ECE 414 Take Home Test

Josh Andrews

April 10, 2018

Contents

1 Introduction

In this report, the use of five different techniques to design a controller for a plant. First, root locus analysis will be used to generate a PID type controller, followed by the use Matlab pidTuner function. Then a PID type controller found by pidsearch will be discussed. Finally a unity feedback and two parameter linear algebraic design controller will be found for the plant. The plant is generated through the use of the *ece414planttf* function provided by Professor Hanselman [1]. The function takes a birthday, in this case June 10^{th}, and an uncertainty variable α that slightly modifies the plant. The range of α is 0-1 with 0.5 corresponding to the nominal plant. There are several constraints set that apply to all designs and variations and are listed in Table 1 below.

Table 1: System design constraints

Parameter	Value
Steady state error	0
Overshoot	<10%
Step Response	Correct direction
Settling time	minimize
Peak controller effort	minimize
Peak sensitivity	minimize
Phase margin	>45dB

The design process as well as the results for each technique are documented in their respective sections. The report is then concluded with a analysis of which technique-controller combination is best for this plant.

2 Root Locus

In this section root locus techniques are used to design a unity feedback PID type controller for the plant. This controller also has the added constraint that any zero of the controller must be at least 2 away from the real part of a plant pole. Since α will cause the plant poles to move slightly, the pole locations for all α must be known before a zero location can be chosen. The Matlab function rlocus was used to generate Figure 2, which includes ten plants where α ranged from 0 to 0.9 in 0.1 increments.

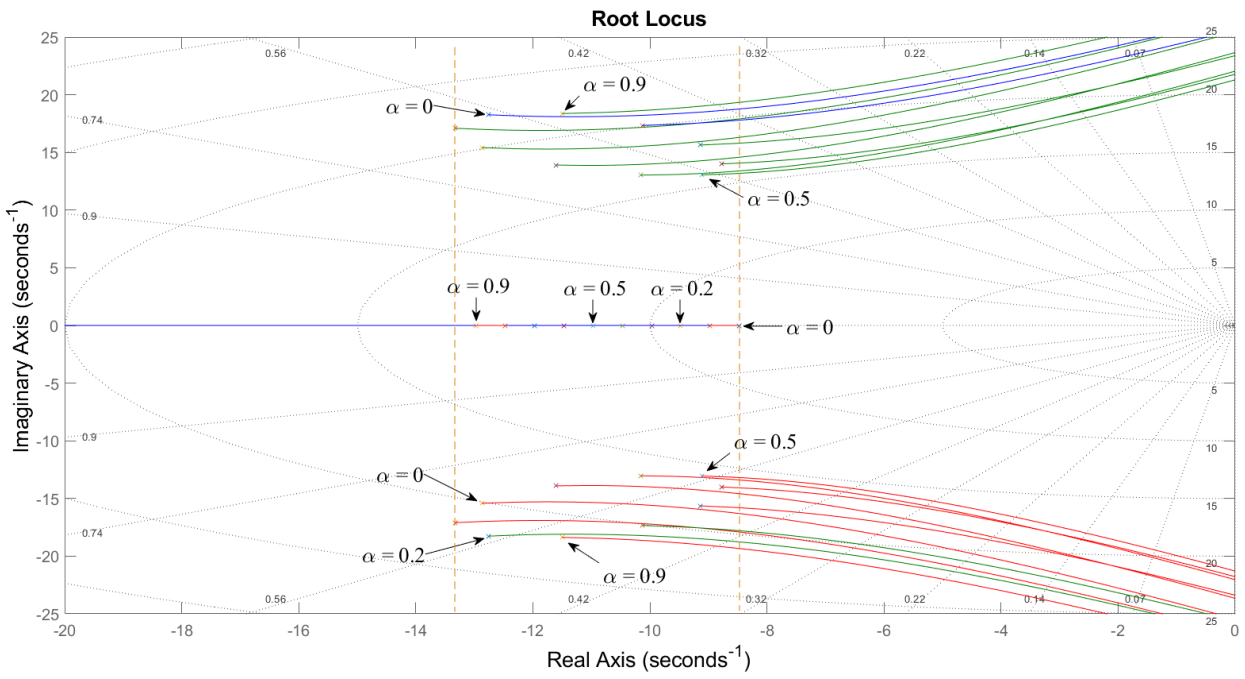

Figure 1: Root locus for plant with various α

As seen in the figure above the pole locations move as α is adjusted from 0 to 0.9. The pole in the top quadrant moves counter clockwise from α=0 while the one in the bottom quadrant moves clockwise. The pole on the real axis moves to the left as α is increased. The vertical dashed lines represent the minimum, -13.5, and maximum, -8.5, real values of all poles across all variations. Given the specification, any poles added must be at least two away from those lines.

The next step was to analyze the plant to see which controllers would maintain the open loop transfer function proper. The nominal plant transfer function is shown below.

$$G(\alpha = 0.5) = \frac{26.972}{(s + 10.97)(s^2 + 18.23s + 253.3)} \tag{1}$$

It can be immediately seen that there is no pole at the origin and therefore the steady state error (e_{ss}) will not be zero. A pole must be added at the origin which means integral control will be required.

It is also known that PD and PID controllers are not practically realizable due to the D term causing the controller effort transfer function (U(s)) to be improper. This means the controller would have to output, and the plant input be able to accept impulses, which is not possible without the use of a low pass filter. The resulting controllers would be classified either PDF or PIDF and will not be evaluated.

The PI controller has the form of

$$D(s) = \frac{k(s+z)}{s} \tag{2}$$

where the zero, z, will be determined by looking at Figure 2 above. Given the specification that settling time and overshoot should be minimized the zero should be far to the left of the imaginary axis. A large value for z will also increase the stability of the system. A balance will have to be found between the values for k and z to increase the overall speed while maintaining all the specifications for the system.

An initial controller was chosen by setting z to 20 and selecting a k value of 20 which should make it reasonably fast while hopefully maintaining ripple low. The controller's transfer function is shown below.

$$D(s) = \frac{20(s+20)}{s} \tag{3}$$

Testing the closed loop step response for the resulting system yielded the response shown in Figure **??** below.

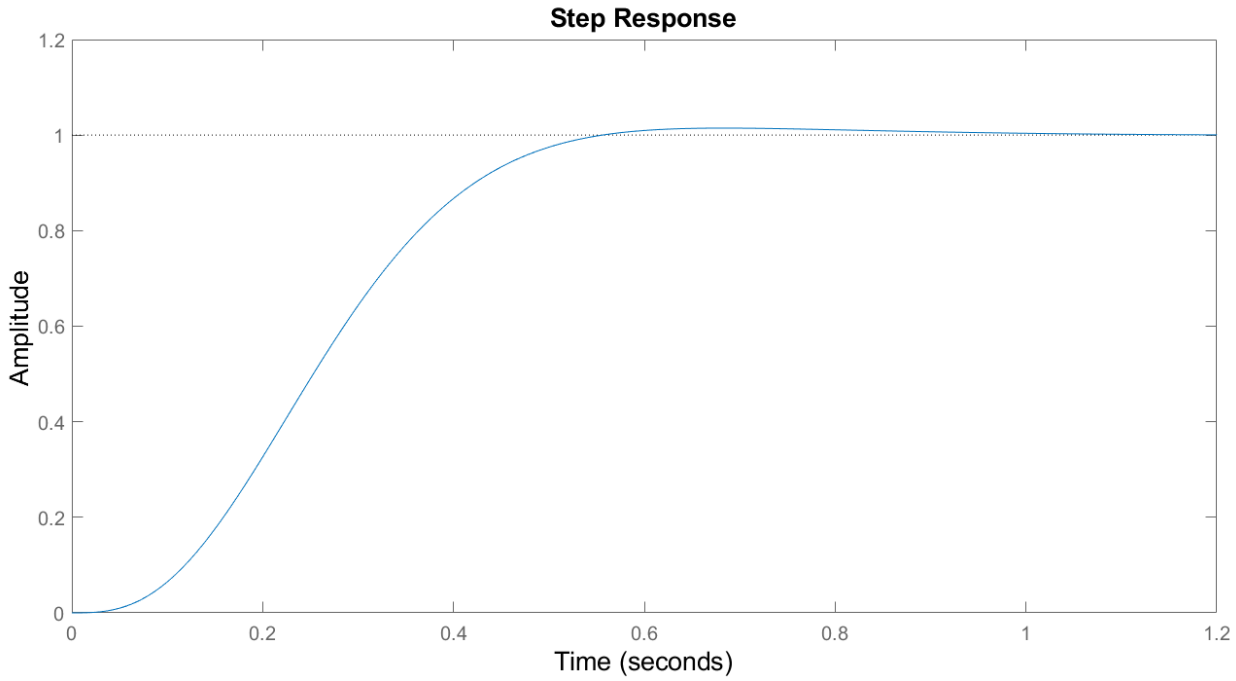

Figure 2: Step response for nominal system

As seen the settling time and overshoot are low using the designed controller with the nominal plant. The other parameters needed to be found include the phase margin (PM), the peak sensitivity, and the peak controller effort. In order to simplify the job of trying to minimize each parameter

individually, all specification parameters were gathered as α was varied from 0 to 1. The results were then plotted to see if there was a correlation between a singular α value and all the extrema of specification parameters and are shown in Figure 3.

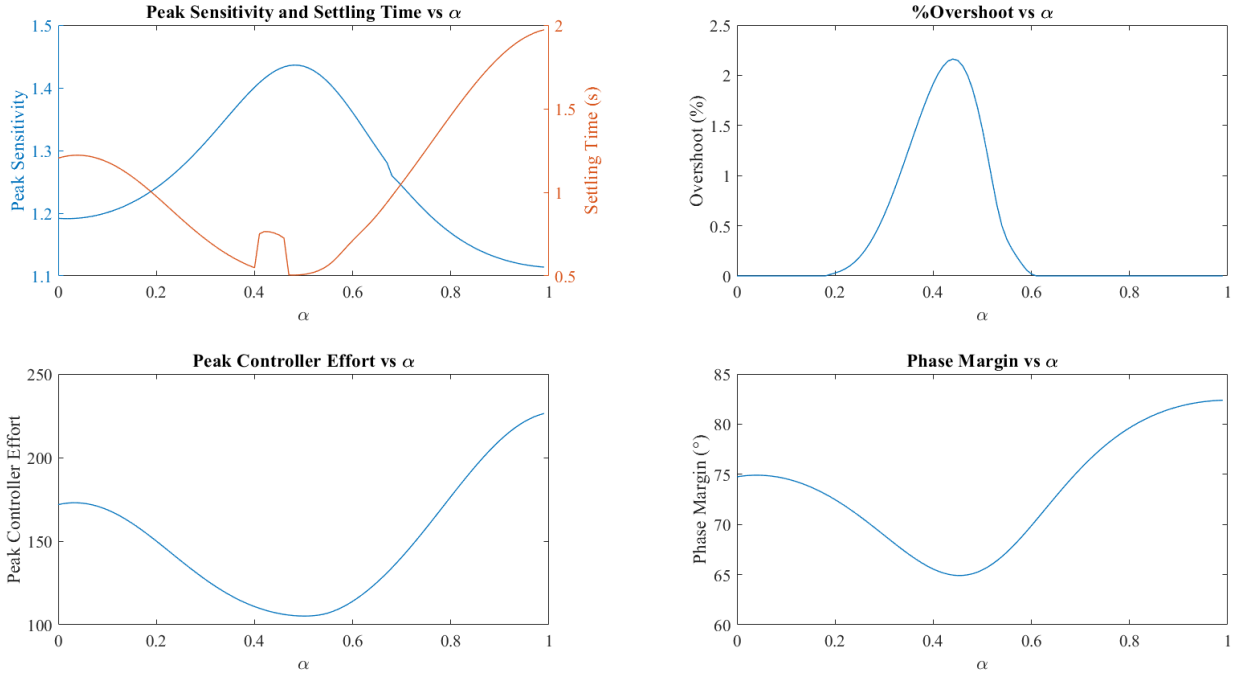

Figure 3: Nominal system parameters versus α

It appears as though there is indeed a correlation between the extrema and the α values of approximately 0.45 and 0.99. Overshoot and the peak sensitivity are at a max at α=0.45 and the phase margin, controller effort, and phase margin are at a minimum.

While this is good information to apply to the design, it must hold true for other controllers with this plant as well. Two more test were run, one with k unchanged and z increased by 10 and another with k increased by 10 and z unchanged from the nominal system. The same correlation was seen at α=0.45 for both controllers, with overshoot increasing and the settling time decreasing while becoming more stable across all plant variations. It was also noticed that the peak sensitivity increased, the phase margin decreased, and the controller effort did not change considerably for both variations. Finally the nominal controller, compared to the variations was better in terms of performance but settling time varied considerably.

With the successful correlation, the PI controller can now be tuned, using a plant with α=0.45. The main parameter that will be looked at is overshoot. If the overshoot is less than 10% at that point, then it will be less than 10% for all α. The PM will also be at a minimum and therefore must be greater than 45◦ along with the controller effort. The controller desired will allow overshoot to get close to 10% for the α=0.45 plant in order to minimize settling time, and will be tuned based on that.

The first step was to increase k and evaluate the performance. The chosen controller was

$$D(s) = \frac{25(s+20)}{s} \tag{4}$$

and testing began from there. Increasing both parameters at the same time cause overshoot to increase too drastically, and was most affected by the zero location. It was found that if k was increased slightly and z was decreased slightly, all specification parameters improved. The final controller found was

$$D(s) = \frac{28(s+18)}{s} \tag{5}$$

which provided the following a system step response shown in Figure 4.

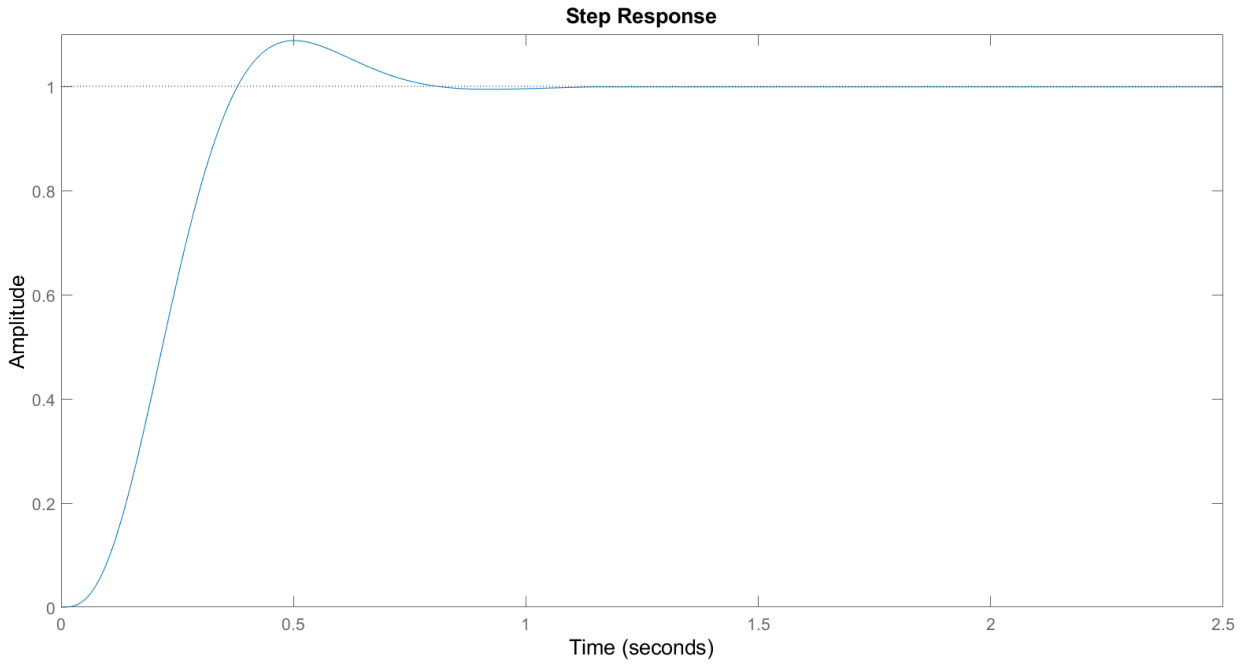

Figure 4: Step response for final system

The top of the graph equals 10% overshoot which is not exceeded. A final check of all parameters across all plant variations is needed to confirm all specifications are met. Figure 5 shows the final result.

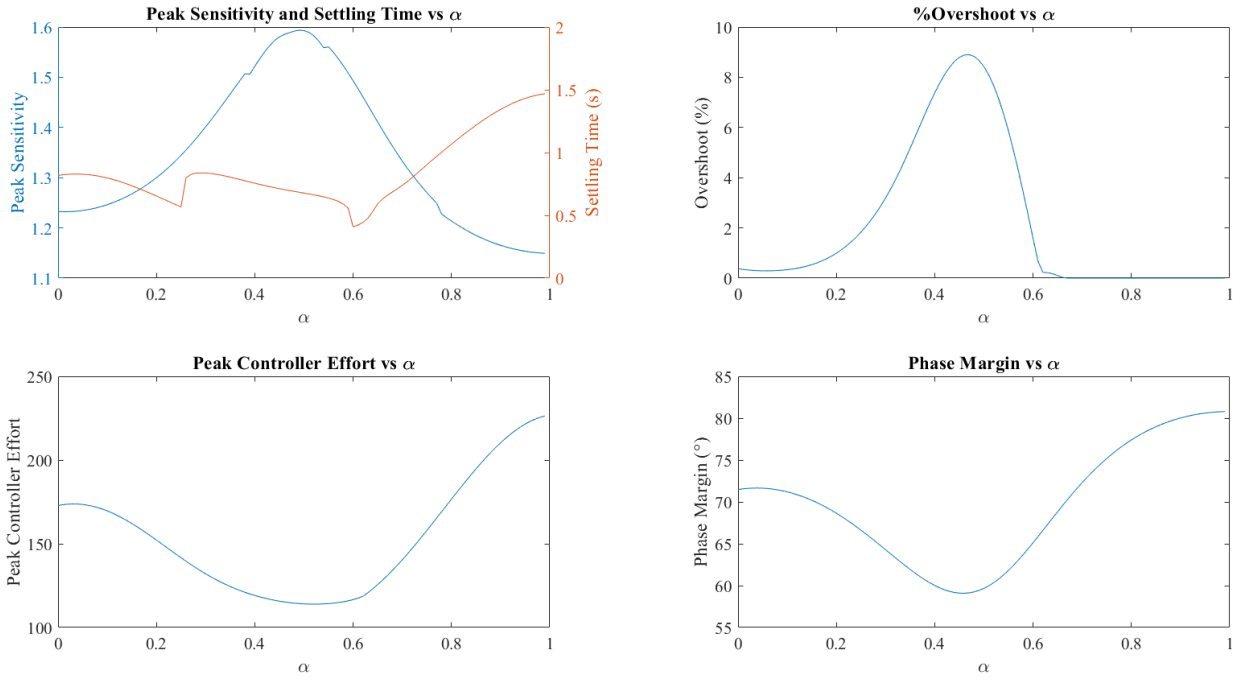

Figure 5: Final system parameters versus α

As seen all specifications are met for all variations of the plant. A summary of the results are listed in Table 2 below.

Table 2: Final system parameter extrema

Parameter	Low Value	High Value
Peak Sensitivity	1.149	1.593
Settling Time	0.4074 sec	1.468 sec
%Overshoot	0%	8.9%
Peak Controller Effort	114	226.4
Phase Margin	59.11	80.79

While the overshoot and peak sensitivity are slightly worse compared to the nominal controller, the settling time is much improved and stabilized. The final root locus for the system is shown in Figure 6.

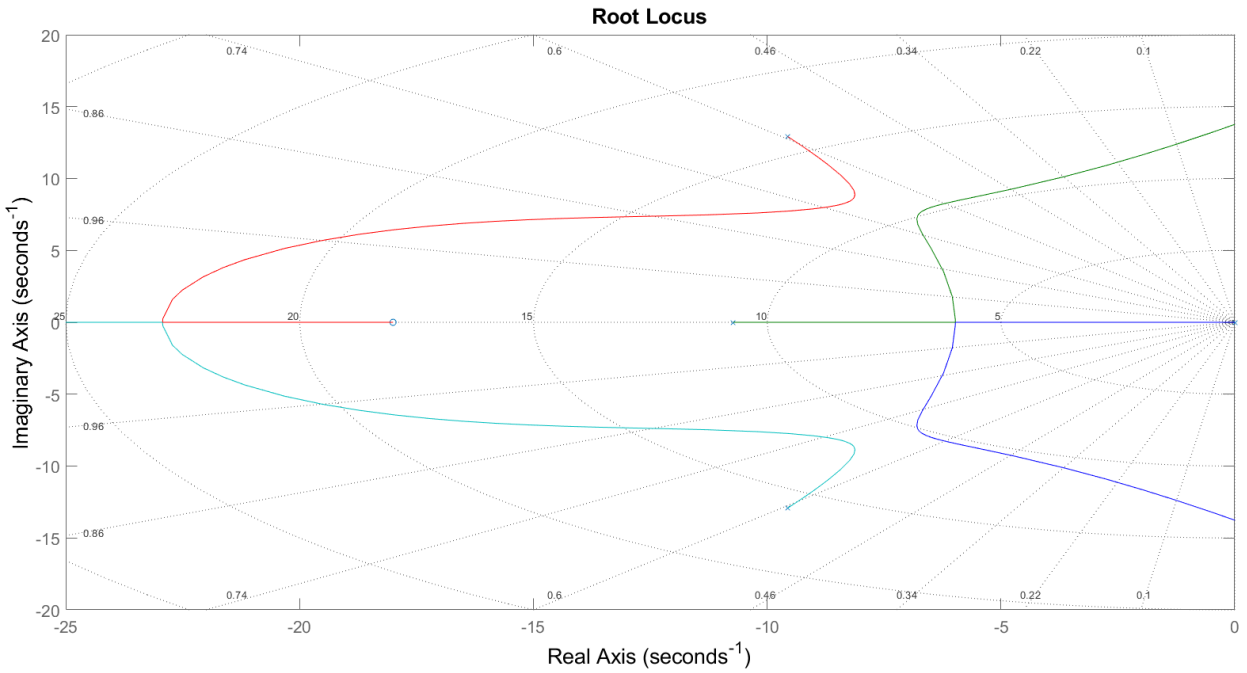

Figure 6: Final system root locus

Using root locus techniques to design a PID type controller is difficult and time consuming, and much easier methods exist to develop a controller. The design in this section focused on trying to minimize and stabilize the settling time in order to achieve a more uniform response given plant uncertainty. With a successful controller designed, the use of Matlab functions pidtune and pidTuner to design a PID type controller will be discussed.

3 pidtune and pidTuner

In this section pidtune is used to generate a baseline controller and then pidTuner is used to fine tune the controller to fit the specifications. The controller must be a unity feedback PID type controller. Again the nominal plant will be generated using an α of 0.45 and a PD or PID controller is not realizable. Inputting the PI controller generated using pidtune,

$$D(s) = \frac{85.8(s + 8.159)}{s}, \tag{6}$$

and the plant to the pidTuner function yields the GUI shown in Figure 7. A plot has been added to show the controller effort and the plant parameters are displayed which allows for easier design.

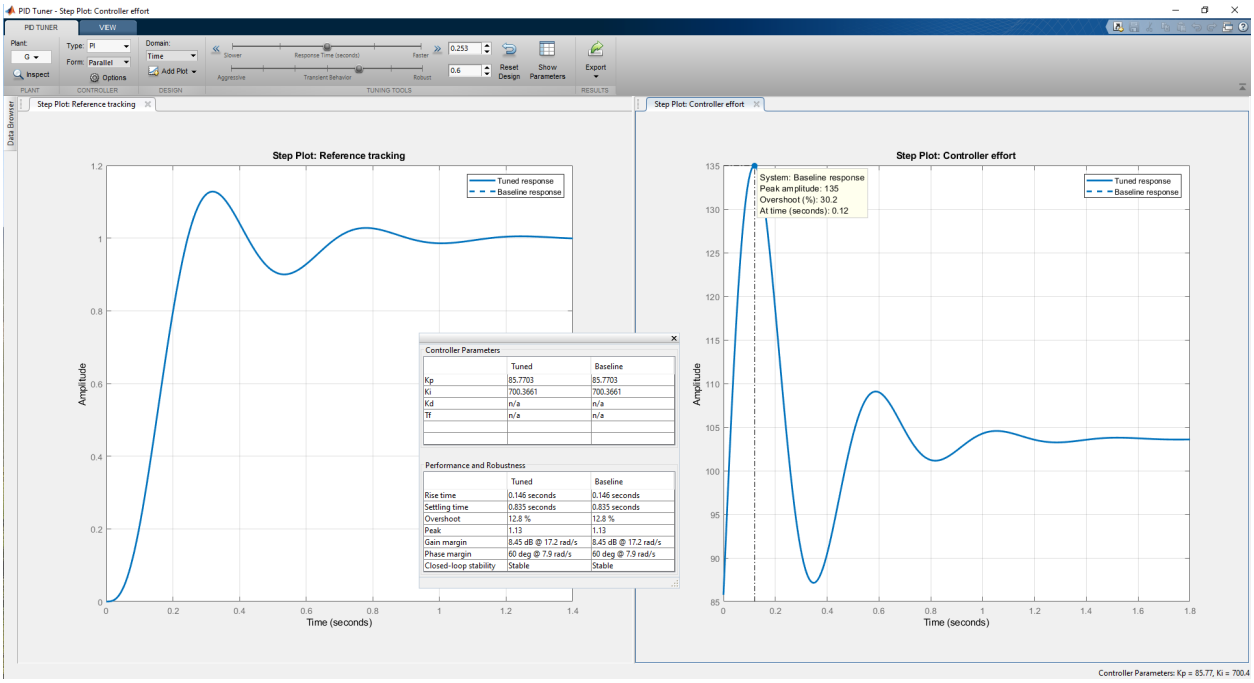

Figure 7: pidTuner graphic user interface

Using this method a controller can easily be designed to minimize the peak controller effort, settling time, and overshoot. The two sliders on the top of the GUI allow the controller to be tuned by adjusting the Kp and Ki controller parameters. The sliders were adjusted until the controller,

$$D(s) = \frac{37.9(s+18)}{s} \tag{7}$$

was found. Then the controller was tested with all plant variations and the results are shown in Figure 8.

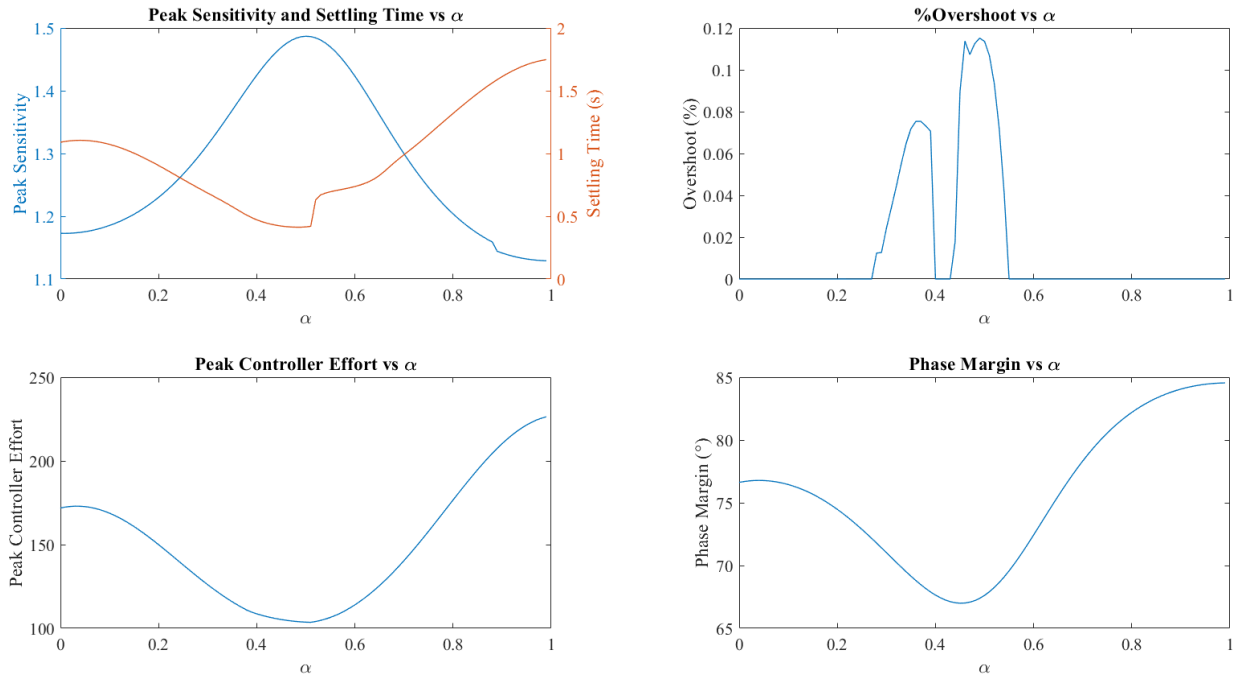

Figure 8: Final system parameters versus α

These results are much better than those achieved using root locus techniques and it was a much easier process. While the settling time varied more over the previous technique, all other parameters are improved. The results are summarized in Table 4 below.

Table 3: Final system parameter extrema

Parameter	Low Value	High Value
Peak Sensitivity	1.129	1.487
Settling Time	0.4132 sec	1.747 sec
%Overshoot	0%	0.12%
Peak Controller Effort	103.6	226.4
Phase Margin	67.01	84.55

With a successful PI controller achieved using pidTuner, a PID type controller designed using the Matlab function pidsearch will be discussed in the next section

4 pidsearch

The pidserach function allows a baseline controller to be optimized with the plant. The function allows, based on the specifications, optimization based on either overshoot or settling time. Since the settling time is at its maximum when α=0.99, that will be minimized and tested first. In order

to generate a baseline controller, the final controller generated from the previous section will be used. Then the pidsearch function was used and the resulting controller is

$$D(s) = \frac{94.028(s + 15.91)}{s} \tag{8}$$

which yielded the close loop step response shown in Figure 9.

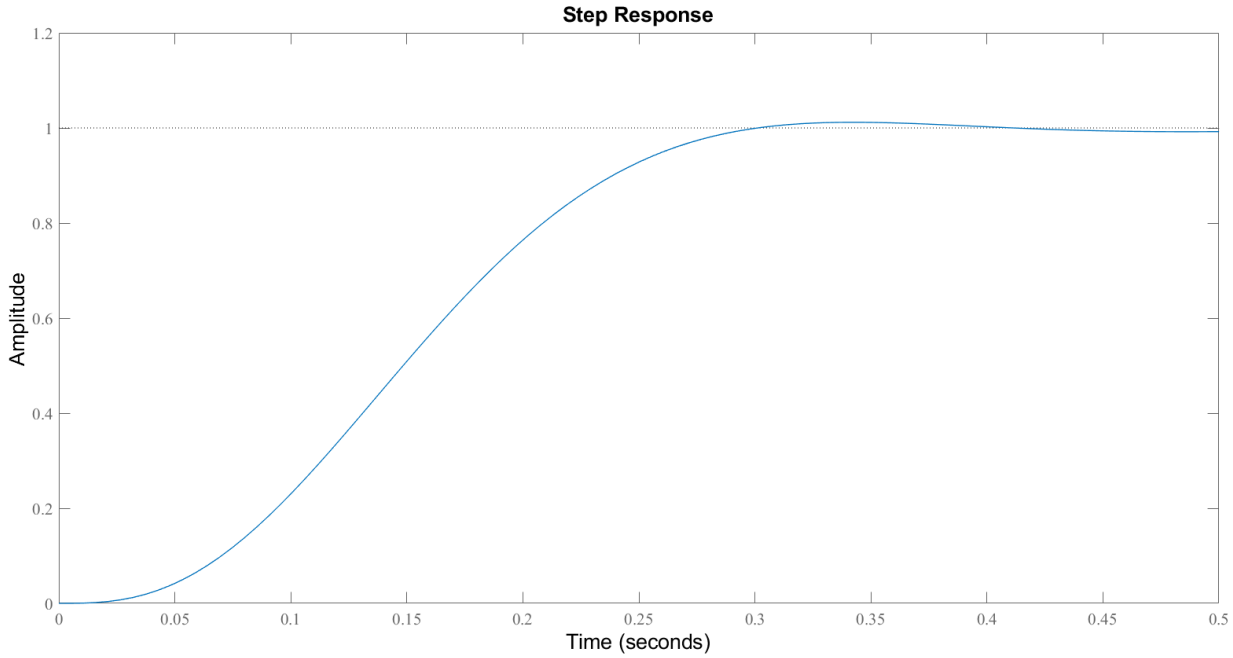

Figure 9: Step response with Ts minimized

The step response looks good and so all parameters will be tested across all variations. The results are shown in Figure 10

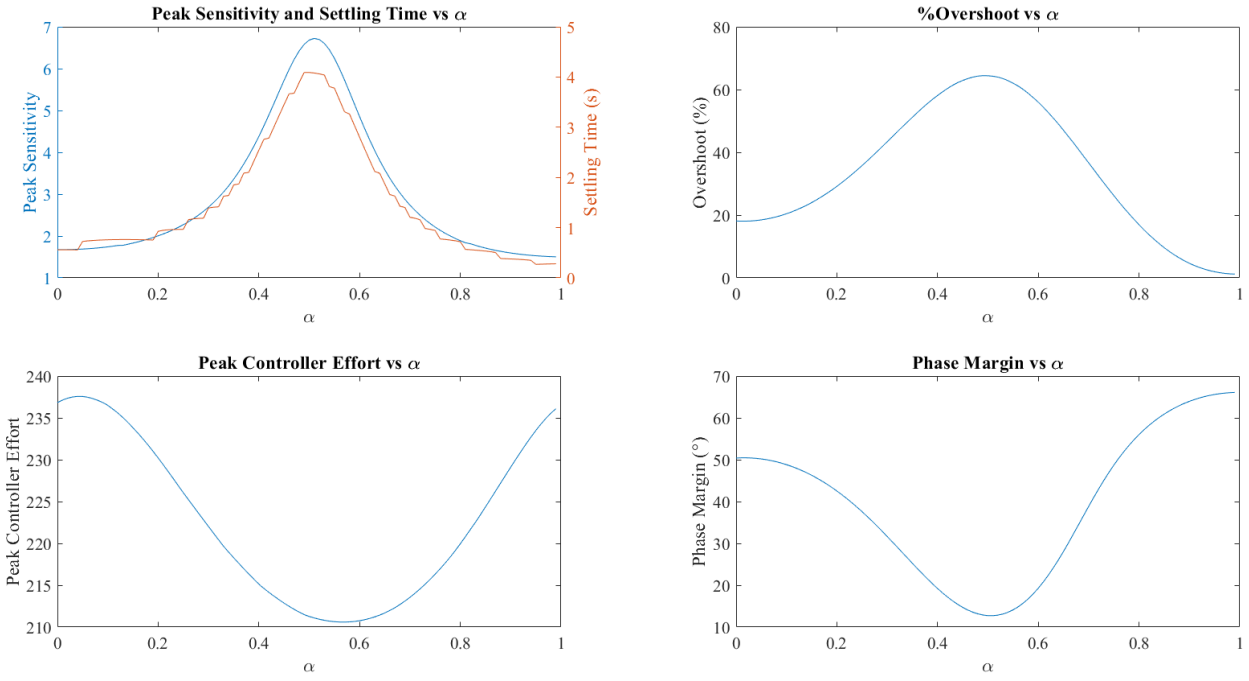

Figure 10: System parameters versus α with Ts minimized (α=0.99)

As seen the parameters vary wildly and way beyond specifications. So while minimizing Ts for α=0.99 did not work, the search was tried again, only with a α=0.45 plant. This resulted in a controller with the transfer function

$$D(s) = \frac{41.012(s + 12.2)}{s} \tag{9}$$

which is drastically different than the previous controller. The step response did not change much between the two however the plant variation performance was much improved and was very similar to the results from pidTuner. The results are shown in Figure 11 below.

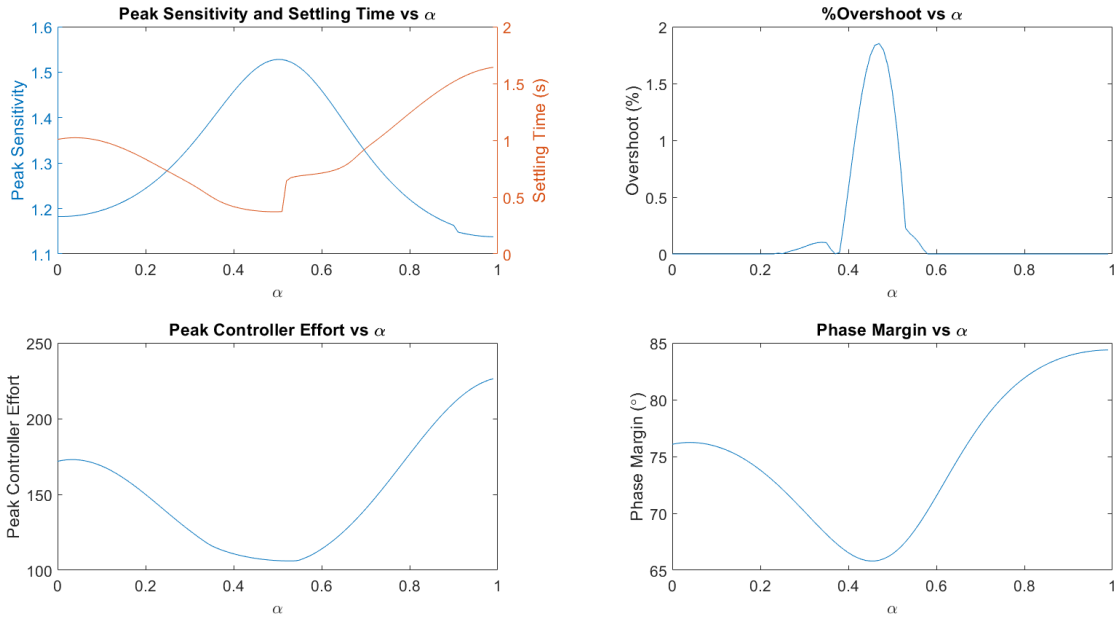

Figure 11: System parameters versus α with Ts minimized (α=0.45)

With a controller found by minimizing Ts, now pidsearch will be used to minimize overshoot. The pidTuner section final controller and a plant using α=0.45 will be used. Again the step response looked very similar to the previous ones and the results are shown in Figure 12.

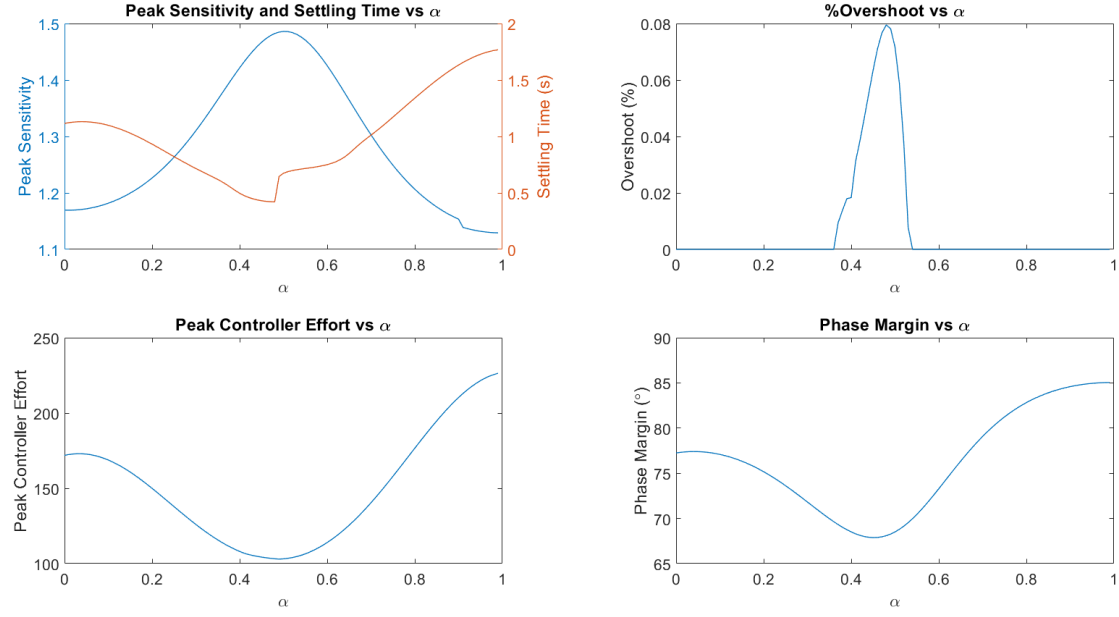

Figure 12: System parameters versus α with %OS minimized (α=0.45)

All parameters but the settling time were improved on by minimizing overshoot instead of minimizing settling time. Therefore the the final controller chosen is the one generated by minimizing overshoot and the transfer function is

$$D(s) = \frac{42.874(s + 11.25)}{s} \tag{10}$$

The final parameter extrema are shown in Table **??** below.

Table 4: Final system parameter extrema

Parameter	Low Value	High Value
Peak Sensitivity	1.127	1.485
Settling Time	0.452 sec	1.793 sec
%Overshoot	0%	0.08%
Peak Controller Effort	104	226.4
Phase Margin	67.86	84.74

This is a slight improvement over the previous design techniques and was also fairly easy to accomplish. Using pidTuner and pidsearch in conjunction is a very fast, easy, efficient, and effective way to design a controller. That statement is based on the quality of system performance and the time it took to get that quality system. With a successful controller designed using pidsearch, a unity feedback LAM controller will be designed in the next section.

5 Unity feedback LAM controller

In this section the lamdesign function is used to design a unity feedback LAM controller. This function accepts a plant and Do, which is the desired closed loop transfer function denominator polynomial. A nominal plant using α=0.5 will be used for optimization. The lamdesign function outputs the controller, controller effort, and open and closed loop transfer functions.

In order to find the desired value of Do, the method outlined in Chen's book on the linear algebraic method was used. This required finding Q(s) which is defined as

$$Q(s) = D(s)D(-s) + qN(s)N(-s) \tag{11}$$

where D(s) is the denominator polynomial of the plant and N(s) is the numerator. The q value is chosen arbitrarily based on the fact that increasing q will increase the controller effort but also make the system faster. An initial value of 100 was chosen for q but may be adjusted later will minimal effort.

Once Q(s) is found, Do can be extracted by finding the negative real part roots of Q(s) and then

using poly to insert them into a polynomial vector. This approach however did not work with the lamdesign function as it required at least 2 more than the 3 valid poles found through Q(s). To counteract this, each pole was added twice. This provided two poles at each of the 3 locations, enough for the lamdesign function and enough to make it a type 1 system.

The plant and the new Do were input into the lamdesign function and the resulting closed loop, and controller effort step response, are shown in Figure 13.

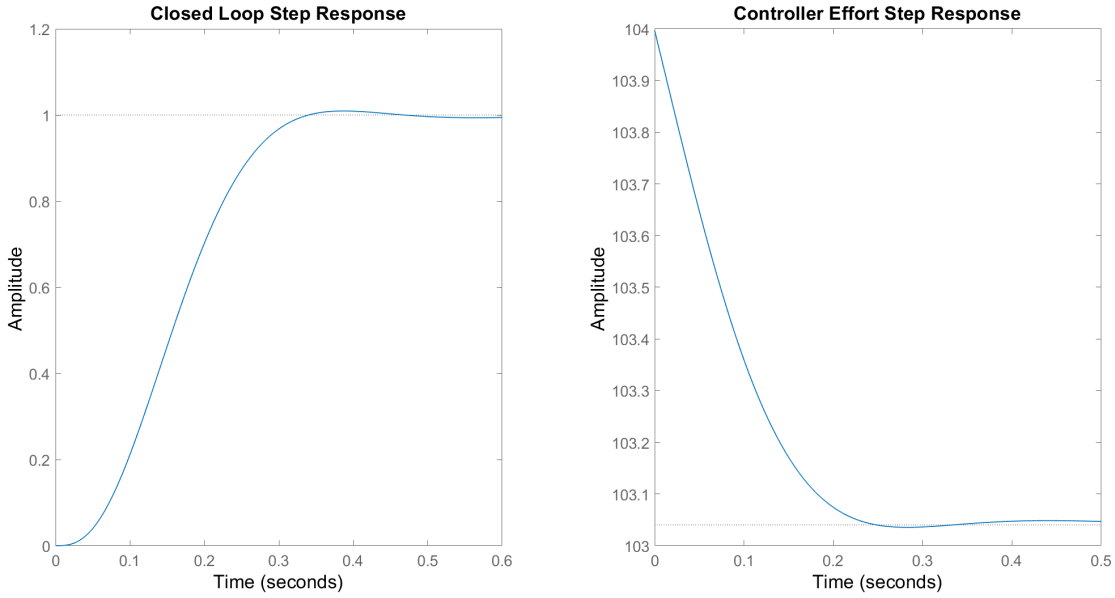

Figure 13: Closed loop step response for q=100

The settling time, %Overshoot, and peak controller effort are very low compared to the previous design methods. Now other values of q need to be check to see how the resulting controller will affect the system. Trying various values for q from 1 to 200 showed that overshoot, settling time, and peak controller effort did not change much. A lower value for q resulted in a lower peak controller effort but increased the settling time slightly and over shoot slightly.

In order to balance the settling time and controller effort, as overshoot was already way below the specification, the initial value of 100 for q was deemed best. It kept settling time low and kept controller effort below 104.

The final controller had the form

$$D(s) = \frac{104(s + 10.97)(s^2 + 18.23s + 253.3)}{s(s^2 + 29.27s + 455.4)} \tag{12}$$

Now that a controller was found, it had to be tested across all plant variations. Figure 14 shows the system parameters versus α.

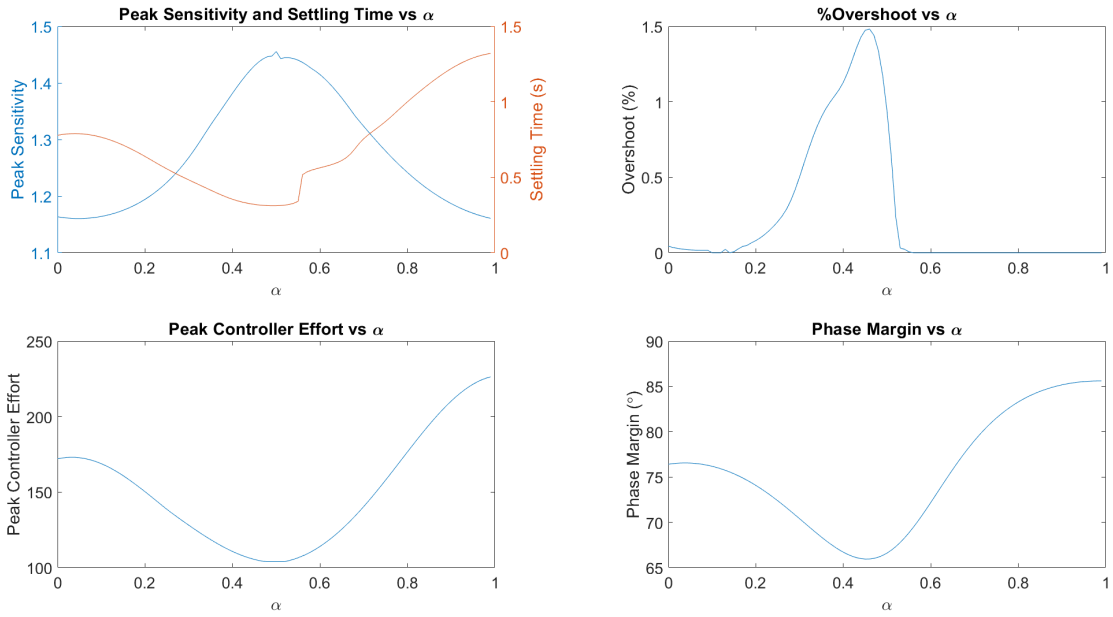

Figure 14: System parameters versus α

The results using this design are very similar to those in Figure 11 and have been placed in Table 5 below.

Table 5: Final system parameter extrema

Parameter	Low Value	High Value
Peak Sensitivity	1.16	1.455
Settling Time	0.3112 sec	1.32 sec
%Overshoot	0%	1.482%
Peak Controller Effort	104	226.4
Phase Margin	65.95	85.61

These results are better than all of the previous systems and it still took less time to design than using the root locus techniques. The resulting controller is more complicated but the means to formulate the design was fairly easy. Matlab takes care of all the hard linear algebra allowing the designer to only formulate a few equations. This is the best controller so far and will be attempted to be improved upon by utilizing a two parameter LAM design as discussed in the next section.

6 Two parameter LAM controller

In this section the lamdesign function is used to design a two parameter LAM controller. This is the same function as the previous section however 3 input arguments are needed for the two

parameter design. This function accepts a plant and T, which is the desired closed loop transfer function. It also accepts a vector of real negative poles that will be arbitrarily chosen if needed.

Since a final closed loop transfer function is needed for the system, ones using previous controller's transfer functions and the nominal plant mentioned were used. The unity LAM and pidsearch transfer functions will be used to find coefficients.

The first controller tested was the unity feedback LAM controller. The closed loop transfer function for that system was

$$T(s) = \frac{2805(s + 10.97)(s^2 + 18.23s + 253.3)}{(s + 11)^2(s^2 + 18.23s + 253.7)^2} \tag{13}$$

This caused an error in the lamdesign function declaring T and G were incompatible. The large number of poles can not be compensated for and thus no solution exists.

The next one tried was the pidsearch closed loop transfer function. The transfer function is defined as

$$T(s) = \frac{1145.7(s + 11.25)}{(s + 9.124)(s + 8.24)(s^2 + 12.48s + 171.4)} \tag{14}$$

Trying this with lamdesign was more successful however it now required a pole to be added. A default pole of -15 was chosen arbitrarily.

This resulted in controller functions of

$$F(s) = \frac{42.477(s + 15)(s + 11.25)}{(s + 15.58)(s + 0.05649)} \tag{15}$$

for the feedforward controller, and

$$H(s) = \frac{35.949(s + 17.12)(s + 11.5)}{(s + 15.58)(s + 0.05649)} \tag{16}$$

for the feedback controller. The controller effort ended up the same and no other parameters were tested as it should also be the same. With all design techniques tested, the report is concluded in the next section.

7 Conclusion

Five different design techniques for controllers were tested and analyzed. Most controllers were fairly similar in terms of the results they provided however the root locus and LAM designs were significantly more complicated and harder to understand. The unity feedback LAM controller proved to be the best performing out of all the controllers but it also had the most number of poles and zeros. It was found that using pidTuner and then optimizing further through pidsearch was the easiest technique and provided the best results of the PID type controllers. The controllers just mentioned were able to minimize and limit the swing of settling time better while also having a higher PM, and a lower peak sensitivity, overshoot, and peak controller effort.