Matlab 各种类型图表演示 (2014)

说明

强大的绘图功能是 Matlab 的特点之一，Matlab 提供了十分丰富的的绘图函数，用户不需要过多的考虑绘图的细节，只需要给出一些基本参数就能得到所需图形。尤其是 Matlab 2014a/2014b 采用了全新的绘图系统,图表颜色配置,输出图片质量大大提高. 这里收集了一些 Matlab 绘图演示实例,从简单的 plot 到复杂的动画都有.

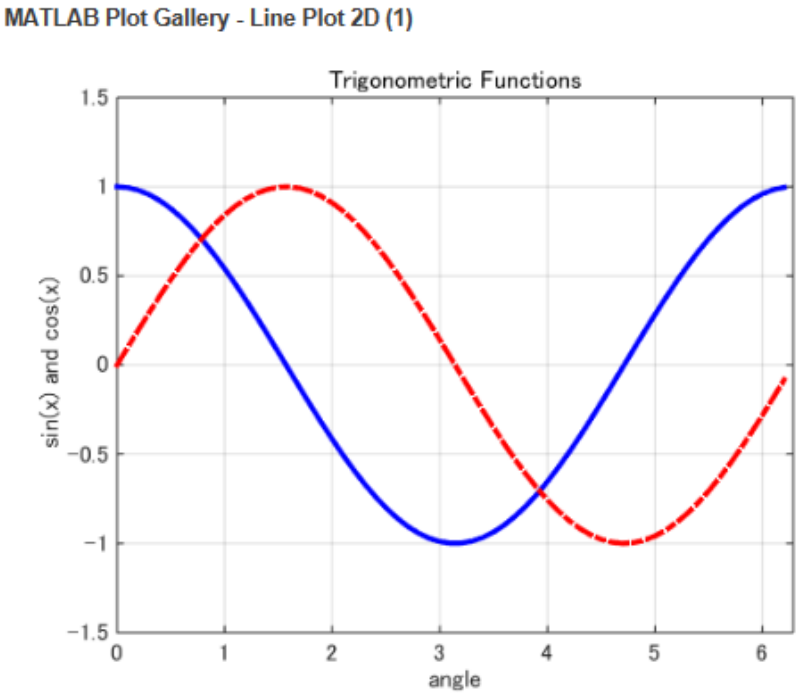
大部分例子来自 Mathworks 官网, 版权归 Mathworks 公司. 本人也将逐步加入自己编的绘图例子, 欢迎持续关注,交流 [QQ: 2953212138]

目录

基本绘图

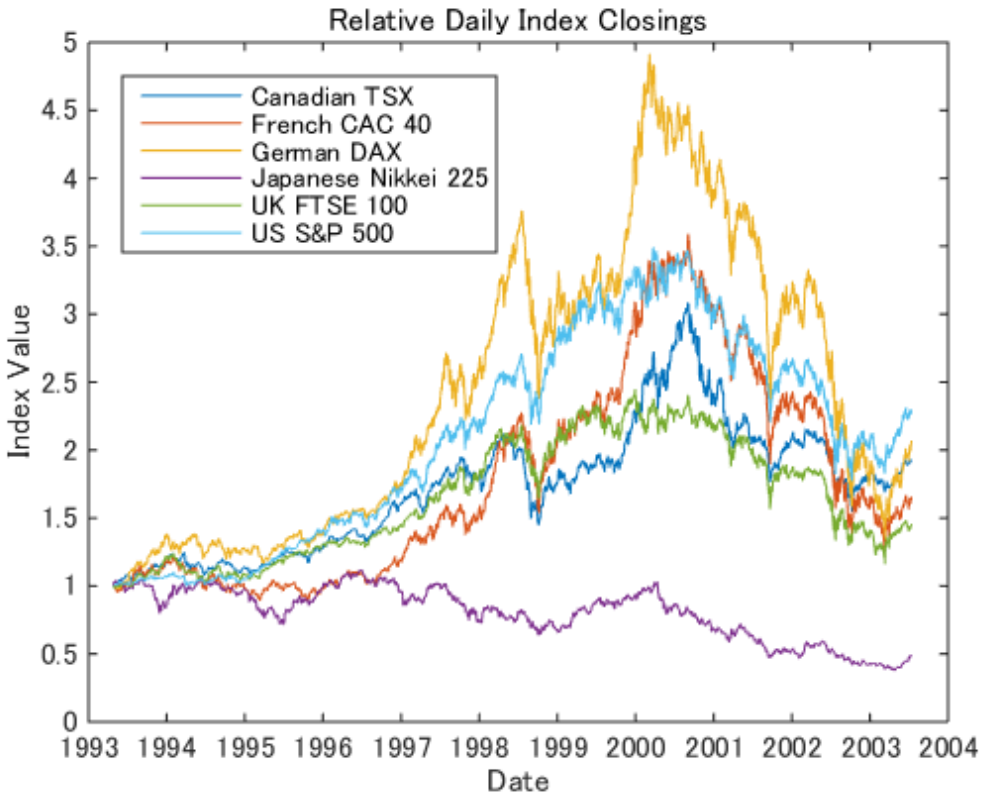
plot (1)

```
% Define values for x, y1, and y2
x    = 0: .1 : 2*pi;
y1 = cos(x);
y2 = sin(x);

% Plot y1 vs. x (blue, solid) and y2 vs. x (red, dashed)
figure
plot(x, y1, 'b', x, y2, 'r-.', 'LineWidth', 2)

% Turn on the grid
grid on

% Set the axis limits
axis([0 2*pi -1.5 1.5])

% Add title and axis labels
title('Trigonometric Functions')
xlabel('angle')
ylabel('sin(x) and cos(x)')
```



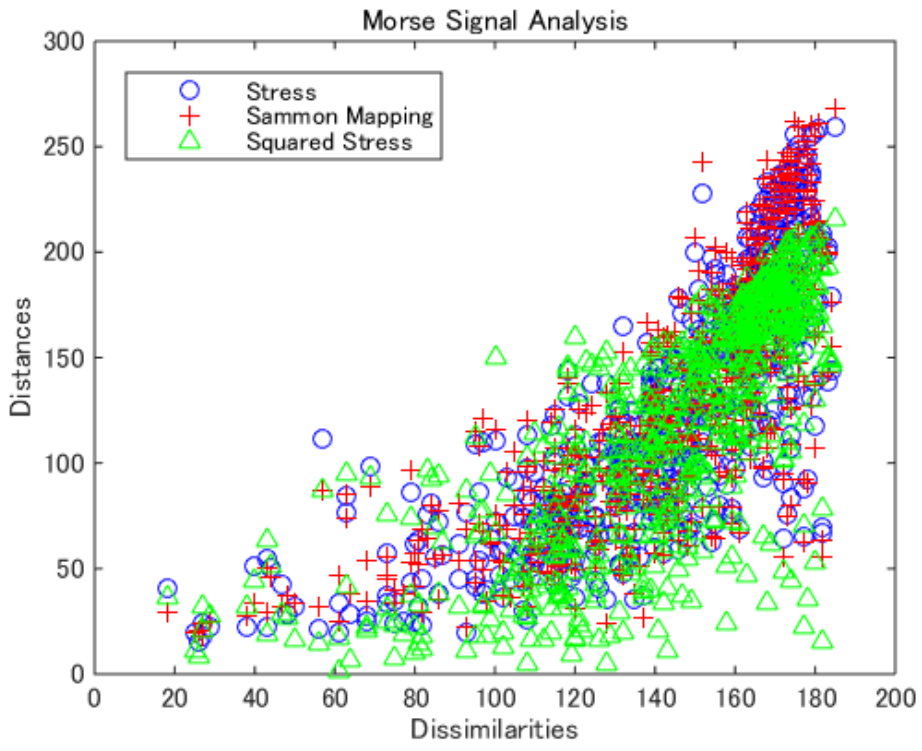
plot (2)

```
% Load data for the stock indices
load IndexData dates values series

% Plot the stock index values versus time
figure
plot(dates, values)

% Use dateticks for the x axis
datetick('x')

% Add title and axis labels
xlabel('Date')
ylabel('Index Value')
title('Relative Daily Index Closings')

% Add a legend in the top, left corner
legend(series, 'Location', 'NorthWest')
```



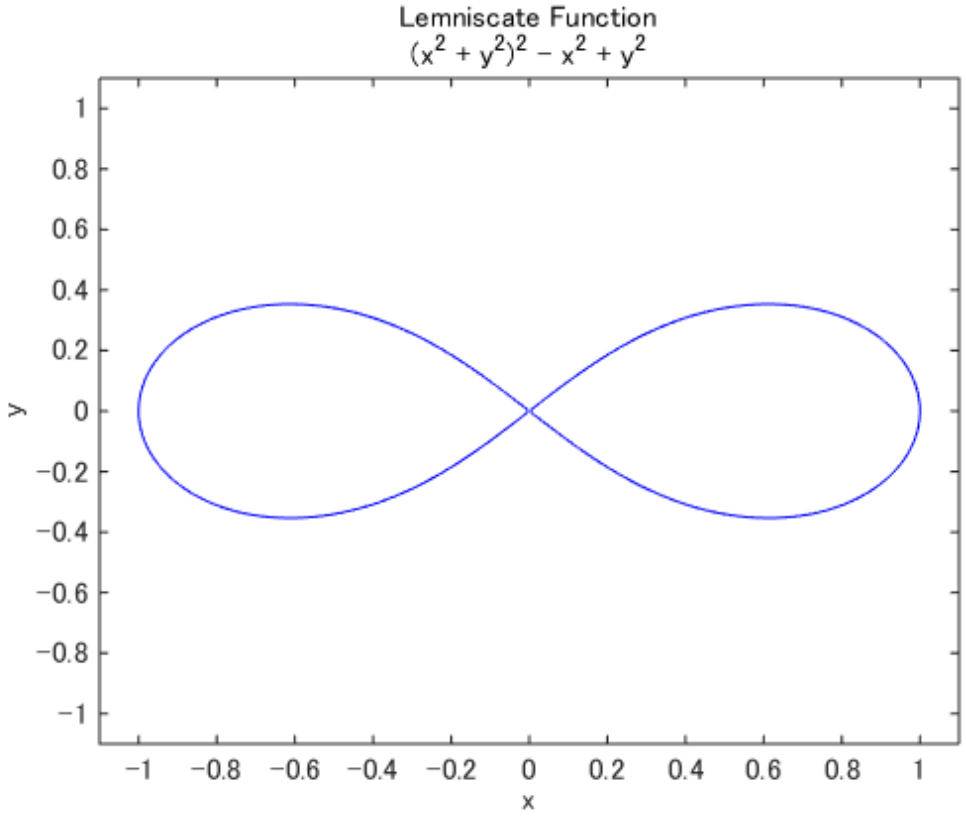
plot (3)

```
% Load Morse data
load MDdata dissimilarities dist1 dist2 dist3

% Plot the first set of data in blue
figure
plot(dissimilarities, dist1, 'bo')
hold on

% Plot the second set of data in red
plot(dissimilarities, dist2, 'r+')

% Plot the third set of data in green
plot(dissimilarities, dist3, 'g^')

% Add title and axis labels
title('Morse Signal Analysis')
xlabel('Dissimilarities')
ylabel('Distances')

% Add a legend
legend({'Stress', 'Sammon Mapping', 'Squared Stress'}, ...
       'Location', 'NorthWest')
```



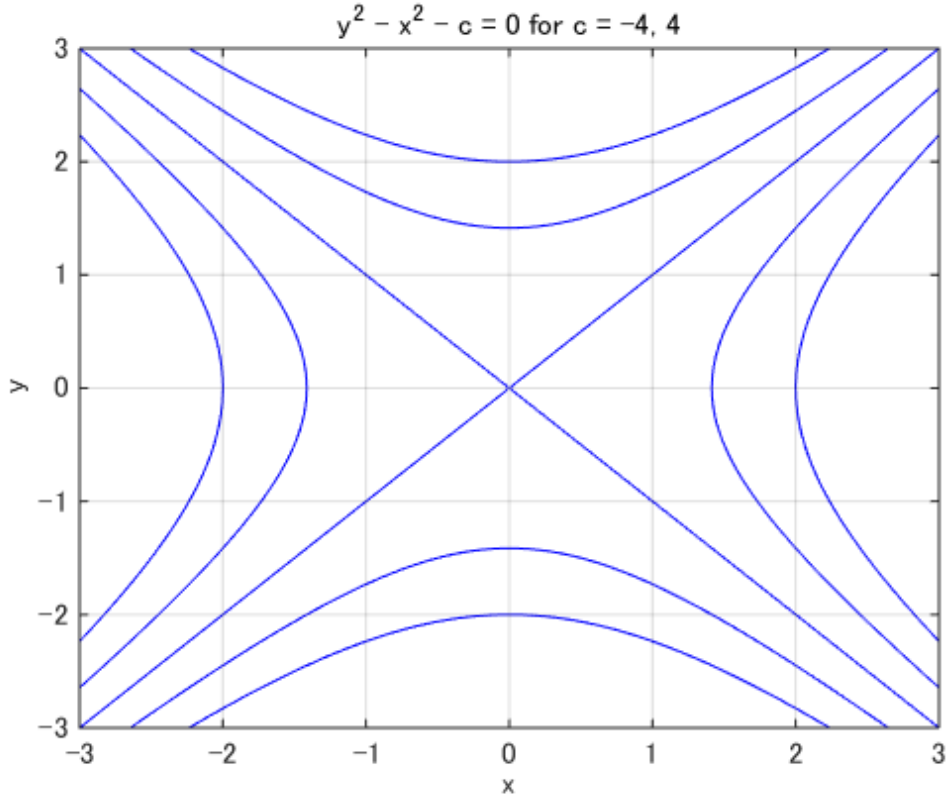
ezplot 1

```
% Create the plot using the lemniscate function f(x,y) = (x^2 + y^2)^2 - x^2 + y^2
figure
ezplot('(x^2 + y^2)^2 - x^2 + y^2', [-1.1, 1.1], [-1.1, 1.1])

% Adjust the colormap to plot the function in blue
colormap([0 0 1])

% Add a multi-line title
title({'Lemniscate Function', '(x^2 + y^2)^2 - x^2 + y^2'})
```



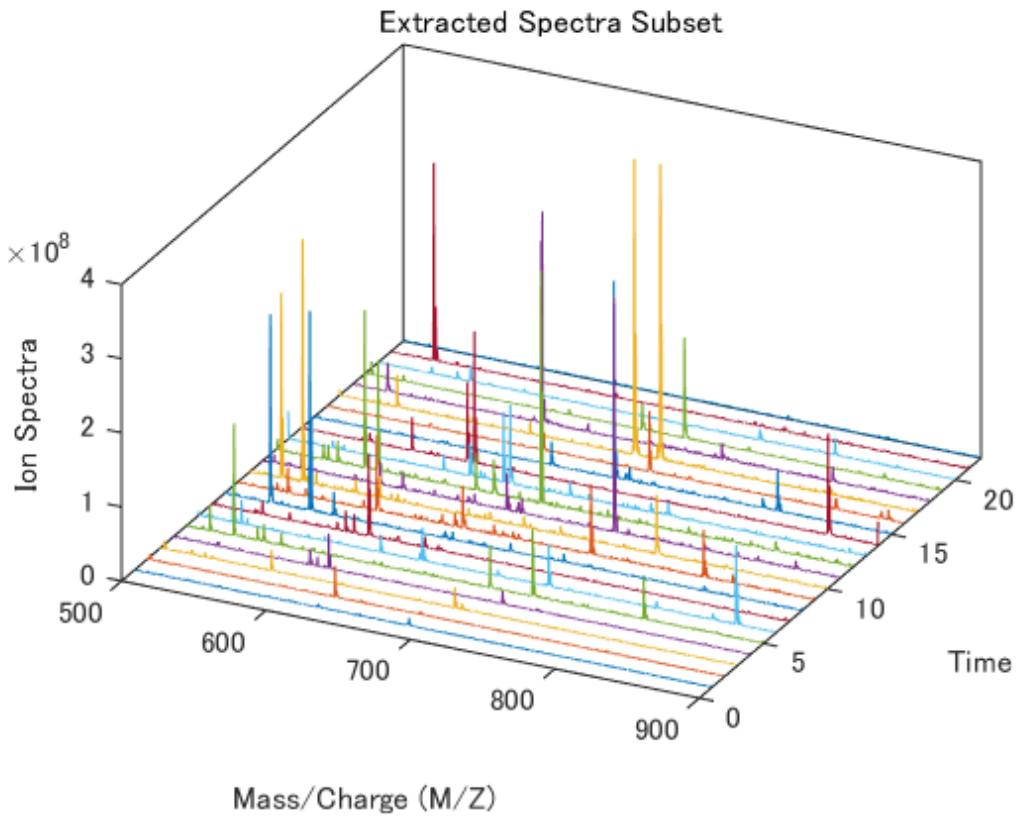
ezplot 2

```
% Create a series of lines for the function f(x,y) = y^2 - x^2 + c
figure
ezplot('y^2 - x^2 + 4', [-3 3], [-3 3])
hold on
ezplot('y^2 - x^2 + 2', [-3 3], [-3 3])
ezplot('y^2 - x^2',          [-3 3], [-3 3])
ezplot('y^2 - x^2 - 2', [-3 3], [-3 3])
ezplot('y^2 - x^2 - 4', [-3 3], [-3 3])

% Adjust the colormap to plot the function in blue
colormap([0 0 1])
grid on

% Add title
title('y^2 - x^2 - c = 0 for c = -4, 4')
```



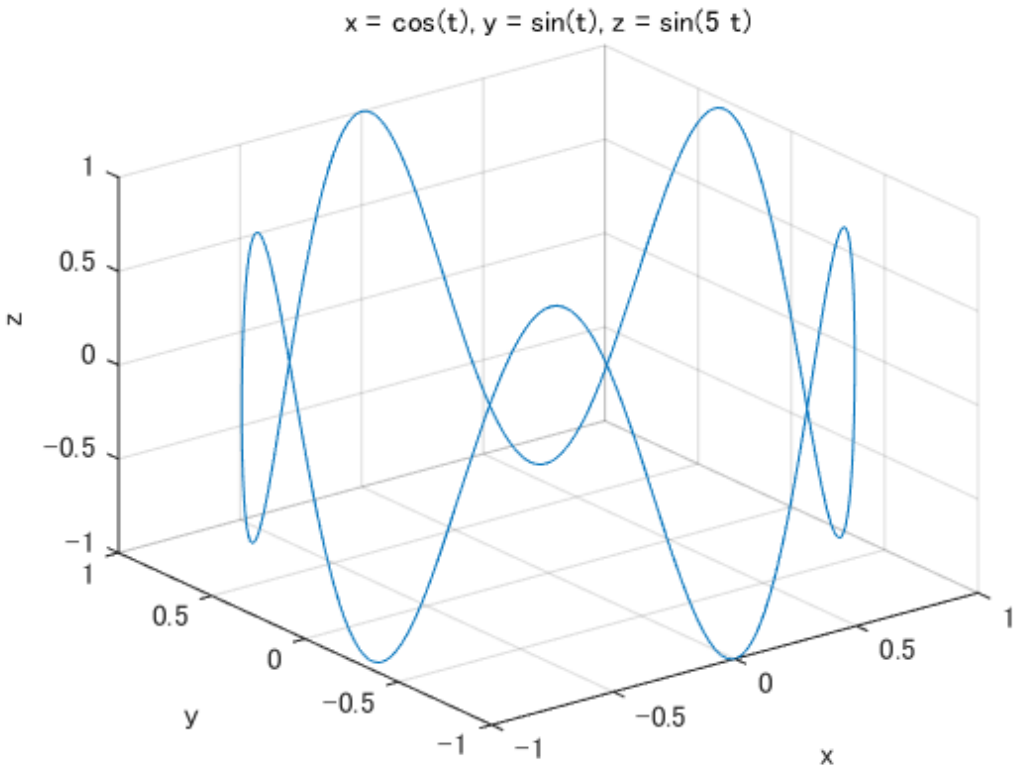
plot3

```
% Load the spectra data
load spectraData masscharge time spectra

% Create the 3D plot
figure
plot3(masscharge, time, spectra)
box on

% Set the viewing angle and the axis limits
view(26, 42)
axis([500 900 0 22 0 4e8])

% Add title and axis labels
xlabel('Mass/Charge (M/Z)')
ylabel('Time')
zlabel('Ion Spectra')
title('Extracted Spectra Subset')
```



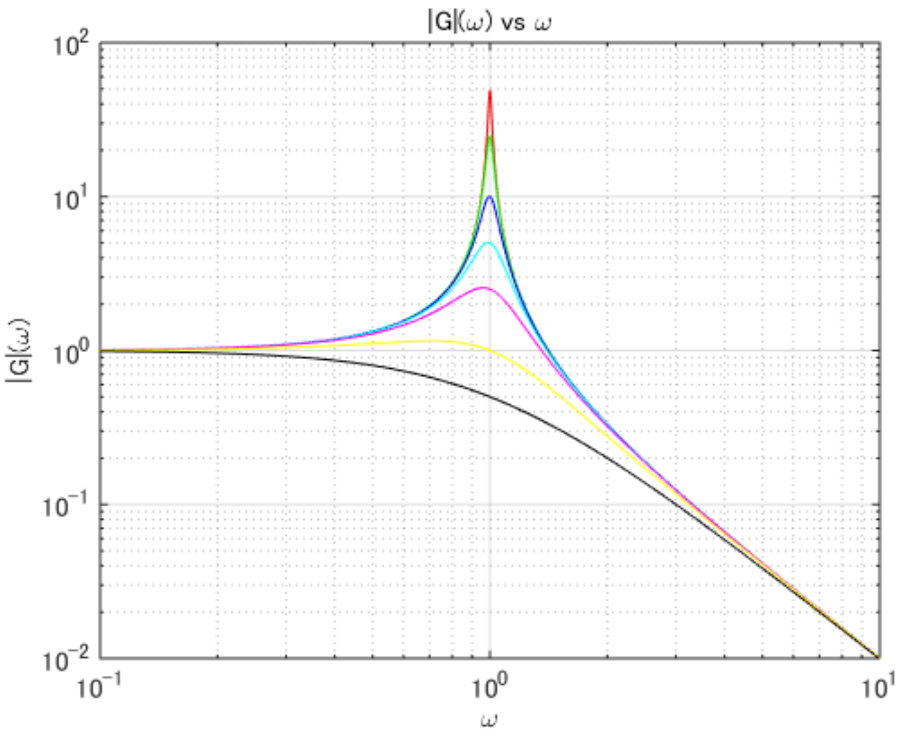
ezplot3

```
% Create the plot using the parametric functions
% x = cost(t), y = sin(t), and z = sin(5*t) for -pi < t < pi
figure
ezplot3('cos(t)', 'sin(t)', 'sin(5*t)', [-pi pi])
```



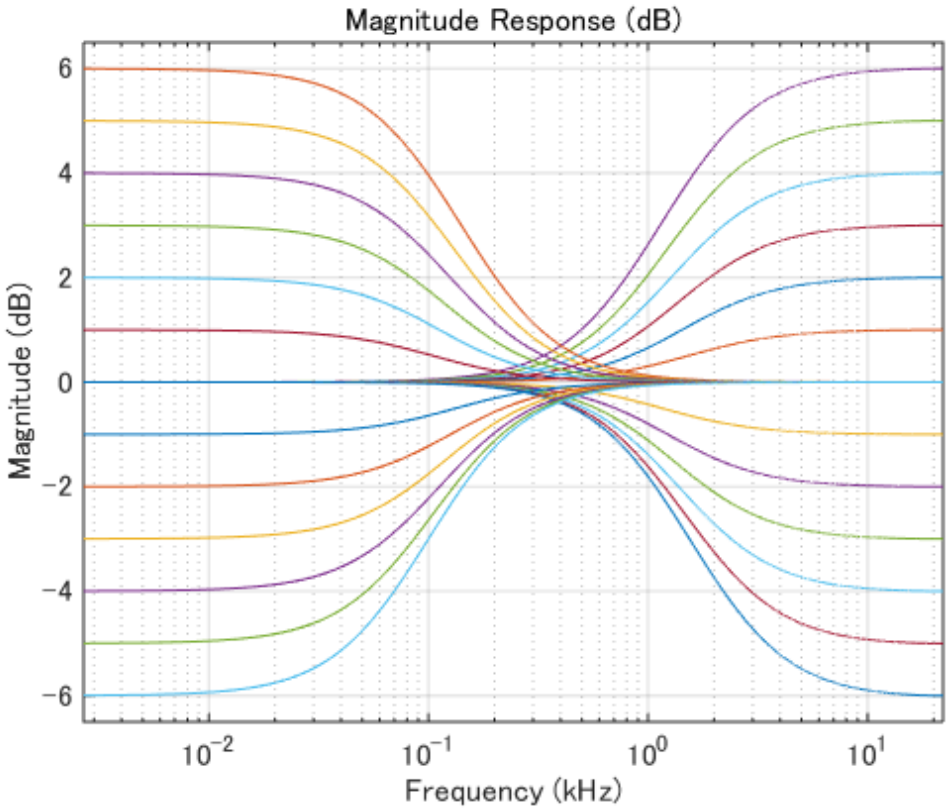
loglog

```
% Create a set of values for the damping factor
zeta = [0.01 .02 0.05 0.1 .2 .5 1];

% Define a color for each damping factor
colors = ['r' 'g' 'b' 'c' 'm' 'y' 'k'];

% Create a range of frequency values equally spaced logarithmically
w = logspace(-1, 1, 1000);

% Plot the gain vs. frequency for each of the seven damping factors
figure
for i = 1:7
    a = w.^2 - 1;
    b = 2*w*zeta(i);
    gain = sqrt(1./(a.^2 + b.^2));
    loglog(w, gain, 'color', colors(i))
    hold on
end

% Set the axis limits
axis([0.1 10 0.01 100])

% Add a title and axis labels
title('|G|(\omega) vs \omega')
xlabel('\omega')
ylabel('|G|(\omega)')

% Turn the grid on
grid on
```



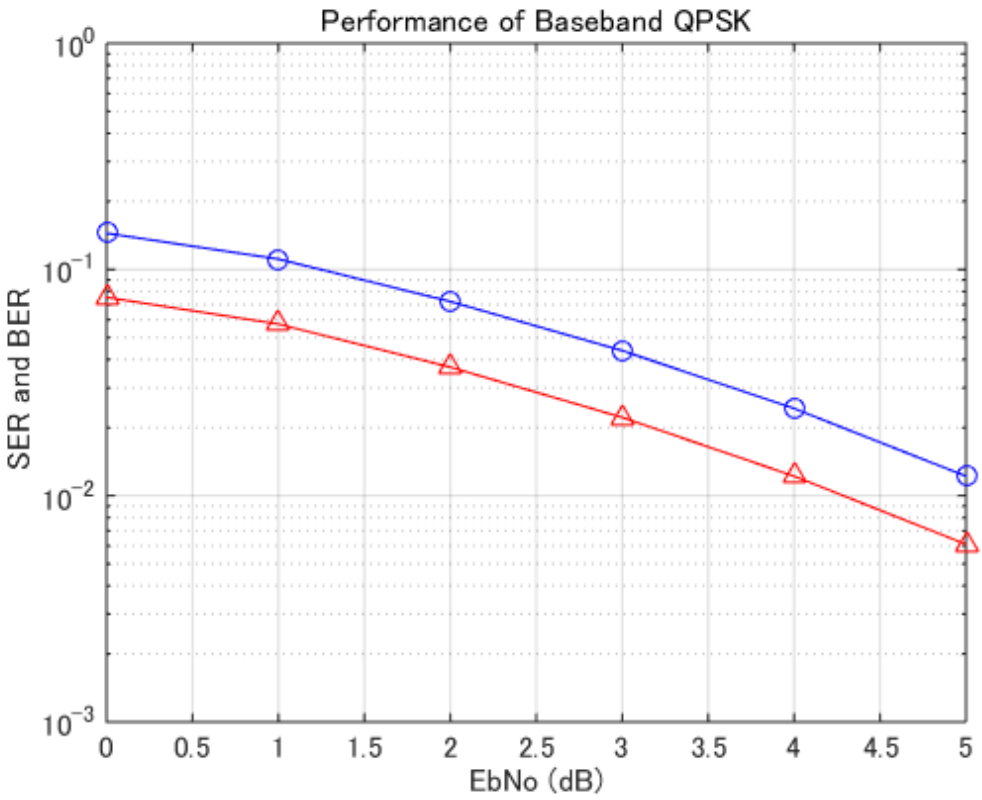
Semilogx

```
% Load the response data
load responseData frequency magnitude

% Create an x-axis semilog plot using the semilogx function
figure
semilogx(frequency, magnitude)

% Set the axis limits and turn on the grid
axis([min(frequency) max(frequency) -6.5 6.5])
grid on

% Add title and axis labels
title('Magnitude Response (dB)')
xlabel('Frequency (kHz)')
ylabel('Magnitude (dB)')
```



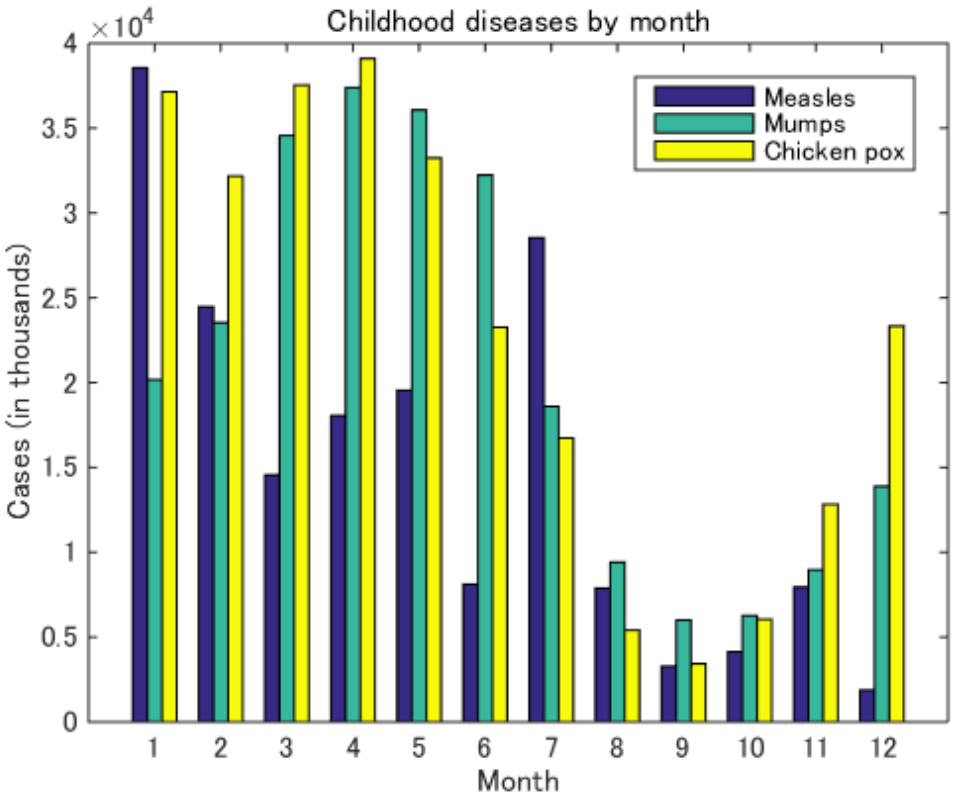
Semilogy

```
% Create some data
eb = 0:5;
SER = [0.1447 0.1112 0.0722 0.0438 0.0243 0.0122];
BER = [0.0753 0.0574 0.0370 0.0222 0.0122 0.0061];

% Create a y-axis semilog plot using the semilogy function
% Plot SER data in blue and BER data in red
figure
semilogy(eb, SER, 'bo-')
hold on
semilogy(eb, BER, 'r^-')

% Turn on the grid
grid on

% Add title and axis labels
title('Performance of Baseband QPSK')
xlabel('EbNo (dB)')
ylabel('SER and BER')
```



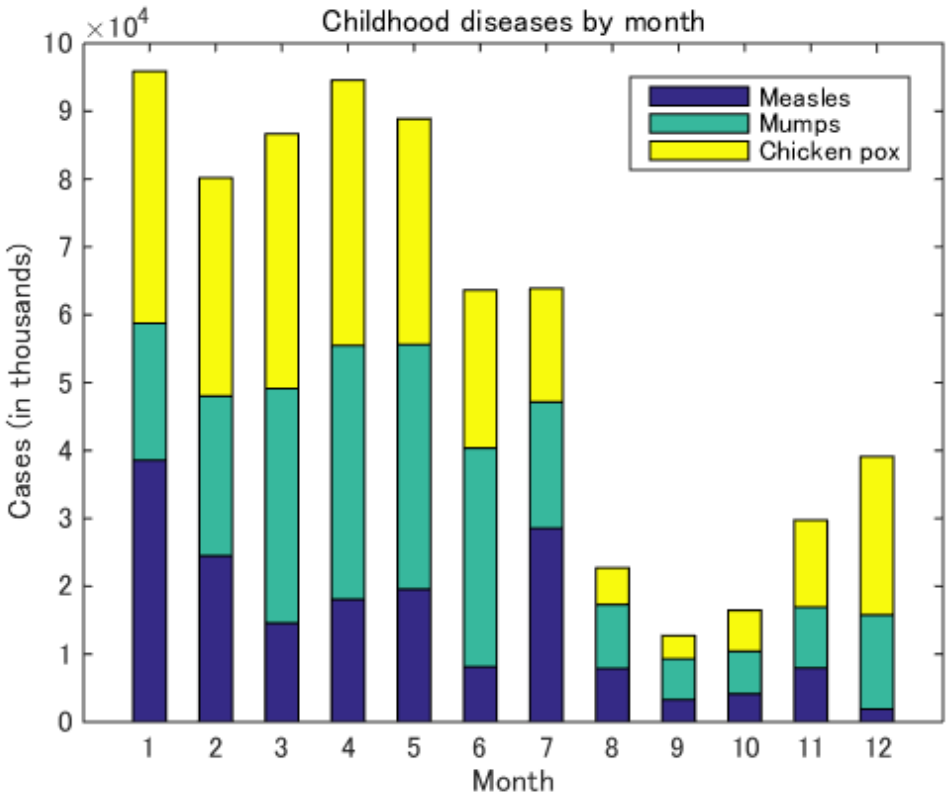
Vertical Bar

```
% Create data for childhood disease cases
measles = [38556 24472 14556 18060 19549 8122 28541 7880 3283 4135 7953 1884];
mumps = [20178 23536 34561 37395 36072 32237 18597 9408 6005 6268 8963 13882];
chickenPox = [37140 32169 37533 39103 33244 23269 16737 5411 3435 6052 12825 23332];

% Create a vertical bar chart using the bar function
figure
bar(1:12, [measles' mumps' chickenPox'], 1)

% Set the axis limits
axis([0 13 0 40000])
set(gca, 'XTick', 1:12)

% Add title and axis labels
title('Childhood diseases by month')
xlabel('Month')
ylabel('Cases (in thousands)')

% Add a legend
legend('Measles', 'Mumps', 'Chicken pox')
```



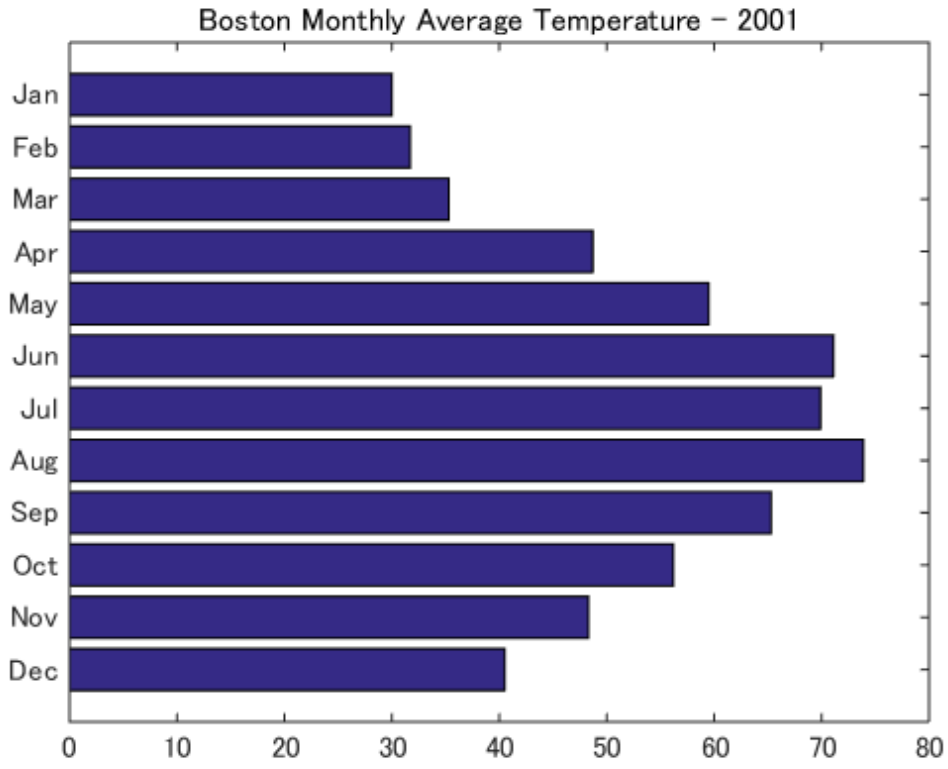
bar

```
% Create data for childhood disease cases
measles = [38556 24472 14556 18060 19549 8122 28541 7880 3283 4135 7953 1884]';
mumps = [20178 23536 34561 37395 36072 32237 18597 9408 6005 6268 8963 13882]';
chickenPox = [37140 32169 37533 39103 33244 23269 16737 5411 3435 6052 12825 23332]';

% Create a stacked bar chart using the bar function
figure
bar(1:12, [measles mumps chickenPox], 0.5, 'stack')

% Adjust the axis limits
axis([0 13 0 100000])
set(gca, 'XTick', 1:12)

% Add title and axis labels
title('Childhood diseases by month')
xlabel('Month')
ylabel('Cases (in thousands)')

% Add a legend
legend('Measles', 'Mumps', 'Chicken pox')
```



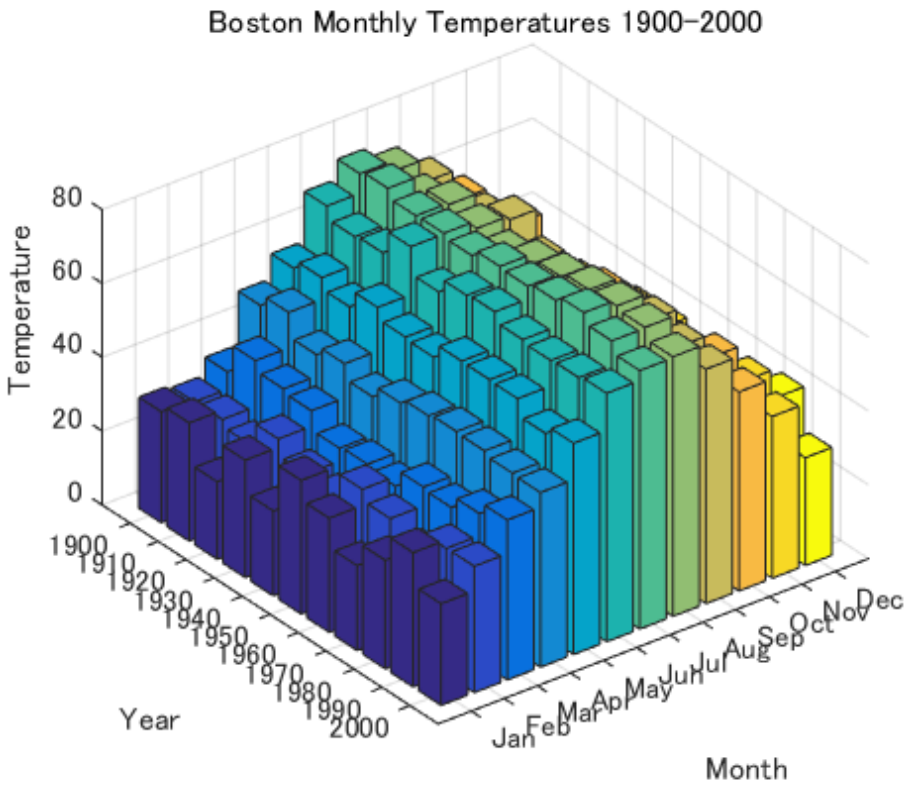
Horizontal Bar

```
% Create the data for the temperatures and months
temperatures = [40.5 48.3 56.2 65.3 73.9 69.9 71.1 59.5 48.7 35.3 31.7 30.0];
months = {'Dec', 'Nov', 'Oct', 'Sep', 'Aug', 'Jul', 'Jun', 'May', 'Apr', 'Mar', 'Feb', 'Jan'};

% Plot the temperatures on a horizontal bar chart
figure
barh(temperatures)

% Set the axis limits
axis([0 80 0 13])

% Add a title
title('Boston Monthly Average Temperature - 2001')

% Change the Y axis tick labels to use the months
set(gca, 'YTick', 1:12)
set(gca, 'YTickLabel', months)
```



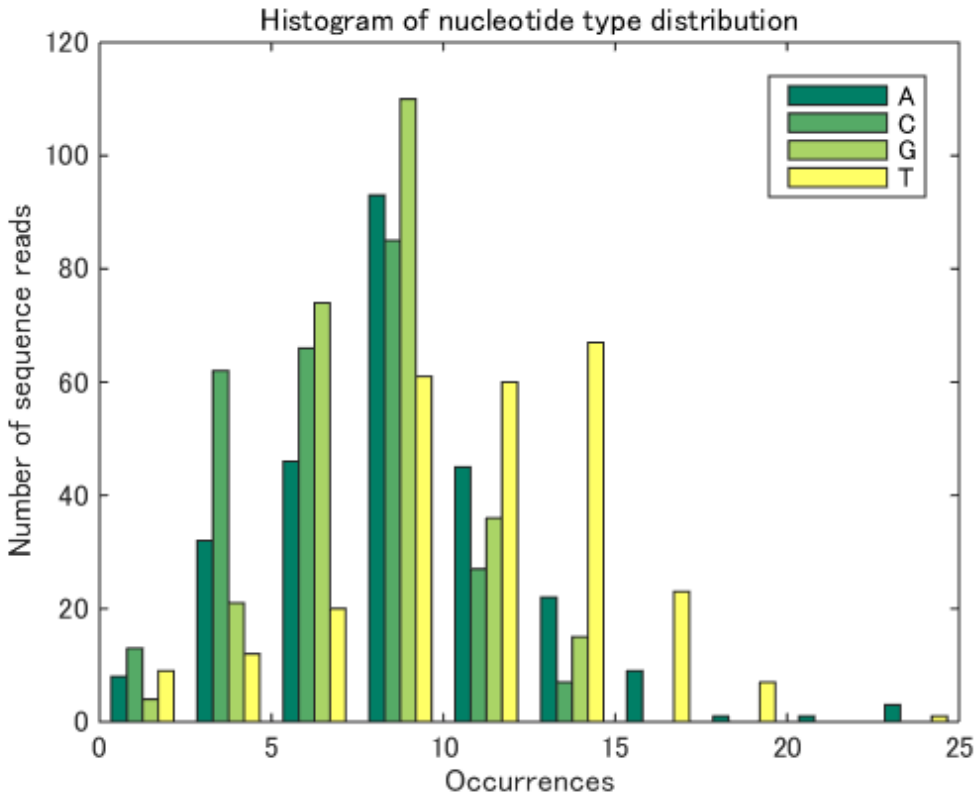
bar3

```
% Load monthly temperature data
load MonthlyTemps temperatures months years

% Create the 3D bar chart
figure
bar3(temperatures)
axis([0 13 0 12 0 80])

% Add title and axis labels
title('Boston Monthly Temperatures 1900-2000')
xlabel('Month')
ylabel('Year')
zlabel('Temperature')

% Change the x and y axis tick labels
set(gca, 'XTickLabel', months)
set(gca, 'YTickLabel', years)
```



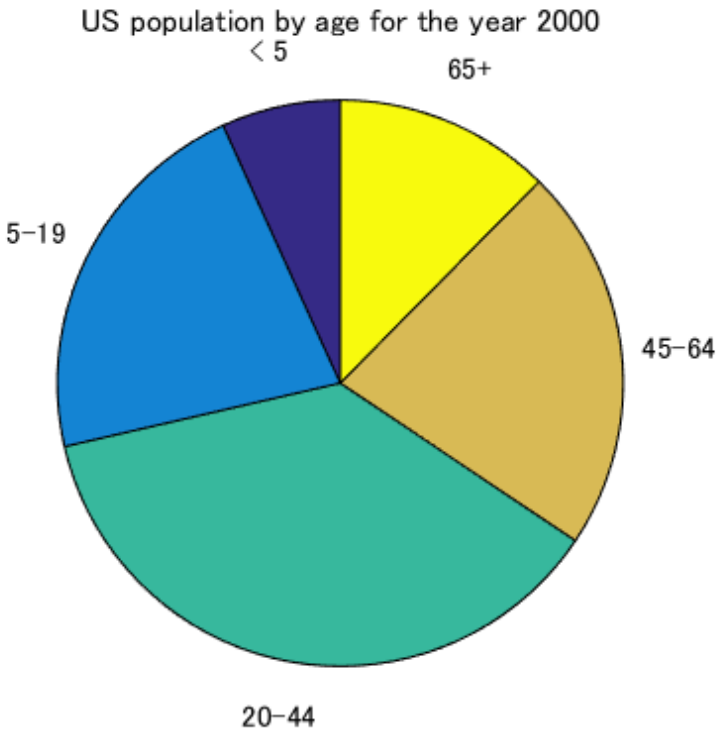
hist

```
% Load nucleotide data
load nucleotideData ncount

% Create the histogram using the hist function
figure
hist(ncount)
colormap summer

% Add a legend
legend('A', 'C', 'G', 'T')

% Add title and axis labels
title('Histogram of nucleotide type distribution')
xlabel('Occurrences')
ylabel('Number of sequence reads')
```



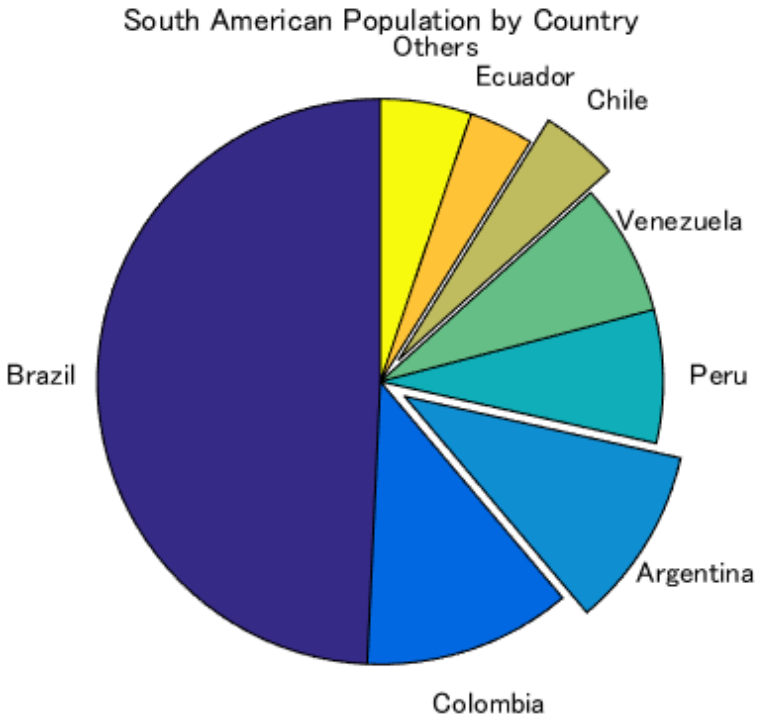
pie (1)

```
% Load the data for US population by age 1860-2000
load populationAge population groups

% Get the population for each age group for the year 2000
age2000 = population(15, :);

% Create a pie chart using the pie function -- use age groups as labels
figure
pie(age2000, groups)

% Add title
title('US population by age for the year 2000')
```



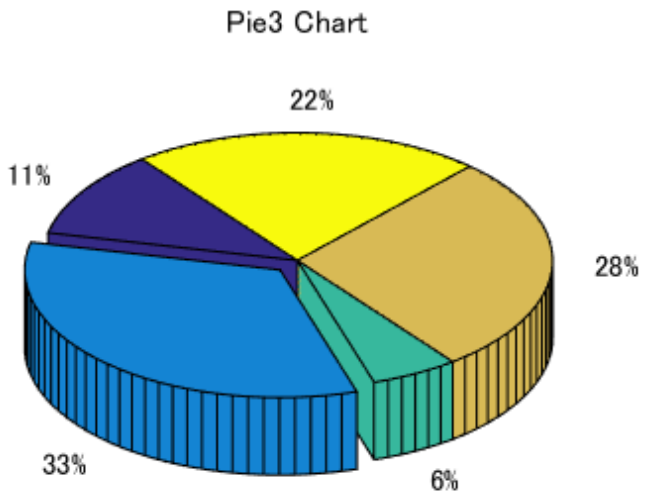
pie (2)

```
% Load the data for South American populations
load SouthAmericaPopulations populations countries

% Calculate the total populations and percentage by country
total = sum(populations);
percent = populations/total;

% Create a pie chart with sections 3 and 6 exploded
figure
explode = [0 0 1 0 0 1 0 0];
pie(percent, explode, countries)

% Add title
title('South American Population by Country')
```



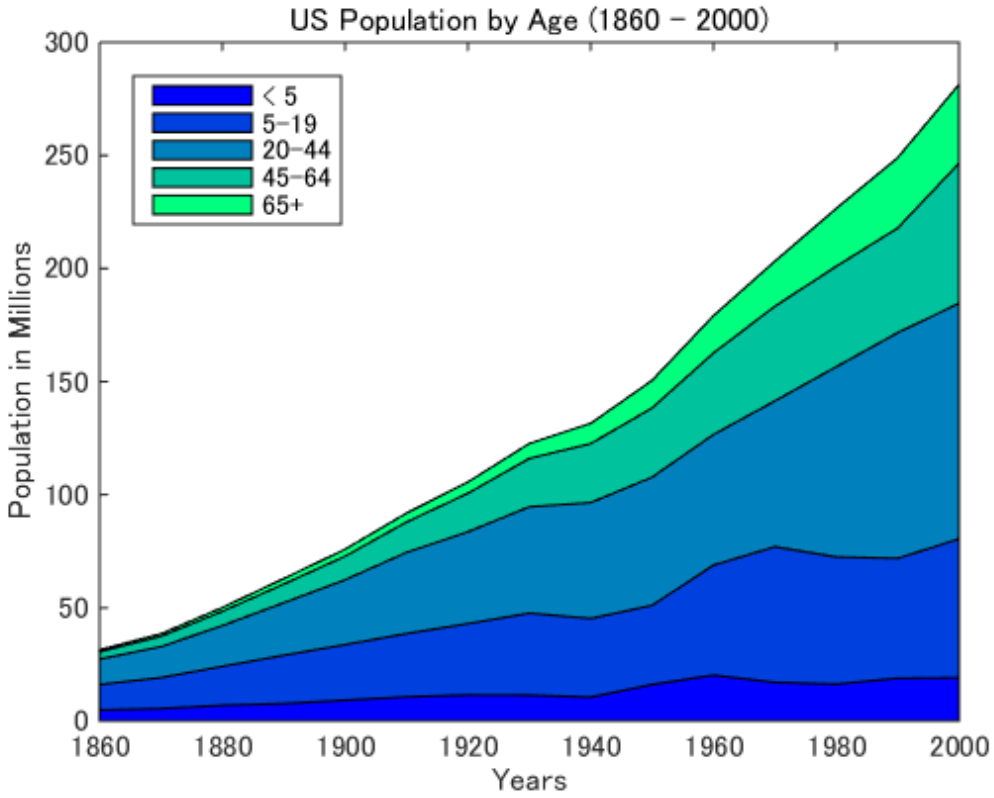
pie3

```
% Create some data
x = [1 3 0.5 2.5 2];

% Create a 3D pie chart using the pie3 function
figure
explode = [0 1 0 0 0];
pie3(x, explode)

% Add a title
title('Pie3 Chart')
```



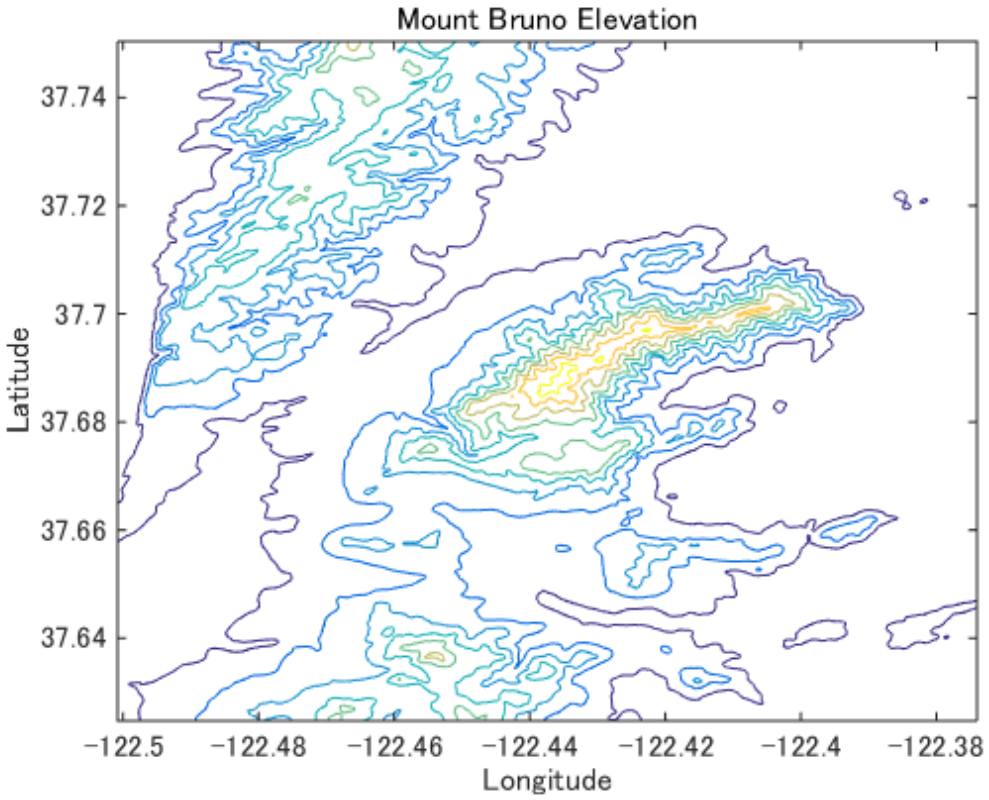
Area

```
% Load population data
load PopulationAge years population groups

% Create the area plot using the area function
figure
area(years, population/1000000)
colormap winter

% Add a legend
legend(groups, 'Location', 'NorthWest')

% Add title and axis labels
title('US Population by Age (1860 - 2000)')
xlabel('Years')
ylabel('Population in Millions')
```



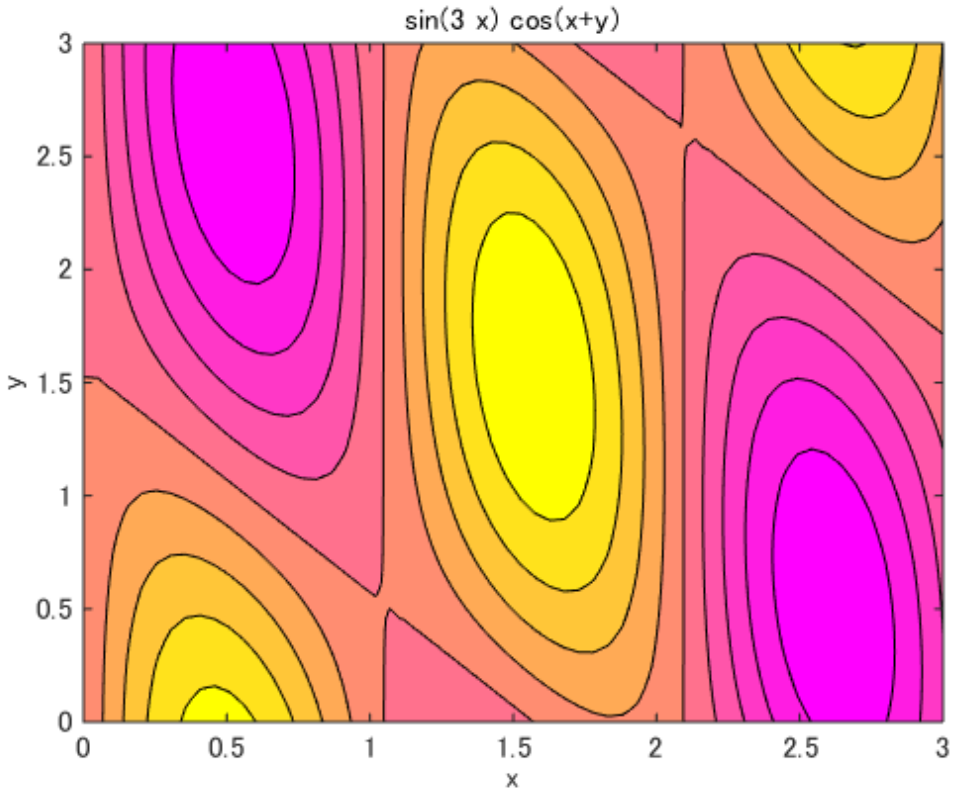
contour

```
% Load position and elevation data for Mount Bruno
load mtBruno Longitude Latitude Elevation

% Create a contour plot with 8 contour levels
figure
contour(Longitude, Latitude, Elevation, 8)

% Add title and exis labels
title('Mount Bruno Elevation')
xlabel('Longitude')
ylabel('Latitude')
```



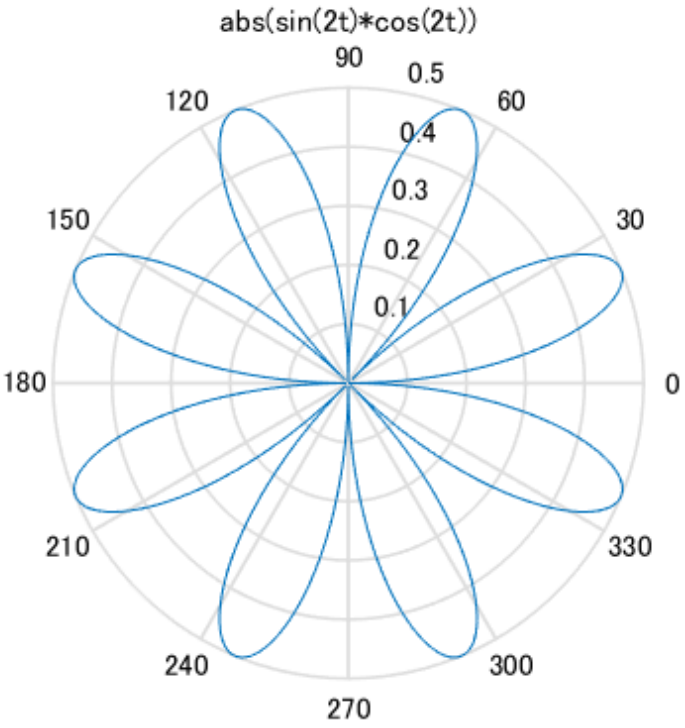
ezcontourf

```
% Create the contour plot using the function f(x,y) = sin(3*x)*cos(x+y)
figure
ezcontourf('sin(3*x)*cos(x+y)', [0, 3, 0, 3])

% Change the default colormap to 'spring'
colormap('spring')
```



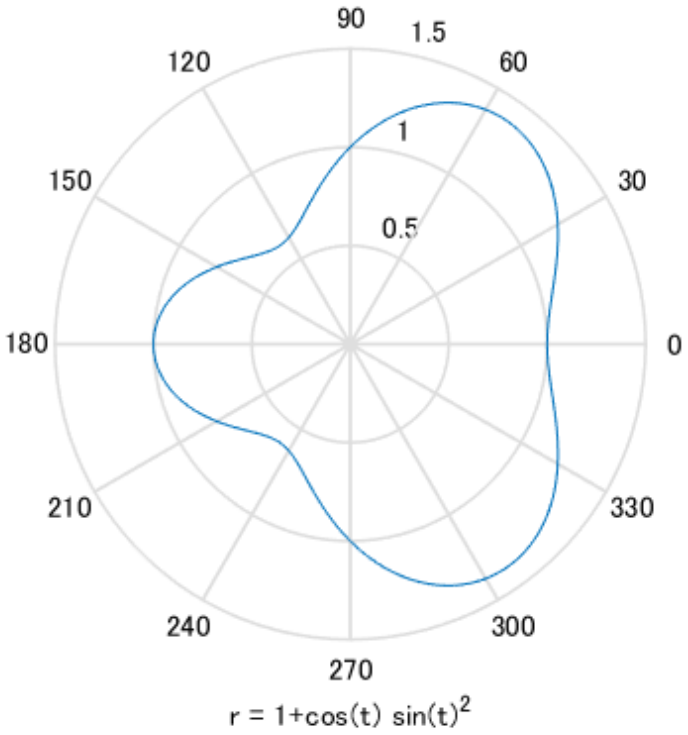
polar

```
% Create data for the function
t = 0:0.01:2*pi;
r = abs(sin(2*t).*cos(2*t));

% Create a polar plot using the function polar
figure
polar(t, abs(sin(2*t).*cos(2*t)))

% Add a title
title('abs(sin(2t)*cos(2t))')
```



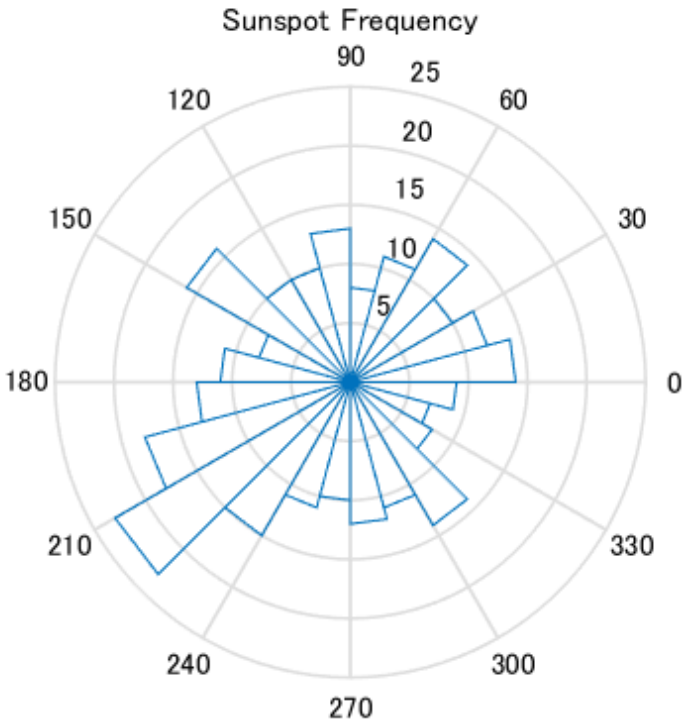
ezpolar

```
% Create the plot using the function r(t) = 1 + cos(t)*sin(t)^2
figure
ezpolar('1+cos(t)*sin(t)^2')
```



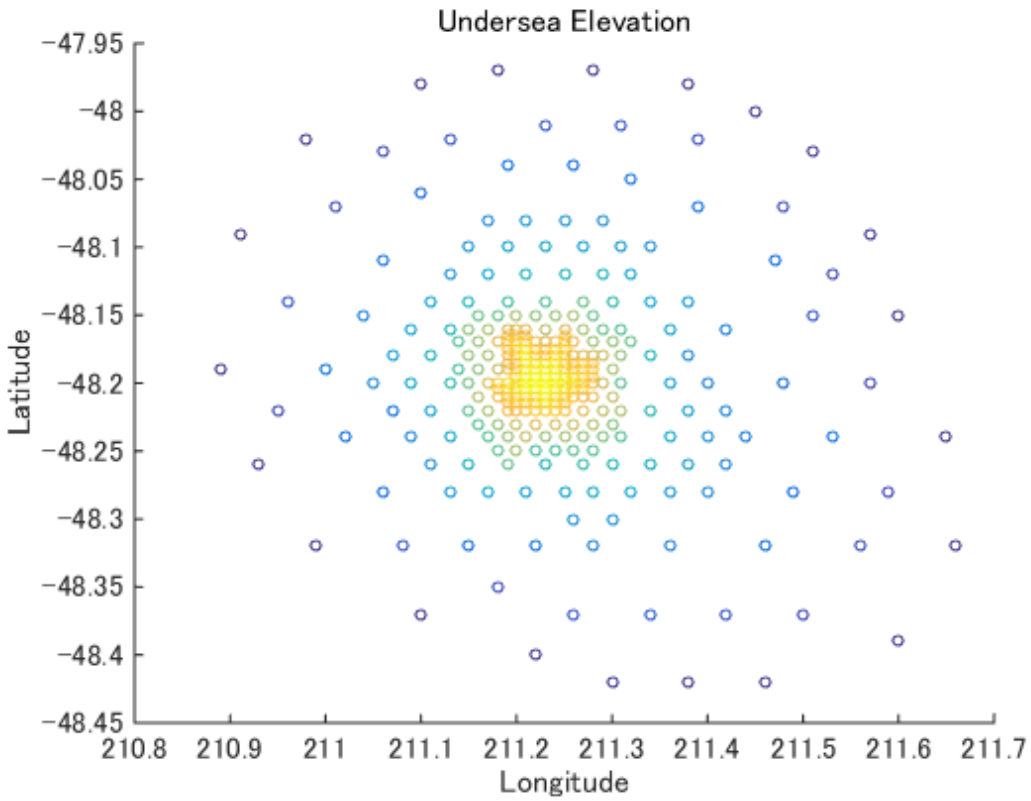
rose

```
% Load sunspot data
load sunspotData sunspot

% Create a rose plot with 24 sectors
figure
rose(sunspot, 24)

% Add title
title('Sunspot Frequency')
```



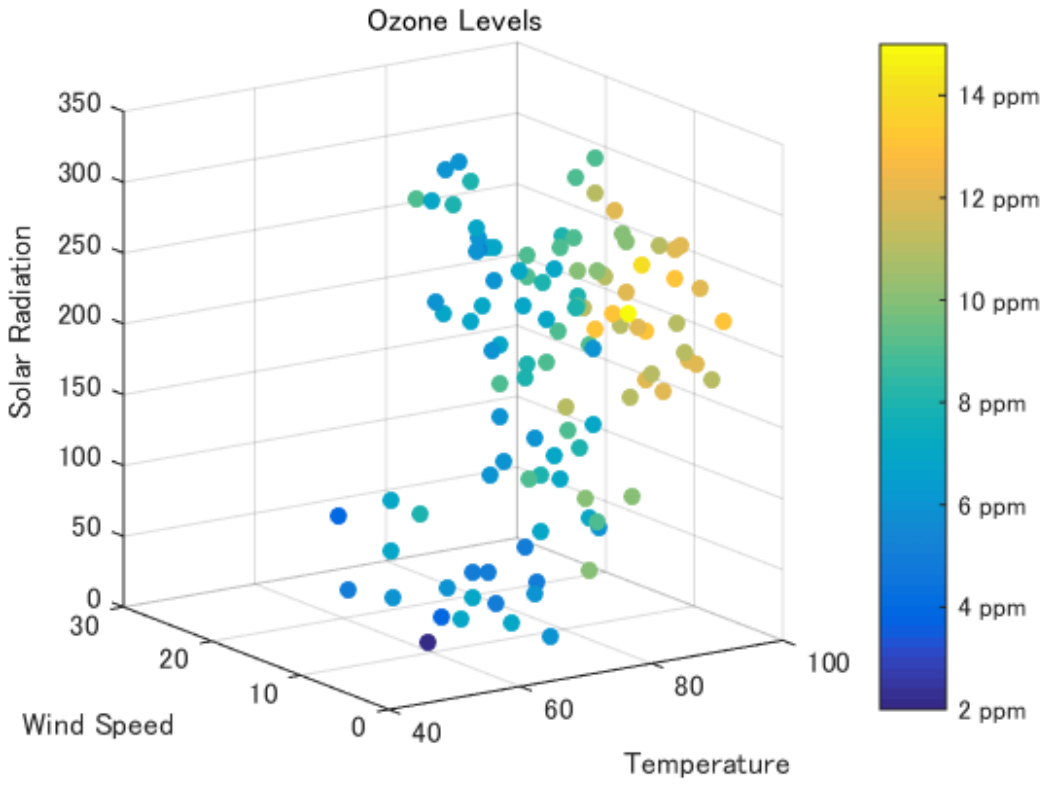
scatter

```
% Load undersea elevation data
load seamount x y z

% Create a scatter plot using the scatter function
figure
scatter(x, y, 10, z)

% Add title and axis labels
title('Undersea Elevation')
xlabel('Longitude')
ylabel('Latitude')
```



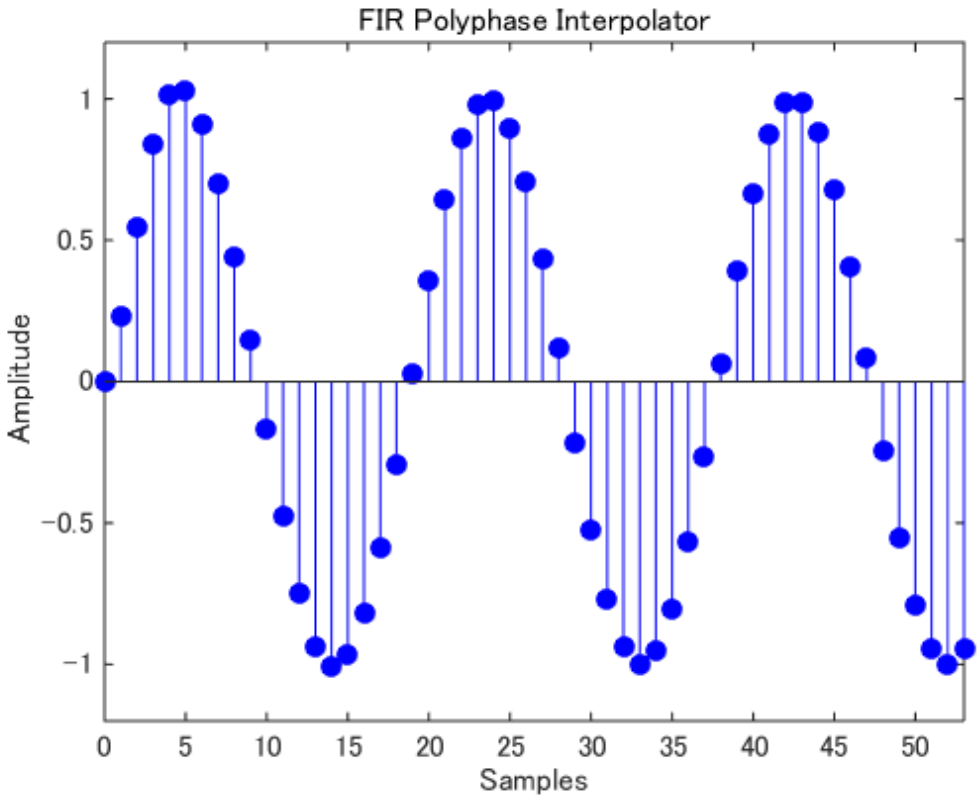
scatter3

```
% Load data on ozone levels
load ozoneData Ozone Temperature WindSpeed SolarRadiation

% Calculate the ozone levels
z = (Ozone).^(1/3);
response = z;

% Make a color index for the ozone levels
nc = 16;
offset = 1;
c = response - min(response);
c = round((nc-1-2*offset)*c/max(c)+1+offset);

% Create a 3D scatter plot using the scatter3 function
figure
scatter3(Temperature, WindSpeed, SolarRadiation, 30, c, 'filled')
view(-34, 14)

% Add title and axis labels
title('Ozone Levels'); xlabel('Temperature'); ylabel('Wind Speed');zlabel('Solar Radiation')

% Add a colorbar with tick labels
colorbar('Location', 'EastOutside', 'YTickLabel',...
      {'2 ppm', '4 ppm', '6 ppm', '8 ppm','10 ppm', '12 ppm', '14 ppm'})
```



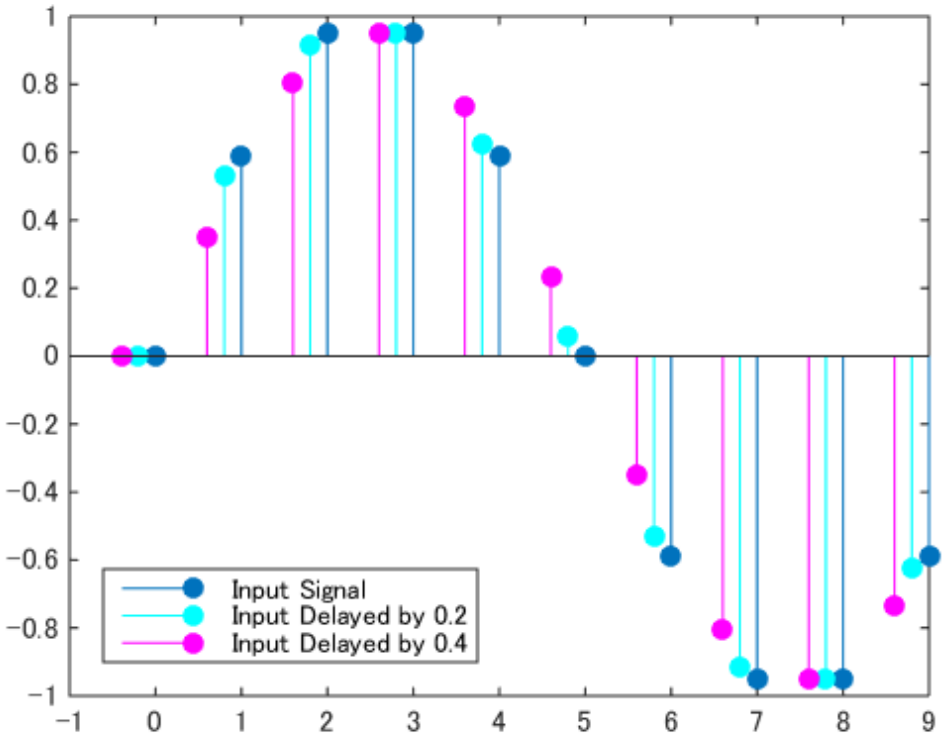
stem

```
% Load amplitude data
load amplitudeData sample amplitude

% Create a stem plot using the stem function
figure
stem(sample, amplitude, 'filled', 'b')

% Adjust the axis limits
axis([0 53 -1.2 1.2])

% Add title and axis labels
title('FIR Polyphase Interpolator')
xlabel('Samples')
ylabel('Amplitude')
```



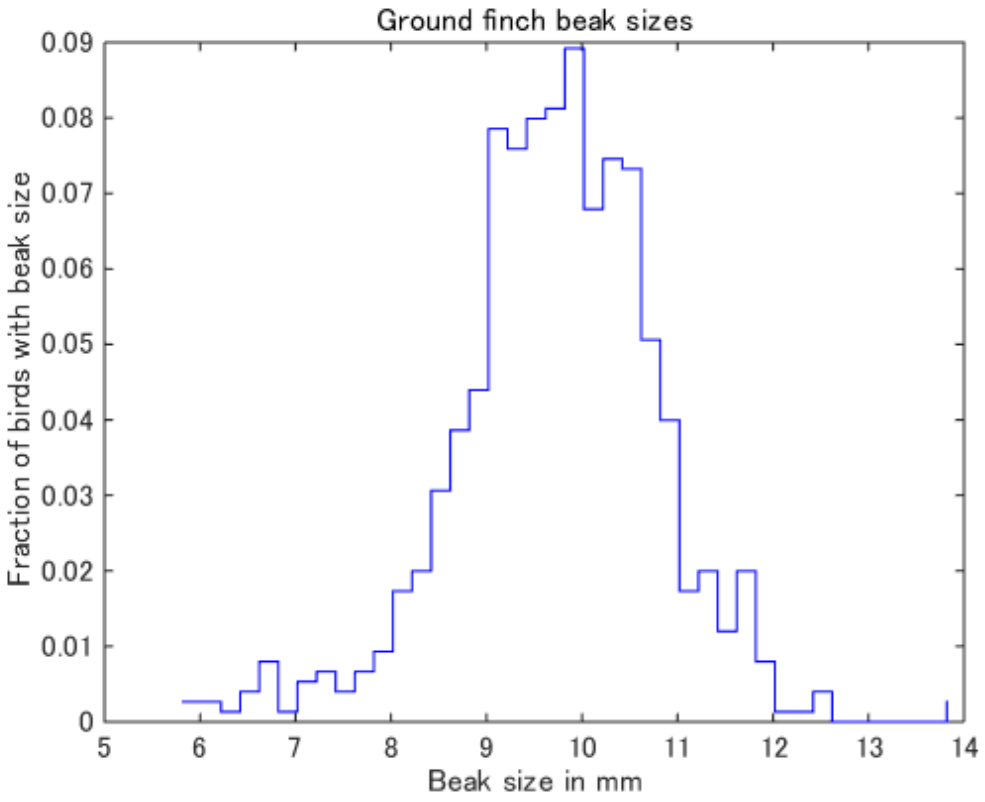
stem

```
% Load the filter data
load filterData time signal filter1 filter2

% Create a stem plot using the stem function
figure
stem(time, signal, 'filled')
hold on

% Add the second and third data sets to the plot
stem(time - 0.2, filter1, 'c', 'filled')
stem(time - 0.4, filter2, 'm', 'filled')

% Add a legend
legend('Input Signal', 'Input Delayed by 0.2', ...
       'Input Delayed by 0.4', 'Location', 'SouthWest')
```



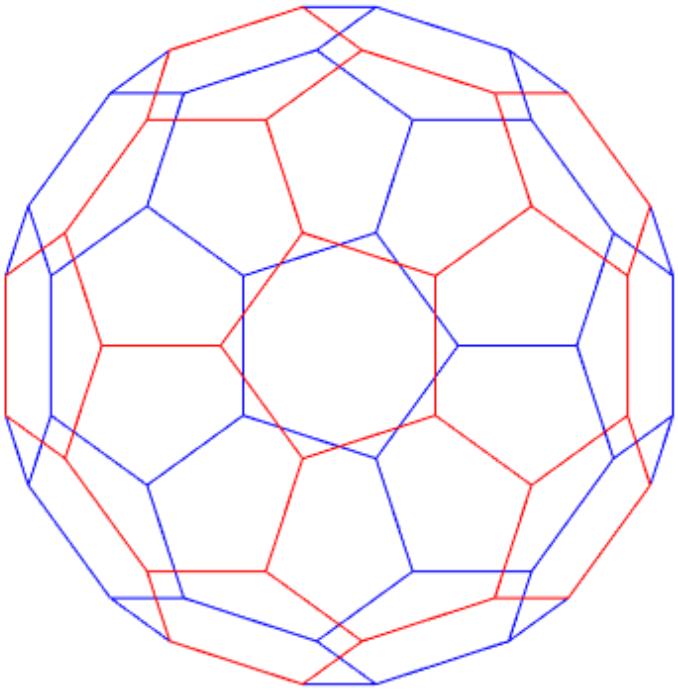
stair

```
% Load data on beak length
load beaksData beaks

% Calculate histograms for the beak data
minBeak = min(beaks);
maxBeak = max(beaks);
nbins = minBeak:0.2:maxBeak;
[nB, xB] = hist(beaks, nbins);

% Create a stair plot of beak sizes
figure
stairs(xB, nB/sum(nB), 'b')

% Add title and axis labels
title('Ground finch beak sizes')
xlabel('Beak size in mm')
ylabel('Fraction of birds with beak size')
```



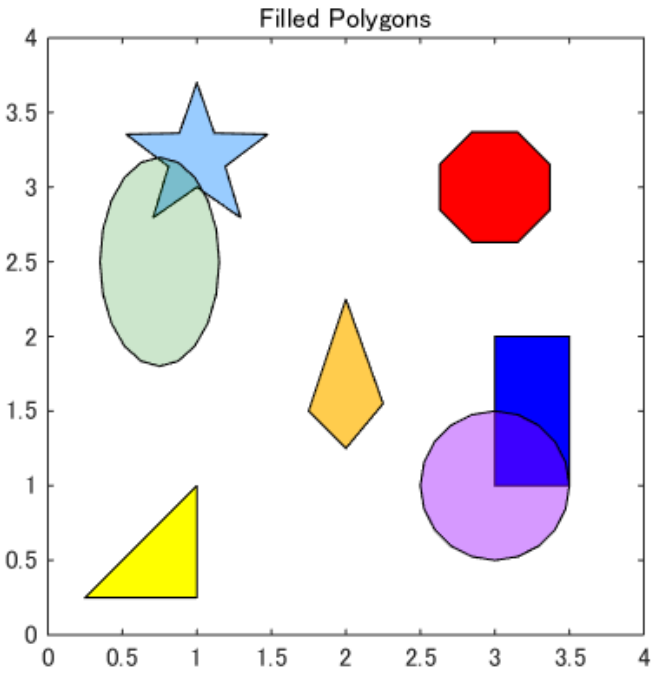
gplot

```
% Create the onnectivity graph of the Buckminster Fuller geodesic dome
[B, V] = bucky;
H = sparse(60, 60);
k = 31:60;
H(k, k) = B(k, k);

% Visualize the graph using the gplot function (blue)
figure
gplot(B - H, V, 'b-')
hold on

% Visualize a rotation of the graph (red)
gplot(H, V, 'r-')
axis off equal
```



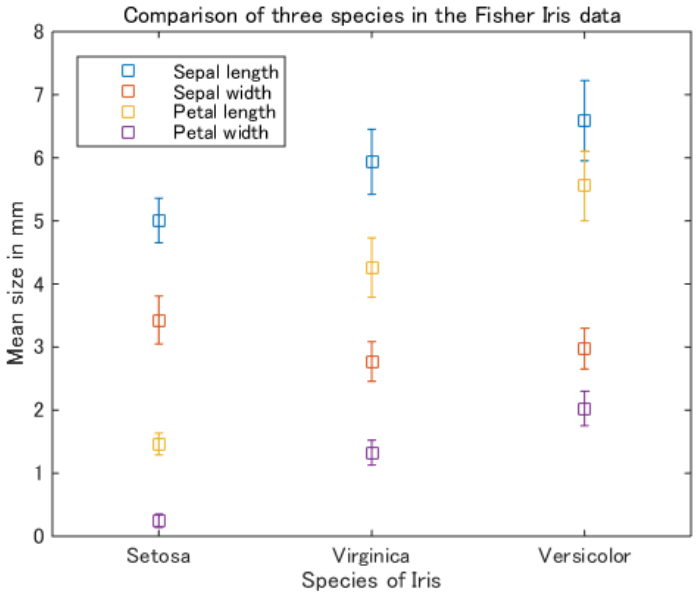
fill

```
% Create a yellow triangle
figure
x = [0.25, 1.0, 1.0];
y = [0.25, 0.25, 1.0];
fill(x, y, 'y')
hold on

% Create an orange diamond
x = [2.0, 2.25, 2.0, 1.75];
y = [1.25, 1.55, 2.25, 1.5];
c = [1 0.8 0.3];
fill(x, y, c)

% Create a blue rectangle
left = 3.0;
right = left + 0.5;
bottom = 1.0;
top = bottom + 1;
x = [left left right right];
y = [bottom top top bottom];
fill(x, y, 'b')

% Create a purple transparent circle
xc = 3.0;
yc = 1.0;
r = 0.5;
x = r*sin(-pi:0.1*pi:pi) + xc;
y = r*cos(-pi:0.1*pi:pi) + yc;
c = [0.6 0 1];
fill(x, y, c, 'FaceAlpha', 0.4)

% Create a light blue star
xc = 1.0;
yc = 3.0;
t = (-1/4:1/10:3/4)*2*pi;
r1 = 0.5;
r2 = 0.2;
r = (r1+r2)/2 + (r1-r2)/2*(-1).^[0:10];
x = r.*cos(t) + xc;
y = r2 - r.*sin(t) + yc;
c = [0.6 0.8 1.0];
fill(x, y, c)

%    Create a green transparent ellipse
xc = 0.75;
yc = 2.5;
x = 0.4*sin(-pi:0.1*pi:pi) + xc;
y = 0.7*cos(-pi:0.1*pi:pi) + yc;
c = [0 0.5 0];
fill(x, y, c, 'FaceAlpha', 0.2)

% Create a red stop sign
t = (1/16:1/8:1)'*2*pi;
x = 0.4*sin(t) + 3;
y = 0.4*cos(t) + 3;
fill(x, y ,'r', 'FaceAlpha', 1)

% Set the axis limits
axis([0 4 0 4])
axis square

% Add a title
title('Filled Polygons')
```



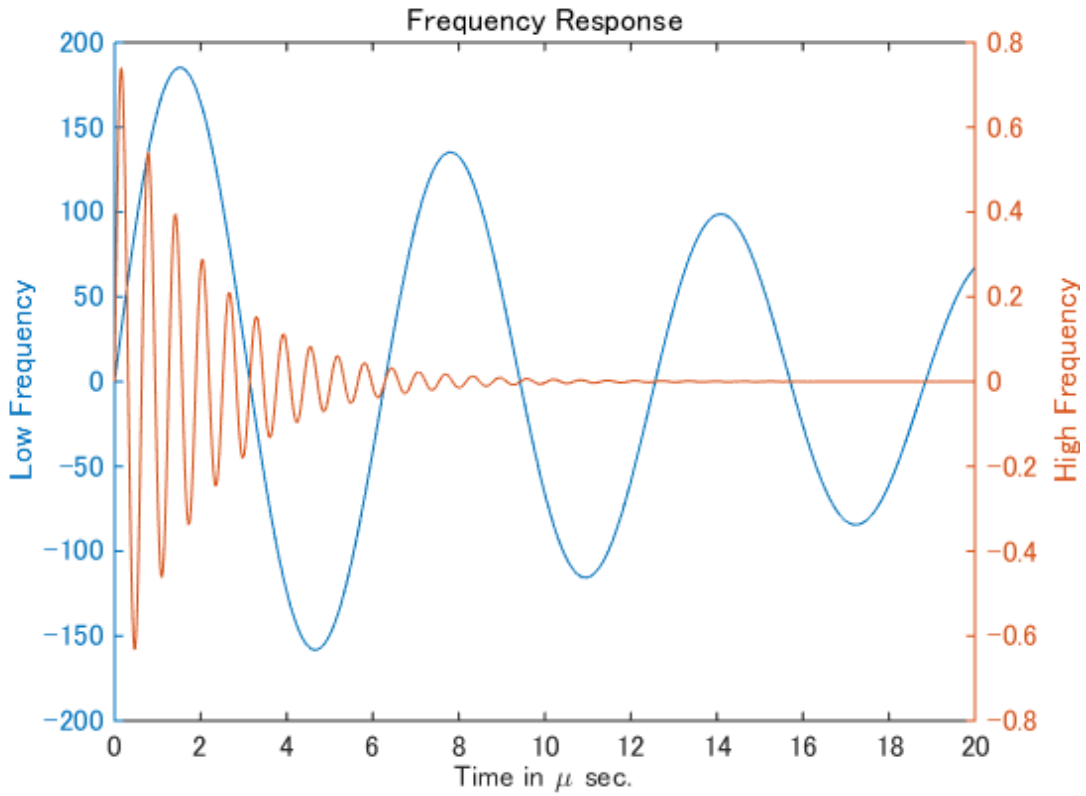
errorbar

```
% Load Fisher data for three varieties of iris
load fisheriris meas

% Extract the data for each variety
setosa = meas(1:50, :);
virginica = meas(51:100, :);
versicolor = meas(101:150, :);

% Calculate the means and standard deviations for each variety
irisMeans = [mean(setosa); mean(virginica); mean(versicolor)];
irisSTDs =     [std(setosa); std(virginica); std(versicolor)];

% Draw error bar chart with means and standard deviations
figure
errorbar(irisMeans, irisSTDs, 's')

% Add title and axis labels
title('Comparison of three species in the Fisher Iris data')
xlabel('Species of Iris')
ylabel('Mean size in mm')
box on

% Change the labels for the tick marks on the x-axis
irisSpecies = {'Setosa', 'Virginica', 'Versicolor'};
set(gca, 'XTick', 1:3, 'XTickLabel', irisSpecies)

% Create labels for the legend
irisMeas = {'Sepal length', 'Sepal width', 'Petal length', 'Petal width'};
legend(irisMeas, 'Location', 'Northwest')
```

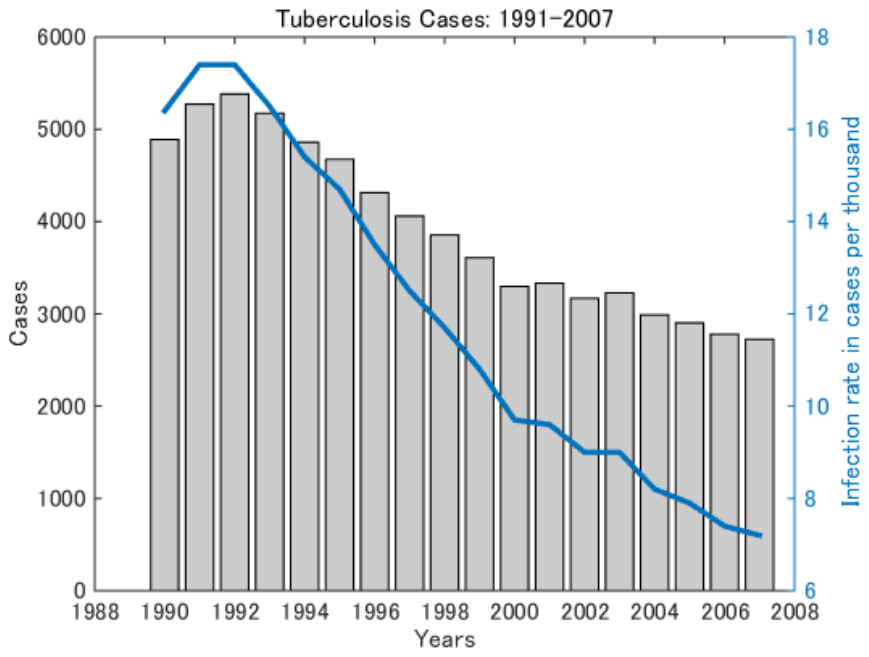


plotyy

```
% Create some data for the two curves to be plotted
x  = 0:0.01:20;
y1 = 200*exp(-0.05*x).*sin(x);
y2 = 0.8*exp(-0.5*x).*sin(10*x);

% Create a plot with 2 y axes using the plotyy function
figure
[ax, h1, h2] = plotyy(x, y1, x, y2, 'plot');

% Add title and x axis label
xlabel('Time in \mu sec.')
title('Frequency Response')
% Use the axis handles to set the labels of the y axes
set(get(ax(1), 'YLabel'), 'String', 'Low Frequency')
set(get(ax(2), 'YLabel'), 'String', 'High Frequency')
```



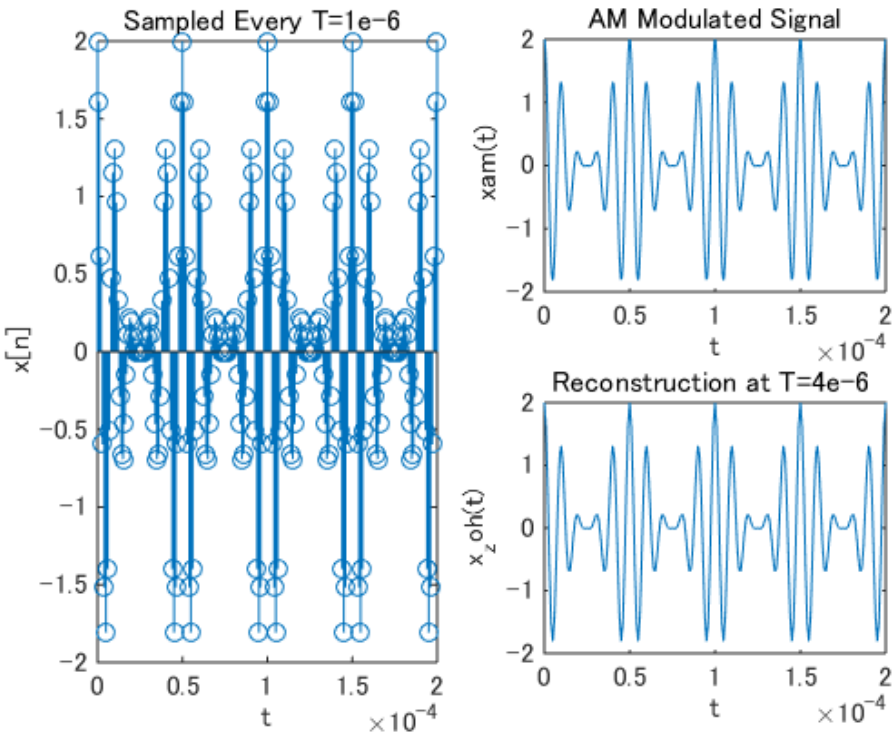
```
plotyy
% Create the data for the plots
TBdata = [1990 4889 16.4; 1991 5273 17.4; 1992 5382 17.4; 1993 5173 16.5;
          1994 4860 15.4; 1995 4675 14.7; 1996 4313 13.5; 1997 4059 12.5;
          1998 3855 11.7; 1999 3608 10.8; 2000 3297  9.7; 2001 3332  9.6;
          2002 3169  9.0; 2003 3227  9.0; 2004 2989  8.2; 2005 2903  7.9;
          2006 2779  7.4; 2007 2725  7.2];
years = TBdata(:,1);
cases = TBdata(:,2);
rate  = TBdata(:,3);
% Create a plot with 2 y axes using the plotyy function
% Cases are represented by a bar chart ; Infection rate is represented by an xy plot
figure
[ax, h1, h2] = plotyy(years, cases, years, rate, 'bar', 'plot');

% Change the bar colors to light gray
set(h1, 'FaceColor', [0.8, 0.8, 0.8])

% Chnage the thickness of the line
set(h2, 'LineWidth', 2)

% Add title and x axis label
title('Tuberculosis Cases: 1991-2007')
xlabel('Years')

% Use the axis handles to set the labels of the y axes
set(get(ax(1), 'YLabel'), 'String', 'Cases')
set(get(ax(2), 'YLabel'), 'String', 'Infection rate in cases per thousand')
```



subplot

```
% Calculate the data for the plots
fm = 20e3;
fc = 100e3;
tstep = 100e-9;
tmax    = 200e-6;
t = 0:tstep:tmax;
xam = (1 + cos(2*pi*fm*t)).*cos(2*pi*fc*t);

T = 1e-6;
N = 200;
nT = 0:T:N*T;
xn = (1 + cos(2*pi*fm*nT)).*cos(2*pi*fc*nT);

% Create the stem plot for the Sampled Signal
% spanning positions 1 & 3 of a 2x2 grid
figure
subplot(2, 2, [1 3])
stem(nT,xn)
xlabel('t')
ylabel('x[n]')
title('Sampled Every T=1e-6 ')
```

```
% Create the xy plot for the AM Modulated
% signal in position 2 of a 2x2 grid
subplot(2, 2, 2)
plot(t, xam)
axis([0 200e-6 -2 2])
xlabel('t')
ylabel('xam(t)')
title('AM Modulated Signal')

% Create the xy plot for the reconstructed
% signal in position 4 of a 2x2 grid
subplot(2, 2, 4)
plot(nT, xn)
xlabel('t')
ylabel('x_zoh(t)')
title('Reconstruction at T=4e-6 ')
```



subplot

```
% Create the data to be plotted
TBdata = [1990 4889 16.4; 1991 5273 17.4; 1992 5382 17.4; 1993 5173 16.5;
          1994 4860 15.4; 1995 4675 14.7; 1996 4313 13.5; 1997 4059 12.5;
          1998 3855 11.7; 1999 3608 10.8; 2000 3297    9.7; 2001 3332    9.6;
          2002 3169    9.0; 2003 3227    9.0; 2004 2989    8.2; 2005 2903    7.9;
          2006 2779    7.4; 2007 2725    7.2];

measles = [38556 24472 14556 18060 19549 8122 28541 7880 3283 4135 7953 1884]';
mumps = [20178 23536 34561 37395 36072 32237 18597 9408 6005 6268 8963 13882]';
chickenPox = [37140 32169 37533 39103 33244 23269 16737 5411 3435 6052 12825 23332]';
years = TBdata(:, 1);
cases = TBdata(:, 2);
rate    = TBdata(:, 3);
% Create the pie chart in position 1 of a 2x2 grid
figure
subplot(2, 2, 1)
pie([sum(measles) sum(mumps) sum(chickenPox)], {'Measles', 'Mumps', 'Chicken Pox'})
title('Childhood Diseases')

      % Create the bar chart in position 2 of a 2x2 grid
subplot(2, 2, 2)
bar(1:12, [measles/1000 mumps/1000 chickenPox/1000], 0.5, 'stack')
xlabel('Month')
ylabel('Cases (in thousands)')
title('Childhood Diseases')
axis([0 13 0 100])
set(gca, 'XTick', 1:12)

% Create the stem chart in position 3 of a 2x2 grid
subplot(2, 2, 3)
stem(years, cases)
xlabel('Years')
ylabel('Cases')
title('Tuberculosis Cases')
axis([1988 2009 0 6000])

% Create the line plot in position 4 of a 2x2 grid
subplot(2, 2, 4)
plot(years, rate)
xlabel('Years')
ylabel('Infection Rate')
```

```
title('Tuberculosis Cases')
axis([1988 2009 5 20])
```

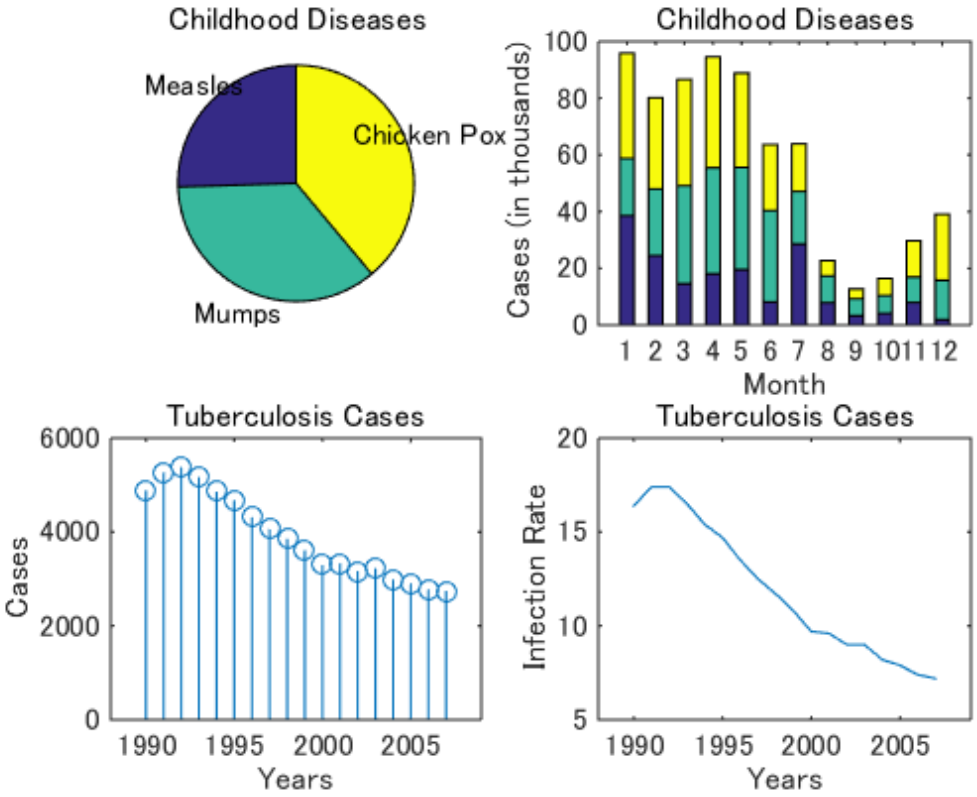


image (1)

```
% Load the data for the North Atlantic image
load NAimage lng lat naimg

% Create the image display using the image command
figure
image(lng, lat, naimg)

% Turn the axes off
axis off

% Add title
title('North Atlantic')
```

image (2)

```
% Load the data for the mandrill image
load mandrill X map

% Create the image display using the image command
figure
image(X)

% Use the colormap specified in the image data file
colormap(map)

% Turn the axes off
axis off

% Add title
title('Mandrill')
```

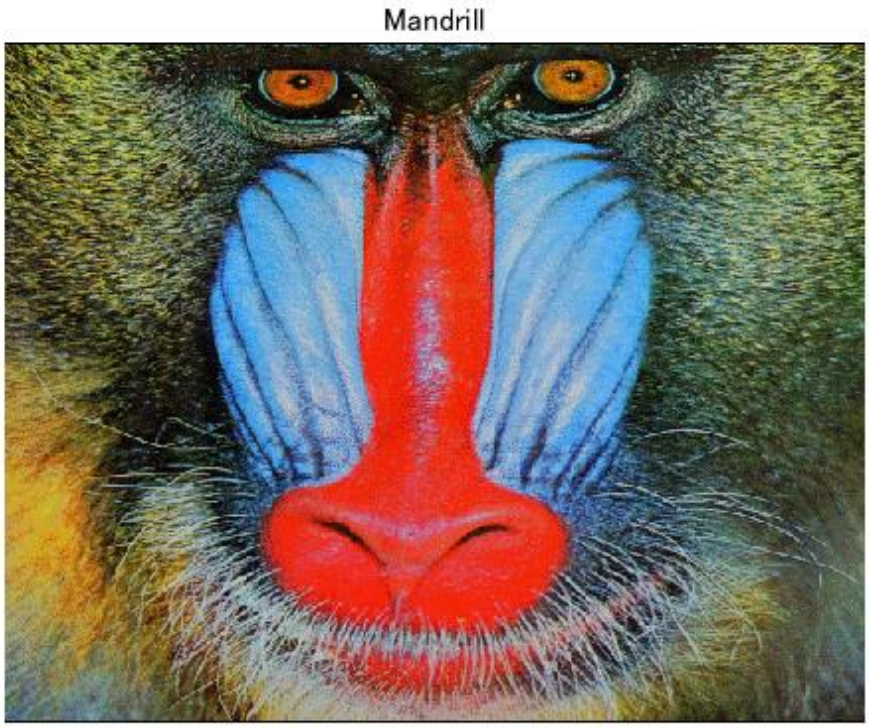
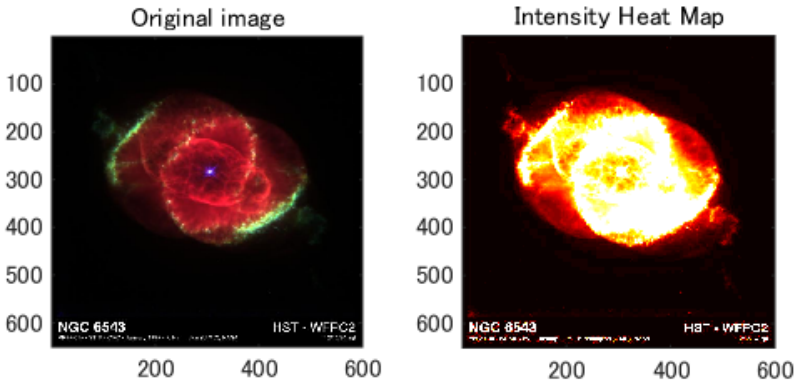


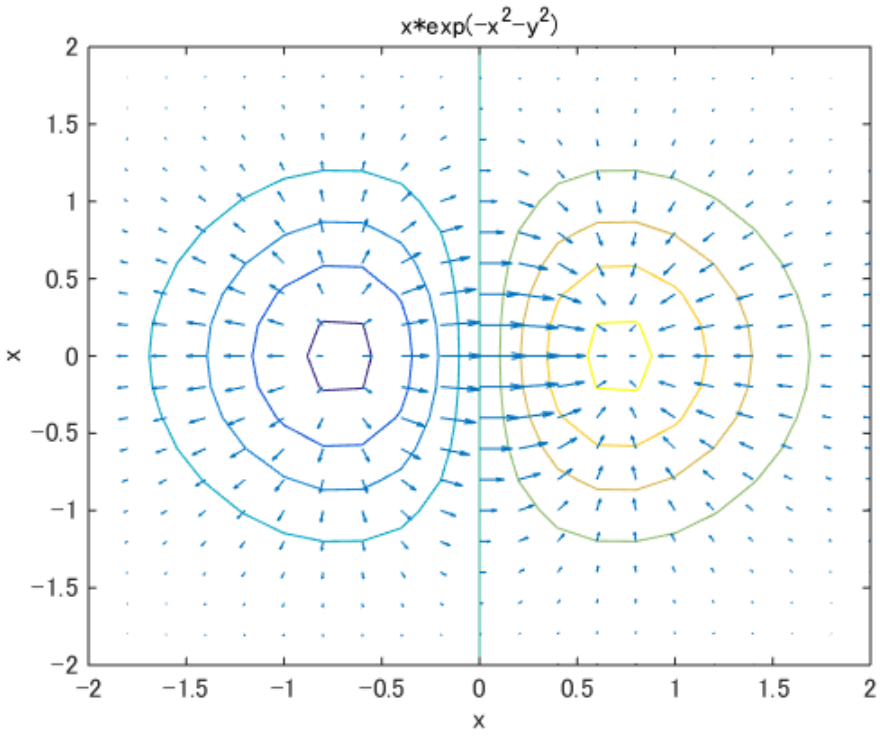
image (3)

```
% Read the data for the original image
original = imread('ngc6543a.jpg');

% Create the first image display using the image command
figure
subplot(1, 2, 1)
image(original)
axis square

% Add title for first image
title('Original image')

% Create the data for the second image
heatmap = mean(original, 3);

% Create the second image display using the image command
subplot(1, 2, 2)
image(heatmap)
colormap(hot)
axis square

% Add title for the second image
title('Intensity Heat Map')
```



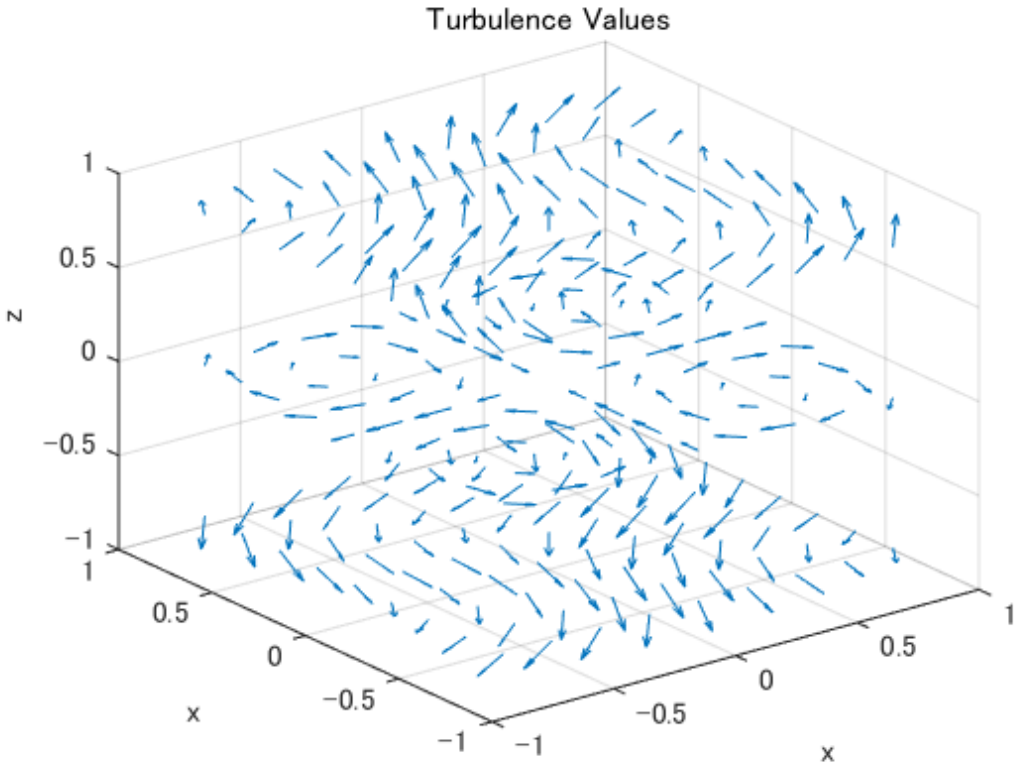
quiver

```
% Create a grid of x and y points
[x, y] = meshgrid(-2:.2:2);

% Create the function z(x,y) and its gradient
z = x.*exp(-x.^2 - y.^2);
[dx, dy] = gradient(z, .2, .2);

% Create a contour plot of x, y, and z using the contour function
figure
contour(x,y,z)
hold on

% Create a quiver plot of x, y, and the gradients using the quiver function
q = quiver(x, y, dx, dy);

% Set the axis limits
xlim([-2 2])
ylim([-2 2])

% Add title and axis labels
title('x*exp(-x^2-y^2)')
xlabel('x')
ylabel('x')
```



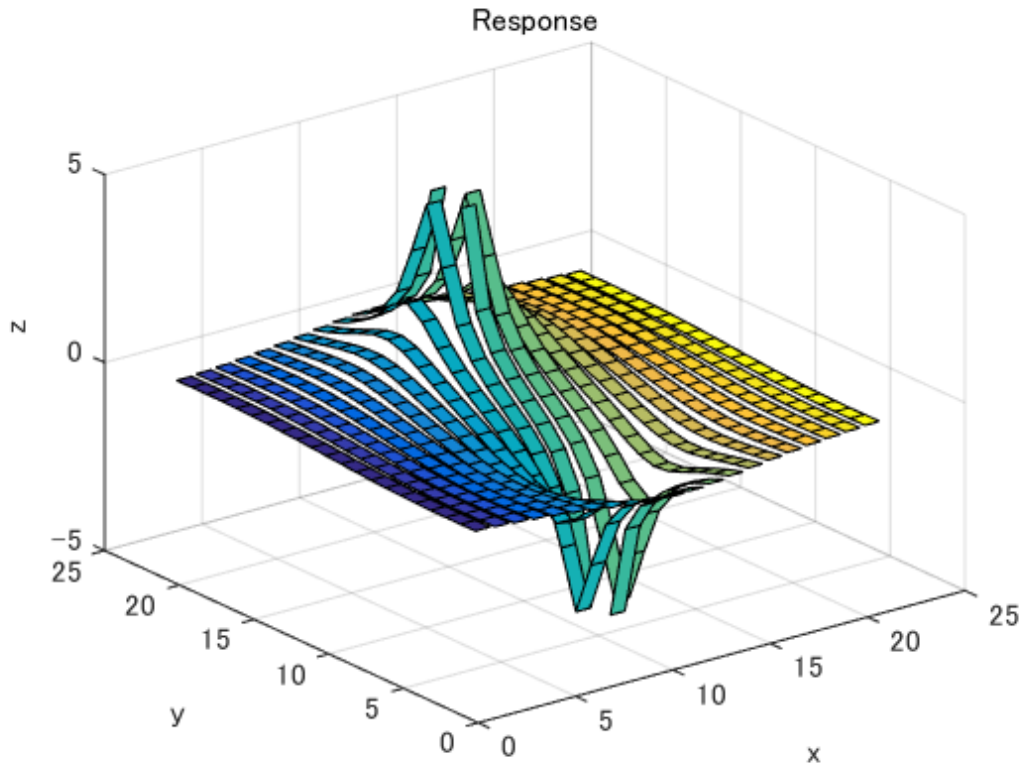
quiver3

```
% Create a grid of x,y, and z values
[x, y, z] = meshgrid(-0.8:0.2:0.8, -0.8:0.2:0.8, -0.8:0.8:0.8);

% Calculate homogenous turbulence values at each (x,y,z)
u = sin(pi*x).*cos(pi*y).*cos(pi*z);
v = -cos(pi*x).*sin(pi*y).*cos(pi*z);
w = sqrt(2/3)*cos(pi*x).*cos(pi*y).*sin(pi*z);

% Draw a 3 dimensional quiver plot using the quiver3 function
figure
quiver3(x, y, z, u, v, z)

% Set the axis limits
axis([-1 1 -1 1 -1 1])

% Add title and axis labels
title('Turbulence Values')
xlabel('x')
ylabel('x')
zlabel('z')
```



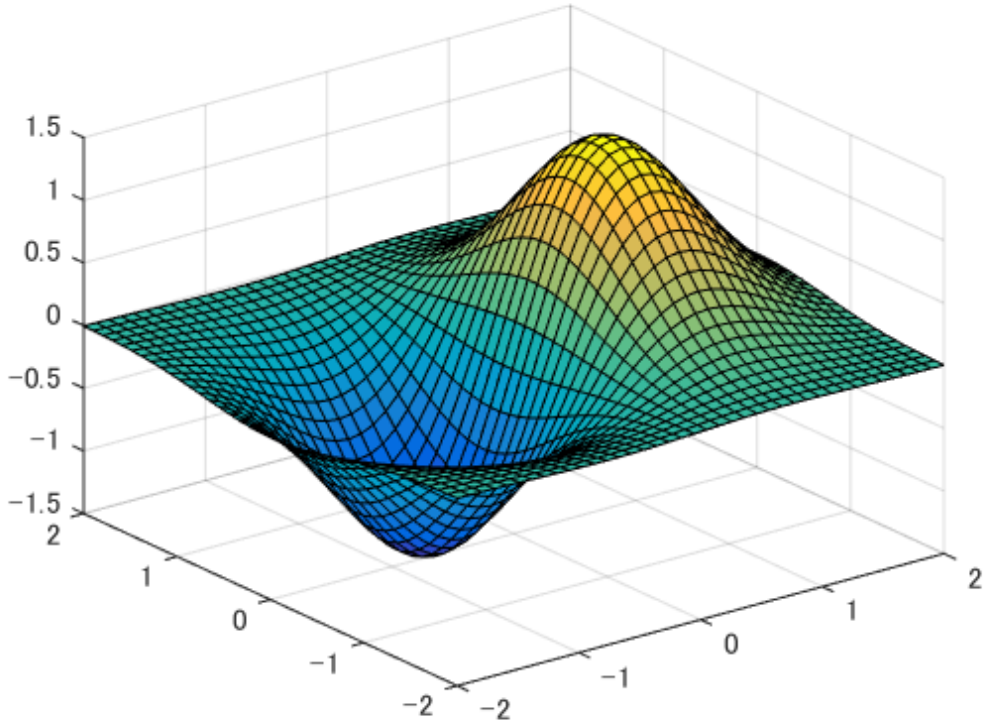
ribbon

```
% Create a grid of x and y points
[x, y] = meshgrid(-2:0.2:2,-2:0.2:2);

% Calculate the response values at each point
R = (1./(x.^2+(y-1).^2).^(1/2)) - (1./(x.^2+(y+1).^2).^(1/2));

% Create a ribbon point using the ribbon function
figure
ribbon(R)

% Add title and axis labels
title('Response')
xlabel('x')
ylabel('y')
zlabel('z')
```



surf (1)

```
% Create a grid of x and y points
points = linspace(-2, 2, 40);
[X, Y] = meshgrid(points, points);

% Define the function Z = f(X,Y)
Z = 2./exp((X-.5).^2+Y.^2)-2./exp((X+.5).^2+Y.^2);

% Create the surface plot using the surf command
figure
surf(X, Y, Z)
```

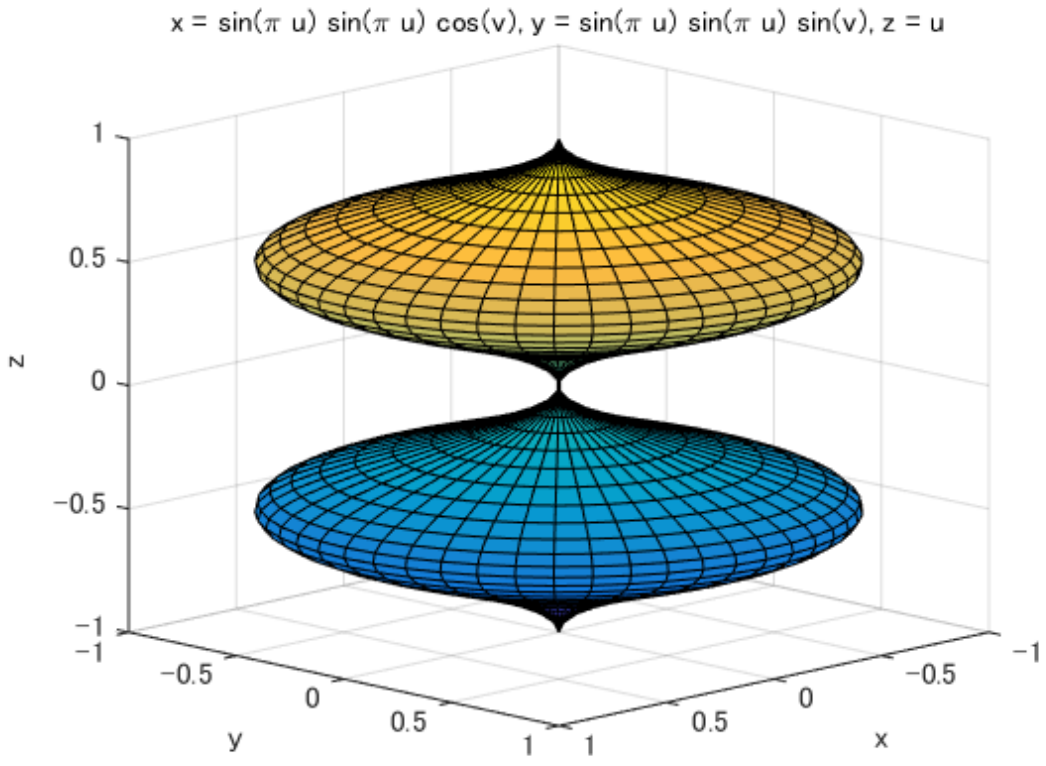


surf (2)

```
% Generate points for a cylinder with profile 2 + sin(t)
t = 0:pi/50:2*pi;
[x, y, z] = cylinder(2+sin(t));

% Create a surface plot using the surf function
figure
surf(x, y, z, 'LineStyle', 'none', 'FaceColor', 'interp')
colormap('summer')

% Turn off the axis and the grid
axis square
axis off
grid off
```

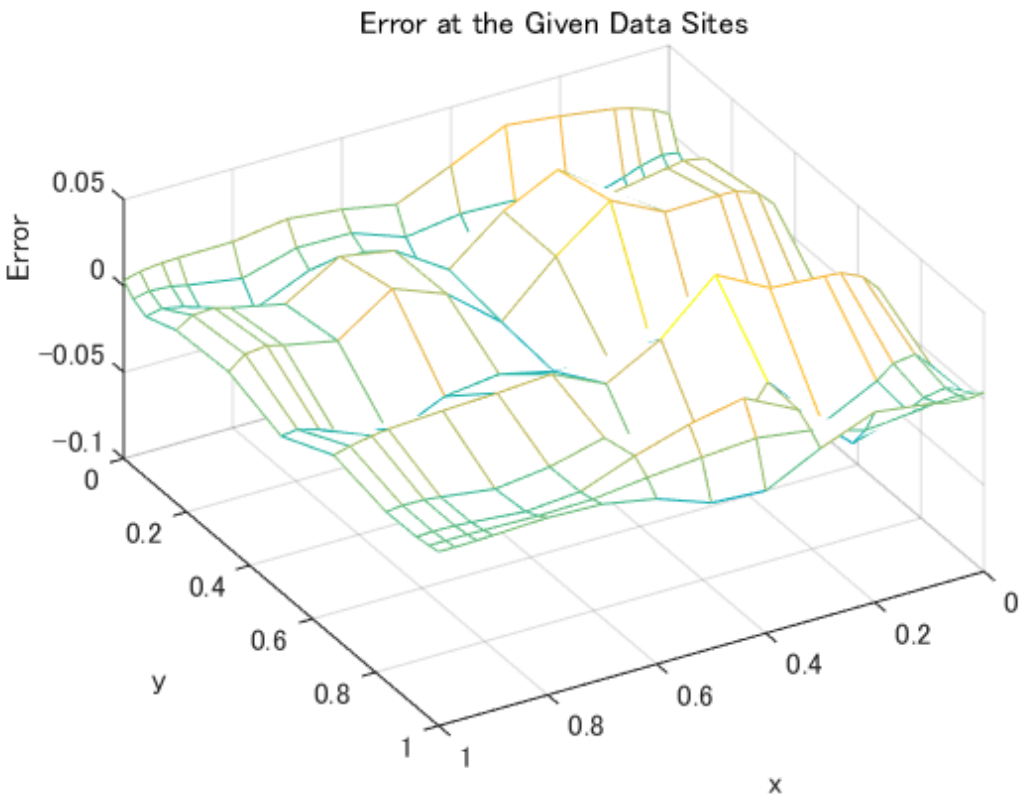
ezsurf

```
% Create the plot using the functions
% x = sin(pi*u)*sin(pi*u)*cos(v) ; y = sin(pi*u)*sin(pi*u)*sin(v) ; z = u
% with - 1 < u < 1 and 0 < v < 2*pi
figure
ezsurf('sin(pi*u)*sin(pi*u)*cos(v)', ...
                'sin(pi*u)*sin(pi*u)*sin(v)', ...
                'u', [-1 1 0 2*pi])

% Change the view angle for the plot
view(135,15)
```



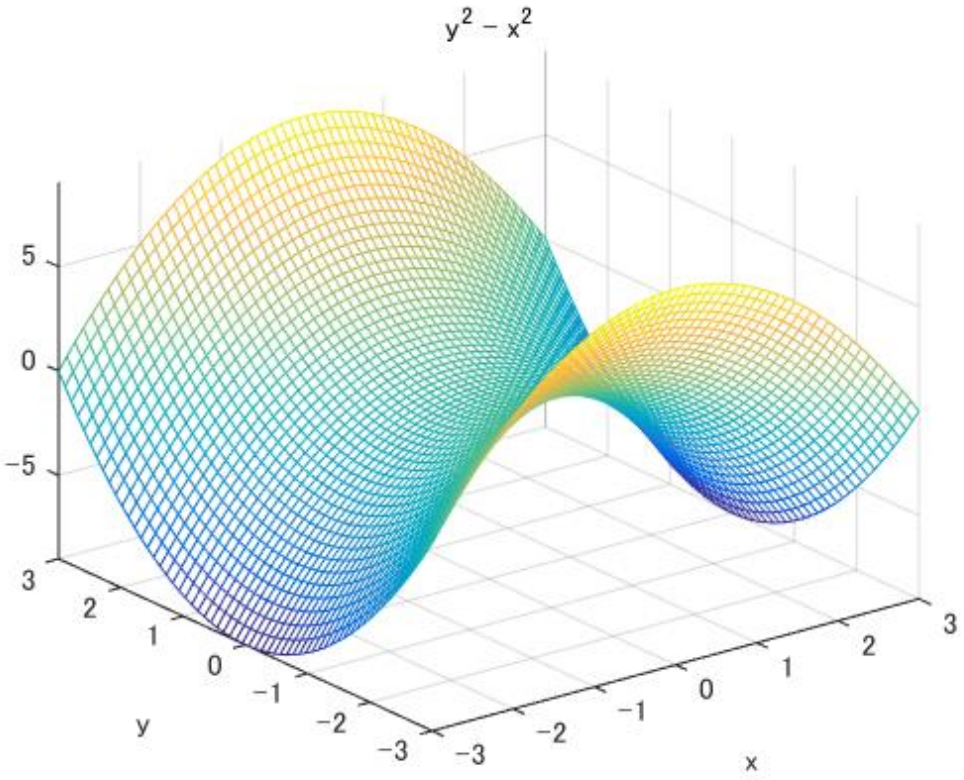
mesh

```
% Load the error data
load errorData x y errors

% Create a mesh plot using the mesh function
figure
mesh(x, y, errors)

% Set the view angle
view(150, 50)

% Add title and axis labels
title('Error at the Given Data Sites')
xlabel('x')
ylabel('y')
zlabel('Error')
```



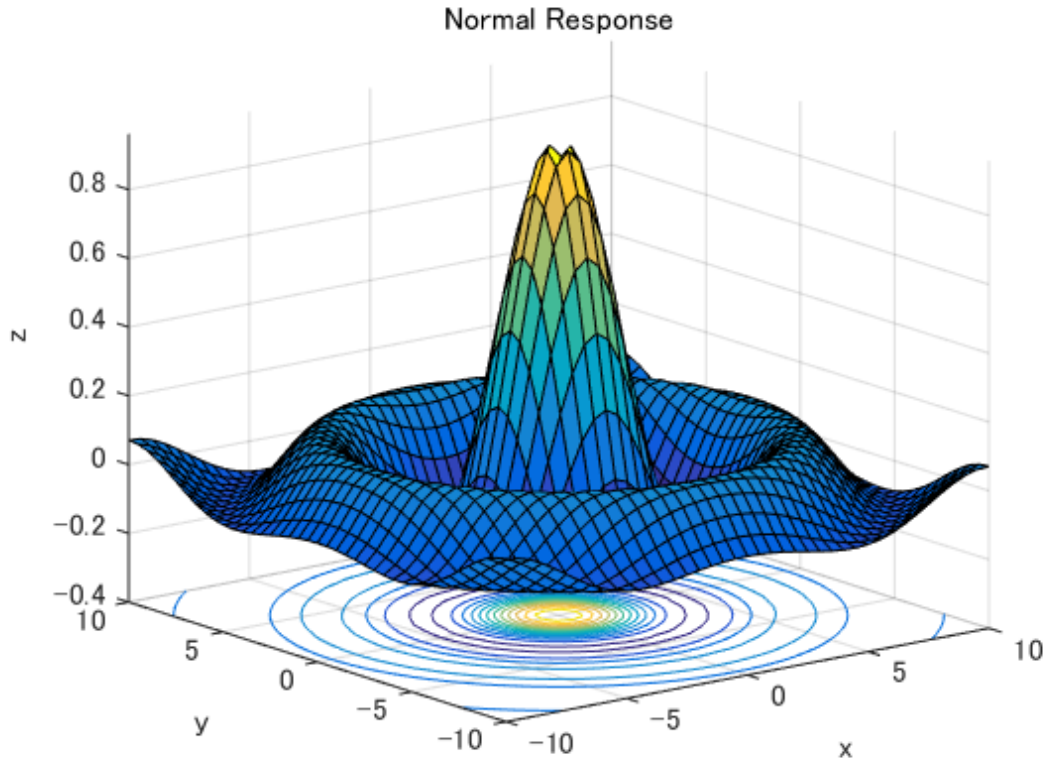
ezmesh

```
% Create the mesh plot using the function f(x,y) = y^2 - x^2
figure
ezmesh('y^2 - x^2', [-3 3], [-3 3])
```



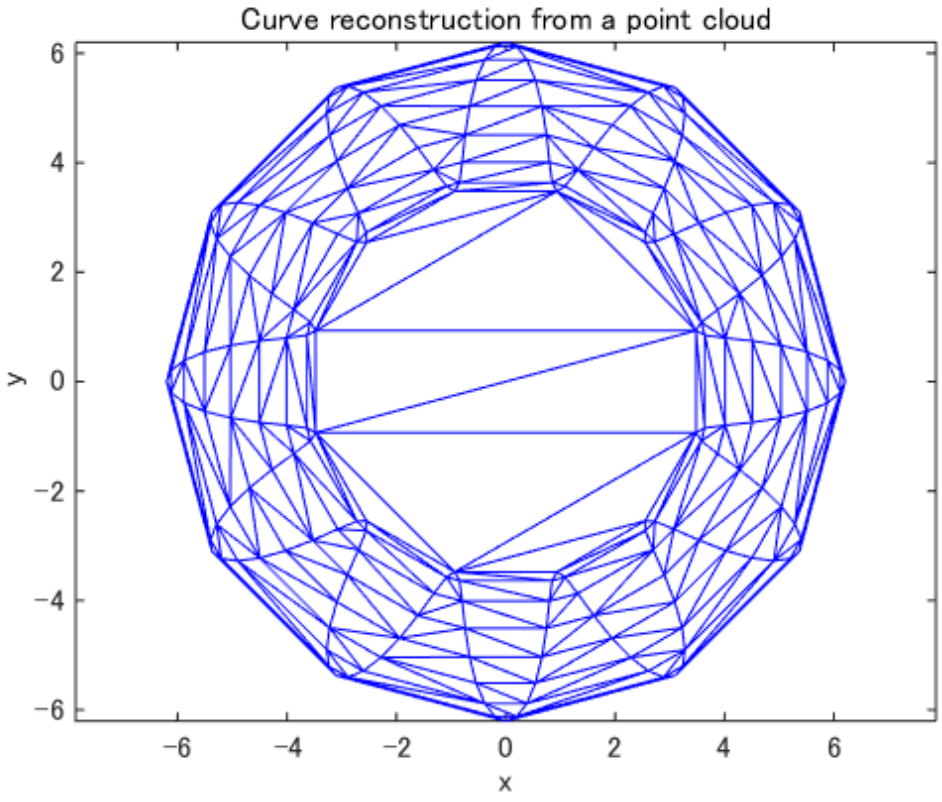
surfc

```
% Create a grid of x and y data
y = -10:0.5:10;
x = -10:0.5:10;
[X, Y] = meshgrid(x, y);

% Create the function values for Z = f(X,Y)
Z = sin(sqrt(X.^2+Y.^2)) ./ sqrt(X.^2+Y.^2);

% Create a surface contour plor using the surfc function
figure
surfc(X, Y, Z)

% Adjust the view angle
view(-38, 18)

% Add title and axis labels
title('Normal Response')
xlabel('x')
ylabel('y')
zlabel('z')
```



triplot

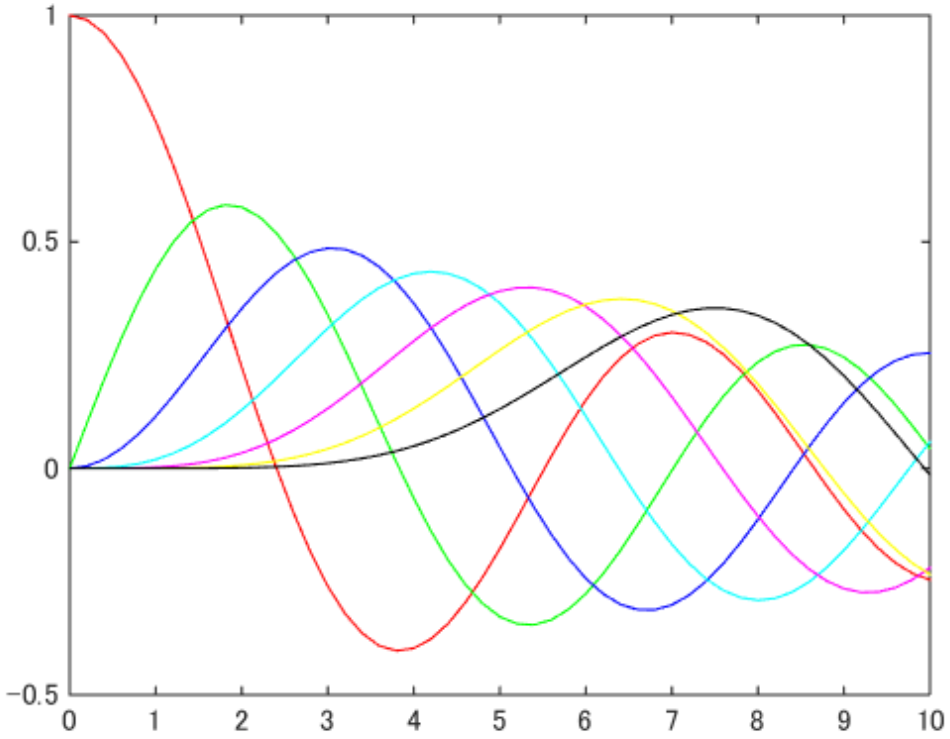
```
% Create a set of points representing a point cloud
numpts = 192;
t = linspace(-pi, pi, numpts+1)';
r = 0.1 + 5*sqrt(cos(6*t).^2 + (0.7).^2);
x = r.*cos(t);
y = r.*sin(t);

% Construct a Delaunay Triangulation of the point set
dt    = DelaunayTri(x,y);
tri = dt(:,:);

% Create a triangle plot of the Delauney Triangulation
figure
triplot(tri,x,y)
axis equal

% Add title and axis labels
title('Curve reconstruction from a point cloud')
xlabel('x')
ylabel('y')
```



定制绘图风格

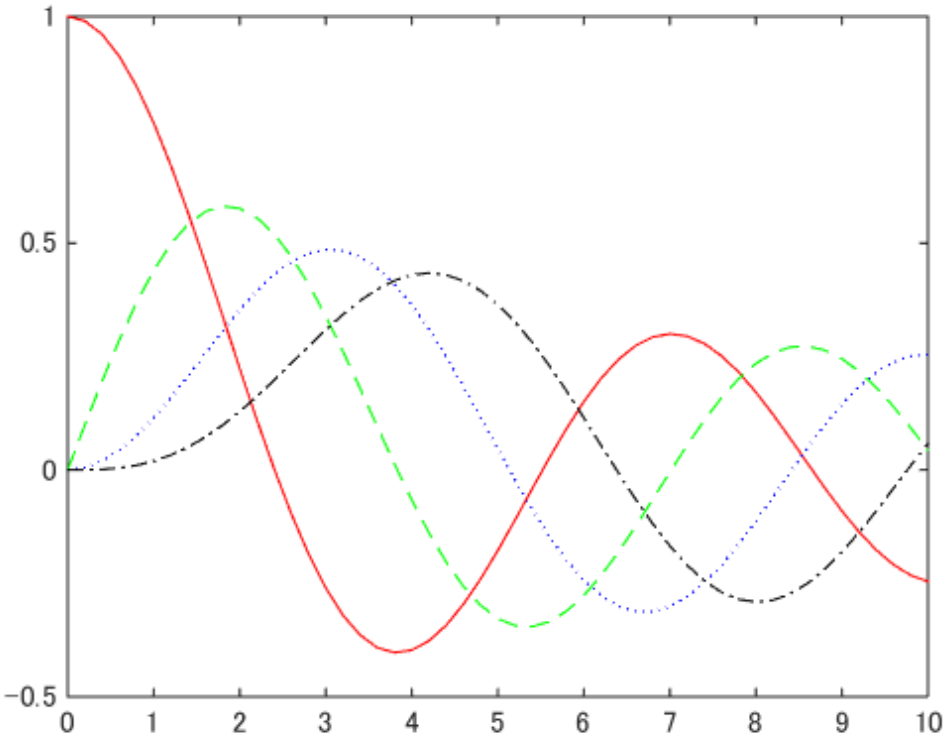
Standard Line Colors

```
% Generate some data using the besselj function
x = 0:0.2:10;
y0 = besselj(0,x);
y1 = besselj(1,x);
y2 = besselj(2,x);
y3 = besselj(3,x);
y4 = besselj(4,x);
y5 = besselj(5,x);
y6 = besselj(6,x);

% Plot the lines from the Bessel functions using standard line colors
figure
plot(x, y0, 'r', x, y1, 'g', x, y2, 'b', x, y3, 'c', ...
     x, y4, 'm', x, y5, 'y', x, y6, 'k')
```



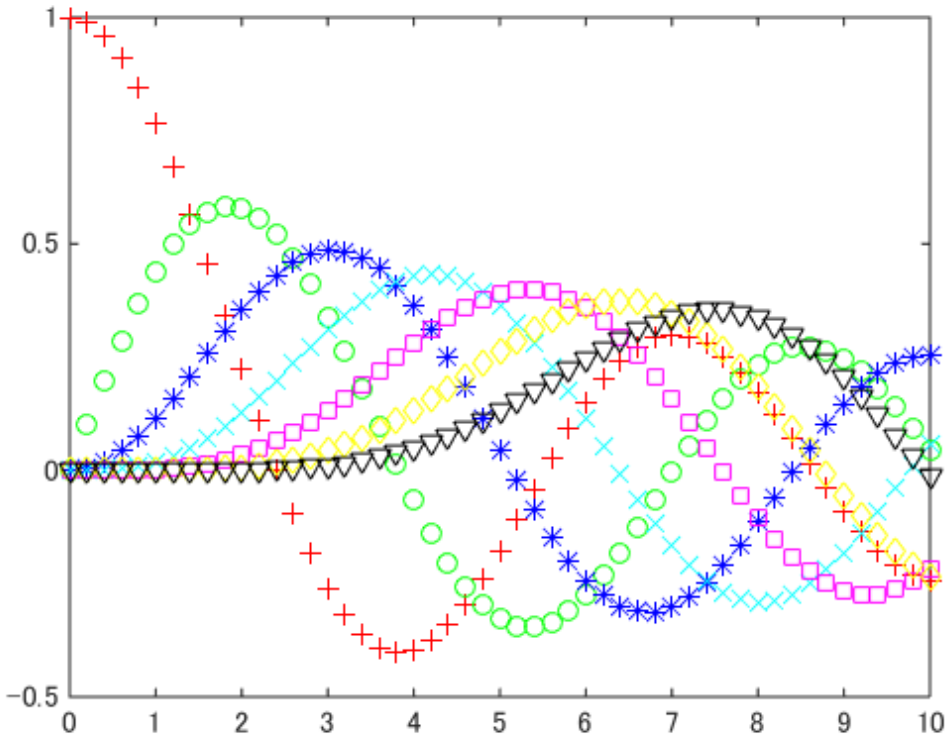
Standard Line Styles

```
% Generate some data using the besselj function
x = 0:0.2:10;
y0 = besselj(0,x);
y1 = besselj(1,x);
y2 = besselj(2,x);
y3 = besselj(3,x);

% Plot the lines from the Bessel functions using standard line styles
figure
plot(x, y0, 'r-', x, y1, 'g--', x, y2, 'b:', x, y3, 'k-.')
```



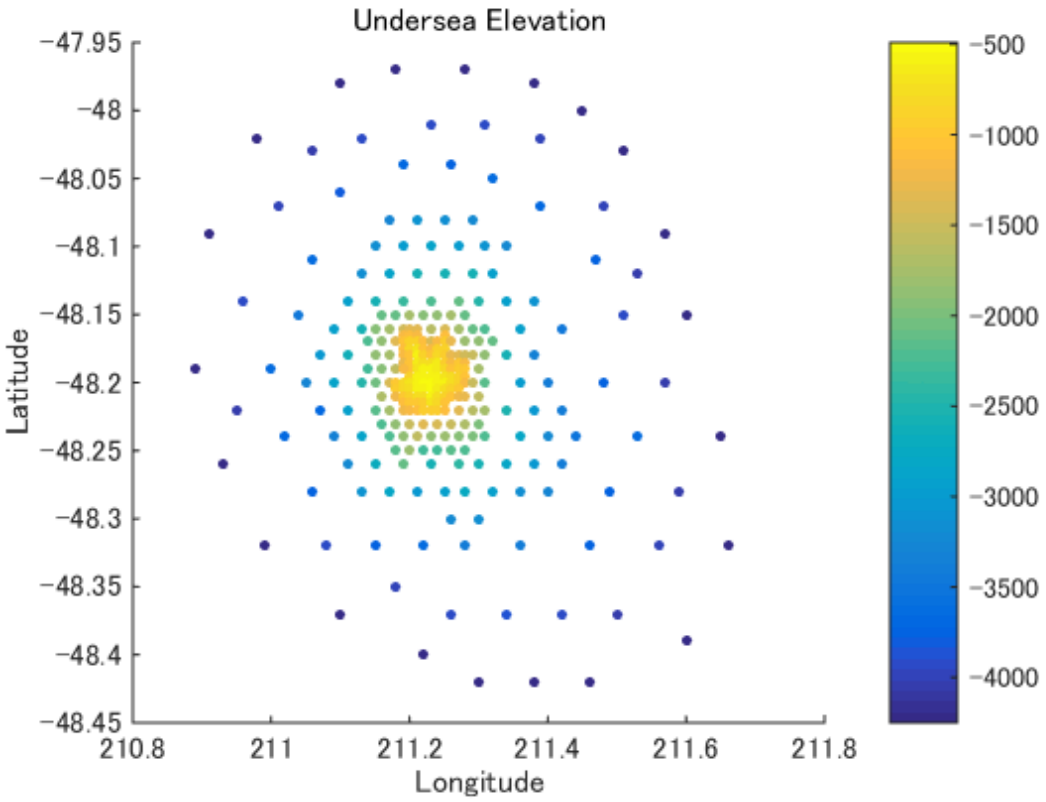
Standard Plot Markers

```
% Generate some data using the besselj function
x = 0:0.2:10;
y0 = besselj(0,x);
y1 = besselj(1,x);
y2 = besselj(2,x);
y3 = besselj(3,x);
y4 = besselj(4,x);
y5 = besselj(5,x);
y6 = besselj(6,x);

% Plot the points from the Bessel functions using standard marker types
figure
plot(x, y0, 'r+', x, y1, 'go', x, y2, 'b*', x, y3, 'cx', ...
     x, y4, 'ms', x, y5, 'yd', x, y6, 'kv')
```



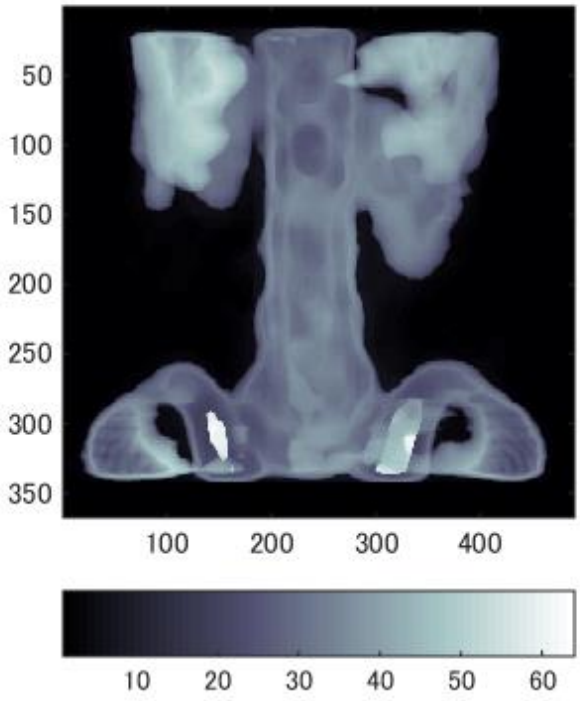
colorbar (1)

```
% Load sea elevation data
load seamount x y z

% Create a scatter plot of the data
figure
scatter(x, y, 10, z, 'filled')

% Add title and axis labels
title('Undersea Elevation')
xlabel('Longitude')
ylabel('Latitude')

% Add a vertical color bar - default position is to the right of the plot
colorbar
```



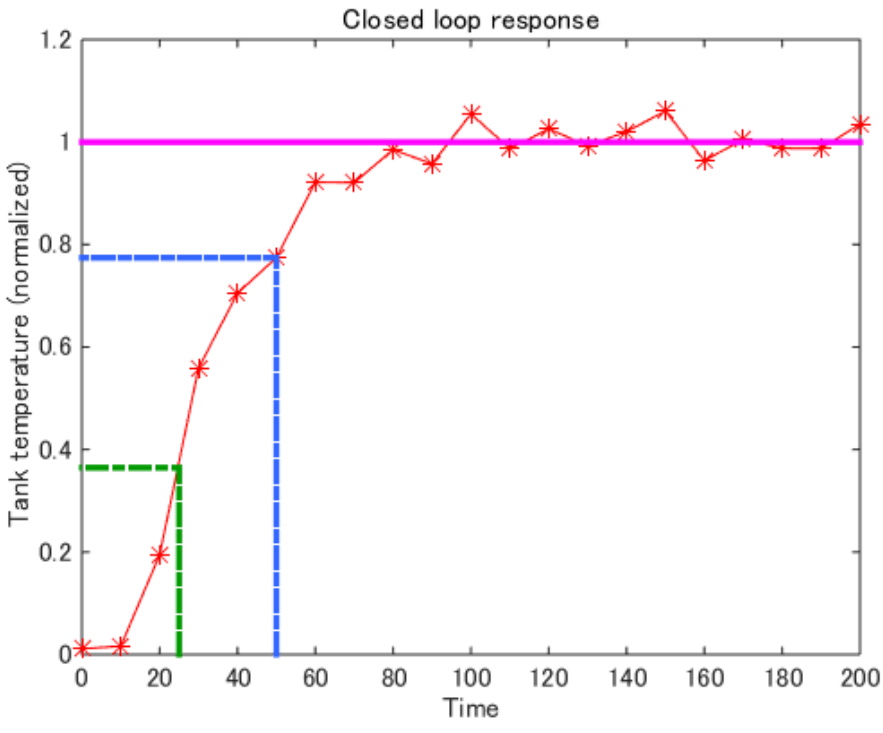
colorbar (2)

```
% Load spine data
load spine X

% Create an image plot of the spine data
figure
imagesc(X)
colormap bone

% Add a horizontal colorbar to the bottom of the plot
colorbar('SouthOutside')
axis square
```



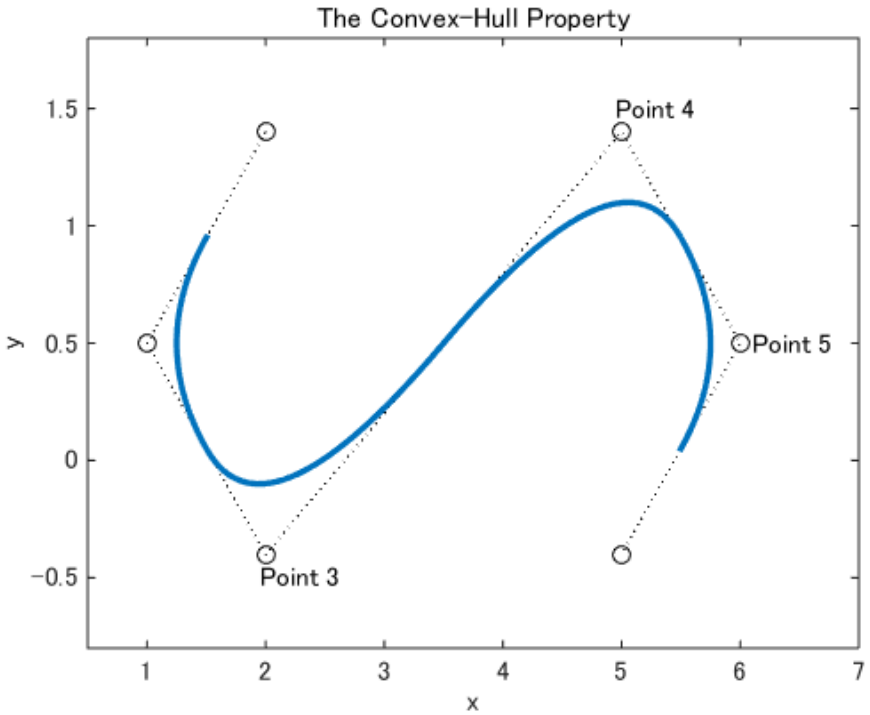
Adding Lines to Plots

```
% Load the step response data
load stepResponse step_data

% Plot the step response
figure
plot(step_data(:,1), step_data(:,2), 'r*-')

% Set the axis limits
axis([0 200 0 1.2])

% Add a title and axis labels
title('Closed loop response')
xlabel('Time')
ylabel('Tank temperature (normalized)')

% Add a horizontal line for the Temperature at steady state
line('XData', [0 200], 'YData', [1 1], 'LineStyle', '-', 'LineWidth', 2, 'Color','m')

% Add lines for the temperature at time = 25
x = 25;
y = (step_data(4,2) - step_data(3,2));
line('XData', [x x 0], 'YData', [0 y y], 'LineWidth', 2, 'LineStyle', '-.', 'Color', [0 0.6 0])

% Add lines for the temperature at time = 60
x = 50;
y = step_data(6,2);
line('XData', [x x 0], 'YData', [0 y y], 'LineWidth', 2,'LineStyle', '-.', 'Color', [0.2 0.4 1.0])
```



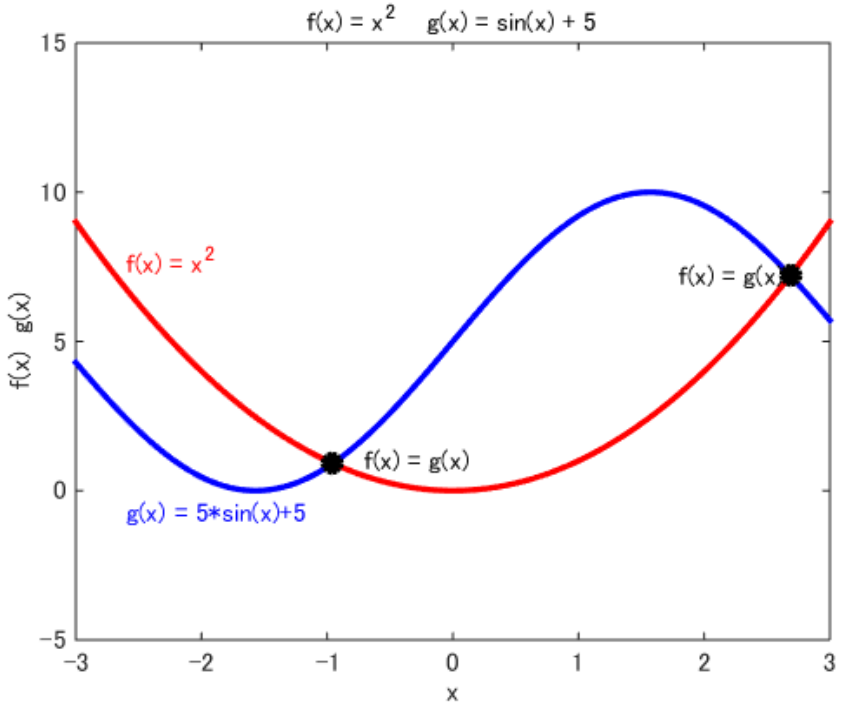
Adding Text to Plots (1)

```
% Load the points for creating a spline curve
load splineData points x y

% Plot the points for the spline curve
figure
plot(points(:,1),points(:,2),':ok')
hold on

% Plot the spline curve
plot(x, y, 'LineWidth', 2)
axis([.5 7 -.8 1.8])

% Add a title and axis labels
title('The Convex-Hull Property')
xlabel('x')
ylabel('y')

% Label points 3, 4, & 5 of the spline curve
xt = points(3,1) - 0.05;
yt = points(3,2) - 0.1;
text(xt, yt, 'Point 3')

xt = points(4,1) - 0.05;
yt = points(4,2) + 0.1;
text(xt, yt, 'Point 4')

xt = points(5,1) + 0.15;
yt = points(5,2) - 0.05;
text(6.1,.5, 'Point 5')
```



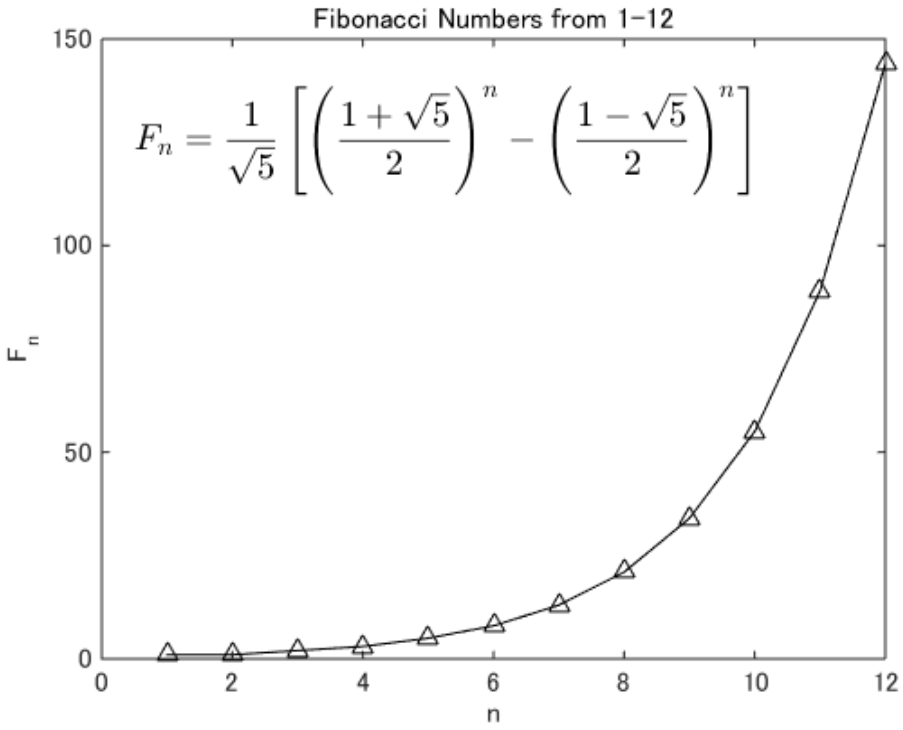
Adding Text to Plots (2)

```
% Define functions f = x^2 and g = 5*sin(x)+5
x = -3.0:0.01:3.0;
f = x.^2;
g = 5*sin(x) + 5;

% Plot function f
figure
plot(x, f, 'r-', 'LineWidth', 2)
hold on

% Plot function g
plot(x, g, 'b-', 'LineWidth', 2)
axis([-3,3,-5,15])

% Add title and axis labels
title('f(x) = x^2        g(x) = sin(x) + 5')
xlabel('x')
ylabel('f(x)    g(x)')

% Label the curve for function f in red
text(-2.6, 7.7, 'f(x) = x^2', 'Color', 'r');

% Label the curve for function g in blue
text(-2.6, -0.75, 'g(x) = 5*sin(x)+5', 'Color', 'b')

% Put markers at the two points where the
functions are equal
xeq(1) = -0.956;
yeq(1) = 0.916;
xeq(2) =    2.685;
yeq(2) = 7.207;
plot(xeq,  yeq,  'o',  'MarkerFaceColor',  'k',
'MarkerEdgeColor','k', ...
      'LineWidth', 2, 'MarkerSize', 6)

% Label the points where the curves are equal
in black
text(-0.7, 1    , 'f(x) = g(x)', 'Color', 'k')
text(1.8 , 7.2, 'f(x) = g(x)', 'Color', 'k')
```



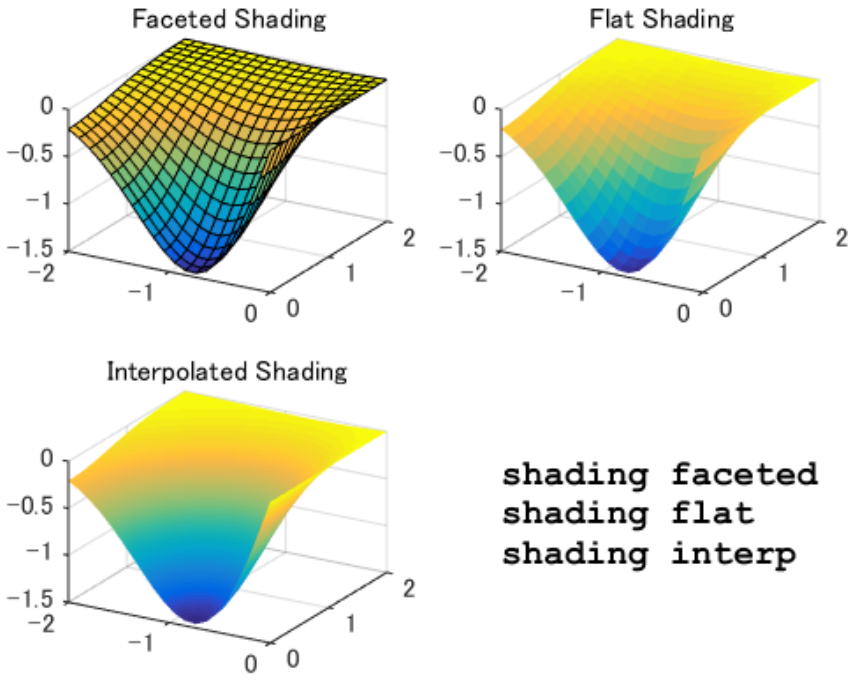
Adding Latex to Plots

```
% Calculate the Fibonacci numbers from 1 to 12
fib = zeros(1, 12);
for i = 1:12
    fib(i) = (((1+sqrt(5))/2)^i - ((1-sqrt(5))/2)^i)/sqrt(5);
end

% Plot the first 12 Fibonacci numbers
figure
plot(1:12, fib, 'k^-')

% Add a title and axis labels
title('Fibonacci Numbers from 1-12')
xlabel('n')
ylabel('F_n')

% Build a string that contains the Latex expression
eqtext = '$$F_n={1 \over \sqrt{5}}';
eqtext = [eqtext '\left[\left({1+\sqrt{5}\over 2}\right)^n -'];
eqtext = [eqtext '\left({1-\sqrt{5}\over 2}\right)^n\right]$$'];

% Add the string containing the Latex expression to the plot
text(0.5, 125, eqtext, 'Interpreter', 'Latex', 'FontSize', 12, 'Color', 'k')
```



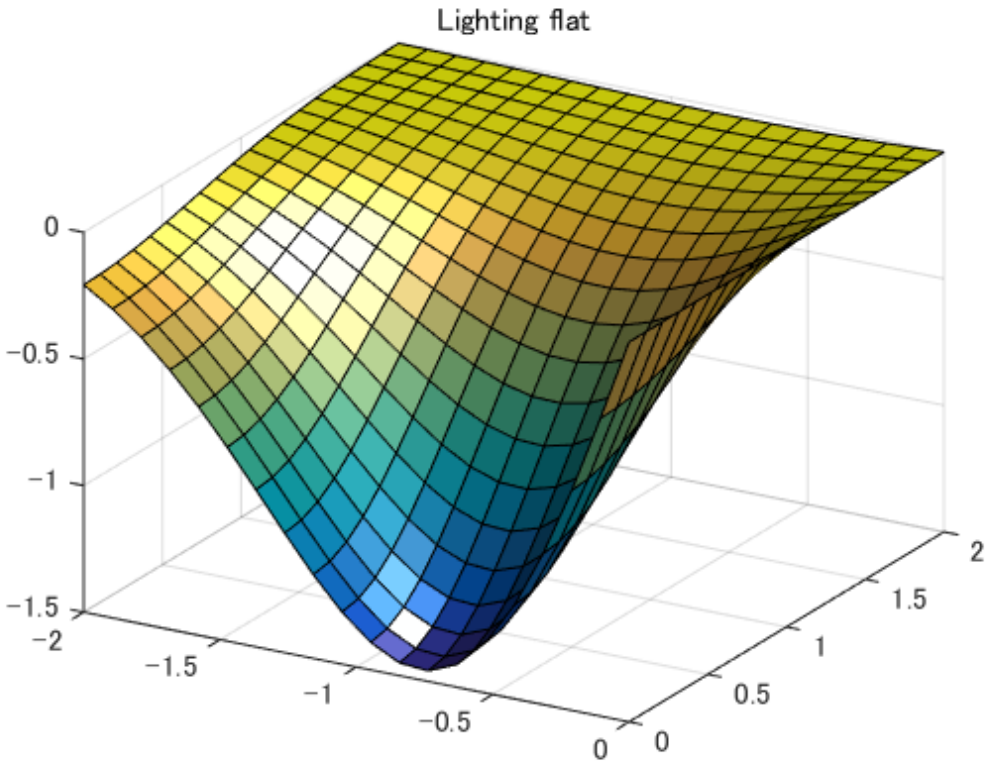
shading

```
% Create a grid of x and y points
points = linspace(-2, 0, 20);
[X, Y] = meshgrid(points, -points);

% Define the function Z = f(X,Y)
Z = 2./exp((X-.5).^2+Y.^2)-2./exp((X+.5).^2+Y.^2);

% Faceted Shading
subplot(2, 2, 1)
surf(X, Y, Z)
view(30, 30)
shading faceted
title('Faceted Shading')

% Flat Shading
subplot(2, 2, 2)
surf(X, Y, Z)
view(30, 30)
shading flat
title('Flat Shading')

% Interpolated Shading
subplot(2, 2, 3)
surf(X, Y, Z)
view(30, 30)
shading interp
title('Interpolated Shading')

% Shading Commands
subplot(2, 2, 4, 'Visible', 'off')
text(0, .5, sprintf('%s\n%s\n%s', ...
    'shading faceted', 'shading flat', 'shading interp'), ...
    'VerticalAlignment', 'middle', ...
    'FontName', 'Courier New', ...
    'FontWeight', 'bold', ...
    'FontSize', 12)
```



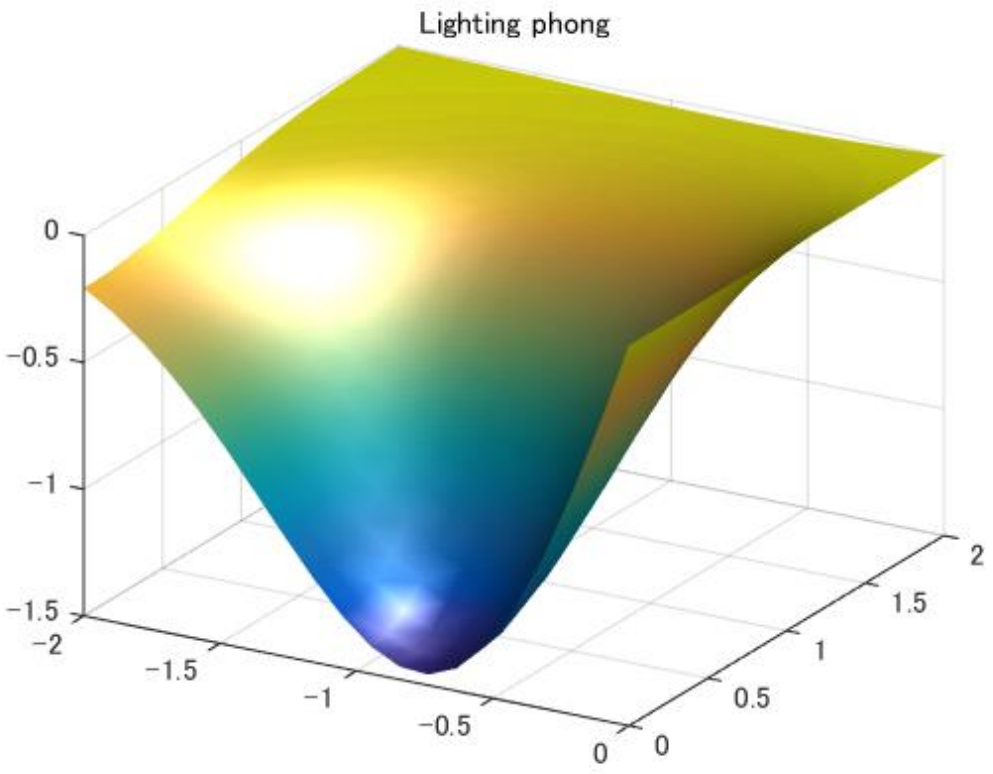
light, lighting

```
% Create a grid of x and y points
points = linspace(-2, 0, 20);
[X, Y] = meshgrid(points, -points);

% Define the function Z = f(X,Y)
Z = 2./exp((X-.5).^2+Y.^2)-2./exp((X+.5).^2+Y.^2);

% "flat" lighting is good for faceted surfaces
surf(X, Y, Z)
view(30, 30)
shading faceted
light
lighting flat
title('Lighting flat')
```



Change Lighting to Phong

```
% Create a grid of x and y points
points = linspace(-2, 0, 20);
[X, Y] = meshgrid(points, -points);

% Define the function Z = f(X,Y)
Z = 2./exp((X-.5).^2+Y.^2)-2./exp((X+.5).^2+Y.^2);

% "phong" lighting is good for curved, interpolated surfaces. "gouraud"
% is also good for curved surfaces
surf(X, Y, Z)
view(30, 30)
shading interp
light
lighting phong
title('Lighting phong')
```



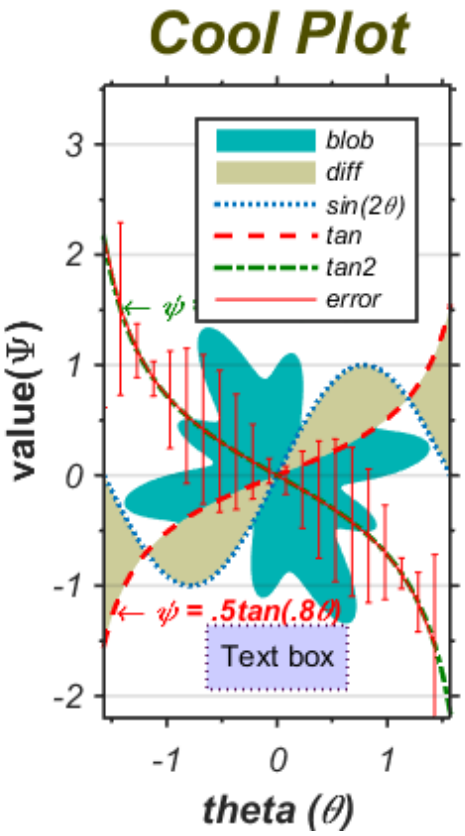
```
% Set up data
t    = 0:0.01:2*pi;
x1 = -pi/2:0.01:pi/2;
x2 = -pi/2:0.01:pi/2;
y1 = sin(2*x1);
y2 = 0.5*tan(0.8*x2);
y3 = -0.7*tan(0.8*x2);
rho = 1 + 0.5*sin(7*t).*cos(3*t);
x = rho.*cos(t);
y = rho.*sin(t);

% Create the left plot (filled plots, errorbars, texts)
figure
subplot(121)
hold on
h(1) = fill(x, y, [0 .7 .7]);
set(h(1), 'EdgeColor', 'none')

h(2) = fill([x1, x2(end:-1:1)], [y1, y2(end:-1:1)], [.8 .8 .6]);
set(h(2), 'EdgeColor', 'none')

h(3) = line(x1, y1, 'LineWidth', 1.5, 'LineStyle', ':');
h(4) = line(x2, y2, 'Linewidth', 1.5, 'LineStyle', '--', 'Color', 'red');
h(5) = line(x2, y3, 'Linewidth', 1.5, 'LineStyle', '-.', 'Color', [0 .5 0]);

% Create error bars
err = abs(y2-y1);
hh = errorbar(x2(1:15:end), y3(1:15:end), err(1:15:end), 'r');
h(6) = hh(1);

% Create annotations
text(x2(15), y3(15), '\leftarrow \psi = -.7tan(.8\theta)', ...
    'FontWeight', 'bold', 'FontName', 'times-roman', ...
    'Color', [0 0.5 0], 'FontAngle', 'italic')
text(x2(10), y2(10),'\leftarrow \psi = .5tan(.8\theta)', ...
    'FontWeight', 'bold', 'FontName', 'times-roman',...
    'Color', 'red', 'FontAngle', 'italic')

text(0, -1.65, 'Text box', 'EdgeColor', [.3 0 .3], ...
    'HorizontalAlignment', 'center', ...
    'VerticalAlignment', 'middle', 'LineStyle', ':', ...
    'FontName', 'palatino', 'Margin', 4, 'BackgroundColor', [.8 .8 1], ...
    'LineWidth', 1)
```

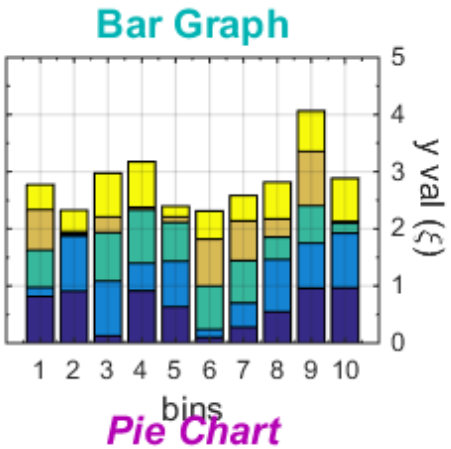
高级绘图

Area Bar Pie Charts with Annotations

```
% Adjust axes properties
axis equal
set(gca, 'Box', 'on', 'LineWidth', 1, 'Layer', 'top', ...
    'XMinorTick', 'on', 'YMinorTick', 'on', 'XGrid', 'off', 'YGrid', 'on', ...
    'TickDir', 'out', 'TickLength', [.015 .015], 'XLim', x1([1,end]),...
    'FontName', 'avantgarde', 'FontSize', 10, 'FontWeight', 'normal', ...
    'FontAngle', 'italic')

xlabel('theta (\theta)', 'FontName', 'bookman', 'FontSize', 12, ...
    'FontWeight', 'bold')
ylabel('value(\Psi)', 'FontName', 'helvetica', 'FontSize', 12, ...
    'FontWeight', 'bold', 'FontAngle', 'normal')
title('Cool Plot', 'FontName','palatino', 'FontSize', 18, ...
    'FontWeight', 'bold', 'FontAngle', 'italic', 'Color', [.3 .3 0])
legh = legend(h, 'blob', 'diff', 'sin(2\theta)', 'tan', 'tan2', 'error', 1);
set(legh, 'FontName', 'helvetica', 'FontSize', 8, 'FontAngle', 'italic')

% Create the upper right plot (bar chart)
subplot(222)
bar(rand(10,5), 'stacked')
set(gca, 'Box', 'on', 'LineWidth', .5, 'Layer', 'top', ...
    'XMinorTick', 'on', 'YMinorTick', 'on', 'XGrid', 'on', 'YGrid', 'on', ...
    'TickDir', 'in', 'TickLength', [.015 .015], 'XLim', [0 11], ...
    'FontName', 'helvetica', 'FontSize', 8, 'FontWeight', 'normal', ...
    'YAxisLocation', 'right')
xlabel('bins', 'FontName', 'avantgarde', 'FontSize', 10, ...
    'FontWeight', 'normal')
yH = ylabel('y val (\xi)', 'FontName', 'bookman', 'FontSize', 10, ...
    'FontWeight', 'normal');
set(yH, 'Rotation', -90, 'VerticalAlignment', 'bottom')
title('Bar Graph', 'FontName', 'times-roman', 'FontSize', 12, ...
    'FontWeight', 'bold', 'Color', [0 .7 .7])

% Create the bottom right plot (pie chart)
subplot(224)
pie([2 4 3 5], {'North', 'South', 'East', 'West'})
tP = get(get(gca, 'Title'), 'Position');
set(get(gca, 'Title'), 'Position', [tP(1), 1.2, tP(3)])
```

```
title('Pie Chart', 'FontName', 'avantgarde', 'FontSize', 12, ...
    'FontWeight', 'bold', 'FontAngle', 'italic', 'Color', [.7 0 .7])
th = findobj(gca, 'Type', 'text');
set(th, 'FontName', 'bookman', 'FontWeight', 'bold', 'FontAngle', 'italic')
```



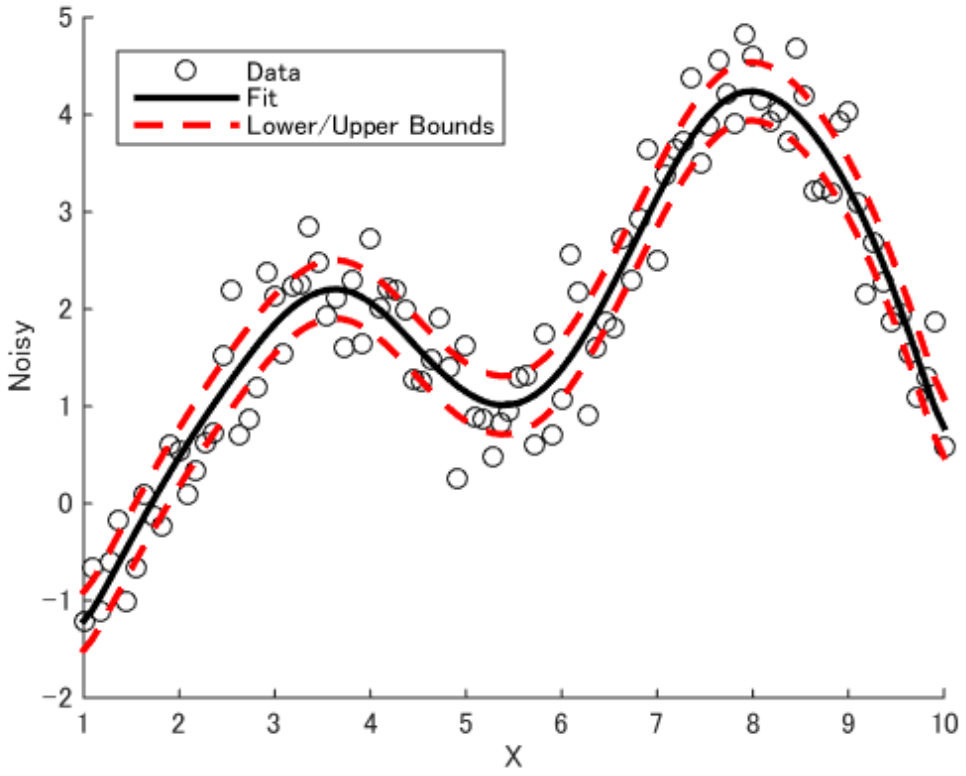


Colormap Chart

```
% Define built-in colormaps
maps = {};
if exist('parula', 'file')
    maps = {'parula'};
end
maps = [maps 'jet', 'hsv', 'hot', 'cool', 'spring', 'summer', 'autumn', ...
    'winter', 'gray', 'bone', 'copper', 'pink', 'lines'];

% Number of color levels to create
nLevels = 16;

figure

% X data points for color patches
xData = [linspace(0, 15, nLevels); linspace(1, 16, nLevels); ...
    linspace(1, 16, nLevels); linspace(0, 15, nLevels)];

% Create each color bar
for iMap = 1:length(maps)
    offset = 2*(length(maps) - iMap);
    yData = [zeros(2, nLevels); 1.5*ones(2, nLevels)] + offset;

    % Construct appropriate colormap.
    cData = feval(maps{iMap}, nLevels);

    % Display colormap chart
    patch('XData', xData, 'YData', yData, ...
        'EdgeColor', 'none', ...
        'FaceColor', 'flat', ...
        'FaceVertexCData', cData)
    rectangle('Position', [0, offset, 16, 1.5], ...
        'Curvature', [0 0])
    text(16, offset, sprintf(' %s', maps{iMap}), ...
        'VerticalAlignment', 'bottom', ...
        'FontSize', 12)
end

axis equal off
title('Built-in Colormaps')
```

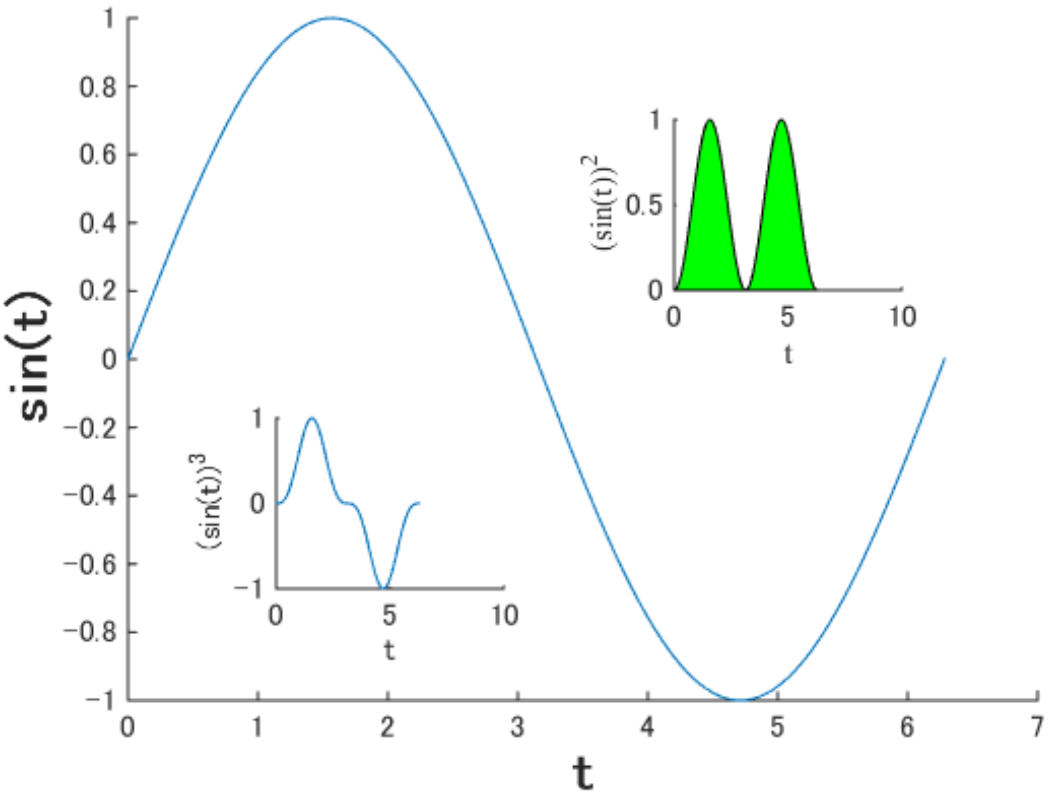
Curve with Lower and Upper Bounds

```
% Load the data for x, y, and yfit
load fitdata x y yfit

% Create a scatter plot of the original x and y data
figure
scatter(x, y, 'k')

% Plot yfit
line(x, yfit, 'Color', 'k', 'LineStyle', '-', 'LineWidth', 2)

% Plot upper and lower bounds, calculated as 0.3 from yfit
line(x, yfit + 0.3, 'Color', 'r', 'LineStyle', '--', 'LineWidth', 2)
line(x, yfit - 0.3, 'Color', 'r', 'LineStyle', '--', 'LineWidth', 2)

% Add a legend and axis labels
legend('Data', 'Fit', 'Lower/Upper Bounds', 'Location', 'NorthWest')
xlabel('X')
ylabel('Noisy')
```



Plot in Plot

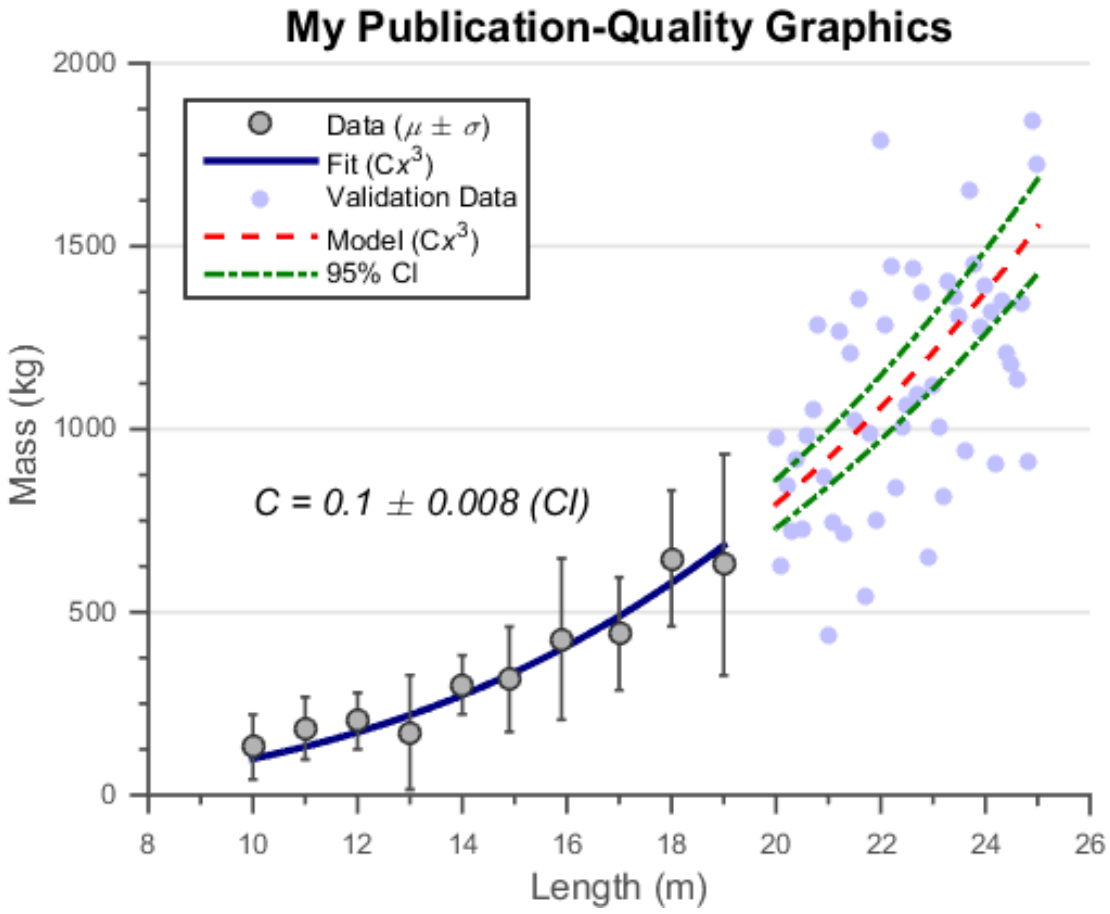
```
% Create data
t = linspace(0,2*pi);
t(1) = eps;
y = sin(t);

% Place axes at (0.1,0.1) with width and height of 0.8
figure
handaxes1 = axes('Position', [0.12 0.12 0.8 0.8]);

% Main plot
plot(t, y)
xlabel('t'); ylabel('sin(t)')
set(handaxes1, 'Box', 'off')

% Adjust XY label font
handxlabel1 = get(gca, 'XLabel');
set(handxlabel1, 'FontSize', 16, 'FontWeight', 'bold')
handylabel1 = get(gca, 'ylabel');
set(handylabel1, 'FontSize', 16, 'FontWeight', 'bold')

% Place second set of axes on same plot
handaxes2 = axes('Position', [0.6 0.6 0.2 0.2]);
fill(t, y.^2, 'g')
set(handaxes2, 'Box', 'off')
xlabel('t'); ylabel('(sin(t))^2')

% Adjust XY label font
set(get(handaxes2, 'XLabel'), 'FontName', 'Times')
set(get(handaxes2, 'YLabel'), 'FontName', 'Times')

% Add another set of axes
handaxes3 = axes('Position', [0.25 0.25 0.2 0.2]);
plot(t, y.^3)
set(handaxes3, 'Box','off')
xlabel('t'); ylabel('(sin(t))^3')
```



Publication Quality Graphics

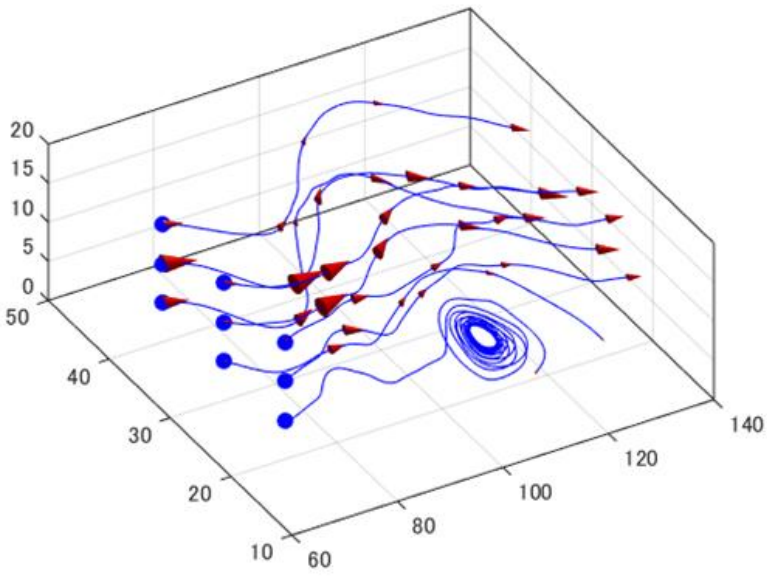
```
% Load data
load data xfit yfit xdata_m ydata_m ydata_s xVdata yVdata xmodel ymodel ...
      ymodelL ymodelU c cint

% Create basic plot
figure
hold on
hFit = line(xfit    , yfit);
hE = errorbar(xdata_m, ydata_m, ydata_s);
hData = line(xVdata, yVdata);
hModel = line(xmodel, ymodel);
hCI(1) = line(xmodel, ymodelL);
hCI(2) = line(xmodel, ymodelU);

% Adjust line properties (functional)
set(hFit, 'Color', [0 0 .5])
set(hE, 'LineStyle', 'none', 'Marker', '.', 'Color', [.3 .3 .3])
set(hData, 'LineStyle', 'none', 'Marker', '.')
set(hModel, 'LineStyle', '--', 'Color', 'r')
set(hCI(1), 'LineStyle', '-.', 'Color', [0 .5 0])
set(hCI(2), 'LineStyle', '-.', 'Color', [0 .5 0])

% Adjust line properties (aesthetics)
set(hFit, 'LineWidth', 2)
set(hE, 'LineWidth', 1, 'Marker', 'o', 'MarkerSize', 6, ...
      'MarkerEdgeColor', [.2 .2 .2], 'MarkerFaceColor' , [.7 .7 .7])
set(hData, 'Marker', 'o', 'MarkerSize', 5, ...
      'MarkerEdgeColor', 'none', 'MarkerFaceColor', [.75 .75 1])
set(hModel, 'LineWidth', 1.5)
set(hCI(1), 'LineWidth', 1.5)
set(hCI(2), 'LineWidth', 1.5)

% Add labels
hTitle = title('My Publication-Quality Graphics');
hXLabel = xlabel('Length (m)');
hYLabel = ylabel('Mass (kg)');

% Add text
hText = text(10, 800, ...
      sprintf('{\\itC = %0.1g \\pm %0.1g (CI)}', c, cint(2)-c));
```

```
% Add legend
hLegend = legend([hE, hFit, hData, hModel, hCI(1)], ...
    'Data ({\it\mu} \pm {\it\sigma})', 'Fit (C{\itx}^3)', ...
    'Validation Data', 'Model (C{\itx}^3)', '95% CI', ...
    'Location', 'NorthWest');

% Adjust font
set(gca, 'FontName', 'Helvetica')
set([hTitle, hXLabel, hYLabel, hText], 'FontName', 'AvantGarde')
set([hLegend, gca], 'FontSize', 8)
set([hXLabel, hYLabel, hText], 'FontSize', 10)
set(hTitle, 'FontSize', 12, 'FontWeight' , 'bold')

% Adjust axes properties
set(gca, 'Box', 'off', 'TickDir', 'out', 'TickLength', [.02 .02], ...
    'XMinorTick', 'on', 'YMinorTick', 'on', 'YGrid', 'on', ...
    'XColor', [.3 .3 .3], 'YColor', [.3 .3 .3], 'YTick', 0:500:2500, ...
    'LineWidth', 1)
```



Streamline

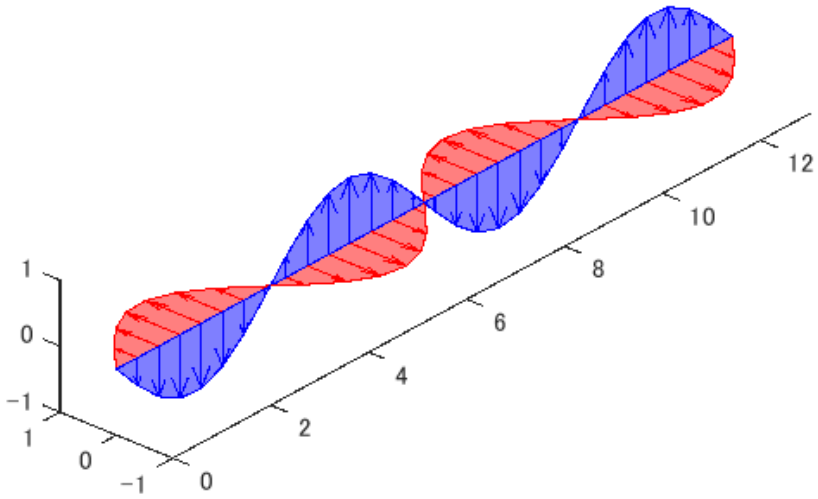
```
% Load wind data
load wind x y z u v w

figure

% Create streamline
[sx, sy, sz] = meshgrid(min(x(:)), linspace(20, 40, 3), linspace(5, 15, 3));
hhh = streamline(x, y, z, u, v, w, sx, sy, sz);
hold on

% Plot start point of the streamlines
plot3(sx(:), sy(:), sz(:), 'bo', 'MarkerFaceColor', 'b')

grid on
box on
view(-30, 60)

% Add velocity cones on top of the streamlines to indicate the velocity
% along the lines.

% Get X/Y/Z data for the stream lines
xx = get(hhh, 'XData');
yy = get(hhh, 'YData');
zz = get(hhh, 'ZData');

% Place 5 velocity cones per stream line
fcn = @(c) c(round(linspace(1, length(c), 5)));
xx = cellfun(fcn, xx, 'UniformOutput', false);
yy = cellfun(fcn, yy, 'UniformOutput', false);
zz = cellfun(fcn, zz, 'UniformOutput', false);
```




```
% Create coneplot
hhh2 = coneplot(x, y, z, u, v, w, [xx{:}], [yy{:}], [zz{:}], 3);
set(hhh2, 'FaceColor', 'r', 'EdgeColor', 'none')
camlight
lighting gouraud
```

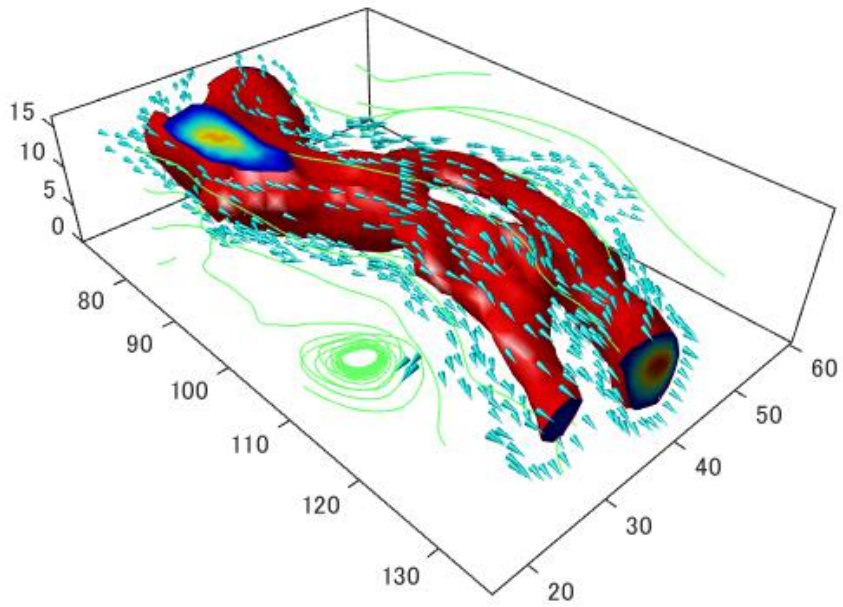
Two Waves

```
% Create some x data
x = linspace(0, 4*pi, 30);

% Create two waves to plot
y = sin(x);
z = sin(x-pi);

% Plot the first wave in red and fill with color
u = zeros(size(x));
figure
fill3(x, y, u, 'r', 'EdgeColor', 'r', 'FaceAlpha', 0.5)
hold on

% Add arrows for the first wave
quiver3(x, u, u, u, y, u, 0, 'r')

% Plot the first wave in blue and fill with color
fill3(x, u, z, 'b', 'EdgeColor', 'b', 'FaceAlpha', 0.5)

% Add the arrows for the second wave
quiver3(x, u, u, u, u, z, 0, 'b')

% Use equal axis scaling
view(-49,28)
axis square
daspect([1 1 1])
xlim([0 13])
ylim([-1 1])
zlim([-1 1])
```



win

```
% Load wind data
load wind x y z u v w

% Compute speed
spd = sqrt(u.*u + v.*v + w.*w);

figure

% Create isosurface patch
p = patch(isosurface(x, y, z, spd, 40));
isonormals(x, y, z, spd, p)
set(p, 'FaceColor', 'red', 'EdgeColor', 'none')

% Create isosurface end-caps
p2 = patch(isocaps(x, y, z, spd, 40));
set(p2, 'FaceColor', 'interp', 'EdgeColor', 'none')

% Adjust aspect ratio
daspect([1 1 1])

% Downsample patch
[f, verts] = reducepatch(isosurface(x, y, z, spd, 30), .2);

% Create coneplot (velocity cone)
h = coneplot(x, y, z, u, v, w, verts(:, 1),…
    verts(:, 2), verts(:, 3), 2);
set(h, 'FaceColor', 'cyan', 'EdgeColor', 'none')

% Create streamline
[sx, sy, sz] = meshgrid(80, 20:10:50, 0:5:15);
h2 = streamline(x, y, z, u, v, w, sx, sy, sz);
set(h2, 'Color', [.4 1 .4])

% Adjust colormap and axes settings
colormap(jet)
box on
axis tight
camproj perspective
camva(34)
campos([165 -20 65])
camtarget([100 40 -5])
camlight left
lighting gouraud
```



Animation

```
% Parameters and initial conditions
%      mass, link length, initial angles, simulation time
m = 1;
L = 1;
theta1 = 3*pi/4;
theta2 = 3*pi/8;
t = linspace(0, 10, 200);

% Solving ODE of a double pendulum
[T,Y] = ode45(@(t, x) double_pendulum(t, x, m, L), ...
       t, [theta1, theta2, 0, 0]);

% Calculating joint coordinates for animation purposes
x = [L*sin(Y(:,1)),    L*sin(Y(:,1))+L*sin(Y(:,2))];
y = [-L*cos(Y(:,1)), -L*cos(Y(:,1))-L*cos(Y(:,2))];

% Convert radians to degrees
ang = Y(:,1:2)*180/pi;

% Set up first frame
figure('Color', 'white')
subplot(2,1,1)
plot(T, ang, 'LineWidth', 2)
hh1(1) = line(T(1), ang(1,1), 'Marker', '.', 'MarkerSize', 20, ...
       'Color', 'b');
hh1(2) = line(T(1), ang(1,2), 'Marker', '.', 'MarkerSize', 20, ...
       'Color', 'r');
xlabel('time (sec)')
ylabel('angle (deg)')

subplot(2,1,2)
hh2 = plot([0, x(1,1);x(1,1), x(1,2)], [0, y(1,1);y(1,1), y(1,2)], ...
       '.-', 'MarkerSize', 20, 'LineWidth', 2);
axis equal
axis([-2*L 2*L -2*L 2*L])
ht = title(sprintf('Time: %0.2f sec', T(1)));

% Get figure size
pos = get(gcf, 'Position');
width = pos(3);
height = pos(4);
```

```
% Preallocate data (for storing frame data)
mov = zeros(height, width, 1, length(T), 'uint8');

% Loop through by changing XData and YData
for id = 1:length(T)
    % Update graphics data. This is more efficient than recreating plots.
    set(hh1(1), 'XData', T(id), 'YData', ang(id, 1))
    set(hh1(2), 'XData', T(id), 'YData', ang(id, 2))
    set(hh2(1), 'XData', [0, x(id, 1)], 'YData', [0, y(id, 1)])
    set(hh2(2), 'XData', x(id, :), 'YData', y(id, :))
    set(ht, 'String', sprintf('Time: %0.2f sec', T(id)))

    % Get frame as an image
    f = getframe(gcf);

    % Create a colormap for the first frame. For the rest of the frames,
    % use the same colormap
    if id == 1
        [mov(:,:,1,id), map] = rgb2ind(f.cdata, 256, 'nodither');
    else
        mov(:,:,1,id) = rgb2ind(f.cdata, map, 'nodither');
    end
end

% Create animated GIF
imwrite(mov, map, 'animation.gif', 'DelayTime', 0, 'LoopCount', inf)
```