Homework Assignment Number Eight Solutions

Problem 1. (6 points)
Perform a single-variable linear regression using the model

$$y = b_0 + b_1 x$$

(a) Report the mean value and standard deviation of the slope, b_1.
(b) Report the mean value and standard deviation of the intercept, b_0.
(c) Report the measure of fit.
Use the data in the file "file.hw08p01.txt" available on the website.
Note the first column in the data file is *y*. The second column is *x*.

Solution
Solution in Python:
I used the function linreg1.py for linear regression with one independent variable.

I wrote the small script hw08_p01.py

```
import numpy as np
from linreg1 import *

datamat = np.loadtxt('file.hw08p01.txt')
ndata= datamat.shape[0]
print(f"There are {ndata} data points.")

x = datamat[:,1]
y = datamat[:,0]

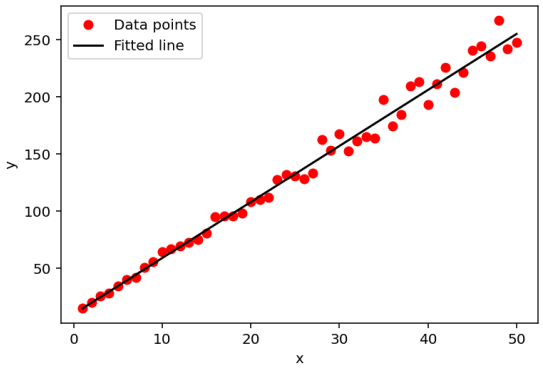
b, bsd, MOF = linreg1(x, y)
print(f"Coefficients: {b}")
print(f"Standard errors: {bsd}")
print(f"Measure of Fit (MOF): {MOF}")
```

At the command line prompt, I executed the script

```
run hw08_p01
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
There are 50 data points.
Coefficients: [10.07408311  4.8964732 ]
Standard errors: [2.34747056 0.08011798]
Measure of Fit (MOF): 0.9873121231309676
```



The code also generated a plot.
(a) The slope is 4.90 +/- 0.08
(b) The intercept is 10.07 +/- 2.35
(c) The MOF is 0.987

Solution in Matlab:
I used the code linreg1.m for linear regression with one independent variable.

I wrote the small script hw08p01.m

```
clear all;

M = [15.10065279    1
20.4980324  2
25.56136963 3
...
242.1551242 49
247.5311911 50];

n = 50;
y = M(1:n,1);
x = M(1:n,2);

[b,bsd,MOF] = linreg1(x, y)
```

At the command line prompt, I executed the script

```
>> hw08p01
```

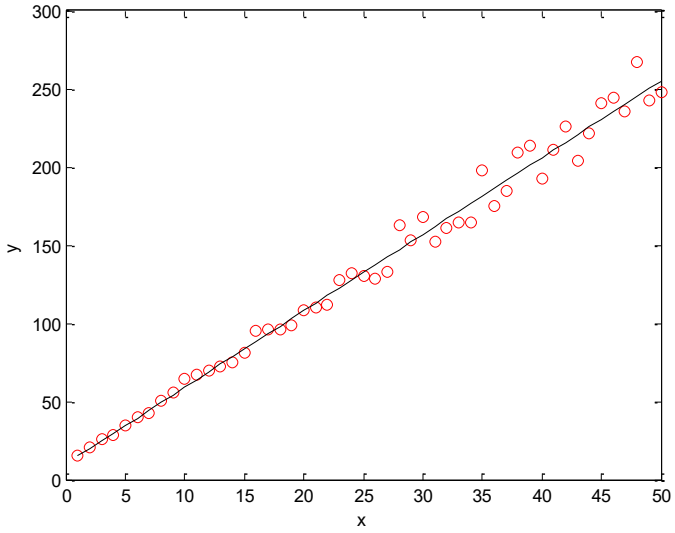
This generated the following output for the means, standard deviations and Measure of Fit.

```
b =
  10.074083107297966
   4.896473196043218

bsd =
   2.347470564943806
   0.080117982695322

MOF =
   0.987312123130968
```

The code also generated a plot.



Problem 2. (8 points)
Perform a single-variable polynomial regression using the model

$$y = b_0 + b_1 x + b_2 x^2$$

(a) Report the mean value and standard deviation of the regression coefficient for the quadratic term.
(b) Report the mean value and standard deviation of the regression coefficient for the linear term.
(c) Report the mean value and standard deviation of the regression coefficient for the constant term.
(d) Report the measure of fit.
Use the data in the file "file.hw08p02.txt" available on the website.
Note the first column in the data file is *y*. The second column is *x*.

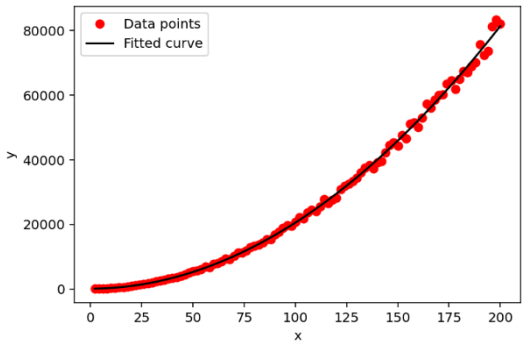
Solution
Solution in Python:
I used the function polyreg.py for polynomial regression with one independent variable.
I wrote the small script hw08_p02.py

```
import numpy as np
from polyreg import *

datamat = np.loadtxt('file.hw08p02.txt')
ndata= datamat.shape[0]
print(f"There are {ndata} data points.")
x = datamat[:,1]
y = datamat[:,0]
norder = 2

b, bsd, MOF = polyreg(norder, x, y)

print(f"Coefficients (b): {b}")
print(f"Standard deviations (bsd): {bsd}")
print(f"Measure of Fit (MOF): {MOF}")
```



At the command line prompt, I executed the script

```
run hw08_p02
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
There are 100 data points.
Coefficients (b): [86.57904445  2.86022206  2.01599001]
Standard deviations (bsd): [3.20497307e+02 7.32379907e+00 3.51272053e-02]
Measure of Fit (MOF): 0.9982025551082161
```

The code also generated a plot.
(a) The coefficient for the quadratic term is 2.016 +/- 0.035
(b) The coefficient for the linear term is 2.860 +/- 7.324
(c) The coefficient for the constant term is 86.579 +/- 320.0497
(d) The MOF is 0.998

Solution in Matlab:
I used the code linregn.m for linear regression with two independent variables.

I wrote the small script hw08p02.m

```
clear all;

M = [28.60333775    2
61.67477585 4
113.330215  6
...
83236.71749 198
81985.43984 200];

n = 100;
y = M(1:n,1);
x(1:n,1) = M(1:n,2);
x(1:n,2) = x(1:n,1).*x(1:n,1);
[b,bsd,MOF] = linregn(2,x, y)
```

In this code, I computed the second variable to be the first variable squared. At the command line prompt, I executed the script

```
>> hw08p02
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
b =
  86.579044451471418
   2.860222062350658
   2.015990010806434

bsd =   1.0e+02 *
   3.204973070484634
   0.073237990705217
   0.000351272052967

MOF =   0.998202555108216
```

The code does not generate a plot.

Problem 3. (8 points)
Perform a multivariate linear regression using the model

$$y = b_0 + b_1 x_1 + b_2 x_2$$

(a) Report the mean value and standard deviation of the regression coefficient for the first independent variable, x_1.
(b) Report the mean value and standard deviation of the regression coefficient for the second independent variable, x_2.
(c) Report the mean value and standard deviation of the regression coefficient for the constant term.
(d) Report the measure of fit.
Use the data in the file "file.hw08p03.txt" available on the website.
Note the first column in the data file is *y*. The second column is x_1. The third column is x_2.

Solution
Solution in Python:
I used the function linregn.py for multivariate linear regression with more than one independent variable. I wrote the small script hw08_p03.py

```
import numpy as np
from linregn import *

datamat = np.loadtxt('file.hw08p03.txt')
ndata= datamat.shape[0]
print(f"There are {ndata} data points.")

# nx = number of independent variables
nx = datamat.shape[1] - 1
print(f"There are {nx} independent variables.")
x = datamat[:,1:nx+1]
y = datamat[:,0]
b, bsd, MOF = linregn(nx, x, y)
print(f"Coefficients (b): {b}")
print(f"Standard deviations (bsd): {bsd}")
print(f"Measure of Fit (MOF): {MOF}")
```

At the command line prompt, I executed the script

```
run hw08_p03
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
There are 100 data points.  There are 2 independent variables.
Coefficients (b): [-19.42616901  -4.96880625   2.99019573]
Standard deviations (bsd): [1.10946128 0.06690303 0.01338061]
Measure of Fit (MOF): 0.9982359425668965
```

(a) The coefficient for the first independent variable is -4.969 +/- 0.067
(b) The coefficient for the second independent variable is 2.990 +/- 0.134
(c) The coefficient for the constant term is -19.426 +/- 1.109
(d) The MOF is 0.998

Solution in Matlab:
I used the code linregn.m for linear regression with two independent variables.

I wrote the small script hw08p03.m

```
clear all;

M = [-0.981903982   2   10
-11.36111514    4   10
-17.92444528    6   10
...
188.825984  18  100
181.0279417 20  100];

n = 100;
y = M(1:n,1);
x(1:n,1) = M(1:n,2);
x(1:n,2) = M(1:n,3);
[b,bsd,MOF] = linregn(2,x, y)
```

At the command line prompt, I executed the script

```
>> hw08p03
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
b =
 -19.426169012360049
  -4.968806248543629
   2.990195728881456

bsd =
   1.109461283774494
   0.066903032685564
   0.013380606537113

MOF =   0.998235942566897
```

The code does not generate a plot.

Problem 4. (6 points)

Consider the isomerization reaction:

$$A \rightarrow B$$

The reaction rate is given by

$$rate = C_A k_o e^{-\frac{E_a}{RT}} \quad \text{[moles/liter/minute]}$$

where

concentration of A: C_A [moles/liter]
prefactor: k_o [1/minute]
activation energy for reaction: E_a [Joules/mole]
constant: $R = 8.314$ [Joules/mole/K]
temperature: T [K]

The reaction is measured at a constant concentration of A, C_A=0.1 mol/liter, over a variety of temperatures. The rate is recorded. The rate as a function of temperature is given in tabular form in the file "file.hw08p04.txt" (containing 108 data points).

(a) Convert this equation to a form, which is linear in rate constants, k_o and E_a.
(b) Report the mean and standard deviation of the activation energy, E_a.
(c) Report the mean and standard deviation of the reaction rate prefactor, k_o.

Solution

(a) Convert this equation to a form, which is linear in rate constants, k_o and E_a.

$$ln(rate) - ln(C_A) = -\frac{E_a}{RT} + ln(k_o)$$

This equation is of the form: $y = b_1 x + b_0$where

$$y = ln(rate) - ln(C_A),\ b_1 = E_a,\ x = -\frac{1}{RT},\ \text{and}\ b_0 = ln(k_o).$$

Solution in Python:
I used the function linreg1.py for linear regression with one independent variable.

I wrote the small script hw08_p04.py

```
import numpy as np
import math
from linreg1 import *

nheader=2
datamat = np.loadtxt('file.hw08p04.txt', skiprows=nheader)
ndata= datamat.shape[0]
print(f"There are {ndata} data points.")
```

```
temp = datamat[:,0]
rate = datamat[:,1]

R = 8.314 # J/mol/K
CA = 0.1 # mol/liter

x = np.zeros(ndata)
y = np.zeros(ndata)

for i in range(ndata):
    x[i] = -1.0 / (R * temp[i])
    y[i] = np.log(rate[i]) - np.log(CA)

b, bsd, MOF = linreg1(x, y)
print(f"Coefficients: {b}")
print(f"Standard errors: {bsd}")
print(f"Measure of Fit (MOF): {MOF}")

Ea = b[1]
ko = math.exp(b[0])
Easd = bsd[1]
kosd = bsd[0]/b[0]*ko

print(f"Ea =  {Ea} +/- {Easd}")
print(f"ko =  {ko} +/- {kosd}")
```

At the command line prompt, I executed the script

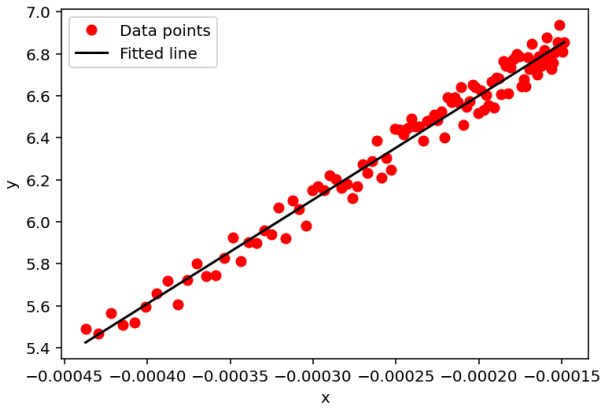
```
run hw08_p04
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
There are 108 data points.
Coefficients: [   7.59088286 4952.40488105]
Standard errors: [1.85387026e-02 7.25258982e+01]
Measure of Fit (MOF): 0.9777721491426371
Ea =  4952.404881054961 +/- 72.52589820299062
ko =  1980.0608526899346 +/- 4.835769435329081
```

The code also generated a plot.
(b) The activation energy is 4952+/- 73
(c) The prefactor is 1980 +/- 5

The MOF is 0.978



Solution in Matlab:
I used the code linreg1.m for linear regression with one independent variable.
I wrote the small script hw08p04.m

```
clear all;

%Temperature    rate of A loss
%K              mole/liter/min

M = [275    24.19186881
280 23.66394411
285 26.15820944
...
805 90.60688637
810 94.70386442];

R = 8.314; % J/mol/K
CA = 0.1; % mol/liter
n = 108;
for i = 1:1:n
    x(i) = -1.0/(R*M(i,1));
    y(i) = log(M(i,2)) - log(CA);
end
[b,bsd,MOF] = linreg1(x, y)
Ea = b(2)
ko = exp(b(1))
```

At the command line prompt, I executed the script

```
>> hw08p04
```

This generated the following output for the means, standard deviations and Measure of Fit.

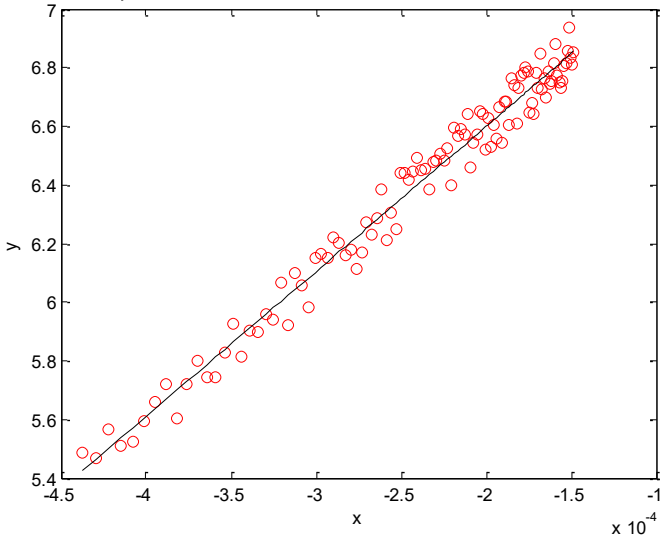
```
b =   1.0e+03 *
   0.007590882856898
   4.952404881054963

bsd =
   0.018538702614459
  72.525898202990689

MOF =   0.977772149142637

Ea =     4.952404881054963e+03

ko =     1.980060852689931e+03
```



The activation energy was 4950 J/mol.
The rate constant was 1980 1/min.

The code also generated a plot.

Problem 5. (12 points)
On the project page of the course website, there is a zip file containing data for the "Data Wrangling with Carbon Fiber Mechanical Properties". This file contains the results from many tensile test experiments on carbon fibers. For the first experiment in the first data set **only**, determine

(a) strain at fracture
(b) ultimate tensile strength
(c) elastic modulus average and standard deviation
(d) report the limits in terms of strain that you chose for your regression and justify.
(e) provide a plot of the entire raw data
(f) provide a plot of the relevant data to (a), (b) and (c) and indicate strain at fracture, ultimate tensile strength and modulus regression on the plot.

Solution
Solution in Python:
To perform this task, I wrote a script, cfiber_script_1_file.py, provided below, which includes the code distributed to read the project data files. The code is modified in two ways. First it only reads the first repetition of the first case. Second, if evaluates the three mechanical properties of interest as required in this problem and generates the plots.

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from linreg1 import linreg1

# Number of data sets
# hardcode in only 1 case
ncase = 1

# Data directory and file base names
data_dir_base = 'data_set_'
data_file_base = 'Specimen_RawData_'

# Column numbers (MATLAB is 1-indexed, Python is 0-indexed)
colstrain = 3  # MATLAB column 4 -> Python index 3
colstress = 4  # MATLAB column 5 -> Python index 4

# number of header lines in data file before data appears
nheader=7

# Initialize repetitions per case
nrep = [0] * ncase

# Loop over data sets
figcount = 0
for ic in range(1, ncase + 1):
    # Construct directory name
    dirname = f"{data_dir_base}{ic}"
#    print(f"ic = {ic} dirname = {dirname}")
```

```
    # List CSV files in directory
    if not os.path.isdir(dirname):
#        print(f"Directory {dirname} does not exist.")
        continue

    file_list = [f for f in os.listdir(dirname) if f.endswith('.csv')]
    nrep[ic - 1] = len(file_list)
# hardcode in only 1 replicate
    nrep[ic-1] = 1
    print(f"ic = {ic} dirname = {dirname} nrep = {nrep[ic - 1]}")

    # New figure for each dataset
    figcount += 1
    plt.figure(figcount)

    # Loop over repetitions
    for ir in range(1, nrep[ic - 1] + 1):
        # Construct full file path
        filename = f"{data_file_base}{ir}.csv"
        filepath = os.path.join(dirname, filename)
 #       print(f"ic = {ic} dirname = {dirname} nrep = {nrep[ic - 1]} ir =
{ir} file = {filename}")

        if not os.path.isfile(filepath):
            print(f"File {filepath} not found.")
            continue

        # Read CSV file
        # the skiprows option causes it to skip the header lines
        # the thousands option tells it to ignore commas in the csv file
        tempdata = pd.read_csv(filepath, skiprows=nheader, thousands=',')
        nrow = tempdata.shape[0]

  #      print(f"case {ic}, rep {ir}, file {filepath} has {nrow} rows of
data.")

        # Extract strain and stress
        strain = tempdata.iloc[:, colstrain]
        stress = tempdata.iloc[:, colstress]

# Homework 8 problem 5 modifications start here

# STEP ONE.  Determine Strain at Failure

        ndata = nrow
        threshold = 0.2
        ifoundit = False
        icount = 0

        while not ifoundit and icount < ndata - 1:
            icount += 1
            deltastress = (stress[icount] - stress[icount - 1]) /
stress[icount - 1]
            if deltastress < -threshold:
                ifoundit = True
```

```
        indexfrac = icount
        strainfrac = strain[icount]
        stressfrac0 = stress[icount - 1]
        stressfrac1 = stress[icount]

        print(f"Strain at failure occurs at point {indexfrac} with strain
{strainfrac:.4e} "
              f"where the stress changes from {stressfrac0:.4e} to
{stressfrac1:.4e} MPa")

# STEP TWO.  Determine Ultimate Tensile Stress

        indexlo = 0
        indexhi = indexfrac

        stressuts = np.max(stress[indexlo:indexhi+1])
        indexuts = np.argmax(stress[indexlo:indexhi+1]) + indexlo  # add
offset due to slicing
        strainuts = strain[indexuts]

        print(f"Ultimate Tensile Stress occurs at point {indexuts} with
strain {strainuts:.4e} "
              f"where the stress is {stressuts:.4e} MPa")


# STEP THREE.  DETERMINE MODULUS OF ELASTICITY via REGRESSION

        indexreglo = 0
        indexreghi = indexuts

        (b0, b1), (b0_sd, b1_sd), MOF =
linreg1(strain[indexreglo:indexreghi+1],

stress[indexreglo:indexreghi+1])

        modulusav = b1
        modulussd = b1_sd
        intercept = b0

        print(f"Elastic modulus is {modulusav:.4e} +/- {modulussd:.4e} with
MOF = {MOF:.4f}.")

# STEP FOUR.  CREATE PLOT

        plt.figure(figsize=(10, 6))
        plt.plot(strain, stress, 'k-', linewidth=2, label='data')

        plt.xlabel('strain')
        plt.ylabel('stress (MPa)')

        # Set axis limits
        stressmin = np.min(stress[indexlo:min(indexhi+10, ndata)])
        ylo = min(0, stressmin)
        yhi = stressuts * 1.1
        plt.axis([0, strainfrac * 1.1, ylo, yhi])

        # Plot strain at fracture line
```

```
        plt.plot([strainfrac, strainfrac], [ylo, yhi], 'r:', linewidth=2,
label='strain at fracture')

        # Plot UTS point
        plt.plot(strainuts, stressuts, 'ro', linewidth=2, label='ultimate
tensile stress')

        # Plot regression line
        x_vals = [strain[indexreglo], strain[indexreghi]]
        y_vals = [modulusav * x + intercept for x in x_vals]
        plt.plot(x_vals, y_vals, 'b-', linewidth=2, label='modulus
regression')

        plt.legend(loc='upper left')
        plt.title("Stress-Strain Curve Analysis")
        plt.grid(True)
        plt.tight_layout()
        plt.show()


# Homework 8 problem 5 modifications end here


        print(f"case {ic}, rep {ir}, file {filepath} has strain from
{strain.iloc[0]:.3e} to {strain.iloc[-1]:.3e} "
              f"& stress from {stress.iloc[0]:.3e} to {stress.iloc[-
1]:.3e}.")

        # Plot stress vs strain
        plt.plot(strain, stress, 'k-')

    plt.xlabel('strain')
    plt.ylabel('stress (MPa)')
    plt.title(f'Data Set {ic}')
    plt.grid(True)


plt.show()
```

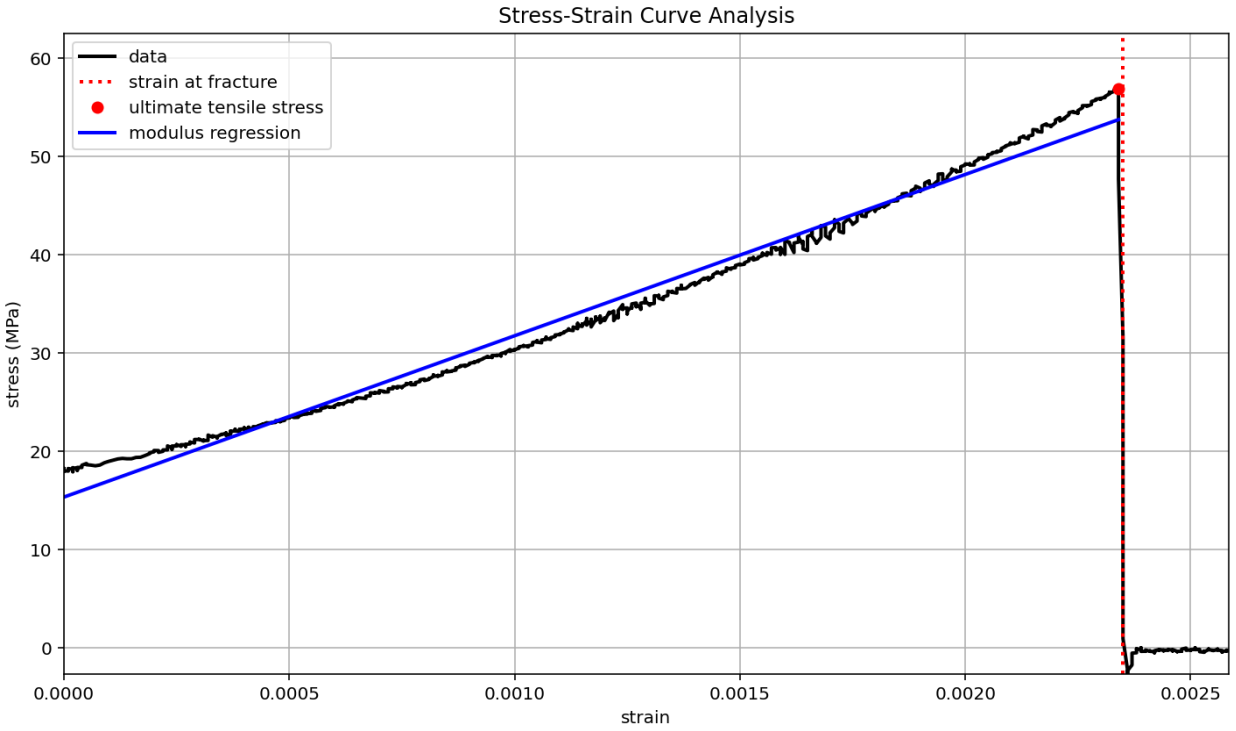
At the command line prompt, I executed the script

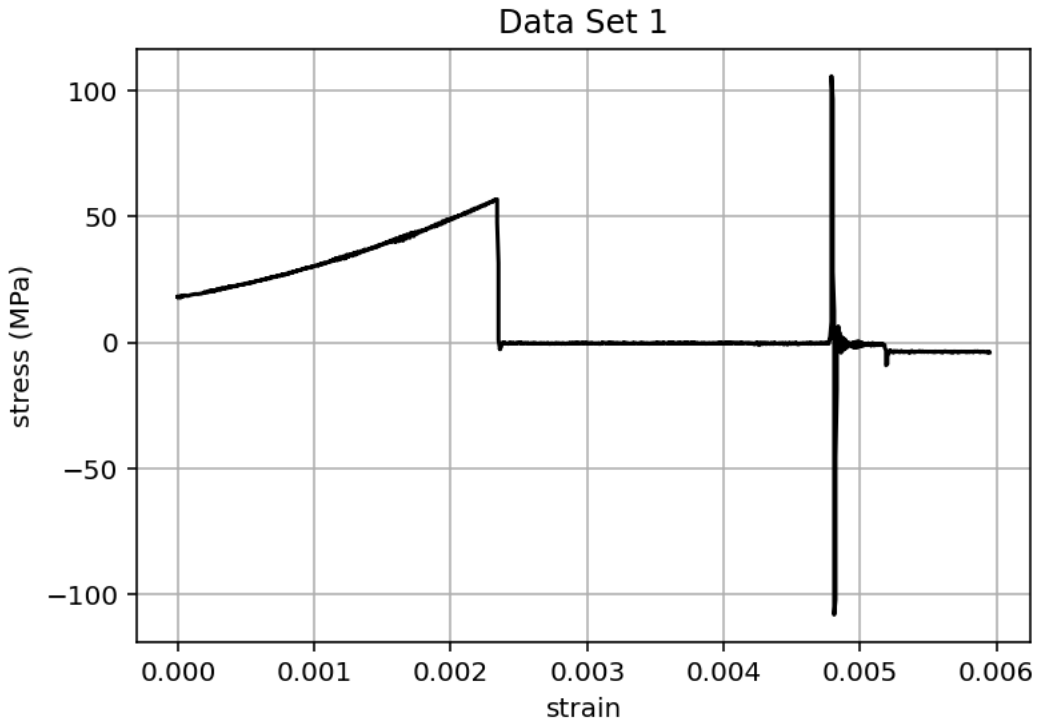
```
run cfiber_script_1_file.py
```

This generated the following output for the means, standard deviations and Measure of Fit.

```
ic = 1 dirname = data_set_1 nrep = 1
Strain at failure occurs at point 715 with strain 2.3500e-03 where the stress
changes from 4.7737e+01 to 3.1532e+01 MPa
Ultimate Tensile Stress occurs at point 712 with strain 2.3400e-03 where the
stress is 5.6859e+01 MPa
Elastic modulus is 1.6427e+04 +/- 7.5492e+01 with MOF = 0.9852.
case 1, rep 1, file data_set_1\Specimen_RawData_1.csv has strain from
0.000e+00 to 5.950e-03 & stress from 1.808e+01 to -3.839e+00.
```

The code also generated two plots.


Stress-Strain Curve Analysis
data
strain at fracture
ultimate tensile stress
modulus regression
stress (MPa)
strain
0
10
20
30
40
50
60
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025


Data Set 1
stress (MPa)
strain
100
50
0
−50
−100
0.000
0.001
0.002
0.003
0.004
0.005
0.006

Solution in Matlab:
To perform this task, I wrote a script, hw10_p5_driver.m, provided below.

*** start hw10_p5_driver.m ***

```
clear all;
close all;

%
%  open file with stress and strain data in two columns
%
cfiber_script_1fileonly;
% number of data points in stress and strain vectors
ndata = length(strain);

%
%  task 1. determine strain at failure
%
%  scan through data and look for first abrupt decrease in stress
%
% threshold = fractional relative change drop in stress that constitutes
% failure, a value of 0.9 means stress must drop 90%
% threshold is an adjustable parameter
threshold = 0.2;
% ifoundit is 1 if you have found the strain at failure, 0 if otherwise
ifoundit = 0;
% icount = data point counter
icount = 1;
while (ifoundit == 0 && icount < ndata)
    icount = icount + 1;
    deltastress = (stress(icount) - stress(icount-1))/stress(icount-1);
    if (deltastress < -threshold)
        ifoundit = 1;
    end
end
% save information about strain at fracture point
indexfrac = icount;
strainfrac = strain(icount);
stressfrac0 = stress(icount-1);
stressfrac1 = stress(icount);
fprintf(1, 'Strain at failure occurs at point %i with strain %e where the
stress changes from %e to %e MPa \n',indexfrac,strainfrac,stressfrac0,
stressfrac1);

%
%  task 2. determine ultimate tensile stress
%
% set range over which to search for UTS
indexlo = 1;
indexhi = indexfrac;
% use max function to find UTS and save information
[stressuts,indexuts] = max(stress(indexlo:indexhi));
strainuts = strain(indexuts);
fprintf(1, 'Ultimate Tensile Stress occurs at point %i with strain %e where
the stress is %e MPa \n',indexuts,strainuts,stressuts);
```

```
%
%  task 3.  determine modulus
%
% set range over which to perform regression
indexreglo = 1;
indexreghi = indexuts;
% perform linear regression
[b,bsd,MOF] =
linreg1(strain(indexreglo:indexreghi),stress(indexreglo:indexreghi));
% save information regarding regression
modulusav = b(2);
modulussd = bsd(2);
intercept = b(1);
fprintf(1, 'Elastic modulus is %e +/- %e with MOF =
%f.\n',modulusav,modulussd,MOF);

%
%  plot results
%
figure(2)
LineWidthValue = 2;
plot(strain,stress,'k-','LineWidth',LineWidthValue);
hold on;
ylabel('stress (MPa)');
xlabel('strain');
%
%  set reasonable axis limits
%
temp = axis;
[stressmin,indexmin] = min(stress(indexlo:indexhi+10));
ylo = min(0,stressmin);
yhi = stressuts*1.1;
axis([0 strainfrac*1.1 ylo yhi]);
%
% add line corresponding to strain at fracture
%
strainfracx(1) = strainfrac;
strainfracx(2) = strainfrac;
strainfracy(1) = ylo;
strainfracy(2) = yhi;
plot(strainfracx,strainfracy,'r:','LineWidth',LineWidthValue);
%
% add point corresponding to ultimate tensile stress
%
plot(strainuts,stressuts,'ro','LineWidth',LineWidthValue);
%
% add regression line
%
yhatx(1) = strain(indexreglo);
yhatx(2) = strain(indexreghi);
yhaty(1) = modulusav*yhatx(1)+intercept;
yhaty(2) = modulusav*yhatx(2)+intercept;
plot(yhatx,yhaty,'b-','LineWidth',LineWidthValue);
lgd = legend('data','strain at fracture','ultimate tensile stress','modulus
regression');
lgd.Location = 'northwest';
```

*** end hw10_p5_driver.m ***

This main script calls another script and a function. The script is cfiber_script_v01.m, which is included in the zip file where the data is located. The script is modified to read only the first file from the first data set.

The script also calls the function, linreg1.m, which performs linear regression of the experimental data to a first order polynomial model. This function is used without modification from its form on the course website.

The results of the script are provided below.

(a) strain at fracture

Strain at failure occurs at point 717 with strain 2.350000e-03 where the stress changes from 4.773658e+01 to 3.153238e+01 MPa

(b) ultimate tensile strength

Ultimate Tensile Stress occurs at point 714 with strain 2.340000e-03 where the stress is 5.685883e+01 MPa

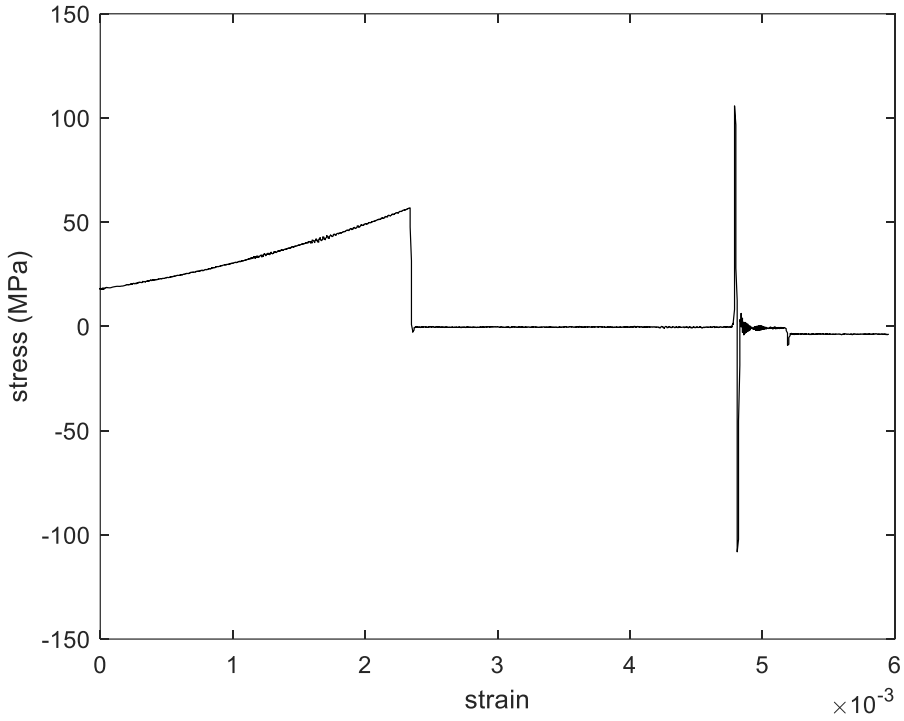
(c) modulus average and standard deviation

Elastic modulus is 1.641727e+04 +/- 7.551521e+01 with MOF = 0.985159.

(d) report the limits in terms of strain that you chose for your regression and justify.

I used the first point to the point corresponding to UTS. This seemed reasonable given that there was some curvature through-out the data and no clear perfectly linear region.

(e) provide a plot of the entire raw data



(f) provide a plot of the relevant data to (a), (b) and (c) and indicate strain at fracture, ultimate tensile strength and

