ME3534 - Lab 2

Post-Lab Processing and Report

Name: Zachary Rohloff

Date: 3/2/2026

Lab Bench #: 9

This LiveScript contains the instructions for the Lab 2 post-lab processing and final report. After you have collected data in the Lab following the detailed instructions in the lab2_instructions.pdf document, you will add your own Matlab code in the lab2_report.mlx document to process the lab data at home. Do NOT attempt to complete the post-lab processing during your lab session. Finally, you must convert this LiveScript to a PDF for submission to Canvas.

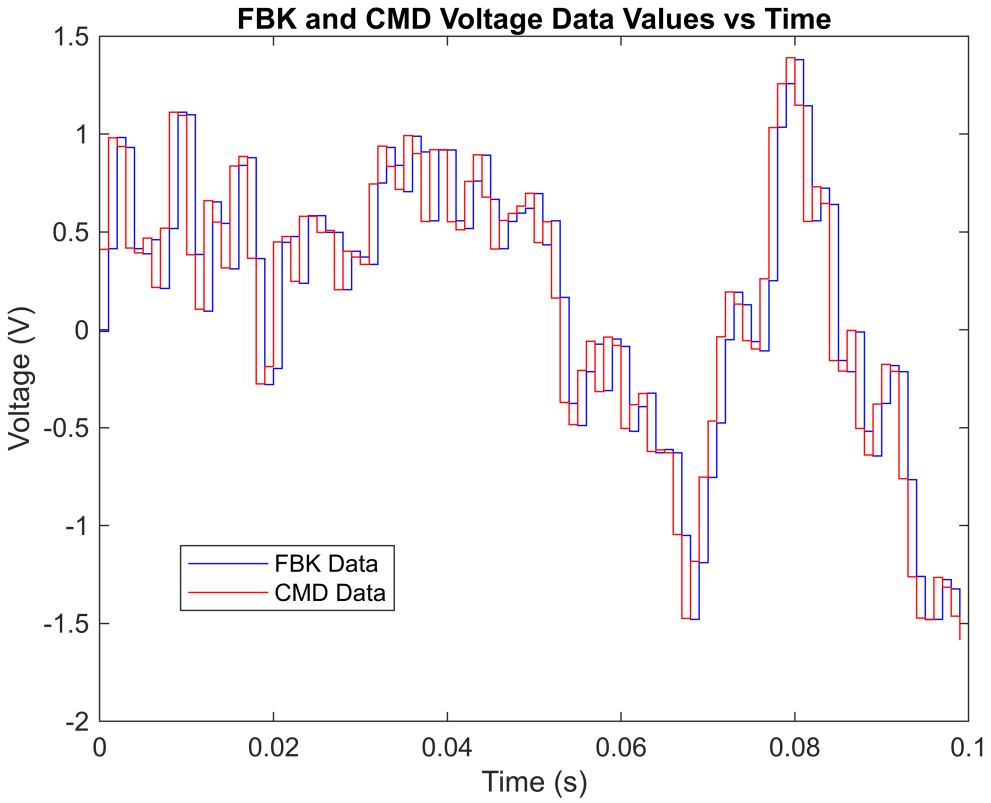
Activities to Complete Outside of the Lab

Task 2b: Demonstrate Sampling Delay

Write Matlab code in the space provided below to do each of the following:

1. Load the saved data from Task 2a into this LiveScript. If you look at the Workspace browser in Matlab after loading the .mat file, you should see four arrays: `t, ref, fbk,` and `cmd`.
2. Generate a plot of fbk vs time and cmd vs time on the same axes. Use the Matlab STAIRS() function (instead of PLOT) to generate a stair-step style plot which will more easily demonstrate the delay. Make sure you include a grid, axis labels, and a legend.
3. This plot must clearly show that the FBK signal is delayed by one sample from the CMD signal.

```
clear;
clc;
load("Controls L2 Data1.mat")
stairs(t,fbk,"b")
hold on
stairs(t,cmd,'r')
xlabel('Time (s)')
ylabel ('Voltage (V)')
legend(["FBK Data", "CMD Data"], "Position", [0.2010 0.2420 0.1929, 0.0774])
title('FBK and CMD Voltage Data Values vs Time')
hold off
```



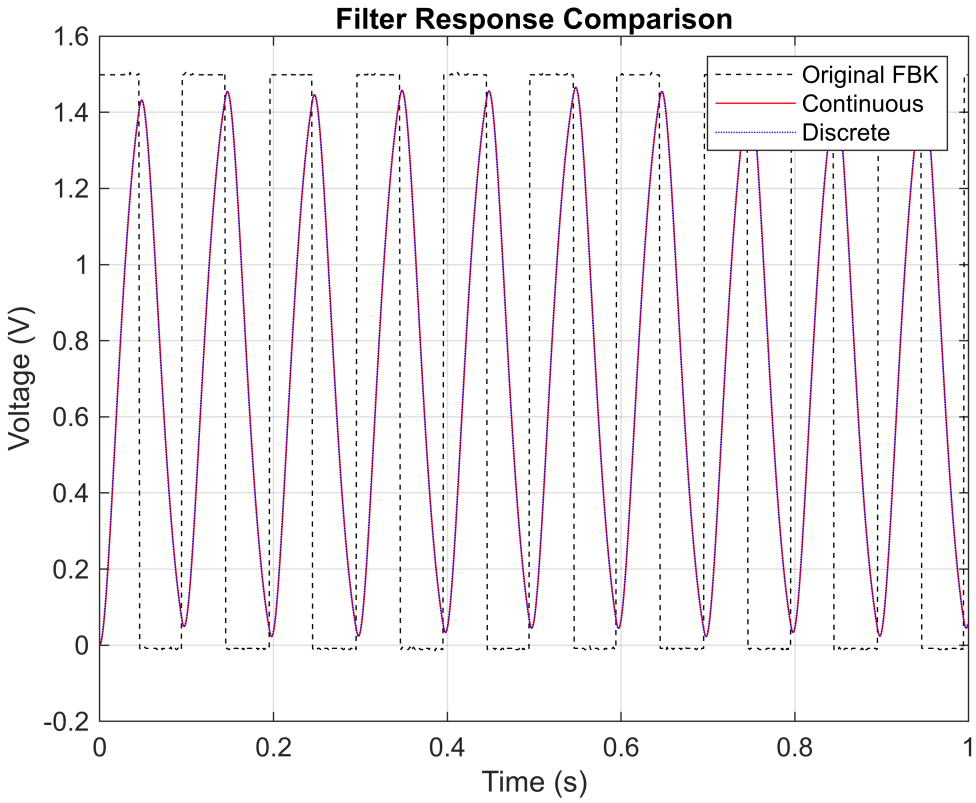
Task 3b: Demonstrate Digital Low-Pass Filtering

Write Matlab code in the space provided below to do each of the following:

1. Generate a continuous-time LTI transfer function object for the following 2nd order "low-pass filter" dynamic system: $\left(\frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}\right)$, where $\zeta = 0.707$, and $\omega\left[\frac{\text{rad}}{s}\right] = 2\pi \times 10[\text{Hz}]$.
2. Load the saved data from Task 3a into this LiveScript
3. Use the `LSIM()` function to simulate the continuous-time response of your LTI transfer function using the fbk data as input to the filter
4. Use the `C2D()` function with the 'Tustin' bilinear transform option to convert the continuous-time LTI object to a discrete-time LTI object. Make sure you specify the sample period associated with a 1000 Hz sample rate!
5. Display both the continuous-time and the discrete-time LTI objects in this LiveScript. To do this, just type the name of the LTI object without a semi-colon at the end. The basic information for the LTI object will automatically display in the LiveScript.
6. Use the `TFDATA()` function to extract the numerator and denominator coefficient arrays from the discrete-time LTI object.
7. Use the `FILTER()` function with the numerator and denominator arrays to simulate the time response of the discrete-time filter using the same fbk signal as the input. Note that `FILTER` is one of several ways to simulate discrete-time transfer functions expressed as numerator and denominator coefficient arrays.

8. Generate a plot of the entire buffer of fbk vs time, the continuous-time filter response vs. time, and the discrete-time filter response vs. time on the same axis. Make sure you include a grid, axis labels, and a legend. If you did everything correctly, both the filter responses should be on top of each other in the plot.
9. Use your knowledge of frequency responses from the Main Course to explain why the steady-state filter time responses both have a phase shift of 90 degrees relative to the square wave fudamental frequency

```
clear;
clc;
load("Controls L2 Data2.mat")
zeta = 0.707;
wn = 2*pi*10;
num = [wn^2];
den = [1, 2*zeta*wn, wn^2];
sys_c = tf(num, den);
y_continuous = lsim(sys_c, fbk, t);
fs = 1000;
Ts = 1/fs;
sys_d = c2d(sys_c, Ts, 'tustin');
[b, a] = tfdata(sys_d,'v');
y_discrete = filter(b, a, fbk);
%plot
plot(t, fbk, 'k--');
hold on;
plot(t, y_continuous, 'r');
plot(t, y_discrete, 'b:');
grid on;
xlabel('Time (s)');
ylabel('Voltage (V)');
legend('Original FBK','Continuous','Discrete');
title('Filter Response Comparison');
hold off
```



Answer to 9. The phaze shift of 90 degrees can be explained by the removal of the higher order portions of the square wave frequency which creates a more rounded graph as the fourie transform responcible for modling the square wave tries to "catch up" to the proper target voltage and ends up with an almost sinusoidal graph

Task 4b: Demonstrate digital Notch Filtering

Write Matlab code in the space provided below to do each of the following:

1. Generate a continuous-time LTI transfer function for the following 2nd order "Notch filter" dynamic system: $\left(\frac{s^2 + 2\zeta_z \omega s + \omega^2}{s^2 + 2\zeta_p \omega s + \omega^2}\right)$, where $\zeta_z = 0.1$, $\zeta_p = 50\zeta_z$, and $\omega\left[\frac{\text{rad}}{s}\right] = 2\pi \times 5[\text{Hz}]$.
2. Load the saved data from Task 4a into this LiveScript
3. Use the `LSIM()` function to simulate the continuous-time response of this LTI transfer function using the fbk data as input to the filter
4. Use the `C2D()` function with the 'Tustin' bilinear transform option to convert the continuous-time LTI object to a discrete-time LTI object. Make sure you specify the sample period associated with a 1000 Hz sample rate!
5. Display both the continuous-time and the discrete-time LTI objects in this LiveScript. To do this, just type the name of the LTI object without a semi-colon at the end
6. Use the `TFDATA()` function to extract the numerator and denominator coefficient arrays from the discrete-time LTI object.

7. Use the `FILTER()` function with the numerator and denominator arrays to simulate the time response of the discrete-time filter using the same fbk signal as the input
8. Generate a plot of the entire buffer of fbk vs time, the continuous-time filter response, and the discrete-time filter response on the same axis. Make sure you include a grid, axis labels, and a legend. If you did everything correctly, both the filter responses should be on top of each other in the plot.
9. Verify that the 5 Hz sinusoidal frequency is not present in the filtered response, i.e. it has been removed by the notch filter. Describe what residual signal is remaining after the initial transient is gone (after about 0.1 seconds). Hint: The residual is NOT "noise", and it is NOT part of the original 5 Hz waveform unless you made a mistake in the filter implementation!

```
clear;
clc;
load("Controls L2 Data3.mat")
zeta_z = 0.1;
zeta_p = 50 * zeta_z;
wn = 2 * pi * 5;
num_n = [1, 2*zeta_z*wn, wn^2];
den_n = [1, 2*zeta_p*wn, wn^2];
sys_notch = tf(num_n, den_n);
y_cont = lsim(sys_notch,fbk,t);
fs = 1000;
Ts = 1/fs;
sys_disc = c2d(sys_notch,Ts,'tustin');
sys_notch
```

```
sys_notch =

  s^2 + 6.283 s + 987
  -------------------
  s^2 + 314.2 s + 987

Continuous-time transfer function.
Model Properties
```

```
sys_disc
```

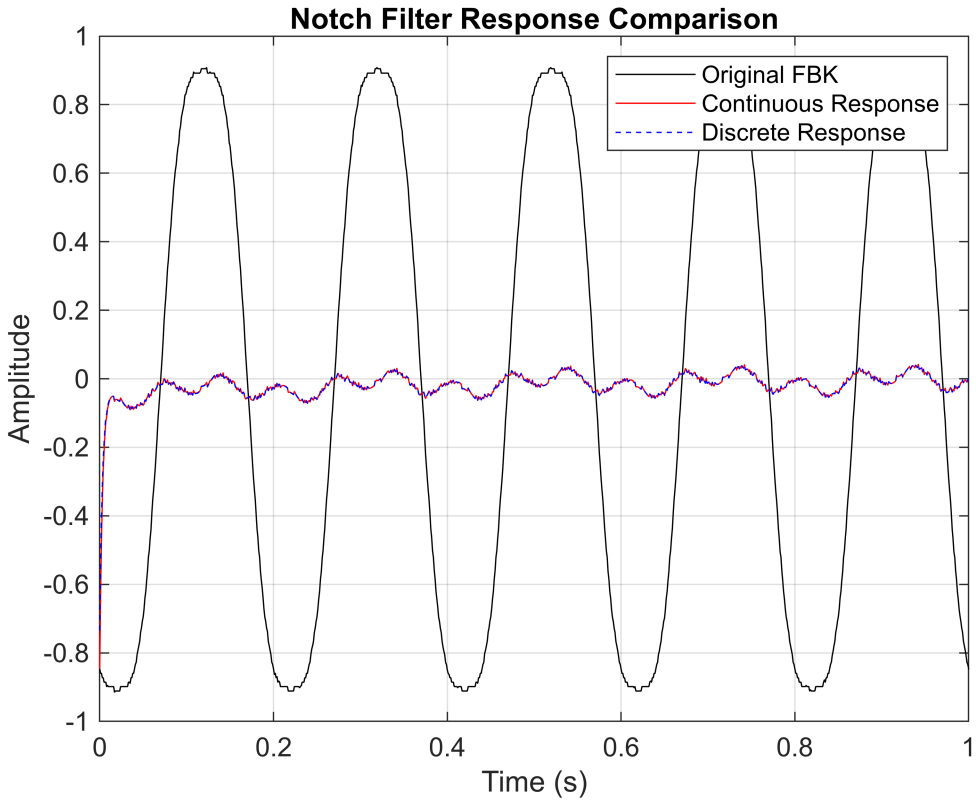
```
sys_disc =

  0.867 z^2 - 1.728 z + 0.8616
  ----------------------------
     z^2 - 1.728 z + 0.7285

Sample time: 0.001 seconds
Discrete-time transfer function.
Model Properties
```

```
[num_d, den_d] = tfdata(sys_disc,'v');
y_disc = filter(num_d, den_d, fbk);
%plot
plot(t, fbk, 'k-');
hold on;
plot(t, y_cont, 'r-');
```

```
plot(t, y_disc, 'b--');
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
title('Notch Filter Response Comparison');
xlim([0,1])
legend('Original FBK', 'Continuous Response','Discrete Response');
```



Answer to 9. The original 5Hz signal is definetly not seen in the responce/output. After the initial transient period is done the signal seems to be almost sinusoidal. Specifically it oscilates with a low peak, high middle peak, high peak, low middle peak, then back to a low peak. This to me signals that mutiple sinusoids are controling the output.