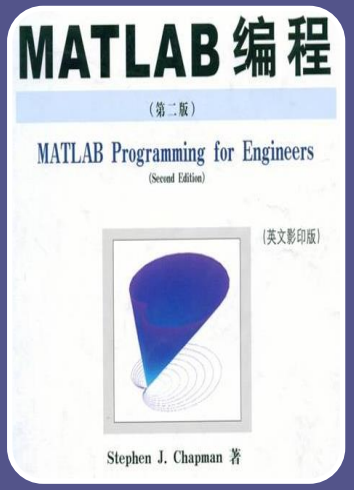
MATLAB Programming (Lecture 6)

Dr. Sun Bing
School of EIE
Beihang University

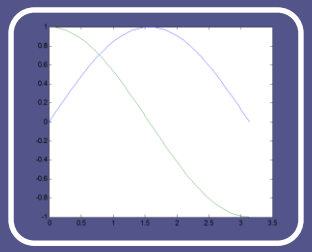
Contents

Graphics processing



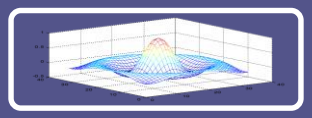
- 2-D graphics
- 3-D graphics
- 4-D graphics
- Basic graphics technology
- High-level graphics technology

Contents

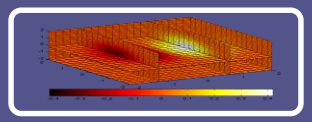


2-D graphics

- 2-D Basic drawing commands
- special 2-D graphics functions

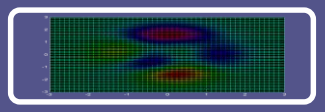


3-D graphics



4-D graphics

Basic graphics technology



High-level graphics technology

1. 1 2-D Basic drawing commands

Plot 2-D line plot

`Plot(Y)` plots the columns of `Y` versus their index if `Y` is a real number. If `Y` is complex, `plot(Y)` is equivalent to `plot(real(Y),imag(Y))`. In all other uses of plot, the imaginary component is ignored.

1. 1 2-D Basic drawing commands

`Plot(X1,Y1,...)` plots all lines defined by `Xn` versus `Yn` pairs. If only one of `Xn` or `Yn` is a matrix, the vector is plotted versus the rows or columns of the matrix, depending on whether the vector's row or column dimension matches the matrix. If `Xn` is a scalar and `Yn` is a vector, disconnected line objects are created and plotted as discrete points vertically at `Xn`.

1. 1 2-D Basic drawing commands

`Plot(X1,Y1,LineSpec,...)`

plots all lines defined by the `Xn,Yn,LineSpec` triples, where `LineSpec` is a line specification that determines line type, marker symbol, and color of the plotted lines. You can mix `Xn,Yn,LineSpec` triples with `Xn,Yn` pairs:

```
plot(X1,Y1,X2,Y2,LineSpec,X3,Y3).
```

1. 1 2-D Basic drawing commands

Plot 2-D line plot

```
Plot(X1,Y1,LineSpec,...)
```

Line Style Specifiers

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Color Specifiers

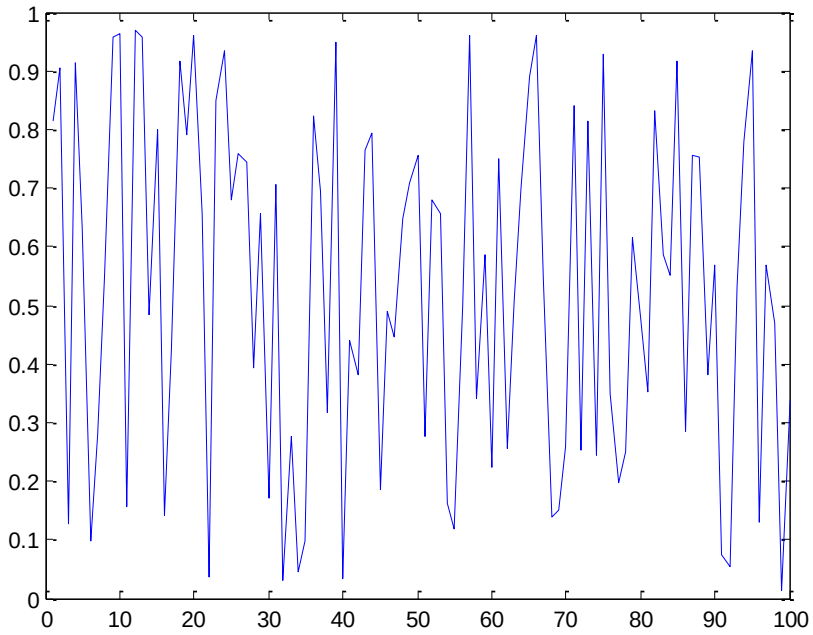
Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

Marker Specifiers

Specifier	Marker Type
+	Plus sign
o	Circle
*****	Asterisk
.	Point (see note below)
x	Cross
'square' or s	Square
'diamond' or d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
'pentagram' or p	Five-pointed star (pentagram)
'hexagram' or h	Six-pointed star (hexagram)

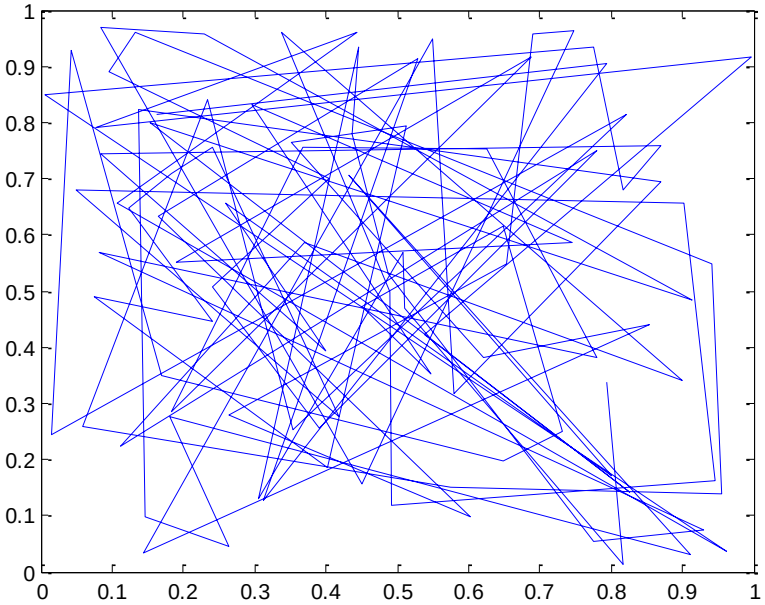
1. 1 2-D Basic drawing commands

Plot 2-D line plot



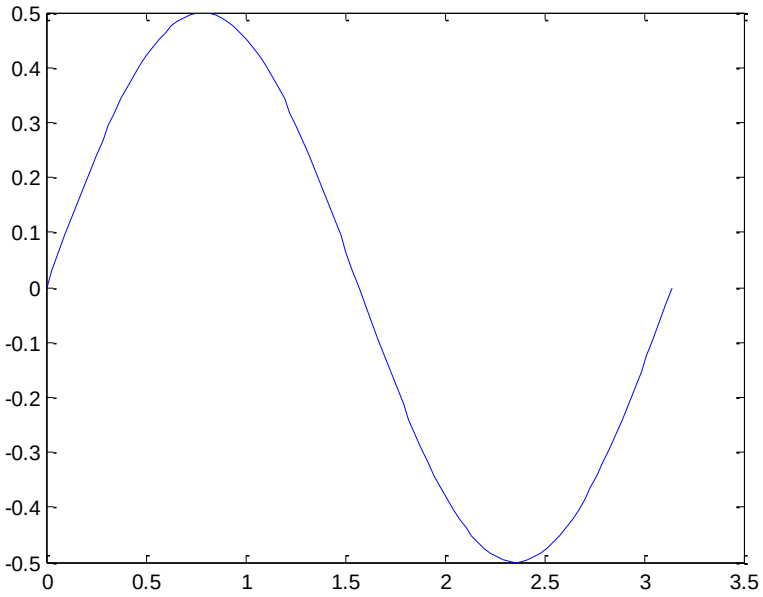

```
y=rand(100,1);
plot(y)
```

```
x=rand (100,1);
z=x+y.*i;
plot(z)
```



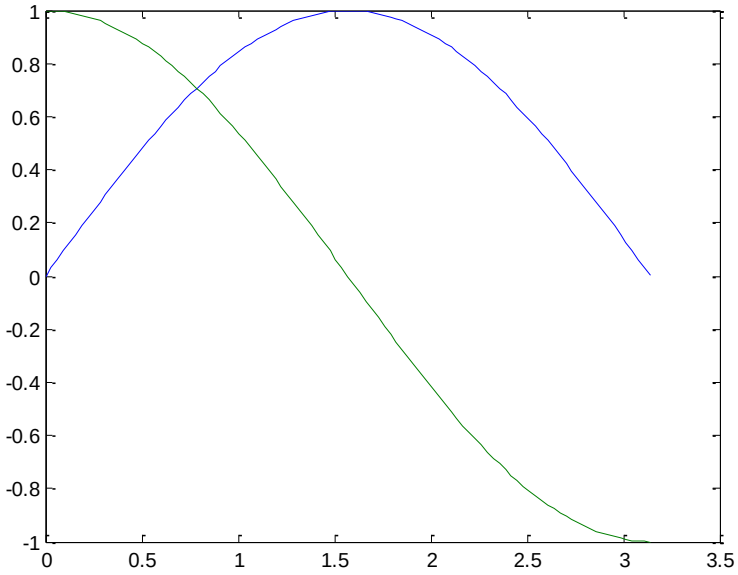
1. 1 2-D Basic drawing commands

Plot 2-D line plot



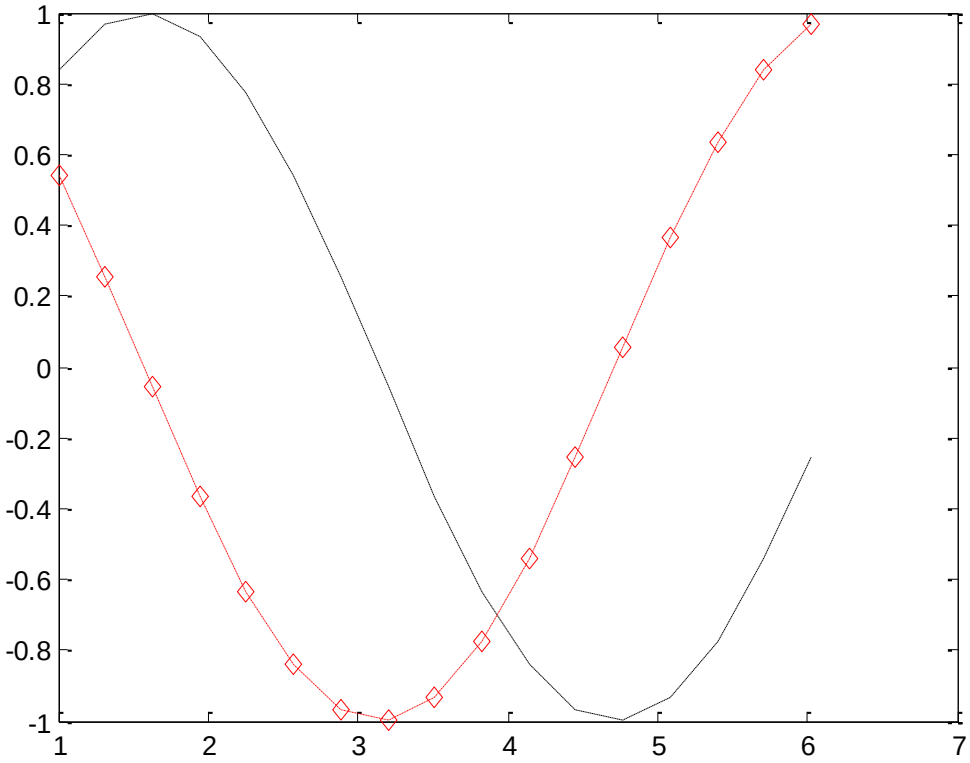

```
x=0:0.01*pi:pi;
y=sin(x).*cos(x);
plot(x,y)
```

```
x=0:0.01*pi:pi;
y=[sin(x'),cos(x')];
plot([x',x'],y)
```



1. 1 2-D Basic drawing commands

Plot 2-D line plot




```
x=1:0.1*pi:2*pi;

y=sin(x);

z=cos(x);

plot(x,y,'--k',x,z,'-.rd')
```

1.2 The special 2-D graphics functions

`Semilogx` Semilogarithmic plots

`Semilogy` Semilogarithmic plots

`semilogx` and `semilogy` plot data as logarithmic scales for the *x*- and *y*-axis, respectively.

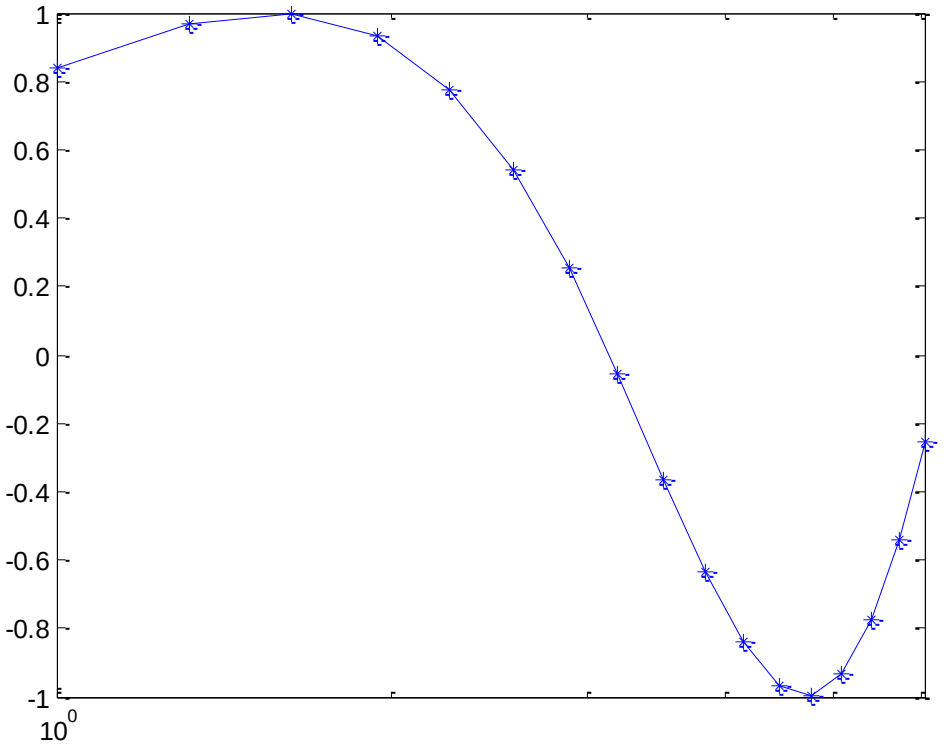
`Semilogx(Y)` creates a plot using a base 10 logarithmic scale for the *x*-axis and a linear scale for the *y*-axis.

Tips: help log, log2, log10

1.2 The special 2-D graphics functions

`Semilogx` Semilogarithmic plots

```
x=1:0.1*pi:2*pi;
y=sin(x);
semilogx(x,y,'-*')
```



1.2 The special 2-D graphics functions

`Loglog` Log-log scale plot

`Loglog(Y)` plots the columns of `Y` versus their index if `Y` contains real numbers. If `Y` contains complex numbers, `loglog(Y)` and `loglog(real(Y),imag(Y))` are equivalent. loglog ignores the imaginary component in all other uses of this function.

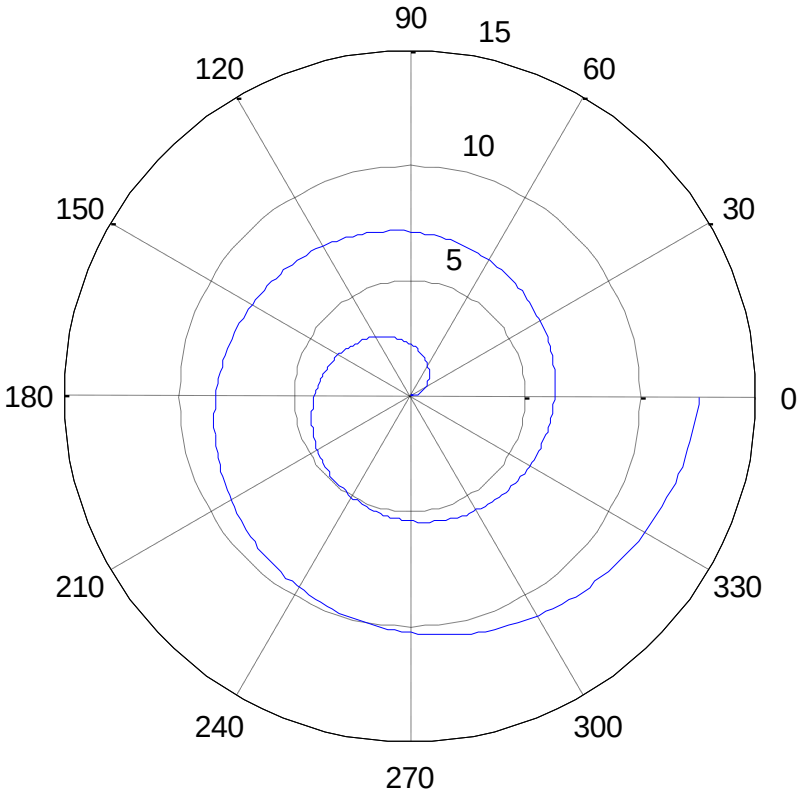
1.2 The special 2-D graphics functions

`Polar` Polar coordinate plot

`polar(theta,rho)` creates a polar coordinate plot of the angle theta versus the radius rho. `theta` is the angle from the *x*-axis to the radius vector specified in radians; `rho` is the length of the radius vector specified in data space units.

1.2 The special 2-D graphics functions

`Polar` Polar coordinate plot




```
x=0:0.01*pi:4*pi;
y=sin(x/2)+x;
polar(x,y,'-')
```

1.2 The special 2-D graphics functions

`Plotyy` 2-D line plots with y-axes on both left and right side

`plotyy(X1,Y1,X2,Y2)` plots `X1` versus `Y1` with *y*-axis labeling on the left and plots `X2` versus `Y2` with *y*-axis labeling on the right.

1.2 The special 2-D graphics functions

`Plotyy` 2-D line plots with y-axes on both left and right side

`plotyy(X1,Y1,X2,Y2,function)` uses the specified plotting function to produce the graph.

function can be either a function handle or a string specifying `plot, semilogx, semilogy, loglog, stem,` or any MATLAB function that accepts the syntax

1.2 The special 2-D graphics functions

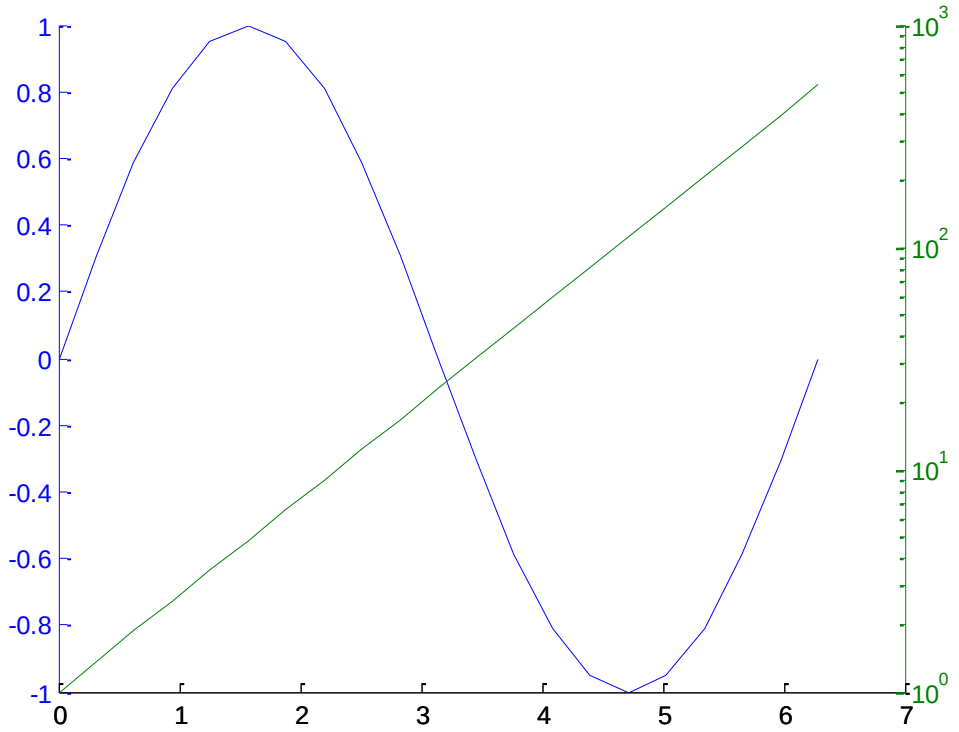
`Plotyy` 2-D line plots with y-axes on both left and right side

```
plotyy(X1,Y1,X2,Y2,'function1','function2‘)
```

uses `function1(X1,Y1)` to plot the data for the left axis and `function2(X2,Y2)` to plot the data for the right axis.

1.2 The special 2-D graphics functions

`Plotyy` 2-D line plots with y-axes on both left and right side




```
x=0:0.1*pi:2*pi;

y=sin(x);

z=exp(x);

plotyy(x,y,x,z,'plot','semilogy')
```

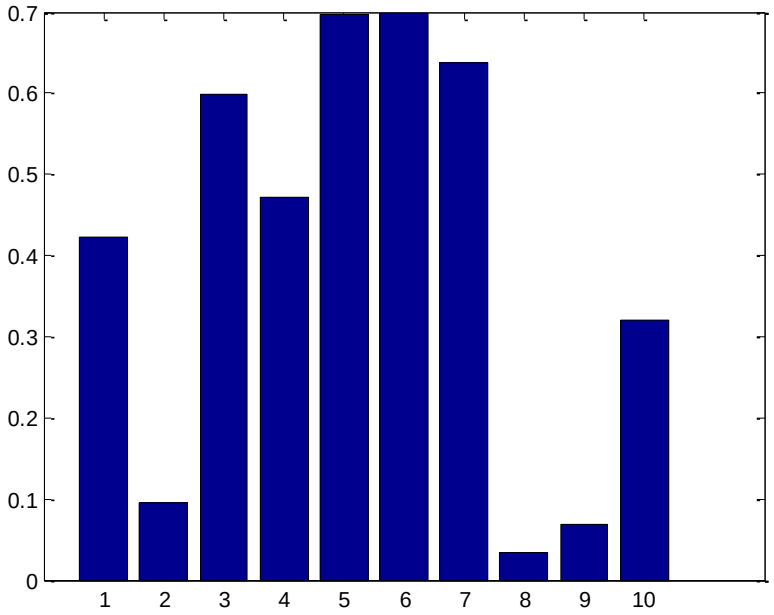
1.2 The special 2-D graphics functions

Other functions

Area	Filled area 2-D plot	**Fplot**	Plot function between specified limits
Bar	Plot bar graph	**Hist**	Histogram plot
Barh	Plot bar graph	**Pareto**	Pareto chart
Comet	2-D comet plot	**Pie**	Pie chart
Errorbar	Plot error bars along curve	**plotmatrix**	Scatter plot matrix
Ezplot	Easy-to-use function plotter	**Ribbon**	Ribbon plot
Ezpolar	Easy-to-use polar coordinate plotter	**Scatter**	Scatter plot
Feather	Plot velocity vectors	**Stem**	Plot discrete sequence data
Fill	Filled 2-D polygons	**Stairs**	Stairstep graph

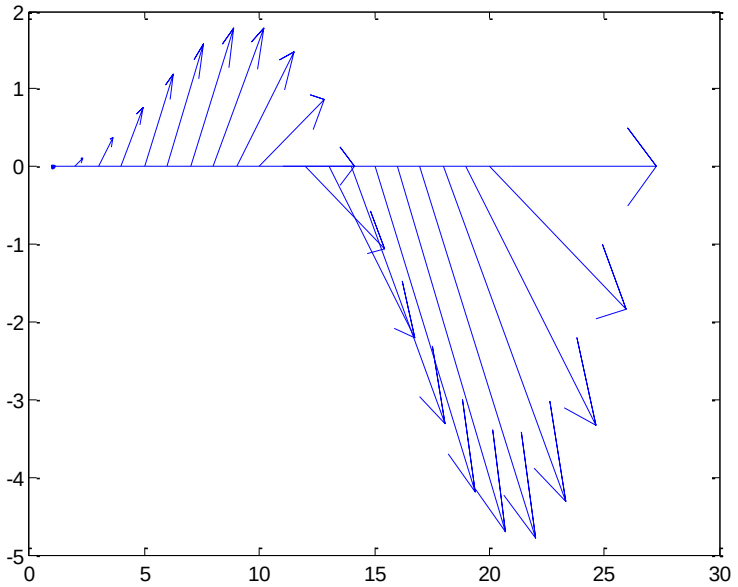
1.2 The special 2-D graphics functions

Other functions



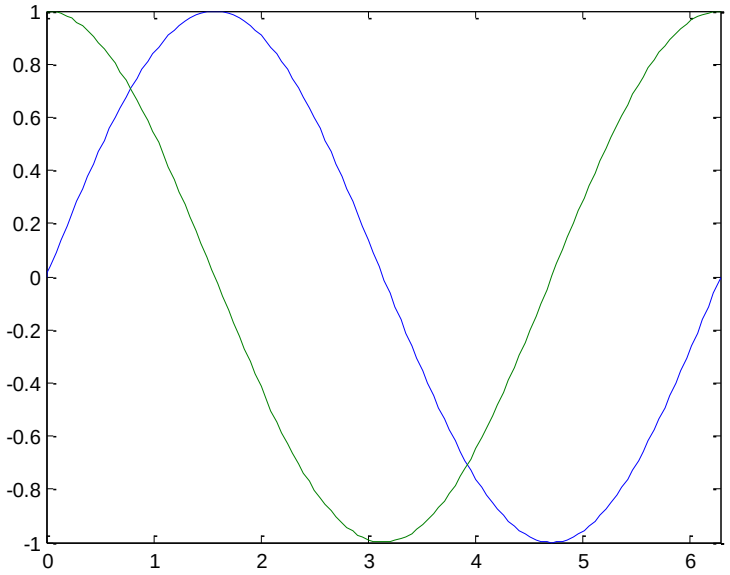

```
x=1:10;
y=rand(10,1);
figure()
bar(x,y);
```

```
x=0:0.1*pi:2*pi;
y=x.*sin(x);
figure()
feather(x,y)
```



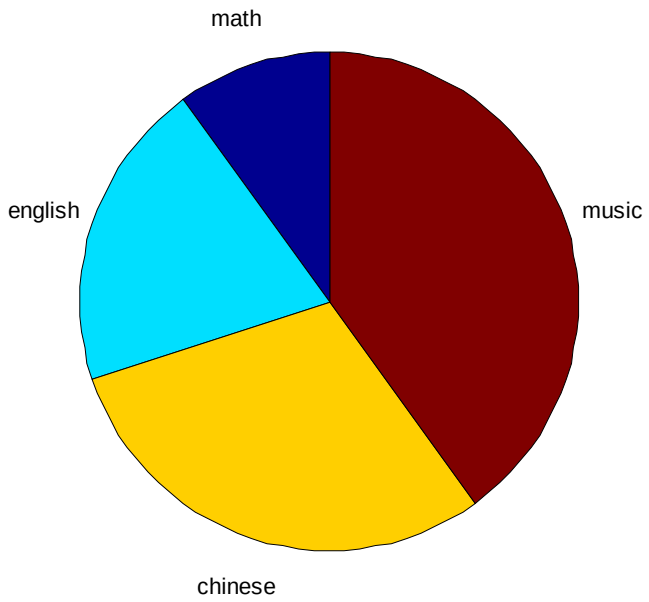
1.2 The special 2-D graphics functions

Other functions



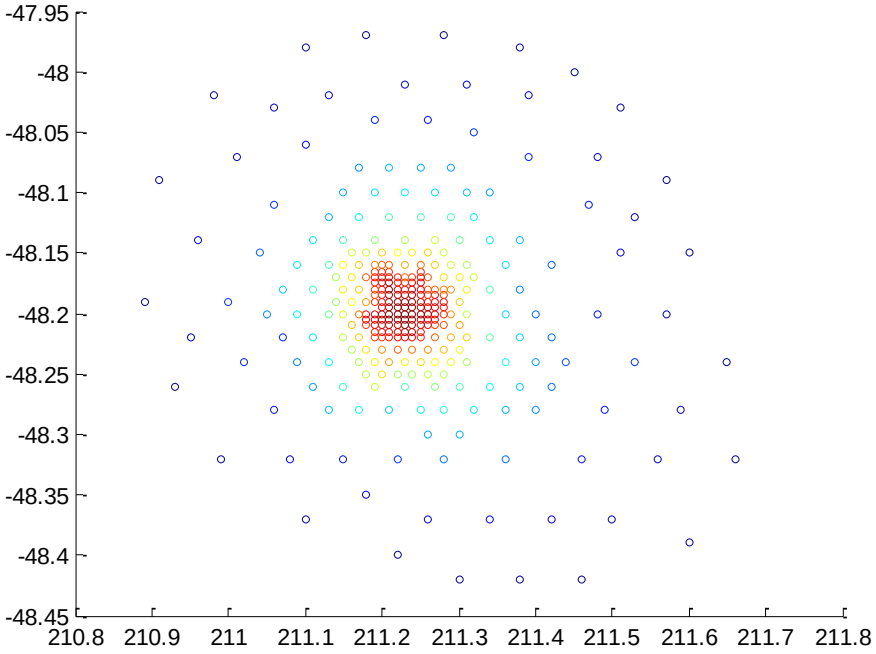

```
lim=[0,2*pi,-1,1];
fplot('[sin(x),cos(x)]',lim)
```

```
x=[2,4,6,8];
pie(x,{'math','english',
    'chinese','music'})
```



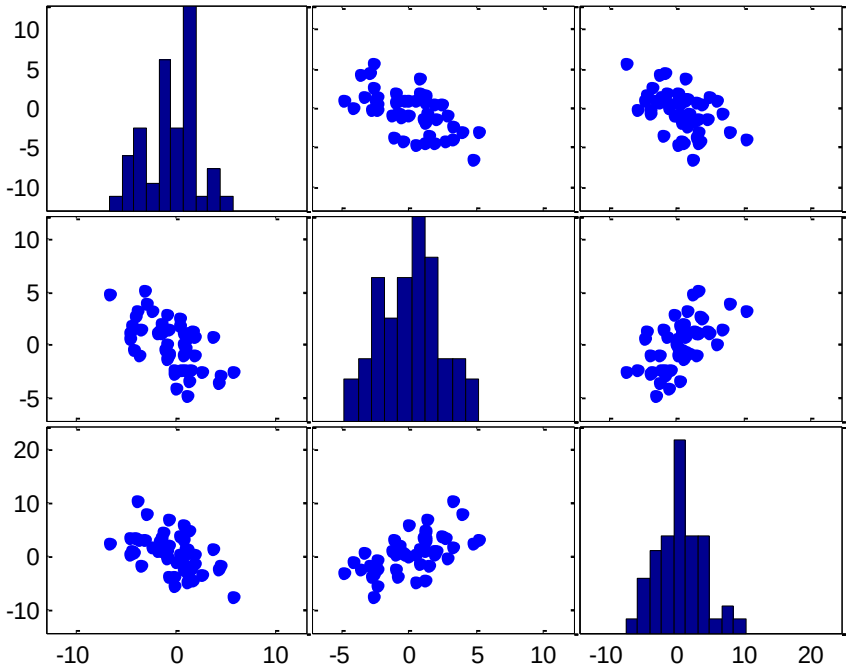
1.2 The special 2-D graphics functions

Other functions




```
load seamount

scatter(x,y,5,z)
```

```
x = randn(50,3);

y = x*[-1 2 1;2 0 1;1 -2 3;]';

plotmatrix(y)
```



1.2 The special 2-D graphics functions

`Contour` Contour plot of matrix

`contour(Z,n)` draws a contour plot of matrix `Z` with `n` contour levels.

`contour(Z,v)` draws a contour plot of matrix `Z` with contour lines at the data values specified in the monotonically increasing vector `v`.

1.2 The special 2-D graphics functions

`Contour` Contour plot of matrix

`contour(X,Y,Z,n)`

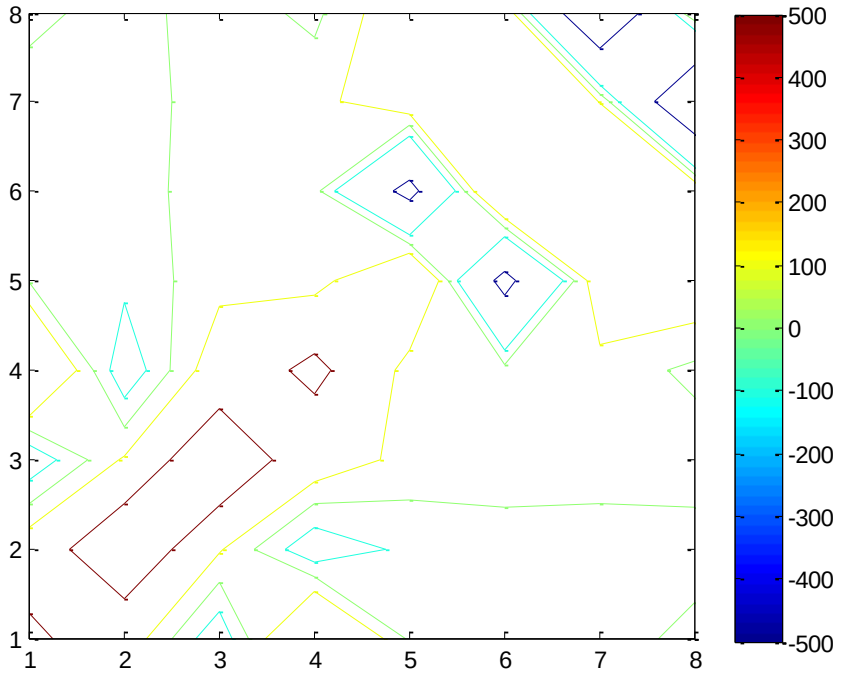
`contour(X,Y,Z,v)`

`contour(X,Y,Z)`, `contour(X,Y,Z,n)`, and `contour(X,Y,Z,v)` draw contour plots of `Z` using `X` and `Y` to determine the *x*- and *y*-axis limits. When `X` and `Y` are matrices, they must be the same size as `Z` and must be monotonically increasing.

1.2 The special 2-D graphics functions

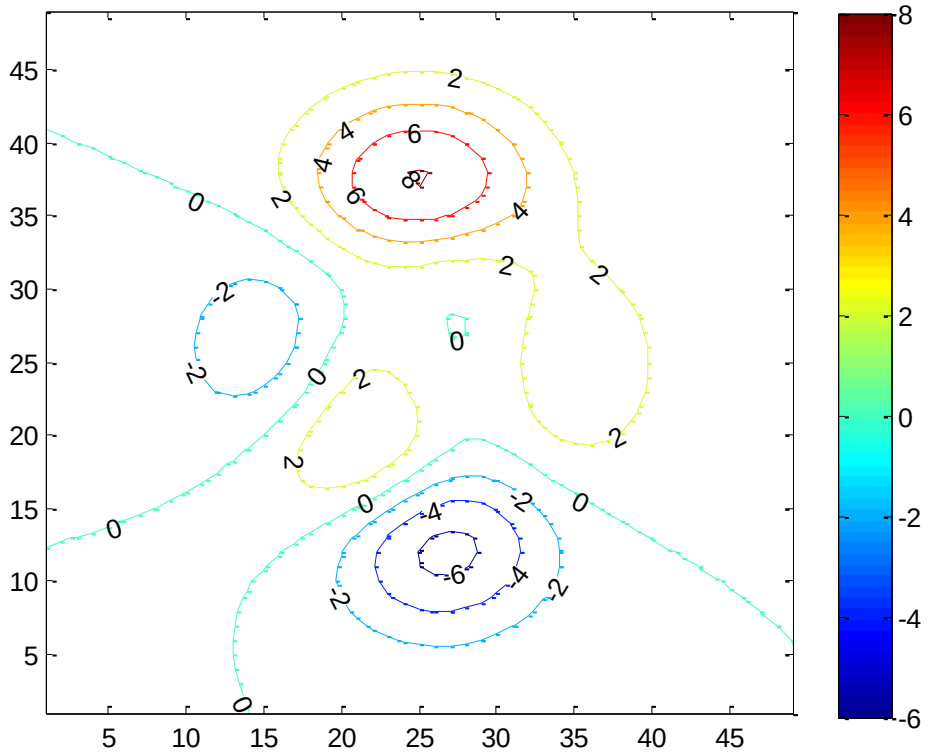
`Contour` Contour plot of matrix

```
A=rosser
v=[-1000,-500,-100, ...
0,100,500,1000];
contour(A,v)
colorbar
```



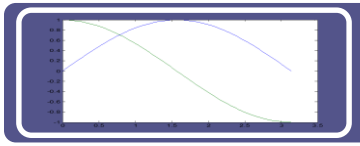
1.2 The special 2-D graphics functions

`Contour` Contour plot of matrix

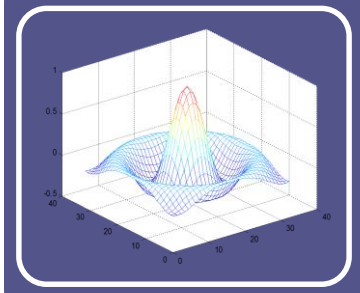



```
[c,h] = contour(peaks);
clabel(c,h), colorbar
```

Contents

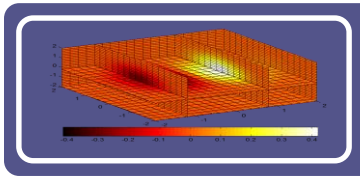


2-D graphics

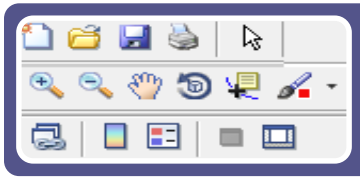


3-D graphics

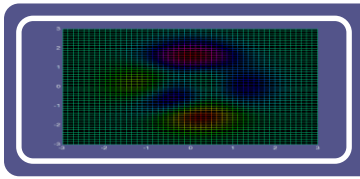
- 3-D Basic drawing commands
- special 3-D graphics functions



4-D graphics



Basic graphics technology



High-level graphics technology

2. 1 3-D Basic drawing commands

`Plot3` 3-D line plot

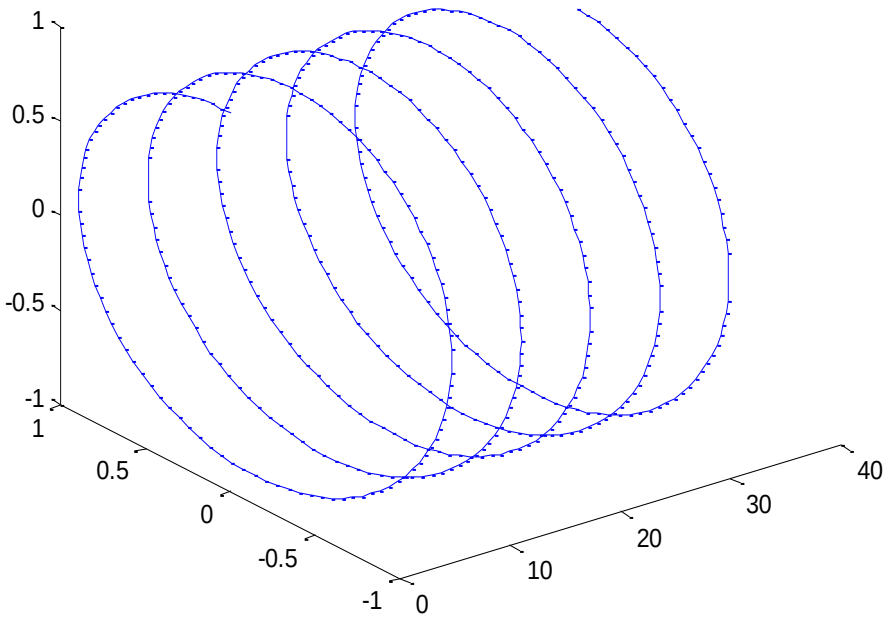
`plot3(X1,Y1,Z1,...)` where `X1, Y1, Z1` are vectors or matrices, plots one or more lines in three-dimensional space through the points whose coordinates are the elements of `X1, Y1,` and `Z1`.

`plot3(X1,Y1,Z1,LineSpec,...)` creates and displays all lines defined by the `Xn, Yn, Zn, LineSpec` quads, where `LineSpec` is a line specification that determines line style, marker symbol, and color of the plotted lines.

2. 1 3-D Basic drawing commands

`Plot3` 3-D line plot

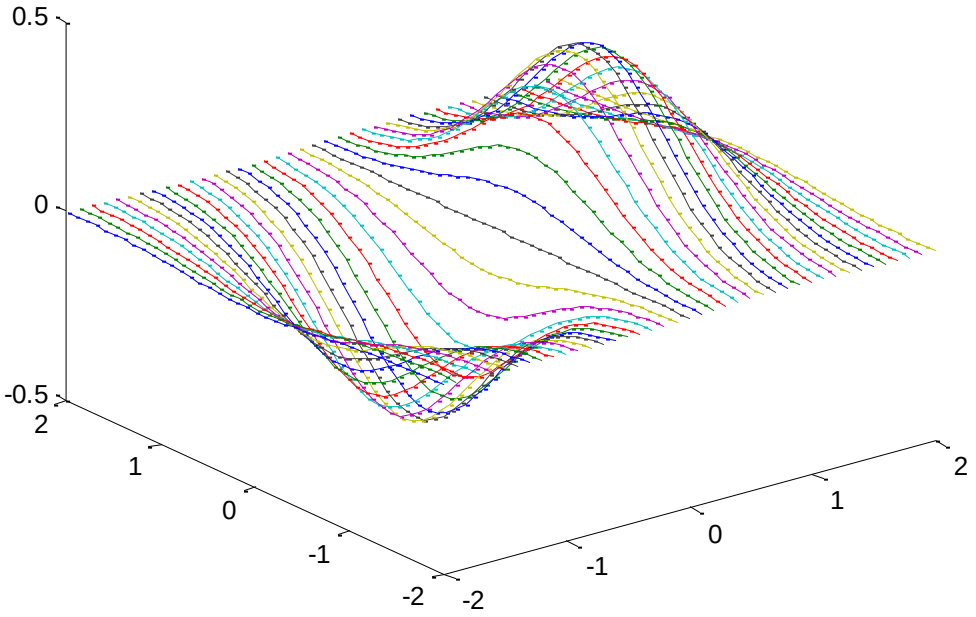
```
x=0:pi/50:10*pi;

y=sin(x);

z=cos(x);

plot3(x, y, z)
```



2. 1 3-D Basic drawing commands

Plot3　　3-D line plot

```
[x,y]=meshgrid(-2:0.1:2,-2:0.1:2);

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

plot3(x,y,z)
```



2. 1 3-D Basic drawing commands

Grid function

`Mesh`	Create mesh plot
`Meshc`	Create mesh plot with contour plot
`Meshz`	Create mesh plot with curtain plot
`meshgrid`	Generate X and Y arrays for 3-D plots

2. 1 3-D Basic drawing commands

`meshgrid` Generate X and Y arrays for 3-D plots

`[X,Y] = meshgrid(x,y)` transforms the domain specified by vectors `x` and `y` into arrays `X` and `Y`, which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. The rows of the output array `X` are copies of the vector `x`; columns of the output array `Y` are copies of the vector `y`.

2. 1 3-D Basic drawing commands

`Mesh` Mesh plots

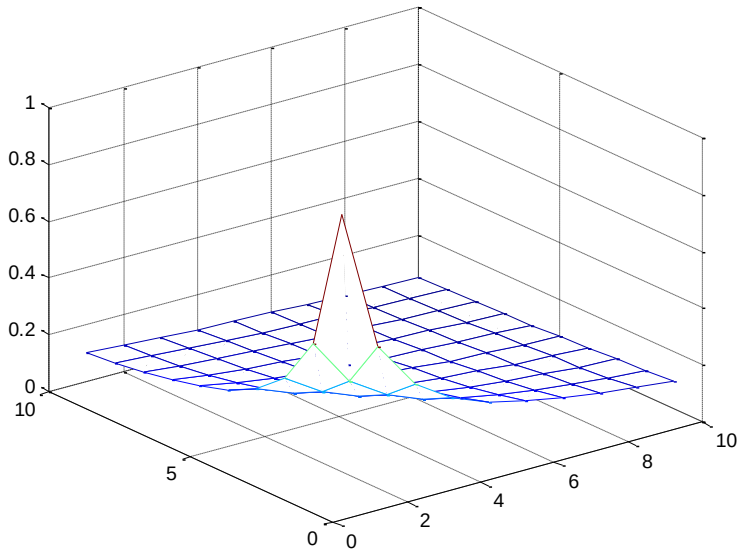
`mesh(X,Y,Z)` draws a wireframe mesh with color determined by `Z` so color is proportional to surface height. If `X` and `Y` are vectors, `length(X) = n` and `length(Y) = m`, where `[m,n] = size(Z)`. In this case, (*X*(*j*), *Y*(*i*), *Z*(*i*,*j*)) are the intersections of the wireframe grid lines; `X` and `Y` correspond to the columns and rows of `Z`, respectively. If `X` and `Y` are matrices, (*X*(*i*,*j*), *Y*(*i*,*j*), *Z*(*i*,*j*)) are the intersections of the wireframe grid lines.

2. 1 3-D Basic drawing commands

`Mesh` Mesh plots

`mesh(...,C)` draws a wireframe mesh with color determined by matrix `C`. MATLAB performs a linear transformation on the data in `C` to obtain colors from the current colormap. If `X, Y,` and `Z` are matrices, they must be the same size as `C`.

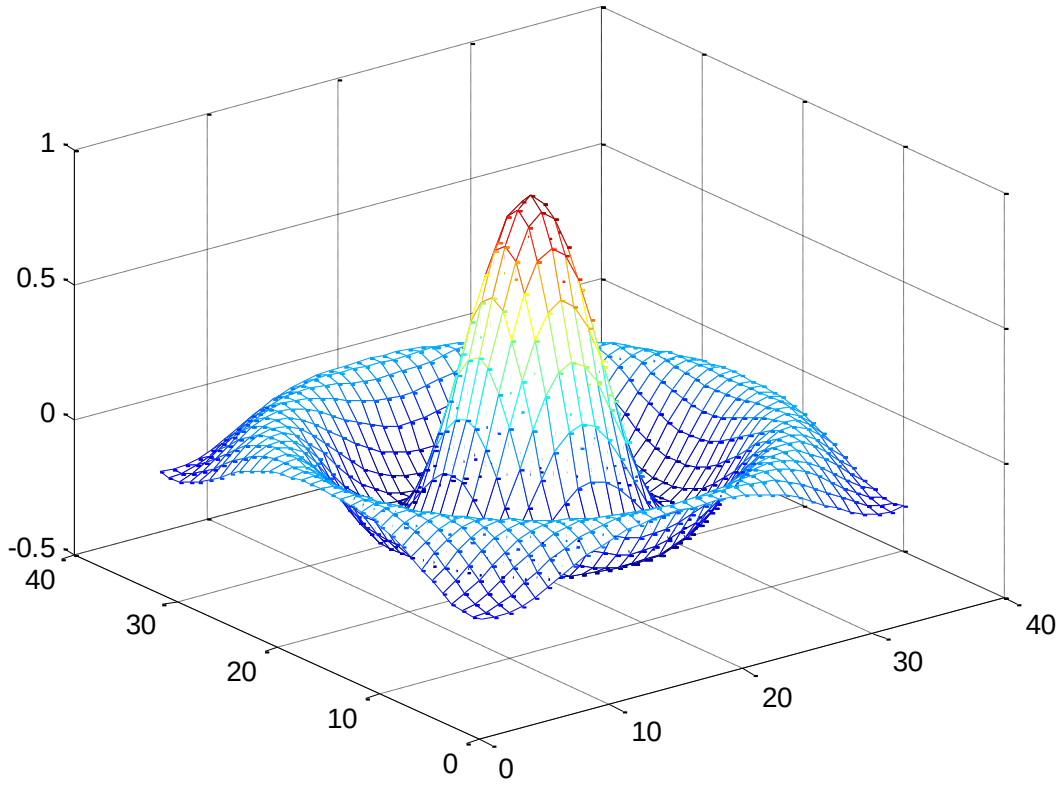
```
z=hilb(10);

mesh(z)
```



2. 1 3-D Basic drawing commands

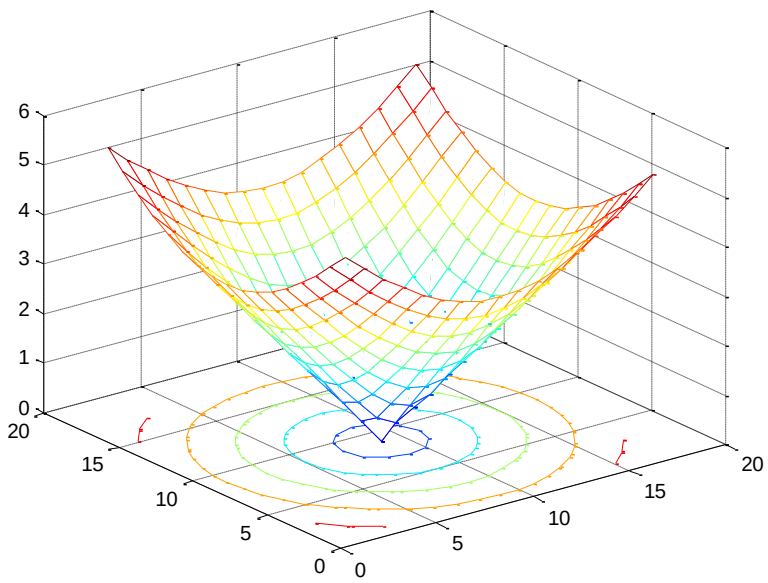
`Mesh` function

```
x=-8:0.5:8;
y=x';
a=ones(size(y))*x;
b=y*ones(size(x));
c=sqrt(a.^2+b.^2)+eps;
z=sin(c)./c;
mesh(z)
```



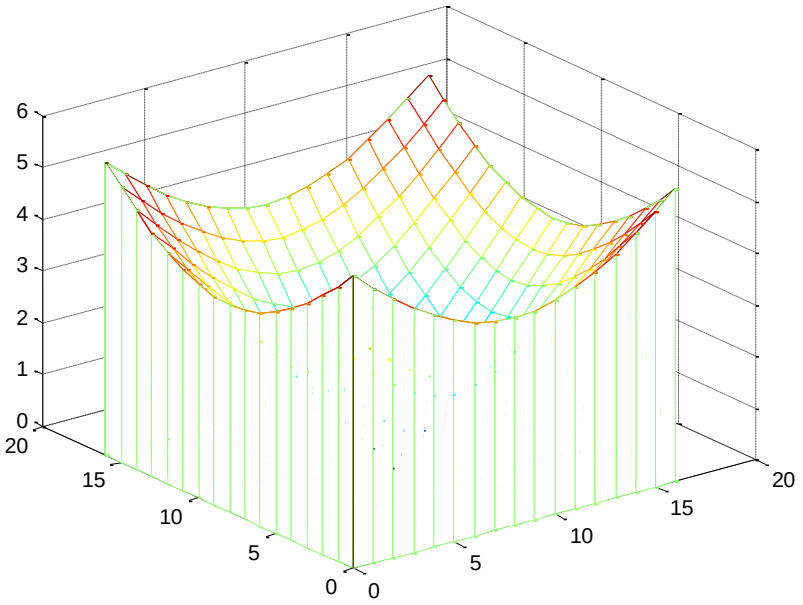
2. 1 3-D Basic drawing commands

`Grid` function




```
[X,Y]=meshgrid([-4:0.5:4]);
Z=sqrt(X.^2+Y.^2);
meshc(Z)
```

```
[X,Y]=meshgrid([-4:0.5:4]);
Z=sqrt(X.^2+Y.^2);
meshz(Z)
```



2. 1 3-D Basic drawing commands

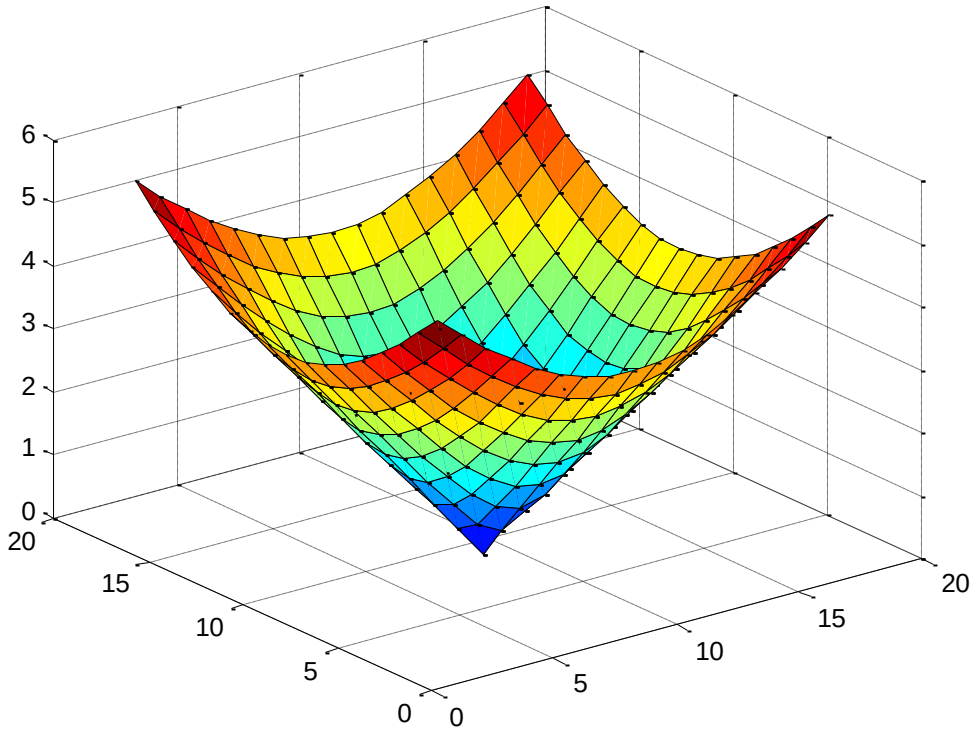
`Surf` 3-D shaded surface plot

`surf(Z)` creates a three-dimensional shaded surface from the *z* components in matrix `Z`, using `x = 1:n` and `y = 1:m`, where `[m,n] = size(Z)`. The height, `Z`, is a single-valued function defined over a geometrically rectangular grid. `Z` specifies the color data as well as surface height, so color is proportional to surface height.

2. 1 3-D Basic drawing commands

`Surf` 3-D shaded surface plot

```
[X,Y]=meshgrid([-4:0.5:4]);

Z=sqrt(X.^2+Y.^2);

surf(Z)
```



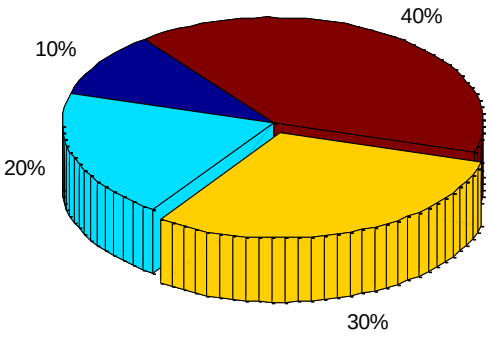
2.2 The special 3-D graphics functions

Usually used functions

`Bar3`	Plot 3-D bar chart	`Surfc`	3-D shaded surface plot
`Comet3`	3-D comet plot	`Trisurf`	Triangular surface plot
`Ezplot3`	Easy-to-use 3-D parametric curve plotter	`Trimesh`	Triangular mesh plot
`Pie3`	3-D pie chart	`Waterfall`	Waterfall plot
`Scatter3`	3-D scatter plot	`Cylinder`	Generate cylinder
`Stem3`	Plot 3-D discrete sequence data	`sphere`	Generate sphere
`tetramesh`	Triangular surface plot	`pcshow`	Plot 3-D point cloud

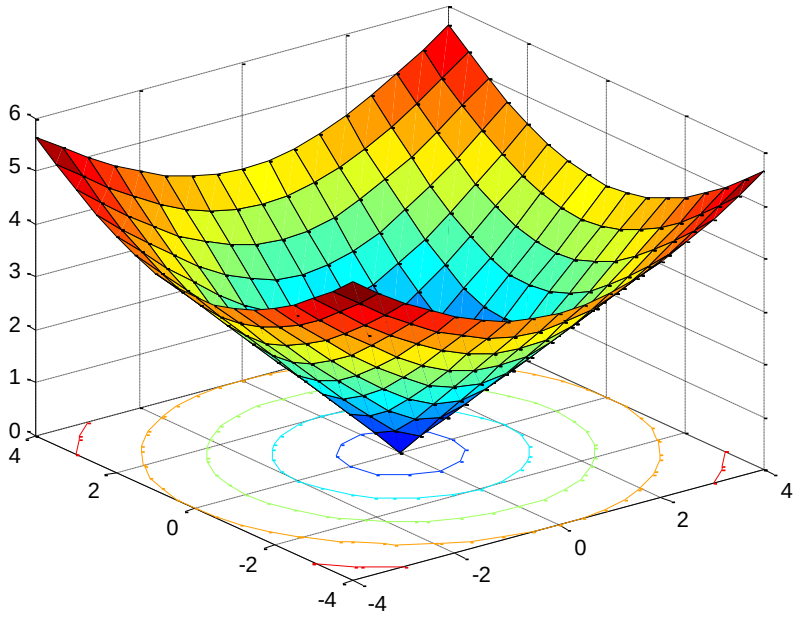
2.2 The special 3-D graphics functions

Usually used functions




```
x=[2,4,6,8];
pie3(x,[0,0,1,0])
```

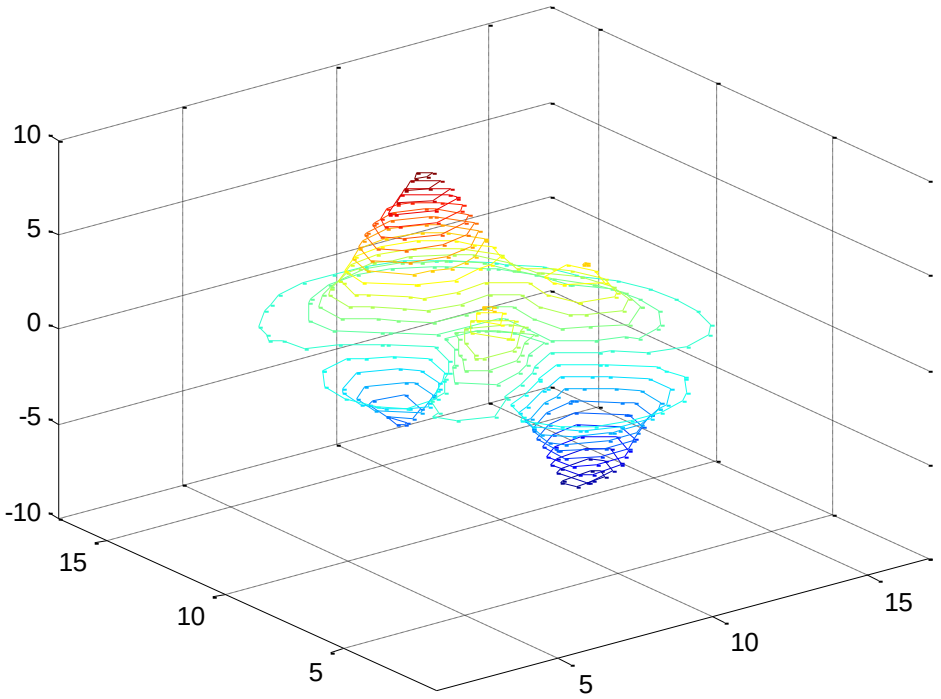
```
[X,Y]=meshgrid([-4:0.5:4]);
Z=sqrt(X.^2+Y.^2);
surfc(X,Y,Z)
```



2.2 The special 3-D graphics functions

Countour3

3-D contour plot




```
[X,Y]=meshgrid([-4:0.5:4]);

contour3(peaks(X,Y),25)
```

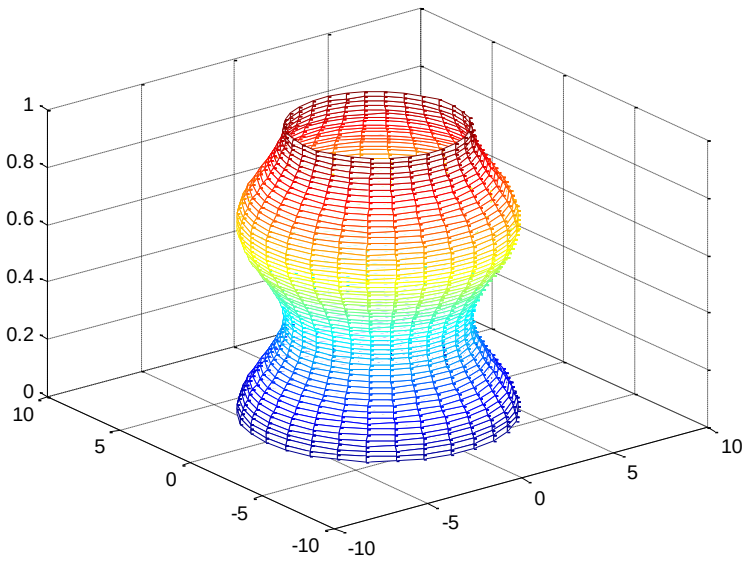
2.2 The special 3-D graphics functions

`Cylinder` Generate cylinder

`[X,Y,Z] = cylinder(r,n)`

returns the *x*-, *y*-, and *z*-coordinates of a cylinder based on the profile curve defined by vector `r`. The cylinder has `n` equally spaced points around its circumference.

```
x=0:pi/20:pi*3;
r=5+cos(x);
[a,b,c]=cylinder(r,30);
mesh(a,b,c)
```

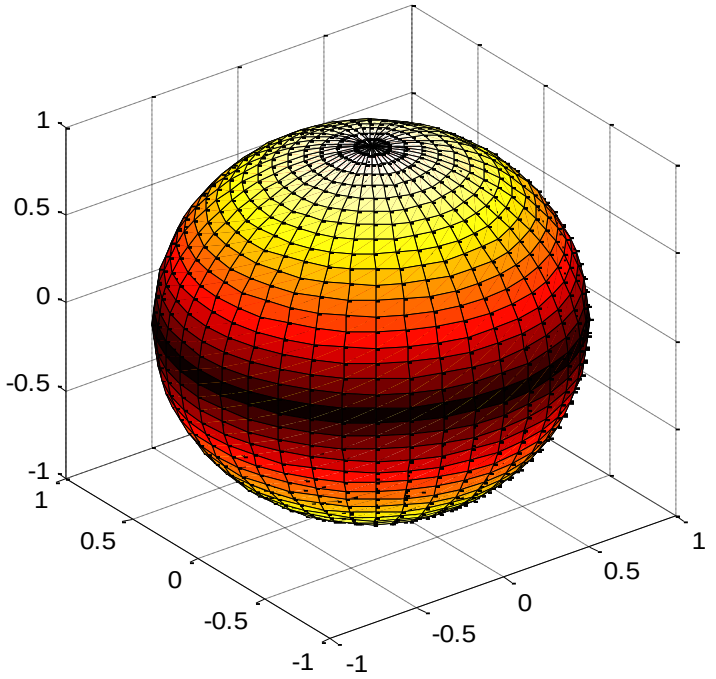


2.2 The special 3-D graphics functions

`Sphere` Generate sphere

`[X,Y,Z] = sphere(n)` returns the coordinates of a sphere in three matrices that are (n+1)-by-(n+1) in size. You draw the sphere with `surf(X,Y,Z)` or `mesh(X,Y,Z)`.

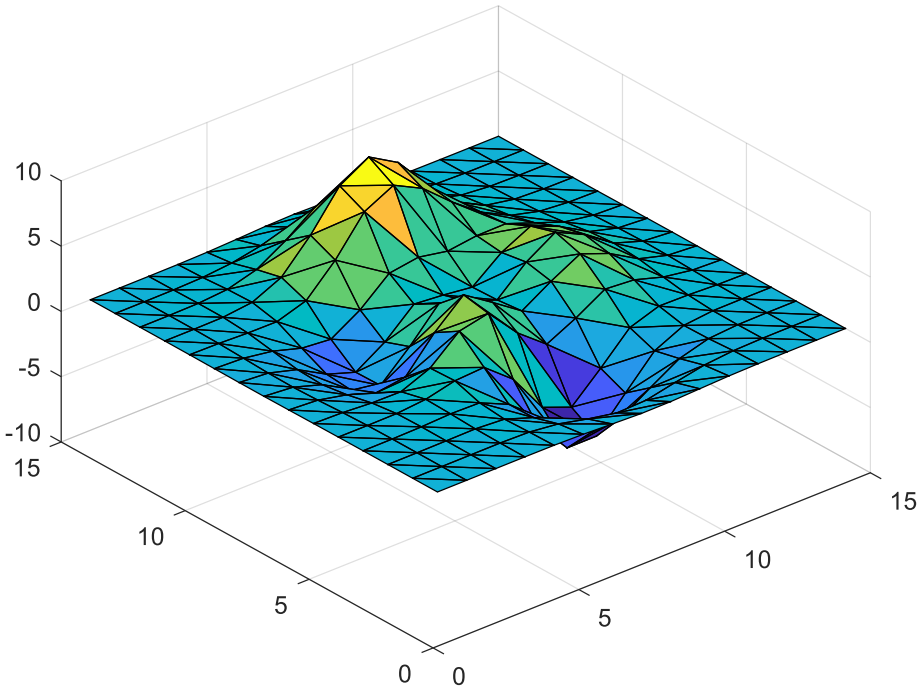
```
[a,b,c]=sphere(40);
t=abs(c);
surf(a,b,c,t);
axis('equal')
axis('square')
colormap('hot')
```



2.2 The special 3-D graphics functions

trisurf Triangular surface plot

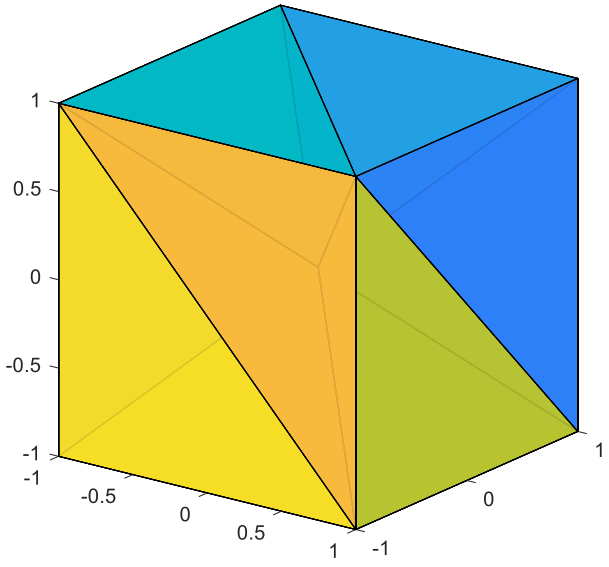
trisurf(TRI,X,Y,Z,C) displays the triangles defined in the M-by-3 face matrix TRI as a surface. A row of TRI contains indexes into the X,Y, and Z vertex vectors to define a single triangular face. The color is defined by the vector C.



2.2 The special 3-D graphics functions

tetramesh Tetrahedron mesh plot

```
tetramesh(T,X,C) displays the tetrahedra
  defined in the M-by-4 matrix T as mesh.

T is usually the output of the Delaunay
  triangulation of a set of points in 3D.
```

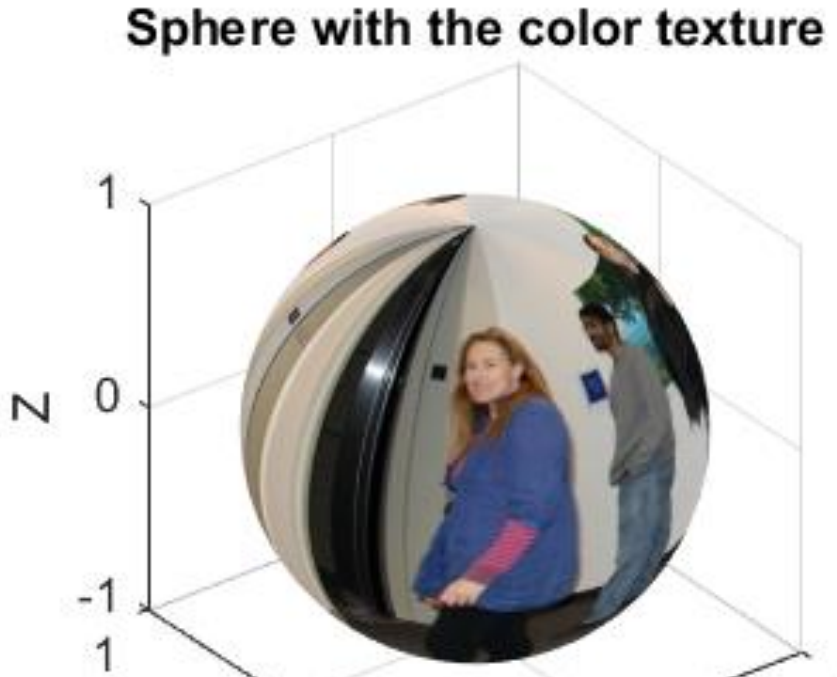


2.2 The special 3-D graphics functions

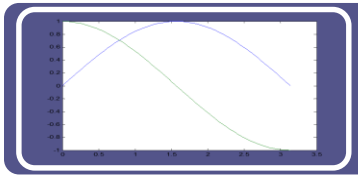
pcshow Plot 3-D point cloud.

pcshow(ptCloud) displays points with locations and colors stored in the pointCloud object ptCloud. Use this function to display static point cloud data.

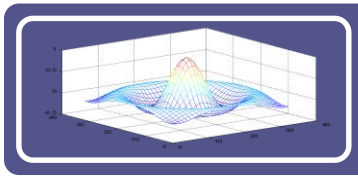
See example pcshow_test.m



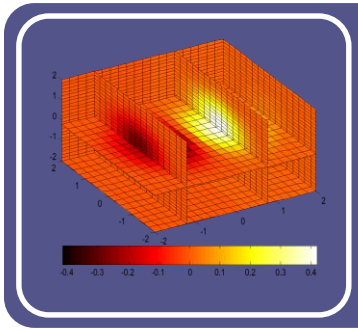
Contents



2-D graphics

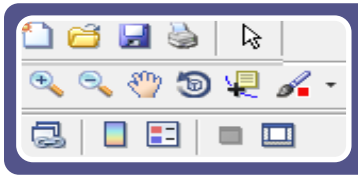


3-D graphics

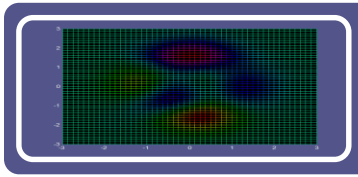


4-D graphics

- slice
- patch



Basic graphics technology



High-level graphics technology

3.1 slice

`slice` Volumetric slice plot

`slice(X,Y,Z,V,sx,sy,sz)` draws slices of the volume `V`. `X`, `Y`, and `Z` are three-dimensional arrays specifying the coordinates for `V`. `X`, `Y`, and `Z` must be monotonic and orthogonally spaced (as if produced by the function meshgrid). The color at each point is determined by 3-D interpolation into the volume `V`.

3.1 slice

`slice` Volumetric slice plot

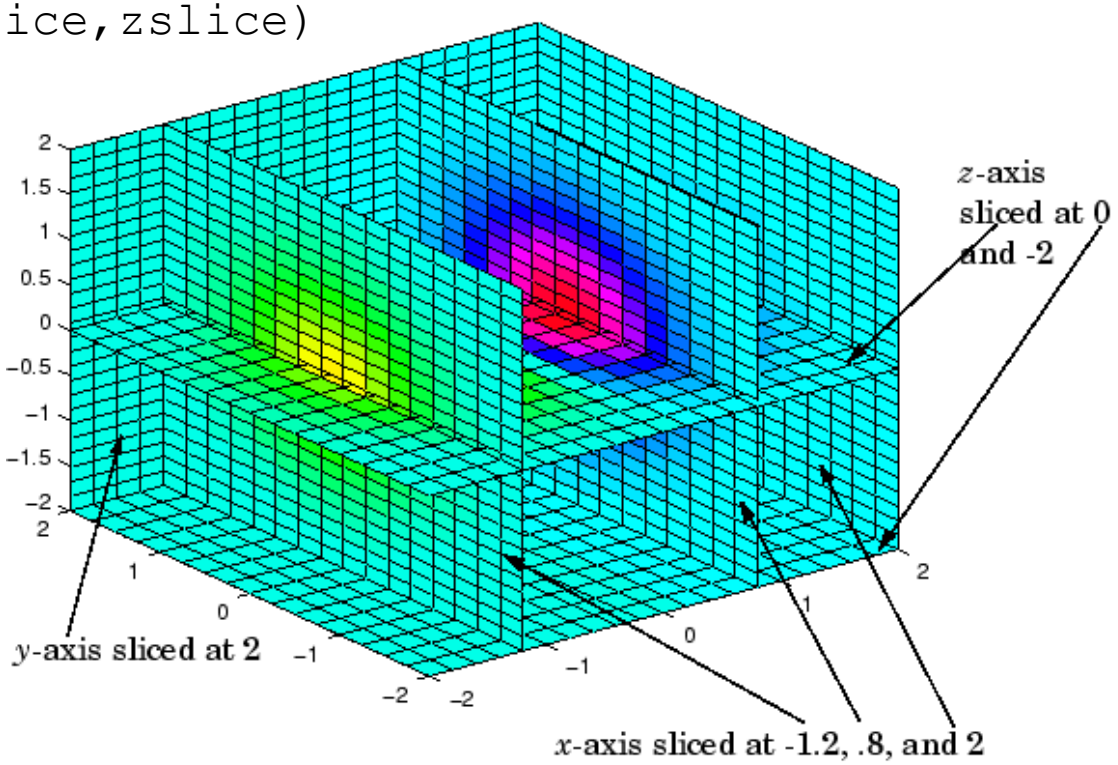
`slice(X,Y,Z,V,XI,YI,ZI)` draws slices through the volume `V` along the surface defined by the arrays `XI, YI, ZI`.

- `slice(V,sx,sy,sz)`
- `slice(V,sx,sy,sz)`
- `slice(...,'method')`

3.1 slice

`slice` Volumetric slice plot

```
[x,y,z] = meshgrid(-2:.2:2,-2:.25:2,-2:.16:2);
v = x.*exp(-x.^2-y.^2-z.^2);
xslice = [-1.2,.8,2];
yslice = 2;
zslice = [-2,0];
slice(x,y,z,v,xslice,yslice,zslice)
colormap hsv
```



3.2 patch

`patch`

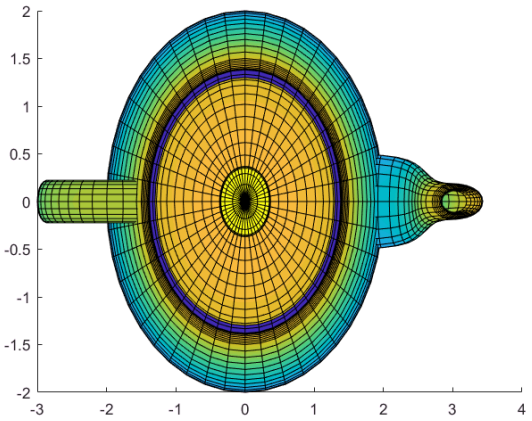
A patch graphics object is composed of one or more polygons that may or may not be connected.

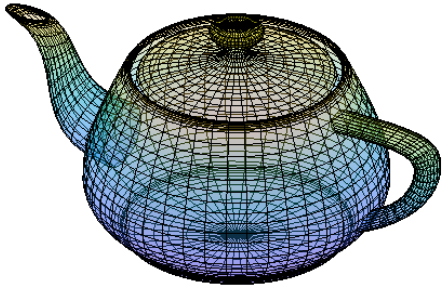
Patches are useful for modeling real-world objects such as airplanes or automobiles, and for drawing 2- or 3-D polygons of arbitrary shape.

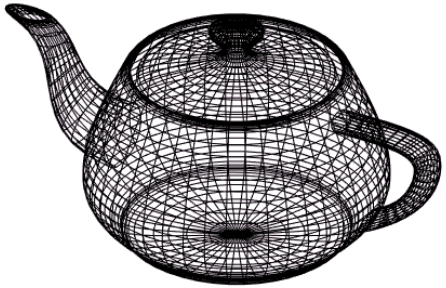
- patch(x-coordinates,y-coordinates,z-coordinates,colordata)

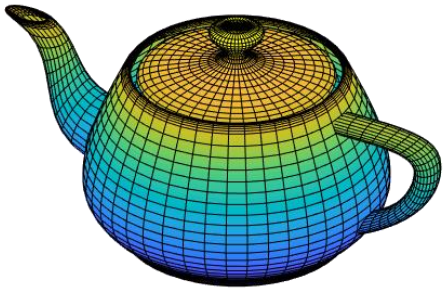
```
openExample('matlab_featured/Displaying3DObjectExample')
```

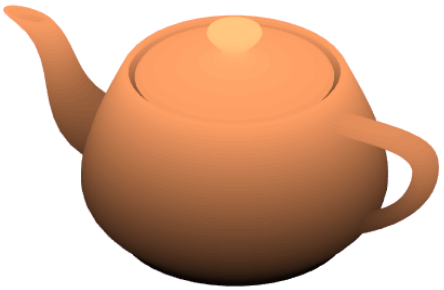
3.2 patch



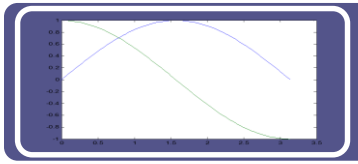




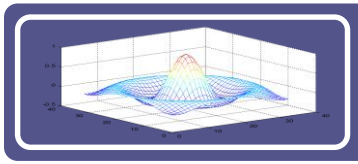




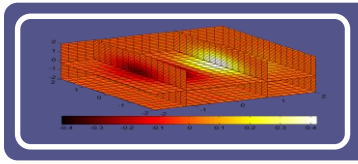
Contents



2-D graphics



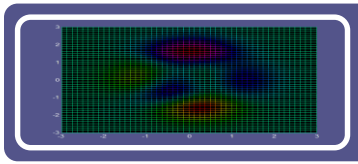
3-D graphics



4-D graphics

Basic graphics technology

- Graphic control
- The graphics mark
- Graphics with sub-plots



High-level graphics technology

4.1 Graphic control

`axis` Axis scaling and appearance

`axis([xmin xmax ymin ymax])`

sets the limits for the *x*- and *y*-axis of the current axes.

`axis([xmin xmax ymin ymax zmin zmax cmin cmax])`

sets the *x*-, *y*-, and *z*-axis limits and the color scaling limits of the current axes.

4.1 Graphic control

`axis` Axis scaling and appearance

`auto`	sets MATLAB default behavior to compute the current axes limits automatically, based on the minimum and maximum values of x, y, and z data.
`manual`	freezes the scaling at the current limits, so that if hold is on, subsequent plots use the same limits.
`tight`	sets the axis limits to the range of the data.
`fill`	sets the axis limits and PlotBoxAspectRatio so that the axes fill the position rectangle.
`ij`	places the coordinate system origin in the upper left corner.
`xy`	draws the graph in the default Cartesian axes format with the coordinate system origin in the lower left corner.
`equal`	sets the aspect ratio so that the data units are the same in every direction.

4.1 Graphic control

`axis` Axis scaling and appearance

`image`	the same as axis equal except that the plot box fits tightly around the data.
`square`	makes the current axes region square (or cubed when three-dimensional).
`normal`	automatically adjusts the aspect ratio of the axes and the relative scaling of the data units so that the plot fits the figure's shape as well as possible.
`vis3d`	freezes aspect ratio properties to enable rotation of 3-D objects and overrides stretch-to-fill.
`off`	turns off all axis lines, tick marks, and labels.
`on`	turns on all axis lines, tick marks, and labels.

4.1 Graphic control

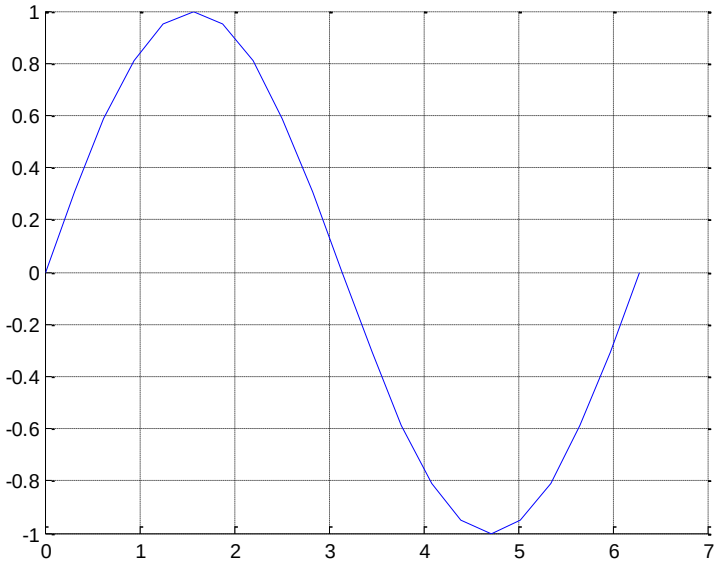
`zoom` Turn zooming on or off or magnify by factor

	toggles the interactive zoom status between off and on.
`(factor)`	zooms in or out by the specified zoom factor, without affecting the interactive zoom mode.
`on`	turns on interactive zooming.
`off`	turns interactive zooming off.
`out`	returns the plot to its initial zoom setting.
`xon`	set zoom on for the *x*-axis, respectively.
`yon`	set zoom on for the *y*-axis, respectively.
`reset`	remembers the current zoom setting as the initial zoom setting.

4.1 Graphic control

`grid`	Grid lines for 2-D and 3-D plots
`grid on`	adds major grid lines to the current axes.
`grid off`	removes major and minor grid lines from the current axes.

```
x=0:0.1*pi:2*pi;
y=sin(x);
plot(x,y);
grid on
```



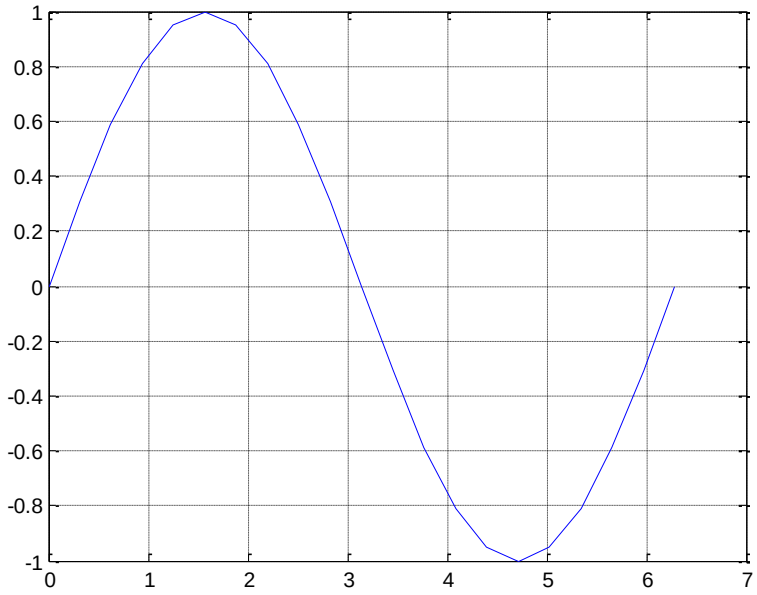
4.1 Graphic control

`box` Axes border

`box on` displays the boundary of the current axes.

`box off` doesn't display the boundary of the current axes.

```
box on
x=0:0.1*pi:2*pi;
y=sin(x);
plot(x,y);
grid on
```

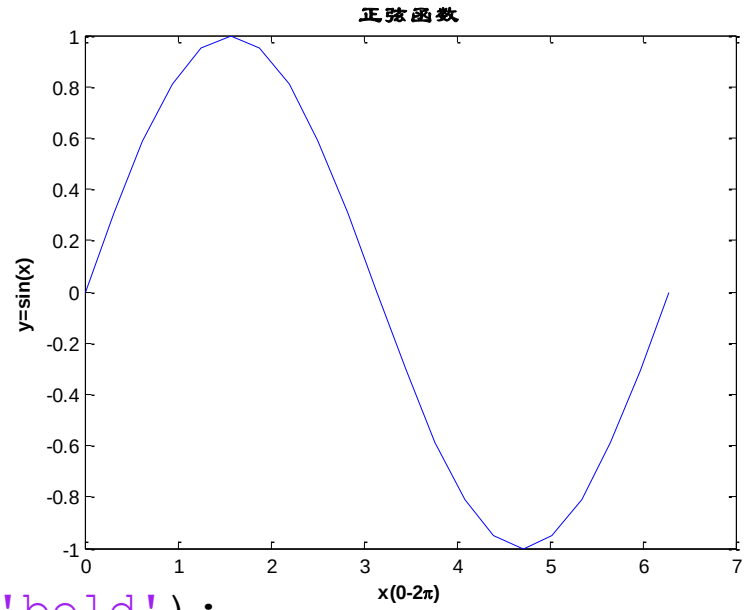


4.2 The graphics mark

`title` Add title to current axes

`xlabel, ylabel, zlabel` Label *x*-, *y*-, and *z*-axis

```
title(...,'PropertyName',PropertyValue,...)
```




```
x=0:0.1*pi:2*pi;y=sin(x);
plot(x,y);
xlabel('x(0-2\pi)','FontWeight','bold');
ylabel('y=sin(x)','FontWeight','bold');
title('正弦函数','FontSize',12,'FontWeight','bold','FontName','隶书')
```

4.2 The graphics mark

`title xlabel, ylabel, zlabel`

Character Sequence	Symbol	Character Sequence	Symbol	Character Sequence	Symbol
\alpha	α	\upsilon	υ	\sim	~
\beta	β	\phi	Φ	\leq	≤
\gamma	γ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	♣
\epsilon	ε	\omega	ω	\diamondsuit	♦
\zeta	ζ	\Gamma	Γ	\heartsuit	♥
\eta	η	\Delta	Δ	\spadesuit	♠
\theta	Θ	\Theta	Θ	\leftrightarrow	↔
\vartheta	ϑ	\Lambda	Λ	\leftarrow	←
\iota	ι	\Xi	Ξ	\uparrow	↑
\kappa	κ	\Pi	Π	\rightarrow	→

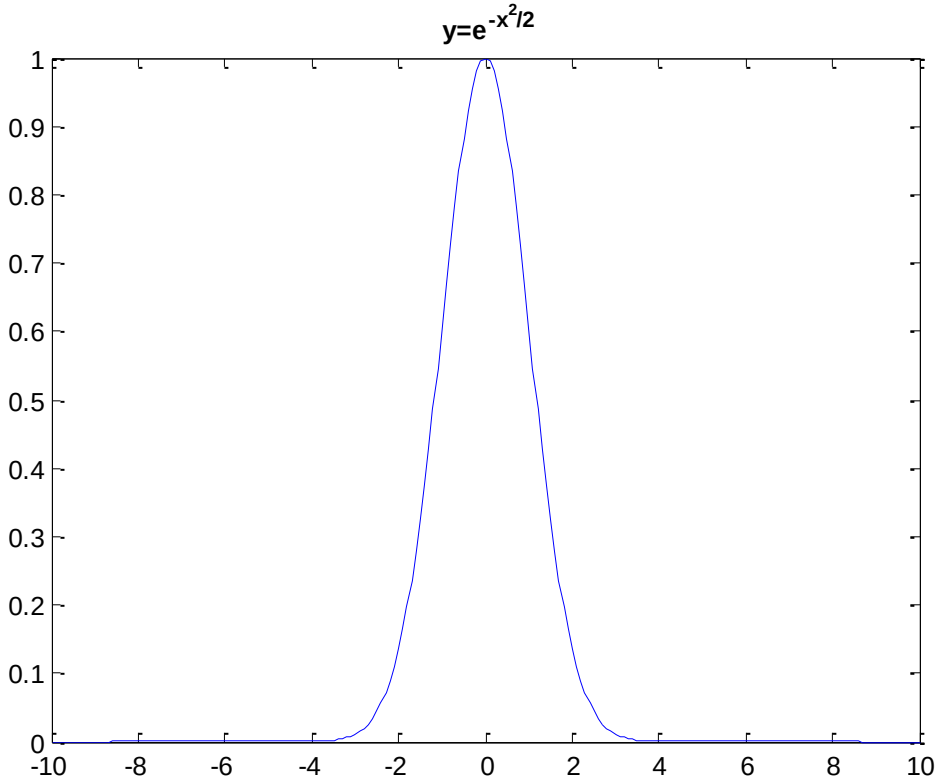
4.2 The graphics mark

`title, xlabel, ylabel, zlabel`

- `\bf` — Bold font
- `\it` — Italic font
- `\sl` — Oblique font (rarely available)
- `\rm` — Normal font
- `\fontname{`*`fontname`*`}` — Specify the name of the font family to use.
- `\fontsize{`*`fontsize`*`}` — Specify the font size in FontUnits.
- `\color(`*`colorSpec`*`)` — Specify color for succeeding characters

4.2 The graphics mark

`title, xlabel, ylabel, zlabel`

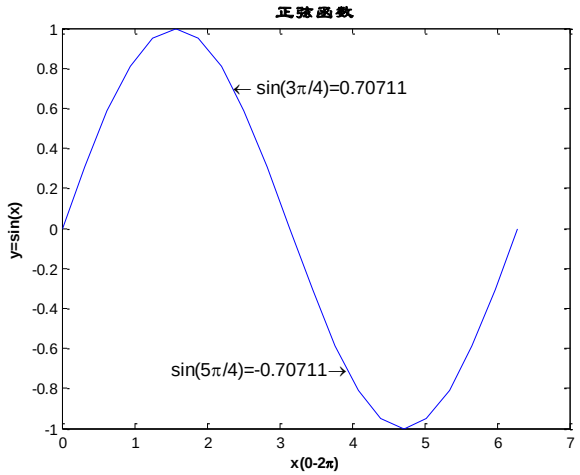



```
x=-10:0.1:10;
y=exp(-x.^2/2);
plot(x,y,'-');
title('\bf y=e^{-x^{2}/2}');
```

4.2 The graphics mark

`text` Create text object in current axes

`text(x,y,'string')` adds the string in quotes to the location specified by the point (x,y) `x` and `y` must be numbers of class double.

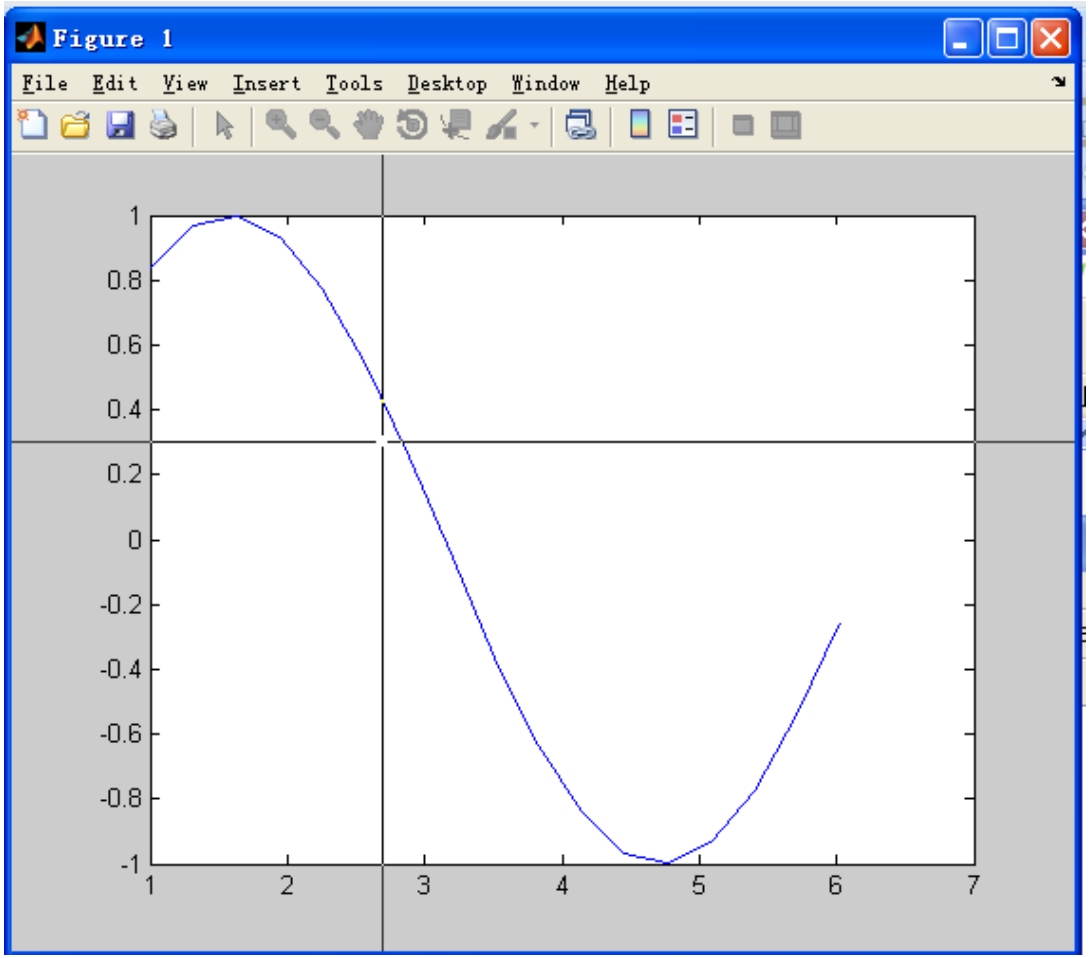



```
text(3*pi/4,sin(3*pi/4),['\leftarrow sin(3\pi/4)=', …
num2str(sin(3*pi/4))],'FontSize',12)
text(5*pi/4,sin(5*pi/4),['sin(5\pi/4)=',num2str(sin(5*pi/4)), …
'\rightarrow'],'HorizontalAlignment','right','FontSize',12)
```

4.2 The graphics mark

`gtext` Mouse placement of text in 2-D view

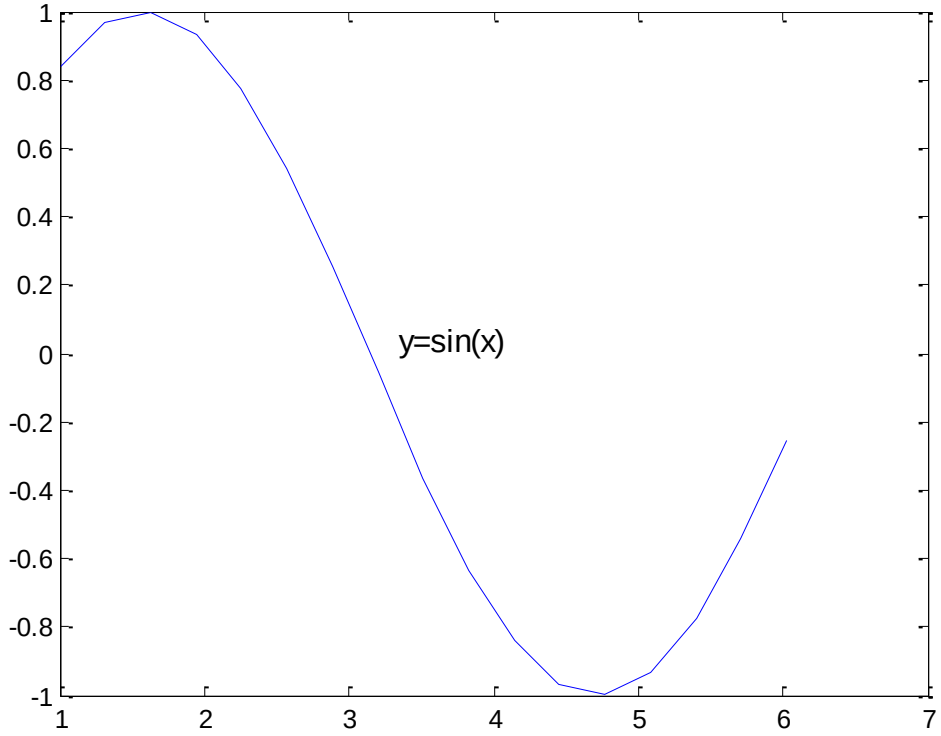
```
x=1:0.1*pi:2*pi;

y=sin(x);

plot(x,y)

gtext('y=sin(x)', …

'FontSize',12)
```



4.2 The graphics mark

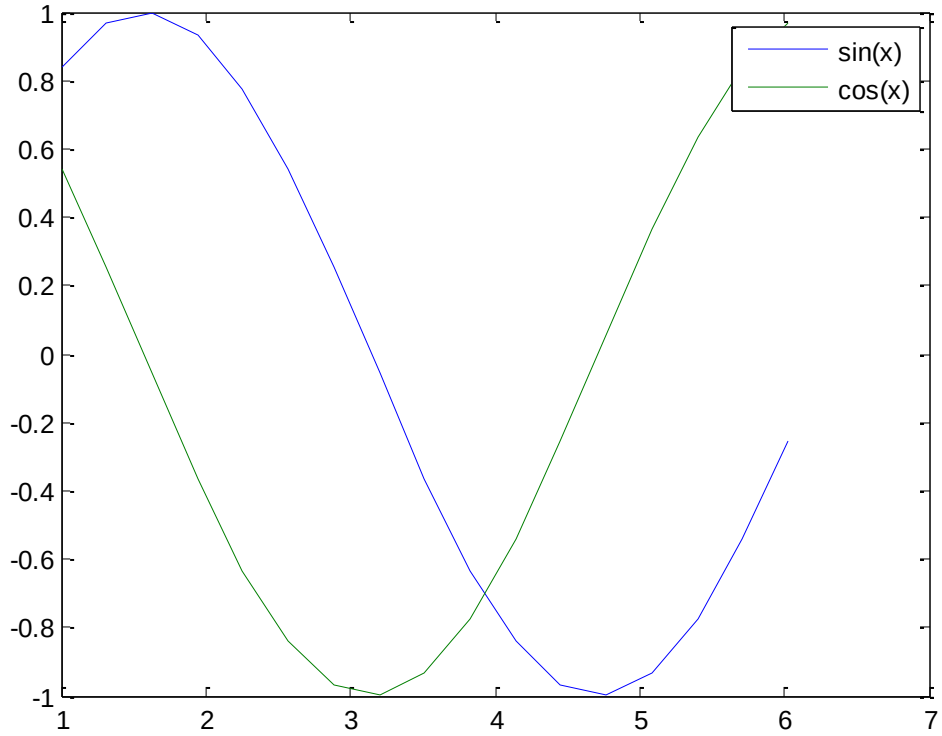
`gtext` Mouse placement of text in 2-D view

```
x=1:0.1*pi:2*pi;

y=sin(x);

plot(x,y)

gtext('y=sin(x)', …

'FontSize',12)
```



4.2 The graphics mark

`legend` Graph legend for lines and patches




```
y=sin(x);

z=cos(x);

plot(x,y,x,z)

legend('sin(x)','cos(x)')
```

4.3 Graphics with sub-plots

`hold` Retain current graph in figure

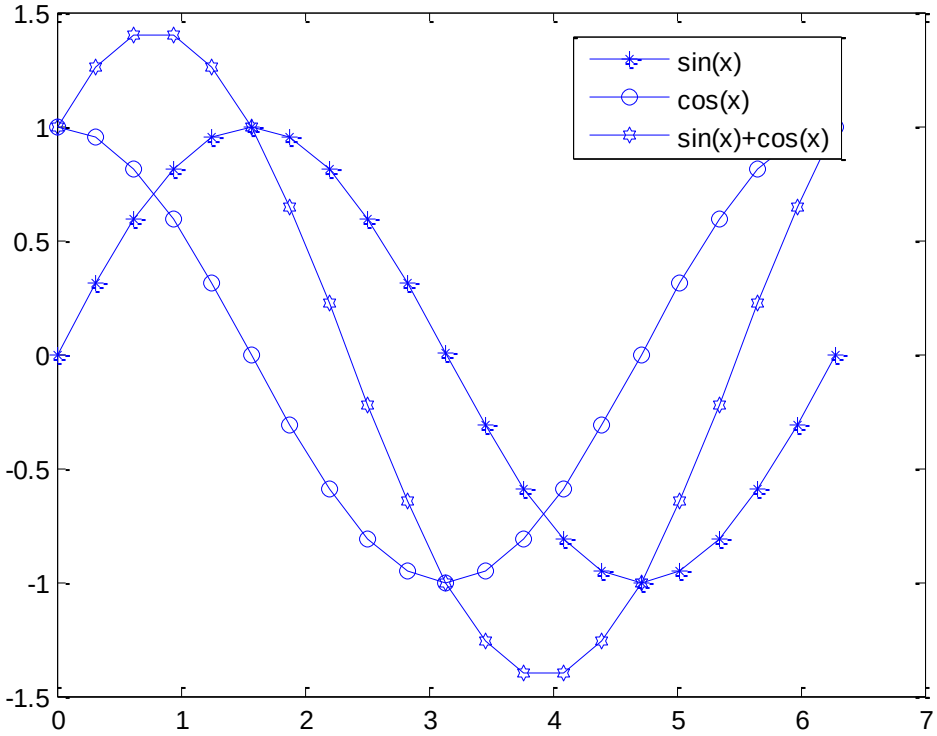
`hold on` retains the current plot and certain axes properties so that subsequent graphing commands add to the existing graph.

`hold off` resets axes properties to their defaults before drawing new plots. hold off is the default.

4.3 Graphics with sub-plots

`hold` Retain current graph in figure

```
x=0:0.1*pi:2*pi;
y=sin(x);
z=cos(x);
plot(x,y,'-*')
hold on
plot(x,z,'-o')
plot(x,y+z,'-h')
legend ('sin(x)','cos(x)','sin(x)+cos(x)',0)
hold off
```



4.3 Graphics with sub-plots

`subplot` Create axes in tiled positions

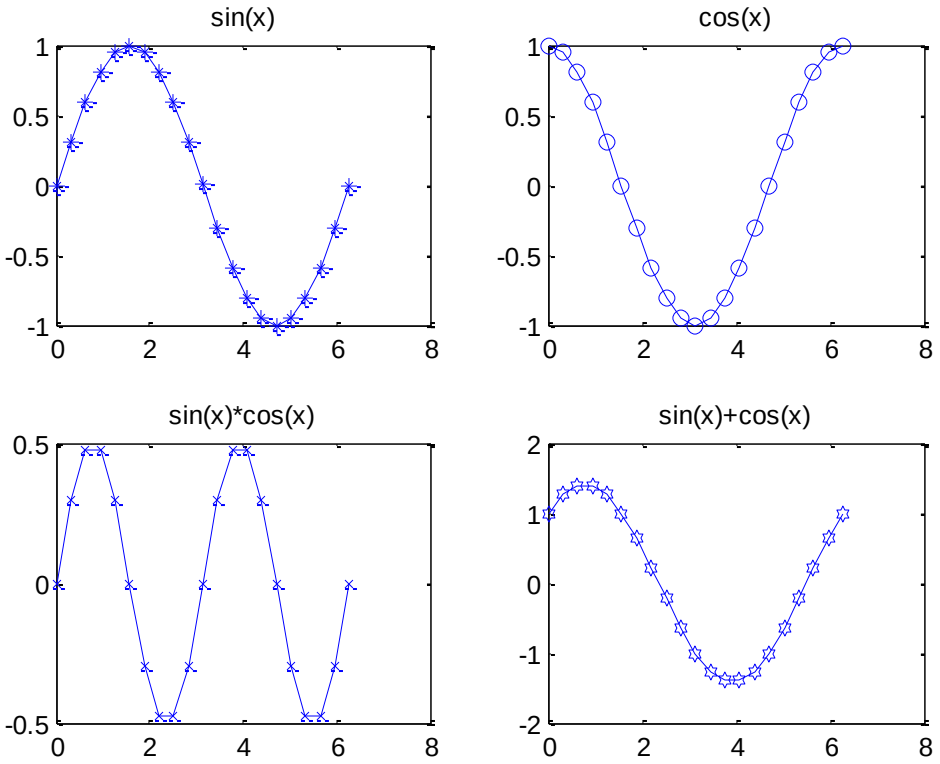
`subplot(m,n,p)`

`h = subplot(m,n,p)` or `subplot(m,n,p)` breaks the figure window into an m-by-n matrix of small axes, selects the pth axes object for the current plot, and returns the axes handle. The axes are counted along the top row of the figure window, then the second row, etc.

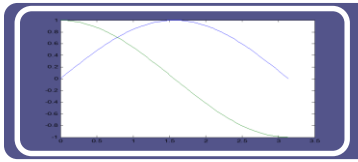
4.3 Graphics with sub-plots

`subplot` Create axes in tiled positions

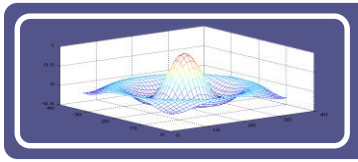
```
x=0:0.1*pi:2*pi;
subplot(2,2,1);
plot(x,sin(x),'-*');
title('sin(x)');
subplot(2,2,2);
plot(x,cos(x),'-o');
title('cos(x)');
subplot(2,2,3);
plot(x,sin(x).*cos(x),'-x');
title('sin(x)*cos(x)');
subplot(2,2,4);
plot(x,sin(x)+cos(x),'-h');
title('sin(x)+cos(x)');
```



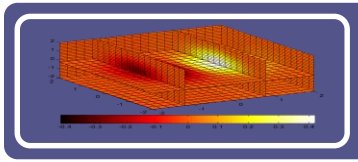
Contents



2-D graphics

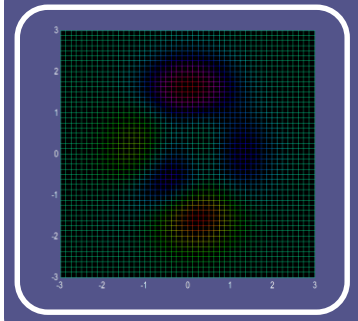


3-D graphics



4-D graphics

Basic graphics technology



High-level graphics technology

- Color image
- Perspective and light
- Image Processing
- Graphic output
- Get and set function

5.1 Color image

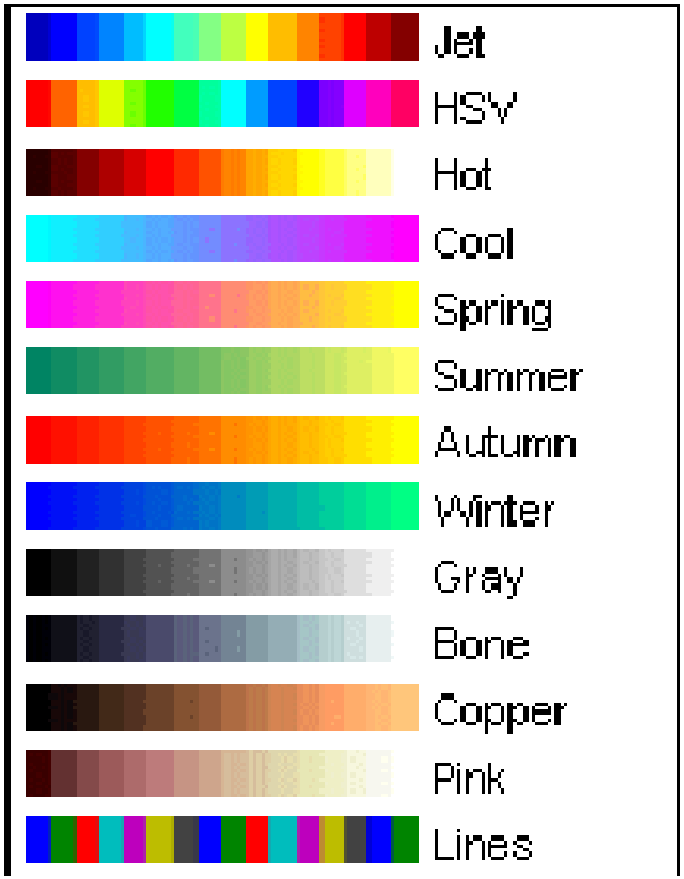
RGB Color Components

This table lists some representative RGB color definitions.

Red	Green	Blue	Color
0	0	0	**Black**
1	1	1	**White**
1	0	0	**Red**
0	1	0	**Green**
0	0	1	**Blue**
1	1	0	**Yellow**
1	0	1	**Magenta**
0	1	1	**Cyan**
0.5	0.5	0.5	**Gray**
0.5	0	0	**Dark red**
1	0.62	0.40	**Copper**
0.49	1	0.83	**Aquamarine**

5.1 Color image

`colormap` Set and get current colormap The named built-in colormaps are the following:




```
>> colormap(pink(8))
>> pink(8)

ans =

    0.3333         0         0
    0.5634    0.3086    0.3086
    0.7237    0.4364    0.4364
    0.7868    0.6299    0.5345
    0.8452    0.7766    0.6172
    0.8997    0.8997    0.6901
    0.9512    0.9512    0.8591
    1.0000    1.0000    1.0000

fx >>
```

5.1 Color image

`pcolor` Pseudocolor (checkerboard) plot

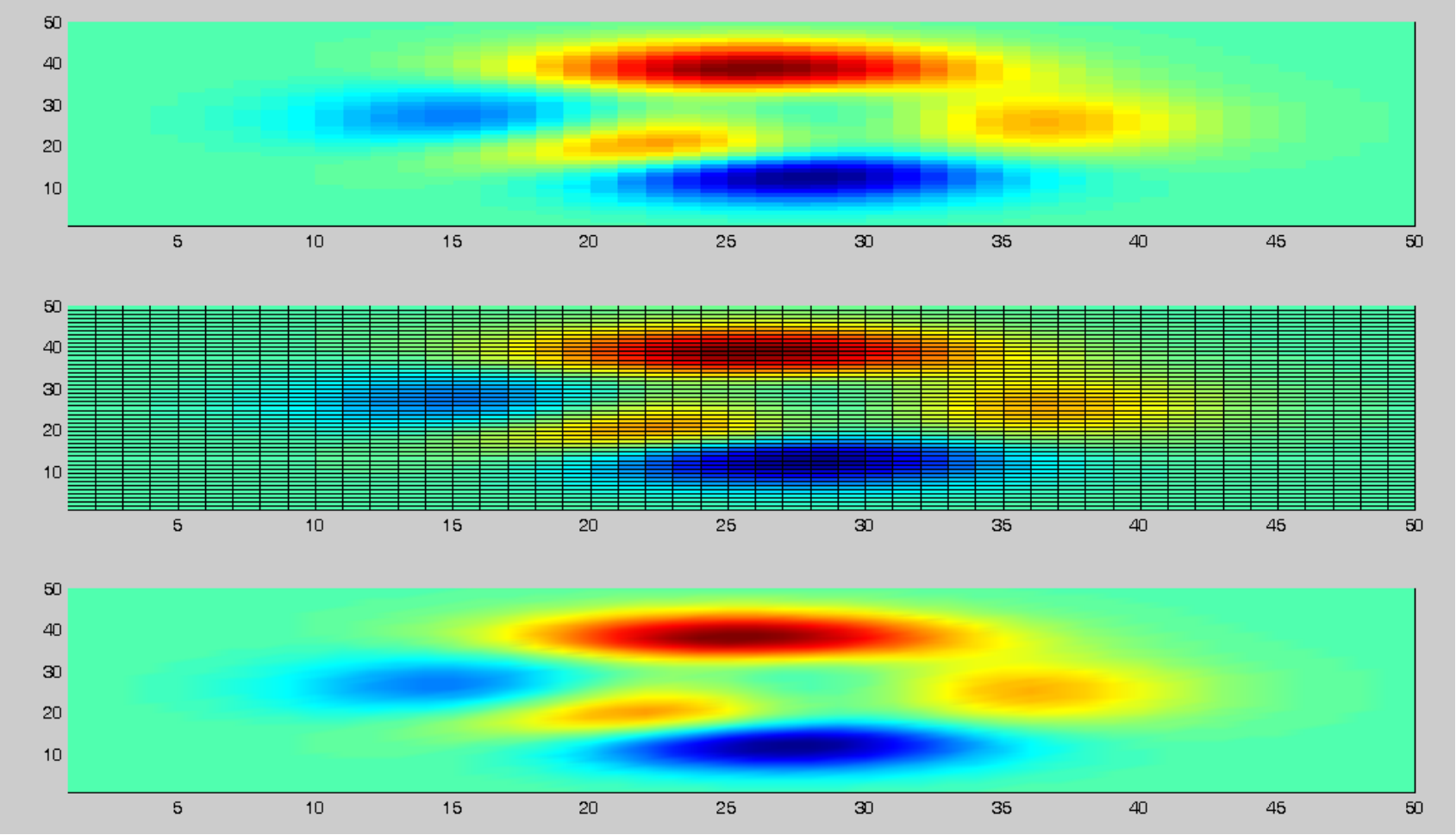
`pcolor(C)` draws a pseudocolor plot. The elements of `C` are linearly mapped to an index into the current colormap. The mapping from `C` to the current colormap is defined by colormap and caxis.

Shading Set color shading properties

- `shading flat`
- `shading faceted`
- `shading interp`

5.1 Color image

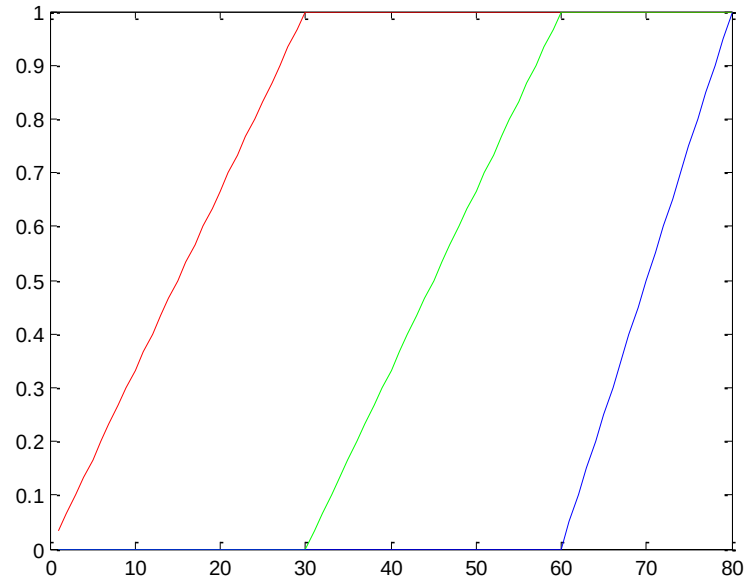
```
subplot(3,1,1),pcolor(peaks(50));shading flat
subplot(3,1,2),pcolor(peaks(50));shading faceted
subplot(3,1,3),pcolor(peaks(50));shading interp
```



5.1 Color image

`rgbplot` Plot colormap

`rgbplot(cmap)` plots the three columns of cmap, where `cmap` is an *m*-by-3 colormap matrix. rgbplot draws the first column in red, the second in green, and the third in blue.



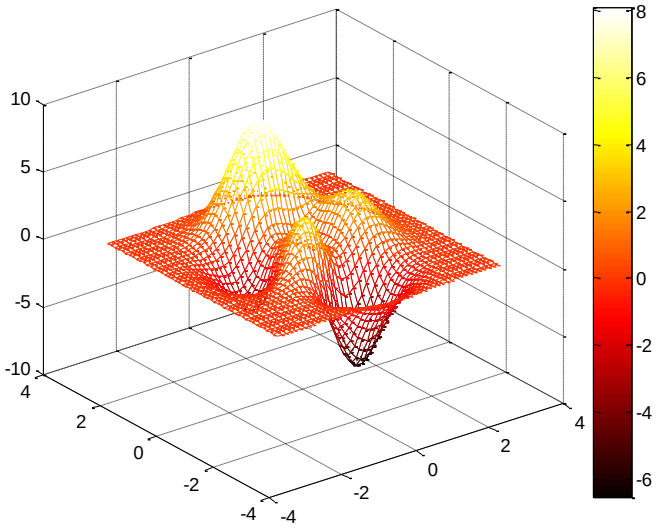

```
rgbplot(hot(80))
```

5.1 Color image

`colorbar` Colorbar showing color scale

`colorbar` adds a new vertical colorbar on the right side of the current axes. If a colorbar exists in that location, colorbar replaces it with a new one. If a colorbar exists at a nondefault location, it is retained along with the new colorbar.

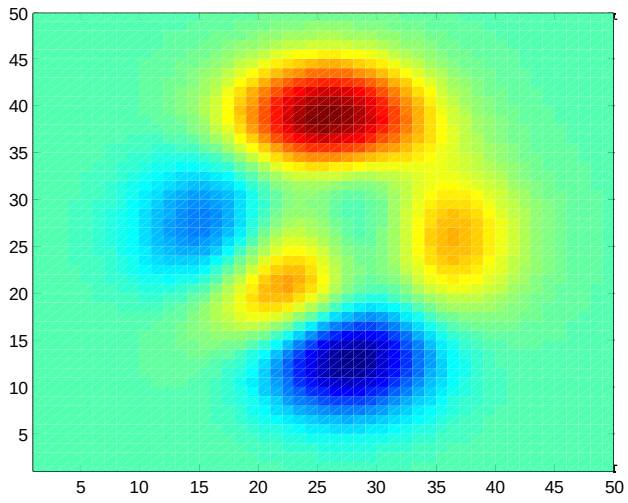
```
[X,Y,Z]=peaks;
mesh(X,Y,Z);
colormap(hot(80));
colorbar
```

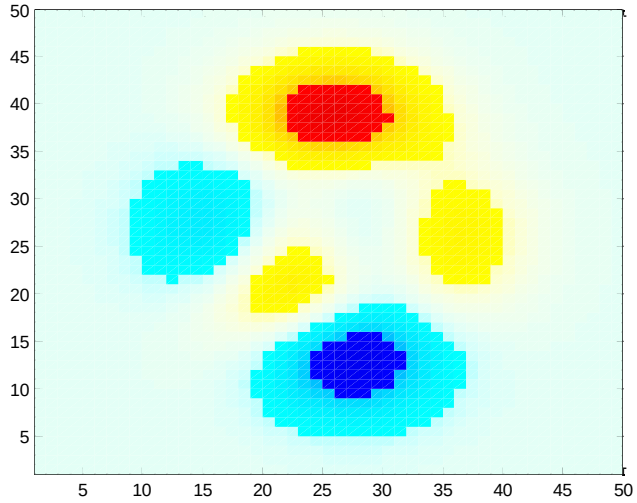


5.1 Color image

`brighten` Brighten or darken colormap

`brighten(beta)` replaces the current colormap with a brighter or darker colormap of essentially the same colors. `brighten(beta)`, followed by `brighten(-beta)`, where `beta < 1`, restores the original map.






```
pcolor(peaks(50));shading interp
figure,pcolor(peaks(50));shading interp,brighten(0.9)
```

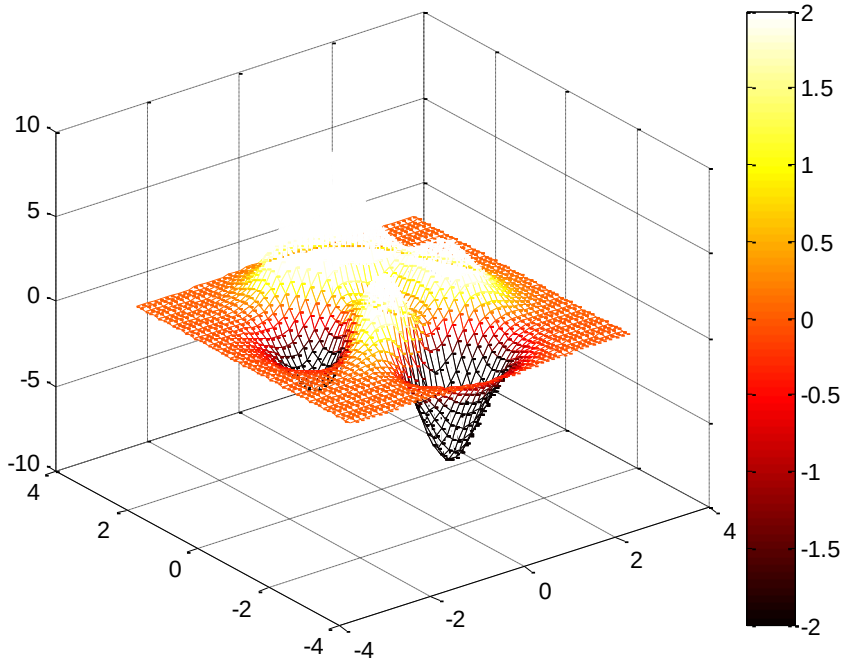
5.1 Color image

`caxis` Color axis scaling

`caxis([cmin cmax])` sets the color limits to specified minimum and maximum values. Data values less than `cmin` or greater than `cmax` map to `cmin` and `cmax`, respectively. Values between `cmin` and `cmax` linearly map to the current colormap.

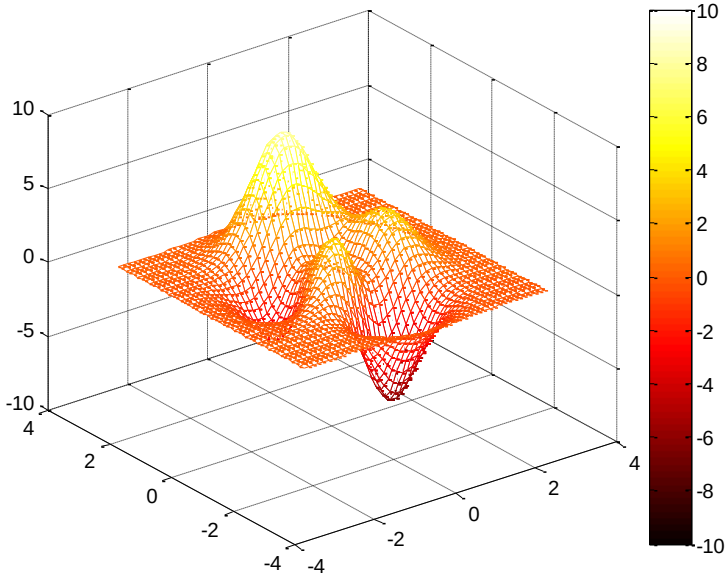
5.1 Color image

`caxis` Color axis scaling



`caxis([-2,2])`

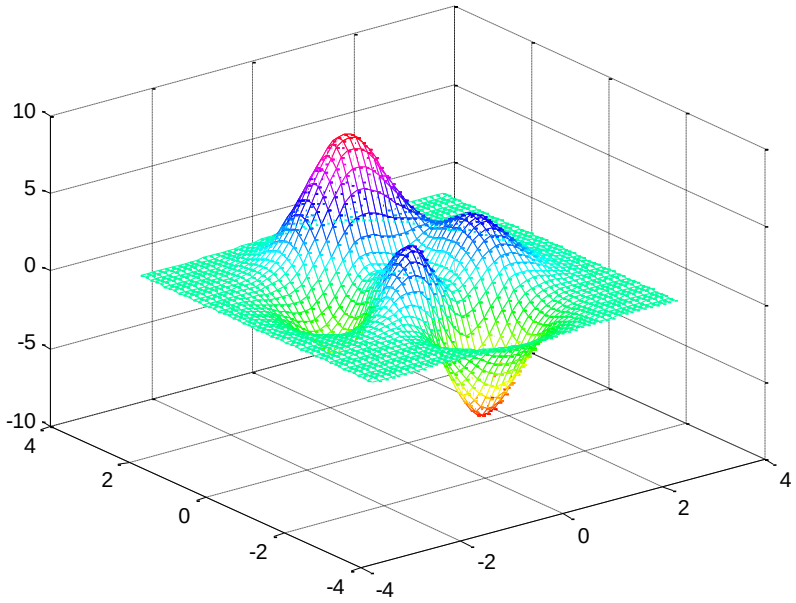
`caxis([-10,10])`



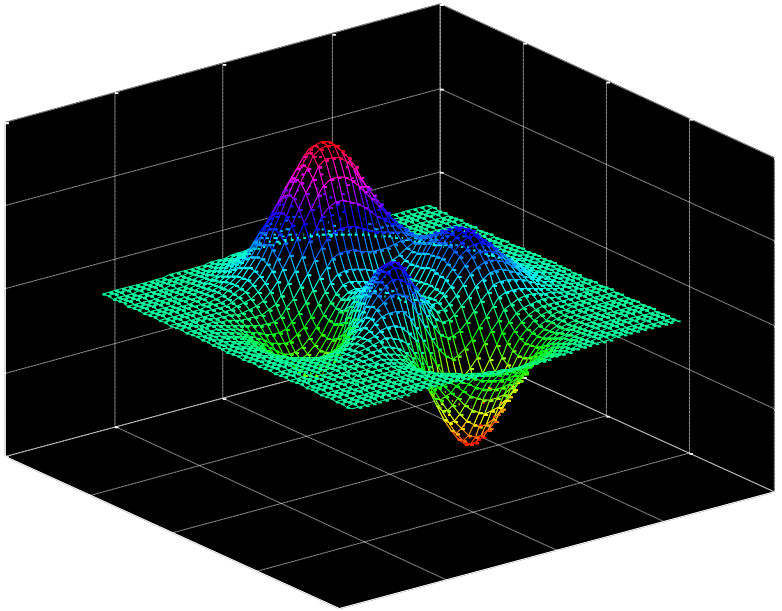
5.1 Color image

`colordef`

Set default property values to display different color schemes.



`colordef white`



`colordef black`

5.2 Perspective and light

`view` Viewpoint specification

`view(az,el)` and `view([az,el])` set the viewing angle for a three-dimensional plot. The azimuth, `az`, is the horizontal rotation about the *z*-axis as measured in degrees from the negative *y*-axis. Positive values indicate counterclockwise rotation of the viewpoint. `el` is the vertical elevation of the viewpoint in degrees. Positive values of elevation correspond to moving above the object; negative values correspond to moving below the object.

5.2 Perspective and light

`view` Viewpoint specification

`view(2)`

sets the default two-dimensional view, az = 0, el = 90.

`view(3)`

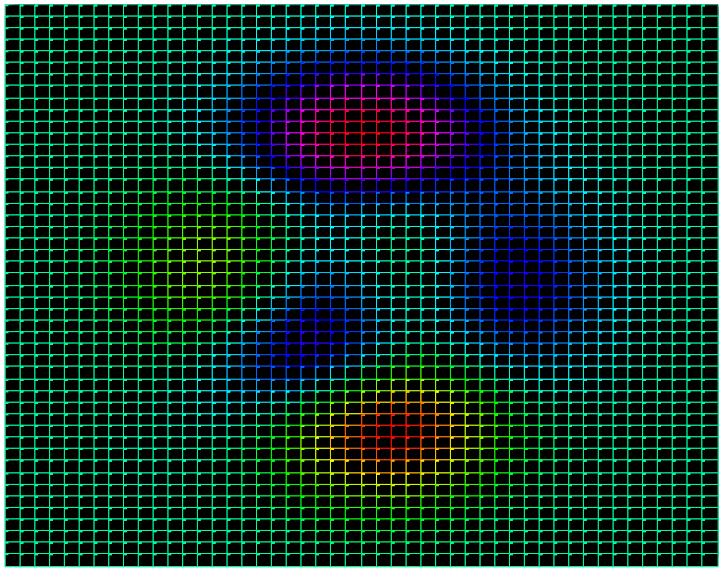
sets the default three-dimensional view, az = -37.5, el = 30.

`[az,el] = view`

returns the current azimuth and elevation.

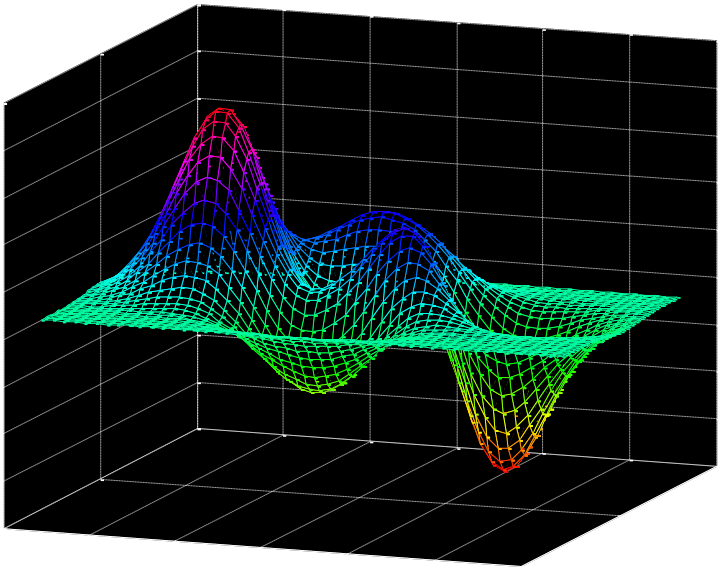
5.2 Perspective and light

`view` Viewpoint specification



```
[X,Y,Z]=peaks;
colormap(hsv(100))
mesh(X,Y,Z)
view(0,90)
```

```
[X,Y,Z]=peaks;
mesh(X,Y,Z);
rotate3d
```



5.2 Perspective and light

lamp-house operating function

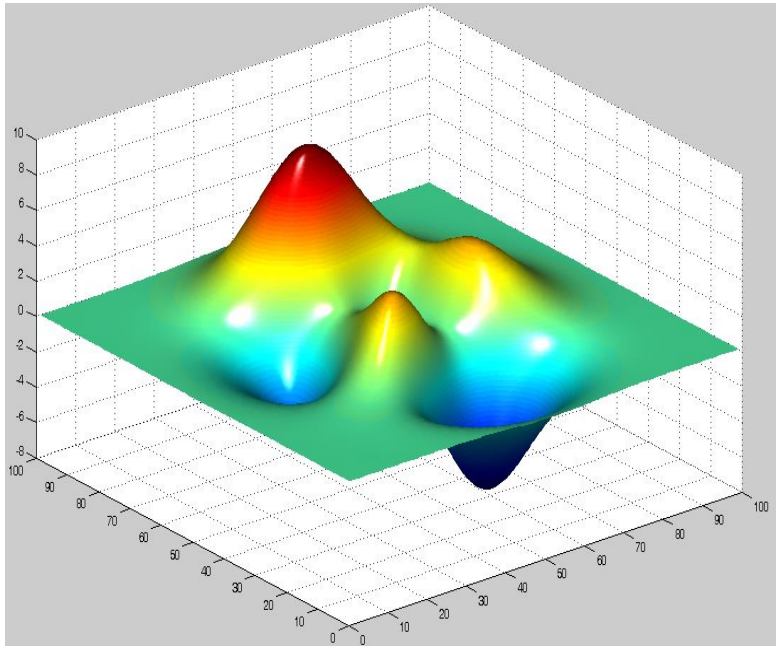
Light	Create light object
`Surfl`	Surface plot with colormap-based lighting
`Lighting`	Specify lighting algorithm
`Material`	Control reflectance properties of surfaces and patches
`Specular`	Calculate specular reflectance
`Diffuse`	Calculate diffuse reflectance
`lightangle`	Create or position light object in spherical coordinates

`light` create light object

`light('`*`PropertyName`*`',propertyvalue,...)`

5.2 Perspective and light

```
surf(peaks(100))
shading interp
lightangle(-45,30)
set(gcf,'Renderer','zbuffer') ,set(findobj(gca,...
'type','surface'),'FaceLighting','phong',...
'AmbientStrength',.3,'DiffuseStrength',.8, ...
'SpecularStrength',.9,'SpecularExponent',25, ...
'BackFaceLighting','unlit')
```



5.3 Image Processing

`imread` Read image from graphics file

`A = imread(filename, fmt)` reads a grayscale or color image from the file specified by the string `filename`. If the file is not in the current directory, or in a directory on the MATLAB path, specify the full pathname.

`imwrite` Write image to graphics file

`imwrite(A,filename,fmt)` writes the image `A` to the file specified by `filename` in the format specified by *`fmt`*.

5.3 Image Processing

Format	Full Name
`Bmp`	Windows bitmap
`gif`	Graphics interchange format
`Hdf`	Hierarchical data format
`Jpg or jpeg`	Joint photographic experts group
`pbm`	Portable bitmap
`pcx`	Windows paintbrush
`pgm`	Portable graymap
`png`	Portable network graphics
`pnm`	Portable anymap
`ppm`	Portable pixmap
`ras`	Sun raster

5.3 Image Processing

`imfinfo` Information about graphics file

`info = imfinfo(filename,`*`fmt`*`)` returns a structure whose fields contain information about an image in a graphics file. `filename` is a string that specifies the name of the graphics file, and *`fmt`* is a string that specifies the format of the file. The file must be in the current directory or in a directory on the MATLAB path. If imfinfo cannot find a file named filename, it looks for a file named filename.*fmt*. The possible values for *fmt* are contained in the MATLAB file format registry.

5.3 Image Processing

`Imfinfo` Information about graphics file

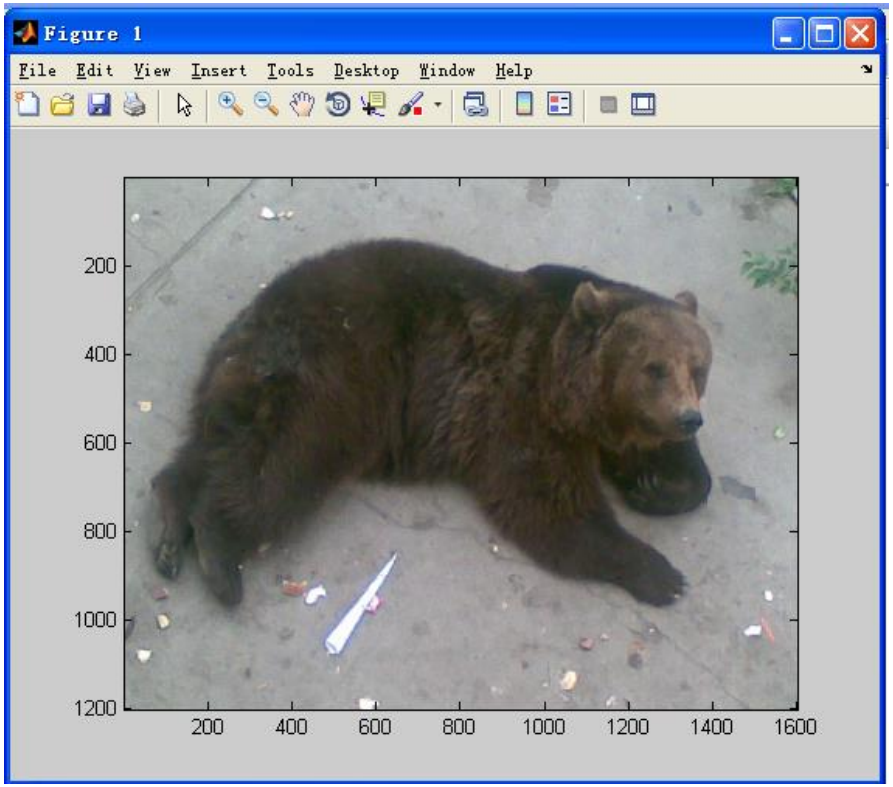
`info = imfinfo(filename,fmt)`

```
>> pictureinfo = imfinfo('picture1','jpg')

pictureinfo =

           Filename: 'picture1.jpg'
        FileModDate: '02-五月-2008 17:20:42'
           FileSize: 182825
             Format: 'jpg'
      FormatVersion: ''
              Width: 1600
             Height: 1200
           BitDepth: 24
          ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
            Comment: {}
               Make: 'Nokia '
              Model: 'E51 '
        Orientation: 1
        XResolution: 300
        YResolution: 300
     ResolutionUnit: 'Inch'
  YCbCrPositioning: 'Centered'
      DigitalCamera: [1x1 struct]
```

5.3 Image Processing

`image` Display image object

`image(C)` displays matrix `C` as an image. Each element of `C` specifies the color of a rectangular segment in the image.

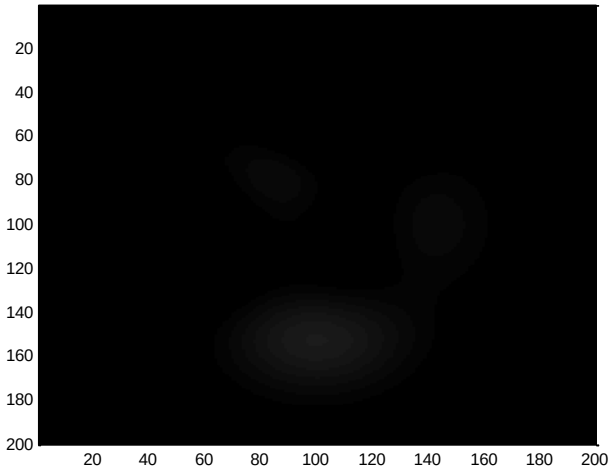


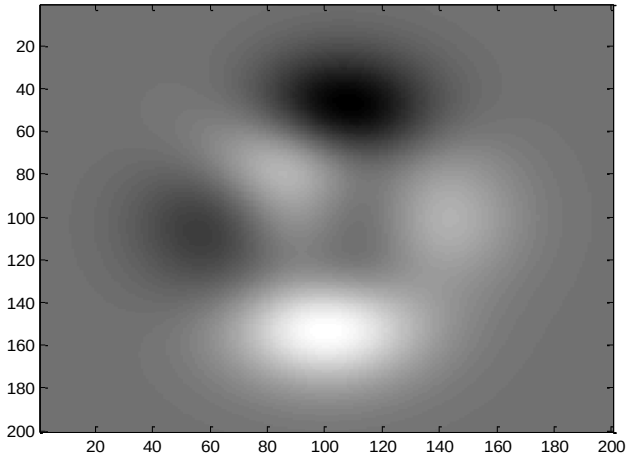

```
A=imread('picture1','jpg');

size(A);

image(A)
```

5.3 Image Processing

Imshow Image Imagesc

```
a=peaks(200);
imshow(a),colormap(gray)
figure,image(a),colormap(gray)
figure,imagesc(a),colormap(gray)
```

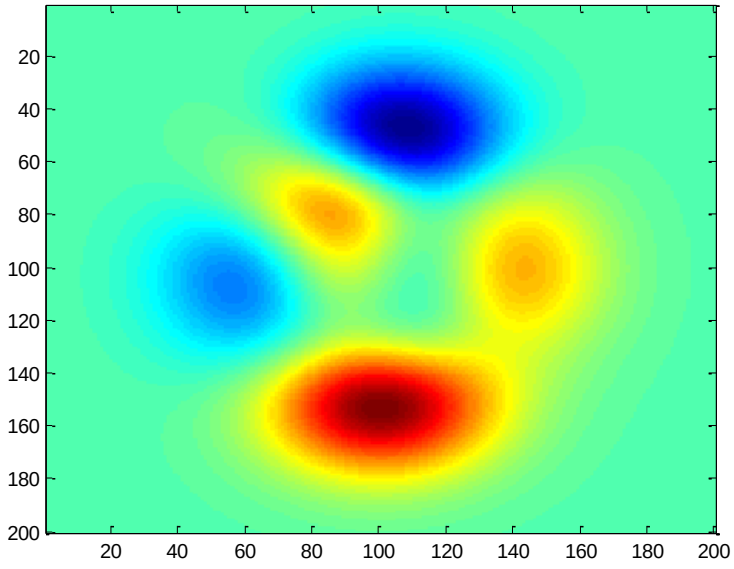




5.4 Graphic output

`print` Print figure or save to file and configure printer defaults

`print file name` directs the output to the PostScript file designated by file name. If file name does not include an extension, print appends an appropriate extension.




```
figure,imagesc(peaks(200))
print( gcf, '-djpeg', 'e:\a.jpg' )
```

5.5 get and set function

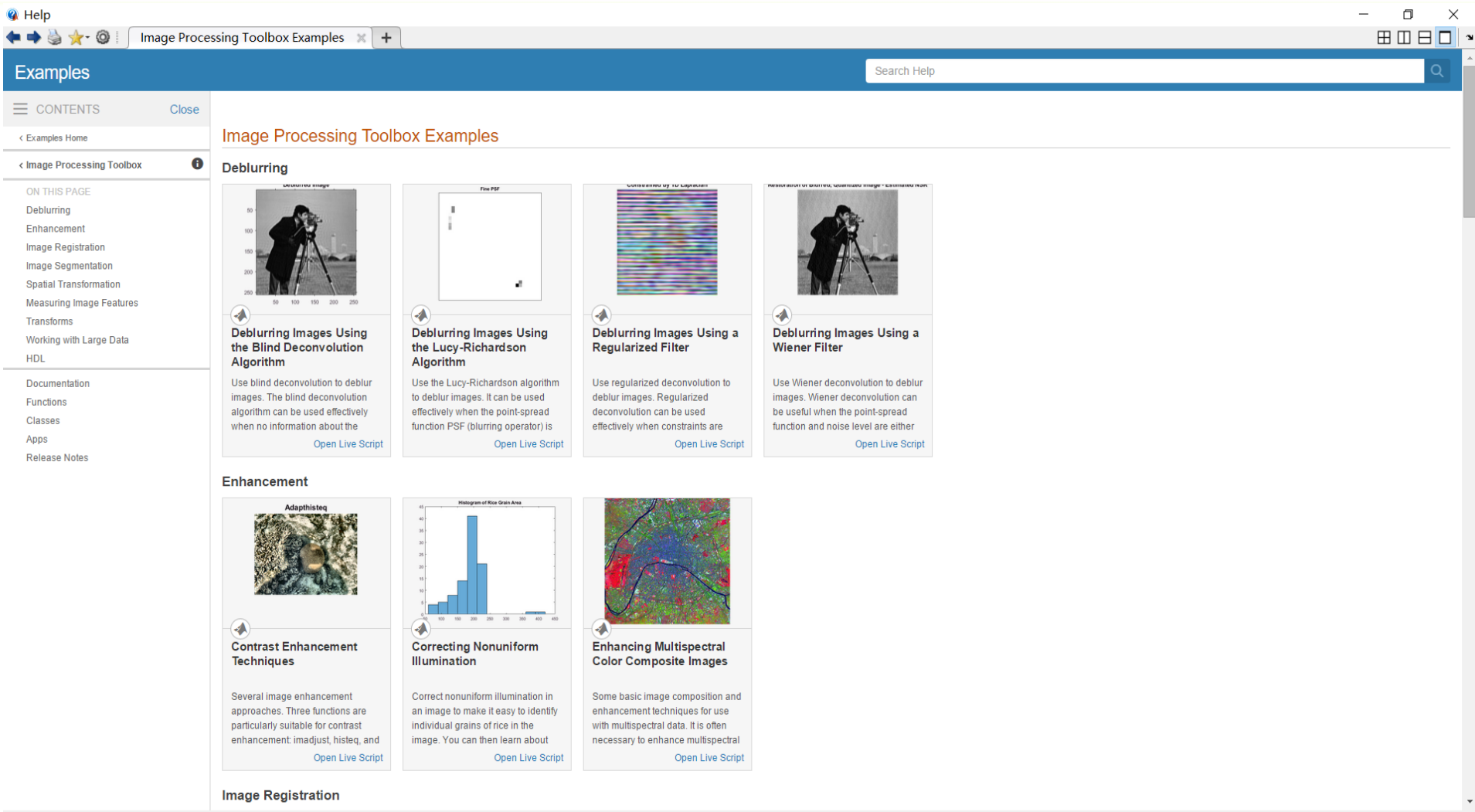
get

- `get(gcf)`
- `get(gcf,'color')`
- `get(gca)`
- `get(gca,'LineWidth')`

set

- `set(gcf,'color',[0.5 0.5 0.5])`
- `set(gca,' LineWidth',3)`

5.6 Image Processing Toolbox



See example CellSegmentationExample.mlx

Homework 5

HW5-1.Create a 3x2 subplot and do the following

1. Plot the function $e^{-0.2x}(\sin(x) - j\cos(x))$, x ranges from 0 to 4pi in increments of pi/30.[Diplay real part with cyan solid line and imaginary part with magenta dashed line]
2. Plot discrete sequence data for the above function [Diplay real part with blue color and imaginary part with red color]
3. Plot bar graph for the above function [real part only with blue color]
4. Polar plot showing magnitude versus angle. [Use Compass Plot]
5. Plot both functions $e^{-x} * \sin(x)$ [with blue line, x ranges from -3 to 0] & $-e^{x} * \sin(x)$ [with red line, x ranges from 0 to 3] using fplot
6. Suppose that Wang, Zhang, Liu, Chen, and Zhao contributed ¥5, ¥15, ¥5, ¥10, and ¥15, respectively, to buy a colleague's Wedding present. Create a pie chart of their contributions. What percentage of the cost was paid by Zhang?

HW5-2. (a)Write a MATLAB program to draw a 3-D heart using isosurface for the following function:

$$f = (2x^2 + 2y^2 + z^2 - 1)^3 - (1/10)x^2z^3 - y^2z^3$$

Let x range from -1.8 to 1.8 in increments of 0.02.

Let y range from -1.2 to 1.2 in increments of 0.01.

Let z range from -1.8 to 1.8 in increments of 0.02

(b)Make the heart 'jumping' in your ways.

Submit homework via online before Nov 5，2019

Thanks