PROBLEM 12.6 =

Given that n-vector c
we have to determine n-vector h
that can minimize function =

$$\| h * c - e_1 \|^2$$

we know that,

$$h * c = c * h$$

also this is linear function of h.

Hence we can write that =

$$c * h = T(c)\, h$$

where T(c) is Toepliz matrix.

The Toepliz matrix T(c) could be written as

$$T(c) = \begin{bmatrix} c_1 & 0 & 0 & \cdots & 0 \\ c_2 & c_1 & 0 & \cdots & 0 \\ \vdots & c_2 & \ddots & \ddots & \vdots \\ c_n & c_{n-1} & c_{n-2} & \cdots & c_1 \\ 0 & c_n & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & 0 & \cdots & 0 & c_n \end{bmatrix}_{(2n-1)\times n}$$

now,

$$\| c * h - e_1 \|^2 = \| T(c) * h - e_1 \|^2$$

That's why we can say that we can find 'h' by solving least square Problem of minimizing $\| T(c)h - e_1 \|^2$

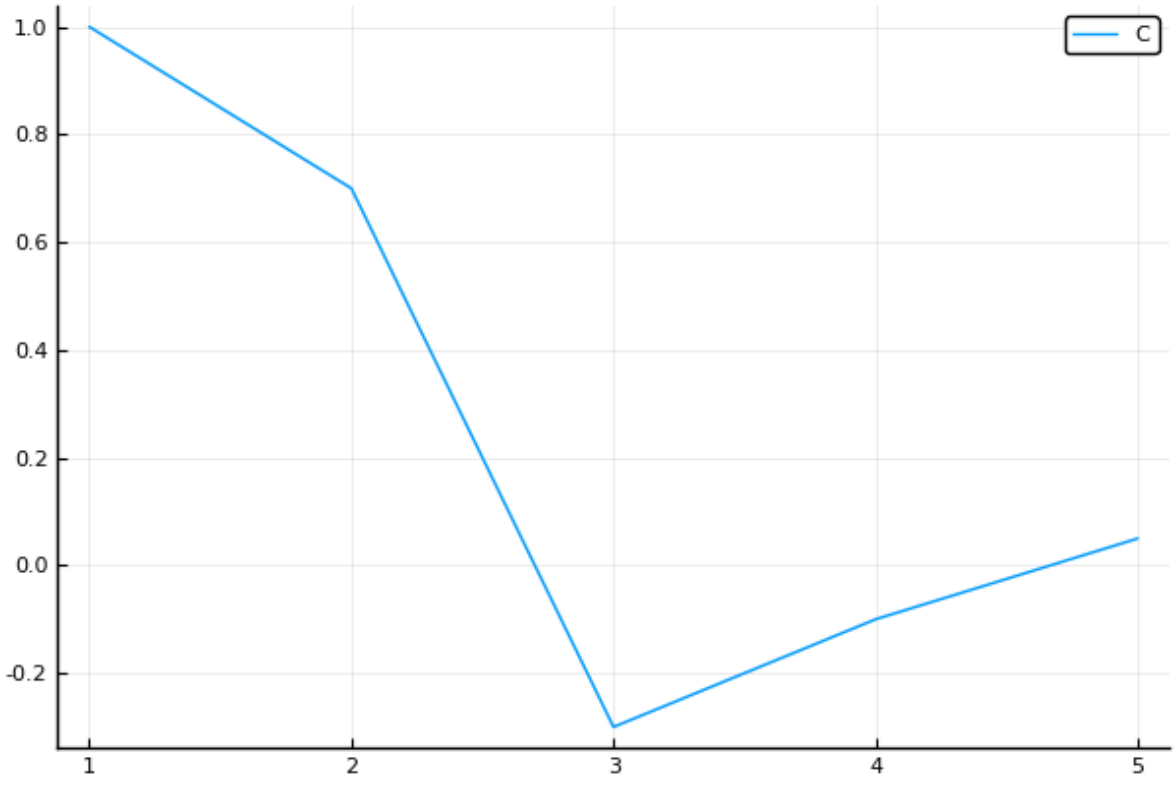
we are given the value of c.

$$c = (1.0, 0.7, -0.3, -0.1, 0.05)$$

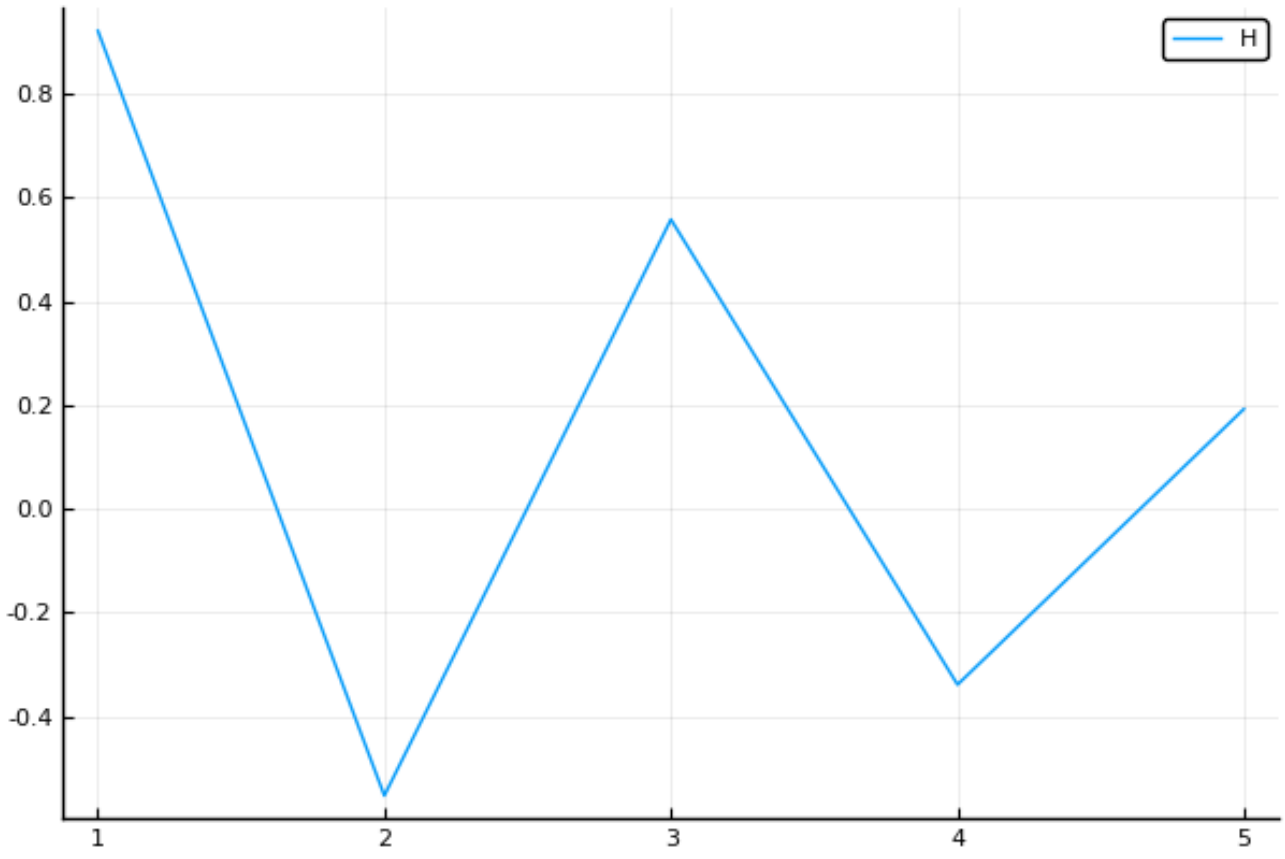
On running the Program & plotting we get

The Julia Program which is run =

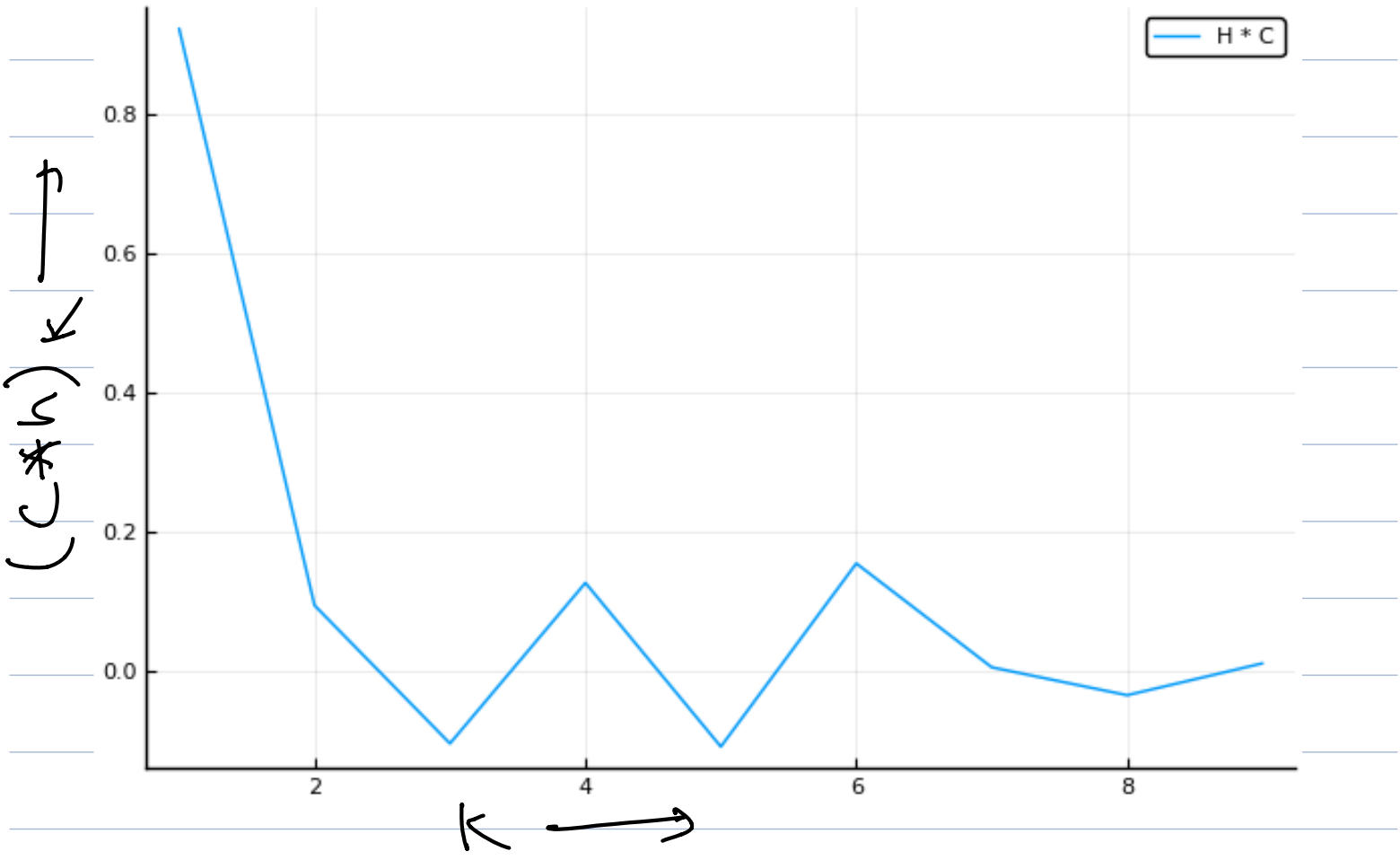
```
using Plots
using DSP
pyplot()

function toeplitz(b,n)
        m = length(b)
        T = zeros(n+m-1,n)
        for i=1:m
                T[i : n+m : end] .= b[i]
        end
        return T
end

c =[1.0, 0.7, -0.3, -0.1, 0.05]
Tc = toeplitz(c, length(c))
e1 = zeros(9)
e1[1] = e1[1] + 1
h = Tc \ e1

plot(c, label="C")
savefig("C.png")
plot(h, label="H")
savefig("H.png")
plot(conv(c,h), label="H * C")
savefig("hxc.png")
```



K →



K →


H * C
0.8
0.6
0.4
0.2
0.0
2
4
6
8
(C*H) K
K

PROBLEM 12.12 =

$$(a) = \|P_{i1} - P_{J1}\|^2 + \|P_{i2} - P_{J2}\|^2 + \cdots + \|P_{iL} - P_{JL}\|^2$$

$$= \left((P_{i1})_1 - (P_{J1})_1\right)^2 + \left((P_{i1})_2 - (P_{J1})_2\right)^2 + \cdots + \left((P_{iL})_1 - (P_{J2})_2\right)^2 + \left((P_{iL})_2 - (P_{J2})_2\right)^2$$

We know $(P_i)_1 = u_i$, $v_i = (P_i)^2$

$$\Rightarrow (u_{i1} - u_{J1})^2 + (u_{iL} - u_{J2})^2 + \cdots + (u_{iL} - u_{JJ})^2 + (v_{i1} - v_{J1})^2 + (v_{iL} - v_{J2})^2 + \cdots + (v_{iL} - v_{JJ})^2$$

$$\Rightarrow D(u) + D(v)$$

(b) = The distance square sum =

$$\|p_{i1} - p_{j1}\|^2 + \cdots + \|p_{iL} - p_{jL}\|^2 = D(u) + D(v)$$

we have proved that in part (a).

given Incidence matrix = B.

let $D(u) + D(v) = \|c\|$

where $C_{2L \times 2N} = \begin{pmatrix} B^T u & 0 \\ 0 & B^T v \end{pmatrix}$

$$= \begin{pmatrix} B^T & 0 \\ 0 & B^T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Rewriting the C in form of Block matrix we get. =

$$C = \begin{pmatrix} D & E & 0 & 0 \\ 0 & 0 & D & E \end{pmatrix} \begin{pmatrix} f \\ G \\ H \\ J \end{pmatrix}$$

$$= \begin{pmatrix} DF + EG \\ DH + EJ \end{pmatrix}$$

$$= \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix} \begin{pmatrix} f \\ H \end{pmatrix} + \begin{pmatrix} EG \\ EJ \end{pmatrix}$$

where D is the first $N-k$ columns of B^T and E is the rest. f is the first $N-k$ elements of u and G is the rest, H is the first $N+k$ elements of v and J is the rest. Hence.

minimize $\|P_{i1} - P_{j1}\|^2 + \cdots + \|P_{i2} - P_{jL}\|^2$

$\Leftrightarrow$ minimize $\|C\| \Leftrightarrow$ minimize $\|Ax - b\|$

Here,

$$A_{2L\times(2N-2k)} = \begin{pmatrix} B^T_{1:(N-k)} & 0 \\ 0 & B^T_{1:(N-k)} \end{pmatrix}$$

$$x_{2(N-k)\times 1} = \begin{pmatrix} u_{1:(N-k)} \\ v_{1:(N-k)} \end{pmatrix}$$

$$b_{2L\times 1} = \begin{pmatrix} -B^T_{(N-k+1):N} & u_{(N-k+1):N} \\ -B^T_{(N-k+1):N} & v_{(N-k+1):N} \end{pmatrix}$$

Problem 13.3 =

(a) = In order to minimize RMS error

we have to minimize following =

$$\sum_{k=1}^{13} \left(\log_{10} n_k - \theta_1 - (t_k - 1970)\theta_2\right)^2 = \|A\theta - b\|^2$$

Hence,

$$A = \begin{bmatrix} 1 & t_1 - 1970 \\ 1 & t_2 - 1970 \\ 1 & t_3 - 1970 \\ \vdots & \vdots \\ 1 & t_{13} - 1970 \end{bmatrix}, \quad b = \begin{bmatrix} \log_{10} N_1 \\ \log_{10} N_2 \\ \vdots \\ \log_{10} N_{13} \end{bmatrix}$$

By Running the Julia Program

we get $\hat{\theta}_1 = 3.126$, $\hat{\theta}_2 = 0.154$

The Julia program which was run =

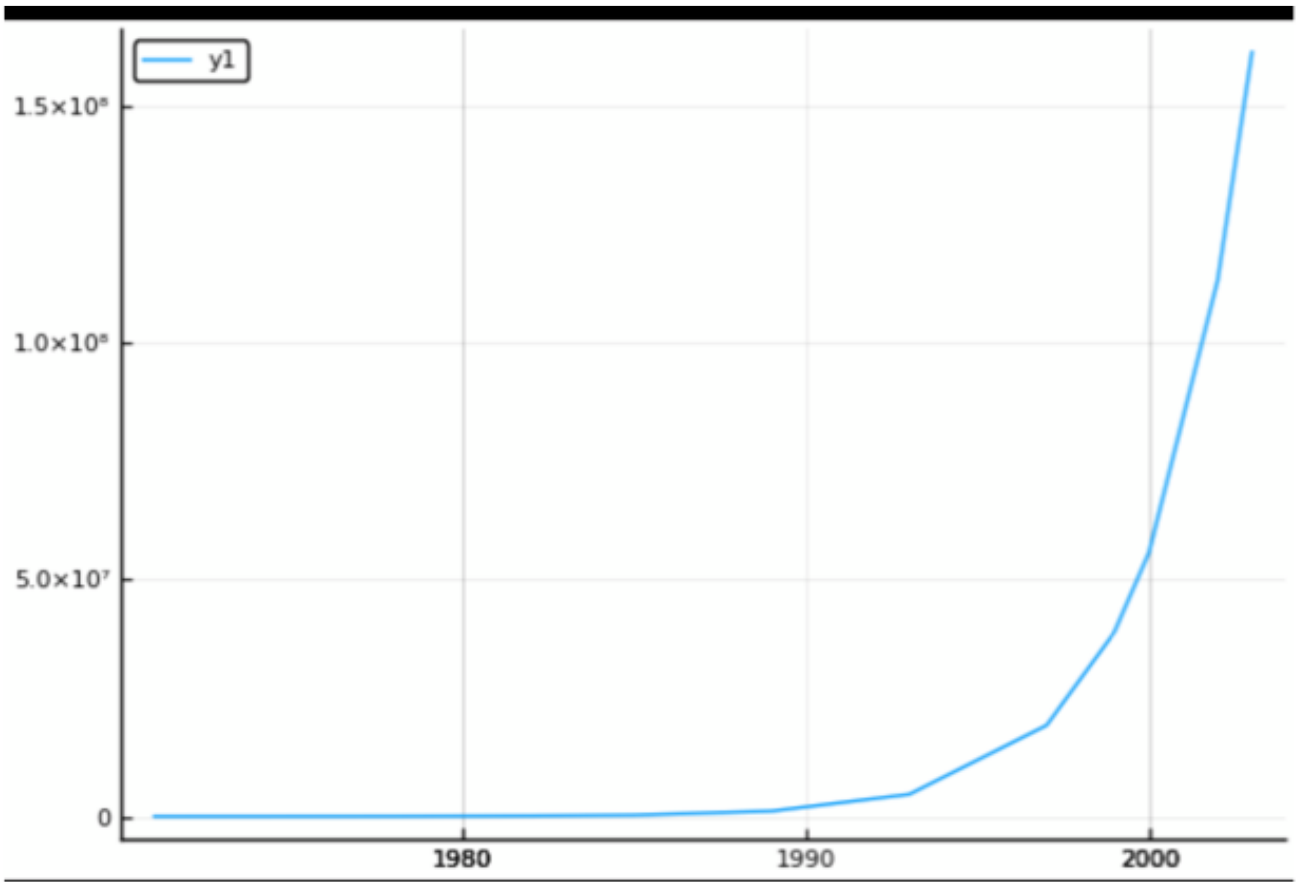
```
using LinearAlgebra
using Plots
A = rand(0:0,13,2)
t = [1971,1972,1974, 1978, 1982, 1985, 1989, 1993, 1997, 1999, 2000, 2002, 2003]
N = [2250, 2500, 5000, 29000, 120000, 275000, 1180000, 3100000, 7500000, 24000000, 42000000,
220000000, 410000000]
b = rand(0.0:0.0,13)
for i = 1:13
        A[i,1] = 1
        A[i,2] = t[i] - 1970
end
for i = 1:13
        b[i] = log(10,N[i])
end
theta = rand(0.0:0.0,13)
theta = A\b
rmsError = norm((A*theta - b))/sqrt(13)
numTransistors = rand(0.0:0.0,13,1)
for i = 1:13
        numTransistors[i] = 10 ^ (theta[1] + theta[2]*(t[i]-1970))
end
pyplot()
plot(numTransistors,t)
println("theta is: ",theta)
println("RMS error is: ",rmsError)
prediction = 10 ^ (theta[1] + theta[2]*(2015-1970))

println("Prediction is: ", prediction)
println("difference in years: " ,log(10,2)/theta[2])
```

Hence RMS error =

$$\frac{1}{\sqrt{13}} \| A\hat{\theta} - b \|$$

$$= 0.20$$

The graph is as below =



(b) = The Prediction of number of Transistors in a microprocessor introduced in 2015 is,

$$10^{Q_1 + Q_2 (2015 - 1970)}$$
$$= 1.14 \times 10^{10}$$

This prediction is about a factor of 3 off from IBM's 213.

(c) = In the model, the number of transistors double approximately every $\log_{10} 2 / Q_2 = 1.95$ years.

We can see that it is consistent with Moore's law.

PROBLEM 13.11 =

(1) = Model A =

Train RMS = 1.355
Test RMS = 1.423

Hence Test RMS value is very close to Train RMS value.

Hence Model A is a good Model.
It is very likely that it will generalize over unseen data.

(2) = Model B =

Train RMS = 9.760
Test RMS = 9.165

Test/Train RMS values are not very close Hence B is a bad model.
But it is likely that it will generalize.

(3) = Model C

Train RMS = 5.033
Test RMS = 0.889

Train / Test RMS values are very different. Model C is Broken.

(4) = Model D

Train RMS = 0.211
Test RMS = 5.072

Test RMS >> Train RMS.

Model D is overfit.

(5) = Model E =

Train RMS = 0.633
Test RMS = 0.633

Test & Train RMS values are exactly equal.

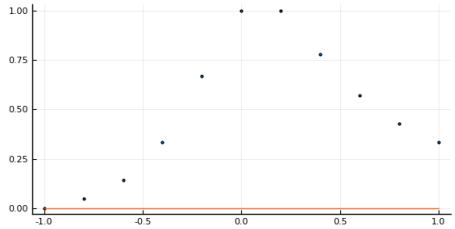
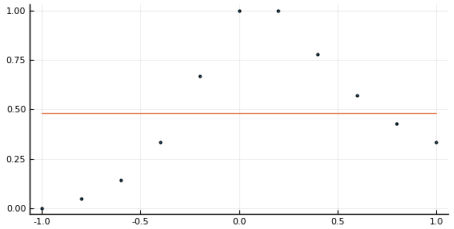
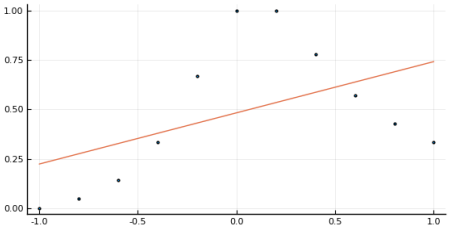
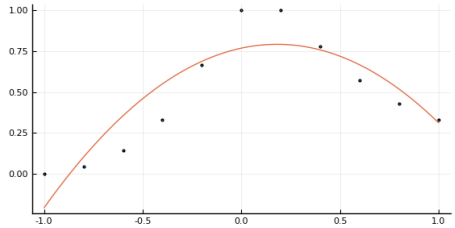
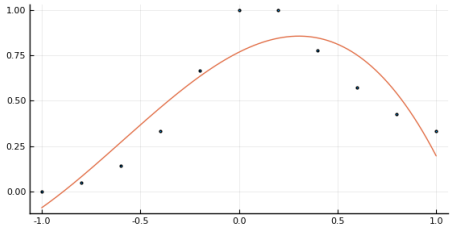
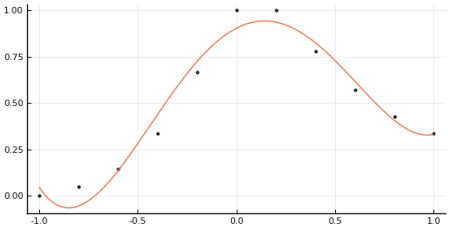
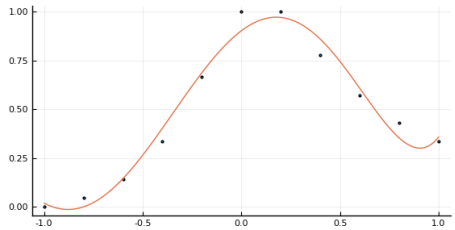
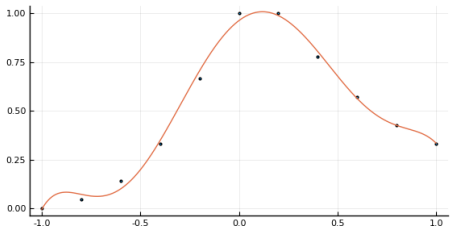
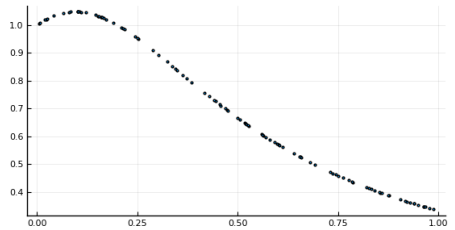
That means Model E is a very very good model.

But it is possible that there was an error in the testing. as both the values are exactly equal. Its not a very general scenario.

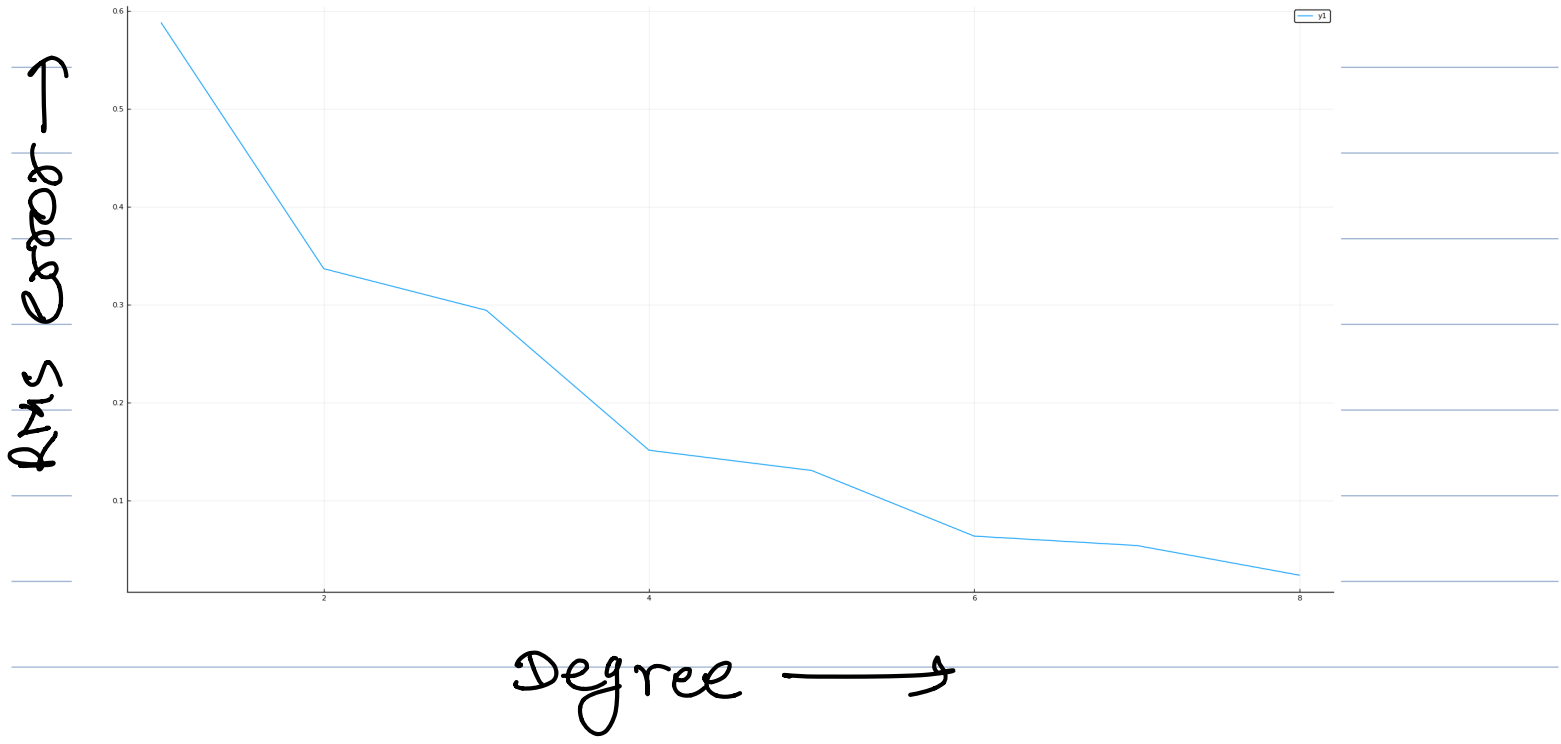
It is possible that model was tested using same training data set.

PROBLEM 13.17 =

fitting a rational function with a polynomial. After running the program following graphs are obtained.

RMS error / degree plot is below =



RMS error was found minimum at degree 8.

The Julia program which was run to get those results is below =

```
using LinearAlgebra
using Plots
using DSP
pyplot()

function vandermonde(t,n)
m = length(t)
V = zeros(m,n)
for i=1:m
        for j=1:n
                V[i,j] = t[i]^(j-1)
        end
end
return V
end
x = rand(0.0:0.0,11,1)
for i = 1:11
        x[i] = -1.0 + 0.2(i-1)
end
y = rand(0.0:0.0,11,1)
for i = 1:11
        y[i] = (1.0 + x[i])/(1+5*(x[i]^2))
end
scatter(x,y,legend=false)
polyfit(x, y, p) = vandermonde(x, p) \ y
theta1 = polyfit(x,y,0)
theta2 = polyfit(x,y,1)
theta3 = polyfit(x,y,2)
theta4 = polyfit(x,y,3)
theta5 = polyfit(x,y,4)
theta6 = polyfit(x,y,5)
theta7 = polyfit(x,y,6)
theta8 = polyfit(x,y,7)
theta8 = polyfit(x,y,7)
polyeval(theta, x) = vandermonde(x,length(theta))*theta;
t_plot = range(-1,stop=1,length=1000);
p = plot(layout=9, legend=false)
scatter!(x, y, subplot=1, markersize = 2)
plot!(t_plot, polyeval(theta1,t_plot), subplot=1)
scatter!(x, y, subplot=2, markersize = 2)
plot!(t_plot, polyeval(theta2,t_plot), subplot=2)
scatter!(x, y, subplot=3, markersize = 2)
```

```
plot!(t_plot, polyeval(theta3,t_plot), subplot=3)
scatter!(x, y, subplot=4, markersize = 2)
plot!(t_plot, polyeval(theta4,t_plot), subplot=4)
scatter!(x, y, subplot=5, markersize = 2)
plot!(t_plot, polyeval(theta5,t_plot), subplot=5)
scatter!(x, y, subplot=6, markersize = 2)
plot!(t_plot, polyeval(theta6,t_plot), subplot=6)
scatter!(x, y, subplot=7, markersize = 2)
plot!(t_plot, polyeval(theta7,t_plot), subplot=7)
scatter!(x, y, subplot=8, markersize = 2)
plot!(t_plot, polyeval(theta8,t_plot), subplot=8)
scatter!(x, y, subplot=9, markersize = 2)
plot!(t_plot, polyeval(theta8,t_plot), subplot=9)
savefig("13_7_.png")
 = 100;
a = rand(m)
b = rand(0.0:0.0,m,1)
for i = 1:m
        b[i] = (1.0 + a[i])/(1+5*(a[i]^2))
end
scatter(a, b)
savefig("13_7_truefunction.png")
m = 10;
aa = rand(0.0:0.0,m)
aa[1]=-0.88
for i = 2:m
        aa[i] = aa[i-1]+0.22
end
aa[5]=0
test = rand(0.0:0.0,m,1)
for i = 1:m
        test[i] = (1.0 + aa[i])/(1+5*(aa[i]^2))
end
error1 = rand(0.0:0.0,8,1)
error2 = rand(0.0:0.0,8,1)
for i = 1:8
        error1[i] = rms(test - vandermonde(aa, i-1)* polyfit(aa,test,i-1))
end
for i = 1:8
        error2[i] = rms(y - vandermonde(x, i-1)* polyfit(x,y,i-1))
end
plot(error1, color= :blue)
plot!(error2, color= :red)
savefig("13_7_error.png")
```

Problem 13.22 =

if the number of submissions are limited.

In that case the should be accurate about the model.

the only way the team can validate the accuracy of prediction is using training data. creating the training - Test Split. and validating the model on Test split.

They can use cross validation to improve the confidence of the model. before predicting.

Case 2 - if there is no limit on number of submissions.

Then they can try different splits of training data.

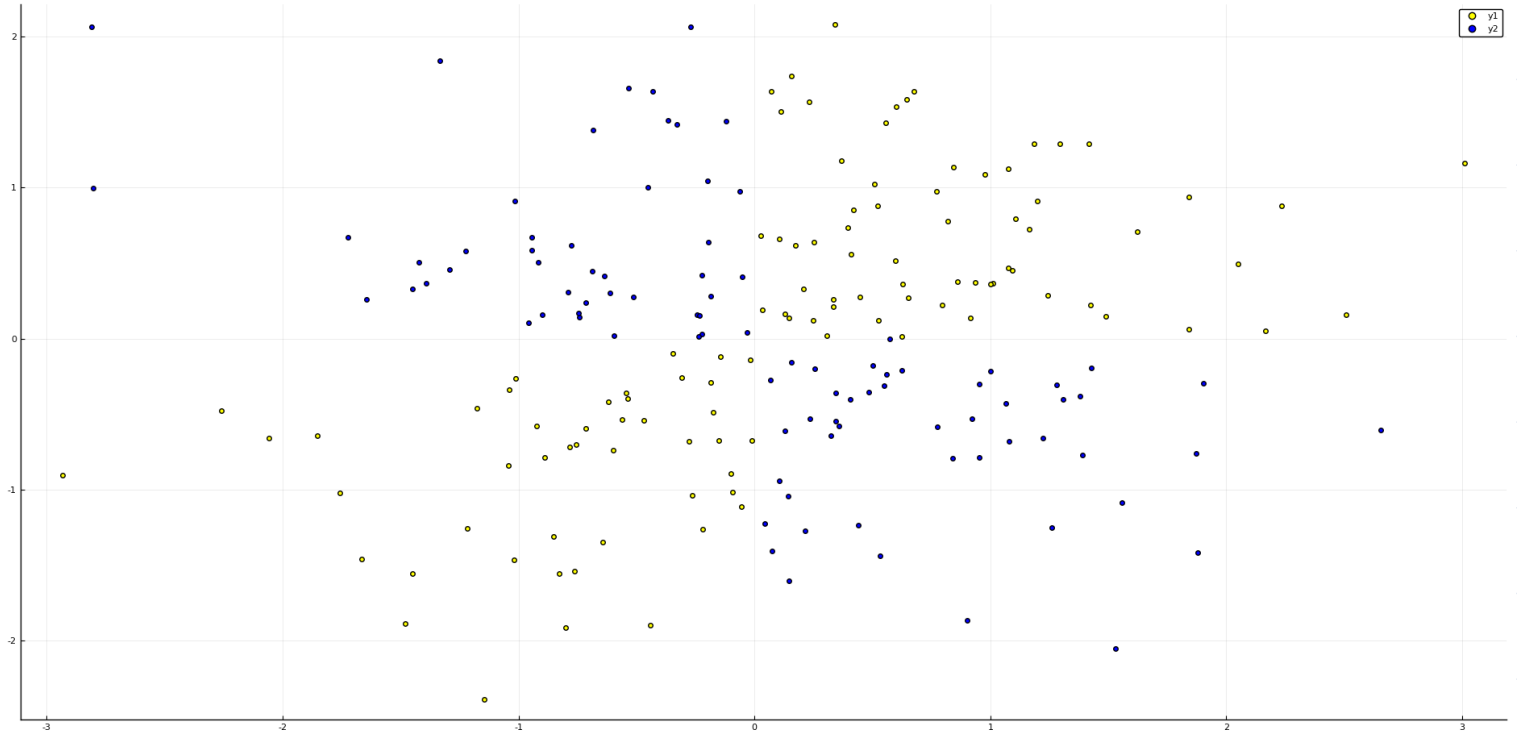
Split = Training data : Test data

try different splits and train the model and Run model to predict the value. and send the Result of every new model.

at some point team can get lucky and train a model. which will given better results.

PROBLEM 14.7 =

The Scatter graph in =



yellow points refer =

$$\hat{f}(x) = 1$$

Blue points refer =

$$\hat{f}(x) = -1$$

Minimizing least Squares -

$$\sum_{i=1}^{200} (\theta_1 + \theta_2 x_1 + \cdots\cdots \theta_6 x_2^2 - y_1)^2$$

$$= \|A\theta - b\|^2$$

Given $\hat{f}(x) = x_1 x_2$

i.e. θ_5 is coefficient of $x_1 x_2$

we observe that $\theta_5 \cong 1$ rest others are near 0.

Error rate of classifier is 5.05.

RMS error = 5.05

(Julia program is below)

The Julia Program is as below =

```
using DSP
using LinearAlgebra
using Plots
x1 = randn(200)
x2 = randn(200)
y = rand(0:0,200)
for i = 1:200
        product = x1[i]*x2[i]
        if product >= 0
                y[i] = 1
        else
                y[i] = -1
        end
end
 f = rand(0.0:0.0,200,1)
 A = rand(0.0:0.0,200,6)
for i = 1:200
        A[i,1] = 1
        A[i,2] = x1[i]
        A[i,3] = x2[i]
        A[i,4] = x1[i] ^ 2
        A[i,5] = x1[i] * x2[i]
        A[i,6] = x2[i] ^ 2
end
theta = rand(0.0:0.0,6,1)
theta = A\y
println(theta)
for i = 1:200
        f[i] = theta[1] + theta[2] * x1[i] + theta[3] * x2[i] + theta[4] * (x1[i] ^ 2) + theta[5] * (x1[i] * x2[i])
+ theta[6] * (x2[i] ^ 2)
end
posSetx = Float64[]
posSety = Float64[]
negSetx = Float64[]
negSety = Float64[]
for i=1:200
        if x1[i]*x2[i]>0
                push!(posSetx,x1[i])
                push!(posSety,x2[i])
        else
                push!(negSetx,x1[i])
                push!(negSety,x2[i])
         end
end
scatter(posSetx,posSety, color= :yellow)
scatter!(negSetx,negSety, color= :blue)
rmsError = rms(A*theta - f)
println("RMS error is: ",rmsError)
```