Development of a Discrete PID Control Laboratory for Undergraduate EET Curriculum: Modeling, Analytical, and Empirical Data Collection Tool

N. Moehring, C. Vogel, J. Porter, J. Morgan
Electronics Engineering Technology Program, Texas A&M University

Introduction

The typical control systems course relies on simulation tools to demonstrate the concepts presented in the lecture. While simulations are useful for visualizing the theory, only a hardware-based experiment can demonstrate how the theory translates to the real world.

During the Fall Semester 2000, the ENTC 462 class at Texas A&M performed an experiment to test the capabilities of a classic PID control system on a plant modeled by a second order equation using a digital controller operating in the discrete time domain.

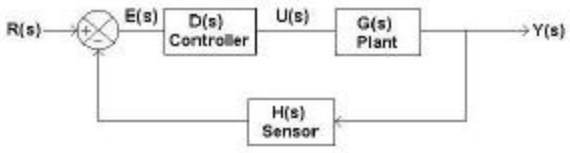

Figure 1 - Classic Closed-Loop Control

This experiment brought controls into the physical realm so that students could see a PID control system manipulate the performance of a real-world device. The plant chosen for the project was a DC motor coupled to an identical DC motor to create a motor/generator system that would provide an analog voltage output for any given analog voltage input. An additional optical sensor was added for purposes of determining the RPM of the motor given a known input.

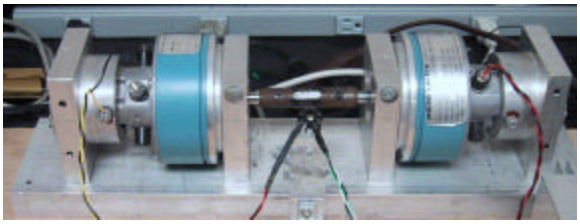

Figure 2 - DC Motor/Generator (Plant)

The controller had to support two inputs (the driving signal and the feedback signal) and one output (the motor control signal) and had to be reprogrammable. There were many different devices used by the class, but this paper will focus on two particular implementations—an MC68HC12 microcontroller by Motorola and a National Instruments implementation using LabVIEW 5.1 and a PCI-1200 data acquisition (DAQ) card.

To aid in the solving of the complex systems of equations, various software packages were used including one that was developed internally. The primary software program used throughout the project was a "real-time" simulation tool written by Justin Ewing, an undergraduate student. The program was designed using National Instrument's LabVIEW Virtual Instrument and was capable of dynamically plotting the output of a system for different PID constants. In addition to the program written by Mr. Ewing, various other simulation and computational programs were used, including Matlab, Simulink, Maple, Curve Expert 1.4, and other LabVIEW tools to help find fourth order roots and solve for coefficients in simultaneous equations.

Characterization of the Plant

To achieve tight control of the plant (in this case a motor), there must be a characteristic equation that describes the operation of the motor over its range of inputs.

Because the motor is a mechanical device, it was assumed that it was overdamped and more importantly, that it could be represented by a second order equation. A second order model is sufficient because it adequately models both the rising and steady state characteristics of the motor response, and it is a comfortable medium between trivial and tedious calculations. Since the intent of the class was to learn how to implement a PID control system, it was decided that modeling the plant with an equation higher than second order would detract from the important principles and cause students to become overwhelmed with the calculations.

Two types of data were collected from the plant, the first was the representation between input voltage and output speed of the motor. The second was the motor's response to a step input. Capturing this data was straightforward using a simple LabVIEW data acquisition program to read the given input and resulting output for a given set of test input amplitudes. The following figure describes the RPM output of the motor for increasing positive input voltages. This relationship turned out to be linear which allowed a generic characteristic equation to be developed—one that was not dependent upon input values.

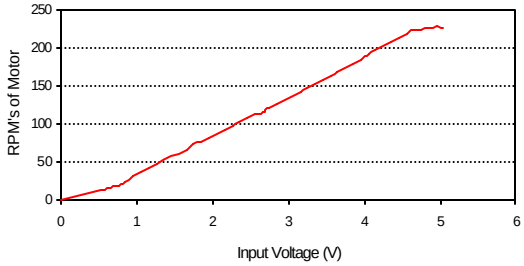

Figure 3 - Linearity between Input and Output

The second data set collected was the motor's response to a step input. Two different methods were used to acquire the data from the motor's response to a step input. One group used LabVIEW to apply the step input and then measure the resulting output. The other group used LabVIEW strictly as a data collection device and used a manual push-button switch to apply the input step. The logic used to support the use of the switch was that there would be an associated latency induced by the DAQ card having to perform multiple functions at the same time. After comparing both sets of data, it was found that any delay caused by the sampling rate of the DAQ card was insignificant and did not produce any irregularities in the data. The following diagram details the input step and corresponding motor characteristic response.

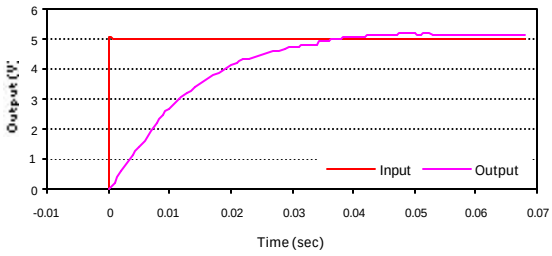

Figure 4 - Input Setup vs. Motor Output

Notice that it takes about 45 ms for the motor to reach its steady state value. During the initial characterization of the motor, the generator actually produced an output voltage greater than the motor input. To understand why this happened, one must examine the laboratory setup in more detail. Because the motor required a high current, it could not be driven directly from the DAQ card. Therefore, the voltage output of the DAQ card was buffered by a voltage-to-current converter that supplied the motor through 60A power supplies. This conversion introduced an error into the system, and as a result, the motors were

driven with a slightly higher voltage than was provided by the DAQ card. Fortunately this error did not affect the final PID experiment.

The data was collected at a rate of one sample every millisecond until the motor reached its steady state output value plus about 20ms to ensure that the motor had settled—about 65 ms. The resulting first order exponential expression that describes the characteristic response of the motor was derived using Curve Expert 1.34, a curve fitting program. The resulting first order equation was:

$$V_{out} = 5.45(0.96 - e^{-75.79t})$$

Again, notice that when given a +5V step input the generator output actually is greater than 5V, this is due to the power conversion error mentioned previously.

The first order equation can model the rising response of the motor, but to model the steady state response of the system, a second order polynomial fit was used. Instead of performing this curve fit, a different method was used. The "LabVIEW_PID" program was used to approximate the open-loop second order equation and hence, one could determine the damping ratio,ξ_n and the undamped natural frequency, ω_n to described the motor's response to a step input. All approximations were performed visually by comparing a Microsoft Excel plot of the actual data and modifying the LabVIEW_PID ξ_n and ω_n values until the response curves "appeared" the same.

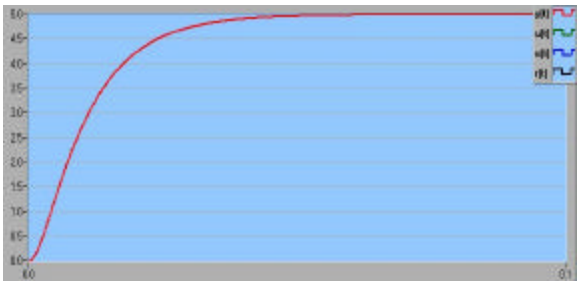

Figure 5 - Plot of Second Order Equation

Using this visual method, the authors arrived at the following second order characteristic equation for describing the motor's response to a +5V step input:

$$Y(s) = \frac{40000}{s^2 + 550s + 40000}$$

Discrete PID Control

The project was then segmented into three distinct sections, each implementing a different method of motor control: Proportional-Integral, Proportional-Derivative, and Proportional-Integral-Derivative. These are described below with both the continuous equation as a reference, along with the implemented discrete equation used by the controller to determine the necessary control signals. Two separate implementations of the controller were used; one was based on a LabVIEW data acquisition system while the second was based on a MC68HC12 microcontroller.

PI Control

Theoretically, Proportional-Integral (PI) control of the motor would provide a system that is more responsive. Unfortunately, this is often too responsive and thus produces overshoot and ringing in the final output. The following equations describe the continuous transfer function and discrete control equations implemented for PI control of the motor.

$$\frac{Y(s)}{R(s)} = \frac{40000(K_i + K_p s)}{s^2 + 550s^2 + 40000(K_p + 1)s + 40000K_i}$$

$$u_{k0} = e_k * ((k_i * T) + k_p) - e_{k-1} * k_p + u_{k-1}$$

To define the terms in the above equation: 'u' is the output of the system to the plant, 'e' is the error signal input to the controller, T is the sampling time of the system, k is the present value and k-1 is the previous value.

The following figure is the output of the motor given a 3V step with constant values for K_p and varying values of K_i.

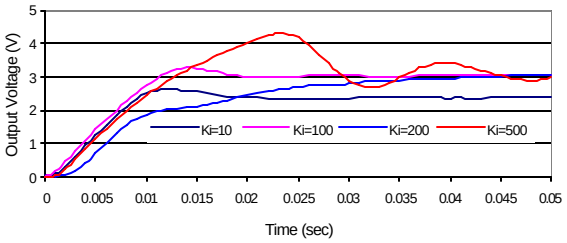

Figure 6 - Proportional-Integral Control

Notice that the system becomes more responsive with increasing values of K_i.

PD Control

Using just Proportional-Derivative (PD) control in the model theoretically provides for a system that settles out faster than with just Proportional or Proportional-Integral control. The following equations describe the continuous transfer function and discrete control equations implemented for PD control of the motor.

$$\frac{Y(s)}{R(s)} = \frac{42000(K_d s + K_p)}{s^2 + (550 + 42000K_d)s + (40000 + 42000K_p)}$$

$$u_{k0} = e_k * ((k_d/T) + k_p) - e_{k-1} * ((2k_d/T) + k_p) + e_{k-2} * (k_d/T) + u_{k-1}$$

The following is a plot of the motor's response to an input step of 2V and using constant K_p with varying K_d.

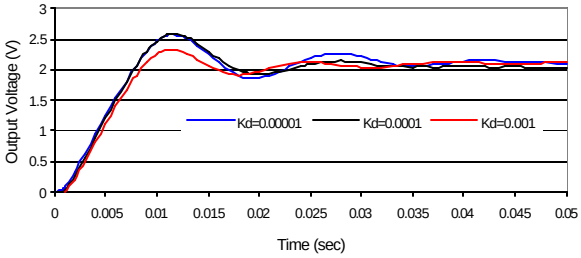

Figure 7 - Proportional-Derivative Control

Notice that for increasing values of K_d, the system settled out quicker. Only very small values of K_d were possible due to the extreme effects of Derivative control—it only takes a small amount of K_d to provide a large effect on the output.

PID Control

Finally, PID control was implemented. As expected, PID control provides for the tightest closed-loop control as compared to any other combination. The following equations are the continuous transfer function and the discrete control equations implemented.

$$\frac{Y(s)}{R(s)} = \frac{40000(K_d s^2 + K_p s + K_i)}{s^3 + (550 + 40000K_d)s^2 + (40000 + 40000Kp)s + 40000Ki}$$

$$u_k = e_k\left[\frac{K_d}{T} + K_p + K_i T\right] - e_{k-1}\left[\frac{2K_d}{T} + K_p\right] + e_{k-2}\left[\frac{K_d}{T}\right] + u_{k-1}$$

Up to this point in the project, the data from the microcontroller and LabVIEW implementations were consistent with each other and provided about the same quality results. In the final implementation of the PID controller, the two groups achieved quite different results. The final data from the microcontroller group was not what they had expected. Although it did maintain tight

control of the motor, it did not succeed in increasing the response of the motor. The constant values that provided the following best response plot are: K_p=2.5, K_i=80 and K_d=0.001.

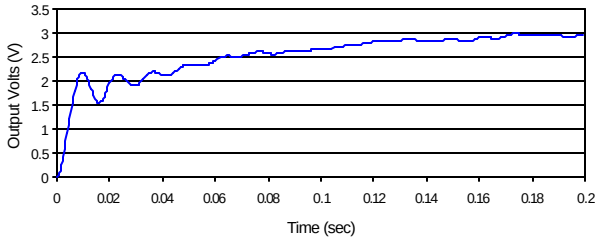

Figure 8 - MC68HC12 PID Output

The LabVIEW implementation, on the other hand, was more successful in achieving the project goals. The following is a plot of the LabVIEW results. The best constant values for this plot were: K_p=7.5, K_i=200 and K_d=0.001.

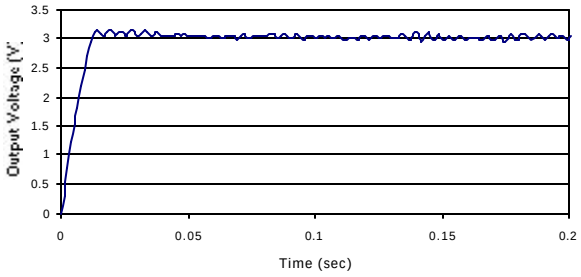

Figure 8 - LabVIEW PID Output

Notice the increased response and tighter control with the LabVIEW implementation compared to the MCHC12.

Intelligent Controllers

As mentioned previously, the authors chose very different controllers for this project—one chose the Motorola MC68HC12 and the other used LabVIEW 5.1 in conjunction with a PCI-1200 DAQ card. Each implementation had both advantages and disadvantages as compared to the other.

Disadvantages

The HC12 microcontroller offered a few limitations not noticed in the LabVIEW version. Possibly the biggest limitation was the fact that it could only be programmed by a low to mid level programming language (Assembly, C). Although it reduced the total amount of overhead in the program, it increased the possibility for programming errors. The group that used a LabVIEW-based system found very little difficulty in getting correct results from the outset. Possibly the biggest limitation of the HC12 was its processing power. The HC12 can only operate with 16-bit values and as a result the operations would often 'rail' out and provide erroneous data as it cycled between the high and low rails of the system. This was due to the extreme sensitivity of the control system to numerical error. LabVIEW, on the other hand, has the ability to operate with 32-bit values and thus can provide a much better result and eliminate the 'railing' observed with the HC12. Another large difficulty experienced by both groups was the sampling time. Although the HC12 had less overhead, it also had less processing power so the sampling rate, as measured by toggling an external port bit, was about 1.2ms. While the groups using the HC12 microcontroller found that the sampling time could be reduced to 0.2ms if the floating point calculations were eliminated, they also found that the decrease in accuracy was unacceptable. Surprisingly, the LabVIEW implementation achieved a 1.3ms sampling rate even with the overhead found in such a high-level programming environment. This rate was sufficient to implement the controller.

Advantages

It appears that most of the advantages are with the LabVIEW implementation, but there were a few that remained with the

HC12. One advantage for the HC12 was price. The HC12 group used an evaluation board that cost $150. They also used a copy of the Introl C Compiler and a basic personal computer (PC) with a serial port. Another advantage of the microcontroller system is that it is similar to the type of system found in industry.

The LabVIEW setup requires a moderate speed Pentium computer with a PCI-1200 DAQ card and LabVIEW software license. Even if the cost were not an issue, setting up a PC to run the PID control system would consume too much space for most industrial applications.

However, the LabVIEW implementation has extraordinary benefits for design, research, and educational purposes. First, the actual program can be done *very* quickly, and it is easy to modify. Second, the ability to add probes and graphs at different points in the program allows the user to take a deeper look into what is actually happening in the control system. Finally, tuning the PID control system is as easy as clicking the mouse. Conversely, tuning the PID system using a microcontroller requires editing the source code, recompiling, and re-downloading the program to the microcontroller. This can quickly become a tedious task.

Conclusions

This project proved to be an exciting learning experience for each member of the class as well as for the instructor. It taught the students how to characterize a plant using a second order model and the importance of using discrete time domain calculations in a controller-based environment. It also allowed the students to experimentally observe the effects of varying the PID coefficients. Finally, this project not only showed that it is possible to provide reliable, closed-loop control of a second-order plant, but that PID control systems can actually improve the performance of a plant. In order to make this project viable over several semesters, a new plant system that has a variable load is being developed. In this manner, the system being controlled can be changed from semester to semester.