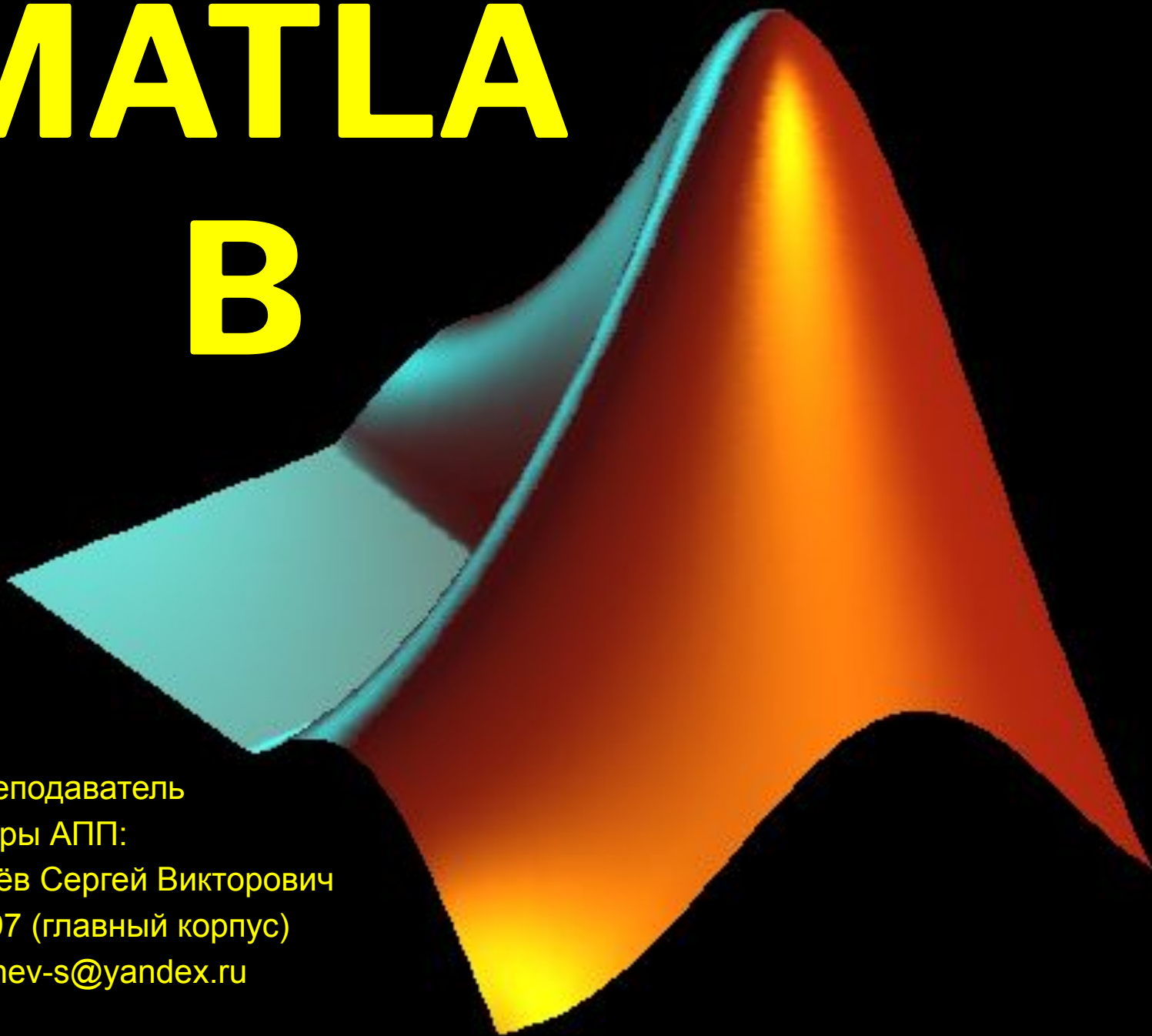


MATLAB

B

Ст. преподаватель
кафедры АПП:
Носачёв Сергей Викторович
Каб.297 (главный корпус)
nosachev-s@yandex.ru



Содержание

- Введение
- Основы Основы Matlab
- Вычисления в Вычисления в Matlab
- Функции для работы с массивами
- Графические возможности
Графические возможности Matlab
- Программирование в
Программирование в Matlab
- Аналитические вычисления в
Аналитические вычисления в Matlab

Введение

Matlab (MATrix LABoratory) –

ЭТО

- ⦿ математические вычисления
- ⦿ создание алгоритмов
- ⦿ моделирование
- ⦿ анализ, обработка и визуализация данных
- ⦿ научная и инженерная графика
- ⦿ разработка приложений с GUI
- ⦿ огромное количество прикладных пакетов

Пакеты, встроенные в Matlab

- 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

ОСНОВНЫЕ ЧАСТИ ПАКЕТА MATLAB:

1. Язык Matlab
2. Среда Matlab
3. Управляемая графика
4. Библиотека математических функций
5. Программный интерфейс

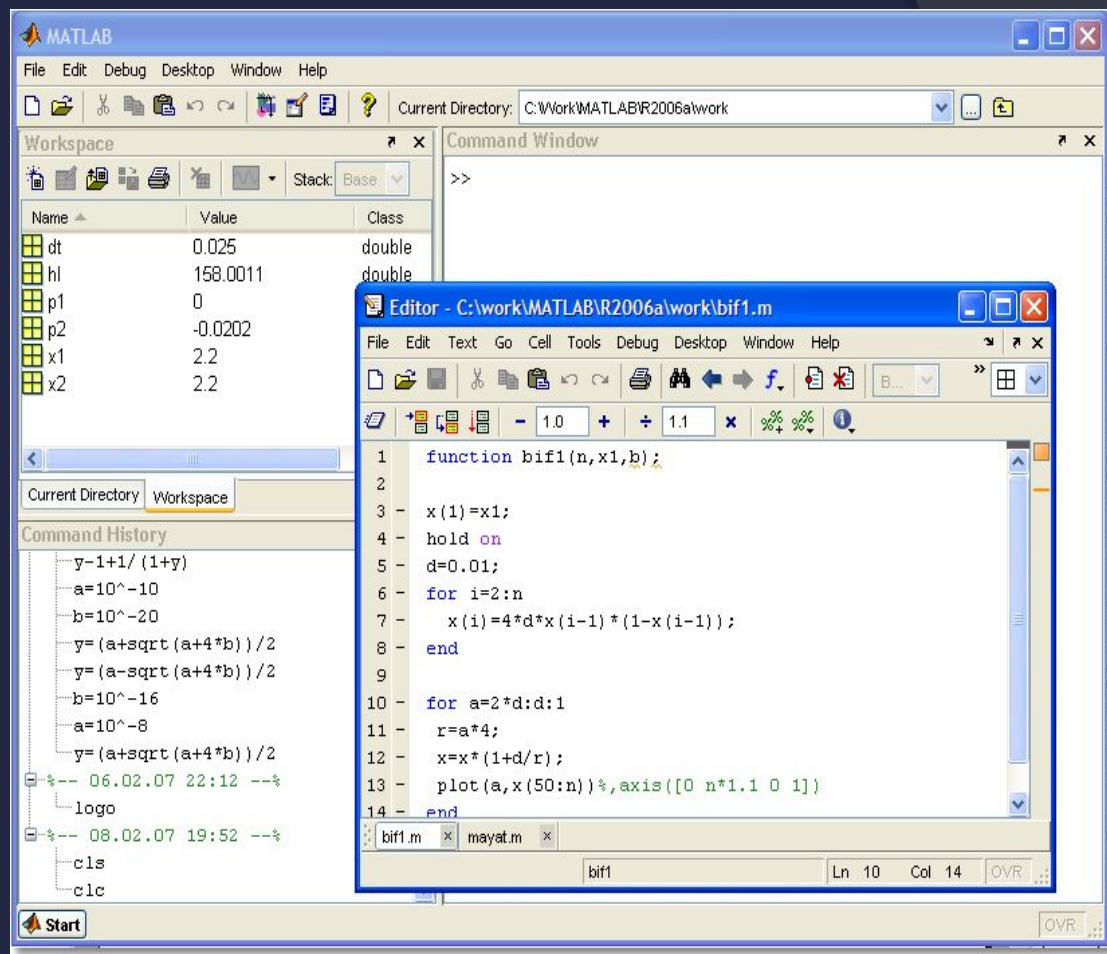
Язык Matlab

- Си- и Паскаль-подобный объектно-ориентированный
- Огромный набор встроенных функций
- Расширяемый пользователем

```
1 - clear;
2 - x1=2.2;
3 - p1=0.0;
4 - dt=0.025;
5 - axis([-pi pi -pi pi]);
6 - hl=line(x1,p1);
7 - set(hl,'EraseMode','none','LineStyle',':','Color','r');
8 - grid on;
9 - pause;
10 - while 1
11 -     x2=x1+p1*dt;
12 -     p2=p1-sin(x2)*dt;
13 -     if x2> pi
14 -         x2=x2-2*pi;
15 -     end;
16 -     if x2< -pi
17 -         x2=x2+2*pi;
18 -     end;
19 -     set(hl,'XData',x2,'YData',p2);
20 -     x1=x2; p1=p2;
21 - end;
```

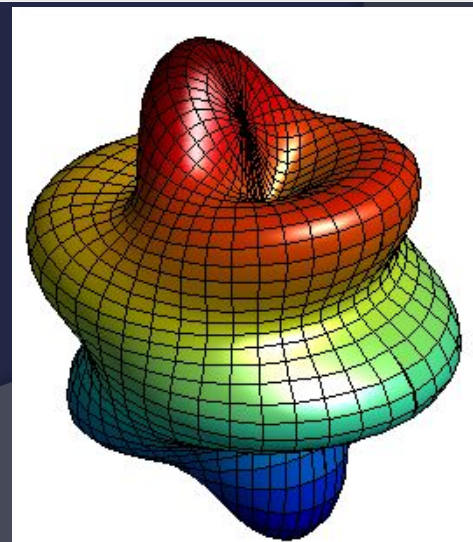
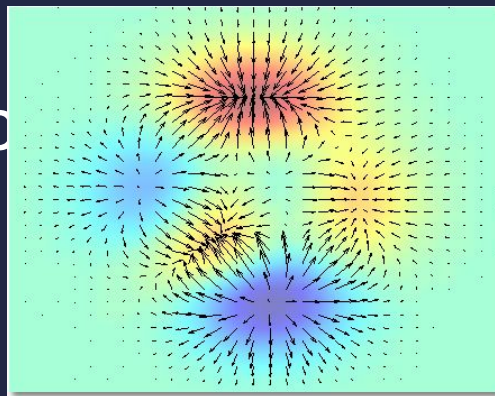
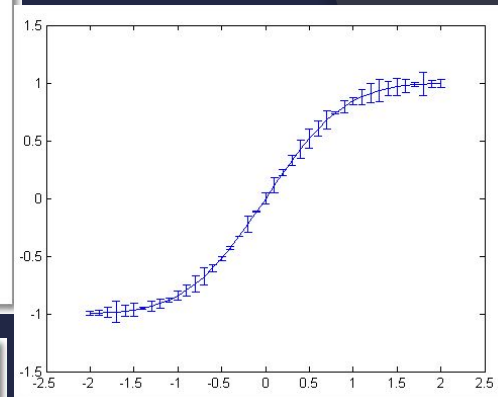
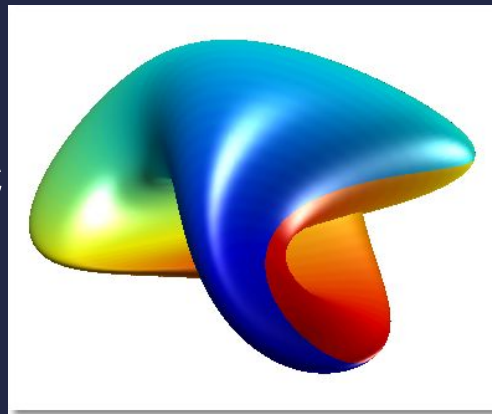
Среда Matlab

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



Управляемая графика

- ⦿ Команды высокого уровня для работы с 2D- и 3D-графикой
- ⦿ Анимация
- ⦿ Команды низкого уровня для работы с графикой



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.



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 (1×1)
- Использование матриц
 - существенно облегчает программирование
 - делает запись формул краткой и наглядной
- В дальнейшем изложении предполагается знакомство с матричной алгеброй и основами программирования

Числа

- Основной базовый тип для матриц
- Хранятся в формате long (double)
 - стандарт плавающей точки IEEE
- Интервал приблизительно от $10E-308$ до $10E+308$
- Комплексные числа строятся с применением суффиксов i или j (мнимая единица): $2.4e7+3.005i$

Другие типы

- Строки
- Массивы структур (записей)
- Массивы ячеек
 - позволяют объединять в массиве элементы разной природы
- Объекты

Переменные и выражения

- Переменные определяются пользователем при помощи оператора присваивания: $x=5$
- В левой части – имя переменной
 - заглавные и строчные буквы различаются
- 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:

```
Command Window
>> a=2*3

a =

     6

>> b=a/7;
>> b

b =

 0.8571

>> |
```

Операторы

- При составлении выражений могут быть использованы операторы:
 - + сложение
 - вычитание
 - * умножение
 - / деление
 - ^ возведение в степень
- Приоритет операций обычный. Изменяется при помощи круглых скобок

Операции отношения

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

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

res =

     2
```

Приоритет ниже, чем у арифметических операций

Логические операции

& И
| ИЛИ
~ НЕ

```
Command Window
>> a = 4; b = 5; c = 6;
>> is_treug = (a < b+c) & (b < a+c) & (c < a+b)

is_treug =

     1
```

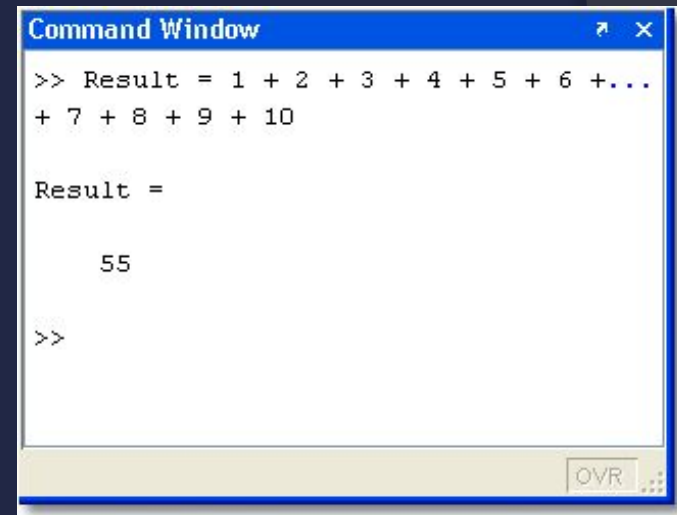
0 – ложь (false)

1 – истина (true)

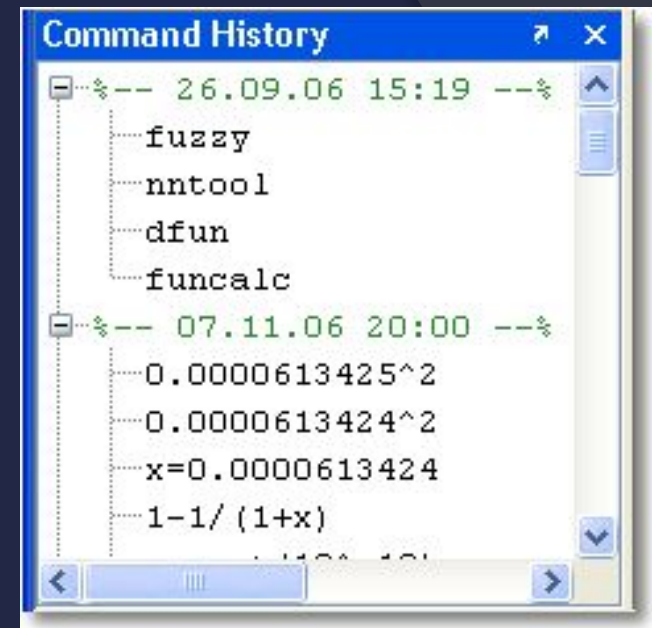
Приоритет ниже, чем у арифметических операций и операций отношения

Командная строка

- 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`

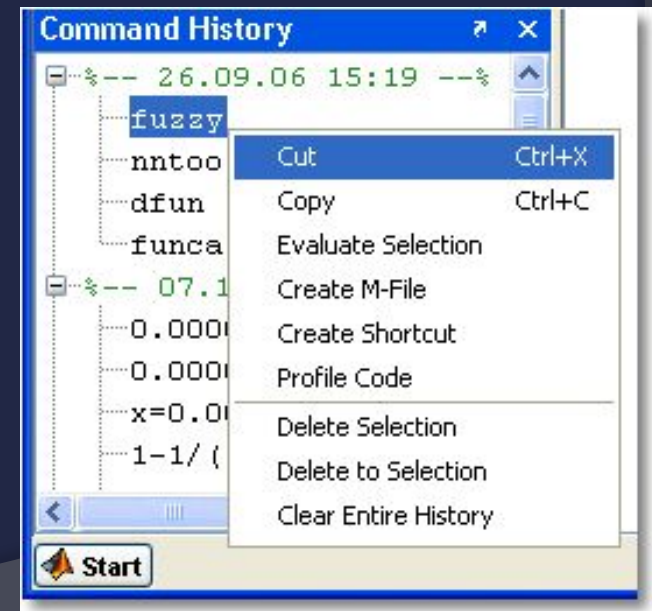


- Работа с КС упрощается благодаря окну Command History (меню Desktop)
- Здесь хранится сессионная запись всех введённых команд
- Их можно скопировать, выполнить и т. п. (см. контекстное меню)



Command History

```
-- 26.09.06 15:19 --  
fuzzy  
nntool  
dfun  
funca  
-- 07.11.06 20:00 --  
0.0000613425^2  
0.0000613424^2  
x=0.0000613424  
1-1/(1+x)
```



Command History

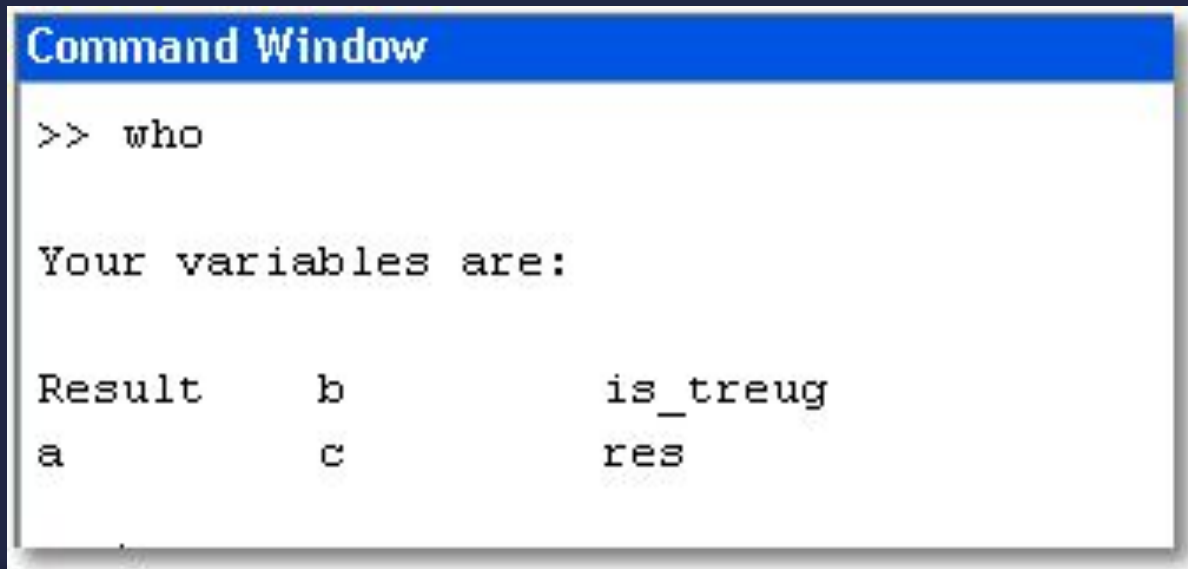
```
-- 26.09.06 15:19 --  
fuzzy  
nntoo  
dfun  
funca  
-- 07.1  
0.0000  
0.0000  
x=0.0  
1-1/(
```

Context Menu:

- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Evaluate Selection
- Create M-File
- Create Shortcut
- Profile Code
- Delete Selection
- Delete to Selection
- Clear Entire History

Рабочее пространство (Workspace)

- Все переменные хранятся в РП
 - порой это отнимает много места
- Просмотреть список существующих в РП переменных можно командой `who`:



```
Command Window
>> who

Your variables are:

Result      b      is_treug
a           c      res
```

- Более подробную информацию о переменных РП можно вывести командой `whos`:

```
Command Window
>> whos

Name          Size          Bytes  Class

Result        1x1           8     double array
a             1x1           8     double array
b             1x1           8     double array
c             1x1           8     double array
is_treug      1x1           1     logical array
res           1x1           8     double array

Grand total is 6 elements using 41 bytes
```

- После закрытия сеанса работы MATLABа все переменные, вычисленные в течение сеанса, теряются. Однако их можно сохранить для последующего использования в иных сеансах, сохранив содержимое РП в файле на диске
 - командой меню: File \ Save Workspace As...
 - командой Matlab: `save`

Команда `save`

- `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
 - `help <имя команды>`
 - или F1

Команда load

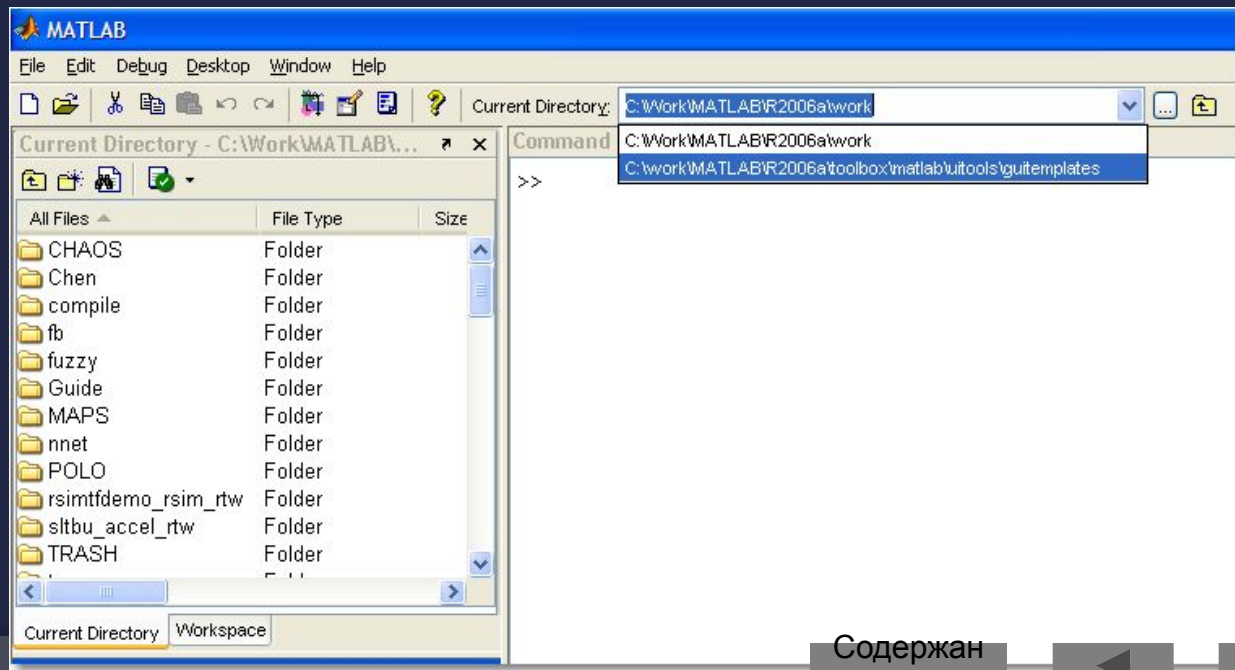
- Служит для загрузки ранее сохранённых данных
- `load` – загружает все переменные из файла *matlab.mat*
- `load filename` – загружает все переменные из файла *filename*
- `load filename x y z` – загружает переменные *x*, *y*, *z* из файла *filename*
- `load -ASCII filename x y z` – загружает переменные *x*, *y*, *z* из текстового файла *filename*
`load('filename', 'a', 'b', '-ASCII')` – процедурная форма вызова команды

Команда `clear`

- ⦿ Служит для удаления переменных из РП
- ⦿ `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`.

Рабочий каталог

- 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



Сохранение рабочей сессии

- `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
- `diary` – переключается между режимами `on/off`, если лог уже ведётся

ВЫЧИСЛЕНИЯ В MATLAB



Элементарные функции

Тригонометрические

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

Элементарные функции

Экспоненциальные

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

Элементарные функции

Округление и остатки

- ⦿ `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 $n = \text{floor}(x/y)$)
- ⦿ `rem(x,y)` – остаток от деления x на y с учётом знака ($x - n*y$, где $n = \text{fix}(x/y)$)

Элементарные функции

Комплексные числа

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

Элементарные функции

- Просмотреть полный список элементарных функций можно командой
 - `help elfun`

Константы

- ⦿ π – число π
- ⦿ Inf – бесконечность
- ⦿ $-\text{Inf}$ – минус бесконечность
- ⦿ NaN (Not a Number) – нечисловое значение

```
Command Window
>> 5/0
Warning: Divide by zero.

ans =

    Inf

>> -5/0
Warning: Divide by zero.

ans =

   -Inf

>> 0/0
Warning: Divide by zero.

ans =

   NaN
```

Одномерные массивы

- Задание массива:
 - $a = [-3\ 4\ 2];$
 - $a = [-3, 4, 2];$
- Диапазоны:
 - $b = -3: 2$ ($b = -3\ -2\ -1\ 0\ 1\ 2$)
 - $b = -3:2:5$ ($b = -3\ -1\ 1\ 3\ 5$)
- Доступ к элементу:
 - $a(3)$ (будет равно 2)
- Изменение элемента:
 - $a(3) = 1$
- Количество элементов в массиве: $\text{length}(a)$ (будет равно 3)
- Нумерация элементов начинается с 1
- Добавление элементов в массив
 - $a(4) = 5;$
 - $a = [a\ 5]$
- Конкатенация массивов:
 - $c = [a\ b]$
- Удаление массива (превращение в пустой массив)
 - $a = []$

Двумерные массивы

- Задание массива:

- $a = [1 \ 2; 3 \ 4; 5 \ 6];$

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

a =

     1     2
     3     4
     5     6
```

- Доступ к элементу:

```
>> a(3,1)

ans =

     5

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

Векторы-столбцы и векторы-строки

- Любая строка и столбец матрицы – это вектор
- Векторы, расположенные вдоль строк – векторы-строки (размер $1 \times n$)
- Векторы, расположенные вдоль столбцов – векторы-столбцы (размер $n \times 1$)
- Задание вектора-столбца:
- К векторам любого типа применима функция `length`

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

c =

     1
     2
     3

>> c = [1 2 3]'

c =

     1
     2
     3
```

Размерность и размер матриц

- Размерность массива определяется функцией `ndims(A)`
- Размер массива – функцией `size(A)`

```
>> 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
```

Конкатенация

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

- При несовпадении размерностей получаем сообщение об ошибке

```
>> c = [c; a]
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

Диапазоны

- Можно использовать как для задания значений векторов, так и для задания диапазонов индексации
- Рассмотрим другие примеры

```
>> 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
```

```
>> 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(3)
```

```
a =
```

```
     8     1     6
     3     5     7
     4     9     2
```

```
>> a(:)
```

```
ans =
```

```
     8
     3
     4
     1
     5
     9
     6
     7
     2
```

- ⦿ Для обращения к последнему элементу любой размерности можно использовать служебное слово `end`:

```
>> 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

>> B = A(:,3:end)

B =

     1     8    15
     7    14    16
    13    20    22
    19    21     3
    25     2     9
```

Удаление строк и столбцов

```
>> 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,:) = [ ]
```

```
a =
```

17	24	1	8	15
10	12	19	21	3
11	18	25	2	9

```
>> 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(:,3:4) = [ ]
```

```
a =
```

17	24	15
23	5	16
4	6	22
10	12	3
11	18	9

Перестановка элементов

```
>> b = 1:3:11

b =

     1     4     7    10

>> b([4 2 1 3])

ans =

    10     4     1     7
```

```
>> 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 = a(:, [3 5 2 4 1])

a =

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

ФУНКЦИИ ДЛЯ РАБОТЫ С МАССИВАМИ В MATLAB



Создание матриц специального вида

- Для работы с матрицами удобно пользоваться следующими функциями
 - `ones` – формирование массива из единиц
 - `zeros` – формирование массива из нулей
 - `eye` – формирование единичной матрицы
 - `rand` – формирование матрицы из числе, равномерно распределённых на отрезке $[0, 1]$
 - `randn` – формирование матрицы из чисел, нормально распределённых с математическим ожиданием 0.
 - `magic` – формирование магического квадрата
 - `pascal` – формирование квадрата Паскаля
 - `diag` – диагональная матрица
 - и др.

МАТРИЦЫ СПЕЦИАЛЬНОГО ВИДА

- Рассмотрим основной синтаксис на примере функции создания единичной матрицы (`eye`)
- `eye(m)` – создание единичной матрицы размера $[m, m]$
- `eye(m, n)` – создание единичной матрицы размера $[m, n]$
 - «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

```
>> 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
```

```
>> 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`: работа с диагональными матрицами
 - у которых ненулевые элементы расположены на диагоналях
- ⊙ Синтаксис:
 - $X = \text{diag}(v)$ – на главной диагонали матрицы X расположены элементы вектора v
 - $X = \text{diag}(v, k)$ – на k -ой диагонали матрицы X расположены элементы вектора v (по умолчанию $k=0$)
 - $v = \text{diag}(X, k)$ – 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    0
    0    0    1    2    0    0
    0    0    0    1    3    0
    0    0    0    0    0    4
    0    0    0    0    0    0
    0    0    0    0    0    0
```

```
>> B = diag(v, -1)
```

```
B =
```

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

```
>> M = magic(6)
```

```
M =
```

```
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
```

```
>> u = diag(M, 1)
```

```
u =
```

```
    1
    7
   22
   10
   16
```

```
>> m = 3
```

```
m =
```

```
3
```

```
>> A = diag(-m:m)+diag(ones(2*m,1),1)+diag(ones(2*m,1),-1)
```

```
A =
```

```
-3    1    0    0    0    0    0
 1   -2    1    0    0    0    0
 0    1   -1    1    0    0    0
 0    0    1    0    1    0    0
 0    0    0    1    1    1    0
 0    0    0    0    1    2    1
 0    0    0    0    0    1    3
```

ВЫЧИСЛЕНИЯ С ЭЛЕМЕНТАМИ МАССИВОВ

- Простейшие операции над элементами массивов:
 - `sum`: сумма элементов
 - `prod`: произведение элементов
 - `cumsum`: кумулятивная сумма элементов
 - `cumprod`: кумулятивное произведение элементов
 - `max`: нахождение максимального элемента
 - `min`: нахождение минимального элемента
 - `sort`: сортировка элементов

- ⦿ Рассмотрим работу некоторых из этих функций на примере `sum`
- ⦿ Для векторов эта функция возвращает сумму элементов
- ⦿ Для массивов – сумму элементов по каждому из столбцов
 - результат – вектор-строка
- ⦿ Остальные функции работают по этому же принципу

```
>> A = round(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:

```
>> 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
```

ЛОГИЧЕСКИЕ ФУНКЦИИ

- `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
```

- 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
```

```
>> v=round(20*rand(1,8)-5)

v =

     5    15     2     6    -1     5     3     8

>> v>7

ans =

     0     1     0     0     0     0     0     1

>> v(v>7)

ans =

    15     8

>> v(v>5 & v<=8)

ans =

     6     8
```

Поиск в массиве

- ◎ `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
```

- Пример применения команды `find` к матрицам:

```
>> a = magic(3)

a =

     8     1     6
     3     5     7
     4     9     2

>> k = find(a>5);
>> [w, i] = find(a>5);
>> [k w i]

ans =

     1     1     1
     6     3     2
     7     1     3
     8     2     3
```

Математические матричные операции

- `det` – вычисление определителя квадратной матрицы

```
>> format long
>> A = round(100*rand(6)-40)

A =

    -21    -7    22    57    40   -12
     5    48    29    42    27   -33
   -39     8    11    -8   -39     8
    -9    16    31    19    16    58
    48    22    12   -27     5    52
    44    26    21   -15    50    16

>> det(A)

ans =

-5.225668154000000e+009
```


МАТРИЧНЫЕ И ПОЭЛЕМЕНТНЫЕ ОПЕРАЦИИ

- ⦿ При работе с матрицами можно использовать два вида операторов:
 - *матричные*: производят действия по правилам матричной алгебры
 - *поэлементные*: производят действия над соответствующими элементами матриц
 - размеры матриц должны быть одинаковыми
 - от матричных операций отличаются точкой перед знаком 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»

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

```
a =
```

```
    1    2  
    3    4
```

```
>> b = [3 1; 2 0]
```

```
b =
```

```
    3    1  
    2    0
```

```
>> a*b
```

```
ans =
```

```
    7    1  
   17    3
```

```
>> a.*b
```

```
ans =
```

```
    3    2  
    6    0
```

- Такие операции часто используются, если нужно применить какую либо функцию ко всем элементам матрицы.

```
>> x = [1 2 3 4 5]

x =

     1     2     3     4     5

>> 1/x^2
??? Error using ==> mpower
Matrix must be square.

>> 1./x.^2

ans =

     1.0000     0.2500     0.1111     0.0625     0.0400

>> format rat
>> 1./x.^2

ans =

     1           1/4           1/9           1/16           1/25
```

- Некоторые операции по умолчанию считаются поэлементными:

```
>> a = magic(3)

a =

     8     1     6
     3     5     7
     4     9     2

>> a+1

ans =

     9     2     7
     4     6     8
     5    10     3

>> sin(a)

ans =

    0.9894    0.8415   -0.2794
    0.1411   -0.9589    0.6570
   -0.7568    0.4121    0.9093
```

Операции «деления» слева и справа

- Применяются для решения систем линейных уравнений (СЛУ)
- Деление слева (\backslash)
 - для квадратных матриц реализует метод Гаусса
 - для прямоугольных матриц – метод наименьших квадратов

```
>> A = [1 5 7;  
        3 2 -5];  
>> b = [4;7]  
  
b =  
  
     4  
     7  
  
>> x = A\b  
  
x =  
  
     0  
  1.7692  
 -0.6923  
  
>> x = inv(A)*b  
??? Error using ==> inv  
Matrix must be square.  
  
>> A*x-b  
  
ans =  
  
  1.0e-015 *  
  
 -0.8882  
  0.8882
```

ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ MATLAB



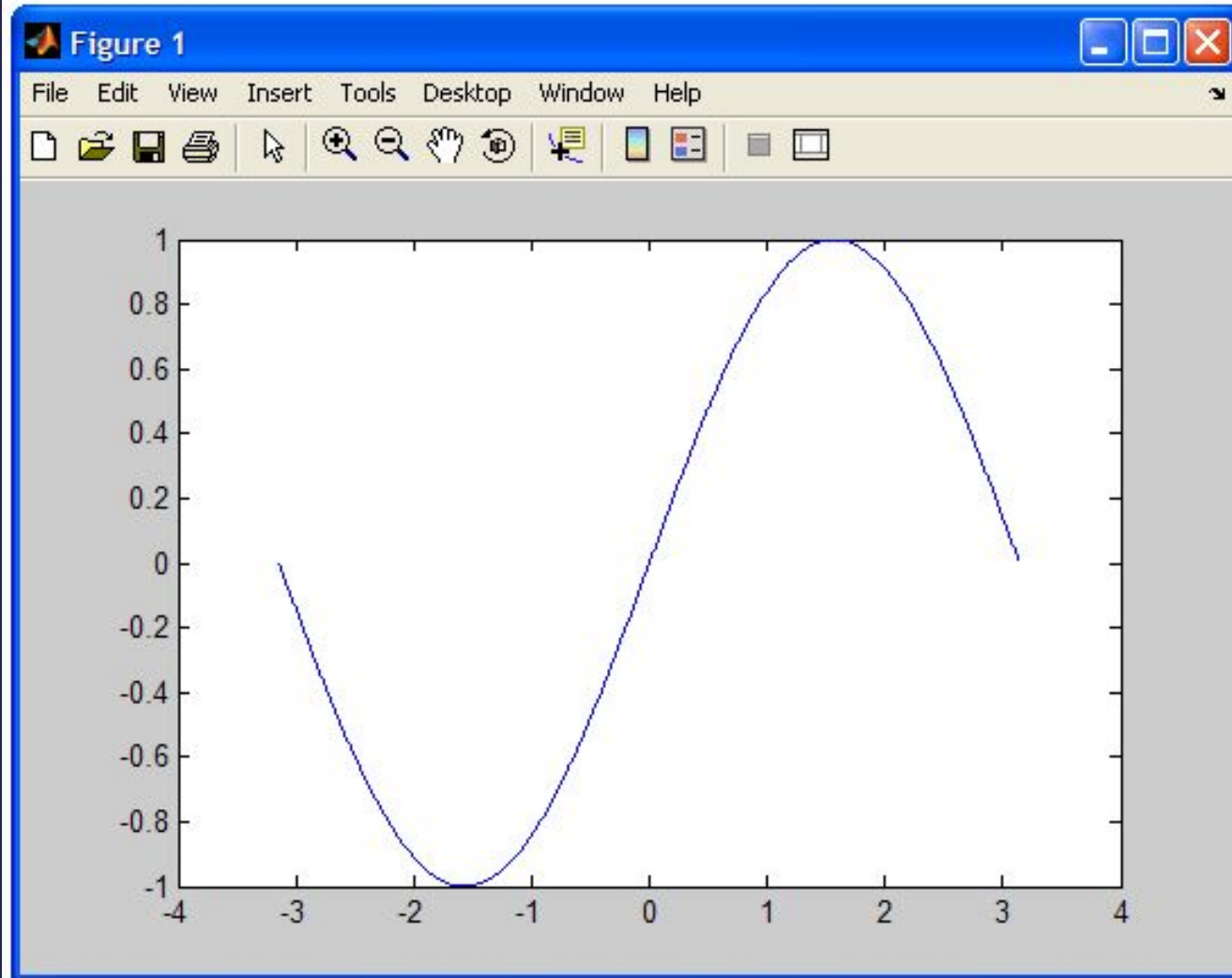
Графика в Matlab

- ⦿ Высокоуровневая
 - не требует от пользователя детальных знаний о работе графической подсистемы
- ⦿ Объектная
 - каждый объект на рисунке имеет свойства, которые можно менять
- ⦿ Управляемая (*handled*)
 - доступ к графическим объектам возможен как через инспектор объектов, так и при помощи встроенных функций (дескрипторная графика)

Двумерные (2D-) графики

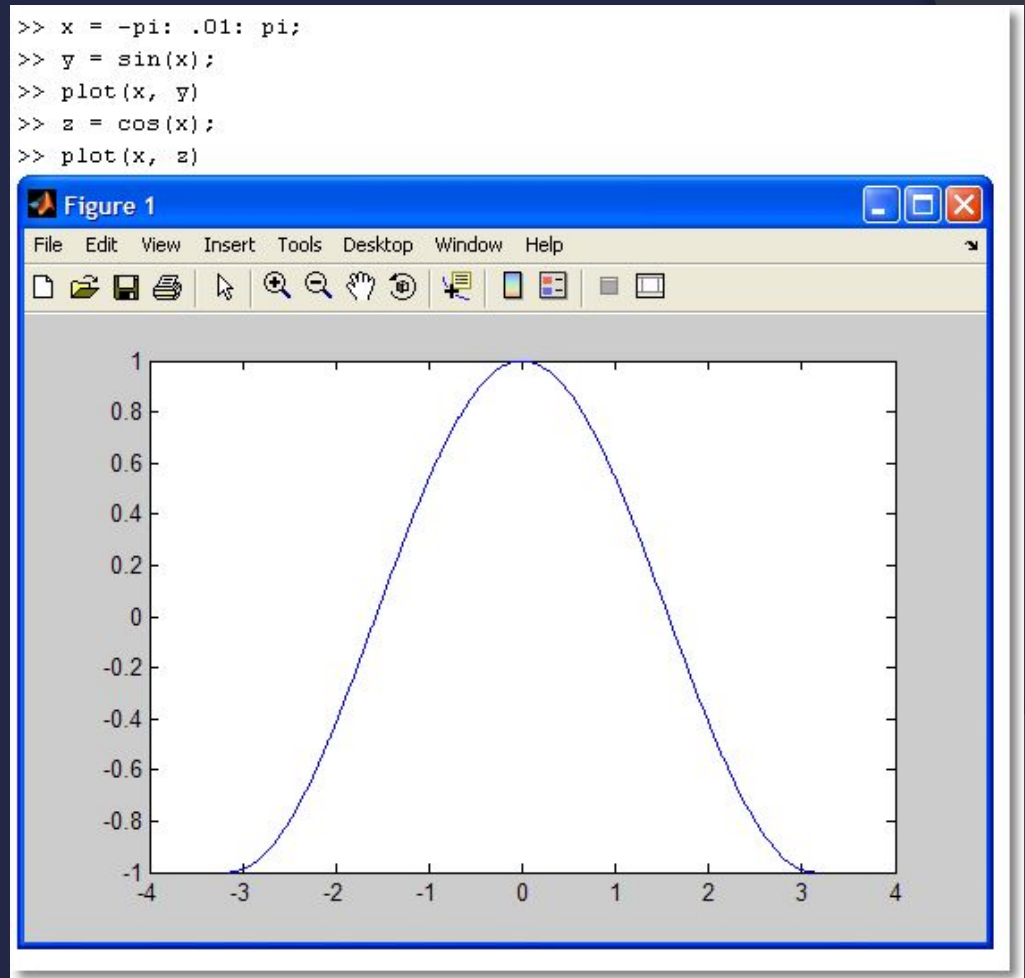
- Простейший способ построения 2D-графика:
 1. задать область построения (диапазон);
 2. вычислить значение функции на области построения
 3. построить график при помощи одной из встроенных функций Matlab

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



Построение второго графика

- Если сразу же построить другой график, то старый график будет удалён из графического окна

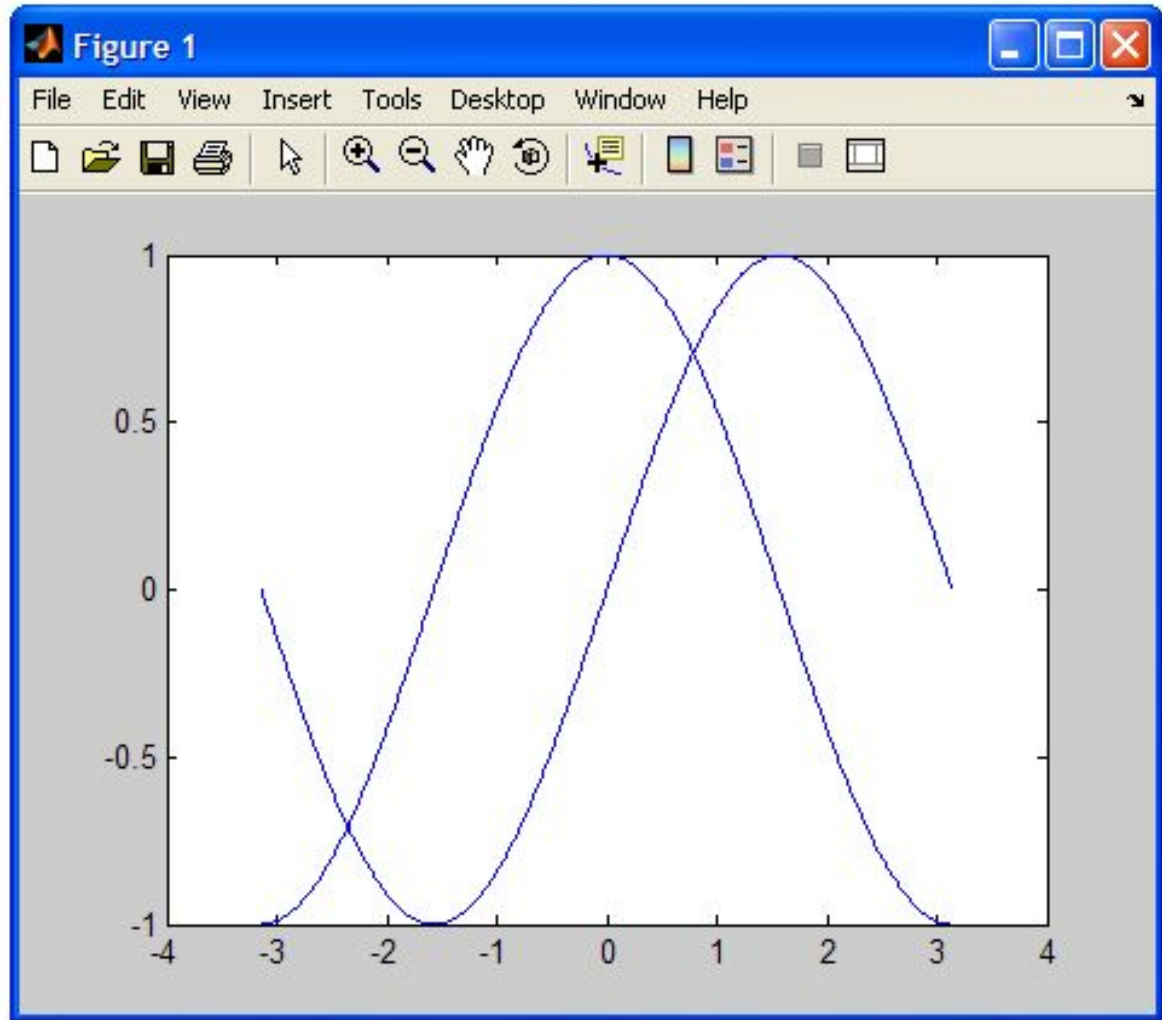


Построение двух графиков в одной системе координат

- ⦿ Два графика в одной СК можно построить следующими способами:
 1. «закрепить» графическое окно при помощи команды *hold on*
 2. применить одну команду *plot*

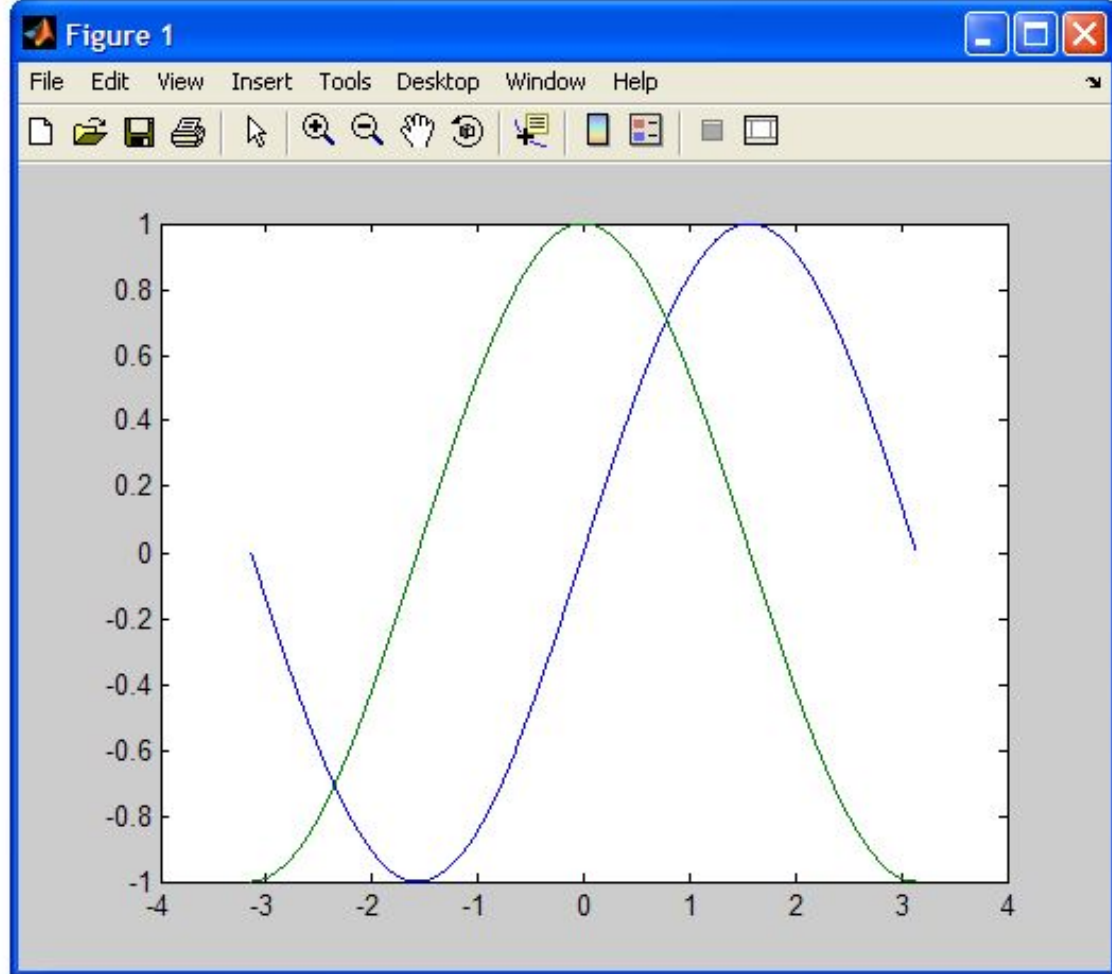
Пример закрепления графиков

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



Пример построения двух графиков одной командой

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



Дополнительные параметры команды *plot*

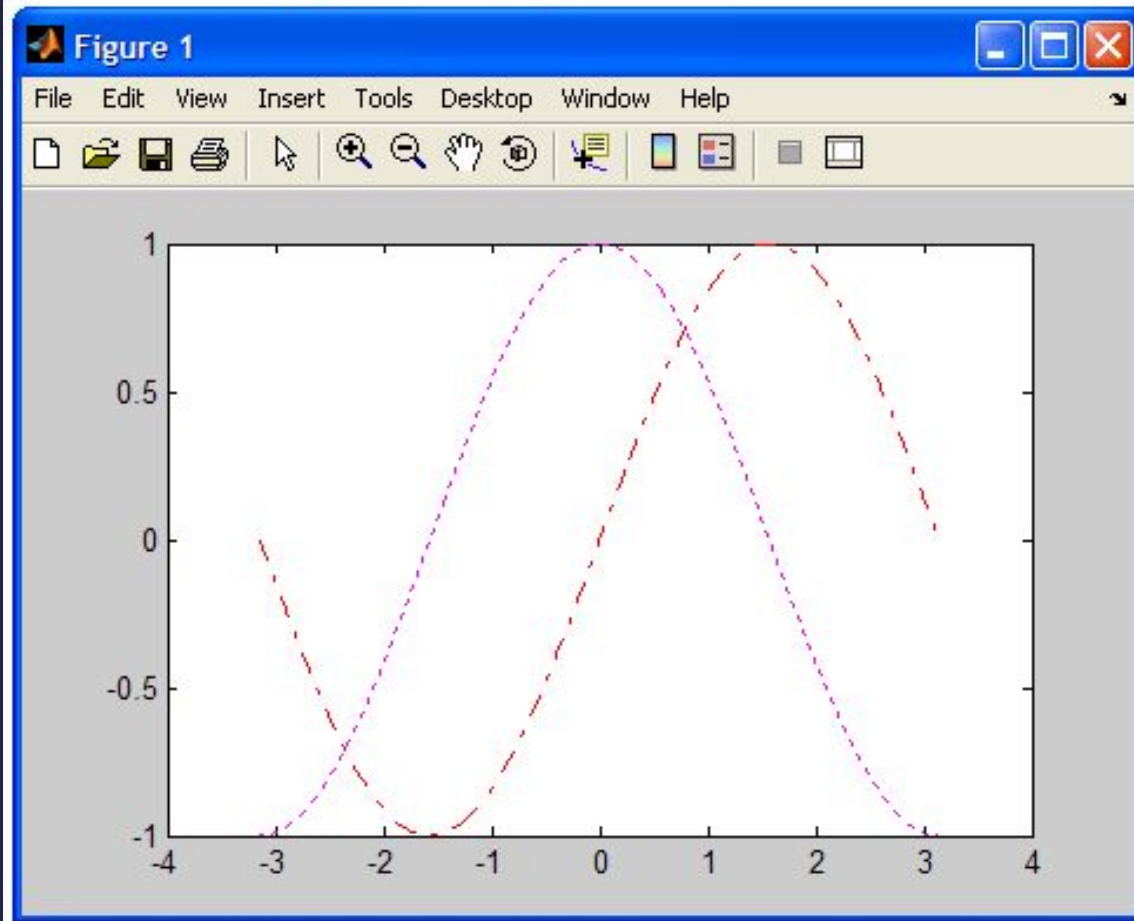
- В команде *plot* можно задать для каждого графика

- цвет линии тип маркера 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		

Пример команды plot

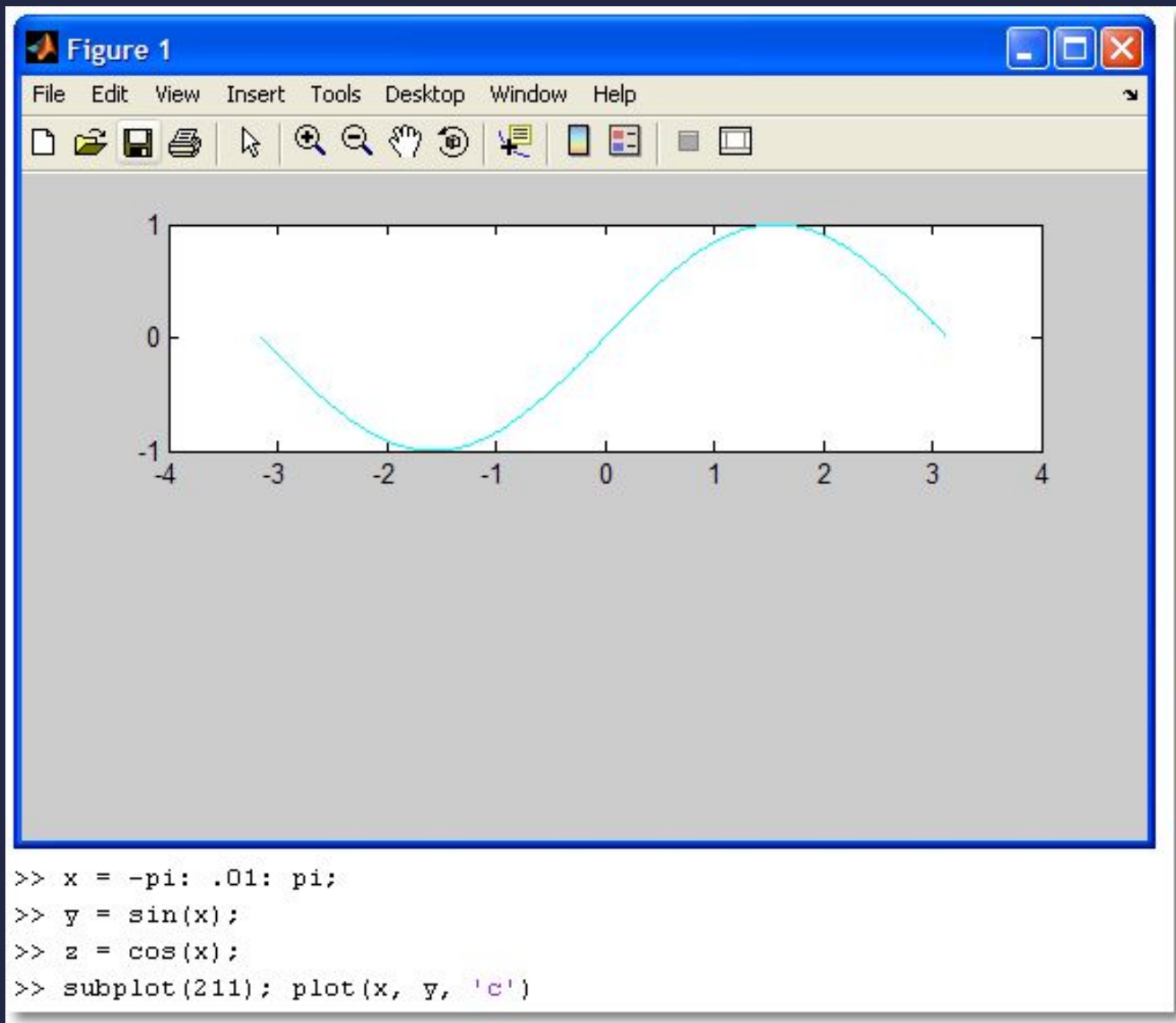
```
>> x = -pi: .01: pi;  
>> y = sin(x);  
>> z = cos(x);  
>> plot(x, y, 'r-.', x, z, 'm:')
```



Построение нескольких графиков в одном окне в разных СК

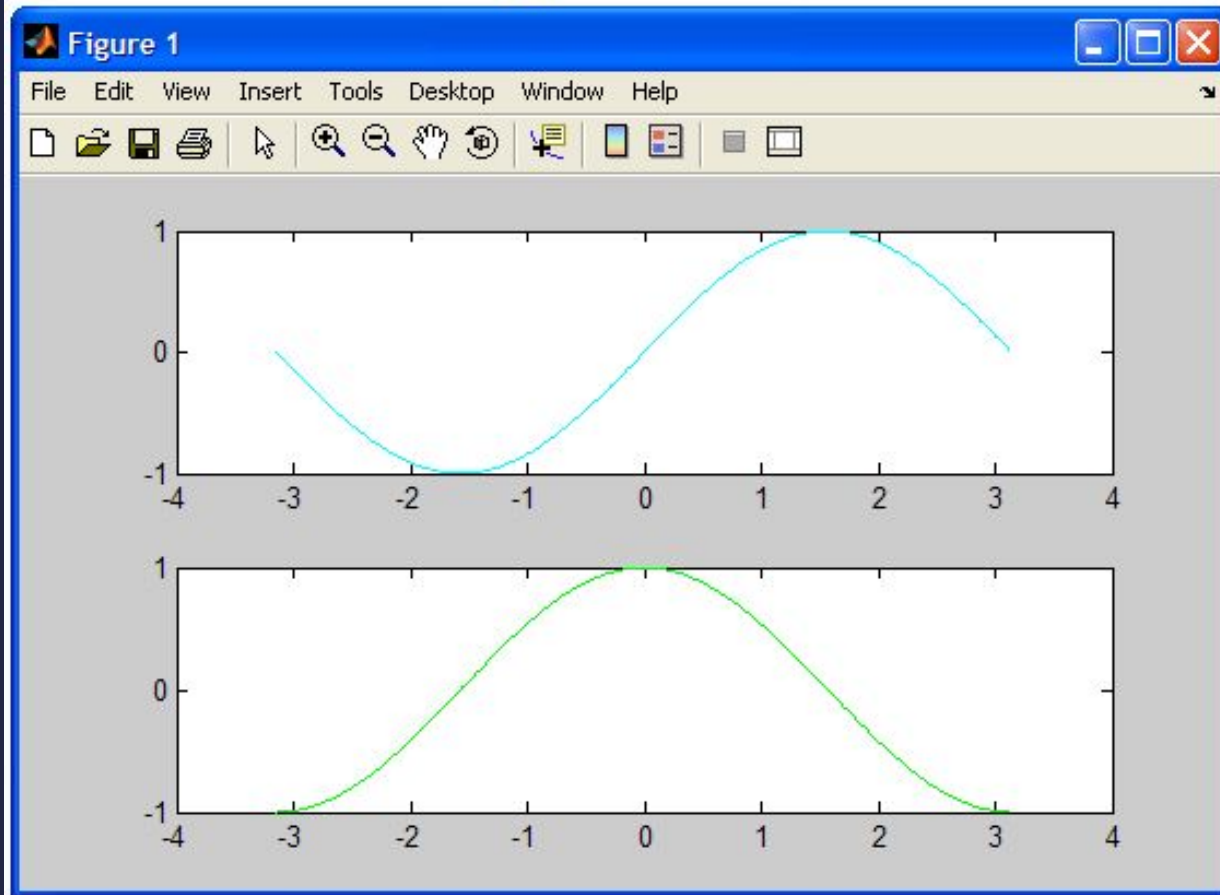
- Поверхность графического окна можно разделить на зоны, в каждой из которых выводить свой график
- Для этого служит команда *subplot*
- В качестве параметров ей передаётся трёхзначное целое число вида *mnk*
- *m* и *n* определяют количество графических «подокон» по горизонтали и вертикали
- *k* задаёт номер графического «подокна»
 - порядок нумерации – по строкам

Первый *subplot*

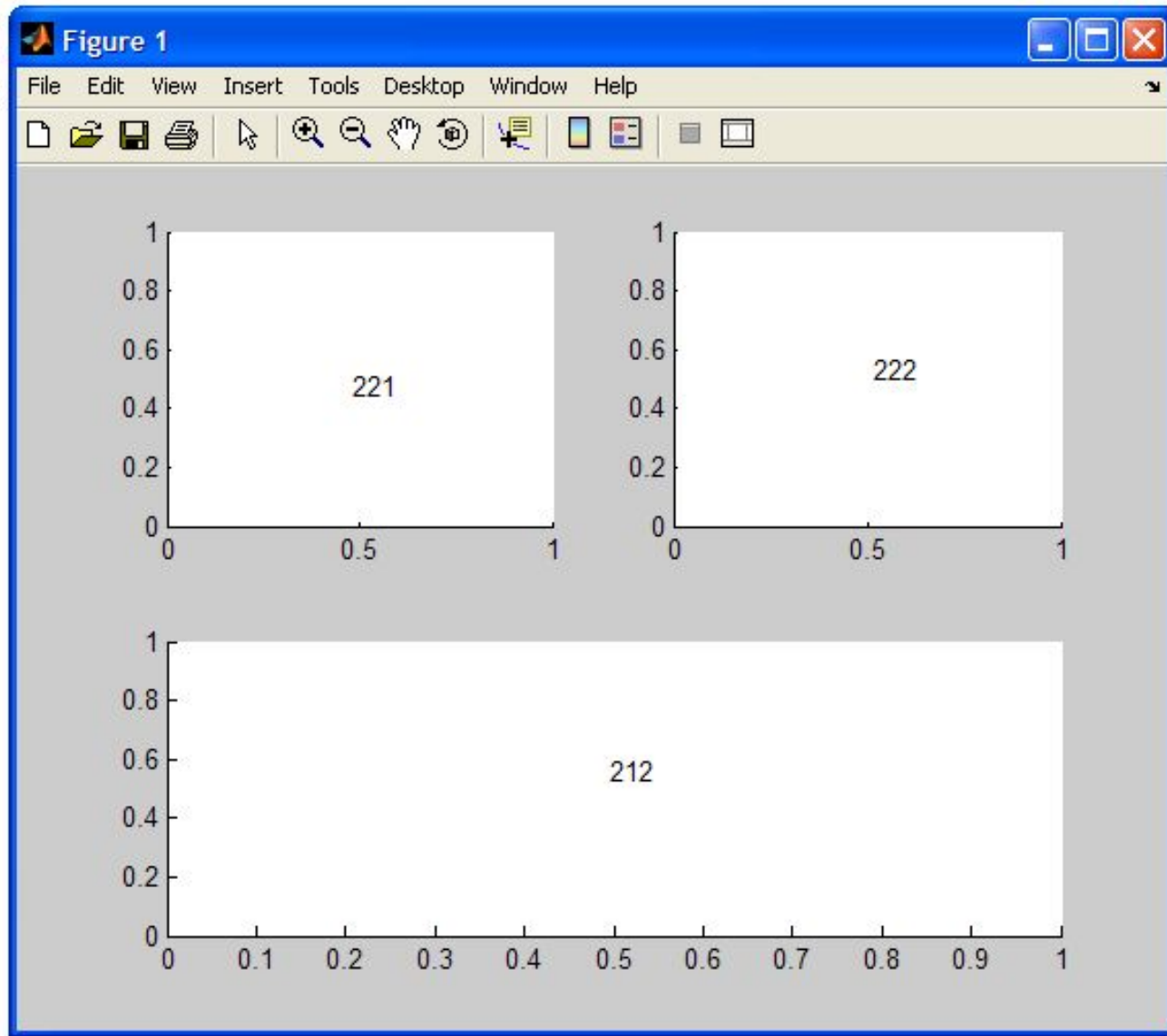


Второй *subplot*

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



```
>> subplot(221);  
>> subplot(222);  
>> subplot(212);  
>>  
>>
```



Построение графиков в разных графических окнах

- ⦿ Создать новое графическое окно можно командой *figure*
- ⦿ Команда *figure* создаёт графическое окно и возвращает указатель на него:
h = figure
- ⦿ Активизировать ранее созданное окно можно командой *figure(h)*

figure: пример использования 1

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

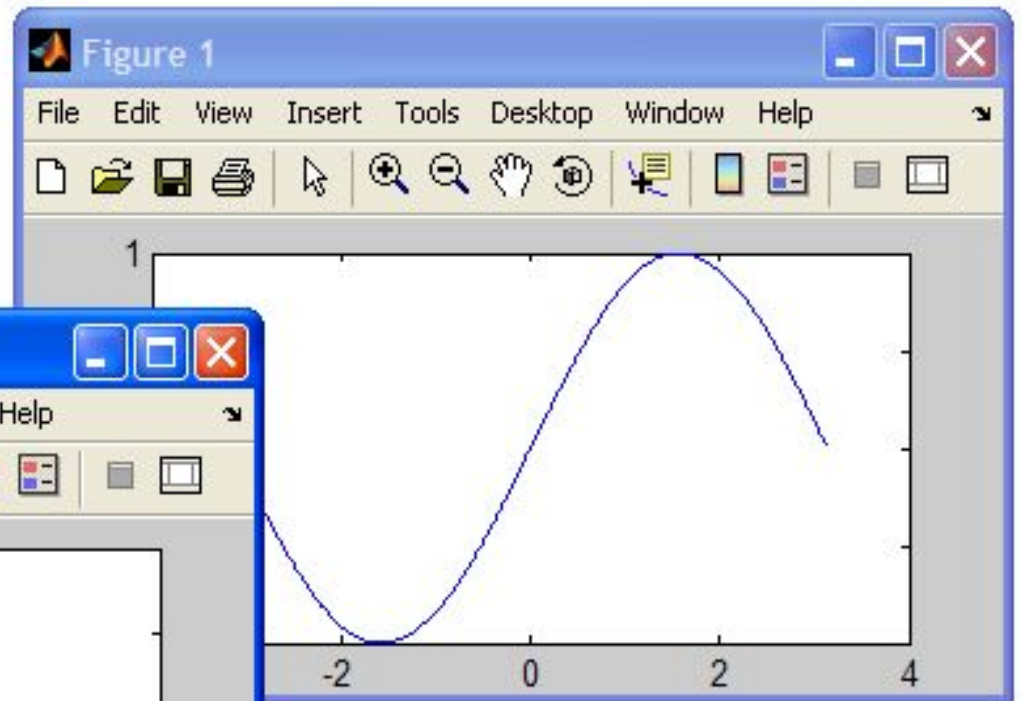
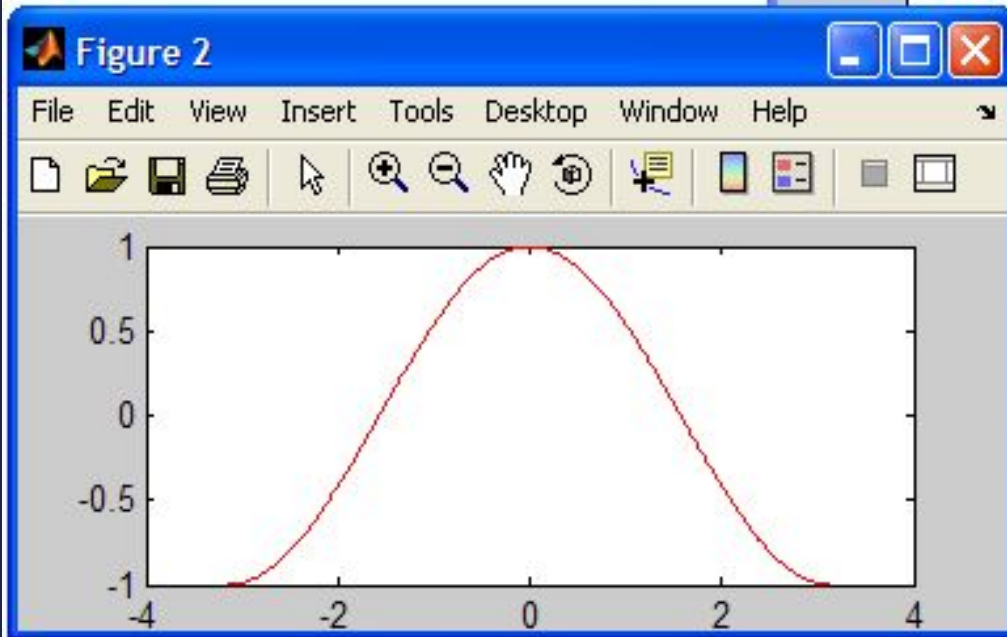
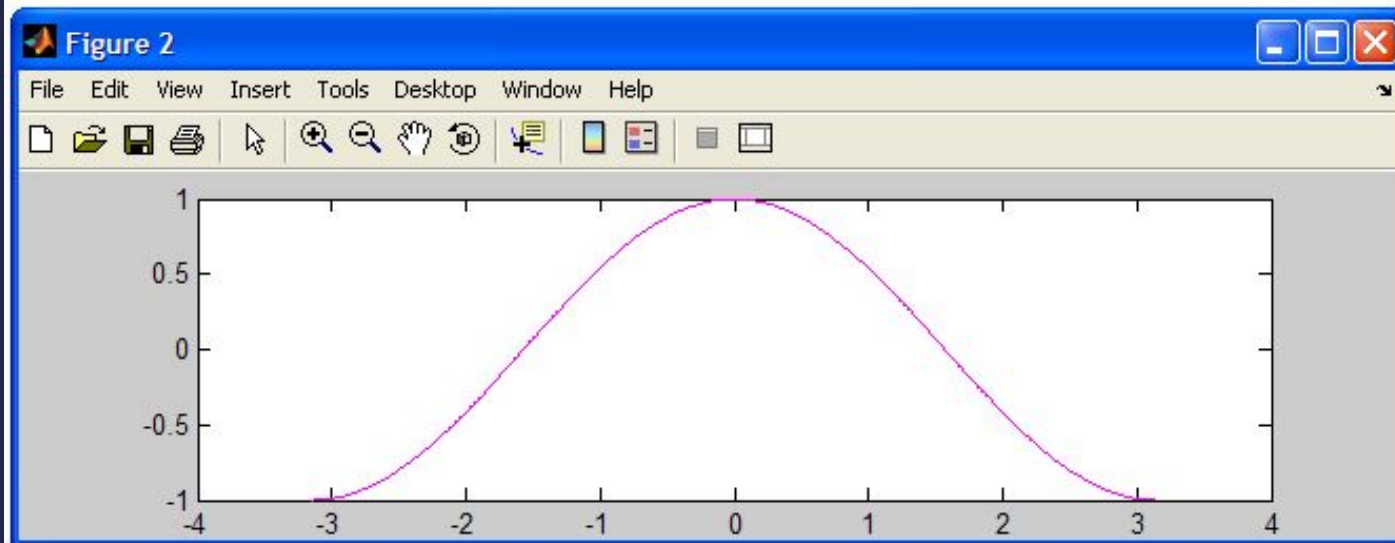
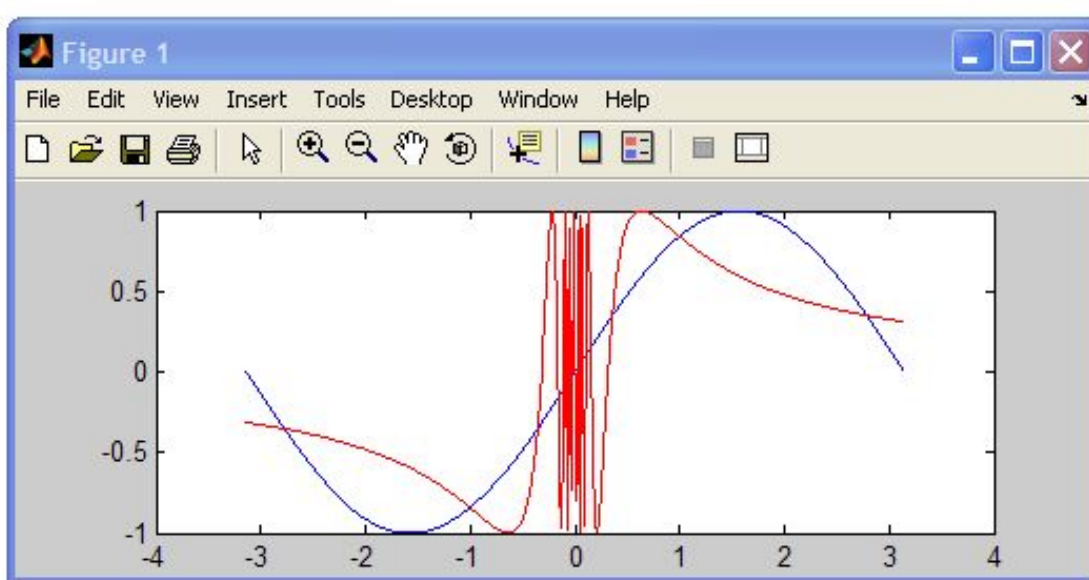


figure: пример использования 2

```
>> x = -pi: .01: pi;  
>> y = sin(x);  
>> z = cos(x);  
>> v = sin(1./x);  
>> f = figure  
  
f =  
  
    1  
  
>> plot(x, y)  
>> figure  
>> plot(x, z, 'm')  
>> figure(f)  
>> hold on  
>> plot(x, v, 'r')
```



Axis: управление масштабом

- Команда

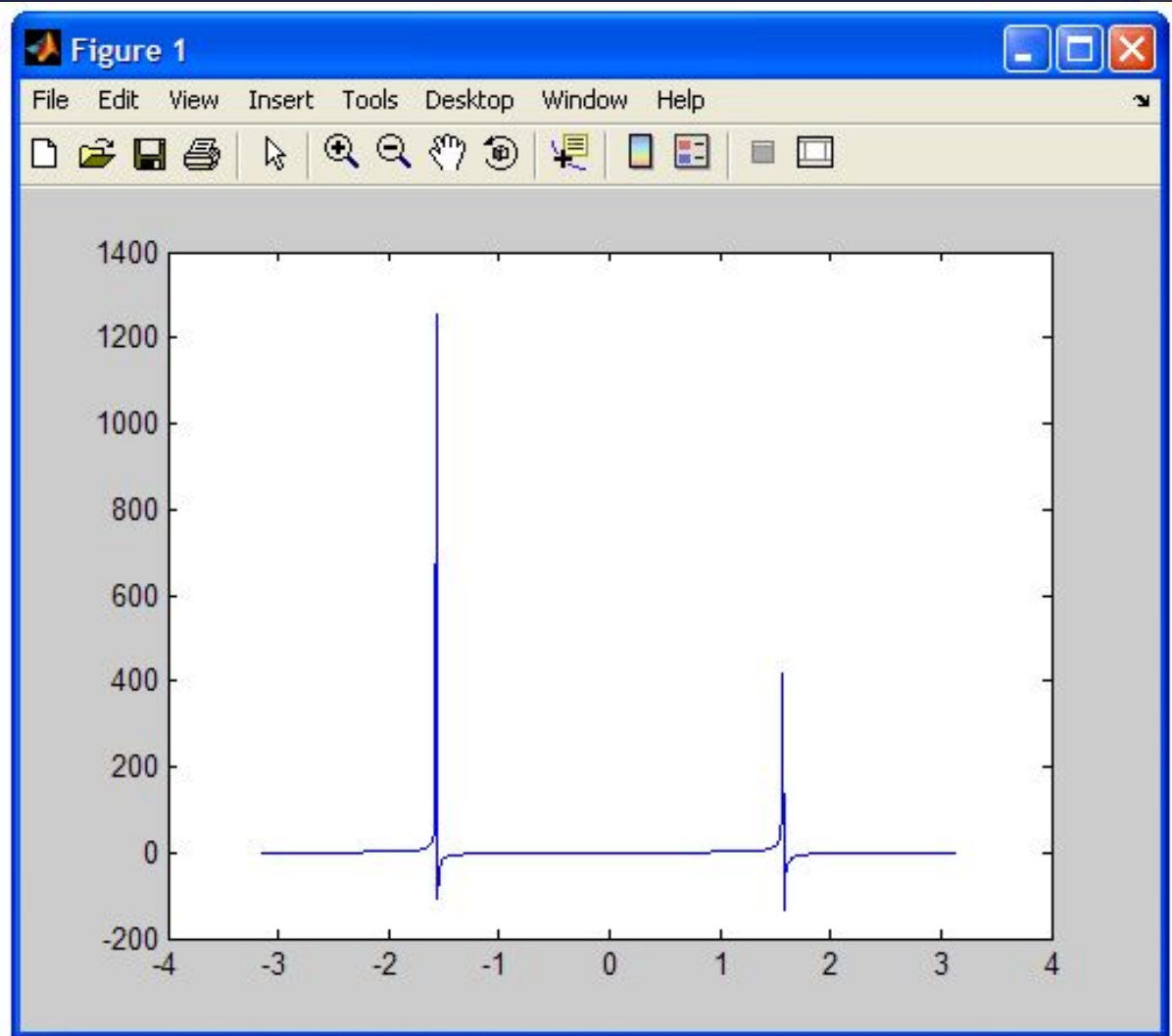
`axis([Xmin Xmax Ymin Ymax])`

задаёт область построения графиков по осям X и Y

- Используется, если результат автомасштабирования неудовлетворителен

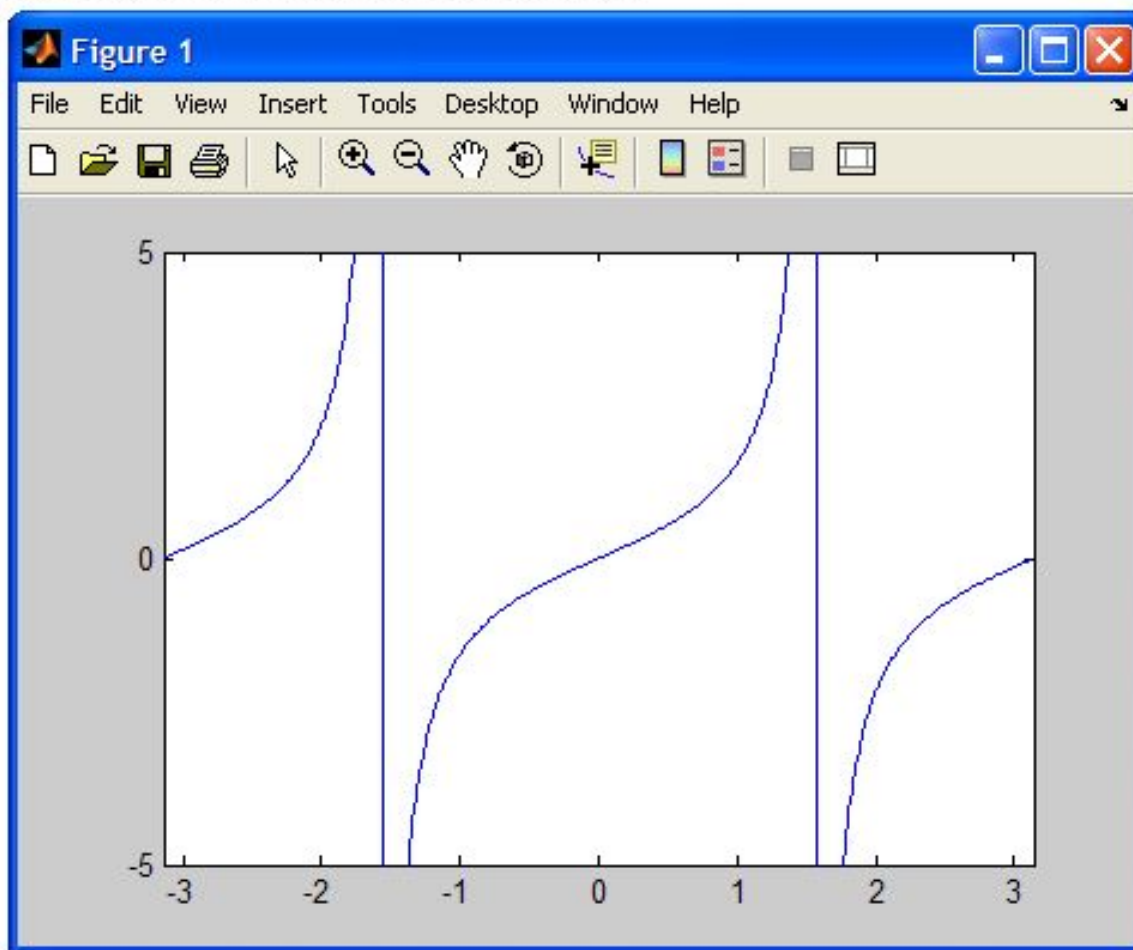
Axis не используется

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



Axis ИСПОЛЬЗУЕТСЯ

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

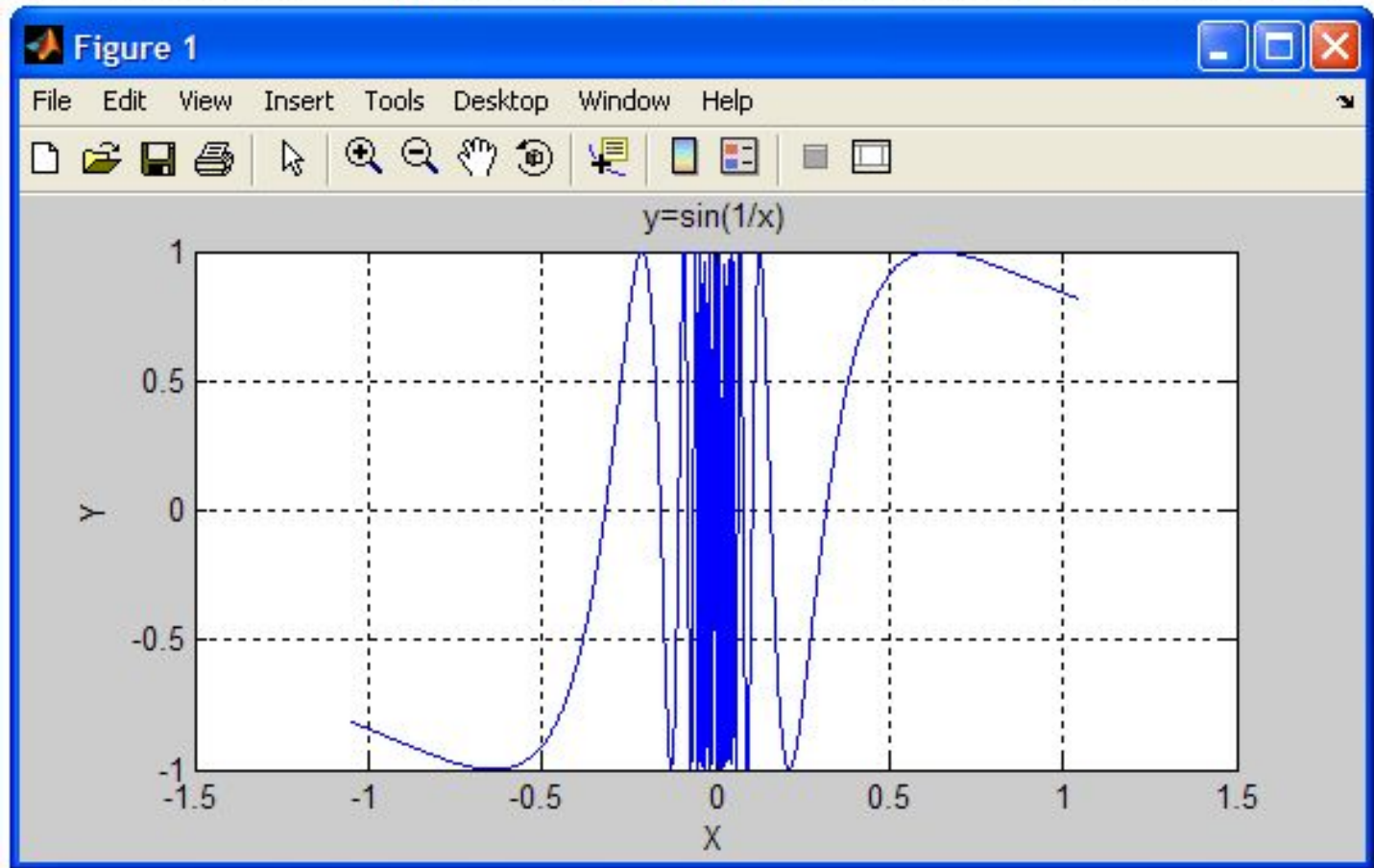


Оформление графиков

- ⦿ Для графиков можно задать
 - масштабную сетку: `grid on`
 - заголовок: `title ('заголовок')`
 - подписи осей: `xlabel ('текст')` и `ylabel ('текст')`
- ⦿ В заголовках и подписях можно использовать нотацию системы **TeX**

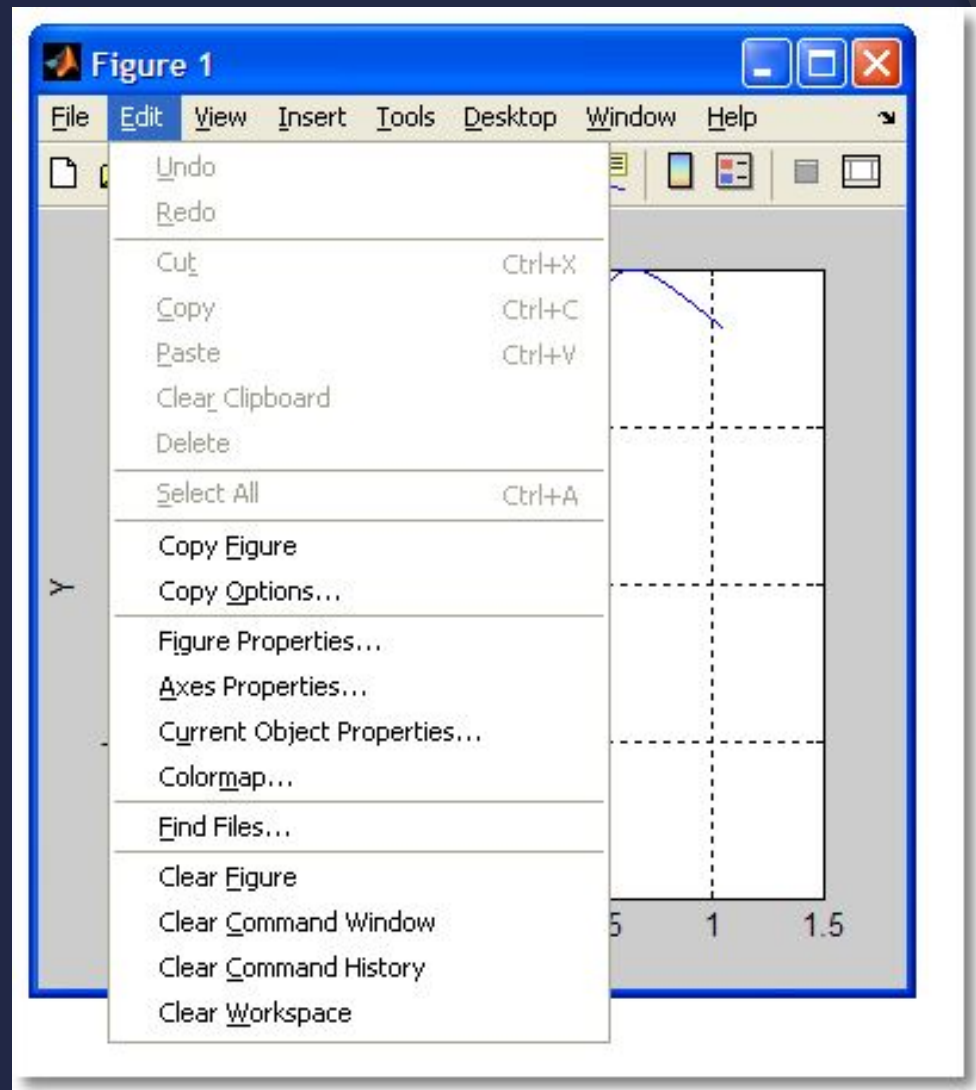
Пример оформления графика

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



Форматирование графиков

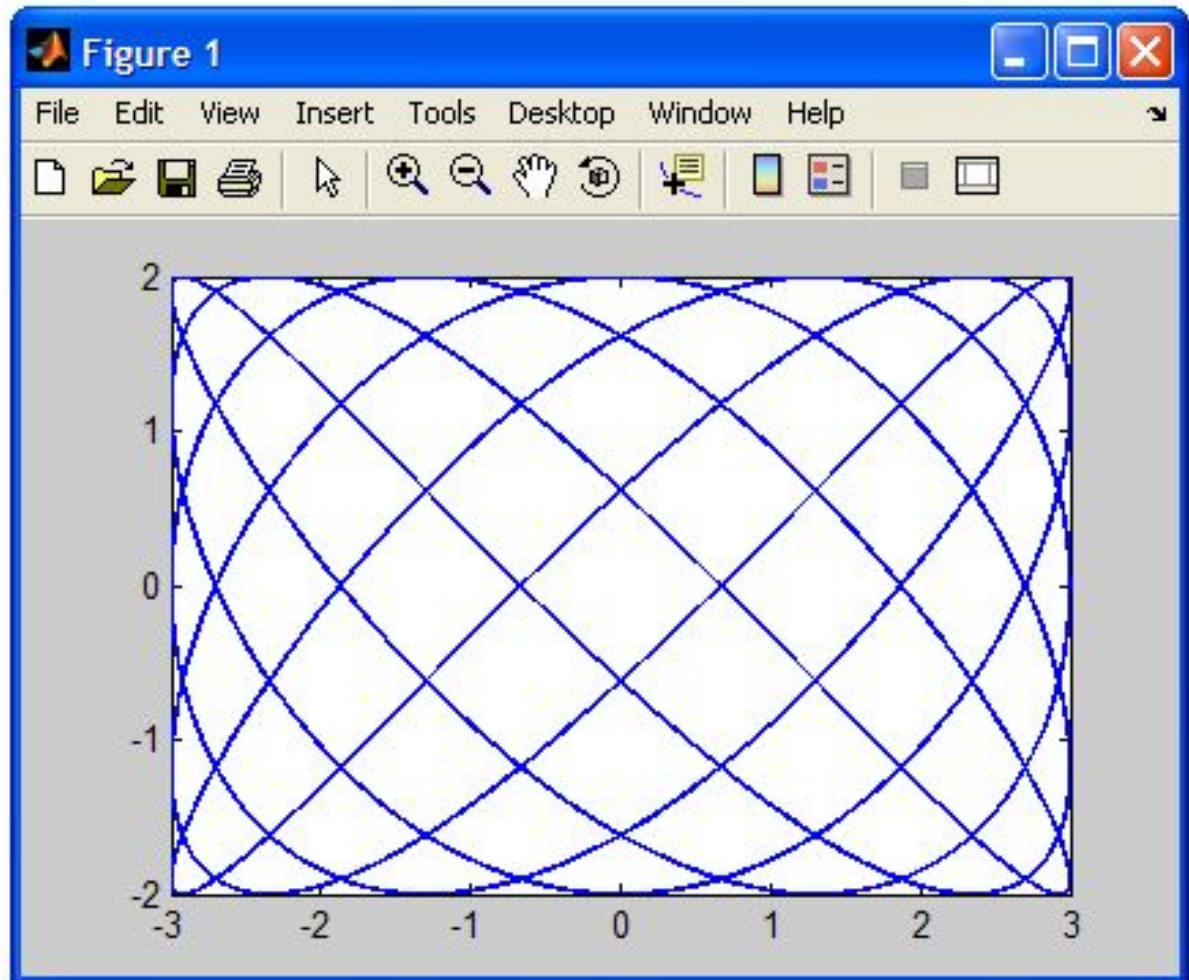
- Доступно из меню **Edit**:



Графики функций, заданных параметрически

- ⦿ Строятся при помощи оператора *plot*
- ⦿ Вначале задаётся диапазон построения t
- ⦿ Затем вычисляются $x(t)$ и $y(t)$
- ⦿ И строится график

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

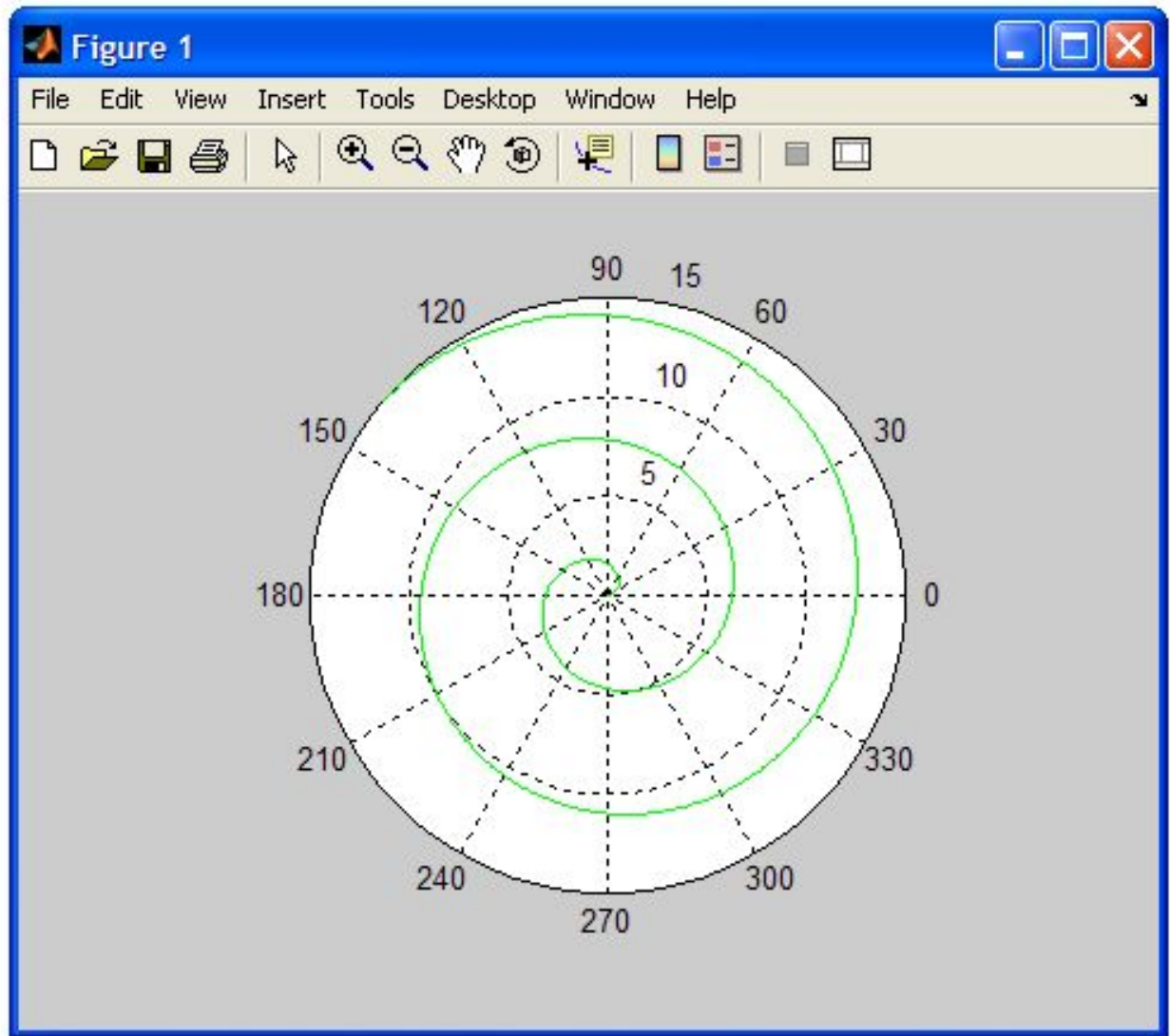


- ⦿ Графики параметрических функций часто возникают в физических приложениях
- ⦿ Независимая переменная t в этом случае имеет смысл времени, x и y — координаты
- ⦿ Для построения динамического графика можно использовать функцию $comet(x, y)$

Функции в полярной СК

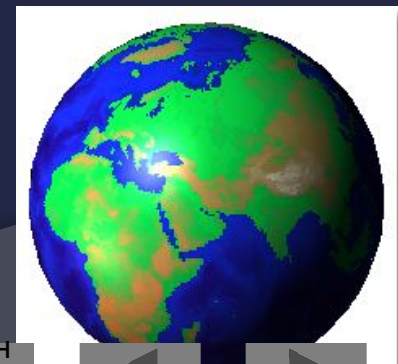
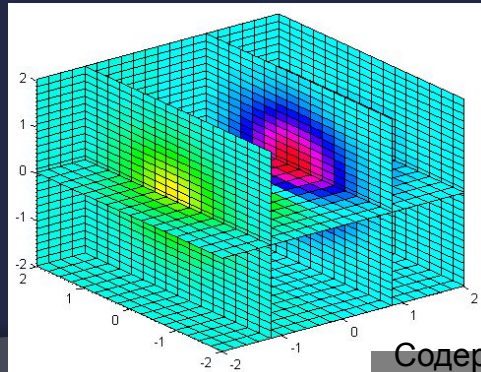
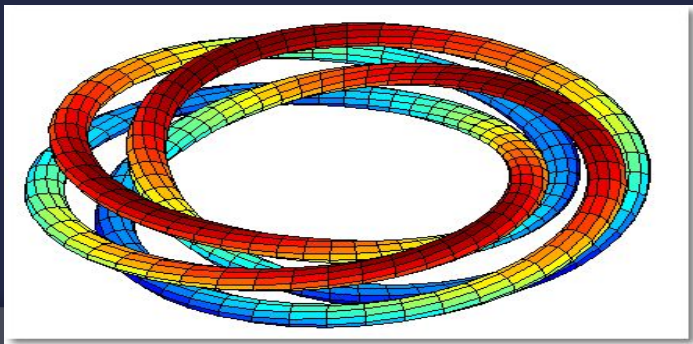
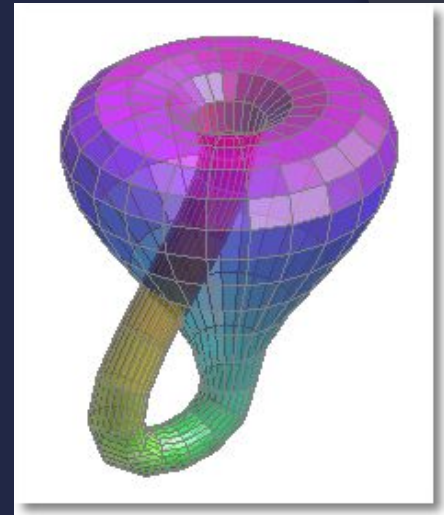
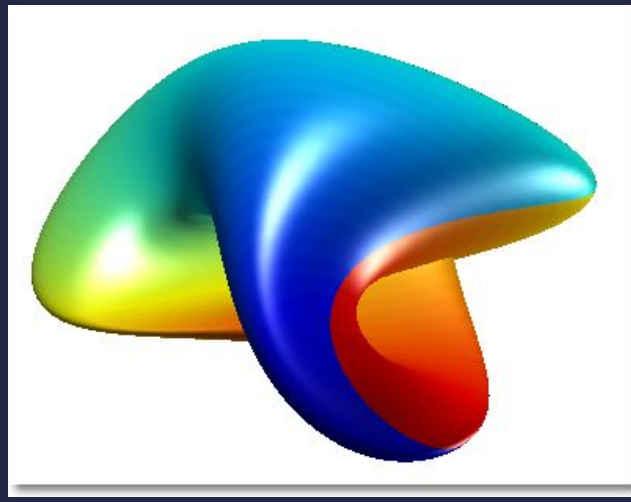
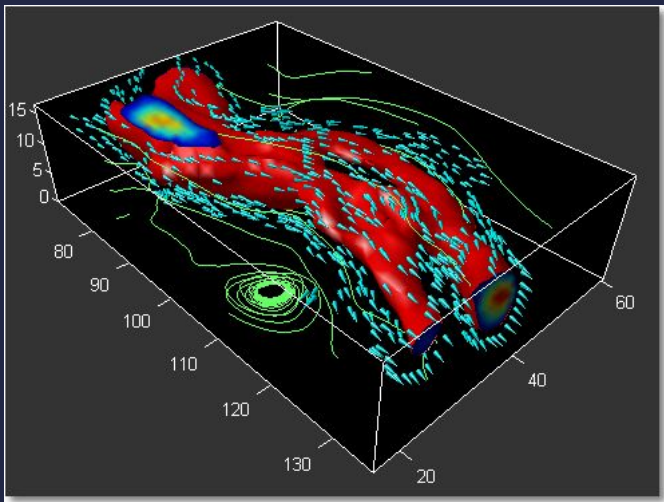
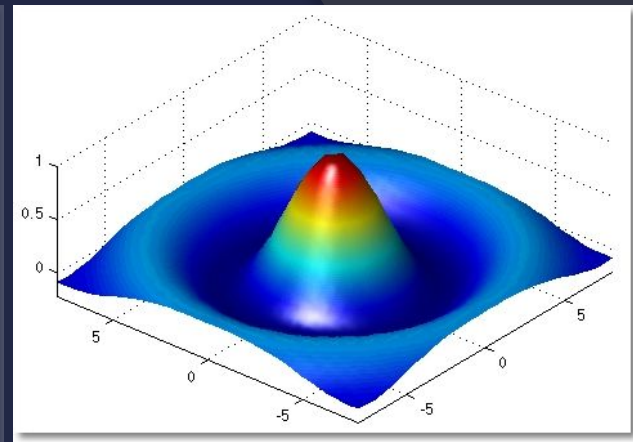
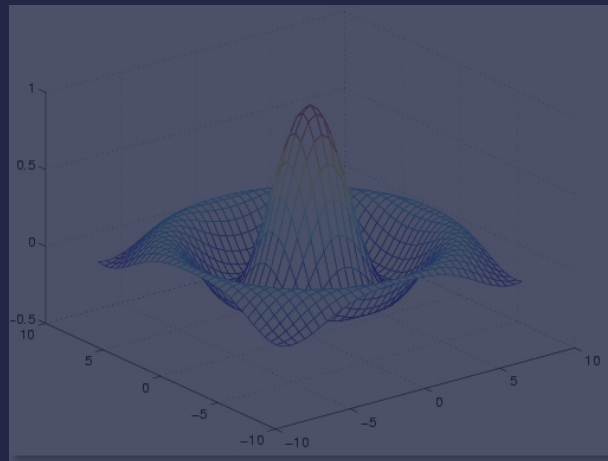
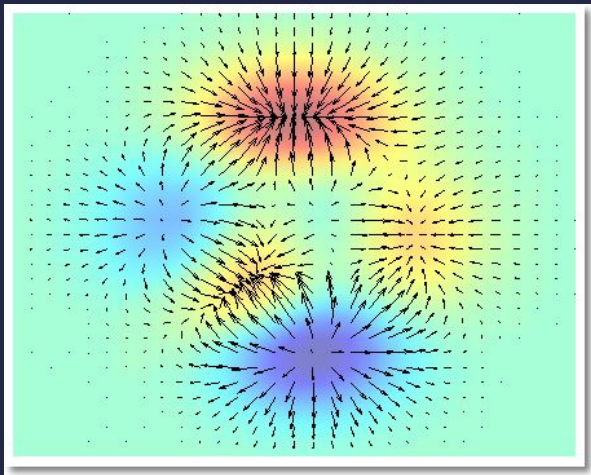
- Строятся аналогично графикам функций в декартовой системе
- Для построения используется команда *polar*

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



Трёхмерная (3D-) графика

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



Содержание

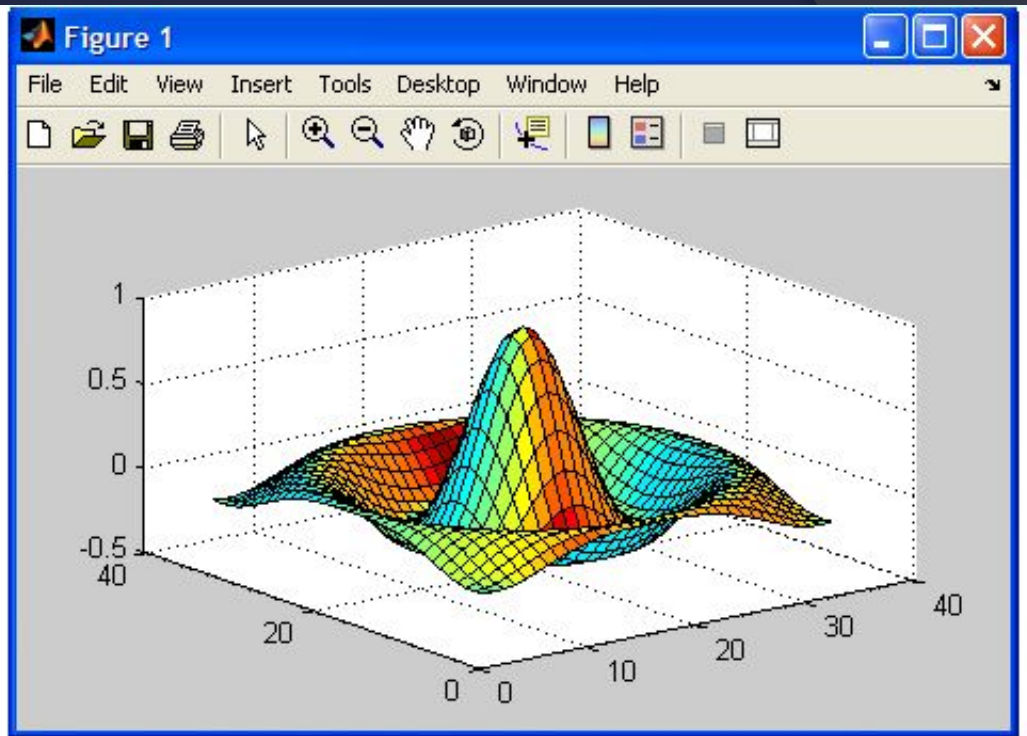
Выход

Построение 3D-поверхности

⦿ Рассмотрим пример:

построить поверхность $f(x,y) = \sin(r)/r$, где $r = \sqrt{x^2 + y^2}$

```
>> [X,Y] = meshgrid(-8:.5:8);  
>> R = sqrt(X.^2 + Y.^2) + eps;  
>> Z = sin(R)./R;  
>> surf1(Z)  
>>
```



- Функция `meshgrid` возвращает две матрицы – X и Y – которые определяют область построения функции
- Если диапазоны по X и Y разные, то функции передаются два диапазона
- Собственно поверхность выводится функцией `surf1`

Функции для построения поверхностей

Функция	Для чего используется
<code>mesh, surf</code>	Построение поверхностей
<code>meshc, surfc</code>	Строит поверхность и контурную диаграмму под ней
<code>meshz</code>	Поверхность на «пьедестале»
<code>surfl</code>	Подсвеченная поверхность
<code>contour</code>	Контурная диаграмма
<code>plot3</code>	Трёхмерная линия (параметрическое задание)
<code>comet3</code>	Движение по трёхмерной линии

- О других графических функциях можно узнать в системе помощи Matlab

ПРОГРАММИРОВАНИЕ В MATLAB



Типы программных файлов

- ⦿ 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*)

Скрипты

- Представляют собой последовательности команд Matlab
 - как если бы мы перенесли их из командного окна в отдельный файл
- Вызываются по имени через командную строку.
- Выполняются в режиме построчного анализа, обработки и выполнения исходных команд

- ⦿ Полезны для автоматизации последовательности действий, которые выполняются многократно.
- ⦿ Не могут принимать параметры и возвращать аргументы.
- ⦿ Хранят значения своих переменных в рабочем пространстве
 - где переменные доступны для других скриптов и из командной строки

Функции

- ⦿ Специальный вид m-файлов.
- ⦿ В отличие от скриптов могут принимать аргументы и возвращать значения.
- ⦿ Использование функций позволяет
 - структурировать программу
 - 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
 - будем соблюдать правило: **имя функции и имя *m*-файла должны быть одинаковы**

Структура функции

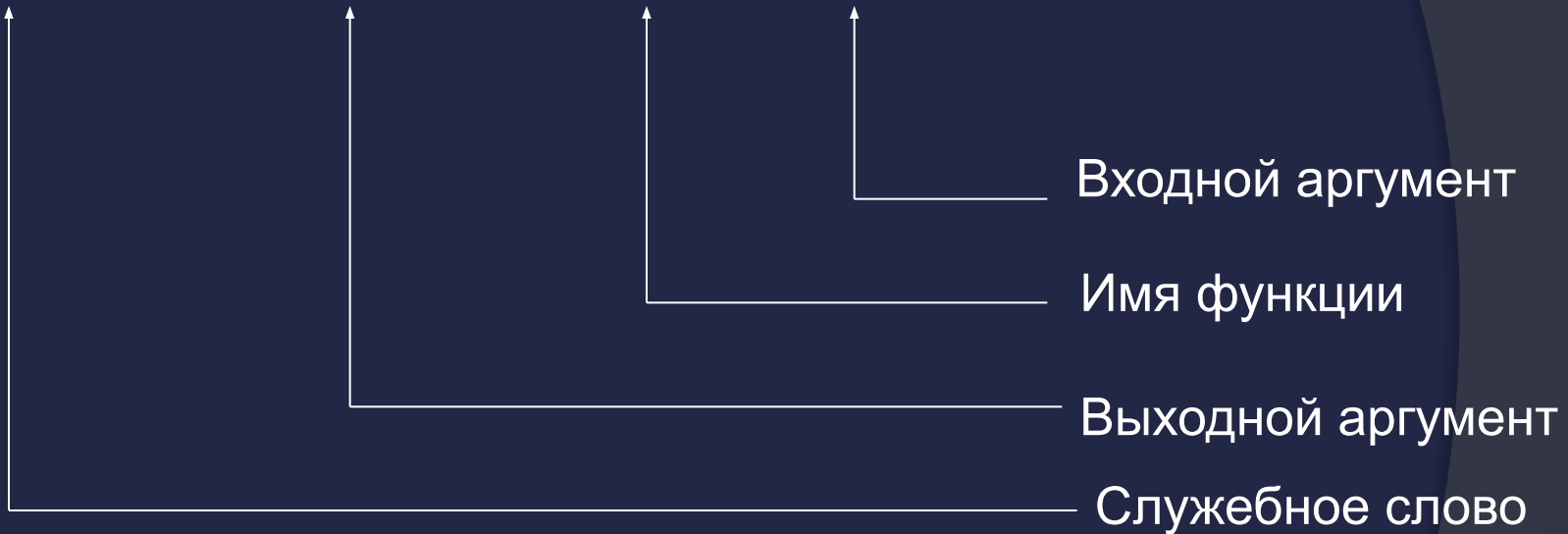
- Функция состоит из заголовка и тела

<code>function f = fact(n)</code>	Заголовок
<code>% Вычисляет факториал.</code>	Линия N1
<code>% FACT(N) возвращает N!,</code>	Help
<code>f = prod(1:n);</code>	Тело функции

- N1 и Help выводятся по команде `help <имя функции>`
- Фактически, функция отличается от скрипта наличием заголовка и способом вызова

Заголовок функции

```
function f = fact(n)
```



Комментарии

- Используются для
 - пояснения кода;
 - временного исключения кода из текста.
- Могут быть *строчными* и *блочными*
- Строчные начинаются с символа «`%`»
 - с этого места и до конца строки всё игнорируется компилятором `%` как в этом примере
- Блочные начинаются с символа «`% {`» и заканчиваются символом «`% }`»:
 - эти символы должны обязательно стоять в отдельных строках!

```
%{
```

```
%}
```


- ⦿ Можно автоматически закомментировать блок текста. Для этого:
 - выделить блок
 - щёлкнуть правой кнопкой
 - выбрать **Comment** (или **Ctrl+R**)
- ⦿ Снять комментарий:
 - выделить закомментированный блок
 - щёлкнуть правой кнопкой
 - выбрать **Uncomment** (или **Ctrl+T**)

Создание функции

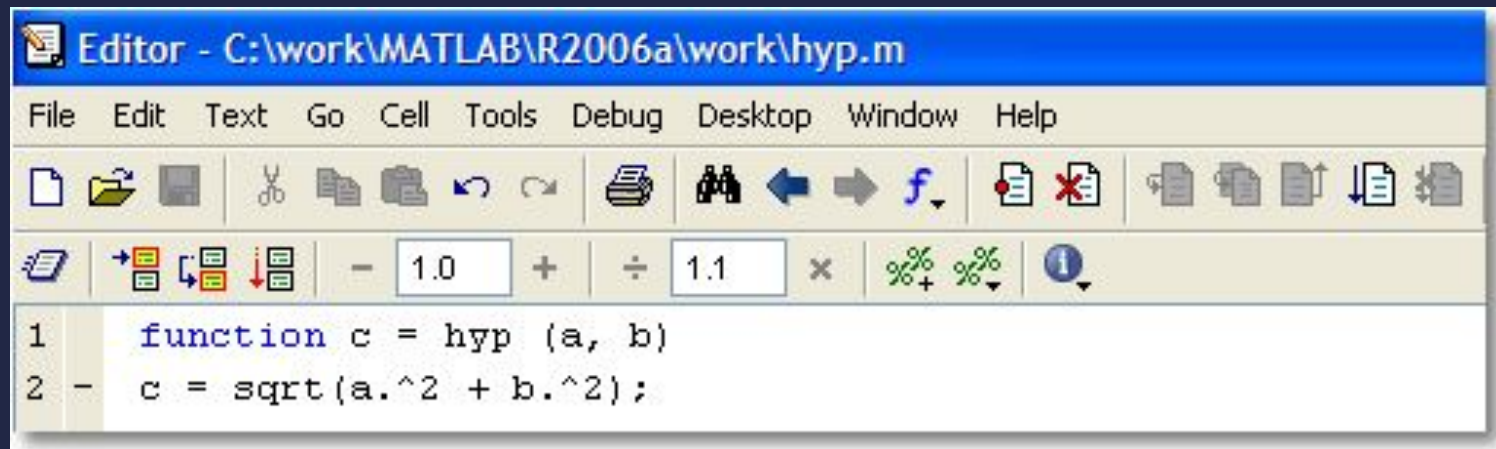
- ⦿ m-файл можно создать в любом текстовом редакторе.
- ⦿ Например, во встроенном редакторе
 - при помощи меню
 - или командой

```
edit <имя файла>
```



Использование функции

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



The screenshot shows the MATLAB Editor window titled "Editor - C:\work\MATLAB\R2006a\work\hyp.m". The menu bar includes File, Edit, Text, Go, Cell, Tools, Debug, Desktop, Window, and Help. The toolbar contains various icons for file operations and editing. The code editor displays the following function definition:

```
1 function c = hyp (a, b)
2 - c = sqrt(a.^2 + b.^2);
```

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

z =

    5.0990    6.3246    7.6158
```

Входные и выходные параметры

- 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

```

1  function [u, v] = wh (a, b)
2
3  -  switch nargin
4  -      case 1
5  -          if nargin == 1
6  -              u = a.^2;
7  -          else
8  -              u = a;
9  -              v = 1/a;
10 -          end
11 -      case 2
12 -          if nargin == 1
13 -              u = a.^2 + b.^2;
14 -          else
15 -              u = a + b;
16 -              v = 1./(a + b);
17 -          end
18 -  end

```

```

>> x = wh(2)
x =
    4
>> [x, y] = wh(2)
x =
    2
y =
    0.5000
>> x = wh(2, 3)
x =
    13
>> [x, y] = wh(2, 3)
x =
    5
|
y =
    0.2000

```

Подфункции

- ⦿ 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
 - 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

```

Вложенные функции

- ⦿ Помимо последовательного вложения в один файл функция может быть описана непосредственно в теле другой функции
- ⦿ Такая функция называется *вложенной*
- ⦿ Вложенная функция, в свою очередь, может содержать другие вложенные функции


```
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
```

Создание р-кода

- При вызове m-файла сравнительно много времени тратится на его компиляцию
- Чтобы сократить время выполнения можно предварительно перевести m-файл в р-код («пи-код»)
 - команда `rcode <имя m-файла>`
- Откомпилированный в псевдокод файл получает расширение «р»
- 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`

Пример использования `input`

```
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.
```

Вывод данных в командное окно

- Для этого 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`

Основные языковые конструкции

- 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`

Оператор `if`

- Простейшая форма:

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

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

Полный формат оператора `if`

- 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`
- ◎ Также имеются
 - оператор досрочного выхода из цикла `break`
 - оператор перехода к следующей итерации `continue`

Цикл с параметром

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

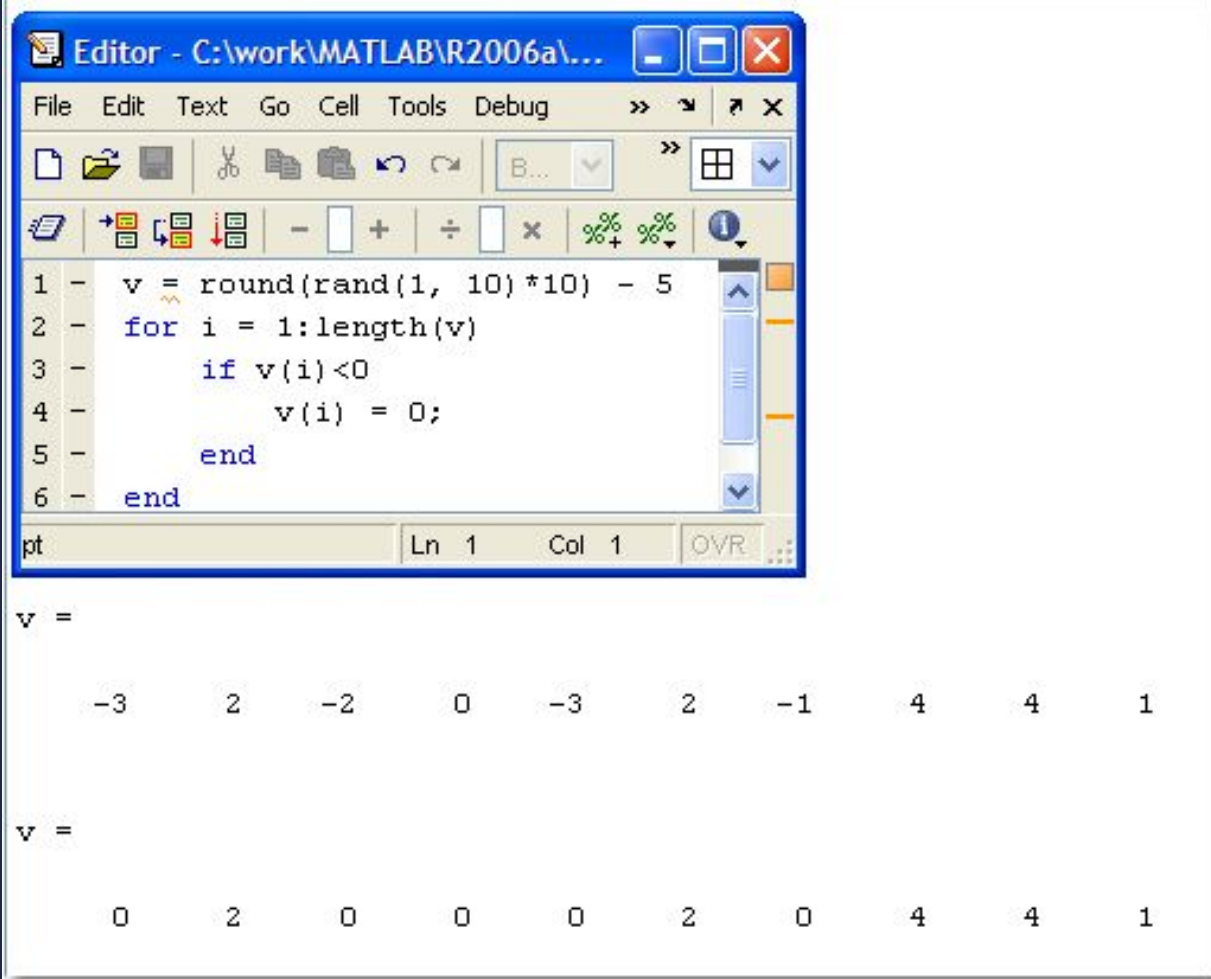
```
for n = 2:6
    x(n) = 2 * x(n - 1);
end
```

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

Замечание по использованию цикла с параметром

- 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)



```
Editor - C:\work\MATLAB\R2006a\...
File Edit Text Go Cell Tools Debug
v = round(rand(1, 10)*10) - 5
for i = 1:length(v)
    if v(i)<0
        v(i) = 0;
    end
end
```

pt Ln 1 Col 1 OVR

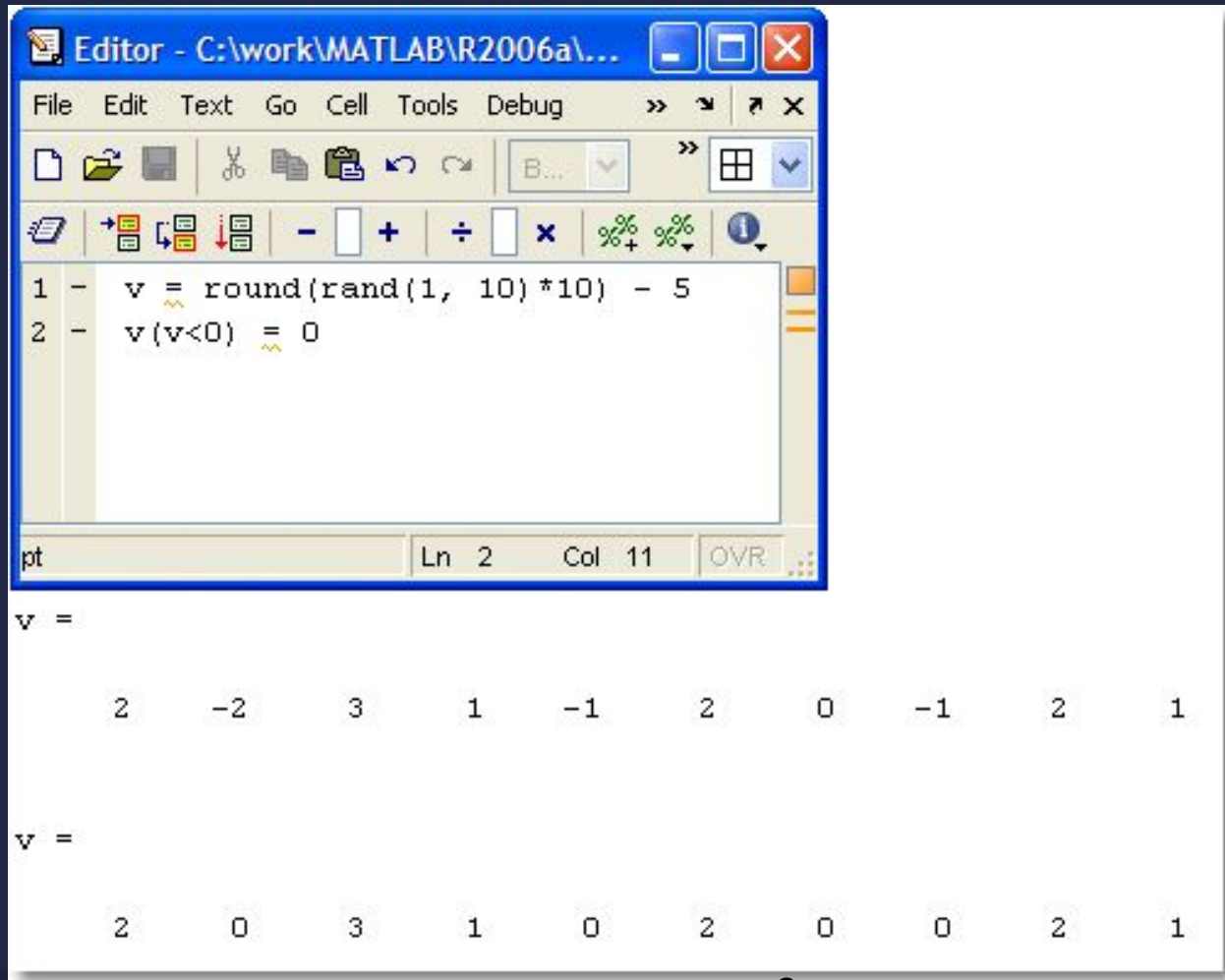
v =

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

v =

0	2	0	0	0	2	0	4	4	1
---	---	---	---	---	---	---	---	---	---

Пример: замена отрицательных элементов вектора на нули (без цикла)



The screenshot shows the MATLAB Editor window with the following code:

```
1 - v = round(rand(1, 10) * 10) - 5
2 - v(v < 0) = 0
```

The output of the code is displayed below the editor:

```
v =
     2     -2     3     1     -1     2     0     -1     2     1

v =
     2     0     3     1     0     2     0     0     2     1
```

Цикл с предусловием

- Синтаксис:

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

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


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

```
Editor - C:\work\MATLAB\R2006a\work\wh.m
File Edit Text Go Cell Tools Debug Desktop Window
File Edit Text Go Cell Tools Debug Desktop Window
1 - n = 1; p = 1;
2 - while n < 10
3 -     p = p * n;
4 -     n = n + 2;
5 - end
6 - disp(['p = ', num2str(p)])
script Ln 6 Col 25 OVR
```

Операторы `break` и `continue`

- Аналогичны одноимённым операторам Паскаля
- `Break` производит досрочный выход из цикла `for` или `while`
- `Continue` прекращает выполнение текущей итерации и переходит к следующей

Операторы `break` и `continue` (пример)

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

Операторы `break` и `continue` (решение)

```
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)])
```

АНАЛИТИЧЕСКИЕ ВЫЧИСЛЕНИЯ В MATLAB



Вычисления в Matlab

- Пример: вычисление определённого интеграла

$$\int_a^b f(x)$$

1. По формуле Ньютона: $F(x)|_a^b = F(b) - F(a)$, где $F(x)$ – первообразная

- получаем *точный* результат
- но первообразную не всегда можно найти

2. Численно: методом прямоугольников, трапеций, Симпсона и пр.

- можно пользоваться даже тогда, когда интеграл «не берётся»
- но при вычислении возникают погрешности

Средства Matlab для СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ

- 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

Создание символьных переменных

- Для символьного анализа требуется создать символьные переменные и функции
- Символьные переменные создаются
 - по одной: `x=sym('x')`
 - так же можно создать целое символьное выражение
 - несколько сразу: `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 – командой `pretty`


```
>> 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)
```

$$\frac{(\sin(x) + a)^2 \cos^2(y)}{\sqrt{|a - y|}}$$

Представление символьных переменных

```
>> 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
```

Символьные вычисления

- Преобразования математического анализа
 - дифференцирование,
 - пределы,
 - интегрирование,
 - разложение в ряд Тейлора
- Упрощение и подстановки
- Точная арифметика
- Линейная алгебра
- Решение уравнений и их систем
 - обычных и дифференциальных

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)    % а теперь -- вторую производную по x

ans =

-25*sin(5*x)

>> c = sym('5'); diff(c)    % производная константы

ans =

0
```

Частные производные

```
>> 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
```

```
>> % вторая частная производная вычисляется так:
>> diff (f, t, 2)

ans =

-sin(s*t)*s^2

>> % или так:
>> diff (f, 2) % по умолчанию -- производная по t

ans =

-sin(s*t)*s^2

>> % или так:
>> diff(diff(f,t),s)

ans =

-sin(s*t)*t*s+cos(s*t)
```

Пределы

```
>> syms h n x
>> limit( (cos(x+h) - cos(x))/h, h, 0 )  % вычисляем предел при h -> 0

ans =

-sin(x)

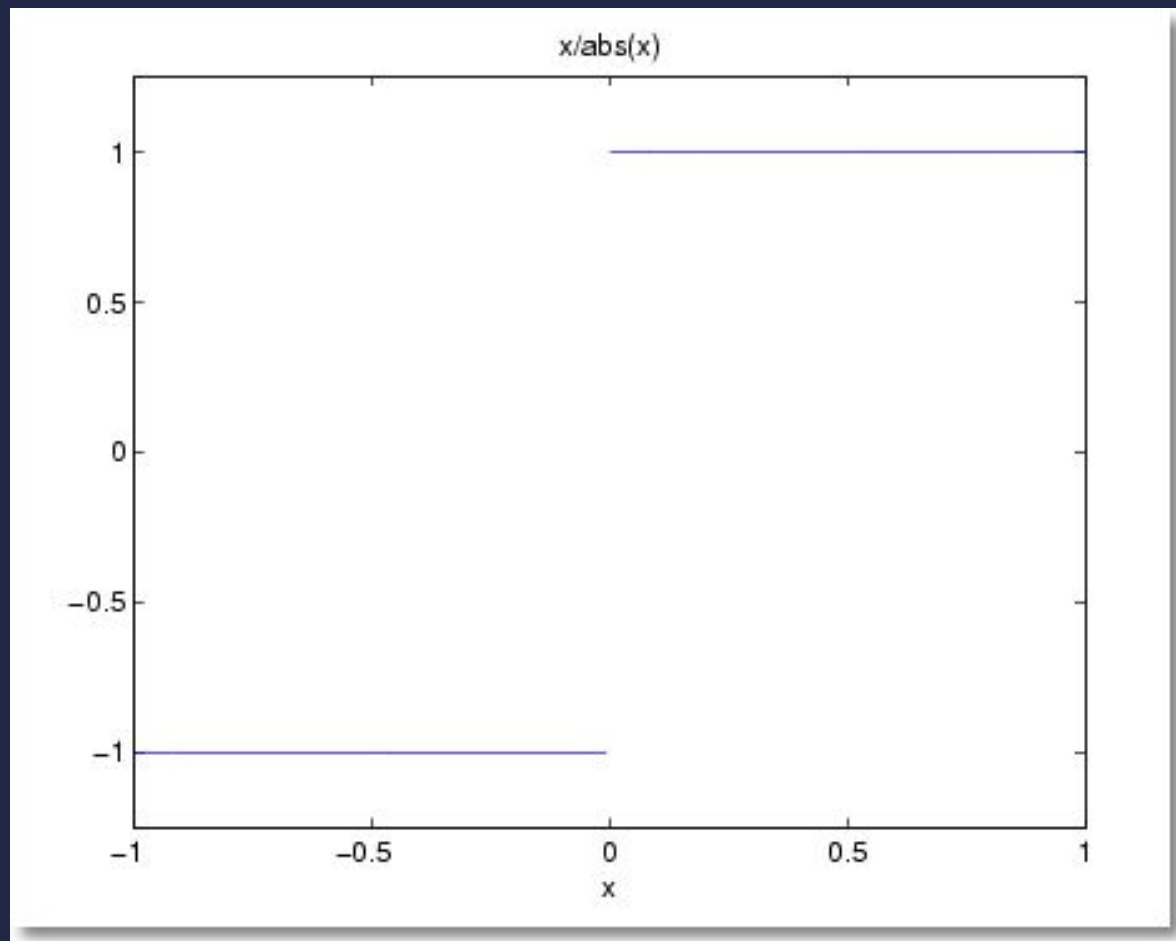
>> limit( (1 + x/n)^n, n, inf )  % а теперь -- при n -> oo

ans =

exp(x)
```

Односторонние пределы

- Рассмотрим функцию $f(x)=x/|x|$




```
>> limit(x/abs(x),x,0,'left')
```

```
ans =
```

```
-1
```

```
>> limit(x/abs(x),x,0,'right')
```

```
ans =
```

```
1
```

Пределы (сводная таблица)

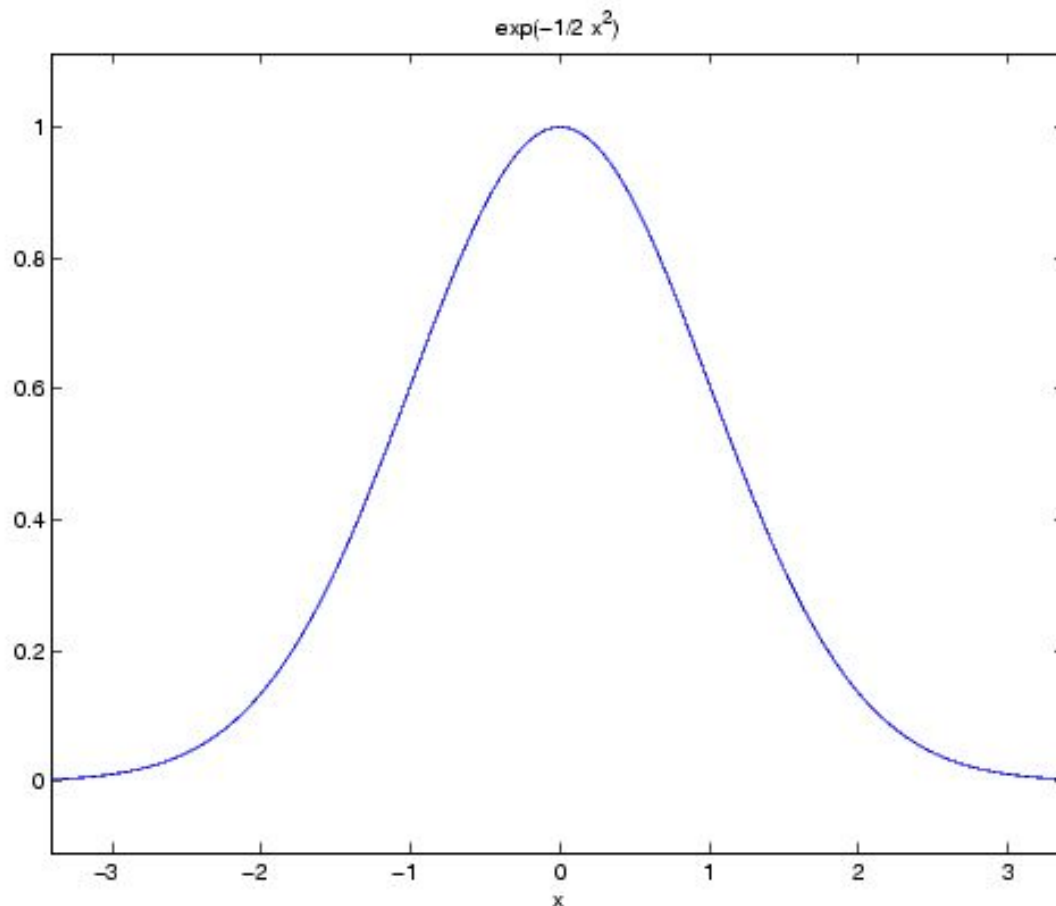
Mathematical Operation	MATLAB Command
$\lim_{x \rightarrow 0} f(x)$	<code>limit(f)</code>
$\lim_{x \rightarrow a} f(x)$	<code>limit(f, x, a)</code> or <code>limit(f, a)</code>
$\lim_{x \rightarrow a^-} f(x)$	<code>limit(f, x, a, 'left')</code>
$\lim_{x \rightarrow a^+} f(x)$	<code>limit(f, x, a, 'right')</code>

Интегралы

Mathematical Operation	MATLAB Command
$\int x^n dx = \frac{x^{n+1}}{n+1}$	<code>int(x^n)</code> or <code>int(x^n, x)</code>
$\int_0^{\pi/2} \sin(2x) dx = 1$	<code>int(sin(2*x), 0, pi/2)</code> or <code>int(sin(2*x), x, 0, pi/2)</code>
$g = \cos(at + b)$ $\int g(t) dt = \sin(at + b)/a$	<code>g = cos(a*t + b)</code> <code>int(g)</code> or <code>int(g, t)</code>
$\int J_1(z) dz = -J_0(z)$	<code>int(besselj(1, z))</code> or <code>int(besselj(1, z), z)</code>

Интегралы с параметрами

```
syms x  
a = sym(1/2);  
f = exp(-a*x^2);  
ezplot(f)
```



```

>> syms a positive;
>> syms x;
f = exp(-a*x^2);
int(f,x,-inf,inf)

ans =

1/a^(1/2)*pi^(1/2)

>> syms a real
f=exp(-a*x^2);
F = int(f, x, -inf, inf)

F =

PIECEWISE([1/a^(1/2)*pi^(1/2), signum(a) = 1],[Inf, otherwise])

>> pretty(F)

      { 1/2
      { pi
      { -----      signum(a~) = 1
      { 1/2
      { a~
      {
      { Inf      otherwise

```

Суммирование

```
>> syms x k
>> s1 = symsum(1/k^2, 1, inf)

s1 =

1/6*pi^2

>> s2 = symsum(x^k, k, 0, inf)

s2 =

-1/(x-1)
```

Разложение в ряд Тейлора

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

Попробуйте также
команду
`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

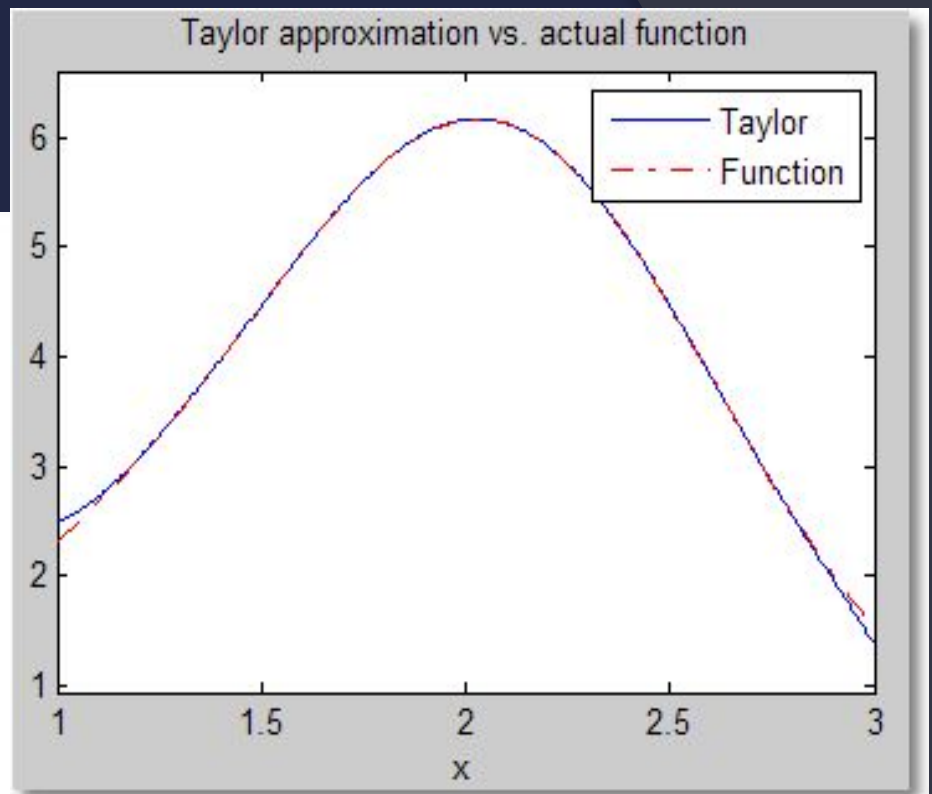
>> pretty(T)

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

```
>> syms x
>> g = exp(x*sin(x))
```

```
g =
exp(x*sin(x))
```

```
>> t = taylor(g,12,2);
>> xd = 1:0.05:3; yd = subs(g,x,xd);
>> ezplot(t, [1,3]); hold on;
>> plot(xd, yd, 'r-.')
>> title('Taylor approximation vs. actual function');
>> legend('Taylor', 'Function')
```



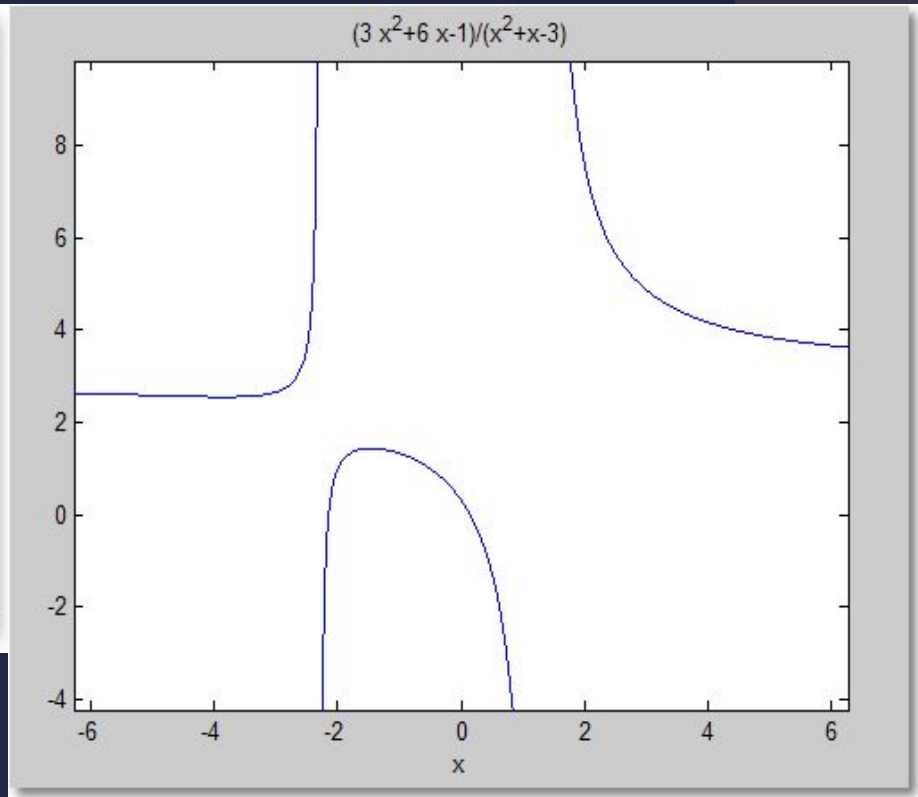
Пример: исследование функции

```
>> % исследуем поведение функции
>> syms x
>> chisl = 3*x^2 + 6*x - 1;
>> znam = x^2 + x - 3;
>> f = chisl / znam

f =

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

$$f(x) = \frac{3x^2 + 6x - 1}{x^2 + x - 3}$$



Найдём горизонтальную асимптоту

```
>> h_as = limit(f, inf)
```

```
h_as =
```

```
3
```

```
>> limit(f, -inf)
```

```
ans =
```

```
3
```

Найдём вертикальные асимптоты

```
>> % для этого решим уравнение      znam = 0
>> roots = solve(znam)

roots =

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

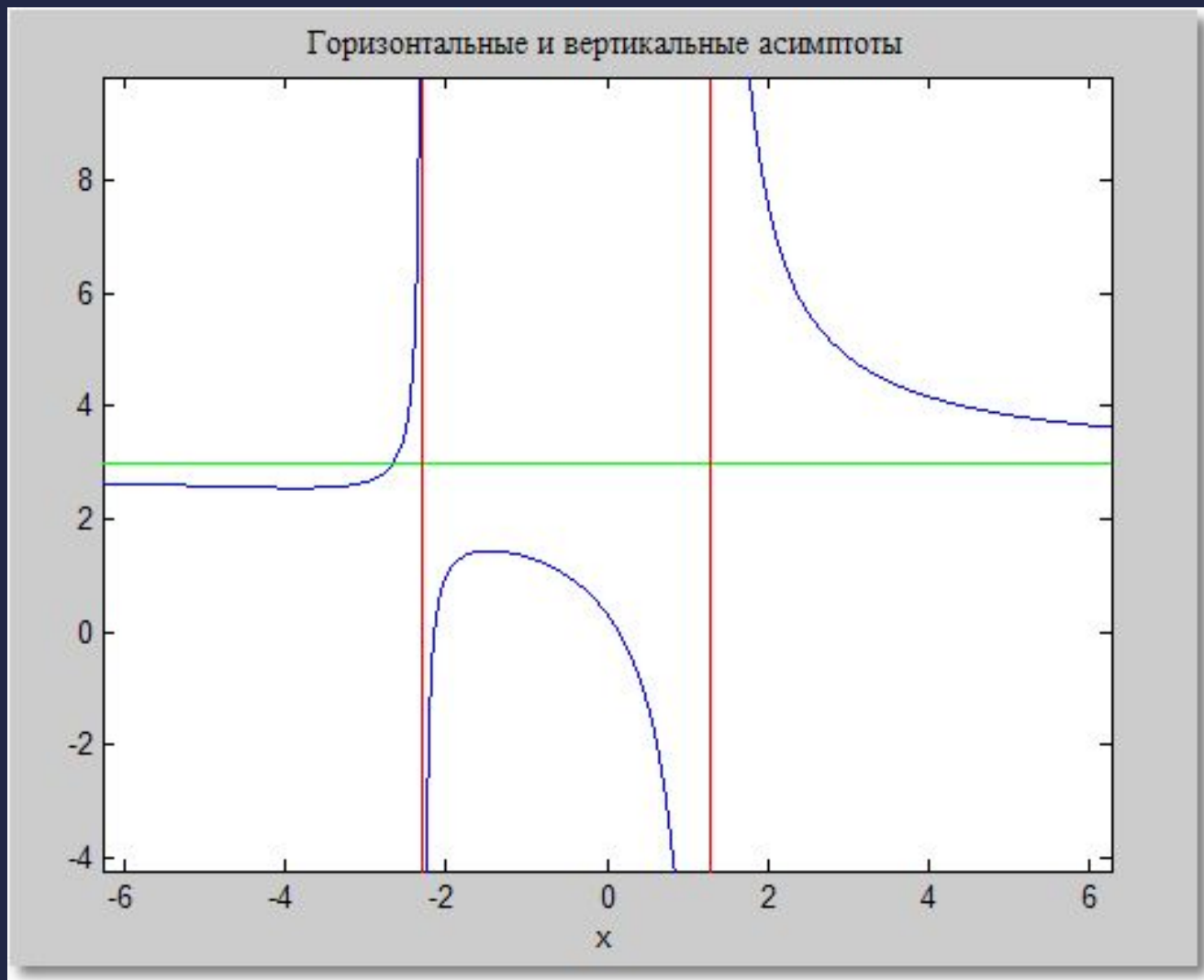
$$x = \frac{-1 + \sqrt{13}}{2}$$

$$x = \frac{-1 - \sqrt{13}}{2}$$

Код для построения асимптот

```
ezplot(f)
hold on
% Строим горизонтальную асимптоту
plot([-2*pi 2*pi], [3 3], 'g')
% Строим вертикальные асимптоты
% Не забываем перевести символьные значения в обычные
% при помощи функции double
plot(double(roots(1))*[1 1], [-5 10], 'r')
plot(double(roots(2))*[1 1], [-5 10], 'r')
title('Горизонтальные и вертикальные асимптоты')
hold off
```

Изображение асимптот



Экстремумы 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)
```

$$\frac{6x + 6}{x^2 + x - 3} - \frac{(3x^2 + 6x - 1)(2x + 1)}{(x^2 + x - 3)^2}$$

```
>> % Приравняем производную к нулю и решим уравнение
```

```
>> extr = solve(f1)
```

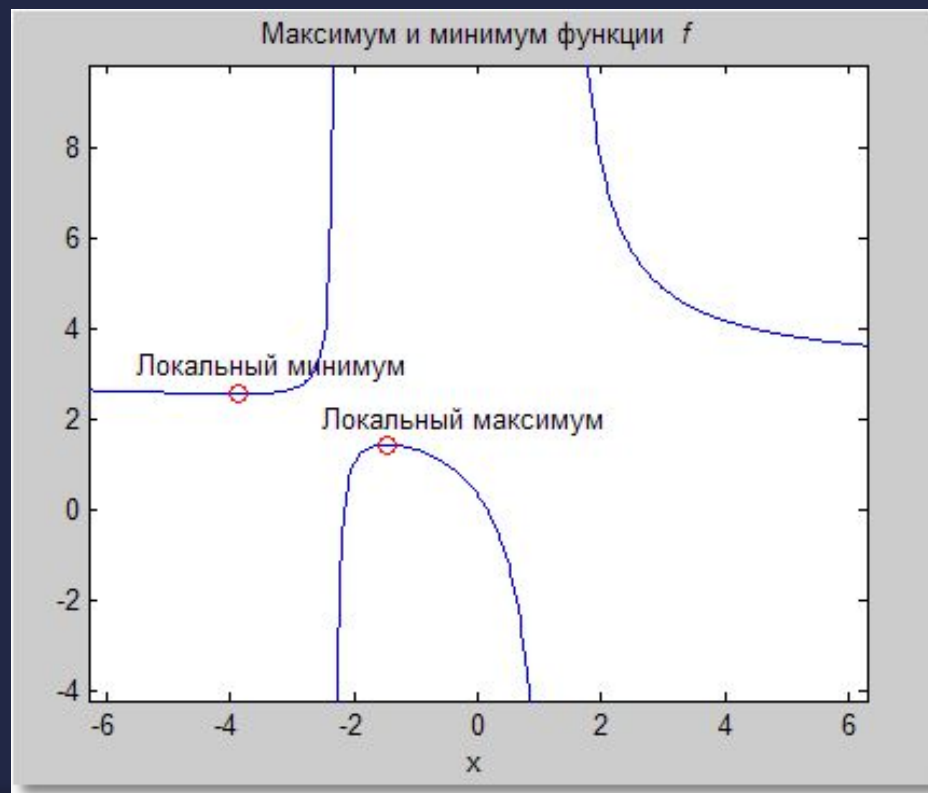
```
extr =
```

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

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

Построение экстремумов

```
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
```



Операции над полиномами

- ⊙ Реализуются при помощи функций
 - `collect`
 - `expand`
 - `factor`
 - `horner`

- ⊙ `collect` – вычисляет коэффициенты при степенях независимой переменной
 - по умолчанию – x
- ⊙ Можно явно задать имя независимой переменной в виде:
 - `collect (f, VarName)`

f	collect(f)
<code>(x-1) * (x-2) * (x-3)</code>	<code>x^3-6*x^2+11*x-6</code>
<code>x * (x * (x-6) + 11) - 6</code>	<code>x^3-6*x^2+11*x-6</code>
<code>(1+x) * t + x * t</code>	<code>2 * x * t + t</code>

- ⦿ `expand` – представляет полином суммой степеней без приведения подобных

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

- ⦿ `factor` – разлагает полином на множители, если эти множители имеют рациональные коэффициенты:

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

- Также `factor` производит каноническое разложение числа:

```
>> a = sym('2469887888872349872340987822200002000')  
  
a =  
  
2469887888872349872340987822200002000  
  
>> factor(a)  
  
ans =  
  
(2)^4*(3)*(5)^3*(53)*(109)*(52517526926913910058161)*(1356811)
```

- ⦿ `horner` – представляет полином в схеме Горнера:

f	horner(f)
$x^3 - 6x^2 + 11x - 6$	$-6 + (11 + (-6 + x) * x) * x$
$1.1 + 2.2 * x + 3.3 * x^2$	$11/10 + (11/5 + 33/10 * x) * x$

Упрощение выражений

◎ `simplify`

- реализует мощный алгоритм упрощения с использованием тригонометрических, степенных, логарифмических, экспоненциальных функций, а также спецфункций (Бесселя, гипергеометрической, интеграла ошибок и пр.)

◎ `simple`

- пытается получить выражение, которое представляется меньшим числом символов, чем исходное, последовательно применяя все функции упрощения Symbolic Math Toolbox

Simplify

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

Simple

```
>> simple(cos(x)^2 + sin(x)^2)

simplify: 1
radsimp: cos(x)^2+sin(x)^2
combine(trig): 1
factor: cos(x)^2+sin(x)^2
expand: cos(x)^2+sin(x)^2
combine: 1
convert(exp): (1/2*exp(i*x)+1/2/exp(i*x))^2-1/4*(exp(i*x)-1/exp(i*x))^2
convert(sincos): cos(x)^2+sin(x)^2
convert(tan): (1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^2)^2+4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2
collect(x): cos(x)^2+sin(x)^2
mwcos2sin: 1

ans =

1
```


Simplify против Simple

- Иногда `simple` даёт более удачное решение, чем `simplify`:

f	simplify(f)	simple(f)
$(1/a^3+6/a^2+12/a+8)^{(1/3)}$	$((2*a+1)^3/a^3)^{(1/3)}$	$(2*a+1)/a$
<pre>syms x y positive log(x*y)</pre>	$\log(x) + \log(y)$	$\log(x*y)$

Simple

- ◎ `simple` особенно эффективна при работе с тригонометрическими выражениями

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

Подстановка

- ⦿ `subs` подставляет одно символьное выражение в другое
- ⦿ Общий формат:
 - `subs (<куда>, <вместо чего>, <что>)`

Пример подстановки

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

$$\frac{a^2 + b^2}{a^2 - b^2} + \frac{a^4}{b^4}$$

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

$$\frac{(\exp(x) + \exp(-x))^2 + (\sin(x) + \cos(x))^2}{(\exp(x) + \exp(-x))^2 - (\sin(x) + \cos(x))^2} + \frac{(\exp(x) + \exp(-x))^4}{(\sin(x) + \cos(x))^4}$$

Подстановка значения в функцию

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

```
>> f = sym('exp(x^3 + 2*x^2 + x + 5)')  
  
f =  
  
exp(x^3 + 2*x^2 + x + 5)  
  
>> q = subs(f, 'x', 1.1)  
  
q =  
  
1.8977e+004
```

Точная арифметика

- Точные вычисления реализуются функцией `vpa` (Variable-Precision Arithmetic)
- Формат вызова:
 - `vpa` (<выражение>, <значащих цифр>)

```
>> vpa(pi,50)
```

```
ans =
```

```
3.1415926535897932384626433832795028841971693993751
```

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

```
ans =
```

```
1.618033989
```

```
>> vpa(exp(1), 70)
```

```
ans =
```

```
2.71828182845904553488480814849026501178741455078125000000000000000000
```

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
 - 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)
```

$$\begin{aligned} & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} + \frac{16}{3} \frac{1}{\left(10 + 6 \sqrt{111} \right)^{1/2}} + \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} \right] \\ & \left[-\frac{1}{6} \left(10 + 6 \sqrt{111} \right)^{1/2} - \frac{8}{3} \frac{1}{\left(10 + 6 \sqrt{111} \right)^{1/2}} + \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} - \frac{16}{3} \frac{1}{\left(10 + 6 \sqrt{111} \right)^{1/2}} \right] \\ & \left[-\frac{1}{6} \left(10 + 6 \sqrt{111} \right)^{1/2} - \frac{8}{3} \frac{1}{\left(10 + 6 \sqrt{111} \right)^{1/2}} + \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} \right] \\ & \left[-\frac{1}{6} \left(10 + 6 \sqrt{111} \right)^{1/2} - \frac{8}{3} \frac{1}{\left(10 + 6 \sqrt{111} \right)^{1/2}} + \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} \right] \\ & \left[-\frac{1}{6} \left(10 + 6 \sqrt{111} \right)^{1/2} - \frac{8}{3} \frac{1}{\left(10 + 6 \sqrt{111} \right)^{1/2}} + \frac{1}{3} \right] \\ & \left[\frac{1}{3} \left(10 + 6 \sqrt{111} \right)^{1/2} \frac{1}{3} \right] \end{aligned}$$

Решение систем

- ⦿ Также выполняет команда `solve`
- ⦿ 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

```

>> syms x1 x2
>> f1 = sym('a*x1^2 + x1*x2 + 1');
>> f2 = sym('x1^2 + b*x2');
>> s = solve(f1, f2, x1, x2);
>> r1 = subs(f1, {x1,x2}, {s.x1(1), s.x2(1)})
r1 =
a*(1/6*(108*b+8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)+2/3*a^2*b^2/(108*b+...
8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)+1/3*a*b)^2-(1/6*(108*b+8*a^3*b^3+...
12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)+2/3*a^2*b^2/(108*b+8*a^3*b^3+12*(81*b^2+12*...
a^3*b^4)^(1/2))^(1/3)+1/3*a*b)^3/b+1

>> simplify(r1)
ans = 0

>> r2 = subs(f2, {x1,x2}, {s.x1(1), s.x2(1)})
r2 = 0

>> r1 = subs(f1, {x1,x2}, {s.x1(2), s.x2(2)})
r1 =
a*(-1/12*(108*b+8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)-1/3*a^2*b^2/(108*b+...
8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)+1/3*a*b+1/2*i^3^(1/2)*(1/6*(108*b+...
8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)-2/3*a^2*b^2/(108*b+8*a^3*b^3+12*(81*...
b^2+12*a^3*b^4)^(1/2))^(1/3)))^2-(-1/12*(108*b+8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(...
(1/2))^(1/3)-1/3*a^2*b^2/(108*b+8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)+1/3*...
a*b+1/2*i^3^(1/2)*(1/6*(108*b+8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)-2/3*a^...
2*b^2/(108*b+8*a^3*b^3+12*(81*b^2+12*a^3*b^4)^(1/2))^(1/3)))^3/b+1

>> simplify(r1)
ans = 0

```

$$\begin{cases} ax_1^2 + x_1x_2 + 1 = 0; \\ x_1^2 + bx_2 = 0 \end{cases}$$

Решение дифференциальных уравнений

- Выполняет команда `dsolve`

Если неизвестная функция обозначена символьной переменной y , то ее производные следует обозначать как $d[n]y$, где в скобках указан порядок производной.

МАТЛАВ В ЗАДАЧАХ ТЕОРИИ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ (ТАУ)



Представление и преобразование математических моделей (ММ) динамических систем (ДС).

- Задание ДС в матрично-векторной (МВ) форме записи
- В современной теории управления для описания ММ ДС используется запись, представляющая собой систему уравнений в переменных состояния [1]

$$\begin{cases} \dot{x} = A \cdot x + B \cdot u; \\ y = C \cdot x + D \cdot u. \end{cases}$$

(1)

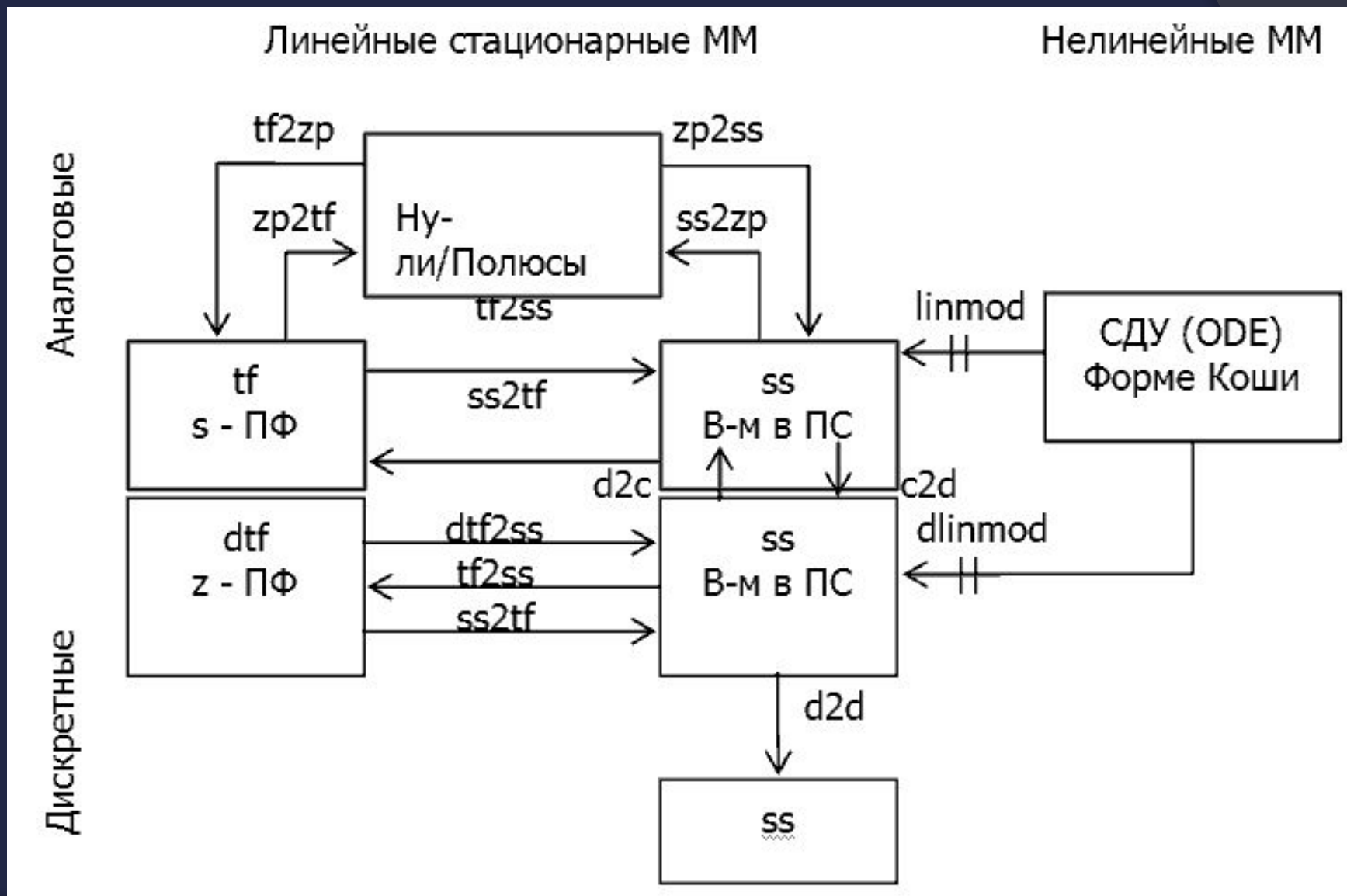
Задание ДС во вход-выходной (ВВ) форме записи

- В классической теории управления для описания ММ ДС используется понятие передаточных функций (ПФ), которые представляют собой дробно-рациональную функцию [2]

$$W(p) = \frac{b_m \cdot p^m + b_{m-1} \cdot p^{m-1} + \dots + b_1 \cdot p + b_0}{a_n \cdot p^n + a_{n-1} \cdot p^{n-1} + \dots + a_1 \cdot p + a_0} \quad (2)$$

- Для ввода ПФ коэффициенты числителя и знаменателя необходимо задавать в виде вектора (в порядке убывания степени).
- NUM=[b_m b_{m-1} ... b_1 b_0]
- DEN=[a_m a_{m-1} ... a_1 a_0]

Анализ математических моделей динамических систем



Преобразование ПФ:

- $[NUMc, DENc]=tfchk(NUM, DEN)$ – проверяет на соответствие порядки числителя и знаменателя, возвращает эквивалентную ПФ с равными порядками (отсутствующие коэффициенты заполняются нулями) или выдает сообщение об ошибке;
- $[Z, P, K]=tf2zp(NUM, DEN)$ – находит нули, полюсы и коэффициент передачи (приведенный);
- $[NUM, DEN]=zp2tf(Z, P, K)$ – обратное преобразование;
- $[A, B, C, D]=tf2ss(NUM, DEN)$ – преобразуем ПФ с одним входом в модель ПС в канонической форме управления. Для перехода также все матрицы необходимо развернуть на 180^0 : $A=rot90(A, 2)$, $B=rot90(B, 2)$ и т. д.;
- $abcdchk(A, B, C, D)$ – проверяет согласованность размерности матриц и в случае ошибки возвращает сообщение о ней;
- $[Wn, ksi]=damp(A)$ – вычисляет сопрягающие частоты и коэффициенты затухания ДС (A может быть либо системной матрицей, либо векторами NUM или DEN, либо векторами корней Z или P);

Построение частотных характеристик:

- $[Mod, Fi] = bode(A, B, C, D, u_i, w)$,
- $[Mod, Fi] = bode(NUM, DEN, w)$ – возвращает вектор амплитуды и фазы (u_i номер входа в МВ ММ). Для построения ЛЧХ вектор w должен в логарифмическом масштабе содержать значения частот в рад/с.
- Построение фазовой частотной характеристики для неминимально-фазовых звеньев выше 1-го порядка с использованием данной функции выполняется неправильно.
- Для коррекции фазовой характеристики имеются следующие характеристики:
- $Fik = fixphase(Fi)$ или $Fik = add360(Fi)$ – устраняет разрыв фазы от -180^0 к 180^0 . Фаза Fi задается в градусах;
- $Fik = addtwopi(Fi)$ – то же для фазы, заданной в радианах.

Вычисление переходных процессов:

- Y – вектор переходного процесса, t – вектор изменения времени,
- X – вектор состояния (необязательный параметр, может быть опущен)
- $[Y,X]=\text{impulse}(A,B,C,D,u_i,t)$,
- $Y=\text{impulse}(\text{NUM},\text{DEN},t)$ – весовая функция,
- $[Y,X]=\text{step}(A,B,C,D,u_i,t)$,
- $Y=\text{step}(\text{NUM},\text{DEN},t)$ – реакция на единичное ступенчатое воздействие,
- $[Y,X]=\text{lsim}(A,B,C,D,U,t,X0)$,
- $Y=\text{lsim}(\text{NUM},\text{DEN},U,t)$ – реакция на произвольно заданное вектором U внешним воздействием ($X0$ – начальное условие).

Анализ управляемости:

- $Q = \text{ctrb}(A, B)$ – вычисляет матрицу управляемости Q ,
- $[A_b, B_b, C_b, T] = \text{ctrbf}(A, B, C, \text{TOL})$ – выделение полностью управляемого подпространства, для частично управляемой системы. (TOL – допуск, необязательный параметр, T – матрица преобразования, $A_b = T \cdot A \cdot T^{-1}$, $B_b = T \cdot B$, $C_b = C \cdot T^{-1}$).

$$A_b = \begin{bmatrix} A_{nc} & 0 \\ A_{21} & A_c \end{bmatrix}, \quad B_b = \begin{bmatrix} 0 \\ B_c \end{bmatrix}, \quad C_b = [C_{nc} \quad C_c]$$

- Индекс nc – неуправляемая часть, c – управляемая.
- Выделение неуправляемой части:
- $nc = \text{length}(A_b) - \text{rang}(Q)$; $A_{nc} = A_b(1:nc, 1:nc)$.
- $\text{eig}(A_{nc})$ - собственные значения неуправляемой части системы.

Анализ наблюдаемости:

- $R = \text{obsv}(A, C)$ – вычисляет матрицу наблюдаемости R ,
- $[Ab, Bb, Cb, T] = \text{obsvf}(A, B, C, \text{TOL})$ – выделение полностью наблюдаемого подпространства, для частично наблюдаемой системы. (TOL – допуск, необязательный параметр, T – матрица преобразования, $Ab = T \cdot A \cdot T^{-1}$, $Bb = T \cdot B$, $Cb = C \cdot T^{-1}$).

$$Ab = \begin{bmatrix} A_{no} & A_{12} \\ 0 & A_o \end{bmatrix}, \quad Bb = \begin{bmatrix} B_{no} \\ B_o \end{bmatrix}, \quad Cb = \begin{bmatrix} 0 & C_o \end{bmatrix}$$

- Индекс no – ненаблюдаемая часть, o – наблюдаемая.
- Выделение ненаблюдаемой части:
- $no = \text{length}(Ab) - \text{rang}(R)$; $A_{no} = Ab(1:no, 1:no)$.
- $\text{eig}(A_{no})$ - собственные значения ненаблюдаемой части системы.

Построение графиков:

- `plot(X,Y,S)` – построение графика функции $Y(X)$ с заданием типа линии с помощью S (необязательный параметр);
- `plot3(X,Y,Z,S)` – построение трехмерного графика функции $Z(X,Y)$ с заданием типа линии с помощью S (необязательный параметр);
- `loglog(X,Y,S)` – аналогична предыдущей команде, но для задания логарифмического масштаба по X и Y ;
- `semilogx(X,Y,S)` или `semilogy(X,Y,S)` – строит график в логарифмическом масштабе по оси X или Y соответственно;
- `subplot(m,n,k)` – разбивает окно на m окон по горизонтали и n окон по вертикали, а k – номер окна, в котором будет выводиться текущий график;