Тема 5. Средства построения графиков

5.1. Построение графика одной переменной

plot*(x1, y1, s1, x2, y2, s2,...)* – функция построения графиков.

Здесь ***x1, y1*** – массивы значений аргумента (*x1*) и функции (*y1*), отвечающие первой кривой графика; ***x2, y2*** – массивы значений аргумента и функции второй кривой и т.д.

s1, s2,... – символьные переменные (их указание не является обязательным). Любая из них может содержать до трех специальных символов, определяющих соответственно:

а) тип линии, которая соединяет отдельные точки графика;

б) тип точки графика;

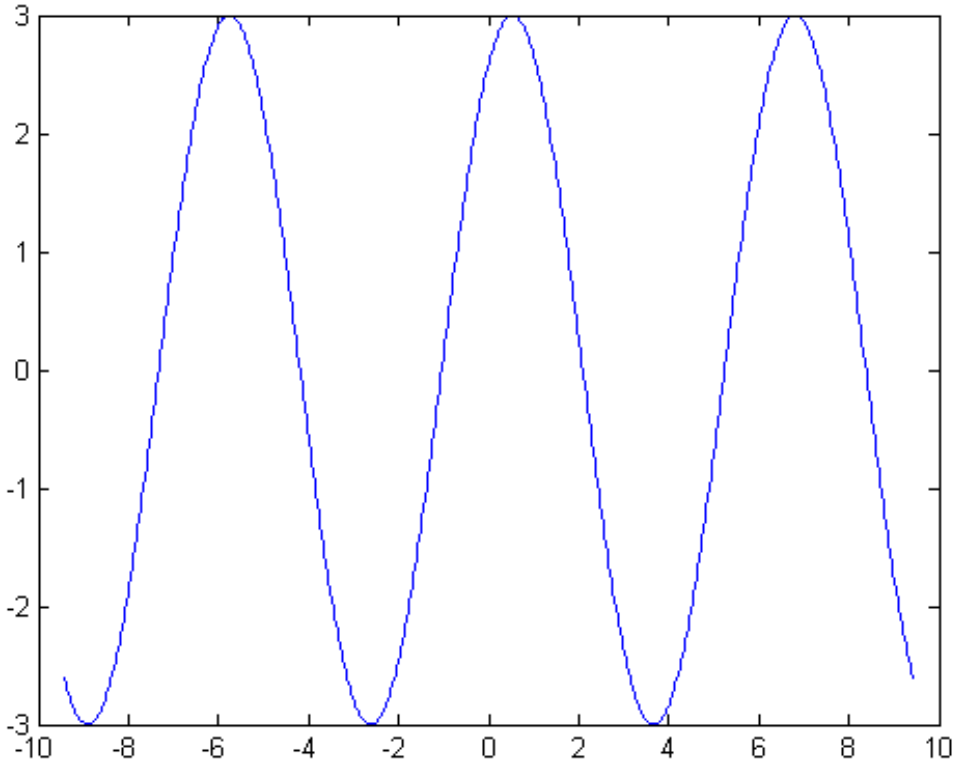
в) цвет линии.

Если переменные *s* не указаны, то тип линии по умолчанию – отрезок прямой, тип точки – пиксел, а цвет устанавливается в такой очередности: синий, зеленый, красный, голубой, фиолетовый, желтый, черный и белый.

Пример:

Построим график функции $y = 3\sin(x + \pi/3)$ на промежутке от -3π до $+3\pi$ с шагом $\pi/100$.

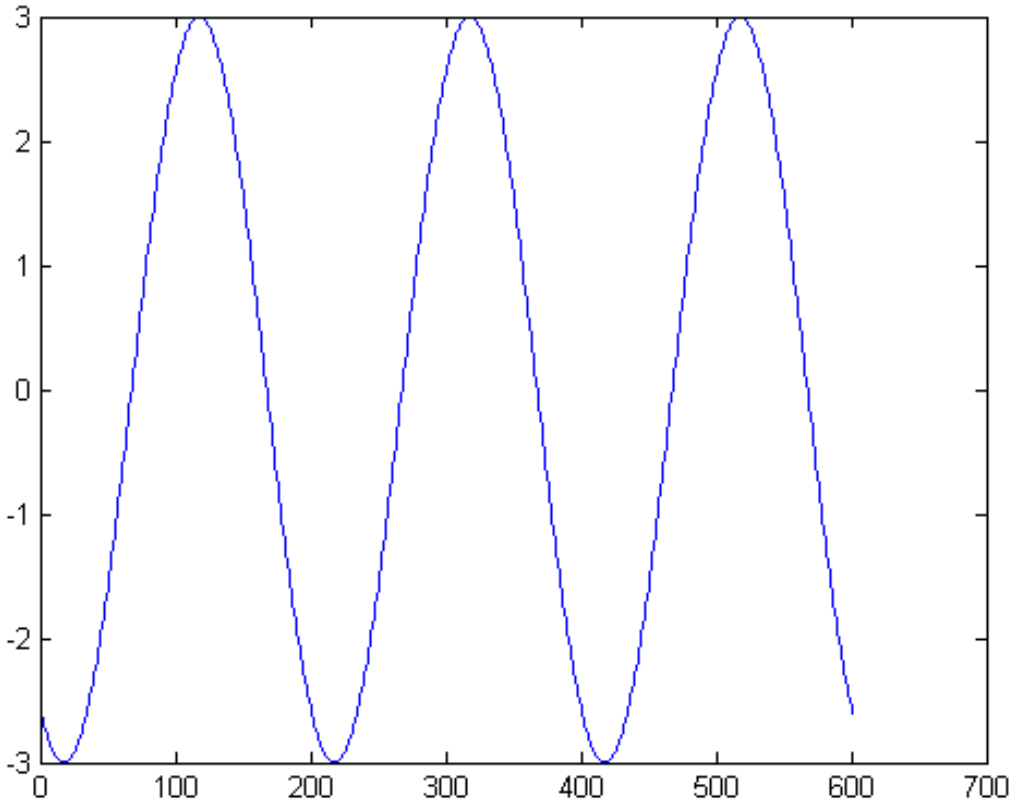
```
» x = -3*pi:pi/100:3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y)
```



Если вектор аргумента при обращении к функции ***plot*** не указан явно, то система по умолчанию принимает в качестве аргумента *номера* элементов вектора функции.

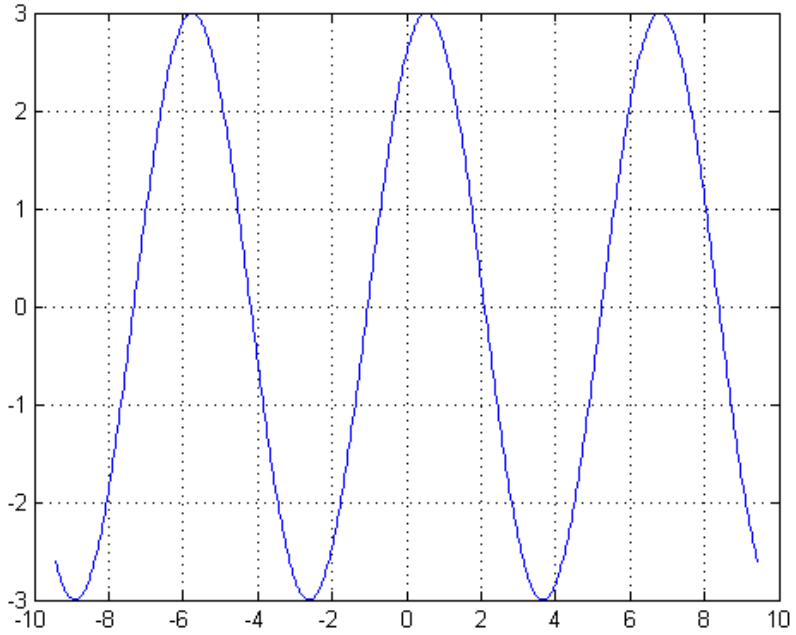
Например:

```
» plot(y)
```



Для нанесения на график сетки из координатных линий служит функция ***grid***. Если к этой функции обратиться сразу после обращения к функции ***plot***:

```
» x = -3*pi:pi/100:3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y), grid
```



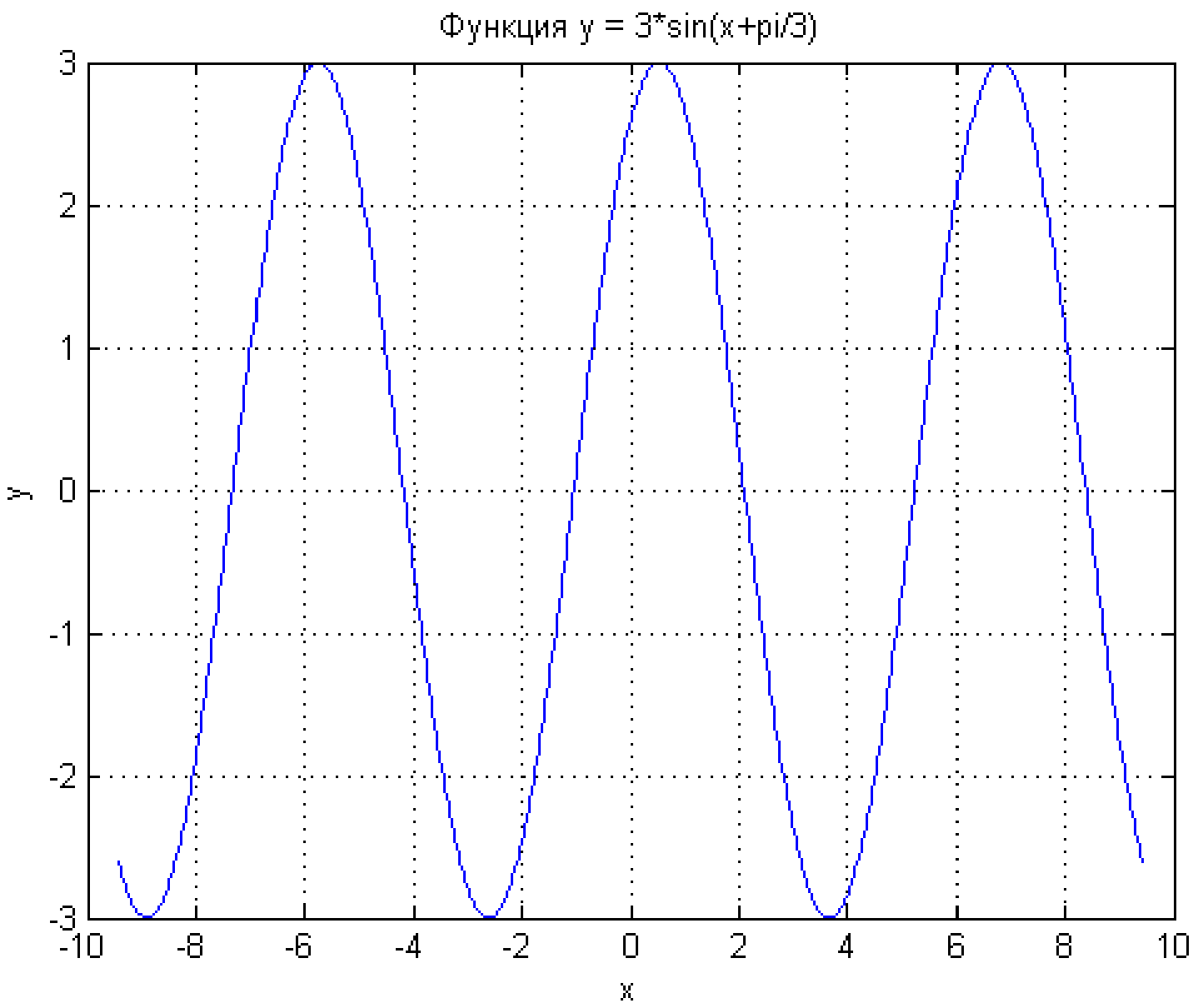
Заголовок графика выводится с помощью процедуры ***title.*** Названия осей – с помощью функции ***xlabel*** (для горизонтальной оси) и ***ylabel*** (для вертикальной оси).

При этом, текст всегда должен помещаться в *апострофы*.

title(‘*<текст>*’)

Пример:

```
» x = -3*pi : pi/100 : 3*pi;
» y = 3*sin(x+pi/3);
» plot(x,y), grid
» title('Функция y = 3*sin(x+pi/3)');
» xlabel('x'); ylabel('y');
```


Функция y = 3*sin(x+pi/3)
3
2
1
0
-1
-2
-3
y
-10
-8
-6
-4
-2
0
2
4
6
8
10
x

fplot – строит функцию заданную в символьном виде в некотором интервале изменения аргумента ***x***.

Синтаксис: *fplot('fun', [lims], tol)*

где

fun – функция, которая заключается в кавычки (*'sin(x)'*), либо перед ней ставится *@(x)*.

lims=[X_{min} X_{max} Y_{min} Y_{max}] – пределы.

tol – относительная ошибка (по умолчанию точность равна 0,2%)

Пример:

```
>> fplot('sin(x)/x', [-15 15])
```

либо

```
>> fplot(@(x) sin(x)/x, 2*pi*[-1 1 -1 1])
```

Если необходимо построить несколько функций, то они заключаются в квадратные скобки.

Пример:

```
>> fplot(@(x) [tan(x), sin(x), cos(x)],
  2*pi*[-1 1 -1 1])
```

или

```
>> fplot('[sin(x)/x, cos(x)]',
   2*pi*[-15 15 -1 1])
```

***bar**(x, y, width)* – построение столбцовых диаграмм,

где

x – аргумент (не должен содержать повторяющихся значений);

y – функция;

width – ширина столбца (от 0 до 1; по умолчанию значение width=0,8).

Пример:

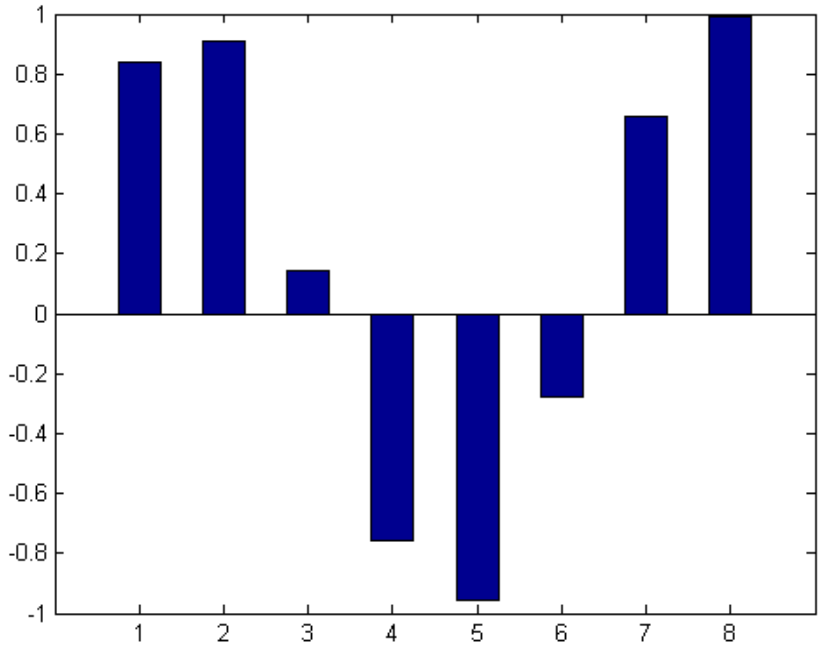
```
» x = [ 1 2 3 4 5 6 7 8]; y = sin(x);
» bar(x, y, 0.5)
```

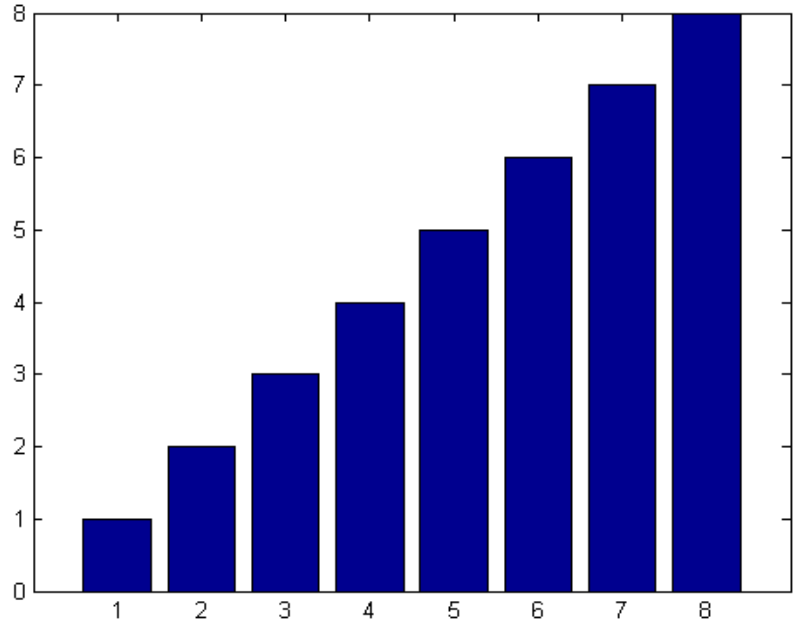
При этом, если не указывать аргумент функции, то в его качестве система принимает номер элемента вектора, график которого строится.

Пример:

```
» bar(x)
```

***barh**(x, y, width)* – горизонтальные столбцы

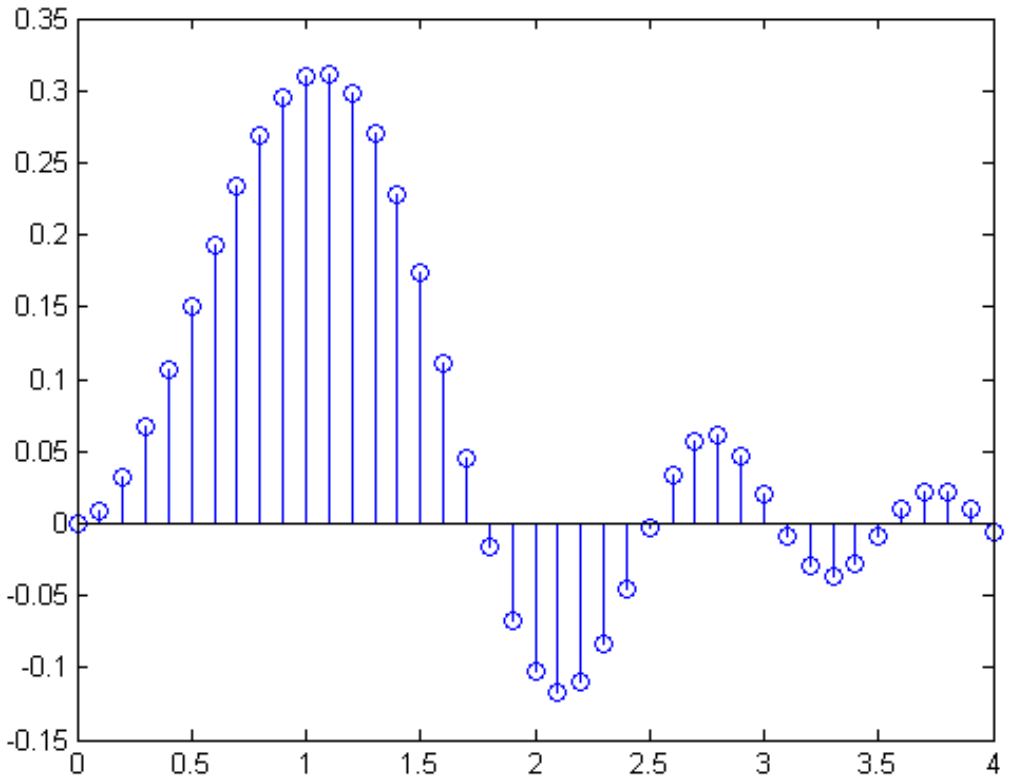

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
2
3
4
5
6
7
8


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

***stem**(x, y)* или ***stem** (y)* – строит графики дискретных отсчётов функций. Удобно использовать при квантовании сигнала или представлении ряда Фурье.

Пример:

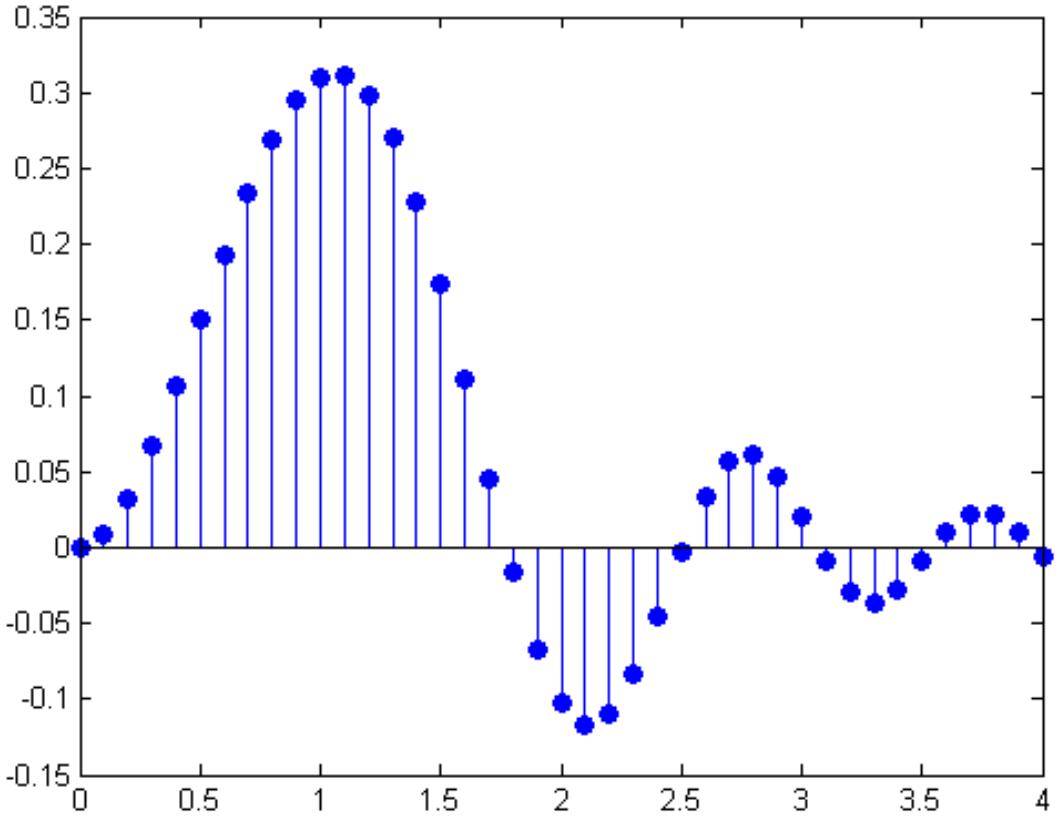
```
>> x=0:0.1:4;
>> y=sin(x.^2).*exp(-x);
>> stem(x,y)
```



***stem**(x, y, 'filled')*– строит график функции с закрашенными маркерами.

Пример:

```
>> stem(x,y,'filled')
```



***hist**(y, x)* – построение графика *гистограммы* заданного вектора.

где **y** – вектор, гистограмму которого нужно построить; **x** – вектор, элементы его определяют интервалы изменения первого вектора, внутри которых подсчитывается количество элементов вектора “y”.

Эта функция осуществляет две операции:

– подсчитывает количество элементов вектора “y”, значения которых попадают внутрь соответствующего диапазона, указанного вектором “x”;

– строит столбцовую диаграмму подсчитанных чисел элементов вектора “y” как функцию диапазонов, указанных вектором “x”.

Пример: гистограмма случайных величин для диапазона изменения этих величин от -2,9 до +2,9.

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

stairs*(x, y)* или ***stairs*** *(y)* – строит лестничные графики, представляющие собой ступеньки огибающие функцию y(x).

Пример:

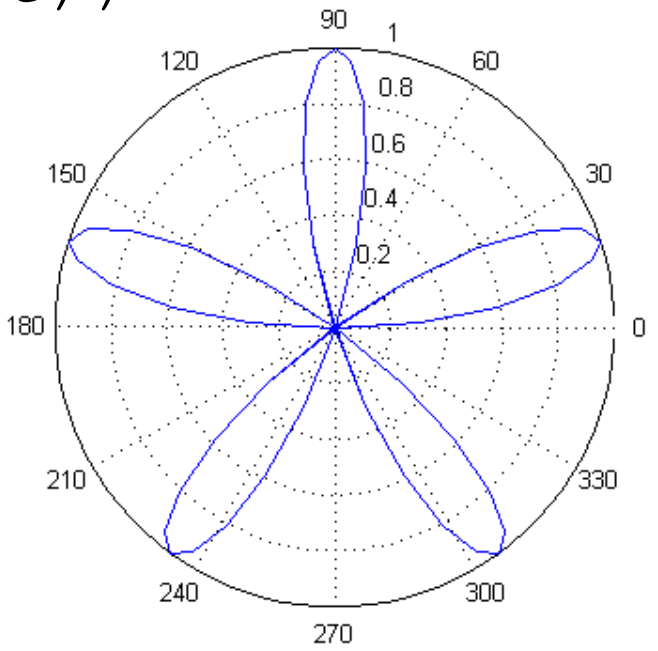
```
>> x=0:0.25:10;
>> stairs(x,x.^2)
```

Функция [N, M]=**stairs**(x, y) сама по себе график не строит, а возвращает векторы N и M по которым можно построить график с помощью команды plot(N, M).

polar(*Θ, ρ*) – строит график в полярной системе координат, представляющий положение конца радиус-вектора с длиной *ρ* и углом *Θ*.

Пример:

```
>> t=0:pi/50:2*pi;
>> polar(t, sin(5*t))
```



Для представления радиус-векторов в их обычном виде (в виде стрелок исходящих из начала координат) служит команда

compass*(U, V)* – строит графики радиус-векторов с действительной (*U*) и мнимой (*V*) частью.

compass*(Z)* – эквивалентно compass(real(Z), imag(Z)).

Пример:

```
>> Z=[-1+2i, -2-3i, 2+3i, 5+2i];
>> compass(Z)
```

***feather**(U, V)* – строит график проекции векторов, заданных действительной (*U*) и мнимой (*V*) частью, на плоскость.

***feather**(Z)* – эквивалентно feather(real(Z), imag(Z)).

Пример:

```
>> x=0:0.1*pi:3*pi;
>> y=0.05+i;
>> z=exp(x*y);
>> feather(z)
```

***errorbar**(X, Y, L, U)* – строит график значений элементов вектора *Y* в зависимости от данных, содержащихся в векторе *X* с указанием нижней и верхней границ значений, заданных в векторах *L* и *U*.

***errorbar**(X, Y, E)* – строит графики функции *Y(X)* с указанием границ в виде [Y-E Y+E], где E – погрешность.

Пример:

```
>> x=-2:0.1:2;
>> y=erf(x);
>> e=rand(size(x))/10;
>> errorbar(x,y,e)
```

***loglog**(X, Y)* – строит график в логарифмическом масштабе.

Пример:

```
>> x=logspace(-1,3);
>> loglog(x,exp(x)./x);
>> grid
```

где

logspace(x1, x2) генерирует вектор из 50 логарифмически равноудаленными точками в промежутке от 10^x1 до 10^x2.

***semilogx**(X, Y)* – строит график функции в логарифмическом масштабе по оси *X* и линейном по оси *Y*.

***semilogy**(X, Y)* – наоборот.

Пример:

```
>> x=0:0.5:10;
>> semilogy(x, exp(x))
>> grid
```

pie(X) – строит круговую диаграмму по данным нормализованного вектора X/ SUM(X). SUM(X)— сумма элементов вектора. Если SUM(X)<=1.0, то значения в X непосредственно определяют площадь секторов;

pie(X,EXPLODE) – строит круговую диаграмму, у которой отрыв секторов от центра задается вектором EXPLODE, который должен иметь тот же размер, что и вектор данных X.

Следующий *пример* строит цветную круговую диаграмму с пятью секторами, причем последний сектор отделен от остальных:

```
» X=[1 2 3 4 5];
» pie(X,[0 0 0 0 2])
```

pie3(…) – аналогична команде ***pie(…)***, но дает построение объемных секторов

5.2. Построение трехмерных графиков

meshgrid – преобразовывает область определяемую векторами x и y в массив (сетку) размером X на Y которая может использоваться для построения графиков 2-х переменных и 3-х мерных графиков.

Синтаксис:

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

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

***meshc**(X, Y, Z)* – комбинированное изображение сетки и контура.

Аналогично: ***mesh**(X, Y, Z)*, ***meshz**(X, Y, Z)*

Пример:

```
>> [X,Y]=meshgrid(-5:0.1:5);
>> Z=X.*sin(X+Y);
>> meshc(X,Y,Z)
```

***surf**(X, Y, Z)* – строит 3-х мерную цветную поверхность.

Аналогично : ***surfc**(X, Y, Z)*, ***surfl**(X, Y, Z)*

Пример:

```
>> [X,Y]=meshgrid(-2:0.2:2);
>> Z=X.*exp(-X.^2-Y.^2);
>> surf(X,Y,Z)
```

***quiver**(X, Y, U, V)* – строит график поля градиентов в виде стрелок для каждой пары элементов массивов *X* и *Y*, причем элементы *U* и *V* указывают направление и размер стрелок.

***quiver**(X, Y)* – строит векторы скорости в равнорасположенных точках на плоскости.

Пример:

```
>> x=-2:0.2:2; y=-1:0.2:1;
>> [a,b]=meshgrid(x,y);
>> c=a.*exp(-a.^2-b.^2);
>> [d,e]=gradient(c, 0.2, 0.2);
>> quiver(a, b,d,e)
```

где [FX,FY] = gradient(F) вычисляет численный градиент матрицы F. FX соответствует dF/dx, FY – dF/dy

***plot3**(X, Y, Z)* – строит массив точек представленных векторами X, Y, Z, соединяя их отрезками прямых.

Пример 1:

```
>> [X,Y]=meshgrid(-3:0.15:3);
>> Z=X.^2+Y.^2;
>> plot3(X,Y,Z)
```

Пример 2:

```
>> t=0:pi/50:10*pi;
>> plot3(sin(t),cos(t), t)
```

***slice**(X, Y, Z, V, Sx, Sy, Sz, 'method') –* строит плоские сечения объемной фигуры V в направлении осей X, Y, Z с позициями задаваемыми векторами *Sx, Sy, Sz.*

Пример:

```
>>[x,y,z]=meshgrid(-2:0.2:2,-2:0.25:2,-2:0.16:2);
>> v=x.*exp(-x.^2-y.^2-z.^2);
>> slice(x,y,z,v,[-1.2 0.8 2], 2, [-2 -0.2])
```

5.3. Дополнительные функции графического окна

Управление свойствами осей графиков

Обычно графики выводятся в режиме автоматического масштабирования. Для установления других режимов масштабирования используется процедура ***axis***.

axis([x_{min} x_{max} y_{min} y_{max}]) устанавливает диапазон координат по осям.

axis auto установка параметров осей по умолчанию.

axis ij перемещает начало отсчета в левый верхний угол и реализует отсчет от него (матричная система координат).

axis xy возвращает декартову систему координат с началом отсчета в левом нижнем углу графика.

axis square устанавливает одинаковый диапазон изменения переменных по осям графика.

axis equal обеспечивает одинаковый масштаб с одинаковым расстоянием между метками.

axis normal восстанавливает масштаб отменяя установки *axis equal* и *axis square*.

axis on/off восстанавливает/убирает обозначения осей и их маркеров.

Наложение графиков друг на друга

Во многих случаях желательно построение многих наложенных друг на друга графиков в одном и том же окне. Для этого служит команда продолжения графических построений hold. Она используется в следующих формах:

hold on – обеспечивает продолжение вывода графиков в текущее окно, что позволяет добавлять последующие графики к уже существующим;

hold off – отменяет режим продолжения графических построений;

hold – работает как переключатель, последовательно включая режим продолжения графических построений и отменяя его.

Разбиение графического окна

Используя процедуру ***subplot*** можно в одном графическом окне, но на отдельных графических полях построить несколько графиков.

Обращение к этой процедуре должно предшествовать обращению к процедурам *plot*, *bar* и т.д.

subplot(m,n,p) – разбивает графическое окно на *mxn* подокон.

здесь ***m*** – число подокон по вертикали, ***n*** – по горизонтали, а ***p*** – номер подокна, в котором будет строиться график.

При этом подокна нумеруются слева направо построчно сверху вниз.

Пример:

```
>> x=-5:0.1:5;
>> subplot(3,1,1), plot(x, sin(x)); grid
>> subplot(3,1,2), plot(x, cos(2*x)); grid
>> subplot(3,1,3), plot(x, tan(x)); grid
```