Lab 1

Introduction to First Order Circuits using MultiSim, MATLAB and Simulink

Objective: In this lab, you will be introduced to some of the computer simulation tools that you will use for your homework, prelab and lab assignments. You will learn to model a first-order system and display measured characteristics.

Equipment: Laptop with MATLAB, SIMULINK and MultiSim installed

Pre-lab: Read this entire lab theory and procedure thoroughly and complete the following tasks before your laboratory session.

- Install MATLAB on your laptop from the DFS Network Drive
- Purchase the ECE205 lab kit and check out the NI myDAQ from the parts room.
- Install the NI myDAQ software suite onto your laptop. The software is available on the diskette in the box. You should use all of the default settings.
- Review the following link and build your first MultiSim circuit and circuit using the MyDAQ. You must demonstrate the MultiSim circuit working correctly at the beginning of lab to receive full credit for the prelab. http://www.youtube.com/ntspress#p/a/u/0/MZiZ_C-ngkY
- Please see your instructor if you have any questions.

Theory:

A *first order circuit* contains a resistor and one energy storage element either an inductor or capacitor. It is referred to as a first order circuit because it can be described by a first-order differential equation. There are many first order systems, not just electrical circuits, and they can all be described by first-order differential equations. The standard form for a first order system is

$$\tau\frac{dy}{dt} + y(t) = Kx(t),$$

where x(t) is the input, y(t) is the output, τ is the time constant, and K is the static gain. The time constant indicates how long it takes the storage element to store or release energy. The static gain represents the ratio between the output and input after the energy has stopped

changing or under steady-state conditions. This lab procedure will focus on transient analysis as the capacitor or inductor is charging or discharging.

In this lab, you will use MultiSim first to examine the step and natural response of first-order circuits. The *step response* is the response of the circuit after a voltage or current source is suddenly applied and the inductor or capacitor charges to a final value or stores energy. The *natural response* is the response of the circuit to the sudden removal of a voltage or current source and the inductor or capacitor releases energy or discharges.

Next, you will use Simulink to model a first-order system and then use MATLAB to run the Simulink model and create plots of the system response. You will be required to submit some screenshots and plots as part of your memo submission and these are stated in the procedure. Note that you may not finish this lab during the lab session and you are allowed to help each other and get help from me, but what you submit ***must be your own***.

Procedure:

Part I: MultiSim simulation of a first- order circuit

Part A: Build the RC circuit in MultiSim

1. Create a lab 1 course folder on your laptop to store all of your simulation files. Start MultiSim 11.0 and click **Evaluate** unless you have received the license from your instructor and then click **Activate** and follow the prompts to enter the license.
2. Once the design window opens, click **Place-> Component**
3. Under group, change the pull down to *Select all groups*
4. Type *ground* in the **Component** box and place the ground in the window by clicking **OK**
5. The place component window will then open up again and you should type *resistor* in the Component box, select **RESISTOR_RATED** and click **OK** to place it in the window.
6. The place component window will then open up again and you should type *capacitor* in the Component box, select **CAPACITOR_RATED** and click **OK** to place it in the window.

7. The place component window will then open up again and you should type *DC_POWER* in the Component box, select **DC_POWER** and click **OK** to place the DC voltage source in the window.
8. The place component window will then open up again and you should type *SPDT* in the Component box, select **SPDT** and click **OK** to place the single pole double throw switch in the window. **Close** the Component Window.
9. Double-click the capacitor and change the value to **1 μF**. Double-click the resistor and change the value to **100 kΩ**. This yields a time constant for the system of $\tau = RC = (100\text{ k})(1\ \mu) = 100$ ms and the gain is K = 1. In order to change the gain, it would be necessary to include more resistors.
10. Double click the DC voltage source and change the value to **5 V**. Next use the mouse to draw lines and connect the components to match Figure 1. Note that you can rotate the components by selecting them and pressing **CTRL-R**.
11. Click place text and put a title in the workspace such as your name, the date, and circuit title. Include a screenshot of this circuit in your lab memo submission.

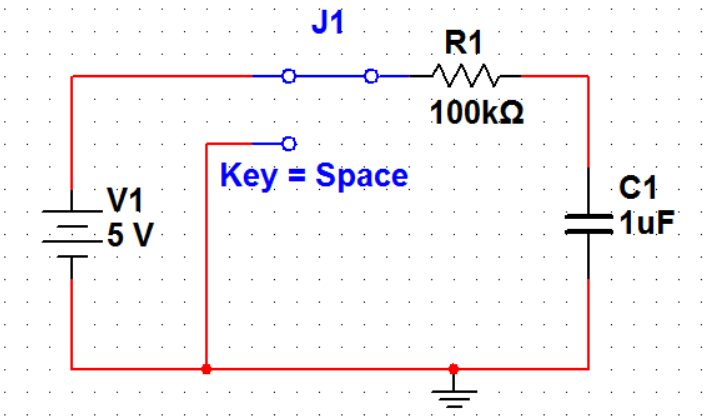


Figure 1: RC Circuit MultiSim Schematic

Part B: Interactive Simulation

1. Click the Agilent oscilloscope in the right tool bar as shown by the arrow in Figure 2 and place it on the schematic window.

2. Double-click on the oscilloscope in order to see an image of how the actual instrument looks.

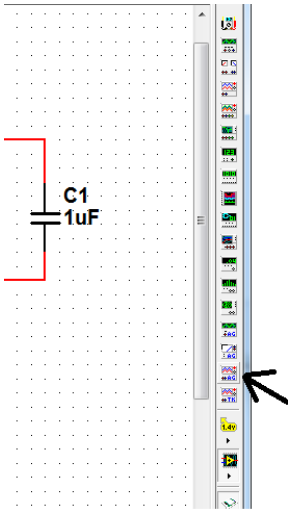


Figure 2: Agilent Oscilloscope

3. Push the power button to turn on the oscilloscope. Push the Analog, Channel 1 button to turn on Channel 1. Click the knob above the Channel 1 button until the window shows **2V/div** on channel 1.
4. Push the Analog, Channel 2 button to turn on Channel 2. Click the knob above the Channel 2 button until the window shows **2V/div** on channel 2.
5. Click the Horizontal knob to **50 ms/div** for both channels. Connect Channel 1 to the DC voltage source and Channel 2 to measure the voltage across the capacitor.
6. When you are finished your diagram should look like Figure 3.

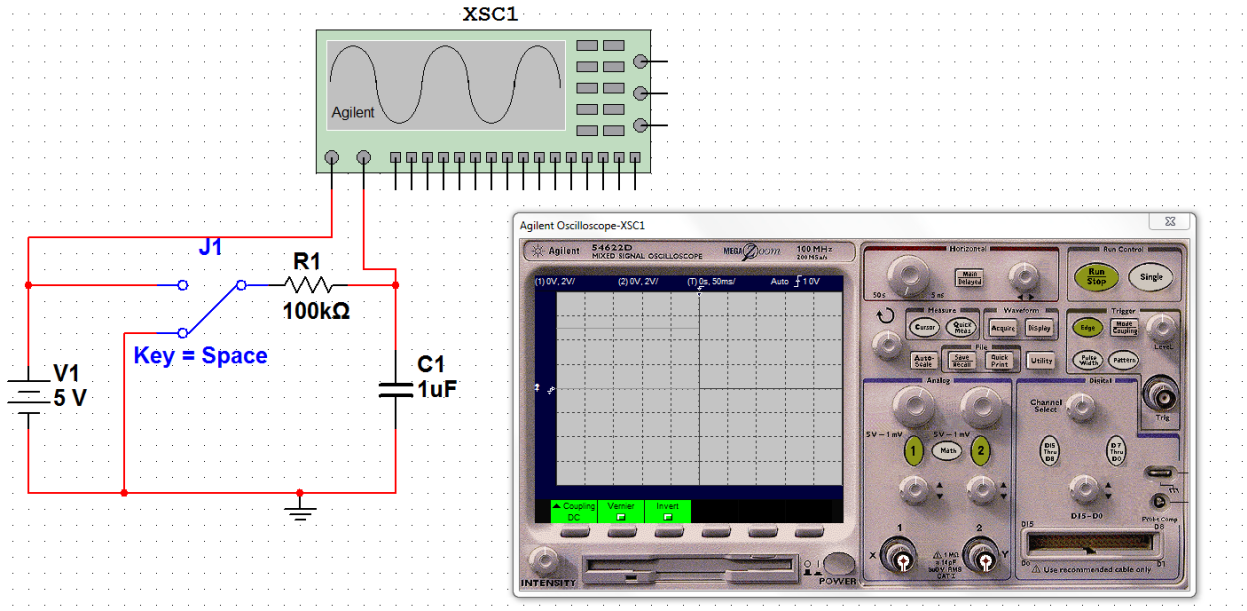


Figure 3: Interactive Simulation Setup

7. Click the green **play (run)** button or the **light switch** in the upper right hand corner to turn on the MultiSim interactive simulation (see Figure 4).

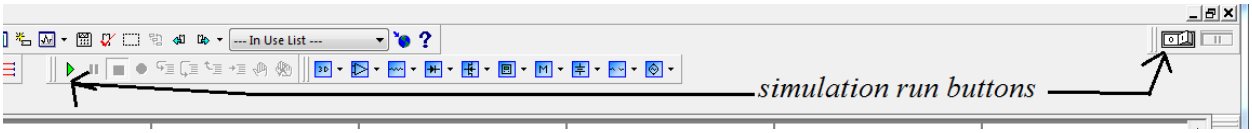


Figure 4: Simulation Run Buttons

8. Now push the spacebar on the keyboard multiple times and examine the first order response of the capacitor charging and discharging up to 5V. This exhibits the step and natural response of the RC circuit and it is an exponential function.
9. Now you will examine the first order response using a function generator as the input. Delete the 5V source and the switch from the circuit. Go to the toolbar on the right side of the window and add the Agilent function generator. The Agilent function generator is 2 buttons above the Agilent oscilloscope.
10. Open up the image, press the **Power** button and press the **Ampl** button and use the knob to set it to **5 Vpp**. Press the **square wave** and press the **Freq** and use the knob to set it to **1 Hz**.

11. Connect the output of the function generator to the input of the circuit to replace the voltage source that was deleted. Channel 1 of the oscilloscope should measure the input also and channel 2 should still measure the voltage across the capacitor. The schematic should now look like Figure 5.

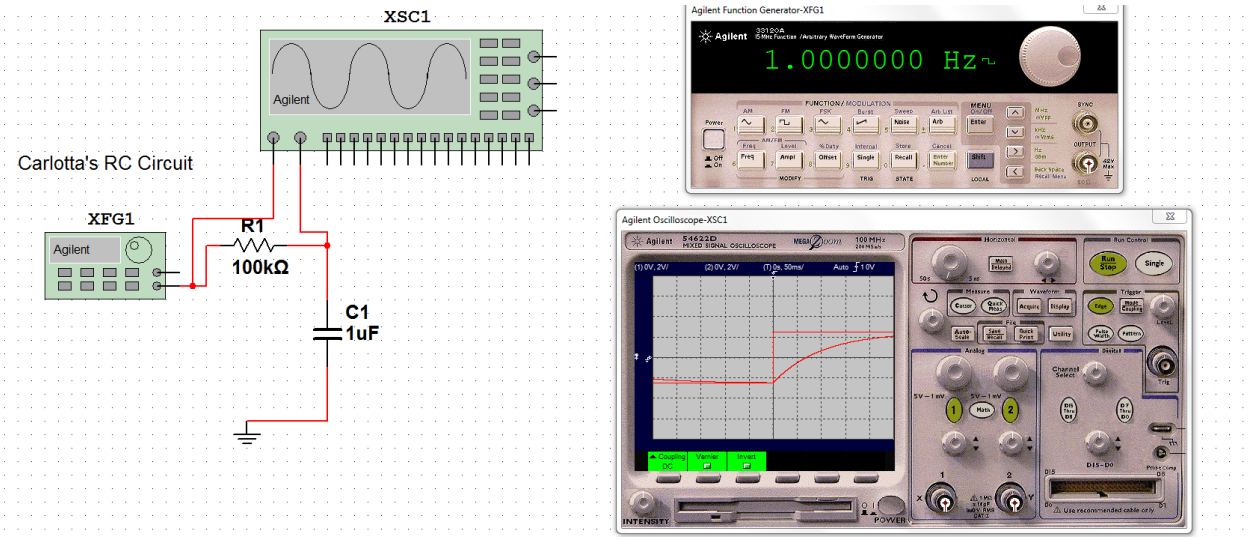


Figure 5: Function Generator and Oscilloscope Setup

12. Turn on the MultiSim simulation by flipping the switch in the upper right hand corner or press the green button. Once again the oscilloscope should show the capacitor charging and discharging.
13. Include a screenshot of the oscilloscope in your lab memo submission with an appropriate number and caption.

Part C: Transient Analysis

1. In this section, you will setup a transient analysis to examine the first-order response of the system. A transient analysis is the plot of the circuit response as a function of time.
2. Delete the function generator and oscilloscope from the circuit schematic. Change the resistor to ***1 kΩ*** and the capacitor to ***0.1 μF***. This makes the time constant for the system, $\tau = RC = (1\text{ k})(0.1\ \mu) = 100\ \mu s$ and the gain is still K = 1.
3. Place the component, **PIECEWISE_LINEAR_VOLTAGE** on the schematic and connect it to the RC circuit as the input source.

4. Double-click the piecewise linear voltage source and note that you can either enter data points from a file or enter them on a table. For this analysis, you will use the table. Change the input to match Table 1.

Table 1: x(t) piecewise linear function

Time	Voltage
0	0
0	2
0.0001	2
0.0001	-3
0.00025	-3
0.00025	4

5. Close the piecewise linear voltage source data table. Double click on the wire above the capacitor and name the node **Vcap** and **check show net name**. Your circuit should now look like Figure 6.

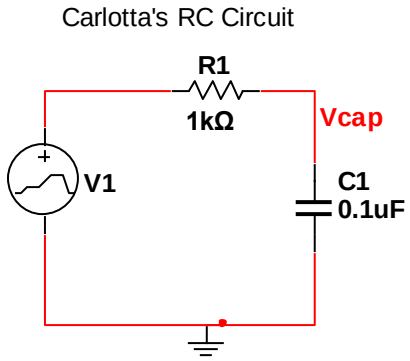


Figure 6: RC circuit with PWL source

6. To set up the transient analysis, click **Simulate->Analyses->Transient Analysis** and the set up window will open up. Leave the start time as zero and set the final time to **0.0008** seconds
7. Click the output tab, select **V(vcap)** and click **Add**. Then, click **V(1**) and click **Add**. Lastly, click **Simulate** and the transient analysis plot should open up. Click on the black and white square under the Cursor menu heading and the background of the plot should

change to white and be similar to Figure 7. Please include this plot in your lab memo submission.

8. You are finished with this part and can close MultiSim. Finally, attach your final Simulink file (**.ms11)** to your submission.

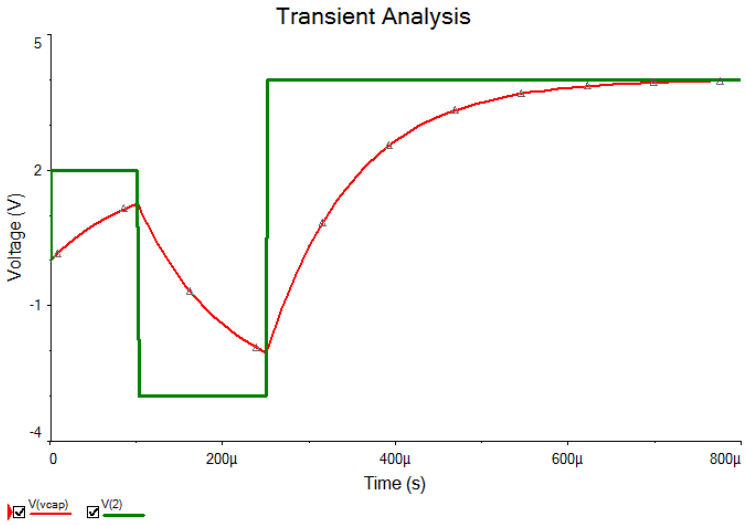


Figure 7: RC Circuit Transient Analysis

For more practice with MultiSim review the following links:

http://cnx.org/content/col10369/latest/

http://www.youtube.com/ntspress#p/a/u/0/MZiZ_C-ngkY

http://inst.eecs.berkeley.edu/~ee100/su06/handouts/EE100-MultiSim-Tutorial.pdf

http://digital.ni.com/manuals.nsf/websearch/193FBEEDC810E62C86257260006545AE

http://digital.ni.com/manuals.nsf/websearch/DB7B3DC28D422B3E862572600067215C

Part II: Build the Simulink Model using XY graph

1. In this part you will simulate first order differential equations by using Simulink and MATLAB. The first step is to solve the standard form of the first order differential equation for the highest derivative power of the output, y(t) as shown here,

$$\frac{dy}{dt} = \frac{1}{\tau}[Kx(t) - y(t)]$$

2. Since the integral of the above equation yields y(t), this will be used to create the simulation model by using a scaling (gain), summing, and integration blocks.

3. Start MATLAB and when it finishes initializing it will show **Ready** near the **Start** button. Change the directory in the drop down to the folder created in Part I.
4. Type **simulink** at the command prompt and hit **enter**. The Simulink Library Browser in Figure 8 will open up.

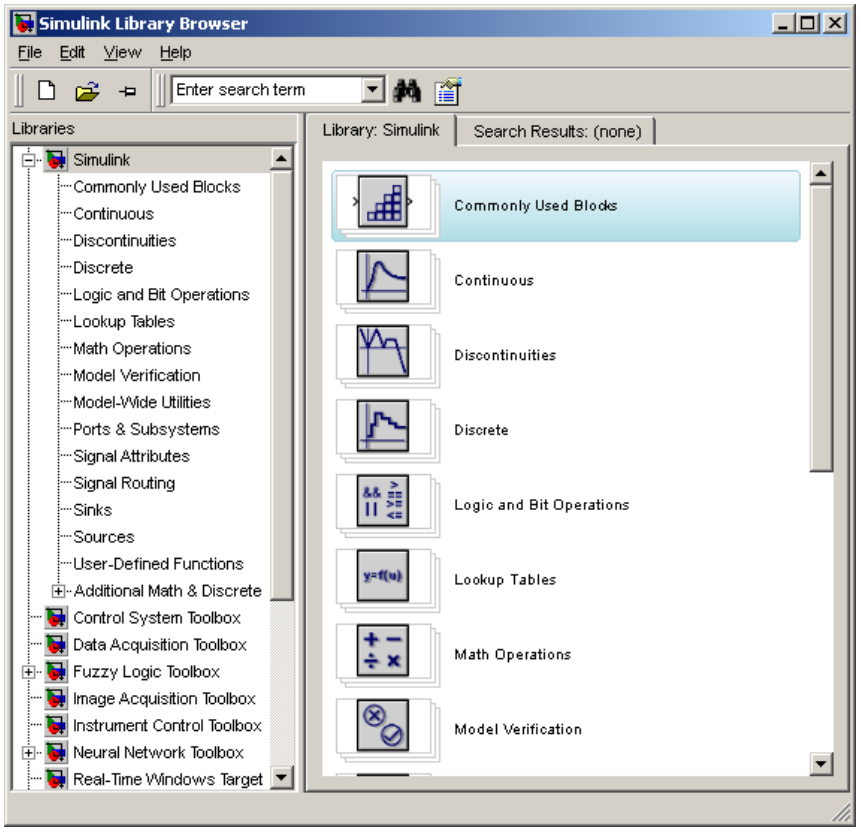


Figure 8: Simulink Library Browser

5. This window gives you access to all of the blocks you will use to make the first-order model.
6. Click **File->New->Model** and this will open the blank window where you will create the SIMULINK model. Click **File->Save As** and name the model **Lab1.mdl**.
7. Assume that τ **= 0.02** seconds, **K = 0.5**, and the input is a step with amplitude, **A = 0.5** and let's build the model.
8. In the **Simulink Library Browser**, scroll down until you see a block named **Sources** as shown in the figure on the next page, and then double click on it. This is where we will find the available system inputs.
9. Drag **Clock** over to your Simulink model file, *Lab1.mdl*, and then scroll down to find **Step** and drag this over to your model file.

10. Next we are going to need a sink in order to save or plot the results. In the left panel, click on **Sinks** and then drag the **XY graph** over to Lab1.mdl. Make sure to save the SIMULINK model often so that you not lose your work.
11. We are going to need an integrator, so click on **Commonly Used Blocks** in the left panel, and drag an **Integrator** (1/s) over to your model file, drag a **Sum** over to your model file, and then drag a **Gain** over to your model file.
12. Rearrange the blocks in the general order you will use them, as shown in Figure 9. At this point you probably realize you will need another **Gain** block. You can go back to the Library Browser, or just right click on the Gain block you have and choose copy and then paste.

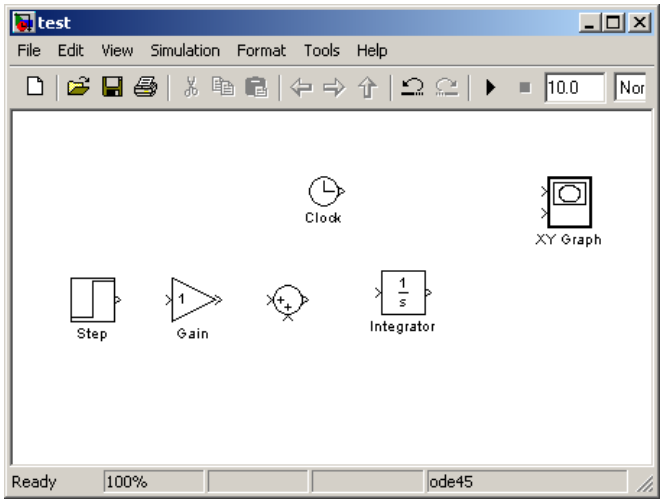


Figure 9: Simulink Model Blocks

13. Now we are ready to connect the blocks. This takes some practice, but you generally click on the source or destination block and drag the line. Connect the blocks so your model now looks like the Figure 10. To get the bottom line into the summer, make a line going down from the sum block, let go, then make a separate line from your first line to the line after the integrator.

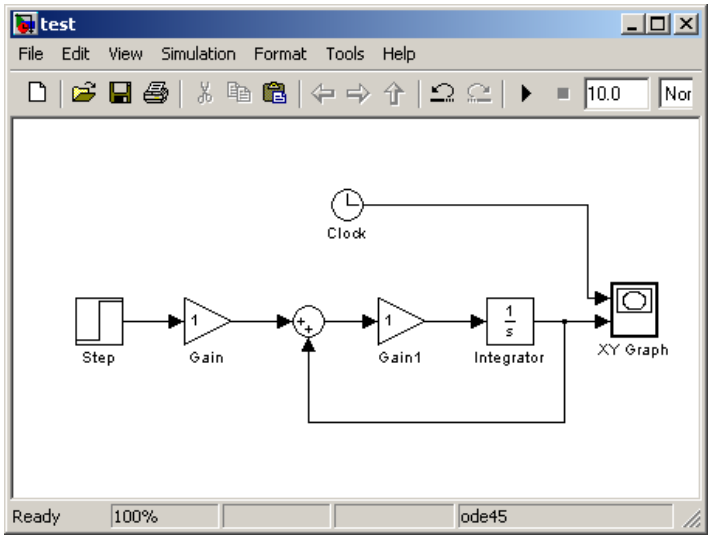


Figure 10: Simulink First Order System Model

14. Double click on the **Step** block, and set the **Step Time** to **0** and the **Final Value** (the amplitude of the step) to **0.5** (the value of A). Then click **OK**.
15. Double click on the **Gain** block next to the Step block, and set the gain equal to **0.5** (the value of K), and then click **OK**.
16. Double Click on the **Sum** block. We will change the parameters, as shown in Figure 11. Let's choose a rectangular sum block (this works better if there are more inputs), and change the bottom sign to -. Also, we will put in extra space by adding some space, denoted by |. Finally, click **OK**.

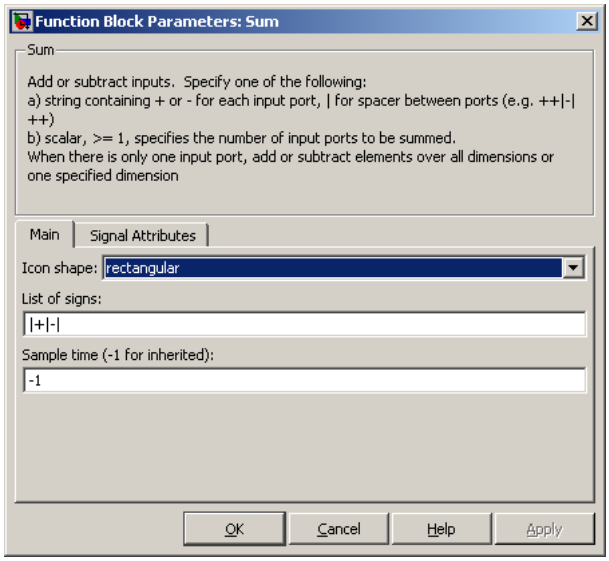


Figure 11: Function Block Parameters for the Summer

17. The **Gain** block represents $1/\tau$ so double click on it and enter **50** and then click **OK**.
18. We want to view the output on the **XY Graph** so double click on the **XY Graph** and enter x-min = 0, x-max = 0.3, y-min = 0, y-max = 0.3, then click OK.
19. Since, the output will only plot from 0 to 0.3 seconds, change the final time of the simulation to 0.3 seconds as shown in Figure 12.

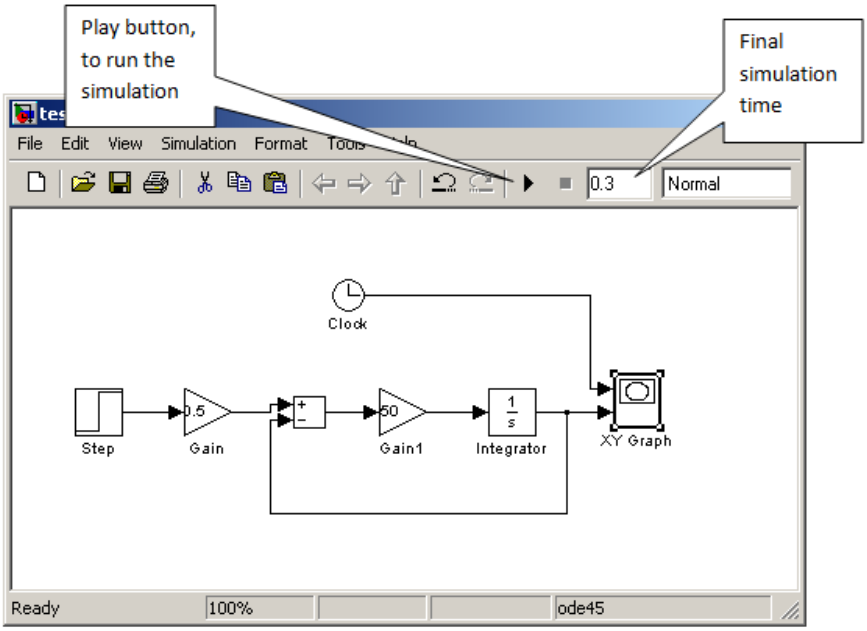


Figure 12: Finally Simulation Model

20. Finally, run the simulation by pressing the play button shown in Figure 12 and the step response graph will be shown in Figure 13.

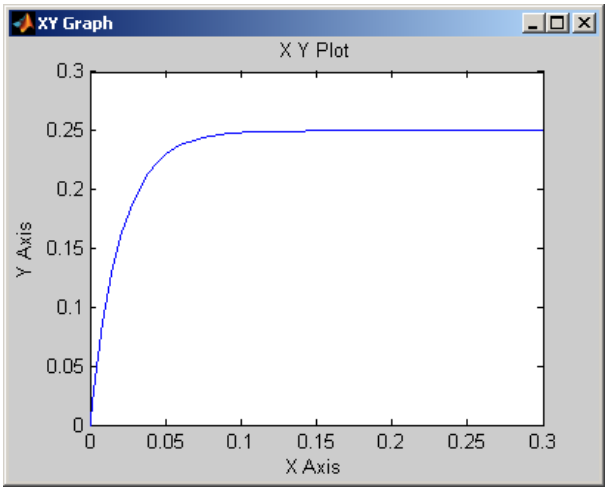


Figure 13: First-order Step Response

Part III: Use MATLAB to plot a step response

1) While the **XY Graph** is nice for getting a basic idea of what is going on, you have little control over how the graph is presented. You also may need to know a lot about the signal you want plotted. You can use MATLAB to create a graph with more capabilities.
2) Go back to the **Simulink Library Browser**, click on **Sinks**, and then drag two **simout (To Workspace)** blocks over to your simulating file (*Lab1.mdl*).
3) Right click on **XY graph** to delete it and then click on data paths to delete them also.
4) Connect the simulation output (y) and the **Clock** to the two different **simout** blocks, as shown in Figure 14.

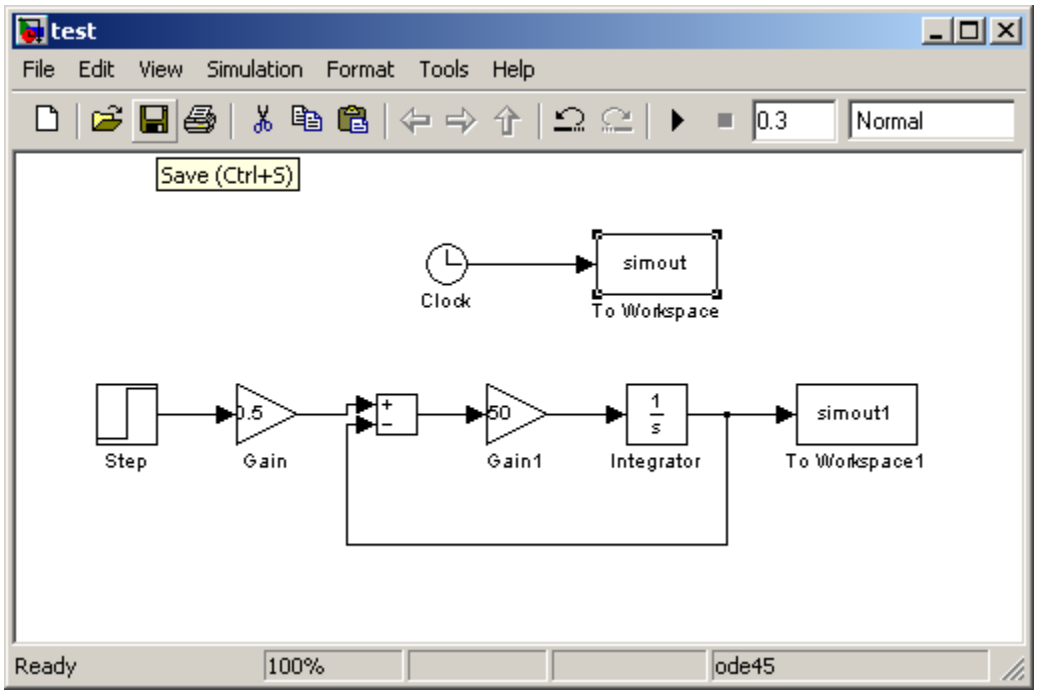


Figure 14: First-order Model with simout

5) Double click on the **simout** block connected to the **clock**. Set the variable name to **time** and the Save format to **Array**.
6) Double click on the **simout** block connected the **integrator**. Set the variable name to **y** and the Save format to Array. Your simulation file should now look like Figure 15.

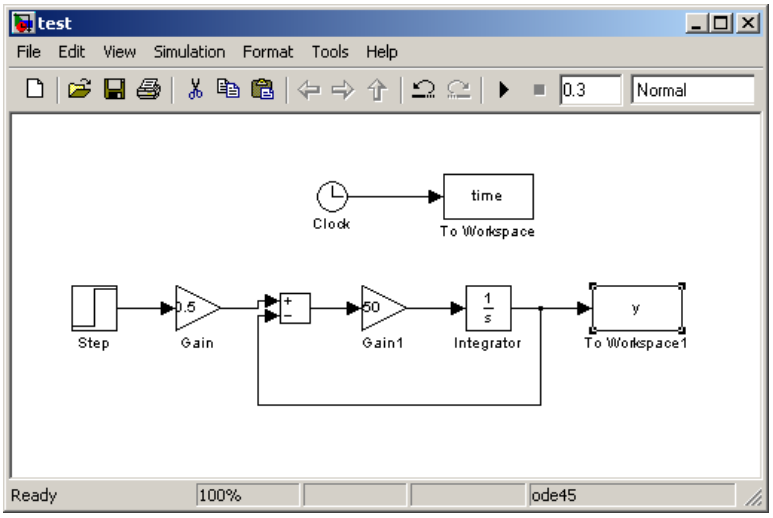


Figure 15: Final First-order Model

7) Run the simulation by pushing the play button. It may not appear that much has happened, (you may get some warning messages, but ignore them for now) but the simulation has saved the data to your workspace.

8) Type **who** in the MATLAB command window, and you should see that both the **time** and **y** (and maybe some other variables) are now in the workspace.

9) To plot the data, type the following lines at the MATLAB command prompt:

```
>> plot(time,y,'Linewidth',3);grid;
>> xlabel('time (sec)');ylabel('y(t)');title('First Order System');;
```

A figure similar to the one shown in Figure 16 should open up.

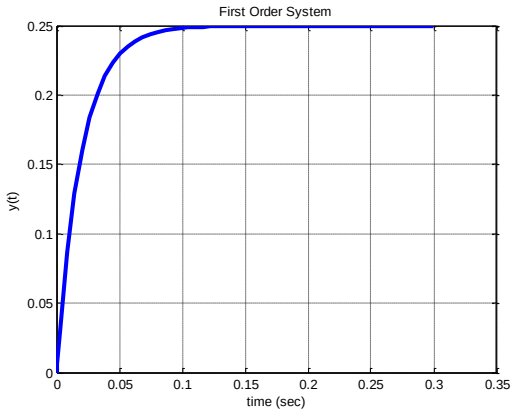


Figure 16: First-Order Step Response

Part IV: Creating a MATLAB script file (m-file) to run simulations

1. Just as in general programming it is a bad habit to "hard code" a variable, the same is true with Simulink. For example, we would like to be able to change the parameters in our simulation, run the simulation, and plot the results in a very convenient way. This is what we will accomplish in this level.
2. In the **MATLAB** command window, select **File→New→Script Ctrl+N.** As an alternative, you can click on the leftmost icon just under File (it looks like a page with a red dot in the upper left corner). Yet another new window will open up!
3. In this new window, select **File->Save As**, and then name it **Lab1_driver.m**.
4. In this new file, type the following lines:

```
%%%%%%%
%% Lab1_driver.m
% Geordi LaForge
% February 22, 2222
% this program runs the file Lab1.mdl to model first-order systems
%
% clear all variables
clear variables;
% close all figure windows
close all;
% simulate the model
sim('Lab1.mdl');
```

5. Any line in MATLAB that begins with a % is a comment line. Note that it is a good practice to always include a header at the top of the m-file with the name of the file, your name, the date it was created and a short description. In addition, it is a good practice to comment your code so that it is easy to debug and easy for a reader to follow. This standard will be expected on all future m-file submissions.
6. The last line, **sim('Lab1.mdl');** actually runs the Simulink file.
7. Save this file and then run it, as shown in Figure 17.

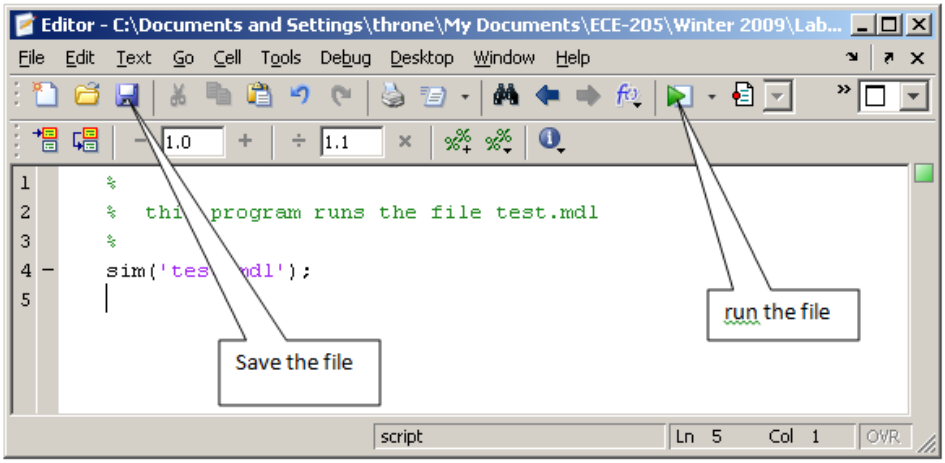


Figure 17: Save and Run an m file

8. To see that this is working, in the MATLAB workspace type **clear**, then **who**. There should be no variables in the workspace.
9. Then run m-file ***Lab1_driver.m*** and type who in the workspace, and you should see variables have been generated.
10. We would also like to automatically generate a plot every time we run the MATLAB program, so enter the following lines into your MATLAB program (after the **sim** command), so that the data is in the workspace.

```
%% plot the data
plot(time,y,'-','Linewidth',3);grid;
xlabel('time(sec)');ylabel('y(t)');
title('Lab 1  - First Order System');
```

11. You should **run (play)** the program again, and if should generate a graph (figure) like before. You may want to remove any figures before you run it to be sure it generated a new figure.
12. If may also be useful for use to plot both the output, y(t) and the steady state value . To do this, first modify your Simulink file **Lab1.mdl** to look like Figure 18. Make sure to double-click on the Kx block and make the save format **Array**. Note that you also may need to select an entire group of elements in Simulink to shift your elements around.

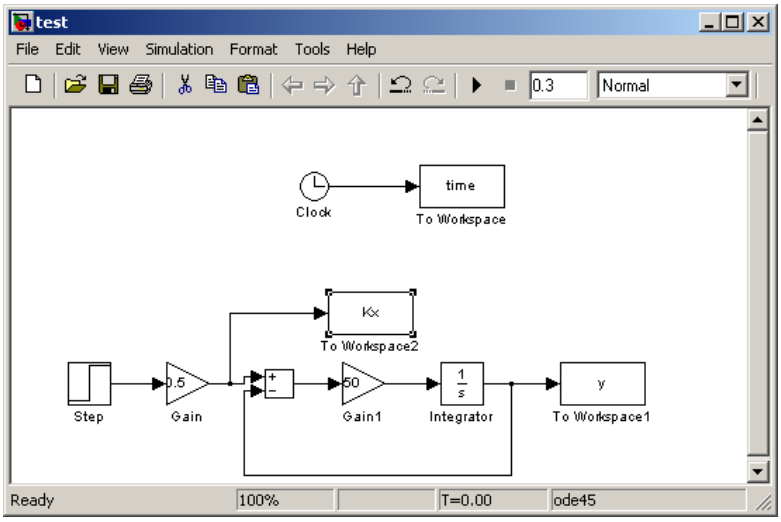


Figure 18: First-order Simulink model with 3 simout variables

13. Modify the plotting commands in your MATLAB file, **Lab1_driver.m** as follows:

```
plot(time,y,'-',time,Kx,'o','Linewidth',2);grid;
legend('y(t)','y_{ss}');
xlabel('time(sec)');ylabel('y(t)');
title('Lab 1  - First Order System');
axis([0 0.3 0 0.3]);
```

14. MATLAB defaults to finding a convenient set of axes, but sometimes you want to tell MATLAB more specifically the axes you want. If you type **help plot** in the MATLAB command window you can find a number of options for plotting. If you run your code now, you should get Figure 19. Note that you can drag the Legend around and put it where you want.

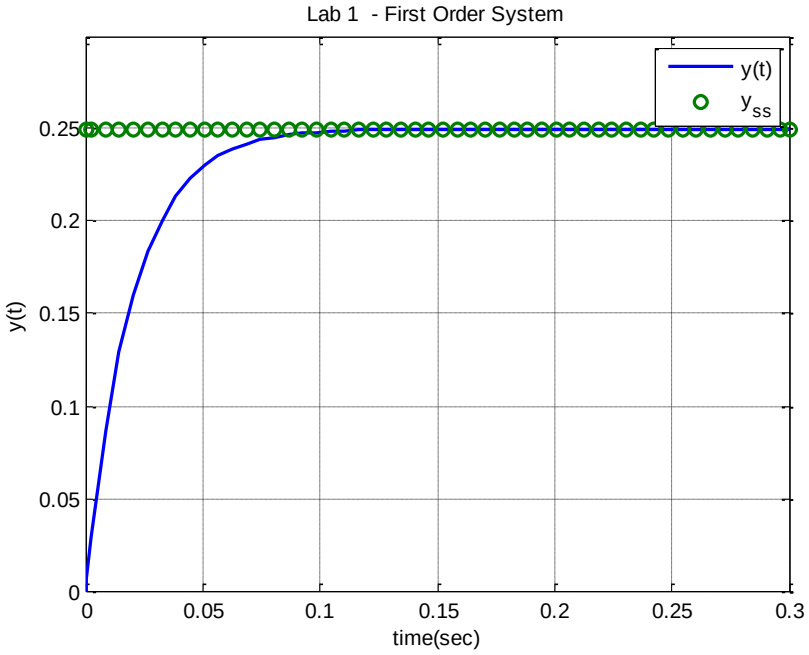


Figure 19: Step Response with input and output

15. Finally, we would like all of the parameters we have in our Simulink model file to be controlled by the MATLAB program. Modify the MATLAB program **Lab1_driver.m** and put the following lines before the simulation command (**sim**)

```
%% set the first-order system characteristics
tau = 0.02;
K = 0.5;
A = 0.5;
Tf = 0.3;
```

16. Next, edit the Simulink file, and use these variable names instead of the values you entered previously. The final value of the **Step** should now be **A**, the gain of the first **Gain** block should be **K**, and the gain of the second **Gain** block should be **1/tau**.
17. The final time of the simulation should be changed to **Tf**. Delete any figures and run your MATLAB driver file again. You should get the same figure as before (see Figure 17). You should include this figure in your lab memo submission with your initials included in the title. To include this graph, select **Edit** and then **Copy Figure**. You can then paste the figure

in your memo. Do not include your figure using any other method! Be sure the figure is large enough and that I can see it and that it has a figure number and caption.

Part V: Generating an input in MATLAB

1. As long as we are putting all of the variables in our MATLAB file (program), we might as well figure out how we can use this to generate an input.
2. Go to the **Simulink Library Browser**, click on **Sources**, and drag a **simin (From Workspace)** block over to your Simulink model file. Delete the **Step** block and connect the **simin** as the input.
3. Double click on the **simin** block and name the data **xt**. This is where Simulink will look for its input data. Note that you will need to produce both **data values ( )** and **time values (t)** and put them into the variable **xt**. One method to do this is presented and Simulink will interpolate between the data values as needed.
4. To create the data, we will use **MATLAB's** anonymous function to generate the signal. To do this type the following lines in your **MATLAB** code before the **sim** command and after you have defined **Tf**:

```
%% generate the input
x = @(t) 1*((t>=0)&(t<0.1)) -1*(t>=0.1);
t = linspace(0,Tf,300);
xt = [t' x(t)'];
```

5. The first piece of code after the comment is the definition of x(t). The second line of code tells MATLAB to generate an array of time values for 0 to Tf, and use 300 evenly spaced points. The third line of code puts the data in the correct form for Simulink. The apostrophe means to take the transpose, the brackets [] mean form an array, and x(t) tells MATLAB to evaluate the function x at the specified times t. (Note that you might feel that it would be easier to use MATLAB's Heaviside function here, but sometimes Simulink complains, since MATLAB defined the Heaviside function evaluated at zero to be not a number (NAN).) Comment out the axis command and run your code, you should get a graph like Figure 20.

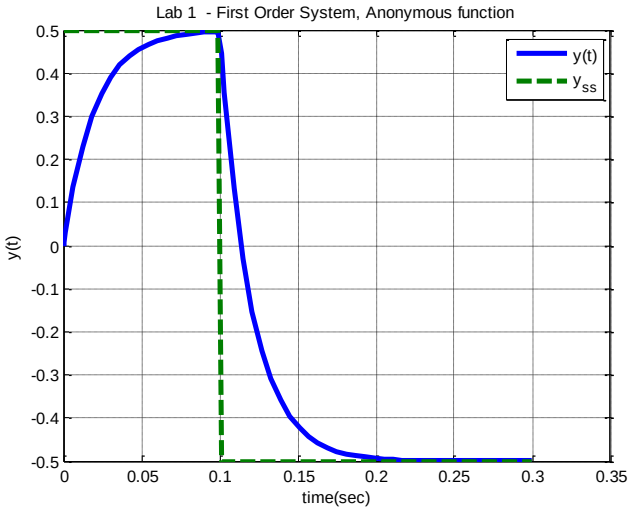


Figure 20: Step Response with an Anonymous Function

6. Now let's try something more complicated. In the notes we have an example for a first order system with parameters, **τ = 0.0001 s**, **K = 0.01**, and **y(0) = 0.01**. Change the time constant and static gain parameters in your MATLAB script file to match these. This is similar to what you did in Part IV of the procedure.
7. set the simulation final time to **Tf = 0.0008** seconds. Make a new variable called **y0**, and set **y0 = 0.01;.** You will no longer use the A variable so you can keep the value the same as before.
8. Click on the **integrator** in your Simulink file and set the Initial condition to **y0**.
9. The system input is

$$x(t) = \begin{cases} 0 & t < 0 \\ 2 & 0 \le t < 0.0001 \\ -3 & 0.0001 \le t < 0.00025 \\ 4 & t \ge 0.00025 \end{cases}$$

10. To enter the input, change the definition of the function as follows:

```
x = @(t) 0*(t<0)+2*((t>=0)&(t<0.0001))...
     -3*((t>=0.0001)&(t<0.00025))+4*(t>=0.00025);
```

11. Note that to continue a MATLAB function or statement on the next line, you end the line with three dots (…)

12. Comment out the axis command, and run the code. If you have done everything correctly, you should get the graph in Figure 21. Include this graph in your lab memo submission.

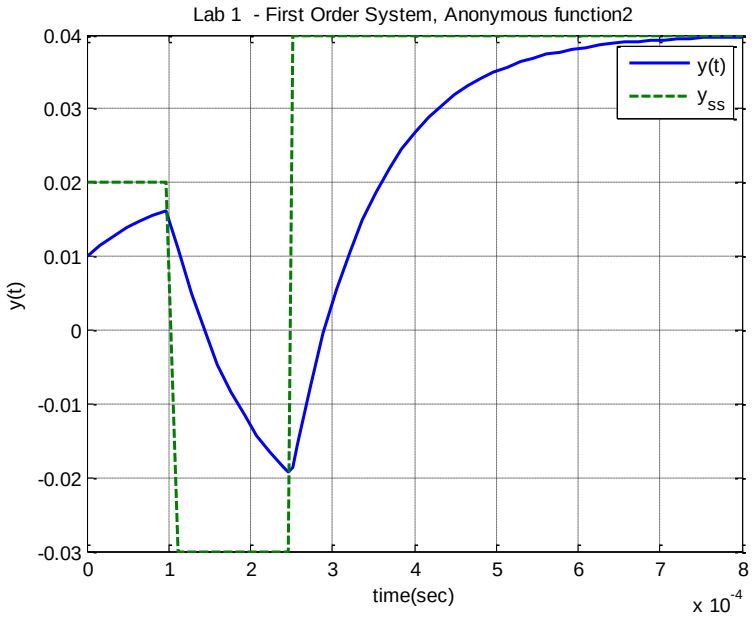


Figure 21: Step Response with an Anonymous Function

Part VI:Creating an analytical function to check your work

1. Finally, when you are doing your homework, you are going to have problems like these, and are going to need a method to check your answers.
2. Anonymous functions and MATLAB will be used to check the answer if the solution is

$$y(t)=\begin{cases} 0 & t<0 \\ -0.01e^{-t/0.0001}+0.02 & 0\le t<0.0001 \\ 0.04632e^{-(t-0.0001)/0.0001}-0.03 & 0.0001\le t<0.00025 \\ -0.05966e^{-(t-0.00025)/0.0001}+0.04 & t\ge 0.00025 \end{cases}$$

3. In MATLAB, type in the following anonymous function just below your definition of your anonymous definition for x,

```
ya = @(t) 0*(t<0)+(-0.01*exp(-
t/0.0001)+0.02).*((t>=0)&(t<0.0001)) ...
+(0.04632*exp(-(t-0.0001)/0.0001)-
0.03).*((t>=0.0001)&(t<0.00025))...
```

```
+(-0.05966*exp(-(t-0.00025)/0.0001)+0.04).*(t >=0.00025);
```

4. Next change your plotting and legend commands to the following:

```
plot(time,y,'b-',time,Kx,'r--
',time,ya(time),'mo',’Linewidth’,s);

legend('y(t)','y_{ss}','y_{analytical}');
```

5. If everything worked correctly, you should get the plot in Figure 22, which shows the analytical solution matches the simulated solution. <u>This graph should be included in your lab memo submission.</u>

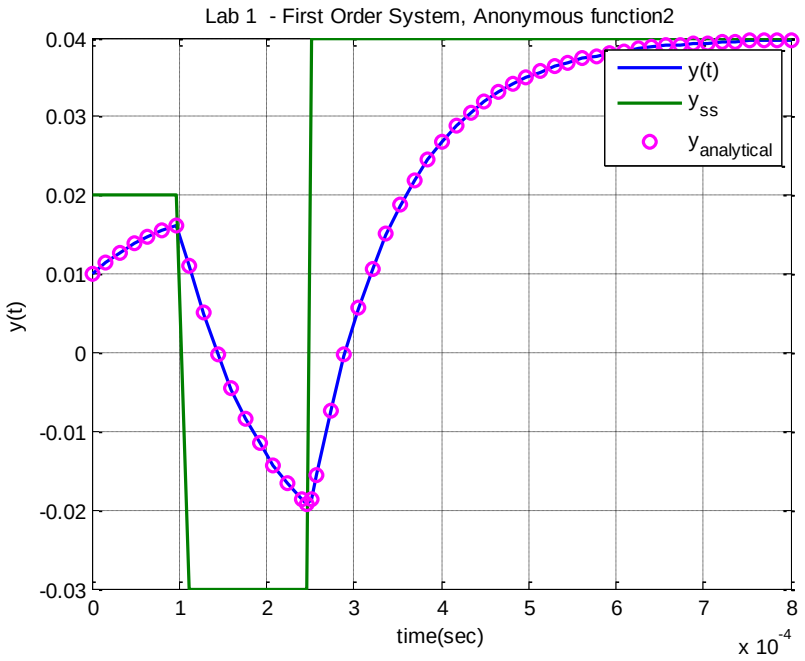


Figure 22: Analytical and Simulate Solution of First Order System

6. At this point, you have completed your lab procedure and should include all of the required figures. There should be six figures in all, 3 from the MultiSim analysis and 3 from the MATLAB/SIMULINK analysis.
7. Finally, attach your final Simulink file (**Lab1.mdl)** and the MATLAB driver file (**Lab1_driver.m**) to your submission.

Submission:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by 11:59 p.m. on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written in first person from you
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment (the procedure should be very short, a high level summary of what you did for each part). The procedure should be two paragraphs at the most.
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text
- Also, you must to include a statement in your memo that this is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results

Lab 2

First and Second Order Circuits

Objective: In this lab, you will use the NI myDAQ and MATLAB with Simulink and MultiSim to measure the response and characteristics of first order and second order circuits. In addition, you will use this analysis to identify characteristics of the first order system such as the static gain and time constant.

Equipment: Laptop with MATLAB and MultiSim

NI my DAQ

RLC Meter

0.01 μF capacitor

1 μF capacitor

1 kΩ resistor (x 2)

10 kΩ variable resistor

33 mH inductor

Pre-lab: Read this entire lab procedure thoroughly and complete the following tasks before your laboratory session.

Become familiar with the NI my DAQ instruments by reviewing the documentation in the box and the tutorials at the following links:

Introduction to NI myDAQ: http://decibel.ni.com/content/docs/DOC-13041

DMM – ohmmeter: http://decibel.ni.com/content/docs/DOC-12880

Analog Output and Input: http://decibel.ni.com/content/docs/DOC-12884

Analytical Derivations

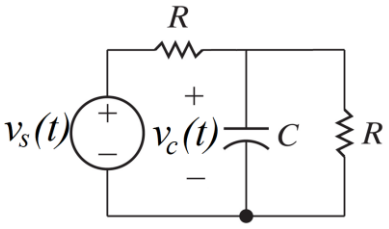


Figure 1: RC Circuit

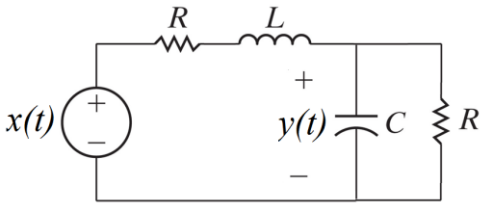


Figure 2: RLC Circuit

- Derive the governing differential equation in standard form for the circuit in Figure 1 and state the static gain and time constant.
- State the solution to the governing differential equation in Figure 1 assuming zero initial conditions, a step input with constant amplitude, A, C = 1 μF and R = R = 1 kΩ.
- Derive the governing differential equation in standard form for the circuit in Figure 2 and state the static gain, natural frequency and damping ratio.
- If A = 1, L = 33 mH, C = 0.01 μF and R = R = 1 kΩ. What is the static gain? natural frequency? and damping ratio?

MATLAB/SIMULINK Problem

One of the standard forms for a second order system is

$$\ddot{y}(t) + 2\zeta\omega_n\dot{y}(t) + \omega_n^2 y(t) = \omega_n^2 K x(t),$$

where ζ is the damping ratio, ω_n is the natural frequency, and K is the static gain. Use this form of the standard second order system in the remainder of this problem. Use a MATLAB driver to simulate a system described by this differential equation. Similar to what you did in Lab 1 for the first order system, solve for the highest power derivative (as a function of the input and lower power derivatives). Next, integrate $\ddot{y}(t)$ to yield $\dot{y}(t)$ and then integrate again to yield y(t). Therefore, you will need two integrators and two feedback loops and one input into your summing block (click on the summing block and modify it to get three inputs) in SIMULINK. You may need to click on the gain block and then choose **flip block** to get the correct direction.

- We want to look at the step response, so the input to your system should be a step. The amplitude of the step should be controlled by the MATLAB program, the length of the simulation should be controlled by the MATLAB program, and the step should start at time zero.
- Your Simulink file should only contain variables (static gain, natural frequency, damping ratio, amplitude of the step, length of the simulation)
- Plot the transient output of the system and the steady state output, KA, on the same graph.

- If you use the parameters, $\zeta = 0.1$, $\omega_n = 20$, K = 2, A = 1, and Tf = 3 (final simulation time), you should get results like that shown below.

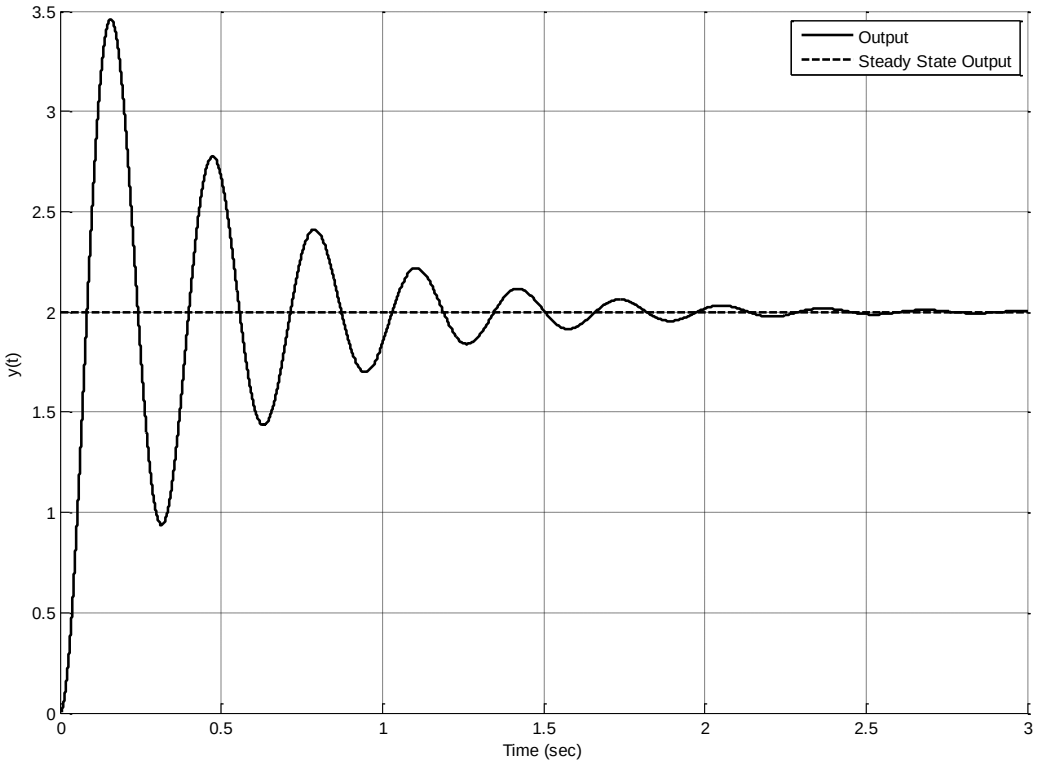


Figure 3: Second-order underdamped step response

- You should submit the following three items as part of your pre-lab submission:
  - A printout of your figure with a title and the axes labeled with units
  - A printout of your MATLAB CODE with a header comment with your name, the assignment and the date, code, and
  - A printout of your SIMULINK model

If you have any questions, please ask your instructor.

Procedure:

Part I – Component Measurements

Launch the NI ELVISmx Instrument Launcher from the National Instruments folder and you should see the toolbar appear on your desktop. Click on the DMM on the Instrument Launcher and use the NI myDAQ to measure the actual values of the two 1 kΩ resistors and compare them to the nominal values. Connect the red and black probes between the HI and COM terminals on the bottom of the NI myDAQ, press the Ω for resistance and press run to acquire the reading. If you get an error, make sure that you change the mode to specify range not auto. See Figure 3 for an example of the desktop.

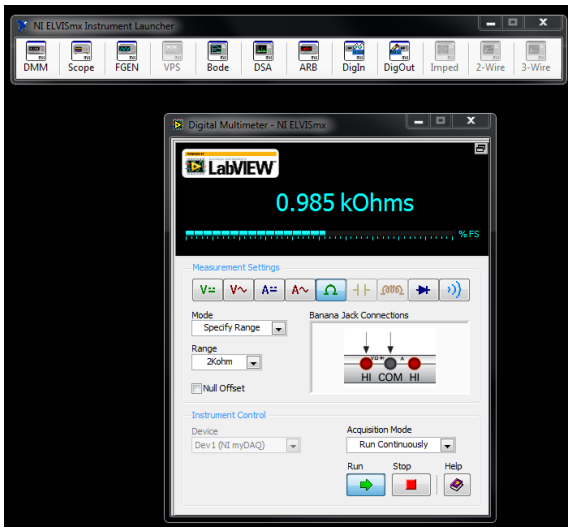


Figure 3: DMM Measurements with the NI myDAQ

You will need to use the RLC meter available in the classroom to measure the **1 μF** capacitor. Compare the measured values to the nominal values of these components by including a data table with error analysis in your lab memo submission (see Table 1). All data tables included in the memo should have a number and caption and be referenced in the text. Note that percent error is calculated by using the formula, $\%\ error = \frac{measured - nominal}{nominal}\ x\ 100.$

	measured	nominal	% error
C, nF		1000	
R, Ω		1000	
R, Ω		1000	

Table 1: Inductor and Capacitor Actual Values

Use the measured values of the resistors and capacitor to calculate the time constants, τ, for the circuit shown in Figures 1 ($\tau = R_{th}C$). This will be the nominal value for the time constant for this circuit should be compared to the theoretical values found in the prelab. Table 2 provides an example of this comparison, $\%\ error = \frac{nominal-theorectical}{theoretical}\ x\ 100$. Note that since you are using 5% resistors and 20% capacitors, the time constant may have up to a 20% error.

	nominal	**theoretical**	**% error**
RC Circuit, τ, ms		0.5	

Table 2: First Order Circuit Time Constant, τ

Use the measured values of the resistors and capacitors to calculate the static gain, K, for the circuit shown in Figure 1. This value will be the nominal static gain for RC Circuit for the remaining of the measurements in the lab (see Table 3).

	nominal	**theoretical**	**% error**
RC Circuit , K		0.5	

Table 3: First Order Circuit Static Gain, K

Part II – First order circuit characteristics

In this part, you will use the NI myDAQ to estimate the time constant of the two RC circuits by measuring the 10% - 90% rise time, t_r, using the formula, $\tau = \frac{t_r}{\ln(9)}$.

a) Connect red leads from to the +15V, and -15V terminals on the MyDAQ. Connect **black** leads to the two AGND terminals on the MyDAQ. Connect **white** leads to the AO 0 terminal on the MyDAQ. These are the power and function generator connections.
b) Connect **green** leads to the AI 0- and AI 1- terminals on the MyDAQ. Connect **yellow** leads to the AI 0+ and AI 1+ terminals on the MyDAQ. These are the oscilloscope connections.
c) Build the circuit in Figure 1 on your breadboard with **R = R = 1 kΩ, C = 1 μF**.
d) You will use the MyDAQ function generator to provide the source voltage so connect the leads from the AO 0 and AO AGND terminals across the input to the RC circuit.
e) You will use the MyDAQ oscilloscope to measure the input and output of the RC circuit. Connect the leads from the AI 0+ and AI 0- to the left side of the resistor and to ground in

parallel with the function generator connections. Connect the leads from the AI 1+ to the positive terminal on the capacitor and the AI 1- to the negative end of the capacitor.

f) Click on the Scope and FGEN to launch the oscilloscope and function generator on the NI MYDAQ. Set the Function Generator to a square wave at 100 Hz and 2.00 V_{pp} and 0.00 V offset. Confirm that signal route is AO 0 and click run to provide the source voltage to the circuit.

g) Set the oscilloscope Chanel 0 settings to Source: AI 0, 500 mV/div and Timebase of 2 ms/div. Set the oscilloscope Channel 1 settings to Source AI 1, 500 mV/div and set the Trigger to Edge and the Chan 0 Source. Make sure to **enable** both channels and click run to run continuously.

h) Your function generator and scope output should look similar to the image in Figure 4 which represents the capacitor charging and discharging repeatedly. *Note that this was a different circuit so there will be some differences!!*

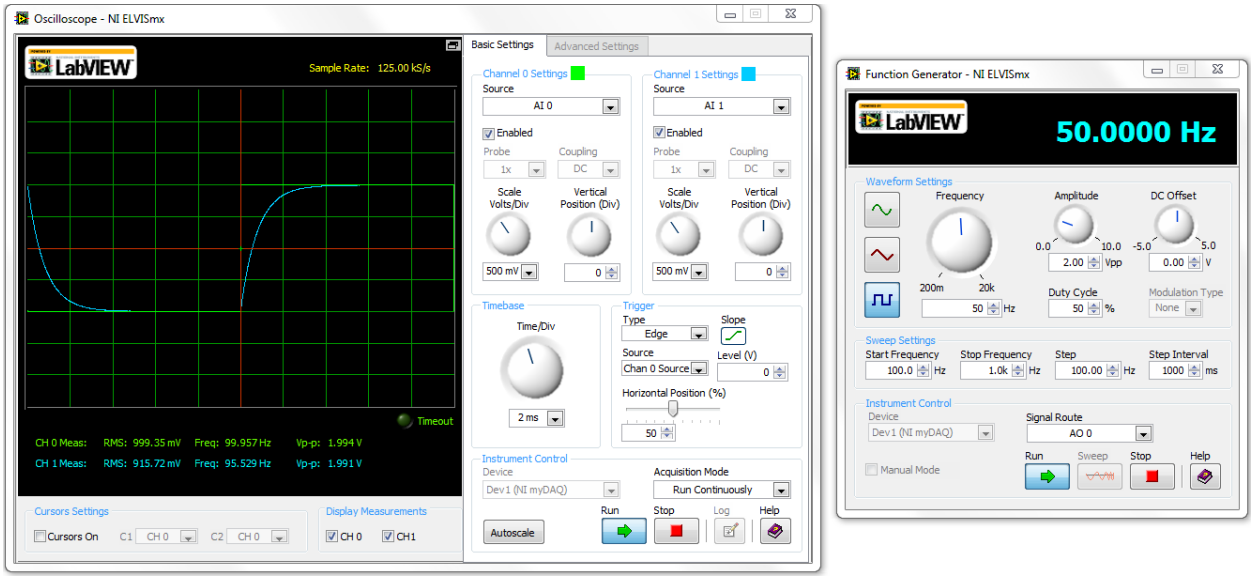


Figure 4: Scope measurements with the NI myDAQ

i) Next, we will measure the rise time on the signal so click the check box to enable the cursors under the display window.

j) Use the drop down menu to set C1 to CH 1 and C2 to CH1 and check both channels to display measurements.

k) Find the yellow C1 and C2 cursors on the left edge of the window and move them toward the rising edge of the output voltage. Align C1 at the bottom of the output signal on top of the rising edge for the input square wave. Align cursor C2 at the top of the signal after it has reached steady state. Find the difference between the Channel 1 low and high value by using C2 – C1.

l) Next, move C1 to the [low value of channel 1 + 0.1(C2-C1)] and move C2 to the [high value of channel 1 – 0.1(C2-C1)]. The time difference between these two cursors is displayed as dT. This time difference represents the time it takes for the waveform to go from 10% to 90% of the final value so dT is also the rise time, t_r. Use the rise time to calculate the time constant, τ and compare it to the nominal value in a data table in your lab memo submission. If this value is not within 15% of the nominal value, then ask for help.

m) In this part, you will measure the static gain of the circuit in and compare the actual results to the theoretical results found in the prelab and the nominal results found in part I.

n) Use the peak to peak voltage for channel 2 and channel 1 after the output reaches steady-state to calculate $K = y_{ss}/x_{ss}$.

o) Create a data table to compare these results and include the analysis in your memo submission.

p) Next use the snipping tool or alt-PrtSc to capture a screen shot of the oscilloscope display for inclusion in your lab memo.

Part III – Time constant estimation – Alternate method

In this part you will use an alternate method to estimate the time constant for the RC circuit in Figure 1. This is estimation based upon the fact that a capacitor will charge to 63.2% of its final value in one time constant, $y(t) = KA(1 - e^{-t/\tau}) = KA(1 - e^{-1}) = 0.632KA$.

a. Similar to part II, enable both cursors and set C1 and C2 to CH 1. Align C1 at the bottom of the output signal at the rising edge of the step input and align C2 at the top of the output signal or maximum value of channel 1. Find the difference between C2 and C1 and calculate 63.2% of this number.

b. Move C2 to the [low value of channel 1 + 0.632(C2-C1)], you may not be able to get the value exactly but get as close as you can. The time difference between these two cursors is dT and represents the estimation of the time constant, τ.

c. Compare this estimate of the time constant to the one found in part II and demonstrate the result of the analysis in a data table in the memo submission.

Part IV – Time constant estimation using MultiSim

In this part, you will build the circuit in Figure 1 in MultiSim and use the result to estime the time constant.

1. Build the circuit in Figure 1 in MultiSim and use the NIELVISmx Function Generator and NIELVISmx oscilloscope to measure the input and output. These instruments are located on the toolbox down the right side of the window near the Agilent instruments.
2. Make the function generator and oscilloscope settings the same as in part II. (see Figure for the proper setup).
3. Confirm that the results are similar to Parts II and III and capture a screen shot for inclusion in your lab memo submission.

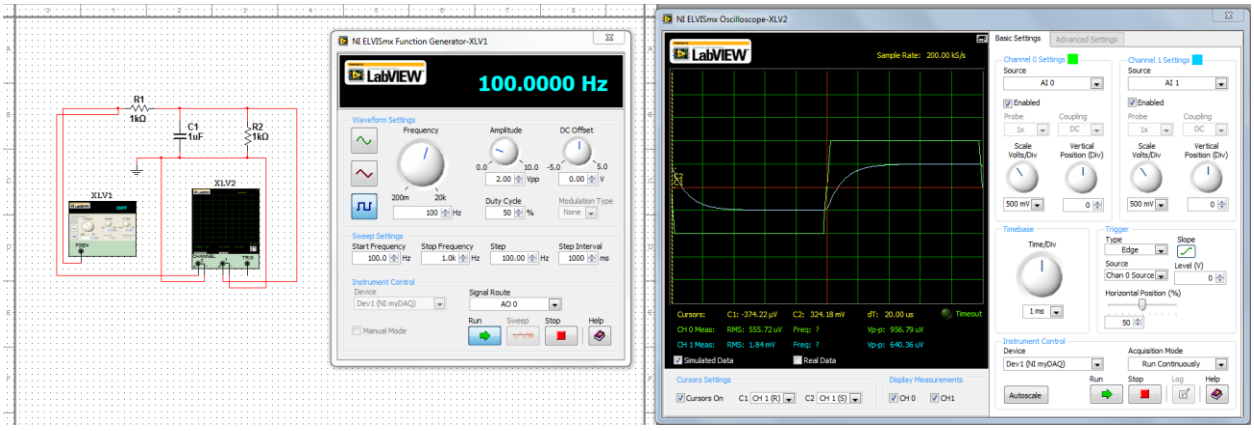


Figure 5: MultiSIM simulation of an RC circuit

Part V – Time constant estimation of a different RC circuit

In this part, you will measure the time constant of different RC circuits to examine how it affects the system output.

1. Replace the 1 kΩ resistor in parallel with the capacitor in the circuit you built with a 10 kΩ variable resistor (potentiometer). Make sure the connection point is between the center tap and one of the side terminals. If you place the leads across the two terminals, the potentiometer acts as a 10 kΩ resistor.
2. Adjust the variable resistor so that the time constant is significantly smaller than the original 0.5 ms.
3. Adjust the **Volts/Div** and **Time/Div** on the oscilloscope so that the waveform is as large as possible and displays one full period. You may also have to adjust the function generator frequency in order to get one full period on the oscilloscope screen.
4. Repeat the steps in part II or III of the lab procedure to estimate the time constant.
5. Include the time constant estimation and oscilloscope screen capture in your lab memo submission
6. Adjust the variable resistor so that the time constant is significantly larger than the original 0.5 ms and repeat steps 3 thorough 5.

Part VI – Time constant estimation for an unknown RC circuit

In this part, you will measure the time constant of an unknown first-order circuit. Note that there will be a question like this on your lab practical to make sure that you completely understand the procedure.

1) Get a numbered black box from your instructor and write the number down. Make sure to include the box number in your lab memo submission.
2) Place the box so that the number is right side up and the red terminals are positive voltage and the black connectors are ground.
3) Note that since you will devise a method to estimate the time constant, these instructions are not complete.
4) Connect the function generator AO and AGND to the left side of the box, the input. Connect the oscilloscope, AI 0+ AGND to the left side of the box, the input.
5) Connect the oscilloscope, AI 1+ AI 1- to the right side of the box, the output

6) Set the function generator input signal to a square wave with 2 Vpp and a frequency between 50 and 200 Hz.
7) Adjust the **Volts/Div** scale on the oscilloscope so that the output signal is are larage as possible on the screen.
8) Adjust the **Time/Div** scale on the oscilloscope so you can view one full period of the output waveform.
9) Use the cursors to measure the time constant for the unknown first-order circuit.
10) You should include this estimate of the time constant and the black box number in your lab memo submission.
11) You should also include a screen shot of the oscilloscope output in your lab memo submission.

Part VII – Underdamped seconder order system characteristics using the NI myDAQ

In this part you will examine the characteristics of an RLC circuit to determine the static gain, natural frequency and damping ratio. Figure 6 shows some of the important characteristics of the step response for an underdamped second order system including Percent Overshoot (PO), Time to peak (Tp), settling time (Ts), and steady state value (y(∞)).

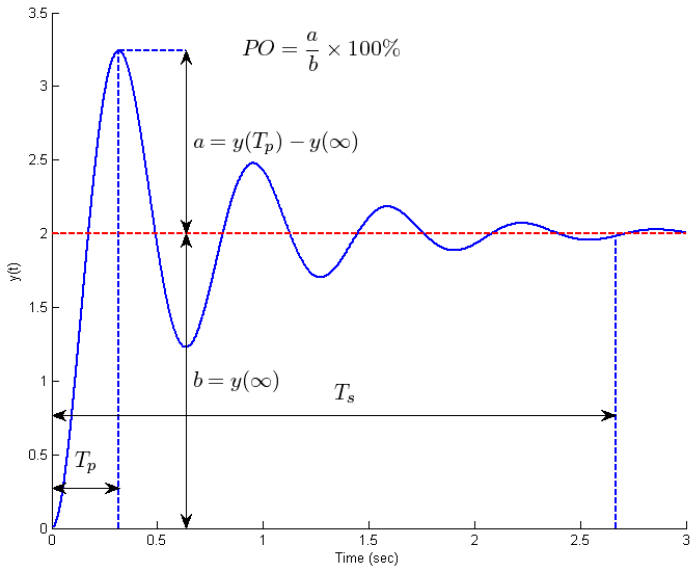


Figure 6: Second-order underdamped system characteristics

- Use the RLC meter and NI myDAQ to measure the nominal value for the 33 mH inductor.
- Build the circuit in Figure 2 **without** the resistor in parallel with the capacitor.
- Calculate the **theoretical** values of the static gain, natural frequency and damping ratio for this circuit using the nominal values for the resistor, inductor and capacitor.
- Calculate the **nominal** values of the static gain, natural frequency and damping ratio for this circuit using the measured values for the resistor, inductor and capacitor.
- Compare the nominal values to the theoretical values in a data table. This comparison should include performing an error analysis and showing the results in a data table in your memo submission. This table should have a table number and caption and be referenced in the text.
- Calculate the theoretical values for the time to peak, percent overshoot and settling time for the RLC circuit using the nominal component values.
- Calculate the nominal values for the time to peak, percent overshoot and settling time for the RLC circuit using the measured component values.
- Compare the nominal values to the theoretical values in a data table. This comparison should include performing an error analysis and showing the results in a data table in your memo submission. This table should have a table number and caption and be referenced in the text.
- The circuit characteristics with the measured component values will be the nominal values for the remainder of the measurements on this circuit.
- You should now have the circuit in Figure 2 built **without** the resistor in parallel with the capacitor. Please confirm that the component values are R = 1 kΩ resistors, L = 33 mH and **C = 0.01 μF** capacitor. ***Note that the capacitor is different from Parts I - VI!***
- Set up the NI myDAQ function generator to supply a 1.5 kHz 2.00 Vpp square wave.
- Set the oscilloscope channels 0 and 1 to measure the input source and output voltage across the capacitor with 500 mV/division and a 100 μs timebase.

- The trigger should be edge with the source as channel 0. Your system response should look similar to the scope display in Figure 7. *Note that this image is from a different circuit so they won't be exactly the same!*

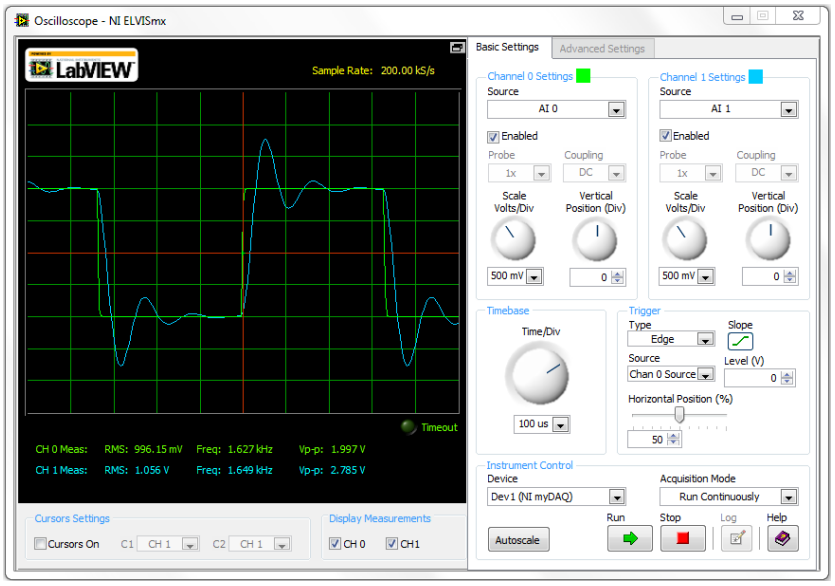


Figure 7: Underdamped system response

- Use the oscilloscope cursors to find the time to peak by putting C1 at the rising edge of the output voltage and C2 at the center of the first peak, dT represents the time to peak.
- Use the oscilloscope cursors to find the damping frequency, ω_d, in rad/s by measuring the period, dT, between two consecutive peaks and using $\omega_d = 2\pi/T_d$.
- Use the oscilloscope cursors to determine the settling time when the output gets within 2% of its final value and stays there. C1 should be at the rising edge of the signal and C2 should be moved until the voltage is within 2% of the final value and at this point dT represents the settling time.
- Use the oscilloscope cursors to measure the percent overshoot using the formula, $\frac{y_{peak}-y_{ss}}{y_{ss}}\ x\ 100$ with C1 at the maximum peak and C2 at the final value
- Use the oscilloscope cursors to find the static gain by measuring the peak to peak value of the input channel and the peak to peak value of the output channel at steady-state.

- The damping ratio can be calculated from the measured percent overshoot by using the following formula, $\zeta = \sqrt{\frac{\ln^2(OS)}{\ln^2(OS)+\pi^2}}$. Note in this formula the overshoot should be expressed as the decimal equivalent of the percentage.
- The natural frequency can be calculated from the measured damping frequency and damping ratio by using the following formula, $\omega_n = \frac{\omega_d}{\sqrt{1-\zeta^2}}$.
- You should summarize all of the characteristics for this circuit in a data table and compare them to the nominal values with an error analysis in the memo submission.

Part VIII – System Types

- Put the 10 kΩ variable resistor (potentiometer) in parallel with the capacitor. Make sure the connection point is between the center tap and one of the side terminals. If you place the leads across the two terminals, the potentiometer acts as a 10 kΩ resistor.
- Use the screw driver to turn the potentiometer all the way clockwise to the 9 and observe the output. What type of response does this represent? You should capture this scope display for inclusion in your lab memo.
- Use the screw driver to turn the potentiometer counterclockwise to the 2 and observe the output. What type of response does this represent? You should capture this scope display for inclusion in your lab memo.
- Use the screwdriver to adjust the potentiometer until the output voltage has the fastest settling time with no overshoot. What type of response does this represent? You should capture this scope display for inclusion in your lab memo.
- Recall that the system parameters are $\omega_n = \frac{1}{\sqrt{LC}}$ and $\zeta = \frac{R_{th}\sqrt{C}}{2\sqrt{L}}$ so there is a direct relationship between resistance and the damping ratio but not the natural frequency.
- Remember all figures should have a number and caption and be referenced in the text of the lab memo.

Submission:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by 11:59 p.m. on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written in first person from you
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment (the procedure should be very short, a high level summary of what you did for each part). The procedure should be two paragraphs at the most.
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text
- Also, you must to include a statement in your memo that this is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results

Lab 3
Time Domain Modeling of One Degree of Freedom Systems

Overview

In this lab you will model two one degree of freedom (second order) systems using time-domain analysis. The goal is develop some intuition into how the parameters in a second order system affect the output by varying parameters in a model to match the step response of two second order systems. You will also use the log-decrement method to estimate these parameters. You will complete this procedure on two different rectilinear (model 210) systems.

Prelab

Read this entire lab procedure and background theory thoroughly. Then answer the following question on engineering paper and submit it in class the day before the lab session.

a) Draw a free body diagram of the forces on the mass for the system shown in Figure 1.

b) Show that the equations of motion can be written as:

$$m_1\ddot{x}_1(t)+c_1\dot{x}(t)+(k_1+k_2)x_1(t)=F(t)$$

or

$$\frac{1}{\omega_n^2}\ddot{x}_1(t)+\frac{2\zeta}{\omega_n}\dot{x}(t)+x_1(t)=KF(t)$$

c) Express the damping ratio,ζ, natural frequency, ω_n, and static gain, K, in terms of m_1, k_1, k_2, and c_1.

d) For the system in Figure 2, express the damping ratio,ζ, natural frequency, ω_n, and static gain, K, in terms of J, c and k.

Background Theory

A one degree of freedom rectilinear mass-spring-damper system can be modeled as shown in Figure 1.

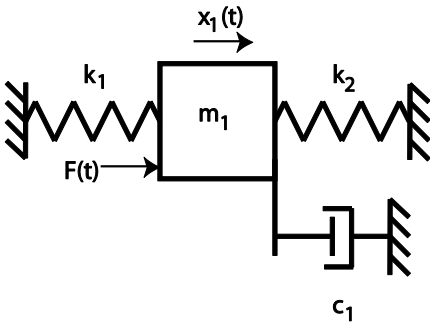


Figure 1: 1 DOF Mechanical Rectilinear System

By drawing a free body diagram and balancing forces, we get the equation of motion:

$$m_1\ddot{x}_1(t)+c_1\dot{x}_1(t)+(k_1+k_2)x_1(t)=F(t)$$

A one degree of freedom rotational mass-spring-damper system can be modeled as

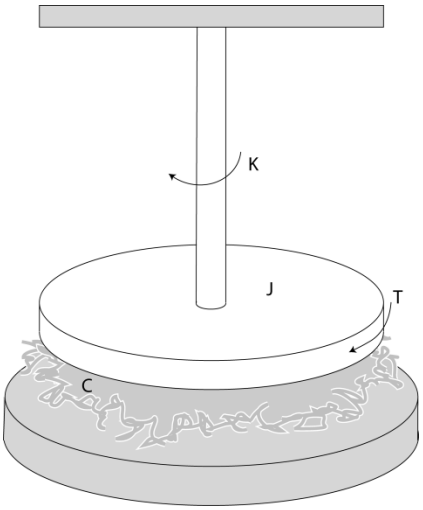


Figure 2: 1 DOF Mechanical Torsional System

By drawing a free body diagram and balancing torques, we get the equation of motion

$$J\ddot{\theta}(t)+c\dot{\theta}(t)+k\theta(t)=T(t)$$

Despite the fact that the systems appear quite different, the transfer functions for both of the one degree of freedom systems can be put into the standard form for modeling second order circuits:

$$\ddot{y}(t)+2\zeta\omega_n\dot{y}(t)+\omega_n^2y(t)=\mathrm{K}\omega_n^2x(t)$$

The system parameters are the static gain, K, the natural frequency, ω_n and the damping ratio, ζ. These parameters must be determined in order to create the system model.

Procedure

PART I - Set up the files

- Create a folder for ECE 205 on the desktop of the computer and create a Lab 3 folder in this folder.
- Download and extract the **Lab3Files.rar** from the Angel course website Lab folder into the folder created on the computer desktop.
- Start MATLAB and change the default folder to the folder where you extracted the Lab 4 files.

PART II - Set up the communication

You will need to go through the following steps for **two different** configurations. This should be two rectilinear systems with different masses and/or springs with or without the damper. In this part, you will get the system ready to run and start communications between Simulink, the miniPMAC card and the ECP system.

1. Press the power button to turn on the ECP system. It is on the top bench next to the computer.
2. To inform the ECP system that we will use Simulink and the real-time windows target, click **Start -> Programs -> ECP**
3. Click on **Utility-> Download Controller Personality File**
4. Select **C: -> Program Files -> ECP Systems -> cn** (it may default to this)
5. Finally select **m210_rtwt_3.pmc for** the Model 210 (or **m205_rtwt_3.pmc** for the Model 205) and click on **open**. Wait for the ECP system to load the personality file, then close the window completely (do not just minimize it).
6. Now you need to reset the system. This should be done each time before you run the system. From MATLAB, type **simulink** at the command prompt to open the Simulink file, **ECPDSPReset.mdl** (see Figure 1).

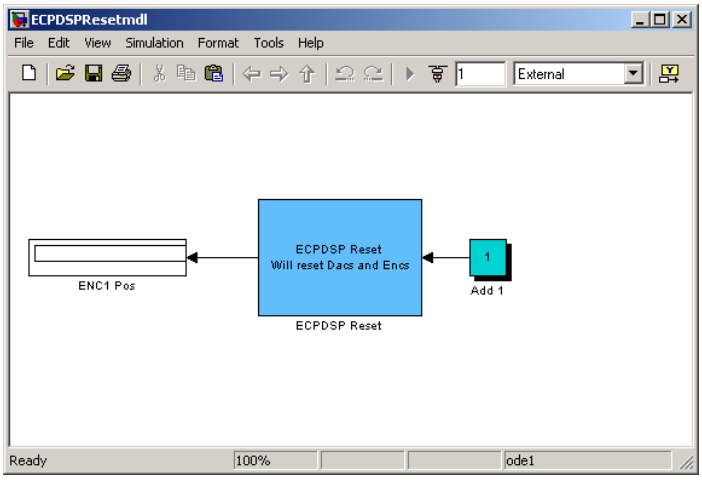


Figure 1: ECPDSPReset.mdl

7. Verify that the **Base I/O Address** is correct for your work station by double clicking on the blue ECPDSP Reset box, and checking the parameters in the window that opens (see Figure 2). Since all of the computers in the room are Dell computers, the **Base I/O Address** should be **'0xD800'**. Also change the **Sample Time (Sec)** to **0.005**. Click Apply→OK and make sure that you save the file if you had to change the address.

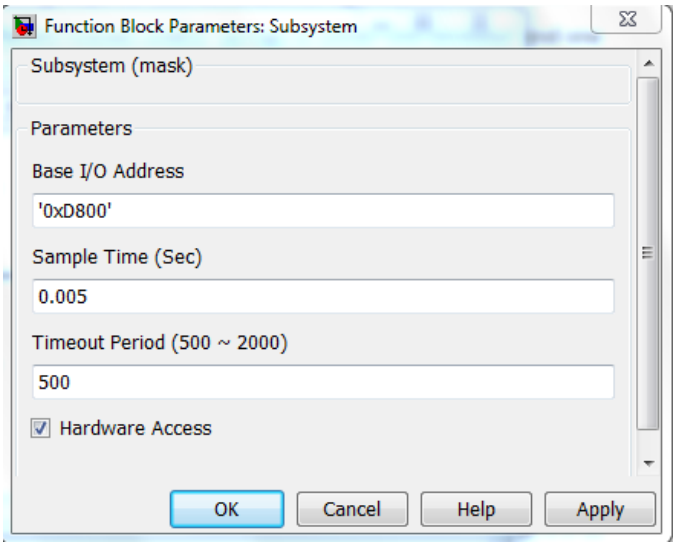


Figure 2: ECPDSP Reset Function Block Parameters

8. **Compile** the **ECPDSPReset.mdl** file by clicking on the icon shown in Figure 3. Make sure to wait for the file to finish compiling when the messages in the MATLAB command window stop. Ignore any errors and continue.

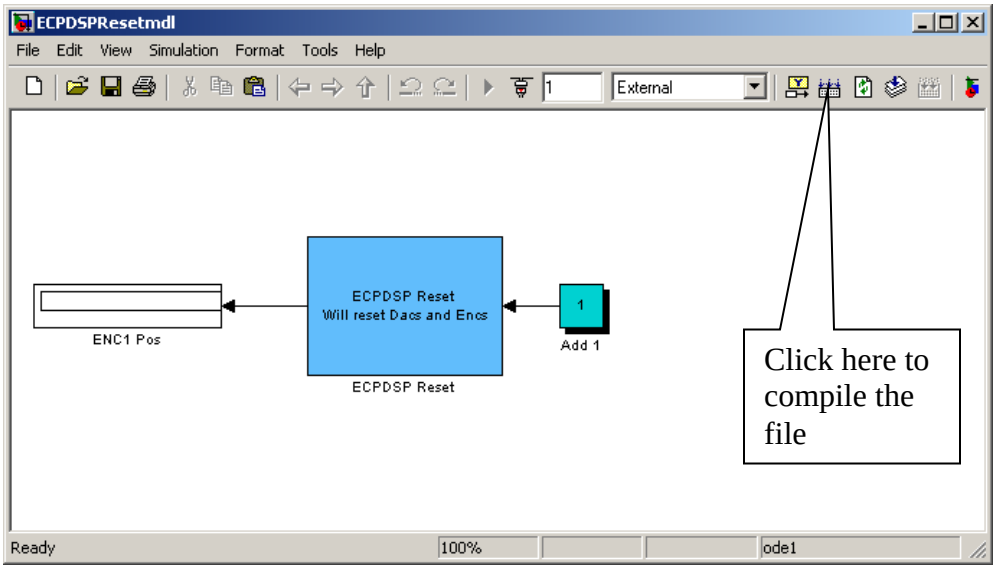


Figure 3: Compiling ECPDSPReset.mdl

9. **Connect** to the system as shown in Figure 4 and run the system by clicking **play** as shown in Figure 5.

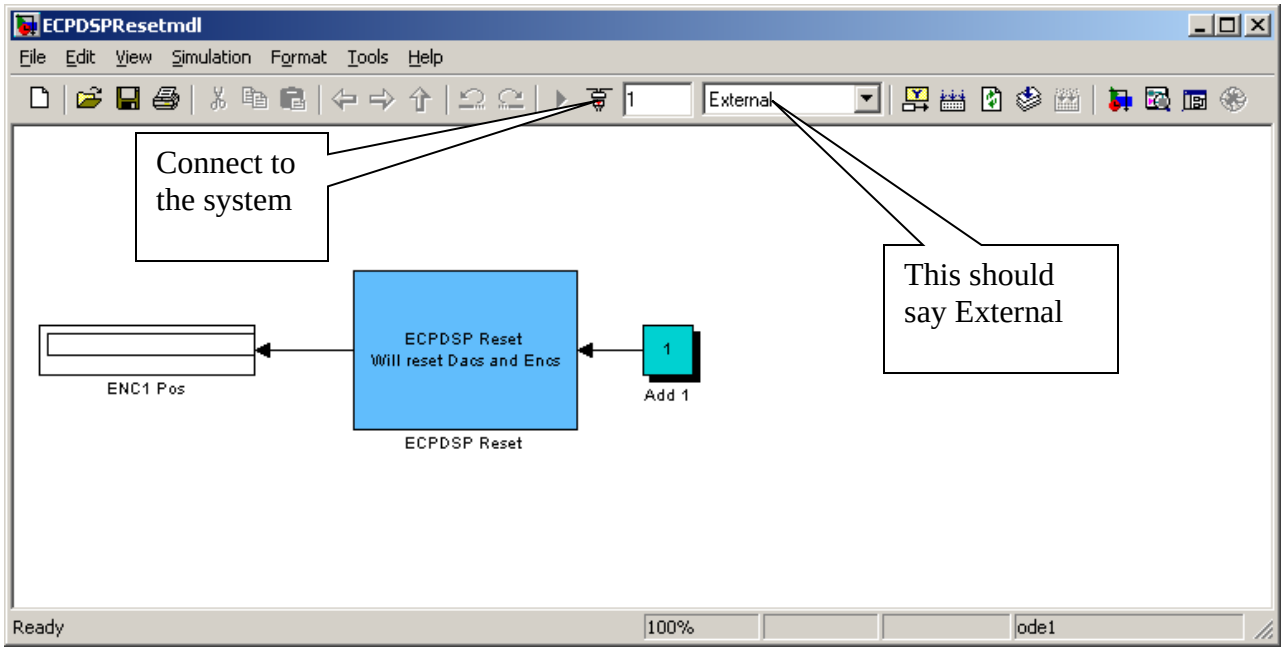


Figure 4: Connecting to the ECP system

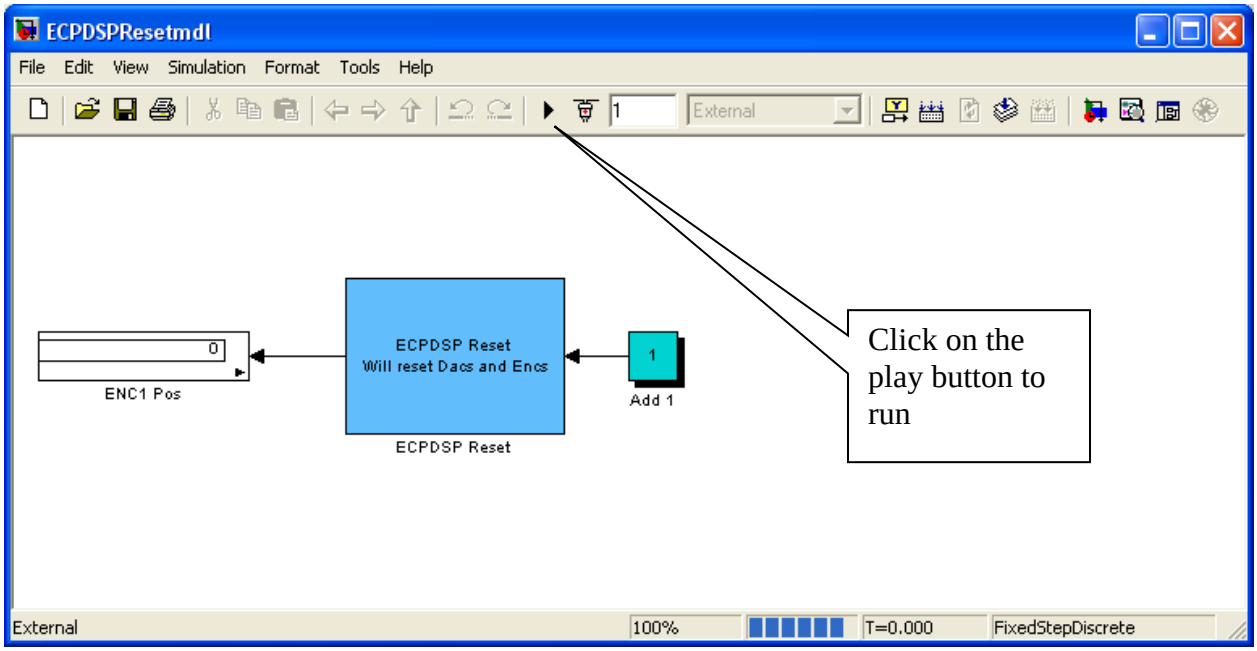


Figure 5: Running the ECP system

10. If you all of this has been done correctly, the MATLAB command window will indicate that you have connected and disconnected from the system (loaded, unloaded). This indicates that the counters have been reset and the system has been zeroed.

PART III - Second order rectilinear system (ECP Model 210)

1. In this part you will set up the mechanical system that you will create the model for. You will need to go through the following steps for **two different** configurations. This will be two rectilinear systems with different masses and/or springs, with or without the damper.
2. Use the Allen wrenches at the bench to lock all of the carts except for the first one closest to the motor. In addition, you need to have at least one spring connected to the cart and at least one mass on the cart.
3. Open up the Simulink file, **Model210_Openloop.mdl**, it should look like Figure 6**.** The yellow block in the middle is what actually connects to the ECP system. Note that the output of this system is labeled x1, the cart position. **Do not change this!**

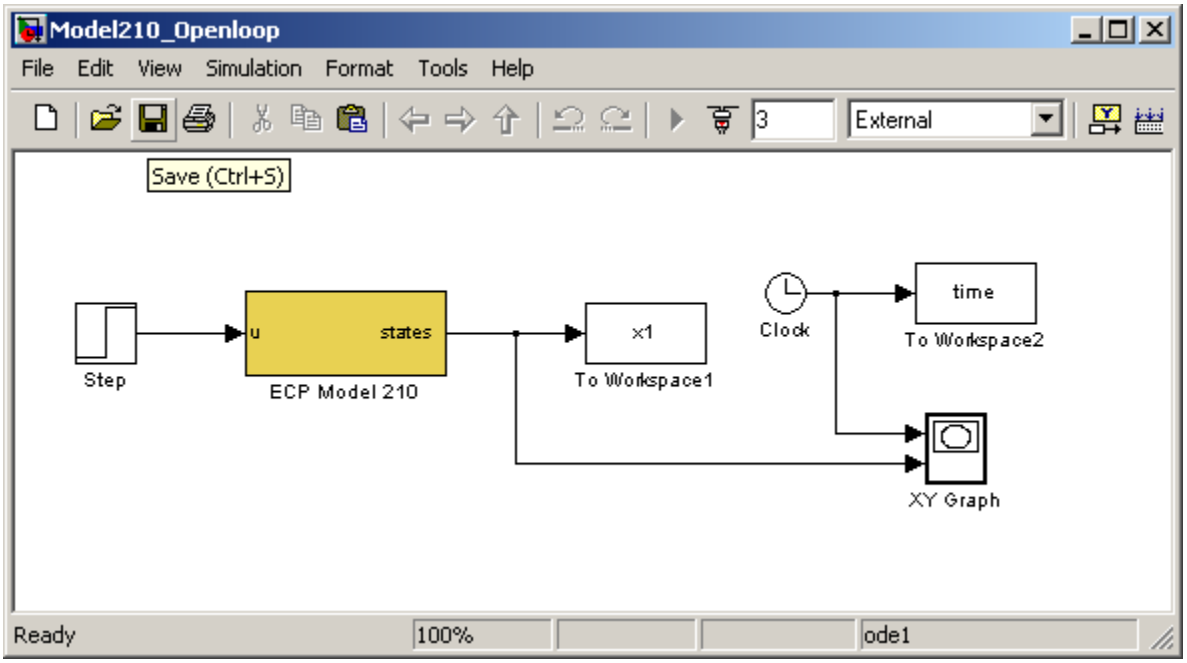


Figure 6: Model210_Openloop.mdl

4. You should also verify that the **Base I/O Address** for this system is set correctly for the Dell computers. You can do this by double clicking on the yellow block and the window in Figure 7 will open up. Double click on the blue ECPDSP Driver block to confirm that the Base I/O Address is correct. Click Apply->OK and then close the windows that were opened and save the **Model210_Openloop.mdl** file.

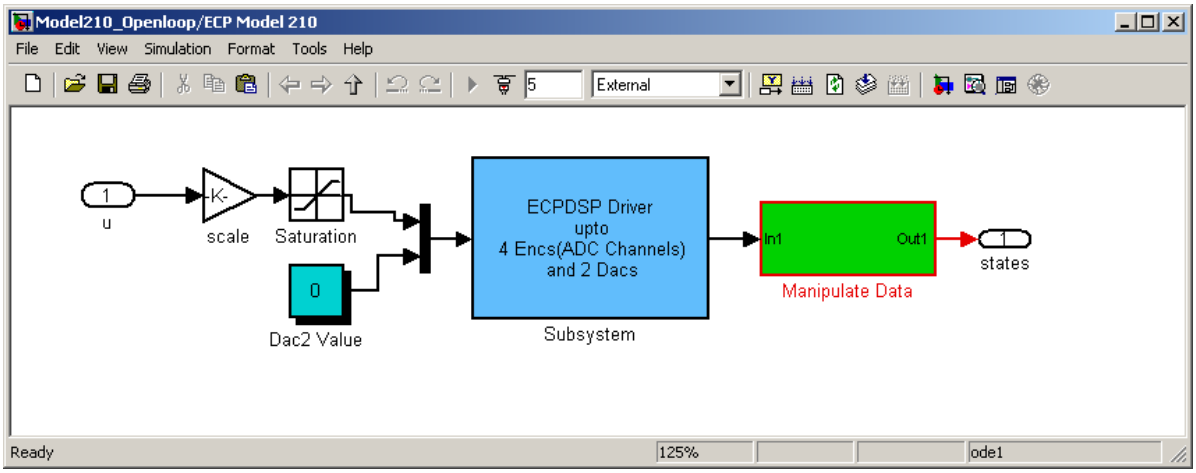


Figure 7: Model210_Openloop/ECP model 210

5. Change the input to the system by double clicking on the **Step** block in Figure 6. To start, enter a small step amplitude or input value such as **0.05** This value is small because some of the systems have a static gain of more than 40 and the cart should not move more than a couple of centimeters. ***You may have to increase the step input to get your particular system to have a steady state value of at least 1 cm.*** Click **OK** and then save the file.
6. **Compile** the file just as you did in PART II and wait for the messages to finish in the MATLAB window. Next, **connect** to the system, and then **run** it and the cart should move. Ignore any errors that open up and continue. There are three (usual) outcomes here:
   a) If the cart does not move, repeat PART II. You don't need to change the Base I/O Address or recompile, but you do need to reload the controller personality file and reset the system.
   b) If the cart moved, but the XY Graph does not look smooth, the communications are still not correctly set up, repeat part II. You don't need to change the Base I/O Address or recompile, but you do need to reload the controller personality file and reset the system.
   c) If the cart moved and the XY Graph looks reasonable, be sure the system recorded data for three seconds. If it did not record for three seconds, first look at the output level. If the maximum value of the output is between 1 and 2 cm (or larger), try resetting (ECPDSPREset) and rerunning the system a few times. If the maximum value of the output is less than about 1.0 cm, then increase the value of the amplitude in the **Step**, reset the system (ECPDSPReset) and rerun the system (Model210_Openloop).

Before you go on, be sure your system has run for 2-3 seconds and recorded enough good data.

7. Modify the file **second_order_driver.m** to plot the data from the ECP system and the results from the simulation on the same graph. Be sure to use different line types and a legend. There are two different time arrays (one from the simulation and one from the ECP system), and two different outputs. You may need to use **axis** command to limit the time axis if your ECP system did not run long enough. Also, be sure the y-axis indicates the output is displacement (in cm).

8. Modify the values of the static gain, then the damping ratio, and then the natural frequency to try and get the model to match the system as closely as possible during the first part of the step (it is likely to get worse as time progresses.) You will have to iterate a bit on this. Once you have a reasonably good fit, copy the MATLAB figure into a word file (use **Edit**, then **Copy Figure**).
9. Before you go on, rename the variables from the ECP system so that you can use them again later, to do this type the following in the MATLAB command window,

   time_save = time;

   x1_save = x1;

10. Before you go on, Include step response graph in your lab memo submission with a figure number and caption referenced in the text.

PART IV - The log decrement method

In this part, you will use the log decrement method to estimate the natural frequency, ω_n and damping ratio, ζ, of a second order system. Note that this method does not determine the static gain.

1) Since the log-decrement algorithm is so useful, it has been automated and you can implement it by using the following steps:

   - Reset the system using **ECPDSPresetmdl.mdl.**
   - Modify **Model210_Openloop.mdl** so the input has **zero** amplitude.
   - Compile **Model210_Openloop.mdl** if necessary.
   - Connect **Model210_Openloop.mdl** to the ECP system. (The mode should be **External**.)
   - Displace the first mass, and hold it.
   - Start **(play) Model210_Openloop.mdl** and let the mass go.
   - After the cart finishes moving, run the m-file, **log_dec.m.** This file should be in the same directory as **Model210_Openloop.mdl** and **log_dec.fig**. This routine assumes the position of the first cart is labeled x1 and the time is labeled time. (These are the defaults in **Model210_Openloop.mdl**)

- The program **log_dec** produces the GUI shown in Figure 8

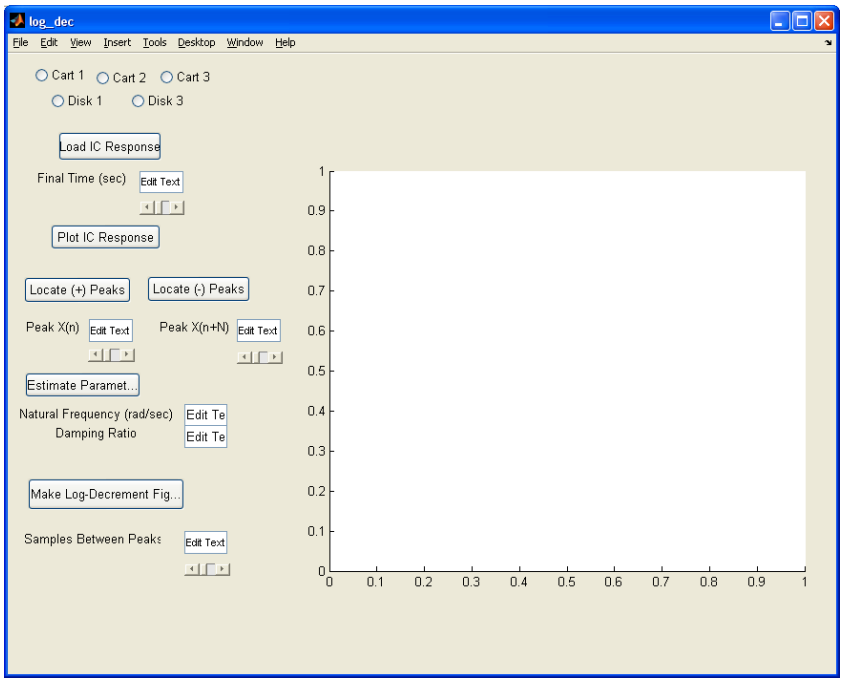


Figure 8: Model210_Openloop/ECP model 210

You need to

- Select **Cart 1**
- Select **Load IC (initial condition) Response** (the variables time and x1 or theta1 will be loaded from the workspace). At this point some initial estimates will be made.
- Set/modify the **Final Time**
- Select **Plot IC Response** to plot the initial condition response
- Choose to identify the positive peaks (**Locate + Peaks**) or negative peaks (**Locate - Peaks**). If the peaks are not numbered consecutively, you need to decrease the **Samples Between Peaks** and try again until all peaks have been identified.
- Choose the initial peak (**Peak x(n)**) and final peak (**Peak x(n+N)**) to use in the log-decrement analysis. These should be fairly close to the beginning of the initial condition response. Don't try and use more than a few peaks.
- Select **Estimate Parameters** to get the initial estimates of □ and □$_n$

- Select **Make Log-Decrement Figure** to get a plot and summary of the results. You need to include this figure in your memo.

2) Use the values of the natural frequency and damping ratio determined using the log decrement method in your simulation. Before you run your simulation you will need to recover the saved variables by typing in the MATLAB command window by typing the following:

   time = time_save;

   x1 = x1_save;

3) Include the log decrement graph in your lab memo submission with a figure number and caption referenced in the text. This means you should now have 3 graphs to submit for the system that you built.

PART V - Model a different second order system

In this part, you will repeat Parts III and IV of the above procedure on a different second order system.

Your memo should contain **3 graphs** for each of the systems built, **six total**. Each graph should include a figure number, caption, descriptive title, and axes labeled with units and referenced in the text. The body of the memo should be a very brief procedure no more than one paragraph and data tables comparing the values used trying to match the step response and using the log decrement method and what you may think caused any differences.

Submission:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject
- Written in first person from you
- Written with minimal spelling and grammar errors

- Purpose, procedure, results and conclusions of the laboratory experiment (the procedure should be very short, a high level summary of what you did for each part). The procedure should be two paragraphs at the most.
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text
- Also, you must to include a statement in your memo that this is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results

ECE-205 Lab 4

System Linearity

Overview

For many engineering systems, the system can be modeled as linear only over a specified region of operation. For example, the operational amplifier only operates when its linear region when the output is between the positive and negative supplies otherwise it is in saturation. In this lab you will construct the simple common emitter BJT amplifier shown in Figure 1. Then you will determine the range of input signals for which the circuit can be modeled as linear. Note that this is not a well-designed amplifier circuit, but it is easy enough for us to build and examine to learn about system linearity.

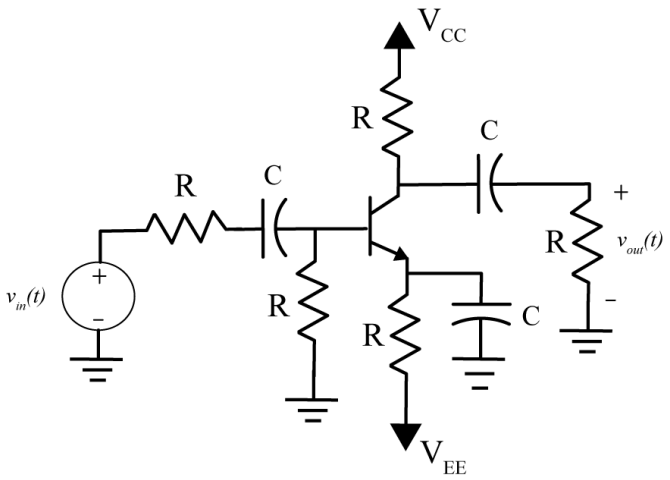


Figure 1: Simple common emitter amplifier.

Prelab

Read this entire lab procedure and background theory thoroughly. Then complete the following simulation using MultiSIM and submit it in class the day before the lab session.

1. Place the following components in the MultiSim drawing: BJT_NPN_VIRTUAL, CAPACITOR_POL_RATED, RESISTOR_RATED, GROUND
2. You will need 4 - 1 kΩ resistors, 3 - 1 μF, 2 - voltage sources (15 V and -15 V)
3. Use the components to create the common emitter amplifier circuit shown in Figure 1.

4. Connect the NI ELVISmx Function Generator to the input of the circuit. Connect Channel 0 of the NI ELVISmx Oscilloscope to the input to the circuit. Connect Channel 1 of the NI ELVISmx Oscilloscope to the output of the circuit. Set the function generator to a 3 Vpp 100 Hz sine wave. Your result should look similar to Figure A1 when you are finished.

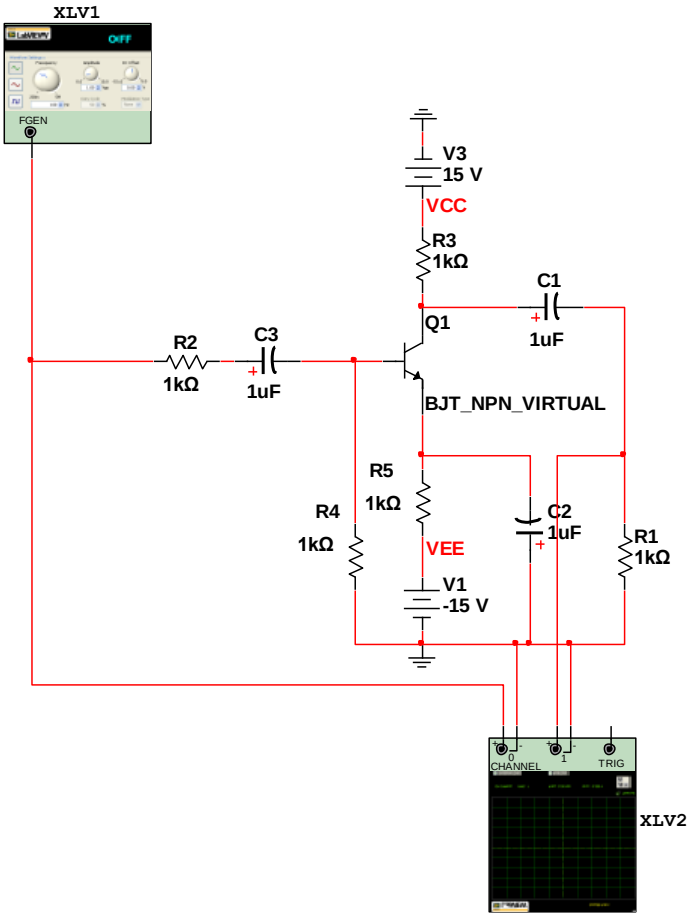


Figure A1: MultiSim Common Emitter Amplifier Circuit

5. Since in this lab you will examine the nonlinear characteristics of the common emitter amplifier, you will examine the input and output characteristics at two different input voltages. Make the waveforms as large as possible in the oscilloscope and capture the screen for inclusion in your prelab submission (see Figure A2). Using the maximum peak positive values of each waveform, what is the apparent gain (**output/input**) of the circuit?

6. Next, increase the input amplitude on the function generator until the output appears distorted. This represents the nonlinear characteristic of the amplifier circuit. Using the peak positive values of each waveform, what is the apparent gain? Is it the same as before?
7. Submit the MultiSim schematic and the screenshots before and after distortion for your prelab submission. Make sure the schematic has your name on it as a text box to indicate that it is your work.

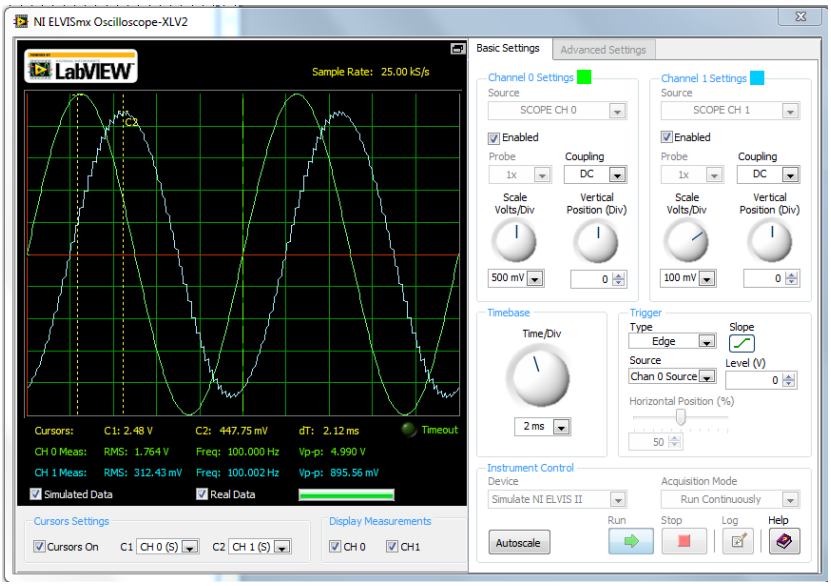


Figure A2: Common Emitter Amplifier Output

Equipment

- 100 μF electrolytic capacitors (x 3)
- 1 kΩ resistors (x 5)
- 2N-2222A-836 NPN BJT

Procedure

PART I: Build the common emitter circuit

In this part you will build the circuit in two stages. DO NOT try to build the circuit all at once. The electrolytic capacitors *must be connected with the correct polarity,* so be careful to read all of the instructions.

1) Set up the power rails on the breadboard. Use a wire from your kit to tie both of the blue buss (common) rails together and connect them to ground on the NI MyDAQ (AGND) next to the -15V terminal. Connect the red buss across the top to the +15V terminal on the NI MyDAQ. Connect the bottom red buss on the breadboard to the -15V terminal on the NI MyDAQ.
2) The NPN BJT is represented by the symbol shown in Figure 2, and it has three terminals. If you hold the BJT with the flat part to your right, then the collector is on top, the base is in the middle, and the emitter is on the bottom.

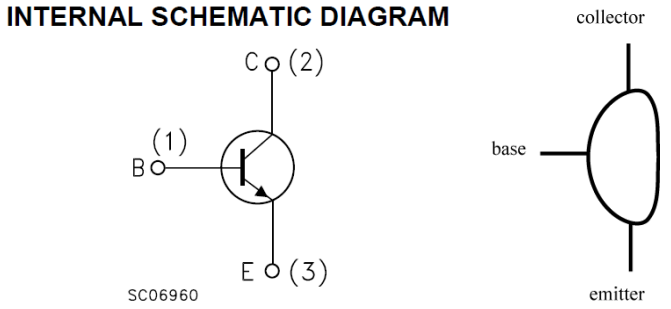


Figure 2: NPN BJT Internal schematic (left) and physical device (right)

3) Next, build the circuit subsystem shown in Figure 3. Note that the collector is connected to the +15 V source, V_{CC}, through a 1 kΩ resistor. The emitter is connected to the -15 V source, V_{EE}, through a 1 kΩ resistor. The base is connected though a resistor to ground. The collector is also connected through an electrolytic capacitor (note the negative sign!) and resistor to ground. There is a minus sign on the capacitor can and the shorter leg represents the negative terminal.

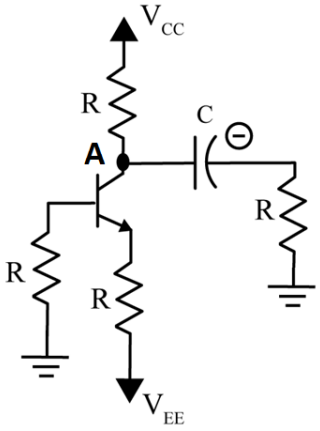


Figure 3: Biasing the transistor circuit.

4) To verify that the circuit is biased correctly, start the NI MyDAQ Digital Multimeter and set it to measure DC Voltage. Measure between node A and ground as shown on the circuit in Figure 3. If you have done everything correctly, this measurement should be approximately **3V**. **Do not move on until this is correct!**

5) Now finish building the circuit as shown in Figure 4. Verify that the polarity is correct on all of the capacitors. In addition, confirm that the remaining items are all connected to ground and not any of the voltage sources.

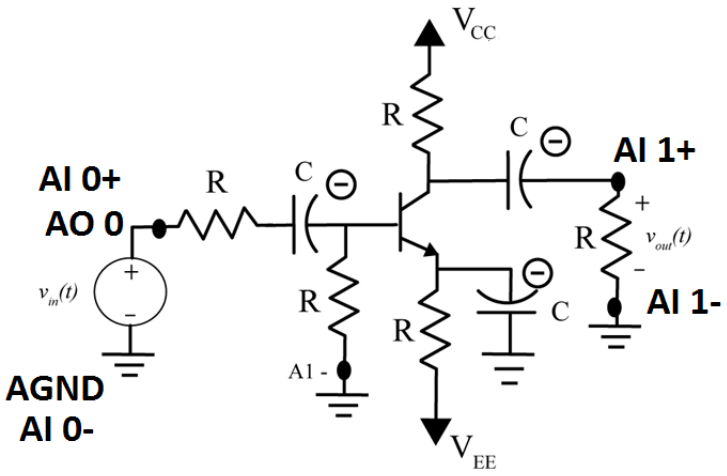


Figure 4: Common Emitter Amplifier

6) Set the function generator to 100 Hz, 1 Vpp sine wave and connect it to the input of the circuit, $v_{in}(t)$. Also connect channel 0 of the oscilloscope to the input of the circuit. Connect channel 1 of the oscilloscope to the output, $v_{out}(t)$. Set both channels of the oscilloscope to 500 mV/div and the timebase scale to 2 ms/div.

7) Use the cursors to measure the *peak positive amplitude* of the input and output signals (**do not** measure peak-to-peak!). The ratio of the output magnitude to the input magnitude is the gain and it should be between 2 and 4. If it is not, check your circuit and then ask for help. Now we are ready to make circuit measurements.

PART II: Measuring Input, Output and Gain for the Amplifier

1. This circuit can only be modeled as a linear system for a limited range of input signals. In this part, you will measure the input and output amplitudes as the input amplitude changes. Assume that the input signal is a sine wave with a frequency of 100 Hz for all of the measurements. Make sure that the function generator has a 0 VDC offset before you start taking the measurements. Set the peak to peak voltage on the function generator to 200 mV.
2. Use the cursors on the oscilloscope to measure the input and output amplitude and the gain of the common emitter amplifier. Be sure to measure only the **maximum** *peak positive amplitudes* of the output signal, **do not** measure peak-to-peak amplitudes or RMS. Adjust the Volts/Div scale in order to make the waveform as large as possible on the screen so that your measurements are most accurate.
3. Measure the input amplitude, output amplitude and gain of the amplifier as the peak to peak voltage on the function generator is increased from 200 mV to 4.6 V in increments of 200 mV. *Note that as the input increases, the output will begin to look more distorted and less like a sinusoid due to saturation and nonlinearities.* However, continue to only measure the ***maximum peak positive voltage*** for the input and output using the cursors. Since you will need to plot this data in MATLAB in the next section, you should record the measured amplitudes in Excel and use a formula to calculate the gain as you increase the input.
4. You should also include **two screen captures** of the oscilloscope screen in your lab memo submission: one with the sinusoidal input and output and one that shows the nonlinearities in the output signal. Do not copy the entire window just the oscilloscope screen. Make sure the figures have a figure number and caption and are referenced in the text of the memo.

PART III: Plotting the Amplifier Data

a. Start MATLAB and create a new m-file (File->New->Script) [CTRL-N]. Give the file a name such as “Uhura_Lab4.m”.
b. Use the % to add a comment at the top of the file with the filename, your name, a brief description of the program and the date.

c. Create an array of the input amplitude data and an array of the output amplitude data in the m-file. You can do this by typing, **Vin = [**, then copy the column vector from Excel and then type, **];**. You should repeat this for the variable, **Vout**.

d. Plot the output versus the input amplitudes by typing **plot(Vin, Vout, 'o')**. Make sure the graph has a grid, descriptive title and axis labels with units. Use the commands **grid, title(), xlabel(), ylabel(),** in order to do this. You should type **help** and the function name in the MATLAB command window if you are not sure how to do this. Your plot should be similar to Figure 5.

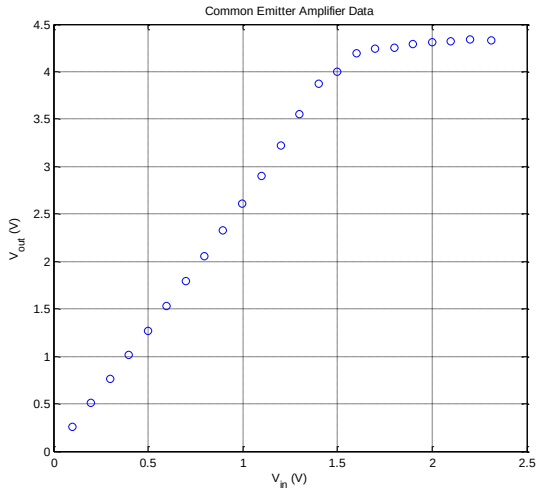


Figure 5: Emitter measured input/output data.

e. Next, you will fit a least squares straight line approximation between the points that show a linear input/output relationship. Use the MATLAB command, **polyfit** and insert the following code into the m-file. Note that this code assumes that your input and output data arrays are named **Vin** and **Vout**. In addition, this code is written assuming that you will create a least squares line that fits the first **N data** points which has been selected to be 10:

```
N = 10;  % select an N value close to the number of linear data points
p = polyfit(Vin(1:N),Vout(1:N),1);
m = p(1); b = p(2);
est_output = m*Vin+b;
plot(Vin,Vout,'o',Vin,est_output);grid;
```

f. Your plot will look similar to Figure 6 and you should iteratively adjust the parameter **N** to include as many points necessary to generate a reasonably straight line. Use the MATLAB Edit -> **Copy Figure** command to make a copy of this plot for inclusion in your lab memo. . This figure should have a figure number and caption referenced in the text.

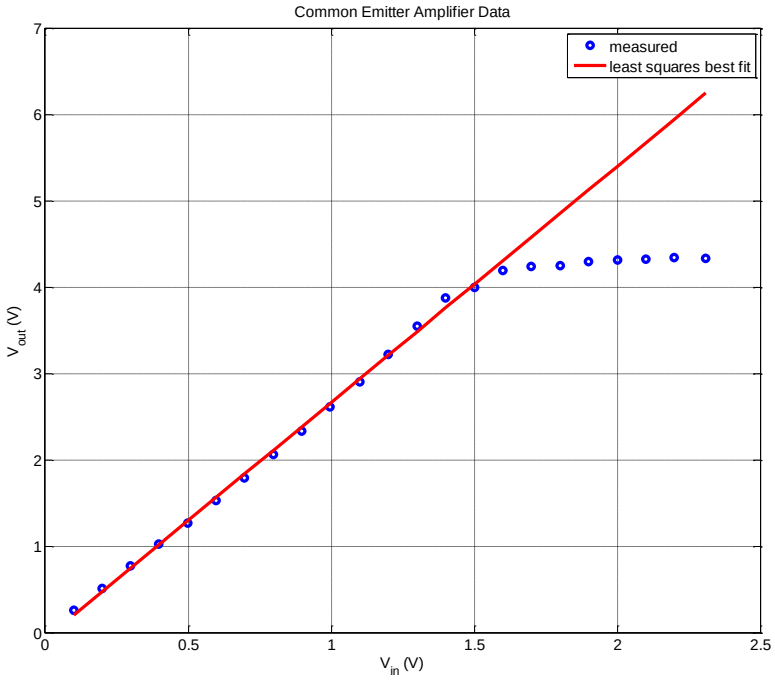


Figure 6: Measured data and least squares fit

g. Copy your m-file and include it in the Appendix of your lab memo.

Your memo should contain 3 graphs: one MATLAB graph and two screen captures. Your memo should also include your MATLAB m-file.

Submission:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- Date, To, From, Subject

- Written in first person from you
- Written with minimal spelling and grammar errors
- Purpose, procedure, results and conclusions of the laboratory experiment (the procedure should be very short, a high level summary of what you did for each part). The procedure should be two paragraphs at the most.
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text
- Also, you must to include a statement in your memo that this is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results

ECE-205 Lab 6

Transfer Functions and an Optical Transmitter and Receiver

Overview:

The purpose of this lab is to examine the transfer functions of systems and circuits in MultiSim, MATLAB and Simulink. Then, you will build an optical transmitter and receiver circuit to examine the input/output characteristics.

Equipment:

IR LED Emitter

IR LED phototransistor receiver

51 Ω resistor

10 kΩ resistor

1 kΩ resistor

Prelab:

Read the background theory and entire lab procedure thoroughly. Then complete the following analytical calculations. Submit the prelab in class the day before the lab session.

1. Show that a first-order system represented by the following differential equation has the given transfer function, *H(s). (Note: you will need to look up the Laplace transform of a derivative to do this. Remember to assume zero initial conditions.)*

$$\tau\dot{y}(t) + y(t) = Kx(t)$$

$$H(s) = \frac{Y(s)}{X(s)} = \frac{K}{\tau s + 1}$$

2. Show that a second-order system represented by the following differential equation has the given transfer function, H(s).

$$\ddot{y}(t) + 2\zeta\omega_n\dot{y}(t) + \omega_n^2 y(t) = K\omega_n^2 x(t)$$

$$H(s) = \frac{Y(s)}{X(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

3. Use the following components to build the circuit in Figure 1 in MultiSim (GROUND, DC_POWER, RESISTOR_RATED). Next, run the transfer function analysis in Multisim by clicking `Simulate→Analyses→Transfer Function…`. Remember you have to double click on the wires to name them `Vin` and `Vout`. The input should be the DC voltage source, `V1`, and the output node is, `Vout` and the reference node is ground, `V(0)`. What were the results? What do you think these three values represent? You should submit the system output, **not the Multisim file**, in your prelab submission.

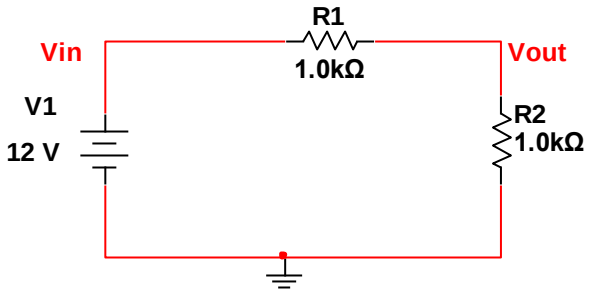
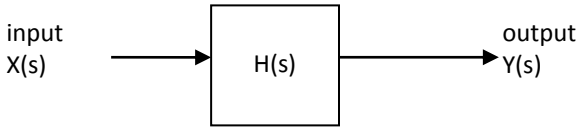


Figure 1: MultiSim Circuit for Transfer Function Analysis

Theory:

In all of the prior labs, systems were represented in terms of the differential equation. However, it is also useful to examine the frequency characteristics of a system. The transfer function is used to represent a system based upon the frequency domain. The **transfer function** of a system is defined as the s-domain ratio of the Laplace transform of the output (response) to the Laplace transform of the input (source). To find the transfer function of a system, all of the **initial conditions must be zero**. In order to determine the transfer function of a system, apply the following steps:

1) Assume that the system has zero initial conditions.
2) If the input is x(t) and the output is y(t), apply the Laplace transform, $\mathcal{L}\{x(t)\} = X(s)$ and $\mathcal{L}\{y(t)\} = Y(s)$
3) The transfer function is the ratio of the output to the input, **H(s) = Y(s)/X(s)**



One benefit of using Laplace transforms to analyze systems is that you can use algebra to solve for system parameters instead of differential equations. Note that this is similar to phasor analysis except the system may not be at sinusoidal steady-state and the complex frequency is $s = j\omega$ instead of $s = \sigma + j\omega$.

In the time domain, the system output is found from the convolution, **y(t) = h(t) * x(t)**

In the frequency domain, the system output is found form the product, **Y(s) = H(s)X(s)**

Procedure:

PART I: Transfer Functions in MATLAB

In MATLAB a polynomial such as p(s) = $3s^2 + 2s + 1$ is represented by the following syntax,

```
p = [3 2 1];
```

Note that the semi-colon suppresses printing the output to the command window. The array indicates that the highest power of s is on the left and the lowest power, s^0 is on the right. You should always right the array with respect to the coefficients for descending powers of s. There must be a number for each power of so if one is missing then use a zero.

For example, to enter the polynomial p(s) = $2s^4 + 3s$ into MATLAB, you would type,

```
p = [2 0 0 3 0];
```

Typically transfer functions are the ratio of two polynomials and they can be created by using the **`tf`** command.

For example to create the transfer function, $H(s) = \frac{s+1}{3s^2+2s+1}$, enter the following code into MATLAB

```
num = [1 1];%numerator polynomial of the transfer function
den = [3 2 1];%denominator polynomial of the transfer function
H = tf(num,den);%construct the transfer function
```

This could also be reduced to one command by writing the following:

```
H = tf([ 1 1],[3 2 1]);%system transfer function
```

If you do not include the semicolon at the end of the commands, MATLAB will print the transfer function to the workspace. Please do this to confirm that the transfer function has been entered correctly.

PART II: System Response in MATLAB

In this section, you will use MATLAB to determine the response of a system represented by a transfer function. The input may be a step or an arbitrary input.

1. Create a new MATLAB m-file (script) file [Ctrl+N] and place it in a lab 6 folder.

2. Insert a header comment by using the % that includes the filename, your name, date created and a brief description of what the program does, see the following example:

```
%{
Lab6.m
Uhura Jones
12/20/55
Transfer function analysis
%}
```

3. Add the following command line to clear all variables

```
clear variables;
```

4. Create the time constant and static gain variables and give them the following values,

```
tau = 0.001;
K = 2.0;
```

5. Use the steps from PART I to create the plant transfer function, $G_p = \frac{K}{\tau s + 1}$

6. Set the final time variable to `Tf = 10*tau;` (be sure to use at least one capitol letter so it is not confused with the transfer function command `tf`).

7. Use the `linspace` command to create a time vector `t` from **0** to `Tf` with 1000 sample points. If you don't recall how to do this type, `help linspace` in the MATLAB command window.

8. Use the following command to create a step input, x(t) = Au(t) with an amplitude of 0.1.

```
A = 0.1;
x = A*ones(1,length(t)); %step function with amplitude, A
```

9. Next, use the `lsim` command to simulate the system output.

```
y = lsim(Gp,x,t);
```

10. Finally, create a plot of the output y(t) as a function of time, it should be similar to Figure 2. If you don't recall how to do this, please type `help plot` in the MATLAB command window. Make sure that your plot has a title with your initials and the axis labeled.

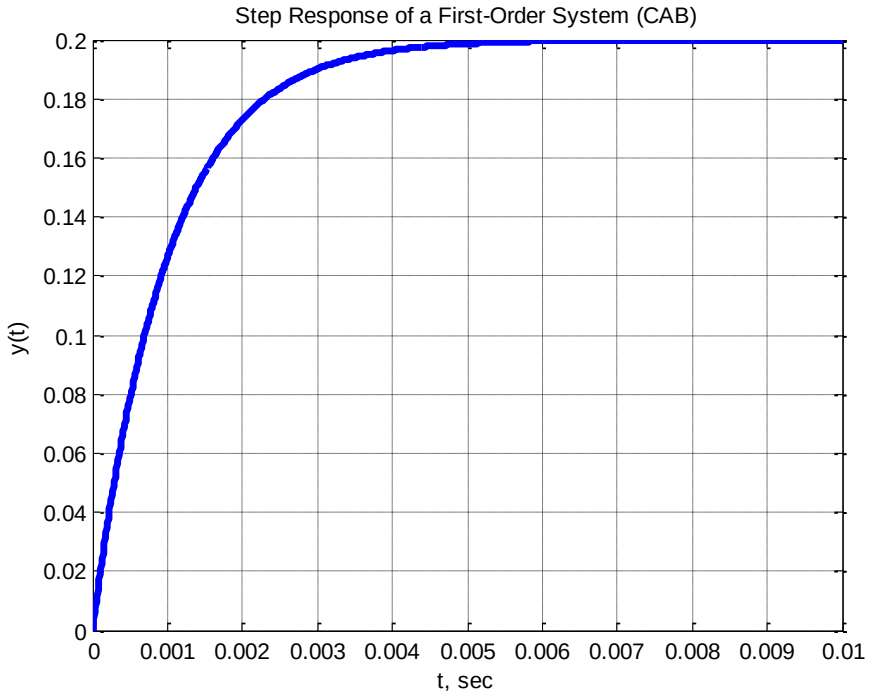


Figure 2: Step Response of a First-Order System

PART III: System Response in Simulink

1. In order to use Simulink to determine the response of a system given the transfer function it is necessary to put the following time and input values from MATLAB in the script file,

```
ut = [ t' x'];
```

2. In order to extract the numerator and denominator of a transfer function from a given transfer function, use the following

```
[num_Gp,den_Gp] = tfdata(Gp,'v');
```

3. Start `simulink` from the MATLAB command window and create a new model file [Ctrl+N] and save the model file as **openLoop.mdl.**
4. Construct the model file shown in Figure 3. You will need to look in the **Sources Library** for the ***from workspace*** block and the **clock** block
5. Go to the **Sink Library** for the ***to workspace*** blocks. Make sure to insert two of them and double-click on them and make the save format as ***array***.
6. Go to the **Continuous Library** for the ***transfer function*** block. Click on the transfer function block to enter the numerator as **num_Gp** and the denominator as **den_Gp.**

7. Change the final time of the simulation to **Tf** as shown in Figure 3.

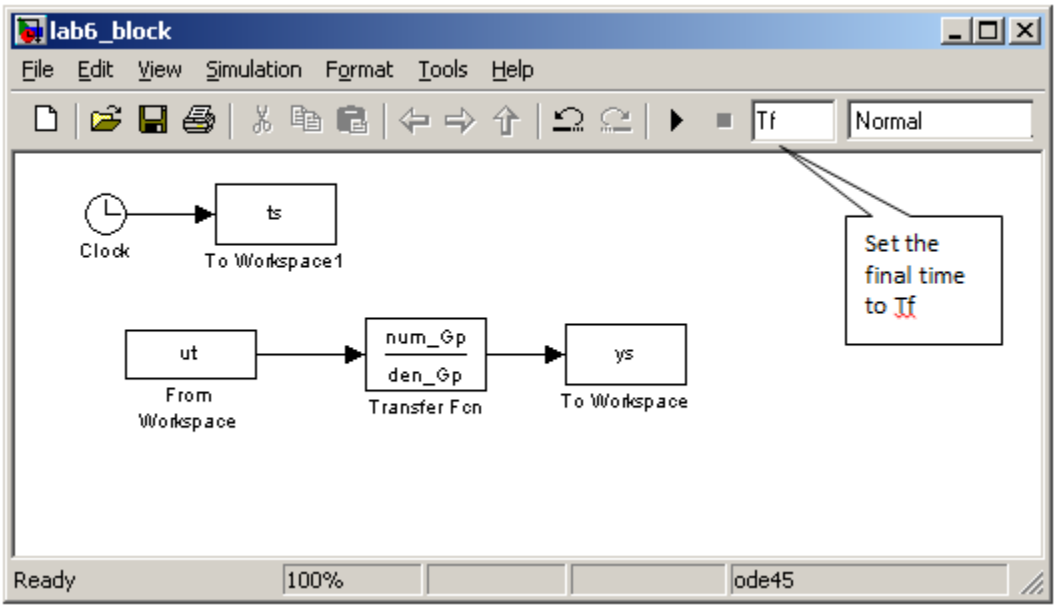


Figure 3: Open Loop Simulink model file

8. Run the simulation from the m-file using the following command,

```
sim('openLoop');
```

9. Modify the m-file (script) file to plot both the results from both MATLAB (`t, y`) and SIMULINK (`ts,ys`) on the same graph. Make sure that you include a descriptive title with your initials, label the x and y axes, include a legend and use two different line types. Your final graph should look similar to Figure 4. *You need to include your graph in your memo, but not your code. The m-file should be included in your lab submission as a separate file.*
10. Now create a new m-file (script) to plot the MATLAB and Simulink results for a second order system. You should start the new code by copying and pasting everything from the file created for the first order system and then modify it.
11. Create new variables for the natural frequency, ω_n, and damping ratio, ζ. The system has a natural frequency of **2000** rad/sec, a damping ratio of **0.2**, and a static gain of **2.0**. The step input amplitude is still **A = 0.1**. If you don't remember how to create the second order transfer function, review the prelab and PART I. Simulate the second order system and plot the results. If you have done everything correctly, you should get a graph like that shown in

Figure 5. *You need to include your graph in your memo, but not your code. The m-file should be included in your lab submission as a separate file.*

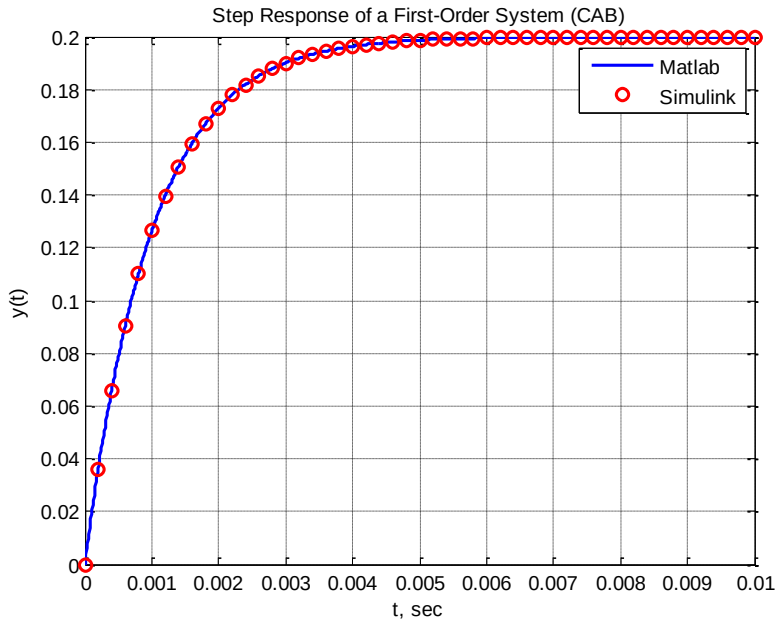


Figure 4: Matlab and Simulink first-order system results

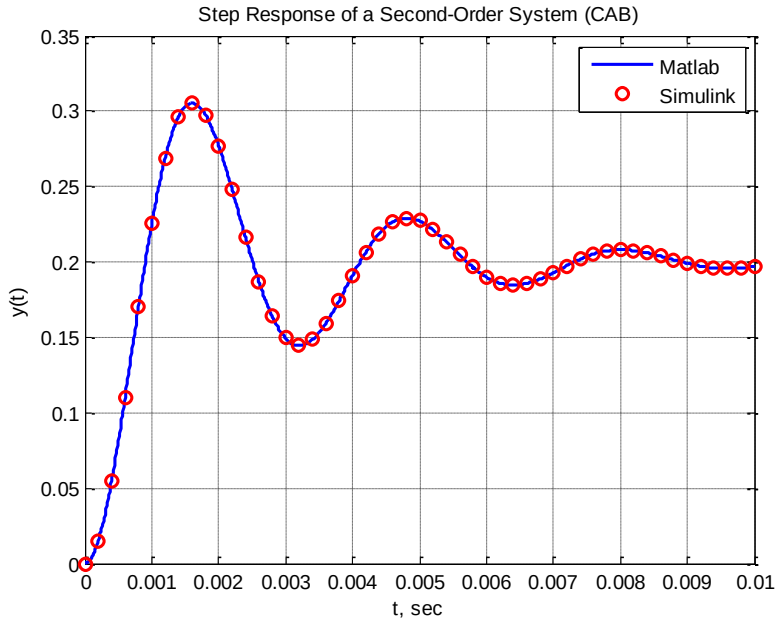


Figure 5: MATLAB and Simulink second order system results

PART IV: Building an Optical Transmitter and Receiver

In this part of the lab, you we will build a simple optical transmitter and receiver that uses infrared light. This system is similar to how a TV remote control works. For the most part this should be fun, but we will also get practice wiring and using a TL072 chip. The transmitter and receiver are shown schematically in Figure 6. Do not start to build this circuit yet!

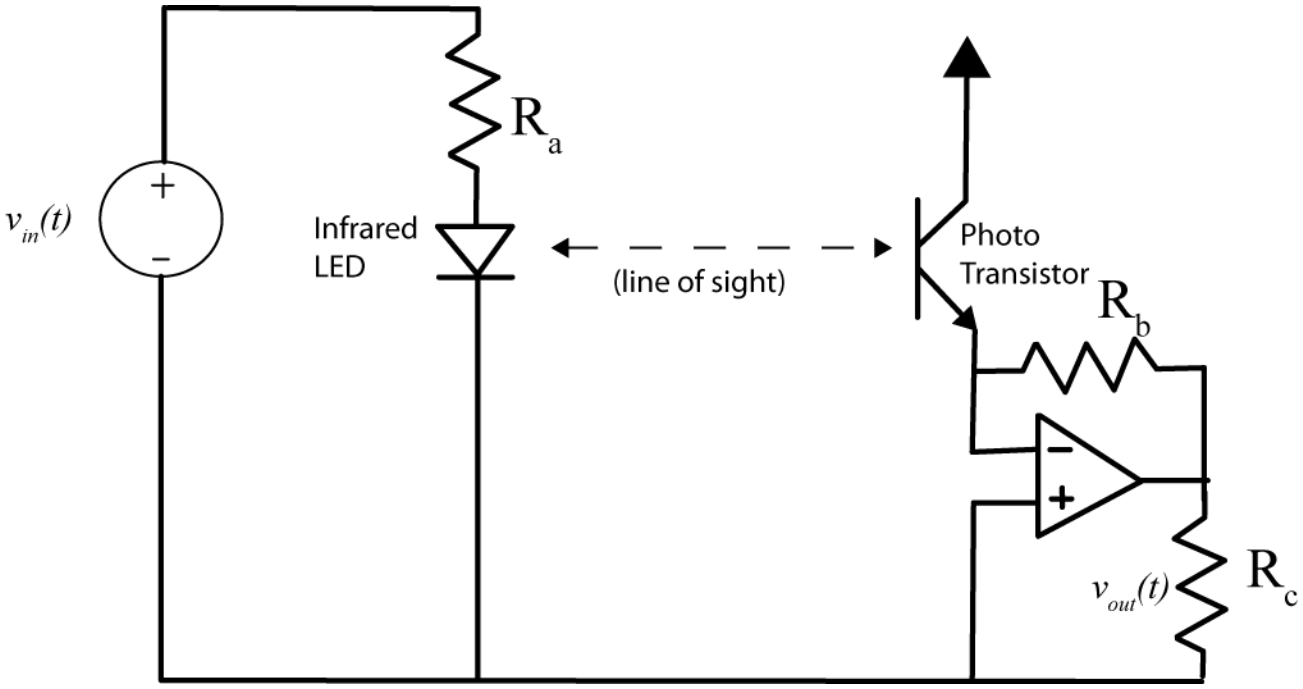


Figure 6: General schematic of an optical transmitter and receiver.

The transmitter on the left side of the schematic consists of an infrared LED. You will not be able to see when this is on, though you may be able to see it if you have a camera on your cell phone and look through that. The right side of the diagram shows the receiver, which consists primarily of a photo transistor and an amplifier.

1. You will create power and ground busses on your breadboard before you start building the circuit. Lay out your breadboard horizontal with the red stripe at the top and the blue stripe at the bottom. Use a wire to connect the two blue lines together which represent the common ground buss. Use a wire to connect the two blue lines for the ground. The top red line will serve as the positive (+Vcc) voltage rail, and the bottom red line will serve as the negative (-Vcc) voltage rail. Figure 6 provides a more detailed schematic of the optical transmitter/receiver circuit and you should refer to it as you build your system.

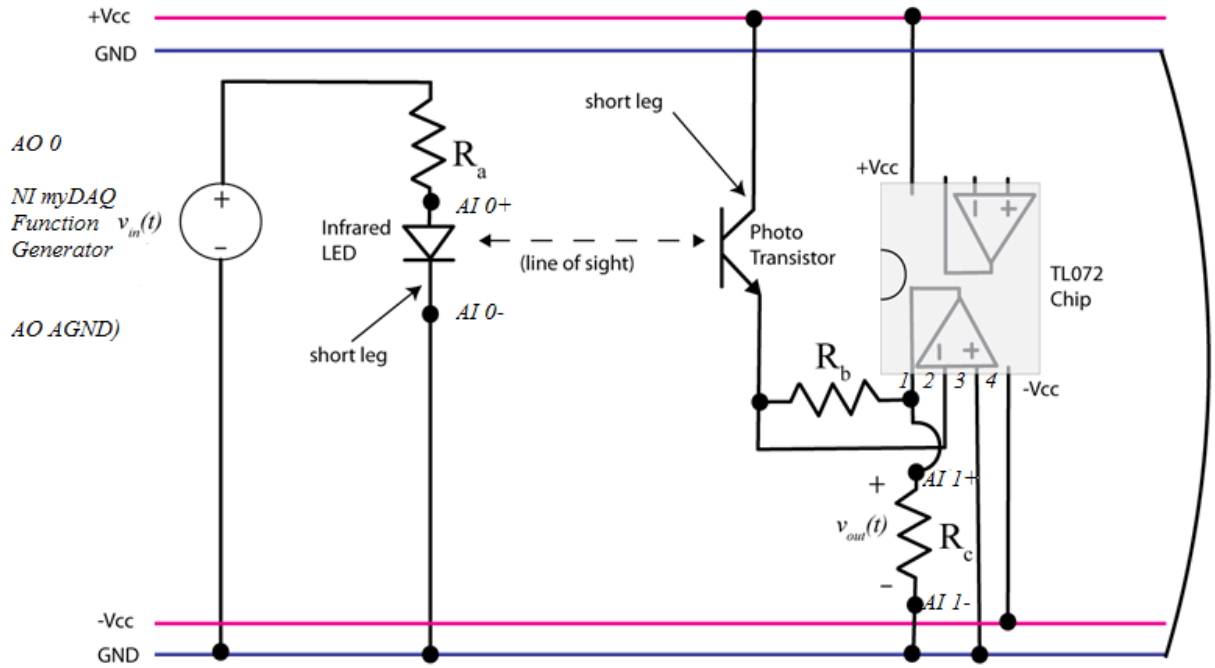


Figure 6: More detailed schematic of the optical transmitter and receiver

2. First, we will build the transmitter on the left of Figure 6. Start building as far as possible to the left side of the breadboard. Get the infrared LED (the clear LED) out of your kit and put it on the breadboard with the short leg on the ground buss. The short leg or the flat part of the rim of the cap on the LED represents ground.
3. Put the resistor, R_a = 51 Ω, in series with the infrared LED and connect the MyDAQ function generator (**AO 0, AO AGND**) as the input between the resistor and ground. Make sure that **AO AGND** is connected to the ground buss.
4. Finally, connect Channel 0 (**AI 0+ AI 0-)** of the oscilloscope to measure the voltage across the infrared LED. Make sure that **AI 0-** is connected to the ground buss. The transmitter portion of the circuit is complete.
5. Next we will build the receiver, which should be built as far as possible to the right on the breadboard. Note that we are using aTL072 chip which includes two op amps on one chip. Be sure that the dot on the chip is to the left (the semicircle on the drawing). The pin out for the TL072 dual op amp chip is shown in Figure 7.

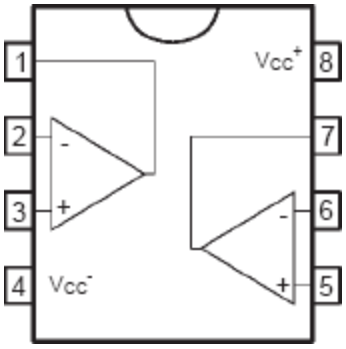


Figure 7: TL072 Dual Op Amp Pin Out

6. We will use the op amp on pins 1, 2, and 3 so connect pin 3 to **ground**. Connect pin 8 to the **+15V** or the red ground buss across the top of the breadboard. Connect pin 4 to the **-15 V** or the red buss across the bottom of the breadboard.
7. Connect the output of the op amp (**pin 1**) to R_c = 1 kΩ and then connect R_c to ground.
8. Also connect $R_b = 10\,\text{k}\Omega$ to the output of the op amp and it goes to the photo transistor receiver. The photo transistor receiver is black and looks similar to the infrared LED. The short leg of the photo transistor or the flat side of rim of the cap should be connected to the **+15 V** buss. Lastly, connect a wire from **pin 2** to the top of the phototransistor. Thus, the input to the op amp is the photo transistor and the op amp serves as a voltage follower and the measured signal of the photo transistor receiver is measured across resistor, R_C.
9. Finally, the output of the receiver circuit is the voltage across the output resistor, R_c, so connect Channel 1 of the oscilloscope (**AI 1+, AI 1-)** across the resistor. Make sure that **AI 1-** is connected the ground buss.
10. The transmitter is at the end or very tip of the LED, and the receiver is at the end or very tip of the phototransistor, so you need to bend these at a 90 degree angle so they are pointing toward each other. You will probably have to tweak the positions once you start sending the signal to adequately measure the receiving signal.
11. Now connect the (**+15V, -15V**) power supply to your circuit. Make sure that you put **+15V** on the top red buss and **-15V** on the bottom red buss and **AGND** on the blue ground buss.
12. Connect the NI MyDAQ to your laptop and open up the function generator and oscilloscope instruments.

13. The infrared LED requires a 5V signal so set the function generator to a **5 kHz square wave** with an offset of **2.5 volts** and a peak to peak value of **5 volts**.
14. Set the oscilloscope to measure **500 mV/Div** for both channels. Set the source on channel 0 to **AI 0** and the source on Channel 1 to **AI 1**. The time/div should be set to **100 µs**. Check enabled on both of the channels.
15. Press run on the function generator and the oscilloscope and your signals should look similar to Figure 8. If they do not then try adjusting the transmitter and receiver so they are pointing more directly at each other.
16. These signals indicate that the LED is transmitting when the voltage across it is high, and the phototransistor is receiving when the voltage across the output/load resistor is low. Set the trigger to **Edge** and the source is **Chan 0** and increase the level to get a stable output. Capture a screen shot of the oscilloscope image only for inclusion in your lab submission.
17. You have now finished with lab 6 and should read the submission requirements before submitting your lab.

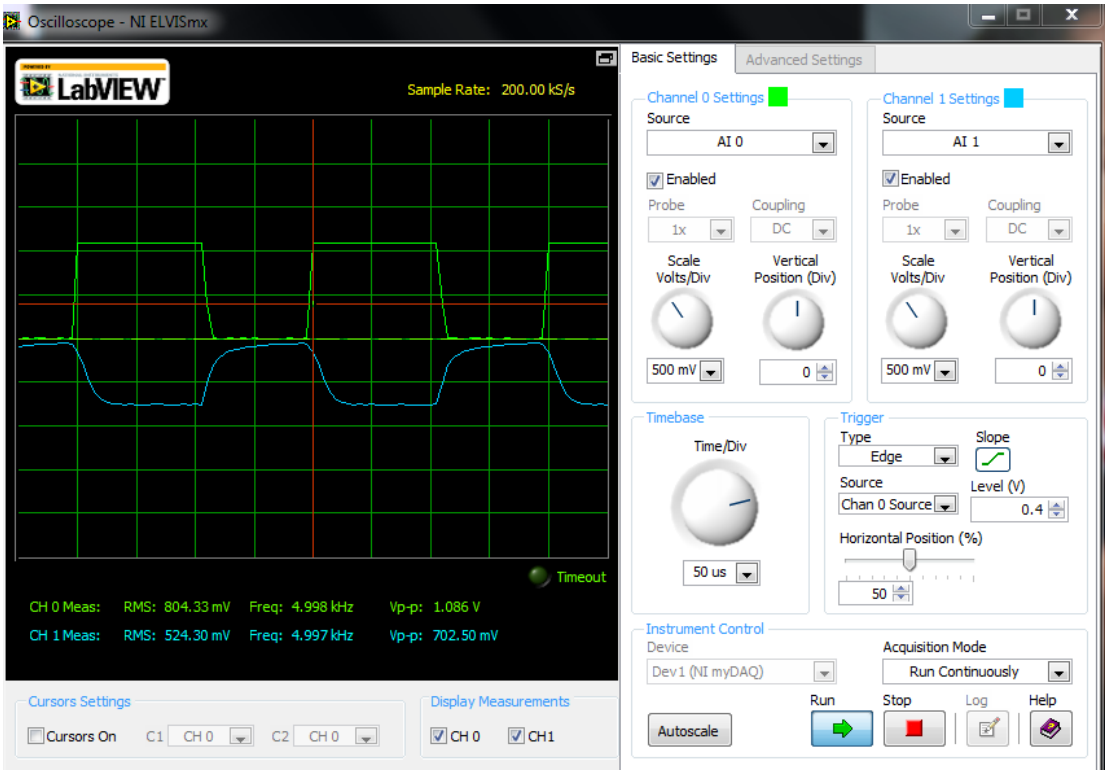


Figure 8: Transmitter and Receiver Signals

Submission Requirements:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- A header at the beginning with *Date, To, From, Subject*
- Written in first person from you
- Written with minimal spelling and grammar errors
- *Purpose, procedure, results* and *conclusions* of the laboratory experiment
  - The very first sentence is the purpose and must explain why you are doing the experiment
  - The *procedure* should be very short, a high level summary of what you did for each part. The *procedure* should be two paragraphs at the most.
  - The *results* should include the tables and figures and results of the data collected and your observations. The results cannot be pages of figures and tables only, it **must** include text that references the figures and explains what they are to the reader.
  - The *conclusions* must be the end and it should be a summary of what you did, what you learned, what you observed and recommendations
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text. **DO NOT SUBMIT SEPARATE FILES,** images must be pasted into the Word document at the appropriate place with respect to the text. **DO NOT** just insert a bunch of figures and tables at the end of the document.
- Also, you must include a statement at the end of your memo that states that this submission is your own work.

- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results. This discussion may be a part of the results and/or conclusions sections of the memo

ECE-205 Lab 7

Simulation of Feedback Control Systems

Overview:

In this lab you will be introduced to the idea of feedback control. We will simulate first and second order systems in a feedback loop with a controller. Next week we will build some of these systems so that you can see how we can relate the math and the block diagrams to a real system.

Equipment:

MATLAB/Simulink

Prelab:

Read the background theory and entire lab procedure thoroughly. Then complete the following analytical calculations. Submit the prelab in class the day before the lab session.

1. Show that the feedback control system represented by the diagram in Figure 1 has the given transfer function $G_o(s) = Y(s)/R(s)$.

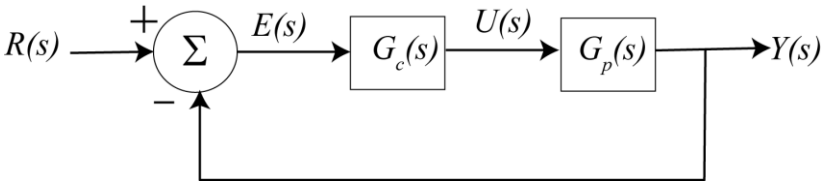


Figure 1: Basic closed loop system with unity feedback

$$G_o(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)}$$

2. For the feedback control system represented by the diagram in Figure 2, determine the transfer function, $G_o(s) = Y(s)/R(s)$. What do you think H(s) represents?

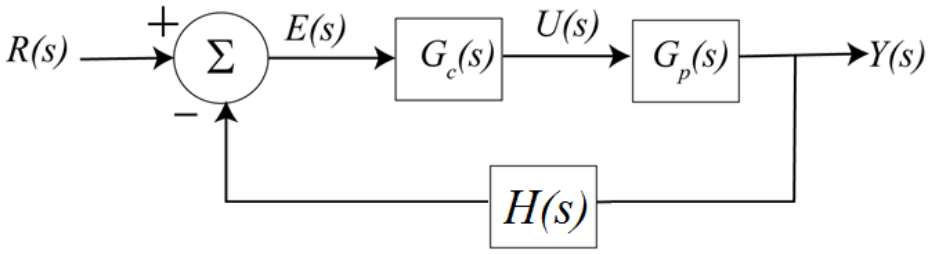


Figure 2: Feedback control system

Theory:

A **control system** is any device that you want to regulate or track at a certain **setpoint**. The human body is a control system that regulates our body temperature to 98.6 °F unless there is some type of disturbance such as trauma or illness. A car's cruise control is an example of a control system that tracks the speedometer set point to keep the car at a certain speed despite wind, rain or pot holes. When you drive a car, you act as a control system as you use your eyes to determine if you are driving straight and staying on the road and use the steering wheel to adjust the car. In your home, when you adjust the thermostat to a certain room temperature and the air conditioning or heater adjusts the blower or damper to meet that set point this is yet another control system.

When a system is set to a certain value (**setpoint**) and there is no measurement of the output to determine whether it actually achieved the set point, this is called **open loop control**. If you drive with your eyes closed that would be an example of open loop control. Open loop control is not a very good way to design a system unless the system is extremely reliable, predictable with no chance of failures. A **closed loop** control system or **feedback control** is when you use some type of sensor (such as your eyes, thermometer, encoder, speedometer) to determine how far the system is away from the set point (**error**) and use the **controller** to make the adjustments. The controller may be an electronic device, human, biological functions, etc. In these examples, the system being controlled is the **plant** (car speed, car position, body temperature, room temperature). A standard unity feedback control system with input, *R(s)*, output, *Y(s)*, controller, $G_c(s)$ and plant, $G_p(s)$, and error, *E(s)*, is shown in Figure 1.

Sometimes we have a system (plant) and the response needs to be modified to meet certain design characteristics. For example, the settling time for a step input may be too long, percent overshoot may be too large or the output never reaches the value of the set point. While it is sometimes possible to directly modify the plant to achieve the design characteristics, such as changing the values of circuit elements to change the time constants, there are many times that the original system cannot be changed. This is the main focus of control systems analysis and design.

Assume that the plant, $G_p(s)$, for the systems we want to control can be modeled as the following first and second order system transfer functions.

$$G_p(s) = \frac{K}{\tau s + 1}, G_p(s) = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

In order to change the behavior of the plant, we are going to do two things, both of which are depicted in Figure 1. The first of these is to provide some feedback so we can compare the input and output of the system. This difference is called the **error signal** and it is just the difference between the input and the output. In the diagram the error signal is depicted as *E(s)*, and it is the difference between the **reference signal** , *R(s)* and the output signal, *Y(s)*, $E(s) = R(s) - Y(s)$. In most control systems we want the output to equal the reference signal. The second thing we will do is to include a new system called a **controller**, that we depict by the transfer function, $G_c(s)$, that operates on the error signal and produces a new signal, the **control effort**, *U(s)*, which is the input into the plant. Note that the control effort is ***not*** the transfer function of the unit step.

In this lab we will look at three common controllers, a proportional (*P*) controller, and integral (*I*) controller, and a proportional plus integral (PI) controller. A **proportional** controller is used to increase the gain of the system and reduce settling time but it may also increase oscillations or make the system go unstable. An **integral** controller is also called a reset controller because it can reduce the error between the system output and the set point to zero but it may also increase settling time. A **proportional-integral** controller attempts to use the characteristics of both the P and I controllers to adjust the system output.

For a ***proportional*** (P) controller, the signal to the plant, *u(t)*, is directly proportional to the error signal, *e(t)* thus, $u(t) = k_p e(t)$. In the Laplace domain, the control signal is represented as $U(s) = k_p E(s)$, and the transfer function for the proportional controller is then

$$G_c(s) = \frac{U(s)}{E(s)} = k_p .$$

For an ***integral*** (I) controller, the signal to the plant is proportional to the integral of the error signal thus, $u(t) = k_i \int_0^t e(\lambda) d\lambda$. In the Laplace domain this equation is $U(s) = \frac{k_i}{s} E(s)$, and the transfer function for the integral controller becomes $G_c(s) = \frac{U(s)}{E(s)} = \frac{k_i}{s}$

For a ***proportional plus integral*** (PI) controller, the input to the plant is proportional to both the error and the integral of the error (with two scaling factors) thus

$u(t) = k_p e(t) + k_i \int_0^t e(\lambda) d\lambda$. In the Laplace domain this is

$$U(s) = k_p E(s) + \frac{k_i}{s} E(s) = \frac{k_p s + k_i}{s} E(s) = \frac{k_p (s + k_i / k_p)}{s} E(s)$$

We often just write the transfer function for the PI controller as $G_c(s) = \frac{U(s)}{E(s)} = \frac{k(s+z)}{s}$

In this lab you will use all three controller types to adjust system characteristics to meet the design specifications.

Procedure:

PART I: Simulating Feedback Systems in MATLAB

In this part we will show you how to simulate closed loop systems using both MATLAB and Simulink.

1. Copy the first MATLAB script files (m-file) and the Simulink model file (mdl-file) that you created for Lab 6 into a Lab 7 folder.

2. Rename the script file to something such as *Lab7(YourInitials).m*. Rename the model file to *closedLoop.mdl*.
3. Use `%{` and `%}` to create a header comment with the filename, your name, date created and a brief description of the file.
4. Use the `%` symbol to comment out the `sim('openloop')%` command, since we will not be using Simulink for a while.
5. Confirm that the `clear variables;` command is at the beginning of the script file.
6. Confirm that the variables are still set to `tau = 0.001;` and `K = 2.0;`
7. Enter the first order transfer function, $G_p = \frac{K}{\tau s + 1}$ into the script file. It may already be there from last week but if not and you don't remember how to create a transfer function, please refer to Lab 6.
8. Set the final time variable `Tf = 0.01` (be sure to use at least one capital letter so it is not confused with the transfer function command **tf**).
9. Modify the **linspace** command to create a time vector t from **0** to **Tf** with 1000 sample points.
10. The system will have a step input with amplitude **0.1** and the following commands will generate the input. Now add the input, r, for the closed loop system. ***Note that the input for the open loop system is x and it should also be in the script!***

```
A = 0.1;
r=A*ones(1,length(t));
```

11. To create the proportional controller, define the gain, `Kp = 0.2;` and then enter the transfer function into MATLAB:

```
Gc = tf(Kp,1);
```

12. Next, construct the closed loop transfer function by entering the following command into the script. Note that the minreal command removes pole/zero cancellations.

```
Go = minreal(Gc*Gp/(1+Gc*Gp))
```

13. To simulate the system, we then use the **lsim** command as follows:

```
yo = lsim(Go,r,t);
```

14. Next plot the open loop system you created last week and the closed loop system with the proportional controller by typing the following command.

```
subplot(121)
plot(t, y,'Linewidth',3);grid;
title('Open Loop Response');
xlabel('t, sec');
ylabel('y(t)');
axis([0 Tf 0 1.1*max(y)]);
subplot(122)
plot(t, yo,'Linewidth',3);grid;
title(['Closed Loop Response, Kp = ', num2str(Kp)]);
xlabel('t, sec');
ylabel('y_{o}(t)');
axis([0 Tf 0 1.1*max(yo)]);
```

15. If you now run the script, the output should look similar to Figure 2.

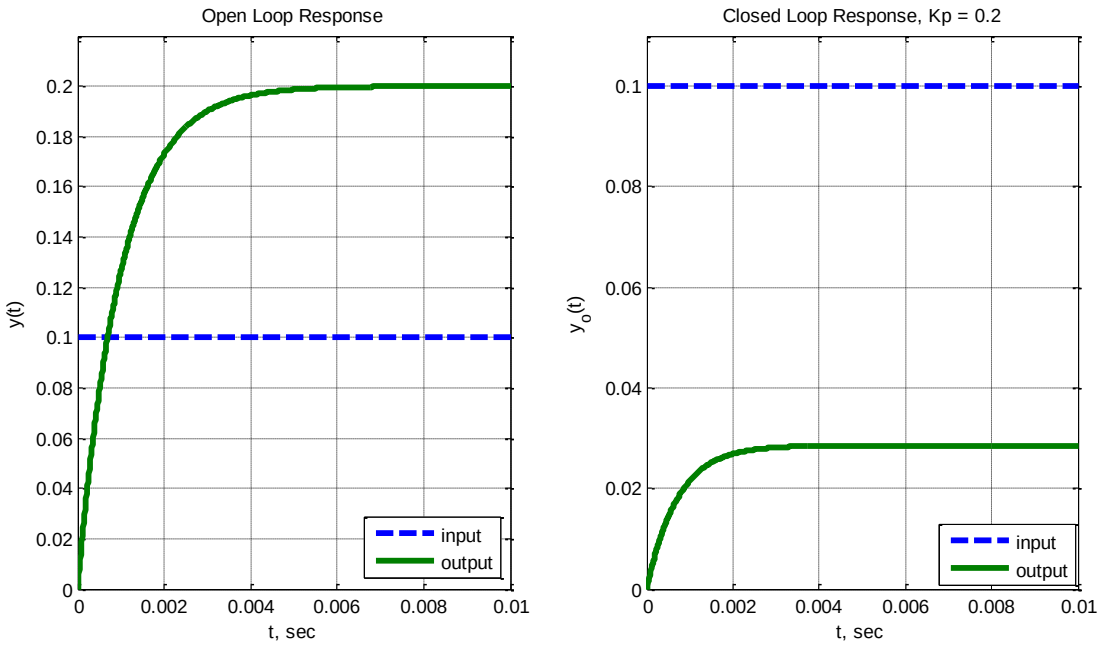


Figure 2: Open Loop and Closed Loop Response of a First Order System

16. If you compare these two figures, you should note two things:

- The settling time is much faster for the system with feedback
- The steady state values are different for the two systems, but neither one of them actually matches the reference input signal (the reference signal in this case is 0.1)

17. Do not save the figures for inclusion in your lab memo yet, go on to PART II.

PART II: Proportional Control

1. Modify your controller so that $k_p = 1$ and generate the output again with the open loop and closed loop system. Examine the output and record the observations in your lab memo. What is the settling time? What is the output amplitude? Calculate the steady **state error = input - output**.

2. Then change the proportional controller gain to $k_p = 5$. Examine the output and record the observations in your lab memo. What is the settling time? What is the output amplitude? Calculate the steady state error = input - output. How do variations in the proportional controller gain affect the system output?

PART III: Integral Control

1. Now change final time to `Tf = 0.02;` seconds and change the proportional controller to the integral controller, $G_c(s) = \frac{200}{s}$ = K_i/s. What do you observe about the settling time and steady-state error? Record your observations in the lab memo submission.

2. Next change the integral controller to $G_c(s) = \frac{500}{s}$. What do you observe about the settling time and steady-state error? Record your observations in the lab memo submission.

3. You should note that the settling time is now larger, the reference input (A = 0.1) is matched perfectly. However, as we increase the integral gain, *Ki*, the percent overshoot increases. The correct response for the second controller is shown in Figure 3.

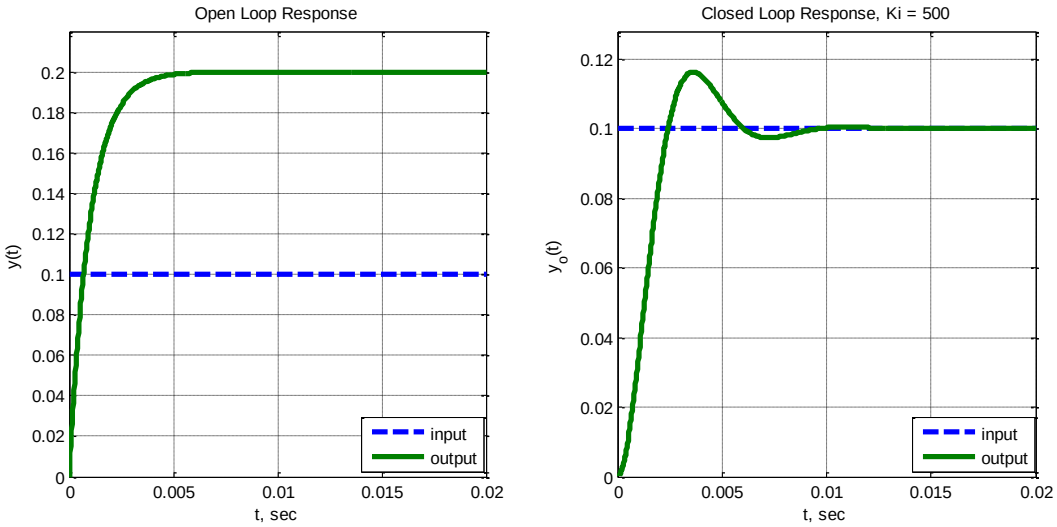


Figure 3: Integral Control

PART IV: Proportional-Integral Control

1. Next change final time to `Tf = 5e-5;` and simulate the system for the following proportional plus integral controller, $G_c(s) = \frac{100(s+10)}{s}$ = $(K_p s + K_i)/s$. What do you observe about the settling time and steady-state error? Record your observations in the lab memo submission.

2. Next change the PI controller to $G_c(s) = \frac{500(s+10)}{s}$. What do you observe about the settling time and steady-state error? Record your observations in the lab memo submission.

3. You should observe that the settling time is shorter and we also match the reference input. The correct response for the first controller is shown in Figure 4.

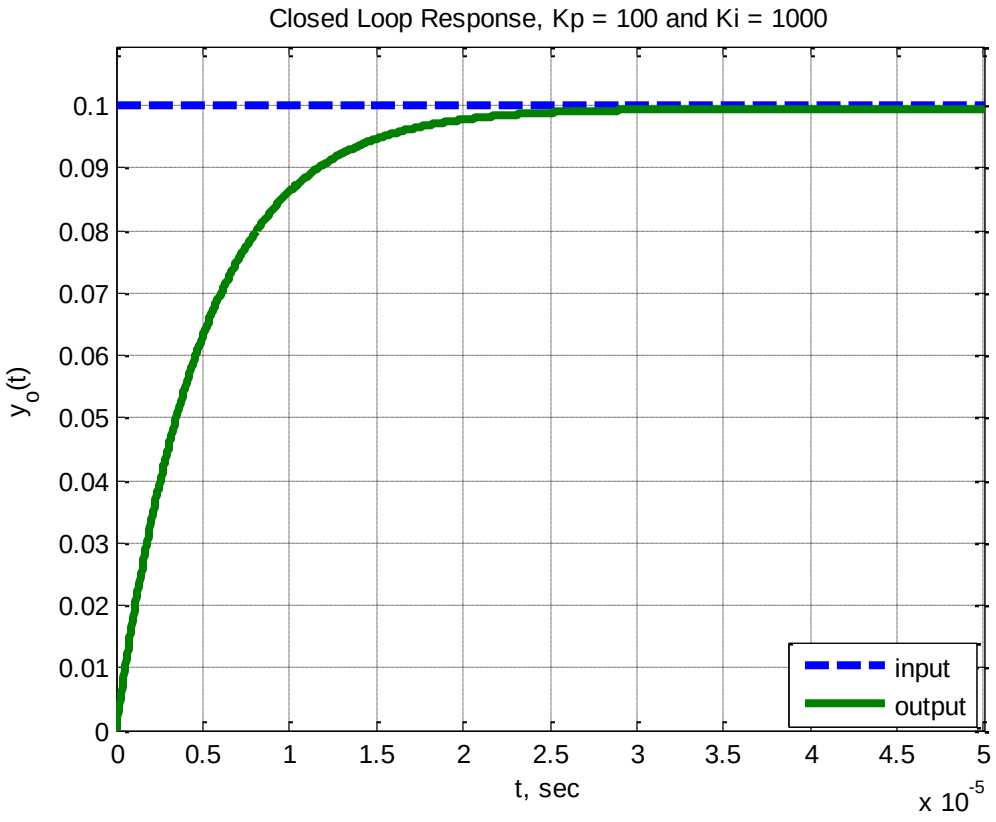


Figure 4: Proportional-Integral (PI) Control

PART V: Simulating Feedback Systems with Simulink

1. Open the Simulink model that you name **closedLoop.mdl.**
2. Modify your model so it looks like that in Figure 5.

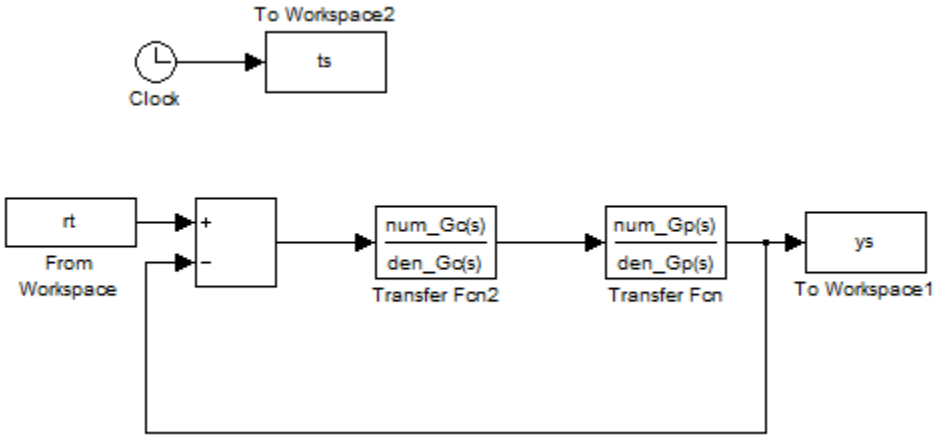


Figure 6: Closed loop Simulink model file.

3. Since we are dealing with small time scales, we will need to modify the maximum default time step in Simulink. To do this click Simulation→Configuration Parameters. Under the **Solver** change the **Max step size** from auto to to **1e-6**. The next two figures show the basic idea. Save the model file when you are done (see Figure 7).

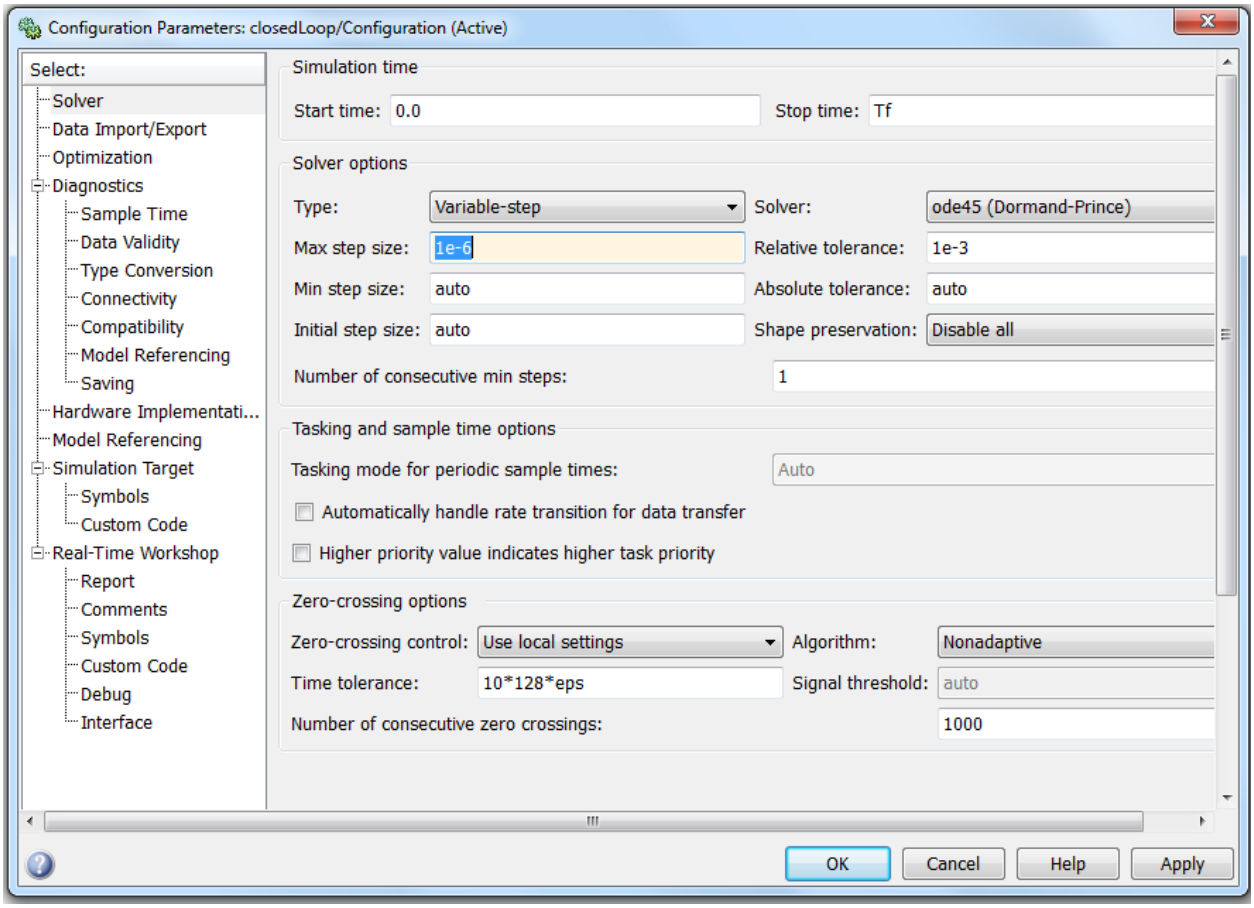


Figure 7: Simulation Configuration Parameters

4. We will use the same time and input values from the MATLAB simulation so type the following command in the script file:

```
rt = [ t' r'];
```

5. Use the following command to extract the numerator and denominator of the controller:

```
[num_Gc,den_Gc] = tfdata(Gc,'v');
```

6. Run the simulation from the m-file by using the following command:

```
sim('closedloop');
```

7. At this point you have results from both Matlab (t, yo) and Simulink (ts, ys). Modify your m-file to plot both of these results on the same graph. The plot should have two different line types and colors, grid, the axes labeled and a descriptive title. See Figure 8 for an example of the system output.

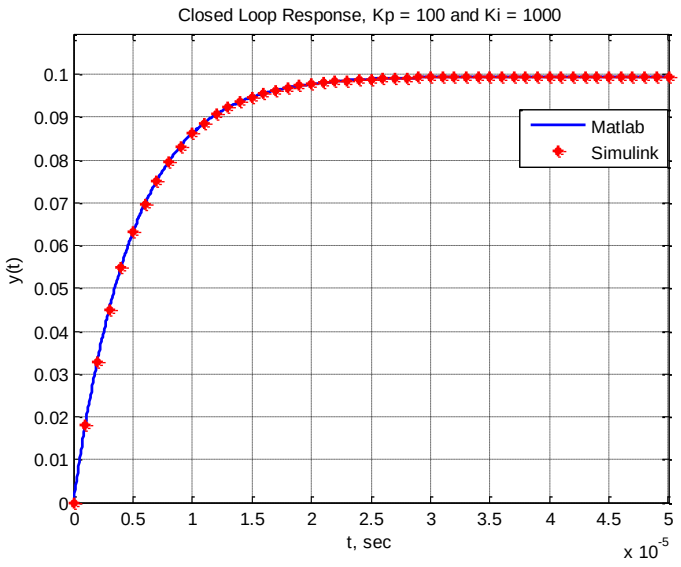


Figure 8: MATLAB and Simulink Results for closed loop control

8. Now, generate the MATLAB and Simulink results for each of the controllers in PARTS II, III and IV. Make sure each plot has a descriptive title with the controller gains similar to Figure 8, axes labeled, grid, and distinct markers for MATLAB and Simulink results. *You must include all **six graphs** in your lab memo submission.*

PART VI: Second-order system closed loop control

1. Modify the script (m-file) so that you plot the closed loop response for the second order system. Similar to Lab 6, the system parameters are a natural frequency of **2000** rad/sec, a damping ratio of **0.2**, and a static gain of **2.0**. Set the final time to **0.02** seconds.

2. Simulate your system in both Matlab and Simulink using the integral controller with a gain of **250**. Your result should look like similar to Figure 9. Note that the steady-state output is the same as the reference input.

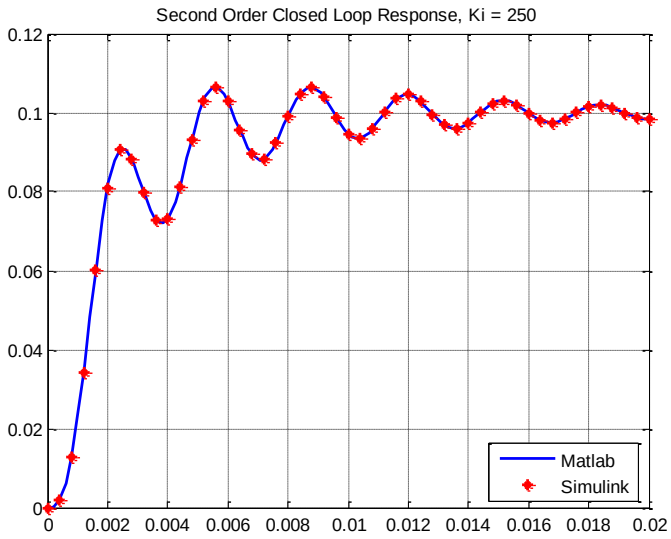


Figure 9: Second order closed loop response

3. *You need to include your graph in your memo. It should have a descriptive title, axes labeled, grid on, line markers and a legend and be referenced in the text.*

PART VII: Prefilters for Modifying Steady State Values

Sometimes when we use feedback control, we need to be able to adjust the steady state output value. This is very apparent in our first order system when we used a proportional controller. In that case, we were able to significantly change the settling time of the system, but the output did not equal the input. What we need is a proper scaling, and this is usually called a **prefilter.** Figure 10 shows our basic feedback loop with a prefilter. The prefilter can either be a transfer function or a constant. In this lab we will assume the prefilter is a constant, so $G_{pf}(s) = G_{pf}$ .The purpose of the prefilter is to take the reference signal and scale it so that the input to the system matches the output of the system.

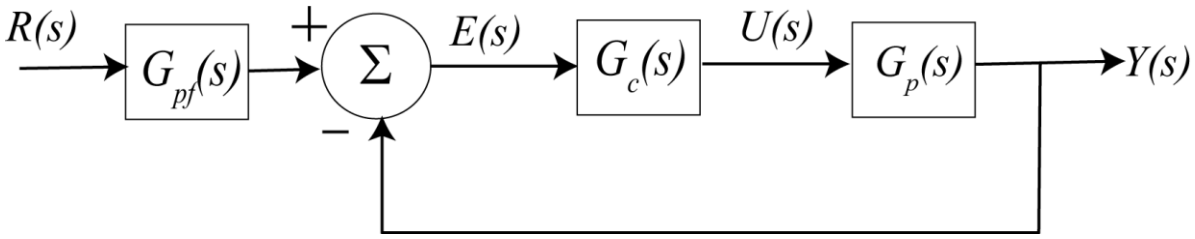


Figure 10: Basic feedback control loop with a prefilter

If the closed loop system has the transfer function, $G_o(s) = \frac{N_0(s)}{D_0(s)}$ then we want the prefilter to have the following value, $G_{pf} = \frac{D_0(0)}{N_0(0)}$. In class, you will learn why you need to use this value.

1. Modify the **closedLoop.mdl** Simulink model you created to match Figure 10 by adding a transfer function block in series with the input rt. The numerator is `num_Gpf` and the denominator is `den_Gpf`.
2. In order to determine the value of the pre-filter for zero steady state error, you need to evaluate a polynomial at zero. There are two ways to evaluate a polynomial at zero, but the easiest way to do this is to use the **`end`** parameter, as follows:

```
[num_Go,den_Go] = tfdata(Go,'v');
num_Gpf = den_Go(end);
den_Gpf = num_Go(end);
```

3. To implement the dynamic prefilter in both your Simulink model and Matlab models, use the following code to redefine Go:

```
Go = Go*num_Gpf/den_Gpf;
```

   Insert these lines of code right after you use `tfdata` to extract the numerator and denominator of the closed loop transfer function and before you create `yo`.
4. Rerun **Parts I and II**, for the first order system with *Kp = 0.2, 1,* and *5* with a final time of 0.005 s. Put all three of the plots on a 1 x 3 grid by using a `subplot` command. Figure 11 shows the beginning of the plots.
5. What the difference between these results and those in parts I and II?

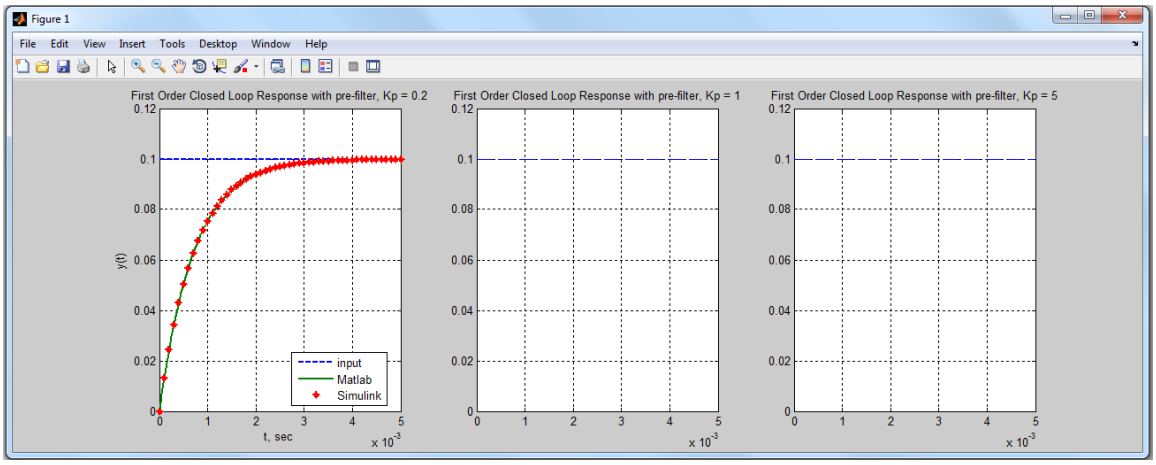


Figure 11: Closed loop control of a first order system using a prefilter

Submit this plot, the MATLAB m-file and the Simulink model file with your lab memo to the Angel Drop Box.

Submission Requirements:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- A header at the beginning with *Date, To, From, Subject*
- Written in first person from you
- Written with minimal spelling and grammar errors
- *Purpose, procedure, results* and *conclusions* of the laboratory experiment
  - The very first sentence is the purpose and must explain why you are doing the experiment
  - The *procedure* should be very short, a high level summary of what you did for each part. The *procedure* should be two paragraphs at the most.

- The *results* should include the tables and figures and results of the data collected and your observations. The results cannot be pages of figures and tables only, it **must** include text that references the figures and explains what they are to the reader.
- The *conclusions* must be the end and it should be a summary of what you did, what you learned, what you observed and recommendations

- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text. **DO NOT SUBMIT SEPARATE FILES,** images must be pasted into the Word document at the appropriate place with respect to the text. **DO NOT** just insert a bunch of figures and tables at the end of the document.
- Also, you must include a statement at the end of your memo that states that this submission is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results. This discussion may be a part of the results and/or conclusions sections of the memo

ECE-205 Lab 8

Implementation of Feedback Control Systems

Overview:

In this lab you will build a simple feedback control system using both a proportional (P) and proportional plus integral (PI) controller. The systems you will control are simple first and second order systems. Similar to last week, you will also use a prefilter on the to adjust the gain on the input so that there is zero steady-state error and the output follows the input.

Equipment:

3 - 10 kΩ variable resistors (potentiometers)

4 - TL072 dual op amps

11- 1 kΩ resistors

1 μF capacitor

0.01 μF capacitor

33 mH inductor

Prelab:

Read the background theory and entire lab procedure thoroughly. Then complete the following analytical calculations and MultiSim simulations. Submit the prelab in class the day before the lab session.

1. Derive the governing differential equation for the circuit in Figure 3. What is the transfer function H(s) = U(s)/E(s)?
2. Derive the governing differential equation for the circuit in Figure 4a. What is the transfer function H(s) = Y(s)/U(s)?
3. Derive the governing differential equation for the circuit in Figure 4b. What is the transfer function H(s) = Y(s)/U(s)?

Multisim simulations

4. In this part you will use Multisim to build a second-order system.

- Place the following parts on the Multisim page: DC_POWER (x3), GROUND (x2), RESISTOR_RATED, CAPACITOR_RATED, INDUCTOR_RATED, OPAMP_5T_VIRTUAL, TD_SW1
- Make the voltage sources **12V**, **-12 V** and **1V.** Change the capacitor to **0.01 uF** and change the inductor to **33 mH.** Change the switch to **TON = 0.005 sec** and **TOFF = .010 sec.**
- Rearrange the components to be the second order system shown in Figure A.

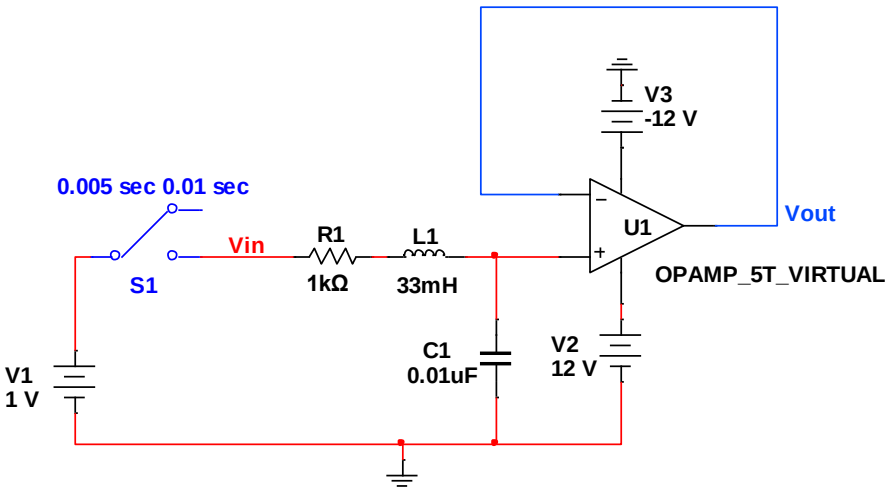


Figure A: Second Order System Plant

- Double click the wire on the output of the op amp and name it **Vout.** Double click the wire on the right side of the switch and name it **Vin.**
- To run the simulation, click Simulate->Analyses->Transient Analysis. Change the Initial conditions drop down to **Set to zero**. Set the End time (TSTOP) to **0.01** s. Check the **Maximum time step settings (TMAX)** box. Select the radio button to set the maximum time step to **1e-005** s.

- Select the output tab in the Transient Analysis settings and click on **V(vin)** and **V(vout)** and click **Add**. Click **Simulate**. Your results should look similar to Figure B. Note that there is a significant percent overshoot for this system and this could be an undesirable characteristic of the system.

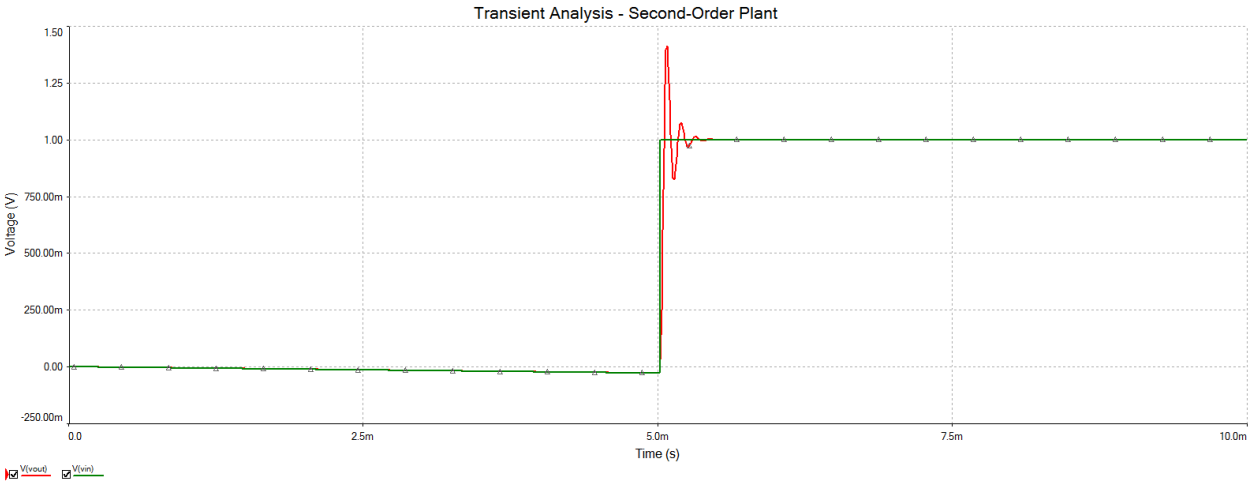


Figure B: Second order plant step response

- Now build the feedback control system to correct for the transient characteristics. You will need to add three more op amps, one for the summer and two for the controller and several more resistors and a capacitor (see Figure C). You may need to increase the Multisim sheet by clicking Options->Sheet Properties. Select the Workspace tab and change the sheet size to 22" x 17" in order to fit the entire schematic.

- Run the simulation again and note that the large percent overshoot has been corrected. In lab you will use potentiometers to adjust the gain on the controller and prefilter in order to get a desired system output.

- This completes your prelab, *you need to submit the Multisim (.ms11) file and the two graphs created (before and after feedback) as part of your prelab submission.*

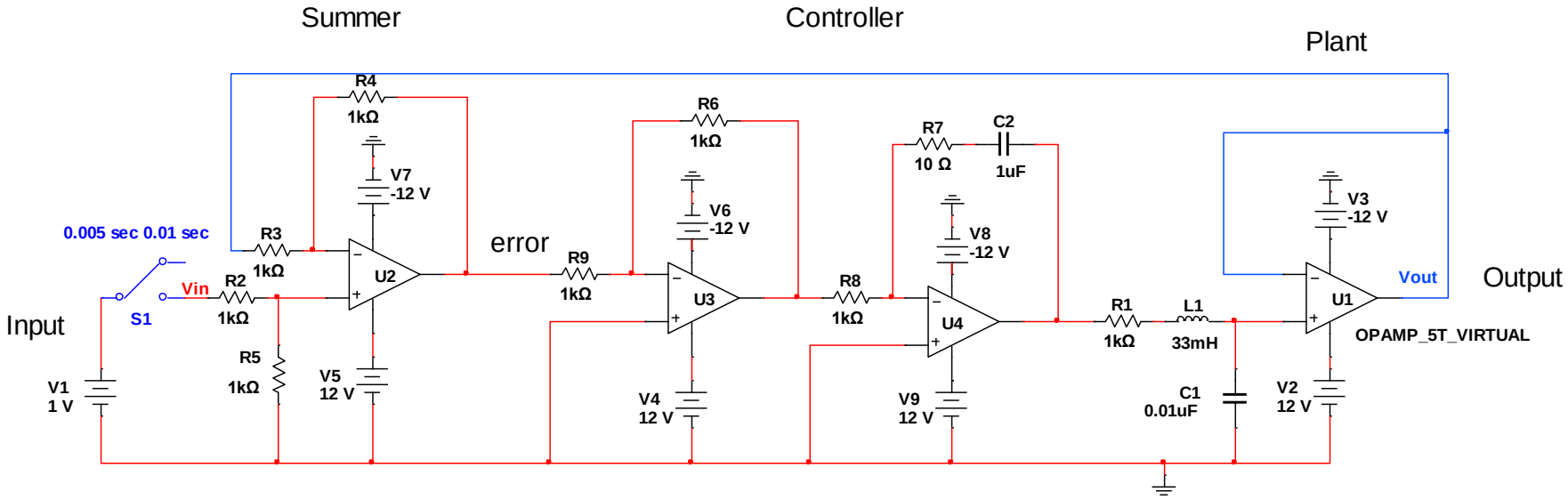


Figure C: Second order system with feedback control

Theory:

In Lab 7, we studied the simulation of open loop and closed loop control systems using MATLAB and Simulink. In this lab, you will actually build a closed loop (feedback) control system to adjust the characteristics of a first order and second order system. Recall that the system characteristics are *settling time, percent overshoot, steady-state error,* and *time to peak*.

There are four key components of the closed loop control system including the prefilter, summer, controller and plant (see Figure 1).

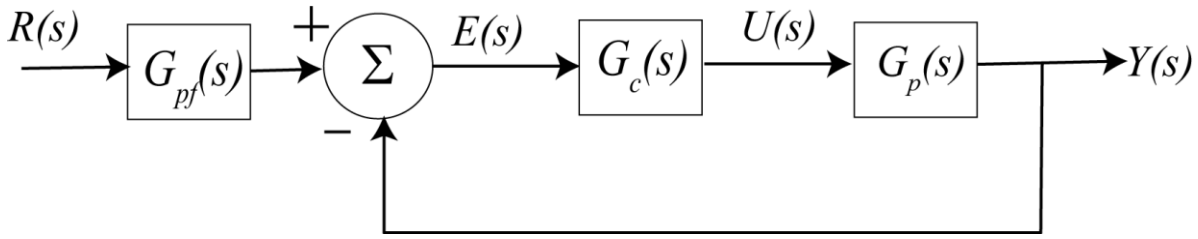


Figure 1: Basic closed loop system with unity feedback

All of these components can be built with an operational amplifier or cascaded op amps. For example, by cascading inverting amplifiers it is possible to produce a constant, *Gpf* (see Figure 2).

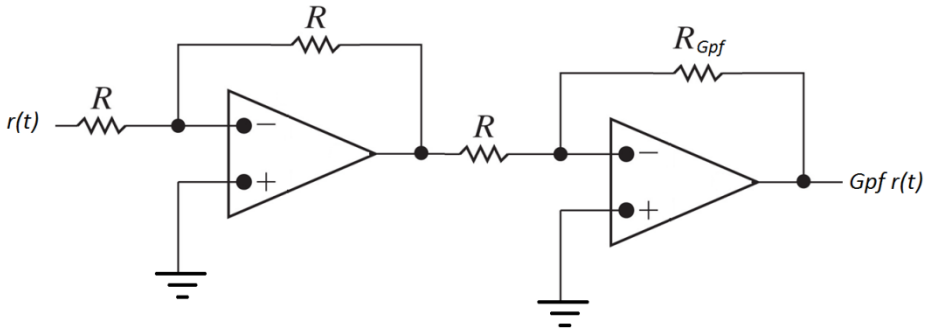


Figure 2: Prefilter (G_{pf})

In addition, cascaded inverting amplifiers can also produce a proportional (P) controller, k_p. By adding a capacitor to the feedback path of an inverting amplifier, it is possible to create a proportional-integral (PI) controller (see Figure 3).

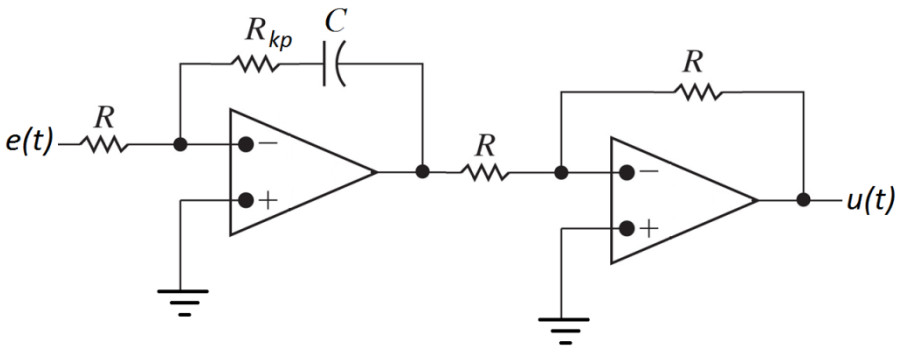
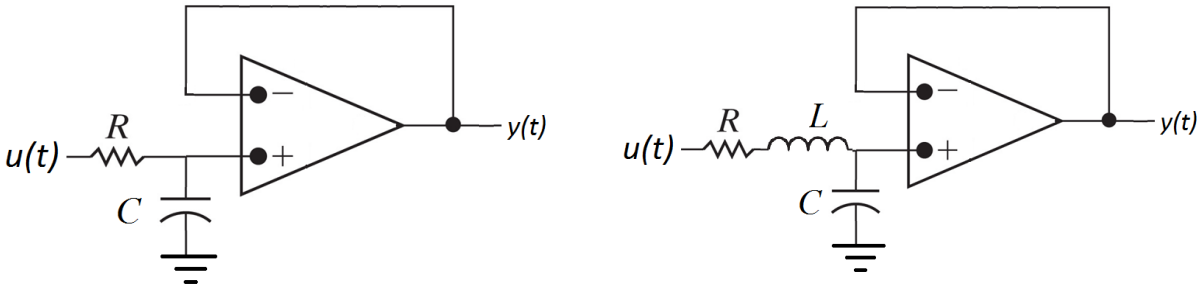


Figure 3: PI Controller (kp + ki/s)

Lastly, the first and second order plants can be represented by operational amplifiers as well (see Figure 4).



a. First order ($K/(\tau s + 1)$) b. Second order ($K\omega_n^2/(s^2+2\zeta\omega_n s+\omega_n^2)$)

Figure 4: System Plants

In the procedure, you will actually use several op amps to build these system components. Since this is a complicated circuit, you will build the circuit in parts, test them and then connect them together to form the overall feedback control system. **Make sure you build and test the circuit in stages!**

Procedure:

PART I: Proportional Control for a First Order System

The proportional controller has five parts, shown in the different shaded regions in Figure 5. The equivalent block diagram is shown below the circuit, so that you can match elements of the block diagram with the circuit subsystems. All of the resistors are either 1 kΩ or a variable 10

kΩ resistor (potentiometer) and a 1 μf capacitor. You will also use the TL072 integrated circuit which has 2 operational amplifiers on it. Please review Lab 6 or perform an internet search if you don't recall the pin out for the TL072. Later when you build the second order system, you will also use a 0.01 μF capacitor and a 33 mH inductor. **We will build and debug this circuit in stages, so do not start building yet!**

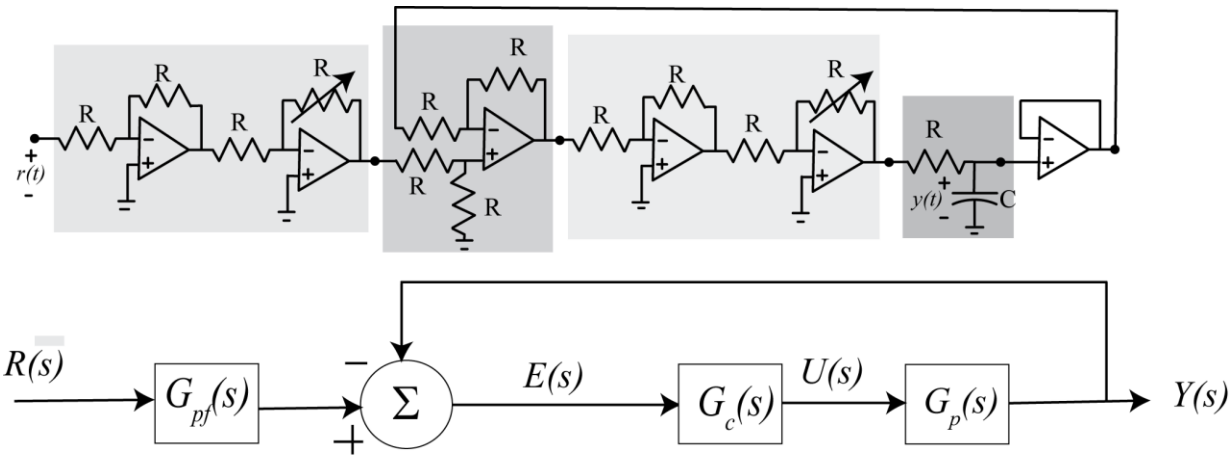


Figure 5: Proportional controller for the first order circuit.

The first shaded part is the prefilter, which changes the gain of the system so the steady state value of the output matches the steady state value of the input. The next stage is the differential amplifier (summer), or the feedback element for finding the difference between the input and output. The third stage is the proportional gain controller. The fourth stage is the plant or first-order system we want to control. The final stage is a simple isolation amplifier to avoid loading effects. Note that the output is the voltage across the capacitor.

PART II: Build the prefilter

1. In this part you will build the prefilter shown in Figure 6. First, set up the power busses on your breadboard by turning the board horizontal such that the red buss is on top. Use a wire to jump the two blue busses together and this will be the **ground** buss. The top red buss will be for the **+Vcc** or 15V on the operational amplifier. The bottom red buss will be for the **-Vcc** or -15V on the operational amplifier. **DO NOT CONNECT THE POWER FROM THE NI MYDAQ UNTIL THE END.**

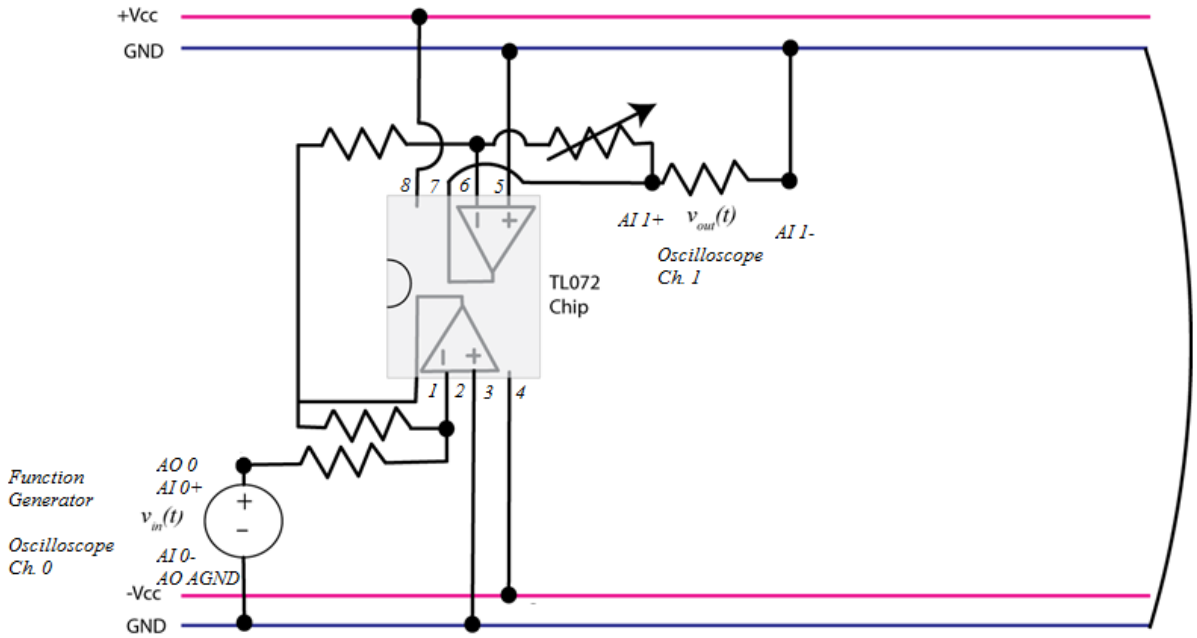


Figure 6: Prefilter (gain circuit)

2. Insert the op amp on the left side of the board with the dot pointing to the left and make sure it straddles the ditch in the middle.

3. Connect pin 3 and pin 5 to **ground** which are the positive terminals on both of the op amps. Connect pin 4 to -**Vcc** and pin 8 to +**Vcc** for the op amp power. Connect a 1 kΩ resistor between pins 1 and 2 and a 1 kΩ resistor from pin 2 to the function generator. The connection for the function generator is **AO 0** and **AGND**. Also, put Channel 0 of the oscilloscope in parallel with the function generator input (**AI 0+**, **AI 0-)**.

4. Connect a 1 kΩ resistor between pins 1 and 6. Connect a 10 kΩ variable resistor between pins 6 and 7. Finally, put a 1 kΩ resistor from pin 7 to ground to measure the output. Put Channel 1 of the oscilloscope across the resistor from pin 7 to ground in order to measure the output and confirm that the prefilter is working correctly (**AI 1+, AI 1-**).

5. Connect the +15V (+Vcc), -15V)-Vcc) and AGND power supply terminals on the MyDAQ to the red and blue breadboard busses.

6. Set the function generator to a **50 Hz square wave** with a **2.00 V peak to peak** amplitude and a **1.00 V offset**.

7. Start the oscilloscope and enable both channels 0 and 1. Set Channel 0 and Channel to **500 mV/div** and **2 ms/div** time scale. Set the trigger to **Edge**, **Chan 0 Source** and increase the level to **0.1**.

8. Use the screwdriver to adjust the variable resistor until the input and output are the same or the gain is **one**. Your final oscilloscope output should look similar to Figure 7.

9. Once the circuit is working, **disconnect** the oscilloscope, function generator, power supplies and **remove** the load resistor at pin 7.

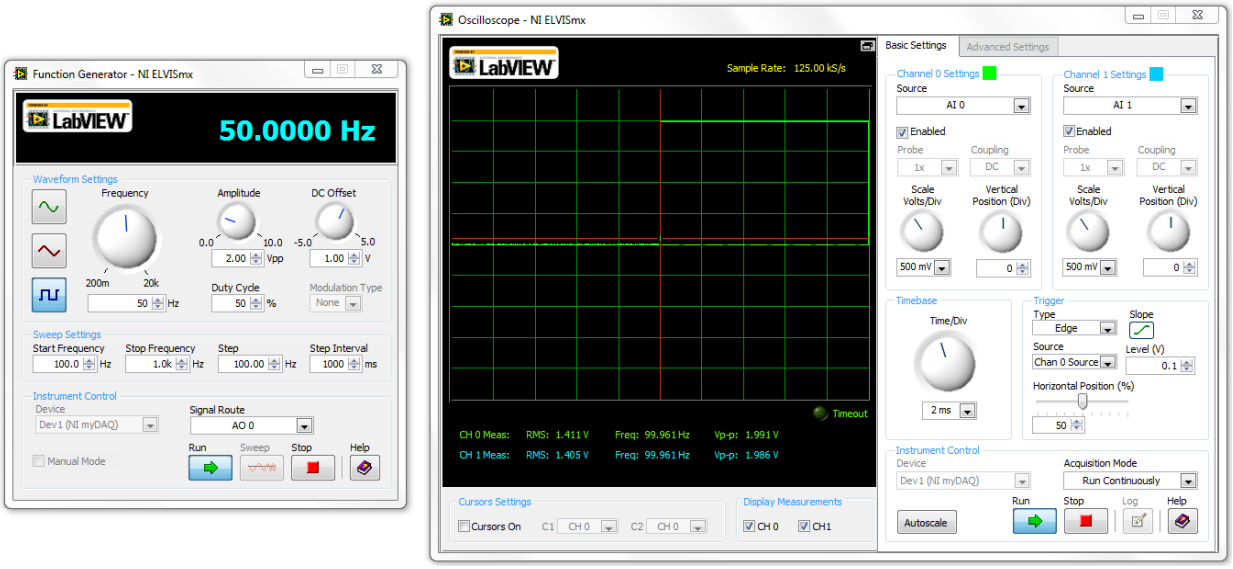


Figure 7: Prefilter oscilloscope output screen

PART III: Build the differential amplifier

1. In this part you will build the differential amplifier in Figure 8. Place another op amp on the breadboard as close as possible to the prefilter circuit. Make sure that the dot is to the left and the op amp straddles the ditch.
2. Connect pin 4 to **-Vcc** and pin 8 to **+Vcc.**

3. Connect a 1 kΩ resistor from pin 6 to pin 7, connect a 1 kΩ resistor from pin 5 to ground and another 1 kΩ resistor from pin 5 to the function generator (**AO 0, AO AGND**). Channel 0 of the oscilloscope will also go to this same resistor (**AI 0+, AI 0-**).
4. Connect a 1 kΩ resistor from pin 6 to ground. *This resistor will eventually be the feedback for the system and be removed from the ground buss.*
5. Connect a 1 kΩ resistor from pin 7 to ground. *This resistor will be the load resistor to measure the output for debugging purposes.*
6. Connect the +15V (+Vcc), -15V)-Vcc) and AGND power supply terminals on the MyDAQ to the red and blue breadboard busses.
7. Set the function generator to a **50 Hz square wave** with a **2.00 V peak to peak** amplitude and a **1.00 V offset**. Start the oscilloscope and enable both channels 0 and 1. Set Channel 0 and Channel to **500 mV/div** and **2 ms/div** time scale. Set the trigger to **Edge**, **Chan 0 Source** and increase the level to **0.1**. If the input and output are not the same then check your wiring.

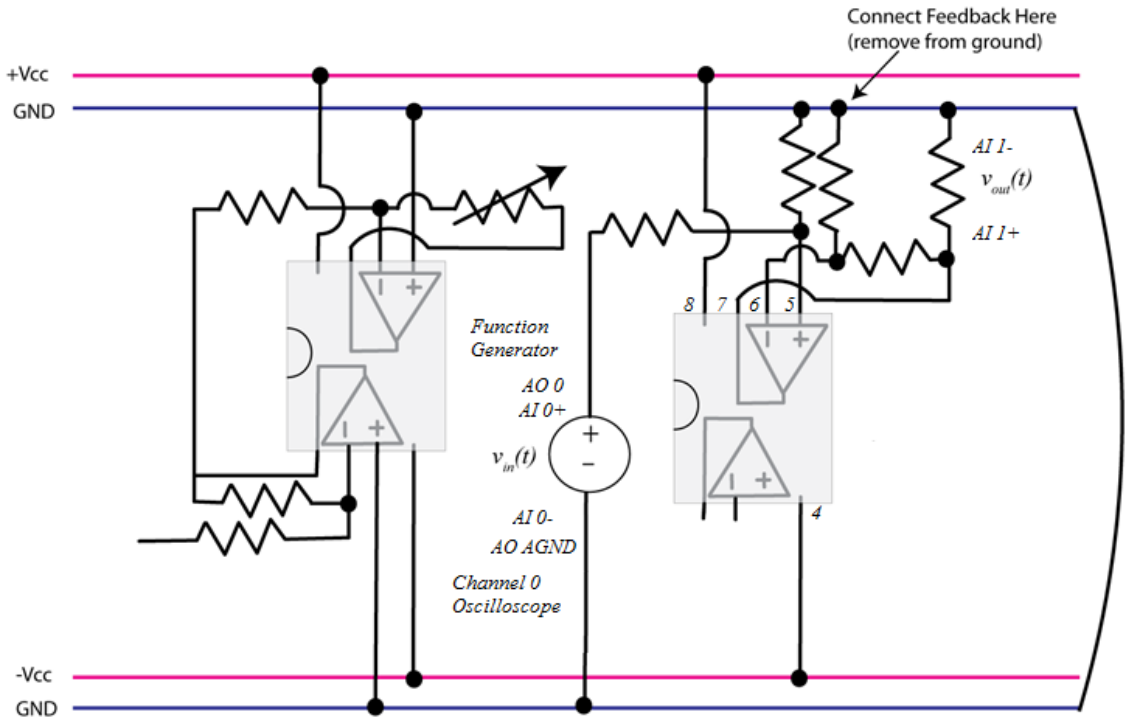


Figure 8: Differential Amplifier Schematic

PART IV: Connect the prefilter and differential amplifier

1. Next, connect the prefilter and differential amplifier as shown in Figure 9.
2. Move the function generator and Channel 0 of the oscilloscope back to the input of the prefilter (the resistor at pin 2 on the first op amp).
3. Connect the resistor from pin 5 of the differential amplifier which was connected to the function generator to pin 7 of the first op amp.
4. Keep the same settings for the function generator and oscilloscope and measure the input and output. The output and input of the circuit should match. If they are a little off then use the variable resistor to make them match. If you cannot get the input and output to match then do not go on until you check your circuit and correct it.
5. Finally, **remove** Channel 1 of the oscilloscope and the load resistor on the differential amplifier from pin 7 to ground.

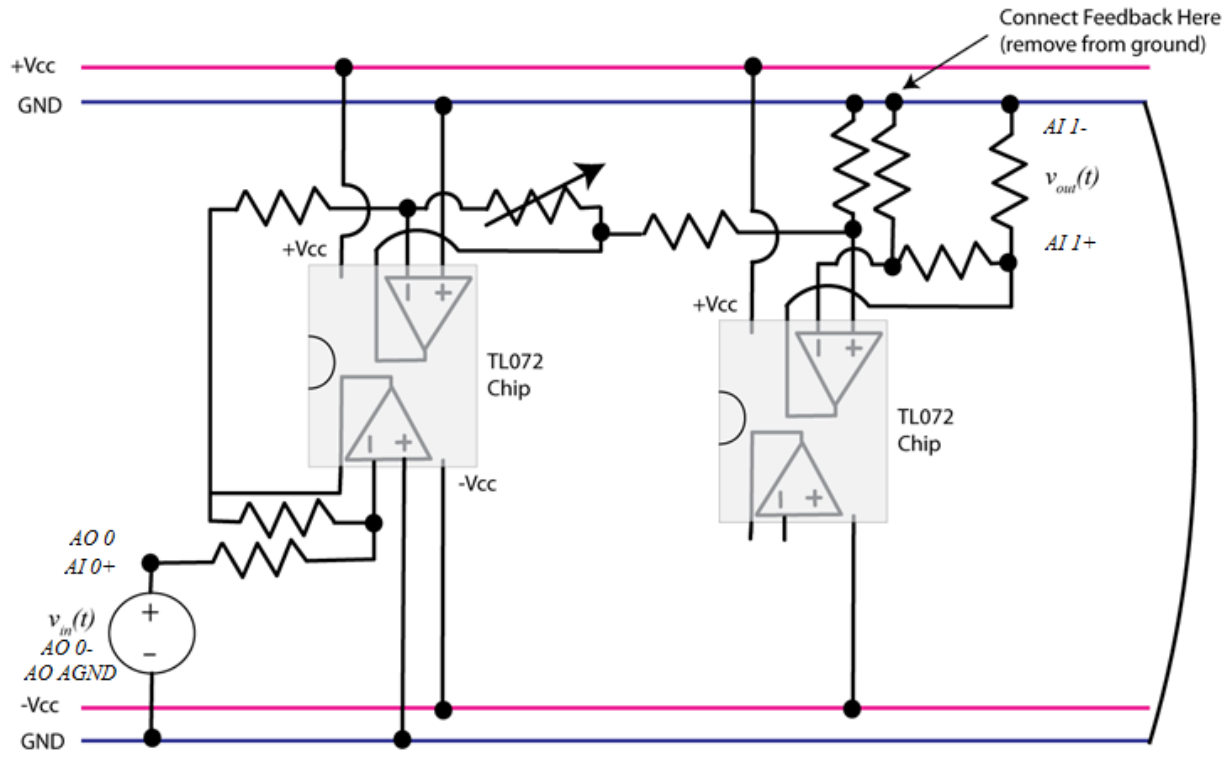


Figure 9: Prefilter and Differential Amplifier Schematic

PART V: Add the proportional controller

1. In this part, you will add the proportional control to the system so that it now looks similar to Figure 10. Place another op amp on the board, with the dot on the left.
2. The proportional controller consists of two cascaded inverting amplifiers. Both of these will be created on the new op amp.
3. On the new op op amp, connect pins 3 and 5 to **ground**, connect pin 4 to -**Vcc** and connect pin 8 to **+Vcc**.
4. Connect another 1 kΩ resistor between pin 7 on the differential amplifier and pin 2 on the proportional controller to cascade them together.
5. Connect a 1 kΩ resistor between pins 1 and 2 on the proportional controller. Try to use wires from your kit to keep the layout as flat as possible to help with debugging later.
6. Connect a 1 kΩ resistor from pin 1 to pin 6. Connect the 10 kΩ variable resistor from pin 7 to pin 6.
7. Lastly, for debugging purposes, place a resistor from pin 7 to ground.
8. Use the oscilloscope to measure the input and the output of the cascaded 3 stage system. Adjust the variable resistor until the output and input are the same. If you cannot get them to match, then check your circuit and do not move on until it does.
9. **Remove** the power supplies, oscilloscope, function generator and load resistor at pin 7.

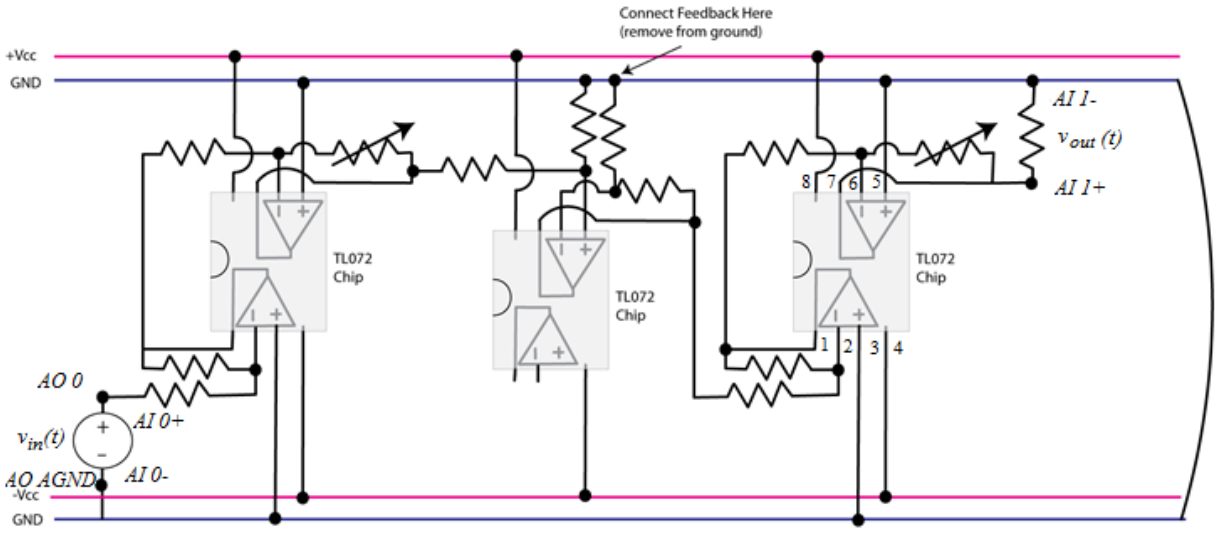


Figure 10: Proportional Control Schematic

PART VI: Add the system plant

1. In this part, you will build the system plant which is a first order RC circuit in series with an isolation amplifier. The purpose of the isolation amplifier is to avoid loading effects on the circuit.
2. Add a 4th op amp to the breadboard and build the circuit in Figure 11. Connect pin 4 to **-Vcc** and pin 8 to **+Vcc**.

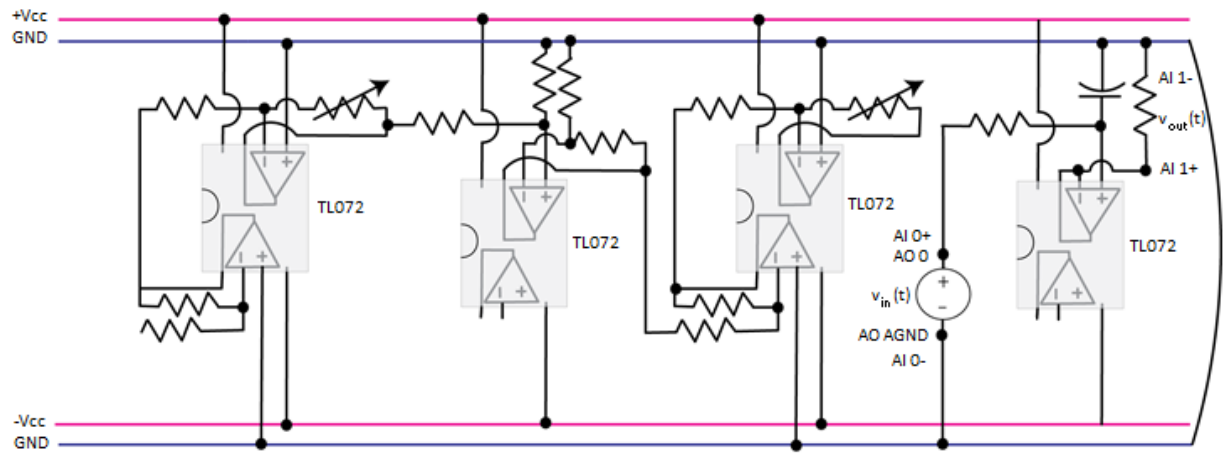


Figure 11: System Plant Schematic

3. Connect a 1 μF (105) capacitor from pin 5 to ground. Connect a 1 kΩ from pin 5 to the function generator and Channel 0 of the oscilloscope.
4. Connect a small wire between pins 6 and 7.
5. Connect a 1 kΩ from pin 7 to ground to measure the output of the system for debugging purposes. Connect Channel 1 of the oscilloscope across the load resistor at pin 7.
6. Measure the output of the plant and it should look like Figure 12. This represents the open loop response of the system plant. If you do not get a response similar to this figure, do not move on until you fix the circuit.
7. **Remove the load resistor on pin 7, function generator, oscilloscope and power supplies**

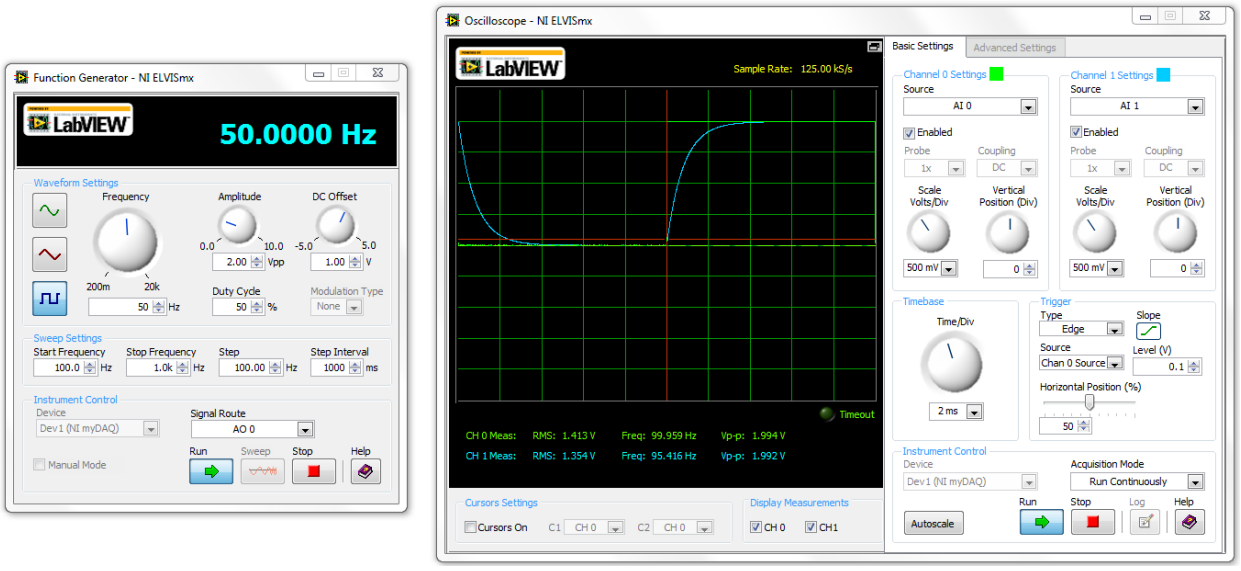


Figure 12: Open loop response of the first order plant

PART VII: Add the feedback

1. In this part, you will add the feedback to the system by modifying the circuit to look like Figure 13. Disconnect the 1 kΩ resistor at pin 6 on the differential amplifier from ground. Add a long wire from that resistor to pin 6 on the isolation amplifier (plant).

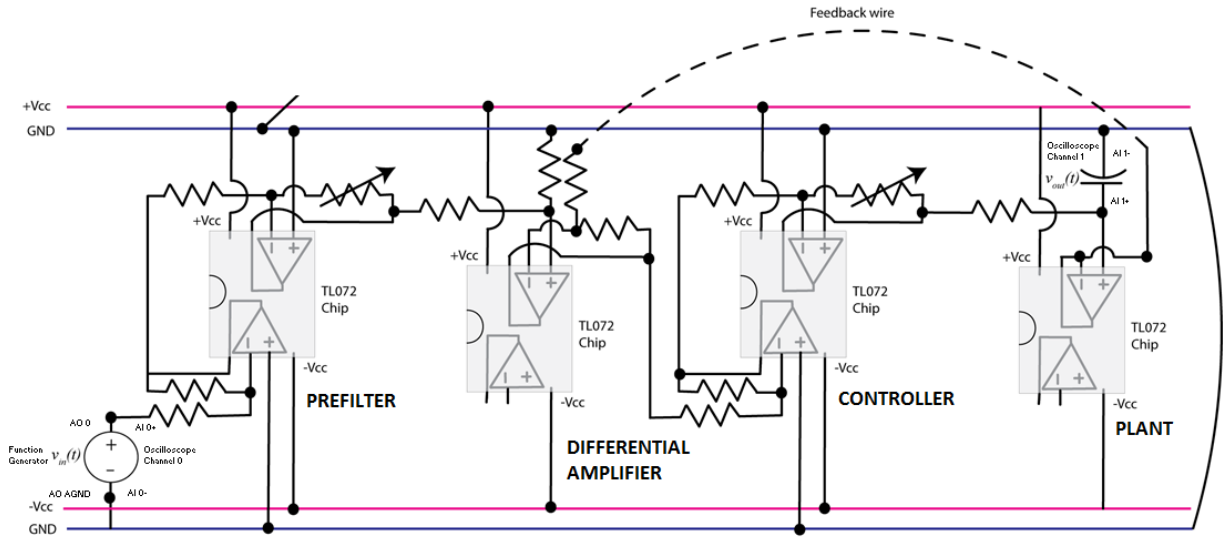


Figure 13: Closed loop system schematic

2. Move the resistor at pin 5 on the isolation amplifier (plant) to pin 7 on the controller.

3. Attach the function generator and Channel 0 of the oscilloscope to the input of the full system at the resistor on pin 2 of the prefilter.
4. Attach Channel 1 of the oscilloscope to the output of the system across the capacitor on pin 5 of the isolation amplifier (plant).
5. Measure the input and output of the system and you should get a response similar to Figure 14. Note that amplitude is about half of the amplitude for the open loop system and the settling time is about half as long.
6. Slowly change the variable resistor on the proportional controller to speed up the response. Then change the variable resistor on the prefilter so that the output is the same amplitude as the input.
7. *You need to submit **two** different screen captures of the system response (oscilloscope) in your memo. They both should have a faster response (smaller settling time) and the same amplitude as the input (zero steady state error) to indicate that the controller was working.*

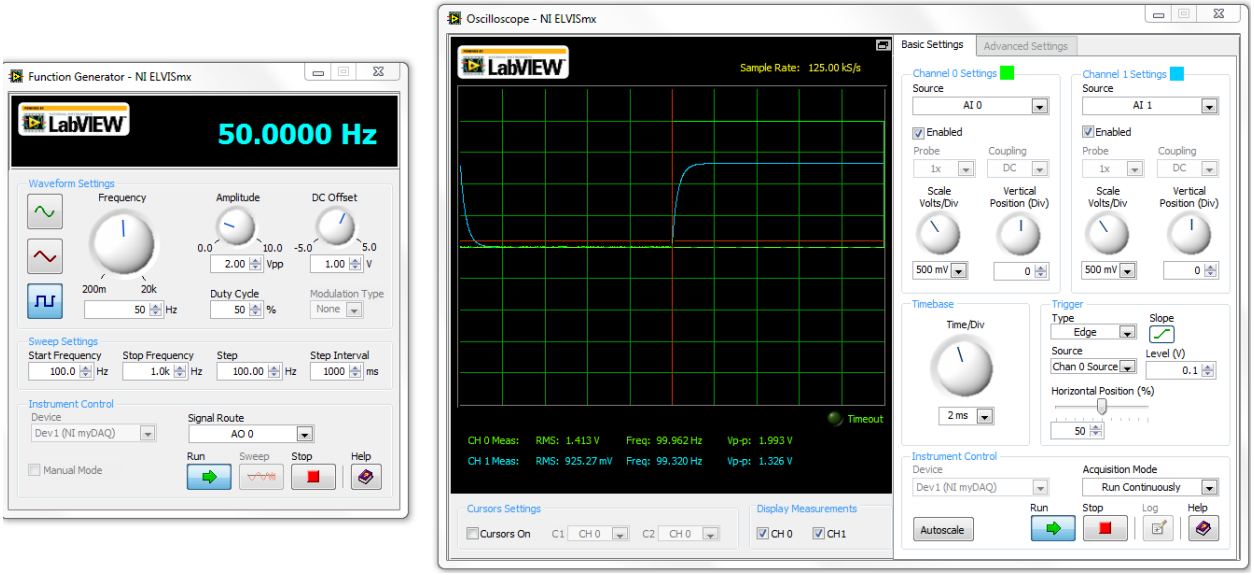


Figure 14: Closed loop response of a first order system

PART VIII: PI Control for a first order system

1. In this part, you will change the proportional controller to a proportional plus integral controller to determine how it affects the first order plant. This requires only slight modification in the circuit to make it look like Figure 15.

2. Remove the 1 kΩ feedback resistor between pins 1 and 2 on the proportional controller with a 1 μF capacitor in series with a variable resistor (see Figure 15).

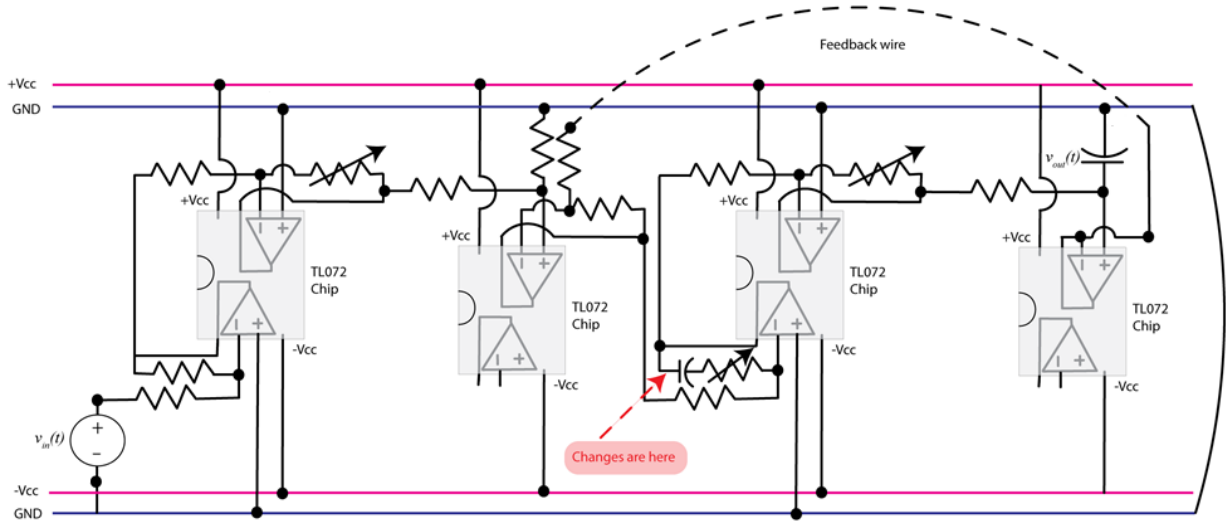


Figure 15: PI Controller Schematic

3. Note that both of the variable resistors affect the proportional gain (k_p) and the second variable resistor affects the integral gain (k_i).
4. Set the both variable resistors on the controller to the middle of the range and adjust the prefilter gain so that steady state output matches the input (zero steady-state error).
5. Now go back and adjust the variable resistors on the controller. We can use these to change the gain of the proportional term and the gain of the integral term. You should note that as you change these resistors, the output still matches the input (as long as the system has time to reach steady state). This is the benefit of having an integral controller!
6. Adjust your controller so you get an output similar to Figure 16 which has a **50%** overshoot and settling time of approximately **5.00 ms**. *You should include a screen shot of this oscilloscope output in your lab memo.*
7. Next, adjust the controller so that the output has virtually **no overshoot** and a settling time of approximately **1.00 ms** similar to Figure 17. *You should include a screen shot of this oscilloscope output in your lab memo.*

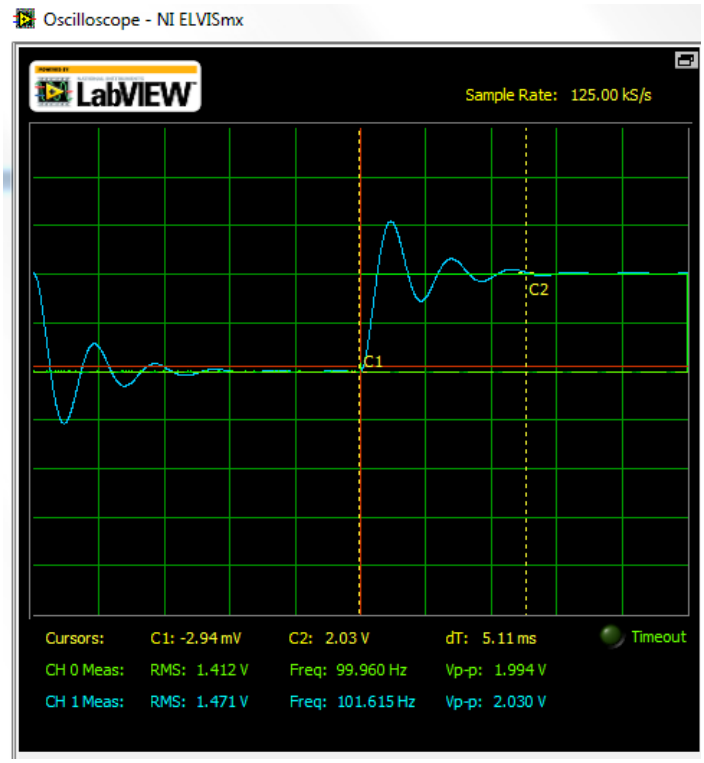


Figure 16: PI Control of a first order system with 50% overshoot

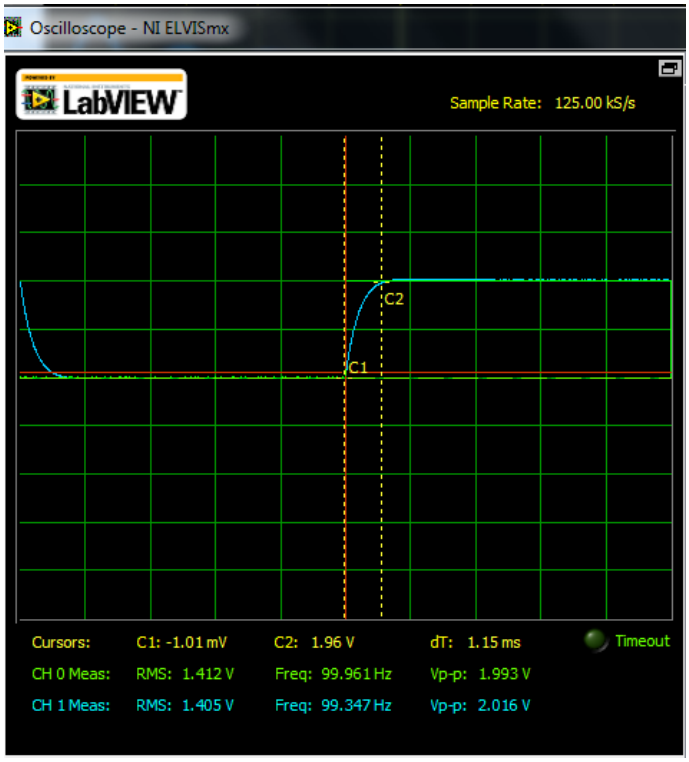


Figure 17: PI Control of a first order system with 0% overshoot

PART IX: Second order system plant

1. In this part, the plant will be changed to a second order system and we will use a PI controller to examine the changes in the system characteristics. This is not as straightforward as the first order system because it is typically more difficult to control a system as the order increases.
2. Remove the first order plant from the circuit. This includes the 1 μF capacitor and the 1 kΩ resistor.
3. Modify the circuit so that the plant and isolation amplifier now look like Figure 18. The resistor is 1 kΩ , the inductor is 33 mH and the capacitor is 0.01 μF (103).

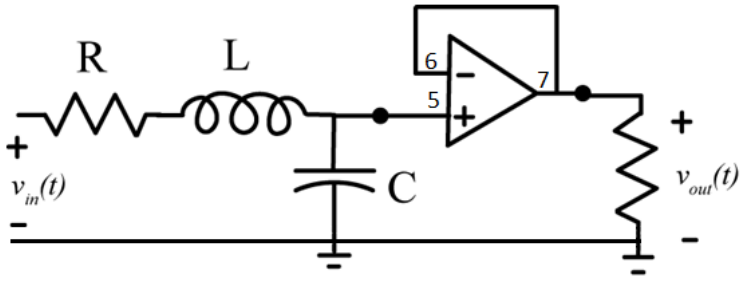


Figure 18: Second order system plant

4. To make sure the plant (system) is built correctly, disconnect the feedback wire to the differential amplifier from the plant. Also, add a 1 kΩ load resistor at pin 7 for debugging purposes.
5. Connect the function generator and Channel 0 of the oscilloscope to the input of the plant at the 1 kΩ resistor. Connect Channel 1 of the oscilloscope to the output of the plant across the load resistor.
6. Set the input to a **1 kHz square wave** with **2 V pp** amplitude and a **1 V peak offset.** Set both oscilloscope channels to **1V/div** and and **100 μs/div.**
7. The output is the voltage across the capacitor and the graph should look similar to Figure 19. Note that this system has a settling time of approximately **200 μs** and about a **30% overshoot**. *You should include a screen shot of this oscilloscope screen in your memo.*
8. **Remove** the function generator, oscilloscope and 1kΩ load resistor at pin 7.

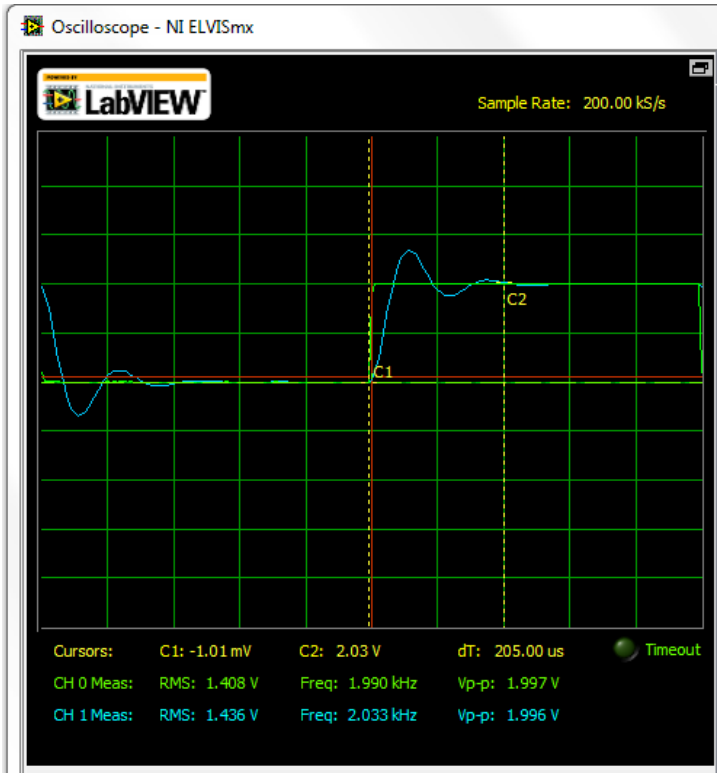


Figure 19: Open loop step response of a second order plant

PART X: PI Control for a second order system

1. In this part you will connect your system, or plant, to the feedback network. Connect the feedback wire back to pin 6 on the isolation amplifier (plant).
2. Connect the input resistor on the plant to pin 7 on the PI controller.
3. Move the function generator and Channel 0 of the oscilloscope back to the input of the system at the prefilter. Use Channel 1 of the oscilloscope to measure the output of the system at the voltage across the .01 μF capacitor.
4. Adjust the system so there is **no overshoot** and the settling time is less than approximately **500 μs** as shown in Figure 20. You may have to adjust the time scale and frequency of the signal generator to get a good picture. Be sure your system has reached steady state (it's flat on the top!). *You should include a screen shot of the system output in your memo.*
5. Note that in this case, we cannot speed up the response of the system if our goal is to reduce the overshoot. You should remember that we are not changing our plants, or basic

systems, but we are changing how they behave using feedback and an active controller to change the system characteristics.

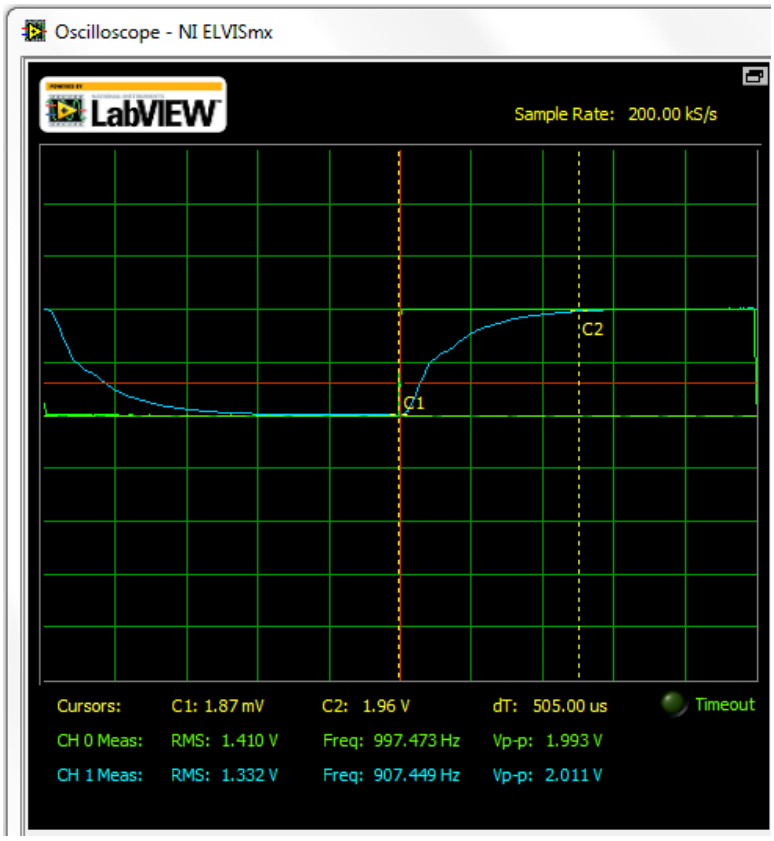


Figure 20: Closed loop step response of a second order plant

Submission Requirements:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- A header at the beginning with *Date, To, From, Subject*
- Written in first person from you
- Written with minimal spelling and grammar errors

- *Purpose, procedure, results* and *conclusions* of the laboratory experiment
    - The very first sentence is the purpose and must explain why you are doing the experiment
    - The *procedure* should be very short, a high level summary of what you did for each part. The *procedure* should be two paragraphs at the most.
    - The *results* should include the tables and figures and results of the data collected and your observations. The results cannot be pages of figures and tables only, it **must** include text that references the figures and explains what they are to the reader.
    - The *conclusions* must be the end and it should be a summary of what you did, what you learned, what you observed and recommendations
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text. **DO NOT SUBMIT SEPARATE FILES,** images must be pasted into the Word document at the appropriate place with respect to the text. **DO NOT** just insert a bunch of figures and tables at the end of the document.
- Also, you must include a statement at the end of your memo that states that this submission is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results. This discussion may be a part of the results and/or conclusions sections of the memo

ECE-205 Lab 9

Bode Plots

Overview:

In this lab you will use MATLAB to contrast the Bode plot of several systems given the transfer function. You will also measure the frequency response of two circuits and then use the NI myDAQ and Multisim to display the measured and simulated data. For many unknown (or complex) systems this is a very common method for determining the transfer function. Lastly, you will use the Bode plot to determine the steady-state response of a system given the input.

Equipment:

1 kΩ resistor

1 μF capacitor

0.01 μF capacitor

33 mH inductor

10 kΩ variable resistor (potentiometer)

Prelab:

In this section, you will use MATLAB to generate the Bode plot of several transfer functions. You will use the **bode** command to generate the Bode plot given a transfer function.

In order to find the Bode plot of the transfer function, $G(s) = \dfrac{0.1\left[\frac{1}{10}s+1\right]}{\left[\frac{1}{100}s+1\right]}$

Type the following code in MATLAB,

```
num =  0.1*[1/10 1];     %define the numerator of the transfer function
den =  [1/100 1];        %define the denominator of the transfer function
G = tf(num,den);         %define the transfer function
w = logspace(-2,5,1000);%define the frequency from w = 10^-2  to 10^5  rad/s
bode(G,w);grid;          %create the Bode plot
```

In order to find the Bode plot of a transfer function that has repeated poles or zeros, use the **conv** command which performs a discrete-time convolution.

In order to find the Bode plot of the following transfer function, $G(s) = \dfrac{\left[\frac{1}{100}s+1\right]^3}{\left[\frac{1}{10}s+1\right]^4}$

Type the following code in MATLAB,

```
nz = [1/100 1];                          %numerator factor
num = conv(nz,conv(nz,nz));              %raised to the third power
dp = [1/10 1];                           %denominator factor
den = conv(conv(dp,dp),conv(dp,dp));     %raised to the fourth power
G = tf(num,den);                         %create the transfer function
w = logspace(-2,5,1000);                 %define the frequency
bode(G,w);grid;                          %create the Bode plot
```

Alternately, you can also use the **`poly`** command which is based upon using the zeros and poles of the transfer function to create the numerator and denominator polynomial. Type the following code in MATLAB to demonstrate this technique,

```
num = poly([-100 -100 -100]);        %raised to the third power
den = poly([-10 -10 -10 -10]);       %raised to the fourth power
G = tf(num,den);                     %create the transfer function
w = logspace(-2,5,1000);             %define the frequency
bode(G,w);grid;                      %create the Bode plot
```

Please try these examples in MATLAB before moving on.

1. Submit the MATLAB Bode plots for the following transfer functions in class the day before lab.
   a. $H(s) = \frac{100}{s^2}$
   b. $H(s) = \frac{0.1\left(\frac{1}{100}s+1\right)^2}{s^2}$
   c. $H(s) = 10\left(\frac{1}{100}s+1\right)$
   d. $H(s) = \frac{0.1\left(\frac{1}{10}s+1\right)^2}{\left(\frac{1}{1000}s+1\right)^2}$
   e. $H(s) = \frac{0.1\left(\frac{1}{10}s+1\right)^2}{\left(\frac{1}{100}s+1\right)^2\left(\frac{1}{1000}s+1\right)^5}$
   f. $H(s) = \frac{1}{\left(\frac{1}{10}s+1\right)\left(\frac{1}{100}s+1\right)}$

2. In this problem, you will examine what happens to a Bode plot as the poles and zeros of a transfer function is varied.
   a. Download the **preLab9.m** file from the Angel course folder. This file plots the magnitude of the frequency response of the transfer function based on the location of the poles and zeros entered. This magnitude plot is not in dB and the amplitude has been scaled to have a maximum value of 1 for all of the plots.
   b. When the file runs, the MATLAB command window will prompt you for the values of the poles and zeros. Make sure to enter the poles and zeros in matrix form, i.e. [-10 +10j -10-10j] and [].
   c. For each of the following sets of poles and zeros create the plots and observe the results. You can enter all of the plots into Microsoft Word for you submission but do not put more than 3 figures per page to insure that they are large enough to read.
      i. Run the code with poles at **$-10 \pm 10j$, $-5 \pm 10j$, $-1 \pm 10j$** and no zeros (leave the variable zeros as an empty array, do not put in a zero). Note that as the magnitude of the imaginary part of the pole gets significantly larger than the magnitude of the real part, there are well defined peaks in the frequency response. *Make sure to submit all three plots.*
      ii. Leave the poles at **$-1 \pm 10j$** and add zeros first at **0**, then at **0** and **$\pm 5j$** , then at **0** and **$\pm 15j$**. Note that at the frequency where a zero of the transfer function occurs, the output of the transfer function is zero! Turn in all three plots.

Theory:

Bode diagrams are graphical techniques that describe the frequency response of a circuit. The Bode diagram helps you quickly visualize how the system responds to a wide range of frequencies. There are two separate plots: amplitude (magnitude) and phase angle and they vary with the frequency (see Figure 1). The plots are on semi log graph paper with the frequency on the horizontal log scale and the amplitude and phase angle on the linear vertical scale. Sometimes the amplitude plot is expressed in **decibels**.

$$|H(j\omega)|_{dB} = 20\log_{10}\frac{v_{out}}{v_{in}}$$

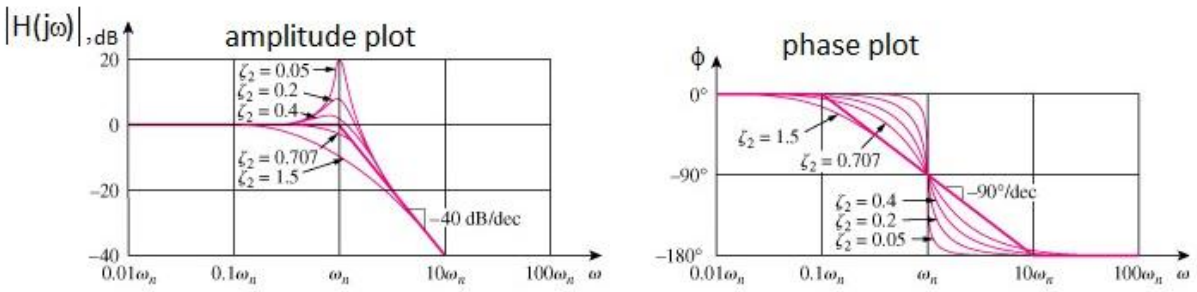


Figure 1: Bode Plots

In order to use a Bode plot find the steady-state response of a system given the input you will use the following process. For an input $x(t) = A\cos(\omega_o t)$ we know the steady state output will be

$$y_{ss}(t) = A|H(j\omega_o))|\cos(\omega_o t + \angle H(j\omega_o))$$

Hence if we measure the input and output amplitude at each input frequency ω_o, it should be straightforward to determine $|H(j\omega_o)|$ at each of these frequencies. To determine the phase, we will look at the time delay between the input and output signals at each input frequency, as follows,

$$\cos(\omega_o(t-t_d)) = \cos(\omega_o t - \omega_o t_d) = \cos(\omega_o t + \angle H(j\omega_o))$$

Note that in this relationship the phase of the transfer function is measured in radians,

$$\angle H(j\omega_o) = -\omega_o t_d$$

We can then convert the phase from radians to degrees using

$$\angle H(j\omega_o)\ (\text{in degrees}) = -2\pi f_o\ t_d \times \left[\frac{180^o}{\pi}\right] = -360^o\ f_o \times t_d$$

Procedure:

PART I: Frequency Response of a First Order System

1. The RC circuit in Figure 2 represents a first order system. Build the circuit on the breadboard with R = 1 kΩ and C = 1 μF. The input will be the function generator and the output will be the voltage across the capacitor.

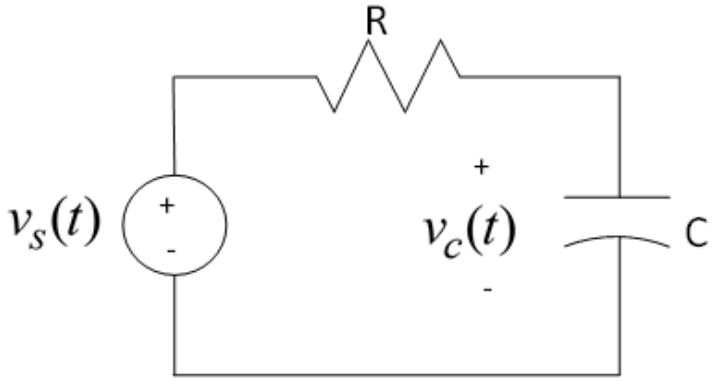


Figure 2: First order system (RC Circuit)

2. The transfer function for this circuit is

$$H(s) = \frac{K}{\tau s + 1}$$

where K is the static gain, τ is the time constant. For this model K = 1 and τ = RC = 1 ms.

3. This circuit is also a **low pass filter** where the **corner** or **cutoff frequency** is ω = 2πf = 1/τ= 1000 rad/s, f = 159 Hz. The cutoff frequency is the frequency where the power in the input signal has been reduced to one half of its maximum value, or equivalently, the magnitude of the transfer function has been decreased to 1/√2 of its maximum value. A **low pass filter** passes low frequencies below the cutoff frequency and attenuates signals above the cut off frequency. The range of frequencies passed by the filter is the **passband**. For this first order circuit, the maximum value of the transfer function is at ω = 0 rad/s so the magnitude at the corner frequency is $|H(j\omega_c)| = \frac{1}{\sqrt{2}}|H(0)|$. Note that this point is also called the **3 dB point**.

4. Connect the NI MyDAQ to the circuit by putting the function generator (**AO 0**, **AO AGND**) as the source. Connect Channel 0 (**AI 0+**, **AI 0-**) of the oscilloscope to the input to the circuit as well. Connect Channel 1 (**AI 1+**, **AI 1-)** of the oscilloscope to the output of the circuit across the capacitor.
5. Connect the NI My DAQ to your laptop and start the Function Generator and Oscilloscope instruments.
6. Set the function generator to a **20 Hz sine wave** with a **2.00 Vpp** amplitude and **0.00 V** DC offset.
7. Set the oscilloscope to **5 ms** per division, enable Channels 0 and 1 and set them both to **200 mV** per division and the Trigger type to **Edge**. As you vary the frequency, you should always adjust the time base so that one or two periods of the input and output wave forms are on the screen.
8. For an input $x(t) = A\cos(\omega_o)t$, the steady state output will be $y_{ss}(t) = A|H(j\omega_o)|\cos(\omega_o t + \angle H(j\omega_o))$ so if you measure the input and output amplitude at each frequency, ω_o, then you can calculate the amplitude, $|H(j\omega_o)|$.
9. To calculate the phase, $\angle H(j\omega_o)$, determine the time delay, t_d, between the input and output signals at each frequency, ω_o. Then use the following formula to calculate the phase angle in degrees,

$$\angle H(j\omega_o) = -\omega_o t_d \times (180/\pi)$$

10. You will use the following steps to create the Bode plot for the system,
    - Choose a frequency for the input signal (keep the input amplitude at 1 volt, or 2 volts peak to peak)
    - Using the cursors, measure the input amplitude (not peak to peak)
    - Using the cursors, measure the output amplitude (not peak to peak)
    - Using the cursors, measure the time delay between the input and output by putting Cursor 1 on Channel 0 and Cursor 2 on Channel 1. Put Cursor 1 on the first peak of Channel 0 and Cursor 2 on the subsequent peak of Channel 1.
11. Open up Excel and create a table with the following 4 column headings.
    - Frequency (in Hz)
    - Amplitude of the input signal (in mV)
    - Amplitude of the output signal (in mV)

- Time delay (in milliseconds)

12. For the initial Bode plot, collect and record the appropriate data in Excel for the input frequencies: *20, 60, 100, 140, 180, 220, 280, 320, 520, 720, 920, 1120 Hz*. Figure 3 provides an example of the system input and output with cursor measurements.

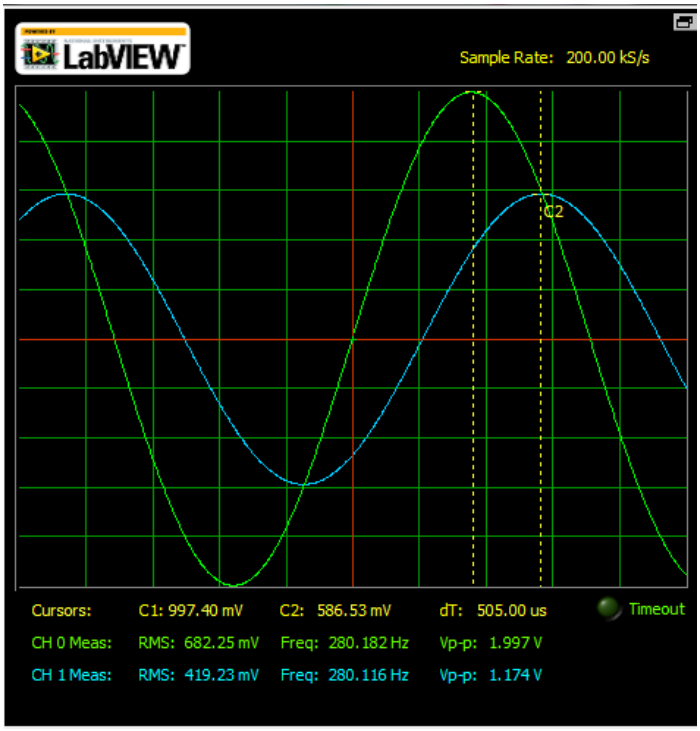


Figure 3: Sinusoidal Steady State Response of the first order system

13. Download the MATLAB script file, ***process_data.m*** from the Angel course Lab 9 folder. This program processes the data collected from the frequency response of the system. Copy and paste the data from Excel into the **measured** variable in this program. In the Matlab window, type

```
data = process_data;
```

14. Download the MATLAB script file, ***model_a.m*** from the Angel course Lab 9 folder. This program uses this data, as well as knowledge of the expected transfer function, to best fit the transfer function to the measured data and produce a couple of nice plots. There are three input arguments to this program, the array data, the estimated gain, and the

estimated time constant. How well this program works in part depends on how well you can guess these values.

15. Assume that the **gain is 1** and the **time constant is 0.001** and run the program by typing the following in the MATLAB command window:

```
model_a( data, 1.0, 0.001);
```

16. MATLAB will print a series of values on the screen and then eventually it will produce the magnitude and phase plots.
17. If the plots are not close then one of two things has happened:
    - Your initial guess was way off
    - Your data, or at least some of it, was not correct
18. At this point you may want to check any data that does not look too good and rerun the programs before you go on. Remember that if you add more data points, you need to rerun ***process_data.m*** to change your data before generating the Bode plot.
19. To add more data points to fill out your graph (spread throughout the frequency range). Your final plot should have at least 15 frequencies. Note that you can just add your new data to the end of the matrix ***measured*** in ***process_data.m*** since it gets sorted. Figure 4 provides an example of an acceptable Bode plot for a first order system.
20. Use the final Bode plot to estimate the system the cutoff frequency. It should be close to 160 Hz.
21. ***Include your final Bode plot figure in your memo and attach the process_data.m in the Angel drop box***

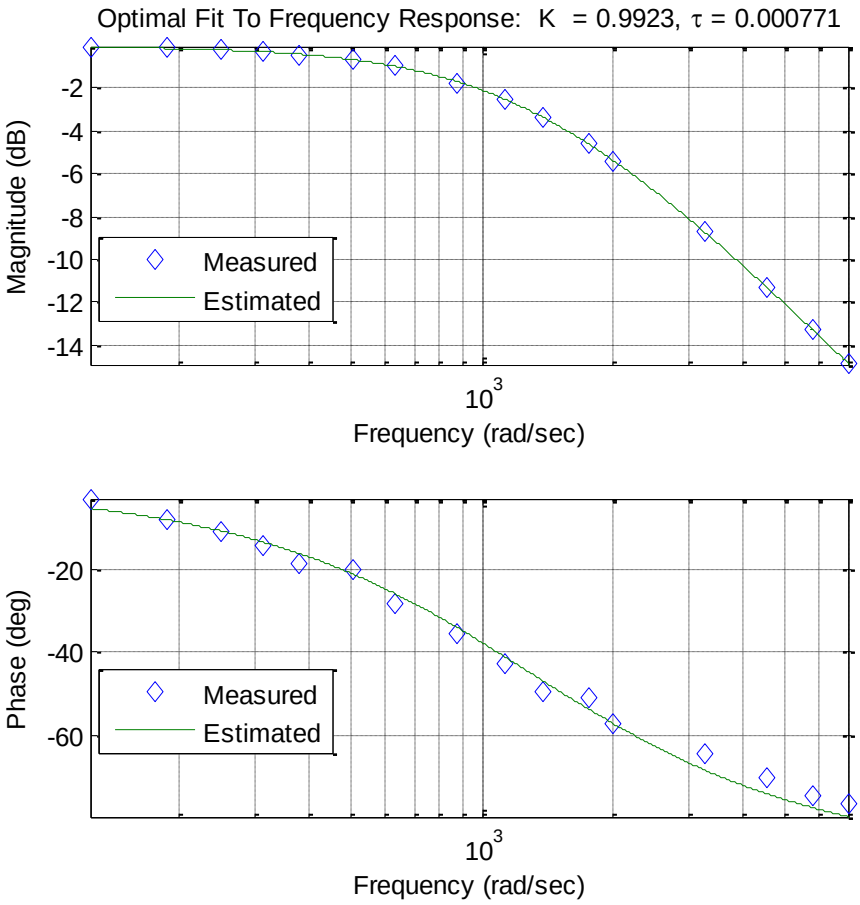


Figure 4: Bode Plot of the first order system

PART II: Frequency Response of a First Order System Again

1. Connect the NI MyDAQ to your laptop and start MultiSIM. You will actually create a design to output the simulation and measured Bode plot data on the same graph.
2. Click File → New → NI myDAQ Design. The screen will be automatically populated with the MyDAQ connection points.
3. Place the following components on the sheet: RESISTOR_RATED, CAPACITOR_RATED.
4. Next connect the RC circuit to the terminals to the MyDAQ terminals in MultiSIM just as you did on the breadboard so that the circuit looks like Figure 4:

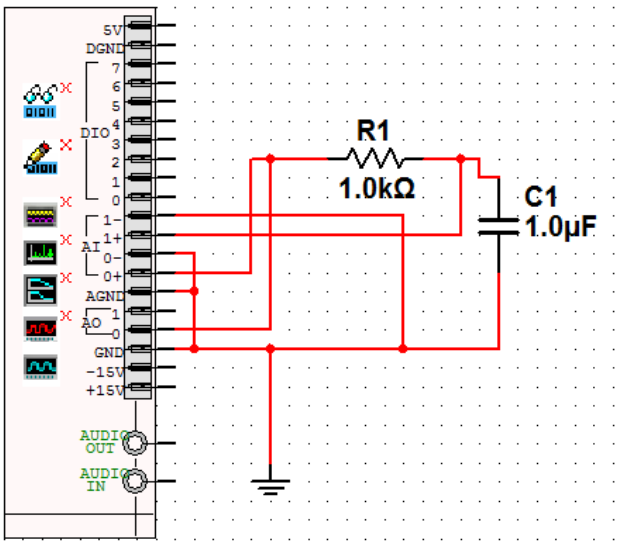


Figure 4: RC Circuit in MultiSim

5. Right click the toolbar at the top of the Multisim window and select NI ELVISmx Instruments. Once the toolbar opens, select the Bode Analyzer which is the last button.
6. Connect the Stimulus to the input the circuit and the Response to the output of the circuit so that it looks like Figure 5.

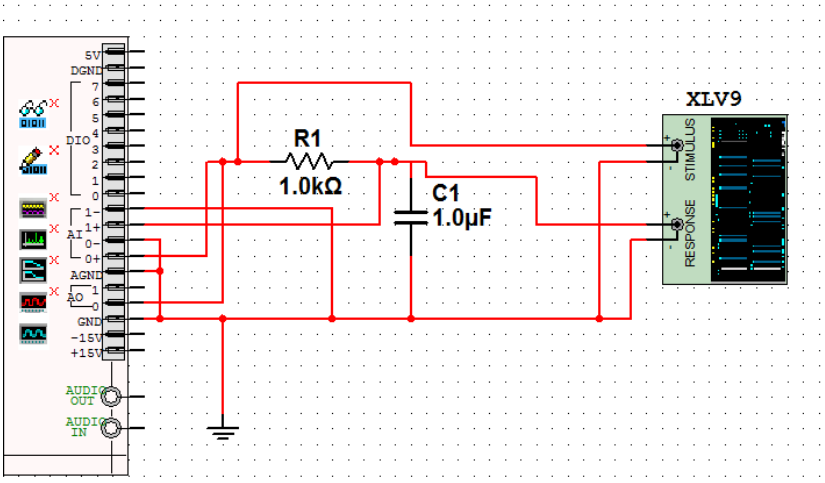


Figure 5: Bode Analyzer

7. Next double click the Bode Analyzer to open up the front panel. Also double click on the Function Generator on the MultiSim MyDAQ graphic at the bottom left.
8. On the Bode Analyzer, change the device to **Simulate NI myDAQ**, the start and stop frequencies to **10 Hz** and **20 kHz** and Steps to **10 (per decade).**

9. You don't have to change any settings on the function generator but click **Run** on the MultiSim window and **Run** on the Bode Analyzer graphic and the Bode plot should display.
10. **Stop** the simulation and change the Device to **Dev1 (NI myDAQ)**. The actual data from the circuit will now plot on the same figure. Your plot should look similar to Figure 6.
11. *Use the Bode plot to estimate the cutoff frequency for your system.* In your memo, you should compare and contrast this result to the MATLAB result.
12. *You should include this graph output in your memo.*

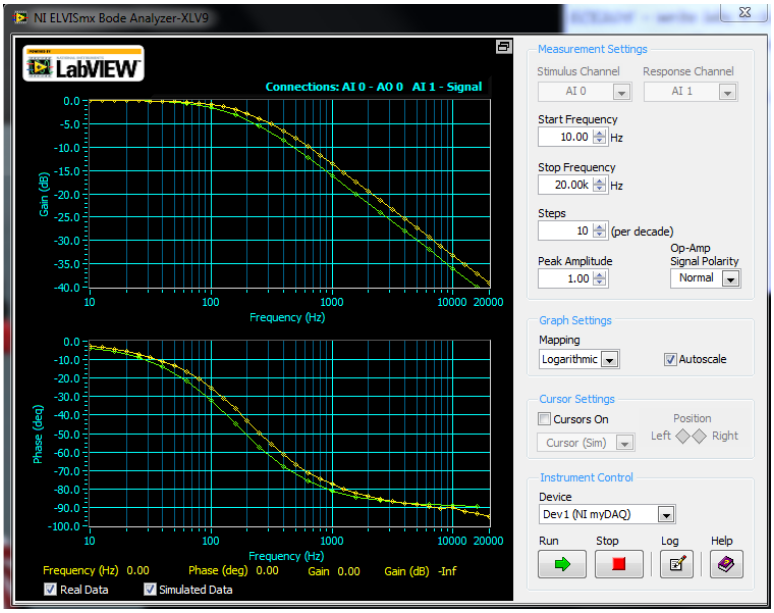


Figure 6: First order circuit Bode Plot (simulated and actual)

PART III: Frequency Response of a Second Order System

1. In this part, you will generate the actual and simulated Bode plot of a second order system.
2. Construct the circuit in Figure 7 on your breadboard with ***L = 33 mH***, ***C = 0.01 μF*** and R is the ***10 kΩ variable resistor***.
3. Connect the function generator and Channel 0 of the oscilloscope to the input. Connect Channel 1 of the oscilloscope across the capacitor to measure the output voltage.

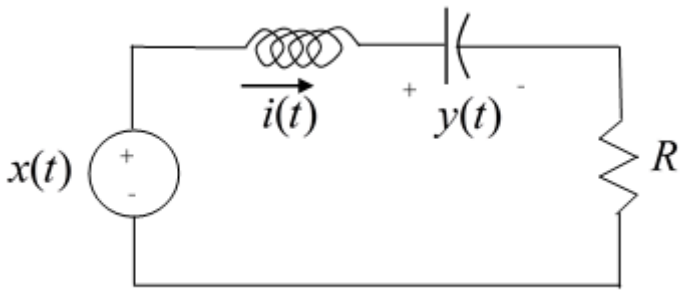


Figure 7: Second order system (RLC Circuit)

4. For this system the transfer function is $H(s) = \dfrac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$

   where K is the static gain, ζ is the damping ratio, and ω_n is the natural frequency.

5. This circuit is a **lowpass filter** with a gain of one. However, since it is second order there is a **resonant frequency** where the amplitude of the output is maximum at a small peak in the Bode plot.

6. Start the MyDAQ function generator and oscilloscope. Set the input to **1 kHz square wave** with **2.0 V peak to peak amplitude** and a **1.0 V offset**. Set the oscilloscope to **100 μs/div** and both channels to **1V/div**. You will need an **Edge** trigger with a level of **0.1 V**.

7. Run the function generator and oscilloscope and adjust the variable resistor so that there is a **50% overshoot** for a step response (see Figure 8). Do not change the variable resistor anymore but remove it and measure the resistance for use in the Multisim simulation. Then insert the resistor back into the circuit on the breadboard.

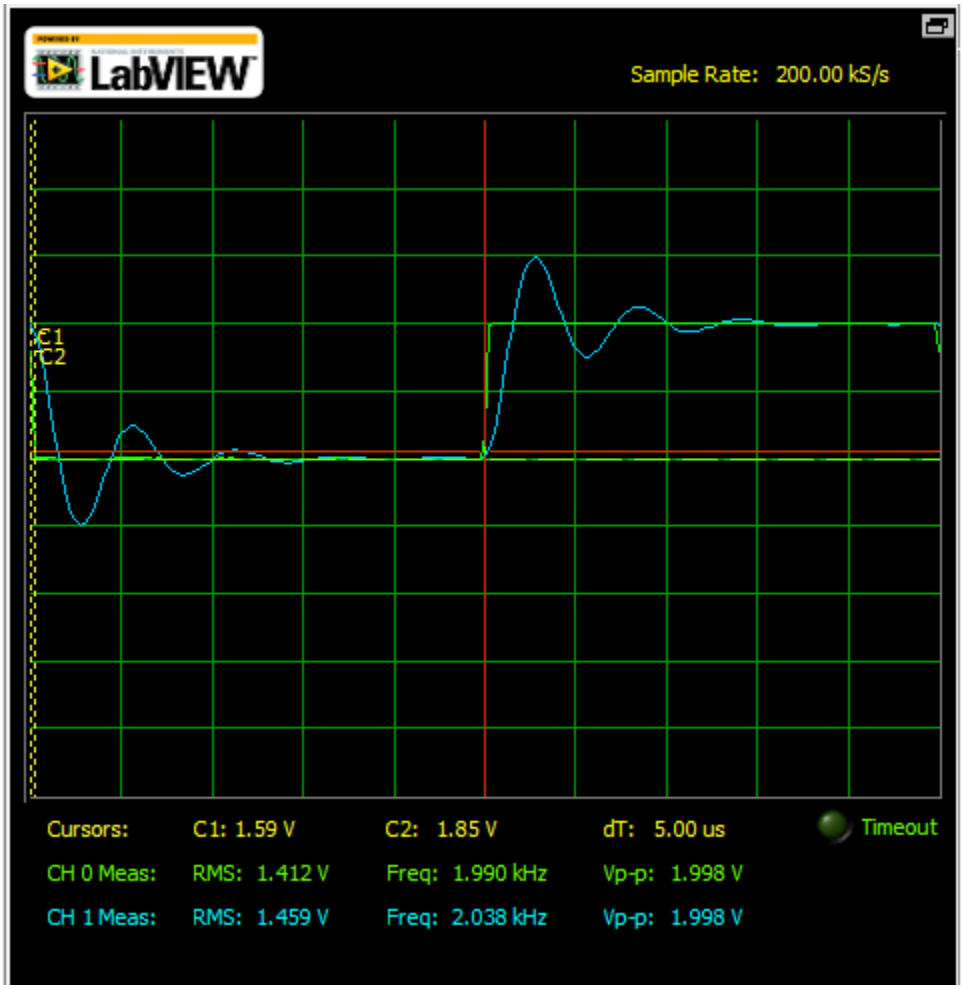


Figure 9: Second order system step response with 50% overshoot

8. Start Multisim and create a new NI MyDAQ design with the inductor, capacitor and resistor with the measured value of the potentiometer. Next, make similar connections to those for the circuit in the Multisim schematic in PART II.

9. Generate the Bode plot for the simulated and actual circuit and the results should look similar to Figure 10. The Bode plot for the actual system is correct but the magnitude and phase plot for the simulated system are actually a little bit off because the internal winding resistance of the inductor and capacitor and this has not been considered in the simulated design. In addition, there is a bug in the simulated phase plot that should be ignored.

10. *Use the Bode plot to estimate the cutoff frequency, resonant frequency and maximum amplitude in absolute and decibel measurements.*

11. *You should include this graph output in your memo.*

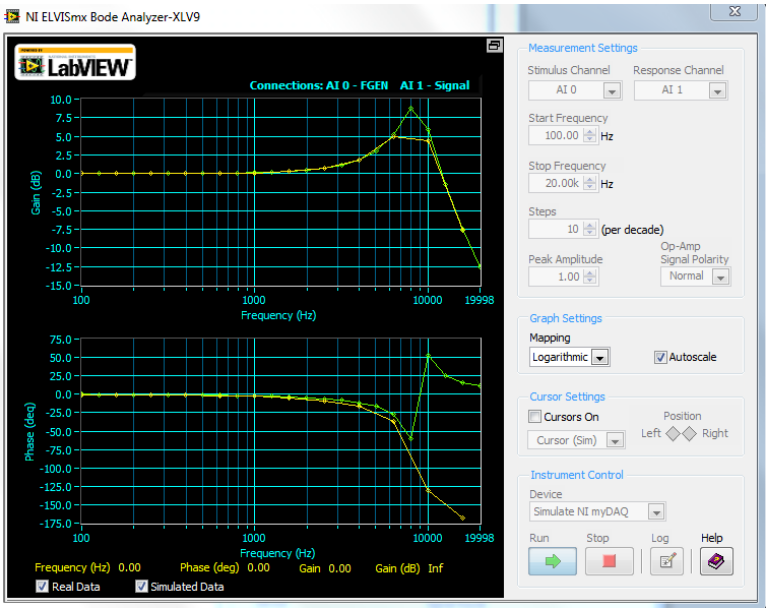


Figure 10: Second order circuit Bode Plot (simulated and actual)

PART IV: AC Analysis of a Second Order System in Multisim

1. In this part, you will build the series RLC circuit in Multisim where the value of the resistor is the measured value of the potentiometer. The source should be an **AC_VOLTAGE** and keep the default values. Your final schematic should look similar to Figure 11.

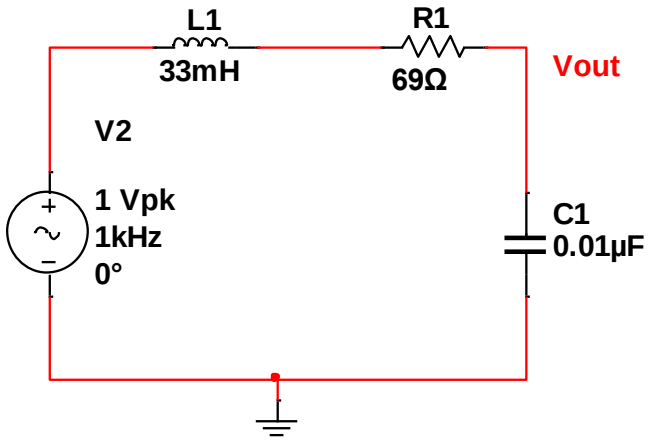


Figure 11: Second order system with AC source in Multisim

2. Click Simulate -> Analyses -> AC Analysis. Set the start frequency to **100 Hz** and the stop frequency to **20 kHz**.
3. Go to the output tab and select the voltage across the output to add to the right column. Click simulate.
4. Estimate the resonant frequency and maximum amplitude from this plot.
5. Compare and contrast these results to those found in Part III.
6. ***You should include a screenshot of this figure in the lab memo submission.***

PART V: Analysis of Bode Plots

Bodes plots are useful for determining the steady state output of a system given the input.

There is nothing to turn in for this part, but you are required to know this!

For system A, whose frequency response is displayed in the Bode plot in Figure 12, verify the following steady state outputs for the given steady state inputs. *If you cannot, please ask for help!*

$$x(t) = 3\cos(10t) \rightarrow y_{ss}(t) \approx 150\cos(10t - 100^o)$$
$$x(t) = 5\sin(40t + 50^o) \rightarrow y_{ss}(t) \approx 1.6\sin(40t - 130^o)$$

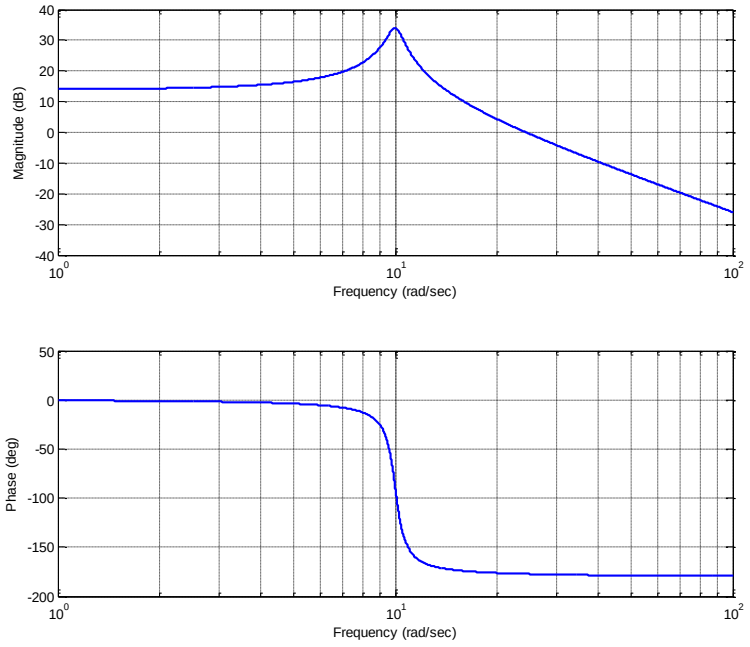


Figure 12: Frequency response of system A

For system B, whose frequency response is displayed in the Bode plot in Figure 13, verify the following steady state outputs for the given steady state inputs. *If you cannot, please ask for help!*

$$x(t) = 50\sin(10t) \rightarrow y_{ss}(t) \approx 0.63\sin(10t + 40^o)$$
$$x(t) = 100\cos(200t + 35^o) \rightarrow y_{ss}(t) \approx 4\cos(200t - 5^o)$$

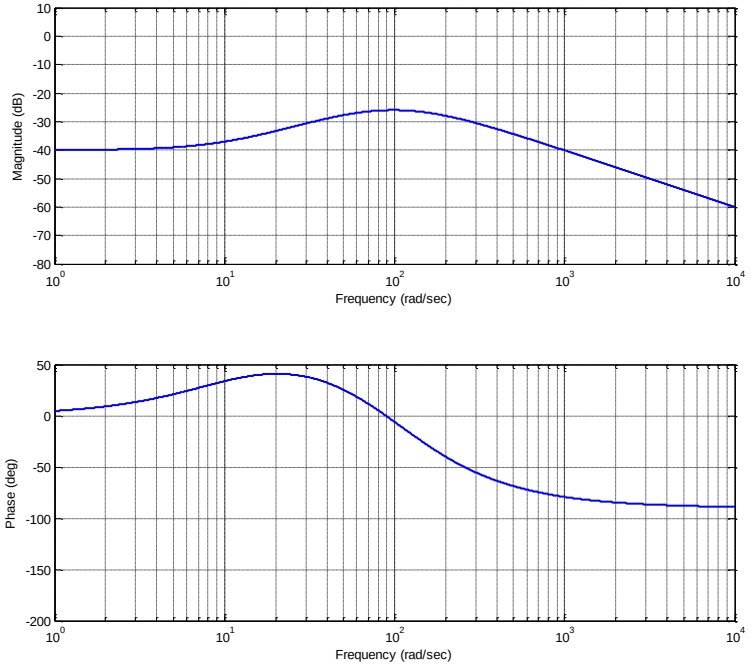


Figure 13: Frequency response of system B

Submission Requirements:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- A header at the beginning with *Date, To, From, Subject*
- Written in first person from you

- Written with minimal spelling and grammar errors
- *Purpose, procedure, results* and *conclusions* of the laboratory experiment
    - The very first sentence is the purpose and must explain why you are doing the experiment
    - The *procedure* should be very short, a high level summary of what you did for each part. The *procedure* should be two paragraphs at the most.
    - The *results* should include the tables and figures and results of the data collected and your observations. The results cannot be pages of figures and tables only, it **must** include text that references the figures and explains what they are to the reader.
    - The *conclusions* must be the end and it should be a summary of what you did, what you learned, what you observed and recommendations
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text. **DO NOT SUBMIT SEPARATE FILES,** images must be pasted into the Word document at the appropriate place with respect to the text. **DO NOT** just insert a bunch of figures and tables at the end of the document.
- Also, you must include a statement at the end of your memo that states that this submission is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results. This discussion may be a part of the results and/or conclusions sections of the memo

ECE-205 Lab 10

Active Filters

Overview:

In lab 9, you built two passive low pass filters and generated the Bode plots. A passive filter has a gain of one and is composed of only resistors, inductors and capacitors. In this lab you will build an active low pass filter, active high pass filter and an all pass filter. An active filter may have a gain greater than one and is composed of operational amplifiers in addition to the passive circuit elements. When these three components are cascaded together, they create a broadband bandpass filter. Finally, you will construct the magnitude portion of the Bode plot for each state of the filters and then put them together.

Equipment:

3 - TL072 operational amplifiers

2 - 1 μF capacitors

4 - 1 kΩ resistors

3 - 10 kΩ potentiometers

Theory:

The bandpass filter in this lab has three parts, shown in the different shaded regions in Figure 1. The equivalent block diagram is shown below the circuit, so that you can match elements of the block diagram with the circuit subsystems. The first subsystem is the low pass filter and it has a cutoff frequency of $1/(R_aC)$ rad/s and a gain of R_a/R. The second subsystem is the high pass filter and it has a gain of R/R_b and a cut off frequency of $1/(R_bC)$ rad/s. The last subsystem is an all pass filter which just adjusts the gain of the system.

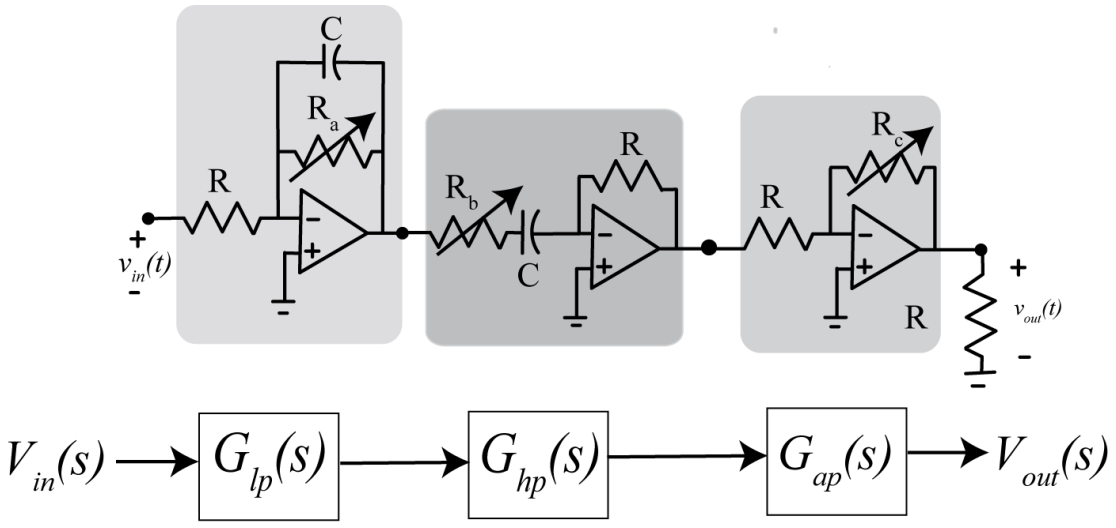


Figure 1: Broadband bandpass filter schematic and block diagram

Prelab:

1. Build the circuit in Figure 2 in MultiSim and run an AC analysis from 10 Hz to 100 kHz, what is the cut off frequency and gain? You should include the gridlines and reverse the plot to have a white background. Include the plot in your prelab submission. Review the end of Lab 9 if you don't recall how to run an AC analysis in MultiSim.

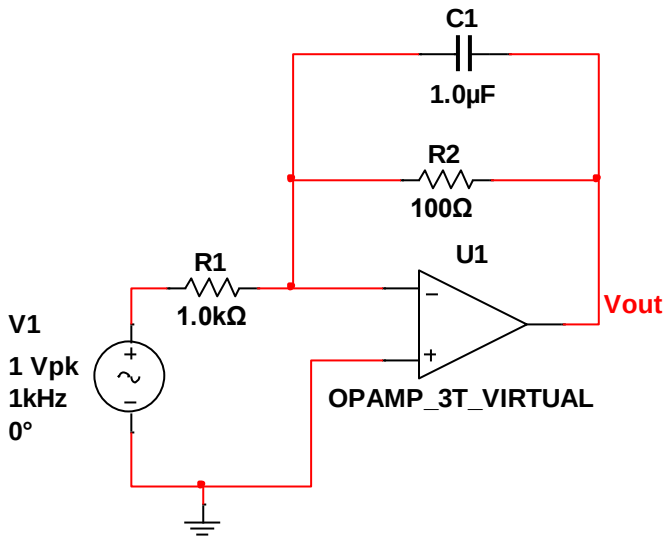


Figure 2: Active lowpass Filter

2. Build the circuit in Figure 3 in MultiSim and run an AC analysis from 10 Hz to 100 kHz, what is the cut off frequency and gain? You should include the gridlines and reverse the plot to have a white background. Include the plot in your prelab submission.

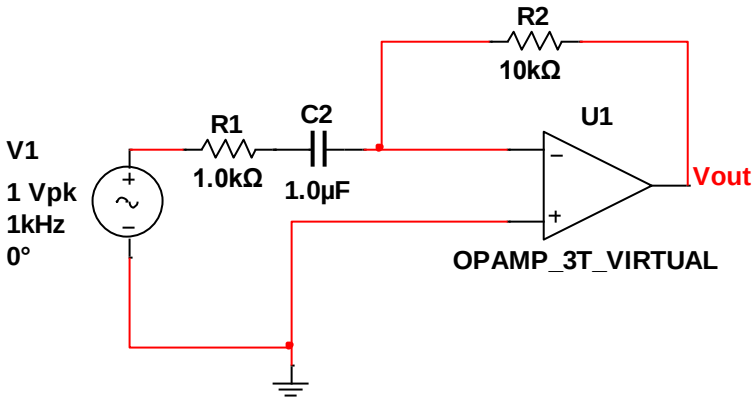


Figure 3: Active highpass Filter

3. Build the circuit in Figure 4 in MultiSim and run an AC analysis from 10 Hz to 100 kHz. What are the two cut off frequencies and the gain? You should include the gridlines and reverse the plot to have a white background. Include the plot in your prelab submission.

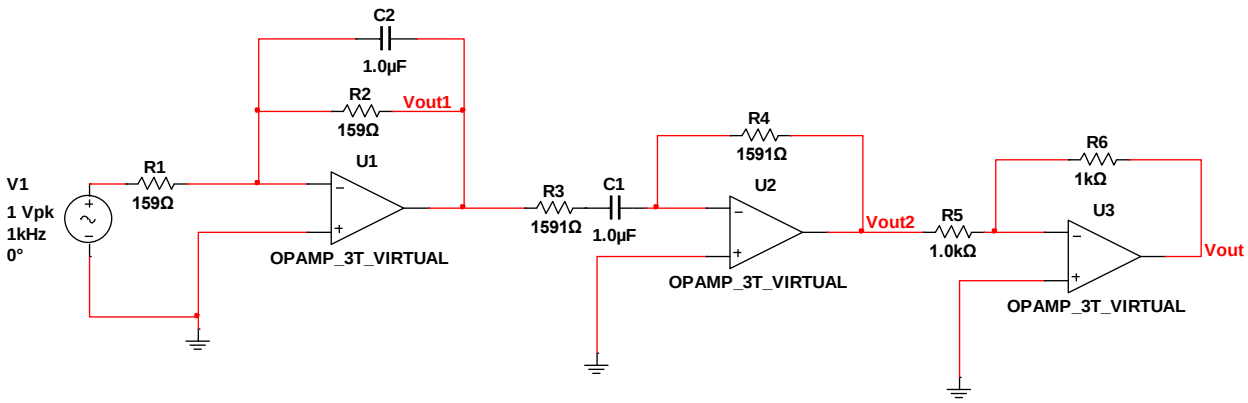


Figure 4: Broadband bandpass filter

4. Adjust the component values so that the bandpass filter has a gain of 10 and a lower cut off frequency of 700 Hz and an upper cut off frequency of 10 kHz. You should include the gridlines and reverse the plot to have a white background. Include the plot in your prelab submission.
5. Finally, include the final schematic for the broadband bandpass filter in your prelab submission. ***In summary, you should print out 4 plots and one schematic for submission.***

Procedure:

PART I: Build the lowpass filter

1. Turn the breadboard so that the positive red buss is on the top.
2. Connect a green wire between the two blue busses to create a common ground.
3. Place the 3 TLO72 op amps equally spaced down the middle of the breadboard ditch with the dot (notch) to the left.
4. Connect pin 4 of all three op amps to the bottom red buss which represents **-Vcc (-15 V)**.
5. Connect pin 8 of all three op amps to the top red buss which represents **+Vcc (+15V)**.
6. Connect pin 3 of all three op amps to the blue buss which represents **AGND**.
7. Connect the **+15 V** on the power supply to the top red buss. Connect the **-15 V** on the power supply to the bottom red buss. Connect the **AGND** on the power supply to the blue buss.
8. Build the lowpass filter in Figure 5 on the first TL072 chip. All of the resistors in this lab are 1 kΩ and all of the capacitors are 1 µF.
9. The input to this circuit should be the function generator (**AO 0, AO AGND**). Set the function generator to a **10 Hz sine wave** with a **2 V peak to peak** amplitude and 0 V offset.
10. The input should be measured by Channel 0 of the oscilloscope (**AI 0+, AI 0-**). The output should be measured at pin 1 (**AI 1+, AI 1-**) on the op amp.

11. Set the time base on the oscilloscope to **20 ms/div**. Set Channel 0 to **500 mV/div** and set Channel 1 to **100 mV/div**. Set the trigger to **Edge** with the **Channel 0** source.

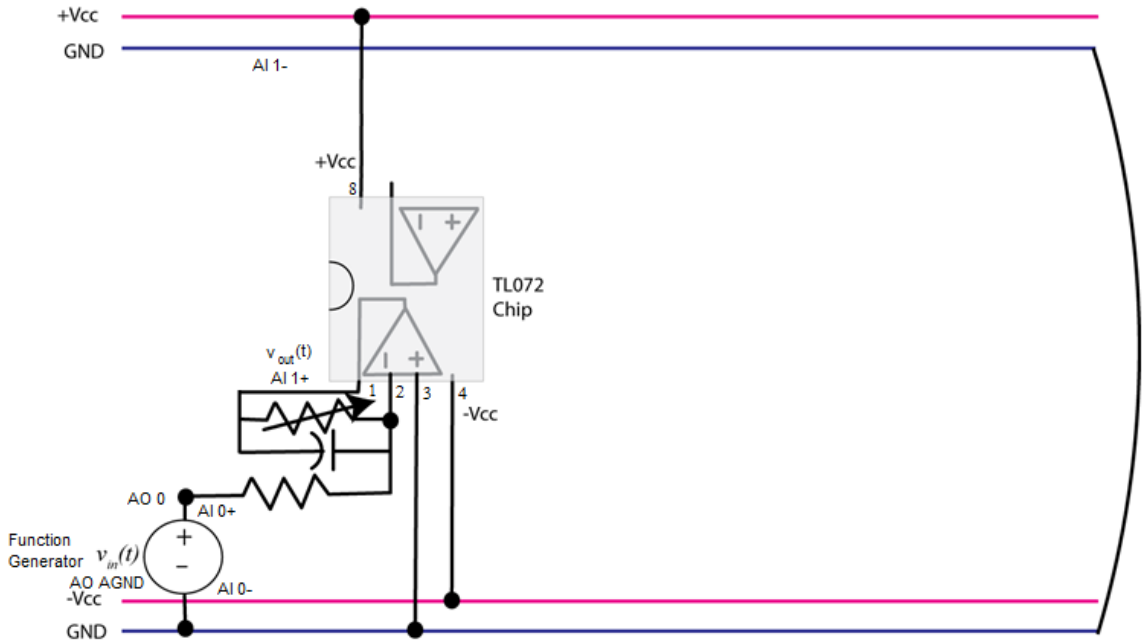


Figure 5: Active lowpass filter schematic

12. Adjust the variable resistor so the output has an amplitude of approximately **0.6 volts peak to peak** or a **gain of 0.3**.

13. The transfer function for this lowpass filter can be shown to be

$$G_{lp}(s) = \frac{-K_{low}\omega_{low}}{s + \omega_{low}}$$

where $K_{low} = \frac{R_a}{R}$ and $\omega_{low} = \frac{1}{R_a C}$

PART II: Construct the Bode amplitude plot for the lowpass filter

1. In this part you will construct the magnitude portion of the Bode plot for the lowpass filter.
2. Create a data table in Excel with the columns frequency (Hz), input (mV), output (mV).
3. For input frequencies of 100, 200, 400, 600, 700, 800, 1000, 1500, 2000, 2500, and 3000 Hz, keep the input amplitude at 1V (2 Vpp) and use the cursors to measure the amplitude of the output as the frequency is varied. Record the output amplitude on the Excel data table. Note that as your input frequency increases, you will need to decrease the Time/Div setting in order to keep only one or two periods of the waveform on the screen. Also, try and keep Channel 1 between **20 mV/div** and **100 mV/**div to keep the waveform as large as possible on the screen.
4. Download the ***process_data_low.m*** file from the Angel course Lab 10 folder. In this file, each row corresponds to a frequency. The three entries in the row are the frequency

(measure in Hz), the amplitude of the input signal (measured in mV), and the amplitude of the output signal (measured in mV).

5. Copy the data from Excel into the ***measured*** matrix variable.
6. In the MATLAB command window type **data = process_data_low;**
7. Now we will fit the data to our model using the program **model_low.m**. The arguments to this file are the data array from part b, the estimated amplitude of the gain, and the estimated value of the cutoff frequency.
8. In the Matlab command window type **model_low(data, 0.3, 2*pi*500);**
9. You should get results similar to that shown in Figure 6. Click on the Datatip icon (just below Desktop) yo identify points on the graph. Once you have identified the maximum value, you may have to use a right click and create a new data tip (it usually helps if you right click when you are on a current datatip). You can fine tune your datatip locations by using the cursor keys on your keyboard. Your data points should be from the peak value to a value 3 dB lower. *Include this graph in your memo. Be sure your graph has two datatip points.*

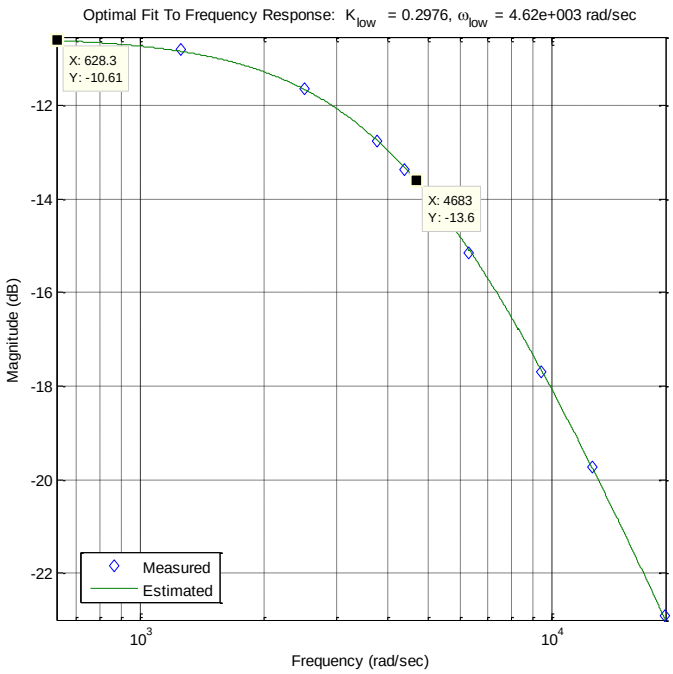


Figure 6: lowpass filter with a bandwidth of approximately 4683 rad/sec (745 Hz) and gain of -10.61 dB (0.295)

PART III: Construct the Bode amplitude plot for the lowpass filter in Multisim

1. Create a new NI myDAQ design in Multisim and place the NI ELVISmx Bode Analyzer on the sheet and wire it as shown in Figure 7.

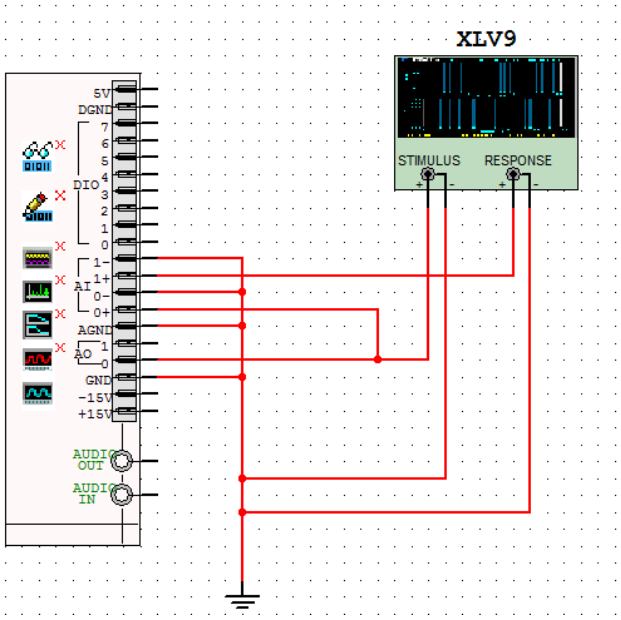


Figure 7: NI myDAQ design in Multisim

2. Open up the Bode Analyzer and change the start frequency to 100 Hz and the stop frequency to 20 kHz.
3. Change the Device to the NI myDAQ.
4. Run the simulation and the results should look similar to Figure 8.
5. Compare and contrast this result to the MATLAB result in PART II.
6. ***Capture the result for inclusion in your lab memo submission.***

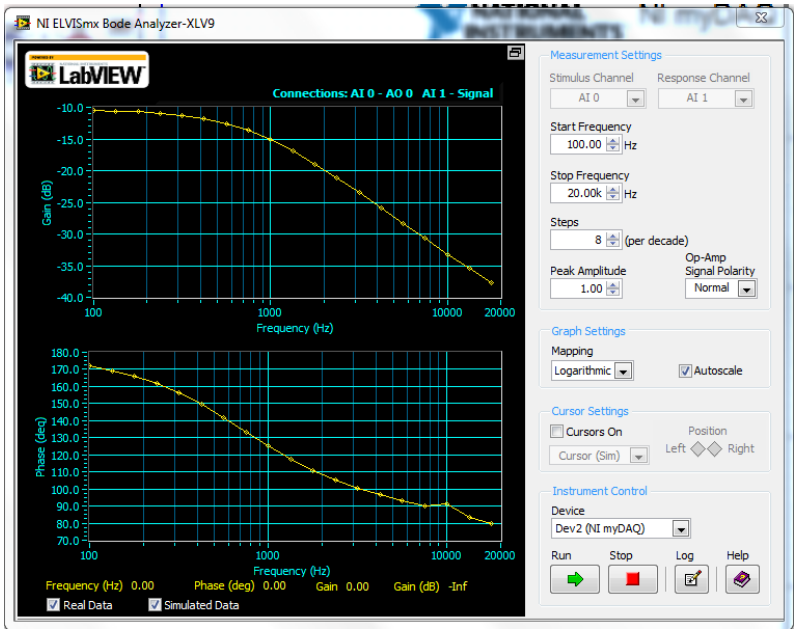


Figure 8: NI myDAQ design lowpass filter Bode Analyzer

PART IV: Construct the highpass filter

1. Now build the highpass filter as shown in Figure 9 on the second TL072 op amp. Do not connect the two subsystems yet.
2. Set the function generator to a **2 V peak to peak sine wave** with **0 V DC offset** at **1 kHz**.
3. Connect function generator and Channel 0 of the oscilloscope to the input of the high pass filter. Connect Channel 1 of the oscilloscope to pin 1 of the high pass filter to measure the output.
4. Set the time base of the oscilloscope to **200 μs/div** and set both channels to **100 mV/div**. Set the trigger type to **Edge** and the **Chan 0 Source**.
5. Adjust the variable resistor that the output signal has approximately **0.8 volts peak to peak** or a **gain of 0.4**.

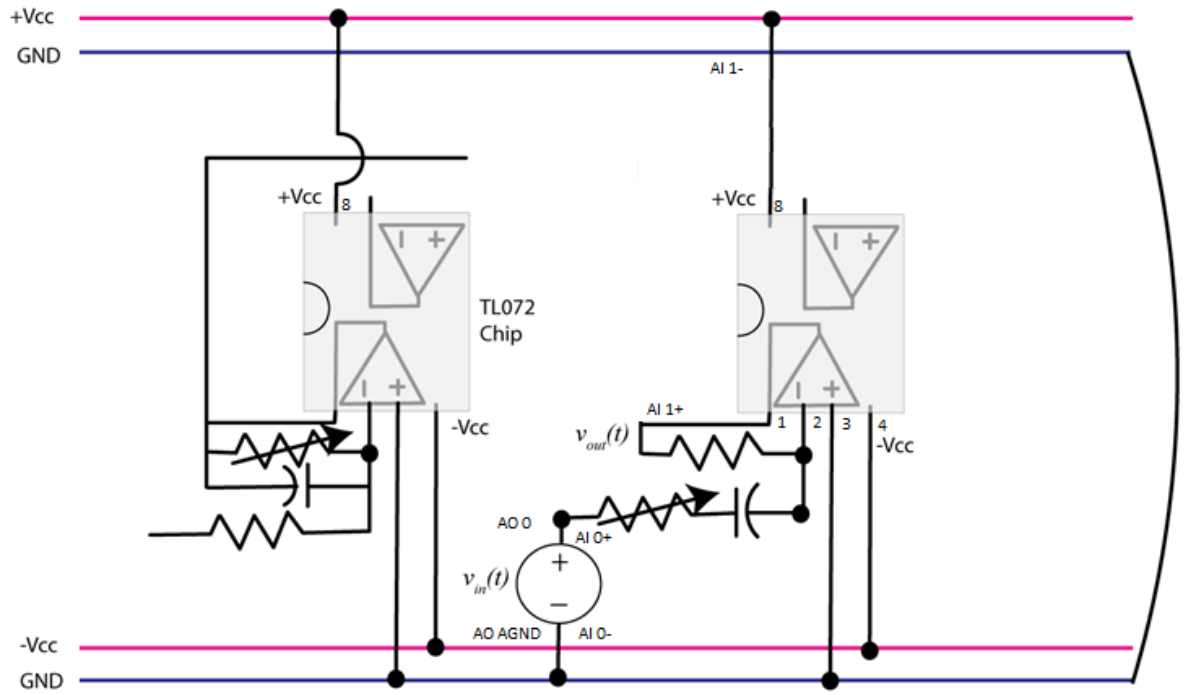


Figure 9: lowpass and highpass subsystems

6. The transfer function for this subsystem can be shown to be

$$G_{hp}(s) = \frac{-K_{high}s}{s + \omega_{high}}$$

where $K_{high} = \frac{R}{R_b}$ and $\omega_{high} = \frac{1}{R_b C}$

PART V: Construct the Bode amplitude plot for the highpass filter

1. Create a new sheet in the Excel workbook for the high pass filter and create the same columns frequency (Hz), input amplitude (mV) and output amplitude (mV).

2. For input frequencies of 10, 25, 50, 75, 100, 200, 300, 400, 500, 750, and 1000 Hz, keep the input amplitude at 2 V peak to peak and use the cursors to measure the amplitude of the output signal and record it on the Excel sheet. You may need to vary the timebase and volts/division in order to display one or two periods of the waveform as large as possible on the screen.
3. Copy the program **process_data_low.m** and rename the file, **process_data_high.m**. and enter your data into this new file for the **measured** matrix. Do not just reuse your previous file.
4. In the MATLAB command window type **data = process_data_high;**
5. Download the **model_high.m** script from the Angel course Lab 10 folder. The arguments to this file are the data array, the estimated amplitude of the gain, and the estimated value of the cutoff frequency.
6. In the MATLAB command window type **model_high(data, 0.4, 2*pi*200);** You should get results similar to that shown in Figure 10. Click on the Datatip icon (just below Desktop) so identify points on the graph. Once you have identified the maximum value, you may have to use a right click and create a new data tip(it usually helps if you right click when you are on a current datatip). You can fine tune your datatip locations by using the cursor keys on your keayboard. Your data points should be from the peak value to a value 3 dB lower.
7. ***Include this graph in your memo. Be sure your graph has two datatip points.***

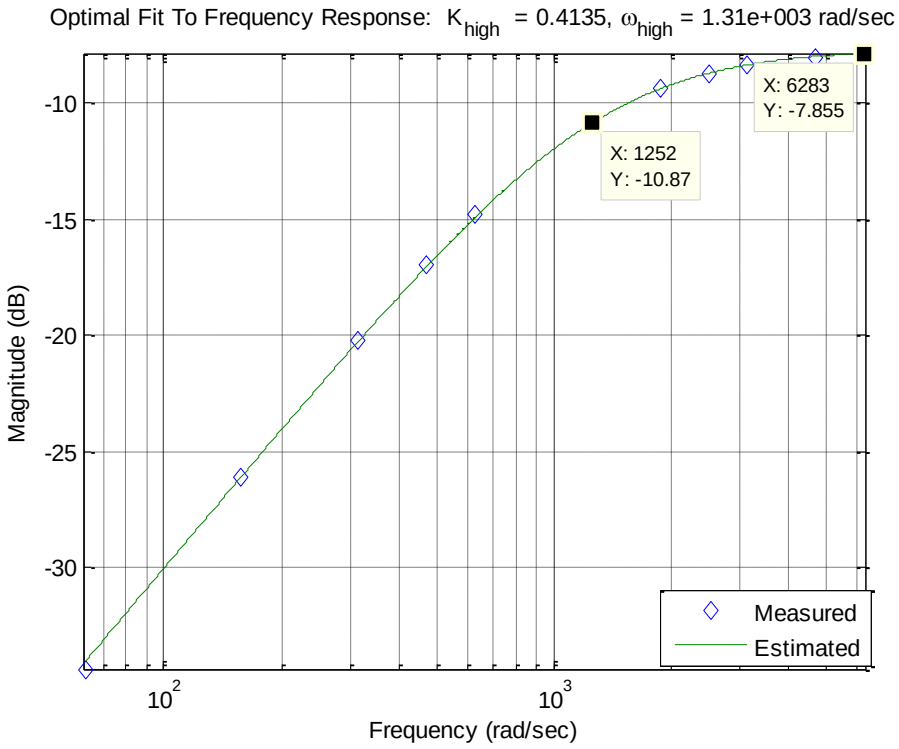


Figure 10: Highpass filter with a cutoff frequency of approximately 1.31 krad/s (208.5 Hz) and gain of -7.855 dB (0.405).

PART VI: Construct the Bode amplitude plot for the highpass filter in Multisim

1. Repeat PART III for the high pass filter to get the Bode amplitude plot using the Bode analyzer in Multisim.
2. You can actually use the same Multisim schematic that you created in Figure 7.
3. Run the simulation and compare and contrast the results with the MATLAB results.
4. ***Capture the result for inclusion in your lab memo submission.***

PART VII: Construct the all pass filter (inverting amplifier)

1. Construct the gain subsystem, as shown in Figure 11.
2. Adjust the variable resistor so that for an input **sine wave** with a **2 Vpp** amplitude at **500 Hz**, the output is approximately **2V pp** or the system has a **gain of 1**.

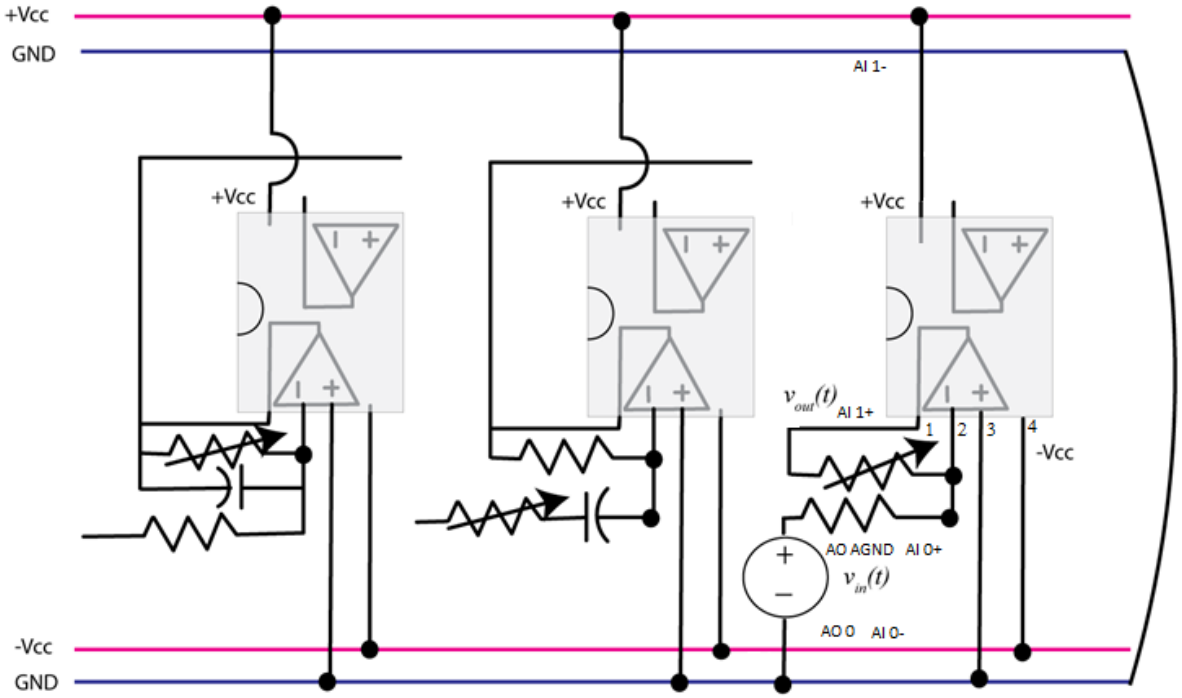


Figure 11: Lowpass, highpass, and all pass subsystems.

3. The transfer function for this subsystem can be easily shown to be

$$G_{ap}(s) = -K_{ap}$$

$$\text{where } K_{ap} = \frac{R_c}{R}$$

PART VIII: Construct the Bandpass filter

1. Now you will use 2 wires to connect all of the of the subsystems together as shown in Figure 12.

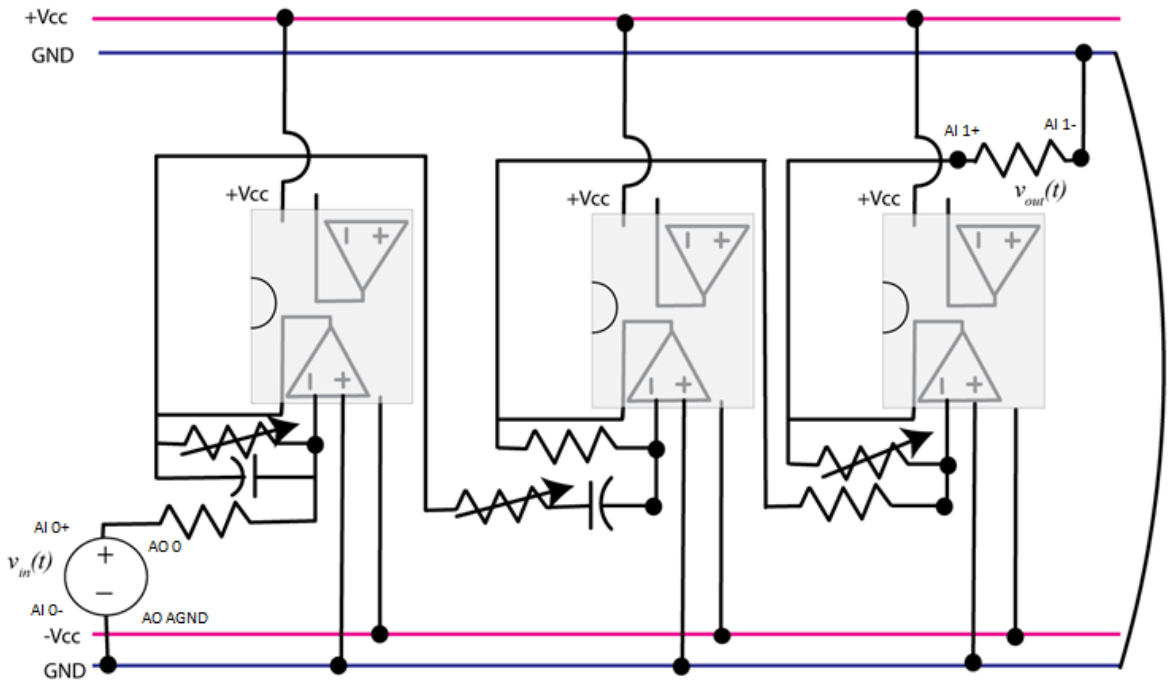


Figure 12: Broadband Active Bandpass filter

2. The transfer function for this system is then the product of the transfer functions of the subsystems,

$$G_{bp}(s) = G_{lp}(s)G_{hp}(s)G_{ap}(s) = \left(-\frac{K_{low}\omega_{low}}{s+\omega_{low}}\right)\left(-\frac{K_{high}s}{s+\omega_{high}}\right)\left(-K_{ap}\right)$$

   We can simplify this as

$$G_{bp}(s) = -\frac{Ks}{s^2 + (\omega_{low} + \omega_{high})s + \omega_{low}\omega_{high}}$$

3. Since we are only interested in the magnitude, we can ignore the negative sign (but you will see it on the output since it will be out of phase with the input).
4. Finally, we will construct the magnitude portion of the Bode plot for this simple bandpass filter.
5. For input frequencies of 10, 15, 25,40, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500 and 2000 Hz, keep the input amplitude at **2 Vpp** and measure the amplitude of the output signal.
6. Initially set Channel 0 to **200 mV/div** and Channel 1 to **10 mV/div** and the timebase to **20 ms/div**. However, you will have to change the timebase and volts/division as the frequency is varied.
7. Copy the program **process_data_high.m** onto a new file **process_data_bandpass.m**, and enter your data into this new file. Do not just reuse your previous file.
8. In the MATLAB command window type **data = process_data_bandpass;**

9. Download the **model_bandpass.m** file from the Angel course Lab 10 folder. The arguments to this file are the data array, the estimated amplitude of the gain, and the estimated value of the high and low cutoff frequencies.
10. In the Matlab command window type

```
model_bandpass( data, 0.3*0.4, 2*pi*500, 2*pi*200);
```

11. You should get results similar to that shown in Figure 13. Note that your peak amplitude may be different depending on how much you could adjust the gain on the final stage. At this point we mostly are looking for the same general shape.
12. Click on the Datatip icon (just below Desktop) so identify points on the graph. Once you have identified the maximum value, you may have to use a right click and create a new data tip (it usually helps if you right click when you are on a current datatip). You can fine tune your datatip locations by using the cursor keys on your keyboard. Your data points should be from the peak value to a value 3 dB lower on both sides.
13. ***Include this graph in your memo. Be sure your graph has three datatip points.***

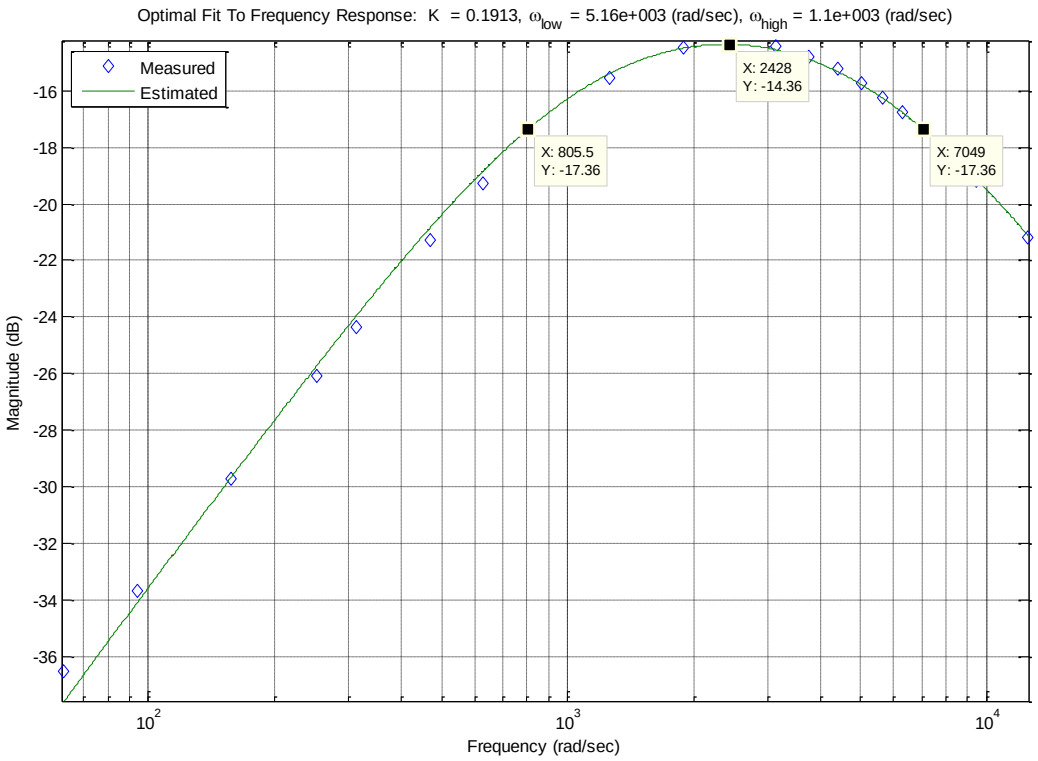


Figure 13: Bandpass filter with an approximate bandwidth of 6244 rad/sec (993 HZ) and Gain of -14.36 dB (0.191)

PART IX: Construct the Bode amplitude plot for the bandpass filter in Multisim

1. Repeat PART III for the bandpass filter to get the Bode amplitude plot using the Bode analyzer in Multisim.

2. You can actually use the same Multisim schematic that you created in Figure 7.
3. Run the simulation and compare and contrast the results with the MATLAB results.
4. ***Capture the result for inclusion in your lab memo submission.***

Submission Requirements:

The lab memo should be submitted to the instructor via the Angel Course Drop Box by midnight on **Sunday**. After midnight on Sunday, the memo is late and incurs a **20% penalty per day**. If it is not submitted by the beginning of the next lab session, the grade is a **zero**. At a minimum it should include the following:

- Typewritten, 12 point font
- A header at the beginning with *Date, To, From, Subject*
- Written in first person from you
- Written with minimal spelling and grammar errors
- *Purpose, procedure, results* and *conclusions* of the laboratory experiment
  - The very first sentence is the purpose and must explain why you are doing the experiment
  - The *procedure* should be very short, a high level summary of what you did for each part. The *procedure* should be two paragraphs at the most.
  - The *results* should include the tables and figures and results of the data collected and your observations. The results cannot be pages of figures and tables only, it **must** include text that references the figures and explains what they are to the reader.
  - The *conclusions* must be the end and it should be a summary of what you did, what you learned, what you observed and recommendations
- The entire memo should be concise and to the point.
- All required figures and files generated using the NI myDAQ or MATLAB or MultiSim with number and caption and they should be referenced in the text. **DO NOT SUBMIT SEPARATE FILES,** images must be pasted into the Word document at the appropriate place with respect to the text. **DO NOT** just insert a bunch of figures and tables at the end of the document.
- Also, you must include a statement at the end of your memo that states that this submission is your own work.
- The discussion should include a compare and contrast of the theoretical results or nominal results to the actual results. This discussion may be a part of the results and/or conclusions sections of the memo