# MATLAB

*for Engineering Applications*

**Fifth Edition**

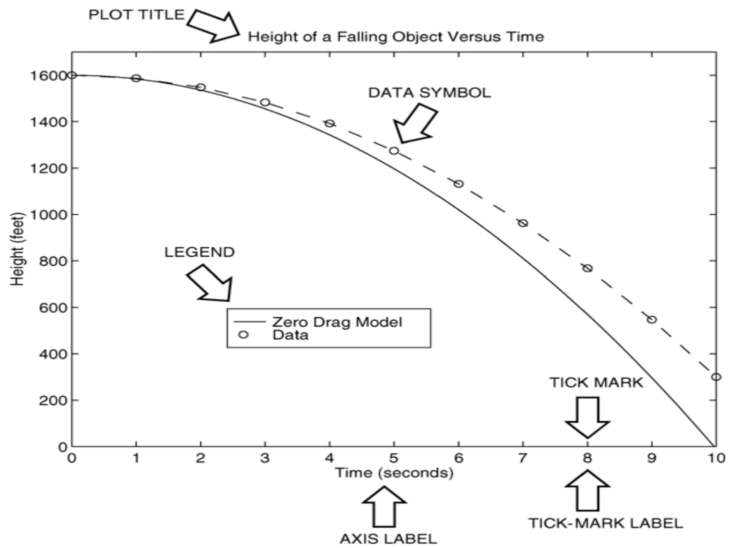
William J. Palm III



# Chapter 05

## Advanced Plotting

## Nomenclature for a Typical *xy* Plot



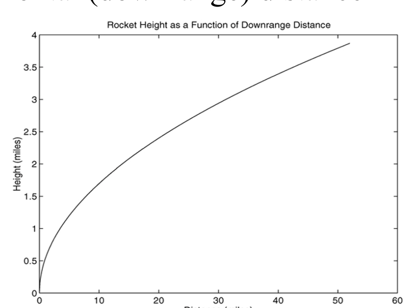

 

## Generating a Plot

**An 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.

The Autoscaling Feature in MATLAB

Selects Tick-Mark Spacing




```
>>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')
```

## Obtaining a Hard Copy of the Plot

The plot will appear in the Figure window. You can obtain a hard copy of the plot in several ways:

1. Use the menu system. Select **Print** on the File menu in the Figure window. Answer **OK** when you are prompted to continue the printing process.
2. Type `print` at the command line. This command sends the current plot directly to the printer.
3. Save the plot to a file to be printed later or imported into another application such as a word processor. You need to know something about graphics file formats to use this file properly. See the subsection **Exporting Figures**.




## Closing the Figure Window

When you are finished with the plot, close the figure window by selecting **Close** from the File menu in the Figure window.

Note that using the **Alt-Tab** key combination in Windows-based systems will return you to the Command window without closing the figure window.

If you do not close the window, it will not reappear when a new `plot` command is executed. However, the figure will still be updated.

## Requirements for a Correct Plot 1

The following list describes the essential features of any plot:

1. Each axis must be labeled with the name of the quantity being plotted *and its units!* If two or more quantities having different units are plotted (such as when plotting both speed and distance versus time), indicate the units in the axis label if there is room, or in the legend or labels for each curve.
2. Each axis should have regularly spaced tick marks at convenient intervals—not too sparse, but not too dense—with a spacing that is easy to interpret and interpolate. For example, use 0.1, 0.2, and so on, rather than 0.13, 0.26, and so on.
3. If you are plotting more than one curve or data set, label each on its plot or use a legend to distinguish them.
4. If you are preparing multiple plots of a similar type or if the axes' labels cannot convey enough information, use a title.
5. If you are plotting measured data, plot each data point with a symbol such as a circle, square, or cross (use the same symbol for every point in the same data set). If there are many data points, plot them using the dot symbol.

## Requirements for a Correct Plot 2

6. Sometimes data symbols are connected by lines to help the viewer visualize the data, especially if there are few data points. However, connecting the data points, especially with a solid line, might be interpreted to imply knowledge of what occurs between the data points. Thus you should be careful to prevent such misinterpretation.
7. If you are plotting points generated by evaluating a function (as opposed to measured data), do *not* use a symbol to plot the points. Instead, be sure to generate many points, and connect the points with solid lines.

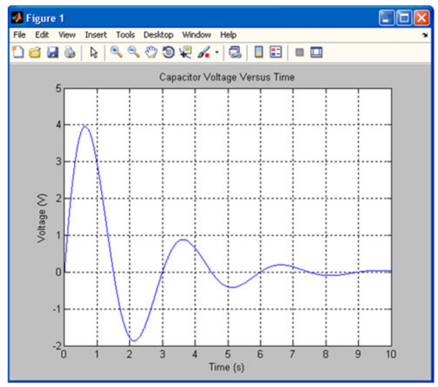
 

# The grid and axis Commands

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.

Example of a Figure Window







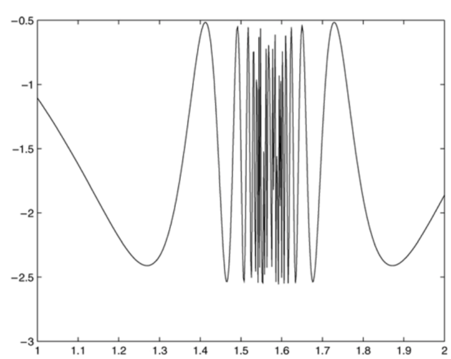
# The `fplot` Function [1]

The `fplot` function chooses a small enough spacing to display the function’s full behavior.

% The Code

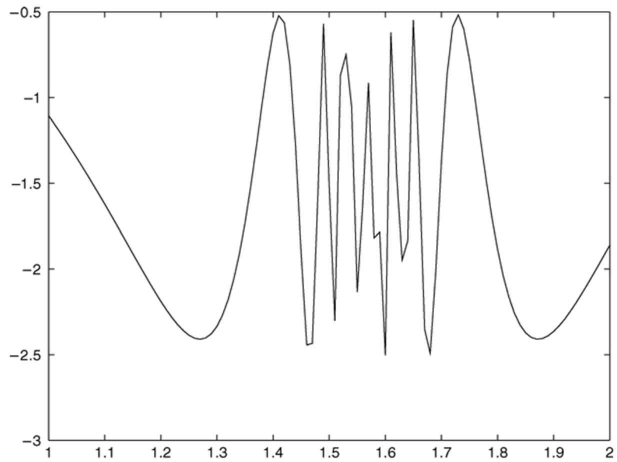
f = @(x) (cos(tan(x)) − tan(sin(x)));

fplot(f,[1 2])



## The `fplot` Function [2]

The same function plotted with the `plot` command. A spacing of 0.01 misses some oscillations.

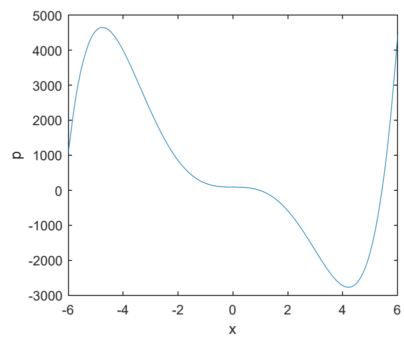





**Plotting polynomials with the polyval function:**

To plot the polynomial $3x^5 + 2x^4 - 100x^3 + 2x^2 - 7x + 90$ over the range $-6 \le x \le 6$ 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')
```

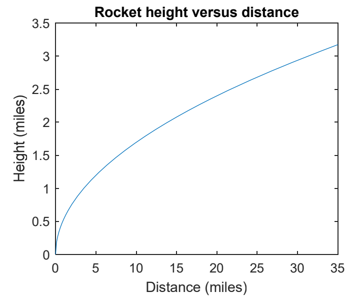


## Question 1

Q1- Plot the equation $y = 0.4\sqrt{1.8x}\ for\ 0 \le x \le 35$ $and\ 0 \le y \le 3.5$.

```
Title = Rocket height versus distance
X axis label= Distance (miles)
Y axis label=Height (miles)
```

```
>> x=0:0.1:35;
>> y=0.4*sqrt(1.8*x);
>> plot(x,y),xlabel('Distance (miles)'),ylabel('Height (miles)')
,axis([0 35 0 3.5]),title('Rocket height versus distance')
```







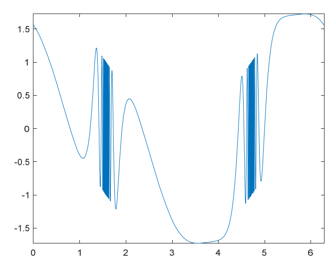
## Question 2

Q2- Use the **fplot** command to plot and investigate the function

$$\tan(\cos x) - sin(tan\ x)$$

$for\ 0 \le x \le 2\pi$. How many values of $x$ are needed to obtain the same plot using the plot command?

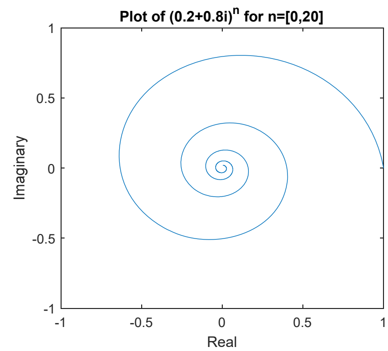
```
>> f=@(x)(tan(cos(x))-sin(tan(x)));
>> fplot(f,[0,2*pi])
>> [x,y]=fplot(f,[0,2*pi]);
Warning: Having two output arguments for fplot will be removed
in a future release. Use the XData and YData properties
instead.
> In fplot (line 180)
>> size(x)

ans =

           1        2674

>> size(y)

ans =

           1        2674
```



## Question 3

Q3- Plot the imaginary part versus the real part of the function $(0.2 + 0.8i)^n$ for $0 \leq n \leq 20$. Choose enough points to obtain a smooth curve. Label each axis and put a title on the plot. Use the axis command to change the tick-label spacing.

```
>> z=0.2+0.8i;
>> n=0:0.01:20;
>> plot(z.^n),xlabel('Real'),ylabel('Imaginary'),
>> title('Plot of (0.2+0.8i)^n for n=[0,20]'),
>> axis([-1 1 -1 1])
```



## Saving Figures

To save a figure that can be opened in subsequent MATLAB sessions, save it in a figure file with the .fig file name extension.

To do this, select **Save** from the Figure window **File** menu or click the **Save** button (the disk icon) on the toolbar.

If this is the first time you are saving the file, the **Save As** dialog box appears. Make sure that the type is MATLAB Figure (*.fig). Specify the name you want assigned to the figure file. Click OK.

## Exporting Figures 1

To save the figure in a format that can be used by another application, such as the standard graphics file formats TIFF or EPS, perform these steps.

1. Select **Export Setup** from the **File** menu. This dialog lets you specify options for the output file, such as the figure size, fonts, line size and style, and output format.
2. Select **Export** from the **Export Setup** dialog. A standard **Save As** dialog appears.
3. Select the format from the list of formats in the **Save As** type menu. This selects the format of the exported file and adds the standard file name extension given to files of that type.
4. Enter the name you want to give the file, less the extension. Then click **Save.**

## Exporting Figures 2

On Windows systems, you can also copy a figure to the clipboard and then paste it into another application:

1. Select **Copy Options** from the **Edit** menu. The **Copying Options** page of the **Preferences** dialog box appears.
2. Complete the fields on the **Copying Options** page and click **OK.**
3. Select **Copy Figure** from the **Edit** menu.

# Hints for Improving Plots

The following actions, while not required, can nevertheless improve the appearance of your plots:

1. Start scales from zero whenever possible. This technique prevents a false impression of the magnitudes of any variations shown on the plot.
2. Use sensible tick-mark spacing. If the quantities are months, choose a spacing of 12 because 1/10 of a year is not a convenient division. Space tick marks as close as is useful, but no closer. If the data is given monthly over a range of 24 months, 48 tick marks might be too dense, and also unnecessary.
3. Minimize the number of zeros in the data being plotted. For example, use a scale in millions of dollars when appropriate, instead of a scale in dollars with six zeros after every number.
4. Determine the minimum and maximum data values for each axis before plotting the data. Then set the axis limits to cover the entire data range plus an additional amount to allow convenient tick-mark spacing to be selected.

For example, if the data on the *x*-axis ranges from 1.2 to 9.6, a good choice for axis limits is 0 to 10. This choice allows you to use a tick spacing of 1 or 2.

# Hints for Improving Plots

5. Use a different line type for each curve when several are plotted on a single plot and they cross each other; for example, use a solid line, a dashed line, and combinations of lines and symbols. Beware of using colors to distinguish plots if you are going to make black-and-white printouts and photocopies.
6. Do not put many curves on one plot, particularly if they will be close to each other or cross one another at several points.
7. Use the same scale limits and tick spacing on each plot if you need to compare information on more than one plot.

## 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 *p*th 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).




## Subplots

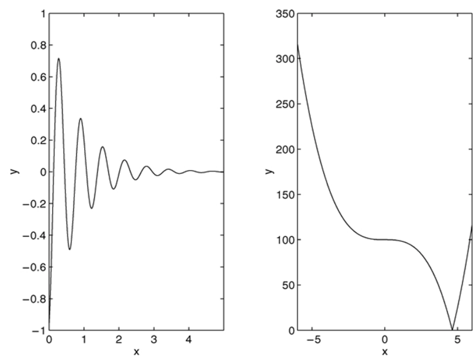
The following script file created Figure 5.2 − 1, which shows the plots of the functions $y = e^{-1.2x}\sin(10x + 5)$ for $0 \leq x \leq 5$ and $y = |x^3 - 100|$ for $-6 \leq x \leq 6$.

```
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

% Code Fragment

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





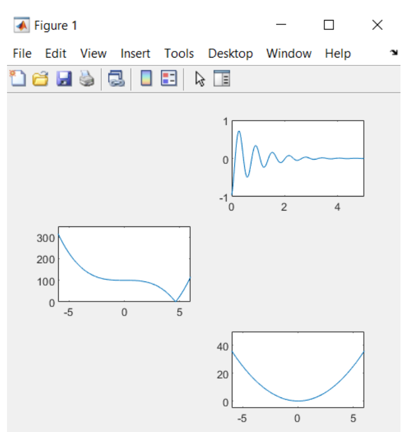


## Subplots

The following script file created a figure, which shows the plots of the functions $y = e^{-1.2x}\sin(10x + 5)$ for $0 \le x \le 5$, and $y = |x^3 - 100|$ and $y = x^2$ for $-6 \le x \le 6$.

Plot.m

```
x=0:0.01:5;
y=exp(-1.2*x).*sin(10*x+5);
subplot(3,2,2);
plot(x,y),axis([0 5 -1 1]);
x=-6:0.01:6;
y=abs(x.^3-100);
subplot(3,2,3);
plot(x,y),axis([-6 6 0 350]);
y=x.^2;
subplot(3,2,6);
plot(x,y),axis([-6 6 -5 50]);
```



## 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 must plot the data twice by typing `plot(x,y,'*',x,y,':')`.

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

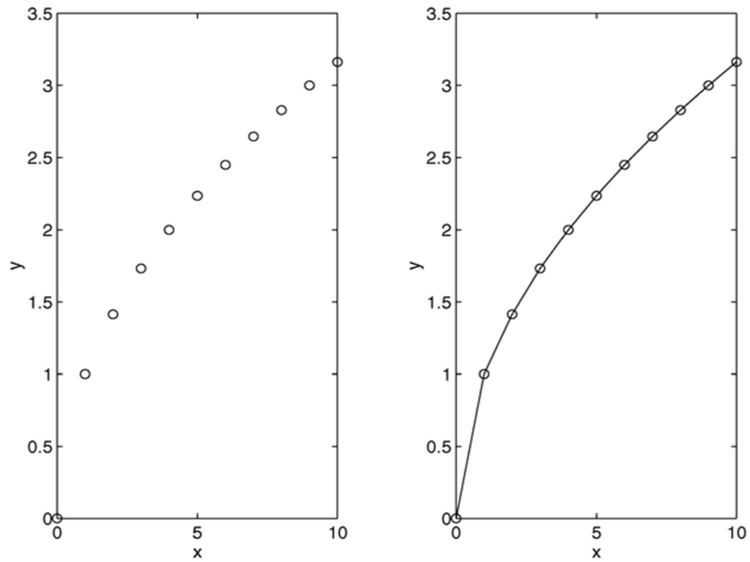
## Specifiers for Data Markers, Line Types, and Colors

| **Data markers**[†] | | **Line types** | | **Colors** | |
|---|---|---|---|---|---|
| Dot (.) | . | Solid line | — | Black | k |
| Asterisk (*) | * | Dashed line | -- | Blue | b |
| Cross (×) | × | Dash-dotted line | -. | Cyan | c |
| Circle (∘) | ∘ | Dotted line | .... | Green | g |
| Plus sign (+) | + | | | Magenta | m |
| Square (▫) | s | | | Red | r |
| Diamond (◊) | d | | | White | w |
| Five-pointed star (w) | p | | | Yellow | y |

[†]Other data markers are available. Search for "markers" in MATLAB help.

# Use of Data Markers



# 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 the next slide.

```
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)')
```

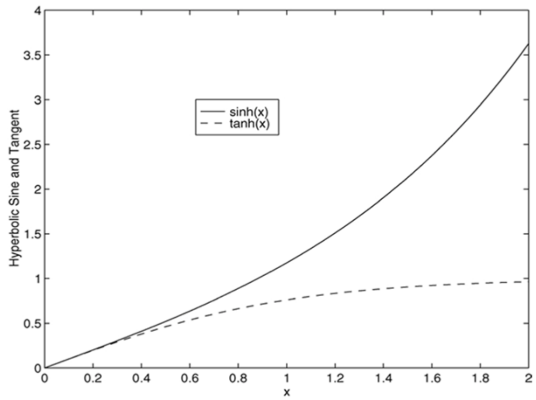
 

# Application of the Legend Command

%Code Fragment

x = 0:0.01:2; y = sinh(x); z = tanh(x);

plot(x,y,x,z,'− −'), legend('sinh(x)','tanh(x)')

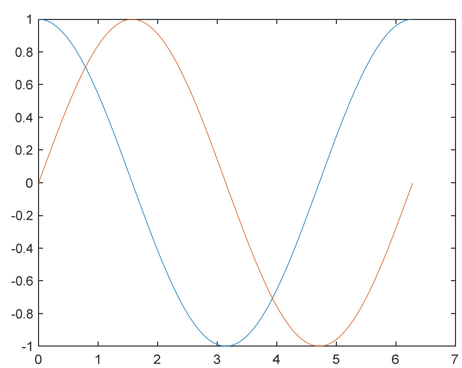






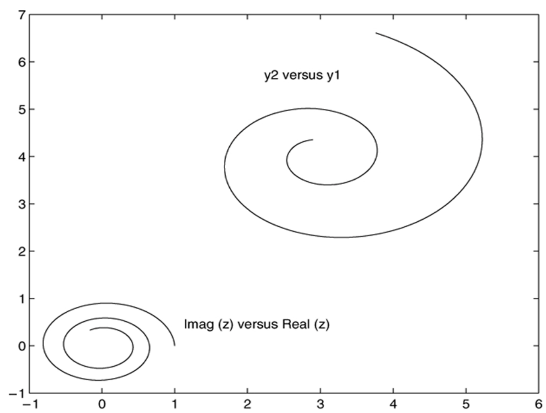
## Application of hold command:

```
>> x=0:0.01:2*pi;
>> y=cos(x);
>> plot(x,y);
>> hold
Current plot held
>> y=sin(x);
>> plot(x,y)
```



# Application of the Hold Command

```
figure
x = -1:0.01:1;
y1 = 3+exp(-x).*sin(6*x)
y2 = 4+exp(-x).*cos(6*x)
plot((0.1+0.9i).^(0:0.01:10)), hold, plot(y1, y2), gtext('y2 vs. y1'), gtext('Imag(x) vs. real(z)')
```



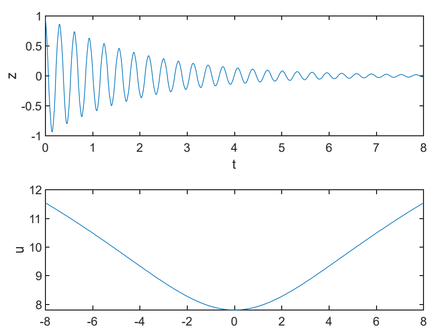



## Question 4

Q4- Pick a suitable spacing for $t = 0{:}0.01{:}8;$ *and* $v = -8{:}0.01{:}8;$ and then use the **subplot** command to plot the following functions

$$z = e^{-0.5t} cos(20t - 6)\ for\ 0 \le t \le 8$$
$$u = 6\ log10\ (v^2 + 20)\ for\ -8 \le v \le 8.$$

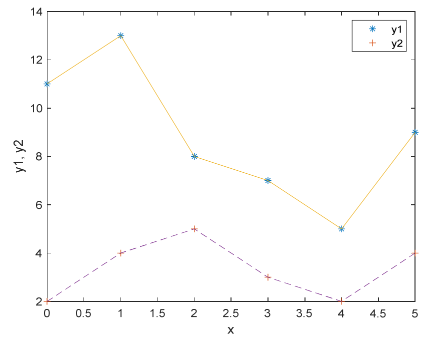
```
>> t=0:0.01:8;
>> v=-8:0.01:8;
>> z=exp(-0.5*t).*cos(20*t-6);
>> u=6*log10(v.^2+20);
>> subplot(2,1,1)
>> plot(t,z),xlabel('t'),ylabel('z')
>> subplot(2,1,2)
>> plot(v,u),xlabel('v'),ylabel('u')
```



## Question 5

Q5- Plot the following two data sets on the same plot. For each set, $x = 0, 1, 2, 3, 4, 5$. Use a different data marker for each set. Connect the markers for the first set with solid lines. Connect the markers for the second set with dashed lines. Use a legend, and label the plot axes appropriately. The first set is $y = 11, 13, 8, 7, 5, 9$. The second set is $y = 2, 4, 5, 3, 2, 4$.

```
>> x=0:5;y1=[11,13,8,7,5,9];
>> y2=[2,4,5,3,2,4];
>> plot(x,y1,'*',x,y2,'+',x,y1,x,y2,'--'),
>> xlabel('x'),ylabel('y1,y2'),
>> legend('y1','y2')
```

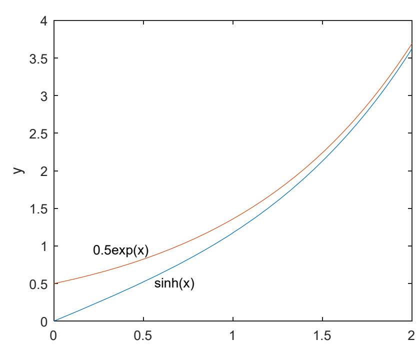


## Question 6

Q6- Plot $y = \sinh x$ and $y = 0.5e^x$ on the same plot for $0 \le x \le 2$. Use a solid line type for each, the ***gtext*** command to label the *sinh x* curve, and the $0.5e^x$ curve. Label the plot axes appropriately.

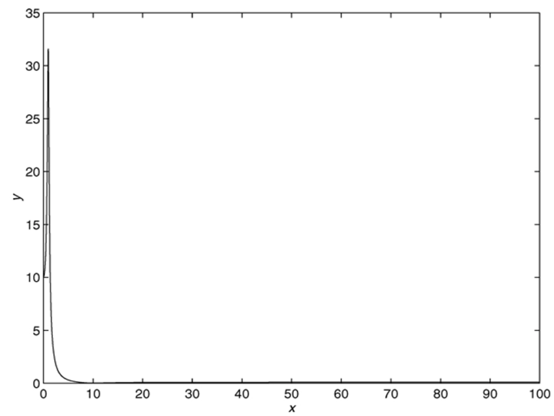
```
>> x=0:0.01:2;
>> y1=sinh(x);y2=0.5*exp(x);
>> plot(x,y1,x,y2),xlabel('x'),ylabel('y'),
>> gtext('sinh(x)'),gtext('0.5exp(x)')
>> sinh(0)
```



# Why use log scales? Rectilinear scales cannot properly display variations over wide ranges

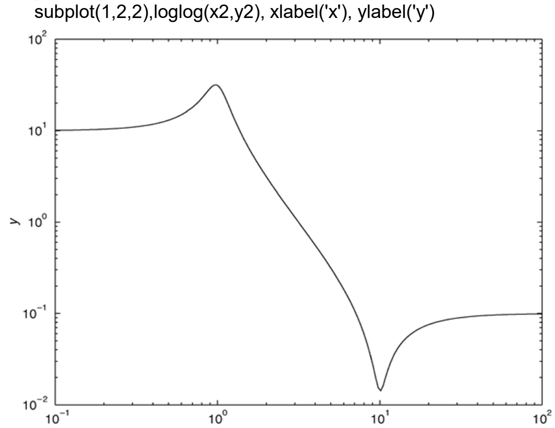
```
x1 = 0:0.01:100;
u1 = x1.^2;
num1 = 100*(1-0.01*u1).^2 + 0.02*u1;
den1 = (1-u1).^2+0.1*u1;
y1 = sqrt(num1./den1);
subplot(1,2,1), plot(x1,y1),xlabel('z'), ylabel('y')
```



# A log-log plot can display wide variations in data values

```
x2 = logspace(-2,2,500);
u2 = x2.^2;
num2 = 100*(1-0.01*u2).^2+0.02*u2;
den2=(1-u2).^2+0.1*u2;
y2=sqrt(num2./den2);
subplot(1,2,2),loglog(x2,y2), xlabel('x'), ylabel('y')
```



## Logarithmic Plots

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 $10^{-1} = 0.1$ 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.




## Logarithmic Plots

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 previous 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*.

# Logarithmic Plots

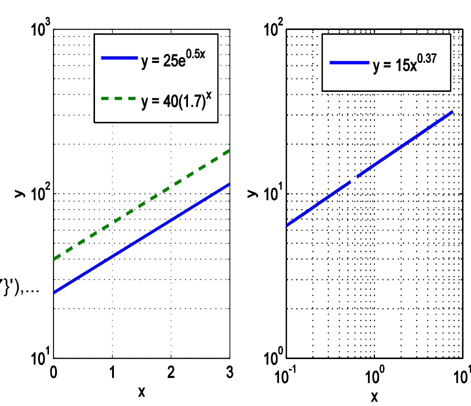
MATLAB has three commands for generating plots having log scales. The appropriate command depends on which axis must have a log scale.

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.




# Exponential and Power Functions Plotted on Log Scales

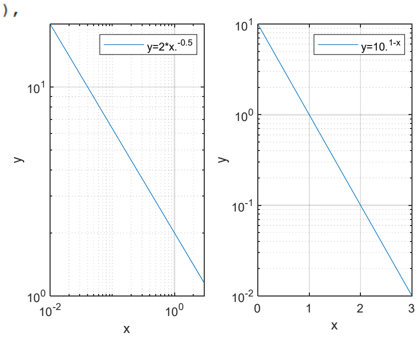
```
x1 = 0:0.01:3;
y1 = 25*exp(0.5*x1);
y2 = 40*(1.7.^x1);
x2 = logspace(-1,1,500);
y3 = 15*x2.^(0.37);
subplot(1,2,1), semilogy(x1,y1,x1,y2,'--'), ...
  legend('y = 25e^{0.5x}', 'y=40(1.7)^x'), ...
  xlabel('x'), ylabel('y'), grid, ...
  subplot(1,2,2), loglog(x2,y3),legend('y=15x^{0.37}'),...
  xlabel('x'), ylabel('y'),grid
```



## Question 7

Q7- Plot the following functions using axes that will produce a straight-line plot. The power function is $y = 2x^{-0.5}$, and the exponential function is $y = 10^{1-x}$.

```
>> x1=0.01:0.01:3;
y1=2*x1.^(-0.5);
subplot(1,2,1),loglog(x1,y1),legend('y=2*x.^{-0.5}'),
xlabel('x'),ylabel('y'),grid,
y2=10.^(1-x1);
subplot(1,2,2),semilogy(x1,y2),xlabel('x'),
ylabel('y'),grid,legend('y=10.^{1-x}')
```

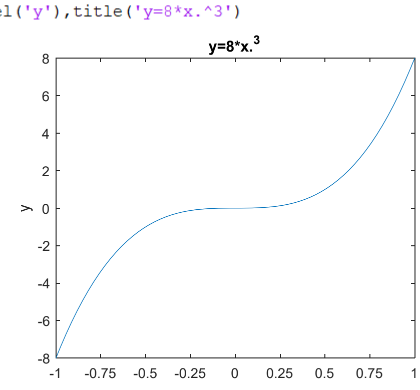


## Question 8

Q8- Plot the function $y = 8x^3\ for\ -1 \leq\ x \leq 1$ with a tick spacing of 0.25 on the $x$ axis and 2 on the $y$ axis.

```
>> x=-1:0.01:1;
>> y=8*x.^3;
>> plot(x,y),set(gca,'XTick',-1:0.25:1,'YTick',-8:2:8),
>> xlabel('x'),ylabel('y'),title('y=8*x.^3')
```



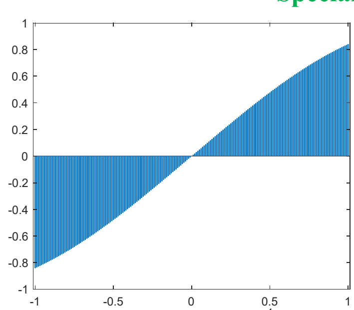

 

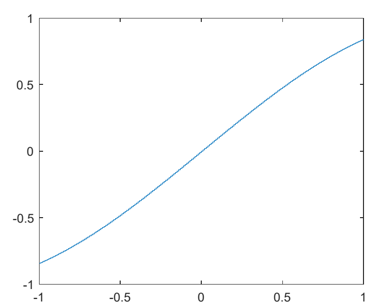
## Specialized plot commands:

| 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. |

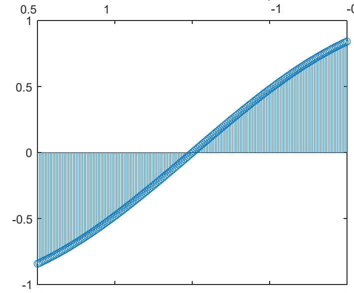
 

## Specialized plot commands:



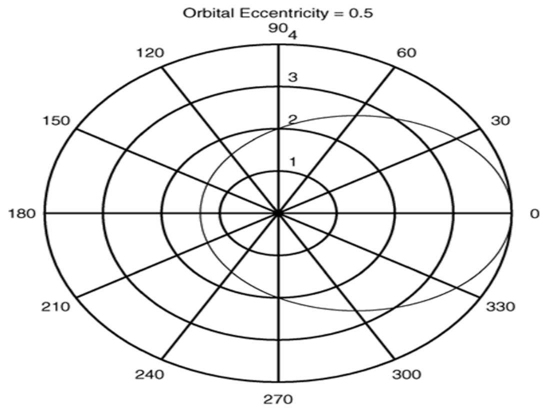

```
>> x=-1:0.01:1;
>> y=sin(x);
>> bar(x,y)
>> stairs(x,y)
>> stem(x,y)
```



# A Polar Plot Showing an Orbit

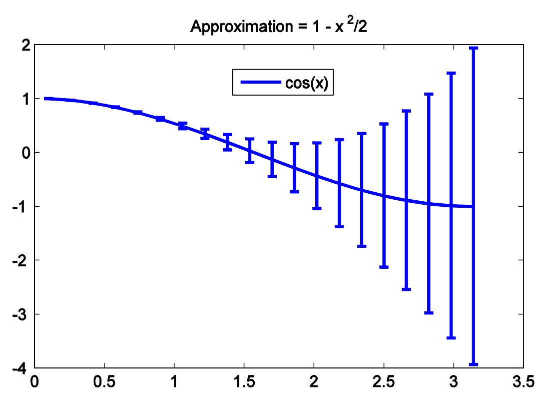
```
% Code Fragment
theta = 0:pi/90:2*pi;
r = 2./(1−0.5*cos(theta));
polarplot(theta,r)   % in Octave use polar(theta,r)
```



# The errorbar Command

```
% Code Fragment
x = linspace(0.1, pi, 20);
approx = 1 - x.^2/2;
error = approx - cos(x);
errorbar(x, cos(x), error), legend('cos(x)'), title('Approximation = 1 - x^2/2')
```



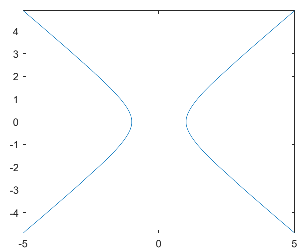
## Plotting Implicit Functions

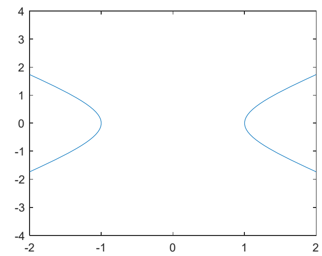
MATLAB provides the function `fimplicit(f)` to plot the implicit function defined by the equation $f(x,y) = 0$ over the default interval [−5 5] for $x$ and $y$. For example, to plot the hyperbola defined by $x^2 - y^2 - 1 = 0$ over the default interval of [−5 5], you type

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

If the limits for $x$ are [−2 2] and the limits for $y$ are [−4 4], you would type

```
>>fimplicit(@(x,y) x.^2 - y.^2 - 1, [-2 2 -4 4])
```







## Publishing Reports Containing Graphics 1

Open the Editor, type in the M-file that forms the basis of the report, and save it. Use the double percent character (%%) to indicate a section heading in the report. This character marks the beginning of a new cell, which is a group of commands. Use the `publish` and `open` functions to create the report in the desired format, such as HTML.

## Publishing Reports Containing Graphics

For example, the code is

```
%% Example of Report Publishing:
% Plotting the cubic y = x^3 – 6x^2 + 10x+4.
%% Create the independent variable.
x = linspace(0, 4, 300); % Use 300 points between 0 and 4.
%% Define the cubic from its coefficients.
p = [1, –6, 10, 4]; % p contains the coefficients.
%% Plot the cubic.
plot(x,polyval(p,x)), xlabel('x'), ylabel('y')
```

The report is shown on the next slide.




## Example of Report Publishing

### Example of Report Publishing:

Plotting the cubic y = x^3 − 6x^2 + 10x+4.

**Contents**

- Create the independent variable.
- Define the cubic from its coefficients.
- Plot the cubic.

**Create the independent variable.**

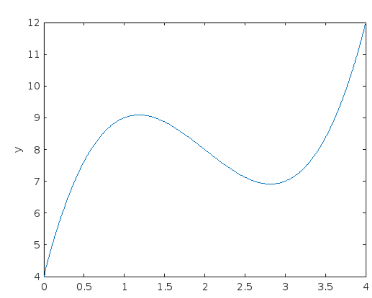
```
x = linspace(0, 4, 300); % Use 300 points between 0 and 4.
```

**Define the cubic from its coefficients.**

```
p = [1, -6, 10, 4]; % p contains the coefficients.
```

**Plot the cubic.**

```
plot(x,polyval(p,x)), xlabel('x'), ylabel('y'),
%publish ('polyplot','html')
```



## The Live Editor

A *live script* is an *interactive* document that contains output, including graphics, along with the code that produced them, together in a single interactive environment called the *Live Editor*. You can also include formatted text, images, hyperlinks, and equations to produce an interactive shareable narrative. Live scripts are stored in a file with the extension **.mlx**. You can convert the scripts to HTML or PDF files for publication.

The Live Editor enables you to work more efficiently because you can write, execute, and test code without leaving the environment, and you can run blocks of code individually or the whole file. You can see the results and graphics next to the code that produced them, and you can see errors at the file location where they occur.




## The Live Editor

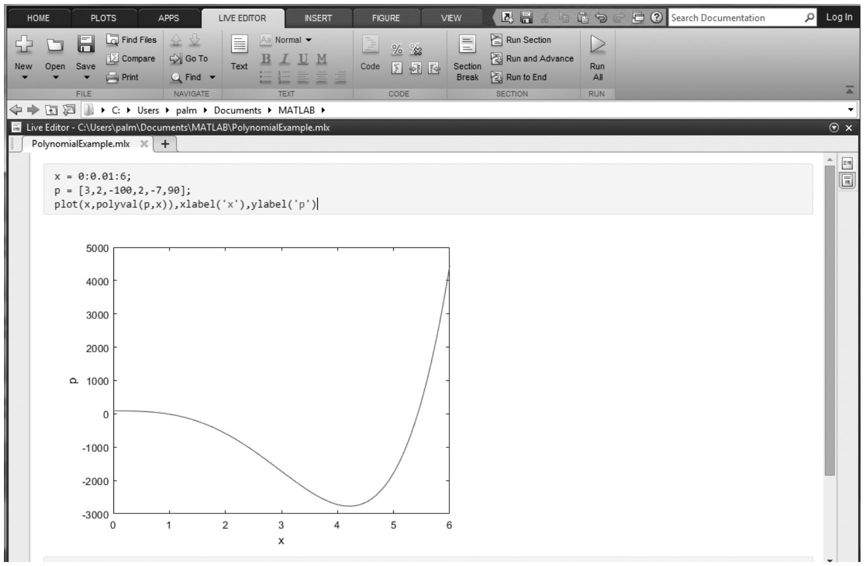
The Live Editor enables you to work more efficiently because you can write, execute, and test code without leaving the environment, and you can run blocks of code individually or the whole file. You can see the results and graphics next to the code that produced them, and you can see errors at the file location where they occur.

Two ways to open a new live script are:

1. On the Home tab, in the New drop-down menu, select **Live Script**. Highlight the desired commands from the Command History, right-click, and select **Create Live Script**, or

2. Enter your code in the Live Editor. See next slide for an example. After entering the code, click at the top of the blue boundary to the left. The code will run. In the example shown, the plot will appear after the third line of code is executed.

# The Live Editor




 Source: MATLAB 

# 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, or
- You want to change plot characteristics such as tick spacing, fonts, bolding, italics, and colors.

The interactive plotting environment in MATLAB is a set of tools for:

- 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, and
- Editing properties of graphics objects, such as their color, line weight, and font.

## The Figure Toolbar Displayed

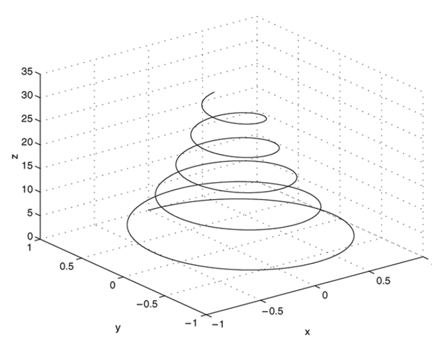

## The Figure and Plot Edit Toolbars Displayed


 Source: MATLAB 

## Three-Dimensional Line Plots

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
```

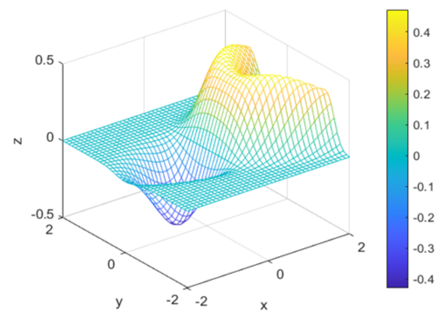


# Surface Plots

The following session shows how to generate the surface plot of the function $z = xe^{-[(x-y^2)^2+y^2]}$, for $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$, 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')
```

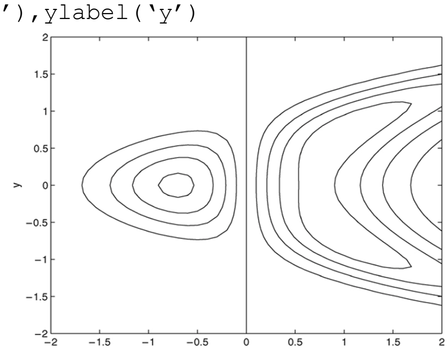


# Contour Plot of the Function

The following session generates the contour plot of the function whose surface plot is shown in the previous slide ; namely, $z = xe^{-[(x-y^2)^2+y^2]}$, for $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$, 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')
```



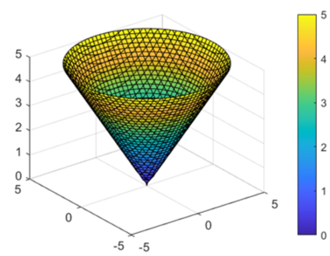

 

# Surface Plots of Implicit Functions

MATLAB provides the function fimplicit3(f) to plot the three-dimensional implicit function defined by the equation $f(x, y, z) = 0$ over the default interval [−5 5] for $x$, $y$, and $z$. You can specify the interval with the syntax fimplicit3(f,interval).

For example, to plot the upper half of the hyperboloid $x$^2 + $y$^2 − $z$^2 = 0 you specify the interval as for $z$ as [0  5], and for $x$ and $y$, use the default interval [−5  5], as follows.

```
>>f = @(x,y,z) x.^2 + y.^2 − z.^2;

>>interval = [−5 5 −5 5 0 5];

>>fimplicit3(f,interval)
```





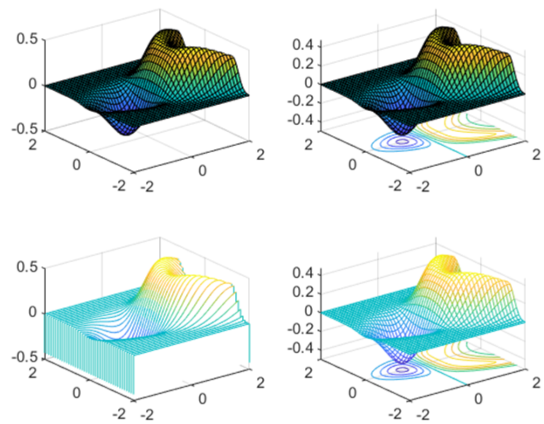

# Other Three-Dimensional Plotting

| **Function** | **Description** |
|---|---|
| `surf(x,y,z)` | Creates a shaded 3D mesh surface plot. |
| `surfc(x,y,z)` | Same as `surf` but draws contours under the surface. |
| `waterfall(x,y,z)` | Same as `mesh` but draws mesh lines in one direction only. |

## Example

Plots of the surface $z = xe^{-((x-y^2)^2+y^2)}$ created with the mesh function and its variant forms: a) surf, b) surfc, c) waterfall, d) meshc,

```
>> [X,Y]=meshgrid(-2:0.1:2);
>> Z=X.*exp(-((X-Y.^2).^2+Y.^2));
>> subplot(2,2,1);
>> surf(X,Y,Z)
>> subplot(2,2,2);
>> surfc(X,Y,Z)
>> subplot(2,2,3);
>> waterfall(X,Y,Z)
>> subplot(2,2,4);
>> meshc(X,Y,Z)
```





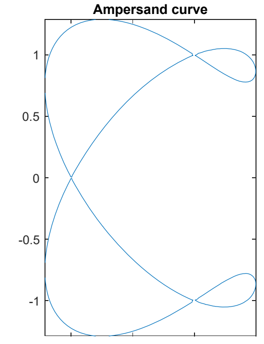
## Question 9

Q9- Obtain the plot of the following implicit function, known as the Ampersand curve. Use the axis equal command.

$$(y^2 - x^2)(x-1)(2x-3) = 2(x^2 + y^2 - 2x)^2$$

axis equal sets the aspect ratio so that the data units are the same in every direction.

```
>> fimplicit(@(x,y)(y.^2-x.^2).*(x-1).*(2*x-3)-2*(x.^2+y.^2-2*x).^2),
>> axis equal,title('Ampersand curve')
```

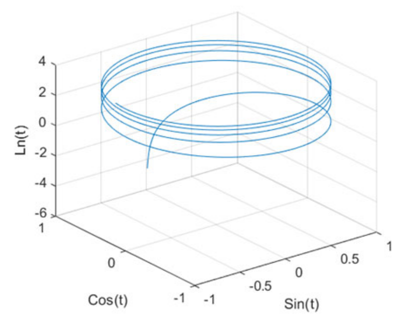


## Question 10

Q10- Use *plot3* and *fplot3* to plot the 3-D line plot described by $x = \sin(t), y = \cos(t), z = \ln(t)$ for $0 < t < 30$.

```
>> t=0:0.01:30;
>> plot3(sin(t),cos(t),log(t)),xlabel('Sin(t)'),ylabel('Cos(t)'),
>> zlabel('Ln(t)'),grid on
```

```
>> fplot3(@(t)sin(t),@(t)cos(t),@(t)log(t),[0,30]),
>> xlabel('Sin(t)'),ylabel('Cos(t)'),zlabel('Ln(t)'),grid on
```



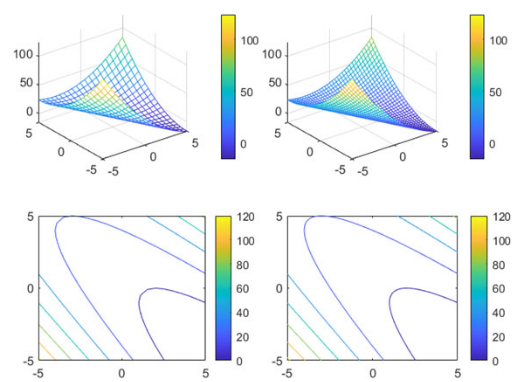




## Question 11:

Q11-Use mesh, fmesh, contour, and fcontour to create a surface plot and a contour plot of the function $z = (x-2)^2 + 2xy + y^2$.

```
>> [X,Y] = meshgrid(-5:0.5:5);
Z = (X-2).^2+2*X.*Y+Y.^2;
subplot(2,2,1);
mesh(X,Y,Z)
subplot(2,2,2);
fmesh(@(X,Y) (X-2).^2+2*X.*Y+Y.^2)
[X,Y] = meshgrid(-5:0.5:5);
Z = (X-2).^2+2*X.*Y+Y.^2;
subplot(2,2,3);
contour(X,Y,Z)
subplot(2,2,4);
fcontour(@(X,Y)(X-2).^2+2*X.*Y+Y.^2);
```

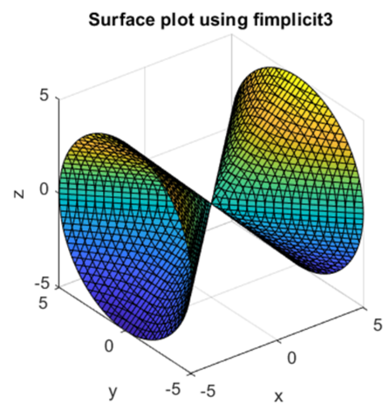


## Question 12

Q12- Use the fimplicit3 function to create a surface plot of the function

$$x^2 - y^2 - z^2 = 0$$

```
>> fimplicit3(@(x,y,z)(x.^2-y.^2-z.^2)),axis equal,
>> xlabel('x'),ylabel('y'),zlabel('z'),
>> title('Surface plot using fimplicit3')
```








Because learning changes everything.®

www.mheducation.com