International University
School of Electrical Engineering

Introduction to Computers for Engineers

Dr. Hien Ta

Lecturely Topics

Lecture 1 - Basics – variables, arrays, matrices
Lecture 2 - Basics – matrices, operators, strings, cells
→ Lecture 3 - Functions & Plotting
Lecture 4 - User-defined Functions
Lecture 5 - Relational & logical operators, if, switch statements
Lecture 6 - For-loops, while-loops
Lecture 7 - Review on Midterm Exam
Lecture 8 - Solving Equations & Equation System (Matrix algebra)
Lecture 9 - Data Fitting & Integral Computation
Lecture 10 - Representing Signal and System
Lecture 11 - Random variables & Wireless System
Lecture 12 - Review on Final Exam

References: H. Moore, *MATLAB for Engineers*, 4/e, Prentice Hall, 2014
G. Recktenwald, *Numerical Methods with MATLAB*, Prentice Hall, 2000
A. Gilat, *MATLAB, An Introduction with Applications*, 4/e, Wiley, 2011

Functions

```
>> help elfun      % elementary functions list
```

Some typical built-in elementary functions are:

```
 sin(x),  cos(x),  tan(x),  cot(x)
asin(x), acos(x), atan(x), acot(x)

 sinh(x),  cosh(x),  tanh(x),  coth(x)
asinh(x), acosh(x), atanh(x), acoth(x)

exp(x), log(x), log10(x), log2(x)

fix(x), floor(x), ceil(x), round(x)

sqrt(x), sign(x), abs(x)

sum(x), prod(x), cumsum(x), cumprod(x)
```

Some more functions:

```
size(x), length(x), class(x)

sinc(x)                          % sin(pi*x)/(pi*x)

max(x), min(x), sort(x)

mean(x), std(x),                 % statistics
median(x), mode(x)

rand, randn,                % random number generators
randi, rng                  % initialize with rng

filter, conv, fft           % DSP functions

clock, date

factorial(n), nchoose(n,k)      % discrete math
```

for a complete list, see Appendix A of your text

Most functions admit scalar or array and matrix input arguments and operate on each element of the array

$$\mathbf{x} = [x_1, x_2, x_3, \ldots]$$

$$f(\mathbf{x}) = [f(x_1), f(x_2), f(x_3), \ldots]$$

```
>> x = [0, pi/4, pi/3, pi/2, pi];

>> sin(x)
ans =
      0    0.7071    0.8660    1.0000    0.0000

>> sin(sym(x))              % use symbolic toolbox
                            % to see exact expressions
ans =
    [ 0,  2^(1/2)/2, 3^(1/2)/2,  1,  0]
```

```
>> x = [2.1, 2.8, -3.1, -3.5, 4.5];

>> y = exp(x)
y =
    8.1662   16.4446    0.0450    0.0302   90.0171

>> z = log(y)        % note log(exp(x)) = x
z =
    2.1000    2.8000   -3.1000   -3.5000    4.5000

>> [fix(x); floor(x); ceil(x); round(x)]
ans =
     2     2    -3    -3     4
     2     2    -4    -4     4
     3     3    -3    -3     5
     2     3    -3    -4     5
```

Example: verify the following geometric-series identity using the function **`sum(x)`**,

$$\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \cdots + \frac{1}{2^N} = 1 - \frac{1}{2^N}$$

summation notation →

$$\sum_{n=1}^{N} \frac{1}{2^n} = 1 - \frac{1}{2^N}$$

```
>> format long;
>> N = 8; n = 1:N;      % n = [1, 2, ..., 8 ]
                        % [1/2^1, 1/2^2, ..., 1/2^8]
>> sum(1./2.^n)         % note the operations ./ and .^
>> 1 - 2^(-N)
ans =
      0.996093750000000
ans =
      0.996093750000000
```

`y = cumsum(x)` – cumulative sum of the elements of x

$$y(1) = x(1)$$

$$y(2) = x(1) + x(2)$$

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

$$\cdots$$

$$y(n) = \sum_{i=1}^{n} x(i) = x(1) + x(2) + \cdots + x(n)$$

```
x = [y(1), diff(y)]      % inverse operation
```

cumsum - Example 1

```
>> N = 8; n = 1:N;          % n is a row vector

>> y = cumsum(1./2.^n);     % y,z should be equal
>> z = 1 - 1./2.^n;

>> fprintf('%d   %10.8f   %10.8f\n',[n; y; z]);
```

```
1   0.50000000   0.50000000
2   0.75000000   0.75000000
3   0.87500000   0.87500000
4   0.93750000   0.93750000
5   0.96875000   0.96875000
6   0.98437500   0.98437500
7   0.99218750   0.99218750
8   0.99609375   0.99609375
```

fprintf operates column-wise on the 3x8 matrix **[n; y; z]**, i.e.,

$$\begin{bmatrix} n_1 & n_2 & n_3 & \cdots \\ y_1 & y_2 & y_3 & \cdots \\ z_1 & z_2 & z_3 & \cdots \end{bmatrix}$$

Random numbers, min, max, mean, std, sort

```
>> seed = 127; rng(seed);
>> x = randn(5,3)
x =
    0.0294   -1.0928    1.6686
   -1.5732   -0.1697   -0.4750
   -1.1899    0.5751   -0.7604
    1.8115    0.6548   -1.1189
    0.0426   -0.0969    0.1698

>> min(x), max(x), mean(x), std(x)
ans =
   -1.5732   -1.0928   -1.1189
ans =
    1.8115    0.6548    1.6686
ans =
   -0.1759   -0.0259   -0.1032
ans =
    1.3248    0.7051    1.0972
```

initialize generator, 5x3 matrix of zero-mean, unit-variance, gaussian, random numbers

```
>> help rng
>> help rand
>> help randn
>> help randi
```

computed column-wise

MATLAB is column-dominant

```
x =
    0.0294   -1.0928    1.6686
   -1.5732   -0.1697   -0.4750
   -1.1899    0.5751   -0.7604
    1.8115    0.6548   -1.1189
    0.0426   -0.0969    0.1698
```

i=2

min,max,sort
act column-wise
on matrix inputs

```
>> [m,i] = min(x), min(min(x))
m =
   -1.5732   -1.0928   -1.1189
i =
     2     1     4
ans =
   -1.5732
```

minimum of each column,
index within each column,
overall minimum

```
>> sort(x)
ans =
   -1.5732   -1.0928   -1.1189
   -1.1899   -0.1697   -0.7604
    0.0294   -0.0969   -0.4750
    0.0426    0.5751    0.1698
    1.8115    0.6548    1.6686
```

sort each column in
ascending order

sort(x,'ascend')
sort(x,'descend')

```
x =
    0.0294   -1.0928    1.6686
   -1.5732   -0.1697   -0.4750
   -1.1899    0.5751   -0.7604
    1.8115    0.6548   -1.1189
    0.0426   -0.0969    0.1698

>> [x2,i2] = sort(x(:,2),'descend');

>>    [x2,i2]
    0.6548    4
    0.5751    3
   -0.0969    5
   -0.1697    2
   -1.0928    1

>> x_sort = x(i2,:)      % sort x relative to column 2

    1.8115    0.6548   -1.1189
   -1.1899    0.5751   -0.7604
    0.0426   -0.0969    0.1698
   -1.5732   -0.1697   -0.4750
    0.0294   -1.0928    1.6686
```

sort column 2 only in descending order
x2 = sorted column, **i2** = sorting index

```
% using sortrows:

  sortrows(x,-2)
```

Make up your own functions using three methods:

1. function-handle, @(x)
2. inline
3. M-file

example: $f(x) = e^{-0.5x}\sin(5x)$

```
>> f = @(x) exp(-0.5*x).*sin(5*x);

>> g = inline('exp(-0.5*x).*sin(5*x)');

% edit & save file h.m containing the lines:
function y = h(x)
y = exp(-0.5*x).*sin(5*x);
```

.* allows vector or matrix inputs x

How to include parameters in functions

example: $f(x) = e^{-ax}\sin(bx)$

```
% method 1: define a,b first, then define f

a = 0.5; b = 5;
f = @(x) exp(-a*x).*sin(b*x);


% method 2: pass parameters as arguments to f

f = @(x,a,b) exp(-a*x).*sin(b*x);

% this defines the function f(x,a,b)
% so that f(x, 0.5, 5) would be equivalent to
% the f(x) defined in method 1.
```

9. Basic Plotting

MATLAB has extensive facilities for the plotting of curves and surfaces, and visualization. We will be discussing these in detail later on.

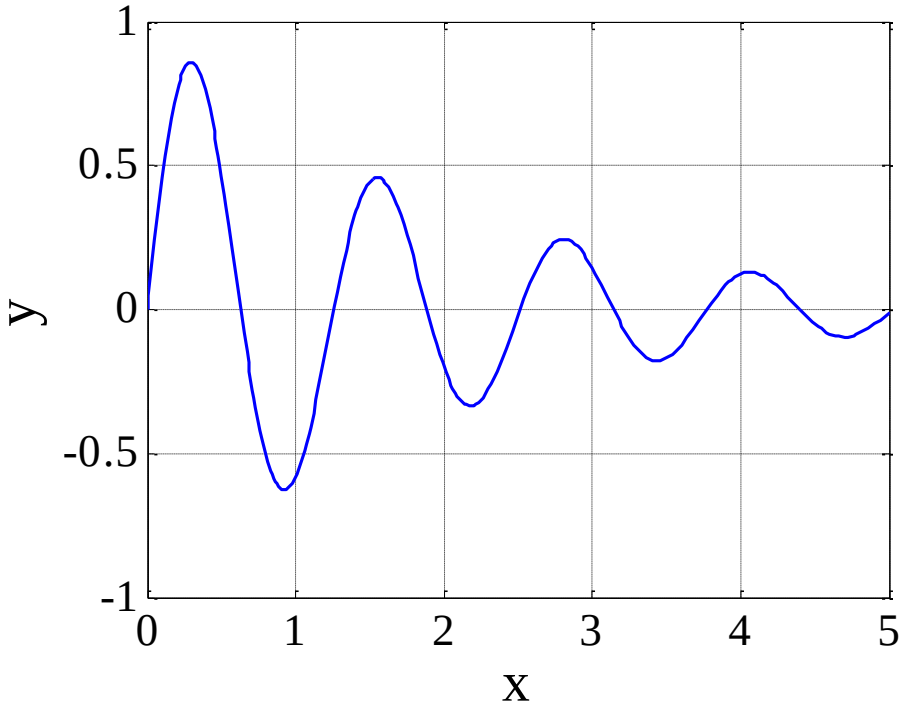
Basic 2D plots of functions and (x,y) pairs can be done with the functions:

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:

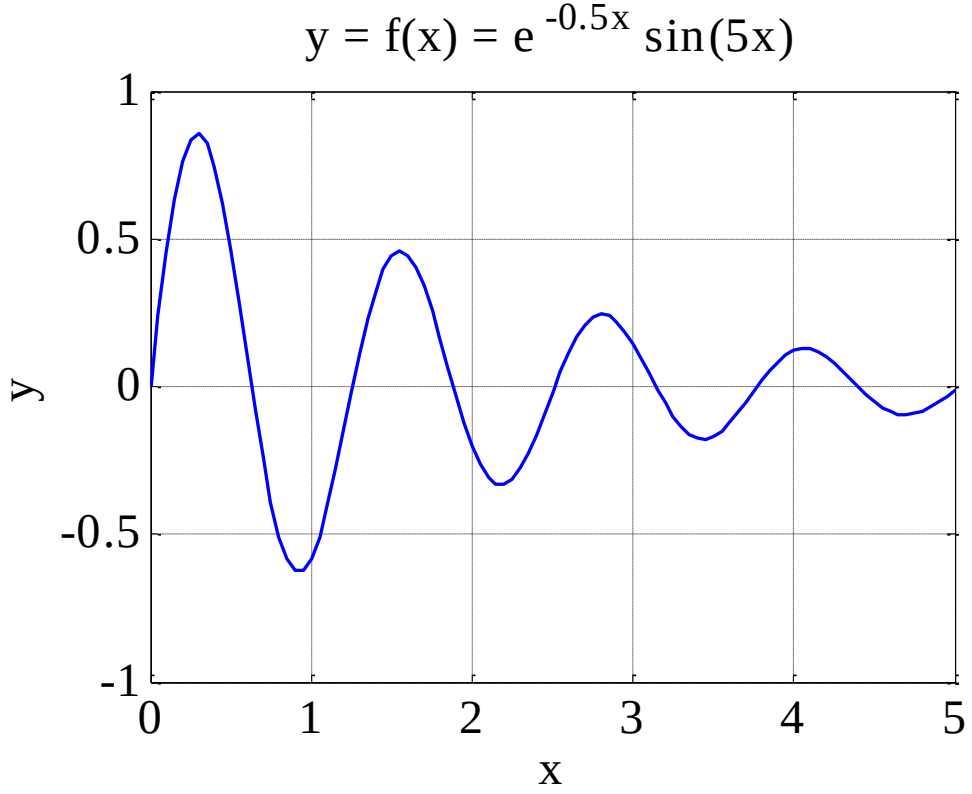
```
>> f = @(x) exp(-0.5*x).*sin(5*x);
>> fplot(f,[0,5]);          % plot over interval [0,5]
```



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.

using the plot function

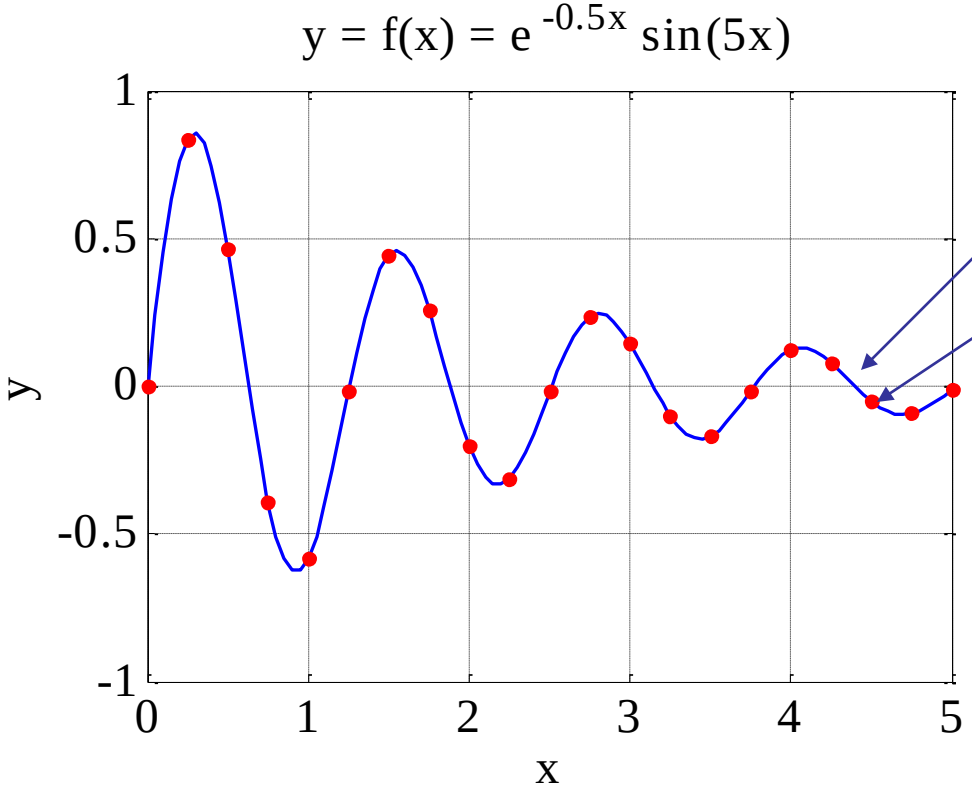
```
>> x = linspace(0,5,101);
>> y = f(x);
>> plot(x,y,'b-');                    % blue-solid line
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x} sin(5x)');
```



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)');
```



(x,y) plotted as blue-solid line

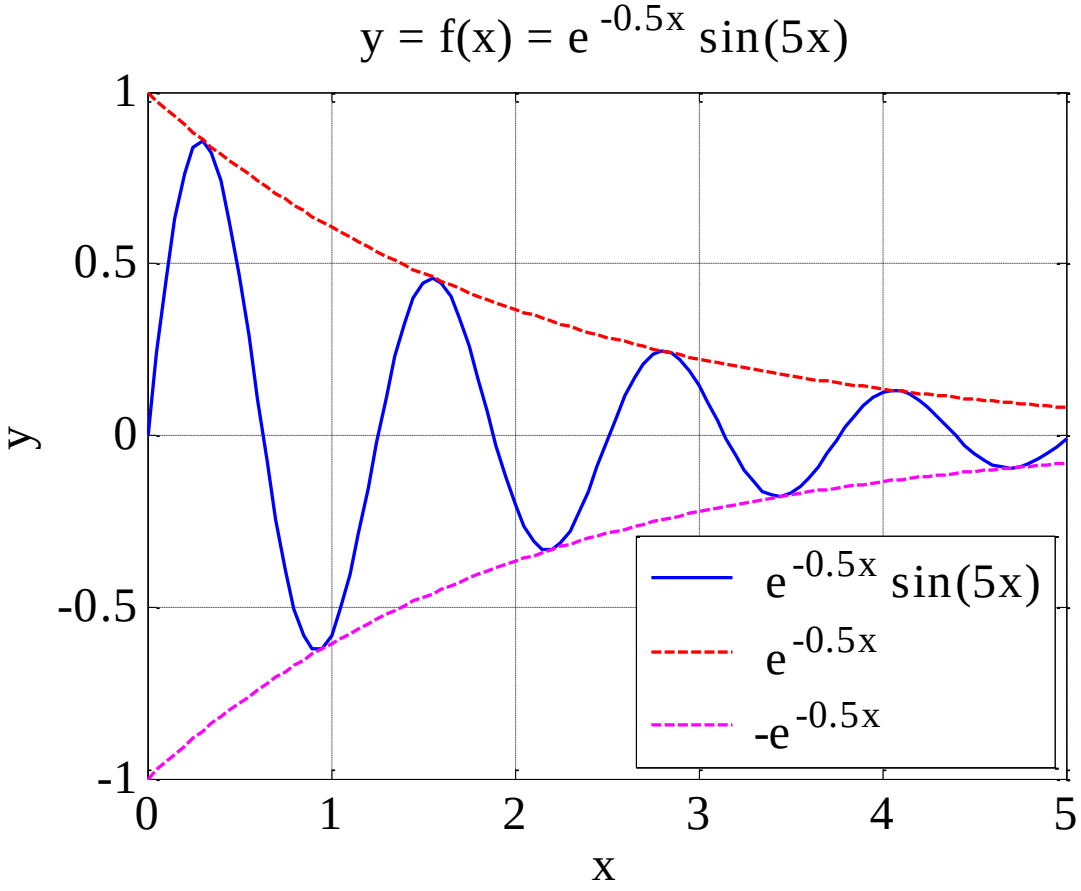
(x5,y5) pairs plotted as red dots

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

```
>> e = exp(-0.5*x);                    % envelope of f(x)
>> plot(x,y,'b-', x,e,'r--', x,-e,'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

10. Function Maxima and Minima

Engineers always like to optimize their designs by finding the best possible solutions. This usually amounts to minimizing or maximizing some function of the design parameters.

Suppose a function f(x) has a minimum (or maximum) within an interval [a,b], or, $a \le x \le b$. The following three methods can be used to find it:

1. Graphical method using the function **`min`** (or **`max`**)
2. Using the built-in function **`fminbnd`**
3. Using the function **`fzero`**, (requires the derivative of f(x))

(use **`fminsearch`** for multivariable functions)

MATLAB implementation of the three methods

```
f = @(x) ...          % define your function here
                      % f(x) must admit vector inputs
                      % and return vector outputs
```

1.

```
x = linspace(a,b,N);          % larger N works better
[fmin,imin] = min(f(x));      % fmin = minimum value
xmin = x(imin);               % x-location of minimum
plot(x,f(x), xmin,fmin,'o');      % display it
```

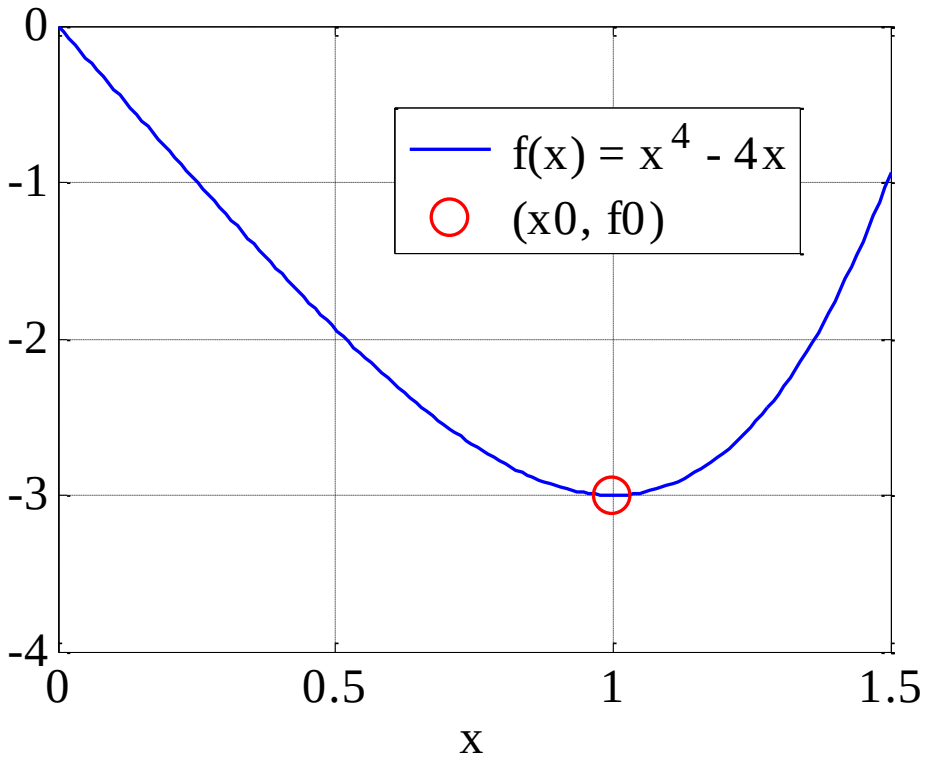
2.

```
[xmin,fmin] = fminbnd(f,a,b);   % search in [a,b]
```

3.

```
F = @(x) ...                  % define derivative of f(x)
                              % or use symbolic toolbox
xmin = fzero(F,x0);           % search near x0
fmin = f(xmin);               % minimum value of f(x)
```

```
f = @(x) x.^4 - 4*x;

x = linspace(0, 1.5, 151);

[f0,i0] = min(f(x)); x0 = x(i0);

plot(x,f(x),'b-', x0,f0,'ro');
xlabel('x'); grid;
legend('f(x)=x^4-4x', '(x0,f0)');
```



Example: finding the minimum of a curve using the function **min**

f0 is minimum of the array **y=f(x)**

i0 is the index of array at its min, i.e., **f0=y(i0)**

x0 is value of **x** at the minimum of **y**

exact values are:

x0 = 1
f0 = -3

finding the minimum of f(x) using the function **fminbnd**

both **fminbnd** and **fzero** admit function handles as inputs

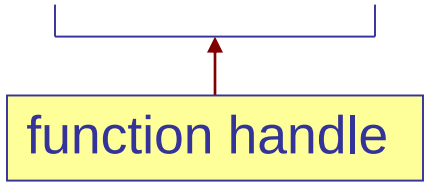
```
f = @(x) x.^4 - 4*x;                  % find minimum of f(x)
[x1,f1] = fminbnd(f,0,1.5);           % in the interval[0,1.5]
```

finding the minimum of f(x) using the function **fzero,** requires derivative F(x) = df(x)/dx

```
F = @(x) 4*x.^3 - 4;                  % derivative of f(x)
x2 = fzero(F, 0.5); f2 = f(x2);

[x0,x1,x2; f0,f1,f2]                  % compare the three methods

ans =
    0.9966    1.0000    1.0000
   -2.9999   -3.0000   -3.0000
```

How to find the maximum of a function f(x) using **fminbnd**

```
f = @(x) ...          % define your function here
                      % f(x) must admit vector inputs
                      % and return vector outputs


[xmax,fmin] = fminbnd(@(x) -f(x), a,b);
fmax = -fmin;
```




```
% fminbnd(-f,a,b) is not allowed
% alternatively, first define negative of f(x)

g = @(x) -f(x);       % i.e., g(x) = -f(x)

[xmax,fmin] = fminbnd(g,a,b);
fmax = -fmin;
```

Plotting

plot, line styles, colors, markers, multiple graphs
adding text, legends, plot editor
axis settings, subplots
fplot, ezplot, loglog, semilogy, plotyy

scatter, stem, stairs
bar graphs, histograms, pie charts, polar plots

3D plotting functions, meshgrid
plot3, stem3, bar3, pie3, comet3
contour, contourf
mesh, meshc, meshz, waterfall, area plots
surf, surfc, colormap, colorbar, shading
surfaces of revolution

convhull, voronoi, spy, gplot
animated plots, drawnow, getframe, movie

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

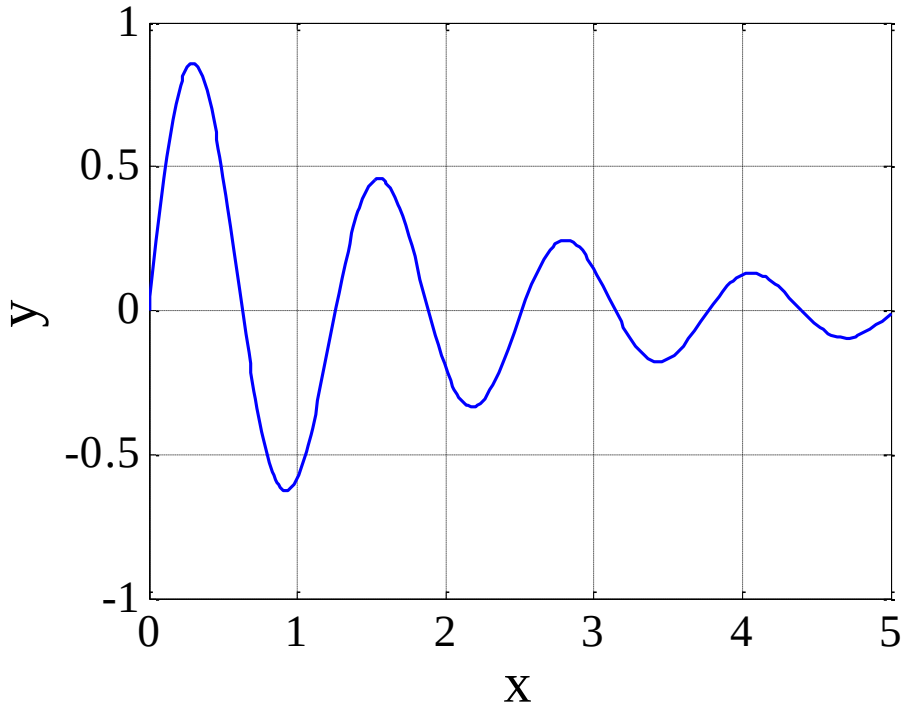
Basic 2D plots of functions and (x,y) pairs can be done with the functions:

`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:

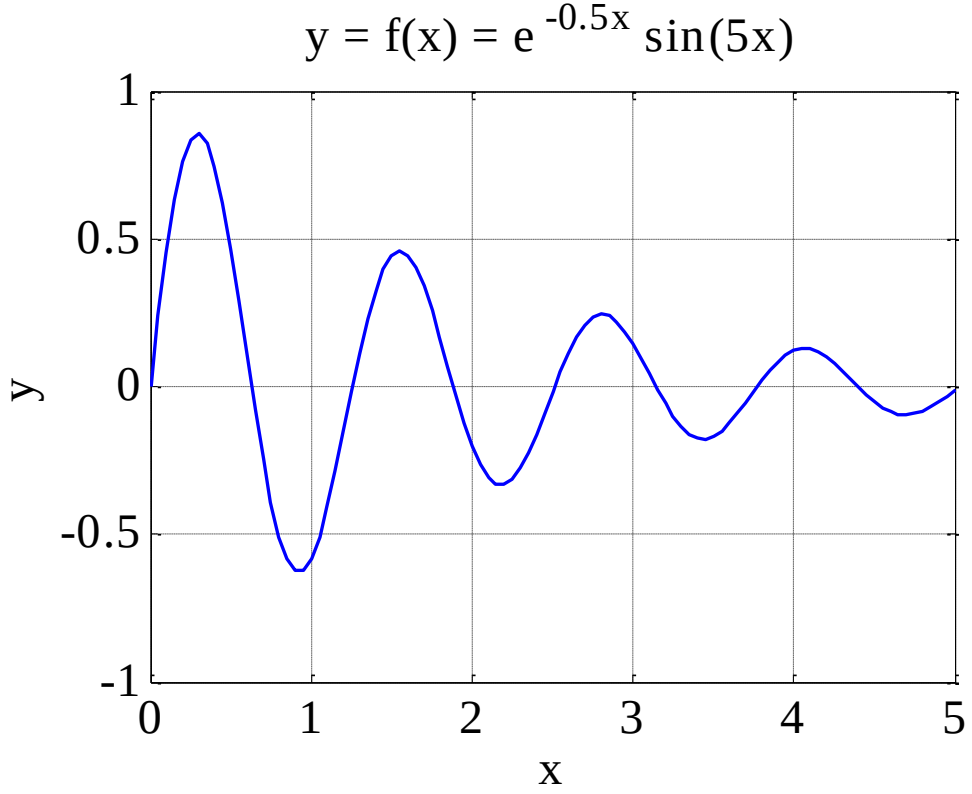
```
>> f = @(x) exp(-0.5*x).*sin(5*x);
>> fplot(f,[0,5]);          % plot over interval [0,5]
```



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

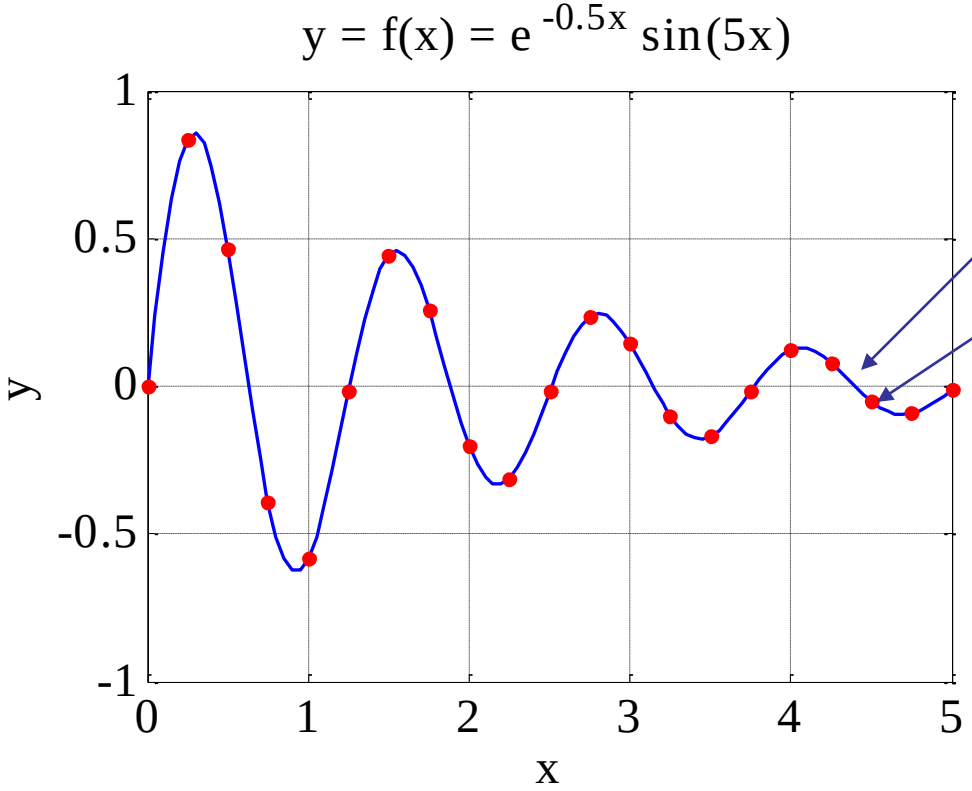
```
>> x = linspace(0,5,101);
>> y = f(x);
>> plot(x,y,'b-');                    % blue-solid line
>> xlabel('x'); ylabel('y'); grid;
>> title('f(x) = e^{-0.5x} sin(5x)');
```



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)');
```

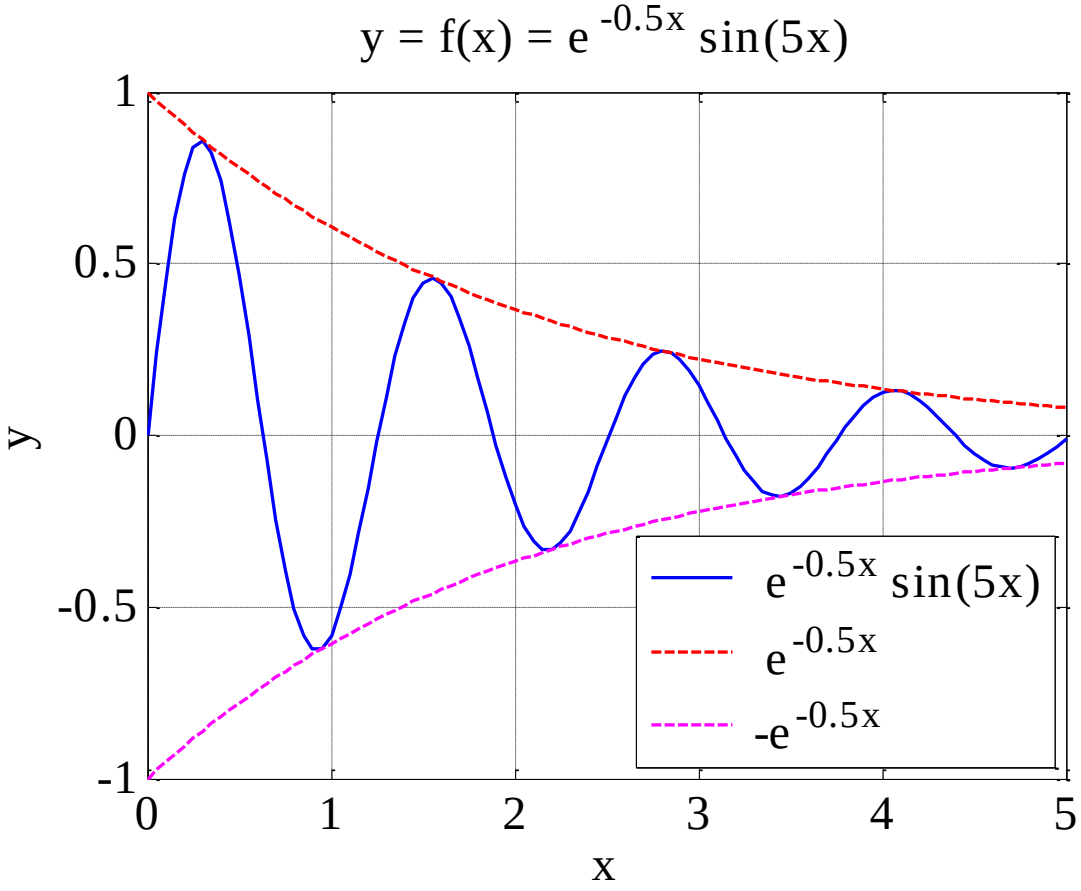


(x,y) plotted as blue-solid line

(x5,y5) pairs plotted as red dots

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

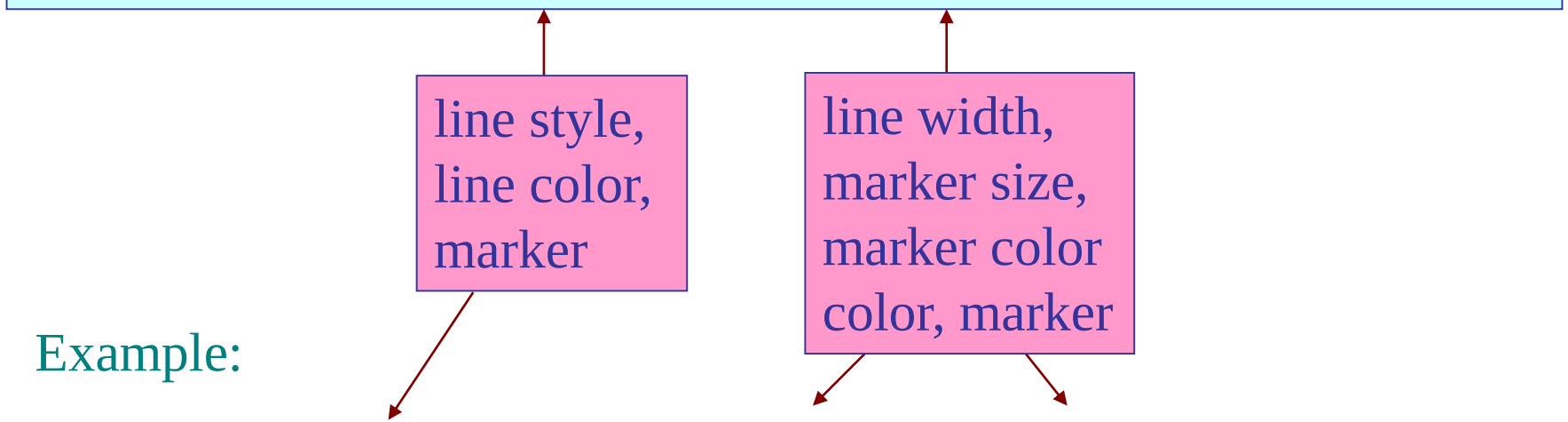
ellipsis continues to next line

plotting multiple curves and adding legends

legends can also be inserted with plot editor

plot

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



Example:

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

default values:
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		

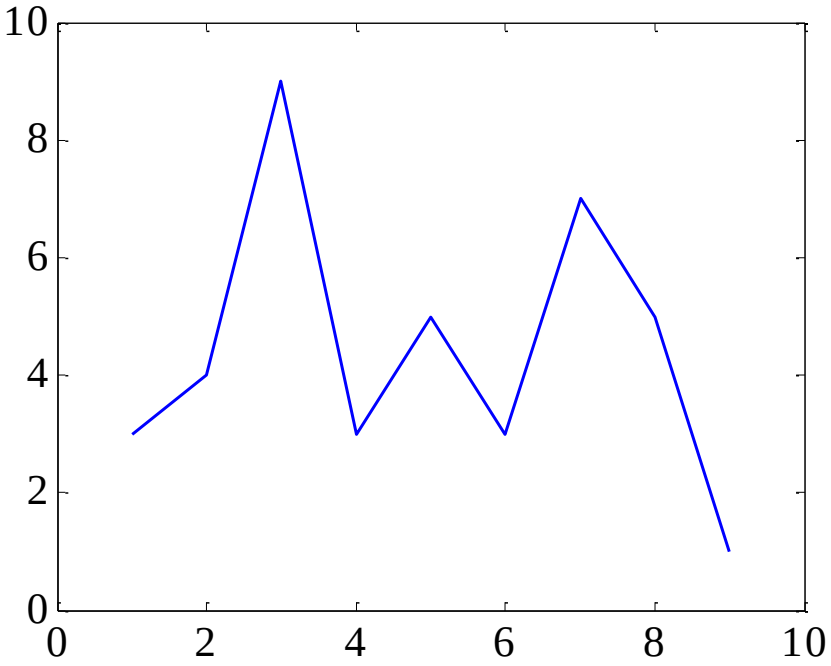
property name

linewidth
markersize
markeredgecolor
markerfacecolor

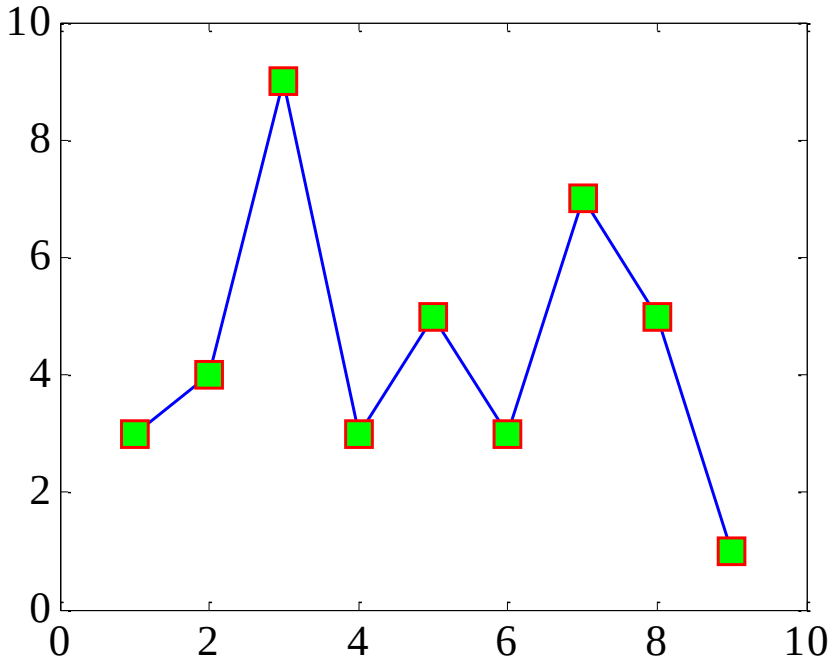
```
x = [1  2  3  4  5  6  7  8  9];
y = [3  4  9  3  5  3  7  5  1];
```

line styles & markers

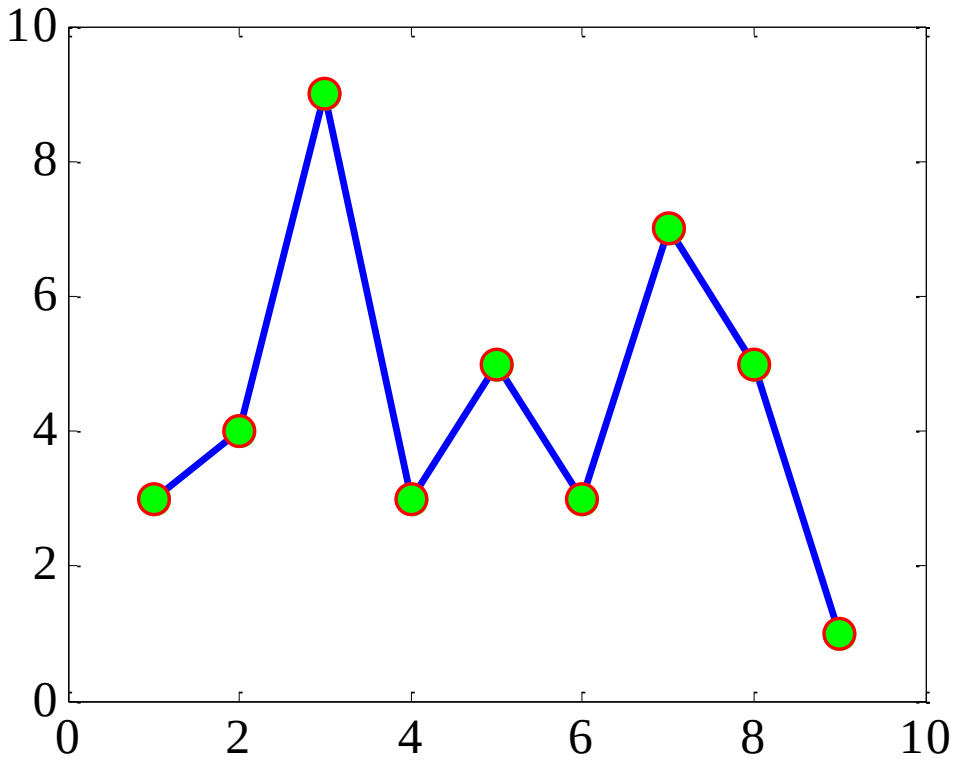
```
plot(x,y,'b-');
```




```
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

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

equivalent

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

exception

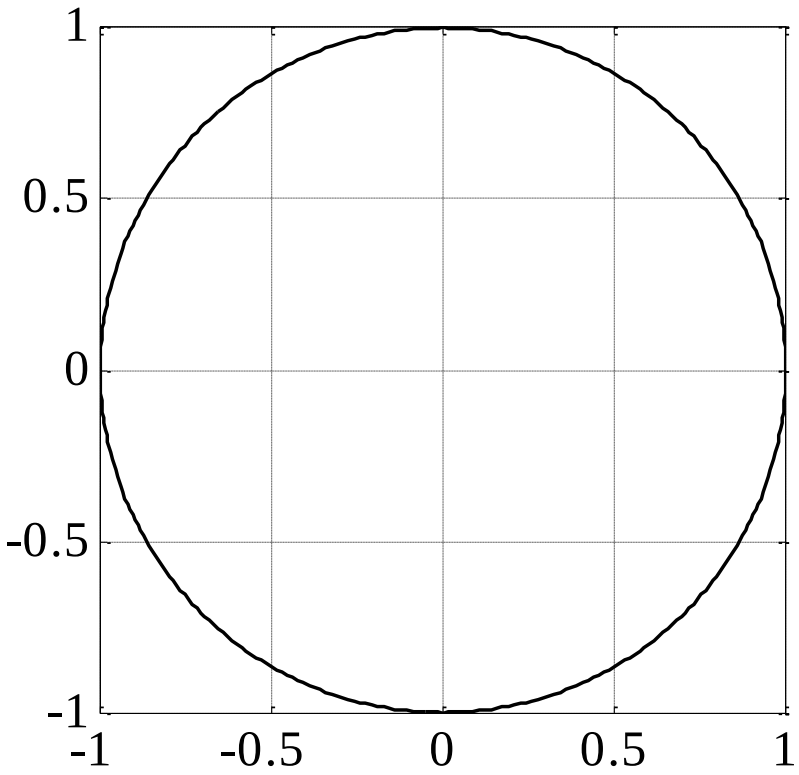
How to plot a circle

```
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');
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

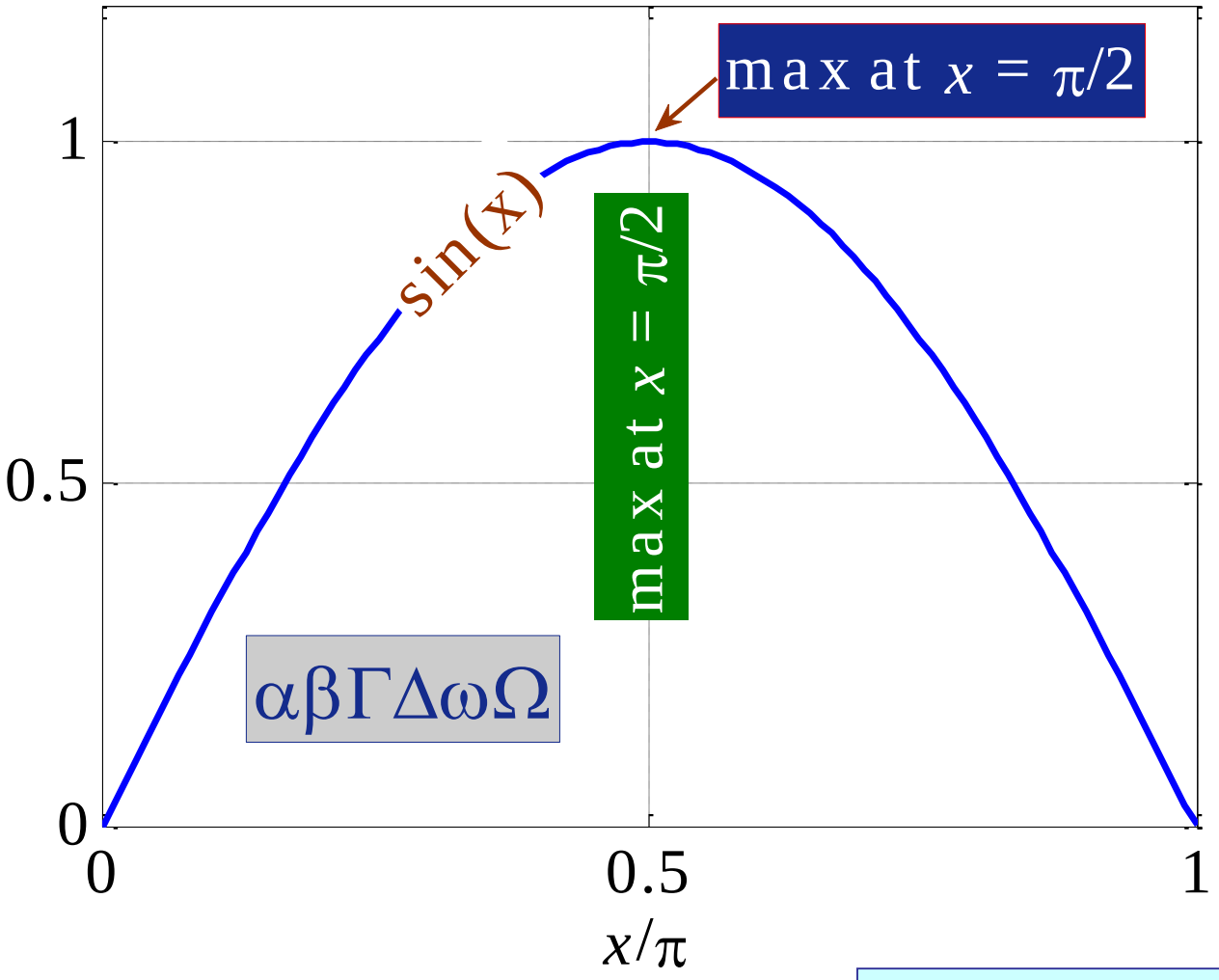
properties can also be set with the plot editor

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);

xaxis(0,1, 0:0.5:1); yaxis(0,1.2, 0:0.5:1);
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
```

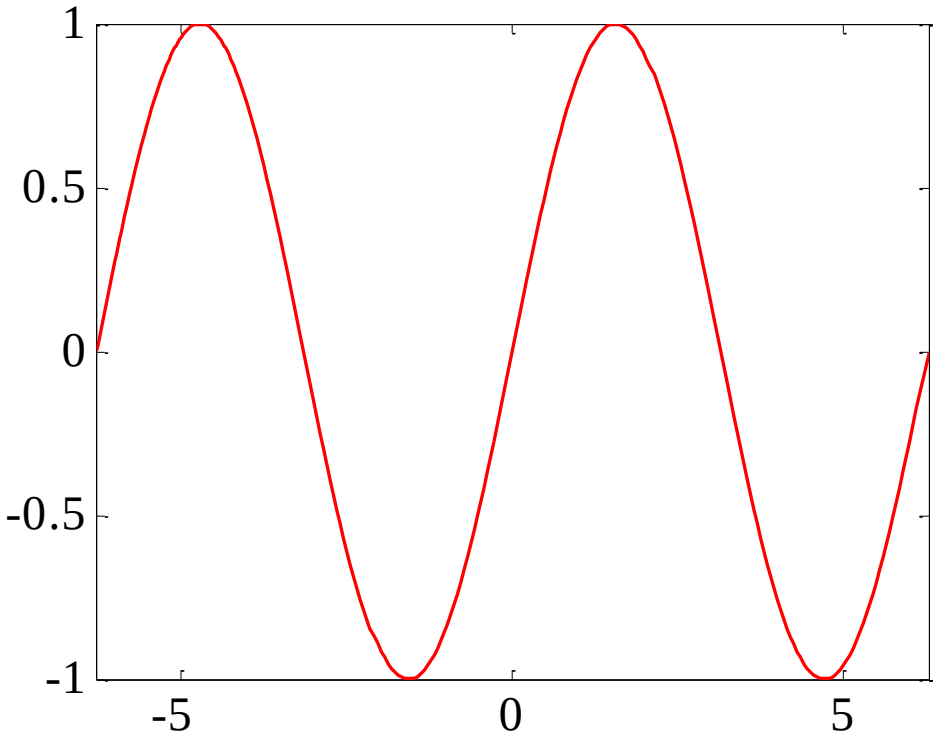
combined into the **xaxis** function in course-functions folder on sakai

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);
```

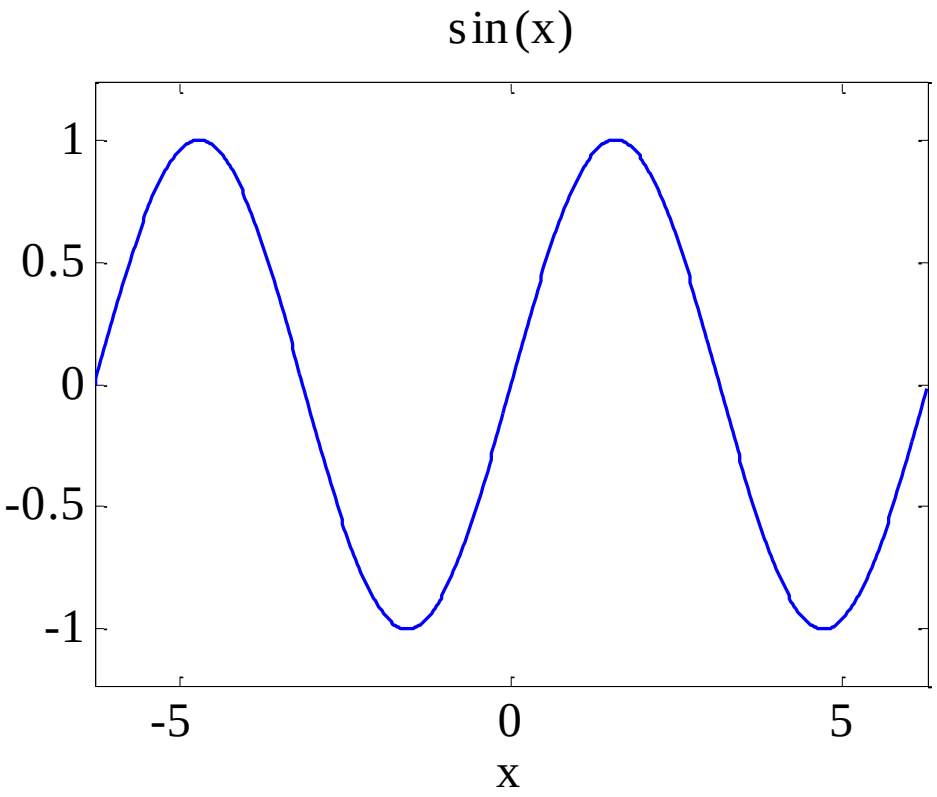


linestyles & colors can be changed from the figure window, or

```
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);
```

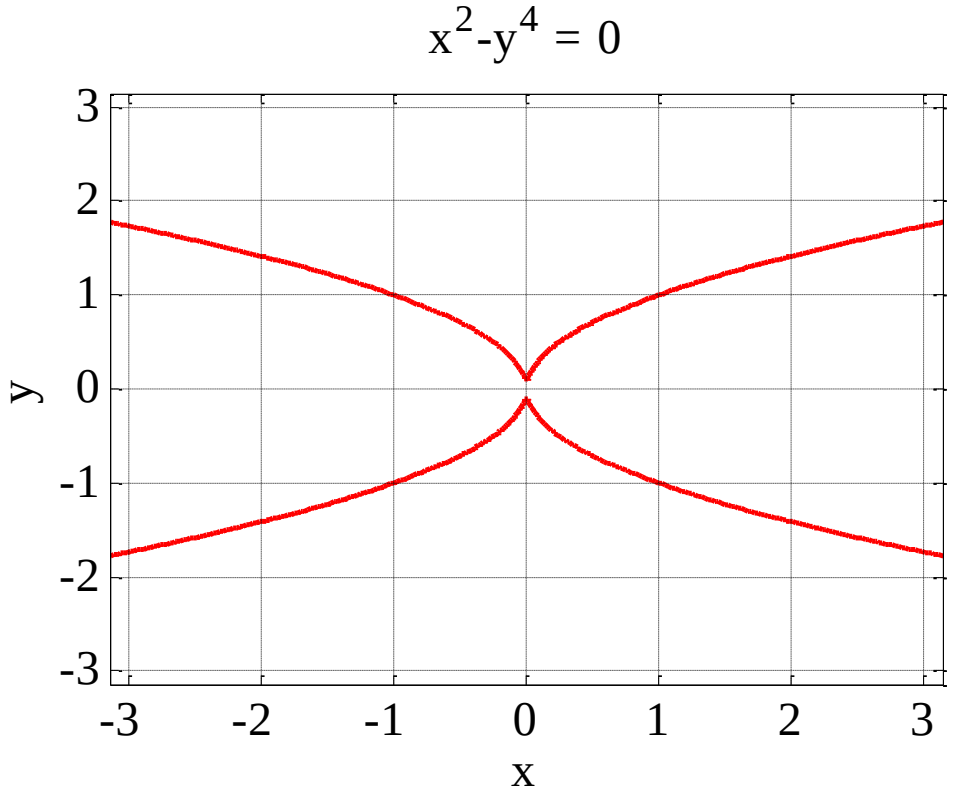


linestyles & colors can be changed from the figure window

fplot, ezplot

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

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

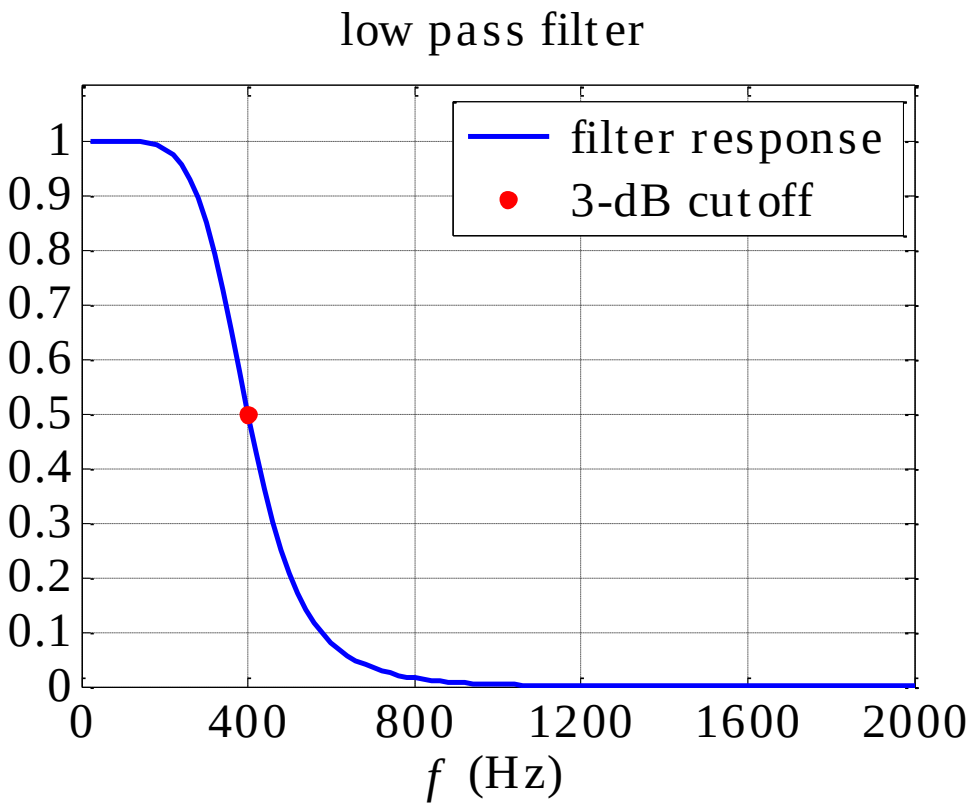
ezplot can plot functions defined implicitly, i.e., f(x,y) = 0



loglog plots

Butterworth lowpass audio filter

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



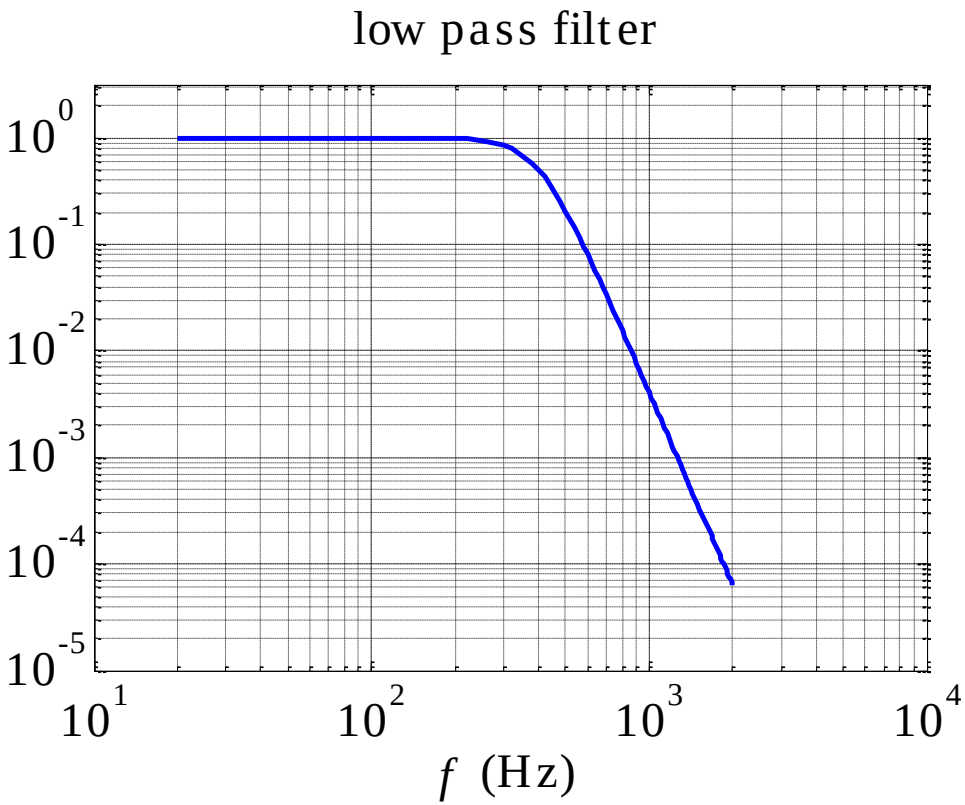
N = 3
f0 = 400 Hz

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');
```

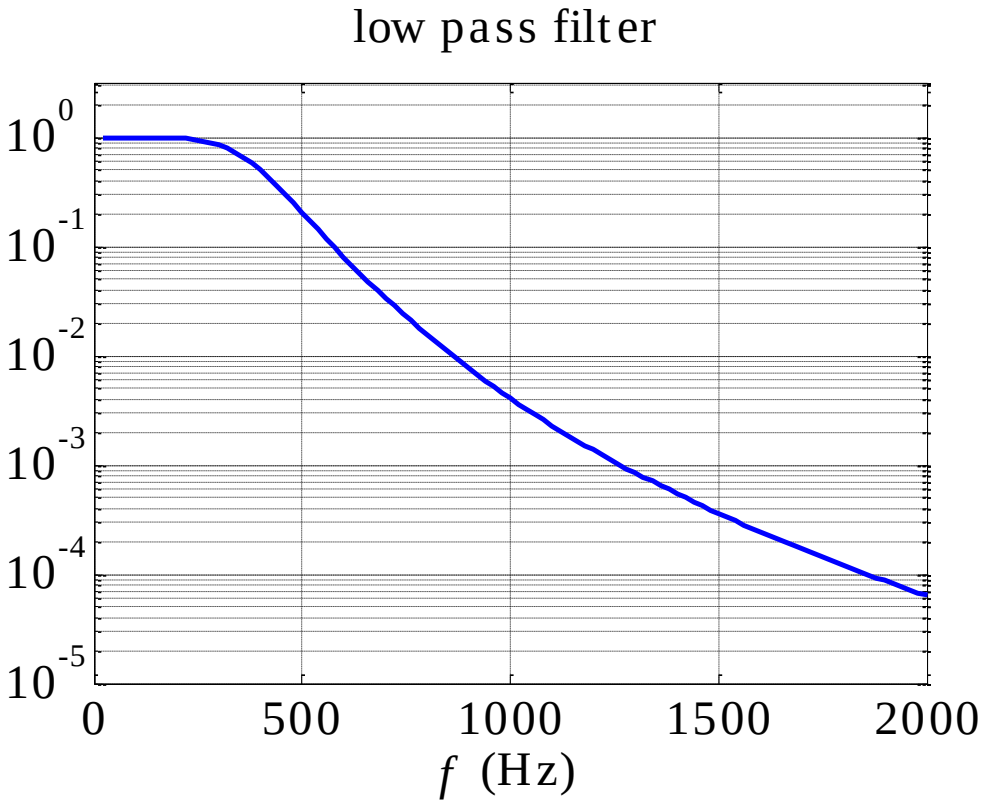
loglog

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

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



semilogy

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

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

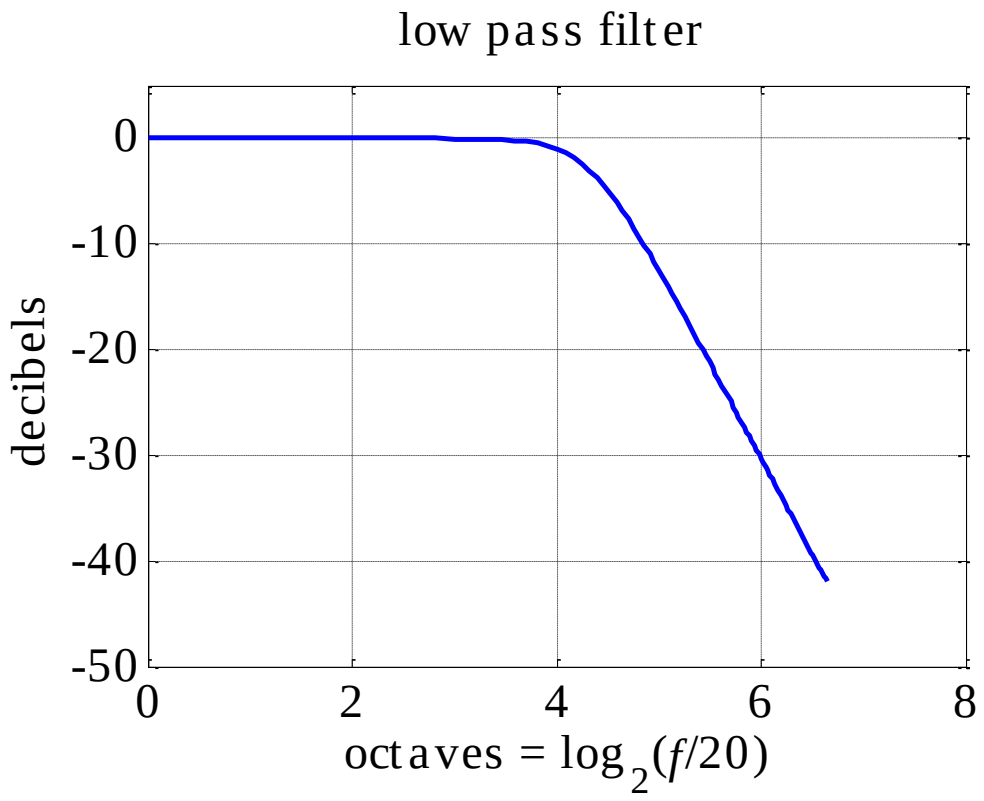


```
plot(log2(f/20), 10*log10(H2),'b');

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

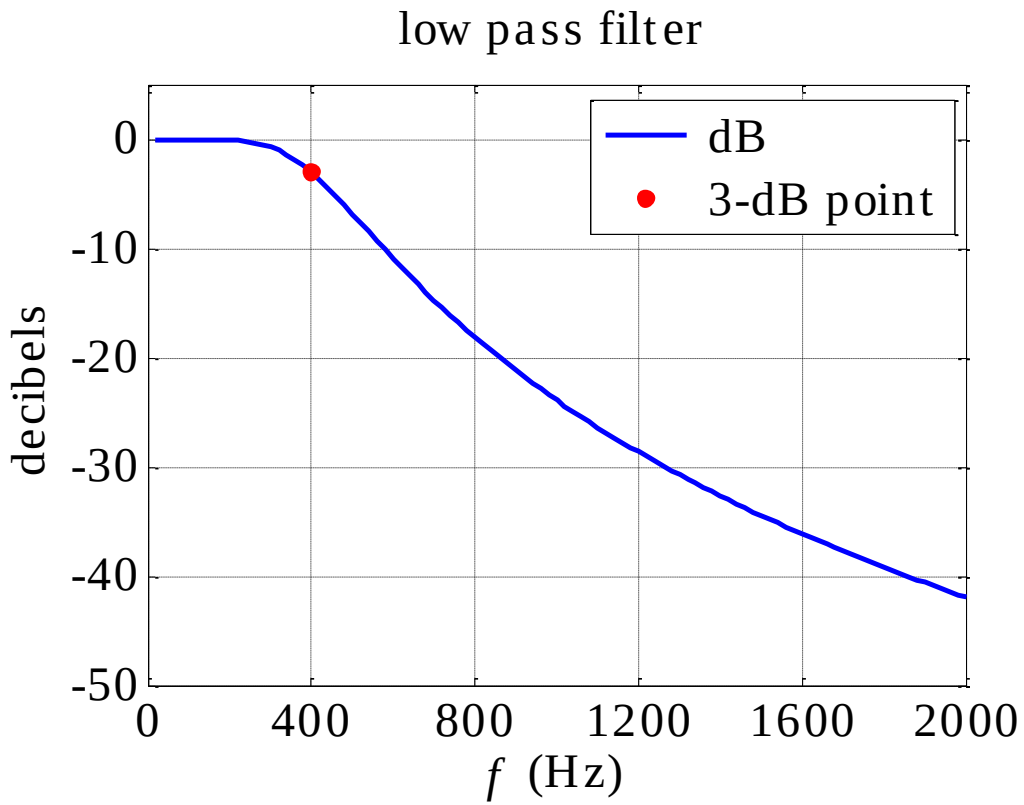
filter gain in dB

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



```
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



subplots

general syntax:

```
subplot(n,m,p);
```

n x m = box pattern

p = counting figures across rows

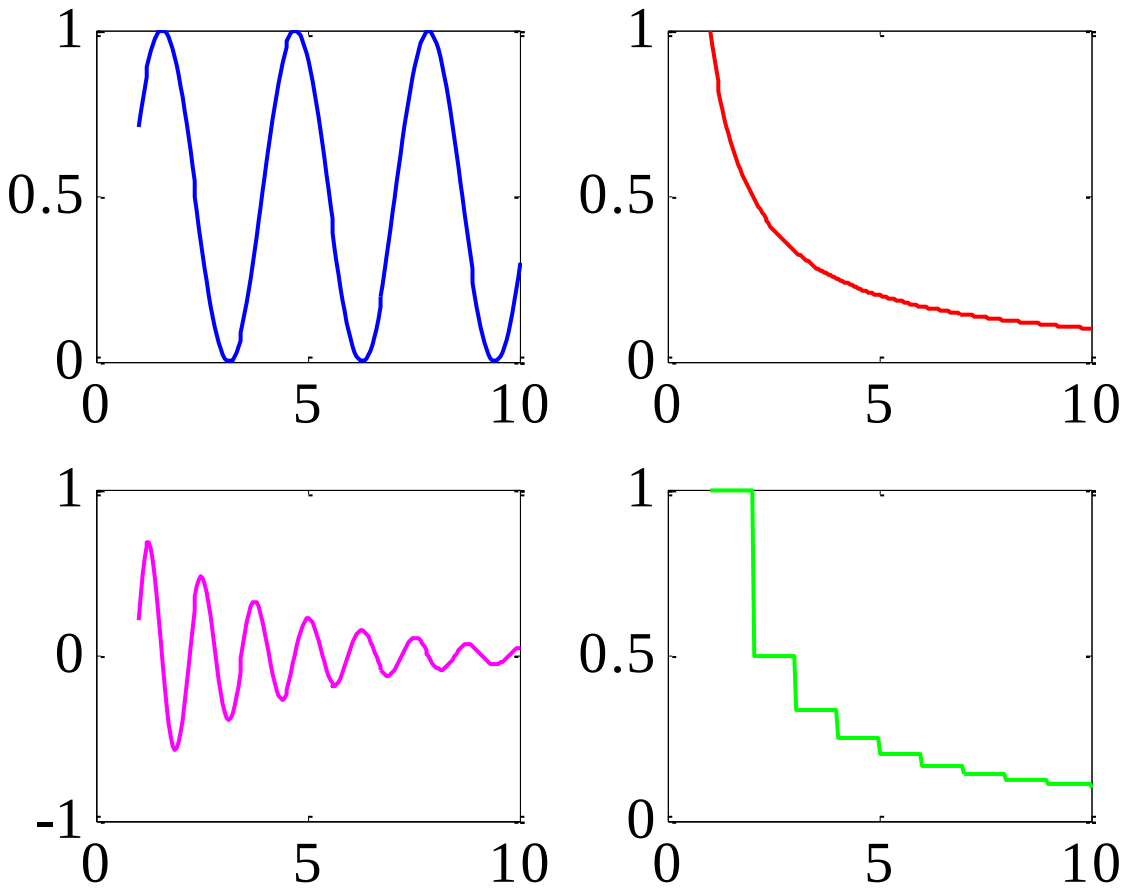
3 x 4 pattern

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

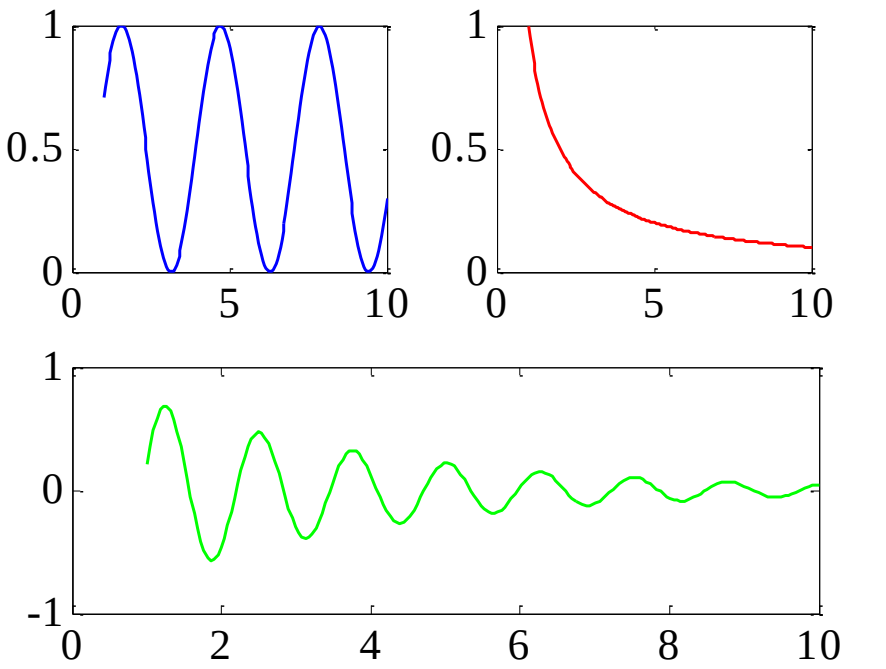
```
subplot(3,4,1)
subplot(3,4,2)
etc.
```

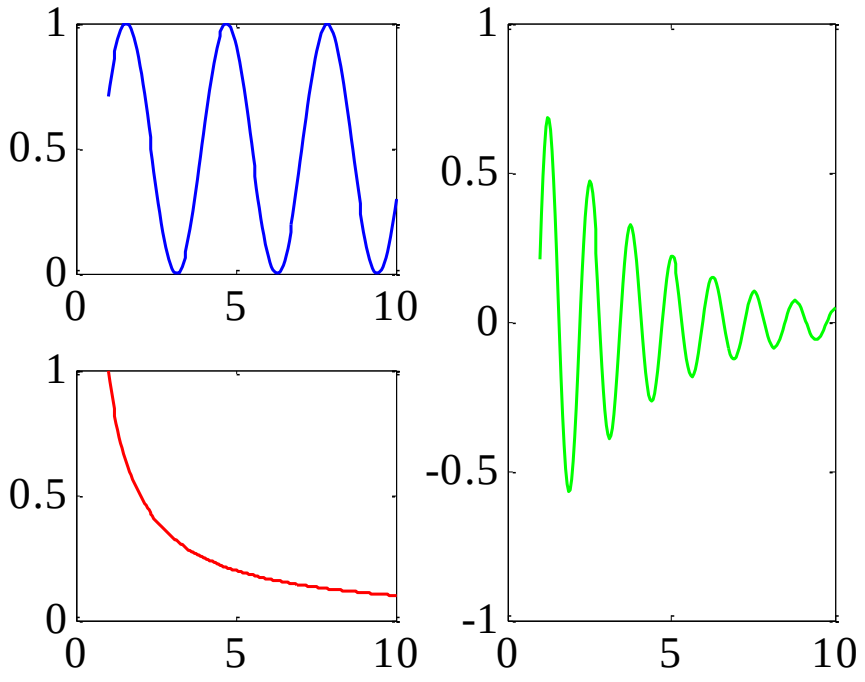
```
x = linspace(1,10,200);

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');
```

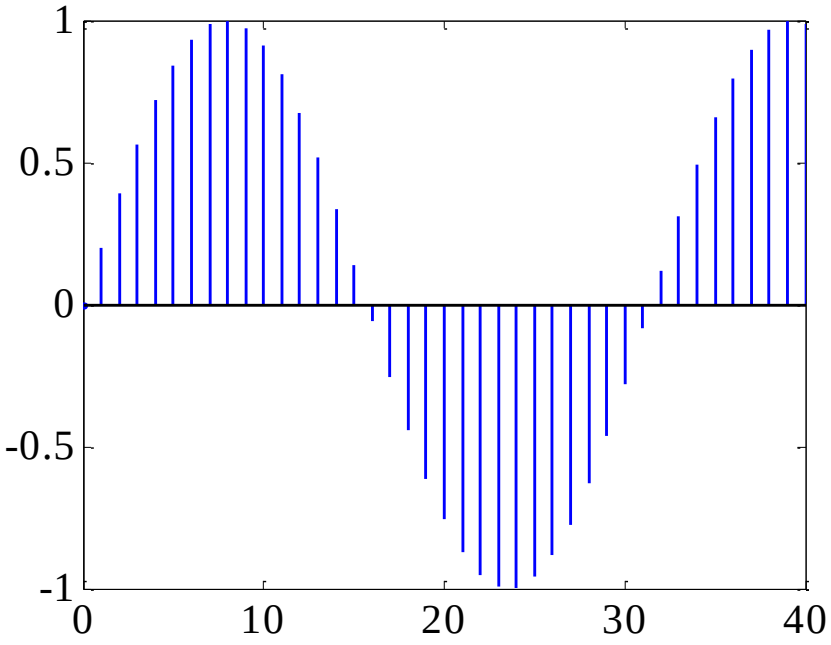


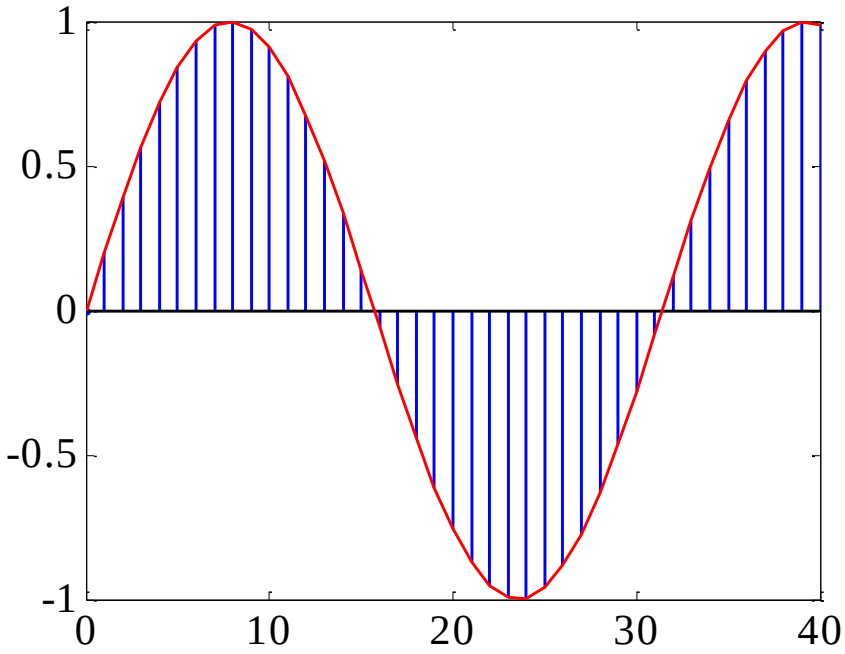



```
subplot(2,2,1); plot(x,y1,'b');
subplot(2,2,3); plot(x,y2,'r');
subplot(1,2,2); plot(x,y3,'g');
```

stem plots

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

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



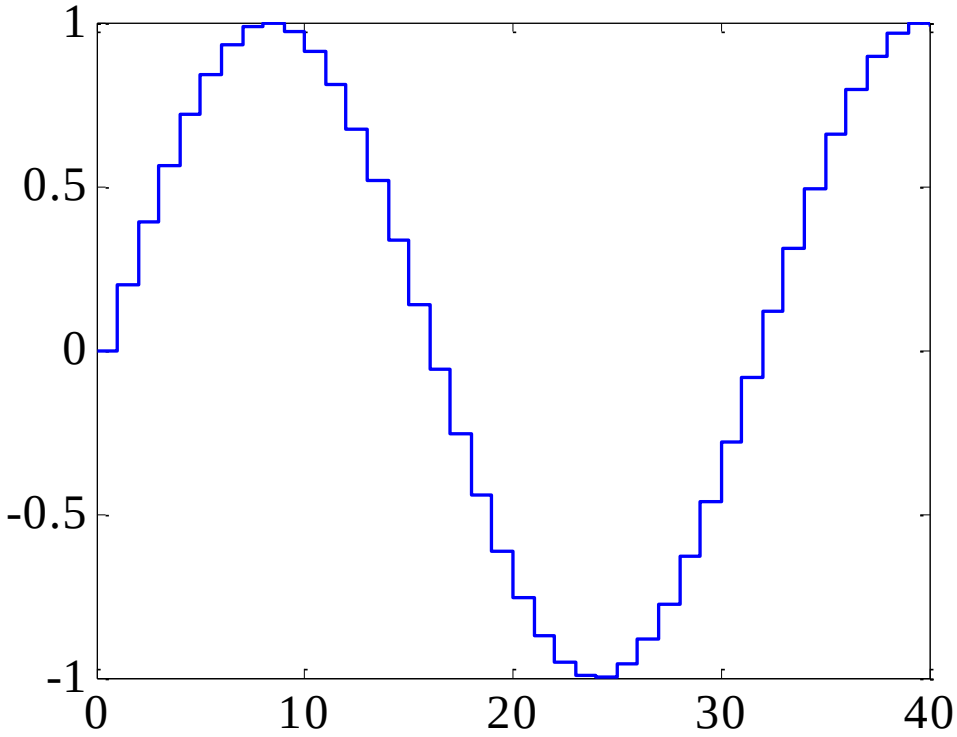


useful for displaying discrete-time signals in DSP applications

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

Stairs plot

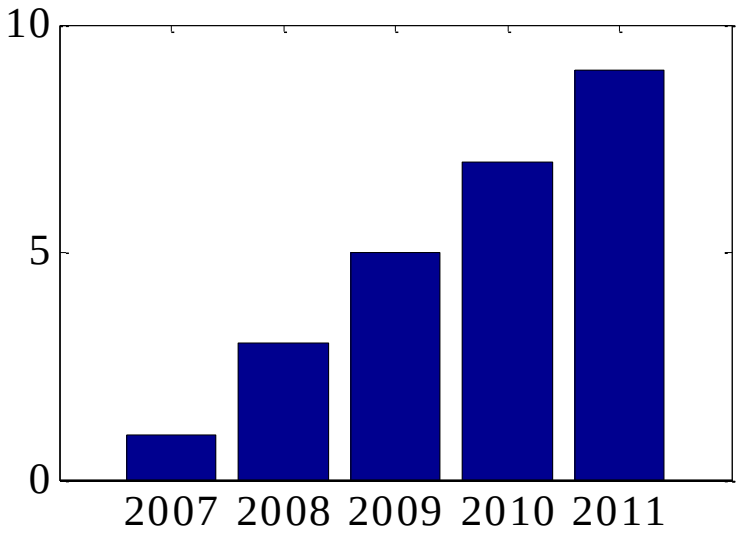
```
x = linspace(0,40,41);
y = sin(x/5);

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

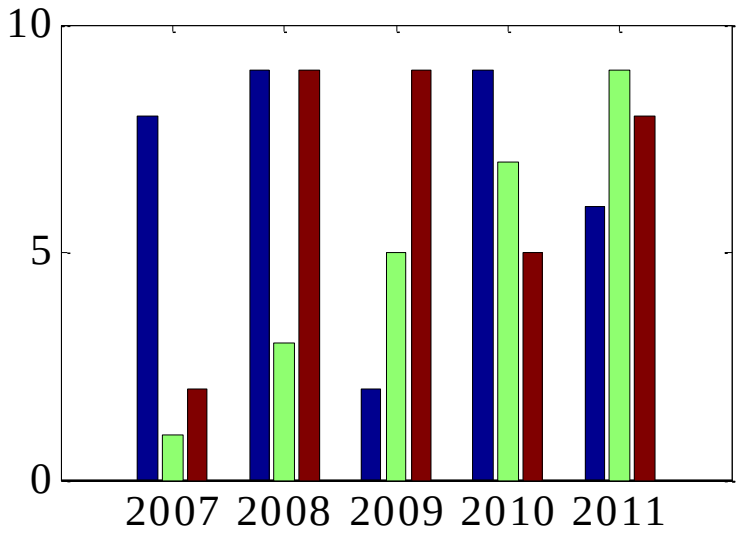


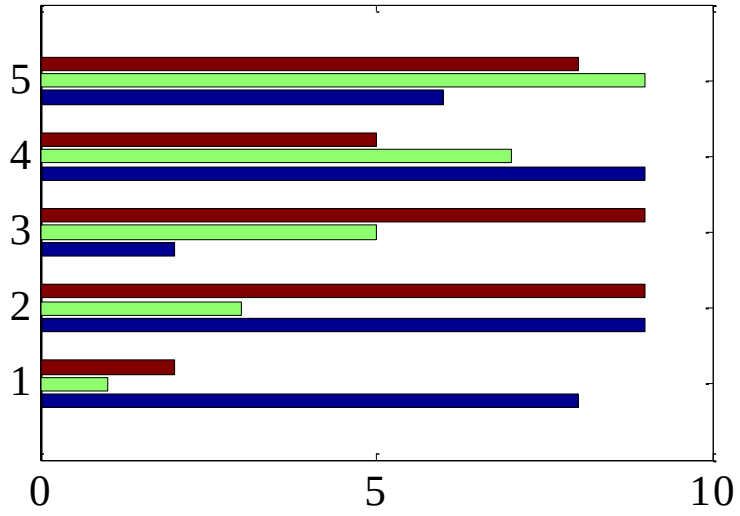
bar graphs

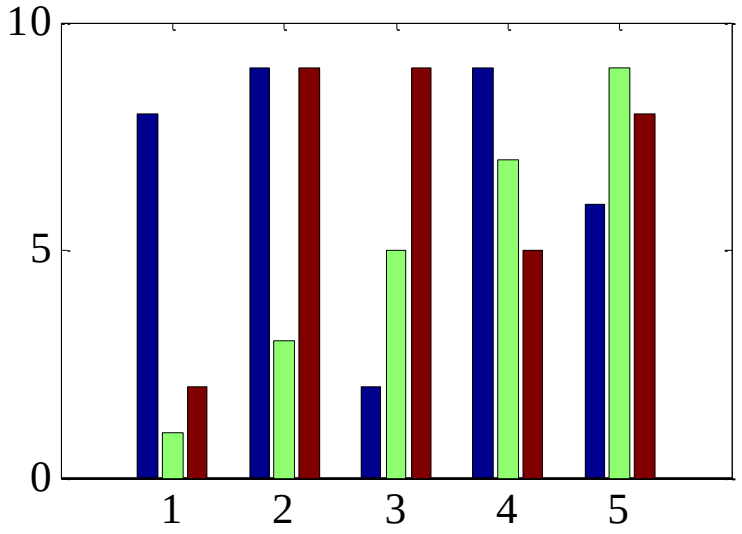
```
Y =[8 1 2
    9 3 9
    2 5 9
    9 7 5
    6 9 8];

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

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

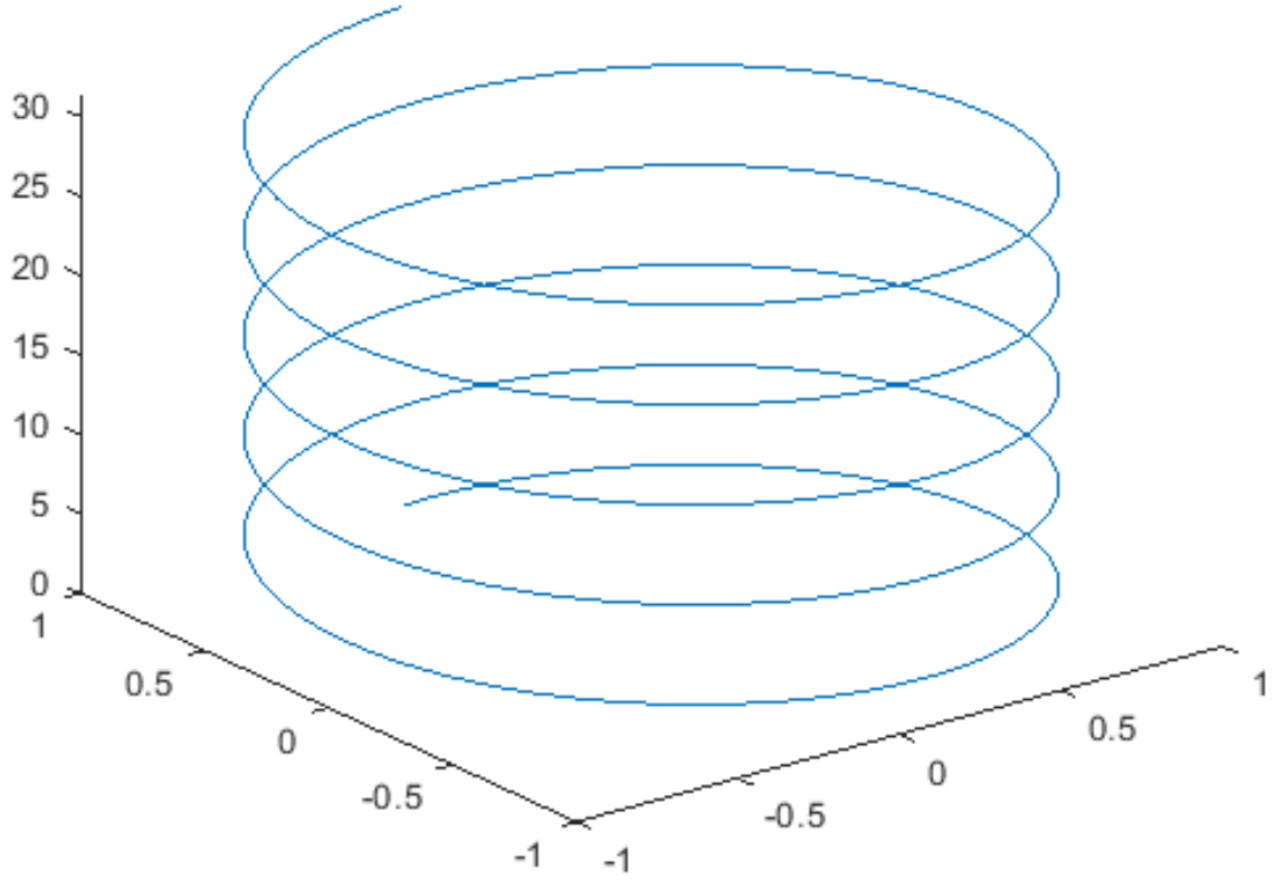

10
5
0
1
2
3
4
5

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**

Example of 3D plot with “plot3”

```
t = 0:pi/50:10*pi;
st = sin(t);
ct = cos(t);
plot3(st,ct,t)
```



Example of 3D plot with “contour”

```
x = linspace(-2*pi,2*pi);
y = linspace(0,4*pi);
[X,Y] = meshgrid(x,y);
Z = sin(X)+cos(Y);
contour(X,Y,Z)
```