



*Российский государственный университет
нефти и газа им. И.М. Губкина*

Кафедра Информатики

Дисциплина: Информатика

Преподаватель:

К.Т.Н., ДОЦЕНТ

Коротаев

Александр Фёдорович

Элементарные математические функции



$\exp(x)$ – e^x

$\text{abs}(x)$ – модуль x

$\text{sqrt}(x)$ – корень квадратный из x

$\log(x)$ – натуральный логарифм x ($\log 2$, $\log 10$)

$\text{round}(x)$ – округление до ближайшего целого

$\text{floor}(A)$ – округление до целого снизу

$\text{ceil}(A)$ – округление до целого сверху

$\text{sign}(A)$ – знак A

$\sin(x)$, $\cos(x)$, $\tan(x)$, ... – тригонометрические

$\text{asin}(x)$, $\text{acos}(x)$, $\text{acot}(x)$, – обратные тригонометрические

Все сведения: **Help / Function Browser (Shift+F1)**



М-функция

Система MatLab даёт возможность создавать свои функции, записывая их в **m-файл** с помощью встроенного редактора.

Имеются отличия **m-функции** от **m-файла-сценария**:

- Функция может компилироваться целиком с последующим размещением исполняемого кода в памяти
- Функция может иметь локальные переменные, размещаемые в собственной рабочей области
- В функции могут быть входные и выходные параметры

Синтаксис определения и вызова М-функций



Текст **М- функции** должен начинаться с **заголовка**, после которого следует **тело функции**. Заголовок имеет следующий вид:

```
function [Ret1,Ret2,...]=fName(par1,par2,...)
```

где **Ret1,Ret2,...** — выходные параметры,
par1,par2,... — входные параметры

Указанное в заголовке **имя функции** должно совпадать с **именем файла**, расширение имени файла должно быть **m**.

Тело функции состоит из инструкций на **m-языке**, с помощью которых вычисляются возвращаемые значения .

Вызывать функцию из командного окна (или другого m-файла) можно, задав её имя с фактическими параметрами.



Пример

```
function ret1=myFunc(x1,x2)
% myFunc calculates x1*x2
% plus  x1^2 +2x1 +3
%-----
ret1=x1.*x2 +AnotherFunc(x1);
```

Изнутри данного m-файла могут вызываться другие функции

```
function ret2 = AnotherFunc(y)
ret2=y.*y + 2*y +3;
```

Обращение к функции

```
>> res=myFunc(1,2)
res = 8
```

В качестве фактических параметров можно использовать переменные и выражения

Справка, содержащаяся в **комментариях(%)**, выдается по команде

```
>>help myFunc
```

```
myFunc calculates x1*x2
plus  x1^2 +2x1 +3
```



Особенности графики системы MATLAB

Для визуализации вычислений используются графические объекты, создаваемые на принципах **дескрипторной (описательной) графики**

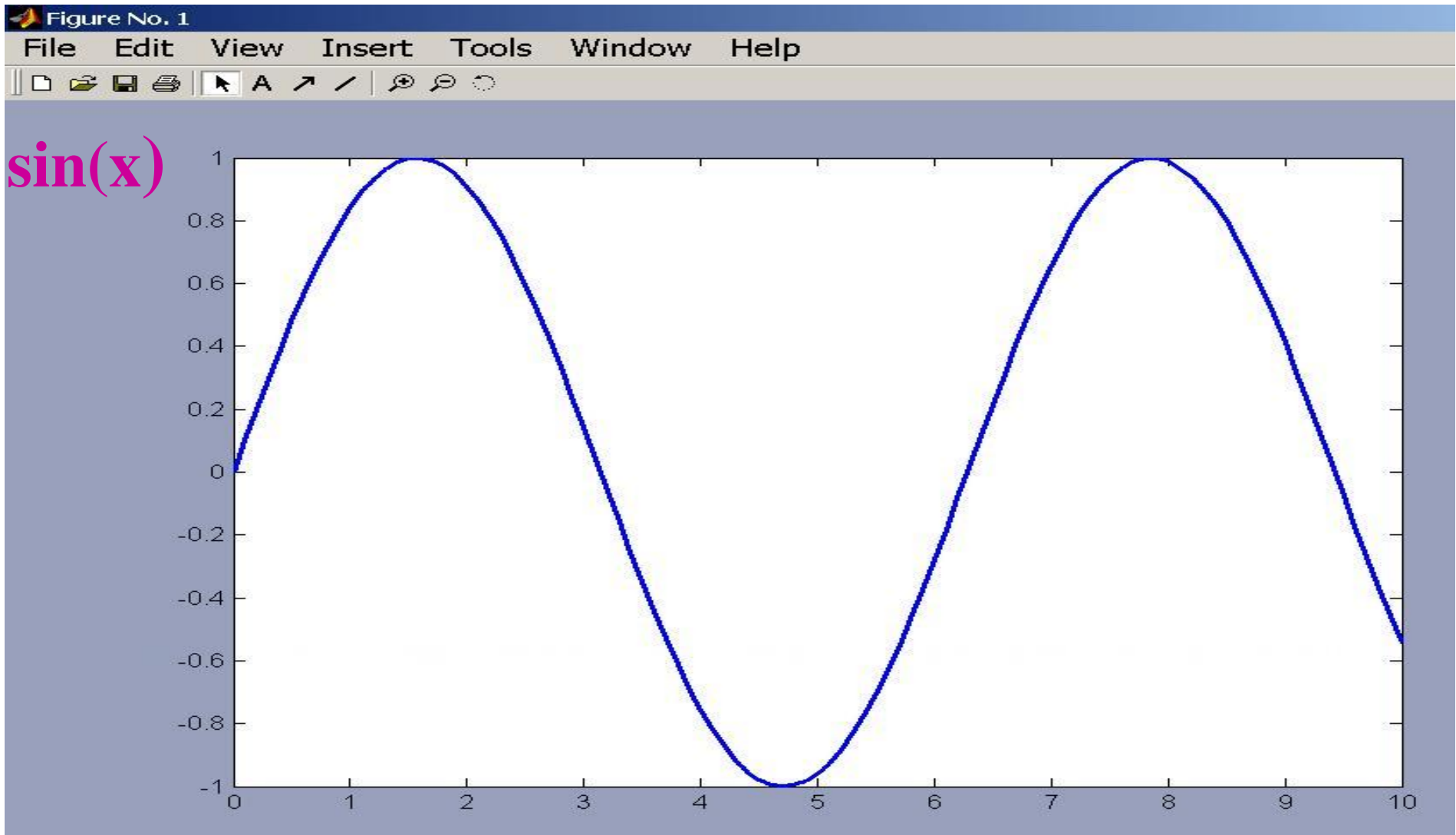
Иерархическая структура объектов дескрипторной графики строится на принципах объектно-ориентированного программирования и состоит из 4-х уровней, связанных по принципу «**родитель-потомок**»:

- ✓ root (корень) — первичный объект, соответствующий экрану компьютера
- ✓ figure (рисунок) — объект создания графического окна
- ✓ координатные оси, меню, панели инструментов и т.д.
- ✓ растровые изображения, линии, тексты и т.д.

Большинство команд **высокоуровневой графики** автоматически устанавливает свойства графических объектов и обеспечивает воспроизведение графики в нужных системе координат, палитре цветов, масштабе и т. д. (т.е ориентировано на конечного пользователя-**непрограммиста**)



Основы графической визуализации вычислений



Построение графика функций одной переменной



Для построения графика функции **$\sin(x)$** на интервале $[0 \ 10]$ зададим шаг изменения аргумента **0.1** :

введём команду вычисления вектора

$x=0:0.1:10$,

а затем команду построения графика

$\text{plot}(x,\sin(x))$

График строится как кусочно-линейная функция по узловым точкам.

Другой вариант команды построения графика

$\text{fplot}('sin(x)',[0 \ 10])$



Построение в одном окне графиков нескольких функций

Можно воспользоваться функцией вида

`plot(a1,f1,a2,f2,a3,f3,...)`

где **`a1, a2, a3,...`** — векторы аргументов функций

`f1, f2, f3,...` — векторы значений функций

Чтобы построить в одном окне графики `sin` и `cos`:

`plot(x,sin(x),x,cos(x))`

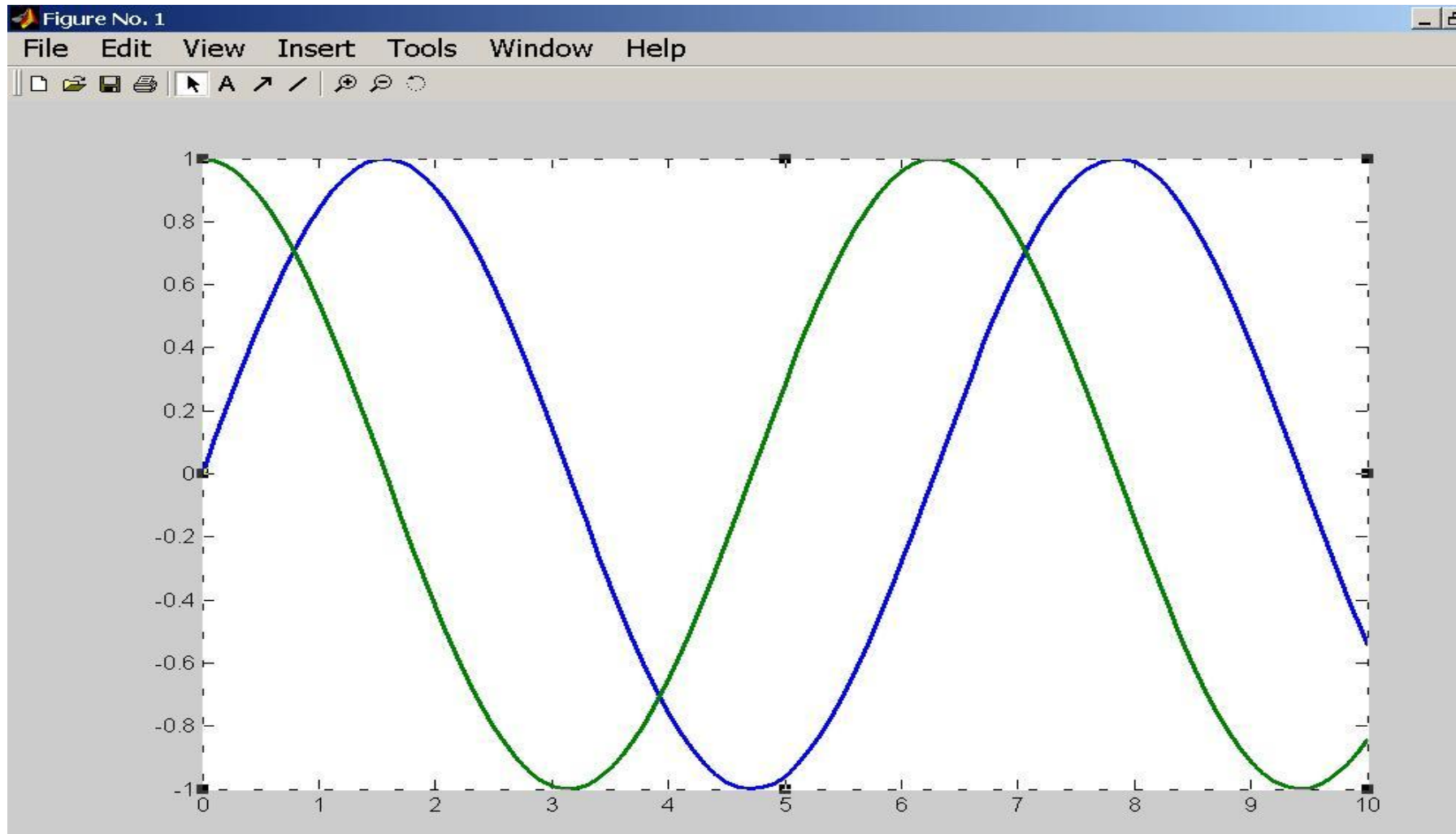
Другой вариант:

`plot(x,sin(x)) ; hold on; plot(x,cos(x))`

`hold on` позволяет удерживать содержимое
графического окна



`plot(x,sin(x),x,cos(x))`



Разбиение графического окна



subplot(m,n,k) – позволяет разбить область вывода графической информации на несколько подобластей, в каждую из которых можно вывести графики различных функций

m-равно числу строк подобластей,

n- числу колонок подобластей,

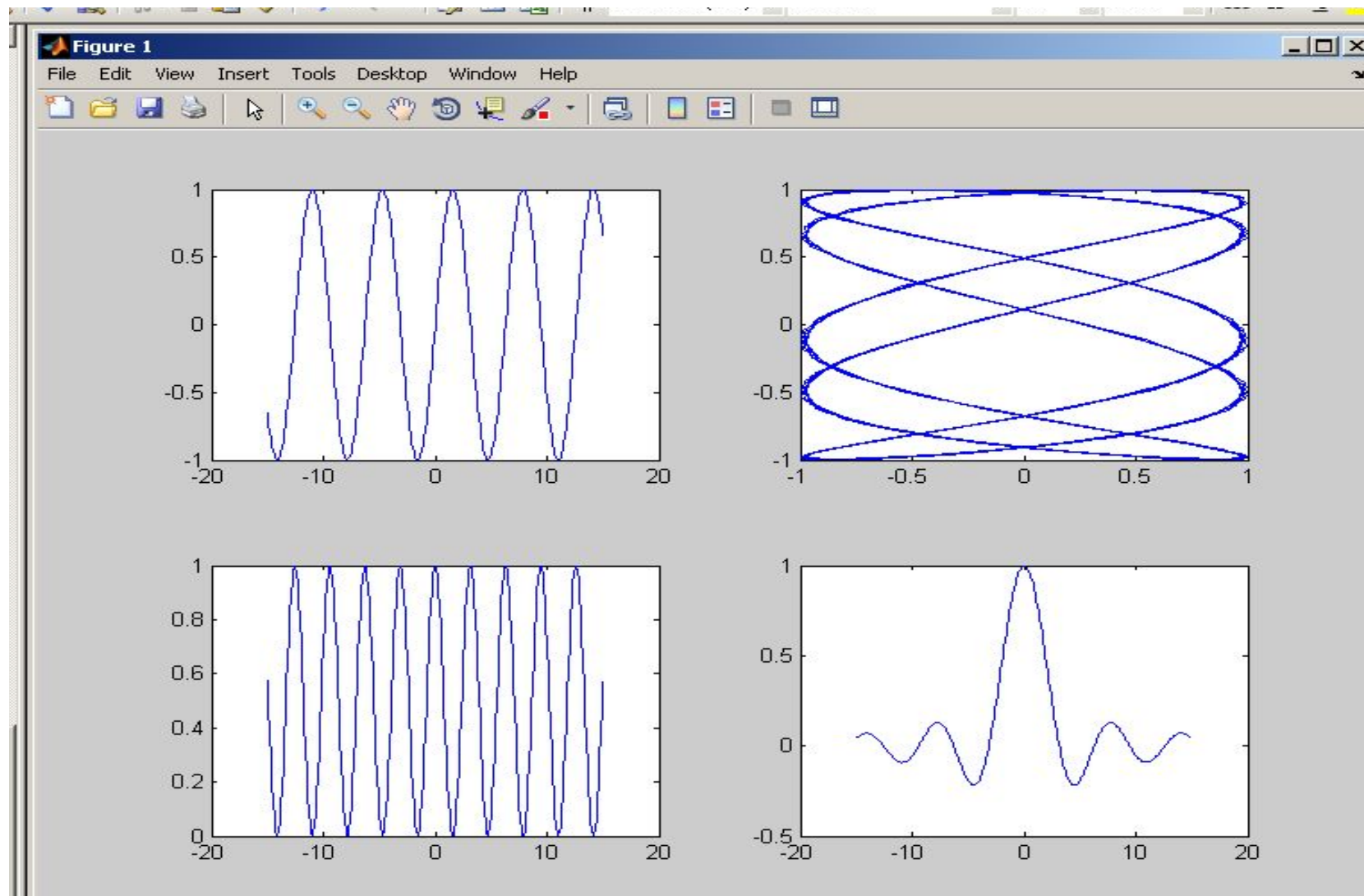
k - номеру подобласти , в которую выводится график (***подобласти нумеруются слева направо по строкам***)

Пример

```
x=-15:0.1:15;  
subplot(2,2,1),plot(x,sin(x))  
subplot(2,2,2),plot(sin(5*x),cos(2*x+0.2))  
subplot(2,2,3),plot(x,cos(x).^2)  
subplot(2,2,4),plot(x,sin(x)./x)
```



Разбиение графического окна





Характеристики линии

В общем случае функция построения графика:

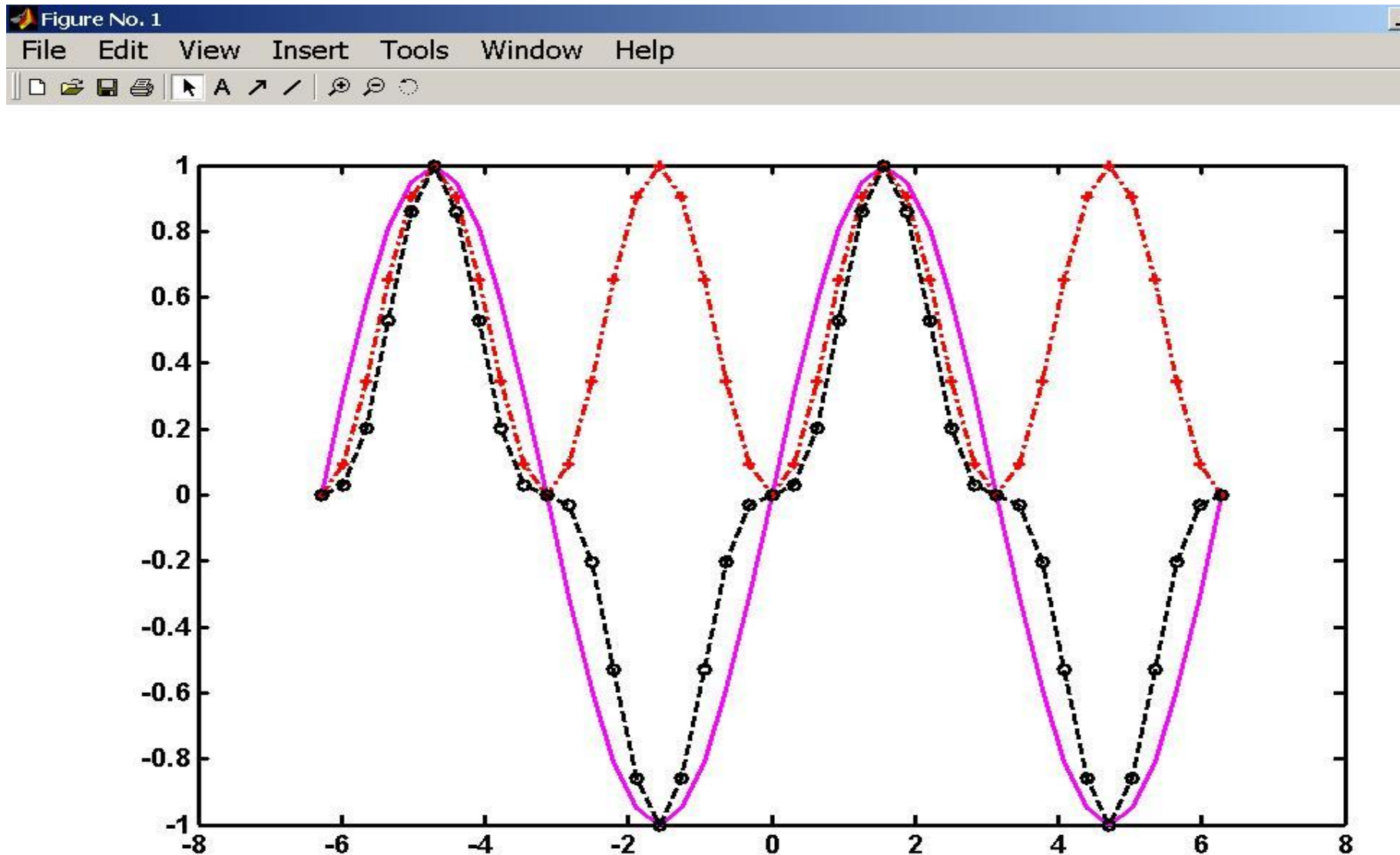
plot(x,y,S)

где строковая константа **S** задаёт тип линии

Цвет		Тип линии		Тип точки	
Y	Желтый				
M	Фиолетовый	-	Сплошная	.	точка
C	Голубой	:	Пунктирная	o	кружок
R	Красный				
G	Зеленый	-.	Штрих- пунктирная	x	крест
B	Синий	--	Штриховая	+	плюс
W	Белый				
K	Черный			*	звёздочка



`plot(x,y1,'-m', x,y2,'-.+r', x,y3,'--ok')`

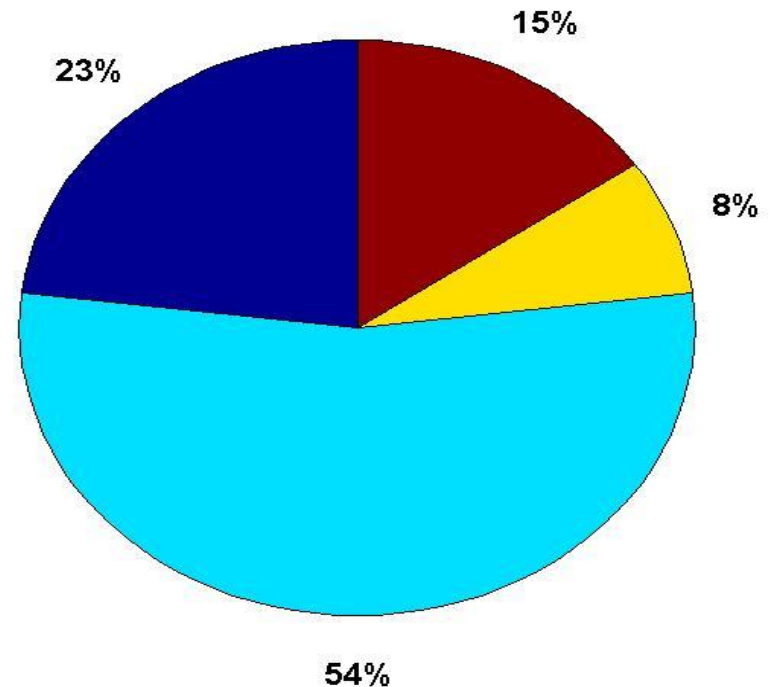
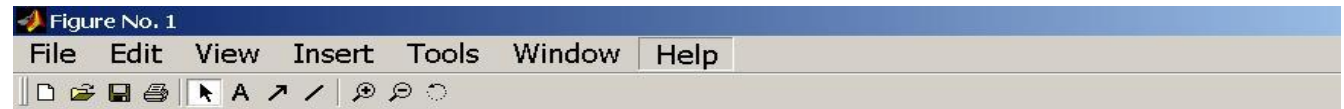




Круговые диаграммы

Круговая диаграмма (функция **pie(x)**) показывает, какой процент от суммы всех элементов составляет конкретный элемент. **pie3** - объёмная диаграмма

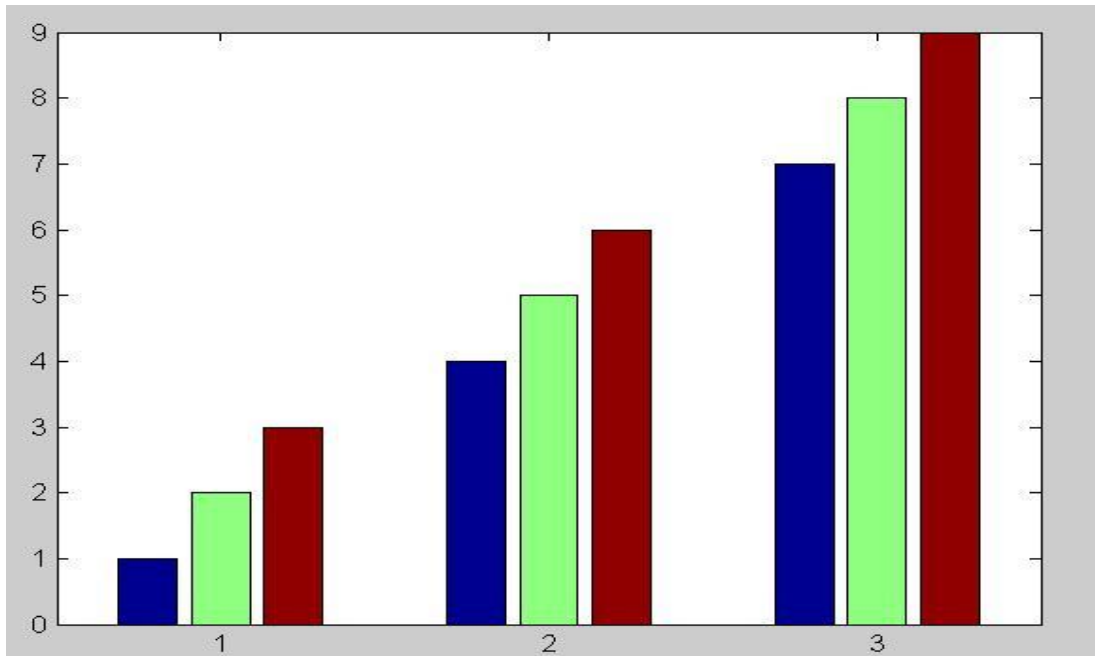
```
>> x=[3,7,1,2];  
>> pie(x)
```



Столбцовые диаграммы



Если Y – матрица, имеющая m строк и n столбцов, то **bar(Y)** строит m групп n вертикальных столбиков по значениям элементов матрицы Y



```
>> y=[1 2 3; 4 5 6; 7 8 9];  
>> bar(y)
```

Что будет,
если Y – вектор?

barh (Y) – столбики будут расположены горизонтально

bar(Y,width) — задаёт ширину столбиков

По умолчанию **width = 0.8**

При **width > 1** столбики в группах перекрываются

bar3 и **bar3h** строят 3-мерные bar-диаграммы



Построение гистограмм

hist(Y,M) - строит гистограмму в виде столбцовой диаграммы, характеризующей число попаданий значений элементов вектора **Y** в каждый из **M** интервалов.

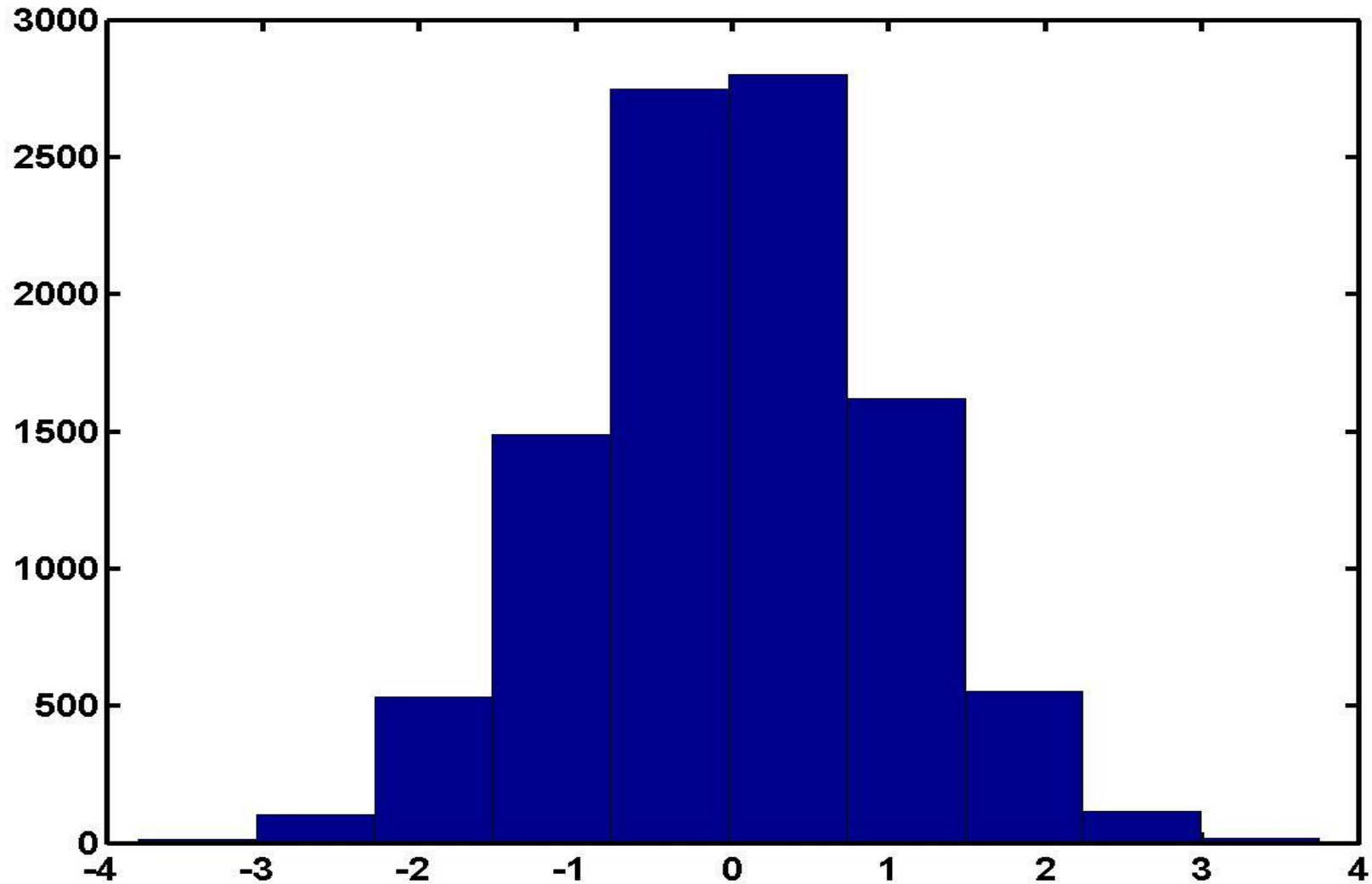
hist(Y) – по умолчанию **M= 10** интервалов.

Для того, чтобы посчитать число попаданий элементов вектора **Y** в заданные интервалы, данную функцию нужно использовать с выходным параметром

N=hist(Y,M)



`x=randn(1,10000); hist(x)`



Оформление графиков



title('string ') — установка **титульной надписи**, заданной строковой константой **'string'**

Функции установки **названий осей x, y и z** :

xlabel('string') ; ylabel('string') ; zlabel('string')

Размещение текста в произвольном месте рисунка :

- **text(x,y, 'string')** — выводит текст в точку с координатами **(x,y)**
- **text(x,y,z, 'string')** — выводит текст в точку с координатами **(x,y,z)**
- **gtext('string')** — выводит текст, который можно установить мышью в нужное место графика

Установка диапазонов координат :

- **axis([XMIN XMAX YMIN YMAX])** — по осям **x** и **y** для текущего двумерного графика



Вывод легенды

`legend('string1','string2', ...,Pos)` — помещает легенду в место, определённое параметром Pos:

Pos = 0 — выбирается автоматически

Pos = 1 — верхний правый угол

Pos = 2 — верхний левый угол

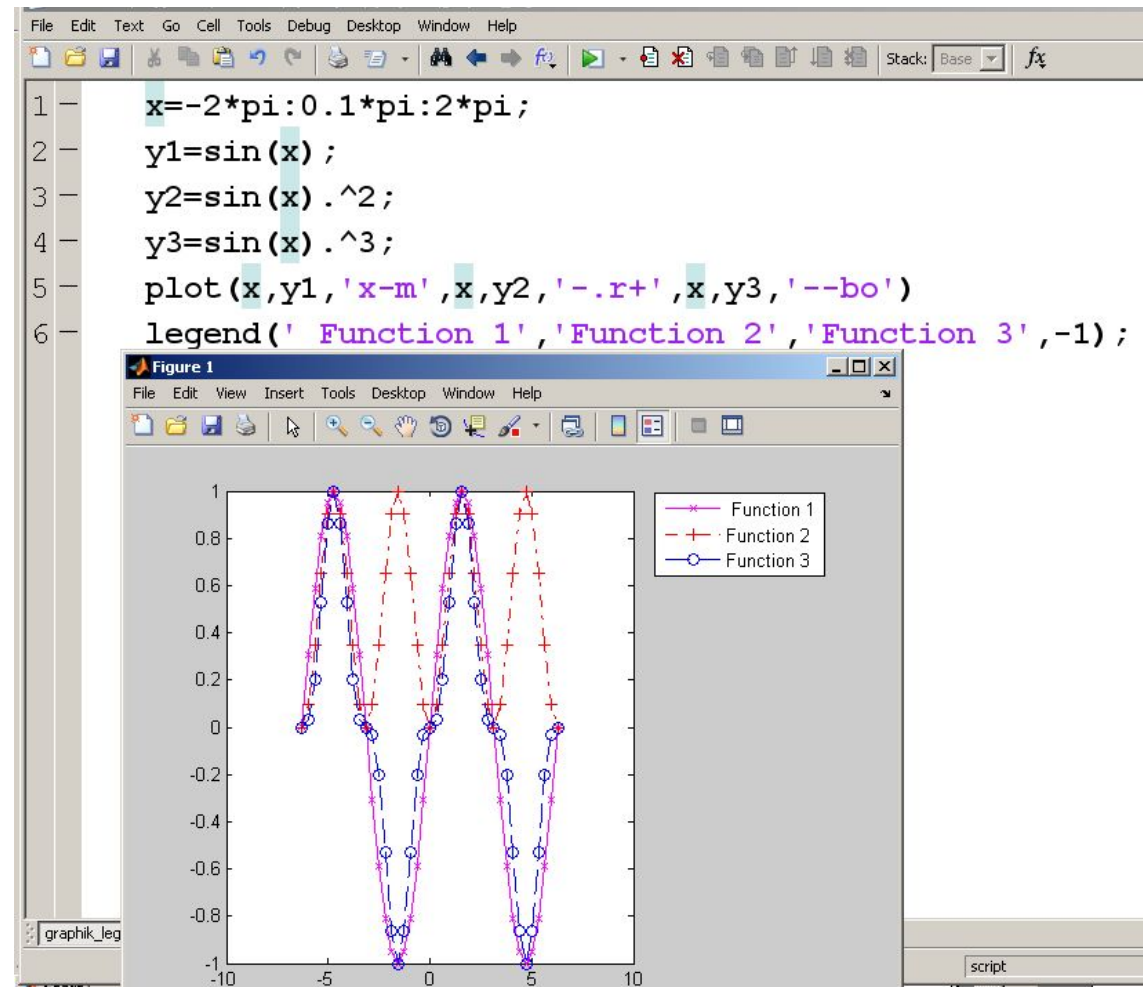
Pos = 3 — нижний левый угол

Pos = 4 — нижний правый угол

Pos = -1 — справа от графика

Можно и без Pos.

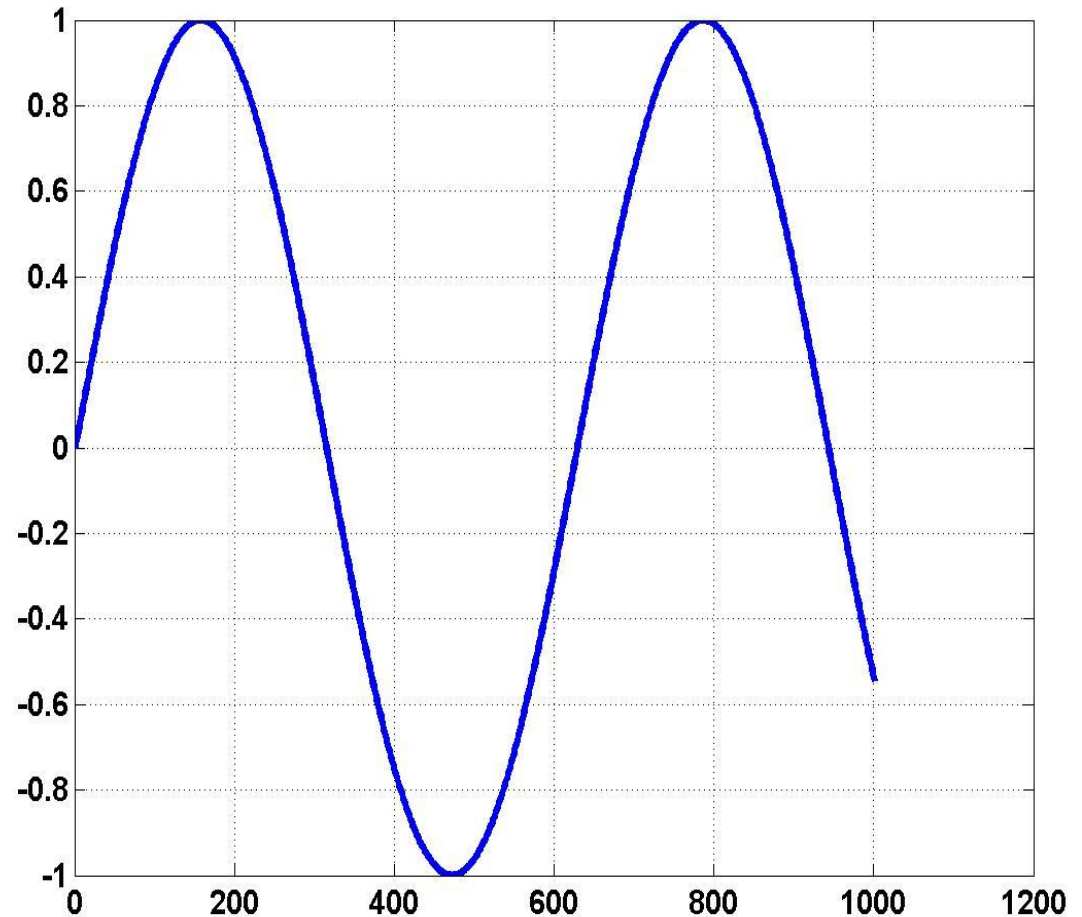
С помощью мыши
легенду легко
перетащить в любое
другое место





Вывод координатной сетки

grid on — добавляет сетку к текущему графику;
grid off — отключает сетку;
grid — последовательно производит включение и отключение сетки



Дополнительные параметры форматирования графиков



(..., 'LineWidth', 5) – ширина линии 5

(..., 'FontSize', 14) – размер шрифта 14

(..., 'MarkerSize', 8) – размер маркера 8

Все рассмотренные ранее функции сами раскрывают окно **figure 1**

Заккрыть текущее окно можно командой **close**

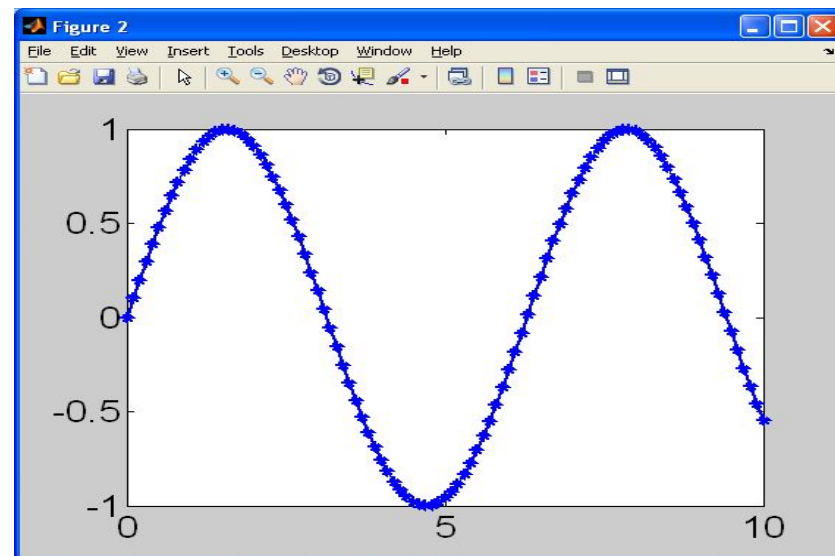
Команда **figure(2)** раскрывает второе окно и т.д. Заккрыть – **close 2**

Все окна сразу закрываются командой **close ALL**

С помощью команды **get** можно вывести значения параметров графика, а командой **set** можно изменить эти значения

Пример

```
>> figure(2)
>> x=0:0.1:10;
>> y=sin(x);
>> hPlot=plot(x,y,'-*');
>> set(hPlot,'LineWidth',2,'MarkerSize',8);
>> get(hPlot)
```





Интерактивное редактирование графиков

В меню окна построенного графика
опции **Edit**, **Insert** и **Tools** позволяют легко
управлять параметрами графиков



Можно также воспользоваться возможностями
панели инструментов

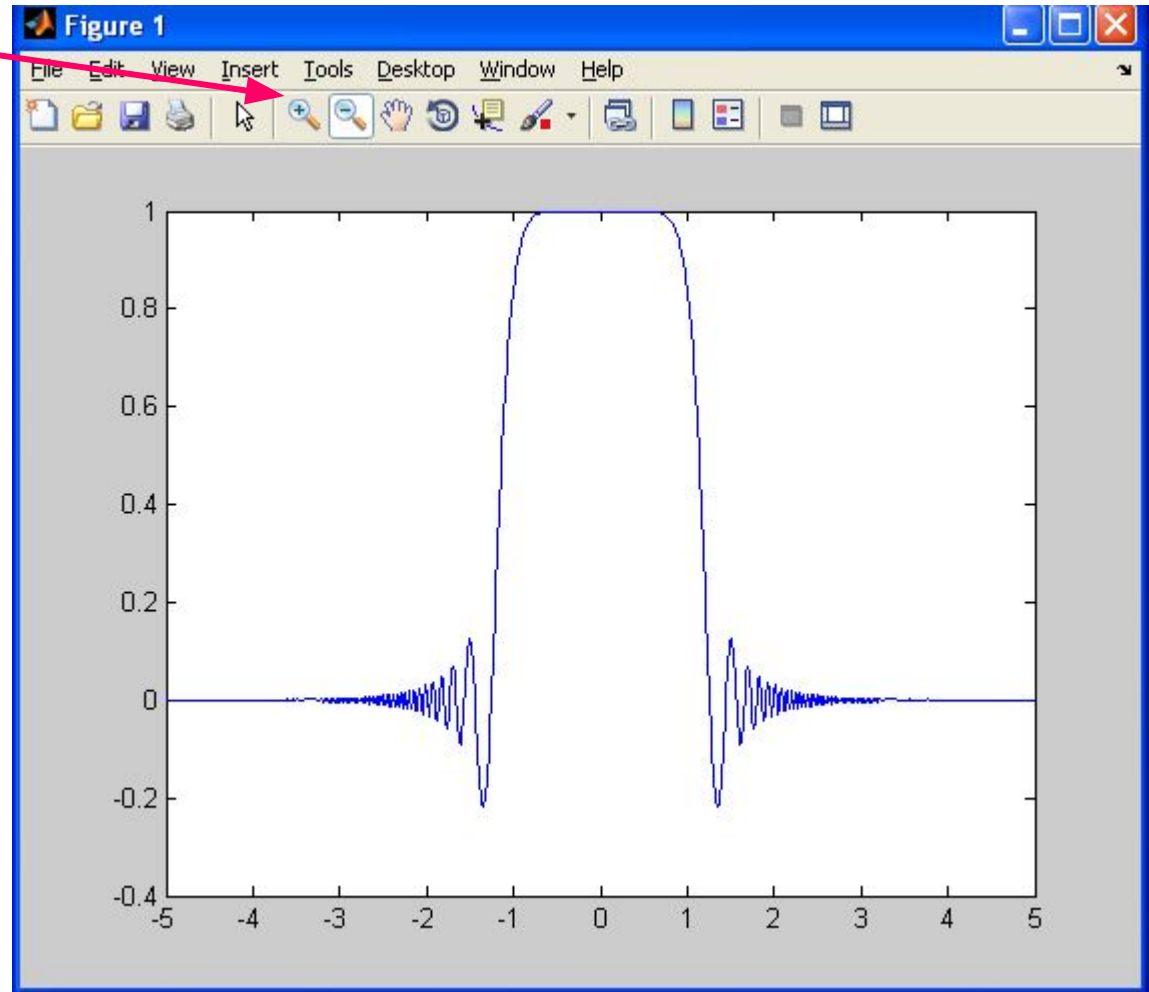


Изменение масштаба графика

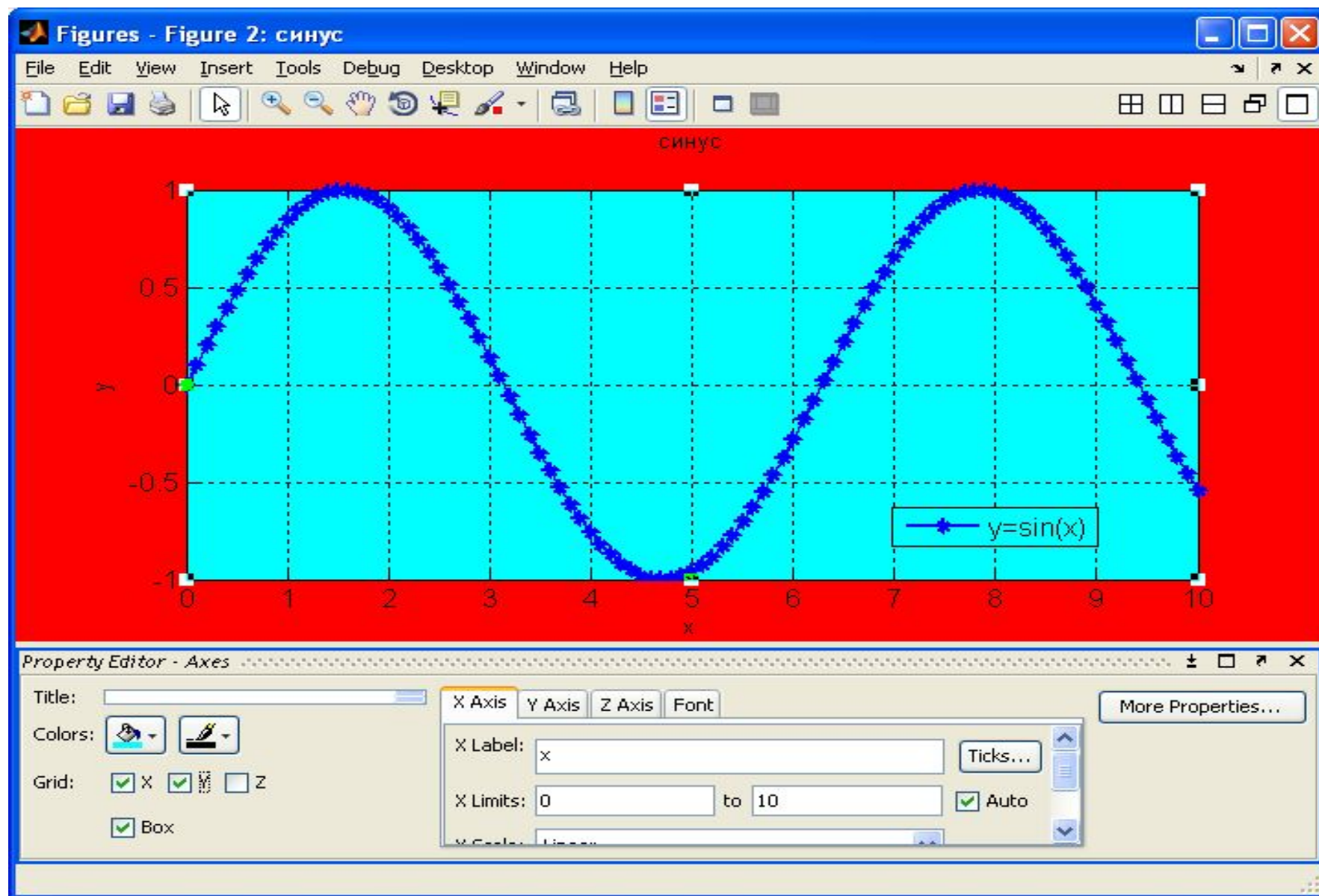
Инструмент **Лупа**

- + увеличивает вдвое
- уменьшает вдвое

Перемещая мышь при нажатой левой клавише, можно выделить **область детализации**; после отпускания клавиши эта область отобразится во **всё окно**



Настройка свойств графика



Вычисление нулей функции одной переменной



(функция *fzero*)

Задача: решить нелинейное уравнения вида $f(x) = 0$

В случае $f_1(x) = f_2(x) \implies f_1(x) - f_2(x) = 0$

Сводится к нахождению значений аргумента x , при котором значение функции $f(x)$ равно нулю.

$x = \text{fzero}(@\text{fun}, x_0)$ — задаётся имя функции **fun** и начальное значение аргумента x_0 ; возвращается уточненное значение аргумента, которое близко к точке, где функция меняет знак, или равно **NaN**, если такая точка не найдена.

$x = \text{fzero}(@\text{fun}, [x_1 \ x_2])$ — возвращает значение аргумента при задании такого интервала поиска $[x_1 \ x_2]$, при котором функция **меняет знак один раз**.

Функция должна быть задана в файле с именем **fun.m**

Вычисление нулей функции одной переменной (функция `fsolve`)



Функция **fzero** рассматривает ноль как точку, где график функции **пересекает ось x, а не касается ее**. Поэтому при поиске нулей нелинейных функций следует предварительно строить их графики для нахождения интервалов, в пределах которых нули находятся (отделение корней).

Кроме того, можно использовать функцию **fsolve**, которая при решении системы нелинейных уравнений вида **$f(x)=0$** ищет не только точки пересечения с осью x , но и точки касания.

»**fsolve(@fun, x0)**

где **x0** – начальное приближение

Вычисление нулей функции одной переменной (функция `solve`)



Можно также использовать функцию `solve`, которая выдаёт результат в символьной форме, а если такого нет, то она позволяет получить решение в численном виде

```
>> res=solve('sin(x)+x-1=0')
```

```
res =
```

```
.510973
```

`solve` не требует информации о начальном значении или области изоляции. При трансцендентных уравнениях может находить не все корни

Поиск минимума функции одной переменной



x=fminbnd(hfunc,x0,x1,option) или

[x,fval] = fminbnd(hfunc,x0,x1,option)

где **hfunc** - описатель функции,

x0 и **x1** задают интервал поиска,

x, **fval** – значения аргумента и функции в точке минимума

Описатель может быть в одном из видов:

@myfun **'(x-3)^2'** **f = inline('x.^3-2*x-5')**

А вызов, соответственно, в виде :

>> [x,val]= fminbnd(@myfun,-5,5)

>>[x,val]= fminbnd('(x-3)^2',-5,5)

>> [x,val]= fminbnd(f,-5,5)

Как будем искать максимум? ?



Решение систем нелинейных уравнений

fsolve (FUN, x0, options) ,

где **FUN** – система уравнений, сохраненная в m-файле

x0 – начальное приближение

Пример: $x_1 x_2 + x_3 = 6.5$; $x_1 x_2^4 + x_3 = 167$; $x_1 x_2^6 + x_3 = 1470$

function F=myfun(x)

F=[x(1)*x(2)+x(3)-6.5 x(1)*x(2)^4+x(3)-167 x(1)*x(2)^6+x(3)-1470];

>> X=fsolve(@myfun,[1 1 1])

X =

2.1512 2.9678 0.1157

Решение системы с помощью функции solve



```
>> Y=solve('3*x+y-z=3','-5*x+3*y+4*z=1', 'x+y+z=0.5')
```

```
Y =
```

```
  x: [1x1 sym]
```

```
  y: [1x1 sym]
```

```
  z: [1x1 sym]
```

```
>> Y.x
```

```
ans =
```

```
-0.10714285714285714285714285714286
```

Можно воспользоваться функцией

vpa(Y.x, n) , где **x** — неизвестное, **n** — число значащих цифр в ответе

```
>> vpa(Y.x,5)
```

```
ans =
```

```
-.10714
```



Хронометраж работы программ

Система **MatLab** позволяет определять время выполнения фрагментов программы

Функция **tic** запускает секундомер

Функция **toc** останавливает секундомер и выдаёт время счёта в секундах

```
A=100*rand(1000); B=10*rand(1,1000); B=B';
```

```
tic
```

```
X=A\B;% метод исключения Гаусса
```

```
toc
```

```
tic
```

```
X=inv(A)*B;
```

```
toc
```

```
Elapsed time is 0.464645 seconds.
```

```
Elapsed time is 1.171958 seconds.
```


Сортировка

sort (A) — в случае одномерного массива A сортирует и возвращает элементы по возрастанию их значений; в случае двумерного массива происходит сортировка и возврат элементов каждого столбца. Допустимы вещественные, комплексные и строковые элементы.

[B,INDEX] = sort(A) — наряду с отсортированным массивом возвращает массив индексов INDEX. Он имеет размер `size(A)`.

sort(A,dim) — для матриц сортирует элементы по столбцам (`dim=1`) или по рядам (`dim=2`)