```
% ELEC4632 lab 4 %
% prelab %
clear
close all
clc

load SysIdenData_4.mat
load SFControlData_0.mat

t = LogData.time;
y_act = LogData.signals(1).values(:,2);
y_actm = LogData.signals(1).values(:,1);
u_act = LogData.signals(2).values;

%truncate the first period, keep data afer 927 seconds
index = max(find(t<=927));
y_act = y_act(index:end);
y_actm = y_actm(index:end);
u_act = u_act(index:end);
t = t(1:length(u_act));
%Ts = (t(end)-t(1))/(length(t)-1); % find sampling time
Ts = t(2)-t(1);
fprintf('sampling time according to calculation is %d\n',Ts);
figure()
subplot(2,1,1)
hold on
plot(t,y_act,'b');

xlabel('Time (sec)');
ylabel('WaterLevel (V)');
title('Actual signal');
grid on
plot(t,y_actm,'r');
legend('Noised-Reduced Output','Measured Output');
hold off
subplot(2,1,2)
plot(t,u_act);
xlabel('Time (sec)');
ylabel('Pump voltage (V)');
title('Actual Input Signal');
legend('Actual input');
ylim([0 2.5]);
grid on
% remove input offset
u_offset = u_act(1);
u = u_act - u_offset;
figure()
subplot(2,1,2)
plot(t,u)
xlabel('Time (sec)');
ylabel('Pump Voltage (V)');
title('Actual Offset-Free Input Signal');
```

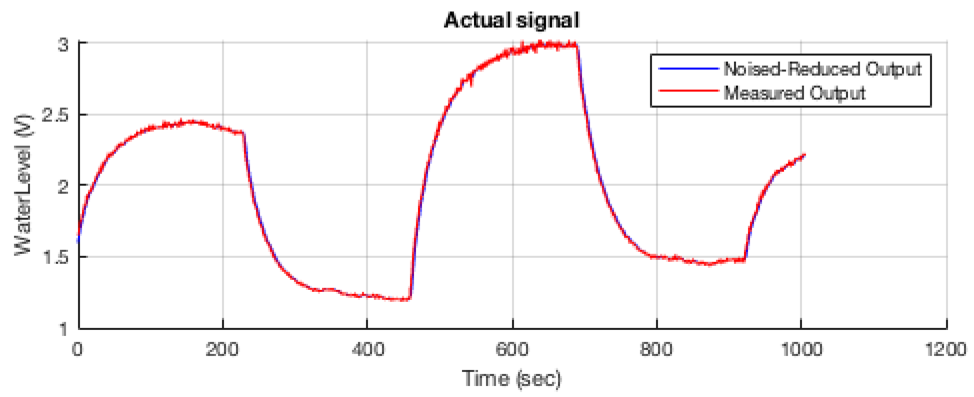
```
ylim([-0.5 0.5])
legend('Actual Input');
grid on
% remove output offset
count = 0;
i = 1;
while(u_act(i+1) == u_act(i))
    i=i+1;
    count = count + 1;
end
y_offset = mean(y_act(1:count));
y = y_act - y_offset;
subplot(2,1,1);
plot(t,y,'r')
grid on
xlabel('Time (sec)');
ylabel('Water Level (V)');
title('Actual Offset-Free Output Signal');
legend('Actual Output');

% start from k = 10, k should be greater than 2
k = 3;
[a1,a2,b1,b2] = second_order_regression(k,y,u);
H = tf([b1 b2],[1 a1 a2],Ts);
fprintf('Info about second order state space model is below:\n');
sys = ss(H)

%{
figure()
%simulate second half
subplot(2,1,1)
b = [b1 b2];
a = [1 a1 a2];
N = round(length(y)/2);
y_simulate_2nd_Half = filter(b,a,u(N:end));
plot(t(N:end),y_simulate_2nd_Half,'--');
hold on
plot(t(N:end),y(N:end),'r');
grid on
xlabel('Time (sec)');
ylabel('Water Level (V)');
legend('Simulated Output','Actual Output');
title('Offset-Free Model Verification (2^{nd} Half)');
hold off
%simulate entire
subplot(2,1,2)
y_simulate_entire_2nd_order = filter(b,a,u);
plot(t,y_simulate_entire_2nd_order,'--');
hold on
plot(t,y,'r')
grid on
xlabel('Time (sec)');
ylabel('Water Level (V)');
legend('Simulated Output','Actual Output');
title('Offset-Free Model Verification (Entire)');
%}
A = sys.A;
eigenVal = eig(A);
%check stability from eigen values
if(isempty(find(eigenVal>0))==0)
    fprintf('Eigenvalues are %f and %f\n',eigenVal);
end
[z,gain] = zero(sys);

G = [0 1; -a2 -a1;];
H = [0; 1;];
C = [b2 b1];
D = 0;

Wc = [H G*H];
Wo = [C; C*G];

if (rank(Wc) == 2)
    fprintf('Wc has full rank, reachable.\n')
else
    fprintf('Wc has no full rank, not reachable.\n')
end

if (rank(Wo) == 2)
    fprintf('Wo has full rank, observable, obeservability implies
 detectability\n\n')
else
    fprintf('Wo has no full rank, not observable,\n\n')
end

% q1 %
% Initialize
x1 = 0;
x2 = 0.3;
% From now on use canonical observer form
G_obsrv = G';
H_obsrv = C';
C_obsrv = H';
D_obsrv = 0 ;
Wc_obsrv = [H_obsrv G_obsrv*H_obsrv];
Wo_obsrv = [C_obsrv; C_obsrv*G_obsrv];

% Dead beat control
L_db = [0 1]*inv([(G_obsrv^-2)*H_obsrv (G_obsrv^-1)*H_obsrv]);
%L_db = [0 1]*inv(Wc_obsrv)*(G^2);

% non dead beat control
ndb_eigVal1 = 0.2348;
ndb_eigenVal2 = 0.9;
p_coeffi = poly([ndb_eigVal1 ndb_eigenVal2]);

% Desired characteristic equation
P = p_coeffi(1)*G_obsrv^2 + p_coeffi(2)*G_obsrv + p_coeffi(3)*eye(2);
```

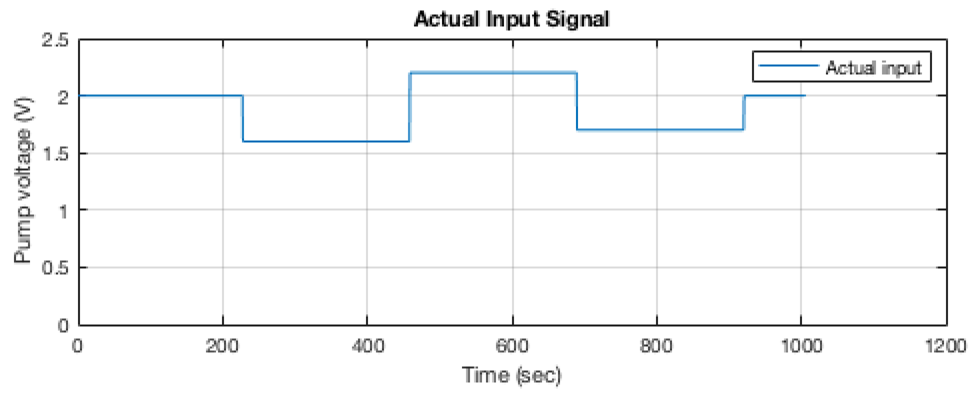
```
% Apply Ackermans's Formula
% L_ndb = [0 1]*inv(Wc_obsrv)*P
L_ndb = place(G_obsrv,H_obsrv,[ndb_eigVal1 ndb_eigenVal2]);

% Simulate y(k)
Tf = 50;
T = [0:Ts:50];
sys_ndb = ss((G_obsrv - H_obsrv*L_ndb),[],C_obsrv,D_obsrv,Ts);
sys_db = ss((G_obsrv - H_obsrv*L_db),[],C_obsrv,D_obsrv,Ts);
[y_ndb,t_ndb,x_ndb]= initial(sys_ndb,[x1 x2],Tf);
[y_db,t_db,x_db] = initial(sys_db,[x1 x2],Tf);

figure()
subplot(2,1,1)
stairs(T,y_ndb);
hold on
stairs(T,y_db);
ylim([-1 1]);
xlim([0 50]);
title('Regulation Response by State Feedback y(k)');
xlabel({'Time (sec)';'(a)'});
ylabel({'Offset-Free';'Water Level(V)'});
legend('None-Deadbeat Response','Deadbeat Response');
grid on
hold off

% simulate u(k)
u_ndb = -L_ndb*x_ndb';
u_db = -L_db*x_db';

subplot(2,1,2)
stairs(T,u_ndb)
hold on
stairs(T,u_db)
drawnow;
ylim([-1 1])
xlim([0 50])
title('Offset-Free Control Input u(k)')
xlabel({'Time (sec)';'(b)'});
ylabel({'Offset-Free';'Pump Voltage (V)'});
grid on
legend('None-Deadbeat Control Input','Deadbeat Control Input');
hold off
% q2 %
% Reference output
%y_ref = [zeros(1,140) 0.7*ones(1,140) -0.2*ones(1,140)
 0.5*ones(1,140) zeros(1,140)];
y_ref = SFLogData.signals(1).values(:,1)';
figure();
subplot(2,1,1)
plot([0:0.75:0.75*(length(y_ref)-1)],y_ref,'g');
grid on
ylim([-1 1]);
```

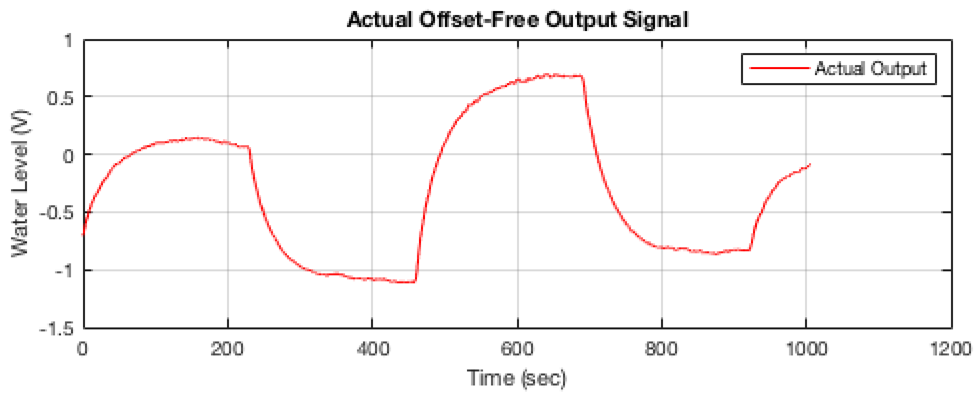
```
xlabel({'Time (sec)';'(a)'});
ylabel({'Offset-Free';'Water Level (V)'});
title({'Set-Point Control Results: Simulation';'Output Signal'});
hold on

% Desired characteristic equation
ndb_eigVal1 = 0.9;
ndb_eigenVal2 = 0.9;
p_coeffi = poly([ndb_eigVal1 ndb_eigenVal2]);
P = p_coeffi(1)*G_obsrv^2 + p_coeffi(2)*G_obsrv + p_coeffi(3)*eye(2);

% Apply Ackermans's Formula
L_ndb2 = [0 1]*inv(Wc_obsrv)*P;
%L_ndb2 = place(G_obsrv,H_obsrv,[ndb_eigVal1 ndb_eigenVal2]);

DC_gain = dcgain(ss(G_obsrv-H_obsrv*L_ndb2, H_obsrv,C_obsrv,
 D_obsrv,Ts));

sys_ndb2 = ss(G_obsrv - H_obsrv*L_ndb2,H_obsrv,C_obsrv,D_obsrv,Ts);
u_ndb2 = y_ref/DC_gain;
T = [0:Ts:Ts*(length(u_ndb2)-1)];
x1 = 0;
x2 = 0;
[y_spt,t_spt,x_spt] = lsim(sys_ndb2,u_ndb2,T,[x1 x2]);
plot(t_spt,y_spt,'r')
drawnow;
legend('Reference Output','Simulated Output')
subplot(2,1,2)

%Simulated Control Input
u_spt = -L_ndb2*x_spt' + y_ref/DC_gain;
plot(t_spt,u_spt)
grid on
xlabel({'Time (sec)';'(b)'});
ylabel({'Offset-Free';'Pump Voltage (V)'});
title('Control Input Sddignal')
legend('Simulated Control Input')

% q3 %
db_eigVal1 = 0.9;
db_eigenVal2 = 0.9;
p_coeffi = poly([db_eigVal1 db_eigenVal2]);
P =G^2; %deadbeat
%K = P*inv(Wo_obsrv)*[0 1]';
K = acker(G_obsrv',C_obsrv',[0; 0]);

figure();

subplot(3,1,1)
plot([0:0.75:0.75*(length(y_ref)-1)],y_ref,'g');
grid on
ylim([-1 1]);
xlabel({'Time (sec)';'(a)'});
ylabel({'Offset-Free';'Water Level (V)'});
```

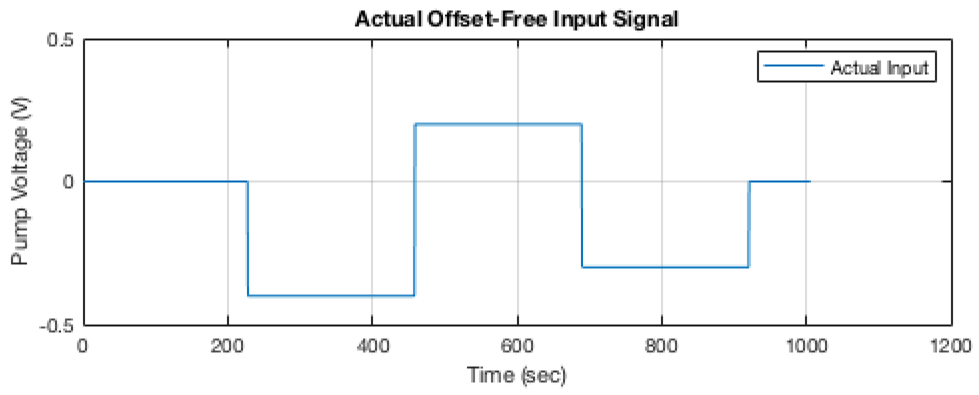
```
title({'Set-Point Control Results: Simulation';'Output Signal'});
hold on
plot(t_spt,y_spt,'r')
drawnow;
legend('Reference Output','Simulated Output')

subplot(3,1,2)
plot(t_spt,u_spt)
drawnow;
grid on
xlabel({'Time (sec)';'(b)'});
ylabel({'Offset-Free';'Pump Voltage (V)'});
title('Control Input Signal')
legend('Simulated Control Input')

subplot(3,1,3)
% use simulink to load data first
fprintf('Fetching result from simulink, might take a moment...\n');
sim('lab4.slx')
fprintf('Simulink data has been fetched.\n');
stairs(linspace(0,3,length(error.signals.values(:,2))),error.signals.values(:,1),'
hold on
stairs(linspace(0,3,length(error.signals.values(:,2))),error.signals.values(:,2),'
title('State Estimation Error');
ylabel('Estimation Error');
xlabel({'Time (sec)','(c)'});
grid on
% plot out the result %
load SFControlData_0.mat

treal = SFLogData.time;
yref = SFLogData.signals(1).values(:,1);
yreal = SFLogData.signals(1).values(:,2);
ureal = SFLogData.signals(2).values;

figure()
subplot(2,1,1)
plot(treal,yref,'g');
hold on
plot(treal,yreal,'r');
plot(t_spt,y_spt,'b');

grid on
xlabel({'Time (sec)';'(a)'});
ylabel({'Water Level (V)'});
title({'Output Feedback Control Results';'Output Signal'});
legend('Referece Output','Actual Output','Simulated Output')
subplot(2,1,2)
plot(treal,ureal,'g');
hold on
plot(treal,u_spt+2,'r')
xlabel({'Time (sec)';'(b)'});
ylabel({'Pump Voltage (V)'});
title('Control Input Signal')
```

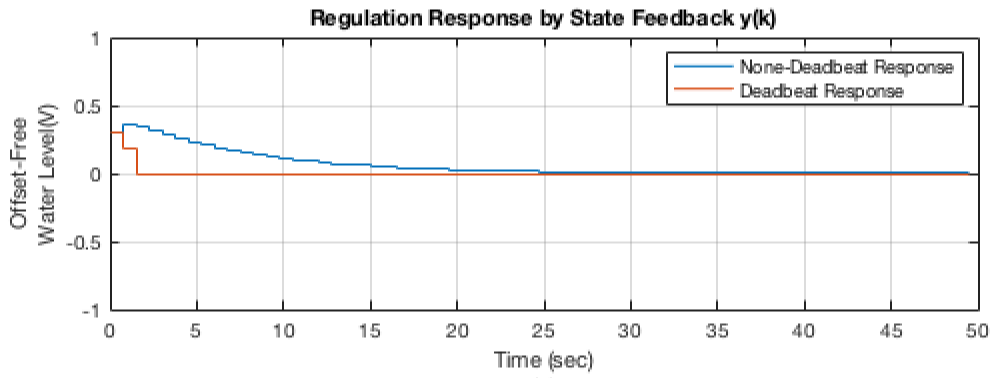
```
legend('Actual Control Input','Simulated Control Input');
grid on

sampling time according to calculation is 7.500000e-01
Info about second order state space model is below:

sys =

  A =
             x1       x2
   x1     1.258  -0.5494
   x2       0.5        0

  B =
          u1
   x1   0.25
   x2      0

  C =
             x1       x2
   y1   0.09166   0.1894

  D =
        u1
   y1    0

Sample time: 0.75 seconds
Discrete-time state-space model.

Eigenvalues are 0.977075 and 0.281148
Wc has full rank, reachable.
Wo has full rank, observable, obeservability implies detectability

Fetching result from simulink, might take a moment...
Simulink data has been fetched.
```


Actual signal
Noised-Reduced Output
Measured Output
WaterLevel (V)
Time (sec)


Actual Input Signal
Actual input
Pump voltage (V)
Time (sec)

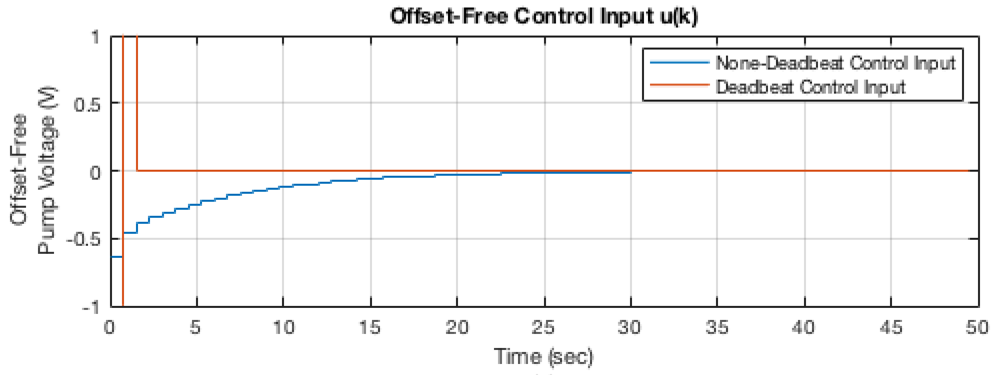

Actual Offset-Free Output Signal
Actual Output
Water Level (V)
Time (sec)


Actual Offset-Free Input Signal
Actual Input
Pump Voltage (V)
Time (sec)



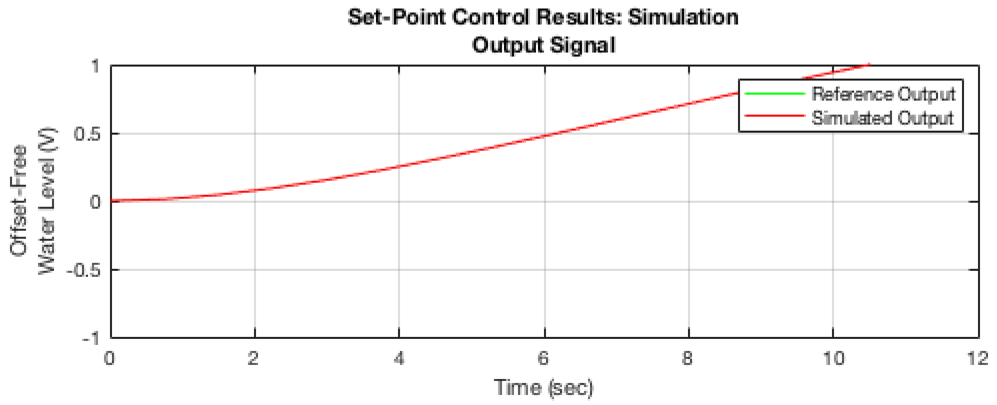
Regulation Response by State Feedback y(k)
Offset-Free
Water Level(V)
Time (sec)
None-Deadbeat Response
Deadbeat Response

(a)



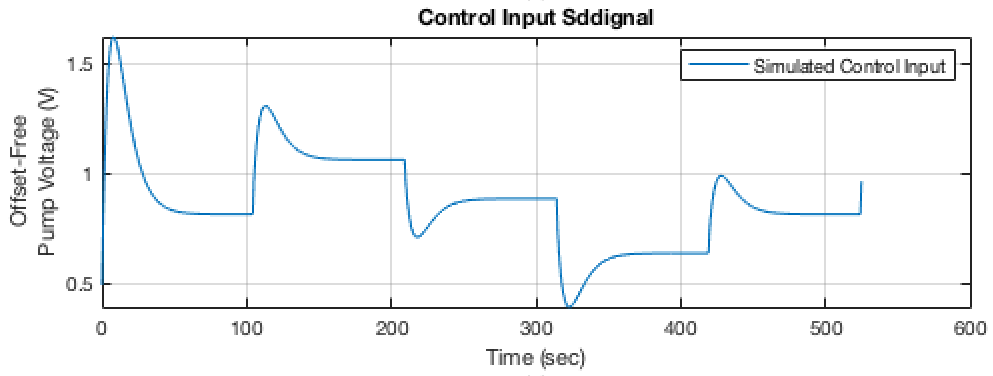
Offset-Free Control Input u(k)
Offset-Free
Pump Voltage (V)
Time (sec)
None-Deadbeat Control Input
Deadbeat Control Input

(b)



Set-Point Control Results: Simulation
Output Signal
Offset-Free
Water Level (V)
Time (sec)
Reference Output
Simulated Output

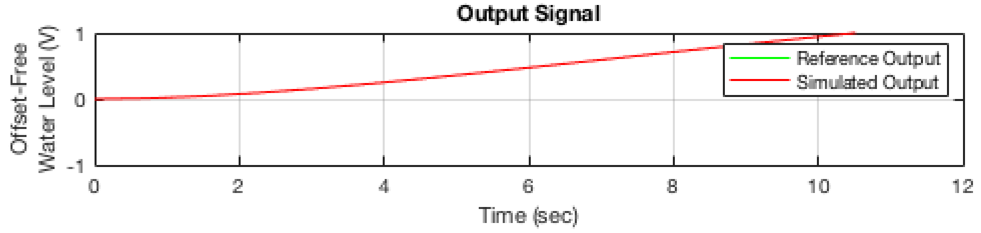
(a)



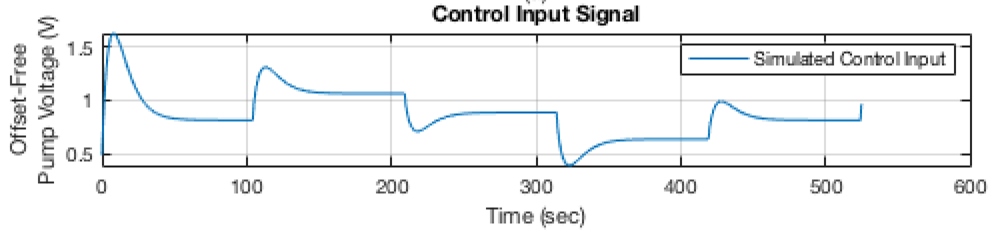
Control Input Sddignal
Offset-Free
Pump Voltage (V)
Time (sec)
Simulated Control Input

(b)

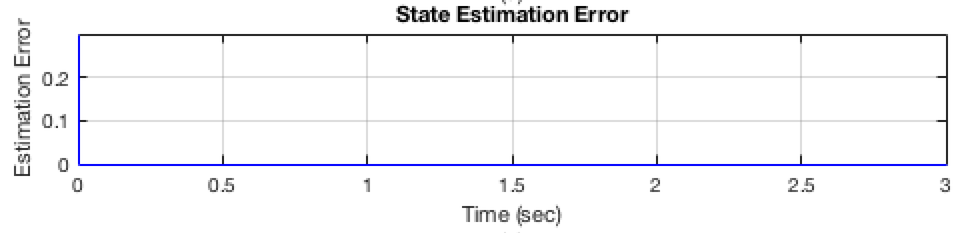
Set-Point Control Results: Simulation



(a)

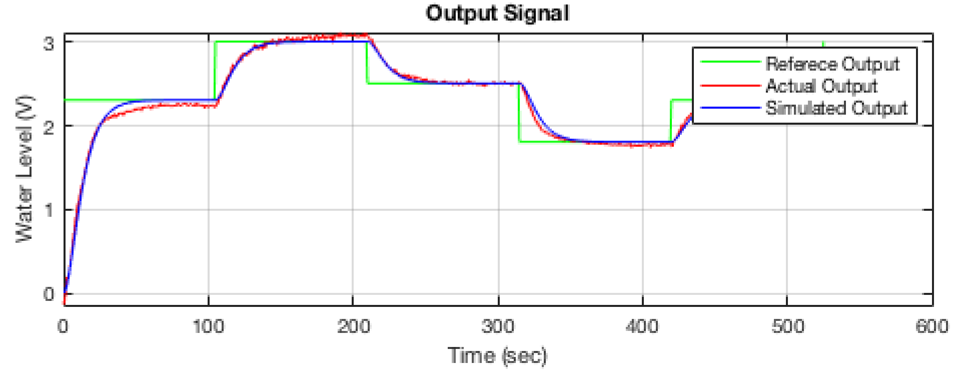


(b)

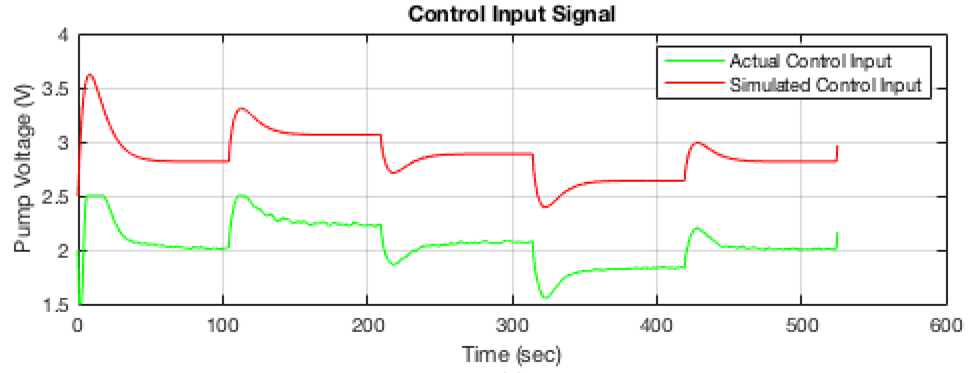


(c)

Output Feedback Control Results



(a)



(b)

Published with MATLAB® R2018a