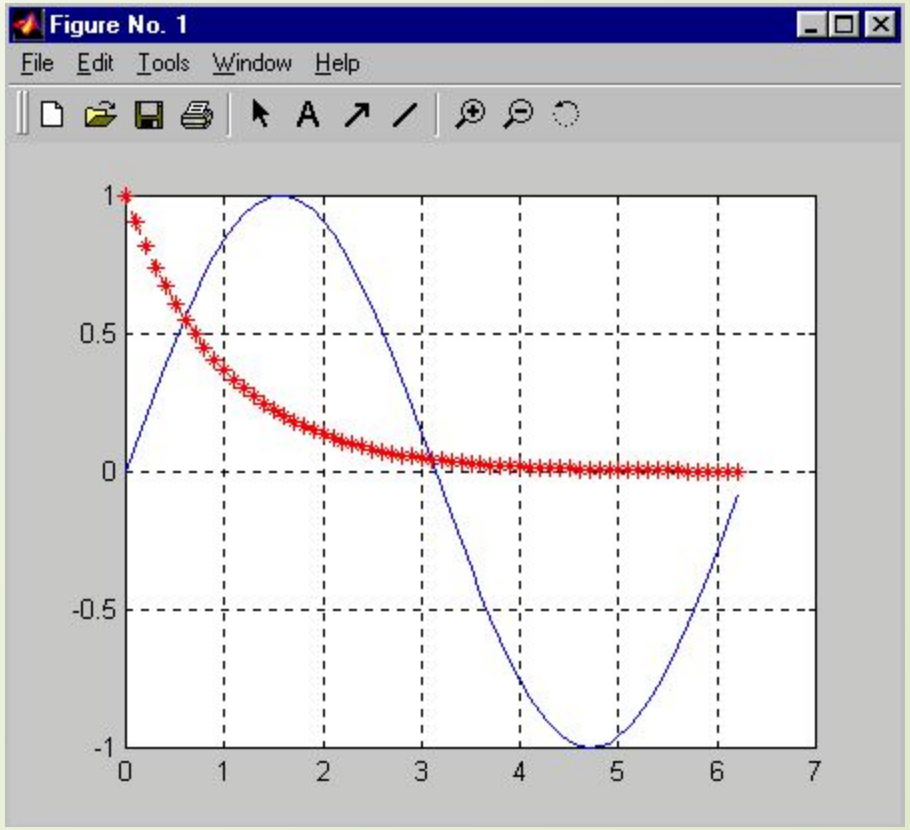
2D plotting

Adding plots to a figure

- HOLD ON holds the current plot
- HOLD OFF releases hold on current plot
- HOLD toggles the hold state

```
» x = 0:.1:2*pi;
» y = sin(x);
» plot(x,y,'b')
» grid on
» hold on
» plot(x,exp(-x),'r:*')
```



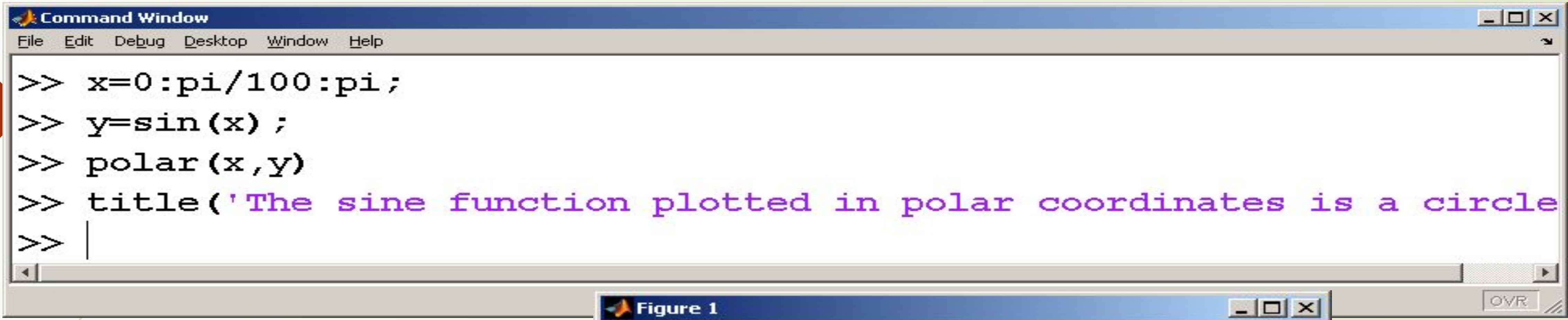
Types of 2-D Plots

- Polar Plots
- Logarithmic Plots
- Bar Graphs
- Pie Charts
- Histograms
- X-Y graphs with 2 y axes
- Function Plot

Polar Plots

- Some functions are easier to specify using polar coordinates than by using rectangular coordinates
- For example the equation of a circle is
  - y=sin(x)

  in polar coordinates


Command Window
File Edit Debug Desktop Window Help
>> x=0:pi/100:pi;
>> y=sin(x);
>> polar(x,y)
>> title('The sine function plotted in polar coordinates is a circle
>>
OVR

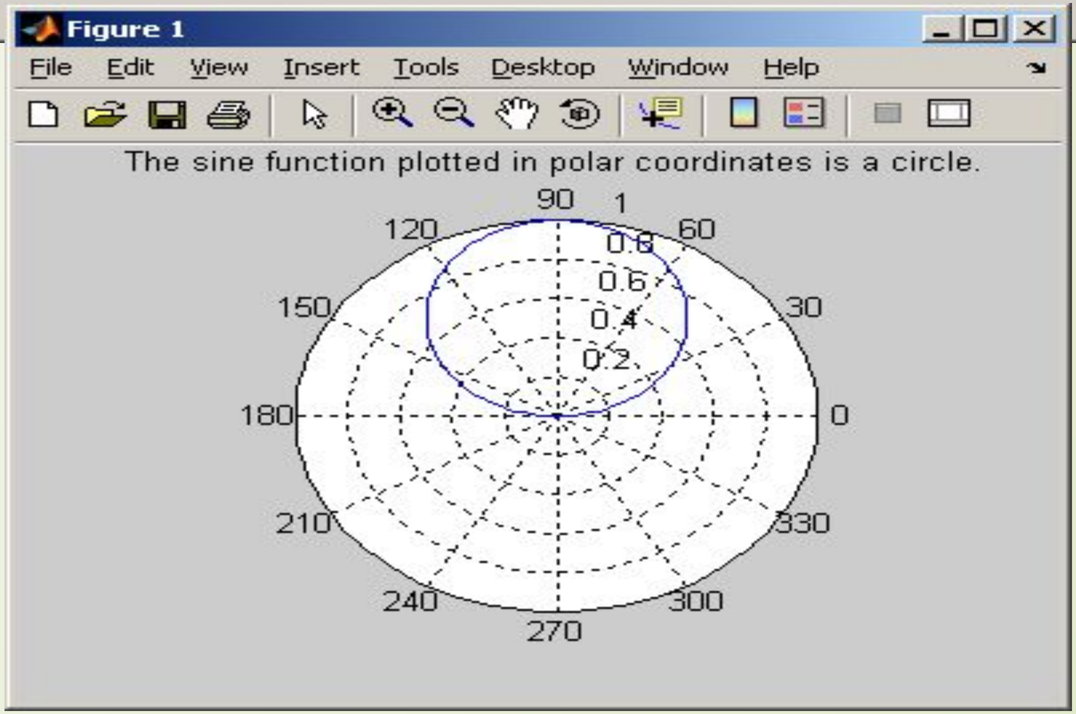


Figure 1
File Edit View Insert Tools Desktop Window Help
The sine function plotted in polar coordinates is a circle.
90
1
120
60
0.8
0.6
150
30
0.4
0.2
180
0
210
330
240
300
270

Logarithmic Plots

- A logarithmic scale (base 10) is convenient when
  - a variable ranges over many orders of magnitude, because the wide range of values can be graphed, without compressing the smaller values.
  - data varies exponentially.

- `plot` – uses a linear scale on both axes
- `semilogy` – uses a $\log_{10}$ scale on the y axis
- `semilogx` – uses a $\log_{10}$ scale on the x axis
- `loglog` – use a $\log_{10}$ scale on both axes

Editor - Untitled2*

```
x = 0:0.5:50;
y = 5*x.^2;
subplot(2,2,1)
plot(x,y)
    title('Polynomial - linear/linear')
    ylabel('y'), grid
subplot(2,2,2)
semilogx(x,y)
    title('Polynomial - log/linear')
    ylabel('y'), grid
subplot(2,2,3)
semilogy(x,y)
    title('Polynomial - linear/log')
    xlabel('x'), ylabel('y'), grid
subplot(2,2,4)
loglog(x,y)
    title('Polynomial - log/log')
    xlabel('x'), ylabel('y'), grid
```

x-y plot – linear on both axes

semilogx – log scale on the x axis

semilogy – log scale on the y axis

loglog – log scale on both axes

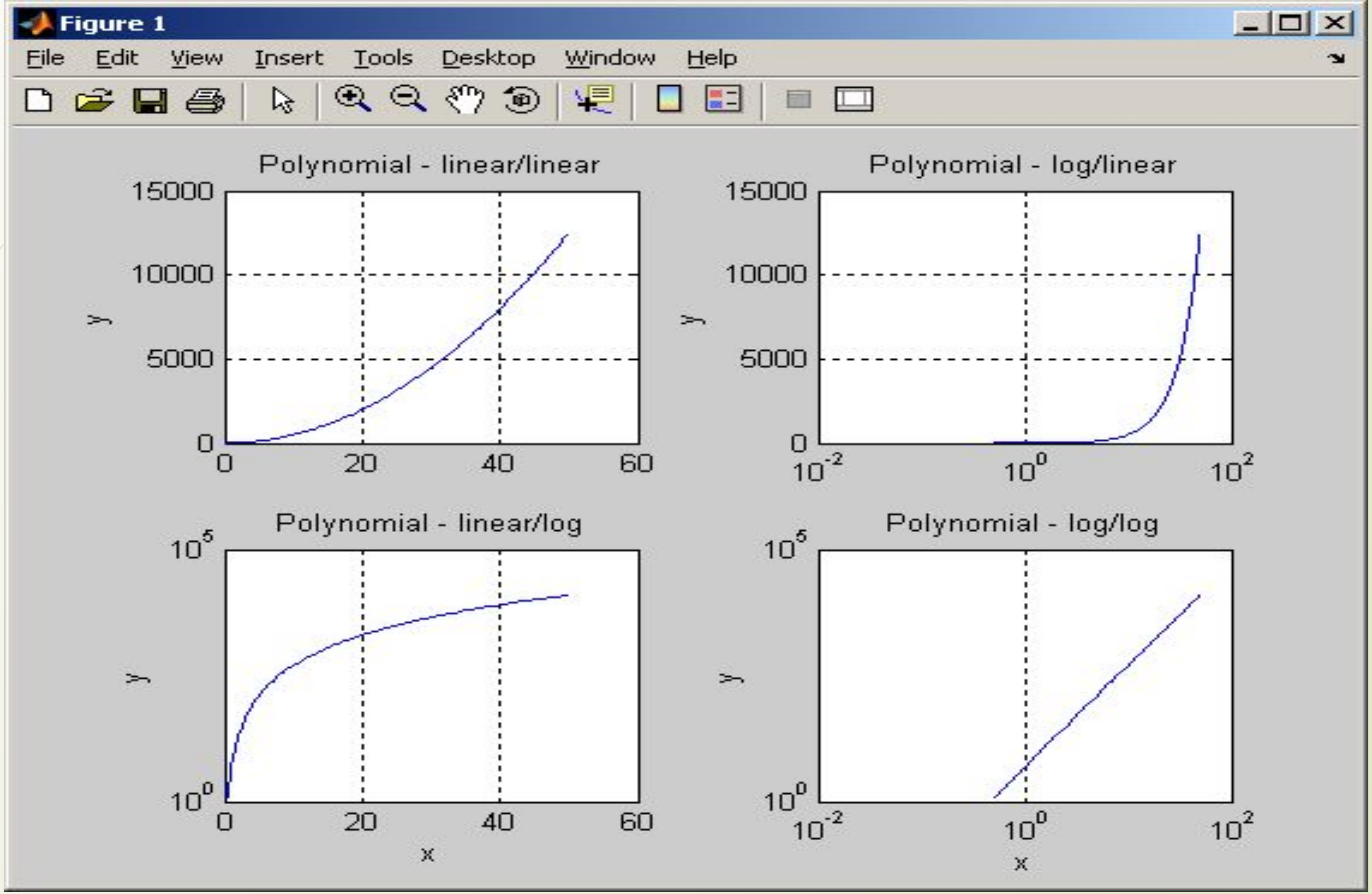
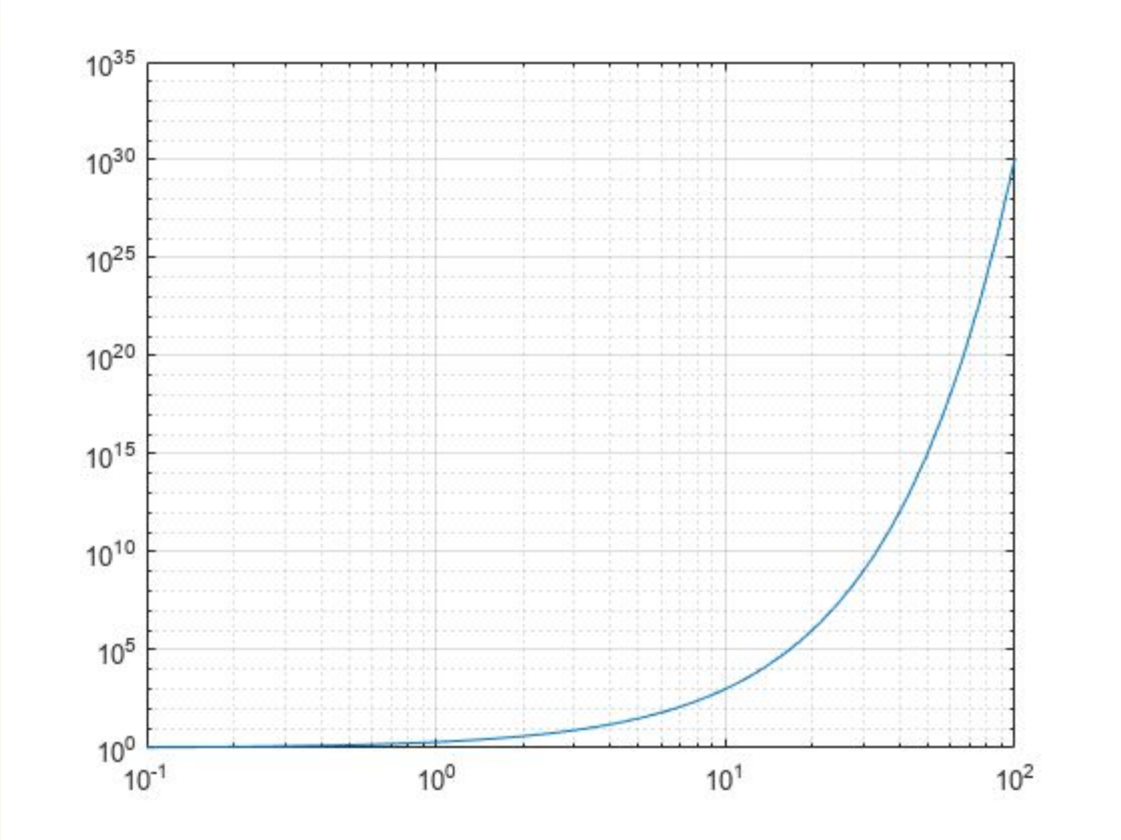


Figure 1
File Edit View Insert Tools Desktop Window Help
Polynomial - linear/linear
Polynomial - log/linear
Polynomial - linear/log
Polynomial - log/log
15000
10000
5000
0
0 20 40 60
10^-2 10^0 10^2
10^5
10^0
y
x

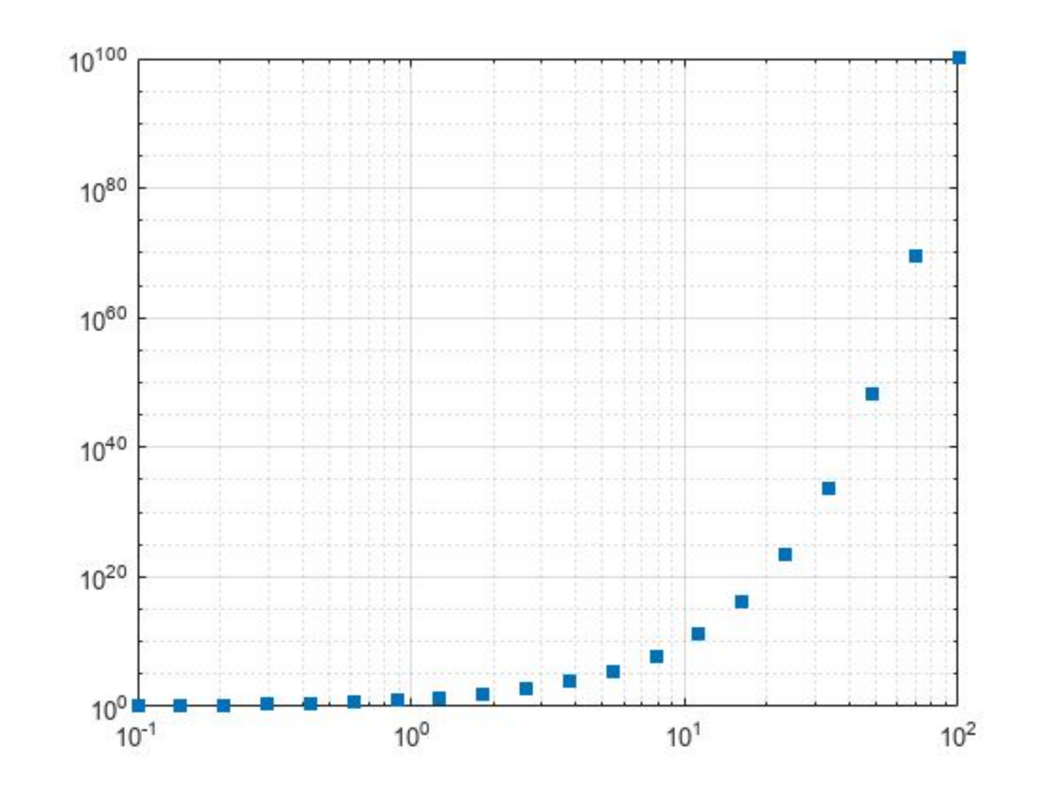
- loglog(X,Y) plots x- and y-coordinates using a base-10 logarithmic scale on the x-axis and the y-axis.
- To plot a set of coordinates connected by line segments, specify X and Y as vectors of the same length.
- To plot multiple sets of coordinates on the same set of axes, specify at least one of X or Y as a matrix.
- loglog(X,Y,LineSpec) creates the plot using the specified line style, marker, and color.
- loglog(X1,Y1,...,Xn,Yn) plots multiple pairs of x- and y-coordinates on the same set of axes. Use this syntax as an alternative to specifying coordinates as matrices.
- loglog(X1,Y1,LineSpec1,...,Xn,Yn,LineSpecn) assigns specific line styles, markers, and colors to each x-y pair. You can specify LineSpec for some x-y pairs and omit it for others. For example, loglog(X1,Y1,'o',X2,Y2) specifies markers for the first x-y pair but not for the second pair.
- loglog(Y) plots Y against an implicit set of x-coordinates.

- If Y is a vector, the x-coordinates range from 1 to length(Y).
- If Y is a matrix, the plot contains one line for each column in Y. The x-coordinates range from 1 to the number of rows in Y.
- If Y contains complex numbers, loglog plots the imaginary part of Y versus the real part of Y. However, if you specify both X and Y, MATLAB® ignores the imaginary part.
- loglog(Y,LineSpec) plots Y using implicit x-coordinates, and specifies the line style, marker, and color.
- loglog(tbl,xvar,yvar) plots the variables xvar and yvar from the table tbl. To plot one data set, specify one variable for xvar and one variable for yvar. To plot multiple data sets, specify multiple variables for xvar, yvar, or both. If both arguments specify multiple variables, they must specify the same number of variables.

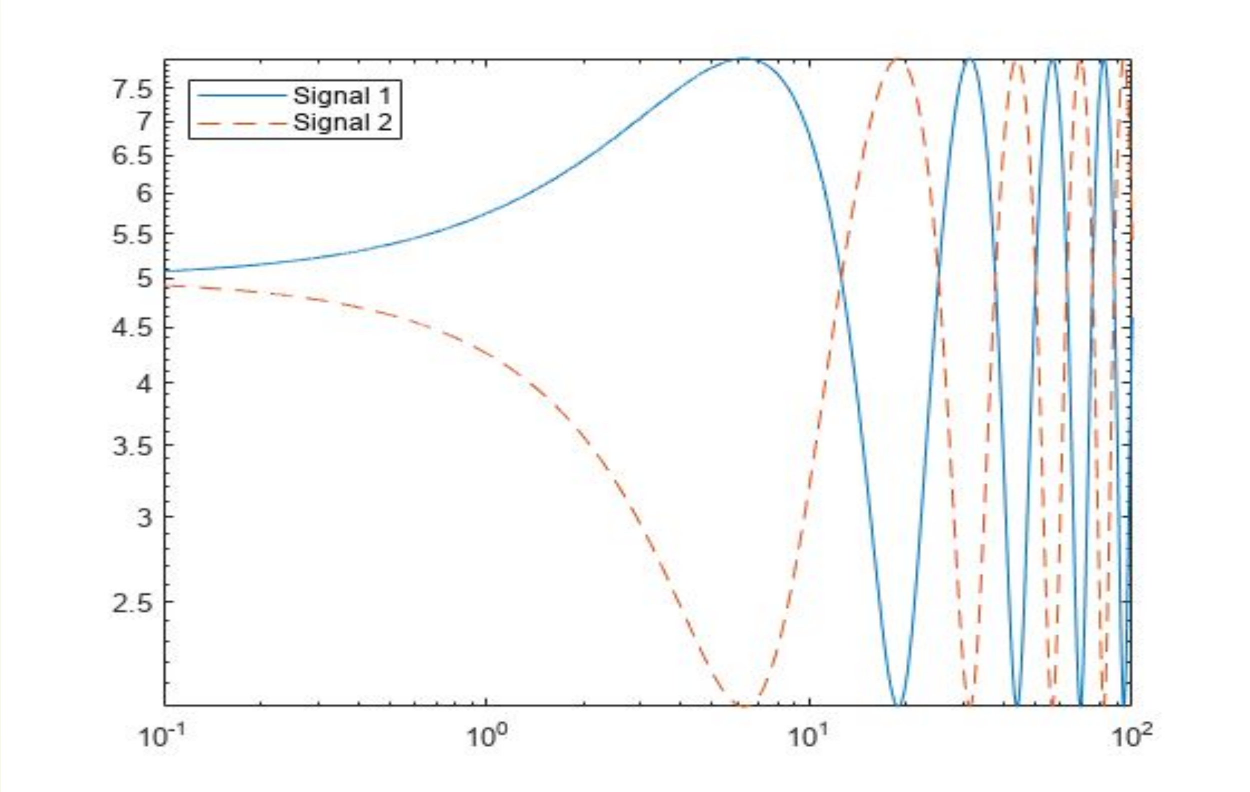
```
x = logspace(-1,2);
y = 2.^x;
loglog(x,y) grid on
```



```
x = logspace(-1,2,20);
 y = 10.^x;
loglog(x,y,'s','MarkerFaceColor',[0 0.447 0.741])
 grid on
```



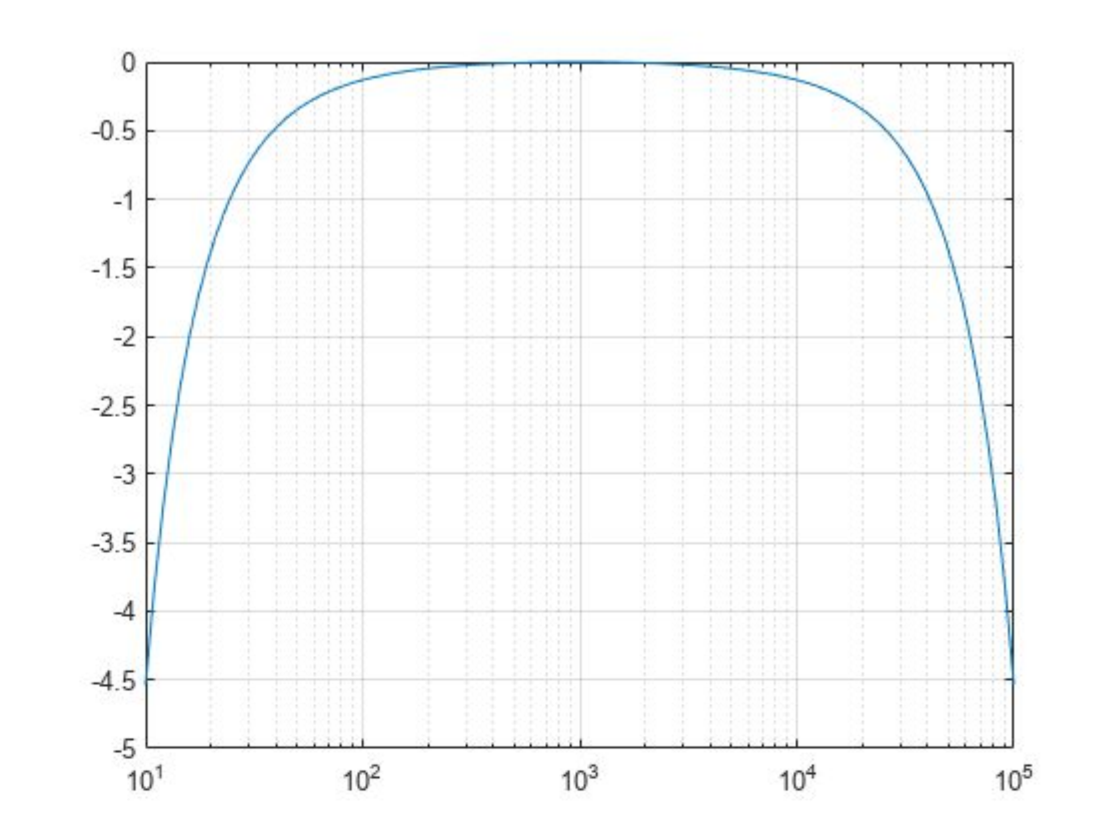
```
x = logspace(-1,2,10000);
 y1 = 5 + 3*sin(x/4);
y2 = 5 - 3*sin(x/4);
 loglog(x,y1,x,y2,'--')
legend('Signal 1','Signal 2','Location','northwest')
```



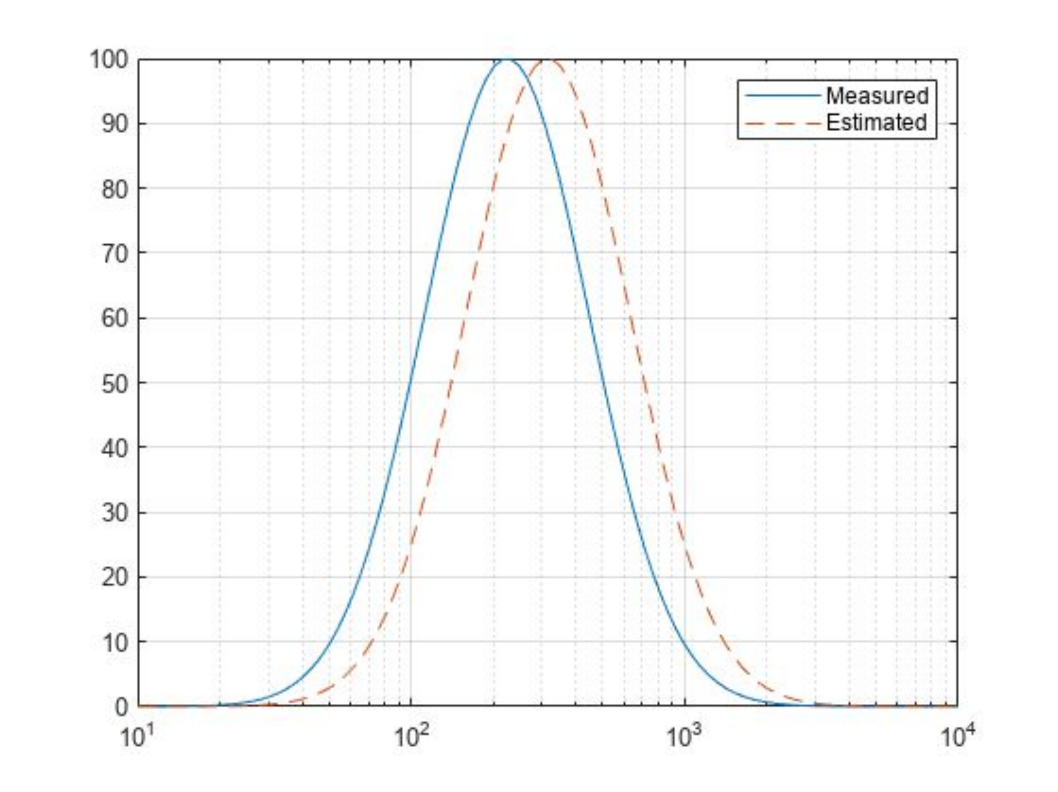
```
x = (0.1:0.1:50);
y = 0.1+0.1*exp(0.1*x);
xmax= 30;
dx = 5;
loglog(x,y);
set(gca,'XLim',[1e-1 xmax]);
set(gca,'YLim',[1e-1 1e1]);
set(gca,'XTick',[1e-1 1 (dx:dx:xmax)]);
set(gca,'YTick',10.^(-1:1));
set(gca,'XMinorTick','on','YMinorTick','on')
```

Semilogx()

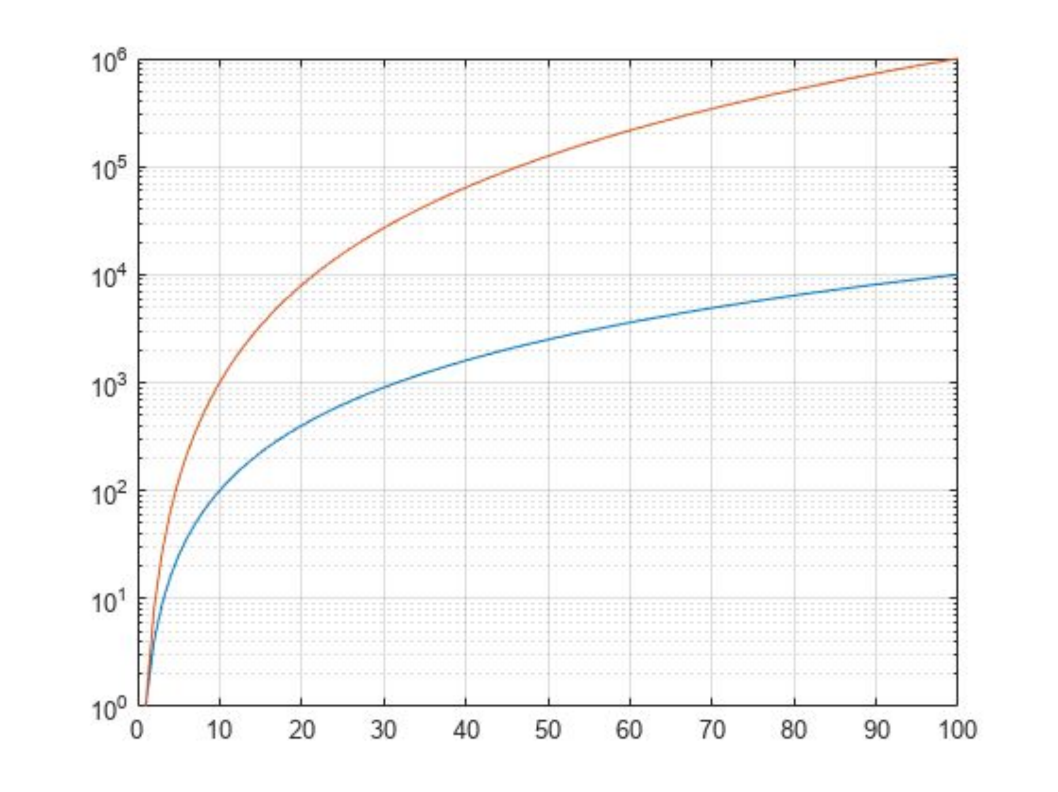
```
f = logspace(1,5,100);
v = linspace(-50,50,100);
gain = (1-exp(5*(2.5*v.^2)./7500))/14;
semilogx(f,gain) grid on
```



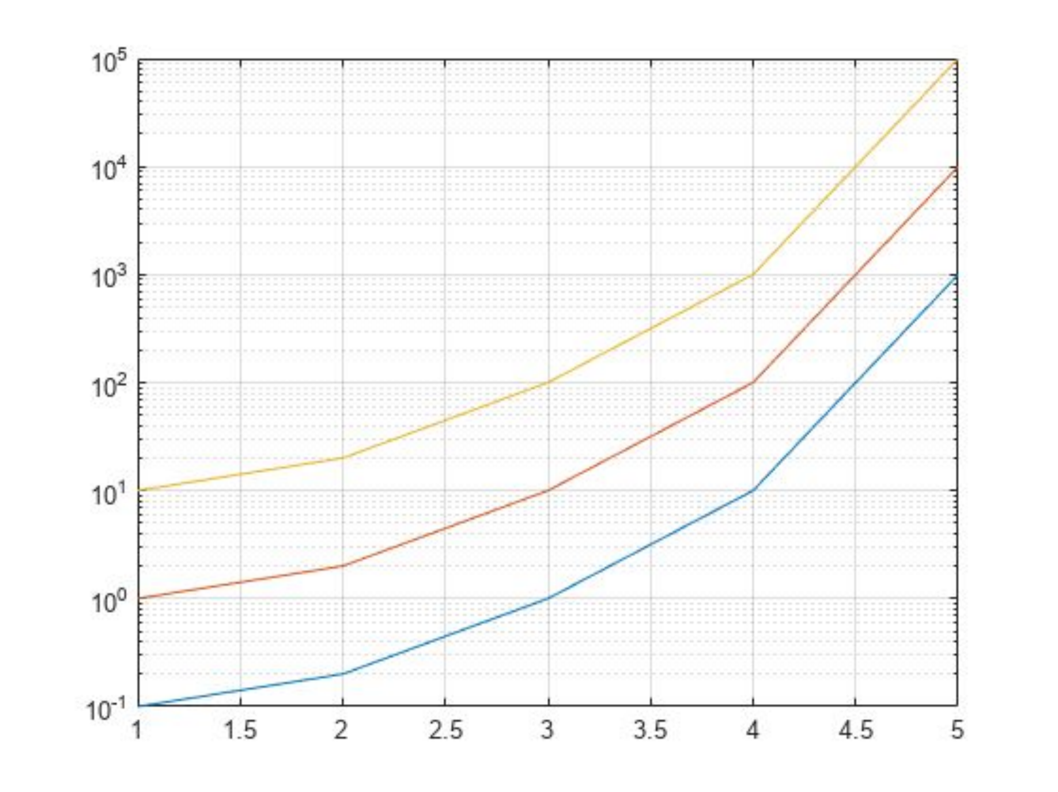
```
x = logspace(1,4,100);
v = linspace(-50,50,100);
y1 = 100*exp(-1*((v+5).^2)./200);
y2 = 100*exp(-1*(v.^2)./200);
semilogx(x,y1,x,y2,'--')
legend('Measured','Estimated')
grid on
```



```
x = 1:100;
 y1 = x.^2;
y2 = x.^3;
semilogy(x,y1,x,y2)
 grid on
```



```
y = [ 0.1 1 10 0.2 2 20 1.0 10 100 10 100 1000 1000 10000 100000];
semilogy(y)
 grid on
```



Subplots

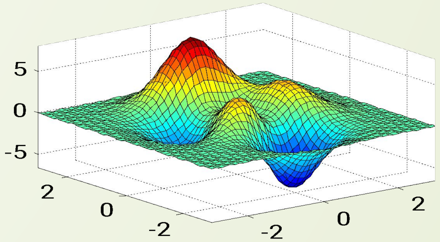
- The **subplot** command allows you to subdivide the graphing window into a grid of m rows and n columns
- **`subplot(m,n,p)`**

rows columns location

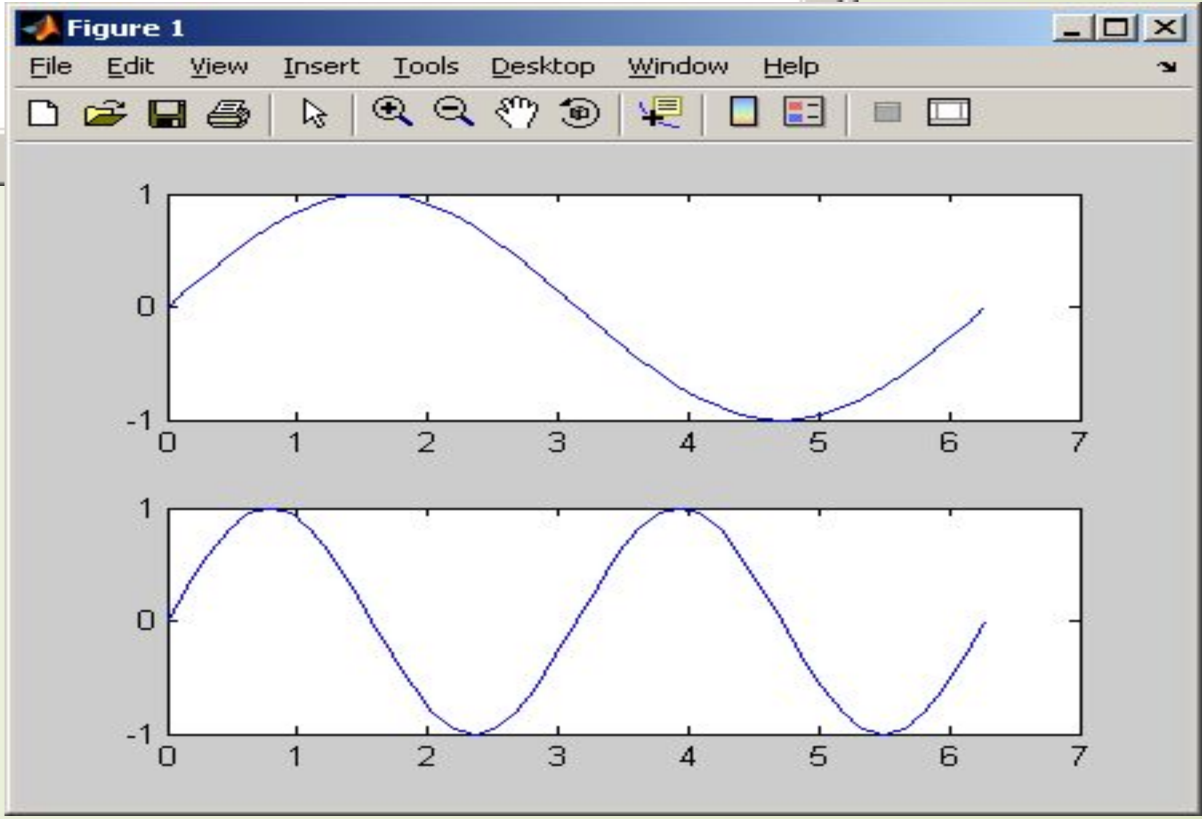
subplot(2,2,1)

2 columns

2 rows

	2
3	4

```
>> x=0:pi/20:2*pi;
>> subplot(2,1,1)
>> plot(x,sin(x))
>> subplot(2,1,2)
>> plot(x,sin(2*x))
>>
```

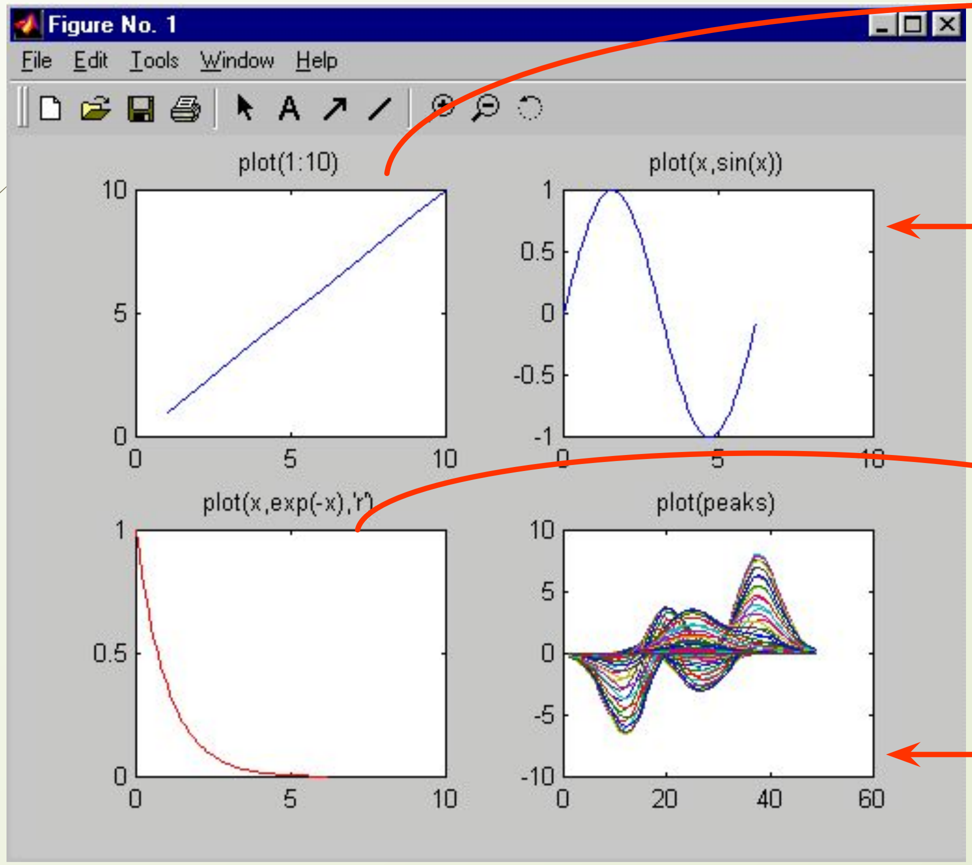


2 rows and 1 column

Subplots

SUBPLOT- display multiple axes in the same figure window

```
subplot(#rows, #cols, index)
```



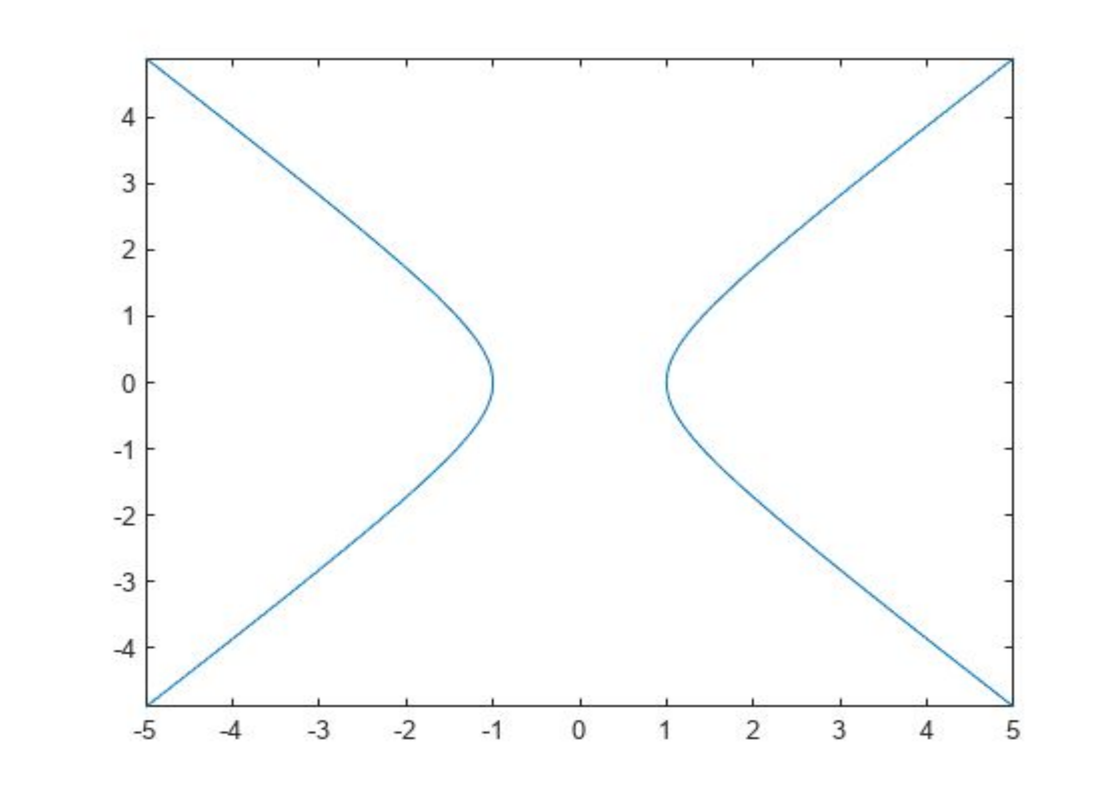

```
»subplot(2,2,1);
»plot(1:10)

»subplot(2,2,2)
»x = 0:.1:2*pi;
»plot(x,sin(x))

»subplot(2,2,3)
»x = 0:.1:2*pi;
»plot(x,exp(-x),'r')

»subplot(2,2,4)
»plot(peaks)
```

fimplicit()

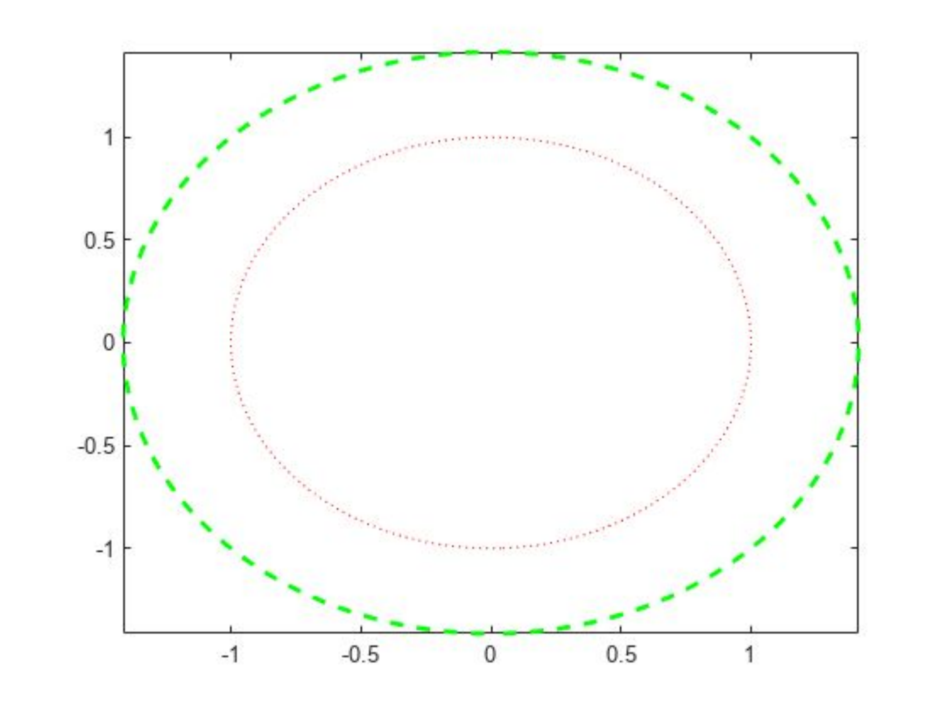
- fimplicit(f) plots the implicit function defined by f(x,y) = 0 over the default interval [-5 5] for x and y.
- fimplicit(f,interval) specifies the plotting interval for x and y.
- fimplicit(ax,___) plots into the axes specified by ax instead of into the current axes. Specify the axes as the first input argument, prior to any of the previous input arguments.
- fimplicit(___,LineSpec) specifies the line style, marker symbol, and line color. For example, '-r' plots a red line.
- fimplicit(___,Name,Value) specifies line properties using one or more name-value pair arguments. For example, 'LineWidth',2 specifies a line width of 2 points.
- fp = fimplicit(___) returns the ImplicitFunctionLine object. Use fp to access and modify properties of the line after it is created.

Questions:

- fimplicit(@(x,y) x.^2 - y.^2 - 1)



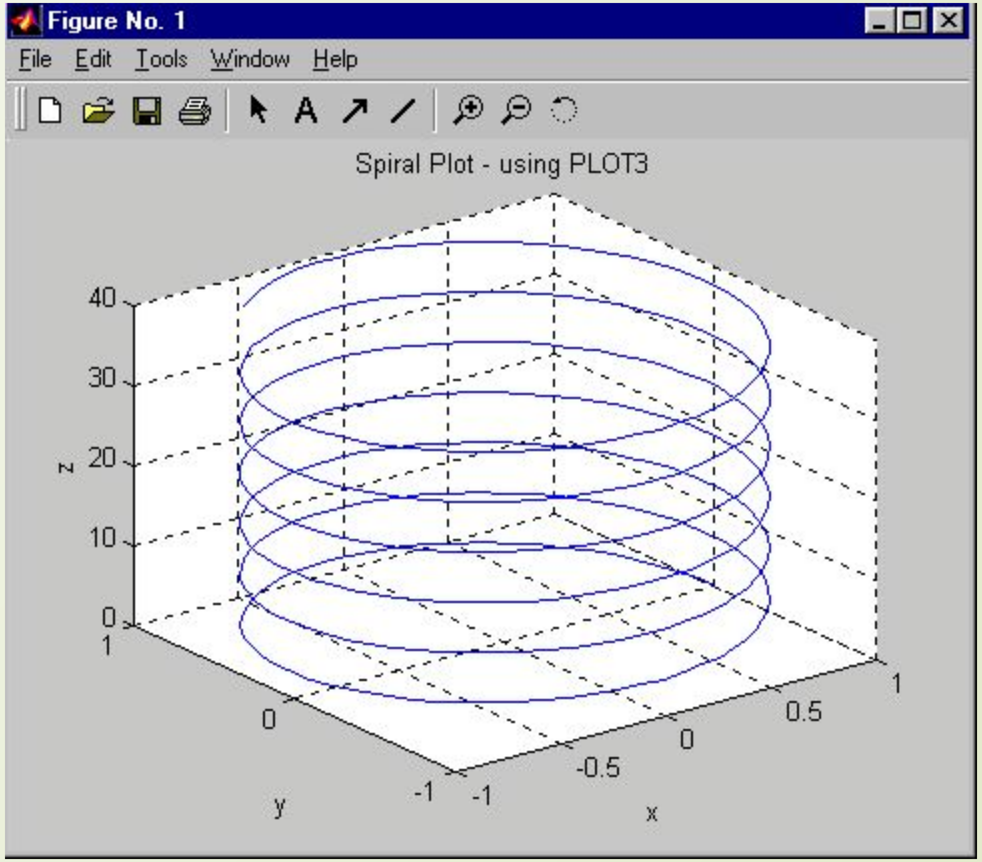
```
f1 = @(x,y) x.^2 + y.^2 - 1;
 fimplicit(f1,':r')
 hold on
 f2 = @(x,y) x.^2 + y.^2 - 2;
 fimplicit(f2,'--g','LineWidth',2)
hold off
```



3-D Line Plotting

```
plot3(xdata, ydata, zdata, 'clm', ...)
```

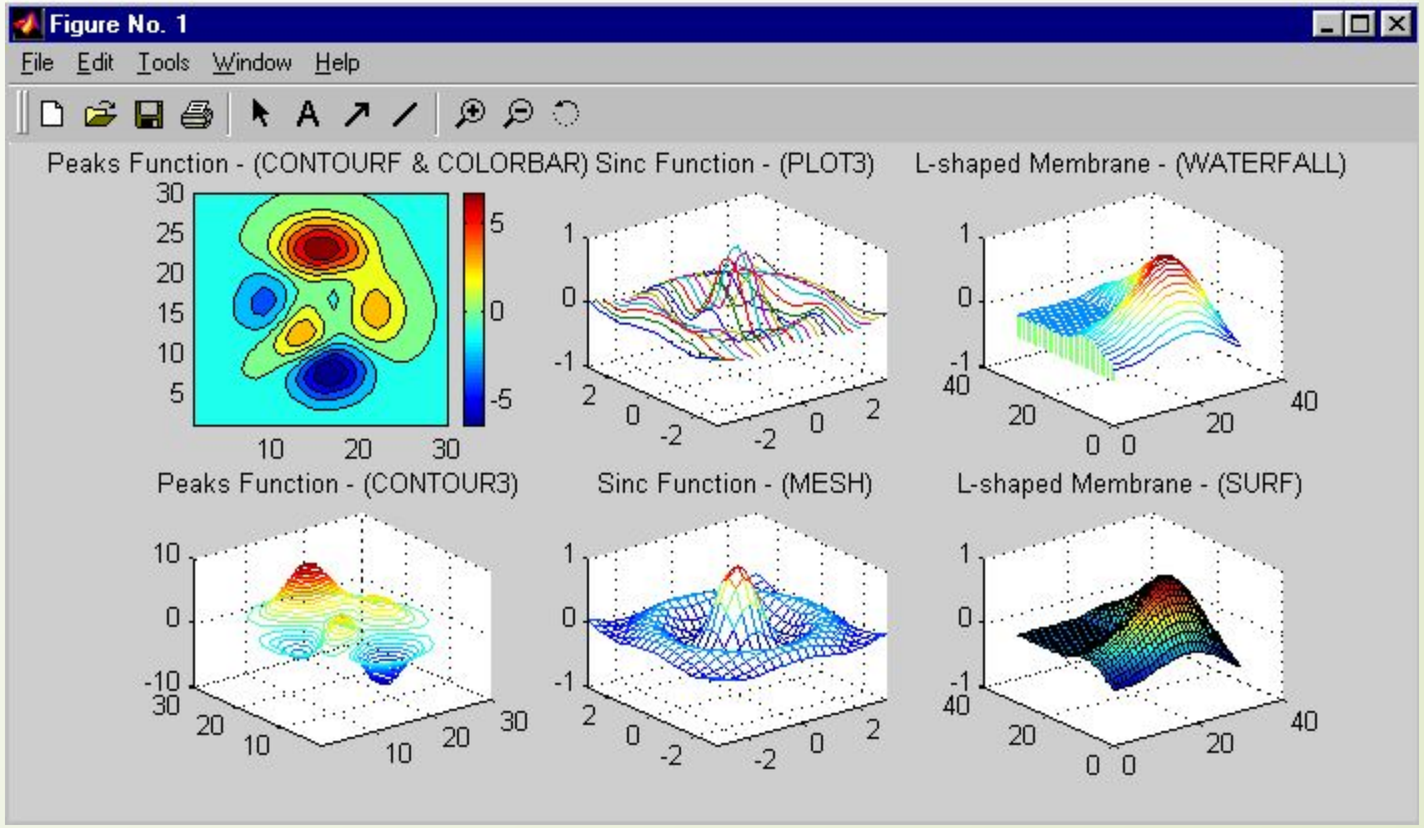
```
» z = 0:0.1:40;
» x = cos(z);
» y = sin(z);
» plot3(x,y,z)
```



»plot_3d

Ref: Color, Linestyle, Marker options

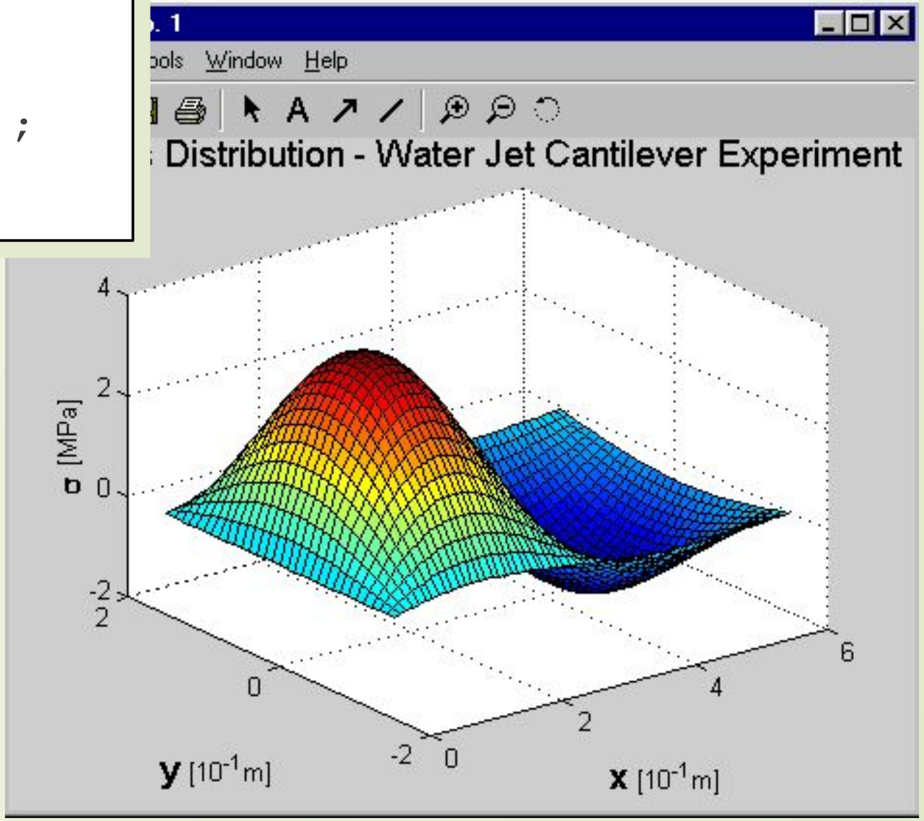
3-D Surface Plotting



»surf_3d

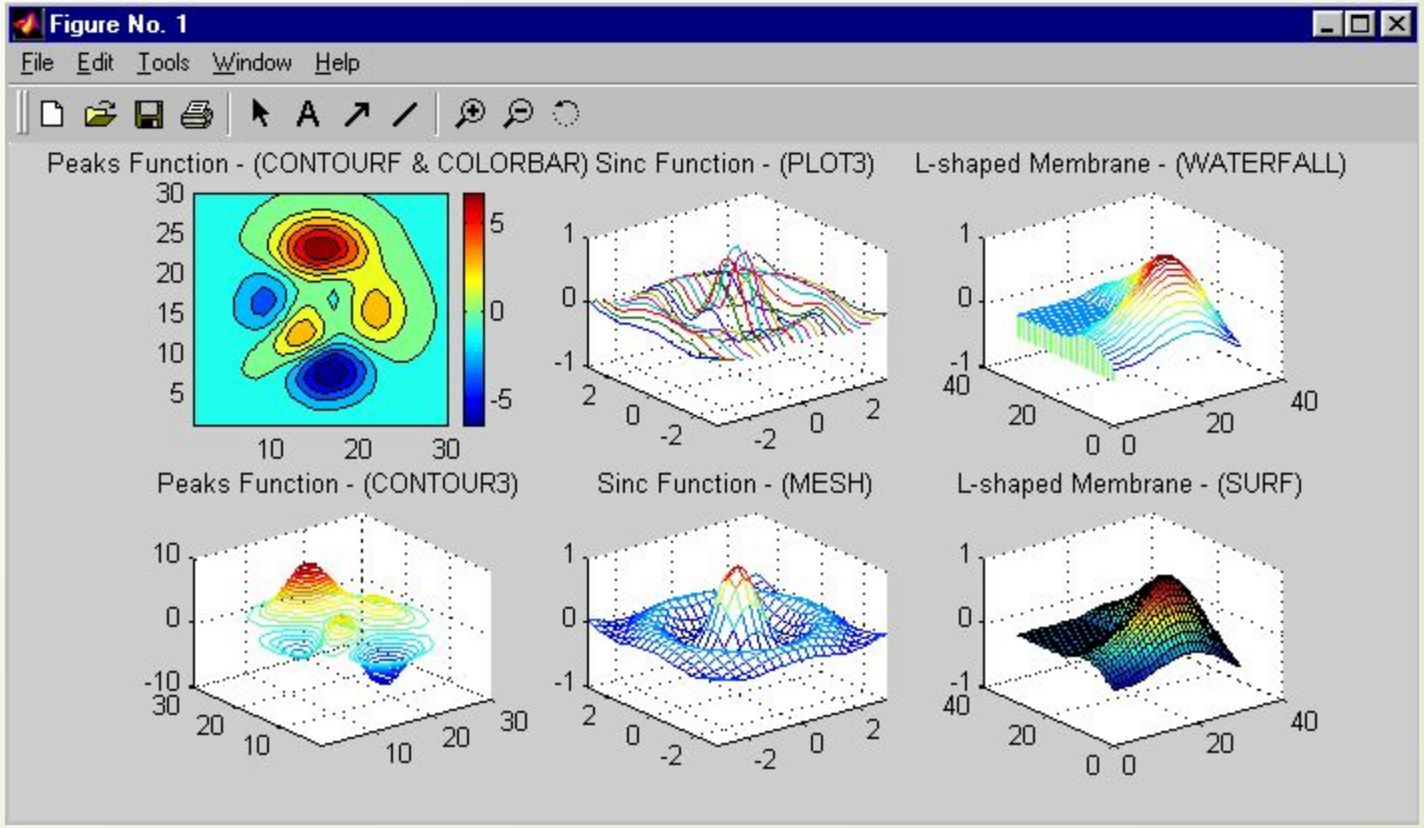
Solution: 3-D Plotting

```
» B = -0.2;
» x = 0:0.1:2*pi;
» y = -pi/2:0.1:pi/2;
» [x,y] = meshgrid(x,y);
» z = exp(B*x).*sin(x).*cos(y);
» surf(x,y,z)
```



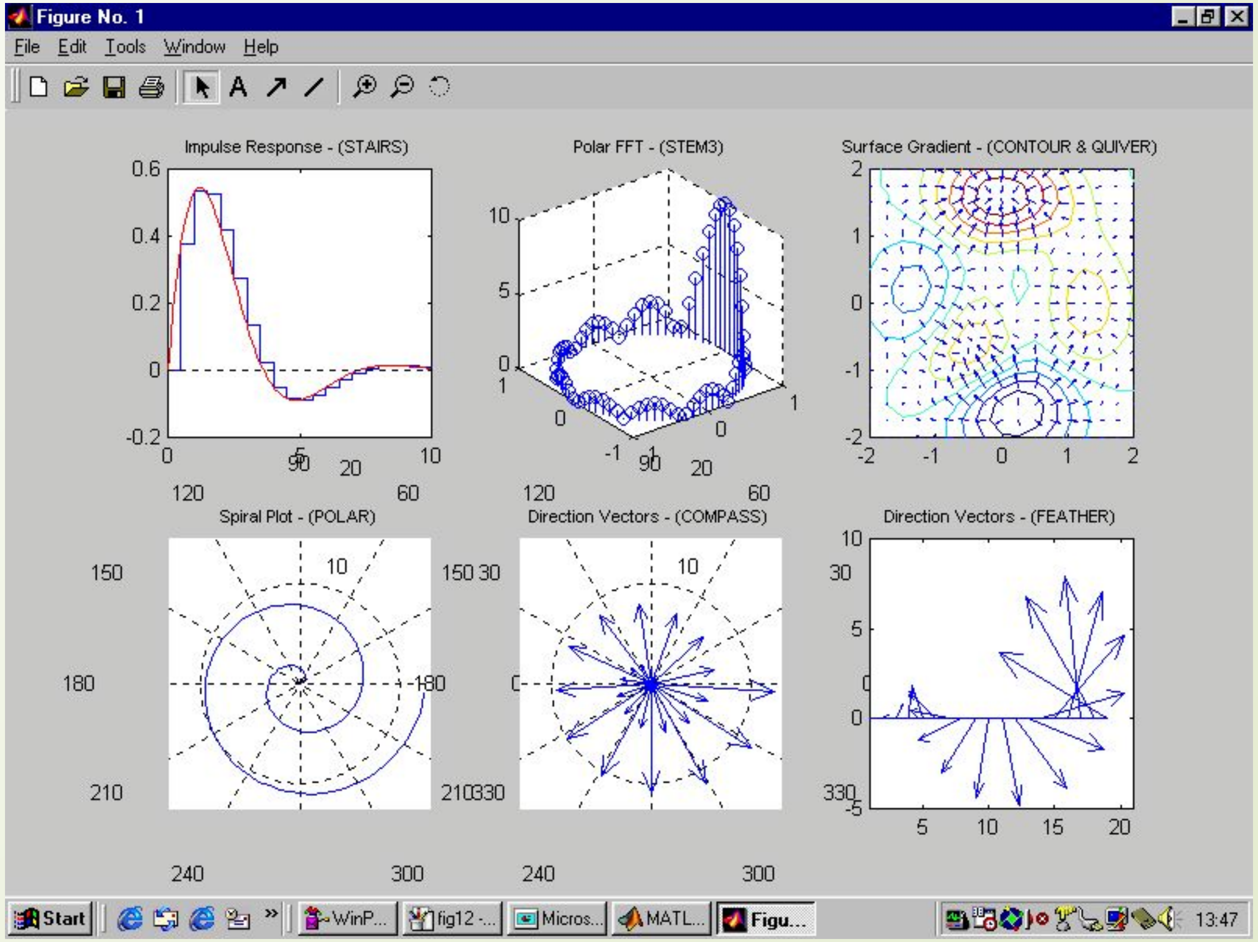
»plot3d_soln

Specialized Plotting Routines



»spec_plots

Specialized Plotting Routines (2)



»spec_plots2

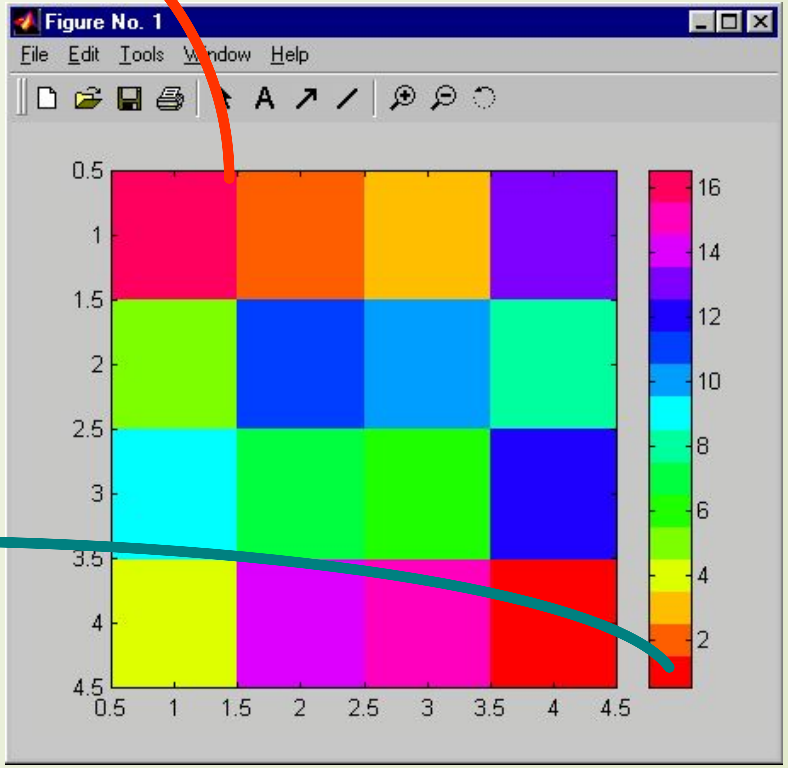
Images

Reduced Memory Requirements:
Images represented as UINT8 - 1 byte

```
» a = magic(4)
a =
  16    2    3   13
   5   11   10    8
   9    7    6   12
   4   14   15    1
» image(a);
» map = hsv(16)
map =
  1.0000         0      0
  1.0000    0.3750      0
  1.0000    0.7500      0 .....
» colormap(map)
```

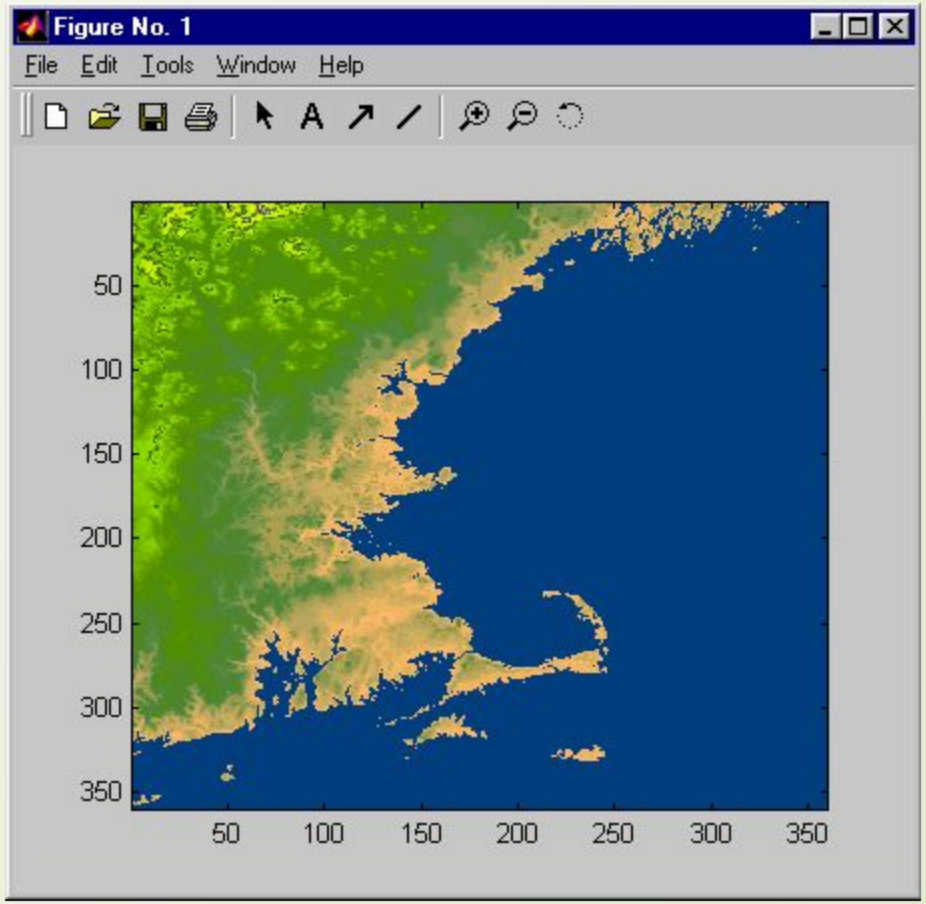
Use Row 2 of colormap for pixel (1,2)

Row 2



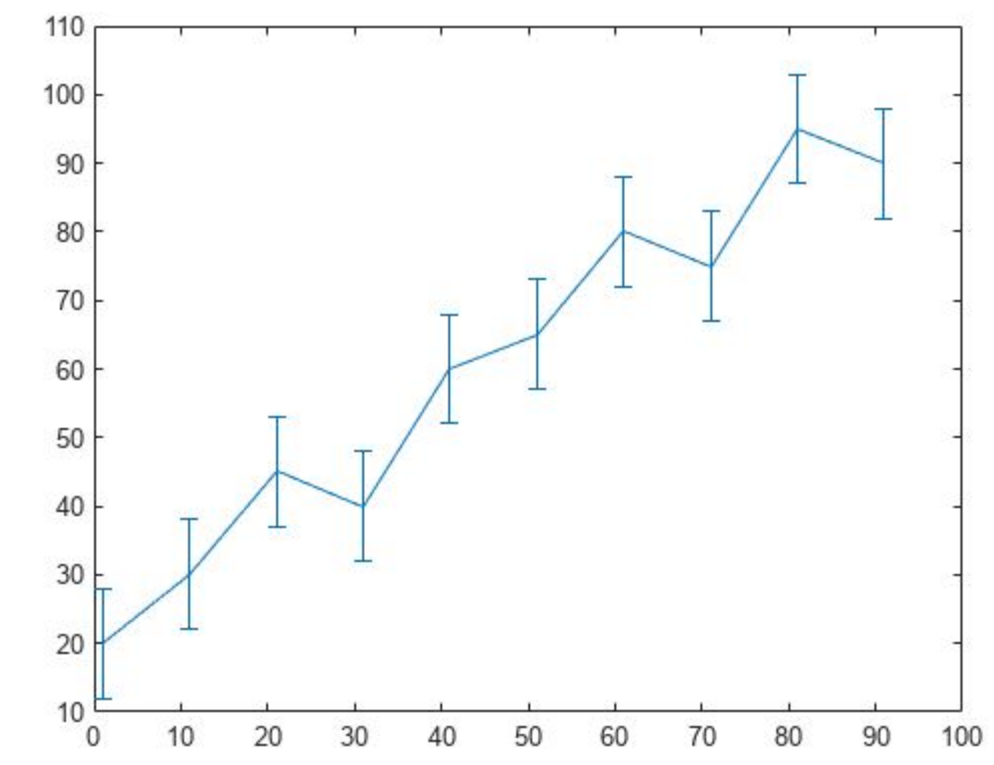
Example: Images

```
» load cape
» image(X)
» colormap(map)
```

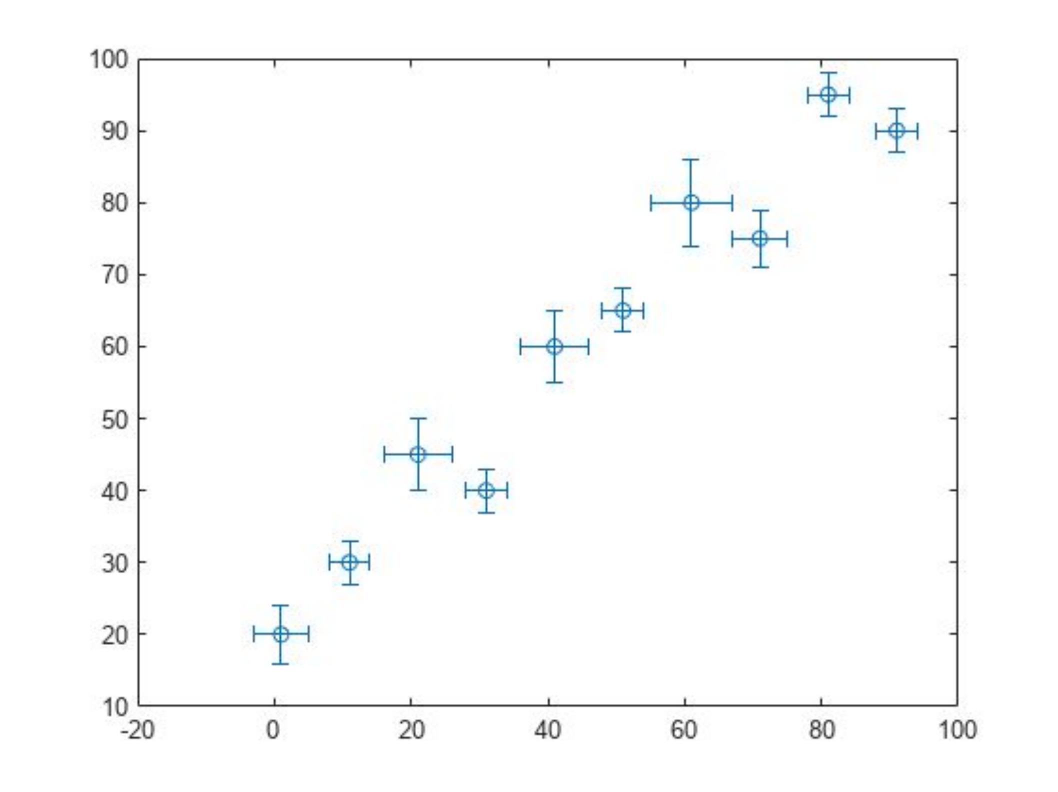


errorbar

```
x = 1:10:100;
y = [20 30 45 40 60 65 80 75 95 90];
 err = 8*ones(size(y));
errorbar(x,y,err)
```



```
x = 1:10:100;
 y = [20 30 45 40 60 65 80 75 95 90];
err = [4 3 5 3 5 3 6 4 3 3];
errorbar(x,y,err,"both","o")
```



Bar Graphs and Pie Charts

- MATLAB includes a whole family of bar graphs and pie charts
  - `bar(x)` – vertical bar graph
  - `barh(x)` – horizontal bar graph
  - `bar3(x)` – 3-D vertical bar graph
  - `bar3h(x)` – 3-D horizontal bar graph
  - `pie(x)` – pie chart
  - `pie3(x)` – 3-D pie chart

Editor - Untitled2*

```
clear, clc
x=[1,2,5,4,8];
y=[x;1:5];
subplot(2,2,1)
     bar(x),title('A bargraph of vector x')
subplot(2,2,2)
     bar(y),title('A bargraph of matrix y')
subplot(2,2,3)
     bar3(y),title('A three dimensional bargraph')
subplot(2,2,4)
     pie(x),title('A pie chart of x')
```

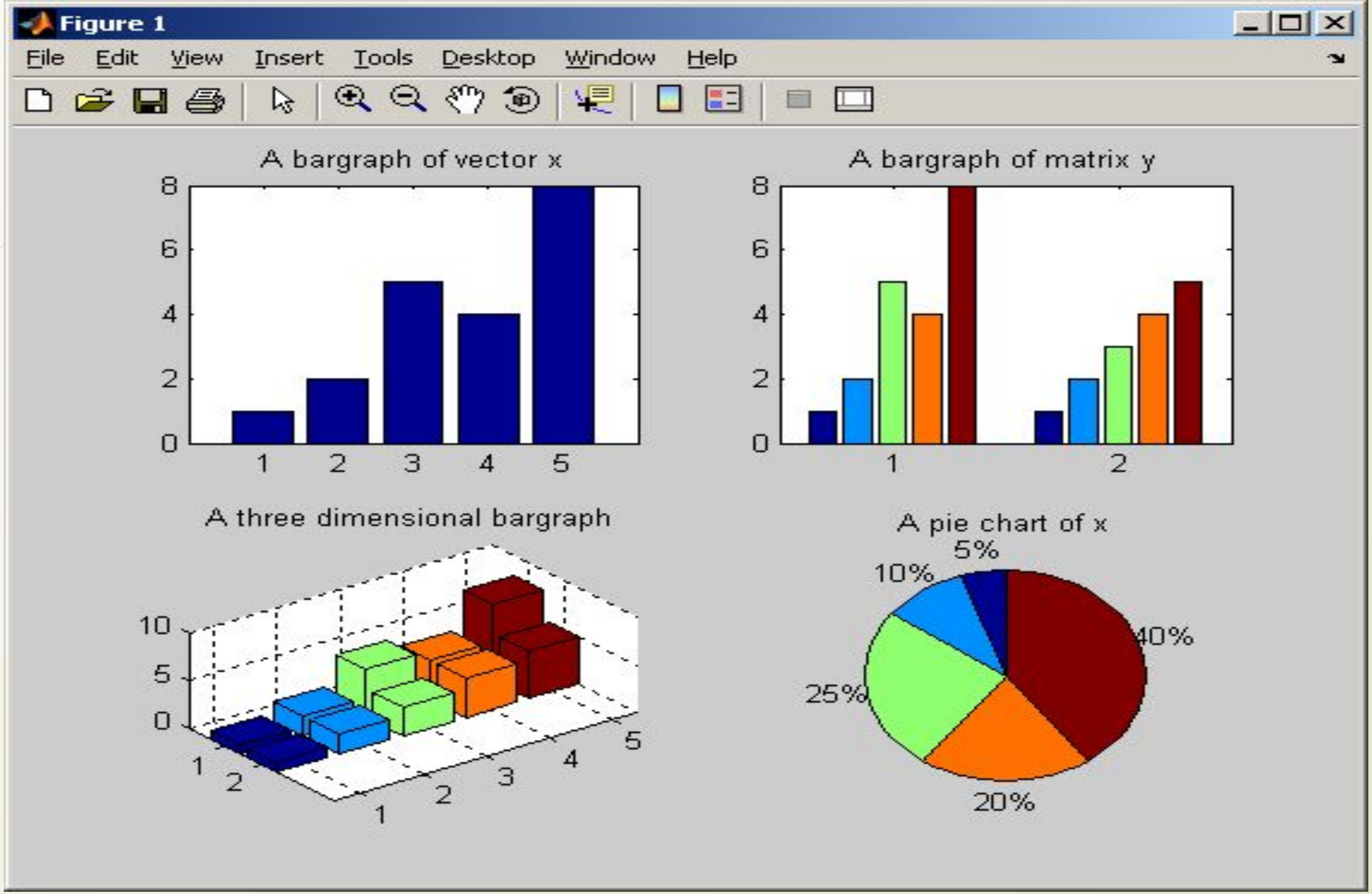
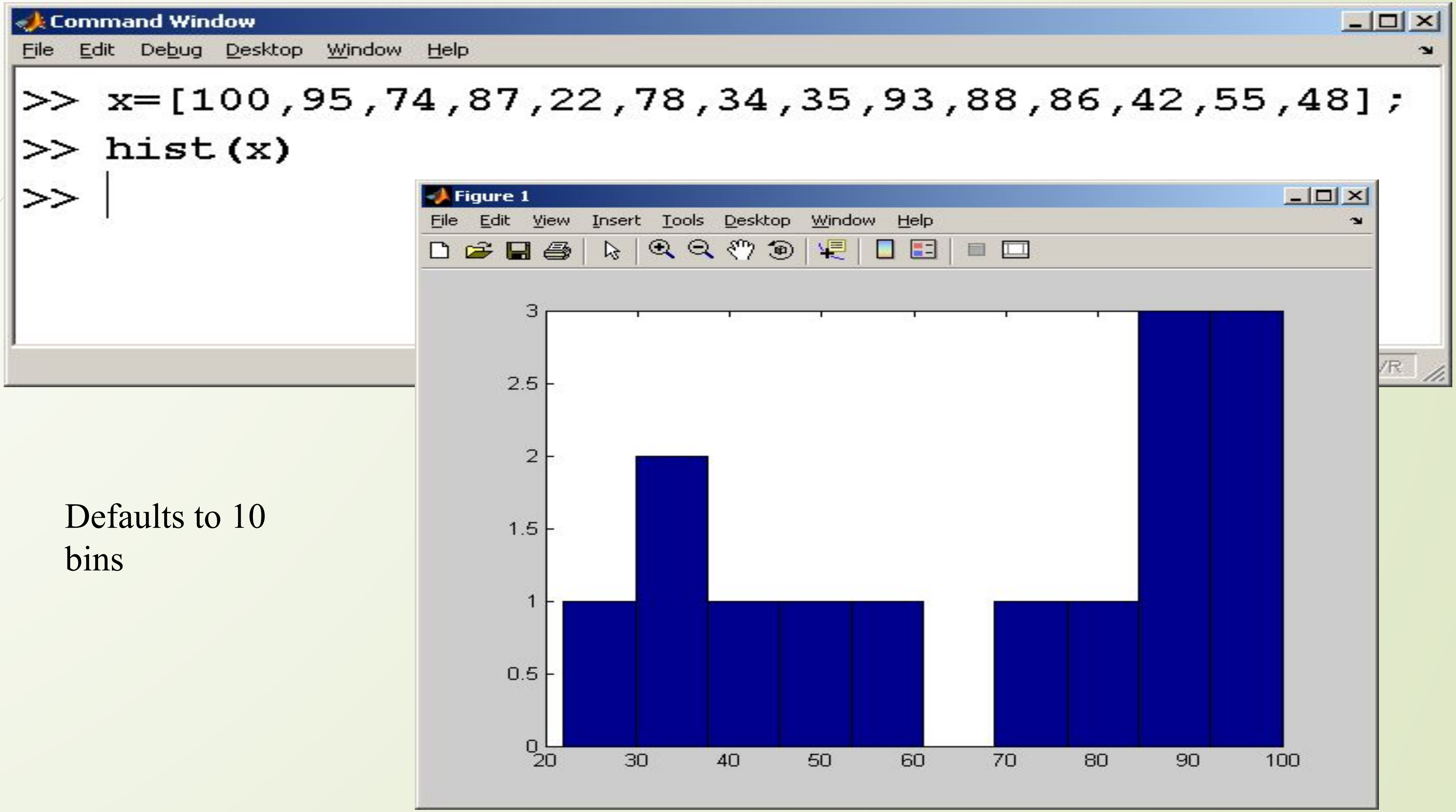


Figure 1
File Edit View Insert Tools Desktop Window Help
A bargraph of vector x
A bargraph of matrix y
A three dimensional bargraph
A pie chart of x
5%
10%
40%
25%
20%

Histograms

- A histogram is a plot showing the distribution of a set of values

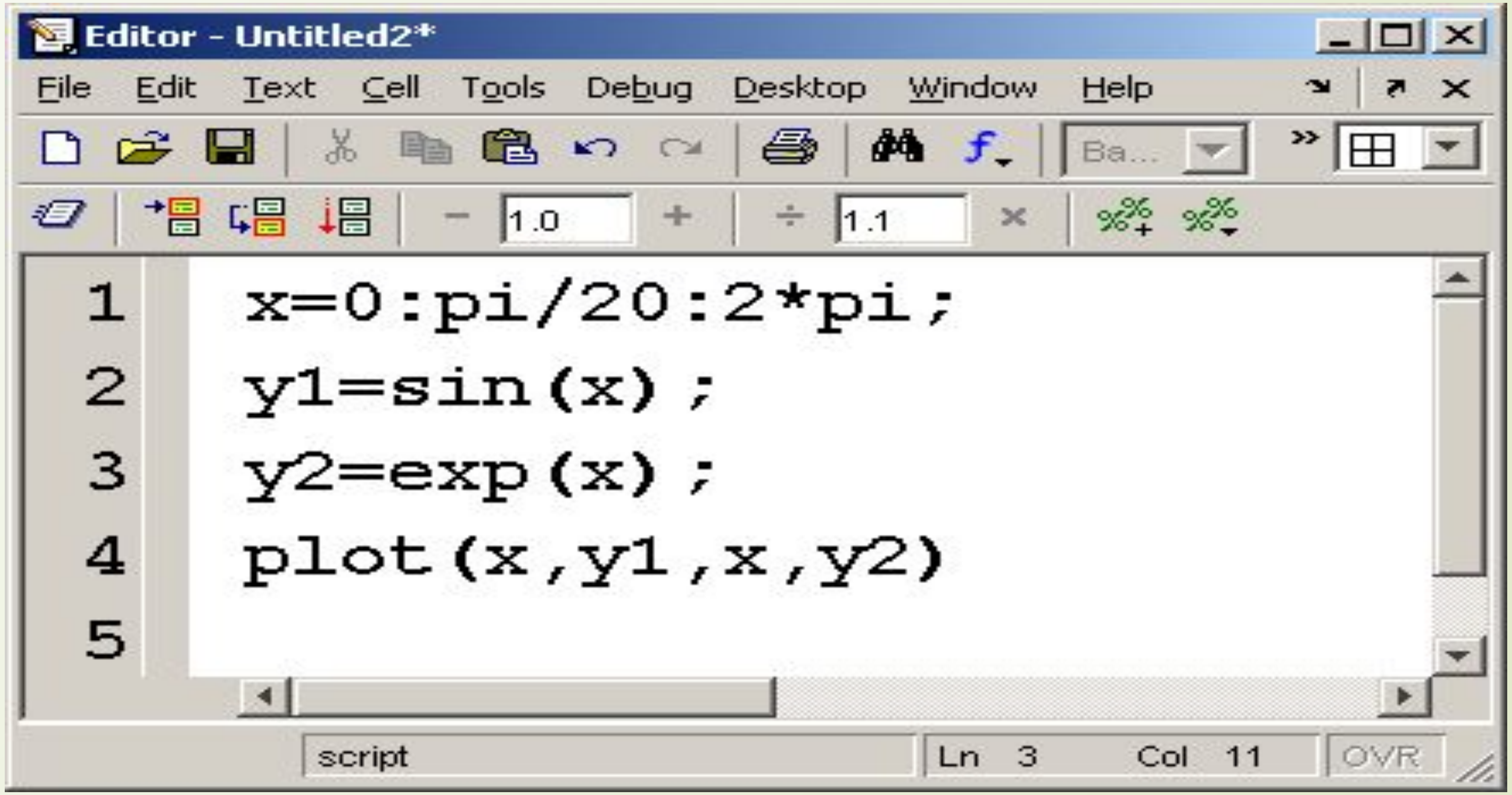


Defaults to 10 bins

X-Y Graphs with Two Y Axes

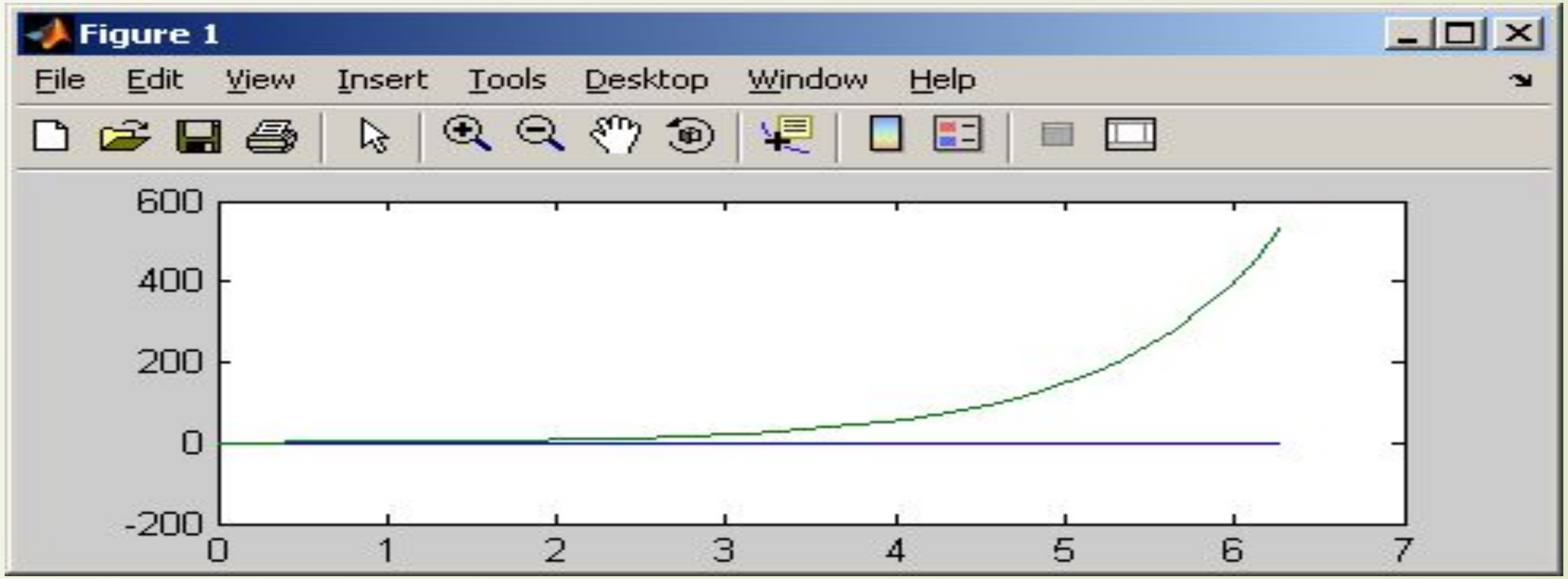
- Sometimes it is useful to overlay two *x-y* plots onto the same figure. However, if the order of magnitude of the *y* values are quite different, it may be difficult to see how the data behave.

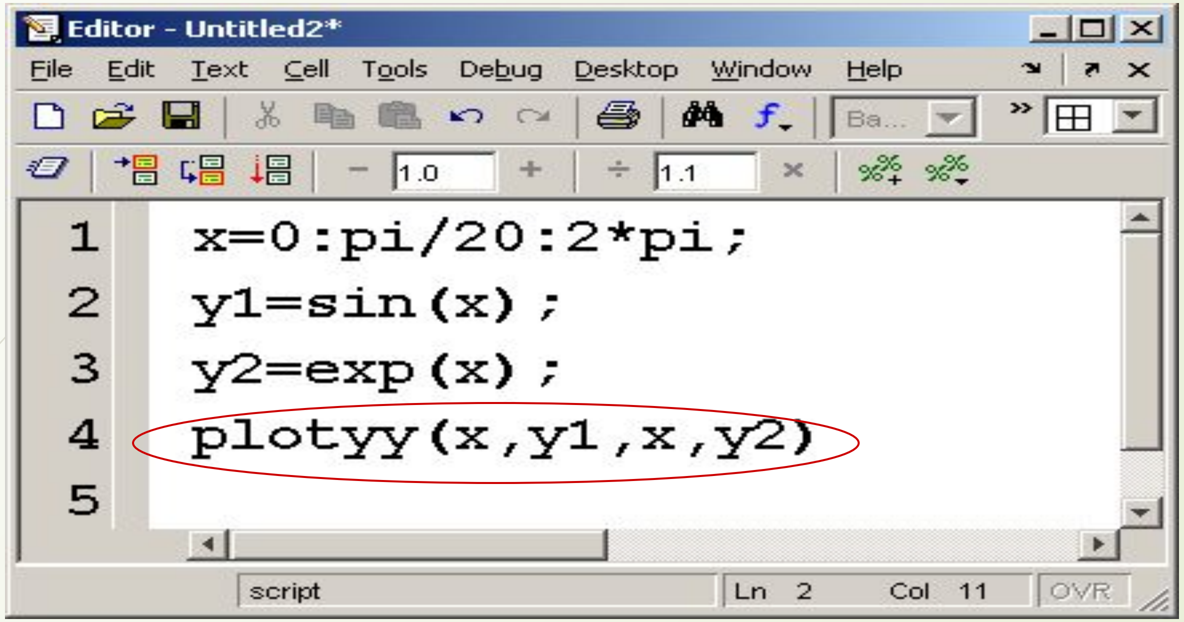
For example

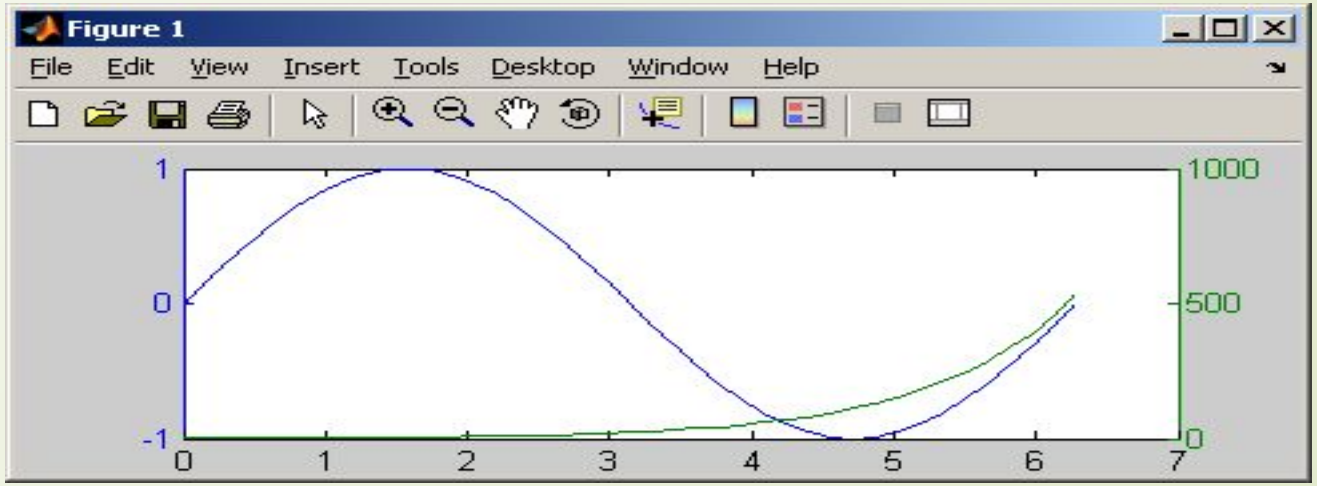

Editor - Untitled2*
File Edit Text Cell Tools Debug Desktop Window Help
Ba...
1.0
1.1
1 x=0:pi/20:2*pi;
2 y1=sin(x);
3 y2=exp(x);
4 plot(x,y1,x,y2)
5
script
Ln 3 Col 11
OVR

Scaling Depends on the largest value plotted

- Its difficult to see how the blue line behaves, because the scale isn't appropriate







The plotyy function allows you to use two scales on a single graph

Function Plots

- Function plots allow you to use a function as input to a plot command, instead of a set of ordered pairs of x-y values
- **fplot('sin(x)',[-2*pi,2*pi])**

function input as a string

range of the independent variable – in this case x

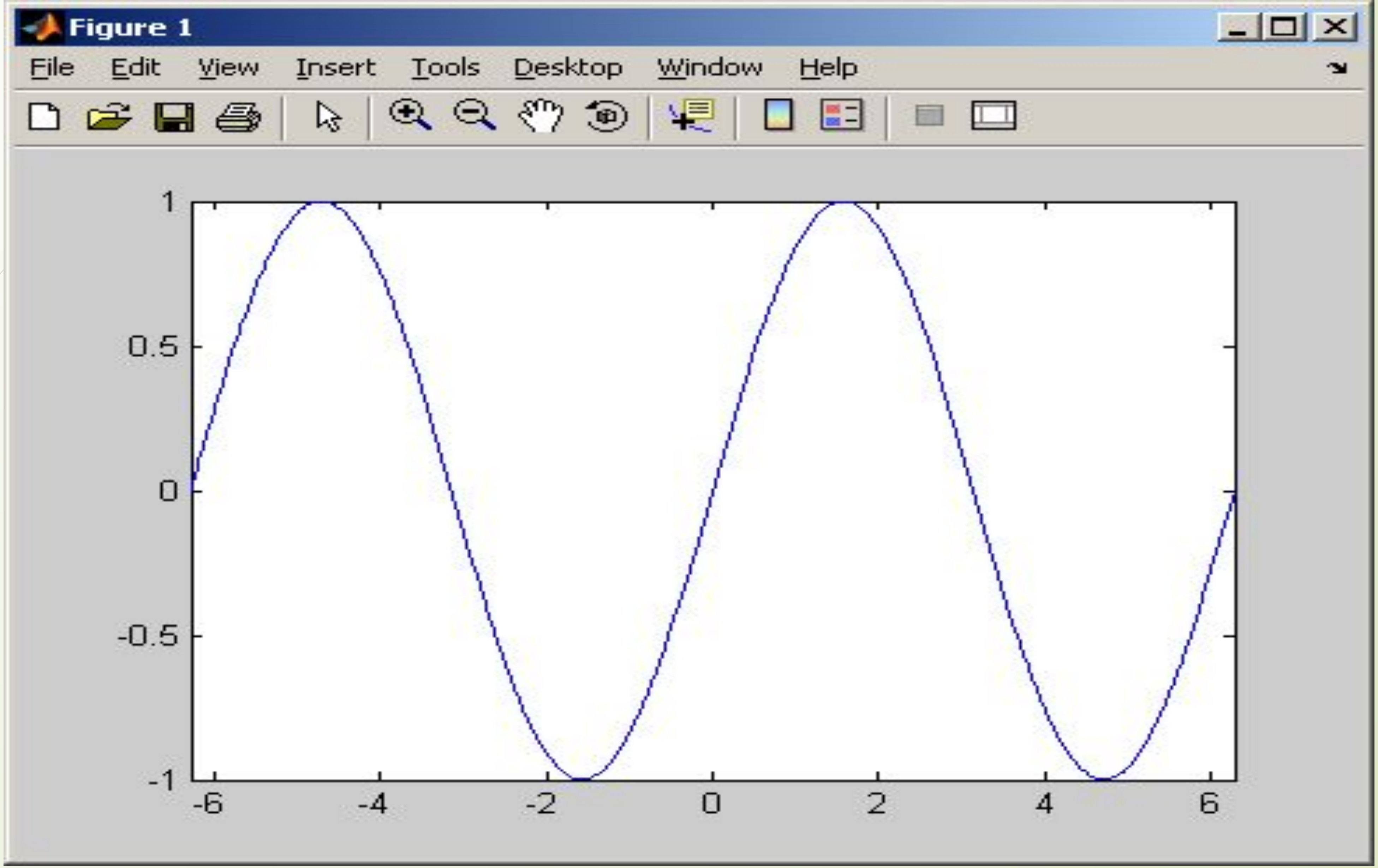


Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.5
0
-0.5
-1
-6 -4 -2 0 2 4 6

Three Dimensional Plotting

- Line plots
- Surface plots
- Contour plots

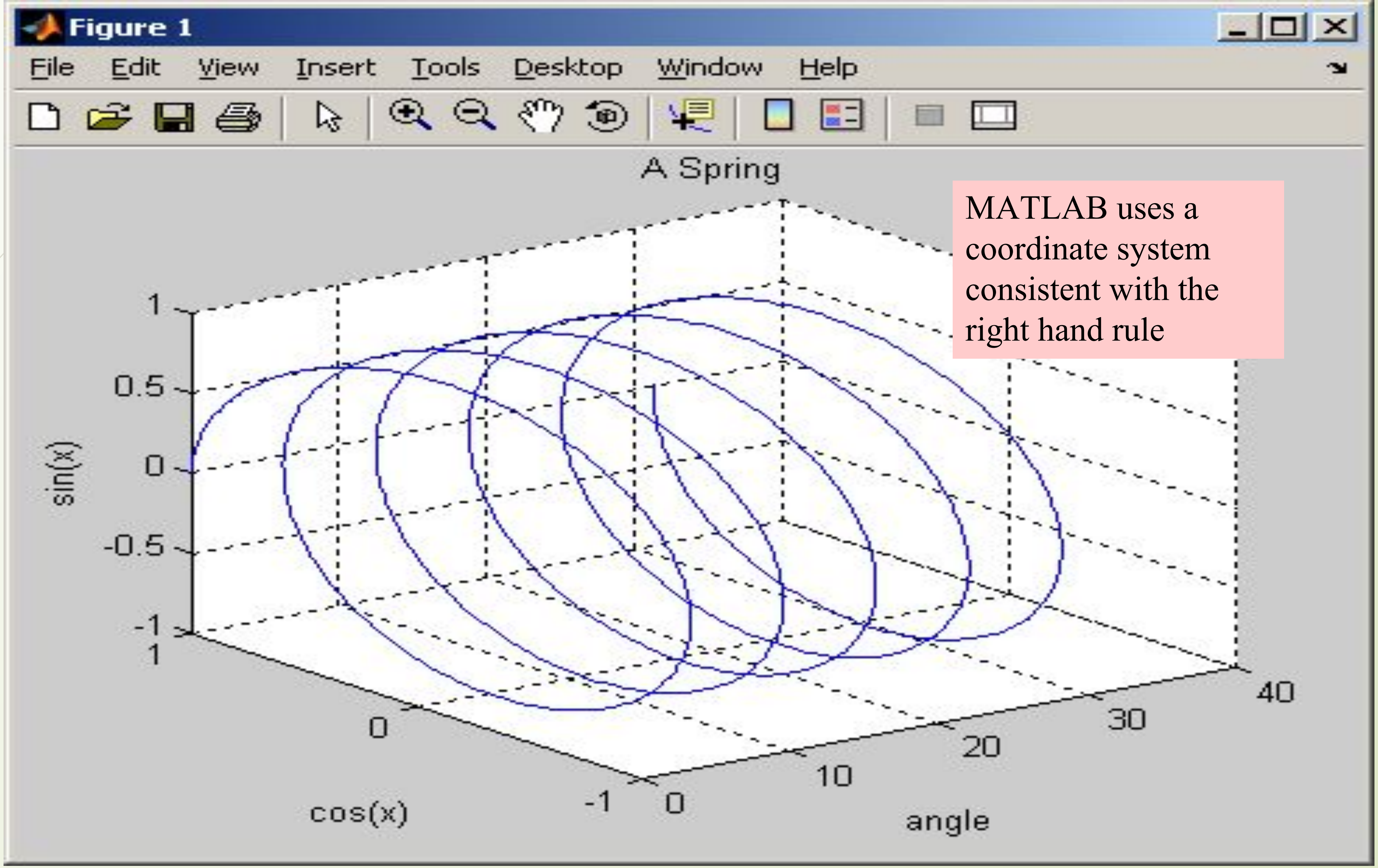
Three Dimensional Line Plots

- These plots require a set of order triples (x-y-z values) as input

Editor - Untitled2*

```
x = linspace(0,10*pi,1000);
y = cos(x);
z = sin(x);
plot3(x,y,z)
grid
xlabel('angle'), ylabel('cos(x)')
zlabel('sin(x)'), title('A Spring')
```

The z-axis is labeled the same way the x and y axes are labeled



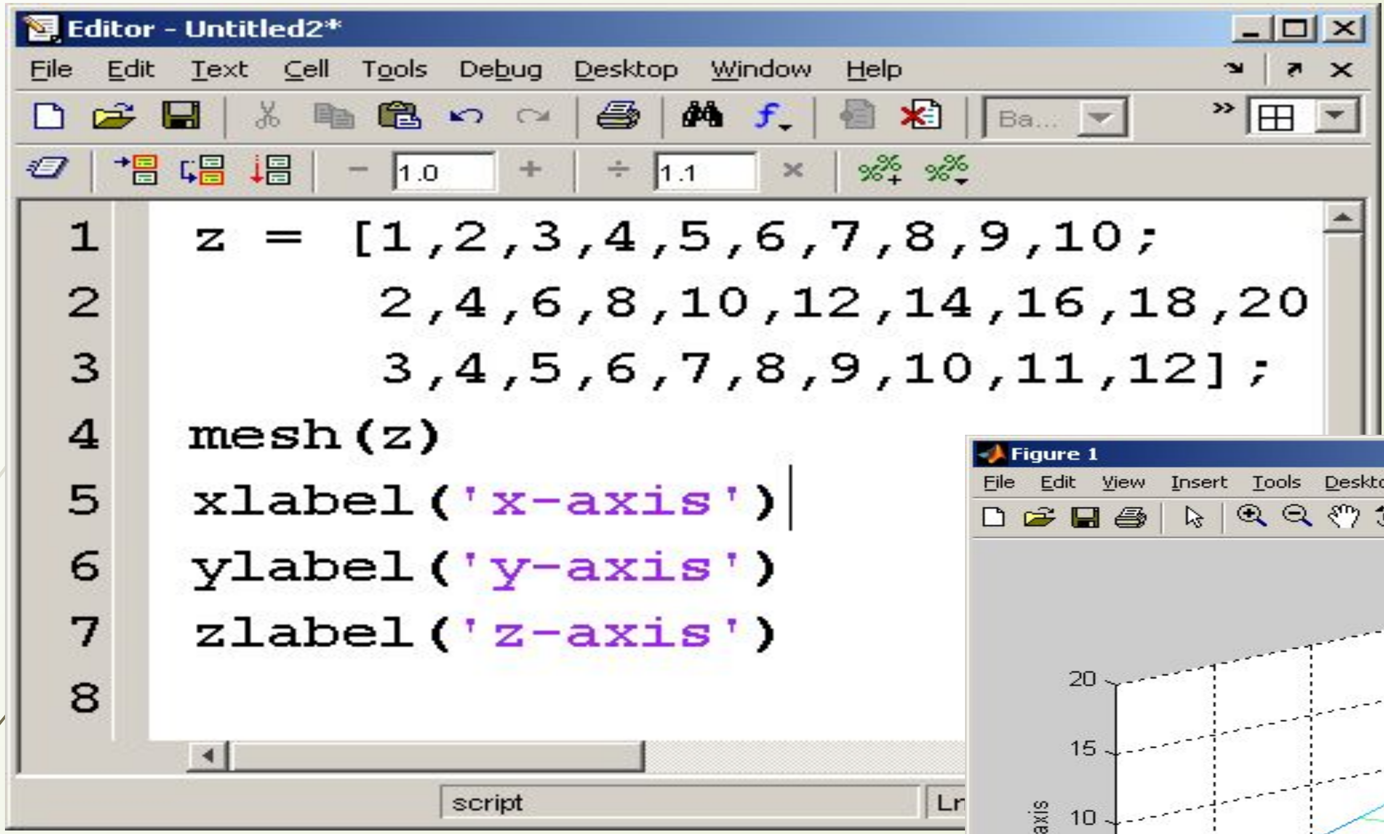
MATLAB uses a coordinate system consistent with the right hand rule

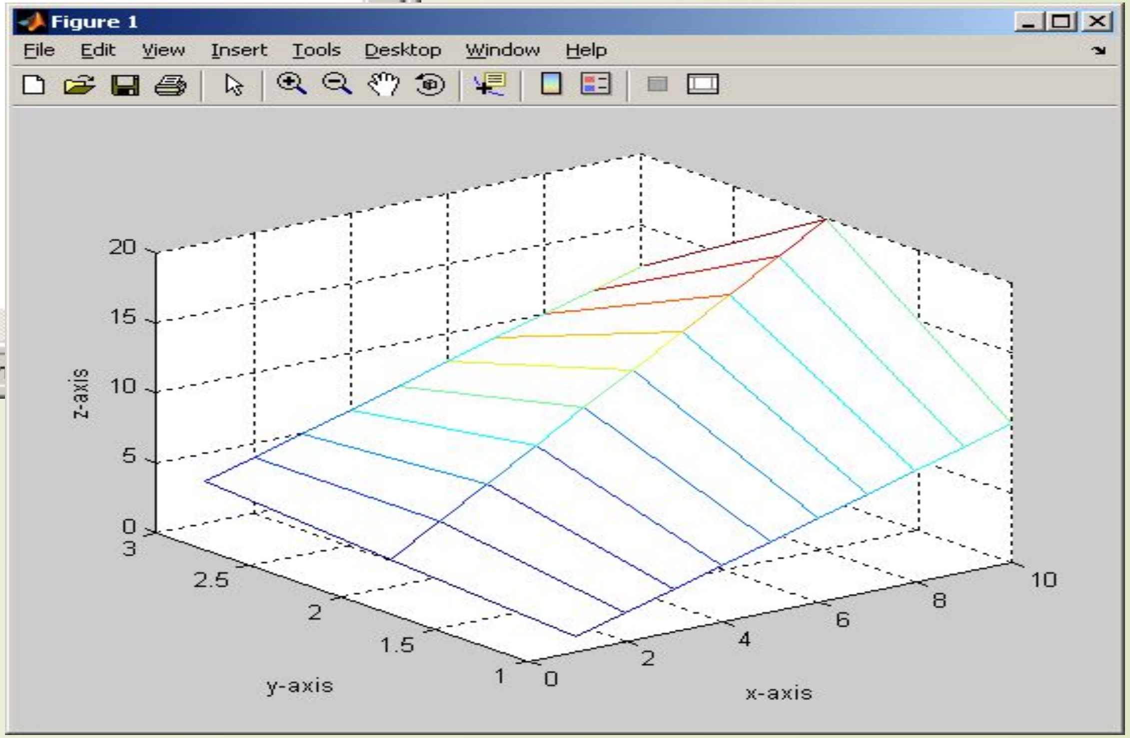
Surface Plots

- Represent x-y-z data as a surface
  - mesh - meshplot
  - surf – surface plot

Both Mesh and Surf

- Can be used to good effect with a single two dimensional matrix

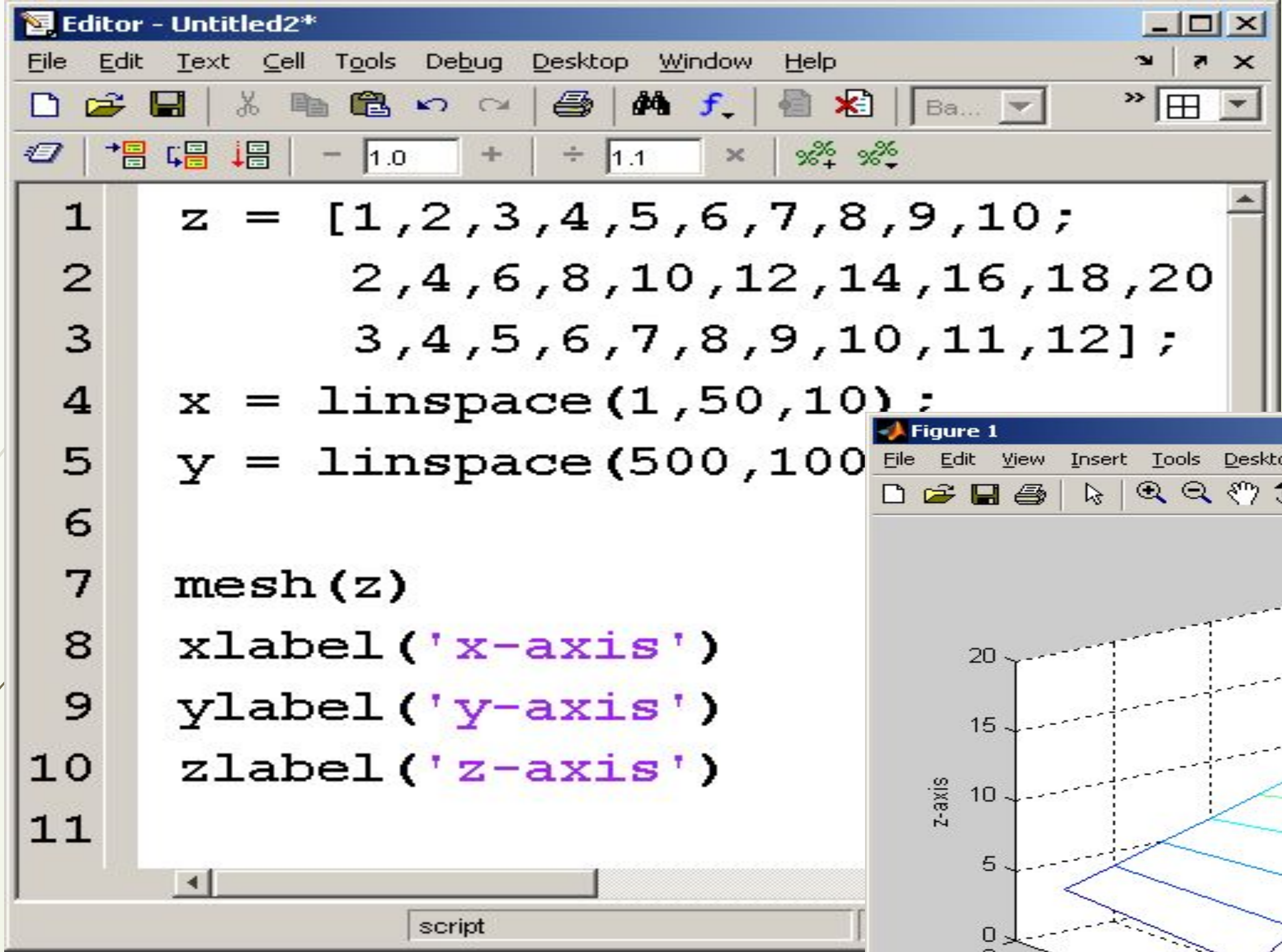




The x and y coordinates are the matrix index numbers

Using mesh with 3 variables

- If we know the values of x and y that correspond to our z values, we can plot against those values instead of the index numbers


Editor - Untitled2*
File Edit Text Cell Tools Debug Desktop Window Help
1.0
1.1
z = [1,2,3,4,5,6,7,8,9,10;
2,4,6,8,10,12,14,16,18,20
3,4,5,6,7,8,9,10,11,12];
x = linspace(1,50,10);
y = linspace(500,100
mesh(z)
xlabel('x-axis')
ylabel('y-axis')
zlabel('z-axis')
script

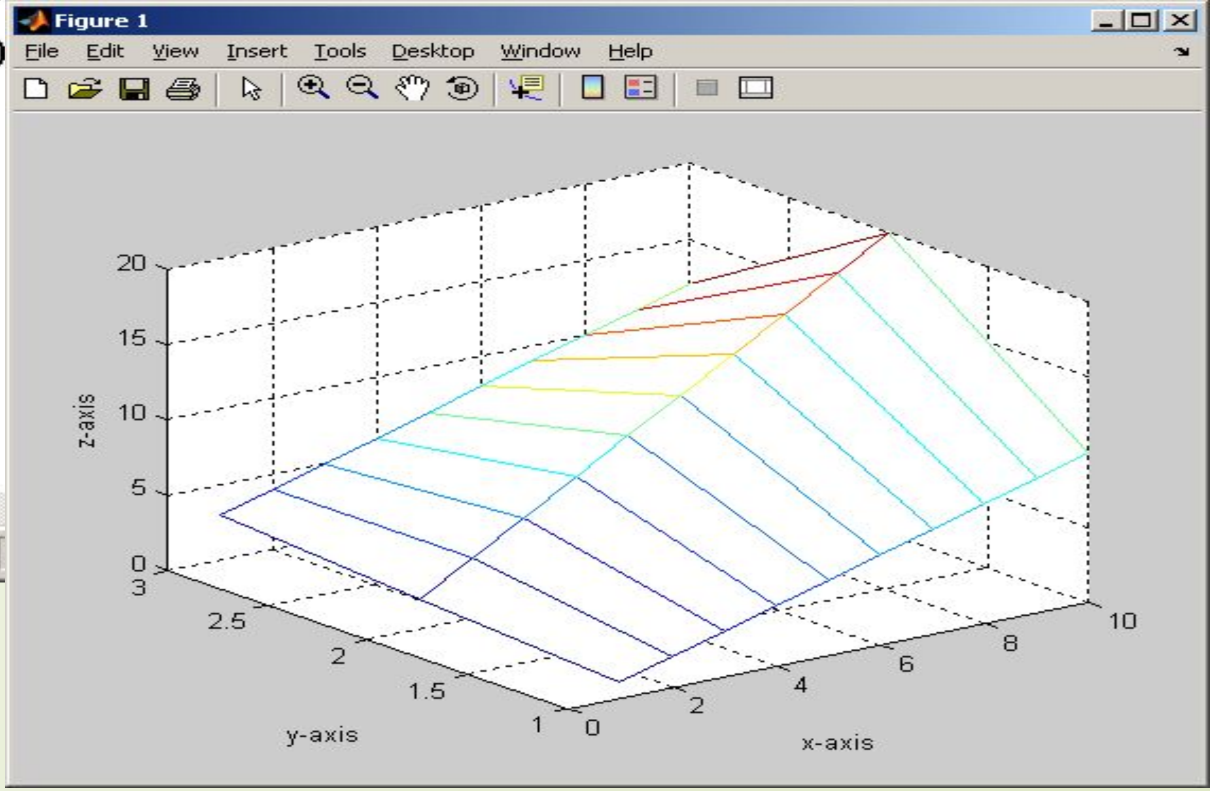
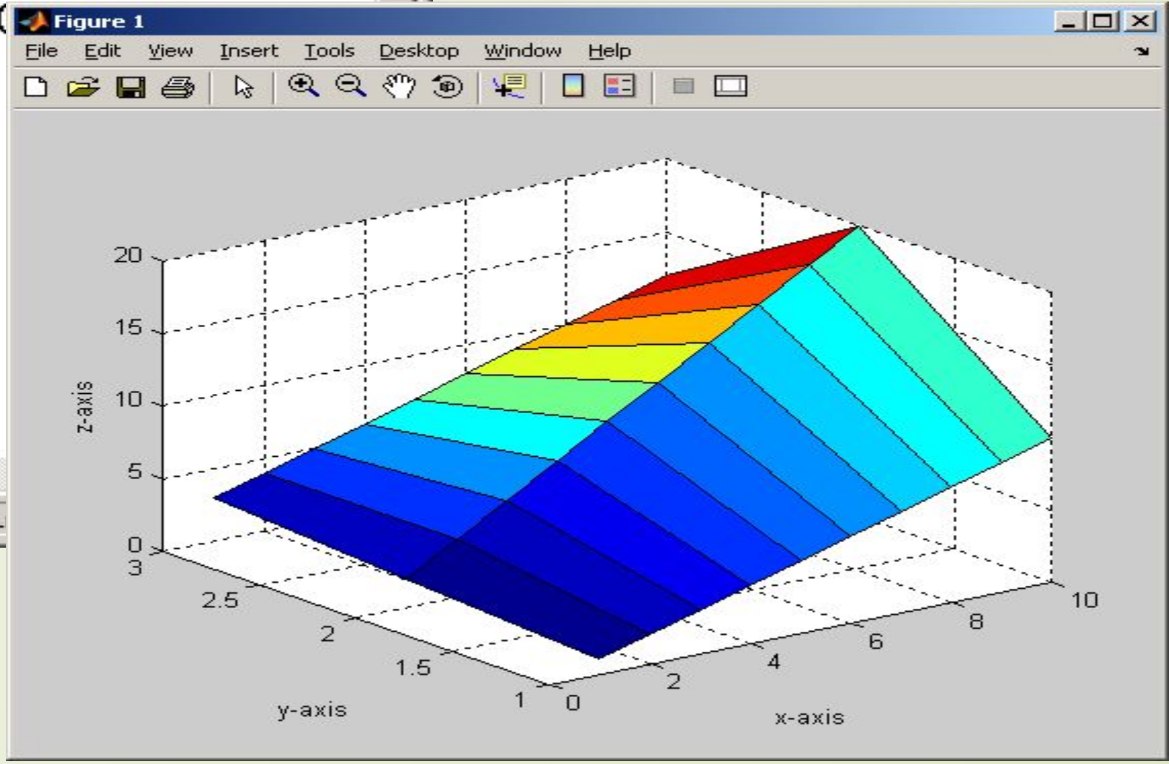


Figure 1
File Edit View Insert Tools Desktop Window Help
z-axis
y-axis
x-axis
20
15
10
5
0
3
2.5
2
1.5
1
0
2
4
6
8
10

Surf plots

- surf plots are similar to mesh plots
  - they create a 3-D colored surface instead of an open mesh
  - syntax is the same

```
z = [1,2,3,4,5,6,7,8,9,10;
      2,4,6,8,10,12,14,16,18,20
      3,4,5,6,7,8,9,10,11,12];
x = linspace(1,50,10);
y = linspace(500,100

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



Shading

- There are several shading options
  - shading interp
  - shading flat
  - faceted flat is the default
- You can also adjust the color scheme with the color map function

Colormaps

autumn	**bone**	**hot**
spring	**colorcube**	**hsv**
summer	**cool**	**pink**
winter	**copper**	**prism**
jet (default)	**flag**	**white**

Contour Plots

- Contour plots use the same input syntax as mesh and surf plots
- They create graphs that look like the familiar contour maps used by hikers

Editor - Untitled2*

File Edit Text Cell Tools Debug Desktop Window Help

Stack: Base

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

subplot(2,2,1)
mesh(X,Y,Z)
title('Mesh Plot'), xlabel('x-axis'), ylabel('y-axis'), zlabel('z-a
subplot(2,2,2)
surf(X,Y,Z)
title('Surface Plot'), xlabel('x-axis'), ylabel('y-axis'), zlabel('
subplot(2,2,3)
contour(X,Y,Z)
xlabel('x-axis'), ylabel('y-axis'), title('Contour Plot')
subplot(2,2,4)
surfc(X,Y,Z)
xlabel('x-axis'), ylabel('y-axis')
title('Combination Surface and Contour Plot')
```

script | Ln 14 | Col 58 | OVR

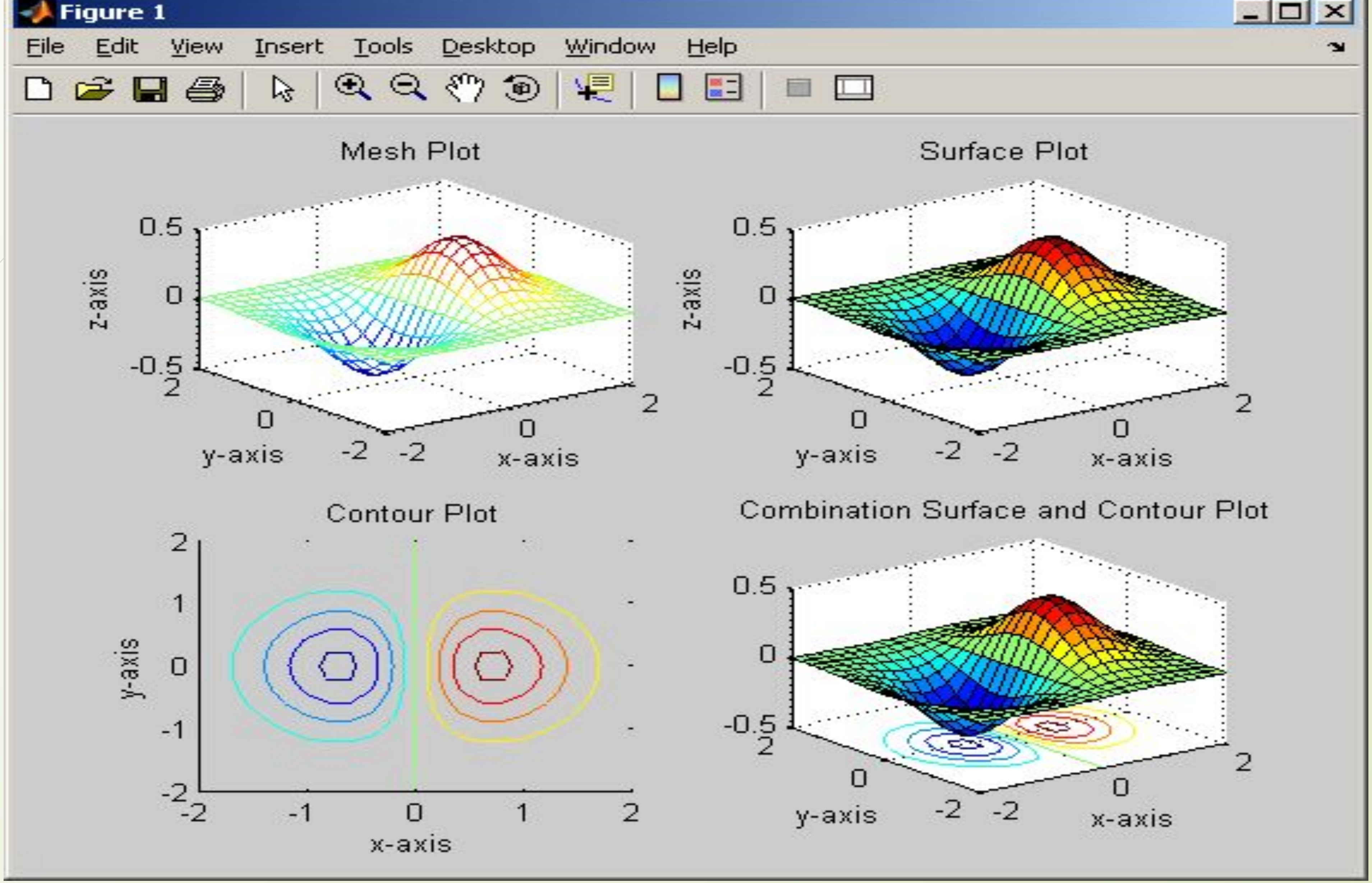


Figure 1
File Edit View Insert Tools Desktop Window Help
Mesh Plot
Surface Plot
Contour Plot
Combination Surface and Contour Plot
z-axis
y-axis
x-axis
0.5
0
-0.5
2
0
-2
-1
1

Editor - Untitled2*

File Edit Text Cell Tools Debug Desktop Window Help

Stack: Base

```
[x,y,z] = peaks;

subplot(2,2,1)
pcolor(x,y,z)

subplot(2,2,2)
pcolor(x,y,z)
shading interp

subplot(2,2,3)
pcolor(x,y,z)
shading interp
hold on
contour(x,y,z,20,'k')

subplot(2,2,4)
contour(x,y,z)
```

script | Ln 15 Col 1 | OVR

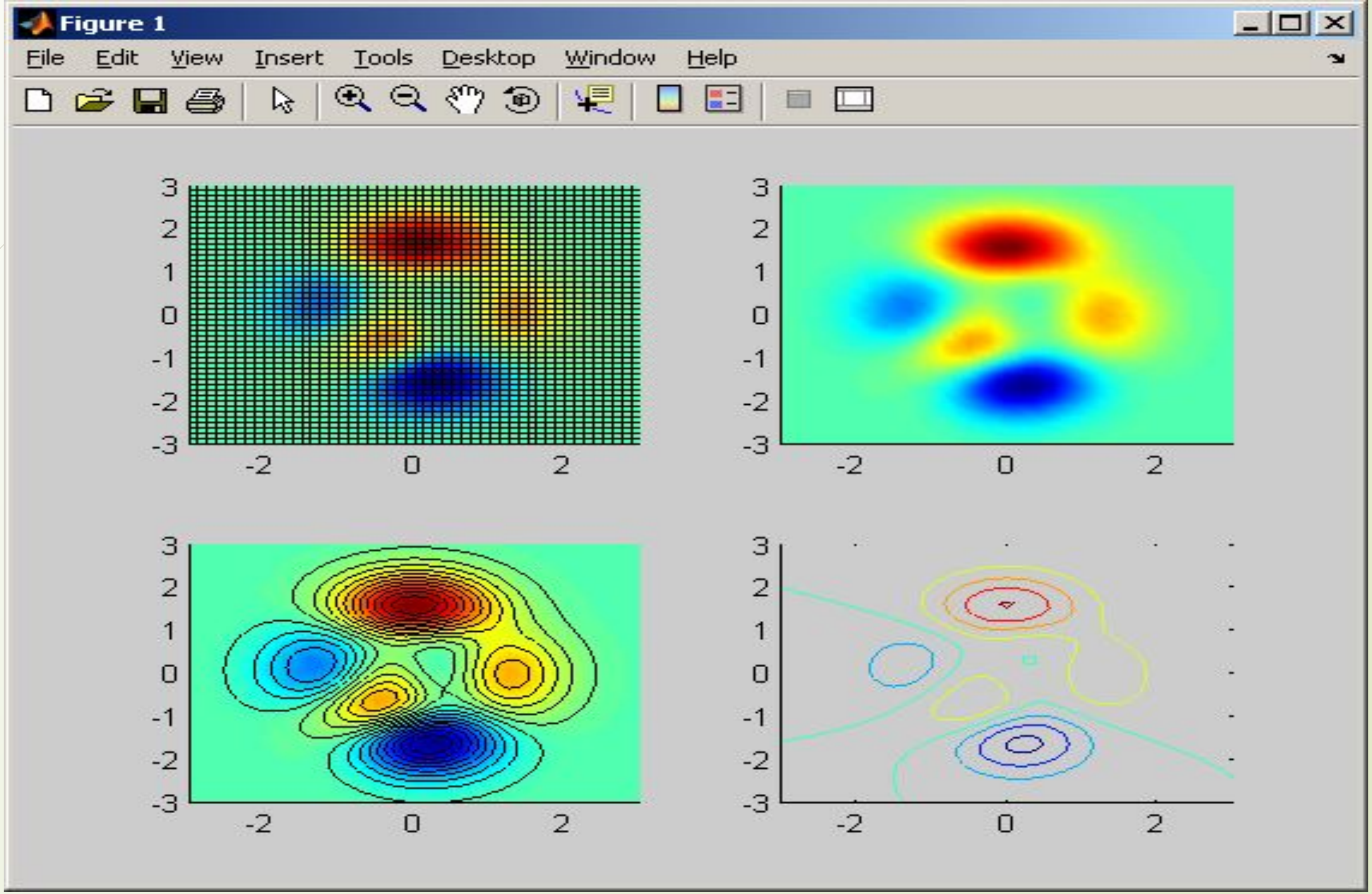


Figure 1
File Edit View Insert Tools Desktop Window Help
3
2
1
0
-1
-2
-3
-2 0 2

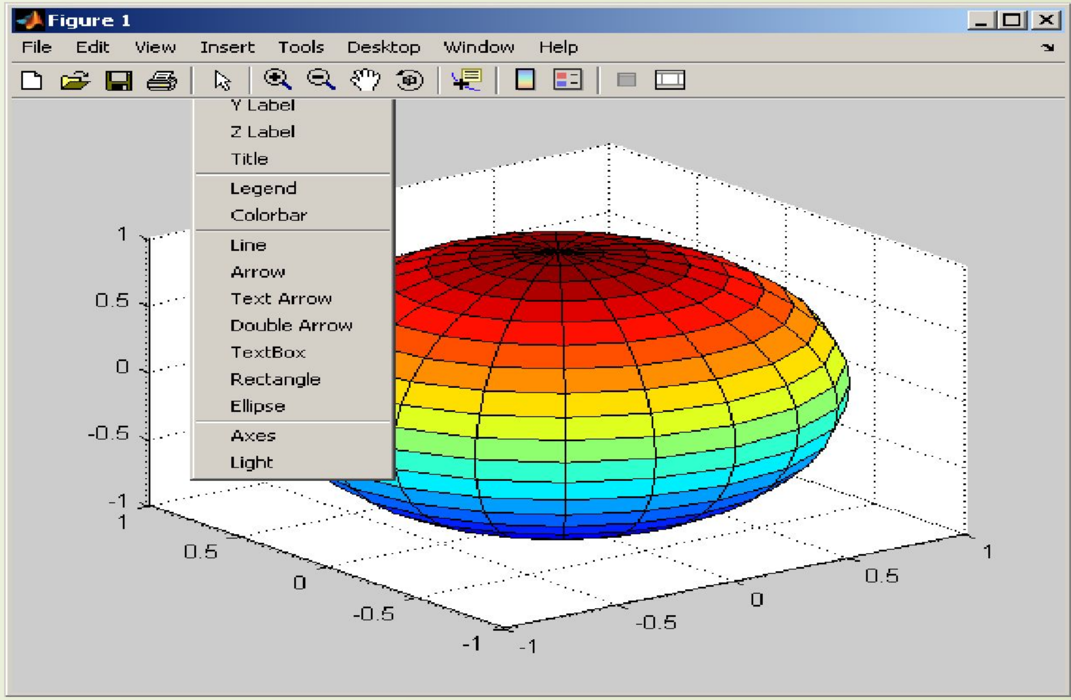
Editing Plots from the Menu Bar

- In addition to controlling the way your plots look by using MATLAB commands, you can also edit a plot once you’ve created it using the menu bar
- Another demonstration function built into MATLAB is

  sphere

Once you've created a plot you can adjust it using the menu bar

- In this picture the insert menu has been selected
- Notice you can use it to add labels, legends, a title and other annotations



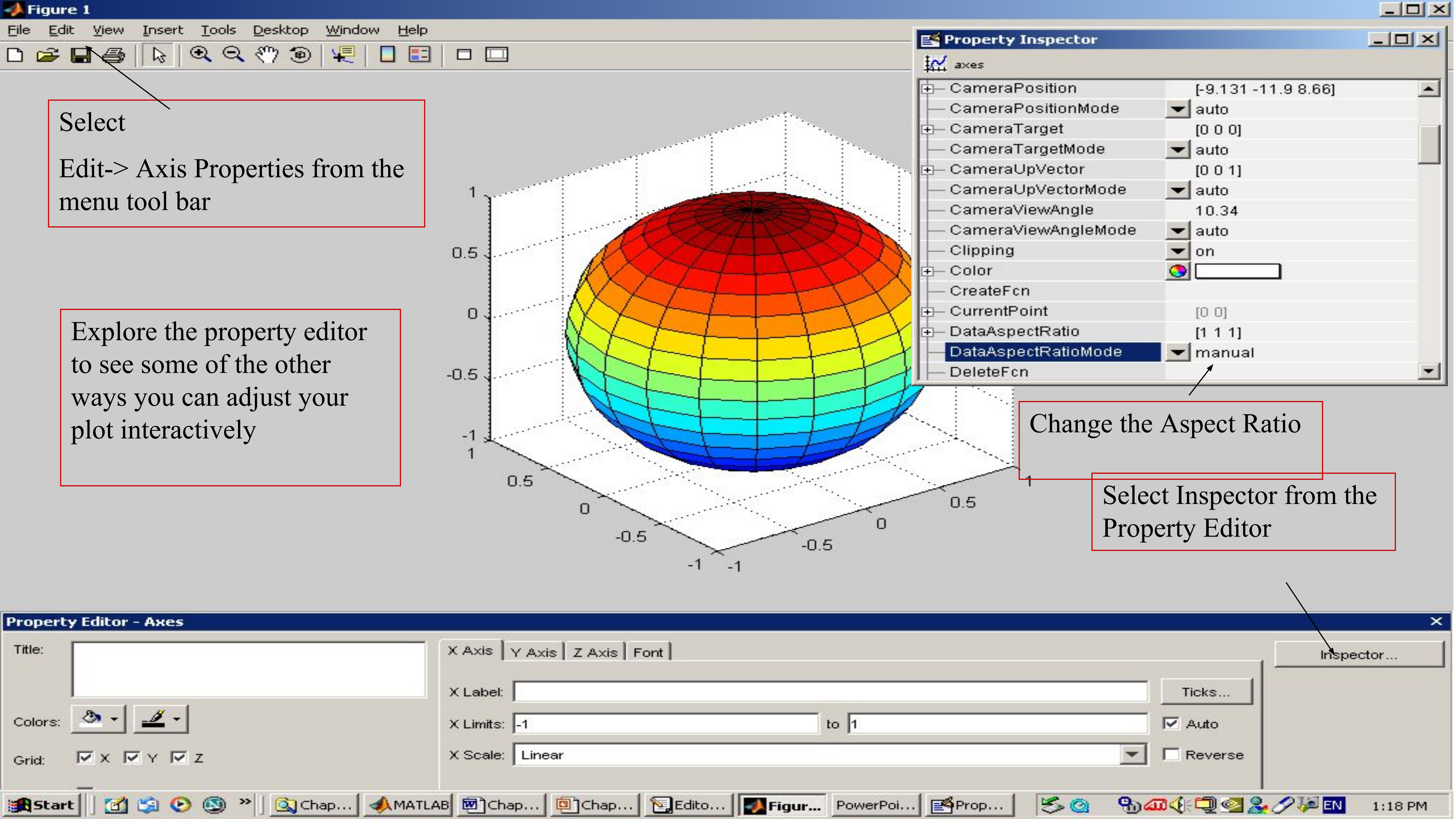
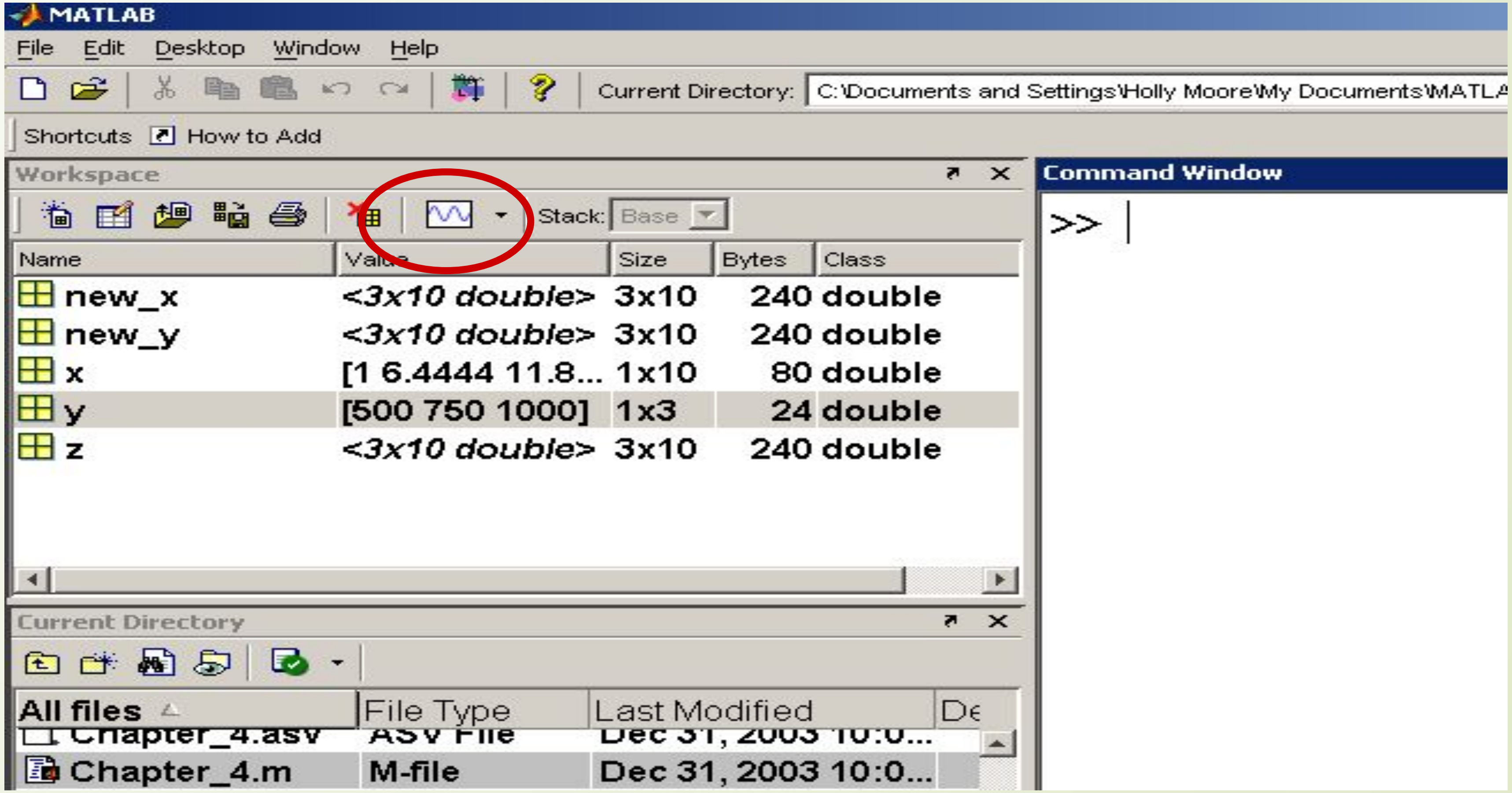


Figure 1
File Edit View Insert Tools Desktop Window Help
Select
Edit-> Axis Properties from the menu tool bar
Explore the property editor to see some of the other ways you can adjust your plot interactively
Property Inspector
axes
CameraPosition [-9.131 -11.9 8.66]
CameraPositionMode auto
CameraTarget [0 0 0]
CameraTargetMode auto
CameraUpVector [0 0 1]
CameraUpVectorMode auto
CameraViewAngle 10.34
CameraViewAngleMode auto
Clipping on
Color
CreateFcn
CurrentPoint [0 0]
DataAspectRatio [1 1 1]
DataAspectRatioMode manual
DeleteFcn
Change the Aspect Ratio
Select Inspector from the Property Editor
Property Editor - Axes
Title:
Colors:
Grid: X Y Z
X Axis Y Axis Z Axis Font
X Label:
X Limits: -1 to 1
X Scale: Linear
Ticks...
Auto
Reverse
Inspector...
Start
1:18 PM

Creating plots from the workspace



Thank You

QR CODE