# Introduction to MATLAB

**Dr./ Ahmed Nagib Elmekawy**

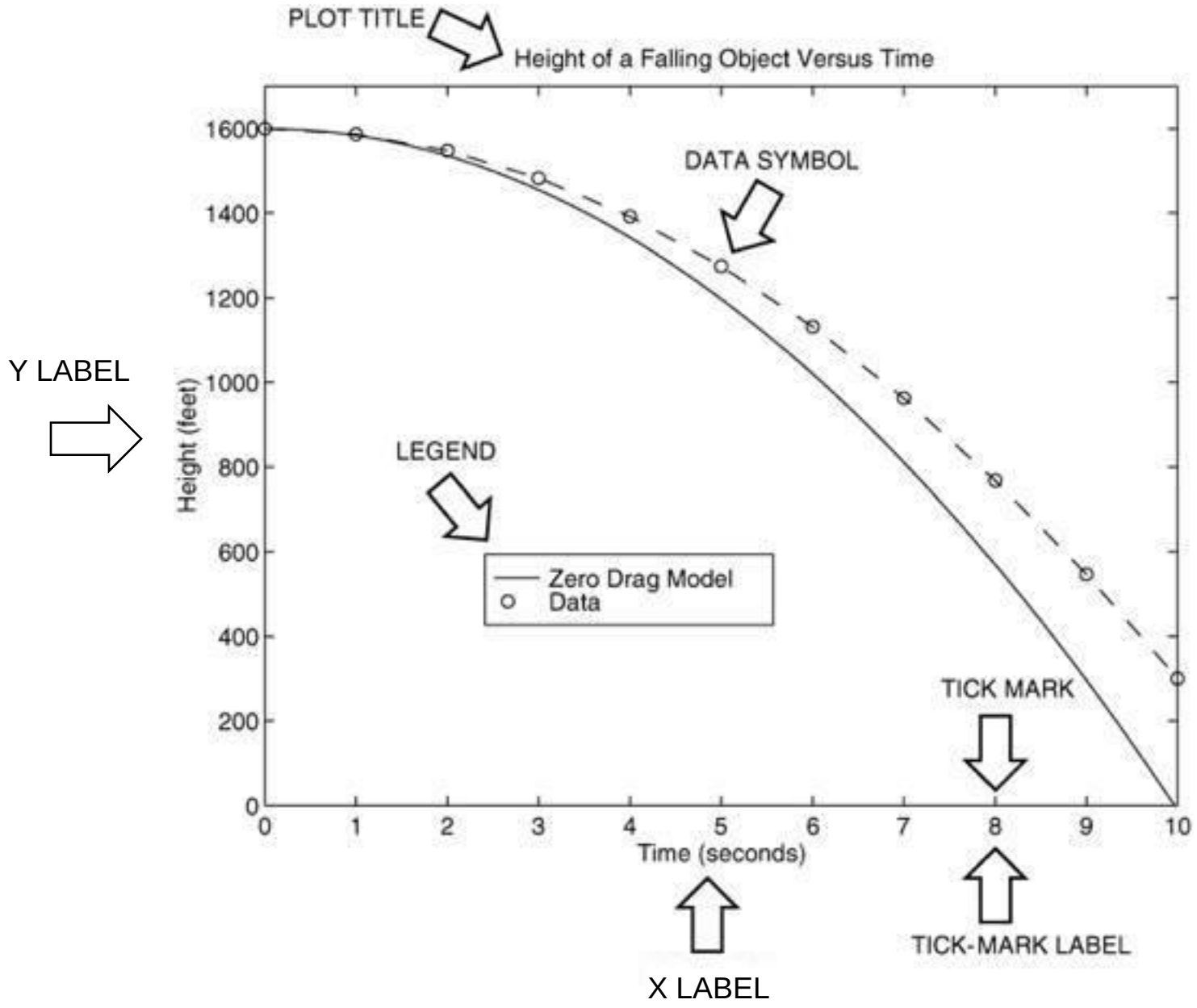
**Mechanical Engineering department, Alexandria university, Egypt**

**Spring 2023**

## Chapter 3

## Plotting in MATLAB

**Nomenclature for a typical *xy* plot.** Figure 5.1–1, page 220



## Command Plot

### Example

As a simple example, let us plot the function

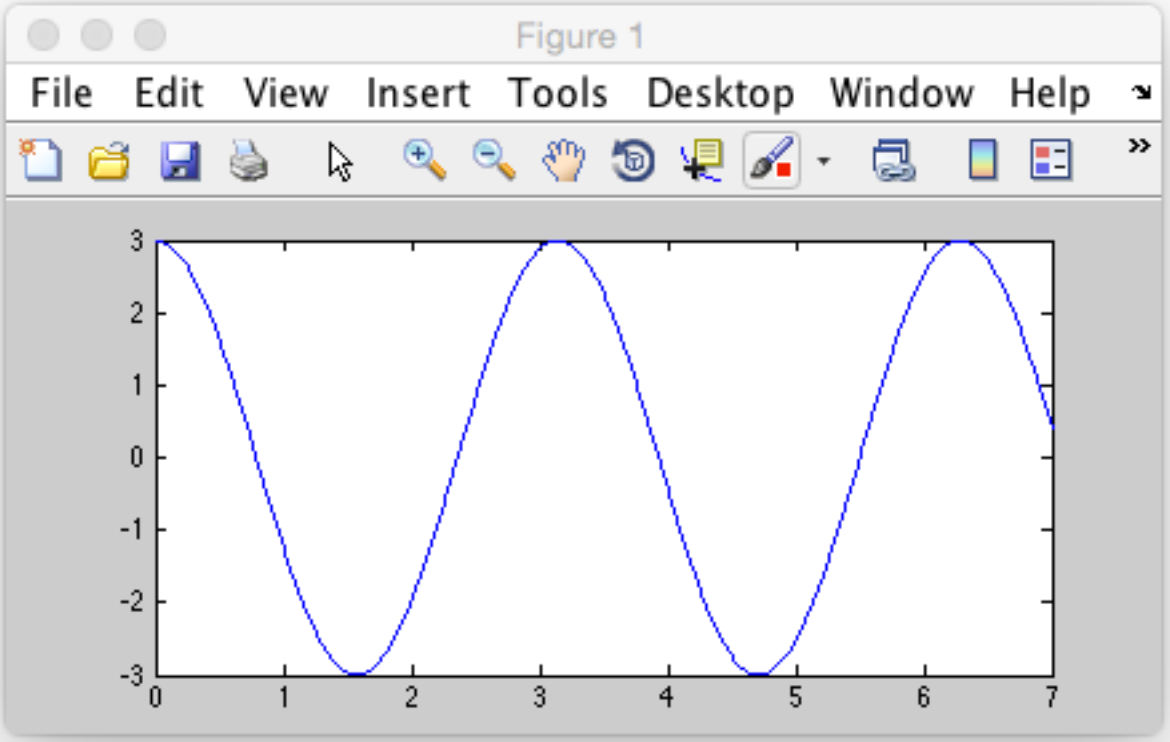
$$y = 3 \cos 2x \qquad \text{for } 0 \leq x \leq 7.$$

We choose to use an increment of 0.01 to generate a large number of x values in order to produce a smooth curve.

The function `plot(x,y)` generates a plot with the x values on the horizontal axis and the y on the vertical access, as seen in the following slide:

# Example 2-1 – cont.
# A graphics window showing a plot

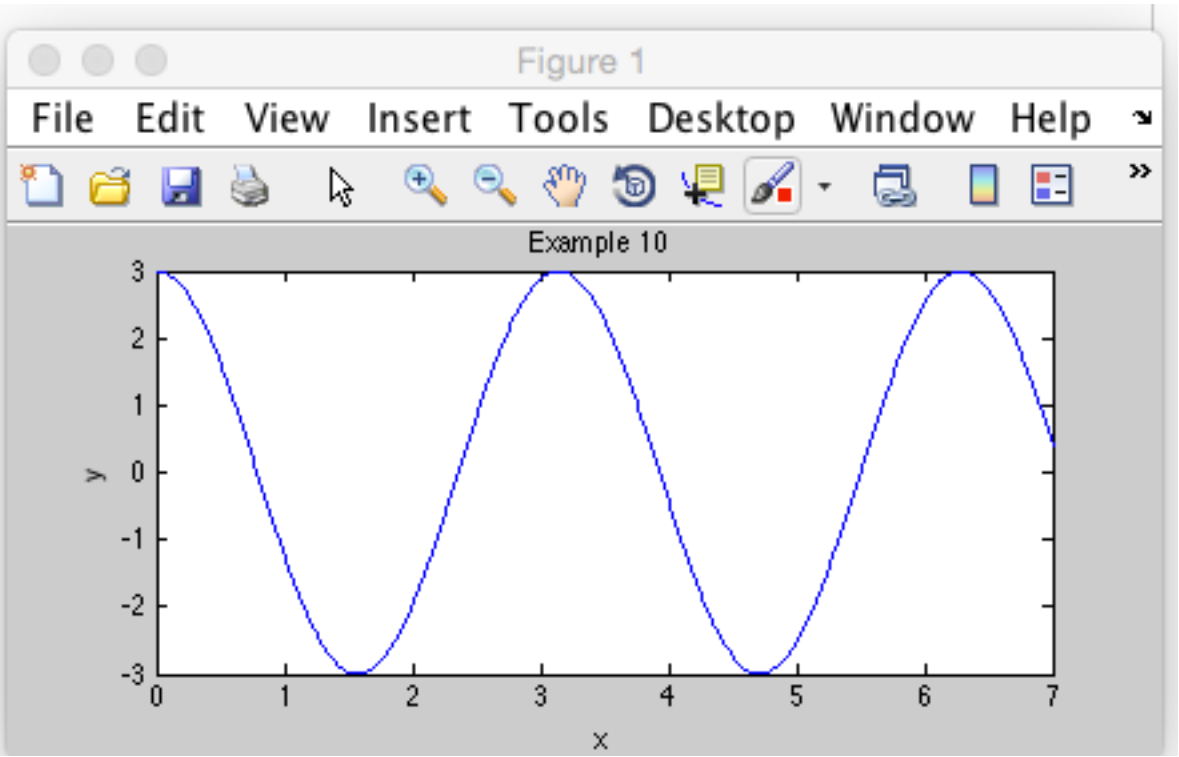
```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    plot(x,y)
```



**To add x and y labels and a title to the graph, we use the following commands:**

```
xlabel('x');
ylabel('y');
title('Example 10');
```

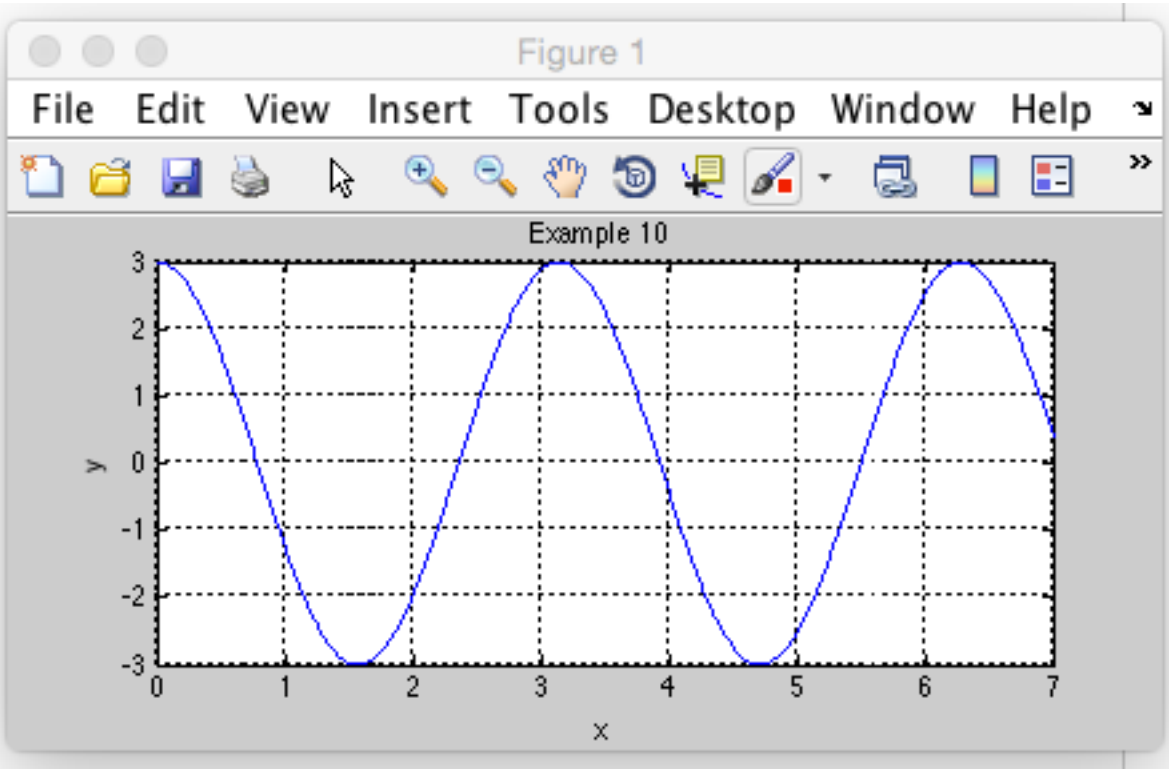
```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    plot(x,y)
4 -    xlabel('x');
5 -    ylabel('y');
6 -    title('Example 10');
7
```



**To add a grid to the graph, we use the following commands:**

`grid`

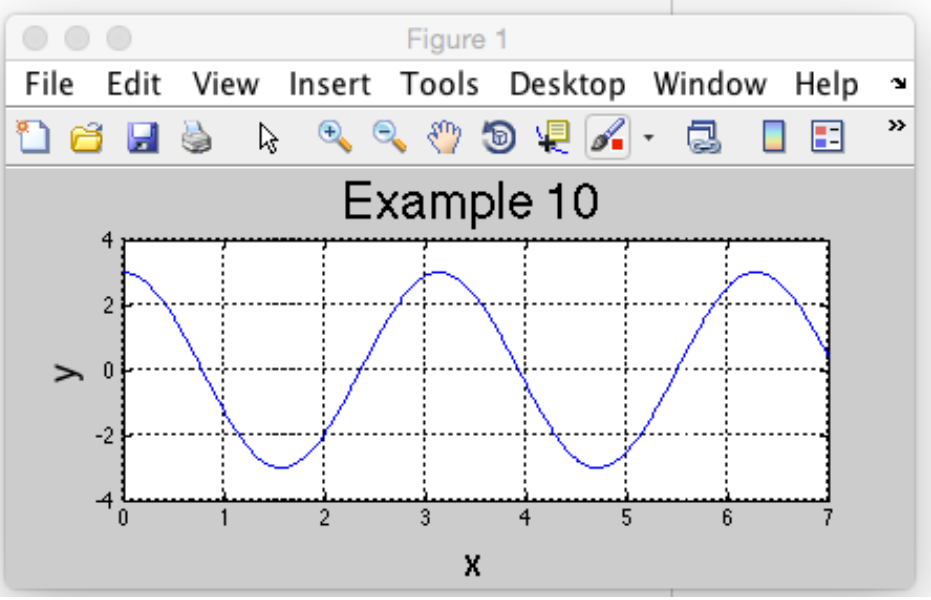
```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    plot(x,y)
4 -    xlabel('x');
5 -    ylabel('y');
6 -    title('Example 10');
7 -    grid
8
```



**To increase the font size of the xlabel, ylabel and title, we use the following commands:**

```
xlabel('x','FontSize',18);
ylabel('y','FontSize',18);
title('Example 10','FontSize',24);
```

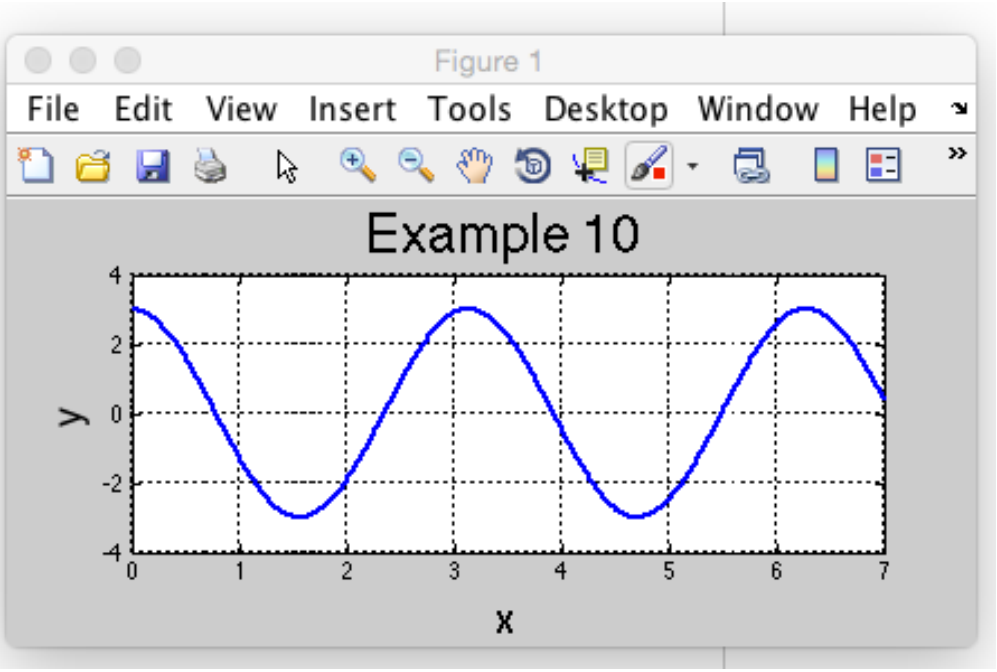
```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    plot(x,y)
4 -    xlabel('x','FontSize',18);
5 -    ylabel('y','FontSize',18);
6 -    title('Example 10','FontSize',24);
7 -    grid
8
9
```



**To increase the plotted line width in Matlab, we use the following commands:**

```
plot(x,y,'linewidth', 2)
```

```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    plot(x,y,'linewidth', 2)
4 -    xlabel('x','FontSize',18);
5 -    ylabel('y','FontSize',18);
6 -    title('Example 10','FontSize',24);
7 -    grid
8
9
```



**To save the figure in Matlab**

1. Use the menu system. Select Save as on the File menu in the Figure window.
2. Type the following command in the m file.

```
saveas(gcf, 'Example 10.tiff');
```

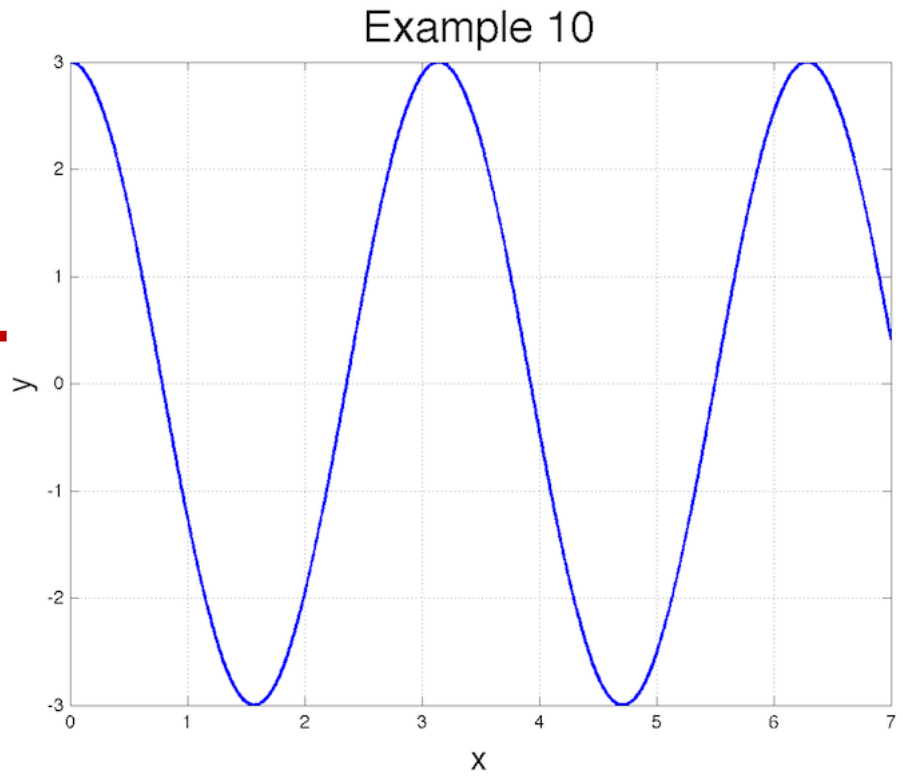
This command saves the current plot directly to working directory.

We recommend using use the previous command to save the graphs.

**To save the figure in Matlab, we use the following commands:**

```
saveas(gcf, 'Example 10.tiff');
```

```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    plot(x,y,'linewidth', 2)
4 -    xlabel('x','FontSize',18);
5 -    ylabel('y','FontSize',18);
6 -    title('Example 10','FontSize',24);
7 -    grid
8 -    saveas(gcf, 'Example 10.tiff');
9
```



## Data Markers and Line Types

In the graph, to choose different color or marker type or line type (solid, dashed, …..)

## Choose from the following table

| 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 |

If you want to see more options, type the following in the command windows

```
>>help plot
```

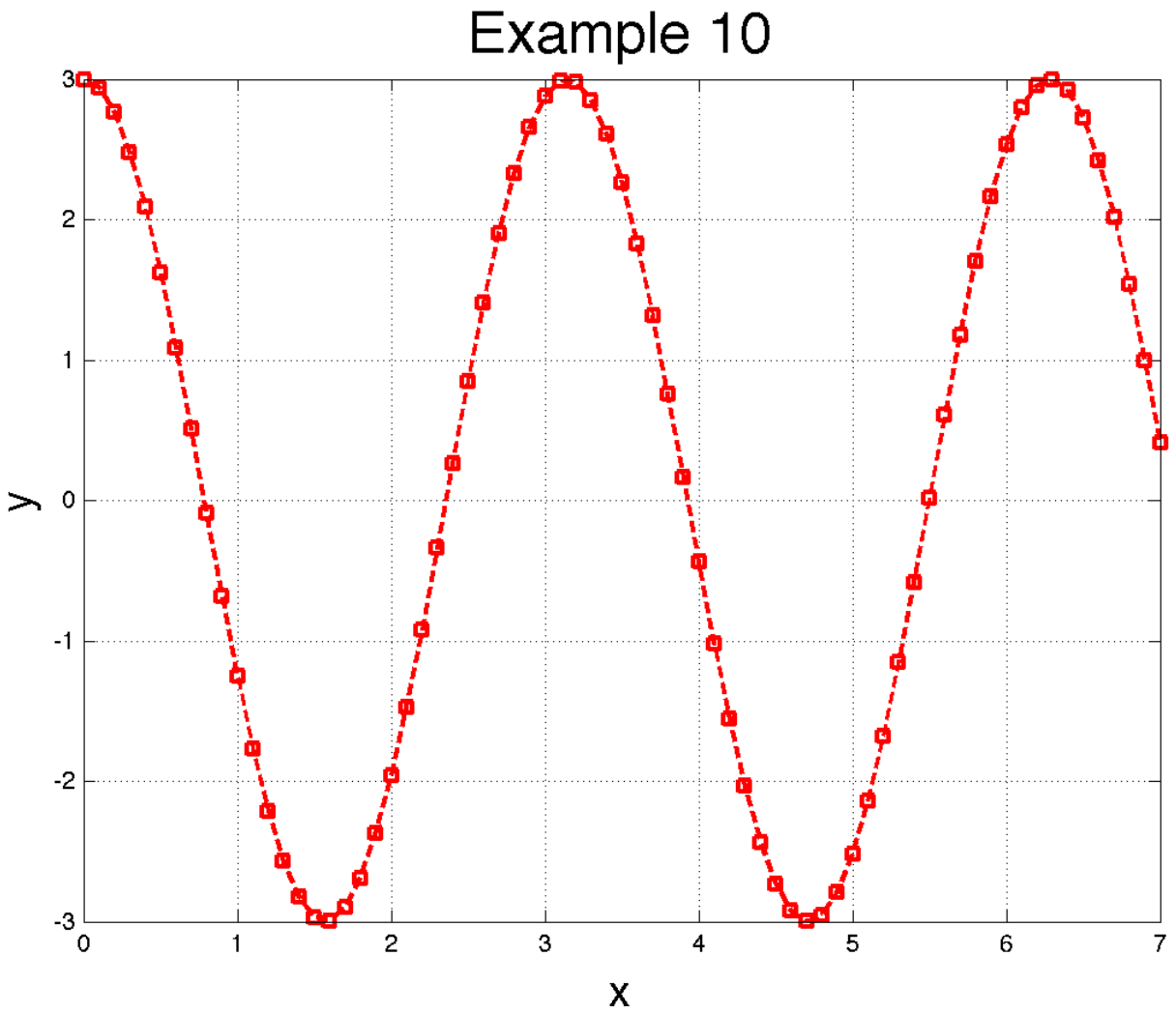
# Data Markers and Line Types

```
x=0:0.1:7;
y=3*cos(2*x);
z=5*sin(2*x);
plot(x,y,'--rs','LineWidth',2)
xlabel('x','FontSize',18);
ylabel('y','FontSize',18);
title('Example 10','FontSize',24);
grid
saveas(gcf, 'Example 10.tiff');
```

| 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 |

# Data Markers and Line Types

```
4 - plot(x,y,'--rs','LineWidth',2)
```



## Data Markers and Line Types

A plotting command with a certain color:

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

A combination of line type & color :

```
plot(x,y, '--r')
```

A combination of mark & color :

```
plot(x,y, '+r')
```

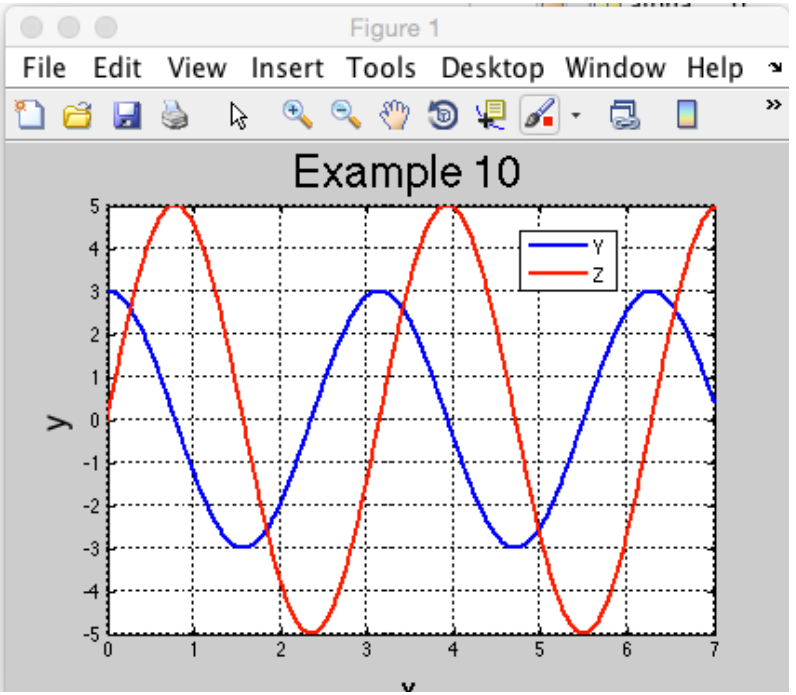
A combination of line type, mark & color :

```
plot(x,y, '--+r')
```

**If we want to plot two variables in the same graph, we use the following commands:**

```
hold on
and
legend('Y','Z', 'Exact','location', 'best')
```

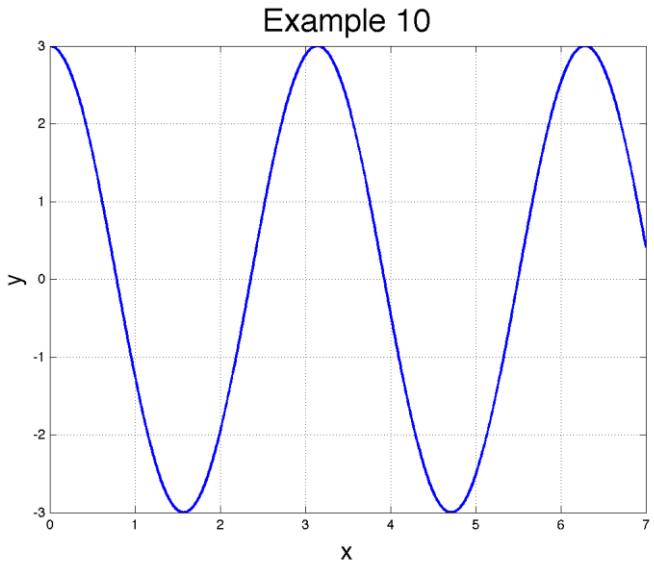
```
1 -    x=0:0.01:7;
2 -    y=3*cos(2*x);
3 -    z=5*sin(2*x)
4 -    plot(x,y,'linewidth', 2)
5 -    hold on
6 -    plot(x,z,'r','linewidth', 2)
7 -    xlabel('x','FontSize',18);
8 -    ylabel('y','FontSize',18);
9 -    title('Example 10','FontSize',24);
10 -   grid
11 -   legend('Y','Z', 'Exact','location', 'best')
12 -   saveas(gcf, 'Example 10.tiff');
13
```

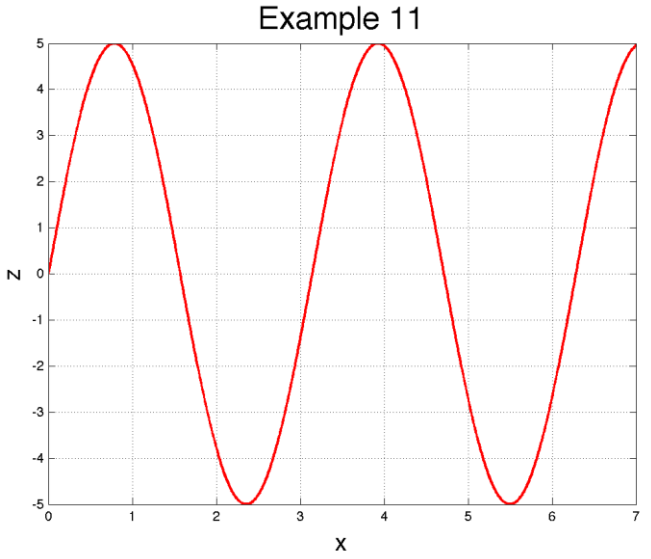



Location in the legend command specify the best location in the graph

# If we want to plot two figures, we use the following commands:

```
x=0:0.01:7;
y=3*cos(2*x);
z=5*sin(2*x);
figure(1)
plot(x,y,'linewidth', 2)
xlabel('x','FontSize',18);
ylabel('y','FontSize',18);
title('Example 10','FontSize',24);
grid
saveas(gcf, 'Example 10.tiff');
figure(2)
plot(x,z,'r','linewidth', 2)
xlabel('x','FontSize',18);
ylabel('z','FontSize',18);
title('Example 11','FontSize',24);
grid
saveas(gcf, 'Example 10_2.tiff');
```



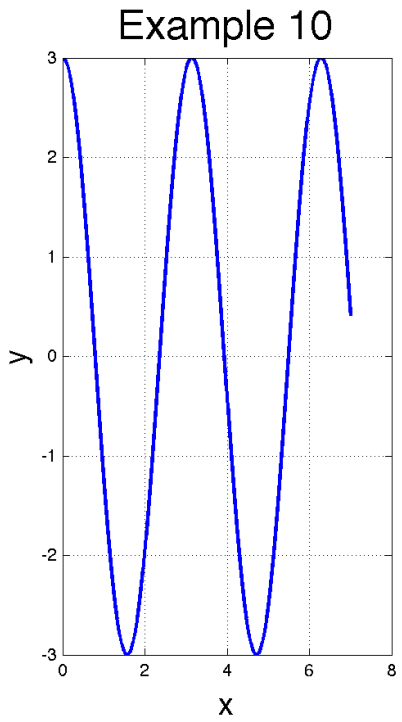
**If we want to plot two plots in the same window, we use the following commands:**

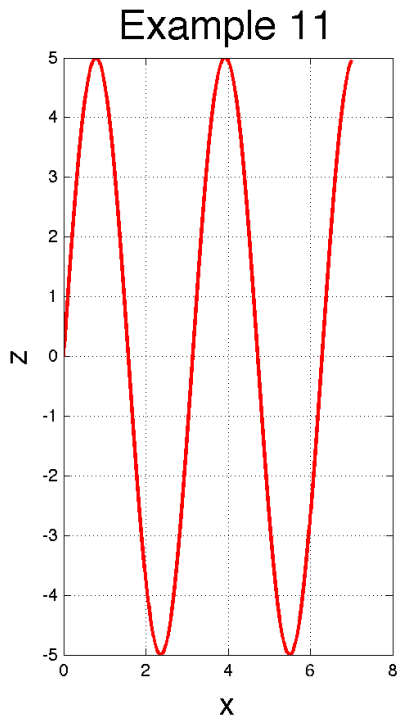
`subplot(1,2,1)`

**and**

`subplot(1,2,2)`

```
x=0:0.01:7;
y=3*cos(2*x);
z=5*sin(2*x);
subplot(1,2,1)
plot(x,y,'linewidth', 2)
xlabel('x','FontSize',18);
ylabel('y','FontSize',18);
title('Example 10','FontSize',24);
grid
subplot(1,2,2)
plot(x,z,'r','linewidth', 2)
xlabel('x','FontSize',18);
ylabel('z','FontSize',18);
title('Example 11','FontSize',24);
grid
saveas(gcf, 'Example 10_subplot.tiff')
```

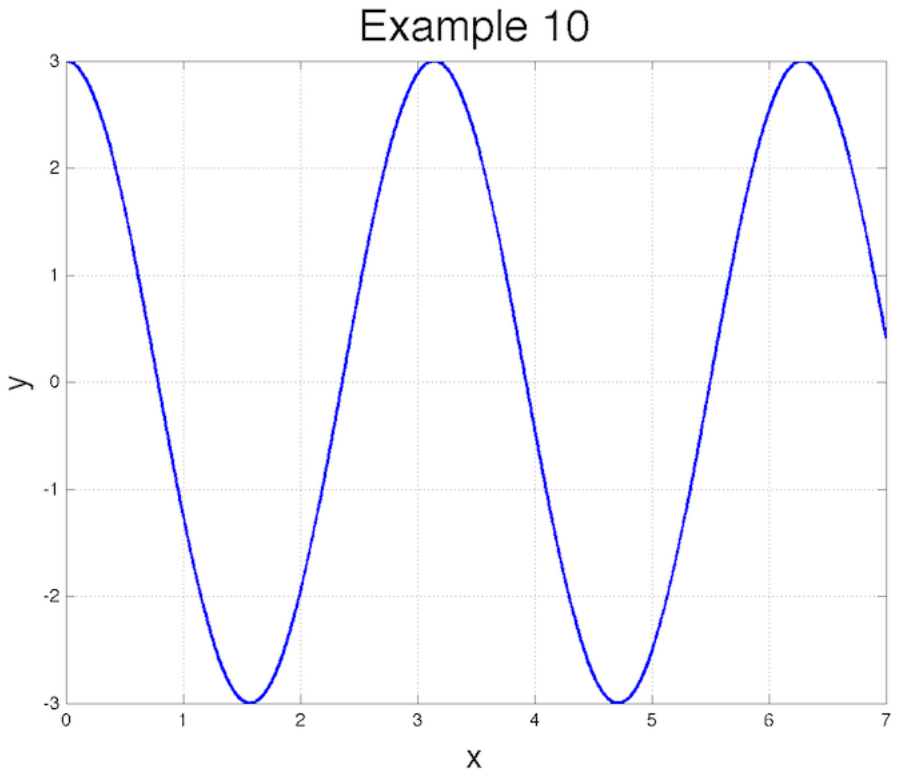


## Plot Summary:

So we will copy and paste the plotting commands shown in the previous example to get a nice looking saved graph.

```
1 -     x=0:0.01:7;
2 -     y=3*cos(2*x);
3 -     plot(x,y,'linewidth', 2)
4 -     xlabel('x','FontSize',18);
5 -     ylabel('y','FontSize',18);
6 -     title('Example 10','FontSize',24);
7 -     grid
8 -     saveas(gcf, 'Example 10.tiff');
9
```



## Axis command

The maximum and minimum of the coordinates on the graph may be specified by the command:

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

A figure can be reshaped by the command:

```
axis('square’)
```

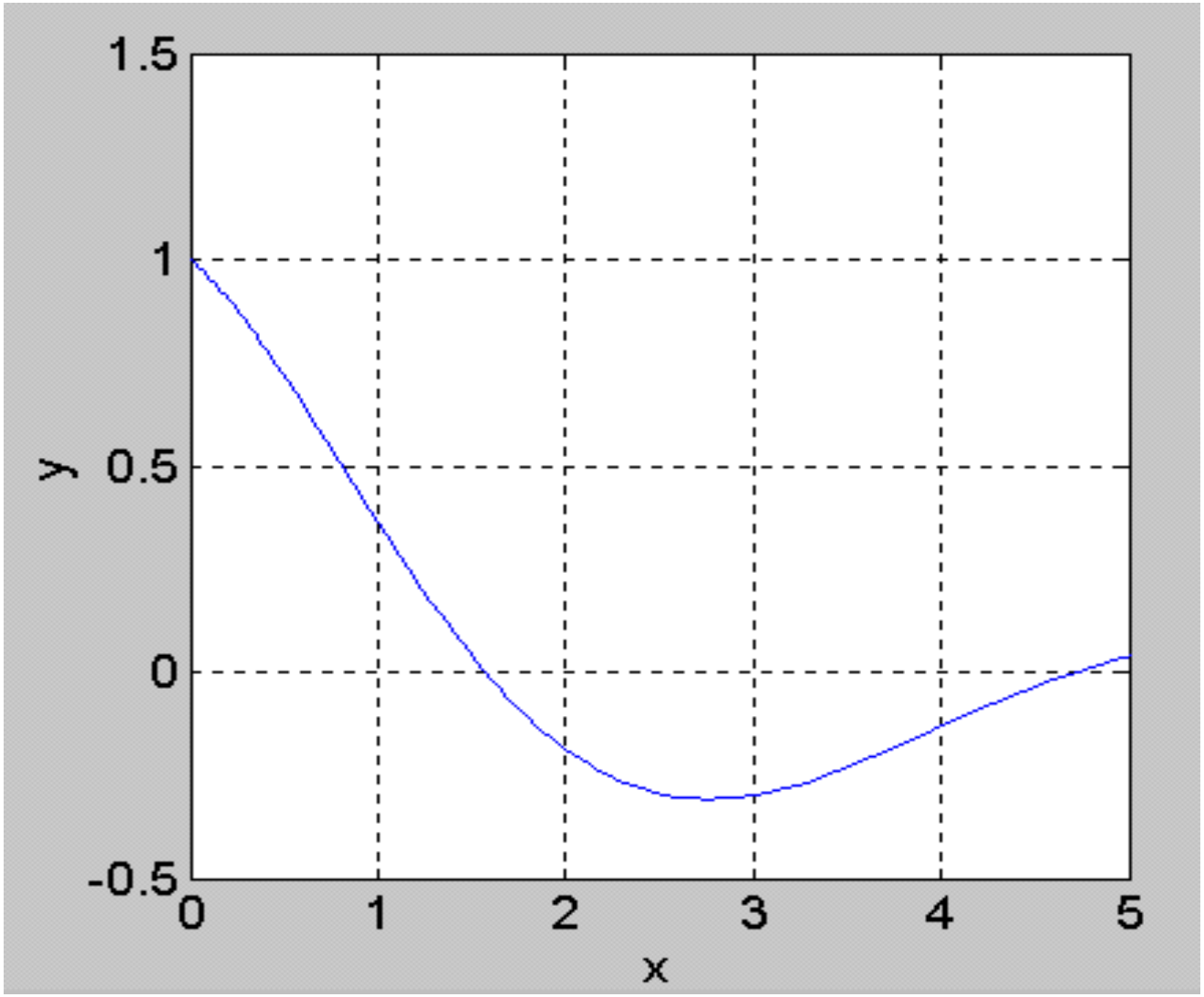
## Figure clearing

The command `clf` clears everything inside the graphic window.

```
clf
```

## Example

```
clf
x=0:0.01:10;
y = cos(x).*exp(-0.4*x);
plot(x,y)
xlabel(’x')
ylabel('y')
axis([0 , 5 , -0.5 , 1.5])
axis('square')
grid on
```


1.5
1
0.5
0
-0.5
y
0
1
2
3
4
5
x

## Text in graph

Text can be written in a graph by the following command:

```
text
```

Example

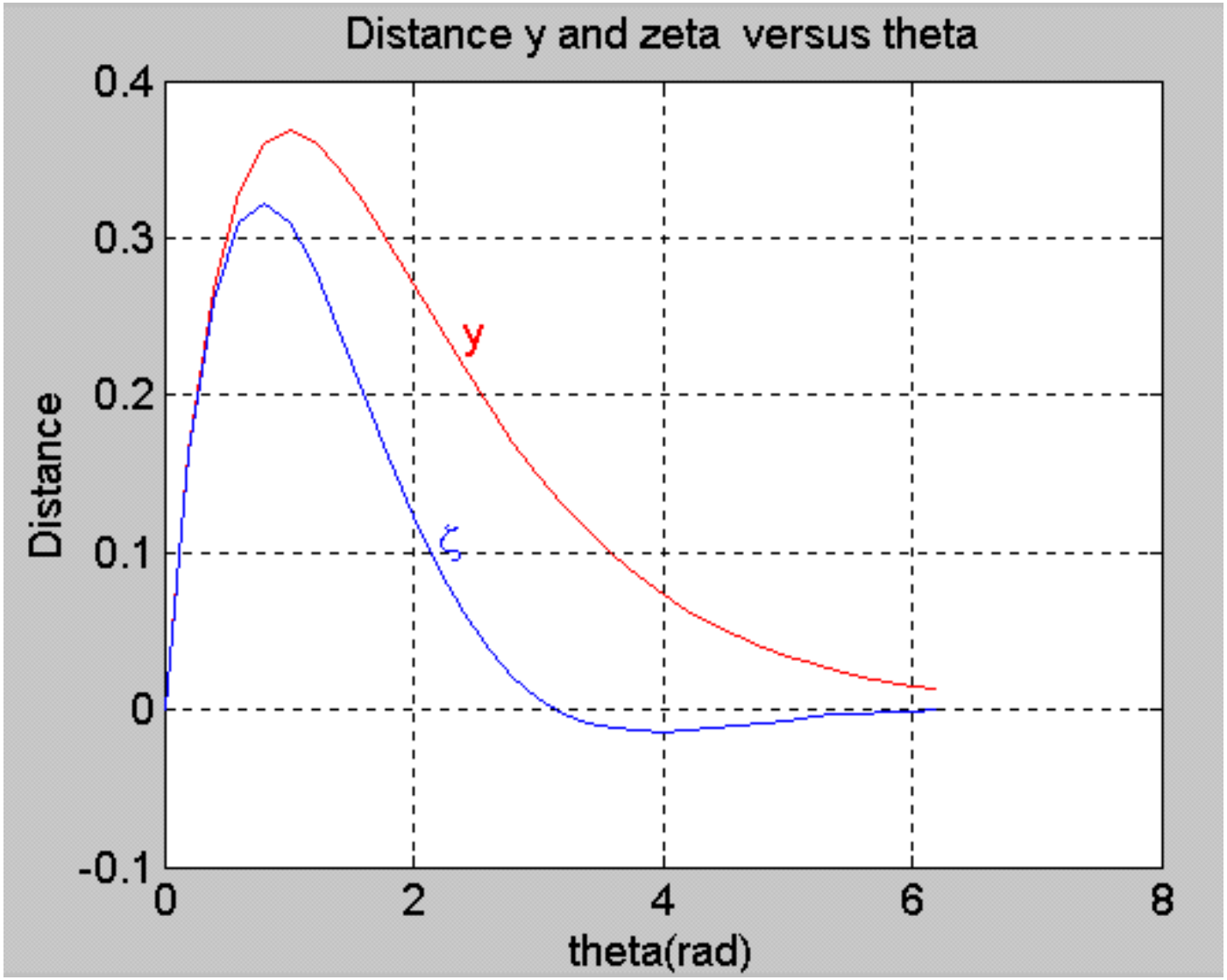
```
text(x , y , ' string ')
```

where `x,y` are the coordinates where the string starts
`string` is the required string

# Example

```
clear ; clf ;
theta = 0:0.2:2*pi ;
y = theta.*exp(-theta) ;
z = sin(theta).*exp(-theta) ;
plot(theta , y , 'r' , theta , z , 'b')
xlabel('theta(rad)') ; ylabel('Distance')
title('Distance y and zeta  versus theta')
text(2.4 , 0.24 , 'y' , 'fontname' ,
'arial','fontsize' , 14 , 'color', 'r')
text(2.2 , 0.11 , 'z' , 'fontname' ,
'symbol','fontsize' , 14 , 'color' , 'b')
grid
```

# Example


Distance y and zeta versus theta
0.4
0.3
0.2
0.1
0
-0.1
Distance
y
ζ
0
2
4
6
8
theta(rad)

## Plotting Polynomials

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

The function polyval(p,x) evaluates the polynomial p at specified values of its independent variable x.

For example, to plot the polynomial

$$3x^5 + 2x^4 - 100x^3 + 2x^2 - 7x + 90$$

over the range

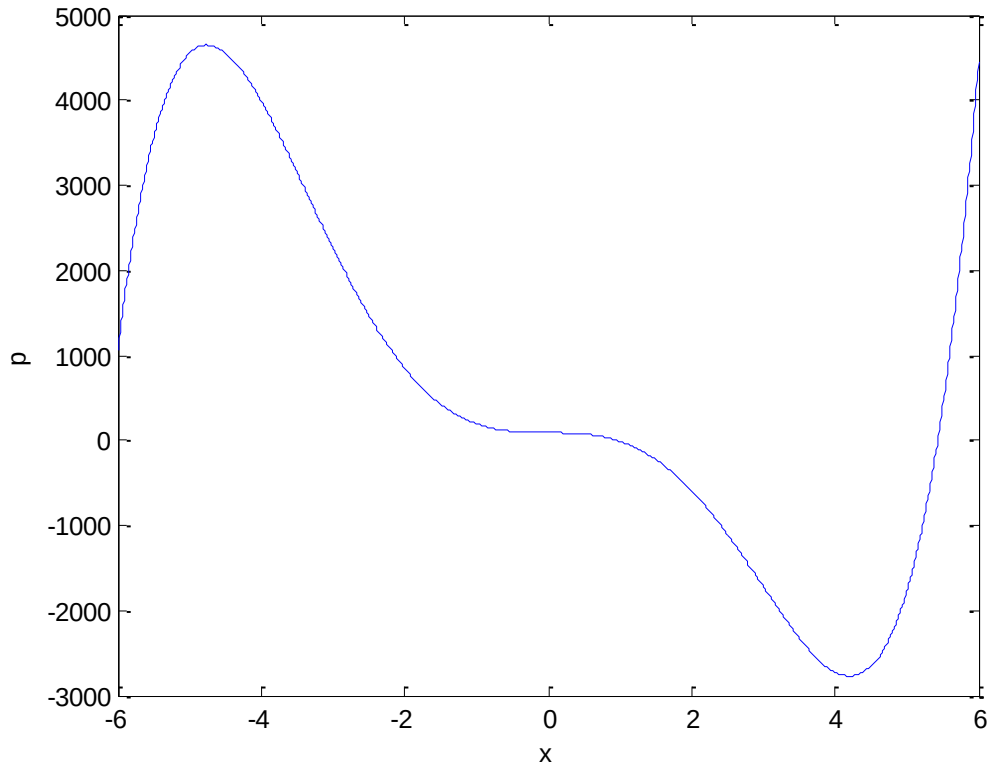
$$-6 \leq x \leq 6$$

with a spacing of 0.01.

M-file

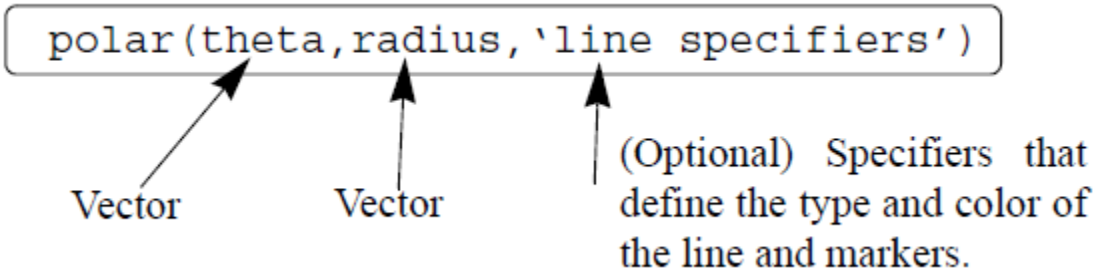
```
>>x = -6:0.01:6;
>>p = [3,2,-100,2,-7,90];
>>plot(x,polyval(p,x)),xlabel('x'),ylabel('p')
```

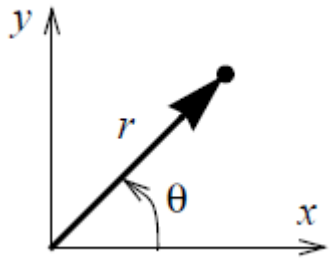
```
1 —    x = -6:0.01:6;
2 —    p = [3,2,-100,2,-7,90];
3 —    plot(x,polyval(p,x))
4 —    xlabel('x')
5 —    ylabel('p')
```



## POLAR PLOTS

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

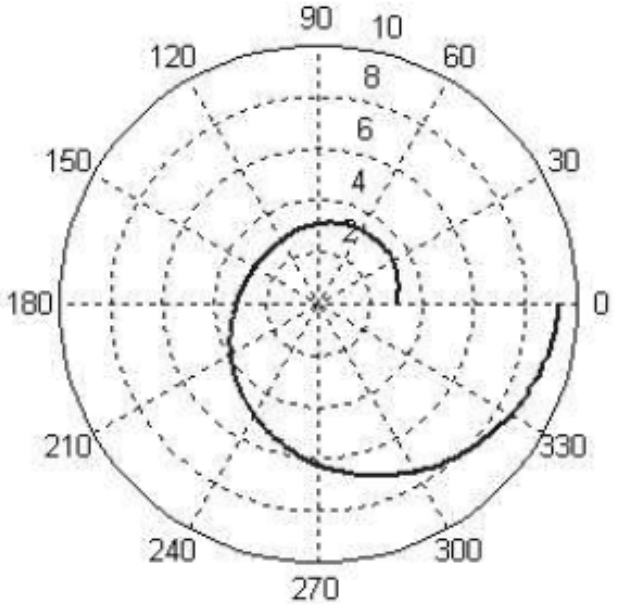



where theta and radius are vectors whose elements define the coordinates of the points to be plotted. The polar command plots the points and draws the polar grid. The line specifiers are the same as in the plot command.

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

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

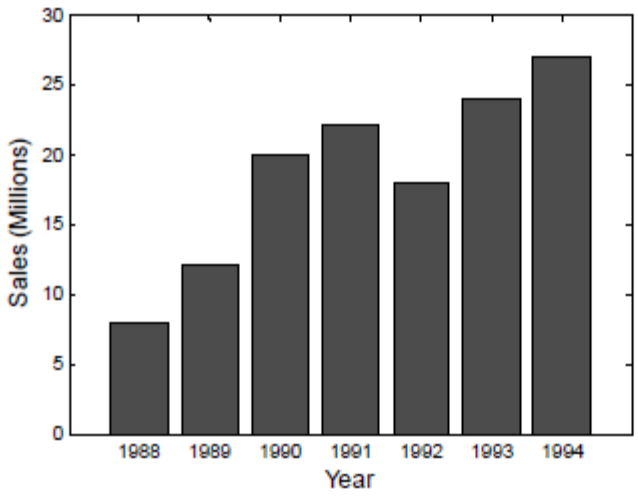


# Bar Plots

## Vertical Bar Plot

Function format:

`bar(x,y)`



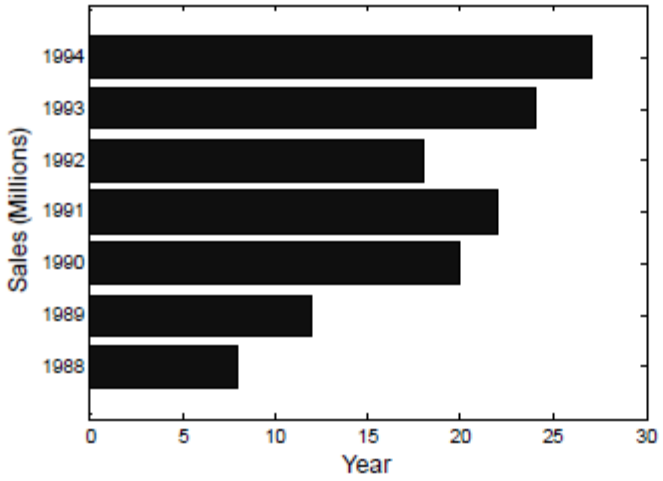

```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
bar(yr,sle,'r')
xlabel('Year')
ylabel('Sales (Millions)')
```

The bars are in red.

## Horizontal Bar Plot

Function format:

`barh(x,y)`




```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
barh(yr,sle)
xlabel('Sales (Millions)')
ylabel('Year')
```

# Pie Plot

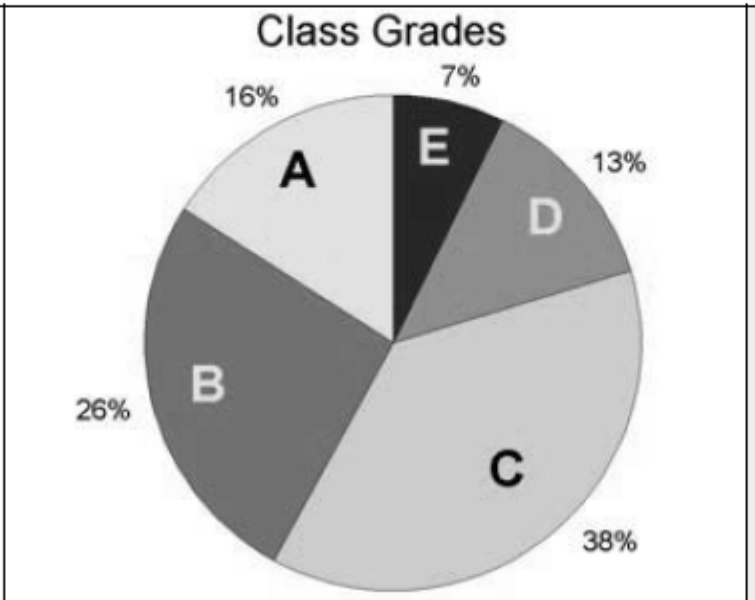
Pie charts are useful for visualizing the relative sizes of different but related quantities. For example, the table below shows the grades that were assigned to a class. The data is used to create the pie chart that follows.

| Grade | A | B | C | D | E |
|---|---|---|---|---|---|
| Number of students | 11 | 18 | 26 | 9 | 5 |

Pie Plot

Function format:

```
pie(x)
```




```
grd=[11 18 26 9 5];
pie(grd)
title('Class Grades')
```

MATLAB draws the sections in different colors. The letters (grades) were added using the Plot Editor.

## Logarithmic Plots

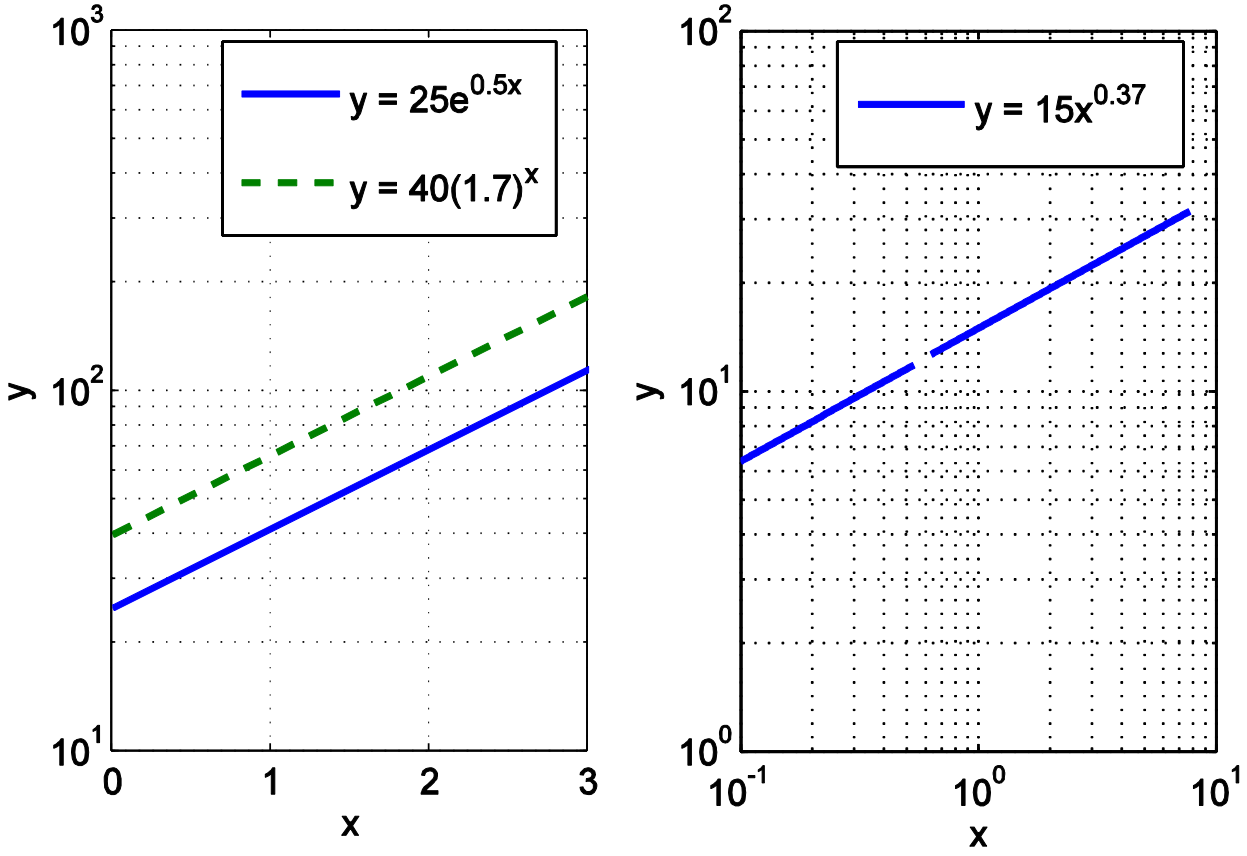
It is important to remember the following points when using log scales:

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.

## Logarithmic Plot Example

```
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
```

# Logarithmic Plots



**Specialized plot commands.** Table 5.2-3, page 236

| 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`. |

## Three-Dimensional 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 curve as the parameter t is varied over some range:

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

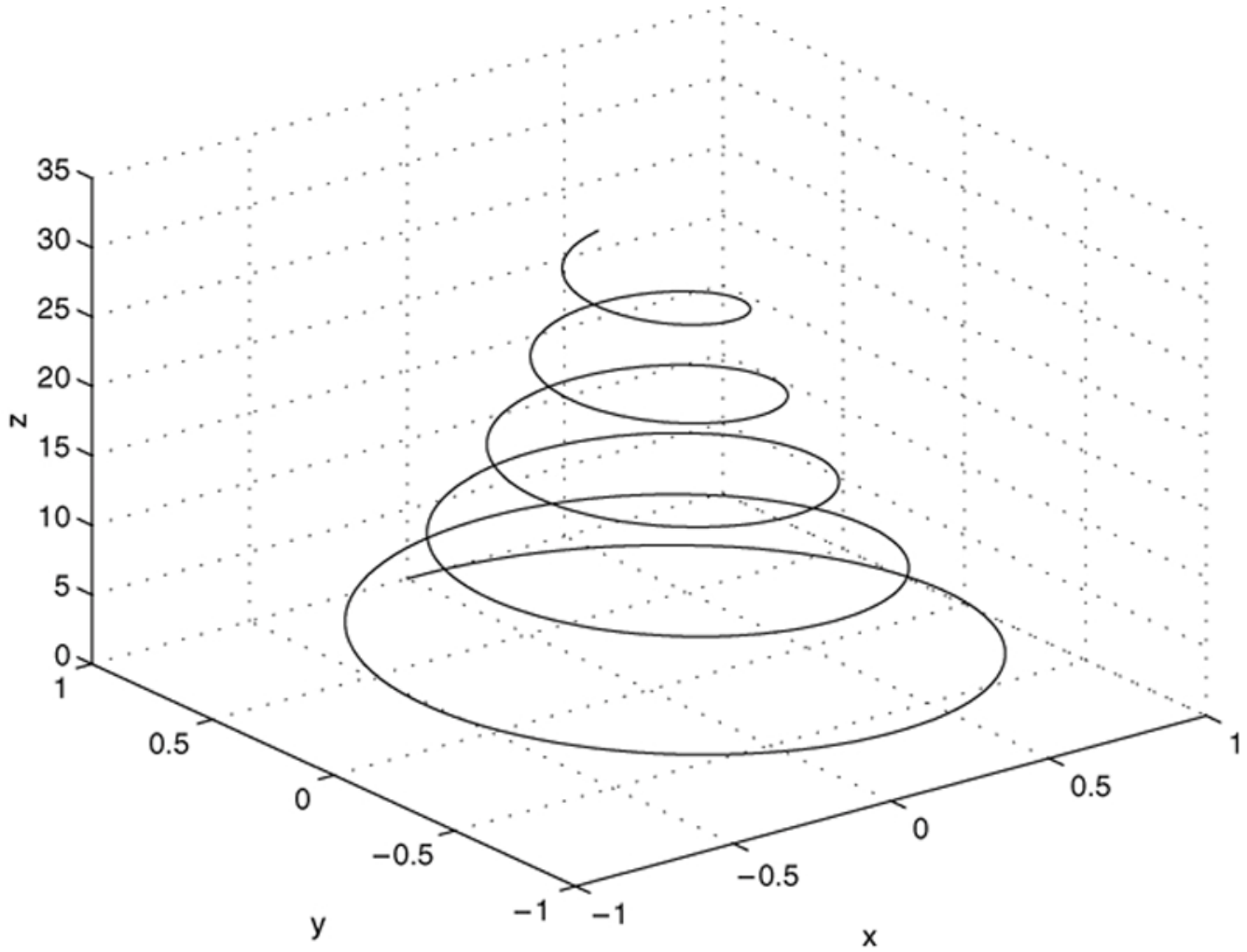
let t vary from t = 0 to t = 10$\pi$,

## Three-Dimensional Line Plots:

M-File

```
>>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
```

**The curve $x = e^{-0.05t} \sin t$, $y = e^{-0.05t} \cos t$, $z = t$ plotted with the `plot3` function.** Figure 5.4–1, page 247.

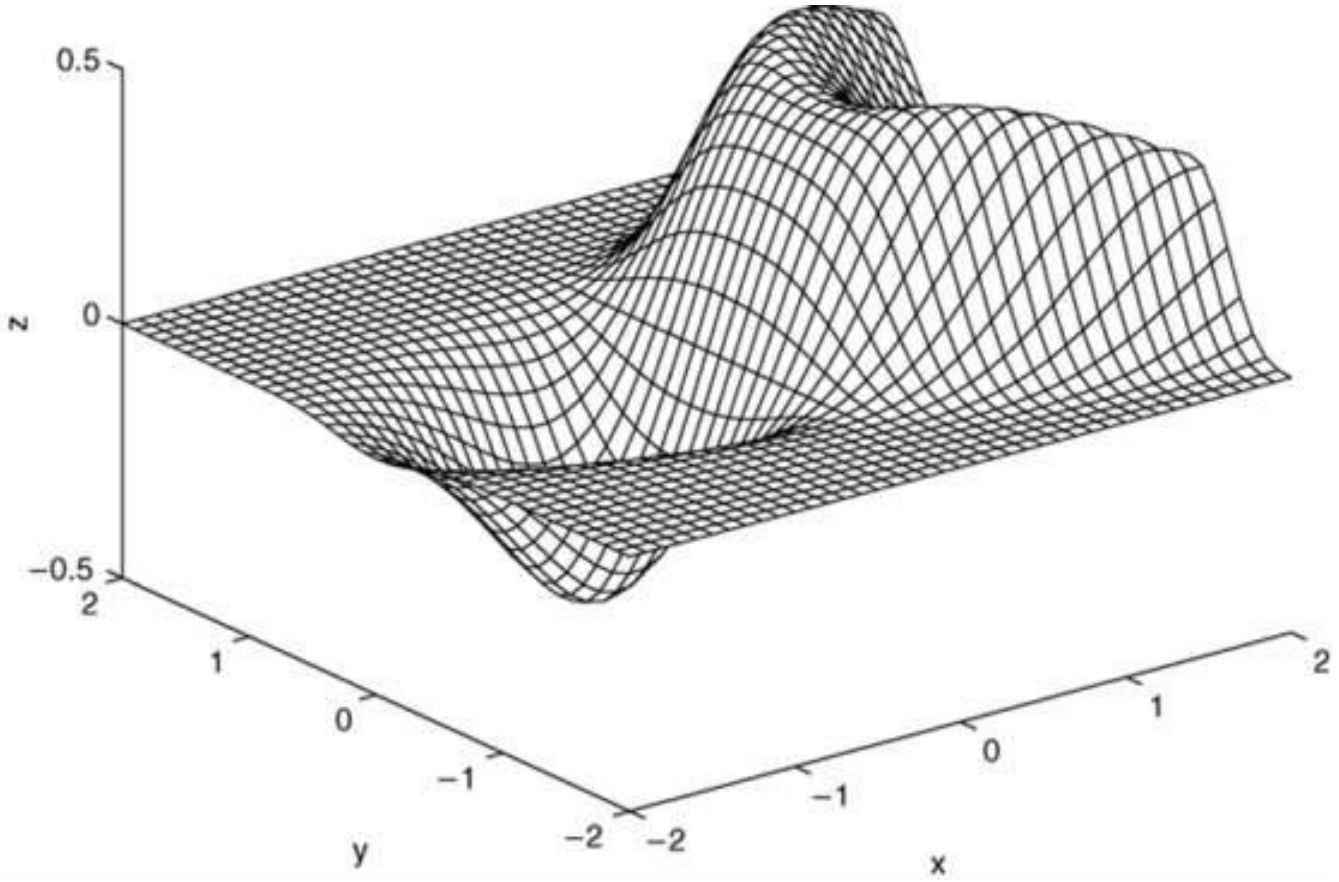


## Surface Plots:

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

**A plot of the surface $z = xe^{-[(x-y^2)^2+y^2]}$ created with the `mesh` function.** Figure 5.8–2

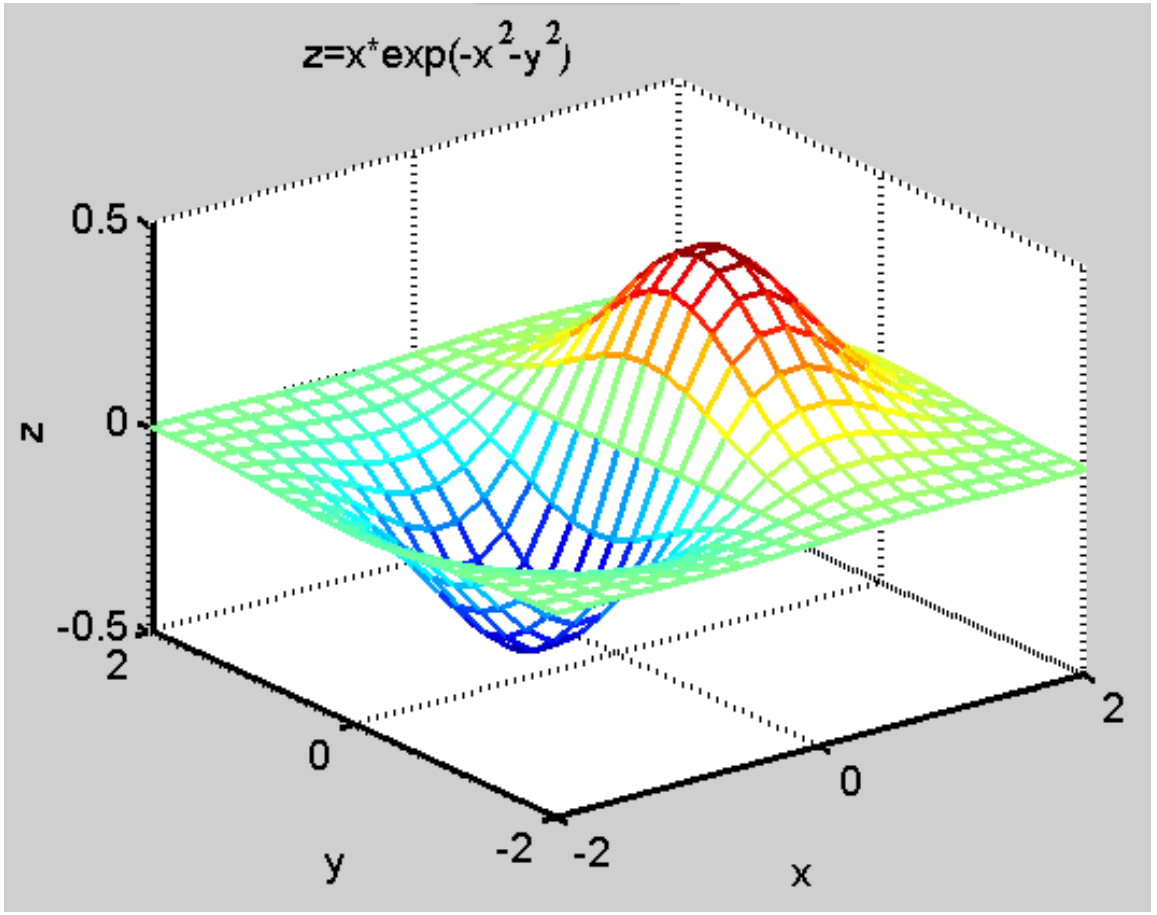


## Example

```
clear ; clf ;
xa = -2:0.2:2;
ya = -2:0.2:2;
[x,y] = meshgrid(xa,ya) ;
z = x.*exp(-x.^2-y.^2) ;
mesh(x,y,z)
title('z=x*exp(-x^2-y^2)')
xlabel('x') ; ylabel('y') ; zlabel('z') ;
```

**Example**

**A surface plot of the surface $z = xe^{-[(x-y^2)^2+y^2]}$ created with the `mesh` function.**



# Contour

Contour of a function in a two-dimensional array can be plotted by:

```
Contour(x , y , z , level)
```

z is the 2-D array of the function

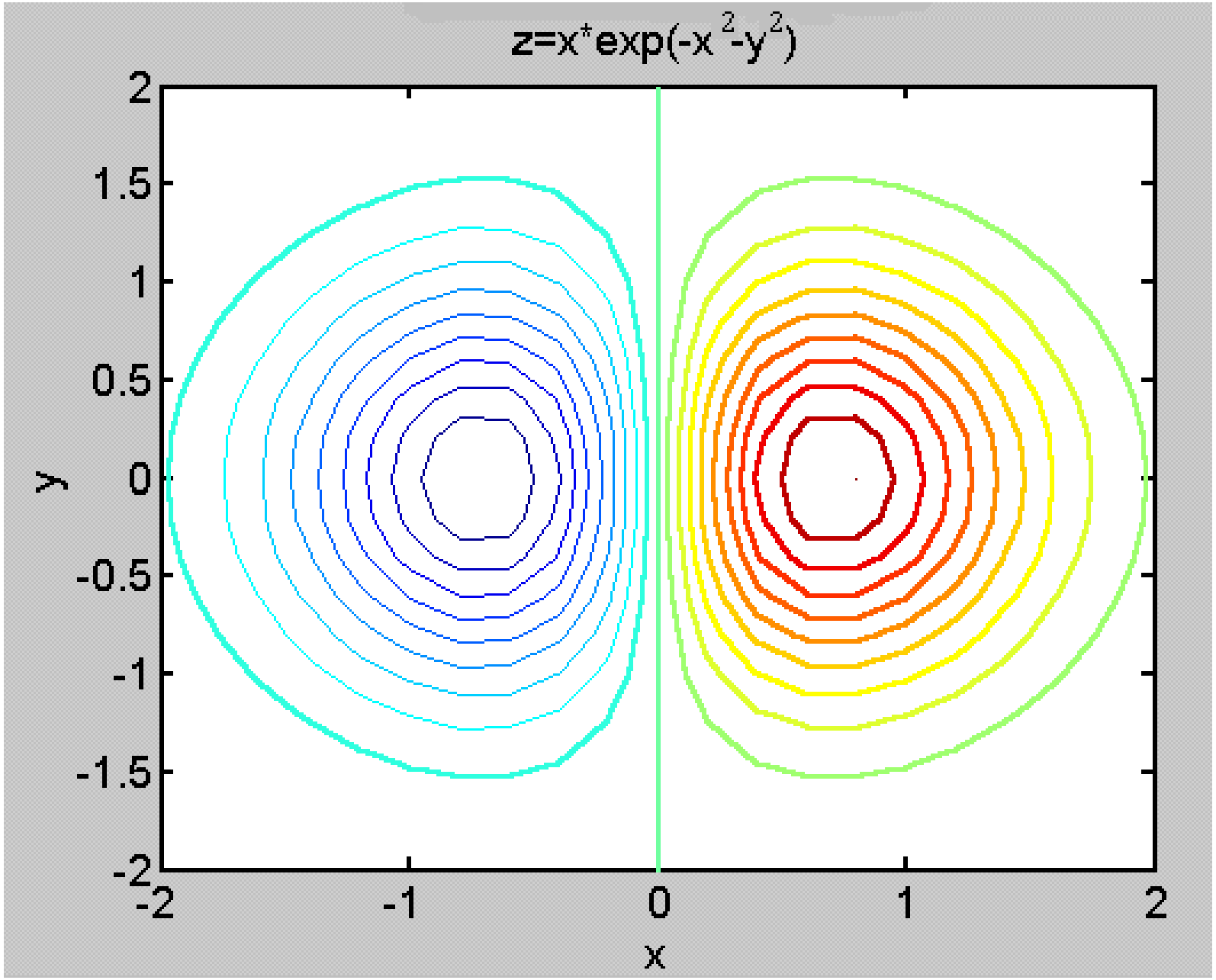
x and y are the coordinates in one or 2D arrays

Level is a vector containing the levels

The contour level is determined by dividing the minimum and maximum values of z into k-1 intervals

# The following is a script for the contour plot:

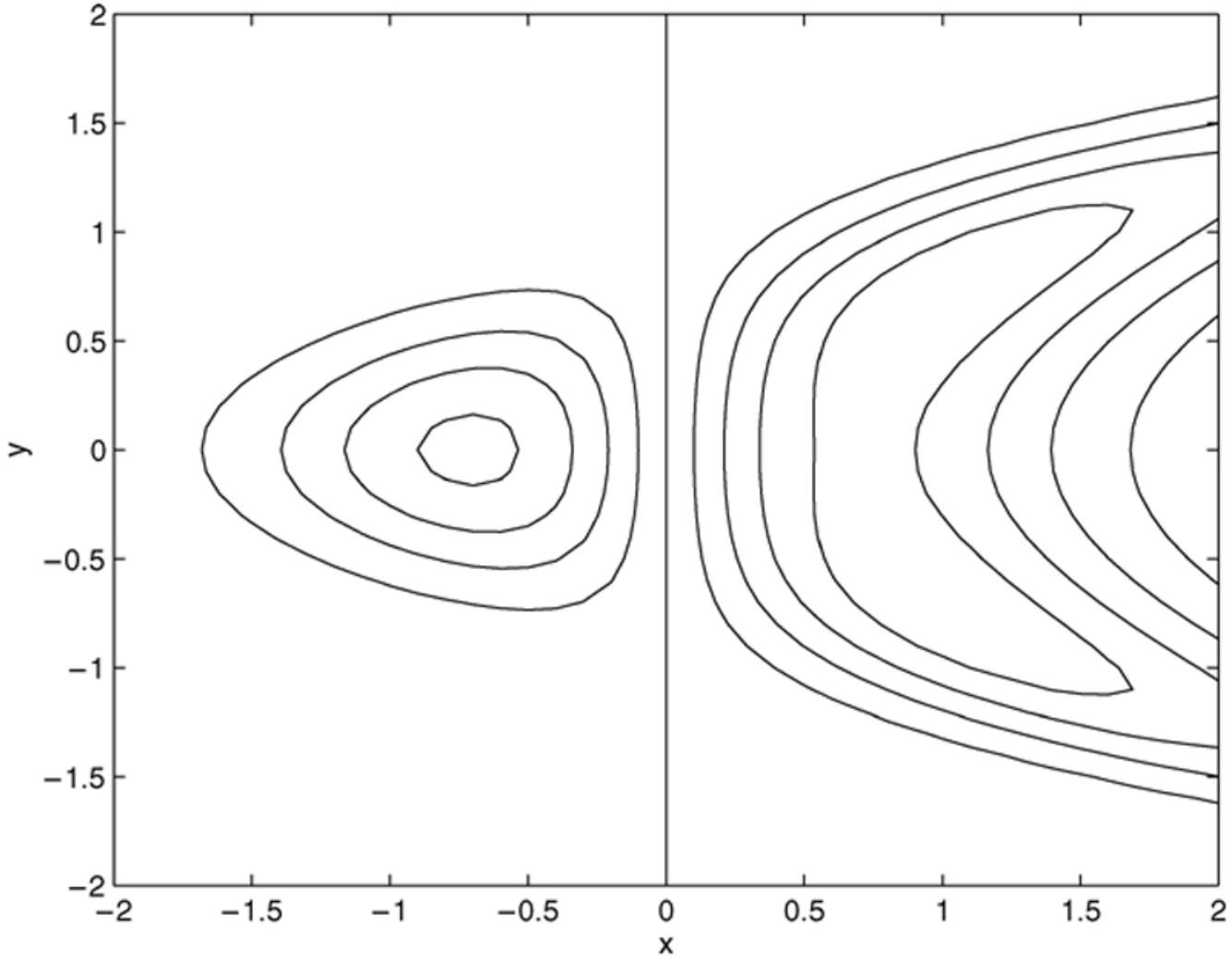
```
clear ; clf ;

xa = -2 : 0.2 : 2  ;  ya = -2 : 0.2 : 2 ;

[x,y] = meshgrid(xa,ya) ;

z = x.*exp(-x.^2-y.^2) ;  zmax=max(max(z)) ;

zmin=min(min(z)) ; k=21 ;

dz=(zmax-zmin)/(k-1) ; level=zmin : dz : zmax ;
h=contour(x , y , z , level) ;

title('z=x*exp(-x^2-y^2)') ;

xlabel('x') ; ylabel('y') ;
```


z=x*exp(-x^2-y^2)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
y
-2
-1
0
1
2
x

The following session generates the contour plot of the function whose surface plot is shown in Figure 5.8–2; 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. This plot appears in Figure 5.4–3, page 249.

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

**A contour plot of the surface $z = xe^{-[(x-y^2)^2+y^2]}$ created with the `contour` function.**



Contour labels may be automatically annotated by : clabel(h)

h is the name of contour

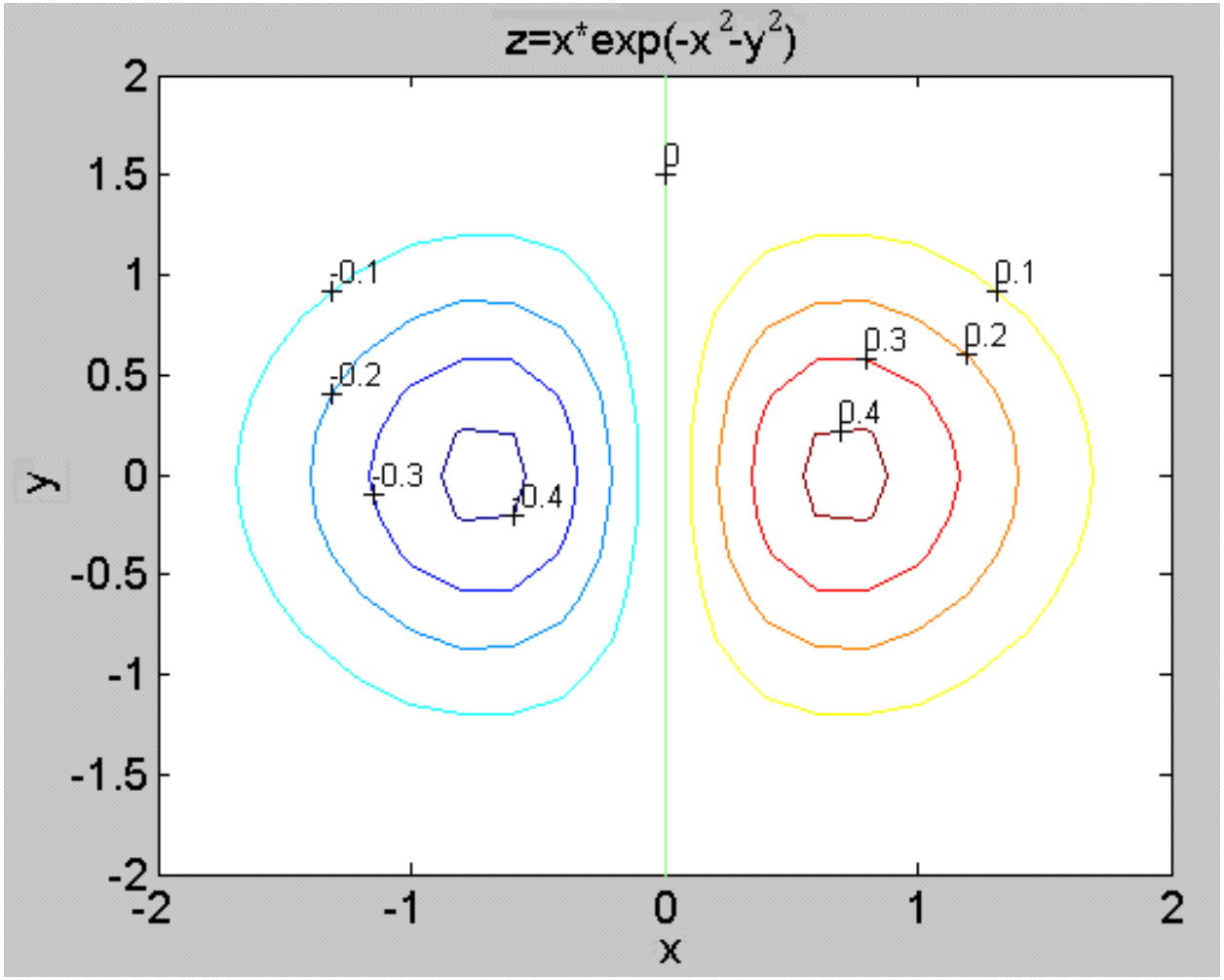
Contour labels may also be placed manually using the mouse and by :

clabel(h,'manual')

Below is a script with clabel :

## Example

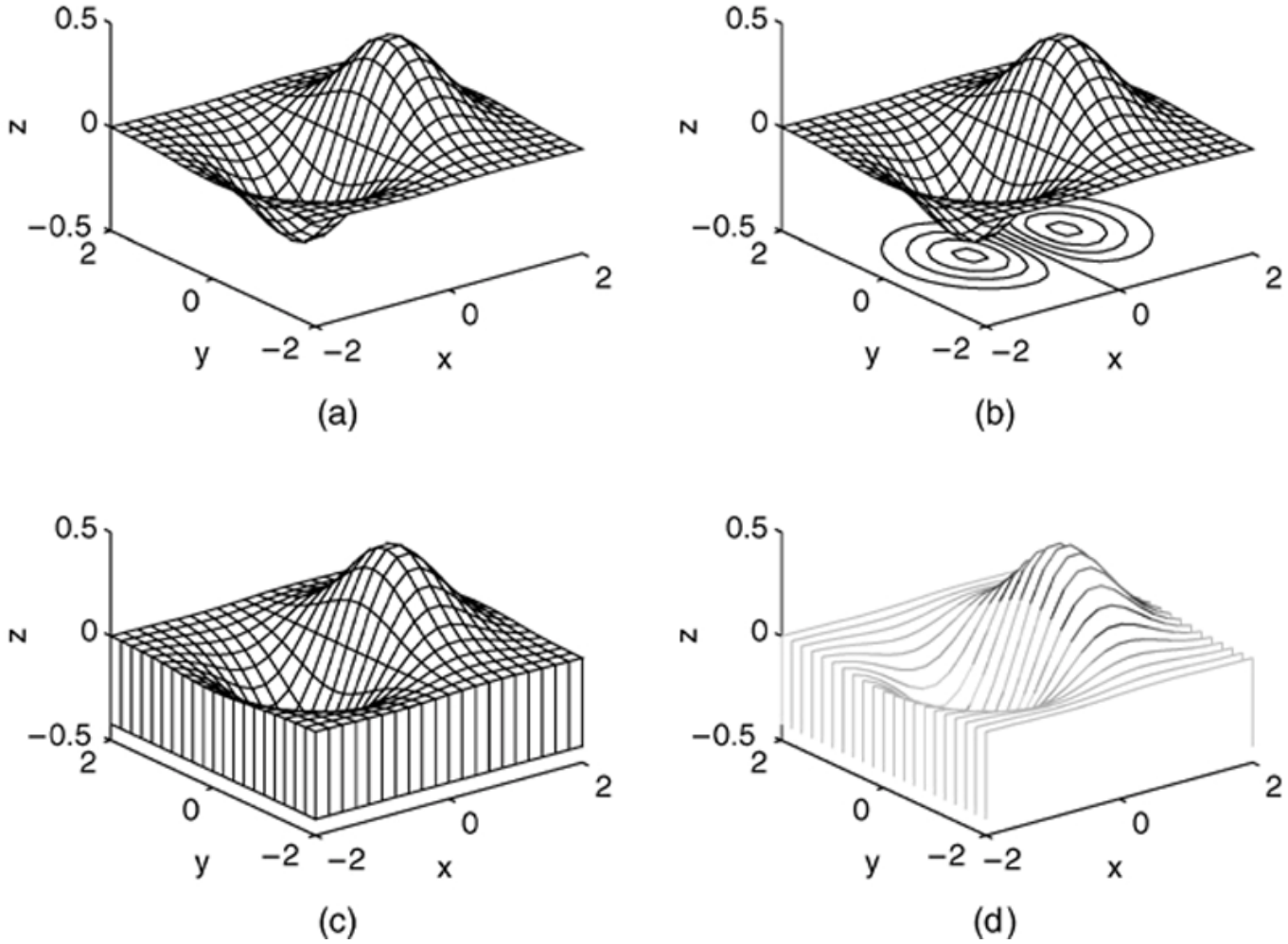
```
clear ; clf ;

xa = -2:0.2:2; ya = -2:0.2:2;

[x,y] = meshgrid(xa,ya) ;

z = x.*exp(-x.^2-y.^2)  ;  zmax=0.5 ;

zmin=-0.5 ; k=11 ;

dz=(zmax-zmin)/(k-1) ;  level=zmin : dz :
zmax ; h=contour(x,y,z,level) ;
clabel(h,'manual') ;

title('z=x*exp(-x^2-y^2)')

xlabel('x') ;  ylabel('y')
```


z=x*exp(-x^2-y^2)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1
0
1
2
x
y
0
0.1
0.2
0.3
0.4
0.1
0.2
0.3
0.4

# **Three-dimensional plotting functions.** Table 5.4–1, page 250.

| 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`**.**
Figure 5.4–4, page 250.



# Vector plot

Quantities at grid points sometimes are required to be plotted in a vector form

The vectors at grid points may be plotted by the quiver command:

```
quiver(x , y , u , v , s)
```

x Array for x-coordinate

y Array for y-coordinate

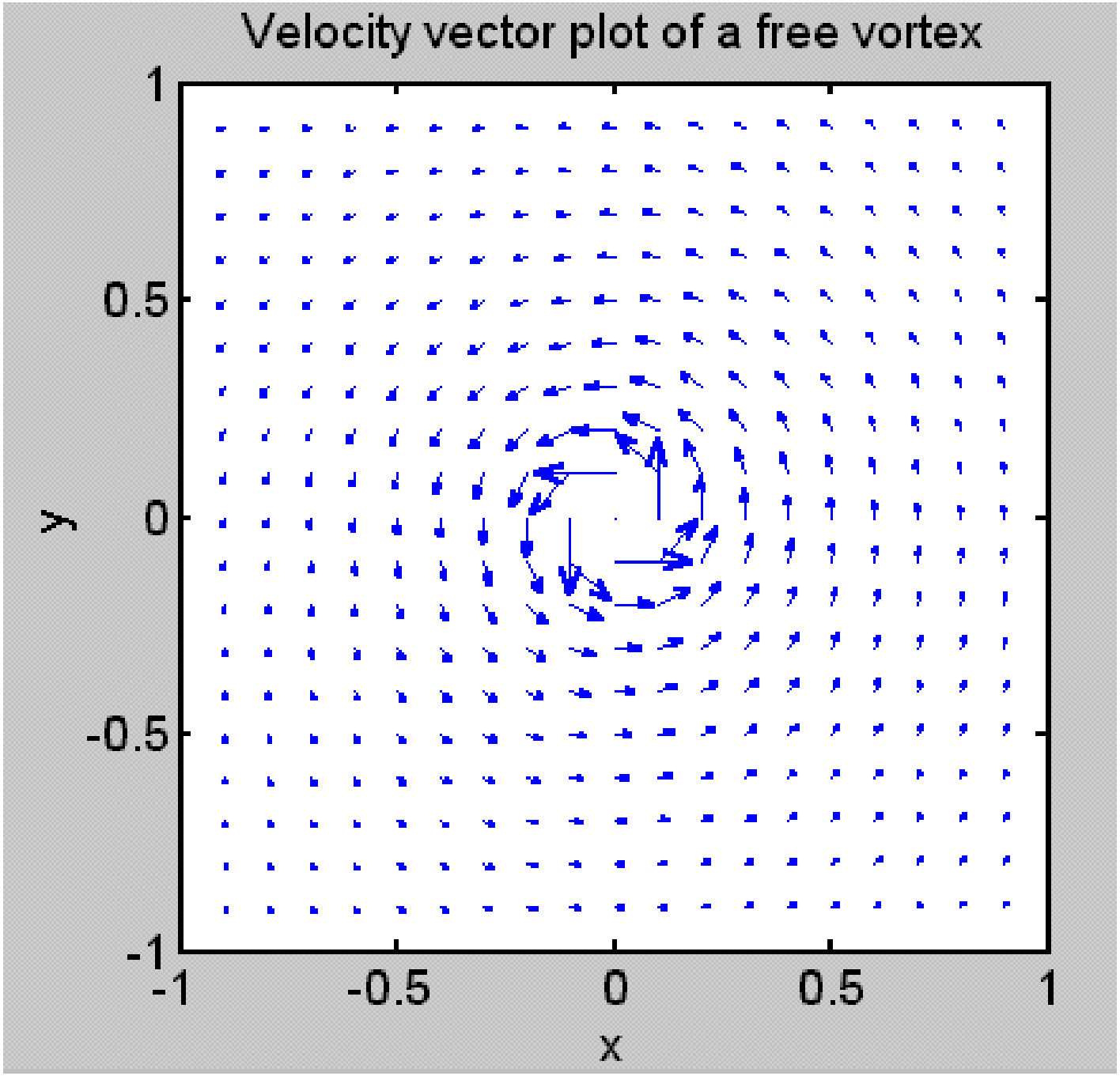
u Array of the function u

v Array of the function v

s Scale factor for vector adjustment

```
clear , clf
xmax= 0.9 ; xmin= -xmax ; dx= 0.1 ;
ymax= xmax ; ymin= -ymax ; dy= dx ;
xm = xmin : dx : xmax   ;
ym = ymin : dy : ymax  ;
[x,y] = meshgrid(xm,ym) ;
u = -0.1*(2*y)./(x.^2+y.^2)  ;
v = -0.1*(-2*x)./(x.^2+y.^2) ;
quiver(x,y,u,v,1.5) ;
title('Velocity vector plot of  free vortex')
xlabel('x') ; ylabel('y')
axis('square')
```

This script is an example of a vector plot for a free vortex plane potential flow


Velocity vector plot of a free vortex
1
0.5
0
-0.5
-1
y
-1
-0.5
0
0.5
1
x

# Polynomial fit to a given data

A polynomial is required to fit the following data:

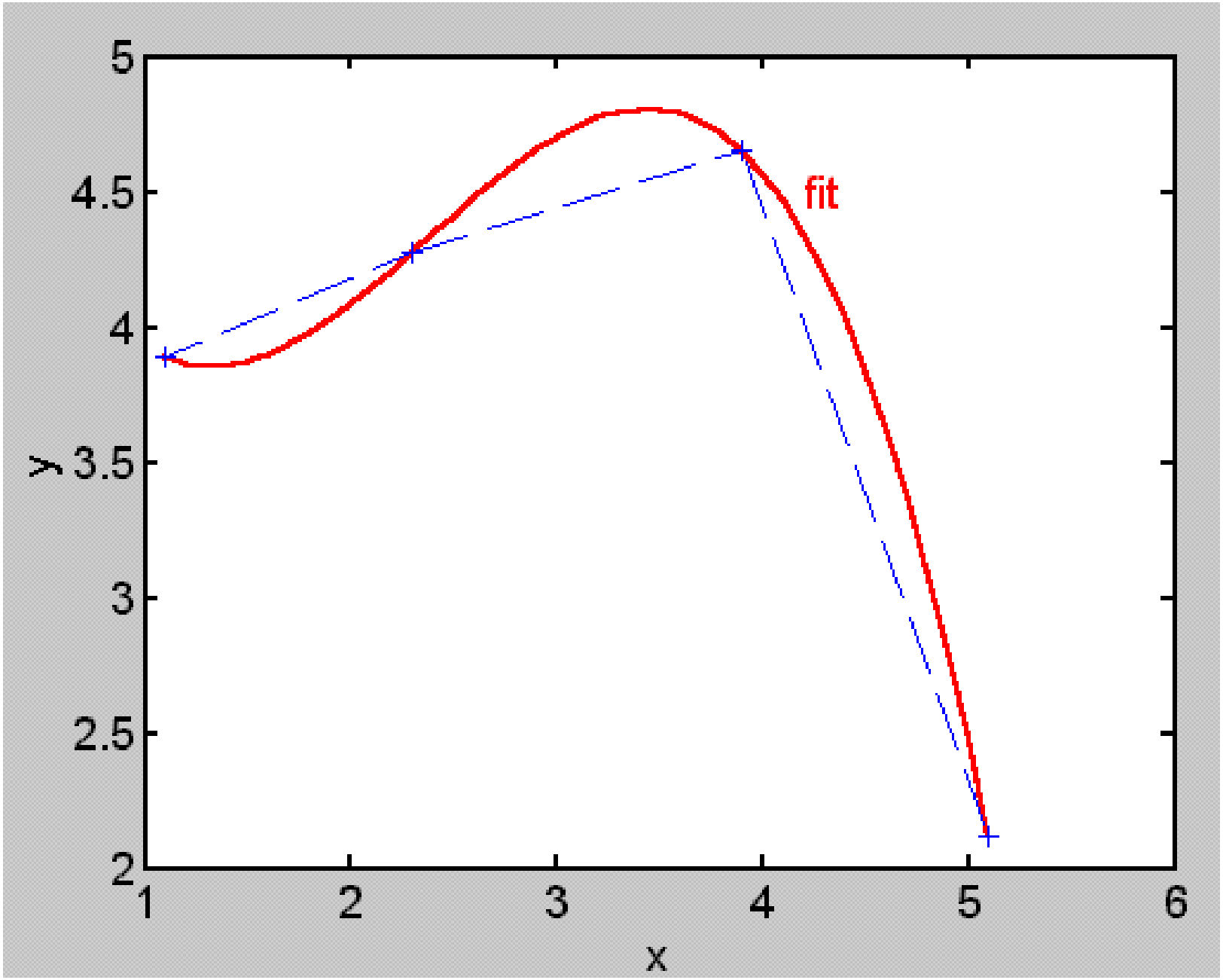
| x | y |
|---|---|
| 1.1 | 3.887 |
| 2.3 | 4.276 |
| 3.9 | 4.651 |
| 5.1 | 2.117 |

The required script for fitting a polynomial and plotting the data is given in the next slide

```
clear , clf  , hold off
x= [1.1,2.3,3.9,5.1] ;
y= [3.887,4.276,4.651,2.117] ;
a= polyfit(x,y,length(x)-1)
xi= 1.1:0.1:5.1 ;
yi= polyval(a,xi) ;
plot(xi,yi,'r')
text(4.2,4.5,'fit','fontsize',[14],'color','r');
xlabel('x') ; ylabel('y')
hold on
plot(x,y,'--',x,y,'+b')
```

The output polynomial coefficients are printed in the command window as:

```
a =

   -0.2015    1.4385   -2.7477    5.4370
```


5
4.5
4
3.5
3
2.5
2
y
1
2
3
4
5
6
x
fit

# Consider the following data:

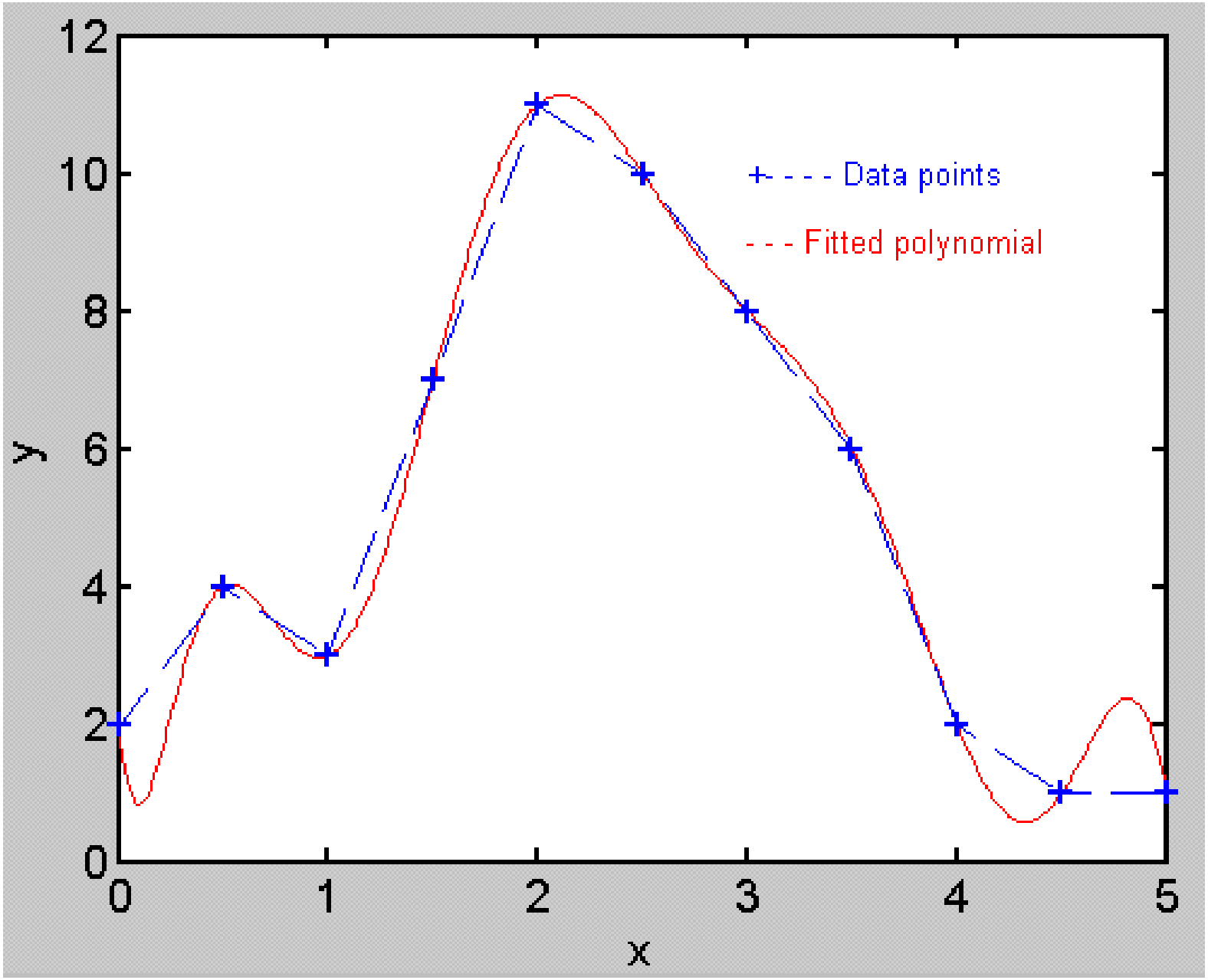
| x | y |
|---|---|
| 0 | 2 |
| 0.5 | 4 |
| 1 | 3 |
| 1.5 | 7 |
| 2 | 11 |
| 2.5 | 10 |
| 3 | 8 |
| 3.5 | 6 |
| 4 | 2 |
| 4.5 | 1 |
| 5 | 1 |

It is required to fit a polynomial to the data shown and to plot the results

```
clear , clf , hold off
a=[0   2 ;  0.5   4  ;  1    3 ; 1.5    7 ; 2   11  ;  2.5   10 ;...
     3   8 ;  3.5   6 ; 4   2  ;  4.5   1  ;  5   1] ;
x= a(:,1) ; y = a(:,2) ;
b=polyfit(x,y,length(x)-1)
xi=0:0.01:5  ;  yi=polyval(b,xi)  ;
plot(xi,yi,'r')
hold on
plot(x,y,'- -',x,y,'+b')
xlabel('x') ; ylabel('y') ;
text(3,10,'+- - - - Data points', ...
                'color','b','fontsize',[10])
text(3,9,'+- - - Fitted polynomial',...
'color','r','fontsize',[10])
```

The polynomial coefficients are as follows:

```
b =

    Columns 1 through 5

0.0248   -0.6787    7.8772  -50.5799
195.9348

    Columns 6 through 9

-467.8019  673.3763  -545.9602
217.2868

    Columns 10 through 11

 -28.4794    2.0000
```


12
10
8
6
4
2
0
y
0
1
2
3
4
5
x
Data points
Fitted polynomial

# Requirements for a correct plot

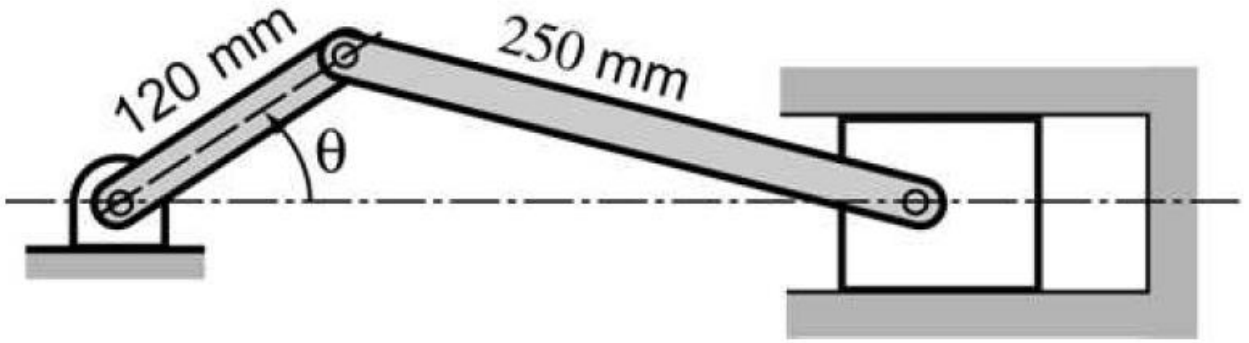
1. Each axis must be labeled with the name of the quantity being plotted *and its units!* If two or more quantities having different units are plotted (such as when in a plot of both speed and distance versus time), indicate the units in the axis label, if there is room, or in the legend or labels for 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, use different line types, 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.

# Requirements for a correct plot

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.
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.

# Sample Problem: Crank-Slider Mechanism

The piston-rod-crank mechanism is used in many engineering applications. In the mechanism shown in the following figure, the crank is rotating at a constant speed of 500 rpm.




Calculate and plot the position, velocity, and acceleration of the piston for one revolution of the crank. Make the three plots on the same page. Set $\theta = 0°$ when t = 0.

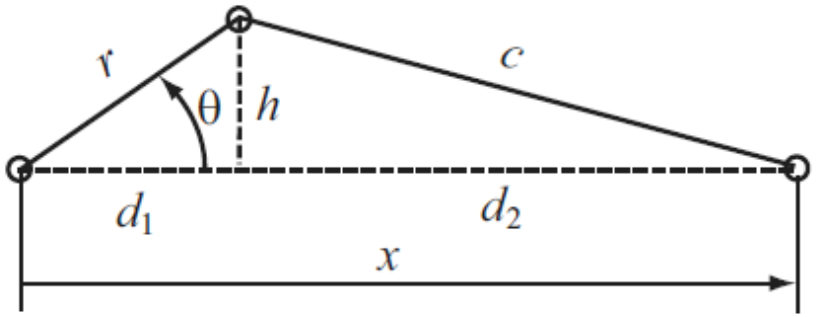
# Sample Problem: Crank-Slider Mechanism

The crank is rotating with a constant angular velocity $\dot{\theta}$. This means that if we set $\theta = 0°$ when $t = 0$, then at time $t$ the angle $\theta$ is given by $\theta = \dot{\theta}t$, and means that $\ddot{\theta} = 0$ at all times.

The distances $d_1$ and $h$ are given by:

$$d_1 = r\cos\theta \quad \text{and} \quad h = r\sin\theta$$

With $h$ known, the distance $d_2$ can be calculated using the Pythagorean Theorem:




$$d_2 = (c^2 - h^2)^{1/2} = (c^2 - r^2\sin^2\theta)^{1/2}$$

# Sample Problem: Crank-Slider Mechanism

The position $x$ of the piston is then given by:

$$x = d_1 + d_2 = r\cos\theta + (c^2 - r^2\sin^2\theta)^{1/2}$$

The derivative of $x$ with respect to time gives the velocity of the piston:

$$\dot{x} = -r\dot{\theta}\sin\theta - \frac{r^2\dot{\theta}\sin 2\theta}{2(c^2 - r^2\sin^2\theta)^{1/2}}$$

The second derivative of $x$ with respect to time gives the acceleration of the piston:

$$\ddot{x} = -r\dot{\theta}^2\cos\theta - \frac{4r^2\dot{\theta}^2\cos 2\theta(c^2 - r^2\sin^2\theta) + (r^2\dot{\theta}\sin 2\theta)^2}{4(c^2 - r^2\sin^2\theta)^{3/2}}$$

In the equation above, $\ddot{\theta}$ was taken to be zero.

# Sample Problem: Crank-Slider Mechanism

A MATLAB program (script file) that calculates and plots the position, velocity, and acceleration of the piston for one revolution of the crank is shown below.

```
THDrpm=500; r=0.12; c=0.25;
THD=THDrpm*2*pi/60;
tf=2*pi/THD;
t=linspace(0,tf,200);
TH=THD*t;
d2s=c^2-r^2*sin(TH).^2;
x=r*cos(TH)+sqrt(d2s);
xd=-r*THD*sin(TH)-(r^2*THD*sin(2*TH))./(2*sqrt(d2s));
```

Define $\dot{\theta}$, $r$, and $c$.

Change the units of $\dot{\theta}$ from rpm to rad/s.

Calculate the time for one revolution of the crank.

Create a vector for the time with 200 elements.

Calculate $\theta$ for each $t$.

Calculate $d_2$ squared for each $\theta$.

Calculate $x$ for each $\theta$.

```
xdd=-r*THD^2*cos(TH)-(4*r^2*THD^2*cos(2*TH).*d2s+
(r^2*sin(2*TH)*THD).^2)./(4*d2s.^(3/2));
subplot(3,1,1)
plot(t,x)
grid
xlabel('Time (s)')
ylabel('Position (m)')
subplot(3,1,2)
plot(t,xd)
grid
xlabel('Time (s)')
ylabel('Velocity (m/s)')
subplot(3,1,3)
plot(t,xdd)
grid
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')
```

Calculate $\dot{x}$ and $\ddot{x}$ for each θ.

Plot $x$ vs. $t$.

Format the first plot.

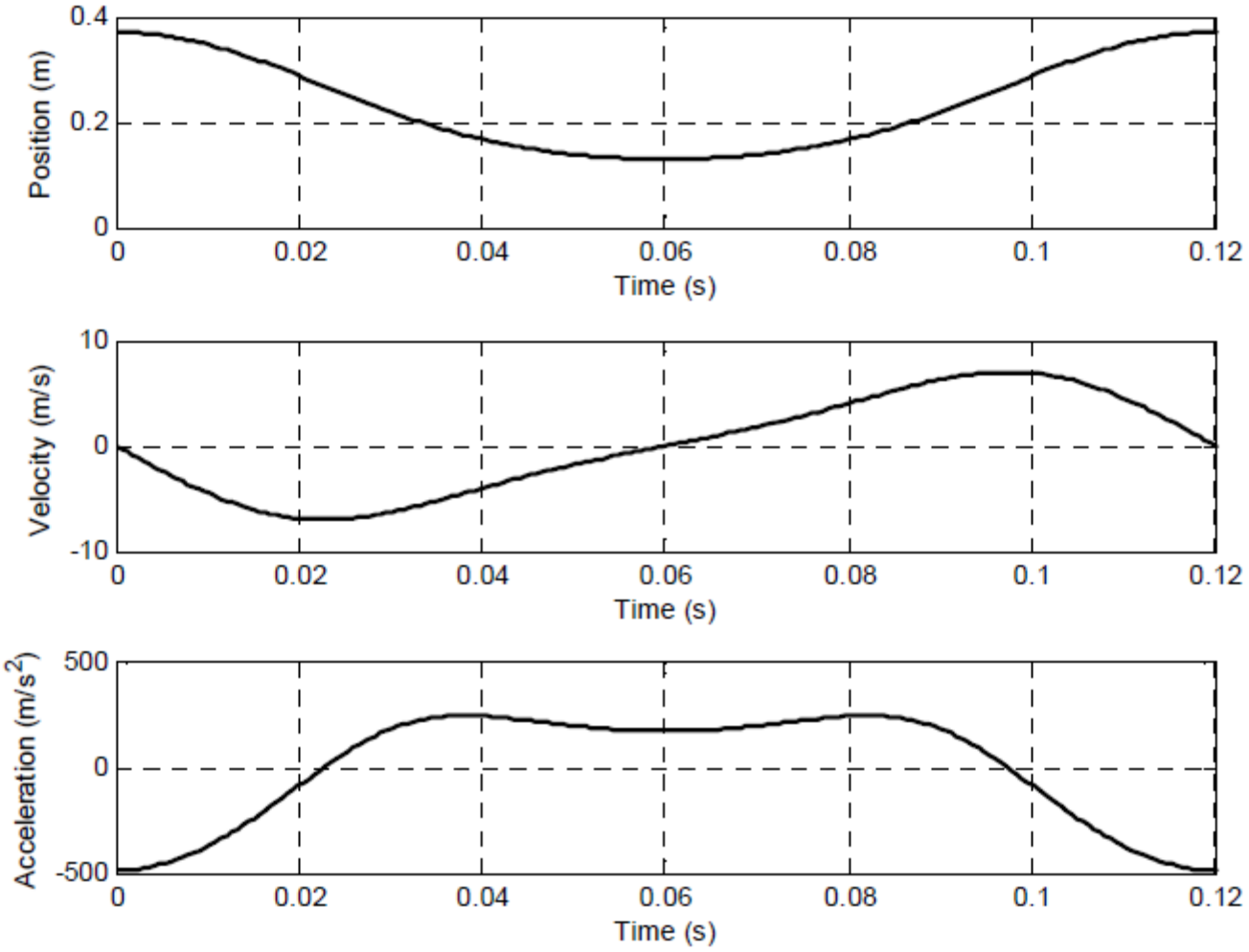
Plot $\dot{x}$ vs. $t$.

Format the second plot.

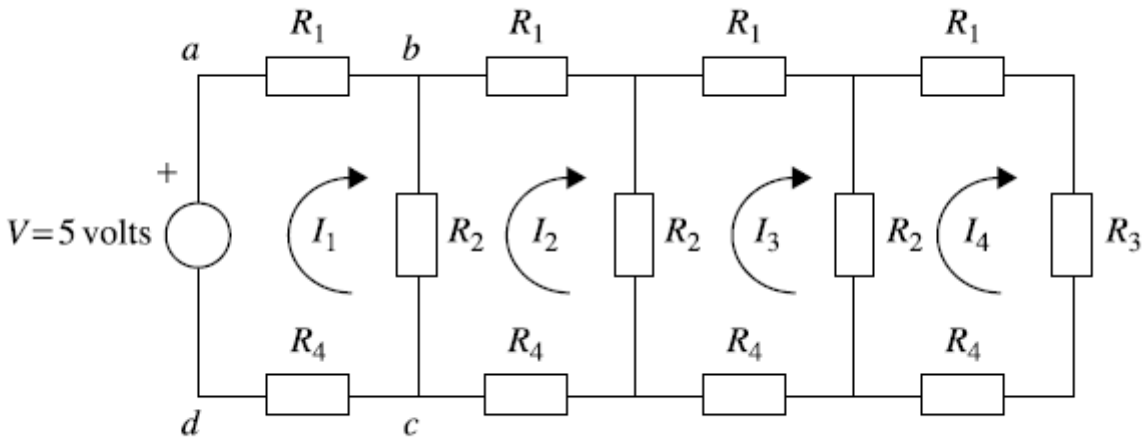
Plot $\ddot{x}$ vs. $t$.

Format the third plot.

# Sample Problem: Crank-Slider Mechanism



# Sample Problem: Electrical network.




$$V_{bc} = R_2(I_1 - I_2)$$

where $R_2$ is the value of the resistor in the link connecting $b$ to $c$. Kirchhoff's voltage law states that the algebraic sum of the voltages around a loop is zero. Applying these laws to the circuit $abcd$ of Fig. 2.1 we have

$$V_{ab} + V_{bc} + V_{cd} = V$$

Substituting the product of current and resistance for voltage gives

$$R_1 I_1 + R_2(I_1 - I_2) + R_4 I_1 = V$$

# Sample Problem: Electrical network.

We can repeat this process for each loop to obtain the following four equations:

$$\begin{aligned}(R_1+R_2+R_4)\,I_1 - R_2 I_2 &= V\\ (R_1+2R_2+R_4)\,I_2 - R_2 I_1 - R_2 I_3 &= 0\\ (R_1+2R_2+R_4)\,I_3 - R_2 I_2 - R_2 I_4 &= 0\\ (R_1+R_2+R_3+R_4)\,I_4 - R_2 I_3 &= 0\end{aligned} \tag{2.1}$$

Letting $R_1 = R_4 = 1\ \Omega$, $R_2 = 2\ \Omega$, $R_3 = 4\ \Omega$ and $V = 5$ volts, (2.1) becomes

$$\begin{aligned}4I_1 - 2I_2 &= 5\\ -2I_1 + 6I_2 - 2I_3 &= 0\\ -2I_2 + 6I_3 - 2I_4 &= 0\\ -2I_3 + 8I_4 &= 0\end{aligned}$$

This is a system of linear equations in four variables, $I_1, \ldots, I_4$. In matrix notation it becomes

$$\begin{bmatrix} 4 & -2 & 0 & 0\\ -2 & 6 & -2 & 0\\ 0 & -2 & 6 & -2\\ 0 & 0 & -2 & 8 \end{bmatrix} \begin{bmatrix} I_1\\ I_2\\ I_3\\ I_4 \end{bmatrix} = \begin{bmatrix} 5\\ 0\\ 0\\ 0 \end{bmatrix} \tag{2.2}$$

# Sample Problem: Electrical network.

This is a system of linear equations in four variables, $I_1, \ldots, I_4$. In matrix notation it becomes

$$\begin{bmatrix} 4 & -2 & 0 & 0 \\ -2 & 6 & -2 & 0 \\ 0 & -2 & 6 & -2 \\ 0 & 0 & -2 & 8 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.2}$$

This equation has the form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is a square matrix of known coefficients, in this case relating to the values of the resistors in the circuit. The vector $\mathbf{b}$ is a vector of known coefficients, in this case the voltage applied to each current loop. The vector $\mathbf{x}$ is the vector of unknown currents. Although this set of equations can be solved by hand, the process is time consuming and error prone. Using MATLAB we simply enter matrix `A` and vector `b` and use the command `A\b` as follows:

```
>> A = [4 -2 0 0;-2 6 -2 0;0 -2 6 -2;0 0 -2 8];
>> b = [5 0 0 0].';
>> x = A\b
```

# Sample Problem: Electrical network.

```
x =
    1.5426
    0.5851
    0.2128
    0.0532

>> A*x

ans =
    5.0000
   -0.0000
    0.0000
   -0.0000
```

# Sample Problem: Free Fall

In this section we are going to use MATLAB to not only investigate the problem of free fall with friction (or air resistance), we are going to use MATLAB to check the theoretical results found in the literature. This example deals with the application of MATLAB for computing and graphically presenting information for analysis of the problem of free fall. This investigation is to examine the effect of air resistance on the distance of free fall in 5 seconds from a location $y = 0$, where the object is initially at rest ($y$ is in the direction of gravity). Hence, we want to determine the distance, $y = L$ that an object falls from a state of rest with and without air resistance. In the book entitled *Introduction to Theoretical Mechanics* by R.A. Becker, that was published by McGraw-Hill Book Company (1954), the equations for free fall are given. The three cases of interest are as follows:

# Sample Problem: Free Fall

1. Without air resistance:

$$a = \frac{d^2y}{dt^2} = g, \quad v = \frac{dy}{dt} = gt, \quad y = \frac{1}{2}gt^2,$$

where $a$ is the acceleration of the object, $v$ is its speed and $y$ is the distance of its free fall from the start of motion at $t = 0$.

2. With resistance proportional to the linear power of velocity:

$$a = \frac{d^2y}{dt^2} = g - k\frac{dy}{dt}, \quad v = \frac{dy}{dt} = \frac{g}{k}\left(1 - e^{-kt}\right), \quad y = \frac{g}{k}t - \frac{g}{k^2}\left(1 - e^{-kt}\right).$$

3. With resistance proportional to the second power of velocity:

$$a = \frac{d^2y}{dt^2} = g - k\left(\frac{dy}{dt}\right)^2, \quad v = \frac{dy}{dt} = \frac{1}{2}\frac{g}{k}\tanh\left(\frac{gk}{2}t\right),$$

$$y = \frac{1}{k}\log_e\left[\cosh\left(gkt/2\right)\right].$$

# Sample Problem: Free Fall

For all three cases the initial condition is $y = v = 0$ at $t = 0$. (You will learn more about ordinary differential equations in your third semester of mathematics. In addition, in a first course in fluid mechanics you will learn about some of the details about air resistance. In particular, for air resistance associated with "laminar flow" problems, case 2 applies. In many practical situations it is case 3 that is usually the case of interest; this is the case where "turbulent flow" is important.)

Let us consider the following problem: Let us assume the above equations are correct. Note that they are written for a unit of mass. Let us also assume $g = 9.81\ \text{m/s}^2$ and $k = 0.2$ for the air-drag parameter. Finally, let us answer the following:

(a) What is $y$ in meters for $t = 5$ seconds of free fall for the three cases.
(b) What are the terminal speeds in meters per second for cases 2 and 3 for the specified air-drag parameter?
(c) Is the terminal speed reached at $t = 5$ seconds?

# Sample Problem: Free Fall

Note:

The terminal speeds are $g/k$ and $(g/k)/2$ for cases 2 and 3, respectively, where the terminal speed is the time independent or steady speed reached after a sufficient distance of free fall. It is the speed at which the gravitational force balances the air resistance force. From Part I, the Essentials, we learned that MATLAB is quite useful in this type of problem because it has the capability of computing the elementary functions in the formulas above.

As part of providing an answer to the questions raised, we want to examine the distance of free fall within the 5 seconds of flight time with the equations for $y$ given above. Let us examine the influence of air resistance on free fall graphically; thus, we need to plot the distance $y$ versus $t$ from $t = 0$ to $t = 5$ seconds. We shall plot all three curves on one figure for direct comparison. We will show that for a short period of time (significantly less than the 5 seconds) after the onset of motion all three curves are on top of each other. We will also show that at $t = 5$ seconds the distances of free fall are quite different.

The steps in the structure plan and the MATLAB code are as follows:

```
%
%     Free fall analysis (saved as FFall.m):
%      Comparison of exact solutions of free
%      fall with zero, linear and quadratic
%      friction for t = 0 to 5 seconds.
%
% Script by D. T. V. ......... September 2006.
% Revised by D.T.V.  ......... 2008/2016/2018.
%
% Step 1: Specify constants
%
% Friction coefficient provided in the problem statement.
k = 0.2;
% Acceleration of gravity in m/s/s.
g = 9.81;
%
% Step 2: Selection of time steps for computing solutions
%
dt = .01;
%
% Step 3: Set initial condition (the same for all cases)
%
t(1) = 0.; v(1) = 0.; y(1) = 0.;
%
t = 0:dt:5;
```

```
%
% Step 4: Compute exact solutions at each time step
%         from t = 0 to 5.
%
%    (a) Without friction:
%
         v = g * t;
         y = g * t.^2 * 0.5;
%
%    (b) Linear friction
%


         velf = (g/k) * (1. - exp(-k*t));
         yelf = (g/k) * t - (g/(k^2)) * (1.-exp(-k*t));
%
%    (c) Quadratic friction
%
         veqf = sqrt(g/k) * tanh( sqrt(g*k) * t);
         yeqf = (1/k) * log(cosh( sqrt(g*k) * t) );
%
% Step 5: Computation of the terminal speeds
%         (cases with friction)
%
         velfT = g/k;
         veqfT = sqrt(g/k);
```

```
%
% Step 6: Graphical comparison
%
 plot(t,y,t,yelf,t,yeqf)
 title('Fig 1. Comparison of results')
 xlabel(' Time, t')
 ylabel(' Distance, y ')
 figure
 plot(t,v,t,velf,t,veqf)
 title('Fig. 2. Comparison of results')
 xlabel(' Time, t')
 ylabel(' Speed, v ')
%
% Step 7: Comparison of distance and speed at t = 5
%
disp(' ');
fprintf(' y(t) = %f, yelf(t) = %f, yeqf(t) = %f at t = %f\n',...
          y(501),yelf(501),yeqf(501),t(501))
disp(' ');
fprintf(' v(t) = %f, velf(t) = %f, veqf(t) = %f at t = %f\n',...
          y(501),yelf(501),yeqf(501),t(501))
%
% Step 8: Comparison of terminal velocity
%
disp(' ');
fprintf('  velfT = %f, veqfT = %f\n',...
   velfT,veqfT)
```

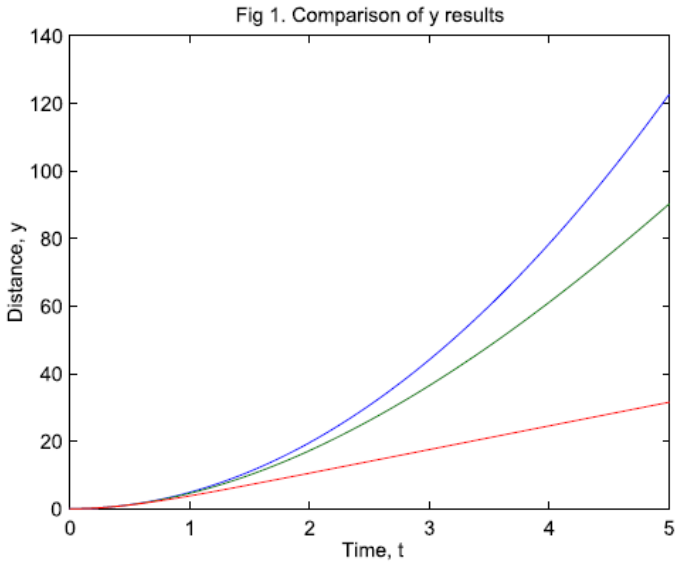



**FIGURE 12.3**
Comparison of free-fall distance: Top curve is for no friction, middle curve is for linear friction and the bottom curve is for quadratic friction.

```
>> FFall

y(t) = 122.625000, yelf(t) = 90.222433, yeqf(t) = 31.552121 at t = 5.000000
v(t) = 49.050000, velf(t) = 31.005513, veqf(t) = 7.003559 at t = 5.000000
velfT = 49.050000, veqfT = 7.003571
```

The figures illustrate, as we may have expected, that for no friction the object falls the furthest distance. The case with quadratic friction reaches terminal velocity well within the five seconds examined. The linear friction case did not reach terminal speed, yet it is moving at a slower velocity as compared with the no friction case. Keep in mind that a unit mass object is falling with a friction coefficient $k = 0.2$. The same $k$ was used for both the second and third cases. Within the first half second from the time of release the three curves are not distinguishable illustrating the fact that it takes a little time before the friction effects are felt. At the time of 5 seconds after release, the speed and the distance fallen are quite different. It is not surprising that quadratic friction slows the object quicker because the air resistance (or friction) is proportional to the speed squared, which is significantly larger than speed to the first power (as it is in the linear-friction case).