An Experimental Evaluation of the Delta Operator in Digital Control

by

Brandon L. Eidson

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 9, 2010

Keywords: Digital Control, Delta Operator, Finite Word-Length

Abstract

Interest in digital control systems has been on the rise over recent decades with the ever decreasing cost and increasing performance of microprocessors and the academic advancements in control theory. Although there are many benefits to digital control, there are many challenges. This study explores one obstacle presented by finite word-length microprocessors: the limited range of numbers that can be represented. The goal is to compare its effects using two different discrete operators—the shift operator and the delta operator.

The experiment is designed in order to accommodate a second challenge in digital controls: most engineers and technicians are far more experienced with continuous controller design. Therefore, this analysis's control design takes place in continuous-time. Both shift and delta-operator-based difference equations are derived from the continuous transfer function. The ability of each competing model to perform under finite word-length conditions is evaluated through mathematical analysis, simulation, and experimentation. These demonstrations are made using a PID controller to compensate a DC-DC buck converter. It is concluded that the delta operator representation is less susceptible to the numerical limitations and shows greater performance resemblance to the original continuous compensator.

Acknowledgments

Soli Deo gloria.

Table of Contents

List of Figures

List of Tables

Chapter 1

Topic Introduction and Background

1.1 Digital Control

Academia's interest in digital control has been increasing proportionally to the increase in digital technologies. The combination of ever faster and lower-cost processors secures a place in future digital control approaches in control system applications. The implementation of complex algorithms with a single processor rather than a multitude of integrated circuits is also increasingly attractive. The ease of modification and superior imperviousness to aging add additional credibility to the overall value of digital control systems. However, the world that engineers desire to control is inherently analog and continuous, so transforming a system from a continuous, closed-loop control system (Figure 1.1) to a discrete, closed-loop control system (Figure 1.2) does not come without challenges.

The implementation of the digital control system shown in Figure 1.2 can be described more specifically for this thesis by Figure 1.3, which points out some of the main weaknesses of digital systems. Time delays are caused by the analog-to-digital converter and the computation of the new control signal. An n-bit processor's resolution is limited to increments of $(x_2 - x_1)/2^n$, where x_1 and x_2 are the maximum and minimum values of the range that is desired to be represented. Also, the processor itself can only work with numbers from 0 to $2^n - 1$. This last limitation is the main focus of this thesis.

The benefits and trade-offs associated with digital control systems versus analog control systems are well recorded and documented at the beginning of nearly every paper on the subject. There is extensive modern research attempting to improve digital control performance through complex control algorithms [1] [2], advanced digital control theory approaches [3]

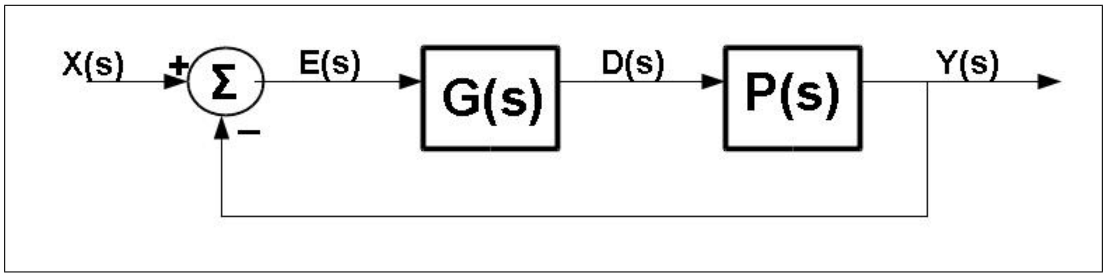

Figure 1.1: Block diagram of generic, continuous control system

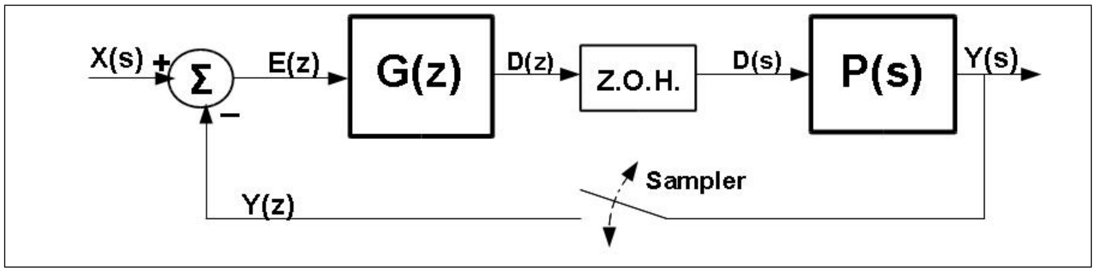

Figure 1.2: Block diagram of generic, digital control system

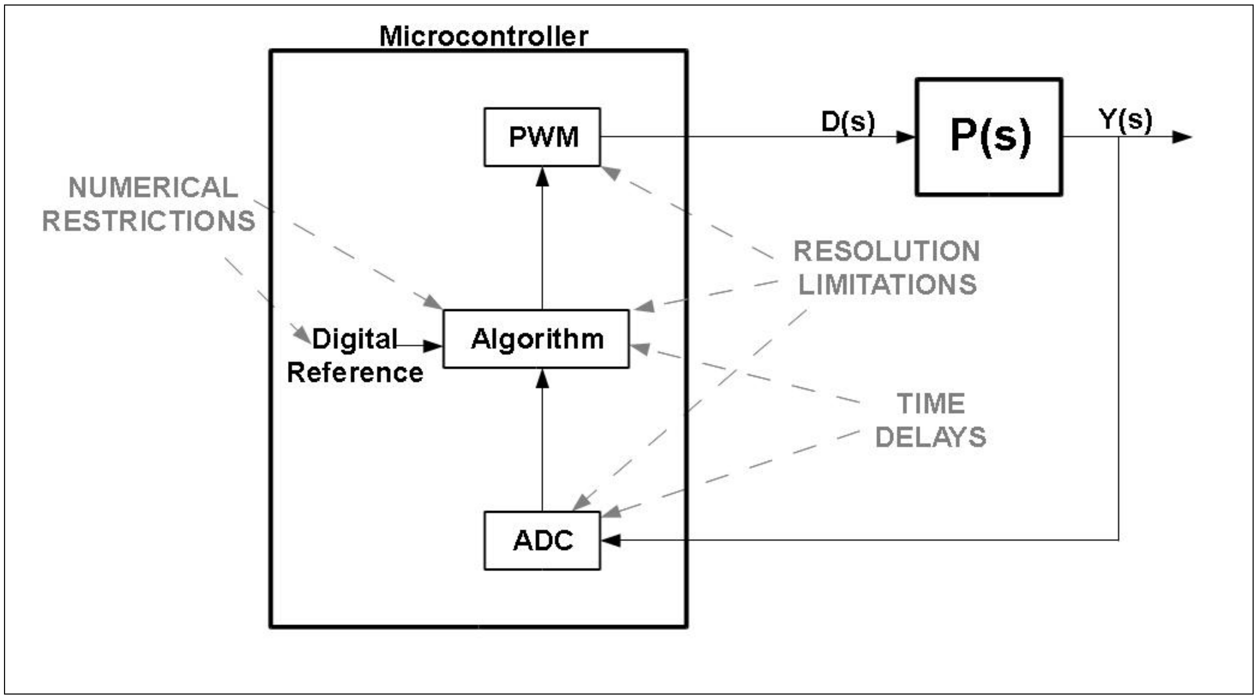

Figure 1.3: Some digital control system weaknesses

[4], reducing effects of numerical limitations [5], and overcoming analog-to-digital, digital-to-analog, and computational delays [6].

This research is rightly placed. However, it is not simply overcoming the various mathematical and physical challenges that will be necessary for the progress in digital control to be utilized. Experienced engineers and academics are far more adept at and comfortable with traditional analog control theory. There are advancements being made, though, that have potential to reduce performance and pedagogical difficulties at the same time.

1.2 The Delta Operator

The incremental difference operator (or delta operator) has been proposed as an alternative to the shift operator when representing discrete functions. A very notable work on the delta operator (and delta transform) has been produced by Richard Middleton and Graham Goodwin [7]. Their work thoroughly introduces all the major areas of digital control theory through the use of the delta operator, proposing that the study of both continuous and discrete controls can be accomplished simultaneously since there is a narrow distinction between the two when considering the delta operator for discrete parameterizations. This thesis introduces some of these basic concepts of the delta operator in Chapter 2, but for a thorough understanding, Middleton and Goodwin's textbook should be referenced [7].

Others have explored the potential benefits of the delta operator over the shift operator, especially when considering the effects of finite word-length. Chen, Wu, Istepanian and Chu have shown that the closed-loop stability margin of delta operator parameterizations are better than the shift operator approach [8] [9]. Li and Gevers demonstrated that the delta operator representations of state-variable models are less sensitive to coefficient errors in the state-space matrices due to finite word-length [10]. Before these, Middleton and Goodwin argued for the delta operator's ability to:

1. better represent controller coefficients,

2. reduce rounding error, and

3. facilitate more straightforward design by pole assignment [11].

The delta operator significantly closes the gap between continuous and discrete control theory. In addition to potential performance improvement over the z-Transform, the conceptual and mathematical similarities between the δ-Domain and the s-Domain can make understanding discrete controls far easier—especially for someone already familiar with analog controls. Some examples of these connections are introduced later in this text, but the main focus will be on the delta and shift operators' coefficient representation restrictions presented by finite word-length.

1.3 The Focus of this Thesis

The proposed performance benefit of the delta operator over the shift operator under finite word-length situations could have many advantages in industry. For instance, consider that most of today's control engineers and technicians are experienced with continuous control design. Also, even though the use of floating-point processors is on the rise, many applications still prefer fixed-point implementations due to their reduced cost and power consumption and increased speed and simplicity. There are similar benefits associated with reduced word-length. Combining these realities reveals a benefit in having delta-operator-based control algorithms.

This is, of course, dependent on the improved effectiveness of the delta operator's performance and continuous-like behavior in practice over the shift operator when transforming from an analog to a digital controller representation. This thesis explores one facet of the proposed benefits: the numerical characteristics of the controller coefficients due to finite word-length. An investigation of the mathematical traits of a shift-operator-based (q-based) difference equation and a delta-operator-based (δ-based) difference equation, both having been derived from the same s-Domain transfer function, is conducted. The results are simulated and tested using an 8-bit programmable interface controller (PIC$^{\text{TM}}$) to implement a proportional, integral, derivative (PID) compensator to control a DC-DC buck converter. Other digital control weaknesses are minimized and/or held constant so as to explore the finite word-length characteristics.

Chapter 2 provides an overview of the delta operator. Chapter 3 conducts the mathematical analysis of the shift and delta operators' resultant difference equations and derives the corresponding PID difference equations. Chapter 4 explains the hardware and software utilized to implement an experiment of the results from Chapter 3. The basic concepts of the representative plant, a buck converter, are introduced, as well as the appropriateness of an 18F series PIC$^{\text{TM}}$ as the microprocessor for this experiment. Chapter 4 concludes with an explanation of the software algorithms. The system modeling and simulation designs are discussed in Chapter 5, and the simulation and experimental results are recorded in Chapter 6. Chapter 7 presents the thesis conclusions and provides recommendations for future work.

Chapter 2
Overview of the Delta Operator

Because the equivalent continuous-time operator is d/dt, one might suspect that it would not bear much resemblance to the discrete shift operator, q. This is exactly what has been observed by academia and industry for decades. However, if a discrete-time operator equivalent to a derivative is used, an increased correspondence would be expected. This premise is the historic motivation behind the development of the delta operator.

2.1 Definition of Delta Operator

The incremental difference operator (or delta operator) is typically defined as:

$$\delta = \frac{q-1}{T_s}$$

where T_s is the sampling period and q is the classic forward-shift operator, defined as:

$$qx_k = x_{k+1}$$

making

$$\delta x_k = \frac{x_{k+1} - x_k}{T_s}$$

where x_k is a state at the k-th sample. This expression demonstrates the correlation the delta operator makes between discrete and continuous time. Notice that the delta operator acts like an approximation of a derivative as T_s approaches 0 (the sampling rate, f_s, approaches ∞). So, at an infinite sampling rate, δ operating on x_k would be identical to d/dt operating on x_k!

Now, for most practical purposes, it is necessary to write the delta operator in a causal form. The causal (backward) delta operator is defined as:

$$\delta^{-1} = \frac{1 - q^{-1}}{T_s} \tag{2.1.1}$$

where q^{-1} is the causal (backward) shift operator:

$$q^{-1}x_k = x_{k-1}$$

Therefore,

$$\delta^{-1}x_k = \frac{x_k - x_{k-1}}{T_s} \tag{2.1.2}$$

2.2 Higher-Order Representations

In most applications, higher-order operators are necessary to implement a control system.

2.2.1 Second-Order, Causal Delta Operator

The second-order operator is:

$$\begin{aligned}\delta^{-2}x_k &= \delta^{-1}(\delta^{-1}x_k) \\ &= \delta^{-1}(\frac{x_k - x_{k-1}}{T_s})\end{aligned}$$

The second-order delta operator can be viewed as the difference of differences, or the approximation of a second-order derivative as T_s approaches 0, in the following form:

$$\delta^{-2}x_k = \frac{\frac{x_k - x_{k-1}}{T_s} - \frac{x_{k-1} - x_{k-2}}{T_s}}{T_s} \tag{2.2.1}$$

For implementing in a real-time program, the following form can be derived from observing equations (2.1.2) and (2.2.1):

$$\delta^{-2}x_k = \frac{\delta^{-1}x_k - \delta^{-1}x_{k-1}}{T_s} \tag{2.2.2}$$

2.2.2 Third-Order, Causal Delta Operator

Applying a third delta operator will help reveal a pattern for the causal delta operator, as shown by the following:

$$\delta^{-3}x_k = \delta^{-1}(\delta^{-2}x_k)$$

$$= \delta^{-1}\left(\frac{\frac{x_k - x_{k-1}}{T_s} - \frac{x_{k-1} - x_{k-2}}{T_s}}{T_s}\right)$$

The third-order derivative approximation for the delta operator is seen here:

$$\delta^{-3}x_k = \frac{\frac{\left(\frac{x_k - x_{k-1}}{T_s}\right) - \left(\frac{x_{k-1} - x_{k-2}}{T_s}\right)}{T_s} - \frac{\left(\frac{x_{k-1} - x_{k-2}}{T_s}\right) - \left(\frac{x_{k-2} - x_{k-3}}{T_s}\right)}{T_s}}{T_s} \tag{2.2.3}$$

If equations (2.1.2), (2.2.2) and (2.2.3) are examined closely, more programmable-friendly forms can be found, such as:

$$\delta^{-3}x_k = \frac{\left(\frac{\delta^{-1}x_k - \delta^{-1}x_{k-1}}{T_s}\right) - \left(\frac{\delta^{-1}x_{k-1} - \delta^{-1}x_{k-2}}{T_s}\right)}{T_s} \tag{2.2.4}$$

or

$$\delta^{-3}x_k = \frac{\delta^{-2}x_k - \delta^{-2}x_{k-1}}{T_s} \tag{2.2.5}$$

2.2.3 In General

From examining equations (2.1.2), (2.2.2), and (2.2.5), a general pattern is observed so that the following definition can be made:

$$\delta^{-n}x_k = \frac{\delta^{-(n-1)}x_k - \delta^{-(n-1)}x_{k-1}}{T_s} \tag{2.2.6}$$

where n represents any integer. This result is conceptually consistent as the delta operator is thought of as a derivative approximation. Higher-order derivatives simply represent changes in changes (e.g., acceleration being the change in velocity, which is the change in position).

2.3 Stability Region Parallels

One additional illustration will serve to introduce the shared relationship and benefit the delta operator provides in relation to continuous control systems. The stability regions of their root locus plots will be examined. These facts are simply recorded; the reader can investigate [7] for further explanation.

All the poles of the Laplace transform model of stable continuous-time systems lie strictly in the left-half of the s-plane (Figure 2.1). The eigenvalues of a system matrix in linear state variable form are found in the same region.

The shift operator's stability region is significantly different, consisting of only the unit circle (Figure 2.2).

When the stability region of the delta operator (Figure 2.3) is observed, an interesting relationship with the continuous-time diagram is seen. The circle's origin is shifted to the point $-1/T_s + j0$. The radius of the stability region circle is also $1/T_s$. As T_s approaches 0, the stability region of δ and d/dt become identical!

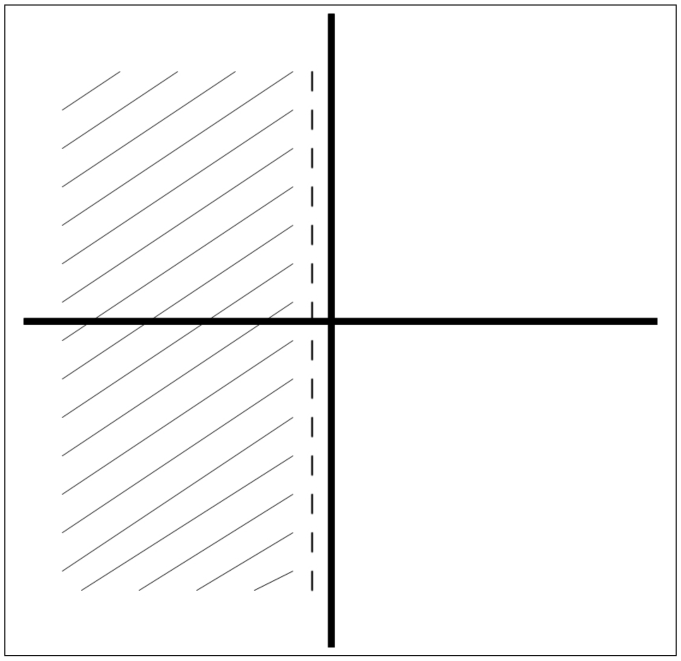

Figure 2.1: Stability region of d/dt operator

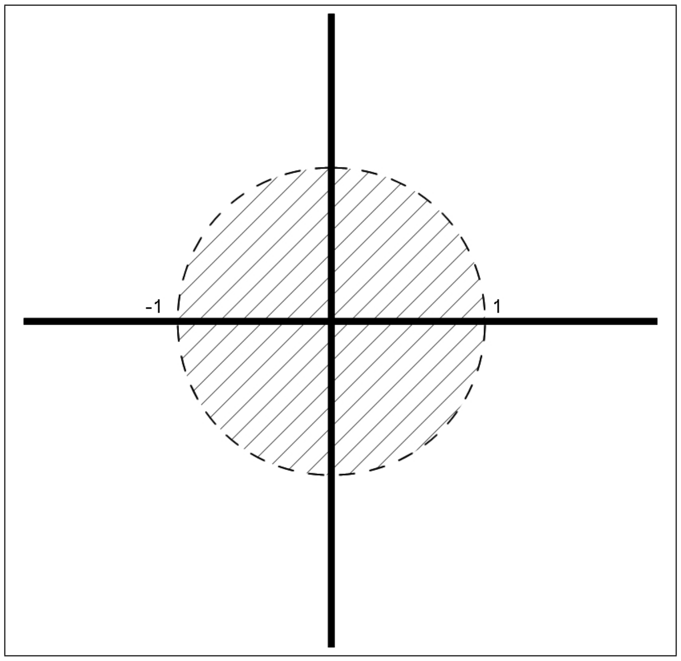

Figure 2.2: Stability region of shift operator

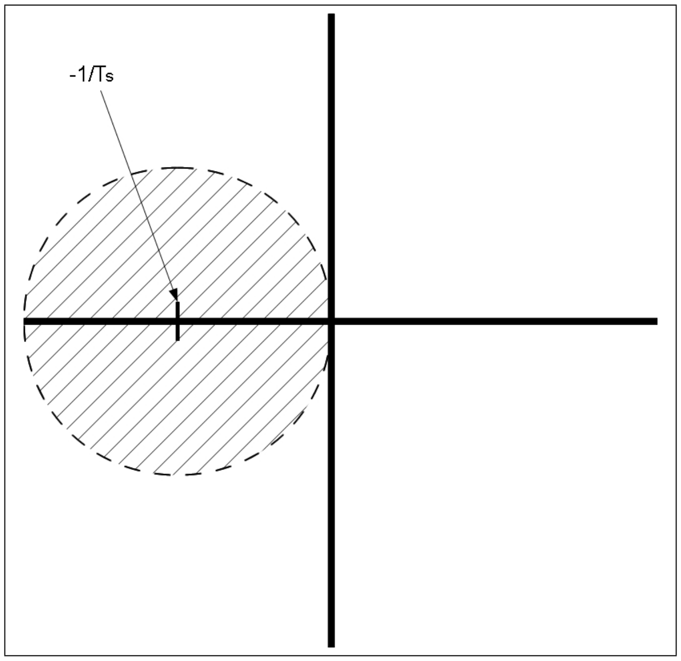

Figure 2.3: Stability region of delta operator

Chapter 3

Mathematical Demonstration of Coefficient Constraints on Discrete Models

In order to implement a digital controller in software, the transfer function must be written as a causal difference equation. The first section of this chapter finds the general difference equations from a generic continuous-time transfer function. Although the solutions are somewhat complex, one can clearly see the numeric benefits of the delta operator.

In the second section, two difference equations are obtained for a classic PID controller. The observed numeric benefits in general are realized in this specific example.

3.1 General Mathematical Analysis

3.1.1 General Transfer Function Conversion to Shift Operator Representation

A transfer function, $G(s)$, is defined as the relation between the input and output of a general system:

$$G(s) = \frac{Y(s)}{X(s)}$$

and its general form is:

$$G(s) = \frac{a_k s^k + a_{k-1} s^{k-1} + ... + a_1 s + a_0}{b_j s^j + b_{j-1} s^{j-1} + ... + b_1 s + b_0}$$

In order to discretize the transfer function, most advanced controls engineers would use the bilinear (or Tustin or Trapezoidal) approximation. In order to reduce the complexity of the math, Euler's backward difference approximation is utilized:

$$s \approx \frac{1 - z^{-1}}{T_s}$$

where T_s is the sampling period. Making the above substitution yields:

$$G(z) = \frac{\frac{a_k}{T_s^k}(1-z^{-1})^k + \frac{a_{k-1}}{T_s^{k-1}}(1-z^{-1})^{k-1} + ... + \frac{a_1}{T_s}(1-z^{-1}) + a_0}{\frac{b_j}{T_s^j}(1-z^{-1})^j + \frac{b_{j-1}}{T_s^{j-1}}(1-z^{-1})^{j-1} + ... + \frac{b_1}{T_s}(1-z^{-1}) + b_0}$$

and after finding common denominators of the numerator and denominator the result can be written as:

$$G(z) = \Big(\frac{T_s^j}{T_s^k}\Big)\frac{a_k(1-z^{-1})^k + a_{k-1}T_s(1-z^{-1})^{k-1} + ... + a_1T_s^{k-1}(1-z^{-1}) + a_0T_s^k}{b_j(1-z^{-1})^j + b_{j-1}T_s(1-z^{-1})^{j-1} + ... + b_1T_s^{j-1}(1-z^{-1}) + b_0T_s^j}$$

Since this thesis's experiment is controlling a buck converter, it will be helpful to define the discrete transfer function as:

$$G(z) = \frac{D(z)}{E(z)}$$

where $d[n]$ and $e[n]$ represent the duty cycle and the error signal at time nT_s, respectively, where $n = 1, 2, 3....$ So, the difference equation can be found by cross multiplying:

$$\begin{gathered} D(z)[b_j(1-z^{-1})^j + b_{j-1}T_s(1-z^{-1})^{j-1} + ... + b_1T_s^{j-1}(1-z^{-1}) + b_0T_s^j] = \\ E(z)\Big(\frac{T_s^j}{T_s^k}\Big)[a_k(1-z^{-1})^k + a_{k-1}T_s(1-z^{-1})^{k-1} + ... + a_1T_s^{k-1}(1-z^{-1}) + a_0T_s^k] \end{gathered}$$

and substituting in the equivalent time-domain components:

$$\begin{gathered} d[n][b_j(1-q^{-1})^j + b_{j-1}T_s(1-q^{-1})^{j-1} + ... + b_1T_s^{j-1}(1-q^{-1}) + b_0T_s^j] = \\ e[n]\Big(\frac{T_s^j}{T_s^k}\Big)[a_k(1-q^{-1})^k + a_{k-1}T_s(1-q^{-1})^{k-1} + ... + a_1T_s^{k-1}(1-q^{-1}) + a_0T_s^k] \end{gathered} \tag{3.1.1}$$

This can be written more concisely by utilizing the binomial theorem:

$$(x+y)^k = \sum_{i=0}^{k}\binom{k}{i}x^{k-i}y^i$$

where

$$\binom{k}{i} = \frac{k!}{i!(k-i)!}$$

A general expression emerges such that

$$(x+y)^k + (x+y)^{k-1} + \ldots(x+y) + 1 = \sum_{m=0}^{k}\left[\sum_{i=0}^{k-m}\binom{k-m}{i}x^{(k-m)-i}y^i\right] \tag{3.1.2}$$

By applying equation (3.1.2) to equation (3.1.1), the difference equation becomes:

$$\begin{aligned} d[n]\sum_{m=0}^{j}\left[b_{j-m}T_s^m\sum_{i=0}^{j-m}\binom{j-m}{i}(-q)^{-i}\right] &= \\ e[n]\left(\frac{T_s^j}{T_s^k}\right)\sum_{m=0}^{k}\left[a_{k-m}T_s^m\sum_{i=0}^{k-m}\binom{k-m}{i}(-q)^{-i}\right] \end{aligned} \tag{3.1.3}$$

The final, causal difference equation for $d[n]$ based on the shift operator can be written as:

$$\begin{aligned} d[n] = \left(\sum_{m=0}^{j} b_{j-m}T_s^m\right)^{-1}\Bigg[-d[n]\sum_{m=0}^{j}\left[b_{j-m}T_s^m\sum_{i=1}^{j-m}\binom{j-m}{i}(-q)^{-i}\right] + \\ e[n]\left(\frac{T_s^j}{T_s^k}\right)\sum_{m=0}^{k}\left[a_{k-m}T_s^m\sum_{i=0}^{k-m}\binom{k-m}{i}(-q)^{-i}\right]\Bigg] \end{aligned} \tag{3.1.4}$$

Although this result is complex, what is important to understand is not. Observe in particular the scaling term T_s^j/T_s^k. Unless $k = j$, every coefficient in the difference equation multiplied by every shifted version of $e[n]$ will have a term scaled by some power of T_s. This same possibility exists for many of the coefficients even if $k = j$. Because a sampling period is typically in ranges of 10^{-5} to 10^{-6} seconds, the coefficients will vary over many orders of magnitude.

There are important implications for this scaling characteristic in finite word-length mircocontrollers. Specifically, the maximum value a microcontroller can represent is $2^\nu - 1$, where ν is the number of bits. For example, an 8-bit microcontroller can only represent values up to 255. The max value due to this hardware-related constraint has the potential to be orders of magnitude smaller than the values required if the sampling period is in a range typical for digital control systems.

3.1.2 Shift Operator Representation to Delta Operator Representation

By rearranging equation (2.1.1), one obtains

$$q^{-1} = 1 - T_s\delta^{-1}$$

and equation (3.1.1) can be rewritten as:

$$d[n][b_j(T_s\delta^{-1})^j + b_{j-1}T_s(T_s\delta^{-1})^{j-1} + ... + b_1T_s^{j-1}(T_s\delta^{-1}) + b_0T_s^j] =$$
$$e[n]\Big(\frac{T_s^j}{T_s^k}\Big)[a_k(T_s\delta^{-1})^k + a_{k-1}T_s(T_s\delta^{-1})^{k-1} + ... + a_1T_s^{k-1}(T_s\delta^{-1}) + a_0T_s^k]$$

Therefore,

$$d[n]\sum_{m=0}^{j} b_mT_s^{j-m}T_s^m\delta^{-m} = e[n]\Big(\frac{T_s^j}{T_s^k}\Big)\sum_{m=0}^{k} a_mT_s^{k-m}T_s^m\delta^{-m}$$

Combining the T_s's and moving them in front of the summations yields:

$$d[n]T_s^j\sum_{m=0}^{j} b_m\delta^{-m} = e[n]\Big(\frac{T_s^j}{T_s^k}\Big)T_s^k\sum_{m=0}^{k} a_m\delta^{-m}$$

This result shows that the sampling periods can be canceled from the equations:

$$d[n]\sum_{m=0}^{j} b_m\delta^{-m} = e[n]\sum_{m=0}^{k} a_m\delta^{-m}$$

If $b_0 \neq 0$, the final, causal difference equation for $d[n]$ based on the delta operator can be written as:

$$d[n] = b_0^{-1}\Big[-d[n]\sum_{m=1}^{j} b_m\delta^{-m} + e[n]\sum_{m=0}^{k} a_m\delta^{-m}\Big] \tag{3.1.5}$$

Because the sampling periods canceled, the scaling issue associated with the coefficients of the shift-operator-based difference equations is no longer as severe!

Each delta operator term, though, factors in dividing by the sampling period. So, even though the coefficients of the delta-operator-based difference equation are not affected by the sampling period in the same way as the shift operator difference equation, the sampling period cannot be ignored altogether.

Also, if $b_0 = 0$, some change in the duty cycle will have to be solved for, rather than the new duty cycle itself, and this will introduce the sampling period into the coefficients. This will be seen with the PID controller's difference equation. However, by using the delta operator approach, the sampling period will be easily absorbed into the design of the coefficients as will be demonstrated later.

3.2 Discrete Representations of a PID Controller

To demonstrate the superior numerical aspects of the delta operator, an experiment is conducted to compare shift operator and delta operator control of a buck converter. A PID compensator is designed in its analog form, then mathematically represented by two different difference equations—one shift-operator-based and the other delta-operator-based. In this section, the two competing difference equations are derived.

3.2.1 Z-Transforming the PID

The PID controller is one of the most common controllers used in industry process system applications. A standard academic practice is to use the PID or compare to the PID when testing new control techniques or aspects; see examples in [12, 13, 14, 15]. The Laplace Transform form of the classical PID compensator is defined as:

$$\frac{D(s)}{E(s)} = K_P + sK_D + \frac{K_I}{s} \tag{3.2.1}$$

As before, Euler's backward difference approximation is utilized to discretize the PID transfer function:

$$s \approx \frac{1 - z^{-1}}{T_s}$$

Therefore,

$$\frac{D(z)}{E(z)} = K_P + \frac{K_D(1 - z^{-1})}{T_s} + \frac{K_I T_s}{1 - z^{-1}}$$

This simplifies to

$$\frac{D(z)}{E(z)} = \frac{K_P T_s(1 - z^{-1}) + K_D(1 - 2z^{-1} + z^{-2}) + K_I T_s^2}{T_s(1 - z^{-1})}$$

and

$$\frac{D(z)}{E(z)} = \frac{(\frac{K_D}{T_s})z^{-2} + (-K_p - \frac{2K_D}{T_s})z^{-1} + (K_P + \frac{K_D}{T_s} + T_S K_I)}{(1 - z^{-1})} \tag{3.2.2}$$

3.2.2 The Shift Operator Difference Equation

To find a difference equation, cross multiply both sides of equation (3.2.2):

$$D(z)(1 - z^{-1}) = E(z)(\frac{K_D}{T_s})z^{-2} + (-K_p - \frac{2K_D}{T_s})z^{-1} + (K_P + \frac{K_D}{T_s} + T_S K_I)$$

Then replace z^{-1} with q^{-1} as follows:

$$d_n(1 - q^{-1}) = e_n[(\frac{K_D}{T_s})q^{-2} + (-K_p - \frac{2K_D}{T_s})q^{-1} + (K_P + \frac{K_D}{T_s} + T_S K_I)]$$

A causal difference equation is obtained to calculate the change in duty cycle. The resultant equation for Δd is:

$$\Delta d = (\frac{K_D}{T_s})e_n q^{-2} + (-K_p - \frac{2K_D}{T_s})e_n q^{-1} + (K_P + \frac{K_D}{T_s} + T_s K_I)e_n \tag{3.2.3}$$

which is equivalent to:

$$\Delta d = (\frac{K_D}{T_s})e_{n-2} + (-K_p - \frac{2K_D}{T_s})e_{n-1} + (K_P + \frac{K_D}{T_s} + T_s K_I)e_n \tag{3.2.4}$$

Finally, equation (3.2.4) will be written as:

$$\Delta d = \alpha_2 e_{n-2} + \alpha_1 e_{n-1} + \alpha_0 e_n \tag{3.2.5}$$

where

$$\alpha_2 = (\frac{K_D}{T_s}) \tag{3.2.6}$$

$$\alpha_1 = (-K_p - \frac{2K_D}{T_s}) \tag{3.2.7}$$

$$\alpha_0 = (K_P + \frac{K_D}{T_s} + T_s K_I) \tag{3.2.8}$$

Equations (3.2.5) through (3.2.8) are the ones that will be used for programming the shift operator's difference equation algorithm in this experiment. After the continuous coefficients, K_P, K_D, and K_I, have been determined, the α coefficients will be calculated.

Notice, however, that all three of these coefficients are substantially larger than the continuous coefficients because of the additions and, more importantly, the scaling by $1/T_s$ on the K_D term. For a ν-bit microcontroller, if

$$\frac{K_D}{T_s} > 2^{\nu} - 1$$

the coefficients cannot be properly represented without truncation (or overflow), much less the results of the multiplications! Although truncation historically refers to removing LSBs, this thesis will use it to refer to removing MSBs (e.g., in an 8-bit microcontroller, the value 290 will have to be truncated to 255). The plot in Figure 3.1 illustrates the possible

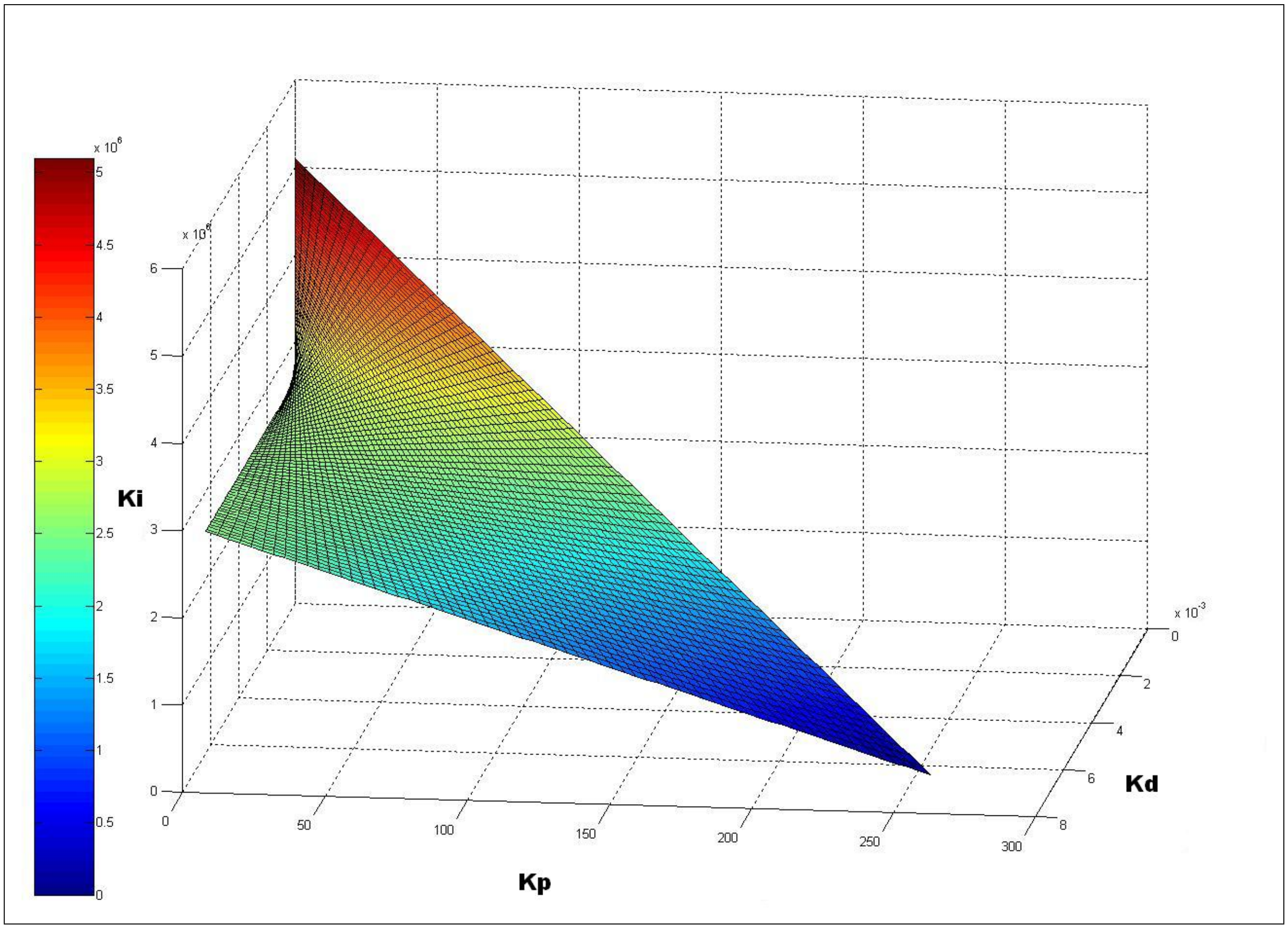

Figure 3.1: Continuous PID coefficients that will not result in truncation using shift operator parameterization

combinations of continuous PID gains that can be selected that will not result in the shift operator's difference equation coefficients being truncated (or overflowing).

3.2.3 The Delta Operator Difference Equation

A different representation of the difference equation presented by equation (3.2.3) can be obtained by making the following substitutions:

$$q^{-1} = 1 - T_s\delta^{-1}$$

and

$$q^{-2} = 1 - 2T_s\delta^{-1} + T_s^2\delta^{-2}$$

The result is as follows:

$$\Delta d = (\frac{K_D}{T_s})T_s^2 e_n\delta^{-2} + (K_p + \frac{2K_D}{T_s} - \frac{2K_D}{T_s})T_s e_n\delta^{-1} + \\ (K_P - \frac{2K_D}{T_s} + T_s K_I - K_P + \frac{K_D}{T_s} + \frac{K_D}{T_s})e_n$$

which simplifies to

$$\Delta d = (K_D T_s)e_n\delta^{-2} + (K_p T_s)e_n\delta^{-1} + (K_I T_s)e_n \tag{3.2.9}$$

As concluded from Section 3.1.2, because there is no constant in the denominator of the original continuous transfer function (i.e., $b_0 = 0$ in (3.1.5)), the sampling period appeared in the coefficients of the delta-operator-based difference equation. However, since it is not possible to divide by T_s in the microcontroller chosen for the experiment (explained in detail later), the sampling period will need to be absorbed into the coefficients. If the definition of the delta operator is substituted into equation (3.2.9),

$$\Delta d = (K_D T_s)\frac{\left(\frac{e_n - e_{n-1}}{T_s}\right) - \left(\frac{e_{n-1} - e_{n-2}}{T_s}\right)}{T_s} + (K_p T_s)\left(\frac{e_n - e_{n-1}}{T_s}\right) + (K_I T_s)e_n$$

This can be rewritten as

$$\Delta d = D[(e_n - e_{n-1}) - (e_{n-1} - e_{n-2})] + P[e_n - e_{n-1}] + Ie_n \tag{3.2.10}$$

where

$$D = \frac{K_D}{T_s} \tag{3.2.11}$$

$$P = K_P \tag{3.2.12}$$

$$I = T_s K_I \tag{3.2.13}$$

Equations (3.2.10) through (3.2.13) are used in the delta-operator-based algorithm. In this form, the "division" will occur at the same time as the coefficient multiplication. Also, notice that now only one term contains K_D/T_s. This way, if K_D is designed so that it must be truncated after being scaled by $1/T_s$, only one coefficient will be thrown off, as opposed to all three coefficients used in equation (3.2.4). The possible continuous PID coefficients that can be used without truncation occurring on the delta-operator-based coefficients are shown in Figure 3.2. These are compared with the shift operator's limitations in Figures 3.3 and 3.4.

Care must be taken when designing K_I because fixed-point processors do not handle decimals. If division is required, it must be by powers of 2. This consideration is described in detail in the hardware section of the experimental implementation chapter.

For now, it is important to note that the continuous PID compensator gains, K_D, K_P, and K_I, are designed so that conveniently-programmable and untruncated difference equation coefficients are much easier and less limited with the delta-operator-based difference equations.

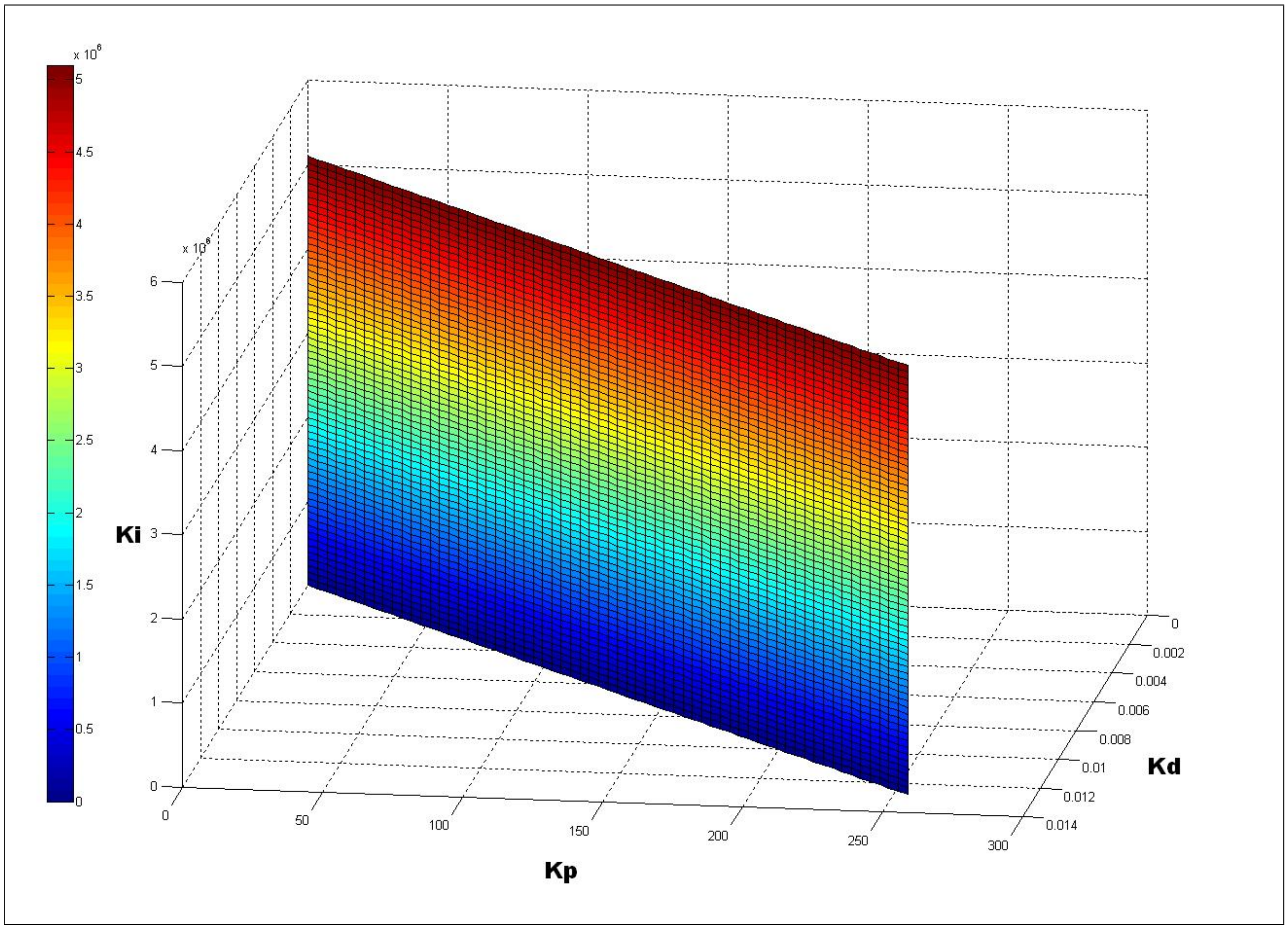

Figure 3.2: Continuous PID coefficients that will not result in truncation using delta operator parameterization

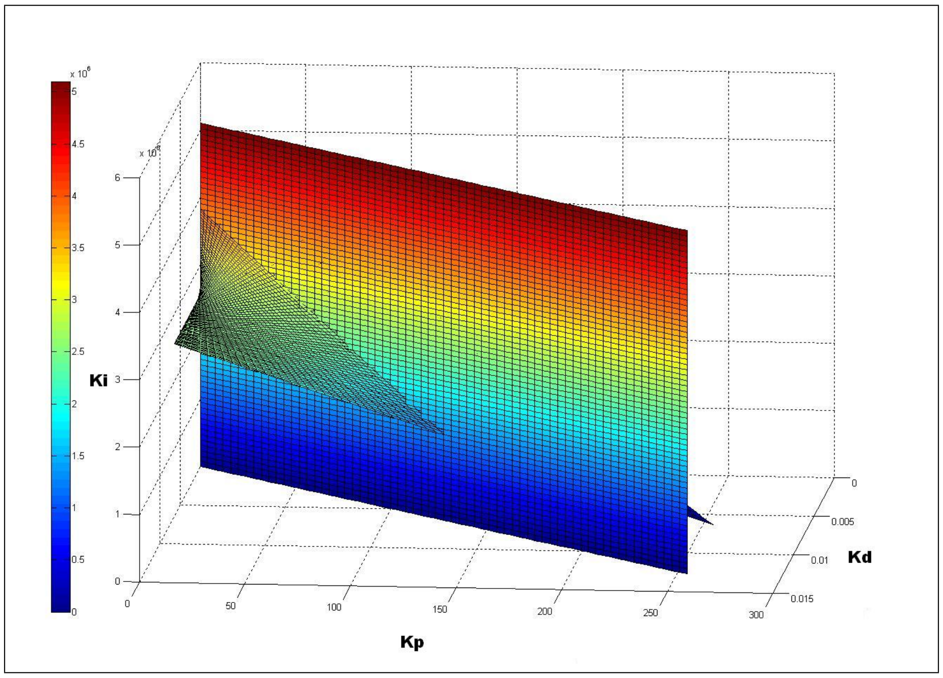

Figure 3.3: Comparison of possible continuous PID coefficients without truncation occurring in the shift and delta operators difference equations, View 1

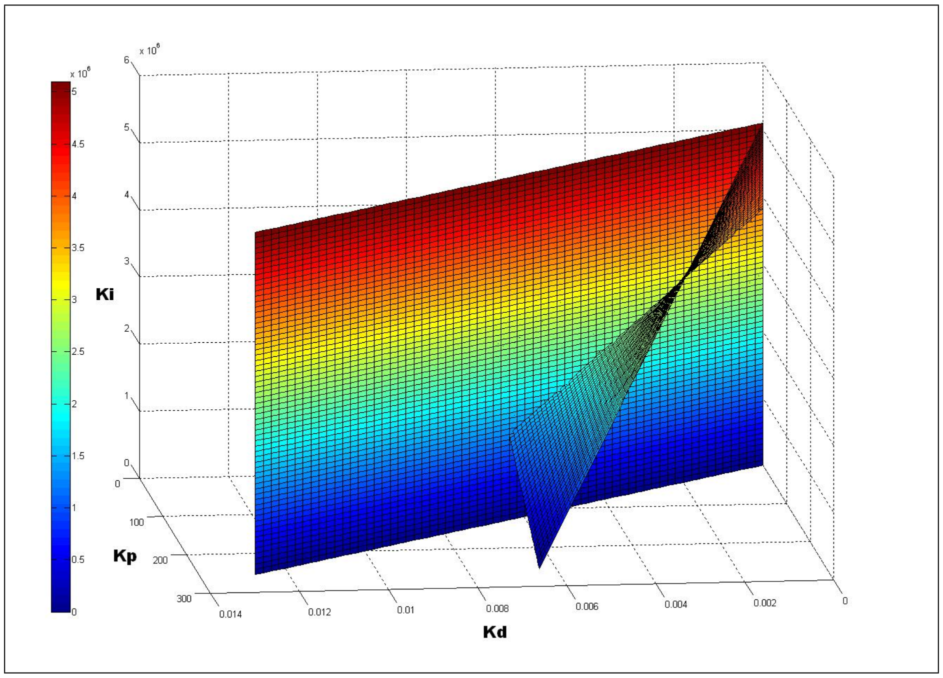

Figure 3.4: Comparison of possible continuous PID coefficients without truncation occurring in the shift and delta operators difference equations, View 2

Chapter 4

Experiment Implementation

In order to test the results from the mathematical demonstration, an experiment is designed and implemented to compare the performance of a shift-operator-based controller with a delta-operator-based controller. Both are implemented based on a continuously-designed PID controller. Most of the schematics and simulation diagrams (Chapter 5) include the capabilities of performing step-changes in load. These are not performed in this thesis; only the initial transients will be observed.

4.1 Hardware and System Configuration

A DC-DC buck converter is provided to serve as a plant process, and an 18F1220 PIC$^{\text{TM}}$ microcontroller is selected as the equipment to create the compensated, closed-loop system. The buck converter and controller are configured as shown in Figure 4.1. The performance characteristics are described in the following subsections.

4.1.1 The Plant: DC-DC Buck Converter

DC-DC converters are electronic devices used to regulate one DC voltage to another. Buck converters are DC-DC converters specifically designed to step down an input voltage and are common control problems. A helpful way to introduce these devices is to derive their ideal gain and efficiency. A typical schematic of an ideal buck converter is illustrated in Figure 4.2.

Although buck converters can operate in a discontinuous conduction mode, this thesis analysis and experiment will be limited to operation in continuous conduction mode only (i.e., the inductor current is assumed to always be > 0). Therefore, for this thesis's purposes,

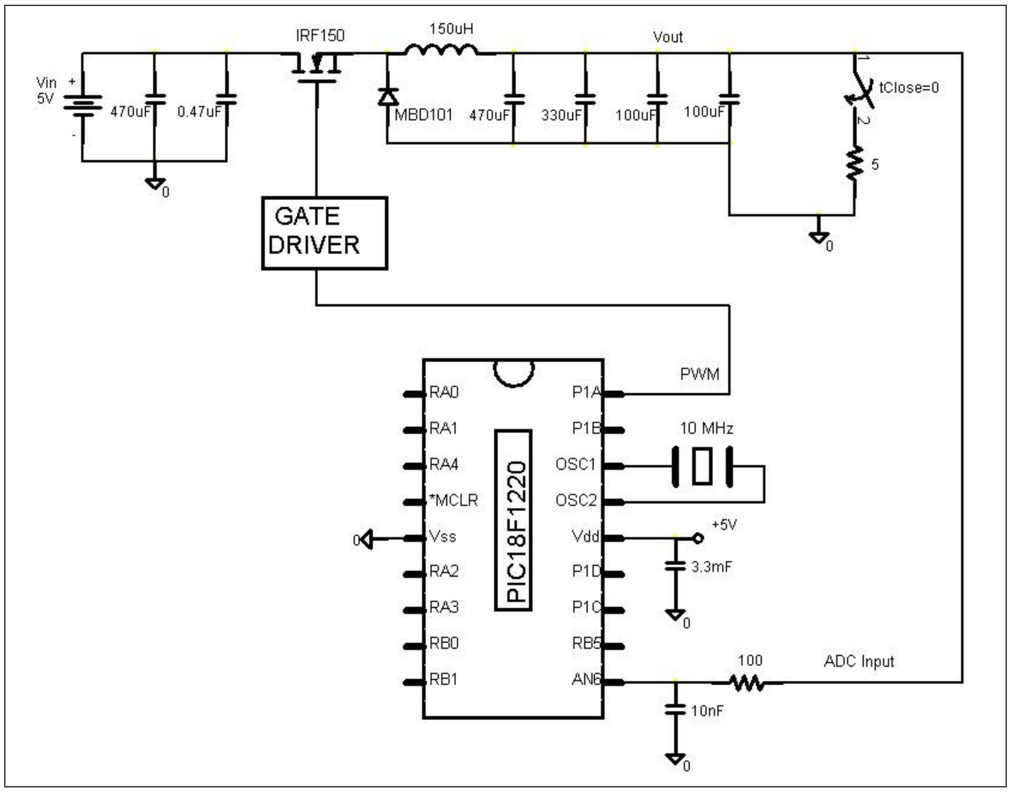

Figure 4.1: Schematic for digitally controlled buck converter

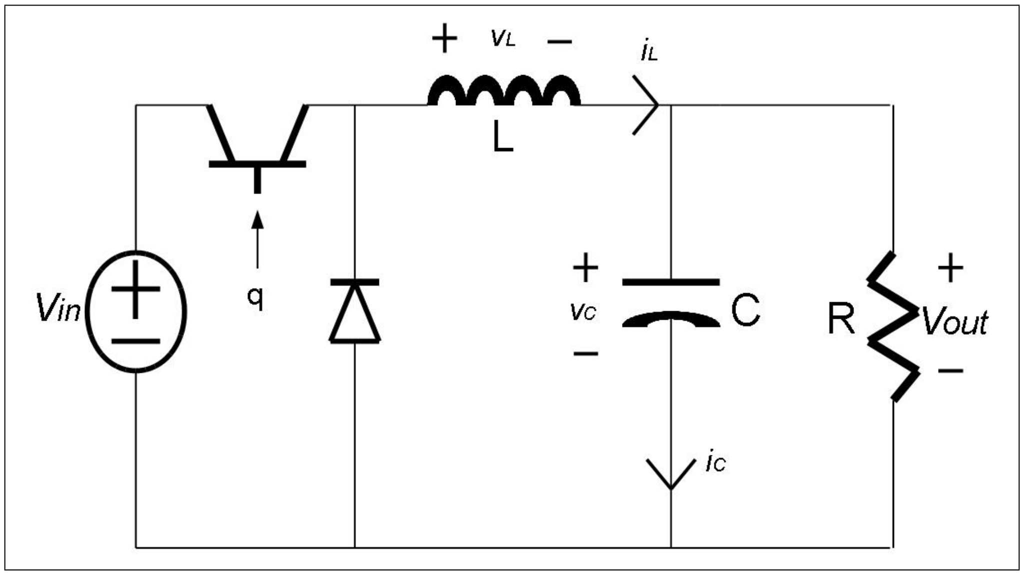

Figure 4.2: An ideal buck converter

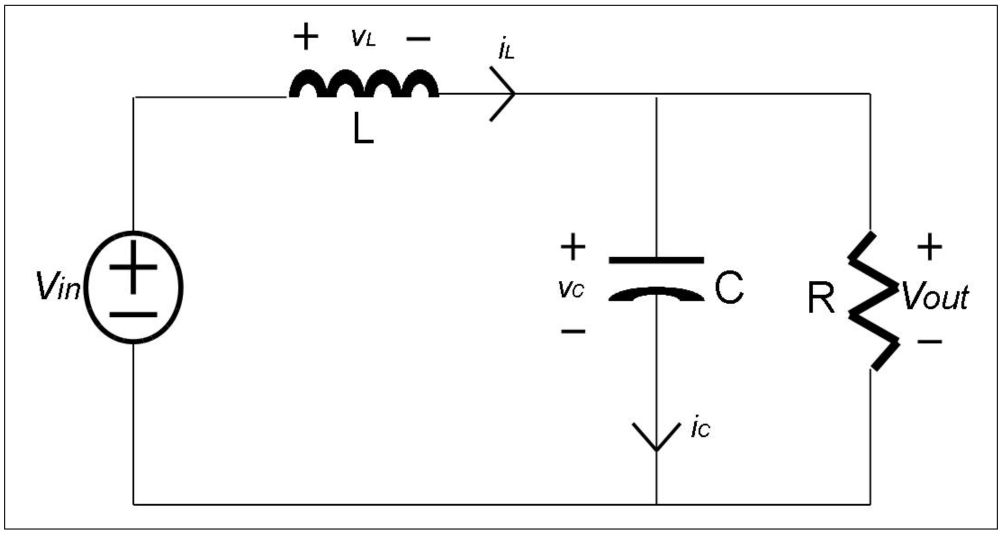

Figure 4.3: Equivalent ideal buck converter with switch closed

the buck converter will be operating in only two states: switch closed ("ON") and switch opened ("OFF"). The signal q is a pulse-width modulated (PWM) signal that varies the power transistor between saturation mode (switch closed or "ON") and cutoff mode (switch open or "OFF"). The switching period of q is denoted by T_{SW}, and the ratio of its "ON" time verses its switching period is defined as the duty cycle, D:

$$D = \frac{T_{ON}}{T_{SW}} \tag{4.1.1}$$

Notice that $0 \leq D \leq 1$.

When the power transistor is conducting, $0 \leq t \leq T_{ON}$, the buck converter is modeled by Figure 4.3. In this case, the transistor acts as a closed switch (a short circuit), and the diode is blocking, acting as an open circuit. When the power transistor enters cutoff mode, $T_{ON} < t \leq T_{SW}$, it begins to act as an open circuit, and the diode begins to conduct. This is represented by Figure 4.4.

To easily analyze the relationship between the input and output voltage, the equal-area criterion is utilized with respect to the inductor voltage, v_L [16]. Using Kirchhoff's Voltage Law around the outer loop in Figure 4.3 reveals that while the transistor switch is conducting,

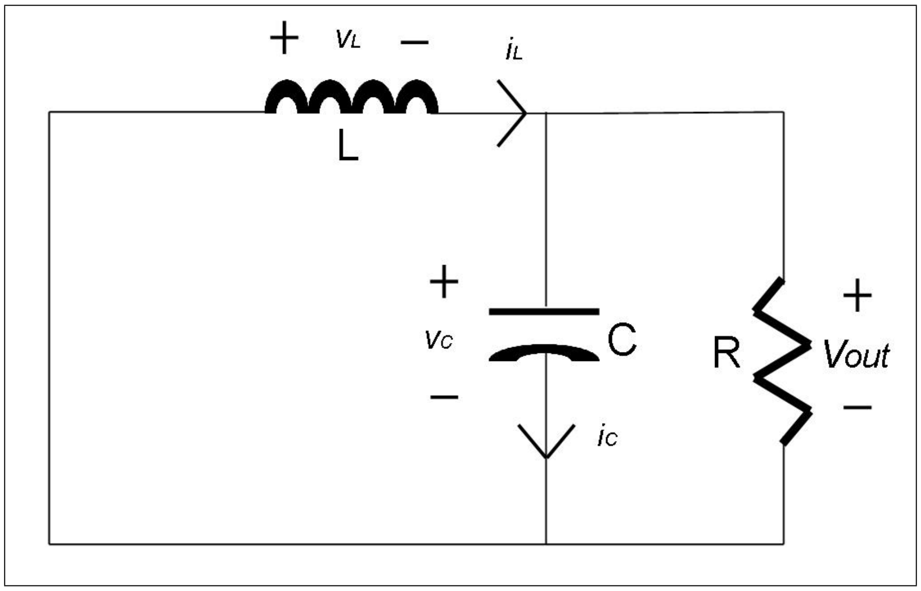

Figure 4.4: Equivalent ideal buck converter with switch open

$v_L = V_{in} - V_{out}$. Likewise, from Figure 4.4, during the remainder of the switching period when the transistor switch is open, $v_L = -V_{out}$. Therefore,

$$T_{ON}(V_{in} - V_{out}) + (T_{SW} - T_{ON})(-V_{out}) = 0$$

Multiplying through yields:

$$V_{in}T_{ON} - V_{out}T_{ON} - V_{out}T_{SW} + V_{out}T_{ON} = 0$$

which simplifies to:

$$V_{out} = \frac{T_{ON}}{T_{SW}}V_{in}$$

Recalling equation (4.1.1) reveals that the duty cycle acts as the voltage gain for the buck converter; therefore:

$$V_{out} = DV_{in}$$

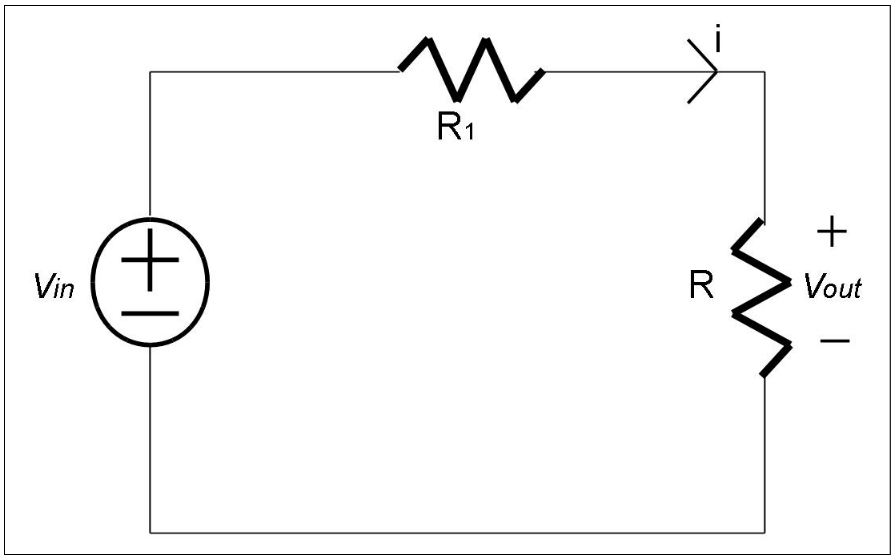

Figure 4.5: A simple voltage divider schematic

Another reason for selecting a buck converter is based on typical performance specifications. Regulating a source voltage to a lower voltage can be done using a simple linear regulator, such as a voltage divider network. However, many electronic circuit applications require performance characteristics that typical voltage dividers cannot provide. Understanding this additional benefit of buck converters will complete a basic introduction.

The circuit shown in Figure 4.5 represents a simple voltage divider circuit whose output voltage is described by:

$$V_{out} = \frac{V_{in}R}{R_1 + R}$$

This equation shows that the output voltage is a function of both the input voltage and the load resistance. Since a constant output voltage is usually the desired effect, using a voltage divider to regulate the output voltage is not an option in any circumstance where the input voltage or load resistance may vary. However, a closed-loop control system can be implemented with a buck converter to vary the duty cycle so that a desired output voltage can be maintained despite variations in the source voltage or load resistance.

Note also a voltage divider's efficiency:

$$\begin{aligned}\eta &= \frac{P_{out}}{P_{in}} \\ &= \frac{I^2R}{I^2R_1 + I^2R} \\ &= \frac{R}{R_1 + R}\end{aligned}$$

In this study's experiments, it is desired to regulate 5 V to 2.5 V. This case would require setting $R_1 = R$, making $\eta = 50\%$!

A buck converter's efficiency can be approximated by only considering the conduction loses (loses while the power transistor is in saturation mode). First, the input power is calculated as follows:

$$\begin{aligned}P_{in} &= V_{in}I_{avg} \\ &= \frac{V_{in}}{T_{SW}}\int_0^{T_{SW}} i(t)\,dx \\ &= \frac{V_{in}}{T_{SW}}\frac{V_{in}}{R}T_{ON} \\ &= D\frac{V_{in}^2}{R}\end{aligned}$$

And the output power is calculated as follows:

$$\begin{aligned}P_{out} &= RI_{RMS}^2 \\ &= R\frac{1}{T_{SW}}\int_0^{T_{SW}} i^2(t)\,dx \\ &= R\frac{T_{on}}{T_{SW}}\left(\frac{V_{in}}{R}\right)^2 \\ &= D\frac{V_{in}^2}{R}\end{aligned}$$

Given the calculated input and output power results, the buck converter efficiency is determined as follows:

$$\eta = \frac{P_{out}}{P_{in}} = 1$$

The efficiency is 100%! However, this calculation for the conduction losses does not factor in losses in the transistor, inductor, etc. More significantly, this completely ignores switching losses. Even though the buck converter's efficiency will not really be 100%, the efficiency is significantly improved over a voltage divider.

4.1.2 The Microcontroller: 18F PIC$^{\text{TM}}$

There are a few aspects of the 18F1220 PIC$^{\text{TM}}$ microcontroller that make its selection appropriate for this experiment. The 18F1220's processor can run up to 40 MHz by enabling the controller's internal phase-locked-loop and adding an external 10 MHz crystal oscillator. The controller also has a hardware multiplier, allowing for multiplications to be performed in one instruction cycle. Together, these features increase computational speed and, therefore, minimize the effects of computational delay in the digital controller.

The hardware multiplication capability also increases the number of possible compensator coefficients from which to select. The typical method of rotating bits to the left only allows for multiplication by powers of two. With the 18F1220's built-in multiplication instruction, one can multiply by any integer between 0 and 255. As a reminder, the microcontroller does not have a hardware divide, so any type of division has to be implemented in software. This issue is addressed in Section 4.2.

Additionally, the 18F1220 PIC$^{\text{TM}}$ has built-in a 10-bit analog-to-digital converter (ADC) and a 10-bit pulse-width modulator (PWM), which are both necessary for a digitally-controlled, closed-loop buck converter. Although these modules are 10-bit and the result of the multiply function is 16-bit, calculations and number representations are held constant at 8 bits to reduce algorithm complexity and to clearly demonstrate the effects of finite word-lengths.

The ADC is used to feedback the analog output voltage and represent it as a value between 0 to 255. The PWM is loaded with values between 0 and 255, indicating minimum and maximum duty cycles, respectively.

There is a trade-off to note with respect to the range of values that one wants to represent. As this range increases, the resolution of numbers that can be represented in the microcontroller decreases since:

$$V_{step} = \frac{V_{max} - V_{min}}{2^n - 1}$$

where V_{step} is the smallest incremental change in voltage that can be represented, n is the number of bits, and V_{max} and V_{min} are the largest and smallest voltages, respectively, that one wants to represent. The larger the range, the larger the potential measurement error. This can be compensated for by windowing circuits as described in [13], but this requires extra circuity and increased algorithm complexity that will unnecessarily complicate the experiment at hand. Therefore, the ADC's positive voltage reference is set to the source voltage (5 V) and the negative voltage reference to ground (0 V), and the buck converter will regulate an input voltage of 5 V to 2.5 V. As a result, for this experiment the voltage per step is:

$$V_{step} = \frac{5 \text{ V}}{255} = 19.61 \text{ mV per step}$$

And, since 2.5 needs to be represented by 127.5, and decimals cannot be represented, $V_{ref} = 127 = (01111111)_2$.

It is important to note this ADC resolution error; i.e., there is an inherent error of $V_{step}/2$, or 9.8 mV, in the reference voltage. Also, the error in the ADC feedback could be as great as 19.61 mV, which is 7.84% of the reference voltage. There is also a resolution problem that affects the accuracy of the PWM output because the maximum resolution of an 8-bit PWM module is 1/255 or 0.39%. The reference and ADC resolution errors are likely to show up in the steady-state error of the system. The PWM resolution error will effect the overall performance (rise-time and settling-time). Since these errors will be the same regardless of the operator used for parameterizing the PID controller's difference equation, the experiment's ability to compare the two approaches is not hindered.

The 18F1220's ADC module has one additional feature that should be discussed. There is a programmable acquisition time that allows for a set delay before beginning the ADC conversion. This allows the user to maintain a constant sampling rate and have sufficient time to complete the necessary calculations in between samples since other operations can be performed during the acquisition and ADC conversion times. This design feature is illustrated more fully in Subsection 4.2.5.

For this experiment, the PWM's switching frequency is set to 156.25 kHz. A PWM switching frequency value that is a few times larger than the sampling frequency, yet low enough to be measured by the available oscilloscope is desired. A variety of frequencies were used ranging from 39.06 kHz to 312.50 kHz. The effect of varying the switching frequency was unnoticeable in the experimental results.

Although many "more powerful" microcontrollers could have been selected to perform this experiment, the practical benefit of using this design approach on a $2.50 microcontroller [17] would have been lost. Digital Signal Processors, etc., are significantly more expensive.

For specifications on the 18F1220's hardware modules, its datasheet should be consulted [18].

4.2 Software

Most digital control systems are based on software-implemented algorithms. Although the 18F series PICs$^{\mathrm{TM}}$ can be programmed in C, the real-time operation of a DC-DC buck converter requires a speed that can only be accomplished through programming in assembly language. Therefore, the PIC$^{\mathrm{TM}}$ assembly language is used for the experiment algorithm.

The following subsections outline the specific parts of the algorithm. A general overview of the algorithm's overall structure is provided by Figure 4.6.

After initializing the algorithm constants and variables, the input/output pins and the ADC and PWM modules are set up. The code does not immediately begin attempting to compensate or else the power supplies' transients would dominate the initial transient measured by the oscilloscope. The program waits for a push-button input and enters the main loop. The main code logic is described simply by waiting for the ADC to complete, calculating the new duty cycle, updating the PWM, and updating the stored "past values" for the next time through the loop. This process continues until power to the microcontroller is removed.

4.2.1 Practical Issues

Many of the weaknesses of digital controls are present to varying degrees in the 18F1220 PIC$^{\text{TM}}$. The general computational scheme of the microcontroller is based on 8-bits. Although the PWM and ADC can operate with up to 10-bits and the multiply result is 16-bits, the experiment will be restricted to 8-bit processes. Once again, increasing the bits would increase the algorithm complexity and, therefore, amplify the effects of computational delay. Additionally, it is easier to draw conclusions about the effects of numerical restrictions if the same number of bits are used for all number representations and calculations. Since the numerical range of a microcontroller is from 0 to $2^n - 1$, the range of numbers that can be dealt with in this experiment is 0 to 255.

This microcontroller is fixed-point, meaning only integers can be represented. There are techniques for addressing this issue. For instance, all numbers, n, could be represented in the microcontroller as n scaled by some power of 10. So, if every number is represented by 100-times its value, 1.83 would be represented as 183. But this approach would mean that the largest number that could be represented would be 2.55! Using this method, the range, in general, would be 0 to $(2^n - 1)/10^x$, where $x = 0, 1, 2, ...$ This experiment will

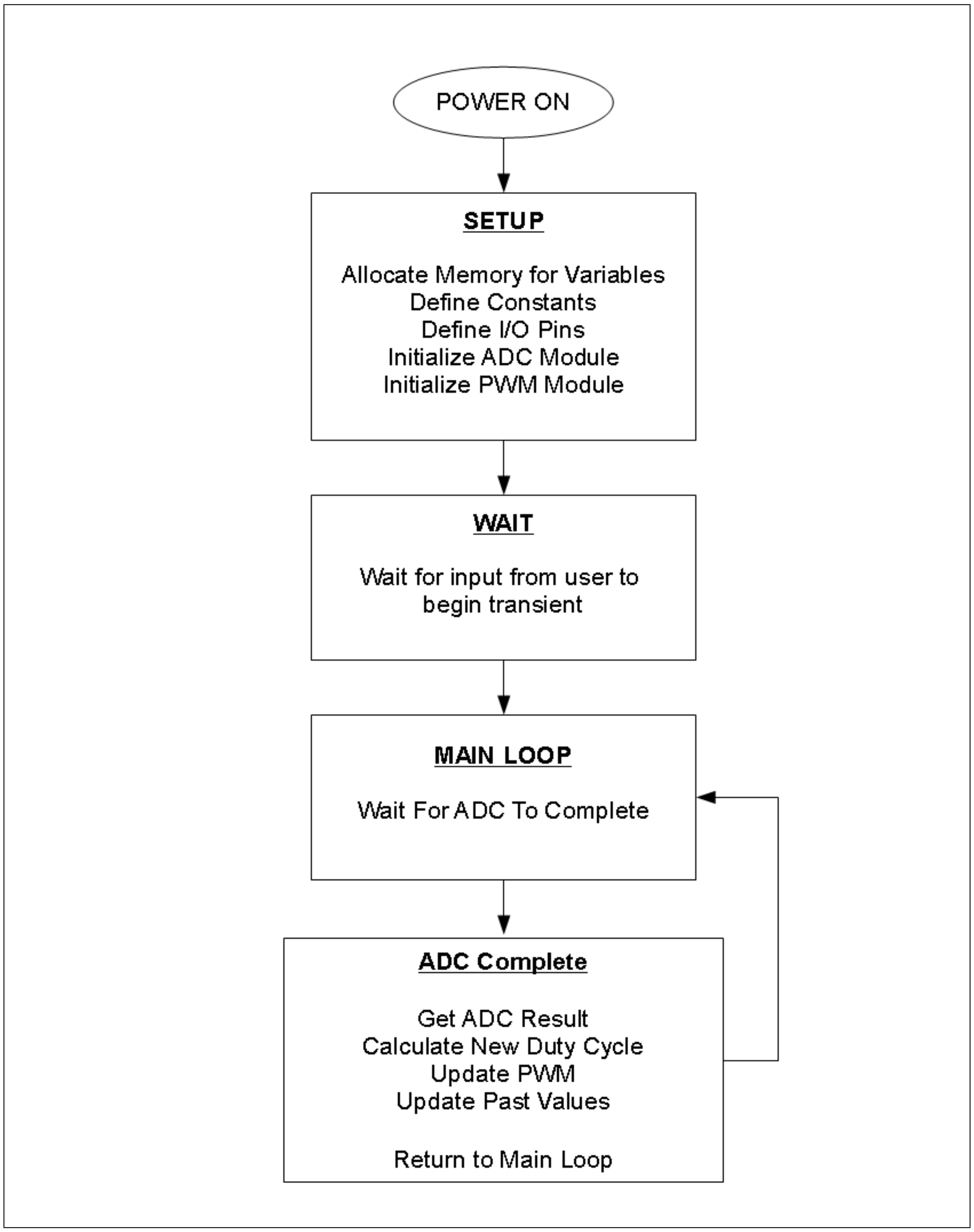

Figure 4.6: General overview of software algorithm

not attempt to handle floating-point numbers (i.e., $x = 0$). Numbers will be rounded to the nearest integer.

Another weakness is the inability to divide. Because of this, the classical method of rotating bits will be utilized. Because numbers will be represented by combinations of eight $1's$ and $0's$, rotating the bits right x times, is the equivalent of dividing by 2^x. Not only does this significantly limit the number of coefficient options, but with every rotate right, resolution in the result is lost since those bits are dropped completely. The inability to represent decimals will be present in both the shift and delta operator representations of the difference equations, so the analysis goals will be unaffected. It will turn out, though, that only the delta operator will have coefficients that will require division, putting it at a slight disadvantage for this study. However, even though the responses will be affected, the simulation and experimental conclusions will not be distorted.

Negative numbers also cannot be inherently represented. The microcontroller is capable of 2's-complement arithmetic, but this would limit subtractions to only 7 bits since the MSB serves as the sign bit. In order to keep the range of numerical representation constant, each 8-bit term has an additional variable associated with its sign. This gives the experiment the ability to represent numbers from $-2^n + 1$ to $2^n - 1$, or -255 to 255. Keeping up with the sign increases algorithm complexity and calculation time, but it is necessary to clearly investigate the delta and shift operators' sensitivity to finite word-lengths.

4.2.2 Addition/Subtraction Subroutines

In order to implement signed addition and subtraction, a few subroutines must be coded. The SimpleSubtraction subroutine is designed to subtract two numbers that are both known to be positive; the result, however, could be positive or negative. The flowchart for this subroutine is presented in Figure 4.7. Note that the nature of this algorithm will never result in an overflow or underflow.

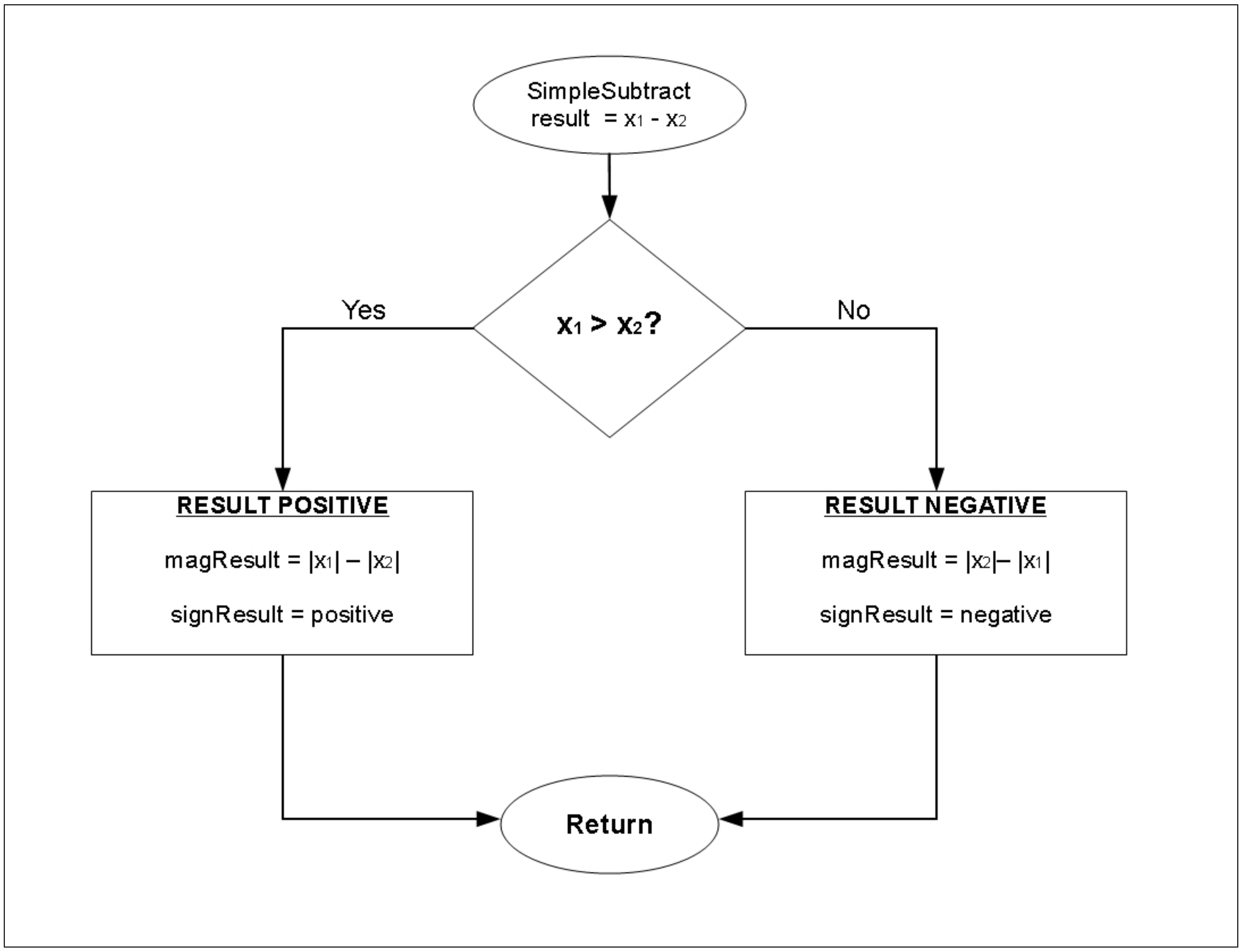

Figure 4.7: Subtraction when both numbers are known to be positive

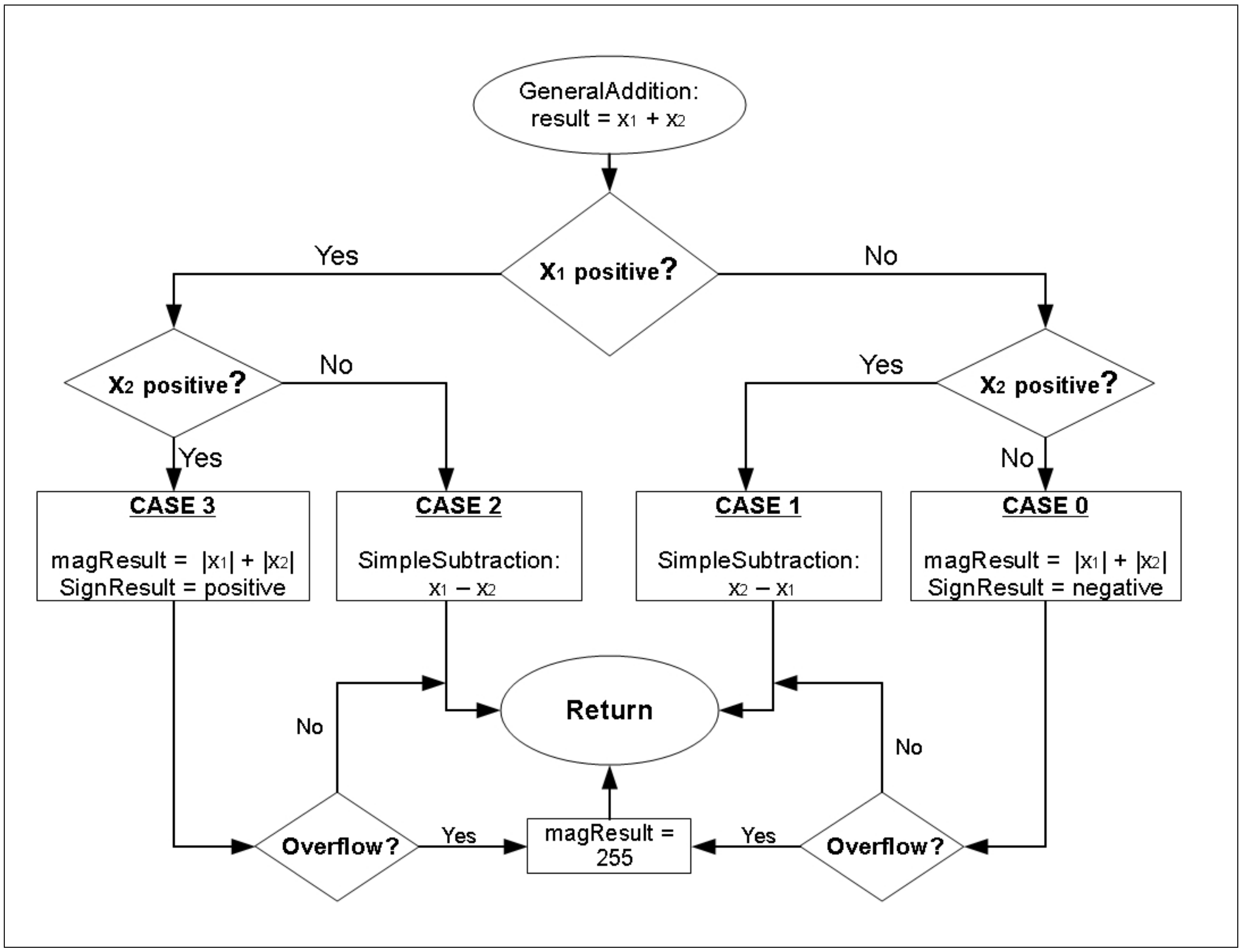

Figure 4.8: Addition of two numbers that could be either positive or negative

The two signed subroutines, GeneralAddition and GeneralSubtraction, follow the flowcharts of Figures 4.8 and 4.9. If a result's magnitude ever causes an overflow (exceeds 255), then the value is truncated to 255. The logic of GeneralAddition and GeneralSubtraction will never result in an underflow.

4.2.3 Duty-Cycle Calculation: Shift Operator

The flowchart of the duty cycle calculation based on the shift operator's PID controller, equation (3.2.5), can be found in Figure 4.10.

The flowchart is quite self-explanatory: the new error value is calculated based on the reference and the ADC result, the new duty cycle is calculated, the PWM is updated, and

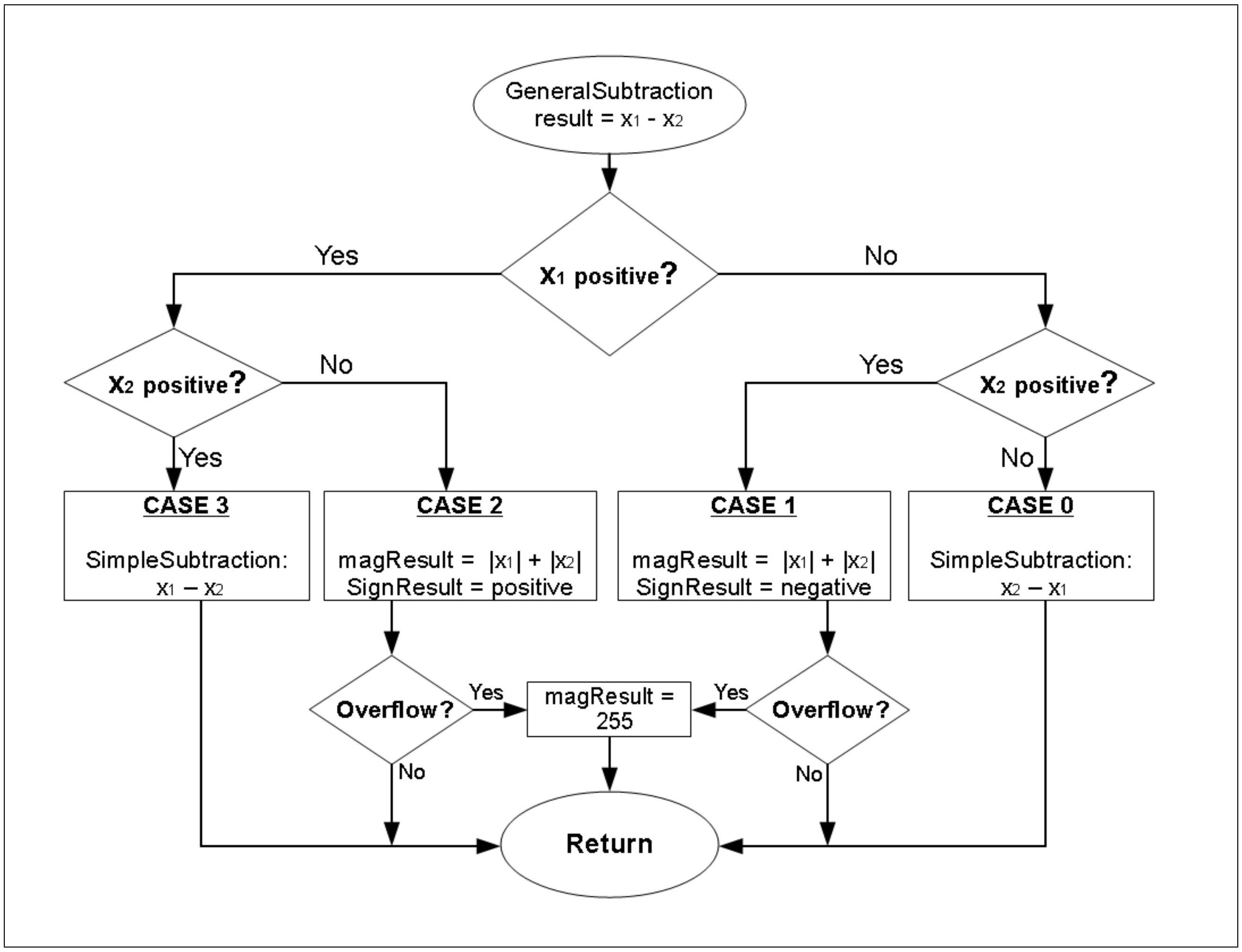

Figure 4.9: Subtraction of two numbers that could be either positive or negative

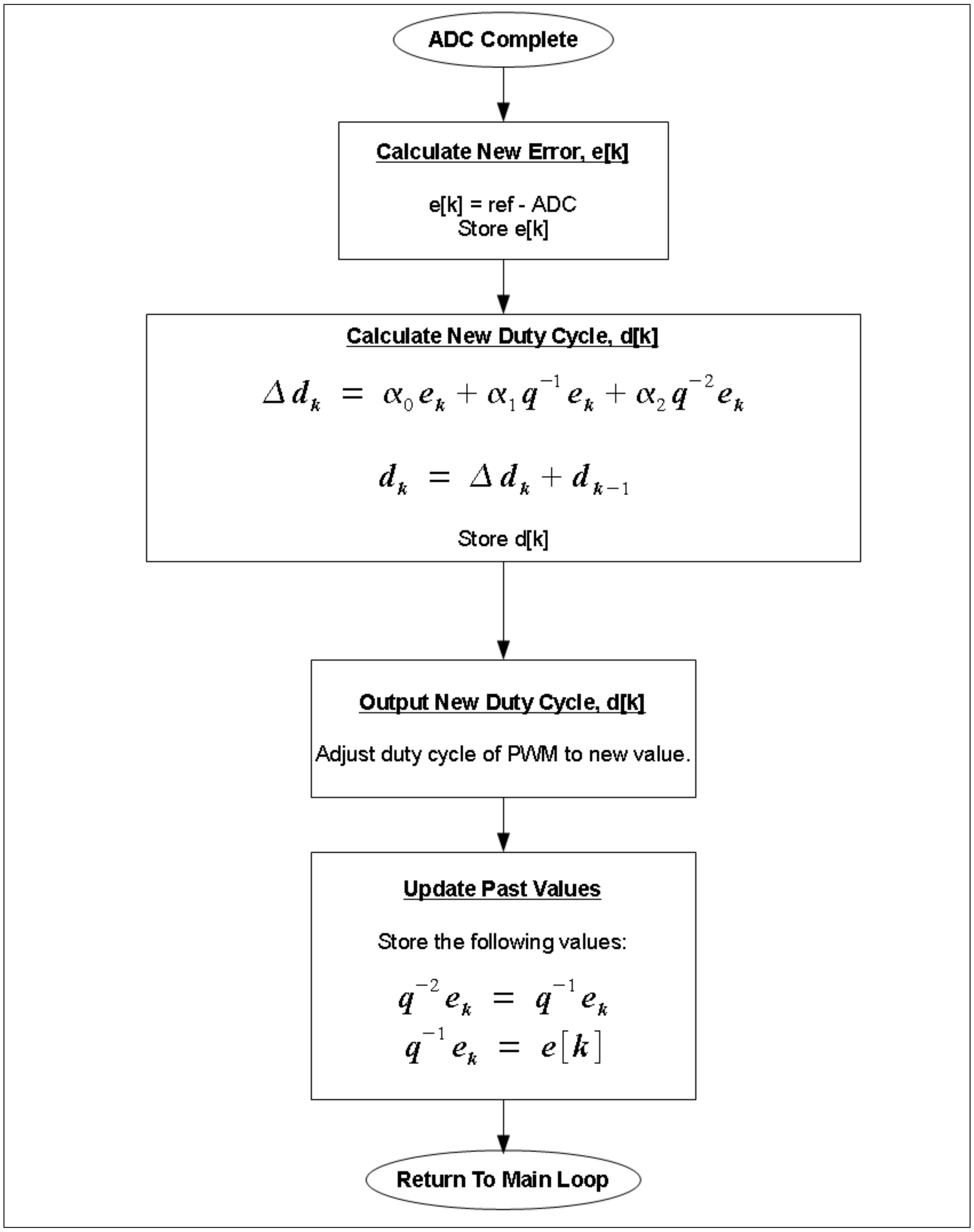
ADC Complete
Calculate New Error, e[k]
e[k] = ref - ADC
Store e[k]
Calculate New Duty Cycle, d[k]
$\Delta d_k = \alpha_0 e_k + \alpha_1 q^{-1} e_k + \alpha_2 q^{-2} e_k$
$d_k = \Delta d_k + d_{k-1}$
Store d[k]
Output New Duty Cycle, d[k]
Adjust duty cycle of PWM to new value.
Update Past Values
Store the following values:
$q^{-2} e_k = q^{-1} e_k$
$q^{-1} e_k = e[k]$
Return To Main Loop

Figure 4.10: Overview of duty cycle calculation based on the shift operator

the past values of the error are updated. Every addition and subtraction that takes place is a result of a call of the GeneralAddition or GeneralSubtraction subroutine (Figures 4.8 and 4.9, respectively), except for the "new error" subtraction. The calculation to find the current error value calls the SimpleSubtraction subroutine (Figure 4.7) because neither the reference value nor the ADC result will ever be negative.

The remainder of the calculations use GeneralAddition or GeneralSubtraction because the variables can be either positive or negative. The logic, overflow handling, etc. are not shown in Figure 4.10 and are accounted for in the addition and subtraction subroutines.

4.2.4 Duty Cycle Calculation: Delta Operator

Equation (3.2.10) lays the foundation to implement the delta-operator-based duty cycle calculation. The flowchart of this section of the algorithm can be found in Figure 4.11.

The main point here is the new error calculation is the only one that does not use the GeneralAddition and GeneralSubtraction subroutines, since that is the only calculation where all the values are known to be non-negative.

Other than the different equations for calculating the new duty cycle, there is only one difference in the delta operator algorithm as compared to the shift operator algorithm. For the shift operator, all the past values are updated after the new duty cycle value is applied to the PWM output. For the delta operator approach, some past values are updated at this point and other values are updated immediately before the new duty cycle is calculated. This design requirement is because the delta operator's difference equation is not only based on previous values of the error, but on previous differences in error values. These values cannot be determined until the new error value is calculated.

In some cases, this dissimilarity may be significant, not so much because the past values are updated at a different point or because it slightly increases algorithm complexity, but because additional calculations must be performed to arrive at these past values. This fact indicates that, in general, the delta operator's difference equation algorithm will take longer

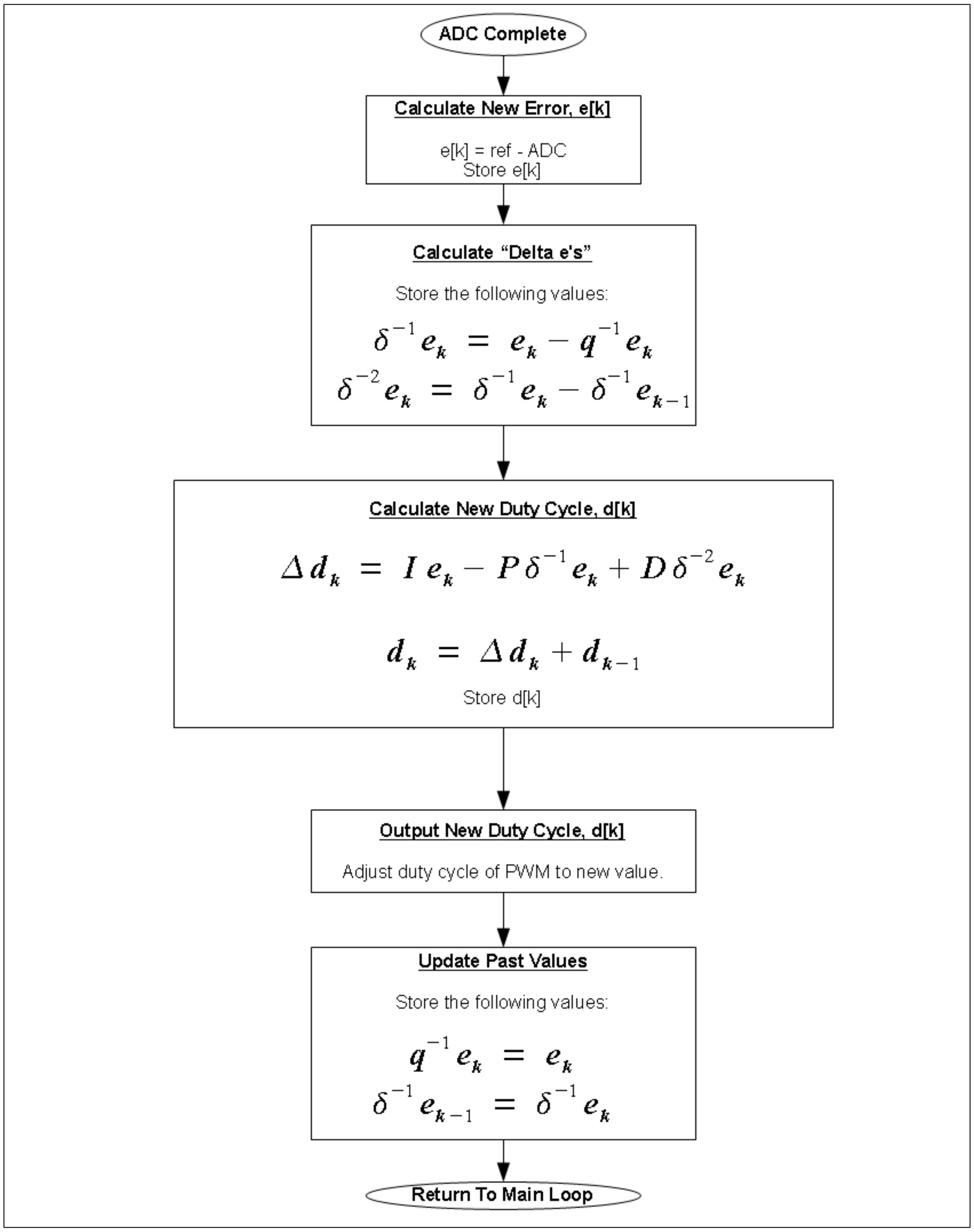
ADC Complete
Calculate New Error, e[k]
e[k] = ref - ADC
Store e[k]
Calculate "Delta e's"
Store the following values:
$\delta^{-1} e_k = e_k - q^{-1} e_k$
$\delta^{-2} e_k = \delta^{-1} e_k - \delta^{-1} e_{k-1}$
Calculate New Duty Cycle, d[k]
$\Delta d_k = I e_k - P \delta^{-1} e_k + D \delta^{-2} e_k$
$d_k = \Delta d_k + d_{k-1}$
Store d[k]
Output New Duty Cycle, d[k]
Adjust duty cycle of PWM to new value.
Update Past Values
Store the following values:
$q^{-1} e_k = e_k$
$\delta^{-1} e_{k-1} = \delta^{-1} e_k$
Return To Main Loop

Figure 4.11: Overview of duty cycle calculation based on the delta operator

to execute than the shift operator algorithm. Therefore a higher maximum sampling rate can be achieved using the shift operator's parameterization. This will be shown in more detail in Subsection 4.2.5. However, the emphasis of this thesis is not the length of time it takes to perform the calculations. All experiments will be performed at constant sampling rates.

4.2.5 Timing Diagram

The two main timing influences of a real-time digital control system are typically the ADC conversion time and the time it takes to calculate the new control signal value. The A/D conversion clock, T_{AD}, is selected by the software programmer, but there is a minimum conversion time. With the processor running at 40 MHz, a T_{AD} of 1.6 μs is required. An analog to digital conversion requires 11 T_{AD} cycles to complete. For this experiment, it will take 17.6 μs for a conversion to complete (i.e., $T_{conv} = 17.6\ \mu\text{s}$).

A 40 MHz processor makes one clock cycle equal to 25 ns. A PIC$^{\text{TM}}$ requires four clock cycles to execute one instruction cycle; therefore, this program will perform one instruction every 100 ns. At most, the current implementation of the shift operator's algorithm executes about 145 instruction cycles to complete the full duty cycle update loop. This equates to a minimum requirement of about 14.5 μs. The current delta operator approach uses up to about 217 instruction cycles, or 21.7 μs. The detailed timing diagrams for these two algorithms are found in Figures 4.12 and 4.13. The explanations of each diagrams' segment are well documented in the flowcharts previously shown in Figures 4.10 and 4.11.

To provide sufficient time for all necessary calculations to complete and to maintain a constant sampling rate, a programmable acquisition time, T_{acq}, will be utilized as described earlier. One available selection option for this value is 20 T_{AD}, or 32 μs. This value was selected for the initial experiments, so the total sampling period, T_s, is:

$$T_s = T_{acq} + T_{conv} = 32\ \mu\text{s} + 17.6\ \mu\text{s} = 49.6\ \mu\text{s}$$

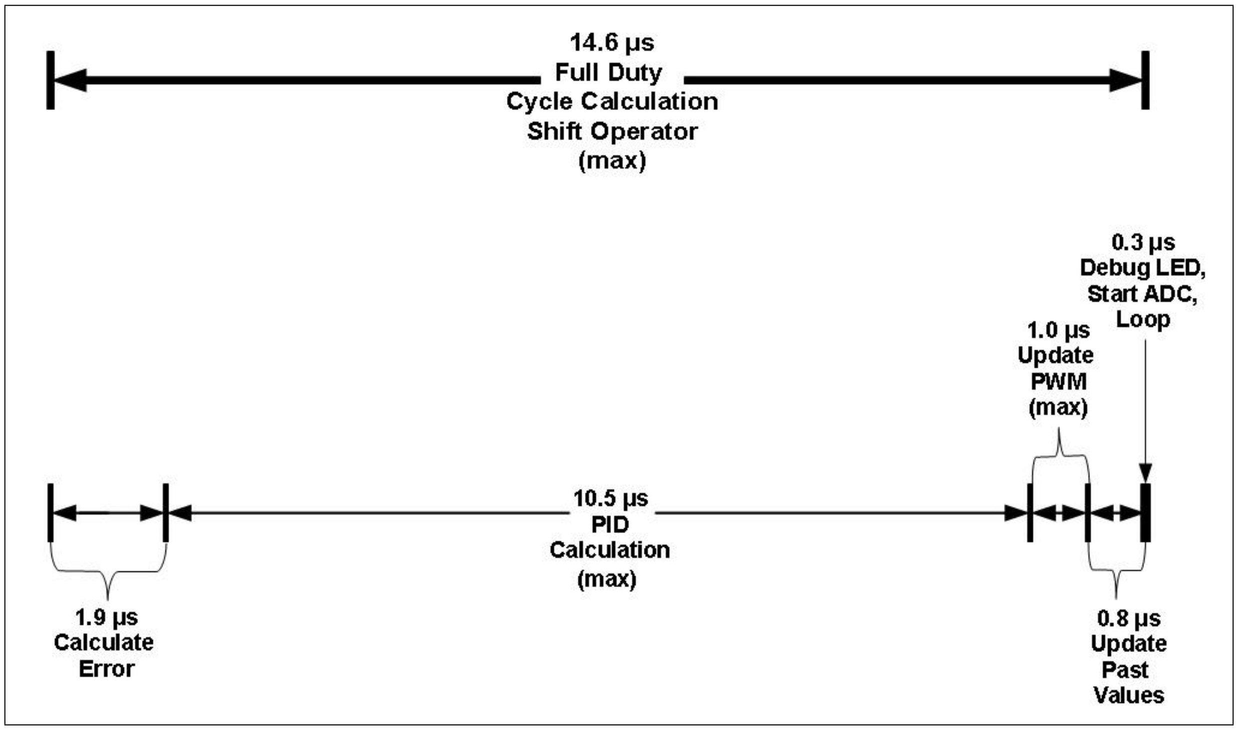

Figure 4.12: The approximate timing diagram for the shift-operator-based controller algorithm

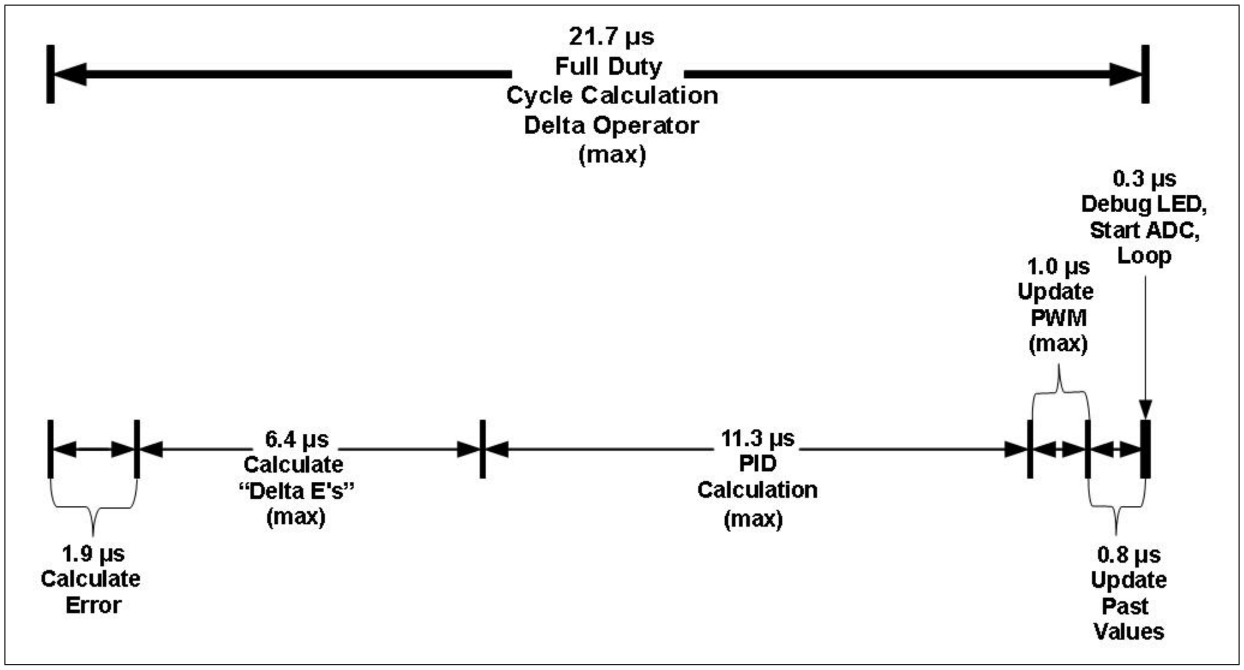

Figure 4.13: The approximate timing diagram for the delta-operator-based controller algorithm

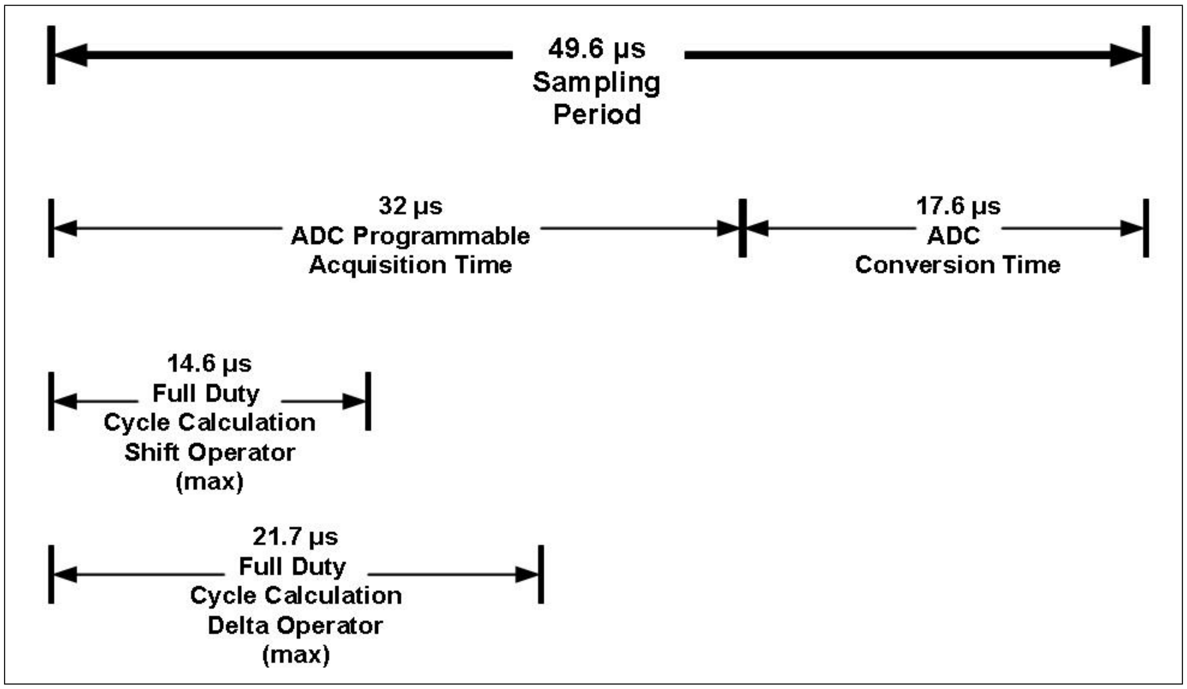
49.6 μs Sampling Period
32 μs ADC Programmable Acquisition Time
17.6 μs ADC Conversion Time
14.6 μs Full Duty Cycle Calculation Shift Operator (max)
21.7 μs Full Duty Cycle Calculation Delta Operator (max)

Figure 4.14: The approximate timing diagram for the overall system

Therefore, the sampling frequency, f_s, is about 20.16 kHz. The overall timing diagram at this sampling rate can be seen in Figure 4.14

Chapter 5

System Modeling and Simulation

Before conducting the experiments, design tools are used to design and simulate the control system. MATLAB/Simulink were chosen for the simulation platform. Simulating the system allows for quick verification and design of the control algorithms.

First, a model of the plant, the DC-DC buck converter, is developed that can be implemented in Simulink. Then, models of the shift-based and delta-based digital PID controllers are developed. Each controller model is implemented in Simulink.

5.1 Modeling of DC-DC Buck Converters

In order to perform designs and simulations with a DC-DC buck converter, it is necessary to develop a mathematical model describing its characteristics. There are multiple ways to approach model development. This thesis develops a set of state equations because of the following:

1. State equations are developed based on common first-principle physics relationships.
2. Corresponding state diagrams are easily implemented in simulation software such as Simulink and LabVIEW.
3. An option to account for non-linear characteristics is available.
4. Adjustments to model parameters are easily made for model updates and variation testing.

To account for some of the non-ideal properties in buck converters, the state equations are based on Figure 5.1.

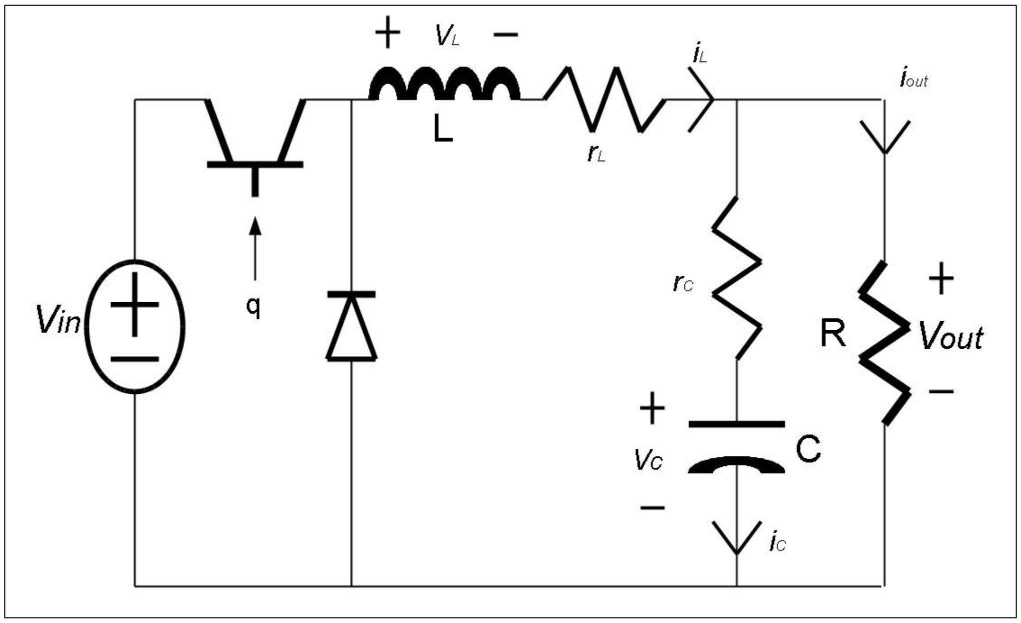

Figure 5.1: A typical buck converter with inductor and capacitor ESRs

The equivalent series resistances (ESRs) of the inductor and capacitor are represented by r_L and r_C, respectively. The transistor will operate in saturation mode when it is "ON" and has a drain-source resistance, r_T. When the transistor switch is "OFF", the diode will conduct and be modeled by a DC voltage source, v_D, in series with a resistance, r_D.

The switching effects cause unavoidable non-linearities in buck converters; nevertheless, these effects can be included in the state-space model. This modeling will be demonstrated in the development of the equations. One output equation is written in addition to the two state equations for both the "ON" and "OFF" states (or operating modes) of the buck converter. The inductor current, i_L, and the capacitor voltage, v_C, will serve as the state variables for modeling. The output current, i_{out}, control signal, q, and the input voltage, v_{in}, are considered "inputs."

By examining either the "ON" state (Figure 5.2) or the "OFF" state (Figure 5.3) of the non-ideal buck converter, the output voltage is:

$$v_{out} = v_C + i_C r_C$$

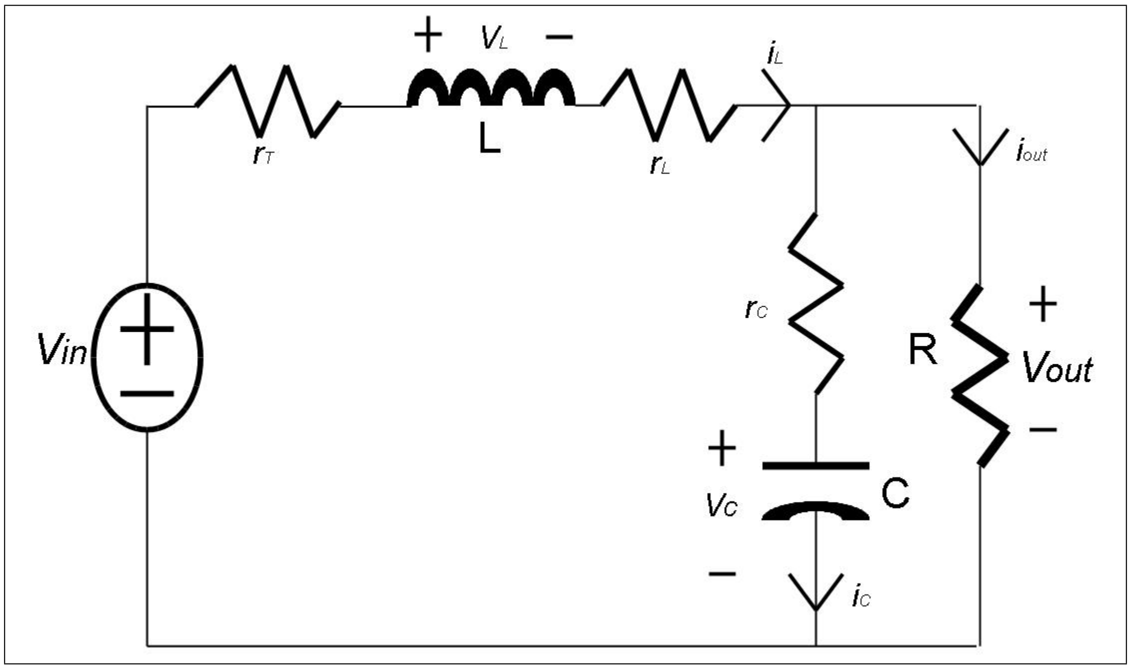

Figure 5.2: Non-ideal buck converter with transistor switch closed

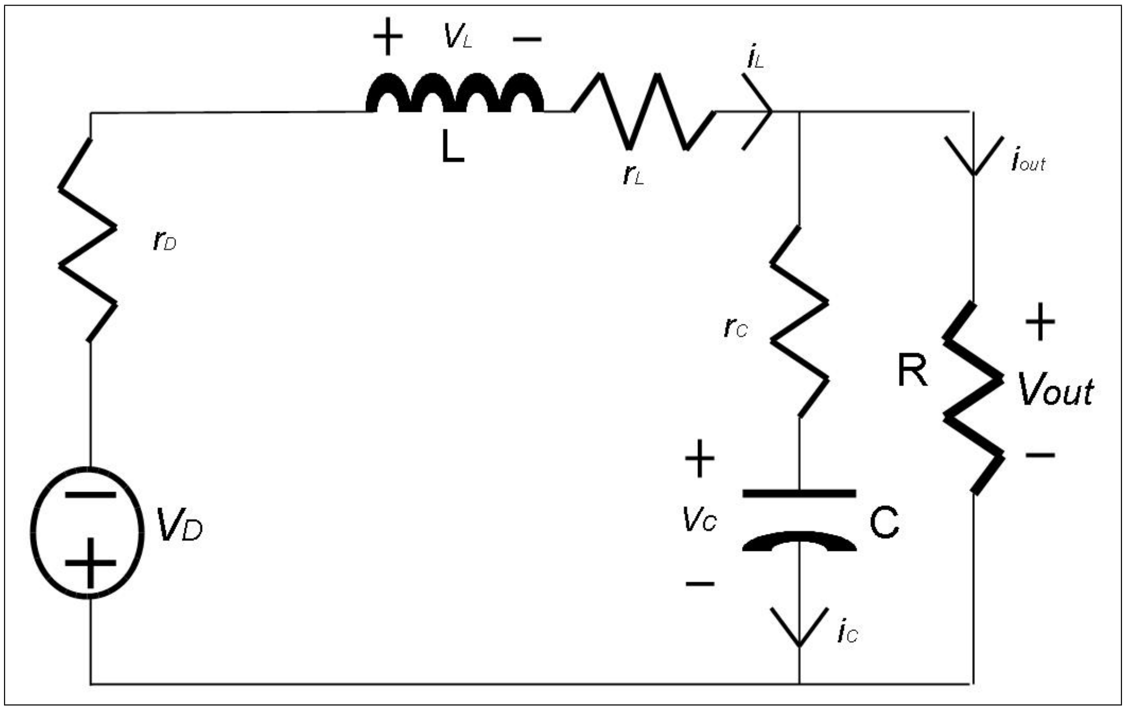

Figure 5.3: Non-ideal buck converter with transistor switch open

Writing this expression in terms of the state variables and inputs yields:

$$v_{out} = v_C + r_C(i_L - i_{out}) \tag{5.1.1}$$

To find the required state equations, the differential equations that relate capacitors' and inductors' currents and voltages are used. Beginning with the inductor current:

$$\frac{\mathrm{d}\, i_L}{\mathrm{d}\, t} = \dot{i}_L = \frac{1}{L} v_L$$

The inductor voltage, v_L, needs to be replaced with a combination of state variables and inputs. When the transistor switch is "ON", the change in inductor current is:

$$\dot{i}_L = \frac{1}{L}(v_{in} - v_o - i_L r_L - i_L r_T)$$

and when the transistor switch is "OFF", the inductor current is:

$$\dot{i}_L = \frac{1}{L}(-v_D - v_{out} - i_L r_L - i_L r_D)$$

Because these are time-based signals, v_{in} and $i_L r_T$ are multiplied by q, and v_D and $i_L r_D$ are multiplied by the inverse of q, $\bar{q}$. This accounts for the non-linearities. Remember that $q = 1$ when $0 \leq t \leq T_{ON}$, and $q = 0$ when $T_{ON} < t \leq (T_{SW} - T_{ON})$. This yields:

$$\dot{i}_L = \frac{1}{L}\big(-v_{out} - i_L r_L + q(v_{in} - i_L r_T) - \bar{q}(v_D + i_L r_D)\big) \tag{5.1.2}$$

The change in the capacitor voltage is:

$$\frac{\mathrm{d}\, v_C}{\mathrm{d}\, t} = \dot{v}_C = \frac{1}{C} i_C$$

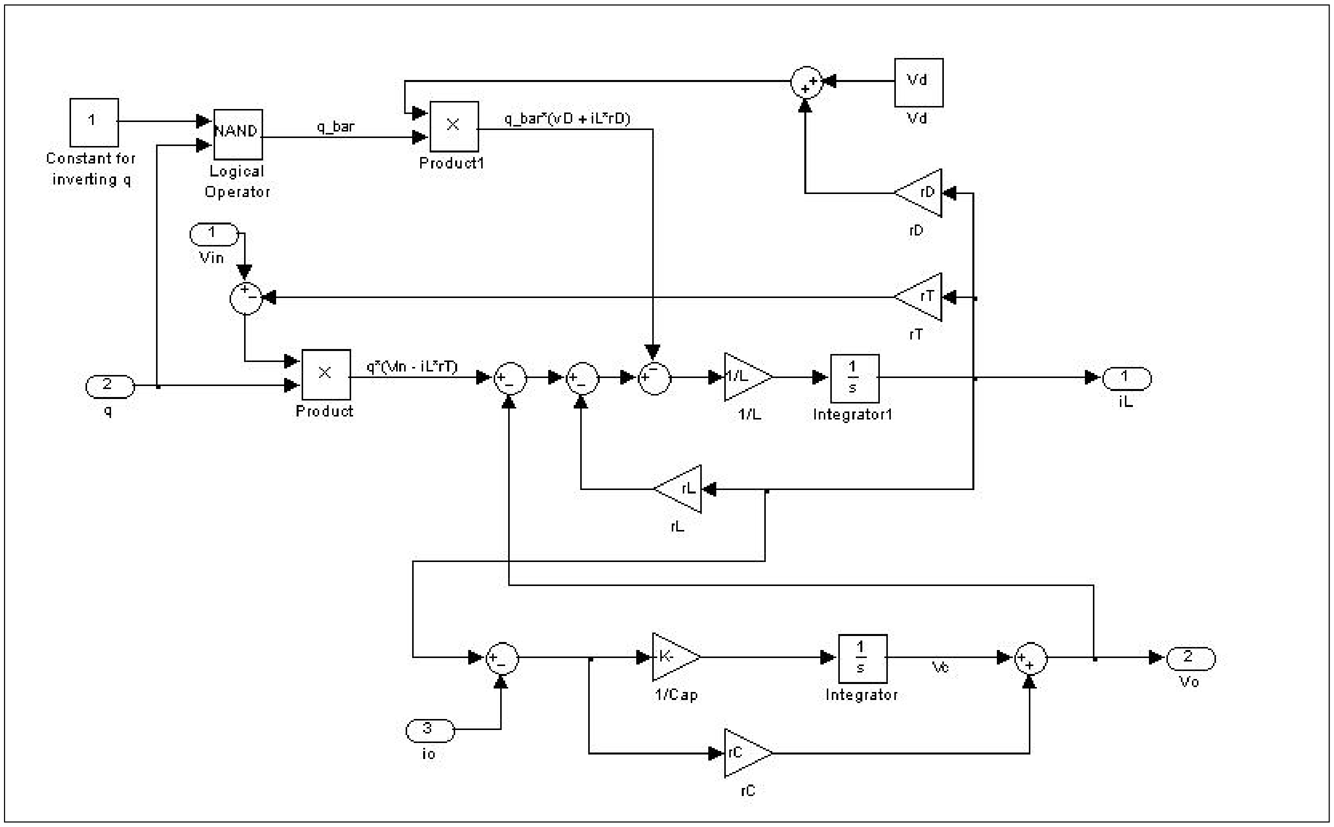

Figure 5.4: State diagram for a buck converter

For both the "ON" and "OFF" states, $i_C = i_L - i_{out}$; therefore, the second state equation is:

$$\dot{v}_C = \frac{1}{C}(i_L - i_{out}) \tag{5.1.3}$$

A state diagram can be composed using equations (5.1.1), (5.1.2), and (5.1.3). The state diagram model is built directly from Simulink's default library as shown in Figure 5.4.

Table 5.1 contains the model parameters for the simulation.

The composite Simulink model of the buck converter is presented in Figure 5.5 where the power stage is the state diagram in Figure 5.4 and the PWM generator is shown in Figure 5.6. The PWM generator works by subtracting a sawtooth wave from the duty cycle. As long as the result is greater than 0, the output will be "ON". D_{nom} is the nominal

Symbol	Parameter	Value	Units
V_{in}	Input Voltage	5	V
R	Load Resistance	5	Ω
L	Series Inductor	150	μH
r_L	ESR of Inductor	10	mΩ
C	Output Capacitor	1000	μF
r_C	ESR of Capacitor	30	mΩ
V_D	Forward Drop Across Diode	0.4	V
r_D	ESR of Diode when Conducting	450	mΩ
r_T	Drain-Source Resistance of Transistor	40	mΩ
V_{ref}	Reference Voltage	2.5	V

Table 5.1: Model parameters for experimental buck converter

duty cycle, V_{ref}/V_{in}. The simulations and experiments are performed with $V_{ref} = 2.5V$. Therefore, $D_{nom} = 0.5$.

The experimental buck converter used for this experiment was used previously by Feng's and Guo's theses [12] [15]. Values for L, r_L, C, and r_C were attained from these two theses. Values for V_{in}, V_{ref}, and R were selected to minimize external circuitry and additional code which would be required to handle resolution and referencing issues. The value of r_T was obtained from the power transistor's data sheet [19]. The value for the diode voltage, v_D, is a typical drop across a Schottky diode and is consistent with the instantaneous forward current to instantaneous forward voltage curve provided on the part's datasheet [20]. The diode resistance, r_D, was selected to give a more reasonable amount of damping to the simulation. This effect is shown in the open-loop responses. The initial transient response of Simulink's open-loop model is shown in Figure 5.7.

To increase confidence in the model results, the open-loop response was plotted using the schematic editor in Pspice; this schematic is shown in Figure 5.8. The result of the Pspice simulation is shown in Figure 5.9.

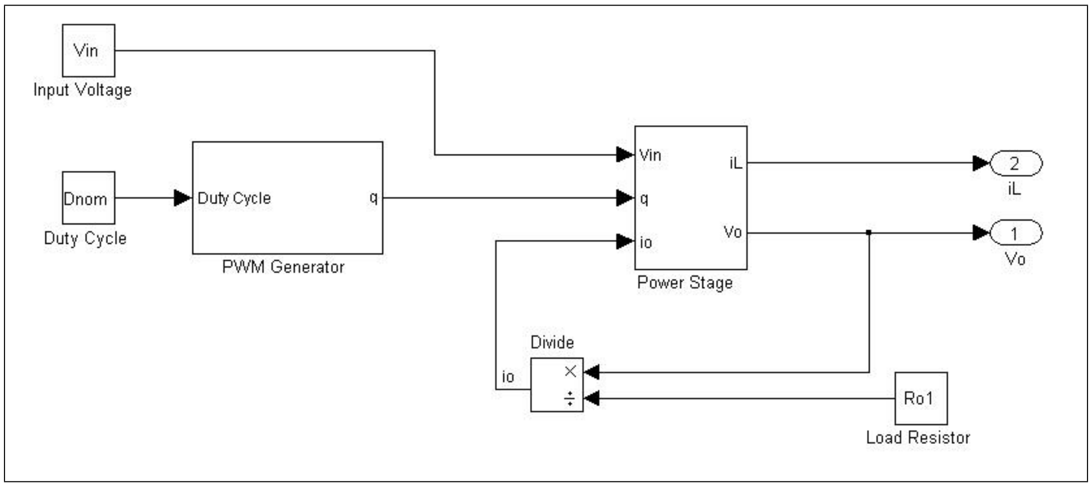

Figure 5.5: Open-loop buck converter Simulink model

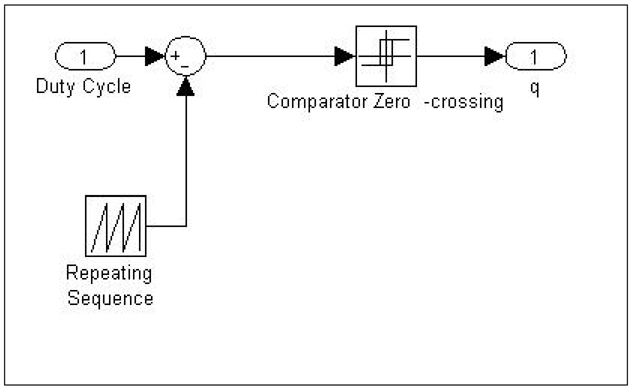

Figure 5.6: PWM generator Simulink model

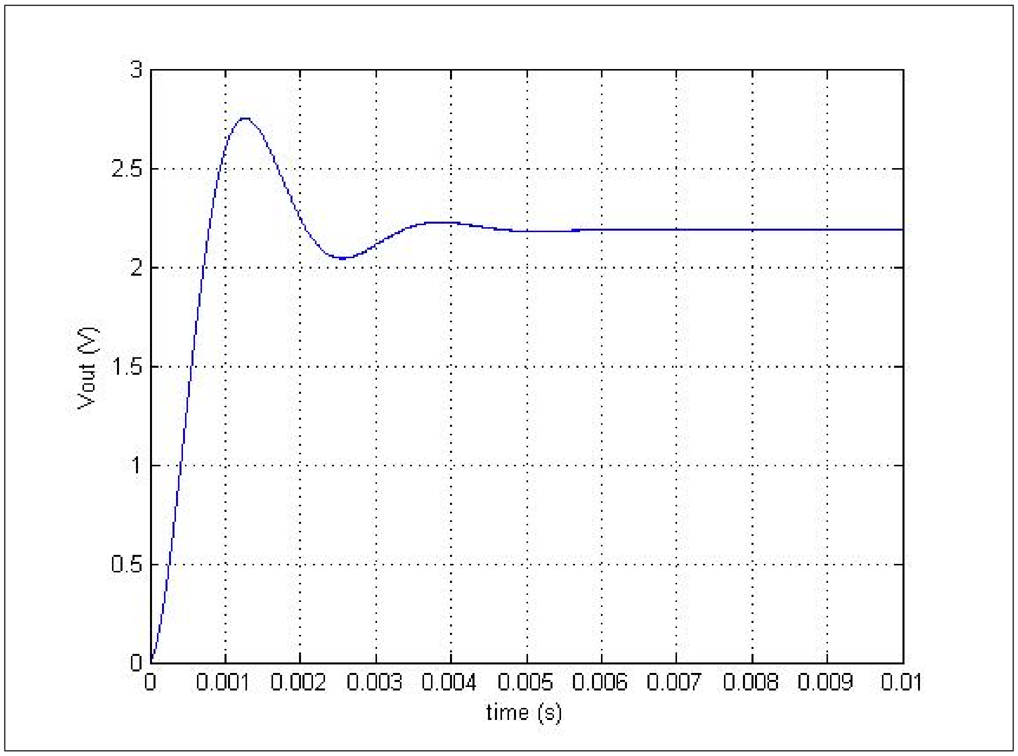

Figure 5.7: Simulink simulation of open-loop response with $V_{ref} = 2.5V$

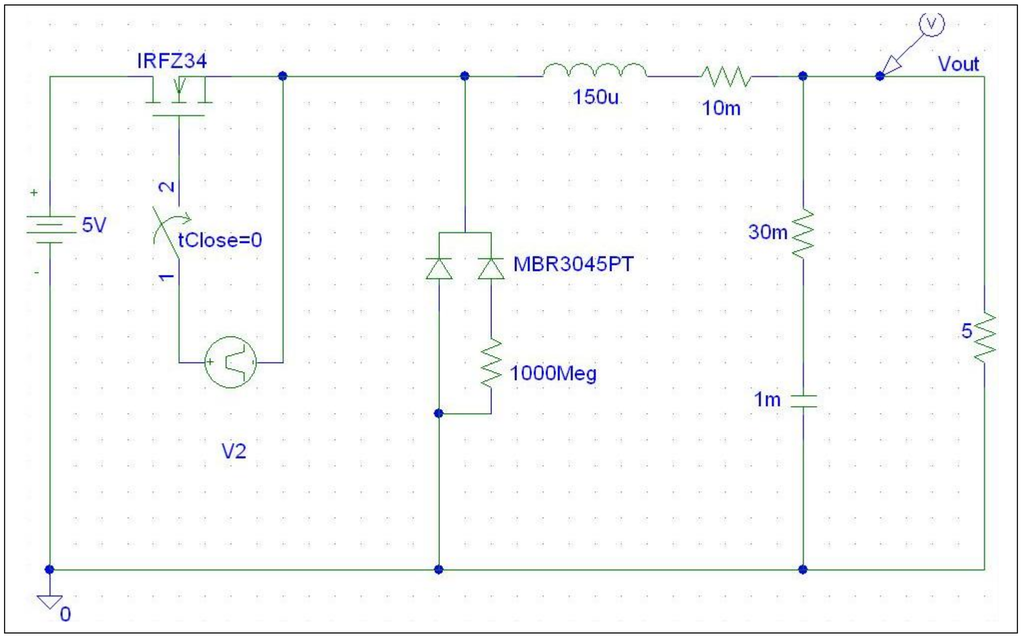

Figure 5.8: Pspice schematic of open-loop buck converter

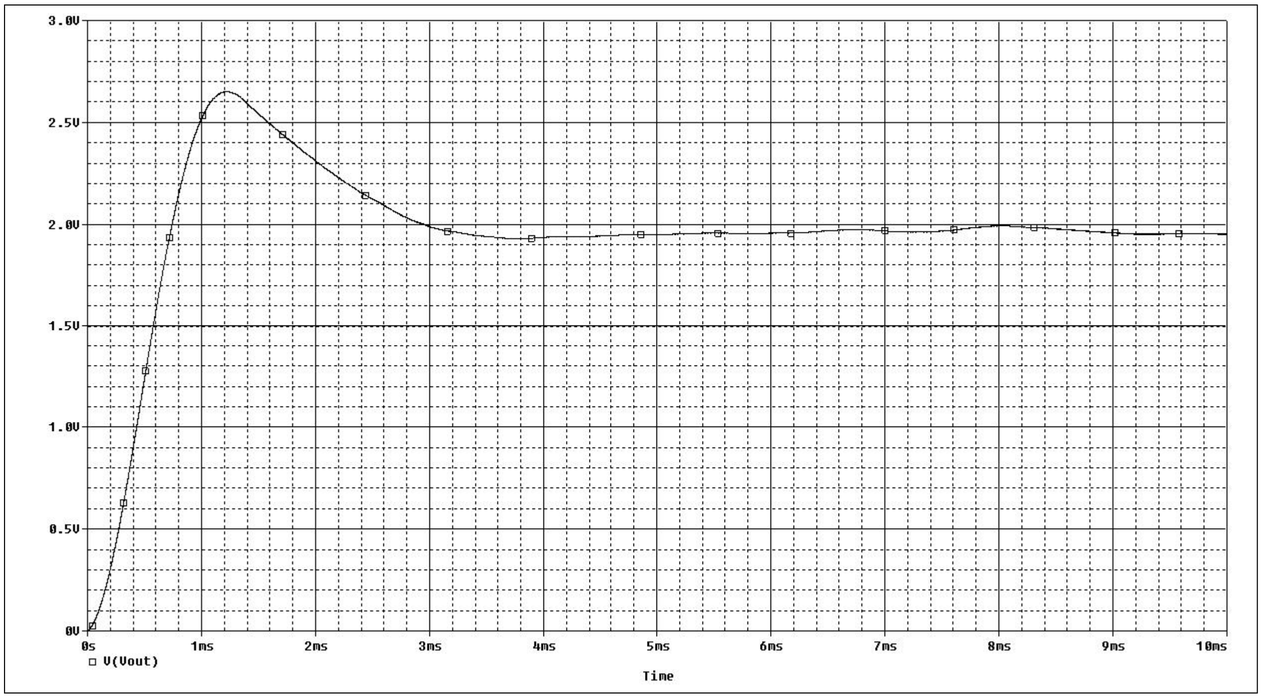

Figure 5.9: Pspice simulation result of open-loop response

With the exception of some additional damping in the Pspice model, the dynamic response of the Pspice model and the Simulink model favor each other well.

The actual open-loop response was measured and is compared to the Simulink simulation in Figure 5.10. The Pspice simulation replicates the actual system response more accurately than the Simulink simulation does. However, the Simulink simulation satisfactorily resembles the actual system so that it can be used for designing the closed-loop system. The digital modeling of the system is immensely easier to accomplish in MATLAB/Simulink rather than Pspice. The additional damping that the actual system will experience will be factored into the design of the closed-loop system in the simulation.

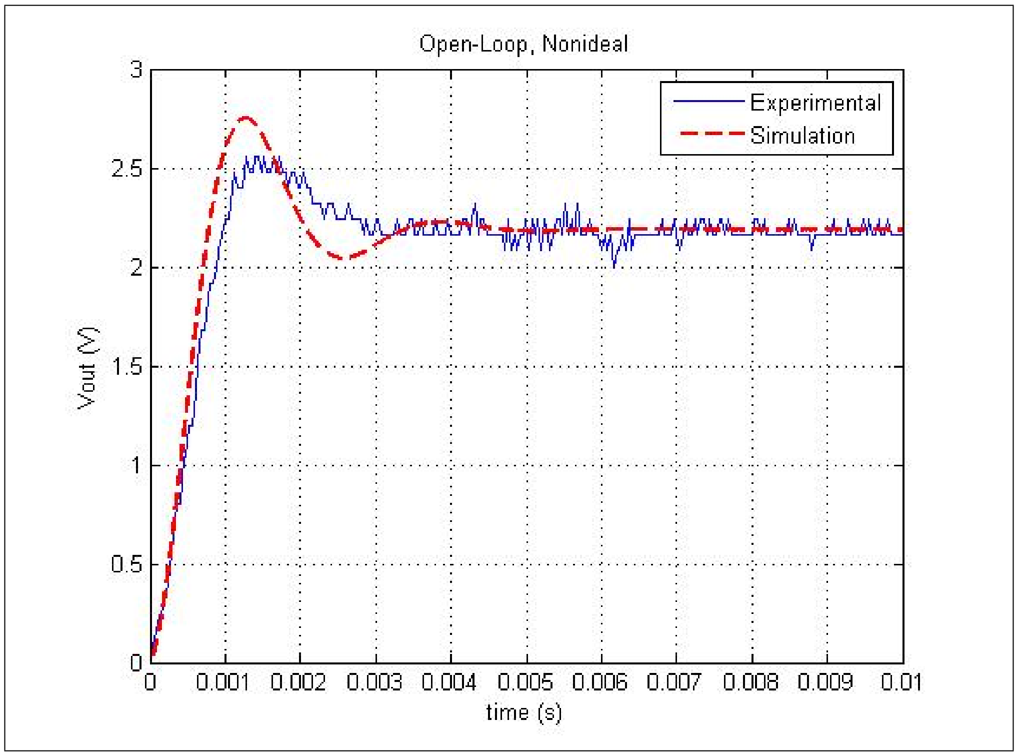

Figure 5.10: Experimental versus simulation open-loop response

5.2 Modeling of Analog PID Controllers

Modeling a closed control loop with an analog PID controller in a graphical simulation program is common practice. Figure 5.11 illustrates the simulation used to design the controller coefficients, K_P, K_I, and K_D, for this experiment. These coefficients correspond to the classic PID equation given previously in equation (3.2.1).

The output voltage, V_o, is fed back and subtracted from the reference voltage, V_{ref}. The resultant error signal is fed into the three compensator terms. The three terms add together to produce the control signal, or the duty cycle, D. As discussed previously, a change in D adjusts the buck converter output voltage. The duty cycle is adjusted until the error signal equals zero (i.e., the output voltage equals the reference voltage).

This simulation is set up to test the initial transient as well as a step change in load. Testing the initial transient is equivalent to a reference step change from $0V$ to whatever

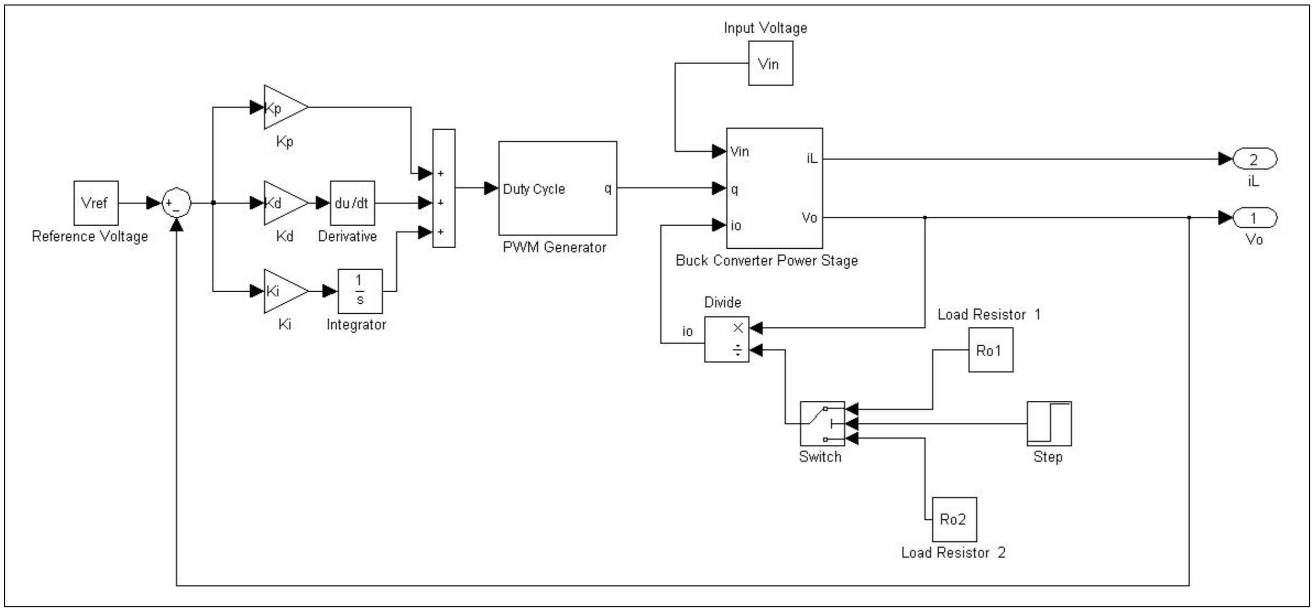

Figure 5.11: Closed-loop buck converter with continuous-time PID controller

reference voltage is desired ($2.5V$ in the case of this experiment). The step change in load will not be demonstrated in this thesis.

The following section describes the models used to simulate the two digital compensators.

5.3 Modeling of Non-ideal Digital Controllers in Simulink

The closed-loop system model using a digital compensator is illustrated in Figure 5.12. Digital controller constraints are modeled in this simulation. The ADC Delay block models the time delay due to the analog-to-digital converter (17.6 μs). This value is then scaled to its equivalent digital value between 0-255 and any decimal is dropped due to the Zero-Order Hold, Scale, Saturate and Quantizer blocks. V_{ref} experiences a similar scaling and quantization.

The output of the digital compensator is the change in duty cycle, Δd_n, so it is combined with the previous value of the duty cycle, d_{n-1}. (Recall that z^{-1} is equivalent to q^{-1}.) The duty cycle is truncated from 0-255 before being applied to the PWM. (The -255 to 255

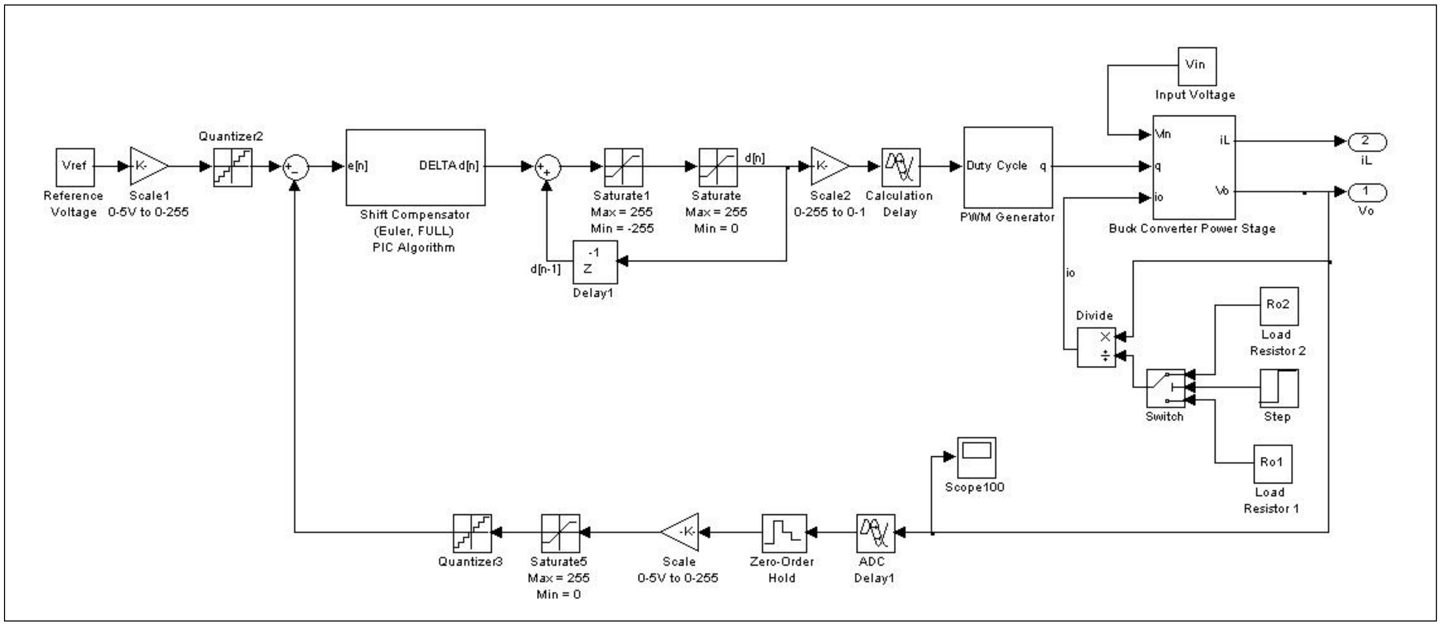

Figure 5.12: Closed-loop buck converter with discrete-time PID controller

saturation block was simply for debugging purposes.) The input of the PWM subsystem must be a value between 0 and 1, so the duty cycle is scaled accordingly. The Calculation Delay block is applied to model the delay in the microcontroller computations. This value is set to 32 μs to represent the entire programmable acquisition time discussed earlier.

5.3.1 Shift-Based Digital PID Controller in Simulink

The digital PID controller simulation model for the shift operator is presented in Figure 5.13. This Simulink model represents difference equation (3.2.5) being calculated consistent with the method used by the 8-bit microcontroller in the experiment. Each multiplication is saturated between -255 and 255 (this restriction was described earlier), and quantized in the case of any floating-point numbers that might arise. Two terms are summed together, saturated, and summed to the final term, followed by one last saturation.

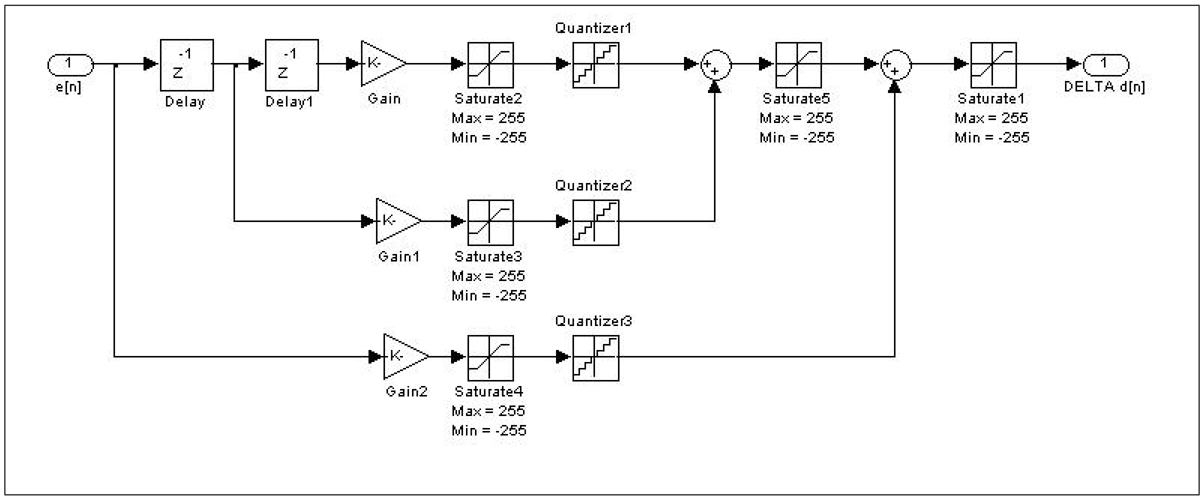

Figure 5.13: Simulation model of the shift-operator-based PID difference equation

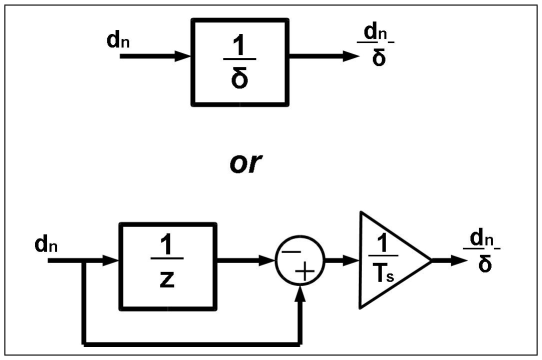

Figure 5.14: Block diagram of the delta operator

5.3.2 Delta-Based Digital PID Controller in Simulink

The closed-loop control system for this model is the same as the one shown in Figure 5.12 except the digital compensator subsystem is based on the delta operator rather than the shift operator.

A block diagram of a delta operator is provided by Figure 5.14. The delta operator used in this experiment is based on the definition of the causal (or backward) delta operator given in equation (2.1.2).

The form of the delta-operator-based difference equation was derived so as to "absorb" the sampling period, T_s, into the coefficients (equation (3.2.10)). Since this is the

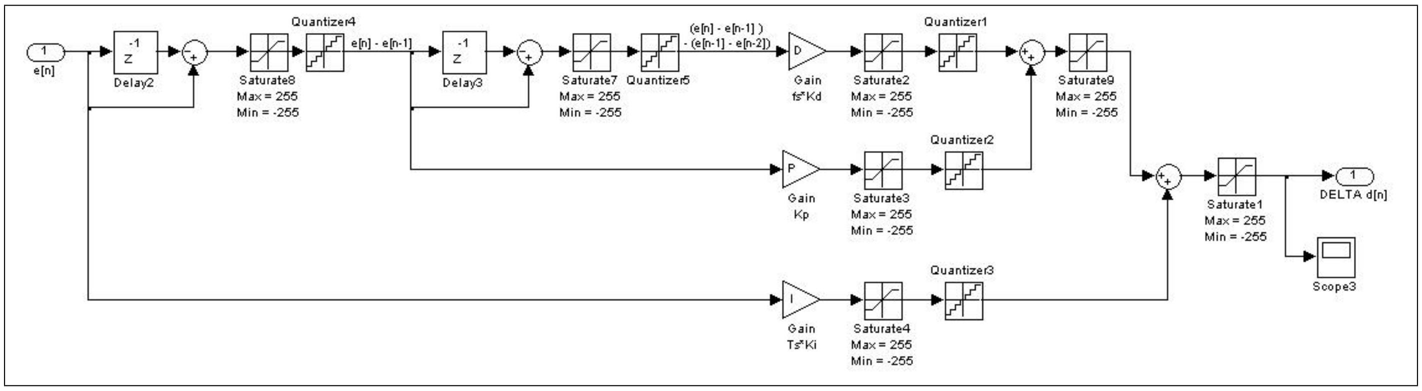

Figure 5.15: Simulation model of the delta-operator-based PID difference equation

programmed form, it must be considered when designing the simulation. The gain block in Figure 5.14, $1/T_s$, is not present in the difference equation block diagram since it is already accounted for in the coefficient gain block. The final form is incorporated in the delta-operator-based PID controller simulation model shown in Figure 5.15.

All the calculations are performed just as they are in the microcontroller's software. The same saturation and quantization effects are applied that have been to all the digital models thus far.

Chapter 6

Simulation and Experimental Results

6.1 Continuous PID Controller Designs

A variety of gains are selected for K_P, K_I, and K_D, to illustrate a variety of responses. No strict methodology, such a Ziegler-Nichols or Cohen-Coon, is used to tune the gains. A handful of responses with an assortment of overshoots, oscillations, and rise-times are desired to provide diverse scenarios to compare the performances of the two digital approaches. The plots in Figures 6.1 $\rightarrow$ 6.8 record the simulation results of the initial transients of the model shown in Figure 5.11 using the gains indicated in each figure description.

6.2 Discrete PID Controllers

The specific values of the continuous gain selections and their corresponding difference equation coefficients can be found in Table 6.1.

The shift and delta coefficients are found using equations (3.2.8) and (3.2.13) with $T_s = 49.6\ \mu$s. Recall that if any of the coefficients for the two digital operators exceeds $|255|$, it is truncated to either -255 or 255. Also, coefficient decimals are rounded off. If a gain is greater than 0, but less than 1, it is approximated by its closest inverse power of 2, i.e., 2^{-n}, multiplied by an integer.

As expected, the truncations experienced by the shift operator coefficients are more severe and numerous than the delta operator coefficients.

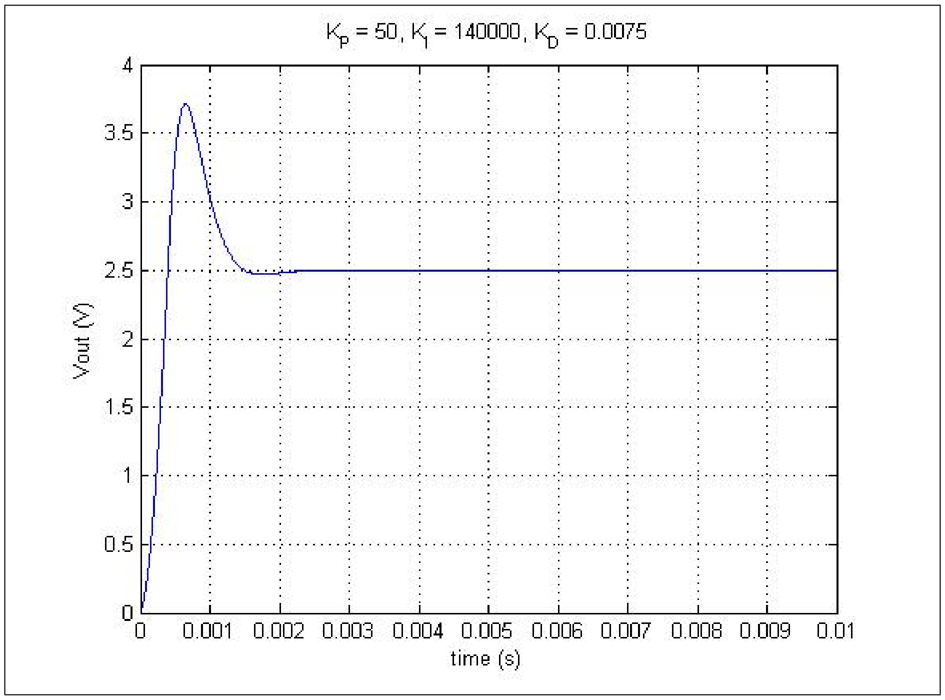

Figure 6.1: Initial transient; Continuous PID; K_P=50, K_I=140000, K_D=0.0075

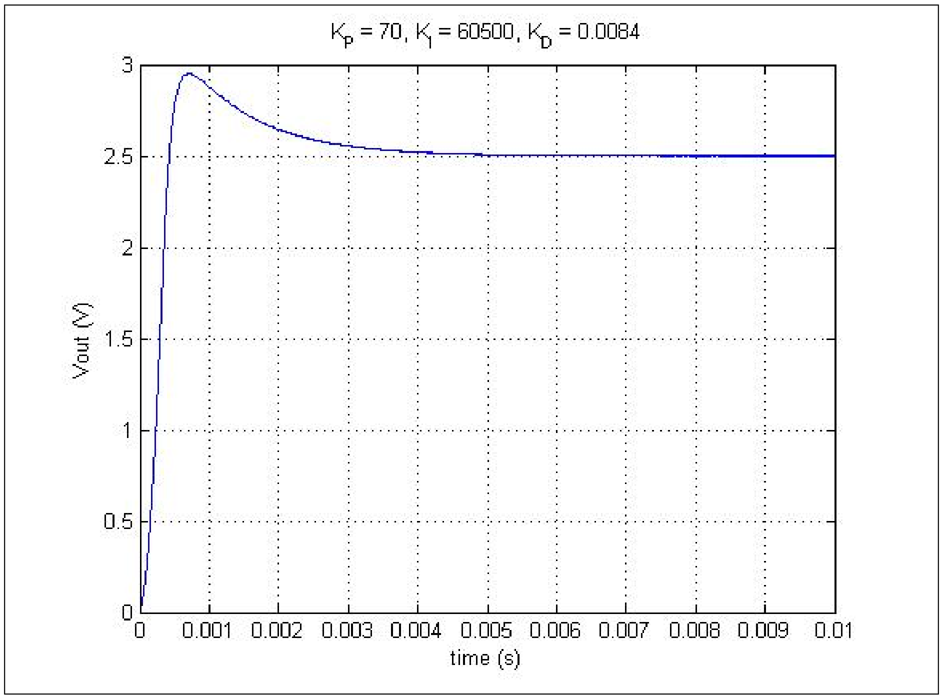

Figure 6.2: Initial transient; Continuous PID; K_P=70, K_I=60500, K_D=0.0084

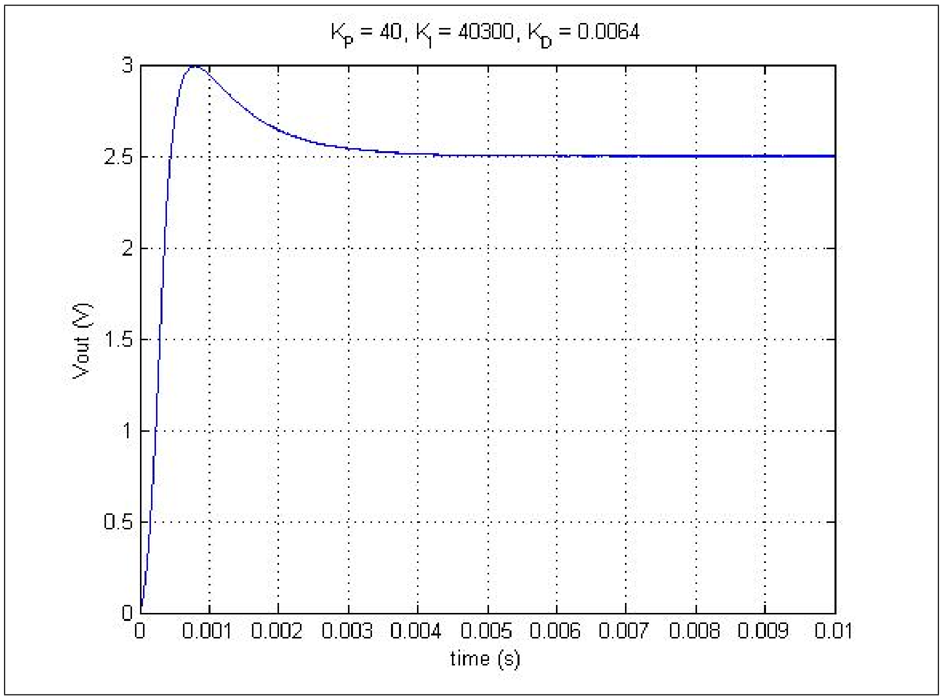

Figure 6.3: Initial transient; Continuous PID; K_P=40, K_I=40300, K_D=0.0064

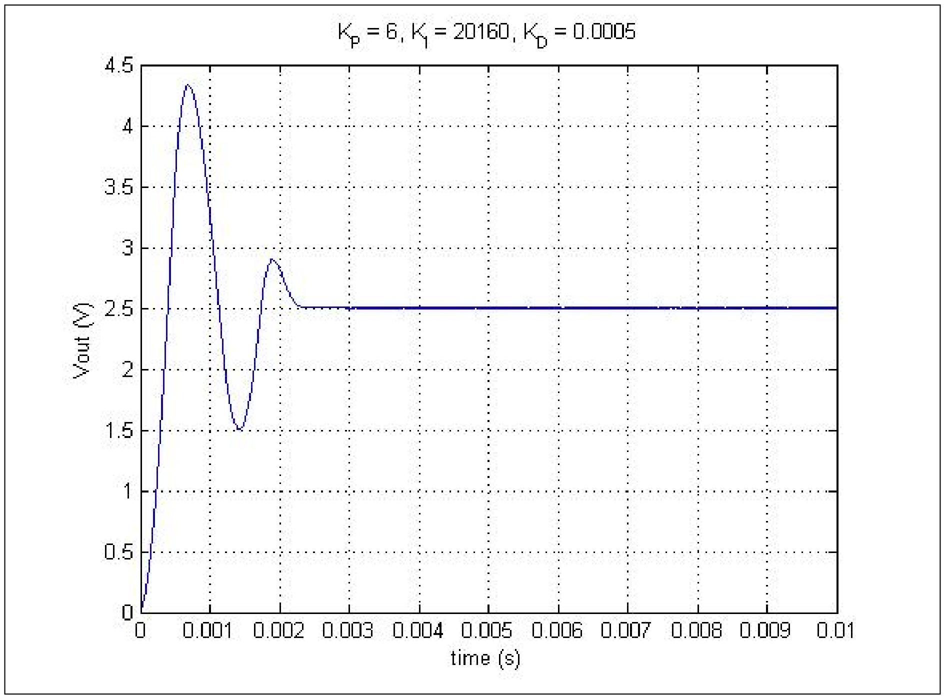

Figure 6.4: Initial transient; Continuous PID; K_P=6, K_I=20160, K_D=0.0005

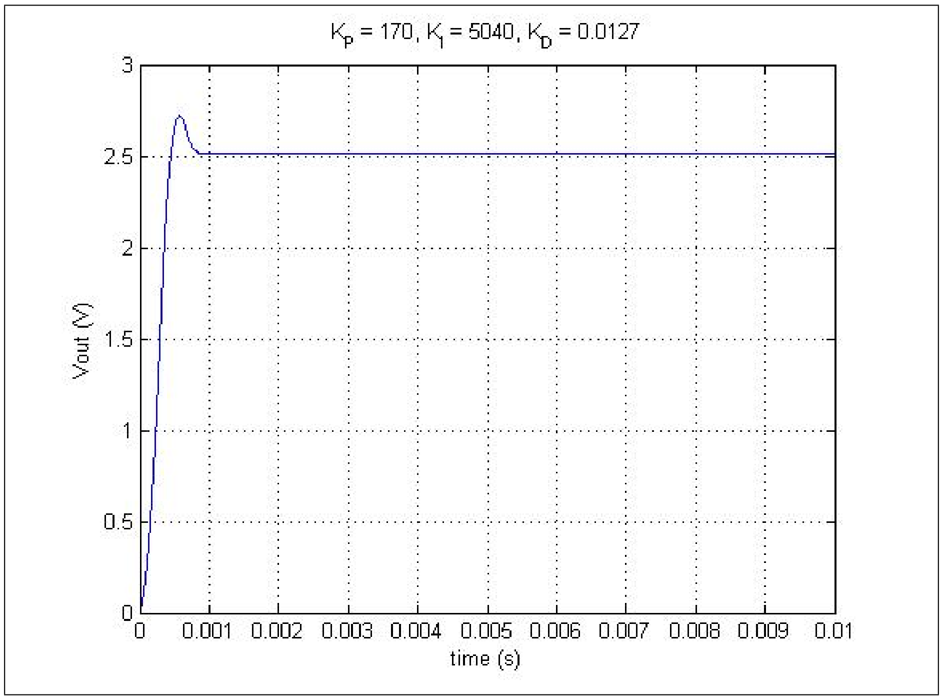

Figure 6.5: Initial transient; Continuous PID; K_P=170, K_I=5040, K_D=0.0127

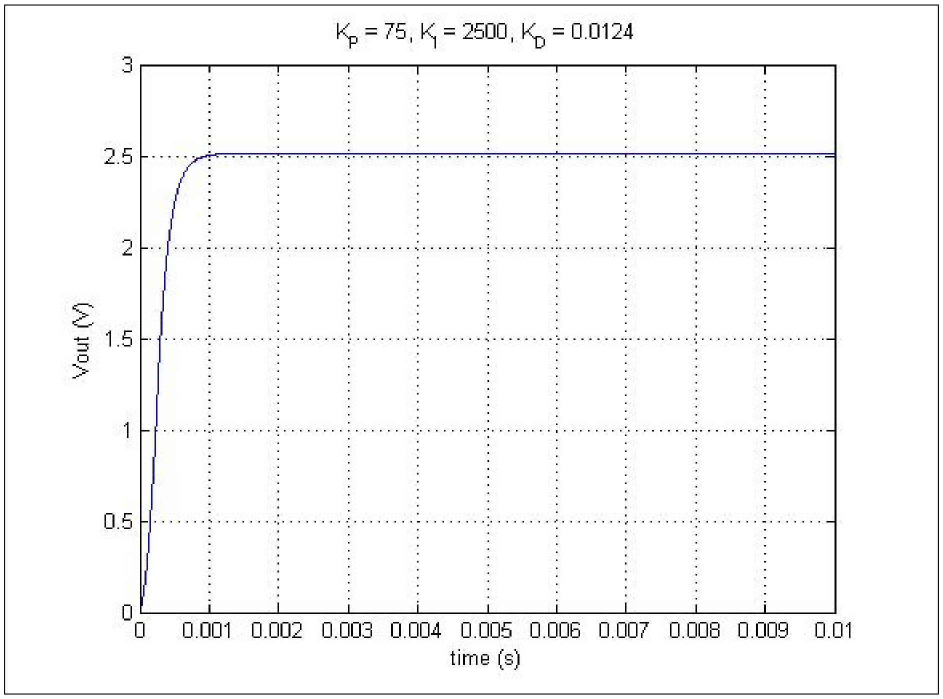

Figure 6.6: Initial transient; Continuous PID; K_P=75, K_I=2500, K_D=0.0124

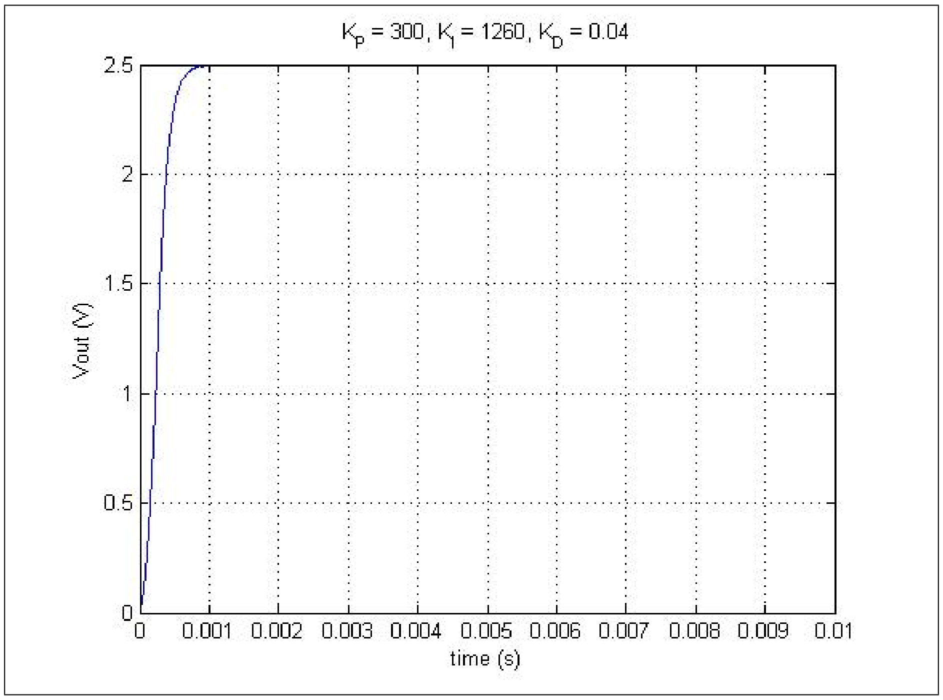

Figure 6.7: Initial transient; Continuous PID; K_P=300, K_I=1260, K_D=0.04

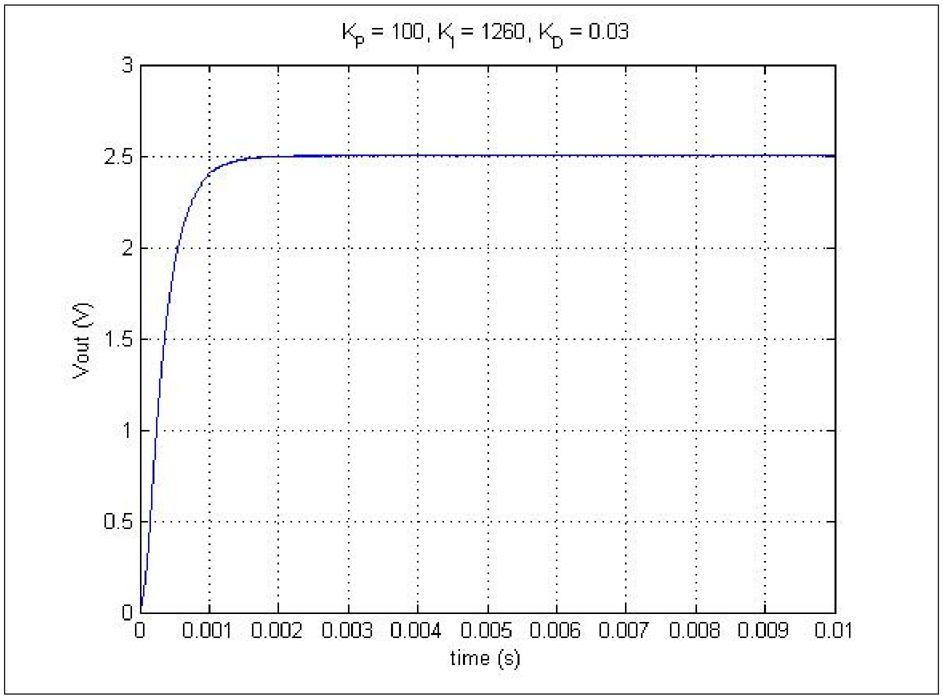

Figure 6.8: Initial transient; Continuous PID; K_P=100, K_I=1260, K_D=0.03

		Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8
Continuous Gains	K_P	50	70	40	6	170	75	300	100
	K_I	140000	60500	40300	20160	5040	2500	1260	1260
	K_D	0.0075	0.0084	0.0064	0.0005	0.0127	0.0124	0.04	0.03
Shift Coefficients	α_0	208	242	171	17	426	325	1107	705
	α_1	-352	-409	-298	-26	-682	-575	-1913	-1310
	α_2	151	169	129	10	256	250	806	605
Delta Coefficients	P	50	70	40	6	170	75	300	100
	I	7	3	2	1	0.25	0.124	0.0625	0.0625
	D	151	169	129	10	256	250	806	605

Table 6.1: Gains and coefficients used in the simulations and experiments for $T_s = 49.6\ \mu$s

6.2.1 Shift Operator Performance

The shift-operator-based PID controller transient results are displayed in the following plots (Figures 6.9 $\rightarrow$ 6.16). Each case is based on the the simulation model shown in Figure 5.12, using the compensator shown in Figure 5.13.

Every response based on the shift operator PID controller is marginally stable except one (Figure 6.12)! This poor performance was expected due to the extensive truncation that occurs on most coefficients and computations within the algorithm.

Notice the additional damping on the experimental results as compared to the simulation results. This outcome is precisely what the open-loop comparisons in Section 5.1 suggested would occur. The experiments also have excessive ripple characteristics. This finding is best attributed to the 0.5% error in the measurement equipment (see its reference manuel [21]), the nonlinear effects the buck converter will have on the measurement equipment, and the quantization error in the ADC and internal voltage reference.

These shift operator PID controller responses are unacceptable for any plant process application and bear no resemblance to the original analog PID controller simulations.

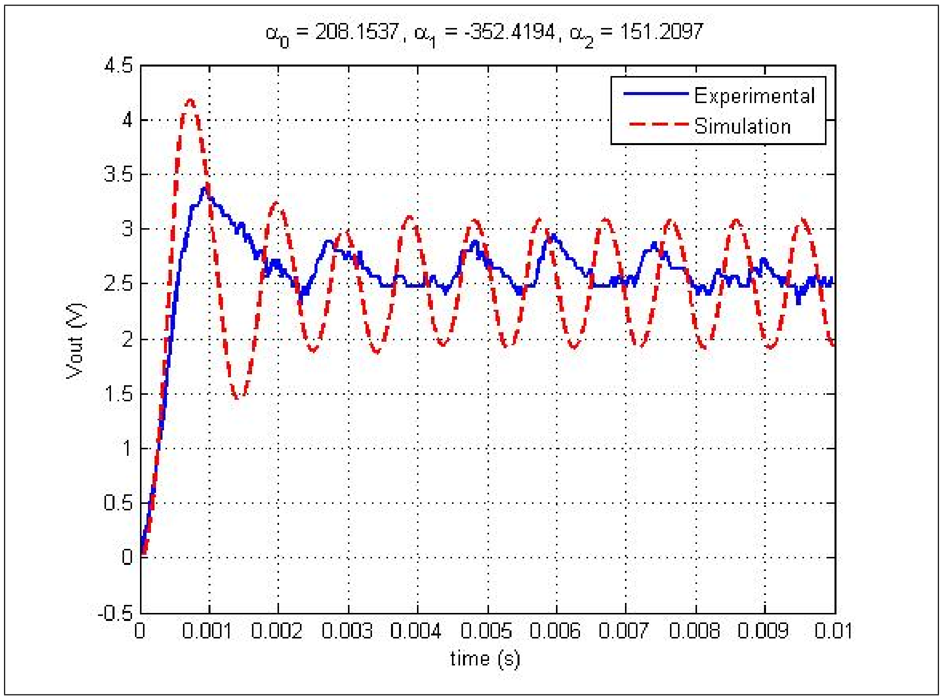

Figure 6.9: Initial transient; shift operator PID; K_P=50, K_I=140000, K_D=0.0075

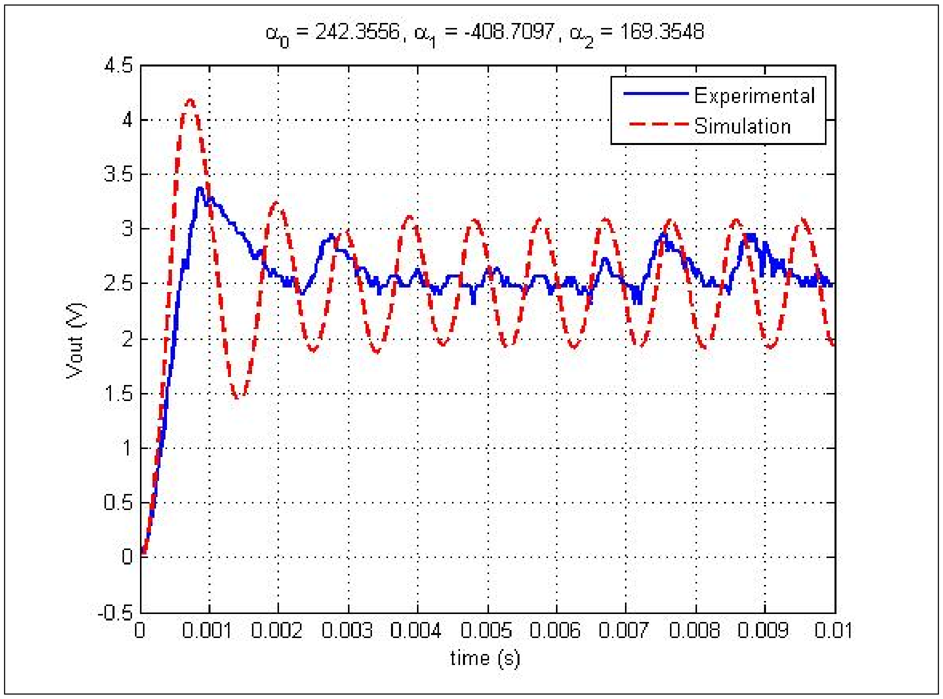

Figure 6.10: Initial transient; shift operator PID; K_P=70, K_I=60500, K_D=0.0084

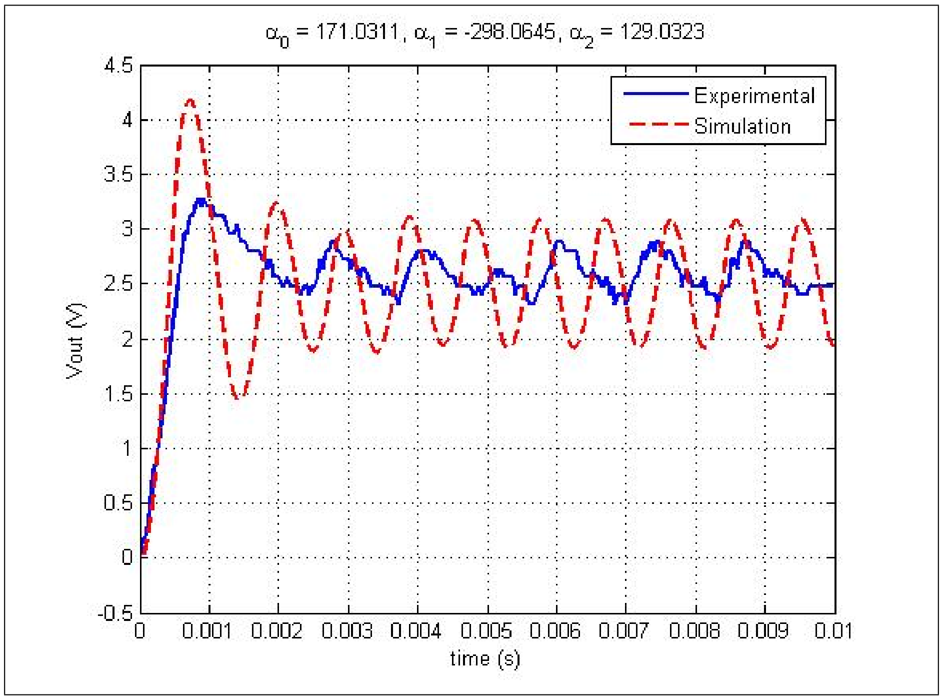

Figure 6.11: Initial transient; shift operator PID; K_P=40, K_I=40300, K_D=0.0064

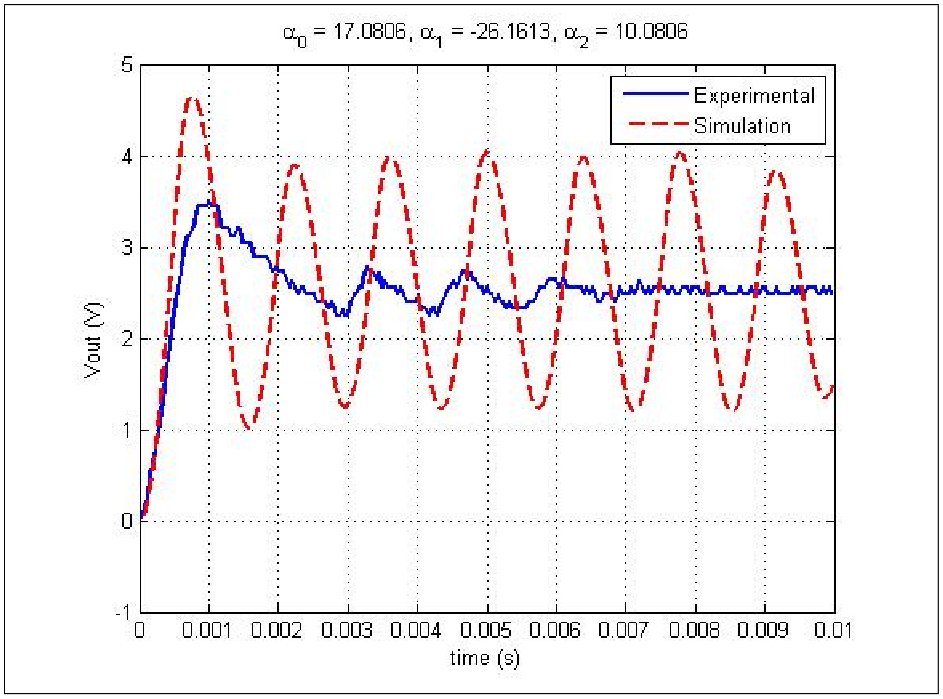

Figure 6.12: Initial transient; shift operator PID; K_P=6, K_I=20160, K_D=0.0005

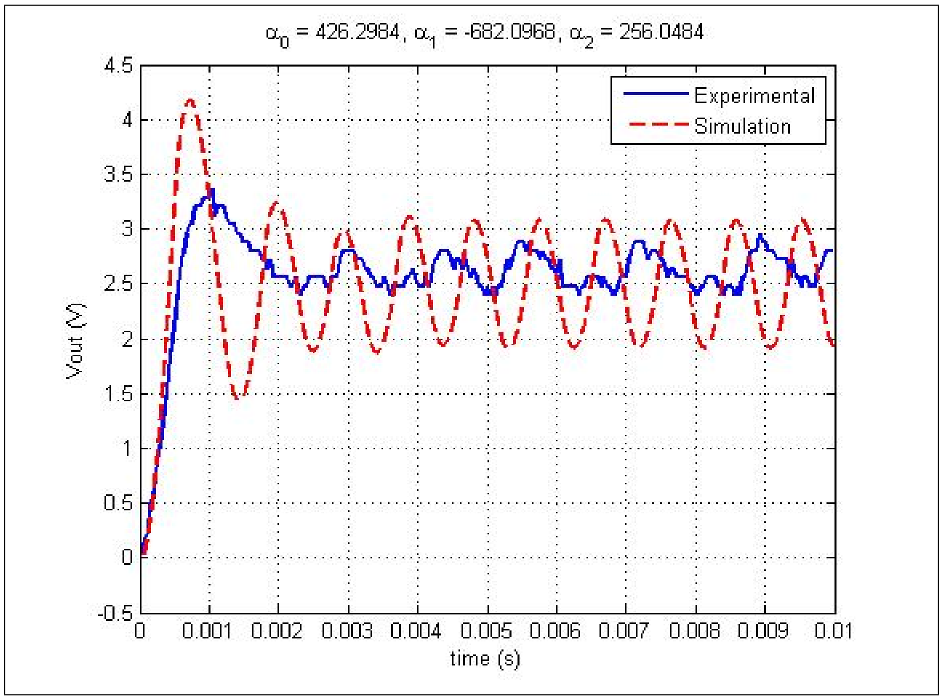

Figure 6.13: Initial transient; shift operator PID; K_P=170, K_I=5040, K_D=0.0127

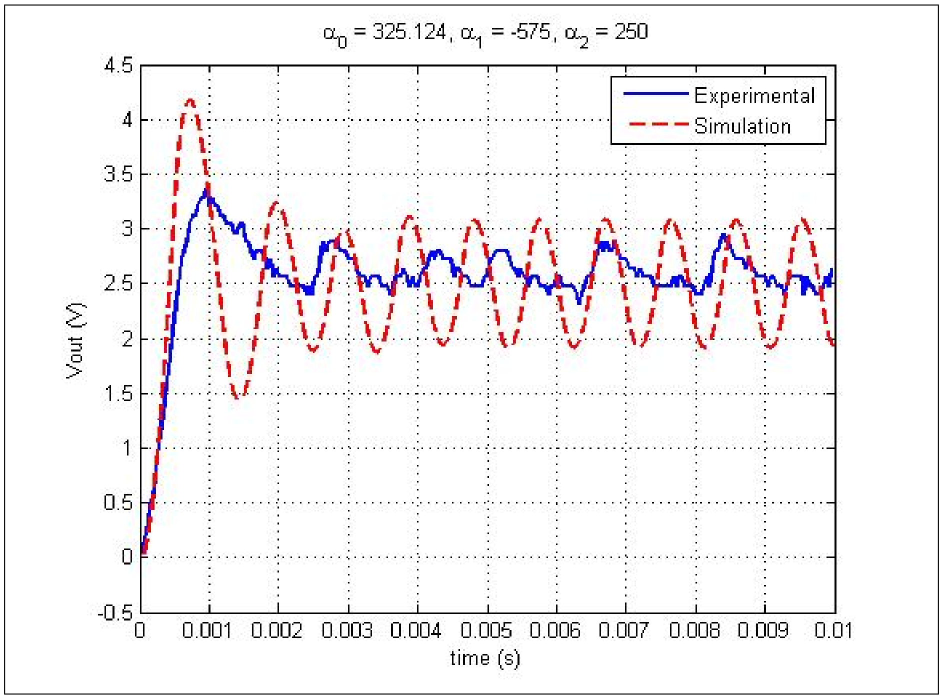

Figure 6.14: Initial transient; shift operator PID; K_P=75, K_I=2500, K_D=0.0124

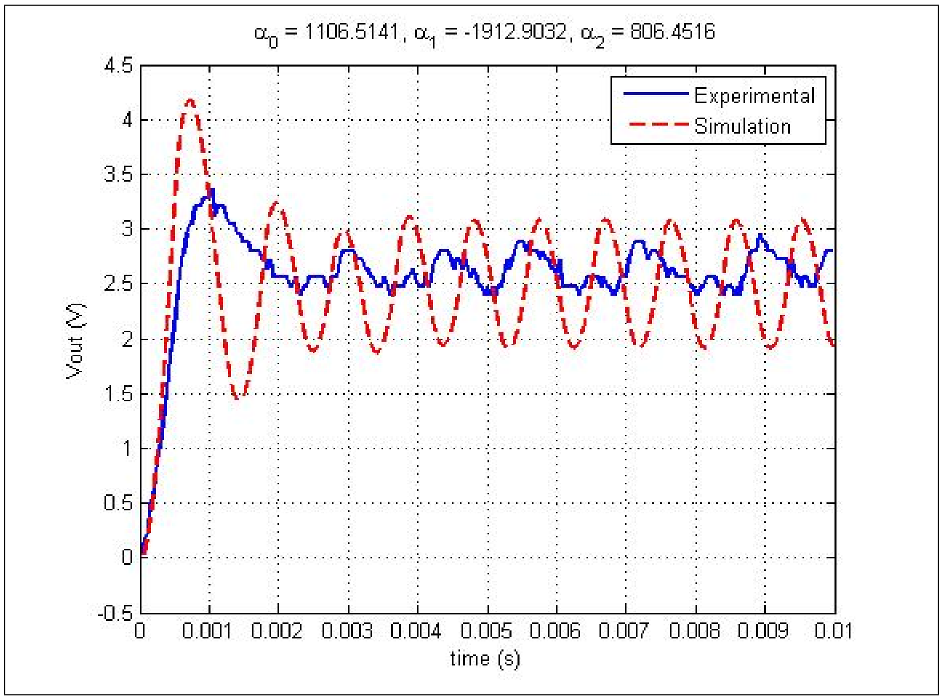

Figure 6.15: Initial transient; shift operator PID; K_P=300, K_I=1260, K_D=0.04

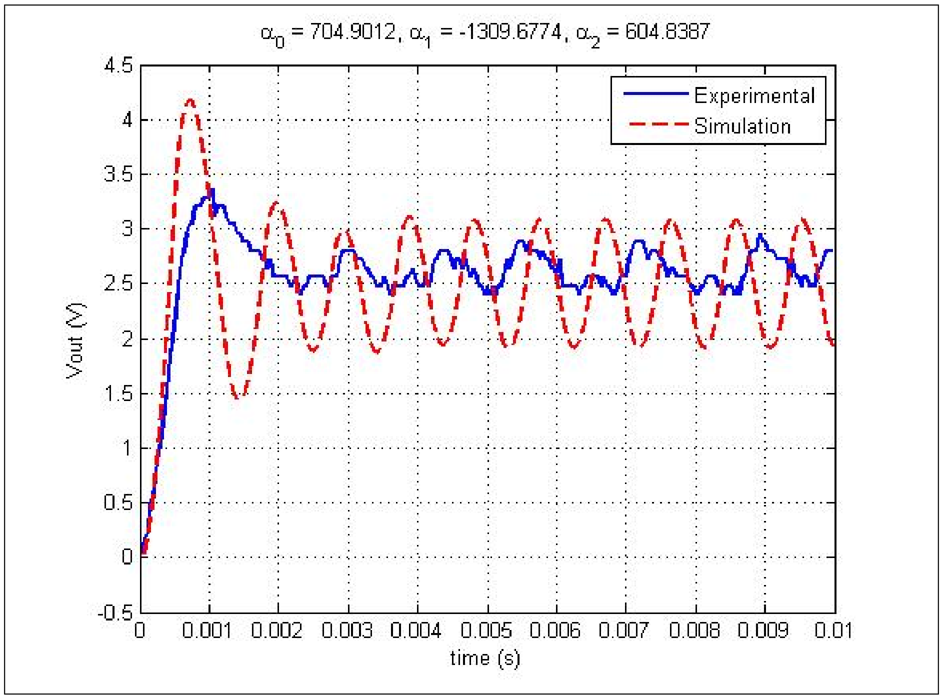

Figure 6.16: Initial transient; shift operator PID; K_P=100, K_I=1260, K_D=0.03

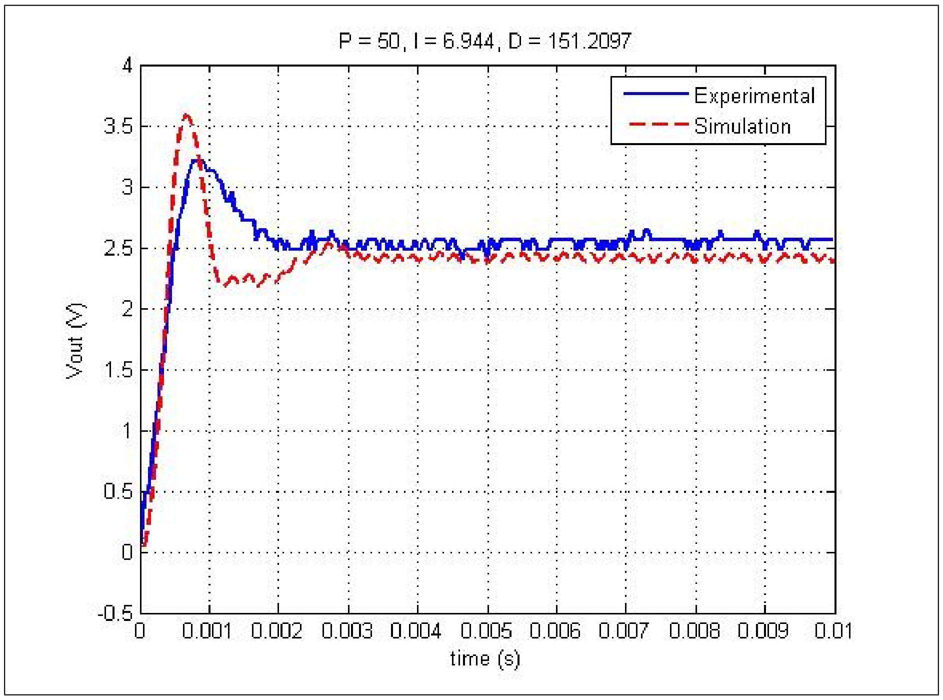

Figure 6.17: Initial transient; delta operator PID; K_P=50, K_I=140000, K_D=0.0075

6.2.2 Delta Operator Performance

The delta operator digital PID controller simulations used the same overall control model as the shift operator (Figure 5.12), except the compensator model is replaced by the one in Figure 5.15. The initial transient simulation and experimental plots follow in Figures 6.17 $\rightarrow$ 6.24.

The overall performance, especially the steady-state error and rise-time, and resemblance between the simulation and experiment degrade in the responses of Figures 6.21 - 6.24. Once again, this was predicted earlier during the discussion of approximating dividing by powers of two with "rotate rights." The precision lost in these cases is significant, especially considering the reference voltage is only 2.5 V.

The delta-operator-based PID controller demonstrates general superior performance over the shift-operator-based PID controller. The delta operator responses, in general, are

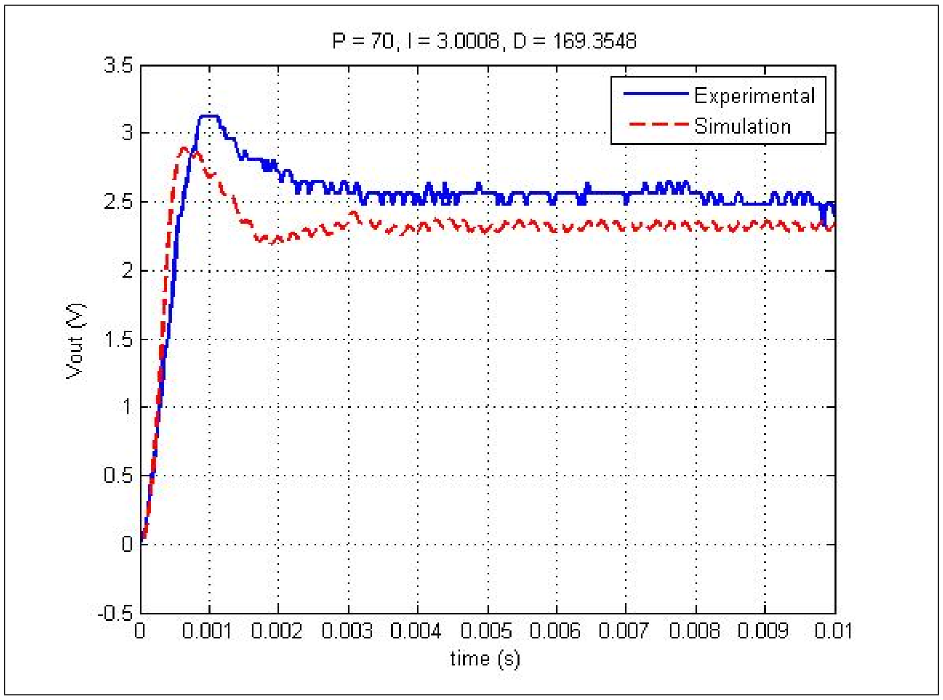

Figure 6.18: Initial transient; delta operator PID; K_P=70, K_I=60500, K_D=0.0084

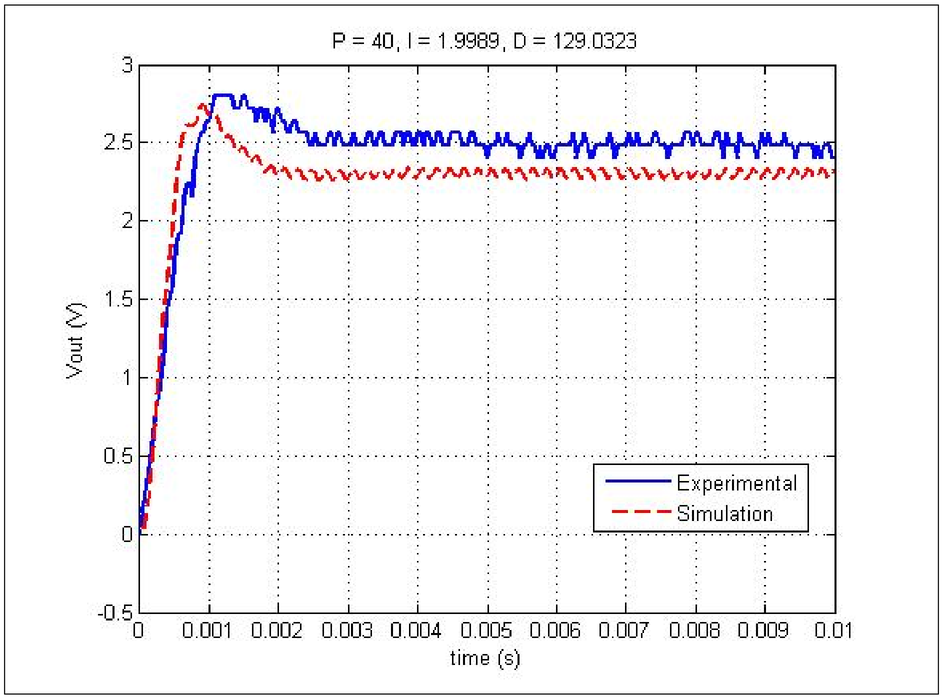

Figure 6.19: Initial transient; delta operator PID; K_P=40, K_I=40300, K_D=0.0064

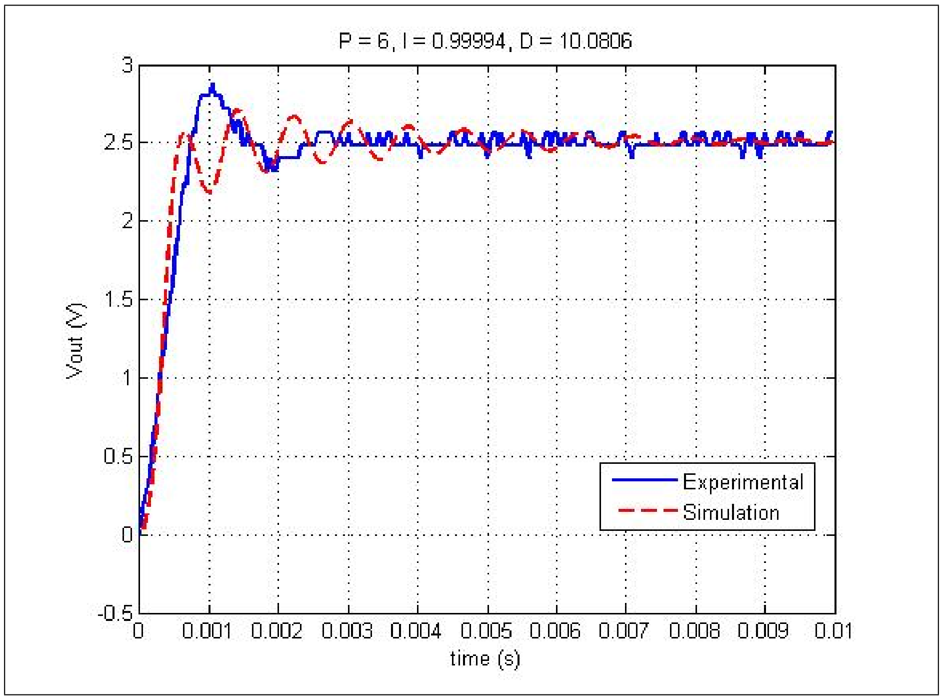

Figure 6.20: Initial transient; delta operator PID; K_P=6, K_I=20160, K_D=0.0005

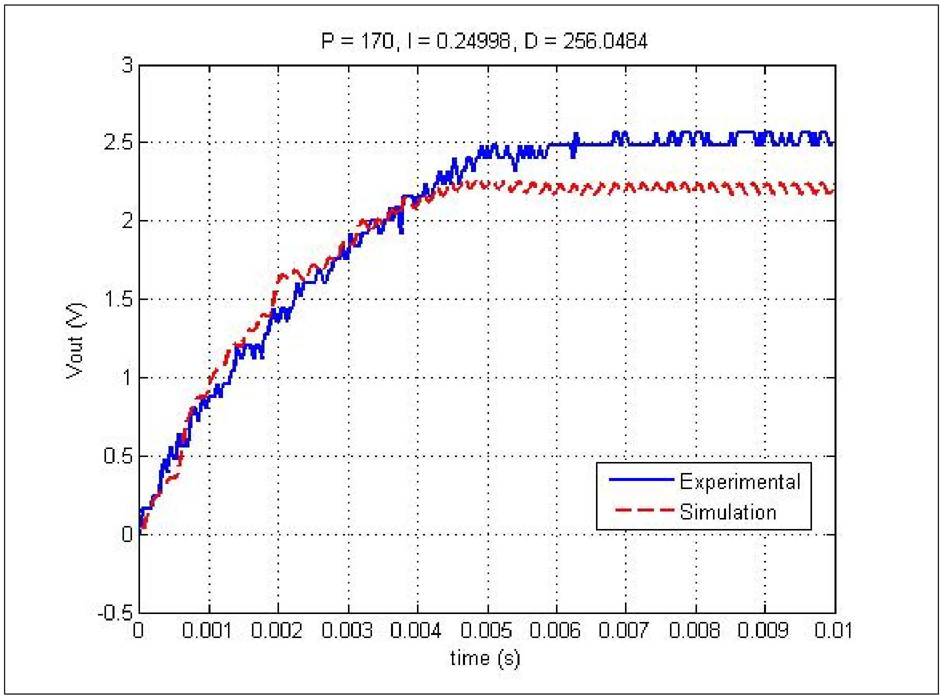

Figure 6.21: Initial transient; delta operator PID; K_P=170, K_I=5040, K_D=0.0127

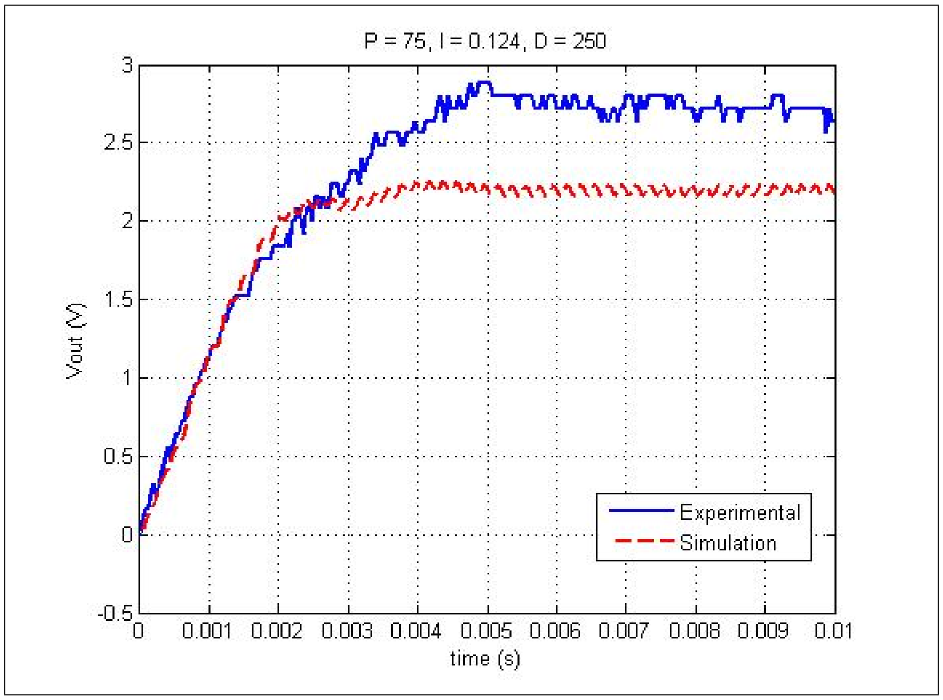

Figure 6.22: Initial transient; delta operator PID; K_P=75, K_I=2500, K_D=0.0124

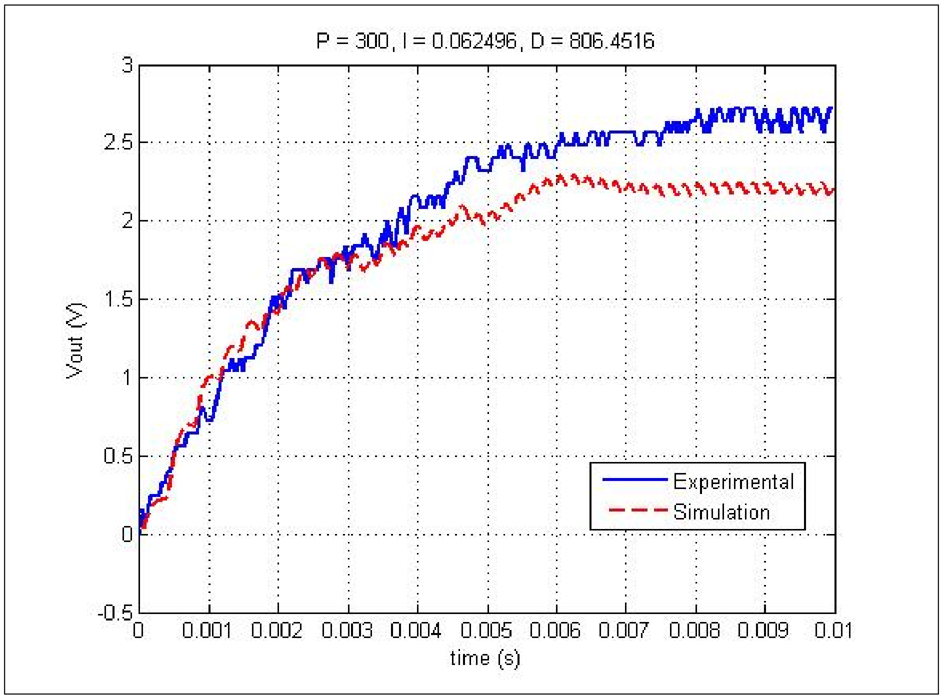

Figure 6.23: Initial transient; delta operator PID; K_P=300, K_I=1260, K_D=0.04

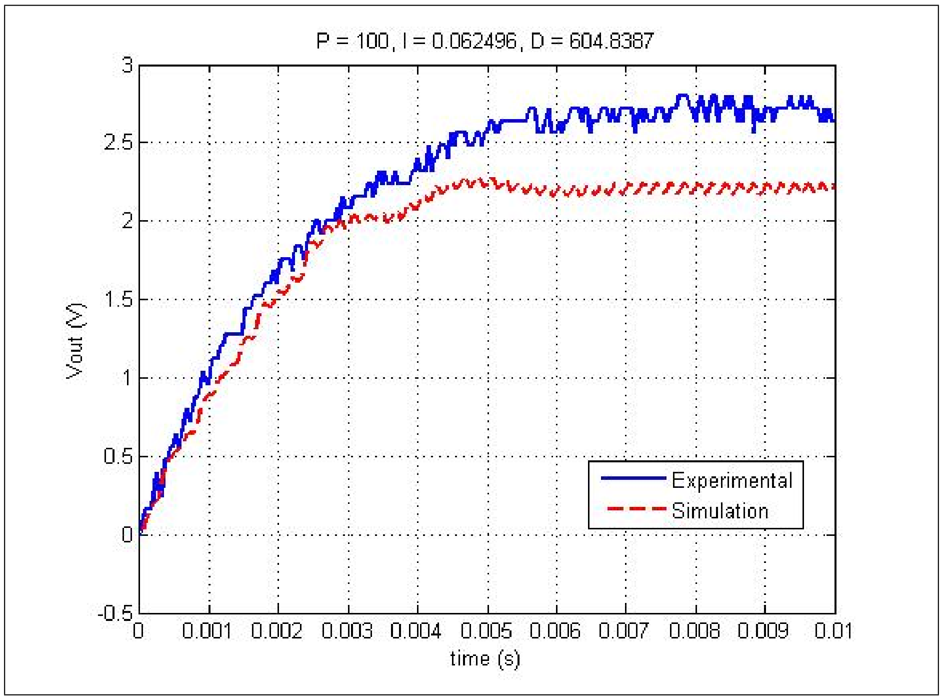

Figure 6.24: Initial transient; delta operator PID; K_P=100, K_I=1260, K_D=0.03

more stable and have less oscillation than the shift operator responses. There is also a favorable resemblance between the simulation and experimental responses, with some additional damping seen in the experimental system.

The advantage of the delta operator's difference equation is further supported if their experimental responses are compared to the corresponding continuous-time PID simulation responses (Figures 6.25 $\rightarrow$ 6.28). The algorithms that incorporated a division will not be displayed because their loss of accuracy decreases this parallel.

Most of these responses coincide well with each other, especially considering the known damping increase when operating in the "real world." With superior performance, increased correspondence between numerical simulations and experimental results, and prevailing resemblance to the continuous-time results, the delta operator proves to outperform the shift operator in discrete PID controllers under finite word-length conditions.

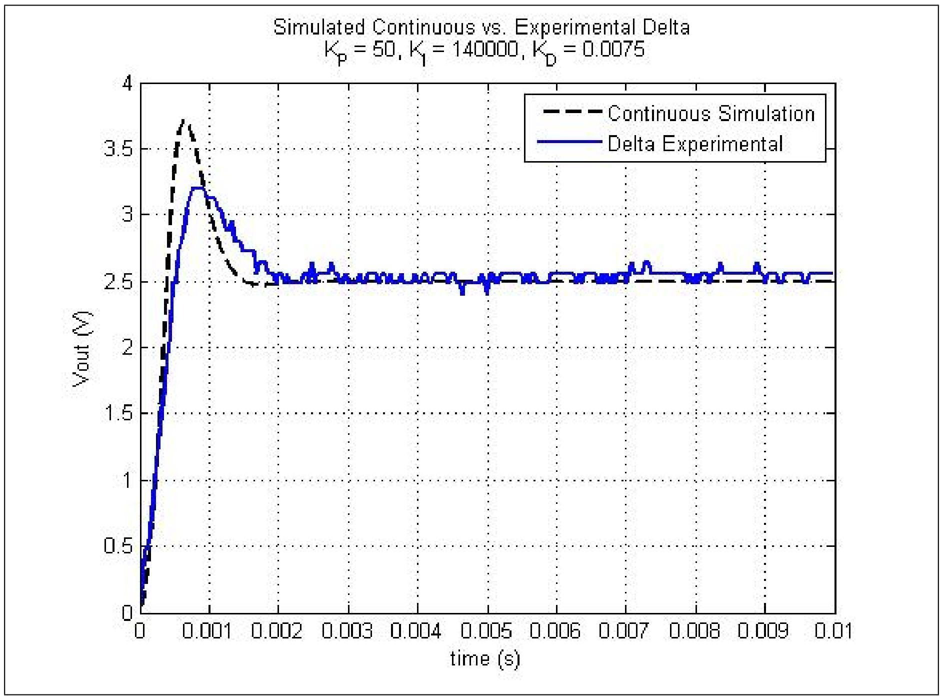

Figure 6.25: Initial transient; Continuous vs. Delta; K_P=50, K_I=140000, K_D=0.0075

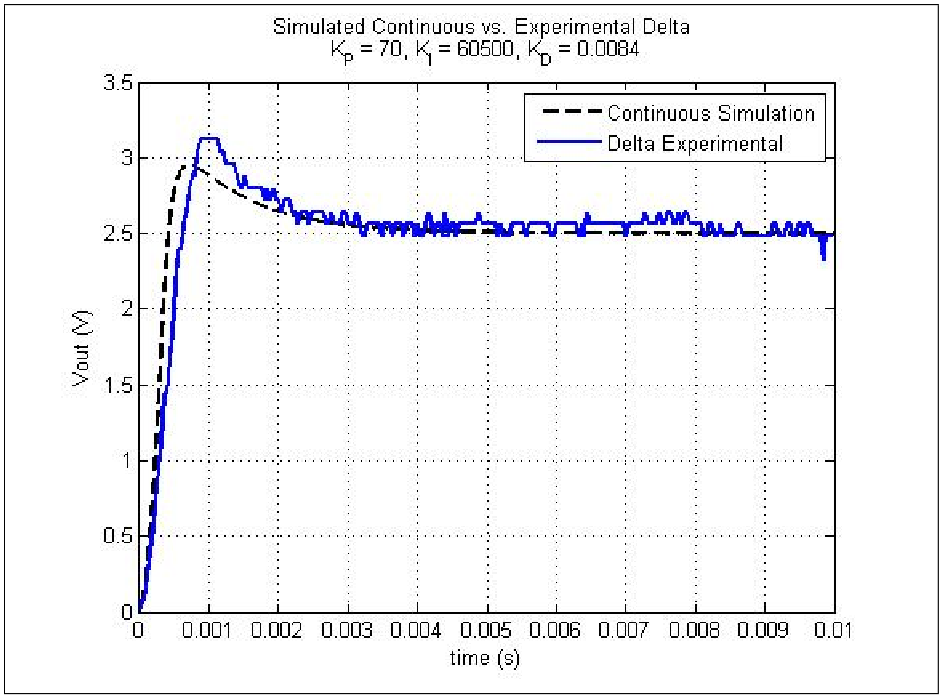

Figure 6.26: Initial transient; Continuous vs. Delta; K_P=70, K_I=60500, K_D=0.0084

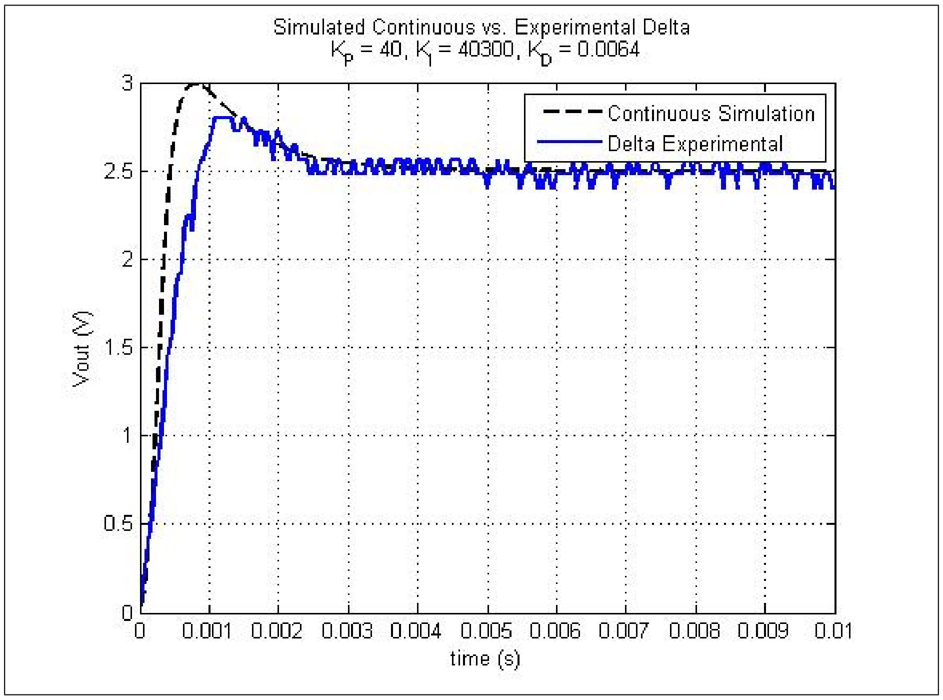

Figure 6.27: Initial transient; Continuous vs. Delta; K_P=40, K_I=40300, K_D=0.0064

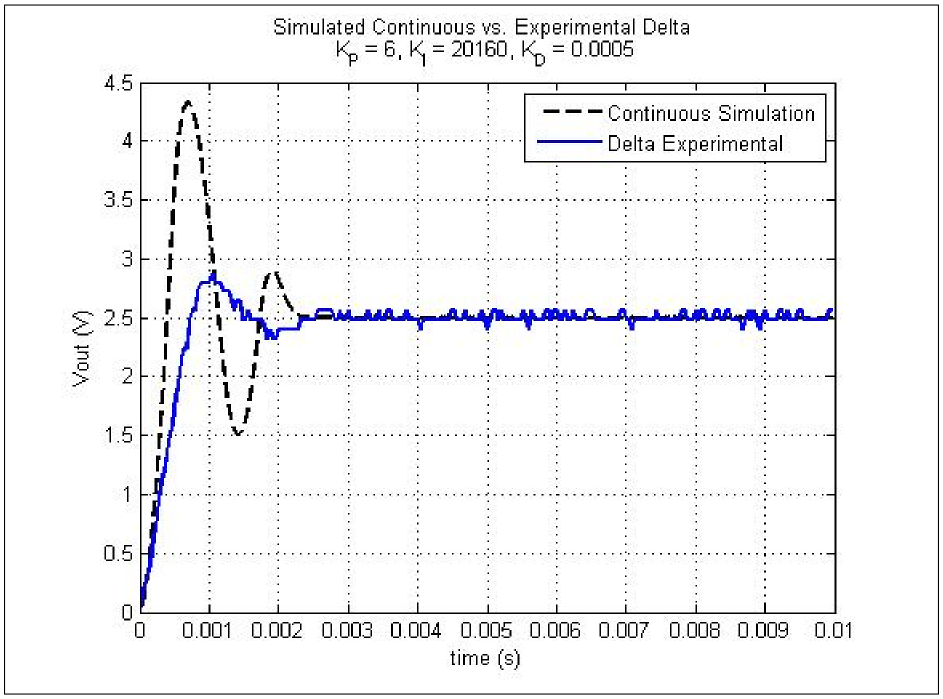

Figure 6.28: Initial transient; Continuous vs. Delta; K_P=6, K_I=20160, K_D=0.0005

Chapter 7

Conclusions and Potential Future Work

The work in this thesis explored the effects of finite word-length on digital controllers implemented using two different digital operators—the shift operator and the delta operator. After a mathematical analysis was performed, simulations and experiments were conducted using a common controls application: a PID-compensated buck converter.

The mathematical derivations, simulations, and experiments performed in this thesis suggest the same conclusion: the delta operator approach to parameterizing continuous controllers exhibits certain numerical and, therefore, performance advantages over the shift operator under finite word-length conditions. Though there are several challenges that fall under the category of "numerical," this thesis focused primarily on truncation that occurs due to the limited range of numbers that can be represented in microcontrollers. When using the approaches employed by this thesis to derive the discrete difference equations, the delta-operator-based coefficients are scaled less than the shift-operator-based coefficients. The additional scale experienced by the shift operator's difference equation makes it more susceptible to clipping caused by finite word-length, signifying a decrease in performance with respect to the delta operator.

The discovery of the delta operator's superior controller performance and correlation to continuous controllers could have benefits considering that most expertise and experience in controller design lies in the continuous realm. The results of this thesis suggest that one could reasonably design a digital controller using continuous techniques. A program could easily be written that would be used to calculate discrete coefficients that would correspond to the continuous-designed coefficients. According to this study, this approach would be far more likely to be successful if the digital controller were derived and implemented using the delta operator rather than the shift operator.

There is additional work that could be done to solidify these claims. Performing these experiments at additional sampling rates could reveal whether or not there are exceptions to the effects of finite word-length found in this thesis. Controllers other than a PID could be used for the simulations and experiments to determine if there are instances where the effects of finite word-length are enlarged or lessened between the delta and shift operators. Part of the mathematical analysis suggests that the delta operator could have an even more noticeable improvement over the shift operator if a controller is selected where the actual control signal could be solved, rather than the change in control signal (Subsection 3.1.2).

Another analysis could be performed using an alternative method to obtain the discrete representations of the continuous transfer function. One could use a different emulation technique, or derive the delta-based model directly from the continuous rather than the shift-based model.

Lastly, the increase in time that it takes to perform the delta-based PID calculation (Subsection 4.2.5) could point to a potential benefit of the shift operator over the delta operator if the limiting factor was sampling rate rather than word-length. Additional experiments would have to be arranged to confirm this hypothesis.

Bibliography

[1] Fujio Kurokawa, Masashi Okamatsu, Yuichi Sumida, Yasuhiro Mimura and Masahiro Sasaki, “A Novel Digital Control Method for DC-DC Converter,” 13th International Power Electronics and Motion Control Conference, 2008, pp.2434-2438.

[2] Aye Aye Mon, “Fuzzy Logic PID Control of Automatic Voltage Regulator System,” World Academy of Science, Engineering and Technology, 2009.

[3] Hiroshi Fujimoto, Yoichi Hori, “Advanced Digital Motion Control Based On Multirate Sampling Control,” IFAC 15th Triennial World Congress, 2002.

[4] Mario E. Salgado, Diego A. Oyarzun, “Two objective optimal multivariable ripple-free deadbeat control,” Department of Electronic Engineering, Universidad Tecnica Federico Santa Maria, 2007.

[5] Jun Wu, Sheng Chen, James F. Whidborne, Jian Chu, “Optimal realizations of floating-point implemented digital controllers with finite word length considerations,” International Journal of Control, Vol. 77, No. 5, 2004, pp. 427-440.

[6] Stephane Bibian, “Time Delay Compensation of Digital Control for DC Switchmode Power Supplies Using Prediction Techniques,” IEEE Transactions on Power Electronics, Vol. 15, No. 5, 2000, pp. 835-843.

[7] Richard H. Middleton and Graham C. Goodwin, “Digital Control And Estimation: A Unified Approach,” Prentice Hall, 1990.

[8] S. Chen, J. Wu, R.H. Istepanian, J. Chu, and J.F. Whidborne, “Optimizing stability bounds of finite-precision controller structures for sampled-data systems in the delta operator domain,” IEEE Proc., Control Theory Appl., 1999, Vol. 146, No. 6, pp. 517-526.

[9] J. Wu, S. Chen, G. Li, R.H. Istepanian and J. Chu, “Shift and delta operator realizations for digital controllers with finite word length considerations,” IEEE Proc.-Control Theory Appl, 2000, Vol. 147, No. 6, pp. 664-672.

[10] Gang Li and Michel Gevers, “Comparative Study of Finite Wordlength Effects in Shift and Delta Operator Parameterizations,” IEEE Trans. Autom. Control, 1993, Vol. 38, No. 5, pp.803-807.

[11] Richard H. Middleton and Graham C. Goodwin, “Improved Finite Word Length Characteristics in Digital Control using Delta Operators,” IEEE Transactions On Automatic Control, Vol. AC-31, NO. 11, November 1986.

[12] Qi Feng, "Design and Implementation of Posicast-Based Feedback Controllers for Buck and Boost Converters," Master Thesis, Electrical and Computer Engineering Department, Auburn University, 2003.

[13] Ralph Raymond Boudreaux, Jr., "Digital Control of a Switchmode Power Converter Using an Embedded Microcontroller," Master Thesis, Electrical and Computer Engineering Department, Auburn University, 1997.

[14] Tooran Emami, "A Unified Procedure for Continuous-Time and Discrete-Time Root Locus and Bode Design," Master Thesis, Department of Electrical Engineering, Wichita State University, 2006.

[15] Liping Guo, "Design and Implementation of Digital PID Controllers for Buck and Boost Converters," Master Thesis, Electrical and Computer Engineering Department, Auburn University, 2001.

[16] Ned Mohan, "First Course on Power Electronics," 2009 Edition, MNPERE, Minneapolis, MN, 2009, pp. 3-1 - 3-5.

[17] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010264.

[18] Microchip PIC18F1220/1320 Data Sheet, Microchip Technology, Inc., USA, 2004.

[19] International IRFZ34N HEXFET Power MOSFET Data Sheet, International IOR Rectifier 1997.

[20] ON Semiconductor MBR3045PT Switchmode Power Rectifier Data Sheet, Semiconductor Components Industries, LLC, 2000.

[21] Tektronix P6139A 10X Passive Probe Instructions, Tektronix, Inc.