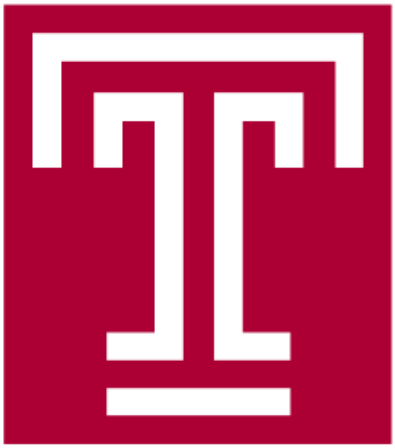
Temple University

College of Engineering

Department of Electrical and Computer Engineering (ECE)

Student Report Cover Page



Course Number: ECE 3412

Lab 10: Practical examples of PID control

Student Name: Devin Trejo; Robert Irwin
TUid: 914924557; 914980083
Due Date: 4/30/2015

TA Name: Michael Korostelev

Grade: / 100

I. Introduction

In this lab we will be experimenting with practical PID control applications. In particular, we will concern ourselves with controlling the speed of a DC motor using both the output of the encoder, and an accelerometer.

II. Procedure

For this lab we will reference our PID control to the throttle vale position. The throttle control is what determines a car's acceleration and is tied to the angle in which the gas pedal is situated. To begin we will showcase different scenarios with varying PID control. We will discuss which levels of PID control correspond to the most desired response.

Next we will introduce an accelerometer to our system.

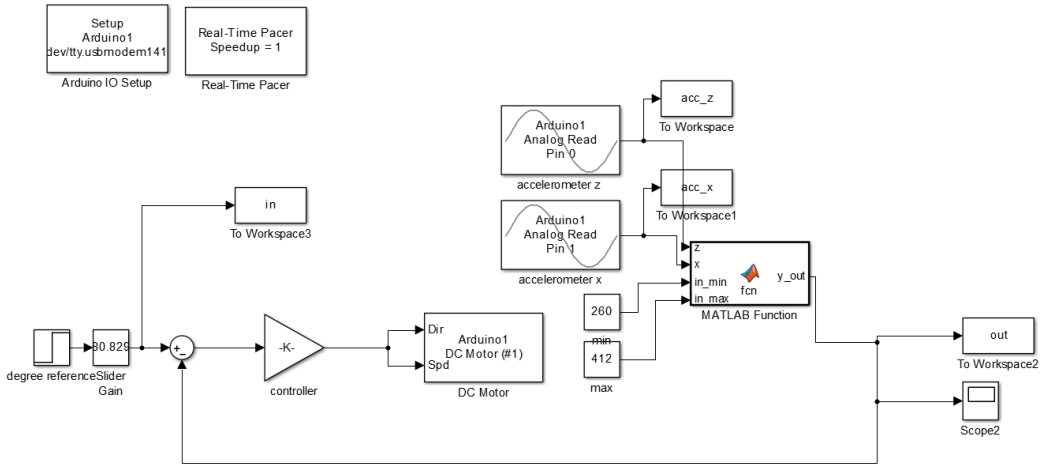


Figure 1: Simulink model of our Accelerometer Feedback control

The accelerometer feeds into a MatLab function that also takes into account the current speed of our motor. The two in combination we be feed back into our DC motor controller. Our accelerometer will be able to determine if our motor is accelerating uncontrollably.

III. Results

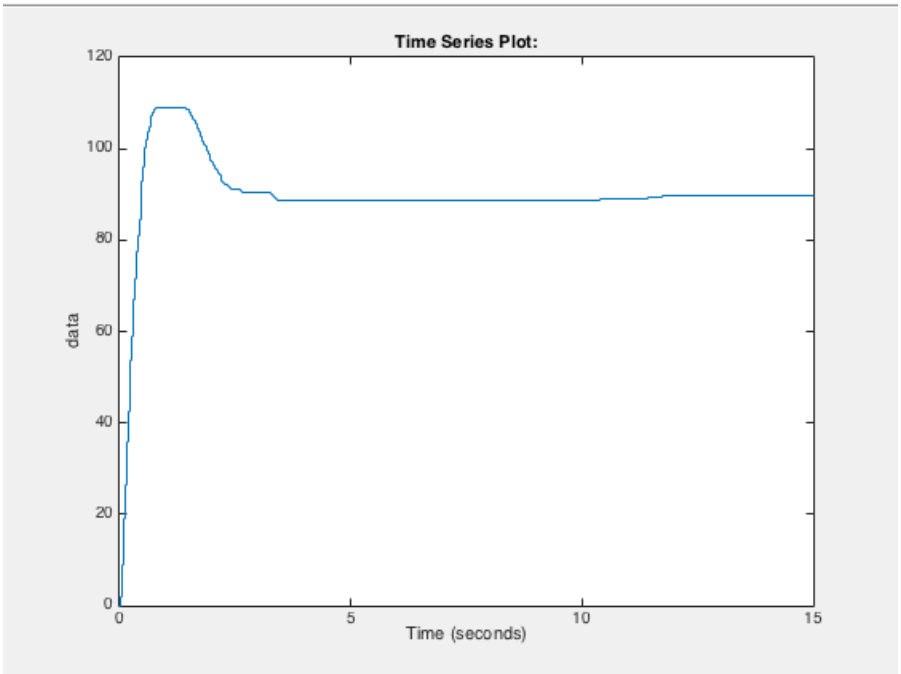


Figure 2: Speed Input=90, P=.7, I=.2, D=0

Above we see that there is no steady state error. However, there is some overshoot. This would result in a very rough ride for the driver.

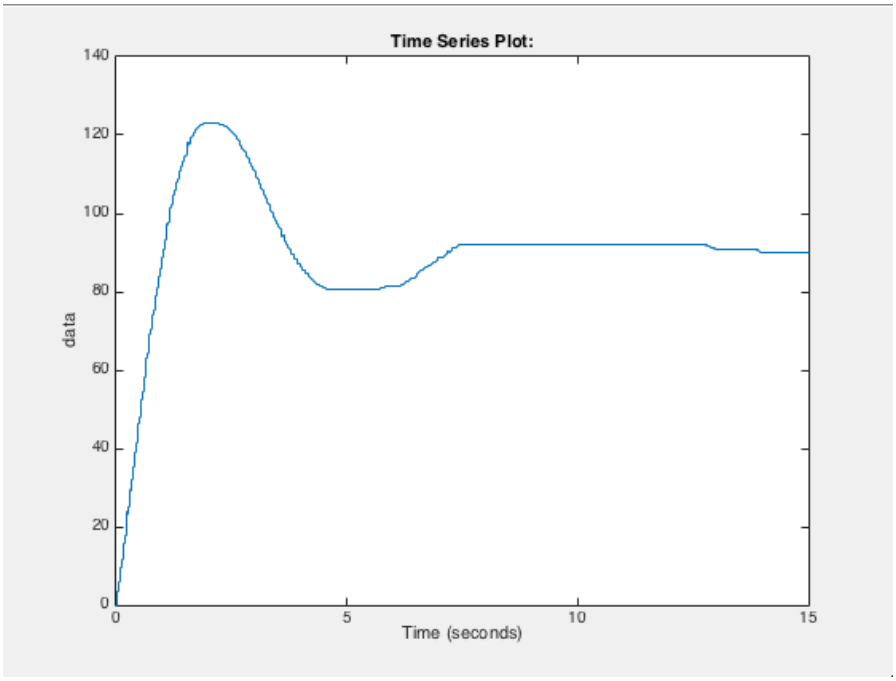


Figure 3: Speed Input=90, P=.7, I=.2, D=.5

We see that if we add a small derivative gain, the percent overshoot, and settling time increase. This is undesirable for this type of system.

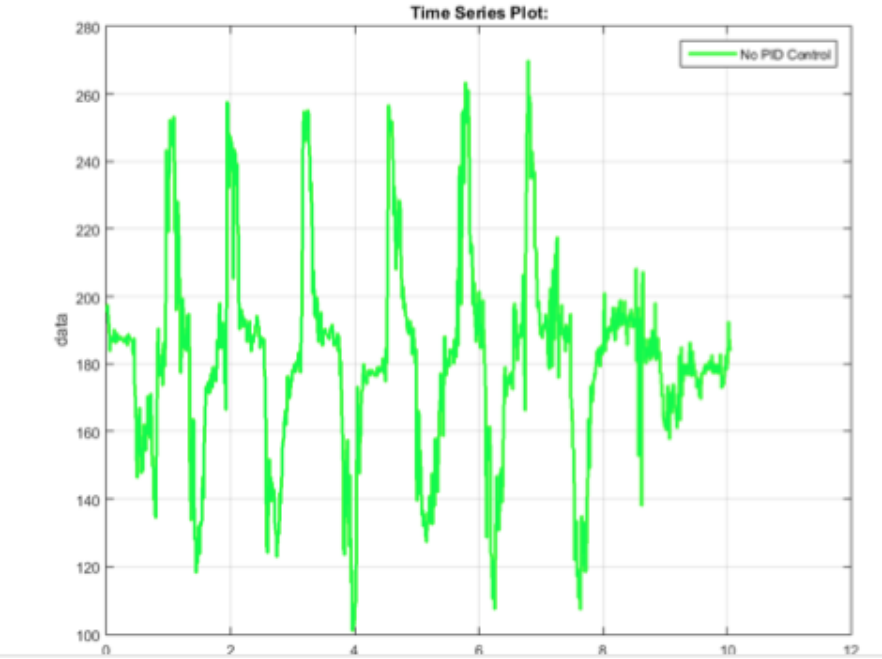


Figure 4: Accelerometer with no PID Control

If we now begin to add PID control to the system, the response stabilizes.

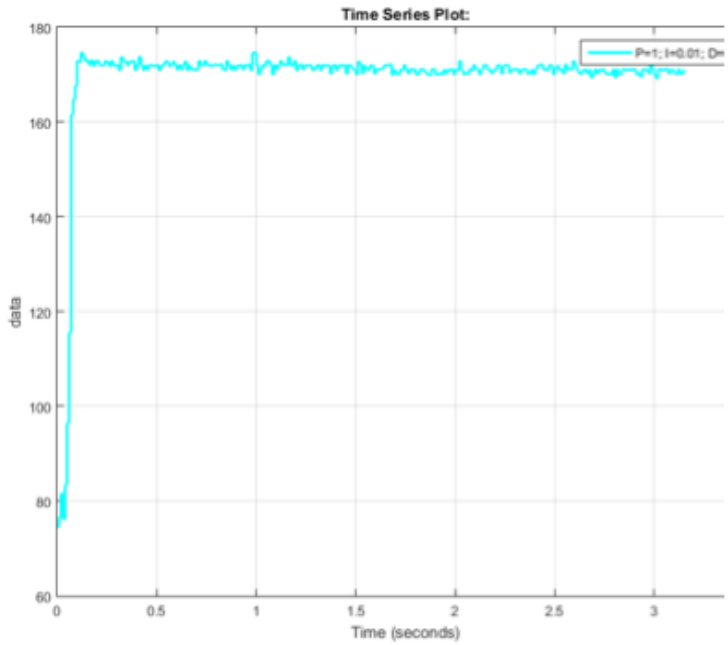


Figure 5: Accelerometer with PID Control

In the figure above, P = 1, I = .01, and D = 0. This produces a very fast transient response with almost now overshoot and a fast settling time.

Different gains effect the drivers experience in the following way. The larger the gain the more sensitive the gas pedal. With a high gain, a slight change of the angle of the gas pedal corresponds to a large change in the vehicle's acceleration. Also, if the gain exceeds the stability condition, the vehicle will accelerate uncontrollably, and there will be no opportunity to recover. A problem with using an accelerometer in our system is that it cannot differentiate between acceleration due to the car and acceleration due to gravity. It makes it difficult to control since if the accelerometer is not securely anchored down a bump can cause the feedback to introduce an in-stability condition. The inaccurate readings will make the car become jerky.

We now introduce the derivative control to the Arduino code. This is shown below.

```
  int index = 1;
  double error= {setpoint - y};
  double errorD[2];
  //initialize errorD
    if(index ==1){
      errorD[1] = setpoint;
    }
    Ii = Ii+error;
    //bring in the D
    errorD[0] = errorD[1];
    errorD[1] = error;
    Dd = errorD[1]-errorD[0];
    index = index+1;
  double pid = P * error + I * Ii + D * Dd;
index = index+1;
  Serial.print("    error: ");
  Serial.println(pid);

  if (pid > 0){
    myMotor->run(FORWARD);
    myMotor->setSpeed(pid);
  }
```

Figure 6: Arduino Code with Derivative Control

As derivative gain measures the rate of change of the error, we have to initialize the first point to be equal to "setpoint". Then we set error equal to setpoint-y which is how far the current sample is from the steady state. The previous sample is subtracted from the current sample, which is the derivative gain. This process is repeated for every sample.

In previous labs we modeled the DC motor as a first order system. First order systems are easy to work with in our calculations thus we stuck with them. In reality the DC motor apparatus should be modeled using a higher order system since you can model motor, the load it is carrying, a whole assortment of parameters governing the motor. The higher order the system we produce the more accuracy we can expect form it and can help with accounting for the non-linarites of our system. For example, we can expect non-linear frictional or moment of inertia models in our system.

To determine if a parameter is a pole or zero you need to look at the differential equations around you model. Reflecting back to lab 3 in the Controls Lab course was the first time we were introduce to the model of DC motor. We took into account the inductance of the motor, the moment of inertia, the torque which in turn gave us angular velocity and the current derivative. It is the ratio of these

parameters that created our transfer function. Depending if it is on the numerator or denominator determine if it a pole or zero.

IV. Discussion

As we come to an end of the controls class laboratory assignments we see how what we learned thus far can applicable to other projects we will encounter. For senior design our team has decided to create a Broadband-Hamnet Microwave Communication System for the city of Philadelphia. Although there are no real applications of control system in constructing the network, we can test the network by creating a network control system. Network controlled systems are widely used today to control HVAC systems, power distribution, lighting systems. Network controlled systems have an interesting aspect in that you need to account for the time delay procured in transmission. With the vast availability of microprocessor one can create individual control systems easily. You can then hook it up to the network and create a grid of control systems that feedback into one central location.