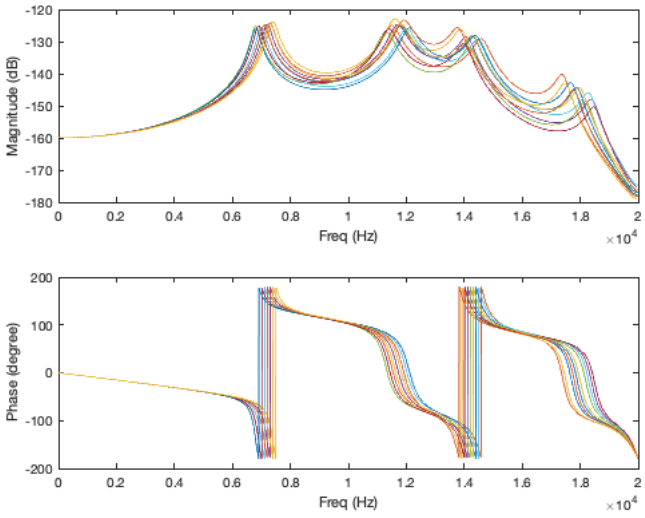
Contents

Anthony Peters, Robust HardDrive Control, 5/5/23

```
clear; close all; clc
load PlantData

u_fft_raw = fft(u);
N = length(u_fft_raw);
u_fft_half = u_fft_raw(1:N/2+1); % input fft

Ts =    0.000025     ;                       % Sampling time
f_nq = (1/Ts)/2;                             % Nyquist frequency
df =    f_nq / (N/2) ;                       % Frequency step
freq = [0:df:f_nq]   ;                       % Frequency vector in Hz (from df to f_nq)


figure; hold on
for i = 1:size(y,1)
    y_fft = fft(y(i,:));
    y_fft_half = y_fft(1:N/2+1);
    P_FR{i} = y_fft_half./u_fft_half;     % Plant frequency response
    P_FRD{i} =frd(P_FR{i},freq*2*pi,Ts) ; % Plant FRD
    bode281(P_FRD{i},freq*2*pi,'Lin') ;
end
```

```
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
Warning: Integer operands are required for colon operator when used as index.
```

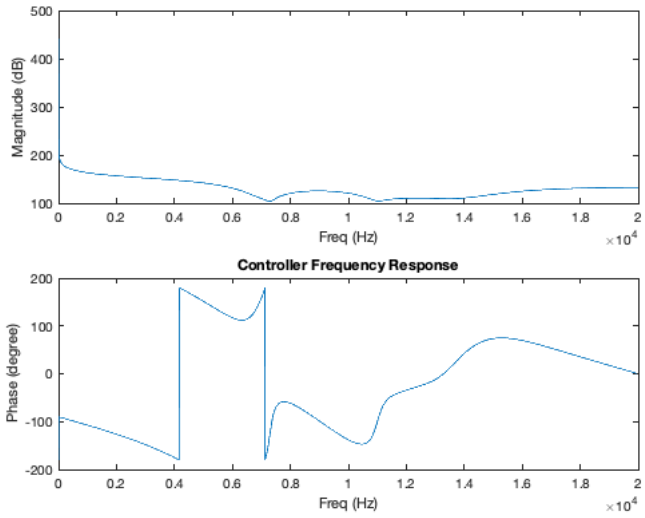


Control Parameters

```
KI = 916e9;  % Integral gain
KP = 1000e4;   % Proportional gain

NotchD = [17     25       20          12       11      10     ];     % Depth of the notch filters
NotchW = [5800  5000     5000        9000     5800    7700 ];     % Width of the notch filters
NotchF = [6900  7300    11000     12000   13600  14300];    % Frequencies of the notch filters
```

Controller Calculation

```
PI_s = tf([KP KI],[1 0]);                    % PI controller
PI_z = c2d(PI_s,Ts,'matched');              % PI controller discretization
C_z = PI_z ;                                   % Full controller initilization

for i = 1:max(6,length(NotchD))          % Maximum 6 notch filters are allowed!
    C_z = C_z * NotchTF(NotchD(i),NotchW(i),NotchF(i),NotchF(i), Ts);
end
figure; bode281(C_z,freq*2*pi,'Lin')
title('Controller Frequency Response')
```

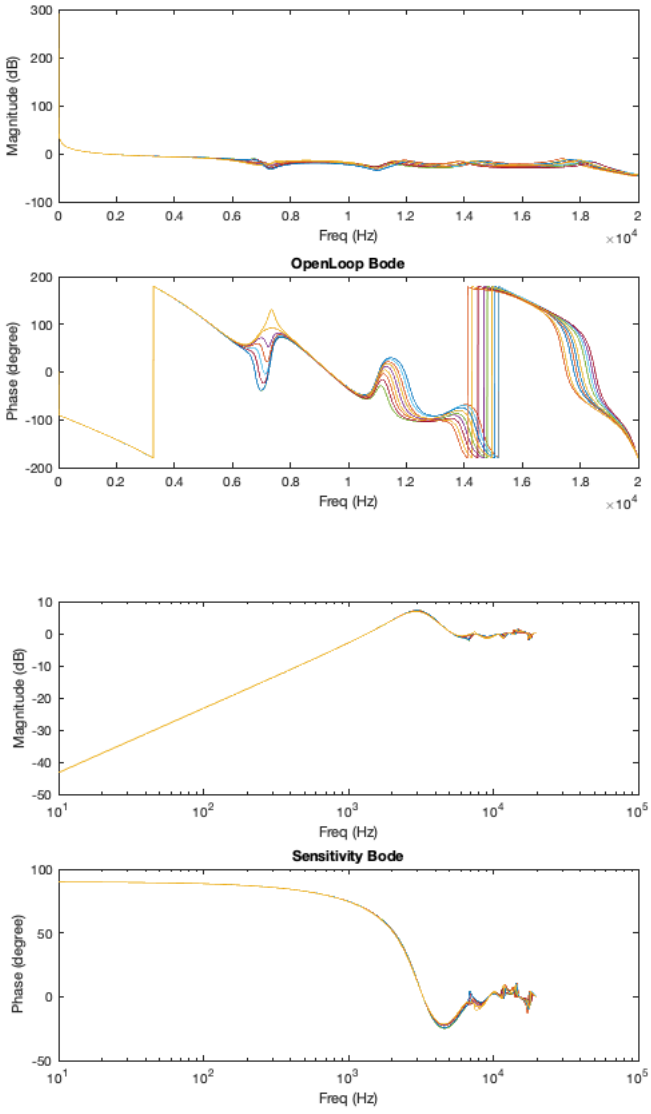


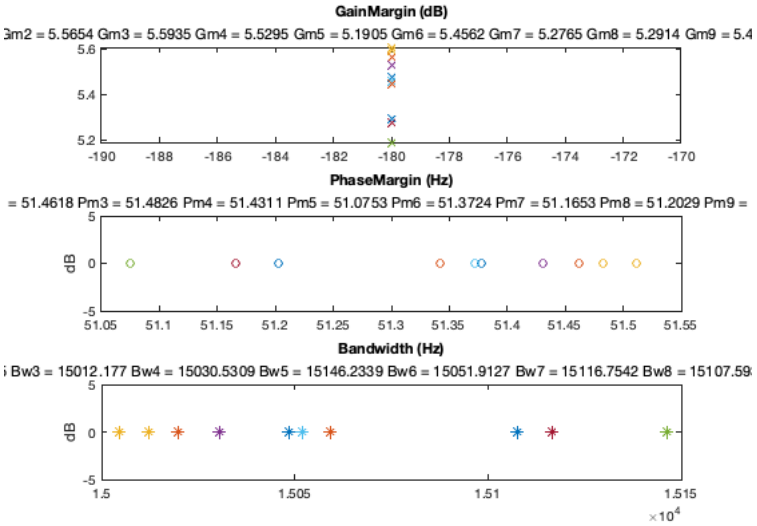
Closed Loop Frequency Response

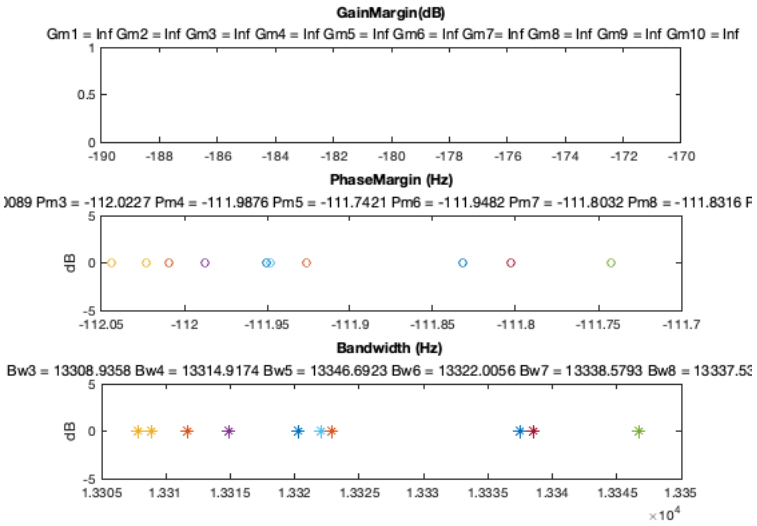
Create Two Responses for the OL and Sensitivity TFs

```
for i = 1:size(y,1)

    figure(3); hold on
    OL_TF = C_z * P_FRD{1, i} ;
    title("OpenLoop Bode"); bode281(OL_TF,freq*2*pi,'Lin'); hold off        % Plot the frequency response of the OL TFs (Fig number, bode281'lin')

    figure(4);  hold on
    Sense_TF= 1 / (1 + OL_TF);                                                  % Calculate the sensitivity tranfer functions
    bode281(Sense_TF,freq*2*pi,'Log');title("Sensitivity Bode"); hold off  % Plot the frequency response of the OL TFs (Fig number, bode281'log')
    Sense_TF_FRD{i} =  1 / (1 + OL_TF) ;



%Margins of OL and SensitivyTF
    [OL_gm{i}, OL_pm{i}, OL_w_cg{i}, OL_w_cp{i}] = margin(OL_TF);                % Gives GM = 1/L , Gives Cp = rad/s / 2*pi
    GainMargin_OL{i} =  20*log10(abs(OL_gm{i}));
    CrossOverGain_HZ_OL{i} = OL_w_cp{i} / 2*pi ;

    [S_gm{i}, S_pm{i}, S_w_cg{i}, S_w_cp{i}] = margin(Sense_TF);
    GainMargin_Sense{i} =  20*log10(abs(S_gm{i}));
    Sense_CrossOverGain_HZ{i} = S_w_cp{i} / 2*pi ;


end

%Sort and plot Gain margins, Phase margins, and Bandwidths in the same
for i = 1:size(y,1)

  %Margins for Open Loop TF
    figure(5); hold on
    subplot(3,1,1)
    plot(-180, GainMargin_OL{i}, 'x');
    title('GainMargin (dB)', [' Gm1 = ' , num2str(GainMargin_OL{1}), ' Gm2 = ', num2str(GainMargin_OL{2}),' Gm3 = ', num2str(GainMargin_OL{3}), ' Gm4 =
    subplot(3,1,2)
    plot(OL_pm{i}, 0, 'o');
    title('PhaseMargin (Hz)', [' Pm1 = ' , num2str(OL_pm{1}), ' Pm2 = ', num2str(OL_pm{2}),' Pm3 = ', num2str(OL_pm{3}), ' Pm4 = ', num2str(OL_pm{4}),
    subplot(3,1,3)
    plot( CrossOverGain_HZ_OL{i},0, '*');
    title('Bandwidth (Hz)', [' Bw1 = ' , num2str( CrossOverGain_HZ_OL{1}), ' Bw2 = ', num2str( CrossOverGain_HZ_OL{2}),' Bw3 = ', num2str( CrossOverGa

  %Margins for Sensitivity TF
    figure(6); hold on
    subplot(3,1,1)
    plot(-180, GainMargin_Sense{i}, 'x');
    title('GainMargin(dB)', [' Gm1 = ' , num2str(GainMargin_Sense{1}), ' Gm2 = ', num2str(GainMargin_Sense{2}),' Gm3 = ', num2str(GainMargin_Sense{3})
    subplot(3,1,2)
```

```
    plot(S_pm{i}, 0, 'o');
    title('PhaseMargin (Hz)', [' Pm1 = ' , num2str(S_pm{1}), ' Pm2 = ', num2str(S_pm{2}),' Pm3 = ', num2str(S_pm{3}), ' Pm4 = ', num2str(S_pm{4}), ' Pm
    subplot(3,1,3)
    plot(Sense_CrossOverGain_HZ{i},0, '*');
     title('Bandwidth (Hz) ', [' Bw1 = ' , num2str(Sense_CrossOverGain_HZ{1}), ' Bw2 = ', num2str(Sense_CrossOverGain_HZ{2}),' Bw3 = ', num2str(Sense_
end
```



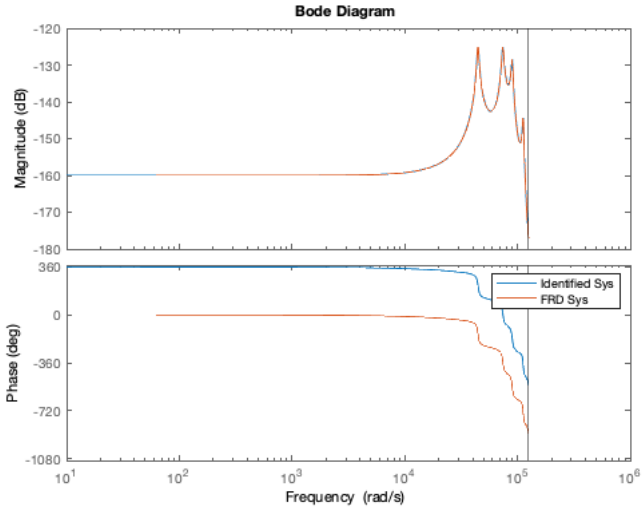


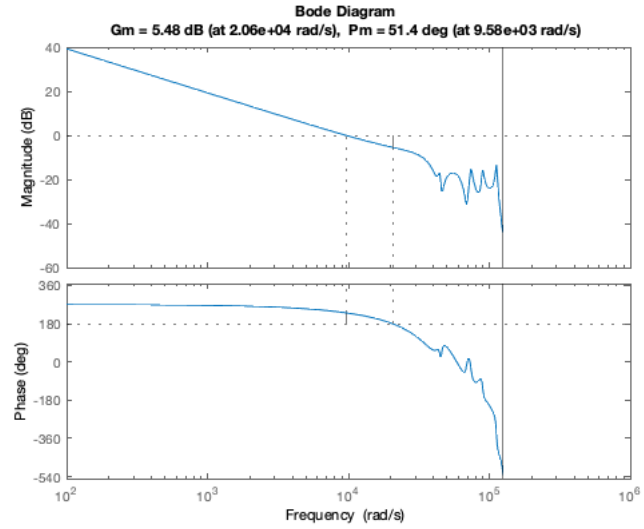


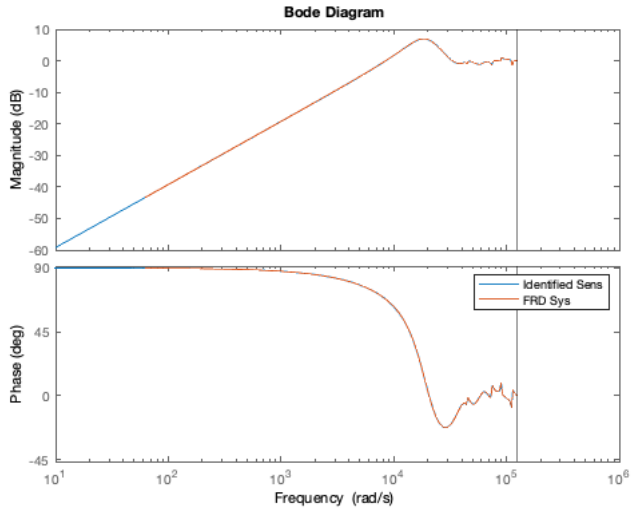
System Identifcation

```
%Compare the Bode Plot of P(Z) and P_FRD
   P_FR_plant_1 = P_FR{1,1};
   P_FRD_plant_1 =P_FRD{1,1};

   [B,A] = invfreqz(P_FR_plant_1, freq/freq(end)*pi,21,21);
   P_ID = tf(B,A,Ts);
   figure(7); bode(P_ID) ;hold on; bode(P_FRD_plant_1); hold on % Plot the Bode plot of the identified plant and the FRD plant
   legend('Identified Sys','FRD Sys')
   figure(8); margin(C_z*P_ID);hold on                          % Check the margins of the identified model:

%Check Bode plot of Sens_TF(z) and Sens_TF_FR
   Sense_z_plant_1 = 1 / (1 + (P_ID * C_z));
   Sense_FRD_plant_1 = Sense_TF_FRD{1,1};
   figure(9); bode(Sense_z_plant_1) ;hold on; bode(Sense_FRD_plant_1); hold on
   legend('Identified Sens','FRD Sys')
```

```
Warning: Matrix is close to singular or badly scaled. Results may be
inaccurate. RCOND =  3.290979e-26.
```







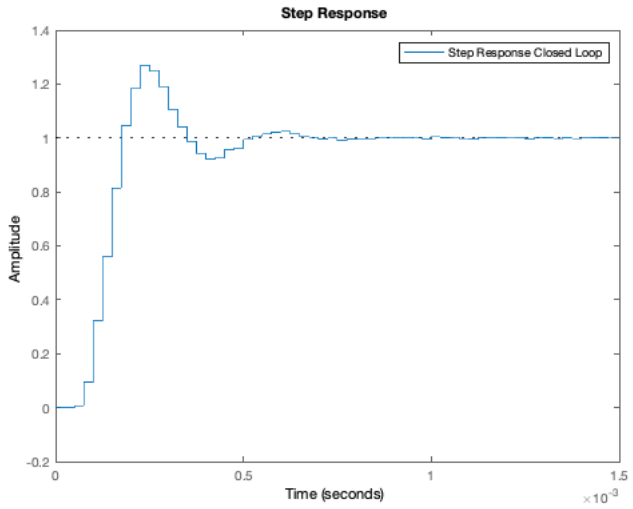
Step Response Tracking

Step response of the closed loop system

```
plant1_OL_ID = C_z*P_ID ;
sys1_ID = feedback(plant1_OL_ID,1);
stepinfo(sys1_ID)
```

```
figure(10); step(sys1_ID); hold on
legend('Step Response Closed Loop');
```

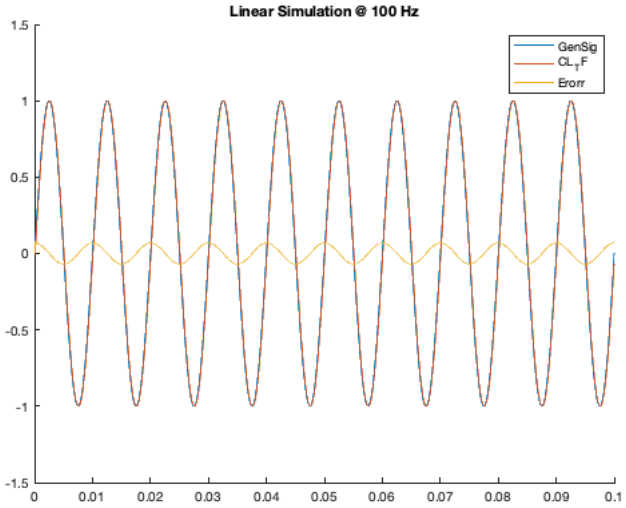
```
ans =

  struct with fields:

        RiseTime: 7.5000e-05
    TransientTime: 6.2500e-04
     SettlingTime: 6.2500e-04
      SettlingMin: 0.9213
      SettlingMax: 1.2678
        Overshoot: 26.7784
       Undershoot: 3.8736e-06
             Peak: 1.2678
         PeakTime: 2.2500e-04
```

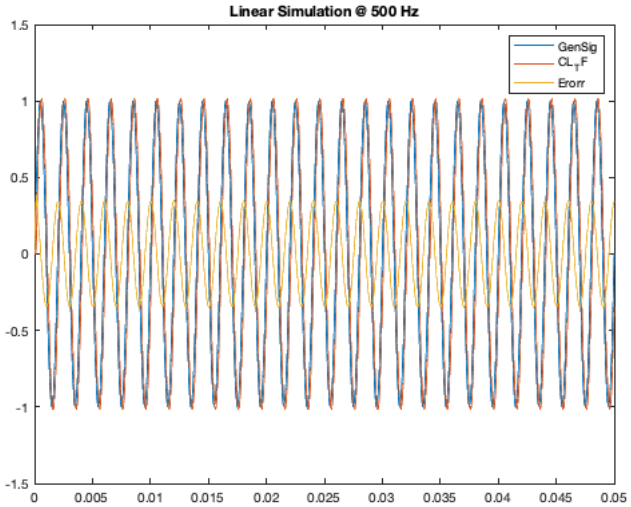


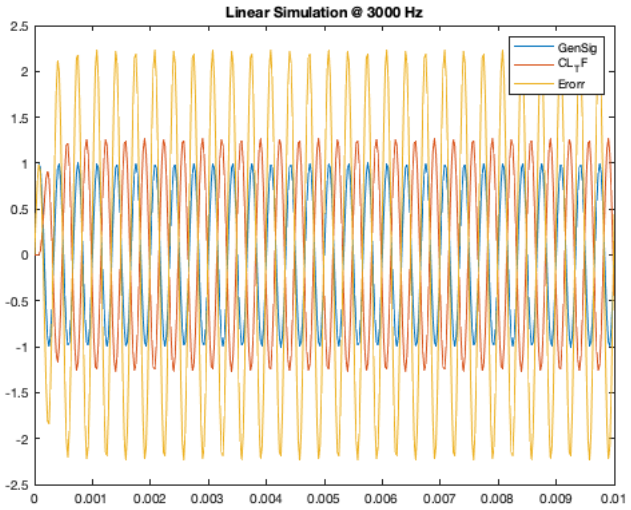
Sine Wave Tracking

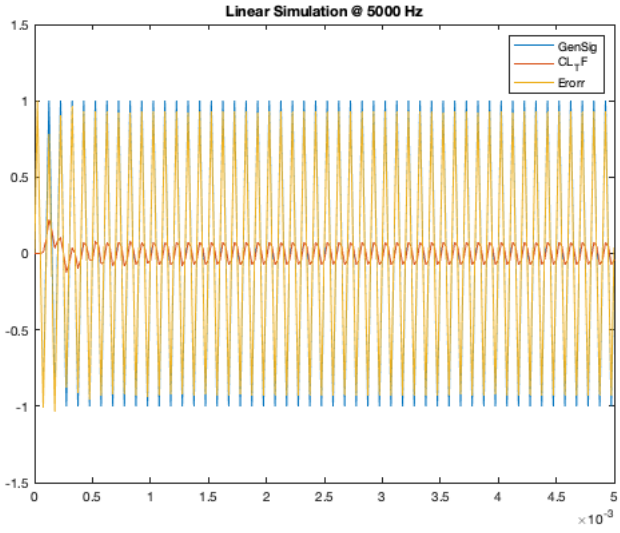
```
%Sine wave tracking simulation using lsim (100 Hz, 500 Hz, 3000 Hz, 10000 Hz)
sim_freq =[100 , 500 , 3000, 10000];
%sim_freq =[2000 , 8000 , 10000, 15000];

    f = sim_freq ;
    setpoint = 1 ;

    figure(11); hold on;
    [u, t] = gensig("sine", 1/f(1) , .1, Ts);

    y = lsim(sys1_ID,u,t);
    plot(t,u); hold on      % Generated Signal
    plot(t,y); hold on      % Simulated Signal
    plot(t, u - y); hold on % Error Signal
    hold on; title('Linear Simulation @ 100 Hz'); hold on
    legend('GenSig','CL_TF','Erorr')

    figure(12);
    [u, t] =gensig("sine", 1/f(2) , .05, Ts);

    y = lsim(sys1_ID,u,t);
    plot(t,u); hold on      % Generated Signal
    plot(t,y); hold on      % Simulated Signal
    plot(t, u - y); hold on % Error Signal
    hold on; title('Linear Simulation @ 500 Hz'); hold on
    legend('GenSig','CL_TF','Erorr')

    figure(13);
    [u, t] =gensig("sine", 1/f(3) , .01, Ts);

    y = lsim(sys1_ID,u,t);
    plot(t,u); hold on      % Generated Signal
    plot(t,y); hold on      % Simulated Signal
    plot(t, u - y); hold on % Error Signal
    hold on; title('Linear Simulation @ 3000 Hz'); hold on
    legend('GenSig','CL_TF','Erorr')

    figure(14);

    [u, t] =gensig("sine", 1/f(4) , .005, Ts);
```

```
y = lsim(sys1_ID,u,t);
plot(t,u); hold on       % Generated Signal
plot(t,y); hold on       % Simulated Signal
plot(t, u - y); hold on % Error Signal
hold on; title('Linear Simulation @ 5000 Hz'); hold on
legend('GenSig','CL_TF','Erorr')
```









Published with MATLAB® R2022a