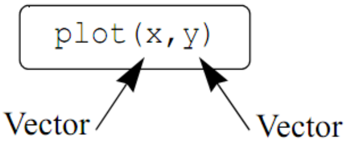
Lecture5

MATLAB plots

-plot command

The `plot` command is used to create two-dimensional plots. The simplest form of the command is:

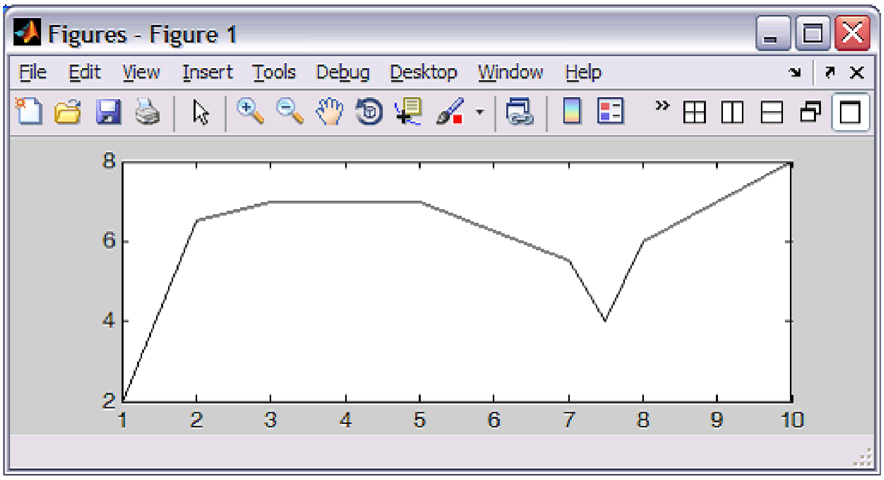


The figure that is created has axes with a linear scale and default range. For example, if a vector x has the elements 1, 2, 3, 5, 7, 7.5, 8, 10, and a vector y has the elements 2, 6.5, 7, 7, 5.5, 4, 6, 8, a simple plot of y versus x can be created bytyping the following in the Command Window:

>> x=[1 2 3 5 7 7.5 8 10];

>> y=[2 6.5 7 7 5.5 4 6 8];

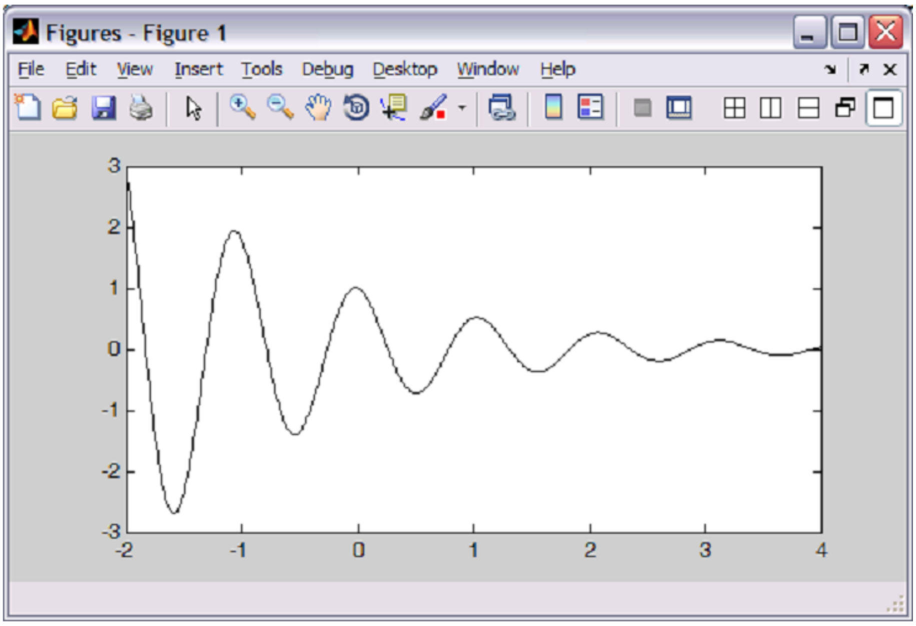
>> plot(x,y)



As an example, the `plot` command is used to plot the function $y = 3.5^{-0.5x}\cos(6x)$ for $-2 \le x \le 4$. A program that plots this function is shown in the following script file.

```
% A script file that creates a plot of
% the function: 3.5.^(-0.5*x).*cos(6x)
x=[-2:0.01:4];
y=3.5.^(-0.5*x).*cos(6*x);
plot(x,y)
```

- `x=[-2:0.01:4];` — Create vector x with the domain of the function.
- `y=3.5.^(-0.5*x).*cos(6*x);` — Create vector y with the function value at each x.
- `plot(x,y)` — Plot y as a function of x.



-THE *fplot* COMMAND

The `fplot` command plots a function with the form $y = f(x)$ between specified limits. The command has the form:

```
fplot('function',limits,'line specifiers')
```

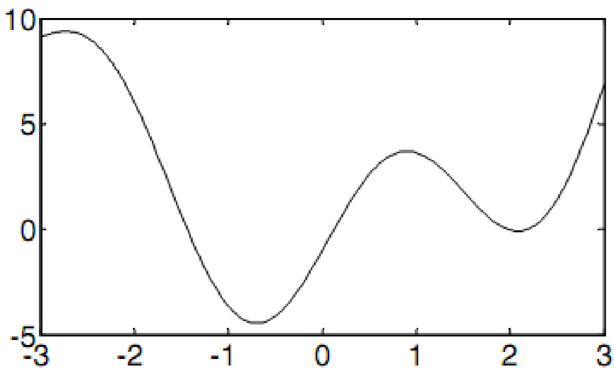
- The function to be plotted.
- The domain of x and, optionally, the limits of the y axis.
- Specifiers that define the type and color of the line and markers (optional).

For example, a plot of the function $y = x^2 + 4\sin(2x) - 1$ for $-3 \le x \le 3$

be created with the `fplot` command by typing:

```
>> fplot('x^2+4*sin(2*x)-1',[-3 3])
```

The figure that is obtained in the Figure Window is shown in Figure



-Plotting multiple graphs in the same plot

In many situations there is a need to make several graphs in the same plot. One is by using the plot command, the second is by using the hold on and hold off commands, and the third is by using the line command

-Using the plot Command

Two or more graphs can be created in the same plot by typing pairs of vacters inside **plot command**

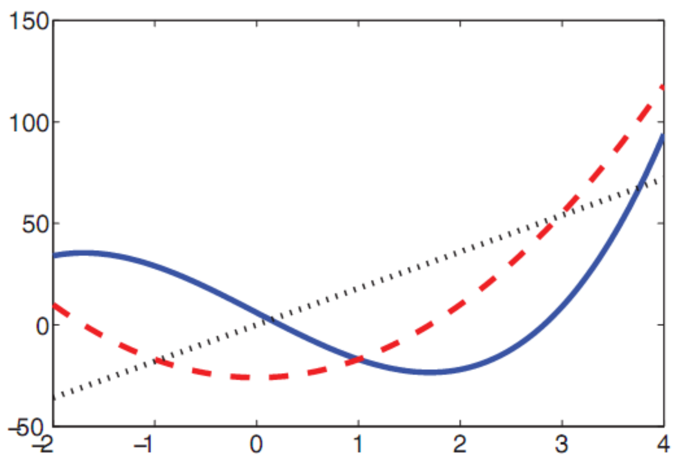
Example:

Plot the function $y = 3x^3 - 26x + 10$, and its first and second derivatives, for $-2 \le x \le 4$, all in the same plot.

The first derivative of the function is: $y' = 9x^2 - 26$.
The second derivative of the function is: $y'' = 18x$.
A script file that creates a vector x and calculates the values of y, y', and y'' is:

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```

- `x=[-2:0.01:4];` — Create vector x with the domain of the function.
- `y=3*x.^3-26*x+6;` — Create vector y with the function value at each x.
- `yd=9*x.^2-26;` — Create vector yd with values of the first derivative.
- `ydd=18*x;` — Create vector ydd with values of the second derivative.
- `plot(x,y,'-b',x,yd,'--r',x,ydd,':k')` — Create three graphs, y vs. x, yd vs. x, and ydd vs. x, in the same figure.



2. Using the *hold on* and *hold off* Commands

To plot several graphs using the hold on and hold off commands, one graph is plotted first with the plot command. Then the hold on command is typed. This keeps the Figure Window with the first plot open, including the axis properties and formatting.

Example\

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b')          The first graph is created.
hold on
plot(x,yd,'--r')        Two more graphs are added to the figure.
plot(x,ydd,':k')
hold off
```

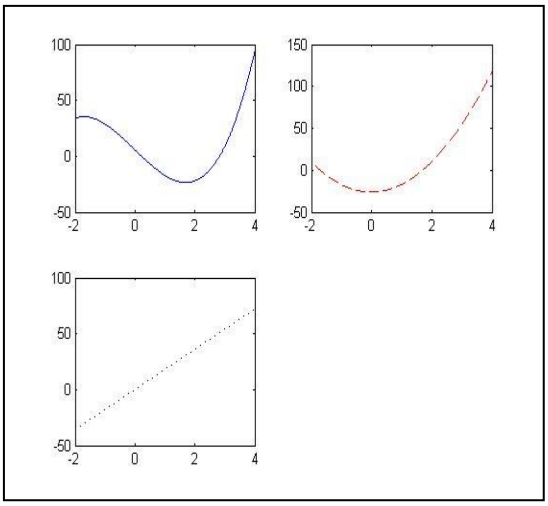
-Putting multiple plots on the same page

Multiple plots can be created on the same page with the subplot command, which has the form:

```
subplot(m,n,p)
```

Example:

```
clear all
clc
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
subplot(2,2,1),plot(x,y,'-b')
subplot(2,2,2),plot(x,yd,'--r')
subplot(2,2,3),plot(x,ydd,':k')
```



-Plots With Special Graphics

Plots with **error bars** can be done in MATLAB with the error bar command.

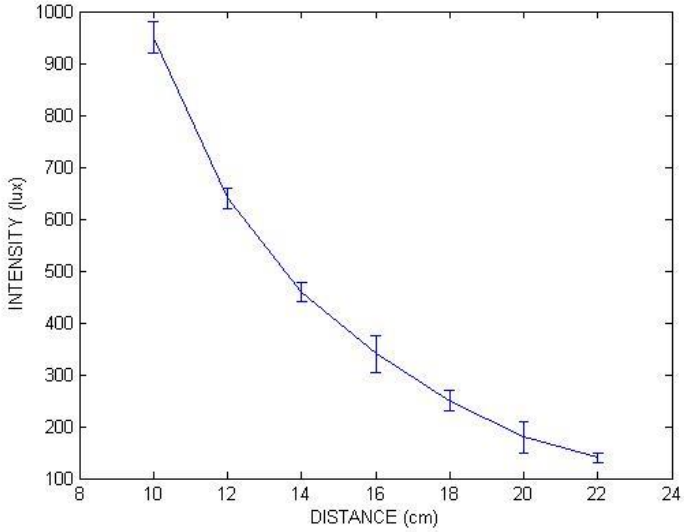
```
errorbar(x,y,e)
```

Vectors with horizontal and vertical coordinates of each point.

Vector with the value of the error at each point.

The plot in Figure, which has symmetric error bars, was done by executing the following code:

```
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
ydErr=[30 20 18 35 20 30 10]
errorbar(xd,yd,ydErr)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
```



3-Histogram plot

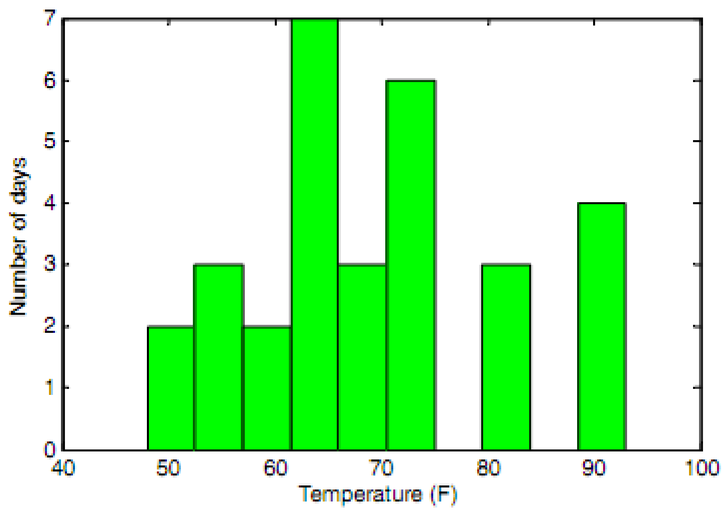
Histograms are created in MATLAB with the **hist command.** The simplest form of the command is:

```
hist(y)
```

y is a vector with the data points. MATLAB divides the range of the data points into 10 equally spaced sub ranges **(bins)** and then plots the number of data points in each bin.

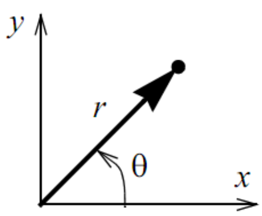
example, the following data points are the daily maximum temperature (in F) in Washington, DC, during the month of April 2002: 58 73 73 53 50 48 5673 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69 (data from the U.S. National Oceanic and Atmospheric Administration). A histogram of this data is obtained with the commands:

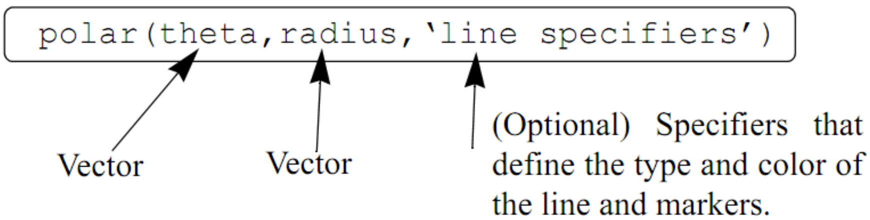
```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89
91 80 59 69 56 64 63 66 64 74 63 69];
>> hist(y)
```



5. **Polar** Plots

Polar coordinates, in which the position of a point in a plane is defined by the angle θ and the radius (distance) to the point, are frequently used in the solution of science and engineering problems. The `polar` command is used to plot functions in polar coordinates. The command has the form:

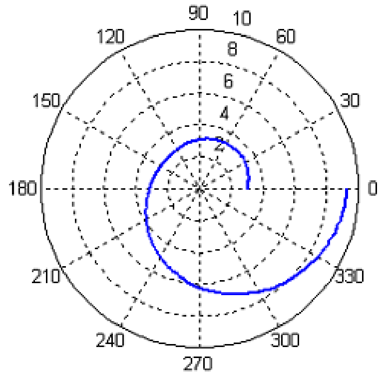




element calculations. The two vectors are then used in the `polar` command.

For example, a plot of the function $r = 3\cos^2(0.5\theta) + \theta$ for $0 \le \theta \le 2\pi$ is shown below.

```
t=linspace(0,2*pi,200);
r=3*cos(0.5*t).^2+t;
polar(t,r)
```



Lecture6

-Three-Dimensional Plots

1. *Line Plots*

A three-dimensional line plot is a line that is obtained by connecting points in three-dimensional space. A basic 3-D plot 'is created with the `plot3` command, which is very similar to the `plot` command and has the form:

```
plot3(x,y,z,'line specifiers','PropertyName',property value)
```

x, y, and z are vectors of the coordinates of the points.

(Optional) Specifiers that define the type and color of the line and markers.

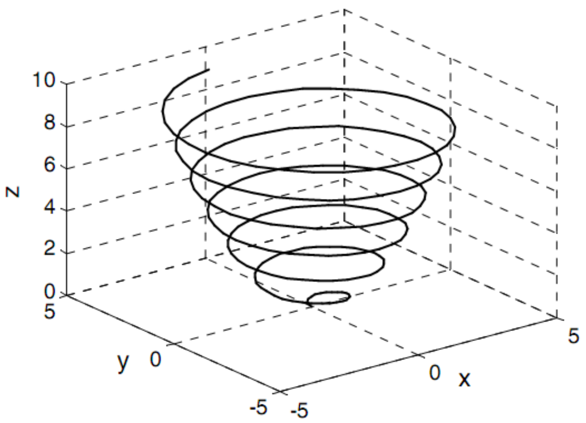
(Optional) Properties with values that can be used to specify the line width, and marker's size and edge and fill colors.

For example, if the coordinates x, y, and z are given as a function of the parameter t by

$$x = \sqrt{t}\sin(2t)$$
$$y = \sqrt{t}\cos(2t)$$
$$z = 0.5t$$

a plot of the points for $0 \le t \le 6\pi$ can be produced by the following script file:

```
t=0:0.1:6*pi;
x=sqrt(t).*sin(2*t);
y=sqrt(t).*cos(2*t);
z=0.5*t;
```



plot3(x,y,z,'k','linewidth',1)

grid on

xlabel('x'); ylabel('y'); zlabel('z')

Making mesh and surface plots:

A mesh or surface plot is created with the `mesh` or `surf` command, which has the form:

```
mesh(X,Y,Z)
```

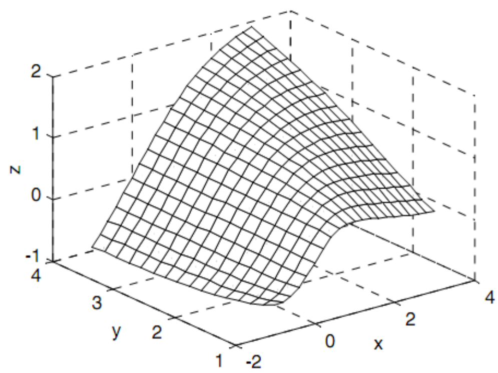
```
surf(X,Y,Z)
```

where X and Y are matrices with the coordinates of the grid and Z is a matrix with the value of z at the grid points. The mesh plot is made of lines that connect the points. In the surface plot, areas within the mesh lines are colored.

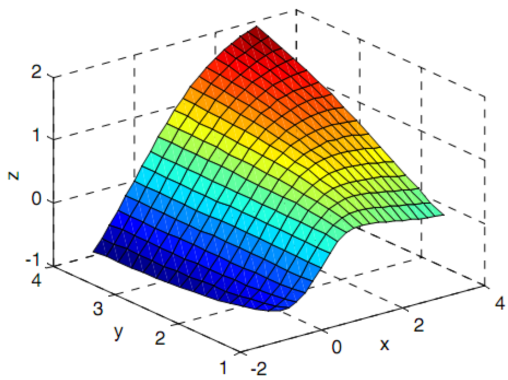
As an example, the following script file contains a complete program that creates the grid and then makes a mesh (or surface) plot of the function $z = \frac{xy^2}{x^2+y^2}$ over the domain $-1 \le x \le 3$ and $1 \le y \le 4$.

```
x=-1:0.1:3;
y=1:0.1:4;
[X,Y]=meshgrid(x,y);
Z=X.*Y.^2./(X.^2+Y.^2);
mesh(X,Y,Z)                    Type surf(X,Y,Z) for surface plot.
xlabel('x'); ylabel('y'); zlabel('z')
```

Note that in the program above the vectors x and y have a much smaller spacing than the spacing earlier in the section. The smaller spacing creates a denser grid. The figures created by the program are:



Mesh plot



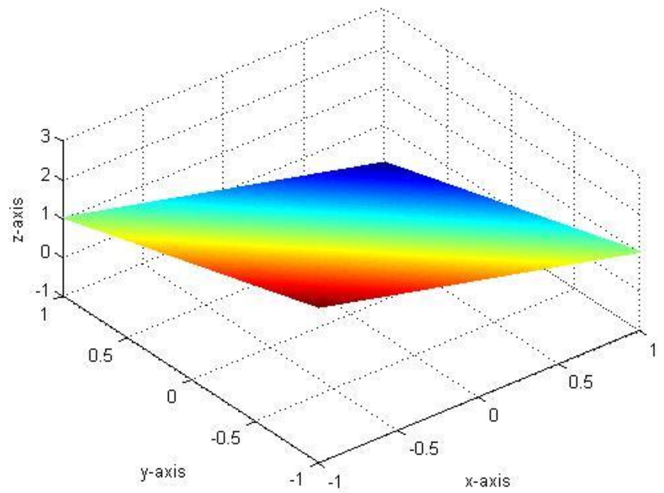
Surface plot

Ex.

1. plot x+y+z=1 in 3D?

Solution:

```
x=-1:.01:1;
y=x;
[x y]=meshgrid(x,y);
z=1-x-y;
mesh(x,y,z)
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```



2. plot $x^2+y^2=z^2$ (Cone) in 3D

```
x=-1:.01:1;
y=x;
[x y]=meshgrid(x,y);
z1=sqrt(x.^2+y.^2);
z2=-sqrt(x.^2+y.^2);
      mesh(z1)
   hold on
    mesh(z2)

xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
```