THE UNIVERSITY OF TEXAS AT ARLINGTON, TEXAS
DEPARTMENT OF ELECTRICAL ENGINEERING

EE 5322 - 002

INTELLIGENT CONTROL SYSTEMS

HW # 5 ASSIGNMENT

by

SOUTRIK PRASAD MAITI

1001569883

Presented to

Dr. Frank Lewis

Nov 21, 2017

EE 5322 Intelligent Control

Fall 2017

Homework Pledge of Honor

On all homeworks in this class - YOU MUST WORK ALONE.

Any cheating or collusion will be severely punished.

It is very easy to compare your software code and determine if you worked together

It does not matter if you change the variable names.

Please sign this form and include it as the first page of all of your submitted homeworks.

....

Typed Name: Soutrik Maiti

Pledge of honor:

"On my honor I have neither given nor received aid on this homework."

e-Signature: Soutrik Maiti

Problem 1

a) *MATLAB Code :*

```
%State Matricies
a = [0,1;-0.89,1.8];                          %Matrix A

b = [0,1]';                                    %Matrix B

k = [1:1:100]';                                %Time Index

u = ones(size(k));                             %Unit Step

x0 = [0,0]';                                   %Initial Conditons

[ki x] = discrete_time_sys(a,b,x0,u);

plot(ki,x)                                     %Plotting StateVariables
title('Response of state 1 and state 2')
xlabel('Time');

ylabel('Amplitude');
legend('x1','x2');
figure
plot(ki,x(:,1))
title('Response of state 1')

xlabel('Time');
ylabel('Amplitude');
legend('x1');
figure
plot(ki,x(:,2))
title('Response of state 2')
xlabel('Time');
ylabel('Amplitude');
legend('x2');
function[ki,x] = discrete_time_sys(a,b,x0,u)
    N = size(u);                           %N = 100s

    ki(1) = 1;                             %Time Index ki

    n = size(x0);

    x = zeros(N(1),n(1)); x(1,:) = x0'; %Initalizing State variable & x(0)

    for k = 1:N(1)-1                       %Loop for calculation of State variables

        ki(k+1) = k+1;

        x(k+1,:) = (a*x(k,:)'+ b*u(k,:)')';
    end

    ki = ki';


 end
```

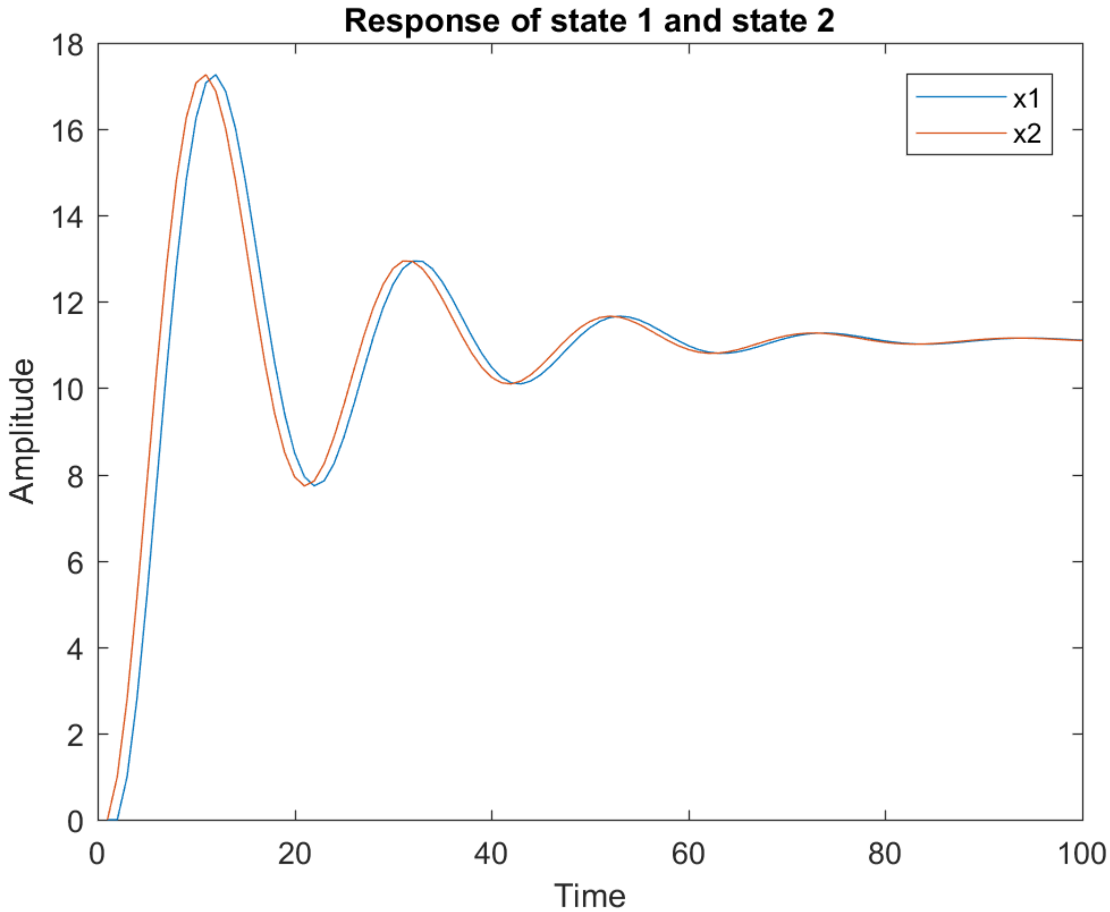

Fig 1.1 Response of both the states

Period of the system – Fig1.2 shows the coordinates of the first and second peak. Hence the period is simply x2-x1+1 i.e. 32-12+1 which is 21.
The period is 21.
The period is same for the second state as inferred from Fig 1.3.
As of the peak overshoot is considered, we know:-

$$\boldsymbol{Peak\ Overshoot\ (State\ 1)} = \frac{V_{Peak} - V_{Constant}}{V_{Constant}} \times \mathbf{100} = \mathbf{54.88\%}$$

For State 1:
Vpeak = 17.27
Vconstant = 11.15 from Fig

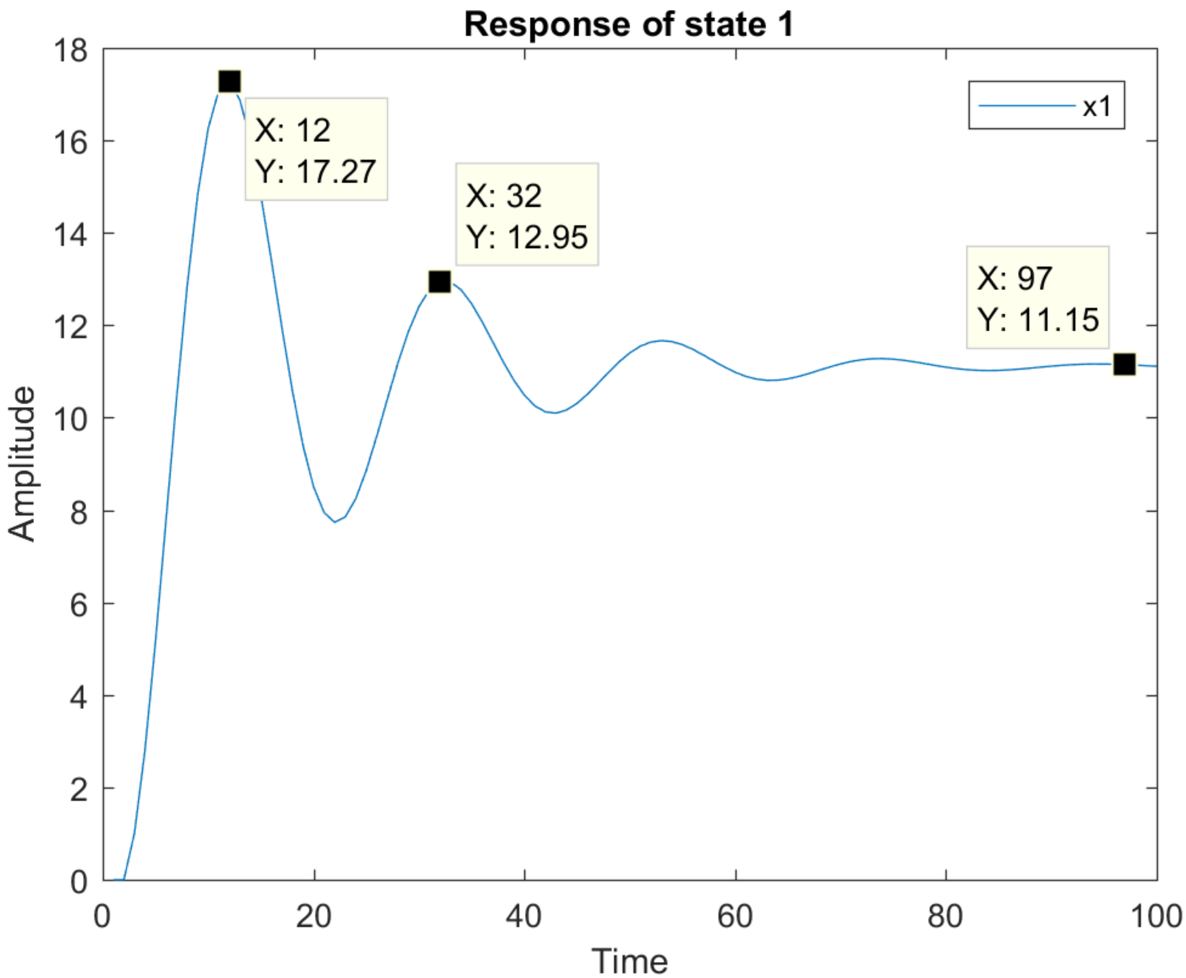

Fig 1.2 Response of state 1

For State 2:
Vpeak = 17.27
Vconstant = 11.15 from Fig

$$\boldsymbol{Peak\ Overshoot\ (State\ 2)} = \frac{V_{Peak} - V_{Constant}}{V_{Constant}} \times \mathbf{100} = \mathbf{54.74\%}$$

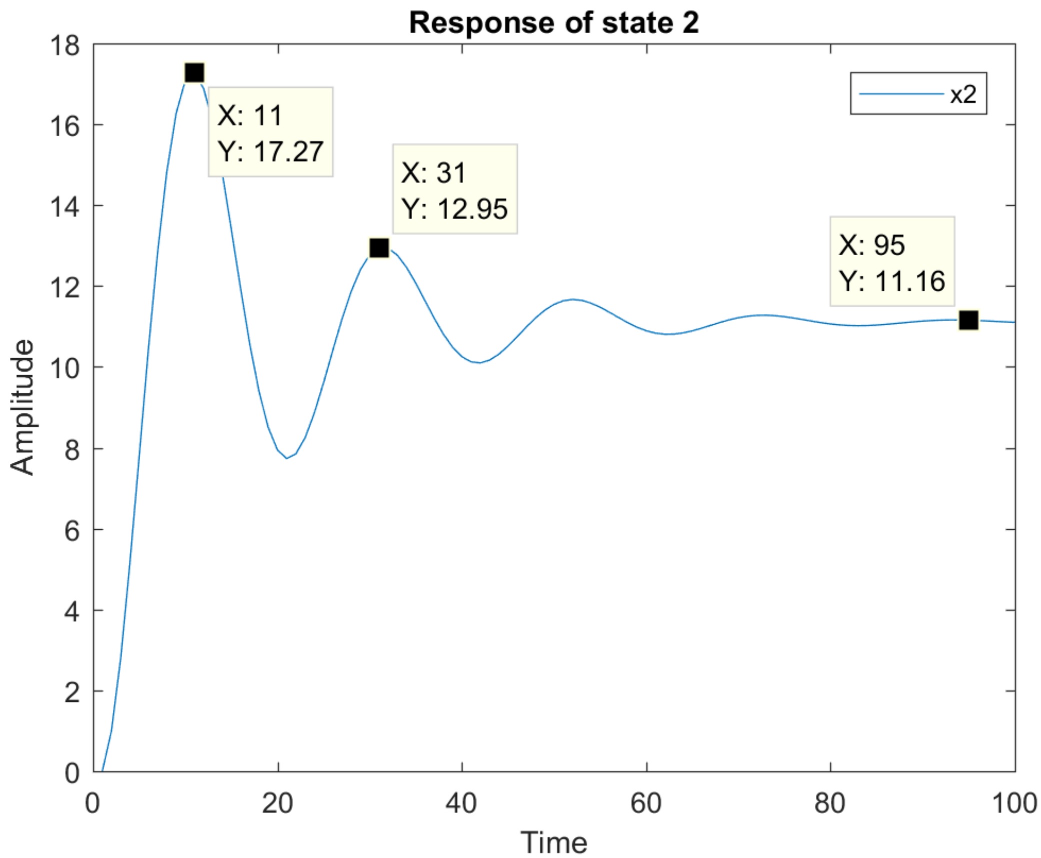

Fig 1.3 Response of state 2

b) MATLAB code for state response with noise

```
%state matricies

a = [0,1;-0.89,1.8];                             %Matrix A

b = [0,1]';                                      %Matrix B



x0 = [0 0]';                                     %Initial State x(0)

k = [1:1:100]';                                  %Time Index

u = ones(size(k));                               %Unit Step input

[ki,x] = disc_time_sys_wnoise(a,b,x0,u);

plot(ki,x)                                       %Plotting State Variables
title('State response with process noise')
xlabel('Time');

ylabel('Amplitude');

function [ki,x] = disc_time_sys_wnoise(a,b,x0,u)

    N = size(u);                                     %N = 100

    n = size(x0);

    ki(1) = 1;
```

```
        x = zeros(N(1),n(1));                                   %Initializing States

        x(1,:) = x0';                                           %x(0)

        r = (0.2)*rand(100,2);                            %Process noise uniformly dist 0-0.2

        for k = 1:N(1)-1                                        %Calculating State Vatiables

            ki(k+1) = k+1;

            x(k+1,:) = (a*x(k,:)' + b*u(k,:)' + r(k,:)')';
        end

        ki =ki';
    end
```

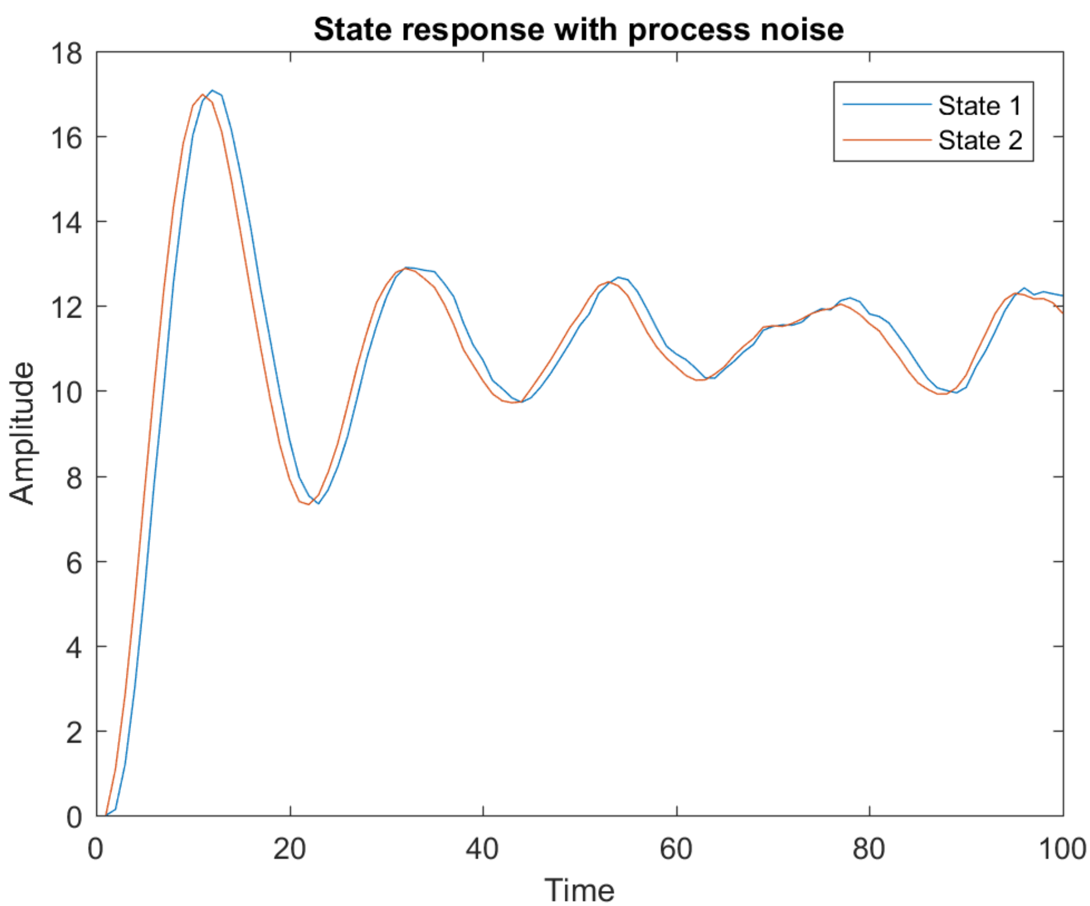

Fig 1.4 State response when there is process noise

Problem 2:

MATLAB Code for Optimal Time Varying Kalman Filter

```
clear all
 clc
 close all
%System Matrices

A = [0,1;-0.89,1.8];

B = [0;1];
```

```
H = [1,0];

Q = 0.1*eye(2);

R = 0.1;

G = eye(2);

wk = sqrt(0.1)*randn(2,100);            %Process noise

vk = sqrt(0.1)*randn(100,1);            %Measurement Noise

P = 35*eye(2);                        %Initial Error Covariance P

I = eye(2);

k = 100;

x0 = [0 0];                           %Initial value of x

uk = ones(k,1);                         %Unit Step

x = zeros(100,2);                       %Initalizing states

xhat(2,1) = 0;                          %Initial x estimate values

x(1,:) = x0';

% State Calculations
for k = 1:100

    x(k+1,:) = (A*x(k,:)'+ B*uk(k,:)'+G*wk(:,k))';

    zk(k,:)= (H*x(k,:)'+vk(k,:))';

end

% Optimal Time Varying DT Kalman Filter

for k = 1:99

    Pm= A*P*A'+ G*Q*G';

    xhatn(:,k+1) =  (A*xhat(:,k) + B*uk(k,:)');

    K = Pm*H'*inv(H*Pm*H'+R);

    P = (I-K*H)*Pm;

    xhat(:,k+1) = xhatn(:,k+1)+K*(zk(k+1,:)-H*xhatn(:,k+1));

end

%Plot for first state
figure(1)
O= plot(1:100,x(1:100,1),'-r',1:100,xhat(1,:)','-b');
title('Optimal Time Varying DT Kalman Filter for state 1');
set(O(1), 'LineWidth', 1);
set(O(2), 'LineWidth', 1.7);
legend('x(1)','x(1)Hat(Estimate)')

%Plot for second state
```

```
figure(2)
U = plot(1:100,x(1:100,2),'-r',1:100,xhat(2,:)','-b');
title('Optimal Time Varying DT Kalman Filter for state 2');
set(U(1), 'LineWidth', 1);
set(U(2), 'LineWidth', 1.7);
legend('x(2)','x(2)Hat(Estimate)')
```

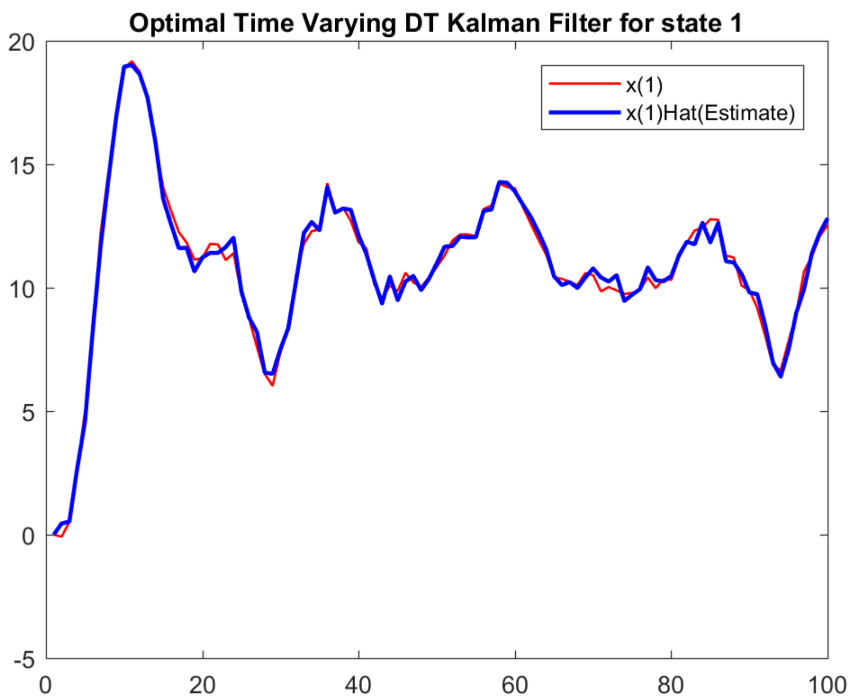

Fig 2.1 DT filter estimates for state 1

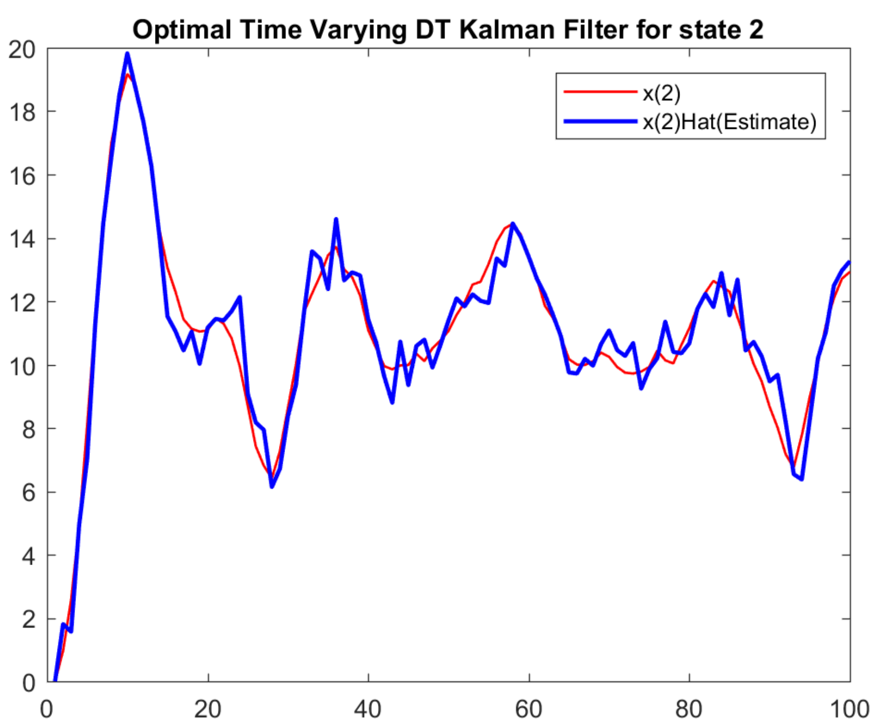

Fig 2.2 DT filter estimates for state 2

Problem 3:

MATLAB code for steady state Kalman filter

```
clear all;
clc;
close all;
%System Matrices

A = [0,1;-0.89,1.8];

B = [0;1];

H = [1,0];

Q = 0.1*eye(2);

R = 0.1;

G = eye(2);

wk = sqrt(0.1)*randn(2,100);            %Process noise

vk = sqrt(0.1)*randn(100,1);            %Measurement Noise

P = 37* eye(2);                       % Initalizing Error covariance

I = eye(2);

k = 100;

x0 = [0 0];                           %Initial value of x

uk = ones(k,1);                         %Unit Step

x = zeros(100,2);                       %Initalizing states

xhat = zeros(2,100);

xhatn = zeros(2,100);

xhat(1,:) = 0;

x(1,:) = x0';

%State Simulation
for k = 1:100

    x(k+1,:) = (A*x(k,:)'+ B*uk(k,1)'+ G*wk(:,k))';
    zk(k,:)= (H*x(k,:)'+vk(k,:)')';

end


% Steady State DT Kalman Filter


%Riccati Equation
for k1 = 1:99

     P = A*(P-P*H'*inv(H*P*H'+R)*H*P)*A'+G*Q*G';
```

```
end
Ks1 = P*H'*inv(H*P*H'+R)   %steady state kalman gain from Riccati equation


%DLQE for the ARE
[M,P2,Z,E] = dlqe(A,G,H,Q,R);

    Ks2 = P2*H'*inv(H*P2*H'+R)  %Steady state kalman gain



for k2=1:99

    xhatm(:,k2+1) =  A*(I-Ks2*H)*x(k2,:)' + B*uk(k2,:)' + A*Ks2*zk(k2,:);
end


%optimal DT Kalman Filter

P = 37* eye(2);                        % Initalizing Error covariance
for k = 1:99

    Pm= A*P*A'+ G*Q*G';

    xhatn(:,k+1) =  (A*xhat(:,k) + B*uk(k,:)');

    Ko = Pm*H'*inv(H*Pm*H'+R);

    P = (I-Ko*H)*Pm;

    xhat(:,k+1) = xhatn(:,k+1)+Ko*(zk(k+1,:)-H*xhatn(:,k+1));

end

%Comparision of first state
figure(1)
O = plot(1:100,x(1:100,1),'-r',1:100,xhat(1,:)','-b',1:100,xhatm(1,:)','-g');
title('Kalman Filters comparision for state 1');
set(O(1), 'LineWidth', 1);
set(O(2), 'LineWidth', 1.7);
set(O(3), 'Linewidth', 1.3);
legend('x(1)','x(1) Optimal','x(2) Steady-State')

%Comparision of second state
figure(2)
U = plot(1:100,x(1:100,2),'-r',1:100,xhat(2,:)','-b',1:100,xhatm(2,:)','-g');
title('Kalman Filters comparision for state 2');
set(U(1), 'LineWidth', 1);
set(U(2), 'LineWidth', 1.7);
set(U(3), 'LineWidth', 1.3);
legend('x(2)','x(2) Optimal','x(2) Steady-State')
```

```
Ks1 =

    0.8607
    1.1552


Ks2 =

    0.8607
    1.1552

>>
```

Fig 3.1 Comparison of Kalman gains from two approaches

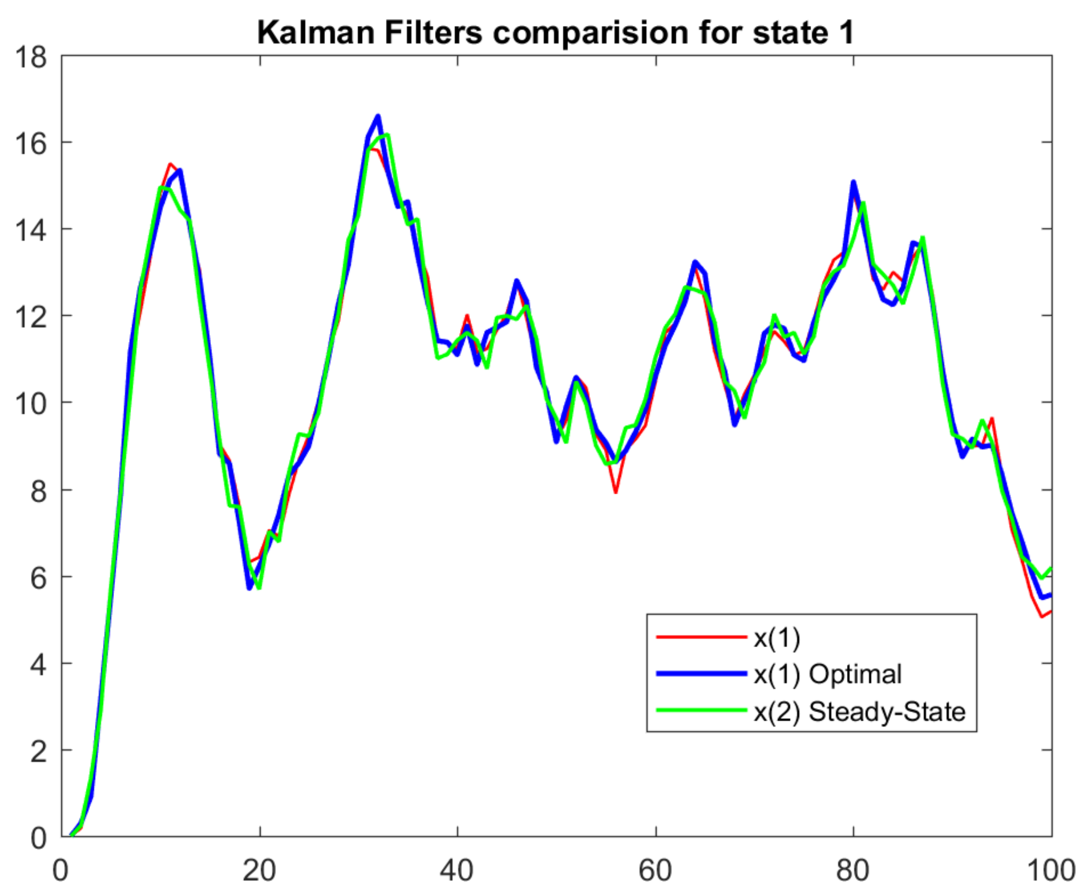

Fig 3.2 Comparison of estimates from two different Kalman filters for state 1

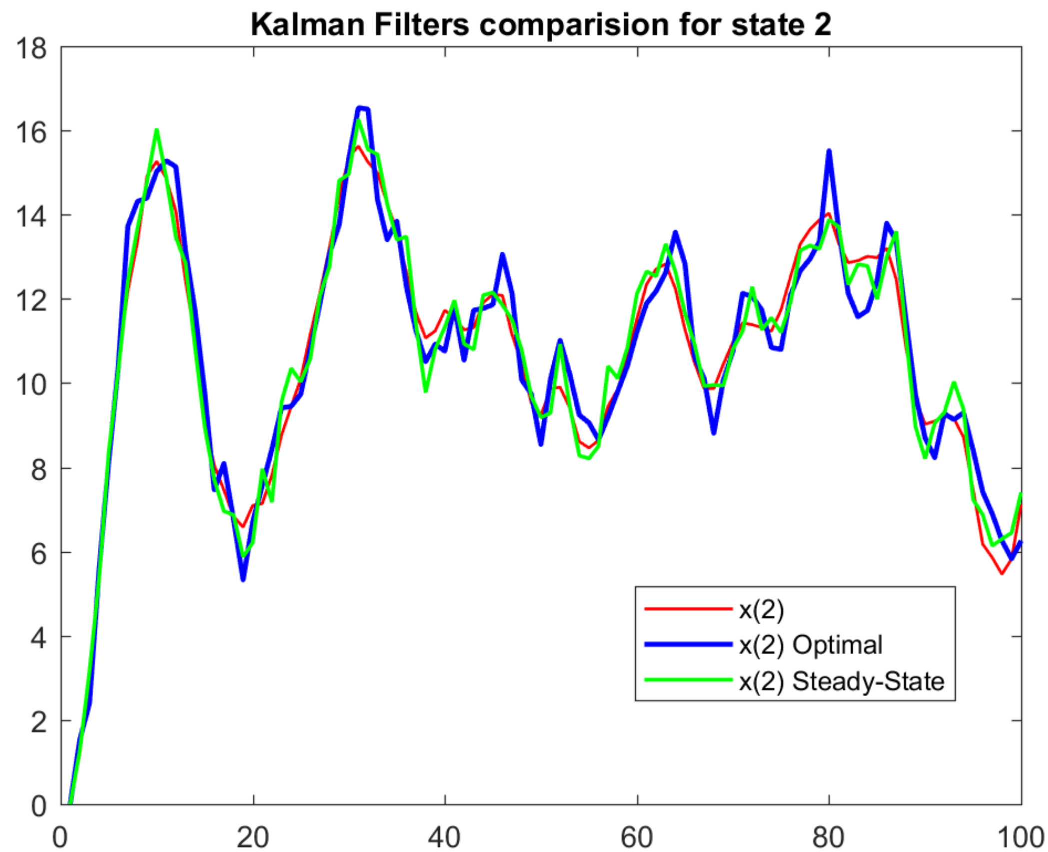

Fig 3.3 Comparison of estimates from two different Kalman filters for state 2