

## Chapter 9 MATLAB Graphic

### % 9.1 BASIC 2-D GRAPHS

```
figure, plot(rand(1, 20));
```

```
a=rand(1, 20);
```

```
%-----
```

```
    x = 0:pi/40:4*pi;
```

```
figure, plot(x, sin(x))
```

```
%-----
```

```
    figure, plot([0 4], [1 3])
```

```
ezplot('log(x)')
```

```
x = 0:pi/40:2*pi;
```

```
figure, plot(sin(x), cos(x))
```

```
%-----
```

```
% Line type
```

```
x = 0:pi/40:4*pi;
```

```
figure, plot(x, sin(x), '--') % dashed line
```

```
figure, plot(x, sin(x), 'o') % point type
```

```
figure, plot(x,sin(x), x, cos(x), 'om--') % dashed with point
```

```

% Various line types, plot symbols and colors may be obtained with
%   plot(X,Y,S) where S is a character string made from one element
%   from any or all the following 3 columns:
%
%       b    blue      .    point      -    solid
%       g    green     o    circle     :    dotted
%       r    red       x    x-mark     -.   dashdot
%       c    cyan      +    plus       --   dashed
%       m    magenta   *    star       (none) no line
%       y    yellow   s    square
%       k    black    d    diamond
%       w    white    v    triangle (down)
%                   ^    triangle (up)
%                   <    triangle (left)
%                   >    triangle (right)
%                   p    pentagram
%                   h    hexagram
%
%   For example, plot(X,Y,'c+:') plots a cyan dotted line with a plus
%   at each data point; plot(X,Y,'bd') plots blue diamond at each data
%   point but does not draw any line

% Example
%       x = -pi:pi/10:pi;
%       y = tan(sin(x)) - sin(tan(x));
%       plot(x,y,'--rs','LineWidth',2,...
%           'MarkerEdgeColor','k',...
%           'MarkerFaceColor','g',...
%           'MarkerSize',10)
%-----Sec. 9.1.1-----

```

### Sec. 9.1.1 Labels

`gtext('text')`: writes a string ('text') in the graph window. Interactively by mouse to pointer the position of text in the figure.

`grid on` (or `off`) adds/removes grid lines to/from the current graph.

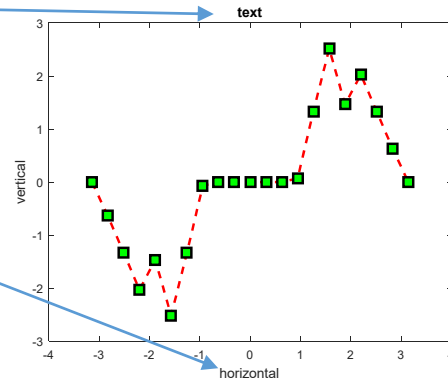
`text(x, y, 'text')` writes text in the graphics window at the point specified by `x` and `y`.

title('text') writes the text as a title on top of the graph.

xlabel('horizontal') labels the  $x$ -axis.

ylabel('vertical') labels the  $y$ -axis.

```
%-----
% 9.1.2 Multiple plots in a
figure: on a same plot, & separate
plots (subplot)
```



```
figure, plot(x,sin(x), x, cos(x), 'om--') % dashed with point
% to have independent y-axis labels on the left and the
% right,
```

```
x = 0:pi/40:4*pi;
figure(4), plotyy(x,sin(x), x, 10*cos(x))
%-----
[x, y] = meshgrid(-3:0.3:3);
z = x .* exp(-x.^2 - y.^2);
```

```
subplot(2,2,1)
mesh(z),title('subplot(2,2,1)')
subplot(2,2,2)
mesh(z)
% after mesh, use different view for the subplot
view(-37.5,70),title('subplot(2,2,2)')
```

```
subplot(2,2,3)
mesh(z)
view(37.5,-10),title('subplot(2,2,3)')
```

```
subplot(2,2,4)
mesh(z)
view(0,0),title('subplot(2,2,4)')
```

```

% Sec. 9.1.3 Line type
x = 0:pi/40:4*pi;
figure, plot(x, sin(x), '--') % dashed line
figure, plot(x, sin(x), 'o') % point type
figure, plot(x,sin(x), x, cos(x), 'om--') % dashed with point

% Various line types, plot symbols and colors may be obtained with
%   plot(X,Y,S) where S is a character string made from one element
%   from any or all the following 3 columns:
%
%
%       b    blue        .    point        -    solid
%       g    green       o    circle       :    dotted
%       r    red         x    x-mark       -.   dashdot
%       c    cyan        +    plus         --   dashed
%       m    magenta     *    star         (none) no line
%       y    yellow      s    square
%       k    black       d    diamond
%       w    white       v    triangle (down)
%                       ^    triangle (up)
%                       <    triangle (left)
%                       >    triangle (right)
%                       p    pentagram
%                       h    hexagram
%
%   For example, plot(X,Y,'c+:') plots a cyan dotted line with a plus
%   at each data point; plot(X,Y,'bd') plots blue diamond at each data
%   point but does not draw any line

% Example
%       x = -pi:pi/10:pi;
%       y = tan(sin(x)) - sin(tan(x));
%       plot(x,y,'--rs','LineWidth',2,...
%           'MarkerEdgeColor','k',...
%           'MarkerFaceColor','g',...
%           'MarkerSize',10)

```

%-----Sec---9.1.4 -----

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

clear all;

x = 0:pi/40:4\*pi;

figure, plot(x, sin(x), '--') % dashed line

axis( [0 pi\*4 -1 1] )

% graphical input : allows you to select an unlimited number of points

% from the current graph using a mouse or arrow keys. press Enter terminate to  
terminate

clc;clear all;

[x,y]=ginput;

[x,y]=ginput(n);

% 9.1.8 Logarithmic plots

x = 0:0.01:4;

figure, semilogy(x, exp(x)), grid

% See also semilogx, loglog.

%-----

% 9.1.9 Polar plots

x = 0:pi/40:2\*pi;

figure, polar(x, sin(2\*x)),grid

%-----

% 9.1.10 Plotting rapidly changing mathematical functions: fplot

x = 0.01:0.001:0.1;

figure, plot(x, sin(1./x))

%-----

% fplot evaluates it more frequently over regions where it changes more rapidly

figure, fplot(@(x)sin(1./x), [0.01 0.1]) % no, 1./x not needed!

% 9.1.11 The property editor

**Edit -> Figure Properties** from the figure window. Editor figure.

Check table 9.1

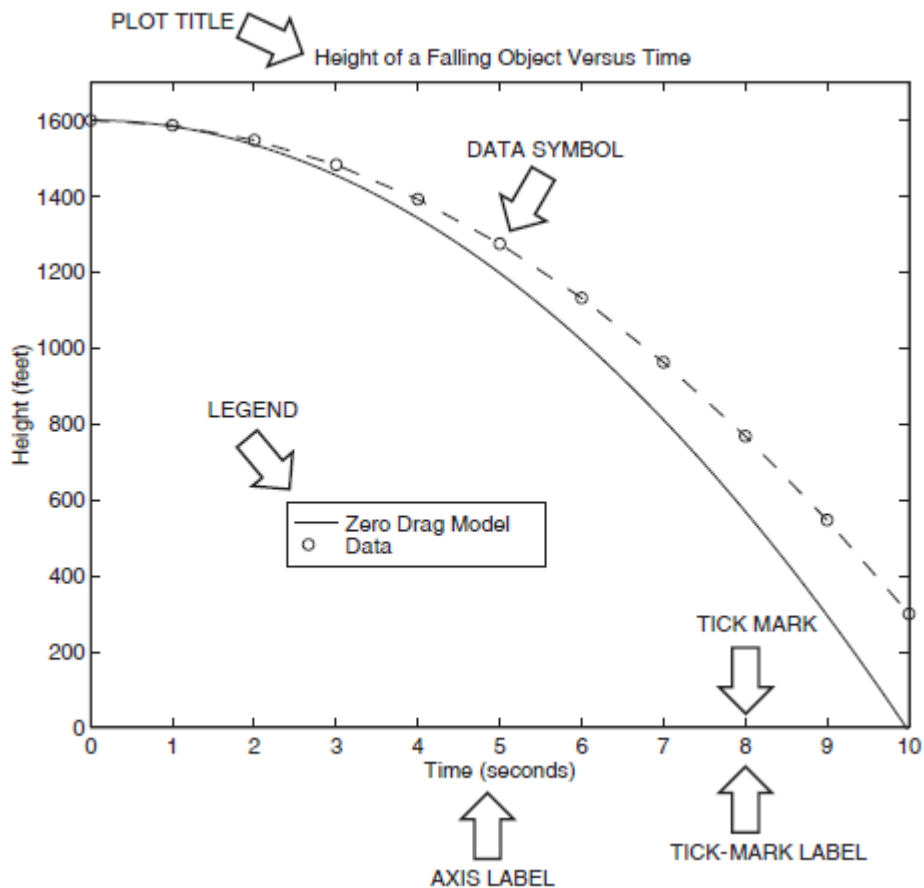
```

%=====
=====
% Exercise: Example
    x = -pi:pi/10:pi;
    y = tan(sin(x)) - sin(tan(x));
    plot(x,y,'--rs','LineWidth',2,...
          'MarkerEdgeColor','k',...
          'MarkerFaceColor','g',...
          'MarkerSize',10)
text(x, y, 'text');

title('text') % writes the text as a title on top of the graph.
xlabel('horizontal') % labels the x-axis.
ylabel('vertical') % labels the y-axis.

% Please use this as an example to change the figure properties :
% Like line and label fonts, etc. one can also use figure editor in the
figure window.

```



**Table 5.1–2** Basic xy plotting commands

Command	Description
<code>axis([xmin xmax ymin ymax])</code>	Sets the minimum and maximum limits of the <i>x</i> and <i>y</i> axes.
<code>fplot(function, [xmin xmax])</code>	Performs intelligent plotting of functions, where <i>function</i> is a function handle that describes the function to be plotted and <code>[xmin xmax]</code> specifies the minimum and maximum values of the independent variable. The range of the dependent variable can also be specified. In this case the syntax is <code>fplot(function, [xmin xmax ymin ymax])</code> .
<code>grid</code>	Displays gridlines at the tick marks corresponding to the tick labels.
<code>plot(x,y)</code>	Generates a plot of the array <i>y</i> versus the array <i>x</i> on rectilinear axes.
<code>plot(y)</code>	Plots the values of <i>y</i> versus their indices if <i>y</i> is a vector. Plots the imaginary parts of <i>y</i> versus the real parts if <i>y</i> is a vector having complex values.
<code>polyval(p,x)</code>	Evaluates the polynomial <i>p</i> at specified values of its independent variable <i>x</i> .
<code>print</code>	Prints the plot in the Figure window.
<code>title('text')</code>	Puts text in a title at the top of a plot.
<code>xlabel('text')</code>	Adds a text label to the <i>x</i> axis (the abscissa).
<code>ylabel('text')</code>	Adds a text label to the <i>y</i> axis (the ordinate).

## % Sec 9.2 3-D plot

### % 9.2.1 plot3

```
clear all; close all;
```

```
figure, plot3(rand(1,10), rand(1,10), rand(1,10))
```

```
%-----
```

```
t = 0:pi/50:10*pi;
```

```
figure, plot3(exp(-0.02*t).*sin(t), exp(-0.02*t).*cos(t),t), ...
```

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

```
%-----
```

### % 9.2.2 Animated 3-D plots with comet3

```
t = 0:pi/50:10*pi;
```

```
% it draws with a moving ¦comet head¦.
```

```
figure, comet3(exp(-0.02*t).*sin(t), exp(-0.02*t).*cos(t),t), ...
```

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

### % 9.2.3 Mesh surfaces

```
% The first step is to set up the grid in the x-y plane over which the  
surface is to be plotted.
```

```
[x y] = meshgrid(0:5); % generate a meshgrid
```

```
z = x.^2 - y.^2;
```

```
figure;mesh(x,y,z)
```

```
dis=sqrt(x.^2 + y.^2);
```

```
mesh(x,y,dis)
```

```
axis([0 5 0 5 0 10])
```

```
% generate the surface points:
```



```
[x, y] = meshgrid(-3:0.3:3);
z = x.^2 - y.^2;
```

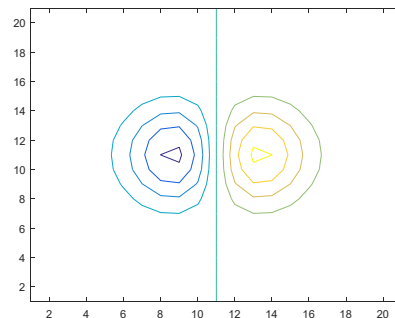
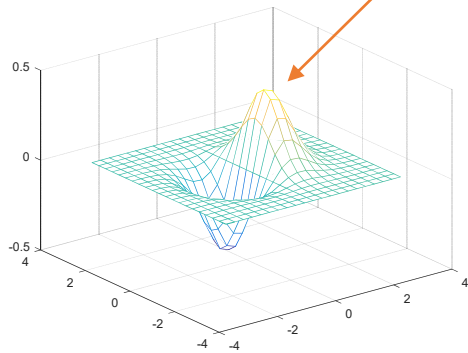
```
% plot the surface points:
figure, mesh(z);
```

```
[x, y] = meshgrid(-3:0.3:3);
z = x .* exp(-x.^2 - y.^2);
```

```
% mesh(x,y,Z) and mesh(x,y,Z,C), with two vector arguments replacing
% the first two matrix arguments, must have length(x) = n and
% length(y) = m where [m,n] = size(Z). In this case, the vertices
% of the mesh lines are the triples (x(j), y(i), Z(i,j)).
% Note that x corresponds to the columns of Z and y corresponds to
% the rows.
```

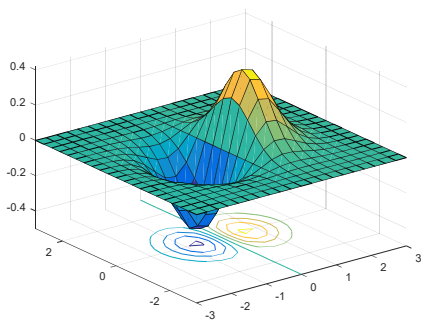
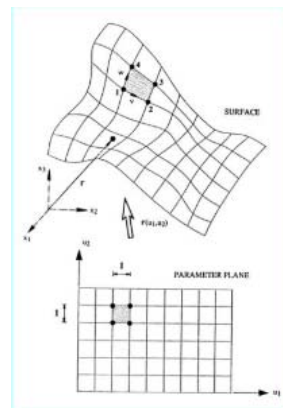
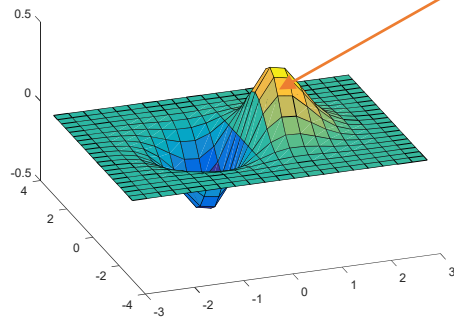
```
figure(2);mesh(x,y,z)
```

```
figure(4);contour(z)
```



```
% In this case, the vertices of the surface facets;
% are the triples (x(j),y(i),Z(i,j)). Note that x corresponds to the
% columns of Z and y corresponds to the rows of Z. For a complete
% discussion of parametric surfaces, using the tilted surface
% for the representation: see the SURF function.
```

```
figure(3);surface(x,y,z)
figure(5);surfc(x,y,z)
```



```
%-----
```

```
% This drawing is another example of a mesh surface.
```

```
[x y ] = meshgrid(-8 : 0.5 : 8);
```

```
r = sqrt(x.^2 + y.^2) + eps;
```

```
z = sin(r) ./ r;
```

```
figure, mesh(z);
```

```
figure, mesh(x,y,z);
```

```
figure, surface(z); % You can change the viewpoint in the figure window
```

```
figure, surface(x,y,z)
```

## Exercises

1. Draw the surface shown in [Figure 9.7](#) with a finer mesh (of 0.25 units in each direction), using

```
[x y] = meshgrid(0:0.25:5);
```

(the number of mesh points in each direction is 21).

2. The initial heat distribution over a steel plate is given by the function

$$u(x, y) = 80y^2 e^{-x^2 - 0.3y^2}$$

Plot the surface  $u$  over the grid defined by

$$-2.1 \leq x \leq 2.1, \quad -6 \leq y \leq 6,$$

where the grid width is 0.15 in both directions. You should get the plot shown in

[Figure 9.8](#).

```
%-----
```

```
% 9.2.4 Contour plots
```

```
[x y] = meshgrid(-2.1:0.15:2.1,-6:0.15:6); % x-y-grids different
```

```
z = 80 * y.^2 .* exp(-x.^2 - 0.3*y.^2);
```

```
figure(9), contour(z)
```

```
figure(10);meshc(z)
```

```
figure, contour(z, 20)
```

```
figure, contour3(z)
```

```
%-----
```

```
[x y] = meshgrid(-2.:2:2);
```

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

```
figure, meshc(z)
```

```
figure, surfc(z)
```

```
%-----
```

```
% 9.2.5 Cropping a surface with NaNs
```

```
% If a matrix for a surface plot contains NaNs, these elements are not plotted. This
```

```
% enables you to cut away (crop) parts of a surface.
```

```
[x y] = meshgrid(-2.:2:2, -2.:2:2);
```

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

```

c = z; % preserve the original surface
figure;mesh(c)
c(1:11,1:21) = NaN*c(1:11,1:21);
figure;mesh(c), xlabel('x-axis'), ylabel('y-axis')
%-----

```

% 9.2.6 Visualizing vector fields

```

[x y] = meshgrid(-2:.2:2, -2:.2:2);
V = x.^2 + y;
% [FX,FY] = gradient(F) returns the numerical gradient of the
% matrix F. FX corresponds to dF/dx, the differences in x (horizontal)
% direction. FY corresponds to dF/dy, the differences in y (vertical)
% direction. The spacing between points in each direction is assumed to
% be one.
[dx,dy]=gradient(V,0.2,0.2);

```

```

% dx = 2*x;
% dy=ones(size(dx));
% dy = dx; % dy same size as dx
% dy(:,.) = 1; % now dy is same size as dx but all 1's
figure;
contour(x, y, V), hold on
quiver(x, y, dx, dy), hold off

```

%-----

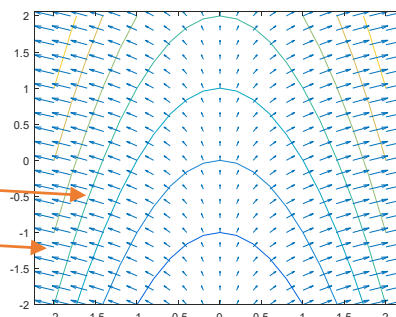
% you can use the gradient function to estimate the derivatives:

```

[x y] = meshgrid(-2:.2:2, -2:.2:2);
V = x.^2 + y;
[dx dy] = gradient(V, 0.2, 0.2);
contour(x, y, V), hold on
quiver(x, y, dx, dy), hold off

```

%-----



% 9.2.7 Visualization of matrices: use mesh to check the values in the  
% matrix.

```

a = zeros(30,30);
a(:,15) = 0.2*ones(30,1);
a(7,:) = 0.1*ones(1,30);
a(15,15) = 1;
mesh(a)
%-----
% 9.2.8 Rotation of 3-D graphs

```

```

clear all;close all;
figure;
a = zeros(30,30);
a(:,15) = 0.2*ones(30,1);
a(7,:) = 0.1*ones(1,30);
a(15,15) = 1;
el = 30;
% rotate around z-axis
for az = -37.5:30:-37.5+360
    mesh(a), view(az, el)
    pause(0.5)
end

```

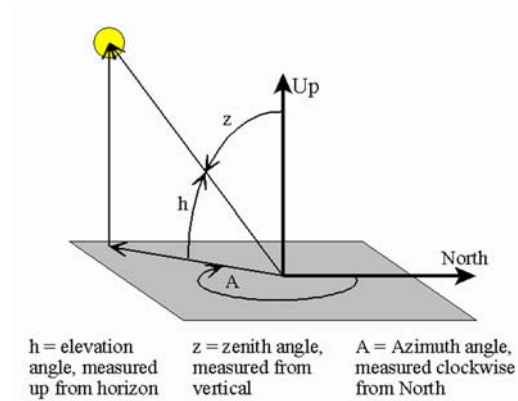
% The program therefore rotates you in a counter-clockwise direction about the  
% z-axis in 15° steps starting at the default position.

```

figure; mesh(a);
figure;
az=30;
for el = -37.5:30:-37.5+360
    mesh(a), view(az, el)
    pause(1.0)
end

```

The second argument of view is the vertical elevation **el** (in degrees). This is the angle a line from the viewpoint makes with the  $x$ - $y$  plane. A value of 90° for **el** means you are directly overhead. Positive values of the elevation mean you are above the  $x$ - $y$  plane; negative values mean you are below it.



## Sec. 9.3 handle graphics

- online documentation **MATLAB Help: Graphics.**

### Graphic object: Graphs Are Composed of Specific Objects

When you create a graph, for example by calling the plot function, MATLAB automatically performs a number of steps to produce the graph.

(1) creating objects,

(2) setting the properties of these objects to appropriate values for your specific graph.

The objects are arranged in a parent-child inheritance structure as shown in Figure 9.13. For example, Line and Text objects are children of Axes objects.

Graphics objects are organized into a hierarchy, as shown by the following diagram.

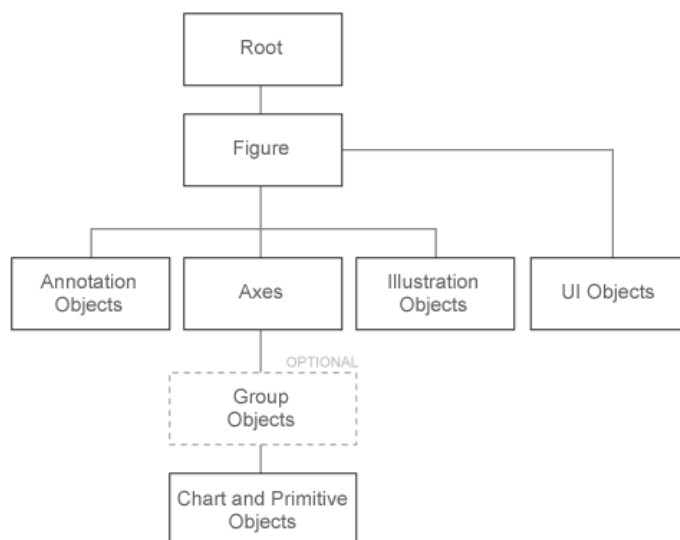
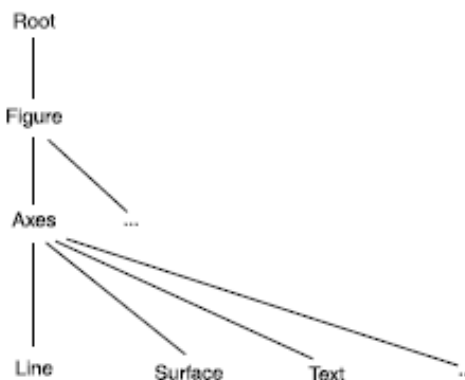
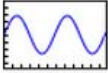
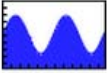
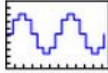

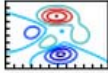
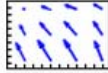


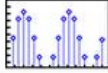

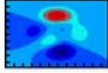
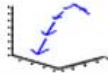
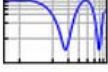

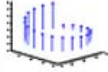


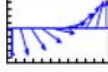
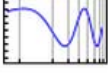

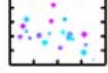

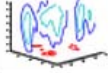
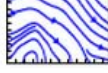
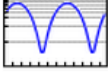
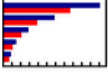
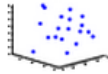
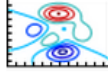
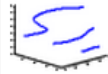
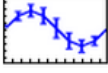
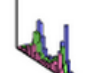

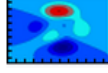

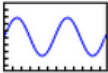

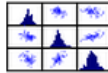

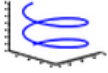


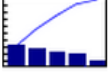


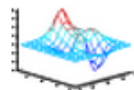

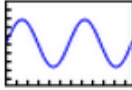
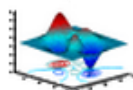
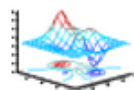

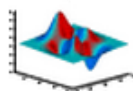
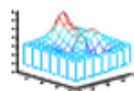


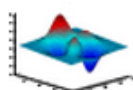
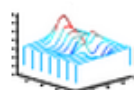
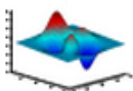
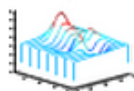
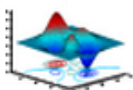
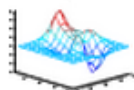
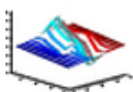
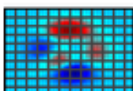
Fig. 9.1 Hierarchical structure of the graphic objects



**FIGURE 9.13** Parent-child relationships of Handle Graphics objects (from top to bottom).

You can use following function to create an chart & primitive object.

Line Plots	Pie Charts, Bar Plots, and Histograms	Discrete Data Plots	Polar Plots	Contour Plots	Vector Fields
plot 	area 	stairs 	polar 	contour 	quiver 
plot3 	pie 	stem 	rose 	contourf 	quiver3 
loglog 	pie3 	stem3 	compass 	contour3 	feather 
semilogx 	bar 	scatter 	ezpolar 	contourslice 	streamslice 
semilogy 	barh 	scatter3 		ezcontour 	streamline 
errorbar 	bar3 	spy 		ezcontourf 	streamribbon 
ezplot 	bar3h 	plotmatrix 			streamtube 
ezplot3 	histogram 				coneplot 
	pareto 				

Surface and Mesh Plots		Polygons	Animation
surf 	mesh 	fill 	animatedline 
surfc 	meshc 	fill3 	comet 
surf1 	meshz 	patch 	comet3 
ezsurf 	waterfall 		
ezsurf 	waterfall 		
ezsurfc 	ezmesh 		
ribbon 	ezmeshc 		
pcolor 			

For example: See the Fig. 9.1 that we have two Chart Line objects in the structure.

```
Figure(11);
x = 0:pi/20:2*pi;
hsin=plot(x,sin(x)) % hsin is the handle of the figure
hold on
hcos=plot(x,cos(x))
hx=xlabel('x-axis') % hx is the handle of the x-axis
```



- Whenever MATLAB creates a graphics objects it automatically creates a handle to that object.
- MATLAB creates a figure object, two chartline object
- You can **get the handle** of the object using a function.
- And use the handle to change the object properties, this is one way to change the properties. In the next section, we will tell you another way to change the properties.
- How to **change the object properties through the object handle?**
- online documentation MATLAB Help: Graphics object properties. -> Chart Line Properties

Example:

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

```
hsin.Color='red';
```

```
hsin.LineWidth=4;
```

```
% Chart Line properties: the help MATLAB documentation
```

There are three functions that return the handle of particular graphics objects:

**gcf** gets the handle of the current figure, e.g., hf = gcf;

**gca** gets the handle of the current axes.

**gco** gets the handle of the current graphics object.

- Use the handle to change or manipulate your graphics object.

In command window: >> get(hsin)

You will get all properties of this figure. In p. 216

Then use the command:

```
hsin.Color='red';
```

```
hsin.LineWidth=4;
```

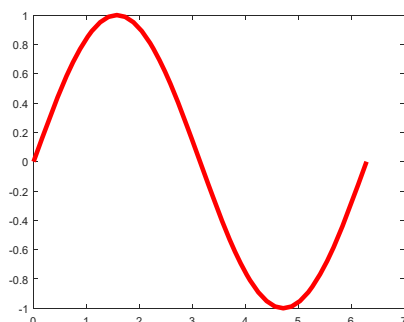
Also, you can use the old version with the

set function:

```
set(handle, 'PropertyName', PropertyValue)
```

```
et(hsin)
Color = [0 0 1]
EraseMode = normal
LineStyle = -
LineWidth = [4]
Marker = none
MarkerSize = [6]
MarkerEdgeColor = auto
MarkerFaceColor = none
XData = [ (1 by 41) double array]
YData = [ (1 by 41) double array]
ZData = []

BeingDeleted = off
ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
```



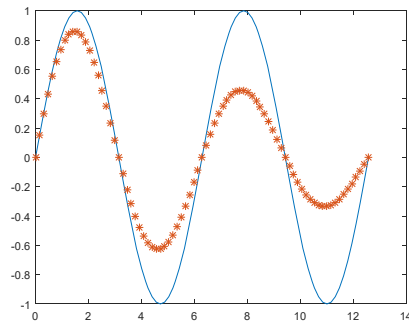


Figure 9.14

- You can find the object through the properties of the objects. In the original version of the plots in Figure 9.14 the decaying plot can be identified by its marker property:  
`hdecay = findobj('marker', 'o') % return the handle of the 'decaying line'.`

### % 9.3.3 A vector of handles

One figure , two Chart Line.

```
x = 0:pi/20:4*pi;
plot(x,sin(x))
hold on
plot(x,exp(-0.1*x).*sin(x),'o')
hold off

% get the object handles of the Childs of the figure
hkids=get(gca,'child');
set(hkids(1),'marker','*') % set the line property of the line 1
set(hkids(2),'LineWidth',4) % set the property of line 2
%%

hkids(1).Marker='o';
hkids(2).LineWidth=2;hkids(2).Color='blue';
```

## % Sec. 9.4 Editing plots

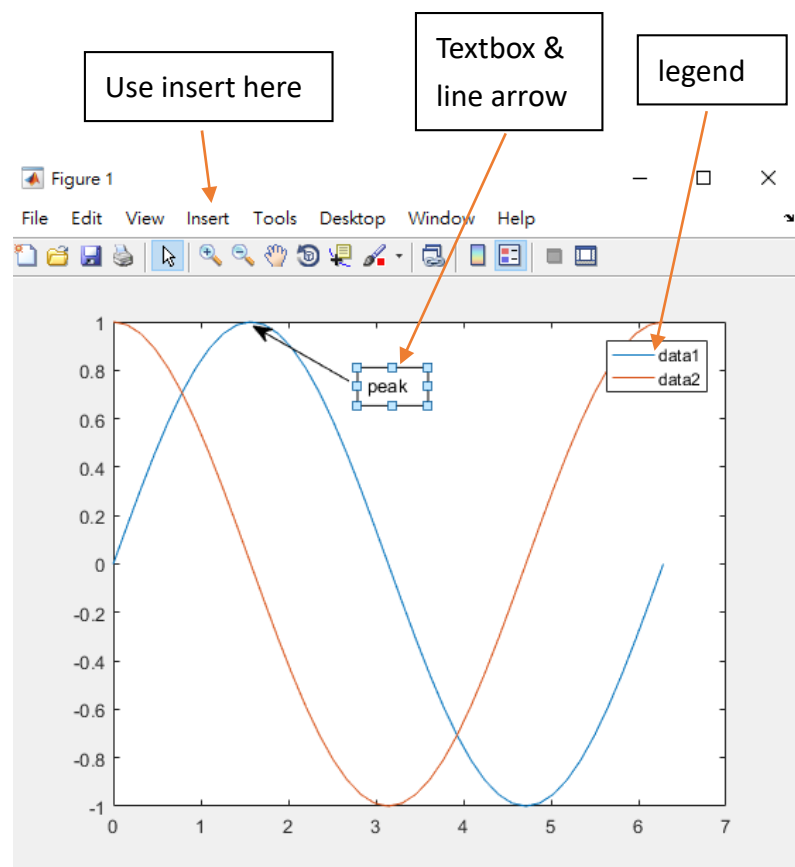
Select **Tools** -> **Edit Plot** in the figure window.

In PLOT EDIT MODE to

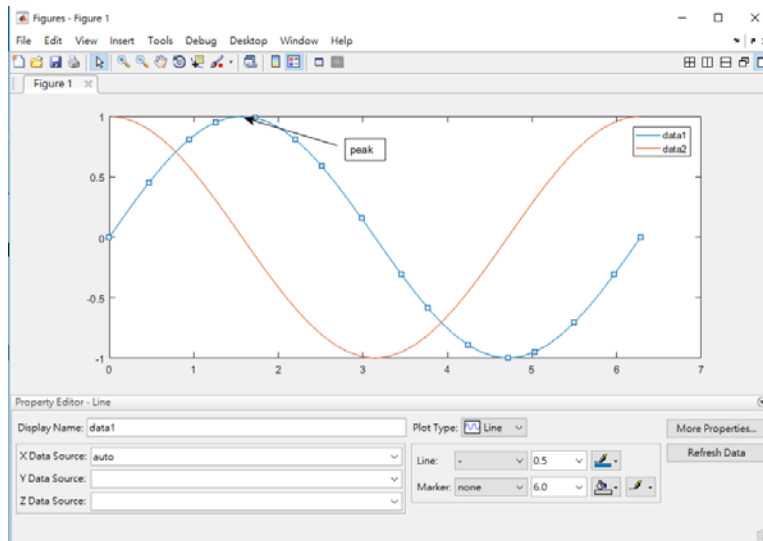
1. Change the line type of the figure.
2. Change the label, Axis.
3. Insert title, legend & textbox & line arrows

```
% 9.4 To experiment with the Property Editor it will be useful  
% to have multiple plots in a figure:
```

```
x = 0:pi/20:2*pi;  
figure, plot(x,sin(x), x, cos(x), 'om--') % dashed with point  
hsin = plot(x,sin(x))  
hold on  
hcos = plot(x,cos(x))  
hold off  
% check text p. 220 to insert Legend, textbox, label  
% of the graphic insert a line and arrow
```



Sec. 9.4.2: In PLOT EDIT MODE, **Double click on an object**, you are starting the PROPERTIES EDITOR  
 you can change the properties for the objects in this figure: figure line, Axes, text, etc.



## % 9.5 ANIMATION

% It generates 16 frames from the Fast Fourier Transforms of complex matrices:

```
clear all;
close all;
for k = 1:16
    plot(fft(eye(k+16)))
    axis equal
    drawnow % draw inside the loop this frame now
    M(k) = getframe;
end
% Movies outside the loop also call playback
figure(11), movie(M, 5)
```

% movie(M,n) plays the movie n times.

% movie(M,n,fps) plays the movie at fps frames per second.

% The default is 12 frames per second.

% try to use this command to change the rate of the movie

```

figure(22)
Z = peaks;
surf(Z)
axis tight manual
ax = gca; % get handle of the current axes
% To fix the figure properties by just change only the children
% check the fig. 9.13 in p. 215 that surface is the children of the
Axes
ax.NextPlot = 'replaceChildren';
loops = 40;
% Initialization of the Movies through structure
% F(loops) = struct('cdata',[],'colormap',[]);
for j = 1:loops
    X = sin(j*pi/10)*Z; % redefine the surface data
    surf(X,Z)
    drawnow % draw this frame now
    % % get a frame of image with the definition by loop
    F(j) = getframe; % to generate a sequence of frames (Movies)
end
figure(33), movie(F, 3)

```

```

%-----

```

### % 9.5.1 Animation with Handle Graphics

```

% animated sine graph
x = 0;
y = 0;
dx = pi/40;
figure;
p = animatedline;
p.LineStyle='none';
p.Marker= 'o';
% since the 'EraseMode','none' function has been remove in the new-
version, one replace it by animatedline

```

```

axis([0 20*pi -2 2])
for x = dx:dx:20*pi;
    x = x + dx;
    y = sin(x);
    % change the properties of the plot object
    % one can use the command >> get(p) to find all properties
    % of the plot
    % set(p, 'XData', x, 'YData', y)
    % p.XData=x;p.YData=y;
    p.Color=rand(1,3);
    addpoints(p,x,y);
    pause(0.5);
    drawnow
end

```

```

    % another example
figure;
h = animatedline;
axis([0 4*pi -1 1])
x = linspace(0,4*pi,100);
h.LineStyle='none';
h.Marker= 'o';

for k = 1:length(x)
    y = sin(x(k));
    addpoints(h,x(k),y);
    drawnow
end

%-----

% The Lorenz strange attractor.
figure;
A = [ -8/3 0 0; 0 -10 10; 0 28 -1 ];
y = [35 -10 -7]';
h = 0.01;
p = plot3(y(1), y(2), y(3), 'o', ...
'erasemode', 'none', 'markersize', 2);

```

```

axis([0 50 -25 25 -25 25])
hold on
i = 1;
while 1
    A(1,3) = y(2);
    A(3,1) = -y(2);
    ydot = A*y;
    y = y + h*ydot;
    % Change colour occasionally
    if rem(i,500) == 0
        set(p, 'color', [rand, rand, rand])
    end
    % Change co-ordinates
    p.XData=y(1);p.YData=y(2);p.ZData=y(3);
    drawnow
    i=i+1;
end

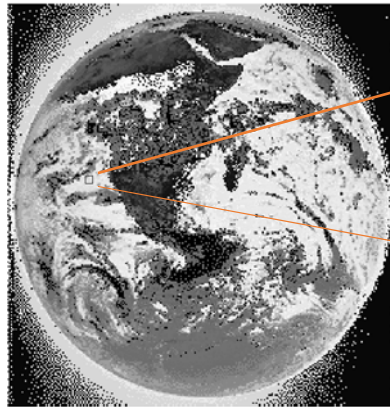
% since the function : 'erasemode','none', has been removed
% so, we use comet3 to plot this figure
% The Lorenz strange attractor.
clear all;close all;
figure;
A = [ -8/3 0 0; 0 -10 10; 0 28 -1 ];
y (1,:)= [35 -10 -7]';
h = 0.01;
i = 1;
for t=1:1000
    A(1,3) = y(t,2);
    A(3,1) = -y(t,2);
    ydot = A*y(t,:)';
    y(t+1,:) = y(t,:) + (h.*ydot)';
end
comet3(y(:,1),y(:,2),y(:,3))

```

% 9.6 color etc.

### Color map representation

The values in the indexed image represent the address of the color in the colormap.  
(indexed image)



```
>> X(50:55,110:115)
```

ans =

6	62	63	22	8	22
63	63	61	24	27	28
5	2	63	63	22	20
52	63	11	5	20	22
61	63	26	4	8	8
5	22	8	4	20	22

```
>> map(26,:) 
```

ans =

0.6784	0.6784	0.6471
--------	--------	--------

Colormap(map)

(color image)



Help document for the colormap

*fx* **colormap** - View and set current **colormap**

This MATLAB function sets the **colormap** for the current figure to the built-in **colormap** specified by name  
[MATLAB > Graphics > Formatting and Annotation > Colormaps](#)



```

%% Sec. 9.6
% indexed image of a color image

load earth
imshow(X,[])
figure(2)
imshow(X,map)
imrect

% application to the different color plot
figure
surf(peaks)
colormap winter % Build-in map with 64 color

% Self-defined color
% Color RGB Triplet
% yellow [1,1,0] % magenta [1,0,1] % cyan [0,1,1]
% red [1,0,0] % green [0,1,0] % blue [0,0,1] % white [1,1,1]
% black [0,0,0]

close all;clear all;
z=peaks
cmap = [0.2, 0.1, 0.5
        0.1, 0.5, 0.8
        0.2, 0.7, 0.6
        0.8, 0.7, 0.3
        0.9, 1, 0];
% application to the different color plot
figure(1); surface(z);view(-35,35);colormap default
figure(2); surface(z);view(-35,35);colormap autumn
% self defined color & indicate the color bar
figure(3); surface(z);view(-35,35);colormap(cmap);colorbar

z=peaks(25);
c(:, :, 1)=rand(25); % 25*25 random values 0~1
c(:, :, 2)=rand(25);
c(:, :, 3)=rand(25);
surf(z,c);

```

```

% The three pages of c specify the values of RGB,

%% 9.7 lighting and camera

% camlight : Create or move a light with respect to the camera position
% lightangle : Create or position a light in spherical coordinates
% light : Create a light object

figure;
surf(peaks)
axis vis3d
h = light; % return a light handle
get(h)
% h = camlight('left');
for az = -50:10:50
    lightangle(h,az,30) % camera fixed & lighting changed with az angle
    pause(.4)
end

% camera Lighting form the right
% Create or move light object in camera coordinates
figure(2);
surf(peaks)
axis vis3d
h = light; % return a light handle
for i=1:4
    camlight('right') % add the light for the graphic
    pause(.5)
end

% camera Lighting form the left
figure(2);
surf(peaks)
axis vis3d
for i=1:4
    camlight left
    pause(.5)
end

```

```

figure(3);
surf(peaks)
axis vis3d
h = camlight('left'); % return a camera lighting handle
for i = 1:20;
    camorbit(10,0) % rotate the camera along z=axis by 10 degree
    camlight(h,'left') % lighting change with of the cmaera position
    pause(.1)
end

```

% Sec 9.8

