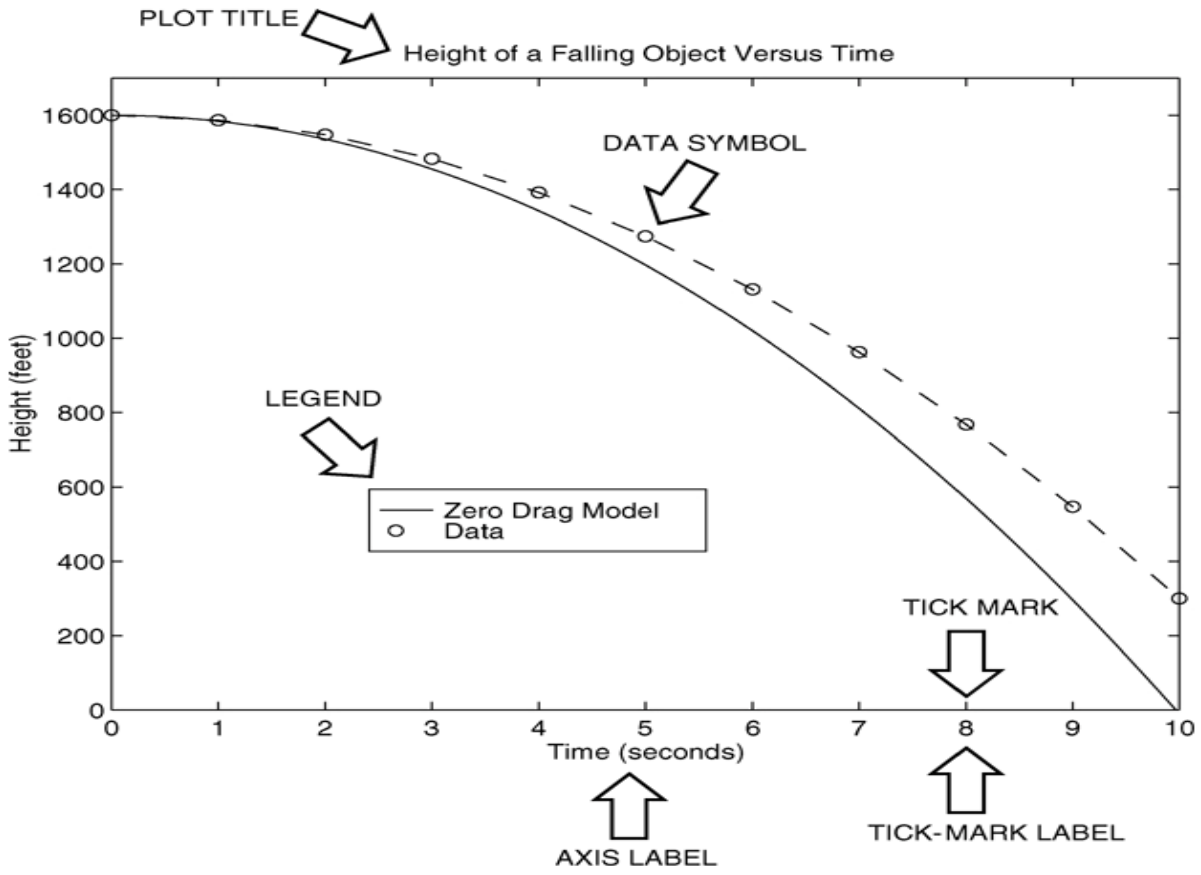



MATLAB Course
For Engineers
lecture 4


Plotting → **Two-Dimensional plots**

→ **Three-Dimensional plots**

➢ **Nomenclature for a typical 2-D plot**



MATLAB (2-D) plotting commands

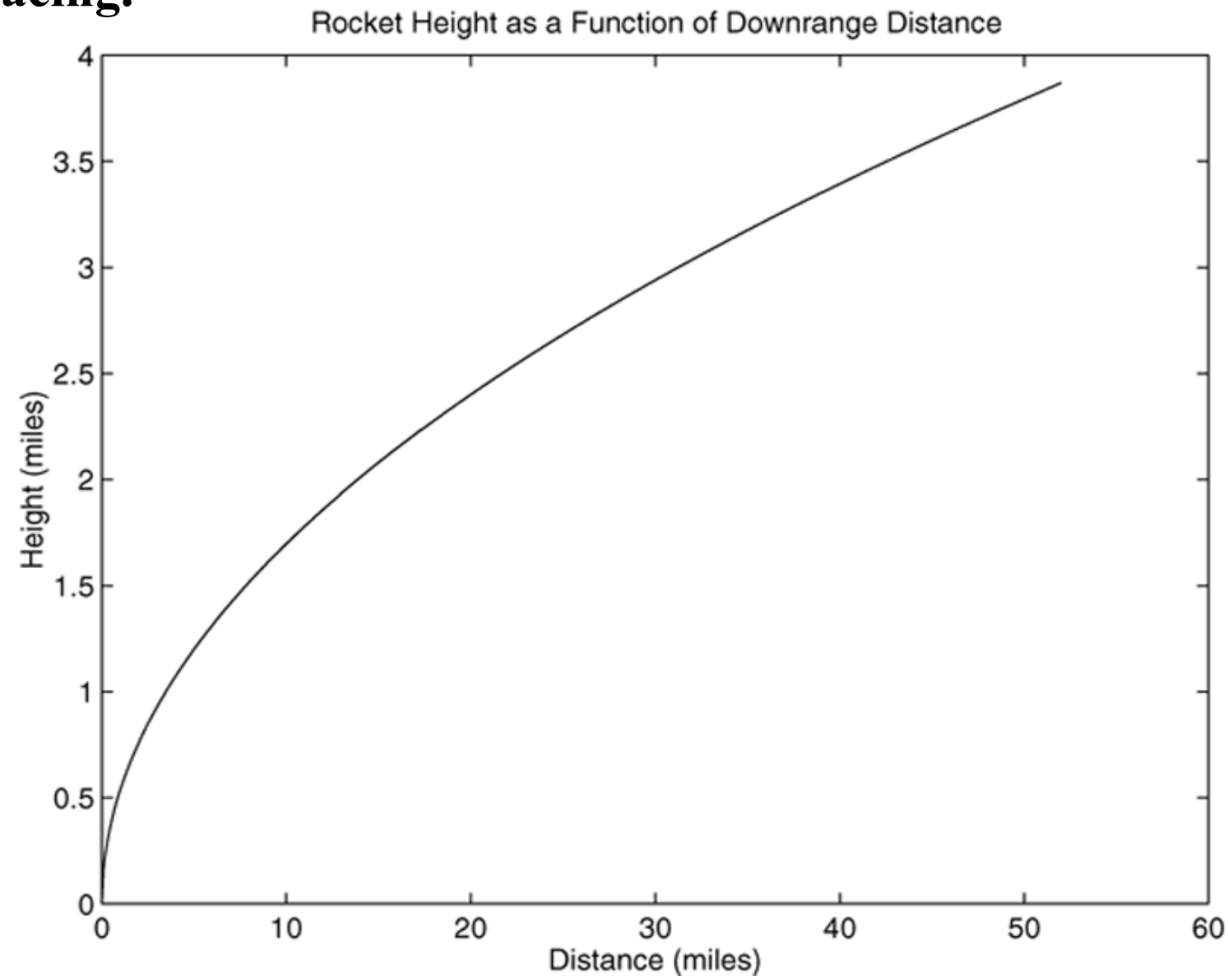
Command	Description
`plot(x,y)`	Generates a plot of the array y versus the array x on rectilinear axes.
`title('text')`	Puts text in a title at the top of the plot.
`xlabel('text')`	Adds a text label to the horizontal axis
`ylabel('text')`	Adds a text label to the vertical axis
`grid`	Puts grid lines on the plot.

Example

plot $y = 0.4\sqrt{1.8x}$ for $0 \le x \le 52$, where y represents the height of a rocket after launch, in miles, and x is the horizontal (downrange) distance in miles.

```
>>x = [0:0.1:52];
>>y = 0.4*sqrt(1.8*x);
>>plot(x,y)
>>xlabel('Distance (miles)')
>>ylabel('Height (miles)')
>>title('Rocket Height as a Function
  of Downrange Distance')
```

The auto-scaling feature in MATLAB selects tick-mark spacing.

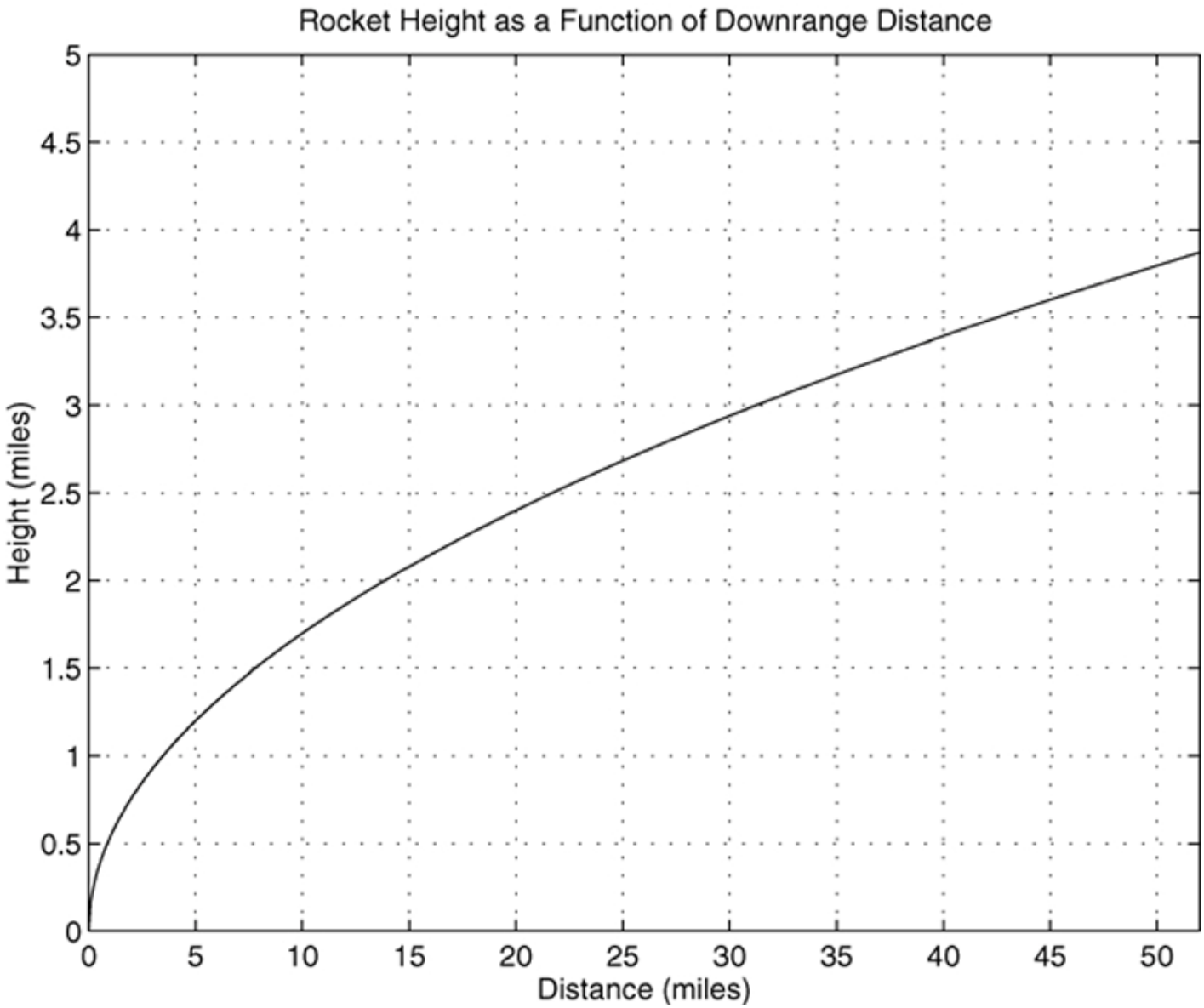


The `grid` and `axis` Commands

The `grid` command displays gridlines at the tick marks corresponding to the tick labels. Type `grid on` to add gridlines; type `grid off` to stop plotting gridlines. When used by itself, `grid` toggles this feature on or off, but you might want to use `grid on` and `grid off` to be sure.

You can use the `axis` command to override the MATLAB selections for the axis limits. The basic syntax is `axis([xmin xmax ymin ymax])`.
This command sets the scaling for the *x*- and *y*-axes to the minimum and maximum values indicated.
Note that, unlike an array, this command does not use commas to separate the values.

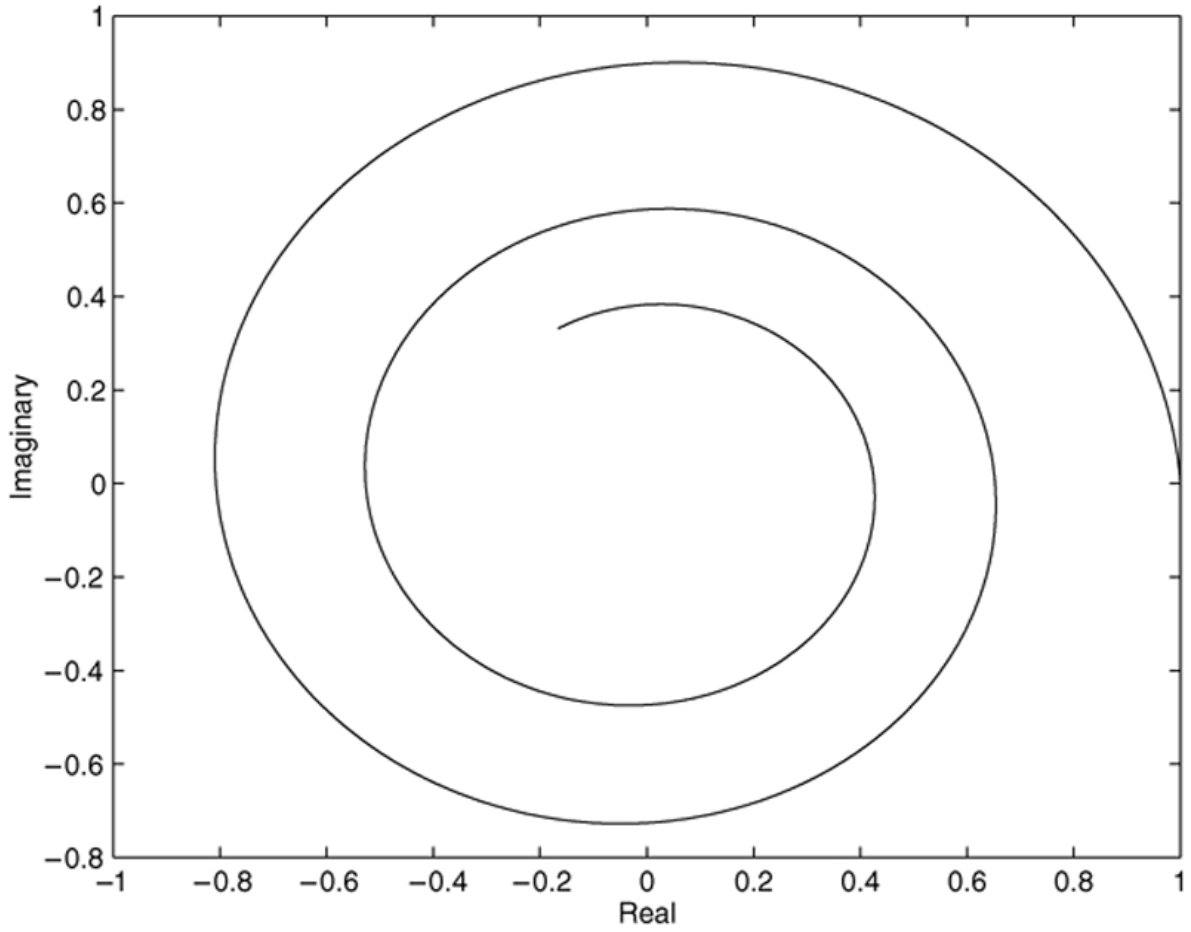
The effects of the `axis` and `grid` commands.



The `plot(y)` function plots the values in y versus the indices.

Example:

```
x = 0.1+0.9j; n = [0:0.01:10];y = x.^n;
plot(y), xlabel('Real'),ylabel('Imaginary')
```



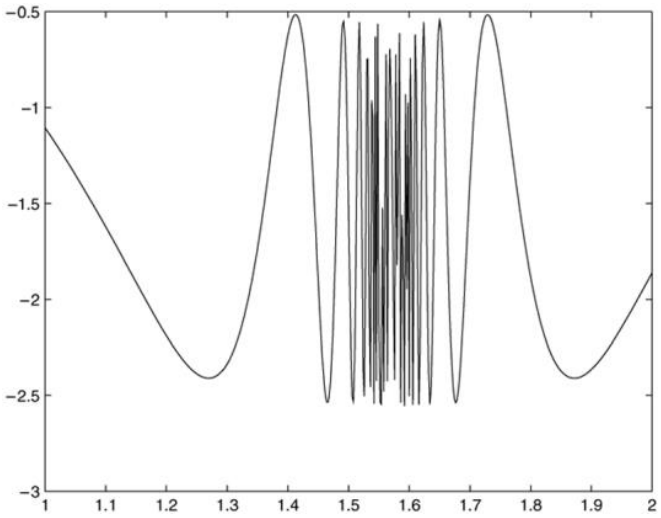
The Function Plot Command fplot

The fplot command automatically analyzes the function to be plotted and decides how many plotting points to use so that the plot will show all the features of the function. Its syntax is

```
fplot('function ',[xmin xmax ymin ymax])
```

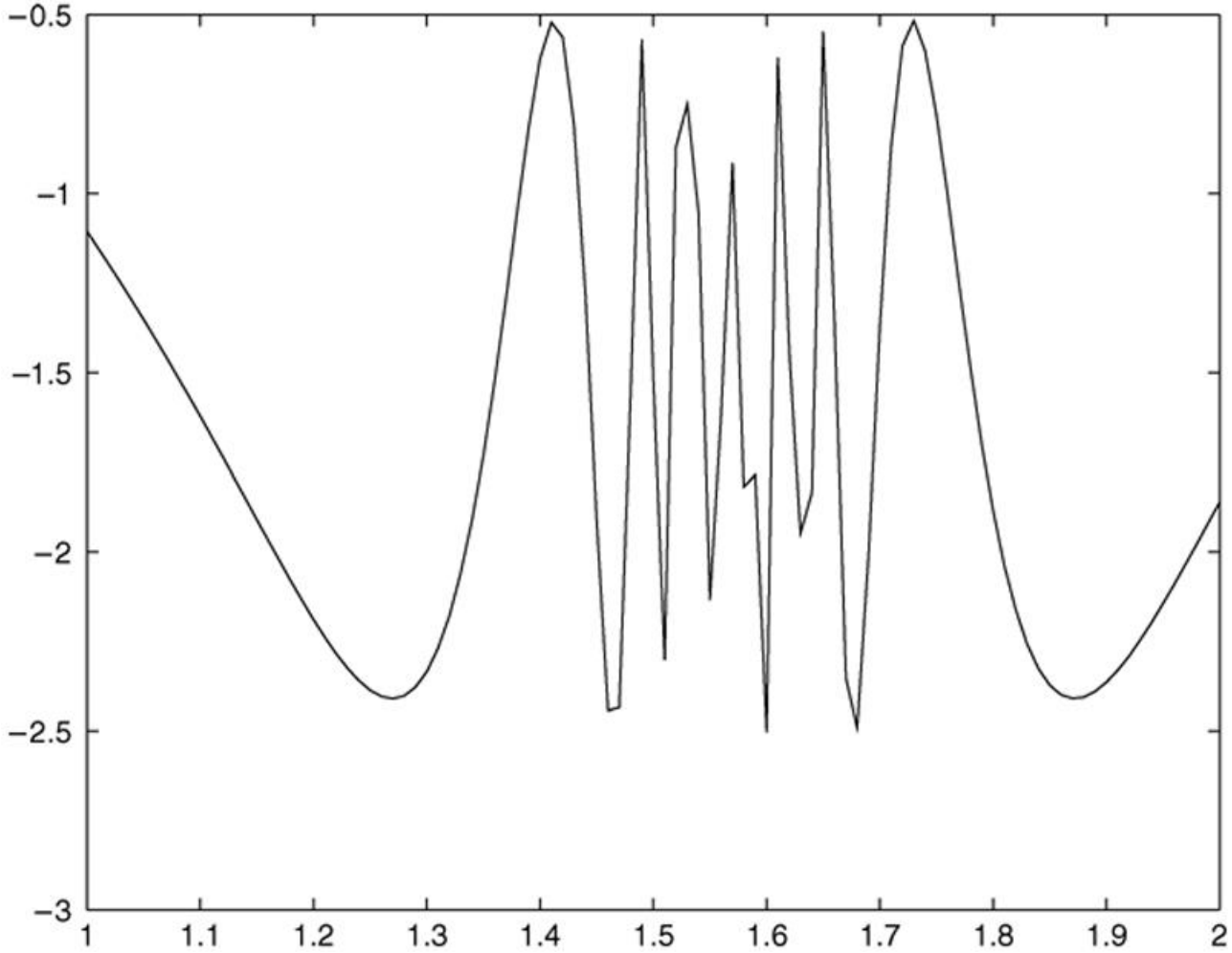
For Example:

```
>> f = @(x) (cos(tan(x))-tan(sin(x)));
>> fplot(f,[1 2])
```



The command `plot` gives more control than the `fplot` command.

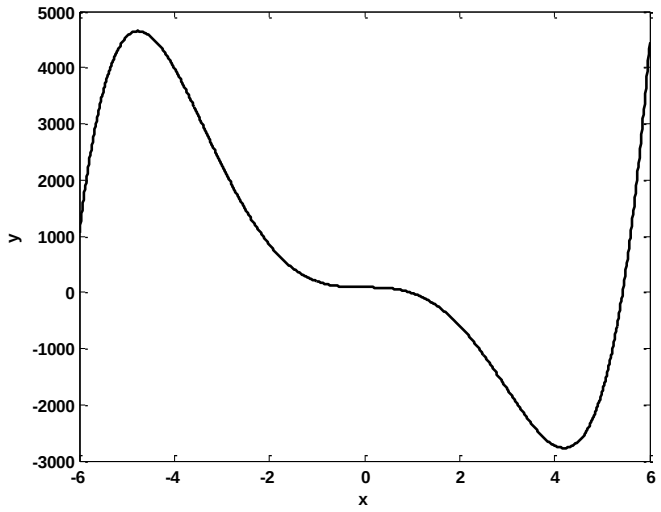
```
>>x=[1:0.01:2]; y=cos(tan(x))-tan(sin(x));
>>plot(x, y)
```



Plotting Polynomials with the `polyval` Function.

To plot the polynomial $3x^5 + 2x^4 - 100x^3 + 2x^2 - 7x + 90$ over the range $-6 \leq x \leq 6$ with a spacing of 0.01, you type

```
>>x = [-6:0.01:6];
>>p = [3,2,-100,2,-7,90];
>>plot(x,polyval(p,x)),xlabel('x'), ...
ylabel('p')
```



Subplots

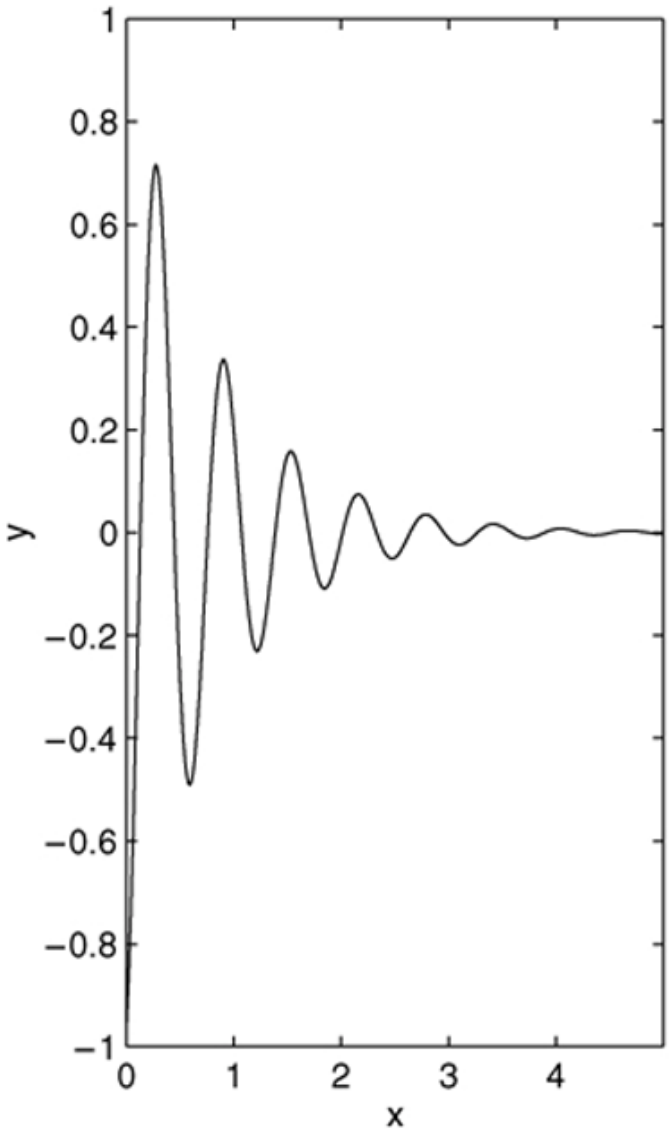
You can use the `subplot` command to obtain several smaller "subplots" in the same figure. The syntax is `subplot(m,n,p)`. This command divides the Figure window into an array of rectangular panes with m rows and n columns. The variable p tells MATLAB to place the output of the `plot` command following the `subplot` command into the pth pane.

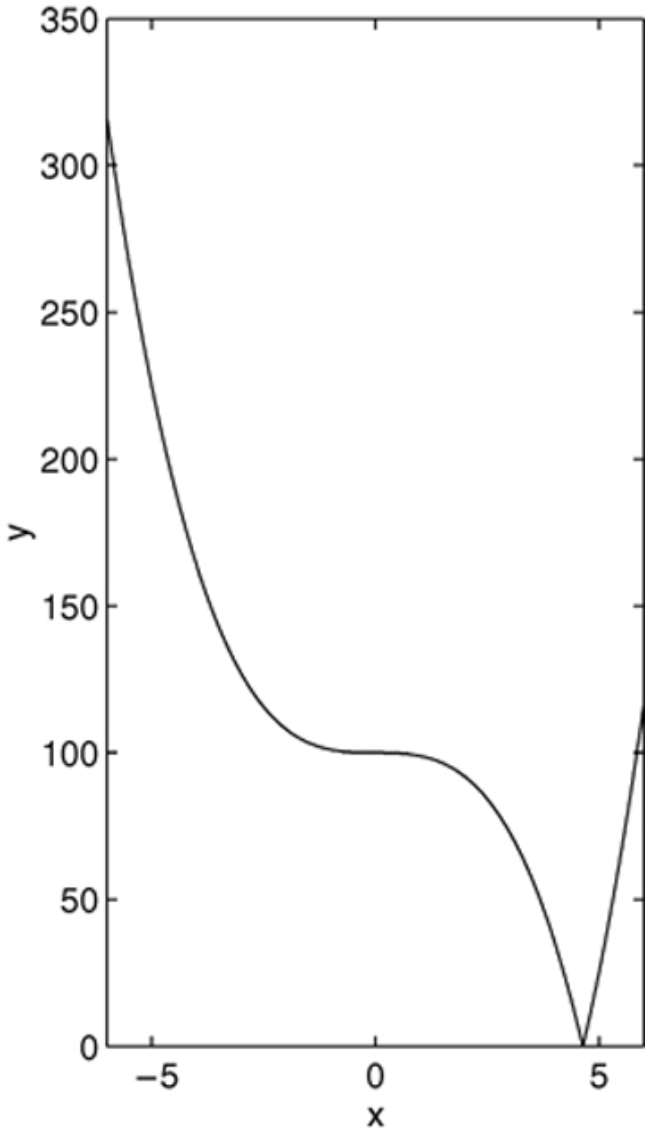
For example, `subplot(3,2,5)` creates an array of six panes, three panes deep and two panes across, and directs the next plot to appear in the fifth pane (in the bottom-left corner).

The following script file created Figure, which shows the plots of the functions $y = e^{-1.2x} \sin(10x + 5)$ for $0 \le x \le 5$ and $y = |x^3 - 100|$ for $-6 \le x \le 6$.

```
x = [0:0.01:5];
y = exp(-1.2*x).*sin(10*x+5);
subplot(1,2,1)
plot(x,y),axis([0 5 -1 1])
x = [-6:0.01:6];
y = abs(x.^3-100);
subplot(1,2,2)
plot(x,y),axis([-6 6 0 350])
```

Application of the subplot command





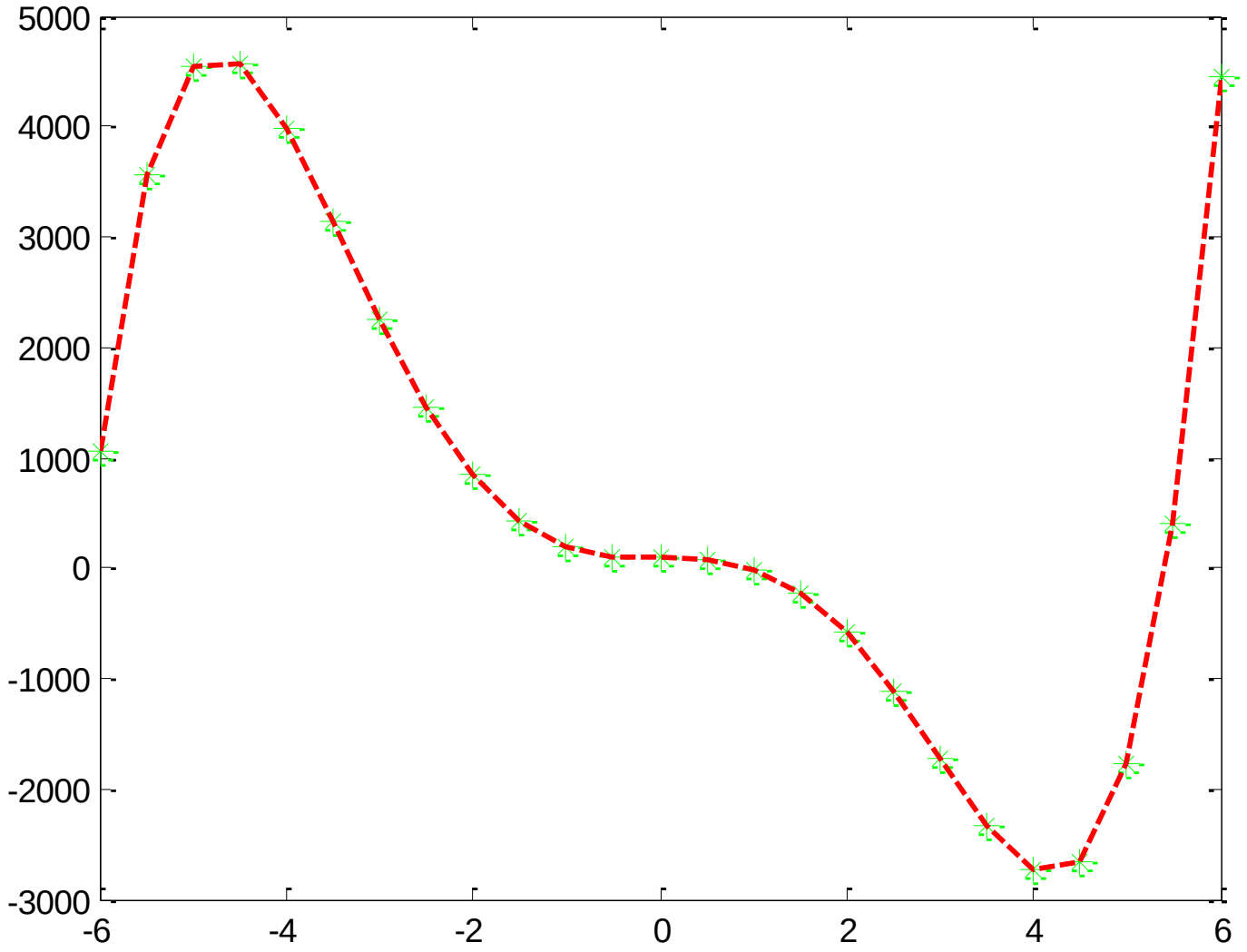
Data Markers and Line Types

To plot y versus x with a solid line and u versus v with a dashed line, type `plot(x,y,u,v,'--')`, where the symbols `'--'` represent a dashed line.

To plot y versus x with asterisks (*) connected with a dotted line, you plot the data by typing `plot(x,y,'*:')`.

To plot y versus x with green asterisks (*) connected with a red dashed line(--), you plot the data by typing `plot(x,y,'g*',x,y,'r--')`.

Data plotted using green asterisks(*) connected with a red dashed(--) line.



Specifiers for data markers, line types, and colors.

Data markers†		Line types		Colors	
Dot (·)	·	Solid line	-	Black	k
Asterisk (*)	*	Dashed line	- -	Blue	b
Cross (×)	×	Dash-dotted line	-.	Cyan	c
Circle (o)	o	Dotted line	:	Green	g
Plus sign (+)	+			Magenta	m
Square (□)	s			Red	r
Diamond (◇)	d			White	w
Five-pointed star (★)	p			Yellow	y

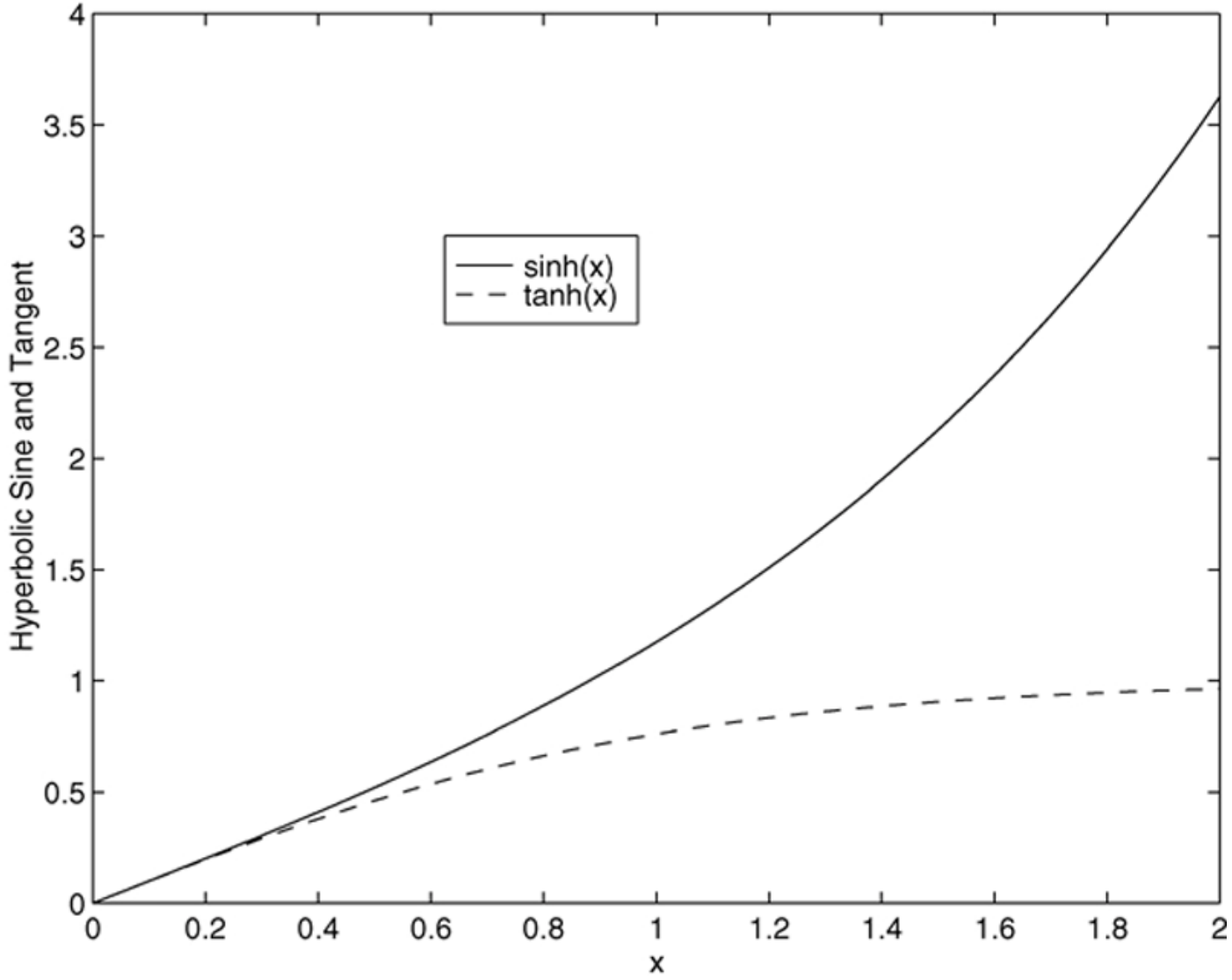
†Other data markers are available. Search for "markers" in MATLAB Help.

Labeling Curves and Data

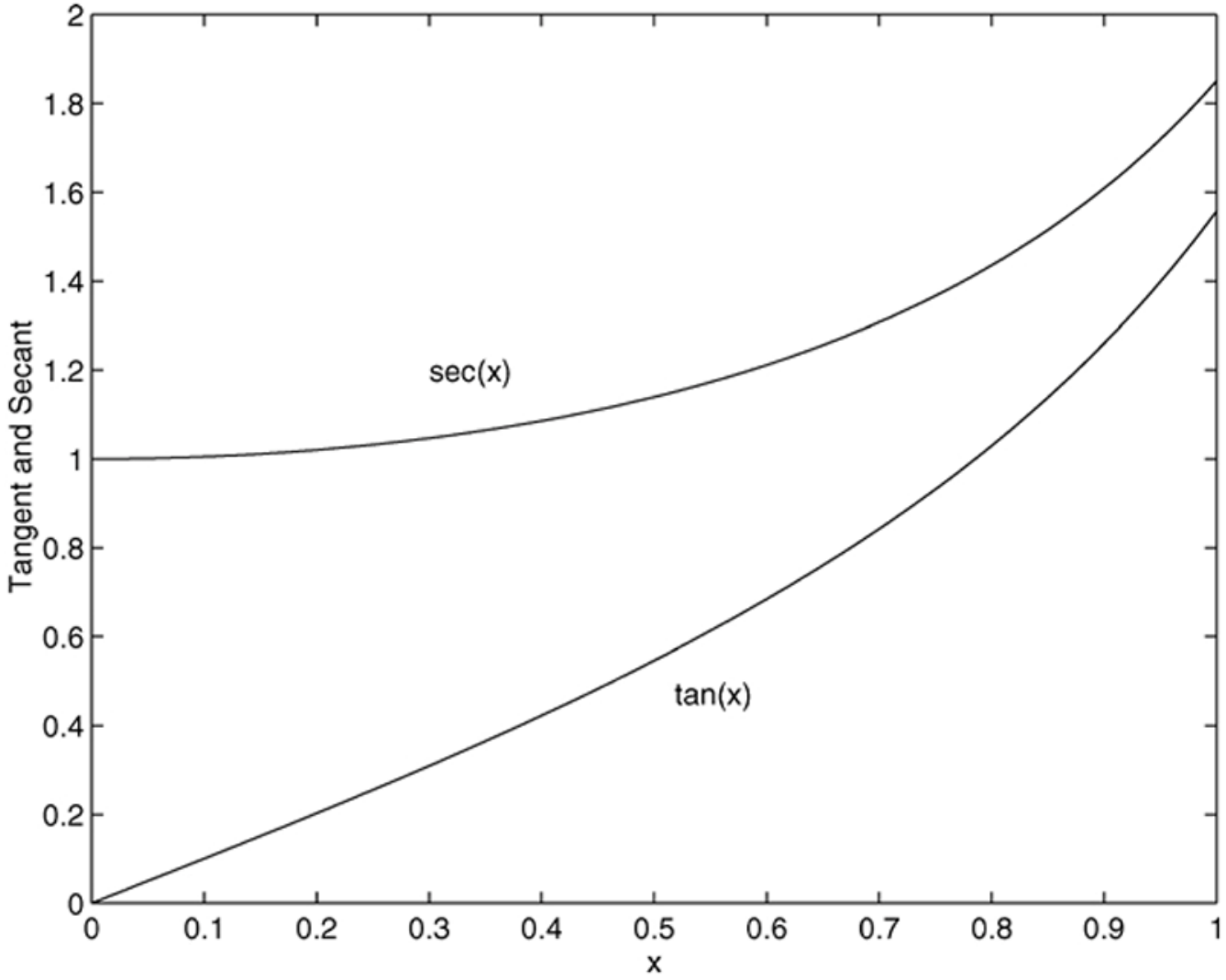
The `legend` command automatically obtains from the plot the line type used for each data set and displays a sample of this line type in the legend box next to the string you selected. The following script file produced the plot in Figure.

```
x = [0:0.01:2];
y = sinh(x);
z = tanh(x);
plot(x,y,x,z,'--'),xlabel('x'), ...
ylabel('Hyperbolic Sine and Tangent'), ...
legend('sinh(x)','tanh(x)')
```

Application of the `legend` command.



The `gtext('string')` and `text(x,y,'string')` commands are also useful.



The `hold` Command

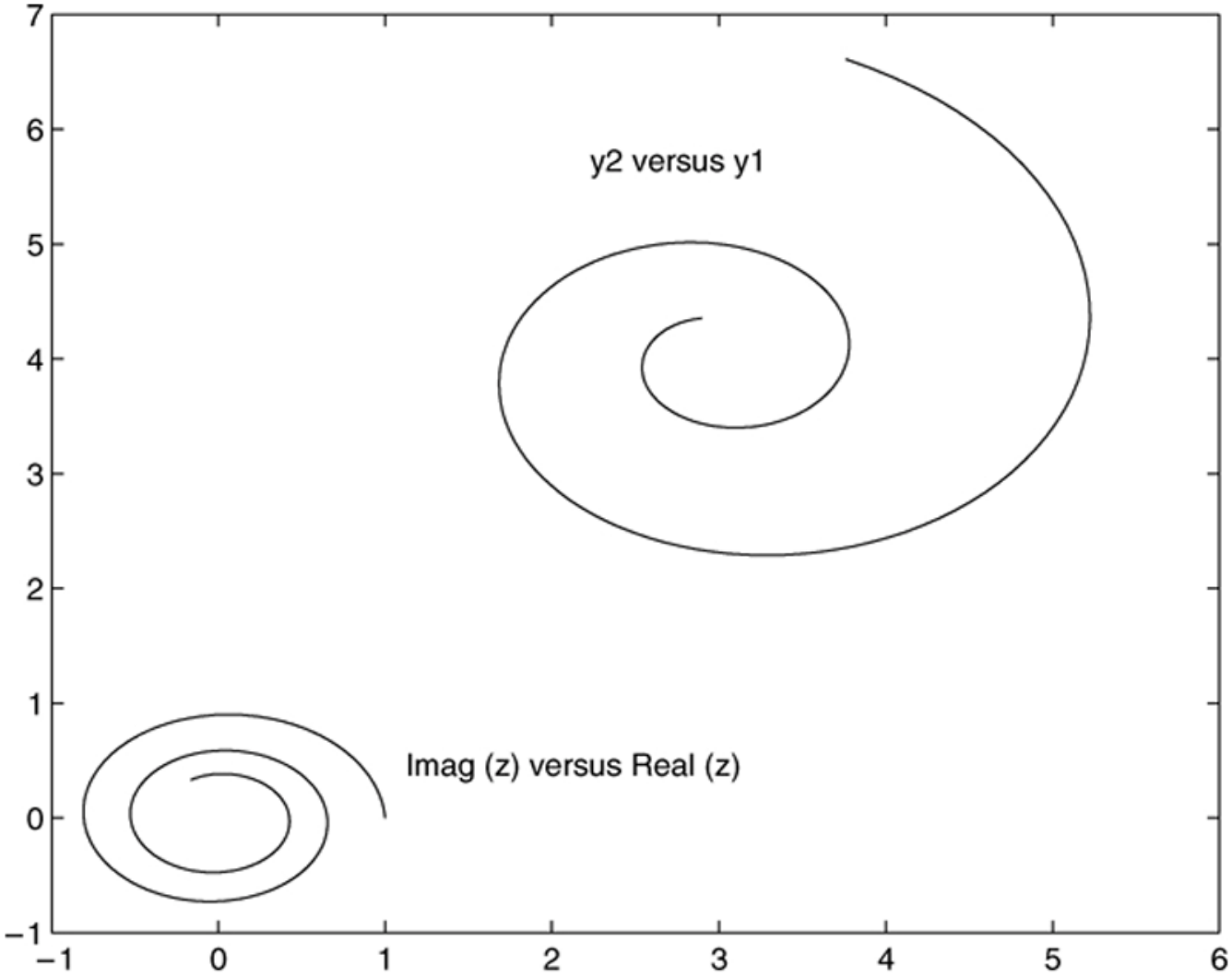
The `hold` command creates a plot that needs two or more plot commands.

Example:

Plot $y2= 4+e^{-x}\cos(6x)$ **versus** $y1= 3+e^{-x}\sin(6x)$, $-1 \le x \le 1$ **on the same plot with the complex function** $z=(0.1+0.9i)^n$ **,where** $0 \le n \le 10$**. The following script file creates the plot in figure.**

```
x=[-1:0.01:1];,n=[0:0.01:10];
y1=3+exp(-x).*sin(6*x);
y2= 4+exp(-x).*cos(6*x);
z=(0.1+0.9i).^n;
plot(z),hold on
plot(y1,y2),gtext('y2 versus y1'),…
gtext('Imag(z) versus Real(z)')
```

Application of the `hold` command.



Logarithmic Plots

Why use log scales?

Rectilinear scales cannot properly display variations over wide ranges.

MATLAB has three commands for generating plots having log scales. The appropriate command depends on which axis must have a log scale.

1. **Use the `loglog(x,y)` command to have both scales logarithmic.**
2. **Use the `semilogx(x,y)` command to have the x scale logarithmic and the y scale rectilinear.**
3. **Use the `semilogy(x,y)` command to have the y scale logarithmic and the x scale rectilinear.**

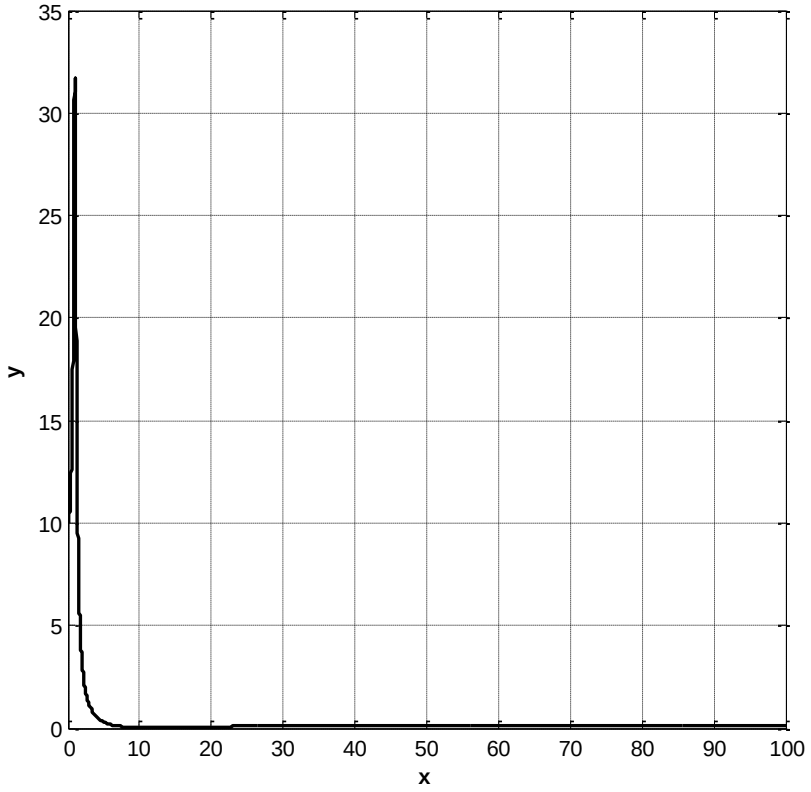
Example :

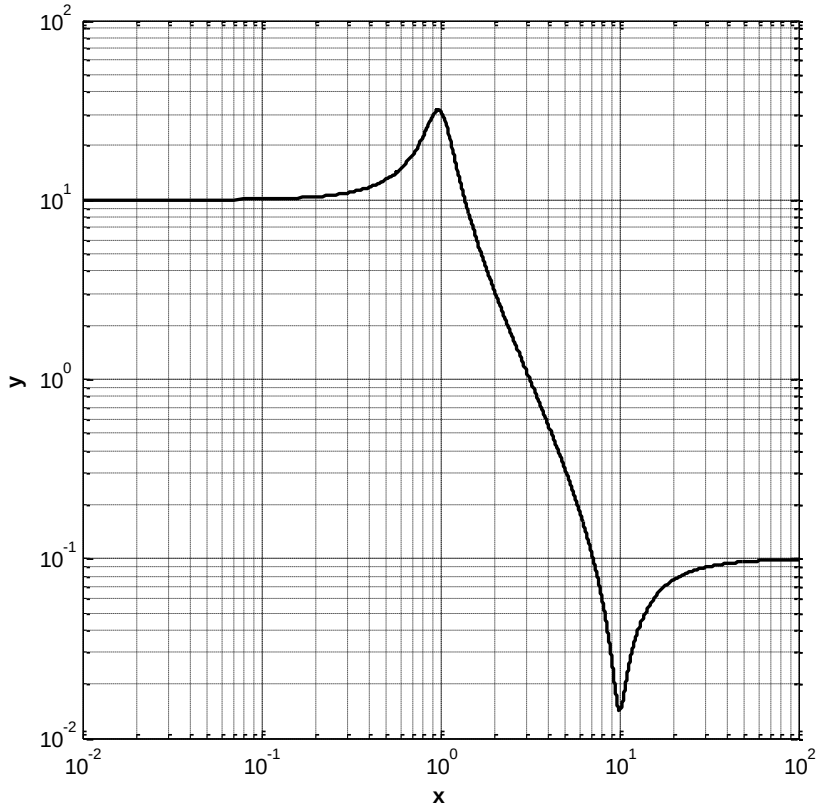
Plot the function $y = \sqrt{\dfrac{100(1 - 0.01x^2)^2 + 0.02x^2}{(1 - x^2)^2 + 0.1x^2}}$ $0.1 \le x \le 100$

The following script file creates the plot in figure.

```
%Create the Rectilinear plot
x1=[0:0.01:100]; u1=x1.^2;
num1=100*(1-0.01*u1).^2+0.02*u1;
den1=(1-u1).^2+0.1*u1;
y1=sqrt(num1./den1);
subplot(1,2,1),
plot(x1,y1),xlabel('x'),ylabel('y')
%Create the log log plot
subplot(1,2,2)
loglog(x1,y1),xlabel('x'),ylabel('y')
```

$$y = \sqrt{\frac{100(1 - 0.01x^2)^2 + 0.02x^2}{(1 - x^2)^2 + 0.1x^2}} \qquad 0.1 \le x \le 100$$





It is important to remember the following points when using log scales:

1. **You cannot plot negative numbers on a log scale, because the logarithm of a negative number is not defined as a real number.**
2. **You cannot plot the number 0 on a log scale, because $\log_{10} 0 = \ln 0 = -\infty$. You must choose an appropriately small number as the lower limit on the plot.**
3. **The tick-mark labels on a log scale are the actual values being plotted; they are not the logarithms of the numbers. For example, the range of x values in the plot in previous Figure is from $10^{-2} = 0.1$ to $10^{2} = 100$.**

4. **Gridlines and tick marks within a decade are unevenly spaced. If 8 gridlines or tick marks occur within the decade, they correspond to values equal to 2, 3, 4, . . . , 8, 9 times the value represented by the first gridline or tick mark of the decade.**
5. **Equal distances on a log scale correspond to multiplication by the same constant (as opposed to addition of the same constant on a rectilinear scale).**

For example, all numbers that differ by a factor of 10 are separated by the same distance on a log scale. That is, the distance between 0.3 and 3 is the same as the distance between 30 and 300. This separation is referred to as a decade or cycle.

The plot shown in pervious Figure covers three decades in x (from 0.1 to 100) and four decades in y and is thus called a four-by-three-cycle plot.

Specialized plot commands.

Command	Description
`bar(x,y)`	Creates a bar chart of y versus x.
`plotyy(x1,y1,x2,y2)`	Produces a plot with two *y*-axes, y1 on the left and y2 on the right.
`polar(theta,r,'type')`	Produces a polar plot from the polar coordinates theta and r, using the line type, data marker, and colors specified in the string type.
`stairs(x,y)`	Produces a stairs plot of y versus x.
`stem(x,y)`	Produces a stem plot of y versus x.

Interactive Plotting in MATLAB

This interface can be advantageous in situations where:

- **You need to create a large number of different types of plots.**
- **You must construct plots involving many data sets.**
- **You want to add annotations such as rectangles and ellipses.**
- **You want to change plot characteristics such as tick spacing fonts, bolding, *italics*, and colors.**
- **Creating different types of graphs.**
- **Selecting variables to plot directly from the Workspace Browser.**
- **Creating and editing subplots,**
- **Adding annotations such as lines, arrows, text, rectangles, and ellipses.**
- **Editing properties of graphics objects, such as their colors, line weight, and font.**

Three-Dimensional Plot → **Line Plots**

→ **Surface Mesh Plots**

1. Line Plots

Lines in three-dimensional space can be plotted with the `plot3` function.

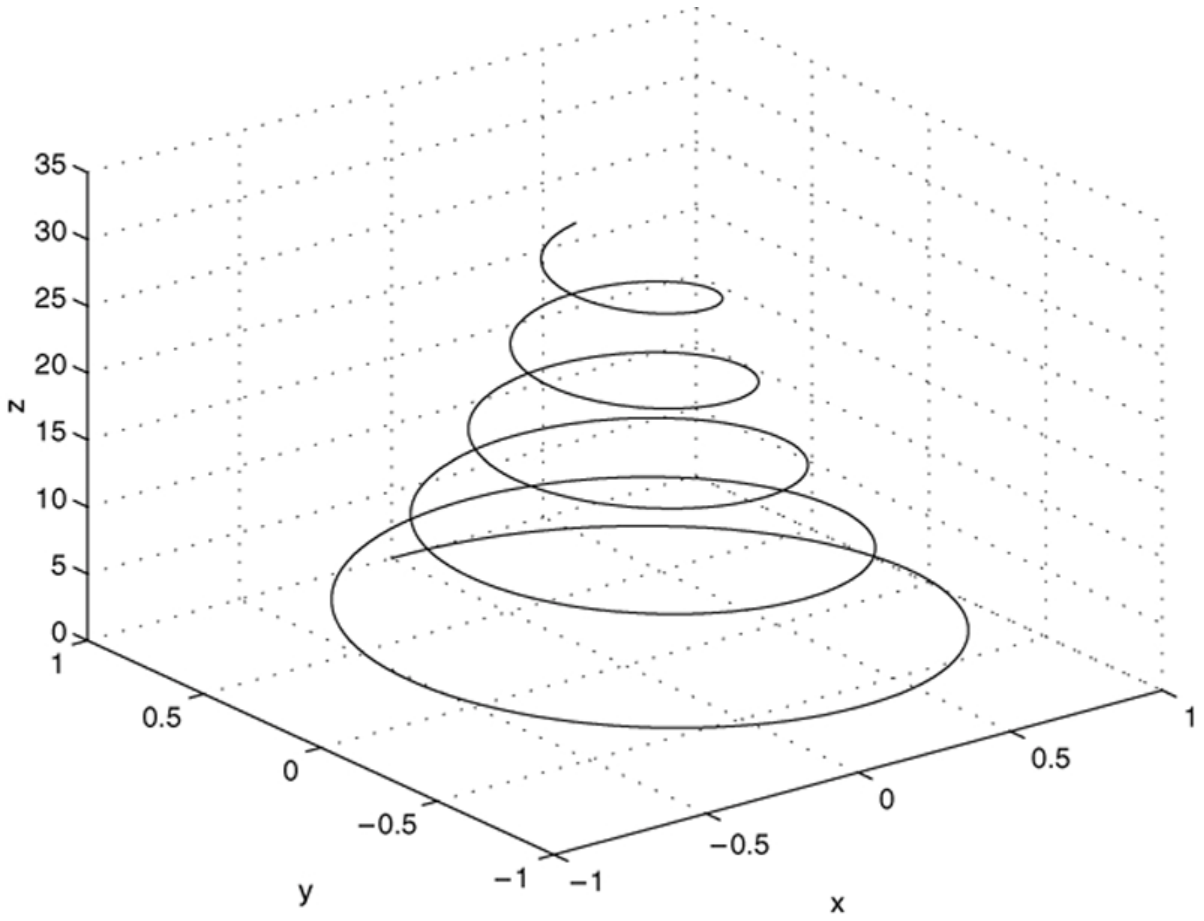
Its syntax is:

```
plot3(x,y,z)
```

For example; the following equations generate a three-dimensional curves as the parameter t is varied over some range: $x = e^{-0.05t}\sin(t)$, $y = e^{-0.05t}\cos(t)$, $z = t$, $0 \le t \le 10\pi$

The following program uses the `plot3` function to generate the spiral curve

```
t = [0:pi/50:10*pi];
plot3(exp(-0.05*t).*sin(t),exp(0.05*t).*cos(t),t)
xlabel('x'),ylabel('y'),zlabel('z'),grid
```



2. Surface Mesh Plots $f(x,y)$

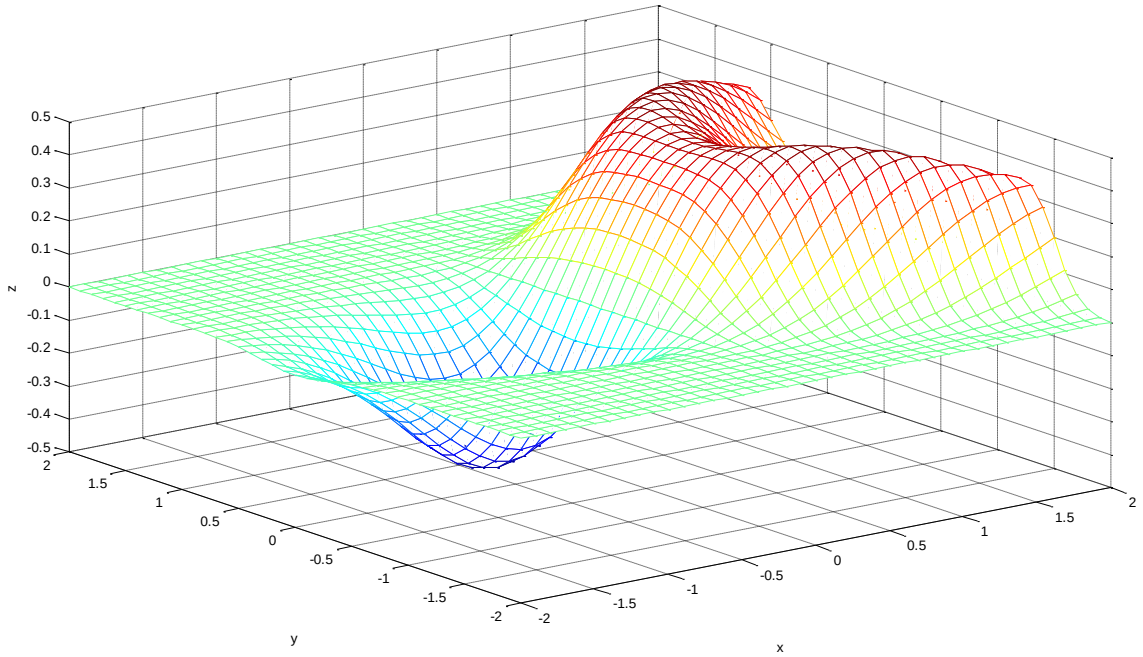
Mesh and surface plots are three-dimensional plots that are used for plotting function of the form $z= f(x,y)$ where the x and y are the independent variables and z is the dependent variable. It means that within a given domain the value of z can be calculated for any combination of x and y.

Mesh and surface plots are created in three steps.

1. **Create a grid in the x-y plane that covers the domain of the function.**
2. **Calculate the value of z at each point of the grid.**
3. **Create the plot.**

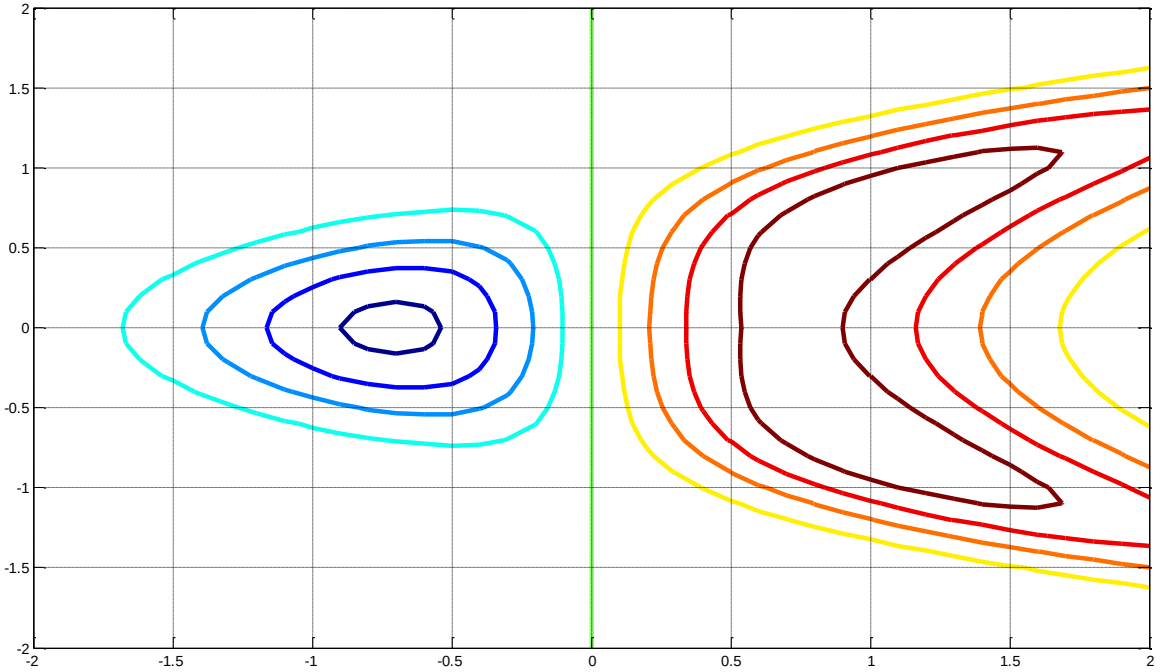
For example; the following equations generate a three-dimensional plot $z = xe^{-[(x-y^2)^2+y^2]}$, for $-2 \le x \le 2$ and $-2 \le y \le 2$, with a spacing of 0.1.

```
[X,Y] = meshgrid(-2:0.1:2);
Z = X.*exp(-((X-Y.^2).^2+Y.^2));
mesh(X,Y,Z),xlabel('x'),ylabel('y'),...
zlabel('z')
```



The following session generates the contour plot of the function whose surface plot is shown in Figure; namely, $z = xe^{-[(x-y^2)^2+y^2]}$, for $-2 \le x \le 2$ and $-2 \le y \le 2$, with a spacing of 0.1.

```
[X,Y] = meshgrid(-2:0.1:2);
Z = X.*exp(-((X- Y.^2).^2+Y.^2));
contour(X,Y,Z),xlabel('x'),ylabel('y')
```



Three-dimensional plotting functions.

Function	Description
`contour(x,y,z)`	Creates a contour plot.
`mesh(x,y,z)`	Creates a 3D mesh surface plot.
`meshc(x,y,z)`	Same as mesh but draws contours under the surface.
`meshz(x,y,z)`	Same as mesh but draws vertical reference lines under the surface.
`surf(x,y,z)`	Creates a shaded 3D mesh surface plot.
`surfc(x,y,z)`	Same as surf but draws contours under the surface.
`[X,Y]=meshgrid(x,y)`	Creates the matrices X and Y from the vectors x and y to define a rectangular grid.
`[X,Y]=meshgrid(x)`	Same as `[X,Y]= meshgrid(x,x)`.
`waterfall(x,y,z)`	Same as mesh but draws mesh lines in one direction only.

Plots of the surface $z = xe^{-(x^2+y^2)}$ created with the mesh function and its variant forms: meshc, meshz, and waterfall.

a) mesh, b) meshc, c) meshz, d) waterfall.