Portable Controls Experiments

by

Richard Winston Larson

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Engineering as Recommended by the
Department of Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2012

Author .... [illegible]
Department of Mechanical Engineering
May 18, 2012

Certified by .... [illegible]
David L. Trumper
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .... [illegible]
John Lienhard
Samuel C. Collins Professor of Mechanical Engineering
Undergraduate Officer

Portable Controls Experiments

by

Richard Winston Larson

Submitted to the Department of Mechanical Engineering
on May 18, 2012, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Engineering as Recommended by the Department of
Mechanical Engineering

Abstract

Experiments for controls classes like MIT's 2.004 require large lab setups and expensive equipment such as oscilloscopes and function generators. We developed a series of controls experiments based on National Instruments' myDAQ platform. These experiments, which are small enough to fit on a single PCB board and weigh less than a pound, allow students to work on controls labs at their own pace wherever they please, increasing the ease of learning. We designed and prototyped a double integrator experiment and a DC motor experiment. We implemented the control software in NI LabView, and we have produced accompanying documentation. We will make the circuit layouts, controller software, and lab documentation available to the public through the NI myDAQ website. Other schools will be able to use our designs in their courses.

Thesis Supervisor: David L. Trumper
Title: Professor of Mechanical Engineering

Acknowledgments

Professor Trumper has been a fantastic advisor and mentor. I have taken multiple courses from him during my undergraduate years, and they have been the best I have taken at the Institue. His teaching has helped me find a passion for controls, and he has helped me focus my curiosity and sharpen my ability. I deeply appreciate his patience and hard work in helping me be successful.

Roberto Melendez was instrumental in the completion of this project. He developed with me the circuitry and control software for the double integrator experiment, and we worked together to write the integrator write-up and test and refine the experiment. I thank Harrison Chin for helping troubleshoot the hardware and software for the motor control experiment.

We gratefully acknowledge National Instruments for providing funding for this research and donating myDAQ systems and LabView software.

I would like to thank the many anonymous donors who have made my MIT education possible. Their generosity has enabled me to fulfill my dream of attending MIT and becoming an engineer. It means the world to me.

I have been mentored and guided by many excellent professors and teaching assistants during my undergraduate career, and I am grateful for their patience, perseverance, and vision. In particular, I want to thank my undergraduate advisor Franz Hover, my UROP advisor Kamal Youcef-Tuomi for having faith in a sophomore wanting to do robotics, and Doug Hart for showing me what engineering is all about.

Most deserving of my gratitude is my family. My parents Allison and Gordon Larson have handed me the world on a silver platter. They have always been there for me, and have supported me while I chase my dreams. Without them, nothing I have done would have been possible. I am eternally in their debt, and love them dearly.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Controls Laboratory Experiences

The essence of engineering is the ability to apply physical and mathematical knowledge in solving problems. To be successful in doing so, experience in using knowledge learned in the classroom is absolutely necessary in engineering instruction, making the laboratory a key element of education. Controls and mechatronics, in particular, require intuition and experience when solving problems. Students gain intuition through experimentation with and observation of real physical systems reacting to changes in the environment. They must be able to modify controller design and see the effects on the system. Students learn best when they work at their own pace and have opportunities to tinker.

Unfortunately, controls experimentation in an educational setting is generally limited to being conducted in a well-equipped campus laboratory. Complex and expensive equipment are a barrier to entry, and include function generators, oscilloscopes, DAQ equipment, and the lab setups themselves. Some of this equipment is shown in an MIT lab in Fig. 1-1. The need for equipment restricts experimentation time to assigned lab hours, limiting the possibilities for learning and experimentation. The ideal laboratory experience would be one in which the student could work outside of the laboratory, spending as much time as necessary to fully understand the lab work. In this approach, assigned lab times can be used to refine students' understanding of

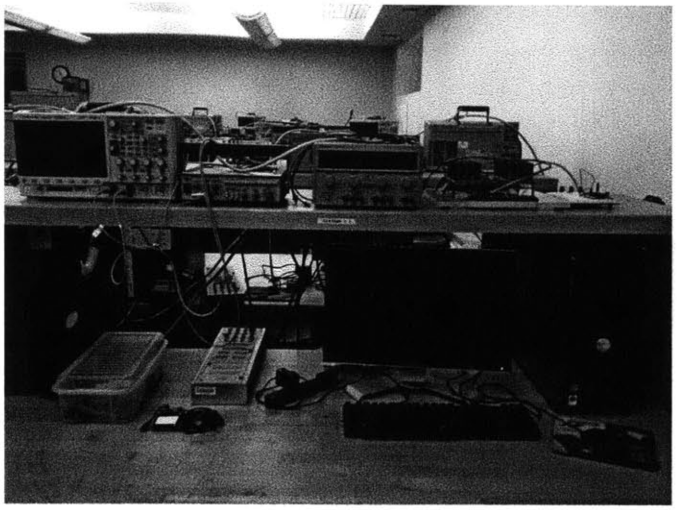


Figure 1-1: Large and expensive equipment may only be used in a laboratory setting, limiting learning opportunities.

the material, or for helping students through difficulties.

1.1.1 National Instruments myDAQ

The myDAQ system, built by National Instruments, makes it possible to take controls experimentation out of the laboratory, enabling students to conduct labs in their dormitories, libraries or any space with access to a computer. The myDAQ, shown in Fig. 1-2, is a small ($147 \times 89 \times 23$ mm, 173 g) and relatively simple DAQ solution that is connected to a computer by USB and which runs on NI LabView software. The myDAQ's small form factor, rugged construction, and wide range of capabilities allow it to not only be part of a portable controls lab, but also to facilitate a variety of lab setups. With appropriate software, the myDAQ may simultaneously act as a function generator, oscilloscope, and controller platform. By using myDAQ, students can spend more time with controls experiments, and they will gain better intuition for physical systems and how controllers interact with them.

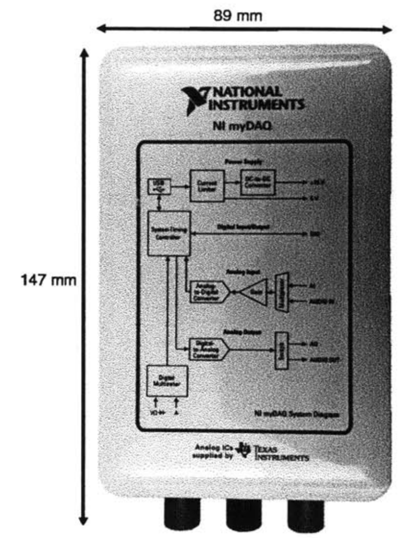


Figure 1-2: National Instruments myDAQ, a portable and inexpensive DAQ solution.

1.1.2 Experiment-on-a-PCB

We developed two simple controls experiments for use with the myDAQ device. These experiments each fit on a single PCB which is approximately the size of the myDAQ itself and which can connect directly with the myDAQ I/O connector. The designs highlight the portability of this new lab setup. Everything the student needs for a complete controls lab experience weighs less than a pound and can fit in a gallon-sized ziplock bag.

We designed two physical plant setups. First we studied a series of operational amplifiers which act as single or double integrators. Our setup provides a reference input as well as disturbance and noise inputs. We implement proportional, proportional-derivative, and proportional-integral control digitally through the myDAQ.

The second plant we developed was a voltage-controlled DC motor attached to a small rotational inertia. The controller acts as a cruise-control for the DC motor. As in the integrator experiment, we implemented proportional, proportional-integral, and proportional-integral-derivative control digitally through the myDAQ. The experiment board includes the DC motor, a motor driver, a battery, and a tachometer to measure the speed of the flywheel.

Both experiments each fit on a single PCB equipped with a connector which allows it to be plugged directly into the myDAQ. An emphasis was placed on keeping the experiments as simple as possible.

1.1.3 Open Availability

Our goal was to develop the hardware and software necessary to allow students to conduct controls experiments anywhere using portable equipment. Because of the unique nature of the myDAQ and the accompanying experiments developed by this project, we wrote reports documenting the hardware, software, and their use. Our reports explain the hardware and software setups, theoretical backgrounds on the plants and controllers, the equipment involved, and a user's guide to using the labs. They also include problem sets and laboratory work for the students to complete using the setup.

We intend that these lab experiences will be useful in controls classes at many universities. We will make the reports, LabView software, and circuit board layouts available on the Internet via the National Instrument's website. Professors and students can use and modify the experiments freely for their curriculums or self-instruction.

Chapter 2

Physical Plant Dynamics and Classical Control Theory

We explain the physics underlying the implementation of the integrator circuit and DC motor experiments, as well as basic continuous and discrete control theory. Our explanation is a basic overview of the physical principles of the physical plants. For a deeper understanding of the modeling of physical systems and classical controls, we recommend a textbook such as that by Franklin, Powell, and Emami-Naeini [1].

2.1 Classical Control Theory

2.1.1 Negative Feedback

Controls systems attempt to make a system behave as desired. A controller acts on the error between the desired outcome and the measured value of a system output to be controlled. Reducing error by acting on its value is known as negative-feedback control, which is often described visually using block diagrams.

The experiments we developed are based on negative feedback control, and Fig. 2-1 shows an example control block diagram. Here we assume linearity and thus a transfer function representation is valid.The error is the difference between the current state of the system and the reference signal. The control effort is the product of the error and

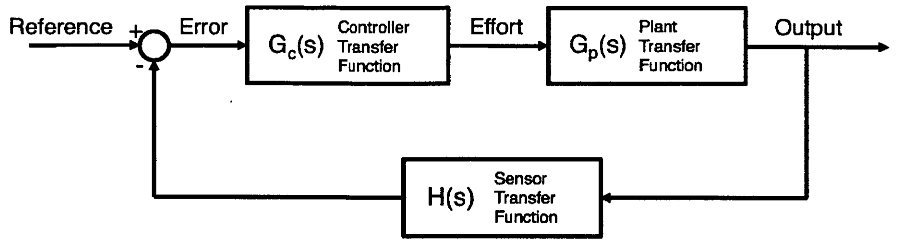

Reference
+
-
Error
$G_c(s)$ Controller Transfer Function
Effort
$G_p(s)$ Plant Transfer Function
Output
H(s) Sensor Transfer Function

Figure 2-1: The basic control block diagram for negative feedback control.

a controller transfer function G_c. The control effort acts on the system, also called the plant, which is represented mathematically by a plant transfer function G_p, and the result is the output of the system. A sensor, with sensor transfer function H, is in the feedback path and its output reports the current system state.

2.1.2 Model and Controller Representations

Engineers study system behaviors and design controllers using models and controller representations.

The root-locus method plots transfer functions in the complex plane, and represents system singularities as poles and zeros. As system gain is increased, the root locus shows the movement of poles and zeros. Root locus allows engineers to understand oscillatory behavior and settling times, as well as evaluate stability.

Bode plots represent the phase and magnitude of the response of a system to a sinusoidal input. Sinusoidal inputs not only capture the full behavior of a system, but are common in engineering applications. Bode plots evaluate system stability and the effects of controllers on system behavior.

2.1.3 Proportional Integral Derivative (PID) Control

Control algorithms calculate control effort based on the error between the desired and actual system states. Proportional integral derivative (PID) control is one of the most commonly used algorithms in classical control theory. As its name suggests, the control effort is a combination of a term proportional to the error, a term proportional to the integral of the error (or, in other words, the accumulated error), and a term proportional to the derivative of the error (or the rate at which the error is changing). We can express a PID controller mathematically as

$$G_c(s) = K\left(1 + \frac{1}{T_i s} + \frac{T_d s}{T_f s + 1}\right), \tag{2.1}$$

where K is the proportional gain, T_i is the integral time, and T_d is the derivative time. A pure derivative controller cannot be implemented in a real system, as its gain grows unbounded as the system frequency approaches infinity. Hence we include the high-frequency term T_f, bounding the magnitude of the system response at high frequencies. As is common practice, let $T_f = 0.1T_d$.

We determine the values for K, T_i, and T_d based on the desired performance of the system. We use root locus and Bode analysis to find optimal pole and zero placement. Controller design is a complicated task requiring skill and finesse, and the interactions of controller gains are complex. However, generally speaking, in a first order system, increasing proportional gain decreases response time, integral control eliminates steady-state error, and increasing derivative gain reduces system overshoot.

2.1.4 Noise and Disturbance Signals

Noise and disturbance signals are common non-idealities. The block diagram in Fig. 2-2 show their addition to a control loop.

Noise signals simulate the presence of sensor noise in the system. Noise affects the error measurement which is acted upon by the controller. Generally, a higher-bandwidth controller incurs larger noise-driven errors.

Disturbance signals simulate the presence of disturbance voltages or forces which act on the plant in addition to the control effort. Here, conversely, a higher bandwidth controller generally can better suppress errors due to disturbances

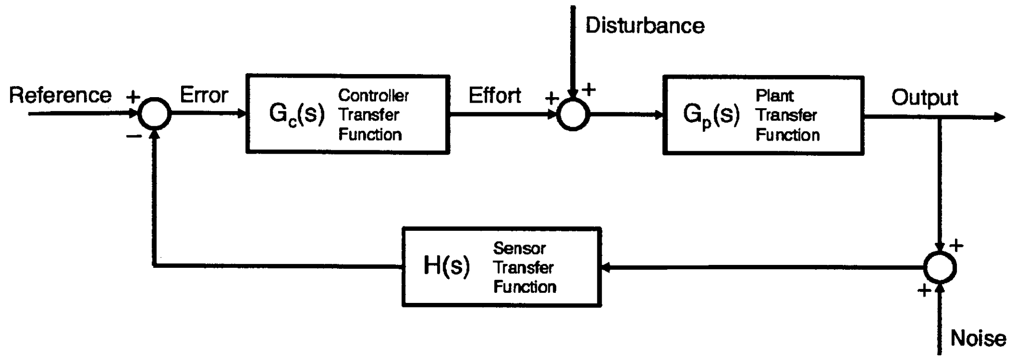

Disturbance
Reference
Error
$G_c(s)$ Controller Transfer Function
Effort
$G_p(s)$ Plant Transfer Function
Output
H(s) Sensor Transfer Function
Noise

Figure 2-2: A control block diagram including measurement noise and disturbance inputs.

2.2 Integrator Circuit Plant

A basic integrator is an ideal circuit for practicing basic control. While it is not a complex system, it offers a robust platform for understanding the principles of control theory, and developing an intuition for control systems. We used the operational amplifier in designing and building our integrator circuitry.

2.2.1 Operational Amplifiers

The operational amplifier is an integrated circuit configured as a differential amplifier. That is, the output of an op-amp is the difference between its two input terminals is multiplied by a very high gain. Here we assume an ideal op-amp, a simple approach which is sufficient in developing our integrator circuits.

We show the op-amp circuit symbol in Fig. 2-3. There are two inputs, v_+ and v_-, and an output v_{out}. The op-amp is a powered device, with voltage supply inputs V_{cc+} and V_{cc-}. Our circuit holds these pins at ± 15 V.

The op-amp may be modeled using two basic rules described below. More precise models can be found for instance in Roberge [2].

1. The output is the difference between the inputs multiplied by a very high gain

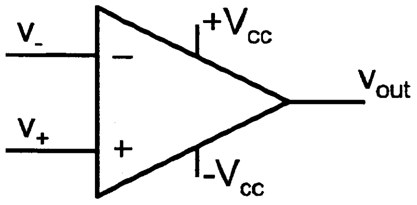


Figure 2-3: An operational amplifier circuit symbol.

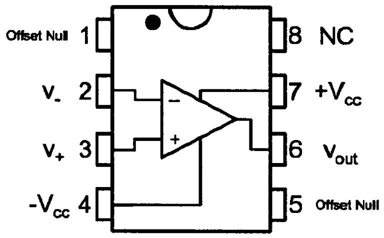


Figure 2-4: Operational amplifier pin-outs.

A. Ideally, $A = \infty$. We express the relation of output to input as

$$v_{out} = A(v_+ - v_-). \qquad (2.2)$$

We assume that the op-amp output acts as an ideal voltage source. In the limit of $A \to \infty$, we assume that $v+$ and $v-$ are at the same potential.

2. Ideally, an op-amp has infinite input impedance. That is, no current flows into the op-amp through its inputs.

The pin-outs for a 741, which we show in Fig. 2-4, include the input pins, output pin, and voltage supply pins for powering the amplifier. Pins 1 and 5, used for offset nulling, are not used in our circuit.

2.2.2 Op-amp Integrator

We may arrange a resistor, a capacitor, and an op-amp to build an integrator circuit, as seen in Fig. 2-5. Remembering the second rule of op-amps (no current flows into the op-amp), and using Kirchoff's current law, we know that

$$i_{in} + i_{out} = 0. \tag{2.3}$$

Using Ohm's Law

$$v = iR, \tag{2.4}$$

and the constitutive equation for a capacitor

$$i_C(t) = C\frac{dv_C(t)}{dt}, \tag{2.5}$$

and since we assume $v+ = v- = 0$, we find that

$$\frac{v_{in}(t)}{R} + C\frac{dv_{out}(t)}{dt} = 0. \tag{2.6}$$

Taking the Laplace transform of both sides, assuming zero initial conditions, we find

$$\frac{V_{in}(s)}{R} + CV_{out}(s)s = 0. \tag{2.7}$$

We rearrange Eq. 2.7 to obtain

$$\frac{V_{out}(s)}{V_{in}(s)} = -\frac{1}{RCs}. \tag{2.8}$$

This is the input/output plant transfer function for the integrating op-amp circuit, and is of the form of an integrator with an inverting gain of $-1/RC$. We can place integrators in series to make a double integrator.

Two other op-amp circuits were used in the experiments developed, namely the inverting amplifier and the summing amplifier. The transfer functions for these amplifiers may be derived using the rules outlined earlier. A full derivation is left as an exercise for the reader, and the transfer functions are simply stated here. The circuit diagram for an inverting amplifier is shown in Fig. 2-6, and it is governed by

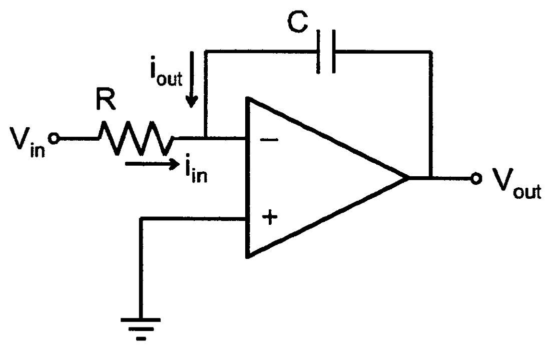


Figure 2-5: A single integrator op-amp circuit.

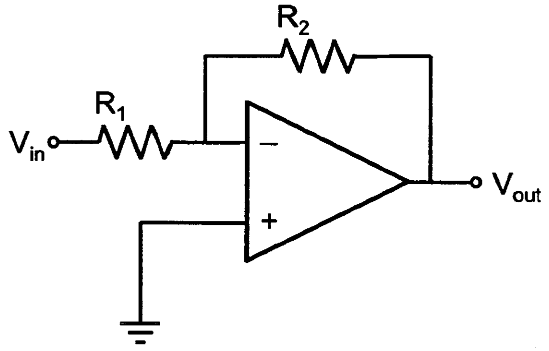


Figure 2-6: An inverting amplifier op-amp circuit.

the equation

$$\frac{V_{out}}{V_{in}} = -\frac{R_2}{R_1}. \tag{2.9}$$

The circuit diagram for a summing amplifier is shown in Fig. 2-7, and its governing equation is

$$V_{out} = -R_f\left(\frac{V_1}{R_1} + \frac{V_2}{R_2}\right). \tag{2.10}$$

2.3 DC Motor Plant

Brushed DC motors are common actuators in many modern robotic systems. An ideal DC motor, like the op-amp integrator, is a simple physical plant, but highlights many important controls concepts. DC motor are advantageous because students

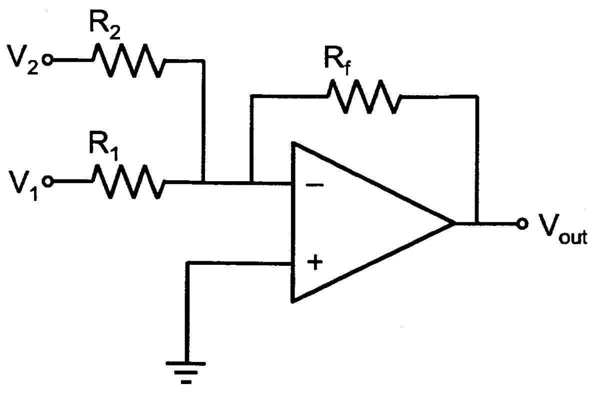


Figure 2-7: A summing amplifier op-amp circuit.

can interest with them. If the motor is not too large, students can see and feel the changes in the system due to modifications to the controller, and introducing disturbance forces is as simple as grabbing the motor shaft with a finger. We base our theoretical development on Rowell [3].

2.3.1 DC Motor Physics

The actuating force in a DC motor can be ascribed to the Lorentz force. A wire carrying current in the presence of a uniform magnetic field experiences a force

$$F = ilB, \tag{2.11}$$

where i is the current in the wire, flowing the conventional direction, and l is the length of the wire in the magnetic field B. Here, we assume that i, B, and F are mutually perpendicular.

A motor takes advantage of this force, placing a series of current-carrying windings wrapped around a core in a magnetic field, as shown in Fig. 2-8. The torque on the windings due to the Lorentz force is

$$\tau = nilBr, \tag{2.12}$$

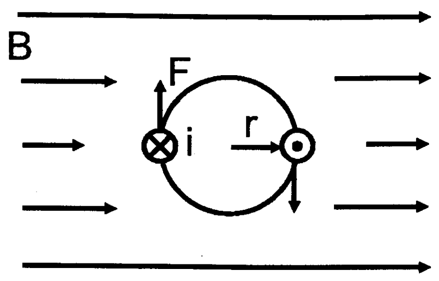


Figure 2-8: The basis for a DC motor based on the Lorentz force.

where n is the number of windings and r is the radius of the armature. We can simplify Eq. (2.12) to find

$$\tau = iK_T, \tag{2.13}$$

where K_T is the torque constant of the motor in N-m/A.

As the armature rotates at velocity ω, the magnetic field generates a voltage in the windings

$$\varepsilon = n\omega lBr. \tag{2.14}$$

The voltage ε, called the *back emf*, is opposite in sign to the voltage applied to the motor to induce a current and cause it to rotate. We can also express the back-emf ε as

$$\varepsilon = \omega K_V, \tag{2.15}$$

where K_V is the motor's armature constant in rad/s/V. In SI units, $K_T = K_V$, which is appropriate as seen by examining Eq. (2.12) and (2.14). From this point on, the we will refer to the motor constant as $k_m = K_T = K_V$ to avoid confusion with other gains used in designing the controller.

When a motor has reached a steady state velocity, the voltage applied to the motor and the back emf are nearly equal. If the armature has a resistance R, and the inductance in the motor windings is ignored, the voltage in a DC motor circuit as shown in Fig. 2-9 is

$$V = iR + \varepsilon. \tag{2.16}$$

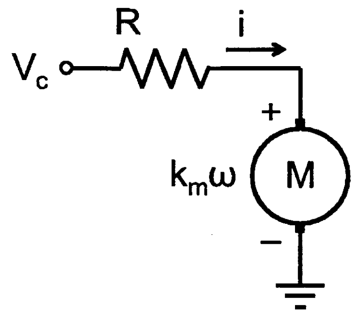


Figure 2-9: The electrical circuit powering the DC motor actuator.

We will use this equation when designing a control system for a DC motor.

2.3.2 DC Motor Physical Plant

The actuator physics of the DC motor are only half the story. We also need to model the mechanical properties of the motor. Figure 2-10 shows a representation of the motor's mechanical properties. Through a gearbox with gear ratio N, the motor actuator drives a rotational inertia J. The inertia represents the combined inertia of the motor core, the tachometer core, the motor shafts, and the coupling which attaches the shafts to each other (the coupling is also referred to as the flywheel). A linear damping term B represents damping in the system, the most significant contribution being the back emf of the motor. We can write the differential equation for a linearly damped rotational inertia driven by a torque τ as

$$J\dot{\omega}(t) + B\omega(t) = \tau(t), \tag{2.17}$$

where ω is the rotational velocity of the inertia. Transforming to the Laplace domain using Ω as the velocity of the flywheel and T as the torque from the motor, we find that

$$\Omega(s)\,(Js + B) = T(s). \tag{2.18}$$

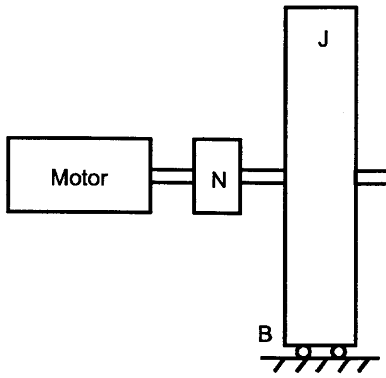


Figure 2-10: A simplified representation of the the mechanical motor model.

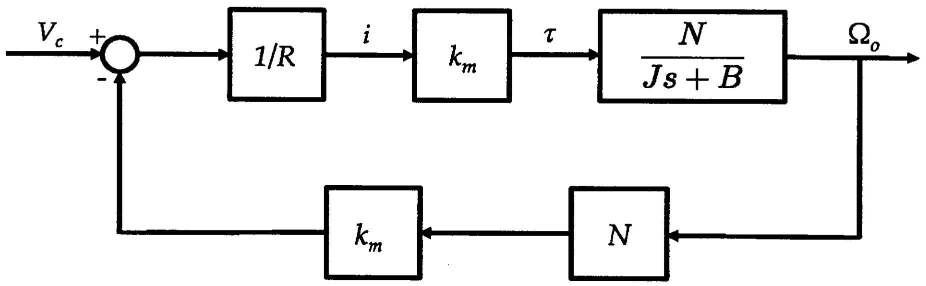


Figure 2-11: The control block diagram used to find the plant transfer function for the DC motor system.

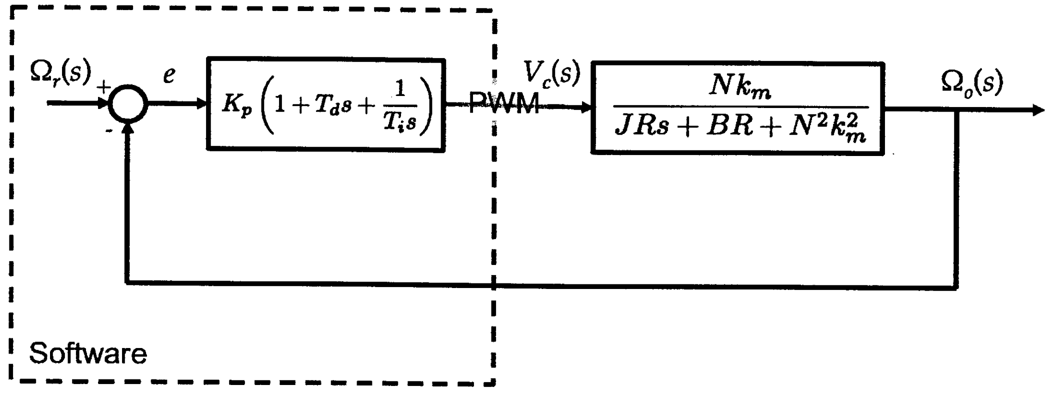
We express mechanical plant transfer function, including the gearbox with ratio N, as

$$\frac{\Omega(s)}{T(s)} = \frac{N}{Js+B}. \tag{2.19}$$

We show in Fig. 2-11 a control block diagram of the motor actuator, motor circuit, and mechanical plant. The motor back emf completes the loop. Using block diagram algebra, we simplify the control loop into a single transfer function, representing the motor system's use of the input voltage from the controller V_c to turn the rotational inertia at rotational velocity Ω_o. We calculate the transfer function to be

$$\frac{\Omega_o(s)}{V_c(s)} = \frac{Nk_m}{JRs + BR + N^2k_m^2} \tag{2.20}$$

Because these terms are difficult to find using data sheets, we measured the values



$\Omega_r(s)$
+
-
e
$K_p\left(1+T_d s+\frac{1}{T_i s}\right)$
PWM
$V_c(s)$
$\frac{Nk_m}{JRs+BR+N^2k_m^2}$
$\Omega_o(s)$
Software

Figure 2-12: The control block diagram for the entire DC motor experiment, including the motor model and PID controller.

of J, B, k_m, and R experimentally, as described later in this paper. We show the overall control system block diagram for the DC motor experiment, including a model of the motor system and a PID controller, in Fig. 2-12.

Chapter 3

Hardware Construction

The purpose of the experimental hardware is to provide controls experiments which are compatible with the myDAQ device. We explain the circuit layouts and construction of the experiments, including pin connections to the myDAQ.

3.1 Integrator Plant

The single and double integrator plant uses three 741 op-amps as shown in Fig. 3-1. Switches are used to change operating parameters, including single/double integrator mode and noise/disturbance mode.

Power is provided to the op-amps from the +15/-15 V pins of the myDAQ. Small bypass capacitors are placed close to the op-amp power pins.

3.1.1 Single Integrator

The first op-amp always functions as an integrator. The myDAQ provides the controller output voltage, also known as the control effort, at pin AO0. The voltage signal is integrated by the first op-amp integrator circuit, which is shown in Fig. 3-2. Our integrator circuit used $R_1 = 100$ kΩ and $C_1 = 0.1$ μF, making our time constant $\tau = 0.01$ s.

The first op-amp circuit passes its output to a second op-amp circuit. A switch

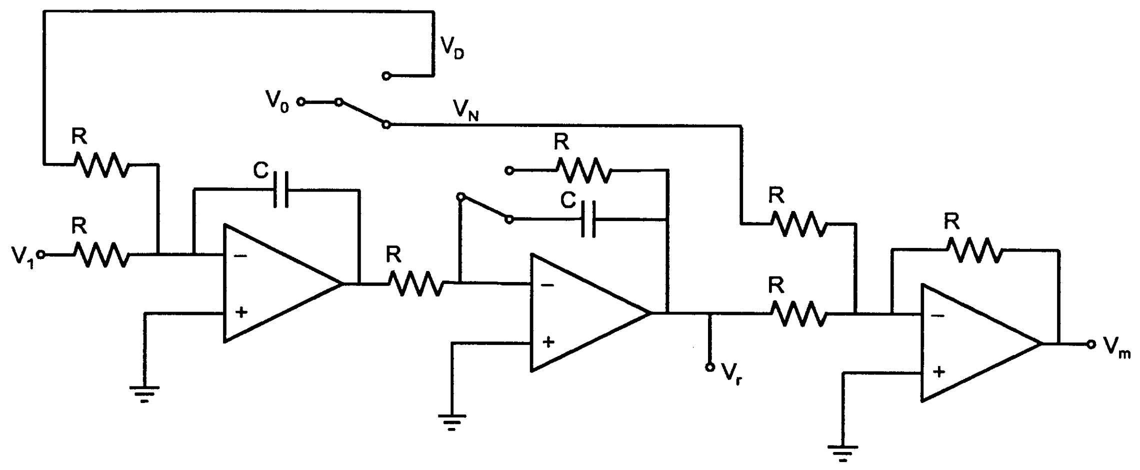


Figure 3-1: The full experimental circuit, with the single/double integrator, summing amplifier, and noise and disturbance inputs.

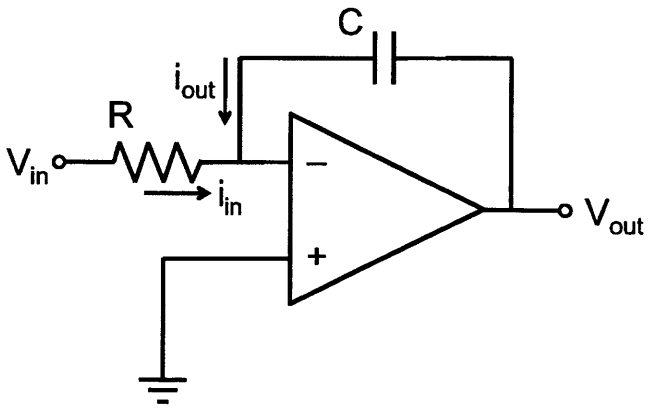


Figure 3-2: The single integrator op-amp circuit.

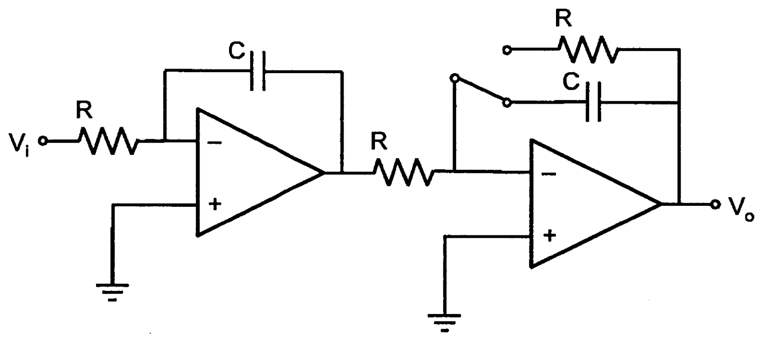


Figure 3-3: The double integrator op-amp circuit. If a single integrator is desired, the switch will turn the second op-amp into an inverting amplifier with unity gain.

determines whether a resistor or a capacitor is in the op-amp feedback loop. If the resistor path is taken, this circuit acts as a inverting amplifier. The gain is unity, as resistors used have the same value (10 kΩ).

3.1.2 Double Integrator

If the capacitor is used in the feedback path of the second op-amp circuit, the circuit acts as a second integrator, with the same time constant as the first integrator. We can change the time constants for both integrator circuits by modifying the resistances and capacitances in the circuits. The double integrator circuit with a switch for the second amplifier is shown in Fig. 3-3.

3.1.3 Noise and Disturbance Signals

The second op-amp circuit passes its output to the third, which acts as a summing amplifier as shown in Fig. 3-1. The summing amplifier adds a noise signal to the output signal of the integrator plant. If a noise signal is to used, this op-amp adds it on top of the output signal, simulating the effects of sensor noise. All the resistors in this circuit are the same value (10 kΩ), making the gain of the amplifier unity. The output of the third op-amp circuit goes back to the myDAQ through pin AI0, and the controller software processes the signal. The full single/double integrator circuit, including the final summing amplifier, is shown in Fig. 3-1.

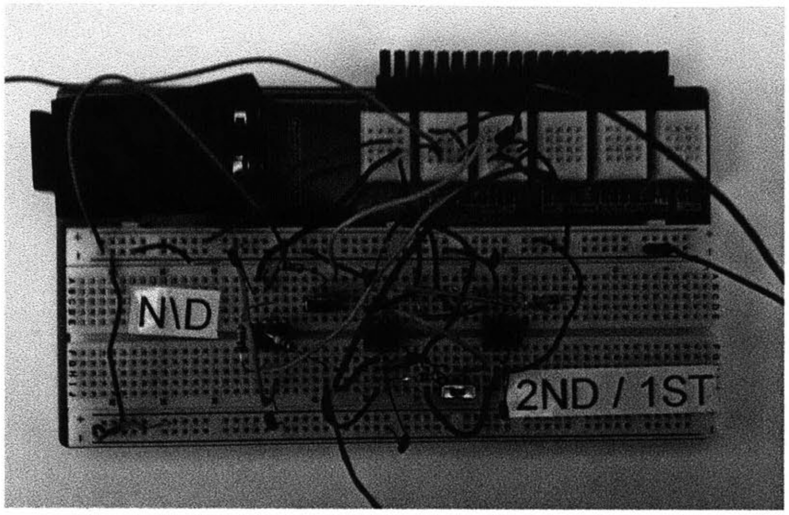


Figure 3-4: The breadboard circuit hardware for the integrator experiment.

The myDAQ may also provide a noise or disturbance signal to the integrator system. The noise/disturbance signal is output from pin AO1. We use a switchto add this signal as a noise or disturbance. If the signal is to be used as a disturbance, we add it before the first integrating op-amp, effectively making the first op-amp circuit a summing-integrator circuit. If the signal is to used to simulate measurement noise, we add it using the summing op-amp circuit (the third op-amp).

A picture of the breadboard circuit hardware is shown in Fig. 3-4. We have labeled the switches for choosing single and double integrators, as well as noise and disturbance inputs.

3.2 DC Motor Plant

The DC motor experiment hardware includes the DC motor, gearbox, flywheel, the motor driver, a tachometer, and a 9 V battery for powering the motor. A motor driver and 9 V battery must be used when powering the motor since the myDAQ is capable of sourcing no more than 500 mW, and will be unable to supply the current necessary to turn the motor.

A PWM signal from the myDAQ, produced at pin AO0, controls the motor

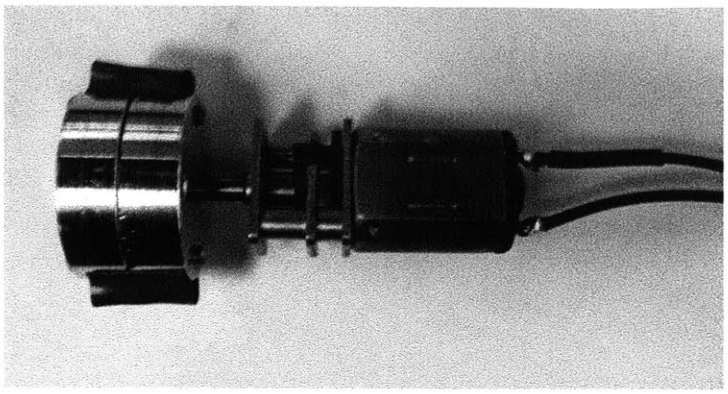


Figure 3-5: The Pololu micro-metal gear motor used in the experimental setup. Also shown is the flywheel and magnets.

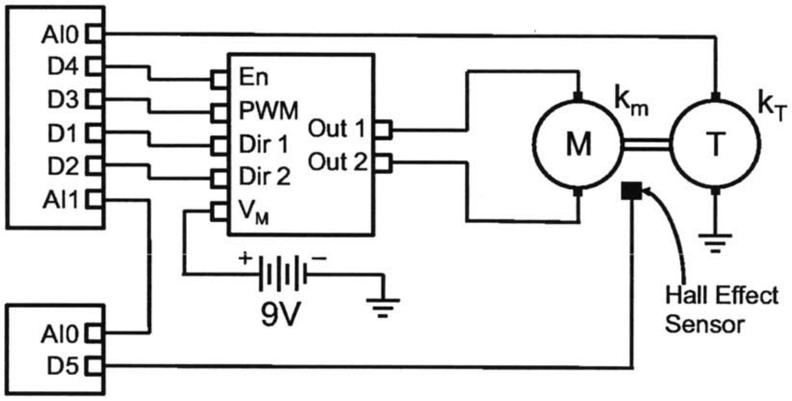


Figure 3-6: The full experimental circuit, with the DC motor, motor driver, and tachometer.

through the motor driver. We used a Pololu micro-metal gear motor, pictured in Fig. 3-5, with a reduction of 5:1.

In our experiment, we control a small DC motor attached to a rotational inertia J through a gearbox with ratio N. There is a linear damping B on the flywheel due to back-emf in the motor. The motor has an armature resistance R, a negligible armature inductance, and a motor constant k_m. A diagram of the physical system is shown in Fig. 3-6. V_c is the voltage generated by the controller (as explained in the software section) and supplied to the motor circuitry as PWM. This section explains the motor circuit hardware setup in detail.

Table 3.1: Physical quantities

Name	**Quantity**	**Value**
Gear ratio	N_g	5:1
Rotational Intertia	J	9.50×10^{-6} Kg-m^2
Rotational Damping	B	3.95×10^{-4} N-m-s/rad
Armature or Torque Constant	k_m	3.57×10^{-3} V-s/rad or N-m/A
Armature Resistance	R_m	7.8 Ω

The values of the physical quantities used in this setup are given in Table 3.1. We experimentally measured the values of the physical system parameters.

Hardware Description

The output of the software controller is the control voltage V_c, which is output from the DAQ at the digital counter pin D3. The control voltage is a pulse width modulated (PWM) signal. The Pololu motor driver processes the PWM signal at pin PWMA. The motor driver chip, based on Toshiba's TB6612FNG, is capable of driving two DC motors at 4.5–13.5 V and peak current of 3 A (1 A continuous). The pin-outs of the motor driver are shown in Fig. 3-7. The motor driver is configured as an H-bridge, and it contains all of the transistors and diodes necessary to protect the motor and the myDAQ. The driver is powered from the myDAQ 5 V rail at pin VCC. All three ground pins are connected to the circuit's common ground.

Voltage from a 9 V battery is routed through the motor driver at pin VMOT. The PWM signal from the myDAQ determines the motor voltage. Since the motor acts as a low-pass filter, the PWM signal acts as a voltage with value proportional to the duty cycle. The motor leads are connected to the motor driver at pins AO1 and AO2.

The motor driver pins AIN1 and AIN2 determine the direction of the motor. AIN1 and AIN3 are respectively controlled by myDAQ pins D1 and D2. The state of the pins must always be opposite each other, i.e. if AIN1 is HIGH, AIN2 will be LOW,

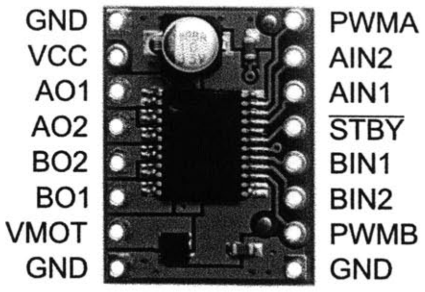


Figure 3-7: A picture of the Pololu Motor Driver Carrier, with accompanying pin-outs.

and vice versa. The STBY pin, connected to myDAQ pin D4, toggles the motor driver on and off. For the motor to operate, STBY pin must be held HIGH.

The speed of the flywheel is measured using two sensors. First, a DC motor acting as a generator senses the direction of rotation. As the motor is turned, it generates a voltage measured by the myDAQ at pin AI0. The software controller uses the generator voltage to determine the rotational direction of the flywheel. The armature constant for this motor $k_s = 38$ V-s/rad was found experimentally.

The output of the generator is noisy. To address this problem, we used a Hall effect sensor to determine the motor speed. In order to ensure that the direction signal from the generator was as accurate as possible, and to decrease the risk of chatter at low rotational speeds, we placed an active low pass filter between the generator output and myDAQ pin AI0. The low pass filter, built using an op-amp circuit, is shown in Fig. 3-8, and has unity gain at frequencies below the cutoff. Using the values of resistor R and capacitor C, we can calculate the cutoff frequency of the filter as

$$f_c = \frac{1}{2\pi RC}. \tag{3.1}$$

To achieve a cutoff frequency of 30Hz, which is well above the operating frequency of the motor, we used $R = 39$ kΩ and $C = 10$ μF.

We measured the speed of the motor with a latching Hall effect sensor. A magnetic field of one polarity passing in front of Hall effect sensor causes it to change its state

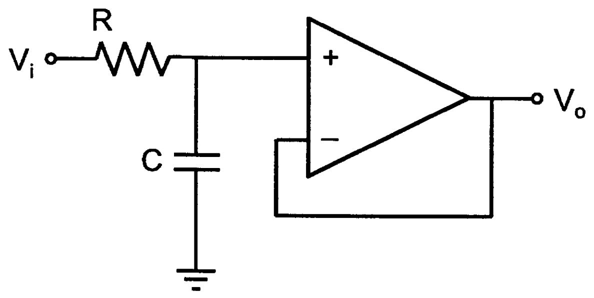


Figure 3-8: An active low pass filter op-amp circuit.

to HIGH. This state will not change until a magnetic field of opposite polarity passes in front of it again.

We attached eight small rare-earth magnets to the flywheel spaced 45° apart. We took care to ensure that their polarities were opposite each other's. As the flywheel spins, the magnets pass in front of the Hall effect sensor, and the sensor output completes a four waveform cycles for every rotation of the flywheel. The myDAQ has the capability to measure the frequency of a PWM input at any digital pin. The frequency of the waveform produced by the Hall effect sensor is the rotation velocity of the motor in revolutions per second (rps) multiplied by four. Unfortunately, including the frequency-measuring operation in the software control loop slows the program. We solved this problem by using a second myDAQ to measure the frequency of the Hall effect sensor signal at its pin D5. The second myDAQ produces an analog voltage at its AO1 that is proportional to the motor speed. The voltage produced by the second myDAQ is read by the main, controlling myDAQ at pin AI1.

The Hall effect sensor is a powered device, and besides its connection to the myDAQ, has two other pins. One pin connects to the myDAQ's 5 V power rail, and the other connects to the circuit's common ground. In our mechanical design, we placed the Hall effect sensor so that it could reliably measure the motor speed.

A picture of the breadboard hardware setup for the DC motor experiment is shown in Fig. 3-9. In this picture, the motor is not placed close to the Hall effect sensor, which is implemented on the breadboard.

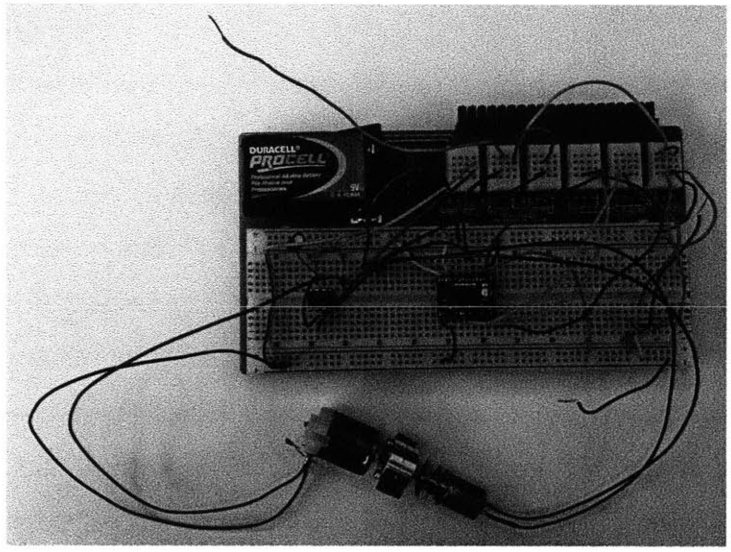


Figure 3-9: A picture of the breadboard hardware setup for the DC motor experiment.

Chapter 4

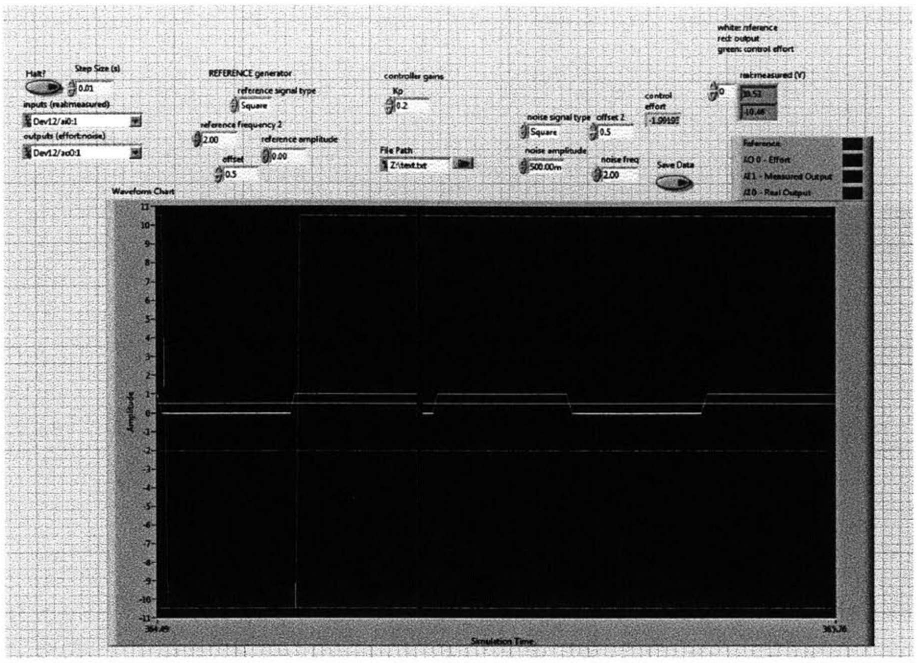
Software Controllers

4.1 Software Introduction

LabView software, used to program the myDAQ, allows for rapid controller development and easy software modification. There are two parts to a LabView control program. First, rather than using textual code to program controllers and interact with DAQ hardware, LabView uses a graphical programming language based on Virtual Instruments, or VI's. VI's are arranged in "block diagrams," with data going from one block to the next. VI's exist for mathematic operations, control algorithms, interacting with DAQ hardware, displaying data, user inputs, and many other functions.

Second, LabView allows users to quickly setup an attractive and useful graphical user interface (GUI). A LabView program's GUI is called the "control panel." The control panel automatically displays blocks used in the block diagram. For example, a block for displaying waveform information automatically generates a graph with plots of inputs in the control panel. Users interact with the control panel while the VI is running, and can use it to specify myDAQ pin-outs, change controller gains, and view plots of measurements.

We used Labview to develop a graphical user interface (GUI) interacting with the myDAQ and the plant for both experiments. The basic parts of the LabView programs are the same for both experiments, and we therefore discuss them together. We



Step Size (s) 0.01
inputs (real:measured) Dev12/ai0:1
outputs (effort:noise) Dev12/ao0:1
REFERENCE generator
reference signal type Square
reference frequency 2 2.00
reference amplitude 0.00
offset 0.5
controller gains
Kp 0.2
File Path Z:\text.txt
noise signal type Square
offset 2 0.5
noise amplitude 500.00m
noise freq 2.00
control effort -1.99195
white: reference
red: output
green: control effort
real:measured (V) 0 10.53 10.46

Figure 4-1: The control panel for the integrator experiment LabView program.

describe unique aspects of the software developed for each experiment as appropriate. We show the control panel for the integrator experiment in Fig. 4-1 and for the motor experiment in Fig. 4-2.

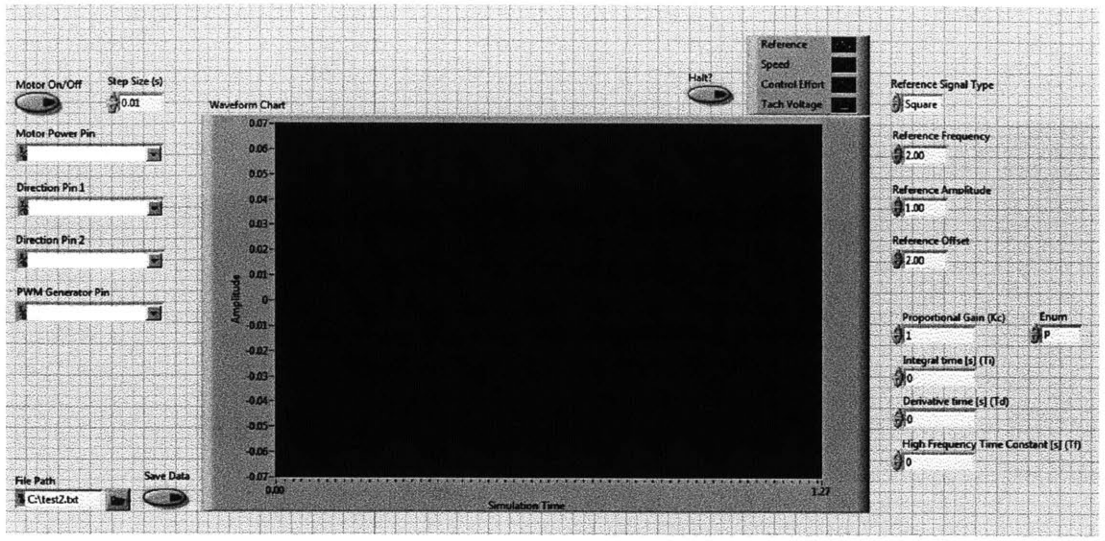
The basic functions shared are as follows:

1. A **function generator** allows the user to choose the reference inputs into the system.

2. The **oscilloscope** allows the user to examine plots of the reference signal, control effort, plant output, and other signals generated and measured in the experiment.

3. **Controllers** use the error between the reference signal and the signal coming into the myDAQ from the plant to output a control effort back to the plant.

Another strength of the LabView software is that it is commonly used in developing control systems, and users can quickly and easily modify VI's to be used with



Step Size (s) 0.01

Motor Power Pin

Direction Pin 1

Direction Pin 2

PWM Generator Pin

File Path C:\test2.txt

Reference Signal Type Square

Reference Frequency 2.00

Reference Amplitude 1.00

Reference Offset 2.00

Proportional Gain (Kc) 1

Enum P

Integral time [s] (Ti) 0

Derivative time [s] (Td) 0

High Frequency Time Constant [s] (Tf) 0

Figure 4-2: The control panel for the motor experiment LabView program.

other NI equipment. Furthermore, if the user is competent in LabView programming, they can modify the software developed in this project. Software flexibility is a powerful tool in using the myDAQ system as an education tool for controls projects.

4.2 LabView Software Explanation

4.2.1 Controller and Control Effort Output

The PID controller is implemented inside of a control and simulation loop, which runs at a sampling frequency of $f_s = 100$ Hz. The controller is a transfer function block (the transfer function block may be continuous or discrete). It takes as inputs the gains specified by the user, as well as the error signal, which is the difference between the reference signal generated by the function generator and the current system value measured by the myDAQ.

If the user desires to use continuous PID control, they may enter calculated gains K, T_i, T_f, and T_d into the control panel. Alternatively, the user may turn off the functionality for entering gains and instead open the block diagram and enter the transfer function manually. The user can only change the transfer function block while the program is not running.

The control effort is output through a pin on the myDAQ device. The integrator experiment uses the control effort signal as a voltage output. The output is limited at ± 10 V, as this is the limit of the analog out pins on the myDAQ.

In the DC motor experiment, a series of blocks transforms the control effort into a PWM duty cycle. The duty cycle is equal to the controller voltage divided by 9 V (since a 9 V battery is used as the motor driver power source). We limit the duty cycle to 100%. Trying to complete many complex measurement operations overloads the computational power of the myDAQ, resulting in significant processing lag. We fixed this by using a second myDAQ to measure the frequency of the signal produced by the Hall effect sensor. The second myDAQ's program measures the signal frequency, multiplies it by a gain, and outputs it as an analog voltage. The controlling myDAQ measures the analog voltage produced by the second myDAQ, multiplies it by a gain, and uses the resulting signal to represent the speed of the motor. Jeff C. Jensen wrote the code enabling the frequency measurement and has made it available on NI's website.

We determined the direction of the motor by checking the sign of the voltage generated by the generator-tachometer. Based on the sign of the control effort, two digital pins alternate between HIGH and LOW to change the motors direction as dictated by the motor driver. The control panel has a button for turning the motor on and off. The motor on/off button toggles the value at the digital pin connected to the STBY pin on the motor driver.

4.2.2 Function Generator for Signal Generation

The function generator is the means whereby the myDAQ acts as a standard bench-top function generator, producing a repeating voltage waveform. On the control panel, the user may use a drop-down menu to choose between sine, square, triangle, and saw-tooth waveforms. The user can also set the amplitude, offset, and frequency of the waveform using the fields. On the backend, we programed the function generator using the signal generator block. It takes as inputs the amplitude, offset, and frequency provided by input boxes on the control panel. The signal generator block's output is used as the reference signal to drive the controller.

In the integrator experiment, we measure the reference signal amplitude in volts, and we measure the frequency in hertz. We also use the function generator to produce noise and disturbance outputs (only one of these is used at a time, as this experiment is a single-input-single-ouput (SISO) system). In order to turn off the noise or disturbance output, the user may simply set the amplitude to zero.

The DC motor experiment uses the function generator to produce a reference input with an amplitude in revolutions per second (rps) and a frequency in hertz. To simulate a constant velocity, a very low frequency may be used. To protect the motor gearbox, it is recommended that the maximum rotation frequency be limited to 3 Hz.

4.2.3 Oscilloscope and Plant Output Sampling

The oscilloscope shows the output of the plant as sampled by the myDAQ. It is possible to display the plant output, the control effort, and the reference signal on the scope, and these signals appear in different colors on the same scale. We programmed the scope using the waveform sampler block. The various measurements from the myDAQ and the function generator run into the waveform sampler block, and the block displays the plots on the control panel.

The integrator experiment uses an analog input pin to measure the voltage output of the summing amplifier. We use the measured signal to generate the error given to the controller, and we display the signal on the oscilloscope. We use another analog input pin to measure the "real" signal coming directly out of the second integrator circuit. The "real" signal appears on the scope for comparison with the signal affected by the addition of noise, which is reflected in the "measured" signal.

In the DC motor experiment, we pass the output of the generator, which shows the direction of the motor, into the myDAQ through an analog input pin. The generator signal is measured to add a sign to the motor speed. That speed is measured in the controlling myDAQ as an analog voltage multiplied by some gain to find the speed of the motor flywheel in rps. We use the speed to find the error, which we in turn use to generate the control effort.

4.2.4 Supporting Software Blocks

Input and output pins must be appropriately initiated and terminated in the block diagram. LabView uses a series of blocks to do this easily, and the software provides menus on the control panel to allow users to select the pins to use for the different inputs and outputs.

The user may save data displayed on the scope. The user must enter the name of the .txt file to which the data will be saved before the program begins. If the file name is not changed between experiments, the program will overwrite old data. The program writes to disk the run-time time stamp and the signal values displayed on the scope. The user may begin saving data by pressing a button on the control panel, and after the program has collected sufficient data, the user may stop saving data by clicking the button again.

4.3 Software Sampling Frequencies

The myDAQ has a relatively slow sampling frequency for real time control, in the range of 100-200 Hz. Furthermore, the software implementation of the controller is naturally a discrete-time controller, and in some cases we use this lab to present principles of continuous-time controls. We were careful to design plant hardware should with frequency response breakpoints well below about 50 Hz.

Chapter 5

Expected System Behaviors

We discuss the expected behavior of the experimental systems. In order to guide experimental testing, we developed models of both systems with representative controllers. Since there are many possible combinations, we do not present all possible controllers. Rather, we chose a few representative systems confirm system behavior and detect unanticipated experimental behaviors. Using the negative feedback control block diagram explained in Chapter 2, we derived the closed-loop transfer functions for each representative control system and plug in appropriate numerical values. We applied representative gain values to the systems, and we used MATLAB to predict the step responses and sinusoidal responses of the systems. We used the predictions to evaluate the experimental system designs.

For all experiments, we operated the MyDAQ system with a sampling period $T = 0.01$ s.

5.1 Integrator Experiment

As representative control experiments, we chose the single integrator plant with: 1) a proportional controller, and 2) a discrete-time proportional-integral controller. We controlled the double integrator plant with a discrete-time proportional-derivative controller. We also studied: 1) single integrator discrete-time proportional controlled noise rejection, and 2) single integrator discrete-time proportional controlled disturbance rejection.

As was derived in Chapter 2, the plant transfer function for a single integrator is

$$\frac{v_{out}(s)}{v_{in}(s)} = G_s(s) = \frac{-1}{RCs}. \tag{5.1}$$

The experimental hardware uses $R = 100$ kΩ and $C = 0.1$ μF. We use these values to calculate the continuous time single integrator plant transfer function

$$G_s(s) = \frac{-100}{s} \tag{5.2}$$

The negative sign is canceled out in the controller software. At the summing junction where the error is calculated, we invert the measured signal from the plant.

Since the MyDAQ system is inherently discrete, we need a discrete representation of the integrator system. We express the transfer function in discrete-time using the zero order hold transformation (ZOH), obtaining

$$G_{sd}(z) = \frac{T}{RC}\frac{1}{z-1}. \tag{5.3}$$

Where T is the sampling period of the controller. By using the values from the experimental hardware, and since $T = RC$, we find

$$G_{sd}(z) = \frac{1}{z-1}. \tag{5.4}$$

The plant transfer function for the double integrator is

$$G_d(s) = \frac{1}{R^2C^2s^2}. \tag{5.5}$$

Using the earlier mentioned experimental hardware values, we calculate

$$G_d(s) = \frac{10^4}{s^2} \tag{5.6}$$

Using the ZOH transform, we find that the double integrator plant transfer function in discrete-time becomes

$$G_{dd}(z) = \frac{T^2}{2R^2C^2}\frac{z+1}{z^2-2z+1} \tag{5.7}$$

And this, with the experimental hardware values where $T = RC$, becomes

$$G_{dd}(z) = \frac{0.5z+0.5}{z^2-2z+1} \tag{5.8}$$

5.1.1 Single Integrator with Continuous Time Proportional Control

Closed Loop Transfer Function

The plant transfer function given in Eq. 5.1 is controlled using a proportional controller of form

$$G_c(s) = K_p, \tag{5.9}$$

which yields the closed loop transfer function

$$G_{CL}(s) = \frac{V_o(s)}{V_{ref}(s)} = \frac{K_p}{RCs + K_p}. \tag{5.10}$$

For $RC = 0.01$ s, we choose the gain $K_p = 0.1$. Using the experimental hardware values and the representative gain, we find

$$G_{CL}(s) = \frac{10}{s+10}. \tag{5.11}$$

We can use the closed loop transfer function to determine the output of the system based on reference inputs.

Step Response

We find the step response by taking the inverse Laplace transform of the closed loop transfer function $G_{CL}(s)$ with a step input V_{ref} of magnitude v_r, which in Laplace

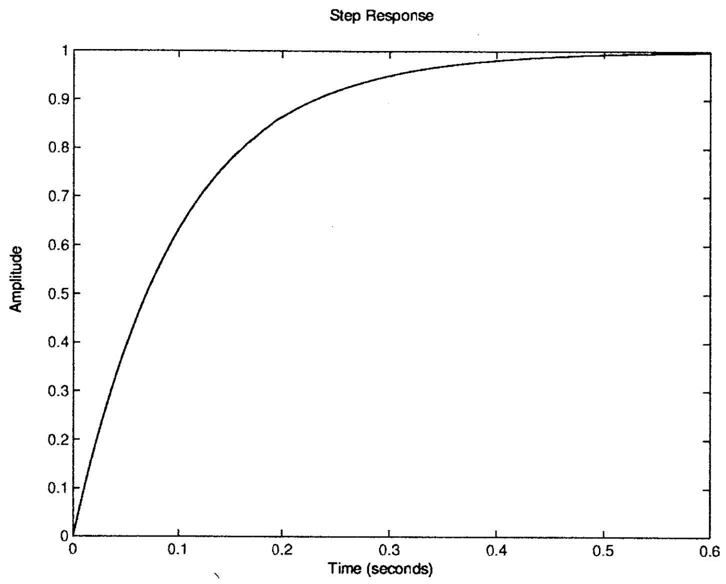


Figure 5-1: The predicted step response of the single integrator continuous time proportional control system with $K_p = 0.1$.

space is

$$V_{ref}(s) = \frac{v_r}{s}. \tag{5.12}$$

The step response of the single integrator continuous time proportional control system is

$$v_o(t) = v_r\left(1 - e^{-\frac{Kt}{RC}}\right). \tag{5.13}$$

Using the experimental hardware values as previously explained, we calculate that the step response with input magnitude 1.0 V becomes

$$v_o(t) = 1 - e^{-10t}. \tag{5.14}$$

The step response is shown graphically in Fig. 5-1.

We can use the step response to evaluate the system's behavior at the selected gain.

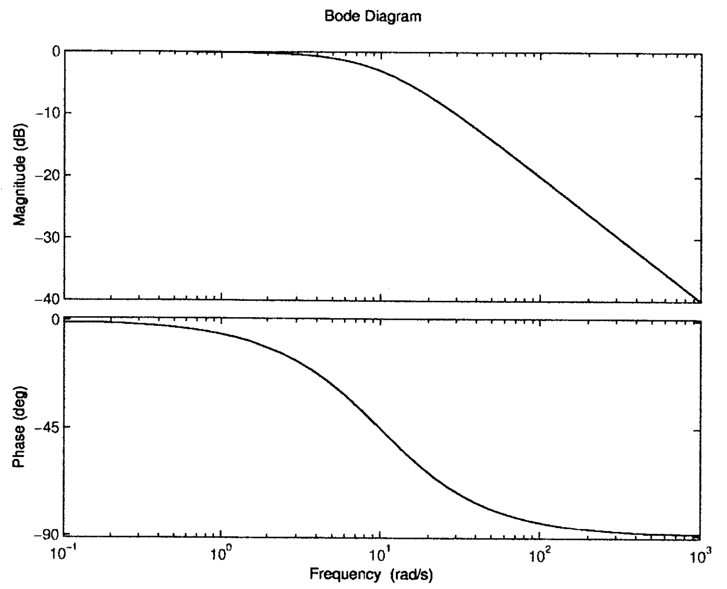


Figure 5-2: The predicted sinusoidal magnitude and phase response of the single integrator continuous time proportional control system with $K_p = 0.1$.

Sinusoidal Response

We found the sinusoidal response of the single integrator continuous proportional control system using Bode analysis. The Bode magnitude and phase plots are shown in Fig 5-2.

We can use the sinusoidal response to evaluate the system's behavior at the selected gain. For example, the magnitude of the sinusoidal response is reduced by $\sqrt{2}/2$ from the reference amplitude at $f = 1.6$ Hz.

5.1.2 Single Integrator Proportional-Integral Discrete Control

Closed Loop Transfer Function

We begin with a proportional-integral controller of the form

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s}\right), \tag{5.15}$$

which we transform to z-space using the forward Euler transform ($s = z - 1$), yielding

$$G_{cd}(z) = K_p\left(\frac{T_i(z-1)+T}{T_i(z-1)}\right) \tag{5.16}$$

Using $K_p = 0.1$ and $T_i = 0.1$, we calculate

$$G_{cd}(z) = \frac{0.1z - 0.09}{z-1} \tag{5.17}$$

We control the plant transfer function given in Eq. 5.3 using the controller here developed, and we find the closed loop transfer function

$$G_{CL}(z) = \frac{V_o(z)}{V_{ref}(z)} = \frac{K_pT_iTz + K_pT(T-T_i)}{RCT_iz^2 + T_i(K_pT - 2RC)z + K_pT(T-T_i) + RCT_i}. \tag{5.18}$$

Using the selected gain values along with the previously stated experimental hardware values, we find

$$G_{CL}(z) = \frac{0.1z - 0.09}{z^2 - 1.9z + 0.91}. \tag{5.19}$$

We can use this transfer function to determine the output of the system based on reference inputs.

Step Response

We could find the step response of this system by taking the inverse z-transform of the closed loop transfer function $G_{CL}(z)$ with a step input V_{ref}. However, the analytical and numerical solutions of this action are computationally intensive and

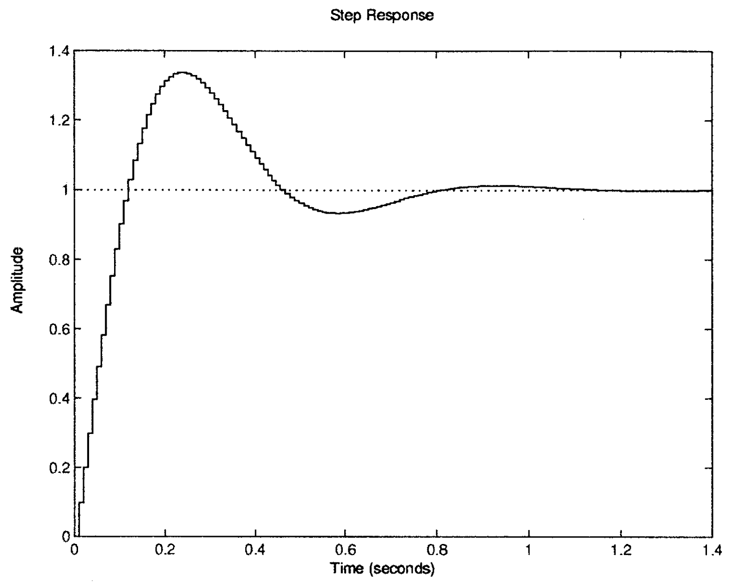


Figure 5-3: The predicted step response of the single integrator discrete-time proportional-integral control system with $K_p = 0.1$ and $T_i = 0.1$.

offer little additional insight into the system. Rather, we computed the step response numerically as in Fig. 5-3. The peak is at approximately 1.33 V, and the peak time is 0.24 s. These data will be useful in evaluating the experimental hardware.

Sinusoidal Response

We calculated the sinusoidal response of the single integrator discrete proportional-integral control system using Bode analysis. The Bode magnitude and phase plots are shown in Fig 5-4.

We can use the sinusoidal response to evaluate the system's behavior at the selected gain. The magnitude of the response is 1.5 times the reference amplitude at a frequency of $f = 1.6$ Hz.

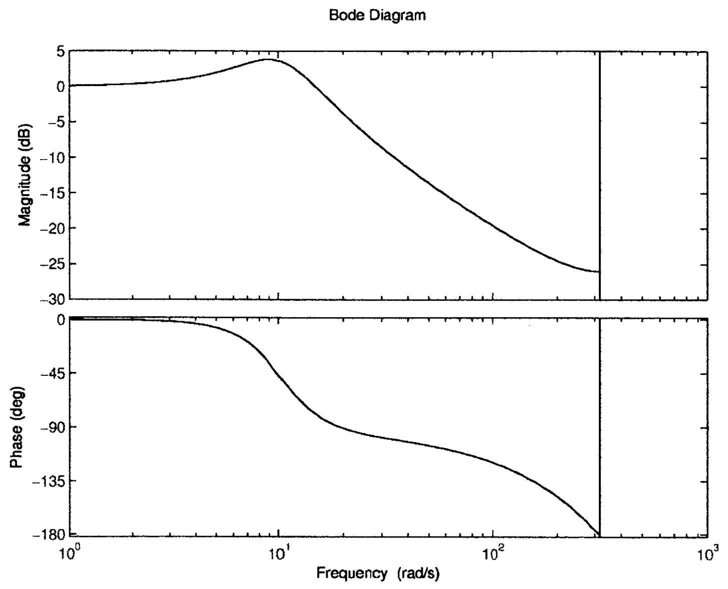


Figure 5-4: The predicted sinusoidal magnitude and phase response of the single integrator discrete-time proportional-integral control system with $K_p = 0.1$ and $T_i = 0.1$.

5.1.3 Double Integrator Proportional-Derivative Discrete Control

Closed Loop Transfer Function

We begin with a proportional-derivative controller of the form

$$G_c(s) = K_p\left(1 + \frac{T_d s}{0.1T_d s + 1}\right), \tag{5.20}$$

which is also known as a lead controller, and we transform it to z-space using the forward Euler transform to find

$$G_{cd}(z) = K_p\left(\frac{1.1T_d\,(z-1) + T}{0.1T_d\,(z-1) + T}\right) \tag{5.21}$$

Using $K_p = 0.1$ and $T_d = 0.07$, we find

$$G_{cd}(z) = \frac{1.1z - 1.024}{z - 0.2397} \tag{5.22}$$

The plant transfer function given in Eq. 5.3, controlled using the controller we developed, yields the closed loop transfer function

$$G_{CL}(z) = \frac{V_o(z)}{V_{ref}(z)} = \frac{K_p T^2}{2R^2C^2} \frac{(1.1T_d(z-1)+T)(z+1)}{(0.1T_d(z-1)+T)(z^2-2z+1)+(1.1T_d(z-1)+T)(z+1)}. \tag{5.23}$$

Using the representative gains and the experimental hardware values, we find

$$G_{CL}(z) = \frac{0.55z^2 + 0.038z - 0.512}{z^3 - 1.69z^2 + 1.517z - 0.7516}. \tag{5.24}$$

This transfer function may be used to determine the output of the system based on reference inputs.

Step Response

We can find the step response of this system by taking the inverse z-transform of the closed loop transfer function G_{CL} with a step input V_{ref}. However, the analytical and numerical solutions of this action are computationally intensive and offer little additional insight into the system. Rather, we compute the step response numerically as in Fig. 5-5. The peak is at approximately 1.81 V, and the peak time is 0.03 s. We can use these data in evaluating the experimental hardware.

Sinusoidal Response

We found the sinusoidal response of the double integrator discrete proportional-derivative control system using Bode analysis. The Bode magnitude and phase plots are shown in Fig. 5-6.

The sinusoidal response is usually used to evaluate the system's behavior at the selected gain. With the gains and experimental hardware values previously mentioned,

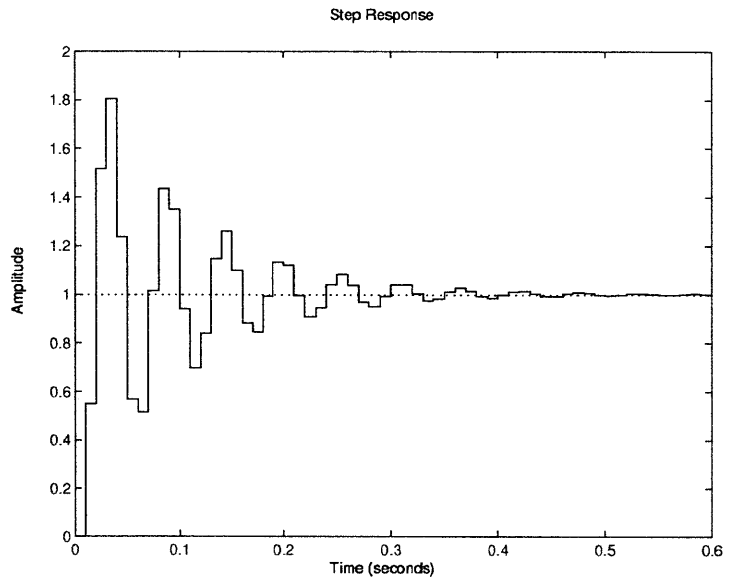


Figure 5-5: The predicted step response of the double integrator discrete-time proportional-derivative control system with $K_p = 0.1$ and $T_d = 0.07$.

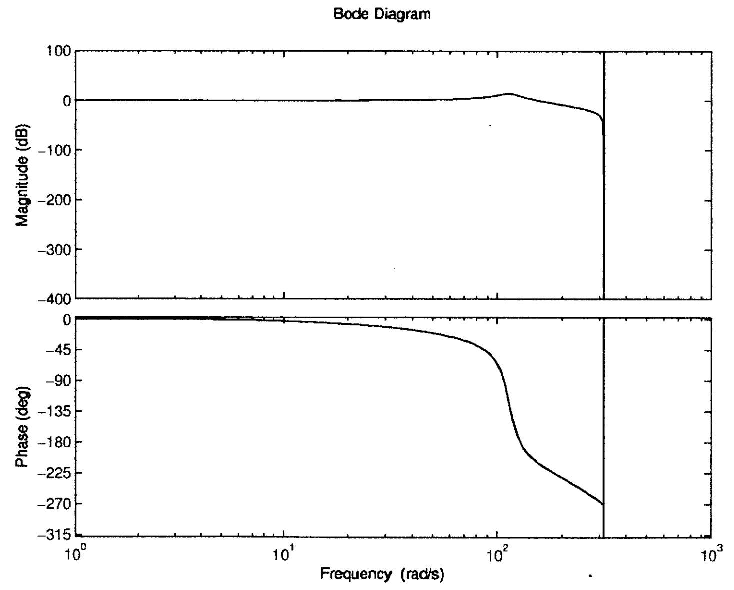


Figure 5-6: The predicted sinusoidal magnitude and phase response of the double integrator discrete-time proportional-derivative control system with $K_p = 0.1$ and $T_i = 0.07$.

the magnitude of the response is 5.4 times the reference amplitude at a frequency of $f = 18$ Hz.

5.1.4 Single Integrator Proportional Noise Rejection

Closed Loop Transfer Function

We derived the closed loop transfer function responding to a noise input using the feedback block diagram for noise shown in Chapter 2. We use the plant transfer function given in Eq. 5.3, controlled using a proportional controller

$$G_c(z) = K_p \tag{5.25}$$

with a noise input and no reference input to find the closed loop transfer function

$$G_{CL}(z) = \frac{-K_p T}{RC\,(z-1) + K_p T}. \tag{5.26}$$

With the exception of the negative sign, the closed loop transfer function is the same form as a closed loop transfer function with a regular reference input. Measured noise is difficult to filter because reducing the effects of noise also reduces the system response to the reference input. We chose a representative gain value $K_p = 0.1$. We use the representative gain, along with the experimental hardware values, to calculate

$$G_{CL}(z) = \frac{-0.1}{z - 0.9}. \tag{5.27}$$

We can use the closed loop transfer function to determine the output of the system based on measured noise inputs.

Step Response

We can calculate the step response of this system by taking the inverse z-transform of the closed loop transfer function G_{CL} with a step input V_{ref}. However, the analytical and numerical solutions of this action are computationally intensive and offer little

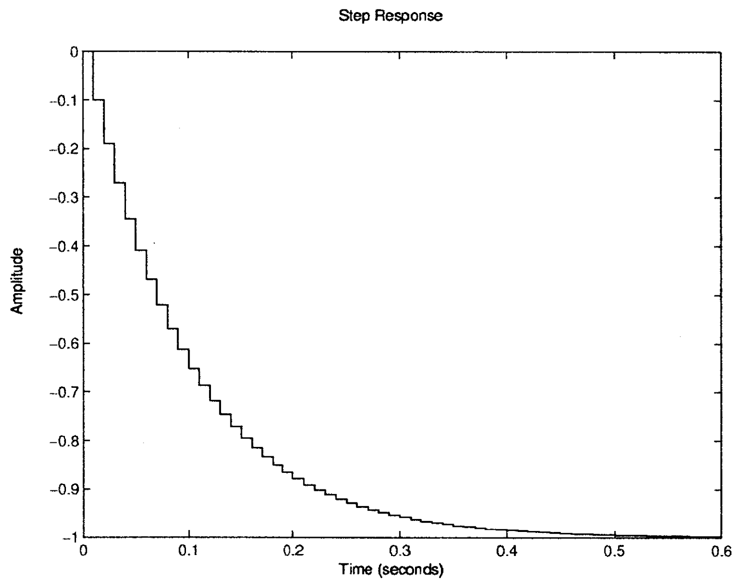


Figure 5-7: The predicted step response of the single integrator discrete-time proportional control noise rejection system with $K_p = 0.1$.

additional insight into the system. In any case, it is clear that it should be similar to the step response shown earlier in Section 5.1.1. The time constant is $\tau = 0.1$ s. The step response is shown graphically in Fig. 5-7.

Sinusoidal Response

We calculated the sinusoidal response of the single integrator discrete proportional-integral control system using Bode analysis. The Bode magnitude and phase plots are shown in Fig 5-8. The phase begins at 180° because of the negative sign in the closed loop noise transfer function.

We can use the sinusoidal response to evaluate the system's behavior at the selected gain. As with the system using a reference input, the magnitude of the response is reduced by $\sqrt{2}/2$ from the noise input amplitude at a frequency of $f = 1.6$ Hz.

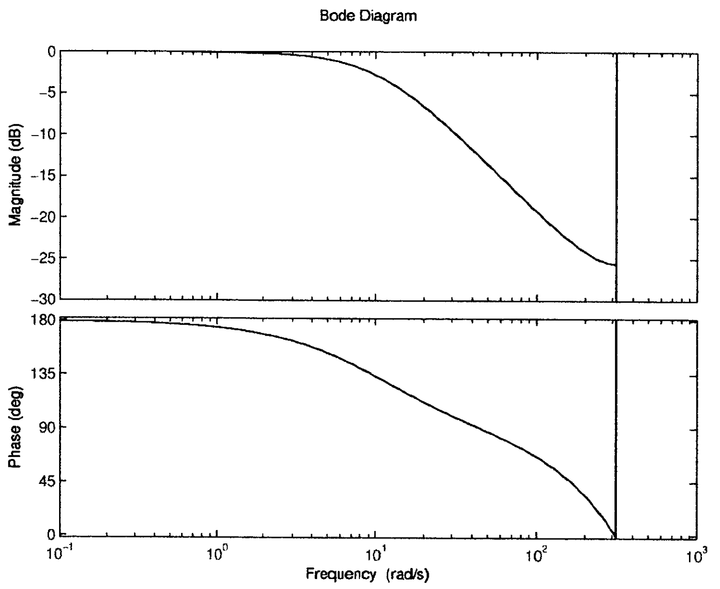


Figure 5-8: The predicted sinusoidal magnitude and phase response of the single integrator discrete-time proportional control noise rejection system with $K_p = 0.1$.

5.1.5 Single Integrator Proportional Disturbance Rejection

Closed Loop Transfer Function

We can derive the closed loop transfer function responding to a disturbance input using the feedback block diagram for disturbance shown in Chapter 2. We use the plant transfer function given in Eq. 5.3, controlled using a proportional controller

$$G_c(z) = K_p \tag{5.28}$$

with a disturbance input and no reference input, to find the closed loop transfer function

$$G_{CL}(z) = \frac{T}{RC\,(z-1) + K_pT}. \tag{5.29}$$

The controller gain K_p is only in the denominator of the transfer function. As gain increases, the magnitude of the disturbance response will decrease. We chose a representative gain $K_p = 0.2$. Using this gain along with the previously stated experimental

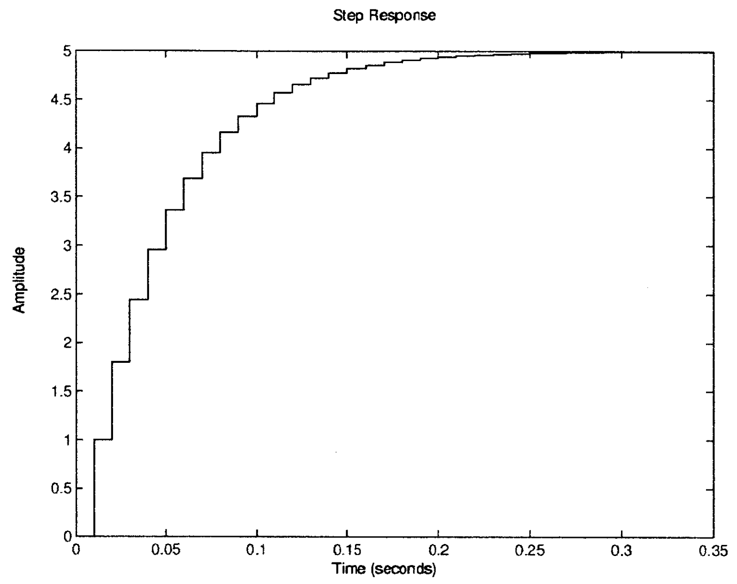


Figure 5-9: The predicted step response of the single integrator discrete-time proportional control disturbance rejection system with $K_p = 0.2$.

hardware values, we calculate the the closed loop transfer function

$$G_{CL}(z) = \frac{1}{z - 0.8}. \tag{5.30}$$

We can use the closed loop transfer function to determine the output of the system based on reference inputs.

Step Response

We can find the step response of this system by taking the inverse z-transform of the closed loop transfer function G_{CL} with a step input V_{ref}. However, the analytical and numerical solutions of this action are computationally intensive and offer little additional insight into the system. Using MATLAB, we computed the numerical step response as in Fig. 5-9. We calculated the time constant $\tau = 0.05$ s.

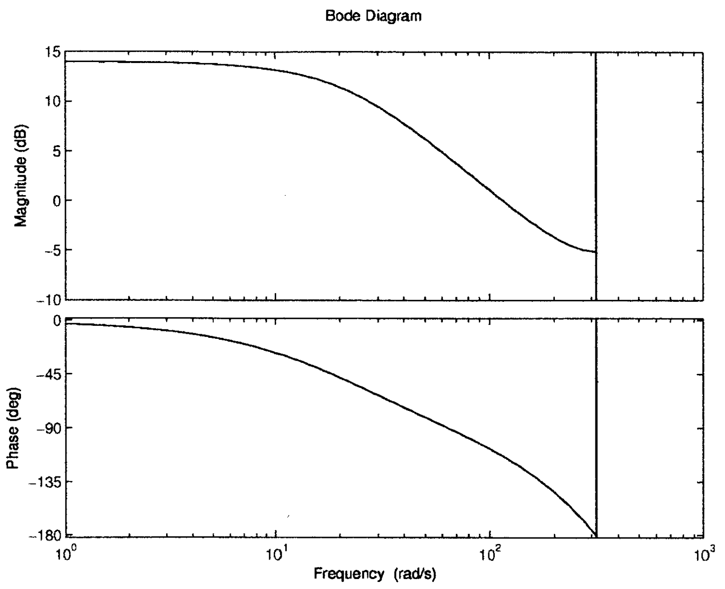


Figure 5-10: The predicted sinusoidal magnitude and phase response of the single integrator discrete-time proportional control disturbance rejection system with $K_p = 0.2$.

Sinusoidal Response

We computed the sinusoidal response of the single integrator discrete proportional control disturbance rejection system using Bode analysis. The Bode magnitude and phase plots are shown in Fig 5-10.

We can use the sinusoidal response to evaluate the system's behavior at the selected gain. The magnitude of the response is magnified by 1.5 from the disturbance input amplitude at a frequency of $f = 11.6$ Hz.

5.2 DC Motor Experiment

We derived the plant transfer function for a DC motor with rotational inertia J, rotational damping B, armature resistance R, armature constant k_m, and a gear ratio N in Chapter 2, and we repeat it here:

$$G_p(s) = \frac{Nk_m}{JRs + BR + N^2k_m^2}. \tag{5.31}$$

The numerical values for N, k_m, J, R, and B are available in Table 1 in Chapter 2, and we find these values in the experiments in Chapter 6. Using the experimental hardware values, we calculate

$$G_p(s) = \frac{0.018}{7.4 \times 10^{-5}s + 0.0034} \tag{5.32}$$

We transform the continuous time transfer function to a discrete-time transfer function using the ZOH transform, and we find

$$G_{pd}(z) = \frac{Nk_mT}{JRz + BRT + N^2k_m^2T - JR}. \tag{5.33}$$

As before, $T = 0.01$. Using these values, the numerical transfer function is

$$G_{pd}(z) = \frac{1.93}{z - 0.632} \tag{5.34}$$

5.2.1 Proportional Continuous Control

Closed Loop Transfer Function

We use the plant transfer function for the DC motor as given in Eq. 2.20, and we control the plant using a proportional controller of form

$$G_c(s) = K_p. \tag{5.35}$$

We find the closed loop transfer function

$$G_{CL}(s) = \frac{\Omega_o(s)}{\Omega_{ref}(s)} = \frac{NKk_m}{JRs + BR + N^2k_m^2 + NKk_m}. \tag{5.36}$$

We selected a representative gain $K_p = 0.01$. Using the experimental hardware values found in Chapter 6 and the representative gain, we find

$$G_{CL}(s) = \frac{1.79 \times 10^{-4}}{7.41 \times 10^{-5} s + 0.0036}. \tag{5.37}$$

We can use the closed loop transfer function to determine the output of the system based on reference inputs.

Step Response

The step response of this system may be found by taking the inverse Laplace transform of the closed loop transfer function G_{CL} with a step input. The step response of the dc motor continuous time proportional control system is

$$\omega_o(t) = \omega_r \frac{NKk_m}{BR + N^2 k_m^2 + NKk_m} \left(1 - e^{-\frac{BR+N^2k_m^2+NKk_m}{JR}t}\right) \tag{5.38}$$

Using the experimental hardware values as previously explained, the step response with input ω_r becomes

$$\omega_o(t) = \omega_r 21 \left(1 - e^{-48.3t}\right). \tag{5.39}$$

This step response is shown graphically in Fig. 5-11.

We can use the step response to evaluate the system's behavior at the selected representative gain. Using an input of $\omega_r = 400$ rps, the steady state velocity will be $\omega_{ss} = 20$ rps, and the time constant will be $\tau = 0.021$ s.

5.2.2 Proportional-Integral Discrete Control

Closed Loop Transfer Function

We controlled the DC motor system using a discrete-time proportional-integral controller of form outlined in Section 5.1.2, which is

$$G_{cd}(z) = K_p \left(\frac{T_i(z-1) + T}{T_i(z-1)}\right). \tag{5.40}$$

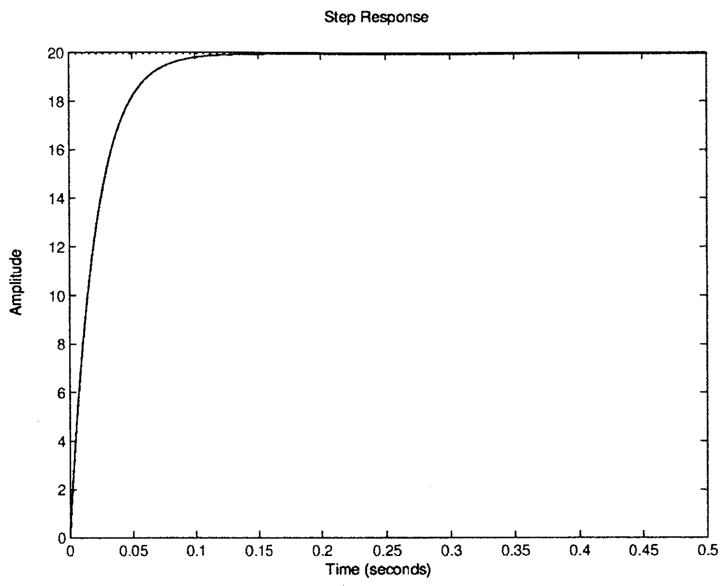


Figure 5-11: The predicted step response of the DC motor proportional control system with $K_p = 0.01$ and $\omega_r = 400$ rev/s.

Using the representative gains of $K_p = 0.1$ and $T_i = 0.025$, we find

$$G_{cd}(z) = \frac{0.1z - 0.06}{z - 1} \tag{5.41}$$

Combining the discrete controller transfer function with the DC motor discrete plant transfer function, we calculate

$$G_{CL}(z) = \frac{\Omega_o(z)}{\Omega_{ref}(z)} = \frac{K_p N k_m T \left(T_i\left(z-1\right)+T\right)}{\left(JRz + BRT + N^2 k_m^2 T - JR\right)\left(T_i\left(z-1\right)\right) + K_p N k_m T \left(T_i\left(z-1\right)+T\right)}. \tag{5.42}$$

Using the experimental hardware values found in Chapter 6 and the representative gains chosen, we find the closed loop transfer function

$$G_{CL}(z) = \frac{0.1932z - 0.1159}{z^2 - 1.439z + 0.5161}. \tag{5.43}$$

We can use the closed loop transfer function to determine the output of the system based on reference inputs.

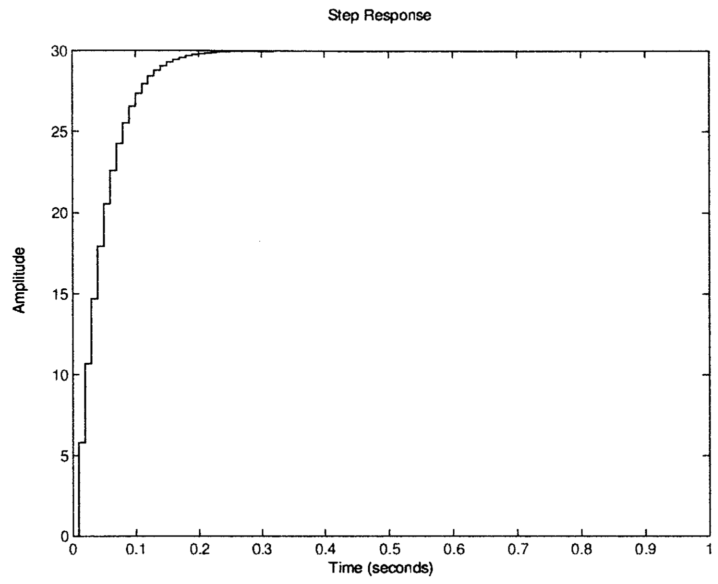


Figure 5-12: The predicted step response of the DC motor discrete-time proportional-integral control system with $K_p = 0.1$, $T_i = 0.025$ and $\omega_r = 30$ rps.

Step Response

We can find the step response of this system by taking the inverse z-transform of the closed loop transfer function G_{CL} with a step input ω_r. However, the analytical and numerical solutions of this action are computationally intensive and offer little additional insight into the system. Rather, we numerically computed the step response to $\omega_r = 30$ rev/s as in Fig. 5-12. The error in the system is eliminated using integral control.

We can use step response to evaluate the system's behavior at the selected gain. With an input of $\omega_r = 30$ rps, the steady state velocity will be $\omega_{ss} = 30$ rps.

5.2.3 Proportional-Integral-Derivative Continuous Control

Closed Loop Transfer Function

We begin with the plant transfer function for the DC motor as given in Eq. 2.20, and we control the plant using a proportional-integral-derivative controller of the form

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + \frac{T_d s}{0.1 T_d s + 1}\right). \tag{5.44}$$

We chose representative gains $K_p = 0.05$, $T_i = 0.05$, and $T_d = 0.1$, and we computed

$$G_c(s) = \frac{0.55s^2 + 6s + 100}{s^2 + 100s}. \tag{5.45}$$

The symbolic form of the closed loop transfer function is exceptionally unwieldy, we do not derive it here. Using the experimental hardware values found in Chapter 6 and the representative gains, we calculated the close loop transfer function

$$G_{CL}(s) = \frac{0.009818s^2 + 0.1071s + 1.785}{7.41 \times 10^{-5}s^3 + 0.02063s^2 + 0.4471s + 1.785} \tag{5.46}$$

We can use the closed loop transfer function to determine the output of the system based on reference inputs.

Step Response

We can compute the step response of this system by taking the inverse Laplace transform of the closed loop transfer function G_{CL} with a step input. The symbolic representation of the step response of the dc motor continuous time proportional control system is out of the scope of this paper. Instead, we computed step response numerically as in Fig. 5-13.

We can use the step response to evaluate the system's behavior at the selected gain. Using an input of $\omega_r = 40$ rps, the steady state velocity will be $\omega_{ss} = 40$ rps.

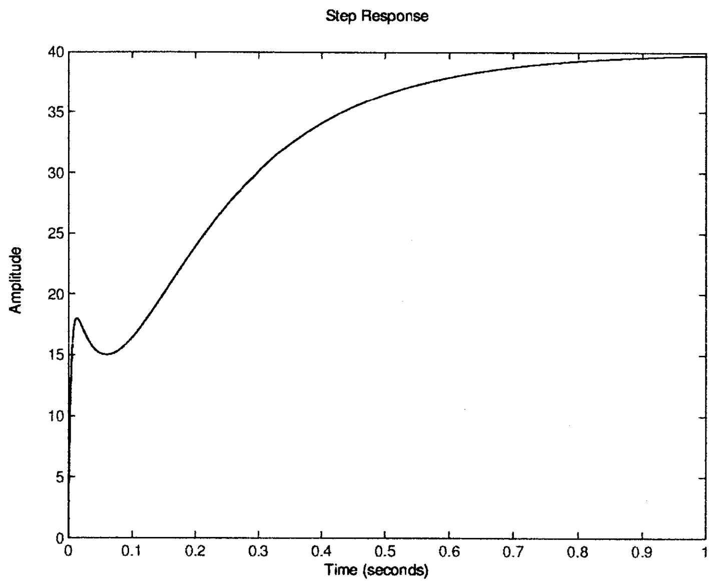


Figure 5-13: The predicted step response of the DC motor proportional-integral-derivative control system with $K_p = 0.05$, $T_i = 0.05$, $T_d = 0.1$ and $\omega_r = 40$ rev/s.

Chapter 6

Laboratory Tests and Implementation

After we designed and built the experimental setups, and once we had completed the design of the software, we drafted write-ups to accompany the experiments. We evaluated the write-ups and the accompanying hardware and software together to ensure proper functionality for use in the classroom, as well as by self-learners. In addition, we tested the hardware and software to find odd behavior and better understand the limits of the equipment and controllers.

6.1 Laboratory Write-ups

We produced write-ups to guide the use of the hardware and software developed in this project. Our write-ups can be used by instructors of controls courses to build laboratory programs which their students may follow. Much of the material is written such that it may be used directly by students in the lab. Also, our write-ups stand alone. An interested person with the necessary skills can use the lab and write-up for self-instruction. These reports were an important part of using the myDAQ system to make controls experimentation more accessible and available to broader audiences.

6.1.1 Laboratory Introduction

In the introduction, the write-up introduces the student and the instructor to the hardware and software developed, as well as the LabView and NI myDAQ platform. We describe the philosophy behind the labs. We also explain the intended use of the systems, and we give ideas for how the labs and write-ups may be used and modified to suit the needs of students and instructors.

6.1.2 Laboratory Background

We included the background information written earlier in this paper, including information about basic control theory, in the write-ups. We intend the background information as a refresher for students, helping them bridge what they have learned in the classroom with the knowledge they will be applying in the laboratory experience. Instructors can remove some of this information depending on what types of modeling and other pre-lab problems they will assign to their students.

6.1.3 Laboratory Hardware and Software Descriptions

We wrote descriptions of the hardware and software, and we provide instructors and students with the knowledge necessary to modify the hardware and software to suit their needs. While the experiments were designed to be used as is, they can be used as a platform for further development by other people. For example, we will share the board layouts and all LabView software on the NI website for use by interested parties. People may order PCB's according to the design provided, which is built for use with the software as-is, or they can modify the layouts in Eagle and change the LabView block diagram to include new functionality or remove unnecessary hardware.

6.1.4 Laboratory User's Guide

We assume that LabView software will be unfamiliar to the majority of users, and naturally every hardware setup is unique. To help users, we wrote a user's guide to help students and instructors use the hardware and software for the experiments. We give step-by-step instructions for using the software, including what buttons to push and common pitfalls in changing controllers, saving data, and making simple modifications to the software block diagrams. We describe the different switches and connections used in the hardware, and we explain the settings used in different parts of the laboratory experiments.

6.1.5 Laboratory Exercises and Problems

We wrote a series of pre-lab and lab problems based on the experiments. Pre-lab problems we wrote included modeling the systems, creating controllers based on root locus and bode analysis, and predicting the behavior of the experimental systems. We hope that these pre-lab problems will prepare students for the experimental part of the lab, and will help them to contrast their predicted behaviors with how actual hardware behaves.

We designed the experimental problems to help students gain an intuition for the systems and controllers. Our problems encourage tinkering with the system and exploring different behaviors. The new portable format of these labs gives users the opportunity to spend more time experimenting and understanding controls. Problems include entering controller gains and then observing how system behaviors change as the gains are changed. We instruct students to note how noise and disturbances affect the system, and also examine how discrete and continuous controllers change behavior.

We included problems to help the instructor design labs for their students. Also, since our lab may be used by self-learners to gain controller design experience, these problems will guide them to learn more about the system and controls. Our problems may be modified to suit the needs of the user. Also, the problem sets we wrote are quite complete, providing learning opportunities to beginners, as well as to more experienced and knowledgeable students. We wrote both continuous and discrete controls problems, and instructors may assign only the subset of problems relevant to their class.

To accompany the problem sets, we wrote sample solution sets to the problems, with solutions to the pre-lab problems, as well as expected behavior for the experimental problems. While we hope that these solutions will be useful to instructors and help self-learners, they are not intended to be complete, since much of the laboratory experience is unpredictable.

6.2 Testing

After we built the hardware and software, and after we had drafted the write-ups, we tested the hardware and software to ensure they would behave as expected. We used the expected behaviors outlined in Chapter 5 as the basis for testing. Our experiments here are based on those previous derivations.

6.2.1 Integrator Experiment

Single Integrator Continuous Time Proportional Control System

Step Response The experimental step response for this system is shown in Fig. 6-1, along with the expected behavior. The expected behavior does not closely match the experimental data. The experimental system is much faster than predicted. Rather than a time constant of 0.1s, the experimental system exhibits a time constant of 0.045 s. However, using the on-screen data (as opposed to data saved to disk and analyzed in MATLAB), the time constant was found to be closer to the calculated value of 0.1 s. We observed that the system sampling slows down noticeably when the program is saving data, and this could be the root of the disparity (we explore this problem further in subsequent sections).

Sinusoidal Response The experimental sinusoidal response for this system is shown in Fig. 6-2. The magnitude, as expected, is reduced by approximately 0.77 at 1.6 Hz, but this is not obvious from the graph we present. Rather, we observed the predicted behavior on the plots on the control panel when the program is not saving data. We again noticed that the sampling slows when the program is saving data.

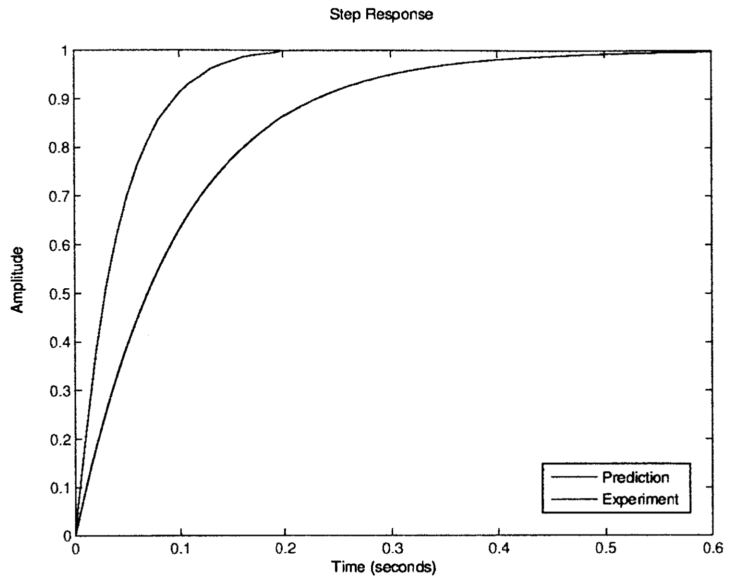


Figure 6-1: Experimental step response of the single integrator continuous time proportional control system compared to the predicted step response.

The response of the system was different while saving, and the slower sampling rate could be the reason why data gathered do not match the calculated values.

Single Integrator Discrete-time Proportional-Integral Control System

Step Response We show the expected and experimental step responses for this system in Fig. 6-3. The experimental system matches closely to the predicted behavior, though it is faster than the prediction. The peak time is approximately 0.15 s, and the peak is at 1.25 V. These values are similar to those predicted in chapter 5, but not the same. Again, this discrepancy could be due to the slowing of the system while saving data.

Sinusoidal Response We show the experimental sinusoidal response for this system in Fig. 6-4. The magnitude was magnified by approximately 1.3 at 1.6 Hz. We did not expect this magnitude response, but rather magnification by 1.5. However, as we have previously mentioned, during experimentation we noticed that when saving

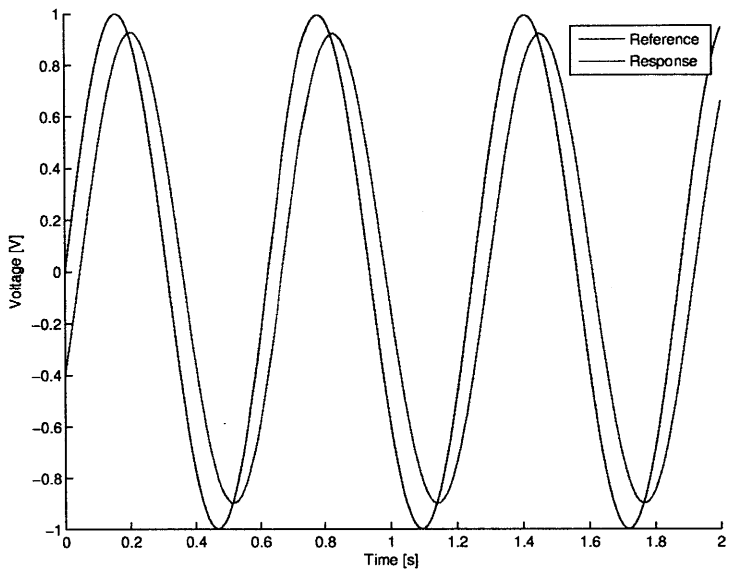


Figure 6-2: Experimental sinusoidal response of the single integrator continuous time proportional control system compared to the reference input.

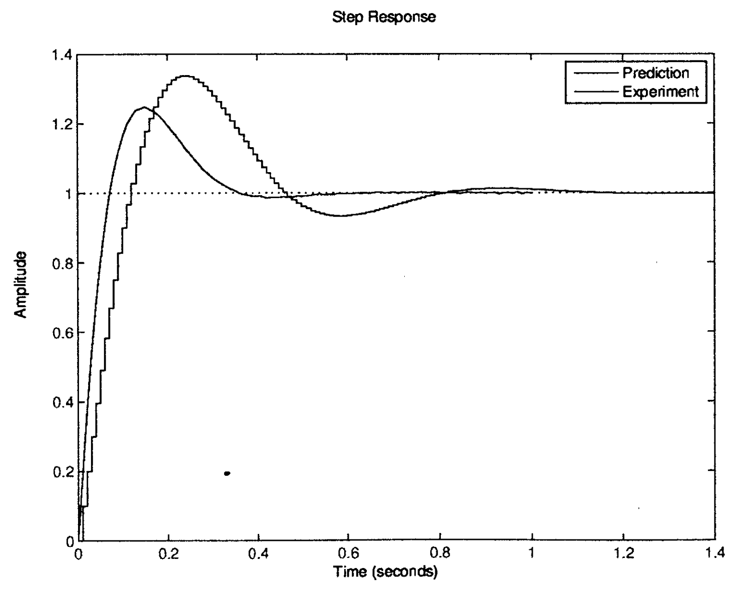


Figure 6-3: Experimental step response of the single integrator discrete-time proportional-integral control system compared to the predicted step response.

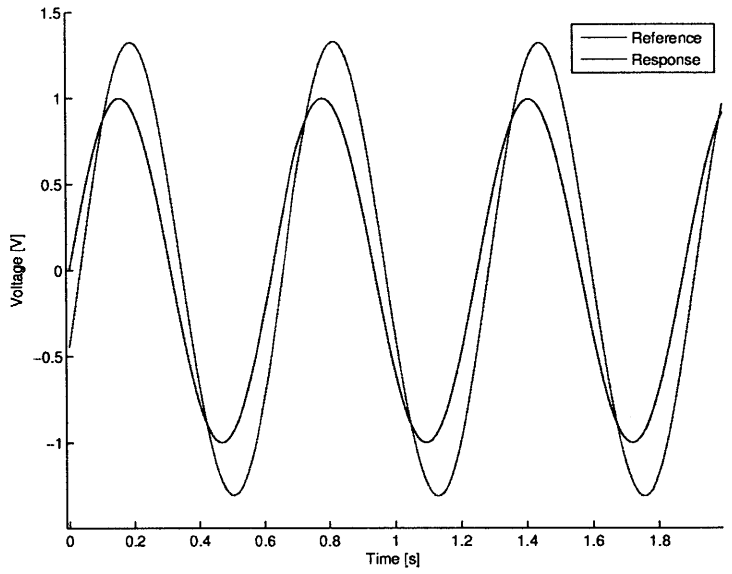


Figure 6-4: Experimental sinusoidal response of the single integrator discrete-time proportional-integral control system compared to the predicted step response.

data, the magnitude was reduced, but when not saving data, the response magnitude was magnified as expected by 1.5. We show unaffected behavior in the screen shot in Fig. 6-5.

Double Integrator Discrete-time Proportional-Derivative Control System

Step Response We show the experimental step response for this system in Fig. 6-6. The gains used in Chapter 5 caused the system to respond as predicted. However, when trying to save data, the system became unstable. In order to achieve stable behavior when trying to save data, another proportional gain $K_{p2} = 0.15$ was added before the discrete controller. The response gathered under these conditions is shown in the plot, along with the expected response with the second proportional gain.

Sinusoidal Response The sinusoidal response suffered from the same problems as the step response. That is, the behavior was as we expected until the data saving feature of the software was used. At that point, the response became unstable. Using

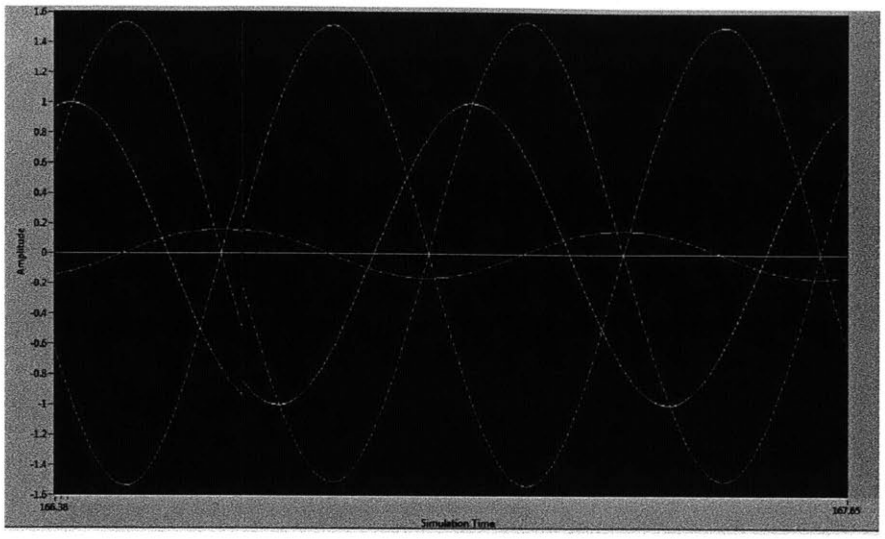


Figure 6-5: Experimental sinusoidal response of the single integrator discrete-time proportional-integral control system compared to the predicted sinusoidal response, as shown on-screen. White is reference, red is control effort, green is real output, and blue is measured output (it has an extra inversion from the extra amplifier circuit).

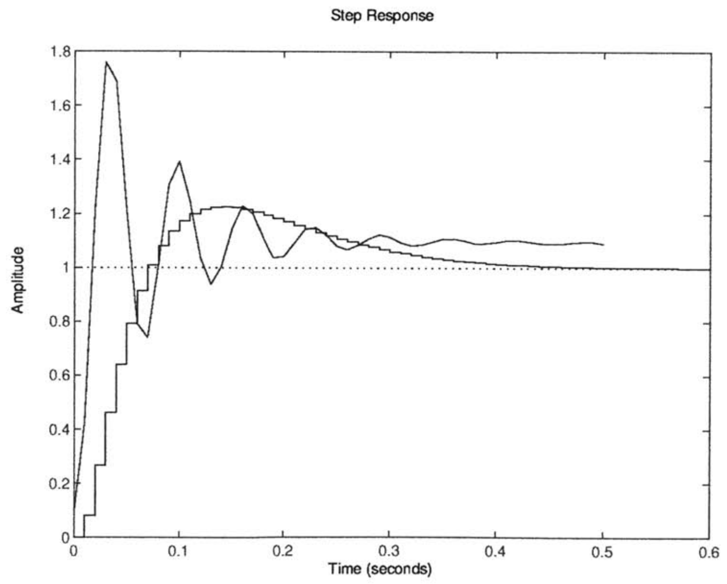


Figure 6-6: Experimental step response of the double integrator discrete-time proportional-derivative control system compared to the predicted step response (both including second proportional gain).

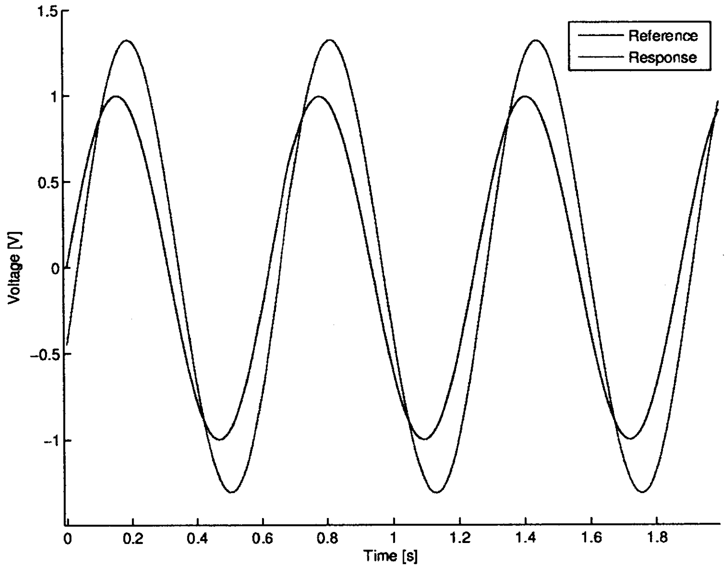


Figure 6-7: Experimental sinusoidal response of the single integrator discrete-time proportional-integral control system compared to the reference input.

the second proportional gain $K_{p2} = 0.15$, we obtained the sinusoidal response shown in Fig. 6-7. The response matches relatively closely the previously expected response of magnifying the reference amplitude by approximately 5.4 at $f = 18$ Hz.

Single Integrator Discrete-time Proportional Noise Rejection System

Step Response We show the experimental step response for this system in Fig. 6-8, along with the expected behavior. As we have come to expect, and similar to the single integrator continuous time proportional controller, the system was faster than expected, with a time constant of 0.045 s for the recorded response.

Sinusoidal Response Our experimental sinusoidal response for this system is shown in Fig. 6-9. As with the reference input system, the magnitude is not reduced by 0.77 at 1.6 Hz when taking data. However, when not taking data, the system matches the predicted behavior of the noise reduction system.

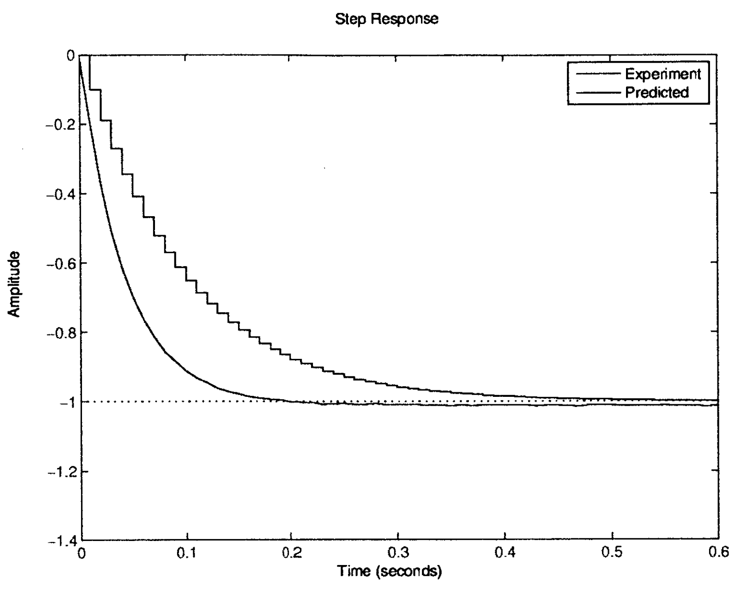


Figure 6-8: Experimental step response of the single integrator discrete-time proportional noise rejection system compared to the predicted step response.

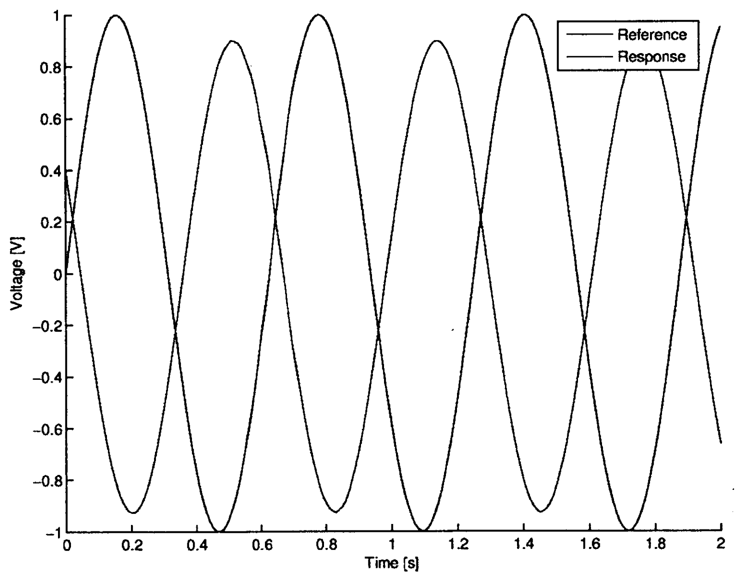


Figure 6-9: Experimental sinusoidal response of the single integrator discrete-time proportional noise rejection system compared to the reference input.

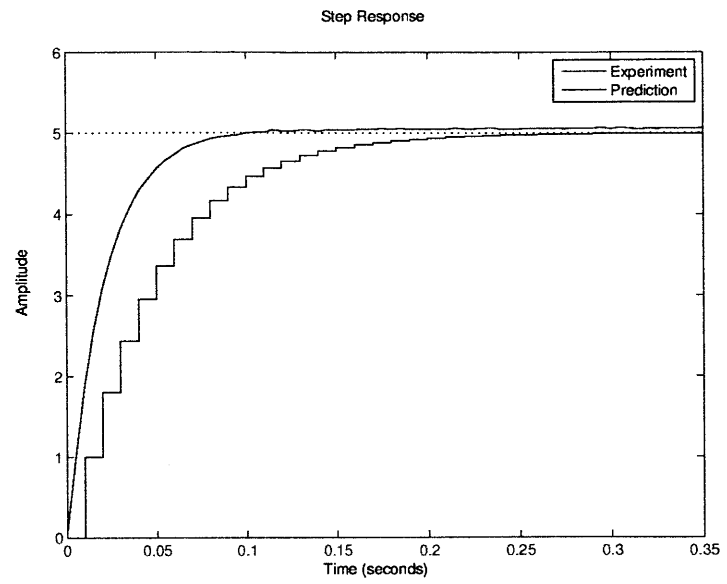


Figure 6-10: Experimental step response of the single integrator discrete-time proportional disturbance rejection system compared to the predicted step response.

Single Integrator Discrete-time Proportional Disturbance Rejection System

Step Response Our experimental step response for this system is shown in Fig. 6-10, along with the expected behavior. Again, the system is faster than expected when data is taken, with a time constant of approximately 0.45 s. The steady-state value of the experimental step response reached 5 V, as we predicted in the disturbance rejection system.

Sinusoidal Response We show the experimental sinusoidal response for this system in Fig. 6-11. Rather than the expected magnification of 1.5 at 11.6 Hz, the response during data saving was nearly unstable. However, as the screen-shot demonstrates in Fig. 6-12, the behavior when not saving matches the predicted behavior of the disturbance reduction system. Once again, the collected data was not accurate, though the screen-shot confirms the behavior we predicted. Again, we believe that the discrepancies between experimental and predicted behavior is due to the system

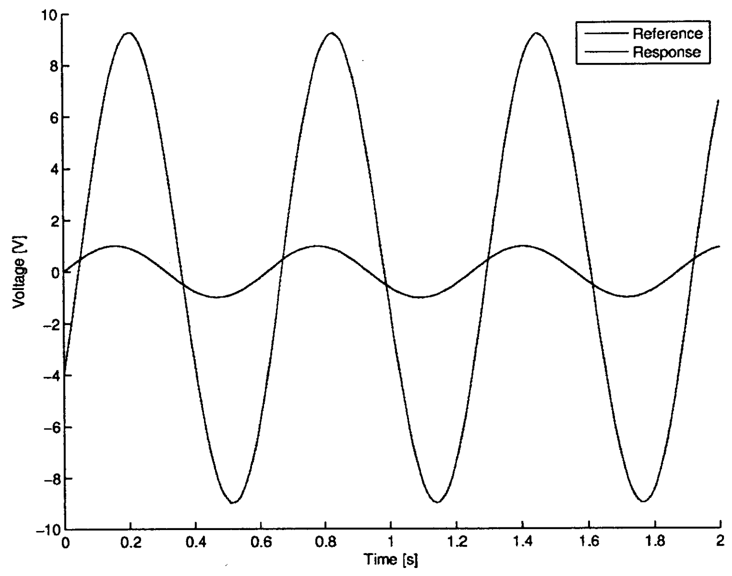


Figure 6-11: Experimental sinusoidal response of the single integrator discrete-time proportional disturbance rejection system compared to the reference input.

running slower when saving data, resulting in a time delay in the system.

Abnormalities and Hardware Notes

myDAQ Computational Delays We observed that the magnitude of the sinusoidal response was not as expected in nearly every case. While we do not fully understand why the experimental system did not match predicted sinusoidal response behavior, it could be due to a computational delay in the myDAQ hardware and software. The delay could change the sample time, or could represent the addition of a time delay to the behavior of the discrete system, though modeling such a delay would be more difficult than simply adding a $1/z$ term as the delay is less than the sample time.

We estimated the computational delay of the system using a bench-top function generator to provide a square wave to the myDAQ through an analog-in pin. Our software fed the square wave directly through the myDAQ and output the signal through an analog-out pin. Our setup represented the minimal computation required

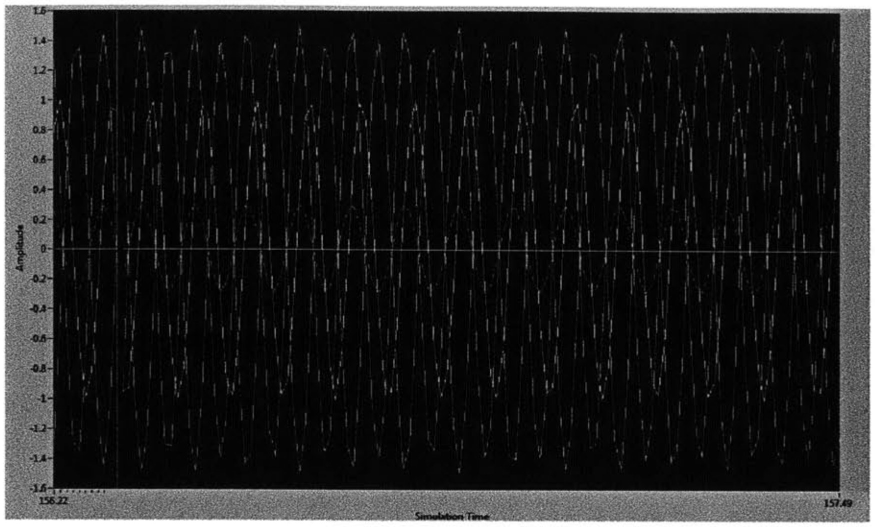


Figure 6-12: A screen shot of the experimental sinusoidal response of the single integrator discrete-time proportional disturbance rejection system compared to the reference input.

from the myDAQ when performing a control action. The time difference between the edges of the square waves is an estimate of the minimum time delay for computation. Our method estimated a delay of approximately 2 ms, which is 20% of the sample time for $T = 0.01$ s.

We observed other effects from the computational delay. For example, we noticed that the system did not reach deadbeat (when a discrete-time system reaches the reference value in one time step) at the gain predicted. We predicted that the single integrator discrete-time proportional control system would reach deadbeat when the gain was approximately $K_p = 1$. However, the system reached deadbeat when the gain was closer to $K_p \approx 0.6$. Furthermore, the system never actually reached deadbeat. As the gain was incremented at fine intervals, the system went straight from non-oscillatory behavior straight to oscillatory behavior. The lack of a deadbeat gain is further indication of the effects the computational delay is having on the control system. Finally, we observed that while the system was expected to go unstable at approximately $K_p = 4$, the system went unstable around $K_p \approx 2.4$. This is another example of the effect of the computational delay.

We also made an attempt to explore the time delay which is added when the myDAQ software is saving data to a file, as it is clear that a significant time delay is introduced while saving. We used the same method as outlined previously, and we found that the total time delay when saving data was 5 ms, which is half of the time step used by the myDAQ. The significant saving delay could very well explain the odd behaviors observed when saving data, and the inability of the system to match predicted performance during saving.

DC Offset When using proportional control in the single integrator experiment, we observed a DC offset of approximately 0.01 V when very low gains ($K_p \approx 0.01$) are applied to 1 V square wave inputs at low frequencies ($f = 0.5$ Hz). This is an addition and not a multiplication, as when the reference is negative, the offset is still in the positive direction. The offset is visible, for example, in the single integrator continuous time proportional control experiment, as demonstrated in Fig. 6-13. We do not expect to observe an offset in a first-order system, and we believe that it originates in the myDAQ equipment. Similar small offsets at low controller gains are common in lower-performance DAQ equipment.

6.2.2 DC Motor Experiment

We found the data from the DC motor system to be quite difficult to analyze, and due to noise the system is very jittery. The Hall effect sensor setup slowed the sampling of the speed of the motor, and these measurements also included an amount of noise. The sensor noise, low sampling frequencies, and lack of adequate counter pins in the myDAQ device made tuning and gathering data from the DC motor system difficult. If this experiment is to be used by students, we must find and implement more robust hardware and sensors, as well as more elaborate noise filtering techniques.

The data given below are the best estimates available from the noisy outputs of the DC motor system, and the dirty output signal is the chief contributor to the abnormal behavior of the experiments and their inability to match predicted performance.

Because the signals are dirty and the response of the motor so jittery, we did

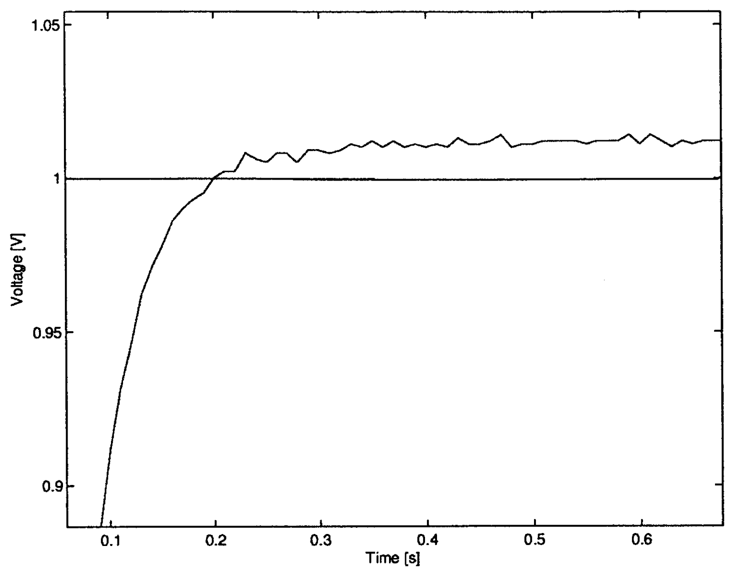


Figure 6-13: The DC offset which occurs in the response at low gains and frequencies.

not attempt sinusoidal response measurements. At the frequencies needed to observe interesting behavior, the sensor noise dominated the response, and damage to the gears was a real possibility. Different sensors and DAQ hardware will be necessary to create experiments capable of exploring motor dynamics.

Motor System Identification

Due to the problems with sampling frequency and sensor noise, the system parameters we describe are estimates. System responses which were not predicted in later experiments could be attributable to the lack of accuracy in finding system parameters.

Motor Constant k_m In order to find the motor constant k_m used in this experimental setup, we spun motor by applying a voltage to the motor generally used as a tachometer. We measured the voltage across the motor terminals using the analog-in pins of the myDAQ, while we measured the motor speed using the Hall effect sensor system described earlier in this paper. We saved the data to a text file, and we plot-

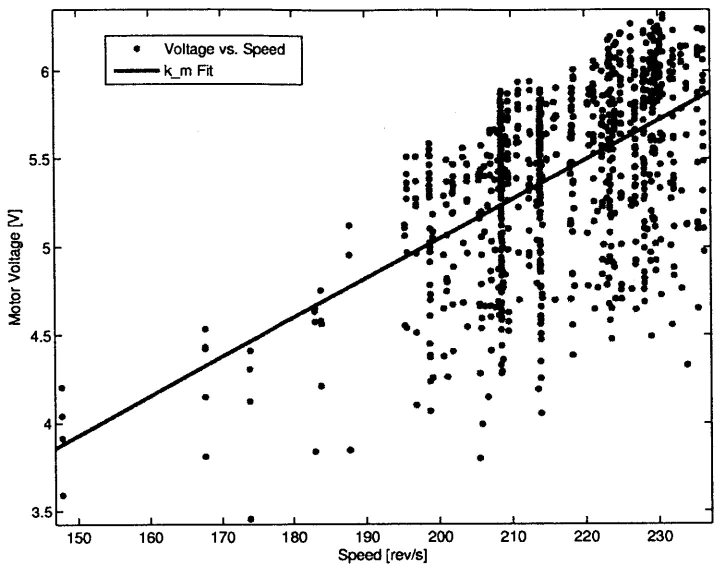


Figure 6-14: The best-fit line for the motor back-emf as a function of rotational speed. We used this line to find an experimental value for k_m.

ted motor voltage against motor speed. The slope of the best-fit line is k_m in V/rps, shown graphically in Fig. 6-14. We found that $k_m \approx 3.57 \times 10^{-3}$ V-s/rad.

Rotational Inertial J and Damping B We used the exponential best-fit of the proportional control response to a low-frequency square wave with $K_p = 0.01$ to find the values of J and B. The input was $\omega_r = 400$ rev/s. The best-fit curve is

$$\omega_o = 21\left(1 - e^{-48.3t}\right). \tag{6.1}$$

The best-fit curve is shown in Fig. 6-15.

We used the steady-state speed of the motor at a given voltage to estimate the value of B using Eq. 5.38. We found this to be $B \approx 3.95 \times 10^{-4}$ N-m-s/rad.

We found the effective rotational inertial J of the motor system, which includes the shafts of the motor and tachometer, as well as the flywheel coupler and other attached masses, using the time constant of the best-fit equation. We found the

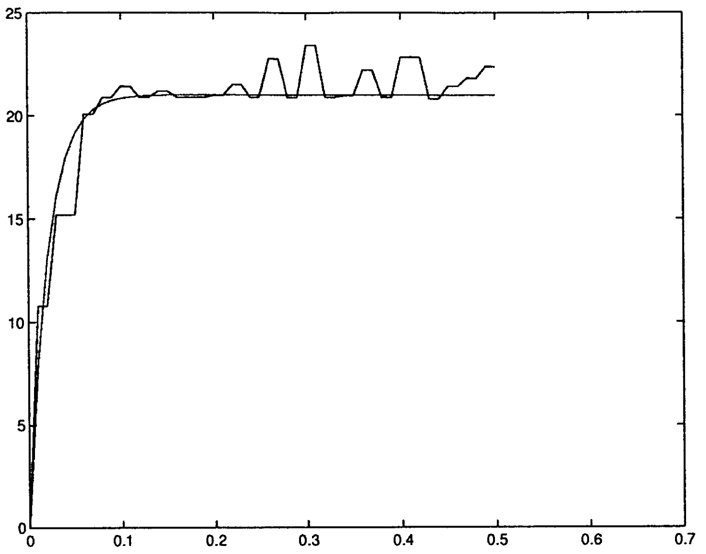


Figure 6-15: The best-fit exponential for the DC motor continuous time proportional control system with gain $K_p = 0.01$ and input $\omega_r = 400$ rev/s. We used the best-fit line to find experimental values of J and B.

effective inertia to be $J \approx 9.50 \times 10^{-6}$ kg-m^2.

Armature Resistance R_m We approximated the armature resistance, or the resistance of the windings in the motor, was found using a digital resistance meter. The meter uses a four-wire measurement to find the resistance. Our method yielded $R_m \approx 7.8\ \Omega$.

DC Motor Continuous Time Proportional Control System

We show a representative step response obtained using the gains outlined in Chapter 5 in Fig. 6-16. Since we used this experiment to identify the DC motor system parameters, the fit to the expected values is reasonably tight. However, in some instances, the system demonstrated an amount of second order behavior, with overshoot. This could be due to a noisy output signal, causing a spike in control effort and hence an overshoot of the reference. Also, the slow sampling rate of the speed sensor is

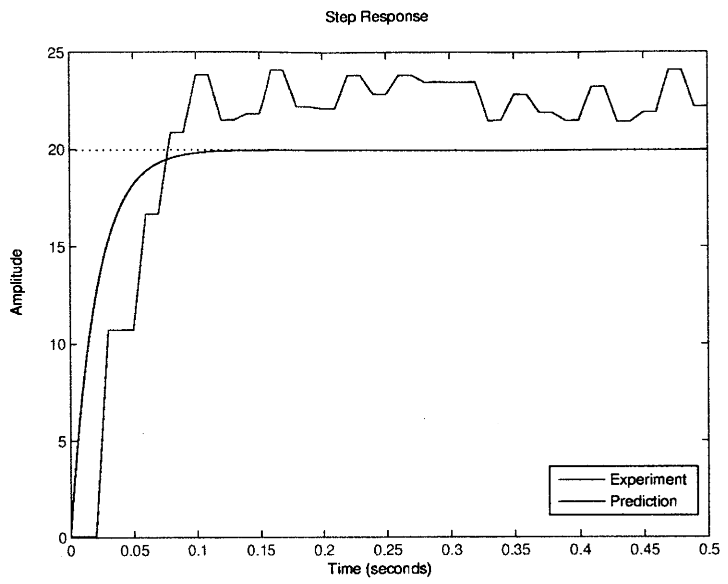


Figure 6-16: The experimental step response of the DC motor proportional control system.

obvious, and has a significant effect on the performance of the system.

DC Motor Discrete-time Proportional-Integral Control System

We show the predicted and experimental step responses for this system in Fig. 6-17. The integral control term successfully eliminates steady-state error. As is shown the graph, we did not expect the system to have any overshoot. However, the experimental system had a considerable amount of overshoot and subsequent oscillatory behavior. The overshoot and oscillations could be due to the sensor noise and sampling frequency difficulties, and also due to predictions based on inaccurate system parameters. It is obvious that the lack of regularity in the hardware makes control of the system difficult.

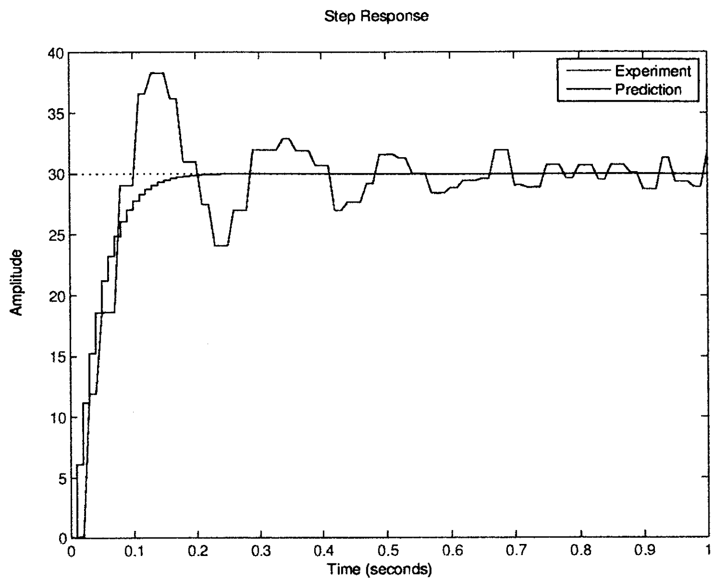


Figure 6-17: The experimental step response of the DC motor proportional-integral control system.

DC Motor Continuous Time Proportional-Integral-Derivative Control System

Figure 6-18 shows the predicted and experimental step responses of this system. As with PI control, we did not expect the system to have any overshoot, but should have a small amount of oscillatory behavior as the response climbs to steady-state. However, the experimental system had a considerable amount of overshoot and subsequent oscillatory behavior. This was not always the case, though, and sometimes the response did not include any overshoot. This is shown in Fig. 6-19, where we see that the experimental motor response follows the predicted behavior quite closely. Again, these irregularities and the unreliable behavior of the system could be due to the sensor noise and sampling frequency difficulties, and also due to predictions based on inaccurate system parameters. We can also see the effects of sensor noise and sampling frequency issues when the reference input was at zero. The motor system was very jittery and rarely sat still, as is seen in Fig. 6-20.

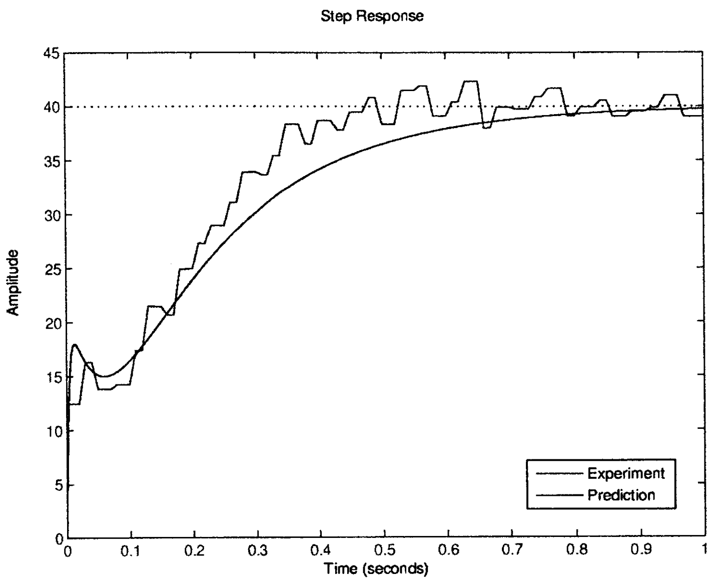


Figure 6-18: The experimental step response of the DC motor proportional-integral-derivative control system, showing overshoot.

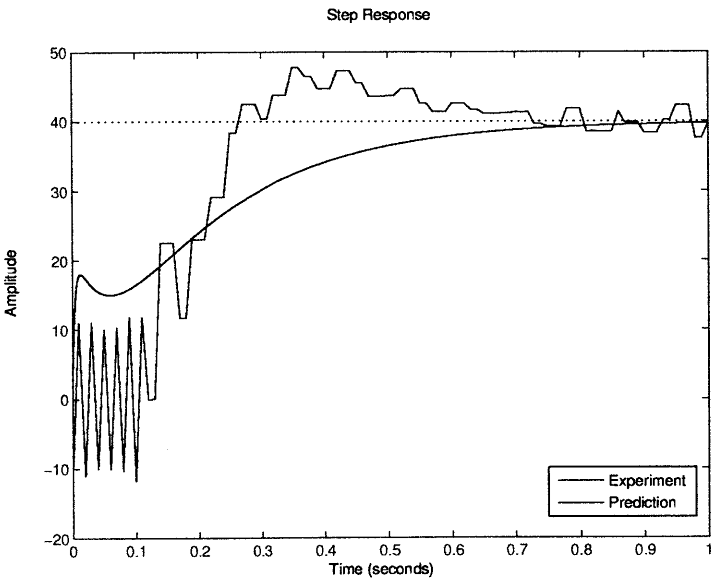


Figure 6-19: The experimental step response of the DC motor proportional-integral-derivative control system, showing a lack of overshoot.

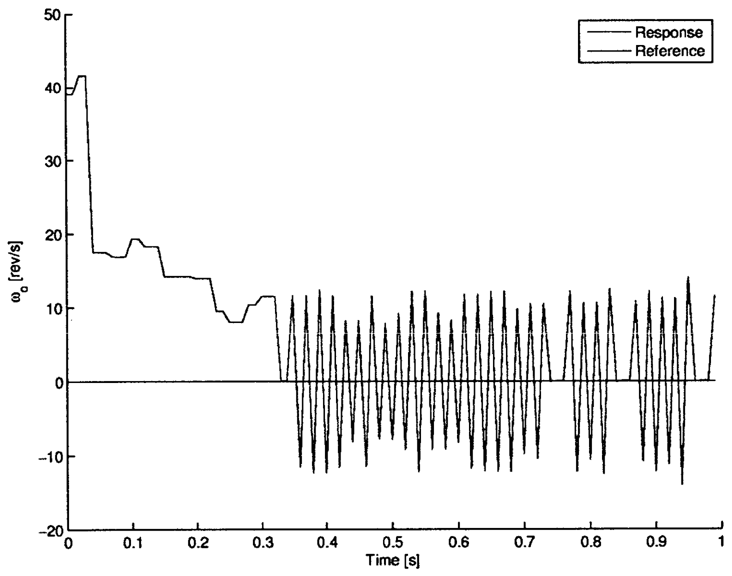


Figure 6-20: Jittery motor dynamics when the reference was held at zero, due to sensor noise and slow sampling frequencies.

Chapter 7

Conclusions

We documented the development of simple classical controls experiments which are portable and help students the time needed to gain controls intuition. Our experiments will allow for controls experimentation outside of the classroom by using the portable myDAQ and small experiments which fit on a single PCB.

We designed the control software we built in LabView to be easy for students to use, as well as simple for instructors to modify. The software provides reference inputs of varying types, visualizes the response of the system graphically, and saves data to disk for further analysis. The software uses controllers including P, PI, PD, and PID control in both the continuous and discrete-time domains.

We modeled, built, and tested an op-amp single/double integrator plant with reference, noise, and disturbance inputs. We also modeled, built, and tested a DC motor plant using a tachometer and Hall effect sensor for measuring motor rotational velocity. We use similar software for control, measurement, and system interaction in both systems.

We derived the expected behaviors of these systems for a series of representative controllers. We tested the experimental systems with the representative controllers and analyzed experimental system performance. We also observed irregularities in the hardware and software. While the op-amp system was reasonably well behaved, the DC motor system suffered from extreme sensor noise and sampling frequency problems. Hardware limitations including software time delays and low sampling frequencies, and sensor noise proved to be significant problems in building robust controls experiments. The motor experiment is not acceptable for classroom use in its current form, but provides the basis for further development. Finding better sensors and improving software design are possible solutions that could be researched in the future.

We will make the experiments designed available for public use on the Internet. To facilitate use by instructors, students, and self-learners, we produced write-ups documenting the control and modeling theory and the hardware and software development of both experiments. Our lab write-ups include user guides for students and instructors, and they also have lab and pre-lab problems for use in classes and by self-instructed learners. We intend the documentation, hardware, and software, to be easy to modify to suit the needs of its users.

Bibliography

[1] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems.* Prentice Hall, Upper Saddle River, New Jersey, sixth edition, 3 October 2009.

[2] James K. Roberge and Kent H. Lundberg. Operation amplifiers: theory and practice. Massachusetts Institute of Technology, Cambridge, Massachusetts, second edition, version 1.8.1, 19 April 2007.

[3] Derek Rowell. Introduction to the permanent magnet DC motor. Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, 25 February 2008.