок полинома.

Система Matlab имеет функцию **roots(P)** которая вычисляет вектор, элементы которого являются корнями заданного полинома, по вектору коэффициентов. Пусть требуется найти корни полинома:

$P(x) = x^5+8x^4+31x^3+80x^2+94x+20$

```
>>P=[1   8   31   80   94   20]
```

>>roots(P)

ans=

```
    -1.0000+3.0000i
    -1.0000+3.0000i
    -3.7321
    -2.0000
    -0.2679
```

Обратная операция – **построение вектора P коэффициентов полинома по заданному вектору его корней** – осуществляется функцией **poly.**

P = poly(R),

где R – заданный вектор корней полинома, P – вычисленный вектор коэффициентов полинома.

Пример:

```
>>P = [1   8   31   80   94   20]
ans =
     1   8  31  80  94  20
>>R= roots (P)
ans  =
     -1.0000+3.0000i
     -1.0000+3.0000i
     -3.7321
     -2.0000
     -0.2679
>>P1 = poly(R)
ans=
     1.0000  8.0000   31.0000   80.0000   94.0000   20.0000
```

Для вычисления значения полинома по заданному значению его аргумента в Matlab предусмотрена функция **polyval.** Обращение к ней происходит по схеме:

y = polyval (P, x),

где P – вектор коэффициентов полинома, x – значение аргумента полинома.

Пример.

```
>>P = [1   8   31   80   94   20]
>>x = 2
>>y = polyval (P, x)
```

ans=

936

Вычисление производной от полинома производится функцией **polyder.** Эта функция создает вектор коэффициентов полинома представляющий собой производную от заданного полинома:

>>**dp = polyder (P)**

dp = 5 32 93 160 94

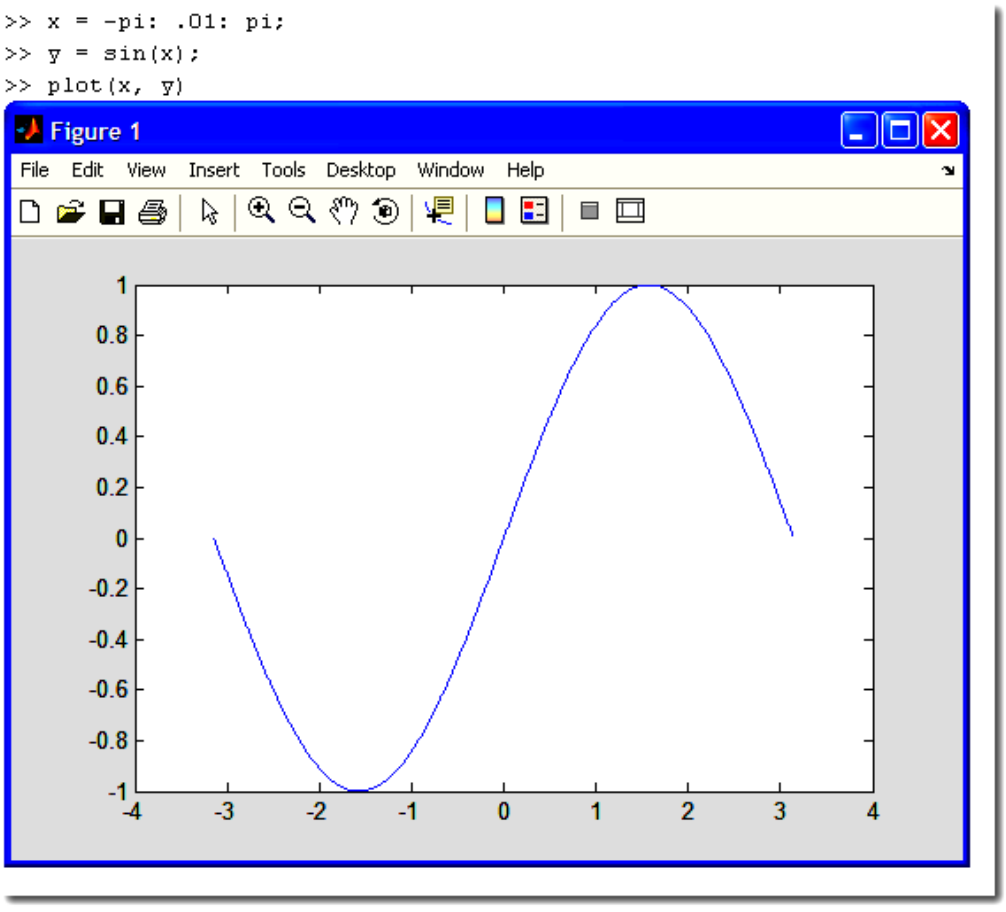
3. Графика в Matlab

В среде *Matlab* имеется высокоуровневая, объектная и управляемая графика. Графика *Matlab* не требует от пользователя детальных знаний о работе графической подсистемы, каждый объект на рисунке имеет свойства, которые можно менять и доступ к графическим объектам возможен как через инспектор объектов, так и при помощи встроенных функций.

Для построения 2D-графика необходимо задать область построения (диапазон), вычислить значение функции на области построения и построить график при помощи одной из встроенных функций Matlab.

Например:

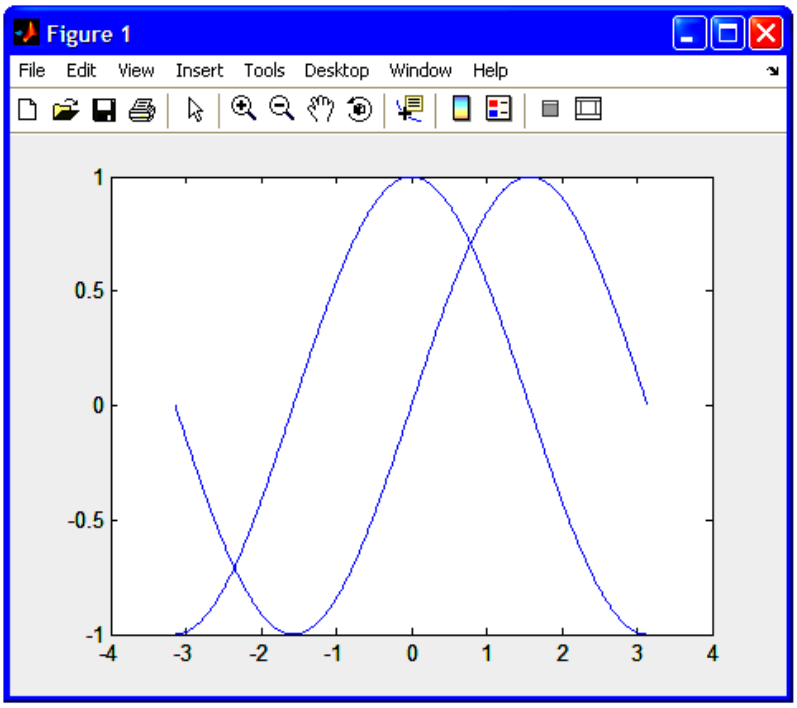
```
>> x = -pi: .01: pi;
>> y = sin(x);
>> plot(x, y)
```



Если сразу же построить другой график, то старый график будет удалён из графического окна. Для построения двух графиков в одной СК необходимо «закрепить» графическое окно при помощи команды *hold on* и применить одну команду *plot.*

Например:

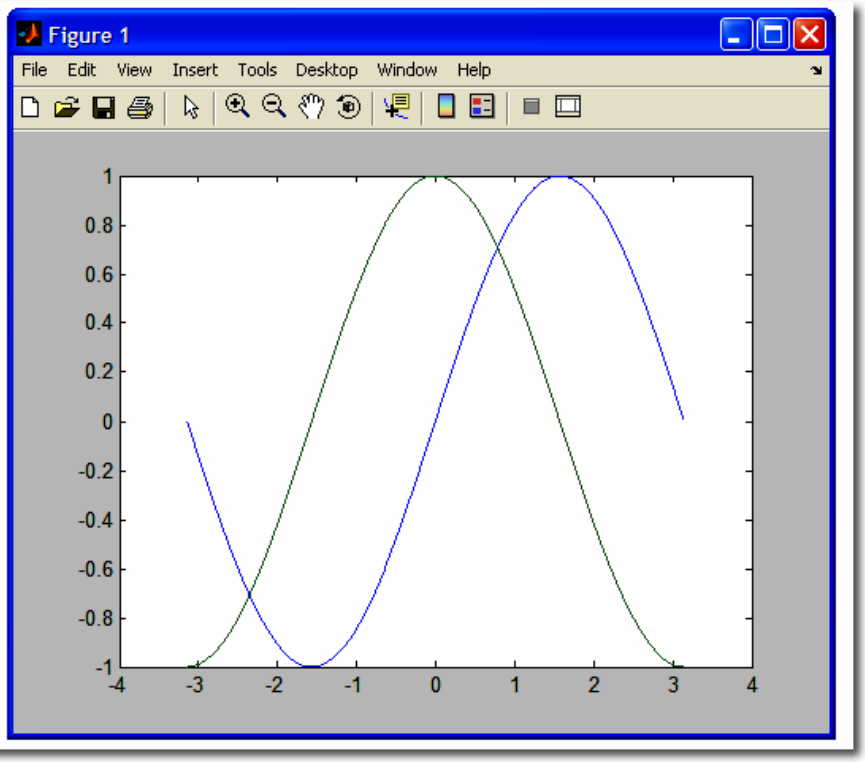
```
>> x = -pi: .01: pi;
>> y = sin(x);
>> plot(x, y)
>> z = cos(x);
>> hold on
>> plot(x, z)
>> z = cos(x);
```



А также два графика в одной СК можно построить с помощью дополнительных параметров команды *plot.*

Например:

```
>> x = -pi: .01: pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y, x, z)
>>
```

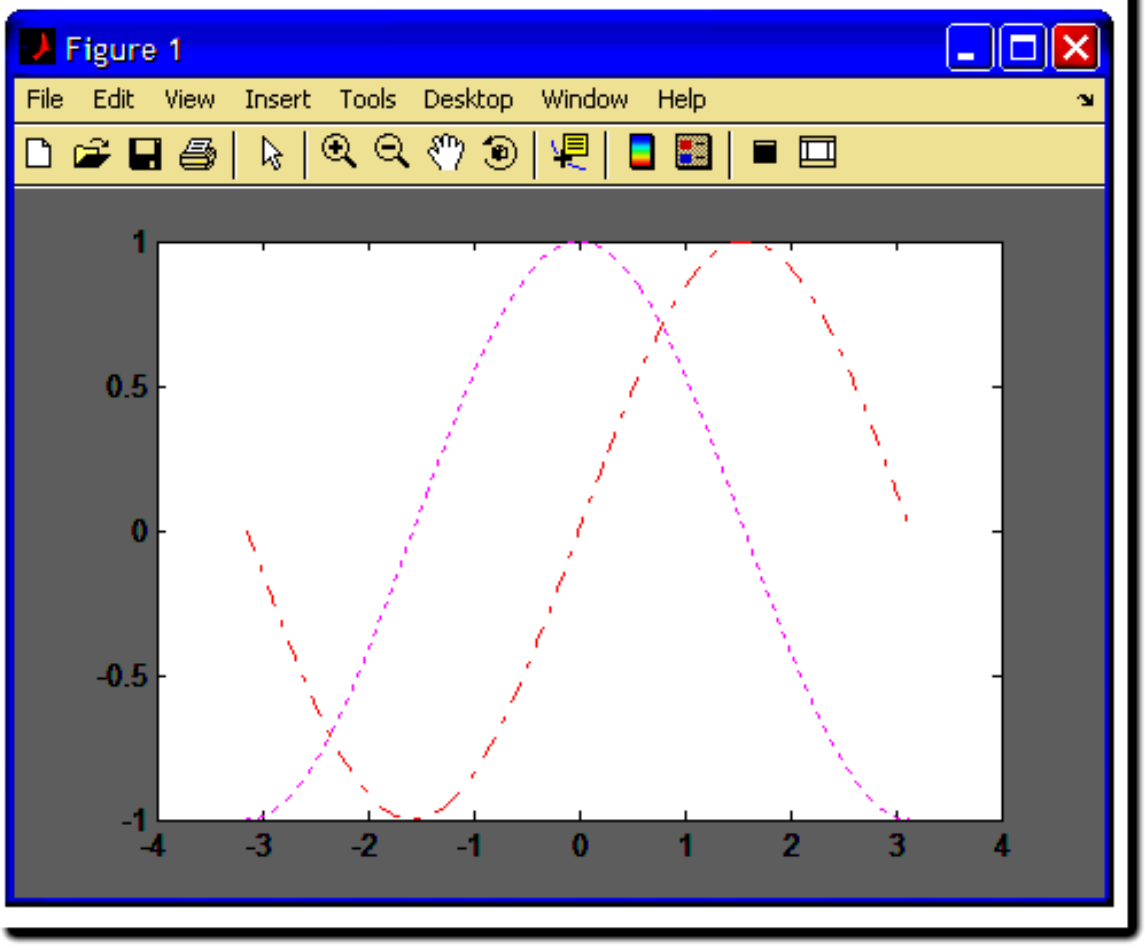


В команде *plot* можно задать для каждого графика цвет линии, тип маркера и тип линии

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		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Например:

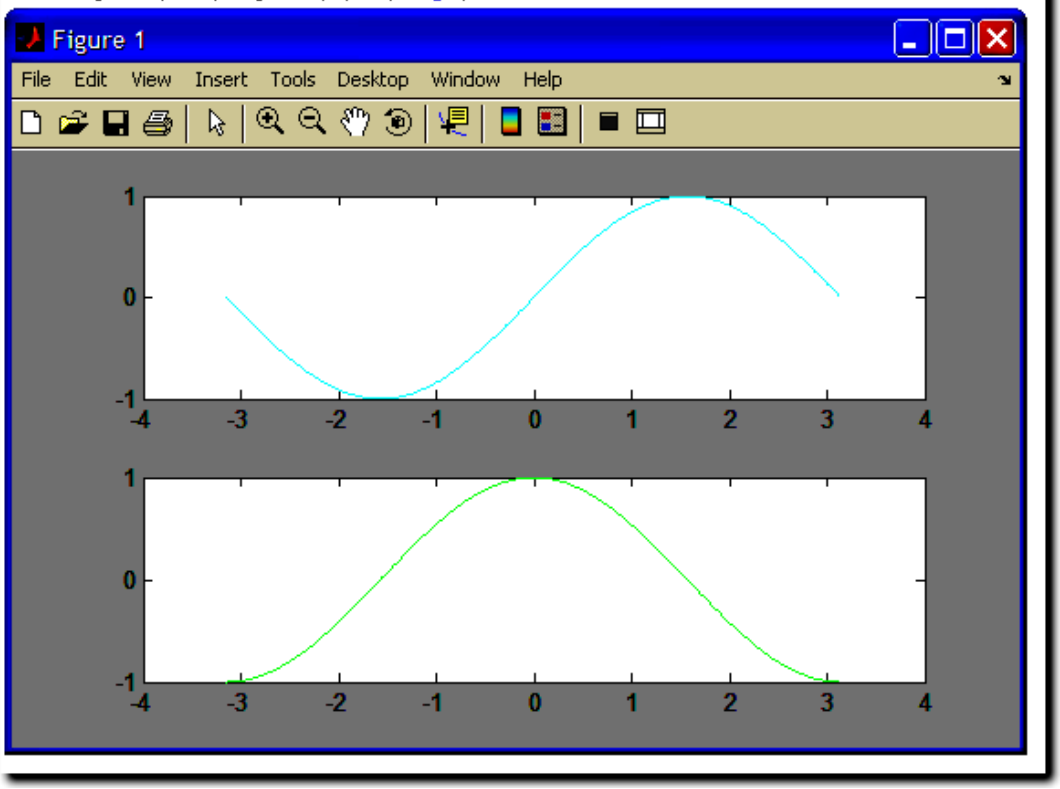
```
>> x = -pi: .01: pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y, 'r-.', x, z, 'm:')
```



В среде Matlab имеется возможность построения нескольких графиков в одном окне но в разных системах координат. Для этого поверхность графического окна с помощью команды *subplot* можно разделить на зоны, в каждой из которых выводится график. В качестве параметров ей передаётся трёхзначное целое число вида *mnk* (m и n определяют количество графических «подокон» по горизонтали и вертикали, а k задаёт номер графического «подокна»). При этом порядок нумерации - по строкам.

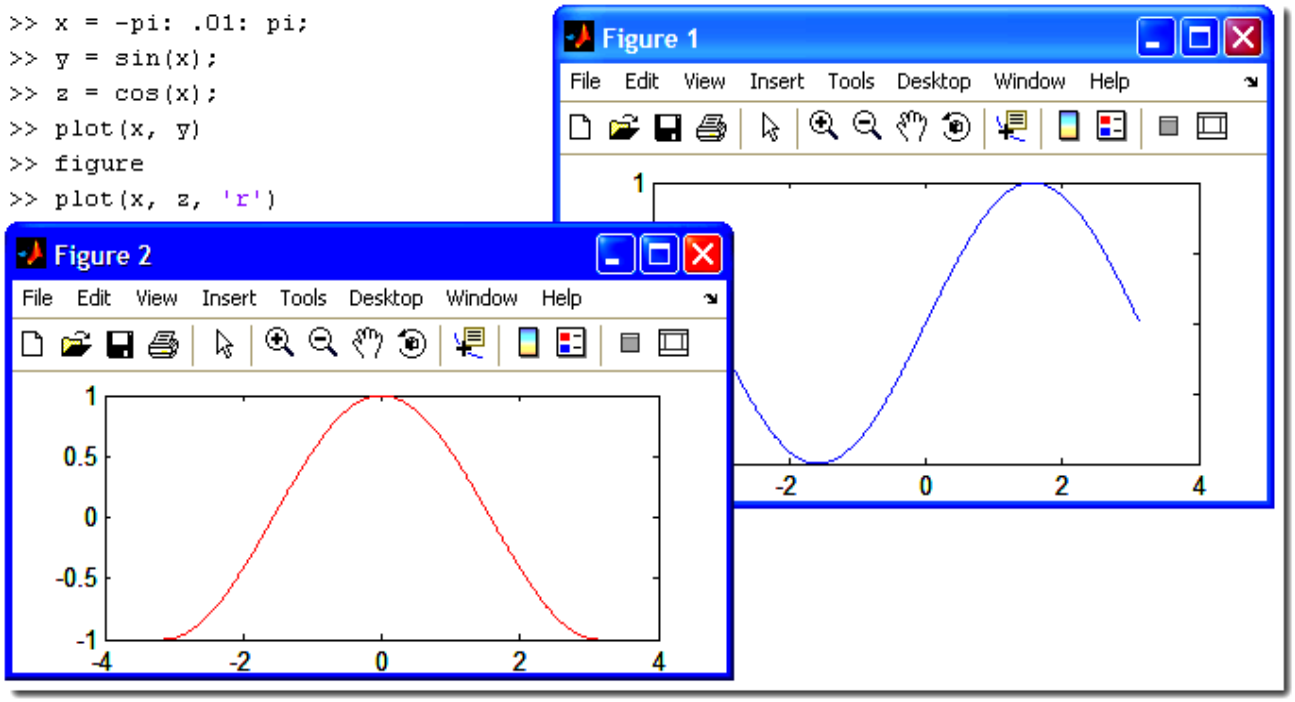
Например:

```
>> x = -pi: .01: pi;
>> y = sin(x);
>> z = cos(x);
>> subplot(211); plot(x, y, 'c')
>> subplot(212); plot(x, z, 'g')
```



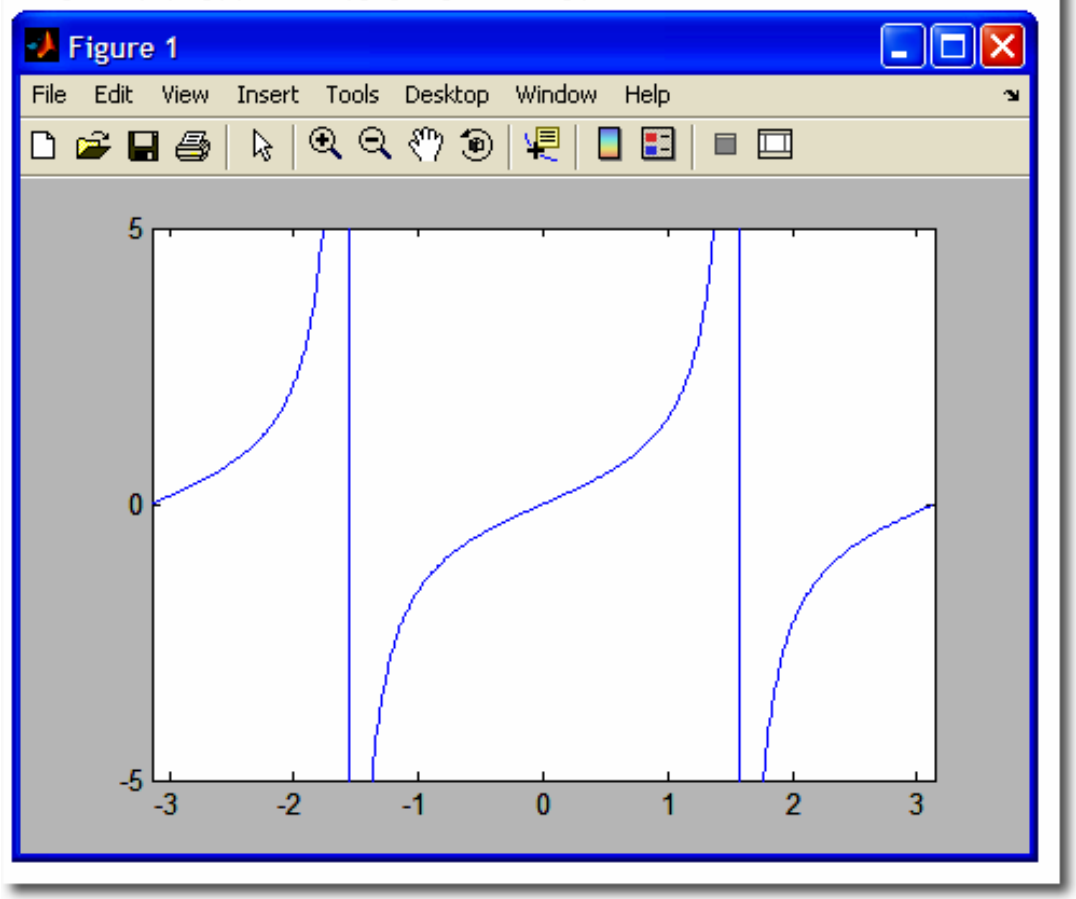
Для построения графиков в разных графических окнах нужно создать новое графическое окно с помощью команды *figure*. Команда *figure* создаёт графическое окно и возвращает указатель на него: *h = figure*, а активизировать ранее созданное окно можно командой *figure(h)*.

```
>> x = -pi: .01: pi;
>> y = sin(x);
>> z = cos(x);
>> plot(x, y)
>> figure
>> plot(x, z, 'r')
```



В среде Matlab для управления масштабом имеется команда *Axis*. Команда ***axis([Xmin Xmax Ymin Ymax])*** задаёт область построения графиков по осям X и Y.Команда используется, если результат автомасштабирования неудовлетворителен.

```
>> x = -pi: .01: pi;
>> y = tan(x);
>> plot(x, y), axis([-pi pi -5 5])
```

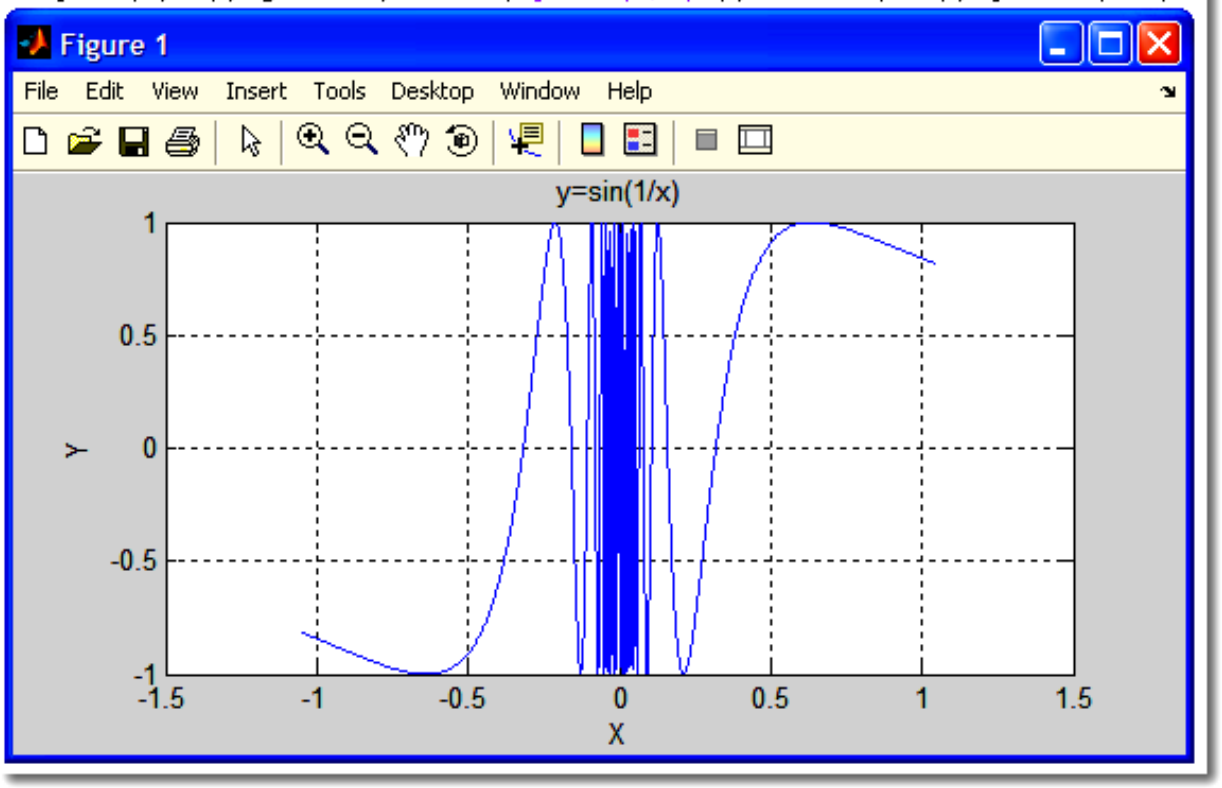


Для графиков можно задать масштабную сетку с помощью команды grid on, заголовок с помощью команды title(’заголовок’), подписи осей координат с помощью команды xlabel(’текст’) и ylabel (’текст’). В заголовках и подписях можно использовать нотацию системы TeX.

А также в Matlab можно построить графики функций, заданных в параметрическом виде. Строятся они при помощи оператора *plot*. Для этого вначале задаётся диапазон построения t, затем вычисляются *x(t)* и *y(t)* и строится график. Графики параметрических функций часто возникают в физических приложениях. Независимая переменная *t* в этом случае имеет смысл времени, а *x* и *y* – координаты. Для построения динамического графика можно использовать функцию *comet(x,y)*.

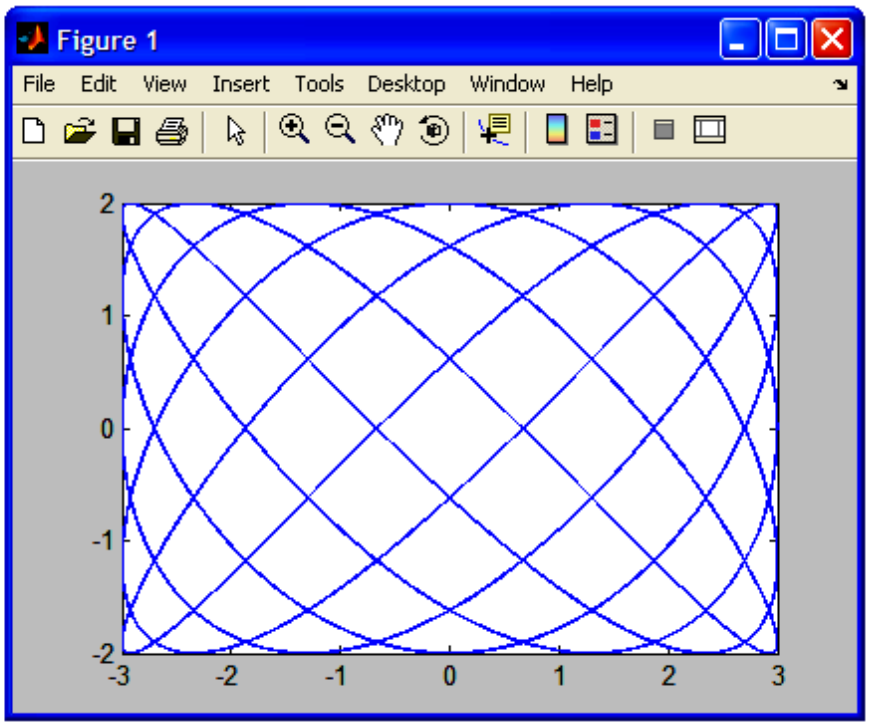
Пример 1 (оформление графика).

```
>> x = -pi/3: .001: pi/3;
>> v = sin(1./x);
>> plot(x, v), grid on, title('y=sin(1/x)'), xlabel('X'), ylabel('Y')
```



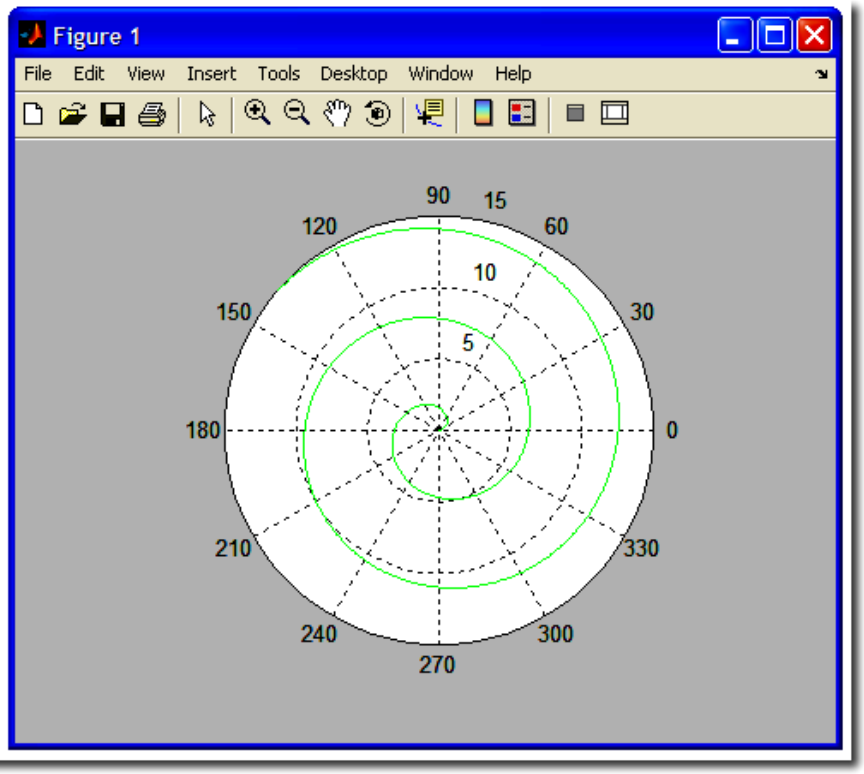
Пример 2 (Графики функций, заданных в параметрическом виде).

```
>> t = 0: .01: 100;
>> x = 3*cos(5*t);
>> y = 2*sin(7*t);
>> plot(x, y)
>>
```



Графики в полярной системе координат строятся аналогично графикам функций в декартовой системе. Для построения используется команда *polar*.

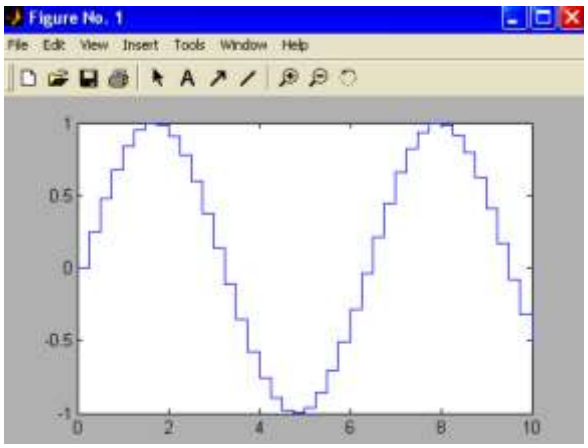
```
>> phi = 0: .01: 15;
>> r = phi;
>> polar (phi, r, 'g')
>>
```



Графическое представление функции в виде ступенчатого графика осуществляется с помощью функции **stairs (x, y).** Для этого сначала надо для значений аргумента x нужно сформировать массив, а затем осуществить вывод функции в виде ступенчатого графика

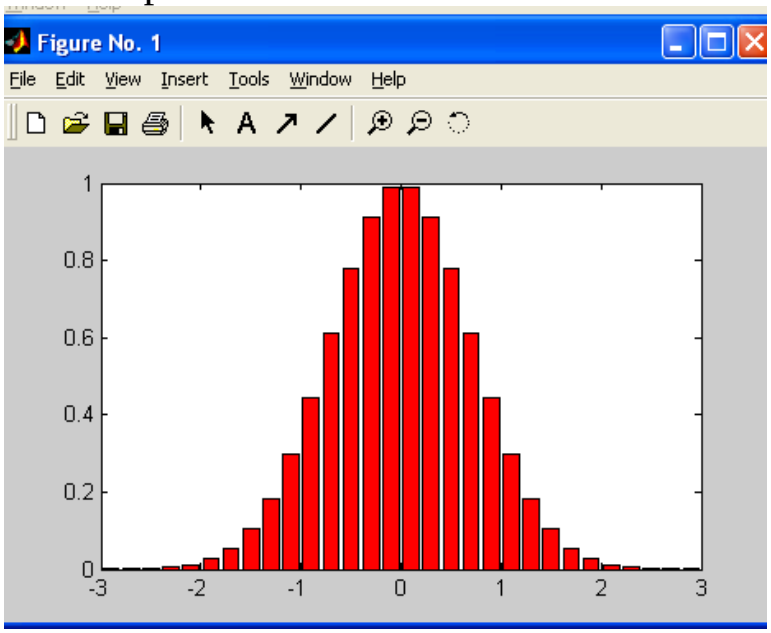
Пример:

```
x = 0:.25:10;
stairs (x,sin(x))
```



В среде Matlab можно также построить график функции в виде столбчатой диаграммы. Это осуществляется с помощью функции **bar(x, y)**

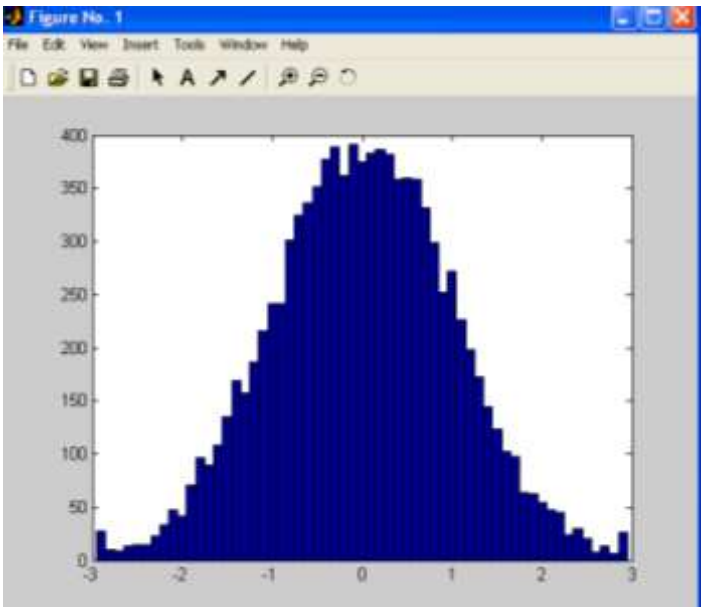
```
x = -2.9:0.2:2.9;
bar(x,exp(-x.*x))
colormap hsv
```



Другим примером является построение графика в виде гистограммы. Это осуществляется с помощью функции **hist(y, x)**

Построим гистограмму случайных величин, которые формируются функцией randn.

```
x = -2.9:0.1:2.9;
y= randn(10000,1);
hist(y,x)
```



Система Matlab имеет очень большие возможности построения трехмерных графиков. В Трёхмерную (3D-) графику Matlab входит построение:

- поверхностей
- контурных диаграмм (линии равного уровня)
- 3D-линий
- векторных полей
- скалярных полей
- и др.

Мы рассмотрим только несколько функций, позволяющих создавать трехмерные графики.

Для построения трехмерного графика z = f(x, y) необходимо иметь матрицы значений переменных x, y. Для определения матриц предназначены следующие функции:

[X,Y] = meshgrid (x, y);

[X,Y] = meshgrid (x);

[X,Y,Z] = meshgrid (x, y, z);

Функция **[X,Y] = meshgrid(x,y)** — преобразует область векторов x, y в массивы X, Y, которые используются для вычисления функции z = f(x, y) и построения графиков.

Строки массива X являются копиями вектора x, а столбцы массива Y - копиями вектора y.

Функция **[X,Y,Z] = meshgrid(x, y, z)** возвращает трехмерный массив для построения трехмерного графика.

Графики трехмерных поверхностей строятся с помощью следующих функций:

plot3 (x, y, z),
plot3 (X, Y, Z),
plot3 (X, Y, Z, s),
plot3 (x l , yl, zl, s1, x2, y2, z2, s2, . . . , xn, yn, zn, sn),

где x, y, z — векторы аргументов функции, X, Y, Z — матрицы одинакового размера, s— стили линий и точек графика, аналогично функции **plot ().**

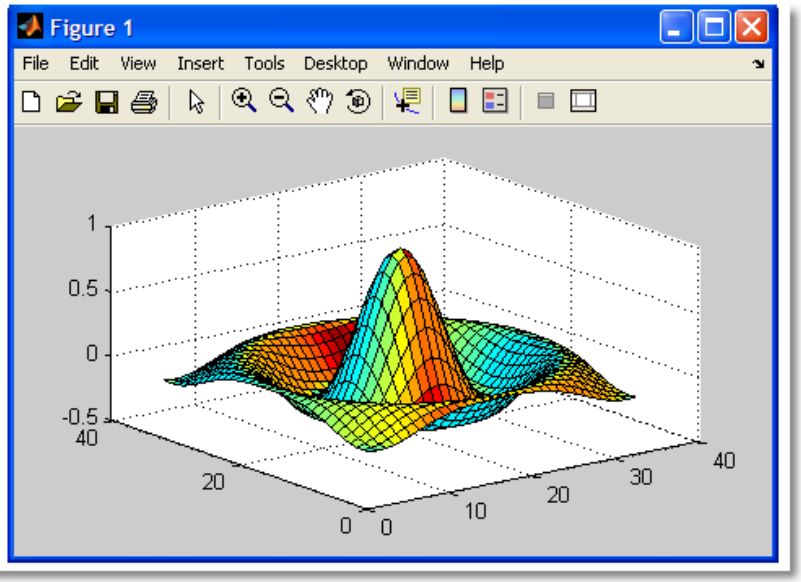
Приведенные функции строят точки трехмерного графика и соединяют их отрезками прямых в соответствии с заданным стилем.

Функция **plot3(xl,yl,zl,sl,x2,y2,z2,s2,...,xn, yn, zn,sn)** строит на одном рисунке n функций.

Пример. Построить график функции **z = ln x + ln y** в диапазоне аргументов [-4; 4] с шагом h = 0.1.

Рассмотрим пример построения поверхности f(x,y)=sin(r)/r, где r=sqrt(x2+y2).

```
>> [X,Y] = meshgrid(-8:.5:8);
>> R = sqrt(X.^2 + Y.^2) + eps;
>> Z = sin(R)./R;
>> surfl(Z)
>>
```



Функции для построения поверхностей:

Функция	*Для чего используется*
mesh, surf	Построение поверхностей
meshc, surfc	Строит поверхность и контурную диаграмму под ней
meshz	Поверхность на «пьедестале»
surfl	Подсвеченная поверхность
contour	Контурная диаграмма
plot3	Трёхмерная линия (параметрическое

	задание)
comet3	Движение по трёхмерной линии

О других графических функциях можно узнать в системе помощи Matlab.

4. Программирование в Matlab

Написание программ – это альтернатива работе в командной строке. Программный код Matlab размещают в файлах с расширением «m» (m-файлах). m-файлы бывают двух видов:

- *скрипты (scripts);*
- *функции (functions).*

К сожалению, Matlab плохо понимает кириллицу…

Скрипты представляют собой последовательности команд Matlab, как если бы мы перенесли их из командного окна в отдельный файл. Скрипт вызывается по имени через командную строку и выполняется в режиме интерпретатора, и они полезны для автоматизации последовательности действий, которые выполняются многократно. Скрипты не могут принимать параметры и возвращать аргументы. Они хранят значения своих переменных в рабочем пространстве, где переменные доступны для других скриптов и из командной строки.

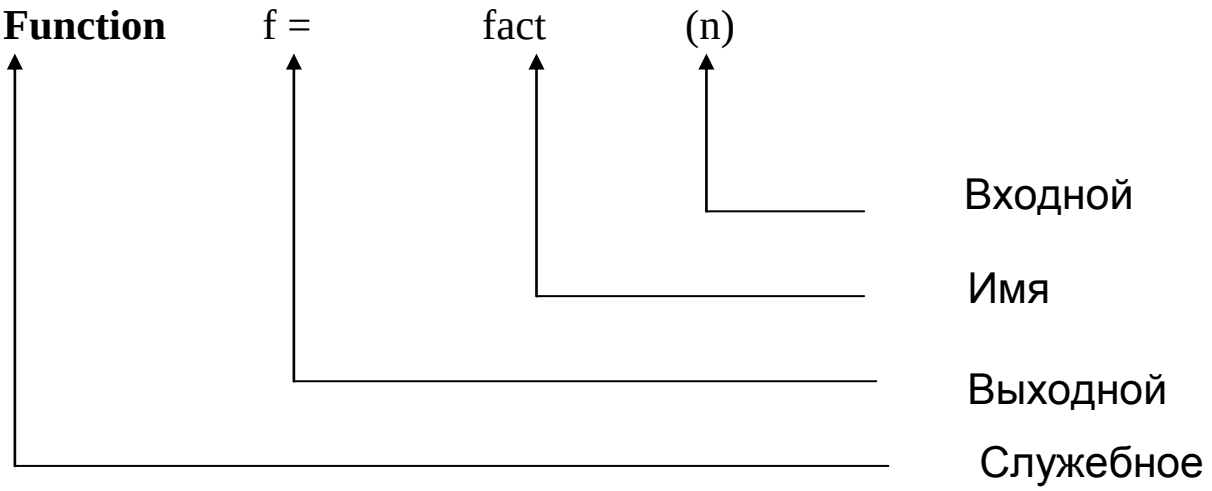
Функции- это специальный вид m-файлов. В отличие от скриптов могут принимать аргументы и возвращать значения. Использование функций позволяет структурировать программу и избежать повторения кода.

Создание функции преследует целью расширение языка. Переменные, определённые внутри функции являются *локальными,* то есть видны только внутри самой функции. Функция имеет собственное имя. Кроме того, с ней связано имя m-файла, в котором функция записана. При этом будем соблюдать правило: имя функции и имя m-файла должны быть одинаковы.

Функция состоит из *заголовка* и *тела*

```
function f = fact(n)          Заголовок
% Вычисляет факториал.       Линия H1
% FACT(N) возвращает N!,     Help
f = prod(1:n);                Тело функции
```

При этом **H1** и **Help** выводятся по команде help <имя функции>. Фактически, функция отличается от скрипта наличием заголовка и способом вызова.



Комментарии используются для пояснения кода и временного исключения кода из текста. Они могут быть *строчными* и *блочными*. Строчные начинаются с символа «%». С этого места и до конца строки всё игнорируется компилятором %. Блочные начинаются с символа «%{» и заканчиваются символом «%}» При этом эти символы должны обязательно стоять в отдельных строках.

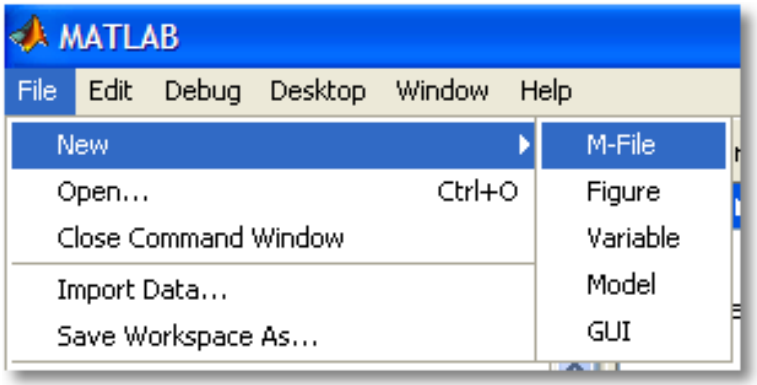
Можно автоматически закомментировать блок текста. Для этого нужно:

- выделить блок;
- щёлкнуть правой кнопкой;
- выбрать Comment (или Ctrl+R).

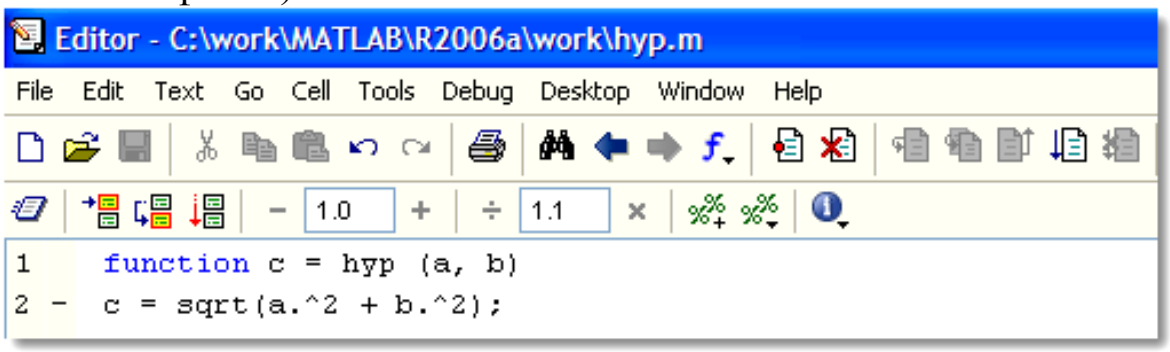
Для снятия комментарий нужно:

- выделить закомментированный блок;
- щёлкнуть правой кнопкой;
- выбрать Uncomment (или Ctrl+T).

Создать **m-файл** можно в любом текстовом редакторе, например, во встроенном редакторе при помощи меню или командой edit <имя файла>.



Функция вызывается по своему имени (которое совпадает с именем её m-файла).




```
>> a = [1:3]; b = [5:7];
>> z = hyp(a, b)

z =

    5.0990    6.3246    7.6158
```

При написании функций в Matlab можно проводить проверку количества входных и выходных параметров. Для этого в описании функции используют служебные слова:

- nargin- количество входных параметров;
- nargout- количество выходных параметров.

В файлах-функциях Matlab могут быть реально описаны несколько функций. Синтаксически это оформляется как две (или более) функций, записанных в одном файле. При вызове такого m-файла происходит запуск самой первой функции. При этом её имя должно совпадать с именем файла. Описание следующих функций локально. Обычно они используются как вспомогательные для первой функции.

```
function [avg, med] = newstats(u) % Primary function
% NEWSTATS Find mean and median with internal functions.
n = length(u);
avg = mean(u, n);
med = median(u, n);

function a = mean(v, n)              % Subfunction
% Calculate average.
a = sum(v)/n;

function m = median(v, n)            % Subfunction
% Calculate median.
w = sort(v);
if rem(n, 2) == 1
   m = w((n+1) / 2);
else
   m = (w(n/2) + w(n/2+1)) / 2;
end
```

```
function x = A(p1, p2)
...
   function y = B(p3)
   ...
   end

   function z = C(p4)
   ...
   end
...
end
```

```
function x = A(p1, p2)
...
   function y = B(p3)
   ...
      function z = C(p4)
      ...
      end
   ...
   end
...
end
```

При вызове m-файла сравнительно много времени тратится на его компиляцию. Чтобы сократить время выполнения можно предварительно перевести m-файл в p-код («пи-код») с помощью команды **pcode <имя m-файла>**. Откомпилированный в псевдокод файл получает расширение «p». Такой файл будет выполняться быстрее, чем обычный m-файл.

Интерактивный ввод данных используется при написании скриптов. Для ввода числовых данных применяют функцию input по формату **x = input('строка приглашения')**. Введённое

пользователем значение сохранится в переменной x. Для ввода строковых данных функция **input** вызывается с дополнительным параметром: **c = input('строка приглашения','s').** Кроме того, имеется Си-подобная функция **sscanf.**

```
1 -  name = input ('Hello! What is your name?\n', 's');
2 -  y = input (['Very good, ', name, '. And how old are you?\n']);
3 -  disp(['Resume: Mr(s) ', name, ' is ', int2str(y), ' years old.'])
```

```
Command Window
>> hyp
Hello! What is your name?
Andy
Very good, Andy. And how old are you?
21
Resume: Mr(s) Andy is 21 years old.
```

Для вывода данных в командное окно используют команду **disp (от *display*)** по формату **disp(<выводимая строка>).** Если выводимое значение – число, то вначале его преобразуют к строковому типу при помощи функций **int2str** или **num2str.** Конкатенацию строк производят как для одномерных векторов-строк.

```
>> x = 2 + pi;
>> disp(['x = ', num2str(x)] )
x = 5.1416
```

Кроме того, имеется Си-подобная функция **sprintf.**

Как и любой процедурный язык высокого уровня, Matlab позволяет использовать при написании программ:

- следование,
- ветвление,
- циклы,
- пользовательские функции.

Следование реализуется перечислением каждого из операторов в отдельной строке, либо в одной строке через запятую (или точку с запятой).

Ветвление реализуется в двух вариантах:

– при помощи оператора **if,**

– при помощи оператора **switch**.

Простейшая форма оператора **if:**

if <логическое выражение>
<операторы>
end

```
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

В полном варианте оператора могут использоваться слова **else** и **elseif.** Слово **elseif** может использоваться в одном операторе многократно с указанием условия. Слово **else** – только один раз в конце оператора и без условия.

```
if n < 0              % If n negative, display error message.
    disp('Input must be positive');
elseif rem(n,2) == 0 % If n positive and even, divide by 2.
    A = n/2;
else
    A = (n+1)/2;      % If n positive and odd, increment and divide.
end
```

Циклы. В Matlab имеется два вида циклов:

- цикл с параметром **for,**
- цикл с предусловием **while.**

А также имеются оператор досрочного выхода из цикла **break** и оператор перехода к следующей итерации **continue.**

Циклы с параметром имеют вид:

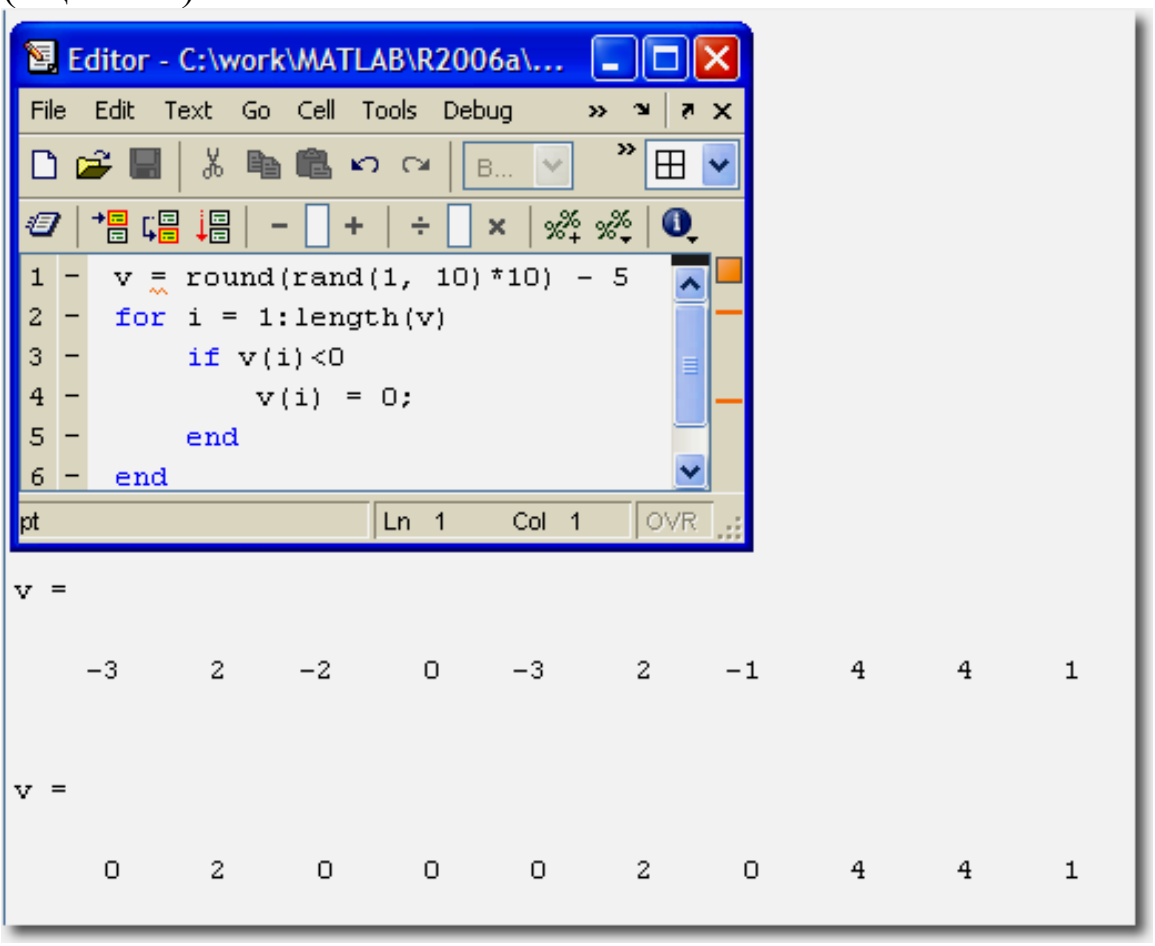
```
for index = start:increment:end
    statements
end
```

```
for n = 2:6
    x(n) = 2 * x(n - 1);
end
```

```
for m = 1:5
    for n = 1:100
        A(m, n) = 1/(m + n - 1);
    end
end
```

Обычно цикл for используется для обработки массивов. Важно помнить, что если есть возможность обойтись без этого цикла (применить матричные или векторные операции), то **лучше избавиться от явного цикла.** В этом случае программа будет работать на порядок быстрее.

Пример. Замена отрицательных элементов вектора на нули (с циклом).

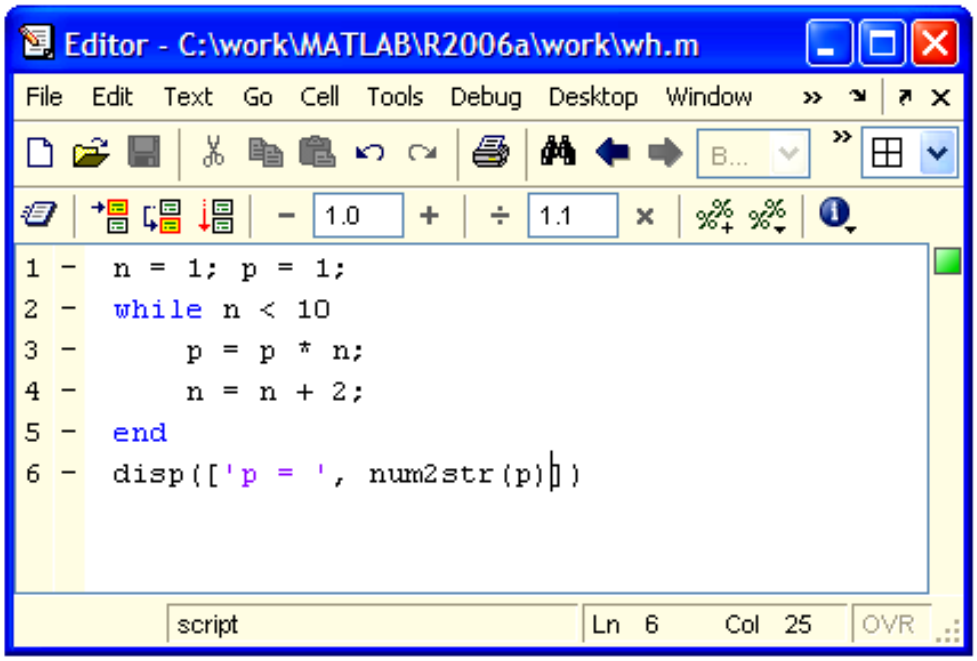


Цикл с предусловием имеет вид:

while <логическое выражение>
<операторы>
end

Операторы выполняются, пока логическое выражение есть истина (true).

```
>> wh
p = 945
>>
```



Операторы **break** и **continue** аналогичны одноимённым операторам Паскаля. Оператор **break** производит досрочный выход из цикла **for** или **while,** а **continue** прекращает выполнение текущей итерации и переходит к следующей.

Напишем скрипт, который вводит с клавиатуры произвольное количество чисел. Если число положительное, то оно прибавляется к сумме, если отрицательное, то пропускается. Ноль – признак окончания работы.

```
Input a number: 1
Input a number: 2
Input a number: -3
Input a number: 3
Input a number: 0
s = 6
>>
```

```
1 - s = 0; p = 1;
2 - while p
3 -     x = input('Input a number: ');
4 -     if x == 0
5 -         break
6 -     end
7 -     if x < 0
8 -         continue
9 -     end
10 -    s = s + x;
11 - end
12 - disp(['s = ', num2str(s)])
```

Выход из функции в вызывающую программу обеспечивается выполнением последнего ее оператора или командой **return**. Кроме упомянутых основных операторов, традиционных для любой системы программирования, остановимся на ряде операторов обеспечения пользовательского интерфейса.

Приостановка выполнения программы может быть предусмотрена включением в текст команды:

pause (приостановка до нажатия любой клавиши),

pause (n) (приостановка на n сек),

keyboard (приостановка с возможностью выполнять практически любые команды и последующим возвратом в программу командой **return**).

Можно построить выбор варианта с клавиатуры созданием **меню**:

<переменная>=**menu**(‘заголовок’,’выбор1’,’выбор2’,...)

Например, команда:

k=menu(‘Использовать метод’,’Гаусса’,’Краута’,’Простой итерации’) создаст на экране всплывающее меню с указанными пунктами клавишами и щелчок по клавише задаст значение переменной k, равное 1, 2 или 3.

4. Аналитические вычисления в Matlab

Изначально Matlab имел средства только для численного анализа. Сегодня в Matlab встроены средства аналитических (символьных) вычислений. **Symbolic Math Toolbox-** является вычислительным ядром системы Maple V. Установка Maple не требуется.

Для символьного анализа в среде Matlab требуется создать символьные переменные и функции. Символьные переменные создаются по одной переменной **x=sym(’x’).** Так же можно создать целое символьное выражение **syms x y z.** Символьные функции определяются через символьные переменные **f=x^2+y.** Для построения символьных функций можно воспользоваться командой **ezplot.** Представить в стандартной форме можно с помощью команды **pretty.**

```
>> whos
  Name       Size                     Bytes  Class

  a          1x1                        126  sym object
  ans        1x1                        126  sym object
  f          1x1                        196  sym object
  x          1x1                        126  sym object
  y          1x1                        126  sym object

Grand total is 45 elements using 700 bytes
```

```
>> x=sym('x')

x =

x


>> syms y a
>> f=(sin(x)+a)^2*cos(y)^2/sqrt(abs(a-y))

f =

(sin(x)+a)^2*cos(y)^2/abs(a-y)^(1/2)


>> pretty(f)

                                   2       2
                            (sin(x) + a)  cos(y)
                           ---------------------
                                      1/2
                              | a - y |
```

В системе Matlab имеются следующие символьные вычисления: преобразования и анализа (дифференцирование, пределы, интегрирование, разложение в ряд Тейлора), упрощения и подстановки, точной арифметики, линейной алгебры и решения уравнений и их систем (обычных и дифференциальных).

Символьное вычисление пределов (сводная таблица).

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')`

Символьное вычисление интегралов (сводная таблица).

Mathematical Operation	MATLAB Command
$\int x^n dx = \frac{x^{n+1}}{n+1}$	`int(x^n)` or `int(x^n,x)`
$\int_0^{\pi/2} \sin(2x)dx = 1$	`int(sin(2*x),0,pi/2)` or `int(sin(2*x),x,0,pi/2)`

$g = \cos(at+b)$
$\int g(t)dt = \sin(at+b)/a$
	`g = cos(a*t + b)`
	`int(g)` or
	`int(g,t)`

$\int J_1(z)dz = -J_0(z)$	`int(besselj(1,z))` or `int(besselj(1,z),z)`

Рассмотрим некоторые из них на примерах.

Пример. Символьное вычисление пределов.

```
>> syms h  n  x
>> limit((cos  (x + h) - cos(x))/h,  h,0)  %ввычисляе предел при h → 0
ans =
- sin(x)
>> limit((1  + x/n)^ n, n, inf)   %a ттепер при n → ∞
ans =
exp(x)
```

Пример. Символьное вычисление односторонних пределов.

```
>> limit(x/ab  s(x), x,0, 'left' )
ans =
-1
>> limit(x/ab  s(x), x,0, 'right' )
ans =
1
```

Пример. Символьное дифференцирование.

```
>> syms  x
>> f  = sin(5 * x)
f  =
     sin(5 * x)
>> diff(f)     % вычисляем производную по x
ans =
       5 * cos(5 * x)
>> diff(f,2)     % ттепер вторую производную по x
ans =
- 25 * sin(5 * x)
>> c = sym('5' ); diff(c)  % производную константы
ans =
       0
```

Пример. Символьное вычисление частных производных.

```
>> syms s t
>> f = sin(s * t);
>> diff(f,t)

ans =

cos(s*t)*s


>> % по умолчанию используется переменная t
>> % убедиться в этом можно при помощи функции findsym:
>> findsym(f,1)

ans =

t
```

Пример. Символьное разложение в ряд Тейлора.

$$\sum_{n=0}^{\infty} (x-a)^n \frac{f^{(n)}(a)}{n!}$$

Можно и с помощью команды taylortool

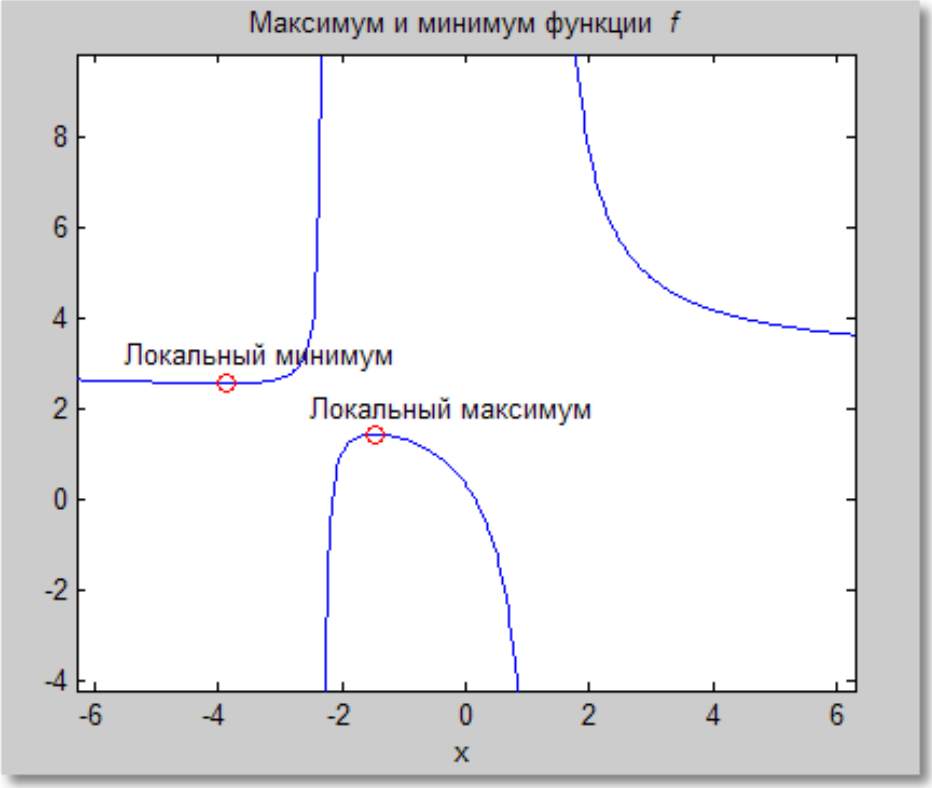
```
>> syms x
>> f = 1/(5+4*cos(x))

f =

1/(5+4*cos(x))


>> T = taylor(f,8)

T =

1/9+2/81*x^2+5/1458*x^4+49/131220*x^6


>> pretty(T)

                                  2           4     49    6
                1/9 + 2/81 x  + 5/1458 x  + ------ x
                                             131220
```

Пример. Нахождение экстремума функции.

```
>> f1 = diff(f)

f1 =

(6*x+6)/(x^2+x-3)-(3*x^2+6*x-1)/(x^2+x-3)^2*(2*x+1)


>> % упростим полученное выражение:
>> pretty(f1)

                                 2
  6 x + 6      (3 x  + 6 x - 1) (2 x + 1)
 ---------- - --------------------------
  2                    2          2
 x  + x - 3          (x  + x - 3)
>> % Приравняем производную к нулю и решим уравнение
>> extr = solve(f1)

extr =

 -8/3-1/3*13^(1/2)
 -8/3+1/3*13^(1/2)
```

Построение экстремумов функции.

```
ezplot(f); hold on
plot(double(extr), double(subs(f,extr)),'ro')
title('Максимум и минимум функции \it f')
text(-5.5,3.2,'Локальный минимум')
text(-2.5,2,'Локальный максимум')
hold off
```



В системе Matlab операции над полиномами реализуются при помощи функций **collect, expand, factor, horner.** Операция **collect** – вычисляет коэффициенты при степенях независимой переменной (по умолчанию – *x*). Можно явно задать имя независимой переменной в виде: **collect (f, VarName).**

f	collect(f)
(x-1)*(x-2)*(x-3)	x^3-6*x^2+11*x-6
x*(x*(x-6)+11)-6	x^3-6*x^2+11*x-6
(1+x)*t + x*t	2*x*t+t

Операция **expand** – представляет полином суммой степеней без приведения подобных.

f	expand(f)
a*(x + y)	a*x + a*y
(x-1)*(x-2)*(x-3)	x^3-6*x^2+11*x-6
x*(x*(x-6)+11)-6	x^3-6*x^2+11*x-6
exp(a+b)	exp(a)*exp(b)
cos(x+y)	cos(x)*cos(y)-sin(x)*sin(y)
cos(3*acos(x))	4*x^3-3*x

Операция **factor** – разлагает полином на множители, если эти множители имеют рациональные коэффициенты.

f	factor(f)
x^3-6*x^2+11*x-6	(x-1)*(x-2)*(x-3)
x^3-6*x^2+11*x-5	x^3-6*x^2+11*x-5
x^6+1	(x^2+1)*(x^4-x^2+1)

Операция **simplify** реализует мощный алгоритм упрощения с использованием тригонометрических, степенных, логарифмических, экспоненциальных функций, а также спецфункций (Бесселя, гипергеометрической, интеграла ошибок и пр.), а операция **simple** пытается получить выражение, которое представляется меньшим числом символов, чем исходное, последовательно применяя все функции упрощения Symbolic Math Toolbox. Иногда **simple** даёт более удачное решение, чем **simplify**

f	simplify(f)	simple(f)
(1/a^3+6/a^2+12/a+8)^(1/3)	((2*a+1)^3/a^3)^(1/3)	(2*a+1)/a
syms x y positive log(x*y)	log(x)+log(y)	log(x*y)

f	simplify(f)
x*(x*(x-6)+11)-6	x^3-6*x^2+11*x-6
(1-x^2)/(1-x)	x+1
(1/a^3+6/a^2+12/a+8)^(1/3)	((2*a+1)^3/a^3)^(1/3)
syms x y positive log(x*y)	log(x)+log(y)
exp(x) * exp(y)	exp(x+y)
besselj(2,x) + besselj(0,x)	2/x*besselj(1,x)
gamma(x+1)-x*gamma(x)	0
cos(x)^2 + sin(x)^2	1

Операция **simple** особенно эффективна при работе с тригонометрическими выражениями.

f	simple(f)
cos(x)^2+sin(x)^2	1
2*cos(x)^2-sin(x)^2	3*cos(x)^2-1
cos(x)^2-sin(x)^2	cos(2*x)
cos(x)+(-sin(x)^2)^(1/2)	cos(x)+i*sin(x)
cos(x)+i*sin(x)	exp(i*x)
cos(3*acos(x))	4*x^3-3*x

Операция **subs** подставляет одно символьное выражение в другое. Общий формат операции имеет вид:

- subs(<куда>, <вместо чего>, <что>)

Подстановка вместо переменной её числового значения приводит к вычислению символьной функции от значения аргумента.

Рассмотрим пример подстановки:

```
>> f = sym ('(a^2 + b^2)/(a^2 - b^2) + a^4/b^4');
>> pretty(f)

                                  2    2     4
                                 a  + b     a
                                 ------- + ----
                                  2    2     4
                                 a  - b     b
>> f = subs(f, 'a', '(exp(x) + exp(-x))');
>> f = subs(f, 'b', '(sin(x) + cos(x))') ;
>> pretty(f)

                          2                       2                         4
        (exp(x) + exp(-x))  + (sin(x) + cos(x))     (exp(x) + exp(-x))
        ---------------------------------------- + ------------------
                          2                       2                         4
        (exp(x) + exp(-x))  - (sin(x) + cos(x))     (sin(x) + cos(x))
```

В системе Matlab точные вычисления реализуются функцией vpa (Variable-Precision Arithmetic). Формат вызова функции имеет вид:

- vpa(<выражение>, <значащих цифр>)
-

```
>> vpa(pi,50)

ans =

3.1415926535897932384626433832795028841971693993751


>> vpa((1 + sqrt(5))/2, 10)

ans =

1.618033989


>> vpa(exp(1), 70)

ans =

2.7182818284590455348848081484902650117874145507812500000000000000000000
```

В системе Matlab решение уравнений и систем выполняет команда **solve.** До 4-го порядка включительно решаются точно. Ответ выводится в степенях рациональных чисел. Уравнения высших порядков и трансцендентные, как правило, точно не решаются и в

этом случае выводится приближённый результат. С целью сокращения записи при выводе могут использоваться подстановки.

```
>> syms x
f = sym('x^3 - x^2 - 5*x + 1');
r = solve(f, x);
pretty(r)

        [                  1/2 1/3                   1                ]
        [1/3 (10 + 6 I 111   )    + 16/3 -------------------- + 1/3]
        [                                        1/2 1/3      ]
        [                                (10 + 6 I 111   )    ]
I -
        [                   1/2 1/3                1
это     [- 1/6 (10 + 6 I 111   )    - 8/3 -------------------- + 1/3
мнимая  [                                          1/2 1/3
        [                                (10 + 6 I 111   )
единица
                  1/2 /                  1/2 1/3                  1          \]
         + 1/2 I 3    |1/3 (10 + 6 I 111   )    - 16/3 --------------------|]
                      |                                              1/2 1/3|]
                      \                                  (10 + 6 I 111   )   /]

        [                   1/2 1/3                1
        [- 1/6 (10 + 6 I 111   )    - 8/3 -------------------- + 1/3
        [                                          1/2 1/3
        [                                (10 + 6 I 111   )

                  1/2 /                  1/2 1/3                  1          \]
         - 1/2 I 3    |1/3 (10 + 6 I 111   )    - 16/3 --------------------|]
                      |                                              1/2 1/3|]
                      \                                  (10 + 6 I 111   )   /]
```

Решение систем также выполняет команда **solve.** При этом входные аргументы левые части уравнений, переменные, по которым нужно разрешить систему, например: s = solve(f1, f2, x1, x2). Выходной аргумент структура (запись) s с полями (в данном случае) x1 и x2, хранящими символьное представление решения.

5. Алгоритмы и технологии численного вычисления в системе Matlab

Система Matlab имеет большое число способов численного интегрирования. Численное интегрирование необходимо в следующих случаях:

• первообразная не выражается через элементарные функции;

• аналитическое выражение интеграла слишком сложное;

• подынтегральная функция задана в табличной форме или в виде матрицы.

При вычислениях интегралов численными методами подынтегральную функцию целесообразно представлять в наиболее простом виде. Это может ускорить вычисления. Упрощение подынтегральной функции можно выполнить, воспользовавшись функцией **simplify(y)**.

Имеют место случаи, когда система до упрощения не может вычислить неопределенный интеграл и легко его определяет после упрощения.

Метод вычисления интеграла выбирает пользователь. В этом особенность системы MATLAB. С помощью MATLAB студент имеет возможность сравнивать различные методы численного интегрирования.

Существует ряд способов численного интегрирования. Во всех таких способах вычисление осуществляется по приближенным формулам, называемым квадратурными. Приведем некоторые из них.

Формулы прямоугольников. Формулы прямоугольников представляются в следующем виде:

$$\int_a^b f(x)dx = \left\{ \begin{array}{l} h\sum_{k=0}^{n-1} y_k \\ h\sum_{k=1}^{n} y_k \end{array} \right\},$$

где: h — шаг интегрирования, y_k — значение подынтегральной функции при аргументе x_k, k=0,1,2,..., n и n=(b-a)/h - число частей, на которые разбивается область интегрирования.

Одна из формул дает значение интеграла с избытком, другая с недостатком. Какая из них выдает решение с избытком или с недостатком, зависит от вида подынтегральной функции.

Формула трапеций. Эта формула имеет вид:

$$\int_a^b f(x)dx = h\left(\frac{y_0}{2} + \sum_{k=1}^{n-1} y_k + \frac{y_n}{2}\right),$$

где y_0 — значение подынтегральной функции при x=a, y_n — значение подынтегральной функции при x=b, h — шаг интегрирования.

Формула парабол (Симпсона). Эта формула имеет вид:

$$\int_a^b f(x)dx = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + 4y_5 + \ldots$$

$+4y_{n-1}+y_n)$.

В этой формуле ординаты с нечетными индексами умножаются на 4, а с четными — на 2. Предполагается, что n — число четное.

При нечетном n формула имеет вид:

$$\int_a^b f(x)dx=\frac{h}{3}(y_0+4y_1+2y_2+4y_3+2y_4+4y_5+...+2y_{n-1}+y_n).$$

Крайние ординаты имеют коэффициент, равный 1.

Существует много других квадратурных формул вычисления интегралов: Котеса, Чебышева, Гаусса и др.

В системе Matlab вычисление интегралов реализовано **численными методами трапеций, парабол (Симпсона) и Ньютона - Котеса.**

Метод трапеций. Метод трапеции реализован в Matlab несколькими функциями, приведенными ниже.

1. Функция cumtrapz(y). Осуществляет вычисление интеграла в случае, когда значения функции y заданы в виде вектора или матрицы неограниченных размеров. Откликом этой функции является *n* интегралов, где *n* — число элементов вектора или число элементов в каждом столбце матрицы. Такое вычисление интеграла называется ***интегрированием с накоплением.***

Пример 1. Пусть функция y(x) имеет значения, представленные в виде следующего вектора: y= [1,2,3,4,5,6,7,8,9,10]. Необходимо вычислить $\int_a^b y(x)dx$

При этом a=1; b=1,2, 3,..., 10.

Функция вычисления интеграла методом трапеций будет иметь вид:

```
>> y=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
>> cumtrapz (y)
ans =
        0    1.5000    4.0000    7.5000   12.0000   17.5000   24.0000
31.5000   40.0000   49.5000
```

Пример 2. Пусть необходимо вычислить интеграл вида $\int_0^{10}(3e^x+\ln x+1)dx$.

Чтобы вычислить этот интеграл с помощью функции cumtrapz(), следует сначала вычислить 10 ординат подынтегральной функции, представив их в виде вектора.

Программа вычисления интеграла с накоплением будет иметь вид:

```
» x=1:1:10;
» y=3*exp(x)+log(x)+l;
» cumtrapz(y)
ans =
  1.0e+004 *
        0    0.0017    0.0060    0.0174    0.0481    0.1311    0.3564
0.9684    2.6313    7.1510
```

Существует модификация данной функции **cumtrapz (x, y).** Основным недостатком метода трапеций является большая погрешность результата вычисления интеграла.

2. Функция trapz(y). Отличие данной функции от функции cumtrapz(y) состоит в том, что осуществляется простое интегрирование без накопления, то есть **trapz(y)** возвращает не столько интегралов, насколько шагов разбивается область интегрирования, а общее значение интеграла.

Пример 3. Вычислить интеграл вида $\int_0^{10}(xe^x + \ln x + 1)dx$ с шагом 0,5.

```
x=1: 0.5: 10;
y=x.*exp(x) + log(x) + 1;
trapz(y)
 ans =  4.0657e+005.
```

Существует модификация данной функции **trapz (x, y).** Следует иметь в виду, что при вычислении интеграла с помощью функции trapz (x, y) его значение зависит от шага интегрирования.

Метод парабол (Симпсона). Для его реализации в системе Matlab используются следующие функции:

```
quad('fun', a, b),
quad('fun', a, b, tol),
quad('fun', a, b, tol, trace),
dblquad('fun', a, b, c, d),
dblquad('fun', a, b, c, d, tol).
```

В этих функциях приняты обозначения:

1) 'fun' - подынтегральная функция, взятая в одинарные кавычки;
2) a, b - пределы интегрирования;
3) tol - относительная погрешность, задаваемая пользователем (по умолчанию tol = 10e $^{-3}$);
4) c,d - пределы интегрирования по другой переменной (внешней) при вычислении двойного интеграла;
5) trace — число, отличное от нуля, по которому система показывает ход вычислительного процесса.

Рассмотрим перечисленные функции и приведем примеры.

1. Функция quad('fun', a, b). Функция вычисляет пределенный интеграл $\int_a^b f(x)dx$ с погрешностью, не превышающей 10^{-3}.

Пример 4. Подынтегральная функция имеет вид: f(x) = e ^{x} + x ^{2} + 2 sin x - 5 .

Необходимо вычислить $\int_1^5 f(x)dx$. интеграл.

Решение:

```
>> y = ‘exp(x) + x.^2 + 2*sin(x) – 5’;
>> quad (y, 1, 5)
 ans = 167.5415
```

Функция может быть представлена одной строкой:

```
quad (‘exp(x) + x.^2 + 2*sin(x) – 5’, 1, 5).
```

2. Функция dblquad('fun',a,b,c,d).В функции dblquad ('fun', a, b, c, d) приняты следующие обозначения:

'fun' - это функция с двумя переменными;
a, b - пределы по внутренней переменной;
c, d - пределы по внешней переменной.

Пример 5. Пусть функция двух переменных имеет вид: z = x^{2} + y^{2} - 2 .Необходимо вычислить интеграл $\int_1^2\int_0^3 z(x,y)dxdy$

Решение:

```
>> z='x.^2 + y.^2 - 2';
>> dblquad (z,1,2,0,3)
 ans =  10.
```

Вычисления кратных интегралов. Наиболее просто кратный интеграл вычислить путем интегрирования ответа, полученного от предыдущего значения интеграла.

Пример. Пусть необходимо вычислить двойной неопределенный интеграл

$$\int\int \frac{x}{1-x^2}dx$$

Решение.

```
>> syms x;
>> y=x/(1-x^2);
>> int(int(y))
 ans =  -1/2*log(x-1)*(x-1)+x-1/2*log(x+1)*(x+1)
```

Лабораторная работа №1

Введение в систему научных и инженерных расчетов Matlab

Целью данной работы является ознакомление с системой научных и инженерных расчетов Mat Lab, получение начальных сведений об окне управления, окне встроенного редактора, ознакомление с простейшими операциями с числами, векторами и матрицами, элементарными математическими функциями и создание М-файлов.

Задание на лабораторную работу

1. Осуществить ввод действительного числа $2{,}15*10^{-7}$.
2. Выполнить простую арифметическую операцию 8,3/6*2,7-0,001^2*3,14
3. Осуществить ввод комплексного числа, действительная часть которого равна 4, а мнимая равна -9.
4. Выполнить простую арифметическую операцию с двумя комплексными числами, используя одну из дополнительных функций комплексного аргумента.
5. Вычислить значение одной из элементарных математических функций (смотри стр. 6 и 7)
6. Сформировать вектор из 5 любых неотрицательных элементов.
7. Сформировать матрицу размером 3x4 с 1 по главной диагонали и нулевыми остальными элементами.
8. В созданной матрице извлечь элемент 2-й строки и 3-столбца
9. Растянуть данную матрицу в один вектор

10. Создать 2 вектора x и y по 3 элемента каждый и провести операции сложения, вычитания, транспонирования векторов, и их перемножения
11. Создать М-файл, реализующий вычисление следующей функции

$$y = d^3 * ctg(x) * \sqrt{\sin^4(x) - \cos^4(x)}$$

Лабораторная работа №2
Графика в Matlab

Целью данной работы является ознакомление с графической системой Matlab, получение начальных сведений об окне управления, окне встроенного редактора, ознакомление с функциями построения двумерных, полярных и трехмерных графиков.

Задание на лабораторную работу

1. Построить график функции y = (cos (x/π+ π)+sin(x))/2; на промежутке от -3π до +3π с шагом π/50.

Этот график выполнить зеленым цветом, точки графика в виде звездочек, линия сплошная.

2. Добавить к полученному графику координатную сетку, заголовок и названия осей.

3. Осуществить вывод функции y = cos (x/π+ π) в виде ступенчатого графика в диапазоне от 0 до 100 с шагом 0.5

4. Построить график функции $y = e^x$ в виде столбчатой диаграммы на отрезке от -3 до 3.

5. Построить 4 графика произвольных функций в одном графическом окне.

6. Построить график функции **z = sin x + 2cos y** в диапазоне аргументов [-3; 3] с шагом h = 0.05.

7. Выполнить построение каркаса поверхности и самой поверхности. Исходными данными является матрица (5x5) из случайных чисел, равномерно распределенных в диапазоне от 0 до 1.

Лабораторная работа №3
Матричные действия над матрицами. Операции с полиномами.

Целью работы является ознакомление с матричными действиями над матрицами в системе Matlab, получение навыков для выполнения операций с полиномами в системе Matlab.

Задание на лабораторную работу

1) Провести операции сложения, вычитания, перемножения произвольных матриц размером (4x4), умножения матрицы на целое положительное число, транспонирования матрицы.

2) Найти обратную матрицу для произвольной матрицы

3) Используя функцию обращения матрицы и функцию деления матриц решить систему линейных уравнений:

$$\begin{cases} 5x_1 + 2x_2 + 15x_3 + 10x_4 = 22 \\ x_1 - x_2 + 3x_3 - 5x_4 = 4 \\ -x_1 + 8x_2 + x_3 = 7 \\ 6x_1 + 5x_2 + 4x_3 + 3x_4 = 2 \end{cases}$$

4) Найти корни полинома $P(x) = 2x^6+3x^5+4x^4+12x^3+5x^2+6x+5$

5)По заданному вектору корней полинома (R = [2; 4; 8; 16; 32]) найти его коэффициенты.

6) Вычислить значение полинома из задания 4 при аргументе равном 5.

7) Вычислить производную от полинома P(x).

Лабораторная работа №4.
Алгоритмы и технологии вычисления интегралов

Целью работы является ознакомление с технологиями вычисления интегралов в системе Matlab.

Задание на лабораторную работу

1) Провести интегрирование с накоплением (шаг интегрирования равен 0,5) для интеграла $\int_3^9 x * e^x - \ln(x) + 6$

2) Вычислить значение интеграла (интегрирование с накоплением) от функции представленной в виде вектора корней полинома: $P(x) = x^5+8x^4+31x^3+80x^2+94x+20$

3) Вычислить с помощью метода трапеции (шаг интегрирования равен 1) значение интеграла

$$\int_1^5 \frac{(15e^x + \ln x + 1)}{5x} dx$$

4) Подынтегральная функция имеет вид:

$$f(x) = -e^x + 8x^4 + 3 \operatorname{ctg} x + 1.$$

Вычислить методом Симпсона значение интеграла от f(x) с точностью 10^{-5}. Пределы интегрирования [1; 10].

5) Вычислить методом парабол значение двойного интеграла от функции z = ln(x)+ln(y). Пределы интегрирования по 1 переменной [1, 5], а по внешней переменной [2; 4].

6) С помощью аналитического метода найти значение неопределенного интеграла $\int \frac{x^2}{a+bx^3-cd}\,dx$

7) С помощью аналитического метода вычислить значение определенного интеграла $\int_1^3 \frac{x}{c+dx^2}\,dx$

8) Вычислить интеграл $\iiint \frac{\ln x + e^x}{3x^2}$

Лабораторная работа № 5
Режим программирования в Matlab

Целью работы является ознакомление с применением команд управления потоками в системе Matlab.

Задание на лабораторную работу:

1) Имеются 3 переменные a, b и i. Переменная b=15, переменная a=i/2 и переменная i=1. На каждом шаге i увеличивается на 1. Определить число шагов, за которое, **a** достигнет большего, чем **b** значения.

2) Имеются 2 переменные n и m. Переменная n может принимать одно из двух значений 0 (m=n) или 1 (m=n+n/2). Используя оператор переключения для переменной n, определить значение переменной m в каждом из этих случаев.

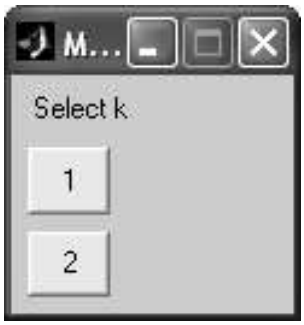
3) Дана матрица размером n x m. Произвести суммирование всех элементов матрицы кроме элементов последнего столбца и последней строки (используя вложенные циклы).

4) Создать функцию зависимости y от k и t в виде:

Если k=1, то y=(k*t)/2

Если k=2, то y=0

Создать меню выбора значения переменной k, которое может принимать значения 1 или 2.



5) Создать файл-функцию для расчета факториала числа 8.

Литература

1. В.П. Дьяконов. MATLAB 6.0/6.1/6.5/6.5 + SP1 + Simulink 4/5. Обработка сигналов и изображений. М.: СОЛОН-Пресс, 2004. - 592 с.
2. В.Потемкин. Вычисления в среде MATLAB. Диалог-МИФИ. 2004.
3. Кривилев А. Основы компьютерной математики с использованием системы MATLAB. Лекс-Книга, 2005.
4. В.П.Дьяконов. MATLAB 6/6.1/6.5 + Simulink 4/5. Основы применения. Полное руководство пользователя . СОЛОН-Пресс, 2004.
5. Н.Мартынов. Введение в MATLAB 6. Кудиц-образ. 2002.