LINEAR AND POLYNOMIAL REGRESSION

The Mathematics behind polyfit()

Introduction:

I've identified the following types of regression:

1. Simple and Multiple Linear Regression
2. Simple and Multiple Polynomial Regression

Simple Linear Regression and Simple Polynomial Regression deal with a dependence of the form *y = function (x)*, whilst Multiple Linear Regression and Multiple Polynomial Regression deal with a dependence of the form *y = function (x1, x2, x3, ...)*.

Simple Linear Regression

Linear Regression is fitting a "best" straight line thr. the points.

The mathematical expression for the straight line is:

$$y = a_0 + a_1 \cdot x + e$$

where: $a_0 -$ y intercept
$a_1 -$ slope
$e -$ error between the model and the observations

Derivation of coeff's of best-fit polynomial:

We need to minimize the error. Minimizing $\sum_{i=1}^{n} e_i$ does not work, so we use "least square regression" and try to minimize $\sum_{i=1}^{n} e_i^2$:

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} [y_i - a_0 - a_1 x_i]^2$$

therefore:

$$min(S_r) = min\left(\sum_{i=1}^{n} e_i^2\right) = min\left(\sum_{i=1}^{n} [y_i - a_0 - a_1 x_i]^2\right)$$

Find the value of coeff's a_0 and a_1:

$$
\begin{aligned}
= \quad & \frac{\partial S_r}{\partial a_0} = \frac{\partial}{\partial a_0} \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} e_i^2 \\
= \quad & \sum_{i=1}^{n} \left[\frac{\partial (y_i - a_0 - a_1 x_i)^2}{\partial a_0} \right] \\
= \quad & -2 \cdot \sum_{i=1}^{n} [y_i - a_0 - a_1 x_i]
\end{aligned}
$$

$$
\frac{\partial S_r}{\partial a_0} = 0 \Leftrightarrow \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a_0 - \sum_{i=1}^{n} a_1 x_i = 0
$$

$$
\Leftrightarrow n \cdot a_0 + a_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i
$$

From the minimization condition we obtain:

$$
\frac{\partial S_r}{\partial a_0} = 0 \Rightarrow a_0 = \frac{1}{n} \sum_{i=1}^{n} y_i - a_1 \cdot \frac{1}{n} \sum_{i=1}^{n} x_i = \overline{y} - a_1 \cdot \overline{x}
$$

$$
\frac{\partial S_r}{\partial a_1} = 0 \Rightarrow a_1 = \frac{n \sum x_i y_i - \frac{1}{n} \sum x_i \sum y_i}{n \sum x_i^2 - \left(\sum x_i \right)^2}
$$

Such that we find the regression polynome: $y = a_0 + a_1 x$

MATLAB code:

This code is equiv. to the function polyfit(x,y,1).

```
x = 1:1:7;
y = [0.5 2.5 2.0 4.0 3.5 6.0 5.5];
n = size(x);
disp(n(2));
```

```
    7
```

```
sumXY = 0,sumXi=0,sumXi2=0,sumYi=0;
```

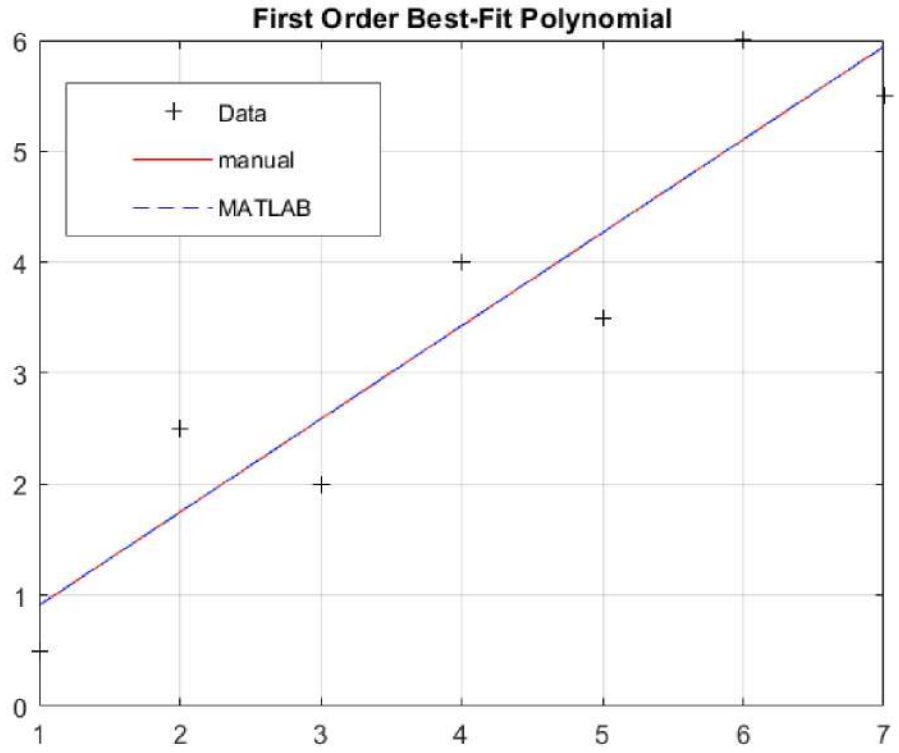
```
sumXY = 0
sumXi = 0
sumXi2 = 0
```

```
for i = 1:n(2)
    sumXY = sumXY + x(i) * y(i);
    sumXi = sumXi + x(i);
    sumXi2 = sumXi2 + x(i)^2;
    sumYi = sumYi + y(i);
end
```

```
medX = sumXi /n(2);

medY = sumYi /n(2);

A1 = (n(2) * sumXY - sumXi * sumYi)/(n(2) * sumXi2 - sumXi^2);
A0 = medY - A1 * medX;

bFit = A0 + A1 * x;

figure
plot(x,y,'k+'),grid,hold
```

```
Current plot held
```

```
plot(x,bFit,'r-')
plot(x,table(polyfit(x,y,1)).Var1(2) + table(polyfit(x,y,1)).Var1(1) .* x,'b--')
legend("Data","manual","MATLAB")
title("First Order Best-Fit Polynomial")

legend(["Data","manual","MATLAB"])
legend("Position",[0.15454,0.6862,0.28916,0.18891])
```



Simple Polynomial Regression

Polynomial Regression is fitting a "best" curve thr. points.

The mathematical expression for the curve is:

$$y = a_0 + a_1 x + a_2 x^2 + e$$

where the meaning of the notation is retained from simple linear regression.

We follow the logic outlined in the first part of this doc., we must minimize the squared error and obtain the coeff's of the best-fit polynomial.

1. Determine $S_r = \sum_{i=1}^{n} e_i^2$
2. Minimize S_r w.r.t a_0, a_1 and a_2 i.e. $\frac{\partial S_r}{\partial a_0} = 0$, $\frac{\partial S_r}{\partial a_1} = 0$, $\frac{\partial S_r}{\partial a_2} = 0$
3. Obtain system with three eq's and three unknowns, from which we derive the values of the coeff's:

$$\begin{bmatrix} n & \Sigma x_i & \Sigma x_i^2 \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i^3 \\ \Sigma x_i^2 & \Sigma x_i^3 & \Sigma x_i^4 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \Sigma y_i \\ \Sigma x_i y_i \\ \Sigma x_i^2 y_i \end{bmatrix}$$

[from Curve Fitting/UTM]

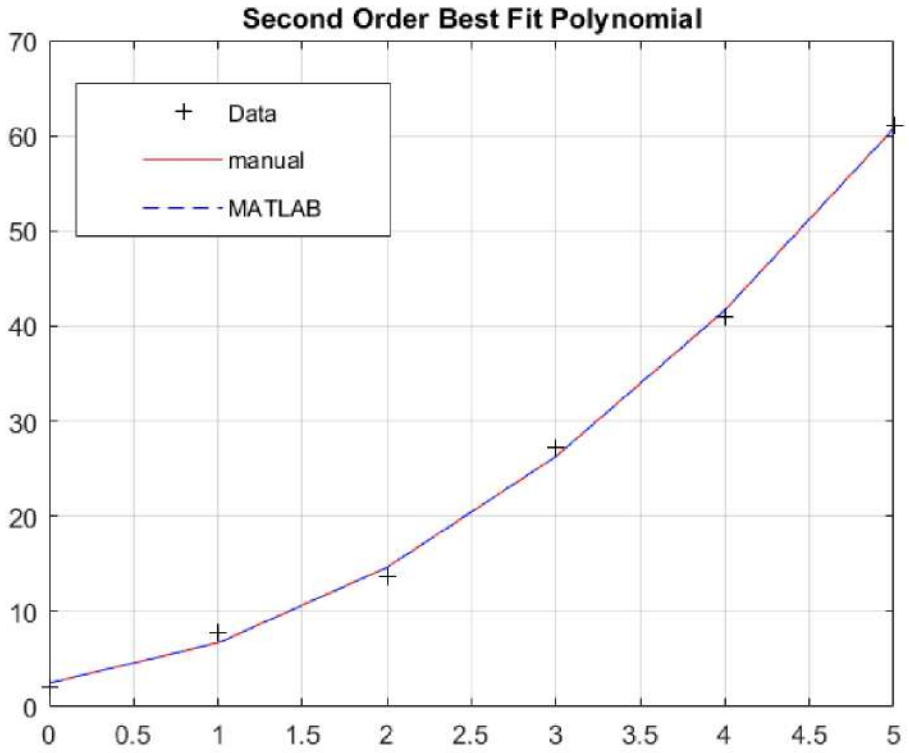
From this point f'ward we need only to use the Cramer Rule and determine the solution of this non-homgenous, compatible and determined system of eq's.

MATLAB code:

This code is equiv. to the function polyfit(x,y,2).

```
promptX = "Introduce values for X - axis:";
promptY = "Introduce values for Y - axis:";

x=0:1:5;
y=[2.1,7.7,13.6,27.2,40.9,61.1];
QLRP(x,y);
```

```
Current plot held
```




```
function status = QLRP(x,y)
status = "success";
A = getA(x);
B = getB(x,y);

a0 = DETA0(x,y) / det(A);
a1 = DETA1(x,y) / det(A);
a2 = DETA2(x,y) / det(A);

figure
plot(x,y,'k+'),grid,hold
plot(x,a0 + a1 .* x + a2 .* x .* x, 'r-'),grid
hold on;
plot(x,table(polyfit(x,y,2)).Var1(3) + table(polyfit(x,y,2)).Var1(2) .* x + table(polyfit(x,y,2
title("Second Order Best Fit Polynomial")
legend("Data","manual","MATLAB")
legend("Position",[0.15454,0.6862,0.28916,0.18891])

function A = getA(x)
n = size(x);
sX1 = sumX1(x);
sX2 = sumX2(x);
sX3 = sumX3(x);
sX4 = sumX4(x);
A = [n(2),sX1,sX2;...
```

```
    sX1,sX2,sX3;...
    sX2,sX3,sX4];
end

function B = getB(x,y)
    n = size(x);
    sY = sumY(y);
    sXY = sumXY(x,y);
    sX2Y = sumX2Y(x,y);
    B = [sY,sXY,sX2Y];
end


%%PARTIAL SUMS%%
function sX1 = sumX1(x)
    sX1 = 0;
    n = size(x);
    for i = 1:n(2)
        sX1 = sX1 + x(i);
    end
end

function sX2 = sumX2(x)
    sX2 = 0;
    n = size(x);
    for i = 1:n(2)
        sX2 = sX2 + x(i) ^ 2;
    end
end

function sX3 = sumX3(x)
    sX3 = 0;
    n = size(x);
    for i = 1:n(2)
        sX3 = sX3 + x(i) ^ 3;
    end
end

function sX4 = sumX4(x)
    sX4 = 0;
    n = size(x);
    for i = 1:n(2)
        sX4 = sX4 + x(i) ^ 4;
    end
end

function sY = sumY(y)
    sY = 0;
    n = size(y);
    for i = 1:n(2)
```

```
        sY = sY + y(i);
    end
end

function sXY = sumXY(x,y)
    sXY = 0;
    n = size(x);
    for i = 1:n(2)
        sXY = sXY + x(i) * y(i);
    end
end

function sX2Y = sumX2Y(x,y)
    sX2Y = 0;
    n = size(x);
    for i = 1:n(2)
        sX2Y = sX2Y + x(i) ^ 2 * y(i);
    end
end

%%CRAMER SUBMATRICI%%

function detA0 = DETA0(x,y)
k = 1;
A0 = getA(x);
B = getB(x,y);
n = size(A0);
for i = 1:n(1)
    for j = 1:n(2)
        if j == 1
            A0(i,j) = B(k);
            k = k + 1;
        end
    end
end
detA0 = det(A0);
end


function detA1 = DETA1(x,y)
k = 1;
A1 = getA(x);
B = getB(x,y);
n = size(A1);
for i = 1:n(1)
    for j = 1:n(2)
        if j == 2
            A1(i,j) = B(k);
            k = k + 1;
        end
```

```
    end
end
detA1 = det(A1);
end

function detA2 = DETA2(x,y)
k = 1;
A2 = getA(x);
B = getB(x,y);
n = size(A2);
for i = 1:n(1)
    for j = 1:n(2)
        if j == 3
            A2(i,j) = B(k);
            k = k + 1;
        end
    end
end
detA2 = det(A2);
end
end
```

FUTURE EXPANSION:

3rd order polynomial regression with dataset

derivation of mth order polynomial regression

multiple linear derivation with dataset

multiple polynomial derivation with datase

MISC:

2nd order polynomial regression dataset

%Introduce values for X - axis:0:1:5

%Introduce values for Y - axis:[2.1,7.7,13.6,27.2,40.9,61.1]

3rd order polynomial regression dataset

Introduce values for X - axis:1:1:15

Introduce values for Y - axis:[3, 14, 23, 25, 23, 15, 9, 5, 9, 13, 17, 24, 32, 36, 46]