1. **Introduction**

This report has two main objectives. The first objective is to design five proper controllers with each controller using different design techniques. The first controller uses the root locus technique. The second controller is a PID type controller that uses either *pidtune* or *pidTuner* functions in MATLAB. The third controller is a PID type controller designed using the *pidsearch* function. The fourth controller uses the unity feedback linear algebra design technique. Lastly, the fifth controller uses the two-parameter linear algebraic design technique. The second objective is to choose the best fit proper controller that archive all listed design goals in Table 1.

Table 1: proper controllers design goals

|  |  |
| --- | --- |
| **Parameter** | **Design Goals** |
| Unit step input | Zero percent steady state error |
| Overshoot | Less than ten percent |
| Step response | may not go in the wrong direction first |
| Phase margin | greater than forty-five degrees |
| Settling time | Minimize |
| Peak control effort | Minimize |
| Peak sensitivity | Minimize |

The design approach for each of the five proper controllers is described below. In each section of the proper controllers, the criteria that met and failed the design goals are described. The results of the best fit controller are shown in the design evaluation section for proper controllers that met design goals.

1. **Finding the Nominal Plant**

In this section, the process of searching for the nominal plant is described. The nominal plant is the plant that can be used as a reference to describe the system characteristics. A provided function called *ece414planttf.p* was usedto base the system controller design on. The function returns 100 random variations of the plan G(s) by variate alpha value for a given day and month. The date passed to the function is December 28th, and the alpha range passed was 1:100.

\*In this design, 28th used for the date and 12 choose for the month. The alpha is varied from 1 to 100.\* The process of searching for plan G(s) started by passing the output results output zero’s of the G(s) results function provided. The method of searching was established by finding the plants that are close to 100 plants variation. While the gain of the nominal plant will be the average gain of the 100 plants variation. The location of the nominal plant can be seen in figure 1.

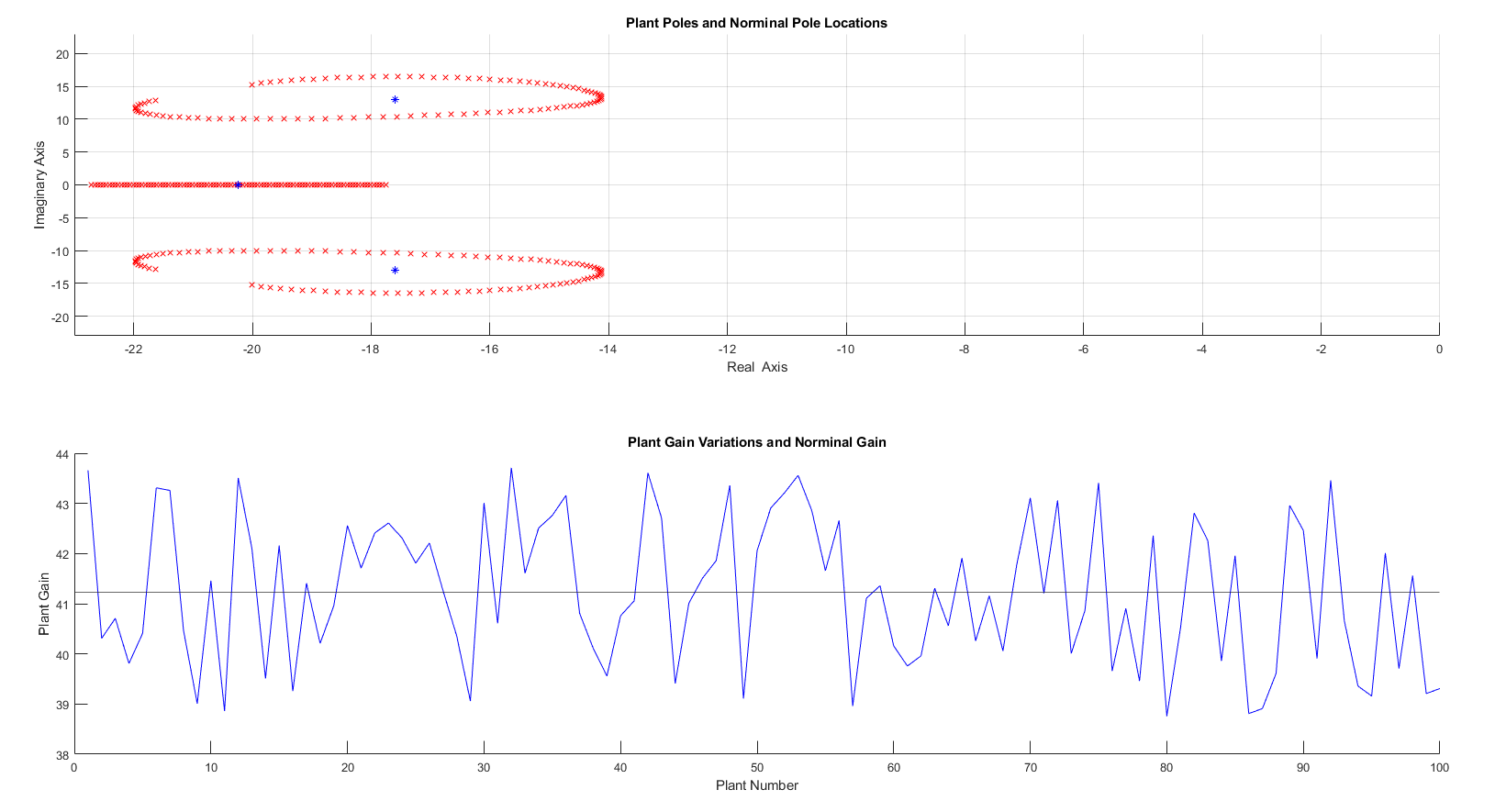


Figure 1: Nominal Plant location

The result of the gain found by taking the average of all gain plants representing the gain of the nominal plant. This can be seen in the Figure 1 as well. By using the found average zero’s location of the plant and using the average gain of all possible random variation. The nominal plant is found to be as the following:

– (1)

The equation (1) used as reference plant G(s) used to build a robust control system.

1. **PID Controller design with Root Locus Technique**

In this section, we are designing a PID type (e.g., P, PD, PI, or PID) using Root locus technique. There are multiple variations of PID type. However, there is only one case out of the listed types that is possible to implement. A ‘P’ type or also known as a proportional Control is completely governed by the G(s) plane. It will not be possible to achieve a zero steady state error which is one of the design goals.+++ By starting to analyze other PID type such as ‘D’ type base (PD and PID). It is important to see the ‘D’ term effect on the controller. When the number of zeros are greater than the number of poles, it implies that the effect is measured before the cause occurred. This is the case with an ideal PD or PID for ‘D’ part of the controller. So, it is not physically realizable. To make it physically realizable, we must introduce a derivative filter in its denominator, thus making the number of zeros equal to number of poles. Such system called PD + Filter (PDF) and PID + Filter (PIDF). For PDF controller, it expresses as follows:

Where the value of the Z, P and KD is the terms that defined the PDF controller. The control plant of the PIDF expresses as follows:

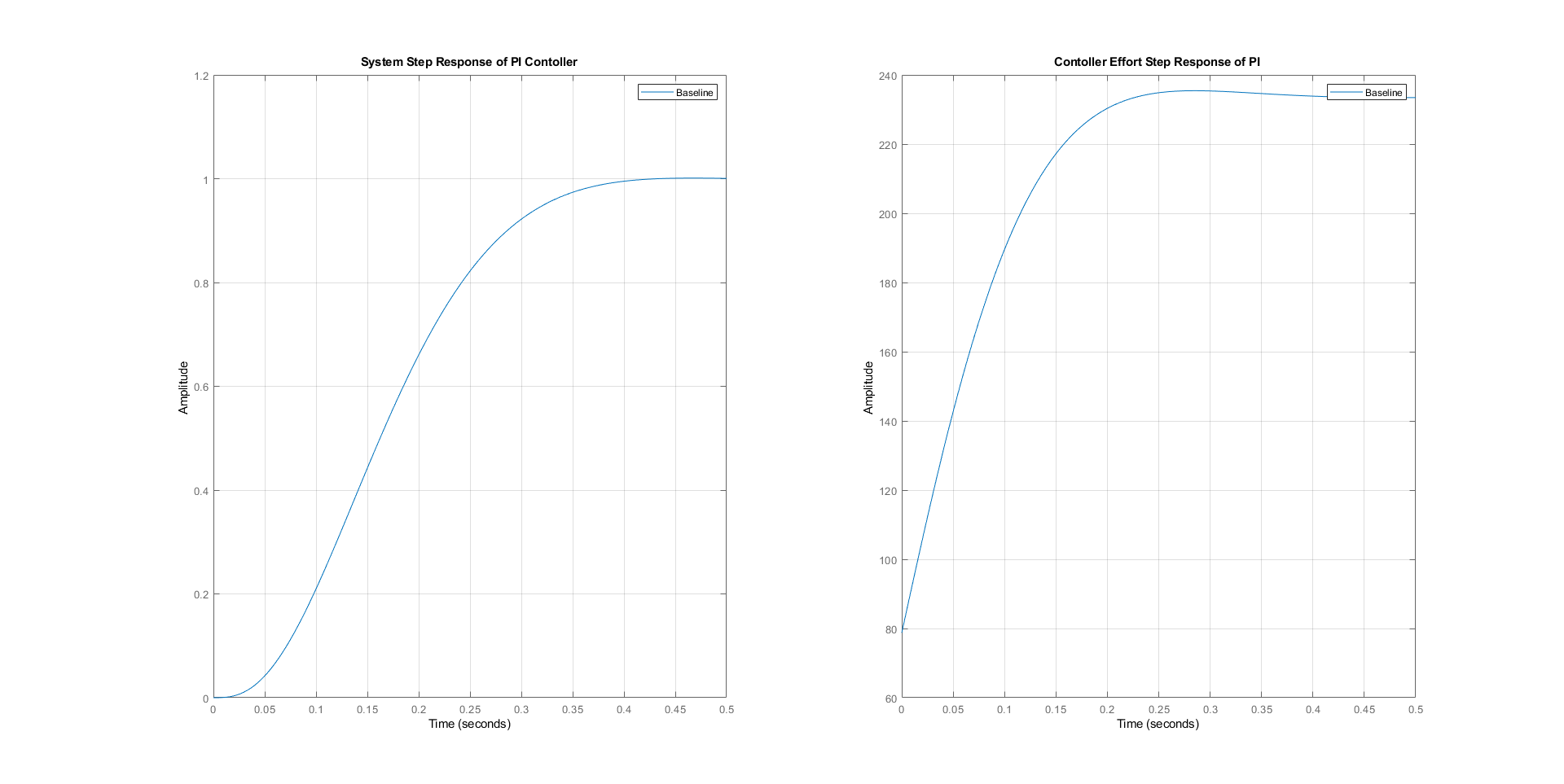
Where the terms to control PIDF will be KD, Z1, P, and Z2. By investigate PDF and PIDF controller plant, it requires three and four term respectively to control the nominal plant. It will be difficult task to achieve a balance controller rallying only on root locus. Also, PDF controller will base on plant G(s) even with filter. It will not be possible to reach zero steady state error. This leave us with the last PID type left as option which is ‘PI’. A PI controller can be described as follows:

It has two terms unlike the PDF and PIDF. By choosing Z term and vary KP, the PI controller can be calibrated to meet the design goals. There are only two terms that need to achieve the system design goal.

By tweak the KP and Z term’s, A value of KP and Z. The method was by find the upper and lower of KP and Z term’s. It found to meet all specification design goal criteria at Kp equal 78.7 and Ki term equal to 1360. The root locus found expression was:

–(5)

Figure 2 shown the step response and control effort of PI controller.



In the Table 2, a list of controller information.

Table 3: Systems info for PI

|  |  |
| --- | --- |
| **Info type** | **PI Controller** |
| Rise Time | 0.2148 |
| Settling Time | 0.3602 |
| Settling Min | 0.9016 |
| Settling Max | 1.0011 |
| Overshoot | 0.1141 |
| Undershoot | 0 |
| Peak | 1.0011 |
| Peak Time | 0.4661 |
| Umax | 235.4305 |
| EssStep | 0 |
| EssRamp | 0.1720 |
| Gain margin in dB (Gm) | 15.9155 |
| Phase margin in degrees (Pm) | 68.0027 |
| Gain crossover frequency in rad/s (Wcg) | 23.2454 |
| Phase crossover frequency in rad/s (Wcp) | 5.7605 |
| Vector margin (Vm) | 0.7271 |
| Vector margin frequency (Wvm) | 14.8168 |
| Peak sensitivity (Smax) | 1.3754 |

1. **PID Controller design with *pidTune* and *pidTuner***

In this section, *pidTune* and *pidTuner* GUI interface used to calibrate PID type controller manually. Only PI and PIDF type controller considered as proper candidate. The reasoning of already established in the toot locus technique section. In Table 3, the terms KP, KD and KI shown.

Table 2: Systems info for PI and PIDF

|  |  |  |
| --- | --- | --- |
| PID Type Terms | PI Controller | PIDF Controller |
| KP | 110.2561 | 470.4864 |
| KI | 1.5274e+03 | 3.9427e+03 |
| KD | - | 13.9307 |

While in Table 4, the parameters associate with step response of PI, PDF and PIDF controller shown was listed.

Table 3: Systems info for PI and PIDF

|  |  |  |
| --- | --- | --- |
| **Info type** | **PI Controller** | **PIDF Controller** |
| Rise Time | 0.1820 | 0.0721 |
| Settling Time | 0.3115 | 0.3158 |
| Settling Min | 0.9022 | 0.9247 |
| Settling Max | 0.9996 | 1.0483 |
| Overshoot | 0 | 4.8280 |
| Undershoot | 0 | 0 |
| Peak | 0.9996 | 1.0483 |
| Peak Time | 0.8720 | 0.1398 |
| Umax | 237.0692 | 3.1841e+04 |
| EssStep | 0 | 0 |
| EssRamp | 0.1533 | 0.0594 |
| Gain margin in dB (Gm) | 14.9450 | 38.6952 |
| Phase margin in degrees (Pm) | 69.0015 | 62.0023 |
| Gain crossover frequency in rad/s (Wcg) | 25.1524 | 223.2869 |
| Phase crossover frequency in rad/s (Wcp) | 6.6663 | 19.7035 |
| Vector margin (Vm) | 0.7052 | 0.6990 |
| Vector margin frequency (Wvm) | 16.3174 | 32.6687 |
| Peak sensitivity (Smax) | 1.4181 | 1.4306 |

After evaluating the step response of the two-systems using MATLAB. Plot of the step response of the PI and PIDF system shown in Figure 3. The controller plans of PI controller can be express as follows:

While the contorl plant expression for PIDF was as follows:

After evaluating the step response of the two-systems using MATLAB. Plot of the step response of the PI and PIDF system shown in Figure 3.

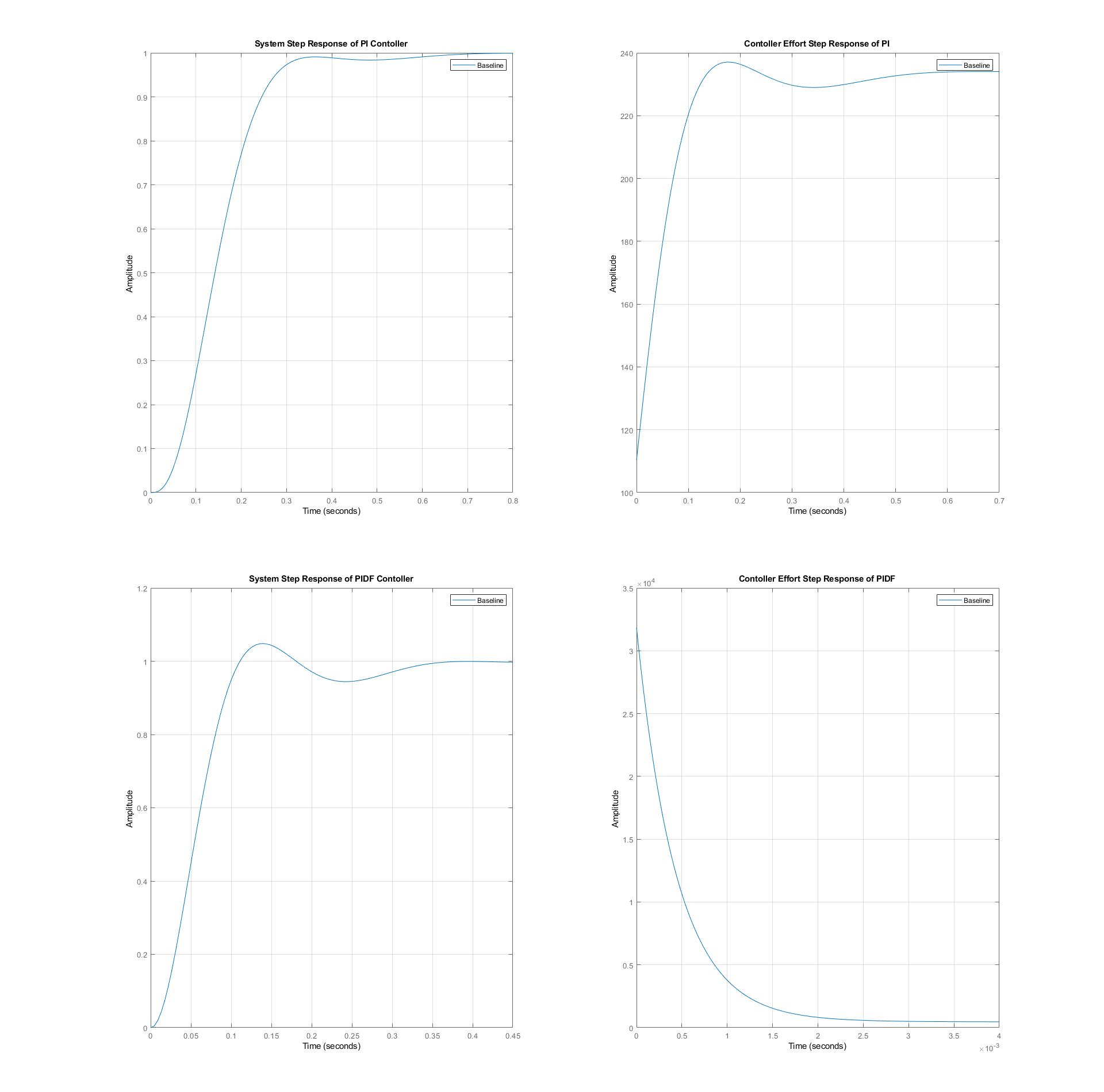


Figure 3: The step response and controller effort for PI and PIDF type Contoller using *pidTuner*

By investigate the listed results from table 3, both PI and PIDF met the design goal. There are three design goal where there one option will prefer over the other. If we look at settling time, PIDF had the least amount of the settling timing. While peak control effort and sensitivity, the PI controller had the least. This make match more design goal over the PIDF. In this design technique, the PI controller will better candidate as best control option.

1. **PID Controller design with *pidsearch***

In this section, *pidsearch used to tone* a *pidTune* control plant. It helps to optimize the baseline of PID controller to control the following certain criteria:

* The integral of time and the absolute error.
* The integral of the square
* The integral of the squared error.
* The integral of the squared error starting at the Rise Time Tr.
* The step response overshoot.
* The step response settling time.
* The product of the overshoot and the settling time.
* The product of the initial control effort U and the settling time.

By looking at the listed available option for tuning, only one criteria limited numerically which is the overshoot. While other prefer to be as low as possible. Therefore, the pidshearch set to be based on the step response overshoot. A function called specs\_table written. It uses *pidTuner* and *pidsearch* to create a excel file that include all controller info in a table. It received the plant G(s) which it will base the design decision on the output result of look up table. It prints out the system that had control effort that is not infinity and zero steady state error. There was only two system found to be matched these two conditions. The two control plants are a PI and PIDF controller.

Table 3: Systems info for PI and PID

|  |  |  |
| --- | --- | --- |
| **Info type** | **PI Controller** | **PIDF Controller** |
| Rise Time | 0.1088 | 0.0710 |
| Settling Time | 0.4908 | 0.3156 |
| Settling Min | 0.9120 | 0.9231 |
| Settling Max | 1.0976 | 1.0672 |
| Overshoot | 9.7624 | 6.7243 |
| Undershoot | 0 | 0 |
| Peak | 1.0976 | 1.0672 |
| Peak Time | 0.2309 | 0.1403 |
| Umax | 310.6495 | 3.0872e+04 |
| EssStep | 0 | 0 |
| EssRamp | -0.1042 | -0.0571 |
| Gain margin in dB (Gm) | 11.1148 | 38.4646 |
| Phase margin in degrees (Pm) | 60 | 60.0003 |
| Gain crossover frequency in rad/s (Wcg) | 27.1423 | 217.0343 |
| Phase crossover frequency in rad/s (Wcp) | 10.9777 | 19.7044 |
| Vector margin (Vm) | 0.5840 | 0.6884 |
| Vector margin frequency (Wvm) | 18.7948 | 31.8196 |
| Peak sensitivity (Smax) | 1.7124 | 1.4527 |

The two evaluated proper system matched met all design goals that listed in Table 1. The control plant expression for PI and PIDF after tune by *pidsearch* function shown in equation 7 and 8.

The close loop step response and control effort of the DPI and DPIDF can be seen for nominal plan in Figure.

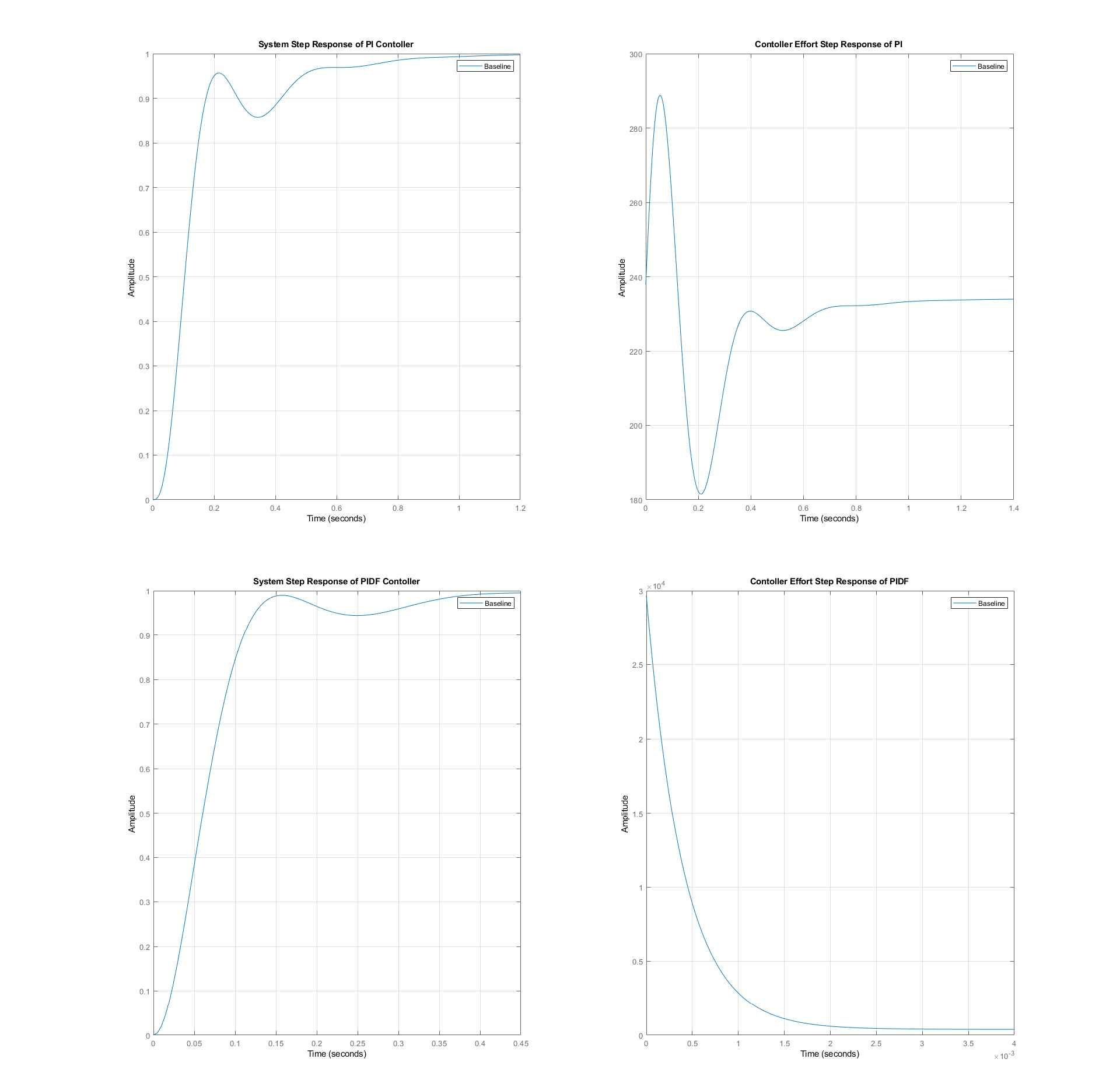


Figure 3: The step response and controller effort for PI and PIDF type Contoller using *pidsearch*

By investigating the collecting result in table 3 and by look at both system response in Figure 3, we can see similar number from *pidTuner* technique. However, this time PIDF controller achieve more goal than PI controller. So, let see what different here, a PI controller have less control effort in general in both cases. However, if we look at button of the Table 3, sensitivity of PI increase over the PIDF. While PIDF controller still maintain the lead over rise time and settling time.

1. **Controller with a Unity Feedback Linear Algebra design**

In this section, the *lamdesign* function used to generate a unity linear algebra system. For design process both *stepitae* and *stepshape* used to generate D0(s). In this case, for any set of specification there will ultimate choose. However, we are still limited in term of design of the controller sensitivity to plant G(S). Where the outcome to achieve the minimum amount of sensitivity. We are using *stepitae* and *stepshape* to generate desired pole for given step response specification.

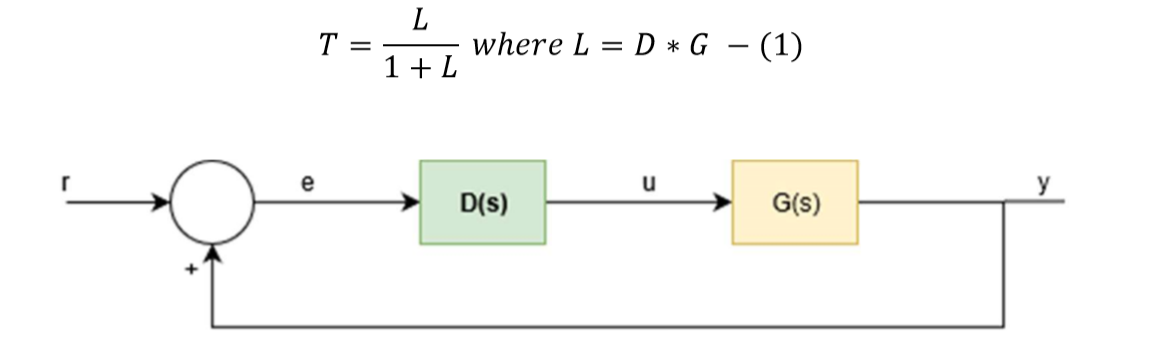


Figure 4: Block diagram of closed-loop of control for Unity Feedback

By pass the plant G(s) to *specs\_table function* which is written function that call *stepitae* and *stepshape* to create table in excel for possible configuration using both methods. For *stepitae, it* returns a 6th order system having zero overshoot and a ten second settling time to within 1%. The same done with *stepshape*. However, because one of the design goals to achieve the minimize the terms such as control effort and sensitivity peak. A unity system does not meet deign goal as seen in Table 4.

Table 4: Systems info of a Unity Feedback Linear Algebra design using *stepitae* and *stepshape*

|  |  |  |
| --- | --- | --- |
| **Info type** | **Unity LAM with *stepitae*** | **Unity LAM with *stepshape*** |
| Rise Time | 6.5743e-07 | 5.0210e-07 |
| Settling Time | 11.7345 | 14.1853 |
| Settling Min | -1.6168e+05 | -2.2207e+05 |
| Settling Max | 4.8818e+05 | 8.8426e+05 |
| Overshoot | 4.8818e+07 | 8.8426e+07 |
| Undershoot | 3.9955e+08 | 8.1980e+08 |
| Peak | 3.9955e+06 | 8.1980e+06 |
| Peak Time | 3.7388 | 4.8890 |
| Umax | 9.3672e+08 | 1.9201e+09 |
| EssStep | 0 | 0 |
| EssRamp | -1.0725e+07 | -3.0408e+07 |
| Gain margin in dB (Gm) | -1.3149e-06 | -7.6369e-07 |
| Phase margin in degrees (Pm) | -8.3643e-06 | -3.9448e-06 |
| Gain crossover frequency in rad/s (Wcg) | 1.2185 | 0.9288 |
| Phase crossover frequency in rad/s (Wcp) | 0.8447 | 0.6438 |
| Vector margin (Vm) | 1.4154e-07 | 6.8624e-08 |
| Vector margin frequency (Wvm) | 1.0458 | 0.6543 |
| Peak sensitivity (Smax) | 7.0649e+06 | 1.4572e+07 |

The peak sensitivity, control effort and settling Time is higher than other candidate from PID type controller. Make the LAM unity is not be drop from designing process for this case nominal plant G(S).

1. **Two Parameter Linear Algebraic Design Controller**

In this section, the *lamdesign* and *steplqr* to design two parameters controller. The tool is the same from unity feedback LAM controller. However, another function called *steplqr* use to help design the contoller. *Steplqr* is a function that receive as parameter the desired settling time in second, the settling time percentage and plant G(s). It returns closed loop transfer function in form of zpk. By testing setting up was found the best 0.2698 sec. The expression of the T(S) is :

*lamdesign* is expecting a vector of polynomial root locations that are chosen from in the design process on an as needed basis. So, a vector of real negative roots of the T(S) used. The result was as sown in Table 5.

Table 5: Systems info for Two Parameter Linear Algebraic Controller

|  |  |
| --- | --- |
| **Info type** | **Unity LAM with *lamdesign*** |
| Rise Time | 0.1776 |
| Settling Time | 0.4465 |
| Settling Min | -7.0351e-06 |
| Settling Max | -6.3503e-06 |
| Overshoot | 0 |
| Undershoot | 25.8434 |
| Peak | 7.0351e-06 |
| Peak Time | 0.6781 |
| Umax | 0.0027 |
| EssStep | 1 |
| EssRamp | Inf |
| Gain margin in dB (Gm) | 103.0497 |
| Phase margin in degrees (Pm) | Inf |
| Gain crossover frequency in rad/s (Wcg) | 0 |
| Phase crossover frequency in rad/s (Wcp) |  |
| Vector margin (Vm) | 1.0000 |
| Vector margin frequency (Wvm) | 0 |
| Peak sensitivity (Smax) | 1 |

There was a lot of advantage of using two parameters LAM controller but these advantage does not compile with all set designed goals.

1. **Conclusion**

**Appendix:**

****

****

**The written code can be found in the following github repository:**

**https://github.com/mohammedalsayegh/ECE414**