Summer 2008

Parameter estimation of systems with deadzone and deadband and emulation using xPC Target

Jeffrey James Lentz

PARAMETER ESTIMATION OF SYSTEMS WITH DEADZONE AND DEADBAND AND EMULATION USING xPC TARGET

by

JEFFREY JAMES LENTZ

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

2008

Approved by

Robert G. Landers, Advisor
K. Krishnamurthy
Jagannathan Sarangapani

PUBLICATION THESIS OPTION

This thesis consists of the following two articles that have been submitted for publication as follows:

Pages 1-64 are intended for submission in a journal regarding automated controls such as *Control Engineering Practice*.

Pages 65-98 are intended for submission in a journal regarding emulation hardware such as *Electronic Measurement and Instruments*.

ABSTRACT

The first paper presents a new approach for online parameter estimation using multiple recursive least squares estimations implemented simultaneously to determine system model parameters, as well as a deadzone and/or deadband. The online adaptive estimation scheme was verified in simulation using MATLAB Simulink and verified experimentally for a DC motor driven cart, an electro-hydraulic pilot valve system, and a free cart loosely coupled to a DC motor driven cart by a pin that fits loosely in a slot. The results for the DC motor driven cart and electro-hydraulic pilot valve predicted the output within 10% in almost every case and as low as 2-3%. The estimation of the velocity of the free cart loosely coupled to the DC motor cart was within 3% for some cases; however, it was highly sensitive to input voltage frequency.

The second paper demonstrates the use of the Mathworks xPC Target environment for validation of a control system and emulation of a physical system using real-time code auto-generated from a simulation environment. A Master/Slave control system is developed for a hydraulic test stand. The Master and Slave Electronic Control Units (ECUs) are emulated using two target PCs running the xPC Target kernel communicating with each other over a Controller Area Network. The emulated and simulated results matched perfectly. Then the emulated Master ECU is used to control the hydraulic test stand by sending current commands and receiving pressure sensor data from the Slave ECU. The task execution time of the emulated Master ECU was the same regardless of whether it was controlling the emulated Slave ECU or the actual Slave ECU. The accuracy of the emulation is shown to be only limited by the accuracy of the hydraulic system plant model.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Landers for his guidance and patience. I am grateful to my committee members Dr. Krishnamurthy and Dr. Sarangapani for their interest in my work. Many thanks are due to my family, friends, coworkers, and fellow students. Also, this work would not have been possible without the generous donation of money and equipment from Caterpillar, Inc.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PAPER

1. PARAMETER ESTIMATION OF SYSTEMS WITH DEADZONE AND DEADBAND

J.J. Lentz, R.G. Landers, and K. Krishnamurthy

Department of Mechanical and Aerospace Engineering

Missouri University of Science and Technology

1870 Miner Circle

Rolla, Missouri 65409-0050

{Jeffrey.J.Lentz, landersr, kkrishna}@mst.edu

1.1. ABSTRACT

A new approach is presented for online parameter estimation using multiple recursive least squares (*RLS*) estimations implemented simultaneously to determine system model parameters, as well as a deadzone and/or deadband. Initial guesses of the deadzone parameters that are larger than the actual parameters are required in some cases. The system input and output signals dictate which of the *RLS* estimation schemes to utilize at each sample period. Comparing each of the *RLS* estimations, the deadzone and/or deadband are determined. The online adaptive estimation scheme was simulated using MATLAB Simulink to verify convergence. It was also verified experimentally for a DC motor driven cart, an electro-hydraulic pilot valve system, and a free cart loosely coupled to a DC motor driven cart by a pin that fits loosely in a slot. The results for the

DC motor driven cart and electro-hydraulic pilot valve were very good, predicting the output within 10% in almost every case, and as low as 2-3%. The estimation of the velocity of the free cart loosely coupled to the DC motor cart was within 3% for some cases; however, it was highly sensitive to input voltage frequency. In the case of a high frequency, the two carts generated high impact forces not modeled by the simple model used.

1.2. KEYWORDS

Deadzone, Deadband, Recursive Least Squares, Electro-Hydraulics, Parameter Estimation

1.3. INTRODUCTION

1.3.1. Need For Online Parameter Estimation. Many mechanical systemshave nonlinearities such as deadzone or deadband. Deadzone is typically present as a result of static friction, spring pretension, or overlapping parts and create a region where the input has no effect on the output [1, 2]. Deadband can result from static friction or from improper mating of parts (e.g., gear backlash) and creates a region where the output remains unchanged upon reversal of direction [1]. The parameters of these nonlinearities are rarely known and, if left unaccounted for, can degrade controller performance. Frequently in practice, a calibration routine will be conducted and deadband and deadzone parameters are determined offline. Then in the control software, the inverses of the deadzone and deadband are implemented using the predetermined static values [3]. While this is better than ignoring the nonlinearities entirely, any change in the nonlinearity is not accounted for in the control software. Factors such as wear, corrosion, lubrication, and temperature change over time and affect the deadzone and deadband regions. Also, cost of the calibration test itself is often prohibitive. Using an algorithm that can eliminate the calibration test and constantly update the nonlinear parameters is advantageous in many applications.

This paper presents an online parameter estimation scheme that estimates system model parameters as well as deadzone and/or deadband parameters. This is verified experimentally for a DC motor driven cart, an electro-hydraulic pilot valve system, and a free cart loosely coupled to a DC motor driven cart by a pin that fits loosely in a slot.

1.3.2. Deadzone. Deadzone will be defined for the purpose of this paper as a range of inputs for which the output remains unchanged [1]. Outside that range there is a linear relationship between the deadzone input *v*(*k*) and deadzone output *u*(*k*). The deadzone is shown in Figure 1.1 and is defined by

$$u(k)=\begin{cases} v(k)-c_l & \mathit{for} \quad v(k)<c_l \\ 0 & \mathit{for} \quad c_l \le v(k) \le c_r \\ v(k)-c_r & \mathit{for} \quad v(k)>c_r \end{cases} \tag{1}$$

where $c_r \ge c_l$. Some papers [2,3] define additional terms to account for any differences in the slope between positive and negative inputs. In this paper the deadzone is always considered before or after a linear system. Separate transfer functions are used to define a piecewise linear system: one for each side of the deadzone. Any terms added to equation (1) to account for slope differences will be accounted for in the steady-state gains of these transfer functions.

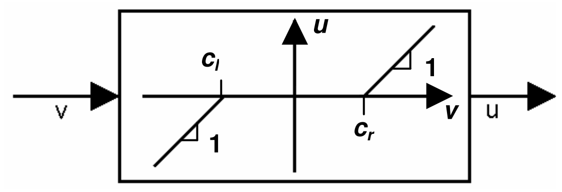

Figure 1.1. Block Diagram of a Deadzone Nonlinearity.

The deadzone may occur at the input or output of an otherwise linear system. For the DC-motor driven cart that will be studied later in this paper, the Coulomb friction results in a deadzone. For this case the system is best modeled as a linear transfer function with a deadzone at the output. The pressure of an electro-hydraulic pilot valve, which will also be studied later, contains a deadzone resulting from spring pretension and an overlap in the spool's land over the output port. For this case, the system is best modeled as a linear transfer function with a deadzone at the input.

1.3.2.1. Deadzone Literature Review. Many articles have been written regarding the online estimation of deadzone and deadband parameters, or the on-line estimation of plant dynamics. However, to the authors' knowledge, no work has incorporated the estimation of multiple non-linearities and plant dynamics simultaneously.

Tao and Kokotovic [2] develop projection methods to determine a symmetric deadzone on an input signal. They show improved controllability in simulation using a deadzone inverse. Er-Wei Bai [3] proves convergence of Tao and Kokotovic's work [2], but assumes the deadzone output is measurable. Wang *et al.* [4] proposes a piecewise and intuitive method for controlling a deadzone without a deadzone inverse. His method assumes equal slopes on each side of the deadzone and prior knowledge of upper and lower deadzone bounds. Ibrir [5] develops a method for an adaptive control algorithm for a symmetric input deadzone without prior knowledge of deadzone bounds. Also, in [5] a method is developed for an adaptive control algorithm for a non-symmetric input deadzone that requires prior knowledge of deadzone bounds using Lyapunov and Algebraic Riccati Equations. Lewis *et al.* [6] derives a Fuzzy Logic control scheme to estimate a deadzone and validates it on a CNC machine tool assuming an input deadzone and that linear plant dynamic terms are known. Jang [7,8] uses a similar technique to estimate an input deadzone for a DC motor and for an *XY* positioning table. Xu *et al.* [9,10,11] uses iterative learning control (*ILC*) to estimate time-varying and state-dependent deadzone for controlling a piezoelectric motor. However, *ILC* can only be used to control actuators performing repetitious motions.

1.3.2.2. Deadzone Example 1. The following is an example of a system with deadzone as shown in Figure 1.2. The input is $v(k) = 700\sin(kT)$. The plant is a linear first order system with a deadzone at the input. The linear transfer function has a sample period of $T = 0.005$ and is

$$\frac{Y(z)}{U(z)} = \frac{0.1}{z - 0.96} \tag{2}$$

The deadzone has the following parameters: $c_l = -300$ and $c_r = 200$. Thus $u(k)$ is

$$u(k) = \begin{cases} v(k) + 300 & for \quad v(k) < -300 \\ 0 & for \quad -300 \le v(k) \le 200 \\ v(k) - 200 & for \quad v(k) > 200 \end{cases} \tag{3}$$

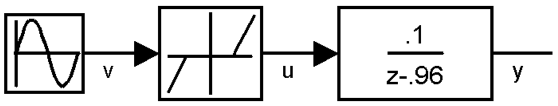

Figure 1.2. Block Diagram of a Plant with Input Deadzone and a Sinusoidal Input.

Figure 1.3 shows the signals plotted with respect to time and Figure 1.4 shows the input-output relationship. The input-output relationships are shown in comparison to a simulation of the transfer function from equation (2) in the absence of the deadzone. The elliptical shape in Figure 1.4 is the result of phase lag. The slower the time constant, the further it deviates from the line $y = Ku$ where K is the steady-state gain. The portion of the line for the output without deadzone that lies inside the ellipse results from the zero initial condition.

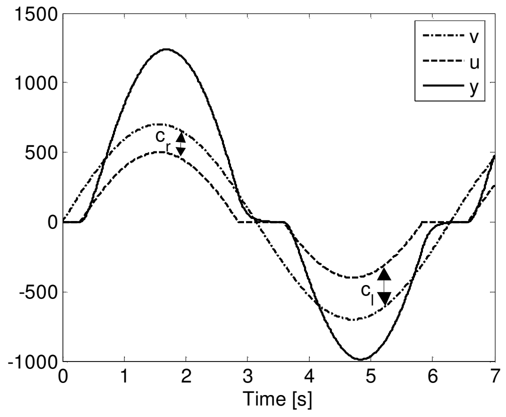

Figure 1.3. Results for the System Defined By Equations (2) and (3) with $v(k)$ = 700sin(kT).

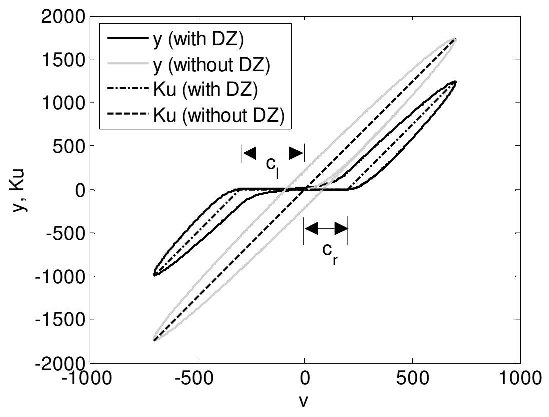

Figure 1.4. *y*(*k*) Versus *v*(*k*) for the Transfer Function in Equation (2) with and without Deadzone Defined in Equation (3) and *v*(*k*) = 700sin(*kT*).

1.3.2.3. Deadzone Example 2. The following is an example of a system with deadzone as shown in Figure 1.5. The input is $v(k) = 300\sin(kT)$. The plant is a first order linear system with a deadzone at the output. The linear portion is defined by equation (2). The deadzone is defined by: c_l = -300 and c_r = 200. Thus $y(k)$ is

$$y(k) = \begin{cases} w(k) + 300 & for \quad w(k) < -300 \\ 0 & for \quad -300 \le w(k) \le 200 \\ w(k) - 200 & for \quad w(k) > 200 \end{cases} \tag{4}$$

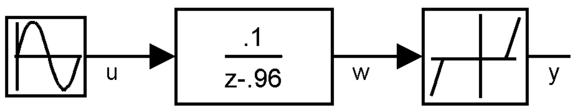

Figure 1.5. Block Diagram of a Plant with an Output Deadzone and a Sinusoidal Input.

The results are shown in Figure 1.6 and Figure 1.7. The resulting plots for an output deadzone have roughly the same shape as the plots for an input deadzone as shown in Figure 1.3 and Figure 1.4. The difference is that the output decays asymptotically to zero when the deadzone is at the input and the output goes to zero without decay when the deadzone is at the output.

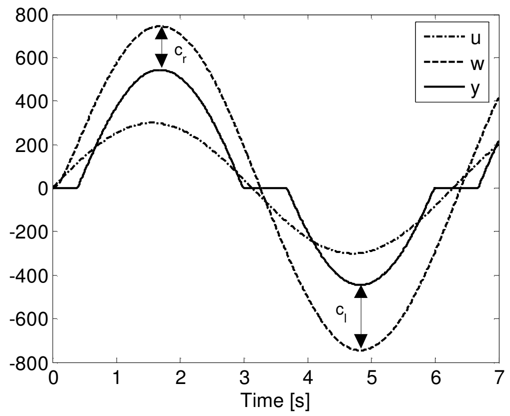

Figure 1.6. Simulated Results for System Defined by Equations (2) and (4) and $u(k)$ = 700sin(kT).

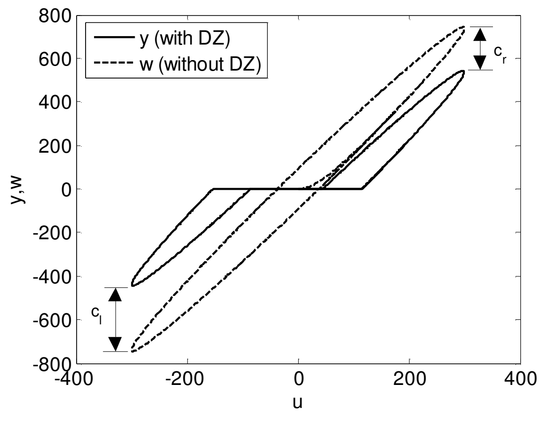

Figure 1.7. $y(k)$ Versus $u(k)$ for the Transfer Function in Equation (2) with and without Deadzone Defined in Equation (3) and Figure 1.5 and $u(k)$ = 700sin(kT).

1.3.3. Deadband. The deadband is the range through which an input signal may be varied, upon reversal of direction, without initiating an observable change in the output signal [1]. The deadband is formulated in accordance with [1] and [12] as

$$y(k)=\begin{cases} w(k)+d & for \quad w(k)<w_l(k) \\ y(k-1) & for \quad w_l(k)\le w(k)\le w_r(k) \\ w(k)-d & for \quad w(k)>w_r(k) \end{cases} \tag{5}$$

where

$$\begin{aligned} w_l(k)&=y(k-1)-d \\ w_r(k)&=y(k-1)+d \end{aligned} \tag{6}$$

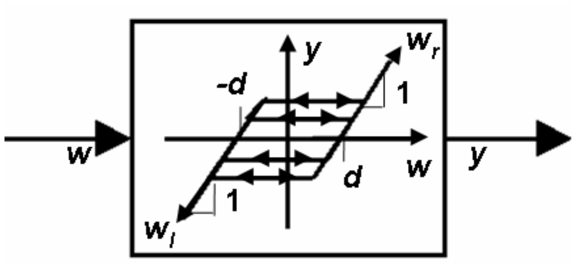

Figure 1.8. Block Diagram of a Deadband Nonlinearity.

Like the deadzone, the deadband can be the result of friction [1]. Another source of deadband is loosely mating mechanical parts often referred to as backlash [12]. A deadband and deadzone can be present simultaneously. Like a deadzone, the deadband can be present at the input or output of an otherwise linear system. The difference between modeling the deadband at the plant input versus modeling it at the output is very subtle. Modeling the plant with the deadband after the transfer function flattens the crests and troughs of an otherwise sinusoidal output. Modeling the plant with the deadband before the transfer function flattens the input to the transfer function to a constant value at the crests and troughs. The plant output will asymptotically approach a steady-state value for those periods of constant input.

1.3.3.1. Deadband Literature Review. Many notable works exist on the detection of deadband. Tao and Kokotovic [12] simulated an adaptive controller to control an unknown linear system with an unknown deadband at the output. Grundelius and Angeli [13] presented an online determination of deadband and linear system parameters using recursive least squares. The model assumed a deadband at the input to a linear system. The estimated parameters were used in deadband inverse in a self tuning regulator in simulation. The method presented in this paper expounds upon Grundelius and Angeli's method to consider systems that also include a deadzone and consider the deadband at the linear system's input as well. Gebler and Holtz [14] simulated the compensation of a deadband determined offline. Woo and Lewis [15] developed a fuzzy controller to compensate for an unknown deadband and demonstrated its utility on a CNC machine tool. Ahmad and Khorrami [16] simulated the use of Lyapunov equations to determine the backlash online and used an adaptive backlash inverse for improved control. Tao [17] developed a fuzzy controller to reduce delay induced by an unknown deadband and verified the controller experimentally with a motor system with output backlash. Zabiri and Samyudia [18] simulated the use of a "Mixed Integer Quadratic Programming" to determine deadband; however, the method had very slow computation time. Lagerberg and Egardt [19] used Kalman filtering to determine the deadband parameters online for the control of an automotive powertrain.

1.3.3.2. Deadband Example. The following is an example of a system with an output deadband as shown in Figure 1.9. The linear transfer function is defined by equation (2). The deadband is defined by *d* = 30. The input is $v(t) = 100\sin(t)$. Equations (5) and (6) become

$$y(k) = \begin{cases} w(k)+30 & for \quad w(k) < y(k-1)-30 \\ y(k-1) & for \quad y(k-1)-30 \le w(k) \le y(k-1)+30 \\ w(k)-30 & for \quad w(k) > y(k-1)+30 \end{cases} \tag{7}$$

The results are shown in Figure 1.10 and Figure 1.11.

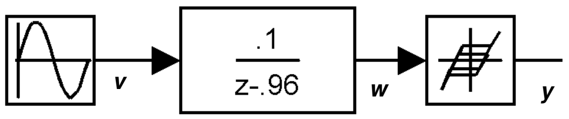

Figure 1.9. Plant with Output Deadband and a Sinusoidal Input.

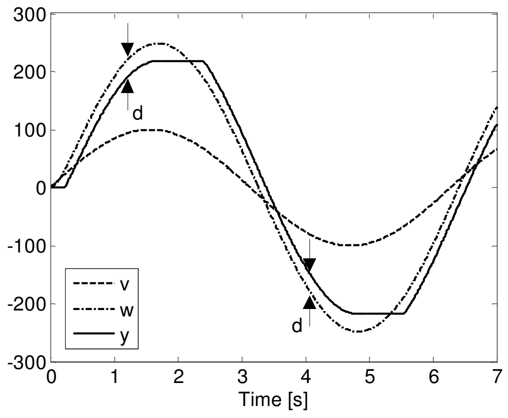

Figure 1.10. Simulated Results from the System Defined by Equations (2) and (7) and by Figure 1.9 with *v*(*k*) = 100sin(*kT*).

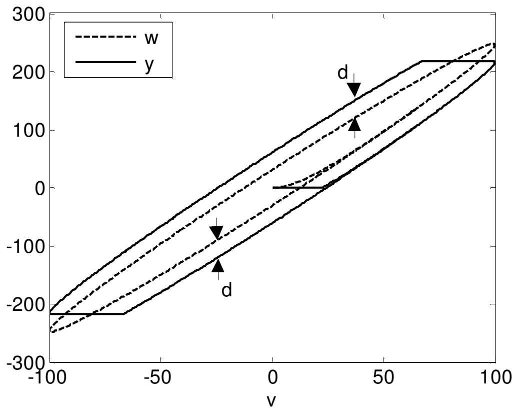

Figure 1.11. ***y*(*k*) and *w*(*k*) versus *v*(*k*) for the system defined by equations (2) and (7) and by Figure 1.9 with *v*(*k*) = 100sin(*kT*).**

Figure 1.10 shows that the deadband "flattens" the peaks and valleys of the output. One way to visualize the deadband is to think about the deadband input being offset on both sides. These offsets can be thought of as one-way tracks. These offsets will be -*d* and *d*. For $\dot{y} > 0$, the offset is *d* and for $\dot{y} < 0$, the offset is -*d*. When $\dot{w}$ changes sign the output remains the same until it intersects the other offset. The output cannot change unless it is on one of the two offsets.

1.3.3.3. Deadzone and Deadband Examples. Consider the following six examples with deadzone and deadband. The linear transfer function is defined by equation (2). The input is $v(k) = 600\sin(kT)$. The deadband parameter is *d* = 100. The deadzone parameters c_r and c_l are 200 and -300, respectively. Plants are defined as shown in Figure 1.12, Figure 1.14, Figure 1.16, Figure 1.18, Figure 1.20, and Figure 1.22. The time history plots as well as input-output plots are shown in Figure 1.13, Figure 1.15, Figure 1.17, Figure 1.19, Figure 1.21, and Figure 1.23, respectively. When the nonlinearity precedes the linear transfer function, the output will asymptotically approach a steady-state value within the deadzone or deadband as the output dynamically responds to the non-linearity. In the case that the linear transfer function precedes the nonlinearity, the output will abruptly or non-asymptotically reach the steady-state value within the deadzone or deadband because the system dynamically responded prior to the introduction of the nonlinearity. Asymptotical and non-asymptotical approaches are denoted "AA" and "NA," respectively, on the input-output plots. Areas in which the output y approaches a non-zero value due to the deadband are denoted as "~0" (where the tilde is used as the logical not).

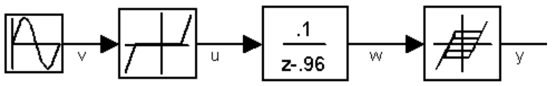

Figure 1.12. Block Diagram of Plant with Input Deadzone Preceded by Sinusoidal Input and Output Deadband.

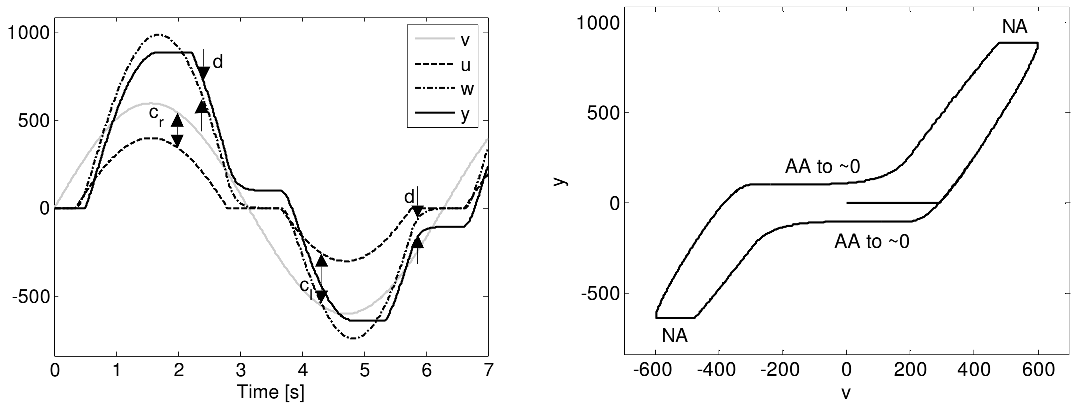

Figure 1.13. Simulated Results for Plant Shown in Figure 1.12 and $v(k) = 600\sin(kT)$.

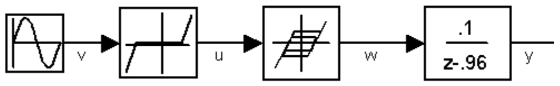

Figure 1.14. Block Diagram of Plant with Input Deadzone Proceeded by Input Deadband and Sinusoidal Input.

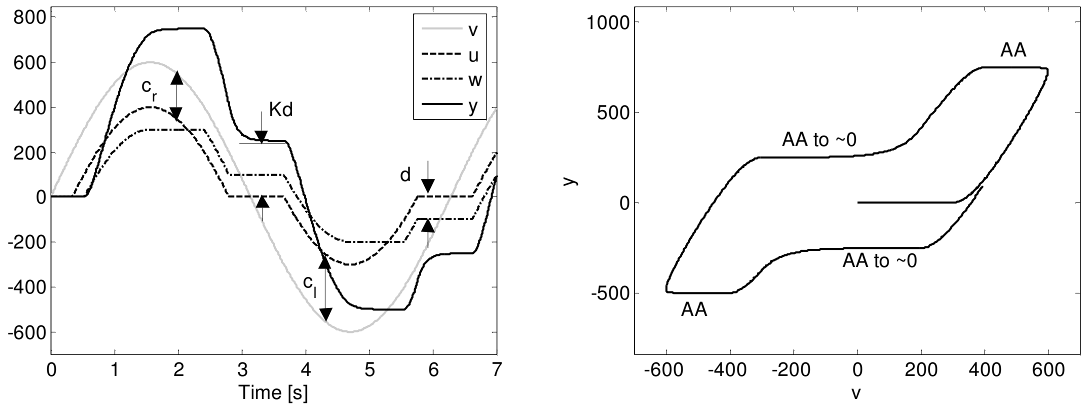

Figure 1.15. Simulated Results for Plant Shown in Figure 1.14 and $v(k) = 600\sin(kT)$.

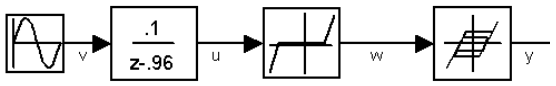

Figure 1.16. Block Diagram of Plant with Output Deadzone Followed By Output Deadband and Preceded by Sinusoidal Input.

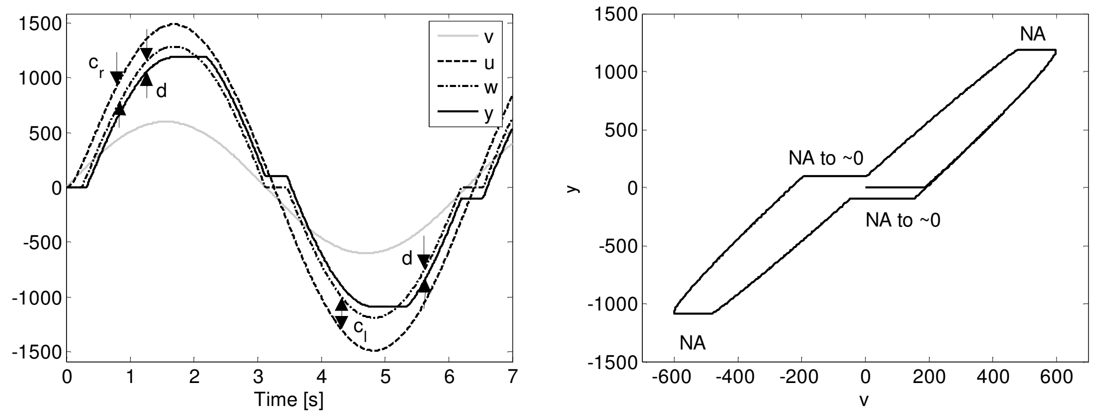

Figure 1.17. Simulated Results for Plant Shown in Figure 1.16 and $v(k) = 600\sin(kT)$.

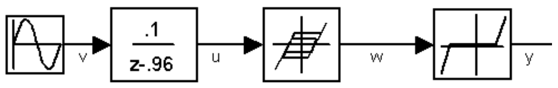

Figure 1.18. Block Diagram of Plant with Output Deadband Followed by Output Deadzone and Preceded by Sinusoidal Input.

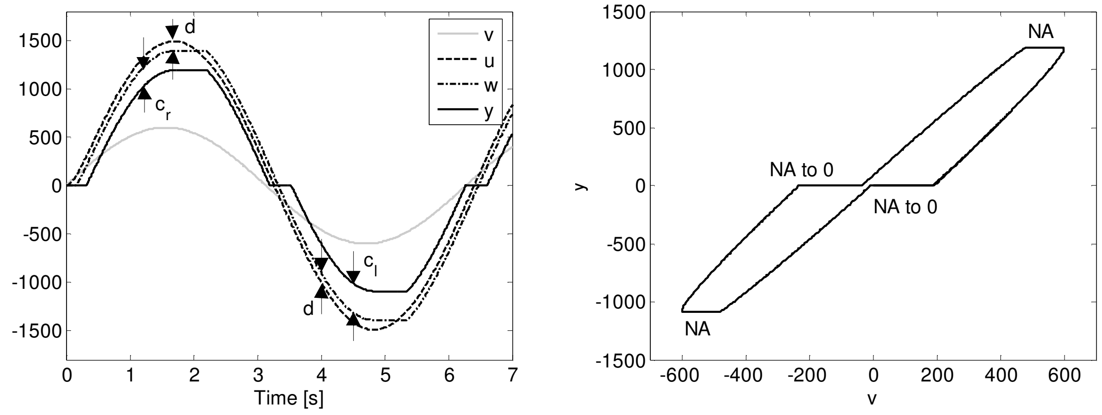

Figure 1.19. Simulated Results for Plant Shown in Figure 1.18 and $v(k) = 600\sin(kT)$.

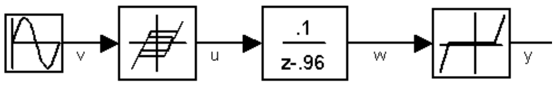

Figure 1.20. Block Diagram of Plant with Input Deadband Preceded by Sinusoidal Input and Output Deadzone.

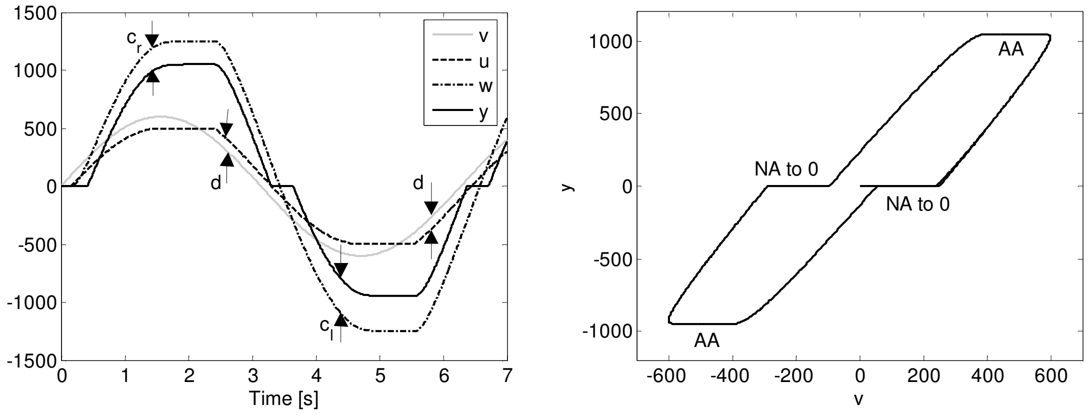

Figure 1.21. Simulated Results for Plant Shown in Figure 1.20 and $v(k)$ = 600sin(kT).

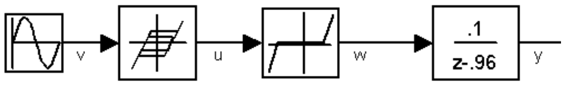

Figure 1.22. Block Diagram of Plant with Input Deadzone Preceded by Input Deadband and Sinusoidal Input.

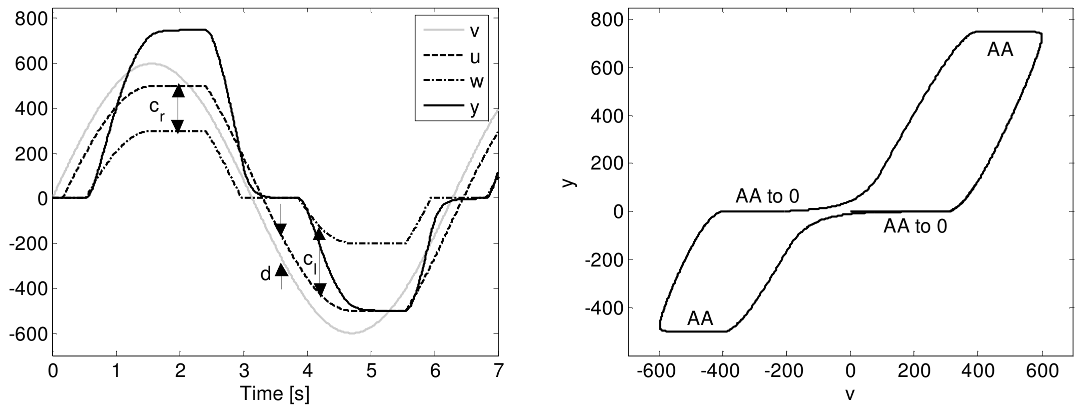

Figure 1.23. Simulated Results for Plant Shown in Figure 1.22 and *v*(*k*) = 600sin(*kT*).

The results from Figure 1.13, Figure 1.15, Figure 1.17, Figure 1.19, Figure 1.21, and Figure 1.23 are very helpful for identifying which nonlinearities are present and where they should be modeled relative to each other and the linear transfer function. When the deadband came before the deadzone (Figure 1.19, Figure 1.21, and Figure 1.23) the plant output went to zero periodically (in this case, every π seconds). However, when the deadband came after the deadzone (Figure 1.13, Figure 1.15, and Figure 1.17) the output went to a non-zero value periodically (in this case, every π seconds). This is also true for a deadband that is much larger than the deadzone. For every case the deadzone was before the transfer function (Figure 1.13, Figure 1.15, and Figure 1.23) the plant output asymptotically approached a constant. However, when the deadzone was after the transfer function (Figure 1.17, Figure 1.19, and Figure 1.21) the plant output went directly to a constant value. When the deadband came before the transfer function, the plant output asymptotically approached a constant value at the crests and troughs. However, when the deadband came after the transfer function, the plant output went directly to a constant value. When the nonlinearity is modeled preceding the transfer function, the output will show a dynamic response to that nonlinearity, hence the asymptotical approach. When the transfer function is modeled preceding the nonlinearity, the output will show an instantaneous non-asymptotical response to the nonlinearity.

1.4. PROBLEM FORMULATION

Now the different types of nonlinearities have been defined in Section 1.1.1 and 1.1.1, the objective is to find the parameters of the linear transfer function, deadzone, and deadband online without the use of the intermediate signals *u*(*k*) and *w*(*k*), which are not measurable. This section begins by applying recursive least squares (*RLS*) to estimate the parameters of a linear plant. Then a single nonlinearity is added and it is shown how to switch between two *RLS* estimation models to estimate the linear and deadzone parameters simultaneously. Without adding more terms to the *RLS* estimation routine, but by switching between four *RLS* estimation models, the linear, deadzone, and deadband parameters can be estimated simultaneously.

1.4.1. Nonlinearities Not Present. A first order linear transfer function with gain *K* and time constant τ has the form

$$\frac{Y(s)}{U(s)} = \frac{K}{\tau s + 1} \tag{8}$$

In the discrete domain this becomes

$$y(k) = -a_1 y(k-1) + b_1 u(k-1) = \boldsymbol{\varphi}(k)^T \, \boldsymbol{\eta} \tag{9}$$

where

$$a_1 = -e^{-\frac{T}{\tau}}, \quad b_1 = K\left(1 - e^{-\frac{T}{\tau}}\right) \tag{10}$$

Using *y*(*k*) as the measurement, the regression vector is

$$\boldsymbol{\varphi}(k) = \left[-y(k-1) \quad u(k-1)\right]^T \tag{11}$$

The system dynamics are described by the unknown parameters a_1 and b_1, which are denoted by the vector $\boldsymbol{\eta}$

$$\boldsymbol{\eta} = \left[a_1 \quad b_1\right]^T \tag{12}$$

Åström *et al.* [20] derive a recursive least squares estimation for the estimated parameters defined in terms of the previous iteration's parameters and a correction factor

$$\hat{\boldsymbol{\eta}}(k) = \hat{\boldsymbol{\eta}}(k-1) + \mathbf{q}(k)\left(y(k) - \hat{y}(k)\right) \tag{13}$$

The estimation $\hat{y}(k)$ is computed from the previous parameter estimates $\hat{\boldsymbol{\eta}}(k-1)$ and the current states $\boldsymbol{\varphi}(k)$

$$\hat{y}(k) = \boldsymbol{\varphi}(k)^T \, \hat{\boldsymbol{\eta}}(k-1) \tag{14}$$

The term **q**(*k*) is the gain of the estimation error which generates a correction factor for the estimate $\hat{\boldsymbol{\eta}}(k-1)$. To find a recursive form of **P**(*k*), Åström *et al.* [20] employ the matrix inversion lemma to define **P**(*k*) solely in terms of **P**(*k*–1) and $\boldsymbol{\varphi}(k)$

$$\mathbf{P}(k) = \frac{1}{\lambda}\left[\mathbf{I} - \mathbf{q}(k)\boldsymbol{\varphi}^T(k)\right]\mathbf{P}(k-1) \tag{15}$$

where

$$\mathbf{q}(k)=\mathbf{P}(k-1)\boldsymbol{\varphi}(k)\left[\lambda\mathbf{I}+\boldsymbol{\varphi}^{T}(k)\mathbf{P}(k-1)\boldsymbol{\varphi}(k)\right]^{-1} \tag{16}$$

An initial value for the matrix **P** is required. Typically it is selected as a large positive diagonal matrix. Åström *et al.* [20] suggest it can be selected arbitrarily. The parameter λ is the exponential forgetting factor. A value of $\lambda = 1$ indicates no forgetting, meaning all of the data is used in the estimation. A lower forgetting factor (typically $0.9 < \lambda < 1$) uses only more recent data to find the parameter estimates. The unknown parameters are updated by equations (13), (14), (15), and (16).

1.4.2. Deadzone Present At Input. Consider the system with a deadzone at the input as shown in Figure 1.2 assuming u is unknown. Using the deadzone definition in equation (1), and replacing u for a function of v in equation (9) yields the following piecewise definition of the output

$$y(k)=\begin{cases}-a_{1,l}y(k-1)+b_{1,l}(v(k)-c_l) & for \quad v(k)<c_l \\ -a_1 y(k-1) & for \quad c_l \le v(k) \le c_r \\ -a_{1,r}y(k-1)+b_{1,r}(v(k)-c_r) & for \quad v(k)>c_r\end{cases} \tag{17}$$

The subscripts r and l indicate right and left-hand planes for the input v. This is helpful in the event plant dynamics are different on either side of the deadzone. It also eliminates the need to include slope terms in the deadzone definition as mentioned in Section 1.1.1. Note neither an r nor an l was used for a_1 within the deadzone. This will be discussed shortly. Since equation (17) is a piecewise definition, it will be necessary to implement two estimation routines: one for $v < c_l$ and one for $v > c_r$. Attempting to estimate b_1, c_r, and c_l for $c_l < v < c_r$ is not possible since the output is not influenced by the input in that region. Because c_l and c_r are unknown, the switching criterion between estimations is unknown. However, for most systems, a general range is known [4,5]. Using this approximation, switching criterion c_{l*} and c_{r*} can be selected such that they are certain (or at least very likely) to meet the following conditions

$$c_{l*} < c_l, c_{r*} > c_r \tag{18}$$

Selecting the parameters too conservatively will only slow the convergence by leaving out usable data. However, not guessing conservatively enough (i.e., one or both conditions in equation (18) are not met) runs the risk that some of the points used in the regression are in the deadzone. Using a constant switching condition c_{l*} and c_{r*} worked well for all of the tests presented in this paper. However, since c_l and c_r are being estimated by the routine, it may be possible to change the switching criterion from the initial approximates c_{l*} and c_{r*} to the estimates of c_l and c_r once they have converged.

The two regression vectors are

$$\boldsymbol{\varphi}_{\mathbf{l}}(k) = \begin{bmatrix} -y(k-1) & v(k-1) & 1 \end{bmatrix}^T \quad for \qquad v(k) < c_{l*} \tag{19}$$

$$\boldsymbol{\varphi}_{\mathbf{r}}(k) = \begin{bmatrix} -y(k-1) & v(k-1) & 1 \end{bmatrix}^T \quad for \qquad v(k) > c_{r*} \tag{20}$$

and the two parameter estimation vectors are

$$\hat{\boldsymbol{\eta}}_{\mathbf{l}}(k) = \begin{bmatrix} a_{1,l} & b_{1,l} & -b_{1,l}c_l \end{bmatrix} \quad for \qquad v(k) < c_{l*} \tag{21}$$

$$\hat{\boldsymbol{\eta}}_{\mathbf{r}}(k) = \begin{bmatrix} a_{1,r} & b_{1,r} & -b_{1,r}c_r \end{bmatrix} \quad for \qquad v(k) > c_{r*} \tag{22}$$

Substituting $\boldsymbol{\varphi}_l$ and $\boldsymbol{\varphi}_r$ for $\boldsymbol{\varphi}$ and $\hat{\boldsymbol{\eta}}_l$ and $\hat{\boldsymbol{\eta}}_r$ for $\boldsymbol{\eta}$ into the equations (11) and (12) respectively and using equations to update the parameters (13), (14), (15), and (16) completes the estimation law. One issue with using separate transfer functions for each side of the deadzone is the question of what to use for the a_1 term in the deadzone as shown in equation (17). The ideal solution would be to estimate a_1 for the decay within the deadzone using *RLS* and $\varphi(k) = y(k\text{-}1)$. However, the unknown deadzone parameters c_r and c_l result in unknown switching conditions. To deal with that, approximates for c_{r*} and c_{l*} must be selected such that $c_{r*} < c_r$, $c_{l*} > c_l$ and $c_{r*} > c_{l*}$. If c_{r*} is not close enough to c_r and the signal decays rapidly, then the *RLS* estimation will "miss" the most significant part of the decay. Then y will appear to be a constant and a_1 will go to -1. It is only possible to use *RLS* to obtain a good estimate of a_1 within the deadzone if the system has a slow decay (i.e., large time constant) and approximations of c_{r*} and c_{l*} close to c_r and c_l are known. If it is not possible to use *RLS* to estimate a_1 within the deadzone, there are a number of different ways to approximate it. One possibility is for all $v(k) \geq 0$ use $a_{1,r}$ and $v(k) < 0$ use $a_{1,l}$. Another possibility is to use the previous a_1. For example if $v(k) > c_r$ then v drops to where $c_r > v(k) > c_l$, continue to use $a_{1,r}$ until $v(k) < c_l$. This latter method will be used throughout the rest of the paper and no estimation of any a_1 terms is performed within the deadzone.

1.4.3. Deadzone Present At Input And Deadband Present At Output.

Now consider the plant shown in Figure 1.12 with a deadzone at the input and a deadband at the output. In equation (17), replacing $y(k)$ with $w(k)$ yields

$$w(k)=\begin{cases}-a_{1,l}w(k-1)+b_{1,l}\left(v(k)-c_l\right) & for \quad v(k)<c_l\\ -a_1 w(k-1) & for \quad c_l\le v(k)\le c_r\\ -a_{1,r}w(k-1)+b_{1,r}\left(v(k)-c_r\right) & for \quad v(k)>c_r\end{cases}$$

(23)

From the deadband equations (5) and (6)

$$w(k)=\begin{cases}y(k)-d & for \quad w(k)\le w_l(k)\Rightarrow y(k)\le y(k-1)\\ y(k)+d & for \quad w(k)\ge w_r(k)\Rightarrow y(k)\ge y(k-1)\end{cases}$$

(24)

Substituting *k*-1 for *k* in equation (24) yields

$$w(k-1)=\begin{cases}y(k-1)-d & for \quad w(k-1)\le w_l(k-1)\Rightarrow y(k-1)\le y(k-2)\\ y(k-1)+d & for \quad w(k-1)\ge w_r(k-1)\Rightarrow y(k-1)\ge y(k-2)\end{cases}$$

(25)

The equations are simplified by defining the derivative as the backwards difference

$$\dot{y}(k)=\frac{y(k)-y(k-1)}{T}$$

(26)

The conditions $y(k) < y(k - 1)$, $y(k) > y(k - 1)$, $y(k - 1) < y(k - 2)$, and $y(k - 1) > y(k - 2)$, become $\dot{y}(k) < 0$, $\dot{y}(k) > 0$, $\dot{y}(k-1) < 0$, and $\dot{y}(k-1) > 0$ respectively. Combining equation (23) and (24) yields the equation for a system defined by a first order transfer function with input deadzone and output deadband

$$y(k)=\begin{cases}-a_{1,l+}y(k-1)+b_{1,l+}v(k)-d_l\left(1+a_{1,l+}\right)-b_{1,l+}c_l & for \quad \dot{y}(k),\dot{y}(k-1)>0, v(k)<c_l\\ -a_{1,l-}y(k-1)+b_{1,l-}v(k)+d_l\left(1+a_{1,l-}\right)-b_{1,l-}c_l & for \quad \dot{y}(k),\dot{y}(k-1)<0, v(k)<c_l\\ -a_{1,r+}y(k-1)+b_{1,r+}v(k)-d_r\left(1+a_{1,r+}\right)-b_{1,r+}c_r & for \quad \dot{y}(k),\dot{y}(k-1)>0, v(k)>c_r\\ -a_{1,r-}y(k-1)+b_{1,r-}v(k)+d_r\left(1+a_{1,r-}\right)-b_{1,r-}c_r & for \quad \dot{y}(k),\dot{y}(k-1)<0, v(k)>c_r\end{cases}$$

(27)

The four regression vectors become

$$\begin{aligned}
\boldsymbol{\varphi}_{\mathbf{l+}}(k) &= \begin{bmatrix}-y(k-1) & v(k-1) & 1\end{bmatrix}^T \quad for \quad v(k)<c_{l*}, y(k)>y(k-1)>y(k-2) \\
\boldsymbol{\varphi}_{\mathbf{l-}}(k) &= \begin{bmatrix}-y(k-1) & v(k-1) & 1\end{bmatrix}^T \quad for \quad v(k)<c_{l*}, y(k)<y(k-1)<y(k-2) \\
\boldsymbol{\varphi}_{\mathbf{r+}}(k) &= \begin{bmatrix}-y(k-1) & v(k-1) & 1\end{bmatrix}^T \quad for \quad v(k)>c_{r*}, y(k)>y(k-1)>y(k-2) \\
\boldsymbol{\varphi}_{\mathbf{r-}}(k) &= \begin{bmatrix}-y(k-1) & v(k-1) & 1\end{bmatrix}^T \quad for \quad v(k)>c_{r*}, y(k)<y(k-1)<y(k-2)
\end{aligned} \tag{28}$$

The four parameter estimation vectors become

$$\begin{aligned}
\hat{\boldsymbol{\eta}}_{\mathbf{l+}}(k) &= \begin{bmatrix}-a_{1,l+} & b_{1,l+} & e_{l+}\end{bmatrix} \quad for \quad v(k)<c_{l*}, y(k)>y(k-1)>y(k-2) \\
\hat{\boldsymbol{\eta}}_{\mathbf{l-}}(k) &= \begin{bmatrix}-a_{1,l-} & b_{1,l-} & e_{l-}\end{bmatrix} \quad for \quad v(k)<c_{l*}, y(k)<y(k-1)<y(k-2) \\
\hat{\boldsymbol{\eta}}_{\mathbf{r+}}(k) &= \begin{bmatrix}-a_{1,r+} & b_{1,r+} & e_{r+}\end{bmatrix} \quad for \quad v(k)>c_{r*}, y(k)>y(k-1)>y(k-2) \\
\hat{\boldsymbol{\eta}}_{\mathbf{r-}}(k) &= \begin{bmatrix}-a_{1,r-} & b_{1,r-} & e_{r-}\end{bmatrix} \quad for \quad v(k)>c_{r*}, y(k)<y(k-1)<y(k-2)
\end{aligned} \tag{29}$$

$$\begin{aligned}
e_{l+} &= -d_l\left(1+a_{1,l+}\right)-b_{1,l+}c_l \quad for \quad v(k)<c_{l*}, y(k)>y(k-1)>y(k-2) \\
e_{l-} &= \ \ d_l\left(1+a_{1,l-}\right)-b_{1,l-}c_l \quad for \quad v(k)<c_{l*}, y(k)<y(k-1)<y(k-2) \\
e_{r+} &= -d_r\left(1+a_{1,r+}\right)-b_{1,r+}c_r \quad for \quad v(k)>c_{r*}, y(k)>y(k-1)>y(k-2) \\
e_{r-} &= \ \ d_r\left(1+a_{1,r-}\right)-b_{1,r-}c_r \quad for \quad v(k)>c_{r*}, y(k)<y(k-1)<y(k-2)
\end{aligned} \tag{30}$$

Note again, the coefficients for each estimation are kept separate. Positive and negative values of *v* are denoted by *r* and *l* subscripts, respectively. The positive and negative conditions of $\dot{y}$ are denoted by + and – subscripts, respectively. Substituting $\boldsymbol{\varphi}_{l+}$, $\boldsymbol{\varphi}_{l-}$, $\boldsymbol{\varphi}_{r+}$, and $\boldsymbol{\varphi}_{r-}$ for $\boldsymbol{\varphi}$ and $\hat{\boldsymbol{\eta}}_{l+}$, $\hat{\boldsymbol{\eta}}_{l-}$, $\hat{\boldsymbol{\eta}}_{r+}$ and $\hat{\boldsymbol{\eta}}_{r-}$ for $\hat{\boldsymbol{\eta}}$ into the equations from Section 1.1.1 completes the estimation law.

The deadzone and deadband parameters are determined from the following equations

$$\begin{bmatrix} d_l \\ c_l \end{bmatrix} = \begin{bmatrix} -\left(1 + a_{1,l+}\right) & -b_{1,l+} \\ \left(1 + a_{1,l-}\right) & -b_{1,l-} \end{bmatrix}^{-1} \begin{bmatrix} e_{1,l+} \\ e_{1,l-} \end{bmatrix} \tag{31}$$

$$\begin{bmatrix} d_r \\ c_r \end{bmatrix} = \begin{bmatrix} -\left(1 + a_{1,r+}\right) & -b_{1,r+} \\ \left(1 + a_{1,r-}\right) & -b_{1,r-} \end{bmatrix}^{-1} \begin{bmatrix} e_{1,r+} \\ e_{1,r-} \end{bmatrix} \tag{32}$$

1.4.4. Other Cases. There are many other possibilities of the order in which the nonlinearities present themselves as shown in Section 1.3.3.3. However the procedure for setting up the estimation routine is the same:

1. Write out the generic linear transfer function in discrete form
2. Perform substitutions from deadzone and deadband equations in (1) and (4), respectively, creating an equation for every possible condition
3. Create an *RLS* estimation law for each condition
4. Solve for the nonlinear terms

1.4.5. Filtering. One of the challenges of implementing the routine presented in Sections 1.1.1 is that knowledge of the sign of $\dot{y}$ is required. The output y is typically a signal coming from an electronic sensor. Voltage signals are inherently noisy. The signal $\dot{y}$ computed by a first order finite backwards difference using equation (26) becomes noisier as the sample time becomes smaller.

Because of the noise, the statement $\dot{y} > 0$ can switch back and forth from true to false very rapidly. For this reason, it is helpful to define limits on either side of zero. For all points between these two limits, $\dot{y} \approx 0$. All points above and below the limits will be considered as $\dot{y} > 0$ and $\dot{y} < 0$, respectively.

If the noise has a Gaussian distribution, it is known that 99.7 % of the points will lie within three standard deviations (+/-3σ) of the mean. Therefore +/- 3σ would be a good place to start for limits. However it can be a problem if the +/- 3σ range spans nearly the entire range of $\dot{y}$. It was necessary to filter $\dot{y}$ and y in the example of the hydraulic test bench presented below. Since filters add delay to the measurement signal, a pure delay was added to the input v to maintain a meaningful relationship between the input and output.

An M-point non-weighted moving average filter that uses the current term delays the signal by ½ (M-1) samples or ½ (M-1)T seconds. The variance (σ^2 denotes population variance, S_N^2 denotes sample variance) is reduced by a factor of 1/M (see the Appendix for derivation). Note that after filtering the signal can no longer be considered as a random number because there is positive covariance between consecutive data points. The variance of $\dot{y}$ when calculated from equation (26) is given by

$$\operatorname{var}(\dot{y}) = 2\operatorname{var}(y)T^{-2} \tag{33}$$

Since the sample period T is usually a very small number, the variance of $\dot{y}$ becomes very large and $\dot{y}$ will need to be filtered. See the Appendix for derivation.

1.5. SIMULATED RESULTS

1.5.1. Simulation Example - Without Noise. Consider the plant piecewise-defined by two discrete transfer functions with a deadzone input nonlinearity and a deadband output nonlinearity. The transfer functions are

$$\frac{Y(z)}{U(z)} = \begin{cases} \dfrac{0.1}{z-0.96} & for \quad v \geq 0 \\ \dfrac{0.15}{z-0.94} & for \quad v < 0 \end{cases}$$

(34)

and have a sample period T = 0.005. Using the regression and parameter estimation vectors given in equations (28) through (30) with recursive least squares, the plant parameters will be estimated. The deadzone and deadband parameters are c_r = 200, c_l = -300, d_r = 30, and d_l = 50. The positive and negative deadzone offset guesses were c_{r*} = 350 and c_{l*} = -350, respectively. The initial covariance matrix was chosen to be diag[1000 1000 1000]. The initial states for $\boldsymbol{\varphi}$ were [0 0 0]. The initial estimates for each set of parameters were chosen to be [1 1 1]. The forgetting factor λ was chosen to be 0.99.

The estimates converged to their proper values. As seen in Figure 1.24, the estimates did not converge smoothly, but rather "stair-stepped" towards the actual value. This is because each estimation routine is not continuously running, but rather triggered when certain conditions are met. The spikes in the estimated output are present each time one of the *RLS* estimation algorithms is used for the first time. The estimates for c_r, c_l, d_r, and d_l converge slower than the a_1's and b_1's. This is because they depend on the convergence of the a_1, b_1 and e_1 term for both positive and negative $\dot{y}$ before they will compute properly. The 1 + a_1 term is likely to be the cause of slow convergence. For example

$$d_r = \frac{e_{r-} + b_{1,r-}c_r}{\left(1 + a_{1,r-}\right)}$$

(35)

If every parameter needed to solve for d_r has converged with the exception of $a_{1,r-}$ which is off by one percent, the error is estimating d_r is 20 percent. This is due to the fact that $a_{1,r-}$ is a number close to -1 making the denominator very small.

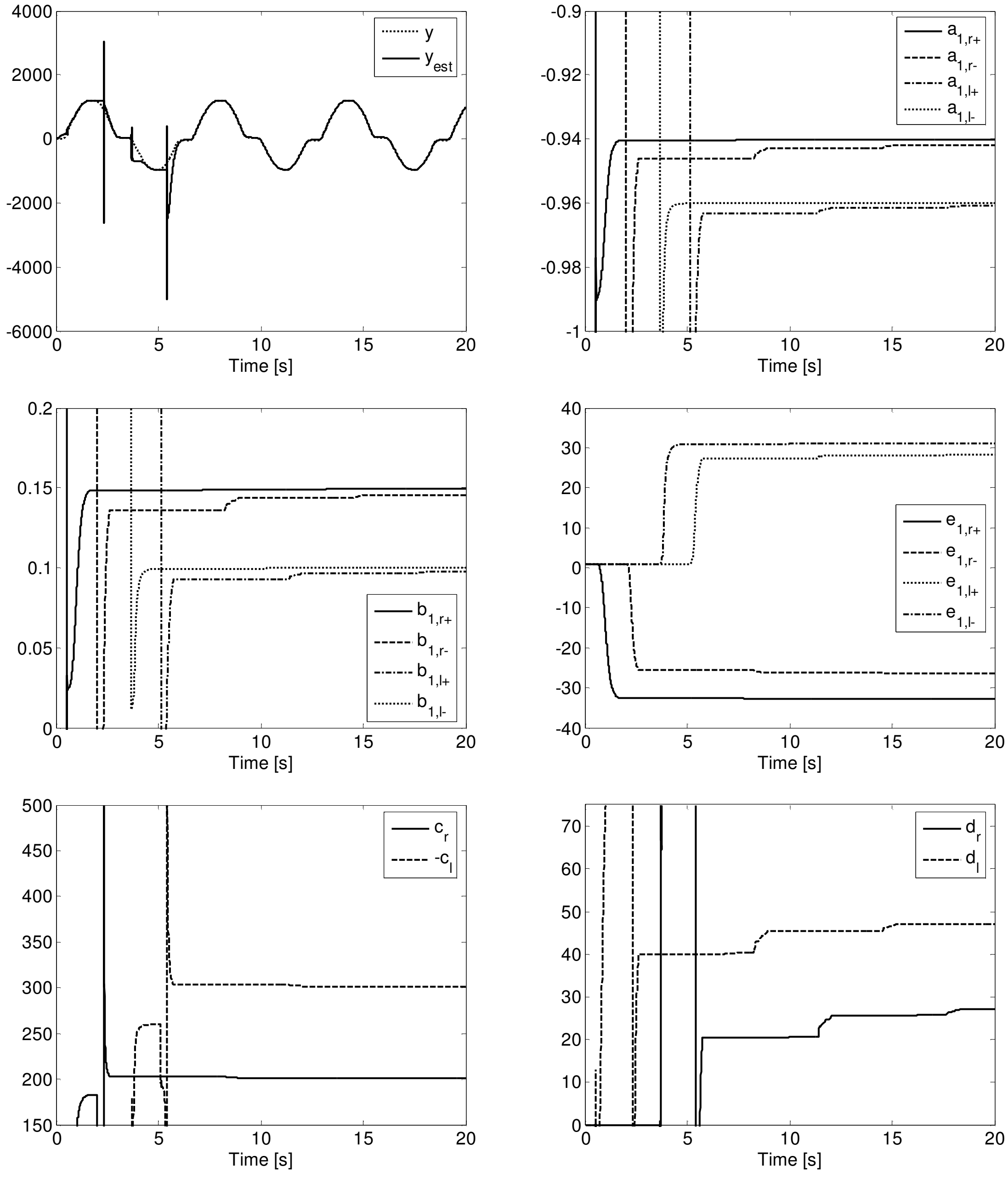

Figure 1.24. Simulated Estimation Results of Equation (34), c_r = 200, c_l = -300, d_r = 30, and d_l = 50.

1.5.2. Simulation Example – With Gaussian Noise. Consider the previous example from Section 1.1.1 with an added zero-mean Gaussian noise to the simulated signal with variance 200. The raw output signal is filtered with a 20-point moving average filter, differentiated, and filtered again with a 20-point moving average filter. The filters have delayed the signal 19 steps (9.5 steps for each of the filters). Every estimated $\dot{y}$ greater than 600 is considered to be positive. Every estimated $\dot{y}$ less than -600 is considered to be negative. Figure 1.25 shows the actual and estimated outputs as well as the parameter convergence.

The parameters converge slower due to the noise as shown in Figure 1.25. After 200 seconds at $T = 0.005$ sec, the estimated parameters b_1 terms still had errors of up to 15%, the a_1 terms had errors up to 1%, c_r and c_l had errors of approximately 1%, and d_r and d_l had errors of up to 50%. As was shown in the previously in 1.1.1, the errors in the a_1, b_1, and e_1 can be magnified in the calculated estimations c_r, c_l, d_r, and d_l. Figure 1.26 showns the estimation error plotted with +/-3σ of the superimposed Gaussian noise ($\pm 3\sqrt{200}$). If the parameters were estimated perfectly, there would theoretically be three data points per every 1000 in which the error is outside of the +/-3σ band. The number of data points observed in which the error was outside the +/-3σ band was 12 per 1000 for 50-200 seconds (16 per 1000 for 50-100 sec, 11 per 1000 for 100 to 150 sec, and 10 per 1000 for 150 to 200 sec.) This indicates that even after 200 seconds (40,000 data points) there is a very small but observable error in the estimation. It is difficult to determine whether or not the parameters will eventually converge to their proper values, but these results indicate that the output can be estimated very closely even in the presence of noise.

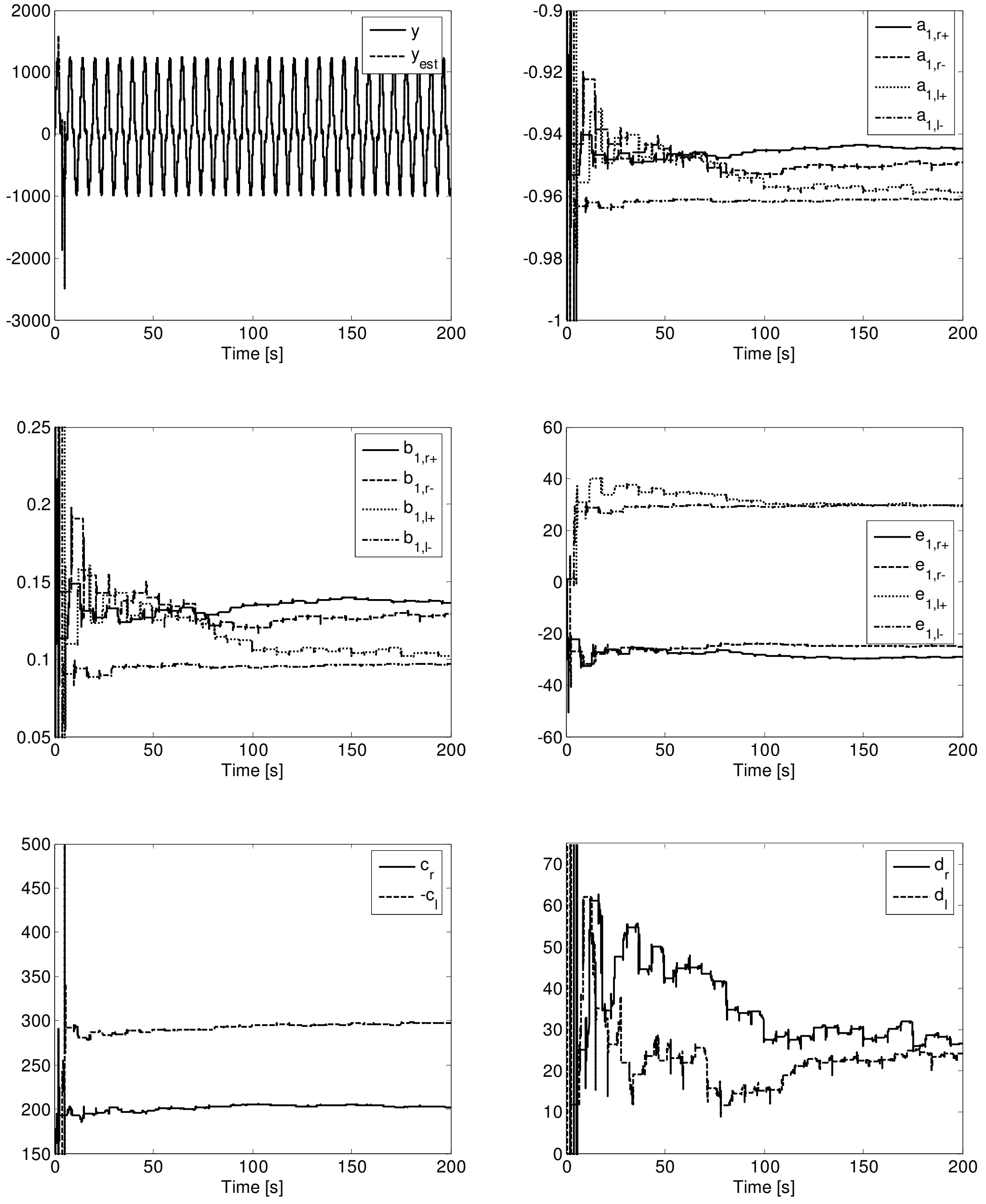

Figure 1.25. Simulated Estimation Results of Equation (34), c_r = 200, c_l = -300, d_r = 30, and d_l = 50 with σ^2 = 200.

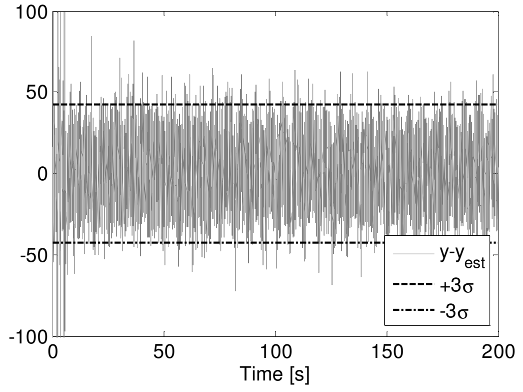

Figure 1.26. Estimation Error

1.6. EXPERIMENTAL RESULTS

The online deadzone and deadband estimation scheme established in Section 1.4 and simulated in Section 1.5 was tested to demonstrate its applicability. Three case studies are performed: (1) modeling the dynamics and the deadzone in a motor-driven cart with voltage input and velocity output, (2) modeling the dynamics and deadzone of an electro-hydraulic pilot valve with current input and pilot pressure output, and (3) modeling the dynamics, the deadzone, and deadband in a cart that is loosely coupled to a motor-driven cart.

1.6.1. Modeling Motor-Driven Cart Velocity. One example of a first order system with a deadzone is a DC-motor driven cart with voltage input and velocity output, as shown in Figure 1.27. There is a range of voltages around 0 *V* in which the cart does not move due to Coulomb friction. This results in an input deadzone. Figure 1.28 shows the input-output curve for 30 seconds (approximately 19 cycles) of the cart for a sinusoidal input voltage of $v(t) = 6\sin(4t)$. The output goes directly to zero in the middle of the plot. This indicates the plant dynamics should be modeled before the deadzone due to Coulomb friction acting at the motor's output. The curve has nearly the same shape as the curve in Figure 1.7 indicating it should be possible to fit the cart's dynamics to a first order system with an output deadzone.

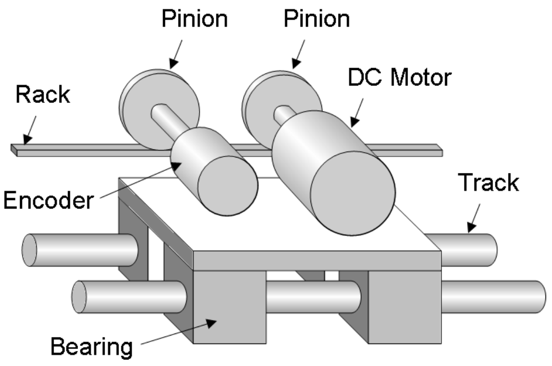

Figure 1.27. DC-Motor Driven Cart.

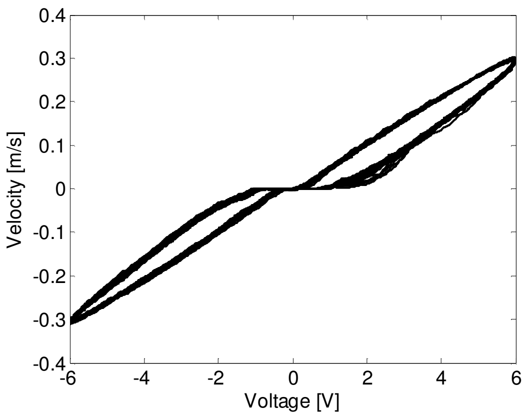

Figure 1.28. Velocity Versus Voltage for DC-Motor Driven Cart with $v(t) = 6\sin(4t)$.

Since there does not appear to be any deadband in the velocity, the online parameter estimation scheme shown in Section 1.1.1 can be used to derive equations for *e_l* and *e_r*

$$\begin{aligned} e_l &= -c_l\left(1+a_{1,l}\right) \quad for \quad y(k)<0 \\ e_r &= -c_r\left(1+a_{1,r}\right) \quad for \quad y(k)>0 \end{aligned} \tag{36}$$

Not only is modeling the deadzone after the linear transfer function more accurate for this system, it has the added benefit of having known switching conditions. If the deadzone is modeled before the transfer function, values larger than c_r and $-c_l$ have to be assumed as switching conditions. In this arrangement, the switching condition is based solely on $y(k)$, which is known and needs little filtering. The initial covariance matrix P was chosen as diag[1000 1000 1000]. The forgetting factor λ was chosen as 0.99. The initial estimation vector η was chosen as [1 1 1]. Since an incremental encoder is use, the initial output is 0. The sample time was 0.001 seconds. Figure 1.29 shows the estimated output is very close to the actual output. The estimated parameters converged to within a few percent of their final values within approximately two seconds.

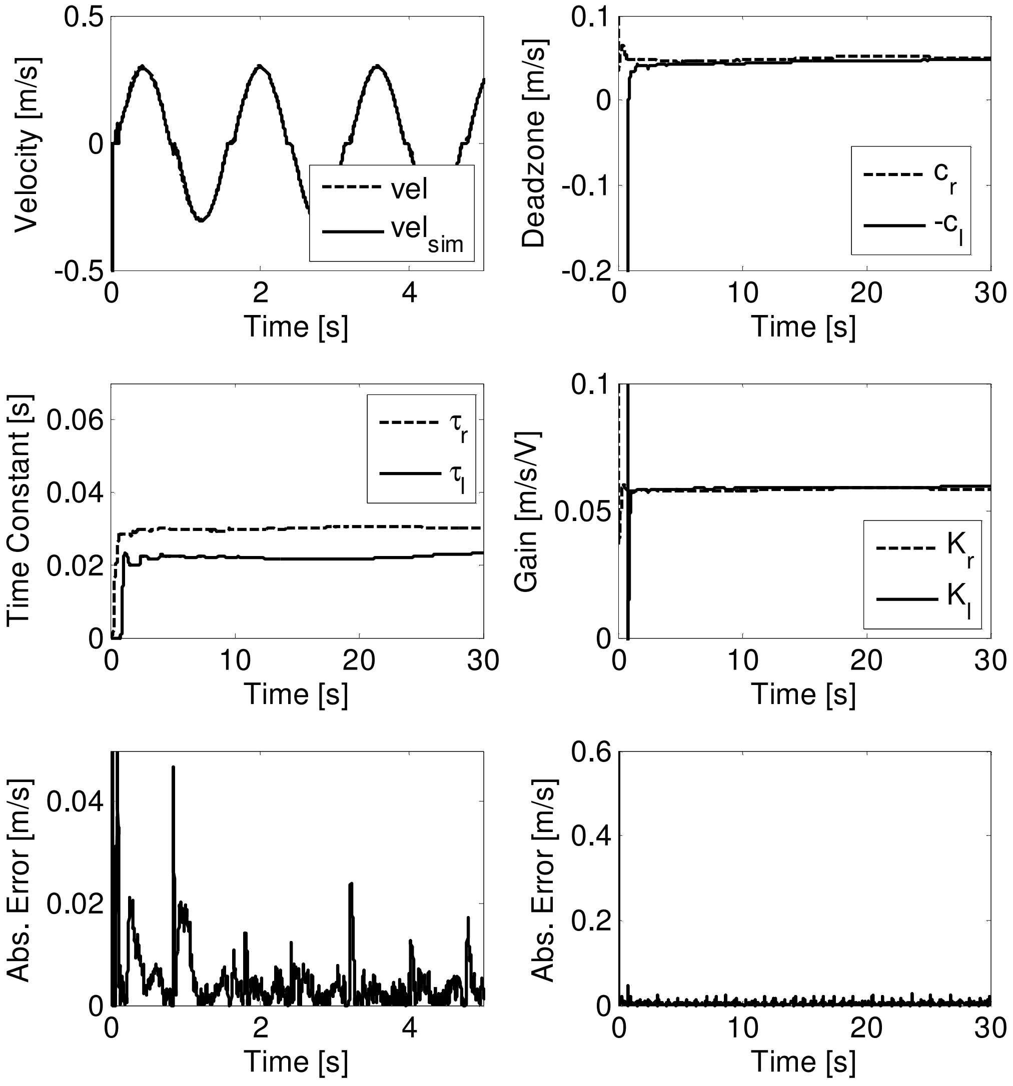

Figure 1.29. Results of Online Estimation of DC-Motor Driven Cart Dynamics and Deadzone for a Sinusoidal Input.

Several more tests were conducted to examine the algorithm at other frequencies, amplitudes, and waveforms. Three tests were conducted for each permutation of 2, 4, 6, and 8 rad/s frequencies and 2, 4, 6, and 8 *V* amplitude for sine, square, and triangle waves, totaling 144 tests. The functions for the square wave and triangle wave were defined as

$$\begin{aligned}\text{Square}(t) &= \mathit{Amplitude}\cdot\text{sign}(\sin(t))\\ \text{Triangle}(t) &= \mathit{Amplitude}\cdot\frac{2}{\pi}\sin^{-1}(\sin(t))\end{aligned} \tag{37}$$

Examples of those results are shown in Figure 1.30 through Figure 1.32. A randomized input signal was created by generating a zero-mean Gaussian distributed random number with variance 10000 every time step and filtering with a first-order low pass filter with a time constant of 0.5 seconds. An example of input signals is shown in Figure 1.33 for sine, square, and triangle waves with amplitude 8 and frequency 1 rad/s and the randomized signal. Figure 1.34 shows the root mean square error of the simulated and actual output calculated from time 7.5 to 30 seconds and normalized about the maximum absolute value of the output.

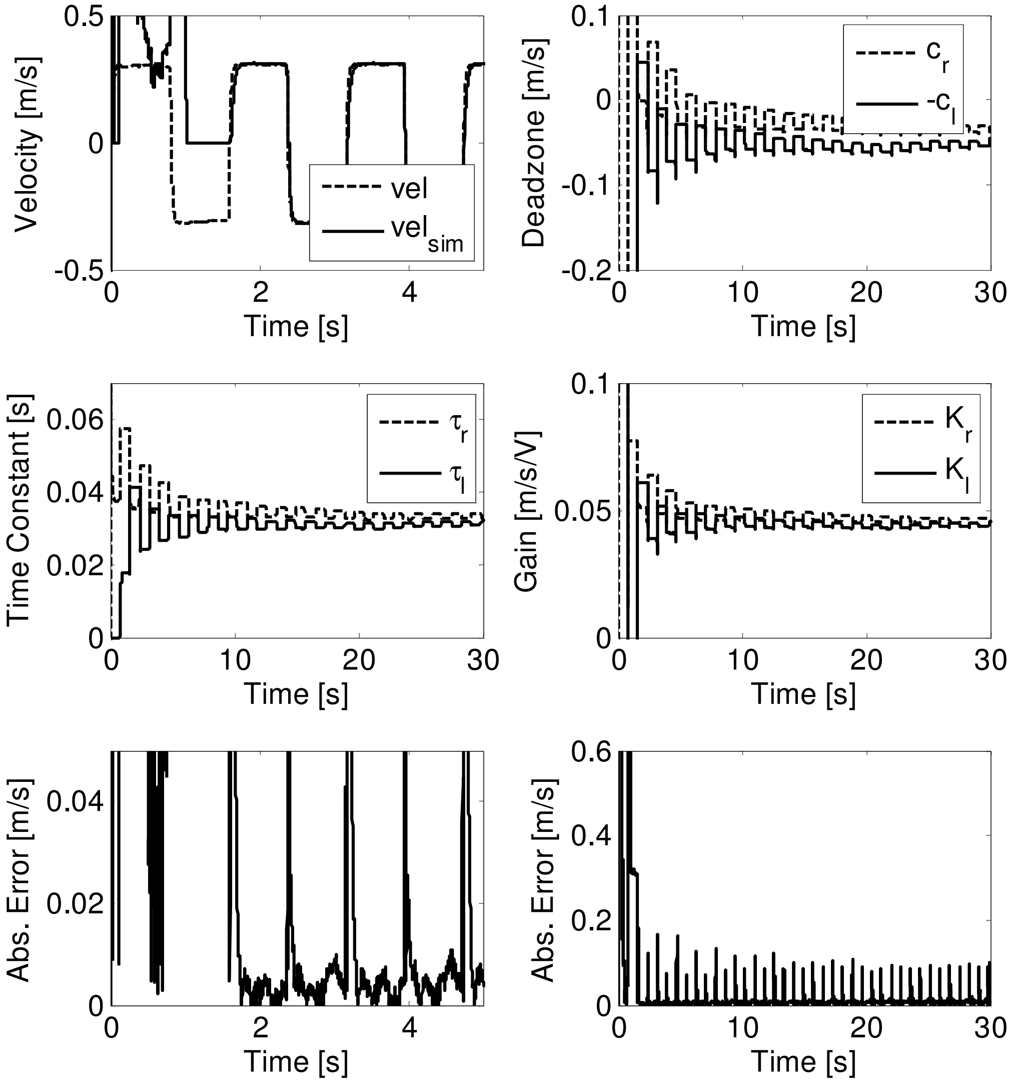

Figure 1.30. Results of Online Estimation of DC-Motor Driven Cart Dynamics and Deadzone for a Square-Wave Input.

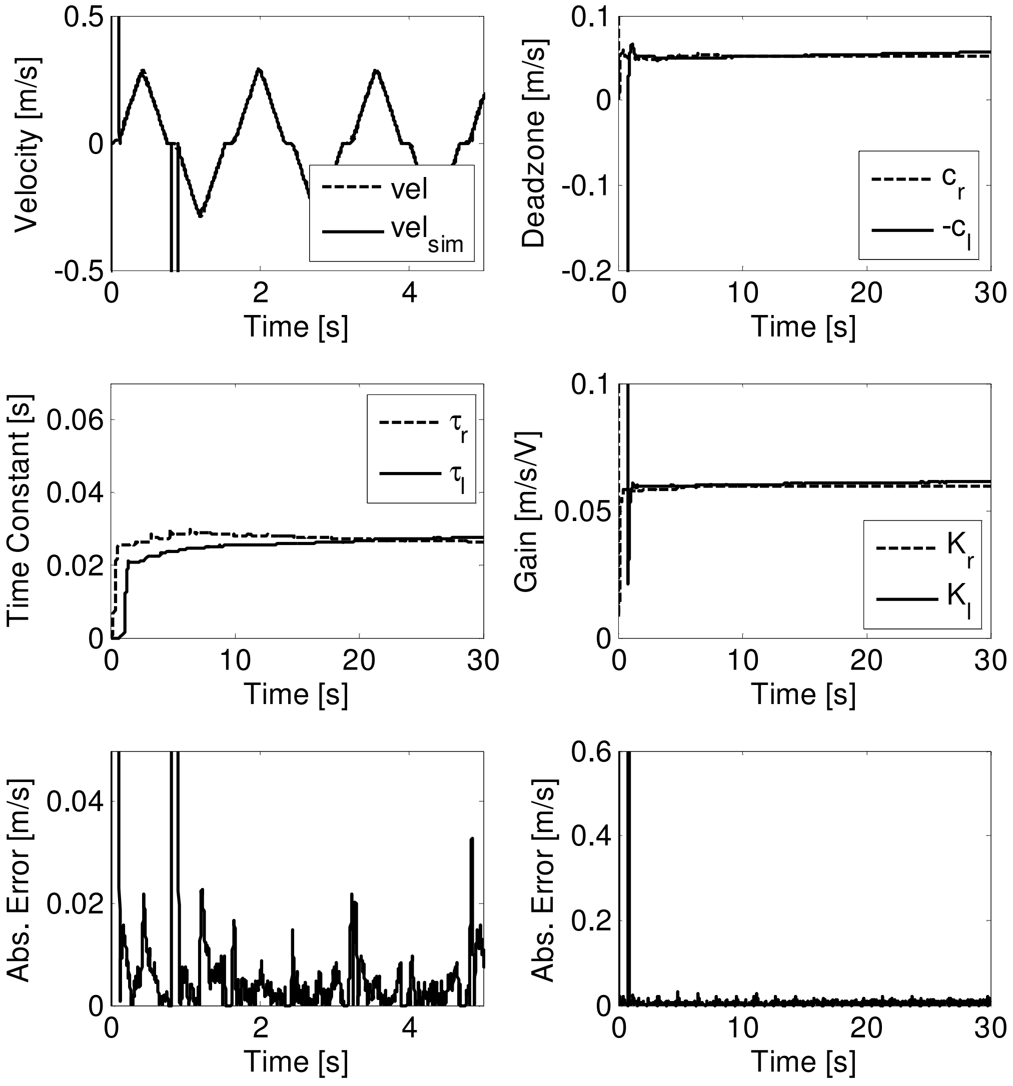

Figure 1.31. Results of Online Estimation of DC-Motor Driven Cart Dynamics and Deadzone for a Triangle-Wave Input.

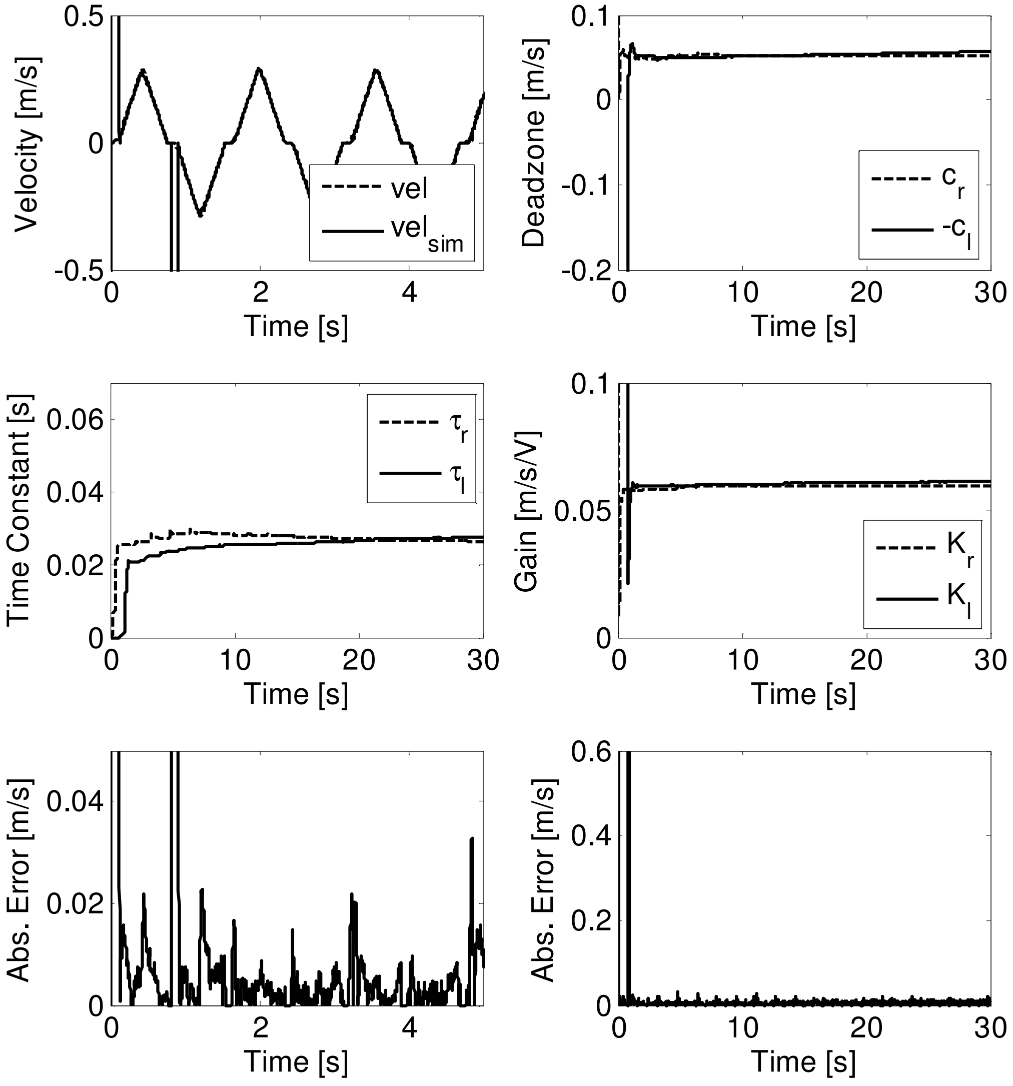

Figure 1.32. Results of Online Estimation of DC-Motor Driven Cart Dynamics and Deadzone for a Randomized Input.

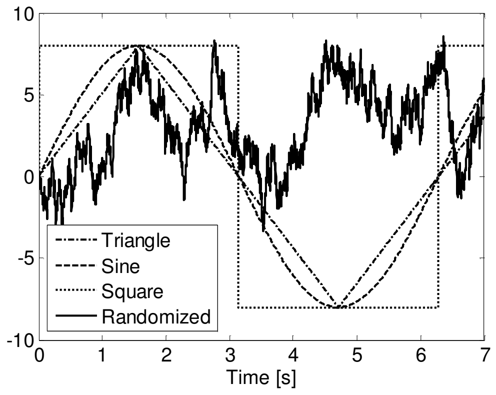

Figure 1.33. Input Signal Waveforms to Motor Driven Cart.

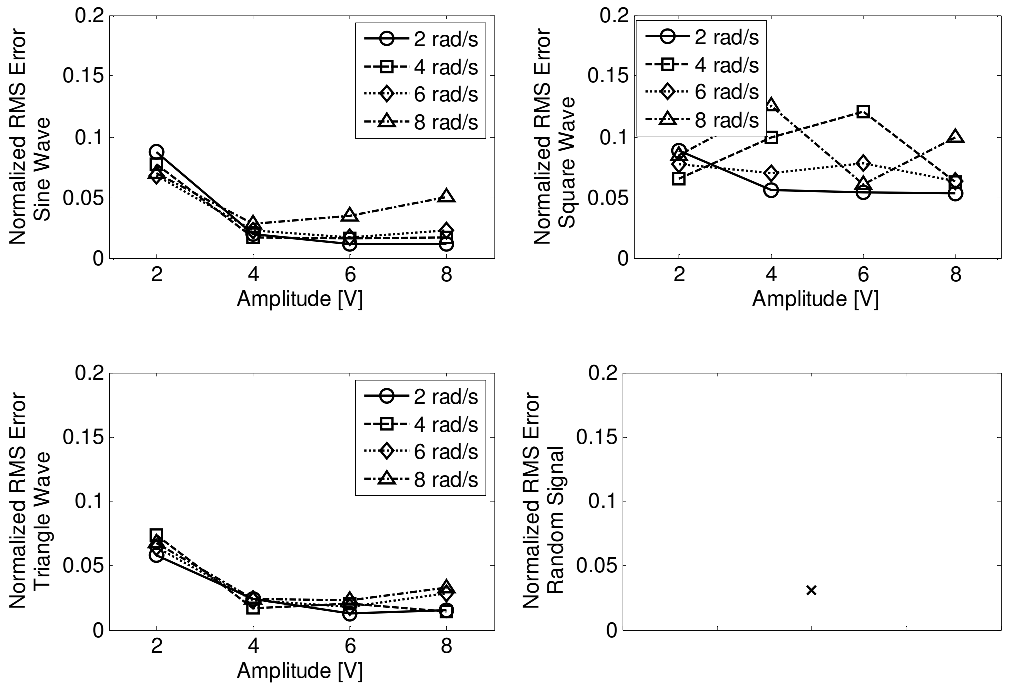

Figure 1.34. Normalized Root Mean Square Error between Simulated and Experimental DC-Motor Drive Cart Velocities.

It can be seen from Figure 1.34 that the error was less than 5% most of the time for the sine and triangle waves and that errors were larger at smaller amplitudes. This is largely due to the fact that for low amplitudes the signal to noise ratio is higher. As was shown in simulation, a spike in the estimated output occurs the first time an estimator was used. Once the estimator has received about 100 milliseconds of data, its values have converged close enough that it will not generate any more spikes while it is used. The randomized signal's error was zero in a few locations as a result of the output remaining in the deadzone for several seconds.

The convergence was the worst using a square wave voltage signal. This is due to the input signal not being sufficiently exciting. For a square wave input, the estimation algorithm only receives two values for the input signal: the voltages equal to the wave's amplitude and the negative of the amplitude. When the output stabilizes such that $y(k) \approx y(k-1)$ and since $u(k) = u(k\text{-}1)$ more than 99% of the time, there becomes an infinite number of parameter estimations that will minimize the estimation error. For instance, under stable conditions when $y(k) = y(k-1)$ and $u(k) = u(k-1)$, the estimates $a_1 = \text{-}1$, $b_1 = 0$, $e_1 = 0$ create an estimation with zero error. However so will $a_1 = 0$, $b_1 = K$, $e_1 = 0$ and $a_1 = 0$, $b_1 = 0$, $e_1 = y(k)$. In the plots of the parameter convergence of the square wave, it is apparent that the estimations spiked every time the input changed. After the input has changed, the output y reaches a steady state value momentarily as well as the estimates until the next input change occurs. This lack of excitation caused the deadzone parameter estimates c_r and $-c_l$ to become negative in some cases, which was compensated for by underestimating the gains. However, having a $c_l > c_r$ does not make physical sense when the deadzone is modeling Coulomb friction.

1.6.2. Modeling Pilot Valve Pressure. The control of hydraulic systems typically requires manual or solenoid-type control valves. If the pressure is low enough, the operator's movement of a joystick or the actuation of a solenoid moves a spool to directly regulate the working pressure. The working pressure acts on an implement such as a cylinder or motor. However, for a high-pressure system, the forces required to move the spool to regulate the working pressure are too high for an operator or solenoid to exert (a solenoid can only exert about 50 to 100 N [21]). In this case the operator merely controls an intermediate pressure called the pilot pressure and the forces generated move the main spool to control working port pressure. It is desirable to have some deadzone in the system to mitigate unintended motion should the joystick be accidentally bumped. However, deadzone can also be detrimental to a controller's performance if not accounted for.

The pilot valve's deadzone results from two main factors: spring pretension and spool land overlap. When the valve is closed the fluid flows from the supply port through the spool (this is shown to flow external to the spool in Figure 1.35 and Figure 1.36 for clarity). It passes through a small port, and flows around a ball, and out the drain port. When current flows through the solenoid coil, a pin extending from the armature presses on the ball restricting flow to the drain. The pressure builds up on the end of the spool opposite the spring. The pressure has to build high enough to overcome the pretension in the spring before the spool can move. Once the spool begins to move, it still needs to move a certain distance for fluid to flow from the supply to the control port. This is because the land on the spool overlaps the supply port by some distance – on the order of

2-3 mm. For reference the full range of motion of the spool is around 7 mm. This spool land overlap and spring pretension result in a significant deadzone.

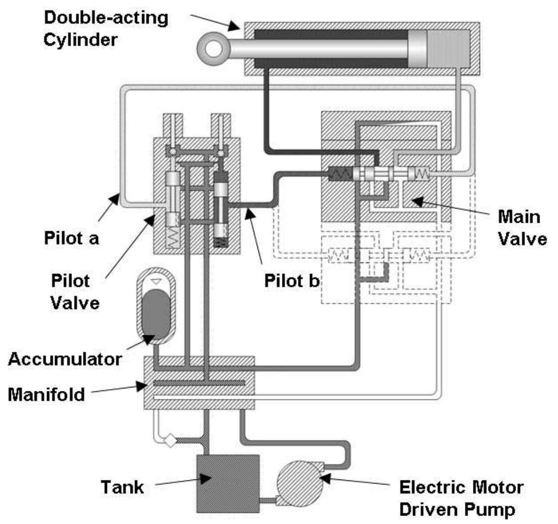

Figure 1.35. Pilot-Operated Hydraulic System.

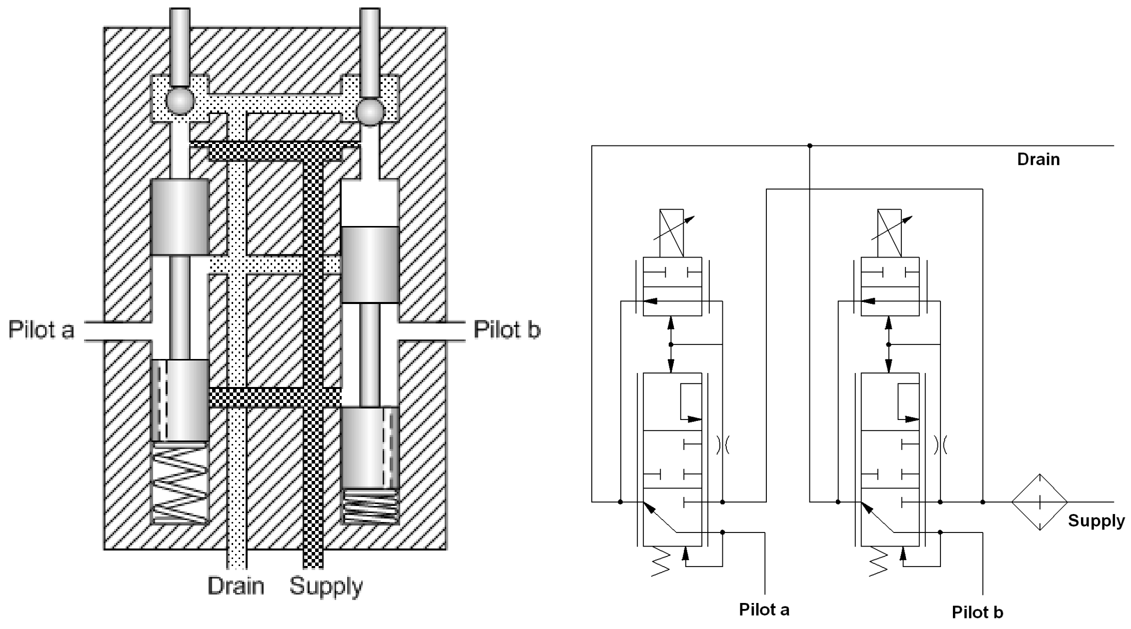

Figure 1.36. Pilot Valve.

The pilot valve is shown in Figure 1.36. Each pilot pressure hose can be considered as a variable-sized control volume with two variable orifices: one that opens the control volume to supply and one that opens it to drain. The volume of the pilot pressure hose changes as the main spool moves as seen in Figure 1.35. It is apparent that a detailed pilot pressure model could easily be 3rd order or higher if the dynamics of the solenoid, ball valve, pilot spool, main spool, and oil compressibility are accounted for. However, since an empirical model is used, the higher order dynamics will be ignored and the *RLS* routine will attempt to capture the dynamics dictated by the most dominant pole.

The *RLS* algorithm developed in the previous sections has been configured for a single input/single output system. However, the pilot pressure in one of the pilot hoses is dependant on the supply and drain pressures and the current supplied to each of the solenoids. A change in the pressure in one pilot hose could move the main spool which changes the oil volume and thereby the pressure in the other pilot hose. To consider this as a single input/single output system a few assumptions are made. The supply pressure is assumed constant. This is a fairly valid assumption since the accumulator attenuates spikes in the supply pressure. The drain pressure is also assumed to be constant since it is an unrestricted path to the tank.

The objective of pilot operation is to control the force exerted on the main valve's spool. Since the spool has equal areas on each end, the difference in the pilot pressures is directly proportional to the force on the spool. Because of this, the difference in the pilot pressures (labeled *pilot a* and *pilot b* in Figure 1.35 and Figure 1.36) will be considered the output. Actuating both solenoids simultaneously is counterproductive since pressure difference is the only thing of concern. All of the positive input commands can be given to one solenoid and the negative input commands can be given to the other. This reduces the model to a single input and single output.

Figure 1.37 shows the input-output curve of the pilot valve for a sinusoidal command current (in mA) of $v(t) = 1200\sin(t)$ for 60 seconds of data (approximately 10 cycles). The curve has nearly the same shape as the curve in Figure 1.4 indicating that it should be possible to fit the pilot valve to a first order system with an input deadzone. There does not appear to be any deadband. The deadzone appears to be within the range between -500 and 500. The initial switching condition c_{l*} was chosen as -500 and c_{r*} was chosen as 500. The initial covariance matrix P was chosen as diag[1000 1000 1000]. The forgetting factor λ was chosen as 0.99. The initial estimation vector $\boldsymbol{\eta}$ was chosen as [1 1 1]. The initial output was 0.

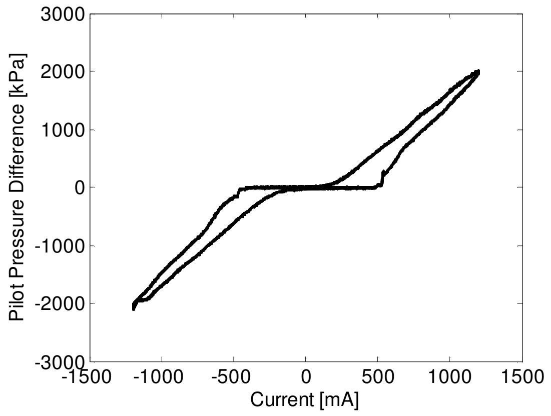

Figure 1.37. Pilot Valve Pressure Difference Versus Current.

Imposing saturation on the estimates $\boldsymbol{\eta}$ keeps the estimated output within a much more realistic range. Limits of 0 to -1 were imposed upon $a_{1,r}$ and $a_{1,l}$. Limits of 0.0001 to 2 were imposed upon $b_{1,r}$ and $b_{1,l}$. If the lower limit would have been chosen at exactly zero, it could cause an error (i.e., division by zero) when calculating c_r and c_l. Limits of 0 to 500 and -500 to 0 where imposed upon e_r and e_l. The resulting actual and simulated outputs are shown in Figure 1.38.

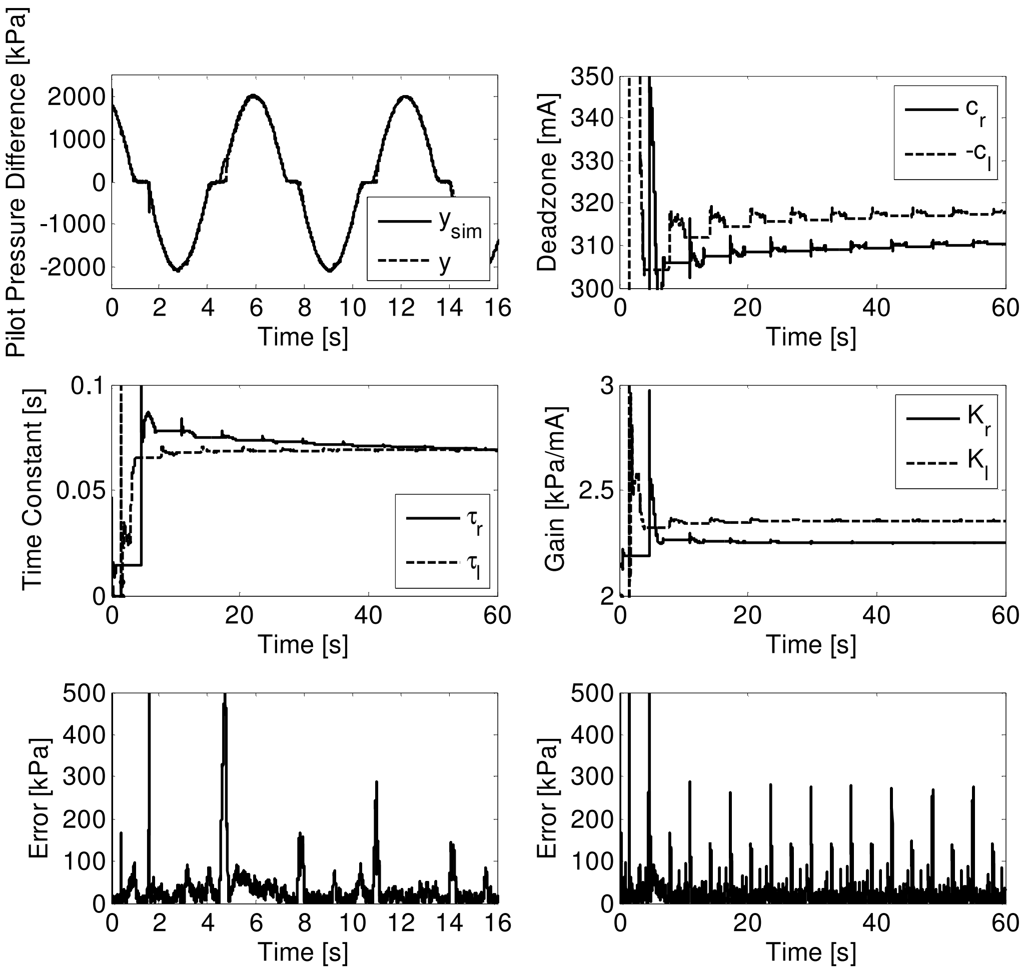

Figure 1.38. Results of Online Estimation of Pilot Pressure Difference Dynamics and Deadzone for a Sinusoidal Input.

Several more experiments were conducted to examine the algorithm at other frequencies, amplitudes, and waveforms. Three tests were conducted for each permutation of 1, 2, 5, and 10 rad/s frequencies and 600, 800, 1200, 2000 mA amplitudes for sine, square, and triangle waves and twenty tests were conducted for a randomized waveform, totaling 164 tests. The root mean square error of the estimated output compared to the actual output was calculated and normalized about the maximum absolute value of the actual output.

Figure 1.39 shows that the error was below 10 percent for most runs. The results for the hydraulic test bench appear to be more frequency dependent than for the motor driven cart. This is understandable since many higher order effects were ignored. The largest error was observed for a triangle wave at low amplitude and high frequency. This is due to the fact that current was barely large enough to induce any pressure change. This is not as much of a problem with the sine wave or square wave because their signals stay above the deadzone estimate (c_{r*} and $-c_{l*} = 500$) longer than for a triangle wave. For the triangle wave with amplitude of 600 mA and frequency of 10 rad/s, the estimator shuts off before the pressure begins to react to the current's change in slope as shown in Figure 1.40.

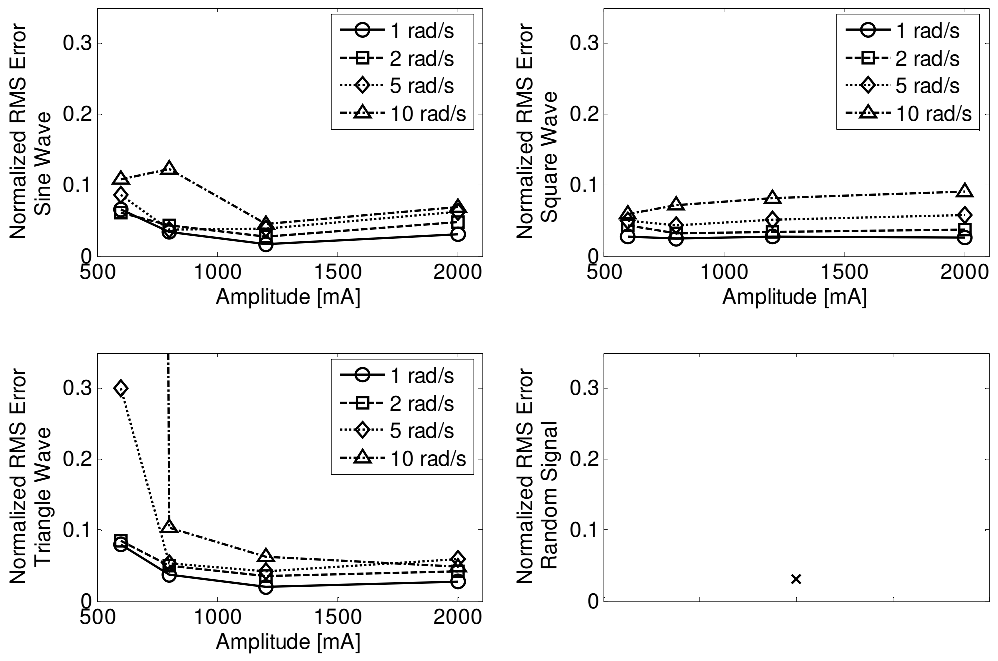

Figure 1.39. Normalized Root Mean Square Error between Simulated and Experimental Pilot Pressures.

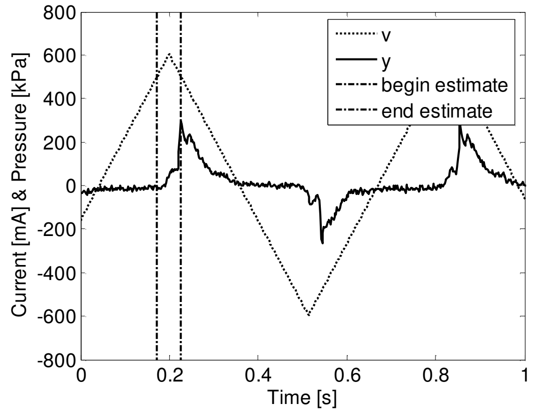

Figure 1.40. Estimation is not Active Long Enough to Capture Dynamics for the 600 mA Amplitude, 10 rad/s Triangle Wave.

1.6.3. Modeling Velocity Of A Free Cart Loosely Coupled To A Motorized Cart. The velocity of a motorized cart was successfully modeled in Section 1.1.1 as a first order transfer function with a deadzone at the output. Now a motorized cart that is coupled to a second free moving cart with a very loose coupling is considered as shown in Figure 1.41. The velocity of the free cart is still modeled as a first order transfer function with a deadzone at the output. However, the position of the cart has a deadband due to the loose coupling. The deadband parameter *d* as defined in equation (5) is equal to half the range of free motion, which is the slot length minus the pin diameter.

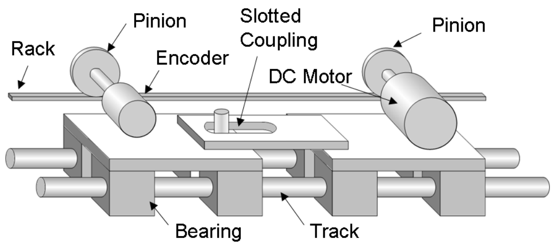

Figure 1.41. Two Carts with Slotted Coupling.

The model shown in Figure 1.43 is created with *u* the input voltage and $\dot{y}$ the free cart's velocity. The intermediate states *v*, $\dot{w}$, and *w* are unobservable where *v* is the theoretical velocity of the motorized cart in the absence of deadzone, $\dot{w}$ is the velocity of the motorized cart, and *w* is the position of the motorized cart. Certain assumptions dictated by the deadband model from equation (4) are made. First, it is assumed the motor driven cart's velocity is not effected by the free cart. If the velocity of the motor driven cart changes depending upon whether or not it is engaged with the free cart, separate dynamics for each condition would have to be considered. Second, it is assumed the pin sliding within the slotted coupling absorbs all of the contact impact between itself and the ends of the slot. If the end of the slot hits the pin with significant force, the connecting pin may oscillate back and forth within the length of the slot. Third, it is assumed the free cart only moves when it is pushed or pulled by the motor driven cart. Therefore, there has to be sufficient friction within the bearings to quickly stop the cart when the pin loses contact with the coupling. Fourth, the motor driven cart velocity must be immediately inherited by the free cart upon contact of the pin and the ends of the slot. These four assumptions can be summed up with one comprehension assumption: the free cart has no mass. Figure 1.42 shows ten cycles of the free cart velocity versus the input voltage to the motor driven cart. With the input voltage used in Figure 1.42, the system appears to fit the model in Figure 1.43. The cart is stationary throughout the deadzone but almost instantaneously assumes the velocity of the motor driven cart.

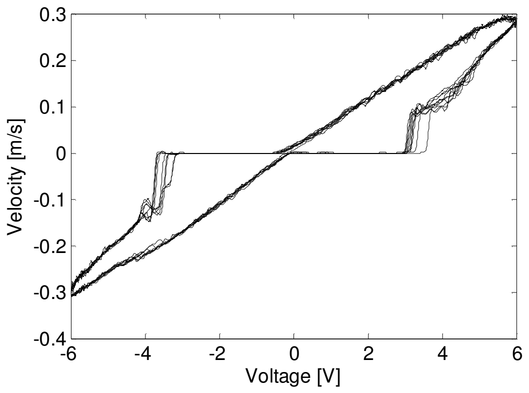

Figure 1.42. Velocity Versus Voltage for Free Cart With *v*(*t*) = 6sin(2*t*).

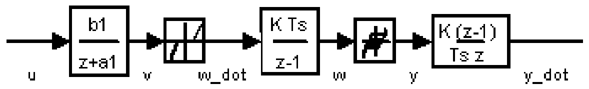

Figure 1.43. Free Cart Velocity Model.

Using the methodology presented in Section 1.4, a switching *RLS* algorithm is derived. First deriving the velocity of the motor driven cart $\dot{w}$

$$\dot{w}(k)=\begin{cases} a_{1,l}\dot{w}(k-1)+b_{1,l}u(k-1)-c_l(1+a_{1,l}) & for \quad \dot{w}(k)<0 \\ a_{1,r}\dot{w}(k-1)+b_{1,r}u(k-1)-c_r(1+a_{1,r}) & for \quad \dot{w}(k)>0 \end{cases}$$

(38)

Using the backwards difference method from equation (26), the motor driven cart's position, *w*, is

$$\frac{w(k)-w(k-1)}{T}=\begin{cases} a_{1,l}\left(\frac{w(k-1)-w(k-2)}{T}\right)+b_{1,l}u(k-1)-c_l(1+a_{1,l}) & for \quad w(k)<w(k-1) \\ a_{1,r}\left(\frac{w(k-1)-w(k-2)}{T}\right)+b_{1,r}u(k-1)-c_r(1+a_{1,r}) & for \quad w(k)>w(k-1) \end{cases}$$

(39)

$$w(k)=\begin{cases} (a_{1,l}+1)w(k-1)-a_{1,l}w(k-2)+b_{1,l}Tu(k-1)-c_lT(1+a_{1,l}) & for \quad w(k)<w(k-1) \\ (a_{1,r}+1)w(k-1)-a_{1,r}w(k-2)+b_{1,r}Tu(k-1)-c_rT(1+a_{1,r}) & for \quad w(k)>w(k-1) \end{cases}$$

(40)

Using the deadband definition in equation (5) and taking the backwards difference from equation (26) shows that the free cart velocity $\dot{y}$ is

$$\dot{y}(k)=\frac{y(k)-y(k-1)}{T}=\begin{cases} \frac{(w(k)+d)-(w(k-1)+d)}{T} & = \dot{w}(k) & for & w(k)<w_l(k) \\ & = 0 & for & w_l(k)\le w(k)\le w_r(k) \\ \frac{(w(k)-d)-(w(k-1)-d)}{T} & = \dot{w}(k) & for & w(k)>w_r(k) \end{cases}$$

(41)

Therefore, the free cart's velocity $\dot{y}$ is equal to the motor driven cart's velocity $\dot{w}$ outside of the deadband and the free cart is motionless within the deadband. Since the free cart's velocity is either equal to the motor driven cart's velocity or zero, it is impossible to estimate the deadband by examining the free cart's velocity alone. Therefore the motor driven cart's velocity $\dot{w}$, which is not directly observable, is assumed to be equal to the free cart's velocity $\dot{y}$ whenever the free cart is moving. Using the motor driven cart's velocity, the parameters $a_{1,r}$, $a_{1,l}$, $b_{1,r}$, $b_{1,l}$, c_r, and c_l are determined using the methodology from Section 1.1.1 regarding the estimation of the velocity of a motor-driven cart. After the estimation of $\dot{w}$ converges, it is integrated to estimate w. The deadband, d, can be determined from y and the estimate of w because $y - w = d$ for $\dot{y}(k) > 0$ and $y - w = -d$ for $\dot{y}(k) < 0$ from equations (5) and (6), respectively. Integrating $\dot{w}$ is problematic since the initial condition is unknown and several inaccurate estimations of $\dot{w}$ are generated before the algorithm converges to an accurate value.

The motor driven and free cart masses are 7.35 and 0.42 kg, respectively: a ratio of 17.5. Also, to avoid high impact forces between the pin and the slot, only low voltages (3, 4, and 6 volts) and low frequencies (2, 4, 6, and 8 rad/s) were considered. Three tests were conducted for each permutation of sine, square, and triangle waves, frequencies of 2, 4, 6, and 8 rad/s, and voltages of 3, 4, and 6 V, for a total of 108 tests.

The results were best for the sine and triangle waves at low frequencies with errors as low as three percent. At higher frequencies the impact speeds between the slot and pin are faster resulting in motion of the free cart even when the motor driven cart was not pushing or pulling on it. The square wave results were far worse than those for the triangle or sine waves. This is most likely the result of lack of excitation. The estimation algorithm was only active when the free cart velocity was non-zero. However, due to the "instantaneous" jump in velocity, accelerations over 1 m/s^2 were excluded. In the case of the square wave, the free cart velocity reaches is close to the steady-state value before the estimation algorithm is activated. This acceleration condition can cause the exclusion of some of the dynamic response data to be lost during the fast acceleration of the free cart in response to a stepped input voltage. As mentioned in Section 1.1.1, the inputs to the estimation algorithm are not persistently exciting enough to generate an accurate estimation.

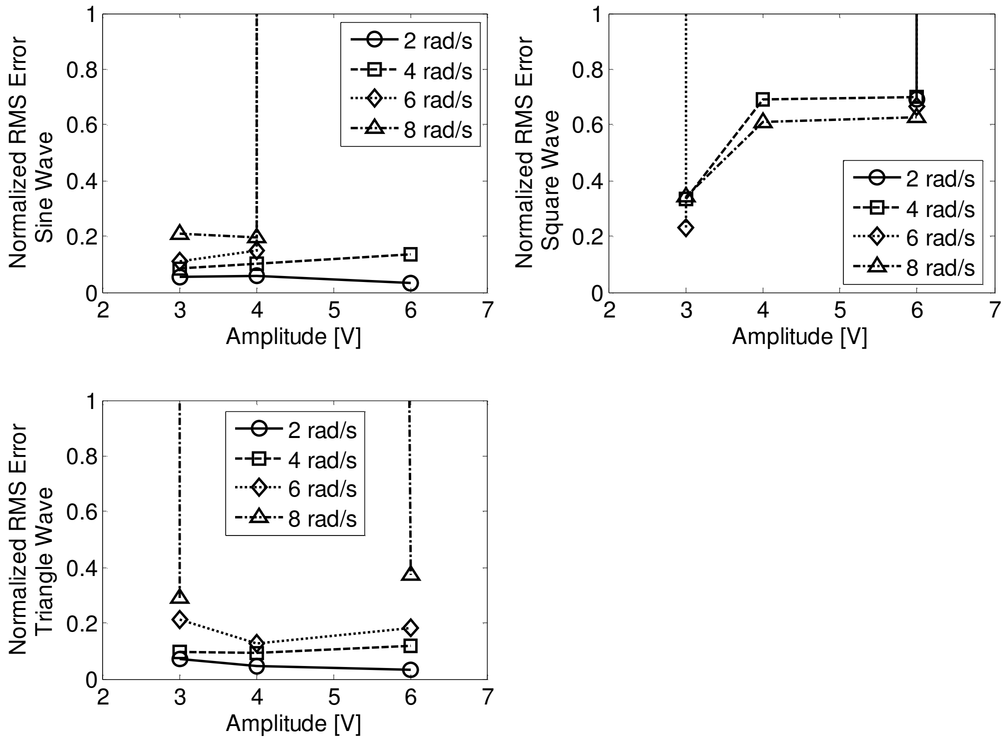

Figure 1.44. Normalized Root Mean Square Error of the Simulated Output of the Free Cart Velocity.

It was observed that when the simulated motor driven cart velocity matched the actual free cart velocity with small error, this error was achieved within approximately five seconds. A time of 7.5 seconds was chosen as the time to begin the deadband estimation by comparing the free cart position *y* to the estimated motor driven cart position *w*. As observed in previous examples and in Figure 1.45 and Figure 1.46, the estimation error is very large momentarily at approximately 0.3 seconds when the estimation begins for positive velocities. The estimation error is also large momentarily at approximately 2.0 seconds when the estimation begins for negative velocities. A spike in the velocity is observed at approximately 7.5 seconds and again at 8 seconds as a result of the deadband estimation beginning. After 8 seconds the simulation matches the actual data to within approximately 0.02 m/s with the exception of spikes that occur when the free cart assumes the velocity of the motor driven cart.

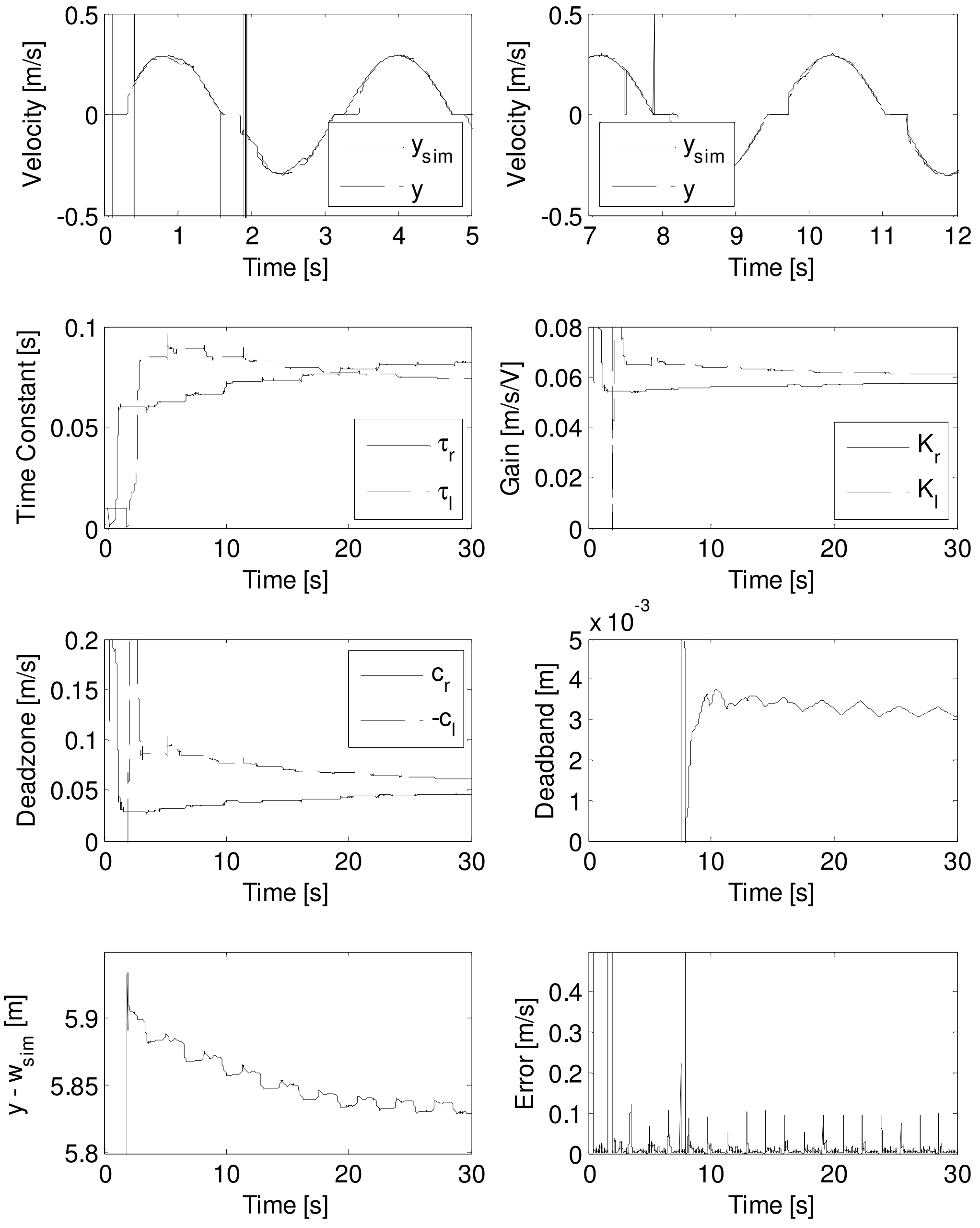

Figure 1.45. Free Cart Parameters for $v(t) = 6\sin(2t)$.

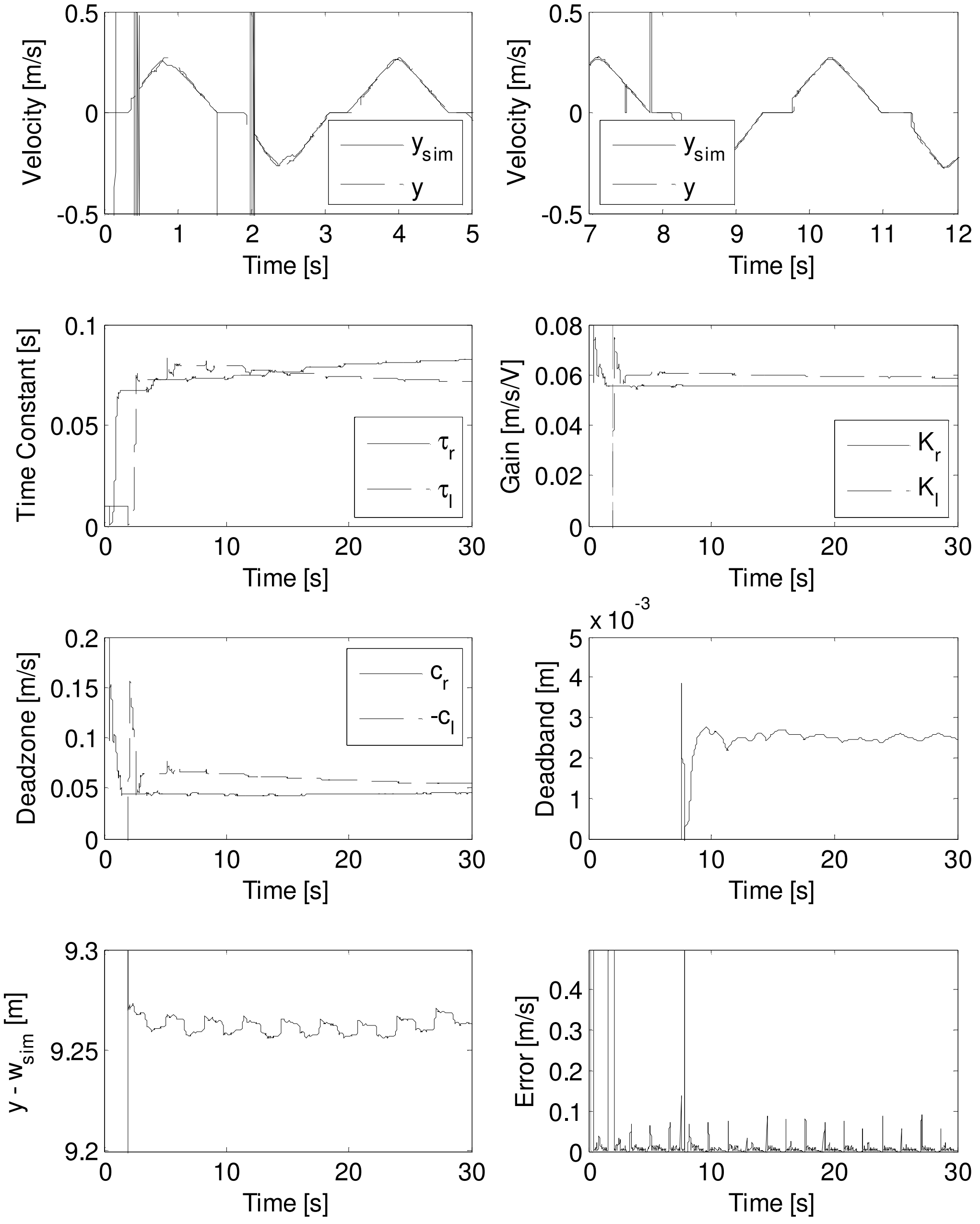

Figure 1.46. Free Cart Parameters for *v*(*t*) = 6triangle(2*t*).

1.7. SUMMARY AND CONCLUSIONS

Nonlinearities such as deadzones and deadbands are present in many mechanical systems. While many of these nonlinear parameters are unknown, they can be determined if the general nature of the system is known. Section 1.3 demonstrates the response of a first order system with deadzone and deadband nonlinearities to help identify which nonlinearities are present and how they should be modeled. Section 1.4 derives an *RLS* estimation law for a linear system, a linear system with an input deadzone, and a linear system with an input deadzone and output deadband. Section 1.5 demonstrates that the parameters converge perfectly to their actual values in simulation. Section 0 implemented the *RLS* estimation laws developed in Section 1.4 on a DC motor driven cart, a hydraulic pilot pressure, and a free cart loosely coupled to a DC motor driven cart.

The results for the DC motor driven cart were very good. The simulated velocity had an error of less than 10% for 94% of the experiments. The results for the hydraulic pilot pressure were similar with 91% of the experiments having less than 10% error between the actual and simulated pressure. A few experiments of the pilot pressure did not converge to accurate values because the estimation did not remain active long enough to capture the system dynamics. The results for the free cart loosely coupled to DC motor driven cart had estimation errors as low as three percent, but only for inputs with low frequencies. At higher frequencies the deadband model fails to describe impact forces between the two carts. For all three experimental scenarios the square waveform performed the worst. For most experiments, the *RLS* algorithm was still able to create a simulated output that was close to the experimental output when a square wave was input; however, it frequently determined inaccurate parameter values. In many cases the square wave caused the gain to be underestimated and compensated for it by allowing the deadzone terms to be negative.

This method for determining nonlinearities and plant dynamics worked well because generic equations for the system were known and only coefficients were needed. Further work is needed to apply these techniques to higher order systems, to use nonlinearities other than deadzone and deadband, and to incorporate the estimations in an adaptive control algorithm.

1.8. NOMENCLATURE

a_1 = coefficient of $y(k\text{-}1)$

$a_{1,l}$ = coefficient of $y(k\text{-}1)$ for $v < 0$

$a_{1,r}$ = coefficient of $y(k\text{-}1)$ for $v > 0$

$a_{1,l+}$ = coefficient of $y(k\text{-}1)$ for $v < 0,\ \dot{y} > 0$

$a_{1,l-}$ = coefficient of $y(k\text{-}1)$ for $v < 0,\ \dot{y} < 0$

$a_{1,r+}$ = coefficient of $y(k\text{-}1)$ for $v > 0,\ \dot{y} > 0$

$a_{1,r-}$ = coefficient of $y(k\text{-}1)$ for $v < 0,\ \dot{y} < 0$

b_1 = coefficient of $u(k\text{-}1)$

$b_{1,l}$ = coefficient of $u(k\text{-}1)$ for $v < 0$

$b_{1,r}$ = coefficient of $u(k\text{-}1)$ for $v > 0$

$b_{1,l+}$ = coefficient of $u(k\text{-}1)$ for $v < 0,\ \dot{y} > 0$

$b_{1,l\text{-}}$ = coefficient of $u(k\text{-}1)$ for $v < 0,\ \dot{y} < 0$

$b_{1,r+}$ = coefficient of $u(k\text{-}1)$ for $v > 0,\ \dot{y} > 0$

$b_{1,r\text{-}}$ = coefficient of $u(k\text{-}1)$ for $v > 0,\ \dot{y} < 0$

c_l = negative deadzone offset, $c_l < c_r$

c_{l*} = negative deadzone offset guess, necessary condition: $c_{l*} < c_l$

c_r = positive deadzone offset, $c_r > c_l$

c_{r*} = positive deadzone offset guess, necessary condition: $c_{r*} > c_r$

d = deadband offset

e = lumped parameter of constant nonlinear terms

I = identity matrix, subscript indicates dimensions

K = steady-state gain

k = iteration

t = time

T = sample time

U, u = input into linear portion of plant, output of deadzone

V, v = desired control input

W, w = input into deadband, output of linear portion of plant

Y, y = plant output

z = Z-transform operator

$\boldsymbol{\eta}$ = unknown parameter vector

$\hat{\boldsymbol{\eta}}$ = unknown parameter estimate vector

λ = forgetting factor

τ = time constant

$\boldsymbol{\varphi}$ = recursive least squares regression vector

1.9. APPENDIX

1.9.1. Derivation Of The *M*-Point Moving Average Filter's Variance.

An *M*-point moving average filters the data by averaging the current point with the *M*-1 preceding terms. If it is filtering Gaussian noise, each point is independent of the point before and after it.[1] Therefore, it is the same as averaging any *M*-number of Gaussian distributed sets and it does not matter that it is the same set shifted multiple times. The sample variance S_N^2 is given by

$$\mathrm{var}(X) = \frac{1}{N}\sum_{i=1}^{N}\left(X_i - \overline{X}\right)^2 \tag{42}$$

where X is the sample set with mean $\overline{X}$ having N terms given by

$$X = [X_1 \quad X_2 \quad \ldots \quad X_N] \tag{43}$$

The covariance between X and a set Y is given by

$$\mathrm{cov}(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right) \tag{44}$$

The covariance is a measure of how well two or more sets of random variables are correlated. For two uncorrelated sets, such as two sets of Gaussian noise, the covariance is zero. Given that each point in X is filtered data from an *M*-point moving average has been implemented on x, the terms in X become

[1] Not all noisy signals can be assumed to be a Gaussian distributed random number.

$$
\begin{aligned}
X_1 &= \frac{x_{1,k} + x_{1,k-1} + \ldots + x_{1,k-M-1}}{M} \\
X_2 &= \frac{x_{2,k} + x_{2,k-1} + \ldots + x_{2,k-M-1}}{M} \\
&\vdots \\
X_N &= \frac{x_{N,k} + x_{N,k-1} + \ldots + x_{N,k-M-1}}{M}
\end{aligned}
\tag{45}
$$

Substituting equation (45) into equation (42)

$$
\operatorname{var}(X) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_{i,k} + x_{i,k-1} + \ldots + x_{i,k-M-1}}{M}\right)^2
\tag{46}
$$

Since this is used to determine the sign of derivative, the mean $\overline{X}$ is dropped because the derivative is the differentiation of a noisy yet static signal. Expanding Equation (46) yields

$$
\operatorname{var}(X) = \begin{array}{l}
\frac{1}{M^2N}\sum_{i=1}^{N}\left(x_{i,k}\right)^2 + \frac{1}{M^2N}\sum_{i=1}^{N}x_{i,k}x_{i,k-1} + \ldots + \frac{1}{M^2N}\sum_{i=1}^{N}x_{i,k}x_{i,k-M-1} \\
+\frac{1}{M^2N}\sum_{i=1}^{N}x_{i,k-1}x_{i,k} + \frac{1}{M^2N}\sum_{i=1}^{N}\left(x_{i,k-1}\right)^2 + \ldots + \frac{1}{M^2N}\sum_{i=1}^{N}x_{i,k-1}x_{i,k-M-1} \\
\vdots \\
\frac{1}{M^2N}\sum_{i=1}^{N}x_{i,k-M-1}x_{i,k} + \frac{1}{M^2N}\sum_{i=1}^{N}x_{i,k-M-1}x_{i,k-1} + \ldots + \frac{1}{M^2N}\sum_{i=1}^{N}\left(x_{i,k-M-1}\right)^2
\end{array}
\tag{47}
$$

Each term with a square is in the form of the variance shown in Equation (42) with a coefficient of $1/M^2$. Since the variance before the M-point moving average was taken is known, all of these terms are known. Furthermore, the variance is constant for the whole sample meaning that all of the terms with squares are equal. Each term without a square is in the form of the covariance. Since each point is independent of the points before and after it, the covariance is zero. There are M number of diagonal terms. Therefore Equation (47) reduces to

$$\mathrm{var}(X) = \frac{1}{M}\mathrm{var}(x)$$

(48)

1.9.2. Derivation Of The Variance Of $\dot{y}$. The numerical derivative $\dot{y}$ is computed from equation (26). The terms of the sample vector $\dot{X}$ defined in equation (43) become

$$\begin{aligned}\dot{X}_1 &= \frac{x_1 - x_0}{T}\\ \dot{X}_2 &= \frac{x_2 - x_1}{T}\\ &\vdots\\ \dot{X}_N &= \frac{x_N - x_{N-1}}{T}\end{aligned}$$

(49)

Substituting into Equation (42) and dropping $\overline{X}$ because it is zero yields

$$\mathrm{var}(X) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - x_{i-1}}{T}\right)^2$$

(50)

Expanding it becomes

$$\mathrm{var}(X) = \frac{1}{NT^2}\sum_{i=1}^{N}x_i{}^2 + \frac{1}{NT^2}\sum_{i=1}^{N}x_{i-1}{}^2 - \frac{1}{NT^2}\sum_{i=1}^{N}x_i x_{i-1}$$

(51)

Once again for Gaussian noise there is no correlation between x_i and $x_{i\text{-}1}$ so the third term is zero. Since the variance of x_i and $x_{i\text{-}1}$ are the same Equation (51) becomes

$$\mathrm{var}(X) = \frac{2}{T^2}\mathrm{var}(x)$$

(52)

1.10. BIBLIOGRAPHY

[1] Choudhury, M.A.A.S.; Thornhill, N.F.; Shah, S.L. “Modelling Valve Stiction.” Control Engineering Practice. Vol. 13. 2005. pp.641-58.

[2] Gang Tao; Kokotovic, P.V., "Adaptive control of plants with unknown dead-zones," *Automatic Control, IEEE Transactions on* , vol.39, no.1, pp.59-68, Jan 1994.

[3] Bai, E-W.; Tao, G. ed.; Lewis. F. ed. “Adaptive Dead Zone Inverses for Possibly Nonlinear Control Systems.” *Adaptive Control of Nonsmooth Dynamic Systems.* Springer. 2001. pp.31-47.

[4] Xing-Song Wang; Chun-Yi Su; Hong, H., "Robust adaptive control of a class of nonlinear systems with unknown dead-zone," *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on* , vol.2, no., pp.1627-1632 vol.2, 2001.

[5] Ibrir, S.; Wen Fang Xie; Chun-Yi Su, "Adaptive tracking of a class of uncertain nonlinear systems subject to unknown dead-zone input nonlinearities: the symmetric and the non-symmetric cases," *American Control Conference, 2006* , vol., no., pp. 6 pp.-, 14-16 June 2006.

[6] Lewis, F.L.; Woo Kam Tim; Li-Zin Wang; Li, Z.X., "Deadzone compensation in motion control systems using adaptive fuzzy logic control," *Control Systems Technology, IEEE Transactions on* , vol.7, no.6, pp.731-742, Nov 1999.

[7] Jun Oh Jang, "A deadzone compensator of a DC motor system using fuzzy logic control," *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* , vol.31, no.1, pp.42-48, Feb 2001.

[8] Jun Oh Jang, "Deadzone compensation of an XY-positioning table using fuzzy logic," *Industrial Electronics, IEEE Transactions on* , vol.52, no.6, pp. 1696-1701, Dec. 2005.

[9] Jian-Xin Xu; Jing Xu; Tong Heng Lee, "Iterative learning control for a linear piezoelectric motor with a nonlinear input deadzone," *Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on* , vol.2, no., pp. 1001-1006 Vol.2, 2-4 Sept. 2004.

[10] Jian-Xin Xu; Jing Xu; Tong Heng Lee, "Iterative learning control for systems with input deadzone," *Decision and Control, 2004. CDC. 43rd IEEE Conference on* , vol.2, no., pp. 1307-1312 Vol.2, 14-17 Dec. 2004.

[11] Jian-Xin Xu; Jing Xu; Tong Heng Lee, "Iterative learning control for systems with input deadzone," *Automatic Control, IEEE Transactions on* , vol.50, no.9, pp. 1455-1459, Sept. 2005.

[12] Tao, G.; Kokotovic, P.V., "Adaptive control of system with unknown output backlash," Automatic Control, IEEE Transactions on , vol.40, no.2, pp.326-330, Feb 1995

[13] Grundelius, M.; Angeli, D., "Adaptive control of systems with backlash acting on the input," *Decision and Control, 1996., Proceedings of the 35th IEEE* , vol.4, no., pp.4689-4694 vol.4, 11-13 Dec 1996.

[14] Gebler, D.; Holtz, J., "Identification and compensation of gear backlash without output position sensor in high-precision servo systems," *Industrial Electronics Society, 1998.*

[15] Woo, K.T.; Li-Xin Wang; Lewis, F.L.; Li, Z.X., "A fuzzy system compensator for backlash," *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on* , vol.1, no., pp.181-186 vol.1, 16-20 May 1998.

[16] Ahmad, N.J.; Khorrami, F., "Adaptive control of systems with backlash hysteresis at the input ," *American Control Conference, 1999. Proceedings of the 1999* , vol.5, no., pp.3018-3022 vol.5, 1999.

[17] Tao, C.W., "Fuzzy control for linear plants with uncertain output backlashes ," *Systems, Man, and Cybernetics, Part B, IEEE Transactions on* , vol.32, no.3, pp.373-380, Jun 2002.

[18] Zabiri, H.; Samyudia, Y., "A self detection and compensation of actuator backlash in the framework of constrained MPC design," *Control Conference, 2004. 5th Asian* , vol.2, no., pp. 1186-1194 Vol.2, 20-23 July 2004.

[19] Lagerberg, A.; Egardt, B., "Backlash Estimation With Application to Automotive Powertrains," *Control Systems Technology, IEEE Transactions on* , vol.15, no.3, pp.483-493, May 2007.

[21] Parr, A., 1998, *Hydraulics and Pneumatics: A Technician's And Engineer's Guide*, Butterworth-Heinemann, Oxford.

2. EMULATION USING xPC TARGET.

J.J. Lentz, R.G. Landers, and K. Krishnamurthy

Department of Mechanical and Aerospace Engineering

Missouri University of Science and Technology

1870 Miner Circle

Rolla, Missouri 65409-0050

{Jeffrey.J.Lentz, landersr, kkrishna}@mst.edu

2.1. ABSTRACT

The Mathworks xPC Target environment allows for rapid prototyping of a controller without the burden of low level coding. This allows for validation of the control system logic early in the design process. It also provides proof that the algorithm can be executed in real-time. With xPC Target, algorithms are developed in a simulation environment and automatically converted into real-time executable code. This paper demonstrates the development of a Master/Slave control system for a hydraulic test stand using xPC Target. The Master and Slave Electronic Control Units (ECUs) are emulated using two target PCs running the xPC Target kernel communicating with each other over a Controller Area Network (CAN). The emulated and simulated results matched perfectly. Then the emulated Master ECU is used to control the hydraulic test stand by sending current commands and receiving pressure sensor data from the Slave ECU. The task execution time of the emulated Master ECU was the same regardless of whether it was controlling the emulated Slave ECU or the actual Slave ECU. The accuracy of the

emulation was shown to be only limited by the accuracy of the hydraulic system plant model.

2.2. INTRODUCTION

In systems engineering it is good practice to validate system components individually before integrating them. For example, when designing a hydraulic excavator, the engine, hydraulic system, and structure should be individually validated before creating a prototype. Control software design and development is similar in that respect. It is desirable to validate a control system's logic and execution time in simulated and emulated environments before going through the time-consuming process of creating embedded code on an Electronic Control Unit (ECU). As vehicle control systems are becoming increasingly complex in order to accommodate more special features, to reduce wiring, and to modularize systems, a need arises for individually validating all levels of the control system's hierarchy.

The xPC Target environment developed by The Mathworks, Inc. allows control system engineers to perform Hardware-in-the-Loop (HIL) testing using an emulated version of the ECU. With the xPC Target environment, a Simulink model can be used to auto-generate real-time executable code which resides on a target PC. In the example shown in this paper, target PCs are used to emulate ECUs. Since the most popular communication protocol is the Controller Area Network (CAN), each target PC is equipped with a CAN board to communicate with any other device with CAN capability. In this case, a target PC will communicate with an ECU or another target PC over CAN.

An example is given to demonstrate the steps for designing, validating, and implementing a controller for a hydraulic test stand. For demonstration purposes, it is required that the control algorithm reside on a Master ECU which is used to control a Slave ECU. The Master ECU commands the Slave ECU to drive current through the hydraulic solenoid valves and receives pressure sensor information back from the Slave ECU. In the example a system model is created of the hydraulic test bench and a controller is created and simulated in Simulink. The controller model is compiled onto one target PC to emulate the Master ECU called the *Master Emulator*. The plant model is compiled onto another target PC called the *Slave and Plant Emulator* to emulate the Slave ECU and hydraulic system to evaluate the performance of the controller. The target PC emulating the Master ECU will be used to control the actual Slave ECU and hydraulic test stand. The simulated and emulated results matched perfectly. The task execution time of the emulated Master ECU was the same regardless of whether it was controlling the emulated Slave ECU or the actual Slave ECU. The accuracy of the two target PC emulation was shown to be only limited by the accuracy of the hydraulic system plant model, which was embedded within the emulated Slave ECU target PC.

2.3. LITERATURE REVIEW

The emulation of control hardware has been used extensively in industry and academia as a way to prove out design concepts. Several papers have been written on the use of the xPC Target environment to this end. Teng [1] compared xPC Target version 1.0 to two third party systems for automatic real-time code generation from the Matlab/Simulink: Real-Time Toolbox (not to be confused with Real-Time Workshop) and WinCon. He implemented all three systems with PID, LQR, and fuzzy logic controllers on various adaptations of the inverted pendulum problem and concluded that while this initial version of xPC Target is somewhat difficult to use, it offers sampling times of less than a millisecond, which the other systems failed to accommodate. Luce and Rahnamai [2] used the xPC Target environment to implement a PID controller for a Multi-Fan Hovering System for use on a vertical take off and landing vehicle. Shiakolas and Piyabongkarn [3] developed a magnetic levitation controller for undergraduate level controls laboratories. Nilsson [4] compared the Matlab/Simulink and xPC Target environment to their analogous counterparts from National Instruments, MatrixX/SystemBuild and Autocode. He demonstrated the use of both tools via a hydraulic test stand, compared the results of both controllers, and concluded that both perform about the same, however, MatrixX has an easy interface. Leonessa *et al.* [5] used the xPC Target environment on an autonomous surface vessel to follow an unmanned underwater vehicle for the purpose of transmitting GPS locations and controlling the underwater vehicle. Gani and Khan [6] implemented active vibration control of a beam with piezoelectric patches using the xPC Target environment. Shangying *et al.* [7] used Labview as a GUI to tune parameters on a target PC running the xPC Target kernel. The controller was for a 6 degree-of-freedom hydraulic robot. Low *et al.* [8] used the xPC Target environment to perform telemanipulation of a robotic hand. Wei *et al.* [9] performed a haptic learning experiment to investigate how to help stroke victims regain motor skills. Kelper [10] examined in real-time the switching impedance in power electronics using the xPC Target environment. Hassan-Zadeh *et al.* [11] used the xPC Target environment to perform teleoperation of a mobile robot using haptic feedback control. Mazon *et al.* [12] used the xPC Target environment to evaluate the behavior of an artificial neural network. Driscoll [13] used the xPC Target environment to evaluate the effectiveness of emulating a hydraulic cylinder using an electric motor driving a hydraulic motor. Ferreira [14] used the xPC Target environment to experiment with a robot-arm controlled by real-time image processing. Zhu [15] used the xPC Target environment to monitor networks of MEMS sensors. Rahnamai *et al.* [16] used the xPC Target environment to develop an automatic guitar tuner. Cao [17] used the xPC Target environment to control fuel injection on a liquefied petroleum gas engine. Anderson and Stone [18] used the xPC Target environment to control a vertical launching unmanned aerial vehicle. Luqiao *et al.* [19] used the xPC Target environment to control a binocular vision robot used for explosive ordnance disposal (i.e., it is used for removing land mines, bombs, or for counter-terrorism).

The previous works were all similar in that they used xPC Target to either test a controller or to collect data. The target PC interfaced with the hardware using either analog to digital (A/D) conversion, digital to analog (D/A) conversion, and/or counter-timers (C/T). They are also similar in that they used a single target PC. However, in two collaborative works by Quinones-Reyes [20] and Ramirez-Gonzalez [21], three target PCs were employed to control a magnetic levitation system. The three target PCs, communicating over CAN, were needed to perform an “earliest deadline first” algorithm. Variable CAN message transmission rates were used to prioritize messages with an algorithm deciding if the messages were important enough to send. McGowan [22] also used the xPC Target environment in conjunction with CAN communication to control a standby diesel generator. He controlled an engine by sending commands from the target PC to the engine ECU and receiving sensor data. Being a Caterpillar generator set, he used the same Caterpillar RPAC (Rapid Prototyping for Automated Controls) toolbox that is used in this paper.

All of the aforementioned papers demonstrate the utility of using the xPC Target environment to evaluate the effectiveness of a controller. However, [23] demonstrated the feasibility of using the xPC Target system as an emulated system to test real controller hardware. He created a virtual diesel engine contained on a single target PC. The engine governor interfaced with the target PC through digital channels rather than CAN.

This paper is similar to [20] and [21] in that multiple target PCs communicating over CAN will be used; however, one of the target PCs will be an emulated system rather than a controller. This work differs from [23] in that it interfaces with the emulated system via CAN and in that it uses multiple target PCs. This work differs from [22] in that not only are Caterpillar RPAC toolboxes used for a controller, they are reverse engineered to create an emulation of the ECU and the plant being controlled on a second target PC.

2.4. CONTROL ARCHITECTURE

Many systems, such as vehicle control systems, are becoming increasingly complex as it is not feasible for one Electronic Control Unit (ECU) to service all of the controls needs. Networks of ECUs are ubiquitous. For example, today’s automobiles can have over 60 ECUs [24]. Each ECU performs some or all of the following functions: reading sensor data, powering sensors and actuators, transmitting sensor measurements, transmitting actuator commands, executing control algorithms, and reporting faults. Many ECUs are needed to make each subsystem modular. On an automobile the engine, powertrain, steering, braking, and HVAC systems may all have their own ECU or network of ECUs. Airplanes have additional systems which may have their own ECU such as the elevators, rudders, flaps, ailerons, landing gear, etc. Earth moving equipment has additional systems such as hydraulic pumps, motors, cylinders, grippers, hammers, and compactors that may have dedicated ECUs.

An increase in automation is also driving up the number of ECUs on control systems. For example, in the automobile industry automatic features such as traction control systems, headlights that adjust automatically when the vehicle turns, collision prevention systems, and parking guidance systems are becoming increasingly prevalent. The earth-moving equipment market also has emerging technologies such as Laser and GPS grade control systems and automatic compaction technologies. These features require additional computation, sensors, and/or actuators. For marketing reasons, manufacturers tend to sell special features as optional modular packages, which can result in separate ECUs.

Another motivation behind having multiple ECUs is for wiring harness optimization and serviceability. It is easier for a service technician to pull and replace wires on an ECU with a 20-pin connector than on an ECU with a 100-pin connector. If every wire on a machine had to be connected to the same ECU, the resulting bundle of wires might end up being several inches in diameter making it difficult to bend and route. Making several smaller wiring harnesses is typically more manageable than one large harness. The length of wire required can be optimized in situations where the sensors and actuators are spread out from each other. Having an ECU for each cluster of sensors and actuators can be more economical and reliable than having just one ECU connected to every sensor and actuator.

2.5. COMMUNICATION

One challenge of having a large number of ECUs is networking them to share information with each other. For example, an engine on a hydraulic earth-moving machine must know the load on the hydraulic pump to operate at the optimal speed. A hydraulic variable displacement piston pump must know how much flow is requested by the implements to optimize the swash angle. The most prevalent communication protocol is Controller Area Network (CAN) developed by Robert Bosch GmbH. It has been the most widely used communication protocol since the early 1990s, when it was first used by Mercedes-Benz [25]. There are a few other protocols that appear very promising for the future such as Flexray and LIN; however, CAN is still the most prevalent.

The CAN protocol is a rugged two-wire communication. It is robust to electromagnetic interference by sending a high signal and a low signal on a pair of wires, and then subtracting the signals. Typically, the wires are twisted to ensure the interference picked up by one wire is picked up equally by the other. The interference is cancelled when the signals are subtracted. There is 60 ohms of resistance between the high and low signals achieved by placing a 120 ohm terminating resistor across the high and low wires at the two nodes that are farthest apart. CAN broadcasts messages with "identifiers" that identify the data contained within the message. In the standard format, identifiers have 11 bits. Messages are broadcast without an intended destination. Each node on the network filters all messages that are not needed and thereby "listens" for the messages it needs. In the standard CAN format, called the Base Frame Format, each message contains up to eight bytes of data. A 4-bit field called the Data Length Code indicates how many of those bytes are used. The data field bits can be designated in any way desired. Dividing the eight bytes into four 16-bit signals or into eight 8-bit signals is common. The bits used for a given signal need not be consecutive. Assigning signals to their respective bits in the message is called “Bit-Packing.” Similarly, extracting signals from a CAN message is called “Bit-Unpacking.” The data is always sent as unsigned integers so it is necessary to scale the signals before sending and after receiving to use the full range [26].

One example illustrating the use of CAN is the feature of headlights that adjust to turn corners. CAN is used to transmit data between the headlight ECU, the transmission ECU, and the steering system ECU. The headlight ECU needs the vehicle speed from the transmission ECU and it needs the angle of the tires from the Steering ECU to calculate where to point the headlights. Therefore, the headlight ECU only accepts messages known to contain the vehicle speed or tire angle and ignores all other messages regarding impertinent data such as transmission gear, transmission oil temperature, etc. The messages containing vehicle speed and tire angle are most likely shared with other signals. The headlight ECU has to unpack the messages to extract the pertinent data. If the tire angle measurement is transmitted as a 10-bit 0 to 100% PWM, then it is apparent that the bits 0 to 2^{10}-1 (or 0 to 1023) correspond to the 0 to 100% duty cycle. To convert the bits to duty cycle a gain of 100/1024 is used to scale the signal. The software also must convert the signals into engineering units before they are used by the control algorithm.

2.6. XPC TARGET ENVIRONMENT

The xPC Target Environment was developed by The Mathworks, Inc. as a rapid-prototyping tool to allow Simulink and Stateflow models to be executed in real-time and to interface with hardware. Two computers are required for the xPC Target Environment: one, called the *target*, to execute the generated real-time code in the xPC Target kernel and one, called the *host*, to upload the real-time code to the target and to tune parameters. The target and host communicate via TCP/IP or RS-232. The Real-Time Workshop and Stateflow Coder toolboxes generate real-time executable code. Input/Output devices such as CAN, digital to analog, analog to digital, counter-timer, or encoder boards are installed on the target PC. The xPC Target toolbox contains a library of blocks for supported I/O devices, which are inserted in the Simulink model to interface with the I/O devices. The devices must use a PCI, CompactPCI, PC/104, or ISA bus.

The target PC can be made from an ordinary PC by installing PCI I/O boards, creating an xPC boot disk, and booting the PC into the xPC Target environment. There are also several commercially available target PCs tailored to the application in which they are used. The Mathworks, Inc. developed the mobile xPC Targetbox® shown in Figure 2.1. Speedgoat GmbH purchased The Mathworks, Inc.'s xPC Targetbox® product line in February 2007. They also manufacture rack-mounted and desktop target PCs as shown in Figure 2.2 and Figure 2.3, respectively.

Figure 2.1. Mobile Mathworks xPC Targetbox® (270mm x 162mm x 82mm).

Figure 2.2. Rack Mountable Speedgoat Target PC.

Figure 2.3. Desktop Speedgoat Target PC.

2.7. HYDRAULIC TEST BENCH EXAMPLE

The following example demonstrates how the xPC Target environment can be used in conjunction with an electro-hydraulic test bench. In 2005 Caterpillar, Inc. donated eight electro-hydraulic test benches to the University of Missouri – Rolla (now called Missouri University of Science and Technology or Missouri S&T). Each bench (Figure 2.4 and Figure 2.5) consists of an electric motor-driven pump to supply pressure, an electro-hydraulic pilot valve (Figure 2.6), a main valve, an ECU, an xPC Targetbox, and a PC running Matlab with Simulink, Real-Time Workshop, Rapid Prototyping for Automated Controls (RPAC), and xPC Target toolboxes. The RPAC toolbox is a Caterpillar proprietary Matlab toolbox that provides input and output library blocks to interface with a Caterpillar ECU.

To administer hydraulic controls experiments, the electrical portion of the test bench is configured as shown in Figure 2.4. The host PC compiles the model into real-time executable code and uploads it to the target PC via TCP/IP. The target PC communicates with the ECU via a CAN link. The software flashed on the ECU configures it to behave as a "dumb" I/O box. The target PC sends CAN messages to the ECU to command the current drivers and the ECU sends CAN messages back to the target PC containing sensor data. This software architecture is typically referred to as a Master/Slave arrangement because the ECU (the slave) is merely being used for its ability to read the sensors and drive currents, while the target PC (the master) is doing all of the high-level decision-making. In industry a Master/Slave configuration like this may be utilized during the development of two types of systems. The algorithm residing on the target PC will eventually be rewritten in a low level language and 1) embedded onto the ECU, making the ECU autonomous or 2) embedded onto a master ECU while the current ECU remains a slave. The latter of the two will be utilized in the following example in which a controller is developed to regulate the pilot valve's outlet pressure. This section of the paper goes through the following steps to create and validate a controller:

1. Model system
2. Design controller and simulate in Simulink
3. Emulate controller and plant using two target PCs
4. Control real system with target PC

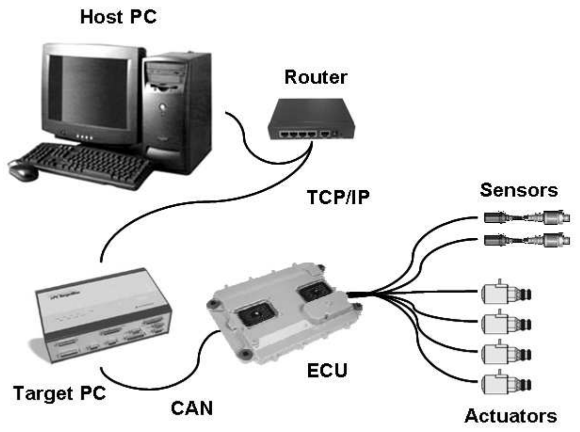

Figure 2.4. Electrical Portion of Electro-Hydraulic Test Bench.

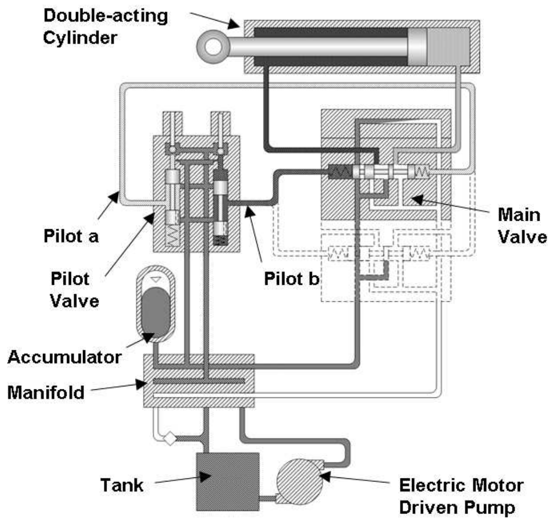

Figure 2.5. Hydraulic Portion of the Electro-Hydraulic Test Bench.

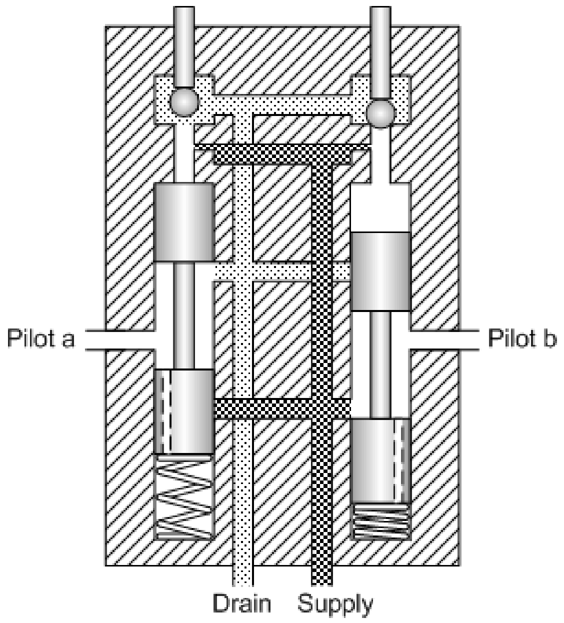

Figure 2.6. Pilot Valve Schematic.

2.7.1. System Model. An empirical model of the hydraulic system is developed from test data with current as the input and pilot pressure as the output. A simple model is constructed in Simulink using the RPAC block as shown in Figure 2.7. The sample time is set to five milliseconds. The pilot pressure reading is converted from percent duty cycle to kPa by subtracting 5.98% duty cycle and multiplying the result by 48.82 kPa/% duty cycle. The RPAC block is configured by the user to select the required inputs and outputs. Within a Simulink s-function, the RPAC block assigns a CAN identifier for each input and output. The RPAC block also sends CAN messages to the ECU at the beginning of execution to tell it which assignments were made. The user does not need to understand CAN to use the RPAC block to communicate with the ECU during this modeling step. A stepped current command signal is sent to one of the pilot valve's solenoids as open loop pressure data is collected. The supply pressure is maintained constant by modulating a variable restrictor valve that connects the supply lines to the return lines.

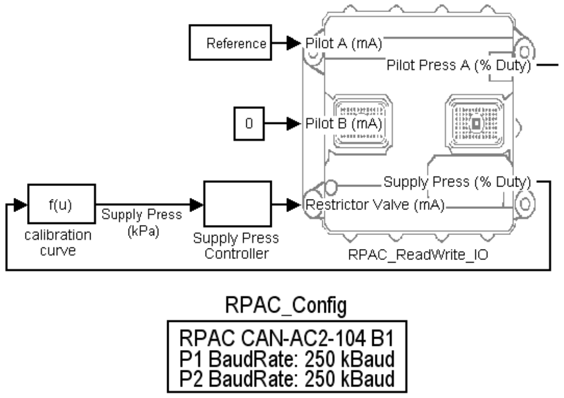

Figure 2.7. Simulink Model Used for Collecting Open Loop Pilot Pressure Data.

Figure 2.8 shows current and pressure time history plots. It can be seen that low levels of current produced no pressure change. This is because the current in the solenoid has to induce a force large enough to alleviate the spring pretension on the pilot spool and has to move the spool a few millimeters before the supply pressure is exposed to the pilot valve's outlet. This is modeled as a deadzone. Figure 2.6 shows that when both sides of the pilot valve are not actuated 100%, there is flow passing through the valve from supply to drain. Since the pilot hoses are connected to the drain at zero current, the pressure in the pilot hoses is nonzero. Averaging the steady-state pressure at zero current yields a pressure 90.1 kPa. Subtracting off this bias zeros the data. Fitting a line to the steady state values of pressure (after zeroing) versus current is a convenient means of determining the steady-state gain and the deadzone as shown in Figure 2.9. The x-intercept is the deadzone, which is 288 mA, and the slope is the steady-state gain, which is 2.15 kPa/mA. The small offsets between rising and falling steady-state pressures, as seen in Figure 2.9, indicates there is little hysteresis and the same model can be used for increasing and decreasing pressures. The rise time appears to become smaller with increased pressure as can be seen in Figure 2.8 and Figure 2.10. This is most likely the result of fluid bulk modulus increasing with increasing pressure, making the oil more incompressible. Since the scope of this paper is emulation of ECUs, not hydraulic system modeling, a single time constant of 0.07 sec is used to characterize the system dynamic response. The time constant was adjusted manually until a good match of the actual data was achieved in simulation. The time constant was selected to be faster than the response at low pressures but slower than the response at high pressures as a compromise. There is also noise in the signal that will be modeled as Gaussian distributed. Selecting a few seconds of steady-state pressure data is sufficient to estimate the standard deviation, which was determined to be 7.28 kPa. The plant model is created in Simulink as shown in Figure 2.11.

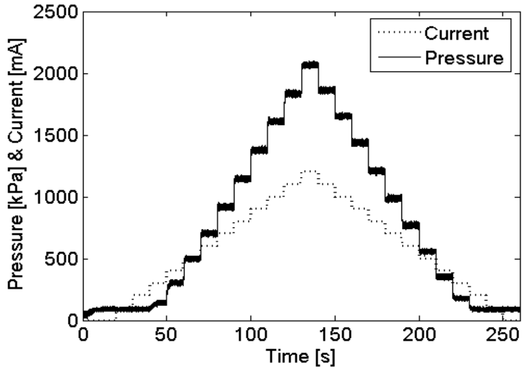

Figure 2.8. Pilot Valve Output Pressure and Input Current Versus Time.

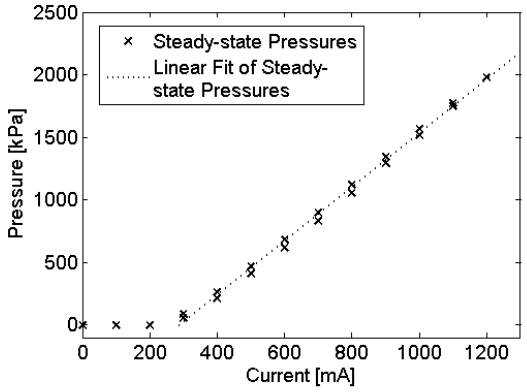

Figure 2.9. Steady-state Pilot Pressure Versus Input Current.

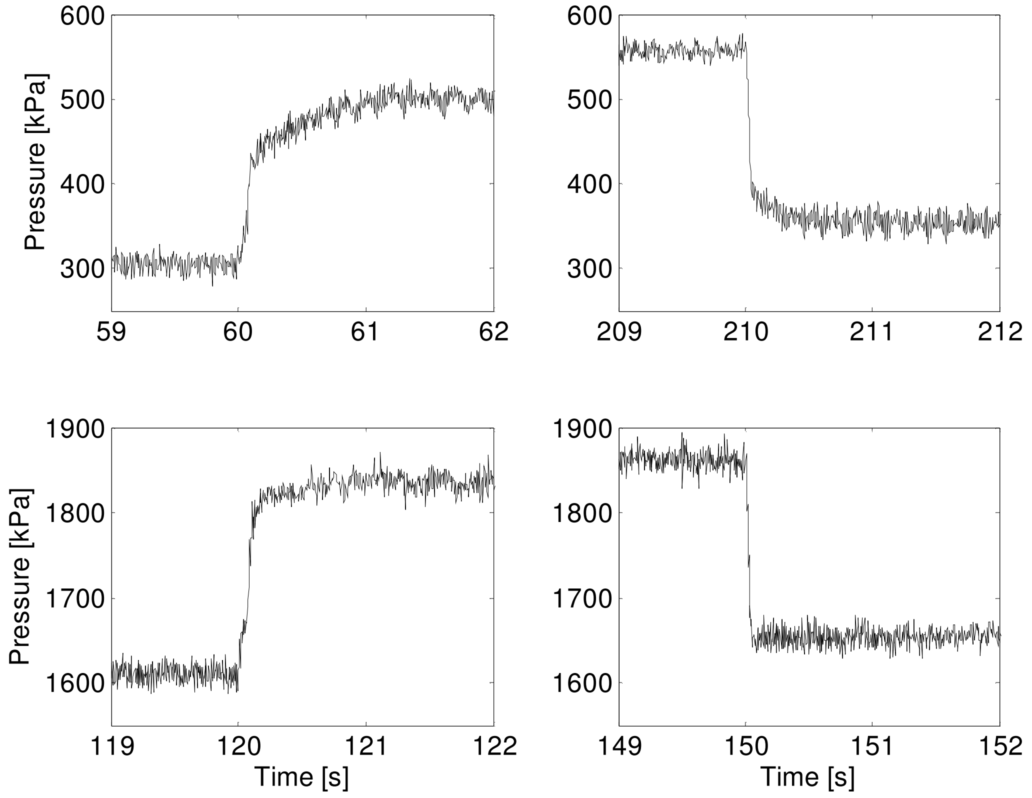

Figure 2.10. Pressure Transient Response (Enlargement of Figure 2.9.)

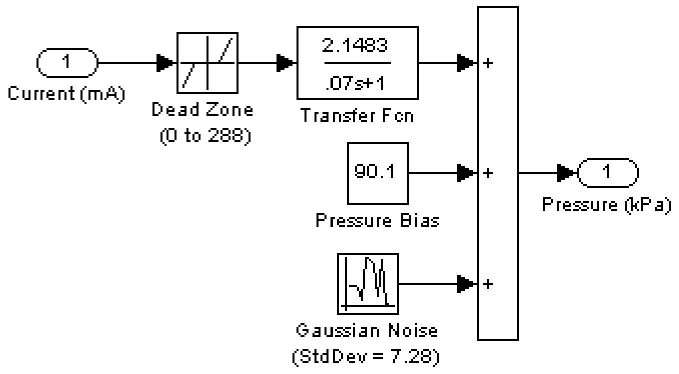

Figure 2.11. Pilot Valve Pressure Plant Model.

2.7.2. Controller Design. Using the model, a PID controller can be tuned to track a square wave, triangle wave, sine wave, and randomized reference. When this controller is implemented on the hydraulic test stand, the supply pressure will be controlled in closed-loop. The closed-loop supply pressure controller is given 20 seconds to stabilize the supply pressure before the pilot pressure controller is initiated. The control signal is offset by a value slightly smaller than the deadzone, known as a deadzone inverse [27], as shown in the controller model in Figure 2.12. The deadzone inverse is chosen about three percent lower than the actual deadzone to prevent unintended actuation should there be a small error in the deadzone estimate. A saturation block is added to insure the current command stays between 0 and 2 A, which are the limits of the ECUs current drivers. The response of the closed loop system as shown in Figure 2.13 is simulated as shown in Figure 2.14. It can be seen that the controller tracks the reference aggressively responding to a step response with a 90% rise time of 0.11 sec, 10% settling time of 0.25 sec, and an overshoot of 59%. The error at steady-state appears to show the same variance as the modeled noise. This indicates the controller would have zero steady-state error in the absence of noise.

The solver Simulink uses has a large impact on the simulated results. By default Simulink uses *ode3 Bogacki-Shampine* to simulate a fixed-step model. However, when real-time code is generated for the xPC Target environment, it behaves as if it is using the Fixed-Step *Discrete* solver. The contrast between the solvers is the most significant when examining overshoot. The overshoot using the *Discrete* solver was 59% but the overshoot using the *ode3 Bogacki-Shampine* solver was 32%. To use the *Discrete* solver, the PID block's library link is disabled and the continuous integrator and derivative blocks are replaced with discrete time integrator and derivative blocks.

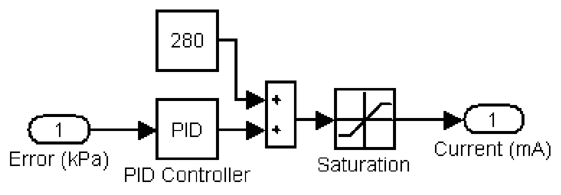

Figure 2.12. PID Controller With Deadzone Inverse.

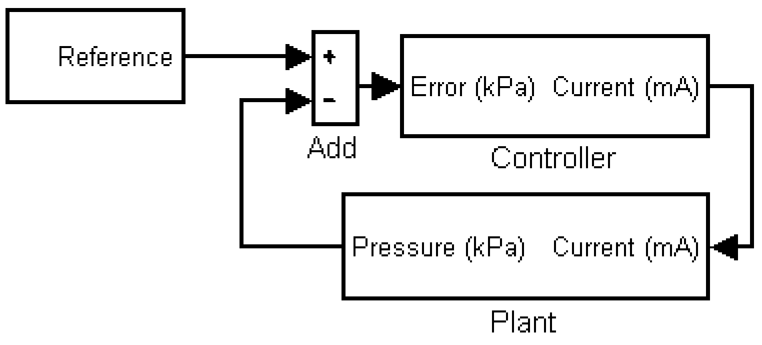

Figure 2.13. Closed-Loop System Model.

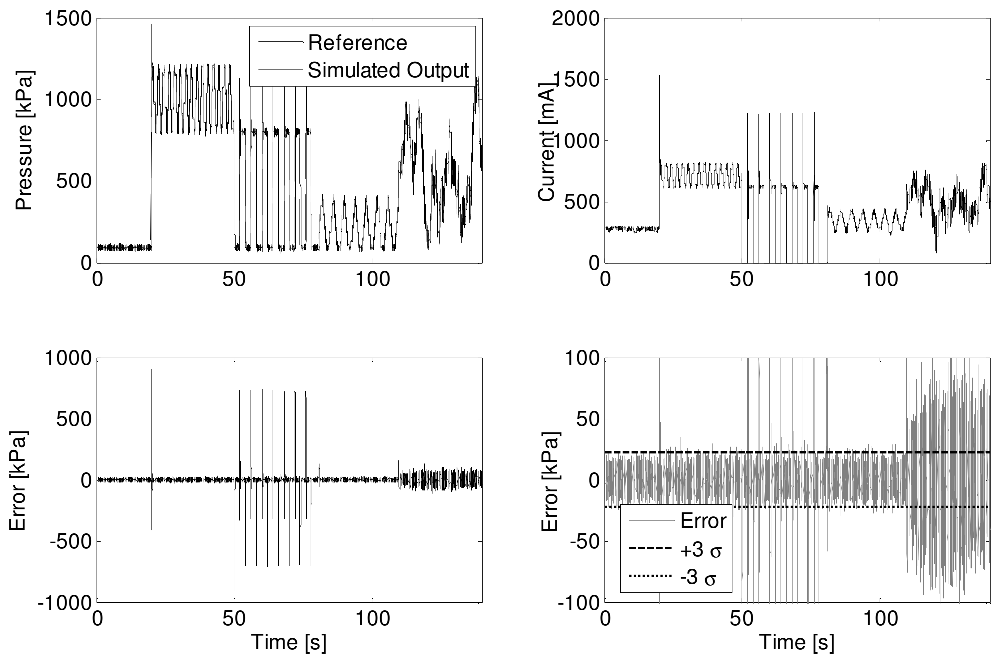

Figure 2.14. Simulated Results.

2.7.3. Emulation. The model predicts the hydraulic system's response but does not indicate anything regarding to the electrical hardware required to implement this controller such as the processor time the controller requires or the load on the CAN bus. One way to further delve into evaluating the electrical hardware requirements is to perform an emulation of the controller and plant models using two target PCs as shown in Figure 2.15. A cable is used to connect the two target PCs with D-sub DB-9 connectors with 120 ohm resistors at each end between CAN high and CAN low. The first target PC emulates the Master ECU and will be referred to as the *Master Emulator*. The *Master Emulator* receives pressure readings via CAN, calculates a current output using the controller from Figure 2.12, and sends the current command to the *Slave and Plant*

Emulator via CAN. The Simulink model in Figure 2.7 used to collect open loop data is modified to include the controller shown in Figure 2.12, creating a closed-loop feedback controller of the pilot pressure shown in Figure 2.16. The model is compiled, generating real-time executable code, which is uploaded to the *Master Emulator*.

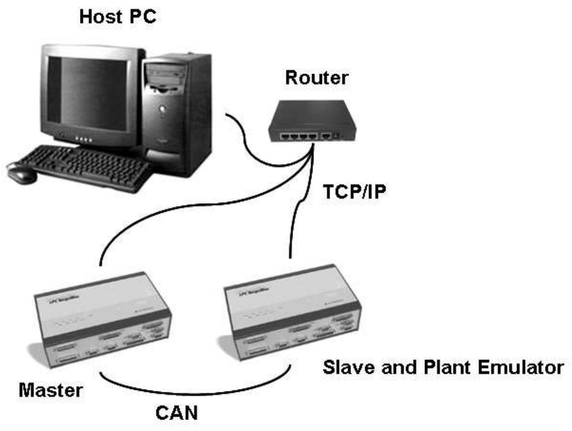

Figure 2.15. Two Target PC Configuration Used for Emulation.

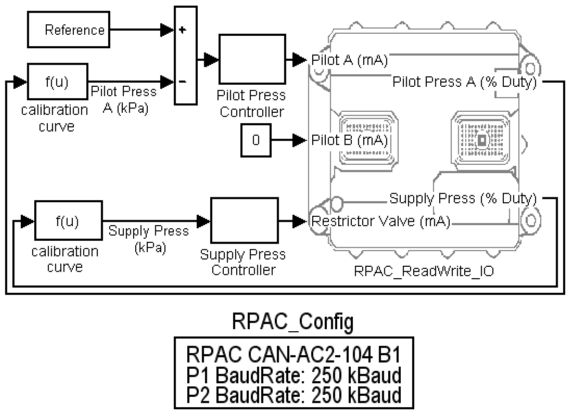

Figure 2.16. Simulink Model Used for Implementing Controller.

The second target PC is used to emulate the slave ECU and the pilot valve pressure dynamics. Since the slave ECU receives current commands and outputs pressure sensor readings, this target PC also contains the plant model and thus will be referred to as the *Slave and Plant Emulator*. Creating the *Slave and Plant Emulator* is more complicated than creating the *Master Emulator* model. With the *Master Emulator* model, the RPAC block contains and automatically sets up the CAN send and receive blocks. With the *Slave and Plant Emulator* model each CAN message has to be configured manually. The *Slave and Plant Emulator* needs to mimic all of the signals the slave ECU would send and receive. Since the *Slave and Plant Emulator* target PC and the *Master Emulator* target PC are the only two nodes on the CAN, every message that is sent by one is received by the other. Therefore the send and receive messages for the *Master Emulator* are interchanged to create the send and receive messages for the *Slave and Plant Emulator* as shown in Figure 2.17 and Figure 2.18 of their respective subsystems. To send out the pilot pressure message, the pressure is received from the plant model in units of kPa, converted to percent duty cycle, converted to an unsigned 16-bit integer by multiplying by a gain of $(2^{16}-1)/100$ and packed into the message using the bit-packing block as shown in Figure 2.17. The message containing the current command is received, unpacked by the bit-unpacking block, and converted from bits to milliamps. The two signals that were terminated Figure 2.18 are unneeded because one of them was being used for controlling the supply pressure (labeled “Restrictor Valve” in Figure 2.16), which is not a part of the pilot pressure model, and the other was unused (labeled “Pilot B” in Figure 2.16). Likewise the pressure signal used for supply pressure feedback in Figure 2.17 is not modeled so a zero is sent in its place.

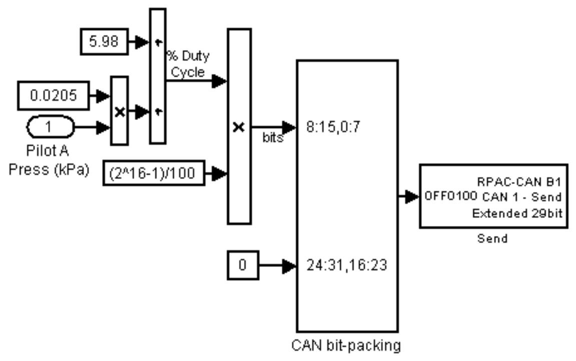

Figure 2.17. ***Slave and Plant Emulator's*** **Recreation of the Messages Sent by the Slave ECU to the Master ECU**

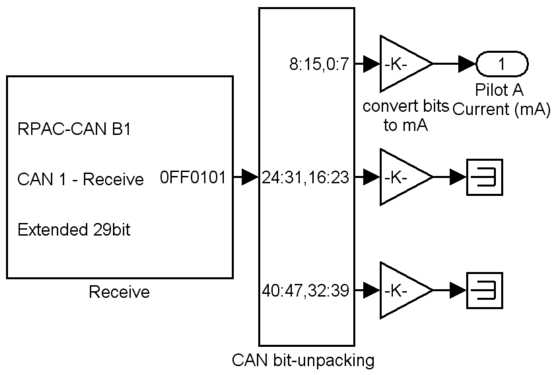

Figure 2.18. ***Slave and Plant Emulator's*** **Recreation of the Messages Received by the Slave ECU from the Master ECU**

The RPAC block, being a development tool, allows the user to specify the number of inputs and outputs to the ECU as well as several configuration parameters such as ECU loop time, which CAN board to use, and signal types and properties. Due to this flexibility, the CAN message identifiers and the order in which the data is packed into the messages is subject to change depending on how the ECU is configured. At initialization the target PC running the model with the RPAC block sends the CRO (Command Request Object) message and the slave ECU responds with the DTO (Data Transmit

Object) message communicating to the slave ECU how it should be configured. How the DTO message (which is sent by the slave ECU) is created and exactly what information it contains is proprietary. Since it is unknown how to recreate the DTO message, it is recorded using CANalyzer from Vector-Cantech, Inc. The *Slave and Plant Emulator* merely plays back the prerecorded DTO message at the first detection of the CRO message. This is a robust method so long as the RPAC block remains unchanged. Figure 2.19 shows the *Slave and Plant Emulator* Simulink model created from the send and receive subsystems from Figure 2.17 and Figure 2.18 and the plant model from Figure 2.11.

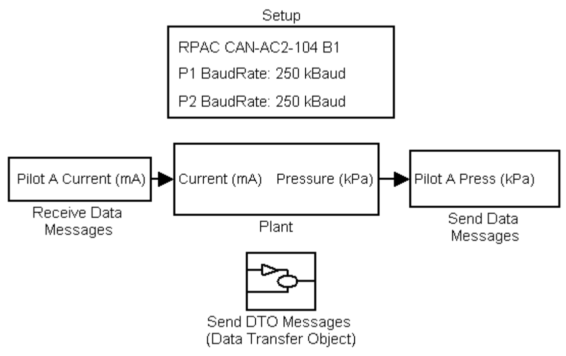

Figure 2.19. ***Slave and Plant Emulator*** **Simulink Model.**

The simulated and emulated pressures shown in Figure 2.20 match identically with the exception of the random noise. Converting the continuous transfer function and the continuous integration block in the PID controller to a discrete transfer function and a discrete integration block respectively allow use of the discrete solver. The simulated results using the discrete solver exactly match the emulated results. These results indicate that during the process of compiling the model into real-time executable code, the continuous blocks are converted to equivalent discrete blocks.

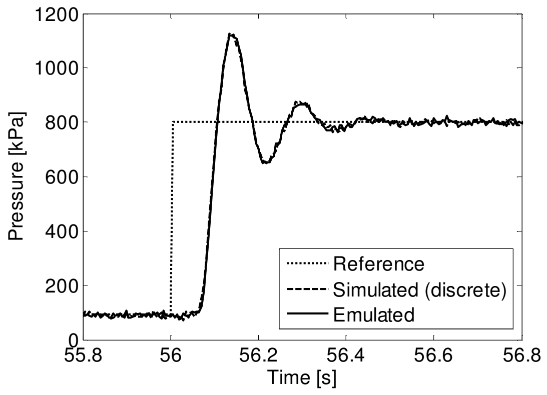

Figure 2.20. Emulated Pressure Compared to Simulated with the Discrete Fixed-Step Solvers.

One of the primary reasons for the two target PC emulation is to examine the execution time of the *Master Emulator*. In the Configuration Parameters dialog box there is an option of logging the task execution time. The task execution time is fairly constant with some variance. The variance is due to variations in the cache, memory access, interrupt latency, and multirate model execution (most blocks in Simulink can be configured to run at any multiple of the model's step-time; models used in this paper had a single rate execution equal to the mode's step-time). [28] However there is a slight jump in the task execution time after between 0.3 and 0.4 sec as indicated with the dotted line in Figure 2.21. This is most likely due to the DTO and CRO messages which are transmitted for 0.355 seconds at the beginning of execution before the RPAC block will allow messages to be sent from the *Master Emulator*. For all times after one second, the target execution time has a mean of $3.01 \cdot 10^{-4}$ sec, maximum of $3.29 \cdot 10^{-4}$ sec, minimum of $2.74 \cdot 10^{-4}$ sec, and standard deviation of $1.11 \cdot 10^{-5}$ sec. It is important that the target execution time be lower than the model step time, which is 5 milliseconds. This leaves the processor roughly 4.7 ms to perform background processes such as TCP/IP communication with the host, updating graphic windows, and calculating the task execution time. Choppy graphics windows or tuned parameters that are slow to update indicate the task execution time is approaching the step time. [28]

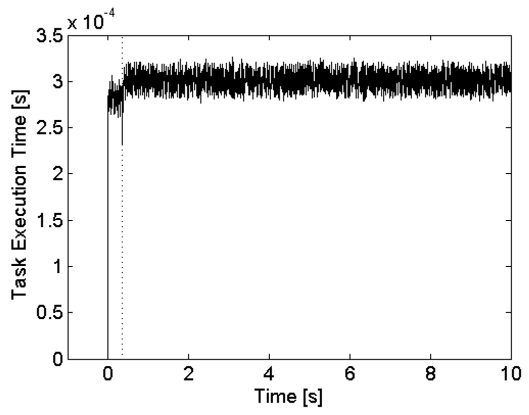

Figure 2.21. ***Master Emulator*** **Task Execution Time during Emulation.**

2.7.4. Control Implementation. After emulation, controlling the real system requires no new Simulink models. The *Master Emulator* controller shown in Figure 2.16 can be used with the real hydraulic system by merely unplugging the CAN line from the *Slave and Plant Emulator* target PC and plugging it into the slave ECU as shown in Figure 2.4. As can be seen from Figure 2.22, the controller tracks the reference closely; however, the overshoot is 181%, which is approximately three times higher than expected from simulation. The 90% rise time decreased slightly from 0.11 sec in simulation to 0.10 sec in the experiment. The 10% settling time decreased significantly from 0.25 sec in simulation to 0.15 sec in the experiment. The differences are most likely due to unmodeled higher order effects not captured in the first order plant model from Figure 2.11. It was observed in Section 2.7.1 that the system seemed to have varying time constants depending on pressure magnitude. It was expected that the model would capture the dynamics of the system better under some circumstances than others.

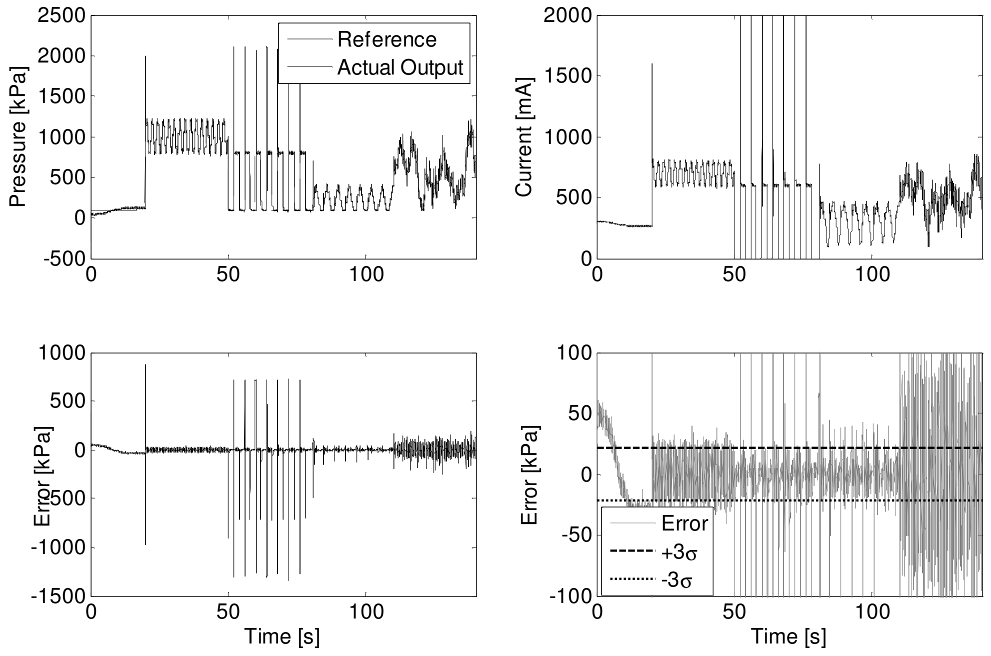

Figure 2.22. Experimental Pressure Reference Tracking.

Since the *Master Emulator* target PC was unchanged from emulation, it is expected that the task execution time will remain the same. Figure 2.23 confirms that the task execution times are indeed very close. For all times after one second, the target execution time has a mean of $2.98 \cdot 10^{-4}$ sec, maximum of $3.26 \cdot 10^{-4}$ sec, minimum of

$2.72 \cdot 10^{-4}$ sec, and standard deviation of $1.12 \cdot 10^{-5}$ sec. The jump in execution time is again present at 0.355 sec. Because the execution time remains the same, this reiterates that the *Master Emulator* target PC does not decipher the difference between the Slave ECU connected to the hydraulic system and a target PC emulating the Slave ECU and hydraulic system. Assuming the performance of the controller is acceptable, the development of the control algorithm is finished and it could be assigned to a programmer to embed on the Master ECU.

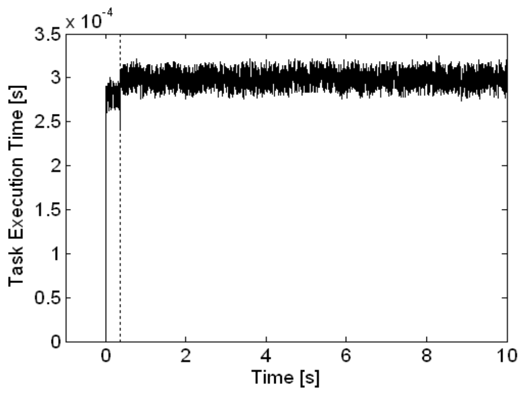

Figure 2.23. Master Task Execution Time during Control of Actual System.

2.8. SUMMARY AND CONCLUSIONS.

As control systems become more complex, more ECUs are being used to accommodate more special features, to reduce wiring, and to modularize systems. The xPC Target environment provides a platform for creating real-time executable code from a simulation environment. Target PCs running the xPC Target environment can be used to emulate a controller, a physical system, or even a combination of the two. This paper demonstrated the use of the xPC Target environment to emulate the ECUs in a Master/Slave architecture communicating over CAN. An emulation of the Master ECU was created on one target PC and an emulation of the Slave ECU and the hydraulic system with which it interfaced was created on a second target PC. Then the emulation of the Master ECU was implemented with the actual Slave ECU and hydraulic system.

The steps for validating a Master/Slave control algorithm have been demonstrated. First the hydraulic system was modeled by recording open loop pressure data using a target PC in conjunction with the Caterpillar RPAC library to execute a master control algorithm to send current commands and receive pressure commands. A plant model was created from the data and a control algorithm was designed and then validated in Simulink. The control algorithm was integrated with the RPAC interface for the slave ECU to create the *Master Emulator* target PC. The *Slave and Plant Emulator* Simulink model was created from scratch by replicating the CAN messages sent by the slave ECU and employing the plant model to determine the pressure output. The simulated results matched the emulated results so long as Simulink's fixed-step discrete solver was used. The *Master Emulator* target PC was connected to the slave ECU and the hydraulic system was successfully controlled. The task execution time of the *Master Emulator* target PC remained the same, indicating the *Master Emulator* target PC perceives the emulated system in the same way as the real system.

While the simulated and emulated results were identical, they did not perfectly predict the performance of the controller when implemented on the actual hydraulic system. The actual pressure output overshoot was much higher than in emulation. However, the hydraulic system's dynamics seemed to depend largely on the pilot pressure, most likely as a result of the decrease in oil compressibility with increased pressures. Since hydraulic modeling was not the focus of this paper, a single time constant was used to capture the dynamics of the system with the expectation that the accuracy of the model's prediction would vary depending on pressure. Nonetheless, this demonstrates the ability of target PCs running the xPC Target environment to emulate ECUs and the physical systems they control.

2.9. BIBLIOGRAPHY

[1] Teng, F.C., "Real-time control using Matlab Simulink," *Systems, Man, and Cybernetics, 2000 IEEE International Conference on* , vol.4, no., pp.2697-2702 vol.4, 2000.

[2] Luce, B.; Rahnamai, K., "Controller design for a multi-fan hovering system," *Electrical Insulation Conference and Electrical Manufacturing & Coil Winding Conference, 2001. Proceedings* , vol., no., pp.311-317, 2001.

[3] Shiakolas, P.S.; Piyabongkarn, D., "On the development of a real-time digital control system using xPC-Target and a magnetic levitation device," *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on* , vol.2, no., pp.1348-1353 vol.2, 2001.

[4] Nilsson, H.; “Rapid Prototyping with Matlab/Simulink –A Case Study”; Masters Thesis, Department of Automatic Control, Lund Institute of Technology, Sweden, 2002.

[5] Leonessa, A.; Mandello, J.; Morel, Y.; Vidal, M., "Design of a small, multi-purpose, autonomous surface vessel," *OCEANS 2003. Proceedings* , vol.1, no., pp.544-550 Vol.1, 2003.

[6] Gani, A.; Salami, M.J.E.; Khan, R., "Active vibration control of a beam with piezoelectric patches: real-time implementation with xPC Target," *Control Applications, 2003. CCA 2003. Proceedings of 2003 IEEE Conference on* , vol.1, no., pp. 538-544 vol.1, 23-25 June 2003.

[7] Zhang Shangying; Han Junwei; Zhao Hui, "RCP and RT control of 6-DOF parallel robot," *Robot Motion and Control, 2004. RoMoCo'04. Proceedings of the Fourth International Workshop on* , vol., no., pp. 133-137, 17-20 June 2004.

[8] Low, K.H.; Heng Wang; Wang, M.Y., "On the development of a real time control system by using xPC Target: solution to robotic system control," *Automation Science and Engineering, 2005. IEEE International Conference on* , vol., no., pp. 345-350, 1-2 Aug. 2005.

[9] Yejun Wei; Bajaj, P.; Scheidt, R.; Patton, J., "Visual error augmentation for enhancing motor learning and rehabilitative relearning," *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on* , vol., no., pp. 505-510, 28 June-1 July 2005.

[10] Bruno De Kelper; Blanchette, H.F.; Dessaint, L.-A., "Switching time model updating for the real-time Simulation of power-electronic circuits and motor drives," *Energy Conversion, IEEE Transaction on* , vol.20, no.1, pp. 181-186, March 2005.

[11] Hassan-Zadeh, I.; Janabi-Sharifi, F.; Yang, A.X., "Internet-based teleoperation of a mobile robot using shared impedance control scheme: a pilot study," *Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on* , vol., no., pp.346-351, 28-31 Aug. 2005.

[12] Mazon, A. J.; Zamora, I.; Valverde, V.; Eguia, P., "Rapid prototyping of artificial neural networks," *Power Tech, 2005 IEEE Russia* , vol., no., pp.1-5, 27-30 June 2005.

[13] Driscoll, S.; "The Design and Qualification of a Hydraulic Hardware-in-the-Loop Simulator"; Masters Thesis, School of Mechanical Engineering, Georgia Institute of Technology, 2005.

[14] Ferreira, P.A.F.; Pinto, J.R.C., "Visual Based Predictive Control for a Six Degrees of Freedom Robot," *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on* , vol., no., pp.846-853, 20-22 Sept. 2006.

[15] Zhu, Guchuan, "A Software Component for Network Based Data Acquisition and Control Applications," *Computer Architecture for Machine Perception and Sensing, 2006. CAMP 2006. International Workshop on* , vol., no., pp.34-37, 18-20 Aug. 2006.

[16] Rahnamai, K.; Cox, B.; Gorman, K., "Fuzzy Automatic Guitar Tuner," *North American Fuzzy Information Processing Society, 2007. NAFIPS '07. Annual Meeting of the* , vol., no., pp.195-199, 24-27 June 2007.

[17] Yunpeng Cao; Wanqing Teng; Huijie Zhang, "Dynamic Modeling and Hardware-in-the-Loop Simulation Testing for LPG Engine," *Mechatronics and Automation, 2007. ICMA 2007. International Conference on* , vol., no., pp.2093-2098, 5-8 Aug. 2007.

[18] Anderson, P.; Stone, H., "Predictive Guidance and Control for a Tail-Sitting Unmanned Aerial Vehicle," *Information, Decision and Control, 2007. IDC '07* , vol., no., pp.148-153, 12-14 Feb. 2007.

[19] Fan Luqiao,; Yao Xifan,; Qi Hengnian,; Jiang Liangzhong,; Wang Wei,, "An automatic control system for eod robot based on binocular vision position," *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on* , vol., no., pp.914-919, 15-18 Dec. 2007.

[20] P. Quinones-Reyes; H. Benitez-Perez; F. Cardenas-Flores; F. Garcia-Nocetti, “Reconfigurable Fuzzy Takagi Sugeno Networked Control using EDF Scheduling in xPC Target,” *Electronics, Robotics and Automotive Mechanics Conference, 2006* , vol.1, no., pp.105-110, Sept. 2006.

[21] Ramirez-Gonzalez, T.; Quinones-Reyes, P.; Benitez-Perez, H.; Laureano-Cruces, A.; Garcia-Nocetti, F., "Reconfigurable Fuzzy Takagi Sugeno Networked Control using Cooperative Agents and Local Fault Diagnosis," *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on* , vol., no., pp.1-5, 3-5 Oct. 2007.

[22] McGowan, D.J., Morrow, D.J., and Fox, B., 2006, “Integrated Governor Control for a Diesel-Generating Set,” *IEEE Transaction on Energy Conversion*, Vol. 21, No. 2, pp. 476-483.

[23] Shao Limin; An Shijie; Chang Hanbao; Gao Hongbin, "Study on Real-Time Test-Bench of Speed Governor Using Matlab/xPC Target," *Electronic Measurement and Instruments, 2007. ICEMI '07. 8th International Conference on* , vol., no., pp.3-27-3-31, Aug. 16 2007-July 18 2007.

[24] Rajnak, A. and Kumar, A., 2007, “Computer-aided Architecture Design and Optimized Implementation of Distributed Automotive EE Systems,” *44th ACM/IEEE Design Automation Conference*, Dates, Location, pp. 556-561.

[25] Navet, N., Song, Y., Simonot-Lion, F., and Wilwert, C., 2005, “Trends in Automotive Communication Systems,” *Proceedings of the IEEE* , Vol. 93, No. 6, pp. 1204-1223.

[26] http://www.semiconductors.bosch.de/pdf/can2spec.pdf. Bosch CAN Specification. June 2008.

[27] Gang, T. and Kokotovic, P.V., 1994, ”Adaptive Control of Plants with Unknown Dead-Zones,” *IEEE Transactions on Automatic Control*, Vol. 39, No. 1, pp.59-68.

[28] http://www.themathworks.com. The Mathworks, Inc. June 2008.

VITA

Jeffrey James Lentz was born on February 6, 1981, son of Rodney and Nancy Lentz. He graduated from Rolla High School in 1999 and completed a Bachelor of Science in Mechanical Engineering from University of Missouri, Columbia in December of 2003. He completed his Master of Science in Mechanical Engineering at Missouri University of Science and Technology over the summer of 2008. Jeff has worked at Caterpillar, Inc. since August 2006 in the CAT Electronics and Connected Worksite Division as an Implement Subsystem Engineer.