Lab. (1&2)
MATLAB

Answer the following questions

Matrices

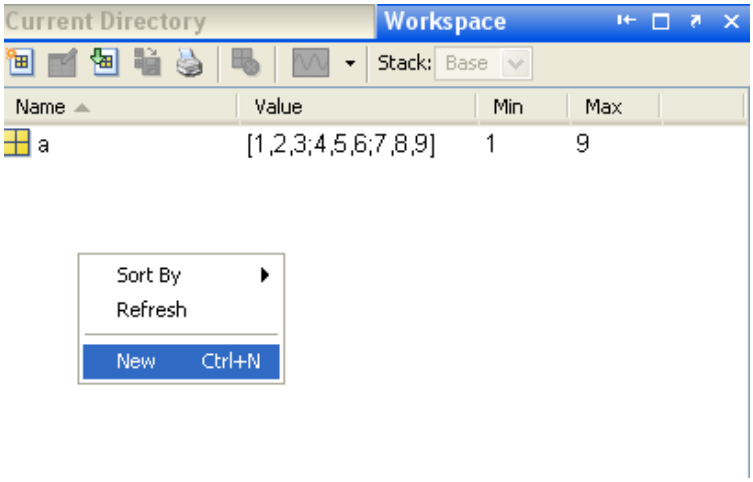
How to enter a matrix into Matlab:

```
>> a=[1 2 3
4 5 6
7 8 9]

a =
   1   2   3
   4   5   6
   7   8   9
```

Or:

```
>> a=[1 2 3;4 5 6;7 8 9]

a =

   1   2   3
   4   5   6
   7   8   9

>> a=[1,2,3;4,5,6;7,8,9]
a =

   1   2   3
   4   5   6
   7   8   9
```

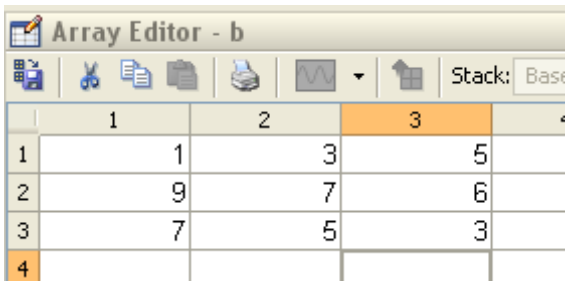
Or: create it directly in workspace



Name the workspace variable b

Name	Value	Min	Max
a	[1,2,3;4,5,6;7,8,9]	1	9
b	[1,3,5;9,7,6;7,5,3]	1	9

Write its rows and columns using array editor



When you call it using command prompt, it will display its values

```
>> b

b =

     1     3     5
     9     7     6
     7     5     3
```

Matrix transpose:

If you want to rotate the matrix (change rows and columns)

```
>> a

a =

   1   2   3
   4   5   6
   7   8   9
>> a'
ans =

   1   4   7
   2   5   8
   3   6   9
```

Diagonal of the matrix:

```
>> diag(a)
```

```
ans =
   1
   5
   9
```

Subscripts; or reading matrix elements:

1. If you want to read an element from a matrix, the first row and second column

```
>> a(2,1)

ans =

   4
```

2. The element of second row and third column

```
>> a(2,3)

ans =

   6
```

3. To find total row; write the row number, and colon for all columns

```
>> a(2,:)

ans =

   4    5    6
```

4. To find complete column; write colon for all rows and the column number.

```
>> a(:,3)

ans =

   3
   6
   9
```

Sum command:

This command computes the sum of each column

```
>> sum(a)

ans =
```

```
  12   15   18
```

To compute the sum of each row

```
>> sum(a')

ans =

   6   15   24
```

You can write it in a form clear to understand, you can transpose it

```
>> sum(a')'

ans =

   6
  15
  24
```

You can sum the diagonal element of a matrix

```
>> sum(diag(a))

ans =

  15
```

The other diagonal, the so-called antidiagonal, is not so important mathematically, so MATLAB does not have a ready-made function for it. But a function originally intended for use in graphics, fliplr, flips a matrix from left to right, so the sum of anti-diagonal

```
>> sum(diag(fliplr(a)))

ans =

  15
```

You can make a sum for any part of a matrix; summing all numbers in last column

```
>> sum(a(:,end))

ans =

  18
```

for summing the elements of third row from the second column to the third column

```
>> sum(a(3,2:3))
```

```
ans =

   17
```

You can make any mathematical operation between any number of elements in matrix

```
>> a(1,2) + a(3,3)

ans =

   11

>> a(1,2)^a(1,3)

ans =

    8

>> sqrt(a(1,2))

ans =

   1.4142
```

Magic matrix:

MATLAB actually has a built-in function that creates magic squares of almost any size.

```
>> magic(3)

ans =

   8   1   6
   3   5   7
   4   9   2
```

Rearranging matrix:

For a predefined matrix **a**, we will change two rows, then two columns.

```
>> a(:,[1 3 2])
ans =

   1   3   2
   4   6   5
   7   9   8


>> a([1 3 2],:)
```

```
ans =

   1   2   3
   7   8   9
   4   5   6
```

Zero matrix:

Make a zero matrix of any size

```
>> zeros(3)

ans =

   0   0   0
   0   0   0
   0   0   0

>> zeros(2,4)
ans =

   0   0   0   0
   0   0   0   0
```

Ones matrix:

To create a matrix of ones of any size or another number.

```
>> ones(3)

ans =

   1   1   1
   1   1   1
   1   1   1

>> ones(3,2)
ans =

   1   1
   1   1
   1   1

>> 3*ones(3)

ans =
```

```
   3    3    3
   3    3    3
   3    3    3
```

Random no:

To create a random numbers matrix

```
>> rand(3)

ans =

   0.8147    0.9134    0.2785
   0.9058    0.6324    0.5469
   0.1270    0.0975    0.9575

>> rand(3,2)
ans =

   0.9649    0.9572
   0.1576    0.4854
   0.9706    0.8003
```

Concatenation:

To make a combined matrix

```
>> a=[1 2 3;4 5 6;7 8 9]

a =

   1    2    3
   4    5    6
   7    8    9

>> b=[a a+5; a-1 a+2]
b =

   1    2    3    6    7    8
   4    5    6    9   10   11
   7    8    9   12   13   14
   0    1    2    3    4    5
   3    4    5    6    7    8
   6    7    8    9   10   11
```

Deleting elements:

```
>> a(:,1)=[]

a =

   2   3
   5   6
   8   9
>> a(2,:)=[]

a =

   2   3
   8   9
```

Determinant and inverse of matrix:

```
>> a

a =

   8   7   2
   4   5   6
   7   8   9

>> det(a)

ans =

   12

>> inv(a)

ans =

   -0.25   -3.92    2.67
    0.50    4.83   -3.33
   -0.25   -1.25    1.00
```

Eigenvalues:

For last modified a matrix

```
>> eig(a)
```

ans =

18.39
3.42
0.19

Unity matrix:

```
>> eye(3)

ans =

     1.00        0        0
        0     1.00        0
        0        0     1.00
```

2.4 Summary of Commands

Command	Description
det	Computes the determinant of a matrix
inv	Computes the inverse of a matrix
eig	Computes the eigenvalues of any matrix
eye	Generates an identity matrix
Rand	Generates a matrix of random numbers between 0 and 1
Ones	Generates a matrix of ones
zeros	Generates a matrix of zeros
Magic	Generates the magic matrix
sum	Sum of array elements (matrix row, column, or diagonal)
diag	Computes the diagonal of the matrix
‘	Find the matrix transpose
rank	Computes rank of a matrix

2.5 Statistic

```
>> a=[1:10]

a =

 Columns 1 through 7

     1.00     2.00     3.00     4.00     5.00     6.00     7.00

 Columns 8 through 10

     8.00     9.00    10.00
```

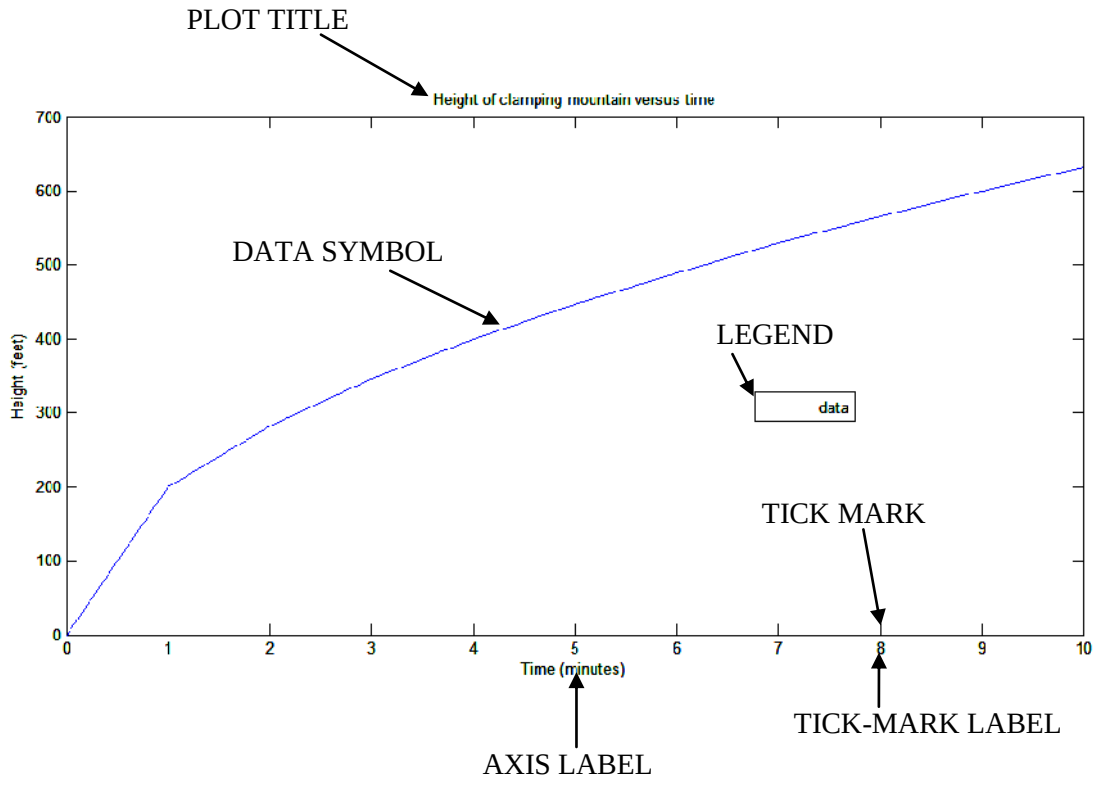
```
>> min(a)
ans =
        1.00
>> max(a)
ans =
       10.00
>> mean(a)
ans =
        5.50
>> std(a)
ans =
        3.03
```

Plotting

Contents:

The most common plot is the xy plot. Its name assumes that we are plotting a function y = f(x). We plot the x values on the horizontal axis (the abscissa) and the y values on the vertical axis (the ordinate). Usually we plot the independent variable, which is the one more easily varied, on the abscissa, and the dependent variable on the ordinate.

3.1 Anatomy of a Plot

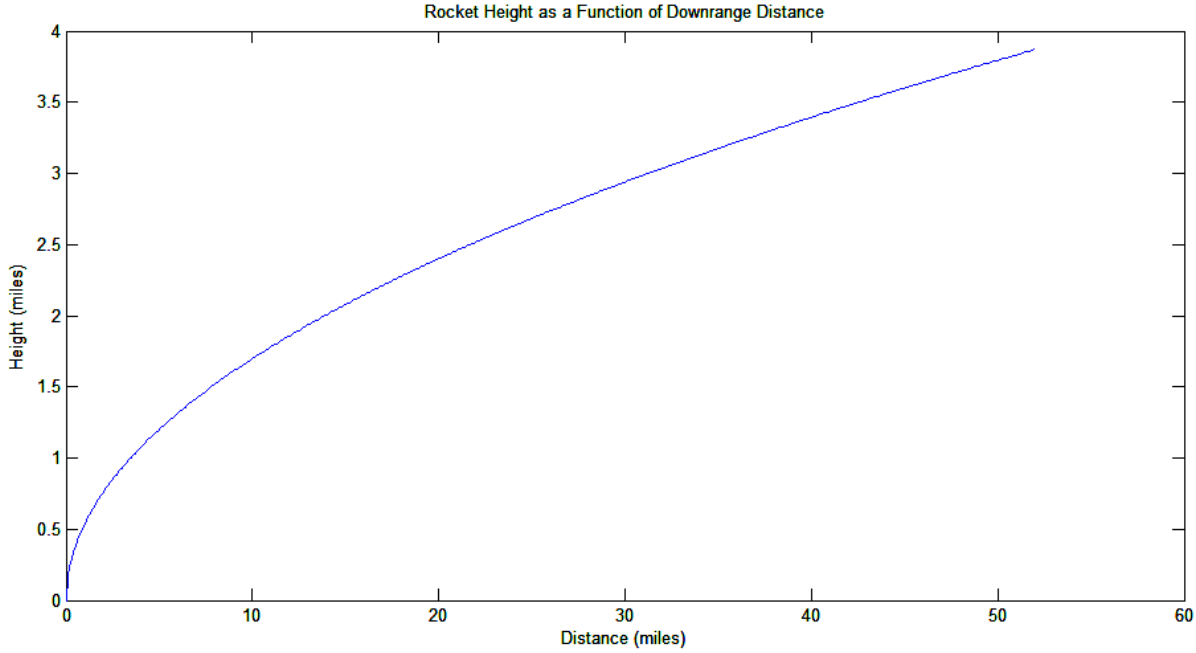


Example:

The following MATLAB session plots $y = 0.4\sqrt{1.8x}$ for $0 \leq x \leq 52$, 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'), grid on, axis ([0 52 0 5])
```



3.1.1 Axis command

This command sets the scaling for the x- and y- axes to the minimum and maximum values indicated.

Syntax:

```
axis ([xmin xmax ymin ymax])
```

The axis command variants:

```
axis square
```

which selects the axes' limits so that the plot will be square

```
axis equal
```

which select the scale factors and tick spacing to be the same on each axis.

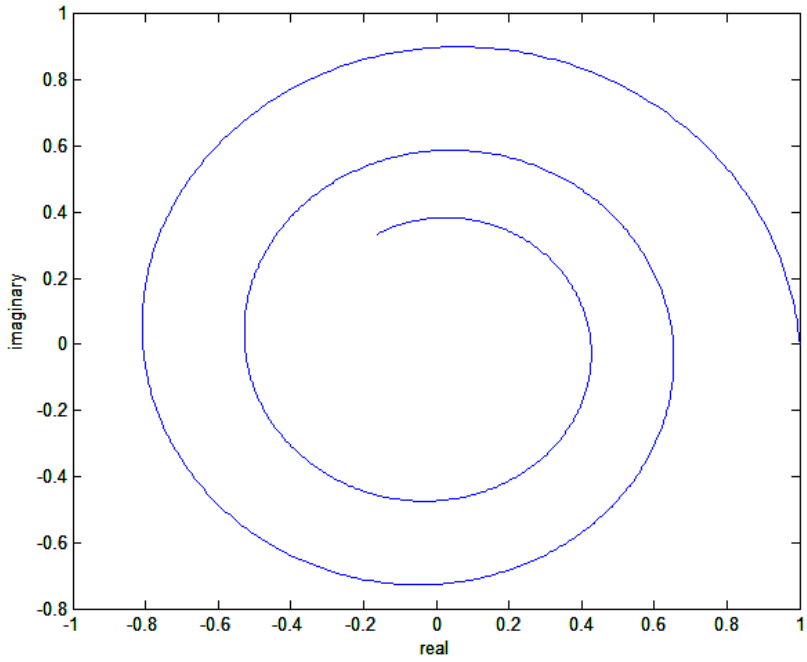
```
axis auto
```

which returns the axis scaling to its default autoscaling mode in which the best axis limits are computed automatically.

3.1.2 Plots of complex numbers

```
>> z=0.1+0.9i;
>> n=[0:0.01:10];
>> plot(z.^n), xlabel('real'), ylabel('imaginary')
```

The resulting plot will be as shown below



The last command can be written as below, and it will have the same result

```
>> x=z.^n;
>> plot(real(x),imag(x)), xlabel('real'), ylabel('imaginary')
```

3.1.3 The function plot command

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.

Syntax:

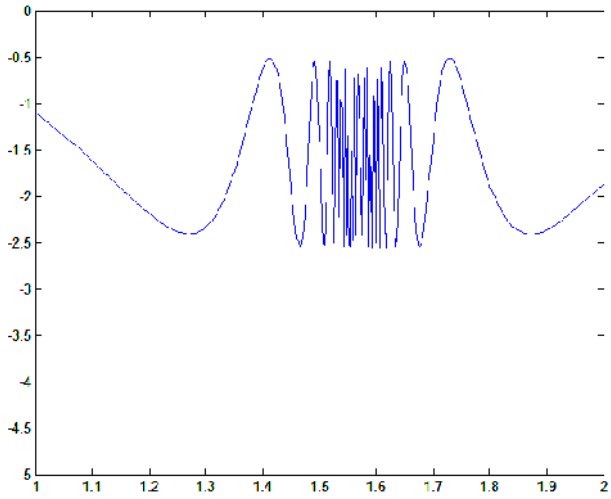
```
fplot('sting', [xmin xmax])
```

where 'string' is a text string that describes the function to be plotted
and [xmin xmax] specifies the minimum and maximum values of the independent variable.
The range of the dependent variable can also be specified in this case by the syntax as

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

Example:

```
>> y='cos(tan(x))-tan(sin(x))';
>> fplot(y,[1 2 -5 0])
```



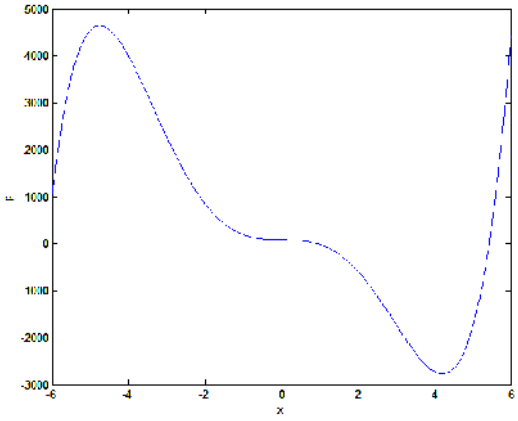
3.2 Plotting Polynomials

We can plot polynomials more easily by using the `polyval` function.

Example:

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.

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



3.3 Subplots

You can use the `subplot` command to obtain several smaller "subplots" in the same figure.

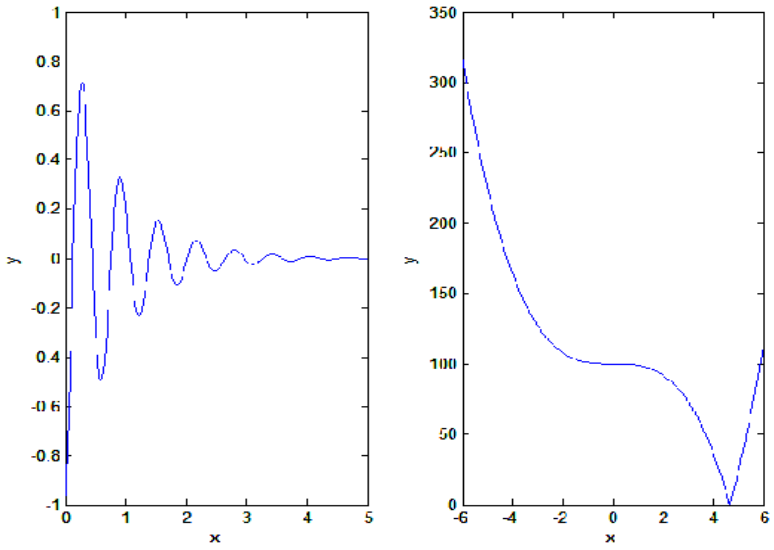
Syntax:

```
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 `subplot` command into the *p*th pane.

Example: Plot 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),xlabel('x'),ylabel('y'),axis([0 5 -1 1])
>> x=[-6:0.01:6];
>> y=abs(x.^3-100);
>> subplot(1,2,2)
>> plot(x,y),xlabel('x'),ylabel('y'),axis([-6 6 0 350])
```



3.4 Data Markers and Line Types

To plot the vector y versus the vector x and mark each point with a data marker, enclose the symbol for the marker in single quotes in the `plot` function.

Syntax:

```
plot(x,y,'o')
```

This notation makes the plot appears with circle data markers.

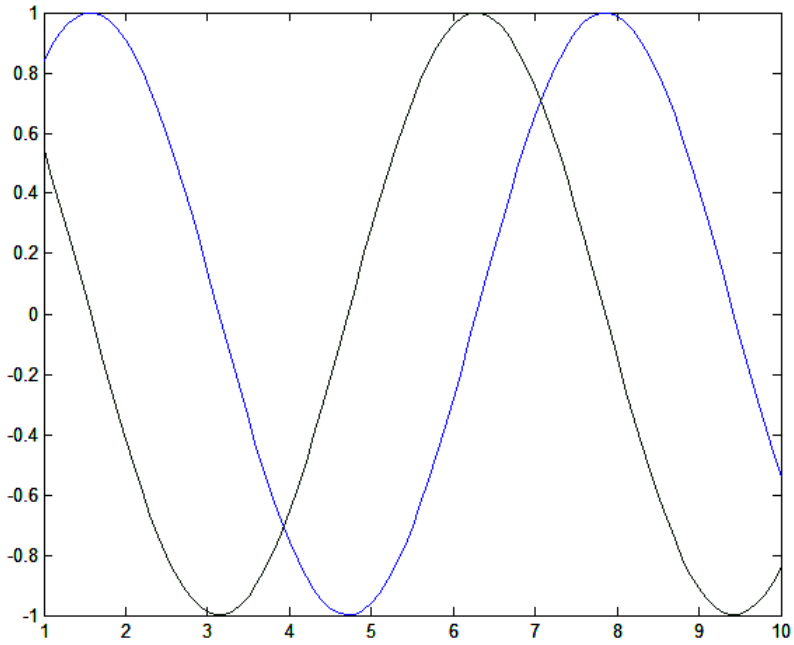
To connect each data marker with a straight line, we must plot the data twice by typing

```
plot(x,y,x,y,'o')
```

Example:

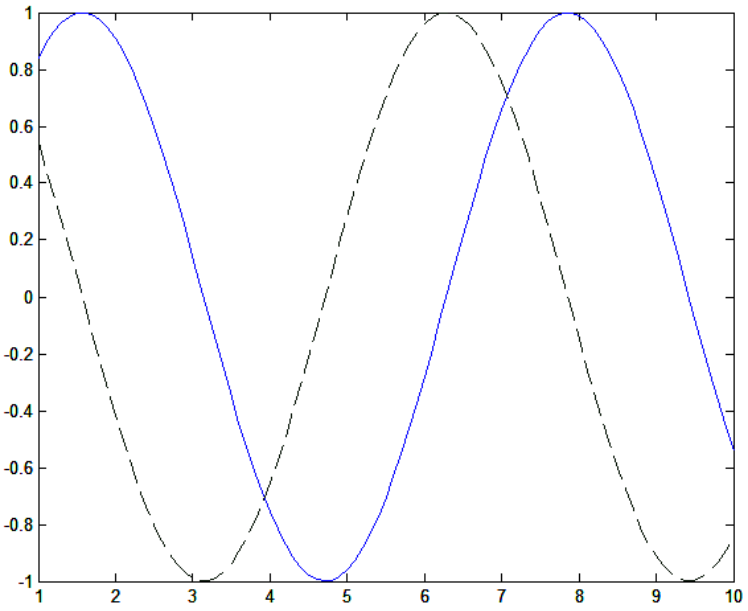
```
>> x=[1:0.1:10];
>> y=sin(x);
>> u=[1:0.1:10];
>> v=cos(x);
>> plot(x,y,u,v)
```

The resulting curves are as shown below.



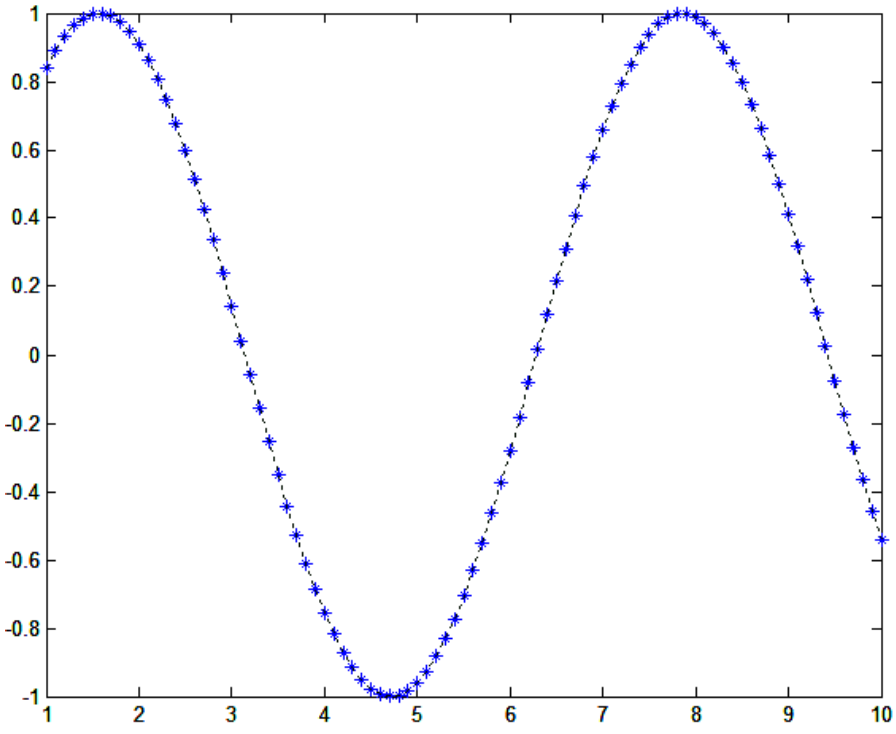
You cannot distinguish between the two curves without colors, so you can change the line type as shown below.

```
>> plot(x,y,u,v,'--')
```



To plot y versus x with asterisks (*) connected with a dotted line, you can do the following.

```
>> plot(x,y,'*',x,y,':')
```



You can obtain symbols and lines of different colors by combining the color symbol with the data marker symbol and the line type symbol.

```
>> plot(x,y,'b*',x,y,'r:')
```

Legend command:

Allows you to add a legend to your plot to distinguish between curves.

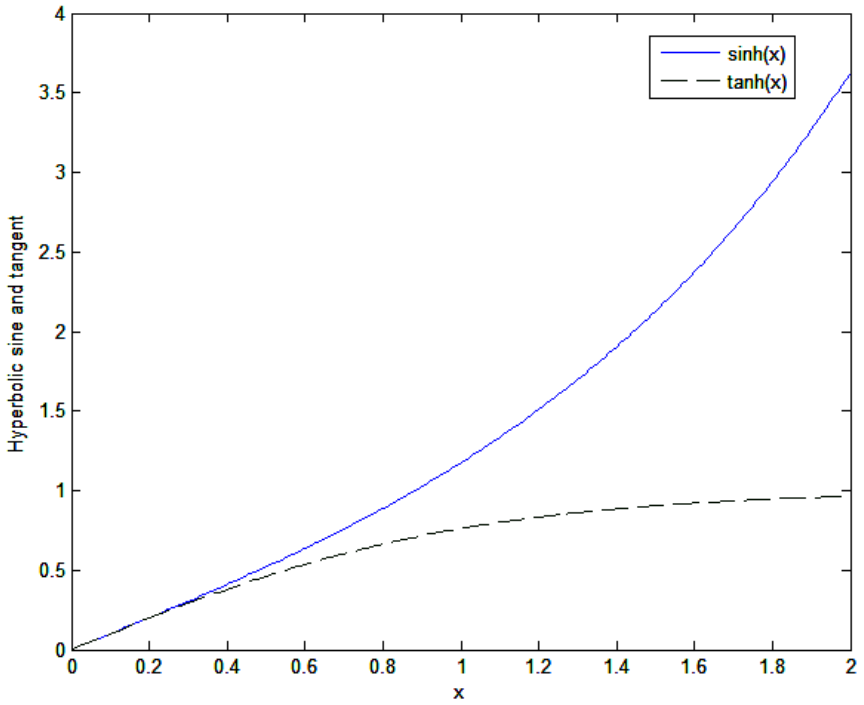
Syntax:

```
legend('string1','string2')
```

where `string1` and `string2` are text strings of your choice. 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.

Example:

```
>> 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)')
```



gtext and text commands:

Syntax:

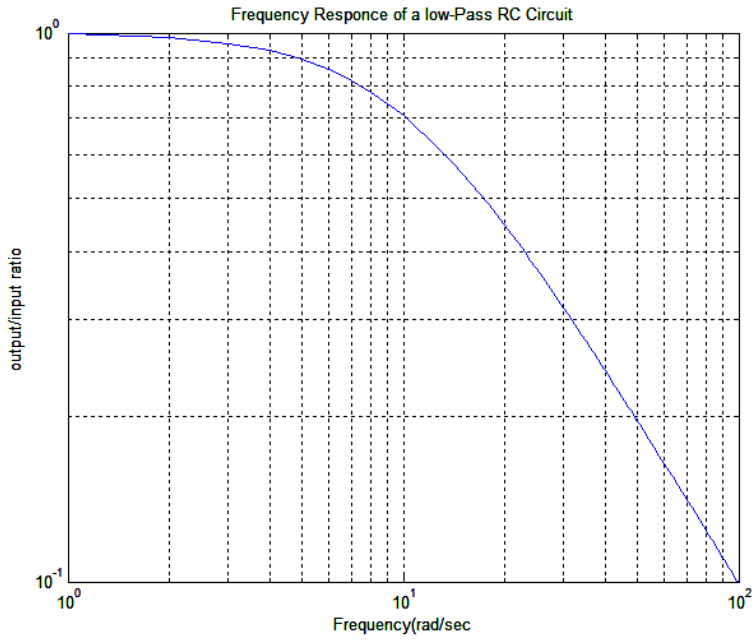
```
gtext('string')
```

Allows you to add a text label and place it using the mouse.

`text(x,y,'string')`
Allows you to add a text string to the plot at the location specified by the coordinates x,y.

3.5 Plotting on log-log scale

```
>> s=[1:100]*i;
>> M=abs(1./(RC*s+1));
>> loglog(imag(s),M), grid,xlabel('Frequency(rad/sec'),…
   ylabel('output/input ratio'),…
   title('Frequency Responce of a low-Pass RC Circuit')
```

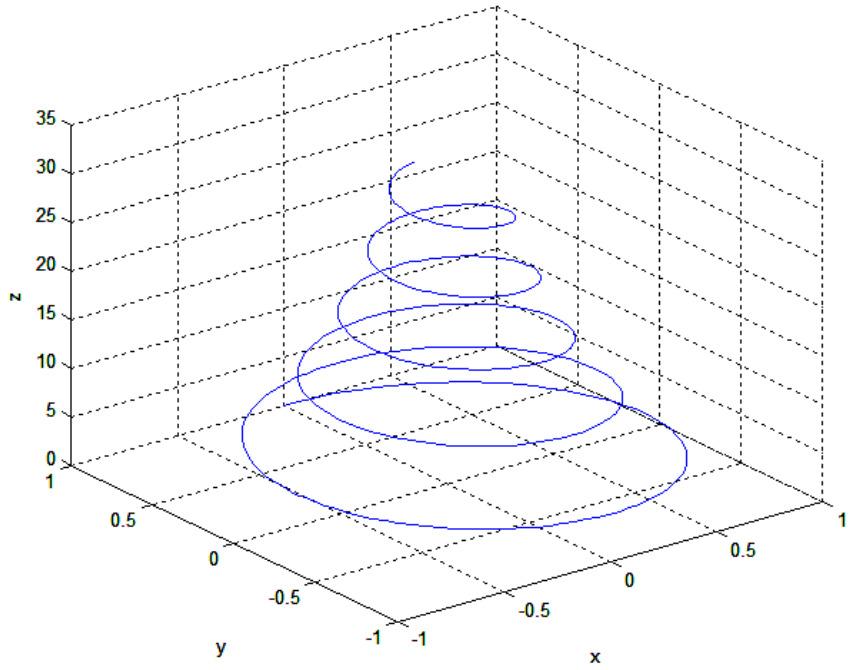


3.6 Three-Dimensional Plots

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 curve as the parameter t is varied over some range (t = 0 to t = 10π) :

$$x = e^{-0.05\,t}\sin(t)\ , y = e^{-0.05\,t}\cos(t)\ \ , z = t$$

```
>> 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
```



3.6.1 Mesh and Surface Plots

MATLAB defines a surface by the z-coordinates of points above a grid in the x-y plane, using straight lines to connect adjacent points. The `mesh` and `surf` plotting functions display surfaces in three dimensions. `mesh` produces wireframe surfaces that color only the lines connecting the defining points. `surf` displays both the connecting lines and the faces of the surface in color.

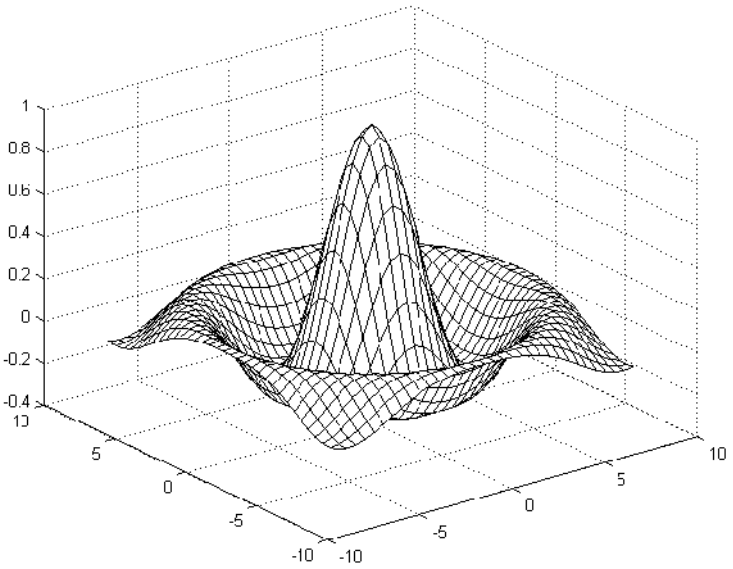
Visualizing Functions of Two Variables

To display a function of two variables, z = f (x,y):

- Generate X and Y matrices consisting of repeated rows and columns, respectively, over the domain of the function.
- Use X and Y to evaluate and graph the function.

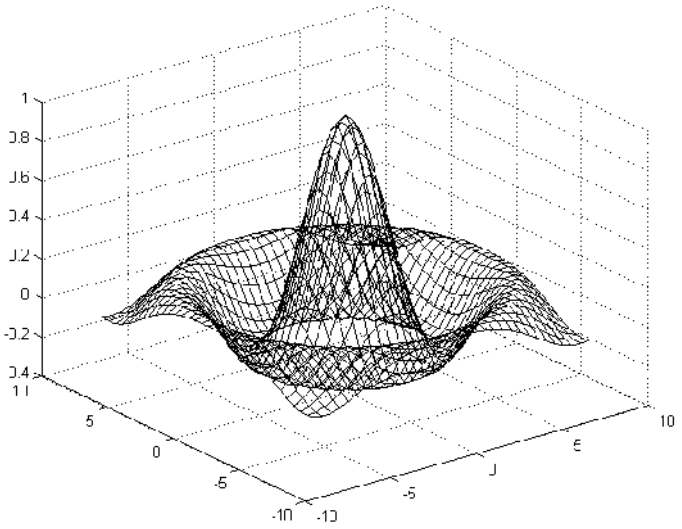
The `meshgrid` function transforms the domain specified by a single vector or two vectors x and y into matrices X and Y for use in evaluating functions of two variables. The rows of X are copies of the vector x and the columns of Y are copies of the vector y.

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



You can create a transparent mesh by disabling hidden line removal.

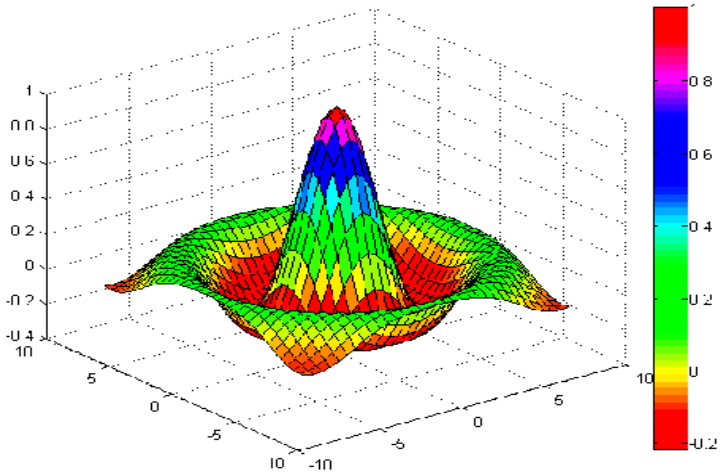
```
>> hidden off
```



3.6.2 Colored Surface Plots

A surface plot is similar to a mesh plot except the rectangular faces of the surface are colored. The color of the faces is determined by the values of Z and the `colormap` (a `colormap` is an ordered list of colors). These statements graph the `sinc` function as a surface plot, select a `colormap`, and add a `color bar` to show the mapping of data to color.

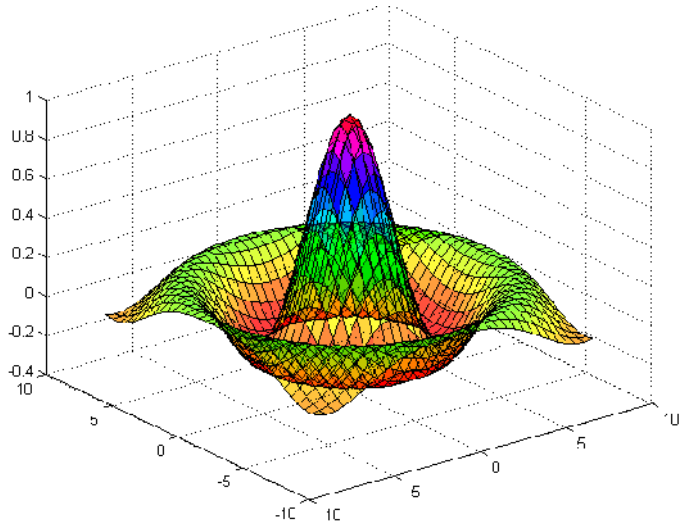
```
>> surf(X,Y,Z)
>> colormap hsv
>> colorbar
```



3.6.3 Transparent Surfaces

You can make the faces of a surface transparent to a varying degree. Transparency (referred to as the `alpha` value) can be specified for the whole object or can be based on an `alphamap`, which behaves in a way analogous to colormaps. For example,

```
>> surf(X,Y,Z)
>> colormap hsv
>> alpha(.4)
```



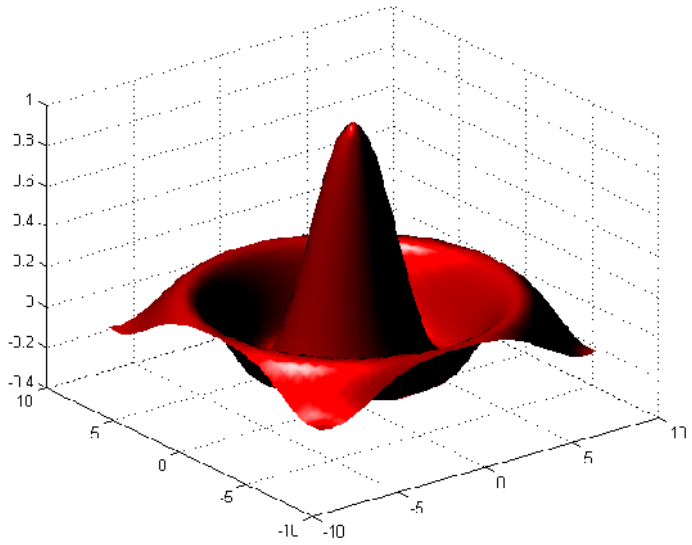
produces a surface with a face alpha value of 0.4. Alpha values range from 0 (completely transparent) to 1 (not transparent).

3.6.4 Surface Plots with Lighting

Lighting is the technique of illuminating an object with a directional light source. In certain cases, this technique can make subtle differences in surface shape easier to see. Lighting can also be used to add realism to three-dimensional graphs. This example uses the same surface as the previous examples, but colors it red and removes the mesh lines. A light object is then

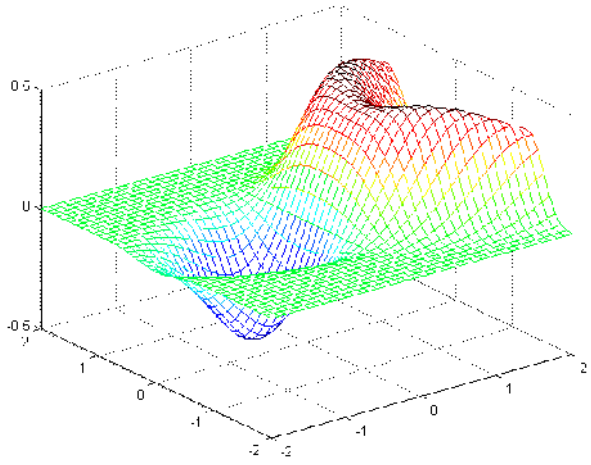
added to the left of the “camera” (that is the location in space from where you are viewing the surface).

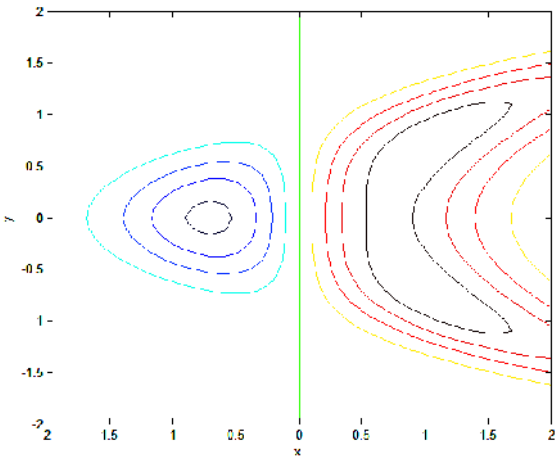
```
>> surf(X,Y,Z,'FaceColor','red','EdgeColor','none')
>> camlight left; lighting phong
```



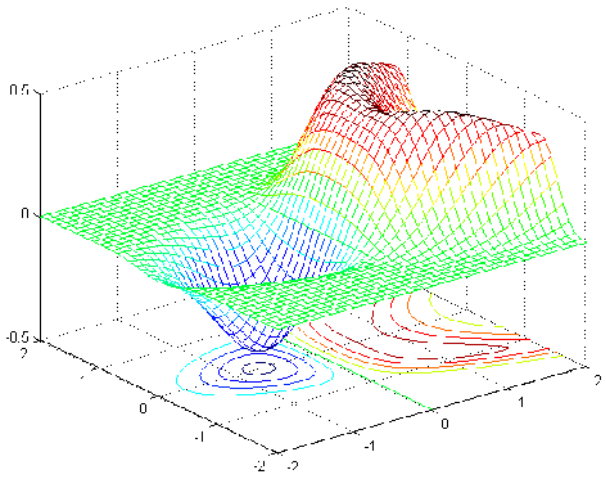
MATLAB does not remove the existing graph; it adds the new data to the current graph, rescaling if necessary. For example, these statements first create a contour plot of the peaks function, then superimpose a pseudo color plot of the same function:

```
>> z=x.*exp(-((x-y.^2).^2+y.^2));
>> mesh(x,y,z)
>> contour(x,y,z), xlabel('x'), ylabel('y')
```



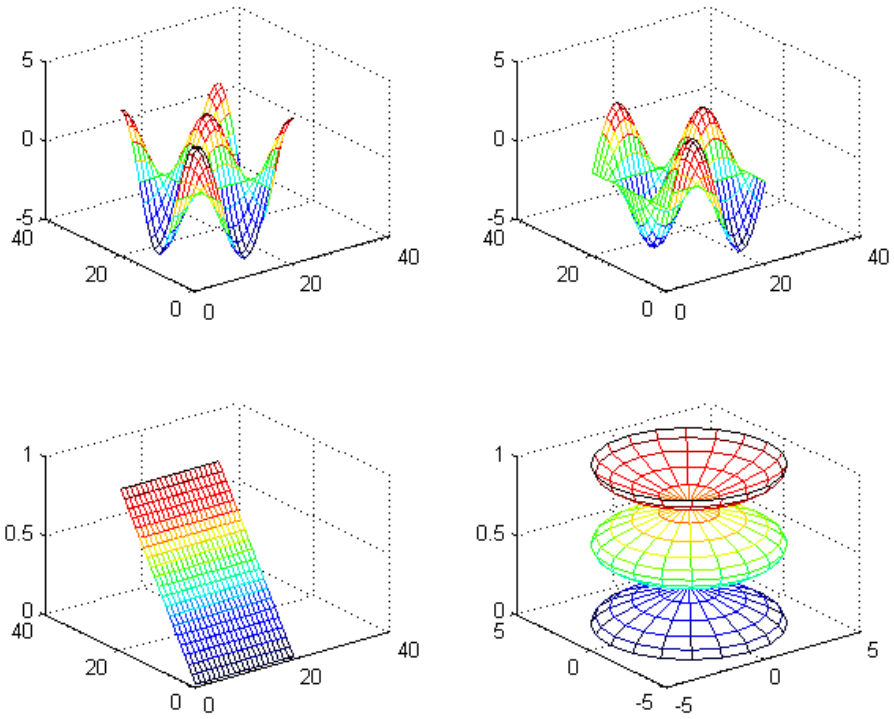


```
>> meshc(x,y,z)
```



Example:

```
>> t = 0:pi/10:2*pi;
>> [X,Y,Z] = cylinder(4*cos(t));
>> subplot(2,2,1)
>> mesh(X)
>> subplot(2,2,2); mesh(Y)
>> subplot(2,2,3); mesh(Z)
>> subplot(2,2,4); mesh(X,Y,Z)
```



3.7 Plotting Commands Summary

Basic xy Plotting Commands

Command	Description
axis	Sets axis limits
fplot	Intelligent plotting of functions
grid	Displays gridlines
plot	Generates xy plot
print	Prints plot or saves plot to a file
title	Puts text at top of plot
xlabel	Adds text label to x-axis
ylabel	Adds text label to y-axis
ginput	Reads coordinates of the cursor position

Plot Enhancement Commands

Command	Description
axes	Creates axes objects
close	Closes the current plot
close all	Closes all plots
figure	Opens a new figure window
gtext	Enables label placement by mouse
hold	Freezes current plot
legend	Legend placement by mouse
refresh	Redraws current figure window
set	Specifies properties of objects such as axes
subplot	Creates plots in subwindows
text	Places string in figure

Specialized Plot Commands

Command	Description
bar	Creates bar chart
loglog	Creates log-log plot
polar	Creates polar plot
semilogx	Creates semilog plot (logarithmic abscissa)
semilogy	Creates semilog plot (logarithmic ordinate)
stairs	Creates stairs plot
stem	Creates stem plot

Colors, Symbols and Line Types

	Command	Description
Color	y	yellow
	m	magenta
	c	cyan
	r	red
	g	green
	b	blue
	w	white
	k	Black
Symbol	.	point
	o	circle
	x	x-mark (cross)
	+	plus
	*	star
	d	diamond
	v	triangle (down)
	^	triangle (up)
	<	triangle (left)
	>	triangle (right)
	p	pentagram (five-pointed star)
	h	Hexagram
	s	square
Line	-	solid
	:	dotted
	-.	dash dotted
	--	dashed

Three-Dimensional Plotting Commands

Command	Description
contour	Creates contour plot
mesh	Creates three-dimensional mesh surface plot
meshc	Same as mesh with contour plot underneath
meshz	Same as mesh with vertical lines underneath
plot3	Creates three-dimensional plots from lines and points
surf	Creates shaded three-dimensional mesh surface plot
surfc	Same as surf with contour plot underneath
meshgrid	Creates rectangular grid
waterfall	Same as mesh with mesh lines in one direction
zlabel	Adds text label to z-axis

Histogram Functions

Command	Description
bar	Creates a bar chart
hist	Aggregates the data into equally spaced bins
histc	Aggregates the data into unequally spaced bins