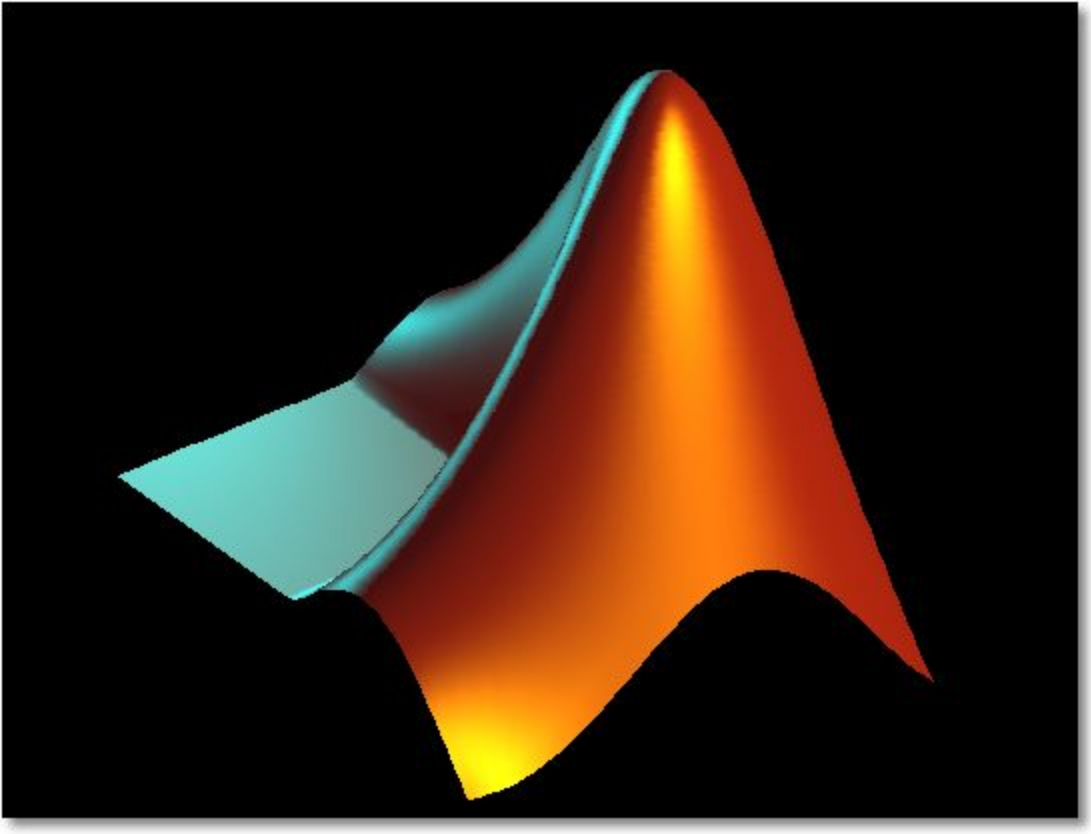


Лекция 4

Графические возможности Matlab

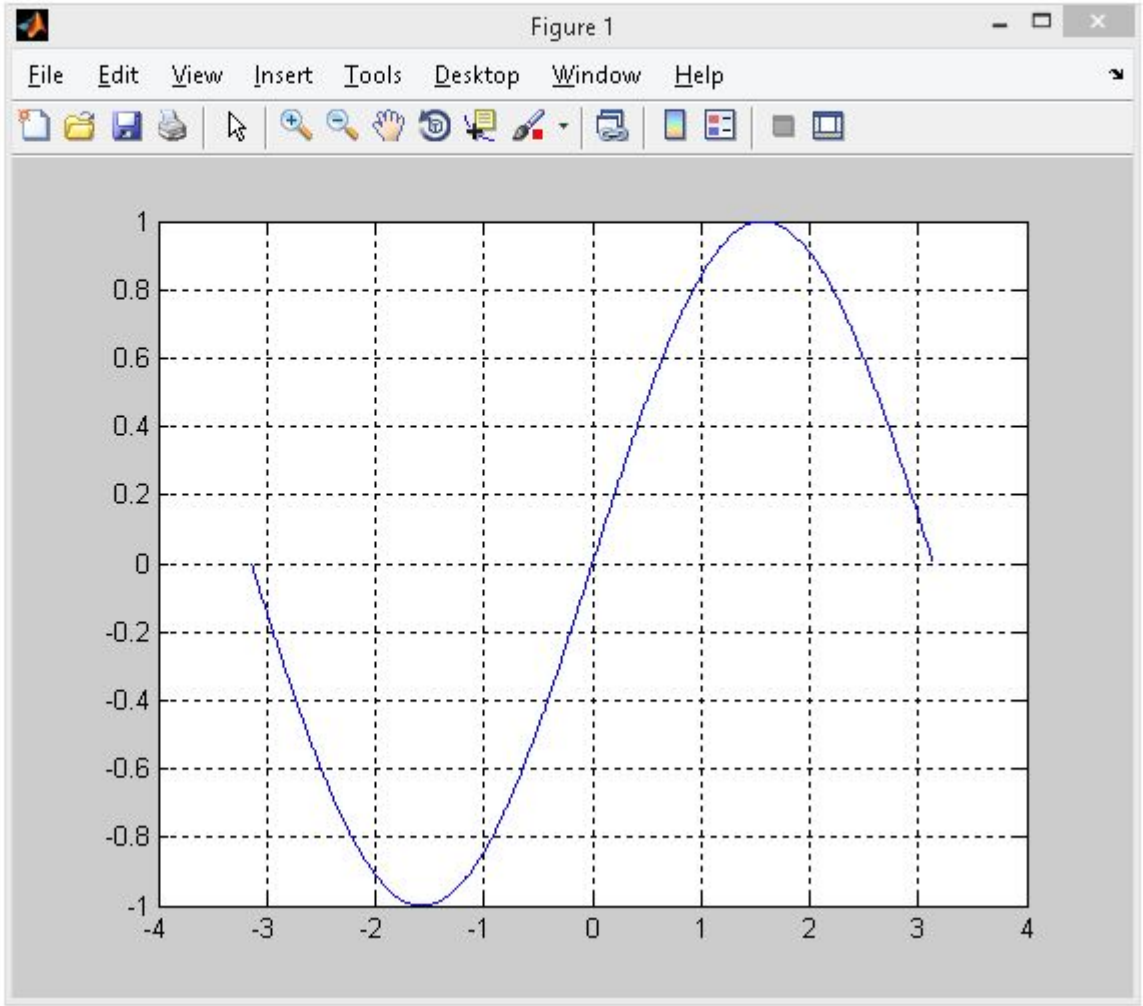
Графика в Matlab

- 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)

Построение графиков функции одной переменной

- 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.

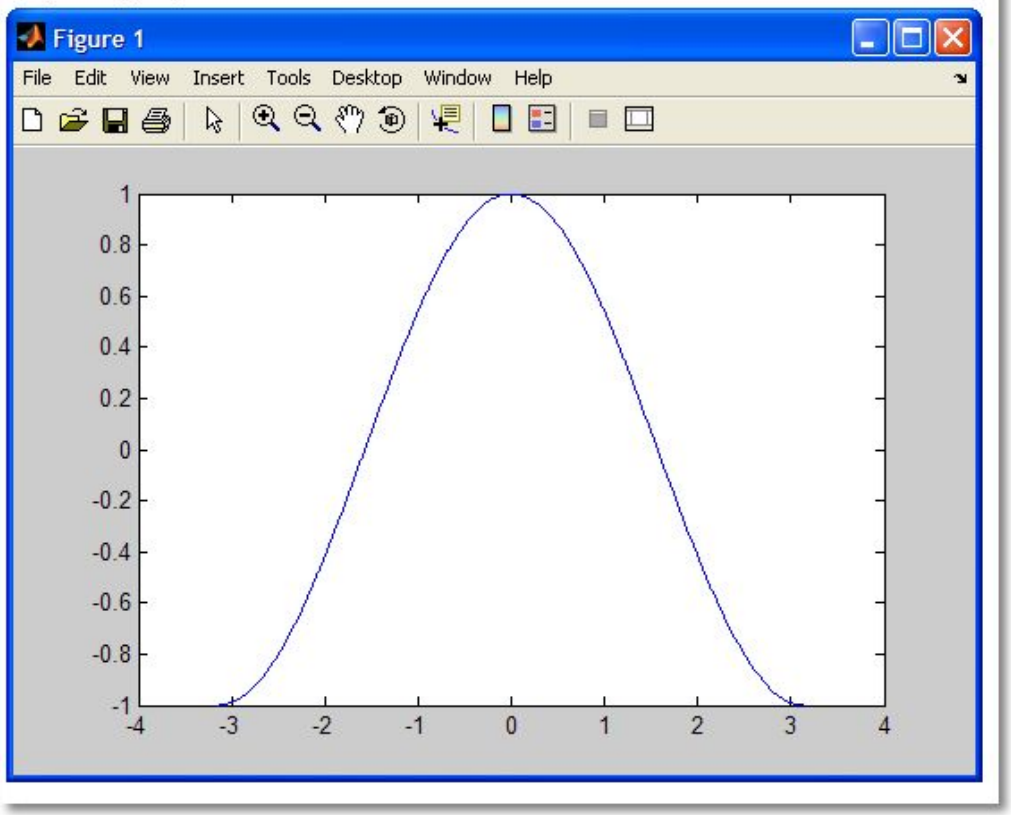
```
>> x=-pi:.01:pi;
>> y=sin(x);
>> plot(x,y)
>> grid on
>>
```



Построение второго графика

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

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

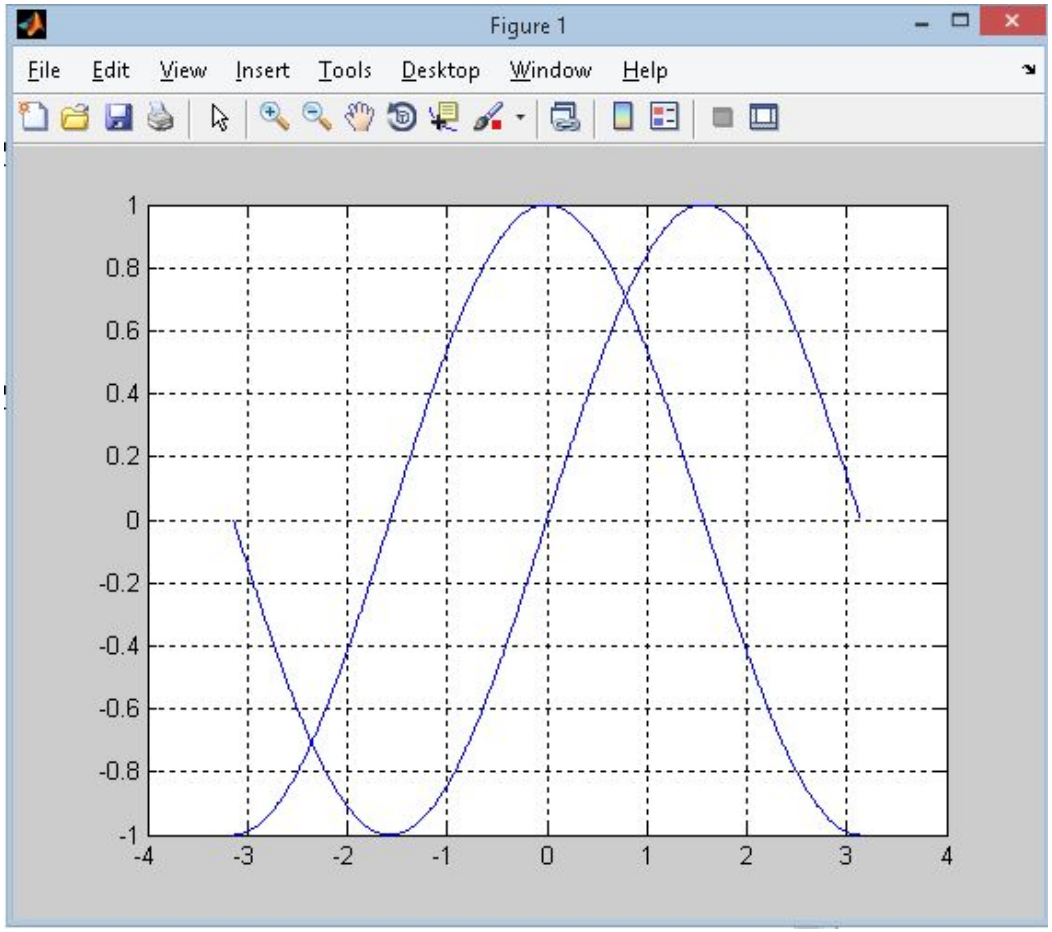


Построение двух графиков в одной системе координат

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

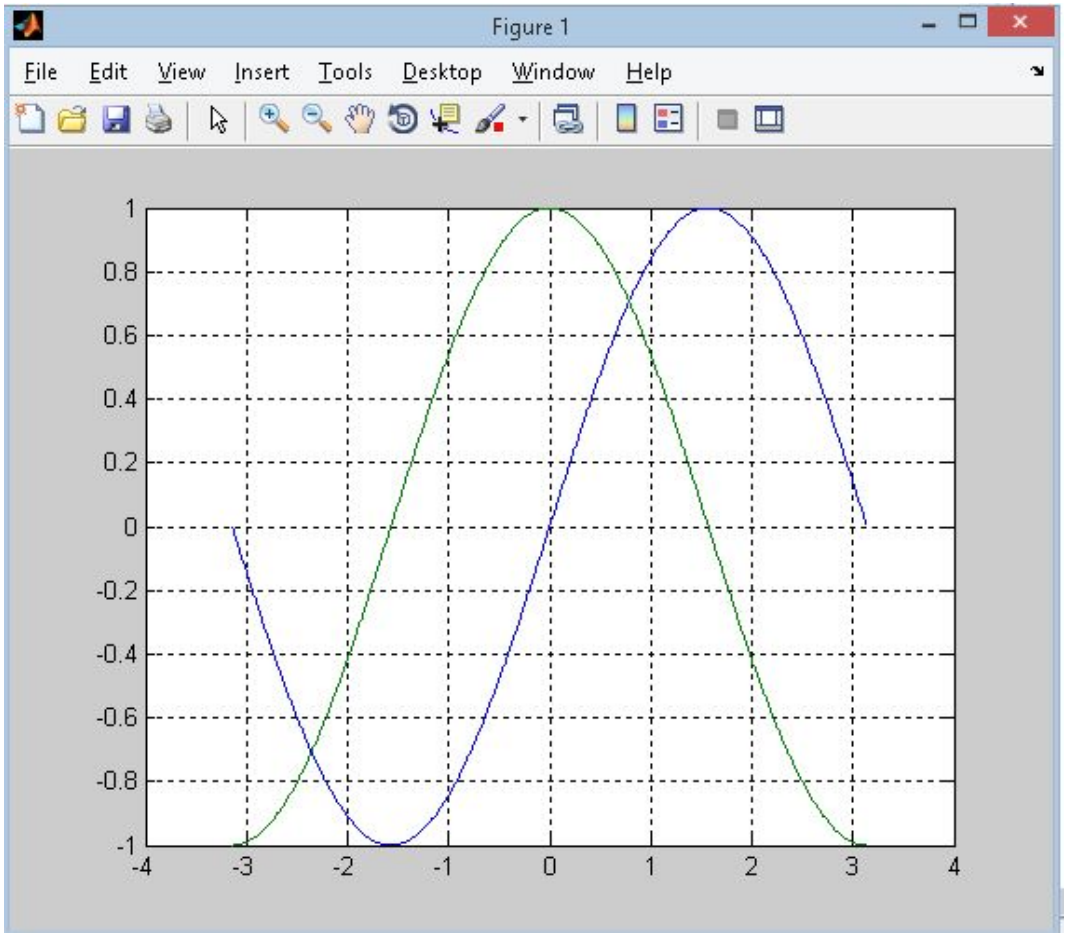
Закрепление графического окна

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



Дополнительные параметры команды *plot*

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



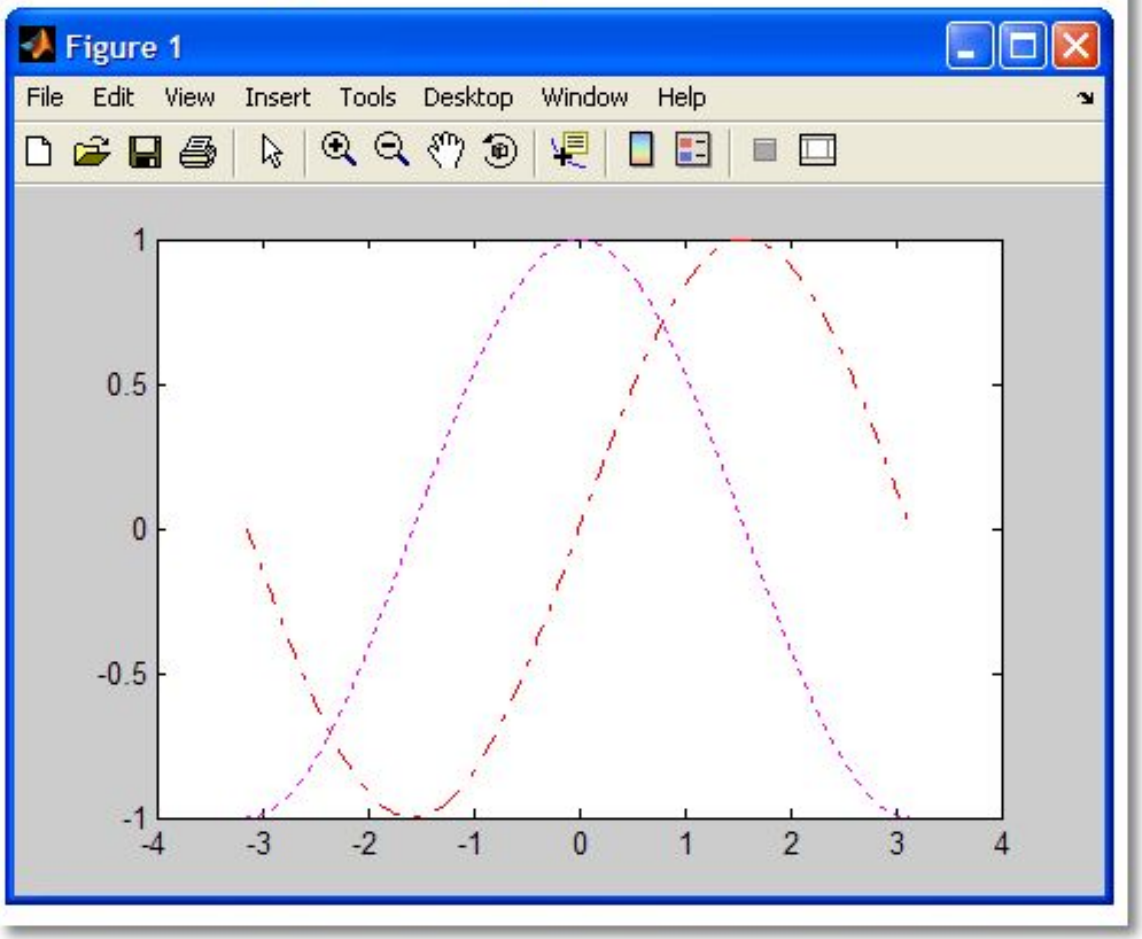
Дополнительные параметры команды ***plot***

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

Цвет линии		Тип точки		Тип линии	
Желтый	y	Точка	.	Сплошная	-
Фиолетовый	m	Окружность	0	Двойной пунктир	;
Голубой	c	Крест	x	Штрих-пунктир	-.
Красный	r	Плюс	+	Штриховая	--
Зеленый	g	Звездочка	*		
Синий	b	Квадрат	s		
Белый	w	Ромб	d		
Черный	k	Треугольник (вниз)	v		
		Треугольник (вверх)	^		
		Треугольник (влево)	<		
		Треугольник (вправо)	>		
		Пятиугольник	p		
		Шестиугольник	h		

Пример команды `plot`

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



Толщина линии указывается с помощью функции **set.**

Пример.

```
>>x = 0 : 0.1 : 3;
>>y = sin( x );
>>p = plot( x, y );
>>set( p, 'LineWidth', 7 );
```

В примере свойство 'LineWidth' (толщина линии), для которого задали новое значение 7 пикселей (по умолчанию - 0.5 пикселей).

Допустима запись:

```
>>plot(x,y,'LineWidth‘,7)
```

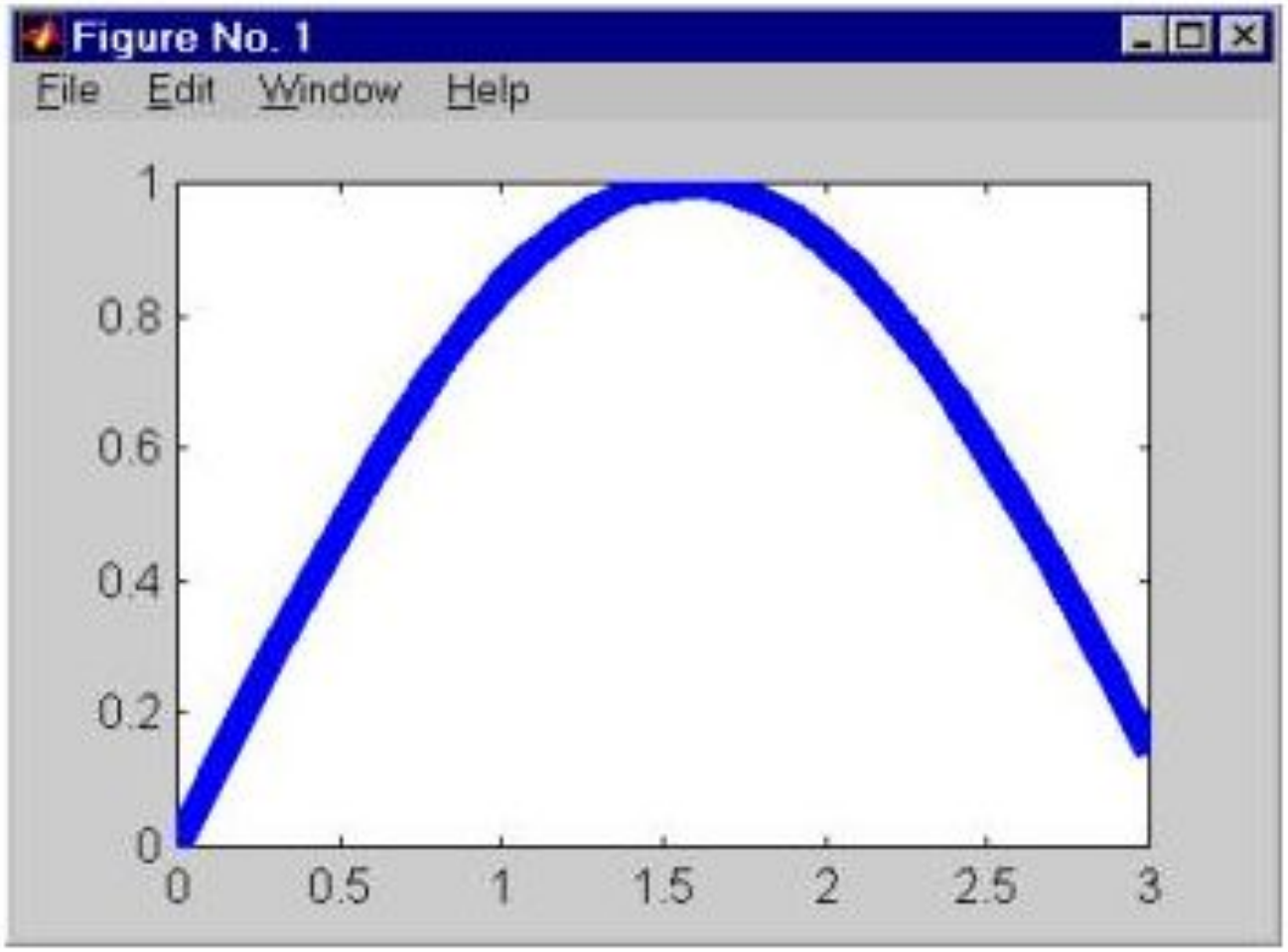
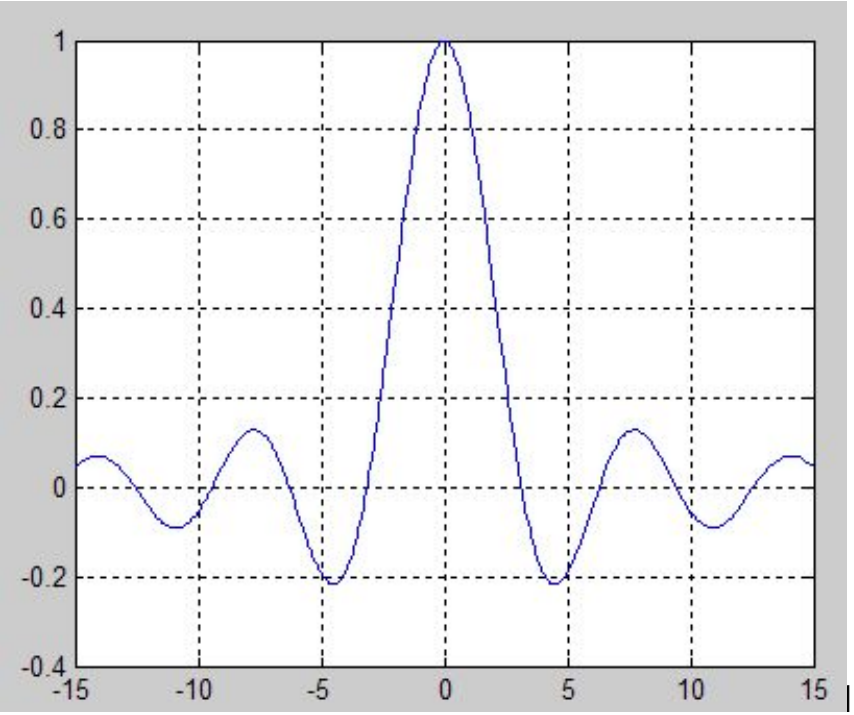


Figure No. 1
File Edit Window Help
1
0.8
0.6
0.4
0.2
0
0 0.5 1 1.5 2 2.5 3

Графическая функция ***fplot***

Используется для построения графиков и таких функций, как **sin(x)/x**, которые имеют устранимые неопределенности.

```
fplot('sin(x)/x', [-15,15]);
grid on
```

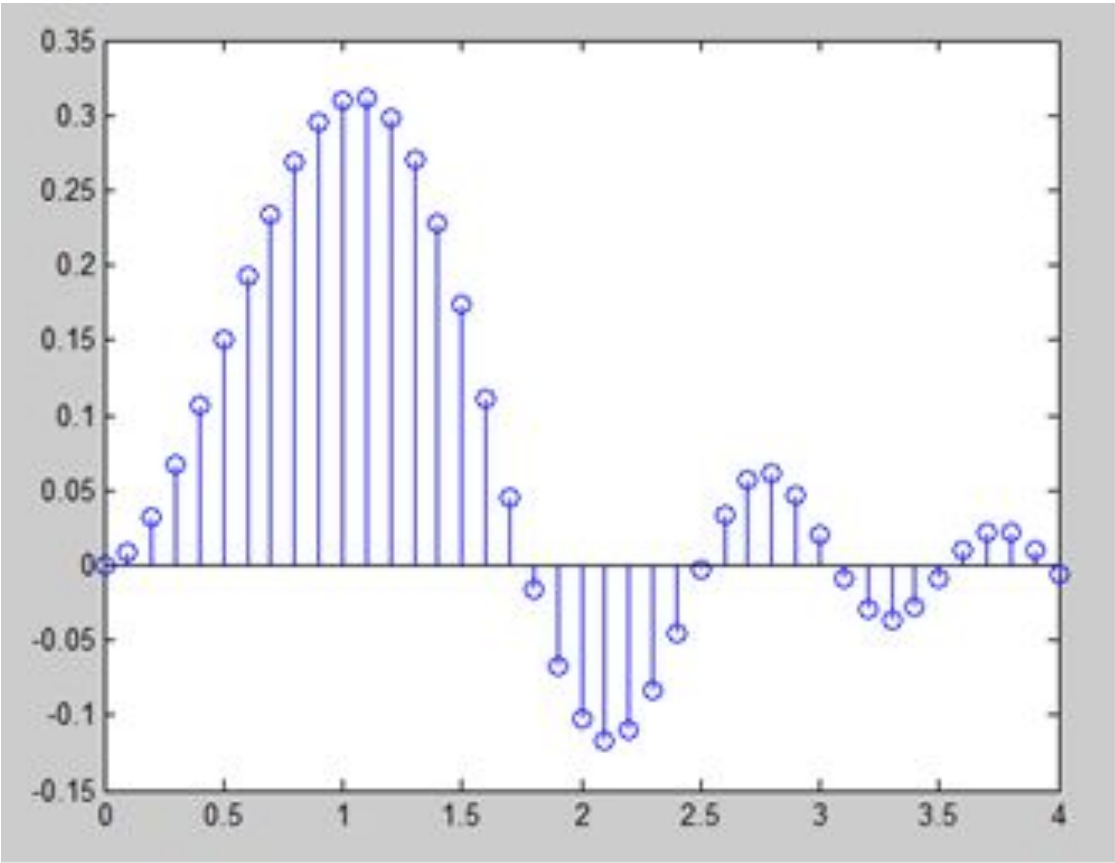


Построение дискретных графиков

Такой график применяется, например, при описании квантования сигналов. Каждый отсчет представляется вертикальной чертой, увенчанной кружком, причем высота черты соответствует ***y***- координате точки.

Для построения графика подобного вида используются команды ***stem(x,y):***

```
>>x = 0:0.1:4;
>>y = sin(x.^2).*exp(-x);
>>stem(x,y)
```



Построение лестничных графиков

Лестничные графики представляют собой ступеньки с огибающей, заданной в виде функции y(t). Они используются, например, для наглядного представления функции y(t), представленной результатами ряда измерений ее значений. При этом в промежутках между измерениями значения функции считаются постоянными и равными величине последнего результата измерения.

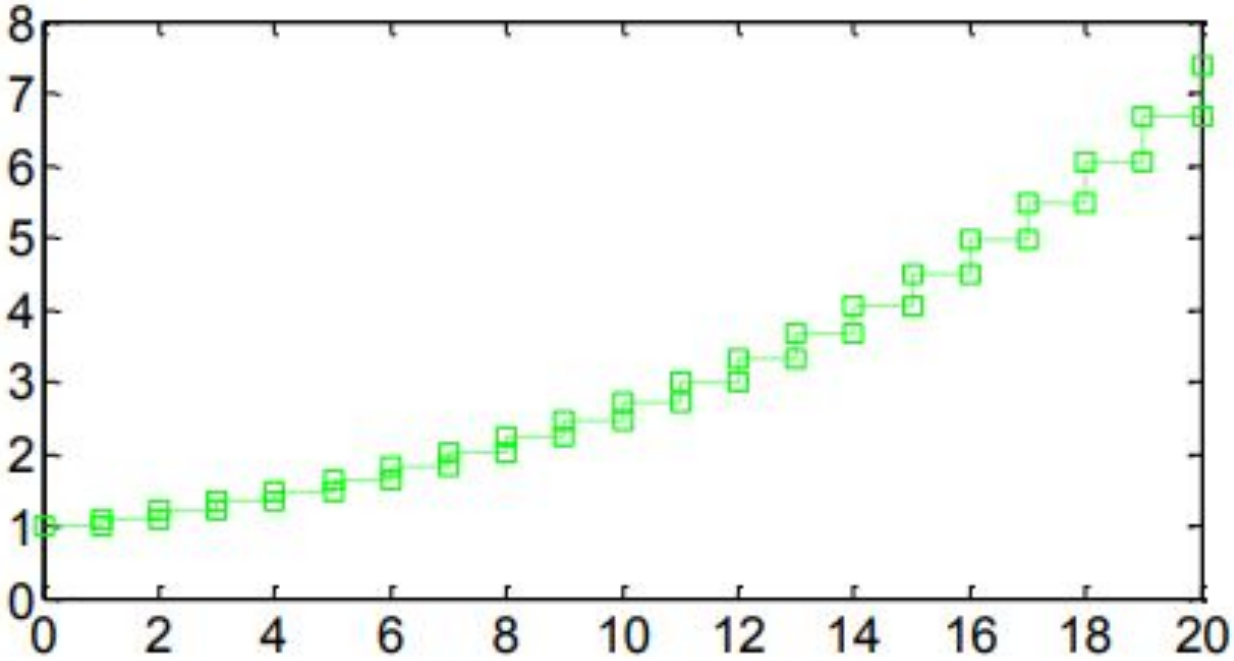
Пример.

Вариант 1

```
t=[0:20];
y=exp(0.1*t);
stairs(t,y,'gs:')
```

Вариант 2

```
t=[0:20];
stairs(t,exp(0.1*t),'gs:')
```



Графики с логарифмическим масштабом

Встречаются случаи, когда диапазон изменения функции настолько велик, что для ее графической визуализации приходится применять логарифмический и полулогарифмический масштаб.

Для построения графиков в логарифмическом масштабе служат следующие функции:

loglog(…) – логарифмический масштаб по обеим осям;

semilogx(…) – логарифмический масштаб по оси абсцисс;

semilogy(…) – логарифмический масштаб по оси ординат.

Аргументы этих функций (все, что стоит внутри скобок) формируются по тем же правилам, что и в функции **plot(…).**

Пример.

Построить в одном окне в полулогарифмическом масштабе с использованием функции semilogx(…) графики зависимостей

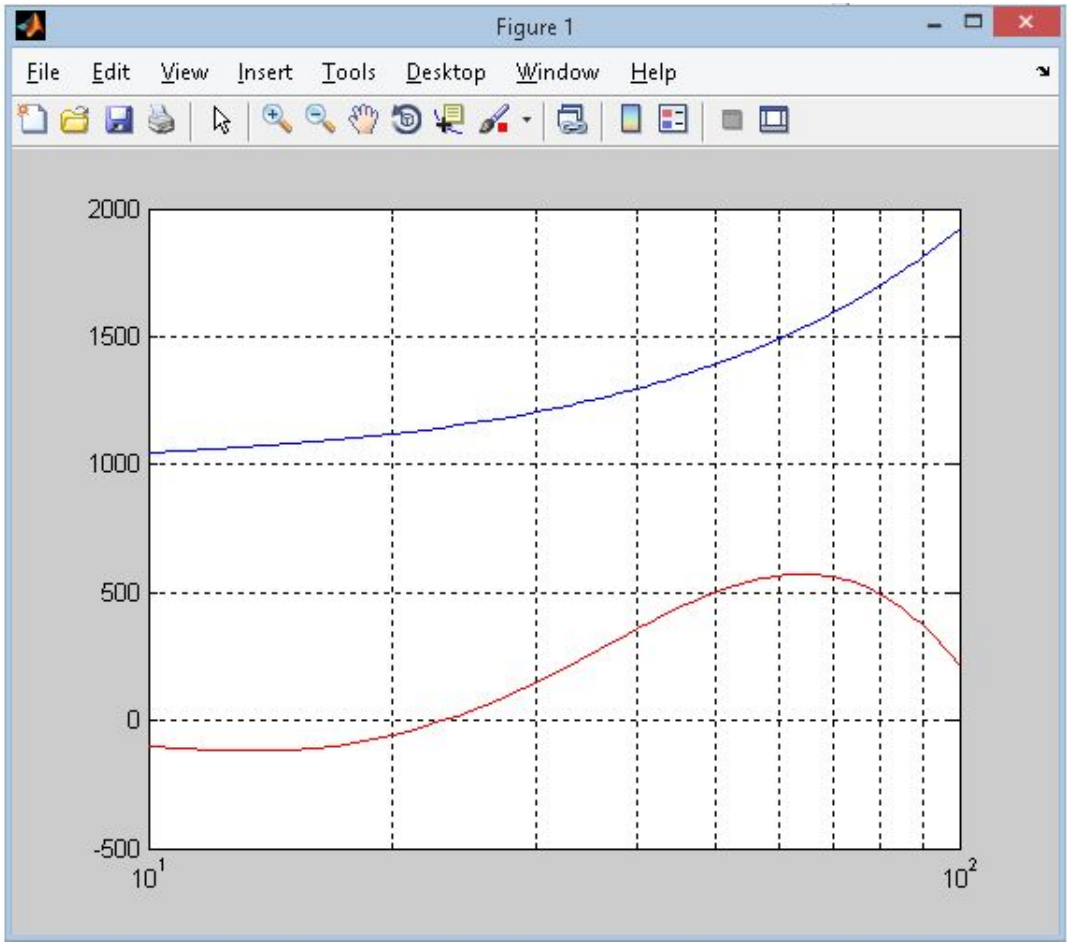
$$f(t) = t \cdot \ln(t^2) + 1000,$$

$$f(t) = 10t \cdot \sin(\ln(t^2))$$

при условии, что t меняется на интервале [10,1000] с шагом 1.

График функции g(t) выделить красным цветом.

```
>> t=10:100;
>> f=t.*log(t.^2)+1000;
>> g=10*t.*sin(log(t.^2));
>> semilogx(t,f,t,g,'r')
>> grid on
```



Построение нескольких графиков в одном окне в разных СК

Поверхность графического окна можно разделить на зоны, в каждой из которых выводить свой график

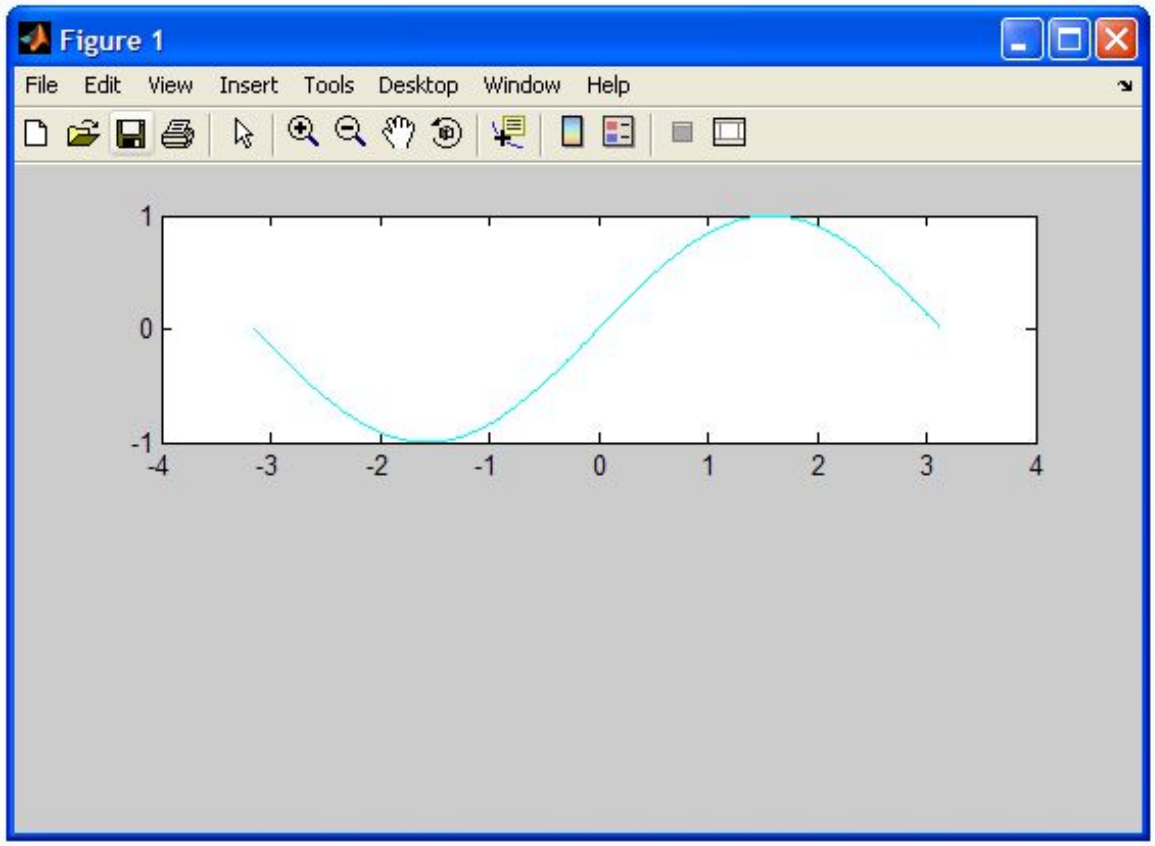
Для этого служит команда ***subplot(mnk)***

m и ***n*** определяют количество графических «подокон» по горизонтали и вертикали

k задаёт номер графического «подокна»

– порядок нумерации – по строкам

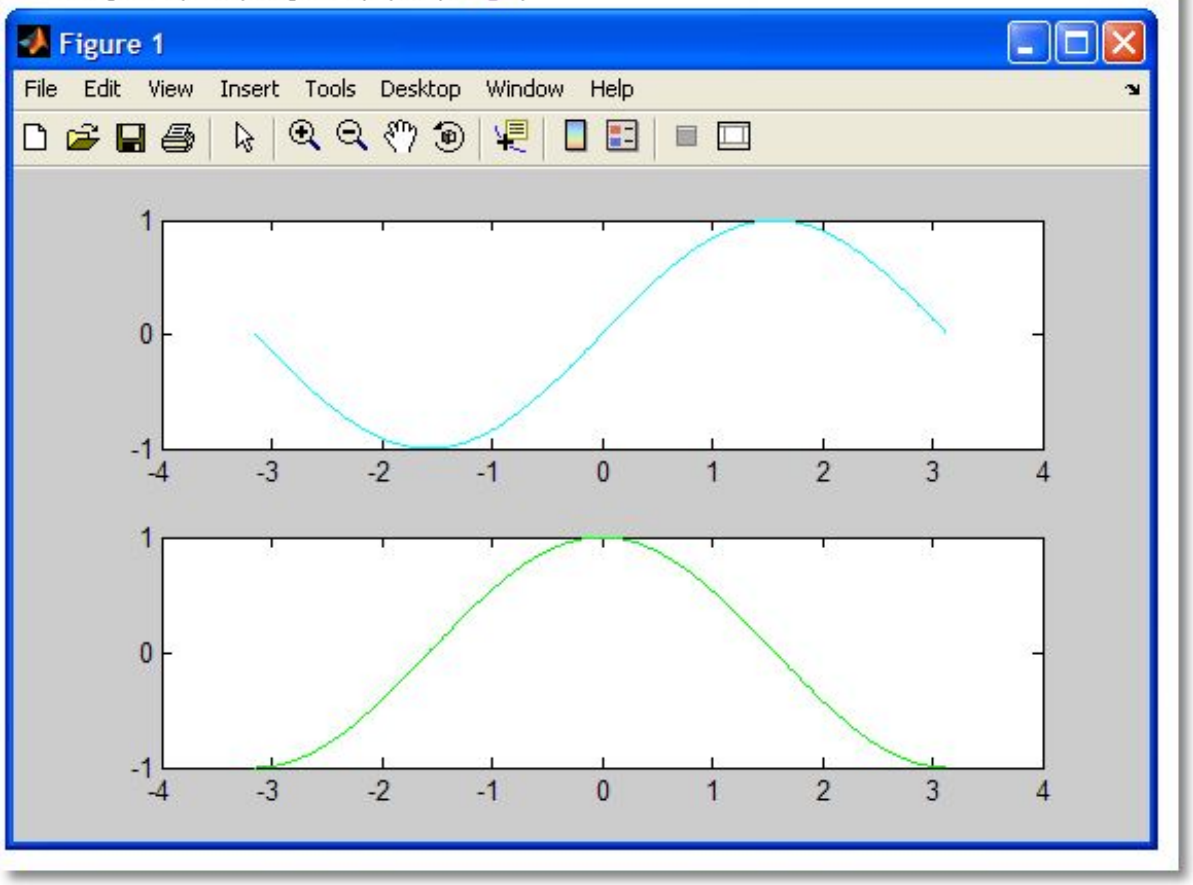
Первый *subplot*




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

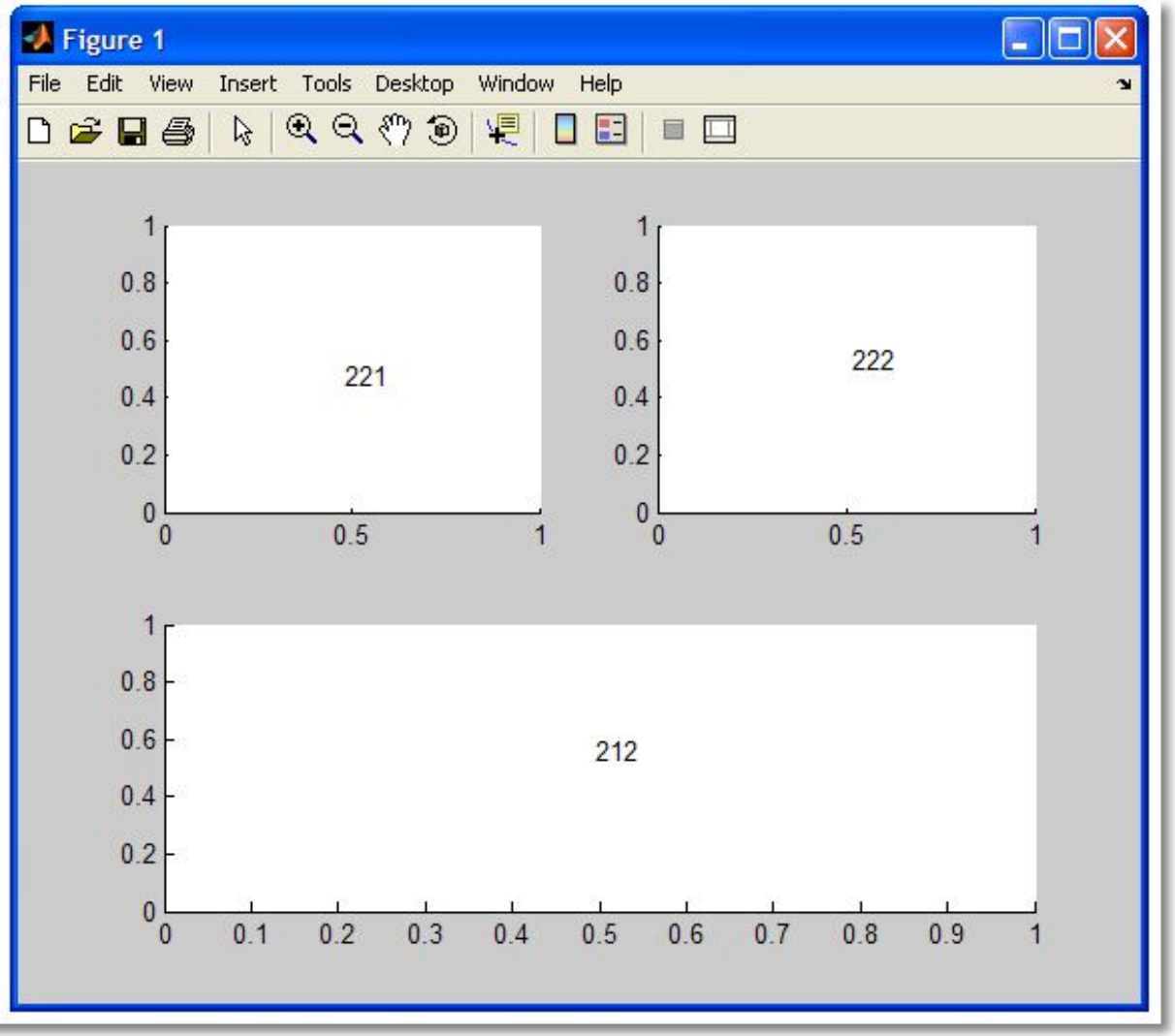
Второй *subplot*

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



Более хитрый пример *subplot*

```
>> 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')
```

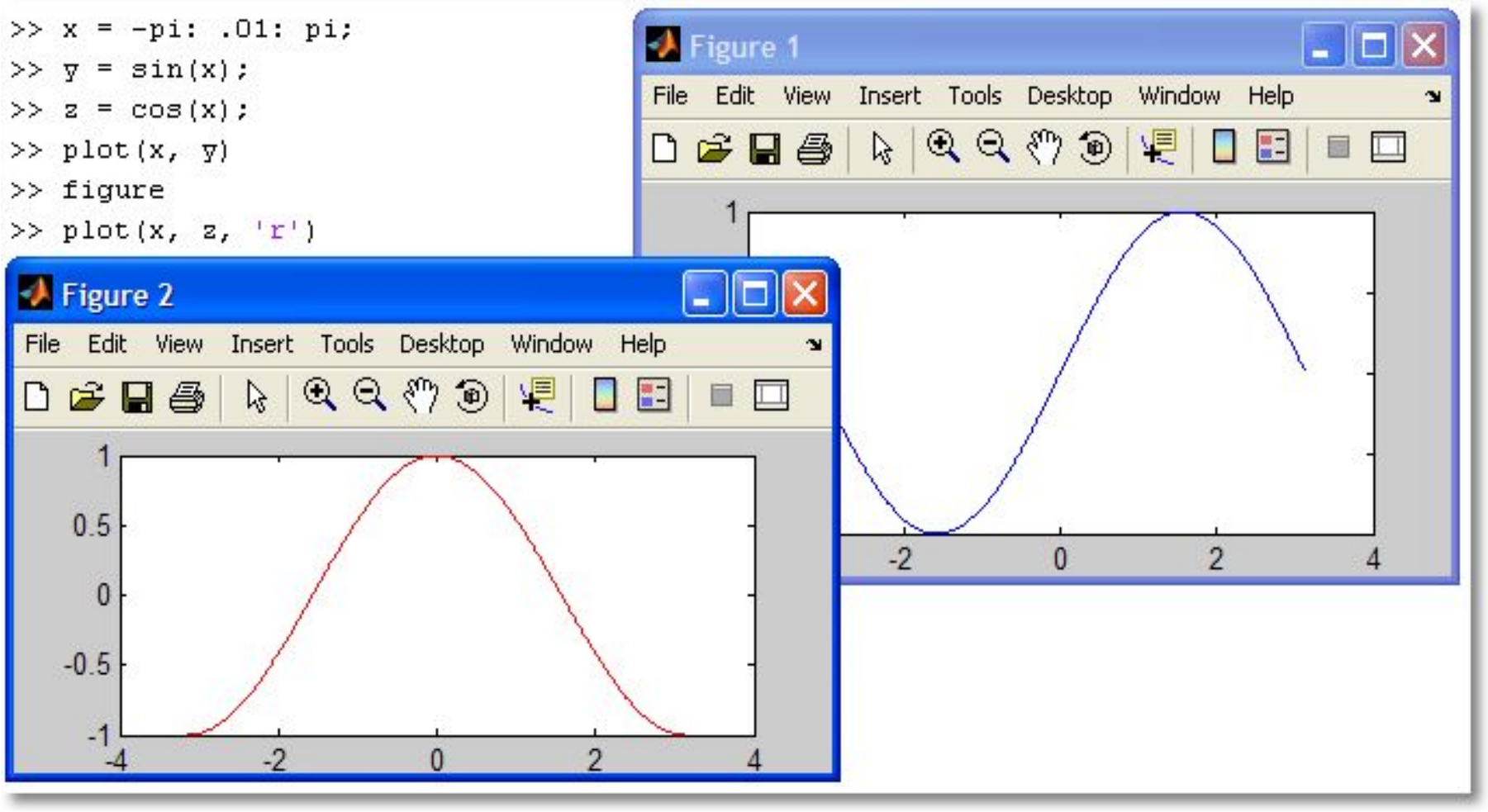


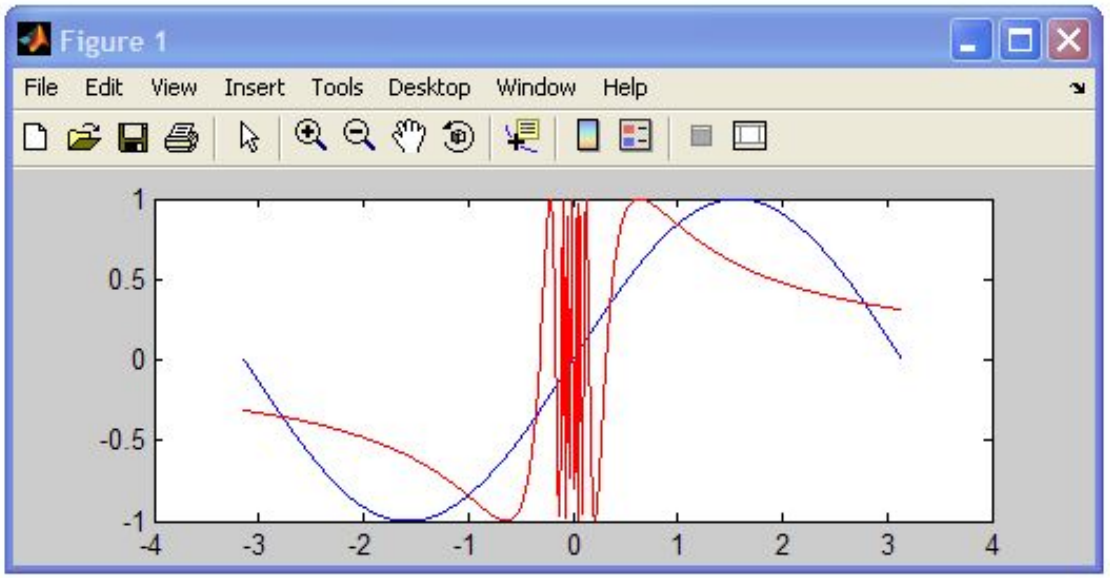
figure : пример использования 2

```
>> 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')
```



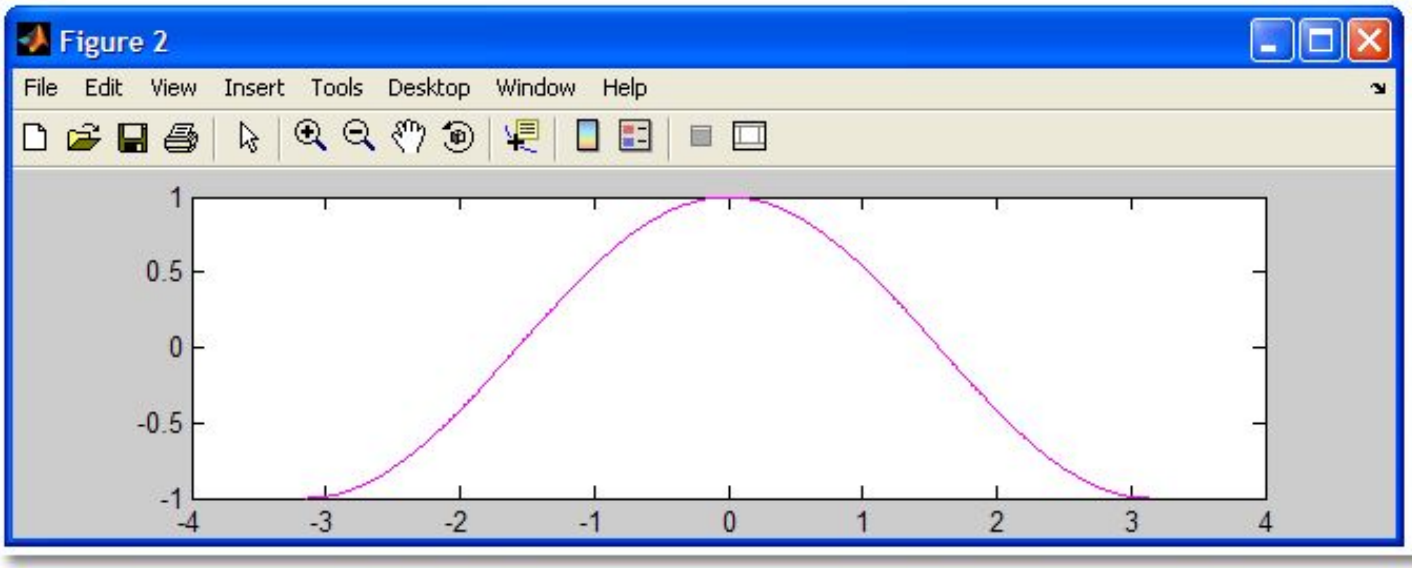


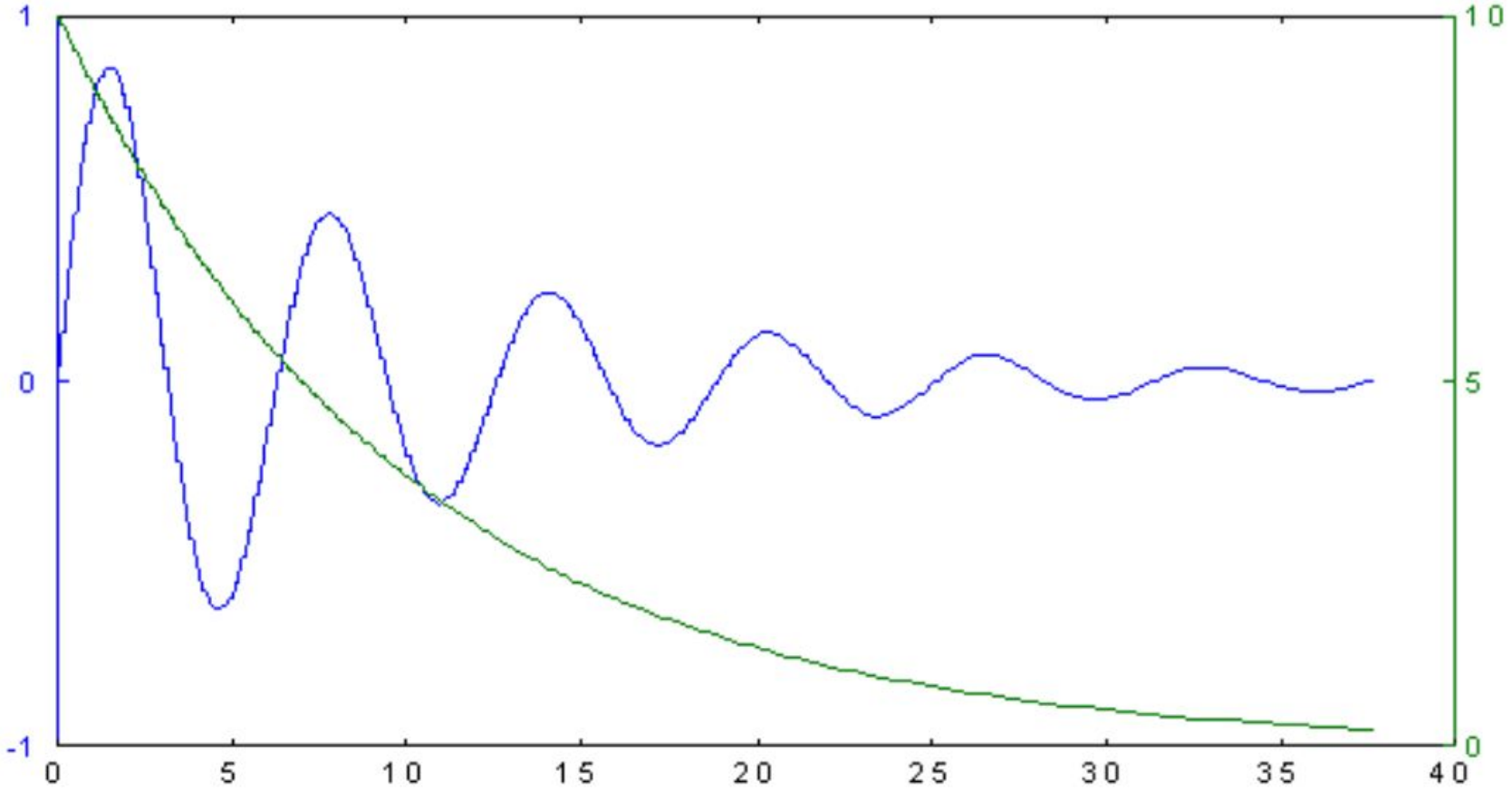
График с двумя осями координат

Одна ось координат отображается слева, другая справа.

График реализуется функцией

plotyy(x1,y1,x2,y2).

```
>> x=0:0.01:12*pi;
>> plotyy(x,sin(x).*exp(-0.1.*x),x, 10*exp(-0.1.*x))
```


1
0
-1
10
5
0
0
5
10
15
20
25
30
35
40

Построение графиков, заданных символьным выражением

- *Пример*.

Построить в одном окне в полулогарифмическом масштабе с использованием функции semilogx(...) графики зависимостей

$$f(t) = t \cdot \ln(t^2) + 1000,$$

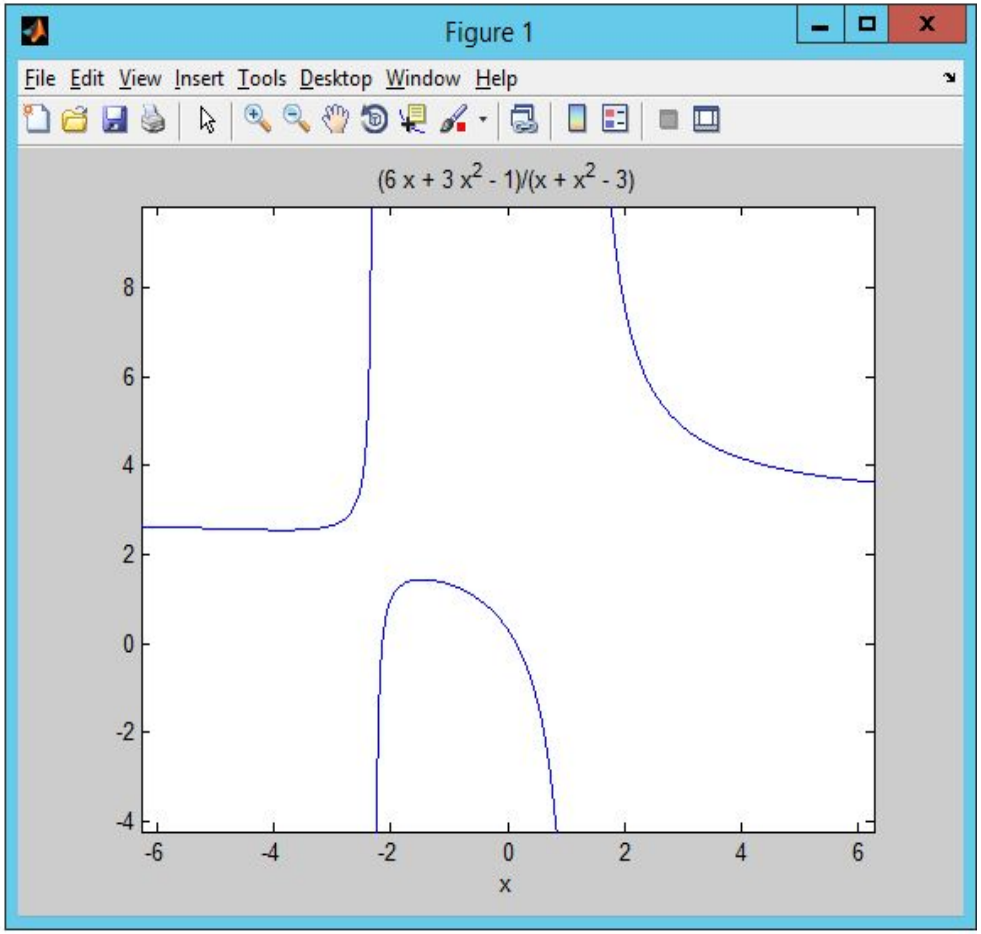
Пример 1. Построить график функции

при усло[illegible]тервале [10,1000

Гра[illegible]ь красным

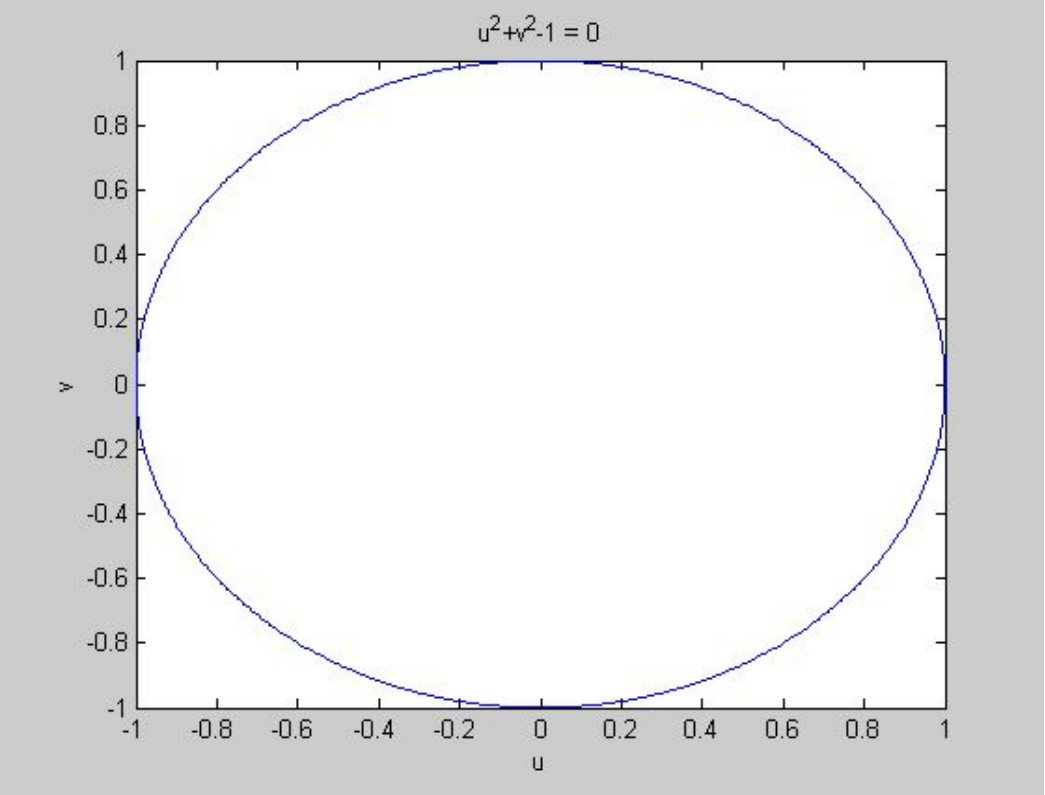
$$f(x) = \frac{3x^2 + 6x - 1}{x^2 + x - 3}$$

```
>> syms x
>> chisl=3*x^2+6*x-1;
>> znam=x^2+x-3;
>> f=chisl/znam;
>> ezplot(f)
>>
```



Пример 2.

```
>> ezplot('u^2+v^2-1', [-1,1], [-1,1])
```

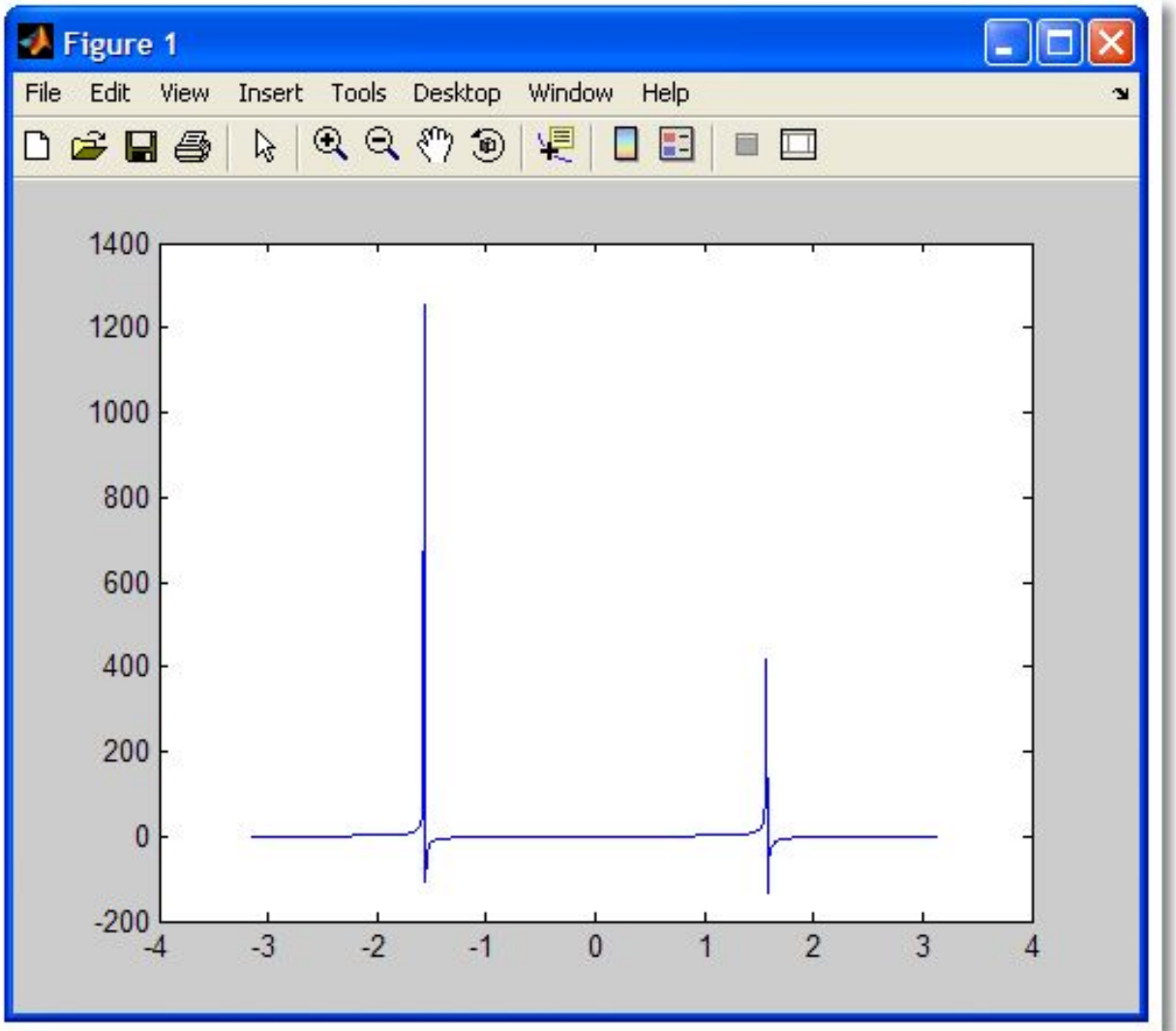


`Axis`: управление масштабом

- Команда
  `axis([Xmin Xmax Ymin Ymax])`
  задаёт область построения графиков по осям X и Y.
- Используется, если результат автомасштабирования неудовлетворителен.

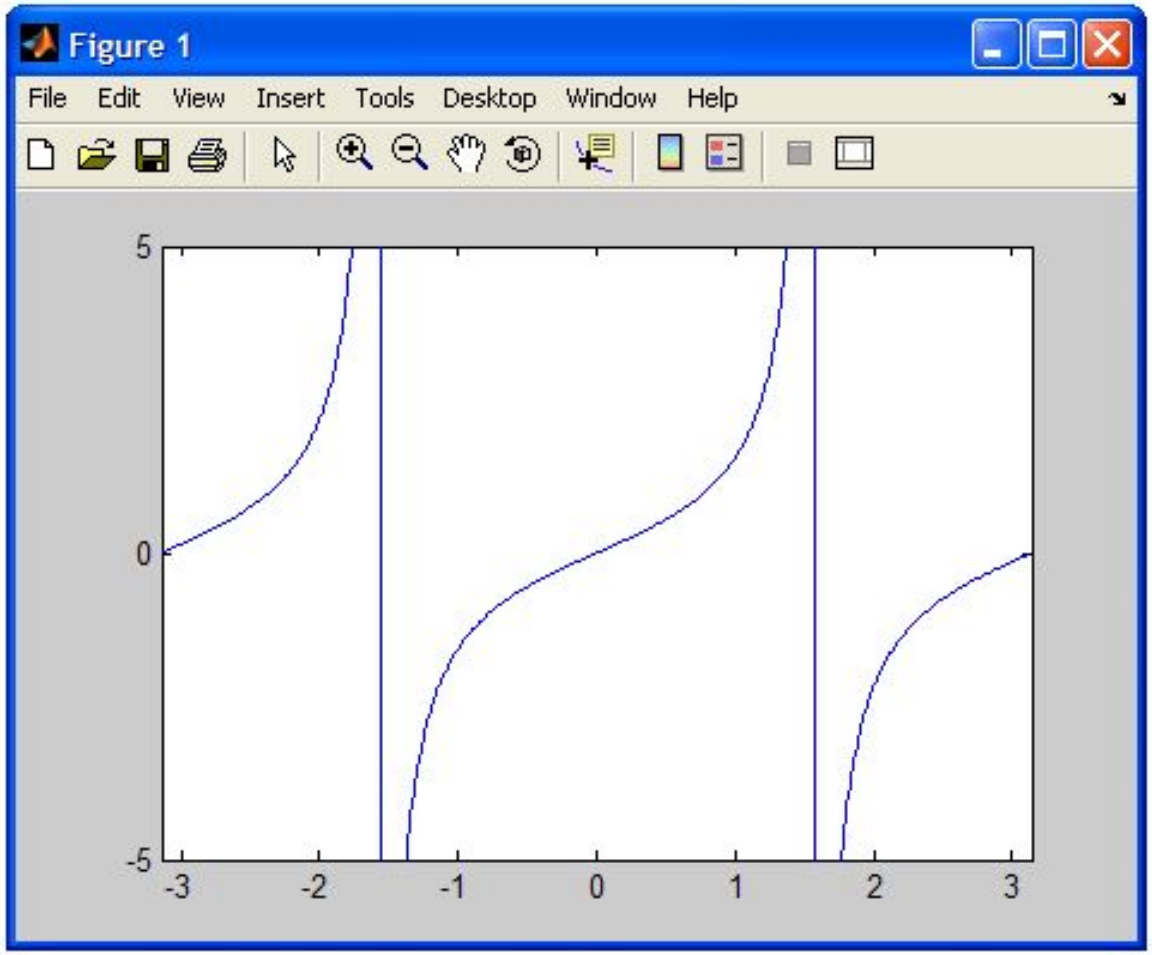
Пример: **`Axis`** не используется

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



Пример: **`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')`
  - легенда: `legend('текст')`
- В заголовках и подписях можно использовать нотацию системы TeX.

Оформление графиков

```
>> time = [0 4 7 9 10 11 12 13 13.5 14 14.5 15 16 17 18 20 22];
>> temp1 = [14 15 14 16 18 17 20 22 24 28 25 20 16 13 13 14 13];
>> temp2 = [12 13 13 14 16 18 20 20 23 25 25 20 16 12 12 11 10];
>> plot(time, temp1, 'ro-', time, temp2, 'go-')
>> grid on
>> title('Суточные температуры')
>> xlabel('Время (час.)')
>> ylabel('Температура(С)')
>> legend('10 мая','11 мая')
```

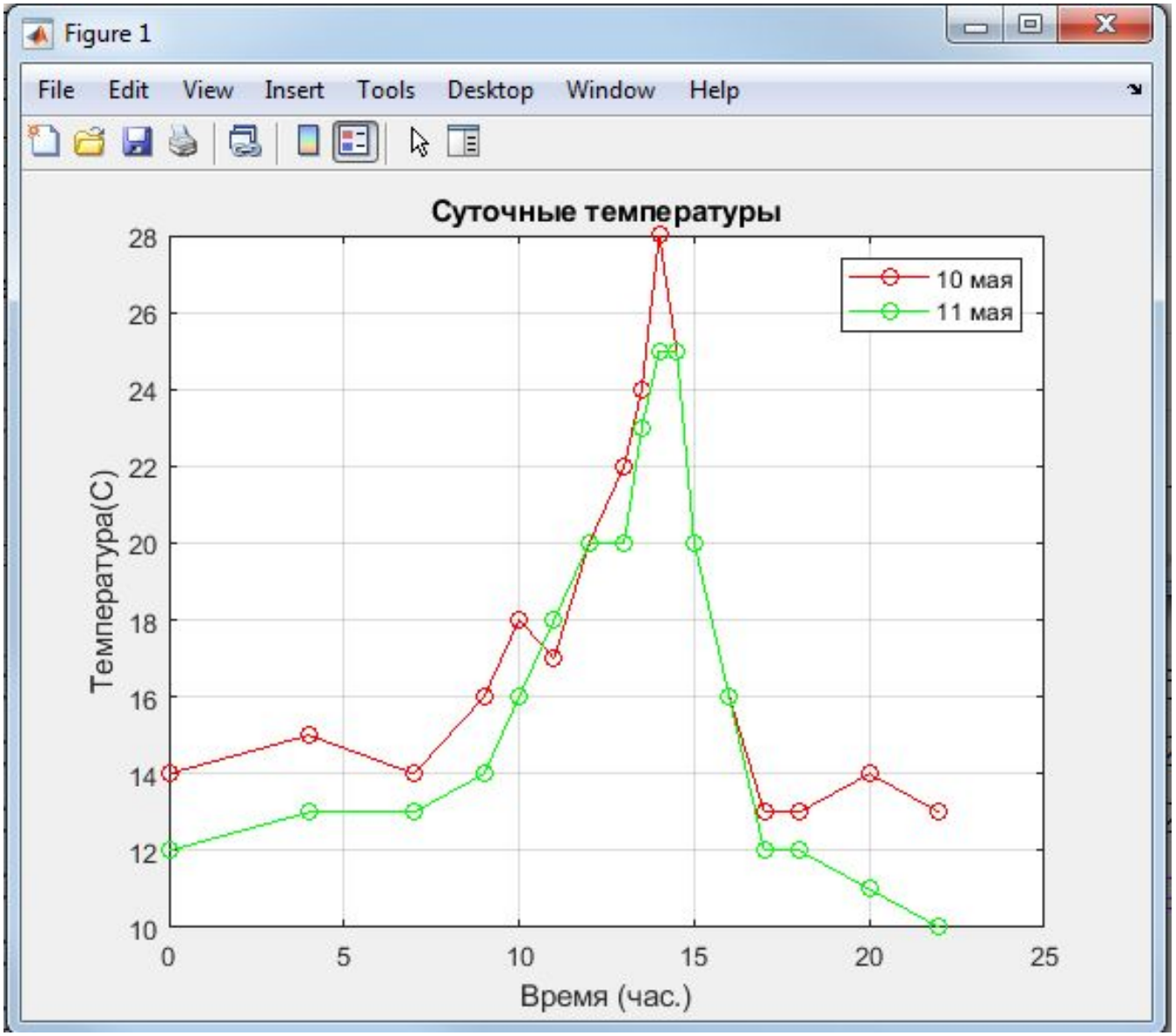
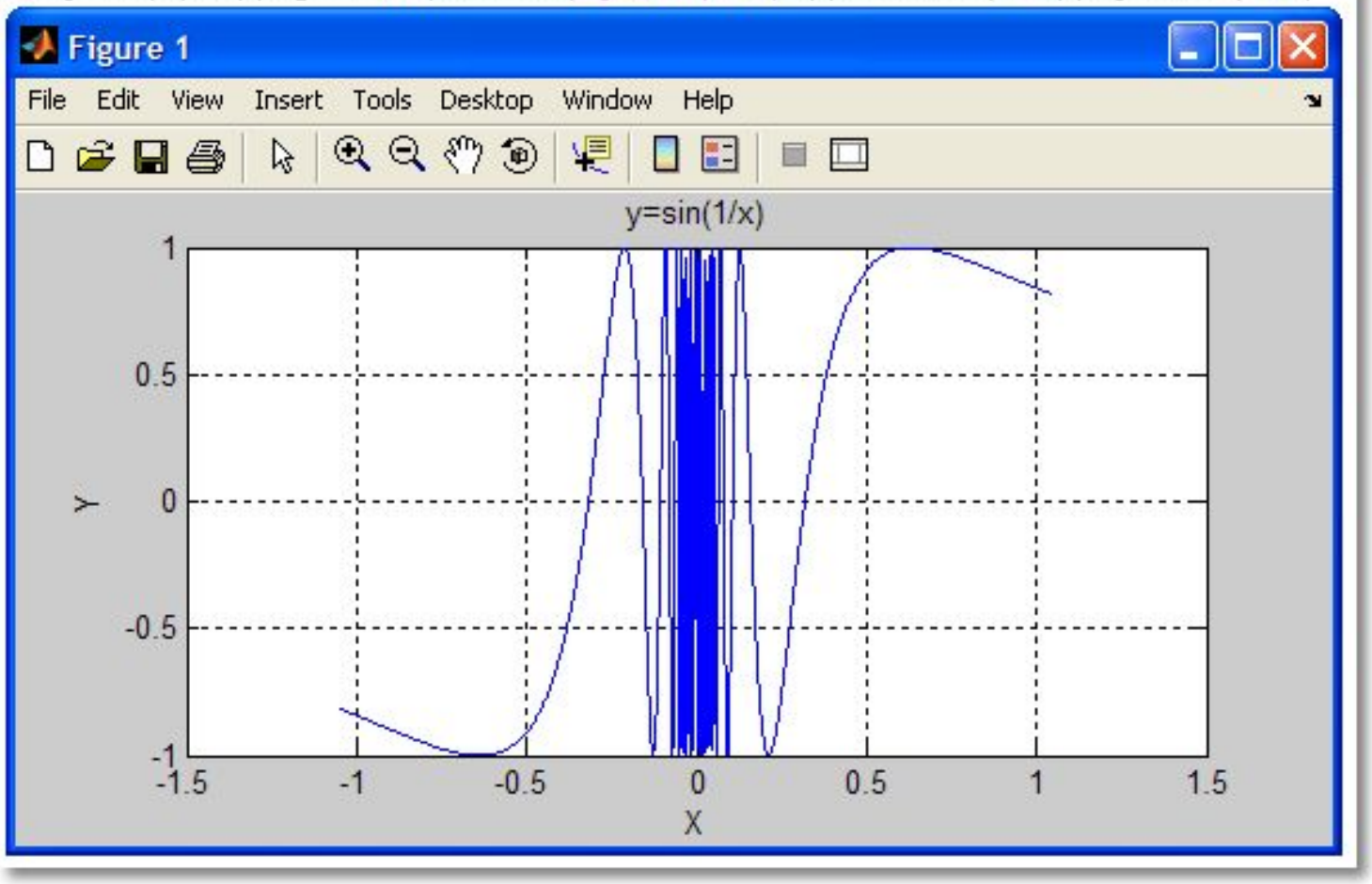


Figure 1
File Edit View Insert Tools Desktop Window Help
Суточные температуры
10 мая
11 мая
Температура(C)
Время (час.)
28
26
24
22
20
18
16
14
12
10
0
5
10
15
20
25

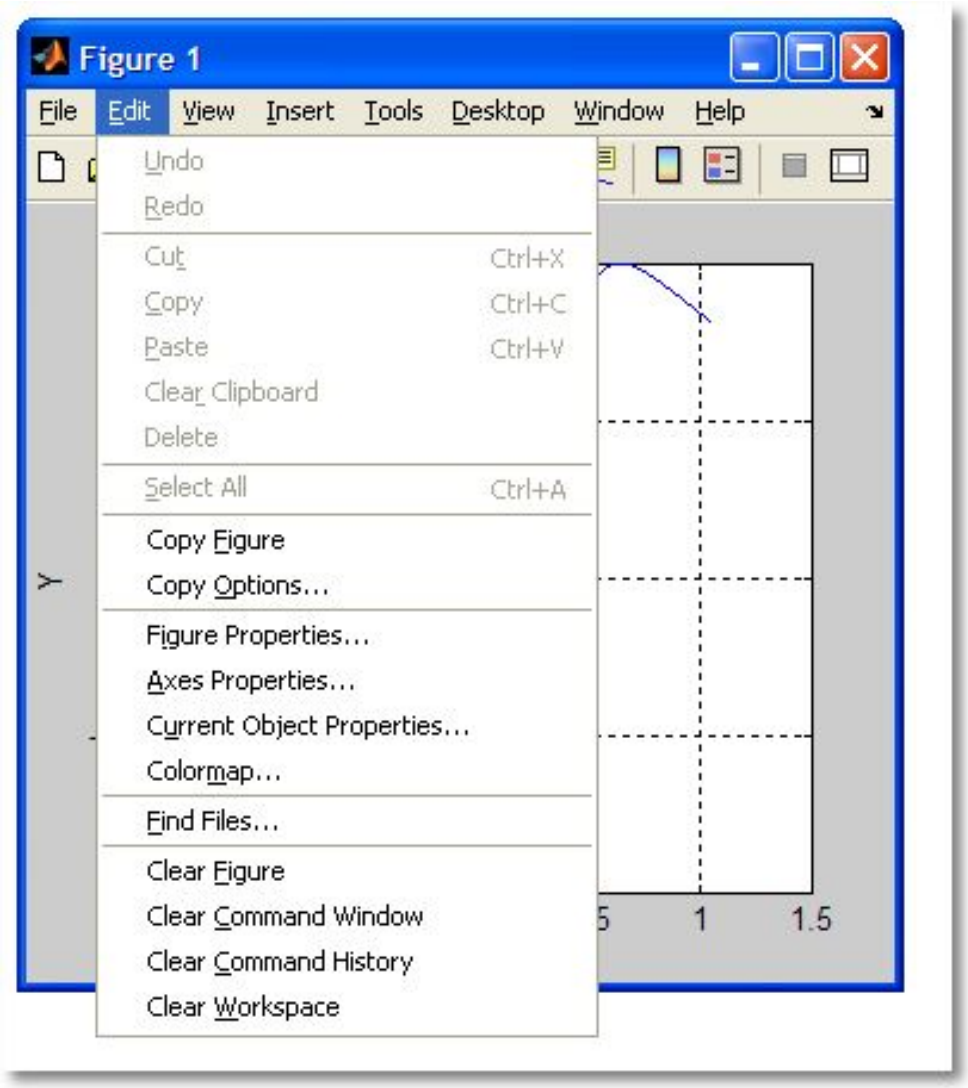
Пример оформления графика

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



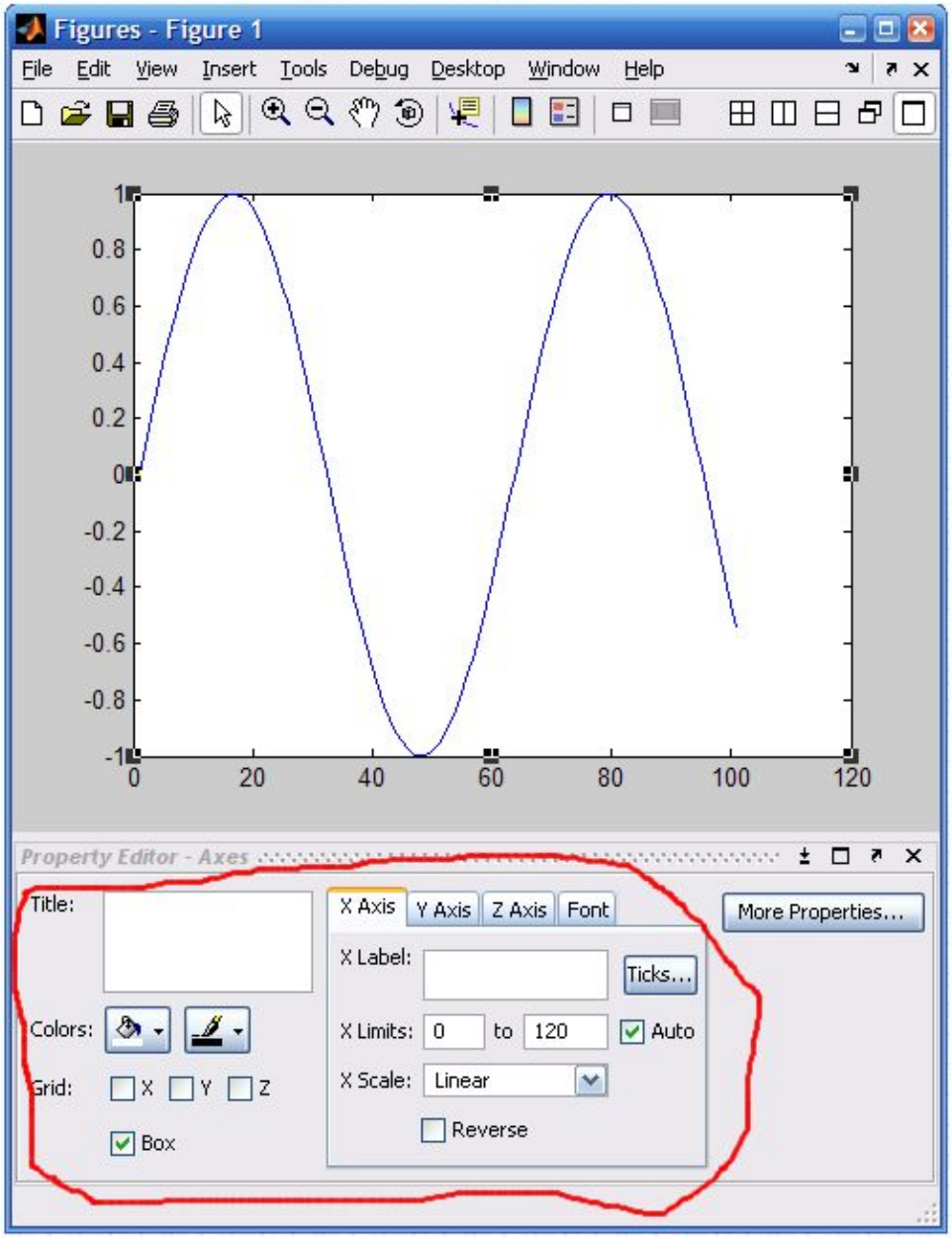
Форматирование графиков

- Доступно из меню `Edit`:

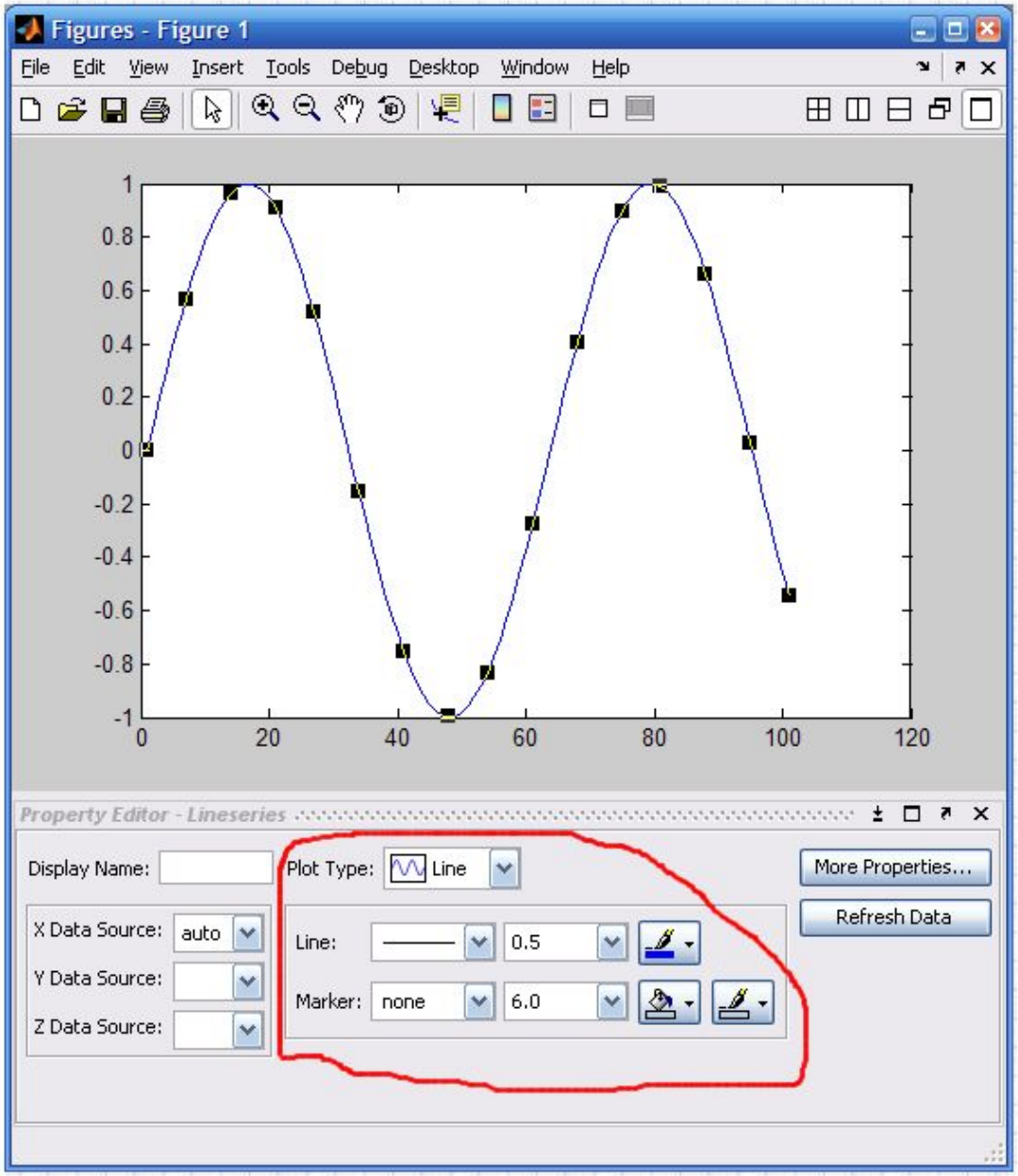


Вид редактора зависит от того, какой элемент графика выделен

Выделена рамка



Выделена линия графика

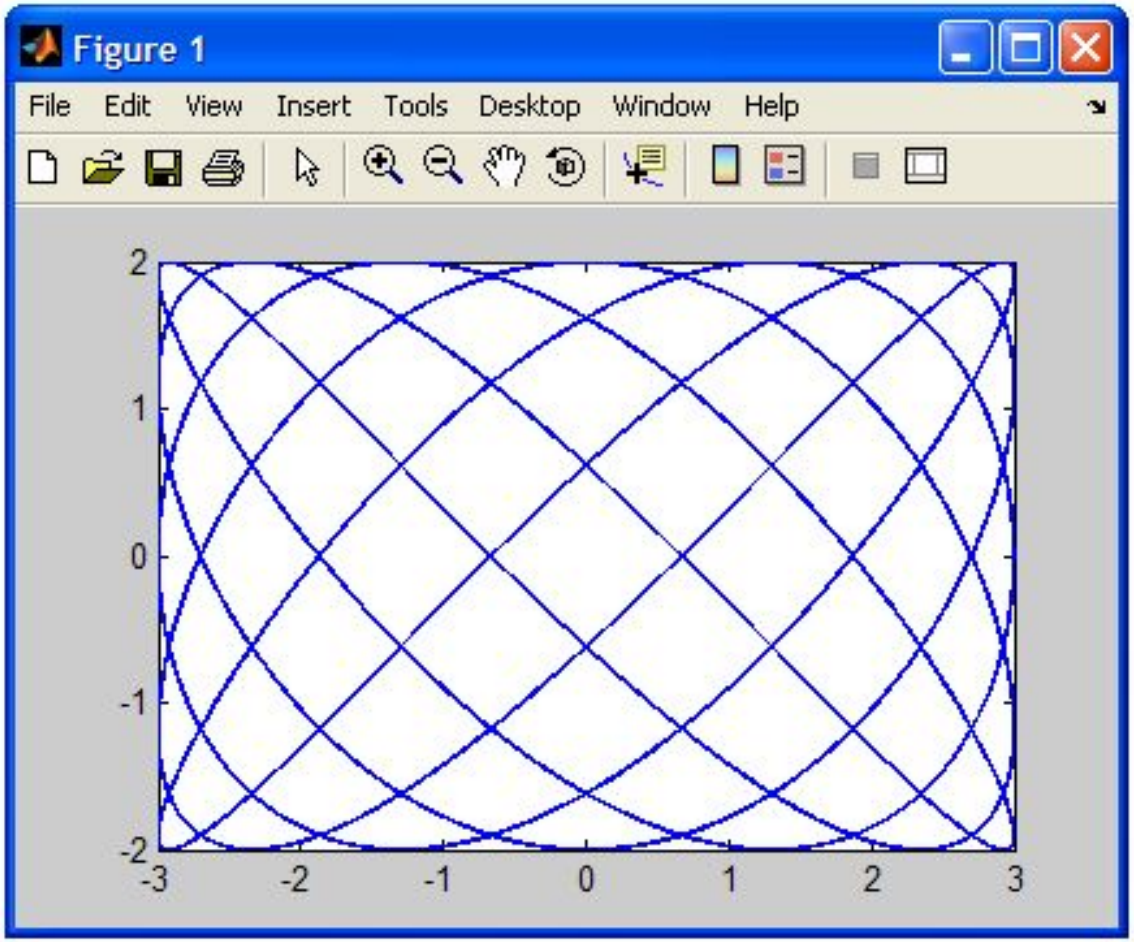


Графики функций, заданных параметрически

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

Графики функций, заданных параметрически

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



Графики функций, заданных параметрически

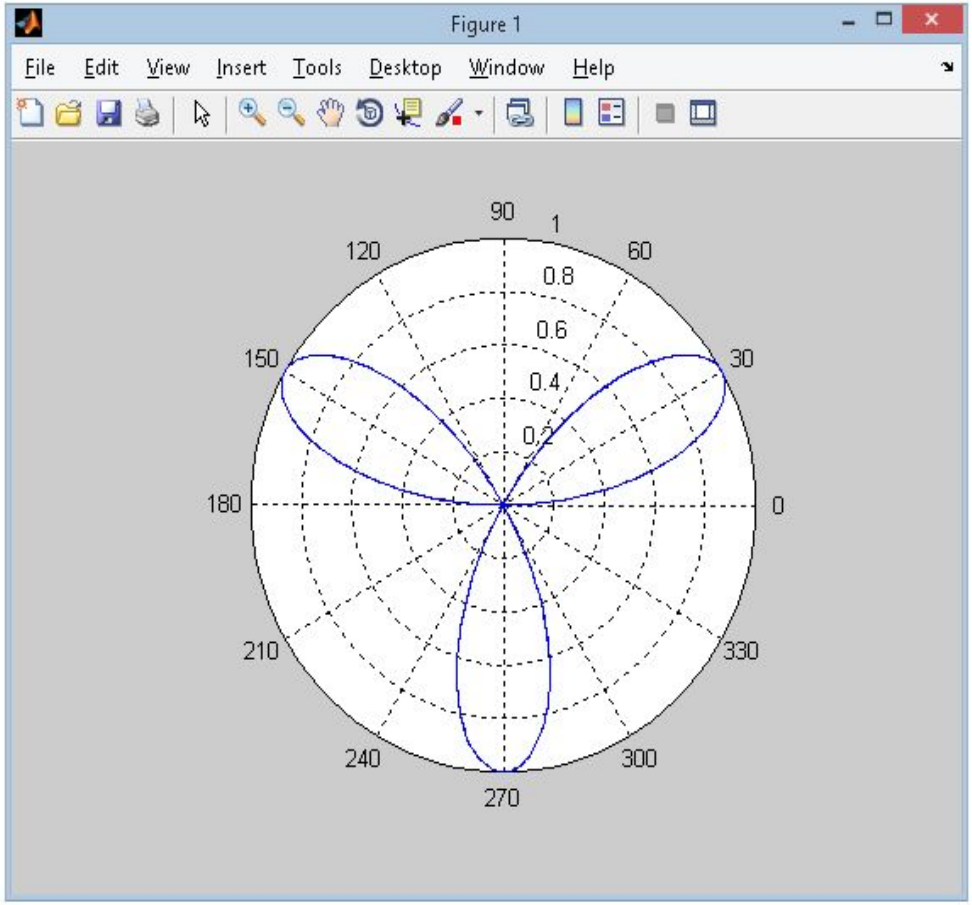
- Графики параметрических функций часто возникают в физических приложениях.
- Независимая переменная ***t*** в этом случае имеет смысл времени, ***x*** и ***y*** – координаты.
- Для построения динамического графика можно использовать функцию ***comet(x,y)*** .

Функции в полярной системе координат

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

Пример построения функции в полярной системе координат

```
>> x=0:.01:2*pi;
>> y=sin(3*x);
>> polar(x,y)
>>
```



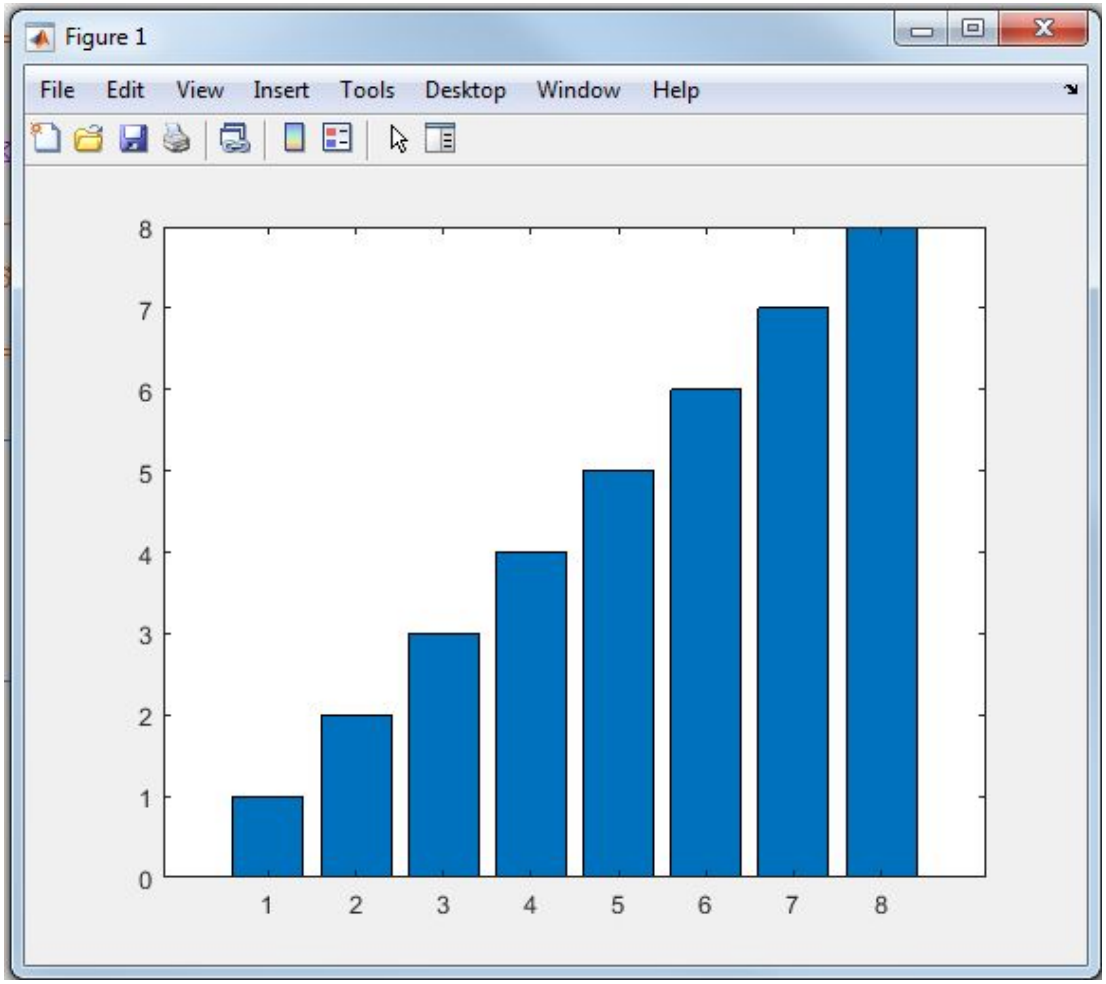
Построение диаграмм

- *Столбиковая диаграмма с вертикальным расположением*

  Строится при помощи команд:

  bar(x,y) – строит столбиковый график элементов вектора ***y*** при заданных значениях вектора ***x***, которые должны идти в монотонно возрастающем порядке.

```
>> y=1:8
 y =
    1  2  3  4  5  6  7  8
>> bar(y)
```



bar*(y)** – строит график элементов матрицы ***y так же, как указано выше, но для построения графика используется вектор ***x=1:m.***

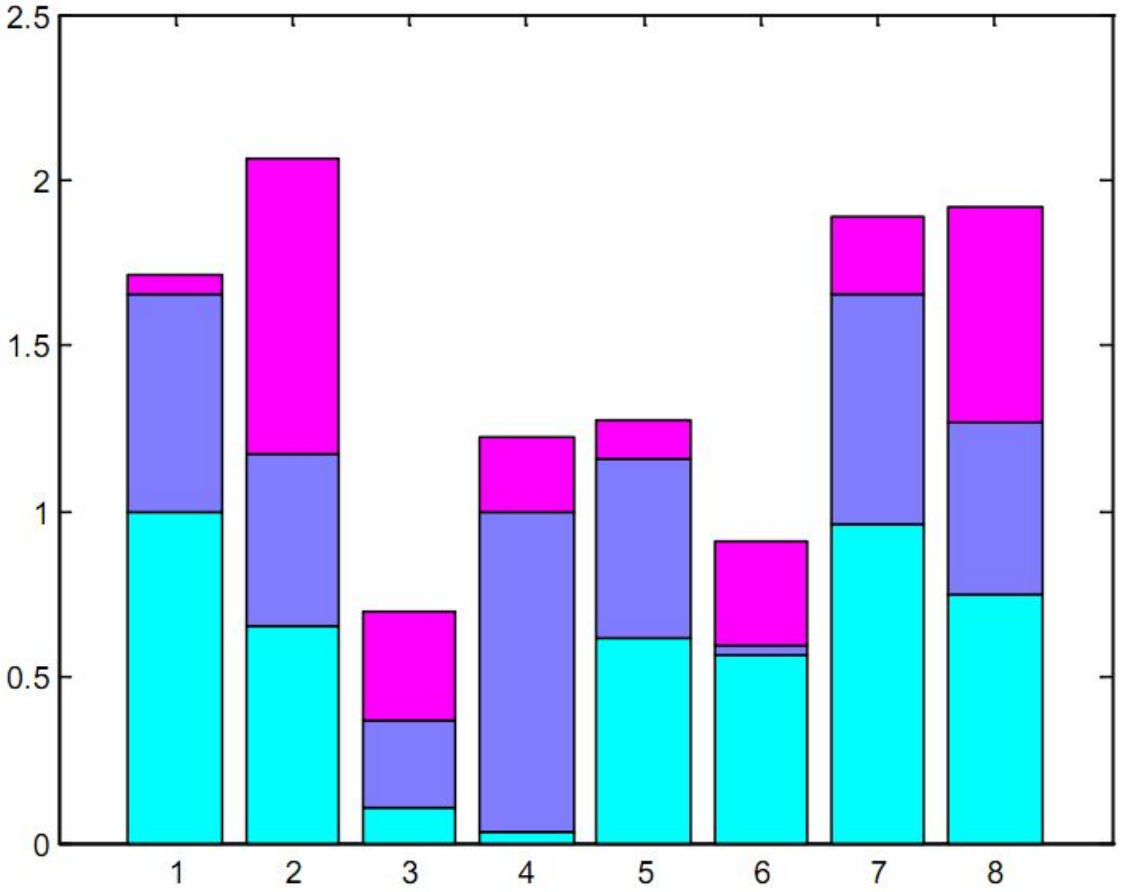
Возможно применение этих команд со спецификацией

***bar*(…,'спецификация'),**

где 'спецификация' – тип линии, цвет и т. д. по аналогии с командой *plot*.

Спецификация ***'stacked'*** задает рисование всех *n* столбцов в позиции *m* друг на друге.

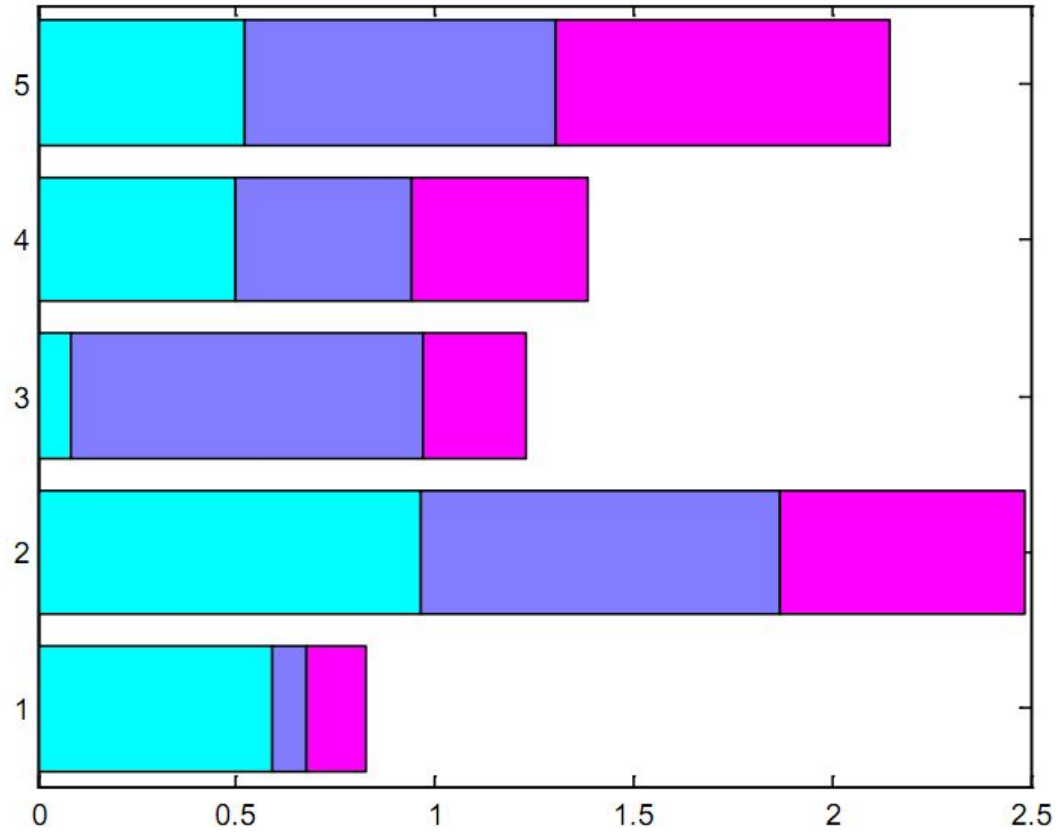
```
>>y=rand(12,3);
>>bar(y,'stacked')
```



- *Столбиковая диаграмма с горизонтальным расположением*

Строится командой ***barh(…)***, аналогичной по синтаксису команде ***bar(…).***

```
>>y=rand(5,3);
>>barh(y,'stacked')
```

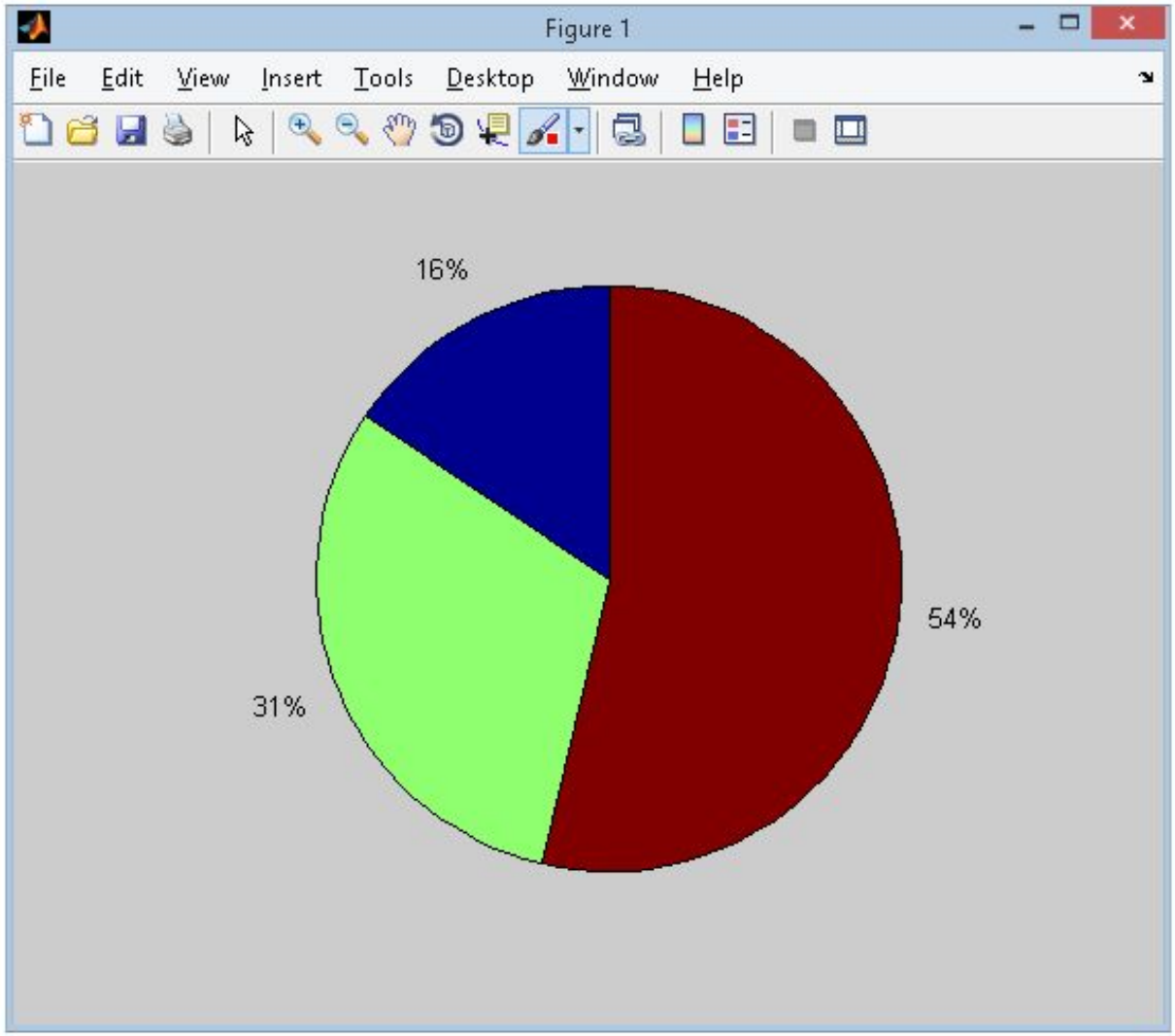


- *Круговая диаграмма*

  Диаграммы состоят из плоских секторов (аналогичных кусков пирога) и строятся с помощью функции ***pie*(…).**

  В простейшем случае вектор ***y***, содержащий ***k*** положительных компонентов генерирует ***k*** секторов, центральный угол которых пропорционален вкладу каждого компонента в общую сумму.

```
>> y=rand(1,3);
>> pie(y)
>>
```

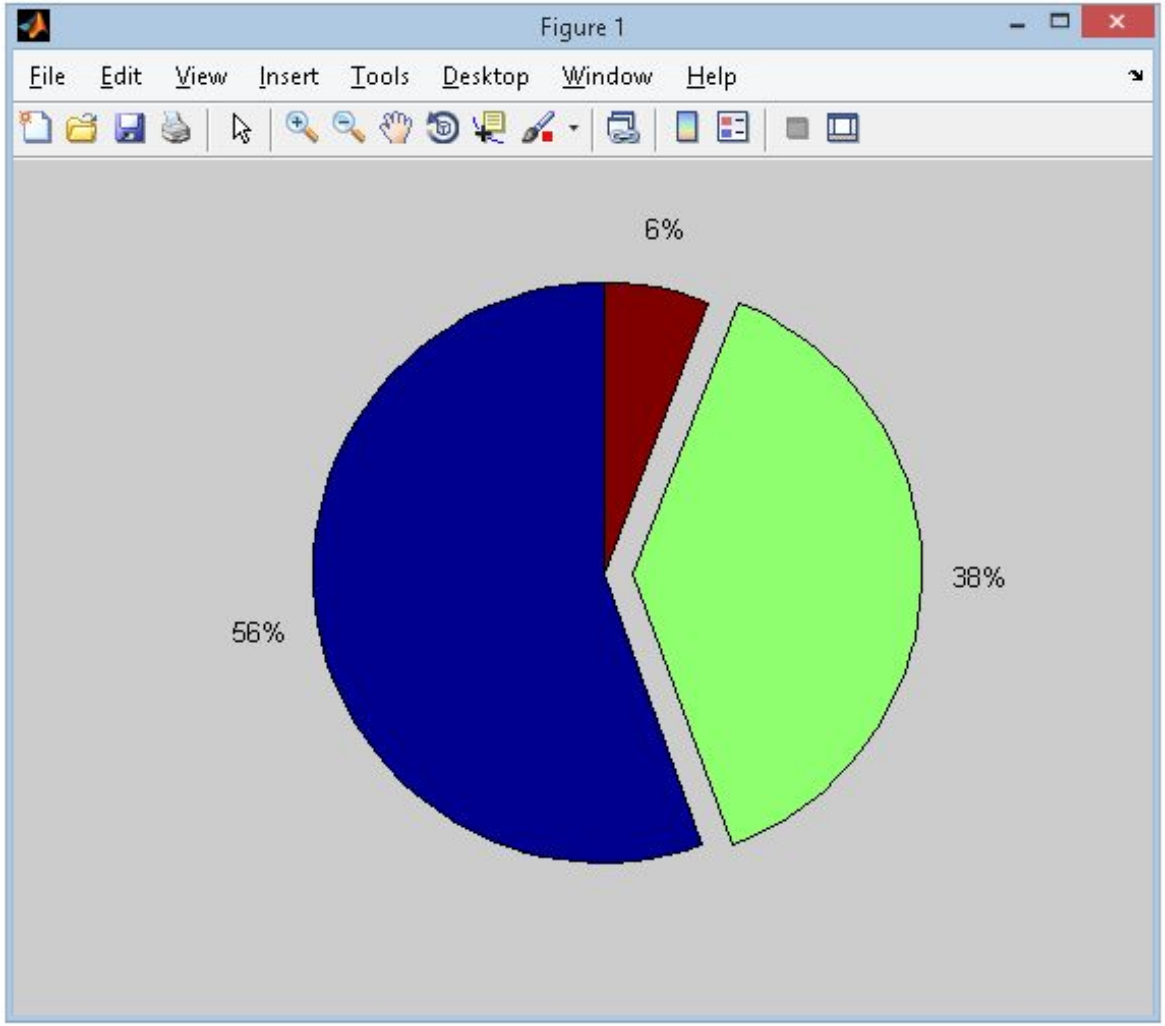


- *Круговая диаграмма с отдельными секторами*

  Создание выделяющихся элементов обеспечивается заданием еще одного аргумента такой же размерности, что и вектор ***y***.

  Выдвигаемым секторам в новом векторе должны соответствовать ненулевые элементы.

```
>> y=rand(1,3);
>> v=[0,1,0];
>> pie(y,v)
>>
```



Построение гистограмм

Классическая гистограмма характеризует число попаданий значений вектора ***y*** в ***m*** интервалов.

Для получения данных для гистограмм служит функция **hist**, записываемая в следующем виде:

- **hist(y)** - возвращает вектор числа попаданий для 10 интервалов, выбираемых автоматически.

Если *y* – матрица, то выдается массив данных о числе попаданий для каждого из столбцов.

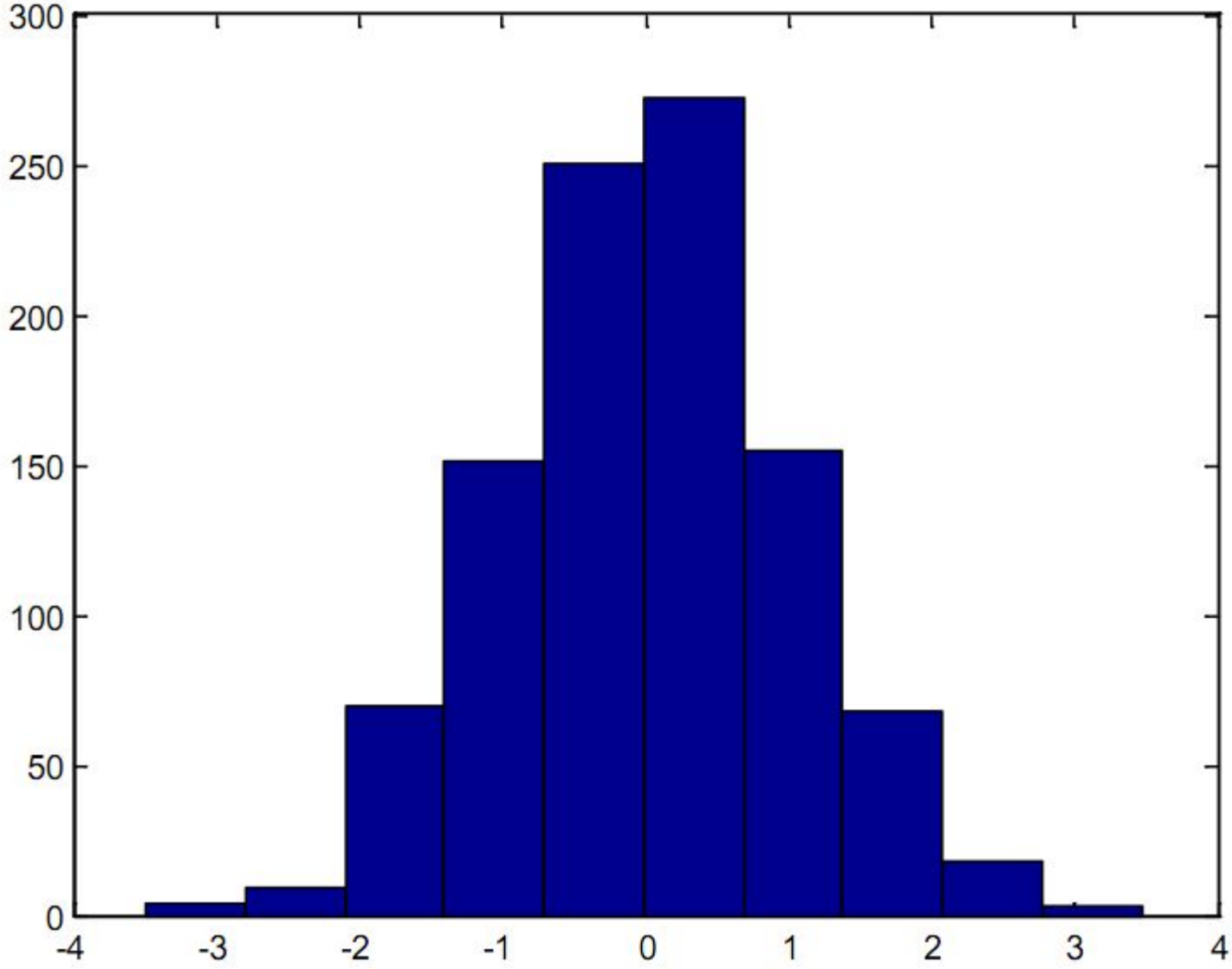
- **hist(y,m)** – аналогична вышерассмотренной, но используется m интервалов (m-скаляр).

- **hist(y,x)** – возвращает числа попаданий элементов вектора ***y*** в интервалы, центры которых заданы элементами вектора ***x***.

Примеры.

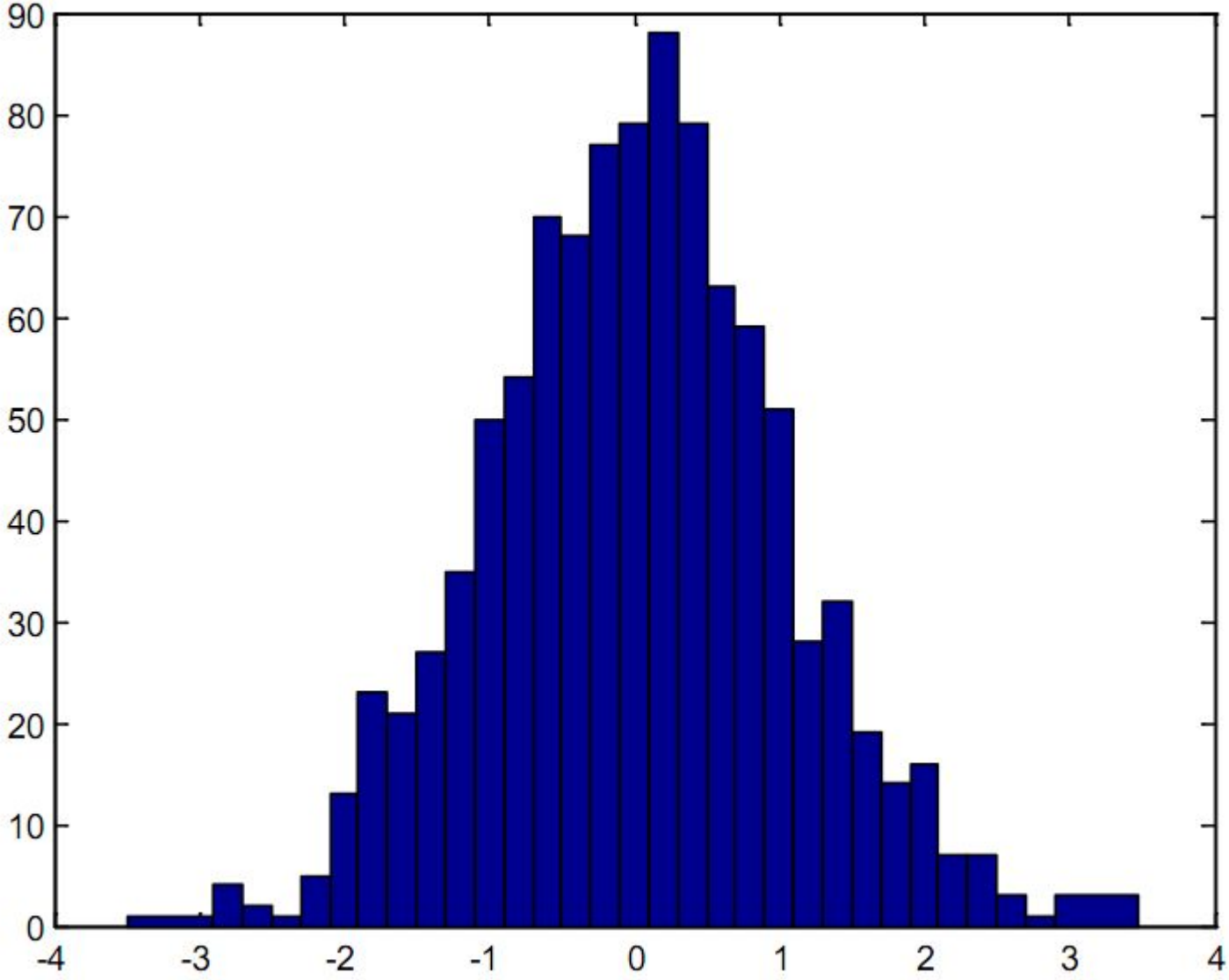
- Построить гистограмму для 1000 случайных чисел и вывести вектор с данными о числах их попаданий для 10 интервалов.

```
>>y=randn(1000,1);
>>hist(y)
```


300
250
200
150
100
50
0
-4
-3
-2
-1
0
1
2
3
4

- Построить гистограмму для элементов массива **y** предыдущего примера и интервалов, центры которых заданы элементами вектора ***x.***

```
>>x= -3,0.2,3;
>>y=randn(1000,1);
>>hist(y,x)
```


90
80
70
60
50
40
30
20
10
0
-4
-3
-2
-1
0
1
2
3
4

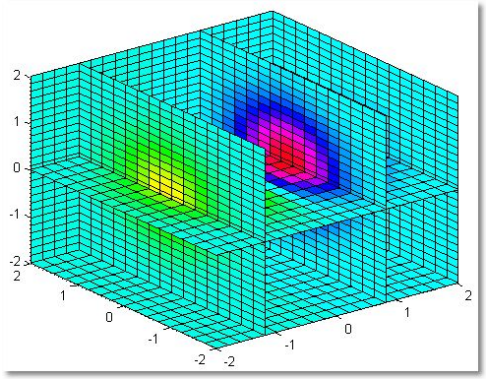
Трёхмерная (3D-) графика

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

1
0.5
0
-0.5
10
5
0
-5
-10
-10
-5
0
5
10

1
0.5
0
5
0
-5
-5
0
5

15
10
5
0
80
90
100
110
120
130
20
40
60


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

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

Построение 3D-поверхности

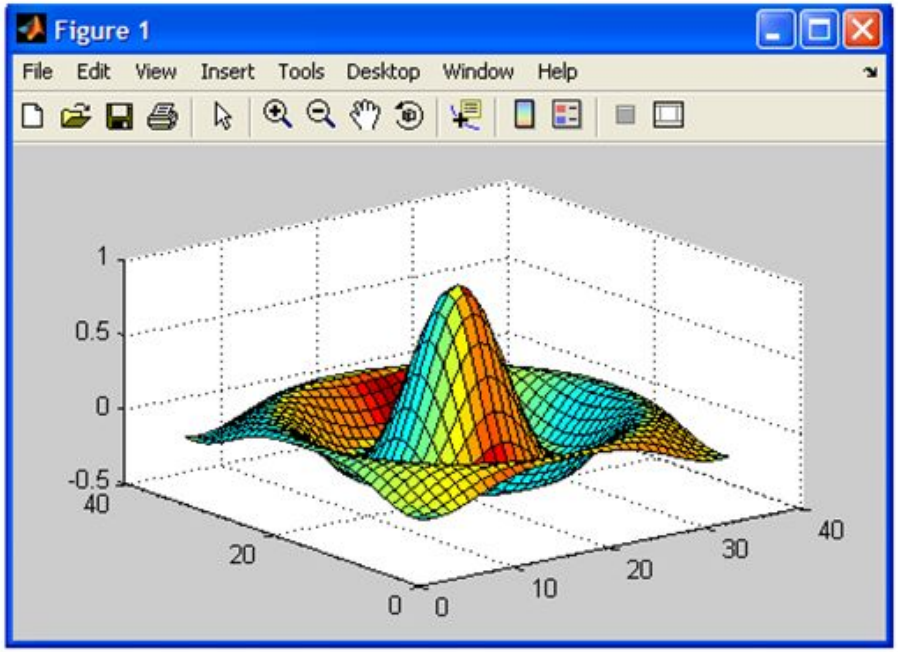
Рассмотрим пример: построить поверхность

$$f(x,y)=sin(r)/r,$$

где $r=sqrt(x^2+y^2)$

Построение поверхостей командой **surf**

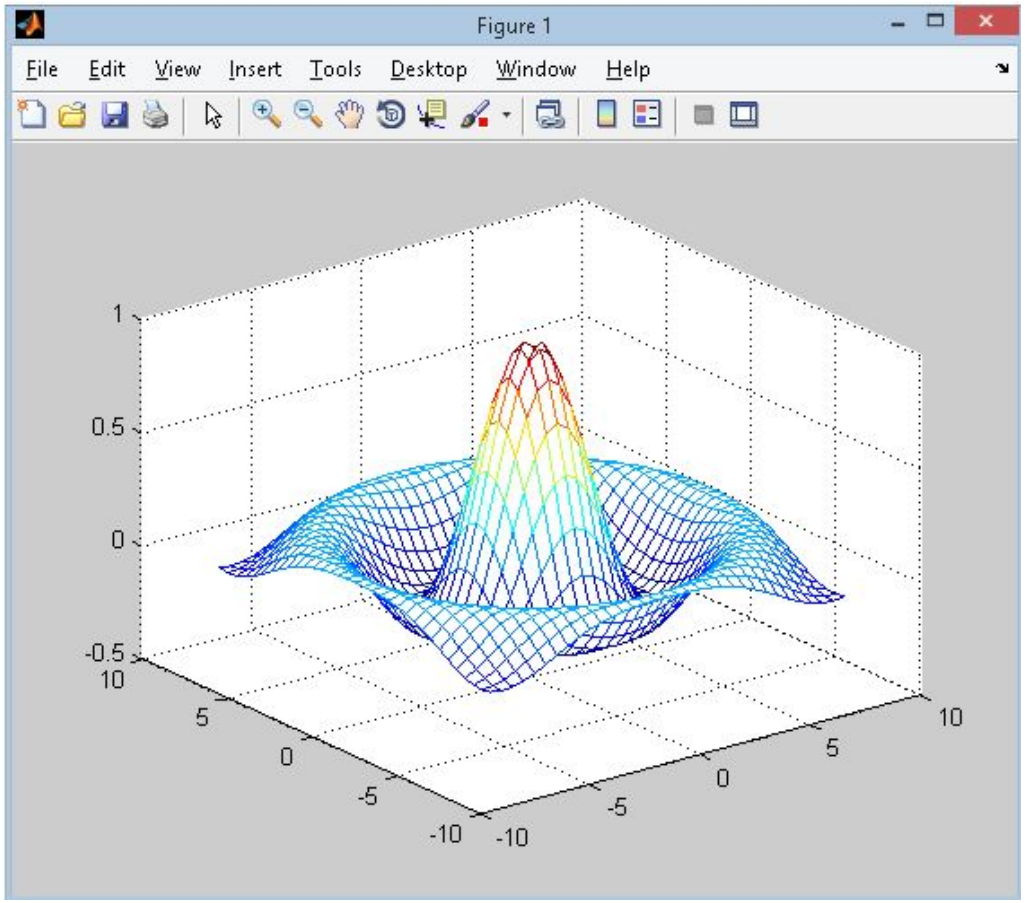
```
>> [X,Y] = meshgrid(-8:.5:8);
>> R = sqrt(X.^2 + Y.^2);
>> Z = sin(R)./R;
>> surf(X,Y,Z)
>>
```



- 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

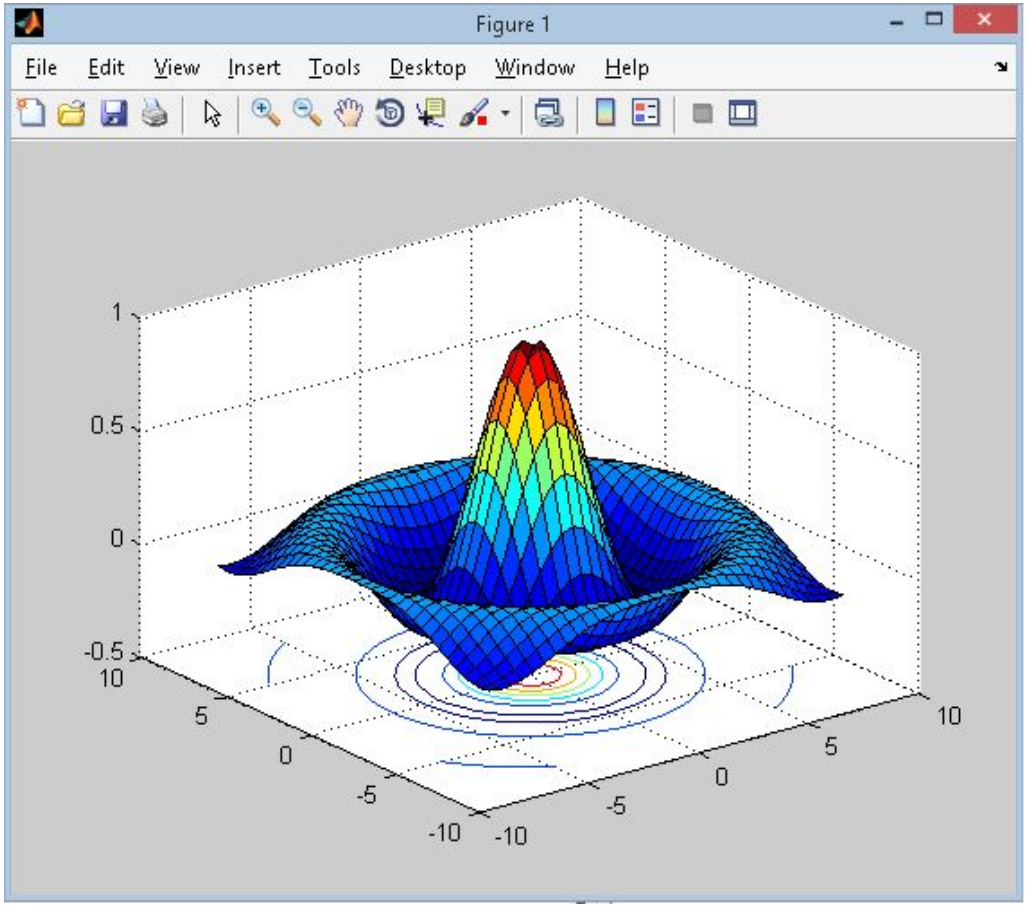
Построение поверхностей командой **mesh**

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



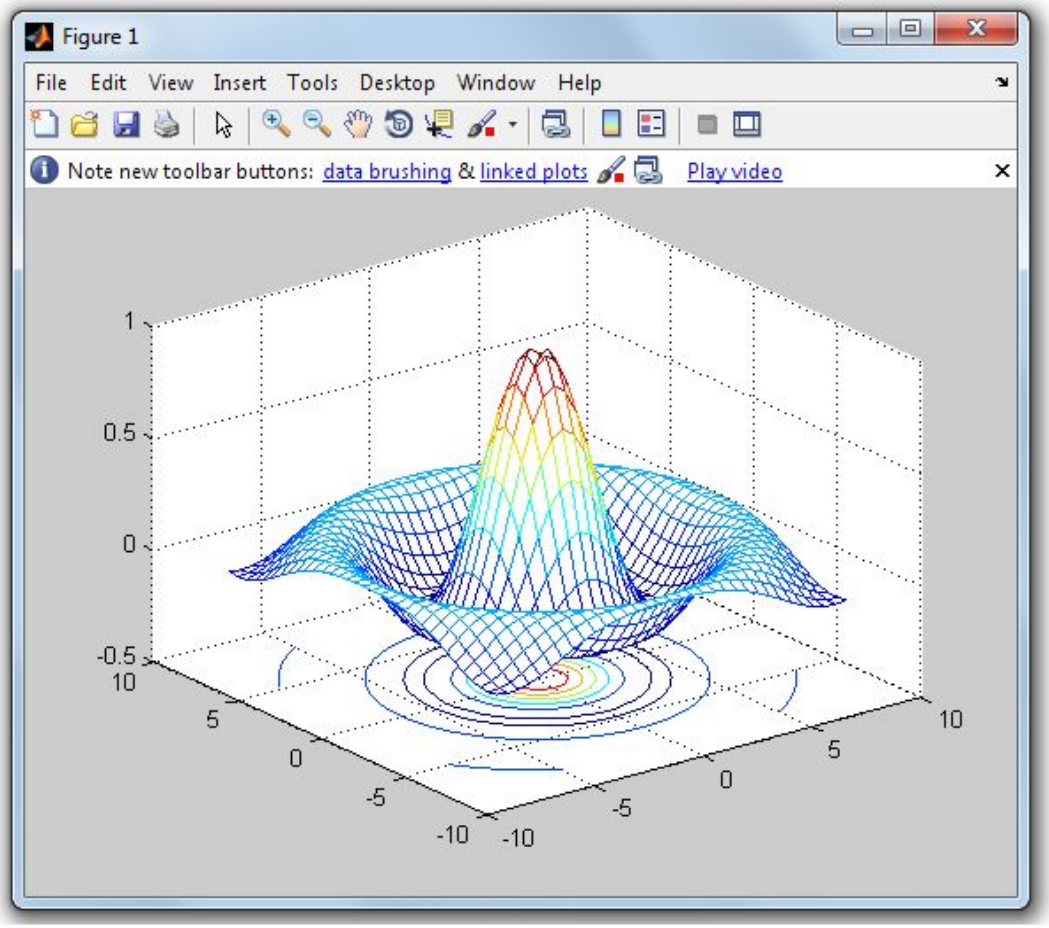
Построение поверхностей командой **surfc**

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



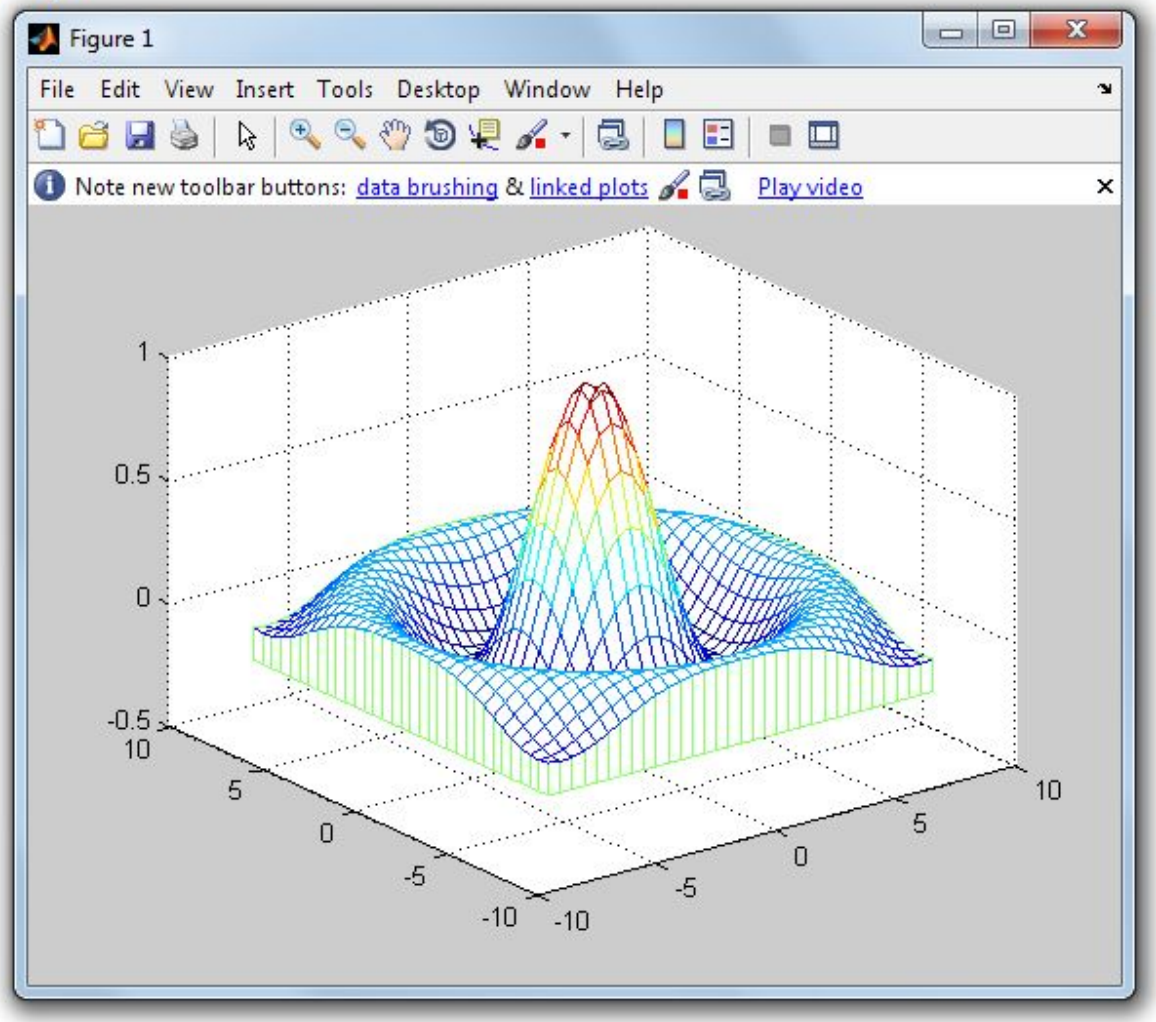
Поверхность и контурная диаграмма под ней

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



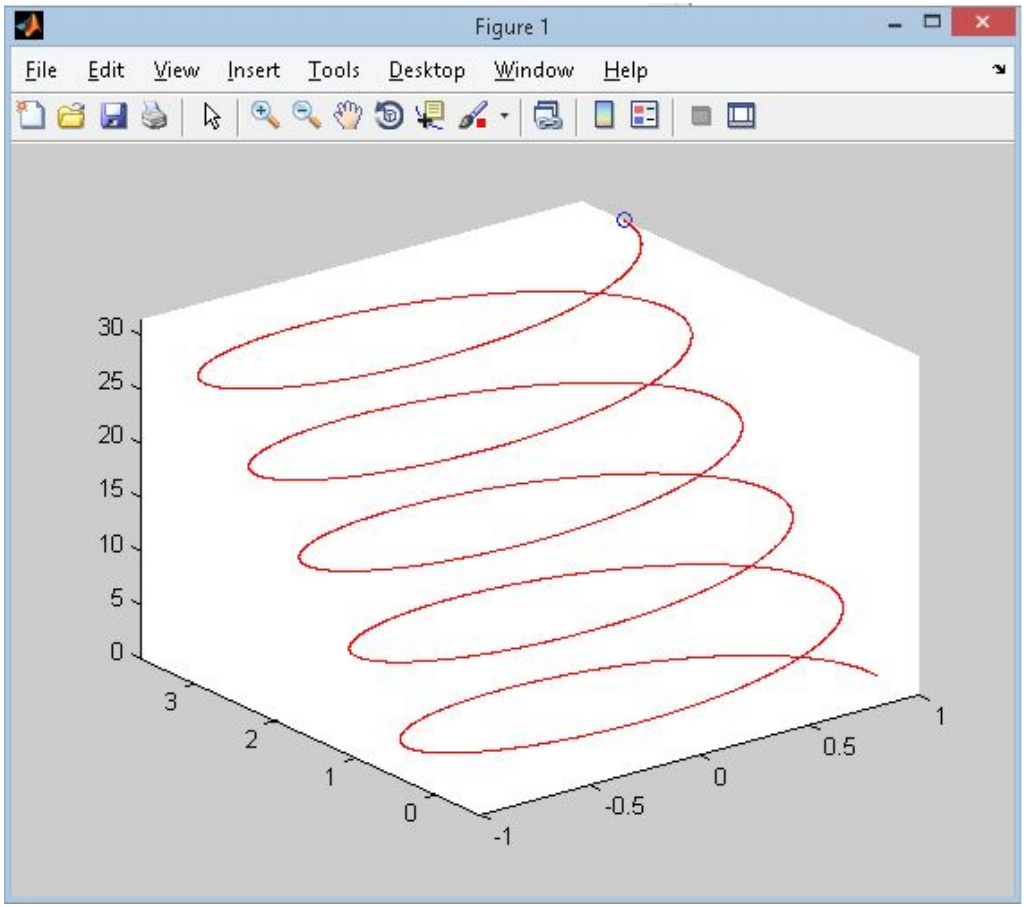
Поверхность на пьедестале

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



Движение по трёхмерной линии

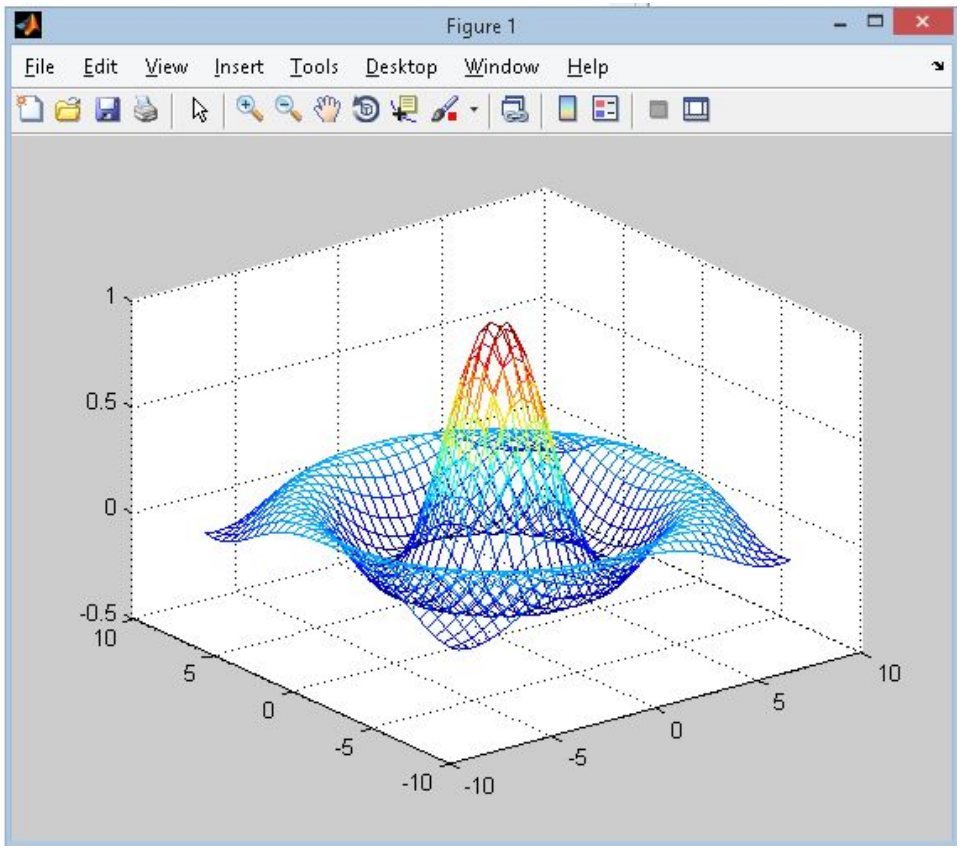
```
>> X=0:pi/500:10*pi;
>> comet3(cos(X),sin(X)+X/10,X)
```



Прозрачная каркасная поверхность

Для того, чтобы сделать каркасную поверхность «прозрачной», следует использовать команду **hidden off**. Предыдущая программа в этом случае имеет следующий вид:

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



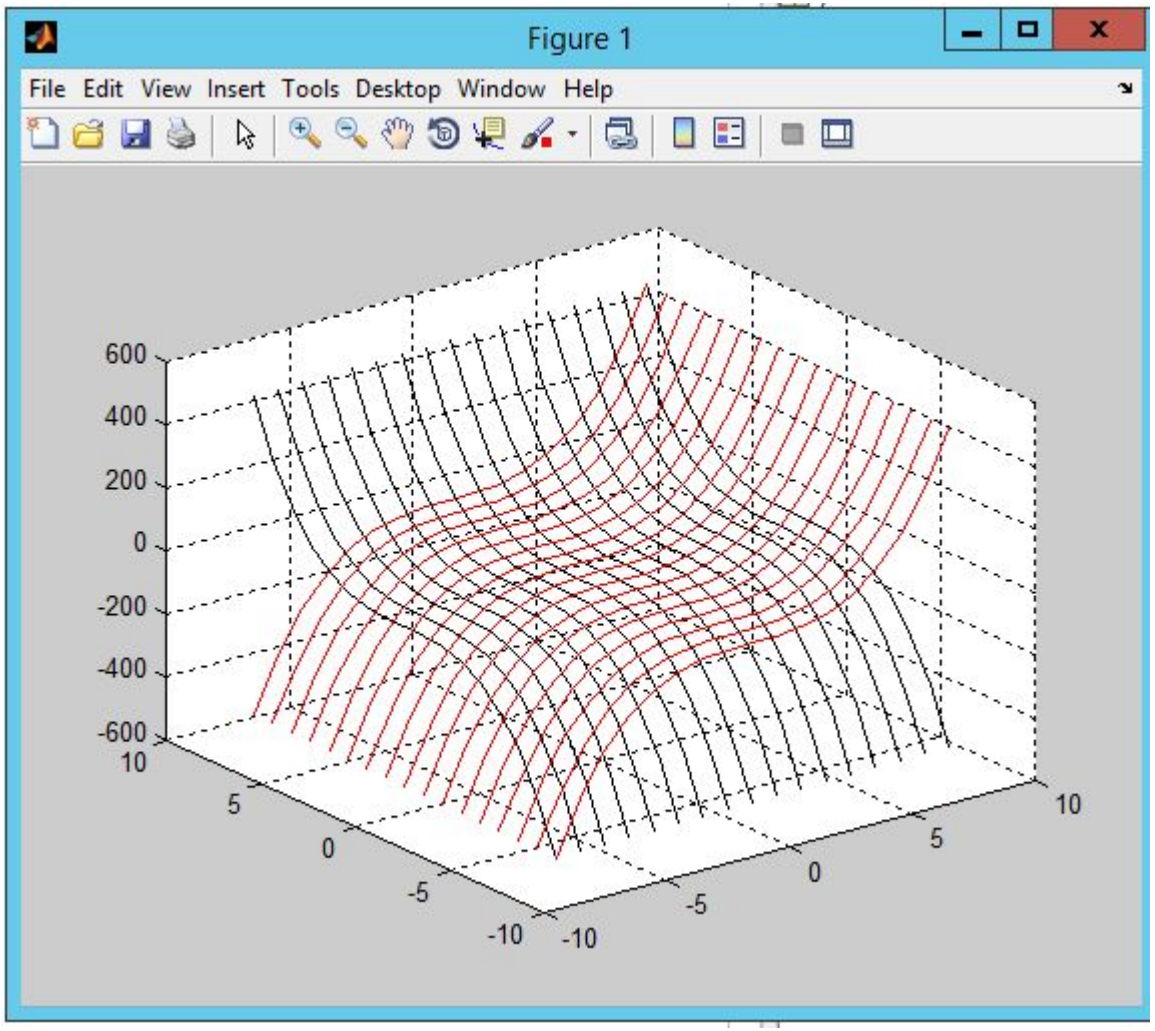
hidden on/off включает или выключает режим удаления невидимых линий (по умолчанию on)

Каркасные поверхности

Команда **plot3(…)** может быть использована для построения так называемой каркасной поверхности, которая представляет собой каркас поверхности геометрического тела, похожий на каркас, сплетенный из проволоки.

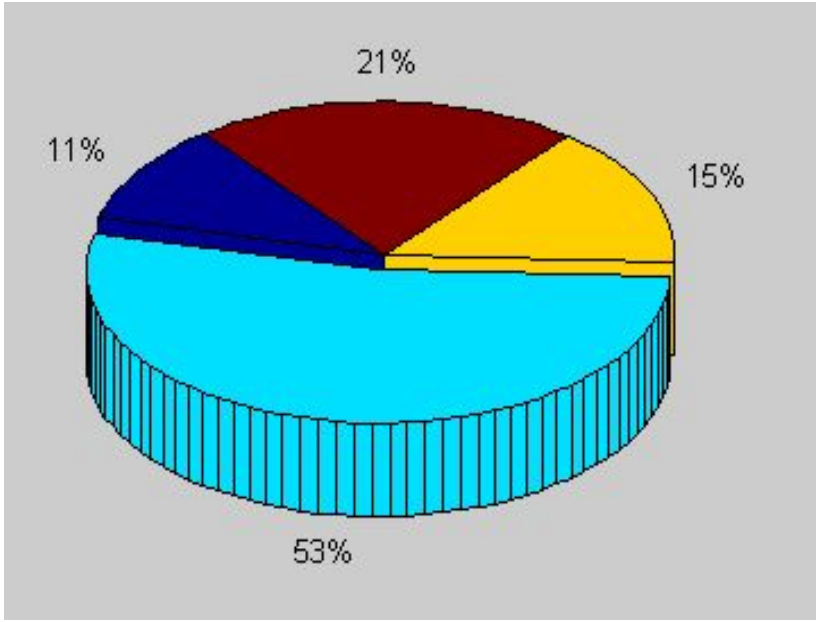
Пример. Построить график каркасной поверхности функции $z(x,y)=x+y^3$, где переменные **x** и **y** изменяются на интервале [-8,8] с шагом 1. Для построения графика использовать непрерывные линии. Линии, расположенные параллельно плоскости **ZOY**, закрасить в черный цвет, а линии, параллельные плоскости **ZOX** – в красный цвет. На плоскостях **XOY**, **ZOX**, **ZOY** сформировать сетку.

```
>> [x,y]=meshgrid(-8:8);
>> z=x+y.^3;
>> plot3(x,y,z,'k',y,x,z,'r')
>> grid on
```



Матричная круговая диаграмма

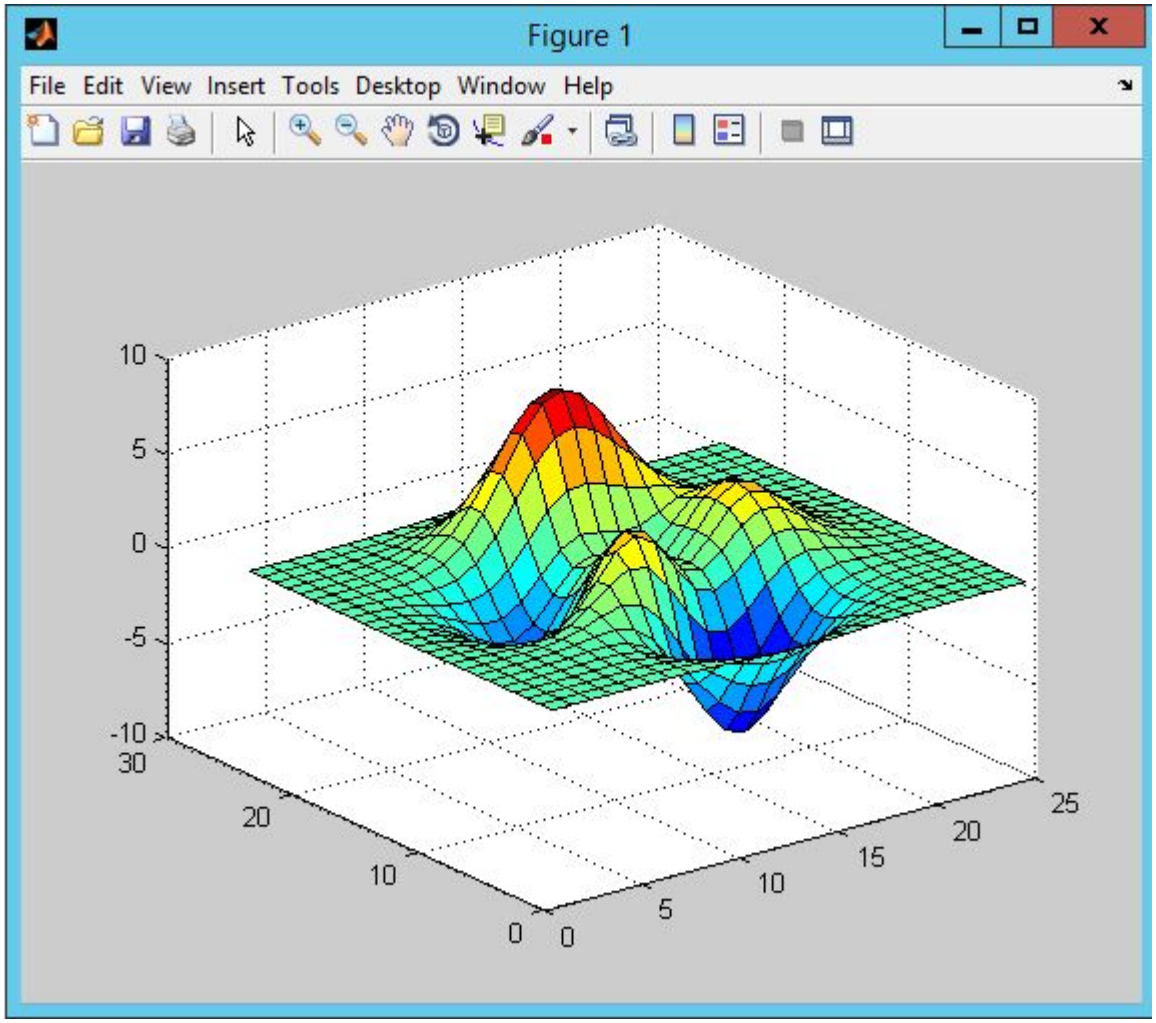
```
>> y=[5 25 7 10];
>> v=[0,1,0,0];
>> pie3(y,v)
```



Матричный образ поверхностей

MATLAB имеет несколько графических функций, возвращающих матричный образ поверхностей. Например, функция **peaks(n)** возвращает матричный образ поверхности с рядом пиков и впадин. Такие функции удобно использовать для проверки работы графических команд трехмерной графики.

```
>> z=peaks(25);
>> surf(z)
```

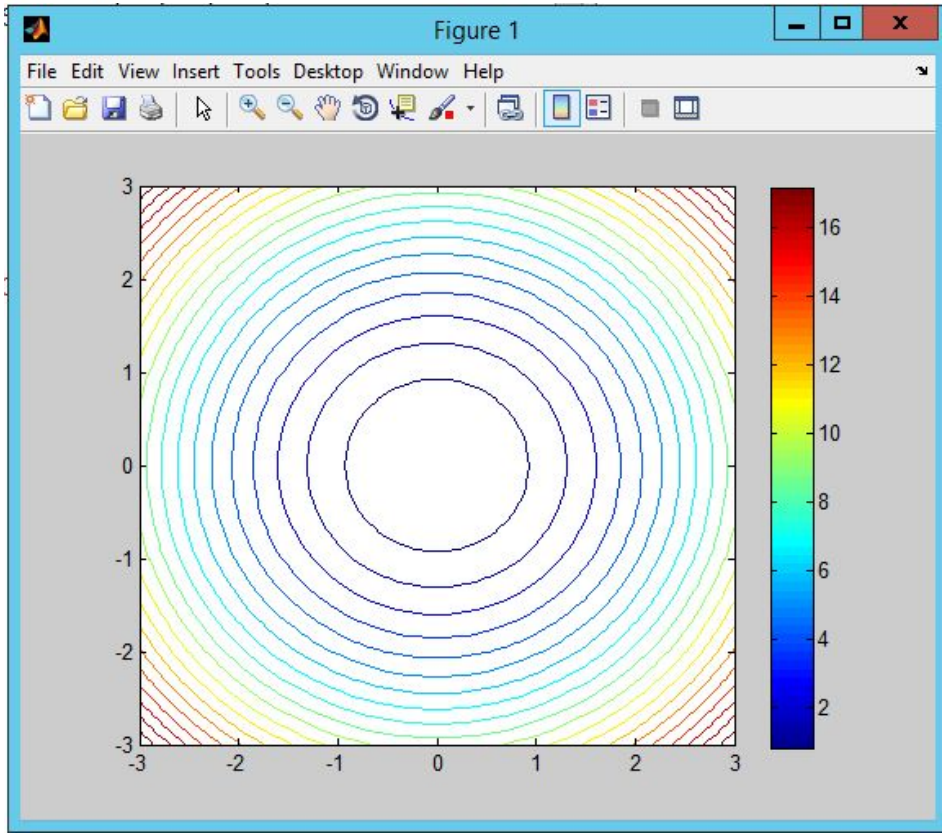


Построение контурных графиков

MatLab позволяет получить различные типы контурных графиков при помощи функций **contour(…)** и **contourf(…)**.

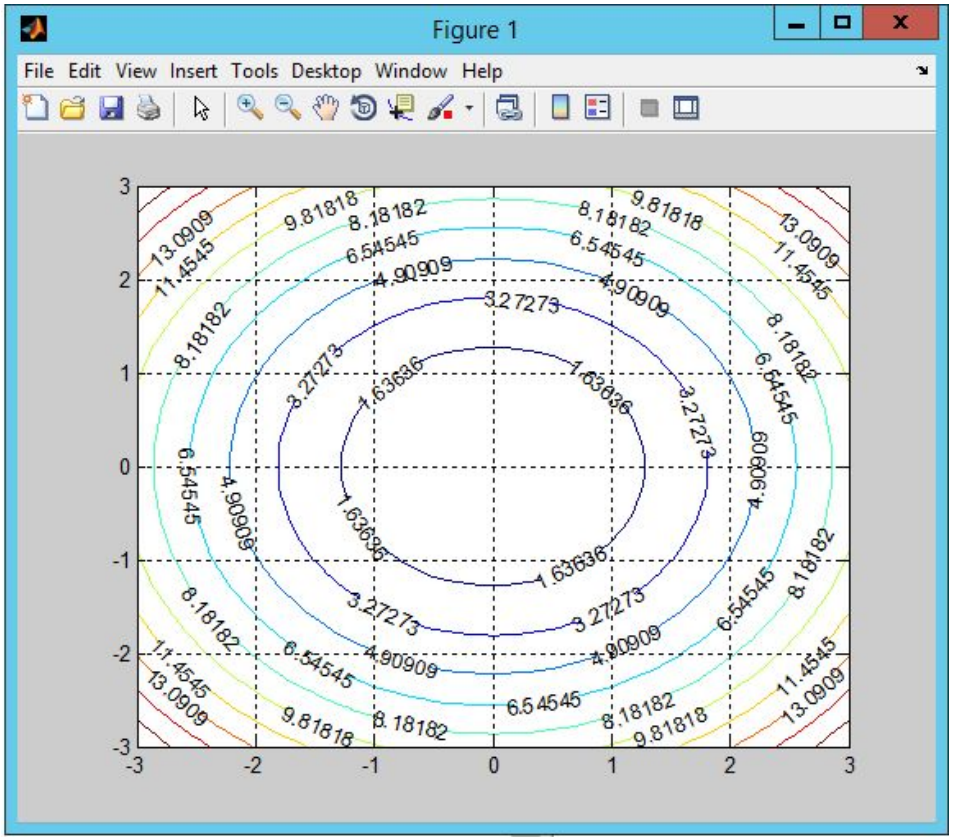
Пример. Построить контурный график функции $z=x^2+y^2$ из двадцати линий. Переменные **x** и **y** изменяются на интервале [-3,3] с шагом 0,1.

```
>> [x,y]=meshgrid(-3:.1:3);
>> z=x.^2+y.^2;
>> contour(x,y,z,20)
>> colorbar
```



Пример. Построить контурный график функции $z=x^2+y^2$, состоящий из десяти оцифрованных линий уровня. Переменные **x** и **y** изменяются на интервале [-3,3] с шагом 0,1.

```
>> [x,y]=meshgrid(-3:.1:3);
>> z=x.^2+y.^2;
>> [c,h]=contour(x,y,z,10);
>> clabel(c,h)
>> grid on
```



Команда clabel(c, h) маркирует линии уровня контура, которые заданы в векторе h.

[C,h]=contour (...) возвращает массив C и вектор дескрипторов, позволяя тем самым продолжить работу с рисунком (давать оцифровку линий).