Teaching PID Controller Design Using Matlab

Robert Barsanti
The Citadel

Abstract

Proportional Integral Derivative (PID) controllers are an industrial workhorse to achieve process control for an assortment of manufacturing physical variables such as position, speed, temperature, or pressure. This paper provides a tutorial on PID controller design and tuning suitable for a course in dynamic system modelling and automatic controls for electrical or mechanical engineering students. The tutorial and associated student project can be easily expanded to suit a graduate level course in digital or optimal control theory. The paper outlines the basic theory of PID control explaining how the proportional, integral, and derivative gain setting effect the closed loop system response. The interaction of these variables is demonstrated by a detailed example. A student project assignment is included in the paper to assist in reinforcing the PID concepts. The project exploits the user-friendly adaptability of the Matlab Simulink toolbox software to encourage students to explore the trial- and- error nature of controller design. The trade-off between the controller output transient response and the system robustness is investigated using the Matlab PID Tuner to illuminate reference tracking and disturbance rejection. The students are encouraged to experiment with controller parameter variations to obtain a firm understanding of the relationship to system response. Students learn first-hand how system rise time, settling time, and percent overshoot can be influence by the parameter settings.

Keywords

Controllers, PID, Matlab, Simulink

Introduction

Proportional Integral Derivative (PID) controllers can be purchased off the shelf on a chip, or as a digital micro controller, so it is rare for a control engineer today to need to design a PID from discrete components or even op amps. Yet, the controller however realized, must have the proportional, integral and derivative gains adjusted to provide the optimum system closed loop response. The methods available include 1) Trial and Error, 2) Zieger-Nichols, 3) Cohen-Coon tuning method. 4) Software [1]. This paper will discuss method 4 using matlab software ® in an academic setting [2]. The information provided is suitable for students familiar with linear systems theory and continuous-time feedback controls systems, at the depth presented in a college text such as reference [3]. The student project presented could be assigned either third or fourth year engineering students a course in dynamic controls.

This paper contains the following additional sections: PID Controller Theory, How to Tune a PID Controllers Using Matlab Software, PID Example, Student Project, and Summary.

PID Controller Theory

The below description is provided as a review of PID controller theory. Controllers are used in closed loop systems to influence the response of the system to setpoint changes and disturbances. The input to the controller is the error signal, which is the difference between the setpoint and feedback of the closed system output. The output of the controller is sent to the plant to influence the system response. See figure 1.

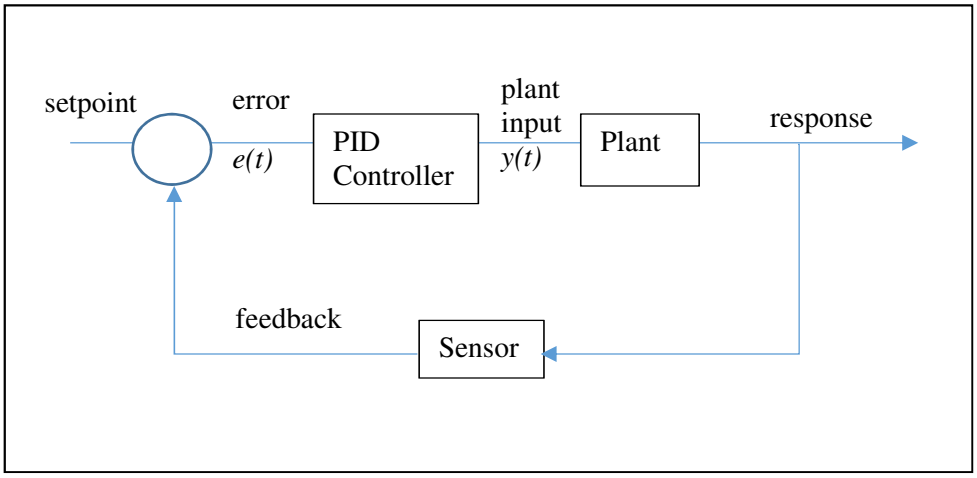


Figure 1. Plant subsytem and PID controller.

As the name implies, the PID controller provides three separate actions on the error signal to produce the controller output, proportional, integral, and derivative. The proportional action produces an output proportional to the error signal $y(t) = Kp * e(t)$. Thus, no proportional action will result when the plant output is equal to the set point, since the error is zero. As the proportional gain Kp is increased the controller will produce a larger signal in response to an error. This amplification of the error signal will result in a faster response to errors, but may also add instability. These effects are quantified by the response having a shorter rise time, and an increase in the percent overshoot. Conversely, reducing the proportional gain will reduce overshoot but will increase the steady state error.

The derivative action provides a control signal proportional to the time rate of change of the error signal. $y(t) = Kd * de(t)/dt$. The derivative term provides an anticipatory element to the controller, providing larger controller response to rapidly changing error signals, and smaller response to slower changes. This permits a faster system transient response without increasing the percent overshoot. The derivative action alone has little effect on the steady state behavior of the system. It cannot remove a fixed error (constant steady state error) since the derivative of a constant is zero. Therefore, the derivative element of the PID controller would produce zero output to a non-changing error.

The integral action produces an output proportional to the accumulated (integral) of the error signal, allowing the controller to zero the steady state error between setpoint and system output. $y(t) = ki \int e(t)$. From a control system theory perspective the integrator term improves the steady

state performance because it increases the "system type" by one by adding a zero at the origin. Intuitively, the integral element insures that even small errors will eventually amass to produce significant controller output. The combination of the three actions is summarized as

$$y(t) = Kp * e(t) + Ki \int e(t)\,dt + Kd * de(t)/dt. \quad EQ\ 1.$$

In equation 1, *y(t)* is the system output, *e(t)* is the error signal, K_p, K_i, and K_d are the proportional, integral, and derivative gain constants. The selection of the three constants *Kp, Ki*, and *Kd*, to achieve a specific response is known as tuning. Manual tuning consists of adjusting the three gain factors in a systematic manner using multiple trials. Table 1 below displays the general effects of each controller parameter on the system response. These are only guidelines and specific results may vary. To learn the actual response will require testing or simulation using your specific plant.

PID GAIN PARAMETER INCREASE	EFFECT ON SYSTEM RESPONSE			
	RISE TIME	% OVERSHOOT	SETTLING TIME	STEADY STATE ERROR
Kp	Decrease	Increase	Minor Change	Decrease
Ki	Decrease	Increase	Increase	Decrease
Kd	Minor Change	Decrease	Decrease	No Change

Table 1. Effect of increasing PID parameters on the system response [4]

Using Matlab Software

If knowledge of the mathematical model of the plant can be obtained then software can be used to determine the values for *Kp*, *Ki*, and *Kd*. One popular software application is Matlab which is Mathworks Corporation commercial software package [2].The matlab control/linear system toolbox [6] contains many built-in functions to assist in representing and computing the properties of linear systems. Matlab's Simulink ® is a model based simulation and design software oriented that can be used to model the PID controller and plant. Using Simulink requires the placement of block icons to represent the components and then connecting the blocks along the signal path. No writing computer code is necessary. Input blocks are provided for a variety of signals and output blocks provide for a variety of graphing possibilities. See figure 2 for an example of a Simulink block diagram.

PID Controller Example using Simulink

For this example, the plant to be controlled will be a motor positioning system. The transfer function for the motor is shown in figure 2 and represents the relationship between the control signal input and the final position of the motor. Also shown are P, I, D elements of the controller, an input source for the setpoint, and a scope to monitor the output. The goal of the example is to demonstrate the effect of the PID controller gains on the system response.

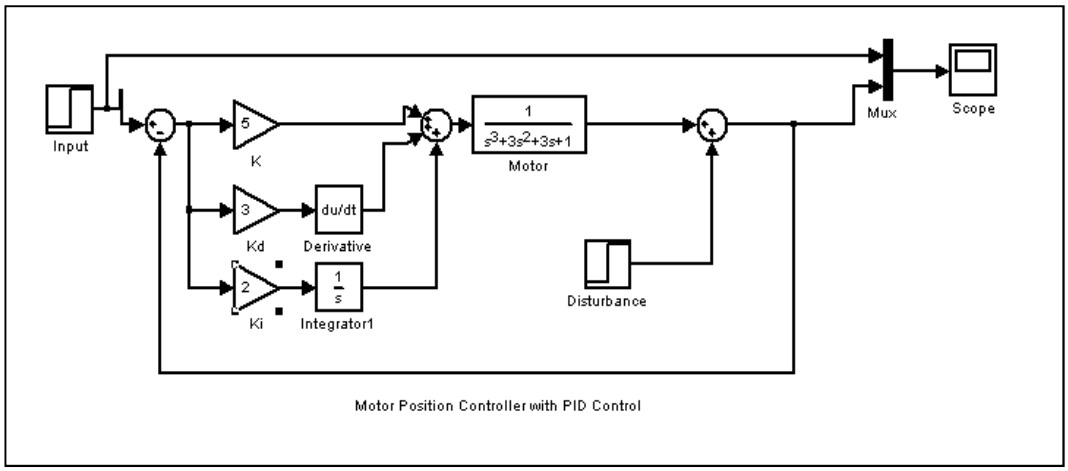


Figure 2. Example of a Simulink block diagram.

The Simulink model shown in figure 2 was constructed and the resulting system response is shown in the plot of figure 3. The figure displays the system response to various controller parameter settings. The motor closed loop response is shown with the solid line for reference. The dotted line represents the response to a proportional gain alone.

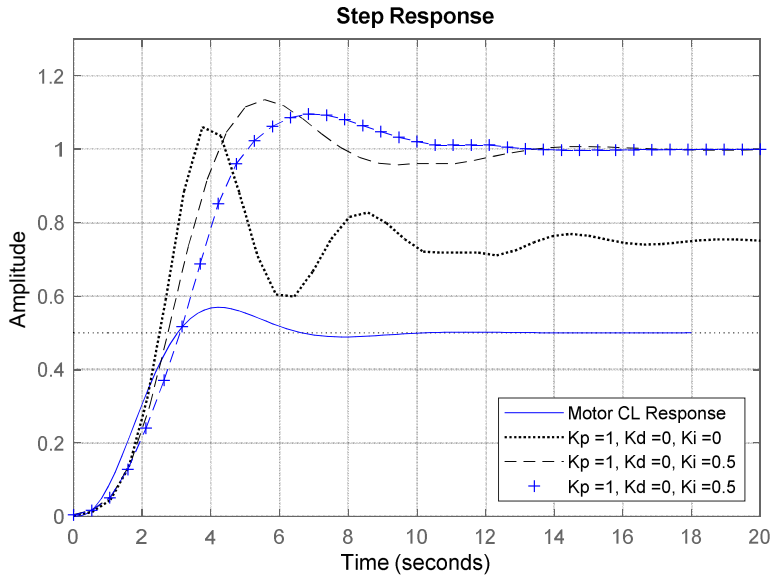


Figure 3. Motor Step response.

Notice that the rise time improves and the steady state error is reduced, but the percent overshoot is higher. Larger values of proportional gain *Kp* will result in a faster response (reduced rise time) and smaller steady state errors but increased overshoot (and possibly oscillations) will result.

When the PID controller gains are set to *Kp* = 1, *Kd* = 0, and *Ki* = 0.5 the steady state error is eliminated by the integral action of the PID controller. With the controller gains set to *Kp* =1, *Kd* = 0.5, *Ki* = 0.5, we can observe the reduced overshoot improvement that comes with the derivative action.

Using the PID tool with Simulink

Matlab software provides a PID tool that can be used to assist in determining a satisfactory set of PID gains. This tool can be called directly from the command window using the *pidtuner()* function, or by use of the *PID controller block* in Simulink. The PID tuner provides a user interface that permits adjustment of the tuning parameters via slide controls. The interface also provides a graphical display of the system step response that updates as the slider controls are adjusted. Figure 4 below displays a Simulink model using the PID block to form the PID controller.

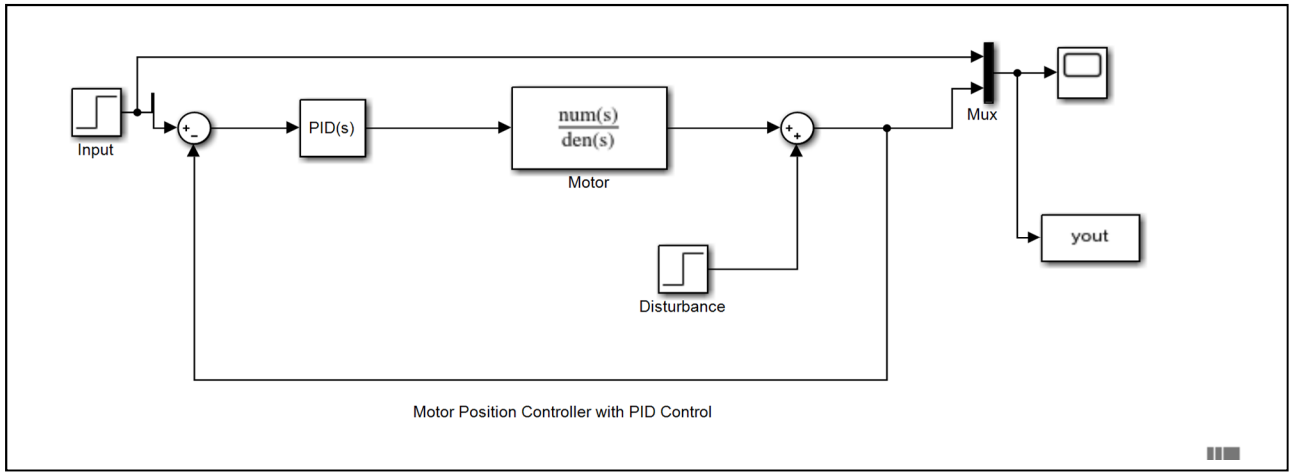


Figure 4. Simulink model with PID Controller block

By double clicking on the *PID block* the model parameters become visible as shown in figure 5. This interface allows a number of interactions. First, it permits selecting continuous time or discrete time models using a radio button. It permits direct entry of the P, I, D gain parameters. Two choices of tuning method are provided. The transfer function based (PID tuner) which open the interface shown in figure 6. The alternate selection of tuning method is frequency response based tuning which permits selection of a target bandwidth and phase margin. We will discuss transfer function based tuning first in this paper.

Clicking the T*une…* button will get the tuning palate to appear as shown in figure 6. This is the main PID tuning interface. It displays the step response for PID parameters for both the original user entered values shown in figure 5 with a dotted line, and the step response produced by the tuner determined values for *Kp, Ki,* and *Kd* with a solid line.

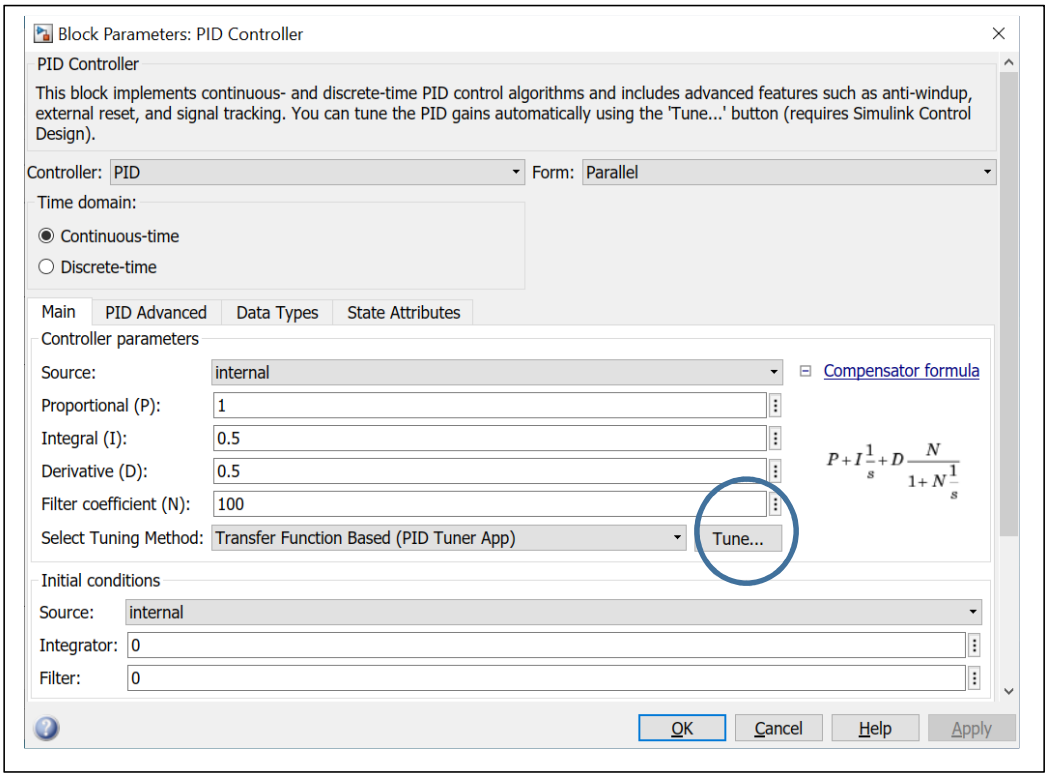


Figure 5. Interface for Simulink Block Parameter PID Controller.

By selecting add plot and further selecting chioce Output disturbance rejection will add a second plot to the palate as shown in figure 7. By clicking on the *show parameters* option to get the parameter chart super imposed in figure 7.

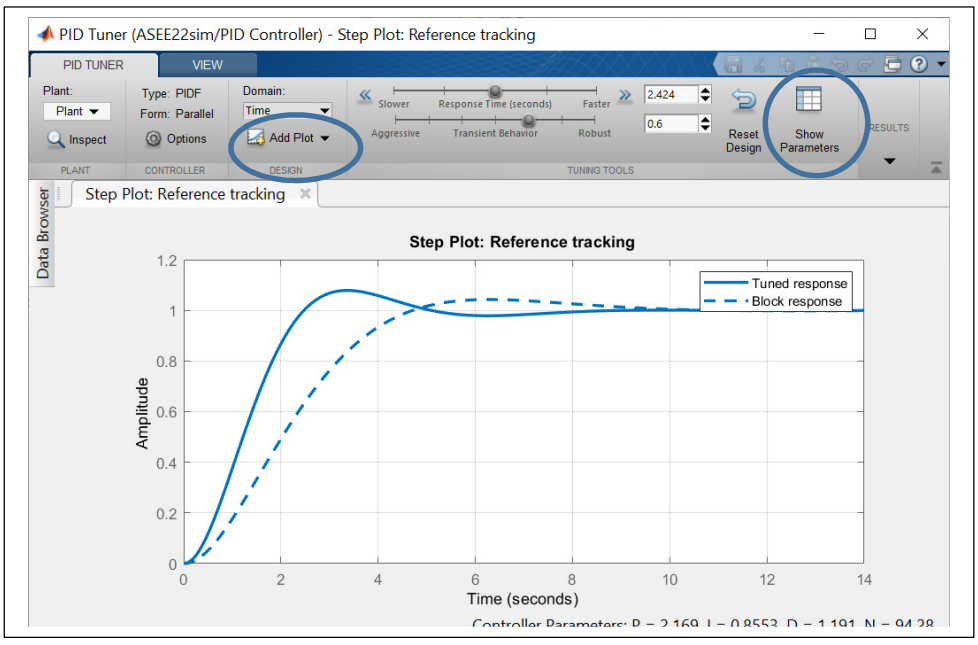


Figure 6. PID tuner block used to evaluate the PID gains.

Adjustments to the tuner are achieve using the sliders at the top of the interface shown in figure 7. One slider for response time and one for transient behavior.

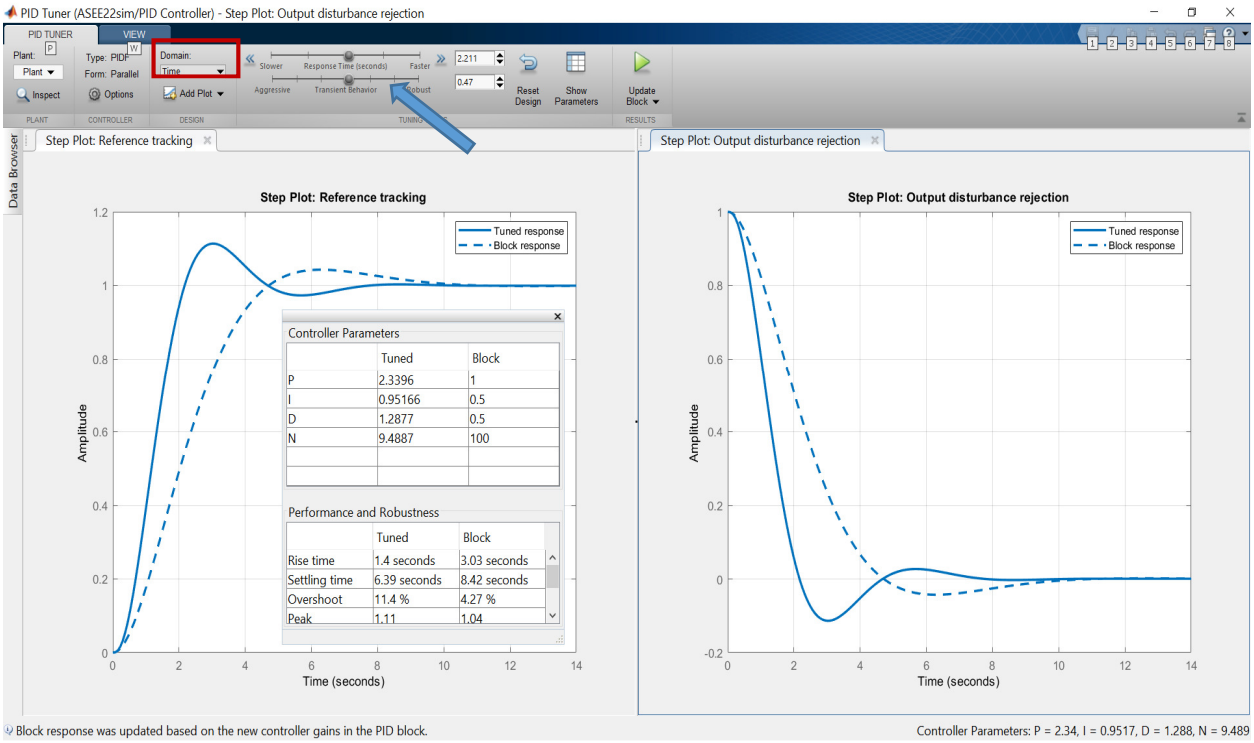


Figure 7. PID tuner block used to evaluate the PID gains.

The response time slider has settings from *slower* on the left to *faster* on the right and changes the response rise time accordingly. The *transient behavior* slider is labeled *aggressive* on the left and *robust* on the right. Changes to the sliders display instantaneously on the step response plot. The response time slider adjusts the speed of response such that rise time is improved but percent overshoot may also increase. The transient behavior slider adjusts the steady state response. Moving this slider to the right will improve the overshoot but may result in slow disturbance rejection. Moving the slider to the robust setting will improve disturbance rejection.

By changing the Domain setting from Time to the Frequency setting (red box in figure 7) the Bandwidth and Phase Margin sliders will appear (arrow figure 8). The system response can now be tuned by setting the bandwidth and phase margin. By increasing the bandwidth the system response will speed up resulting in faster rise times and added overshoot. Increasing the phase margin will improve system stability and reduce the reponse overshoot. The default setting for the simulink algorithm is a 60-degree phase margin.

Once we are satisfied with the system step-response we can update the model by selecting the green arrow labeled *update block* in the PID tuner interface (circle in figure 9). This will automatically transfer the PID gains to the PID controller block inside the Simulink model (figure 6). At this point the Simulink model can then run with the updated parameter setting to verify system performance.

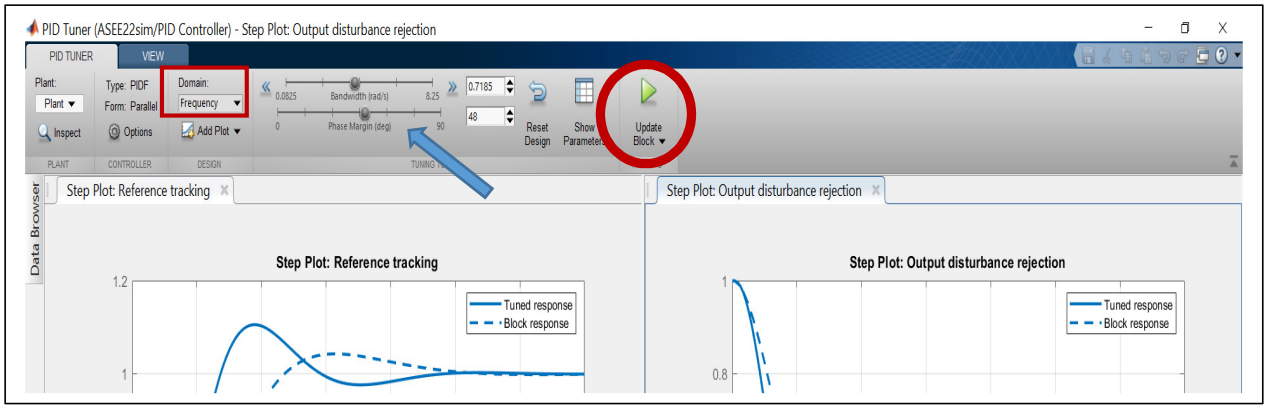


Figure 8: PID tuner block showing Domain setting, Frequency domain sliders for bandwidth and gain margin and the Update block to transfer PID gains to the Simulink model.

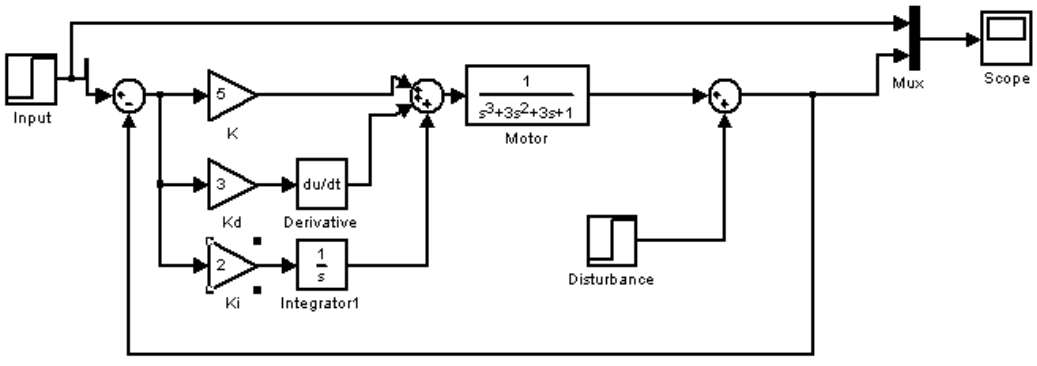
Student Project

A student project is presented below to help explore the details of PID gain selection and investigate the trade- off between transient response and system robustness using the same plant shown in figure 2. This project is suitable for either an undergraduate or graduate class in automatic controls. The student project contains the objective, five deliverables, and three separate activities: plotting the step response, investigating disturbance rejection, and changing the design focus from transient response to disturbance rejection using the PID Tuner. The project handout is shown in full below.

PID Control for a Motor Positioning System Project

Objective: To use Simulink® to simulate the response of a motor position control system and experiment with PID control.

Part 1. Plotting the Step Response



Motor Position Controller with PID Control

a) Construct the above system using Simulink.
   - Set the simulation time for about 20 seconds
   - Set the disturbance step input = 0 (Final Value) for this part

b) Set Kd and Ki to zero. Vary the proportional gain (Kp) from 0 to10, discuss how rise time, steady-state error, and % OS are effected. Record the fastest rise time without overshoot, the steady state error, and K for the report.

c) Set Kp = 8, derivative gain Kd from 0 – 10, and integral gain Ki 0-10, make a table showing Kp, Kd, Ki Rise Time and %OS, and steady state error, at some (about 8) representative values of Kd and Ki.

Part 2. Disturbance Rejection

Set the disturbance for a step of Final Value =1, with a start time at 0 seconds (Step Time). One technique for experimentally determining Kp, Kd, and Ki is the following:

- Set Kd = Ki =0, adjust Kp until oscillations just begins. Record this Kp.
- Set Kd = 0.6 * Kp, and Ki = 0.4 * K

This should provide a reasonable starting point from which to fine adjust the parameters. Perform this procedure and print out the scope output for the report.

Part 3. Design focus using PID Tuner

Insert the PID controller block into the model. Then launch the PID tuner by clicking on the tune icon.

a) Adjust the transient response slider to obtain a satisfactory rise time, percent overshoot and, settling time. Then select *update* to transfer the PID tuner settings to the simulink model. Perform this procedure and print out the scope output for the report.
b) Change the PID Tuner design focus to disturbance rejection under the *options* menu. Then select *tune* to obtain the new controller gains. Then select *update* to transfer the PID tuner settings to the simulink model. You should see improved disturbance rejection but more transient overshoot. Perform this procedure and print out the scope output for the report.

Report Deliverables

1) Print out of your simulation diagram and any one scope print out showing the step response. Label the scope output with the values of Kp, Kd and Ki used to produce it.
2) Discussion of part 1b.
3) Table of part 1c, show the values found for case 1b in your table and highlight. Briefly discuss the effect of Kd, and Ki on the results.
4) Show the part 2 results separately. Provide the scope print out, and indicate on it the the values of Kp, Kd, and Ki used to produce it.
5) Scope print outs from parts 3a and 3b.

Summary

This paper discussed the use of matlab software to teach students about PID control systems. An example was presented to illuminate how each of the controller gains effect the system output. An introduction to the use of Matlab Simulink software was provided and a comprehensive student project using Matlab Simulink to explore PID tuning was provided for classroom use. The student learning objectives of the project are:

1. Learn to use simulation software (Simulink) to model a system and PID controller.
2. Learn the effect on system transient response to changes in PID controller gain settings.
3. Learn the effect on system steady state response to changes in PID controller gain settings.

References

1 What is PID Controller? Learn PID Controller Working, Structure and Tuning Methods (circuitdigest.com)
2 Matlab Software, https://www.mathworks.com/products.html , version 9.4, release R2018a, Simulink 9.1, 2018
3 Control System Engineering, N.S. Nise, 2019 John Wiley and Sons, ISBN 978-1-119-47421
4 Clayton Wilson, How to Manually Tune a Three-Mode PID Controller, March 4, 2010, Yokogawa Corporation of America, Newman, Ga
5 PID Software Tuner, DotX Control Systems, https://www.pid-tuner.com/software/
6 Mathworks Control Systems, https://www.mathworks.com/solutions/control-systems.html

Robert Barsanti

Bob Barsanti is a Professor in the Department of Electrical and Computer Engineering at The Citadel, where he teaches a variety of courses and does research in the area of target tracking and signal processing.