Date: 28 Feb, 2014 M.S. Raghavendra Sriram

1000854840

EE 5322 Intelligent Control Systems

Assignment no 3

# Discrete Time Simulation, Observers, Kalman Filter

1. **Discrete-Time System.**

Q1.a)

Solution :

clc;

clear all;

close all;

X=zeros(2,100);

Xn=zeros(2,100);

A=[0 1

-0.89 1.8];

B=[0

1];

u=ones(100,1);

t=1:100;

for i=1:99

X(:,i+1)=A\*X(:,i)+B\*u(i);

y(i,:) = X(:,i+1)'

hold on

end

figure(1)

plot(t,X(1,:));

hold on;

grid on;

plot(t,X(2,:),'r');

legend t X

grid on;

title('System States Without Noise')

max\_overshoot = max(y(:,1))

SteadyState\_value = max(y(i,:))

peak\_overshoot = (max\_overshoot(1) - SteadyState\_value)/SteadyState\_value \*100

Solution :

y =

0 1.0000

1.0000 2.8000

2.8000 5.1500

5.1500 7.7780

…

11.1361 11.1243

11.1243 11.1126

11.1126 11.1020

max\_overshoot =

17.2655

SteadyState\_value =

11.1126

peak\_overshoot =

55.3691

Plots

:

Q1.b)

Code :

clc;

clear all;

close all;

x=zeros(2,100);

xn=zeros(2,100);

A=[0 1

-0.89 1.8];

B=[0

1];

u=ones(100,1);

t=1:100;

w = 0.2\*rand(2,100);

for i=1:99

x(:,i+1)=A\*x(:,i)+B\*u(i)+w(i);

y(i,:) = x(:,i+1)'

hold on

end

figure (1)

plot(t,x(1,:),t,x(2,:));

hold on;

plot(t,x(2,:),'r');

grid;

title('System States With Noise')

max\_overshoot = max(y())

min\_overshoot = max(y(i,:))

peak\_overshoot = (max\_overshoot(1) - min\_overshoot)/min\_overshoot \*100

Solution :

y =

0.1054 1.1054

1.1782 2.9687

2.9827 5.3091

5.3588 7.9514

7.9632 10.5551

…

11.5183 11.3675

11.4731 11.3158

11.4842 11.3258

11.3617 11.2014

11.2566 11.1058

max\_overshoot =

17.2334

min\_overshoot =

11.2566

peak\_overshoot =

53.0955

Plot:



2.

CODE :

clc;

clear all;

close all;

A=[0 1 % State space description of system

-0.89 1.8];

B=[0

1];

u=ones(100,1);

H=[2 0];

x(:,1)=[0;0];

X\_time\_mes(:,1)=[10

15]; %initial wrong estimate of states

P=100\*eye(2); %initial guess of error covariance

P\_cov(:,1)=diag(P);

G=eye(2);

Mean=.1\*eye(2);

Var=.1;

%% Implement DT Kalman Filter

for i=1:100

x(:,i+1) = A \* x(:,i) + B \* u(i) + 0.1\*randn(2,1); % system at time i+1

z(i+1) = H \* x(:,i+1) + Var \* randn; %measurement at i+1

x\_copy(:,i+1) = x(:,i+1);

X\_time\_update(:,i+1) = A \* X\_time\_mes(:,i) + B \* u(i); %Time Update state at i+1

P = A \* P \* A' + Mean \* (G \* G)'; %Time Update Error Covariance

P\_error\_cov(:,i+1) = diag(P);

P = P - P \* H' \* 1/(H \* P \* H' + Var) \* H \* P; %Cov Measurement Update

P\_updated\_cov(:,i+1) = diag(P); %track time updated dovariance

KalmanGain = P \* H' \* 1/(Var); %Kalman Gain

Kalman\_Gain\_update(:,i)=KalmanGain; %Record kalman gain

X\_time\_mes(:,i+1) = X\_time\_update(:,i+1) + KalmanGain \* (z(i+1) - H \* X\_time\_update(:,i+1)); %Measurement Update

x(:,i+1)=X\_time\_mes(:,i+1); %Feedback of updated state

end

%% Plot the states Estimates and the Actual states

t=1:101;

figure (1)

plot(t,X\_time\_mes(1,:),'b');

hold on;

plot(t,x\_copy(1,:),'r')

hold on;

grid on;

legend('State 1 - Estimated','State 1 - Actual')

figure (2)

plot(t,X\_time\_mes(2,:),'r');

hold on

plot(t,x\_copy(2,:),'b');

grid on;

hold on;

legend('State 2 - Estimated','State 2 - Actual')

Plot :





Q3.

Code:

clc;

clear all;

close all;

A=[0 1 % State space description of system

-0.89 1.8];

B=[0

1];

u=ones(100,1);

H=[2 0];

x(:,1)=[0;0];

X\_time\_mes(:,1)=[10

15]; %initial wrong estimate of states

P=100\*eye(2); %initial guess of error covariance

P\_cov(:,1)=diag(P);

G=eye(2);

Mean=.1\*eye(2);

Var=.1;

%% Implement DT Kalman Filter

for i=1:100

x(:,i+1) = A \* x(:,i) + B \* u(i) + 0.1\*randn(2,1); % system at time i+1

z(i+1) = H \* x(:,i+1) + Var \* randn; %measurement at i+1

x\_copy(:,i+1) = x(:,i+1);

X\_time\_update(:,i+1) = A \* X\_time\_mes(:,i) + B \* u(i); %Time Update state at i+1

P = A \* P \* A' + Mean \* (G \* G)'; %Time Update Error Covariance

P\_error\_cov(:,i+1) = diag(P);

P = P - P \* H' \* 1/(H \* P \* H' + Var) \* H \* P; %Cov Measurement Update

P\_updated\_cov(:,i+1) = diag(P); %track time updated dovariance

KalmanGain = P \* H' \* 1/(Var); %Kalman Gain

Kalman\_Gain\_update(:,i)=KalmanGain; %Record kalman gain

X\_time\_mes(:,i+1) = X\_time\_update(:,i+1) + KalmanGain \* (z(i+1) - H \* X\_time\_update(:,i+1)); %Measurement Update

x(:,i+1)=X\_time\_mes(:,i+1); %Feedback of updated state

end

%% Plot the states Estimates and the Actual states

t=1:101;

figure (1)

plot(t,X\_time\_mes(1,:),'b');

hold on;

plot(t,x\_copy(1,:),'r')

hold on;

grid on;

legend('State 1 - Estimated','State 1 - Actual')

figure (2)

plot(t,X\_time\_mes(2,:),'r');

hold on

plot(t,x\_copy(2,:),'b');

grid on;

hold on;

legend('State 2 - Estimated','State 2 - Actual')

figure(3)

plot(t(1:100),Kalman\_Gain\_update(1,:),'r');

hold on;

grid on;

plot(t(1:100),Kalman\_Gain\_update(2,:),'g');

grid on;

hold on;

legend('Kalman Gain - State 1','Kalman Gain - State 2');

title(' Kalman Gain With Iteration')

[M,P,Z,E] = dlqe(A,G,H,Mean,Var);

M = 'The value of gain by Steady State Solution is'

M

%Simulation of system

X\_time\_update(:,i+1) = A \* X\_time\_mes(:,i) + B \* u(i); %Time Update state at i+1

X\_time\_mes(:,i+1) = X\_time\_update(:,i+1) + M \* (z(i+1) - H \* X\_time\_update(:,i+1)); %Measurement Update end

figure (4)

plot(t,X\_time\_mes(1,:),'r',t,x(1,:),'b');

grid on;

hold on;

legend('State 1 - Estimated','State 1 - Actual ')

title('Fixed Kalman Gain : State 1');

Figure(5)

plot(t,X\_time\_mes(2,:),'r');

grid on;

hold on;

plot(t,x(2,:),'b');

grid on;

hold on;

legend('State 2 - Estimated','State 2 - Actual ')

title('Fixed Kalman Gain : State 2')

clear P

P=1000\*eye(2);

for i=1:40 %some arbitrary number of Iterations to get the solution converged

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

P=P-P\*H'\*inv(H\*P\*H'+Var)\*H\*P;

end

KG\_Iteration=P\*H'\*inv(Var)

Kalman Gain =

0.8607

1.1552

ans =

The value of gain by Steady State Solution is

M =

0.8607

1.1552

Plot:









