DESIGN AND DIGITAL IMPLEMENTATION OF A PID CONTROLLER FOR A SIMULATED ROTARY KNIFE CUTTER

by

ANIRBAN MUKHERJEE

Submitted in Partial Fulfillment of the Requirements
for the degree of
Master of Science in Engineering
in the
Mechanical Engineering Program

YOUNGSTOWN STATE UNIVERSITY
December, 1998

DESIGN AND DIGITAL IMPLEMENTATION OF A PID CONTROLLER FOR A SIMULATED ROTARY KNIFE CUTTER

By

Anirban Mukherjee

Signature: Anirban Mukherjee 12.3.98

Anirban Mukherjee, Student Date

Approvals: [illegible] 12-3-98

Robert H. Foulkes, Thesis Advisor Date

[illegible] 12-3-98

H. W. Kim, Committee Member Date

[illegible] 12-3-98

K. V. Desai, Committee Member Date

[illegible] 12/8/98

Peter J. Kasvinsky, Dean of Graduate Studies Date

ABSTRACT

This investigation deals extensively with control system engineering in both theoretical design as well as direct implementation in both analog and digital terms. A rotary knife cutter, which is a commonly used piece of manufacturing equipment, is simulated in a laboratory environment using two independent servo-mechanisms placed side-by-side. The first servo simulates the infeed roll, while the other simulates the knife cutter. The object of this study is to use classical PID control, designed via Matlab and Simulink, to manipulate the overall system performance. Digital implementation is achieved using control codes programmed in MS-Quick C, interfaced through a PC and a digital-to-analog and analog-to-digital converter. The physical system and its simulation, the mathematical modeling of the servo-systems, and the design and implementation of the controller are all presented within the context of this study. Final implementation shows that direct digital implementation of the PID controller is adequate for the rotary knife cutter. However, there are several problems with the controller when switching from speed control to position control due to the controller design as well as the presence of noise in the system.

ACKNOWLEDGEMENTS

First and foremost, a great deal of gratitude goes to Dr. Robert Foulkes who served as my thesis advisor. His patience and willingness to teach me about controls engineering are immensely appreciated. Dr. H. W. Kim served on my thesis committee, and his willingness to allow me to explore thesis opportunities outside of the department helped greatly. Dr. K. Desai also served on my thesis committee, and his feedback on my work was quite valuable. Finally, Dr. D.H. Suchora and Dr. S. Pansino have provided me with a great deal of advice throughout my studies here at Youngstown State University, and also deserve my gratitude. My thanks go to all of these individuals.

Also, my appreciation goes to my fellow graduate students, Todd C. Werner and Syed Masud, for their help and support.

My ultimate gratefulness is for my parents and family, who have enthusiastically supported all of my academic undertakings. I share this accomplishment with them all.

DESIGN AND DIGITAL IMPLEMENTATION OF A PID CONTROLLER FOR A SIMULATED ROTARY KNIFE CUTTER

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

NOMENCLATURE - LIST OF SYMBOLS

SYMBOL	USES IN TEXT	UNITS
ξ	Damping ratio in the s-plane.	
γ	State-variable vector in the z-plane.	
ϕ,Φ	State-variable matrix in z-plane.	
τ_m	Motor time constant.	
$\theta(t)$	Rotational Displacement.	Degrees
$\dot{\theta}(t)$	Rotational Velocity / Speed.	Degrees/sec
$\ddot{\theta}(t)$	Rotational/ Angular Acceleration	Degress /sec^2
a	i) Real portion of a point in the Laplace domain. ii) A zero of a controller numerator. iii) Part of the Zeigler-Nichols unit step response. iv) A constant multiplier of a differential equation.	
A	i) State variable matrix in s-plane. ii) Magnitude of Lissajous pattern	
A2D / AD	Analog – to – Digital.	
b	i) Imaginary part of a complex number. ii) A zero of a controller numerator.	
B	i) State variable vector in s-plane. ii) Magnitude of Lissajous pattern.	
c	Damping ratio.	Ns/m
C	i) State variable vector in s- or z-plane ii) Magnitude of Lissajous pattern.	
CL	Closed loop function.	
d/dt	Time based derivative of a continuous function.	
D	State variable vector in s- or z-plane.	
D2A / DA	Digital – to – Analog.	
E, e(t), e	Plant response error.	
f	Frequency.	cycles^{-1}
$f(kh)$	Sampled data function.	
f(t)	Time based continuous function.	
F(s)	Function in Laplace domain.	

NOMENCLATURE – LIST OF SYMBOLS (CONT'D)

SYMBOL	USES IN TEXT	UNITS
F(t)	Forcing function.	
F(z)	Function in z-domain, z-transform	
G(s), G(z)	Plant transfer functions.	
h	Sampling frequency / period.	milliseconds
H(s), H(z)	Controller transfer functions.	
Hz	Frequency.	Hertz
$i_a(t)$	Motor circuit armature current.	Amps
I	Identity matrix.	
j	Imaginary number $\sqrt{-1}$	
k	Spring constant.	N/m
K_{PID}, K_c	PID controller gain.	
K_P	i) Proportional gain. ii) Motor model integrator gain.	
K_I	Integral gain.	
K_D	Derivative gain.	
K_U	Ultimate gain.	
K_t , K_e	Motor model constants.	
K_m	Tachogenerator gain.	
$\mathcal{L}$	Laplace Transform.	
L	i) Zeigler-Nichols apparent deadtime. ii) Motor model apparent inductance.	
M	Mass.	Kg
N	PID controller approximate derivative term.	
PID	Proportional, Derivative, and Integral (controller)	
r, r(t)	Input or reference signal.	
R	i) Input or reference signal. ii) Steepest slope in Zeigler-Nichols. iii) Servo modeling armature resistance.	
s, s′	Point in the s-plane in Laplace domain.	

NOMENCLATURE – LIST OF SYMBOLS

SYMBOL	USES IN TEXT	UNITS
T_i	Integration or reset time.	
T_d	Derivation time.	
T(t)	Rotor torque output in motor model.	
T_U	Ultimate or critical period.	
U, u(t), u	Controller output signal.	
upre	Control signal pre-calculation.	
V, v(t)	Voltage output or signal.	Volts
V_a	Tacho input signal during modeling.	"
V_g	Tacho output / integrator input signal during modeling.	"
V_p	Potentiometer /integrator output signal during modeling.	"
x(t)	i) Linear displacement in direction of x-axis. ii) General time based continuous function.	
$\dot{x}(t)$	i) Linear velocity. ii) First derivative of time based function.	
$\ddot{x}(t)$	i) Linear acceleration. ii) Second derivative of time based function.	
x1, x2	Controller states.	
Y, y(t), y	Plant output / response.	
z	Point in z-plane (discrete domain).	
Z	Z-transform.	

LIST OF FIGURES

LIST OF FIGURES (CONT'D)

LIST OF FIGURES (CONT'D)

LIST OF TABLES

CHAPTER 1 -THE ROTARY KNIFE CUTTER – THE PHYSICAL SYSTEM AND ITS SIMULATION

The Rotary Knife Cutter

Rotary knife cutoff applications are commonly used in industrial processes where cut-to-length operations are mandated. Cutters usually consist of a combination of a knife assembly and either a conveyor roll feeder or a hardened anvil roll. Cut-to-length operations are commonly used in a variety of strip-material processing applications such as product extrusion cutoff, paper and film cutting, rubber tire belting cutoff, and notching paper insert products, with industrial cutters generally being robust enough to handle a variety of film, foil, strip, paper, and textile. Many rotary knife cutters are fully self-contained units that are capable of emitting finished products dependent on manufacturing specifications. Conversely, rotary cutters are also designed as stand alone units, which are usually integrated into more complex material processing lines. Commonly, a rotary knife cutter accomplishes a cut by crush cutting, or shearing, material that is being fed through by roll feeders, such as the schematic given in Figure 1.1. A desirable capability of any rotary knife cutter that operates for cut-to-length operations is the ability to have adjustable parameters that explicitly control the length of the finished material. A certain type of achieving such control, known as classical PID control, is the primary focus of this study and is dealt with from a practical design and implementation viewpoint. Topics such as mathematical modeling, design techniques, and implementation are presented.

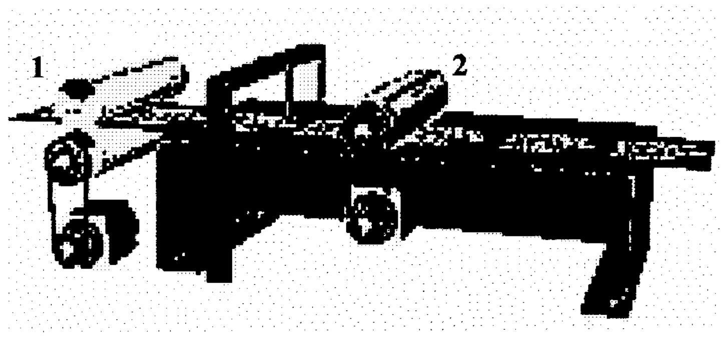


Figure 1.1: Representative schematic of rotary knife cutter. Strip material fed through roll feeders at (1) is cut-to-length by a shearing knife mounted on a rotary cylinder at (2). (photo courtesy ORMEC Systems Corp.) [1]

[1] All photographs and graphics used are copyright, when noted, of said company, and are taken from publicly accessible web servers. Graphics courtesy of ORMEC Systems Corporation are subject to explicit copyright notices as given at www.ormec.com. All copyrighted photographs and graphics have been used for non-commercial, non-profit purposes.

Simple Mechanical Control

The basic mechanism of a rotary knife cutter used for cut-to-length operations is the idea of utilizing a two-axis servo drive. One drive is used to control the feeding process, while the other is used for the rotary cutter. Consider Figures 1.2 and 1.3, which respectively show an independent modular rotary cutter and its schematic. The premises

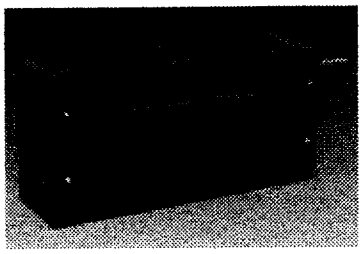


Figure 1.2: A stand alone one-to-one modular rotary cutter used in industrial applications. The rotary cutter has an extended shaft on the cutting cylinder such that a timing pulley or gear can be incorporated to achieve a one-to-one cut. (photo courtesy AZCO Corp.)

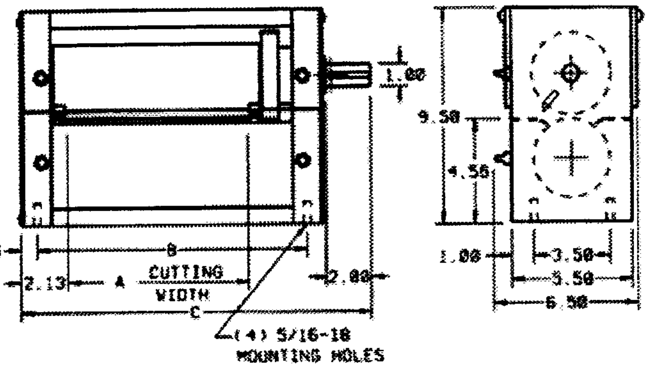


Figure 1.3: Accompanying schematic for rotary cutter shown in Figure 1.2. The cutting action takes place when the knife blade mounted on the topmost cylinder (shown on right) meets up with the hardened anvil roll on the bottom during rotation, causing the material to shear. For this particular model, the cutting width A varies from 6” to 32”. (Photo courtesy AZCO Corp.)

for the cutting process can be seen from the schematic given in Figure 1.3. As the knife blade, mounted on topmost cylinder, rotates against the bottom hardened anvil roll, the material in between is sheared. One noticeable aspect of this rotary cutter is the fact that there is an extended shaft on the cutting cylinder. This is used to handle a timing pulley or gear to coordinate the feeding and cutting process, thereby using what can be termed as strict one-to-one control where the feeder-servo is mechanically linked to the rotary cutter. Conceivably, the rotary cutter can then be thought of as a *dependent* system, since for one revolution of the feeder, the rotary knife cutter achieves only one cut. Thus the sole determining factor for the length of the finished material is the speed of the infeed material passing through the knife roll, thereby making the process *dependent*.

However, there are certain cut-to-length manufacturing processes where the cutting process is vastly more complex. An example of this can be seen from Figure 1.4, which shows a multiple knife system mounted on the rotary cutting cylinder. This particular rotary knife cutter is used in dry food processing lines, mainly for corn chops and processing peppers, spices, and roots, and is composed of eight rotating knives on the cutter cylinder. Of importance is the fact that such complex rotary knife cutters are not usually controlled by simple mechanical control, as discussed previously via a feeder-servo and cutter linkage. Even though in this case one-to-one mechanical control can suffice, depending on the process itself, a greater variety of possible cutting techniques for changes in the cut-to-length product can be introduced using implementation of full electronic control. A look at a typical cutting process can illustrate the importance of electronic control in a rotary knife cutter.

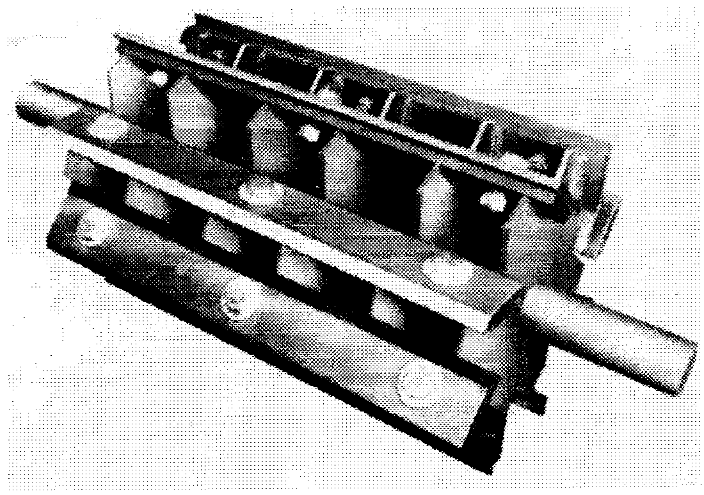


Figure 1.4: A Davis Rotary Knife Cutter used in the food processing industry. This cutter is manufactured as a complete modular unit, and is comprised of eight knives mounted on the rotary cutter, as well as two downstream stationary cutters. (Photo courtesy H.C. Davis & Sons Manufacturing Inc.)

Independent Control and the Cutting Process

The cutting process for a rotary knife cutter for cut-to-length applications can be thought of as either a *dependent process*, where the material infeed and the knife axis are mechanically linked (as described in the above section), or as an *independent process*, where no mechanical linkage is needed for rotary cutting. Consider Figure 1.5, which shows a rotary knife cutter in a modular unit, where the speed of the material being fed is controlled by a servomechanism (1) independent of the servo-system controlling the cutter (2). The benefit of such a system is that the infeed-servo feeds material at some rate that has no relevance to the cut-to-length operation, allowing for the infeed rolls to be a part of a much larger conveyor system within the processing line. The determining

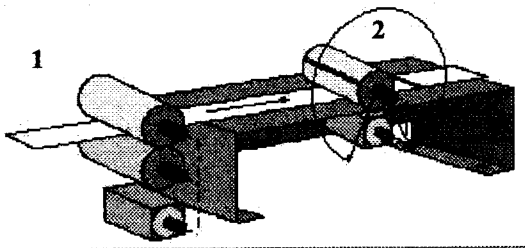


Figure 1.5: A mechanically independent rotary knife cutter. The infeed-material is controlled by a servo system (1) that has no direct relevance on the servo system controlling the knife cutter (2). Electronic control of the cutter-servo is solely dependent on the finished product length. (Photo courtesy ORMEC Systems Corp.)

factor for controlling the rotary knife cutter lies in the prescribed length needed for the finished material. Thus, the cutting process in *independent* of the constant speed at which the infeed material passes through the knife cutter. Ultimately, this type of a process allows for greater flexibility in placing the rotary cutter in the processing line since the cutter can be implemented as an autonomous process irrespective of upstream or downstream operations. Versatility is also achieved in the cut-to-length operation in the sense that the mechanical independence of the cutter gives it the ability to handle a much greater range of cut-to-length processes.

The independent cutting process introduced above can be better explained with the use of Figure 1.6, which shows a typical rotary single-knife cutter. The infeed rolls (1) feed the strip material through the processing line at some given constant speed. The length of the finished product designated by *target points* (2) on the infeed strip material. As a single *target point* traverses through the knife cutter area, the motion of the knife mounted on the cutter is carefully controlled such that while it rotates, the knife-edge precisely meets the *target point* and the strip material is sheared. This action is implemented using synchronous control of either the rotational speed or angular position or the knife blade through a predefined *cutting zone* (3) given by the *wrap angle* θ [2]. Immediately following the cutting process, the knife blade must be either retarded or increased in speed in order to catch the next *target point.* This part of the overall process is known as the *correction zone* (4), and is dependent on the amount of material the infeed roll must pass through for the subsequent *target point* to be lined up for the cutter. The successful completion of two successive *target points* passing the cutter indicates one finished cut-to-length product. Finally, this cut-to-length product is usually allowed

[2] Industrial terminology such as *synchronous cutting zone, wrap angle, correction zone, etc.* are used as given in the Technical Brief section on rotary knife cutters at www.ormec.com, which is copyright of ORMEC Systems Corp.

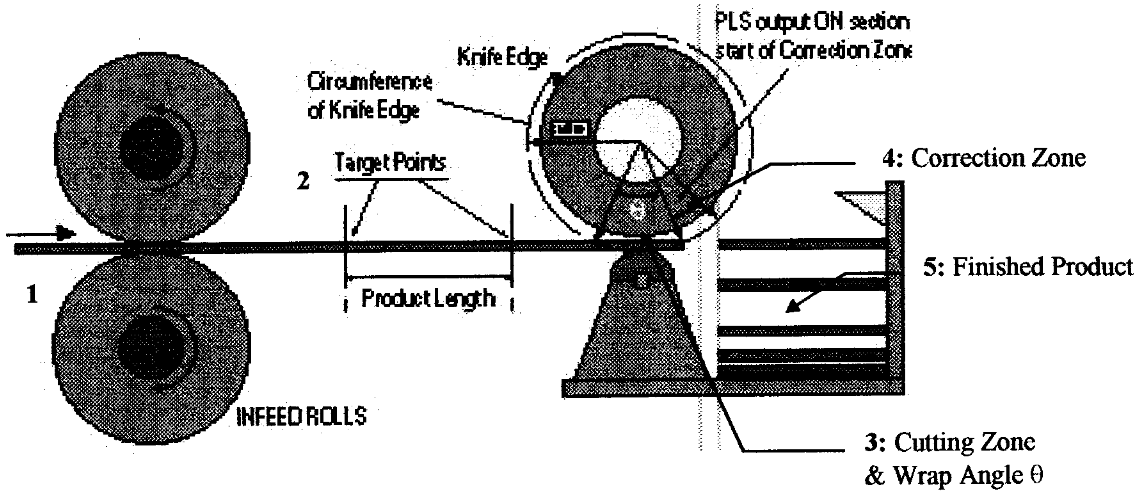


Figure 1.6: Schematic representing the cutting process. The infeed rolls (1) determine the amount of material that is fed through to the rotary cutter. The length of the finished product is set by target points (2) on the infeed material moving at constant velocity on the process line. With precise control of the rotation of the knife blade through the cutting zone (3) for the duration of the wrap angle θ, the knife blade is able to meet the target point, shearing the material. In the correction zone (4), the rotary cutter is either increased or decreased in speed, allowing it to enter the cutting zone again for the next target point, completing one finished cut-to-length product, which is stored in large stackers (5) at the end of the line. (Photo courtesy ORMEC Systems Corp.)

to progress downstream for additional processes, or stacked in bins for storage.

Thus, it is seen that the motion of the rotary knife cutter in this process is dependent on the product length, and, although somewhat dependent on the speed of the infeed servo, is mechanically independent in its behavior. This particular process can be implemented using mechanical devices such as an independent servomechanism and a system of timing belts and pulley. However, it is usually achieved in industrial applications with the use of electronic control devices, and is described in greater detail in the following section.

Electronic Control

The cutting process discussed above is widely used in industrial applications where a rotary cutter is implemented in modular form as part of a larger processing line. In such applications, electronic control is preferred over mechanical control because it can provide a greater flexibility of the finished product and ease of manipulation by using digital real-time process control. Electronic control in this sense means the acquisition and transferal of data during the actual process, which is needed to produce the correct physical behavior of a system. Mechanical components remain an integral part of the process since they comprise the system that achieves the final work output. However, final control is implemented using electronic data acquisition, which in this sense refers to digital, or discrete, data, using modern day microprocessors and to some degree, personal computers (PC's). [3]

As an example of the above statements discussing the idea of electronic versus mechanical, consider taking measurements of the rotational position of a point on the output shaft of a motor. Mechanically, visual measurements can be taken using some optical device of the position at some given point during rotation. However, the accuracy is solely dependent on the precision of the mechanical device and the analog components comprising the data acquisition system. Conversely, if a rotational potentiometer is attached to the output shaft, and the voltage variation across the potentiometer is measured during rotation, the position of any given point can be determined by analyzing the voltage output. Electronically, the accuracy of this output will depend on the frequency of the sampling of the output signal. Using modern day digital data acquisition systems, it would be possible to sample the output signal of the shaft to a much greater accuracy than would be possible by using a combination of mechanical sensing devices and analog data acquisition systems. Essentially, digital data processing, in modern times, has become much faster, accurate, and easier to manipulate than its mechanical and analog predecessor, such that a vast majority of process control now utilizes some form of digital microprocessor data processing.

In the case of a rotary knife cutter, where control of high-speed, continuous rotary motion is desired, electronic control offers a very powerful method of process control. Figure 1.7 shows a rotary knife cutter where the application is implemented using a dual axis digital signal processor (DSP). The DSP samples electronic data from the infeed and cutter servos. The PC based motion controller housed within the DSP executes the

[3] The idea of analog and digital systems in relation to control systems is discussed in detail in the following chapter.

control sequence, which is dictated by the sampled data from the servo systems. After the control sequence is established, the DSP outputs the necessary control signal to the knife cutter servo, such that precise control over the cutting zone, and subsequently, the correction zone, is attained. This procedure of sampling data, generating the control sequence, and outputting the control signal to the cutter servo, is placed in a continuous programmed loop such that real-time process control can be achieved.

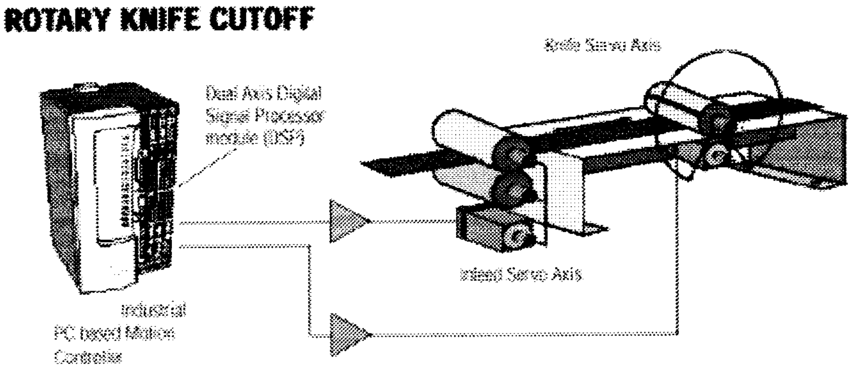

ROTARY KNIFE CUTOFF
Knife Servo Axis
Dual Axis Digital Signal Processor module (DSP)
Infeed Servo Axis
Industrial PC based Motion Controller

Figure 1.7: Electronic process control for the rotary knife cutter shown is implemented using a dual axis digital signal processor (DSP) module. The DSP acts as an analog-to-digital / digital-to-analog converter. Microprocessors within the DSP module process the control strings in between sampling electronic data from the infeed and cutter servos. (Photo courtesy ORMEC Systems Corp.)

Modern day industrial process control for rotary knife cutters are robust enough to offer simple operating parameters such as touch screen interfaces for setting product lengths, as well as on-the-fly product length adjustments. The latter implementation takes advantage of the DSP's ability to queue data such that interruption of the programmed control sequence when changing cut-lengths has no noticeable effects on the flow of the process line. Process control is usually accomplished using a servo controller such as the one shown in Figure 1.8. The M2400 Servo Controller manufactured by Extratech Corporation uses an Intel 32-bit microprocessor in controlling up to 4 coordinated axes of motion. The control algorithm is a discrete version of PID control with velocity and acceleration feed-forward capability. The controller also offers such features as linear, circular, and smooth curve interpolation, unlimited

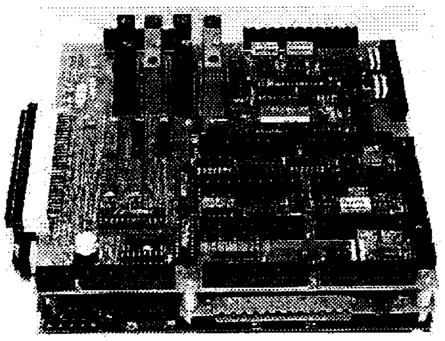


Figure 1.8: The M2400 Servo Controller, manufactured by Extratech Corporation utilizes an Intel 32-bit microprocessor and can control up to 4 coordinated axes of motion. The controller interfaces with a PC through a controller board and cable. (Extratech Corporation)

file sizes, built-in self testing, as well as a PC based graphical tuning program. Other than implementation in a rotary knife cutting operation, the M2400 is also used in processes for plotters, engravers, routers, laser cutting, and water-jet cutting.

Laboratory Simulation of a Rotary Cutter

The primary focus of this study is to design, simulate, and digitally implement a PID controller for a rotary knife cutter. The objective is to evaluate the practicality of a direct digital implementation using an analog design, as well as to assess a cutter with a faster rate of cutting. However, in order to design and implement such a controller, it is first necessary to have an actual physical system. Considering unavailability of an actual industrial rotary knife cutter, an equivalent system is simulated using available servo systems used in the controls engineering laboratory of the Department of Electrical Engineering. Two independent servo systems are placed side-by-side, where one represents the infeed servo, while the other represents the cutter servo. A schematic of the setup is shown in Figure 1.9. In its basic form, each system consists of a 15V-power unit, an attenuator, pre-amp, and op-amp unit, and a tacho-generator. The output of the tacho-generator is mechanically attached to a 30:1 gear reducer, whose output takes the form of an electrical signal generated by a rotational variable resistance potentiometer, which

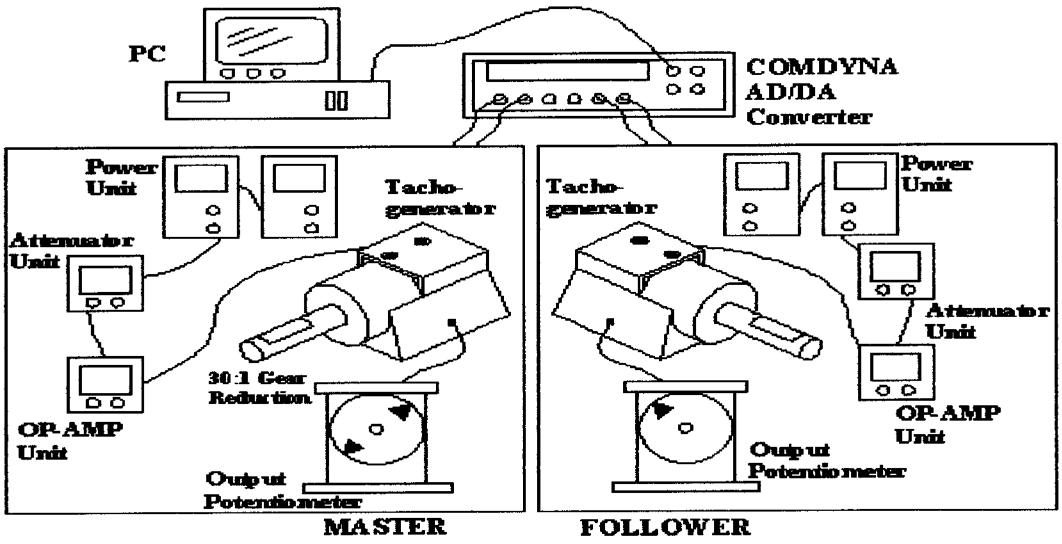


Figure 1.9: The rotary knife cutter is simulated in the laboratory using two servo systems placed side-by-side. Position information is taken from the output shaft of the tacho-generator after the output goes through a 30:1 speed reduction. The infeed rolls , or the cut-length, is represented by the output of the left servo system, labeled as the MASTER, while the right servo system represents the knife cutter, and is identified as the FOLLOWER. Arrows placed on the output disks of each servo system mark target points for the infeed material and the knife blade.

represents the angular position of the output of the servo system. This position output is visually depicted using a disk mounted on the output shaft, so that as the tacho-generator rotates at some high speed, the disk rotates at a speed which is thirty times less and generates a signal depicting this rotation.

The analogy for the rotary knife cutter, in terms the simulated system is best described by the following. The left servo system, labeled the MASTER [4], takes the form of the infeed rolls, or the cut-length. Target points for cuts, placed initially on the material during its motion on the conveyor line, are marked as arrows located 180° apart on the rotating output disk of the servo system. Essentially, for every rotation of the infeed rolls / output disk, there are two target points that pass through the cutter area. The rotary knife cutter takes the form of the servo system located on the right, denoted as the FOLLOWER. Similar to the MASTER, this servo system also has an output disk, on which a single arrow represents the knife blade.

For purposes of controlling the MASTER-FOLLOWER system digitally, several sets of electronic data need to be manipulated. In the laboratory simulation, this corresponds to sampling several electrical signals from both the MASTER and FOLLOWER. Since voltages representing the rotational speed (from the tacho-generator) and angular position (from the output potentiometer) are important indicators of system characteristics, both of these signals are fed into a COMDYNA analog-to-digital / digital-to-analog unit. The COMDYNA, acting as the DSP interface, feeds the digital equivalent of each of the signals into a DAS-8 board located internally in a personal computer, where the system signals and control sequences are analyzed. The control signal for the rotary cutter is output back to the COMDYNA and ultimately, to the FOLLOWER. Hence, the entire setup of the MASTER-FOLLOWER servo system, the COMDYNA AD/DA interface, and the PC can be thought of as representing the rotary knife cutter.

[4] For the purposes of this study, and for simplicity, the entire system of the two servo systems is labeled as a MASTER-FOLLOWER system. Any reference to this nomenclature designates the two servo systems. The infeed roll is called the MASTER and the knife cutter is called the FOLLOWER.

Having established the concepts behind the laboratory simulation of the overall system, the physical characteristics of the control that is to be implemented needs to be discussed. Figure 1.10 shows a schematic of the output disks of the MASTER-FOLLOWER system. The disk for the MASTER is marked with arrows placed 180° apart, representing the target points for the cut-lengths. The FOLLOWER is marked with one arrow representing the knife blade. For correct cutting, the control sequence must show the capability of having the FOLLOWER arrow (knife blade) track, either via speed or position, the target point that is passing through the cutting zone denoted with the hatch marks. Immediately following the end of the cutting zone, the controller must utilize the entire correction zone to force the FOLLOWER arrow to catch up to the next target point before the next cutting zone by speeding up the FOLLOWER. Subsequently, the FOLLOWER arrow must once again track the target point through the cutting zone. A successful completion of one such control sequence will indicate a complete, cut-to-length finished product.

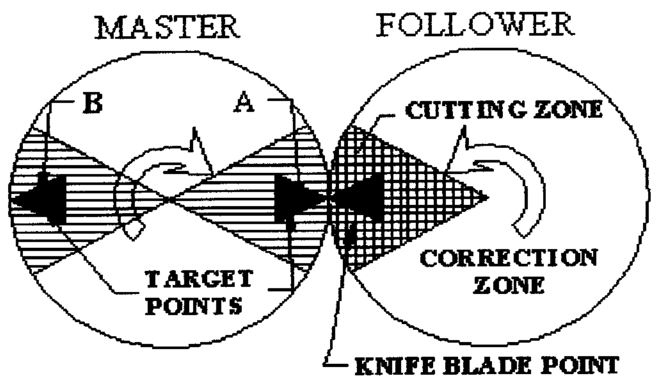


Figure 1.10: Schematic of output disks of MASTER-FOLLOWER system. The cut-lengths of the material are marked as target points A and B on the MASTER, while the knife blade is marked on the FOLLOWER. The FOLLOWER must track a target point through the entire cutting zone, and then use the correction zone to catch up to the next target point immediately before the next cutting zone.

There are several points to consider about the above control strategy at this time. First and foremost is the fact that the two servo systems are completely independent of each other, both mechanically and electronically (initially), although the performance of one is directly contingent upon the other. This will, in most probability, result in slight differences in the physical characteristics of each of the servo systems. Also, for such types of process control in industrial applications, tight tolerances are usually expected. However, since this is a makeshift laboratory simulation, there may be instances where industry-typical tolerances cannot be met, mainly due to several constraints within the laboratory equipment itself. Finally, much of the emphasis of this study is placed on the design and implementation of the controller, at the expense of mechanical factors, such as the internal friction of the motor. Also, intricacies in the electronic circuitry, which also may have influences over physical characteristics of the system performance is not accounted for in this study.

CHAPTER II - PID CONTROL & ANALOG / DIGITAL SYSTEMS– THEORY AND BACKGROUND

Theory of PID Control

Proportional, integral, and derivative (PID) control is used widely in many industrial applications. PID control offers an easy method of controlling processes by varying the PID parameters, even though many industrial controllers often have poor tuning characteristics. Originally, PID controllers were implemented using available analog technology, but have evolved to encompass pneumatic valves, relays, transistors, and recently, integrated circuits. However, most industrial based PID controllers are now implemented digitally, and can include high levels of complexity in their design and performance. [1],[3]

The basic concept of PID control is represented by Eqn. 2.1, which shows the classical "textbook" equation for a PID controller. The parameter *u(t)* is the control signal, and *e(t)* is the control error (difference between the system input/ setpoint and the

Eqn. 2.1 $$u(t) = K_c\left[e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de}{dt}\right]$$

system output/ response). K_c is the control gain, T_i is the integration or reset time, and T_d is the derivative time. It is important to note that this particular form for a PID controller can (and often is) structured in other forms, which utilize other parameters. One such form, using the PID gains, is used throughout this text and will be shown later. The PID output *u(t)* for this from can be separated into individual terms (Eqn.'s 2.2 – 2.4), each of which provide adjustable parameters for tuning the operation of the PID controller. Each

Eqn. 2.2 $$u(t) = K_c e(t)$$ *Proportional Control*

Eqn. 2.3 $$u(t) = \frac{K_c}{T_i}\int e(t)dt$$ *Integral Control*

Eqn. 2.4 $$u(t) = K_c T_d \frac{de(t)}{dt}$$ *Derivative Control*

type of control can be implemented individually, and although interrelated, serve their own distinct purpose. Proportional control (Eqn. 2.2) is used to subject the control signal *u(t)* to an instantaneous change when there is an error. As its name suggests, any change in the control signal is directly proportional to the change in an error signal for a given gain K_c. The integral control portion (Eqn. 2.3) acts to eliminate steady-state error that may be caused by the action of the proportional control, where smaller integration times (T_i) result in faster changes in the control output *u(t)*. However, even though this type of control is effective in eliminating steady state error, it usually takes a longer period of time to process, since integral control must act over a period of time (the limits of the actual integral). Finally, derivative control (Eqn. 2.4) is used to track the changes of error over time, and is effective in controlling the overshoot of the system response. Note that derivative control is only implemented when the error is changing with time. [2],[3],[11]

To determine the effects of PID control in a system's response, the characteristics of a response are discussed. Figure 2.1 shows a typical system response to a unit step input. The characteristics of the system response clarify what physical parameters are actually controlled. For example, rise time and settling time depict how fast the system responds initially to the input, and is called the *transient* response of the system. Physically, overshoot corresponds to the damping attributes of the system, while the steady state error indicates

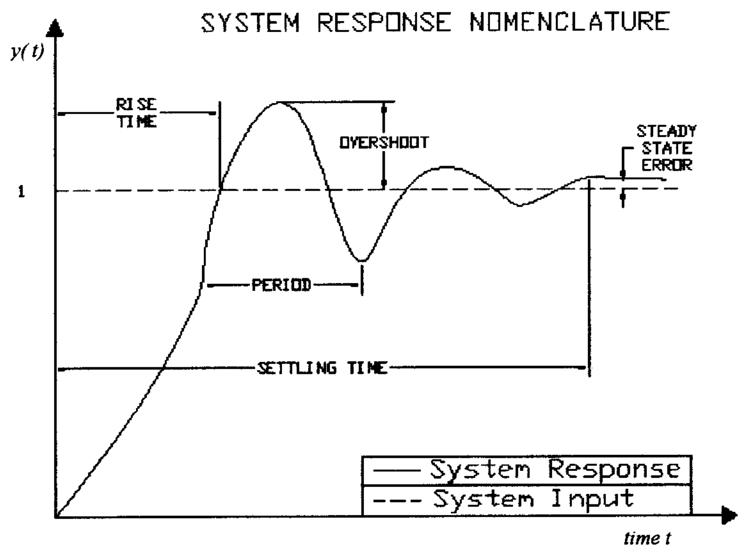


Figure 2.1: Typical 1st order system responses for a given step input includes such parameters as rise time, settling time, overshoot, and steady state error. The object of the controller is to make the system response behave in an acceptable manner.

the system performance in the post-transient response phase. It is the ultimate goal of the PID controller to be able to control each of these parameters in some optimal manner by tuning the PID parameters mentioned above.

In control systems, there are usually two types of systems that are commonly discussed. Figures 2.2 and 2.3 show typical block diagrams of *open loop* and *closed loop (feedback)* systems. Figure 2.2 shows a typical open loop system where the input to the

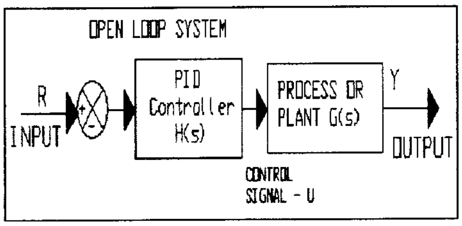


Figure 2.2: Typical open loop system. The input for the system is fed directly into the controller, and no measurement of error is taken. Full use of PID control cannot be achieved.

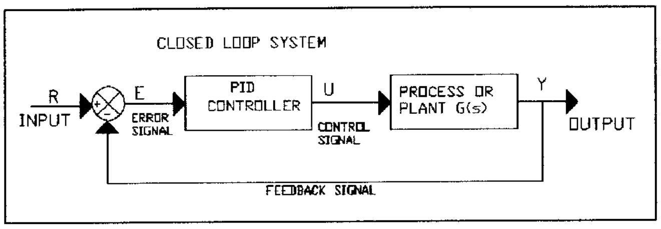


Figure 2.3: Closed loop system with implemented PID controller. The control signal *u(t)* comes from the PID process and the error signal obtained through closing the loop (feedback).

system *R* is fed directly into the controller, which outputs a control signal *U* used to drive the system. Although open loop control may be adequate, this system does not measure the error of the system, and the full potential of the PID control cannot be utilized. Thus, the overall system performance depends greatly on the interaction between the two blocks, and slight variations in the system characteristics may lead to inadequate performance or even instability. However, if the PID controller is implemented in a feedback system shown in Figure 2.3, the error can be controlled with greater ease. By feeding back the output signal, the error (the difference between input *R* and output *Y*), can be directly measured via a summing junction. The feedback loop, or the error signal, is very important to the correct performance of the PID controller.

Having discussed the PID controller, it is necessary to point out concepts in the other blocks in the system. Notice that in the block diagrams of the above two figures, the process or plant is described as a function *G(s)*, and the PID controller is described as a function *H(s)*, both of which are input-output transfer functions. When dealing with control systems, continuous time systems that are initially mathematically modeled as differential equations usually take the form of transfer functions (output over input functions) via the use of Laplace transforms. Laplace transfer functions will be used extensively in this study, since they provide the advantage of separating all entities (inputs, outputs, etc.) algebraically. The general form for the Laplace transform is given below, where the time domain is mapped into the Laplace domain, where s is a complex variable of the form $s=a+bj$. Thus, by knowing the continuous time function $f(t)$, and if

$$\text{Eqn. 2.5} \qquad L[f(t)] = F(s) = \int_{0^-}^{\infty} f(t)e^{-st}dt$$

the integral converges, a form for the Laplace function $F(s)$ in the complex plane can be found, providing a very useful mathematical tool from which control systems can be analyzed. A typical manner of representing a PID controller as a transfer function in the Laplace [1] domain is shown in Eqn. 2.6. This allows for the controller to be tuned by

$$\text{Eqn. 2.6} \qquad H(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

varying K_P *(proportional gain)*, K_I *(integral gain)*, and K_D *(derivative gain)*. Each of the parameters is defined by taking the Laplace transform of each term in Eqn. 2.1 such that the above form is achieved. [11]

Without going into further detail of Laplace transforms and PID control, a simplified example illustrating both these concepts is presented here. Consider Figure 2.4, which shows a simple mass-spring-damper (MSD) system with a forcing function *F(t)*. This is a classical continuous time system with a system input *F(t)*. The uncontrolled system response is directly dependent on various physical parameters such as system

[1] Further details of Laplace transforms, and in particular, transforming the PID equation of Eqn. 2.1 into the form of Eqn. 2.6, can be found in any controls textbook, some of which are listed in the bibliography of this work.

mass, spring stiffness, etc. The objective of the system response would be to track the input (force) using a feedback PID controller.

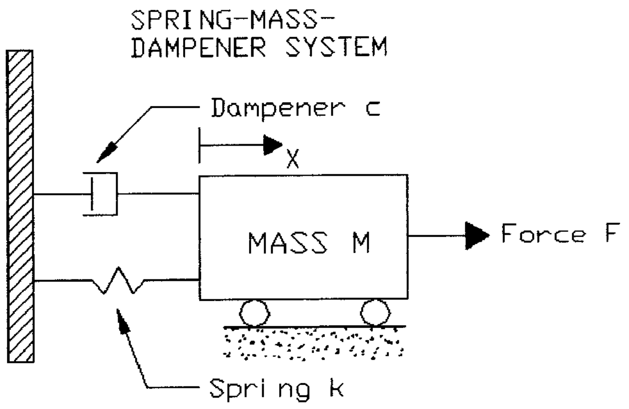


Figure 2.4: Spring-mass-damper continuous time system with forcing function (system input) F. This system can easily be analyzed using Laplace transforms.

Equation 2.7, with numerical values as shown gives the characteristic equation of motion for this system. The friction of the mass sliding on the surface is included in the

Eqn. 2.7 $$M\ddot{x}(t) + c\dot{x}(t) + kx(t) = F(t)$$
$$\text{where}, M=10\ kg,\ c=10\ N\,s/m,\ k=20\ N/m,\ F=1\ N$$

viscous damping coefficient *c,* and motion is constrained to the *x* direction. The Laplace transform for the equation of motion for the system is shown in Eqn. 2.8, from which the system transfer function (an output over input function) is obtained (Eqn. 2.9). From this transfer function model, the open loop characteristics can be found.

Eqn. 2.8 $$Ms^2X(s) + csX(s) + kX(s) = F(s) \quad \textit{(Laplace Transform)}$$

Eqn. 2.9 $$G(s) = \frac{X(s)}{F(s)} = \frac{1}{Ms^2 + cs + k} \quad \textit{(Transfer Function)}$$

Using the numerical values listed above and Matlab (a computational software package), the open loop system can be analyzed. In Matlab, the input is a unit step input, as if the forcing term was instantaneously placed on the system. The open loop response is shown in Figure 2.5. The response is inadequate since it has a very slow rise time, an extremely high steady state error (≈95%), and a high settling time, which are all directly influenced by the open loop system characteristics of the physical system. Thus, a PID controller is implemented in order to change the system response such that the correct displacement of the system will be achieved.

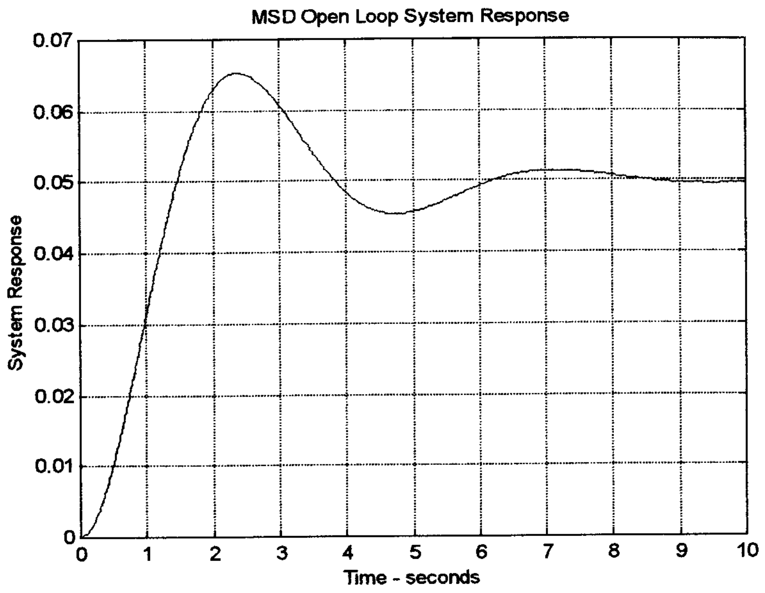


Figure 2.5: Open loop system response for Mass-Spring-Damper system. The response has a slow rise time and settling time, as well as a very large steady-state error. Ideally, for unit input (force), the output (displacement) should also be of unit magnitude.

The closed loop system (Figure 2.6) response is modeled in Simulink, a control system analysis tool in Matlab. The PID controller is designed by tuning the three gains of the controller as given by Eqn. 2.6. The final system response is shown in Figure 2.7, and shows a clear improvement in the response. Although the rise-time of the system remains relatively the same, and the system overshoots initially, the system settles much faster and has no steady state error. The PID controller design results in an adequate performance for the displacement of the mass-spring-damper system. This example shows the simplicity of PID controller design and implementation. [15]

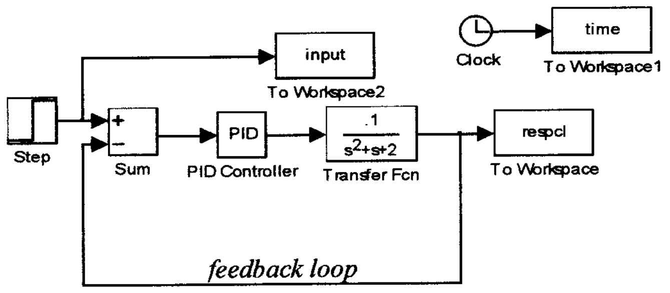


Figure 2.6: Simulink (Matlab) model of closed loop mass-spring-damper system with PID controller implemented.

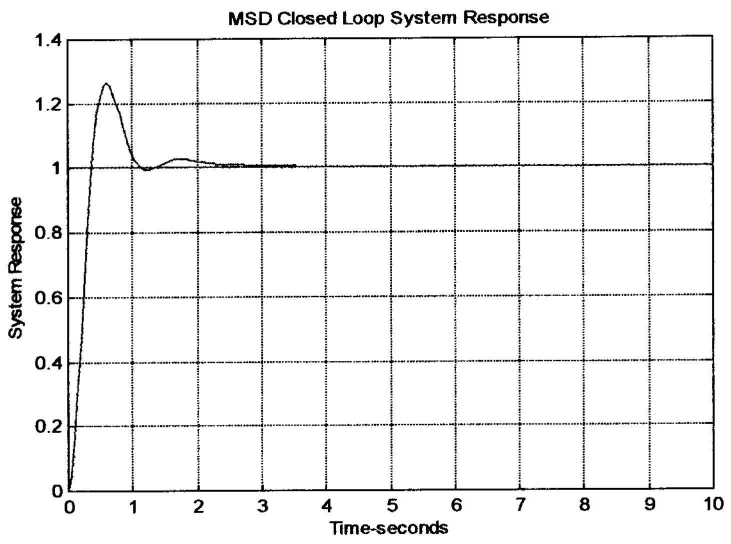


Figure 2.7: Closed loop system response with PID controller implemented. The system has a much better performance characteristics with no steady state error. Final gains are - $K_D = 50$, $K_P = 350$, & $K_I = 300$.

PID Controller Design

It is also important to discuss various techniques used commonly in the design of controllers. PID controllers essentially consist of the three gain components discussed previously, and it is the purpose of the design to find the numeric value of each of these gains such that the overall system response is optimized to some degree. In that sense, many PID controllers are simply a trial and error approach to the tuning of the three gains. However, there are design algorithms that are available to determine starting points for PID controllers. Two such methods that are of use in the design of PID controllers for this study are the Zeigler / Nichols method, and the root-locus method. Both of these approaches are discussed below.

Each of the three gains of the classical PID has an effect on the closed loop response of a system. For example, the integral control gain K_I can reduce or eliminate the steady-state error, but it may have an adverse effect on the transient response of the system. Table 2.1 provides a summary of the effects of each of the control parameters. It should be noted that changing one controller gain in order to achieve a certain effect will affect the other characteristics of the system response. Each control parameter, although acting independently of each other, has some interaction when it comes to the overall system response. Successful control has a give-and-take approach, which ultimately optimizes the closed loop system response. [15]

Table 2.1 – Summary of PID Parameter Effects on Closed Loop System Response

Closed Loop Response	Rise Time	Overshoot	Settling Time	Steady State Error
K_P	Faster	Increases	Small / No Effect	Decreases
K_I	Faster	Increases	Increases	Eliminates
K_D	Small / No Effect	Decreases	Decreases	Small / No Effect

There are two heuristic design algorithms, developed by Zeigler and Nichols (1942) which are traditionally used to find the PID control gains; the step-response method, and the ultimate sensitivity method. In the step-response method, the experimental open-loop step response is used to determine the PID gains. The first of these is the *steepest slope R* of the transient response (Figure 2.8). The intersection of the slope *R* with the positive time axis determines the second parameter, known as the *apparent deadtime L*. Using the relationship of *a=RL* to relate the two characteristics, it is possible to find the PID gains given in Table 2.2. This particular method is useful when the controller must handle large load disturbances. However, damping characteristics are not very desirable. Nevertheless, the ease of finding PID parameters from the open loop system response (if the open loop system response shows this behavior) warrants this approach for quick tuning.

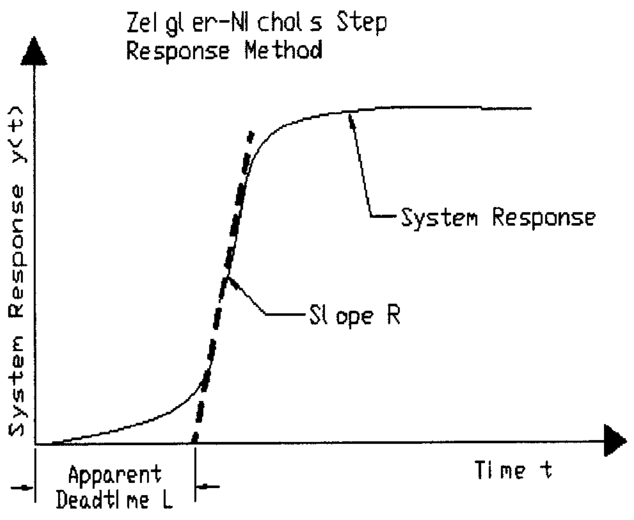


Figure 2.8: Zeigler-Nichols unit step response method parameters.

Table 2.2 – Controller Gains Using Zeigler-Nichols Unit Step Response Parameters

Controller Type	K_c	Ti	T_d
P	1/a	-	-
PI	0.9/a	3L	-
PID	1.2/a	2L	0.5L
NOTE:	$K_P = K_c / T_I$	$K_I = K_c\, T_d$	$K_D = K_c$

The second of the Zeigler-Nichols' methods, the ultimate sensitivity method, is a bit more complicated and involves setting the process in a closed loop feedback system with only pure proportional control as in Figure 2.9. Using a sinusoidal input *r(t)* into the system, the proportional control gain K_P is increased until the system reaches its stability threshold, and the system response shows repeated oscillations. At this point, the *ultimate gain K_U*, and the *ultimate period T_U* (Figure 2.10) are determined experimentally, and PID parameters are given by Table 2.3.

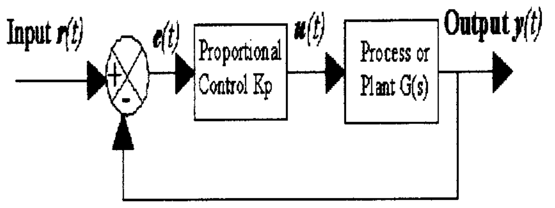


Figure 2.9: Zeigler-Nichols ultimate sensitivity method system. The system input is sinusoidal, and the process control is purely proportional. The gain is adjusted until the system response is similar to that of Figure 2.8, at which point the ultimate gain K_U and the ultimate period T_U can be determined.

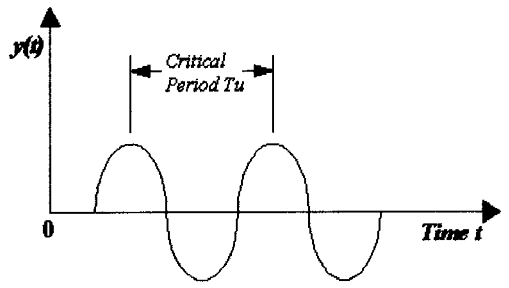


Figure 2.10: The critical period can be determined for the system when the system response reaches oscillatory motion.

Table 2.3 - Controller Gains Using Zeigler-Nichols Ultimate Sensitivity Method

Controller Type	K_c	T_i	T_d
P	0.5 K_U	-	-
PI	0.45 K_U	T_U / 1.2	-
PID	0.6 K_U	T_U / 2	T_U / 8
NOTE:	$K_P = K_c / T_I$	$K_I = K_c T_d$	$K_D = K_c$

Similar to the unit step method, this second method also provides a relatively easy algorithm to find the controller gains from the system response, which makes it a conceptually enticing method of PID design. However, there are several limitations to both methods that require mentioning. Damping characteristics for both methods are quite low, and Zeigler-Nichols does not account for all controller parameters [2]. Generally, processes with no integrator ($\frac{1}{s}$) terms provide a good basis for the application of either Zeigler-Nichols method, dependent on the open-loop system response. It is important to note is the fact that Zeigler-Nichols algorithms provide only a starting point for the PID controller, and usually much finer tuning of the PID gains is needed to achieve optimal control of the system response. [1],[12]

As mentioned above, process transfer functions that include an integrator term tend not to lend themselves well to PID design via Zeigler-Nichols methods. However, another method of design is the root locus method, which gives a powerful tool for analyzing stability and transient response for virtually all closed loop systems. Root locus is a graphical method that shows qualitative parameters from the system performance. The power of root locus lies in the fact that it is useful in analyzing higher order systems by varying certain parameters while others are held constant. For example, for a closed

[2] Although the PID controller discussed so far is the primary basis of this study, previous work done with analog controllers have introduced various other parameters into the PID equation. Åström provides a much deeper look into these parameters (see bibliography).

loop PID system, the system's response to the variation of the proportional gain can be easily determined using root locus techniques.

The example of the mass-spring-damper given in the previous section is revisited to illustrate the concept of using the root-locus method. Using the closed loop system shown in Figure 2.6, the PID equation of Eqn. 2.6 can be rewritten in the form of Eqn. 2.10, and the overall transfer function of the closed loop system can be found (Eqn. 2.11) using algebraic block reduction techniques. The overall closed loop system transfer

Eqn. 2.10
$$H(s)=\frac{K_{PID}(s+a)(s+b)}{s}$$

function is said to contain the closed loop *zeros*, or the roots of the numerator, and the closed loop *poles*, or the roots of the denominator. Even if the closed loop *zeros* (*a* and *b*)

Eqn. 2.11 [3]
$$CL(s)=\frac{H(s)G(s)}{1+H(s)G(s)}=\frac{0.1K_{PID}(s+a)(s+b)}{s(s^2+s+2)+0.1K_{PID}(s+a)(s+b)}$$

are known, the closed loop *poles* are directly influenced by the parameter K_{PID} (*note that* $K_{PID}=K_D$). Thus, it is the path that the closed loop *poles* take as the parameter K_{PID} is varied that is known as the root locus, and it is usually denoted in terms of a graph mapped in the s-plane (Laplace) of the variation of the closed loop *poles*. For the mass-spring-damper system, the root locus takes the form of Figure 2.11, which shows the path of the closed loop *poles* and *zeros* as the gain is varied. Thus, if the closed loop system is set with only one variable remaining, the corresponding behavior of the system can be found by varying that particular variable. For any given point along the root locus, the variable can be easily determined, and subsequently, all controller parameters can be found. For example, if the overall system response criteria included an initial overshoot of approximately 1%, corresponding to a damping coefficient of ξ=0.83, the controller

[3] For the purposes of continuity and simplicity, the process transfer function is defined as *G(s)*, while the controller transfer function is referred to as *H(s)*.

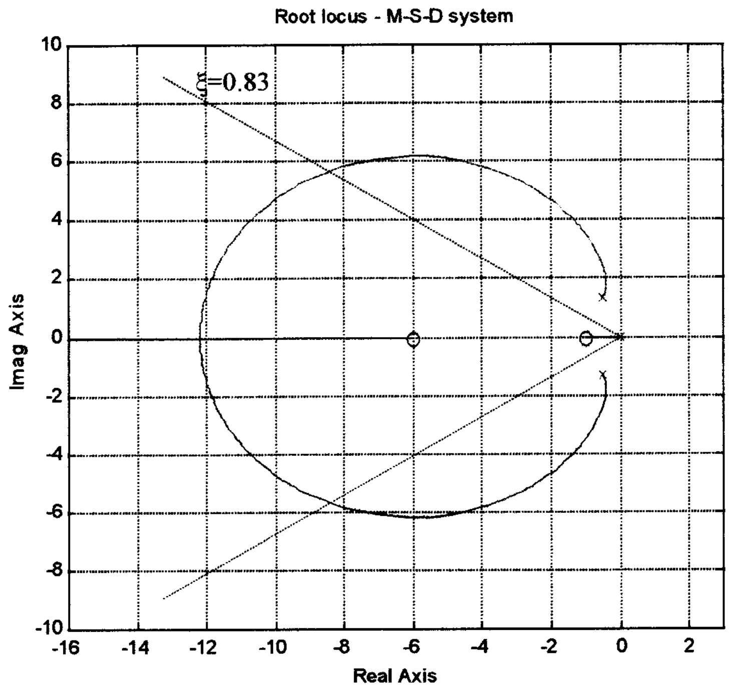


Figure 2.11: The closed loop root locus of the mass-spring-damper system shown before. The PID zeros are set to -6 and -1, and the graph shows the variation of the system poles as the control gain is varied. If a damping coefficient of 0.83 (1% OS) was required, the gain causing such transient response could be found by finding the intersection of the damping coefficient (diagonal) line and the root locus.

gain that causes such a transient response (given the controller *zeros*) can be found by searching for the intersection of the particular damping coefficient line and the root locus (Figure 2.11). [11]

It should be noted that this section serves as a severely limited overview of control systems design algorithms. Its intention is to introduce the idea that algorithms for design purposes, both analytical and graphical, do indeed exist for PID design, which is not solely dependent on mere trial-and-error tuning of the control gains. Both Zeigler-Nichols and root locus techniques are considerations when designing controllers for the rotary cutter.

Analog Versus Digital Control

A vast number of modern day controllers are digital (computer controlled) systems, and in its crudest form, digital control can be thought as approximations of analog systems. However, this does not inherently indicate the full prowess of computer control. The basis of digital control is the ability to implement a *sampled-data* system, where continuous signals can be handled properly with the use of a computer and analog-to-digital and digital-to-analog (A2D/D2A) converters. The greatest advantages of computer control in modern times are reduced cost, flexibility and robustness, and noise handling capabilities. For example, a computer controlled system can handle a vastly complex, multiple loop control system and can replace numerous analog controllers. Also, problems occurring in analog systems, such as noise and drift, can be compensated for in a computer-controlled system. Accuracy is much improved using digital systems, and switching in between loops can be handled inherently within the computer. A single computer and A2D/ D2A converter can replace expansive amounts of analog hardware, and any changes or adjustments to controller parameters can be handled via simple software changes, rather than involved hardware modifications. Thus, although much design of controllers (for this study) is done using analog techniques, final implementation is completely digital. [1],[10]

Analog Systems and Digital Equivalents

The previous sections have thus far outlined the concepts of analog controllers and the advantages of digital implementation. From a basic design standpoint, controllers can be designed using analog techniques, and implemented digitally using approximated equivalents. This section overviews three of the basic ideas found in computer controlled systems; discrete time (sampled) signals, the *z-transform*, and digital approximations.

In analog systems, signals that are used for processing the control commands are considered continuous signals, which theoretically is comprised of a continuum of points within the time limits that the signal has been assessed. However, although this may be true in terms of the theory of continuous signals, practical use of signals indicates that there exists a finite number of points that creates a signal. In terms of a digital system, a signal is created by taking samples of a continuous time signal over some amount of time at certain intervals. This collection of points results in a *sampled data signal*, which is a reconstruction of the continuous signal. The time in between samples is known as the sampling period / frequency *h*. Theoretically, the sampled signal should approach the continuous signal as the frequency of the sampling is increased during the length of sampling. In a digital system, the signal is sampled at a given time and at a given rate. The value of the signal is held constant using a *zero-order hold* circuit until the next successive sample, which results in the resemblance of the sampled data signal to that of a stair-like function. Figure 2.12 is a good example of the difference between a continuous and sampled signal. It shows the comparisons of the response of the mass-spring-damper of the previous example between a continuous and sampled signal.

The importance of sampled data lies in the fact that the computer itself is a discrete-time processor, and inherently cannot process a continuous signal. Thus, the continuous signal must be converted to a sampled data signal via an A2D converter, whereby it can be processed by the computer. The sampled data output of the computer (controller) is then converted back into a continuous time signal (*a process known as signal reconstruction)* by the D2A converter and sent back to the plant.

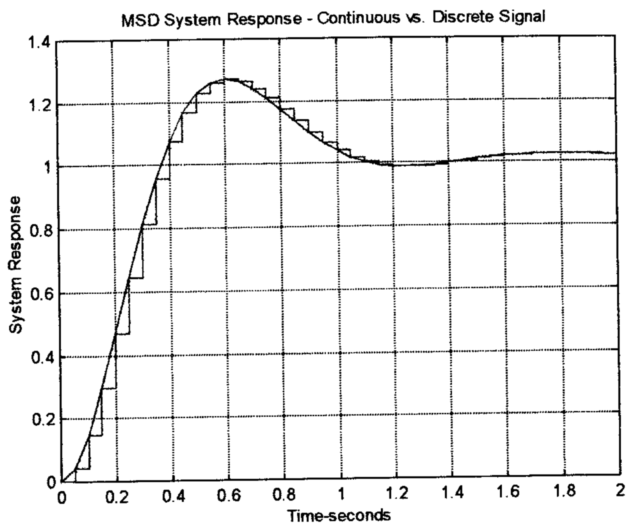


Figure 2.12: Continuous versus sampled data (discrete) signal. One of the fundamental concepts of digital control is the idea of replacing a continuous time signal with a sequence of sampled data in order to perform the control sequence within the computer. This particular signal is sampled every h=0.05 seconds, resulting in 20 samples per second. Faster sampling would result in a smoother curve.

An addition to the idea of sampled data (*also known as discrete)* signals is the concept of the difference equation. In analog systems, the process is defined by a mathematical equation (usually a differential equation), and is represented by a continuous function of time of the form *f(t)* given in state space representation in Eqn. 2.12. The discretized counterpart makes use of the sampled sequence of data to recreate the mathematical model in terms of a difference equation, where the output of the equations are a function of finite points of sampled data. Thus, the output of a particular sample time can be considered a function of sampled data taken during the previous sampling sequence (Eqn. 2.13). Note that the continuous-time system Parameters A and B (both matrix quantities depending on the order of the system) are discretized into parameters Φ and Γ. [1]

The second concept in discrete systems analysis is the idea of the *z-transform*, which can be thought of as the sampled data equivalent of the Laplace transform. Just as the Laplace transform allows for the analysis of continuous time systems, the z-transform permits the study of linear difference equations with or without initial conditions. The z-transform maps the semi-infinite time sequence into a function

Eqn. 2.12
$$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

(Continuous, State Space, Differential Equation Model)

Eqn. 2.13
$$x(kh+h) = \Phi x(kh) + \Gamma u(kh)$$
$$y(kh) = Cx(kh) + Du(kh)$$

(Sampled Data, Discretized, State Space, Difference Equation Mathematical Model)

of the complex variable (in the complex z-plane). Just as the Laplace transform takes a continuous time function *f(t)* and maps it into the complex *s* plane as a function of the form *F(s)*, the z-transform takes the difference sequence equation *f(kh)* and maps it into is shown in Eqn. 2.14, where *z* is a complex variable. The use of the z-transform allows the introduction of the *pulse-transfer* function *H(z)* (Eqn. 2.15). *H(z)* is obtained from the

Eqn. 2.14
$$\mathcal{Z}\{f(kh)\} = F(z) = \sum_{k=0}^{\infty} f(kh)z^{-k}$$

state-space representation in Eqn. 2.13, which is analogous to the continuous time transfer function *H(s)*. The *pulse-transfer* function is used to analyze discrete systems in a similar way that its analog counterpart *H(s)* is used to analyze continuous systems.

Eqn. 2.15
$$H(z) = C(zI - \Phi)^{-1}\Gamma + D = \frac{Z\{y(kh)\}}{Z\{u(kh)\}}$$

As stated previously, a discrete, digital system can be thought of as an approximation to a continuous-time, analog system. Thus, it is necessary to know how to approximate such systems. One method is to take the z-transform of a continuous time system, which is often tedious and cumbersome. In a continuous time system, a transfer function in the s-plane is a representation of a differential equation. Similarly, in a discrete system, a pulse-transfer function represents the difference equation in the z-plane. There are several approximation techniques which work quite well when transforming functions directly from the s-plane to the z-plane. The *pulse-transfer* function *H(z)* is obtained directly from the analog transfer function *G(s)* by replacing the s terms in the transfer function by each s' term given in Eqn.'s 2.16 - 2.18.

Eqn. 2.16 $$s' \overset{\rightarrow}{=} \frac{z-1}{h}$$ *(Forward Difference or Euler's Method)*

Eqn. 2.17 $$s' \overset{\rightarrow}{=} \frac{z-1}{zh}$$ *(Backward Difference Method)*

Eqn. 2.18 $$s' \overset{\rightarrow}{=} \frac{2(z-1)}{h(z+1)}$$ *(Tustin's Approximation or Bilinear Transformation Method)*

All three methods are easy to manipulate, even by hand calculations, in order to map the s-plane onto the z-plane. Tustin's approximation does have the distinct advantage of having the ability to map the entire left half of the s-plane, which comprises the stability region of analog systems, into the unit circle of the z-plane, which is the stability region of discrete systems (Figure 2.13). Thus, using a Tustin's approximation retains the stability characteristics of the closed loop system in the discretized domain. However, as will be shown, Tustin's approximation has a drawback that even though stability remains intact, not all analog poles of a system are mapped correctly, which will warrant further investigation into other approximation techniques. [1]

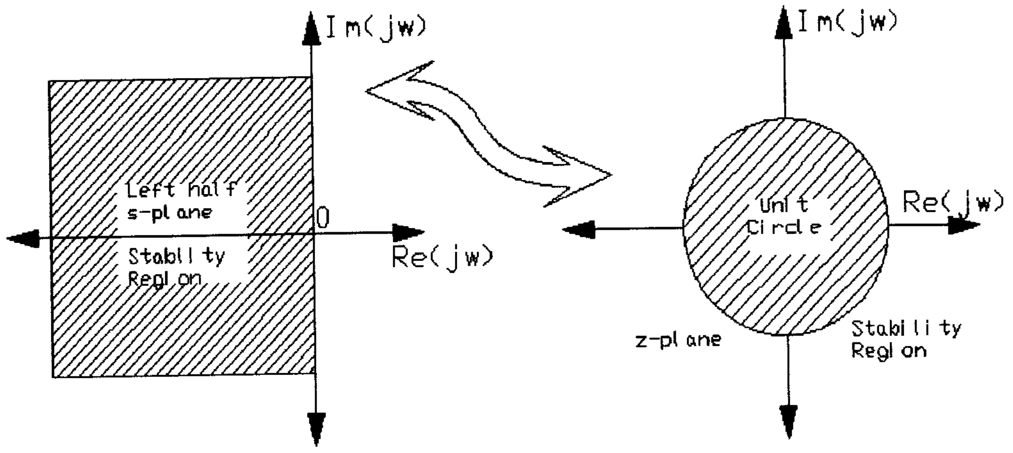


Figure 2.13: Tustin's approximation of the left hand s-plane. Tustin's approximation maps the analog stability region of the s-plane to the discrete stability region (the unit circle) of the z-plane. Thus, the advantage is that a Tustin's approximation of an analog system will not lose its stability characteristics during conversion.

CHAPTER III – MATHEMATICAL MODELING OF SERVO SYSTEM

Theoretical Modeling

The laboratory simulation of a rotary-knife cutter, as mentioned in the first chapter, involves the use of two independent servomechanisms. One of the most important aspects of this study of the control of the rotary-knife cutter is the development of a mathematical model that can be used for design purposes. This chapter deals with the theoretical and experimental modeling of the servomechanism which is designated as the 'master' system, or process plant. This particular section outlines a common method of modeling a motor system using purely theoretical considerations such as Kirchhoff's voltage laws and Newton's laws of motion to relate the internal electrical circuitry to the physical output of the motor. The second section discusses the mathematical model obtained using experimental techniques in the laboratory. Finally, the mathematical model that is implemented in Simulink, a Matlab based control systems toolbox, for design purposes, is shown.

A DC motor, is a common actuator found in many mechanical systems. The motor directly furnishes rotary motion, and when combined with other mechanisms such as drums and cables, it can provide translational motion. The fundamental basis for rotary motion in a servomechanism lies in the use of a current passing through a wire which is subjected to a fixed magnetic field. This wire feels a force due to the magnetic field, which can be used to turn a rotating member (a *rotor*). A continuous use of this effect over a length of wire coiled about a rotor (an *armature)* creates the fixed rotational motion that can be controlled by either increasing or decreasing the strength of the magnetic field or the armature current. Figure 3.1 shows the most common method of representing the electrical circuit that comprises a motor, while Figure 3.2 represents a free-body diagram of the output shaft of the motor. These two figures serve as the basis for the modeling of the motor, as Kirchhoff's voltage laws and Newton's laws of motion can be applied to relate each of the sub-systems. A common derivation used to model a motor servo-system follows.

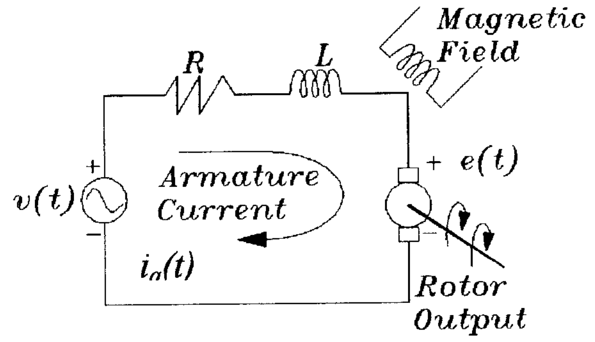


Figure 3.1: Electrical circuit representation of a motor. The armature current passing through the circuit creates a back emf of *e(t)*, which serves as the basis for the output torque and shaft rotation.

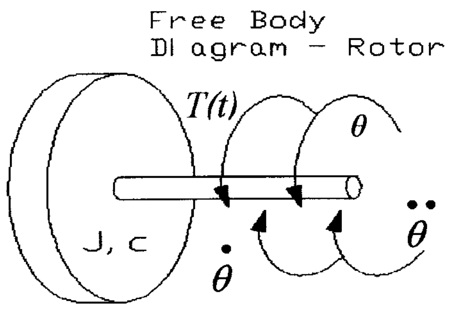


Figure 3.2: Free body diagram of motor shaft. The shaft has a moment of inertia *J*, a damping of *c*, and an input of torque *T(t)* coming from the rotor.

Equations 3.1 and 3.2 show the governing Kirchhoff's voltage law and Newton's law of motion for the two systems shown above. The basis for the circuit is that an *input voltage v(t)* creates an *armature current* $i_a(t)$ through the circuit, which runs through an inherent *armature resistance R,* and an *armature inductance L*. The *armature current* $i_a(t)$ in the presence of a magnetic field causes motion of the rotor output according to Eqn. 3.3. This motion, in turn, induces the *back emf (electromagnetic force) e(t)*, given by Eqn. 3.4. These two equations provide the basis whereby Eqn.'s 3.1 and 3.2 can be linked.

Eqn. 3.1 $$i_a(t)R + L\frac{di_a(t)}{dt} = v(t) - e(t)$$ *(Kirchhoff's Voltage Law)*

Eqn. 3.2 $$J\ddot{\theta}(t) + c\dot{\theta}(t) = T(t)$$ *(Newton's Law of Motion)*

In Eqn.'s 3.2 and 3.3, the terms K_t and K_e are known respectively as the *torque constant* and *motor constant.* In SI units, K_t and K_e are equal, and thus can be replaced by an equivalent constant *K*. Substituting both of these equations into 3.1 and 3.2 results in both

Eqn. 3.3 $$T(t) = K_t i_a(t) = K i_a(t)$$ *(Motor Torque)*

Eqn. 3.4 $$e(t) = K_e \dot{\theta}(t) = K\dot{\theta}(t)$$ *(Back Emf)* system equations being in terms of *armature current* $i_a(t)$, *input voltage* $v(t)$, *rotational speed* $\dot{\theta}(t)$, and *angular acceleration* $\ddot{\theta}(t)$, shown in Equations 3.5 and 3.6.

Eqn. 3.5 $$L\frac{di_a(t)}{dt} + Ri_a(t) = v(t) - K\dot{\theta}(t)$$

Eqn. 3.6 $$J\ddot{\theta}(t) + c\dot{\theta}(t) = Ki_a(t)$$

The subsequent Laplace transform of each equation leads to the forms in Equations 3.7 and 3.8, where the latter equation can be solved for the *armature current* term *I(s)*, which, when substituted back into 3.7, and after some rearrangement, yields Equation 3.9.

Eqn. 3.7 $$LsI(s) + RI(s) = V(s) - Ks\theta(s)$$ *(Laplace Transform)*

Eqn. 3.8 $$Js^2\theta(s) + cs\theta(s) = KI(s)$$ *(Laplace Transform)*

Eqn. 3.9 $$V(s) = Ks\theta(s) + \frac{(Ls+R)(Js+c)s\theta(s)}{K}$$

(Laplace transform notation in terms of input voltage V(s) and output rotational displacement θ(s))

At this point, it is important to differentiate what the modeling accomplishes within the terms that are presented in Eqn. 3.9. The term $s\theta(s)$ represents the *rotational speed* $\dot{\theta}(t)$, since the inverse Laplace transform of the term would be of the form $d\theta(t)/dt$, and $\theta(s)$ represents the *rotational displacement (position)* of the output shaft of the motor. Initially, the motor speed model can be found by simply collecting the like terms of Eqn.

3.9 of the $s\theta(s)$ form in order to find the governing second order transfer function form $G(s)$ of Equation 3.10.

Eqn. 3.10
$$G(s)_{speed} = \frac{\dot{\theta}(s)}{V(s)} = \frac{s\theta(s)}{V(s)} = \frac{K}{JLs^2 + (Lc + RJ)s + (Rc + K^2)}$$

(Governing transfer function for motor speed)

Subsequently, the rotational displacement of the motor shaft, or *position,* transfer function can be determined from the above equation by simply placing the s term (of the $s\theta(s)$ term) on the right hand side (Eqn. 3.11). This extra $1/s$ term is known as an *integrator.*

Eqn. 3.11
$$G(s)_{position} = \frac{\theta(s)}{V(s)} = \frac{K}{s(JLs^2 + (Lc + RJ)s + (Rc + K^2)}$$

In presenting the above derivation for the motor model, the considerations that are accounted for lead to second and third order systems for motor shaft speed and position, respectively. However, as a secondary consideration, many systems are modeled as lower order systems; namely, first order for motor speed and second order for shaft position. This is usually done when taking into account relatively low inductance values within the armature circuit. In essence, the response of the circuit is so fast that the delaying characteristics of the circuit transient response to an input voltage are quite negligible when compared to the similar response of the motor output. In essence, the motor output (speed) has a *time constant* associated with it, which causes a delay in reaching steady state operation. The motor circuit response is so much faster than the motor speed response, that the *time constant* associated with it can be ignored. Thus, the terms in the above two transfer functions can be represented as lower order systems by eliminating the terms containing the inductance L (Eqn.'s 3.12-13). Lastly, the forms for these

Eqn. 3.12
$$G(s)_{speed} = \frac{K}{(RJ)s + (Rc + K^2)}$$

Eqn. 3.13 $$G(s)_{position} = \frac{K}{s[(RJ)s + (Rc + K^2)]}$$

equations can be rearranged in order to express them in terms which can be found experimentally. These expressions take the form of Eqn. 3.14, which show the system model separated into terms for a *motor gain K_m*, a *motor time constant τ_m*, and also an *integrator gain K_p*; the first two which apply to the motor model, and the latter to the integrator model. The system can also be represented by the input-output of the motor and integrator, shown by the latter half of Eqn. 3.14. The following section outlines how the motor model parameters are determined experimentally for the servomechanism used in this study.

Eqn. 3.14 $$G(s) = \frac{K_m}{\tau_m s + 1} \cdot \frac{K_p}{s} = \frac{tacho_{OUT}}{pot_{IN}} \bullet \frac{pot_{OUT}}{tacho_{IN}}$$

Experimental Modeling

The experimental modeling of the motor servomechanism is based on the investigation of the closed loop system response for a given range of variable –frequency inputs. Figure 3.3 shows how the basic block diagram for the experimental procedure. The input to the system is a sine wave obtained via a signal generator where the input frequency can be varied. By tracking the various voltage signals shown in the figure of the model, and by analyzing the relationships of these voltages as the frequency of the input sine wave is varied through a given range, an experimental determination can be made for the motor gain and time constant, as well as the integrator gain.

The comparisons between the various voltage outputs in graphical form is commonly known as a *Lissajous,* or, *Input-Output* pattern. *Lissajous* patterns can take the form of Figure 3.4, where the various magnitude parameters can be determined and used for the experimental analysis to compare the *input-output* relationships for a given transfer function. The *Lissajous* pattern is represented on an imaginary $j\omega$ plane, such that the mathematical relationships shown in Eqn. 3.15 hold true in terms of the magnitude and phase of the output during the system response. The parameters of the pattern, shown in Figure 3.4 as *A, B,* and *C,* are used in these relationships. These parameters vary according to the frequency of the input signal, such that the ratios are usually used to establish a link between the input and output of the given transfer function.

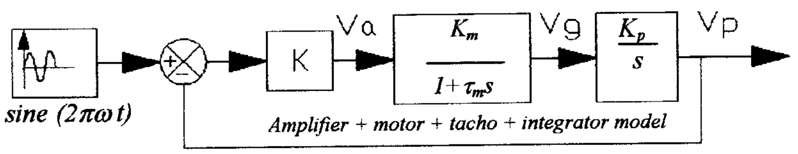


Figure 3.3: The transfer function model for experimental determination of the motor gain K_m, the motor time constant τ_m, and the integrator (position) gain K_p. The system is experimentally set up such that a signal generator feeds a variable frequency sine wave, and that there is unity feedback at all times. The input output voltages V_a, V_g, and V_p are used to determine the system parameters.

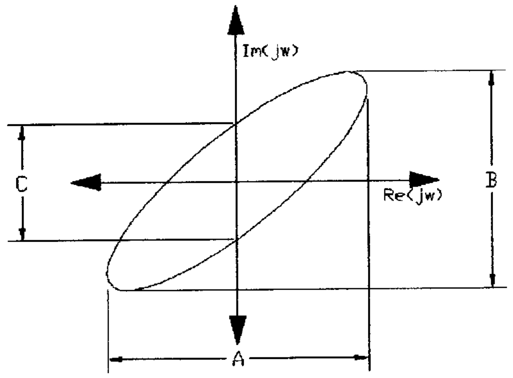


Figure 3.4: A typical Lissajous pattern for an input-output model system. The various magnitudes A, B, and C are used to determine the model parameters mentioned above. The magnitudes vary for each input frequency that is measured, such that the ratios of the magnitudes are used for analysis.

Eqn. 3.15 $$|G(j\omega)| = {}^{B}\!/_{A} \qquad \textit{(magnitude)}$$

$$\langle G(j\omega) = -\sin^{-1}({}^{C}\!/_{B}) \qquad \textit{(angle)}$$

Furthermore, Eqn. 3.16 also holds true in a *Lissajous* pattern with the voltage outputs *Va* on the horizontal axis and *Vg* on the vertical axis, such that the plot of *-tan[(<G(jω)] vs. Frequency f (Hz)* theoretically yields a straight linear line through the origin with a slope of $2\pi\tau_m$. This relationship can be shown by taking the form of 3.16 and placing it into the latter equation of 3.15. Some manipulation yields a form for the

Eqn. 3.16 $\langle G(j\omega) = -tan^{-1}(\omega\tau_m)$, where $\omega=2\pi f$

given relationship akin to a simple linear line of the form $y=mx$, where the y portion is the $-tan[(<G(j\omega)]$ and the x is the *frequency f (Hz)*, which ultimately leaves a slope of $2\pi\tau_m$. Therefore, using the magnitude values found by the experimental *Lissajous* patterns and the above relations, it is possible to graphically find the *motor time constant* τ_m for the tachogenerator transfer function.

Another relationship that holds true for the *Lissajous* pattern found for the tachogenerator transfer function by plotting *Va* vs. *Vg* is shown in Eqn. 3.17, which relates the magnitude of the pattern to the *motor time constant* and the *tachogenerator gain* K_m. In rearranged form (Eqn. 3.18), the tachogenerator gain can be found for a given frequency from the resulting *Lissajous* pattern and the first half of Eqn. 3.15. Thus, with

Eqn. 3.17 $$|G(j\omega| = \frac{K_m}{\sqrt{1+(\omega\tau_m)^2}}$$

Eqn. 3.18 $$K_m = |G(j\omega)| \bullet \sqrt{1+(\omega\tau_m)^2}$$

the use of these relationships that hold true between the input voltage *Va* of the tachogenerator transfer function and the output voltage *Vg* of the same function, it is possible to determine *tacho gain* K_m, and the *motor time constant* τ_m.

In the case of the output potentiometer (position) transfer function in the last block of the motor model, the *integrator gain* K_p is found by looking at the relationship between the input voltage *Vg* of the transfer function as well as the output voltage *Vp* for a given range of frequency inputs. The relationship that holds true for this transfer function is shown in Eqn. 3.19, which can be directly found from a *Lissajous* pattern with the x-axis as *Vg* and the y-axis as *Vp.* A noticeable feature of this particular relationship is that there is a phase angle of 90° associated with it such that C=B (see Fig. 3.4).

Eqn. 3.19 $$\frac{V_g}{V_p} = \frac{B}{A} = \frac{K_p}{s} = \frac{K_p}{2\pi f}$$

The experimental model is quite simple to set up, as a function generator is used to simulate a sine wave of variable frequency as the input to the model. The frequency of the sine input is varied from 0.1 Hz to 1 Hz in increments of 0.1 Hz, and the complete data for the various voltage signals are stored as data files. The *Lissajous* patterns are plotted using Matlab, and the parameters *A, B,* and *C* are graphically determined. The final results are tabulated in Tables 3.1 and 3.2, and the average of the ten patterns for the values of the two gains is determined. A linear regression analysis using Microsoft Excel is used to determine the slope of the experimental data in order to find the motor time constant (Figure 3.5).

Although the analysis of the experimental data is cursory, and though graphical techniques are not generally considered extremely accurate, the model parameters that are determined are adequate for the purposes of this study. The essential need for a model is so that controllers can be designed using the actual numbers found in this experiment. The resulting capability of the controllers to hold a high level of accuracy for the overall system response will ultimately determine the general degree of precision for the knife cutter. Fundamentally, if the controllers can hold a precision where the error between the master and follower is zero, then the integrity of the original model is not of such great importance.

Table 3.1 – Experimental Data from *Lissajous* Patterns Comparing *Va* vs. *Vg* for Determination of Motor Model Transfer Function Parameters.

Input Frequency (Hz)	A	B	C	$\lvert G(jw)\rvert=(B/A)$	$<G(jw)=-\sin^{-1}(C/B)$	$-\tan[<G(jw)]$	K_m
0.1	0.202	1.178	0.099	5.839	-0.084	0.084	5.9124
0.2	0.367	2.100	0.540	5.718	-0.260	0.266	6.0014
0.3	0.601	3.378	1.631	5.626	-0.504	0.551	6.2347
0.4	0.868	4.880	2.886	5.620	-0.633	0.733	6.6635
0.5	1.210	6.100	4.106	5.040	-0.738	0.910	6.4426
0.6	1.501	6.874	4.833	4.579	-0.780	0.989	6.3332
0.7	2.171	9.056	6.757	4.171	-0.842	1.121	6.2465
0.8	2.979	11.543	9.056	3.874	-0.902	1.265	6.2744
0.9	3.753	13.466	10.886	3.588	-0.941	1.373	6.2705
1	4.573	15.249	12.763	3.335	-0.992	1.529	6.2703
						Average K_m =	**6.2649**

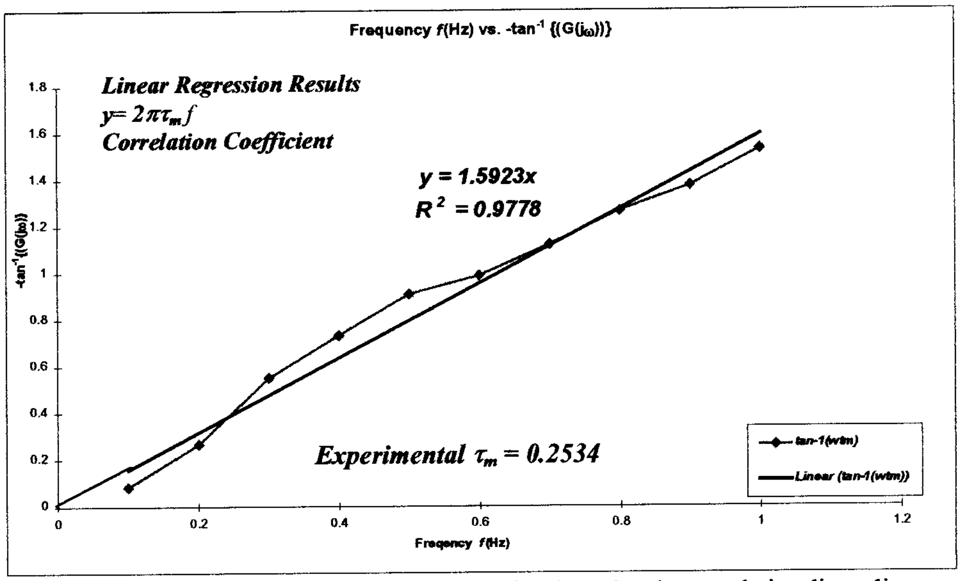


Figure 3.5: A plot of the frequency of the input signal vs. the phase angle is a linear line through the origin with a slope of $2\pi\tau_m$, which is found by using a regression analysis tool to fit a linear line through the experimental data points. The resulting motor time constant is found to be **0.2534.**

Thus, from the analysis of the experimental data found by tracking the input-output voltages of the tachogenerator transfer function, the motor time constant τ_m is found to be 0.2542, while the motor gain K_m is found to be 6.2649.

Table 3.2 – Experimental Data from *Lissajous* Patterns Comparing *Vg* vs. *Vp* for Determination of Integrator Gain K_P.

Input Frequency (Hz)	A	B	K_P
0.1	1.183	11.701	6.2172
0.2	2.134	10.428	6.1410
0.3	3.372	11.331	6.3347
0.4	4.739	12.106	6.4201
0.5	6.116	12.434	6.3875
0.6	6.910	11.824	6.4509
0.7	9.145	13.513	6.4988
0.8	11.501	14.938	6.5288
0.9	13.450	15.889	6.6800
1	15.261	16.522	6.8024
		Average K_P =	**6.4461**

From these experimentally derived parameters, it is possible to set up a mathematical model for the motor using transfer functions (Figure 3.6). This mathematical model can also be implemented in Simulink, along with various controllers, for design and system response analyses. Figure 3.7 shows such a Simulink model, with both the motor model transfer functions implemented along with a positional PID controller shown in the forward loop. With this type of a Simulink model, it is possible to design, implement, and test various types of controllers for actual use in the rotary knife cutter.

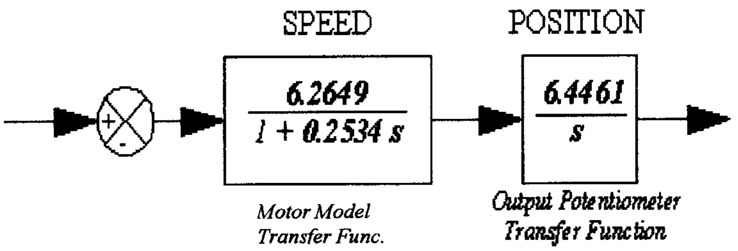

SPEED
POSITION
+
-
$\frac{6.2649}{1 + 0.2534 s}$
$\frac{6.4461}{s}$
Motor Model Transfer Func.
Output Potentiometer Transfer Function

Figure 3.6: Open loop model with experimental parameters placed in the transfer functions. The tachogenerator transfer function can be rearranged in the denominator to yield a form shown in Fig. 3.7.

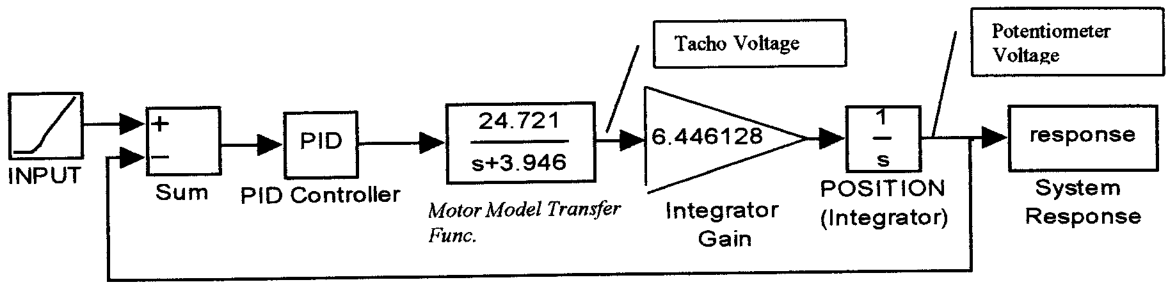


Figure 3.7: Simulink model of the motor model with experimental parameters placed in. The system is shown with unity feedback, a ramp input, and a positional PID controller in the forward loop. It is this type of model that is used in the design process for position control.

CHAPTER IV – CONTROLLER DESIGN, MODELING, AND COMPUTER CONTROL CODE

Controller Design and Implementation Approach

Chapter 1 briefly discussed the physical nature of the laboratory simulation of the rotary knife simulated by two side-by-side servo systems where the speed and position of one (referred to as the follower system) is directly dependent on the speed and position of the other (the master system). Figure 1.9 of Chapter 1 shows schematics of the servo-systems, while Fig. 1.10 shows an overall outline of the particular type of control sequence that will take place for this particular simulation. Essentially, for every one revolution of the master output, where there are two targeted cutting points, the follower output will have to match each of these cutting points in position during the cutting phase, while speeding up in between the cutting phases in the correction zone. Thus, for one revolution of the master output, there will be two cutting phases in which position control is utilized, and two correction zones, where speed control is utilized. Final control of the rotary cutter will be highly dependent on the performance, both transient and steady-state, of these two controllers, as well as the integrity of the switching logic in the implemented computer control code.

This chapter will focus on the design approaches used for the speed and position controllers as they are modeled and tested in the Matlab and Simulink environment. While Matlab is used to perform and analyze design algorithms, the Simulink models simulate the implemented controllers on the experimental servo-system model found in Chapter 3. The Simulink models serve a very important purpose because they return theoretical responses of the plant (the servo-system) and controller interaction for a wide variety of controller designs and initial condition stages. Several problems that arise with the controller designs when trying to obtain a discrete approximation to the analog controllers are also discussed in this chapter. Finally, the computer code logic that is used to implement the controllers is discussed.

Position Controller Design

The first controller design is the position controller because of two reasons. First and foremost, the position controller is the primary control phase within the cutting zone, and second, the control is performed on the second order plant model. It is an assumption within the controller design stage that if the higher order, more complex, plant can be controlled first, the second controller (the speed controller) will be an easier design because of the lack of an integrator in the plant model. There are several design algorithms that can be applied to the design of the position controller. Two of these approaches, Zeigler-Nichols, and root-locus, have been previously discussed. For the design of the controller for this particular plant, a root-locus method is used. Both of the Zeigler-Nichols methods have the drawback that they require some effort in obtaining open loop system characteristics for the initial considerations, and yet they do not pinpoint PID parameters with great accuracy. Much greater tuning is eventually needed when using Zeigler-Nichols methods. Root locus offers the advantage that the system model (controller + motor) can be simulated in Matlab and Simulink, and various test cases for the controller can be analyzed. Therefore, the root-locus method is chosen for the position controller design.

Figure 4.1 shows the basic model of the SISO (Single-Input Single-Output) system that is used to develop the root-locus equations needed for the design of the position controller. Equations 4.1 and 4.2 show the two transfer function models for the controller and the plant. The latter half of Eqn. 4.1 is the PID controller equation rearranged in the form of a gain K_{PID}, and the numerator zeros. In algebraic terms,

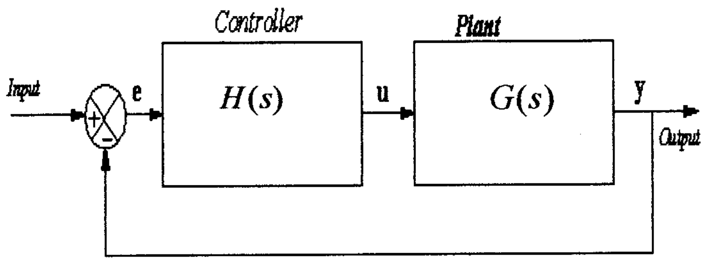


Figure 4.1: SISO (Single-Input Single-Output) model of the system. This model serves as the basis for the development of the root-locus design for the controller.

it turns out that the controller gain K_{PID} is the same as the derivative gain K_D, while the ratios of the other PID gains with the derivative gain form the controller zeros *a* and *b*.

Eqn.4.1 $$H(s) = \frac{K_D s^2 + K_P s + K_I}{s} = \frac{K_D(s^2 + \frac{K_P}{K_D}s + \frac{K_I}{K_D})}{s} = \frac{K_{PID}(s+a)(s+b)}{s}$$

Eqn. 4.2 $$G(s) = \frac{159.3567}{s^2 + 3.945981s}$$ *,Servo system plant transfer function*

Subsequently, if the controller zeros can be defined, the root-locus of the closed loop system will give an indication of the behavior of the system poles and zeros as the controller gain K_D is varied. From a given point on the root locus, the PID gains can be evaluated using the loop closure equation in Eqn. 4.3. In looking at the quantity *G(s)H(s)* in 4.3 from a design standpoint, it is advantageous to set one of the controller zeros equal

Eqn. 4.3 $$1 + K_{PID}G(s)H(s) = 0 \Rightarrow K_{PID} = {}^{-1}\!/_{G(s)H(s)}$$

to the plant root 3.945981 to produce a zero-pole cancellation. Also, the other controller zero can be placed further in the left hand plane such that the root-locus of the resulting closed loop system is larger. With this approach in mind, a Matlab program *controller1.m* (see Appendices for code) is setup to evaluate the PID gain parameters. Three values of the second controller zero are evaluated along with three points along the resulting root-locus of the system. The three evaluation points are approximated at the second controller zero, at the negative real limit of the root-locus, and approximately half way in between. Figure 4.2 shows one example of a resulting root-locus, while Table 4.1 summarizes the PID gains found from the various root-locus responses. This approach gives a starting point for the PID controller parameters that can be further modeled and refined in Simulink.

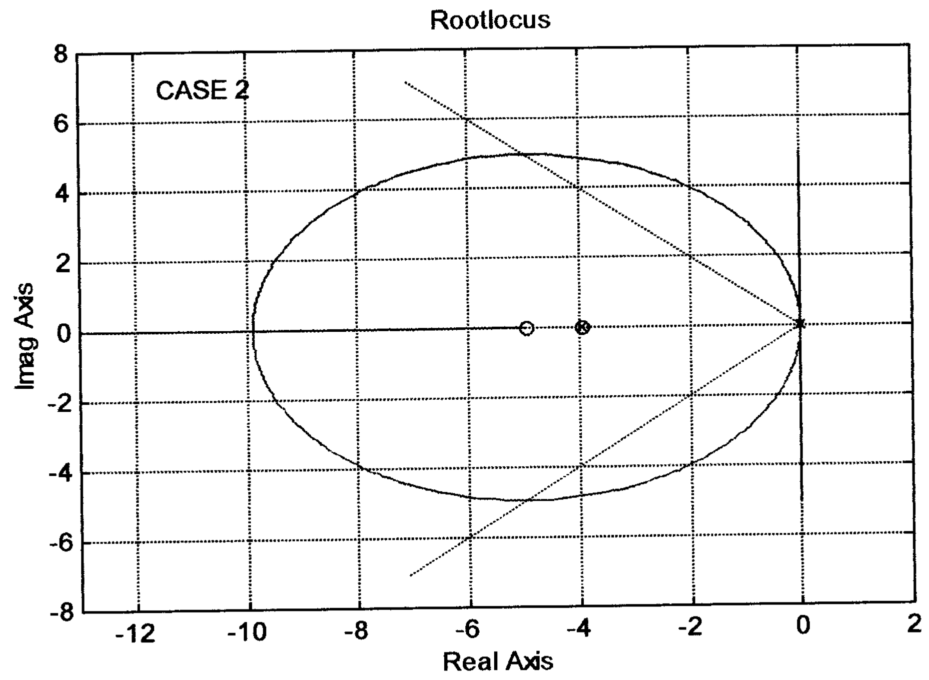


Figure 4.2: Root-locus plot of Case 2 of Table 4.1. One controller zero cancels the plant pole while the other is placed further in the stability region of the Laplace domain. The two remaining system poles branch off and meet back at a point along the negative real axis. Choosing a point along the axis allows us to evaluate the system equations to obtain the gain that causes this type of behavior.

Table 4.1- Summary of PID Gains Found from Root-Locus Cases

Evaluation Cases	Controller Zeros		Root-Locus Evaluation Point	K_D	K_P	K_I
Case 1a	3.946	4.446	4.4194	0.0138	0.116	0.2425
Case 1b	3.946	4.446	6.1935	0.0226	0.1899	0.3969
Case 1c	3.946	4.446	8.871	0.0371	0.3112	0.6506
Case 2a	3.946	4.946	5.0507	0.016	0.1424	0.3125
Case 2b	3.946	4.946	7.2281	0.0269	0.2395	0.5256
Case 2c	3.946	4.946	9.8548	0.0412	0.3661	0.8036
Case 3a	3.946	8.946	8.9816	0.0282	0.364	0.9968
Case 3b	3.946	8.946	12.8157	0.0474	0.6106	1.6719

The subsequent step in the design of the position controller is to use the above PID parameters to check the system response of the plant (the servo system). This is done using a Simulink model consisting of a PID controller and plant. Analyzing various system response parameters such as the servo- and controller-outputs, as well as the error signal by further tuning the PID gains optimizes the design. Figure 4.3 shows the

Simulink model that is used to analyze and optimize the PID controller. There are several points of consideration for this model that need to be addressed. First is the input signal

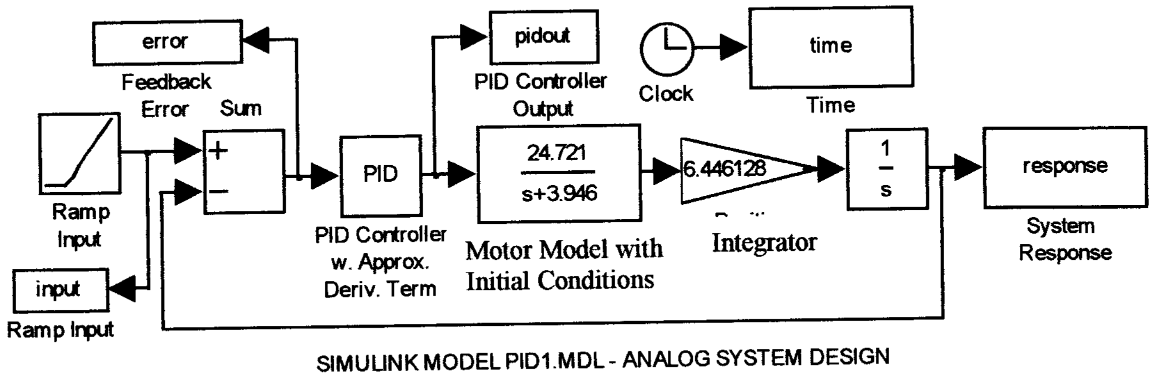


Figure 4.3 – Simulink model PID1.MDL used to test and optimize the position controller for the servo-system plant shown above. The input to the system is a linear ramp which is modeled after the master servo-system's output potentiometer signal. The PID controller in this model is implemented with an extra approximate derivative term.

to the system. The input signal is what the follower servo system will have to follow in terms of position. Figure 4.4 shows the concept behind the input signal for the Simulink model. The follower, during the position control phase, will have to track the same signal as the output potentiometer of the master servo system. So it is this signal that becomes the input to the servo system that is being controlled. The output potentiometer is comprised of a rotary variable resistance potentiometer which, as it is turning clockwise, goes from a –15Volt output to a +15V output, with a physical break in the signal in between the two extremes. Therefore, the signal actually looks very similar to that shown in Figure 4.4, such that the input to the Simulink model can be modeled as a ramp input starting at –15. The slope and finishing point of the ramp signal is based on how long it takes for one rotation of the potentiometer. In the case of the Simulink model, the ramp input has a 4-second duration ending at a +15 level, corresponding to a signal that can actually be recreated on the experimental setup. All the designing and refining of the PID controller for position control are done using this ±15-Volt signal over a 4-second period. However, the final signal that the follower servo system tracks is only half of the ±15 Volt signal, due to the fact that the Comdyna interface can only handle ±10 Volt range. The master potentiometer output is halved using an operational amplifier circuit with a gain of 0.5.

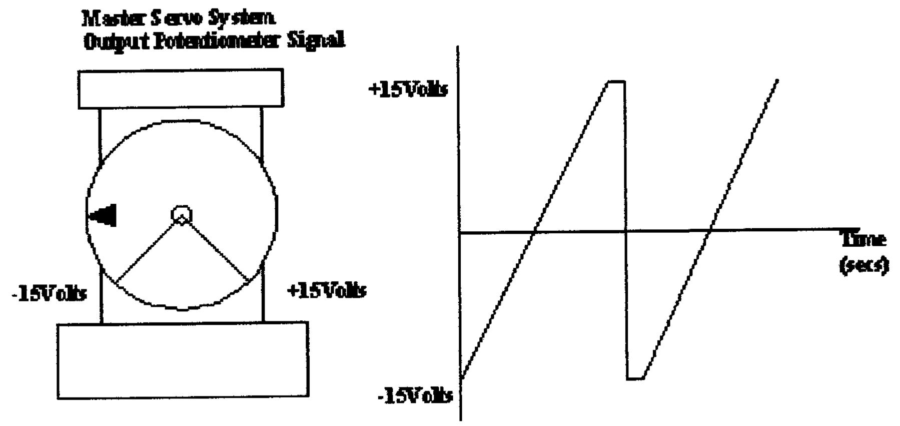


Figure 4.4 – The signal generated by the output potentiometer of the master servo-system takes the form of a linear ramp from –15 Volts to +15 Volts for a clockwise rotation. There is a physical break in the potentiometer that causes the non-uniformity in the signal as the signal jumps from its positive extreme to its negative extreme. One

Another feature of the PID position control model is the introduction of an extra approximate derivative term N in the PID control equation. This is done because a pure derivative term $(K_D\ s)$ found in the control equation can give rise to large amounts of amplification of measurement noise when implemented. Eqn. 4.4 shows the form of the PID control equation when an approximate derivative term is implemented. The

Eqn. 4.4
$$H(s) = \frac{K_D s}{(\frac{1}{N}s+1)} + K_P + \frac{K_I}{s} = \frac{(K_P + K_D N)s^2 + (K_I + K_P N)s + (K_I N)}{s(s+N)}$$

advantage of using the extra derivative term N is that the derivative term $(K_D\ s)$ is approximated quite well at low frequencies but is limited to N at high frequencies associated with noise. Åström states that N is typically in the range of 3 to 20. For the purpose of developing the position controller, N is set to a value of 17 for the Simulink model, which turns out to be an adequate setting for the system performance.

The last consideration to make in the Simulink modeling is the initial condition of the plant. Assuming that the follower position will be in relative close proximity to the master position when starting a phase, the initial condition of the integrator term in the plant model can be set close to the –15 starting point of the ramp input. However, it is not necessarily known if the speed of follower will match that of the master. Hence, the slopes of the two signals, or the speeds of the servos, may be quite different. Thus, the Simulink model incorporates this idea and allows for the initial condition of the tachogenerator plant to be varied. The slope of the input ramp signal is 7.5, and the Simulink model is varied by placing initial conditions of 0, 5, 7.5, and 10 on the motor-plant. The first initial condition simulates a start-from-rest condition, the second simulates a slower initial speed, the third simulates a similar initial speed, while the last simulates a faster initial speed. The final controller must be able to handle any of these initial conditions for the motor.

The Simulink model is run using the PID parameters shown in Table 4.1 as the starting point for the gains. The controller is optimized by tuning the gains for the various initial conditions of the motor. Figures 4.5 – 4.7 show the system responses for the final final controller that is chosen. The main criteria in choosing a controller is that of a fast settling time, no steady-state error, and the least over- and under-shoot in the motor response. Upon closer inspection of the various responses, it can be seen that all the responses for the model show adequate performance based on the previous criteria.

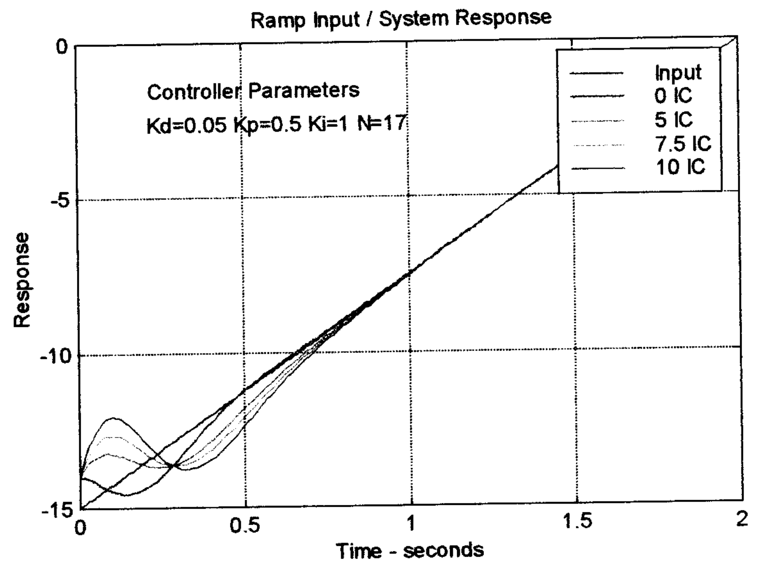


Figure 4.5 – The Simulink model plant response for a ramp input for various initial conditions. As shown by the graph, the response is adequate irrespective of speed (initial conditions). Comparatively, there is a fast settling time, and no steady state error.

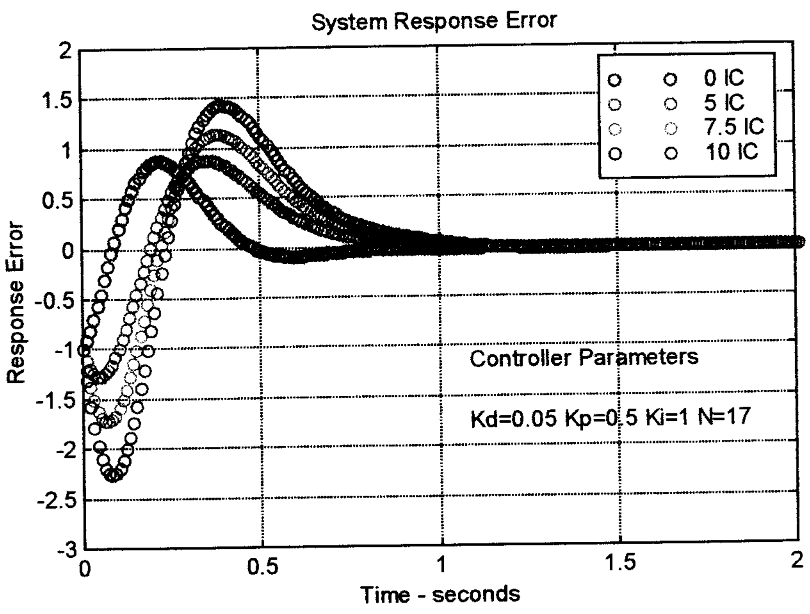


Figure 4.6 – The system error over time. The error due to variances in speed does not cause a great change in the overall system response, thereby providing a more robust ability for position control.

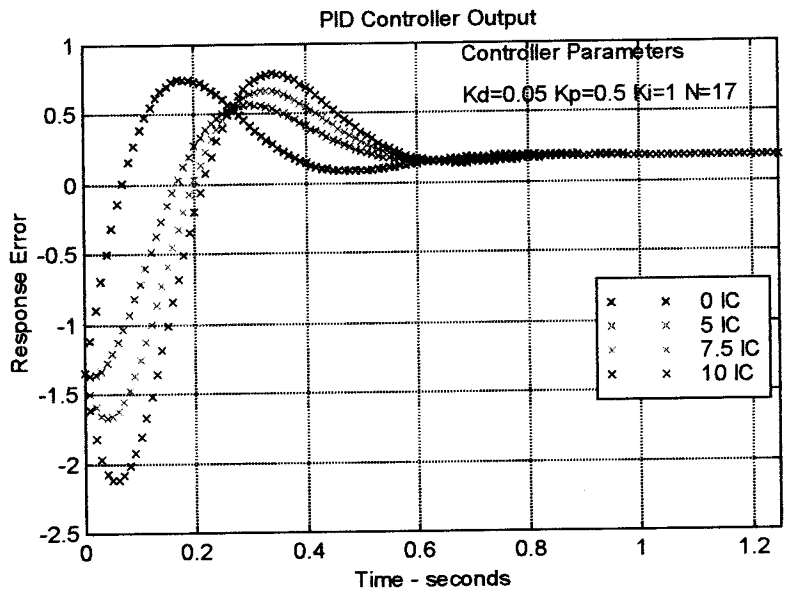


Figure 4.7- The analog signal of the controller output over time. After the initial crossover of the 0 axis, there is no further oscillation of the control signal, and the final signal is not very large in magnitude.

The final PID controller is shown in Eqn. 4.5 with controller gains $K_D = 0.05$, $K_P = 0.5$, $K_I = 1$, and N=17. This controller is adequate from an analog standpoint since the gains are not very large and could be set with relative ease in an analog controller.

Eqn. 4.5 $$H(s) = \frac{0.05s}{(\frac{1}{17}s+1)} + 0.5 + \frac{1}{s} = \frac{1.35s^2 + 9.5s + 17}{s^2 + 17s}$$

Speed Controller Design

The speed controller design is a bit simpler than the position controller design since it deals with a first-order plant model. A full PID controller is not needed for the speed control phase. If the controller is to match the same order of the plant, then the derivative term of the PID controller, which adds the second order term, can be omitted to produce a PI controller. Eqn. 4.6 shows the basic equation of the PI controller in various forms, the last of which is applied to the speed control problem. The speed

Eqn. 4.6 $$H(s)_{soeed} = K_P + \frac{K_I}{s} = \frac{K_P(s + \frac{K_I}{K_P})}{s} = \frac{K_P(s+a)}{s}$$

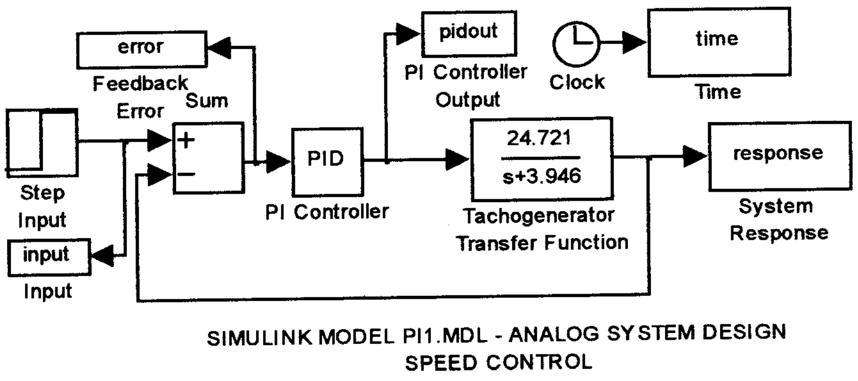


Figure 4.8 – The speed controller Simulink model. A PI controller is implemented along with the tachogenerator plant model to test and refine the speed controller. The input to the model is a step input which is modeled after the sampled signal of the tachogenerator.

controller has a pole at 0, and a zero that can be placed arbitrarily along the negative left-hand plane of the Laplace domain. If the zero is placed at the same location of that of the plant pole (s = -3.945981), then the only parameter to ascertain is the proportional gain K_P, since the integral gain is given by $K_I = K_P\, a$. Thus, the speed controller comes down to finding a suitable proportional gain. The proportional gain is varied from 0.1 to 0.9 in order to produce a small integral gain, and subsequently tested in Simulink. The Simulink model (Figure 4.8) of the speed controller is used to refine the speed controller based upon the resulting set of gains. The input to the model is an instantaneous step input, which is modeled after actual signal from the tachogenerator. Although noisy, the signal is a constant value for the duration of the sampling. Figures 4.9 and 4.10 show the various responses for the final PI controller (Eqn. 4.7) that is chosen from the tuning. From these responses, it can be seen that the PI speed controller results are adequate.

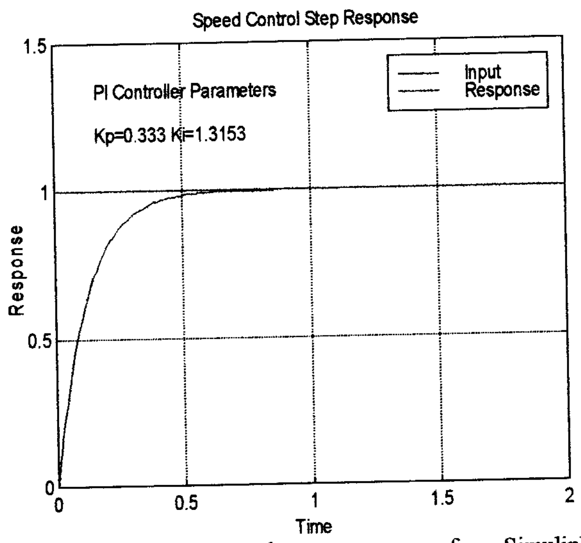


Figure 4.9- Speed control system response from Simulink model. The initial condition for the plant is set to 0 for a start-from-rest scenario. The overall response is very fast with no overshoot and steady-state error.

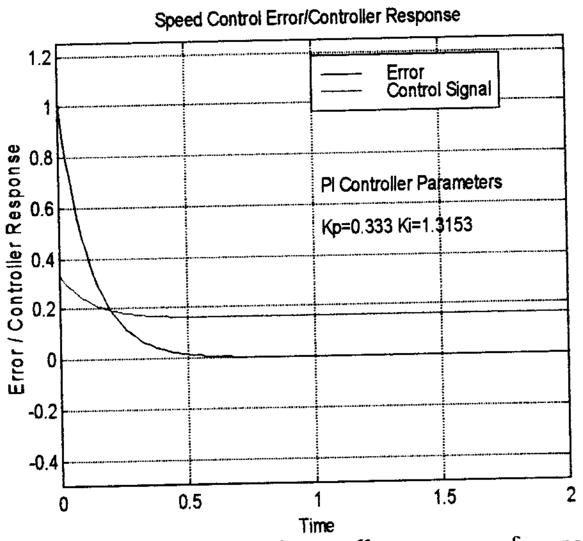


Figure 4.10- The error and controller responses for speed control. It is seen that a very small signal is required to control the plant.

The step-input response for the plant has very good transient and steady-state characteristics, while the controller signal is not large.

Eqn. 4.7 $$H(s)_{SPEED} = \frac{0.3333s + 1.3153}{s}$$

Digital Approximations and Selection of Sampling Interval

Two of the most important aspects of implementing the final computer-controlled system are the discrete approximation of an available analog controller and the selection of the sampling frequency of the system. This section presents some of the considerations made for the digital approximation of the position controller as well as for the selection of the sampling period h.

A direct approach to digital implementation of the controller is to develop an approximate discrete model whose performance is adequate based on a sampling interval that can be handled by the computer. The position controller provides a good model for discussion about the digital approximation techniques that can be used to obtain a discrete model. The analog position controller is discretized using an approximation technique and checked for adequate performance using Simulink. Equation 4.5 shows the final, second-order, analog PID controller that has two zeros and two poles. Discussions in Chapter 2 outlined several available discrete approximation techniques including *Tustin's, first-order-hold, zero-order-hold,* and a *backward-difference* method for approximating an analog transfer function. Matlab contains several of these approximations as well as some others that can be used for approximating the position controller. Two such methods are the *matched-pole-zero* method, and the *first-order hold (foh)* method available in Matlab. Of these available methods, *Tustin's, matched-pole-zero,* and *foh* are all used to approximate the position controller. The difference between the three techniques lies in the approximated poles of the controller. Table 4.2 shows that Tustin's method does not approximate the controller poles correctly, whereas the foh does. Table 4.2 shows by using a direct z-transform of the two analog poles (e^{sh}) and comparing the results to the Tustin's and foh approximations, that the pole at –17 is not approximated correctly by Tustin's method. The foh technique is chosen for approximating the digital controllers for both the position and speed analog controllers. The resulting digital approximations are modeled again in Simulink to check for errors in the responses, and are discussed below.

Table 4.2 – Comparison of PID Controller Pole Approximations.

			Approximation Techniques					
Sampling Interval *h*	Analog Poles *s*		e^{sh}		*Tustin's*		*First-Order-Hold*	
100 ms	0	-17	1	0.1827	1	0.0811	1	0.1827
50 ms	0	-17	1	0.4274	1	0.4035	1	0.4274
25 ms	0	-17	1	0.6538	1	0.6495	1	0.6538

Another important consideration is the sampling frequency *h* of the discrete model. Intuitively, it is known that the faster a signal is sampled, the better it is constructed. However, this is not always possible. The ideal sampling period is one that is small enough to have the ability to correctly construct a sampled signal from a continuous one, but not so small that it does not give the hardware enough time to process information. Also, a slow sampling period will result in a digital controller that does not perform adequately even though it may been approximated correctly. An appropriate sampling period for the position controller is found by modeling the resulting controller for all three of the sampling intervals shown in Table 4.2 for a foh. The motor plant and the input signal are discretized using a *zero-order-hold* found in an A2D conversion. Figure 4.11 shows the resulting Simulink model using a state-variable representation.

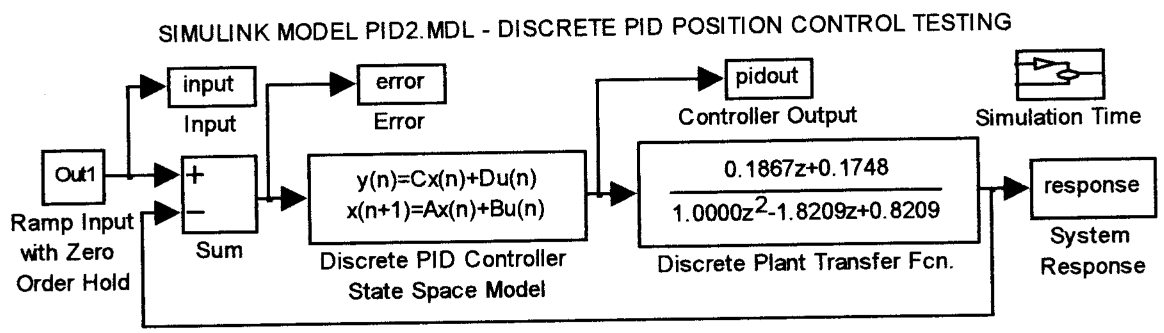

SIMULINK MODEL PID2.MDL - DISCRETE PID POSITION CONTROL TESTING
input
Input
error
Error
pidout
Controller Output
Simulation Time
Out1
Ramp Input with Zero Order Hold
+
−
Sum
y(n)=Cx(n)+Du(n)
x(n+1)=Ax(n)+Bu(n)
Discrete PID Controller State Space Model
0.1867z+0.1748
1.0000z2-1.8209z+0.8209
Discrete Plant Transfer Fcn.
response
System Response

Figure 4.11- Discrete position control Simulink model. The ramp input and plant models are discretized using a zero-order-hold, while the controller is a predictive-first-order-hold equivalence. System performance is tested using various sampling frequencies. A sampling frequency of 25 ms yields an adequate response.

Figures 4.12- 4.14 show the effects of varying the sampling frequency. Although in each case the system is approximated correctly, *h* has a strong effect on overall system performance. From the various Simulink responses, it is found that a 25ms sampling

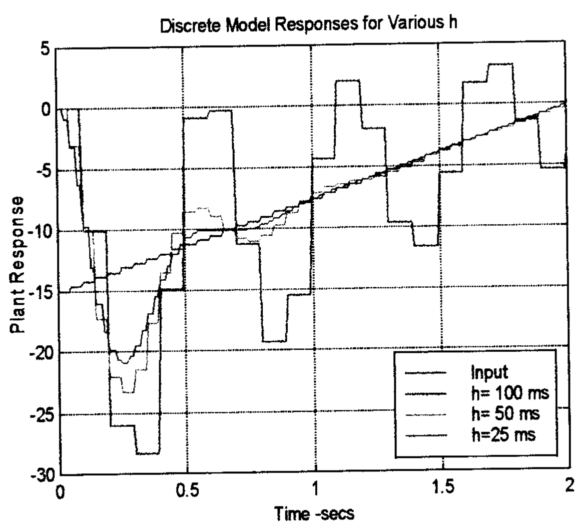


Figure 4.12- Simulink discrete model plant responses for various sampling frequencies. The faster the sampling, the better the system response.

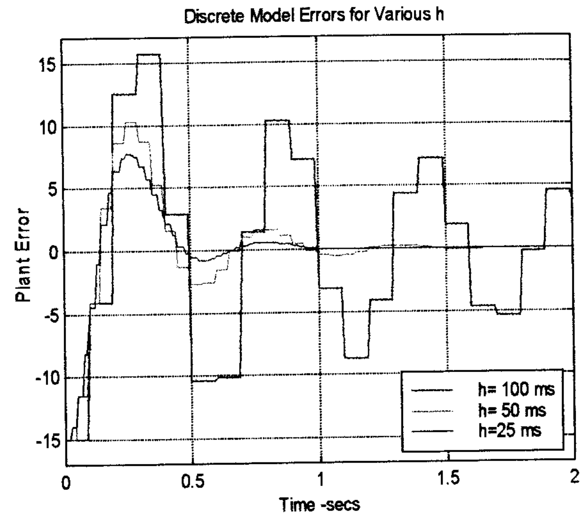


Figure 4.13- The plant response error at various sampling frequencies.

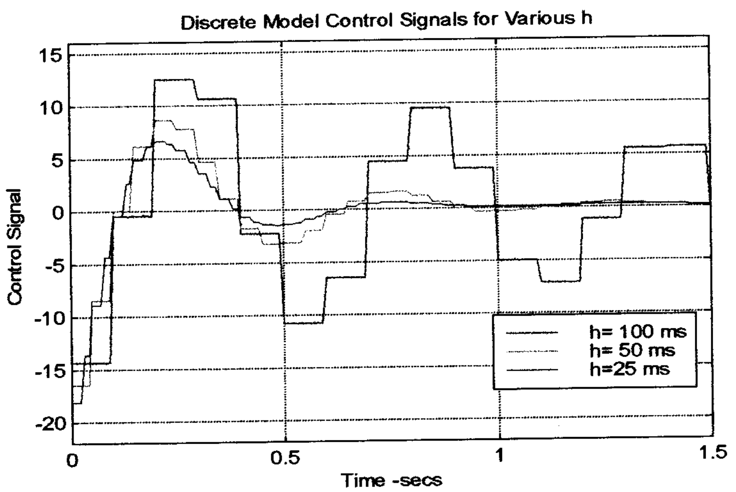


Figure 4.14 – Control signal for discrete Simulink model for various sampling periods. The large initial controller output in all three models is caused by the large error caused by the initial condition placed on the plant. The main consideration is that the control signal not be highly oscillatory in its response since that may lead to instability in the implemented controller.

results in adequate system performance. This sampling period seems warranted since the time constant of the motor is approximately 0.25 seconds, such that a sampling frequency of 25ms would mean taking 10 samples per time constant. This would allow a better construction of the transient response of the motor. Using a sampling period of 25ms, the resulting discrete controller pulse transfer functions are given by Eqn.'s 4.8 and 4.9, for position and speed, respectively.

$$\text{Eqn. 4.8} \quad H(s)_{POSITION} = \frac{1.35s^2 + 9.5s + 17}{s^2 + 17s} \Rightarrow H(z) = \frac{1.2050z^2 - 2.2075z + 1.0112}{z^2 - 1.6538z + 0.6538}$$

(discrete PID position controller using h=25ms and predictive foh)

$$\text{Eqn. 4.9} \quad H(s)_{SPEED} = \frac{0.333s + 1.3153}{s} \Rightarrow H(z) = \frac{0.3498z - 0.3169}{z - 1}$$

Computer Code for Controllers

The next step in the implementation of the rotary knife cutter controller is transferring the existing digital controllers from their present pulse-transfer function equations to computer code. This is done using the state-space representation of the controllers. Figure 4.15 shows the controller input-output representation. Using the state-space representation of the controllers, it is possible take the error as the input to the

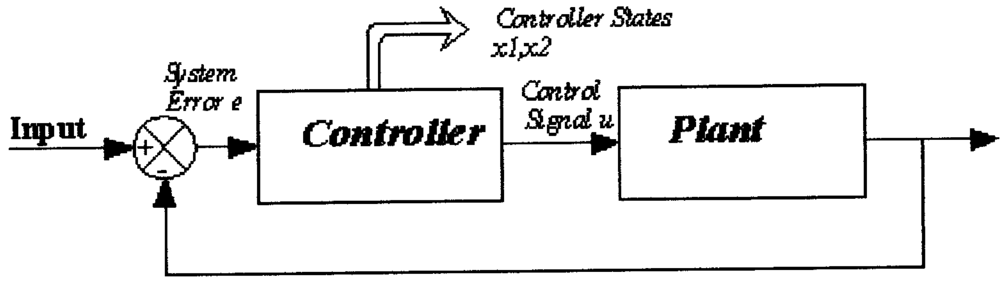


Figure 4.15- The state-space representation of the controller utilizes the controller states and the resulting control equations to take the error input to the block and output an appropriate control signal u.

controller and output the appropriate control signal by programming the state-variable equation in the computer code. The state-variable representation for the two controllers are found in Matlab using a state-space-to-transfer-function (ss2tf) conversion. For the second order position controller, the computer code is generated mainly from the state-variable representation of the form in Eqn. 4.10. The corresponding state equations, which are difference equations, are shown in Eqn.'s 4.11 – 4.13, and serve as the basis for the computer code.

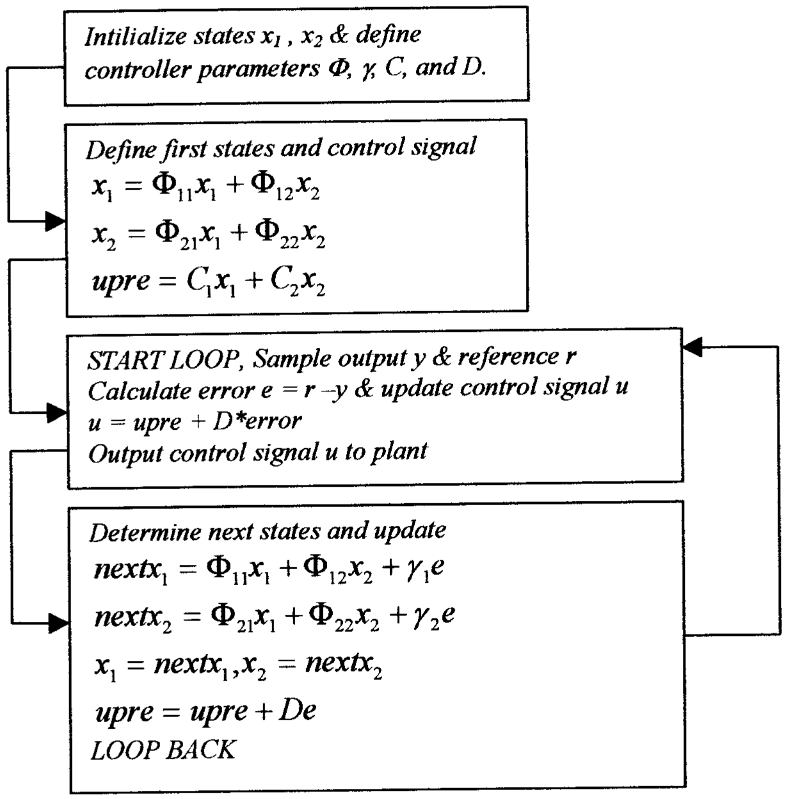
Eqn. 4.10
$$x(k+1) = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} x(k) + \begin{bmatrix} \gamma_{11} \\ \gamma_{21} \end{bmatrix} e(k)$$
$$u(k+1) = \begin{bmatrix} C_1 & C_2 \end{bmatrix} u(k) + [D] e(k)$$

Eqn. 4.11
$$x_1(k+1) = \Phi_{11} x_1(k) + \Phi_{12} x_2(k) + \gamma_{11} e(k)$$

Eqn. 4.12
$$x_2(k+1) = \Phi_{21} x_1(k) + \Phi_{22} x_2(k) + \gamma_{21} e(k)$$

Eqn. 4.13
$$u(k+1) = C_1 x_1(k) + C_2 x_2(k) + De(k)$$

The above three equations serve to compute the next states as well as the control signal, and thereby need to be implemented into the computer code. The following listing is a generalization of the computer code that is programmed for the control sequence. The sequence is simply a loop within which the reference and output signals are sampled, the states of the controller are updated, and subsequently, the control signal is computed and output to the plant. The computer code takes the form of the following –


Intilialize states x_1, x_2 & define controller parameters Φ, γ, C, and D.
Define first states and control signal
$x_1 = \Phi_{11}x_1 + \Phi_{12}x_2$
$x_2 = \Phi_{21}x_1 + \Phi_{22}x_2$
$upre = C_1x_1 + C_2x_2$
START LOOP, Sample output y & reference r
Calculate error $e = r - y$ & update control signal u
$u = upre + D*error$
Output control signal u to plant
Determine next states and update
$nextx_1 = \Phi_{11}x_1 + \Phi_{12}x_2 + \gamma_1 e$
$nextx_2 = \Phi_{21}x_1 + \Phi_{22}x_2 + \gamma_2 e$
$x_1 = nextx_1, x_2 = nextx_2$
$upre = upre + De$
LOOP BACK

This flow chart for a second order controller serves as the model for the implementation of the computer control code. The code for the speed controller, which is a first order model, follows the same logic, but with lesser calculations. In the following chapter, the actual implementation of the code is discussed, as well as the logic that is used to implement the switching between position and speed control.

CHAPTER V – DIGITAL IMPLEMENTATION OF CONTROLLER, RESULTS & CONCLUSIONS, AND RECOMMENDATIONS FOR FUTURE WORK

Initial Digital Testing of Individual Controllers

The next step in the overall process is the direct computer implementation of the two controllers. This implementation proceeds in a step-by-step fashion, starting with the implementation of the two controllers separately in order to check the individual controllers. The experimental setup remains the same with two servo-systems side by side with the both tachogenerator (speed) signals and the two output potentiometer (position) signals being sampled. Using an existing program with the analog-to-digital / digital-to-analog functions pre-programmed as a model, the two controllers are programmed using 'C' language. The speed controller is tested under a condition that the follower output potentiometer must be able to speed up and rotate faster than the master output potentiometer during the correction zone. Thus, the tacho master signal is multiplied by a constant to achieve this effect. For example, if the master tacho signal is multiplied by a factor of 2 and defined as the new reference signal, then the result is that

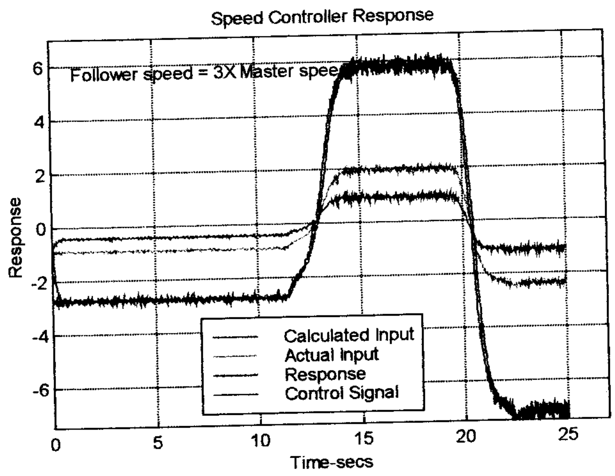


Figure 5.1 – Speed controller response. The sampled tacho signal is multiplied by a factor of 3 to achieve a faster rotation speed for the follower. Changes in direction of rotation and speed are handled well by the first order controller.

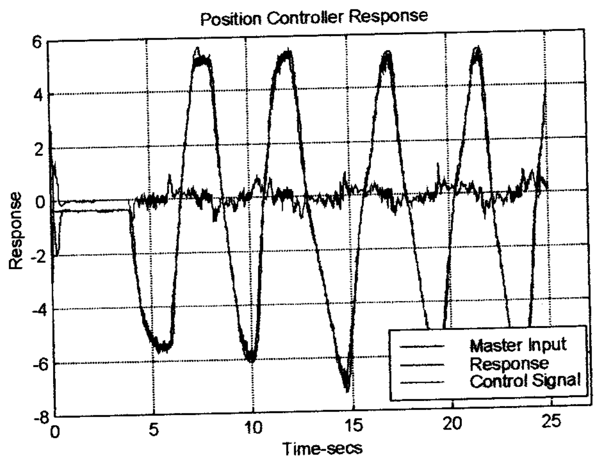


Figure 5.2 – Position control response. The input to the controller is a signal from a manually rotated input potentiometer.

the follower rotates at twice the speed of the master and in the same direction. A negative speed multiplier changes the direction of rotation for the follower, and can also be set in the speed controller program. Figure 5.1 shows the speed control response when a speed multiplier of 3 is set. The response shows that the controller performs very well irrespective of speed and direction of rotation. The control signal, in magenta, is seemingly not very large in magnitude, even during transient response stages when either the speed or direction of rotation is changed.

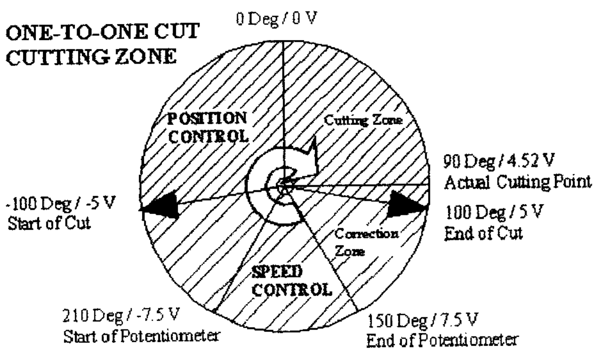
Similarly, the position controller is also tested using the servo systems, although the input to the system is achieved using a manually rotated input potentiometer, which circumvents the need to condition the program to handle the physical break in between the two leads of the master potentiometer. The manual input potentiometer lets the signal remain between values that the test program can handle, but returns an indication of the position controller performance. Figure 5.2 shows that the position controller response is also accurate. One noticeable aspect of both the plots is the presence of noise in the signals, which later on turns out to be one of the contributing factors to overall system performance. Thus, from this initial testing of the two controllers with a sampling period of $h = 25ms$, it is determined that the controllers have adequate stand-alone performance capabilities.

Implementation of a One-to-One Controller

The following step implements a one-to-one controller, which achieves one cut per revolution of the master potentiometer. This is a necessary step so that the switching logic in between the two different types of controllers can be checked. If the switching logic is satisfactory for repeated cuts, then it can serve as a springboard to the development of the two-cut controller.

Previous to setting up the one-to-one controller, it is essential to look at some of the physical characteristics of the output potentiometers such that some pre-set levels for switching can be determined. From experimental data collected in the laboratory, it is determined that for one 360° revolution of the potentiometer, approximately 17% of the rotation lies within the physical break of the positive and negative leads. Also, during implementation of the digital system, it is seen that the Comdyna A2D / D2A interface can only handle a range of ± 10V, whereas the output from the potentiometers is of a range of ± 15V. It is therefore necessary to cut the potentiometer voltage down such that the Comdyna can be used properly. This is done by using an op-amp circuit with a gain of 0.5 which cuts the ± 15V voltage by half to a range of ± 7.5 V volts. Instead of the master position signal linearly increasing from –15V to +15 V during one rotation, it now increases linearly from –7.5V to +7.5V for the same time period. Along with the new limits of the sampled signal and the 17% dead zone, one half revolution of the potentiometer is from –7.5 V to 0V, accounting for approximately 160°'s of rotation. This corresponds to a conversion rate from degrees to volts of approximately 20° / Volt, or 0.05 Volts / ° of Revolution. This conversion rate helps to determine what limits to set the switching logic at in the program.

Figure 5.3 and 5.4 show the control sequences for a one-to-one cut. The limits of the



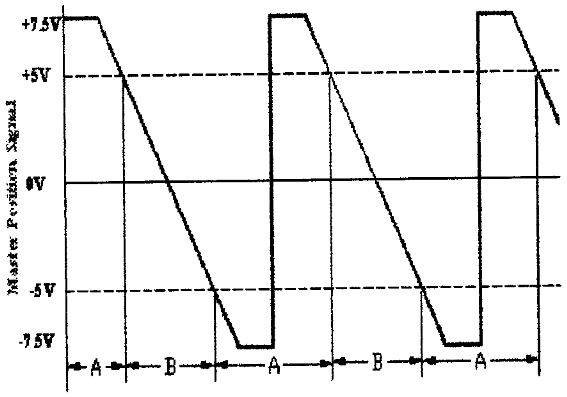


Figure 5.3 – Definition the cutting zone and correction zone on the master position potentiometer output. The computer code must allow for the switching to take place dependent on these preset values.

Figure 5.4 – Speed and position control phases as shown on the master position signal. The signal is inverted first such that the follower rotates in the opposite direction. Phase A defines the speed control zones while Phase B defines the position control zone.

switching is set to ± 5V, corresponding to a rotation of approximately 200° with the cut level at approximately +4.5V. A flow chart for the one-to-one controller is shown in

Figure 5.5. The program takes into account an initialization mini-loop (*at the START LOOP block)* which allows for the follower to be moved to a pre-determined position very quickly without having to track any of the master signal even though it is in motion. Test 1 checks the error for the follower position tracking, and detects when the master position signal is approaching the same level as where the follower position is, whereby it breaks off of the initialization mini-loop and moves on to the main cutting loop. It is here that the follower starts to follow the master position signal, which is done by calling a position control function outside of the main loop. The position control

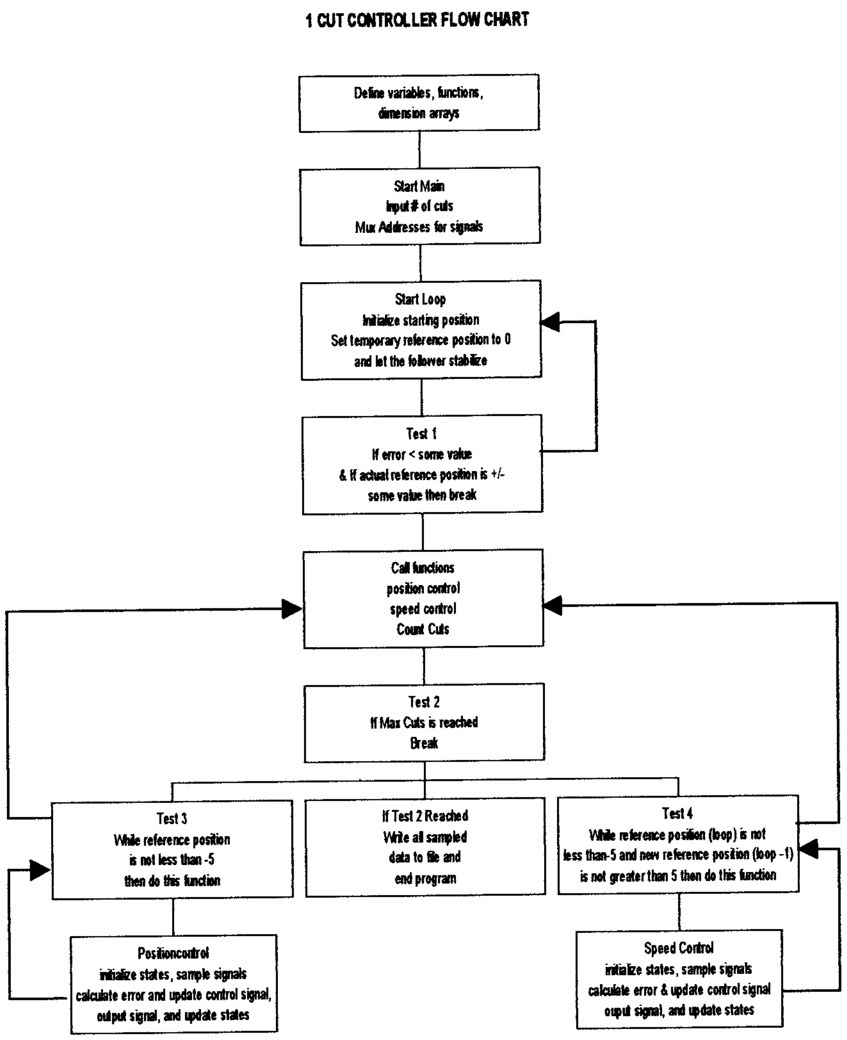

1 CUT CONTROLLER FLOW CHART
Define variables, functions, dimension arrays
Start Main
Input # of cuts
Mux Addresses for signals
Start Loop
Initialize starting position
Set temporary reference position to 0 and let the follower stabilize
Test 1
If error < some value
& If actual reference position is +/- some value then break
Call functions
position control
speed control
Count Cuts
Test 2
If Max Cuts is reached
Break
Test 3
While reference position is not less than -5 then do this function
If Test 2 Reached
Write all sampled data to file and end program
Test 4
While reference position (loop) is not less than-5 and new reference position (loop -1) is not greater than 5 then do this function
Positioncontrol
initialize states, sample signals
calculate error and update control signal, output signal, and update states
Speed Control
initialize states, sample signals
calculate error & update control signal
ouput signal, and update states

Figure 5.5 – Flow chart for the one-to-one controller programmed as K121.c that can be found in the appendices. The switching is based on the master position signal, which determines if the controller is in position control or speed control.

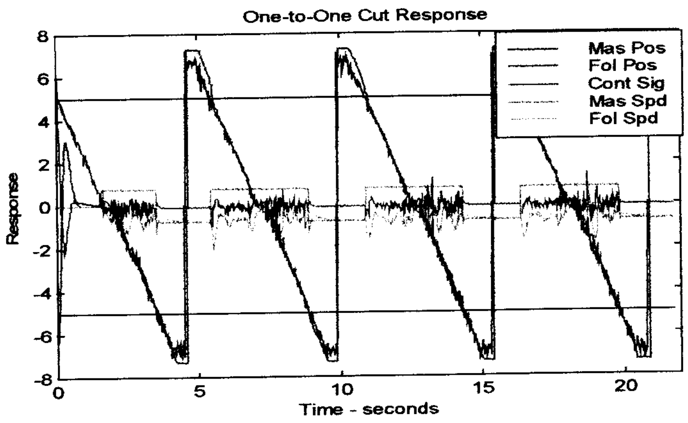


Figure 5.6- One-to-one cut system response. All 5 sampled signals are plotted together to illustrate relative magnitudes as well as noise. The black horizontal lines at +/- 5 show the switching limits. The controller is in position control in between the two limits.

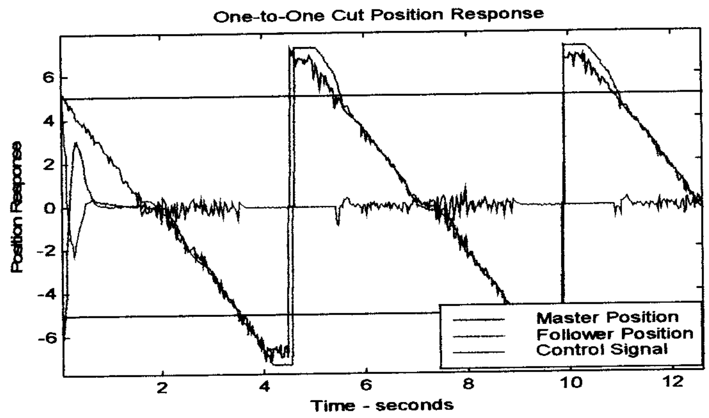


Figure 5.7- One-to-One controller position response. During the position control phase, the controller tracks relatively well even though the input signal is very noisy. The initialization stage is illustrated at the start of the sequence.

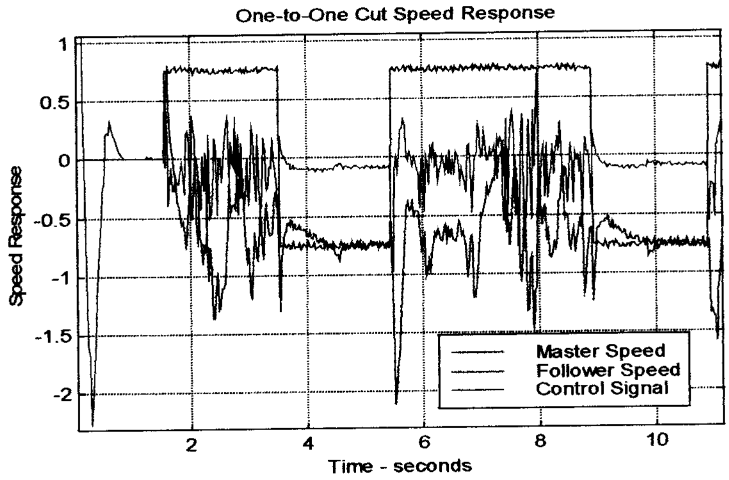


Figure 5.8 – One-to-One controller speed response. During the position control phase, the master speed signal is positive, but is inverted during speed control so the follower rotates in the opposite direction than that of the master. Notice the relative smoothness of the control signal during speed control vs. position.

function remains in effect till the *while* statement in Test 3 is not true, whereby the program returns to the main loop and moves on to the speed control function. This process is repeated until the number of cuts is achieved and the main loop can be broken. Finally, the data that is collected in the loop is written to a file which can be manipulated at a later time. Figures 5.6 –5.8 show a typical response of the one-to-one controller that has the limits of ± 5V programmed into the switching logic. From the plots, it can be seen that the position controller tracks the master signal while it is in between the two limits (Fig. 5.9). The initialization mini-loop at the beginning of the program allows that the control sequence always starts off in position control. Speed control utilizes a speed multiplier of 1x, which simply means that the follower goes through the potentiometer break at the same speed that the master does. Figure 5.8 shows the speed response of the system. During the position control portion of the control sequence, the program samples the actual master speed signal, which is positive indicating a clockwise rotation. However, during speed control, the signal is inverted such that the follower speed rotational direction is opposite to that of the master. Somewhat similarly, the master position signal is always inverted in the program and them sampled to ensure that the position control rotational direction is always opposite so that the knifepoint on the follower meets the cut point on the master correctly. Thus, although the master position signal is actually linearly increasing, all the plots resulting from the programs show a linearly decreasing signal since there is an inversion taking place within the program. One noticeable feature of the sampled signals is the presence of noise from the servo-systems themselves. Although it does not present a considerable problem in the case of the one-to-one controller, noise is a foreseeable problem in the future for the double-cut controller.

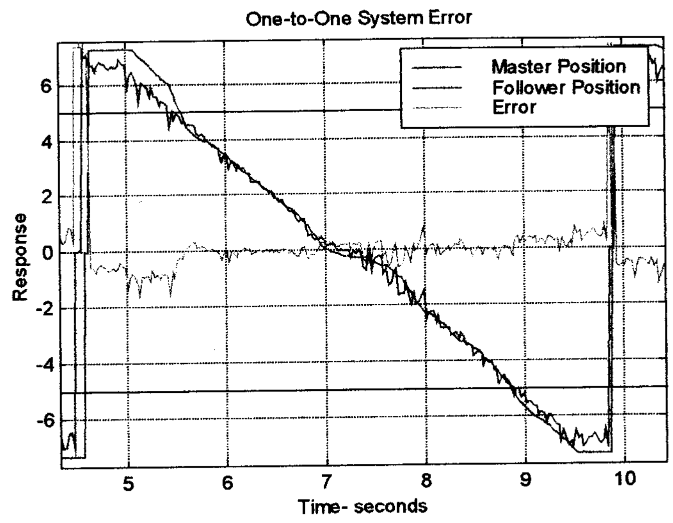


Figure 5.9- One-to-one controller error response. During position control, the error between the master and follower signals is minimal. The differences that are seen can mainly be attributed to the noise present in the master position signal.

<u>Implementation of a Double-Cut Controller</u>

A double-cut controller is the overall goal of this study since the master output potentiometer disk is fitted with two target points, which are 180° apart from each other. For one revolution of the master potentiometer, there are two target points that the follower potentiometer must track through the pre-defined zone. In between the zones, the follower must speed up to catch the other target point of the master. There are two plausible approaches to this problem that can be tested by using the one-to-one controller program with some adjustments. Approach 1, illustrated in Figures 5.10 and 5.11, is a sequence which does not interfere with the sampled master position signal. This approach chooses to explicitly define two distinct zones for the sampled master position signal in which position control is performed (red), and two distinct zones in which speed control is performed (blue). This approach has the advantage of keying directly off of the master position signal, but also has the disadvantage of having a large number of switch points to program. Also, the switch from speed control to position control in Cut 1 (Fig. 5.11) could occur near the physical break of the potentiometer, causing inadequate system response.

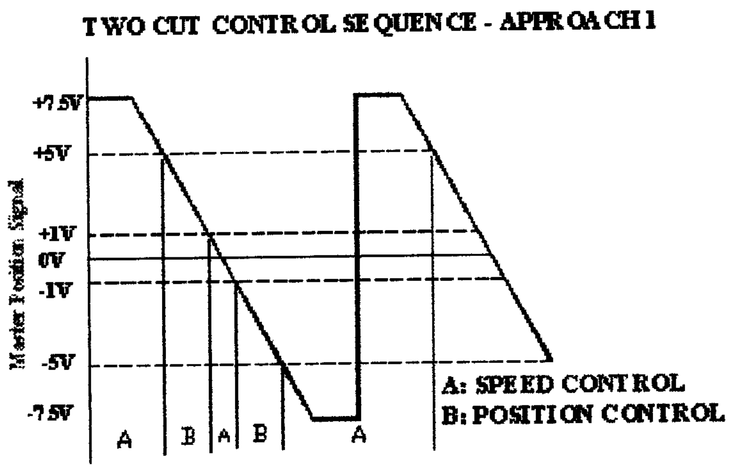


Figure 5.10 – Approach 1 to the double cut controller. The original master position signal is divided up into explicit phases for position or speed control and all the switching is done based off of this signal. The disadvantage is that numerous switches need to be programmed.

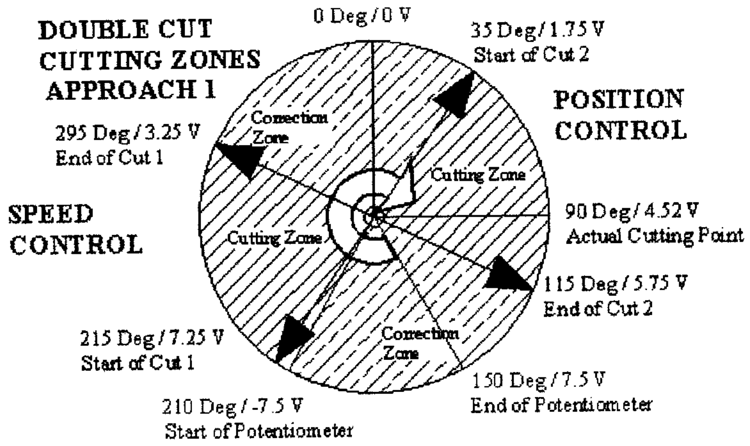


Figure 5.11 – Position and speed zones of Approach 1 as seen on the master potentiometer. For an 80° position control zone, the follower must be able to speed up and catch the next tracking phase, one of which starts near the end of the physical break of the potentiometer.

Conversely, Approach 2, called the *sawtooth double-cut* controller, chooses to take the master position signal and manipulate it algebraically within the program. Since the master position signal is inverted within the program, the cutting zone is in the negative region of the master position signal as it goes from 0V to 7.5V. Thus, if it is possible to take the positive portion of the sampled master position signal and make it negative, such that the resulting signal resembles a sawtooth pattern (Figure 5.12), a double cut can be achieved. As is the case with Approach 1, there are some advantages and drawbacks. The drawback, in this case, is that the sawtooth signal is generated by an algebraic manipulation within the program that states that if the reference signal from the master is positive, then subtract 7.5 from it (Eqn. 5.1).

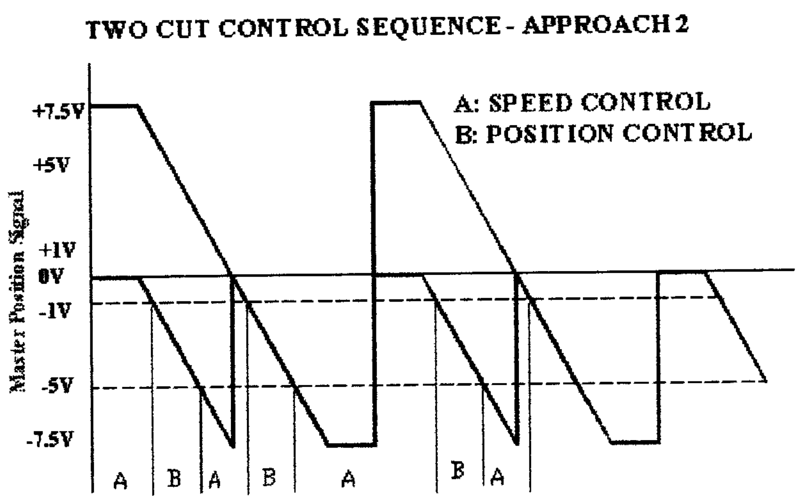


Figure 5.12 – Sawtooth double-cut controller master signal generation. The entire positive half of the master position signal is mapped onto the negative half by algebraic manipulation within the controller program. This allows for only one definition for the switching logic.

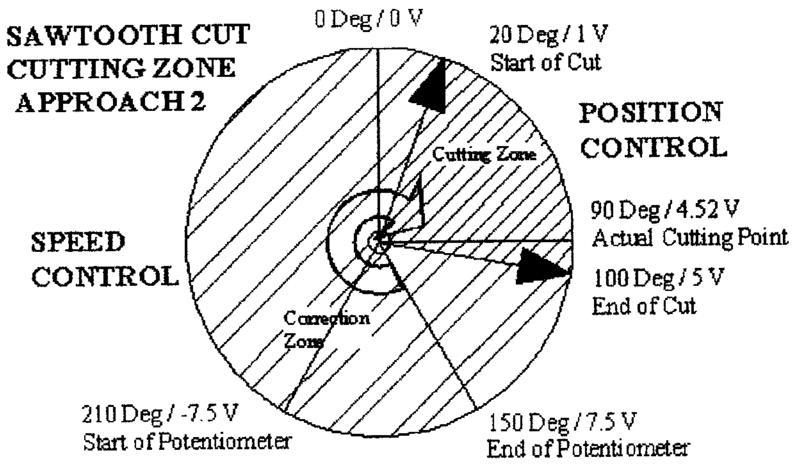


Figure 5.13 – Cutting and correction zones for the sawtooth double-cut controller. The big advantage of this approach over the first is that only control is achieved using only one position control zone and one speed control zone.

Eqn. 5.1 *if reference_position≥ 0, then*
reference_position = reference_position – 7.5
else reference_position = reference_position

This algebraic manipulation becomes highly dependent on the op-amp hardware controlling the scaling from the original ±15V to the ±7.5V signal. If there is a considerable change in the master signal or op-amp, the mapping to the negative half of the axis will not function properly. However, assuming that the op-amp can handle the scaling for the Comdyna with relatively accuracy, Approach 2 has one great advantage over the previous approach. By creating the sawtooth signal, it is possible to define only one position control (cutting) phase, and only one speed control (catch-up) phase for the control sequence, which also means only one set of switch points. Because of this advantage, Approach 2 is implemented into the controller program *K222.c* (found in the appendices), whose flow chart is shown in Figure 5.14. The flow is similar to that of the one-to-one controller, except that for the newly calculated reference signal, the control sequence always enters into speed control and then position control, which is opposite to the previous controller. Also, because of that reversal, the switching logic also inverts.

Once the sawtooth double-cut controller program is operational, the controller performance needs to be optimized. Using the sawtooth idea with the newly calculated master reference position signal, there are three variables that can be looked at when trying to optimize this particular controller. First, the limits for the position control phase can be varied. Second, the speed of rotation for the master potentiometer can be adjusted, and finally, the speed multiplier in the speed control phase can also be modified. For the purposes of this study, the latter two categories will be adjusted and the system responses analyzed. The limits for the switching between the two controllers is set to –1V and –5V, corresponding to approximately an 80° position control phase. The cutting point always occurs at –4.52V or at 90° clockwise from the top of the potentiometer (0V) during one revolution. Furthermore, the speed of the master servo, or the period of revolution, can be varied quite a bit with the experimental setup. Since the original controllers were designed with a 4-second period in mind, the period is varied between 4 and 8 seconds. Finally, the speed multiplier in the program during the speed control phase, which determines how much faster the follower rotates than the master, is varied between 2X and 3.5X. Table 5.1 summarizes the cases that are run using the experimental setup and the sawtooth double-cut controller program.

2 CUT CONTROLLER FLOW CHART

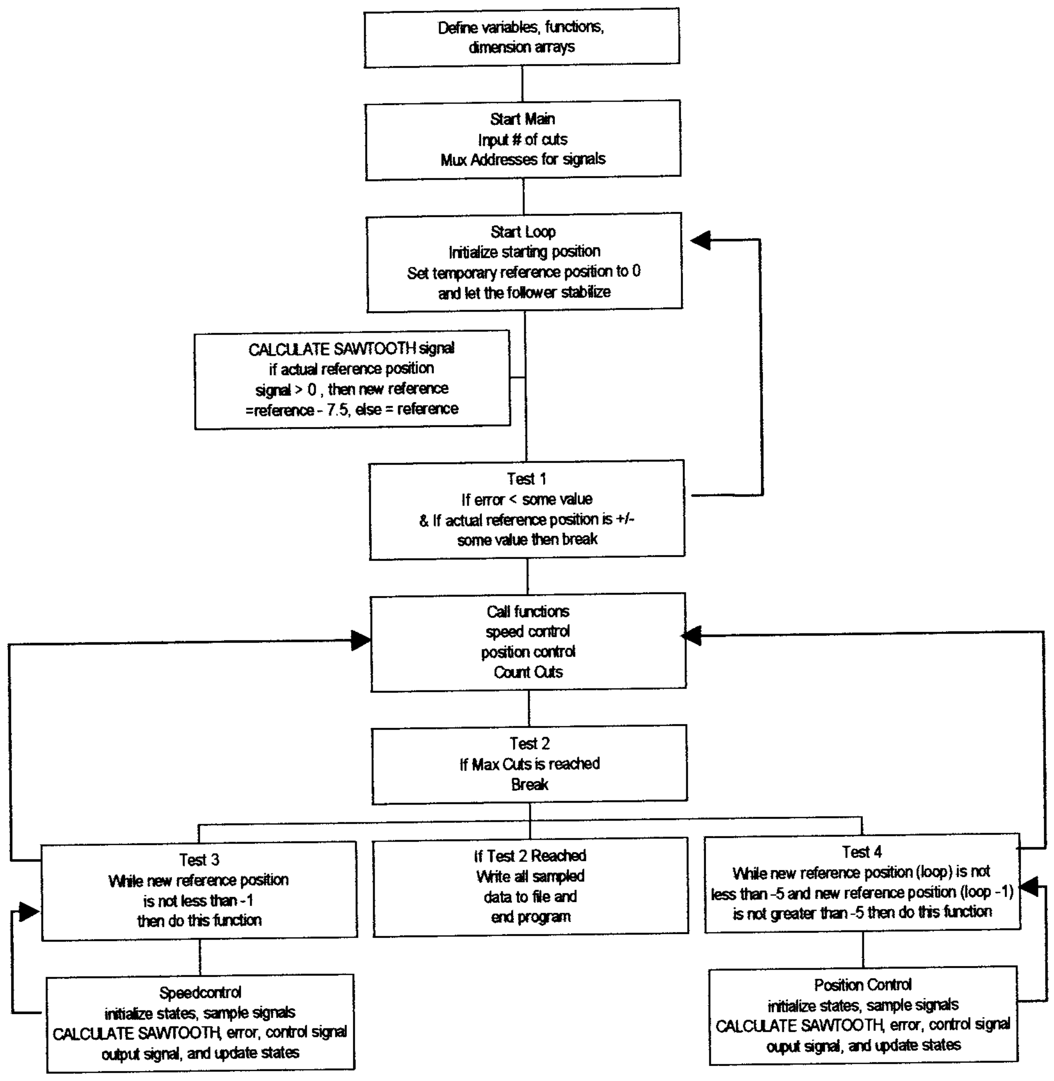

Define variables, functions, dimension arrays
Start Main
Input # of cuts
Mux Addresses for signals
Start Loop
Initialize starting position
Set temporary reference position to 0
and let the follower stabilize
CALCULATE SAWTOOTH signal
if actual reference position
signal > 0 , then new reference
=reference - 7.5, else = reference
Test 1
If error < some value
& If actual reference position is +/-
some value then break
Call functions
speed control
position control
Count Cuts
Test 2
If Max Cuts is reached
Break
Test 3
While new reference position
is not less than -1
then do this function
If Test 2 Reached
Write all sampled
data to file and
end program
Test 4
While new reference position (loop) is not
less than -5 and new reference position (loop -1)
is not greater than -5 then do this function
Speedcontrol
initialize states, sample signals
CALCULATE SAWTOOTH, error, control signal
output signal, and update states
Position Control
initialize states, sample signals
CALCULATE SAWTOOTH, error, control signal
ouput signal, and update states

Figure 5.14- Flow chart for the sawtooth double-cut controller programmed as K222.c that can be found in the appendices. The logic is much like the one-to-one controller with the exception of the calculation of the sawtooth signal within the program. For one of the two cuts during one revolution, the follower is tracking a computer generated signal.

Table 5.1 – Cases for Optimization of Sawtooth Double-Cut Controller Performance

	CASE 1 - REFERENCE POSITION REVOLUTION PERIOD – 4 SECONDS						
CASES	1A	1B	1C	1D	1E	1F	1G
FOLLOWER SPEED MULTIPLIER	2X	2.25X	2.5X	2.75X	3.0X	3.25X	3.5X
	CASE 2 - REFERENCE POSITION REVOLUTION PERIOD – 6 SECONDS						
CASES	2A	2B	2C	2D	2E	2F	2G
FOLLOWER SPEED MULTIPLIER	2X	2.25X	2.5X	2.75X	3.0X	N/A	N/A
	CASE 3 - REFERENCE POSITION REVOLUTION PERIOD – 8 SECONDS						
CASES	3A	3B	3C	3D	3E	3F	3G
FOLLOWER SPEED MULTIPLIER	2X	2.25X	2.5X	2.75X	3.0X	N/A	N/A

Results of Double Cut Controller

Of the 17 cases of controller parameters that were run, Case 3C provided the overall best performance based on several criteria that are discussed in the next section. Figures 5.15 –5.17 illustrate the overall system responses.

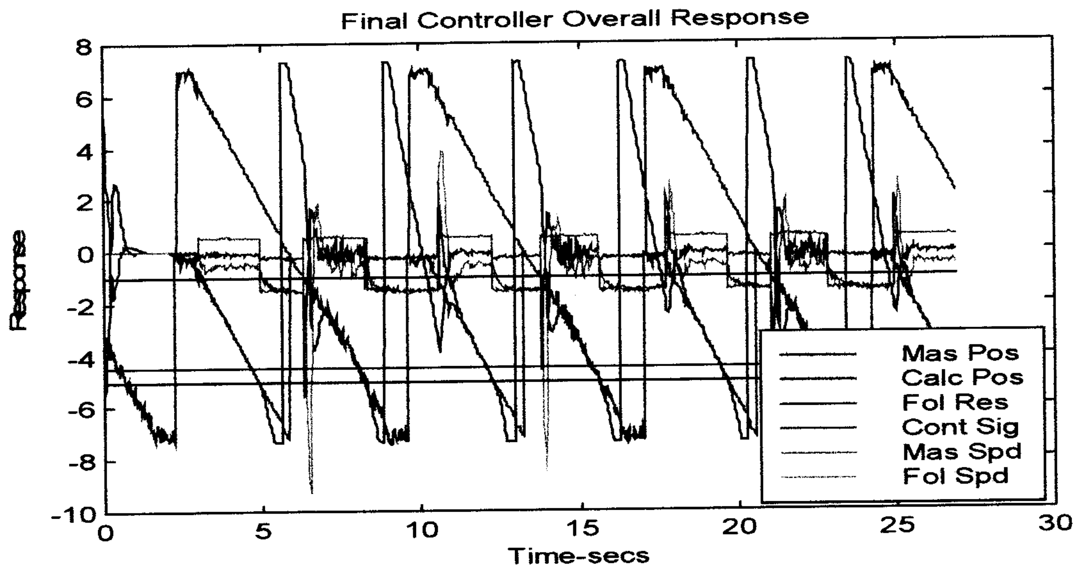


Figure 5.15- Overall response from best case sawtooth double cut controller. Horizontal black lines mark the switching limits and cut point. The controller provides adequate control during the cutting zone, but there are problems during the switching.

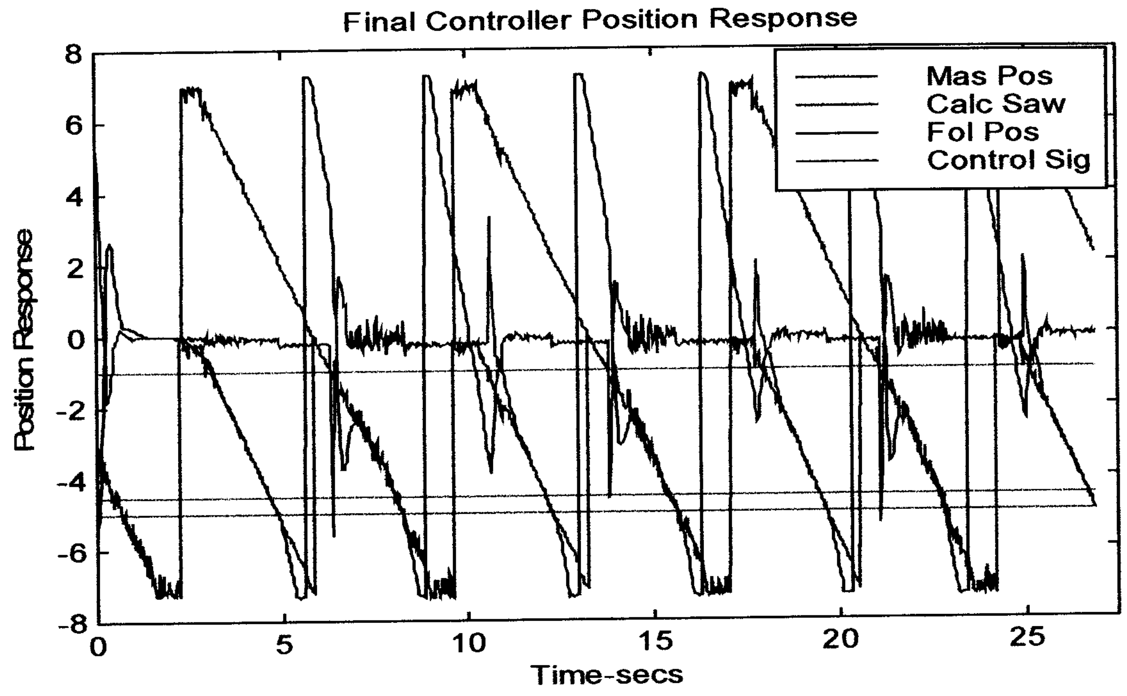


Figure 5.16 – Overall sawtooth double-cut controller position response. The switching limits and cut point are marked in green. Tracking during the cutting phase is adequate, but there are noticeable differences in the transient response of the position controller immediately after switching from speed to position control.

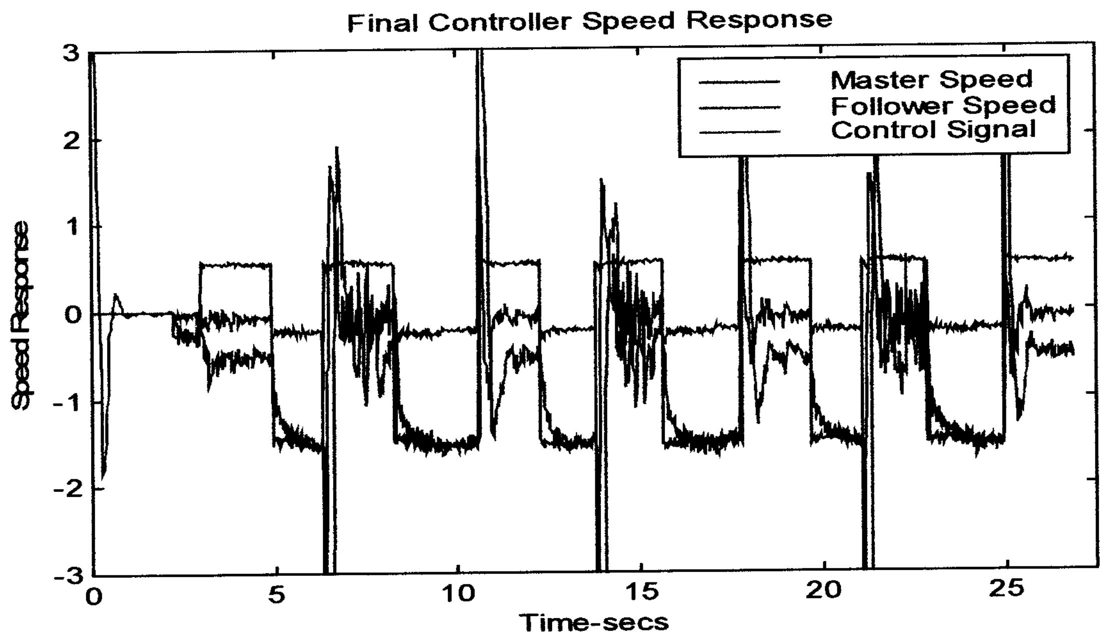


Figure 5.17 – Overall sawtooth double-cut controller speed response. The master signal, during speed control phases is inverted by a factor of -2.5X. A noticeable aspect of the response is the high spikes in the follower response and control signal at the switching limits. However, the control signal during speed control is very low in magnitude.

Analysis of Results and Conclusions

Case 3C is chosen as the best case double cut controller from the experimental runs. This case is chosen because the follower tracks the master signal through the cutting zone with minimal adverse conditions (overshoot) in the transient response when switching from speed to position control. Also, the error at the point of the cut is minimal.

Although the plots indicate that the cutting during the position control phase is adequate in the sense that the follower tracks the master position signal closely through the cutting point as seen in Figure 5.18, there are some problems present in the controller performance. One of the first noticeable aspects of the controller from the responses shown above is the fact that there is a considerable overshoot and oscillation of the follower position signal when the control sequence switches from speed to position, which is the transient response of the position controller. The overshoot can be a function of many things, including the controller itself. When the switching takes place, although the follower position may be near the master position, there is an error in the speed, or slope, of the follower signal, which generates a large error, causing the controller to try to compensate very quickly. This results in the overshoot and oscillation found in the system responses. Also, one of the cuts takes place where there is no physical break in the potentiometer, while the other takes place immediately following the sawtooth calculation in the program (without the 17% deadtime due to the break). Thus, in one cut, there is more time for the controller to be in the speed zone, allowing for more compensation, while there is less time for the other speed controller phase to allow the follower to catch up. This results in a larger

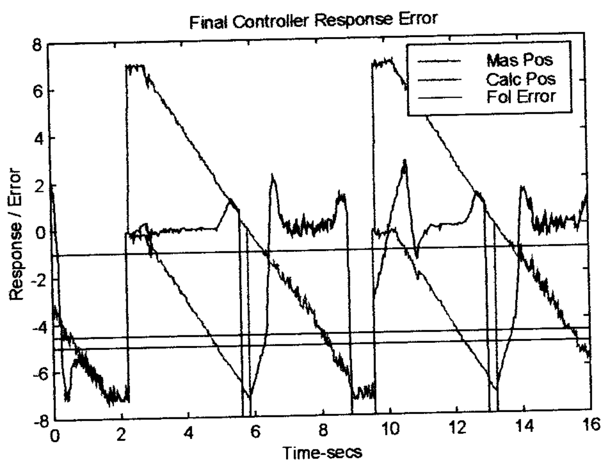


Figure 5.18- Error response for the sawtooth double cut controller. Between the cutting zone limits, the error is minimal, indicating that the follower tracks the master position through the zone. Noise in the actual signal leads to noise appearing in the error signal as well.

initial error when switching from speed to position control. The result is that the position controller has to compensate more, and even though the magnitudes of the overshoot remain the same for both cuts, the follower starts to track the reference at a later stage in one cut than another.

In Figure 5.19, the effect (shown inside the circle) is pronounced. The overshoot / unwanted transient response is much more evident in the cutting zones where the switch takes place immediately in the program, as the cut shown between 21 and 25 seconds in the figure. When the potentiometer goes through the break, allowing for the speed control to let the follower catch up, there is less initial error, and thus less compensation during the transient portion of the position control, as seen by the cut from 17 to 21 seconds above. However, even though there is this unwanted transient portion, one thing to keep in mind is that the overshoot never causes the follower signal to reach the cut point line, which in terms of the physical rotary knife cutter would mean that the knife blade would reach the material on the conveyor, causing an unwanted cut, and then return to its normal tracking and create another cut. Since this does not occur in this model, the controller is deemed acceptable.

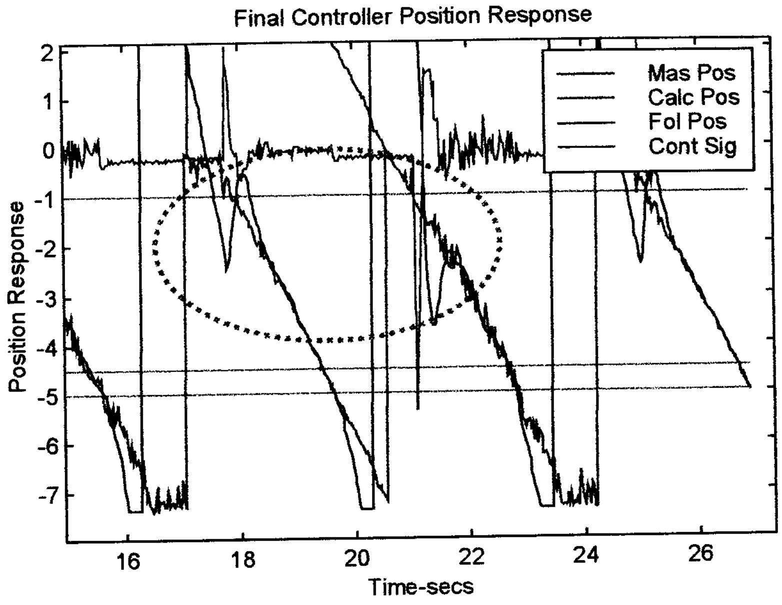


Figure 5.19- Effect of cutting zone performance of the potentiometer going through the break versus simply being switched in the program. As the potentiometer goes through the break, it allows for more time in the correction zone (speed control), yielding better performance with less overshoot.

The last piece of information to look at from the plots of the responses is the effect noise has on the controller performance. Figure 5.20 shows a typical noise related problem that occurs during control sequences. The blue line is the calculated sawtooth signal upon which all the switching logic is based. From the plot, it is seen that the noise inherent within the signal causes the signal to jump back and forth between one of the limits set at –1, as seen within the dotted circle. Since the switching logic is based off a comparison between the reference signal and the switch limit, the controller responds by switching too early from the speed control phase to the position control phase,

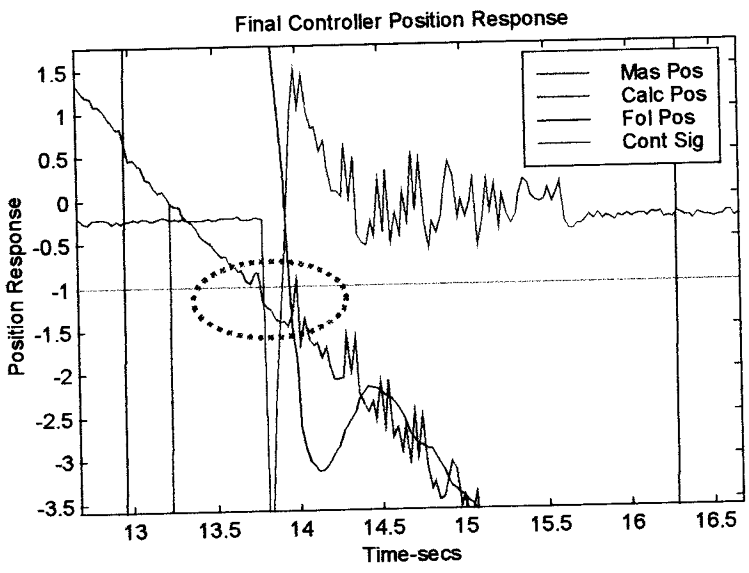


Figure 5.20 – Effect of noise in the switching logic. The reference signal, in blue, crosses and re-crosses the switch limit (inside the circle), causing the controller to switch control modes inadvertently. The result is poor transient performance.

resulting in a large error which needs to be compensated for by the controller. This can be seen from the figure above as there is a large spike in the control signal when noise causes an early switch in the program.

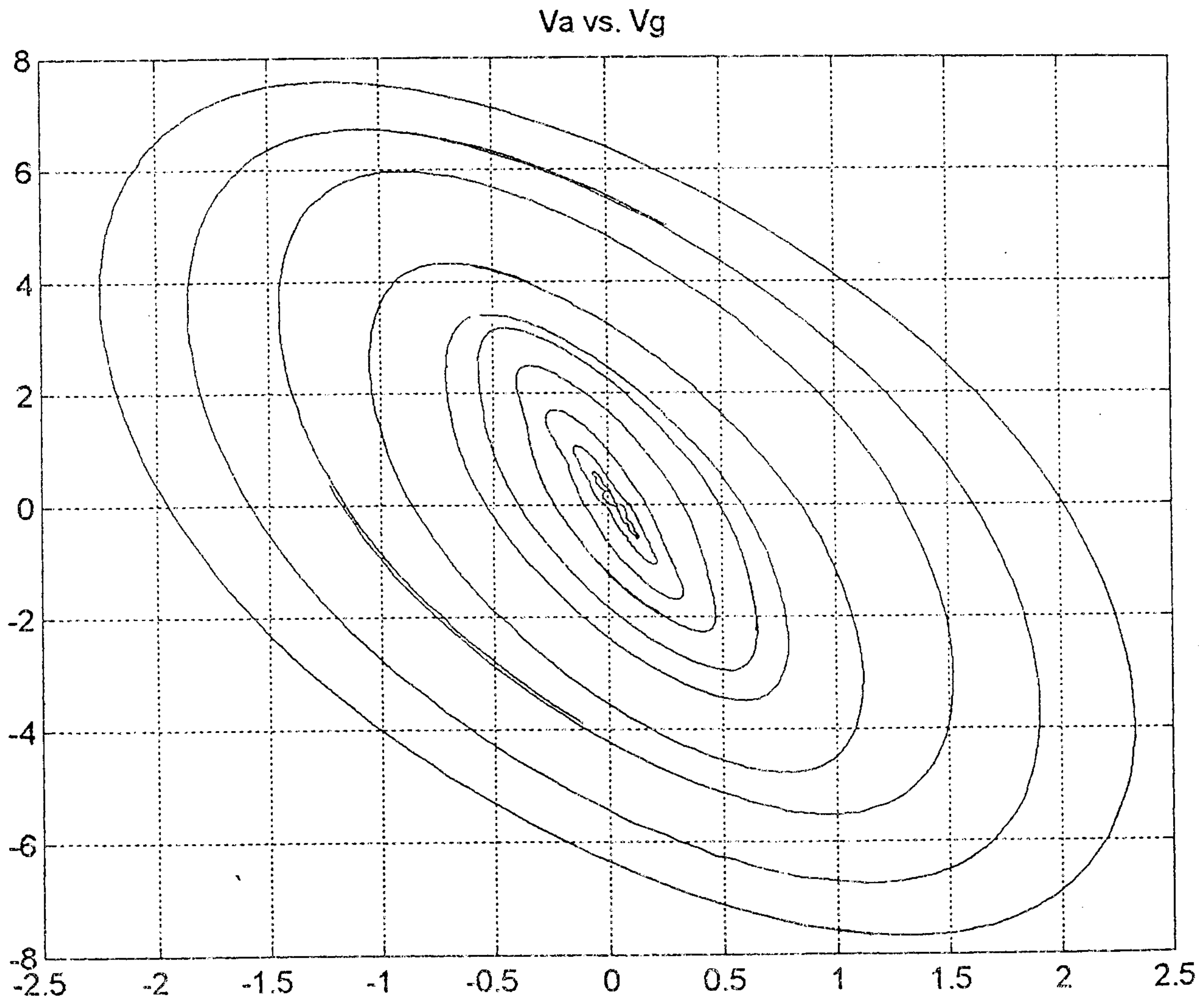
In conclusion, it is possible to state that the overall controller performance is adequate, even though there are several problems associated with the final double cut controller. It is seen from the experimental model that slower periods of revolution for the master output potentiometer generally yields better overall performance, although it introduces a considerable amount of noise to the system. Finally, the practicality of direct digital implementation of controllers designed using analog techniques and digital approximations is shown in this study.

Recommendations for Future Research

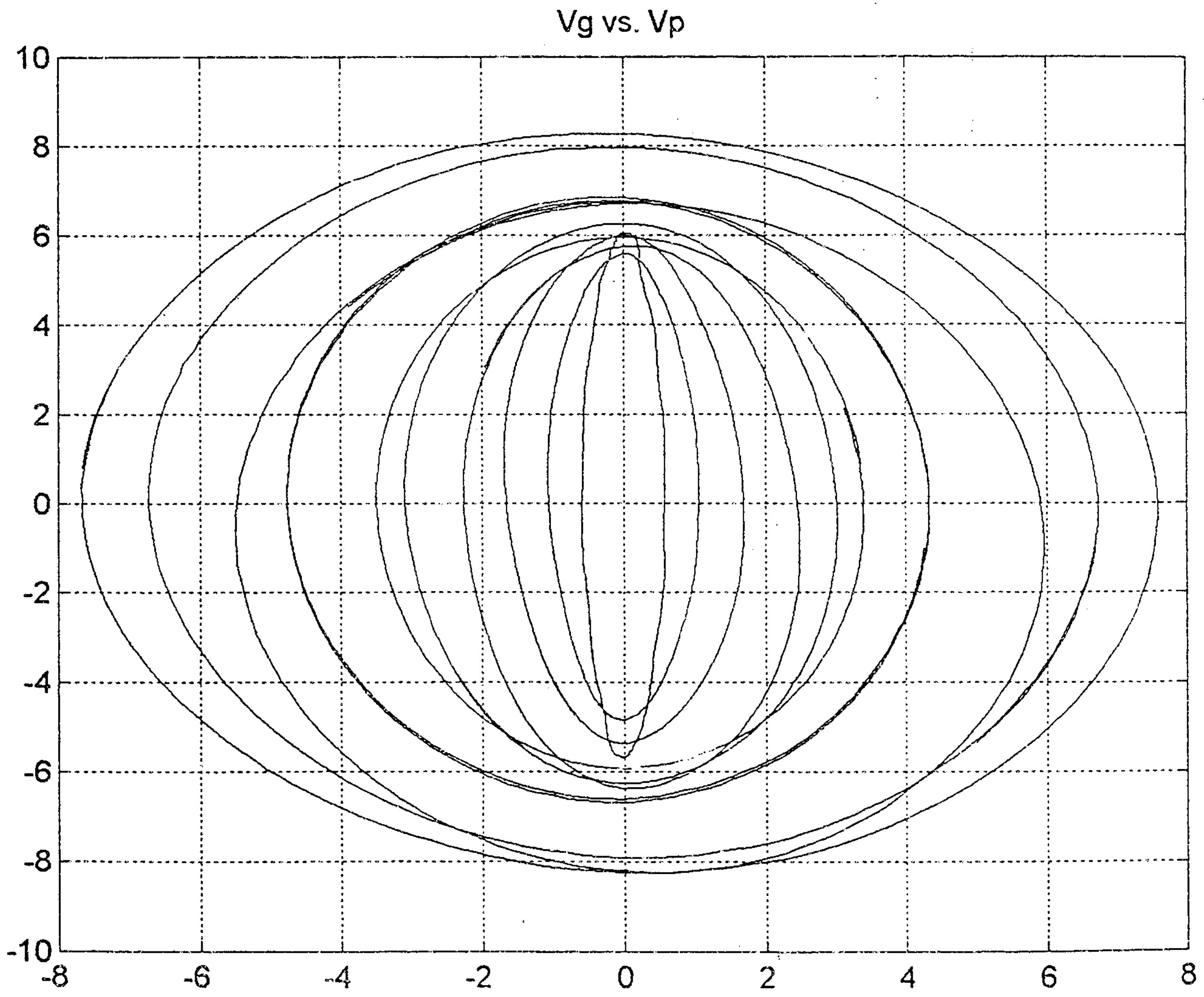
There are several sub-topics of interest within the design and implementation of the rotary knife cutter controller that can be discussed for future research. Foremost is the possibility of implementing an altogether different type of controller, such as a fuzzy logic controller, for the same purpose. However, there are several concepts that can be further researched using the PID controllers presented here. First is the introduction of a first-order filter, either in terms of hardware, or as software within the controller program, which can act to cut down the noise that is present within the signals. Also, further variables, such as the switch limits, can be investigated in trying to optimize the controller. From a design standpoint, controller performance can be further investigated by looking into the initial conditions of the controllers such that the transient response can be improved. As a matter of better control equations, the controller states can be investigated and implemented in a different state-variable model that present (a state feedback model should be investigated). Finally, as a different approach to the sawtooth model presented in this study, an input-output model can be looked into in order to achieve the same sawtooth reference signal.

APPENDICES

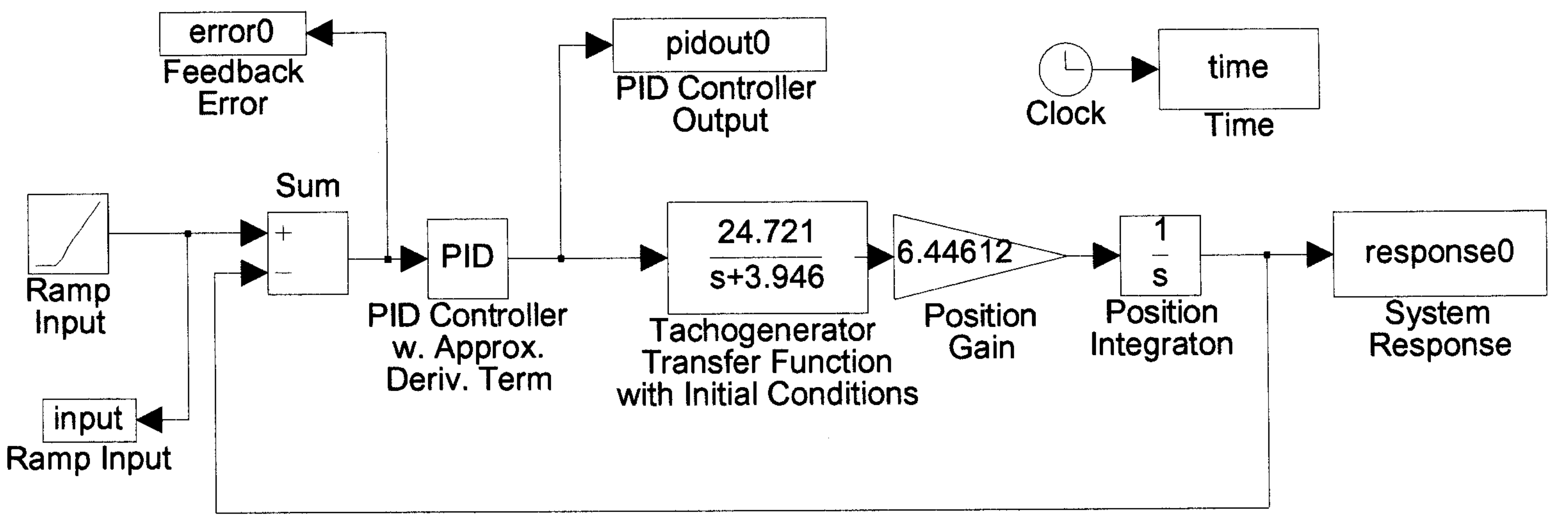
APPENDIX A: EXPERIMENTAL LISSAJOUS PATTERNS FOR SERVO MODELING



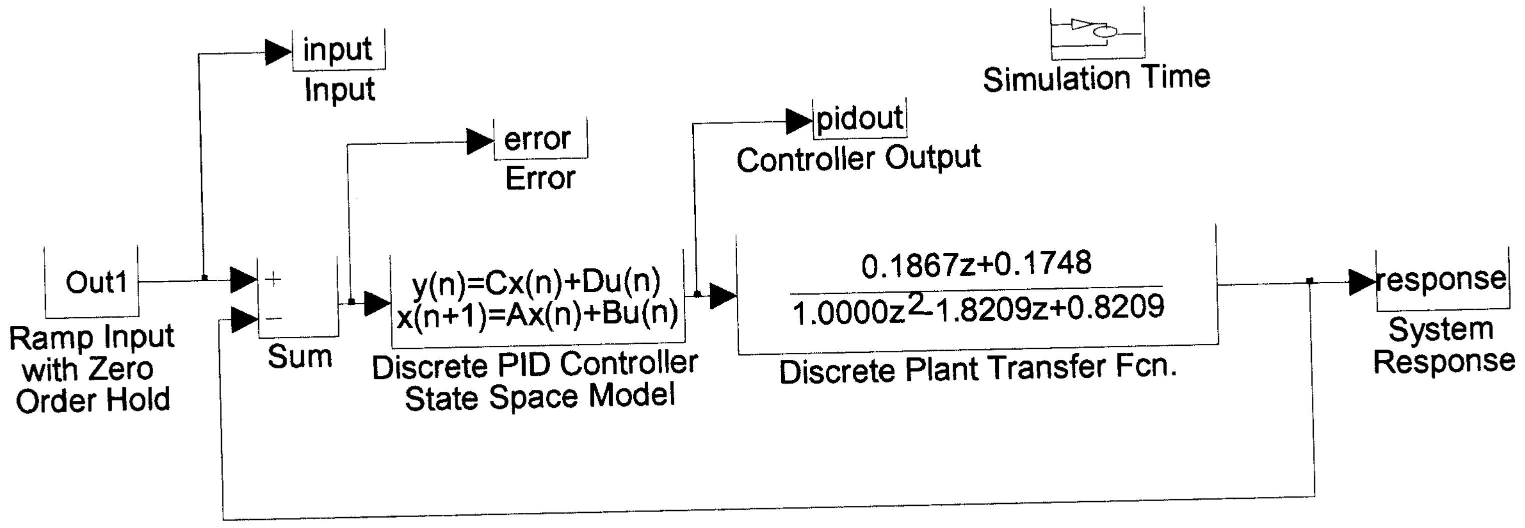
APPENDIX A: EXPERIMENTAL LISSAJOUS PATTERNS FOR SERVO MODELING



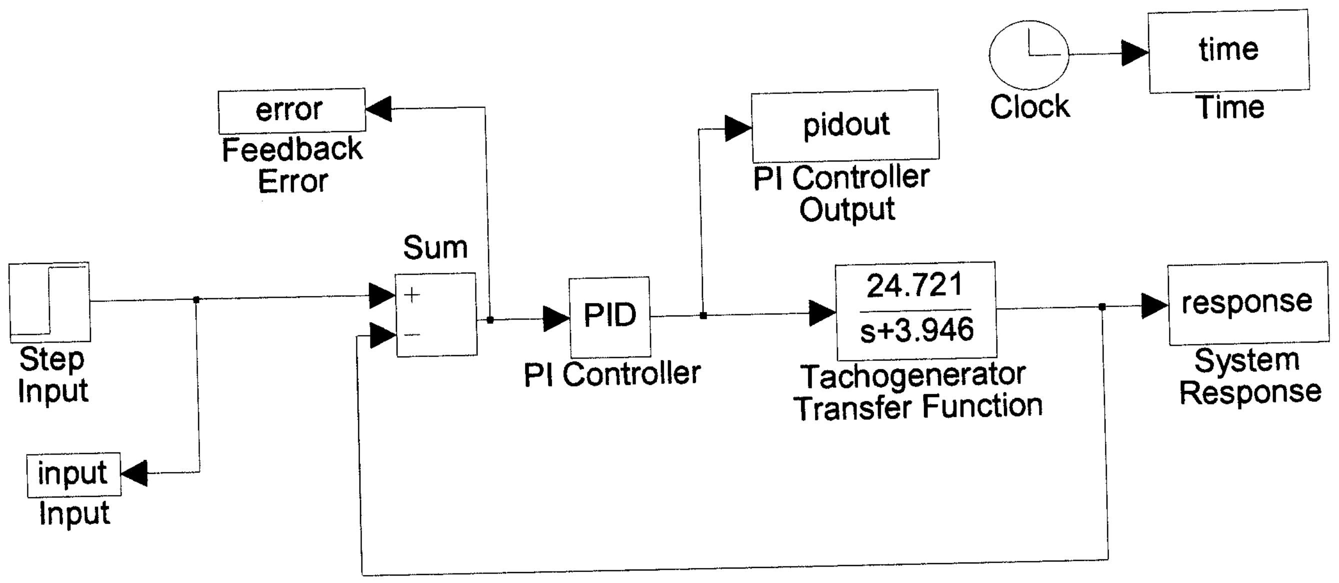
SIMULINK MODEL PID1.MDL - ANALOG SYSTEM DESIGN



SIMULINK MODEL PID2.MDL - DISCRETE PID POSITION CONTROL TESTING



SIMULINK MODEL PI1.MDL - ANALOG SYSTEM DESIGN
SPEED CONTROL



APPENDIX C: Matlab .m files

```
% ***********************************************
% * Matlab program for finding PID              *
% *    Controllers Parameters                   *
% ***********************************************
% *       CONTROLLER1.M                         *
% ***********************************************
% This Matlab program is used to find           *
% PID controller parameters of the system       *
% using the root-locus of the closed loop,      *
% unity feedback, system response. The          *
% output of the program are the PID control     *
% parameters Kd,Kp,Ki, and N, as well as        *
% the PID transfer function. Also, the first    *
% order hold ('foh') discrete approximation     *
% of the controller is also given.              *
% ***********************************************

num1=[3.945981*6.26494*6.446128];  % Plant numerator
den1=[1 3.945981 0];           % Plant denominator
r = roots(den1);               % Poles of denominator
r1= r(2,1);                    % Set controller numerator a
r2= r(2,1)-0.5;                % Set controller numerator b
r12=[r1, r2];                  % Controller numerator zeros


% Set PID controller parameters
num2=poly(r12);                       % (s+a)(s+b)/
den2=[0 1 0];                         %    s

nums=conv(num1,num2);     % Put Plant and
dens=conv(den1,den2);     % PID Controller together

rlocus(nums,dens)         % Closed loop system root locus
grid, hold on
zeta=[0.707];
% zeta defines damping corresponding to 1% OS
sgrid(zeta,0);    % Overlay damping ratios on root locus
title('Rootlocus')
hold
s1=ginput;   % Allow user to pick a point off the root locus
s1a = (s1(1,1));% so that point can be used for finding K
```

```
%*********************************************************
% Calculate PID Gains from root locus point selection
*
%
*********************************************************
% Derivative Gain
Kd =  -(polyval(dens,s1a) / polyval(nums,s1a));
% Proportional Gain
Kp=Kd*num2(1,2);
% Integral Gain
Ki=Kd*num2(1,3);
% *******************************************************

%Prompt user for approximate derivative term N and wait for
keystroke

N=input('Enter approx. derivative term N & hit any key ')
den2=[1 N 0];  % Set new PID denominator dependent on N

% *****************************************************
%Calculate transfer function for PID controller
Gpidnum=[(Kp+Kd*N) (Ki+Kp*N) (Ki*N)];
Gpidden=[1 N 0];


%Calculate SS representations of system and PID Controller
[A,B,C,D]=tf2ss(num1,den1); % Analog system model

% SV model for servo system
% ANALOG PID CONTROLLER
[A1,B1,C1,D1]=tf2ss(Gpidnum,Gpidden); % SV model for PID
Controller

% Find Discrete State Space representations of each
% Continous SS representation
SERVO=ss(A,B,C,D);
CONT=ss(A1,B1,C1,D1);
% Discrete representation of system model using
% Zero Order Hold
SERVOd=c2d(SERVO,0.025,'zoh');
CONTd=c2d(CONT,0.025,'zoh');

% PRINT OUT ANALOG SYSTEM MODEL TRANSFER FUNCTION
print('ANALOG SYSTEM MODEL TRANSFER FUNCTION')
print('Numerator = '),num1;
print('Denomiator = '),den1;
```

APPENDIX C: Matlab .m files

```
% Matlab program for plotting Simulink model responses
%         plotcontroller.m
% Subscripts on system variables must be varied if they are
% changed in the Simulink model.




figure
plot(time,input,'r'),grid,hold on
plot(time,response0,'b')
plot(time,response5,'m')
plot(time,response75,'c')
plot(time,response10,'g')
title('PID SIMULINK MODEL RESPONSE')
ylabel('Response'),xlabel('Time-secs')
legend('Input','0 IC','5 IC','7.5 IC','10 IC')


figure
plot(time,error0,'b'),hold on,grid
plot(time,error5,'m')
plot(time,error75,'c')
plot(time,error10,'g')
title('Tracking Error ')
ylabel('ERROR'), xlabel('Time-secs')
legend(''0 IC','5 IC','7.5 IC','10 IC')


figure
plot(time,pidout0,'b'),grid,hold on
plot(time,pidout5,'m')
plot(time,pidout75,'c')
plot(time,pidout10,'g')
title('PID Controller Output')
ylabel('Controller Output Voltage'),xlabel('Time-secs')
legend(''0 IC','5 IC','7.5 IC','10 IC')
```

APPENDIX C: Matlab .m files

```
% Matlab program dataplot.m
% used to plot experimental data results
% for the double-cut sawtooth controller


testname=input('Please input the *.dat filename :','s')
eval(['load ' testname '.dat']);
eval(['samples=' testname ';']);
testcase=input('Please input the testcase number :','s')
maxsample=max(size(samples));
novector=min(size(samples));

t=linspace(0,(0.025*maxsample),maxsample);
color=['r' 'b' 'k' 'c' 'm' 'g' 'y'];

for I=1:novector,
   plot(t,samples(:,I),color(I));
        hold on
end

grid
title('Experimental Data Collection from Lab');
xlabel('Time- seconds')
ylabel('Response')
legend('Mas Ref','Calc Ref','Pos Res','Cont Sig','Mas
Speed','Speed Res',0)

text(7.25,7.5,testname)
text(3.25,7.5,testcase)
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
/* Digital Compensator Program K222.c */
/* Position + Speed Rotary Knife Cutter Controller */
/* Control using Discrete PID SS Models */
/* NOTE THAT h=0.025s ALWAYS WHEN RUNNING THIS PROGRAM */

#include <stdio.h>
#include <conio.h>
#include <math.h>

#define basadr 0x300

void setup_das8(unsigned cnt2, unsigned cnt1);
void high_low_bytes(unsigned *ptrnum, unsigned *ptrlo, unsigned *ptrhi);

int read_a2d(void);

void set_d2a(int hb, int lb);
void pause(void);
void start_clock(unsigned lo, unsigned hi);
void set_gp6_op(void);
void set_gp6_ic(void);
void check_clock(void);
void wait_eoc(void);
void find_rise_edge(void);
void poscontrol(void);        /* Add controllers as separate functions */
void speedcontrol(void);
float get_a2d_value(unsigned next_adrs,float *ptr_a2d);
void limit_out_u(float *ptr_u,float *ptr_d2a,float *ptr_a2d,
int *ptr_max,int *ptr_min);

float range,a2d,d2a,volts=10.0,
/*    Define all position variables here */
      ypos[2001],u[2001],
      refpos[2001],refpos1,refpos2[2001],       /* new references */
      errpos,uprepos,nx1pos,nx2pos,x1pos,x2pos,

      /* Define all speed variables here */
      yspd[2001],refspd[2001],errspd,
      uprespd,nx1spd,x1spd,

      /* Define all position controller PID Parameters here */
      f1pos=1.6538, f2pos=-0.6538, f3pos=1.0, f4pos=0,
      g1pos=1,    g2pos=0,
      c1pos=-0.2148,    c2pos=0.2234,
      d1pos=1.2050,

      /* Define all speed controller PI Parameters here */
      f1spd=1.0,  g1spd=1.0, c1spd=0.033333,    d1spd=0.3497;
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
    /* Define User Input Parameters */
      unsigned tsamp,cnt0,cnt1,cnt2,ip,hi0,lo0,ncuts;
      unsigned loop,npts,ypos_adrs,yspd_adrs,N;
      unsigned nbits=12,refpos_adrs,refspd_adrs;
      int code,max_code=2047,min_code=-2047;

      /* Add / Remove variables as needed to change order of controllers*/
      /* ------------------------------------------------------------- */
      /* START OF MAIN CODE */
 main()
{/* Initialize  */
FILE *fid;
      set_d2a(0,0);/* Initialize controller output to 0 */

    /* USER INPUTS */
      printf("Enter sample period in milliseconds :\n");
      scanf("%u",&tsamp);

      printf("Enter number of cuts required [<=20] :\n");
      scanf("%u",&ncuts);

      printf("Enter address for follower position:\n");
      scanf("%u",&ypos_adrs);

      cnt2 = 395;
      cnt1 = 10*tsamp;
      cnt0 = 6*tsamp;
      range = (float)(1<<(nbits-1));
      a2d = volts/range;
      d2a = range/volts;
      setup_das8(cnt2,cnt1);
      high_low_bytes(&cnt0,&lo0,&hi0);
      outp(784,ypos_adrs);    /* Initialize Y MUX address for POSITION */


      printf("Enter address for master position :\n");
      scanf("%u",&refpos_adrs);

      printf("Enter address for follower speed :\n");
      scanf("%u",&yspd_adrs);

      printf("Enter address for master speed :\n");
      scanf("%u",&refspd_adrs);

      pause();                    /* Prompt to begin */
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
/* Initial Values for Position and Speed for 1st loop */
/* First mini loop brings the follower to 0 and lets it stay there */
            /* POSITION  CONTROL*/
            x1pos=0;
            x2pos=0;

            x1pos=f1pos*x1pos+f2pos*x2pos;
            x2pos=f3pos*x1pos+f4pos*x2pos;
            uprepos = c1pos*x1pos+c2pos*x2pos;

      find_rise_edge();
      {outp(797,64);}    /* Put the GP-6 in the OP mode */
                         /* end of set_gp6_op */
      refpos1 = 0;       /* Set intialization position to 0*/

      for (loop=0; ;loop++)   /* Let loop run indefinitely */
      { start_clock(lo0,hi0); /* begin sample period */

      ypos[loop]=get_a2d_value(refpos_adrs,&a2d); /* Sample follower position */
      refpos[loop]=get_a2d_value(ypos_adrs,&a2d); /* Sample ref position */
            refpos[loop]= -1.0*refpos[loop];      /* INVERT */


        /* CREATE SAWTOOTH FUNCTION */
        if (refpos[loop]>=0 )
            refpos2[loop] = refpos[loop]-7.25;
            else refpos2[loop]=refpos[loop];

        errpos= refpos1 - ypos[loop];       /* Calculate error */
        u[loop]=uprepos + d1pos*errpos;     /* Update control signal */

      limit_out_u(&u[loop],&d2a,&a2d,&max_code,&min_code);
        /* Output the updated control signal */
            /* Determine next states  */
            nx1pos= f1pos*x1pos + f2pos*x2pos + g1pos*errpos;
            nx2pos= f3pos*x1pos+f4pos*x2pos + g2pos*errpos;

            /* Pre calc for next sample period */
            x1pos=nx1pos;
            x2pos=nx2pos;
            uprepos=c1pos*x1pos + c2pos*x2pos;


if (((refpos1-ypos[loop])*(refpos1-ypos[loop]))<0.5 &&
((refpos1-refpos[loop])*(refpos1-refpos[loop])<0.5))
      break;
/* Check for acceptable error and initial position and break */

check_clock();    /* loop back when time */
            }
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
/*MAIN FOR NEXT LOOP TO DETERMINE CUTTING SEQUENCE AND NO.OF CUTS */

      for (N=1;N<ncuts;N++)
      {speedcontrol();          /* Call speed control function */
            poscontrol();       /* Call speed control function */
                        }
            set_d2a(0,0);
            {outp(797,0);       /* Put the GP-6 in the IC mode */
                        }       /* end of set_gp6_ic */
             pause();
             npts = loop;       /* Set npts to final value of loop */

      /* Write data to file "c:\bon\data1.dat" */
      if ((fid=fopen("c:\\bon\\data\\data1.dat","w")) != NULL)
      {for (loop=0;loop<npts;loop++)
      fprintf(fid,"%10.2f %10.2f %10.2f %10.2f %10.2f %10.2f\n",
      refpos[loop],refpos2[loop],ypos[loop],u[loop],refspd[loop],yspd[loop]);
      fclose(fid);}
                  }/* END OF MAIN */

/* START OF FUNCTIONS THAT ARE CALLED IN MAIN PROGRAM */

void poscontrol(void)
{/* POSITION CONTROL */
            x1pos=0;    /* Initialize control states and control signal */
            x2pos=0;
            x1pos=f1pos*x1pos+f2pos*x2pos;
            x2pos=f3pos*x1pos+f4pos*x2pos;
            uprepos = c1pos*x1pos+c2pos*x2pos;

            find_rise_edge();

while (!((refpos2[loop]<-5) && (refpos2[loop-1]>-5)))
/* Designates check on duration of pos control */

{start_clock(lo0,hi0);  /* begin sample period */
    loop++;             /* update loop index */

/* Sample y & compare to set ref - MAIN CONTROL ALGORITHM */
ypos[loop]=get_a2d_value(refpos_adrs,&a2d);  /* Sample follower position */
refpos[loop]=get_a2d_value(yspd_adrs,&a2d);  /* Sample master position */
yspd[loop]=get_a2d_value(refspd_adrs,&a2d);  /* Sample follower speed */
refspd[loop]=get_a2d_value(ypos_adrs,&a2d);  /* Sample master speed */

      refpos[loop] = -1* refpos[loop];    /* Invert tracking direction */

      /* CREATE SAWTOOTH SIGNAL */
      if (refpos[loop]>=0)
      refpos2[loop]=refpos[loop]-7.25;
      else refpos2[loop]=refpos[loop];
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
errpos = refpos2[loop] - ypos[loop];      /* Find error */
                                          /* USING NEW REFERENCE */
u[loop]=uprepos+d1pos*errpos;             /* Update control signal */

limit_out_u(&u[loop],&d2a,&a2d,&max_code,&min_code);
/* output control signal to d2a converter */

/* Determine states after control sequence */
    nx1pos= f1pos*x1pos+f2pos*x2pos + g1pos*errpos;
    nx2pos= f3pos*x1pos+f4pos*x2pos + g2pos*errpos;
/* Pre calc for next sample period */
        x1pos=nx1pos;
        x2pos=nx2pos;
        uprepos=c1pos*x1pos + c2pos*x2pos;

check_clock();    /* loop back when time */
        }         /* End of loop */
        }         /* End of poscontrol function */

void speedcontrol(void)
     {/* SPEED CONTROL */
    x1spd=0;     /* Initialize states */
    uprespd = c1spd*x1spd;
    find_rise_edge();

    while (!((refpos2[loop]>-5) && (refpos2[loop]<-1)))
    /* Designates duration of speed control */
    {start_clock(lo0,hi0);  /* begin sample period */
        loop++;             /* update loop index */

/* Sample ypos,yspd,refspd,refpos, and update u  */
    ypos[loop] = get_a2d_value(refpos_adrs,&a2d);    /* Sample */
    refpos[loop]=get_a2d_value(yspd_adrs,&a2d);      /* Sample */
    yspd[loop]=get_a2d_value(refspd_adrs,&a2d);      /* Sample */
    refspd[loop] = get_a2d_value(ypos_adrs,&a2d);    /* Sample */
        refpos[loop]=-1.0*refpos[loop];              /* INVERT */


        /* CREATE SAWTOOTH SIGNAL */
        if (refpos[loop]>=0)
        refpos2[loop]=refpos[loop]-7.2;
        else refpos2[loop]=refpos[loop];

        if (N<=1)   /* Set the speed for the first cut only */
        refspd[loop]=-.5*refspd[loop];
        else refspd[loop]= -2.80*refspd[loop];
        /* Set speed for other cuts */
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
            errspd = refspd[loop] - yspd[loop];
            u[loop] =uprespd + d1spd*errspd;
            limit_out_u(&u[loop],&d2a,&a2d,&max_code,&min_code);

            /* Determine states after control sequence */
            nx1spd= f1spd*x1spd + g1spd*errspd;

        /*  Pre-calc for next sample period  */
            x1spd = nx1spd;
            uprespd = c1spd*x1spd;
            check_clock();          /* loop back when time */
                    }               /* End of for loop */
                        }           /* End of speed control function */


void setup_das8(unsigned cnt2, unsigned cnt1)
{unsigned lo, hi;

/* Set counter 2 as a 10 kHz square wave */
    high_low_bytes(&cnt2,&lo,&hi);
    outp(basadr+7, 0xB6); /* cntr 2 as square wave generator */
    outp(basadr+6, lo); /* low byte */
    outp(basadr+6, hi); /* high byte */

/* Set counter 1 as a 1/T Hz square wave */
    high_low_bytes(&cnt1,&lo,&hi);
    outp(basadr+7, 0x76); /* cntr 1 as square wave generator */
    outp(basadr+5, lo); /* low byte */
    outp(basadr+5, hi); /* high byte */

/* Set counter 0 to pulse on terminal count */
    outp(basadr+7, 0x30); /* cntr 0 pulse on terminal count */

}   /* end of setup_das8 */

void start_clock(unsigned lo, unsigned hi)
{
    outp(basadr+4, lo); /* low byte */
    outp(basadr+4, hi); /* high byte */
}

void high_low_bytes(unsigned *ptrnum, unsigned *ptrlo, unsigned *ptrhi)
{
    *ptrlo = *ptrnum & 0x00FF;
    *ptrhi = *ptrnum >> 8;
}
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
/* The pause function */
void pause(void)
{char ch;
    printf("\n Press the ENTER key to continue.");
    while (1)
      {
        ch = getch();
        if (ch == '\r')  break;
      }
}

/* The read_a2d function */
int read_a2d()
{
    unsigned hb,lb;
    int code;
    hb = inp(787);
    lb = inp(786);
    code = 16*hb + (lb>>4);
    return code;
}

/* Set the LDAC  */
void set_d2a(int hb, int lb)
{
    lb = lb & 0x00F0;   /*  old code:  16*(int)lb/16 */
    outp(788,lb);
    outp(789,hb);
}

/* Put the GP-6 in the OP mode */
void set_gp6_op(void)
{
   unsigned byte;
   outp(785,0);          /* set C3 low to put GP-6 in op mode */
   do
      byte = inp(786) & 0x0004;
      while (byte == 0);
}                  /* end of set_gp6_op */

/* Put the GP-6 in the IC mode */
void set_gp6_ic(void)
{
   unsigned byte;
   outp(785,8);          /* set C3 high to put GP-6 in ic mode */
   do
      byte = inp(786) & 0x0004;
      while (byte != 0);
}                  /* end of set_gp6_ic */
```

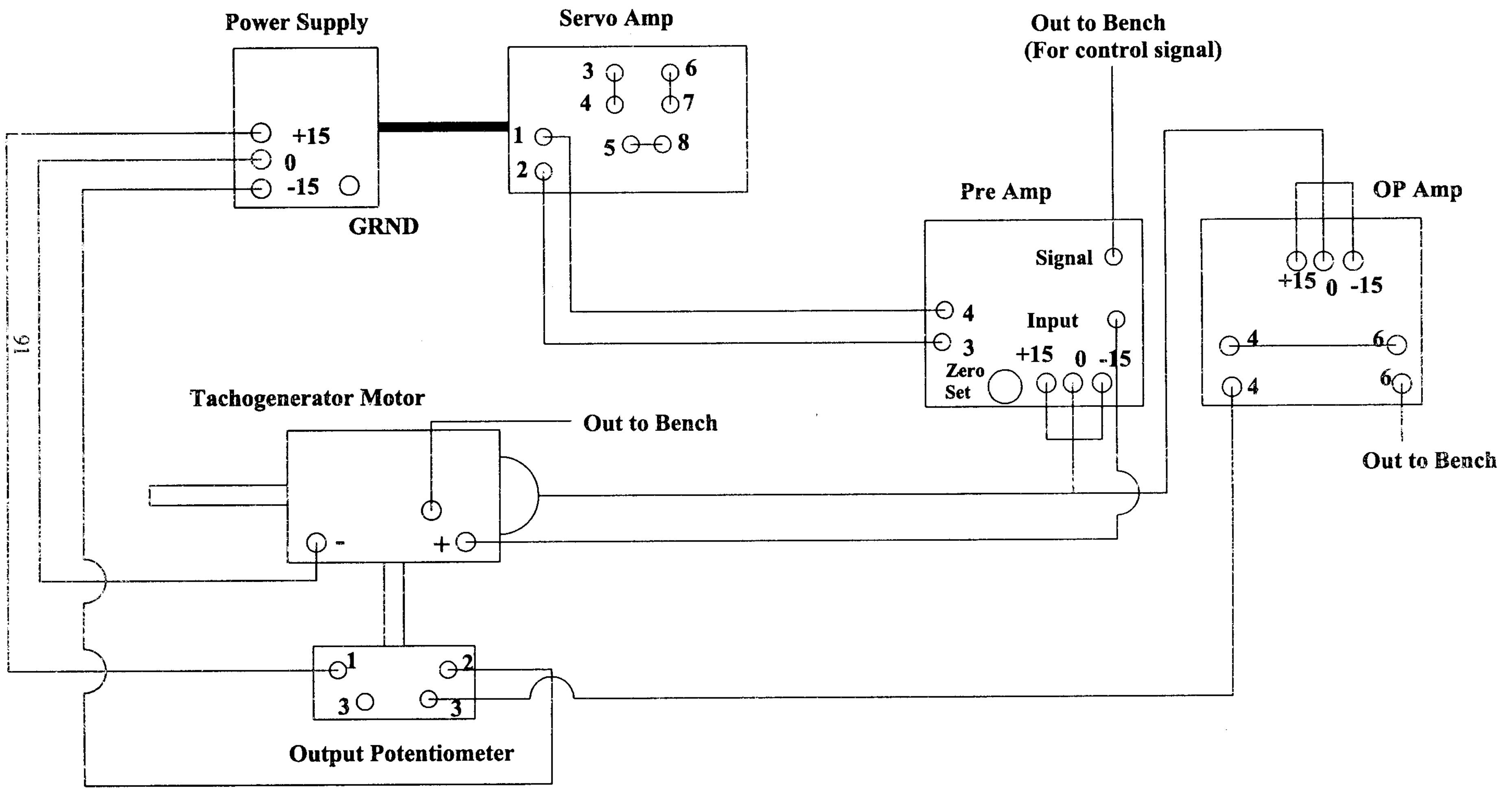
APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
void find_rise_edge(void)
{
   unsigned ip,ip_past;
   ip = (inp(basadr+2))&0x0020;
   do {
            ip_past = ip;
            ip = (inp(basadr+2))&0x0020;
      }
        while (!(ip_past==0 && ip==32));
}

void check_clock(void)
{
   unsigned ip;
   ip = (inp(basadr+2)>>4)&0x0007;
   if (ip != 3)
      { do
            ip = (inp(basadr+2)>>4)&0x0007;
        while (ip != 3);
      }
   else            printf ("Too Short\n");
}

/* Wait for the EOC signal */
void wait_eoc(void)
{
   unsigned eoc;
   do
      eoc = inp(786) & 0x0008;
      while (eoc == 0);
}                  /* end of wait_eoc */

/* Get analog value from ADC */
float get_a2d_value(unsigned next_adrs,float *ptr_a2d)
{
      int code;
      float value;
      outp(786,next_adrs);    /* start ADC & set MUX adrs */
      wait_eoc();       /* wait for EOC */
      code = read_a2d();      /* read ADC */
      if (code>2047)  code = code - 4096;
      value = (float)code* *ptr_a2d;
      return value;
}

/* Limit and output the control signal */
```

APPENDIX D: CONTROLLER CODES - K222.C
SAWTOOTH DOUBLE CUT CONTROLLER

```
void limit_out_u(float *ptr_u,float *ptr_d2a,float *ptr_a2d,
            int *ptr_max,int *ptr_min)
{
      int ucode;
      unsigned lb, hb,code;
      ucode = (int)(*ptr_u * *ptr_d2a);
/*          printf("%f %i ",*ptr_u,ucode);  */
      if (ucode > *ptr_max) ucode = *ptr_max;
      if (ucode < *ptr_min) ucode = *ptr_min;
      *ptr_u = (float)(ucode * *ptr_a2d);
/*          printf("%f %i ",*ptr_u,ucode);  */
      code = (unsigned) ucode;
      if (ucode < 0) code = (unsigned)(ucode + 4096);
      lb = (code<<4) & 0x00FF;
      hb = (code>>4) & 0x00FF;
      outp(788,lb);
      outp(789,hb);
/*          printf("%u %u %u %i\n",lb,hb,code,ucode);   */
}
```

WIRING DIAGRAM - SERVO SYSTEMS



APPENDIX E: WIRING DIAGRAM / EXPERIMENTAL SETUP

COMDYNA WIRING –

FOLLOWER POSITION SIGNAL – INPUT TO AMP 7
MASTER POSITION SIGNAL – INPUT TO AMP 4
FOLLOWER SPEED SIGNAL – INPUT TO AMP 8
MASTER SPEED SIGNAL – INPUT TO A7

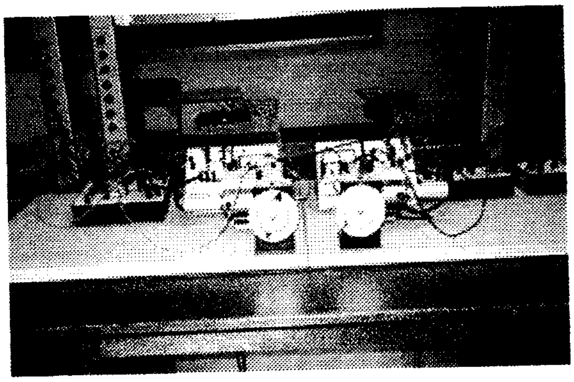
GROUND TO AGND

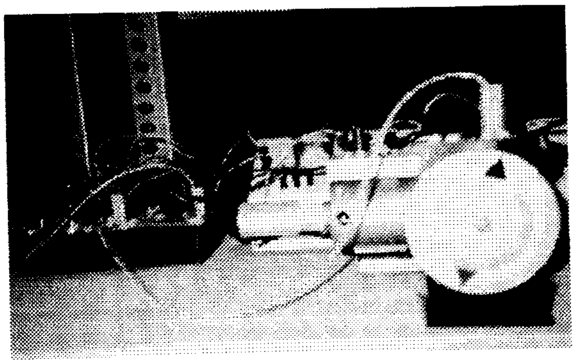
CONTROLLER OUTPUT FROM LDAC
+10 TO VDAC

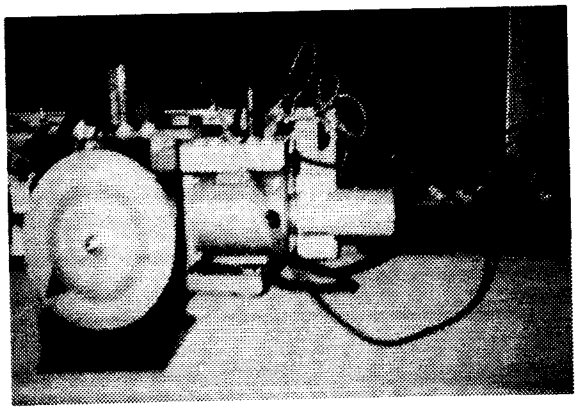
OUTPUT AMP 7 TO INPUT AMP 2
OUTPUT AMP 8 TO A5

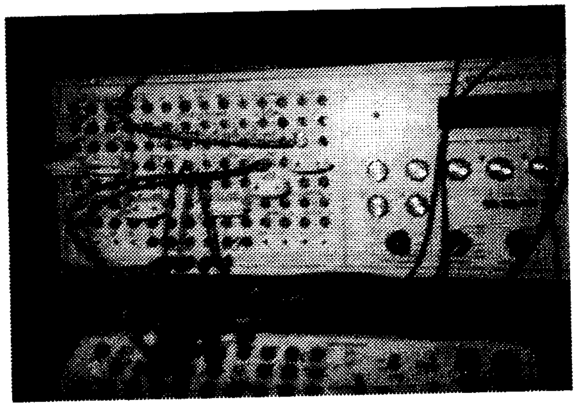
APPENDIX E: WIRING DIAGRAM / EXPERIMENTAL SETUP

EXPERIMENTAL SETUP PICTURES FROM LABORATORY


MASTER FOLLOWER SETUP IN LAB – LEFT SERVO SYSTEM – MASTER – RIGHT SERVO SYSTEM –FOLLOWER


MASTER SERVO SYSTEM WITH TWO ARROWS ON OUTPUT DISK


FOLLOWER SERVO SYSTEM WITH ONE ARROW ON OUTPUT DISK


COMDYNA A2D /D2A INTERFACE

REFERENCES / BIBLIOGRAPHY

For clarity, this section is divided into alphabetical listings of references used in terms of literature and in terms of web-sites from the World-Wide-Web

LITERATURE

1. Åström, K., & Wittenmark, B., *Computer Controlled Systems – Theory and Design, 3rd ed.*. Prentice Hall, Inc., New Jersey, 1997.

2. Barr, A.J., *The Fuzzy Logic of an Active Suspension System*. Master's Thesis, Youngstown State University, 1996.

3. Bates, S., *PID Auto Tune Control: A Practical Implementation*. Master's Thesis, Youngstown State University, 1994.

4. Feedback Instruments, Inc., *Modular Servo System MS150- Book 1: DC, Synchro, & AC Basic Experiments*. Sussex, England.

5. Feedback Instruments, Inc., *Modular Servo System MS150- Book 2: Circuit Notes and Maintenance*. Sussex, England.

6. Feedback Instruments, Inc., *Modular Servo System MS150- Book 6: Proportional, Integral, Derivative Unit PID 150Y.* Sussex, England.

7. Franklin, G., Powell, J., & Workman, M., *Digital Control of Dynamic Systems, 2nd ed.*. Addison Wesley Co., Reading, Massachusetts, 1990.

8. Harbour, R., & Phillips, C., *Feedback Control Systems, 3rd ed.*. Prentice Hall, New Jersey, 1996.

9. Mathworks, Inc., *Matlab: High Performance Numeric Computation and Visualization Software Reference Guide.* 1994.

10. Microsoft Corporation., *Microsoft Quick C: C for Yourself ver. 2.0.* 1998.

11. Nise, N. , *Control Systems Engineering, 2nd ed.*. Benjamin / Cummings Publishing Co., Inc., New York, 1995.

12. Ogata, K., *Modern Control Engineering, 2nd ed.*. Prentice Hall, New Jersey, 1990.

WORLD-WIDE-WEB

13. AZCO Corp., *Cut-to-Length Assemblies,* Author Unknown. http://www.azcocorp.com/assemb.html

14. AZCO Corp., *Rotary Knife Assemblies,* Author Unknown. http://www.azcocorp.com/rotaknf.html

15. Carnegie Mellon University, *Control Tutorials for Matlab,* Author Unknown. http://hpme12.me.cmu.edu/matlab/html/

16. Extratech Corp., *The M2400 Servo Controller,* Author Unknown. http://www.extratech.com/product2.html

17. HC Davis, Inc., *Davis Rotary Knife Cutter,* Author Unknown. http://www.hcdavis.com/FeedMaking/KnifeCutter.htm

18. Mathworks, Inc., *Digital Control of a Disk Drive Demo,* Author Unknown. http://www.mathworks.com/demos/diskdemo.html

19. Ormec, Inc., *ORMEC APPS: Rotary Knife Cutoff,* Author Unknown. http://www.ormec.com/mktdocs/rotknife.htm