EE 4210 Control Systems

Lab 10 – Servo PID Design

Kennesaw State University

Marietta, GA

Kyle Maldonado

Johnny Lozano

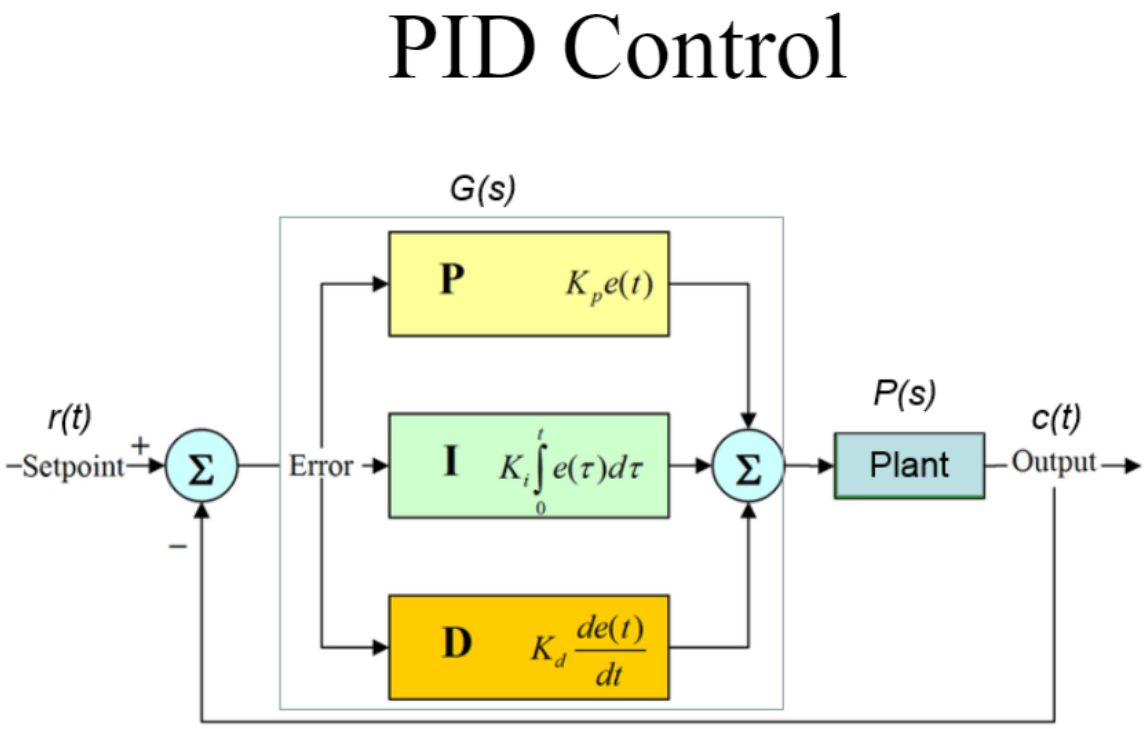
April 26, 2022

Abstract

The main purpose of this lab is to design an electromechanical system that needs to a mechanical arm accurately and precisely for a robotic pick-and-place application. The first main object is to design and simulate a PD servo closed loop system in order to design a controller for hardware implementation in the subsequent steps. The second main objective is the close the loop for P, PD, and PID controllers and demonstrate that the PD simulation adheres to the performance requirements. The last objective is to manually tune the PID systems in order to achieve the design requirements and to achieve the highest compositely weighted performance score.

Background

Proportional Integral Derivative (PID) controllers are ubiquitous, and they are widely used in most automatic process control applications. A PID controller is used in industrial control applications to regulate temperature, airflow, pressure, speed, and other factors through the use of a control loop feedback mechanism.



$$G(s) = K_p + \frac{K_i}{s} + K_d s$$

Component Diagram of PID Controller

By adding an integrator to the controller, the steady state error in the overall all system is removed which is what makes PID controllers efficient at regulating automatic processes. However, due to the fact that an integrator is added to a PD controller, a PID controller is difficult to design/compute due to the equation being a third order transfer function. Subsequently, manually tuning the PID controller is the preferred method for implementation in a control system.

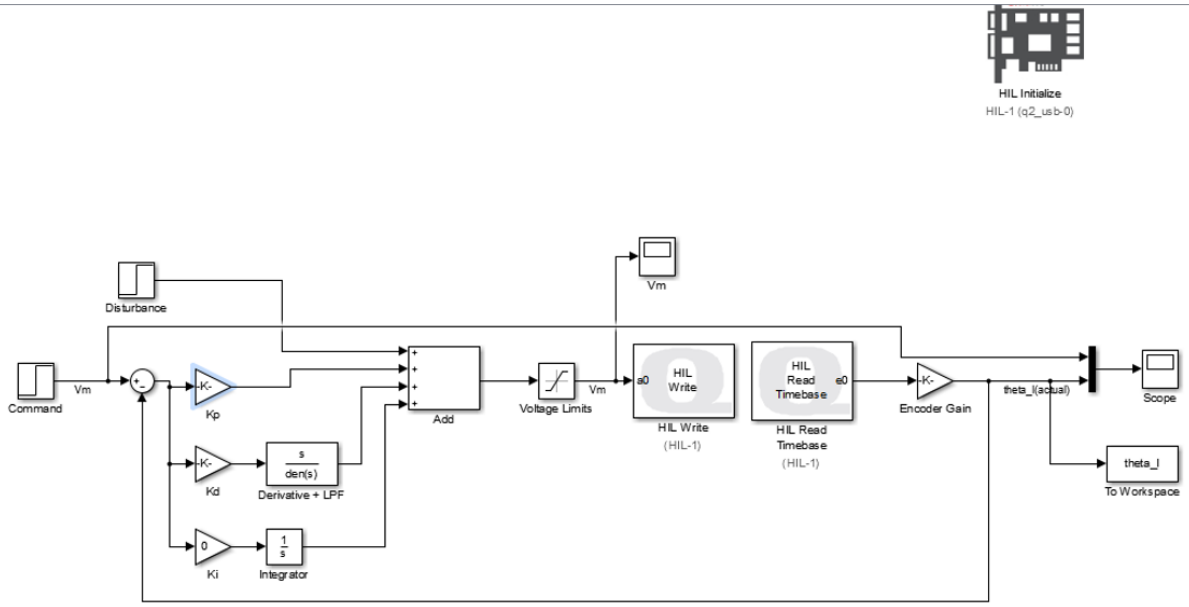
Procedures and Results Discussion

I. PD Servo Closed Loop Design and Simulation

1. Placeholder text
2. Placeholder text

II. Building the PID System

3. In Simulink, a PID control system was modeled using mostly the Simulink library and a few Quarc HIL blocks.

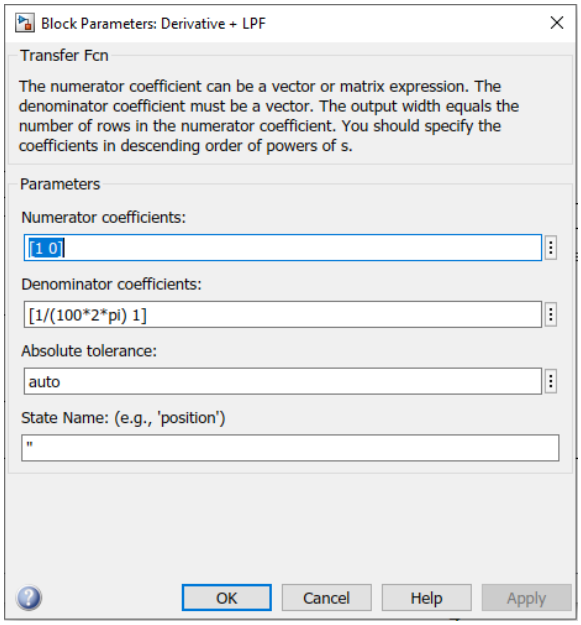


PID Simulink Design Model

4. The *voltage limits* saturation block was set to +/- 24V.
5. The *step command* was set with Step time = 0.5 sec and Final value = 25 degrees.
6. The *step disturbance* was set with Step time = 0.5 sec and Final value = 1V.
7. The *To Workspace* block was labeled as *theta_l* and formatted to *Structure with Time*.
8. In order to minimize system noise, a low pass filter was set in the derivative path as

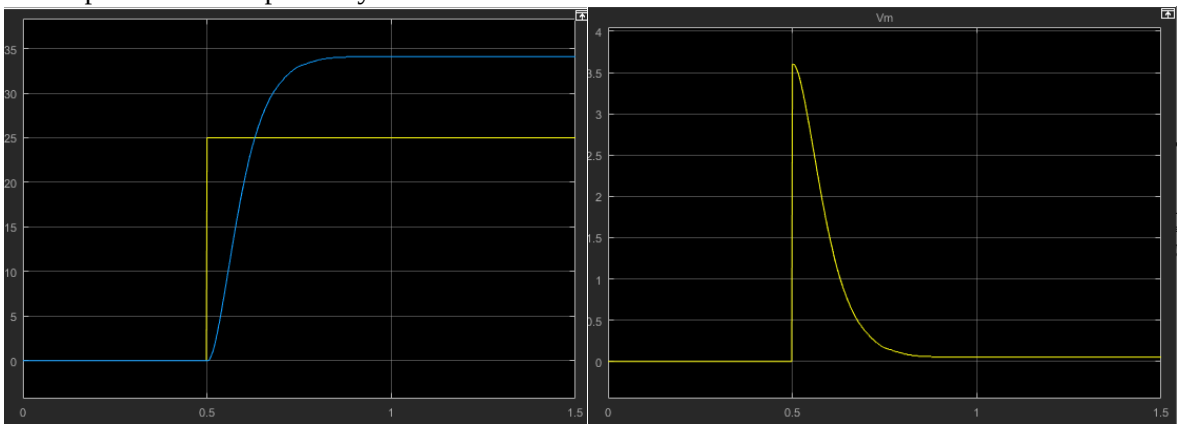
$$\frac{(100 * 2 * \pi)}{s + (100 * 2 * \pi)}$$

with a cut off frequency at 100Hz. The configuration looks as follows:



Block Parameters of Derivative + LPF

9. The simulation time was set to 1.5 seconds.
10. The design was configured as a simple P controller in order to ensure that the model was behaving correctly. Kp was set to 0.104 and Kd and Ki were set to zero. The simulation was run, and the plots are output and Vm respectively.



Output (left) and Vm(right) of Simple P Controller

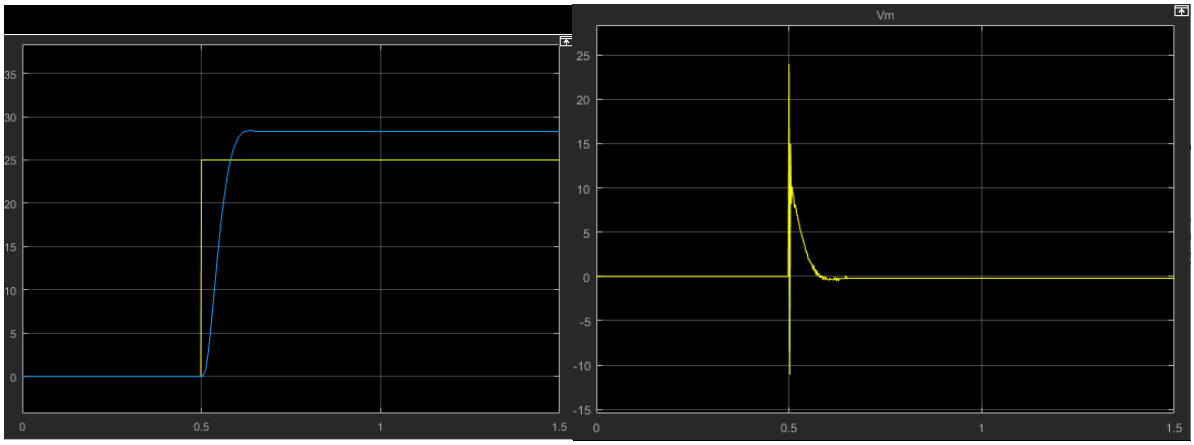
11. The file *controller_score.m* (see Annexure) was downloaded to the working directory in order to display the performance metrics of the system.

```
>> control_score(theta_1)
RiseTime: 0.0399 sec
SettlingTime: 0.0732 sec
Overshoot: 0 percent
Steady State Error: 2.3 percent
```

Performance Metrics

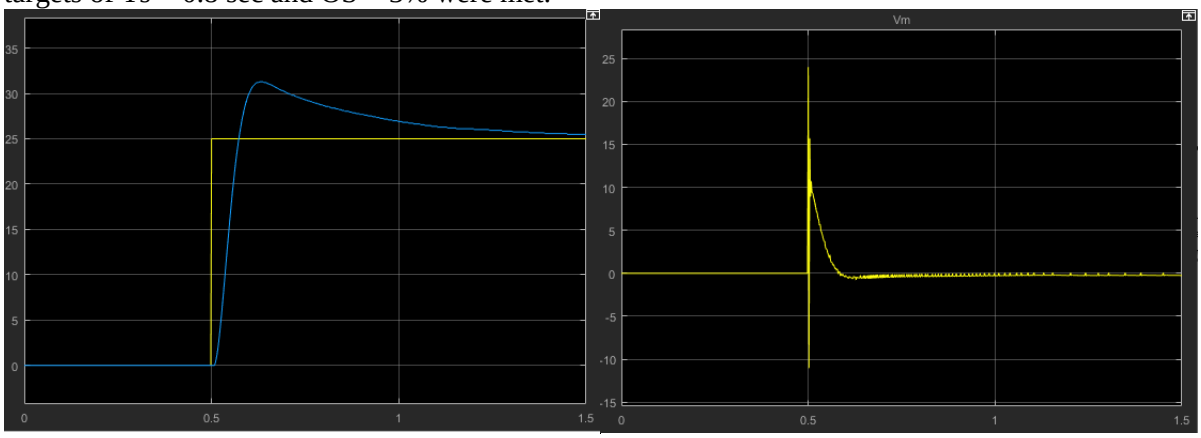
III. PD & PID Controller

12. The values for Kd and Kp calculated in Section I were configured into the gain blocks of the model, and the simulation was run.



PD Controller Initial Output

13. As shown above, the blue graph is slightly above the step input which shows the presence of steady state error. The Ki value was incremented until steady state error was eliminated and the design targets of Ts < 0.8 sec and OS < 3% were met.



PID Controller Output with Configured Ki Value

IV. PID Tuning

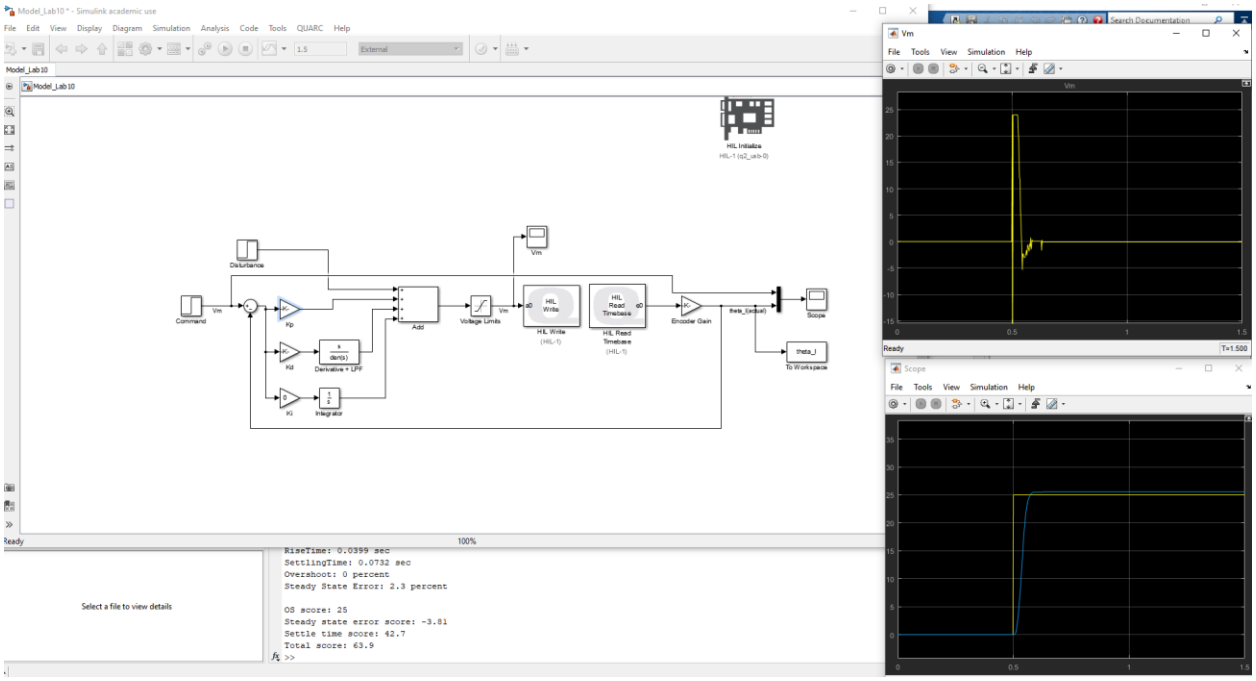
14. Our final goal was to test our proficiency in understanding and tuning a PID control system. We started with all three gain values (Kp, Kd, and Ki) set to zero, and we tried to reach the following performance parameters:
    - Overshoot less than 5%
    - Steady state error less than 2% after one second
    - Fastest possible settle time Ts < 0.5 seconds

    The goal was to achieve the highest possible performance, so a formula (implemented in *control_score.m*) was used to measure the composite weighted performance score:

$$X = 25\frac{(5\% - \%OS)}{5\%} + 25\frac{[2 - \%e¿¿ss(t=1)]}{2} + 50\frac{(.5 - T_s)}{.5}¿$$

Equation for Composite Weighted Performance Score

After much configuring, we managed to achieve a total score of 63.9 with the Kp set to, Kd set to, and Ki set to 0. Our results are as follows:



Final PID Tuned Design and Outputs

```
>> control_score(theta_1)
RiseTime: 0.0399 sec
SettlingTime: 0.0732 sec
Overshoot: 0 percent
Steady State Error: 2.3 percent

OS score: 25
Steady state error score: -3.81
Settle time score: 42.7
Total score: 63.9
```

Final PID Tuned Performance Metric Score

Conclusion

In conclusion, the objective of this lab was to design and simulate a PD servo closed loop system, to design controllers for hardware implementation, and to manually tune the PID system to achieve specific performance targets. The lab provided hands-on experience with designing and implementing PID controllers, which are widely used in automatic process control applications. The objectives of the lab were met by first designing and simulating a PD controller, followed by closing the loop for P, PD, and PID controllers, and manually tuning the PID system to achieve a composite weighted performance score of 63.9. The lab helped to reinforce the theory and concepts of control systems while also highlighting the challenges and obstacles involved in designing and implementing a PID system. Overall, the results of the

lab matched the theory and mathematical models, and the experience gained was useful for understanding the theory and practice of control systems. In the future, it would be beneficial to explore different tuning methods and to consider any limiting assumptions that may have influenced the experiments.

Annexure

```
%---Control_Score.m-----------------------------------------------

function [] = control_score( theta_l)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
% grab tehta_l from workspace that was deposited by simulink run
step_start=.5;
step_final=25;
Si=stepinfo(theta_l.signals.values,theta_l.time);
ess=100*abs(step_final-theta_l.signals.values(end))/step_final;
Ts=Si.SettlingTime-step_start;
OS=Si.Overshoot;
fprintf('RiseTime: %0.3g sec\n',Si.RiseTime);
fprintf('SettlingTime: %0.3g sec\n',Ts);
fprintf('Overshoot: %0.3g percent\n',Si.Overshoot);
fprintf('Steady State Error: %0.3g percent\n',ess);
score1=25*(5-OS)/5;
score2=25*(2-ess)/2;
score3=50*(.5-Ts)/.5;
fprintf(' \n');
if(1)
    fprintf('OS score: %0.3g \n',score1);
    fprintf('Steady state error score: %0.3g \n',score2);
    fprintf('Settle time score: %0.3g \n',score3);
    fprintf('Total score: %0.3g \n',score1+score2+score3);
end

end
```