Introduction to Applied Scientific Computing using MATLAB

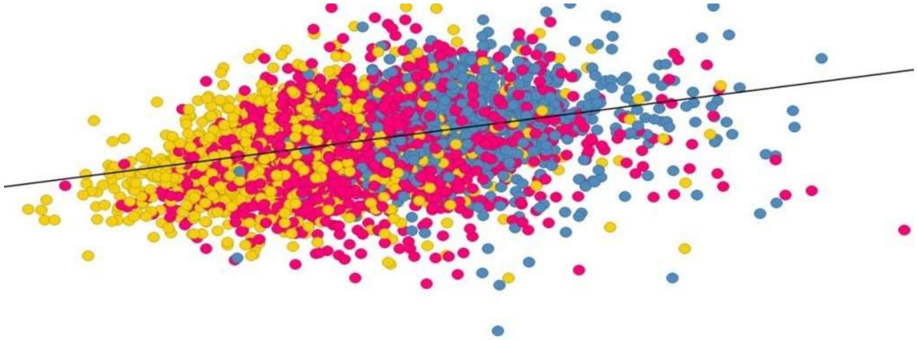
Mohsen Jenadeleh

In this lecture, slides from MIT, Rutgers and Waterloo University are used to form the lecture slides

Connect Activity

Question:

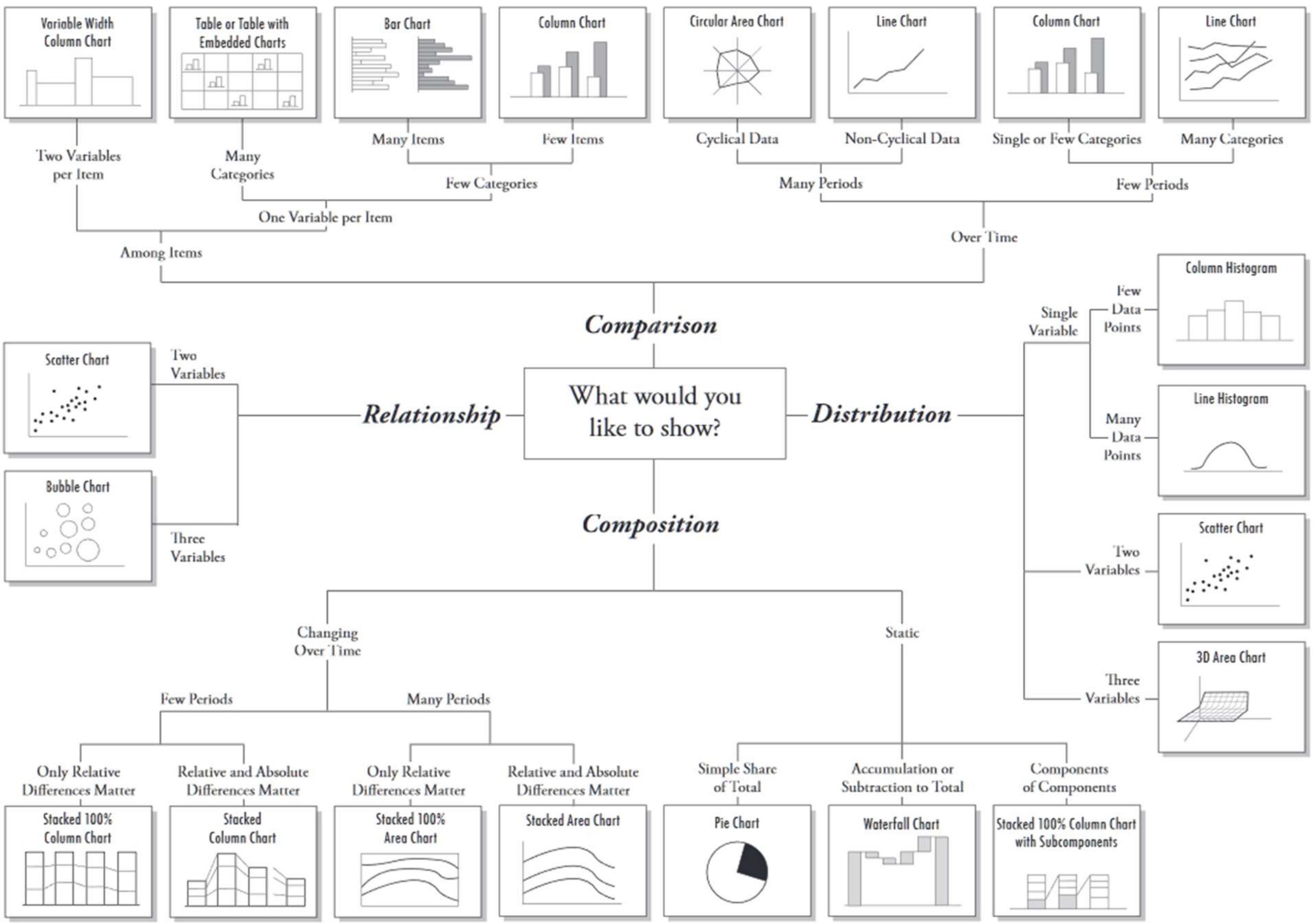
What is the purpose of a data visualization?



Information Visualization?

- The study of visual representations of data to reinforce human cognition.
- “Help people understand the, structure, relationships meaning in data.”
- Techniques: Charts, Graphs, Maps

Chart Suggestions—A Thought-Starter



www.ExtremePresentation.com

Review from Weeks 1 & 2

MATLAB has extensive facilities for the plotting of curves and surfaces, and visualization.

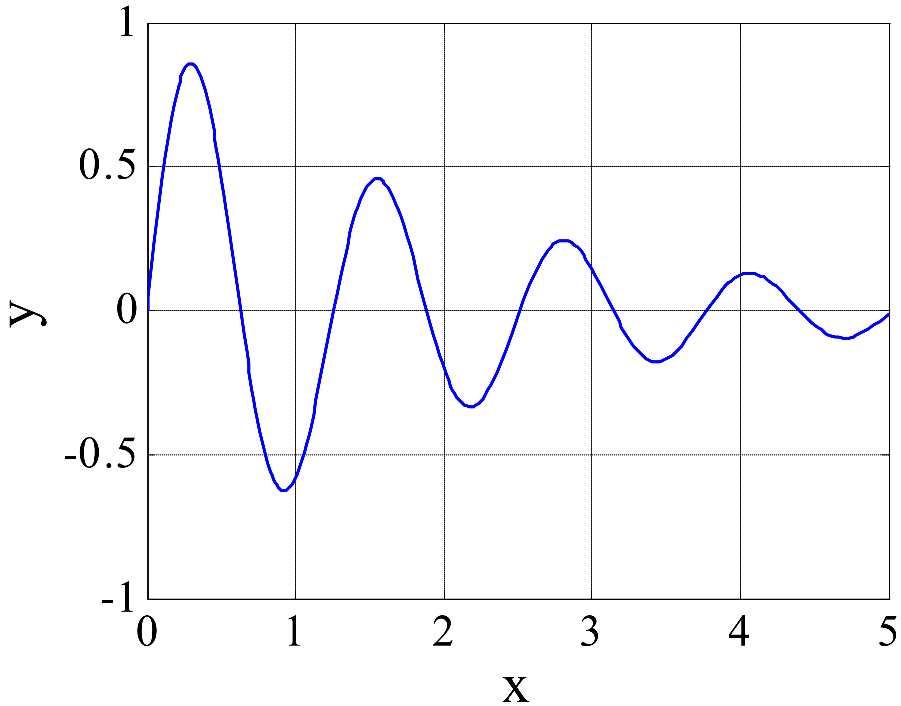
Basic 2D plots of functions and (x,y) pairs can be done with the functions:

plot, fplot, ezplot

```
>> help plot       % 2-D plotting
>> help fplot      % function plotting
>> help ezplot     % easy function plotting
```

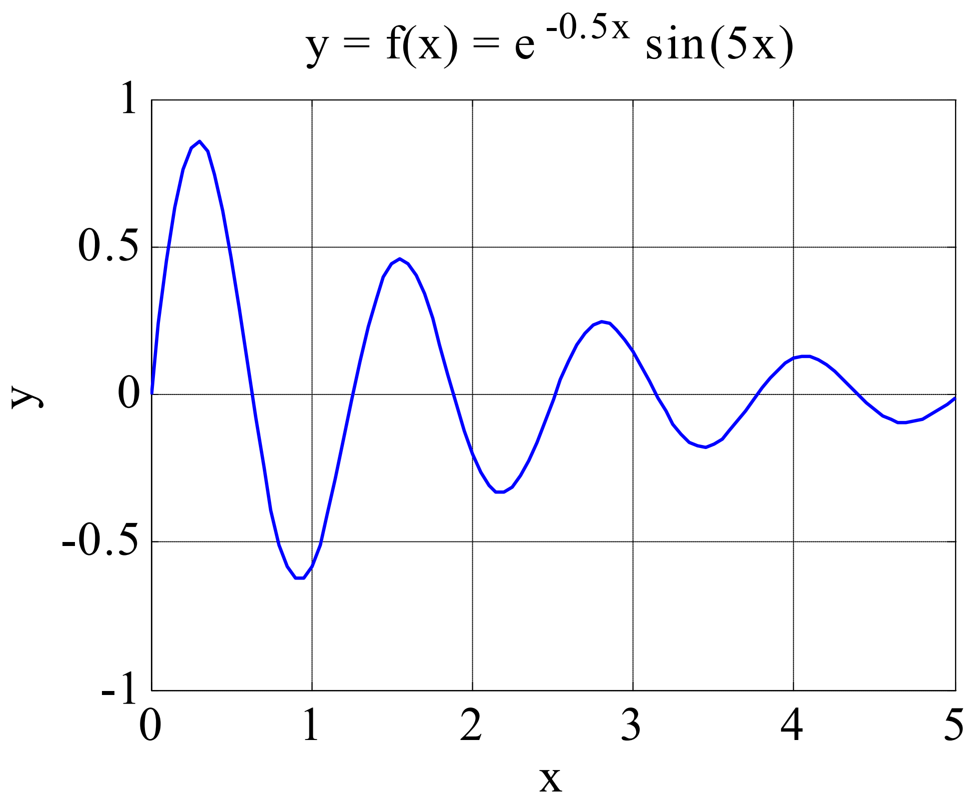
If a function f(x) has already been defined by a function-handle or inline, it can be plotted quickly with **fplot**, **ezplot**, which are very similar. One only needs to specify the plot range. For example:

```
>> f = @(x) exp(-0.5*x).*sin(5*x);
>> fplot(f,[0,5]);          % plot over interval [0,5]
```



A figure window opens up, allowing further editing of the graph, e.g., adding x,y axis labels, titles, grid, changing colors, and saving the graph is some format, such as WMF, PNG, or EPS, EPSC using the plot function

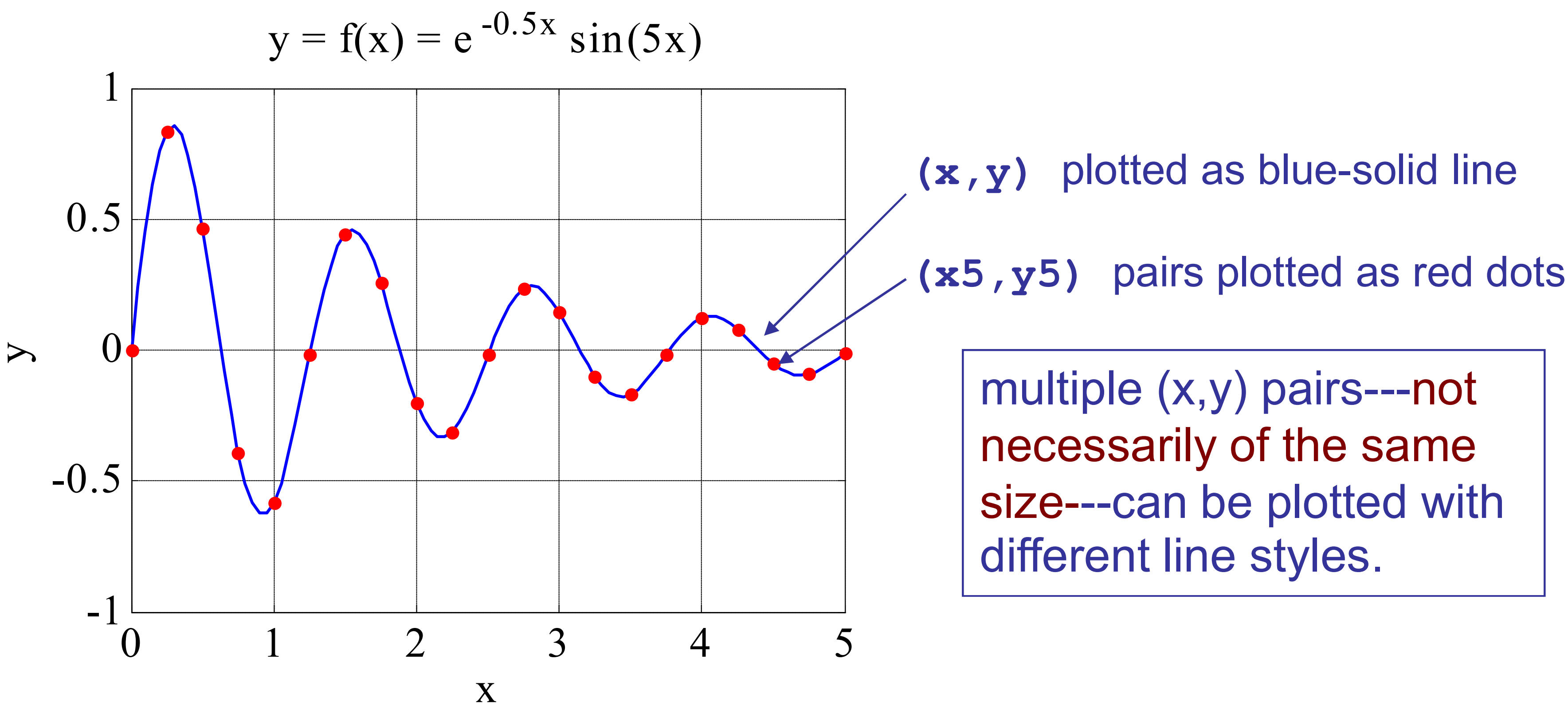
```
>> x = linspace(0,5,101);
>> y = f(x);
>> plot(x,y,'b-');                        % blue-solid line
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x} sin(5x)');
```



plot annotation can be done by separate commands, as shown above, or from the plot editor in the figure window.

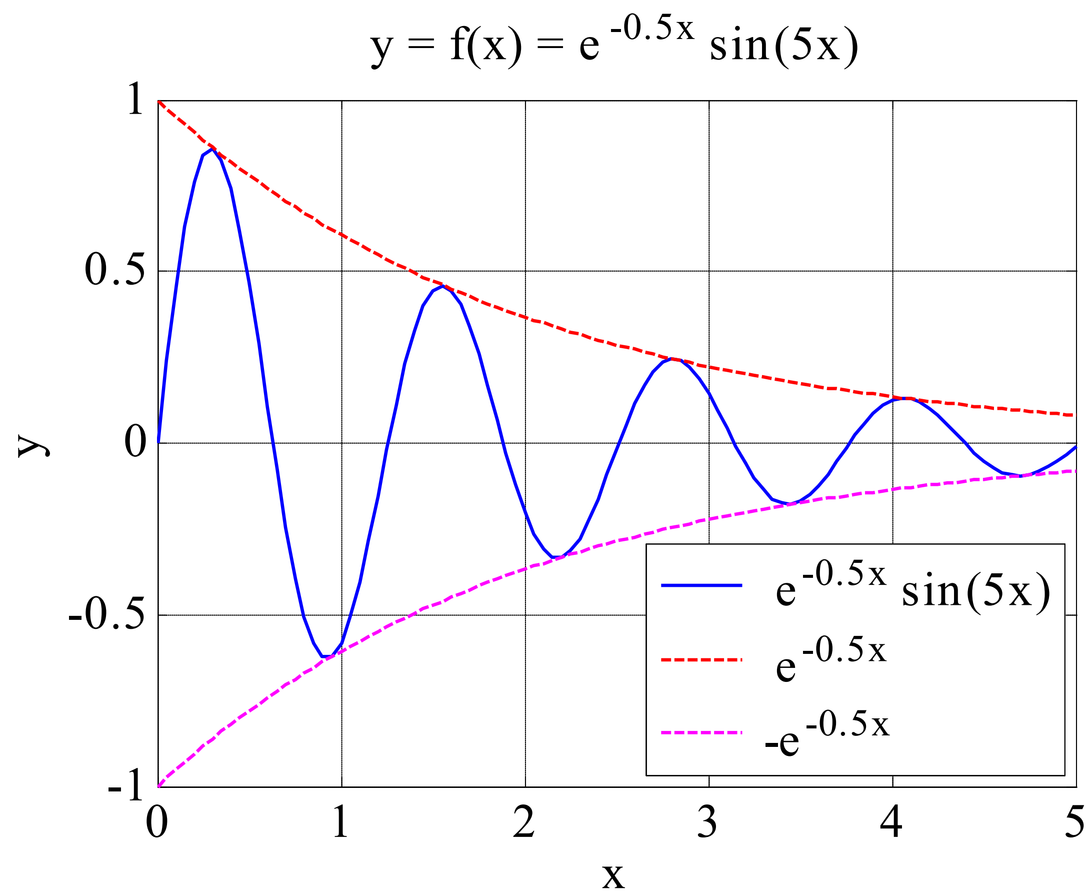
multiple graphs on same plot

```
>> x5 = x(1:5:end);            % plot every 5th data point
>> y5 = y(1:5:end);
>> plot(x,y,'b-', x5,y5, 'r.');    % blue-line, red dots
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x} sin(5x)');
```



multiple (x,y) pairs---not necessarily of the same size---can be plotted with different line styles.

```
>> ye = exp(-0.5*x);                  % envelope of f(x)
>> plot(x,y,'b-', x,ye,'r--', x,-ye,'m--');
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x} sin(5x)');
>> legend('e^{-0.5x} sin(5x)', 'e^{-0.5x}', ...
   '-e^{-0.5x}', 'location','SE');
```



south-east

ellipsis continues to next line

plotting multiple curves and adding legends

legends can also be inserted with plot editor

Multiple plots : use "hold on" after each plot

plot

```
plot(x,y, 'specifiers', 'property', prop_value);
```

line style,
line color,
marker

line width,
marker size,
marker color
color, marker

Example:

```
plot(x,y,'b-','linewidth',2,'markersize',12,...
      'markeredgecolor','r',...
      'markerfacecolor','g');
```

<u>default values:</u>
LineWidth = 0.5 points
MarkerSize = 6
FontSize = 10

Line Styles, Point Types, Colors, and Properties

Style		Type		Color	
solid	`-`	point	`.`	blue	`b`
dotted	`:`	circle	`o`	green	`g`
dash-dot	`-.`	x-mark	`x`	red	`r`
dashed	`--`	plus	`+`	cyan	`c`
		star	`*`	magenta	`m`
		square	`s`	yellow	`y`
		diamond	`d`	black	`k`
		triang dn	`v`		
		triangle up	`^`		
		triang left	`<`		
		triang right	`>`		
		pentagram	`p`		
		hexagram	`h`		

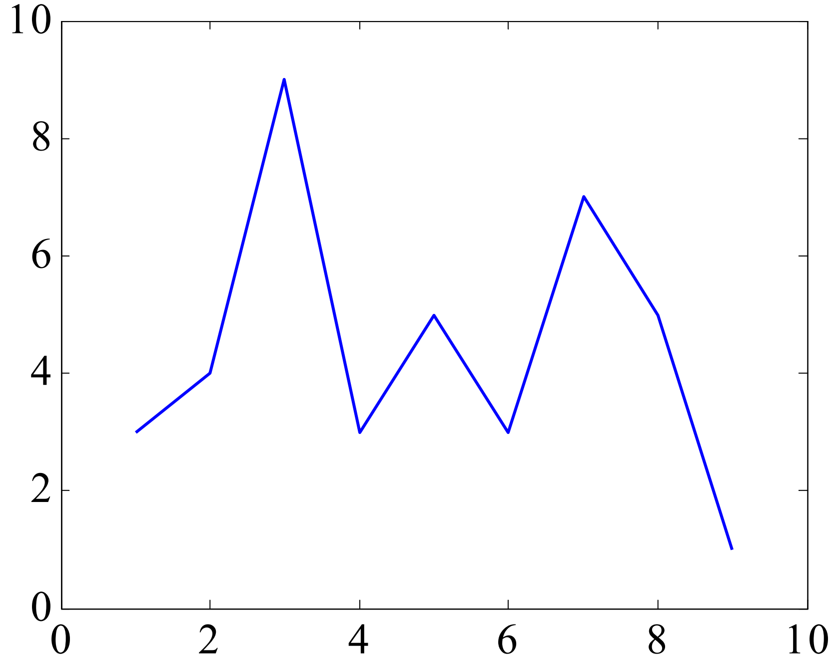
property name

linewidth
markersize
markeredgecolor
markerfacecolor

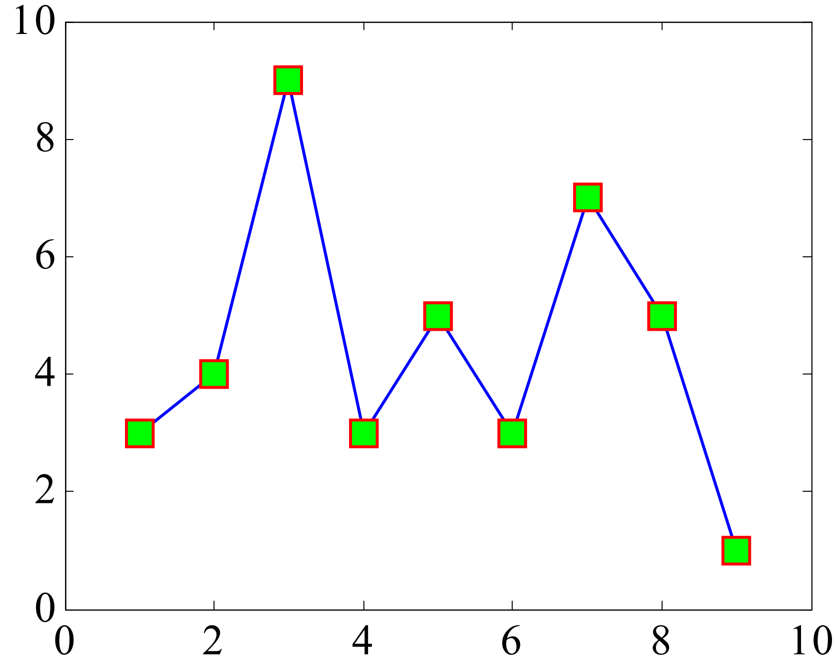
```
x = [1  2  3  4  5  6  7  8  9];
y = [3  4  9  3  5  3  7  5  1];
```

line styles & markers

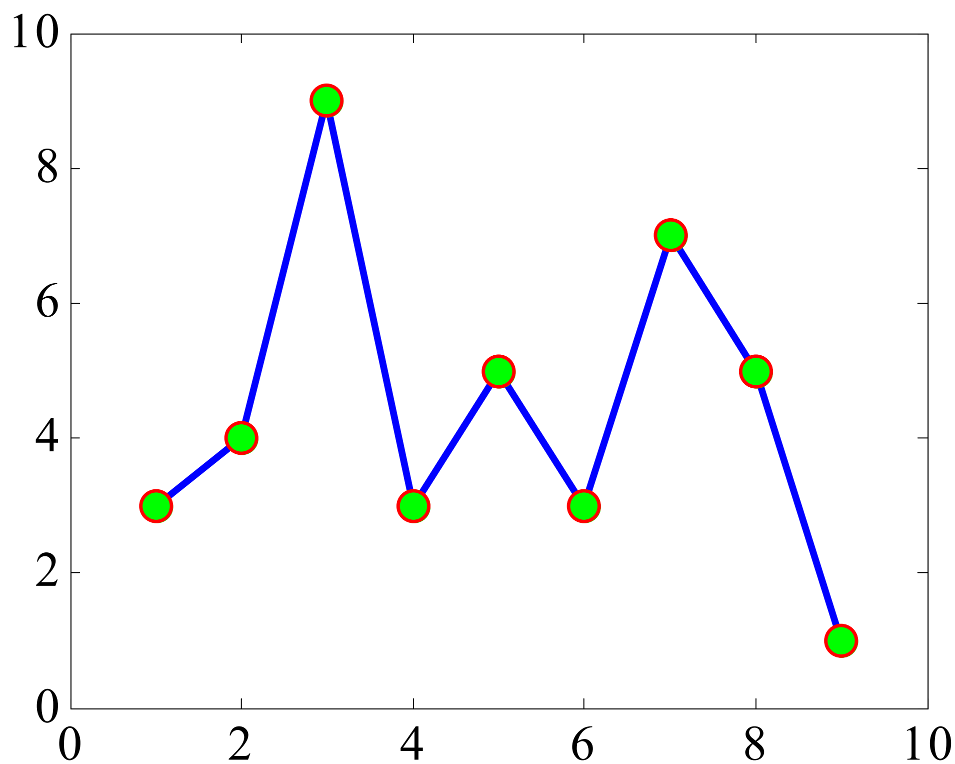
```
plot(x,y,'b-');
```




```
 plot(x,y,'bs-', ...
'MarkerEdgeColor','r',...
'MarkerFaceColor','g')
```



```
plot(x,y,'b-', 'LineWidth',3);
hold on;
plot(x,y,'or', 'MarkerSize', 12, ...
                  'MarkerFaceColor','g');
```



default values

LineWidth = 0.5 points

MarkerSize = 6

FontSize = 10

plot summary

insert additional option strings

```
plot(x1,y1,'opt1', x2,y2,'opt2', x3,y3,'opt3');
```

x1,y1 may have different size than **x2,y2**, or **x3,y3**

```
plot(x1,y1,'specs1','prop1',val1);
hold on;
plot(x2,y2,'specs2','prop2',val3);
plot(x3,y3,'specs3','prop3',val3);
hold off;
```

hold on/off allows independent specification of plot parameters

plot variants

```
% x = M-vector, Y = MxN matrix
plot(x,Y);
```

plot each column of **Y** against **x**

```
% X = MxN matrix, Y = MxN matrix
plot(X,Y);
```

plot each column of **Y** against each column of **X**

```
% Y = MxN real-valued matrix
plot(Y);
```

plot **Y** columns against their index

for complex **X,Y** only their real parts are used, and imag parts ignored,

exception

```
% Z = MxN complex-valued matrix
plot(Z);
plot(real(Z),imag(Z));
```

equivalent

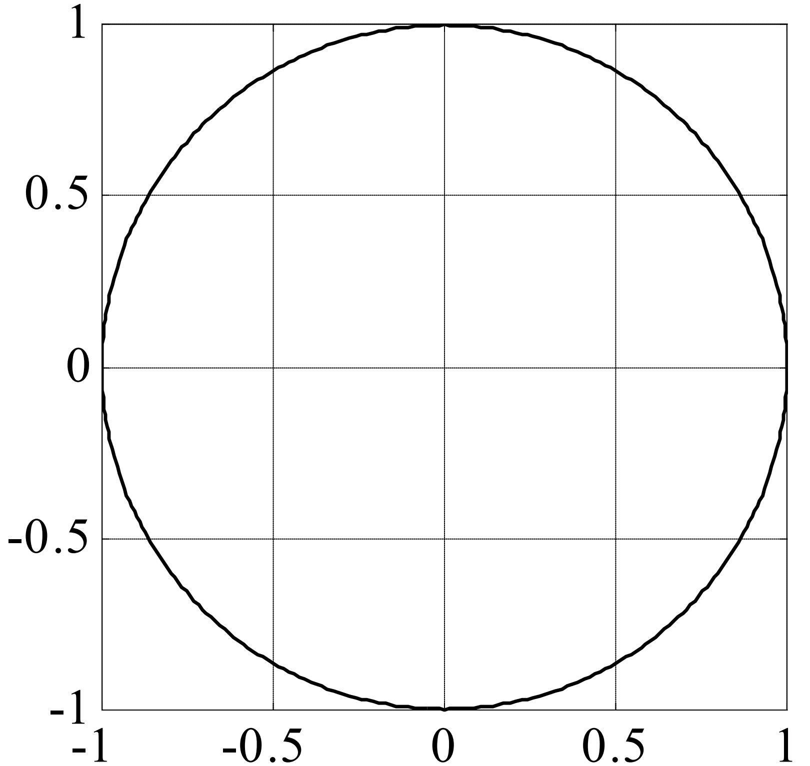
How to plot a circle

```
theta = linspace(0,2*pi,361);
z = exp(j*theta);
figure; plot(z);
axis equal;
axis square;
grid;
```

Euler's formula

$$e^{j\theta} = \cos\theta + j\sin\theta$$

imaginary unit, *j* or *i*



adding text

```
gtext('text_string');   add by mouse
text(x,y,'text_string','property',value);
```

property

fontsize	size of text font
color	text color
fontangle	normal, italic
fontweight	normal, bold
backgroundcolor	rectangular area of text
edgecolor	edge of rectangular box
linewidth	rectangular box
rotation	text orientation
fontname	specify font

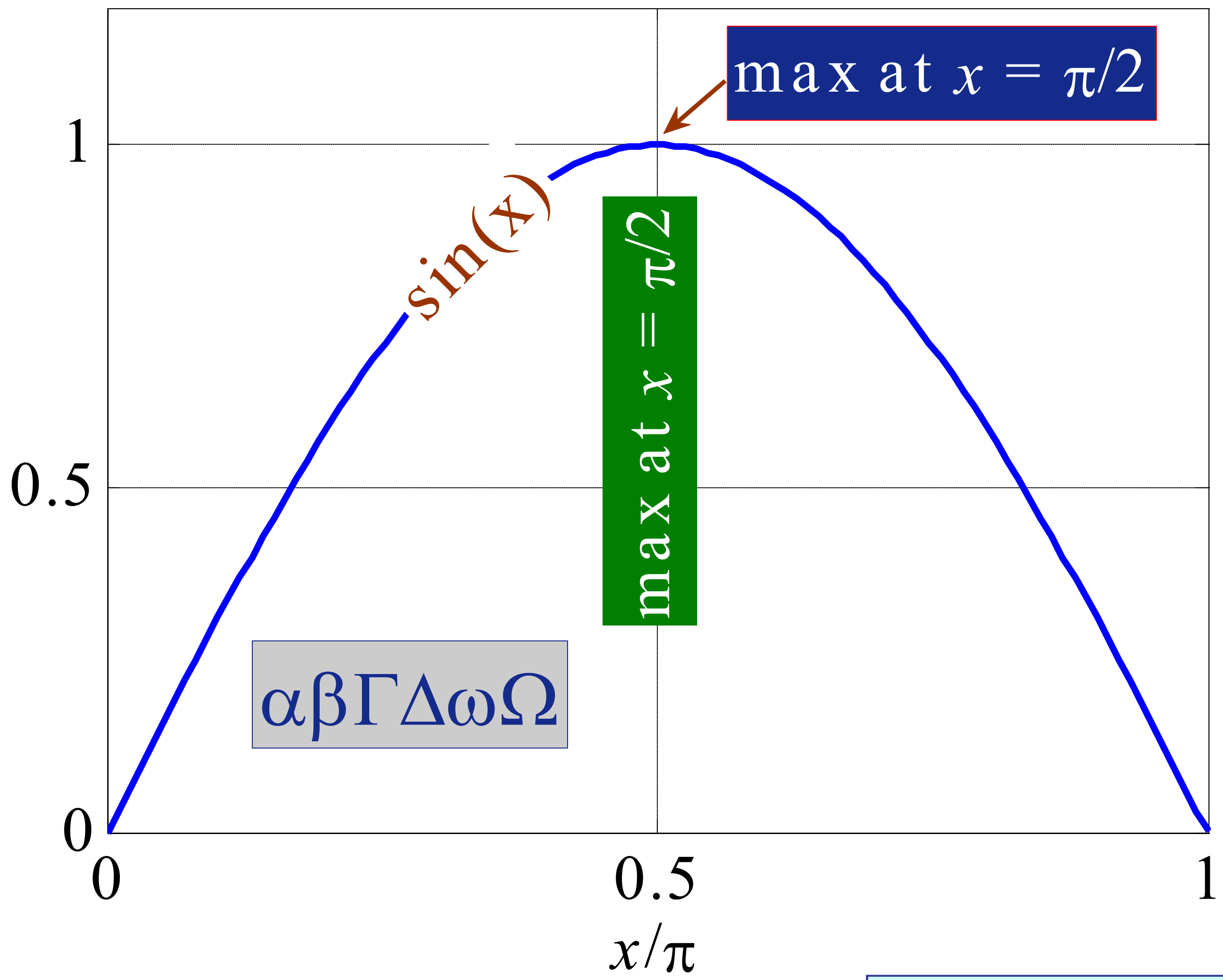
properties can also be set with the plot editor

can also be used in **title, xlabel, ylabel, legend**

adding text

```
x = linspace(0,pi,100); y = sin(x);

plot(x/pi,y,'b','linewidth',2);

xlabel('{\itx}/\pi'); grid on;

str = 'max at {\itx} = \pi/2';

gtext(str,'fontsize',20);
gtext(str,'fontsize',20,'rotation',90);
gtext('sin(x)','fontsize',20,'rotation',60);
gtext('\alpha\beta\Gamma\Delta\omega\Omega');
```

text positions, colors, sizes, and background colors can be fine-tuned from the plot editor (see net page)



adding text

find out the [x,y] coordinates of a point using

```
[x,y] = ginput;
```

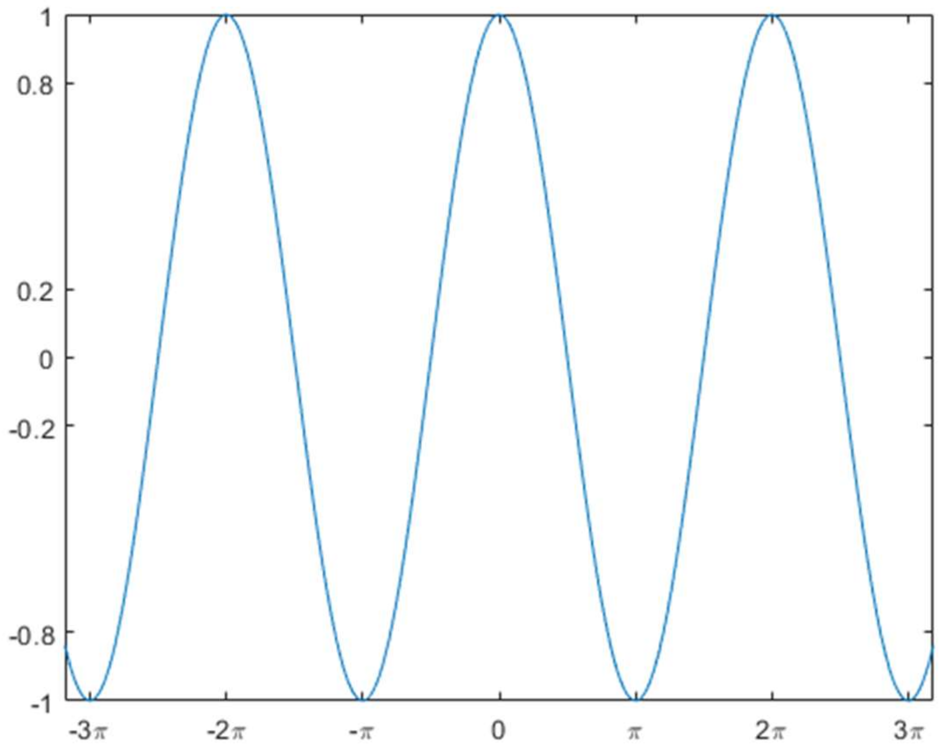
axis settings

```
axis auto;          % default settings
axis equal;         % equal x,y units
axis square;        % square box
axis off;           % remove axes
axis on;            % restore axes
axis tight;         % limits from data range
axis ij;            % matrix mode (i=vert, j=horiz)
axis xy;            % cartesian mode
axis normal;        % default axis

axis([xmin,xmax,ymin,ymax]);              % limits
axis([xmin,xmax,ymin,ymax,zmin,zmax]);

xlim([xmin,xmax]);          % set x-axis limits
ylim([ymin,ymax]);
zlim([zmin,zmax]);

set(gca, 'xtick', v);        % v = tickmark vector
set(gca, 'ytick', v);        % e.g., v = 0:2:10
```

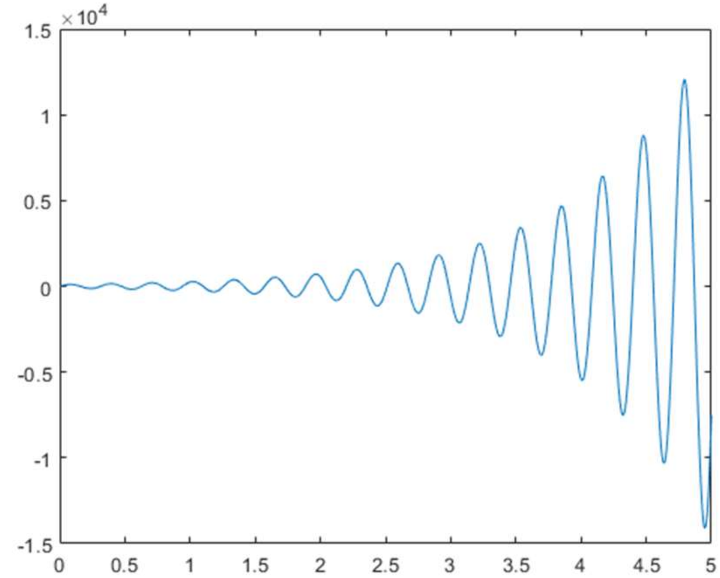
correlated

```
x = linspace(-10,10,200);
y = cos(x);
plot(x,y)
```

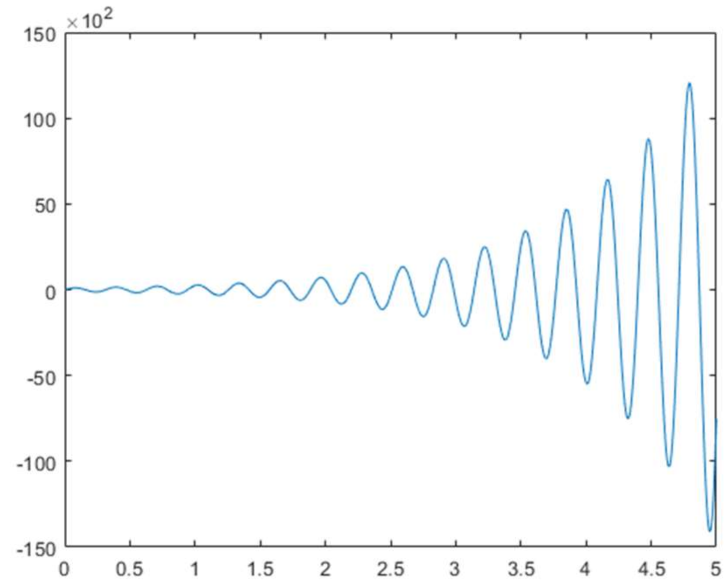
```
xticks([-3*pi -2*pi -pi 0 pi 2*pi 3*pi])
xticklabels({'-3\pi','-2\pi','-\pi','0','\pi','2\pi','3\pi'})
yticks([-1 -0.8 -0.2 0 0.2 0.8 1])
```



```
x = linspace(0,5,1000);
y = 100*exp(x).*sin(20*x);
plot(x,y)
```




```
ax = gca;
ax.YAxis.Exponent = 2;
```

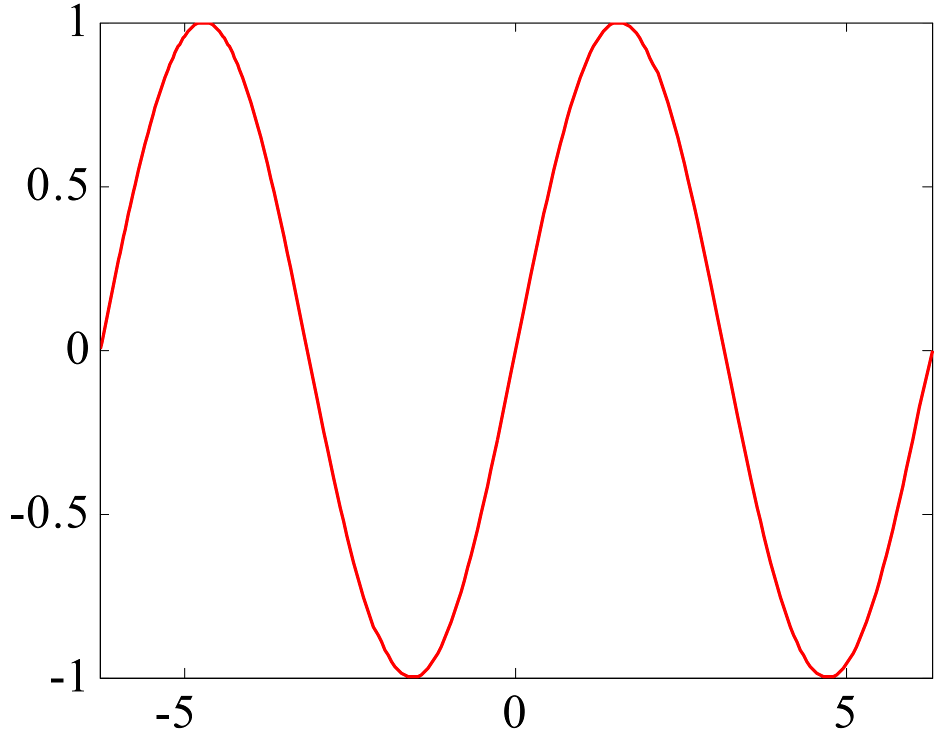


2D plotting functions

plot	basic x-y plot
fplot	function plot
ezplot	function plot
loglog	log x,y axes
semilogx	log x-axis
semilogy	log y-axis
plotyy	left & right y-axes
polar	polar plot
ezpolar	polar
comet	animated x-y plot
errorbar	plot with error bars
stem,stairs	stem and staircase
scatter	scatter plot
bar,barh	bar graphs
pie	pie chart
hist	histogram
fill,area	polygon & area fill

fplot, ezplot

```
fplot(@sin, [-2,2]*pi);
fplot('sin', [-2,2]*pi);
fplot('sin(x)', [-2,2]*pi);
f = @(x) sin(x);
fplot(f, [-2,2]*pi);
```

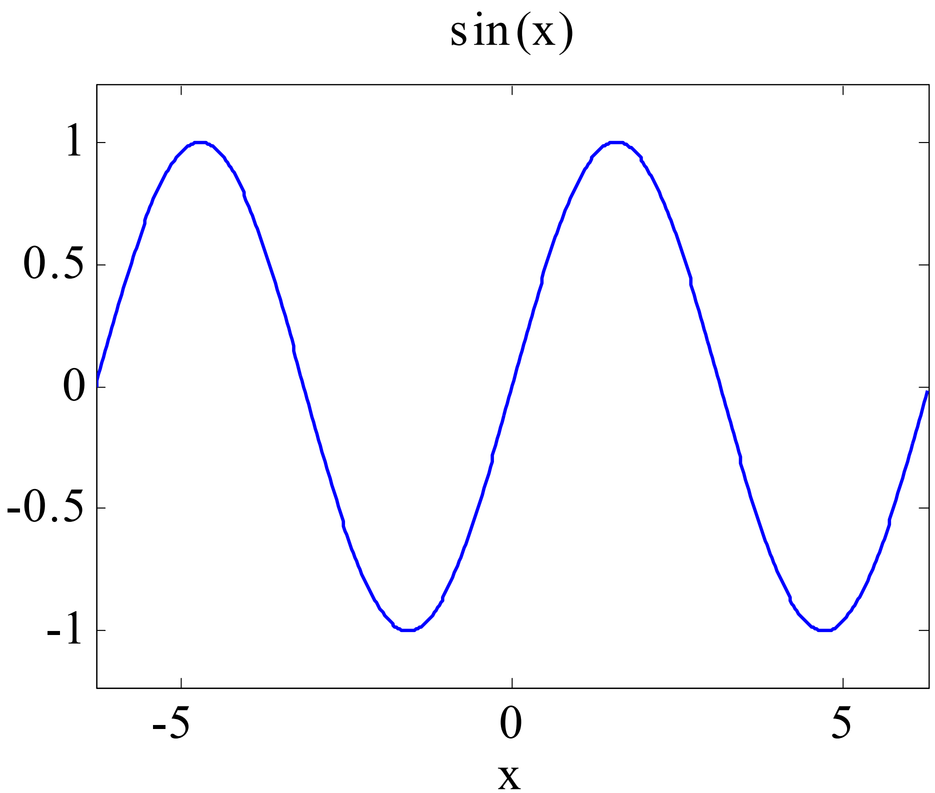


linestyles & colors can be changed from the figure window, or

```
fplot(f, [-2,2]*pi, 'r');
```

fplot, ezplot

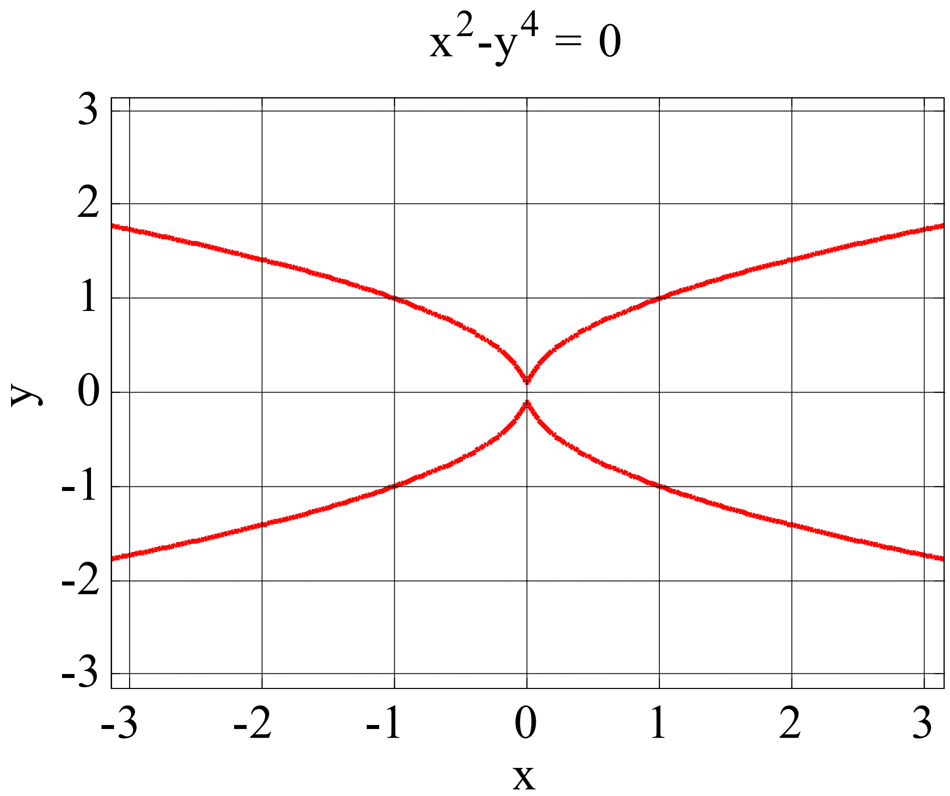
```
ezplot(@sin, [-2,2]*pi);
ezplot('sin', [-2,2]*pi);
ezplot('sin(x)', [-2,2]*pi);
f = @(x) sin(x);
ezplot(f, [-2,2]*pi);
```



linestyles & colors can be changed from the figure window

fplot, ezplot

```
ezplot('x^2-y^4', [-pi,pi]);

f = @(x,y) x.^2 - y.^4;
ezplot(f, [-pi,pi]);
```



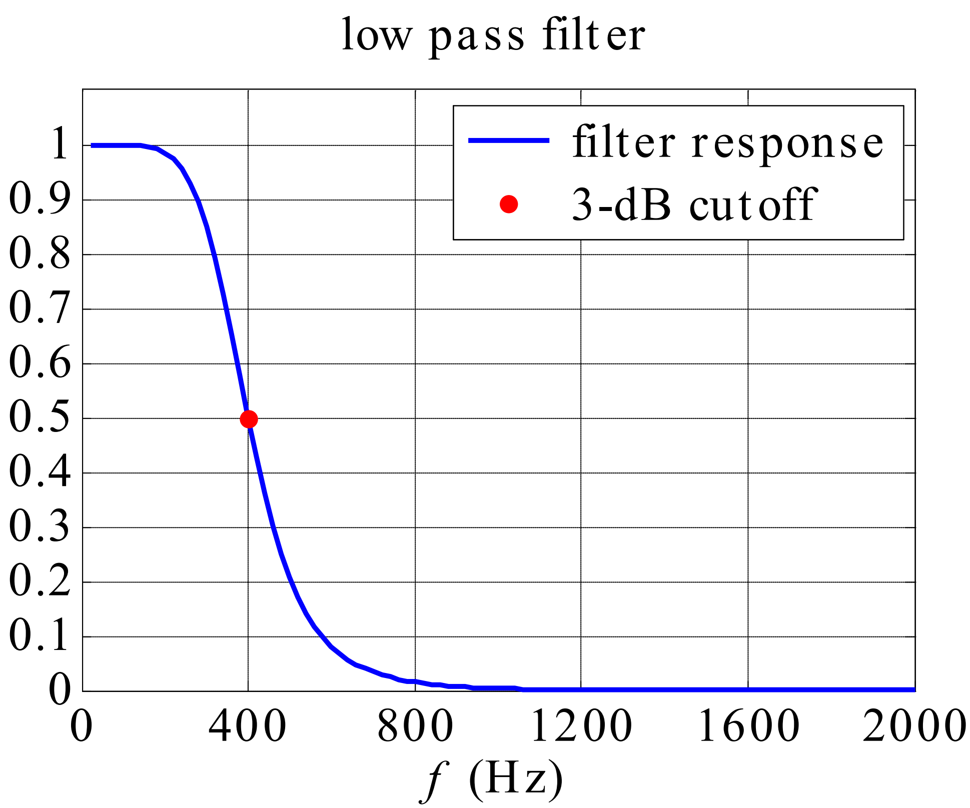
ezplot can plot functions defined implicitly, i.e., f(x,y) = 0

fplot plots function with single variable

loglog plots

Butterworth lowpass audio filter

$$|H(f)|^2 = \frac{1}{1 + \left(\frac{f}{f_0}\right)^{2N}}$$



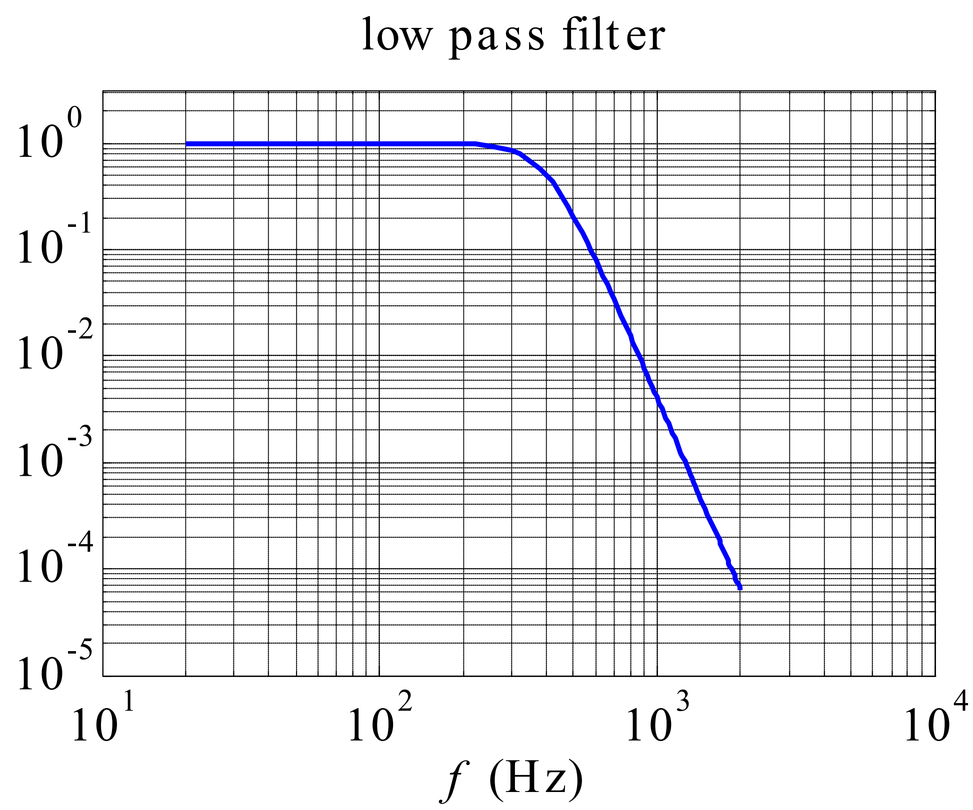
N = 3
f0 = 400 Hz

10*log10(0.5) = –3.01 dB

```
f = linspace(20,2000,100);    % 20 Hz to 2 kHz
f0 = 400;                     % 3-dB frequency

H2 = 1./(1+ (f/f0).^6);       % magnitude square

plot(f,H2,'b', 'linewidth',2);
hold on;
plot(f0,0.5,'r.', 'markersize',20);

xaxis(0,2000, 0:400:2000);
yaxis(0,1.1, 0:0.1:1); grid;
xlabel('{\itf}  (Hz)');
title('low pass filter');

legend(' filter response', ' 3-dB cutoff',...
'location', 'ne');
```

loglog

```
loglog(f,H2, 'b', 'linewidth',2);

yaxis(10^(-5), 10^(0.5), 10.^(-5:0));
xlabel('{\itf}  (Hz)'); grid;
title('low pass filter');
```

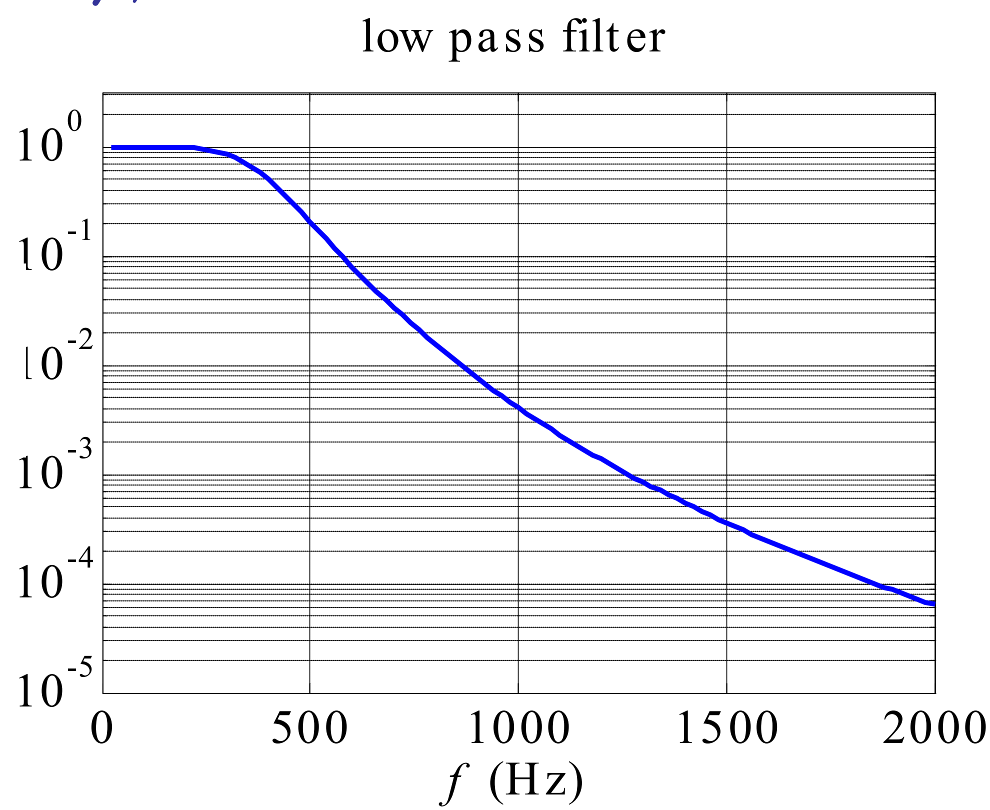


semilogy(X1,Y1,...) plots all Yn versus Xn pairs. If only one of Xn or Yn is a matrix,

semilogy

```
semilogy(f,H2, 'b', 'linewidth',2);

yaxis(10^(-5), 10^(0.5), 10.^(-5:0));
xlabel('{\itf}  (Hz)'); grid;
title('low pass filter');
```

semilogy
creates a plot using a base 10 logarithmic scale

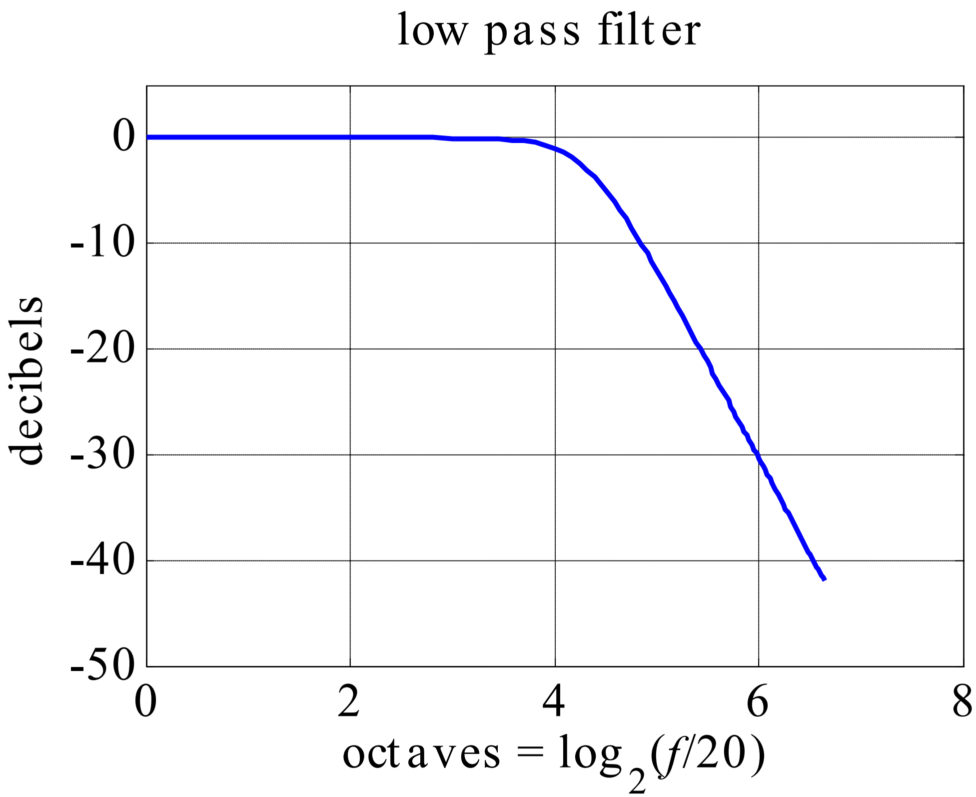


```
plot(log2(f/20), 10*log10(H2),'b');

xaxis(0,8, 0:2:8); yaxis(-50,5,-50:10:0);
xlabel('octaves = log_2({\itf}/20)');
ylabel('decibels'); grid;
title('low pass filter');
```

dB vs. octaves

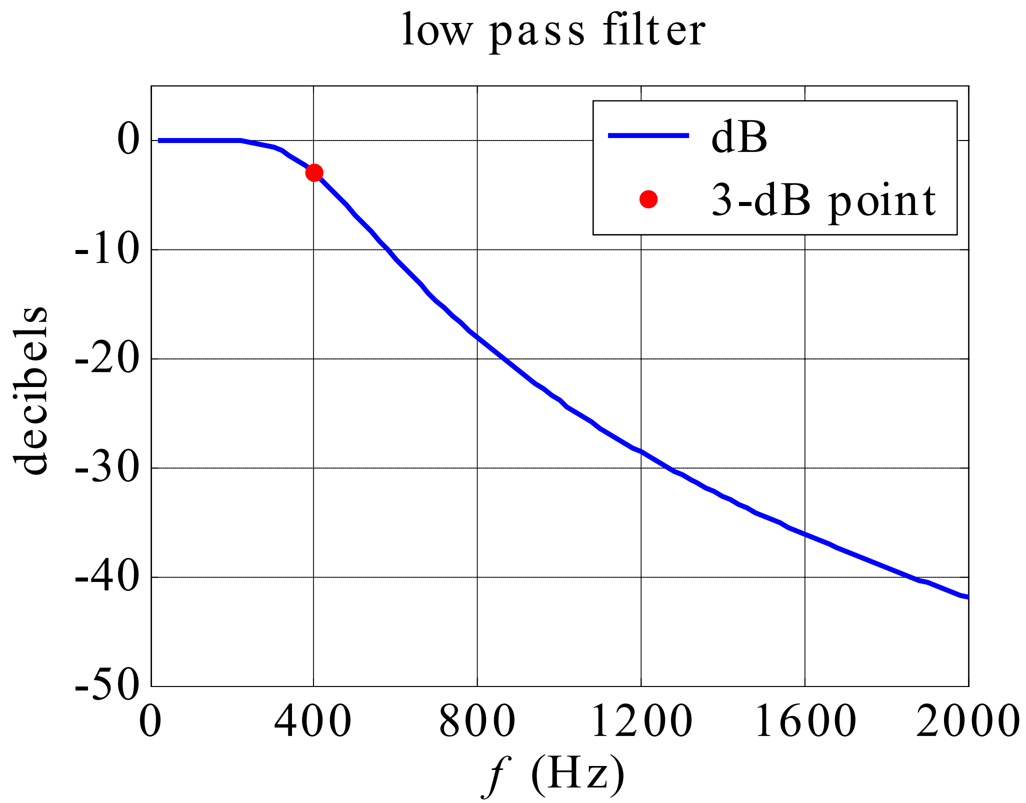
filter gain in dB

$$10\log_{10}\left(|H(f)|^2\right)$$



```
plot(f, 10*log10(H), 'b', 'linewidth',2);
hold on; plot(f0,10*log10(0.5), 'r.', ...
'markersize',20);

xaxis(0,2000, 0:400:2000); yaxis(-50,5,-50:10:0);
xlabel('{\itf}  (Hz)'); ylabel('decibels'); grid;
title('low pass filter');
legend(' dB', ' 3-dB point',...
'location', 'ne');
```

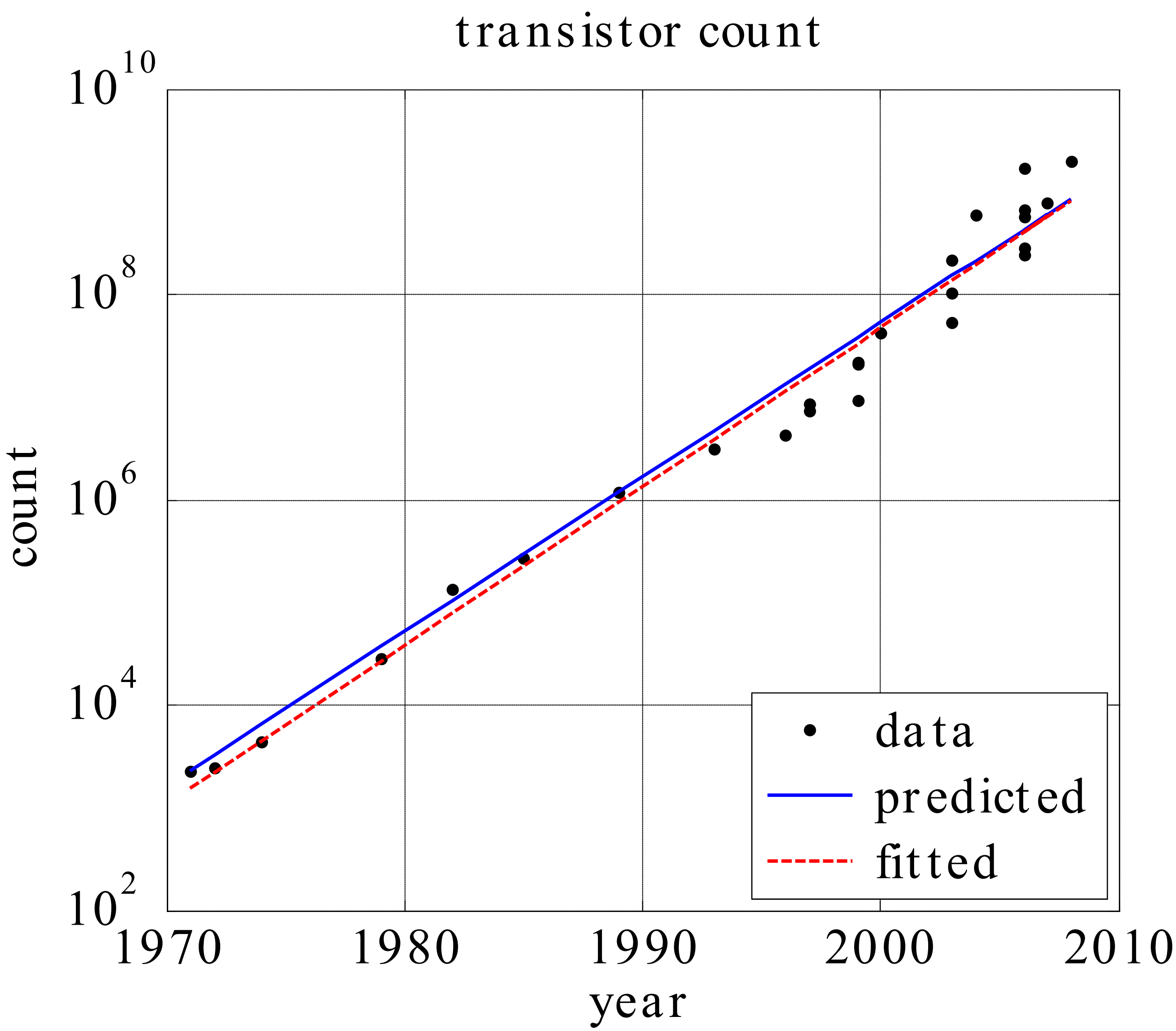
dB vs. Hz



semilogy example – Moore's law

```
Y = load('transistor_count.dat'); % file on webpage

y = Y(:,1);                   % transistor count
t = Y(:,2);                   % year

c1 = y(1);                    % c1 = 2300, starting count
t1 = t(1);                    % t1 = 1971, starting year

c = c1 * 2.^((t-t1)/2);                % predicted count

a = 1540.1813; b = 0.5138;             % least-squares fit
f = a * 2.^(b*(t-t1));                 % fitted curve

figure;
semilogy(t,y,'k.', t,c,'b-', t,f,'r--');
xlabel('year'); ylabel('count');
title('transistor count');
legend(' data', ' predicted', ' fitted',...
       'location','se');
```

semilogy example – Moore's law



3d order Butterworth lowpass filter

Plotyy:

Create graph with two y-axes

frequency response

imaginary unit

$$H(f) = \frac{1}{(1+s)(1+s+s^2)}, \quad s = \frac{jf}{f_0}$$

$$G(f) = 10\log_{10}(|H(f)|^2)$$

magnitude response (dB)

$$\theta(f) = -\mathrm{Arg}(H(f))$$

phase response (radians)

$$|H(f)|^2 = \frac{1}{1+(f/f_0)^6}$$

magnitude response in absolute units

plotyy

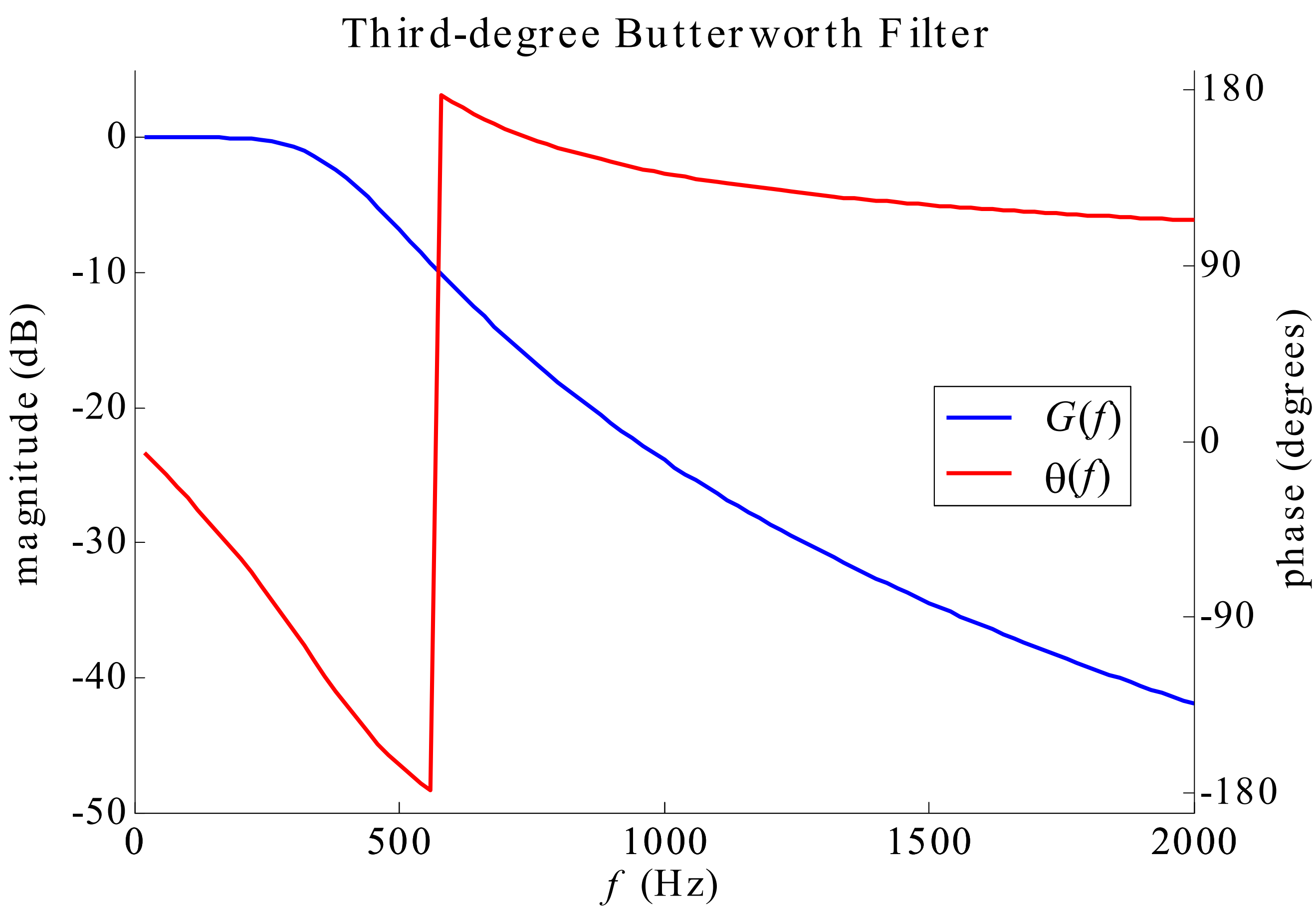
```
f = linspace(20,2000,100); f0 = 400; s = j*f/f0;

H = 1./((1+s).*(1 + s + s.^2));
G = 10*log10(abs(H).^2);
th = angle(H) * 180/pi;%convert radian to degrees

[a,h1,h2] = plotyy(f,G, f,th);

xlabel('{\itf}  (Hz)');

axes(a(1));
yaxis(-50,5, -50:10:0);
ylabel('magnitude (dB)');

axes(a(2));
yaxis(-190,190, -180:90:180);
ylabel('phase (degrees)');

set(h1, 'linewidth',2, 'color', 'b');
set(h2, 'linewidth',2, 'color', 'r');
legend([h1,h2], ' G(f)', ' \theta(f)');
```

a=[a(1),a(2)],h1,h2 are axis and line handles,

axes activates left, then right axis

set line properties

plotyy



title, x-y axis labels, linestyles, colors, legends, and tickmarks can also be set from the figure window (select left or right y-axis from the plot browser)

scatter plots

```
scatter(x,y, area, color);
```

```
plot(x,y, '.');
```

similar to this,

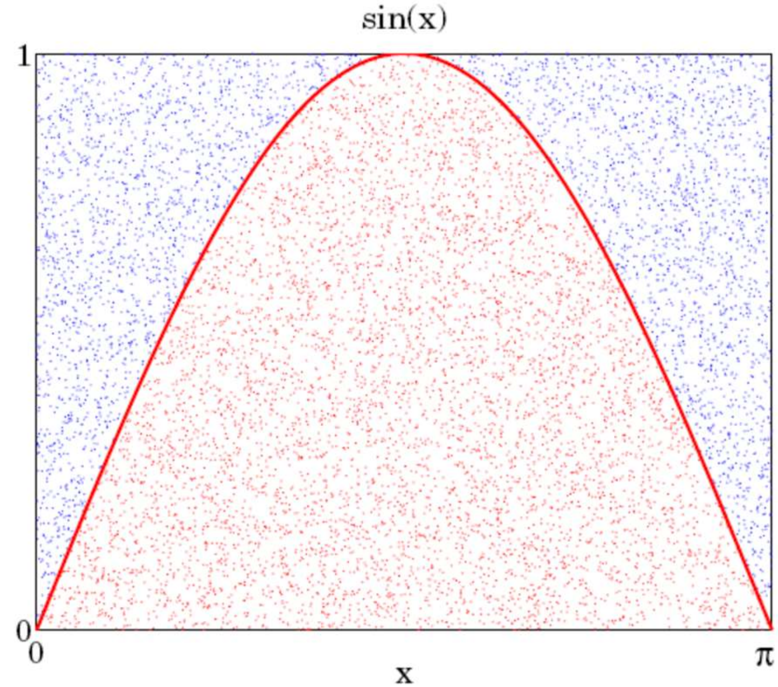
but **scatter** allows more control of the area and color of dots

```
>> help scatter
>> doc scatter
```

```
N=10000; rng(101);
x = pi * rand(1,N);
y = rand(1,N);

i = find(y < sin(x));
j = find(y > sin(x));

scatter(x(i),y(i),1,'r');
hold on;
scatter(x(j),y(j),1,'b');

x = linspace(0,pi,100);
y = sin(x);
plot(x,y,'r-');

A = length(i)/N * pi

A =
    1.9915
```

Example of a Monte Carlo calculation of the area under the curve: $\sin(x)$, $0 \le x \le \pi$

actual area is: $A = 2$



estimated area is the rectangular area times the fraction of the (x,y) pairs lying under the curve

subplots

general syntax:

subplot(n,m,p);

n x m = box pattern

p = counting figures across rows

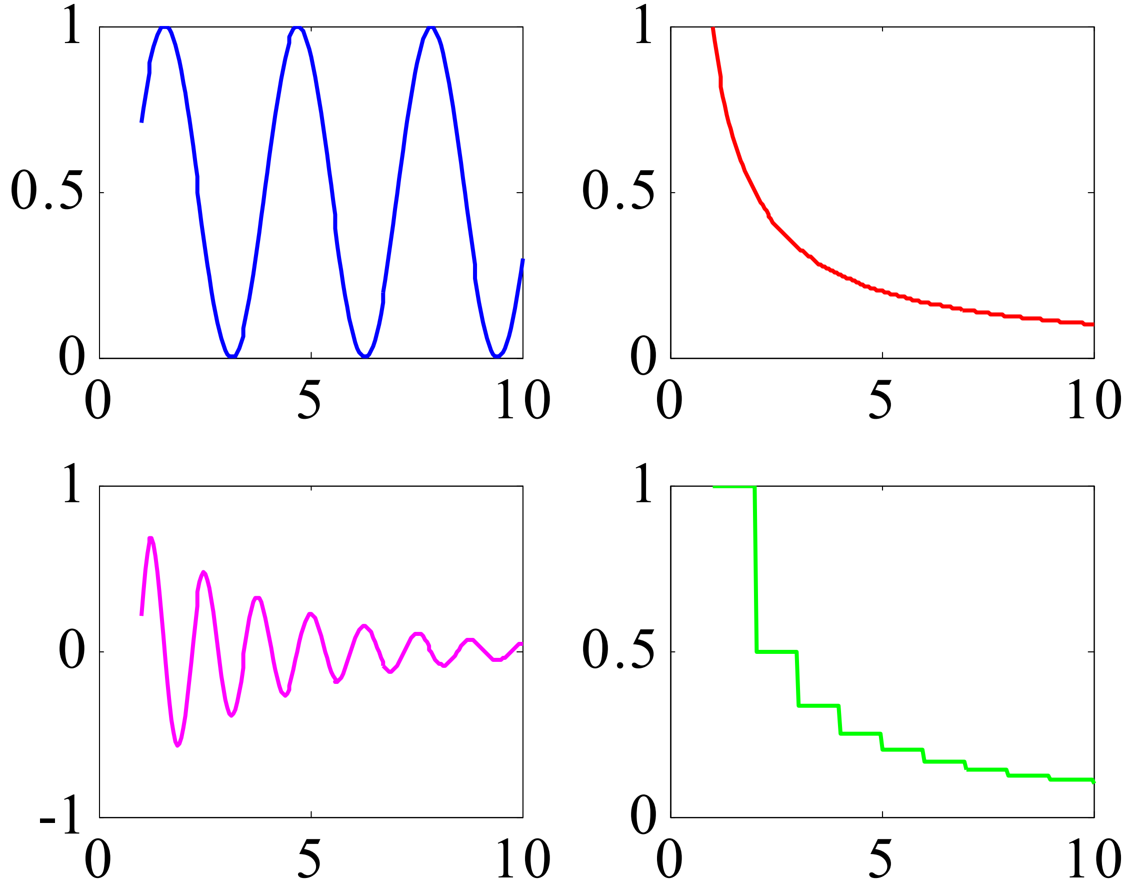
3 x 4 pattern

1	2	3	4
5	6	7	8
9	10	11	12

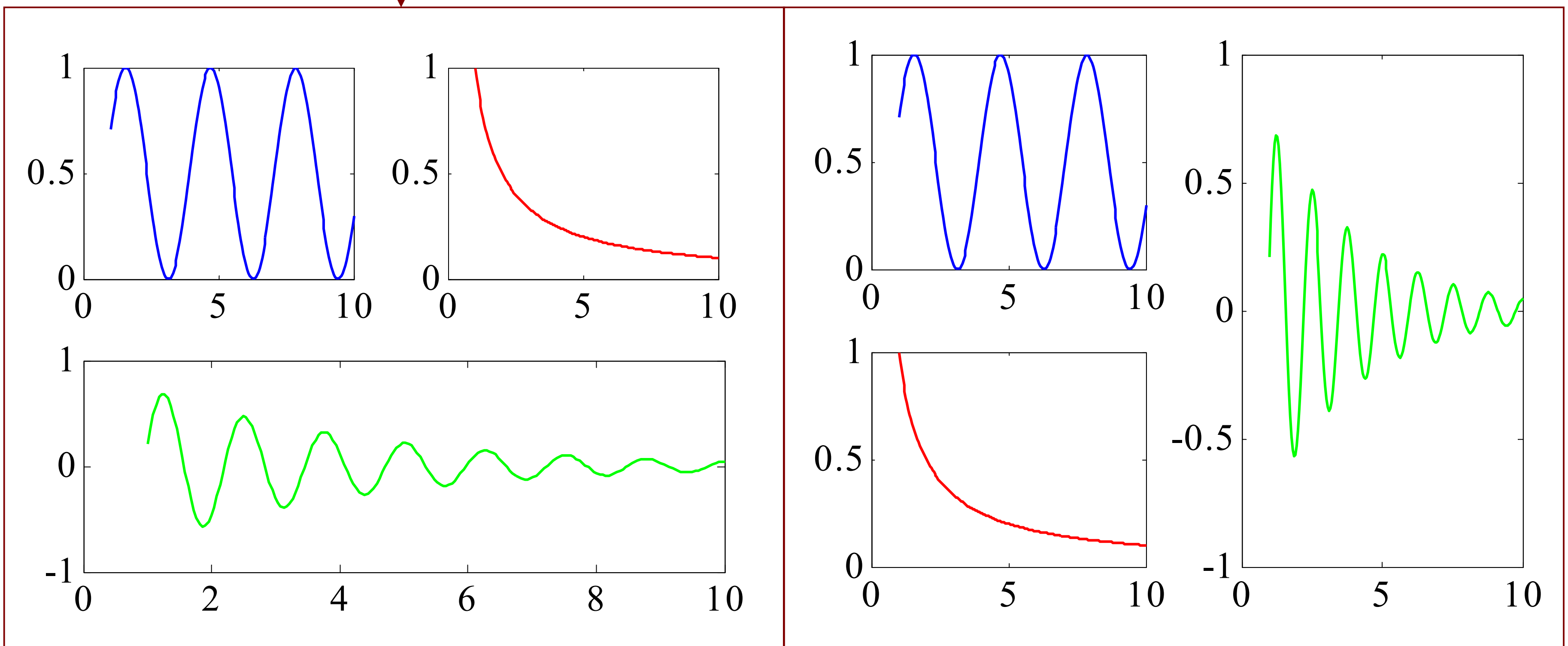
```
subplot(3,4,1)
subplot(3,4,2)
etc.
```

```
x = linspace(1,10,200);

y1 = sin(x).^2;
y2 = 1./x;
y3 = exp(-0.3*x).*cos(5*x);
y4 = 1./floor(x);
```

```
subplot(2,2,1); plot(x,y1,'b');
subplot(2,2,2); plot(x,y2,'r');
subplot(2,2,3); plot(x,y3,'m');
subplot(2,2,4); plot(x,y4,'g');
```



```
subplot(2,2,1); plot(x,y1,'b');
subplot(2,2,2); plot(x,y2,'r');
subplot(2,1,2); plot(x,y3,'g');
```

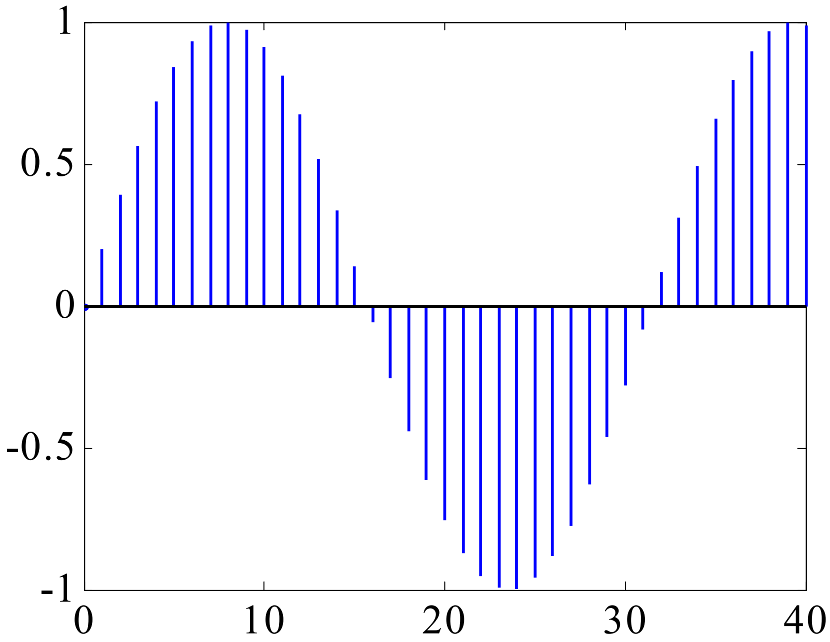


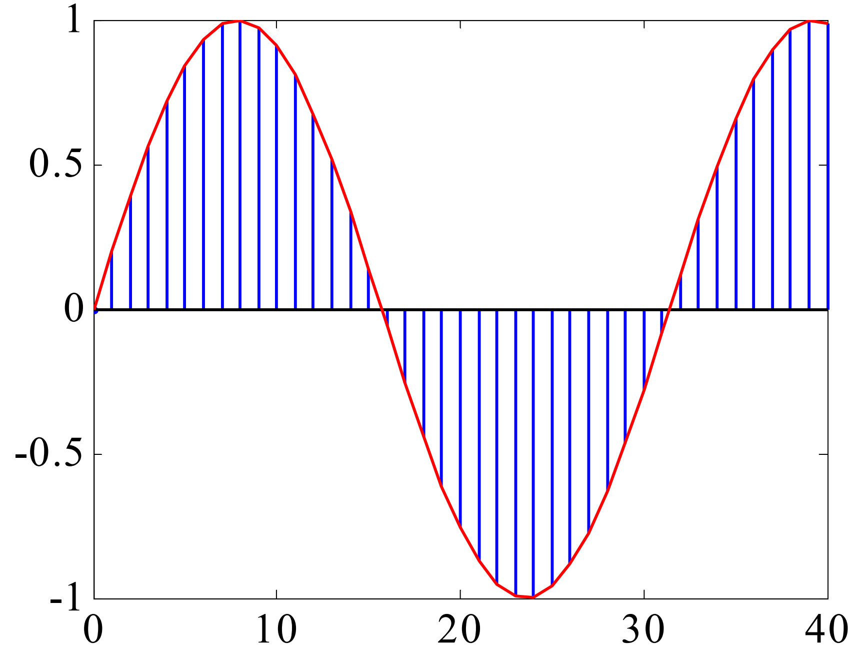

```
subplot(2,2,1); plot(x,y1,'b');
subplot(2,2,3); plot(x,y2,'r');
subplot(1,2,2); plot(x,y3,'g');
```

stem plots

Plot discrete sequence data

```
x = linspace(0,40,41);
y = sin(x/5);

stem(x,y,'b','marker','none');
```



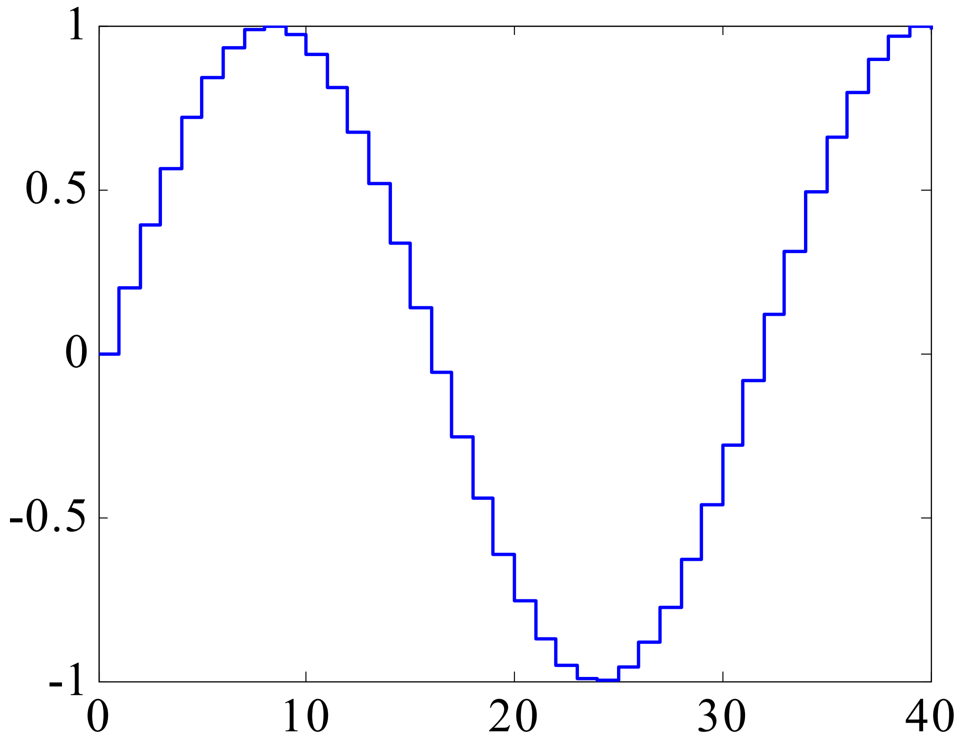



```
stem(x,y,'b','marker','none');
hold on; plot(x,y,'r-');
```

useful for displaying discrete-time signals in DSP applications

stairs

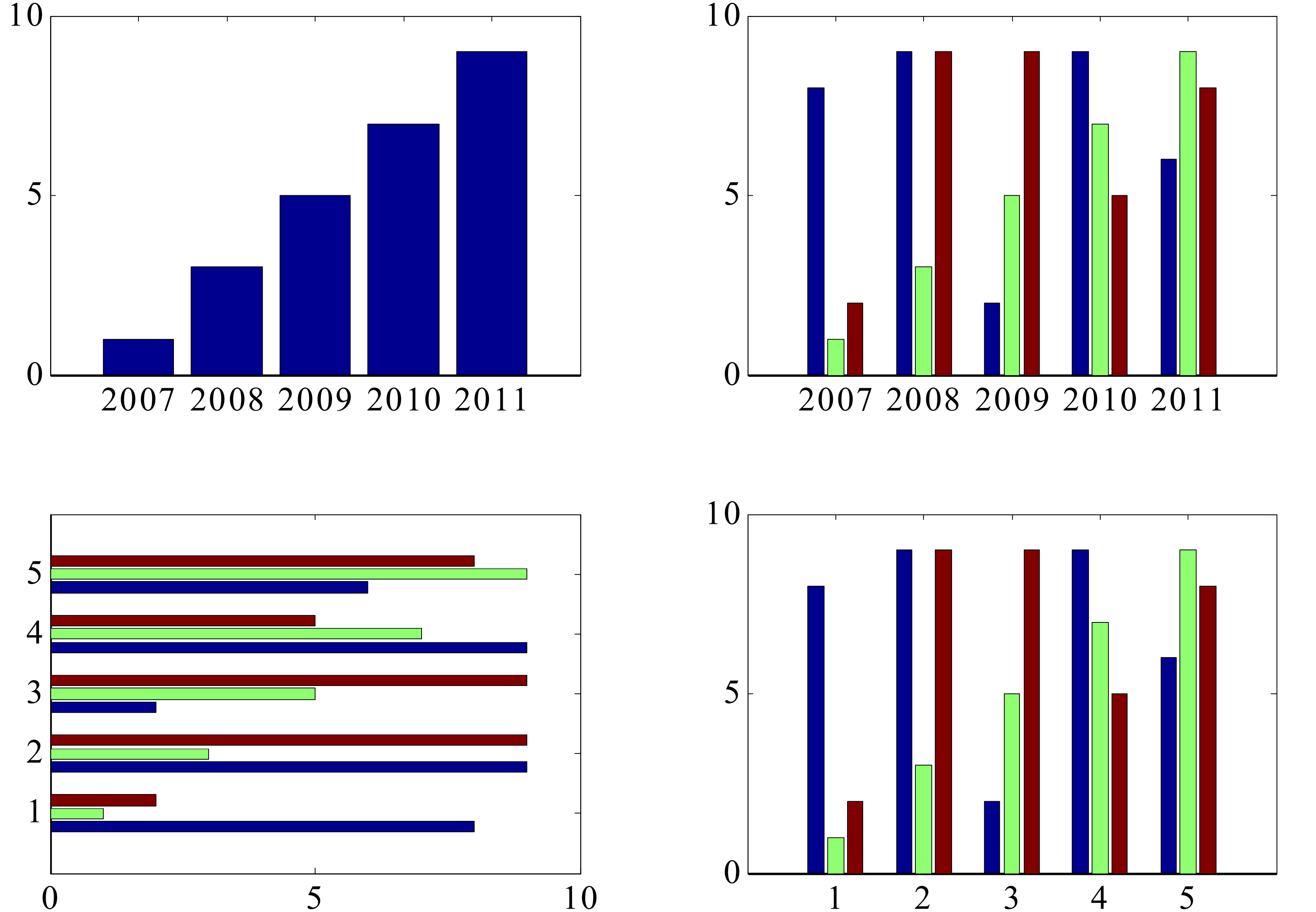
```
x = linspace(0,40,41);
y = sin(x/5);

stairs(x,y,'b');
```



bar graphs

```
Y =[8 1 2
    9 3 9
    2 5 9
    9 7 5
    6 9 8];

x = 2007:2011; y = Y(:,2);

subplot(2,2,1); bar(x,y);
subplot(2,2,2); bar(x,Y);
subplot(2,2,3); barh(Y);
subplot(2,2,4); bar(Y);
```

bar graphs


10
5
0
2007 2008 2009 2010 2011
10
5
0
2007 2008 2009 2010 2011
5
4
3
2
1
0
5
10
10
5
0
1
2
3
4
5

histograms

initialize generator

define bins

simulate 600 random grades

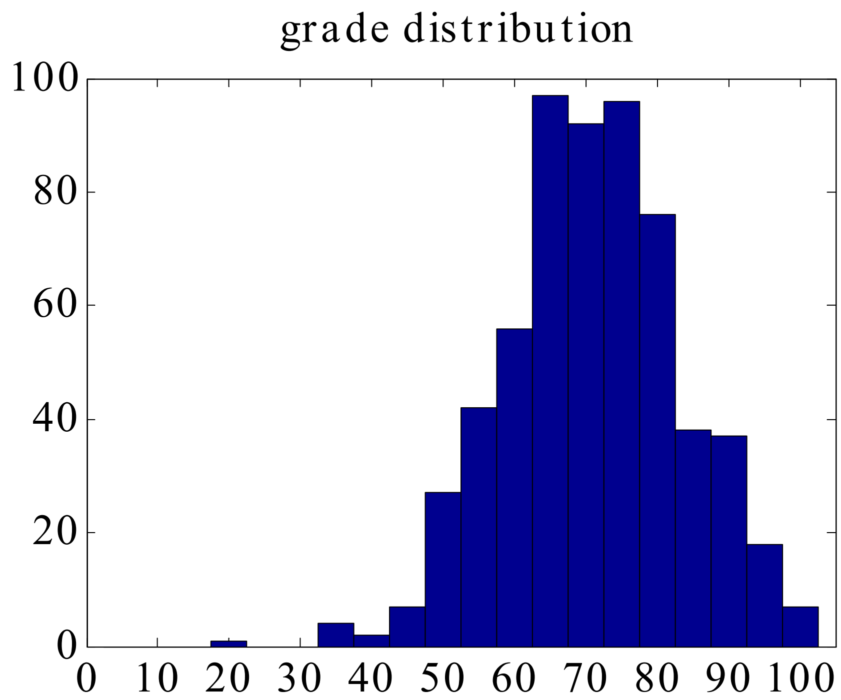
H = vector of histogram values

```
rng(101);
b = 0:5:100;
g = ceil(70 + 12 * randn(1,600));

figure; H = hist(g,b);

xaxis(0,105, 0:10:100);
title('grade distribution');
```

improved version

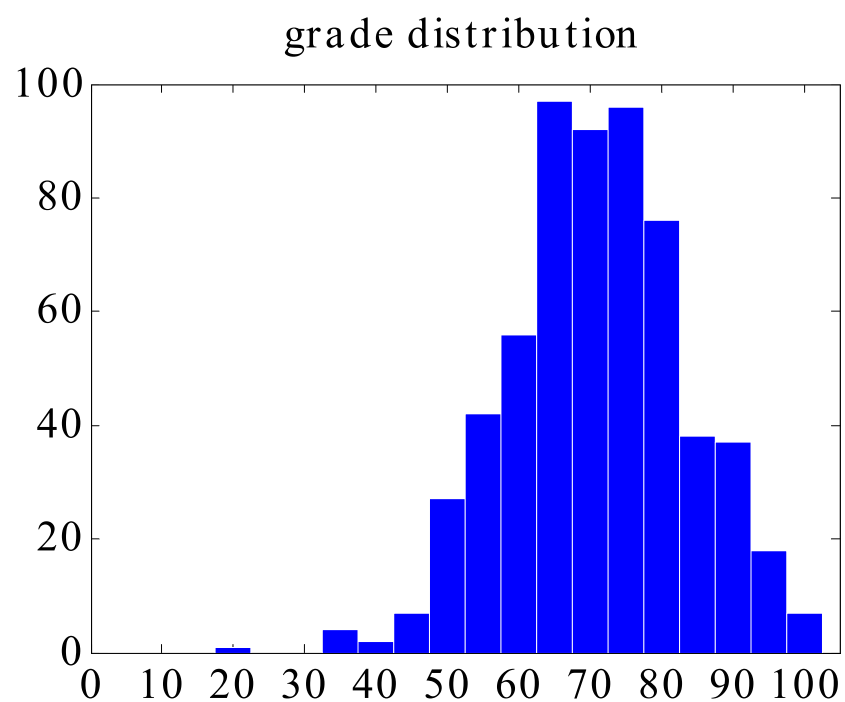
```
figure; H = hist(g,b);

h = findobj(gca,'Type','patch');
set(h,'FaceColor','b','EdgeColor','w');

xaxis(0,105,0:10:100);
title('grade distribution');
line([0,105],[0,0],'linewidth',0.3);
```

histograms



default

improved



mean = 70.86
std = 12.39
median = 71
mode = 69

pie charts

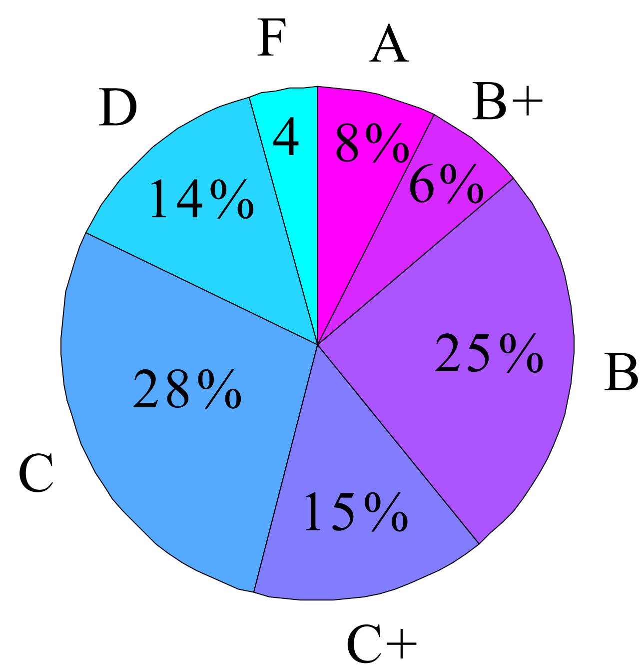
```
Na  = length(find(g>=90));
Nbp = length(find(g<90 & g>=85));
Nb  = length(find(g<85 & g>=75));
Ncp = length(find(g<75 & g>=70));
Nc  = length(find(g<70 & g>=60));
Nd  = length(find(g<60 & g>=50));
Nf  = length(find(g<50));
N = [Nf, Nd, Nc, Ncp, Nb, Nbp, Na];
pie(N, {'F','D','C','C+','B','B+','A'});
colormap cool;
```

number of A's, B+'s, B's, etc.

```
Nper = round(100 * N/sum(N))
```

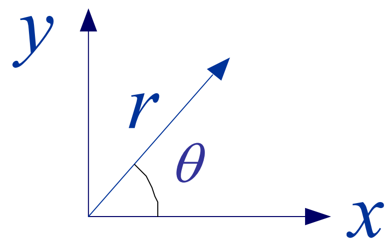
percentages were added using the plot editor

```
% N = [26 81 169 89 152 38 45];
      F  D   C   C+  B   B+  A
```



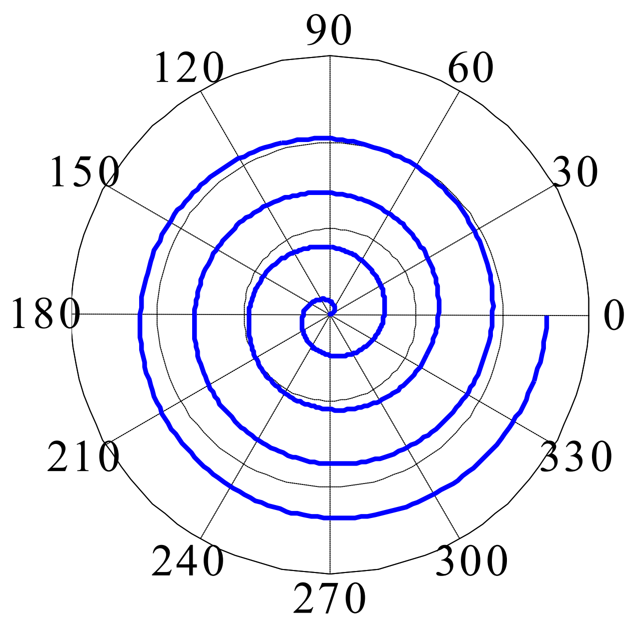
polar functions

$r = f(\theta)$



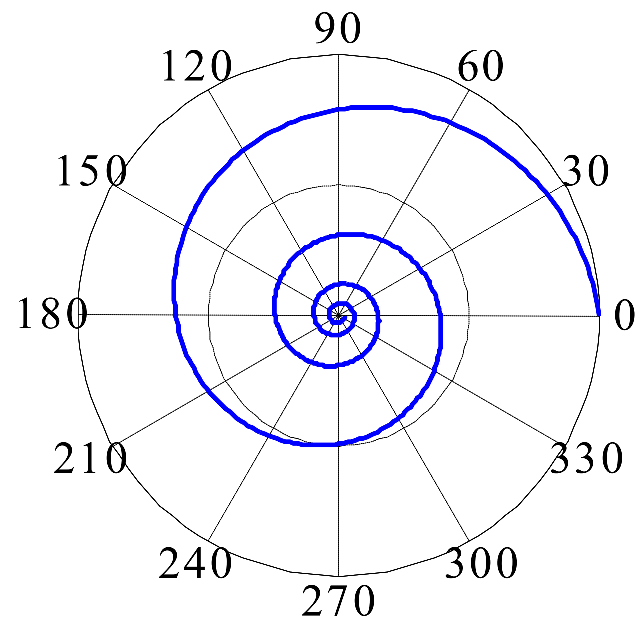

```
th = linspace(0,8*pi,800);

r = th;
polar(th,r);
```

Archimedean spiral



polar plots

logarithmic spiral




```
r = exp(-0.15*th);
polar(th,r);
```

3D plotting functions

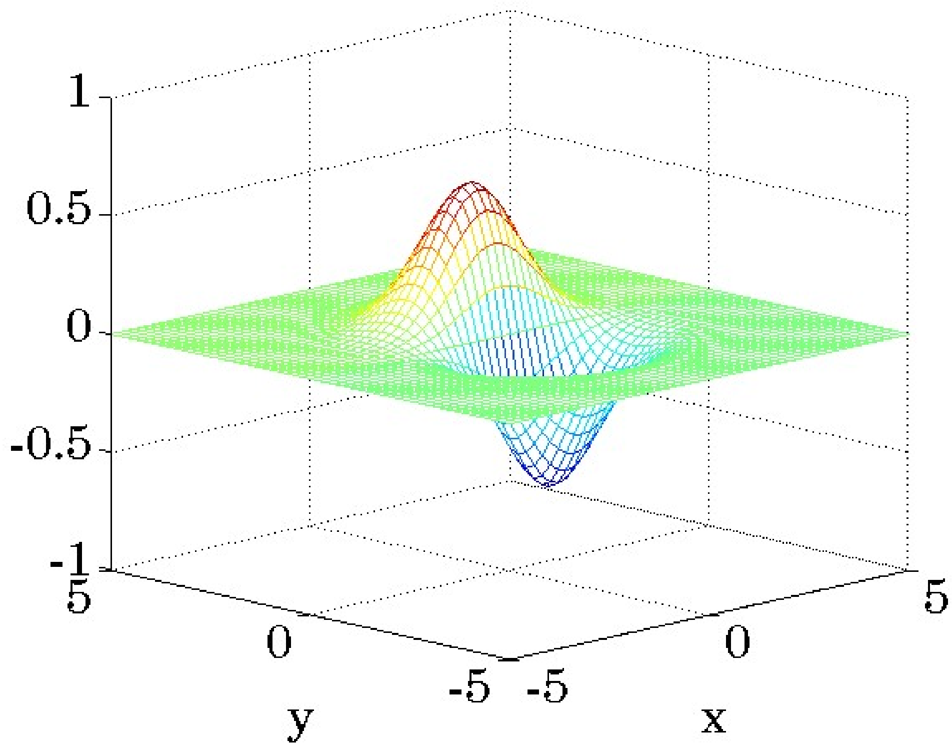
plot3,ezplot3	x-y-z line plot
contour,ezcontour	contour plot
contourf,ezcontourf	filled contour plot
mesh,ezmesh	wireframe surface plot
meshc,ezmeshc	wireframe plus contour
meshz	wireframe with curtain
surf,ezsurf	solid surface plot
surfc,ezsurfc	surface plot plus contour
waterfall	waterfall plot
stem3,scatter3	3D stem and scatter
bar3,bar3h,pie3	3D bar & pie charts
fill3	polygon fill
comet3	animated **plot3**

meshgrid was discussed in week-3

```
x = linspace(-5,5,51);
y = linspace(-5,5,51);

[X,Y] = meshgrid(x,y);

Z = Y .* exp(-(X.^2 + Y.^2)/2);

mesh(X,Y,Z);
```

mesh

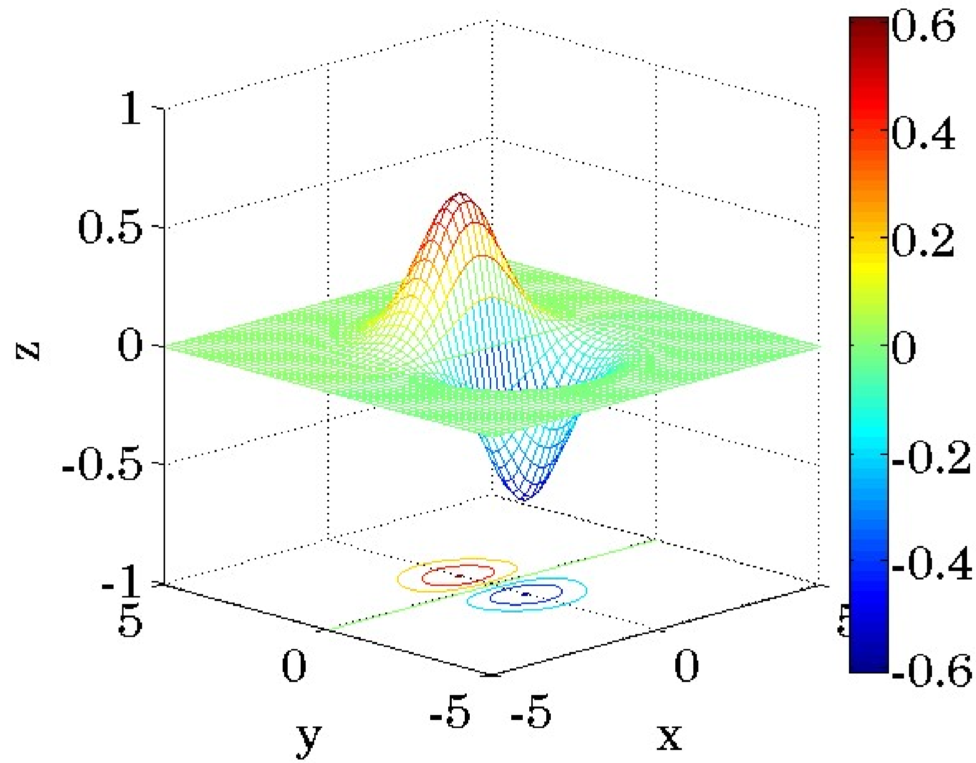
$$z = f(x, y) = y \exp(-(x^2 + y^2)/2)$$



meshc

```
meshc(X,Y,Z);
view(-45,15);
colorbar;
```

```
>> doc view;
>> doc colorbar;
>> doc colormap;
```



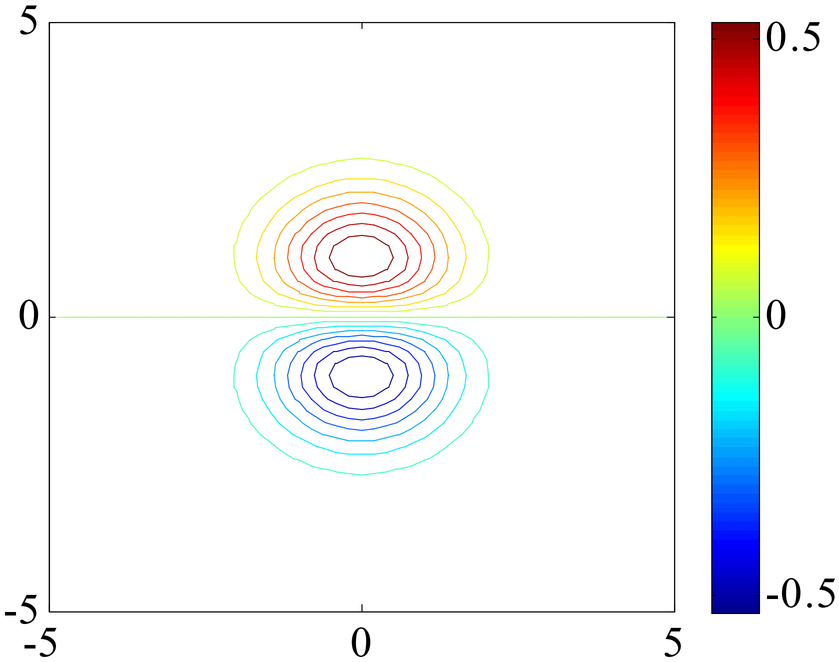
contour
contourf

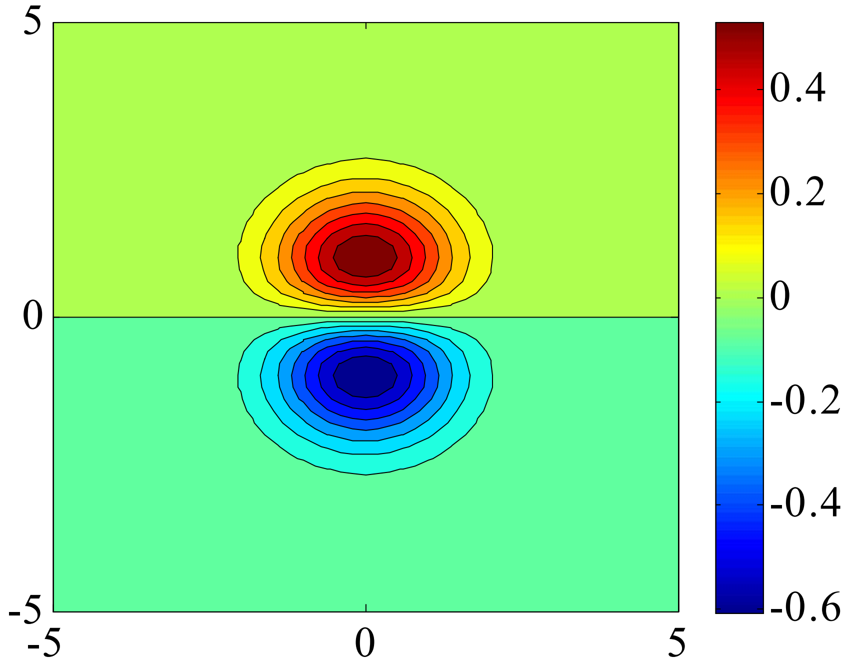
number of contour levels

```
contour(X,Y,Z,15);
colorbar;
```

filled contour

```
contourf(X,Y,Z,15);
colorbar;
```



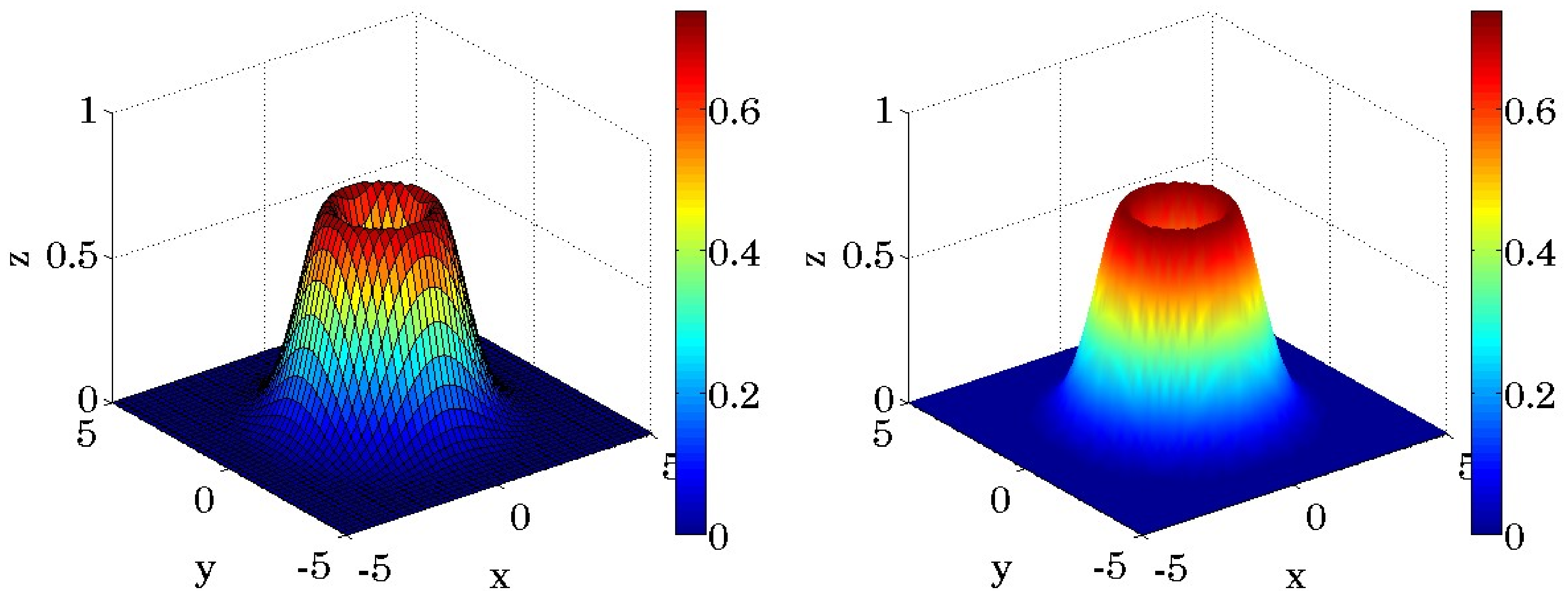


surf

```
x = linspace(-5,5,51);
y = linspace(-5,5,51);
[X,Y] = meshgrid(x,y);
Z = (X.^2 + Y.^2) .* exp(-(X.^2 + Y.^2)/2);
```

```
surf(X,Y,Z);
colorbar;
```

```
surf(X,Y,Z);
shading interp;
colorbar;
```

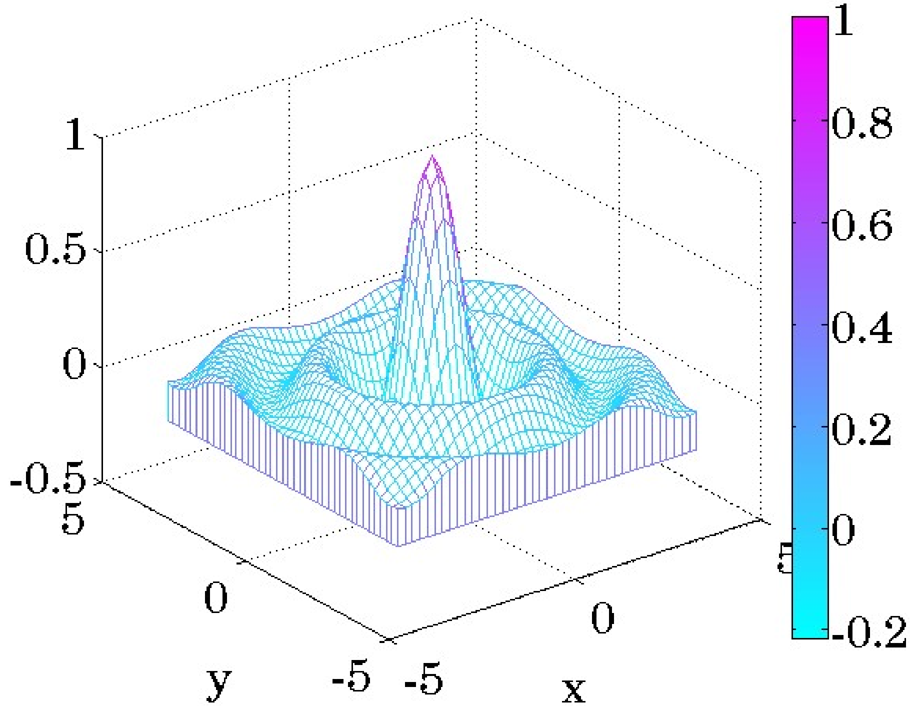


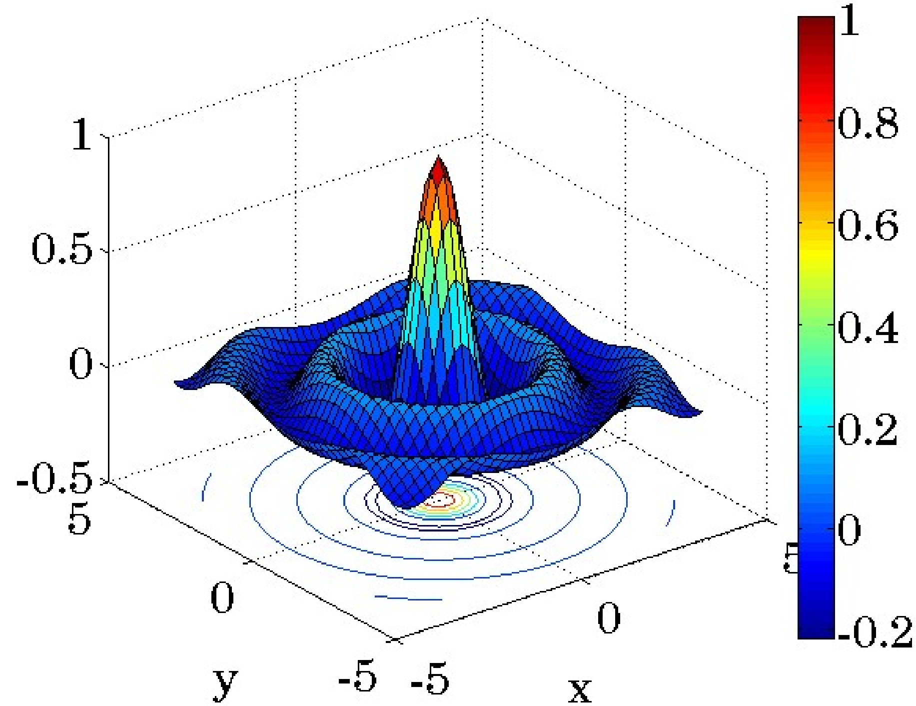
```
x = linspace(-4,4,41);
y = linspace(-4,4,41);
[X,Y] = meshgrid(x,y);
Z = sinc(sqrt(X.^2 + Y.^2));   % help sinc
```

meshz
surfc

```
meshz(X,Y,Z);
colormap cool;
colorbar;
```

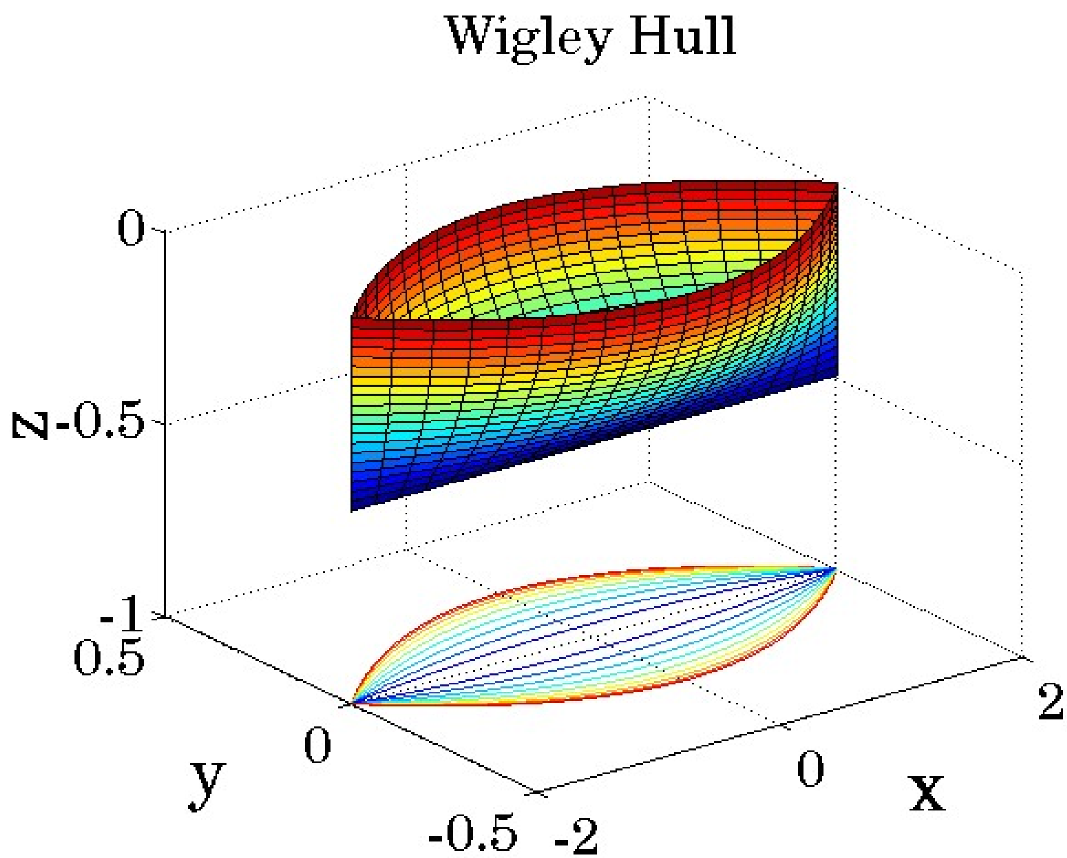
```
surfc(X,Y,Z);
colorbar;
```





surfc

```
B = 0.5; L = 4; D = 0.5;
x = linspace(-L/2, L/2, 21);
z = linspace(-D, 0, 21);
[X,Z] = meshgrid(x,z);

Y = B/2 * (1 - 4*X.^2/L^2) .* (1 - Z.^2/D.^2);

figure;
surfc(X,Y,Z);
hold on;
surfc(X,-Y,Z);
```



Wigley Hull

$$y = \pm \frac{B}{2}\left(1 - \frac{4x^2}{L^2}\right)\left(1 - \frac{z^2}{D^2}\right)$$

Reference:
A. Gilat, MATLAB, An Introduction with Applications, 4/e, Wiley, 2011

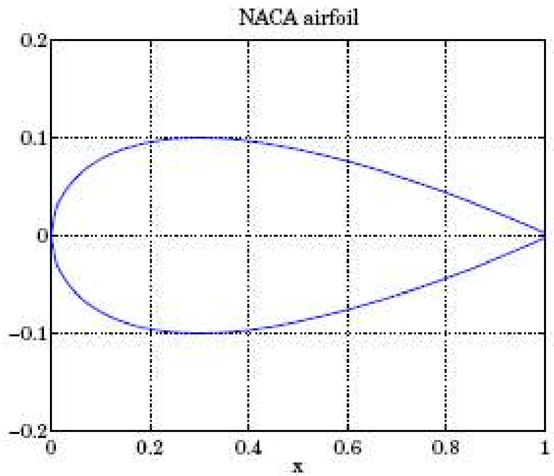
National Advisory Committee for Aeronautics

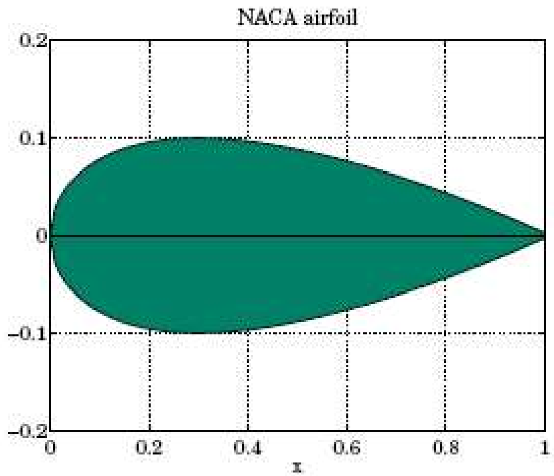
NACA airfoils

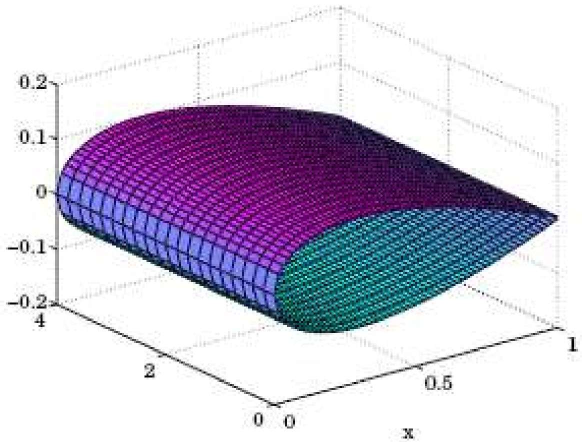
area

$$y = \pm \frac{tc}{0.2}\left[0.2969\sqrt{\frac{x}{c}} - 0.1260\left(\frac{x}{c}\right) - 0.3516\left(\frac{x}{c}\right)^2 + 0.2843\left(\frac{x}{c}\right)^3 - 0.1015\left(\frac{x}{c}\right)^4\right]$$

$$0 \le x \le c$$



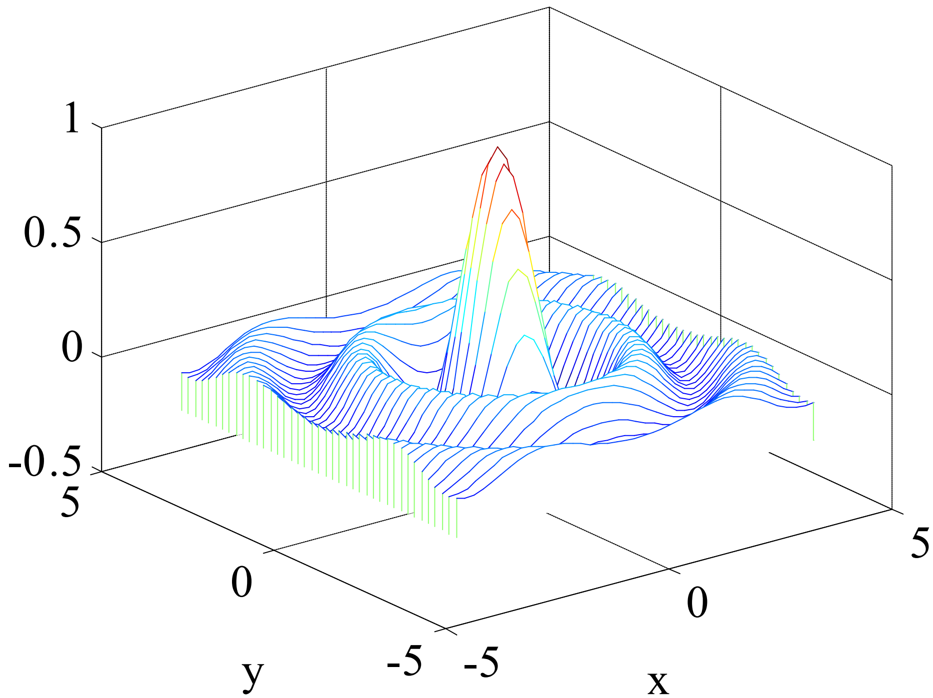




Reference:
A. Gilat, MATLAB, An Introduction with Applications, 4/e, Wiley, 2011

area

```
c = 1; t = 0.2;

f = @(x) t*c/0.2 * (0.2969*sqrt(x/c) ...
         - 0.1260*(x/c) -0.3516*(x/c).^2 ...
         + 0.2843*(x/c).^3 - 0.1015*(x/c).^4);

x = linspace(0,c,101);

y = f(x);

figure; plot(x,y,'b-', x,-y,'b-');
figure; area(x,y); hold on; area(x,-y);
colormap summer;

w = 4;
y = 0:w/20:w;
x = 0:c/60:c;

[X,Y] = meshgrid(x,y);
Z = f(X);

figure; surf(X,Y,Z); hold on; surf(X,Y,-Z);
colormap cool;
```

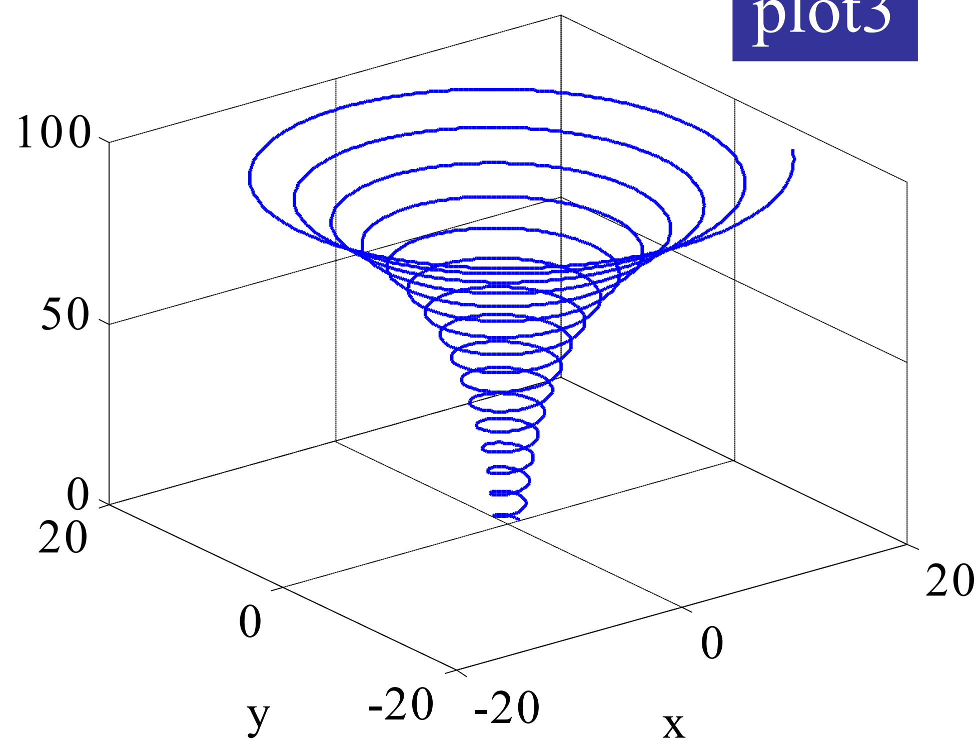


waterfall

unidirectional mesh plot

```
waterfall(X,Y,Z);
```

plot3

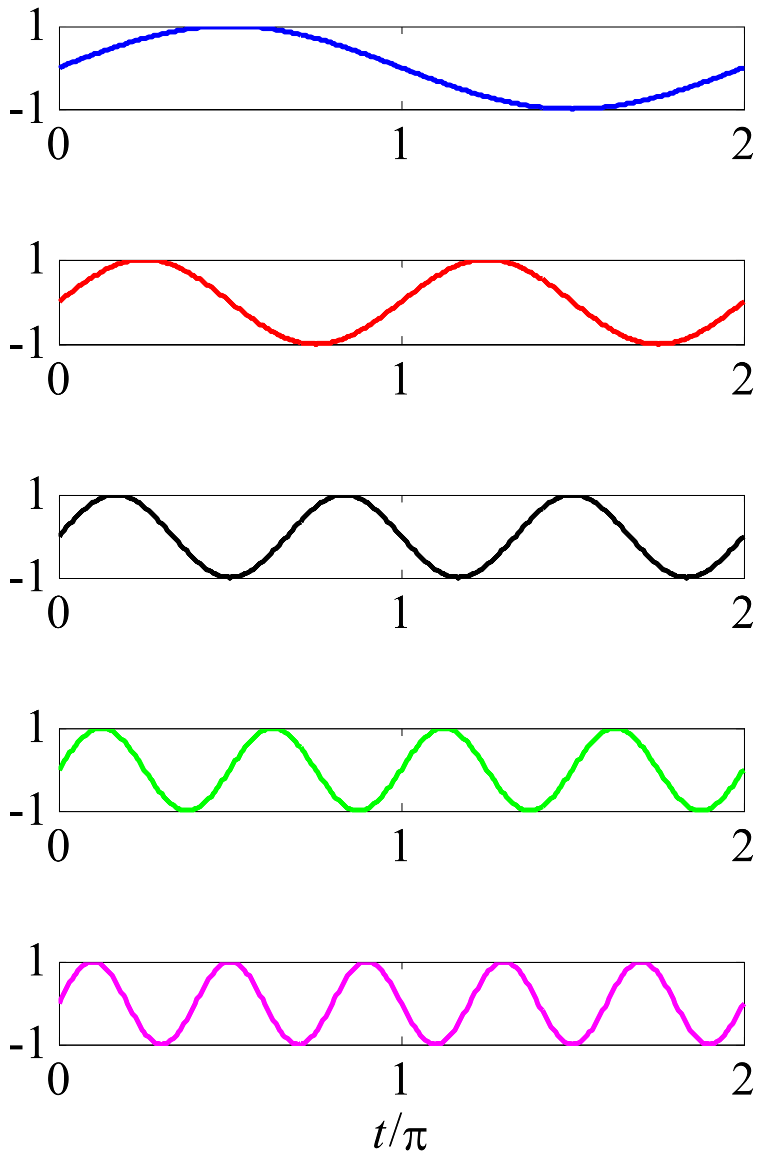



```
t = 0:0.01:100;
x = exp(0.03*t).*cos(t);
y = exp(0.03*t).*sin(t);
z = t;
plot3(x,y,z,'b');
grid on;
```

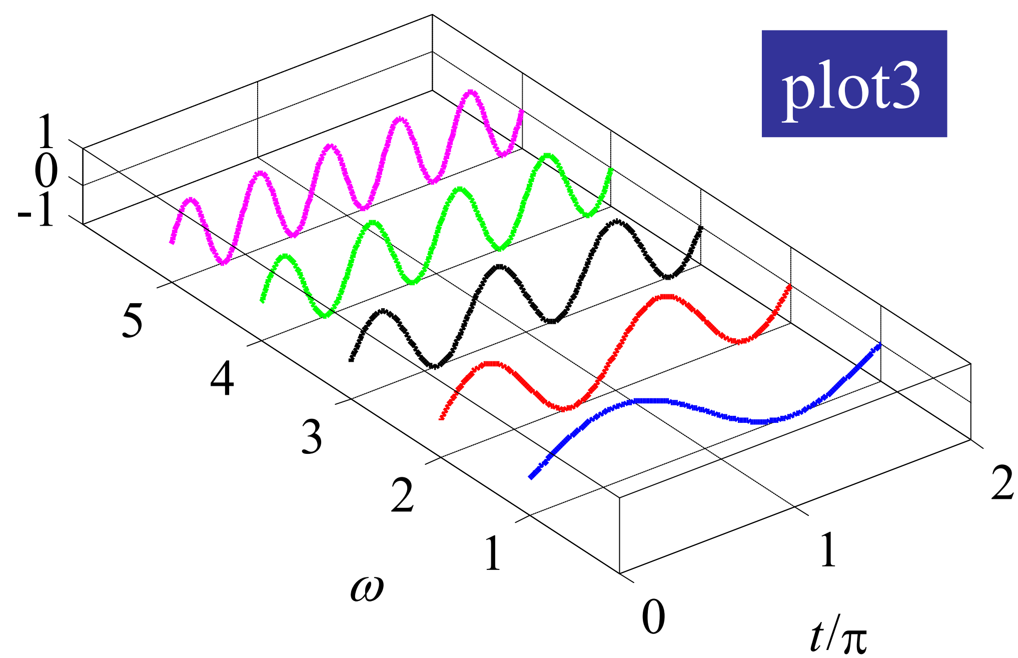
How to display multiple curves three-dimensionally

e.g., sin(ω t),
for ω = 1, 2, 3, 4, 5

subplot



plot3

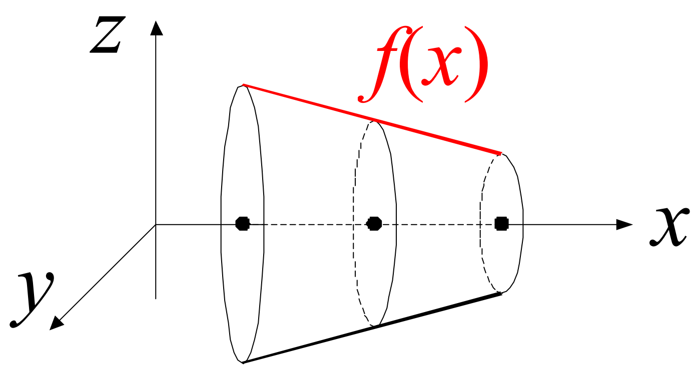


```
t = linspace(0,2*pi,361);

C = {'b', 'r', 'k', 'g', 'm'};

for k=1:5,
   subplot(5,1,k);
   z = sin(k*t);
   plot(t/pi,z,'color',C{k});
   xaxis(0,2, 0:2);
   yaxis(-1,1, [-1,1]);
end

xlabel('t/\pi');
```

subplot

```
t = linspace(0,2*pi,361);
y1 = ones(size(t));

C = {'b', 'r', 'k', 'g', 'm'};

for k=1:5,
   z = sin(k*t);
   plot3(t/pi, k*y1, z, 'color',C{k});
   hold on;
end

hold off; box on; grid on;
xaxis(0,2, 0:2); yaxis(0,6, 1:5);
xlabel('t/\pi'); ylabel('\omega');

set(gca,'DataAspectRatio',[1, 1.5, 5]);
```

plot3



How to generate surfaces of revolution, e.g., rotating a function $z = f(x)$ about the x-axis

```
x = linspace(a,b,N);
theta = linspace(0,2*pi,M);

[X,Th] = meshgrid(x,theta);

Y = f(X) .* cos(Th);
Z = f(X) .* sin(Th);

surf(X,Y,Z);      % or mesh()
```

assume $f(x)$ is defined over $a \le x \le b$

to rotate a function $f(z)$ about the z-axis, simply interchange roles of x, z, but do **`surf(X,Y,Z)`**

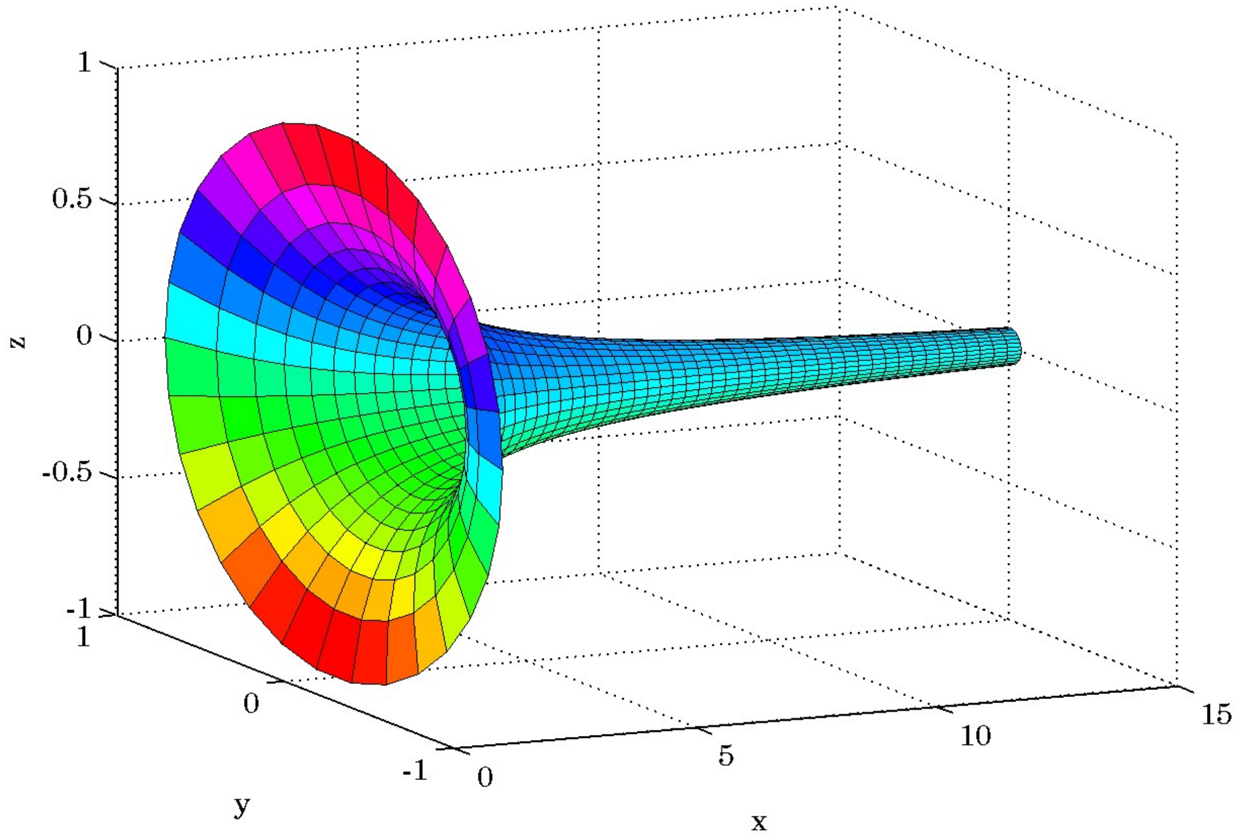
or, use the built-in function **`cylinder`**

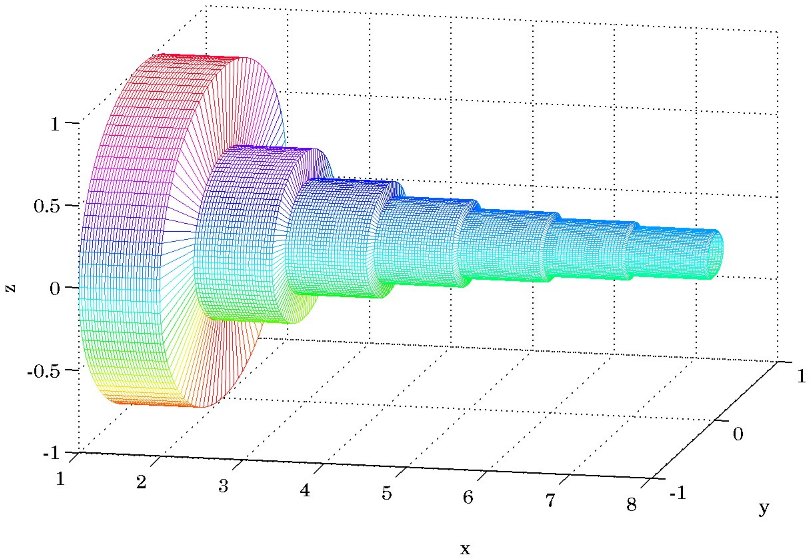
```
x = linspace(1,15,50);
th = linspace(0,2*pi,31);

[X,Th] = meshgrid(x,th);

F = 1./X;
Y = F.*cos(Th);
Z = F.*sin(Th);

surf(X,Y,Z);
view(-25,15);
colormap hsv;
```

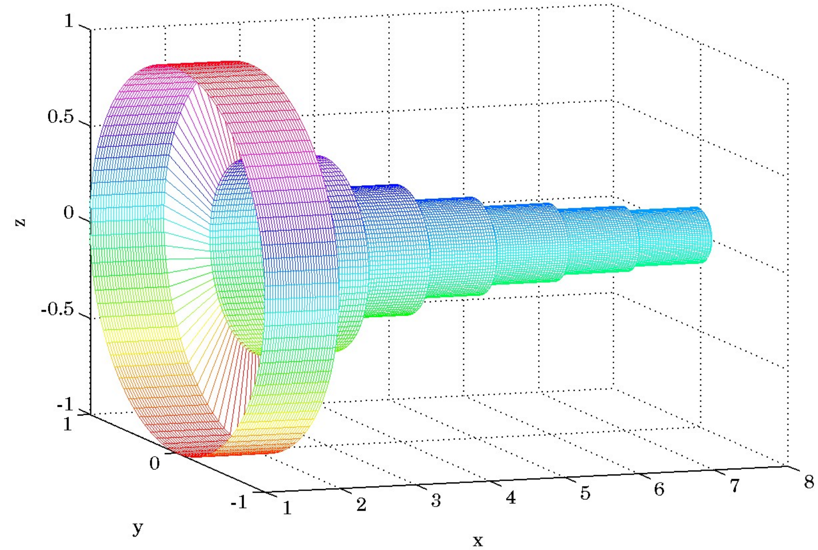
Torricelli's Trumpet,
aka Gabriel's Horn,

$f(x) = 1/x, \quad 1 \le x < \infty$

has finite volume,
but infinite area

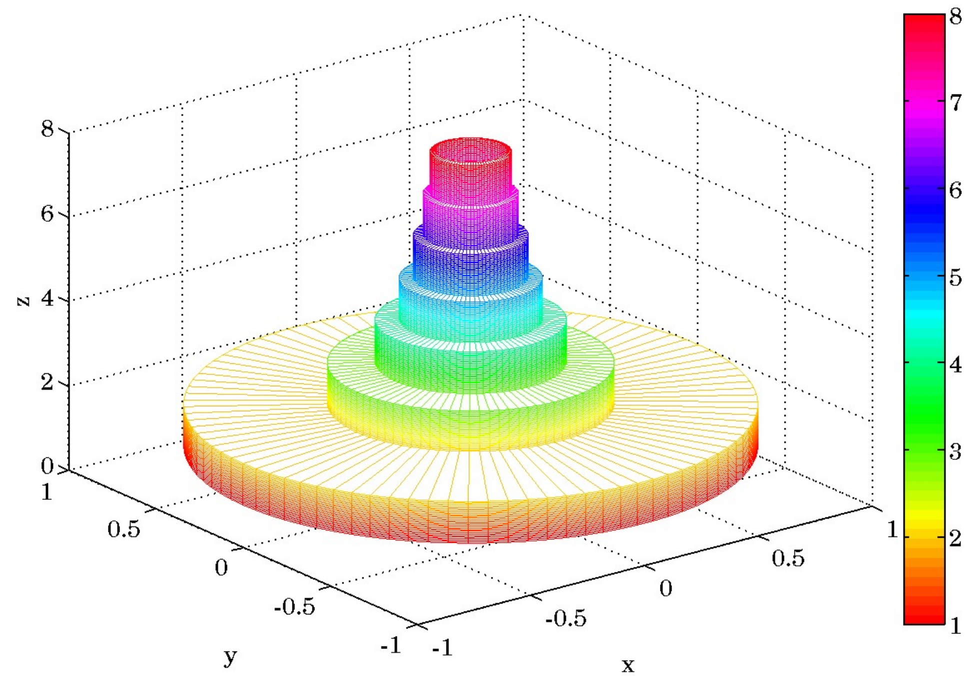






<u>Gabriel's Cake</u>

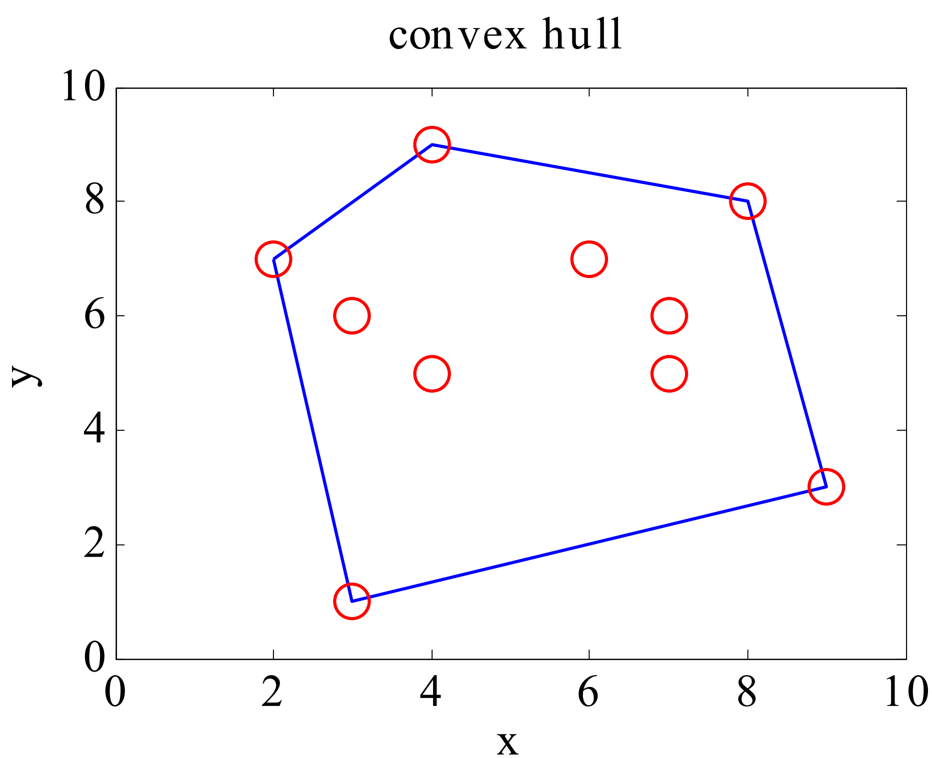
uses a step version of $f(x) = 1/x$

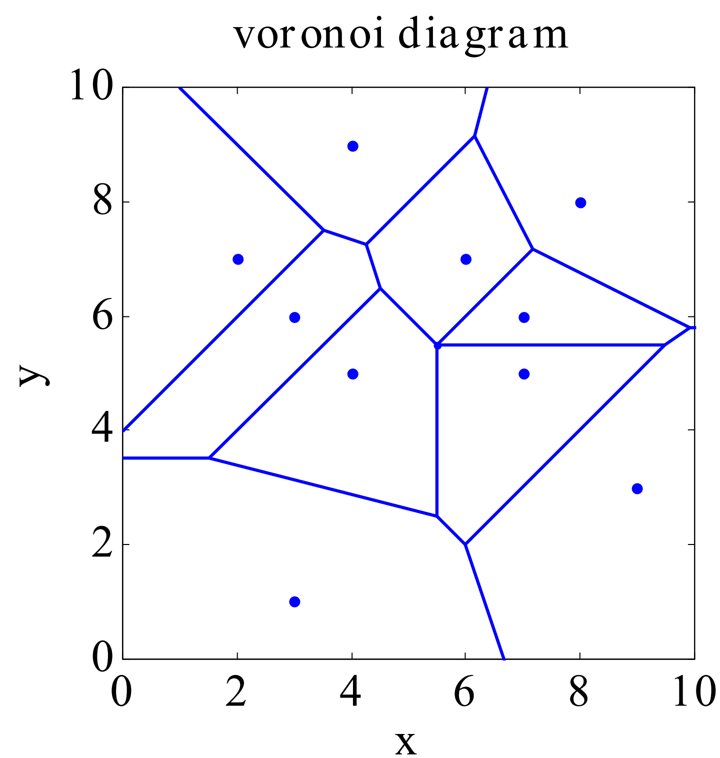


```
x = [6,3,2,7,4,3,9,4,8,7];
y = [7,6,7,6,5,1,3,9,8,5];

n = convhull(x,y);
plot(x(n),y(n),'b-',x,y,'ro');
```

convhull
voronoi

```
voronoi(x,y,'b-');
```

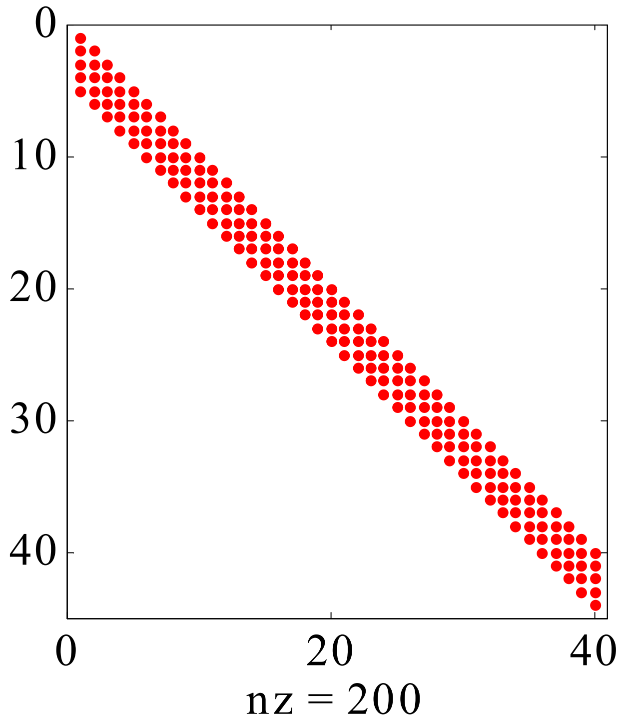




spy

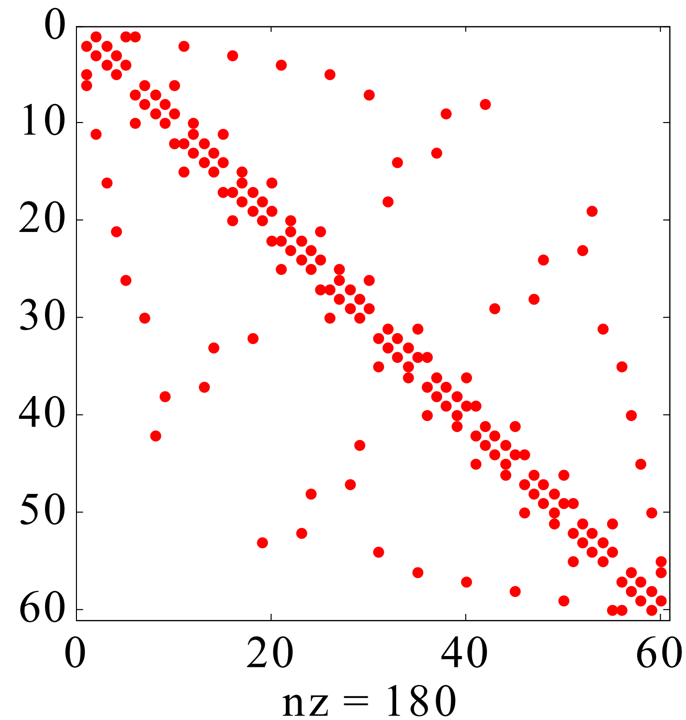
sparsity pattern

```
h = [2 3 5 8 4]';
N = 40;
H = convmtx(h,N);
spy(H,'r.');
```



convolution matrix

```
B = bucky; spy(B,'r.');
```



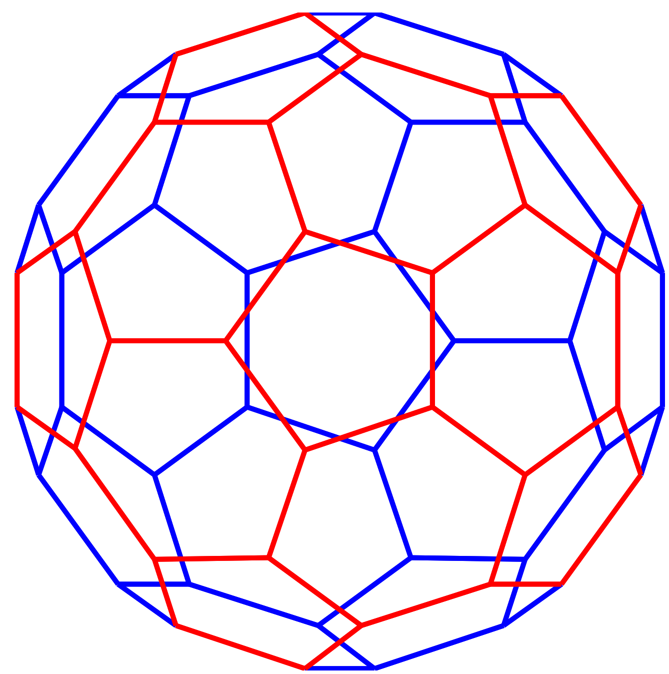
60 x 60 sparse adjacency matrix of the connectivity graph of the Bucky ball, geodesic dome, soccer ball, and the carbon-60 fullerene molecule

```
[B,V] = bucky;
H = sparse(60,60);
k = 31:60;
H(k,k) = B(k,k);

% Visualize the variables
gplot(B-H,V,'b-');
hold on
gplot(H,V,'r-');
axis off equal square
```

MATLAB code from here

gplot

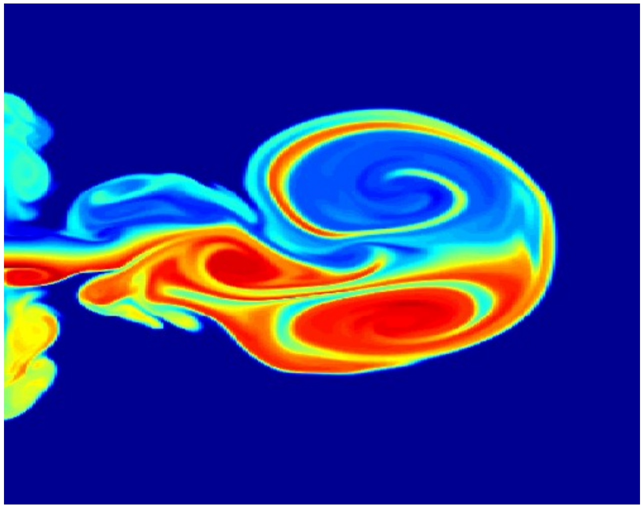
plotting connectivity, or, adjacency matrices



Examples of loading images

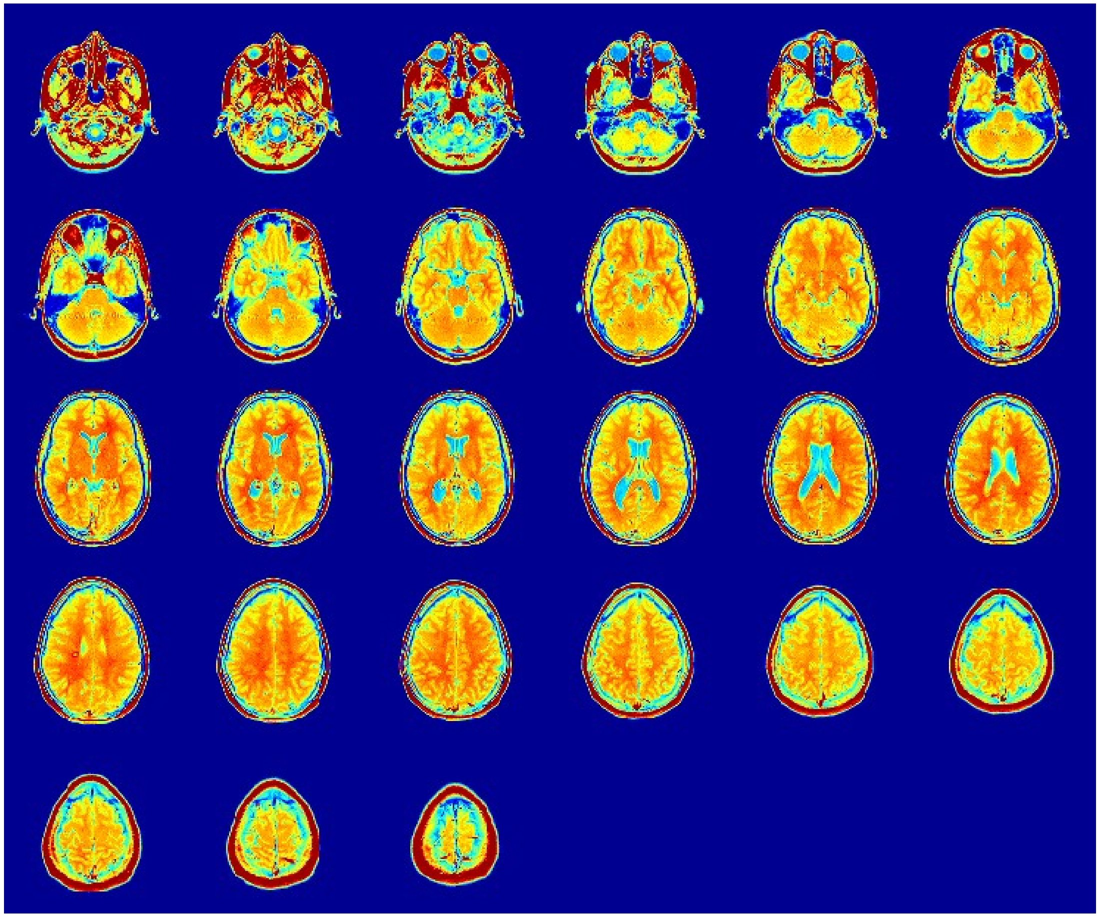
```
load earth;
image(X);
colormap(map);
axis square; axis off
```

```
load flujet;
image(X);
axis off
```



```
load mri;
montage(D,jet);
title('Horizontal Slices');
```

Horizontal Slices



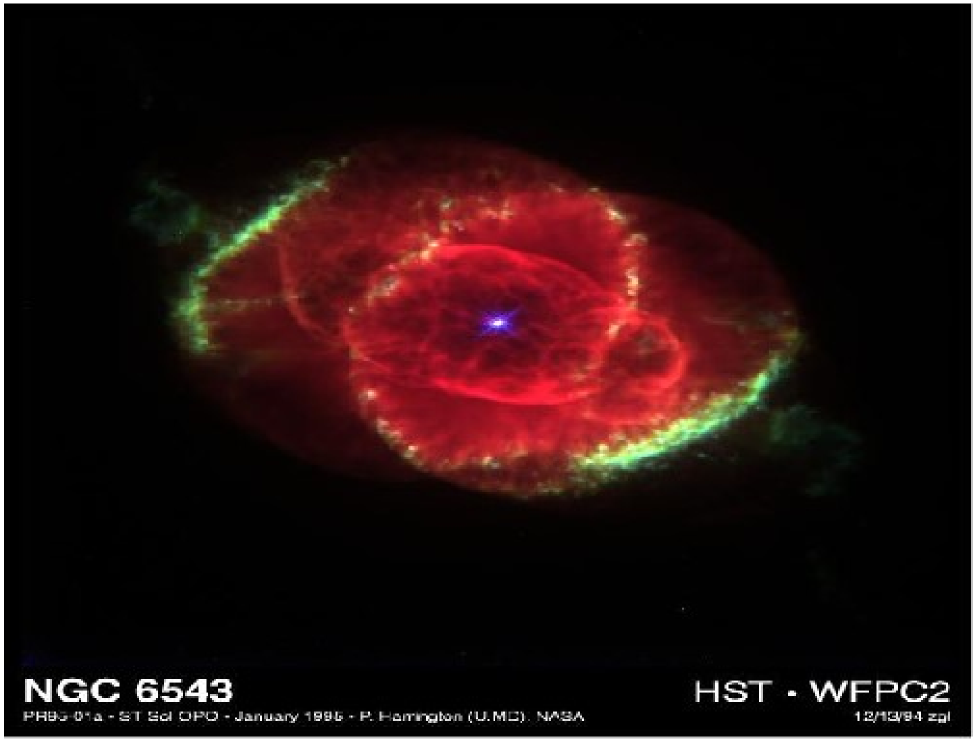
Reading & writing image files

```
A = imread(filename, fmt);
[A, map] = imread(filename, fmt);

imwrite(A,filename,fmt);

imfinfo(filename);


fmt: 'jpg', 'jp2', 'tiff', 'png',
     'gif', 'bmp',  and others
```

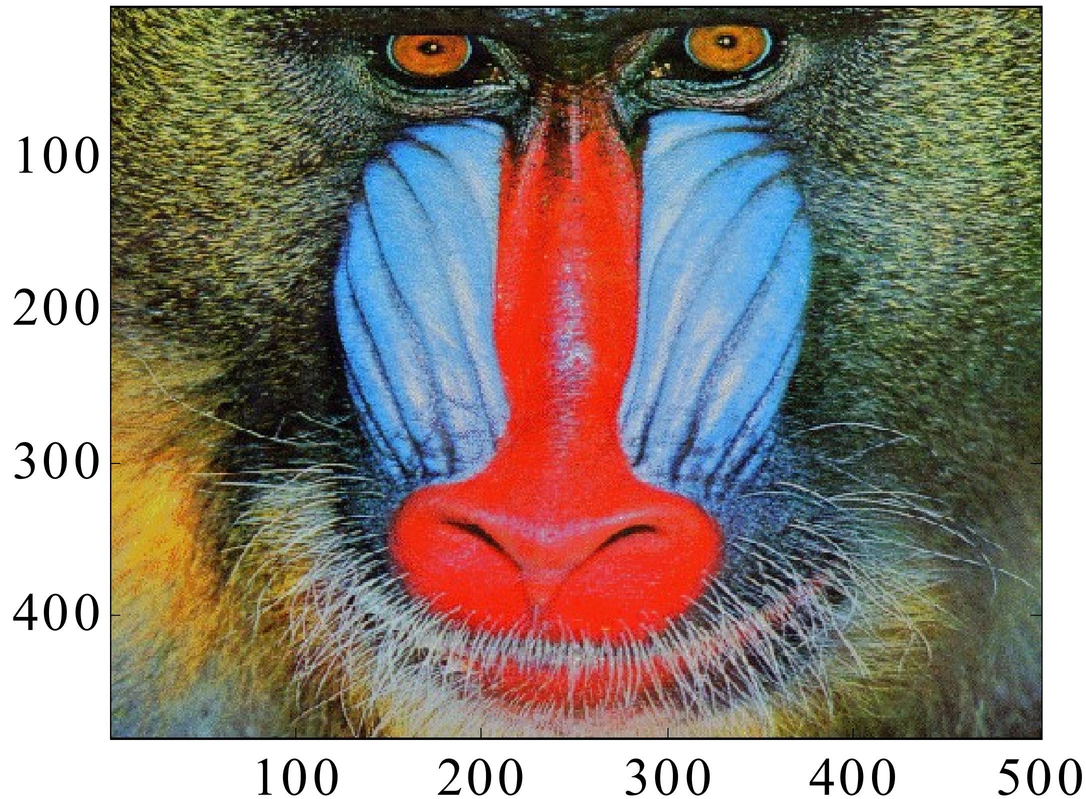
these functions have additional input/output options

```
y = imread('ngc6543a.jpg', 'jpg');
image(y);
title('NGC 6543 Nebula'); axis off;
```

NGC 6543 Nebula



```
load mandrill;     % MATLAB demo image
image(X);          % X,map are part of the
colormap(map);     % saved mandrill.mat file
```



```
s1 = 'http://upload.wikimedia.org/';
s2 = 'wikipedia/commons/d/de/';
s3 = 'St_Louis_night_expblend.jpg';
filename = [s1,s2,s3];

y = imread(filename,'jpg');
image(y); axis off;
```

'http://upload.wikimedia.org/wikipedia/commons/d/de/St_Louis_night_expblend.jpg'

Finally, movies…

Animated plots can be made with the functions
`drawnow, getframe, movie`

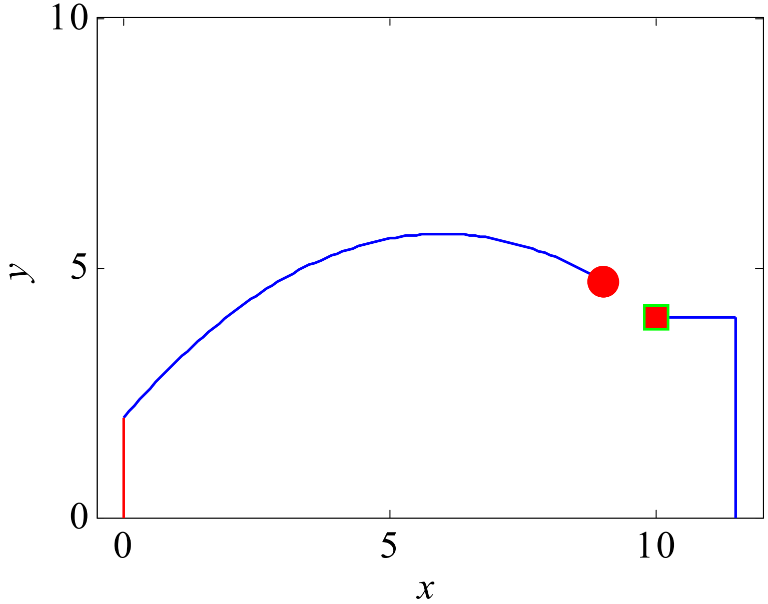
Please study and run the following M-files included
in **`movies.zip`** (placed on sakai) :

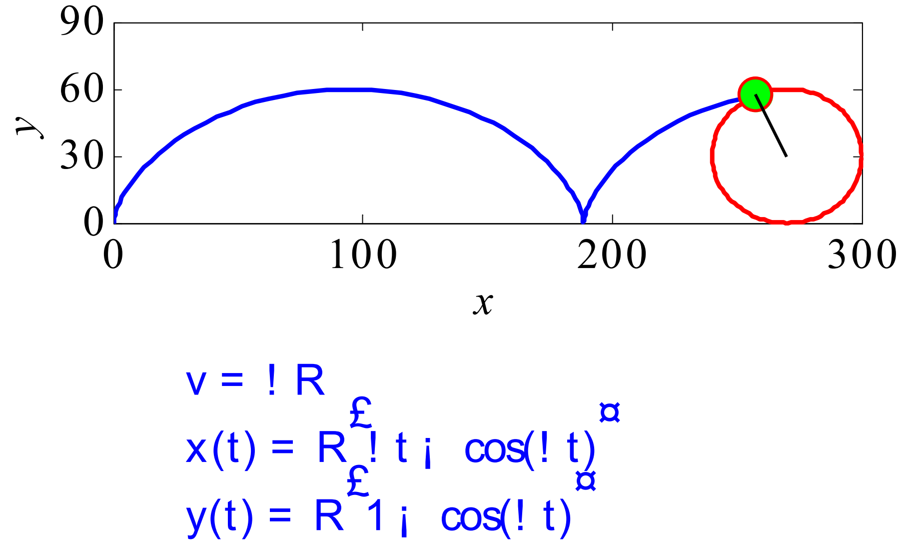
`hoops.m` – throwing the perfect basketball shot

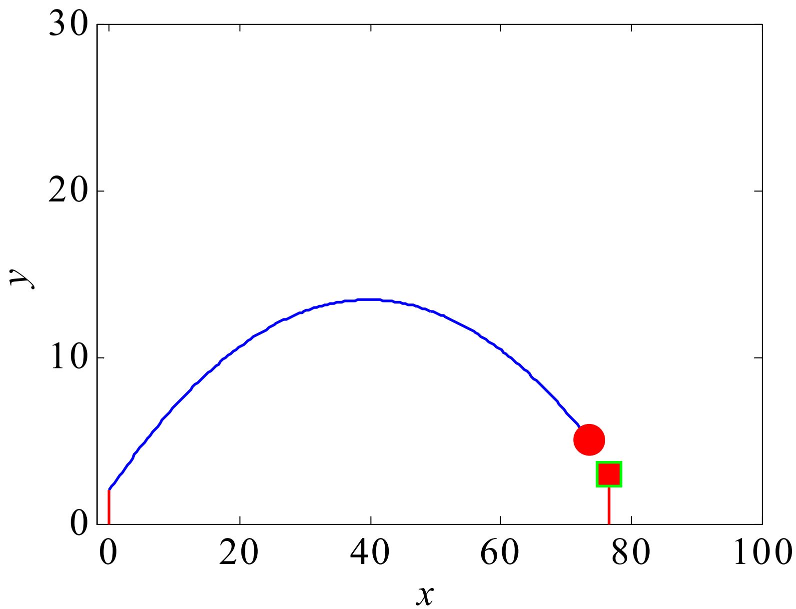
`receiver.m` – moving wide-receiver catching a ball thrown by the QB

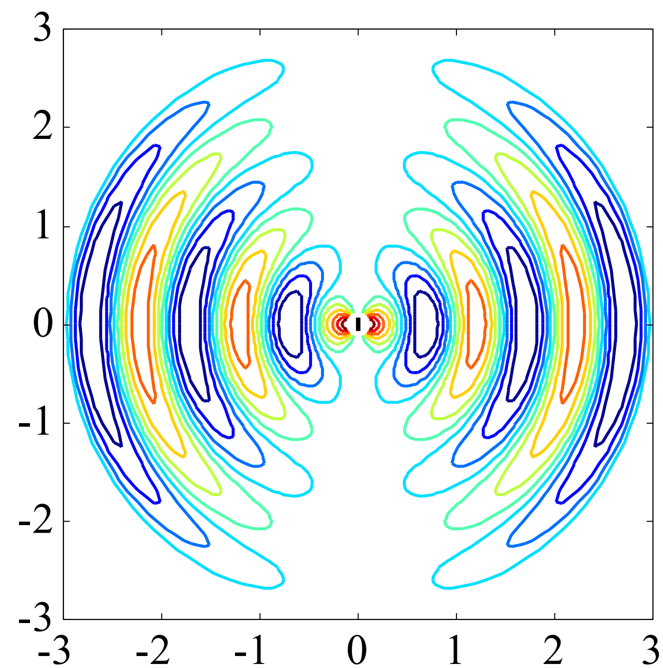
`cycloid.m` – cycloid curve traced by a point on a rolling wheel

`dipmovie.m` – EM wave emitted by a dipole antenna, e.g., your cell phone (see Ref. ch.14)


10
5
0
0
5
10
x
y


90
60
30
0
0
100
200
300
x
y
v = ! R
x(t) = R £! t ¡ cos(! t)¤
y(t) = R £1 ¡ cos(! t)¤


30
20
10
0
0
20
40
60
80
100
x
y


3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3