# DISCRETE DATA PLOTS

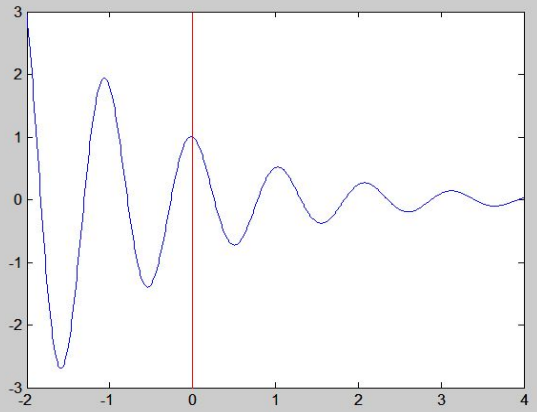
# TWO-DIMENSIONAL PLOTS

- Stem
- Stair
- Bar
- Pie
- compass

# TWO-DIMENSIONAL PLOTS: EXAMPLES

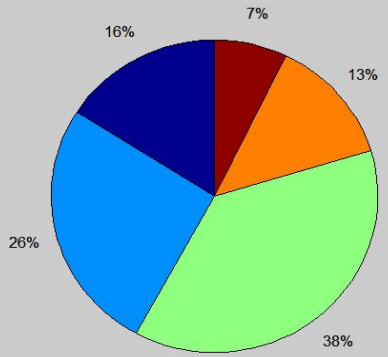
- Line plot

```
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

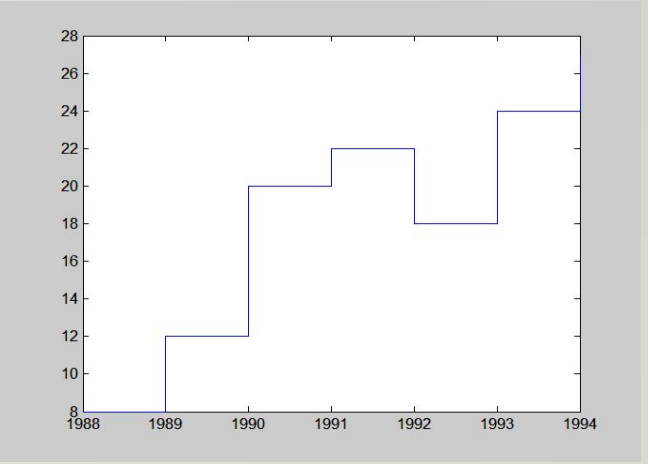
```
grades = [ 11 18 26 9 5 ];
pie(grades);
```



# TWO-DIMENSIONAL PLOTS: EXAMPLES

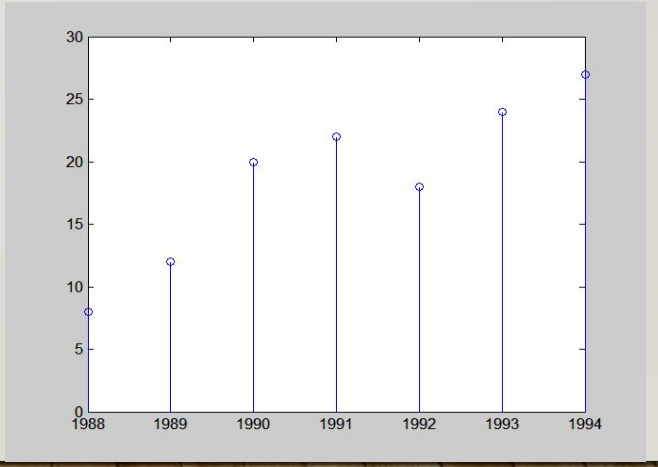
- Stairs plot

```
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);
```



# STAIRS PLOT

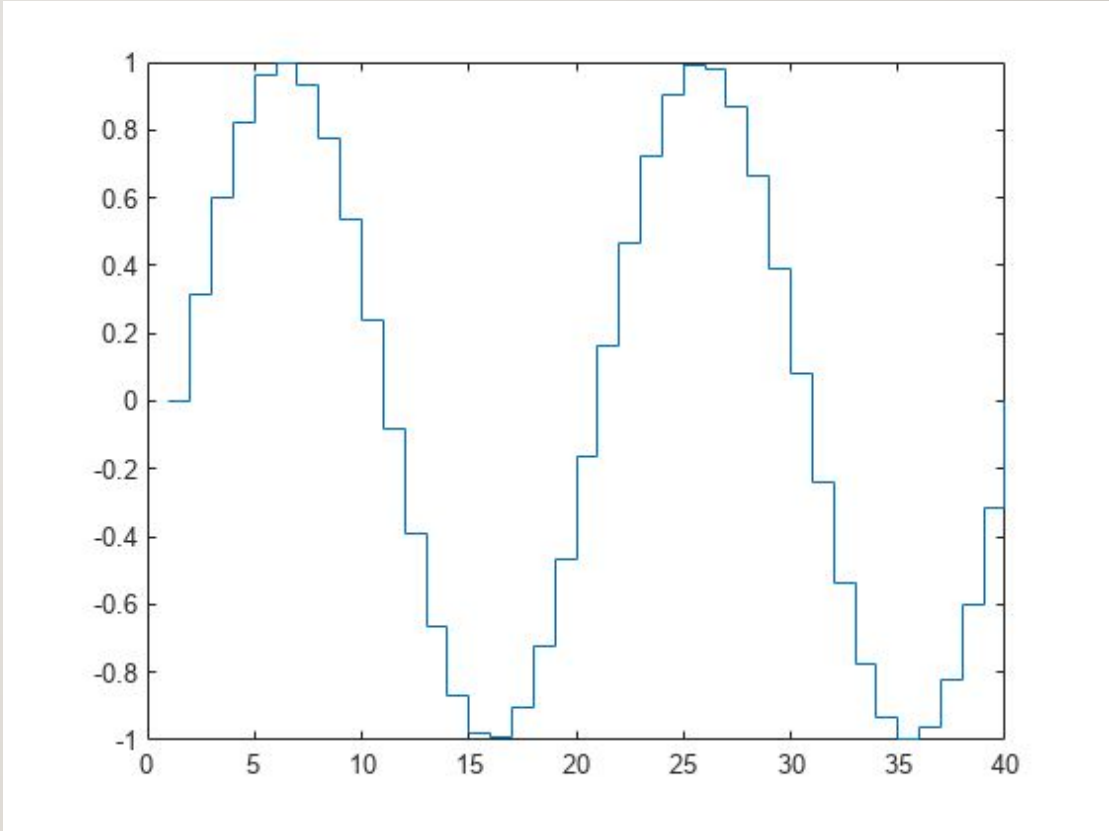
- stairs(Y) draws a stairstep graph of the elements in Y.
- If Y is a vector, then stairs draws one line.

- If Y is a matrix, then stairs draws one line per matrix column.
- stairs(X,Y) plots the elements in Y at the locations specified by X. The inputs X and Y must be vectors or matrices of the same size. Additionally, X can be a row or column vector and Y must be a matrix with length(X) rows.
- stairs(___,LineSpec) specifies a line style, marker symbol, and color. For example, ":*r" specifies a dotted red line with asterisk markers. Use this option with any of the input argument combinations in the previous syntaxes.

# STAIRS PLOT

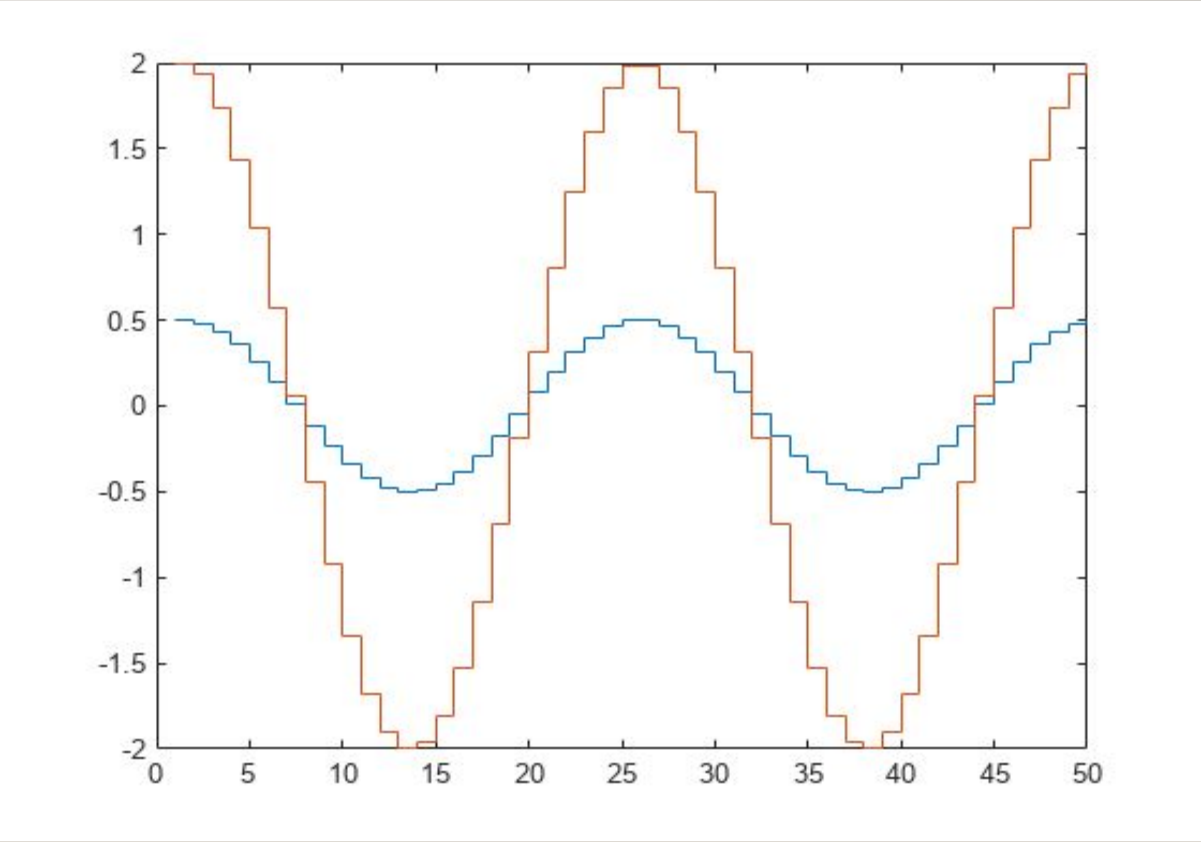
1. Create a stairstep plot of sine evaluated at 40 equally spaced values between 0 and 4π.

```
X = linspace(0,4*pi,40);
Y = sin(X);
figure
stairs(Y)
```



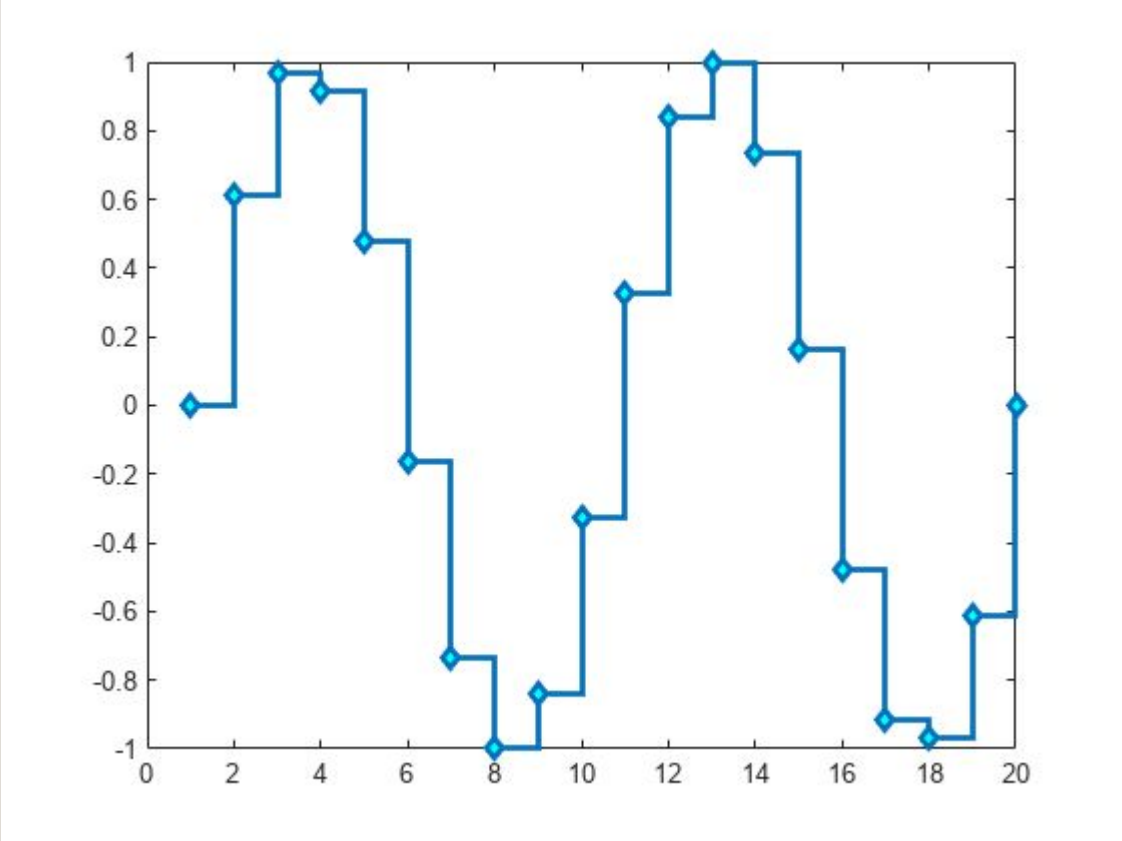
2. Create a stairstep plot of two cosine functions evaluated at 50 equally spaced values between 0 and 4π.

```
X = linspace(0,4*pi,50)';

Y = [0.5*cos(X), 2*cos(X)];

figure

stairs(Y)
```



3. Create a stairstep plot and set the line width to 2, the marker symbols to diamonds, and the marker face color to cyan using Name,Value pair arguments.

```
X = linspace(0,4*pi,20);
Y = sin(X);
figure
stairs(Y,'LineWidth',2,'Marker','d','MarkerFaceColor','c')
```



# STEM PLOT

- stem(Y) plots the data sequence, Y, as stems that extend from a baseline along the x-axis. The data values are indicated by circles terminating each stem.
- If Y is a vector, then the x-axis scale ranges from 1 to length(Y).
- If Y is a matrix, then stem plots all elements in a row against the same x value, and the x-axis scale ranges from 1 to the number of rows in Y.
- stem(X,Y) plots the data sequence, Y, at values specified by X. The X and Y inputs must be vectors or matrices of the same size. Additionally, X can be a row or column vector and Y must be a matrix with length(X) rows.
- If X and Y are both vectors, then stem plots entries in Y against corresponding entries in X.
- If X is a vector and Y is a matrix, then stem plots each column of Y against the set of values specified by X, such that all elements in a row of Y are plotted against the same value.
- If X and Y are both matrices, then stem plots columns of Y against corresponding columns of X.
- stem(___,"filled") fills the circles. Use this option with any of the input argument combinations in the previous syntaxes.
- stem(___,LineSpec) specifies the line style, marker symbol, and color.

# STEM PLOT

1. Create a stem plot of 50 data values between −2π and 2π.

figure

Y = linspace(-2*pi,2*pi,50);

stem(Y)

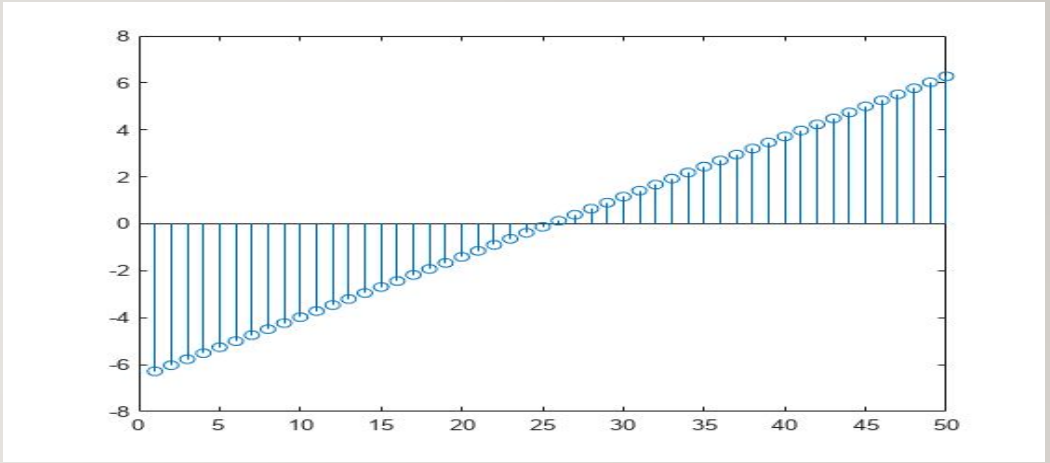
2. Create a stem plot and set the line style to a dotted line, the marker symbols to diamonds, and the color to red using the LineSpec option.

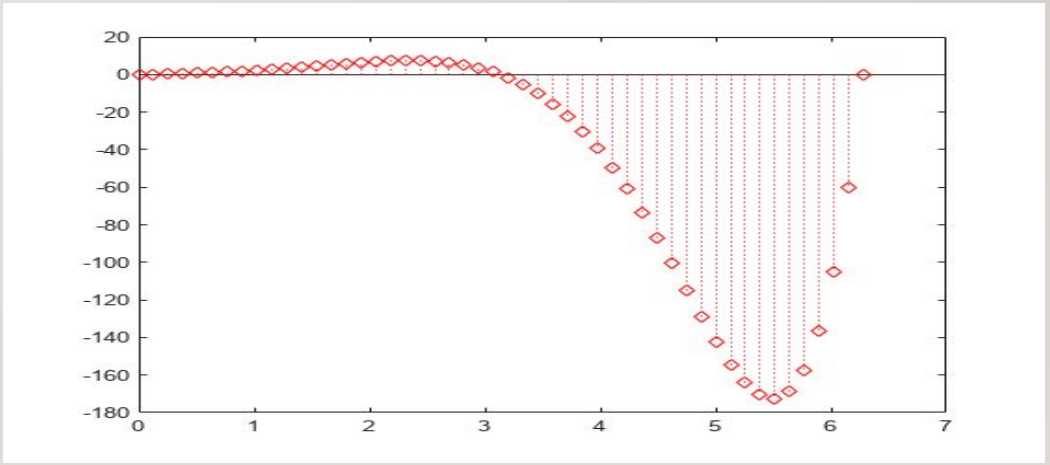
figure

X = linspace(0,2*pi,50)';

Y = (exp(X).*sin(X));

stem(X,Y,':diamondr')

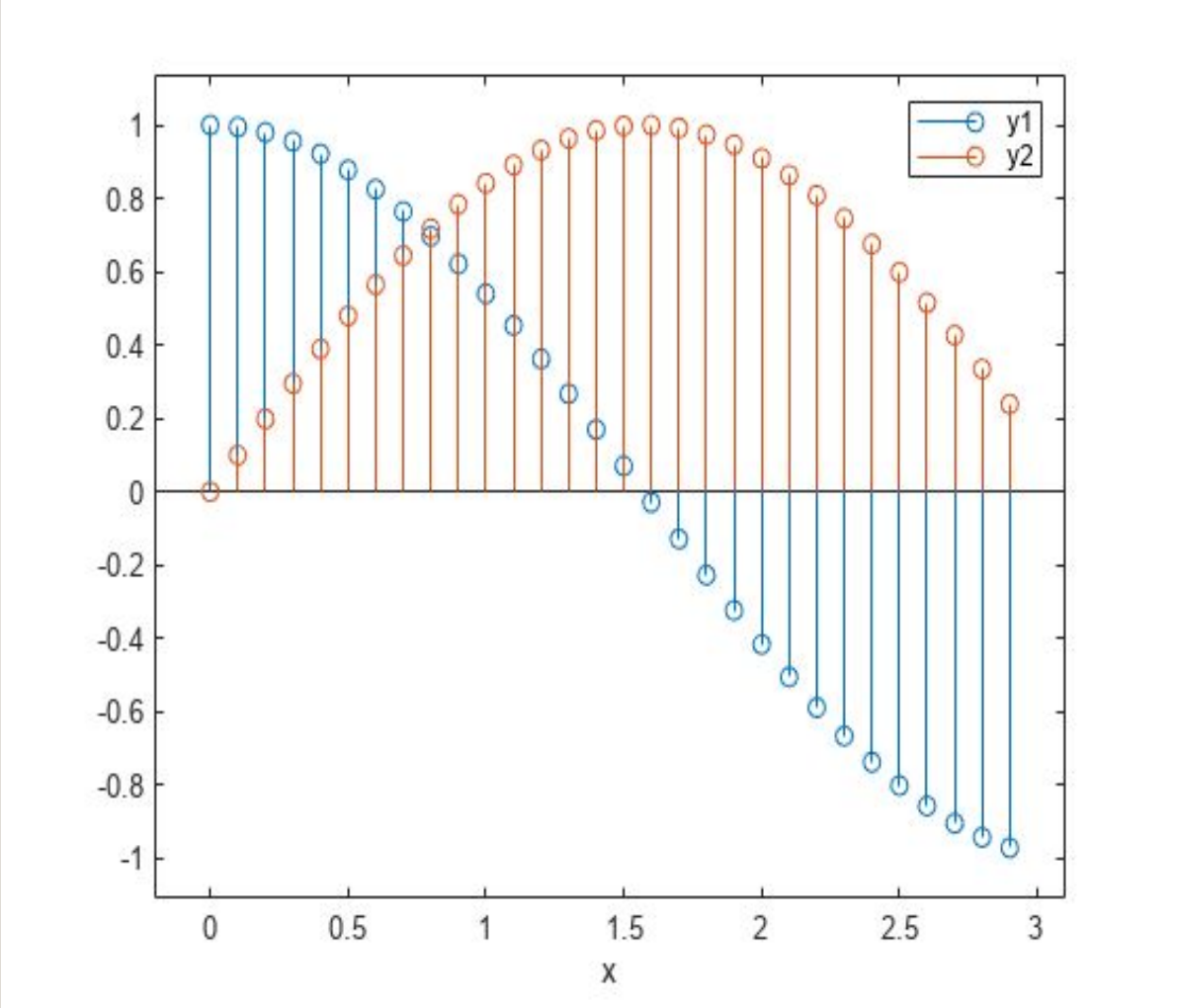


# STEM PLOT

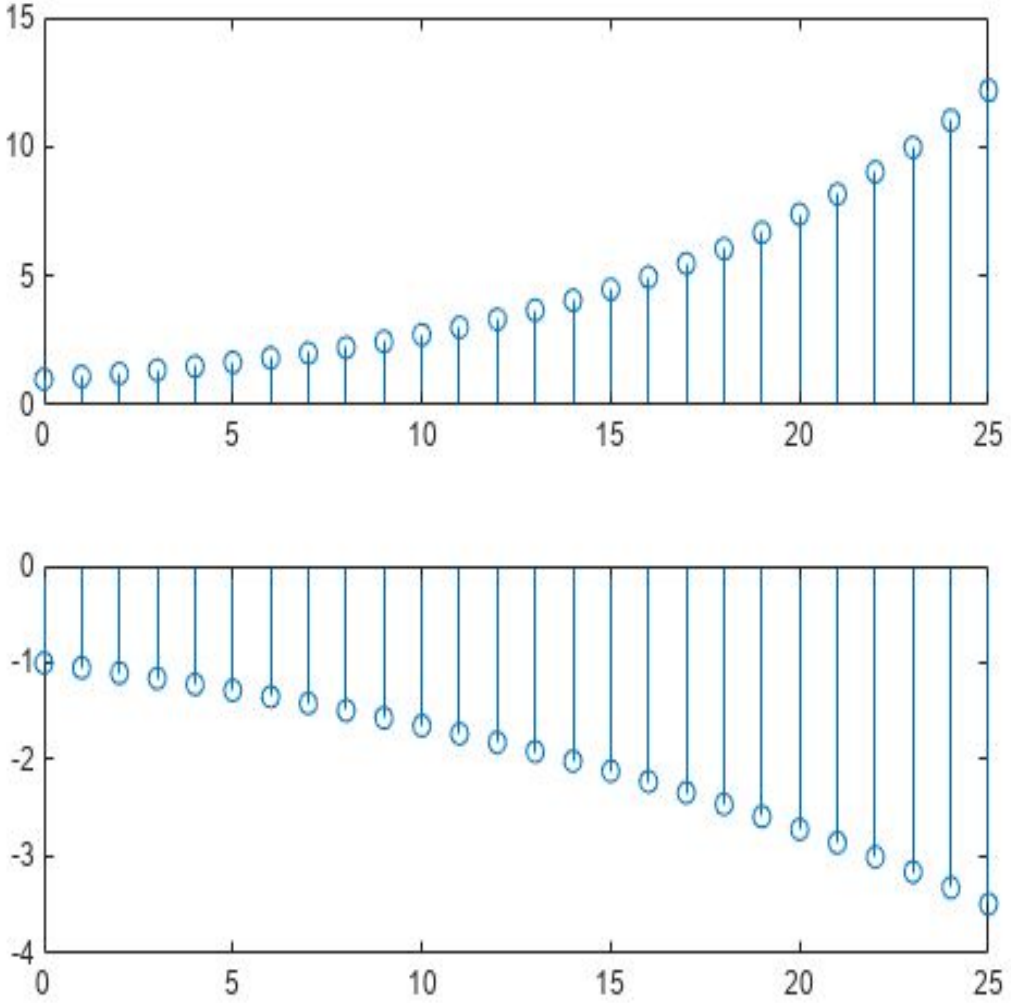
Create vectors x, y1, and y2, and use them to create a table. Plot the y1 and y2 variables against the x variable, and use the axis padded command so that the stems do not overlap with the plot box. Then add a legend, and notice that the legend labels match the table variable names.

```
x = (0:0.1:2.9)';

y1 = cos(x);

y2 = sin(x);

tbl = table(x,y1,y2);

stem(tbl,"x",["y1","y2"]);


% Pad axes and add a legend

axis padded

legend
```



You can display a tiling of plots using the tiledlayout and nexttile functions. Call the tiledlayout function to create a 2-by-1 tiled chart layout. Call the nexttile function to create the axes objects ax1 and ax2. Create separate stem plots in the axes by specifying the axes object as the first argument to stem.

```
x = 0:25;
y1 = exp(0.1*x);
y2 = -exp(.05*x);
tiledlayout(2,1)
% Top plot
ax1 = nexttile;
stem(ax1,x,y1)
% Bottom plot
ax2 = nexttile;
stem(ax2,x,y2)
```



Create a stem plot and change properties of the baseline.

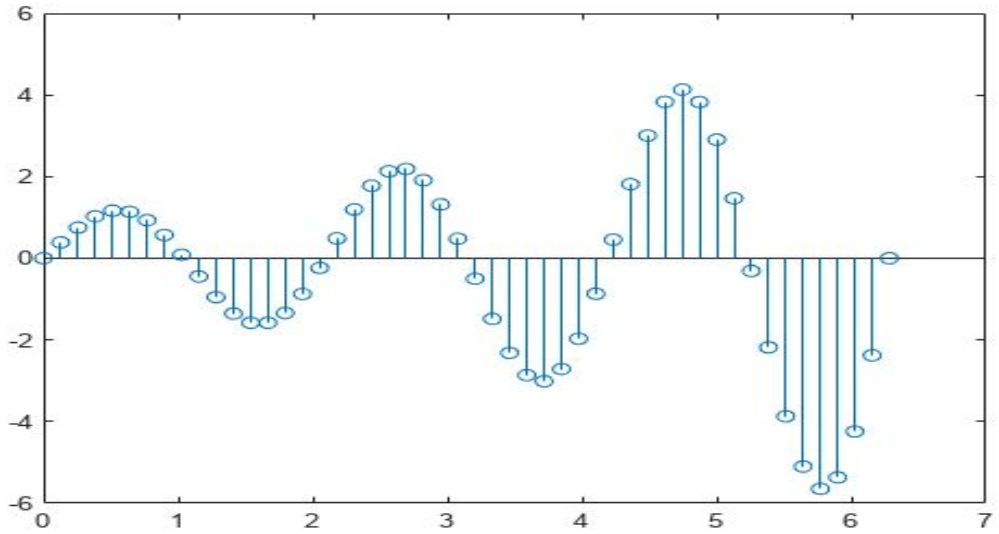
```
X = linspace(0,2*pi,50);

Y = exp(0.3*X).*sin(3*X);

h = stem(X,Y);
```

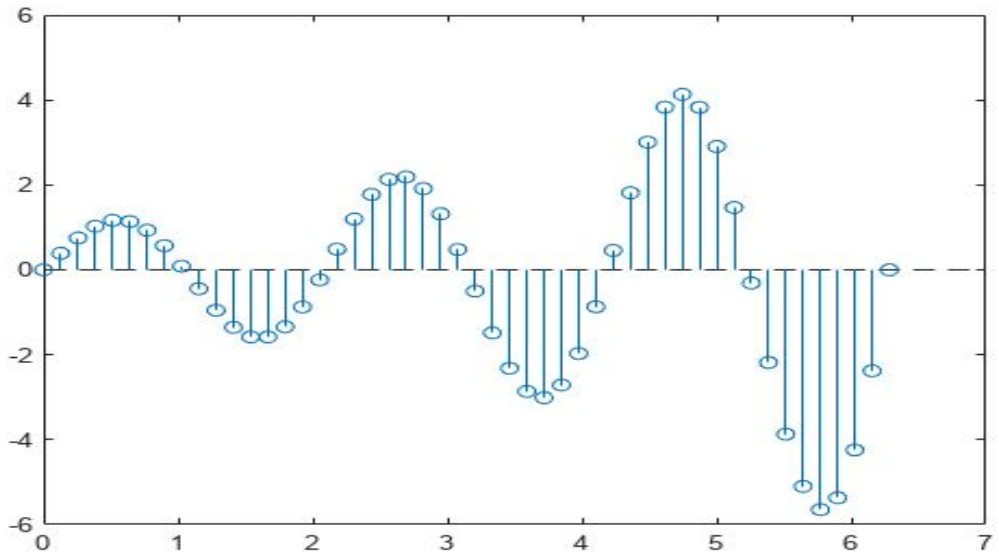
Change the line style of the baseline. Use dot notation to set properties.

```
hbase = h.BaseLine;

hbase.LineStyle = '--';
```

Hide the baseline by setting its Visible property to 'off' .

```
hbase.Visible = 'off';
```



# SCATTER PLOT

- scatter(x,y) creates a scatter plot with circular markers at the locations specified by the vectors x and y.
- To plot one set of coordinates, specify x and y as vectors of equal length.
- To plot multiple sets of coordinates on the same set of axes, specify at least one of x or y as a matrix.
- scatter(x,y,sz) specifies the circle sizes. To use the same size for all the circles, specify sz as a scalar. To plot each circle with a different size, specify sz as a vector or a matrix.
- scatter(x,y,sz,c) specifies the circle colors. You can specify one color for all the circles, or you can vary the color. For example, you can plot all red circles by specifying c as "red".

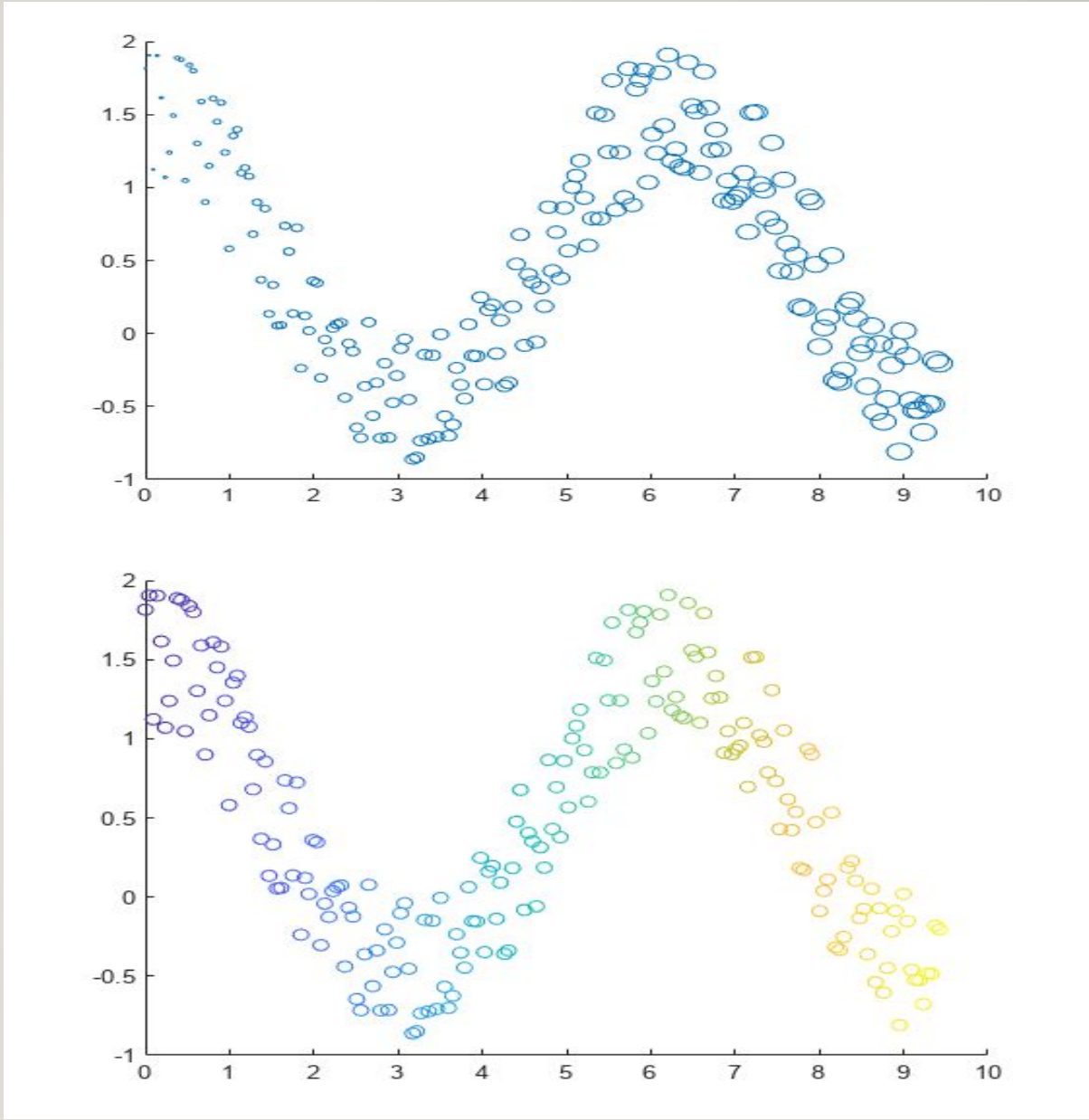
- scatter(___,"filled") fills in the circles. Use the "filled" option with any of the input argument combinations in the previous syntaxes.
- scatter(___,mkr) specifies the marker type.

Create a scatter plot using circles with different sizes. Specify the size in points squared

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
sz = linspace(1,100,200);
scatter(x,y,sz)
```

Create a scatter plot and vary the circle color.

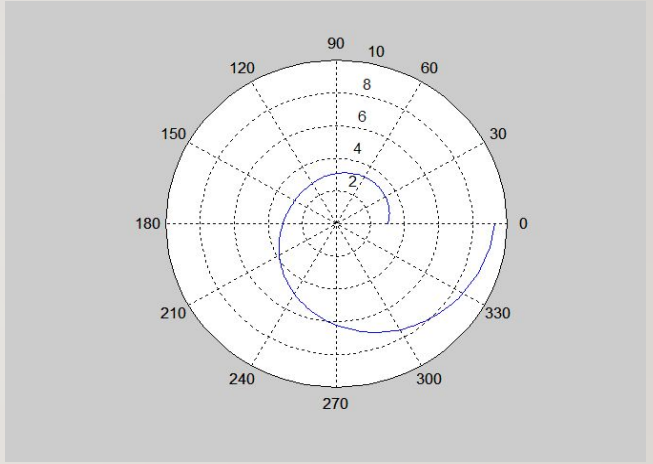
```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
c = linspace(1,10,length(x));
scatter(x,y,[],c)
```



# TWO-DIMENSIONAL PLOTS: EXAMPLES

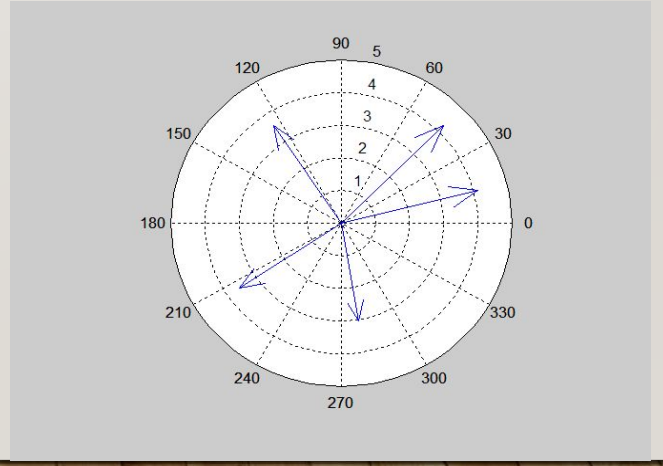
- Polar plot

```
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);
```

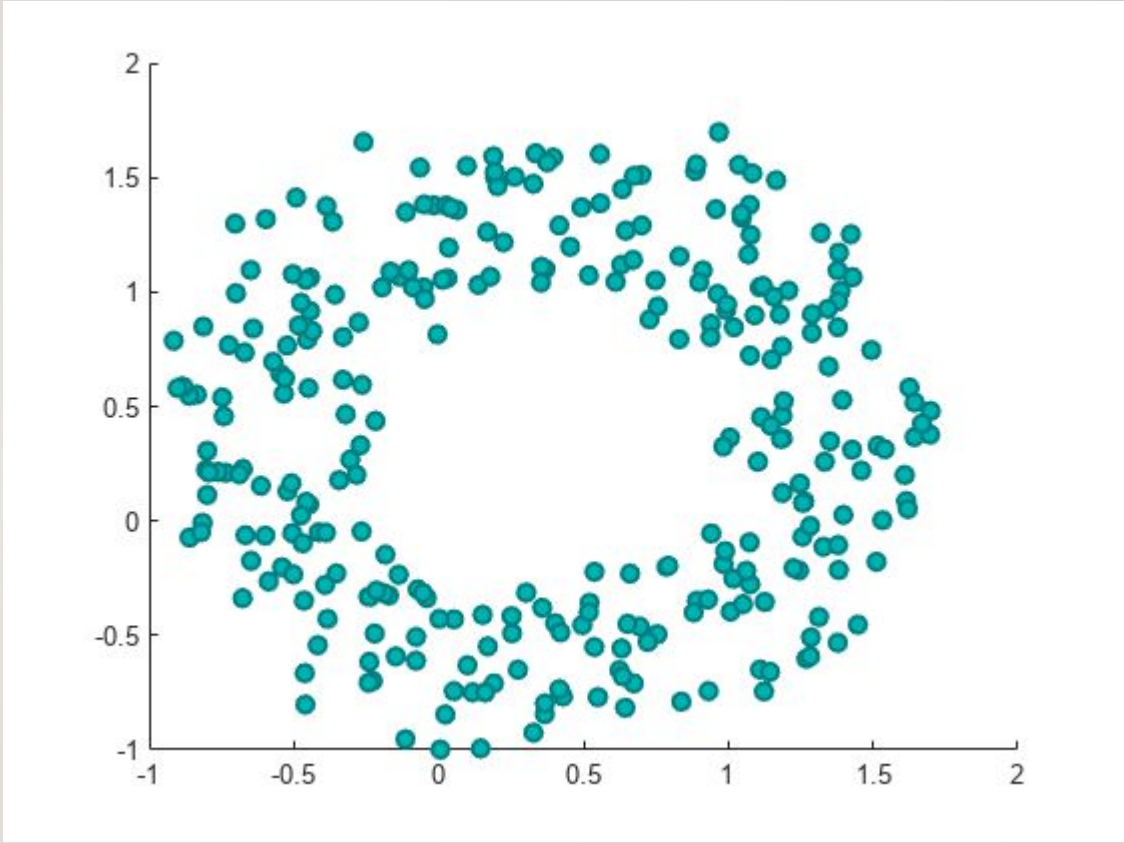


# POLAR PLOT

- polarplot(theta,rho) plots a line in polar coordinates, with theta indicating the angle in radians and rho indicating the radius value for each point. The inputs must be vectors of equal length or matrices of equal size. If the inputs are matrices, then polarplot plots columns of rho versus columns of theta. Alternatively, one of the inputs can be a vector and the other a matrix as long as the vector is the same length as one dimension of the matrix.
- polarplot(theta,rho,LineSpec) sets the line style, marker symbol, and color for the line.
- polarplot(theta1,rho1,...,thetaN,rhoN) plots multiple rho,theta pairs.
- polarplot(theta1,rho1,LineSpec1,...,thetaN,rhoN,LineSpecN) specifies the line style, marker symbol, and color for each line.
- polarplot(rho) plots the radius values in rho at evenly spaced angles between 0 and 2π.
- polarplot(rho,LineSpec) sets the line style, marker symbol, and color for the line.
- polarplot(Z) plots the complex values in Z.
- polarplot(Z,LineSpec) sets the line style, marker symbol, and color for the line.

Create vectors x and y as sine and cosine values with random noise. Create a scatter plot and set the marker edge color, marker face color, and line width.

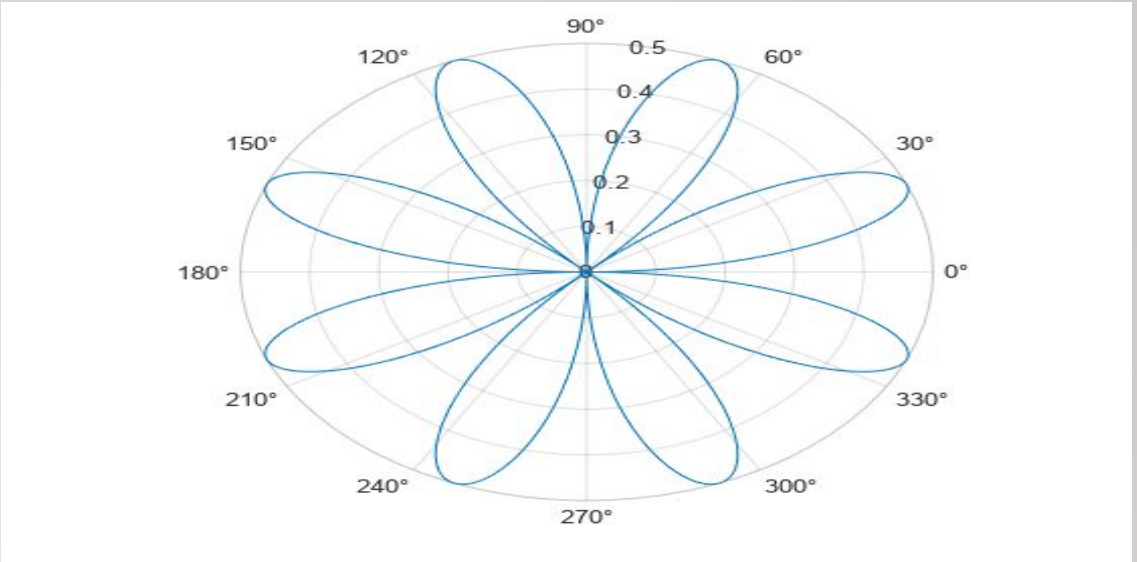
```
theta = linspace(0,2*pi,300);
x = sin(theta) + 0.75*rand(1,300);
y = cos(theta) + 0.75*rand(1,300);
sz = 40;
scatter(x,y,sz,'MarkerEdgeColor',[0 .5 .5],...
            'MarkerFaceColor',[0 .7 .7],...
            'LineWidth',1.5)
```



# POLAR PLOT

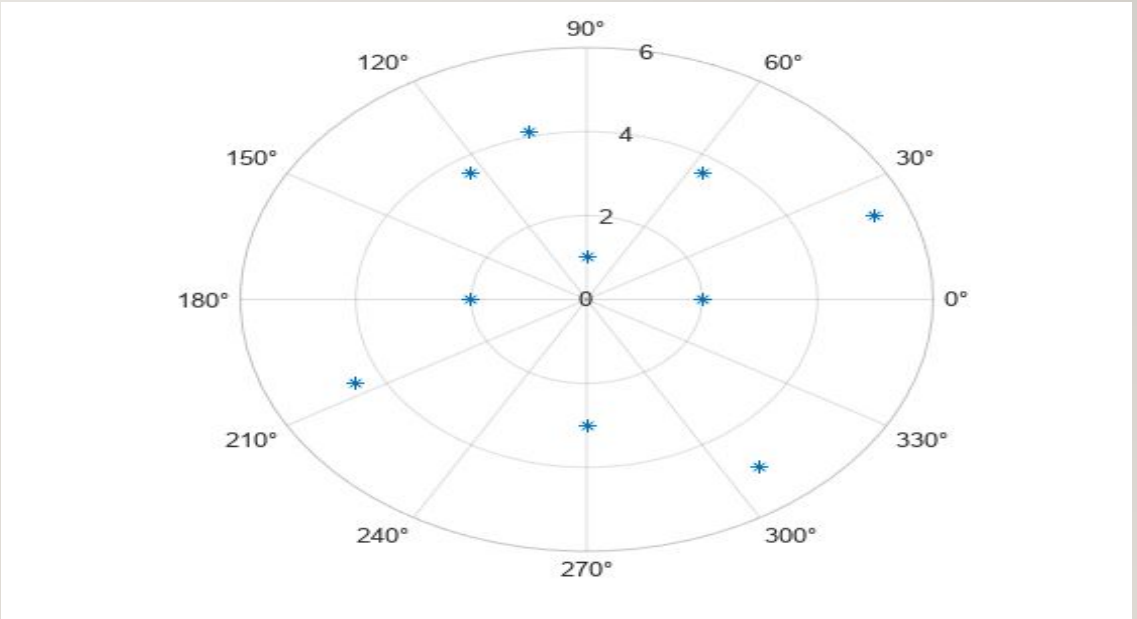
Plot a line in polar coordinates.

```
theta = 0:0.01:2*pi;

rho = sin(2*theta).*cos(2*theta);

polarplot(theta,rho)
```




Plot complex values in polar coordinates. Display markers at each point without a line connecting them.

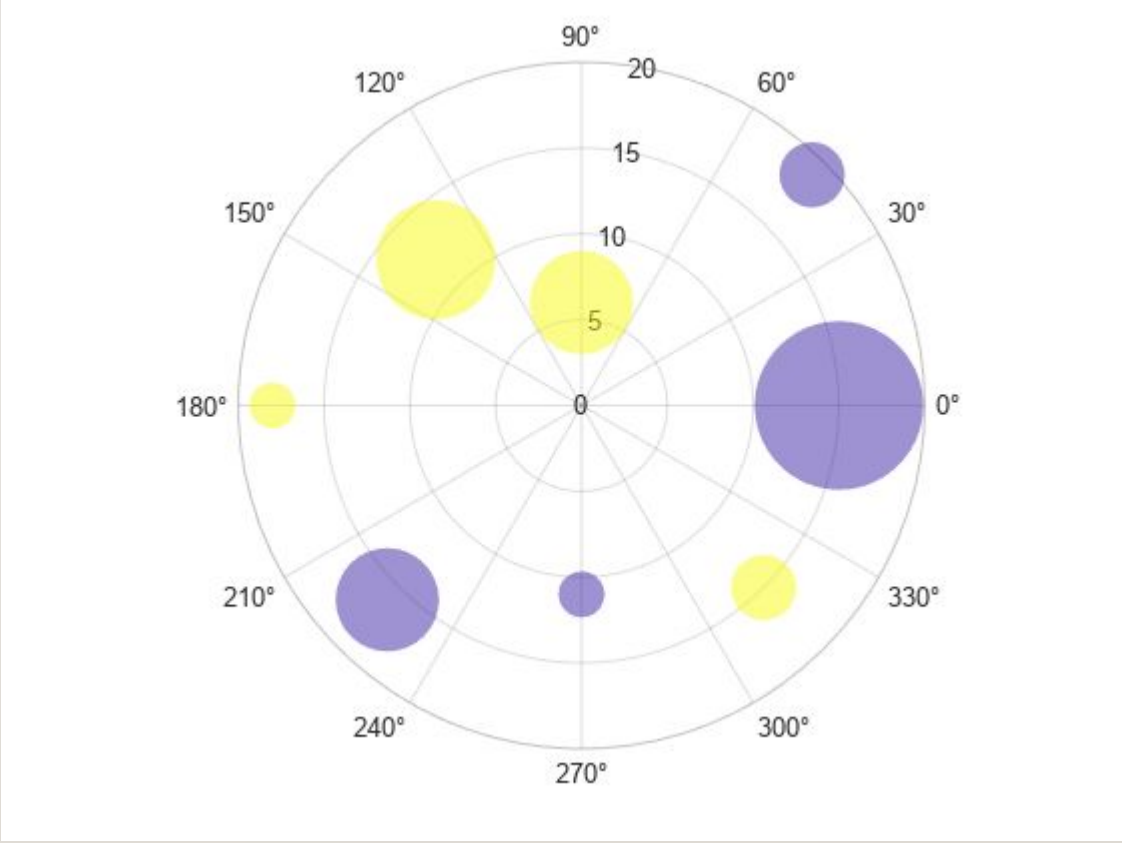
```
Z = [2+3i 2 -1+4i 3-4i 5+2i -4-2i -2+3i -2 -3i
3i-2i];

polarplot(Z,'*')
```



# POLARSCATTER PLOT

Create a scatter chart with markers of varying sizes and colors. Specify the optional size and color input arguments as vectors. Use unique values in the color vector to specify the different colors you want. The values map to colors in the colormap.

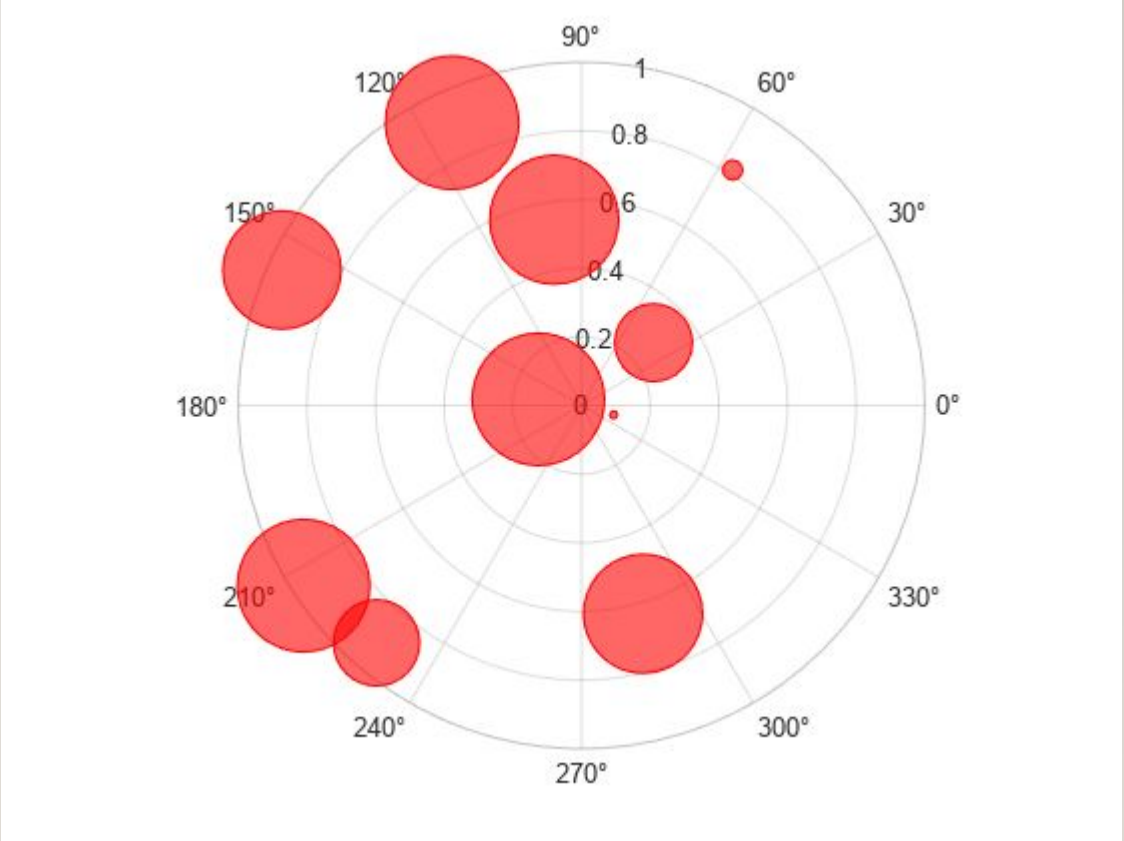
```
th = pi/4:pi/4:2*pi;

r = [19 6 12 18 16 11 15 15];

sz = 100*[6 15 20 3 15 3 6 40];

c = [1 2 2 2 1 1 2 1];

polarscatter(th,r,sz,c,'filled','MarkerFaceAlpha',.5)
```



# POLARBUBBLECHART

Define a set of bubble coordinates as the vectors th and r. Define sz as a vector of bubble sizes. Then create a bubble chart and specify the color as red. By default, the bubbles are partially transparent.

```
th = 1:10;

r = rand(1,10);

sz = rand(1,10);

polarbubblechart(th,r,sz,'red');
```

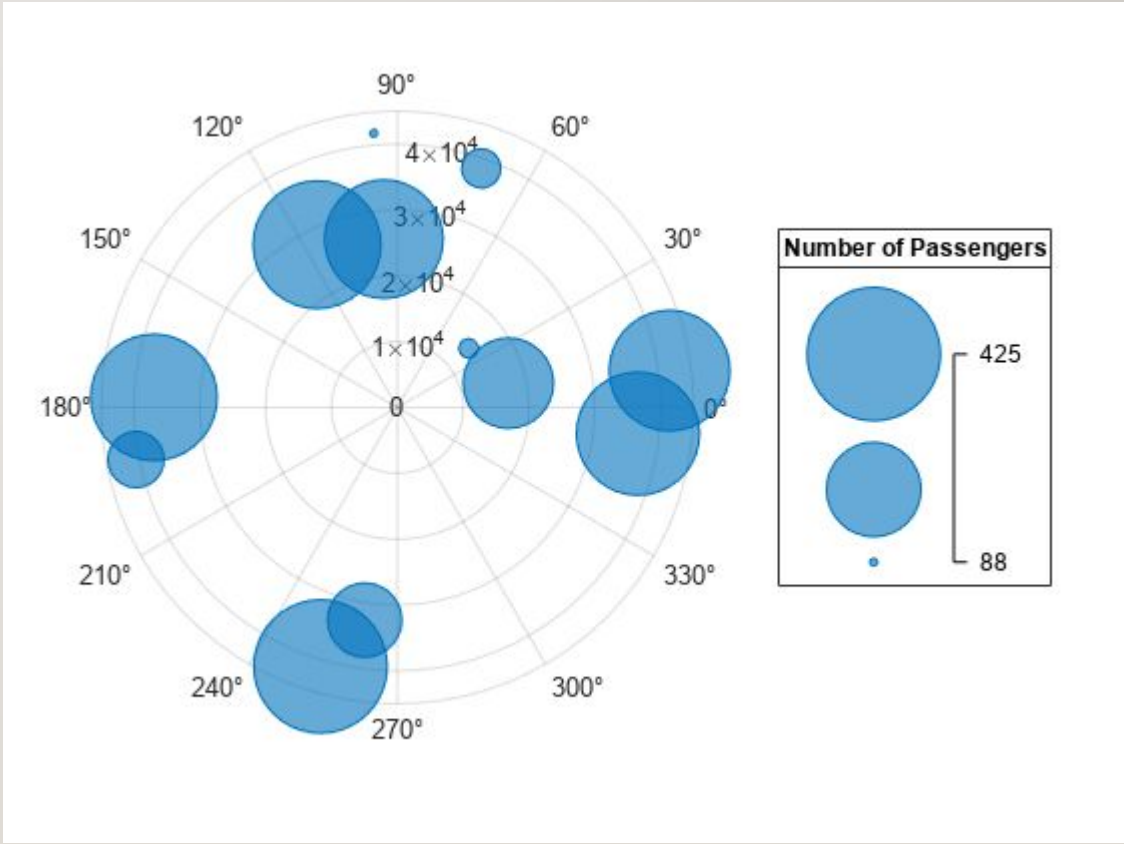


# POLARBUBBLECHART

Define planesize as a vector of plane sizes, measured in the number of passengers.

Then display the data in a bubble chart with a bubble legend that shows the relationship between the bubble sizes and the number of passengers on the planes.

```
theta = repmat([0 pi/2 7*pi/6],1,4) + 0.25*randn(1,12);

altitude = randi([13000 43000],1,12);

planesize = randi([75 500],[1 12]);

polarbubblechart(theta,altitude,planesize)

bubblelegend('Number of Passengers','Location','eastoutside')
```

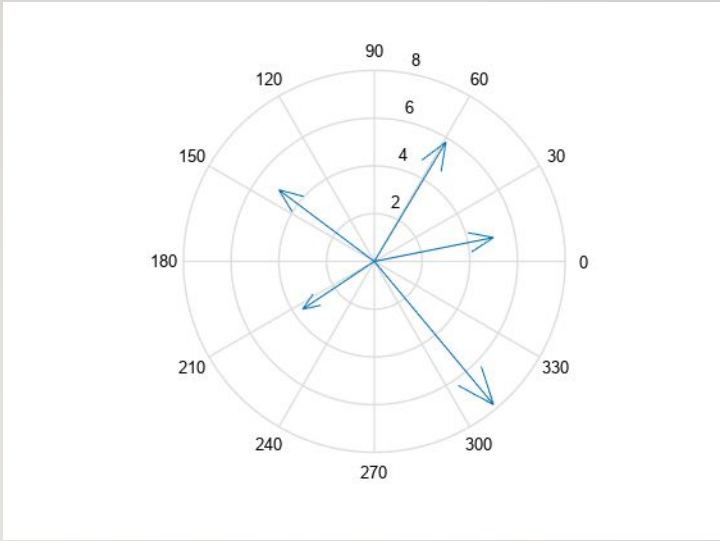


# COMPASS PLOT

- compass(U,V) plots arrows originating from the point (0, 0). Specify the direction of arrows using the Cartesian coordinates U and V, with U indicating the x-coordinates and V indicating the y-coordinates. The number of arrows matches the number of elements in U.
- The compass function plots arrows on a circular grid with theta-axis and r-axis tick labels within an Axes object. Therefore, the coordinates you specify do not match the labels displayed on the plot.
- compass(Z) plots arrows using the real and imaginary parts of the complex values specified by Z, with the real part indicating the x-coordinates and the imaginary part indicating the y-coordinates. This syntax is equivalent to compass(real(Z),imag(Z)).
- compass(___,LineSpec) sets the line style, marker symbol, and color for the arrows.
- compass(ax,___) plots arrows in the specified axes instead of the current axes.
- c = compass(___) returns a vector of Line objects. This syntax is useful for controlling the appearance of arrows.

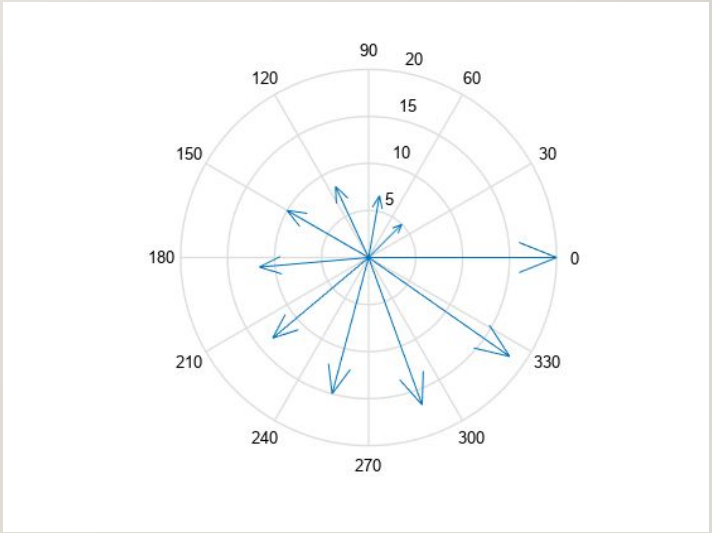
Create a compass plot by specifying the Cartesian coordinates of each arrow.

```
u = [5 3 -4 -3 5];

v = [1 5 3 -2 -6];

compass(u,v)
```




Convert them to Cartesian coordinates using the pol2cart function. Then, create the plot.

```
th = linspace(pi/4,2*pi,10);

r = linspace(5,20,10);

[u,v] = pol2cart(th,r);

compass(u,v)
```

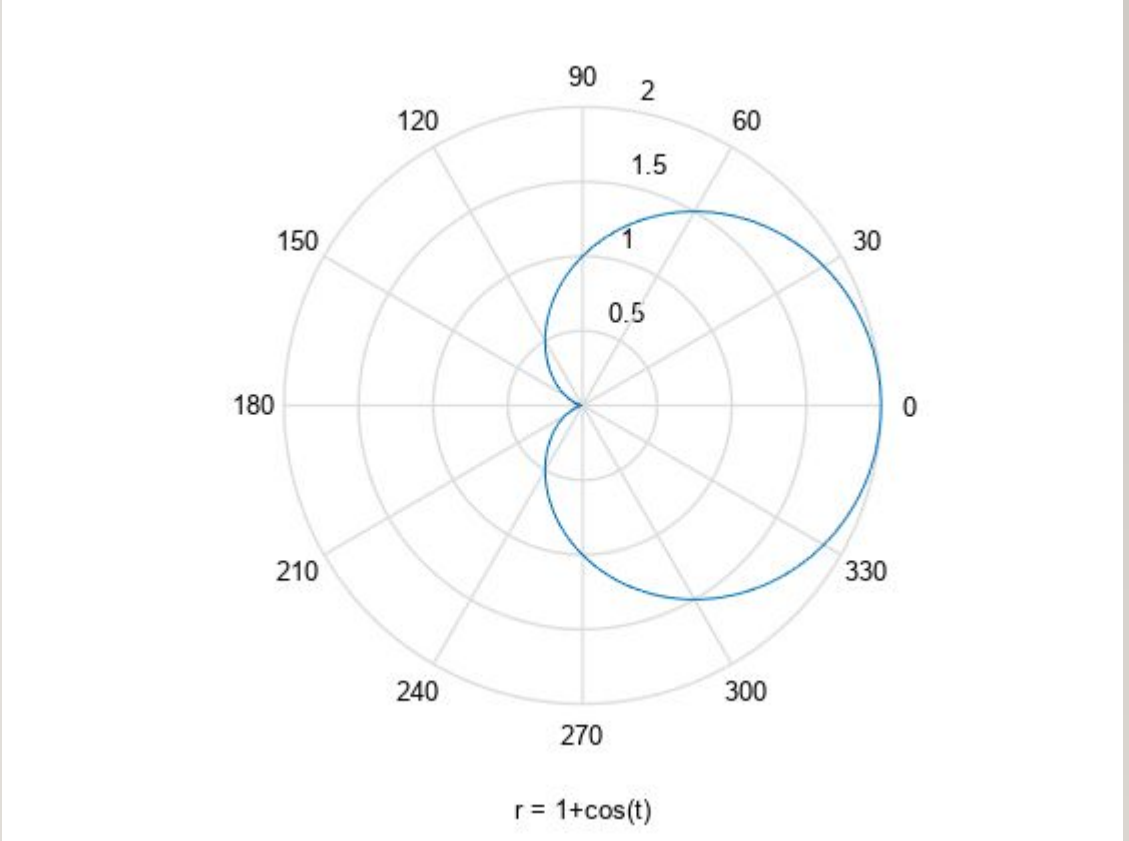


# EZPOLAR PLOT

- ezpolar(fun) plots the polar curve rho = fun(theta) over the default domain 0 < theta < 2π.
- fun can be a function handle, a character vector, or a string (see the Tips section).
- ezpolar(fun,[a,b]) plots function for a < theta < b.
- ezpolar(axes_handle,...) plots into the axes with handle axes_handle instead of the current axes (gca).
- h = ezpolar(...) returns the handle to a line object in h.

Plot the function 1+cos(t) over the domain [0,2π].

```
figure

ezpolar('1+cos(t)')
```



# TWO-DIMENSIONAL PLOTS: PLOTTING FUNCTIONS

- Easy to plot: plot a function directly, without the necessity of creating intermediate data arrays.
  - ezplot(fun);
  - ezplot(fun, [xmin xmax]);
  - ezplot(fun, [xmin xmax], figure);
- Example:

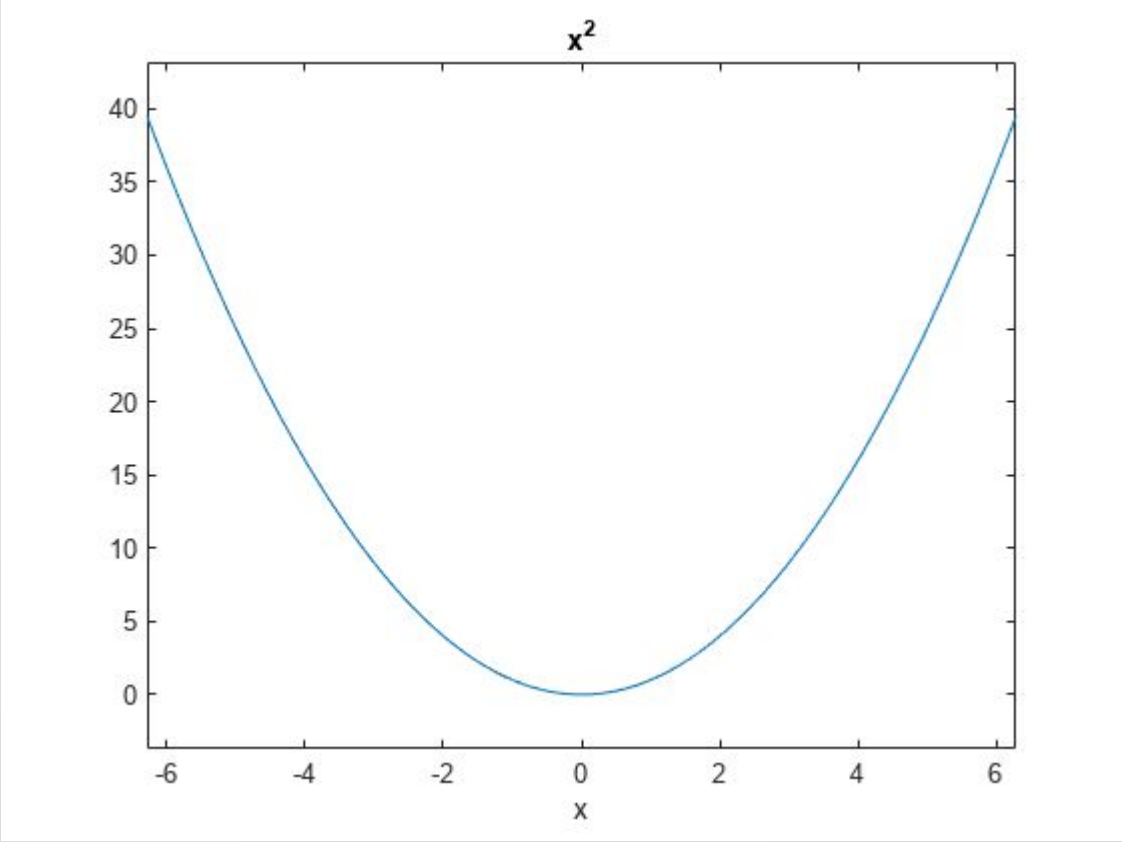
```
ezplot('sin(x)/x',[-4*pi 4*pi]);
title('Plot of sinx/x');
grid on;
```

- ezplot(f) plots the curve defined by the function y = f(x) over the default interval [-2π 2π] for x.
- ezplot automatically adds a title and axis labels to the plot.
- ezplot(f,xinterval) plots over the specified interval. Specify the interval as a two-element vector of the form [xmin xmax].
- ezplot(f2) plots the curve defined by the implicit function 0 = f2(x,y) over the default interval [-2π 2π] for x and y.
- ezplot(f2,xyinterval) plots over the specified interval. To use the same interval for both x and y, specify xyinterval as a two-element vector of the form [min max]. To use different intervals, specify a four-element vector of the form [xmin xmax ymin ymax].
- ezplot(funx,funy) plots the parametrically defined planar curve defined by x = funx(u) and y = funy(u) over the default interval [0 2π] for u.

Plot the explicit function x^2, over the domain [−2π,2π].

ezplot('x^2')

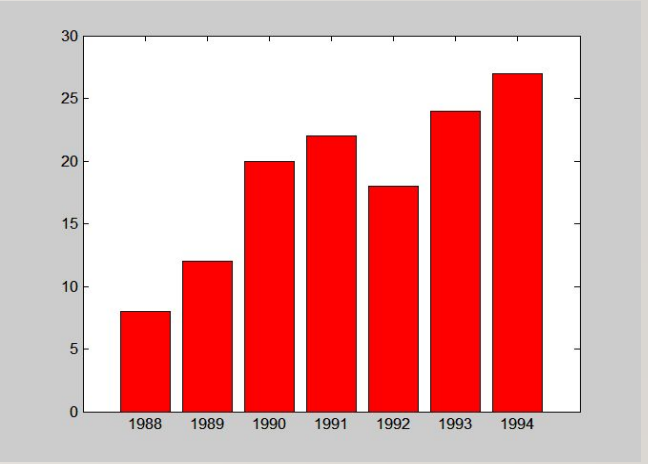
The default domain is [−2π,2π].



# TWO-DIMENSIONAL PLOTS: EXAMPLES

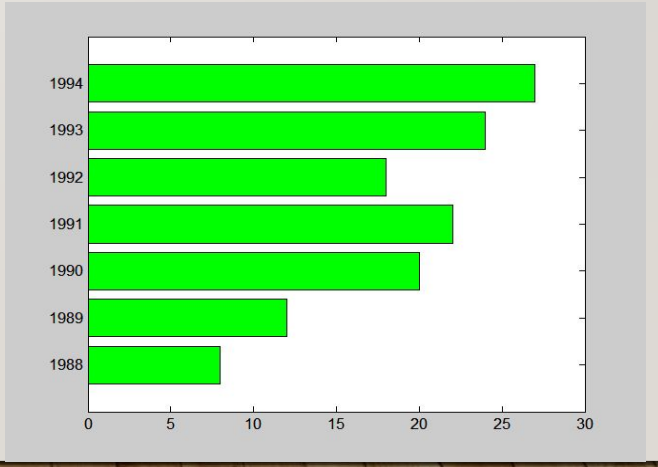
- Vertical bar plot

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




- Horizontal bar plot

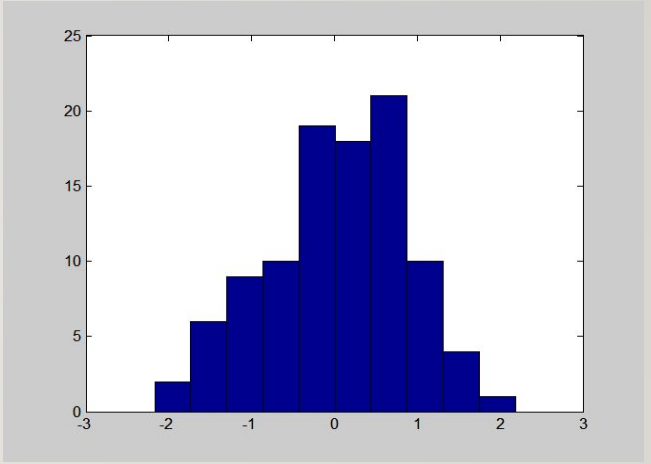
```
y = 1988:1994;
s = [ 8 12 20 22 18 24 27 ];
barh(y,s,'g');
```



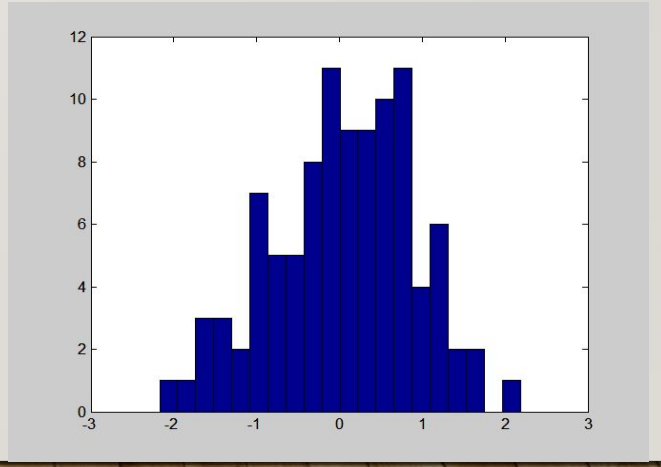
# TWO-DIMENSIONAL PLOTS: EXAMPLES

- Histogram

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




```
hist(x,20);
```



# TWO-DIMENSIONAL PLOTS: EXAMPLES

- Error bar plot

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