ME3534 - Lab 4 - Simple Compensator Design and Validation

Name: Zachary Rohloff

Lab Bench #: 4

This LiveScript documents the results of the in-lab and post-lab processing for ME3534 Lab 4. A PDF version of this document will serve as your Lab 4 report. After you have collected dat in the lab, you will then add your own Matlab code to process this data outside of the lab. Finally, you must convert this LiveScript to a PDF for submission to Canvas.

In this lab, you will have the opportunity to apply material you learned in ME3534 to evaluate simple proportional feedback compensator designs.

Note that your code for Tasks 3, 4, and 5 should be identical except for different filenames of data that you load. It is highly recommended that you get one Task complete, then copy/paste the Matlab code to the other two tasks.

```
clear, clc

fs = 1000;      % sample rate (Hz)
ts = 1/fs;      % sample period (sec)

set(0,'defaultaxesfontsize',14);
set(0,'defaultlinelinewidth',2);
figpos = [0 0 1300 900];  % good size for displaying figures:
figure('position',figpos);
format long g
```

Task 1a: Experimental Low-Bandwidth Proportional Control (In-Lab)

Define an LTI object for the proportional compensator using a proportional gain of **KP=1**

```
KP_1 = 1; % example proportional feedback gain
ControlL = tf(KP_1); % this creates a simple LTI proportional feedback
```

Make sure that the controller LTI object is in the Matlab workspace, then load it into the me3534_lab_gui.

Confirm that the GUI is displaying the correct controller

Collect closed-loop response data and save the results to a unique .mat file.

Task 1b: Experimental Medium-Bandwidth Proportional Control (In-Lab)

Define an LTI object for the proportional compensator using a proportional gain of **KP=5**

```
KP_2 = 5; % example proportional feedback gain
ControlM = tf(KP_2); % this creates a simple LTI proportional feedback
```

Make sure that the controller LTI object is in the Matlab workspace, then load it into the me3534_lab_gui.

Confirm that the GUI is displaying the correct controller

Collect closed-loop response data and save the results to a unique .mat file.

Task 1c: Experimental High-Bandwidth Proportional Control (In-Lab)

Define an LTI object for the proportional compensator using a proportional gain of **KP=10**

```
KP_3 = 10; % example proportional feedback gain
ControlH = tf(KP_3); % this creates a simple LTI proportional feedback
```

Make sure that the controller LTI object is in the Matlab workspace, then load it into the me3534_lab_gui.

Confirm that the GUI is displaying the correct controller

Collect closed-loop response data and save the results to a unique .mat file.

Task 2: Load the motor Model from Lab 3 (Post-Lab)

Load and **display** the continuous-time motor model from Lab 3. You will need to display the LTI object in the LiveScript in order to receive full credit. To do this, simply type the name of your model (e.g. **`Pmotor`**) into a LiveScript code block, and when the LiveScript is executed, Matlab will display the s-domain model in the LiveScript. Do NOT use the **`disp()`** function.

Do NOT regenerate your motor model from Lab 3 here! The model from Lab 3 must be saved as a .mat file and then that .mat file must be loaded into the Matlab workspace.

NOTE: If your motor model from Lab 3 is not correct or is not accurate enough, you will not be able to complete Lab 4! It is important to fix any errors you may have from the Lab 3 s-domain motor model **before** starting this lab.

```
load("motor_model.mat")
Pmotor = motor_sys;
Pmotor
```

```
Pmotor =

           0.05
  -------------------------
  0.0003229 s^2 + 0.01046 s

Continuous-time transfer function.
Model Properties
```

Task 3: Low-Bandwidth Proportional Controller (Post-Lab)

Construct Closed-Loop Models for Simulation

Construct closed-loop models for simulating the time response as well as the control signal

```
fs = 1000;      % sample rate (Hz)
ts = 1/fs;      % sample period (sec)
```

```
set(0,'defaultaxesfontsize',14);
set(0,'defaultlinelinewidth',2);
figpos = [0 0 1300 900];
format long g
%Above included just in case
open_loopL = Pmotor * ControlL;
T_sysL = minreal(open_loopL / (1 + open_loopL));
T_cmdL = minreal(ControlL / (1 + open_loopL));
```

Time Domain Analysis

Load the experimental closed-loop test data from Task 1a into the Matlab workspace.

```
load('Controls L4 DLow.mat');
t = t(:);
ref = ref(:);
cmd = cmd(:);
fbk = fbk(:);
```

Simulate the closed-loop time responses of the simulated feedback signal (fbk_sim) as well as the simulated control signal (cmd_sim):

```
fbk_simL = lsim(T_sysL, ref, t);
cmd_simL = lsim(T_cmdL, ref, t);
```

Generate a figure with three properly annotated subplots (grid lines, x-labels, y-labels, signal units, legend, etc.). The subplots must show the following:

Top subplot

- plot the experimental control signal vs. time
- plot the simulated control signal vs. time
- use the `yline` function to draw horizontal lines at -1.65V and +1.65V to indicate the min and max limits for the control voltage

Middle subplot

- plot the desired reference vs. time
- plot the experimental closed-loop feedback response vs. time
- plot the simulated closed-loop feedback response vs. time

Bottom subplot

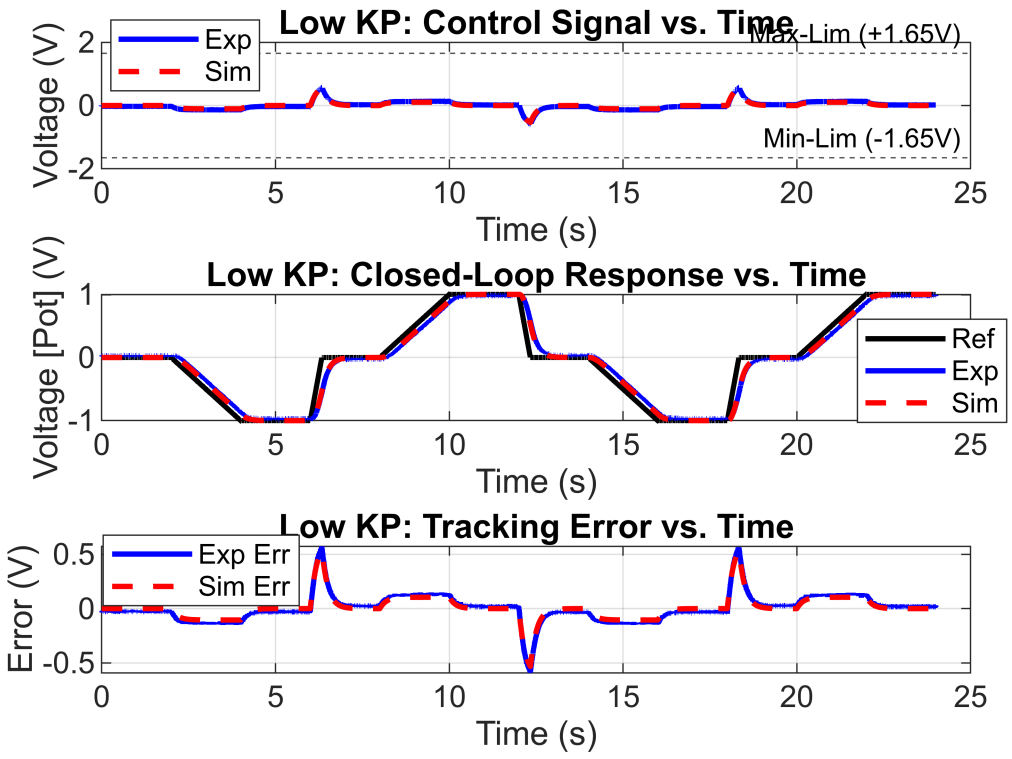
- plot the error between the reference and the experimental closed-loop feedback vs. time
- plot the error between the reference and the simulated closed-loop feedback vs. time

```
figure;
% Top subplot
subplot(3,1,1)
plot(t, cmd, 'b');
```

```
hold on
plot(t, cmd_simL, 'r--');
yline(1.65, 'k--', 'Max-Lim (+1.65V)');
yline(-1.65, 'k--', 'Min-Lim (-1.65V)');
grid on;
title('Low KP: Control Signal vs. Time');
xlabel('Time (s)');
ylabel('Voltage (V)');
legend(["Exp", "Sim"], "Position", [0.1376 0.8430 0.1339, 0.0821])
hold off

% Middle subplot
subplot(3,1,2)
plot(t, ref, 'k');
hold on
plot( t, fbk, 'b');
plot(t, fbk_simL, 'r--');
grid on;
title('Low KP: Closed-Loop Response vs. Time');
xlabel('Time (s)');
ylabel('Voltage [Pot] (V)');
legend(["Ref", "Exp", "Sim"], "Position", [0.8037 0.4496 0.1339, 0.1190])
hold off

% Bottom subplot
subplot(3,1,3)
err_expL = ref - fbk;
err_simL = ref - fbk_simL;
plot(t, err_expL, 'b');
hold on
plot(t, err_simL, 'r--');
grid on;
title('Low KP: Tracking Error vs. Time');
xlabel('Time (s)');
ylabel('Error (V)');
legend(["Exp Err", "Sim Err"], "Position", [0.1312 0.2309 0.1750, 0.0821])
hold off
```



Using the computed control signal from your closed-loop simulation, **estimate and display** the maximum absolute value of the control signal.

```
fprintf('Maximum absolute simulated control signal (low): %.4f V\n',
max(abs(cmd_simL)));
```

```
Maximum absolute simulated control signal (low): 0.5329 V
```

Using only the plot above, **estimate and display** the absolute value of the steady-state error during the "dwell" and "step" sections of the reference. The steady-state error should be the same for all dwell and step sections, so you only need to report a single estimated value.

$e_{ss} \approx$

Using only the plot above, **estimate and display** the absolute value of the steady-state error during the "ramp" sections of the reference. The absolute value of the steady-state error should be the same for all ramp sections, so you only need to report a single estimated value.

$e_{ss} =$

Use the "Position Error Constant" and the "Velocity Error Constant" from the lecture notes (main course) to **compute and display** the theoretical steady-state error for a step input and a linear ramp input.

```
visual_ess_stepL = 0;
visual_ess_rampL = 0.2;
fprintf('Estimated Absolute Steady-State Step Error: %.4f V\n', visual_ess_stepL);
```

```
Estimated Absolute Steady-State Step Error: 0.0000 V
```

```
fprintf('Estimated Absolute Steady-State Ramp Error: %.4f V\n', visual_ess_rampL);
```

```
Estimated Absolute Steady-State Ramp Error: 0.2000 V
```

```
[num_OL, den_OL] = tfdata(open_loopL, 'v');
Kv_L = num_OL(end) / den_OL(end-1);
fprintf('Theoretical Steady-State Error (Step): %.4f V\n', 0);
```

```
Theoretical Steady-State Error (Step): 0.0000 V
```

```
fprintf('Theoretical Steady-State Error (Ramp): %.4f V\n\n', 1/Kv_L);
```

```
Theoretical Steady-State Error (Ramp): 0.2092 V
```

Do these predicted steady-state error metrics match the steady-state errors shown in the time-domain error plot? Hint: They should match!

Use "Type Numbers" to explain these results for the step and ramp?

The type number of the given motor is 1 which can be found from the fact that its fundamental G(s) eqtion has one lone s variable. This means that the ess for dwell and step should be 1/1+Kp given that Kp approaches infinity for all type 1 systems all values of KP (not the same as Kp) should be zero or near zero given real world error can skew results slightly. That is what was observed

The ramp however is dependent on the velocity error constant Kv. Specifically the error is A/Kv indicating higher Kv = lower error. Kv is directly related to Kp as such it is expected that as is seen the Low Kp has the highest ramp error and the High Kp the lowest ramp error. This is also what was observed

Frequency Domain Analysis

Use the `logspace` function to generate a frequency vector covering 1000 frequencies from 0.01Hz to 100Hz. Use the `freqresp` function to compute the frequency response of your closed-loop transfer function over this frequency range. Generate a figure with two subplots:

Top subplot:

- use `semilogx` to plot the closed-loop frequency response phase (degrees) vs. log frequency (Hz)

Bottom subplot:

- use `semilogx` to plot the closed-loop frequency response magnitude (dB) vs. log frequency (Hz)
- use `yline` to draw a horizontal reference line at -3dB
- use xline to draw a vertical reference line at the frequency where the magnitude crosses -3dB. This is the closed-loop bandwidth.

Display your estimate of the closed-loop bandwidth.

```
f_hz = logspace(log10(0.01), log10(100), 1000);
w_rad = 2 * pi *f_hz;

%Compute
```

```
H_complexL = freqresp(T_sysL, w_rad);
H_complexL = squeeze(H_complexL);
magL = abs(H_complexL);
phaseL = angle(H_complexL) * 180/pi;
mag_dBL = 20*log10(magL);

figure;

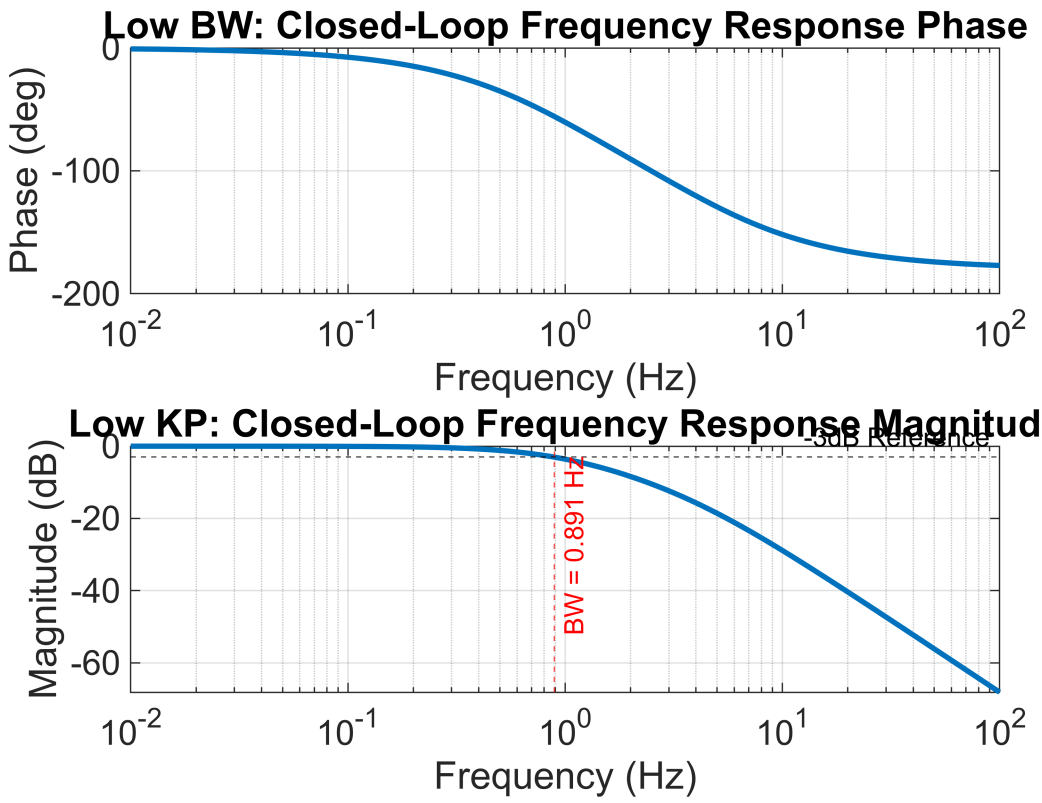
% Top subplot
subplot(2,1,1)
semilogx(f_hz, phaseL);
hold on
grid on;
title('Low BW: Closed-Loop Frequency Response Phase');
xlabel('Frequency (Hz)');
ylabel('Phase (deg)');
hold off

% Bottom subplot
subplot(2,1,2)
semilogx(f_hz, mag_dBL);
hold on
grid on;
yline(-3, 'k--', '-3dB Reference');

% Calculate and plot bandwidth
bw_idxL = find(mag_dBL <= -3, 1);
if ~isempty(bw_idxL)
    bw_valL = f_hz(bw_idxL);
    xline(bw_valL, 'r--', sprintf('BW = %.3f Hz', bw_valL));
    fprintf('Estimated closed-loop bandwidth Low-KP: %.4f Hz\n\n', bw_valL);
end
```

```
Estimated closed-loop bandwidth Low-KP: 0.8911 Hz
```

```
grid on;
title('Low KP: Closed-Loop Frequency Response Magnitude');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
hold off
```



Task 4: Low-Bandwidth Proportional Controller (Post-Lab)

Construct Closed-Loop Models for Simulation

Construct closed-loop models for simulating the time response as well as the control signal

```
fs = 1000;      % sample rate (Hz)
ts = 1/fs;      % sample period (sec)

set(0,'defaultaxesfontsize',14);
set(0,'defaultlinelinewidth',2);
figpos = [0 0 1300 900];
format long g
%Above included just in case
open_loopM = Pmotor * ControlM;
T_sysM = minreal(open_loopM / (1 + open_loopM));
T_cmdM = minreal(ControlM / (1 + open_loopM));
```

Time Domain Analysis

Load the experimental closed-loop test data from Task 1a into the Matlab workspace.

```
load('Controls L4 DMed.mat');
t = t(:);
ref = ref(:);
cmd = cmd(:);
```

```
fbk = fbk(:);
```

Simulate the closed-loop time responses of the simulated feedback signal (fbk_sim) as well as the simulated control signal (cmd_sim):

```
fbk_simM = lsim(T_sysM, ref, t);
cmd_simM = lsim(T_cmdM, ref, t);
```

Generate a figure with three properly annotated subplots (grid lines, x-labels, y-labels, signal units, legend, etc.). The subplots must show the following:

Top subplot

- plot the experimental control signal vs. time
- plot the simulated control signal vs. time
- use the `yline` function to draw horizontal lines at -1.65V and +1.65V to indicate the min and max limits for the control voltage

Middle subplot

- plot the desired reference vs. time
- plot the experimental closed-loop feedback response vs. time
- plot the simulated closed-loop feedback response vs. time

Bottom subplot

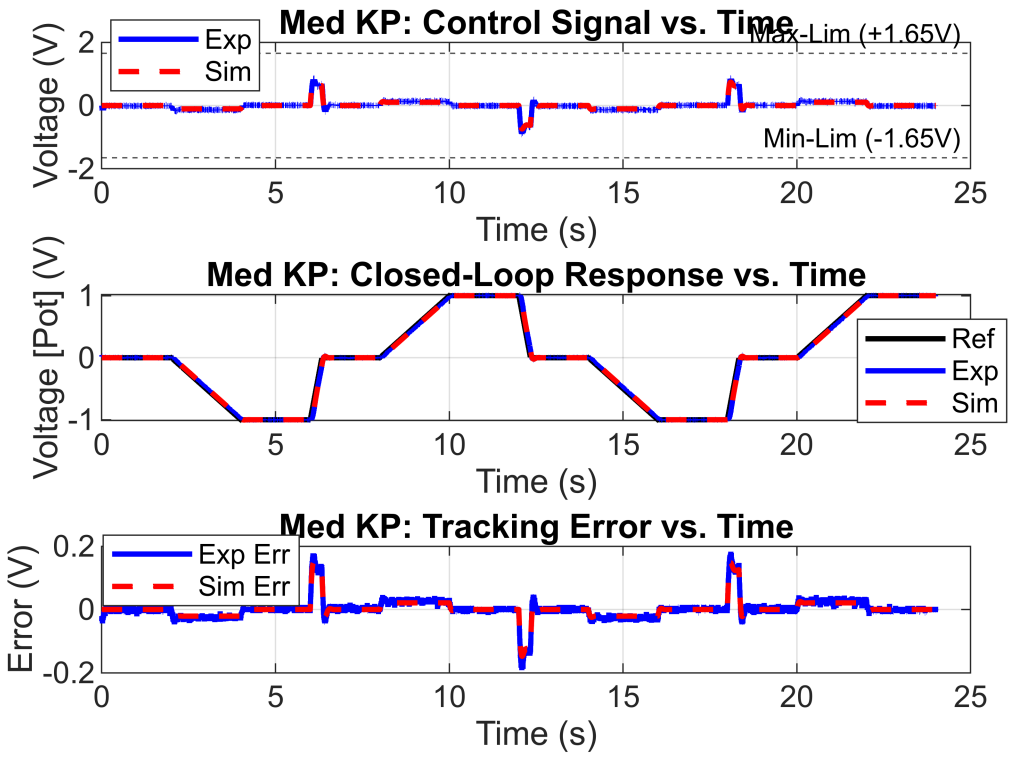
- plot the error between the reference and the experimental closed-loop feedback vs. time
- plot the error between the reference and the simulated closed-loop feedback vs. time

```
figure;
% Top subplot
subplot(3,1,1)
plot(t, cmd, 'b');
hold on
plot(t, cmd_simM, 'r--');
yline(1.65, 'k--', 'Max-Lim (+1.65V)');
yline(-1.65, 'k--', 'Min-Lim (-1.65V)');
grid on;
title('Med KP: Control Signal vs. Time');
xlabel('Time (s)');
ylabel('Voltage (V)');
legend(["Exp", "Sim"], "Position", [0.1376 0.8430 0.1339, 0.0821])
hold off

% Middle subplot
subplot(3,1,2)
plot(t, ref, 'k');
hold on
plot(t, fbk, 'b');
```

```
plot(t, fbk_simM, 'r--');
grid on;
title('Med KP: Closed-Loop Response vs. Time');
xlabel('Time (s)');
ylabel('Voltage [Pot] (V)');
legend(["Ref", "Exp", "Sim"], "Position", [0.8037 0.4496 0.1339, 0.1190])
hold off

% Bottom subplot
subplot(3,1,3)
err_expM = ref - fbk;
err_simM = ref - fbk_simM;
plot(t, err_expM, 'b');
hold on
plot(t, err_simM, 'r--');
grid on;
title('Med KP: Tracking Error vs. Time');
xlabel('Time (s)');
ylabel('Error (V)');
legend(["Exp Err", "Sim Err"], "Position", [0.1312 0.2309 0.1750, 0.0821])
hold off
```



Using the computed control signal from your closed-loop simulation, **estimate and display** the maximum absolute value of the control signal.

```
fprintf('Maximum absolute simulated control signal (Med): %.4f V\n',
max(abs(cmd_simM)));
```

```
Maximum absolute simulated control signal (Med): 0.7398 V
```

Using only the plot above, **estimate and display** the absolute value of the steady-state error during the "dwell" and "step" sections of the reference. The steady-state error should be the same for all dwell and step sections, so you only need to report a single estimated value.

$e_{ss} \approx$

Using only the plot above, **estimate and display** the absolute value of the steady-state error during the "ramp" sections of the reference. The absolute value of the steady-state error should be the same for all ramp sections, so you only need to report a single estimated value.

$e_{ss} =$

Use the "Position Error Constant" and the "Velocity Error Constant" from the lecture notes (main course) to **compute and display** the theoretical steady-state error for a step input and a linear ramp input.

```
visual_ess_stepM = 0;
visual_ess_rampM = 0.04;
fprintf('Estimated Absolute Steady-State Step Error (Med): %.4f V\n',
visual_ess_stepM);
```

```
Estimated Absolute Steady-State Step Error (Med): 0.0000 V
```

```
fprintf('Estimated Absolute Steady-State Ramp Error (Med): %.4f V\n',
visual_ess_rampM);
```

```
Estimated Absolute Steady-State Ramp Error (Med): 0.0400 V
```

```


[num_OLM, den_OLM] = tfdata(open_loopM, 'v');
Kv_M = num_OLM(end) / den_OLM(end-1);
fprintf('Theoretical Steady-State Error (Step): %.4f V\n', 0);
```

```
Theoretical Steady-State Error (Step): 0.0000 V
```

```
fprintf('Theoretical Steady-State Error (Ramp): %.4f V\n\n', 1/Kv_M);
```

```
Theoretical Steady-State Error (Ramp): 0.0418 V
```

Do these predicted steady-state error metrics match the steady-state errors shown in the time-domain error plot? Hint: They should match!

Use "Type Numbers" to explain these results for the step and ramp?

The type number of the given motor is 1 which can be found from the fact that its fundamental G(s) eqtion has one lone s variable. This means that the ess for dwell and step should be 1/1+Kp given that Kp approaches infinity for all type 1 systems all values of KP (not the same as Kp) should be zero or near zero given real world error can skew results slightly. That is what was observed

The ramp however is dependent on the velocity error constant Kv. Specifically the error is A/Kv indicating higher Kv = lower error. Kv is directly related to Kp as such it is expected that as is seen the Low Kp has the highest ramp error and the High Kp the lowest ramp error. This is also what was observed

Frequency Domain Analysis

Use the `logspace` function to generate a frequency vector covering 1000 frequencies from 0.01Hz to 100Hz. Use the `freqresp` function to compute the frequency response of your closed-loop transfer function over this frequency range. Generate a figure with two subplots:

Top subplot:

- use `semilogx` to plot the closed-loop frequency response phase (degrees) vs. log frequency (Hz)

Bottom subplot:

- use `semilogx` to plot the closed-loop frequency response magnitude (dB) vs. log frequency (Hz)
- use `yline` to draw a horizontal reference line at -3dB
- use xline to draw a vertical reference line at the frequency where the magnitude crosses -3dB. This is the closed-loop bandwidth.

Display your estimate of the closed-loop bandwidth.

```
f_hz = logspace(log10(0.01), log10(100), 1000);
w_rad = 2 * pi * f_hz;

H_complexM = freqresp(T_sysM, w_rad);
H_complexM = squeeze(H_complexM);
magM = abs(H_complexM);
phaseM = angle(H_complexM) * 180/pi;
mag_dBM = 20*log10(magM);

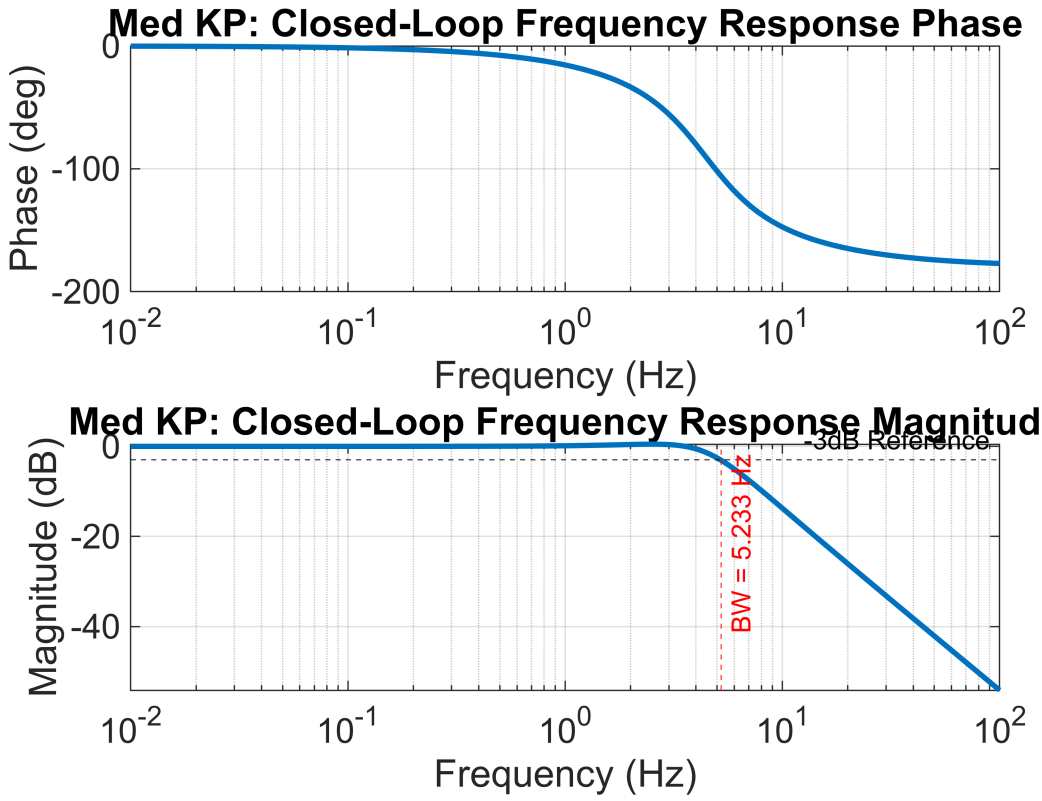
figure;
% Top subplot
subplot(2,1,1)
semilogx(f_hz, phaseM);
hold on
grid on;
title('Med KP: Closed-Loop Frequency Response Phase');
xlabel('Frequency (Hz)');
ylabel('Phase (deg)');
hold off

% Bottom subplot
subplot(2,1,2)
semilogx(f_hz, mag_dBM);
hold on
grid on;
yline(-3, 'k--', '-3dB Reference');
```

```
bw_idxM = find(mag_dBM <= -3, 1);
if ~isempty(bw_idxM)
    bw_valM = f_hz(bw_idxM);
    xline(bw_valM, 'r--', sprintf('BW = %.3f Hz', bw_valM));
    fprintf('Estimated closed-loop bandwidth Med-KP: %.4f Hz\n\n', bw_valM);
end
```

```
Estimated closed-loop bandwidth Med-KP: 5.2326 Hz
```

```
grid on;
title('Med KP: Closed-Loop Frequency Response Magnitude');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
hold off
```



Task 5: High-Bandwidth Proportional Controller (Post-Lab)

Construct Closed-Loop Models for Simulation

Construct closed-loop models for simulating the time response as well as the control signal

```
fs = 1000;      % sample rate (Hz)
ts = 1/fs;      % sample period (sec)

set(0,'defaultaxesfontsize',14);
set(0,'defaultlinelinewidth',2);
figpos = [0 0 1300 900];
```

```
format long g
%Above included just in case
open_loopH = Pmotor * ControlH;
T_sysH = minreal(open_loopH / (1 + open_loopH));
T_cmdH = minreal(ControlH / (1 + open_loopH));
```

Time Domain Analysis

Load the experimental closed-loop test data from Task 1a into the Matlab workspace.

```
load('Controls L4 DHigh.mat');
t = t(:);
ref = ref(:);
cmd = cmd(:);
fbk = fbk(:);
```

Simulate the closed-loop time responses of the simulated feedback signal (fbk_sim) as well as the simulated control signal (cmd_sim):

```
fbk_simH = lsim(T_sysH, ref, t);
cmd_simH = lsim(T_cmdH, ref, t);
```

Generate a figure with three properly annotated subplots (grid lines, x-labels, y-labels, signal units, legend, etc.). The subplots must show the following:

Top subplot

- plot the experimental control signal vs. time
- plot the simulated control signal vs. time
- use the `yline` function to draw horizontal lines at -1.65V and +1.65V to indicate the min and max limits for the control voltage

Middle subplot

- plot the desired reference vs. time
- plot the experimental closed-loop feedback response vs. time
- plot the simulated closed-loop feedback response vs. time

Bottom subplot

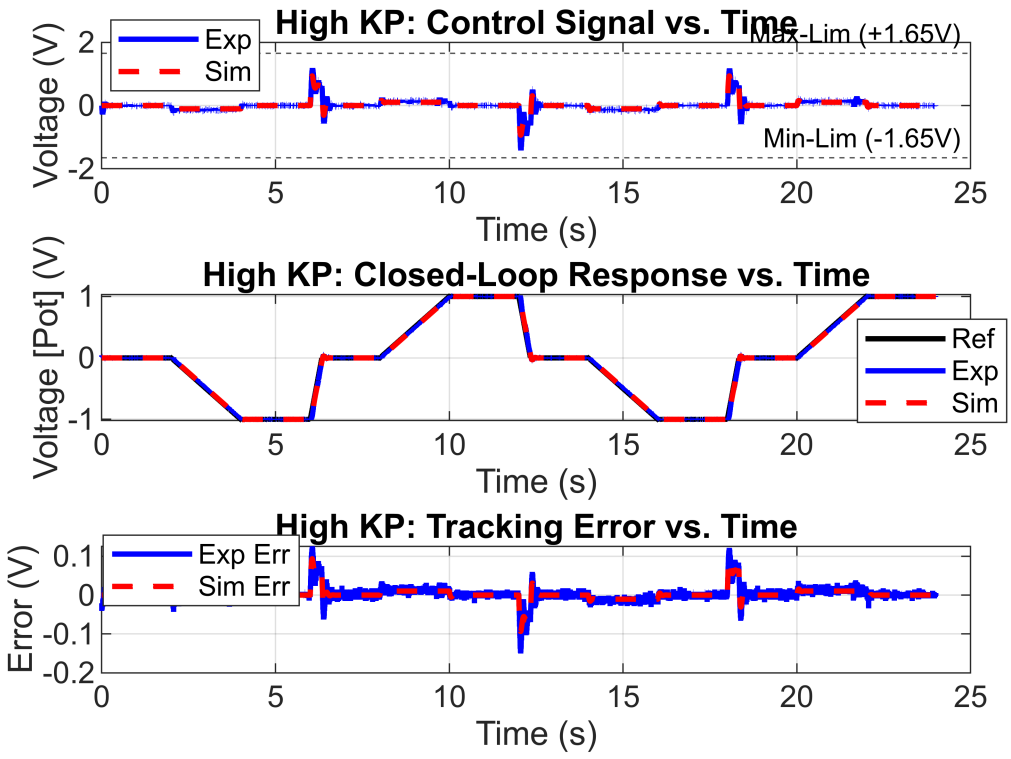
- plot the error between the reference and the experimental closed-loop feedback vs. time
- plot the error between the reference and the simulated closed-loop feedback vs. time

```
figure;
% Top subplot
subplot(3,1,1)
plot(t, cmd, 'b');
hold on
plot(t, cmd_simH, 'r--');
yline(1.65, 'k--', 'Max-Lim (+1.65V)');
```

```
yline(-1.65, 'k--', 'Min-Lim (-1.65V)');
grid on;
title('High KP: Control Signal vs. Time');
xlabel('Time (s)');
ylabel('Voltage (V)');
legend(["Exp", "Sim"], "Position", [0.1376 0.8430 0.1339, 0.0821])
hold off

% Middle subplot
subplot(3,1,2)
plot(t, ref, 'k');
hold on
plot(t, fbk, 'b');
plot(t, fbk_simH, 'r--');
grid on;
title('High KP: Closed-Loop Response vs. Time');
xlabel('Time (s)');
ylabel('Voltage [Pot] (V)');
legend(["Ref", "Exp", "Sim"], "Position", [0.8037 0.4496 0.1339, 0.1190])
hold off

% Bottom subplot
subplot(3,1,3)
err_expH = ref - fbk;
err_simH = ref - fbk_simH;
plot(t, err_expH, 'b');
hold on
plot(t, err_simH, 'r--');
grid on;
title('High KP: Tracking Error vs. Time');
xlabel('Time (s)');
ylabel('Error (V)');
legend(["Exp Err", "Sim Err"], "Position", [0.1312 0.2309 0.1750, 0.0821])
hold off
```



Using the computed control signal from your closed-loop simulation, **estimate and display** the maximum absolute value of the control signal.

```
fprintf('Maximum absolute simulated control signal (High): %.4f V\n',
max(abs(cmd_simH)));
```

```
Maximum absolute simulated control signal (High): 0.9372 V
```

Using only the plot above, **estimate and display** the absolute value of the steady-state error during the "dwell" and "step" sections of the reference. The steady-state error should be the same for all dwell and step sections, so you only need to report a single estimated value.

$e_{ss} \approx$

Using only the plot above, **estimate and display** the absolute value of the steady-state error during the "ramp" sections of the reference. The absolute value of the steady-state error should be the same for all ramp sections, so you only need to report a single estimated value.

$e_{ss} =$

Use the "Position Error Constant" and the "Velocity Error Constant" from the lecture notes (main course) to **compute and display** the theoretical steady-state error for a step input and a linear ramp input.

```
visual_ess_stepH = 0;
visual_ess_rampH = 0.02;
fprintf('Estimated Absolute Steady-State Step Error (High): %.4f V\n',
visual_ess_stepH);
```

```
Estimated Absolute Steady-State Step Error (High): 0.0000 V
```

```
fprintf('Estimated Absolute Steady-State Ramp Error (High): %.4f V\n',
visual_ess_rampH);
```

```
Estimated Absolute Steady-State Ramp Error (High): 0.0200 V
```

```
[num_OLH, den_OLH] = tfdata(open_loopH, 'v');
Kv_H = num_OLH(end) / den_OLH(end-1);
fprintf('Theoretical Steady-State Error (Step): %.4f V\n', 0);
```

```
Theoretical Steady-State Error (Step): 0.0000 V
```

```
fprintf('Theoretical Steady-State Error (Ramp): %.4f V\n\n', 1/Kv_H);
```

```
Theoretical Steady-State Error (Ramp): 0.0209 V
```

Do these predicted steady-state error metrics match the steady-state errors shown in the time-domain error plot? Hint: They should match!

Use "Type Numbers" to explain these results for the step and ramp?

The type number of the given motor is 1 which can be found from the fact that its fundamental G(s) eqtion has one lone s variable. This means that the ess for dwell and step should be 1/1+Kp given that Kp approaches infinity for all type 1 systems all values of KP (not the same as Kp) should be zero or near zero given real world error can skew results slightly. That is what was observed

The ramp however is dependent on the velocity error constant Kv. Specifically the error is A/Kv indicating higher Kv = lower error. Kv is directly related to Kp as such it is expected that as is seen the Low Kp has the highest ramp error and the High Kp the lowest ramp error. This is also what was observed

Frequency Domain Analysis

Use the `logspace` function to generate a frequency vector covering 1000 frequencies from 0.01Hz to 100Hz. Use the `freqresp` function to compute the frequency response of your closed-loop transfer function over this frequency range. Generate a figure with two subplots:

Top subplot:

- use `semilogx` to plot the closed-loop frequency response phase (degrees) vs. log frequency (Hz)

Bottom subplot:

- use `semilogx` to plot the closed-loop frequency response magnitude (dB) vs. log frequency (Hz)
- use `yline` to draw a horizontal reference line at -3dB
- use xline to draw a vertical reference line at the frequency where the magnitude crosses -3dB. This is the closed-loop bandwidth.

Display your estimate of the closed-loop bandwidth.

```
f_hz = logspace(log10(0.01), log10(100), 1000);
w_rad = 2 * pi * f_hz;
```

```
H_complexH = freqresp(T_sysH, w_rad);
H_complexH = squeeze(H_complexH);
magH = abs(H_complexH);
phaseH = angle(H_complexH) * 180/pi;
mag_dBH = 20*log10(magH);

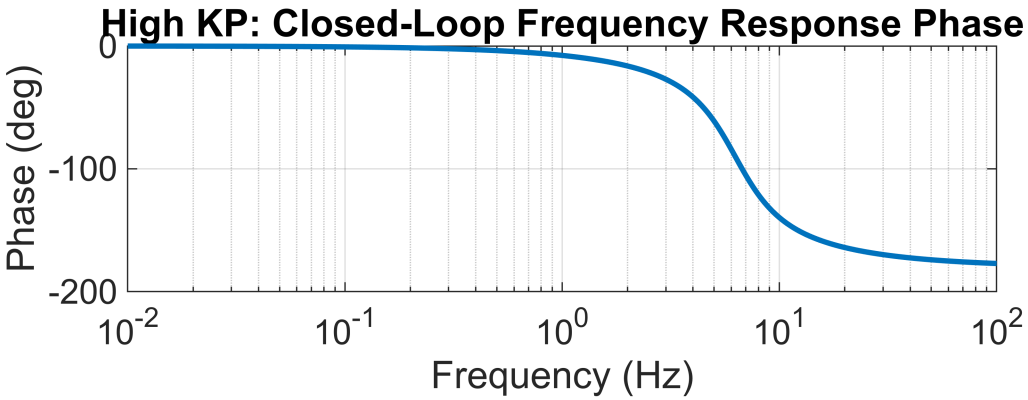
figure;
% Top subplot
subplot(2,1,1)
semilogx(f_hz, phaseH);
hold on
grid on;
title('High KP: Closed-Loop Frequency Response Phase');
xlabel('Frequency (Hz)');
ylabel('Phase (deg)');
hold off

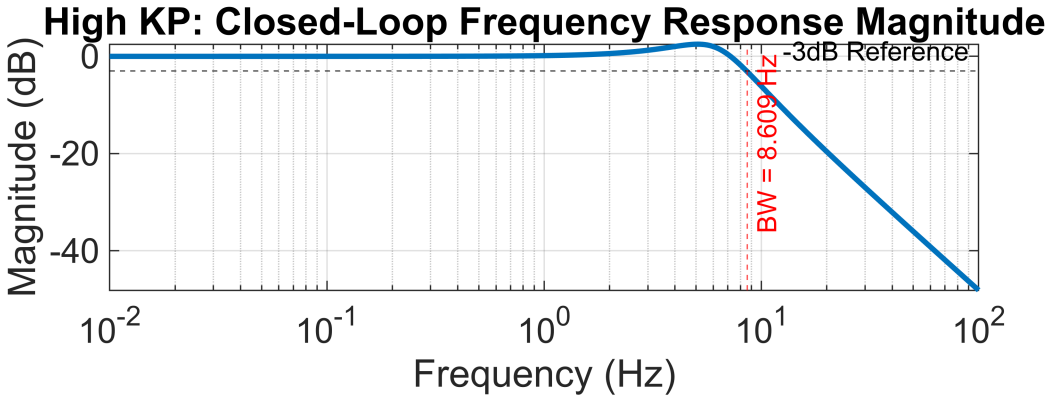
% Bottom subplot
subplot(2,1,2)
semilogx(f_hz, mag_dBH);
hold on
grid on;
yline(-3, 'k--', '-3dB Reference');

bw_idxH = find(mag_dBH <= -3, 1);
if ~isempty(bw_idxH)
    bw_valH = f_hz(bw_idxH);
    xline(bw_valH, 'r--', sprintf('BW = %.3f Hz', bw_valH));
    fprintf('Estimated closed-loop bandwidth High-KP: %.4f Hz\n\n', bw_valH);
end
```

```
Estimated closed-loop bandwidth High-KP: 8.6086 Hz
```

```
grid on;
title('High KP: Closed-Loop Frequency Response Magnitude');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
hold off
```


High KP: Closed-Loop Frequency Response Phase
Phase (deg)
0
-100
-200
10-2
10-1
100
101
102
Frequency (Hz)


High KP: Closed-Loop Frequency Response Magnitude
Magnitude (dB)
0
-20
-40
-3dB Reference
BW = 8.609 Hz
10-2
10-1
100
101
102
Frequency (Hz)