Initializing Vectors

- colon operator
  - x = 1:2:10

    ```
    x =
         1     3     5     7     9
    ```
  - y = 0:0.1:0.5

    ```
    y =
         0    0.1    0.2    0.3    0.4    0.5
    ```
- built-in functions
  - zeros(), ones()

Initializing Vectors

- linspace(x1,x2) generates a row vector of 100 linearly equally spaced points between x1 and x2
- linspace(x1,x2,N) generates N points between x1 and x2
  - x = linspace(10,20,5)

    ```
    x =
       10.00   12.50   15.00   17.50   20.00
    ```
- logspace(x1,x2) can be used for logarithmically equally spaced points

Vector Input to Functions

- You can call built-in functions with array inputs
- The function is applied to all elements of the array
- The result is an array with the same size as the input array

Vector Input to Functions

- Examples:
  - x = [3 -2 9 4 -5 6 2];
  - abs(x)

    ```
    ans =
         3     2     9     4     5     6     2
    ```
  - sin([0 pi/6 pi/4 pi/3 pi/2])

    ```
    ans =
            0    0.5000    0.7071    0.8660    1.0000
    ```
  - a = 1:5;
  - log(a)

    ```
    ans =
            0    0.6931    1.0986    1.3863    1.6094
    ```

Vector Operations

- Scalar-vector operations

  -
    ```
    x = 1:5
    x =
         1     2     3     4     5
    ```

  -
    ```
    y = 2 * x              ← scalar multiplication
    y =
         2     4     6     8    10
    ```

  -
    ```
    z = x + 10             ← scalar addition
    z =
        11    12    13    14    15
    ```

Vector Operations

- Vector-vector operations (element-by-element operations)
  - `x = [ 1 2 3 4 5 ];     y = [ 2 -1 4 3 -2 ];`
  - `z = x + y`
    ```
    z =
         3     1     7     7     3
    ```
  - `z = x .* y`
    ```
    z =
         2    -2    12    12   -10
    ```
  - `z = x ./ y`
    ```
    z =
        0.5000   -2.0000    0.7500    1.3333   -2.5000
    ```

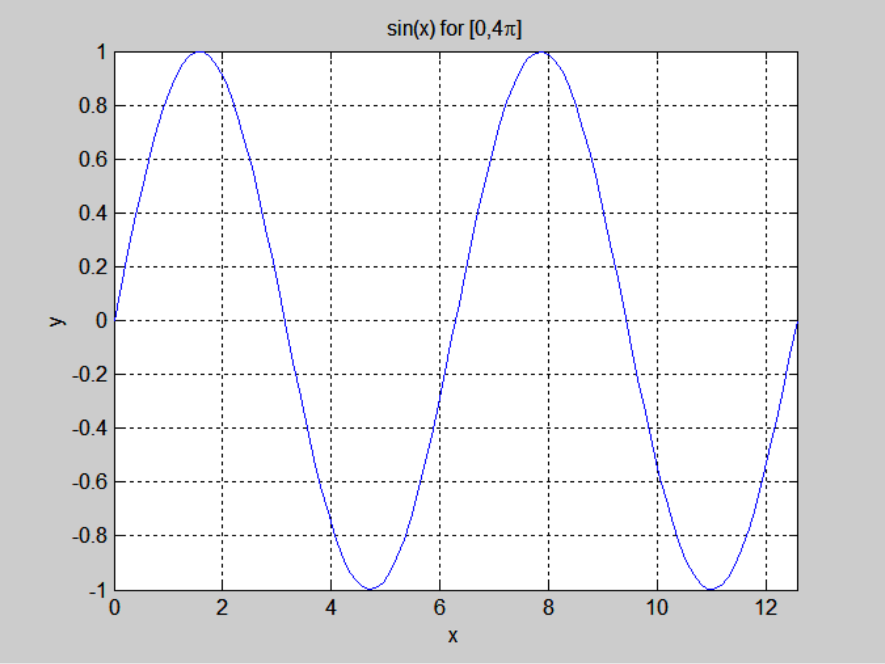
Vector Operations

- Vector-vector operations (element-by-element operations)
    - z = x .^ y
    - z =

      ```
          1.00    0.50   81.00   64.00    0.04
      ```
- Use .*, ./, .^ for element-by-element operations
- Vector dimensions must be the same

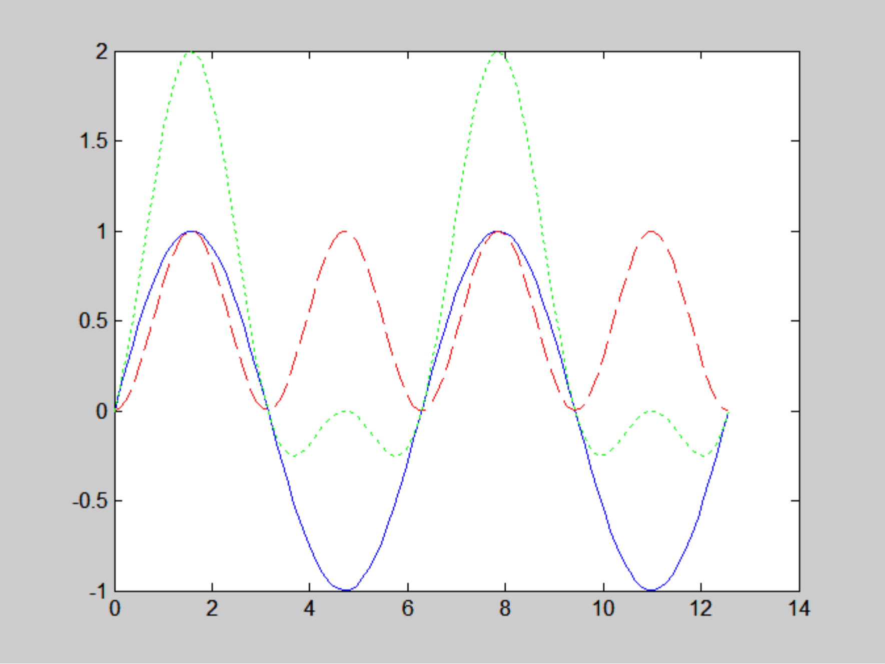
Plotting

```
x = linspace(0, 4*pi);
y = sin(x);
plot(x,y);
title( 'sin(x) for [0,4\pi]' );
xlabel( 'x' );
ylabel( 'y' );
grid on;
axis( [ 0 4*pi -1 1 ] );
```



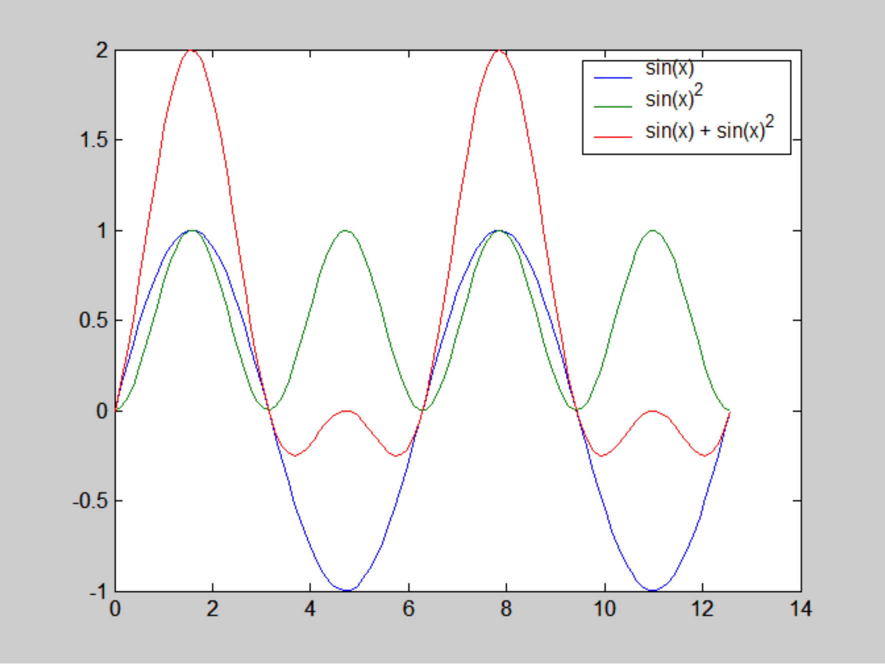
Plotting: Multiple Graphs

```
x = linspace(0, 4*pi);
y1 = sin(x);
y2 = sin(x) .^ 2;
y3 = y1 + y2;
plot(x,y1,'b-');
hold on;
plot(x,y2,'r--');
plot(x,y3,'g:');
hold off;
```



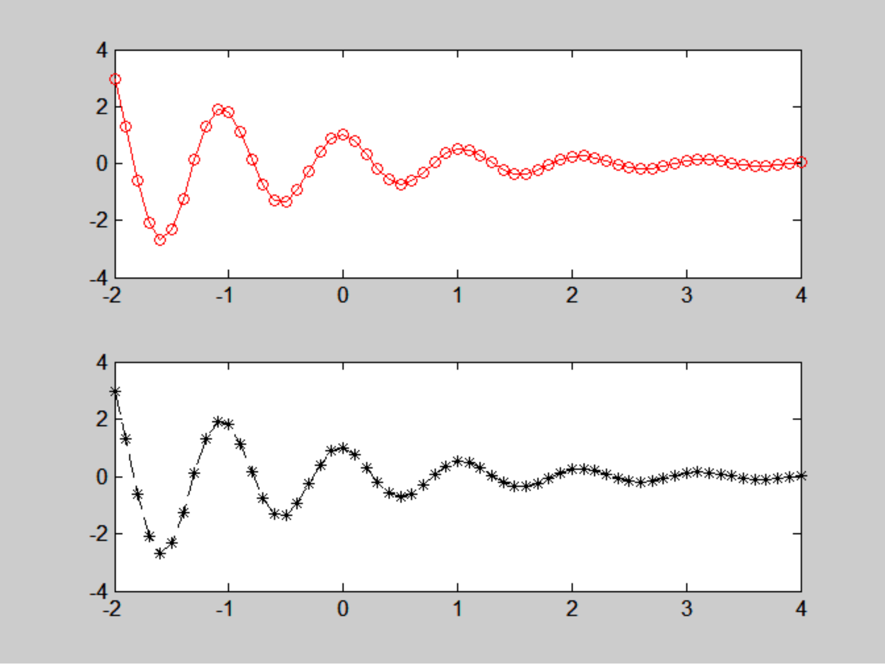
Plotting: Multiple Graphs

```
x = linspace(0, 4*pi);
y1 = sin(x);
y2 = sin(x) .^ 2;
y3 = y1 + y2;
plot(x,y1,x,y2,x,y3);
legend( 'sin(x)', ...
    'sin(x)^2', ...
    'sin(x) + sin(x)^2' );
```



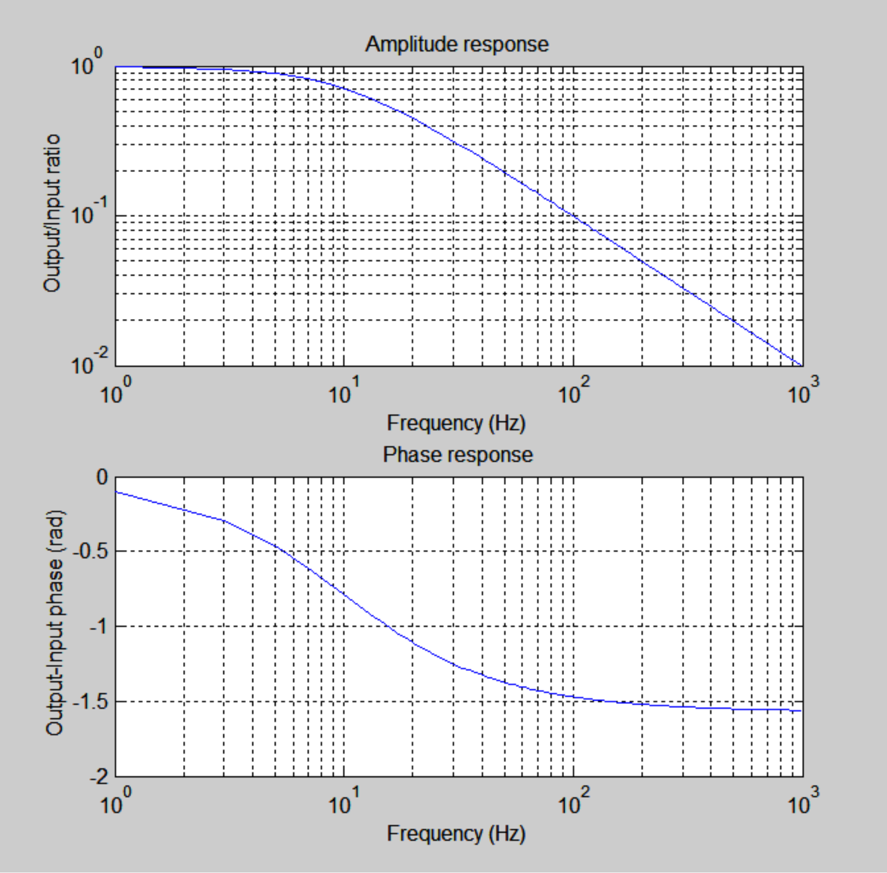
Plotting: Subplots

```
x = -2:0.1:4;
y = 3.5 .^ (-0.5*x) .* ...
   cos(6*x);
figure(1);
subplot(2,1,1);
plot(x,y,'r-o');
subplot(2,1,2);
plot(x,y,'k--*');
print -f1 -dtiff myplot.tif
```



Plotting: Logarithmic Plots

```
r = 16000; c = 1.0e-6;
f = 1:2:1000;
res = 1 ./ ( 1 + j*2*pi*f*r*c );
amp = abs(res);
phase = angle(res);
subplot(2,1,1);
loglog(f,amp);
title( 'Amplitude response' );
xlabel( 'Frequency (Hz)' );
ylabel( 'Output/Input ratio' );
grid on;
subplot(2,1,2);
semilogx(f,phase);
title( 'Phase response' );
xlabel( 'Frequency (Hz)' );
ylabel( 'Output-Input phase (rad)' );
grid on;
```



Plotting Summary

- plot(x,y)
  linear plot of vector y vs. vector x
- title('text'), xlabel('text'), ylabel('text')
  labels the figure, x-axis and y-axis
- grid on/off
  adds/removes grid lines
- legend('string1', 'string2', 'string3', ...)
  adds a legend using the specified strings
- hold on/off
  allows/disallows adding subsequent graphs to the current graph

Plotting Summary

- axis([xmin xmax ymin ymax])
  sets axes’ limits
- v = axis
  returns a row vector containing the scaling for the current plot
- axis equal
  sets the same scale for both axes
- axis square
  makes the current axis square in size

Plotting Summary

line color		line marker		line style	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Plotting Summary

- semilogy(x,y), semilogx(x,y), loglog(x,y)
  logarithmic plots of vector y vs. vector x
- figure(k)
  makes k the current figure
- subplot(m,n,p)
  breaks the figure window into an m-by-n matrix of small axes and selects the pth axes for the current plot
- clf
  clears current figure

Plotting Summary

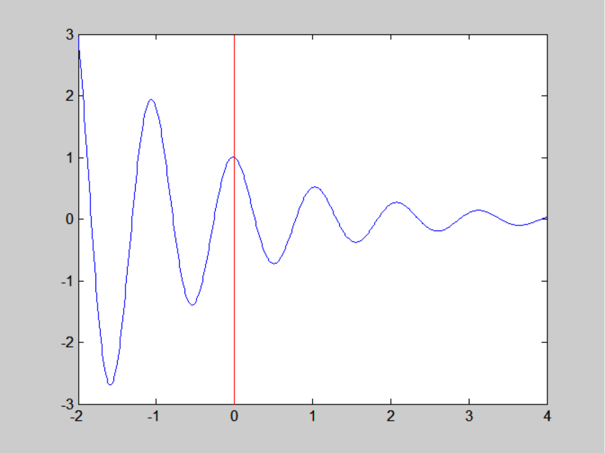
- print –f<handle> -d<device> <filename> saves the figure with the given handle in the format specified by the device

-deps	Encapsulated PostScript
-depsc	Encapsulated Color PostScript
-deps2	Encapsulated Level 2 PostScript
-depsc2	Encapsulated Level 2 Color PostScript
-djpeg<nn>	JPEG image with quality level of nn
-dtiff	TIFF image
-dpng	Portable Network Graphics image

Plotting Examples

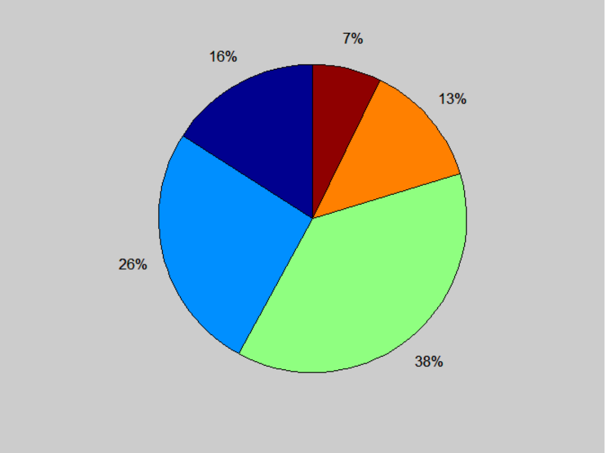
- Line plot

```
x = -2:0.01:4;
y = 3.5.^(-0.5*x).*cos(6*x);
plot(x,y);
line([0 0],[-3 3],'color','r');
```



- Pie plot

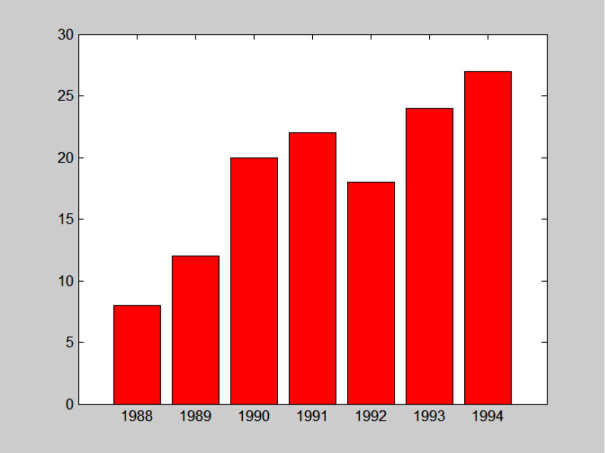
```
grades = [ 11 18 26 9 5 ];
pie(grades);
```



Plotting Examples

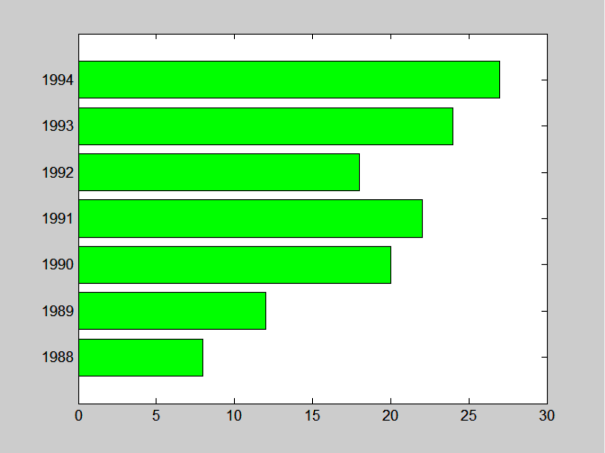
- Vertical bar plot

  y = 1988:1994;
  s = [8 12 20 22 18 24 27];
  bar(y,s,'r');



- Horizontal bar plot

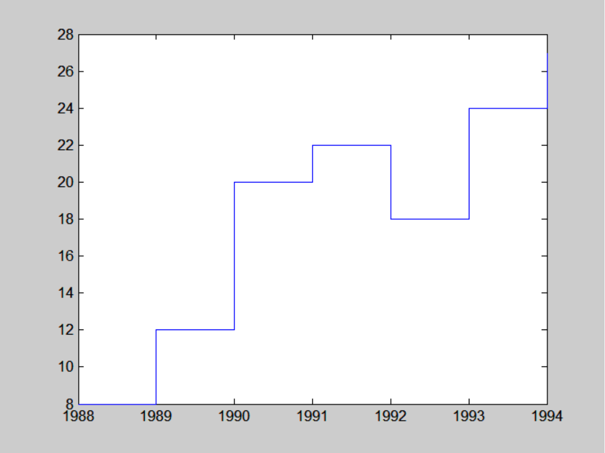
  y = 1988:1994;
  s = [8 12 20 22 18 24 27];
  barh(y,s,'g');



Plotting Examples

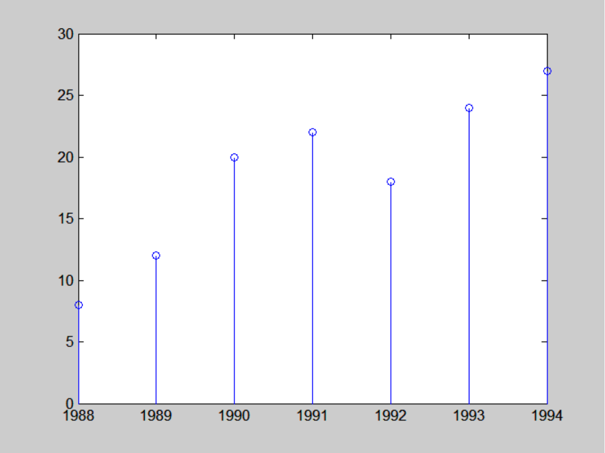
- Stairs plot

  ```
  y = 1988:1994;
  s = [ 8 12 20 22 18 24 27 ];
  stairs(y,s);
  ```



- Stem plot

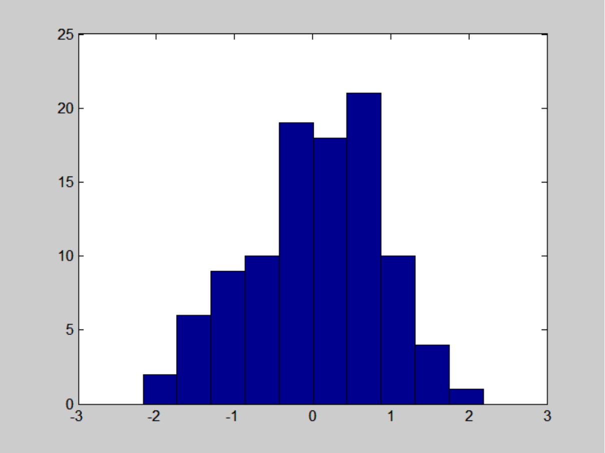
  ```
  y = 1988:1994;
  s = [ 8 12 20 22 18 24 27 ];
  stem(y,s);
  ```



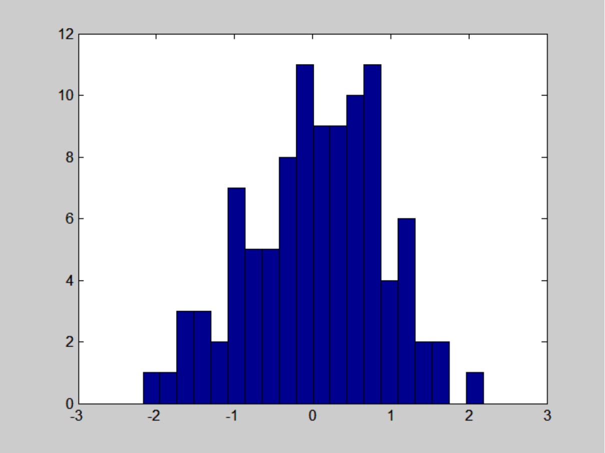
Plotting Examples

- Histogram

  x = randn(1,100);
  hist(x,10);



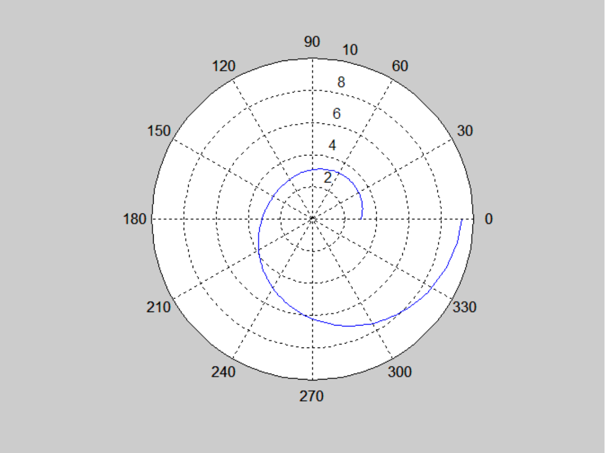
  hist(x,20);



Plotting Examples

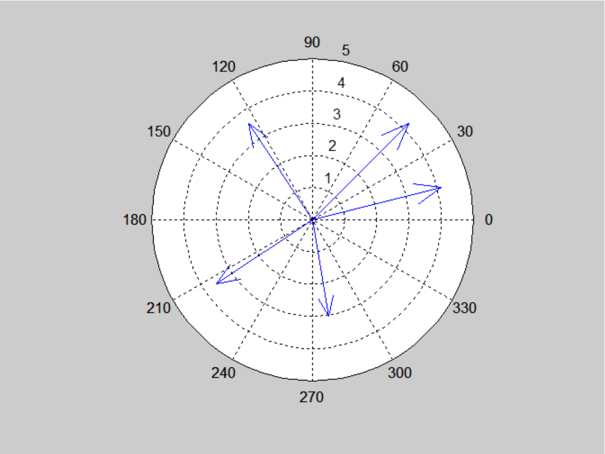
- Polar plot

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



- Compass plot

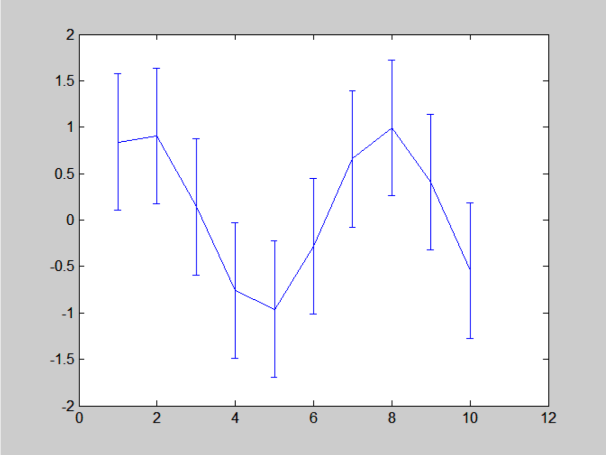
```
u = [ 3 4 -2 -3 0.5 ];
v = [ 3 1 3 -2 -3 ];
compass(u,v);
```



Plotting Examples

- Error bar plot

```
x = 1:10;
y = sin(x);
e = std(y) * ones(size(x));
errorbar(x,y,e);
```


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
2
4
6
8
10
12