# Implementation of a High-speed Sinusoidal Encoder Interpolation System

Charles F. Lepple

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Pushkin Kachroo, Chair
Hugh F. VanLandingham
A. Lynn Abbott

January 23, 2004
Blacksburg, Virginia

Keywords: brushless motor control, sinusoidal encoder feedback,
ellipse-fitting, velocity observer

# Implementation of a High-speed Sinusoidal Encoder Interpolation System

Charles F. Lepple

(ABSTRACT)

In order to meet higher performance demands on brushless motor controllers, motor feedback signals must meet correspondingly higher standards. Brushless motor controllers require angular feedback for motor commutation, and generally for one or more of the following: torque, velocity or position regulation. These feedback categories impose different requirements on the control system.

In many brushless motor applications, sinusoidal encoders have significant advantages over square-wave encoders. Signal processing techniques for sinusoidal encoder feedback signals are considered in the context of a brushless motor positioning system. In particular, a method is presented for correcting gain and offset measurement errors based on ellipse-fitting techniques.

# Acknowledgments

"If I have seen farther than others, it is because I have stood on the shoulders of giants."

– *Sir Isaac Newton, (1642–1727)*

I can only begin to describe the debt of gratitude that I owe to Jason Lewis. Jason went out of his way to bring me up to speed on brushless motor technology, always making sure that I saw how each component fit into the overall system. This is the sort of priceless learning that one simply cannot get from textbooks.

While I worked at MCS, Worth Burruss was very understanding of my academic schedule. Between Worth and Jason, I couldn't have asked for better managers. I also owe a big thanks to the rest of the folks at MCS who tolerated the occasional release of "magic smoke" from lab prototypes, and generally made work enjoyable: Sylvan, Allen, Tim, Sandie, Morgan, Tracy and Wayne.

Thanks are due to the fellow graduate students who helped out in more ways than I can count. Anbu Subramanian provided some crucial references for circle-fitting, and Joel Donahue offered many signal processing insights. Thanks also to Patricia Mellodge, Eric Moret, and Mark Morton for many interesting discussions at the FLASH lab. In addition, Brian Gold deserves credit for always raising the bar in the world of LaTeX typesetting.

Finally, I cannot forget the friends and family members who provided much-needed encouragement by constantly asking "when are you going to finish that thesis?"

# Contents

# List of Figures

# Chapter 1

# Introduction

At the most basic level, a motor controller is responsible for making a motor track a velocity or torque request input. To achieve high performance, the system requires either a high frequency feedback loop, or higher resolution on the feedback device.

This thesis presents new research on the derivation of higher-resolution feedback from sinusoidal encoders. Existing research on the fundamentals of encoder operation are presented in Chapter 4, and new applications of existing ellipse-fitting and observer algorithms are covered in Chapters 5 and 6.

One of the advantages of sinusoidal encoder signal processing as presented herewithin is that the processing techniques can be used as given, or in concert with traditional square-wave encoder processing systems. When the system speed is so high that the added resolution provided by the sinusoidal encoder is below the noise floor for the velocity signal, the sinusoidal portion can be ignored, thus allowing the system to reach even higher top speeds without the tracking problems associated with other popular processing techniques.

The target calculation platform for the signal processing algorithm is the Analog Devices ADSP-21992, a 16-bit fixed-point digital signal processor (DSP). While hardware and budgetary limitations prevented the re-

alization of a complete interpolator prototype, various aspects of the algorithms were benchmarked on the 21992 to prove feasibility.

# Chapter 2

# Target Motor Controller

The initial target for this interpolator/positioner (hereafter known by its product designator "PX") is the LA-2000 Model 62 brushless motor controller [21], designed and manufactured by Motion Control Systems, Inc. The LA-2000 has a linear power amplifier stage to drive the motor windings. By using a linear power stage as opposed to a switching power stage, the radiated electromagnetic interference is much lower.

Figure 2.1 depicts a block diagram of the LA-2000. In the interest of simplicity, features not related to this paper are not shown. These features include the safety interlocks, the frequency-locked loop interface, and the HED inputs necessary for commutation before the encoder index pulse has been located.

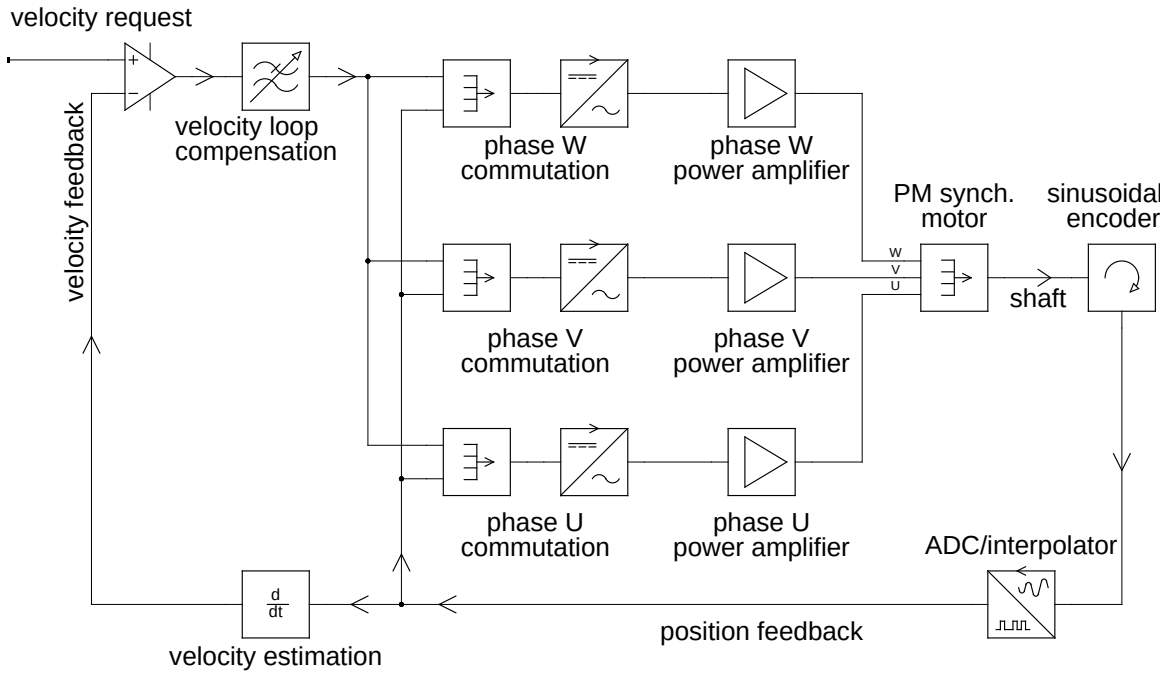



Figure 2.1: LA-2000 block diagram

Linear power stages are not especially power- or space-efficient, so the PX has also been designed to interface with the AX series [20] of switching motor amplifiers. The AX line is useful for larger motors where the driven system is less susceptible to electrical switching noise from the power stage, or where a linear amplifier would be prohibitively large.

## 2.1 Commutation

The LA-2000 has two principle modes of commutation. At startup (and when encoder signals are not available) the controller generates six-step commutation waveforms based on the approximate electrical angle returned by Hall Effect devices embedded in the motor. During this time, the controller is receiving higher resolution angle feedback from the encoder; however, the absolute reference for the encoder is not yet known. After the motor has rotated one full turn in either direction, the controller should have detected the encoder index pulse (or north marker). When the rising edge of the index pulse is received (or the falling edge, if the motor is rotating in the opposite direction), the encoder count is reset, and all subsequent encoder updates are performed modulo the encoder line count, referenced to the now-known zero angle. Besides providing extra angular resolution, the encoder also provides the mechanical angle of the motor to the controller. Further details of the interaction between the encoder and the commutation block are provided in Section 4.

In either case, the instantaneous electrical angle is known at all times (albeit with less precision if the encoder count has not yet been reset). In a synchronous machine, the desired commutation waveform travels around the stator at the same frequency as the rotor (hence, the term "synchronous"). The velocity error (output of the "velocity loop compensation" block in Figure 2.1) scales the commutation waveforms which are retrieved on a per-phase basis from a lookup table.

## 2.2 Power stage

The commutation waveform is specified in terms of phase current, and so the output of the commutation block can be seen as the input to a transconductance amplifier. As with any amplifier, there are associated system dynamics, but these dynamics are an order of magnitude faster than most outer control loops, and can effectively be ignored if the current loops are properly compensated for the inductance of the attached motor.

## 2.3 Torque regulation

Taken together, the commutation blocks and the power stage regulate the torque produced by the motor. This is because the commutation block adjusts the relative magnitudes of the phase current requests such that the machine operates in synchronous mode, and the power stage ensures that the actual phase currents track the phase current requests.

To complete the picture, the torque request signal is simply a filtered version of the velocity error (generated by the differential amplifier in Figure 2.1). The velocity error is simply the difference between the velocity request and the velocity feedback, which closes the control loop.

# Chapter 3

# DSP Selection

Several interpolator and positioner designs preceded this particular project, all using various members of Texas Instruments' floating-point digital signal processor families. While TI DSPs have achieved remarkable scores on general-purpose signal processing benchmarks [4], real-world code written for them requires a certain amount of hand-optimization to use the hardware efficiently. This project was an opportunity to investigate alternative signal processing solutions that could be used in future products as well.

Several design criteria played a part in the selection of a suitable signal processing solution. For purposes of this discussion, the most important were the minimization of the number of firmware images needed for a single product revision, minimization of power consumption, and ease of rapid prototyping.

## 3.1 Firmware and Processing Technologies

Since the PX was designed to be added on to another product with a processor and FPGA, any complete system would have at least two firmware images to synchronize with whatever was in use on the positioner.

To keep the number of separate firmware images to a minimum, it was decided that the interpolation and positioning calculations would be done in the same chip as communication and housekeeping functions. This could be accomplished with either an FPGA (and a few external data converters), or with a highly integrated DSP.

### 3.1.1 Field Programmable Gate Arrays

Advantages of field programmable gate arrays (FPGAs) have been touted countless times in digital design literature. However, the flexibility of a "sea of gates" comes with a price: all signal processing flows must be explicitly designed (either by hand-coding in a hardware description language, or using a hardware description meta-language such as Confluence [17]). The proliferation of pre-tested general-purpose DSPs frees a designer from having to reinvent the wheel when it comes to implementing (and verifying) the arithmetic operations for an FPGA processing core.

An FPGA also requires a source of configuration information, and due to the interconnect and storage constraints, neither of the processing elements on the amplifier control board could easily configure the PX FPGA.

Power consumption can also be a limiting factor in selecting an FPGA as a processing element. In many cases, the pin count of an FPGA is in rough proportion to the number of logic elements contained within. If an FPGA design uses many unmultiplexed input/output pins to eliminate bus contention, a larger chip with more pins and logic elements is required, and logic elements are wasted. This is not a critical problem during normal operation, as clocks can be gated away from unused areas of the chip. During power-up, however, the inrush current is generally proportional to the number of logic elements, and can be substantially more than the runtime current requirements.

Calculating the runtime dynamic power consumption can also be a time-consuming, iterative process. DSP power consumption is usually much easier to estimate. In the case of fixed-point DSPs, power calculation can

be as simple as determining how much current is needed to drive the bus capacitance at the external system clock frequency, and adding in a power consumption factor proportional to the core clock frequency.

### 3.1.2 Floating point DSPs

Analysis of early PX system performance revealed that the interpolation and positioning routines do not lend themselves to high optimization levels. Both the 'C3x and 'C6000 DSP families have single-cycle throughput on multiply-and-accumulate (MAC) sequences, but the MAC instructions have register allocation requirements that penalize shorter loops. Long filtering operations (as seen in communications and audio/video processing applications) benefit greatly from MAC hardware, but in these cases, the register setup time (which is not a function of the filter length) is a smaller fraction of the total computation time.

Another consideration for the positioning algorithm in particular is that the worst-case execution time (WCET) must be bounded to ensure that the desired control loop update rate is met. Related to the WCET is the interrupt latency, or the amount of time between when a processor interrupt line is asserted, and when the interrupt handler begins execution. On the 'C3x (and to a greater extent, on the 'C6000), certain compiler optimizations generate sequences of instructions which cannot be interrupted. The interrupt is handled once the instruction sequence completes. However, ADC sampling timer interrupts are typically not synchronous to the control loop, and the variable delay between interrupt line assertion and the execution of the interrupt handler code introduces timing jitter into the system. Excessive jitter invalidates uniform sampling assumptions used in the main control loop.

In short, the WCET (after taking uninterruptable code sequences and interrupt latency into account) is comparable to that obtained with general-purpose (non-DSP) microprocessors.

One possibility for future study is to implement "Approach B" in [8] (pp.

16–17) by disabling the processor global interrupt flag, and enabling the appropriate interrupt enable flags. At the end of the control loop (during dead time), the interrupt flags can be polled to see which (if any) devices require service. In theory, this has the same effect as disabling and re-enabling the global interrupt flag while executing critical code (and having individual interrupts just set a flag and return). On processors with deep pipelines, however, there is latency between when the global interrupt enable is changed, and when the interrupt subsystem recognizes this change.

### 3.1.3 Fixed point DSPs

While fixed-point DSPs lack many of the more advanced mathematical capabilities of their floating-point counterparts, they are generally simpler, and as a result, tend to achieve relatively predictable execution times.

Another benefit of the simplicity of fixed-point math is an overall savings in power. While it can be argued that more computation is necessary to achieve the same precision as with a floating-point processor, the utilization level of the floating-point hardware is usually not as great. The 'C6000 has at least eight separate functional units which can operate in parallel, and in real-world applications, it is rare to see more than four of these units active in loop kernels. This wasted hardware must be factored into the calculation of overall power consumption.

The Analog Devices ADSP-21992 was chosen over other 16-bit fixed-point DSPs for its integrated encoder interface unit and on-board analog-to-digital converter (ADC). The ADC is a 14-bit simultaneous-sampling model, with two 4-input multiplexers driving the two synchronized converters.

Simultaneous sampling is almost a mandatory selection criterion in motion control ADCs because the signals can change quickly relative to the sampling period. For measuring both sinusoidal encoder outputs and motor currents, it is crucial that both channels are sampled at the same time instant (rather than in succession). If there is a constant time delay

between corresponding samples of each channel, then this delay should be modeled as an all-pass filter with a constant group delay. If the sampling frequency is high enough, this group delay is negligible for sampling and controlling motor currents. However, since the electrical frequency of most sinusoidal encoders is a substantial fraction of the sampling frequency, this delay can contribute to errors in the angle estimation. Since the sinusoidal encoder would typically be used for velocity estimation, it is difficult to compensate for the group delay while still providing a reliable velocity estimate.

## 3.2 Typical processing tasks

In addition to the calculations covered in Chapters 5 and 6, a full-fledged positioning system must also handle the polynomial evaluation and control loop compensation for trajectory following.

Tyler [27] presents a case study of trajectory generation necessary to reduce harmful vibrations in antenna pointing applications. Wells [28] describes a multi-axis antenna trajectory generator in further detail, including example code. Both systems can be generalized beyond antenna pointing, since many general-purpose motion control systems include a motor that has a finite acceleration time (relative to the controller sampling time), and require a time-optimal solution which does not exceed system jerk, acceleration and velocity limits.

While trajectory generation and the resultant velocity profiling are desirable features for a high-performance single-axis controller, they are almost a necessity for many multi-axis systems. As an example, consider the act of drawing a diagonal line on a pen plotter. Even if the speed of the pen over the paper does not matter (path following), the velocity profiles of each axis must be matched so that one axis does not reach top speed before the other (resulting in a curve on the paper). Often, such as in machining applications, there is a portion of the path where the axes must produce

uniform-velocity motion (multi-axis trajectory following). This usually involves iteratively recalculating the velocity profiles to find a solution that satisfies the system limits.

By choosing a processor for interpolation with computational power to spare, the interpolation and positioning functions could be combined without redesigning the system.

# Chapter 4

# Position Feedback

Motor controllers vary widely in their feedback requirements. A DC brush motor velocity controller may only need a relative velocity signal as generated by a tachometer or a rotary encoder. Some setups may even go as far as "sensorless" control, where the velocity estimate is derived from an observer which considers motor current and voltage. While such systems are outside the scope of this investigation, their existence becomes important in cost-sensitive, low performance velocity control applications.

Note that the terms "position" and "angle" are often used interchangeably in motion control literature. Systems which require absolute angular feedback (as opposed to relative angular velocity feedback) are generally designed such that a given motor angle maps to a position in space for the object driven by the motor. In addition, linear motors are typically designed as an "unwrapped" rotary motor [22], and many manufacturers provide linear equivalents of common rotary feedback devices. The term "positioner" thus is commonly used to denote a motor electrical angle regulation system, regardless of the rotary or linear nature of the actual motor.

When velocity feedback is required, the traditional method is to use a tachometer. The tachometer is usually a DC brush motor with a specific back-EMF constant (usually expressed in units of V/kRPM). In an ana-

log velocity loop, the tachometer signal is subtracted from the velocity request. The velocity error is passed through a compensator to generate a torque request. The time constant of the tachometer is usually neglected, and the output voltage is taken to be proportional to the actual shaft speed.

Brushless motor controllers (including those for switched reluctance, induction, and PM synchronous machines) generally require two types of rotary feedback. Some form of angular velocity estimate is needed to provide feedback for a velocity loop. Also, since commutation in these motors is not performed by a mechanical device, an absolute angular position estimate is needed as well.

The signal quality requirements of the angular position and velocity estimates are not necessarily the same. In systems with medium-to-high inertia and few poles, commutation does not need to be especially precise. As long as the system can track the electrical angle to within roughly $\pm 45^\circ$, the motor will continue to rotate in the desired direction. By dividing each electrical cycle into six segments, and using Hall Effect sensors to determine the position of a rotor pole relative to the nearest stator pole, the electrical angle can be resolved to within $\pm 30^\circ$. This form of commutation is known as "six-step," and the resultant commutation waveforms can be directly generated from the torque request. (For precise commutation of three-phase induction machines, and brushless motors with sinusoidal back EMF, six-step commutation is replaced by three balanced sinusoids, scaled by the torque request, with each voltage waveform leading or lagging the previous phase waveform by $120^\circ$.)

Even if six-step commutation is sufficient, many velocity control and positioning applications require higher precision than what Hall Effect sensors can provide. Hall Effect devices are convenient for commutation, but exhibit a certain amount of hysteresis.

## 4.1 Resolvers

A resolver can be modeled as a synchronous, polyphase generator. The circuit model is equivalent to a transformer with a single primary and two secondary windings, where the mutual inductance between the primary and each secondary is a sinusoidal function of the rotor angle. [19, 24] The rotor winding is excited through slip rings (or a separate auxiliary transformer) with a known AC signal, and the stator pole pairs are connected to a demodulator circuit to remove the excitation signal. See Figure 4.1 for a schematic diagram. In most applications, the resolver shaft rotates at the same speed as the motor shaft via a direct mounting arrangement (such as in a hollow-shaft resolver), or through a flexible coupling.

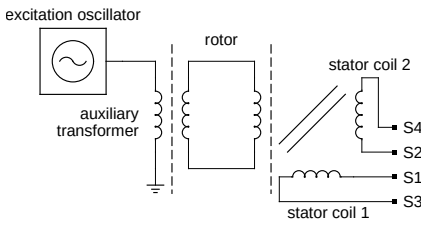



Figure 4.1: Resolver schematic diagram

The auxiliary transformer is intended to extend the useful life of the resolver by eliminating any conducted current (and the resulting maintenance of the contacts) between the rotor and the stator. The excitation signal is applied to the primary of the auxiliary transformer (on the stator), and a signal is induced on the secondary winding, mounted on the rotor. (Before the introduction of the auxiliary transformer, this connection was made via slip rings or brushes.) This design improvement also enables resolver operation in corrosive atmospheres.

In the most common case, two stator pole pairs are used (with perpendicular magnetic axes). The excitation signal undergoes a slight phase shift between the primary and the secondary phases. A phase accumulator-based

PLL is typically used to track the angle of the output signal envelopes as the rotor spins.

Using the notation in [24], the output relationships for Figure 4.1 are as follows (where $\omega_C$ is the carrier frequency in radians/second):

$$S1 - S3 = A\sin(\omega_C \cdot t + \phi)\sin(\theta_M) \quad (4.1)$$

$$S4 - S2 = A\sin(\omega_C \cdot t + \phi)\cos(\theta_M) \quad (4.2)$$

To improve noise immunity, a variation on this basic design uses three or more stator windings (each spaced 120 electrical degrees apart), and the tracking loop attempts to minimize the error from the resulting overdetermined system of equations. This configuration is not given special consideration in this analysis, since the final angular position and velocity information is essentially equivalent to the two-phase case (with a lower resultant noise floor).

The phase shift between primary and secondary (including the auxiliary transformer, if applicable) is denoted by $\phi$ in Equations 4.1 and 4.2. Since the excitation signal and phase shift are known *a priori*, the secondary phase voltages can be sampled at a point in time corresponding to the peak of the primary voltage plus the phase shift.

This method differs from conventional radio quadrature demodulation in that the modulation (primary excitation) signal is directly available to the demodulation process.[1] The resolver demodulation is actually a cross between the quadrature demodulation technique used for linear FM transmissions, and the sampling used for PSK decoding. [23]

The biggest advantage of the resolver is that once the PLL has acquired a lock on the stator signals, the absolute angle of the motor shaft is known.

[1] It is interesting to note that while quadrature demodulation techniques are used for analog signals in resolvers and encoders, the index pulse on many high-end encoders is generated by optical correlation similar in principle to the digital correlation used to demodulate direct sequence spread spectrum transmissions.

Not only does this eliminate the need for a separate index pulse, but the commutation circuitry can start up instantly without resorting to any number of workarounds to determine the electrical orientation of the rotor. [2, 9]

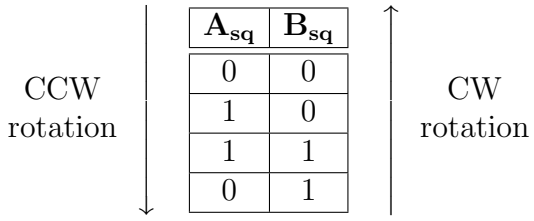
The drawbacks, however, stem mainly from the angle tracking loop in the PLL. It is difficult to compensate the tracking loop for a wide range of speeds, since the compensation depends on the excitation frequency, rotor speed, and the noise rejection requirements of the system. Resolver tracking loop compensation is especially important, since noise and aliasing of the stator signals can cause the PLL to lose its lock. Loss of lock can be destructive in high-power systems, as the commutation circuitry would then energize the wrong windings (potentially generating enough torque to exceed acceleration and jerk limits in the system). Also, the necessary loop filters introduce lag into the system, which can affect the dynamics of the outer control loops. [7] Another undesirable limitation comes from the digital implementation of most modern resolver tracking loops. There is a maximum slew rate for the angle estimate, and it is measured in bits per second. When additional bits of accuracy are requested from the tracking loop (going from 12 bits to 14 or 16), the tracking bandwidth decreases. [24]

Sources of error in the angle signal include mechanical imperfections which would cause a change in coupling as the rotor turns (due to an uneven air gap). The rotor excitation voltage must be large enough to satisfy the system noise margins on the stator output sensing circuitry (taking into account the overall transformation ratio, which is typically on the order of 0.5), and small enough to avoid saturating both the rotor and stator laminations. Since these errors will show up starting as first and second harmonics of the rotor mechanical frequency, the effect on overall system performance can be considerable.

## 4.2 Quadrature encoders

Quadrature (or incremental) encoders are an inexpensive alternative to resolvers. While the resolution of a resolver is effectively limited by the noise floor of the tracking loop, quadrature encoders are designed to produce an exact number of transitions per rotation.

Low-cost quadrature encoders (such as those found in the ubiquitous opto-mechanical computer mouse throughout the 1990's) generally use one slotted plastic disc, one LED, and two or more photoreceptors per axis. The plastic disc may be replaced by a punched metal or inscribed glass disc if necessitated by resolution or environmental requirements. The photoreceptors are placed such that the output square-wave signals are $90^\circ$ out of phase (hence the term "quadrature"). Since the transitions of the outputs are never coincident with each other during normal operation, the outputs are often considered to be a two-bit Gray code.




| $\mathbf{A_{sq}}$ | $\mathbf{B_{sq}}$ |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 1 | 1 |
| 0 | 1 |

Figure 4.2: Quadrature encoder output sequence

For a given encoder output state, there are two valid Gray code next-state transitions. These correspond to clockwise and counterclockwise rotation, and relative position can be tracked by incrementing or decrementing a counter at each transition. The encoder can then be treated as an analog-to-digital converter, converting an analog angle to a digital position word. Quadrature encoders are thus analyzed in the context of traditional ADC specifications (such as integral and differential nonlinearity: INL and DNL) by Kavanagh in [15].

One obvious disadvantage when replacing a resolver with a quadrature

encoder is the loss of absolute angle information. Some systems use additional bits, returning position information as an $n$-bit Gray binary code (such that the most significant bit has only one on-off transition per rotation, and each succeeding bit doubles the resolution). However, this method suffers from low overall resolution—each additional bit requires another portion of the disc radius, as opposed to quadrature outputs, where two bits can typically be generated from a single track (by displacing the second channel detectors by a distance corresponding to one quarter of an electrical degree).

By using gears to reduce the shaft speed, it is possible to push the envelope and obtain a few extra bits of precision. Care must be taken to ensure that the gears do not introduce excess backlash, which could increase uncertainty, and offset the resolution gains which the gears were intended to provide.

Another common workaround to obtain an absolute angle reference is to add an index channel which pulses once per revolution. This pulse resets a position counter, but still presents problems when trying to determine the initial rotor position (for commutation purposes) at power-up.

Entry-level encoders use open-collector outputs, and therefore require pull-up resistors when interfaced to TTL or CMOS circuitry. [1] Because of the time constant (and resultant low-pass filter action) associated with the pull-up resistor and the capacitance of the encoder signal cable, the upper speed limit of a quadrature encoder system typically involves a trade-off with low-speed resolution. The addition of totem-pole outputs (single-ended or differential) extends this upper limit somewhat, but the system is still limited by the sampling frequency of the encoder interface circuitry.

## 4.3 Sinusoidal encoders

### 4.3.1 Electrical and Optical Characteristics

To overcome some of the drawbacks of quadrature encoders, the encoder optics can be modified to utilize diffraction pattern techniques in creating the output signals. Instead of a sharp on-off transition, a diffraction pattern generated by the encoder disc and a mask in front of the photoreceptor produces a pair of sinusoidally varying signals (again with a $90^\circ$ phase shift). An obvious advantage over the resolver is the lack of a carrier, which eliminates the phase lag from the demodulation stage.

Typically, the signal conditioning circuitry inside the encoder contains differential drivers for the sinusoidal outputs, and the recommended termination network is accurately specified by the encoder manufacturer. When coupled to a differential receiver with a good common-mode rejection ratio, differential encoder outputs preserve signal integrity even in the presence of electrical switching noise.

As with the resolver, the analog nature of the outputs means that the output resolution is limited only by the noise floor. However, the sinusoidal encoder lends itself to dynamic resolution adjustment since there is no tracking loop to re-compensate. Simply detecting the zero crossings of the signals yields "4x" interpolation, since the number of detectable states is four times the number of lines on the encoder disc. This level of interpolation only requires discriminating between two levels on each channel, and the interpolation level can be increased with finer quantization. Since the channels are usually digitized with eight bit accuracy or greater, a variable number of low-order bits can be averaged or discarded as necessary to accommodate the noise present on each channel.

In addition, the output of zero-crossing detectors form a digital quadrature signal pair which can be processed by conventional quadrature decoding circuitry. The quadrature decoder can provide a coarse position, and the sinusoidal signal interpolator is then responsible for determining

the approximate error of the coarse position. As long as the conversion delays are taken into account, the encoder output frequency can be above the Nyquist rate for the interpolator ADC (although it must still remain below the ADC sample-and-hold bandwidth limit, and the maximum input rate of the quadrature decoder).

It may seem that the sinusoidal encoder and the quadrature encoder are both limited by sampling rates, and that the analog circuitry for the sinusoidal encoder would most likely be slower than a quadrature encoder. To make a fair comparison, it should be noted that the sinusoidal encoder requires a lower "line count" to achieve the same position accuracy as an incremental encoder. This is because every edge that a quadrature encoder generates would be equivalent to one quarter of a sine wave on a sinusoidal encoder. The sine wave can be digitized at its intermediate levels, yielding more potential positions. This means that the coarse position limits the sinusoidal encoder's top speed, and with fewer quadrature pulses per unit time, the top speed can be proportionally higher. At low speed, the interpolation circuitry can provide better sub-line-count accuracy since more time is available for A/D conversion on the sinusoidal channels.

# Chapter 5

# Encoder Signal Processing

## 5.1 Signal Non-idealities

In an actual system, the sinusoidal encoder outputs will not be ideal. To model the system, a few variables and parameters will be introduced.

$$\theta_E = N\theta_M \tag{5.1}$$

In Equation 5.1, the relationship between the electrical and mechanical frequencies is stated. $N$ is the number of lines per revolution on the encoder in question, and is therefore an integer. Common values include 512, 1000, 1024, 2000 and 2048, although $N$ may be as high as 10,000 or more in larger special-purpose encoders.

It is important to remember that with a simple two-channel encoder, only $\theta_E$ can be directly determined at any given time—$\theta_M$ must be determined by tracking the changes in $\theta_E$ over time.[1]

A certain amount of analog processing must be done before digitizing the

[1]This is what the `unwrap` function in Matlab does—when the difference between a pair of successive elements in an array is more than $\pi$, Matlab adds or subtracts $2\pi$ to the second element to minimize the relative change in angle.

$$A(\theta_E(t)) = a_1 \cos(\theta_E(t)) + a_2 \quad (5.2)$$
$$B(\theta_E(t)) = b_1 \sin(\theta_E(t)) + b_2 \quad (5.3)$$

Figure 5.1: Output equations for sinusoidal encoder

inputs. This is because most sinusoidal encoder outputs are 1 V pk-pk differential with a common mode voltage of 2.5 VDC.[2] With CMOS feature sizes shrinking, the core voltages and the resulting maximum allowable input signal magnitudes have decreased to the point where it is difficult to find a high-speed ADC chip which will allow a 2.5 V common mode voltage, let alone a 2.5 V single-ended signal.

Although TI makes a differential input ADC chip [26] which can interface directly to these signal levels, it has a top sampling rate of only 6 MSPS per channel at 12 bits (for an overall throughput of 3 MSPS), and has not proved to be reliable in testing. In particular, between two and four successive samples had to be averaged to mitigate noise observed in the low-order bits. This further reduced the effective upper frequency limit. Most other high-speed ADC choices have different common mode and full-scale voltage requirements, so the incoming signals must be scaled and level-shifted.

If the ADC has single-ended inputs, the differential encoder signals can be subtracted at the input to the level-shifting network. This has been implemented as shown in Figure 5.2. Note that the feedback resistors have been chosen for a gain of slightly less than unity. This allows software gain correction to attenuate a signal which would otherwise swing past the ADC input rails.

Because of the impracticality of hand-matching component values used in the gain and level-shifting networks, it is desirable to compensate for gain and offset mismatches in software. If both channels are sampled with

[2]Setbacken mentions that the 1 V pk-pk standard is more prevalent in Europe; however, its popularity has increased worldwide, and few manufacturers still provide 100 mV or 2.5 V pk-pk signal output options.

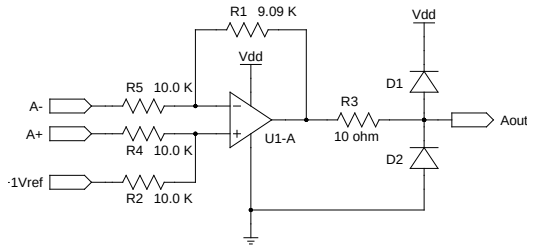



Figure 5.2: Differential level shifter for sinusoidal encoder ADC

the same effective gain, the arctangent interpolation method (discussed in Section 5.3) will still yield valid results. (If the gain is lower than the design point, the signals may contribute to a slightly higher output noise floor because of quantization.) However, PLL-based methods [7, 19, 2] have loop gains which are dependent on normalized signal amplitudes, and performance will degrade if either or both of the channels have amplitudes which differ from the design point. With both methods, a non-zero offset in either channel will result in a certain amount of high-frequency position error.

In Figure 5.3, the A channel has been offset by 10% of the amplitude. The angle error for this plot is $\arctan \frac{\sin\theta+0.1}{\cos\theta} - \theta$. Likewise, Figure 5.4 shows the effect of a 20% gain mismatch, described by the following expression: $\arctan \frac{1.2\sin\theta}{\cos\theta} - \theta$. Finally, a $+10^\circ$ phase error (B leads A by $100^\circ$, rather than the standard $90^\circ$) is shown in Figure 5.5.

Various mechanical tolerances affect the quality of the sinusoidal output signals. Signal non-idealities can be divided into two main categories, based on their fundamental frequency: those related to the mechanical frequency, and those related to the electrical frequency (which is simply the mechanical frequency times the line count).

The non-idealities can be further subdivided by their relative frequency. Ignoring random noise (specifically, noise which has a low correlation to

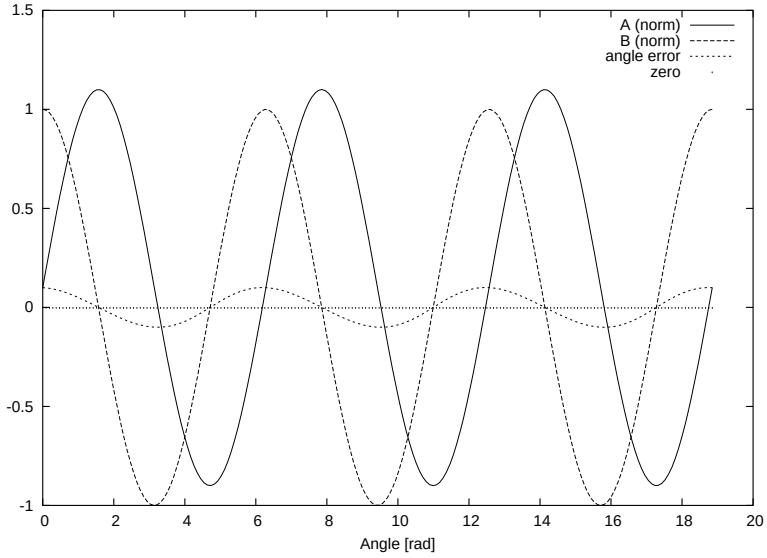



Figure 5.3: Plot depicting effects of 10% offset on calculated angle

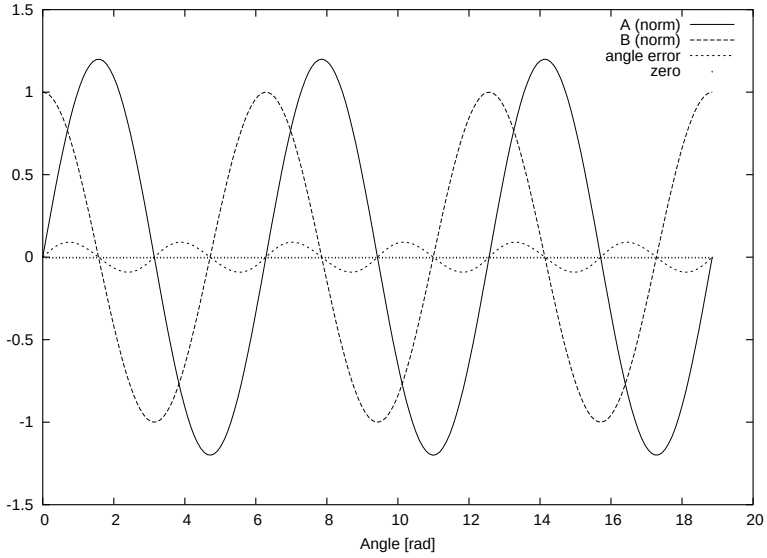



Figure 5.4: Plot depicting effects of 20% gain error on calculated angle

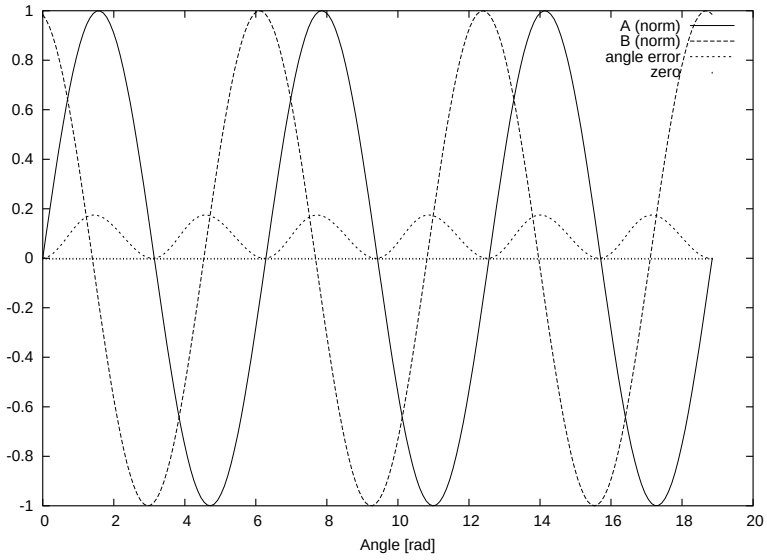



Figure 5.5: Plot depicting effects of +10 degrees phase error on calculated angle

the electrical or mechanical frequency of rotation), the deviation of the calculated angle will have predominant components of either the fundamental or the first harmonic of the electrical or mechanical frequency. For the electrical frequency, the fundamental is denoted as "1E," and the first harmonic is "2E."

When considering the effects of the various nonidealities, it should be noted that gain and phase errors produce 2E disturbances, while an offset error produces a 1E disturbance. Also, the angle error resulting from both gain and offset errors has a zero mean (over one electrical rotation), where the phase error yields a biased error waveform.

The nature of the system unknowns (due to component values) is such that there is a slight temperature correlation, but over the life of the system, the parameters are not expected to drift significantly. A reasonable solution is to perform offline adaptation of parameters when the system is being tested. If necessary, the adaptation algorithm could be optimized for inclusion in the system diagnostic routines for use after the factory burn-in procedures.

## 5.2 Gain, Phase and Offset Correction

### 5.2.1 Data collection

Ideally, the outputs from a sinusoidal encoder are equal in amplitude, centered around a known common-mode offset, and have a phase difference of $90^\circ$. Thus, the ideal outputs would trace a perfect circle (with constant angular velocity) in the $A$–$B$ plane as the encoder shaft rotates.

Two sinusoidal encoders were used to obtain real-world data. Both were manufactured by Heidenhain Gmbh., and each had a resolution of 2048 cycles per revolution. (A third sinusoidal encoder with 5000 CPR was available, but some interface issues were encountered during data collection.)

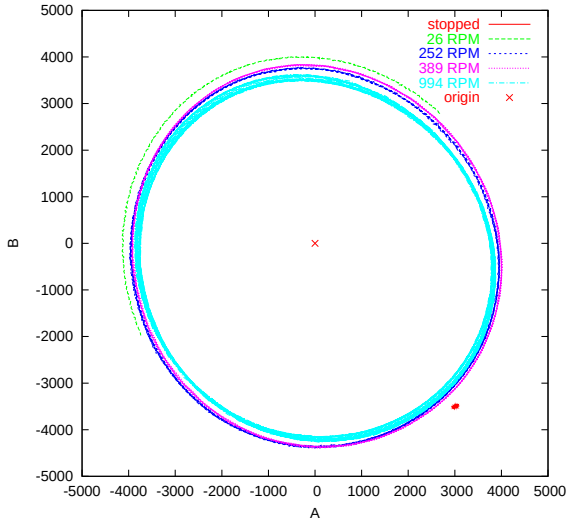



Figure 5.6: Lissajous plot of ABT spindle encoder outputs at various speeds

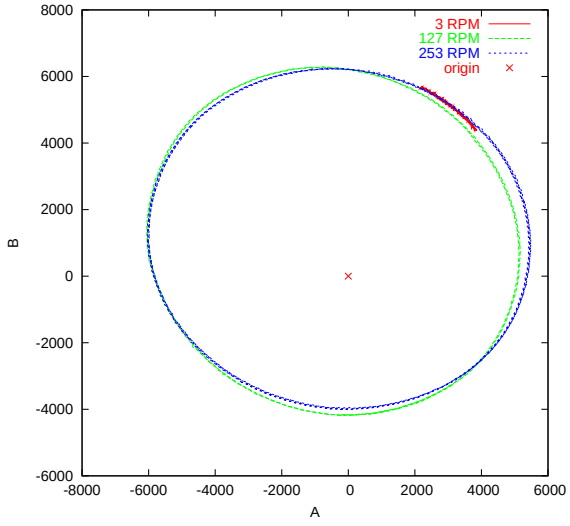



Figure 5.7: Lissajous plot of PI spindle encoder outputs at various speeds

Both encoders were mounted on air-bearing spindles. One spindle was manufactured by Air Bearing Technologies (ABT), and the encoder was aligned well axially. The second spindle, manufactured by Professional Instruments (PI), required a custom adapter and shims to mount the encoder, and as such, produced signals with more 1E error than the encoder on the ABT spindle. The signals obtained from each at various rotational speeds are plotted in Figures 5.6 and 5.7, respectively.

It can be seen from the figures that the signals are attenuated at higher speeds. On the ABT spindle, the fastest speed plotted corresponds to an output frequency of 23 kHz. At this frequency, an attenuation of a few percent is not uncommon, and this is noted on the Heidenhain datasheets.

The data collection process was limited to roughly 2000 samples at a time. This stemmed from a limitation in the memory available on the DSP platform, and the problem could have been mitigated with a different board layout. However, using the ADSP-21992 direct memory access (DMA) engine to fill external memory with ADC samples may trigger some silicon errata which were still present in the chip as late as November 2003. Lab tests showed that using the DMA engine to fill internal memory was the only way to reliably maintain a constant sample rate.

An alternative to DMA is to poll the ADC subsystem, precisely determine the time of each sample, and factor the resulting time delta into velocity calculations. This method is much harder to implement, as most discrete observers are designed for a constant sampling time. A variable sampling time would necessitate recalculating gains on-the-fly, and is probably not worth the extra effort.

It should also be noted that hardware was not in place to synchronize the start of sampling to a specific mechanical angle. Therefore, an electrical angle (equivalent to a geometric angle when plotted in this form) may correspond to any of 2048 different mechanical angles. An improvement on this setup should include some form of trigger to start sampling at a given mechanical angle (such as on a rising or falling edge of the index pulse).

Because the mechanical angle is unknown, and because the sampling period is relatively small compared to the period of rotation, compensating for gain and offset errors may only cancel out errors in the electrical angle, and may miss the "big picture" by not canceling mechanical angle errors. This situation cannot be completely analyzed without the sort of synchronization mentioned in the preceding paragraph (or the use of a much larger sample buffer).

One advantage of the smaller buffer size is that for most cases, the speed of the spindle over the buffer sampling period can be assumed to be constant. A timer-counter unit was used to ensure that the average speed (over a 200 ms interval) was regulated to within 1% before the sampling took place. In some cases, the spindle inertia was increased by rigidly coupling it to larger objects with higher moments of inertia. This was intended to mitigate the drag from windage and cogging of the spindle motor. For further study (especially for mapping and correcting 1M disturbances), a high-inertia setup such as the one described in Kavanagh's papers would be preferred.

### 5.2.2 From ellipse to circle

In order to correct for errors in the input signals, the most direct way would be to find the transfer function of the nonidealities, and apply the inverse transfer function to the measured signals.[3] Since the locus of the ideal signals is a circle, and a distorted circle can be approximated by an ellipse, it makes sense to consider methods to fit a set of points to an ellipse.

Assuming the existence of an ellipse which fits the data well, it is a trivial matter to find a 2-dimensional transformation which shifts, rotates and scales an ellipse to be a circle. When this transformation is applied to the

[3] Although gain and offset are linear operations, phase error is not. On the other hand, if the phase error is so large that a small-angle approximation cannot be made, then the signal is probably uncorrectable.

input signals, the high-frequency errors shown in Section 5.1 are greatly reduced.

### 5.2.3 Iterative ellipse-fitting methods

A survey of existing iterative ellipse-fitting methods is presented in [11]. Most concentrate on the problem of somehow reducing the sum of the distance between each point and the proposed ellipse. Some methods even constrain the solution to a circle, which will not allow correction of gain mismatches between the two channels.

The advantage of least-squares iterative methods is that the error term can be used to take an early exit from the loop when the error is sufficiently low. Conversely, since the execution time is not bounded, excessive noise may prevent the method from converging in a timely fashion.

In addition, Fitzgibbon points out that solutions to some classes of iterative methods may converge to conic sections other than circles or ellipses. In the context of correcting the inputs, these solutions (parabolic or hyperbolic, in particular) are useless because they cannot be used for extrapolation.

### 5.2.4 Non-iterative method

The Fitzgibbon method (an adaptation of which appears in Section B.2) departs from convention by replacing the iterative search with an eigenvector calculation. By constraining the parameter search to an ellipse, the algorithm is designed not to be led astray by data sets which appear to fit other conic sections.

The `gain_offset_corr.m` routine (Section B.1) uses the output of `fit-ellipse.m` to correct gain and offset errors in the input signals.

The effect of this correction can be seen in Figure 5.8. For purposes of visual comparison, the inputs have been normalized such that the average

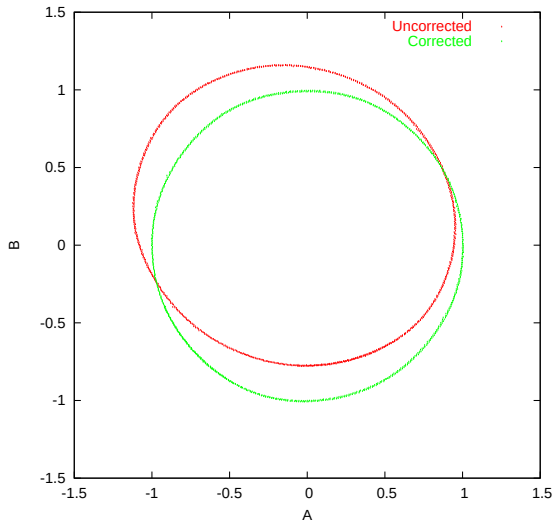



Figure 5.8: Plot of normalized encoder inputs and corrected outputs

of the major and minor radii are equal to one (since the outputs are normalized to an average of radius one by definition).

Because only the mean radius of the encoder inputs has been forced to zero, noise is still present, and the average signal-to-noise ratio is unchanged. Figure 5.9 shows what is effectively the radius error in the signal locus (i.e., the distance from corrected sample point to the origin, minus one).

## 5.3 Arctangent position feedback

Using the arctangent function to resolve angles results in a highly nonlinear system. However, as it is essentially a rectangular-to-polar coordinate transformation, certain approximations can be made to analyze the system.

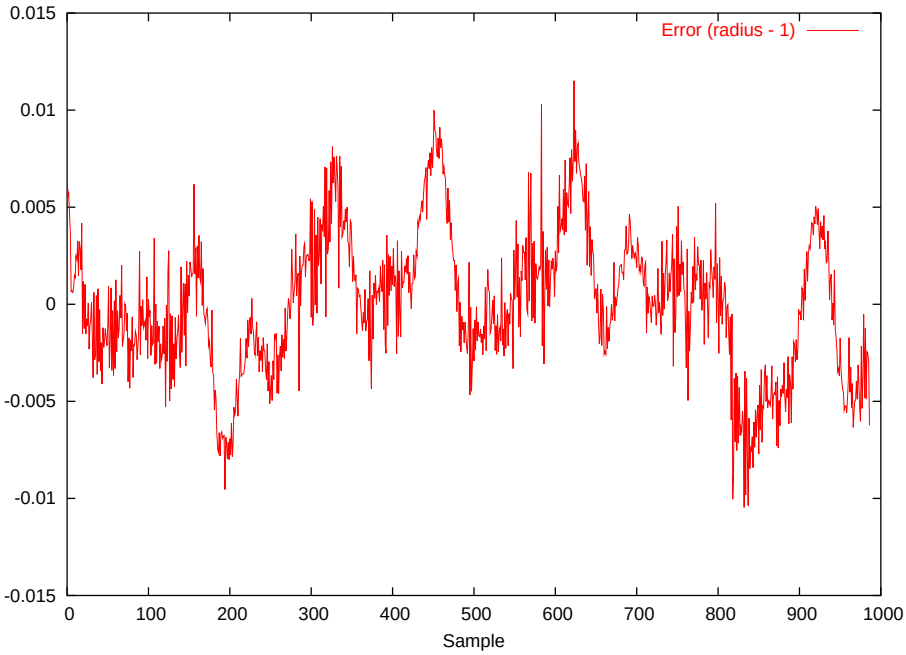



Figure 5.9: Plot of noise on corrected encoder outputs

### 5.3.1 Sinusoidal signal preprocessing

It is tempting to use a high-pass filter to remove any DC offset in the encoder signals. While this would work well for constant-velocity operation, one of the primary motivations for sinusoidal encoder interpolation is positioning precision. When using a high-pass filter, for an arbitrary break frequency, there exists a rotational speed which produces sinusoidal signals with a frequency below the filter breakpoint. After passing low-speed feedback signals through a high-pass filter, the inputs to the arctangent function would become so small as to be meaningless. With a fixed-point FIR filter, both signals would eventually fall into the quantization domain for a zero output. In a typical four-quadrant arctangent, the inputs are divided (the larger by the smaller), and this particular scenario would result in division by zero.

Another approach to remove DC offset is to perform parameter adaptation on a pair of offset signals. Ljung points out on page 363 the necessity of persistence of excitation for the parameters to converge [18], but persistence of excitation is not guaranteed if the interpolation system is permitted to operate around zero speed. Unless provisions are made to block adaptation once the speed drops below a certain point, the parameters will naturally adapt to cancel the "offset" generated by a constant angular input, and effectively result in the same erroneous operation described in the previous paragraph.

The major difficulty in controlling parameter adaptation is that the criteria for adaptation would all depend on the output of the adaptive filter. A possibility for future work would be to incorporate a speed estimate from the binary quadrature signals, but many sources on quadrature encoder velocity estimation point out the difficulties in resolving low speeds in the presence of noise, as summarized in [29].

### 5.3.2 Position Estimation

Once the sinusoidal input signals have been prepared for processing, they are passed to the arctangent function. For instantaneous position estimation, the result of the arctangent is the electrical angle estimate.

Because the electrical angle is not especially useful if the mechanical angle is not tracked as well, electrical angle wraparound at $\pm 2\pi$ must be detected. By scaling the electrical and mechanical angles by $k$ such that $\theta_E \cdot k \in [0, 2\pi \cdot k) = 2^M$ (where $M$ is a natural number), the binary representation of $\theta_E$ can be combined via Boolean operators with the estimate of $\theta_M$ obtained from edge-counting.

### 5.3.3 Velocity Estimation

Given a series of position estimates, the most straightforward way to estimate velocity is by a first-order backwards differentiation. A constant sampling time is assumed ($\Delta t = T = 1/f_s$).

$$\omega(t) = \frac{d}{dt}\theta \doteq \frac{\theta(t) - \theta(t - \Delta t)}{\Delta t} \tag{5.4}$$

Neglecting wraparound, the following equation substitutes in the arctangent method described in the previous section:

$$\hat{\omega}(z) \cdot T = \frac{d}{dt}\left(\arctan\left(\frac{A}{B}\right)\right) \tag{5.5}$$

$$= \arctan\left(\frac{A}{B}\right) - \arctan\left(\frac{Az^{-1}}{Bz^{-1}}\right) \tag{5.6}$$

Using trigonometric identities as proposed in [13], Equation 5.6 can be rewritten as follows:

$$\hat{\omega}(z) \cdot T = \arctan\left(\tan\left[\arctan\left(\frac{A}{B}\right) - \arctan\left(\frac{Az^{-1}}{Bz^{-1}}\right)\right]\right) \quad (5.7)$$

$$= \arctan\left(\frac{A \cdot Bz^{-1} - B \cdot Az^{-1}}{B \cdot Bz^{-1} + A \cdot Az^{-1}}\right) \quad (5.8)$$

The advantage of Equation 5.8 over 5.6 is that if the encoder output frequency is below the Nyquist frequency, the velocity estimate has a more reasonable bound, assuming $\arctan(\frac{y}{x})$ is a four-quadrant arctangent such as the ISO Standard C library `atan2(y,x)` function.

$$\left|\arctan\left(\frac{y_2}{x_2}\right) - \arctan\left(\frac{y_1}{x_1}\right)\right| < 2\pi \quad (5.9)$$

$$\left|\arctan\left(\frac{y}{x}\right)\right| < \pi \quad (5.10)$$

The implication of these relationships is that for a simulation involving the backwards differentiation technique of Equation 5.6, a workaround such as the MATLAB `unwrap` function must be employed. Otherwise, the velocity estimate (in units of radians per sample) may be off by a factor of $2\pi$.

If the implementation performs the velocity estimation modulo $2\pi$ (such as by mapping the domain and range of the arctangent to an integral power of two, per the suggestion for position estimation in the previous section), then the `unwrap` command is not necessary. This is an important optimization, since the modulo arithmetic comes at no added cost (for a judiciously chosen power-of-two modulus), and the `unwrap` function involves a conditional test. The conditional test has the potential to change the program flow (which flushes the pipeline on most superscalar processors), and therefore should be avoided in timing-critical applications such as the velocity estimator.

# Chapter 6

# Observers

## 6.1 Filtering the arctangent angle estimate

The angle estimate obtained from the arctangent method is still fairly noisy for use as an input to a differentiator to obtain a velocity feedback signal. To obtain cleaner feedback signals, an observer can be employed.

One of the benefits to the observer approach is that the observer gains can be tuned to fit the system. For instance, if it is known that the inertia of the load (plus expected disturbances) will not allow the system to accelerate or decelerate beyond a certain limit, then the filtering action of the observer can be increased accordingly. On the other hand, if it is known that the system may be subject to large torque disturbances, then the observer tracking gain may need to be increased.

In any case, the addition of an observer provides added flexibility to compensate for a wide range of load conditions and feedback quality. As an added bonus, in a second-order observer, the two state variables can be chosen to be position and velocity estimates. This configuration makes the error variable an approximation of acceleration. In this manner, the velocity and acceleration estimates, which would ordinarily be computed by a discrete-time derivative, are now generated as byproducts of observ-

ing the position input.

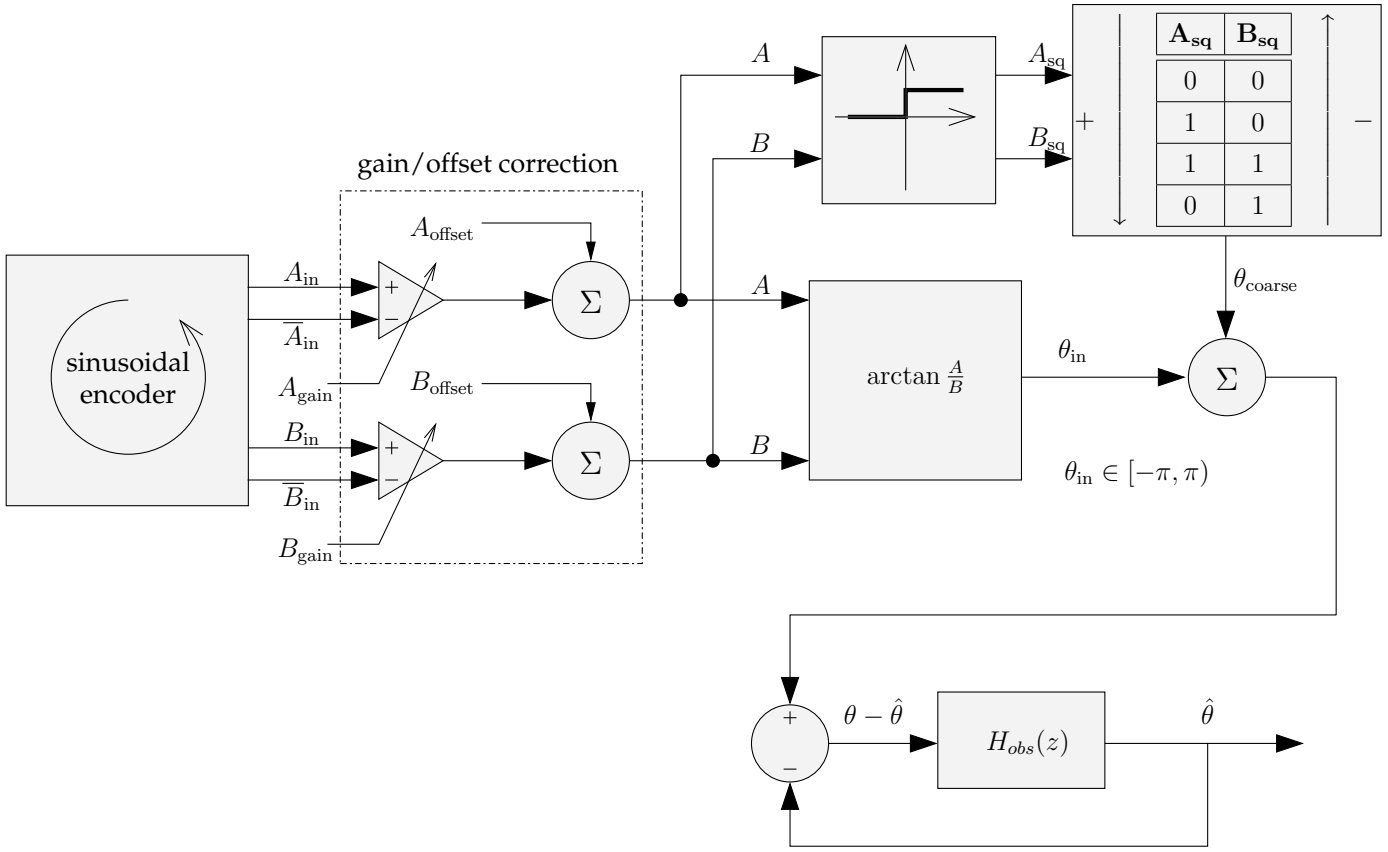



Figure 6.1: Structure of arctangent angle observer

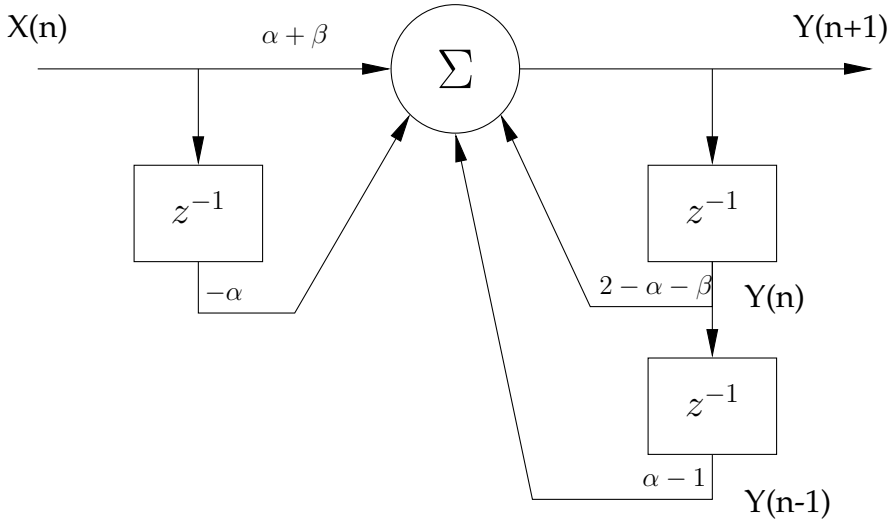



Figure 6.2: Second-order observer structure ($H_{obs}(z)$)

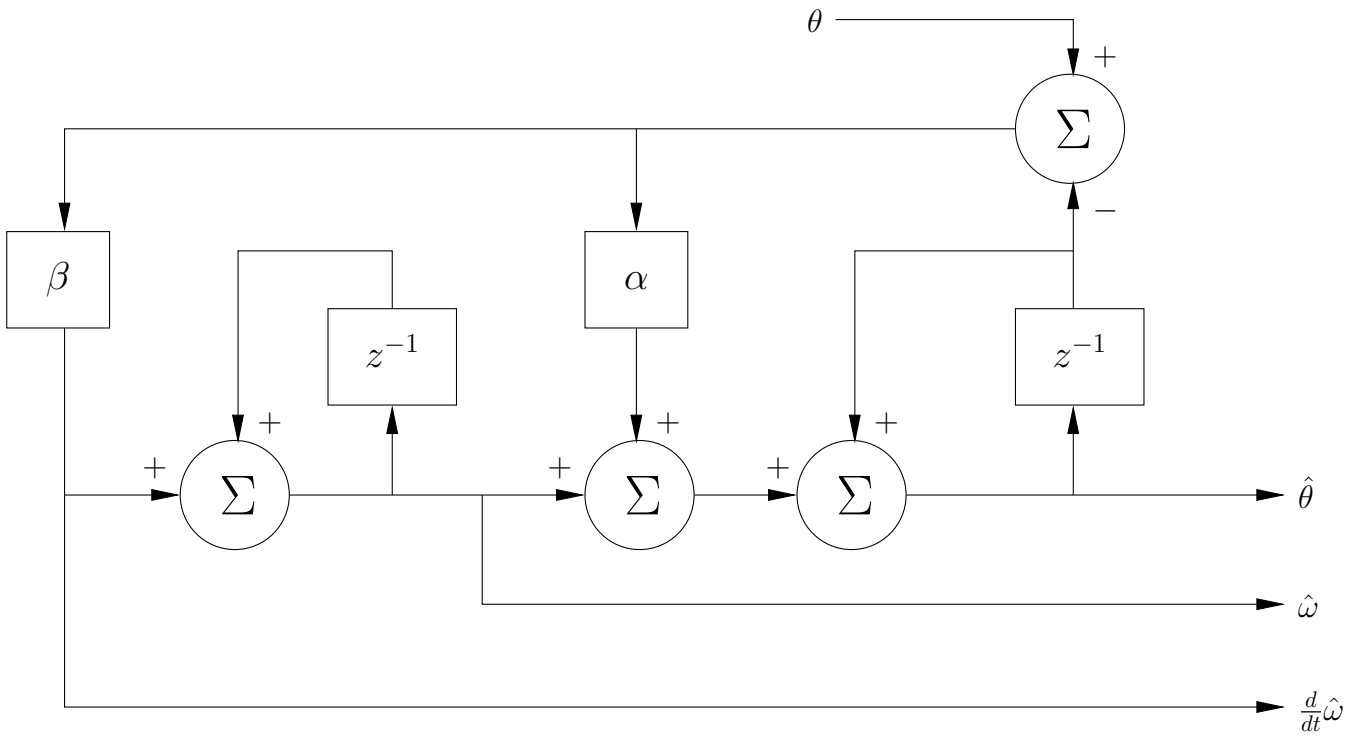



Figure 6.3: Re-structured second-order observer

## 6.2 Direct estimation of the angle via cross-multiplication

Another method of using an observer is to perform cross-multiplication of the input signals with the sine and cosine of an angle estimate, and to assume that the small-angle approximation of $\sin\theta \sim= \theta$ holds (for $\theta$ near zero). As mentioned previously, the position estimate will contain 1E and 2E components if the gains are mismatched. However, the dynamics of the observer will also be affected if the gains are not equal to unity. Unlike the arctangent angle method, there is no ratio calculation involved, so for larger input signal gains, the observer error feedback signal will be proportionally larger. Because the signal is not known *a priori*, adaptation is difficult without resorting to an on-line normalization of the input signals. Such a normalization would, on most platforms, involve more calculations than the arctangent observer described in the previous section.

# Chapter 7

# Conclusion

## 7.1 Summary of Findings

This thesis has shown that it is possible to correct 1E and 2E errors in sinusoidal encoder signals by making use of ellipse-fitting techniques. It also provides a context for analyzing such periodic errors, such that with more capable hardware, it would be possible to extend such corrective techniques to 1M and 2M errors (once the electrical errors have been minimized).

During the course of this work, special attention was paid to the eventual implementability of algorithms on a low-power fixed-point DSP. While it is probably true that the featured ellipse-fitting algorithm is not especially suited for such a processor, any on-line adaptation of the system does not need to be performed very often (given the source of the errors). The adaptation could be executed as a low-priority background task, with the other real-time calculations executing as interrupt handlers. On the other hand, proof-of-concept code showed the chosen DSP is well-suited for the calculations necessary to perform the corrections on-line, even if the adaptation is performed elsewhere.

## 7.2 Future Work

### 7.2.1 Interfacing to Quadrature Encoder Inputs

The aforementioned algorithms are appropriate for an interpolation system which is integrated with the motor controller. That is, the interpolated position is directly available to the motor control loops, and it is not transformed to an alternate representation.

There is a precedent in the motion control industry to provide standalone interpolation systems which interface to a quadrature encoder input on a motor controller. In this scenario, the motor controller would be programmed for the interpolated line count: the actual encoder line count times the interpolation factor.

Advantages include the elimination of the system integration phase in the motor controller design (assuming the motor controller already has an encoder interface unit), and the ability to bypass the interpolation unit in the feedback path to eliminate it as a source of potential error when debugging the system.

However, this introduces two potential performance concerns. First, the interpolator must reconstruct (in real time) a quadrature pulse train at the interpolated electrical frequency. Second, the motor controller must convert the quadrature pulse train into its own preferred representation of the motor position.

#### 7.2.1.1 Pulse Train Reconstruction

On the interpolator itself, its position and velocity estimates must be used to shape the simulated quadrature encoder outputs such that the quadrature decoding system will infer the correct motor position, and perhaps more importantly, motor velocity. The basic issue at hand is how to space

the transitions with the least amount of processing power, and yet without exceeding the maximum allowable jitter on the encoder interface inputs.

The most straightforward implementation is to create a discrete observer which executes a periodic task in the interpolation system. At each iteration, the task subtracts the current position from the last position, and gates a clock to a quadrature encoder simulator until the simulator has generated enough encoder edges such that the motor controller position counter matches that of the interpolator. This task then sleeps until the next control cycle. The simulated encoder pulses are thus generated in bursts.

The primary drawback of this approach is evident if the control cycle periods of the interpolator and motor controller are similar in magnitude. Assume a constant motor velocity, and a 50% duty cycle for the duration of the simulated encoder pulse bursts. Depending on how the control cycles overlap, the motor controller instantaneous velocity may fluctuate between several values (depending on how many encoder pulse bursts are observed in the preceding control cycle).

By making use of the interpolator's velocity estimate, it is possible to space the simulated encoder output edges such that the instantaneous velocity is almost constant. Many of the design requirements for such a spacing algorithm are similar to those of Bresenham's line algorithm [6], which seeks an optimal solution (in both error minimization, and computation time) for the placement of pixels to approximate a line segment on a gridded (integer) coordinate system. Instead of dealing in $x$ and $y$ coordinates, the interpolator is determining the discretized times (mapped to the $x$ axis) at which the quadrature state machine should advance (corresponding to a step along the $y$ axis).

Bresenham's algorithm is designed to optimize the rasterization of lines with angles between $0^\circ$ and $45^\circ$, noninclusive. Horizontal, vertical and $45^\circ/135^\circ$ diagonal lines are degenerate cases, and the remaining angles can be transformed into the aforementioned range through symmetry. While Bresenham solved the problem in the context of plotter path generation,

his solution as applied to encoder simulation minimizes the least-squares distance from the ideal line in time-position space. It also has the advantage of eliminating multiplication or division which might ordinarily be necessary to perform the least-squares regression.

However, this reveals an issue with pulse train reconstruction at high speed. If the sinusoidal encoder velocity is high enough, the electrical frequency (coupled with a high interpolation factor) could mean that the ideal encoder pulse train is above the frequency capabilities of the interpolator output. The velocity observer can be implemented with saturation arithmetic (and an unsaturated position accumulator) such that short bursts of high velocity can be tolerated (e.g. from feedback noise when running near the velocity limit). However, the observer would not be able to catch up until long after the velocity dips below the upper limit. This is because the error signal is effectively being integrated, and the integrator must wind down at the top speed of the simulated encoder.

#### 7.2.1.2 Pulse Train Decoding

While the effects are mild in comparison to the trouble caused by regenerating the encoder pulse train, the decoding stage adds yet another delay to the feedback path.

### 7.2.2 Merging Coarse/fine Position Data

In [7], the authors propose that the coarse position data (obtained by converting the sinusoidal encoder outputs to square waves, and decoding the quadrature transitions) should be merged with the fine position data (from the arctangent method, for instance) by simply replacing the two low-order bits of the coarse position with the two upper bits of the fine position. The assumption is that no matter how bad the noise is on the fine position, the upper two bits of the fine position should be valid.

Preferring the fine position bits over the corresponding coarse position bits implies that there is either some advantage to using the fine position, or both sets of bits are equally valid (and one was chosen to simplify a given implementation of the system).

If the fine position bits are preferable, this would suggest that the low-order coarse position bits are somehow less reliable. However, if this is the case, then the whole premise of using the coarse position bits is invalid, since the overflow from counting the low-order bits is used to determine the state of the remaining coarse position bits.

On the other hand, if the overlapping coarse and fine position bits carry the same information most of the time, then an algorithm is needed to decide which of the overlapping bits to use. This algorithm could be extended to detect failure of either the quadrature encoder interface circuitry, or the sinusoidal signal processing section.

# Appendix A

# Derivation of arctangent backwards-difference formula

The backwards difference formula for estimating the derivative of an angle computed from an arctangent is presented without proof in [13]. The following derivation was performed relying heavily on the identities and tables in [12].

Most of the derivation is straightforward; however, following the steps of the derivation allows the reader to see exactly where the range of the expression is reduced from $[-2\pi, 2\pi)$ to $[-\pi, \pi)$, and what this implies about the useful domain of this expression.

Unless otherwise noted, all instances of $\arctan$ refer to the four-quadrant principal arctangent $\arctan \frac{y}{x}$ which takes into consideration the signs of $y$ and $x$.

Equation A.1 is the original expression from Section 5.3.3. Since each operand of the subtraction operation has a range of $[-\pi, \pi)$, the result covers the range $[-2\pi, 2\pi)$.

The range of the $\tan$ function is $(-\infty, \infty)$. However, on a manifold of $[-2\pi, 2\pi)$, the $\tan$ function does not have a unique inverse, owing to wrap-around which occurs outside the domain $(-\pi, \pi)$. Because the domain of

the outer $\arctan$ function is the wrapped range of the $\tan$ function, the range of the expression is therefore limited to $[-\pi, \pi)$.

The wraparound manifests itself as an upper bound on the trackable electrical frequency, as discussed in Section 5.3.3.

$$\arctan \tfrac{A}{B} - \arctan \tfrac{Az^{-1}}{Bz^{-1}} \tag{A.1}$$

$$= \arctan\left[\tan\left(\arctan \tfrac{A}{B} - \arctan \tfrac{Az^{-1}}{Bz^{-1}}\right)\right] \tag{A.2}$$

$$= \arctan\left[\tan \tfrac{1}{2}\left(2\arctan \tfrac{A}{B} - 2\arctan \tfrac{Az^{-1}}{Bz^{-1}}\right)\right] \tag{A.3}$$

$$= \arctan\left[\frac{\sin\left(2\arctan \frac{A}{B}\right) - \sin\left(2\arctan \frac{Az^{-1}}{Bz^{-1}}\right)}{\cos\left(2\arctan \frac{A}{B}\right) + \cos\left(2\arctan \frac{Az^{-1}}{Bz^{-1}}\right)}\right] \tag{A.4}$$

$$= \arctan\left[\frac{2\sin\left(\arctan \frac{A}{B}\right)\cos\left(\arctan \frac{A}{B}\right) - 2\sin\left(\arctan \frac{Az^{-1}}{Bz^{-1}}\right)\cos\left(\arctan \frac{Az^{-1}}{Bz^{-1}}\right)}{2\cos^2\left(\arctan \frac{A}{B}\right) - 1 + 2\cos^2\left(\arctan \frac{Az^{-1}}{Bz^{-1}}\right) - 1}\right] \tag{A.5}$$

$$= \arctan\left[\frac{2\left(\frac{A}{\sqrt{A^2+B^2}}\right)\left(\frac{B}{\sqrt{A^2+B^2}}\right) - 2\left(\frac{Az^{-1}}{\sqrt{(Az^{-1})^2+(Bz^{-1})^2}}\right)\left(\frac{Bz^{-1}}{\sqrt{(Az^{-1})^2+(Bz^{-1})^2}}\right)}{2\left(\frac{B}{\sqrt{A^2+B^2}}\right) - 2\left(\frac{Bz^{-1}}{\sqrt{(Az^{-1})^2+(Bz^{-1})^2}}\right) - 2}\right] \tag{A.6}$$

$$= \arctan\left[\frac{\frac{AB}{A^2+B^2} - \frac{Az^{-1}Bz^{-1}}{(Az^{-1})^2+(Bz^{-1})^2}}{\frac{B^2}{A^2+B^2} - \frac{(Bz^{-1})^2}{(Az^{-1})^2+(Bz^{-1})^2} - 1}\right] \tag{A.7}$$

$$= \arctan\left[\frac{AB\left((Az^{-1})^2+(Bz^{-1})^2\right) - Az^{-1}Bz^{-1}\left(A^2+B^2\right)}{B^2((Az^{-1})^2+(Bz^{-1})^2) + (Bz^{-1})^2(A^2+B^2) - (A^2+B^2)((Az^{-1})^2+(Bz^{-1})^2)}\right] \tag{A.8}$$

$$= \arctan\left[\frac{AB(Az^{-1})^2 + AB(Bz^{-1})^2 - A^2Az^{-1}Bz^{-1} - B^2Az^{-1}Bz^{-1}}{B^2(Bz^{-1})^2 - A^2(Az^{-1})^2}\right] \tag{A.9}$$

$$= \arctan\left[\frac{ABz^{-1}\left(BBz^{-1} - AAz^{-1}\right) - BAz^{-1}\left(BBz^{-1} - AAz^{-1}\right)}{(BBz^{-1} + AAz^{-1})(BBz^{-1} - AAz^{-1})}\right] \tag{A.10}$$

$$= \arctan\left[\frac{ABz^{-1} - BAz^{-1}}{AAz^{-1} + BBz^{-1}}\right] \tag{A.11}$$

# Appendix B

# Source Code

The following source code was developed for GNU Octave [10], which is a mathematical language mostly compatible with MATLAB from The MathWorks.

Usage information on all of the Octave functions can be obtained at run-time by simply typing "`help` *`name-of-function`*" (if the Octave files are in the current directory).

## B.1 `gain_offset_corr.m`

This routine takes the ellipse parameters calculated in Appendix B.2 and performs the inverse transformation on the input data points.

---

```
function [Anew,Bnew] = gain_offset_corr (A,B,par)

  %% usage: [Anew,Bnew] = gain_offset_corr (A,B,par)
  %%
  %% Using par obtained from the fitellipse function, correct the gain
  %% and offset of the input data.

  %% par = [uCentre, vCentre, Ru, Rv, thetarad];
```

```
  %% Order of operations in fitellipse is scale (radii), rotate, translate
  %% (so we undo them in reverse order).
  A = A - par(1);
  B = B - par(2);
  rot = -par(5);

  new = [A,B]*[ cos(rot), sin(rot);
                -sin(rot), cos(rot)];

  Anew = new(:,1) / par(3);
  Bnew = new(:,2) / par(4);
endfunction
```

## B.2 fitellipse.m

Fitzgibbon et al. first published their circle fitting algorithm in 1999, but Dr. Fitzgibbon's web page [11] includes a refined version of the code which the authors claim to be more numerically stable when implemented in C.

The code reproduced here is slightly different than Dr. Fitzgibbon's version, incorporating a number of minor syntax changes for execution in Octave. These changes are denoted by the following comment sequence: '%%%'

```
function a = fitellipse(X,Y)

% FITELLIPSE Least-squares fit of ellipse to 2D points.
%       A = FITELLIPSE(X,Y) returns the parameters of the best-fit
%       ellipse to 2D points (X,Y).
%       The returned vector A contains the center, radii, and orientation
%       of the ellipse, stored as (Cx, Cy, Rx, Ry, theta_radians)
%
% Authors: Andrew Fitzgibbon, Maurizio Pilu, Bob Fisher
% Reference: "Direct Least Squares Fitting of Ellipses", IEEE T-PAMI, 1999
%
% @Article{Fitzgibbon99,
%  author = "Fitzgibbon, A.~W.and Pilu, M. and Fisher, R.~B.",
```

```
%   title = "Direct least-squares fitting of ellipses",
%   journal = pami,
%   year = 1999,
%   volume = 21,
%   number = 5,
%   month = may,
%   pages = "476–480"
% }
%
% This is a more bulletproof version than that in the paper, incorporating
% scaling to reduce roundoff error, correction of behaviour when the input
% data are on a perfect hyperbola, and returns the geometric parameters
% of the ellipse, rather than the coefficients of the quadratic form.
%
% Example:  Run fitellipse without any arguments to get a demo
if nargin == 0
  % Create an ellipse
  t = linspace(0,2);

  Rx = 300;
  Ry = 200;
  Cx = 250;
  Cy = 150;
  Rotation = .4; % Radians

  NoiseLevel = .5; % Will add Gaussian noise of this std.dev. to points

  x = Rx * cos(t);
  y = Ry * sin(t);
  nx = x*cos(Rotation)−y*sin(Rotation) + Cx + randn(size(t))*NoiseLevel;
  ny = x*sin(Rotation)+y*cos(Rotation) + Cy + randn(size(t))*NoiseLevel;

  % Clear figure
  figure
  %%% clf
  % Draw it
  plot(nx,ny,'o');
  % Fit it
  params = fitellipse(nx,ny);
  % Note it may return (Rotation - pi/2) and swapped radii, this is fine.
  Given = round([Cx Cy Rx Ry Rotation*180])
```

```
  Returned = round(params.*[1 1 1 1 180])

  % Draw the returned ellipse
  t = linspace(0,pi*2);
  x = params(3) * cos(t);
  y = params(4) * sin(t);                                          60
  nx = x*cos(params(5))−y*sin(params(5)) + params(1);
  ny = x*sin(params(5))+y*cos(params(5)) + params(2);
  hold on
  plot(nx,ny,'r-')
  hold off

  return
end

% normalize data                                                   70
mx = mean(X);
my = mean(Y);
sx = (max(X)−min(X))/2;
sy = (max(Y)−min(Y))/2;

x = (X−mx)/sx;
y = (Y−my)/sy;

% Force to column vectors
x = x(:);                                                          80
y = y(:);

% Build design matrix
D = [ x.*x   x.*y    y.*y    x   y   ones(size(x)) ];

% Build scatter matrix
S = D'*D;

% Build 6x6 constraint matrix
C(6,6) = 0; C(1,3) = −2; C(2,2) = 1; C(3,1) = −2;                  90

% Solve eigensystem
if 0
  % Old way, numerically unstable if not implemented in matlab
  [gevec, geval] = eig(S,C);
```

```
  % Find the negative eigenvalue
  I = find(real(diag(geval)) < 1e-8 & ~isinf(diag(geval)));

  % Extract eigenvector corresponding to negative eigenvalue
  A = real(gevec(:,I));
else
  % New way, numerically stabler in C [gevec, geval] = eig(S,C);

  % Break into blocks
  tmpA = S(1:3,1:3);
  tmpB = S(1:3,4:6);
  tmpC = S(4:6,4:6);
  tmpD = C(1:3,1:3);
  tmpE = inv(tmpC)*tmpB';
  [evec_x, eval_x] = eig(inv(tmpD) * (tmpA - tmpB*tmpE));

  % Find the positive (as det(tmpD) < 0) eigenvalue
  I = find(real(diag(eval_x)) < 1e-8 & ~isinf(diag(eval_x)));

  % Extract eigenvector corresponding to negative eigenvalue
  A = real(evec_x(:,I));

  % Recover the bottom half...
  evec_y = -tmpE * A;
  A = [A; evec_y];
end

% unnormalize
par = [
  A(1)*sy*sy,     ...
      A(2)*sx*sy,     ...
      A(3)*sx*sx,     ...
      -2*A(1)*sy*sy*mx - A(2)*sx*sy*my + A(4)*sx*sy*sy, ...
      -A(2)*sx*sy*mx - 2*A(3)*sx*sx*my + A(5)*sx*sx*sy, ...
      A(1)*sy*sy*mx*mx + A(2)*sx*sy*mx*my + A(3)*sx*sx*my*my ...
      - A(4)*sx*sy*sy*mx - A(5)*sx*sx*sy*my ...
      + A(6)*sx*sx*sy*sy     ...
      ]';

% Convert to geometric radii, and centers
```

```
thetarad = 0.5*atan2(par(2),par(1) - par(3));
cost = cos(thetarad);
sint = sin(thetarad);
sin_squared = sint.*sint;
cos_squared = cost.*cost;
cos_sin = sint .* cost;

Ao = par(6);
Au =  par(4) .* cost + par(5) .* sint;
Av = - par(4) .* sint + par(5) .* cost;
Auu = par(1) .* cos_squared + par(3) .* sin_squared + par(2) .* cos_sin;
Avv = par(1) .* sin_squared + par(3) .* cos_squared - par(2) .* cos_sin;

% ROTATED = [Ao Au Av Auu Avv]

tuCentre = - Au./(2.*Auu);
tvCentre = - Av./(2.*Avv);
wCentre = Ao - Auu.*tuCentre.*tuCentre - Avv.*tvCentre.*tvCentre;

uCentre = tuCentre .* cost - tvCentre .* sint;
vCentre = tuCentre .* sint + tvCentre .* cost;

Ru = -wCentre./Auu;
Rv = -wCentre./Avv;

Ru = sqrt(abs(Ru)).*sign(Ru);
Rv = sqrt(abs(Rv)).*sign(Rv);

a = [uCentre, vCentre, Ru, Rv, thetarad];
```

## B.3 inst_vel.m

This function performs instantaneous velocity calculation by computing the difference in angle between successive samples. Units are therefore radians per sample.

```
function vel = inst_vel (A,B)
```

```
  %% usage: vel = inst_vel (A,B)
  %%
  %% Calculate instantaneous angular velocity, considering A and B to be
  %% sine and cosine channels of a sinusoidal encoder. vel is one
  %% element shorter than A and B.
  %%
  %% Units are radians/sample.
                                                                        10
  len = length(A);

  %% Alternate form of d/dt atan2(A,B): (doesn't need "unwrap")
  Q    = A(2:len);
  OldQ = A(1:len-1);
  I    = B(2:len);
  OldI = B(1:len-1);
  vel  = atan2(OldI .* Q - OldQ .* I, OldI .* I + OldQ .* Q);
endfunction
```

## B.4 load_samples.m

This function loads samples from an Analog Devices (ADI) debugger memory dump. The ADI VisualDSP++ suite allows the user to inspect all of the target board memory, presenting the results as a table or graph. To obtain the sampled sinusoidal encoder data, a sample ADSP-21992 program was adapted to acquire a specified number of samples and terminate. Then, the VisualDSP++ debugger was used to copy the target memory to a CSV file, which Octave can load. This particular function separates out the actual samples (columns 2 and 4) from their acquisition times (columns 1 and 3, resp.), and transforms the samples to the range $[-2^{13}, 2^{13} - 1]$.

```
function [A,B] = load_samples (name)

  %% usage: [A,B] = load_samples (name)
  %%
  %% Loads encoder ADC samples from Analog Devices VisualDSP output
```

```
  %% file. Since the ADC returns 14 bits left-aligned in a 16-bit signed
  %% word, we divide by 4 and truncate.

  ret = load(name);

  ret = floor(ret/4);

  A = ret(:,2);
  B = ret(:,4);
endfunction
```

## B.5 plot_enc.m

The following is an encoder plotting procedure which simply shows encoder outputs versus time.

```
function plot_enc (A,B,title)

  %% usage: plot_enc (A,B, [title])
  %%
  %% Plots encoder channels versus sample number. Title is optional.

  fs = 2.0e6;                    % sampling frequency is 2 MHz

  xlabel(sprintf("Time [usec]; %d samples", length(A)))
  %% ylabel("Sample value")
  if nargin == 3
    title(title)
  end

  t = (0:length(A)-1)/fs*1e6;
  plot(t, A, ";channel A;", t, B, ";channel B;")
endfunction
```

# Bibliography


[1] Agilent Technologies. *Two Channel Optical Incremental Encoder Modules*, May 2002. http://literature.agilent.com/litweb/pdf/5988-6712EN.pdf.

[2] Analog Devices, One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106. *Low Cost, Complete 12-Bit Resolver-to-Digital Converter (AD2S90)*, 1999. http://www.analog.com/Analog_Root/static/marketSolutions/motorControl/pdf/AD2S90_d.pdf.

[3] Dr.-Ing. Stephan Beineke and Dr.-Ing. Andreas Bünte. Geberauswertung für geregelte Antriebe mit hoher Rundlaufgüte. Technical report, Lust Antriebstechnik, Lahnau, 2001. http://www.lust-tec.ch/de/produktneuheiten/GPOC_d.pdf.

[4] Berkeley Design Technology, Inc. Pocket guide to processors for DSP. Web page, October 2003. http://www.bdti.com/pocket/pocket.htm.

[5] Steve Bowling. Understanding A/D converter performance specifications. Technical Report AN693, Microchip Technology, Inc., 2000.

[6] J. E. Bresenham. Algorithms for computer control of a digital plotter. *IBM Systems Journals*, 4(1):25–30, 1965.

[7] J. Burke, J. F. Moynihan, and K. Unterkofler. Extraction of high resolution position information from sinusoidal encoders. Technical report, Analog Devices, 2000. http://www.analog.com/library/whitepapers/dsp/pdf/2000_sin_encoder.pdf.

[8] Giorgio C. Buttazzo. *Hard real-time computing systems: predictable scheduling algorithms and applications*. Kluwer Academic Publishers, Norwell, Massachusetts, 1997.

[9] Data Device Corporation. *Motion Feedback Technology Components: RDC-19222*, November 2003. http://www.ddc-web.com/products/Components/MFT.asp.

[10] John W. Eaton. GNU Octave. C++ source code (version 2.1.52), Copyright 1996–2003. http://www.octave.org/.

[11] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least-squares fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999. http://www.robots.ox.ac.uk/~awf/ellipse/.

[12] Charles D. Hodgman, editor. *C. R. C. Standard Mathematical Tables*, pages 403–409. Chemical Rubber Publishing Company, Cleveland, Ohio, 12th edition, October 1962.

[13] Cătălin Ionescu. Using the ADSP-21990 EZ LITE board for implementing a narrow band digital radio receiver. Technical report, Radio Consult SRL, office@radioconsult.ro, February 2003. http://www.radioconsult.ro/England/Library/UsingADSP21990ForDSR/.

[14] D. Jouve and D. Bui. Influence of the motor feedback sensor on AC brushless servo drive performances. In *PCIM 2003 Conference, Nürnberg*, 2003. http://www.infranor.fr/download/publications/FeedbSensor.pdf.

[15] Richard C. Kavanagh. Shaft encoder characterization via theoretical model of differentiator with both differential and integral nonlinearities. *IEEE Transactions on Instrumentation and Measurement*, 49(4):795–801, August 2000.

[16] Richard C. Kavanagh and John M. D. Murphy. The effects of quantization noise and sensor nonideality on digital differentiator-based rate measurement. *IEEE Transactions on Instrumentation and Measurement*, 47(6):1457–1463, December 1998.

[17] Launchbird Design Systems, Inc. Confluence logic design language. Web page, January 2004. http://www.launchbird.com/.

[18] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, 1987.

[19] Martin Mienkina, Pavel Pekarek, and Frantisek Dobes. *DSP56F80x Resolver Driver and Hardware Interface*. Motorola, Inc., March 2002. AN1942/D.

[20] Motion Control Systems, Inc. Digital brushless servo drive amplifiers. Product web page, October 2003. http://www.motcon.com/axseries.html.

[21] Motion Control Systems, Inc. High performance linear amplifiers. Product web page, August 2003. http://www.motcon.com/la2000.html.

[22] Krishnan Ramu. *Switched Reluctance Motor Drives: Modeling, Simulation, Analysis, Design, and Applications*. Industrial Electronics Series. CRC Press, Boca Raton, FL, 2001.

[23] Theodore S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ, July 1999.

[24] Robert M. Setbacken. *Feedback Devices in Motion Control Systems*. Fithian Press, 1997. See also http://www.renco.com/106014.htm.

[25] Robert M. Setbacken. Dynamic resolution for optical encoders. Technical report, Renco Encoders, Inc., 26 Coromar Drive, Goleta, CA 93117, 2001. http://www.renco.com/106016.htm.

[26] Texas Instruments Inc., P.O. Box 655303, Dallas, TX 75265. *THS1206 12-Bit 6 MSPS Simultaneous Sampling Analog-to-Digital Converters*, November 2000. SLAS217E.

[27] S. R. Tyler. A trajectory preprocessor for antenna pointing. Technical Report 42–118, JPL, August 1994. http://tmo.jpl.nasa.gov/tmo/progress_report/42-118/title.htm.

[28] Donald C. Wells. Jerk-minimizing trajectory generator in C. GBT Memo 203, National Radio Astronomy Observatory, 520 Edgemont Road, Charlottesville, VA 22903–2475 USA, December 1999. http://www.cv.nrao.edu/~dwells/.

[29] David L. Wilson. Motor control seminar. Presented in Roanoke, VA by Motorola and Avnet, October 2003.

# Vita

Charles Lepple was born and raised in Alexandria, Virginia. A 1996 graduate of Thomas Jefferson High School for Science and Technology, Charles entered Virginia Tech in the fall of that year. Despite numerous diversions such as the Virginia Tech Solar Car Team, the Autonomous Vehicle Team, and being exposed to more than what is generally considered to be a healthy level of LaTeX and open source software, he still managed to earn his BSEE degree in four years. Charles spent the next four years attending graduate school at Tech (concentrating in controls and signal processing) while dabbling in RF propagation measurement, software engineering, and even the occasional stint of customer support at Wireless Valley Communications in Blacksburg. After it became painfully obvious that the dot-com bubble had burst, he spent a year and a half working at Motion Control Systems in New River, VA. Charles is currently employed by the Advanced Programs group at Raytheon in Falls Church, VA.