**ASSIGNMENT 2**

**OPTIMAL FILTER AND ESTIMATION**

**QUESTION 1:**

The program uses polyfit to do the curve fitting and ployval to get the predicted value.

**Syntax** : polyfit(x value, y value , degree of the polynomial used for curve fitting).

polyval(p,x) – gives the value of polynomial p at value x.

**PROGRAM**

% Consider year 1946 to be the initial value for starting

x = [];

x\_graph = [];

for i=0:10

x(i+1)=i;

x\_graph(i+1)=1946+i;

end

y = [66.6 84.9 88.6 78.0 96.8 105.2 93.2 111.6 88.3 117.0 115.2];

% Fitting the polynomial

p\_1 = polyfit(x,y,1);

p\_2 = polyfit(x,y,2);

p\_3 = polyfit(x,y,3);

p\_4 = polyfit(x,y,4);

% To find the value of the polynomial fitting function

y\_1 = polyval(p\_1,x);

y\_2 = polyval(p\_2,x);

y\_3 = polyval(p\_3,x);

y\_4 = polyval(p\_4,x);

% plotting the graphs

subplot(2,2,1);

plot(x\_graph,y,'o');

hold on;

xlabel('Year'); ylabel('Production of steel');

plot(x\_graph,y\_1); % plotting linear

subplot(2,2,2);

plot(x\_graph,y,'o');

hold on;

xlabel('Year'); ylabel('Production of steel');

plot(x\_graph,y\_2); % plotting quadratic

subplot(2,2,3);

plot(x\_graph,y,'o');

hold on;

xlabel('Year'); ylabel('Production of steel');

plot(x\_graph,y\_3); % plotting cubic

subplot(2,2,4);

plot(x\_graph,y,'o');

hold on;

xlabel('Year'); ylabel('Production of steel');

plot(x\_graph,y\_4); % plotting quartic

hold off;

% Finding the rms error

rms(1) = sqrt(mean((y\_1 - y).^2));

rms(2) = sqrt(mean((y\_2 - y).^2));

rms(3) = sqrt(mean((y\_3 - y).^2));

rms(4) = sqrt(mean((y\_4 - y).^2));

y\_pre =[];

% Predicting the value for 1957

y\_pre(1) = polyval(p\_1,11);

y\_pre(2) = polyval(p\_2,11);

y\_pre(3) = polyval(p\_3,11);

y\_pre(4) = polyval(p\_4,11);

y\_pre

rms

**OUTPUT:**

**(Predicted value for 1957)**

**Order : Linear , Quadratic Cubic and Quartic**

**y**\_pre =

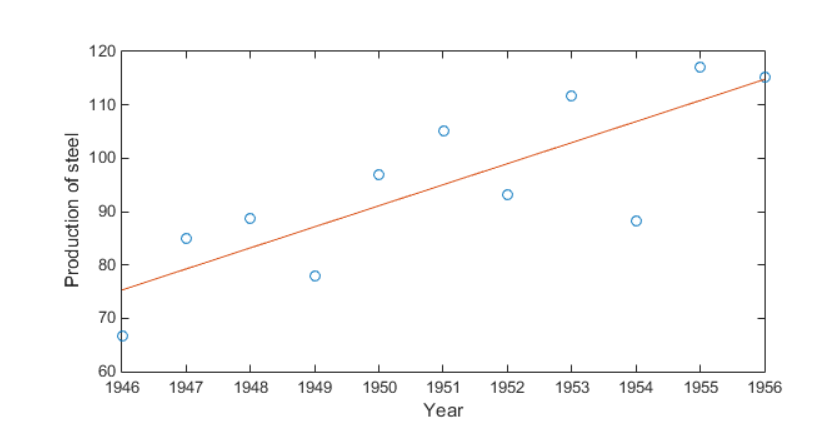
118.7145 114.5297 126.1879 128.8606

rms =

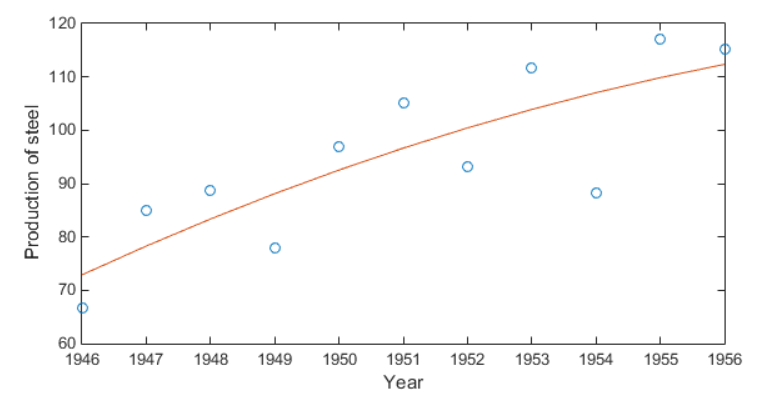
8.7823 8.6665 8.2889 8.2816

**GRAPH:**

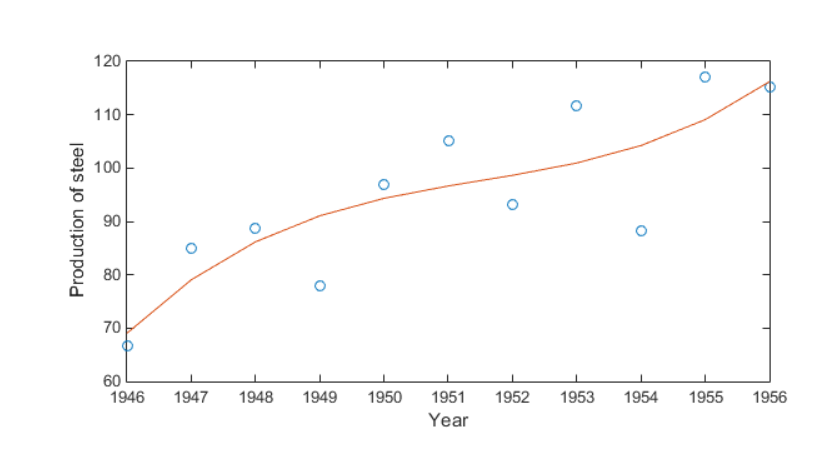
**LINEAR CLASSIFICATION**



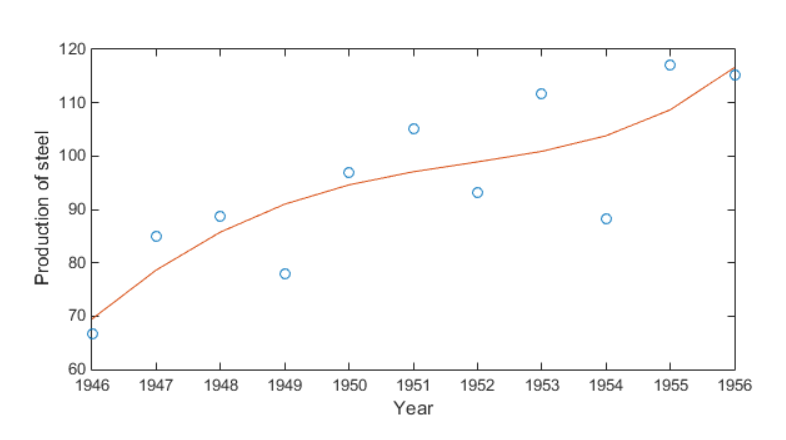
**QUADRATIC CLASSIFICATION**

****

**CUBIC CLASSIFICATION**

****

**QUADRIC CLASSIFICATION**

****

**QUESTION 2:**

**CODE FOR CHECKING MATRIX CALCULATION:**

x\_0 = [1;2;3];

p\_0 = [4 0 1; 0 4 1; 1 1 4];

H = [1 3 2];

R\_1 = 6;

y = 6;

K = p\_0\*H'\*inv(H\*p\_0\*H'+R\_1)

x\_1 = x\_0+K\*(y-H\*x\_0)

p\_1 = (eye(3)-K\*H)\*p\_0\*(eye(3)-K\*H)'+(K\*R\_1\*K')

**QUESTION 3:**

**PROGRAM:**

**(b) When the output without noise is considered**

num = xlsread('Optimal\_filter.xlsx');

[row,column] = size(num);

% Number of unknown variabls is 6 and hence we require 6 independent

% Equations to solve for the true value of state.

H = zeros(6,6);

% we require 9 values including the prior value to calculate values

i = 1;

j = i+3;

% Running loop and storing H as a matrix

while(j<=row)

H(j-3,1) = num(j-1,3);

H(j-3,2) = num(j-2,3);

H(j-3,3) = num(j-3,3);

H(j-3,4) = num(j-1,2);

H(j-3,5) = num(j-2,2);

H(j-3,6) = num(j-3,2);

j= j+1;

end

x = pinv(H)\*num(4:row,3)

size(x)

**OUTPUTS:**

**Mean State Values:**

1.3165

0.2700

-0.5974

0.0307

0.0495

0.0190

**(c) Considering Output with noise – Using the formula given:**

**PROGRAM**

% To generate the value of beta\_k

num = xlsread('Optimal\_filter.xlsx');

[row,column] = size(num);

mean\_value = zeros(row,1);

% Setting the covariance matrix

cov\_value= zeros(row,row);

for i=1:row

for j=1:row

if(i==j)

cov\_value(i,j) = 0.01;

end

end

end

%To convert the covariance matrix to satisfy the equation

%This is done because first 3 rows are not available for calculation (H)

R = zeros(row-3,row-3);

for k=1:row-3

for l=1:row-3

if(k==l)

R(k,l) = 0.01;

end

end

end

% To generate random Gaussian noise with 0 mean and specified variance

v = mean\_value(:) + chol(cov\_value)'\*randn(length(mean\_value),1);

z = num(:,3);

beta = z+v;

ind= 4;

while(ind<=row)

H(ind-3,1) = beta(ind-1,1);

H(ind-3,2) = beta(ind-2,1);

H(ind-3,3) = beta(ind-3,1);

H(ind-3,4) = num(ind-1,2);

H(ind-3,5) = num(ind-2,2);

H(ind-3,6) = num(ind-3,2);

ind= ind+1;

end

% Using the given formula to calculate the results

p = inv(H'\*pinv(R)\*H)

x = p\*H'\*pinv(R)\*beta(4:row,1)

**OUTPUT:**

Mean Value of the State Matrix:

1.0080

0.2842

-0.2982

-0.1598

0.3290

0.0412

Covariance Value :

0.0046 -0.0048 0.0002 -0.0000 0.0009 -0.0017

-0.0048 0.0093 -0.0046 -0.0000 -0.0008 0.0011

0.0002 -0.0046 0.0045 0.0000 -0.0001 0.0006

-0.0000 -0.0000 0.0000 0.0036 -0.0033 0.0002

0.0009 -0.0008 -0.0001 -0.0033 0.0068 -0.0037

-0.0017 0.0011 0.0006 0.0002 -0.0037 0.0043

* The actual value generated for the noiseless output is a bit different from this one. This is because addition of noise to the system will affect the estimation made.

**(d) TO USE MONTE CARLO METHOD**

**PROGRAM:**

% To generate the value of beta\_k

num = xlsread('Optimal\_filter.xlsx');

[row,column] = size(num);

mean\_value = zeros(row,1);

% Setting the covariance matrix

cov\_value= zeros(row,row);

% Running for 1000 iterations

for k= 1:1000

for i=1:row

for j=1:row

if(i==j)

cov\_value(i,j) = 0.01;

end

end

end

% To generate random Gaussian noise with 0 mean and specified variance

v = mean\_value(:) + chol(cov\_value)'\*randn(length(mean\_value),1);

z = num(:,3);

beta = z+v;

z= 4;

while(z<=row)

H(z-3,1) = beta(z-1,1);

H(z-3,2) = beta(z-2,1);

H(z-3,3) = beta(z-3,1);

H(z-3,4) = num(z-1,2);

H(z-3,5) = num(z-2,2);

H(z-3,6) = num(z-3,2);

z= z+1;

end

if (k ==1)

x = pinv(H)\*beta(4:row,1);

else

value = pinv(H)\*beta(4:row,1);

x = horzcat(x,value);

end

end

%To find the value of mean and covariance

mean(x,2)

cov(x')

**OUTPUT:**

Mean Value of State matrix:

1.1193

0.3350

-0.4621

0.0107

0.0489

0.0968

Covariance Value of State Matrix:

0.0090 -0.0147 0.0057 0.0014 -0.0002 -0.0022

-0.0147 0.0326 -0.0181 -0.0006 0.0013 -0.0003

0.0057 -0.0181 0.0126 -0.0008 -0.0010 0.0024

0.0014 -0.0006 -0.0008 0.0100 -0.0129 0.0031

-0.0002 0.0013 -0.0010 -0.0129 0.0251 -0.0130

-0.0022 -0.0003 0.0024 0.0031 -0.0130 0.0110

* It can be seen that the calculated covariance is lesser than the simulated covariance.
* This is because the simulation done is just an approximation of the actual scenario. The variation can be attributed to the fact that we are considering 1000 random Gaussian noise values in simulation but we consider only one random Gaussian noise in the actual result computed.
* However, when we compare the actual state value computed we can see that the Monte Carlo values is close to the actual noiseless values.

**QUESTION 5:**

**PROGRAM:**

dt = 0.25;

Time = 5;

N = (Time/dt)+1;

% Graph plotting values

x = zeros(N,1);

p = zeros(N,1);

t = zeros(N,1);

% Defined values

x(1,1) = 1;

p(1,1) = 2;

t(1,1) = 0;

f = -0.5;

q\_c = 1;

% Condition execution

i = dt; j = 2;

while(i<=5)

x(j,1) = exp((j-1)\*f\*dt)\*x(1,1);

p(j,1) = p(j-1,1)\*(1+(2\*f\*dt))+q\_c\*dt;

t(j,1) = i;

i = i+dt;

j = j+1;

end

%plotting graphs

subplot(2,1,1);

plot(t,x);

xlabel('Time');ylabel('Mean');

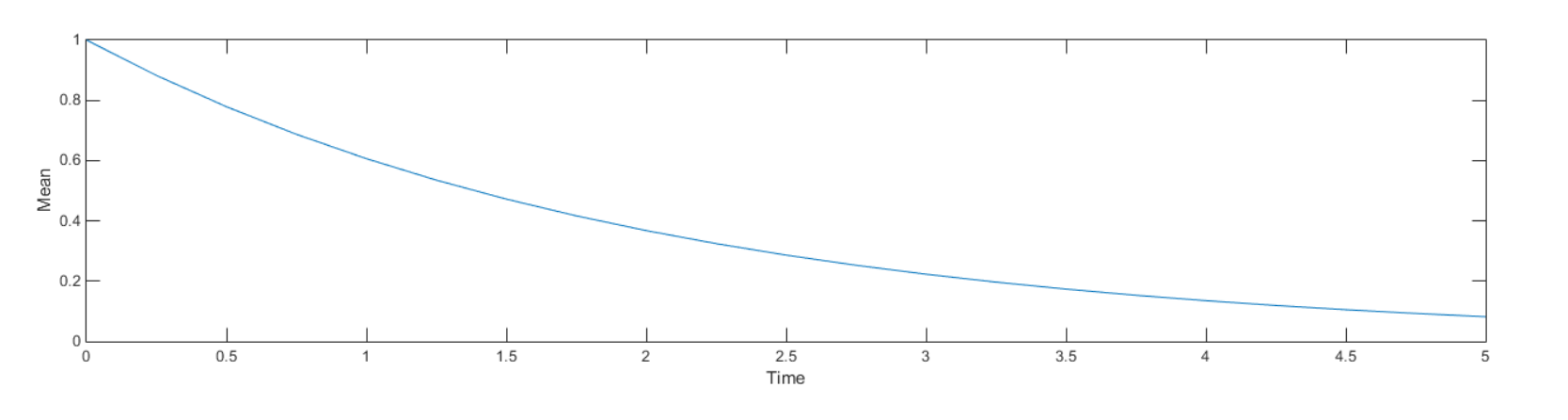
subplot(2,1,2);

plot(t,p);

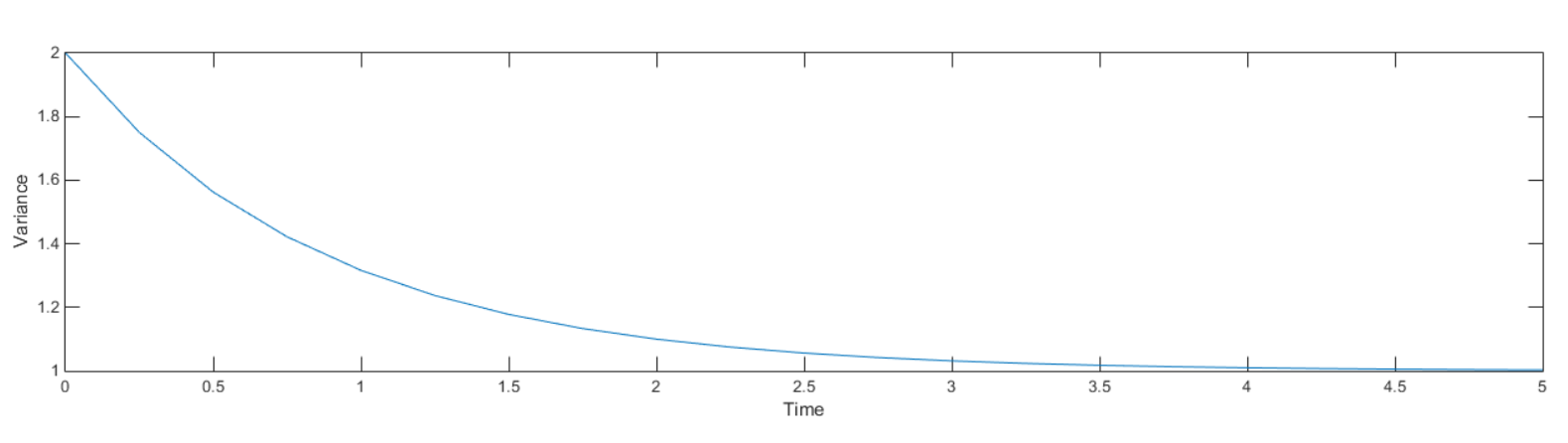
xlabel('Time');ylabel('Variance');

**GRAPH:**

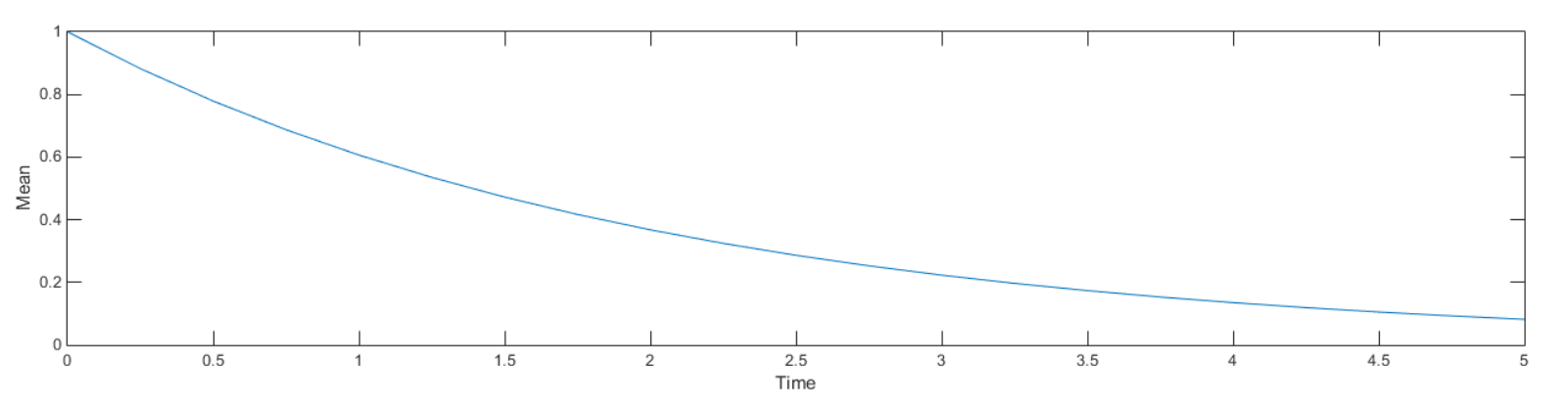
**MEAN FOR P\_0 = 2**

****

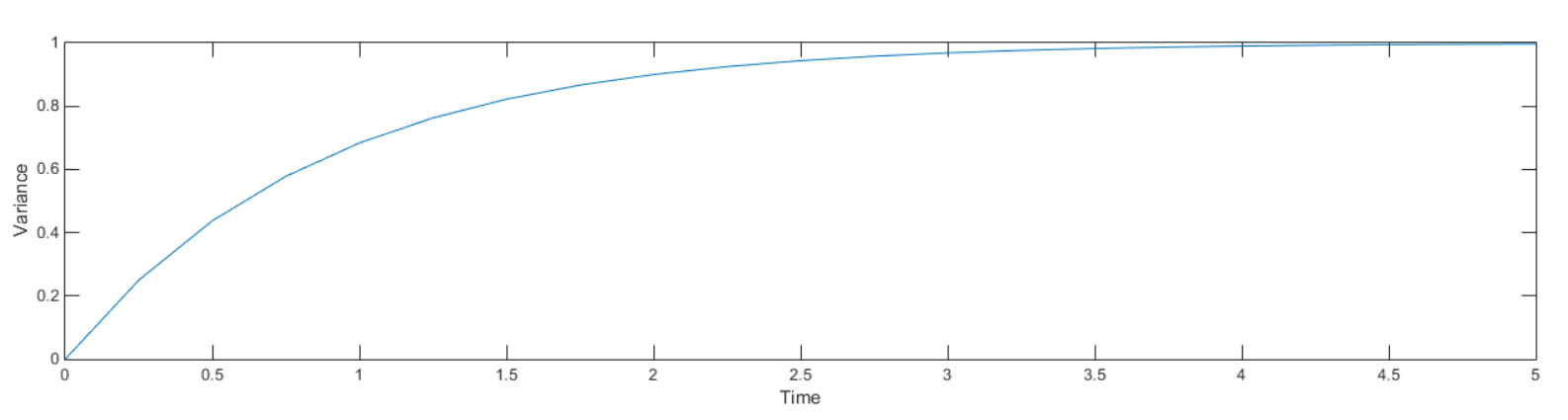
**VARIANCE FOR P\_0 = 2**

****

**MEAN FOR P\_0 = 0**

****

**VARIANCE FOR P\_0 = 0**

****