**ChE 310** **Problem Set 3 (15 pts)** **Due Wed 9/18/19**

Collect all m-files in a single .zip file and upload the .zip file to the course webpage by midnight on Wednesday, September 18, 2019. Please note any collaborations in the comments. Each student must upload their own individual copy of the work.

**3.1** One reason we care about Taylor series (or Maclaurin series when they are commonly centered at zero) is that they are linear approximations of non-linear functions. This allows us to use them for linear algebra and on matrices. How many terms are needed for a good approximation, and does this depend on the function?

For each of the following common series, plot the % error (compared to the TRUE value computed by using MATLAB's built-in functions) vs. number of terms on a ***single set of axes***. Use a log-log plot with n from 0 to 100 terms. Place a marker on the plot to note which function requires the most terms to minimize the error at x = 0.5.

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \ldots = \sum_{n=0}^{\infty} x^n$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \ldots = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

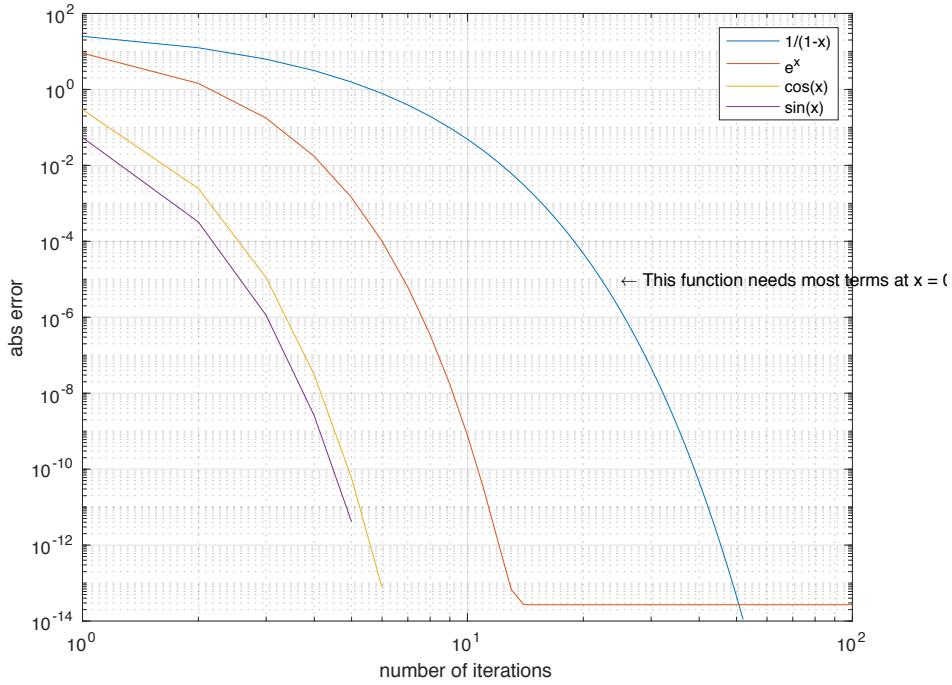
$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots = \sum_{n=1}^{\infty} (-1)^{(n-1)} \frac{x^{2n-1}}{(2n-1)!} \overset{\text{or}}{=} \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

**Solution:**

```
% PSET 3_1
% Coded by Nigel F. Reuel on 9.8.2016
%
% Preallocate memory for the error of the four functions (C1 = 1/(1-x), C2
% = exp(x), C3 = cos(x), and C4 = sin(x)
Error = zeros(101,4);
% Start all the sum values at zero
S1 = 0;
S2 = 0;
S3 = 0;
S4 = 0;
% Evaluate at x = 4;
x = 0.5;
% Evaluate the true value for the four functions
T1 = 1/(1-x);
T2 = exp(x);
T3 = cos(x);
T4 = sin(x);
for i = 0:100
    S1 = S1 + x^(i);
```

```
    S2 = S2 + x^(i)/factorial(i);
    S3 = S3 + (-1)^(i)*x^(2*i)/factorial(2*i);
    S4 = S4 + (-1)^(i)*x^(2*i+1)/factorial(2*i+1);
    % Compute the error for the four functions
    E1 = abs((S1-T1)/T1)*100;
    E2 = abs((S2-T2)/T2)*100;
    E3 = abs((S3-T3)/T3)*100;
    E4 = abs((S4-T4)/T4)*100;
    % Place the errors into our storage matrix
    Error(i+1,1) = E1;
    Error(i+1,2) = E2;
    Error(i+1,3) = E3;
    Error(i+1,4) = E4;
end
% Generate the loglog plot
n = (0:1:100);
loglog(n,Error)
legend('1/(1-x)','e^x','cos(x)','sin(x)')
xlabel('number of iterations');
ylabel('abs error');
grid on % Personal preference when using log-log plots
% Make the marker
txt = '\leftarrow This function needs most terms at x = 0.5';
text(25,10^-5,txt)
%
```

**Ouput:**



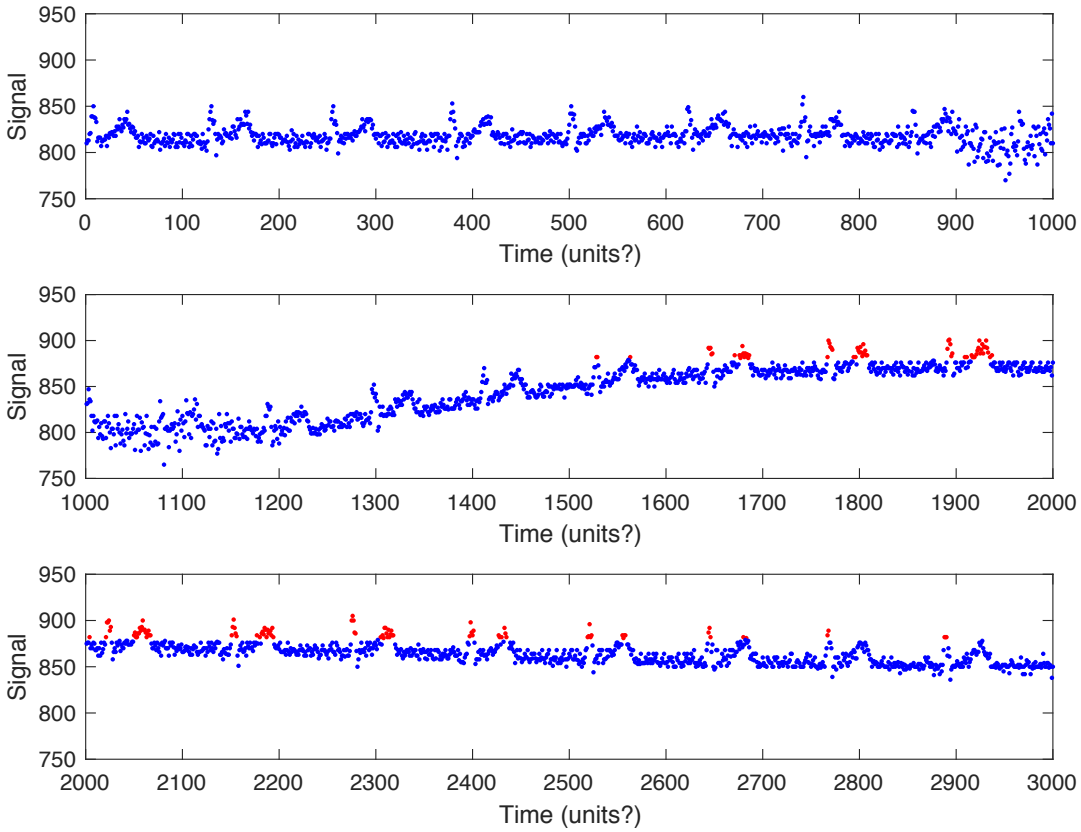

**3.2** An electrocardiogram (EKG or ECG) is a test that checks for problems with the electrical activity of your heart. The spikes and dips in the electrical signal are called waves. The 'data.csv' file contains some EKG readings taken every second. The clinician is worried about readouts greater than 880. From the provided data, create a scatter plot of the EKG reading vs.

time that marks all the worrisome points > 880 in red. To help the clinician, plot the data in three equal segments, vertically stacked (3x1 subplot).

**Solution:**

```
% PSET 3_3
% Coded by Nigel F. Reuel on 9.7.2016
%
% Read in the EKG vector
Data = csvread('data.csv');
% Divide the data into three equal sections:
N = length(Data)/3;
x = 1;
% Like all REAL numerical problems, there are multiple routes to the
% solution.  Here are a few:
% Slow Method - plot each point (take block comment off if you want to run)
%{
while x <= length(Data)
    % Define the subplot position based on the point location
    pos = ceil(x/N);
    subplot(3,1,pos)
    y = Data(x,1);
    if y > 880
        plot(x,y,'r.')
        ylim([750 950])
        hold on
    else
        plot(x,y,'b.')
        hold on
        ylim([750 950])
    end
    x = x+1;
end
xlabel('Time (units?)')
ylabel('Signal')
%}
% Fast Way - use logic on vectors
%Just need to know that there are 1000 points so, need to keep that in mind
Dhigh = Data.*(Data>880);
Dlow = Data.*(Data<880);
tvec = 1:length(Data);
subplot(3,1,1)
plot(tvec(1:1000),Dhigh(1:1000),'r.',tvec(1:1000),Dlow(1:1000),'b.')
ylim([750 950])
xlabel('Time (units?)')
ylabel('Signal')
subplot(3,1,2)
plot(tvec(1001:2000),Dhigh(1001:2000),'r.',tvec(1001:2000),Dlow(1001:2000),'b
.')
ylim([750 950])
xlabel('Time (units?)')
ylabel('Signal')
subplot(3,1,3)
plot(tvec(2001:end),Dhigh(2001:end),'r.',tvec(2001:end),Dlow(2001:end),'b.')
ylim([750 950])
xlabel('Time (units?)')
ylabel('Signal')
xlim([2000 3000])
```

**Output:**



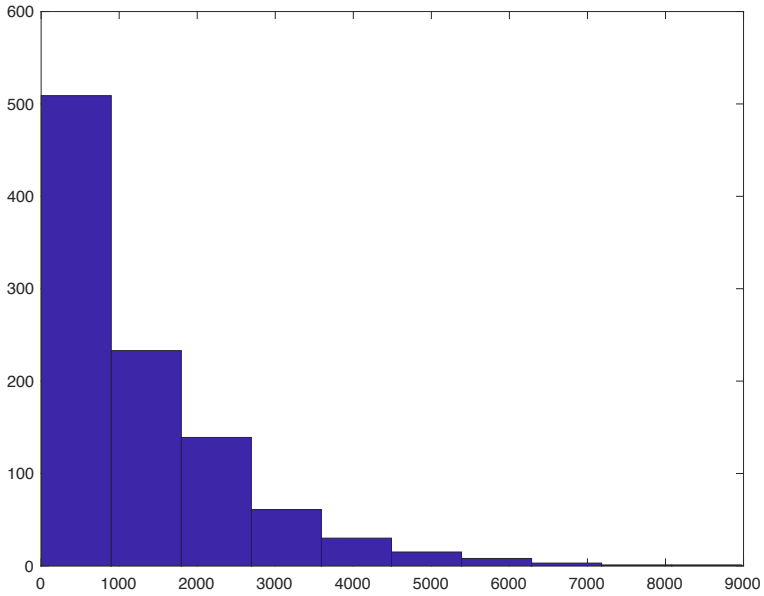

**3.3** The 'HorseKickData.xlsx' file records the number of deaths by horse kick in the Prussian army for 16 different cavalry units from 1875 to 1894. Create an M-file that does the following:

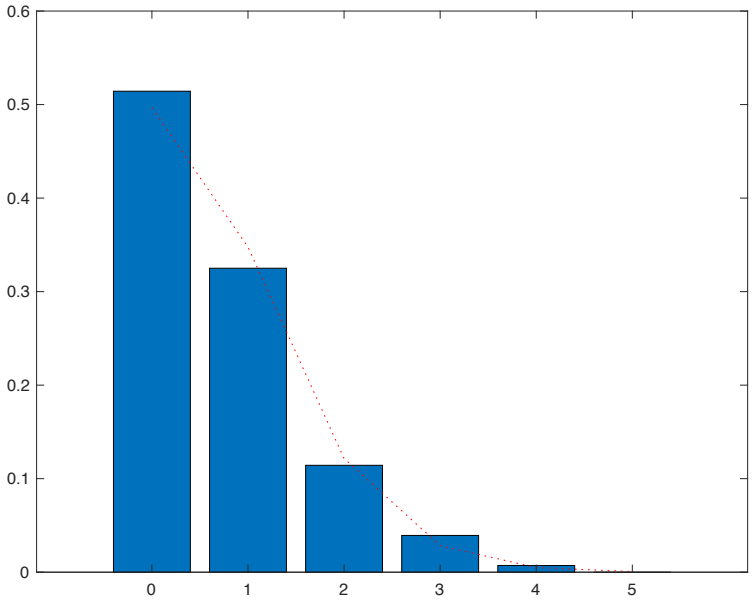
a. Create one histogram that includes all the measured counts in all units over all years.
b. Fit the data with a Poisson distribution.
c. Overlay the PDF on the normalized histogram, following the procedure from class.
d. In your m-file, answer: Why is the PDF only defined at integer values?
e. What is the probability of observing >4 deaths in a single unit in a given year? (Hint: use the Poisson CDF)
f. What is the average number of samples you would need to collect in order to see > 4 deaths, based on your answer in part e? Use a random number generator to simulate and validate this.

**Solution:**

```
% PSET 3_4
% Coded by Nigel F. Reuel on 9.7.17
% Solves the Poisson Horse Kick Problem
% Part a)
[num,txt,~] = xlsread('HorseKickData.xlsx');
Data = num(:,2:end);
DataVec = Data(:);
bins = [0 1 2 3 4 5];
counts = hist(DataVec, bins);
% Part b and c)
Param = poissfit(DataVec);
% Adjust the histogram to show probability density
nc = length(DataVec);
Prob = counts./nc;
figure
bar(bins,Prob)
x = [0,1,2,3,4,5];
y = poisspdf(x,Param);
hold on
plot(x,y,':r')
hold off
% Part d)
% Why is the PDF undefined for non-integer values?
     % Answer - Poisson is used for EVENT counting...A fractional event is
un-defined...
% What is the probability of a calvary having >4 deaths in one year from
% horse kicks?
% Part e and d)
ProbDeath = 1-poisscdf(4,Param)
% What is the average number of samples before >4 occurance occurs?
obsInterval = 1/ProbDeath
% Part f)
% Simulate this probability with random number generator
NS = 1000;
CountVec = zeros(NS,1);
for i = 1:NS
    flag = 0;
    count = 1;
    while flag == 0
        if poissrnd(Param)> 4
            flag = 1;
            CountVec(i,1) = count;
        end
        count = count + 1;
    end
end
figure
hist(CountVec)
```

**Output:**




>> PS3_3_real

ProbDeath =

7.8554e-04

obsInterval =

1.2730e+03

**3.4** Polynomial interpolation is not perfect, and higher-order polynomials may not always give us the most accurate results. Consider the following data, which were generated from the function $f(x) = \sin^2 x$:

| $x$ | 0.0000 | 0.7500 | 1.5000 | 2.2500 | 3.0000 |
|---|---|---|---|---|---|
| $f(x)$ | 0.0000 | 0.4646 | 0.9950 | 0.6054 | 0.0199 |

a. Plot these five data points, along with the original function $f$.
b. Use an interpolating polynomial of order 4 and the best possible interpolating polynomial(s) of order 3 to determine the values of $f(x)$ at the following points. Also calculate the error of these interpolations with respect to the TRUE value of $f$. (You are calculating a total of 4 values and 4 errors; print these to the command window.)
   i. $x = 1.7$
   ii. $x = 0.2$
c. Plot the polynomials over their valid ranges (i.e., avoid extrapolation!). Ensure your plot is properly labeled, including a legend.

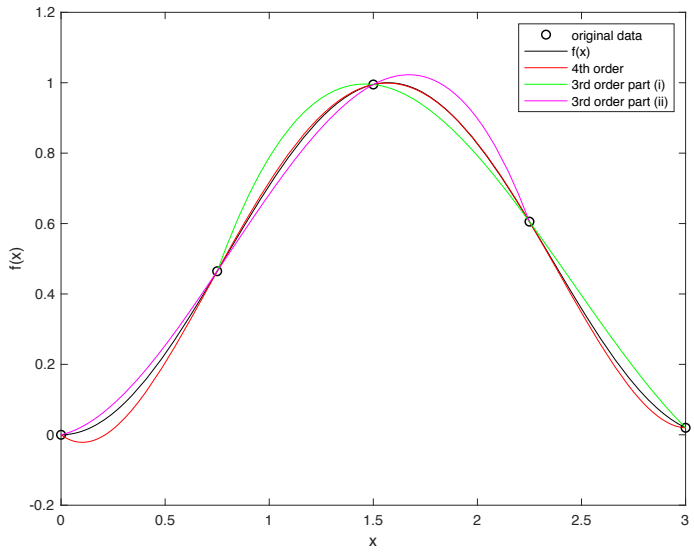
**Solution:**

```
%PS3_4.m
%Coded by LTR, 9/11/19
clear; clf

%Import data from table
x = 0:.75:3;
y = [0 .4646 .995 .6054 .0199];

%Actual function
f = @(x) sin(x).^2;

%Values we'll be evaluating at
v1 = 1.7; v2 = 0.3;

%Plot original data, and original function.
plot(x,y,'ko')
hold on
fplot(f,xlim,'k')

%Fit one fourth-order polynomial to full dataset (5 points):
p4 = polyfit(x,y,4);
fplot(@(x) polyval(p4,x),xlim,'r')

%Should use different 3rd order polynomials,
%depending on which point we are considering.
p3_1 = polyfit(x(2:5),y(2:5),3);
p3_2 = polyfit(x(1:4),y(1:4),3);

%Calculate true values
```

```
true1 = f(v1);
true2 = f(v2);

%Evaluate polynomials and print errors.
V1_O4 = polyval(p4,v1)
E1_O4 = (V1_O4-true1)/true1*100 ;
fprintf('Relative error is %2.1f percent for fourth order interpolating
polynomial at 1.7\n',E1_O4);
V1_O3 = polyval(p3_1,v1)
E1_O3 = (V1_O3-true1)/true1*100 ;
fprintf('Relative error is %2.1f percent for third order interpolating
polynomial at 1.7\n',E1_O3);
V2_O4 = polyval(p4,v2)
E2_O4 = (V2_O4-true2)/true2*100 ;
fprintf('Relative error is %2.1f percent for fourth order interpolating
polynomial at 0.2\n',E2_O4);
V2_O3 = polyval(p3_2,v2)
E2_O3 = (V2_O3-true2)/true2*100 ;
fprintf('Relative error is %2.1f percent for third order interpolating
polynomial at 0.2\n',E2_O3);

%Part c: plot
fplot(@(x) polyval(p3_1,x),[x(2) x(5)],'g')
fplot(@(x) polyval(p3_2,x),[x(1) x(4)],'m')
xlabel('x'); ylabel('f(x)')
legend('original data','f(x)','4th order','3rd order part (i)','3rd order
part (ii)')
```

**Output:**

```
>> PS3_4

V1_O4 =

   0.9818

Relative error is -0.2 percent for fourth order interpolating polynomial at 1.7

V1_O3 =

   0.9512

Relative error is -3.3 percent for third order interpolating polynomial at 1.7

V2_O4 =

   0.0437

Relative error is -49.9 percent for fourth order interpolating polynomial at 0.2

V2_O3 =

   0.1149
```




Relative error is 31.5 percent for third order interpolating polynomial at 0.2

**3.5** The internal energy (U – given in BTU/lb) of superheated steam varies depending on its temperature and pressure. Use the attached table from the Smith, Van Ness, Abbott thermodynamics text to create a matrix for interpolation (8 pressures vs. 7 temperatures – NOTE: you can do this by hand or use OCR tools). Show a surface plot to illustrate how the internal energy changes with temperature and pressure. Use bilinear interpolation to find the internal energies at the following conditions:

| **P** (psia) | 388 | 424 | 378 | 405 | 432 | 391 |
|---|---|---|---|---|---|---|
| **T** (°F) | 743 | 908 | 1172 | 791 | 617 | 863 |

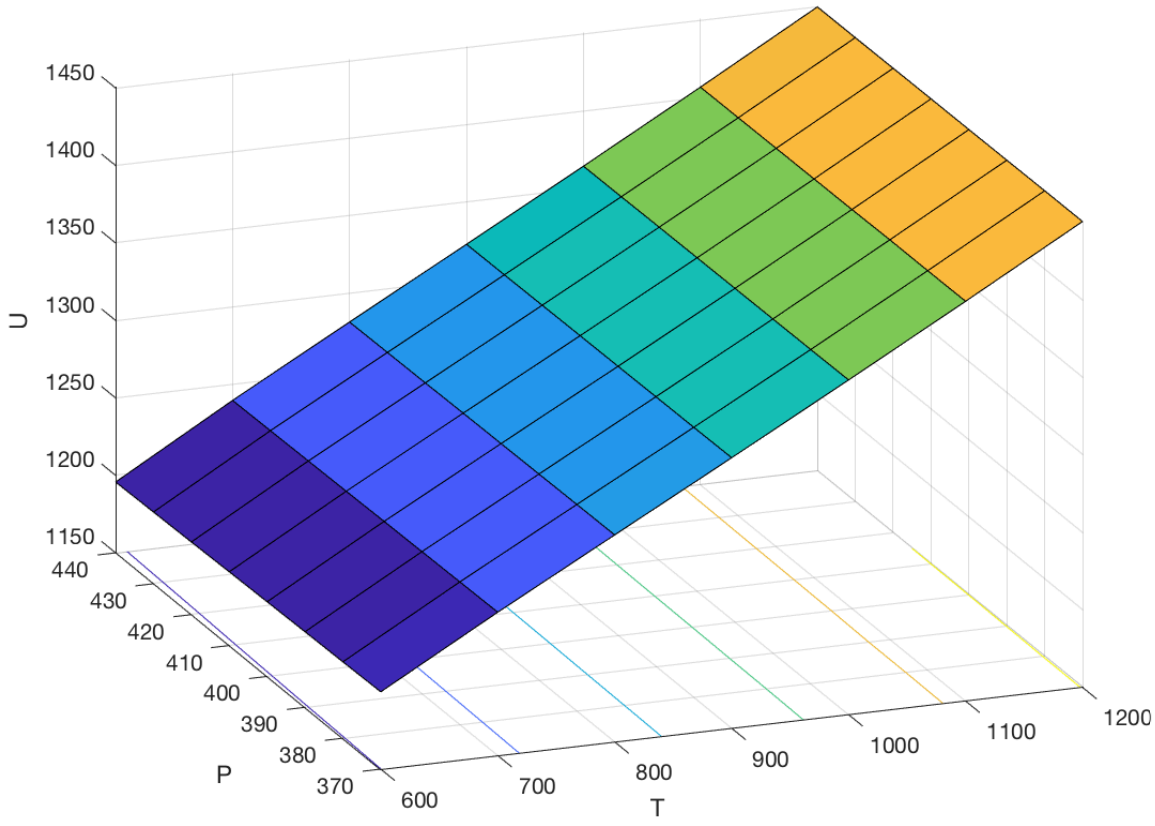
**Solution:**

```
% PSET3_2
% Coded by Nigel F. Reuel on 9.2.2016
% This solves problem 5 of the problem set
%
% I used adobe acrobat to turn the image into text and then cut and pasted
% the rows I needed into excel.  Here I import the data table:
Data = xlsread('Steam.xlsx');
T = (6:12).*100;
P = (37:44).*10;
% Create surface plot
surfc(T,P,Data);
xlabel('T')
ylabel('P')
zlabel('U')
% Fit the values given
pi = [388 424 378 405 432 391];
ti = [743 908 1172 791 617 863];
display('The internal energies for these given values are:')
U_out = interp2(T,P,Data,ti,pi)
```

**Excel File:**

| | | | | | | |
|---|---|---|---|---|---|---|
| 1199.9 | 1242.5 | 1283.6 | 1324.6 | 1366 | 1408 | 1450.7 |
| 1199.3 | 1242.1 | 1283.3 | 1324.3 | 1365.7 | 1407.8 | 1450.6 |
| 1198.8 | 1241.7 | 1283 | 1324.1 | 1365.5 | 1407.8 | 1450.4 |
| 1198.2 | 1241.3 | 1282.7 | 1323.8 | 1365.3 | 1407.4 | 1450.2 |
| 1197.6 | 1240.8 | 1282.4 | 1323.6 | 1365.1 | 1407.2 | 1450.1 |
| 1196.9 | 1240.4 | 1282 | 1323.3 | 1364.9 | 1407 | 1449.9 |
| 1196.3 | 1240 | 1281.7 | 1323 | 1364.6 | 1406.8 | 1449.7 |
| 1195.7 | 1239.6 | 1281.4 | 1322.8 | 1364.4 | 1406.6 | 1449.6 |

**Output:**




---

>> PS3_5

The internal energies for these given values are:

U_out =

1.0e+03 *

1.2595  1.3265  1.4386  1.2788  1.2036  1.3089