ASSIGNMENT: Curve Fit

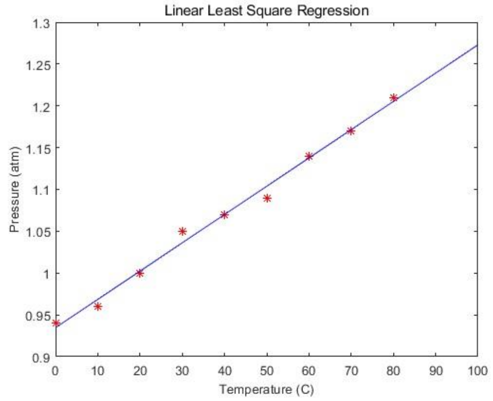
What you need to submit

- Submit the report+source files as a zip file online (LMS)
- **Report:** including pseudocode, output results, and source codes as instructed
- **Src Code:** (1) **Assignment_curvefit_Name_ID.cpp, (2) myNP.h, (3) myNP.cpp**
- All the functions you have created should be updated in myNP.h and myNP.cpp

Problem 1

An experiment is conducted that measures the pressure of a gas by heating it in a closed chamber.

Predict the pressure at T=100C



Part 1. Linear Least Square Regression

Create a function for Linear least square regression to predict the pressure at T=100C.

Matrix linearRegression(Matrix _x, Matrix _y);

or

double linearRegression(x, y)   // where   x, y are 1D array

Input: dataset (xi,yi),   // #m data sets

Output: coefficient z=[a0, a1]   // $f(x)=a_0+a_1x$

Procedure

- Write down a pseudocode for the function of **z=linearRegression(x,y)**
- Use MATLAB's function command " polyfit()" to solve for the answer and plot the results.
- Create your own C/C++ function.

  Find the predicted value at T=100C. Compare your answer with MATLAB results

< PROBLEM 1 >

PSEUDOCODE

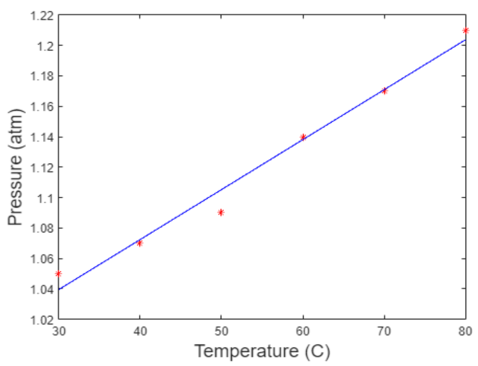
```
    m = length(x);

    if length(x) ~= length(y)
        exit
    end

Sx=0; Sxx = 0; Sy = 0; Sxy =0;

    for i=1:m
        Sx = Sx + x(i);
        Sxx = Sxx + x(i) * x(i);
        Sxy = Sxy + x(i) * y(i);
        Sy = Sy + y(i);
    end

    S = 1 / (m*Sxx -Sx*Sx);
    a = S * [m -Sx ; -Sx Sxx] * [Sxy; Sy];
    a1 = a(1)
    a0 = a(2)
return

end
```

MATLAB

$$\begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \frac{1}{nS_{xx} - S_x S_x} \begin{bmatrix} n & -S_x \\ -S_x & S_{xx} \end{bmatrix} \begin{bmatrix} S_{xy} \\ S_y \end{bmatrix}$$

```
[a1, a0]=linearRegression_student(T,P);

a1 = 0.0033
a0 = 0.9410
```

```
Fopt=a0+a1*T;

figure

plot(T,P, '*r')
hold on
plot(T,Fopt, '-b')
xlabel('Temperature (C)','fontsize',15)
ylabel('Pressure (atm)','fontsize',15)
```




```
function [a1,a0] = linearRegression_student(x, y)
% LinearRegration calculates the coefficients a1 and a0 of the linear
% equation y = a1*x + a0 that best fit n data points.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variable:
% a1   The coefficient a1.
% a0   The coefficient a0.


%% Your code goes here

% y = a1*x + a0


%   1. Get data sets of (x, y),  k=1 to m

%   2. Check m=length(X) and length(Y)
%   3. Is length(X)~= length(Y) ? Exit: Continue
    m = length(x);

    if length(x) ~= length(y)
        exit
    end

%   4. Initialize Sx, Sxx, Sy, Sxy
    Sx=0; Sxx = 0; Sy = 0; Sxy =0;
%   5. Solve for Sx, Sxx, Sy, Sxy,  for k=1 to m
    for i=1:m
        Sx = Sx + x(i);
        Sxx = Sxx + x(i) * x(i);
        Sxy = Sxy + x(i) * y(i);
        Sy = Sy + y(i);
    end

%   6. Solve for a1, a2
%     den = (n*Sxx-Sx*Sx);
%     a1 = (n*Sxy + (-Sx*Sxy))/den;
%     a0 = (-Sx*Sxy+Sxx*Sy)/den;

    S = 1 / (m*Sxx -Sx*Sx);
    a = S * [m -Sx ; -Sx Sxx] * [Sxy; Sy];
    a1 = a(1)
    a0 = a(2)
%   7. Return a1, a2
 return

end   % end of function
```

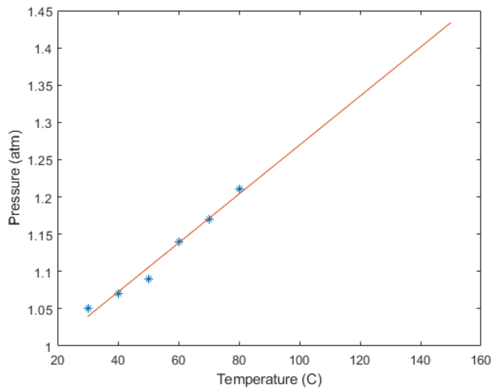
C function

```c
double linearRegression(double x[], double y[], int m, double z[]) {

    double sx = 0;
    double sxx = 0;
    double sxy = 0;
    double sy = 0;
    double S = 0;

    if (sizeof(x) != sizeof(y))
    {
        printf("ERROR: size of x and y is not same");
        system("pause");
        return 0;
    }
    for (int i = 0; i < m; i++) {
        sx += x[i];
        sxx += x[i] * x[i];
        sxy += x[i] * y[i];
        sy += y[i];
    }
    S = 1 / (m * sxx - sx * sx);
    z[0] = (sxx * sy - sxy * sx) * S;          // = a0
    z[1] = (m * sxy - sx * sy) * S;            // = a1

    return 0;
}
```

```c
int main(int argc, char* argv[]) {

    printf("==================PROBLEM 1==================₩n₩n");
    printf("        < Linear_Regression > ₩n₩n");

    double xi[] = { 30, 40, 50, 60, 70, 80 };
    double yi[] = { 1.05, 1.07, 1.09, 1.14, 1.17, 1.21 };
    double z[2] = { 0 };
    int m = sizeof(xi) / sizeof(xi[0]);

    linearRegression(xi, yi, m, z);

    printf("a1 = z[0] = %f ₩na0 = z[1] = %f ₩n₩nP = a0 * T + a1₩n₩n", z[0], z[1]);      // a1 = z[0], a0 = z[1]
    double T = 100;
    double Pr = 0;
    Pr = z[1] * T + z[0];                    // a0*T + a1
    printf("Pressure  = %f₩n₩n", Pr);
```

C and MATLAB RESULT

```
====================PROBLEM 1====================

          < Linear_Regression >

a1 = z[0] = 0.940952
a0 = z[1] = 0.003286

P = a0 * T + a1

Pressure  = 1.269524
```



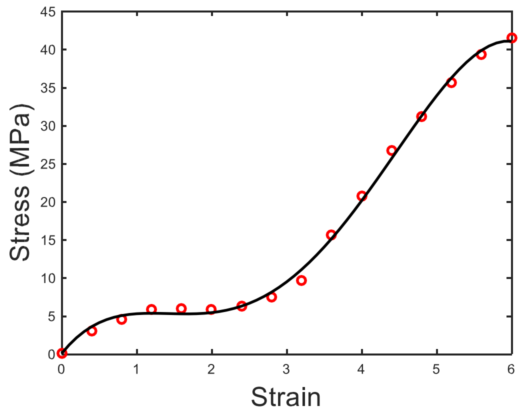

```
a = 1×2
    0.0033    0.9410

ans = 1.2695
```

Problem 2
Curve fit with a high order polynomial

Find the curve fit (least square method) for a m sets of measurements (x_i, y_i), i=0 to $m-1$, with n^{th} order polynomial of

$$f(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$$ Assume that $n<m$.

Fit with 3^{rd} and 4^{th} order polynomial using the following measurement data $y = a_3x^3 + a_2x^2 + a_1x + a_0$ % third order polynomial $y = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ % fourth order polynomial



strain = 0:0.4:6;
stress = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];

Procedure

a) First, write down a pseudocode for the function of polyfit()
b) Use MATLAB's function command " polyfit()" to solve for the answer and plot the results.
c) Then, create your own C/C++ function of polyfit(). Find the coefficients for 3^{rd} and 4th order polynomial. Compare your answer(coefficients) with MATLAB results

Example function:

void polyfit(**x**, **y**, **z,** n)

or you can use Matrix instead of 1D array

- *x, y*: 1-D array double, data points (*m* sets of points)
  * check whether the size of **x** is equal to size of **y**
- *n*: Integer scalar, order of polynomial. $1 \leq n < m$

- *z*: 1-D array double, coefficient a_i , i=n to *0*

Challenge: (Option, bonus point)

- Make **polyfit( )** efficient in terms of reducing iteration numbers. You can submit in MATLAB code.

: The reference MATLAB code in your textbook is NOT efficient. See Appendix

#iteration= (3*n+1)*m+(n+2)*n (e.g. iter= 232 for m=16, n=4)

: Reduce the iterations for given data sets *m* and polynomial order *n* .

: For this example, use (m=16, n=4)

Basic: #iter < (3n+1)m+(n+2)n (e.g, iter=232)

Good: #iter <= m(n+1)+n(n+2) (e.g. iter=104)

Excellent: #iter <= nm+(n-1)(n-2)/2 (e.g. iter=67)

< PROBLEM 2 >

PSEUDOCODE

```
m = length(x);
my = length(y);
Zopt=zeros(1,n+1);

if m ~= my
    disp('ERROR: The number of elements in x must be the same as in y.')
end
SX=zeros(n+1); Sxy=zeros(n+1,1); Zopt=zeros(n+1,1);
S=zeros(n+1,n+1);  b=zeros(n+1,1);
for i=0:2*n
    SXtemp=0;
    for k=1:m
        SXtemp=SXtemp+x(k)^i;
    end
    SX(i+1)=SXtemp;
end
for i = 1:n+1
    for j=1:n+1
        S(i,j)= SX((2*n)-(i-1)-(j-1)+1);
    end
end
for j=n:-1:0
    Sxytemp=0;
    for k=1:m
        Sxytemp=Sxytemp+(y(k)*((x(k))^(j)));
    end
    Sxy(j+1,1)=Sxytemp;
end
for i = 1:n+1
  b(i,1)=Sxy((n+1)-i+1);
end
Zopt=(S\b);
```

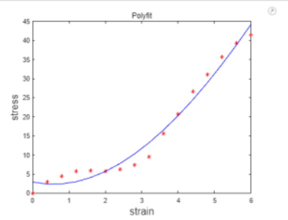
MATLAB

```
function [a1,a0] = linearRegression_student(x, y)
    m = length(x);

    if length(x) ~= length(y)
        exit
    end

%   4. Initialize Sx, Sxx, Sy, Sxy
    Sx=0; Sxx = 0; Sy = 0; Sxy =0;
%   5. Solve for Sx, Sxx, Sy, Sxy,  for k=1 to m
    for i=1:m
        Sx = Sx + x(i);
        Sxx = Sxx + x(i) * x(i);
        Sxy = Sxy + x(i) * y(i);
        Sy = Sy + y(i);
    end

%   6. Solve for a1, a2
%     den = (n*Sxx-Sx*Sx);
%     a1 = (n*Sxy + (-Sx*Sxy))/den;
%     a0 = (-Sx*Sxy+Sxx*Sy)/den;

    S = 1 / (m*Sxx -Sx*Sx);
    a = S * [m -Sx ; -Sx Sxx] * [Sxy; Sy];
    a1 = a(1)
    a0 = a(2)
%   7. Return a1, a2
 return

end   % end of function
```

```
Xdata = 0:0.4:6;
Ydata = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
n=3;

% Matlab function
ZoptMATLAB=polyfit(Xdata,Ydata,n)
```

```
ZoptMATLAB = 1x4
    -0.0541    1.8030   -1.9882    2.893
```

```
% Zopt =   -0.2644    3.1185  -10.1927   12.8780   -0.2746

% NP  function
Zopt=polyFit_student(Xdata,Ydata,n);
Yopt=polyval(Zopt,Xdata);  % Matlab function

figure
plot(Xdata,Ydata, '*r')
hold on
plot(Xdata,Yopt, '-b')
xlabel('strain','fontsize',15)
ylabel('stress','fontsize',15)
title('Polyfit')
```

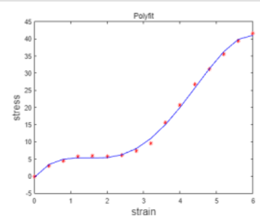


3rd polynomial

```
Xdata = 0:0.4:6;
Ydata = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
n=4;

% Matlab function
ZoptMATLAB=polyfit(Xdata,Ydata,n)
```

```
ZoptMATLAB = 1x5
    -0.2644    3.1185  -10.1927   12.8780   -0.274
```

```
% Zopt =   -0.2644    3.1185  -10.1927   12.8780   -0.2746

% NP  function
Zopt=polyFit_student(Xdata,Ydata,n);
Yopt=polyval(Zopt,Xdata);  % Matlab function

figure
plot(Xdata,Ydata, '*r')
hold on
plot(Xdata,Yopt, '-b')
xlabel('strain','fontsize',15)
ylabel('stress','fontsize',15)
title('Polyfit')
```



4th polynomial

C and MATLAB RESULT

```
==================PROBLEM 2==================
                 < Polyfit >
iteration = 49
polynomial : [ 3 ]
z =
      -0.054117
       1.803028
      -1.988223
       2.892982
```

```
Xdata = 0:0.4:6;
Ydata = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
n=3;

% Matlab function
ZoptMATLAB=polyfit(Xdata,Ydata,n)
```

```
ZoptMATLAB = 1×4
    -0.0541    1.8030   -1.9882    2.893
```

```
==================PROBLEM 2==================
                 < Polyfit >
iteration = 67
polynomial : [ 4 ]
z =
      -0.264389
       3.118549
     -10.192668
      12.877980
      -0.274607
```

```
Xdata = 0:0.4:6;
Ydata = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
n=4;

% Matlab function
ZoptMATLAB=polyfit(Xdata,Ydata,n)
```

```
ZoptMATLAB = 1×5
    -0.2644    3.1185  -10.1927   12.8780   -0.274
```