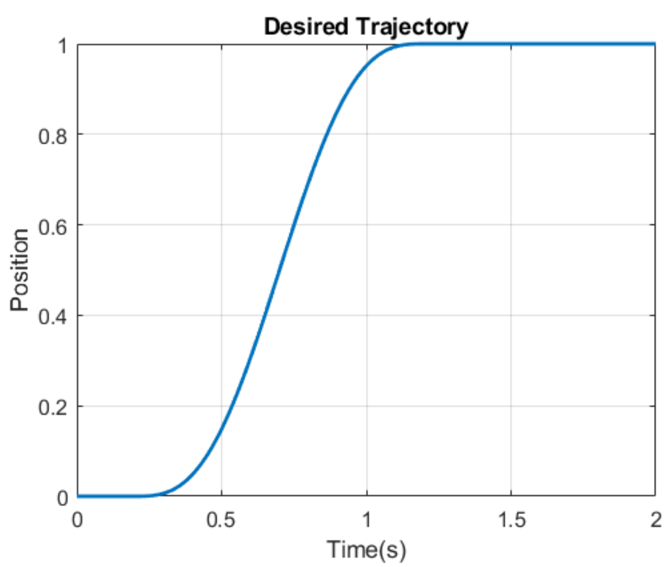
Contents


```
close all; clear all; clc
%%%%%%% Comments %%%%%%%
% Places that need manual modification will have the following
% Load vehicle_steering_YK.slx in the same path

%%%%%%% Modify %%%%%%%
%
% Instructions are provided
%
%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%% Final Project: inv YK Parameterization and disturbance rejection %%%%%%%%%%%%%%%%%%%%%%
```

Desired velocity profile

```
fs = 160; % sampling freuency, 10x times the max disturbance
Ts = 1/fs;
wth = 2; % linewidth for plotting

nfig = 0;
T = 1; % T is the time period in second
Npre = 0.2;  Npost = 0.8; % pre and post actuaction time
Tend = T+Npre+Npost; % Ending point for plotting
y_final = 1;
[yd,ydd,yd2d,t,nfig] =  sinusodial_yacc(Npre,(Npre+T),(Npre+T+Npost), 0, y_final, nfig,Tend);
```



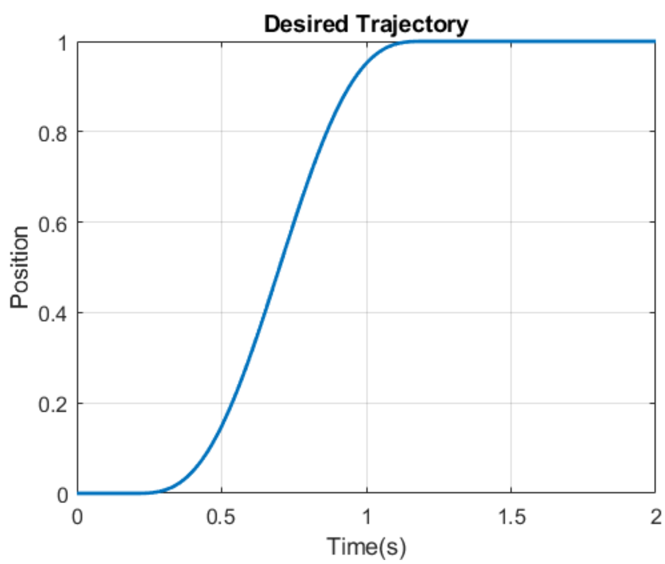
Vehicle Parameters

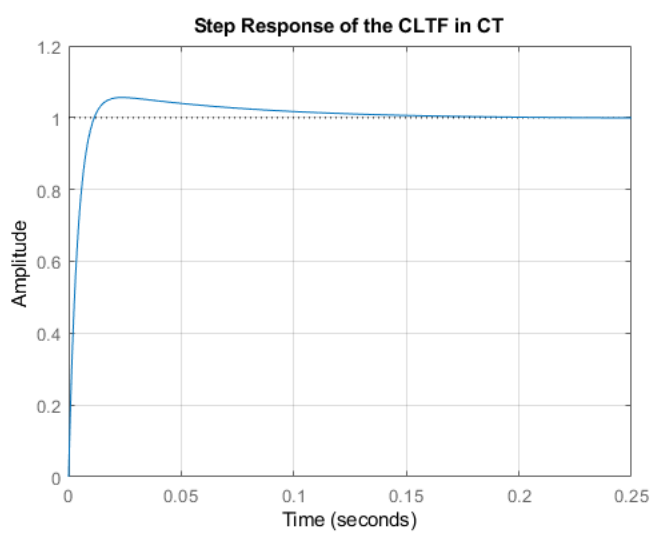
```
a = 2.5; % length from rear wheels to center of mass
b = 5; % length of car
v_0 = 30; % m/s

% Open Loop system, linearized
A = [0 v_0; 0 0];
B = [a/b*v_0 v_0/b]';
C = [1 0; 0 1];
D = [0; 0];
ctrb(A,B);
s = tf('s');
[num, den] = ss2tf(A,B,C,D);

% Defining state names
states = {'theta' 'Y'};
inputs = {'delta'};
outputs = {'Y';'Theta'};
```

```
sys = ss(A,B,C,D,'statename',states,'inputname',inputs,'outputname',outputs);
sys_y = sys(1,1); % sigma to y
sys_th = sys(2,1); % sigma to theta
Kcr = 1; % ideal gain
[Ay By Cy Dy] = ssdata(sys_y); % Extract state space
[num den] = ss2tf(Ay,By,Cy,Dy);
Gy_ol = tf(num,den);
P = Gy_ol; % open loop plant
```

PI controller (using Ziegler Nichols tuning)

```
L = 0.05;
T = 0.95;
Kp = 0.9*T/L;
Ti = L/0.3;
Td = 0;
C_ct = Kp*(1+1/(Ti*s)+Td*s); % PI controller in CT
Gy_cl = feedback(C_ct*Gy_ol,1); % CLTF system with PI controller
Gy_cl = minreal(Gy_cl); % Remove redundancy
% check stability
figure()
step(Gy_cl)
grid
title('Step Response of the CLTF in CT')
```

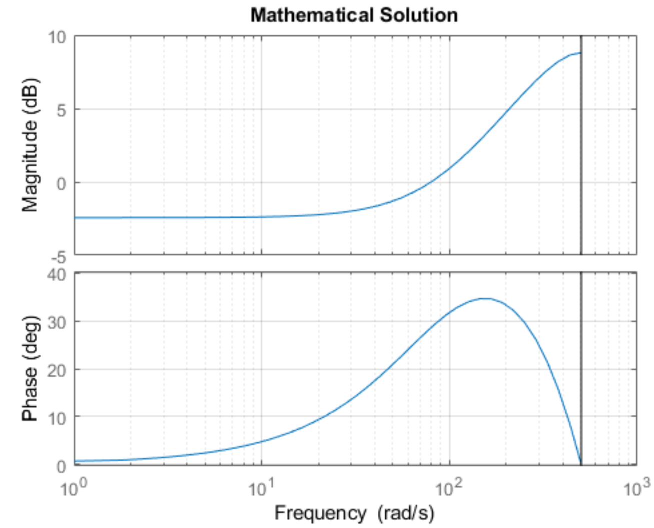


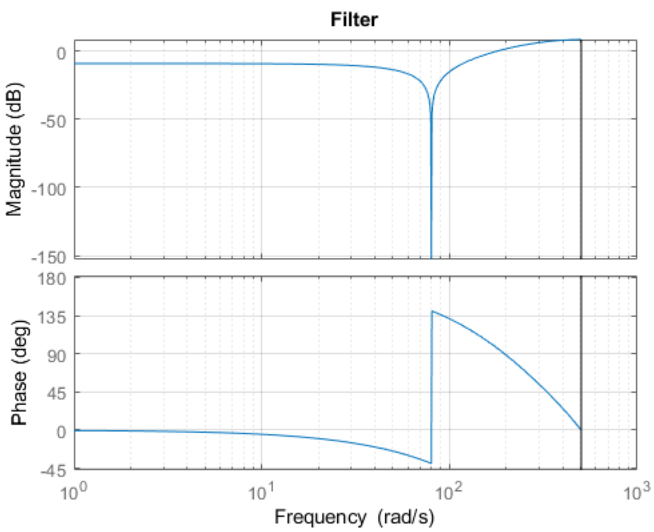


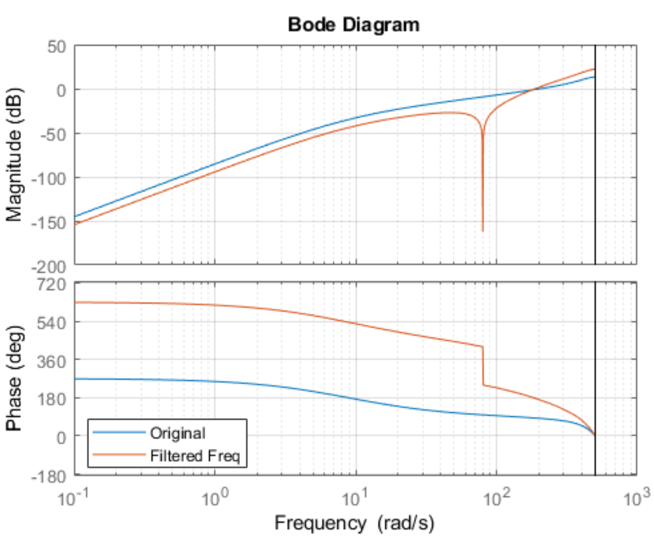
System in DT with disturbance filter

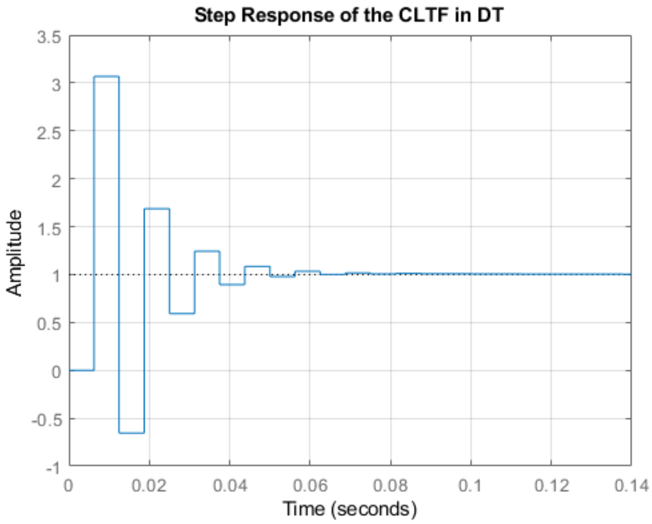
```
%%%%%%% Modify %%%%%%%
% Uncomment one of the lines to save the MATLAB plots
% save = 1; % save == yes
save = 0; % save == no
%%%%%%%%%%%%%%%%%%%%%%
```

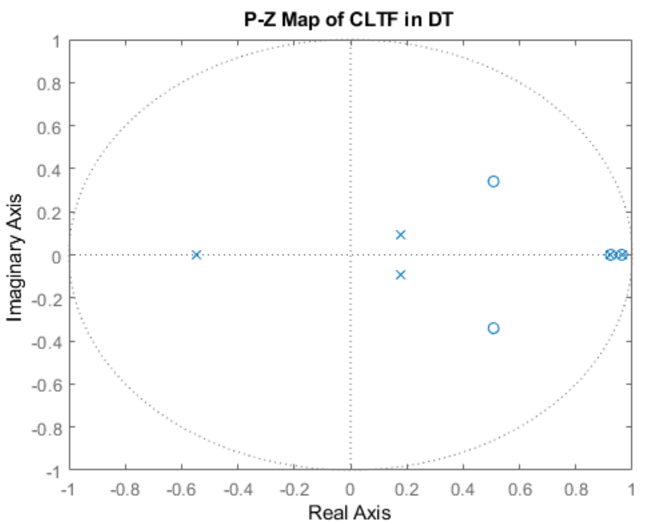
```
for i_a = 1:3
    param_a = i_a;
if param_a == 1
    alpha = 0.2;
elseif param_a == 2
    alpha = 0.99;
else
    alpha = 0.6;
end

z = tf('z',Ts);
m = 1; % relative degree
Zm = z^-m;
P_dt = c2d(P,Ts,'ZOH'); % convert plant to DT
invP = inv(P_dt)*Zm; % invert plant to DT and make strictly proper
wn = 80; % filtered out frequency
w_dt = wn*Ts;

% parameterized Q
% mathematical solution
Q_math = 2*cos(w_dt)-z^-1;
figure()
bode(Q_math)
grid
title('Mathematical Solution')

Q = minreal(((2-2*alpha)*cos(w_dt)+(alpha^2-1)*z^-1)/(1-2*alpha*cos(w_dt)*z^-1+alpha^2*z^-2));
A = 1-Zm*Q; % filtered out frequency
figure()
bode(A)
title('Filter')
grid on
% Sensivity function
C_dt = c2d(C_ct,Ts);% PI controller in DT
S_dt = minreal(1/(1+P_dt*C_dt)); % original sensitivity
figure()
bode(S_dt)
hold on
S_dt_q = minreal(1/(1+P_dt*C_dt)*(1-z^-m*Q));
bode(S_dt_q) % parameterized sensitivyt function
legend('Original','Filtered Freq','location','best')
grid on
hold off

if save == 1 && param_a == 1
    title('Sensitivity Function, alpha = 0.2')
    saveas(gcf,'Sensitvity_S_alpha0.2.png')
elseif save == 1 && param_a == 2
    title('Sensitivity Function, alpha = 0.99')
    saveas(gcf,'Sensitvity_S_alpha0.99.png')
elseif save == 1 && param_a == 3
    title('Sensitivity Function, alpha = 0.6')
    saveas(gcf,'Sensitvity_S_alpha0.6.png')
else
    save = 0;
end

% equivalent C_all for the parameterized controller
C_all = minreal((C_dt+invP*Q)/(1-Zm*Q));
G_cl = feedback(P_dt*C_all,1);
G_cl = minreal(G_cl);

% check DT is stable
figure()
step(G_cl)
grid
title('Step Response of the CLTF in DT')
figure()
pzmap(G_cl)
title('P-Z Map of CLTF in DT')
end
```

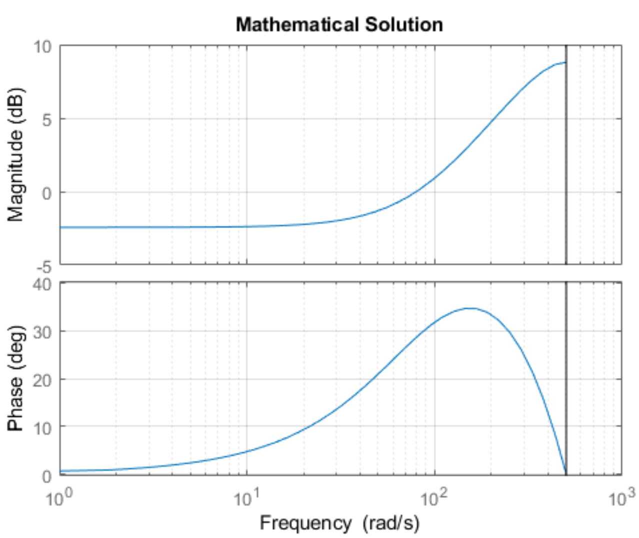

Mathematical Solution
Magnitude (dB)
10
5
0
-5
Phase (deg)
40
30
20
10
0
10^0
10^1
10^2
10^3
Frequency (rad/s)

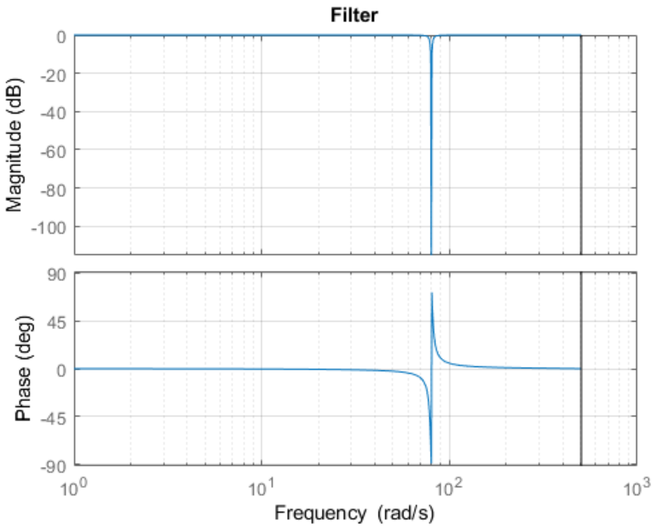

Filter
Magnitude (dB)
0
-50
-100
-150
Phase (deg)
180
135
90
45
0
-45
10^0
10^1
10^2
10^3
Frequency (rad/s)

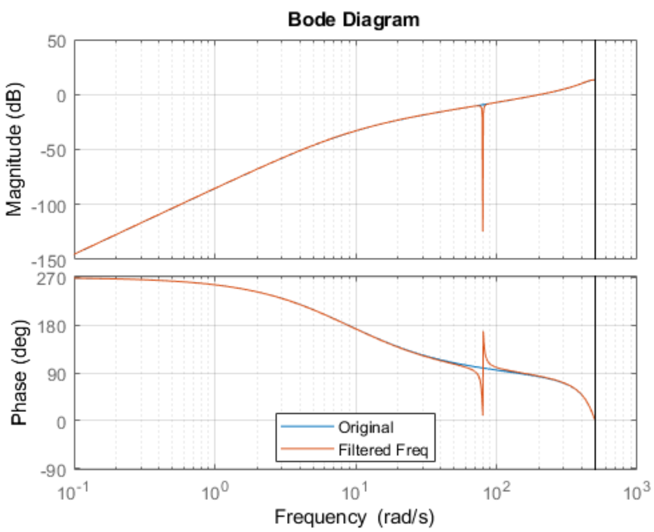

Bode Diagram
Magnitude (dB)
50
0
-50
-100
-150
-200
Phase (deg)
720
540
360
180
0
-180
Original
Filtered Freq
10^{-1}
10^0
10^1
10^2
10^3
Frequency (rad/s)

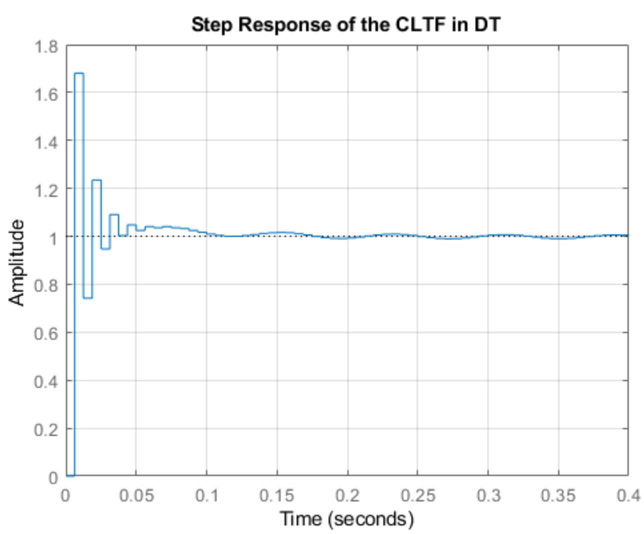

Step Response of the CLTF in DT
Amplitude
Time (seconds)

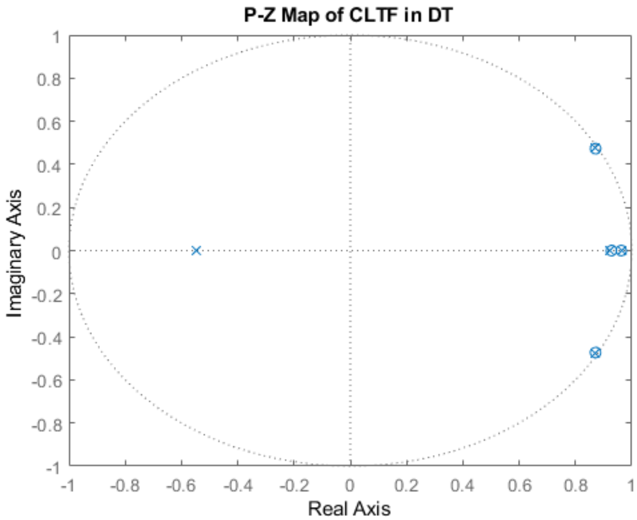

P-Z Map of CLTF in DT
Imaginary Axis
Real Axis

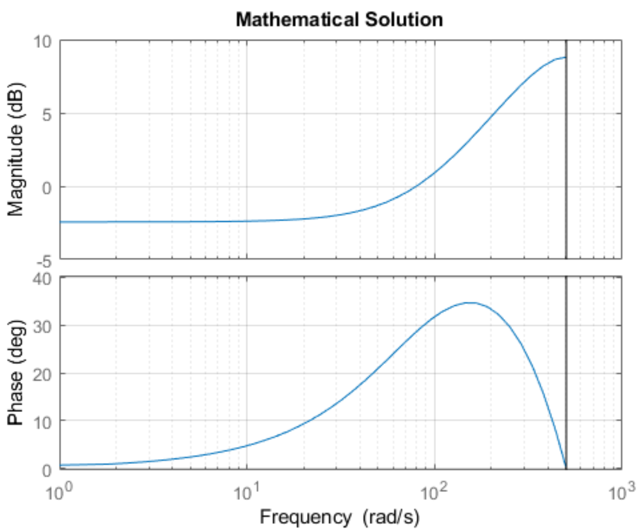

Mathematical Solution
Magnitude (dB)
Phase (deg)
Frequency (rad/s)

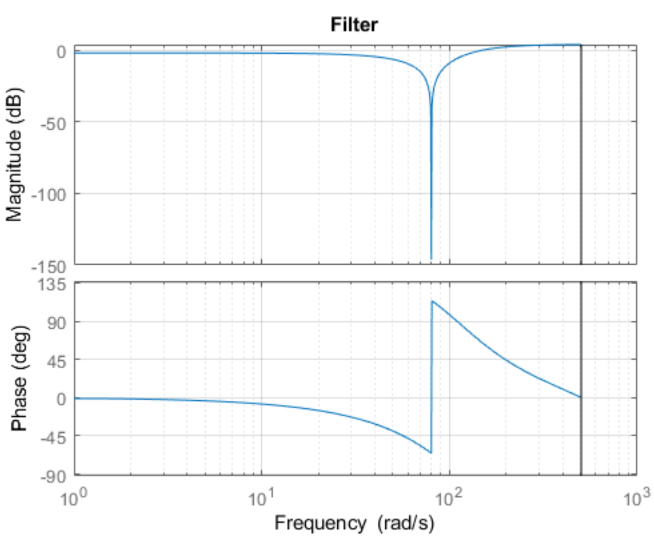

Filter
Magnitude (dB)
0
-20
-40
-60
-80
-100
Phase (deg)
90
45
0
-45
-90
10^0
10^1
10^2
10^3
Frequency (rad/s)

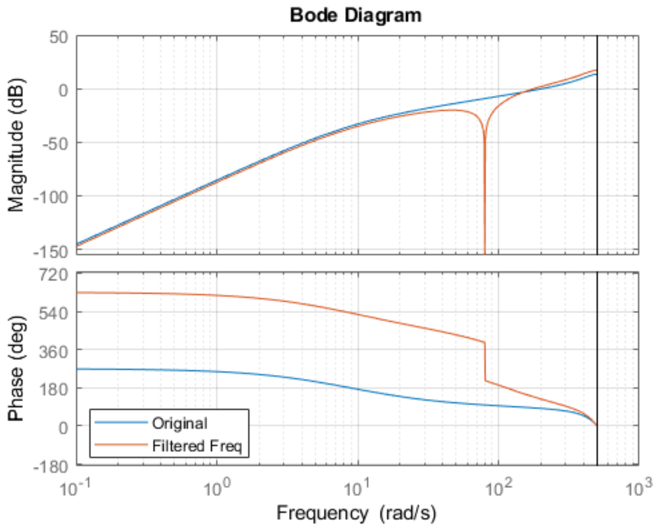

Bode Diagram
Magnitude (dB)
50
0
-50
-100
-150
Phase (deg)
270
180
90
0
-90
Original
Filtered Freq
10^{-1}
10^0
10^1
10^2
10^3
Frequency (rad/s)

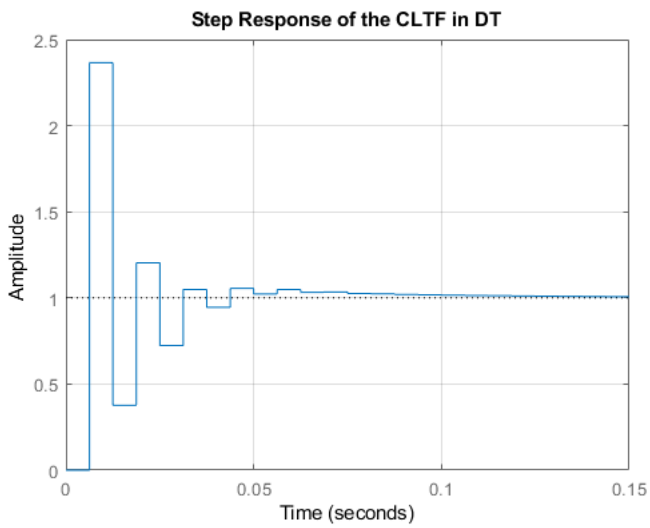

Step Response of the CLTF in DT
Amplitude
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
Time (seconds)

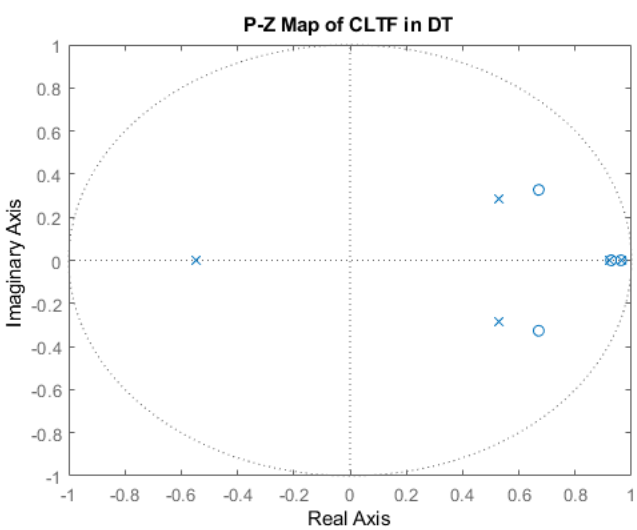

P-Z Map of CLTF in DT
Imaginary Axis
Real Axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Mathematical Solution
Magnitude (dB)
10
5
0
-5
Phase (deg)
40
30
20
10
0
10^0
10^1
10^2
10^3
Frequency (rad/s)


Filter
Magnitude (dB)
0
-50
-100
-150
Phase (deg)
135
90
45
0
-45
-90
10^0
10^1
10^2
10^3
Frequency (rad/s)





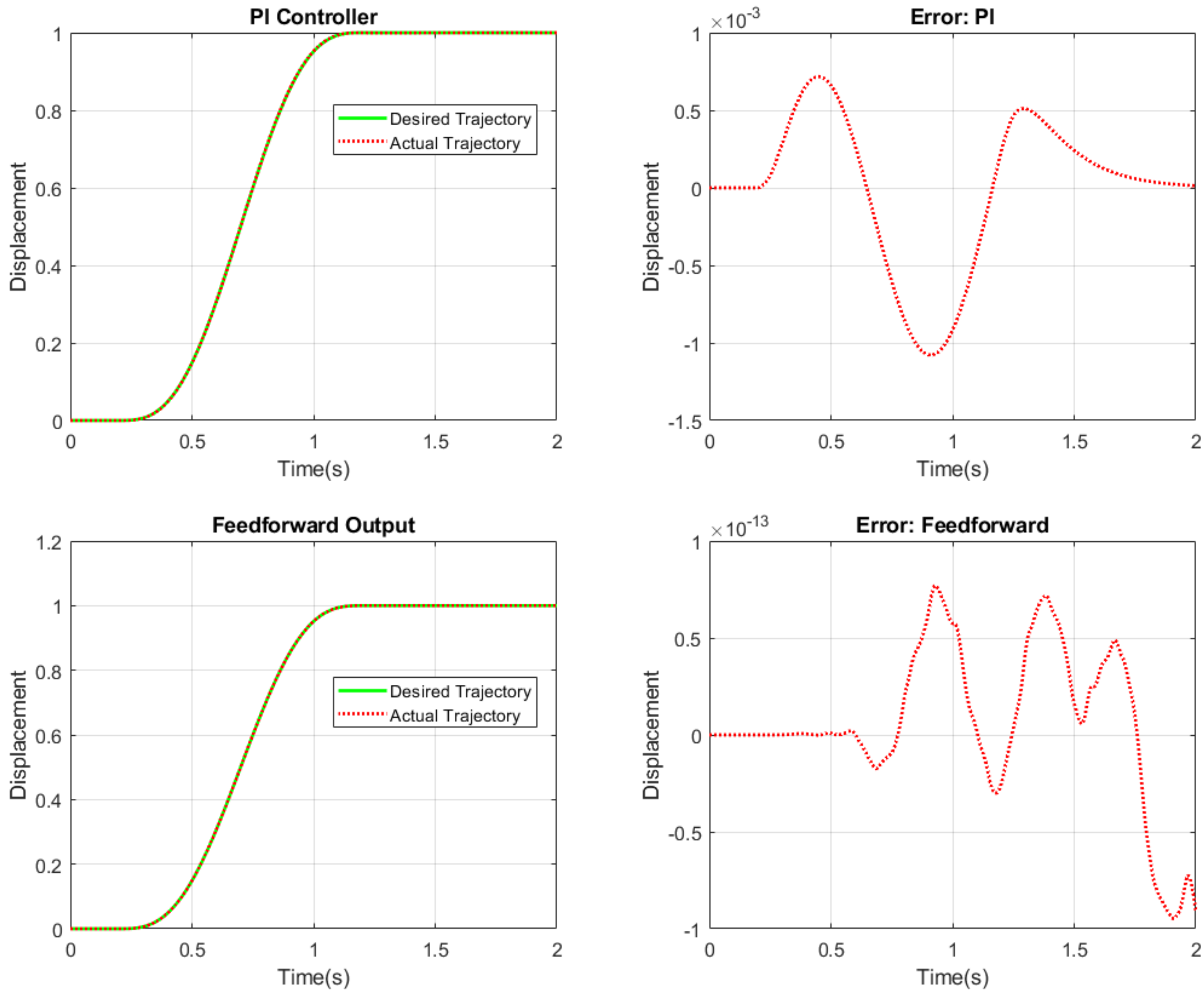


MATLAB simulation: PI controller only

input to the controller: desired trajectory

```
%%%%%%% Modify %%%%%%%
% Uncomment one of the lines to save the MATLAB plots
% save = 1; % save == yes
save = 0; % save == no
```

```
%%%%%%%%%%%%%%%%%%%%%%

y_out1 = lsim(G_cl,yd,t);
y_out1 = y_out1';
% plotting
figure('Position', [100, 100, 1200, 400]);
subplot(1,2,1)
plot(t,yd,'g',t,y_out1,'r:','linewidth',wth) % output y
xlim([0 Tend])
xlabel('Time(s)'),ylabel('Displacement')
title('PI Controller')
legend('Desired Trajectory','Actual Trajectory','location','best')
set(gca,'Fontsize',12), grid
error1 = yd-y_out1; % error
subplot(1,2,2)
plot(t,error1,'r:','linewidth',wth)
xlim([0 Tend])
title('Error: PI')
xlabel('Time(s)'),ylabel('Displacement')
set(gca,'Fontsize',12), grid

% save file
if save == 1
    saveas(gcf,'MATLAB_PI.png')
else
    save = 0;
end

% Feedforward with PI controller: Inversion methd
invG = inv(G_cl/Zm); % inverse of the CL system with delay z^-1
yd_1k = yd(1,(1+m):end); % shift y(k+1) to reduce phase lag
end_k = length(yd_1k);
for i = 1:m % match the size of vector, holding final value
    yd_1k(1,(end_k+i)) = y_final;
end
uinv = lsim(invG,yd_1k,t); % U_inv for the closed loop system
uinv = uinv';
y_out2 = lsim(G_cl,uinv,t); % output y
y_out2 = y_out2';
% plotting y
figure('Position', [100, 100, 1200, 400]);
subplot(1,2,1)
plot(t,yd,'g',t,y_out2,'r:','linewidth',wth) % output y plotting
xlim([0 Tend])
legend('Desired Trajectory','Actual Trajectory','location','best')
title('Feedforward Output')
xlabel('Time(s)'),ylabel('Displacement')
set(gca,'Fontsize',12), grid
%plotting error
subplot(1,2,2)
error2 = yd-y_out2;
plot(t,error2,'r:','linewidth',wth)
xlim([0 Tend])
title('Error: Feedforward')
xlabel('Time(s)'),ylabel('Displacement')
set(gca,'Fontsize',12), grid

% save file
if save == 1
    saveas(gcf,'MATLAB_FF.png')
else
    save = 0;
end
```



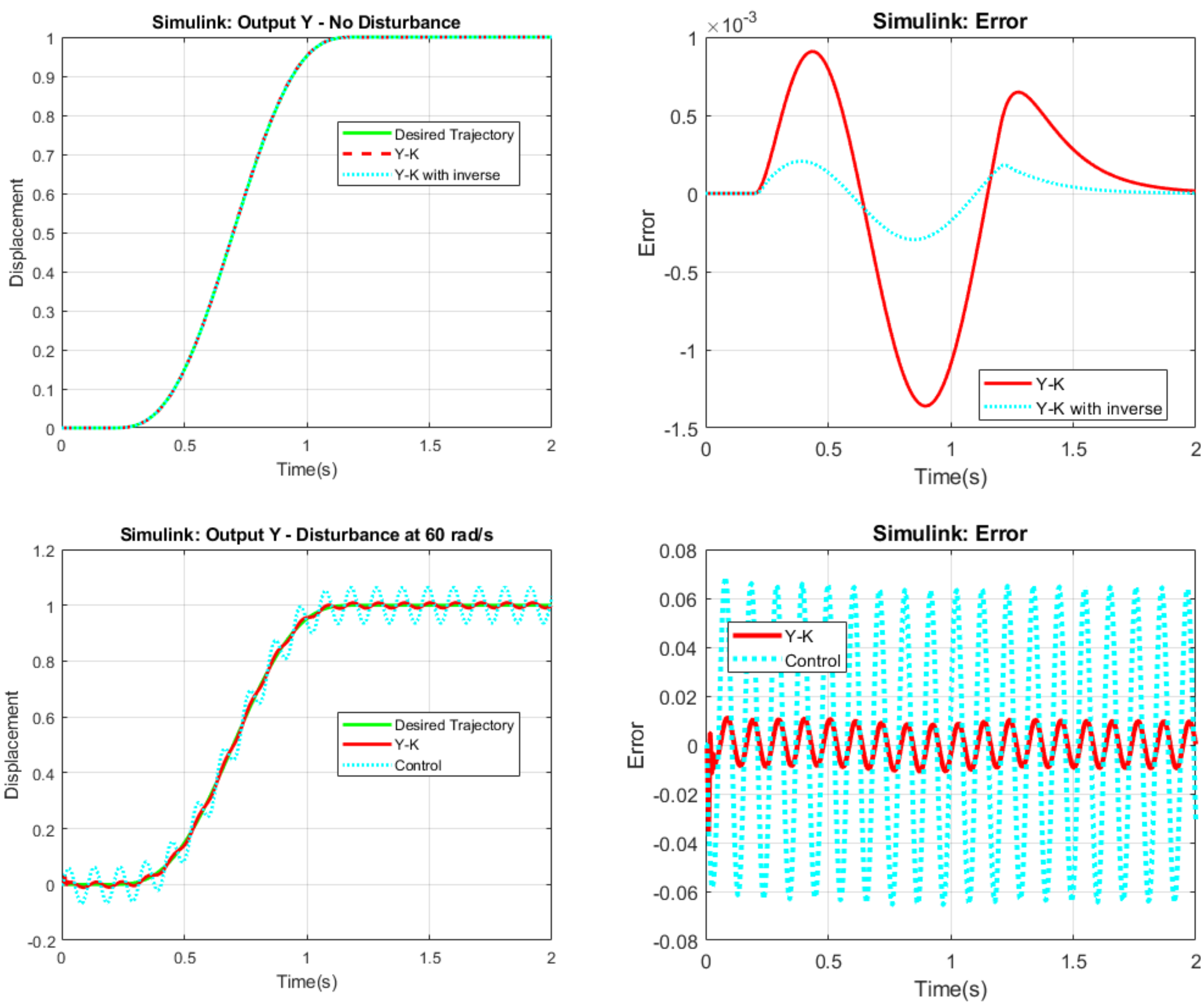
Simulink

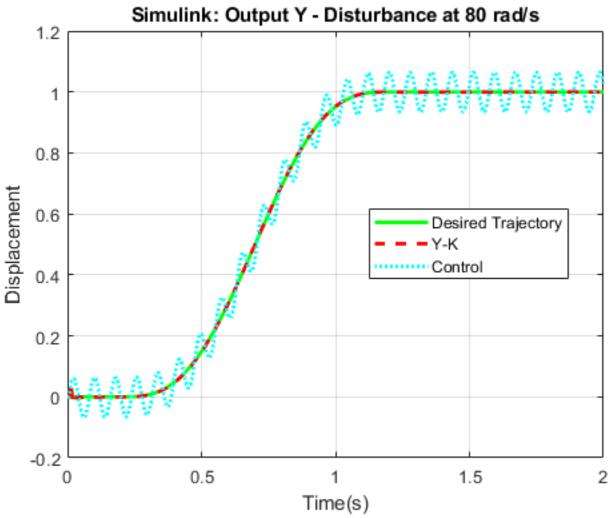
Export input data for Simulink

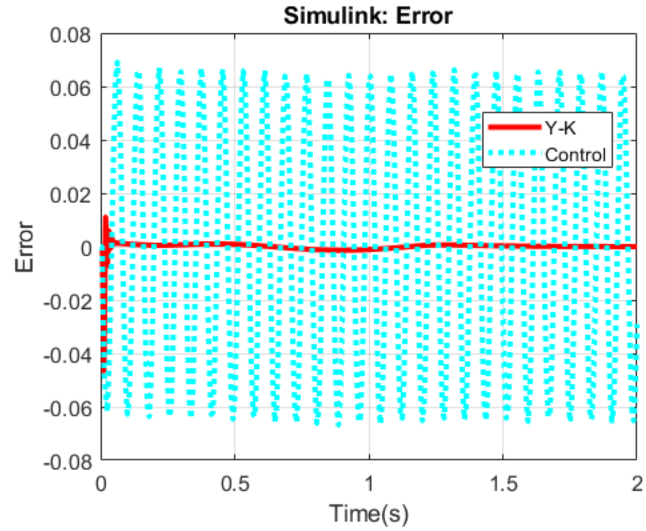
```
ydes.signals.values = [yd'];
ydes.time = [t']; % y(k+1) equivalent
ydes.signals.dimensions = 1;
ydes1k.time = [t']; % y(k+1) equivalent
ydes1k.signals.values = [yd_1k'];
ydes1k.signals.dimensions = 1;
M_u = 1; % input magntidue

%%%%%%% Modify %%%%%%%
% Uncomment one of the lines to save the SIMULINK plots
% save = 1; % save == yes
save = 0; % save == no
%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Modify %%%%%%%
% Choose one of the alpha values
alpha = 0.2;
% alpha = 0.6;
% alpha = 0.99;
%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Notes %%%%%%%%
% Parmaeters to test the disturbance at that frequency
% param = 1; % no disturbance
% param = 2; % disturbance at 60 rad/s
% param = 3; % disturbance at 80 rad/s
% param = 4; % disturbance at 100 rad/s
%%%%%%%%%%%%%%%%%%%%%%

Q = minreal(((2-2*alpha)*cos(w_dt)+(alpha^2-1)*z^-1)/(1-2*alpha*cos(w_dt)*z^-1+alpha^2*z^-2));
% equivalent C_all for the parameterized controller

for i = 1:4
    param = i;
```

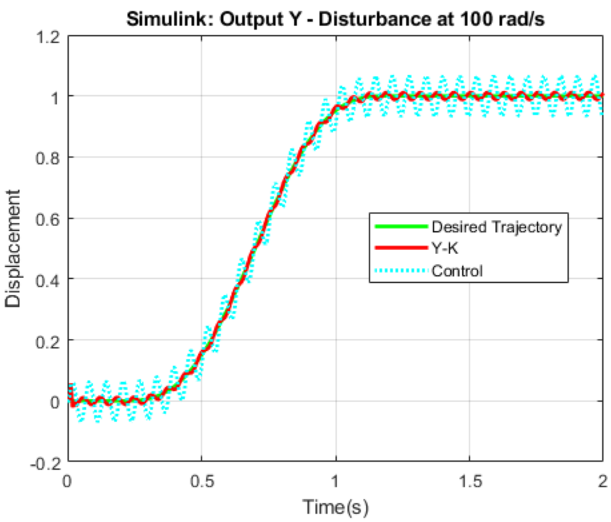
```matlab
% initiates the case to run
if param == 1
   M_dist = 0;
   w_dist = 0;
elseif param == 2
   M_dist = 1;
   w_dist = 60;
elseif param == 3
   M_dist = 1;
   w_dist = 80;
else
   M_dist = 1;
   w_dist = 100;
end

% Initiate simulation
% simout is the output that simulink returned
simout=sim('vehicle_steering_YK','StopTime','Tend', ...
          'SaveTime','on','TimeSaveName','timeout', ...
          'SaveOutput','on','OutputSaveName','yout');
t_sim = simout.timeout'; % time used in Simulink
y_sim1 = simout.yout{1}.Values.Data'; % parameterized Q system with no FF
y_sim2 = simout.yout{2}.Values.Data'; % parameterized Q with FF inverse
y_sim3 = simout.yout{3}.Values.Data'; % Control, controller + plant
e_sim1 = simout.yout{4}.Values.Data'; % error of parameterized Q
e_sim2 = simout.yout{5}.Values.Data'; % error of Q with FF
e_sim3 = simout.yout{6}.Values.Data'; % error of control
u_sim1 = simout.yout{7}.Values.Data'; % input for u(k)
u_sim2 = simout.yout{8}.Values.Data'; % input for u(k+1)

% plotting the output and error
figure('Position', [100, 100, 1200, 400]);
subplot(1,2,1)
if param == 1
    plot(t,u_sim1,'g',t_sim,y_sim1,'r--',t_sim,y_sim2,'c:','linewidth',wth)
    title('Simulink: Output Y - No Disturbance')
    legend('Desired Trajectory','Y-K','Y-K with inverse','location','best')
    xlabel('Time(s)'),ylabel('Displacement'), xlim([0 Tend]), grid
    subplot(1,2,2)
    plot(t_sim,e_sim1,'r',t_sim,e_sim2,'c:','linewidth',wth)
    legend('Y-K','Y-K with inverse','location','best')
    set(gca,'Fontsize',12), grid
    xlabel('Time(s)'),ylabel('Error'), xlim([0 Tend]), grid
    title('Simulink: Error')
    set(gca,'Fontsize',12), grid
elseif param == 2
    plot(t,u_sim1,'g',t_sim,y_sim1,'r',t_sim,y_sim3,'c:','linewidth',wth)
    title('Simulink: Output Y - Disturbance at 60 rad/s')
    legend('Desired Trajectory','Y-K','Control','location','best')
    xlabel('Time(s)'),ylabel('Displacement'), xlim([0 Tend]), grid
    subplot(1,2,2)
    plot(t_sim,e_sim1,'r',t_sim,e_sim3,'c:','linewidth',3)
    legend('Y-K','Control','location','best')
    set(gca,'Fontsize',12), grid
    xlabel('Time(s)'),ylabel('Error'), xlim([0 Tend]), grid
    title('Simulink: Error')
    set(gca,'Fontsize',12), grid
elseif param == 3
    plot(t,u_sim1,'g',t_sim,y_sim1,'r--',t_sim,y_sim3,'c:','linewidth',wth)
    title('Simulink: Output Y - Disturbance at 80 rad/s')
    legend('Desired Trajectory','Y-K','Control','location','best')
    xlabel('Time(s)'),ylabel('Displacement'), xlim([0 Tend]), grid
    subplot(1,2,2)
    plot(t_sim,e_sim1,'r',t_sim,e_sim3,'c:','linewidth',3)
    legend('Y-K','Control','location','best')
    set(gca,'Fontsize',12), grid
    xlabel('Time(s)'),ylabel('Error'), xlim([0 Tend]), grid
    title('Simulink: Error')
    set(gca,'Fontsize',12), grid
else
    plot(t,u_sim1,'g',t_sim,y_sim1,'r',t_sim,y_sim3,'c:','linewidth',wth)
    title('Simulink: Output Y - Disturbance at 100 rad/s')
    legend('Desired Trajectory','Y-K','Control','location','best')
    xlabel('Time(s)'),ylabel('Displacement'), xlim([0 Tend]), grid
    subplot(1,2,2)
    plot(t_sim,e_sim1,'r',t_sim,e_sim3,'c:','linewidth',wth)
    legend('Y-K','Control','location','best')
    set(gca,'Fontsize',12), grid
    xlabel('Time(s)'),ylabel('Error'), xlim([0 Tend]), grid
    title('Simulink: Error')
    set(gca,'Fontsize',12), grid
end

% save graphs
if alpha == 0.2
    if param == 1 && save == 1
        saveas(gcf,'no_dist_0.2.png')
    elseif param == 2 && save == 1
        saveas(gcf,'dist_60_0.2.png')
    elseif param == 3 && save == 1
```

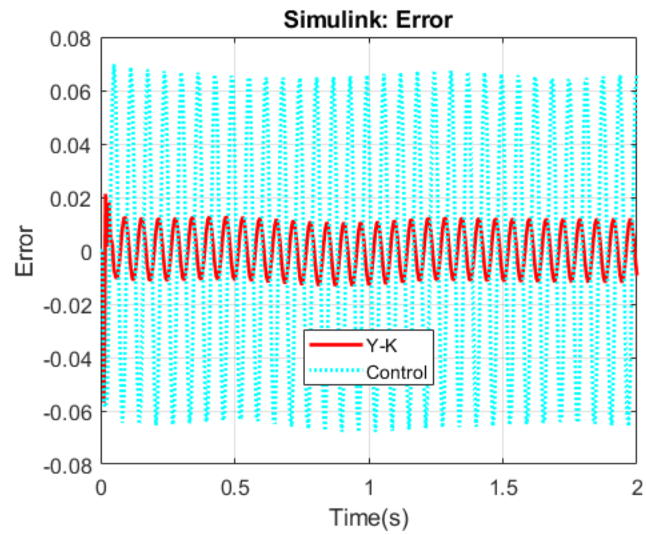
```
        saveas(gcf,'dist_80_0.2.png')
    elseif param == 4 && save == 1
        saveas(gcf,'dist_100_0.2.png')
    else
        save = 0;
    end % end save cases
elseif alpha == 0.6
    if param == 1 && save == 1
        saveas(gcf,'no_dist_0.6.png')
    elseif param == 2 && save == 1
        saveas(gcf,'dist_60_0.6.png')
    elseif param == 3 && save == 1
        saveas(gcf,'dist_80_0.6.png')
    elseif param == 4 && save == 1
        saveas(gcf,'dist_100_0.6.png')
    else
        save = 0;
    end % end save cases
else alpha == 0.99
    if param == 1 && save == 1
        saveas(gcf,'no_dist_0.99.png')
    elseif param == 2 && save == 1
        saveas(gcf,'dist_60_0.99.png')
    elseif param == 3 && save == 1
        saveas(gcf,'dist_80_0.99.png')
    elseif param == 4 && save == 1
        saveas(gcf,'dist_100_0.99.png')
    else
        save = 0;
    end % end save cases
end
end % end of freq disturb (4 loops)
```











Velocity profile

```
function [xd,vd,ad,t,nfig] =  sinusodial_yacc(t_i,t_f,tmax,y_i,y_f,nfig,Tend);
T =(t_f-t_i);  A = (y_f-y_i)*2*pi/(T*T); % this would make max yd =1
t = 0:0.00625:tmax;
ad = 0*t; % initialize desired acceleration;
vd = 0*t; % initialize desired velocity;
xd = 0*t; % initialize desired position;
for jj = 1:length(t)
    if  t(jj)< t_i
        ad(jj) = 0;
        vd(jj) = 0;
        xd(jj) = y_i;
    elseif t(jj) < t_i+T
        tt = t(jj)-t_i;
        ad(jj) = A*sin(2*pi*tt/T);
        vd(jj) = (A*T/(2*pi))*(1 -cos(2*pi*tt/T));
        xd(jj) = y_i + (A*T/(2*pi))*tt  -(A*T*T/(4*pi*pi))*sin(2*pi*tt/T);
    else
        ad(jj) = 0;
        vd(jj) = 0;
        xd(jj) = y_f; %(A*T/(2*pi))*T  -(A*T*T/(4*pi*pi))*sin(2*pi*T/T);
    end
end

figure()
plot(t,xd,'LineWidth',2)
xlabel('Time(s)')
ylabel('Position')
title('Desired Trajectory')
xlim([0 Tend])
grid; set(gca,'FontSize',12)
end
```

Published with MATLAB® R2020a