1. Description

The purpose of this lab was to code and implement a discrete approximation to a continuous SISO time-invariant system. The continuous system used as a basis was a third order low-pass filter using analog inputs and outputs. This approximation was coded using a timer based interrupt and a biquad cascade. Discrete system response was then compared to theoretical continuous response for step and frequency inputs.

Hierarchy:

```
main()
|_MyRio_Open()                          // Opens a session with the MyRio
|_AIO_initialize()                      // Initializing Analog Input/Output
|_Aio_Write()                           // Further initialization
|_Irq_RegisterTimerIrq()                // Register the interrupt
|_pthread_create()                      // Create a new thread for the interrupt
|       |_Timer_Irq_Thread()  (argument) // Interrupt service function
|_while()                               // Main loop
|       |_getkey()                      // Loop will continue until DEL key pressed
|_printf_lcd()                          // Loop termination message on LCD
|_pthread_join()                        // Part of the interrupt termination sequence
|_Irq_UnregisterDiIrq()                 // Interrupt unregistered
|_openmatfile()                         // creates MATLAB file
|_matfile_addstring()                   // adds a name string to the MATLAB file
|_matfile_addmatrix()                   // adds matrix of values to the MATLAB file
|_matfile_close()                       // completes/saves MATLAB file
|_MyRio_Close()                         // Closes the session with the MyRio

Timer_Irq_Thread()
|_while()                               // Interrupt until thread stopped by main
|       |_Irq_Wait()                    // wait for IRQ to assert or time out
|       |_NiFpga_WriteU32()             // Scheduling next interrupt
|       |_NiFpga_WriteBool()            // More timer interrupt parameter setting
|       |_if()                          // if the IRQ is asserted
|               |_cascade()             // call cascade and process input/output
|               |_Aio_Write()           // transmit y0 to analog output
|               |_Irq_Acknowledge()     // Interrupt acknowledged to the scheduler
|_pthread_exit()                        // Terminate the new thread

cascade()
|_for()                                 // difference eq used to calculate y0
```

2. Testing

Testing involved subjecting the system to a step input and various frequency inputs to determine how closely the discrete implementation matched the theoretical continuous model. The step input test was conducted by feeding a square wave from a function generator into the system. Discrete system output data was saved into a MATLAB file for comparison against the system's theoretical step response. The system's frequency response was measured by feeding sine waves of various frequencies into the system and recording the output's magnitude and phase shift. These values were then compared against the continuous system's bode magnitude and phase plots.

3. Results

The discrete system step response was very close to the theoretical continuous step response, as seen in Figure 2. Notably, the rise time, overshoot, settling time, and peak voltage are all very close. Figure 1 also represents the expected step response from a relatively underdamped dynamic system.

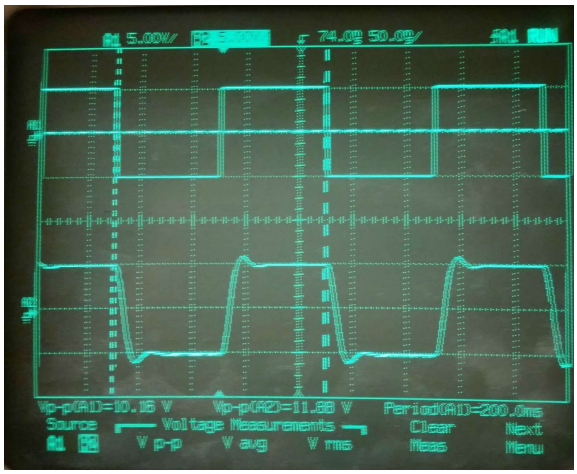

Figure 1: Step Response Input (Top) and Output (Bottom)

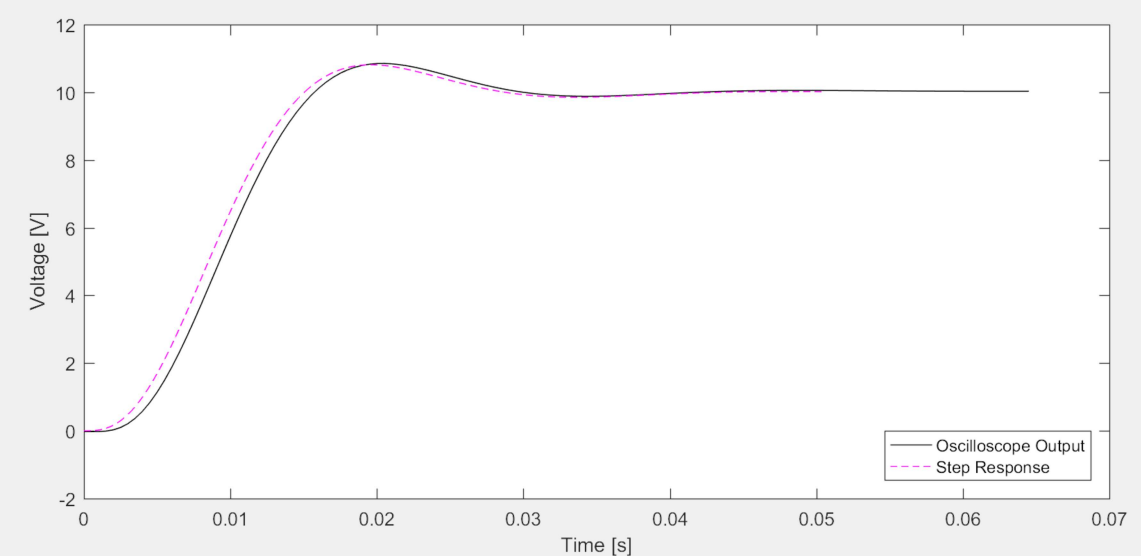

Figure 2: Theoretical and Experimental Step Response

Sine waves with frequencies of 5, 10, 20, 40, 60, 100, 140, and 200 Hertz were input into the system and the resulting output magnitude and phase were recorded. These points were then plotted against the theoretical response magnitude and phase as seen in Figures 3 and 4. The discrete filter attenuated the higher frequency input signals as expected and the reduction in output magnitude is essentially equal to that of the theoretical continuous time filter. The phase of the discrete filter was generally lower than the theoretical filter, but this could also be due to human and measurement error. The oscilloscope output became fairly granular as the frequency increased and the accuracy of the later phase shifts could be inaccurate due to the increased difficulty in selecting the wave peak.

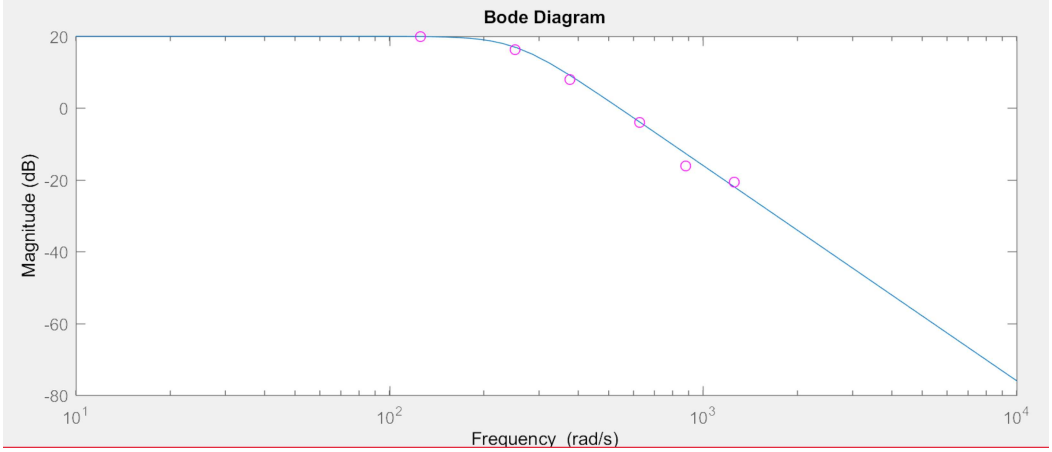

Figure 3: Theoretical and Experimental Frequency Response: Magnitude

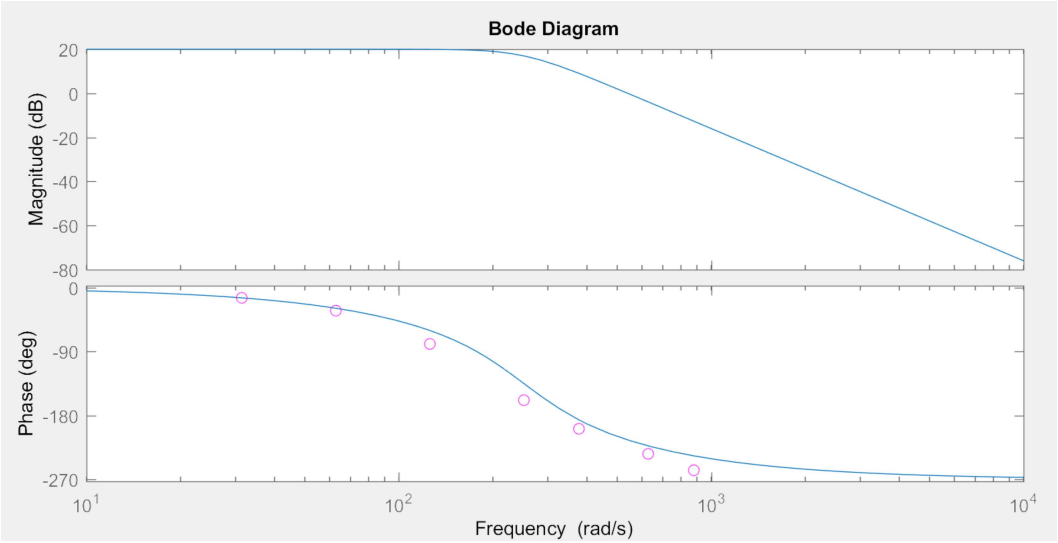

Figure 3: Theoretical and Experimental Frequency Response: Phase

Overall, the results from this experiment show that the biquad cascade used to approximate a third order low-pass filter is reasonable approximation for a continuous system. The step response performance characteristics for the discrete system is very close to those seen in the theoretical continuous system. The frequency response for the discrete system is also very close to that of the theoretical continuous system in that the magnitude and phase diagrams are mostly in agreement, with some potentially mitigable overestimation.

Appendix A: MATLAB Code

```
clear all; close all; clc
load('Lab6.mat')
%Transfer Function
wn=2*pi*40;
zeta=0.5;
sys1=tf([wn^3],[1,wn]);
sys2=tf([1],[1,2*zeta*wn,wn^2]);
systot=10*sys1*sys2;

%Step Response
t=0:0.0005:(499*0.0005);
yp=y+5;
[ys,ts]=step(systot);
plot(t(271:400)-0.135,yp(271:400),'k'), hold on          % Adjusting data for overlapping
plot(ts,ys,'m--')                                        % graphs of output response and
xlabel('Time [s]')                                       % theoretical step response
ylabel('Voltage [V]')
legend('Oscilloscope Output','Step Response','Location','SE')

%Frequency Response
Hz=[5 10 20 40 60 100 140 200];
rad=2*pi*Hz;
lograd=log10(rad);
Vpp=[10.16 10.16 10 6.562 2.5 0.625 0.1562 0.09376];
dBVpp=20*log10(Vpp);
dt=[8 9 11 11 9.2 6.5 5.1 3.8];
dtdegree=-360*(dt.*10^-3)./(1./Hz);

[MAG,PHASE,W]=bode(systot);
Wrad=log10(2*pi*W);
MAGx=squeeze(MAG);
dBMAGx=20*log10(MAGx);
PHASEx=squeeze(PHASE);

figure
bodemag(systot), hold on
plot(rad,dBVpp,'mo')

figure
bode(systot), hold on
plot(rad,dtdegree,'mo')
```