EE551 Assignment:

Name:Mallavaram Sai Harini

Roll No:214102508

Problems

1. Consider the following discrete-time system

$$\begin{aligned} \boldsymbol{x}(k+1) &= \begin{bmatrix} x_1(k+1) \\ \\ x_2(k+1) \end{bmatrix} = \begin{bmatrix} \frac{x_1(k)}{1+x_2^2(k)} \\ \\ \frac{x_1(k)x_2(k)}{1+x_2^2(k)} \end{bmatrix} + \boldsymbol{w}(k) \\ y(k) &= x_1(k) + v(k) \end{aligned}$$

where, $\boldsymbol{w}(k)$ and $v(k)$ are independent Gaussian noises with covariance matrices Q and R. Write a program to estimate the system states using an extended Kalman filter. Show the convergence of the estimated states starting from a differenet initial condition. Show the effect of noise level on the estimated states by taking at least three different Q and R.

Matlab code:

```
clear all;
clc;
%Extended kalman filter for non linear time varying systems
%State space model
%x(k+1)=f(x(k))+w(k)
%y(k)=h(x(k))+v(k)
%v(k) - Measurement noise, w(k) -  Process noise
%Q - Covariance of w(k) - diagonal matrix , R - Covariance of v(k)
%f(x(k)) - 2*2 , w(k)- 2*2 , h - 1*2 , v - 1*1
N=200;
%Initialization - Assume
x=zeros(2,N);
x(:,1) = [0;0];%True values
xhat(:,1)= [1;2];%Estimated values
%Assuming Q and R
Q1=[2 0;0 2];
Q2=[2 0;0 2];
%Covariance of Q(k)
Q=Q1^(1/2)*Q2^(1/2)
R=3;
p=(x(:,1)-xhat(:,1))*(x(:,1)-xhat(:,1))';
p=[p];
%Construction of jacobian matrix
syms x1 x2
f1= x1/(1+x2^2);
f2= (x1*x2)/(1+x2^2);
A11=diff(f1,x1);
A12=diff(f1,x2);
A21=diff(f2,x1);
A22=diff(f2,x2);
A=[A11 A12;A21 A22]
h=x1;
C11=diff(h,x1);
C12=diff(h,x2);
C=[C11 C12]
%Generation of true states
for k=2:N
    x(1,k)=subs(f1,{x1,x2},{x(1,k-1),x(2,k-1)});
```

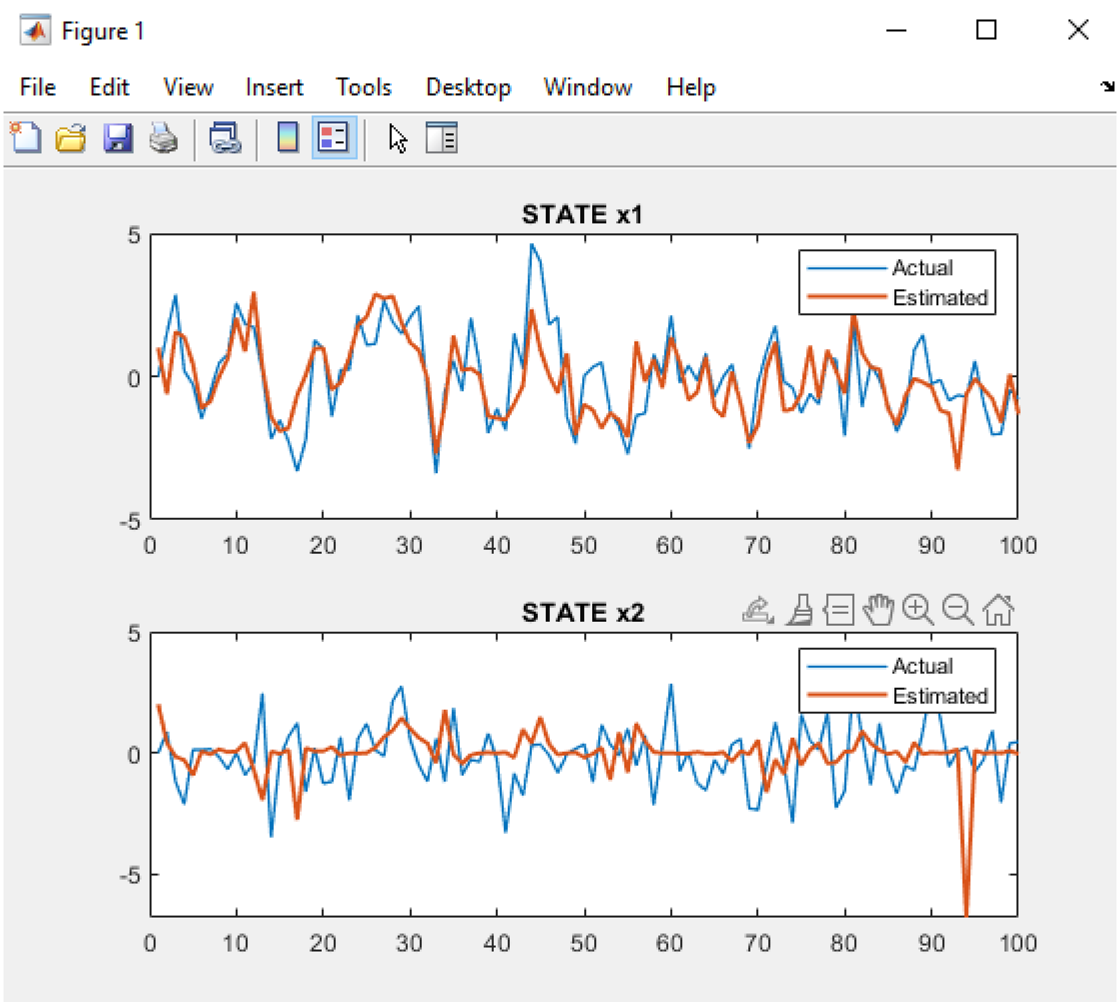
```
    x(2,k)=subs(f2,{x1,x2},{x(1,k-1),x(2,k-1)});
    x(:,k)=[x(1,k);x(2,k)]+[sqrt(2)*randn;sqrt(2)*randn];
end

%Generation of output
%Generate random noise v(k)
v=sqrt(R)*randn(1,N);
y=C*x+v;

for k=2:200
    a11=subs(A11,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    a12=subs(A12,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    a21=subs(A21,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    a22=subs(A22,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    %a11=subs(a11,x2,x(2:k))
    a=[a11 a12;a21 a22];
    size(a);
    %Step 1 : State estimate propogation
    xhatbar1=subs(f1,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    xhatbar2=subs(f2,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    xhatbar=[xhatbar1;xhatbar2];
    size(xhatbar1);
    size(xhat);
    %Step 2: Error covariance propogation
    pnew=a*p*a'+Q;
    %Compute Kalman gain
    c11=subs(C11,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    c12=subs(C12,{x1,x2},{xhat(1,k-1),xhat(2,k-1)});
    c=[c11 c12];
    K=pnew*c'/(c*pnew*c'+R);
    %Step 3: State estimate update
    yhat=subs(h,{x1,x2},{xhatbar1,xhatbar2});
    xhat(:,k)=xhatbar+K*(y(k)-yhat);
    %Step 4: Error covariance update
    p=(eye(2)-K*c)*pnew;

end
s=size(xhat)
%Plotting the states
t=(1:200);
%Plotting state x1
subplot(211);
plot(t,x(1,:),'Linewidth',1);
hold on;
plot(t,xhat(1,:),'Linewidth',1.5);
%legend('Actual','Estimated')
title("STATE x1")
%Plotting state x2
subplot(212);
plot(t,x(2,:),'Linewidth',1);
hold on;
plot(t,xhat(2,:),'Linewidth',1.5);
%legend('Actual','Estimated')
title("STATE x2")
```

Output:
When Q=[2 0;0 2]
And

```
x(:,1) = [0;0];%True values
xhat(:,1)= [1;2];%Estimated values
```

```
Command Window

Q =

    2.0000         0
         0    2.0000


A =

[ 1/(x2^2 + 1),                    -(2*x1*x2)
[x2/(x2^2 + 1), x1/(x2^2 + 1) - (2*x1*x2^2)


C =

[1, 0]


s =

     2   200
```

Plot:



With different Q and initial conditions

Q=[3 0;0 3]

```
x(:,1) = [0;0];%True values
xhat(:,1)= [1;1];%Estimated values
```

Q=[1 0;0 1]

```
x(:,1) = [0;0];%True values
xhat(:,1)= [2;1];%Estimated values
```

It is observed that when Q is low Actual and expected values converge faster

Extended Kalman Filter :-

Kalman filter discussed before deals with LTI system so if we have to extend for non linear time varying systems to estimate states we we estimated states in control system

Extended K F => Non linear & Linear time varying systems

It is converted (not whole thing) when we have Kalman filtering part into LTI system by using some linearization techniques like Taylor series. We construct A, C matrices (B not needed) by using linearization.

Ex consider Non linear Time invarying system

Unforced Non linear system

$$x(k+1) = f(x(k)) + w(k) \rightarrow \text{Process noise}$$

$$y(k) = h(x(k)) + v(k) \rightarrow \text{measurement noise}$$

$w(k)$, $v(k)$ are WGN with covariance matrices as Q and R

$f: R^n$ to R^n : | $h: R^n$ to R^p ($y = p \times 1$)

N/L → $x(k) = n \times 1$ vector

Stage 1 : Following two matrices are constructed

$$A = \frac{\partial f(x(k))}{\partial x(k)} \Bigg|_{x(k) = \hat{x}(k)} \quad \text{Jacobian matrix } (n \times 1)$$

$$C = \frac{\partial h(x(k))}{\partial x(k)} \Bigg|_{x(k) = \hat{x}(k)} \quad (p \times 1)$$

First these two matrices A & C are constructed

If $f = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$ and $h = \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_p \end{pmatrix}$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \Bigg|_{x(k) = \hat{x}(k)}$$

~~$x_k = x$~~

$$C = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial h_p}{\partial x_1} & \frac{\partial h_p}{\partial x_2} & \cdots & \frac{\partial h_p}{\partial x_n} \end{bmatrix} \Bigg|_{x(k) = \bar{\hat{x}}(k)}$$

Stage 2 :- Once the A, and C matrices are computed the non linear state equation is approximated to a linear one using Taylor's series as

$$x(k+1) \cong A x(k) + w(k) + d(k)$$

(for non zero operating point only)

$d(k) \Rightarrow$ known term

$\Downarrow$ Using this to derive KF

$$\bar{y}(k) \cong C x(k) + d(k)$$

$\Rightarrow$ we have to derive like KF, we get same expression as before KF

With we will now apply the Kalman filter theory as follows:

State space model :

$$x(k+1) = f(x(k)) + w(k)$$
$$y(k) = h(x(k)) + v(k)$$

Compute $A = \left(\frac{\partial f}{\partial x}\right)\Big|_{x(k) = \hat{x}(k)}$

$C = \left(\frac{\partial h}{\partial x}\right)\Big|_{x(k) = \bar{\hat{x}}(k)}$

$\Rightarrow$ Algorithmic notation

$\hat{x}(k-1)$ to compute A matrix

For propogation we use previous $\hat{x}$ state. For update we use $\bar{\hat{x}}$ (current)

~~we~~ Initialization : For $k=0$ we know that

$$\hat{x}(0) = E[x(0)] \qquad P(0) = E\left[\{x(0) - \hat{x}(0)\}\{x(0) - \hat{x}(0)\}^T\right]$$

are available

Compute :- For $k = 1, 2 \ldots$

1) State estimate propogation :- we have to find $\bar{\hat{x}}$

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

$f, h \Rightarrow$ known function

Error Covariance propogation :-

$$\bar{P}(k) = A P(k-1) A^T + Q$$

where A is evaluated ~~for~~ at $\hat{x}(k-1)$.

then we compute Kalman Gain matrix

$$K(k) = \bar{P}(k)\, C^T \left[C\, \bar{P}(k)\, C^T + R\right]^{-1}$$

where C is evaluated at $\bar{\hat{x}}(k)$

State estimate update : $\hat{x}(k) = \bar{\hat{x}}(k) + K(k)\left[y(k) - h(\bar{\hat{x}}(k))\right]$

Error covariance update :- $P(k) = (I - K(k)\, C)\, \bar{P}(k)$

when ~~com~~ linear approximation only to calculate A & C

2. Consider the nonlinear system model

$$y(k) = -0.605y(k-1) - 0.163y^2(k-2) + 0.588u(k-1) - 0.240u(k-2) + \xi(k)$$

where $\xi(k)$ is a Gaussian white noise sequence with zero mean and standard deviation 0.2. Generate 200 input-output data by taking the input $u(k)$ as a uniformly distributed random sequence between $[-1,\ 1]$. The objective is to identify both the model structure and the unknown model parameters based on the recorded samples. Start with a full polynomial model with nonlinear degree 3, $n_y = n_u = 2$ and $n_e = 0$. Use an FROLS algorithm to fit an NARX model for the data. Take the value of ESR as 0.05 to terminate the algorithm. Write down the final model terms and corresponding model parameters.

```
clear all;
clc;
%Total no of data points
N=200;
%Initializing u(k) randomly between [-1,1]
u(1)=-1+2*rand;
u(2)=-1+2*rand;
y(1)=0;
y(2)=0;
%White gausian noise with 0 mean and 0.04 variance
%(0.2)^2*randn
%Generating output data
for k=3:200
    y(k)=-0.605*y(k-1)-0.163*(y(k-2)^2)+0.588*u(k-1)-0.240*u(k-2)+0.04*randn;
    u(k)=-1+2*rand;
    %length(y)
end
%z=[u' y']
ybar=y';

ny=2;
nu=2;
ne=0;
l=3;
n=ny+nu+ne;

%Total no of terms
M=factorial(n+l)/(factorial(n)*factorial(l));
%Dictionary D
pm=zeros(198,M);
p0=1;
%l=1
for k=3:200
    p(k,:)=[1 y(k-1) u(k-1) y(k-2) u(k-2) y(k-1)^2 y(k-1)*u(k-1) y(k-1)*y(k-2)
y(k-1)*u(k-2) u(k-1)^2 u(k-1)*y(k-2) u(k-1)*u(k-2) y(k-2)^2 y(k-2)*u(k-2) u(k-
2)^2 (y(k-1)^2)*u(k-1) (y(k-1)^2)*y(k-2) (y(k-1)^2)*u(k-2) (u(k-1)^2)*y(k-1)
(u(k-1)^2)*y(k-2) (u(k-1)^2)*u(k-2) (y(k-2)^2)*y(k-1) (y(k-2)^2)*u(k-1) (y(k-
```

```
2)^2)*u(k-2) (u(k-2)^2)*y(k-1) (u(k-2)^2)*u(k-1) (u(k-2)^2)*y(k-2) y(k-1)^3 u(k-
1)^3 y(k-2)^3 u(k-2)^3 y(k-1)*u(k-1)*y(k-2) y(k-1)*u(k-1)*u(k-2) u(k-1)*u(k-
2)*y(k-2) y(k-1)*y(k-2)*u(k-2)];
end
%step 1
s=1;
q1=p;
size(q1);
size(ybar);
sigma=ybar'*ybar;
for m=1:M
    g(m)=(ybar'*q1(:,m))/(q1(:,m)'*q1(:,m));
    err1(m)=((g(m))^2*q1(m)'*q1(m))/sigma;
end
 [val,idx]=max(err1);
 l1=idx;
 alpha1=p(:,l1);
 q1=p(:,l1);
 a11=1;
 a(1,1)=1;
 g1=g(l1);
 g=[g1];
 err(1)=err1(l1);
 alpha=[alpha1];
 q=[q1];
 l=[l1];
 size(q);
 %step s
 esr=1;
 for x=1:4
 %while esr>0.005
      s=s+1;
      for m=1:M
          count=0;
          for i=1:length(l)
            if(m==l(i))
                l(i);
                count=count+1;
                errs(m)=0;
            end
          end
          if(count==0)
              res=0;
            for r=1:s-1
                temp=((p(:,m)'*q(:,r))/(q(:,r)'*q(:,r)))*q(:,r);
                res=res+temp;
            end
            qm(:,m)=p(:,m)-res;
            g(m)=(ybar'*qm(:,m))/(qm(:,m)'*qm(:,m));
            errs(m)=((g(m)^2)*qm(:,m)'*qm(:,m))/sigma;
          end
      end
      [val,idx]=max(errs);
      ls=idx;
      l=[l ls];
      alphas=p(:,ls);
      alpha=[alpha alphas];
      q=[q qm(:,ls)];
      for r=1:s-1
          a(r,s)=(q(:,r)'*p(:,ls))/(q(:,r)'*q(:,r));
      end
      a(s,s)=1;
      err(s)=errs(ls);
      sum=0;
      for k=1:s
          sum=sum+err(k);
      end
      esr=1-sum;
 end
  x=[];
  for i=1:length(l)
```

```
    x=[x p(:,l(i))];
end
gbar=alpha;
%x*beta=ybar
%beta is found using least squares method
%Sparse model -  Only necessary terms
disp("Sparse model terms")
l
beta=inv(x'*x)*x'*ybar
size(beta);
```

Output:

```
Command Window
Sparse model terms

l =

    25     3     2     5    13


beta =

   -0.0260
    0.5972
   -0.6020
   -0.2347
   -0.1605
```

Problem 2 :

From the matlab code

Sparse model contains the terms

25 3 2 5 13

& beta = $\begin{bmatrix} -0.0260 \\ 0.5972 \\ -0.6020 \\ -0.2347 \\ -0.1605 \end{bmatrix}$

$\therefore$ Sparse model $\Rightarrow$ Identified

$$y(k) = -0.6020\, y(k-1) + 0.5972\, u(k-1) - 0.2347\, u(k-2) - 0.1605\, y^2(k-2) + -0.0260\, u^2(k-2)\, y(k-1)$$

EROLS Algorithm :

OLS $\Rightarrow$ we take first vector as first basis vector

EROLS $\Rightarrow$ Search ERR values for all terms. Highest ERR value vector $\Rightarrow$ First basis vector.

Consider

Linear in the parameters model with $m_e = 0$

(Y) $\bar{y} = [y(1) \ y(2) \ \cdots \ y(N)]^T$

$\bar{p}_m = [p_m(1) \ p_m(2) \ \cdots \ p_m(N)]^T$ for $m = 1, 2 \cdots M$

Let D be the dictionary with all the candidate model terms (depends on l, n_y, n_u)

$D = [\bar{p}_1 \ \bar{p}_2 \ \cdots \ \bar{p}_M]$ $\quad$ D is redundant

D generally has redundant terms

The objective is to find the subset $D_{M_0} = \{\alpha_1 \ \alpha_2 \ \cdots \ \alpha_{M_0}\}$ (which has high ERR values) $= \{\bar{p}_{i_1} \ \bar{p}_{i_2} \ \cdots \ \bar{p}_{i_{M_0}}\}$

Such that $\bar{y} = \theta_1 \alpha_1 + \theta_2 \alpha_2 + \cdots + \theta_{M_0} \alpha_{M_0} + e$

$$\bar{Y} = A\theta + \bar{e}$$

where $A = [\alpha_1 \ \alpha_2 \ \cdots \ \alpha_{M_0}]$

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{M_0} \end{bmatrix}$$

Step 1 :- Start the search with initial full model and the dictionary D

For $m = 1, 2 \cdots M$ let $q_m = \bar{p}_m$ and

let $\sigma = \bar{y}^T \bar{Y}$, calculate $g_m^{(1)} = \dfrac{\bar{Y}^T q_m}{q_m^T q_m}$ (step 1)

ERR step-1 for all candidate model terms

$$ERR^{(1)}(m) = \frac{(g_m^{(1)})^2 \, q_m^T q_m}{\sigma}$$

let $l_1 \Rightarrow$ Index for max ERR value

$$l_1 = \arg\max \ ERR^{(1)}(m) \quad 1 \leq m \leq M$$

$$\Rightarrow \quad ERR^{(1)}(l_1) = \max\{ERR^{(1)}(m) \ ; \ 1 \leq m \leq M\}$$

Choose the first significant basis as

$$\alpha_1 = \bar{p}_{l_1} \ \text{and} \ q_1 = \alpha_1 = \bar{p}_{l_1}$$

$$a_{11} = 1, \quad g_1 = g_{l_1}^{(1)}, \quad err[1] = ERR^{(1)}(l_1)$$

Step s $(s \geq 2)$:

Assume that at step s, a subset D_{s-1} consisting of $(s-1)$ significant model terms / bases $\alpha_1, \alpha_2, \cdots, \alpha_{s-1}$ has already been determined. (At the end of step $s-1$)

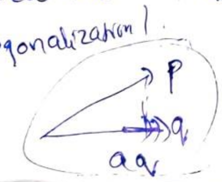
This basis have been transformed into a new orthogonal basis $q_1, q_2 \cdots q_{s-1}$

let $m \neq l_1, m \neq l_2 \cdots m \neq l_{s-1}$

For $m = 1, 2 \cdots M$ (except for the above),

calculate (orthogonalization)

$$q_m^{(s)} = \bar{p}_m - \sum_{r=1}^{s-1} \frac{\bar{p}_m^T q_r}{q_r^T q_r} \cdot q_r$$



$$\bar{p}_m \in D - D_{s-1}$$

$$g_m^{(s)} = \frac{\bar{y}^T q_m^{(s)}}{(q_m^{(s)})^T q_m^{(s)}}$$

1 Moo x

$$ERR^{(s)}(m) = \frac{(g_m^{(s)})^2 (q_m^{(s)})^T q_m^{(s)}}{\sigma}$$

let $$l_s = \underset{\substack{1 \le m \le M \\ m \ne l_1 \cdots, m \ne l_{s-1}}}{\arg\max} \{ERR^{(s)}(m)\}$$

The s^{th} significant basis can then be chosen as

$$\alpha_s = \bar{P}_{l_s} \qquad q_s = q_{l_s}^{(s)}$$

$$a_{r,s} = \frac{q_r^T \bar{P}_{ls}}{q_r^T q_r} \quad \text{for } r = 1, \cdots s-1$$

$$a_{s,s} = 1$$

$$err(S) = ERR^{(s)}(l_s)$$

~~Repeat this step from 2~~

These steps are repeated unless we reach a threshold for ESR, i.e., $ESR = 1 - \sum_{s=1}^{M_0} err(S) \le \delta$ (threshold specified)

(M_0 =) no. of steps)

The final model is the linear combination of Mo significant terms selected from the M candidate terms.

$$y(k) = \sum_{i=1}^{M_0} g_i q_i(k) + e(k)$$

This is equivalent to $y(k) = \sum_{m=1}^{M_0} \beta_{lm} P_{lm}(k) + e(k)$

where ~~If~~ $\bar{\beta} = (\beta_{l_1}, \beta_{l_2} \cdots \beta_{lm})^T$ is calculated from the relation $A\bar{\beta} = \bar{g}$ ($\bar{g}$ we already know, A is calculated)

$$A = \begin{pmatrix} 1 & a_{1,2} & a_{1,3} & \cdots & a_{1,M_0} \\ 0 & 1 & a_{2,3} & \cdots & a_{2,M_0} \\ \vdots & & & & \\ 0 & 0 & \cdots & 1 & a_{M-1,M_0} \\ 0 & 0 & \cdots & & 1 \end{pmatrix} \Rightarrow$$

(It is found when we run FROLS algorithm)

$\bar{\beta}$ can be found

Lecture-34 : FROLS Algorithm : Getting simpler model

Example: Consider a non linear system

$$y(k) = -0.605\, y(k-1) - 0.163\, y^2(k-2) + 0.588\, u(k-1) - 0.240\, u(k-2) + \xi(k)$$
noise

$\xi(k)$: WGN $N(0, 0.01)$ 0 mean, Variance => 0.01

Generate input $u(k)$: (stable s/m)

$u(k)$: Take randomly between $[-1, 1]$

By putting $u(k)$ in initial $y(k)$ (or) $y(0)$ =>

Generate data from the equation

Simulate and generate 200 data

Consider we do not know the system, Identify the system from the data generated. We are using simulated data. In actual cases we have experimental data

(or) If the system is complex use data and identify simpler model

Identification : Take $n_y = 2$, $n_u = 2$, $n_e = 0$, $l = 3$

Check (starting from huge model terms to less model terms)

1 : Constant term ($l = 0$)

4 : linear terms ($l = 1$) => $y(k-1)$, $u(k-1)$, $y(k-2)$, $u(k-2)$

Non linear degree 2

$y^2(k-1)$, $y(k-1)u(k-1)$, $y(k-1)y(k-2)$, $y(k-2)u(k-2)$,
$u^2(k-1)$, $u(k-1)y(k-2)$, $u(k-1)u(k-2)$,
$y^2(k-2)$, $y(k-2)u(k-2)$, $u^2(k-2)$

NL degree 2 => $4 + 3 + 2 + 1 = 10$

Non linear degree 3 :

=> $y^2(k-1)u(k-1)$, $y^2(k-1)y(k-2)$, $y^2(k-1)u(k-2)$,
$u^2(k-1)y(k-1)$, $u^2(k-1)y(k-2)$, $u^2(k-1)u(k-2)$,
$y^2(k-2)y(k-1)$, $y^2(k-2)u(k-1)$, $y^2(k-2)u(k-2)$,
$u^2(k-2)y(k-1)$, $u^2(k-2)u(k-1)$, $u^2(k-2)y(k-2)$,

$y^3(k-1)$, $u^3(k-1)$, $y^3(k-2)$, $u^3(k-2)$,
$y(k-1)\,u(k-1)\,y(k-2)$, $y(k-1)\,u(k-1)\,u(k-2)$,
$y(k-1)\,u(k-2)\,y(k-2)$, $y(k-1)\,y(k-2)\,u(k-2)$)

NL degree 3 => Total => 20 terms

In total => $1+4+10+20 = 35$

Total no $= \dfrac{(n+l)!}{n!\,l!}$ where $n = n_y + n_u + n_e = 4$

(Initial model term) $= \dfrac{(4+3)!}{4! \times 3!} = \dfrac{7\times6\times5\times\cancel{4!}}{\cancel{4!}\times3\times2} = 35$

Run FROLS Algorithm

For a threshold $\rho = 0.05$. (ESR < 0.05) => stop the algorithm

FROLS algorithm will produce the following result

Index	Model terms	Parameters	ERR
1	$y(k-1)$	-0.610	52.15 %
2	$u(k-1)$	0.588	38.35 %
3	$u(k-2)$	-0.239	4.21 %
4	$y^2(k-2)$	$-.0.162$	2.63 %
			Σ 97.34 %

Contribution of $y(k-1)$ is largest => (It is a good model)
(.) ERR is high

lowest contribution => $y^2(k-2)$ => Low ERR