A Kalman Filter Calibration Method
For Analog Quadrature Position Encoders

by

Steven C. Venema

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in Electrical Engineering

University of Washington

1994

Approved by_____
(Chairperson of Supervisory Committee)

_____

_____

Program Authorized
to Offer Degree _____

Date _____

Signature _____

Date _____

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to express my appreciation and gratitude to my advisor and committee chair, Professor Blake Hannaford, for his patient and thorough technical and editing advice during the preparation of this thesis. Thanks also to Professors Deirdre Meldrum and Juris Vagners for taking the time to serve on my committee and for their many helpful suggestions. I also want to thank Frederik Boe, Dal-Yeon Hwang, and Darwei Kung for their willingness to act as "sounding boards" for many of the ideas put forth in this thesis. Finally, and most importantly, I want to thank my wife Meg and daughters Katie and Liz for their love, support, and tolerance during the several busy months that thesis was being written.

The development of this thesis was partially supported by the National Science Foundation through a PYI research grant awarded to Professor Blake Hannaford.

CHAPTER 1: INTRODUCTION

1.1 Motivation

The need for precision position sensors is driven by a variety of industries that require precision motion. Rotary position sensors are especially common since the majority of precision motion devices are driven by rotary motors. Typically, a rotary electromagnetic motor drives either a rotary or linear motion stage using some sort of gearing arrangement. If the gearing ratio between the motor and the motion stage is high, then a relatively low resolution rotary position sensor at the motor will result in high resolution motion measurements at the motion stage. However, backlash in the gearing system will limit precision.

New technological demands in diverse fields such as multi-chip semiconductor modules and biomedical genetics research have motivated the development of small, direct-drive motion systems with very high positioning resolution. The lack of gearing in these system implies that higher resolution position sensors are needed to achieve the higher positioning precision.

Sensors with analog quadrature outputs are of particular interest for high-precision positioning because their analog nature allows a degree of interpolation between the quadrature phases. However, this type of sensor suffers from nonlinear effects which are difficult to calibrate out. The objective of this thesis is the development of a new technique for the in-situ calibration of analog quadrature position sensors. The analog rotary incremental optical encoder is used as the example for this analysis; however the calibration technique is applicable to many other analog quadrature sensors.

1.2 Rotary Position Encoders

1.2.1 Functional Overview

The rotary incremental optical encoder consists of three basic components: a slotted disk, a light source, and a dual light-detector configured as shown in Figure 1.1. A light source shines on a disk which is covered with an evenly-spaced radial pattern of transmissive and reflective/absorptive elements called “encoder lines.” Lower resolution encoders use a metal disk with radially-cut slots to vary optical transmissivity of the disk. Higher res-

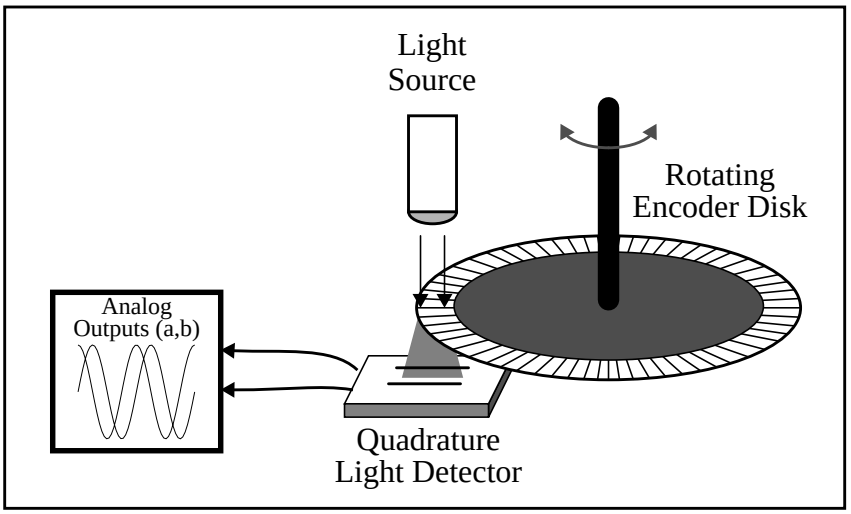

Figure 1.1: Encoder Schematic Diagram

olution encoders typically use a radial pattern of chromium lines evaporated onto clear glass. A dual light detector array on the opposite side of the disk detects varying intensities of light as the disk rotates due to the pattern on the disk. The two detectors are placed 0.25 of a line apart so that the peak output signal of one detector occurs 0.25 of a line before/ after the other (i.e., the outputs of the two detectors are in quadrature phase). This type of optical encoder was made feasible by the development of solid state light sources and photo detectors.

A typical encoder installation consists of mounting the light source and detector array on a fixed frame such as a motor body and mounting the encoder disk on the motor's rotor shaft. As the rotor turns, the encoder disk turns as well, alternately obstructing and transmitting light between the light source and the detectors. The detectors output two quadrature signals, called a and b, which are correlated with the intensity of the light striking each of the two detectors. These (a, b) signals can then be used together to determine the rotary position of the motor shaft as described below. Linear versions of this type of sensor are also possible, using a lines on a linear bar instead of a rotating disk.

1.2.2 Digital Encoders

Most incremental encoders in use today are digital encoders. In this type of encoder, the two detector outputs are converted into digital quadrature signals using thresholding cir-

cuits. These signals are then fed into a quadrature decoder logic circuit which monitors the two signals for changes in logic level. Due to the quadrature phase between the two signals, position changes of 0.25 of a line may be converted to pulses on separate clockwise (CW) and counterclockwise (CCW) signals. These position change signals are fed into “up” and “down” inputs of a digital counter which maintains an absolute position count in units of 0.25 of a line relative to some predefined 0 rotational position.

Logic chips such as Hewlett-Packard’s HCTL2016 [1] are available which contain both the quadrature decoder logic and the digital counter as well as a byte-wide output port suitable for direct interface to a microprocessor.

1.2.3 Analog Encoders

The primary difference between an analog incremental rotary encoder and its digital counterpart is that the detector outputs are not internally thresholded into digital quadrature signals. Instead, the analog quadrature signals are typically passed through operational amplifiers within the encoder which perform signal conditioning (offset and gain adjustments) and drive the output lines.

Ideally, the output signals from two detectors are sinusoidal waveforms in quadrature phase which go through one cycle per encoder-disk line as the disk rotates. Assuming that the signal is symmetric about 0-volts, the zero-crossings of the two signals can be used like the transitions in the digital quadrature signal to determine rotary position to 0.25-line resolution. However, the analog values of the two signals can be used to further interpolate the intra-line position to much higher resolutions.

This ability to interpolate between lines is the primary advantage of analog encoders: it *multiplies* the angular resolution afforded by the lines on the encoder disk by whatever amount of resolution that can be interpolated between a pair of lines. For example, if an encoder has $N_L = 1000$ lines then the digital resolution would be

$$R_D = \frac{360\left(\frac{\text{Degrees}}{\text{Revolution}}\right)}{1000\left(\frac{\text{Lines}}{\text{Revolution}}\right)\times 4\left(\frac{\text{Transitions}}{\text{Line}}\right)} = 0.09\left(\frac{\text{Degrees}}{\text{Transition}}\right) \qquad (1\text{-}1)$$

However, if the analog signals are used to interpolate the intra-line position to, for example,

one part in 100, then the angular resolution can be improved to

$$R_A = \frac{360\left(\frac{\text{Degrees}}{\text{Revolution}}\right)}{1000\left(\frac{\text{Lines}}{\text{Revolution}}\right)\times 100\left(\frac{\text{Transitions}}{\text{Line}}\right)} = 0.0036\left(\frac{\text{Degrees}}{\text{Transition}}\right), \quad (1\text{-}2)$$

a factor of 25 improvement in angular resolution.

1.2.4 Nominal Configuration

The (a, b) outputs of an analog encoder are periodic in angular distance with a frequency of N_L, the number of lines on the encoder disk. In order to obtain a position measurement, some standard hardware is needed to process these signals. Figure 1.2 shows

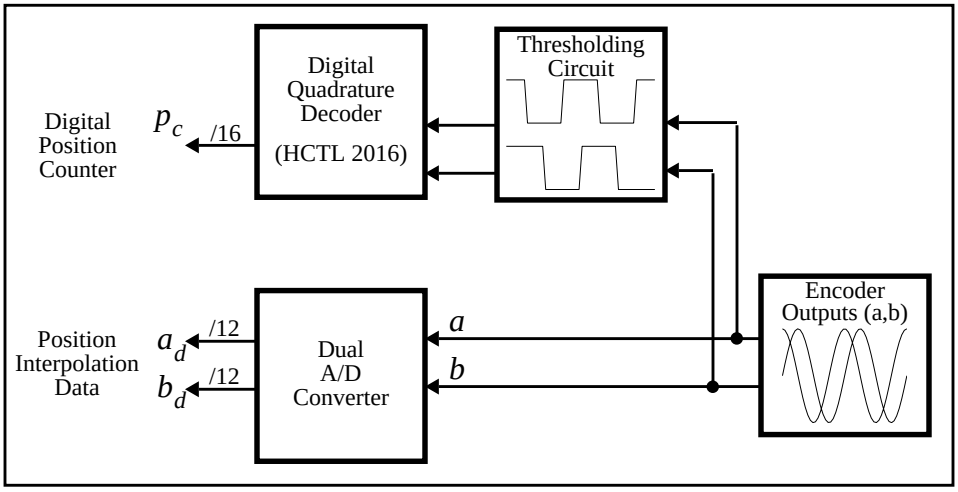

Figure 1.2: Nominal Analog Encoder Signal Processing

a typical configuration. Each of the encoder outputs (a, b) is passed through a thresholding circuit which emulates the thresholding circuits found in digital encoders. The thresholded versions of these signals are then treated just like the signals from a digital encoder as discussed in Section 1.2.2 above, connecting to a quadrature decoder which, in turn, drives a digital counter. The counter value is used as an estimate of the encoder position, p. The least significant bit of the counter represents 0.25 of an encoder line so the digital position estimate, p_d (in units of encoder lines) is the digital value of the counter, p_c divided by four:

$$p_d = \frac{p_c}{4} \tag{1-3}$$

The (a, b) signals from the encoder are also passed directly to a digitizing circuit which converts their analog voltages to digital equivalents. The digitized values are then used to generate a more accurate intra-line interpolation of the encoder position. Special care must be taken so that the (a, b) signals are sampled simultaneously to avoid changes in apparent position due to a time lag between samples.

1.3 Analog Encoder Calibration

1.3.1 Non-ideal Encoder Outputs

Most manufacturers of analog encoders claim that their devices output sinusoidal waveforms. Ideally, the two sinusoids would have exactly the same frequency (2π) and amplitude (α) and differ in phase by exactly 90°. These ideal signals could then be modeled by,

$$\begin{aligned} a(\tau) &= \alpha \sin(2\pi\tau) \\ b(\tau) &= \alpha \cos(2\pi\tau) \end{aligned} \tag{1-4}$$

where τ is the distance between an adjacent pair of encoder lines, normalized to the real interval [-0.5,+0.5). Assuming that the output signals accurately follow this model, the intra-line distance can be determined using the two-argument arctangent function to avoid quadrant ambiguity problems:

$$\tau = \frac{\operatorname{atan2}(a, b)}{2\pi}. \tag{1-5}$$

In reality, the output signals of the analog encoders, $a(\tau)$ and $b(\tau)$ rarely match the ideal quadrature sinusoid model of (1-4). The output waveform is very sensitive to the relative distance and alignment between the light-source and the encoder-disk and between the encoder-disk and the quadrature photodetectors. Most high-resolution analog encoders are sold as pre-assembled units with high-precision bearings to minimize these alignment distortions. “Kit” encoders are assembled by the customer and are very difficult to align correctly. Even in the pre-assembled units, the encoder signals rarely match the ideal sinusoidal model given in (1-4).

The outputs of 3 different encoders are shown as Lissajous plots (a vs. b) in Figure 1.3. The left-most figure is from a 1024-line factory-assembled encoder with high-preci-

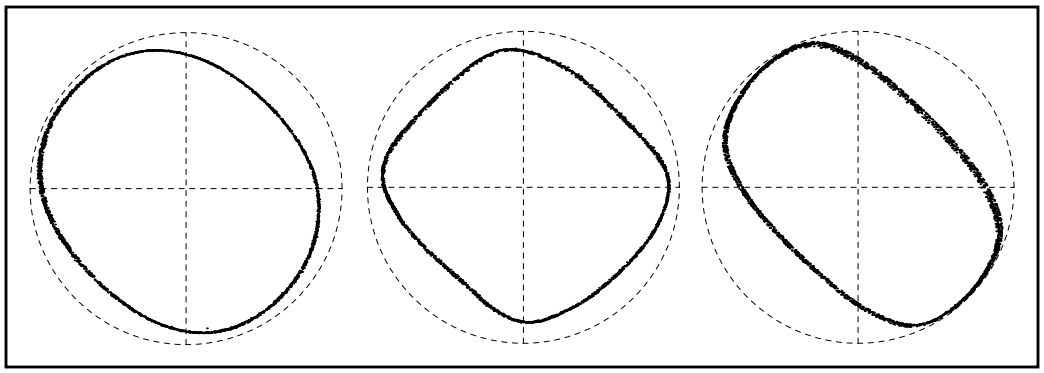

Figure 1.3: Lissajous plots for typical encoder outputs

sion bearings. The remaining two figures are 1000-line hollow-shaft "kit" encoders mounted with the light emitter and detectors bolted to the motor body and the encoder disk mounted on the motor shaft. The dotted-circular shape is the output from an ideal encoder characterized by (1-4). These plots demonstrate that the kit encoders output waveforms are further from the ideal circle than the factory-assembled encoder, but all three encoders suffer from non-ideal characteristics.

1.3.2 One-to-Many Mapping Problem

At first glance it seems that one could use the Lissajous plot directly to determine the intra-line position. However, without an accurate model of the encoder signals, $a(\tau)$ and $b(\tau)$, it is impossible to accurately map these signals to the intra-line position, τ. This is illustrated graphically in Figure 1.4 which shows that it is possible to invent two functions $[a_1(\tau), b_1(\tau)]$ and $[a_2(\tau), b_2(\tau)]$ which map to the same Lissajous plot. In fact, any number of functions $[a(\tau), b(\tau)]$ can map to the same Lissajous plot. This implies the converse as well: it is impossible to use the Lissajous plot alone to determine intra-line position, τ for a given encoder time sample, $[a(t), b(t)]$.

1.3.3 Goals

Experience has shown that analog encoder output signals roughly approximate the ideal

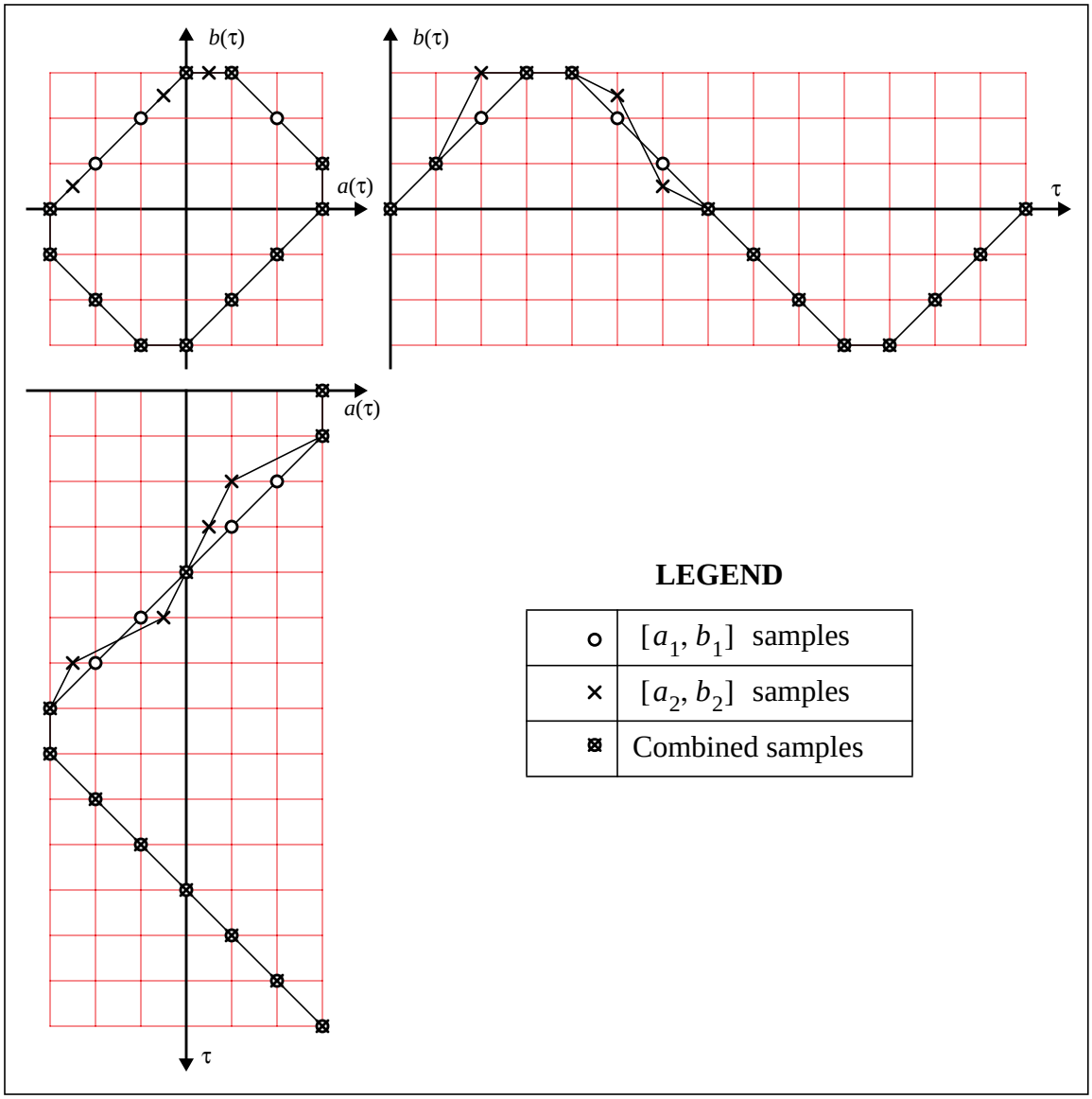

Figure 1.4: Encoder Output "One-to-Many" Mapping Example

model given in (1-4). If this ideal model is used to generate a rough intra-line position estimate, τ_a, using (1-5), the true intra-line position may be modeled as

$$\tau = \tau_a + \varepsilon(\tau_a) \tag{1-6}$$

where $\varepsilon(\tau_a)$ is a calibration term which corrects for the estimate error in τ_a due to the use of the ideal model.

The goal of this thesis is the development of a robust technique to accurately determine the calibration function $\varepsilon(\tau_a)$ for an analog encoder which

1. does not require expensive and time-consuming external calibration hardware such as micrometers or laser interferometers,
2. applies to any analog encoder (even the ones with highly non-circular Lissajous plots),
3. may be used on encoders in their final installed configuration (since the encoder output functions may change with each installation), and
4. uses the same signal measurement hardware and sampling rates as the final control system so that calibration can be performed in the final installed system at any time.

1.4 Review of Previous Work

1.4.1 Parametric Calibration Method

A parametric calibration algorithm for analog optical encoders was developed by P. Marbot in [2] for use in a 3 degree-of-freedom direct-drive minirobot. The algorithm assumes that the encoder outputs are sinusoidal as in (1-4), but relaxes the 90° phase and the equal-amplitude constraints. The outputs are modeled by the functions,

$$\begin{aligned} a(\tau) &= \alpha \cos(2\pi\tau) \\ b(\tau) &= \beta \sin(2\pi\tau - \Phi) \end{aligned} \tag{1-7}$$

where the amplitudes, α and β, and the quadrature phase error, Φ, are constant but unique to each encoder and mounting configuration. The Lissajous plot of the (a, b) outputs of this model forms an ellipse where the parameters (α, β) control the length of the major and minor axes and the parameter Φ controls the amount of rotation. Using this modified model, Marbot first rewrites the second equation in (1-7) as

$$b(\tau) = \beta \sin(2\pi\tau)\cos(\Phi) - \beta \sin(\Phi)\cos(2\pi\tau) \tag{1-8}$$

by trigonometric identity and then solves for τ:

$$\begin{aligned} \frac{b(\tau)}{a(\tau)} &= \frac{\beta \sin(2\pi\tau)\cos(\Phi) - \beta \sin(\Phi)\cos(2\pi\tau)}{\alpha \cos(2\pi\tau)} \\ &= \frac{\beta}{\alpha} \tan(2\pi\tau)\cos(\Phi) - \frac{\beta}{\alpha}\sin(\Phi) \end{aligned} . \tag{1-9}$$

Equation (1-9) can be rearranged algebraically to

$$\tan(2\pi\tau) = \frac{\alpha b(\tau)}{\beta a(\tau)\cos(\Phi)} + \tan(\Phi) \tag{1-10}$$

so that

$$\tau = \frac{\operatorname{atan2}(\alpha b(\tau) + \beta a(\tau)\sin\Phi, \beta a(\tau)\cos\Phi)}{2\pi}. \tag{1-11}$$

The new encoder model in (1-11) was applied to factory-assembled encoders with output waveforms similar to that shown in the left-most Lissajous plot in Figure 1.3. Marbot named this model the “Potato Algorithm” since it partially corrects for the potato-like shape of the Lissajous plot of his encoder outputs. Note that it is the unequal amplitudes $(\alpha \neq \beta)$ of the two signals which gives the plot its elliptical shape and the phase error $(\Phi \neq 0)$ that causes the rotated orientation of the ellipse. If the amplitudes are exactly equal and there is no phase error, then the plot would form a perfect circle. In this case, the ideal model in (1-5) would be applicable.

The Potato algorithm was implemented by lookup table on an 8-bit microcontroller (Motorola 68HC11) and increased the intra-line resolution by a factor of 16 over the standard 0.25-line resolution of a digital encoder while running at a rate of 1000 Hz. An off-line calibration procedure was used to find the model parameters, $[\alpha, \beta, \Phi]$. This calibration required multiple samples from a moving encoder to be uploaded from the microcontroller to a personal computer via a serial port where the amplitude and phase estimates were made by examining the maximum values and phase difference between the two signals for a single cycle.

Marbot points out the following important features of this “potato algorithm”:

1. The parameters $[\alpha, \beta, \Phi]$ are constant for a given encoder and mounting configuration. Once these parameters are identified, much of the computational complexity of (1-11) can be reduced by precomputing the constant terms such as $\sin\Phi$ and $\cos\Phi$. This was especially important in view of the relatively low computational performance of the selected microcontroller (0.5MIPS, 16-bit integer arithmetic) and the high servo rate (1000 Hz.). Precomputed lookup tables for some of the sub-expressions of (1-11) were also used to increase computational performance.
2. The model assumes that the encoder signals are symmetric about the origin (i.e.,

the potato shape in the Lissajous plot is centered at the origin). If either of the signals have some small amount of offset, this should be subtracted out before being passed to the model.

3. The parameters $[\alpha, \beta]$ are expressed in (1-10) as a ratio. Therefore, this algorithm is implicitly insensitive to common-mode changes in the peak-to-peak amplitude of the (a, b) encoder signals.

Using the Potato algorithm on factory assembled encoders, Marbot was able to increase resolution by a factor of 16 (4-bits) over the 0.25-line digital resolution. However, a better method of calibration is needed if higher precision interpolation is desired or if the output of an encoder deviates significantly from the ellipsoidal shape assumed by (1-11).

1.4.2 Constant-Velocity Calibration Method

Hagiwara, et.al., developed a lookup-table based calibration ("code compensation") technique in [3]. The paper discusses the design of a hardware encoder interface which performs the following functions:

1. Reads the analog (a, b) encoder signals at a 100-kHz rate, converting them to digital values using A/D converters.
2. Uses the digital (a_d, b_d) values as an index into a ROM-based lookup table which contains the arctangent function.
3. Uses the output of the arctangent table as an index into a ROM-based lookup table of correction factors.
4. Combines the outputs of the two ROM lookup tables into an estimated position output using a collection of adders, latches, and counters.

This digital interface hardware computes an interpolated intra-line position by first assuming an ideal encoder model (i.e., equation (1-5) using the first ROM), and then subtracting off the error between the ideal model and true position using the second ROM lookup table. Additional adders, latches, and counters are used to keep track of the incremental rotary position in a manner similar to that of the HCTL 2016 chip mentioned in Section 1.2.2.

The index (actually a digital address) for the first ROM, is formed by combining the two digitized encoder values using the relation,

$$A_{\text{ROM1}} = \left(2^m \times a_d \right) + b_d \tag{1-12}$$

where m is the number of significant bits from the A/D converter. This maps all possible A/D value-pairs (a_d, b_d) to unique addresses, implying that the ROM must have a depth of 2^{2m} words. The values at each address correspond to a conveniently scaled relation, $\text{atan2}\,(a_d, b_d)$, and the accuracy is limited to the ROM width. Hagiwara, et.al. used 6-bit A/D converters and an 8-bit ROM, requiring a 4096×8-bit ROM for the first lookup table. The 8-bit ROM width limited the intra-line interpolation precision to $1/2^8$ or 256 unique locations.

The contents of the second, "correction-factor" ROM were generated by high-speed sampling (100 kHz) of the encoder outputs while the encoder was spinning at about 9 lines per second to yield about 220 samples per line. By assuming that the rotational velocity was constant over the sampling ensemble, a "true" position was computed by integrating the average velocity (i.e., average-velocity $\times$ sample-interval $\times$ sample-number). The correction factor for a given ideal-model position (e.g., $\text{atan2}\,(a_d, b_d)$) was then the difference between the ideal-model position and the integrated position for a given sample. The error for each ideal-model position (256 possible positions due to the 8-bit width of the first ROM) was computed over multiple experiments and averaged to generate a final lookup table of correction factors.

Using this method on a 3600-line encoder, an 8-bit (256 element) correction table was generated. Experimental confirmation of the accuracy of this correction table indicated that there were residual errors of 3 counts (about 3 bits). This implies that only the upper 5-bits of the 8-bit correction table were significant, yielding an intra-line resolution of $2^5 = 32$, a factor of 8 increase in intra-line resolution over the 0.25-line resolution of a digital encoder. Regarding the 3-count error in the calibration table, Hagiwara concluded that,

> As every phase code is compensated, the deviation is expected to be less than ± 1 count. But in experiments, so far as this time, we have not been successful in minimizing the error to this level. This may be caused by noise and the fluctuation of the rotating speed which we assumed to be constant.

1.4.3 “Hummingbird” Minipositioner

Karidis, et.al. developed a specialized quadrature rotary position encoder as part of their “Hummingbird” Minipositioner [4], a small ($13 \times 13 \times 1$ mm work volume), high-performance (>50-G accelerations) manipulator for high-precision positioning (1-μm) applications. The encoder uses the same principles as a typical rotary optical encoder such as the one shown in Figure 1.1. However the optical path is slightly different with the light passing from the LED through a grid-plate (instead of a rotary disk), to a corner reflector which sends the light back through a second grid plate, to a pattern of 4 photodetectors which generate two differential quadrature signals as a function of position.

The encoder line resolution is 10 lines per degree, yielding a digital resolution of 0.025 degrees. Karidis, et.al. state that they are able to utilize intra-line interpolation to increase resolution to 0.00089 of a degree, a factor of 29 improvement. However no mention is made of the calibration method used for their encoder.

1.4.4 Sawyer Sensor

The Sawyer motor [5] is a planar motion analog to the rotary stepper motor. A polished, flat steel surface, called a “platen” acts as the stator. The surface is inscribed with a fine rectangular grid which acts as the stator’s “teeth”. The motor contains an arrangement of electromagnets and linearly arranged, quadrature-phased groups of magnetic “teeth” that allow it to move along and slightly above the plane of the steel platen by interacting with an air bearing. The magnitude and phasing of the current profiles in the electromagnets control motion along the two cartesian degrees of freedom as well as a small amount of orientation around the axis perpendicular to the platen. As with traditional rotary stepper motors, the Sawyer motor supports open-loop position-control as well as micro-stepping control modes. However, the finest stepping precisions are prone to errors (e.g., missed steps) so some sort of position sensing is desirable.

J. Ish-Shalom [6] developed a Sawyer sensor for use with the Sawyer motor. The sensor uses the same type of structure as the Sawyer motor: groups of magnetic teeth in quadrature phase (relative to the tooth pitch) act as a variable magnetic reluctance sensor, with a high-frequency magnetic field from a driver coil acting as the magnetic signal source. The sensor

outputs two signals in quadrature phase which are periodic in distance relative to the pitch of the platen teeth.

The Lissajous plot of the Sawyer sensor signals, as shown in [6], appears very similar to the middle plot of Figure 1.3. Ish-Shalom used a laser interferometer system to calibrate non-ideal properties of the output signals relative to an ideal sinusoidal function similar to (1-4). The error correction values, as defined by the difference between the interferometer-measured position and the ideal sinusoidal model position, were fitted to a 3rd order polynomial. This polynomial was then used to generate corrections from the ideal model for future position measurements.

Ish-Shalom tested the sensor on an experimental system with 1-mm pitch lines on the platen, implying a standard digital resolution of 250-μm. He reports maximum errors of less than ±1 -μm using the 3rd-order polynomial model for intra-line interpolation, a factor of 250 improvement over the digital position estimate.

1.5 Mini Direct-Drive Robot

Marbot [2] developed a miniature 3 degree-of-freedom (DOF) biomedical research robot for high-precision manipulation of small fluid samples using thin glass pipettes. Direct-drive motors and precision sensors were used on its three joints to maximize positioning control and resolution over its 17-cm^3 work volume. The first joint was a prismatic (linear motion) joint and used an LVDT positioning sensor. The other two joints were revolute and used analog quadrature encoders for position sensing. These factory assembled encoders (Model CP-320-1024 from Computer Optical Products, Inc. (COPI) of Chatsworth California) produced Lissajous plots similar to the one shown in the left-most plot of Figure 1.3. The “Potato” algorithm described in Section 1.4.1 was used to calibrate these analog encoders.

This minirobot has subsequently been upgraded to incorporate a 2-axis wrist, for a total of 5 joints to be used in more demanding biomedical research tasks as well as a general research tool for scaled teleoperation. Figure 1.5 shows the current physical implementation of the robot. A general overview of the design and implementation of the robot and its control system may be found in [7]. The mechanism design and implementation is documented

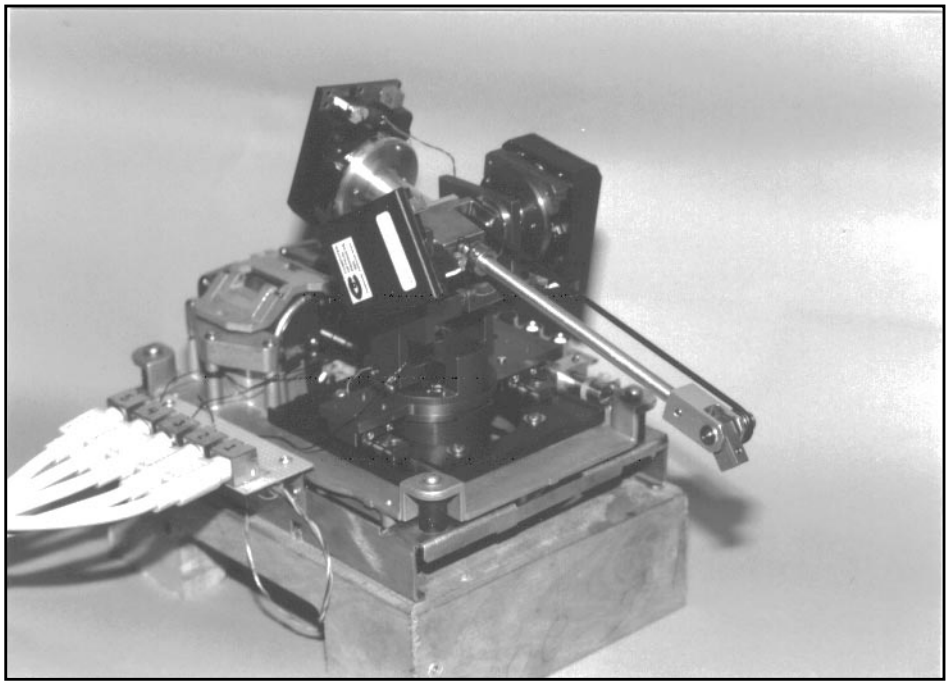

Figure 1.5: 5-Axis Mini Direct Drive Robot

in [8]. The control system processor is documented in [9].

The upgrade from 3 joints to 5 necessitated the use of smaller, more light-weight “kit” encoders (COPI Model CP-200-1000) which have less ideal outputs such as the ones shown in the central and right Lissajous plots in Figure 1.3. It was these less ideal encoders which motivated the calibration work in this thesis because the “potato” algorithm discussed in Section 1.4.1 was no longer a good fit.

CHAPTER 2: PROBLEM ANALYSIS

In Section 1.3, Lissajous plots for several analog encoders were shown which each have a different non-circular shape. The variations in output waveforms between different encoders motivates the development of a general calibration method to determine intra-line position. One way to view this calibration problem is to model the intra-line position as given in (1-6):

$$\tau = \tau_a + \varepsilon(\tau_a), \tag{2-1}$$

Here the true intra-line position, τ, is modeled as the sum of a rough intra-line position, τ_a, as determined by the ideal model in (1-4) and $\varepsilon(\tau_a)$, a calibration term which compensates for the non-ideal properties of the encoder. The challenge is to determine this calibration function, $\varepsilon(\tau_a)$, for $-0.5 \le \tau_a < 0.5$. A new approach to determining this function in the form of a lookup table is presented in the next chapter. However, before this approach is presented, it is important to understand the final limitations on any calibration approach.

2.1 Noise Model

The use of intra-line interpolation of the ideal analog encoder allows theoretically infinite angular position resolution. However, the limitations of manufacturing tolerance, measurement noise, quantization error, and calibration errors in real encoders provide physical limitations to position resolution. The purpose of this chapter is to analyze the sources of error in the process of intra-line position estimation for non-ideal encoders.

Figure (2-1) shows that the accuracy of the actual intra-line position, τ, is dependent on the accuracy of both the measured rough position estimate, τ_a, and the calibration term, $\varepsilon(\tau_a)$. Since τ_a is a computed function of the (a, b) encoder signal measurements which are themselves subject to measurement noise, τ_a is an inherently stochastic signal. The calibration function, $\varepsilon(\tau_a)$, is also stochastic because noise in τ_a will cause the incorrect calibration term, ε, to be extracted from the function. Also, $\varepsilon(\tau_a)$ may be imperfectly known due to errors in the calibration process or due to variations in the line width and intra-line distance on the encoder disk. The noise on the (a, b) encoder signals is *measurement noise*, and the error in the correction factor, $\varepsilon(\tau_a)$, is *calibration error*. Quantization noise from the A/D conversion is not specifically modeled; however, it can be

lumped together with the measurement noise for the purposes of the following analysis.

Using the above definitions, the effects of these errors on the final intra-line position estimate, τ, can be analyzed. Figure 2.1 shows the signal flow for the intra-line estimator

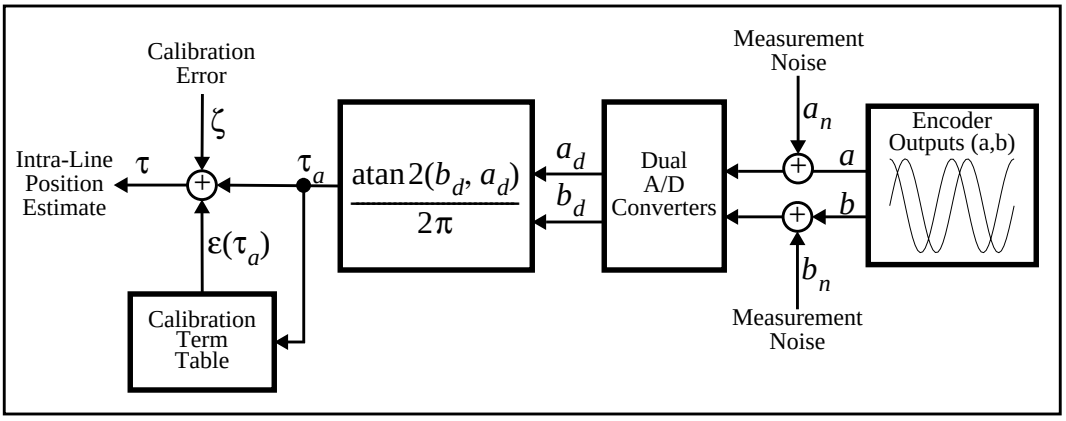

Figure 2.1: Encoder Signal Flow with Error Model

in the nominal configuration (see Section 1.2.4). The measurement noise is shown as additive noise in the encoder analog outputs which is then quantized by the A/D converters and used to compute τ_a using the ideal encoder model relation in (1-5). The calibration noise is shown as additive noise to the corrected intra-line position estimate, τ.

2.2 Measurement Noise

2.2.1 Noise Model

The analog encoder signals are generated by optical receivers within the encoder. These signals must pass through a variety of amplifiers and cables before finally reaching A/D converters. Noise may added at any stage of this signal path due to internal amplifier noise and electromagnetic interference (EMI) in cable transmission lines. The noisy signals are then sampled at periodic intervals by the A/D converters.

It is convenient to model the discrete measurements, (a_d, b_d), as the sum of deterministic signals, $(a(t), b(t))$, and some stationary noise sources with zero-mean gaussian distributions using the relations,

$$\begin{aligned} a_d(t) &= a(t) + a_n(t) \\ b_d(t) &= b(t) + b_n(t) \end{aligned} \qquad (2\text{-}2)$$

where

$$\begin{aligned} a_n &\triangleq N(0, A) \\ b_n &\triangleq N(0, B) \end{aligned} \tag{2-3}$$

and (A, B) are the variances of the two noise sources. Since a_d and b_d are each a sum of a deterministic variable and a gaussian random variable, and since a linear function of a gaussian random variable is also gaussian, a_d and b_d are also gaussian random variables. Thus (2-2) can be rewritten as

$$\begin{aligned} a_d(t) &= N(a(t), A) \\ b_d(t) &= N(b(t), B) \end{aligned}. \tag{2-4}$$

In order to justify this model, some experimental data is needed. By holding the encoder position constant, the statistical properties of the measurement noise on an experimental encoder may be studied. Approximately 3000 samples of the (a_d, b_d) signals for a stationary analog encoder were taken at 1-millisecond intervals. Figure 2.2 shows the histograms of each of the encoder signals with one bin per A/D unit. The plots show roughly gaussian distributions with two different mean values which are the stationary $a(t)$ and $b(t)$ deterministic positions in (2-3). Numeric computations of the variance in the two signals yield

$$\begin{aligned} A &= 1.68 \\ B &= 3.04 \end{aligned} \tag{2-5}$$

using A and B as defined in (2-3). The variances of the two signals are quite different; this was found to be the case for several different encoders at various intra-line positions using the same experimental configuration. The source of this difference was traced to the actual encoder outputs (i.e., it is not due to noise in the experimental electronics) but the cause of this difference is unknown.

Another important aspect of the measurement noise is the degree of correlation between the two noise sources as modeled in (2-2). Figure 2.3 shows the joint distribution of the experimental dataset as both a mesh plot and a contour plot. The fact that the major and minor axes of the constant probability ellipses in the contour plot align with the measurement axes

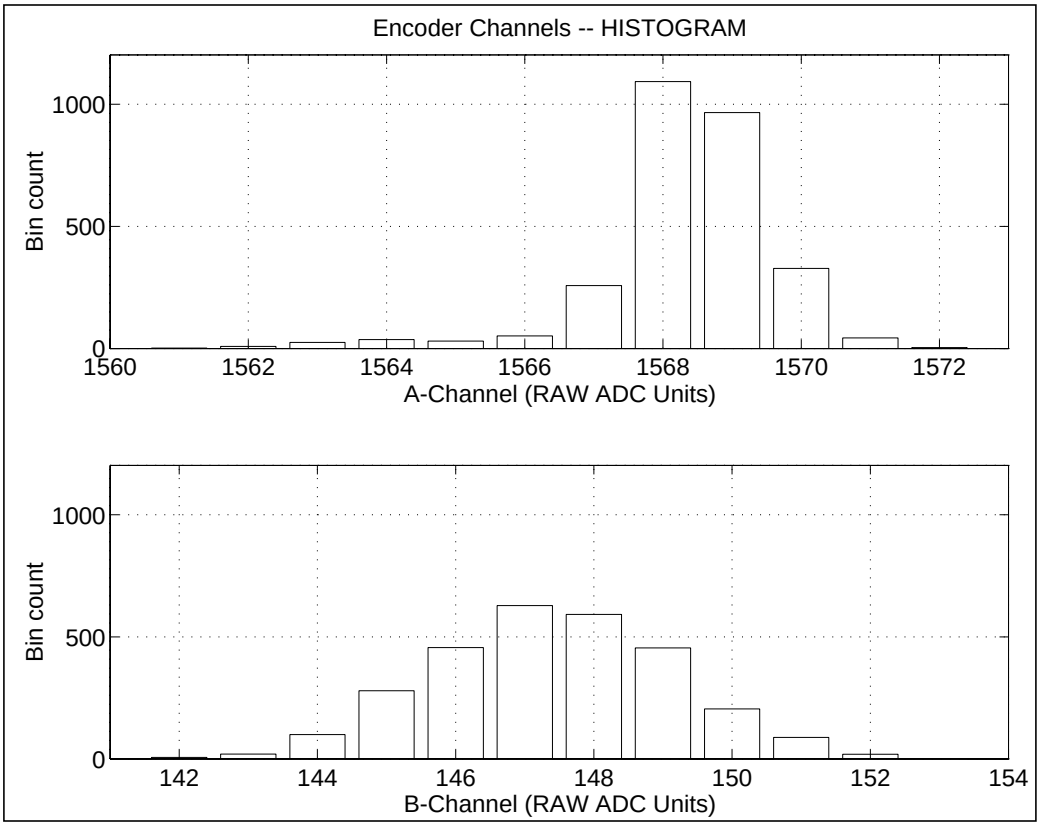

Figure 2.2: Histograms of Encoder Measurement Noise

indicates that the two random noise sources are highly uncorrelated. The computed correlation coefficient, $\rho = -0.167$, confirms this observation. The low correlation implies that the random variables used to model the two gaussian noise sources are independent. When (a_d, b_d) are combined together to compute τ_a, the independence of these random variables allows for an improved estimate of τ_a.

2.2.2 Arctangent of Noisy Measurements

The rough intra-line position, τ_a, as defined in (2-1), is computed using the ideal model:

$$\tau_a(t) = \frac{\operatorname{atan2}\,[a_d(t), b_d(t)]}{2\pi}, \tag{2-6}$$

Since τ_a is dependant upon a_d and b_d, the variance in $a_d(t)$ and $b_d(t)$ will be transferred in some form to $\tau_a(t)$. The arctangent function in (2-6) is a nonlinear mapping

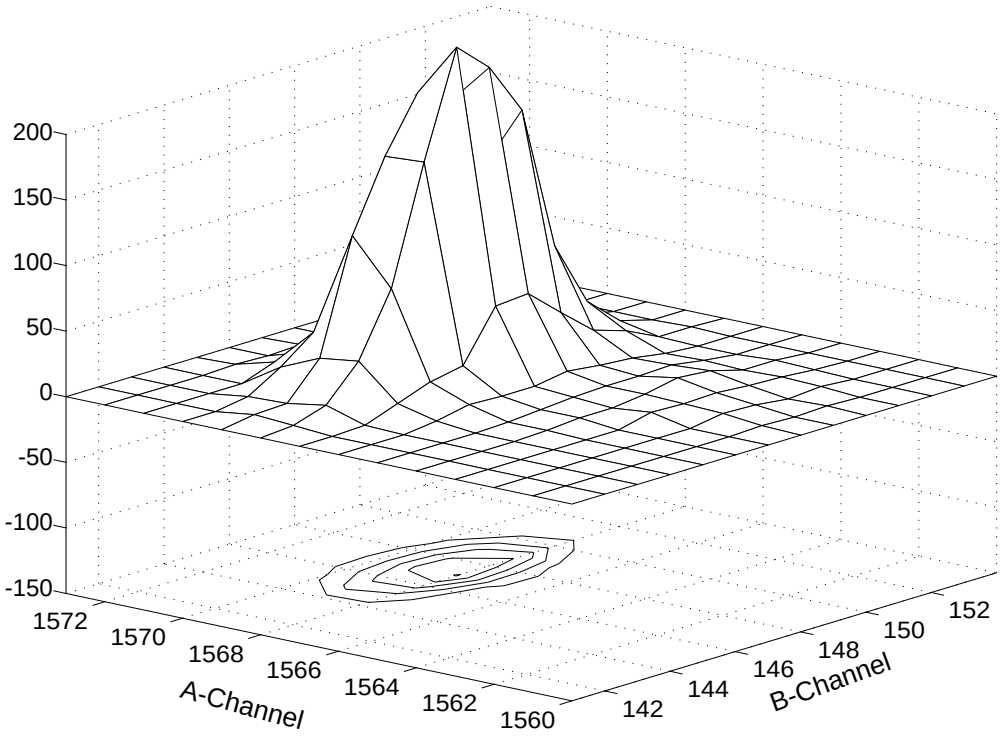

Figure 2.3: Joint Distribution of Encoder Measurement Noise

function which maps the random variables, (a_d, b_d), to a new random variable, τ_a. The variance of τ_a will be dependant on the input variances (A, B) but may also depend upon the mean (or "true") position, (τ, r), where r is the radius from the $(a, b) = (0, 0)$ origin. This variance mapping problem is illustrated in Figure 2.4, a first quadrant subsection

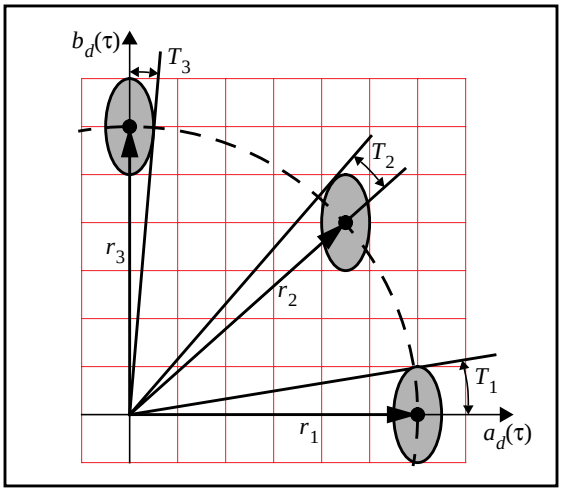

Figure 2.4: Variance Mapping Problem

of a Lissajous plot. A constant-probability contour for the jointly gaussian input signals, a_d and b_d, are shown for three different time samples, with the samples located at radii (r_1, r_2, r_3) and intra-line positions $(0, 0.125, 0.25)$ respectively. Note that the three radii shown in the figure are identical. The Lissajous plots in Figure 1.3 show that this is not generally the case. Figure 2.4 shows that while the variances, (A, B), and the radii of the input signals are constant for all three samples, the *apparent* variance in τ_a, called T, is different for each of the samples and depends upon both τ and the radius r. If the two input variances happen to be equal (i.e., $A = B$), then the probability contour will be symmetric about its mean value so that T would only depend on r. However, as was shown in the histograms of Figure 2.2, this is not generally the case.

The challenge then is to determine how the variance from the input signals is mapped to variance on the output signal which is the rough intra-line position estimate. It can be shown that a linear vector function of jointly gaussian random variables yields jointly gaussian random variables. However the arctangent function is not linear. A more general approach is to examine how the (nonlinear) arctangent function maps the joint probability density function (PDF) of its input random variables.

2.2.3 Variance of Rough Position Estimate

Leon-Garcia [10] showed that if $\mathbf{X} \in \Re^n$ is a random vector, the joint PDF of $\mathbf{Y} = f(\mathbf{X})$, where $\mathbf{Y} \in \Re^n$ is related to the joint PDF of $\mathbf{X}$ by:

$$p_Y(\mathbf{Y}) = p_X(g(\mathbf{Y})) \cdot |J(g(\mathbf{Y}))| \tag{2-7}$$

where p_X is the joint PDF over the input vector $\mathbf{X}$, $g(\mathbf{Y})$ is the inverse of $f(\mathbf{X})$ [i.e., $\mathbf{X} = g(\mathbf{Y})$], and $|J(g(\mathbf{Y}))|$ is the determinant of the Jacobian of $g(\mathbf{Y})$, $\partial g(\mathbf{Y}) / \partial \mathbf{Y}$. This approach assumes that the function $f(\mathbf{X})$ is invertible.

Since the arctangent in (2-6) is part of a conversion from rectangular to polar coordinates, this function may be represented as part of the mapping of two cartesian random variables, (a_d, b_d) to the polar random variables, (τ_a, r_a). This conversion can be fitted into the structure of (2-7) using the following definitions:

$$\mathbf{X} \triangleq \begin{bmatrix} a_d \\ b_d \end{bmatrix}, \tag{2-8}$$

$$\mathbf{Y} \triangleq \begin{bmatrix} r_a \\ \tau_a \end{bmatrix}, \tag{2-9}$$

where $\mathbf{Y}$ is the nonlinear function $f(\mathbf{X})$:

$$\mathbf{Y} = f(\mathbf{X}) = \begin{bmatrix} \left(x_1^{\,2} + x_2^{\,2} \right)^{1/2} \\ \dfrac{\text{atan}\,(x_2 / x_1)}{2\pi} \end{bmatrix}. \tag{2-10}$$

Equation (2-10) is the standard conversion from rectangular to polar coordinates with the angle τ_a on the interval, [-0.5,0.5). The nonlinear inverse function, $g(\mathbf{Y})$, is

$$\mathbf{X} = g(\mathbf{Y}) = \begin{bmatrix} y_1 \cos(2\pi y_2) \\ y_1 \sin(2\pi y_2) \end{bmatrix} \tag{2-11}$$

which is the mapping from polar to cartesian coordinates. From (2-11), the determinant of the Jacobian of $g(\mathbf{Y})$ is:

$$\begin{aligned} |J(g(\mathbf{Y}))| &= \begin{vmatrix} \dfrac{\partial}{\partial y_1} y_1 \cos(2\pi y_2) & \dfrac{\partial}{\partial y_1} y_1 \sin(2\pi y_2) \\ \dfrac{\partial}{\partial y_2} y_1 \cos(2\pi y_2) & \dfrac{\partial}{\partial y_2} y_1 \sin(2\pi y_2) \end{vmatrix}. \\ &= \begin{vmatrix} \cos(2\pi y_2) & \sin(2\pi y_2) \\ -2\pi y_1 \sin(2\pi y_2) & 2\pi y_1 \cos(2\pi y_2) \end{vmatrix} \\ &= 2\pi y_1 \end{aligned} \tag{2-12}$$

The PDF for a vector $\mathbf{X}$ of n jointly gaussian random variables is defined as

$$p_x(\mathbf{X}) = \frac{1}{(2\pi)^{n/2} |\mathbf{K}|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{X} - \mathbf{M})^T \mathbf{K}^{-1} (\mathbf{X} - \mathbf{M}) \right] \tag{2-13}$$

where $\mathbf{M}$ is the vector of means, i.e.,

$$\mathbf{M} = \left[E[x_1] \;\; E[x_2] \;\; \ldots \; E[x_n] \right]^T \tag{2-14}$$

and $\mathbf{K}$ is the covariance matrix for $\mathbf{X}$, i.e,

$$\mathbf{K} = \begin{bmatrix} E\left[x_1^2\right] & E[x_1 x_2] & \ldots & E[x_1 x_N] \\ E[x_2 x_1] & E\left[x_2^2\right] & \ldots & E[x_2 x_n] \\ \vdots & \vdots & \ddots & \vdots \\ E[x_n x_1] & E[x_n x_2] & \ldots & E\left[x_n^2\right] \end{bmatrix} \tag{2-15}$$

Since the (a, b) encoder signals are approximately gaussian, as seen in Figure 2.2 and modeled in (2-4), $p_x(\mathbf{X})$ will be a bivariate jointly gaussian distribution,

$$p_x(\mathbf{X}) = \frac{1}{2\pi \left|\mathbf{K}_X\right|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{X} - \mathbf{M}_X)^T \mathbf{K}_X^{-1} (\mathbf{X} - \mathbf{M}_X) \right] \tag{2-16}$$

with $\mathbf{X}$ as defined in (2-8), the means as defined in (2-4) are:

$$\mathbf{M}_X = \begin{bmatrix} a \\ b \end{bmatrix}, \tag{2-17}$$

and the covariance as defined in (2-4) is:

$$\mathbf{K}_X = \begin{bmatrix} A & \rho\sqrt{AB} \\ \rho\sqrt{AB} & B \end{bmatrix}. \tag{2-18}$$

Since (a_n, b_n) were found to be uncorrelated, the bivariate correlation coefficient, ρ_x is zero so that $\mathbf{K}$ simplifies to

$$\mathbf{K}_X = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}. \tag{2-19}$$

The variance, T, of τ_a may then be computed using the relation,

$$\text{VAR}[y_2] = E\left[y_2^2\right] - E[y_2] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_2(\mathbf{X}) p_x(\mathbf{X}) dx_1 dx_2 - \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_2(\mathbf{X}) p_x(\mathbf{X}) dx_1 dx_2 \tag{2-20}$$

where $f_2(\mathbf{X})$ is the second element in $f(\mathbf{X})$ as defined in (2-10), and $p_x(\mathbf{X})$ is defined in (2-16). However, the integrals in (2-20) are integrals of the product of an arctangent of the

variables of integration and an exponential of transcendental functions of the variables of integration for which it is very difficult to get a closed-form solution.

2.2.4 Tangent Line Approximation

Figure 2.5 shows an equal-probability contour of a joint Gaussian distribution in

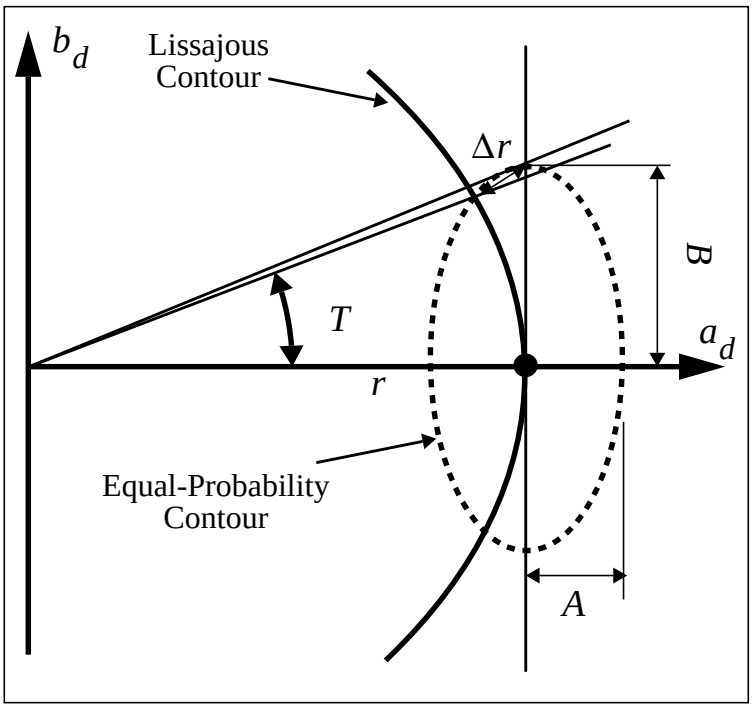

Figure 2.5: Tangent-Line Approximation

(a_d, b_d) as well as a small section of a Lissajous plot. A line tangent to the Lissajous plot at the central moment is shown as well. The variance T, as measured by the angle between the central moment and the equal-probability contour, is slightly different than that of the tangent line. As the radius r increases in proportion to the size of the contour, this difference will decrease up to the limiting case where the Lissajous plot becomes a straight line constant-probability contour.

The proportions of Figure 2.5 are exaggerated such that the long axis of the constant-probability contour is approximately half of the radius of the Lissajous plot. In the experimental system, the largest experimentally measured variance is about 16 A/D units while the smallest measured Lissajous radius is about 1600 A/D units. Assuming that the con-

stant-probability contour traces the σ^2 value, then the largest probability contour radius in the experimental system would be 16 A/D units, a factor of 100 smaller than the smallest Lissajous radius. Therefore the errors in T due to the tangent-line assumption will be negligible.

2.2.5 Approximate Probability Density Function of Intra-Line Position

The tangent-line assumption simplifies the efforts to determine T, the variance of τ_a, for three reasons:

1. The different orientations of the PDF equal-probability contours in Figure 2.4 can now be represented as a simple rotation about the central moment of angle, $-2\pi\tau$. The left-hand diagram of Figure 2.6 shows this rotation graphically. This rotation

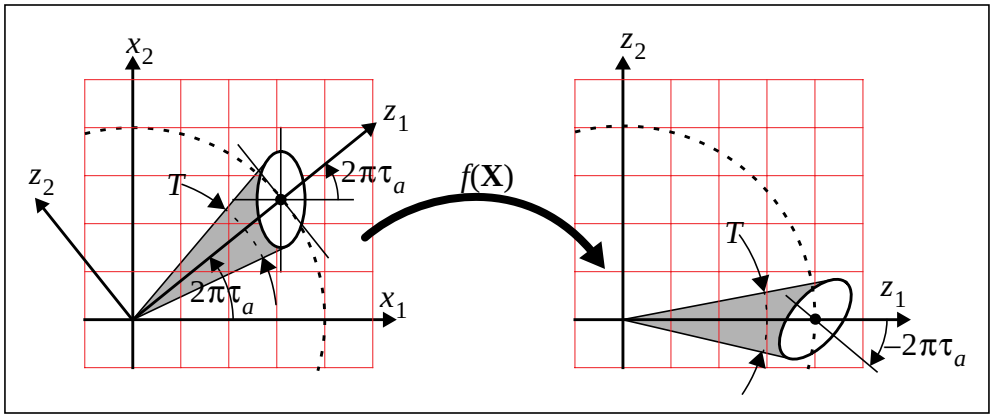

Figure 2.6: Coordinate System Rotation of Joint Gaussian PDF

is equivalent to a change in coordinate systems as shown by the function, $f(\mathbf{X})$, in the figure. The right-hand diagram in Figure 2.6 shows the PDF equal-probability contour in the new coordinate system, $\mathbf{Z}$. Unlike the arctangent operator, rotation is a linear operation on the jointly gaussian encoder signals. Therefore, the resulting distribution is still jointly gaussian, but with possibly nonzero correlation, ρ_Z. Also, since the variance T of τ_a is an angular measure, and since the radius of the central moment of the distribution is unchanged by $f(\mathbf{X})$, T is also unchanged by this rotation.

2. The mathematical difficulties in performing the integrations in (2-20) are due primarily to the fact that the integration is being performed along the axis of a polar

formulation (e.g., $[r, \tau_a]$) of a 2-variable jointly Gaussian distribution. With the tangent-line assumption, the integration is in the rotated coordinate frame, $\{y_1, y_2\}$, allowing a cartesian formulation for $p_X(\mathbf{X})$.

The rotation in Figure 2.6 can be represented as an operator on the two encoder signals, (a_d, b_d) using the following definitions:

$$\mathbf{X} \triangleq \begin{bmatrix} a_d \\ b_d \end{bmatrix}, \tag{2-21}$$

$$\mathbf{Z} \triangleq \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \tag{2-22}$$

so that $\mathbf{Z}$ is the linear function $f(\mathbf{X})$:

$$\mathbf{Z} = f(\mathbf{X}) = A\mathbf{X}. \tag{2-23}$$

where

$$A = \begin{bmatrix} \cos(2\pi\tau) & \sin(2\pi\tau) \\ -\sin(2\pi\tau) & \cos(2\pi\tau) \end{bmatrix}. \tag{2-24}$$

The PDF of $\mathbf{Z}$, $p_Y(\mathbf{Y})$ may be derived using (2-7). Since (2-23) is a linear mapping, the inverse function, $g(\mathbf{Y})$ is simply

$$\mathbf{X} = g(\mathbf{Z}) = A^{-1}\mathbf{Z}. \tag{2-25}$$

A rotation function such as (2-24) is orthonormal so the determinant, $|J(g(\mathbf{Y}))|$ is

$$\left|A^{-1}\right| = \frac{1}{|A|} = 1. \tag{2-26}$$

Therefore, using (2-7), the PDF of $\mathbf{Z}$ is

$$p_Z(\mathbf{Z}) = p_X(g(\mathbf{Z})) \cdot \left|A^{-1}\right| = p_X(A^{-1}\mathbf{Z}). \tag{2-27}$$

Substituting the PDF of $\mathbf{X}$ as given (2-16) into (2-27) yields:

$$\begin{aligned} p_Z(\mathbf{Z}) &= p_x(A^{-1}\mathbf{Z}) \\ &= \frac{1}{2\pi|\mathbf{K}_X|^{1/2}}\exp\left[-\frac{1}{2}\left(A^{-1}\mathbf{Z}-\mathbf{M}_X\right)^T\mathbf{K}_X{}^{-1}\left(A^{-1}\mathbf{Z}-\mathbf{M}_X\right)\right] \\ &= \frac{1}{2\pi\sqrt{|\mathbf{K}_X|}}\exp\left[-\frac{1}{2}(\mathbf{Z}-A\mathbf{M}_X)^T A^{-T}\mathbf{K}_X{}^{-1}A^{-1}(\mathbf{Z}-A\mathbf{M}_X)\right] \\ &= \frac{1}{2\pi\sqrt{|\mathbf{K}_X|}}\exp\left[-\frac{1}{2}(\mathbf{Z}-A\mathbf{M}_X)^T\left(A\mathbf{K}_XA^T\right)^{-1}(\mathbf{Z}-A\mathbf{M}_X)\right] \end{aligned} \tag{2-28}$$

where $\mathbf{K}_X$ and $\mathbf{M}_X$ are as defined in (2-17) and (2-19), respectively. Since $|A| \equiv 1$ from (2-26), equation (2-28) is another jointly Gaussian distribution with mean vector,

$$\mathbf{M}_Z = A\mathbf{M}_X \tag{2-29}$$

and covariance matrix,

$$\mathbf{K}_Z = A\mathbf{K}_XA^T. \tag{2-30}$$

Substituting (2-17) and (2-24) into (2-29) yields

$$\begin{aligned} \mathbf{M}_Z &= \begin{bmatrix} M_{Z1} \\ M_{Z2} \end{bmatrix} \\ &= \begin{bmatrix} \cos(2\pi\tau) & \sin(2\pi\tau) \\ -\sin(2\pi\tau) & \cos(2\pi\tau) \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} \\ &= \begin{bmatrix} a\cos(2\pi\tau) + b\sin(2\pi\tau) \\ -a\sin(2\pi\tau) + b\cos(2\pi\tau) \end{bmatrix} \end{aligned} \tag{2-31}$$

and substituting (2-19) and (2-24) into (2-30) yields

$$\begin{aligned} \mathbf{K}_Z &= \begin{bmatrix} K_{Z11} & K_{Z12} \\ K_{Z12} & K_{Z22} \end{bmatrix} \\ &= \begin{bmatrix} \cos(2\pi\tau) & \sin(2\pi\tau) \\ -\sin(2\pi\tau) & \cos(2\pi\tau) \end{bmatrix}\begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}\begin{bmatrix} \cos(2\pi\tau) & -\sin(2\pi\tau) \\ \sin(2\pi\tau) & \cos(2\pi\tau) \end{bmatrix} \\ &= \begin{bmatrix} A\cos^2(2\pi\tau) + B\sin^2(2\pi\tau) & (B-A)\sin(2\pi\tau)\cos(2\pi\tau) \\ (B-A)\sin(2\pi\tau)\cos(2\pi\tau) & A\sin^2(2\pi\tau) + B\cos^2(2\pi\tau) \end{bmatrix} \end{aligned} \quad . \tag{2-32}$$

2.2.6 Variance of Intra-Line Position

The rotated distribution, as expressed in (2-28), aligns the axes so that z_1 is parallel to the radial line through the distribution mean while z_2 is perpendicular to z_1. The random variables, τ and z_2 are related by

$$\tau = \frac{\text{atan}\,(z_2/z_1)}{2\pi}. \tag{2-33}$$

Since $M_{Z1} = z_1$ from (2-21), (2-23), (2-24) and (2-31), (2-33) may be rewritten as

$$\tau = \frac{\text{atan}\,(z_2/M_{Z1})}{2\pi}. \tag{2-34}$$

By looking at the marginal distribution along z_2 an expression for the approximate variance, $\hat{T}$ of τ may be derived as

$$\begin{aligned}
\hat{T} &= E\left[\left(\frac{\text{atan}\,(z_2/M_{Z1})}{2\pi}\right)^2\right] \\
&= \frac{1}{4\pi^2}\int_{-\infty}^{\infty} \text{atan}^2(z_2/M_{Z1}) \cdot p_{Z2}(z_2)dz_2 \\
&= \frac{1}{4\pi^2}\int_{-\infty}^{\infty} \text{atan}^2\left(\frac{z_2}{M_{Z1}}\right) \cdot \frac{\exp\left(\frac{-z_2^{\,2}}{2K_{Z22}}\right)}{\sqrt{2\pi K_{Z22}}}dz_2 \\
&= \frac{1}{4\pi^2\sqrt{2\pi K_{Z22}}}\int_{-\infty}^{\infty} \text{atan}^2\left(\frac{y_2}{M_{Z1}}\right) \cdot \exp\left(\frac{-z_2^{\,2}}{2K_{Z22}}\right)dz_2
\end{aligned} \tag{2-35}$$

The first three terms of the Taylor series for $\text{atan}^2(z_2/M_{Z1})$ are

$$\text{atan}^2(z_2/M_{Z1}) \cong \frac{z_2^2}{M_{Z1}^2} - \frac{2z_2^4}{3M_{Z1}^4} + \frac{23z_2^6}{45M_{Z1}^6}. \tag{2-36}$$

Since $y_2 \ll M_{Z1}$, the higher-order terms in (2-36) are negligible. The integral (2-35) may be computed as the sum of each of the Taylor-series terms multiplied by an exponential:

$$\hat{T} \cong \frac{1}{4\pi^2}\left(\frac{\int_{-\infty}^{\infty} z_2^2 \exp\left(\frac{-z_2^{\,2}}{2K_{Z22}}\right) dz_2}{M_{Z1}^2\sqrt{2\pi K_{Z22}}} - \frac{2\int_{-\infty}^{\infty} z_2^4 \exp\left(\frac{-z_2^{\,2}}{2K_{Z22}}\right) dz_2}{3M_{Z1}^4\sqrt{2\pi K_{Z22}}} + \frac{23\int_{-\infty}^{\infty} z_2^6 \exp\left(\frac{-z_2^{\,2}}{2K_{Z22}}\right) dz_2}{45M_{Z1}^6\sqrt{2\pi K_{Z22}}} \right) \tag{2-37}$$

The expression (2-37) was simplified with the *MAPLE-V* symbolic mathematics package to

$$\hat{T} \cong \frac{1}{4\pi^2\sqrt{K_{Z22}}}\left(\frac{K_{Y22}^{3/2}}{M_{Z1}^2} - \frac{2K_{Y22}^{5/2}}{M_{Z1}^4} + \frac{23K_{Y22}^{7/2}}{3M_{Z1}^6} \right). \tag{2-38}$$

where M_{Z1} and K_{Z22} are functions of (a, b, A, B, τ_a) defined in (2-31) and (2-32), respectively.

The $3\sigma = 3\sqrt{\hat{T}}$ constant-probability contour for the dataset used to generate the histograms in Figure 2.2 and Figure 2.3 may be computed from (2-38) and plotted as function of τ_a. Figure 2.7 shows this 3σ contour for three different noise scenarios (i.e., 3 different

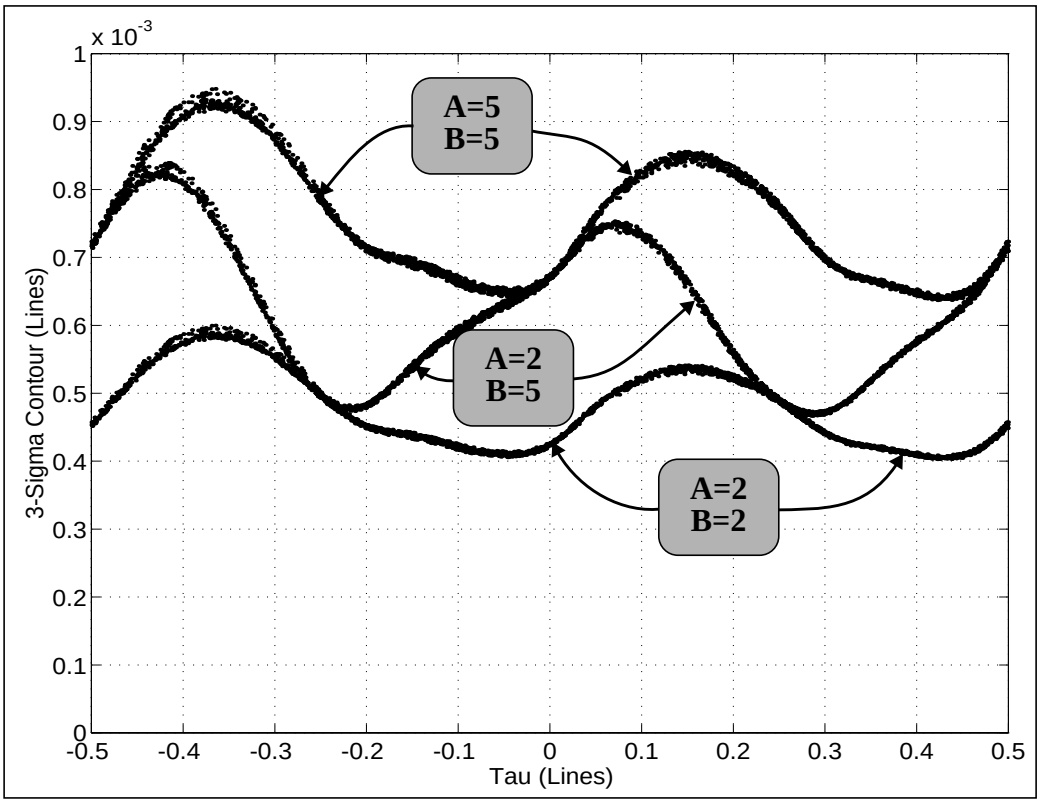

Figure 2.7: 3-Sigma Contour of Rough Position vs. Rough Position

constant values for A and B). Note that the even if the (A, B) variances are equal, the variance, T, of τ still changes with τ because the radius of the experimental dataset varies as a function of τ.The variance T represents a physical limitation on the accuracy of any calibration technique. The plot in Figure 2.7 indicates that measurement noise in the experimental system may cause almost 0.001-line errors in the value of τ_a.

2.3 Calibration Error

The signal flow diagram in Figure 1.2 on page 4 shows that the actual intra-line position, τ, is dependant on three terms:

$$\tau = \tau_a + \varepsilon(\tau_a) + \zeta, \tag{2-39}$$

where τ_a is the rough intra-line position, $\varepsilon(\tau_a)$ is the calibration term to correct for non-ideal properties of the encoder, and ζ is any remaining calibration error due to inaccuracies in the calibration term, $\varepsilon(\tau_a)$. Ideally, $\varepsilon(\tau_a)$ will correct for all errors in the rough position estimate, τ_a, so that $\zeta = 0$.

The properties of the calibration error are more difficult to quantify than the properties of the measurement noise. Small inaccuracies in $\varepsilon(\tau_a)$ may be caused by variations in the encoder line width or spacing for a particular adjacent line-pair on the encoder disk. Inaccuracies may also be caused by inaccurate calibration technique. The accuracy of the line spacing/size on the encoder disk is unknown; however these distortions are known to be small relative to the general calibration distortions addressed by the calibration function, $\varepsilon(\tau_a)$.

CHAPTER 3: MODEL-BASED CALIBRATION TECHNIQUE

3.1 Approach

In Section 1.4, two methods for analog encoder calibration were reviewed. The first method [2] was based on a parametric model of the encoder output waveforms as a function of intra-line position. The second method [3] developed a table-based calibration using a position estimate that was based on the assumption that the encoder is moving at a constant velocity. In this chapter a new approach is developed for estimation of the true position during the calibration process using a Kalman filter [11]. The filter uses a stochastic dynamic model of the encoder, the actuator, the current input, and the load to generate a minimum-variance estimate of the position as a function of time based on the rough intra-line position estimate, τ_a, from the ideal encoder model.

The intent of this approach is to provide an off-line technique for generating a calibration table where an encoder/actuator is moved over a small range of motion for a short time while the encoder outputs signals and command actuator currents are sampled and recorded at the normal servo frequency (e.g., 1-msec. for a 1-kHz servo system). Figure 3.1 shows

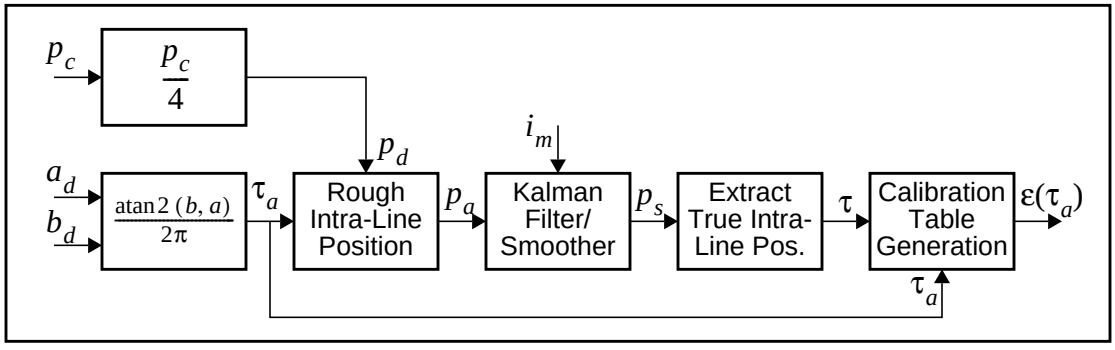

Figure 3.1: Kalman Calibration Algorithm Schematic

the algorithm in schematic form. The dataset is passed through a Kalman filter/smoother to provide a minimum-error estimate of true position. The difference between the minimum-error estimate, τ, and the rough estimate, τ_a, (based on the ideal model Section 1.3.1) can then be used to generate a lookup-table of correction factors, $\varepsilon(\tau_a)$, that map the rough (and easily-computed) intra-line position estimate to the true intra-line position.

A key assumption in this new approach to developing a calibration table is that a single

table will satisfy the relation in (2-1) for *all* adjacent line pairs on the encoder wheel. Therefore this approach is unsuited for encoders with large differences between adjacent pairs of encoder lines on the encoder disk.

3.2 Process Model Derivation

3.2.1 Continuous-Time Model Derivation

The encoder to be calibrated is assumed to be mounted on a rotary-joint system with an electromagnetic actuator. A linear state-space model of the system is needed to formulate the Kalman filter used in estimating the encoder position. While the Kalman filter used for the calibration supports highly complex models (even nonlinear models using the Extended Kalman Filter [12]), good calibration results are obtained from an experimental system using a linear process model.

The actuator is assumed to be driven by an ideal current source, so that the dynamics due to the inductance and back-EMF of the motor can be ignored. This simplified dynamic model is described by the second-order torque equation,

$$J\ddot{\theta} + B_F\dot{\theta} + K_T i_m = 0 \tag{3-1}$$

where J is the inertia of the joint as seen by the axis of rotation, B_F is the viscous damping, K_T is the motor torque constant, i_m is the commanded current, and $(\dot{\theta}, \ddot{\theta})$ are the first and second time-derivatives of the joint's angular position, θ. The system parameters, $[J, B_F, K_T]$ are assumed to be time-invariant and that the current, i_m is assumed to be a deterministic control input to the system.

Equation (3-1) represents a linear, time-invariant system which can be expressed in state-space notation by,

$$\dot{x}(t) = Ax(t) + Bu(t)\,. \tag{3-2}$$

where $x \triangleq \left[\theta\ \dot{\theta}\right]^T$, $u \triangleq i_m$,

$$A \triangleq \begin{bmatrix} 0 & 1 \\ 0 & \frac{-B_F}{J} \end{bmatrix} \tag{3-3}$$

and

$$B \triangleq \begin{bmatrix} 0 \\ \hline \dfrac{-K_T}{J} \end{bmatrix}. \tag{3-4}$$

3.2.2 Continuous Gauss-Markov Process Model

The system equation in (3-2) forms a deterministic model of the system. This model is only an approximation; there may be unmodeled effects and errors in the model parameters $[J, B, K_T]$. Therefore, it is useful to augment this model to incorporate the unmodeled effects as stochastic disturbances.

A continuous Gauss-Markov process (CGMP) is a continuous linear Markov process whose state derivative depends linearly on the current state value, x, a control input, u, and a zero-mean Gaussian purely random input, w_c. This is expressed as,

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) + \Gamma_c(t)w_c(t) \tag{3-5}$$

where $w_c(t)$ is a vector of Gaussian random variables with zero-mean and covariance W_c, denoted $N(0, W_c)$.

Comparing (3-2) and (3-5) the system equation in (3-2) can be formulated as a CGMP by simply augmenting it with a stochastic disturbance input:

$$\dot{x}(t) = Ax(t) + Bu(t) + \Gamma_c(t)w_c(t). \tag{3-6}$$

With this modification, the state vector is now itself a Gaussian random variable (x is dependant on a linear combination of Gaussian variables) and can be expressed as

$$x(t) \cong N(\bar{x}(t), X(t)). \tag{3-7}$$

The mean value of the state vector, $\bar{x}(t)$, propagates as

$$\dot{\bar{x}}(t) = A\bar{x}(t) + Bu(t). \tag{3-8}$$

Note that $\bar{w}_c(t)$ does not appear in (3-8) because $w_c(t)$ is assumed zero-mean. Also note that the state vector covariance is independent of B since $u(t)$ is considered a deterministic input to the system.

Bryson [14] showed that if the time correlations of the process noise inputs, $w_c(t)$, approach 0 for time-intervals near or larger than the characteristic times of the system, then

the process noise of the system may be treated as a white-noise input with some spectral density Q. In this case the state covariance vector, $X(t)$ propagates in time according to

$$\dot{X}(t) = AX(t) + X(t)A^T + \Gamma_c Q \Gamma_c^T . \tag{3-9}$$

Equation (3-9) is the continuous-time Lyapunov equation. In order to simplify subsequent expressions, the time-function notation will be dropped at this point although it remains implicitly. The CGMP equations for this system can then be written more simply as

$$\dot{\bar{x}} = A\bar{x} + Bu \tag{3-10}$$

and

$$\dot{X} = AX + XA^T + \Gamma_c Q \Gamma_c^T \tag{3-11}$$

3.2.3 Conversion to a Discrete Gauss-Markov Process

A discrete Gauss-Markov process (DGMP) is a discrete linear Markov process whose next state depends linearly on the current state, x, a deterministic input, u, and a Gaussian process noise input, w with distribution, $N(\bar{w}(k), W(k))$. The general expression for a time-invariant DGMP is a mean state vector,

$$\bar{x}(k+1) = \Phi(k)\bar{x}(k) + \Psi(k)u(k) + \Gamma(k)\bar{w}(k) \tag{3-12}$$

and a state covariance vector,

$$X(k+1) = \Phi(k)X(k)\Phi^T(k) + W(k) . \tag{3-13}$$

Note that the process noise, $w(k)$ is usually assumed to be zero mean, implying that $\bar{w}(k) \equiv 0$.

Equation (3-13) is the discrete-time Lyapunov equation. Assuming that the control inputs are applied by a zero-order hold, the conversion of this CGMP to a DGMP representation involves the following relations:

$$\Phi = \exp(A \cdot T_s) , \tag{3-14}$$

$$\Psi = \int_0^{T_s} e^{At} dt \cdot B , \tag{3-15}$$

and

$$\Gamma = \int_0^{T_S} e^{At} dt \cdot \Gamma_c . \tag{3-16}$$

where T_s is the sampling interval of the discrete system. In addition, an expression for $W(k)$, the covariance of the DGMP process noise at sample-time k, is needed which relates to the spectral density, Q, of the process noise in the continuous-time system. Bryson [14] showed that for zero-mean CGMP process noise (i.e., $\bar{w}_c(t) = 0$) with time-correlations, $E\left[w(t)w^T(\tau)\right] \to 0$ for $|t-\tau| > T_c$ where T_c « than the characteristic times of the system matrix, A, the relation between Q and W is approximately:

$$W \cong \int_0^{T_s} e^{At} \Gamma_c Q \Gamma_c{}^T \left[e^{At}\right]^T dt \tag{3-17}$$

The DGMP covariance, W, will have the same covariance as the continuous-time system at times that are integer multiples of the sampling interval.

While the integral computations in (3-14) through (3-17) can be done individually, a technique developed by Van Loan [13] can be used to compute these terms simultaneously in terms of a single matrix exponential. This technique is especially useful since well-conditioned numeric matrix exponential algorithms exist. A detailed review of the algorithm is given in Appendix A.

The integrals in (3-14) through (3-17) may be computed by defining a matrix $\hat{C}$ in terms of the CGMP parameters $[A, \Gamma, Q]$ such that

$$\hat{C} \triangleq \begin{bmatrix} -A^T & I & 0 & 0 \\ 0 & -A^T & \Gamma_c Q \Gamma_c{}^T & 0 \\ 0 & 0 & A & I \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3-18}$$

and its matrix exponential, $\hat{S}$:

$$\hat{S} \triangleq \exp(\hat{C}t) = \begin{bmatrix} \hat{F}_1 & \hat{G}_1 & \hat{H}_1 & \hat{K}_1 \\ 0 & \hat{F}_2 & \hat{G}_2 & \hat{H}_2 \\ 0 & 0 & \hat{F}_3 & G_3 \\ 0 & 0 & 0 & \hat{F}_4 \end{bmatrix} \tag{3-19}$$

From the definitions, the DGMP parameters may be extracted from $\hat{S}$ as follows:

$$\Phi = \hat{F}_3^T \tag{3-20}$$

$$\Psi = \hat{G}_3^T B \tag{3-21}$$

$$\Gamma = \hat{G}_3^T \Gamma_c \tag{3-22}$$

$$W = \hat{F}_3^T \hat{G}_2 \tag{3-23}$$

Computation of (3-19) through (3-23) is normally done numerically. However, a symbolic representation was derived to see the relations between the CGMP and the DGMP parameters directly for the system in equations (3-14) through (3-17):

$$\Phi = \begin{bmatrix} 1 & -\frac{J}{B_F}\left[e^{-\alpha} - 1\right] \\ 0 & e^{-\alpha} \end{bmatrix} \tag{3-24}$$

$$\Psi = \begin{bmatrix} \frac{K_T J}{B_F^2}\left(1 - e^{-\alpha}\right) - \frac{K_m T_s}{B_F} \\ \frac{K_m}{B_F}\left(e^{-\alpha} - 1\right) \end{bmatrix} \tag{3-25}$$

$$W = \frac{Q\Gamma_{C_{21}}^2}{2} \begin{bmatrix} \frac{J^3\left(4e^{-\alpha} - e^{-2\alpha} - 3\right) + 2J^2 B_F T_S}{B_F^3} & \frac{J^2\left(1 + e^{-2\alpha} - e^{-\alpha}\right)}{B_F^2} \\ \frac{J^2\left(1 + e^{-2\alpha} - e^{-\alpha}\right)}{B_F^2} & \frac{J\left(1 - e^{-2\alpha}\right)}{B_F} \end{bmatrix} \tag{3-26}$$

where

$$\alpha \triangleq \frac{B_F T_S}{J} \tag{3-27}$$

Note that in (3-26), $\Gamma_C(1, 1)$ was assumed to be zero to simplify the symbolic expression for W. This turns out to be a good assumption since most of the process noise will probably come in through the second state due to uncertainty in the actuator model and the current command through the second state. However, there is no need to require this assumption in the numerical implementation since the Van Loan algorithm will give the correct result for any Γ_c. The important thing to note from the symbolic derivations in (3-24) through (3-27) is the relative importance of each of the physical system parameters on the discrete formulation.

Equations (3-24) through (3-27) can be used directly in the final DGMP formulation given in (3-12) and (3-13). Note that the computation for Γ from (3-16) is not shown here since it is not used in the final DGMP formulation.

3.3 Kalman Calibration Filter Derivation

In the previous section, a DGMP model was derived for a simplified second-order plant consisting of an encoder mounted on an actuator with known inertia, viscous damping and actuator constant (J, B_F, and K_T respectively). This process model must now be augmented to incorporate the encoder output signals.

3.3.1 Measurement Vector

In Section 1.3.1, an ideal model of the encoder outputs was used to derive a function which maps these outputs, (A, B) to an intra-line position, τ according to the relation,

$$\tau_a = \frac{\operatorname{atan2}(B, A)}{2\pi}. \tag{3-28}$$

This model assumes that the outputs are exactly sinusoidal with identical magnitudes and frequencies of 2π, with exactly 90° of phase difference (quadrature phase).

The Kalman filter formulation models the sensor measurement vector, z, as the linear combination of a true measurement of the system states, x, and measurement noise, v:

$$z = Cx + v. \tag{3-29}$$

If the ideal model is used to compute a position estimate, then any errors in this estimate can be incorporated as measurement "noise." The filter assumes that this noise is Gaussian with zero-mean and variance V. Since the ideal model provides an estimate of only the first (position) state, the sensor distribution matrix, C, is defined as

$$C \triangleq \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{3-30}$$

3.3.2 Kalman Filter Formulation

The Kalman filter can be computed efficiently in a recursive formulation which, for every sample time k, generates an optimal estimate of the mean and covariance of the state vector based on (1) the previous mean and covariance estimates at time $k-1$, and (2) the process inputs and measurements at time k. The filter is really a two-stage algorithm; it first generates a "time update" which is a prediction of the state mean and covariance based on the previous state and process inputs. Then it does a "measurement update" which generates a new estimate which is based on the "time update" estimate and the sensor measurements. The relative influence of the time update estimate and the measurements on the measurement update depends on the "Kalman Gain," K, which is a time-varying parameter that indicates the relative reliability of the information sources based on the current state-covariance, P, and the known sensor covariance, V.

The filter algorithm is outlined below using notation from [14]:

TIME UPDATE

$$\bar{x}(k+1) = \Phi\hat{x}(k) + \Psi u(k) \tag{3-31}$$

$$P(k+1) = \Phi\hat{P}(k)\Phi^T + W \tag{3-32}$$

MEASUREMENT UPDATE

$$K(k+1) = P(k+1)C^T\left[V + CP(k+1)C^T\right]^{-1} \tag{3-33}$$

$$\hat{P}(k+1) = P(k+1) - K(k+1)\left[V + CP(k+1)C^T\right]K(k+1)^T \tag{3-34}$$

$$\hat{x}(k+1) = \bar{x}(k+1) + K(k+1)\left[z(k+1) - C\bar{x}(k+1)\right] \tag{3-35}$$

where z is the position estimate as defined in (3-29) for $0 \le k < N$, with initial conditions $\hat{x}(0) = x_0$ and $\hat{P}(0) = P_0$.

3.3.3 Optimal Smoother Formulation

The Kalman filter provides an optimal estimate of the state. The filter is formulated as a DGMP, which only looks to previous states to generate a new estimate. However, since the proposed calibration algorithm is implemented off-line, the entire dataset is available. Clearly, improved estimates may be made if all available data before *and* after a given point is used to make given estimate.

One type of smoother is the forward and backward smoother which consists of two Kalman filters, one which runs forward though the data set from $k = 1$ to $k = N$, while the other runs backwards through the dataset from $k = N$ to $k = 1$. The forward filter generates an optimal estimate based on all previous data while the backward filter generates an optimal estimate based on all subsequent data. Since the two estimates are statistically independent (see [14]), they may be combined to form an estimate that is optimal over the entire data set.

Running a Kalman filter in time-reverse order requires a few modifications to the forward filter formulation of equations (3-31) through (3-35). The backward Kalman filter is formulated as follows:

TIME DOWNDATE

$$\bar{x}(k-1) = \Phi^{-1}\hat{x}(k) - \Phi^{-1}\Psi u(k) - \Phi^{-1}\Gamma\bar{w}(k) \tag{3-36}$$

$$P(k-1) = \Phi^{-1}\hat{P}(k)\Phi^{-T} + W \tag{3-37}$$

MEASUREMENT DOWNDATE

$$K(k-1) = P(k-1)C^T\left[V + CP(k-1)C^T\right]^{-1} \tag{3-38}$$

$$\hat{P}(k-1) = P(k-1) - K(k-1)\left[V + CP(k-1)C^T\right]K(k-1)^T \tag{3-39}$$

$$\hat{x}(k-1) = \bar{x}(k-1) + K(k-1)\left[z(k-1) - C\bar{x}(k-1)\right] \tag{3-40}$$

with $k = (N+1)$ to 2, and initial conditions, $\hat{x}(N+1) = x_N$ and $\hat{P}(N+1) = P_N$.

The smoother uses both the forward and reverse estimates of the state mean and covariance to generate an optimal estimate. If the forward and reverse state mean estimates at time k are $\hat{x}_F(k)$ and $\hat{x}_B(k)$, respectively, and the forward and reverse state covariance estimates are similarly $\hat{P}_F(k)$ and $\hat{P}_B(k)$, then the smoother algorithm can be formulated as follows:

$$K_s(k) = \hat{P}_F(k)\,[\hat{P}_F(k) + \hat{P}_B(k)]^{-1} \tag{3-41}$$

$$\bar{x}_S(k) = \bar{x}_F(k) + K_s(k)\,[\bar{x}_F(k) - \bar{x}_B(k)] \tag{3-42}$$

$$P_s(k) = \hat{P}_F(k) - K_s(k)\hat{P}_F(k) \tag{3-43}$$

with $k = 1$ to N. Bryson and Frazier [15] showed that the backwards filter in (3-36) through (3-40) and the smoother in (3-41) through (3-43) may be combined into a single step. However, these steps were kept separate in the experimental implementation to simplify debugging efforts.

3.4 Correction Table Generation

The result of applying the Kalman filter/smoother in the previous section to the system in equations (3-12) through (3-17) is an optimal estimate of the system states, $\bar{x}_s(t)$, based on the measured rough position estimate, τ_a and the known motor current, i_m. The intra-line position, τ may be extracted from the first state mean, $\bar{p}_s(t)$ using

$$\tau(t) = \text{fract}(\bar{p}_s(t)) - \begin{cases} 1 & \text{fract}(\bar{p}_s(t)) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}. \tag{3-44}$$

where the conditional part of (3-44) maps the values of τ from the interval, (0,1] to the interval, [-0.5,0.5) as defined in (1-5).

Assuming that $\tau(t)$ is an accurate estimate of the true intra-line position, the calibration term, $\varepsilon(\tau_a(t))$, for the rough position estimate, $\tau_a(t)$, as determined from the encoder measurements (a_d, b_d) at time t, is

$$\varepsilon(\tau_a(t)) \cong \tau(t) - \tau_a(t) \qquad 0 \leq t \leq N \tag{3-45}$$

as defined by (2-39). The time vectors, $\tau_a(t)$ and $\varepsilon(\tau_a(t))$ can both be sorted by τ_a yielding a correction term lookup table with τ_a as the key and $\varepsilon(\tau_a)$ as the value. This table can then be used to compute the correct intra-line position, τ, based on the rough position estimate, τ_a using the relation,

$$\tau = \tau_a + \varepsilon(\tau_a). \tag{3-46}$$

If a future measurement results in a value for τ_a which is not a key in the lookup table, then linear interpolation may be used to estimate the correct calibration term, $\varepsilon(\tau_a)$.

CHAPTER 4: EXPERIMENTAL IMPLEMENTATION

4.1 Experiment Setup

The minirobot discussed in Section 1.5 is used as the experimental platform. Experiments were limited to a single joint with an analog encoder, some interface circuitry, and a digital control system, configured as shown in Figure 4.1. The encoder is connected to a

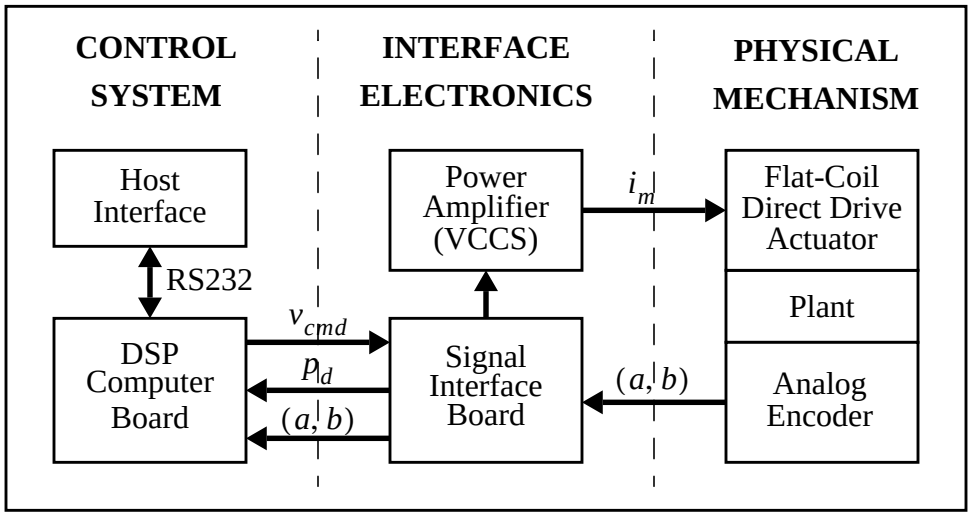

Figure 4.1: Experiment Configuration

rigid mechanical link with known inertia (J) and viscous damping (B_F) and is driven by a direct-drive actuator with a known torque constant, (K_T). The actuator is powered by a linear power amplifier configured as a voltage controlled current source (VCCS). The signal interface board provides signal conditioning for the encoder outputs (offset and gain adjustments), a digital encoder interface that maintains the encoder position to 0.25 line resolution, and interface electronics for the power amplifier. The digital encoder interface first thresholds the (a, b) signals (negative and positive voltages map to digital '0's and '1's respectively) to simulate the output of a digital encoder, and then passes the signals to a digital quadrature decoder/counter circuit which yields an absolute line position to a resolution of 0.25 of an encoder line.

The DSP computer board contains a TMS 320C30 DSP processor system for implementing control software, multiple 12-bit A/D and D/A converters and a digital I/O parallel port (see [9] for more details). The processor controls motor current by commanding par-

ticular voltages to one of its D/A converters. This voltage $(-10 \le v_{\text{cmd}} \le 10)$ determines the motor current, i_m in the VCCS power amplifier. The processor reads the digital position counter, p_c from the digital parallel port and the (a, b) encoder signals via the A/D converters. The A/D converters are configured to sample the analog (a, b) signals simultaneously (maximum 4-nanosecond phase delay) to avoid phase-induced position estimate distortions. The 12-bit digital encoder values, (a_d, b_d) from the A/D converters are used to determine the intra-line position, τ while the position counter values are used to compute the digital encoder position, p_d to a resolution of 0.25 of a line as shown in 1.2 on page 4.

The system is configured to run at a 1-kHz servo rate, sampling both A/D converters and the digital encoder count every millisecond and changing the D/A converter output every 1 millisecond as necessary.

4.2 Data Collection

4.2.1 Trajectory Generation

During data collection for calibration, the normal PID control algorithm is suspended. Instead, a simple open loop current trajectory is used which will generate a desirable position trajectory. The following aspects of the trajectory are important:

1. The trajectory should avoid low velocities as much as possible to avoid high short-time correlations of the position correction factor, $\varepsilon(t)$, for successive samples. High correlation will cause the apparent variance in the measurement noise to decrease so that the Kalman filter relies on the measurement estimates too much.
2. The trajectory should avoid high accelerations which might excite unmodeled dynamics such as mechanical resonances and distort the calibration process.

In order to meet the above criteria, an open-loop triangle-shaped current profile is chosen for a position trajectory centered about the full range of joint motion. The trajectory is designed to avoid joint limit stops (unmodeled high acceleration events) while still providing continuous acceleration.

4.2.2 Data Collection Method

During data collection, the motor current trajectory described above is sent to the actuator. At each 1-millisecond sample interval, the encoder digital position, p_d, the digitized encoder outputs, (a_d, b_d), and the commanded current, i_m, are sampled and added to a large data buffer in the DSP processor's on-board memory. The sample buffer is configured to contain 2849 samples.

Upon completion of the data collection, the data buffer is uploaded to a host computer via the DSP processor's RS232 port. Currently, the remainder of the calibration algorithm is implemented in MATLAB[1] on the host computer (Sun Workstation) instead of the DSP board for reasons of software development/debugging convenience. However, there is no reason why the complete algorithm could not be re-written in a high-level language like *C* on the DSP processor itself.

4.2.3 Raw Encoder Data

The Lissajous plot for the experimental encoder is shown in Figure 4.2. A modified version of this plot, called a "Spiraled Lissajous plot," is shown in Figure 4.3. This plot shows the Lissajous plot as a function of time by increasing the radius of the samples as a function of time and interpolating a splined curve through the set of points. Note that only the first 1000 of the 2849 data samples are shown in this plot to improve the visual clarity of the figure. This plot shows the time-distribution of the samples around the Lissajous contour. The regions where the points become very close together occur at the times when the joint velocity approaches zero during a change in the direction of motion.

A rough estimate of angular position, p_a, can be made by merging the digital position estimate, p_d, with the ideal intra-line position estimate, τ_a, at every time sample using,

---

1. MATLAB is a commercially available software package that specializes in matrix computations and supports an interpretive script-like language. The package is available from The MathWorks, Inc. in Natick, Massachusetts.

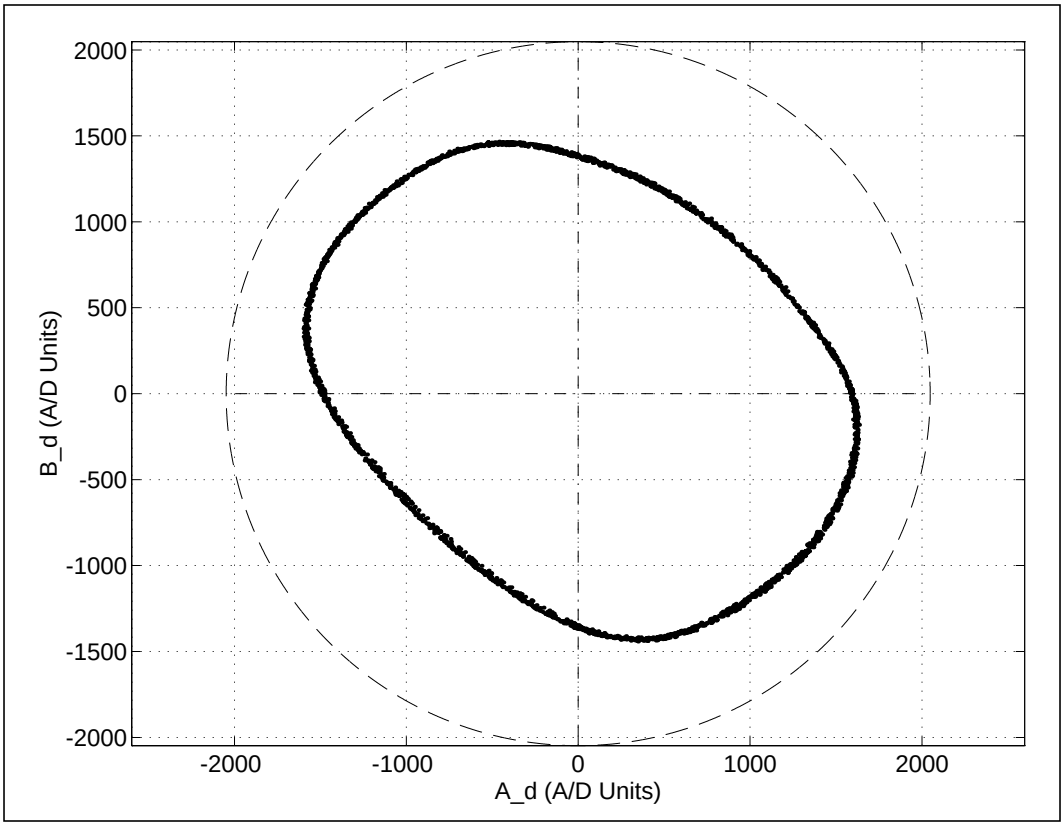

Figure 4.2: Lissajous Plot of Experiment Data

$$p_a = \frac{2\pi}{N_L}\left[\text{int}(p_d) + \tau_a + \begin{cases} 1 & \text{fract}(p_d) - \tau_a > \frac{1}{2} \\ -1 & \text{fract}(p_d) - \tau_a < -\frac{1}{2} \\ 0 & \text{otherwise} \end{cases}\right] \tag{4-1}$$

where the intra-line distance, τ_a is as defined in (1-5):

$$\tau_a = \frac{\text{atan2}\,(a, b)}{2\pi} \tag{4-2}$$

and p_d as defined in (1-3). The domain of τ_a is a closed contour over the interval [0.5,0.5) (i.e., the values -0.5 and 0.5 are identical). The conditional part of (4-1) is needed to "un-wrap" τ_a so that it may be used to compute an estimate of the inter-line position, p_a. Table 1 shows several examples of how this computation is performed. Note that (4-1) assumes the digital position counter value, p_d is aligned such that its value is a multiple of 4 when the encoder (a, b) outputs are in the first quadrant. If this is not true then a modified version of (4-1) will be needed to compensate.

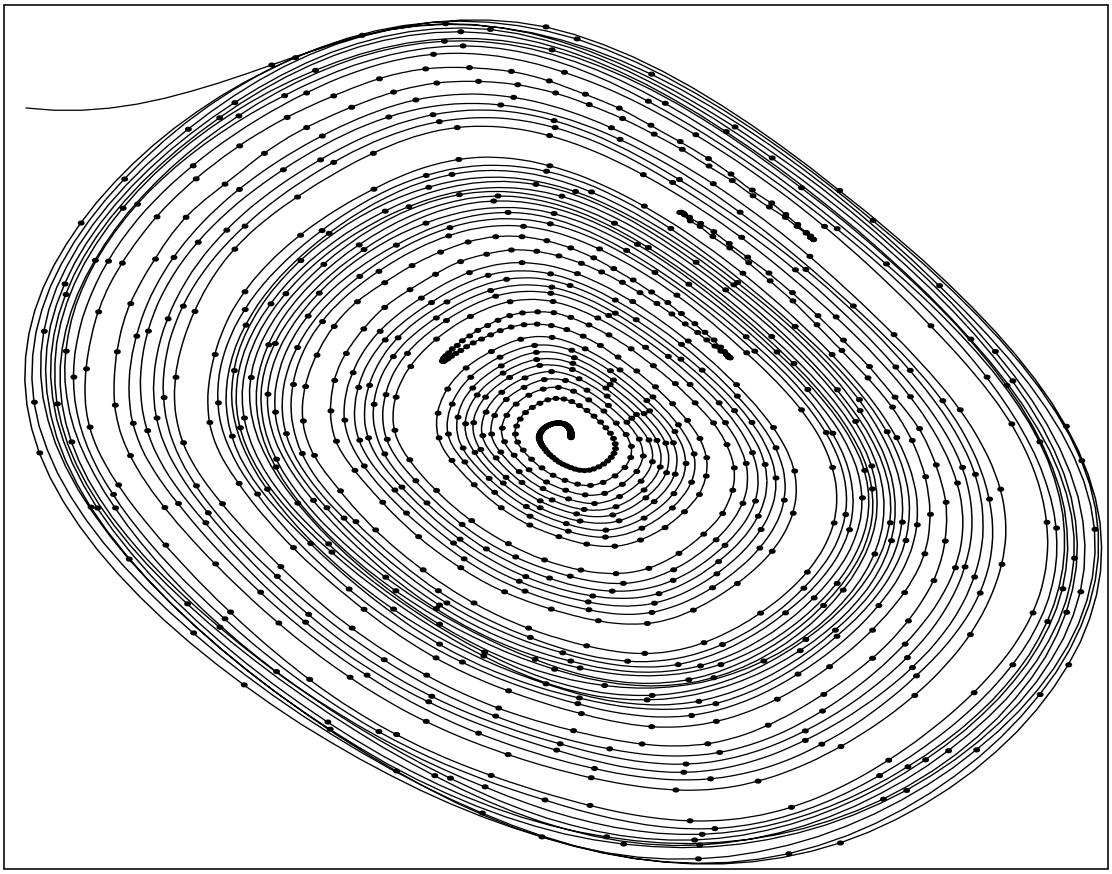

Figure 4.3: Spiraled Lissajous Plot of Raw Encoder Data (1-second)

Table 1: Examples of Inter-Line Position Computation

p_d	τ_a	fract(p_d) – τ_a	Value of Conditional	p_a
12.25	0.33	-0.08	0	12.33
12.75	-0.20	0.95	-1	12.80
-6.25	-0.33	0.08	0	-6.33
-6.75	0.20	-0.95	1	-6.80

This approximate inter-line position, p_a, as computed from the rough intra-line position, τ_a, can be used to generate a rough estimate of the state vector, x_r, over the entire data set:

$$x_r(k) = \begin{bmatrix} p_a(k) \\ p_a(k) - p_a(k-1) \end{bmatrix} \qquad 2 \le k \le N_{ds} \tag{4-3}$$

where N_{ds} is the number of samples in the data set, k is the sample index $(1 \le k \le N_{ds})$, and the velocity state is computed as the first difference of the position state. The initial condition, $x_r(0)$, is assumed to be

$$x(0) = \begin{bmatrix} p_a(0) \\ 0 \end{bmatrix}. \tag{4-4}$$

Figure 4.4 shows the values of x_r for the experimental dataset. Note that while the position estimate looks fairly smooth, the velocity estimate shows a large amount of variance. Since the physical system is undergoing smooth commanded accelerations during the data collection period, most of the variance in the velocity is due to the inaccuracy of the ideal encoder model assumption. The Kalman filter/smoother in the next section is shown to pro-

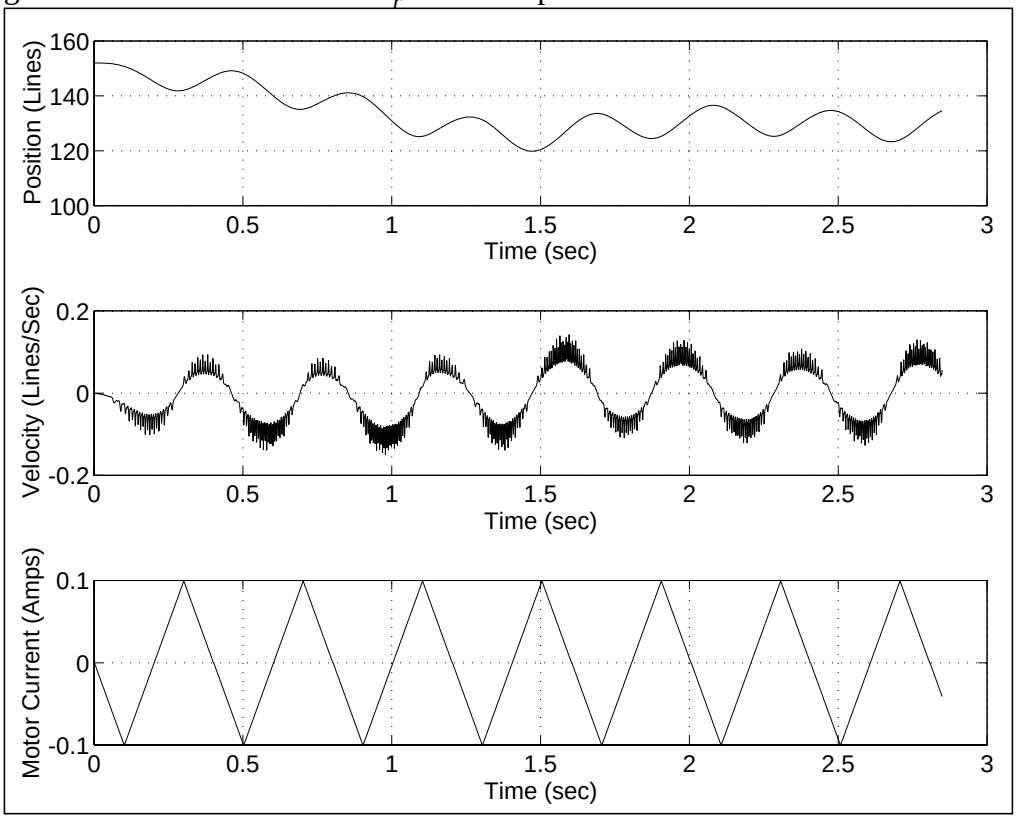

Figure 4.4: Rough State Estimates and Commanded Motor Current

vide a much better estimate of true position and velocity.

4.3 Calibration

The calibration procedure consists of first running the Kalman filter/smoother using the position estimate, p_a, as the sensor input and the commanded motor current, i_m, as the process input. The resulting smoothed position estimate is then used to generate the calibration terms, $\varepsilon(\tau_a)$.

4.3.1 Kalman/Smoother Parameters

The Kalman filter, as formulated in Section 3.3, requires a DGMP model as derived in Section 3.2. This process model incorporates four model parameters which are listed in Table 2 for reference. The K_T and B_F values were determined experimentally while the J value was derived from a CAD model of the system [8]. The sampling interval, T_s, is a function of the control system.

Table 2: Physical Model Parameters

Model Parameter	Assigned Value	Units	Meaning
K_T	0.053	$N \cdot m / A$	Motor Torque Constant
B_F	0.0001	$N \cdot m \cdot \sec$	Viscous Friction
J	0.00092	$N \cdot m \cdot \sec^2$	Joint Inertia
T_s	0.001	sec	Sampling Interval

These parameters can then be used to generate the numeric transition and control-input distribution matrices for the CGMP model using (3-3) and (3-4):

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -0.011 \end{bmatrix} \tag{4-5}$$

and

$$B = \begin{bmatrix} 0 \\ -0.576 \end{bmatrix}. \tag{4-6}$$

The sensor noise variance, V, is bounded by the maximum error in the position estimate, p_a, which assumes the ideal encoder model. The maximum error (i.e., the maximum magnitude of $\varepsilon(\tau_a)$) for the experimental encoders is typically about 0.05 of a line. This knowledge can be used to get a rough estimate of V using

$$V = \left[\frac{2\pi}{N_L}(0.05)\right]^2 = 9.9\times10^{-8}\ \text{Radians}^2 \tag{4-7}$$

and also good estimates of the initial and final state covariances as needed to start the forward and backward Kalman filters respectively:

$$P_0 = P_N = \begin{bmatrix} V & 0 \\ 0 & \sqrt{V} \end{bmatrix} = \begin{bmatrix} 9.9\times10^{-8} & 0 \\ 0 & 3.1\times10^{-4} \end{bmatrix}. \tag{4-8}$$

where N_L is the number of lines on the encoder disk. If the maximum error is not well known, a more crude approximation is acceptable; it only increases the convergence time of the Kalman filter.

For this experiment, a single process noise input was used, feeding into only the second state (velocity). This was done under the assumption that much of the process noise was due to unknown properties of the actuator or noise on the commanded motor current, i_m. This assumption implies that

$$\Gamma_c = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4-9}$$

and that the process noise spectral density, Q, is a scalar quantity. The value of $Q = 0.01$ radians was determined by trial and error.

With the above definitions for the CGMP model $[A, B, \Gamma_c, V, Q, T_s]$, the DGMP model may be derived using (3-20) through (3-23), yielding,

$$\Phi = \begin{bmatrix} 1 & 0.001 \\ 0 & 0.99 \end{bmatrix}, \tag{4-10}$$

$$\Psi = \begin{bmatrix} -2.9\times10^{-5} \\ -5.8\times10^{-2} \end{bmatrix}, \tag{4-11}$$

$$\Gamma = \begin{bmatrix} 5.0\times10^{-7} \\ 1.0\times10^{-3} \end{bmatrix}, \tag{4-12}$$

and

$$W = \begin{bmatrix} 3.3\times10^{-12} & 5.0\times10^{-9} \\ 5.0\times10^{-9} & 1.0\times10^{-5} \end{bmatrix}. \tag{4-13}$$

The forward and backward filters also require initial and final state mean and covariance estimates respectively. The rough state estimate, x_r, derived above in (4-3) may be used to estimate the initial and final state means as follows:

$$x_0 = \begin{bmatrix} x_{r1}(1) \\ \frac{1}{10}\sum_{k=1}^{10} x_{r2}(k) \end{bmatrix} \tag{4-14}$$

and

$$x_N = \begin{bmatrix} x_{r1}(N_{ds}) \\ \frac{1}{10}\sum_{k=N_{ds}-9}^{N_{ds}} x_{r2}(k) \end{bmatrix}. \tag{4-15}$$

where the notation, x_{r1} indicates the first row of x_r.

4.3.2 Smoother results

The forward and backward Kalman filters were run using the parameters given in (4-10) through (4-15) with the formulations given in (3-31) through (3-35) and in (3-36) through (3-40). The forward and backward state estimates from these two filters were combined with the smoother formulation in (3-41) through (3-43).

An important aspect of the Kalman filter is the behavior of the state covariance estimates, $[P_f, P_b, P_s]$ and the Kalman gains $[K_f, K_s]$ over time. For the time-invariant stable system, the covariances and the Kalman gains should converge after a short period of time to steady-state values. Failure to converge is a good indicator that something is

wrong with the formulation or that the system is unstable.

Figure 4.5 shows the state variances and Kalman gains. The variance plots show that

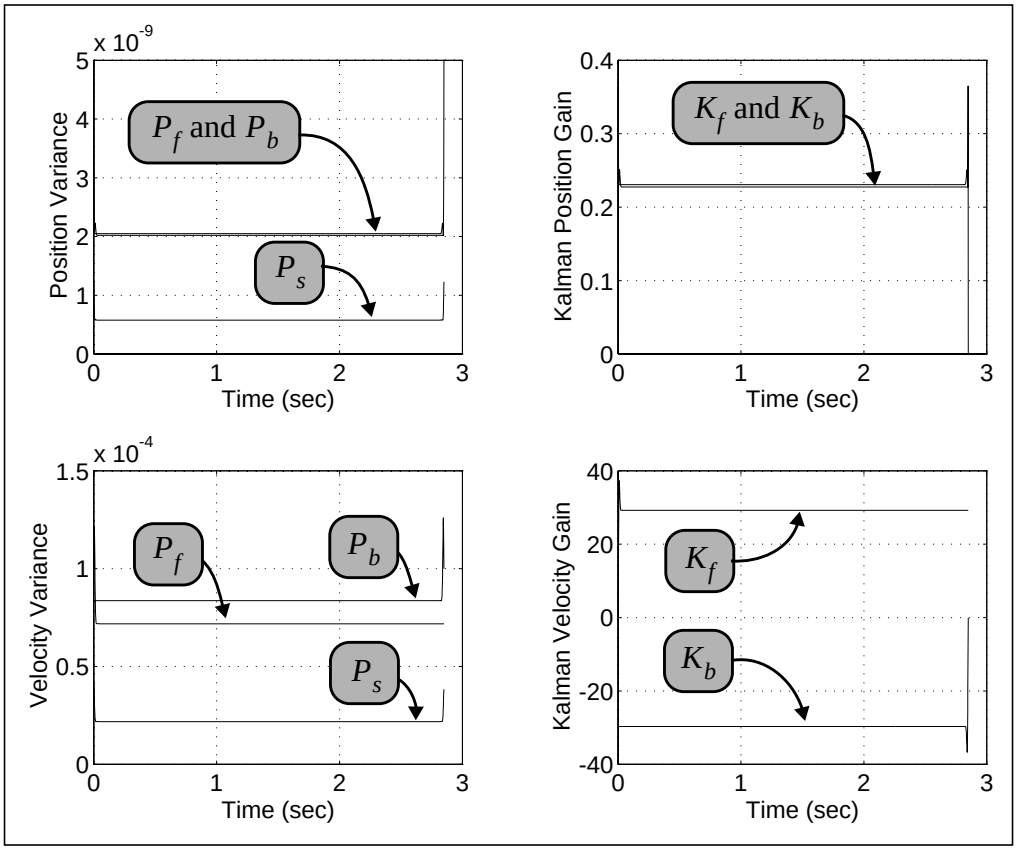

Figure 4.5: State Variance and Kalman Gain plot

the smoothed state variance, P_s is lower than the forward or backward filter variances, P_f and P_b, respectively. This is a graphical indication that the state mean estimate of the smoother will be more accurate than the state mean estimates of either the forward or backward filters alone. The variances and the Kalman gains converge in just a few samples (10 to 20) to the steady-state values.

The smoothed estimate for the second state (e.g., angular velocity) is shown in Figure 4.6. The dots in this figure are the instantaneous velocity estimates from the initial rough state estimate, x_r in (4-3). The line passing through these points is the smoother's estimate of the mean value of this state. Figure 4.7 shows the first-difference of this state (i.e., the angular acceleration) as a function of time (with dots) overlaid with the expected acceleration from the commanded motor current (i.e., $K_T \cdot i_m$). The plot shows that the smoother's

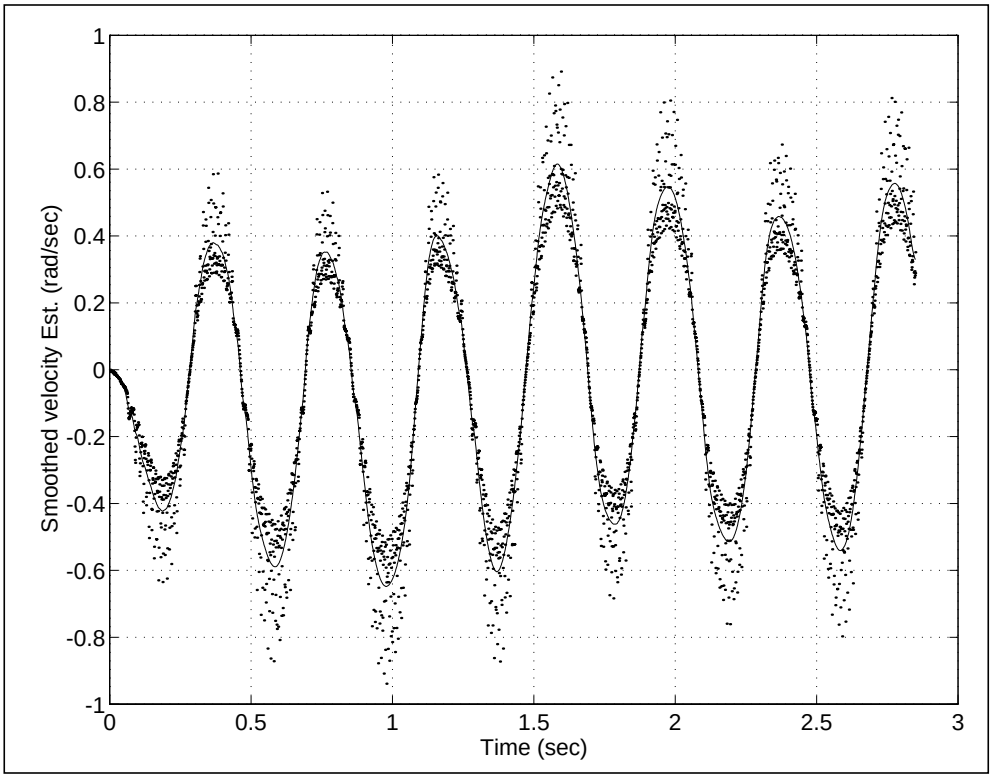

Figure 4.6: Smoothed Velocity Estimate

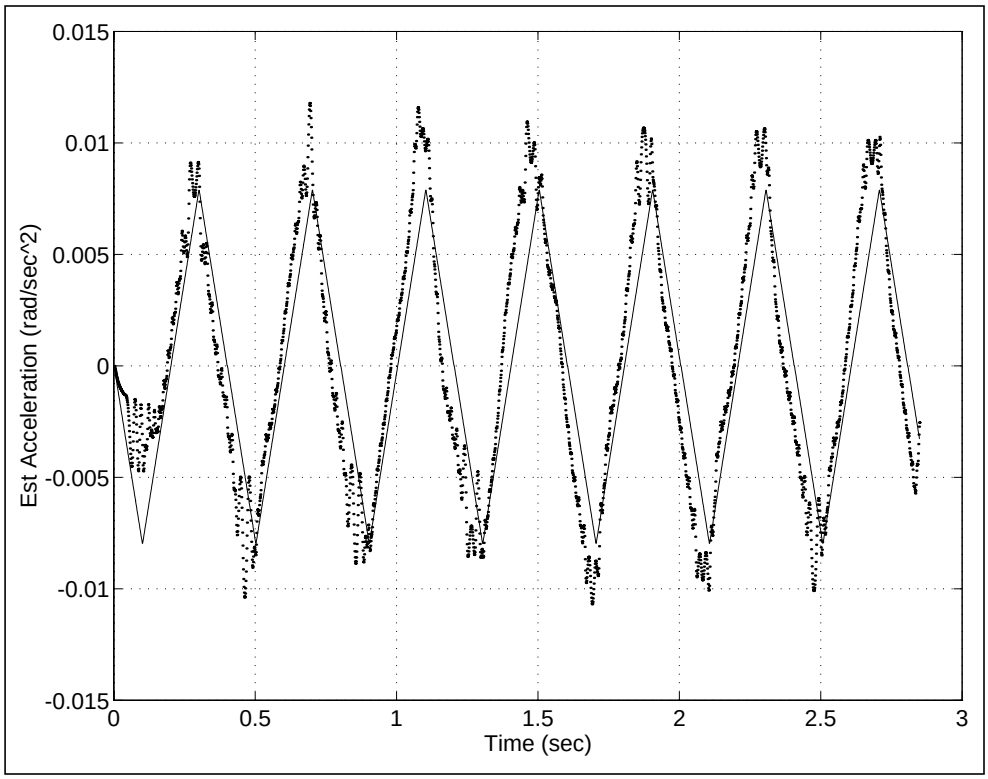

Figure 4.7: Acceleration Estimate

velocity estimate still has some noise in it; however, as a second difference of the position state, this noise is relatively small. The apparent phase lag between the commanded acceleration and the smoothed acceleration is not yet understood.

4.3.3 Calibration Table Results

The smoother in the previous section provides the minimum error smoothed estimate of angular position. By extracting the intra-line position information, τ, from the smoothed position estimate, the difference between $\tau(t)$ and the rough intra-line position estimate, $\tau_a(t)$, may be determined at each sample. This is the calibration term, ε, as shown in (3-45). Figure 4.8 shows $\varepsilon(t)$ vs. t. If $\tau_a(t)$ and $\varepsilon(t)$ are used to form a table, and this table is

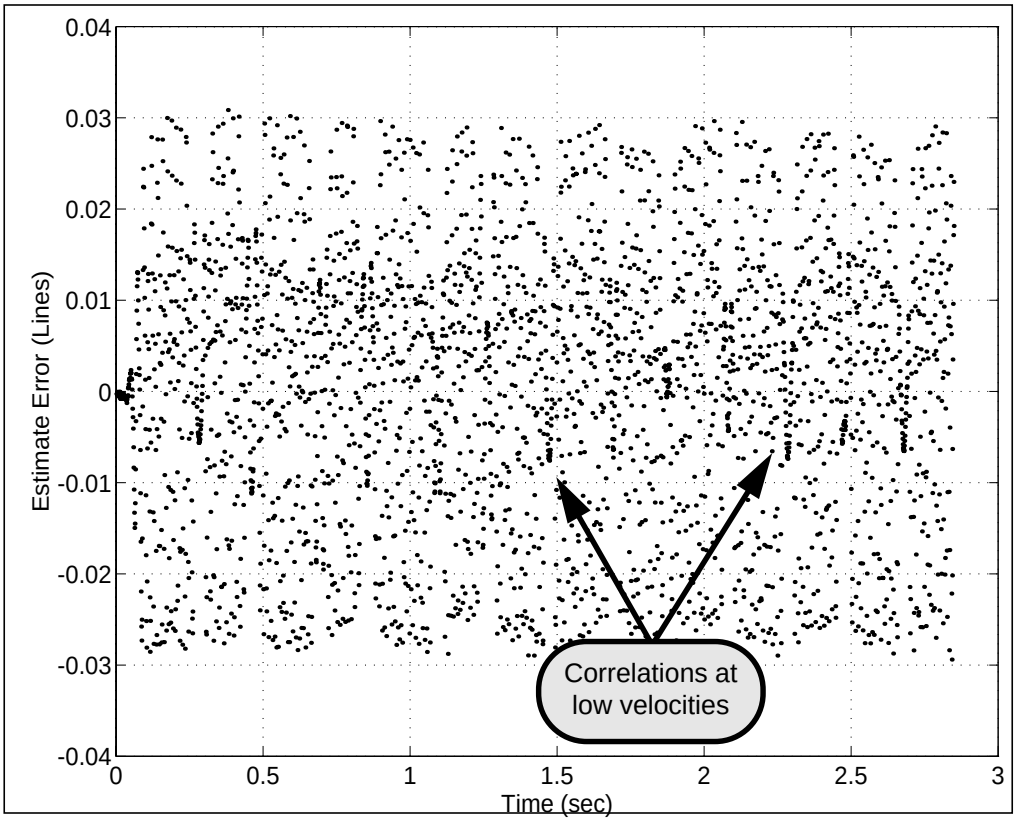

Figure 4.8: Measurement Noise Estimate

then sorted by τ_a, the calibration terms, $\varepsilon(\tau_a)$, in this table may be used to compute the true position τ for any future rough position estimate, τ_a using the relation, as given in (2-1):

$$\tau = \tau_a + \varepsilon(\tau_a). \tag{4-16}$$

It is interesting to note the apparent distribution of $\varepsilon(t)$ in Figure 4.8 above. Since the Kalman filters use τ_a as a position measurement, $\varepsilon(t)$ is the "noise" in the τ_a signal. In Section 3.3.1 it was noted that the Kalman filter assumes that the measurement noise has low correlation between samples. This plot shows that this assumption is reasonably accurate for this signal. The small sections where the plot looks time-correlated occur when the velocity is near zero. These sections must be removed before the data is used to generate a lookup table.

Figure 4.9 shows the contents of the raw sorted table computed from the data shown in

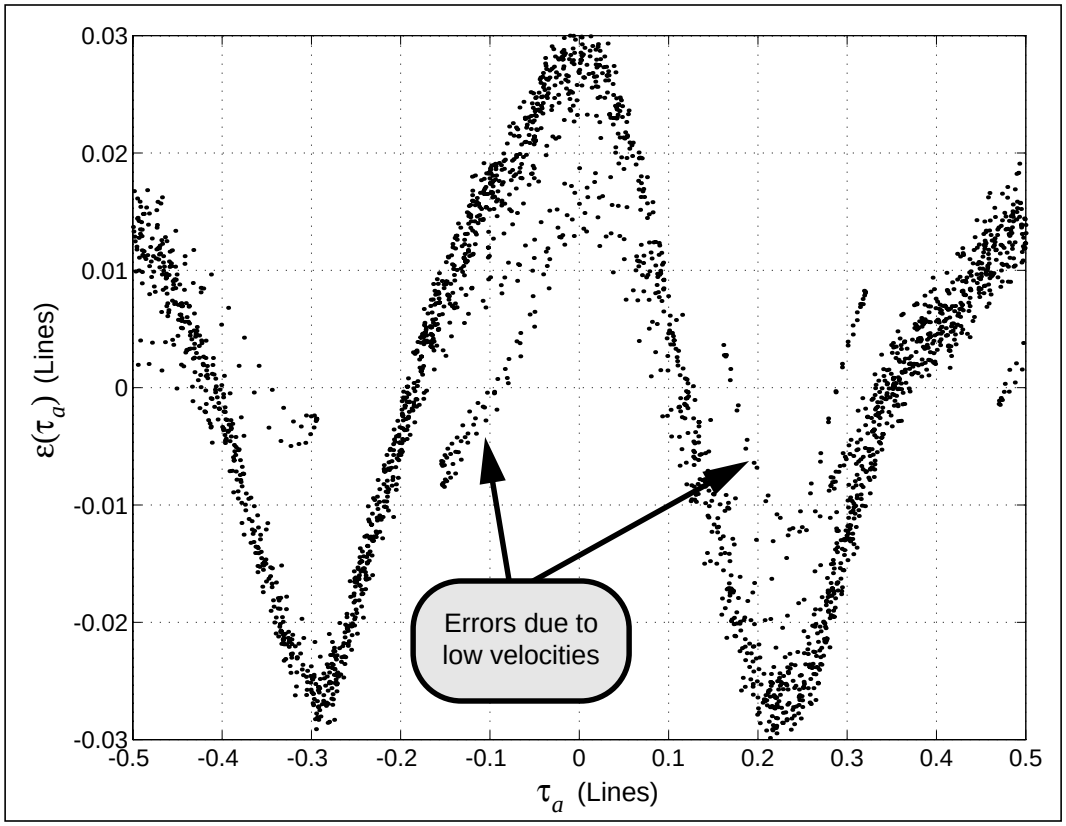

Figure 4.9: Raw Calibration Table

Figure 4.8. Note the general shape of the curve and the highly divergent regions pointed out by the arrows in the figure. These regions occur because the filter estimate becomes inaccurate when the measurement noise (i.e., $\varepsilon(t)$) becomes time-correlated due to low velocities. These data are filtered out of the table using the magnitude of the smoothed estimate of the second state (velocity) as the criteria function. If all points with speed (i.e.,

absolute value of angular velocity) less than 0.1 radians/second are removed from the table, the table then looks like the one shown in Figure 4.10 where the data variance is much low-

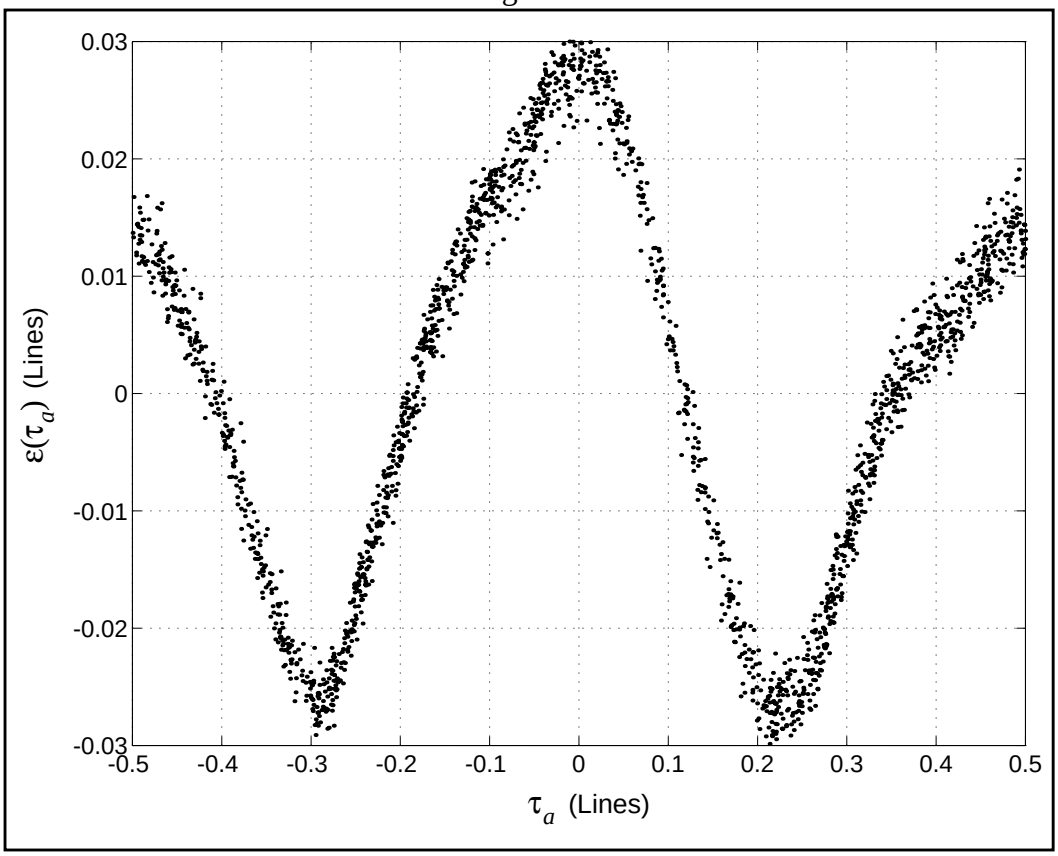

Figure 4.10: Velocity-Filtered Calibration Table

er. The first and last 100 points are also removed from the table to avoid the regions where the filter has not yet converged and thus the state estimates are not as accurate.

However, even the velocity filtered calibration table is still somewhat noisy. One solution is to fit a smoothed curve through the data with the constraint being that the curve be continuous across the $\tau = -0.5$ and $\tau = 0.5$ boundaries since τ_a is a closed contour on the interval -[0.5,0.5). The natural choice for this type of fitting is a Fourier series since its formulation assumes the dataset is periodic. The FFT of the dataset can be used to perform a lowpass filtering operation (basically just truncation of all of the high frequencies). The inverse-FFT can then be made from the filtered frequency-domain data to get a smoothed τ_a-domain table. However, the data in the table is not spaced at equal intervals; this is a problem since the FFT assumes equal intervals between samples. Therefore an equal-inter-

val resampling of the data must first be performed before the FFT is computed.

Figure 4.11 shows the resampled data and the fitted curve. Due to the smoothness of

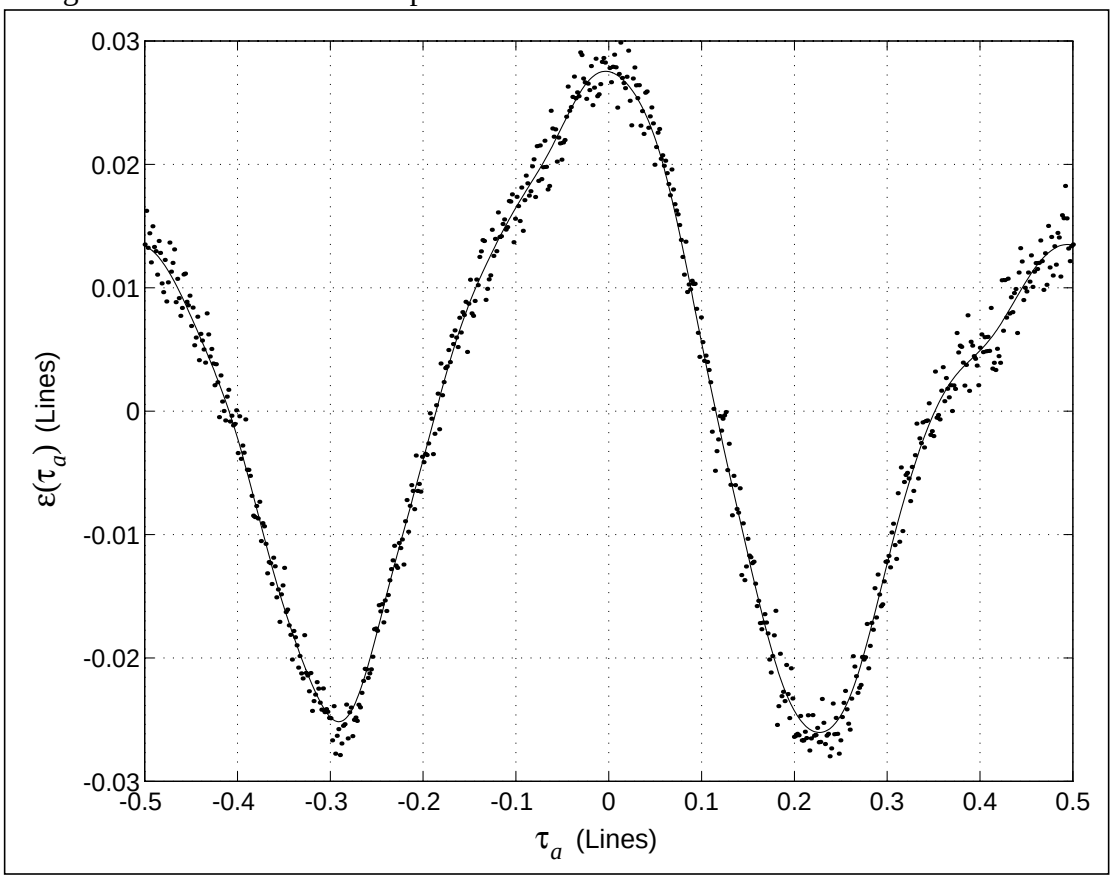

Figure 4.11: Smoothed Resampled Correction Function, $\varepsilon(\tau_a)$

the filtered table, the fit was generated by retaining only the 15 lowest order terms in the Fourier series. These parameters could be used directly as a parametric model of the correction function, $\varepsilon(\tau_a)$. However this parametric approach to the calibration function is somewhat computationally inefficient since the control system would have to compute the series for every position sample. Instead, the values of this series at 600 equidistant points in τ_a on the interval $[-0.5, 0.5)$ are stored as a lookup table. In order to get a corrected estimate using the lookup table, only a linear interpolation is needed between the nearest two τ_a values in the table.

This 600-point table is the final result of the calibration efforts. It can be used to compute a corrected estimate of the true position, τ, using the rough position estimate, τ_a, using the relation,

$$\tau(t) = \tau_a(t) + \varepsilon(\tau_a)\,. \tag{4-17}$$

Linear interpolation into the lookup table is used to determine the actual value for $\varepsilon(\tau_a)$ when τ_a does not fall exactly on one of the table elements.

4.3.4 Algorithm Implementation

The experimental algorithm was implemented in a *MATLAB* function. The full MATLAB implementation can be seen in Appendix B. The function is called with the rough state estimates, x_r, and the motor current commands, i_m, as input vectors and returns a lookup table as a 2×600 array where the first row contains the table index (τ_a) and the second row contains the associated correction factor, $\varepsilon(\tau_a)$.

CHAPTER 5: EXPERIMENTAL RESULTS

The previous chapter showed the derivation of a calibration table using experimental data. The validity of this table should be verified under experimental conditions so that the performance of this table-based correction of the rough position estimate may be evaluated.

5.1 Off-line Comparison

The calibration table is generated from data taken from a physically operational system. One way to verify this calibration is to use the original, rough intra-line position estimates, τ_a, and the calibration terms from the new calibration table, $\varepsilon(\tau_a)$, to compute the corrected position estimates. These corrected position estimates and their derivatives may then be compared to the smoothed state estimates generated by the Kalman filter/smoother during the calibration process.

The top and bottom graphs in Figure 5.1 show the difference between the smoothed position estimate and the rough position estimate, τ_a, before and after it has been corrected by the calibration lookup table. The periodic large spikes in the corrected position error are due to the position estimate errors in the Kalman filter induced by measurement noise correlation at low velocities. The plot shows that the rough position estimate varies by ± 0.03 of a line from the correct position (assuming that the smoothed position estimate is really correct). The corrected position estimate varies by ± 0.004 of a line, a factor of 7.5 increase in precision over the rough position estimate. Thus, for the experimental setup, the corrected position estimate improves the digital encoder precision of ± 0.25 line by a factor of 62.5(~6.0 bits compared to the rough position estimate's improvement by a factor of only 8.3(~3.0 bits).

Another method of comparing the measurement is by examining the velocity as computed by the first difference of position. Figure 5.2 shows the velocity estimate from the Kalman filter/smoother, overlaid by the first-difference of the rough position estimates and the first-difference of the corrected position estimates. The plot shows that the velocity estimates based on the corrected rough intra-line position, (i.e., $\tau_a + \varepsilon(\tau_a)$) are much closer to the underlying Kalman filter/smoother velocity estimate. These velocity estimates provide an important qualitative measure of improvement because typical control algorithms such

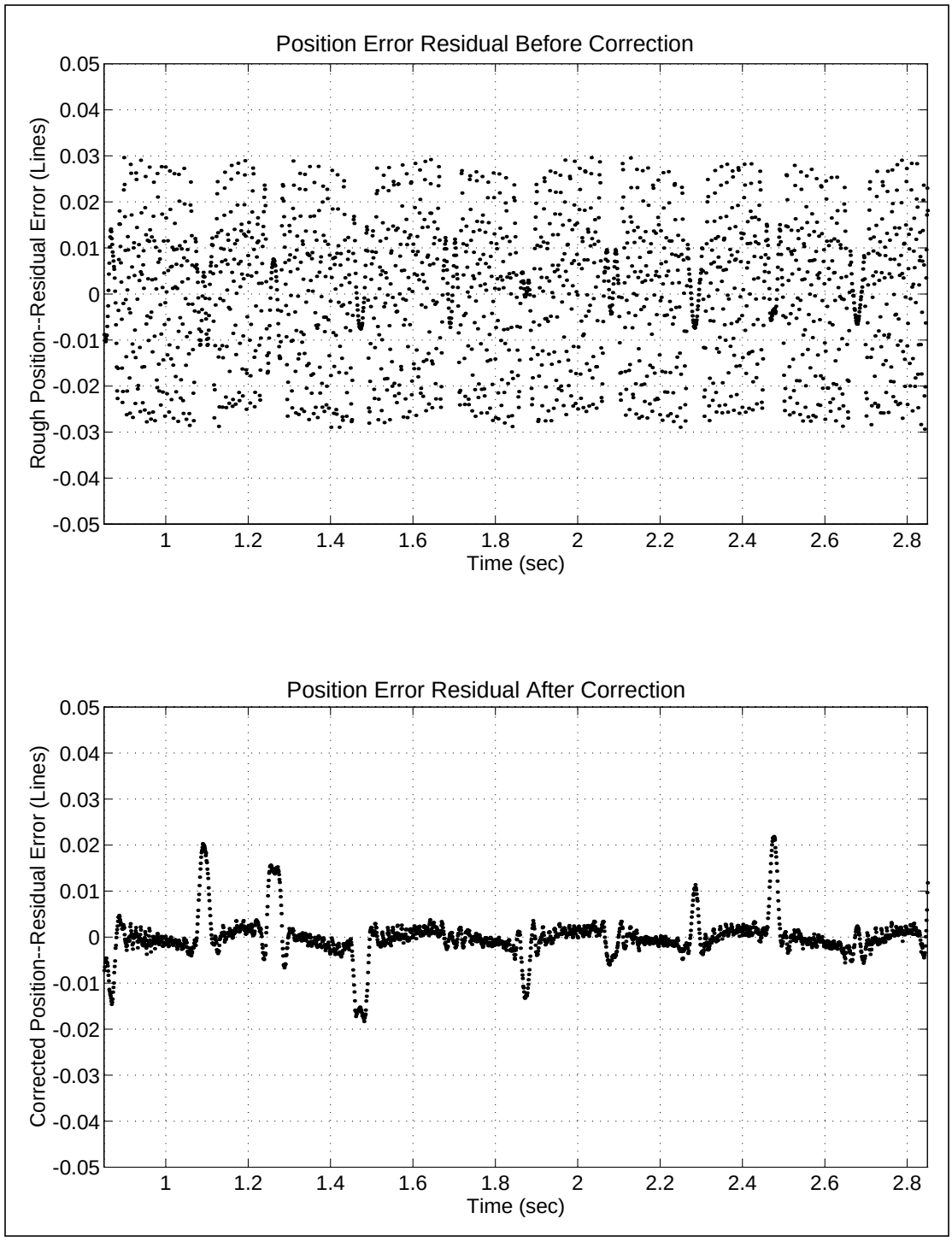

Figure 5.1: Position Error Residuals Before and After Correction

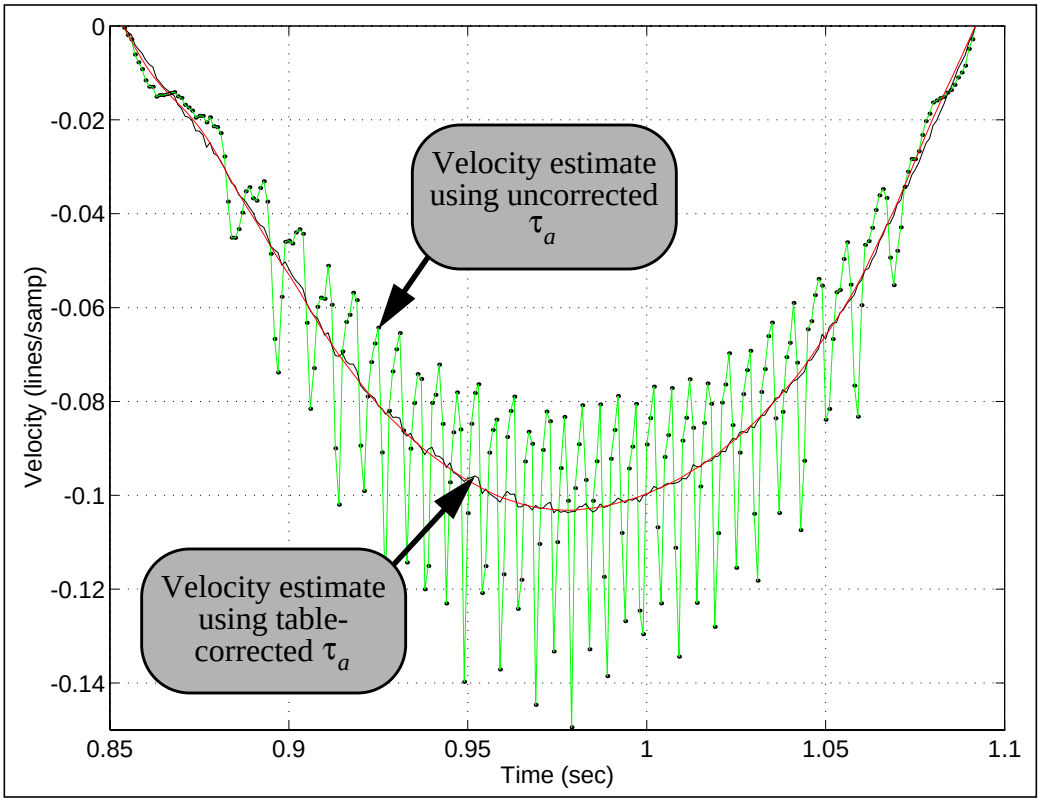

Figure 5.2: Velocity Estimate Comparison

as the PID algorithm use the velocity (as computed by the first difference of position) as a control input. If the velocity estimate is noisy, it will induce significant disturbances to the control system.

5.2 Physical Verification

The previously described calibration method may be verified by direct physical measurements. If the encoder signals are sampled at known intra-line positions τ, then τ_a may be computed from the sampled signals using

$$\tau_a = \frac{\operatorname{atan2}(b, a)}{2\pi} \tag{5-1}$$

and the calibration terms may be computed using

$$\varepsilon(\tau_a) = \tau - \tau_a . \tag{5-2}$$

The challenge with this approach to calibration is that high line-count encoders will require very high-precision angular measurements to obtain accurate estimates of τ_a. These mea-

surements may be difficult and/or expensive to obtain.

5.2.1 Physical Calibration Setup

The physical calibration setup for the experimental system is shown in Figure 5-3. A

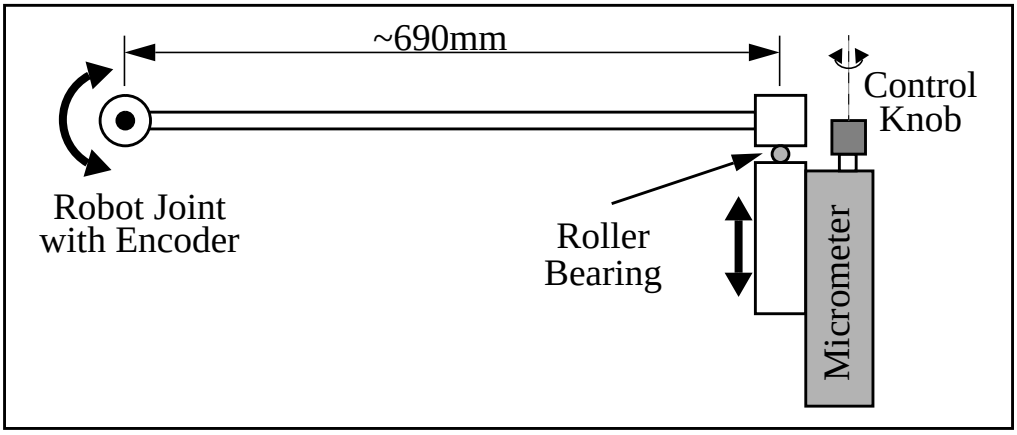

Figure 5.3: Physical Calibration Setup

linear motion stage is used to control angular displacement of the robot joint. Since the 12-bit A/D converters are theoretically capable of resolving down to $\delta\tau = 1/2^{12} = 0.0002$ of a line, the 152mm link length joint would require linear position precisions of

$$\begin{aligned} \Delta y &= r\sin\left(\frac{2\pi}{N_l}\cdot\Delta\tau\right) \\ &= \left(152mm\cdot\sin\left[\frac{2\pi\,(\text{Radians})}{1000\,(\text{Lines})}\cdot 0.0002\,(\text{Lines})\right]\right) \\ &= 0.19\mu m \end{aligned} \tag{5-3}$$

at the end of the robot link. A steel rod is used to extend the radius of rotation for this joint to about 690mm. This decreases the linear position precision requirements at the end of the rod to $0.87\mu m$. At this radius, a linear motion range of

$$y = r\sin\theta = 690\text{mm}\cdot\sin\left[\frac{2\pi\,(\text{Radians})}{1000\,(\text{Lines})}\right] = 4.34\text{mm} \tag{5-4}$$

is needed to scan one encoder line. Only a laser interferometer has the linear motion precision necessary to resolve the $190nm$ increments implied by (5-3). Since no interferometer was immediately available, a $10\mu m$-precision linear micrometer stage was used at the expense of some lost calibration precision. Even at this resolution, special care had to be taken to allow the motions to occur at a constant radius of rotation. Since the distance measure-

ments y are known accurately relative to each other but not to some absolute coordinate system, the true value of τ may be represented by

$$\tau = \frac{\text{atan}\,(y/r)}{2\pi} + k \tag{5-5}$$

where k is some unknown position offset. The computation of $\tau(y)$ may be approximated by the linear relation,

$$\tau = y/(2\pi r) + k \tag{5-6}$$

since the maximum error induced by this assumption is

$$\text{error}(\tau) = \frac{1}{2\pi} \cdot \text{atan}\left(\frac{2.15mm}{690mm}\right) - \frac{2.15mm}{690mm} = 1.3\times10^{-8}\ \text{Lines} \tag{5-7}$$

which is negligible. The effects of the offset, k, will be seen in the derivation of the physical calibration table in Section 5.2.3.

5.2.2 Physical Calibration Data

The physical calibration involved taking 500 samples of the encoder (a, b) signals separated by linear position increments of $10\mu m$ at the end of the 690mm joint extension. The (a, b) values at each sample were computed from the mean of 100 samples of (a_d, b_d) taken at 1-millisecond intervals. The resulting signals are shown in Figure 5.4. The 500 samples actually covered more than one encoder line. By examining the data, the cycle interval was determined to be 432 samples. This implies that one (a, b) cycle occurred in $4.32mm$ of displacement. Since the trigonometric relation, $y = r\sin(\theta)$ holds, this knowledge may be used to compute the true radius of rotation, r, using

$$r = \frac{4.32mm \cdot 2\pi}{\sin\left(\frac{2\pi\,(\text{Radians})}{1000\,(\text{Lines})}\right)} = 687.6mm. \tag{5-8}$$

5.2.3 Calibration Table from Physical Calibration

The correction terms, $\varepsilon(\tau_a)$ may be found using

$$\varepsilon(\tau_a) = \tau(y) - \tau_a(a, b). \tag{5-9}$$

Substituting (5-1) and (5-6) into (5-9) yields

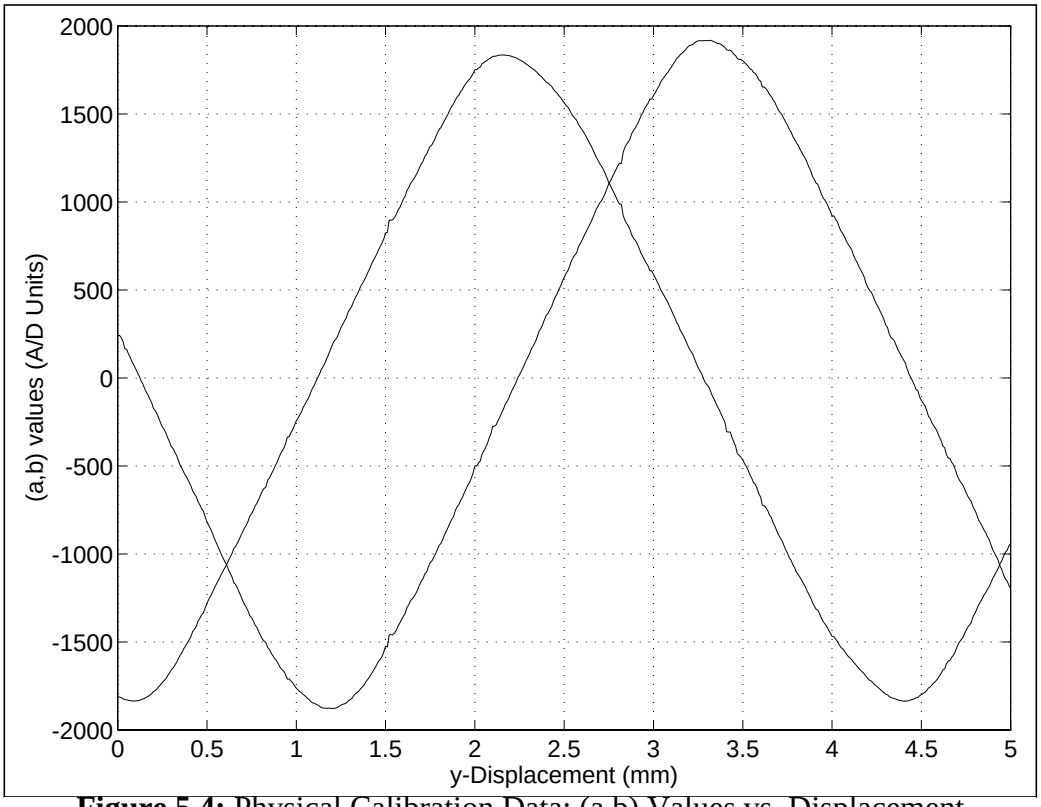

Figure 5.4: Physical Calibration Data: (a,b) Values vs. Displacement

$$\varepsilon(\tau_a) = \frac{y}{2\pi r} + k - \frac{\operatorname{atan2}(b, a)}{2\pi}. \qquad (5\text{-}10)$$

The calibration terms, $\varepsilon(\tau_a)$, may be sorted by τ_a to form a physical calibration table. Figure 5.5 shows the calibration table generated by this physical calibration method. Note the constant, k, in (5-10) causes an unknown offset constant offset in the physical calibration table. This offset must be determined before the physical calibration table can be compared to the calibration table generated by the Kalman filter/smoother.

5.2.4 Comparison of Physical Calibration to Kalman Calibration

Figure 5.6 shows the results of this calibration table overlaid with the calibration table from the Kalman filter approach. The plot shows that the two calibration methods have only slight differences in phase and amplitude as a function of τ_a. The worst-case error (residual) in $\varepsilon(\tau_a)$ is about 0.005 lines. If the physical calibration is assumed to be correct, then the 0.005 residual is the worst-case correction error from the Kalman calibration method.

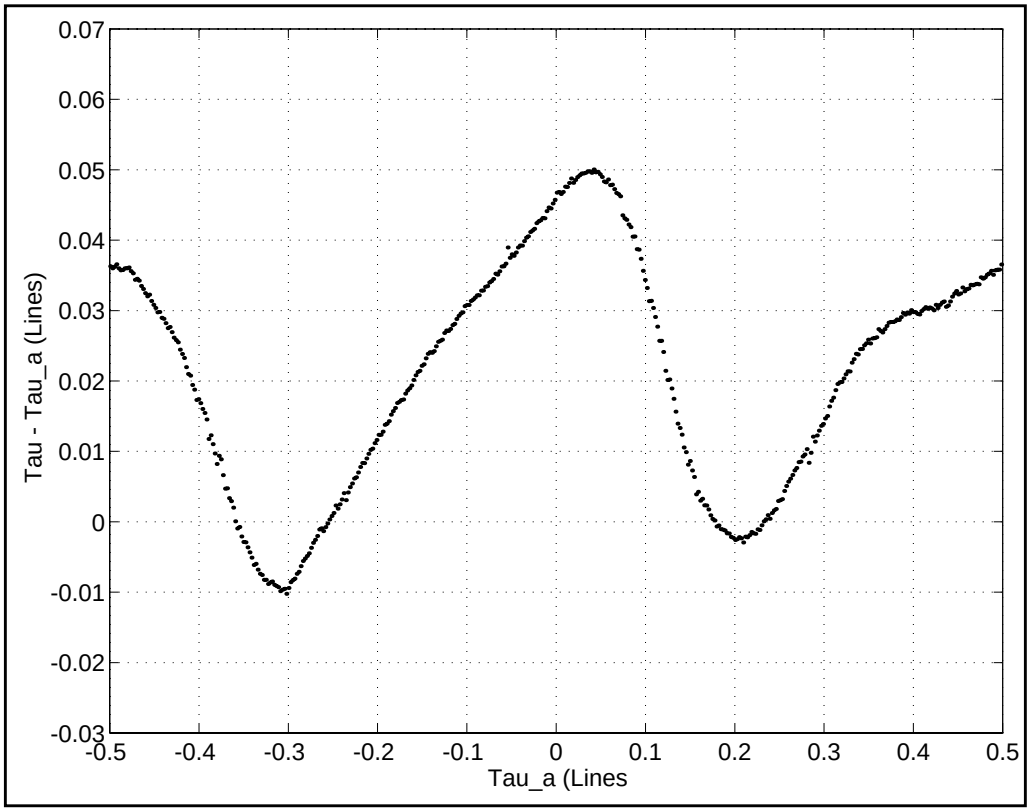

Figure 5.5: Calibration Table from Physical Calibration

This corresponds to a resolution of 1/200th of an encoder line, a factor of 50 improvement over the 0.25-line digital resolution.

The Kalman calibration and physical calibrations leading to the plots in Figure 5.6 were performed again after disassembly and reassembly of the encoder. Figure 5.7 shows this new calibration comparison. Note that while the shape of the calibration table is very different (e.g., it has four peaks instead of two), the Kalman calibration and the physical calibration are still in very close agreement with the worst case residual error between the two methods of about 0.007 lines.

5.2.5 Relative Precision of Physical and Kalman Calibration

In Section 5.2.1, it was shown that the necessary linear positioning resolution to resolve 0.0002 of a line was $0.19\mu m$. Since the micrometers have $10\mu m$ resolution, $\pm 5\mu m$ precision is assumed. The corresponding intra-line precision may be solved using (5-3) rewritten in terms of $\Delta\tau$:

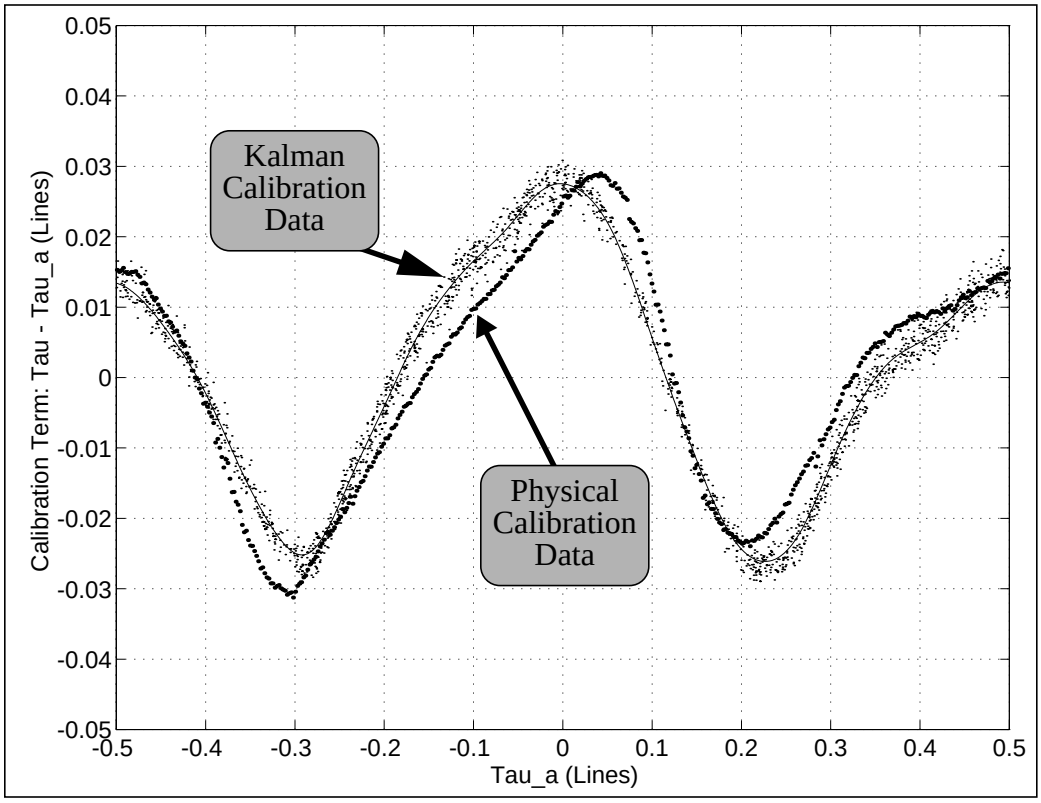

Figure 5.6: Comparison of Kalman Calibration with Physical Calibration

$$\begin{aligned} \Delta\tau &= \frac{N_L}{2\pi} \cdot \operatorname{asin}\left(\frac{\Delta y}{r}\right) \\ &= \frac{1000\,(\text{Lines})}{2\pi\,(\text{Radians})} \cdot \operatorname{asin}\left(\frac{\pm 0.005mm}{687.6mm}\right) \cdot \\ &= \pm 0.0012\,(\text{Lines}) \end{aligned} \tag{5-11}$$

Therefore, the precision of the physically measured intra-line position is only slightly smaller than the worst-case residual error seen in Figure 5.6. This implies that a more precise position measurement scheme is needed to determine the accuracy of the Kalman calibration technique.

One qualitative measure of a calibration table accuracy is the smoothness of the velocity estimate of a moving system as computed from first-difference of the corrected position estimates. Figure 5.2 shows this velocity estimate as extracted from position estimates that have been corrected using the Kalman calibration table. Figure 5.8 shows the velocity estimate for the same dataset using a calibration table derived from the physical calibration.

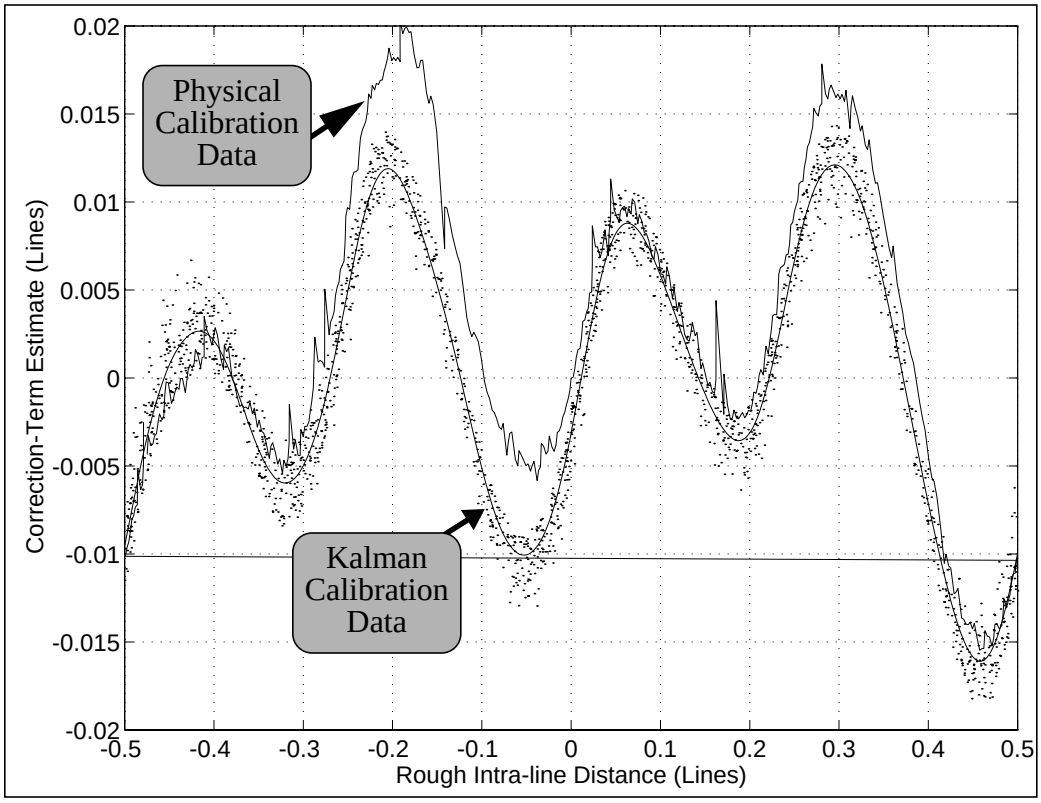

Figure 5.7: Calibration Comparison after Encoder Reassembly

Note that the velocity estimate in this case is much less smooth than the corresponding velocity computed using the Kalman calibration table. This implies that the Kalman calibration table is more accurate than the calibration table derived from the physical measurements.

5.3 Physical Model Parameter Sensitivity Analysis

One disadvantage of the Kalman calibration method is that it needs a dynamic model of the physical system. Obtaining correct model parameters is not always easy, even for the relatively simple second-order model used in the calibration experiments. Therefore, it is particular interest to see how the accuracy of the Kalman calibration table decreases with modeling errors.

The second-order model has three physical parameters: the joint inertia, J, the viscous friction, B_F, and the motor torque constant, K_T. These three parameters were varied individually to observe the effects of these changes on the accuracy of the calibration table.

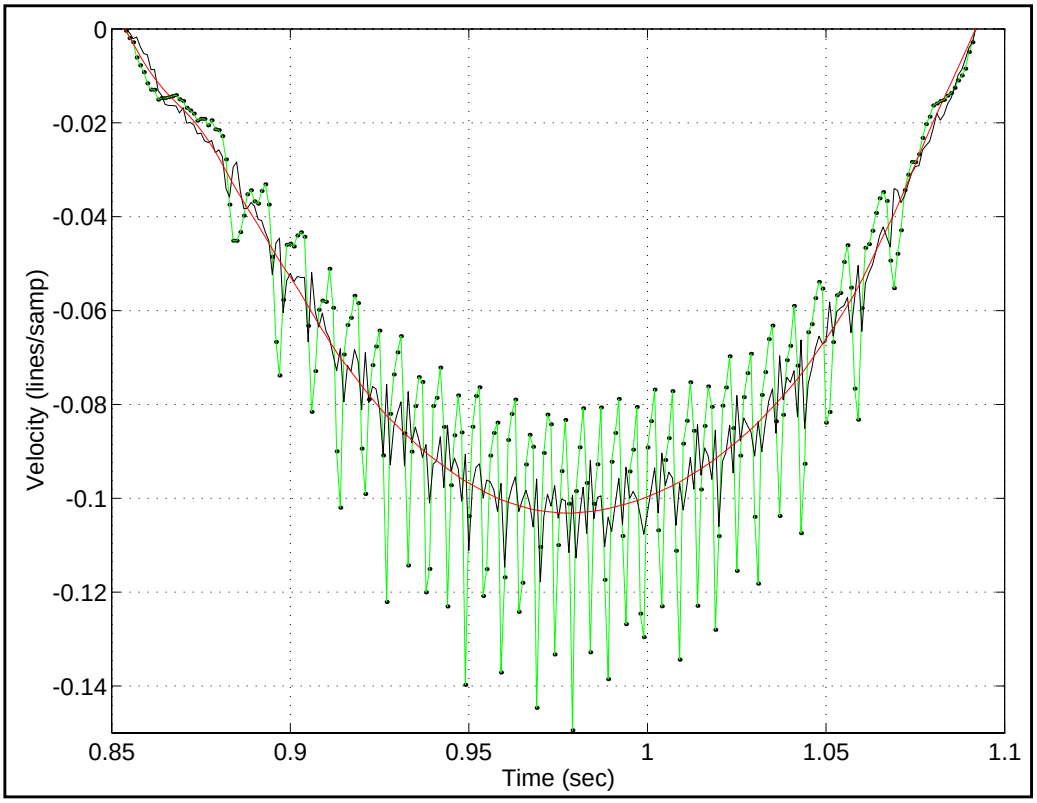

Figure 5.8: Velocity Estimate from Physical Calibration Table

Figure 5.9, Figure 5.10, and Figure 5.11 show the changes in the nominal calibration table (i.e., the table derived from the correct physical parameters) for a variety of values of J, B_F, and K_T respectively. These three plots show that the error in the calibration table induced by any one these physical model parameter errors is less than 0.01 lines. This is still a factor of 3 better than the rough position estimate (see Figure 4.8 on page 52).

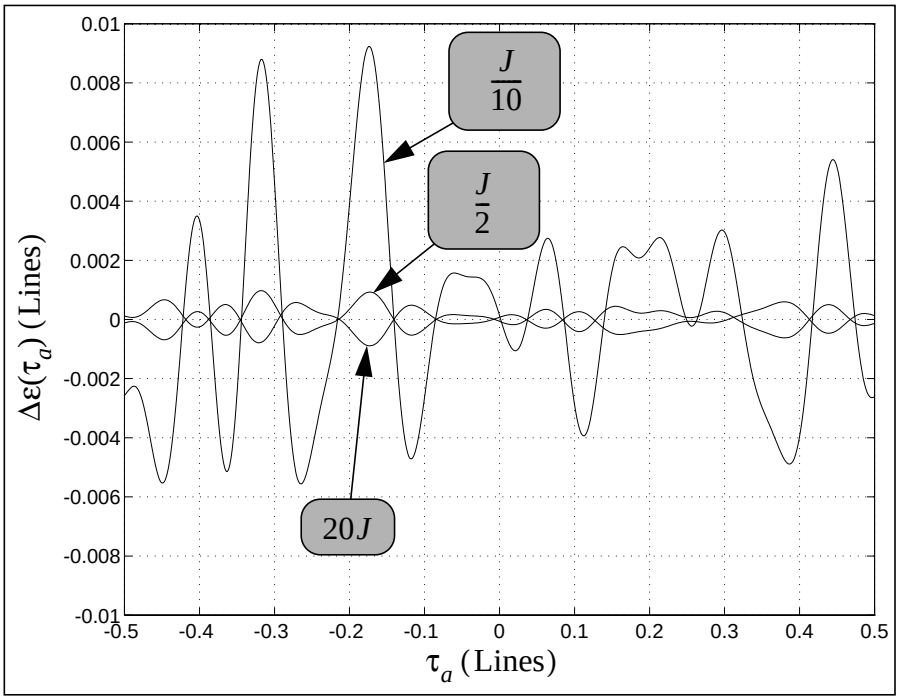

Figure 5.9: Calibration Table Sensitivity to Changes in Inertia (J)

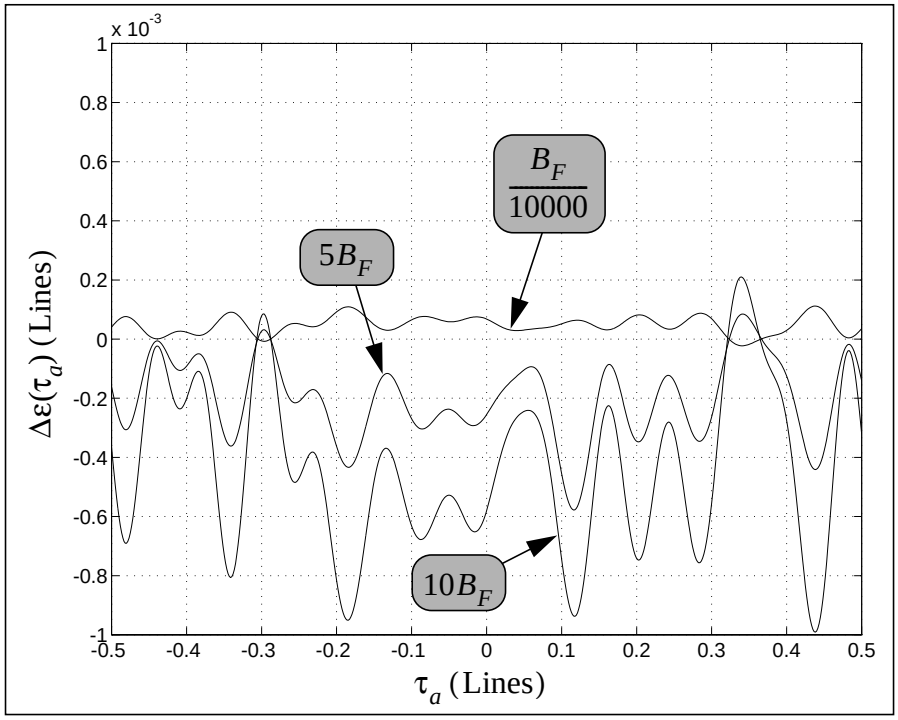

Figure 5.10: Calibration Table Sensitivity to Changes in Damping (B_F)

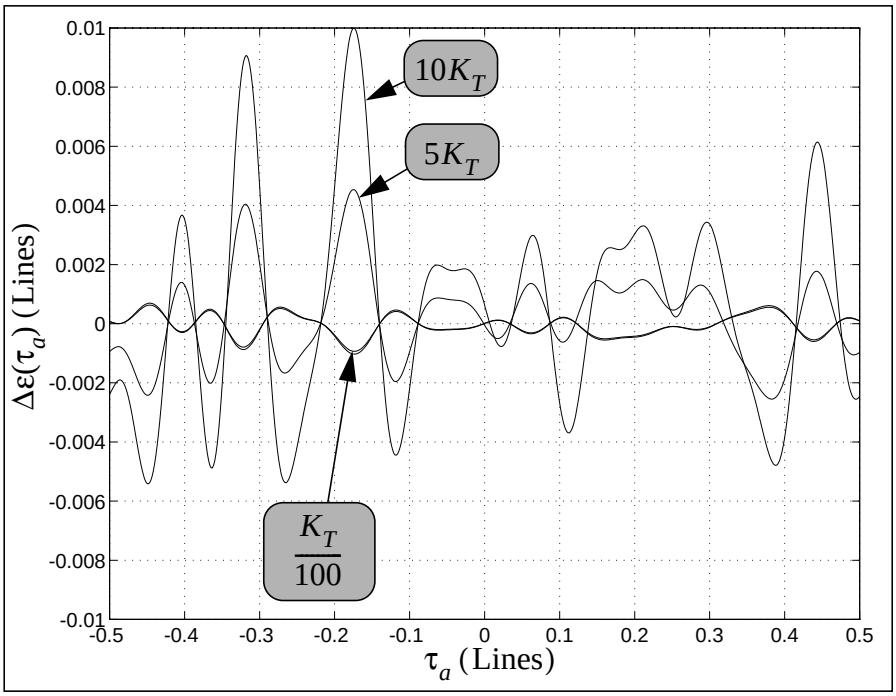

Figure 5.11: Calibration Table Sensitivity to Changes in Torque Constant (K_T)

CHAPTER 6: CONCLUSIONS

6.1 Summary of work

An off-line method for generating a calibration table for an analog position encoder has been presented. The method employs a Kalman filter/smoother to estimate true intra-line position, τ, of the encoder based on the rough position estimate, τ_a, and a dynamic model of the encoder/actuator system including a deterministic commanded current input. The difference between the Kalman filter/smoother estimate of position and the rough position estimate is used to generate the calibration table of correction terms, $\varepsilon(\tau_a)$.

A mathematical derivation of the variance of the estimated intra-line position as a function of the encoder signal variances and the output values was derived in Chapter 2. This derivation showed that, for the experimental system, the measurement noise from the encoders imposes a resolution limitation of about 0.001 of a line.

An experimental implementation of this calibration method was done using a 1000-line analog quadrature rotary encoder. A physical calibration experiment using a $10\mu m$-resolution micrometer indicates that the Kalman calibration method agrees with the physical calibration to at least 0.005-line intra-line interpolation precision. Qualitative comparisons between the Kalman calibration results and the physical calibration (Section 5.2.5) indicate that the Kalman calibration method is more accurate than the physical calibration; however, a more precise physical calibration using a laser interferometer is needed to verify this claim quantitatively.

6.2 Advantages of the Kalman Calibration Method

The Kalman calibration method allows an encoder to be calibrated on an installed system without the need for external precision calibration hardware such as micrometers or laser interferometers. This is an important feature for many encoders which suffer significant changes in their calibration tables after removal and re-installation on a given system. Additionally, it is often not feasible to make a physical calibration using external devices due to space and time constraints.

Unlike the parametric calibration method in [2] (which assumes that the encoder outputs functions $a(\tau)$ and $b(\tau)$ are sinusoidal), the Kalman calibration method is applicable

to a wide variety of encoder output functions which may not fit a parametric model. Both the constant velocity calibration method [3] and the Kalman calibration method generate a calibration table which can be used to estimate true intra-line position, τ, from an ideal intra-line position, τ_a, using the relation,

$$\tau = \tau_a + \varepsilon(\tau_a) \qquad \text{where } \tau_a = \frac{\operatorname{atan2}(b, a)}{2\pi} \tag{5-12}$$

and $\varepsilon(\tau_a)$ is the correction term from the calibration table for a given τ_a. However, unlike the constant velocity method, this calibration method does not require constant velocity motion during the calibration process (which is difficult to obtain). Instead, the method works best with *changing* velocities, which is much easier to obtain. In addition, the constant velocity calibration method requires very high-speed (for example, 100-kHz) sampling of the encoder signals, while the Kalman calibration method may be used with the normal operational sampling rate of the embedded control system (for example, 1-kHz).

6.3 Future Directions

6.3.1 Improved Verification of Calibration Precision

The current experimental results indicate that the Kalman calibration method yields intra-line position estimates that are at least as precise as the physical calibration setup used to verify its precision. A more accurate physical calibration method is needed using a laser interferometer to verify the accuracy of this Kalman calibration method.

The Kalman calibration method, as currently derived, implicitly assumes that the encoder output signals, $a(\tau)$ and $b(\tau)$ are identical for all adjacent line pairs on the encoder disk. A laser interferometer could be used to get a better estimate of these differences. If significant, the Kalman calibration method could be reformulated to generate separate calibration tables for each adjacent line-pair on the encoder disk, although this may substantially increase the storage requirements for the calibration table. More extensive experimentation with a wider range of analog quadrature position encoders is needed to gain more knowledge of how prevalent this adjacent line-pair variance problem may be.

6.3.2 Improved Noise Modeling

The bottom plot in Figure 5.1 shows a slight periodicity in the residual error of the corrected rough position estimates over time. This indicates that the Kalman filter/smoother estimates may be improved if a more accurate model of the measurement noise is known. Correlations of this noise are known to become significant, as shown in Figure 4.8, at lower magnitude velocities. It may be that even at higher velocities, the slight coloration of this noise is affecting the filter performance. Gelb, et.al. [12], showed that the DGMP model can be augmented to incorporate colored measurement noise by adding states which act as first-order systems forced by white noise inputs. This type of augmentation may improve the Kalman filter/smoother estimation performance.

6.3.3 Embedded System Implementation

The current Kalman calibration algorithm is implemented in MATLAB. The algorithm can be reformulated in a high-level programming language such as *C* and implemented on an embedded control system. Calibration could then be done at any time with several seconds of computational time on more powerful control system architectures such as the DSP-based controller used in the experimental system discussed in Chapter 4. It may be possible to formulate the calibration process to run as a background process on the controller which updates the calibration table based on observing the sensor signals and commanded outputs of the real-time control system during operations.

REFERENCES

[1] Hewlett Packard Components Group, "Quadrature Decoder/Counter Interface IC," Optoelectronics Designer's Catalog, Hewlett Packard Corp., pp. 1-61 through 1-76, 1992.

[2] P.H. Marbot, "Mini Direct Drive Robot for Biomedical Applications," MSEE Master's Thesis, Biorobotics Laboratory, University of Washington, Seattle, 1992.

[3] N. Hagiwara, Y. Suzuki, H. Murase, "A Method of Improving the Resolution and Accuracy of Rotary Encoders Using a Code Compensation Technique," IEEE Transactions on Instrumentation and Measurement, Vol. 41, No. 1, pp. 98-101, February, 1992.

[4] J.P. Karidis, et.al., "The Hummingbird Minipositioner—Providing Three-Axis Motion at 50 G's With Low Reactions," Proceedings of the IEEE International Conference on Robotics and Automation, pp. 685-692, Nice, France, May, 1992.

[5] B.A. Sawyer, US Patent No3,457,482, July 1969.

[6] J. Ish-Shalom, "Sawyer Sensor for Planar Motion Systems," Proceedings of the IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2652-2658, San Diego, May, 1994.

[7] B. Hannaford, P.H. Marbot, M. Moreyra, S. Venema, "A 5-Axis Mini Direct Drive Robot for Time Delayed Teleoperation," To be published in Proceedings of IROS, Munich, September, 1994.

[8] M.R. Moreyra, "Design of a Five Degree of Freedom Direct Drive Mini-robot using Disk Drive Actuators," MSME Master's Thesis, Biorobotics Laboratory, University of Washington, Seattle, 1994.

[9] I. MacDuff, S. Venema, B. Hannaford, "The Anthroform Neural Controller: A System for Detailed Emulation of Neural Circuits," Proceedings of IEEE International Conference on System's, Man, and Cybernetics, Vol. 1, pp. 117-122, Chicago, 1992.

[10] A. Leon-Garcia, "Probability and Random Processes for Electrical Engineering," Addison-Wesley, pp. 233-236, 1989.

[11] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering (ASME), 82D, pp. 35-45, March, 1960.

[12] A. Gelb, et.al., "Applied Optimal Estimation," Analytic Sciences Corporation, M.I.T. Press, pp. 182-203, 1988.

[13] C.F. Van Loan, “Computing Integrals Involving the Matrix Exponential,” IEEE Transactions on Automatic Control, Vol. AC-23, No. 3, pp. 395-404, June 1978.

[14] A.E. Bryson, Jr., “Dynamic Optimization with Uncertainty,” Pre-publication draft, Stanford University, March, 1993.

[15] A.E. Bryson and M. Frazier, “Smoothing for Linear and Nonlinear Dynamic Systems,” Technical Report, ASD-TDR-63-119, pp. 496-503, Wright Patterson Air Force Base, Ohio, September, 1962.

APPENDIX A: VAN-LOAN ALGORITHM REVIEW

Van Loan's algorithm defines a matrix, $\hat{C}$, whose elements are several sub-matrices:

$$\hat{C} \triangleq \begin{bmatrix} -\hat{A}^T & I & 0 & 0 \\ 0 & -\hat{A}^T & \hat{D} & 0 \\ 0 & 0 & \hat{A} & \hat{B} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{A-1}$$

where Δ is the interval of integration, $[\hat{A}, \hat{B}, \hat{D}]$ are real matrices of dimensions $[n \times n, n \times p, n \times n]$ respectively, and $\hat{D}$ is symmetric positive semidefinite. The matrix, $\hat{S}$, is defined as the matrix exponential of $\hat{C}$ and is used to compute a new set of sub-matrices:

$$\hat{S} \triangleq \exp(\hat{C}t) = \begin{bmatrix} \hat{F}_1 & \hat{G}_1 & \hat{H}_1 & \hat{K}_1 \\ 0 & \hat{F}_2 & \hat{G}_2 & \hat{H}_2 \\ 0 & 0 & \hat{F}_3 & G_3 \\ 0 & 0 & 0 & \hat{F}_4 \end{bmatrix}. \tag{A-2}$$

Using these relational definitions for $\hat{C}$ and $\hat{S}$, Van Loan showed that the following relations hold true for the sub-matrices of $\hat{C}$ and $\hat{S}$ as defined above:

$$\begin{aligned} H(\Delta) &= \int_0^\Delta \exp(\hat{A}s)\,\hat{B}\,ds &&= \hat{G}_3(\Delta) \\ Q(\Delta) &= \int_0^\Delta \exp\left(\hat{A}^T s\right)\hat{D}\exp(\hat{A}s)\,ds &&= \hat{F}_3(\Delta)^T\hat{G}_2(\Delta) \\ M(\Delta) &= \int_0^\Delta \exp\left(\hat{A}^T s\right)\hat{D}H(s)\,ds &&= \hat{F}_3(\Delta)^T\hat{H}_2(\Delta) \\ W(\Delta) &= \int_0^\Delta H(s)^T\hat{D}H(s)\,ds &&= \left[\hat{B}\hat{F}_3(\Delta)^T\hat{K}_1(\Delta)\right] + \left[\hat{B}^T F_3(\Delta)^T\hat{K}_1(\Delta)\right]^T \end{aligned} \tag{A-3}$$

By comparing (3-15) through (3-17) to (A-3), the first integral in (A-3) is identical in

form to the integrals needed to compute (3-15) and (3-16), while the second integral in (A-3) is identical in form to (3-17). In addition, it can be shown [14] that the relation,

$$\exp(\hat{A}t) = \hat{F}_3(\Delta) \tag{A-4}$$

holds true, allowing the computation of (3-14) as well. Therefore the following assignments are made to compute these integrals:

$$\hat{A} = A^T, \tag{A-5}$$

$$\hat{B} = I, \tag{A-6}$$

$$\hat{D} = BQB_T, \tag{A-7}$$

and

$$T_s = \Delta \tag{A-8}$$

which allows $\hat{S}$ to be computed as follows:

$$\hat{S} = \exp\left(\begin{bmatrix} -A^T & I & 0 & 0 \\ 0 & -A^T & \Gamma_c Q \Gamma_c{}^T & 0 \\ 0 & 0 & A & I \\ 0 & 0 & 0 & 0 \end{bmatrix} t\right) \tag{A-9}$$

From the definitions, the discrete system parameters are extracted from $\hat{S}$:

$$\Phi = \hat{F}_3^T = \begin{bmatrix} \hat{S}_{55} & \hat{S}_{56} \\ \hat{S}_{65} & \hat{S}_{66} \end{bmatrix} \tag{A-10}$$

$$\Psi = \hat{G}_3^T B = \begin{bmatrix} \hat{S}_{57} & \hat{S}_{58} \\ \hat{S}_{67} & \hat{S}_{68} \end{bmatrix} B \tag{A-11}$$

$$\Gamma = \hat{G}_3^T \Gamma_c = \begin{bmatrix} \hat{S}_{57} & \hat{S}_{58} \\ \hat{S}_{67} & \hat{S}_{68} \end{bmatrix} \Gamma_c \tag{A-12}$$

$$W = \hat{F}_3^T \hat{G}_2 = \begin{bmatrix} \hat{S}_{55} & \hat{S}_{56} \\ \hat{S}_{65} & \hat{S}_{66} \end{bmatrix} \begin{bmatrix} \hat{S}_{35} & \hat{S}_{36} \\ \hat{S}_{45} & \hat{S}_{46} \end{bmatrix}. \quad \text{(A-13)}$$

APPENDIX B: MATLAB KALMAN CALIBRATION IMPLEMENTATION

```
%
% MATLAB FUNCTION
%       kalcal -- Kalman filter/smoother calibrator
%
% USAGE
%       kalcal(states, i_cmd)
%
% DESCRIPTION
%     Generates a statistically optimal estimate of the position/velocity
%     of a second-order system connected to the encoder. The torque input
%     and the sensor measurements of states are assumed noisy with known
%     covarience. The smoothed position estimate is used to generate a
%     calibration lookup table.
%
% ARGUMENTS
%     states   Rough estimate of system states from sensors [p(t),v(t)]
%     i_cmd    Current input command (Amps)
%
% RETURN VALUE
%     Calibration lookup table of the form [tau_a;eta_tau_a].
%
% GLOBALS ACCESSED
%     None.
%
% RESTRICTIONS
%
% BUGS
%
% FUTURE DIRECTIONS
%
% REVISION
%       $Revision$
%

function [lut, xs] = kalsmooth(states, i_cmd);

Ts = 0.001;% Sampling time (sec)
Nlines = 1000;% # lines/circle on encoder disk

z = [ states(1,:); ...
      states(2,:) .* (2*pi/Nlines);...  % Scale lines      => radians
      states(3,:) .* (2*pi/Nlines/Ts)...  % Scale lines/samp => radians/
sec
    ];
z1 = z(2:3,:);
z2 = z(2,:);

%
% System model: J*th'' + B*th' + u = 0
```

```
%     where th is angular position (theta)
%               J is rotational inertia
%               B is viscous friction
%               u is torque input (from actuator)
%

%
% System parameters
%
Bf = 1e-3% System viscous friction [ Nm/(rad/sec^2) => Kg*m^2     ]
J  = 0.000916% System rotational inertia [Nm/(rad/sec)  => Kg*m^2/sec ]
Km = 0.053% Motor torque constant (Nm/A) [Determined experimentally]
V  = 8.88e-9% measurement noise variance (0.015-line^2, => rad^2)
Q  = 1e-2% Spectral Density of process noise
p_hat0 = [5e-9, 0.0;% Initial state covariance
        0.0,  1e-4]

%
% Continuous 2nd-order system parameters
%
Ac = [0,1,;0,-Bf/J]% Cont-time state transition matrix
Bc = [0;-Km/J] % Cont-time cmd input distribution
Gc = [0;1]     % Cont-time process noise distribution

C_hat = [ -Ac,        eye(2,2),   zeros(2,2), zeros(2,2);...
        zeros(2,2), -Ac,        Gc*Q*Gc',   zeros(2,2);...
        zeros(2,2), zeros(2,2), Ac',        eye(2,2);...
        zeros(2,2), zeros(2,2), zeros(2,2), zeros(2,2) ];

S_hat = expm(C_hat .* Ts);

Phi = S_hat(5:6,5:6)'% State transition matrix
Psi = S_hat(5:6,7:8)' * Bc% Cmd input distribution
Cd = [1,0]     % Sensor input distribution (radians)
Gd  = S_hat(5:6,7:8)' * Gc% Process noise distribution
W   = S_hat(5:6,5:6)' * S_hat(3:4,5:6)% Process noise covariance

%
% Assume that the sensor data is in matrix z(nsensors,nsamples).
% Guess at the initial state and initial state variance
%
nstates = 2;   % Number of process states
ninputs = 1;   % Number of process inputs
nsensors = 1;  % Number of sensor outputs
nsamps = size(z,2);% Number of sensor samples
vel0_guess = mean(z1(2,1:10));
velK_guess = mean(z1(2,nsamps-10:nsamps));
x_hat0 = [z2(:,1);vel0_guess]% Initial state estimate
x_hatK = [z2(:,nsamps);velK_guess]% Final state estimate

fprintf(2,'\n');
```

```
fprintf(2,'\t*****************************************\n');
fprintf(2,'\t************Starting FWD Filter************\n');
fprintf(2,'\t*****************************************\n');

%
% Initialize the output matricies for speed
%
xf_bar= zeros(nstates,nsamps);
xb_bar= zeros(nstates,nsamps);
xf_hat= zeros(nstates,nsamps);
xb_hat= zeros(nstates,nsamps);
pf    = zeros(nstates,nsamps*nstates);
pb    = zeros(nstates,nsamps*nstates);
ps    = zeros(nstates,nsamps*nstates);
kf    = zeros(nstates,nsamps);
kb    = zeros(nstates,nsamps);
ks    = zeros(nstates,nsamps);

%
% Do forward filter
%
xf_hat(:,1) = x_hat0;
xf_bar(:,1) = x_hat0;
p_hat = p_hat0;
pf(:,1:2) = p_hat0;
tile_ndx = 3;
for ndx = 2:nsamps,
      %
      % Time update
      %
      xf_bar(:,ndx) = Phi * xf_hat(:,ndx-1) + Psi * i_cmd(ndx);
      p_bar = Phi * p_hat * Phi' + W;

      %
      % Measurement update
      %
      tmp = V + Cd * p_bar * Cd';% common sub-expression
      k_gain = p_bar * Cd' * inv(tmp);
      p_hat = p_bar - k_gain * tmp * k_gain';
      xf_hat(:,ndx) = xf_bar(:,ndx) ...
                     + k_gain * ( z2(ndx) - Cd * xf_bar(:,ndx) );

      pf(:,tile_ndx:tile_ndx+1) = p_hat;
      kf(:,ndx) = k_gain;
      tile_ndx = tile_ndx + nstates;
end

%
% Estimate acceleration from 1st difference of estimate velocity
%
acc1 = diff(xf_bar(2,:));
```

```
acc2 = diff(xf_hat(2,:));

p_bar
p_hat
k_gain

%
% Plot out forward filter results
%
figure(1)
clf
subplot(221)
plot(z(1,:),z(2,:),'r-',z(1,:),xf_hat(1,:)','y-');
xlabel('Time (sec)')
ylabel('xf_hat(1,:) -> pos (rad)')
grid
subplot(222)
plot(z(1,:),z(2,:)-xf_hat(1,:),'r.')
xlabel('Time (sec)')
ylabel('xf_hat[bar](:,1) - atan')
grid
subplot(223)
plot(z(1,:),z(3,:),'r.',z(1,:),xf_hat(2,:)','y-');
xlabel('Time (sec)')
ylabel('xf_hat(2,:) -> vel (rad/sec)')
grid
subplot(224)
plot(z(1,2:nsamps),acc2,'y.',z(1,:),Psi(2,1)*i_cmd,'r-')
xlabel('Time (sec)')
ylabel('Est Acceleration (rad/sec^2)')
grid
drawnow

%
% Extract forward filter covariances (state1^2, state2^2)
%
pf4plot = zeros(2,nsamps);
tile_ndx = 1;
for ndx = 1:nsamps,
      %
      % Get covariance for first state (position)
      %
      pf4plot(1,ndx) = pf(1,tile_ndx);
      pf4plot(2,ndx) = pf(2,tile_ndx+1);
      tile_ndx = tile_ndx + nstates;
end
%
% Plot forward filter covariances
%
figure(2)
clf
```

```
subplot(221);
plot(z(1,:),pf4plot(1,:),'y-');
xlabel('Time (sec)')
ylabel('Fwd Position Covariance')
grid
subplot(223);
plot(z(1,:),pf4plot(2,:),'y-');
xlabel('Time (sec)')
ylabel('Fwd Velocity Covariance')
grid
subplot(222);
plot(z(1,:),kf(1,:),'y-');
xlabel('Time (sec)')
ylabel('Fwd Position Gain')
grid
subplot(224);
plot(z(1,:),kf(2,:),'y-');
xlabel('Time (sec)')
ylabel('Fwd Velocity Gain')
grid
fprintf(2,'FWD filter done. Press any character to continue...');
pause
fprintf(2,'\n');
fprintf(2,'\t*****************************************\n');
fprintf(2,'\t***********Starting BKWD Filter***********\n');
fprintf(2,'\t*****************************************\n');

%
% Do backward filter
%
xb_hat(:,nsamps) = x_hatK;
xb_bar(:,nsamps) = x_hatK;
Phi_inv = inv(Phi);
p_bar = p_hat0;
pb(:,2*nsamps-1:2*nsamps) = p_hat0;
tile_ndx = 2*nsamps - 3;
for ndx = nsamps-1:-1:1,
      %
      % Time downdate
      %
      xb_hat(:,ndx) = Phi_inv * xb_bar(:,ndx+1)...
                     - Phi_inv * Psi * i_cmd(ndx);
      p_hat = Phi_inv * p_bar * Phi_inv' + W;

      %
      % Measurement downdate
      %
      tmp = V + Cd * p_hat * Cd';% common sub-expression
      k_gain = p_hat * Cd' * inv(tmp);
      p_bar = p_hat - k_gain * tmp * k_gain';
      xb_bar(:,ndx) = xb_hat(:,ndx)...
```

```
                    + k_gain * (z2(ndx) - Cd * xb_hat(:,ndx));

      pb(:,tile_ndx:tile_ndx+1) = p_bar;
      kb(:,ndx) = k_gain;
      tile_ndx = tile_ndx - nstates;
end

%
% Estimate acceleration from 1st difference of estimate velocity
%
acc1 = diff(xb_bar(2,:));
acc2 = diff(xb_hat(2,:));

p_bar
p_hat
k_gain

%
% Plot out backward filter results
%
figure(2)
clf
subplot(221)
plot(z(1,:),z(2,:),'r-',z(1,:),xb_hat(1,:)','y-');
xlabel('Time (sec)')
ylabel('xb_hat(1,:) -> pos (rad)')
grid
subplot(222)
plot(z(1,:),z(2,:)-xb_hat(1,:),'r.')
xlabel('Time (sec)')
ylabel('xb_hat[bar](:,1) - atan')
grid
subplot(223)
plot(z(1,:),z(3,:),'r.',z(1,:),xb_hat(2,:)','y-');
xlabel('Time (sec)')
ylabel('xb_hat(2,:) -> vel (rad/sec)')
grid
subplot(224)
plot(z(1,2:nsamps),acc2,'y.',z(1,:),Psi(2,1)*i_cmd,'r-')
xlabel('Time (sec)')
ylabel('Est Acceleration (rad/sec^2)')
grid
drawnow

%
% Extract backward filter covariances (state1^2, state2^2)
%
pb4plot = zeros(2,nsamps);
tile_ndx = 1;
for ndx = 1:nsamps,
      %
```

```
      % Get covariance for first state (position)
      %
      pb4plot(1,ndx) = pb(1,tile_ndx);
      pb4plot(2,ndx) = pb(2,tile_ndx+1);
      tile_ndx = tile_ndx + nstates;
end
%
% Plot backward filter covariances
%
figure(3)
clf
subplot(221);
plot(z(1,:),pb4plot(1,:),'y-');
xlabel('Time (sec)')
ylabel('Bkwd Position Covariance')
grid
subplot(223);
plot(z(1,:),pb4plot(2,:),'y-');
xlabel('Time (sec)')
ylabel('Bkwd Velocity Covariance')
grid
subplot(222);
plot(z(1,:),kb(1,:),'y-');
xlabel('Time (sec)')
ylabel('Bkwd Position Gain')
grid
subplot(224);
plot(z(1,:),kb(2,:),'y-');
xlabel('Time (sec)')
ylabel('Bkwd Velocity Gain')
grid

fprintf(2,'BKWD filter done. Press any character to continue...');
pause
fprintf(2,'\n');
fprintf(2,'\t*****************************************\n');
fprintf(2,'\t*************Starting Smoother*************\n');
fprintf(2,'\t*****************************************\n');


%
% Now generate the smoothed estimates
%
xs = zeros(nstates,nsamps);
tile_ndx = 1;
for ndx = 1:nsamps,
      k_gain = pf(:,tile_ndx:tile_ndx+1) ...
            * inv(pf(:,tile_ndx:tile_ndx+1) + pb(:,tile_ndx:tile_ndx+1));
      xs(:,ndx)   =   xf_bar(:,ndx)   +   k_gain   *   (xb_bar(:,ndx)   -
xf_bar(:,ndx));
```

```
      ps(:,tile_ndx:tile_ndx+1) = pf(:,tile_ndx:tile_ndx+1) ...
                  - k_gain * pf(:,tile_ndx:tile_ndx+1);
      ks(:,tile_ndx:tile_ndx+1) = k_gain;
      tile_ndx = tile_ndx + nstates;
end

%
% Estimate acceleration from 1st difference of estimate velocity
%
acc1 = diff(xs(2,:));

%
% Plot out smoother results
%
figure(3)
clf
subplot(221)
plot(z(1,:),z(2,:),'r-',z(1,:),xs(1,:)','y-');
xlabel('Time (sec)')
ylabel('xs(1,:) -> pos (rad)')
grid
subplot(222)
plot(z(1,:),z(2,:)-xs(1,:),'r.')
xlabel('Time (sec)')
ylabel('xs(:,1) - atan')
grid
subplot(223)
plot(z(1,:),z(3,:),'r.',z(1,:),xs(2,:)','y-');
xlabel('Time (sec)')
ylabel('xs(2,:) -> vel (rad/sec)')
grid
subplot(224)
plot(z(1,2:nsamps),acc1,'y.',z(1,:),Psi(2,1)*i_cmd,'r-')
xlabel('Time (sec)')
ylabel('Est Acceleration (rad/sec^2)')
grid
drawnow

%
% Extract backward filter covariances (state1^2, state2^2)
%
ps4plot = zeros(2,nsamps);
ks4plot = zeros(2,nsamps);
tile_ndx = 1;
for ndx = 1:nsamps,
      ps4plot(1,ndx) = ps(1,tile_ndx);
      ps4plot(2,ndx) = ps(2,tile_ndx+1);
      ks4plot(1,ndx) = ks(1,tile_ndx);
      ks4plot(2,ndx) = ks(2,tile_ndx+1);
      tile_ndx = tile_ndx + nstates;
end
```

```
%
% Plot backward filter covariances
%
figure(4)
clf
subplot(221);
plot(z(1,:),ps4plot(1,:),'y-');
xlabel('Time (sec)')
ylabel('Sm Position Covariance')
grid
subplot(223);
plot(z(1,:),ps4plot(2,:),'y-');
xlabel('Time (sec)')
ylabel('Sm Velocity Covariance')
grid
subplot(222);
plot(z(1,:),ks4plot(1,:),'y-');
xlabel('Time (sec)')
ylabel('Sm Position Gain')
grid
subplot(224);
plot(z(1,:),ks4plot(2,:),'y-');
xlabel('Time (sec)')
ylabel('Sm Velocity Gain')
grid


fprintf(2,'SMOOTHER done. Press any character to continue...');
pause
fprintf(2,'\n');
fprintf(2,'\t*****************************************\n');
fprintf(2,'\t**************Generating LUT**************\n');
fprintf(2,'\t*****************************************\n');

%
% Plot out all covariances
%
figure(4)
clf
subplot(211);
plot(z(1,:),pf4plot(1,:),'y-', ...
     z(1,:),pb4plot(1,:),'r-', ...
     z(1,:),ps4plot(1,:),'g-');
xlabel('Time (sec)')
ylabel('Position Covariance')
subplot(212);
plot(z(1,:),pf4plot(2,:),'y-', ...
     z(1,:),pb4plot(2,:),'r-', ...
     z(1,:),ps4plot(2,:),'g-');
xlabel('Time (sec)')
ylabel('Velocity Covariance')
```

```
%
% Convert the states from radians back into lines
%
xs = xs .* (Nlines/2/pi);
z2 = z2 .* (Nlines/2/pi);

%
% Extract angular data from smoothed position estimate
%     Ignore parts of dataset where |velocity| is < 0.05;
%
%s_st = 100;
%s_end = nsamps-100;
s_st = 200;
s_end = nsamps-200;
ns = s_end-s_st+1

sm_fract = xs(1,s_st:s_end) - fix(xs(1,s_st:s_end));
a2_fract = z2(s_st:s_end) - fix(z2(s_st:s_end));
sm_veloc = xs(2,s_st:s_end);
for ndx = 1:ns,
      if (sm_fract(ndx) < 0)
               sm_fract(ndx) = sm_fract(ndx) + 1;
      end
      if (a2_fract(ndx) < 0)
               a2_fract(ndx) = a2_fract(ndx) + 1;
      end
end

%
% Now generate lut
%     Note: This loop throws out points which are below a minimum velocity
%           to improve the calibrator's performance
%
lut = zeros(ns,2);
lndx = 1;
[foo,sndx] = sort(a2_fract);
for ndx = 1:ns,
      if (abs(sm_veloc(sndx(ndx))) > 30)
               lut(lndx,1) = a2_fract(sndx(ndx));
               if ((a2_fract(sndx(ndx)) < 0.2) ...
                 & (sm_fract(sndx(ndx)) > 0.8))
               lut(lndx,2) = sm_fract(sndx(ndx)) - 1;
               elseif ((a2_fract(sndx(ndx)) > 0.8) ...
                     & (sm_fract(sndx(ndx)) < 0.2))
               lut(lndx,2) = sm_fract(sndx(ndx)) + 1;
               else
               lut(lndx,2) = sm_fract(sndx(ndx));
               end
               lndx = lndx + 1;
      end
```

```
end

tossed_pntcnt = ns - lndx - 1

%
% Trim lut to final length
%
lut = lut(1:lndx-1,:);
lns = lndx-1;%LUT #samps


%
% Flattened lut
%
fl_lut = lut;
fl_lut(:,2) = fl_lut(:,2) - fl_lut(:,1);

foo = find(fl_lut(:,1) >= 0.5);
ndx = foo(1)
fl_lutr = [fl_lut(ndx:lns,1)-1,fl_lut(ndx:lns,2); ...
          fl_lut(1:ndx-1,:)];

figure(5)
clf
plot(fl_lutr(:,1),fl_lutr(:,2),'y.')
axis([-0.5,0.5,-0.02,0.02])
xlabel('Rough Intra-line Distance (Lines)')
ylabel('Correction-Term Estimate (Lines)')
grid
pause

%
% Force input table atan2 values to be monotonic so that the interp1()
% call below isn't broken
%
cnt = 1;
ndx = 1;
tmp_lut = zeros(lns,2);
while (ndx <= lns),
      %
      % Look for as many duplicates in x-axis (atan2 values) as possible
      %
      foo = ndx + 1;
      if (foo <= lns) while (fl_lut(ndx,1) == fl_lut(foo,1)),
               foo = foo + 1;
      end, end
      ndups = foo - ndx - 1;% # of duplicates found
      tmp_lut(cnt,2) = mean(fl_lut(ndx:ndx+ndups,2));
      tmp_lut(cnt,1) = fl_lut(ndx,1);
      ndx = ndx + ndups + 1;
      cnt = cnt + 1;
```

```
end
tns = cnt - 1
ndx
cnt
tmp_lut = tmp_lut(1:tns,:);
fprintf(1, '%d duplicatate(s) removed\n', ndx - cnt);

%
% Rotate LUT around so that it is in the range of [-0.5,0.5]
%
size(tmp_lut)
foo = find(tmp_lut(:,1) >= 0.5);
tns
ndx = foo(1)
tmp_lut1 = [tmp_lut(ndx:tns,1)-1,tmp_lut(ndx:tns,2); ...
          tmp_lut(1:ndx-1,:)];
%tmp_lut1 = [tmp_lut(:,1) - 0.5, tmp_lut(:,2)];

%
% Tack on single wrap-around values to beginning and end so that
interpolation
% can occur
%
tmp_lut = [tmp_lut1(tns,1)-1, tmp_lut1(tns,2); ...
         tmp_lut1; ...
         tmp_lut1(1,1)+1, tmp_lut1(1,2)];

%
% Resample the flattened lut to be periodic in column 1
%
rns = 600;% Number of samples in resampled output
rs_lut = zeros(rns,2);
rs_lut(:,1) = [-0.5:1/(rns-1):0.5]';
rs_lut(:,2) = interp1(tmp_lut(:,1), tmp_lut(:,2), rs_lut(:,1));

%
% Smooth the resampled LUT
%
kersiz = 25% Must be odd number
kernel = ones(kersiz,1) / kersiz;
sm_rs_lut = [rs_lut(:,1), circconv(rs_lut(:,2),kernel)];

%
% Compute the fft of the error function
%
rs_fft = fft(sm_rs_lut);
foo = [rs_fft(1:15,2);zeros(rns-30,1);rs_fft(rns-14:rns,2)];
sm_rs_lut = real(ifft(foo));

figure(6)
clf
```

```
subplot(211)
plot(fl_lut(:,1),fl_lut(:,2),'y.')
axis([0,1,-0.02,0.02])
title('Flattened Error estimate')
xlabel('Intra-line distance (normalized)')
ylabel('Error estimate')
grid
subplot(212)
plot(rs_lut(:,1),rs_lut(:,2),'y.',rs_lut(:,1), sm_rs_lut,'r-')
axis([-.5,.5,-0.02,0.02])
title('Resampled Error estimate')
xlabel('Intra-line distance (normalized)')
ylabel('Error estimate')
grid

figure(5)
hold
plot(rs_lut(:,1), sm_rs_lut,'r-')

lut = [rs_lut(:,1),sm_rs_lut];
return
```