ECE141 Final Design Project TEMPLATE

Table of Contents

Kambiz Shoarinejad, Feb 26, 2018 Based on MIT 2.14 Design Project By: Darya Amin-Shahidi, May 3., 2011 DL Trumper April, 27, 2014

Important Notes

Please use this template to present your design.

```
% - You are required to add your code to perform the different design
 tasks
% and initialize the transfer functions and variable declared as []
 within the
% template.

% - The template will generate the required plots based on your
 design.

% - The variable names used here match the diagrams and the text of
 the
%   design problem.

% - Do not change any variable names declared here and do not
 overwrite
% the values given in the problem set.
```

Values and data given in the problem statement

```
Rc=4;       %coil resistance
Lc=2e-4;    %coil inductance
Kf=20;      %motor constant
R1=10e3;    %value of resistance R1
```

```
Rs=0.2;      %value of sense resistance
Ga=-0.5;     %closed loop low-freq gain of the current amplifier
G1=5e5;      %capacitive probe sensor's gain

load 'testfreqdata';     % loads the experimental frequency response
 data on the
                         % mechanical plant Gp=X/F
                         % Make sure the data file is stored in the
 same
                         % folder

s=tf('s'); % Specify the Laplace variable s as a tf object so you can
 specify your transfer functions in variable s
```

Electrical Loop Design

Initialize the following transfer functions and the component values

```
P_elec = 25*Rs/(s*Lc + Rc + Rs) %electrical plant transfer function:
 Vs/Vc

% Designing the current controller
R2 = 5000 %value of resistor R2
R3 = 121287 %value of resistor R3
C1 = 1.37415*10^(-10) %value of capacitor C1
C2 = 1.38831*10^(-12) %value of capacitor C2

C_elec = (R3*C1*s + 1)/(s*R2*(C1 + C2 + R3*C1*C2*s)) %electrical
 controller transfer function -Vc/Vs
           %specify the transfer function in terms of the resistor
 and
           %capacitor values given above
H_vsic = 1             % transfer function: Ic/Vs
H_vsetvr = -0.5           % transfer function: Vr/Vset

L_elec = P_elec * C_elec;    %electrical loop transfer function
```

```
P_elec =

        5
  --------------
  0.0002 s + 4.2

Continuous-time transfer function.


R2 =

        5000


R3 =
```

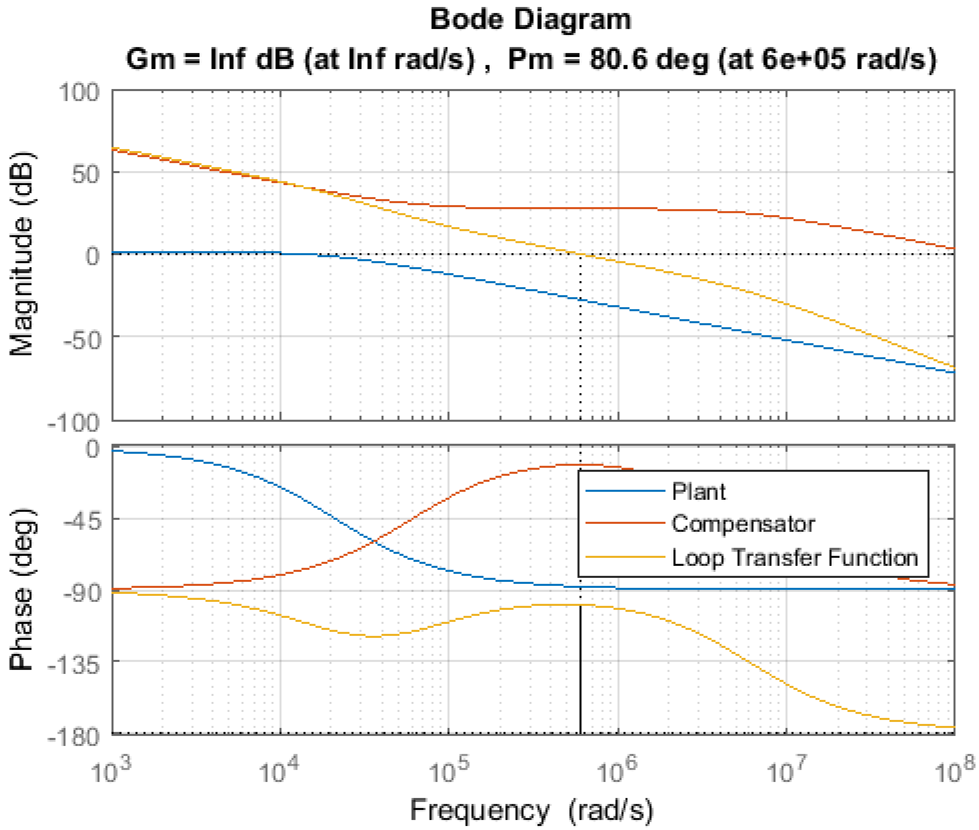
```
      121287


C1 =

   1.3742e-10


C2 =

   1.3883e-12


C_elec =

       1.667e-05 s + 1
  --------------------------
  1.157e-13 s^2 + 6.94e-07 s

Continuous-time transfer function.


H_vsic =

     1


H_vsetvr =

   -0.5000
```

Plot and report Loop-shaping results for the electrical loop

Find and print the DC gain from Vset to Ic

```
curr_cl = H_vsetvr*feedback(L_elec,1,-1)*H_vsic;     % current closed
 loop transfer function: Ic/Vset
fprintf('DC gain from Vset to Ic: %4.2f\n',dcgain(curr_cl));

% Plot the Bode diagrams for the plant, the compensator, and the loop
 transfer function for the
% electrical loop, and confirm phase margin and gain crossover
 frequency
figure;
bode(P_elec);
hold on;
bode(C_elec);
margin(L_elec);
grid on;
legend('Plant','Compensator','Loop Transfer Function');
```

```
DC gain from Vset to Ic: -0.50
```



Mechanical Loop Design

Initialize the following transfer functions

```
% fitted mechanical plant Gp_fit=X/F

Gp = 10^2/(s^2 + 100.452*s + 10^6) * exp(-1.2*10^(-5)*s)
%data = frd(Gpmag.*exp(1j*Gpphase*pi/180),ww);
%Gp = tfest(data, 2)
C_mech = -0.1*(1+(3.38*10^4/5)/s)*((1+3.1715e-4*s)/(1+3.1529e-5*s))  %
 mechanical controller V_set/V_e

L_mech = C_mech*Ga*Kf*Gp*G1; % mechanical loop transfer function
```

```
Gp =

                                   100
  exp(-1.2e-05*s) * --------------------
                    s^2 + 100.5 s + 1e06

Continuous-time transfer function.
```

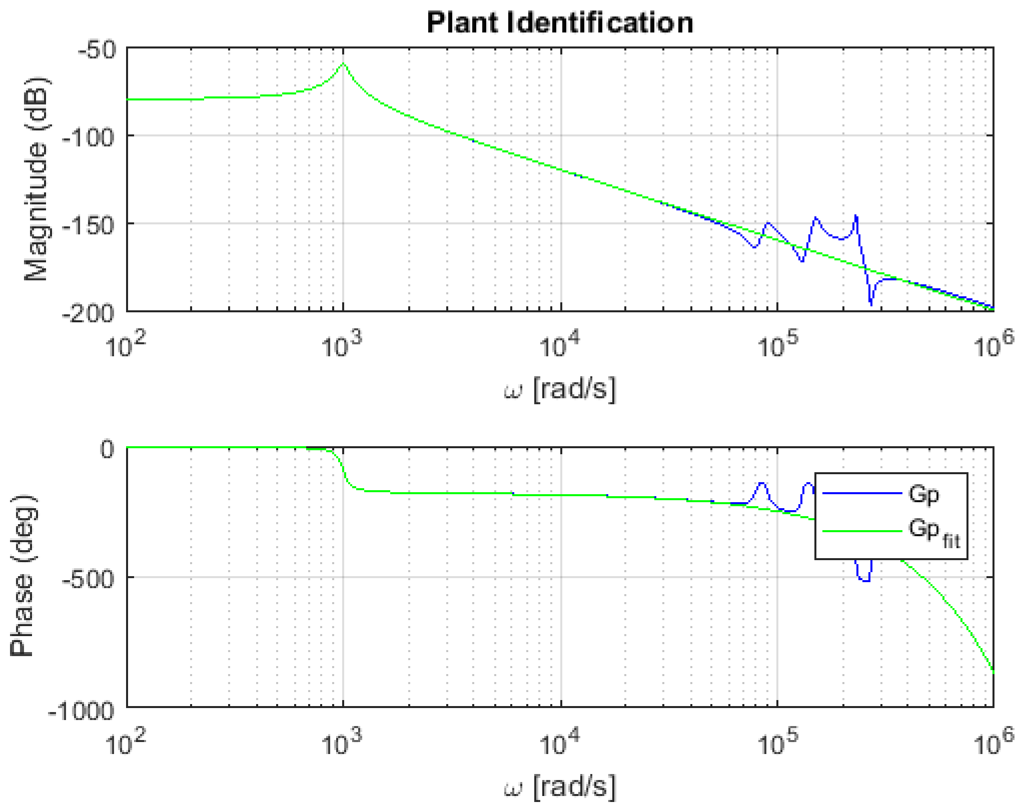
```
C_mech =

  -3.171e-05 s^2 - 0.3144 s - 676
  -------------------------------
         3.153e-05 s^2 + s

Continuous-time transfer function.
```

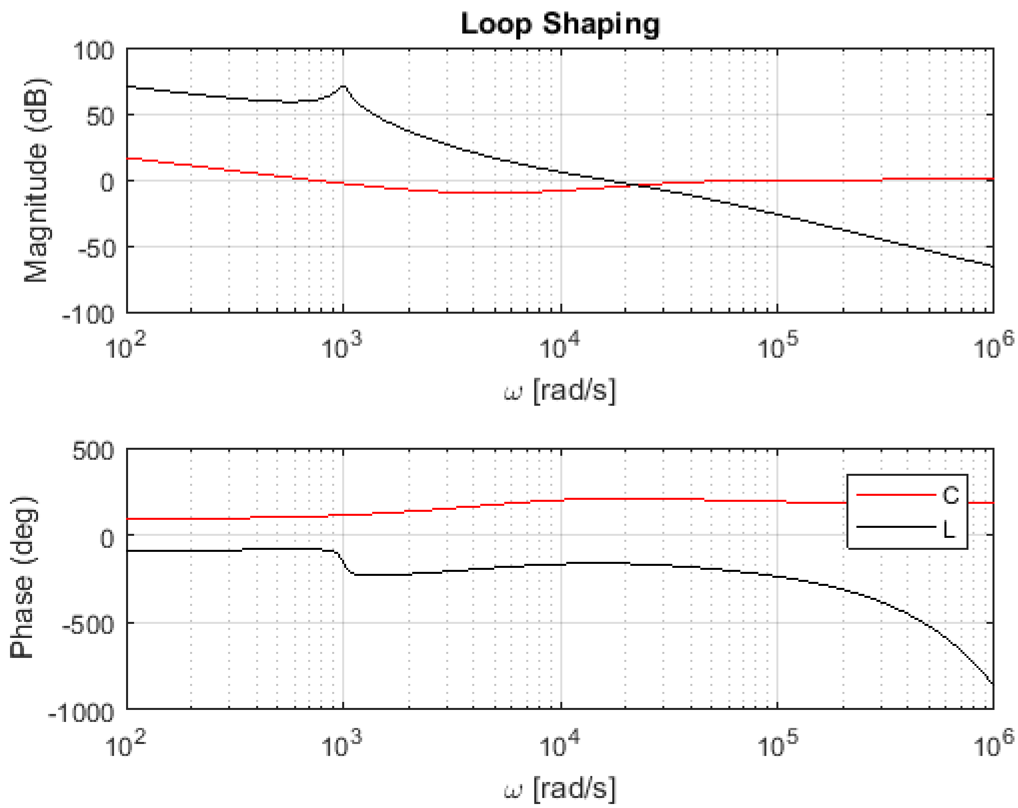
Plot the loop-shaping results for the mechanical loop

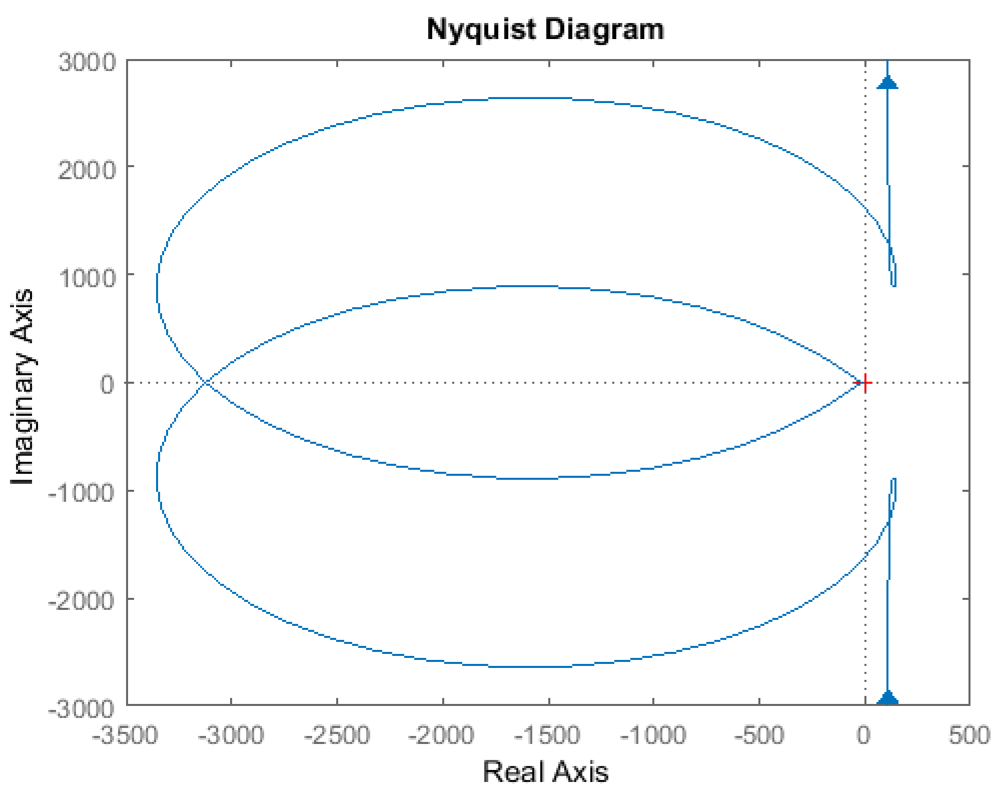
```
[Pm Pp]=bode(Gp,ww); Pm=Pm(:); Pp=Pp(:);
[Cm Cp]=bode(C_mech,ww); Cm=Cm(:); Cp=Cp(:);
[Lm Lp]=bode(L_mech,ww); Lm=Lm(:); Lp=Lp(:);

% Plot and compare the Bode plots of the measured freq response data
 vs the fitted
% model
figure;
subplot(2,1,1)
semilogx(ww,20*log10(Gpmag),'b',ww,20*log10(Pm),'g');
grid on;
title('Plant Identification')
ylabel('Magnitude (dB)');
xlabel('\omega [rad/s]');
subplot(2,1,2)
semilogx(ww,Gpphase,'b',ww,Pp,'g');
legend('Gp','Gp_f_i_t')
ylabel('Phase (deg)');
xlabel('\omega [rad/s]');
grid on;

% Plot the Bode diagrams for the compensator and the loop transfer
 function
figure;
subplot(2,1,1)
semilogx(ww,20*log10(Cm),'r',ww,20*log10(Lm),'k');
grid on;
title('Loop Shaping')
ylabel('Magnitude (dB)');
xlabel('\omega [rad/s]');
subplot(2,1,2)
semilogx(ww,Cp,'r',ww,Lp,'k');
legend('C','L')
ylabel('Phase (deg)');
xlabel('\omega [rad/s]');
grid on;

% Plot the Nyquist diagram of the loop transfer function
figure;
nyquist(L_mech);        % Nyquist Plot
```

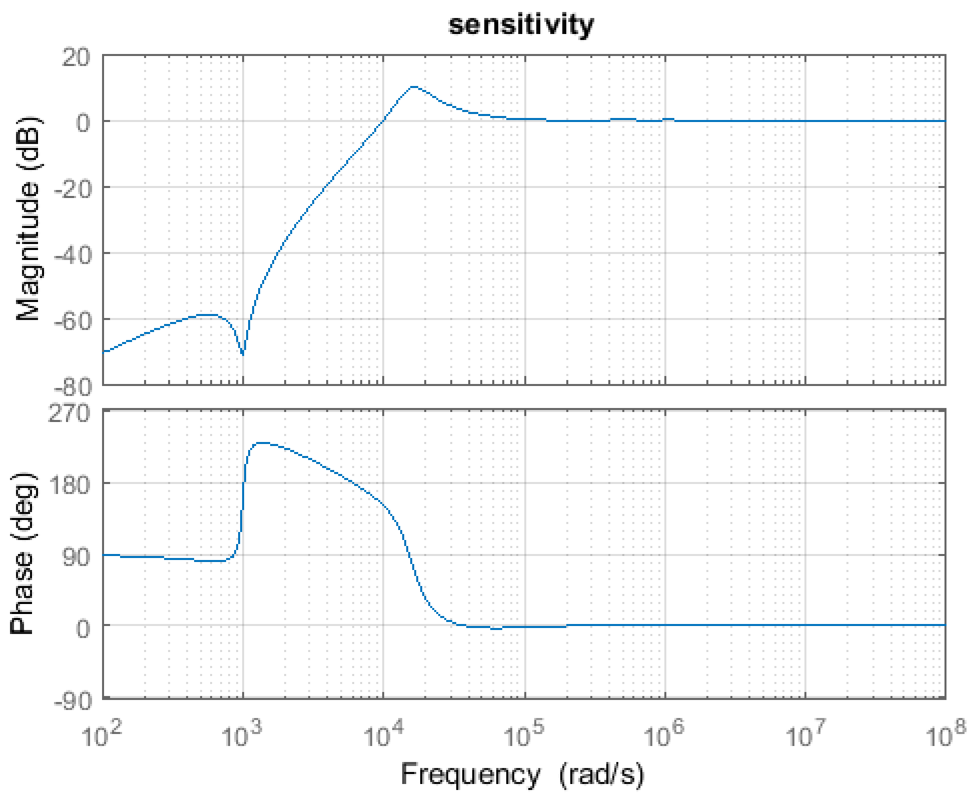
```
L_mechS=ss(L_mech);     % the transfer function is converted to state
 space
                        % to allow arithmetic with the delay term

% Obtain and plot the Sensitivity function, and confirm max
 sensitivity
S=feedback(1,L_mechS);  %sensitivity transfer function
figure;
[Sm,Sp]=bode(S);
bode(S);
title('sensitivity')
grid on;
maxS = max(Sm);
fprintf('Maximum Sensitivity (dB): %6.2f\n',20*log10(maxS));
CL=feedback(L_mechS,1); % closed loop x/x_ref transfer function
```

Maximum Sensitivity (dB):   9.96




Loop Shaping
Magnitude (dB)
100
50
0
-50
-100
10^2
10^3
10^4
10^5
10^6
ω [rad/s]
Phase (deg)
500
0
-500
-1000
C
L
ω [rad/s]


Nyquist Diagram
Imaginary Axis
3000
2000
1000
0
-1000
-2000
-3000
-3500
-3000
-2500
-2000
-1500
-1000
-500
0
500
Real Axis



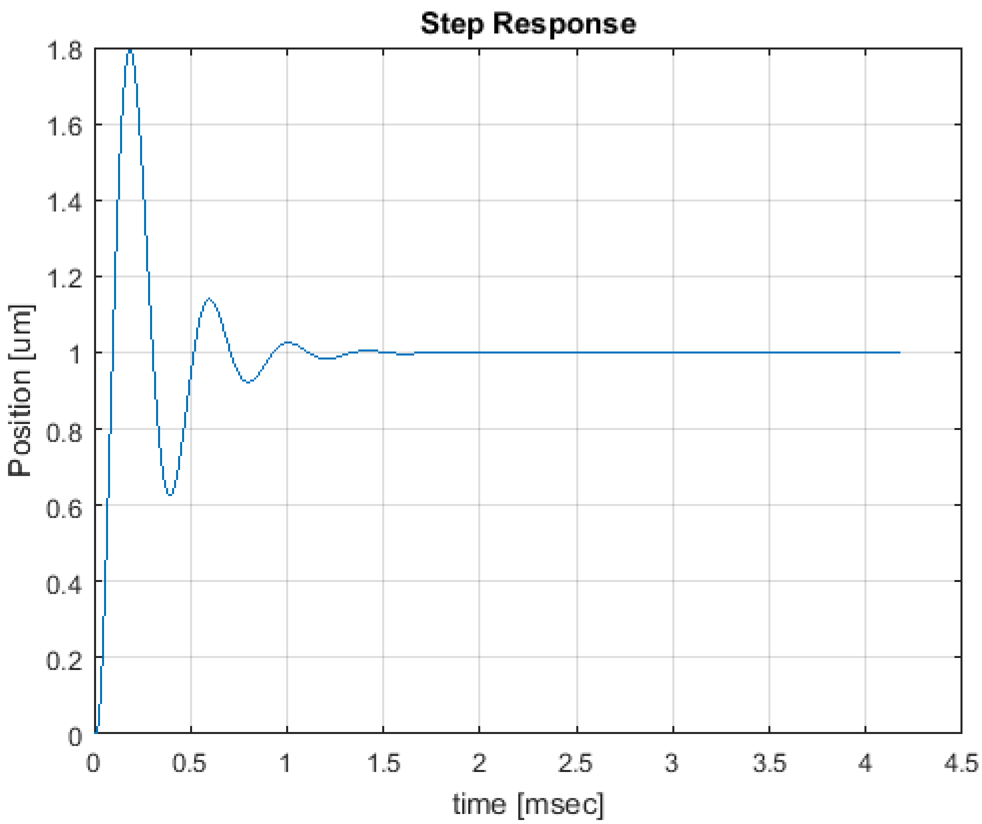
Simulate Time Response

Simulate and plot the closed-loop step response and report the transient specs as well as the steady-state error

```
figure;
step_amp = 1e-6;   % 1um step input in position
step_resp_params = stepinfo(CL)
ts = step_resp_params.SettlingTime;
t=0:ts/100:6*ts;
opt = stepDataOptions;
opt.StepAmplitude = step_amp;
[x]=step(CL,t,opt);
e=x'-step_amp; % SS tracking error
t_indx = t<4*ts;
plot(t(t_indx)*1e3,x(t_indx)*1e6); % plot step response
ss_indx = t>5*ts;
title('Step Response')
ylabel('Position [um]')
xlabel('time [msec]')
grid on;
```

```
fprintf('RMS SS step tracking error (nm): %6.4f
\n',rms(e(ss_indx))*1e9);
```

```
step_resp_params =

  struct with fields:

        RiseTime: 5.6722e-05
    SettlingTime: 0.0010
     SettlingMin: 0.6242
     SettlingMax: 1.7939
       Overshoot: 79.3900
      Undershoot: 0
            Peak: 1.7939
        PeakTime: 1.8719e-04

RMS SS step tracking error (nm): 0.0000
```

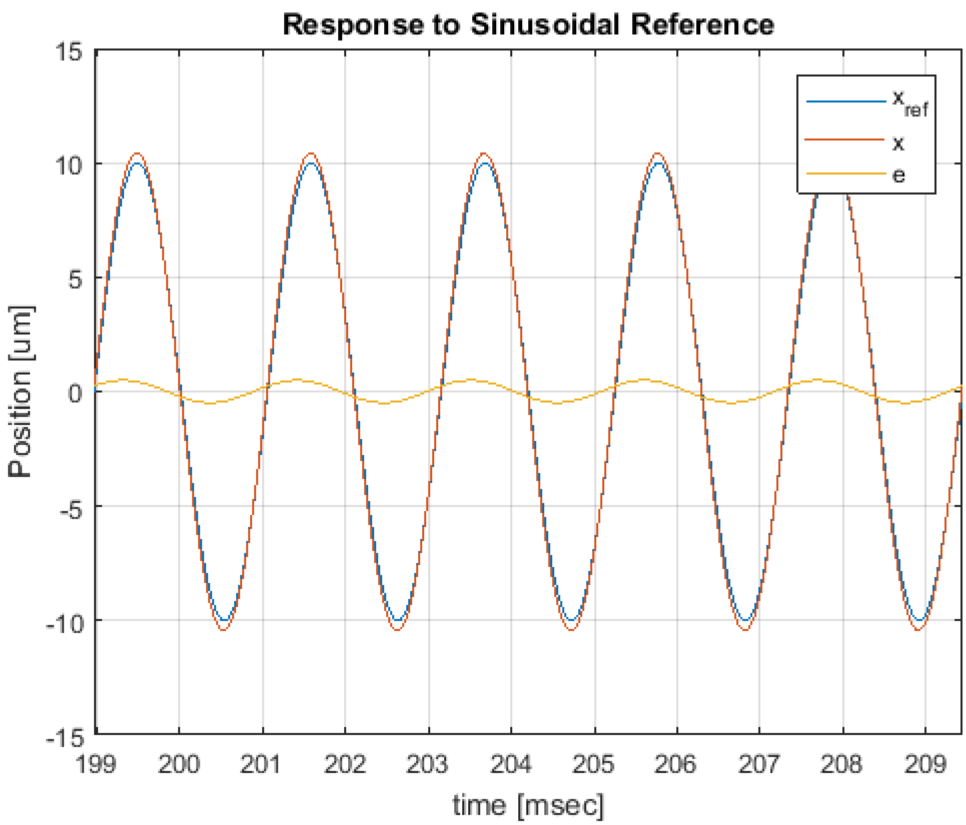


Simulate and plot response to sinusoidal x_ref and report the steady-state error

```
figure;
wr=3e3;                         % frequency of the reference signal [rad/
s]
```

```
fr=wr/(2*pi);                  % frequency of the reference signal [Hz]
t=0:(1/fr/100):(1/fr)*100;     % simulation time = 100 cycles
x_ref=1e-5*sin(wr*t);          % position reference signal (x_ref) [m]
x=lsim(CL,x_ref,t);            % Simulate
e=x'-x_ref;                    % tracking error
plot(t*1e3,x_ref*1e6,t*1e3,x*1e6,t*1e3,e*1e6);
ss_indx = t>0.95*t(end);
ss_time = 1e3*t(ss_indx);
xlim([ss_time(1) ss_time(end)]);
legend('x_{ref}','x','e');
title('Response to Sinusoidal Reference')
ylabel('Position [um]')
xlabel('time [msec]')
grid on;
fprintf('RMS SS sinuosoidal tracking error (um): %6.4f
\n',rms(e(ss_indx))*1e6);
```

RMS SS sinuosoidal tracking error (um): 0.3463

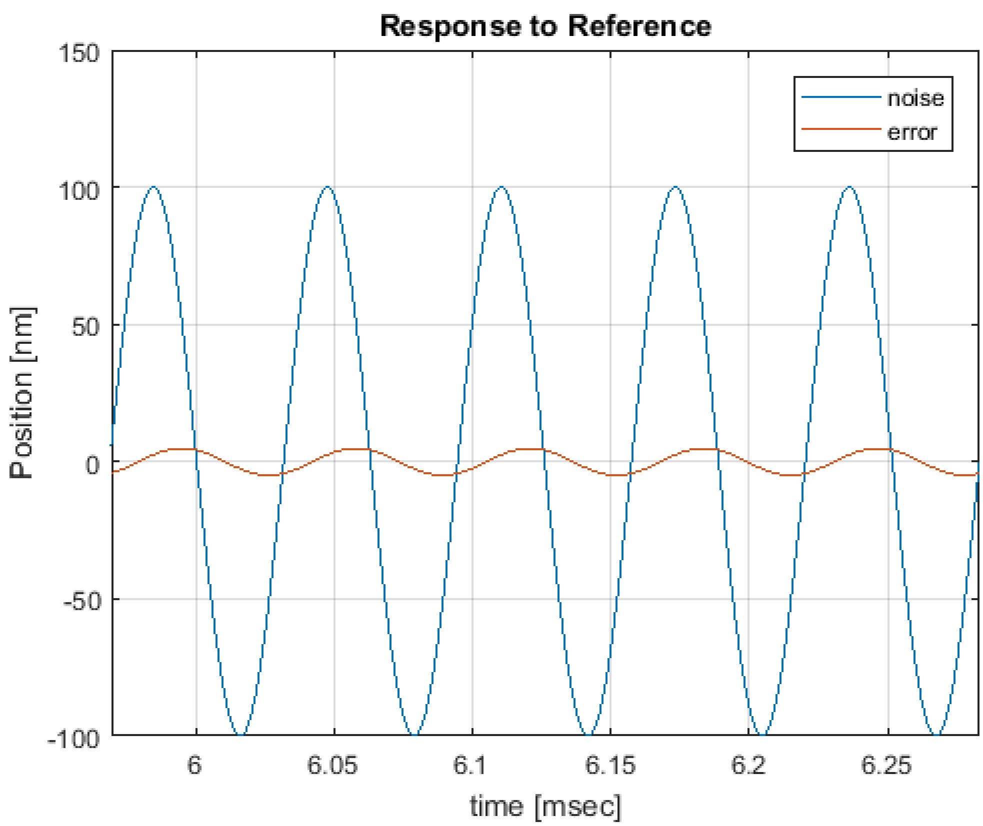


Simulate and plot response to measurement noise and report the steady-state output due to noise

```
figure;
```

```
wn=1e5;                         % frequency of the noise [rad/s]
fn=wn/(2*pi);                   % frequency of the noise signal [Hz]
t=0:(1/fn/100):(1/fn)*100;      % simulation time = 100 cycles
Vn=5e-2*sin(wn*t);              % noise signal (x_ref) [V]
CLn=(-L_mechS/G1)/(1+L_mechS); % x/Vn transfer function
e=lsim(CLn,Vn,t);               % Simulate the noise response
plot(t*1e3,Vn/G1*1e9,t*1e3,e*1e9);
ss_indx = t>0.95*t(end);
ss_time = 1e3*t(ss_indx);
xlim([ss_time(1) ss_time(end)]);
legend('noise','error');
title('Response to Reference')
ylabel('Position [nm]')
xlabel('time [msec]')
grid on;
fprintf('RMS SS noise error (nm): %6.4f\n',rms(e(ss_indx))*1e9);
```

RMS SS noise error (nm): 3.4876



Published with MATLAB® R2017a