**ASSIGNMENT: Curve Fit**

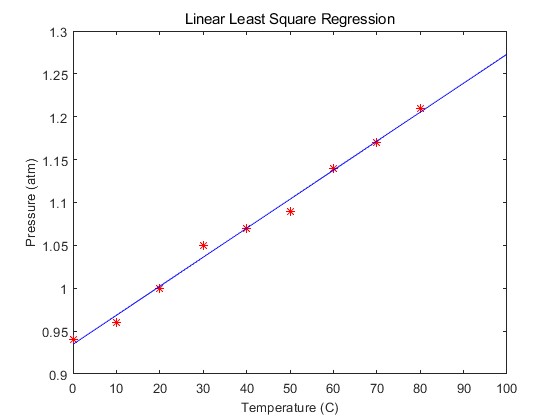
**What you need to submit**

* 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)=a0+a1x

## 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];

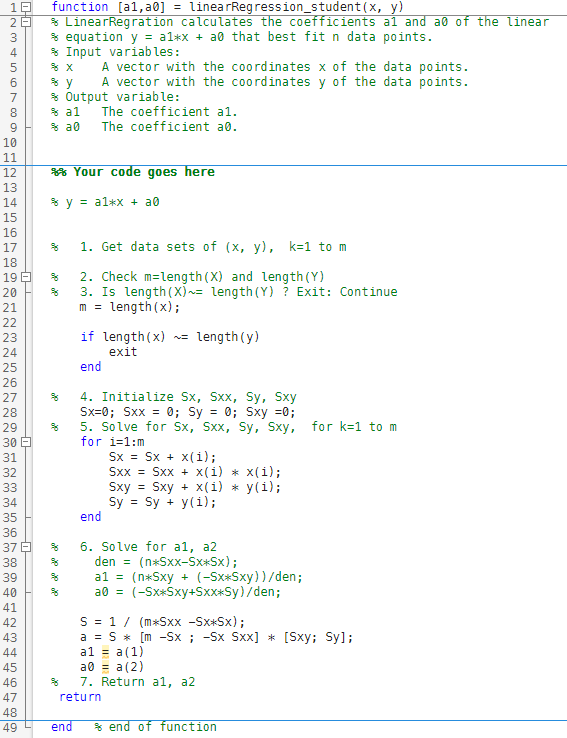
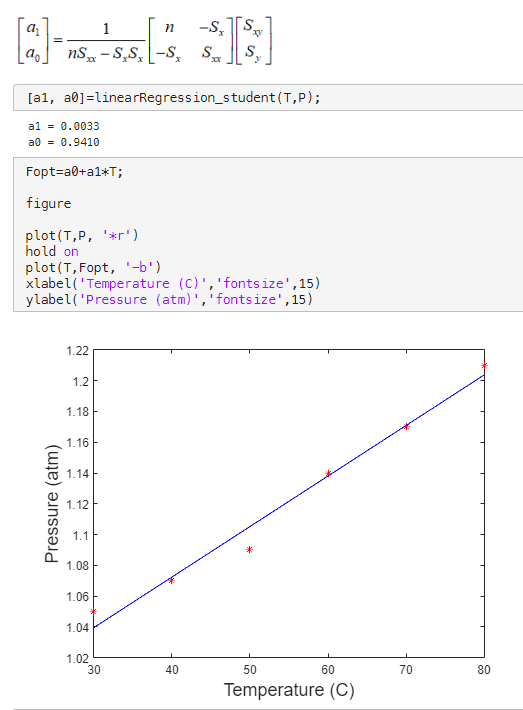
a1 = a(1)

a0 = a(2)

return

end

**MATLAB**



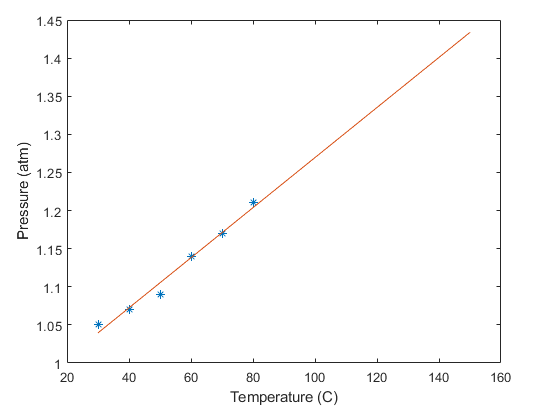
**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명C function**

**C and MATLAB RESULT**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**

# Problem 2

## Curve fit with a high order polynomial

Find the curve fit (least square method) for a *m* sets of measurements

(*xi,yi*), *i*=0 to *m-1,* with nth order polynomial of

𝑓(𝑥)=𝑎0+𝑎1𝑥 +𝑎2𝑥2+⋯+𝑎𝑛𝑥𝑛 Assume that *n<m.*

Fit with 3rd and 4th order polynomial using the following measurement data *y* = *a*3*x*3 +*a*2*x*2 + *a*1*x* + *a*0  % third order polynomial *y* = *a*4 *x*4+*a*3*x*3 +*a*2*x*2 + *a*1*x* + *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

1. First, write down a pseudocode for the function of polyfit()
2. Use MATLAB’s function command “ polyfit()” to solve for the answer and plot the results.
3. Then, create your own C/C++ function of polyfit(). Find the coefficients for 3rd 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≤𝑛 <𝑚 |
| - *z*: | 1-D array double, coefficient *ai* , 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

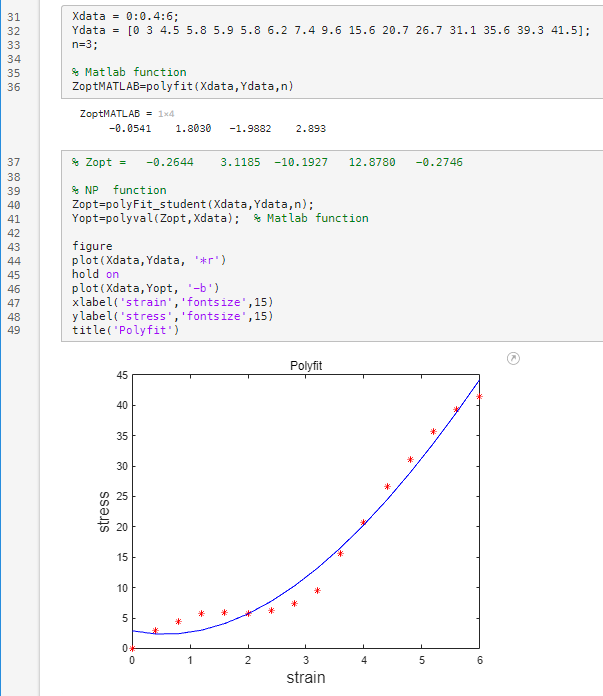
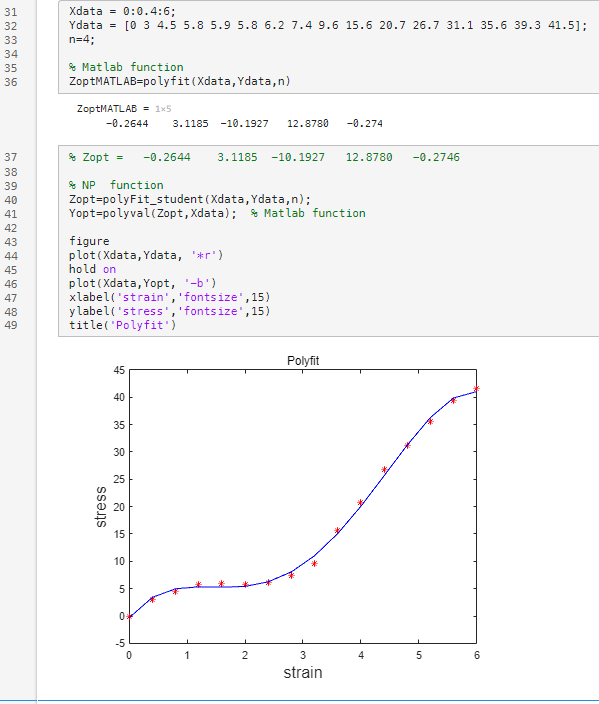
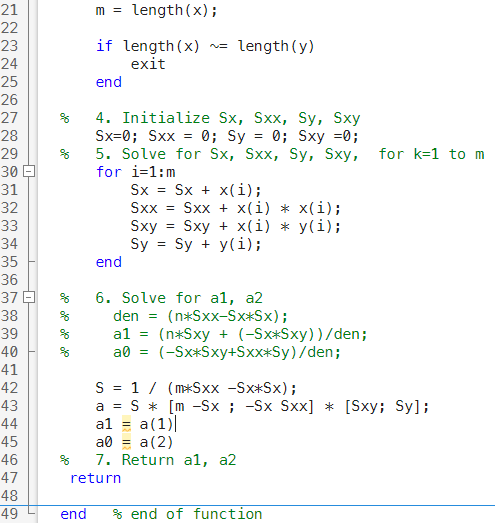
for i = 1:n+1

b(i,1)=Sxy((n+1)-i+1);

end

Zopt=(S\b);

**MATLAB**



4th polynomial

3rd polynomial

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**C and MATLAB RESULT**

텍스트이(가) 표시된 사진

자동 생성된 설명