

Assignment 7: Plots and Functions in Matlab

Date Due: March 31, 2017

Instructor: Trani

Problem 1

Select a problem of your choice that you have done in another CEE class. The problem requires that you do some computations to obtain a result given a set of parameters and assumptions.

Task 1

Explain the problem in simple words in one sentence.

Task 2

Create a Matlab function to solve the problem explained in Task 1. The function requires at least two input parameters and one output variable. Test the function with numerical values

Task 3

Create a Matlab script (not a function) that calls the function created in Task 2. The script should perform a parametric analysis of one of the input variables of the function created in Task 2 and obtain multiple solutions of the output variable of your function. See the example worked in class with function [cantileverBeam](#) and its associated script file called [callingCantileverBeam](#).

Task 4

Use the results of the parametric analysis in Task 3 to make a plot. The plot should have the variations in the input variable and the results of the output variable of your function created in Task 2 in a single graph.

Problem 2

To solve this problem, you need the car data file included in Week 11 of the course syllabus. Retrieve the car data file and save the contents in a text file. Name the file carData.txt.

Task 1

Create a Matlab script to read the car data text file created. Create individual variables for each data field (i.e., column) contained in the car data file. Make a combo plot that contains two figures in one window. The first figure plots car weight vs horsepower and the second one car weight vs. gas tank size.

```
% Import Car Data
```

```
% Script for importing data from the following spreadsheet:
```

```
%
```

```
%% Import the data
```

```
[~, ~, raw] = xlsread('/Users/atrani/University work/courses/cee3804/Practice Files 2013/carData.xlsx','carData.txt');
```

```
raw = raw(2:end,:);
```

```
raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw)) = {};
```

```
cellVectors = raw(:,[1,2,3]);
```

```
raw = raw(:,[4,5,6,7,8]);
```

```

%% Create output variable
data = reshape([raw{:}],size(raw));

%% Allocate imported array to column variable names
Model = cellVectors(:,1);
Country = cellVectors(:,2);
Type = cellVectors(:,3);
Weight = data(:,1);
TurningCircle = data(:,2);
Displacement = data(:,3);
Horsepower = data(:,4);
GasTankSize = data(:,5);

%% Clear temporary variables
clearvars data raw cellVectors;

```

```

% Plot of Weight and the Horesepower

```

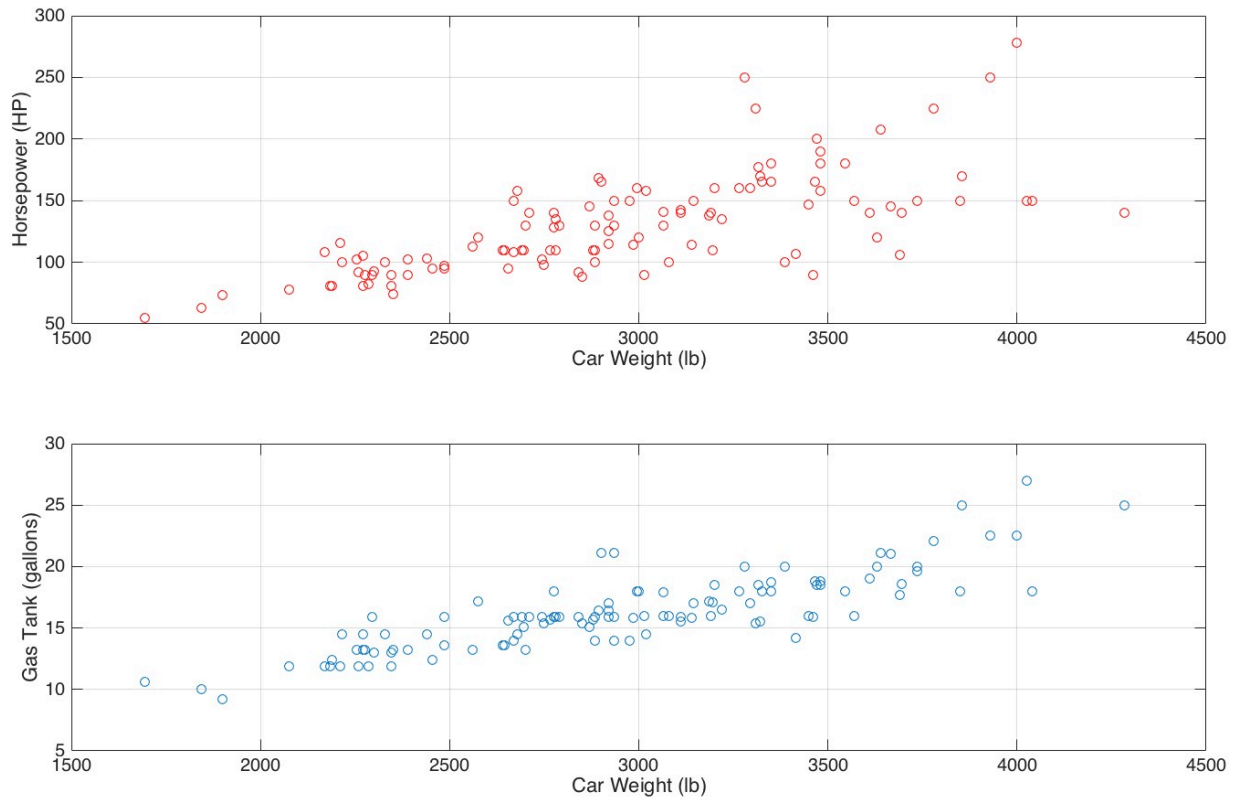
```

figure
subplot(2,1,1)
plot(Weight,Horsepower,'or')
xlabel('Car Weight (lb)', 'fontsize',24)
ylabel('Horsepower (HP)', 'fontsize',24)
grid

subplot(2,1,2)
plot(Weight,GasTankSize,'or')
xlabel('Car Weight (lb)', 'fontsize',24)
ylabel('Gas Tank (gallons)', 'fontsize',24)
grid

```

Figure 1. Car Data Plot.



Task 2

Create a separate plot of car weight vs. gas tank and use the Basic Curve Fitting function in the plot to estimate the best second order polynomial that fits the data. In your plot show the residuals in a subplot. Explain the meaning of the residuals.

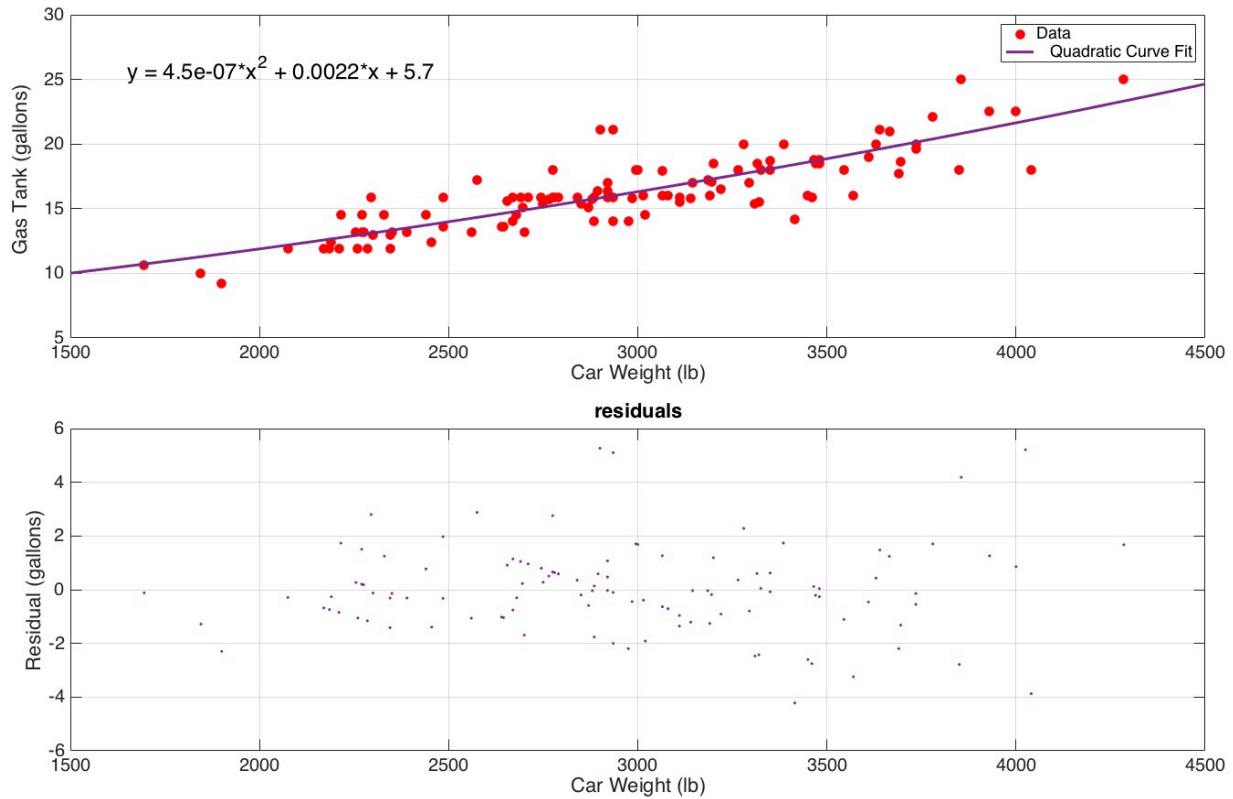


Figure 2. Car Data Plot with Residuals.

Task 3

Using the Matlab polynomial functions, estimate the best second order polynomial that fits the car data (car weight vs. gas tank size). Store the second order polynomial coefficients in a variable and evaluate the second order polynomial for values of car weight ranging from 2,000 to 4,500 pounds. Plot the results of the polynomial evaluation against the original data.

```
coeff = polyfit(Weight, GasTankSize, 2);

% Create a new weight vector to compare the polynomial fitted with the data
newWeight = 2000:100:4500;
estimatedGasTankSize = polyval(coeff, newWeight);

% Make a plot and compare to original data
figure
plot(Weight, GasTankSize, 'or')
xlabel('Car Weight (lb)', 'fontsize', 24)
ylabel('Gas Tank (gallons)', 'fontsize', 24)
hold on
plot(newWeight, estimatedGasTankSize, '^--b')
grid
```

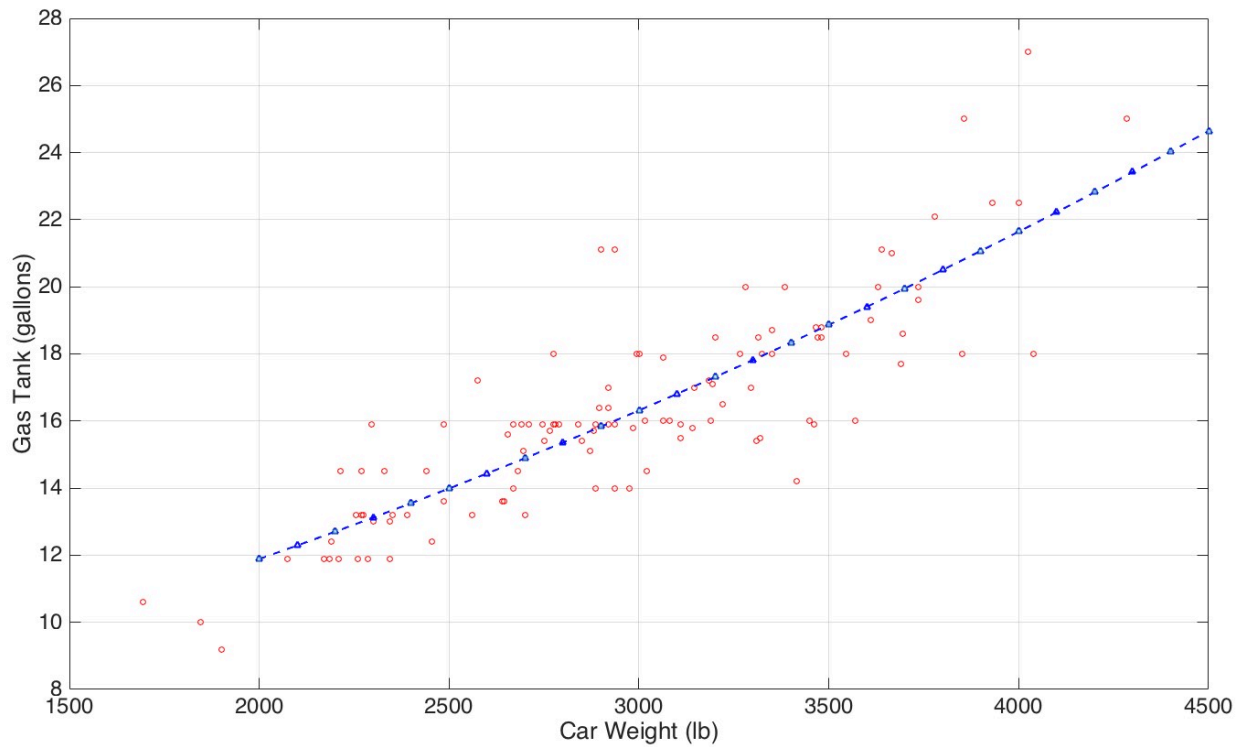


Figure 3. Second Order Polynomial Fit.

Task 4

Use the coefficients of the second order polynomial obtained in Task 3 to estimate the size of the gas tank for a 4,180 lb car.

Task 5

Use the Matlab polynomial functions to determine the polynomial fit to the relationships between car weight and turning circle. Comment if there is any statistical relation between the two variables.

Problem 3

Retrieve the GPS data file included in the homework page. The file contains GPS data collected in Arizona during a traffic study. A sample of the data is shown below.

% GPS car data collected in rural roads in Arizona

% Car = Toyota Corolla (1992)

% Column 1 = Time of observation (seconds)

% Column 2 = Distance traveled (m)

% Column 3 = Speed (km/hr)

% Column 4 = Acceleration (m/s-s)

0	0.0	0.0	0.0
2	23.0	41.4	0.54

```

4      52.0    52.1    1.48
6      82.9    55.6    0.49

```

Task 1

Create a Matlab script to read the GPD data. Create individual variables for each field contained in the data. Convert the speed of the car from km/hr to m/s. Plot the data using the subplot command and create three figure in one window: a) distance vs time, b) speed vs time and c) acceleration vs time.

```
% Script to read GPS data and perform tasks for A7 (S2017)
```

```
load GPSDataShort.m
```

```
% GPS car data collected in rural roads in Arizona
```

```
% Car = Toyota Corolla (1992)
```

```
% Column 1 = Time of observation (seconds)
```

```
% Column 2 = Distance traveled (m)
```

```
% Column 3 = Speed (km/hr)
```

```
% Column 4 = Acceleration (m/s-s)
```

```
% 0      0.0      0.0      0.0
```

```
% 2      23.0     41.4     0.54
```

```
% 4      52.0     52.1     1.48
```

```
% 6      82.9     55.6     0.49
```

```
% Task 1
```

```
% Create a Matlab script to read the GPD data. Create individual variables for each field
```

```
% contained in the data. Convert the speed of the car from km/hr to m/s.
```

```
% Plot the data using the subplot command and create three figures in one window:
```

```
% a) distance vs time, b) speed vs time and c) acceleration vs time.
```

```
convert_kmh_to_ms = 1000/3600;
```

```
time      = GPSDataShort(:,1);
```

```
distance  = GPSDataShort(:,2);
```

```
speed     = GPSDataShort(:,3);
```

```
acceleration = GPSDataShort(:,4);
```

```
speed_ms  = speed * convert_kmh_to_ms;
```

```
figure
```

```
subplot(3,1,1)
```

```
plot(time,distance,'o-r')
```

```
xlabel('Time (s)','fontsize',24)
```

```
ylabel('Distance (m)', 'fontsize', 24)
```

```
grid
```

```
subplot(3,1,2)
```

```
plot(time,speed_ms,'^-k')
```

```
xlabel('Time (s)', 'fontsize', 24)
```

```
ylabel('Speed (m/s)', 'fontsize', 24)
```

```
grid
```

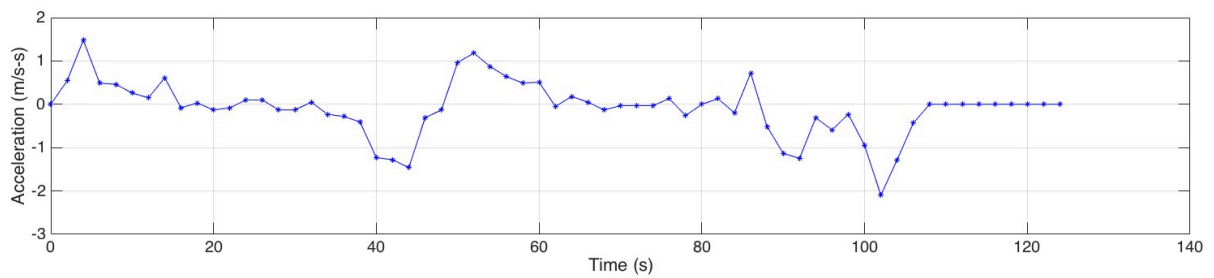
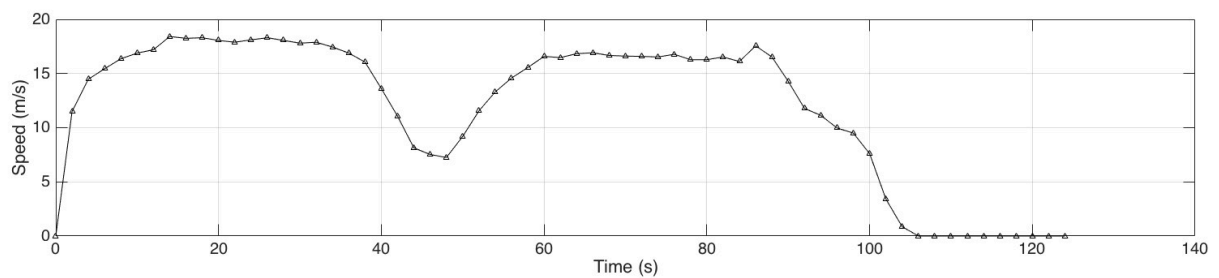
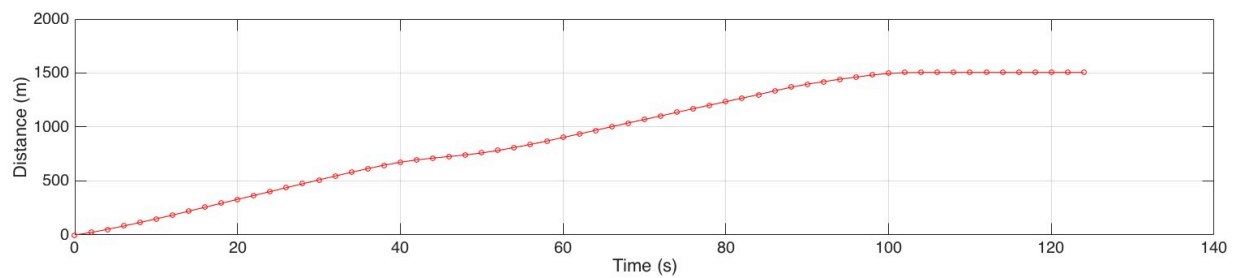
```
subplot(3,1,3)
```

```
plot(time,acceleration,'*-b')
```

```
xlabel('Time (s)', 'fontsize', 24)
```

```
ylabel('Acceleration (m/s-s)', 'fontsize', 24)
```

```
grid
```



Task 2

Use the numerical integration function TRAPZ in Matlab to estimate the distance traveled by the car during the first two minutes of travel. In this analysis, do a numerical integration of the speed vs time. Recall, distance traveled by a particle is the numerical integral of the speed profile. Report the distance traveled to the command window and include the units of distance. Compare the estimated distance to the one reported in column 1.

```
% Calculate the distance using the Trapezoidal rule function
```

```
distanceTraveled = trapz(time,speed_ms);  
clc  
disp(' ')  
disp(['Distance Traveled is = ',num2str(distanceTraveled) ])
```

Distance Traveled is = 1505.8333 meters

Task 3

Find the maximum speed (in km/hr) reached by the vehicle during the 2-minute period of travel. Find the maximum acceleration or deceleration in the same time period. Report both to the command window. Estimate the average speed (in m/s) for this car.

Task 4

Repeat Task 2 but now find the cumulative distance traveled using numerical integration every two seconds. Compare the results obtained with those shown in column 1.

Problem 4

Concrete is one of the most important materials used in Civil Engineering. A civil engineer wants to estimate the distribution of the concrete compressive strength for various specimens tested in the lab. The engineer collects data and finds that the compressive strength of concrete is normally distributed. The equation of the Probability Density Function (PDF) of the Gaussian (or Normal) distribution is given in any statistics textbook and repeated here for completeness (http://en.wikipedia.org/wiki/Normal_distribution).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left[\frac{(x-\mu)^2}{2\sigma^2}\right]}$$

where: x is the random variable in question (concrete compressive strength in psi) and μ and σ are the mean and standard deviation (in psi) of the random phenomena modeled. After testing 1000 samples, the engineer estimates the mean concrete compressive strength to be 11,300 psi. The standard deviation is found to be 405 psi.

The area under probability density function has a fundamental interpretation in random processes. For example, the area under the curve in Figure 1 can be interpreted as the probability that random variable x (i.e., concrete compressive strength) will take values between a and infinity.

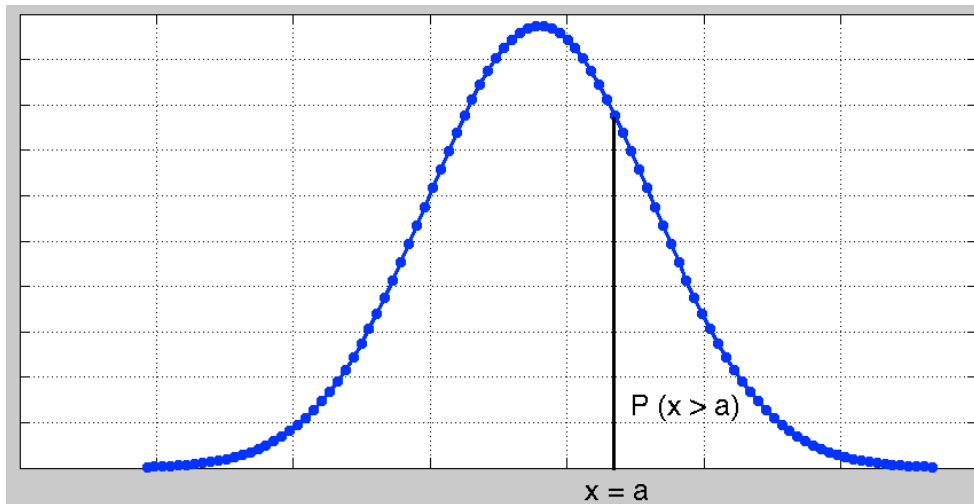


Figure 1. Sample Normal Probability Density Function. Interpretation of the Area Under the Curve.

Task 1

Evaluate the function $f(x)$ using the values of μ and σ observed. Plot the resulting PDF as a function of concrete compressive strength (in the x axis). Label accordingly. Verify that the PDF plot is the well-known bell-shape curve of the Normal distribution.

Task 2

Create a Matlab function to calculate the value of the Normal distribution PDF ($f(x)$). The function needs to take three arguments: x , μ and σ . The output of the function is $f(x)$.

```
% Function to evaluate the area under the normal distribution
```

```
function fx_normal = normal(x)
    global mu sigma

    fx_normal = 1./(sigma .* sqrt(2*pi)) .* exp(-1/2 .* ((x-mu)./sigma).^2 );
end
```

Script to use the function normal.m

*% Normal Distribution Analysis
% Concrete compressive strength*

*clear
clc*

% Function calls:

global mu sigma

% Define the parameter of the distribution (MU) and standard deviation (SIGMA)

*mu = 11300; % psi
sigma = 405; % psi*

% Define upper and lower bounds to get a nice plot of the Gaussian PDF

*npoints = 100;
parameter = 3.5;
low = mu - parameter*sigma;
high = mu + parameter*sigma;
interval = (high - low) / npoints;*

% Define random variable x - concrete compressive strength

x=low:interval:high;

% Define the function of the random variable x (PDF function)

*fx = 1/(sigma * sqrt(2*pi)) * exp(-1/2 * ((x-mu)/sigma).^2);*

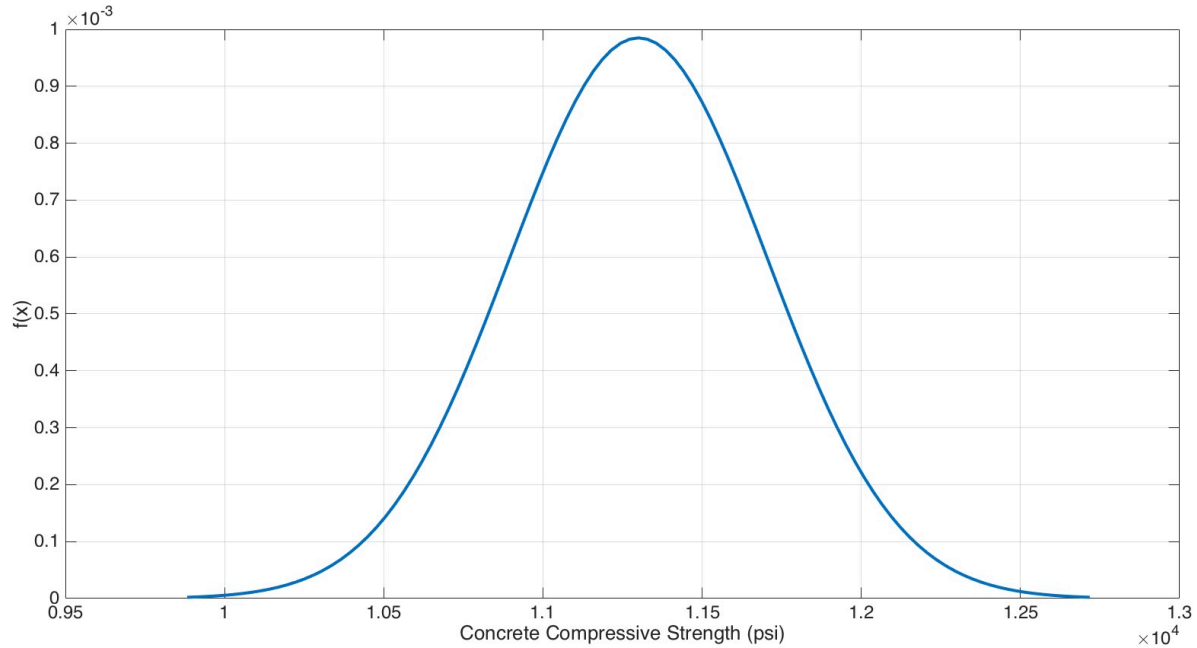
% Plot the random variable x versus the PDF function

*plot(x,fx)
xlabel('Concrete Compressive Strength (psi)', 'fontsize',24)
ylabel('f(x)', 'fontsize',24)
grid*

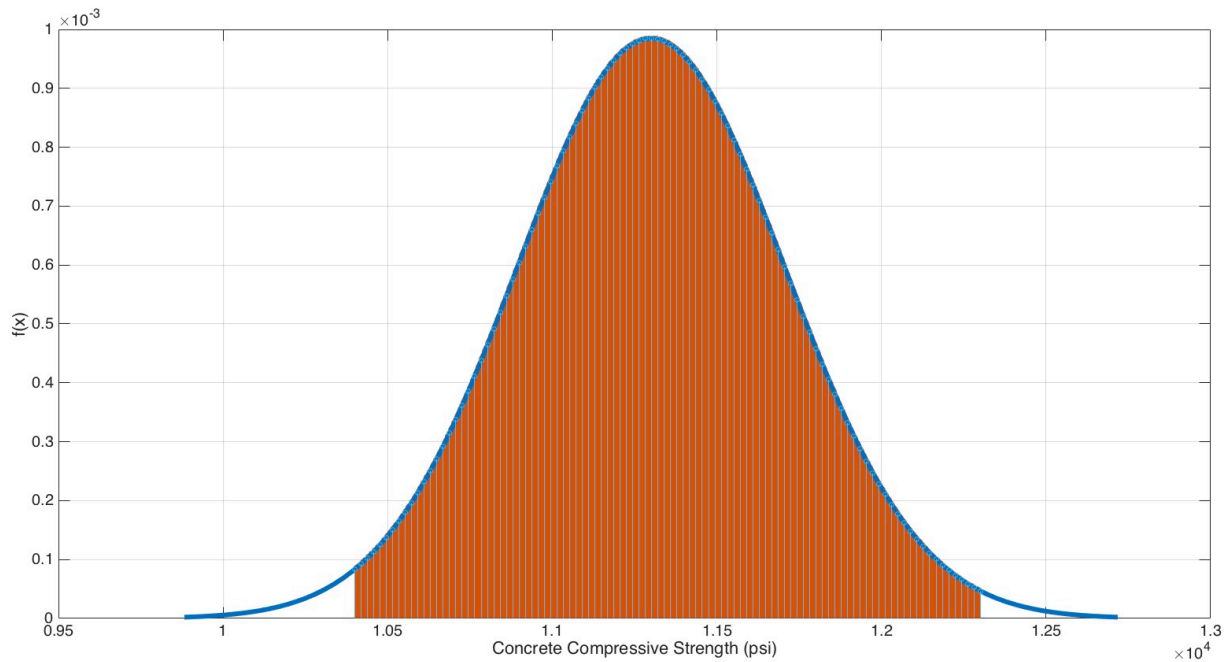
*% Compute area under the curve f(x) from -infinity to infinity
% Change the values of the last two arguments if you
% want another range of values*

*lowerLimit =9000;
upperLimit = 10200;
Pn=quad('normal',lowerLimit,upperLimit);*

*disp(' ')
disp([blanks(5), 'Probability between two limits of integration : '])
disp(' ')
disp([blanks(5), ' P(x>0)=', num2str(Pn)])*



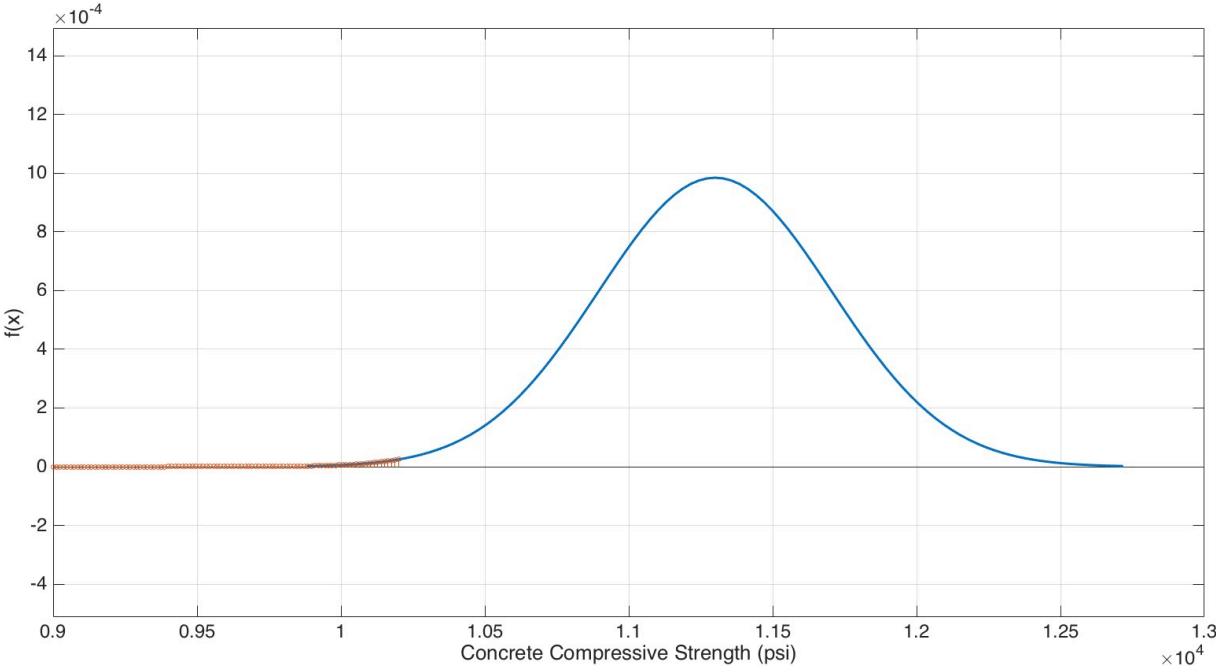
`disp('')`



Task 3

Create a separate Matlab script to evaluate the Matlab function created in Task 2 to estimate area under the curve between concrete compressive strength values of 10,400 and 12,300 psi. What is the probability that concrete compressive strength is below 10,200 psi?

The probability that concrete compressive strength is between 10,400 and 12,300 psi is $P(x>0)=0.99323$ (shown above)



The probability that concrete compressive strength is below 10,200 psi is $P(x>0)=0.0033034$ (shown above)