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

**EE 5322 - 003**

**INTELLIGENT CONTROL SYSTEMS**

# HW # 3
# ASSIGNMENT

**by**

**SOUTRIK PRASAD MAITI**

**1001569883**

**Presented to**

**Dr. Frank Lewis**

**Sept 26, 2017**

# <u>EE 5322 Intelligent Control</u>

## Fall 2015

## 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: <u>Soutrik Maiti</u>

***Pledge of honor:***

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

e-Signature: <u>Soutrik Maiti</u>

# EE 5322 Homework 3

# Discrete Time Simulation & RLS

## Problem 1:

a)

MATLAB Code:

```
%State Matricies
a = [0,1;-0.9801,1.6];                          %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 State Variables

xlabel('Time');

ylabel('Amplitude');
legend('x1','x2');
figure
plot(ki,x(:,1))
xlabel('Time');
ylabel('Amplitude');
legend('x1');
figure
plot(ki,x(:,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
```

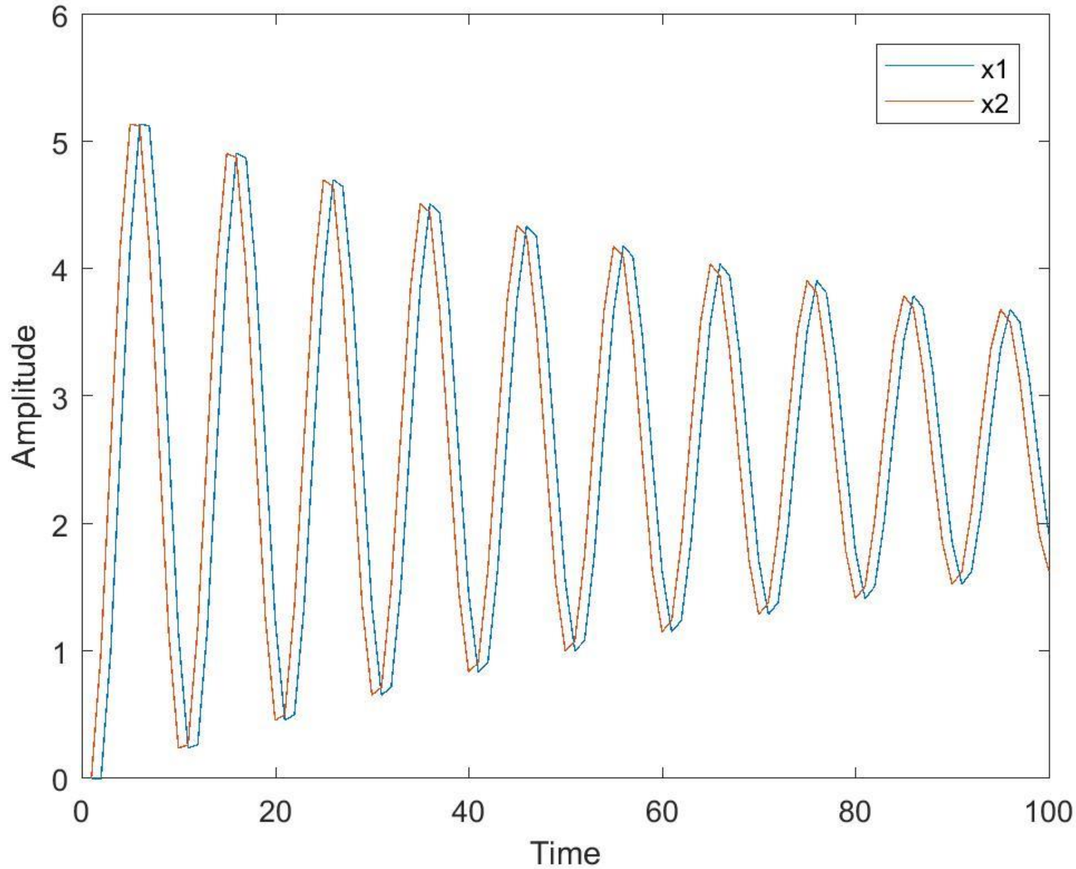


Fig 3.1 – Discrete Time simulation of the system

Period of the system – Fig 3.2 shows the coordinates of the first and second peak.
Hence the period is simply x2-x1+1 i.e. 15-6+1 which is 10.
*The period is 10*.
As of the peak overshoot is considered, we know :-

$$\boldsymbol{Peak\ Overshoot} = \frac{V_{Peak} - V_{Constant}}{V_{Constant}} \times \mathbf{100} = \mathbf{95.01\%}$$

Where
Vpeak = 5.127
Vconstant = 2.629 from Fig 3.3

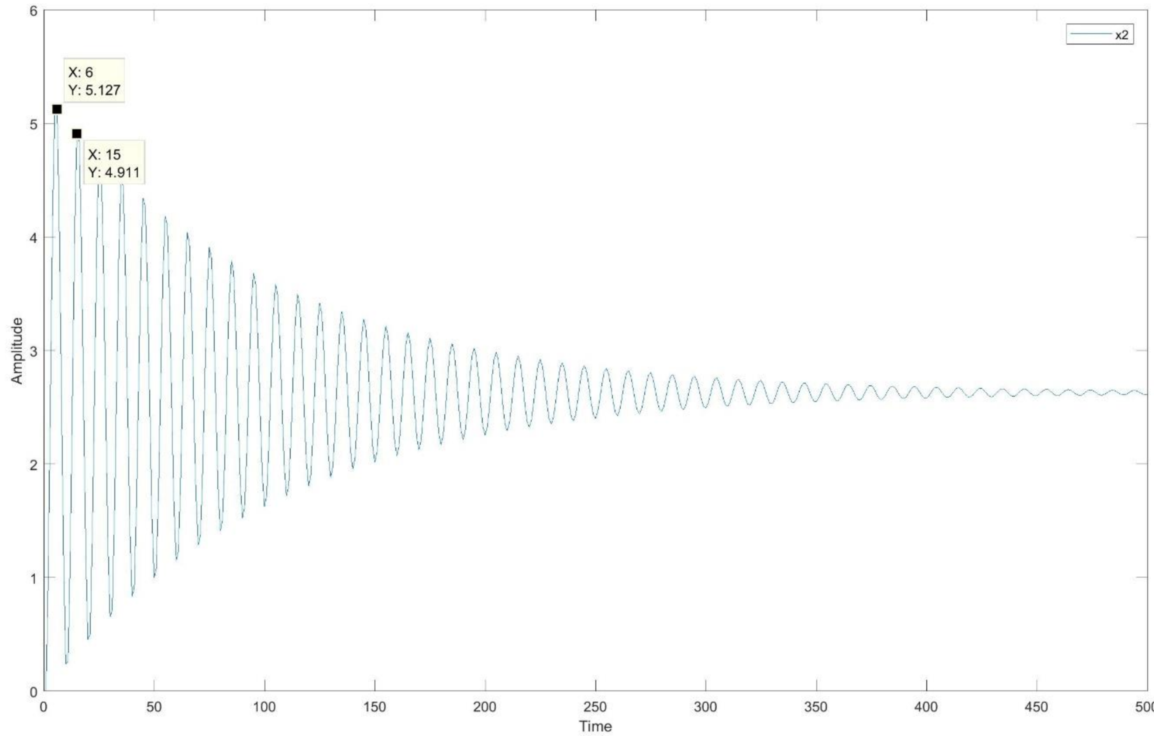


Fig 3.2 – Figure for the calculation of period

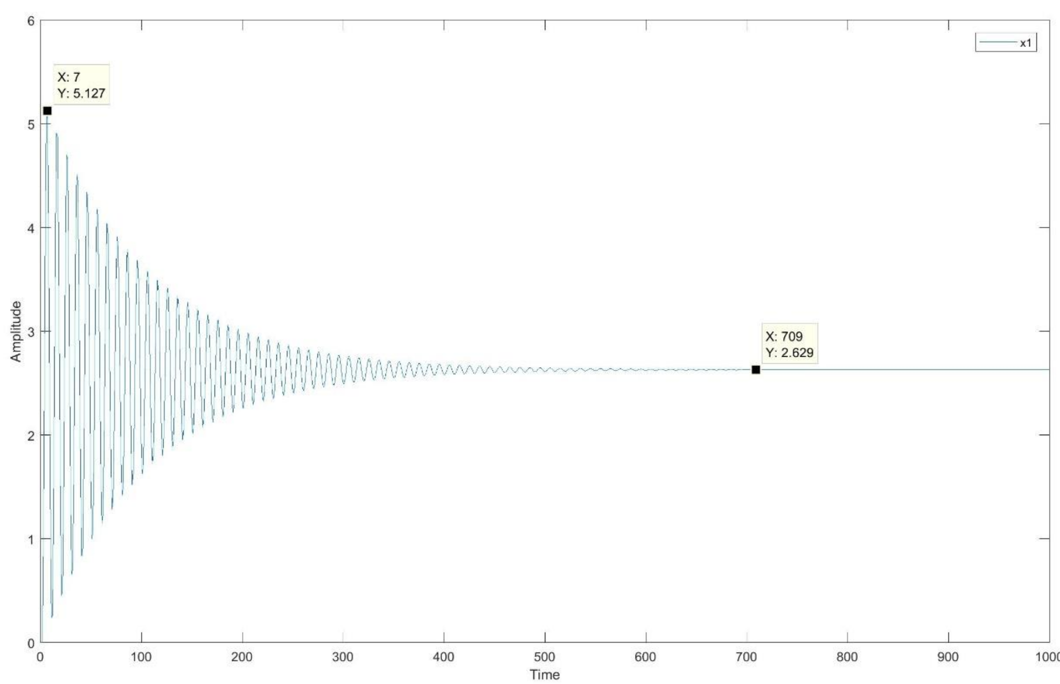


Fig 3.3 – Figure for the calculation of peak overshoot

b) Same system with process noise

```
%state matricies

a = [0,1;-0.9801,1.6];                          %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

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 = sqrt(0.1)*randn(N(1),2);                    %Process noise
with Covar=0.1

    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
```

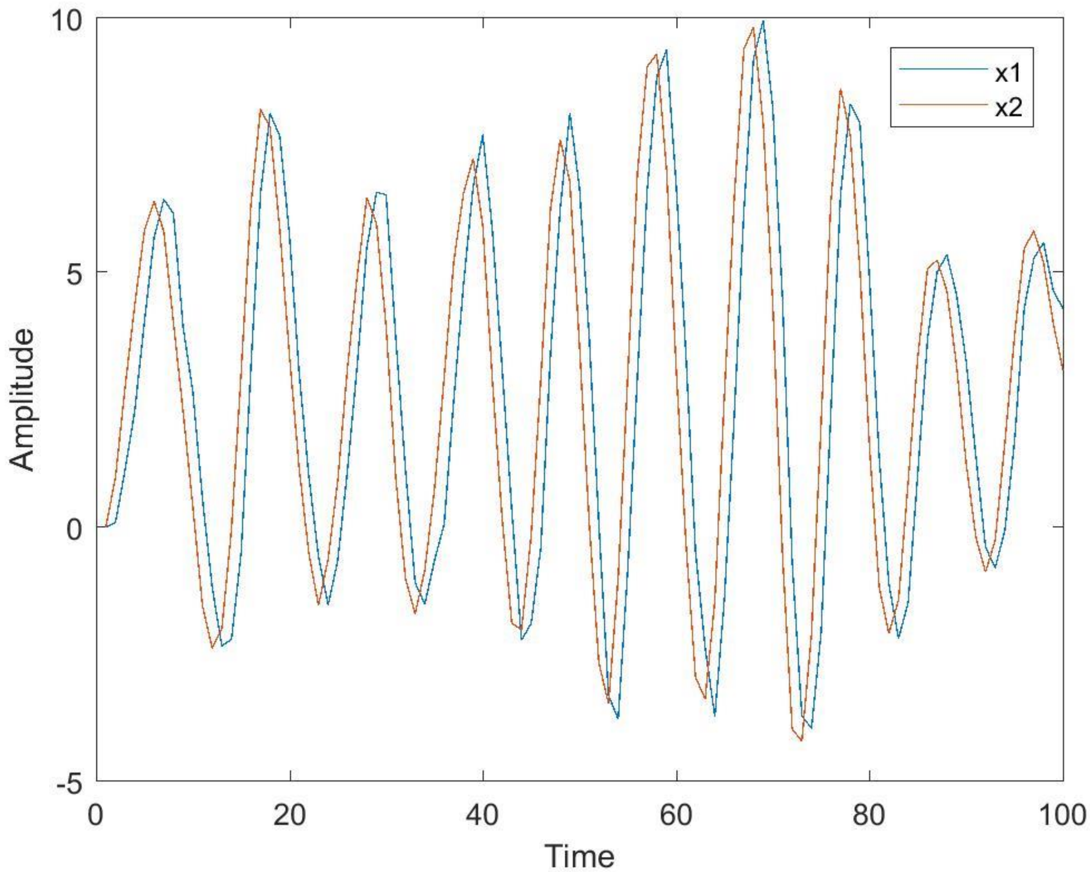


Fig 3.4 - Discrete Time Simulation of system with process noise of mean=0 & Covar =0.1

## Problem 2:

### MATLAB Code

```
clc;
uk=xlsread('hwk RLS ID data.xls','C4:C604');
yk=xlsread('hwk RLS ID data.xls','E4:E604');

d = 2;
n = 2;
ht = zeros(601,3);
%Calculation of Regression Matrix


for k = 3:601
    ht(k,:) = [-yk(k-1) -yk(k-2) uk(k-2)];
end

sigma = 0.1;
p1 = 1000*eye(3);
theta_cap = zeros(3,601);% Initial Estimates
%Calculation of Pk & Theta_Cap Using RLS Algorithm
```

```
for k = 2:600
    pk = p1 - p1*ht(k+1,:)'*(inv(ht(k+1,:)*p1*ht(k+1,:)'+
sigma))*ht(k+1,:)*p1;
    theta_cap(:,k+1) =
theta_cap(:,k)+(pk*(ht(k+1,:)'/sigma)*(yk(k+1,1)-
(ht(k+1,:)*theta_cap(:,k))));
p1 = pk;
end
ye = ht*theta_cap;
ye = diag(ye); %Y estimate
% Difference of estimate and given output (Error in Estimation)
diff = ye-yk;
figure
plot(diff)

%System Transfer function
theta_cap(:,601)
num=[0 0 theta_cap(3,601)];
den=[1 theta_cap(1,601) theta_cap(2,601)];
sys=filt(num,den)
%simulating the obtained transfer function using the input
out= lsim(sys,uk,1:601);
figure
plot (1:601,out)
hold on
plot (3:603,yk)
hold off
```

Output :-

```
ans =

  -1.9000
   0.9500
   0.2000


sys =

        0.2 z^-2
  -------------------------
  1 - 1.9 z^-1 + 0.95 z^-2
```

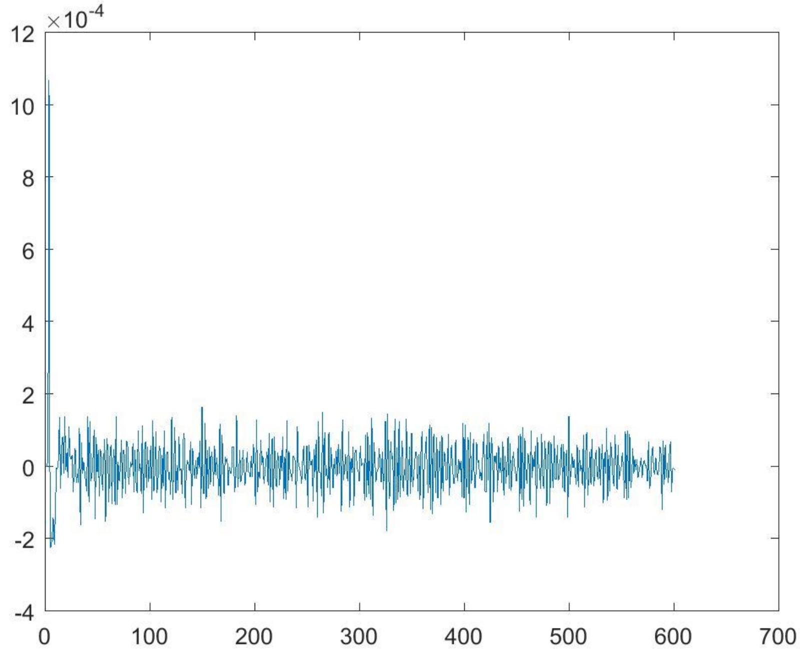


Fig 3.5 – The difference between y estimate and y actual

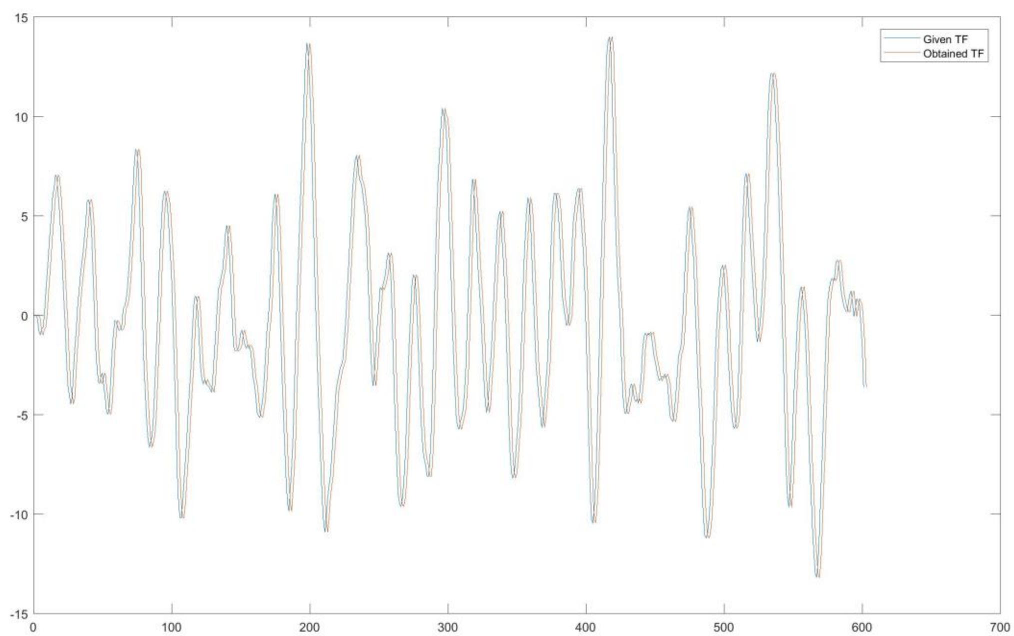


Fig 3.6 – The comparision of the transfer functions. The time has been shifted for the second transfer function to observe the similarity between the two.