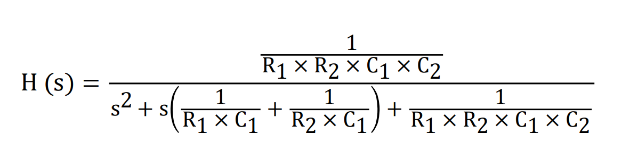
Unit 1: Characterizing the Plant

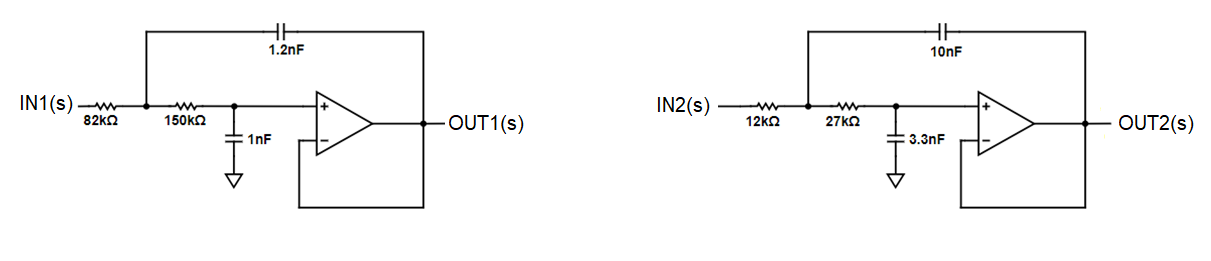
U1: Theory

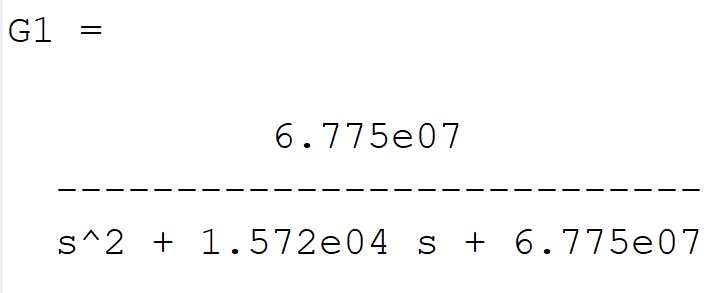
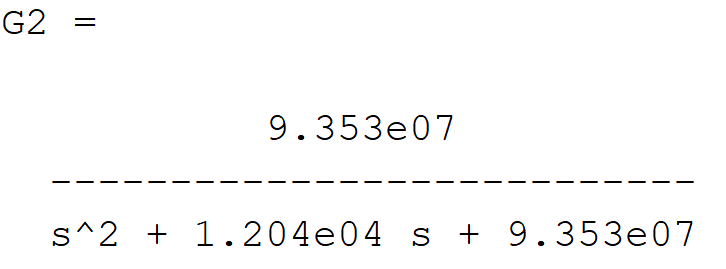
The design of electronic filters is a well-established process in Electrical Engineering. The building block for many electronic filters is the Sallen-Key architecture shown below. The transfer function, show to the right of the circuit, realizes a second order system. In other words, the transfer function has two poles.



Using the transfer function equation above, determine the transfer function for

and .





A pole of a transfer function is a value of s that causes the denominator of the transfer function to equal 0. In general, the poles of a transfer function are complex numbers. Determine the poles of G1(s) and G2(s), plot them on the graph in Figure 1 and fill in their values in the Theory row of Table 1. Note that the axes in Figure 1 are scaled by a factor of 1,000.

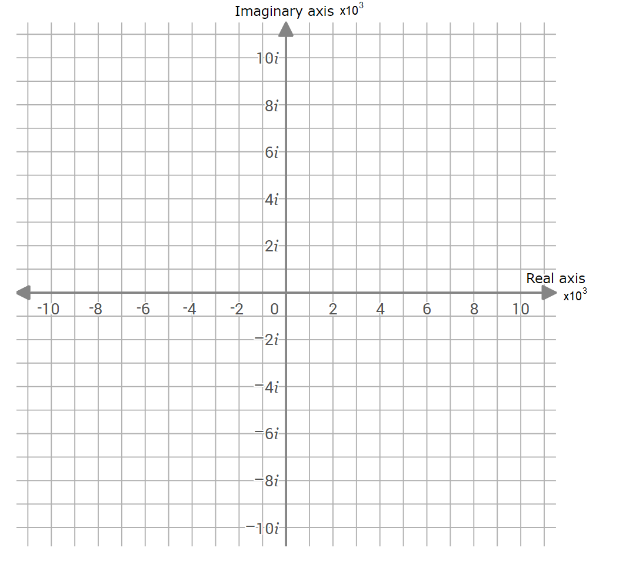


Figure 1: Poles of G1(s) and G2(s).

Now, determine the poles of the composite function G1(s)\*G2(s). To determine this, let’s call the denominator of G1(s), d1(s) and the denominator of G2(s), d2(s). Now, the denominator of G1(s)\*G2(s) is d1(s)\*d2(s) (the numerator is unimportant). To find the pole locations, consider what happens to the product d1(s)\*d2(s) when s is equal to a pole location of d1(s). Hint, d1(s) will equal 0 for this value of s.

U1: Simulation

Verify your conclusions about pole location of G1(s)\*G2(s) using MATLAB. Before launching MATLAB, make sure that the control systems toolbox is installed by going to your MATLAB install directory, go into the toolbox directory and make sure that there is a subdirectory called “control”. If not, you will need to install the control systems toolbox when you next launch MATLAB.

Create a new script called **poleZero.m** with the following text.

Listing 1: The MATLAB script to plot the poles and zeros of G1(s)\*G2(s).

s=tf(‘s’);

G1 = a1/(s^2 + b1\*s + c1); // replace a1, b1 and c1 with correct values

G2 = a2/(s^2 + b2\*s + c2); // replace a2, b2 and c2 with correct values

pzmap(G1\*G2); // plots the poles of G1\*G2 as “x”

Before running this script, make sure to replace the constants a1… c2 with the values you found for G1(s) and G2(s). Note, the pzmap function will plot the poles (shown as “x”) of the transfer function given as the argument to pzmap. You can click on the x’s to show their Cartesian coordinates. Do this to complete

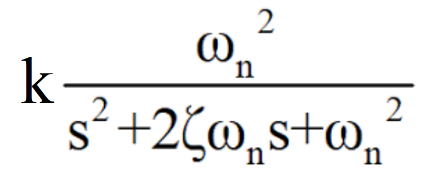
Table 1: Poles of G1(s) and G2(s) determines in two different ways.

|  |  |  |
| --- | --- | --- |
|  | Poles of G1(s) | Poles of G2(s) |
| Theory | s = -7.86\*103 ± 2.45\*103 | s = -6.02\*103 ± 7.57\*103 |
| Simulation | s = -7.86\*103 ± 2.45\*103 | s = -6.02\*103 ± 7.57\*103 |

Unit 2: Open-Loop Step Response

U2: Theory

In the previous lab, we discovered that the transfer function for the two filters, which compose our plant, are second order systems. The canonical form for a second order system is shown in the equation below.



When a second order systems is put into its canonical form, you can predict its step response. Remember that the step response is the output of the system when a step waveform is applied to the input.

Convert the transfer functions G1(s) and G2(s) you found in the previous unit into their respective canonical forms. From these canonical forms, determine the values of k, ω and ζ for G1(s) and G2(s) and put them into the Theory row of Table 2. From the values of ω and ζ estimate the rise time, settling times using and overshoot the equations and graphs introduced in class. Put these values into the Theory row of Table 2.

Now use the information you just entered into the Theory row of Table 2 to hand-plot the expected step response of G1(s) and G2(s) in Figure 2.

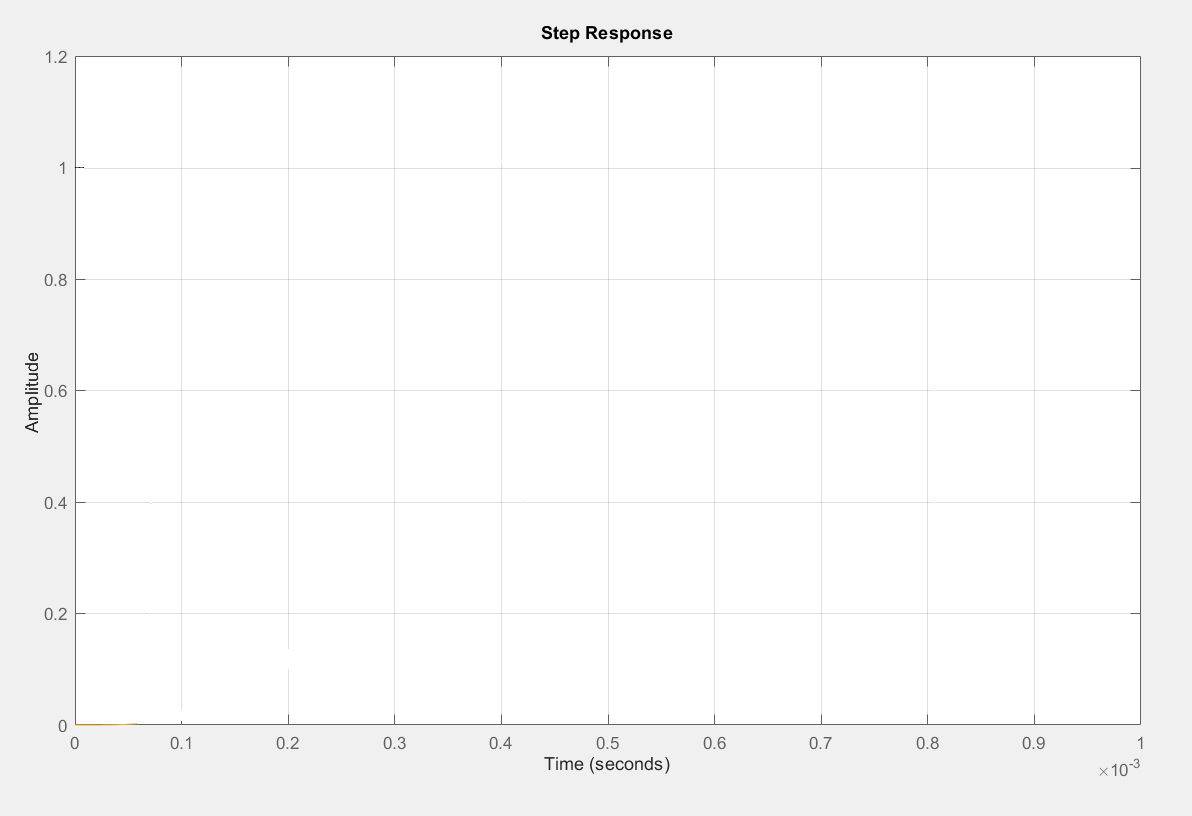


Figure 2: Step response of G1(s), G2(s) determine by approximations for rise and settling time.

U2: Simulation

Before launching MATLAB, make sure that the control systems toolbox is installed by going to your MATLAB install directory, go into the toolbox directory and make sure that there is a subdirectory called “control”. If not, you will need to install the control systems toolbox when you next launch MATLAB.

Let’s first check that our open-loop description of G1(s) and G2(s) are consistent with the values found in the first table.Now, create a new script called **openLoopStepResponse.m** with the following text.

Listing 2: A MATLAB script to plot the step response of G1(s), G2(s) and G1(s)\*G2(s).

s=tf(‘s’);

G1 = a1/(s^2 + b1\*s + c1); // replace a1, b1 and c1 with correct values

G2 = a2/(s^2 + b2\*s + c2); // replace a2, b2 and c2 with correct values

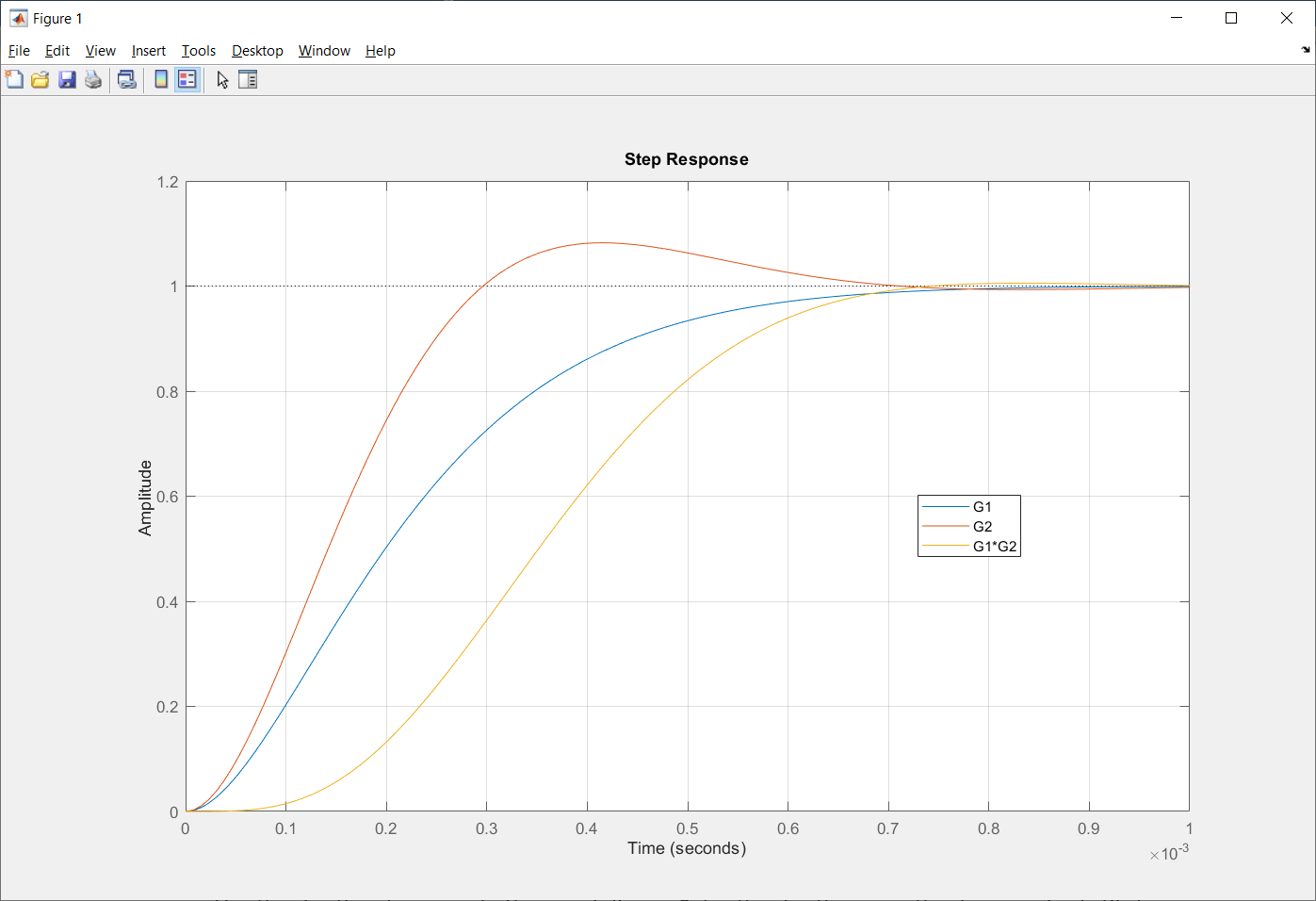
stepinfo(G1)

stepinfo(G2)

stepinfo(G1\*G2)

step(G1,G2,G1\*G2);

legend({'G1','G2','G1\*G2'})



Use the information from this MATLAB script to fill in the Simulation row Table 2 as follows.

* Use the information provided by the stepinfo MATLAB command to fill in the rise and settling time columns. Note the MATLAB settling time is 2%. In order to convert it into a 1% settling time you used to compute the Theory row. In order to convert the MATLAB time to our theory time use the relationship ts(2%) = 0.85\*ts(1%).
* Use the final value of the step responses to fill in the k column values.
* Measure the peak overshoot. Use this to compute the percentage overshoot and fill in this column.
* Use the rise time to compute the ω as follows. Solve the rise time equation for ω and substitute in a value of rise time to determine the corresponding value of ω.
* Use the settling time to compute ζ as follows. Solve the settling time equation for ζ and substitute in a value of settling time and ω to determine the corresponding value of ζ. Note, ζ cannot be greater than 1. If your calculations produce a value larger than 1, set ζ equal to 1.

U2: Experimental

Now it’s time to go into the laboratory and check that your theory and simulations agree with an actual circuit. To do this, you will need to

* Power Supply with cables and alligator clips
* Oscilloscope and two probes

PID Development board configuration:

|  |  |
| --- | --- |
| Input select | STEP |
| Jumper wire | **STEP header**  **LPF\_IN header** |
| Jumper | **LPF\_OUT**  **LOAD\_OUT** |

Power Supply configuration:

|  |  |
| --- | --- |
| Ch 1 Voltage | 9V |
| Ch 1 Current | **0.1A** |
| Ch 2 Voltage | **9V** |
| Ch 2 Current | **0.1A** |

Wire channel 1 and 2 in series, creating a +9V, GND and -9V supply. Verify these voltages with a DMM prior to connecting anything to the PID development board.

Configure the oscilloscope as follows:

|  |  |
| --- | --- |
| Ch1 probe | SET header |
| Ch ground clip | **GND loop** |
| Ch 2 probe | **OUT header** |
| Horizontal scale | **200us** |
| Ch 1 scale | **2V** |
| Ch 2 scale | **2V** |
| Ch 1 offset | **Center reticule** |
| Ch 2 offset | **Center reticule** |
| Trigger mode | **Auto** |
| Tigger source | **Ch 1** |
| Trigger slope |  |
| Trigger level | **0V** |

When correctly configured, your setup should look like the image below. Now go ahead and record the performance values from the oscilloscope trace and put your values in Table 2.

Press and release the STEP button to produce a step input to the plant G1(s). Repeat for G2(s) and then G1(s)\*G2(s).

Table 2: Open performance of the plant. Round your values to 3 significant figures.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | k | ω | ζ | tr | ts | %OS |
| Theory  G1  G2  G1\*G2 | 1  1  --------- | 8,231 rad/sec  9,671 rad/sec  --------- | 0.96  0.62  --------- | 0.27ms  0.23ms  --------- | 0.58ms  0.77ms  --------- | 0%  10%  --------- |
| Simulation  G1  G2  G1\*G2 | 1  1  1 | 5,789 rad/sec  11,000 rad/sec  7,700 rad/sec | 1.0  0.79  1.0 | 0.38ms  0.20ms  0.38ms | 0.55ms  0.53ms  0.57ms | 0%  8.2%  0.54% |
| Experimental  G1  G2  G1\*G2 |  | ---------  --------- |  |  |  |  |

Unit 3: Closed Loop Step Response

U3: Theory

Now let’s turn our attention to the closed loop system G1(s)\*G2(s). Figure 3 shows a closed loop system with plant G1(s)\*G2(s). Note that the transfer function of this

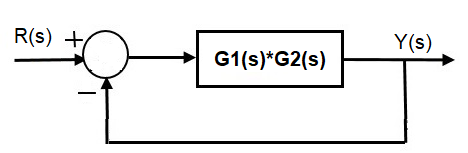
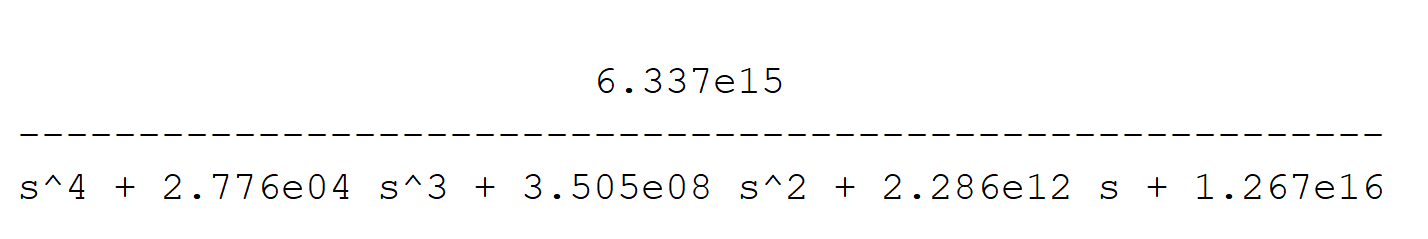


Figure 3: The closed loop system with plant G and feedback transfer function H.

Start by computing the closed loop transfer function using the transfer functions for G1(s)\*G2(s) found in the previous unit.



This fourth order system is difficult to analyze so let’s approximate it with a 2nd order system approximation. To do this, first determine the poles of the fourth order system describing .

The roots of are: s = -1,870±7,820j -12,000±7,200j

Now, let’s bend our approximation rules and call the pair of poles closest to the origin (at around -2000) dominant and the pair furthest from the origin (at around -12,000) subordinate. Write an approximation for using the pair of poles closest to the origin and dropping the pair of poles furthest from the origin. To determine k, set the

The dominant pole is p1 = -1,870±7,820j we will drop the pole pair -12,000±7,200j When we solve for k, we get

Now put the approximation, into canonical form to determine k, ω and ζ.

Using k, ω and ζ from the canonical form, find the final value, rise time, settling time, percentage overshoot, oscillatory frequency of the overshoot (radians per second) of the . Put these values into the table in the Experimental section in the row labeled “Theory”.

A fundamental tool in understanding this system’s closed-loop behavior is the root loci diagram. Remember that the root loci describe the pole locations of the closed loop system as the gain is changed from 0 to infinity. In previous units you discovered that G1(s) is a second order system with a pair of poles in the left-hand plane. Accordingly, the loci for G1(s) will never wander into the right half plane. You should verify this for yourself if it is not obvious. The same result applies to G2(s); its loci will never wander into the right-half plane.

However, when we put G1(s) and G2(s) in series, the resulting transfer function has four poles and some of the loci will wander into the right-half plane. To check this, draw the root loci for G1(s)\*G2(s) in Figure 4. Start by plotting the poles of G1(s)\*G2(s). Then use the heuristics introduced in class to produce a hand-drawn root locus.

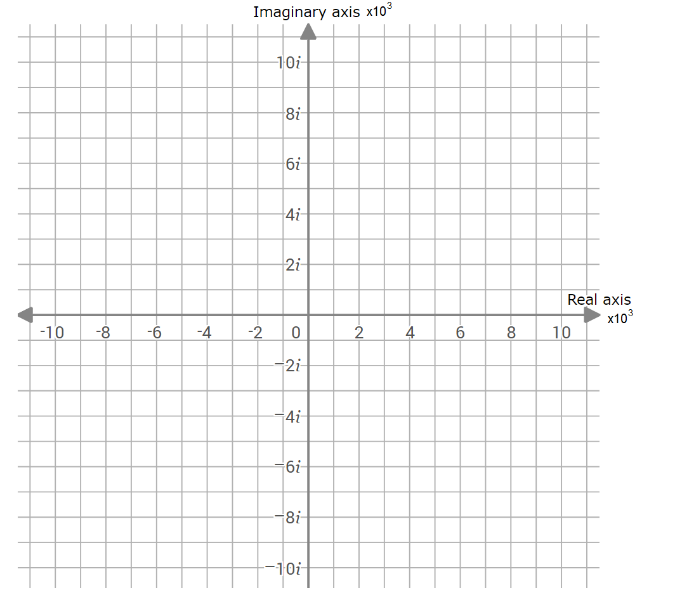
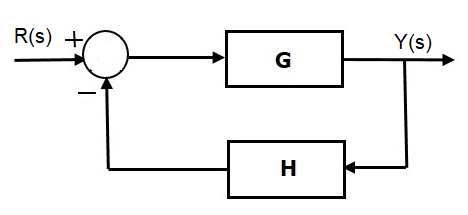


Figure 4: The hand-drawn root loci for G1(s)\*G2(s).

U3: Simulation

Let’s make quick work of the theoretical calculations with MATLAB. To do this, we will use the MATLAB function pzmap to look at the poles of the closed loops transfer function. In addition, we will use the MATLAB function feedback, to compute the closed loop transfer function.

The feedback function computes the transfer function of a feedback loop like that shown in the following figure. The first argument of the feedback function is the transfer function of the forward going arc, F in the following figure. The second argument in the feedback function is the transfer function of the backward going arc, G in the following figure.



So, in MATLAB, to compute the transfer function you would type feedback(G,H). Note, the return values from this MATLAB call is the symbolic form of the transfer function. This is handy because you can use this transfer function as the input to other function as shown in the following script.

Now, create a script called **closedLoopStepResponse.m** with the following text.

Listing 3: A MATLAB script to check the closed-loop behavior of G1(s)\*G2(s).

s=tf(‘s’);

G1 = a1/(s^2 + b1\*s + c1); // replace a1, b1 and c1 with correct values

G2 = a2/(s^2 + b2\*s + c2); // replace a2, b2 and c2 with correct values

pzmap(feedback(G1\*G2,1));

waitforbuttonpress();

rlocus(G1\*G2);

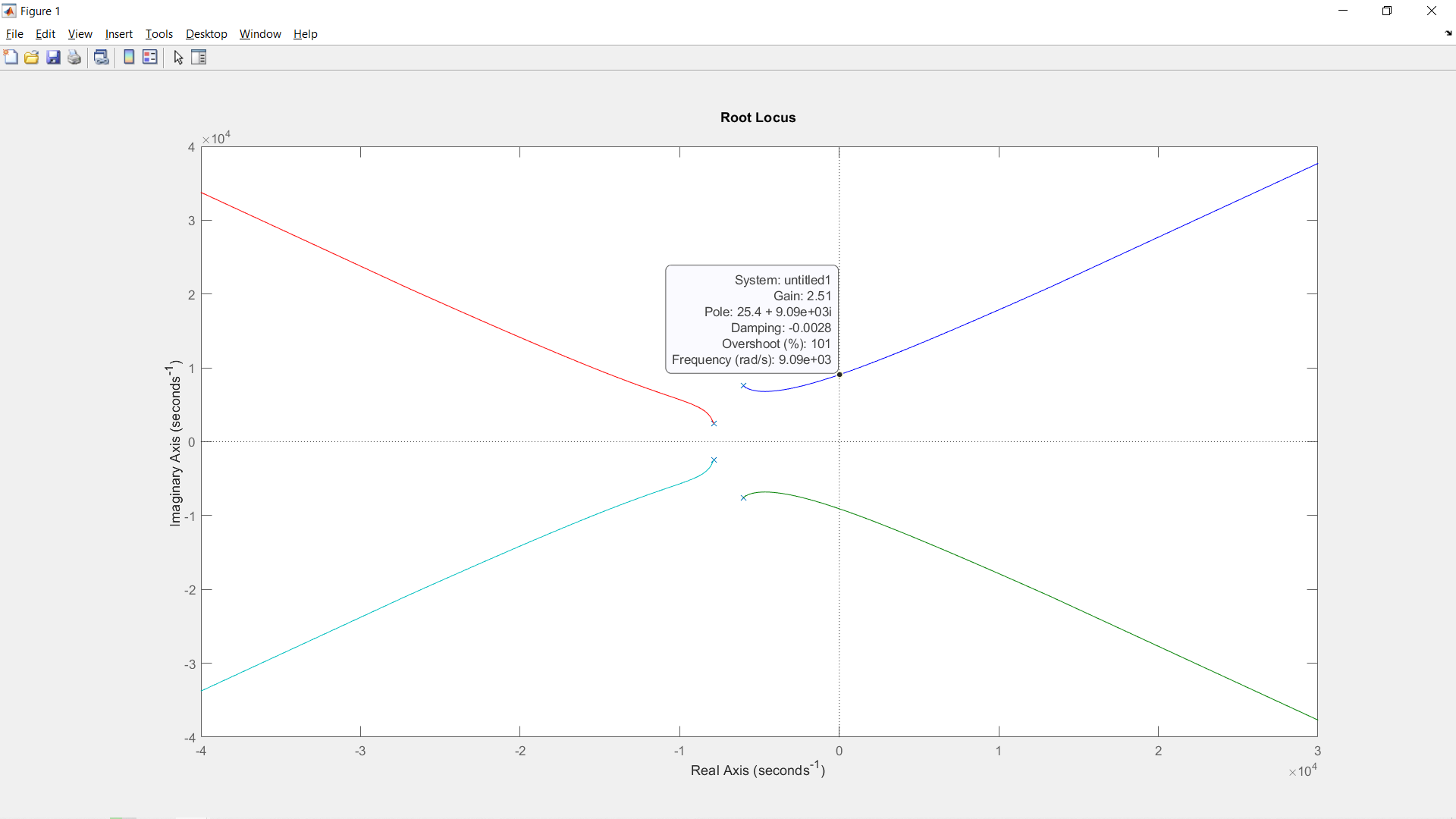
waitforbuttonpress();

step (feedback(G1\*G2,1));

waitforbuttonpress();

stepinfo (feedback(G1\*G2,1));

Verify that the root locus you drew by hand is similar to that produced by MATLAB (note the hand-drawn diagram should look slightly different than that produced by MATLAB). Click on the root locus produced by MATLAB at the point where one of the loci cross the imaginary axis. From the information provided in the pop-up, determine the gain at this pole location. Hint, the gain should be between 2 and 3.



Use the information in the step response graph and the stepinfo report to find the final value, rise time, settling time, percentage overshoot, oscillatory frequency of the overshoot (radians per second). Put these values in the complete the “Simulation” row of the table in the Experimental section.

U3: Experimental

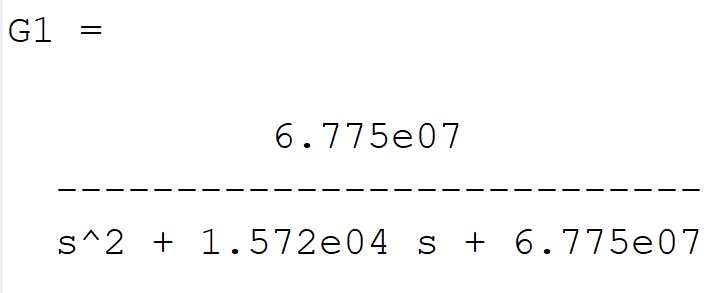
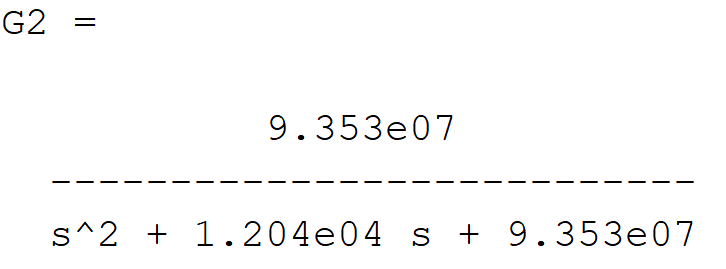
Table 3: Closed-loop performance of G1(s)\*G2(s)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Final Value | ω | tr | ts | %OS |
| Theory | 0.5 | 8,000 rad/sec | 0.28ms | 2.5ms | 40% |
| Simulation | 0.5 | 7,800 rad/sec | 0.2ms | 1.9ms | 40% |
| Experimental |  |  |  |  |  |

Unit 4: Open Loop Sinusoidal Response

U4: Theory

Now let’s turn our attention to the frequency response of the system G(s) = G1(s)\*G2(s), specifically the Bode plot. Let’s remember that the transfer function for each of these filters is given by the equations below.



Your first task will be to determine the magnitude and phase of G(s) = G1(s)\*G2(s) in Table 4. To simplify this process, compute the magnitude and phase for G1(s) and G2(s) separately, then combine the magnitudes and phase using the identities:

You can find the entries in Table 4 using any computational tool you want. I would suggest using Excel, because that is what I most often use to perform these reasonably straight-forward computations.

Table 4: The values of magnitude and phase of G(s) determined using equations.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ω (rad/sec) |  | radians |  | radians |  | 20log(|G|)  dBs | degrees |
| 100 | 1 | -0.02 | 1 | -0.01 | 1 | 0 | -2.08 |
| 1000 | 0.99 | -0.23 | 1 | -0.13 | 0.99 | -0.09 | -20.78 |
| 2200 | 0.94 | -0.51 | 1.01 | -0.29 | 0.95 | -0.44 | -45.63 |
| 4700 | 0.77 | -1.02 | 1.03 | -0.67 | 0.79 | -2.01 | -96.7 |
| 10000 | 0.42 | 1.37 | 0.78 | 1.52 | 0.33 | -9.73 | -194.17 |
| 22000 | 0.12 | 0.7 | 0.2 | 0.6 | 0.02 | -32.07 | -285.67 |
| 47000 | 0.03 | 0.34 | 0.04 | 0.26 | 0 | -57.81 | -325.72 |
| 100000 | 0.01 | 0.16 | 0.01 | 0.12 | 0 | -83.92 | -343.94 |
| 220000 | 0 | 0.07 | 0 | 0.05 | 0 | -111.29 | -352.71 |
| 470000 | 0 | 0.03 | 0 | 0.03 | 0 | -137.66 | -356.59 |

Plot this information on a log/log graph. From the plot,

* Find the frequency at which the magnitude is at -3dB
* Determine the slope in the stop band

U4: Simulation

We can use MATLAB to make short work of the Bode plot.

Now, create a script called **openLoopFreqResponse.m** with the following text.

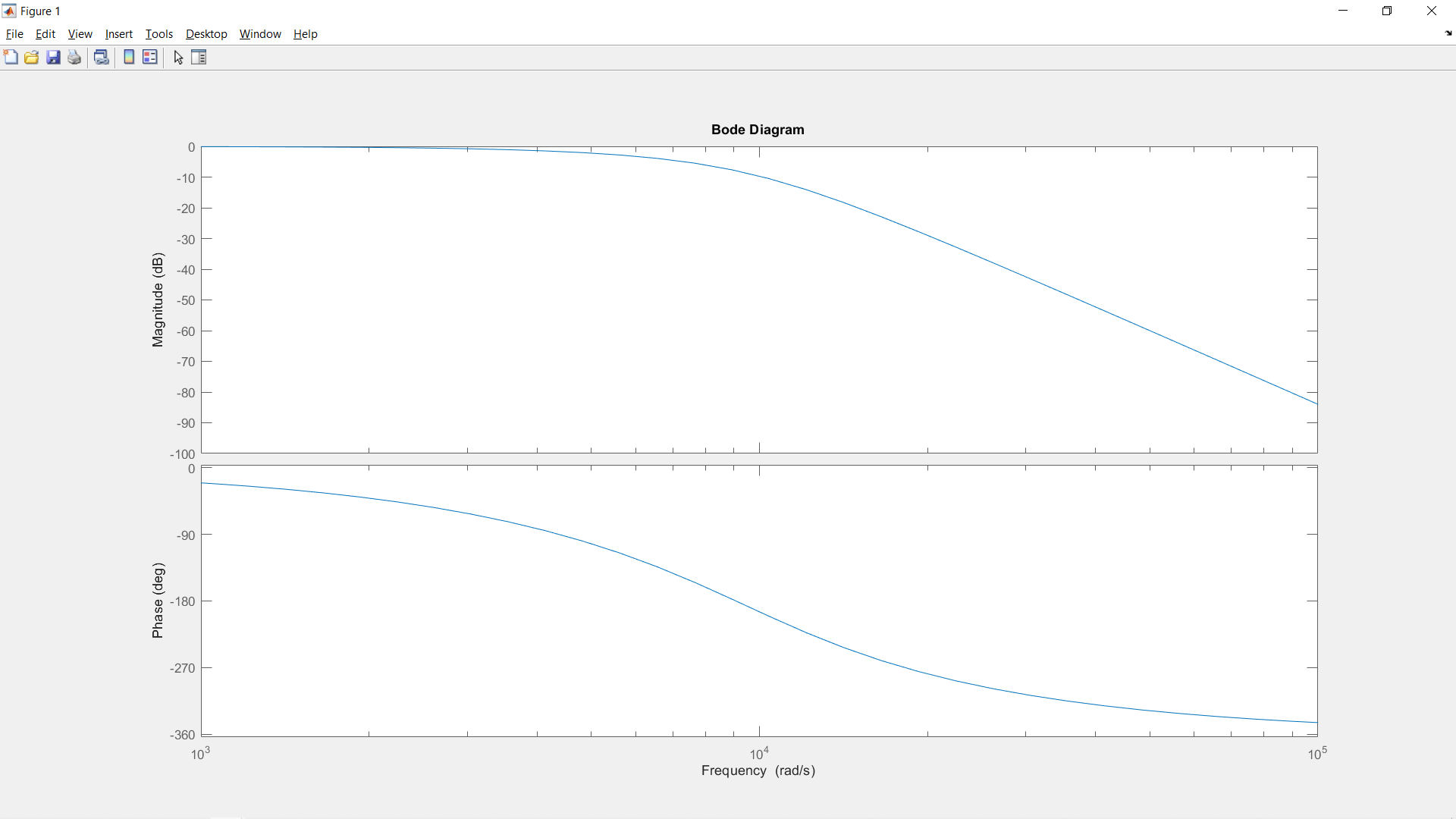
Listing 3: A MATLAB script to check the closed-loop behavior of G1(s)\*G2(s).

s=tf(‘s’);

G1 = a1/(s^2 + b1\*s + c1); // replace a1, b1 and c1 with correct values

G2 = a2/(s^2 + b2\*s + c2); // replace a2, b2 and c2 with correct values

bode(G1\*G2);



-3dB frequency at 5,730 rad/sec = 912Hz

Table 5: The values of magnitude and phase of G(s) determined using a MATLAB simulation.

|  |  |  |
| --- | --- | --- |
| ω (rad/sec) | 20log(|G|)  dBs | degrees |
| 100 |  |  |
| 1000 |  |  |
| 2200 |  |  |
| 4700 |  |  |
| 10000 |  |  |
| 22000 |  |  |
| 47000 |  |  |
| 100000 |  |  |
| 220000 |  |  |
| 470000 |  |  |

U4: Experimental

<copy lab instructions from EENG 385 labs>

Table 6: The values of magnitude and phase of G(s) determined using a MATLAB simulation.

|  |  |  |
| --- | --- | --- |
| ω (rad/sec) | 20log(|G|)  dBs | degrees |
| 100 |  |  |
| 1000 |  |  |
| 2200 |  |  |
| 4700 |  |  |
| 10000 |  |  |
| 22000 |  |  |
| 47000 |  |  |
| 100000 |  |  |
| 220000 |  |  |
| 470000 |  |  |

U4: Conclusions

Table 7: Summary of the magnitude and phase calculations performed in this unit.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ω (rad/sec) | dBs | degrees | dBs | degrees | dBs | degrees |
| 100 | Theory | | Simulation | | Experimental | |
| 1000 | 0 | -2.08 |  |  |  |  |
| 2200 | -0.09 | -20.78 |  |  |  |  |
| 4700 | -0.44 | -45.63 |  |  |  |  |
| 10000 | -2.01 | -96.7 |  |  |  |  |
| 22000 | -9.73 | -194.17 |  |  |  |  |
| 47000 | -32.07 | -285.67 |  |  |  |  |
| 100000 | -57.81 | -325.72 |  |  |  |  |
| 220000 | -83.92 | -343.94 |  |  |  |  |
| 470000 | -111.29 | -352.71 |  |  |  |  |
| 100 | -137.66 | -356.59 |  |  |  |  |

Unit 5: Closed Loop Sinusoidal Response

U5: Theory

To predict the behavior of the closed loop system, take the bode plot date collected in Table 4 and turn it into a Nyquist plot. Do this by using the following trigonometric identities to convert the polar coordinates in the Bode plot into Cartesian coordinates in Table 8.

* X coordinate =
* Y coordinate =

Table 8: The Bode plot data from Table 4 converted into Cartesian coordinates.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ω (rad/sec) |  |  | X coordinate | Y coordinate |
| 100 | 1 | -2.08 | 0.999236619 | -0.036283379 |
| 1000 | 0.99 | -20.78 | 0.925184836 | -0.351138209 |
| 2200 | 0.951 | -45.63 | 0.664863533 | -0.679539307 |
| 4700 | 0.794 | -96.7 | -0.09266317 | -0.788413321 |
| 10000 | 0.326 | -194.17 | -0.316120756 | 0.079828474 |
| 22000 | 0.025 | -285.67 | 0.006726568 | 0.023980867 |
| 47000 | 0.001 | -325.72 | 0.001063268 | 0.000724894 |
| 100000 | 0 | -343.94 | 6.11912E-05 | 1.7614E-05 |
| 220000 | 0 | -352.71 | 2.70443E-06 | 3.46142E-07 |
| 470000 | 0 | -356.59 | 1.30737E-07 | 7.79818E-09 |

Now plot the data from Table 8 into the X,Y graph below. Make sure to put arrows indicating the direction of increasing ω and draw both lobes of the Nyquist plot. From this Nyquist plot, determine the gain and phase margin.

U5: Simulation

Let’s make quick work of the theoretical calculations with MATLAB. To do this, we will use the MATLAB function nyquist to plot the Nyquist plot of a transfer function.

Now, create a script called **closedLoopFreqResponse.m** with the following text.

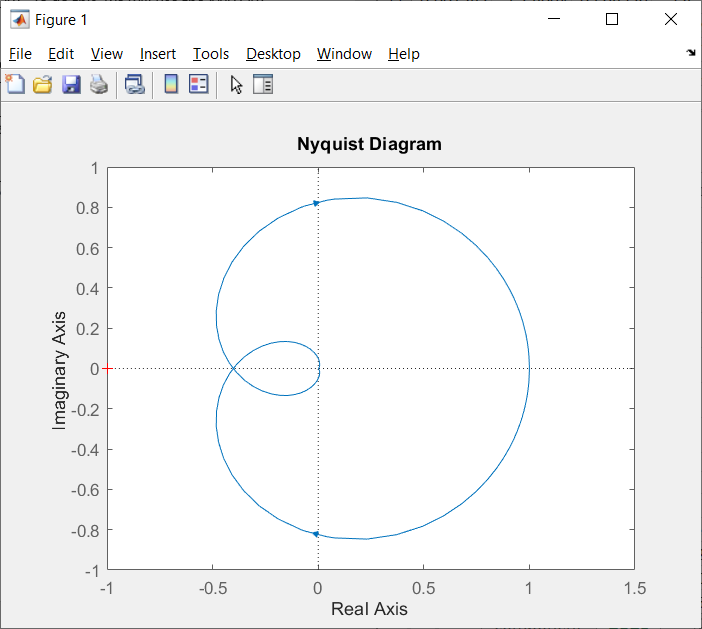
Listing 3: A MATLAB script to generate the Nyquist plot for G1(s)\*G2(s). This will provide us with information about the closed-loop behavior of G1(s)\*G2(s).

s=tf(‘s’);

G1 = a1/(s^2 + b1\*s + c1); // replace a1, b1 and c1 with correct values

G2 = a2/(s^2 + b2\*s + c2); // replace a2, b2 and c2 with correct values

nyquist(G1\*G2);



Unit 6: PID Control

U6: P-control

The results from unit 3 indicate that there is steady state error. No matter how much we increase the gain, we will have steady state error.

From the root loci we produced in a simulation in Unit 4, we see that a gain of roughly 2.5 will push the poles of the closed-loop system into the right half plane making the feedback system unstable. Let’s experiment with proportional control on the PID development board to see if we can replicate this unstable phenomena.

PID Development board configuration:

|  |  |
| --- | --- |
| Input select | STEP |
| Jumper wire | **STEP header**  **LPF\_IN header** |
| Jumper | **LPF\_OUT**  **LOAD\_OUT** |

Power Supply configuration:

|  |  |
| --- | --- |
| Ch 1 Voltage | 9V |
| Ch 1 Current | **0.1A** |
| Ch 2 Voltage | **9V** |
| Ch 2 Current | **0.1A** |

Wire channel 1 and 2 in series, creating a +9V, GND and -9V supply. Verify these voltages with a DMM prior to connecting anything to the PID development board.

Configure the oscilloscope as follows:

|  |  |
| --- | --- |
| Ch1 probe | SET header |
| Ch ground clip | **GND loop** |
| Ch 2 probe | **OUT header** |
| Horizontal scale | **200us** |
| Ch 1 scale | **2V** |
| Ch 2 scale | **2V** |
| Ch 1 offset | **Center reticule** |
| Ch 2 offset | **Center reticule** |
| Trigger mode | **Auto** |
| Tigger source | **Ch 1** |
| Trigger slope |  |
| Trigger level | **0V** |