Amplitude and Frequency Demodulation Controller for MEMS Accelerometer

by

Lane Gearle Brooks

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

Bachelor of Science

and

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

© Lane Gearle Brooks, MMI. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly paper and electronic copies of this thesis document in whole or in part.

Author .....

Department of Electrical Engineering and Computer Science

December 15, 2000

Certified by .....

Paul Ward

Charles Stark Draper Laboratory

Thesis Supervisor

Certified by .....

Rahul Sarpeshkar

Assistant Professor

Thesis Supervisor

Accepted by .....

Arthur C. Smith

Chairman, Department Committee on Graduate Students

Amplitude and Frequency Demodulation Controller for MEMS Accelerometer

by

Lane Gearle Brooks

Submitted to the Department of Electrical Engineering and Computer Science
on December 15, 2000, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science
and
Master of Engineering in Computer Science and Engineering

Abstract

Draper Laboratory is developing a high precision MEMS accelerometer. This thesis describes and analyzes the electronics which produce the acceleration estimate and control the actuators within the sensor. The Vector Readout method of amplitude and frequency demodulation is described and shown to be a high precision, environmentally stable method of demodulation. The Vector Readout method of demodulation uses a Hilbert Transform filter and the CORDIC algorithm to simultaneously estimate both the amplitude and phase of a signal. A feedback controller is designed to hold the oscillation of a mass resonator at a constant amplitude.

Thesis Supervisor: Paul Ward
Title: Charles Stark Draper Laboratory

Thesis Supervisor: Rahul Sarpeshkar
Title: Assistant Professor

Acknowledgments

The thesis was prepared at The Charles Starke Draper Laboratory, Inc., under Draper Laboratory IR&D project 13066. Publication of this thesis does not constitute approval by Draper or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

Author _____

Lane Brooks

I must thank many people who have helped me to get to this point. There are the obvious people of my parents, wife, family, and teachers. There are also several people that I must specifically acknowledge in helping me with this project:

Paul Ward: Paul is my advisor at Draper Laboratory, and he has generously given of his time in discussions with me. It is through these discussions and his written documents that I learned a lot of this material. He and others had been thinking about this project well before I started it, and they really laid the foundation of it.

Dave McGorty: I worked with Dave in implementing this project. His digital logic implementation experience greatly helped me. Furthermore, he helped to develop the demodulation method used for this project.

Rich Elliott, Marc Weinberg: I worked for Rich and Marc for two summers testing MEMS gyros prior to working on this thesis. It is their time and efforts with me that enabled me to understand much of the physics and control issues with this project.

Rahul Sarpeshkar: Professor Sarpeshkar and others in his group provided very useful feedback on this project. His probing questions really forced me to think about alternatives and subtleties. Furthermore, Professor Sarpeshkar was my recitation instructor who taught me almost everything I know about classical control, so he is the one who enabled me to figure out and write Chapter 6 of this thesis.

Other Draper Employees: Tom King, Amy Duwel, Nathan St. Michael, Brian Johannson, Judy Nettles all helped me out in various ways with this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project Overview

Draper Laboratory is working to build a high precision MEMS Vibratory Accelerometer (VA), and the project discussed in this thesis is part of that effort. The tasks of this thesis are to design, analyze, and implement two components of the system. One is the signal processing which produces the acceleration readout, and the other is the controller which controls the resonator within the accelerometer to ensure an accurate acceleration readout.

Project Objectives

Within the Vibratory Accelerometer is an oscillating mass whose spring stiffness changes as a function of the acceleration. This changing spring stiffness changes the resonant frequency of the oscillator—just as when one tunes a guitar string, the natural frequency of the string changes as the spring stiffness of the string changes. As a result, there are two specific tasks that the electronics of the accelerometer must perform.

Readout Objectives: One task is to generate a readout signal indicating the amount of acceleration the device is experiencing. This is done by determining the instantaneous frequency of the output signal of the accelerometer and translating

that frequency into an acceleration estimate.

Controller Objectives: The second task is to generate a control signal to hold the amplitude of the resonating mass constant. This is necessary to obtain an accurate readout measurement because the resonator has non-linearities which couple the amplitude and frequency.

Design Specifications

The accelerometer must meet very aggressive design specifications which are:

Dynamic Range: 1μg to 100g

Acceleration Error: $< 1\mu$g

Velocity Random Walk: $< 0.014 \frac{\text{ft}}{\text{s}\sqrt{\text{hr}}}$

To ensure that the electronics can meet these constraints, the system will be analyzed under two conditions. The first is under the assumption that the resonator within the accelerometer is an ideal second order system. Then the intrinsic noise sources of the electronics can be characterized and analyzed to ensure that they can meet the specification. The second condition which will be analyzed is when various noise sources that model the imperfections of the system are injected into the signal path. This will ensure that the the electronics are designed well enough so that the system as a whole can meet the design specifications.

1.2 Sensor Details

For this project, the accelerometer can be represented as a block as shown in Figure 1-1*. One input, labeled $f(t)$, is the force applied to the resonating mass within the sensor. This force keeps it oscillating. The other input signal, labeled $g(t)$, is the acceleration that the device experiences, and it is the signal that we are trying to estimate for readout. The output signal, labeled $x(t)$ in Figure 1-1, is a measurement

*For simplicity, only one oscillator is included in this discussion. In reality there is a second oscillator which produces a fully differential signal. The second oscillator is discussed only when it is applicable.

Figure 1-1: Accelerometer Input/Output Representation

of the position of a mass resonator within the sensor. Modeling the resonator as a second-order system, the position can be described by the differential equation

$$m\ddot{x}(t) + b\dot{x}(t) + k(t)x(t) = f(t), \tag{1.1}$$

where m is the mass, b is the viscous damping, and $k(t)$ is the spring constant of the mass resonator. Notice that $k(t)$ is not constant. It can be modeled as an affine function[†] of $g(t)$:

$$k(t) = k_1 + k_g g(t). \tag{1.2}$$

1.2.1 Readout Details

The goal of this accelerometer is to obtain an estimate $\hat{g}(t)$ of the actual acceleration $g(t)$ using the position signal $x(t)$. The model of the position and acceleration signals as expressed in Equations 1.1 and 1.2 provides a the relationship between $x(t)$ and $g(t)$. However, since $k(t)$ is not constant, solving Equation 1.1 for $x(t)$ in a closed form is difficult if not impossible. Several books, such as Reference [6], have been devoted to techniques of finding approximations to the solution of Equation 1.1 in closed form under certain constraints. We will use some of these approximations later. Now, we make a simple approximation to the solution by realizing that when $k(t)$ is constant and when the force being applied exactly cancels the damping ($f(t) = b\dot{x}(t)$) that the system will ring at its natural frequency, which is $\omega_n = \sqrt{\frac{k(t)}{m}}$. Thus, if the controller is built to force the system to ring at its natural frequency, then the relation between the frequency of the position signal and the acceleration will be

$$\omega_n(t) \;=\; \sqrt{\frac{k_1 + k_g g(t)}{m}}. \tag{1.3}$$

[†]Since the affine relationship dominates, higher order relations between $k(t)$ and $g(t)$ will be neglected.

This means that if the position signal $x(t)$ is frequency demodulated to find the instantaneous operating frequency of the resonator, we will have obtained an estimate $\hat{\omega}_n(t)$ of the actual natural frequency $\omega_n(t)$. Since it is possible to invert Equation 1.3, the acceleration can then be estimated as

$$\hat{g}(t) = \frac{m}{k_g}\hat{\omega}_n^2(t) + \frac{k_1}{k_g}. \tag{1.4}$$

Therefore, we have a means of estimating the acceleration $g(t)$ by frequency demodulating the position signal $x(t)$ and then performing the nonlinear compensation described in Equation 1.4. This is shown in the block diagram in Figure 1-2.

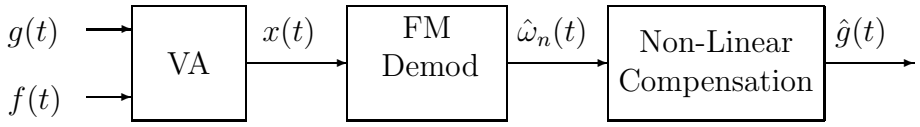


Figure 1-2: Block Diagram of General Readout Tasks

1.2.2 Control Details

In order to find an accurate estimate of the acceleration, the electronics must hold the amplitude of the resonators constant. The reason can be seen when a cubic spring force is added to the differential equation in Equation 1.1. The resulting equation is a nonlinear differential equation as follows:

$$m\ddot{x}(t) + b\dot{x}(t) + k(t)x(t) + k_3x^3(t) = f(t). \tag{1.5}$$

With this additional term in the equation, the natural frequency described in Equation 1.3 becomes

$$\omega_n(t) = \sqrt{\frac{k_1 + k_g g(t) + \frac{3}{4}k_3\alpha^2(t)}{m}} \tag{1.6}$$

where $\alpha(t)$ is the oscillation amplitude of the resonator and k_3 is the cubic spring

constant [12]. This shows that the amplitude is coupled with the frequency and that if the amplitude of the oscillators is not held constant, then false estimates of $g(t)$ will be obtained because changes in amplitude $\alpha(t)$ will be interpreted as changes in $g(t)$.

The first task to controlling the amplitude is to estimate the amplitude. This must be done by amplitude demodulating the position signal $x(t)$. Then a controller can adjust the driving force $f(t)$ in such a way as to hold the amplitude fixed. These tasks are included in the block diagram in Figure 1-3.

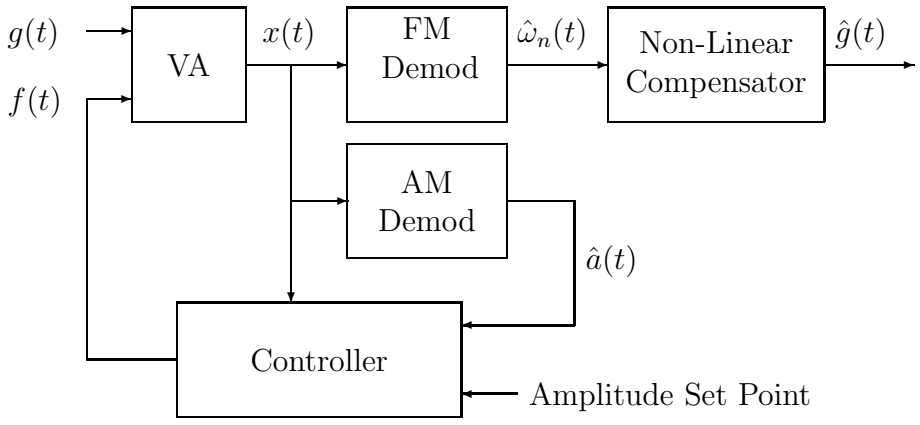


Figure 1-3: Block Diagram of Readout & Controller Tasks

1.3 Thesis Overview

Figure 1-3 shows the blocks that are designed and analyzed in this thesis. Chapter 2 introduces an AM and FM demodulation technique called the *Vector Readout Method*. This method is not a well known method, but it presents the best solution to this design problem. Thus in Chapter 3 the FM demodulation abilities of this technique are rigorously analyzed and compared to other frequency demodulation methods, and then in Chapter 5 the amplitude demodulation abilities are analyzed and compared to other methods. In Chapter 4 the non-linear compensator is analyzed and designed to meet the specifications. Finally, the controller is designed in Chapter 6.

Chapter 2

Vector Readout Method of Demodulation

The *Vector Readout* method of demodulation is a technique which can be used to AM and FM demodulate a signal simultaneously. It was invented for this project by Paul Ward and Dave McGorty of Draper Laboratory [12]. Subsequently, similar ideas were found in other publications [4], but these neglect implementation details. The method invented by Paul Ward and Dave McGorty and presented here is unique in that it provides very efficient algorithms which make the Vector Readout method practical.

2.1 Vector Readout Overview

Given a signal that is both amplitude and frequency modulated

$$x(t) = a(t)\cos\theta(t),$$

the goal in AM demodulation is to find the instantaneous amplitude $a(t)$, and the goal in FM demodulation is to find the instantaneous frequency $\omega(t)$, which is the derivative of the phase

$$\omega(t) = \frac{\mathrm{d}\theta(t)}{\mathrm{d}t}.$$

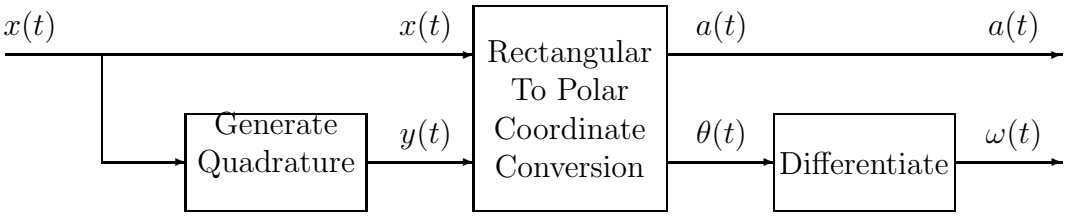


Figure 2-1: Vector Readout Method of AM and FM Demodulation.

The block diagram of Figure 2-1 illustrates the procedures of the Vector Readout method of demodulation. The first step is to generate a quadrature (90° phase shifted) signal $y(t)$ which is

$$y(t) = a(t) \sin \theta(t).$$

The quadrature signal $y(t)$ and in-phase signal $x(t)$ can be thought of as a complex rotating vector represented in rectangular coordinates as shown in Figure 2-2. This vector has a radius of $a(t)$ and an angle of $\theta(t)$, so the instantaneous amplitude and phase can be obtained if the vector is converted from rectangular to polar coordinates. The phase can then be differentiated to yield the frequency.

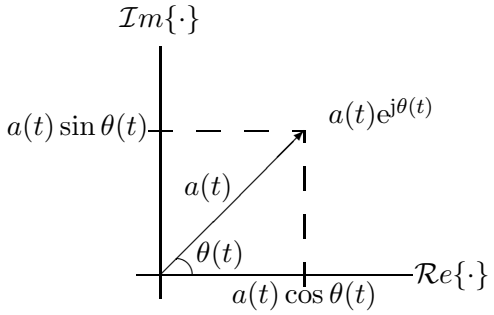


Figure 2-2: In-Phase & Quadrature Signals Represented as a Complex Vector.

The tasks presented above are not easily implemented in analog circuits, but there are digital algorithms which can perform the tasks of Figure 2-1 efficiently. Namely, the quadrature generation can be performed by a Hilbert Transform filter, and the rectangular to polar coordinate conversion can be performed by the CORDIC algorithm. Furthermore, it is shown in Chapters 3 and 5 that using these specific algorithms, the Vector Readout method is able to perform a high-precision, stable AM and FM demodulation.

The remainder of this Chapter focuses on the implementation details of the Vector Readout method. A block diagram of the implementation details is provided in

Figure 2-3. Notice that since this is a digital method, that an A/D converter has

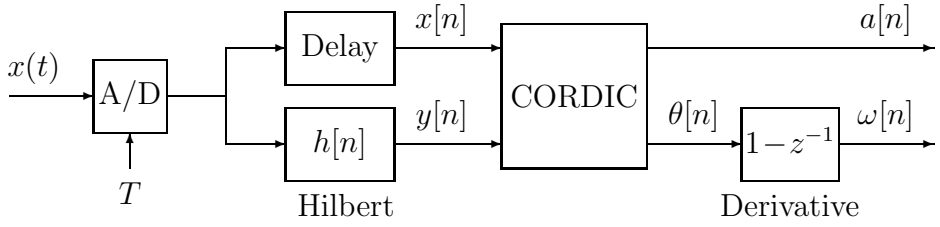


Figure 2-3: Implementation Block Diagram of Vector Readout Method.

been introduced. Also notice that a delay block has been added to the in-phase signal path. This is to keep the in-phase and quadrature channels synchronous as will be explained later.

2.2 A/D Converter

The first step of the Vector Readout Method is to sample the input signal $x(t)$. The two concerns with sampling a signal are that quantization and aliasing be suppressed to acceptable levels. The effects of quantization will be considered in Chapters 3 and 5, but here we find the appropriate sampling rate to ensure that aliasing is below acceptable levels.

The signal to be demodulated can be expressed as

$$\begin{aligned} x(t) &= a(t)\cos\theta(t) \\ &= a(t)\cos[\omega_c t + \kappa_c v(t)] \\ &= \underbrace{a(t)\cos\kappa_c v(t)}_{a_1(t)}\cos\omega_c t - \underbrace{a(t)\sin\kappa_c v(t)}_{a_2(t)}\sin\omega_c t \\ &= a_1(t)\cos\omega_c t - a_2(t)\sin\omega_c t \end{aligned} \tag{2.1}$$

where ω_c is the carrier frequency, κ_c is the modulation scale factor, and $v(t)$ is the phase modulation (or the integral of the frequency modulation). Furthermore, Equation 2.1 allows us to view $x(t)$ as the difference of two AM signals where $a_1(t)$ and $a_2(t)$ are the amplitude modulations. If we assume that $a_1(t)$ and $a_2(t)$ are band limited to Ω_c, then the carrier frequency ω_c must be greater than Ω_c in order that $a(t)$

and $\theta(t)$ be recoverable [9], and since the total bandwidth of $x(t)$ will be $\Omega_c + \omega_c$, the total bandwidth will always be less than $2\omega_c$. Thus, to ensure no aliasing when $x(t)$ is sampled, the sampling frequency must be at least $4\omega_c$, which means the sampling time must be

$$T < \frac{\pi}{2\omega_c}.$$

When this condition is satisfied, the discrete-time signal can be expressed as

$$\begin{aligned} x[n] = x(nT) &= a[n]\cos\theta[n] \\ &= a[n]\cos(\omega_d n + \kappa_d v[n]) \\ &= \underbrace{a[n]\cos\kappa_d v[n]}_{a_1[n]}\cos\omega_d n - \underbrace{a[n]\sin\kappa_d v[n]}_{a_2[n]}\sin\omega_d n \\ &= a_1[n]\cos\omega_d n - a_2[n]\sin\omega_d n \end{aligned} \tag{2.2}$$

where $\omega_d = \omega_c T$ and $\kappa_d = \kappa_c T$. Thus, the discrete-time signal also looks like an amplitude modulation and has all the information the continuous-time signal has insofar as aliasing does not occur.

2.3 Quadrature Generation Via Hilbert Transform

2.3.1 Ideal Hilbert Transform Filter

Following the A/D converter is the Hilbert Transform filter which generates the quadrature signal. The transfer function of the ideal discrete-time Hilbert Transform is

$$H(\mathrm{e}^{\mathrm{j}\Omega}) = \begin{cases} \mathrm{j}, & -\pi < \Omega < 0 \\ -\mathrm{j}, & 0 < \Omega < \pi. \end{cases} \tag{2.3}$$

The Hilbert Transform filter has unity gain and causes a -90° phase shift to real signals. Since it is linear, we can consider how each part of $x[n]$ as defined in Equation 2.2 gets affected by the Hilbert Transform filter separately. If $a_1[n]$ has a Fourier

transform $A_1(e^{j\Omega})$, then

$$a_1[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} A_1(e^{j\Omega})e^{j\Omega n}\,d\Omega,$$

so we define the first term of Equation 2.2 as

$$\begin{aligned} x_1[n] &= a_1[n]\cos\omega_d n \\ &= \frac{1}{4\pi}\int_{-\pi}^{\pi} A_1(e^{j\Omega})\left(e^{j(\Omega+\omega_d)n} + e^{j(\Omega-\omega_d)n}\right)\,d\Omega. \end{aligned}$$

and when $x_1[n]$ is sent into the Hilbert Transform filter, the output will be

$$\begin{aligned} y_1[n] &= \frac{1}{4\pi}\int_{-\pi}^{\pi} A_1(e^{j\Omega})\left(-je^{j(\Omega+\omega_d)n} + je^{j(\Omega-\omega_d)n}\right)\,d\Omega \\ &= \underbrace{\left(\frac{e^{j\omega_d n} - e^{-j\omega_d n}}{2j}\right)}_{\sin\omega_d n}\underbrace{\frac{1}{2\pi}\int_{-\pi}^{\pi} A_1(e^{j\Omega})e^{j\Omega n}\,d\Omega}_{a_1[n]} \\ &= a_1[n]\sin\omega_d n \end{aligned}$$

A similar derivation shows that when the second term of $x[n]$ as define in Equation 2.2 goes through the Hilbert Transform filter it becomes

$$x_2[n] = -a_2[n]\sin\omega_d n \quad \stackrel{H(e^{j\Omega})}{\longrightarrow} \quad y_2[n] = a_2[n]\cos\omega_d n.$$

The complete result is the sum of $y_1[n]$ and $y_2[n]$ which is

$$\begin{aligned} y[n] &= a_1[n]\sin\omega_d n \;+\; a_2[n]\cos\omega_d n \\ &= a[n]\cos\kappa_d v[n]\sin\omega_d n \;+\; a[n]\sin\kappa_d v[n]\cos\omega_d n \\ &= a[n]\sin\left(\omega_d n + \kappa_d v[n]\right) \\ &= a[n]\sin\theta[n]. \end{aligned}$$

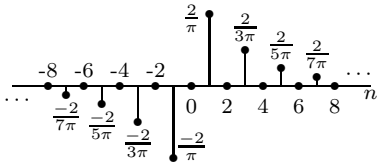
Thus, the Hilbert Transform filter produces a perfect quadrature signal

$$x[n] = a[n]\cos\theta[n] \quad \stackrel{H(e^{j\Omega})}{\longrightarrow} \quad y[n] = a[n]\sin\theta[n].$$

2.3.2 FIR Hilbert Transform Filter

The problem with using the ideal Hilbert Transform filter as defined in Equation 2.3 is that its unit sample response is [7]

$$h[n] = \begin{cases} \dfrac{2\sin^2(\pi n/2)}{\pi n}, & n \neq 0, \\ 0, & n = 0, \end{cases}$$



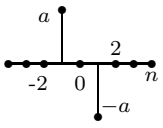
and this is a non-causal, IIR filter (which is impossible to implement). Thus one can only approximate the ideal Hilbert Transform filter. Anti-symmetric FIR filters make good candidates for this approximation because, while their gain has ripple on it, they produce the desired $-90°$ phase shift over all frequencies.

In discussing anti-symmetric FIR filters it is easier to analyze them with their unit sample response centered at the origin. Such a filter is non-causal, but since the unit sample response is finite, it can be delayed or shifted to the right to be made causal. Since the filter is anti-symmetric, this shift will result in an additive linear phase term, which is a constant group delay, so the in-phase channel must also be delayed by the same amount to keep it synchronous with the quadrature channel.

The unit sample response of the ideal Hilbert Transform filter is anti-symmetric, so one could employ any number of windowing techniques to produce an anti-symmetric FIR filter that approximates the ideal Hilbert Transform filter [7]. However, since any anti-symmetric filter produces an exact $-90°$ phase shift over all frequencies, one need not restrict themselves to an all-pass filter approximation. One can use the Hilbert Transform filter as a bandpass filter so that it rejects certain signal bands while producing the desired phase shift on other bands (however, as we shall see, there can be hardware savings if the frequency response is kept symmetric).

The fact that the phase of any anti-symmetric FIR filter causes an exact $-90°$ phase shift over all frequencies can be seen by considering its frequency response. Consider an example of a length 3 filter with unit sample response $h[n]$ defined as

$$h[n] = a\delta[n+1] - a\delta[n-1].$$



Its Fourier transform is

$$\begin{aligned} H(\mathrm{e}^{\mathrm{j}\Omega}) &= a\mathrm{e}^{\mathrm{j}\Omega} - a\mathrm{e}^{-j\Omega} \\ &= 2\mathrm{j}a\sin\Omega. \end{aligned}$$

It is not hard to extend this example to show that any odd-length anti-symmetric filter of length N has a frequency response

$$H(\mathrm{e}^{\mathrm{j}\Omega}) = 2\mathrm{j}\sum_{k=1}^{(N-1)/2} a_k \sin k\Omega.$$

This has a magnitude and phase of

$$\left|H(\mathrm{e}^{\mathrm{j}\Omega})\right| = 2\left|\sum_{k=1}^{(N-1)/2} a_k \sin k\Omega\right| \tag{2.4}$$

$$\angle H(\mathrm{e}^{\mathrm{j}\Omega}) = \begin{cases} 90^\circ, & -\pi < \Omega < 0 \\ -90^\circ, & 0 < \Omega < \pi \end{cases} \tag{2.5}$$

So the phase is a perfect -90° shift, but since the magnitude is a sum of sine waves, the gain response will have a ripple on it. Therefore, the filter coefficients need to be chosen so that the gain of the filter is as flat as possible over the bandwidth of interest. Note that this filter has an automatic zero at zero and π, and if the gain response is kept symmetric, then the odd harmonics of the sum of sine waves will be zero [8]. This means that half the filter coefficients will be zero and that the filter can be implemented with half the multiplications it would require otherwise. This makes for an efficient implementation.

The gain ripple can be made arbitrarily small, but this is at the expense of making the filter larger, which requires more hardware and increases the delay needed to make the filter causal. The gain ripple introduces error in demodulation, and its effects are characterized in Chapters 3 and 5. If the *Optimal* or *Equiripple* filter approximation [7] technique is to design the Hilbert Transform filter, then the gain ripple will be bounded over the bandwidth of interest (which means we can bound the error which

results in the demodulation).

Once the filter is designed, it will need to be made causal. If the number of taps in the filter is N and N is odd, then the unit sample response must be shifted by $(N-1)/2$ samples. Then to account for this delay in the quadrature channel, the in-phase channel also needs to be delayed by $(N-1)/2$ samples. This suggests that an odd-length filter is better so as to avoid implementing a half-sample-delay filter.

If the FIR filter is implemented in Direct Form then the delay lines between the in-phase and quadrature channels can be shared as shown in Figure 2-4. This Figure also shows that the even filter coefficients are zero thus saving on the number of multiplications and additions.

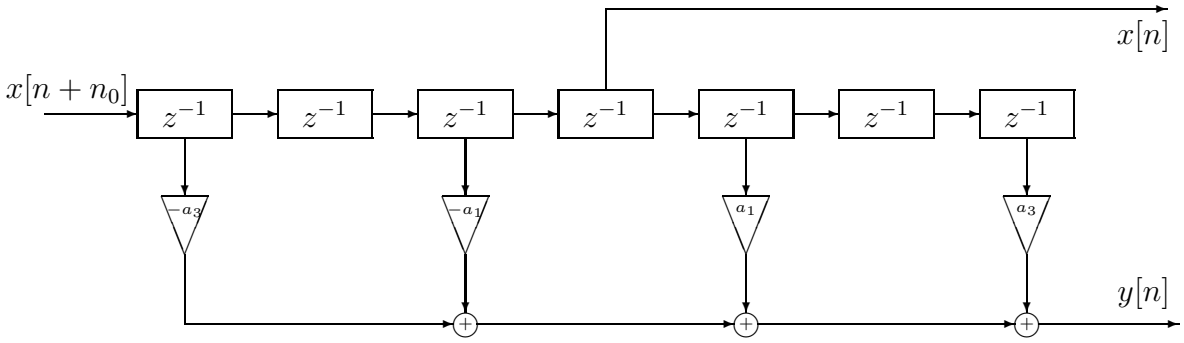


Figure 2-4: Implementation of Hilbert Transform Filter and In-Phase Delay.

While folding is not shown in Figure 2-4, this figure makes it obvious that another savings in the number of multiplications could be obtained by folding the filter so that taps with opposite coefficients are subtracted first and then multiplied by the coefficient. Folding would reduce the number of multiplies by a factor of 2, and so by keeping the frequency response symmetric and by folding the filter, the number of multiplications can be cut by a factor of 4.

2.4 CORDIC Algorithm

As shown in Figure 2-3, the CORDIC block follows the Quadrature generation block, and it is used to convert the vector from rectangular to polar coordinates. This algorithm was first published in 1959 by Jack Volder [10] and since then has been

extended to handle many different transcendental functions such as ln(), tanh(), multiplication, division, sin(), and cos() [11]. The beauty of this algorithm is that it requires no multiplications. It only requires shift registers, adders/subtractors, and a small lookup table. This is amazing considering that a conversion from rectangular to polar coordinates is defined mathematically as

$$\begin{aligned} a &= \sqrt{x^2 + y^2} \\ \theta &= \arctan\left(\frac{y}{x}\right) \end{aligned}$$

The CORDIC algorithm works by taking the vector stored in rectangular coordinates and successively rotating it to have zero angle (which means the y register gets mapped to 0 and the x register gets mapped to the amplitude a).

$$[x\ y] \longrightarrow [a\ 0]$$

The total amount of rotation is tracked to produce the angle estimate. On the i^{th} iteration, the vector is rotated by an angle α_i according to the following relation:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} \cos\alpha_i & \sin\alpha_i \\ -\sin\alpha_i & \cos\alpha_i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{2.6}$$

The angle of rotation α_i is picked a priori to be $\alpha_i = \arctan 2^{-i}$, which means that $\sin\alpha_i = \dfrac{2^{-i}}{\sqrt{1+2^{-2i}}}$ and $\cos\alpha_i = \dfrac{1}{\sqrt{1+2^{-2i}}}$. So if we plug these relations into Equation 2.6, it yields

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \frac{1}{\sqrt{1+2^{-2i}}} \begin{bmatrix} 1 & 2^{-i} \\ -2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \tag{2.7}$$

If we did not have that scale factor in front of the matrix in the above equation, it would be easy to perform the above rotation in digital logic since it is easy to multiply a number by 2^{-i} (this corresponds to a shift). So we if choose to neglect performing the normalization imposed by the scale factor in Equation 2.7, then on each iteration,

our vector will grow in magnitude. For this application this amplitude growth can be neglected since after a fixed number of iterations the total growth scale factor will be fixed. In other applications this gain may be an issue, so at the end, the vector can be returned to its actual magnitude in a single normalization step. It turns out that this gain approaches approximately $2.647\ldots$ as the number of iterations gets large, so it is very reasonable to neglect it for this application.

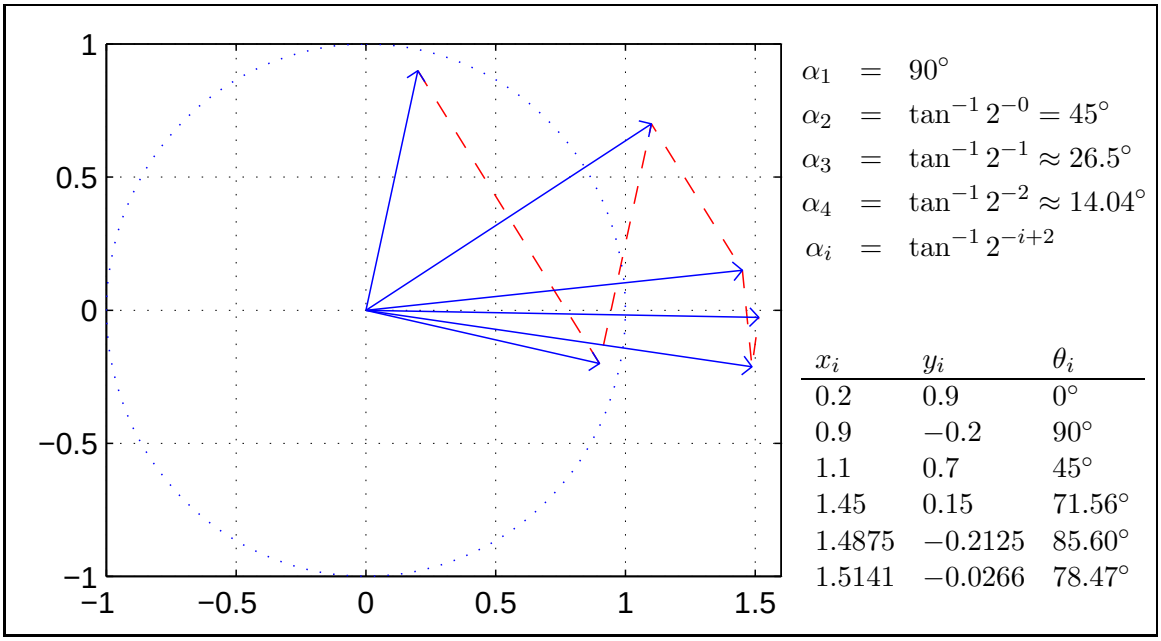
Remember the goal is to rotate the vector to have an angle of zero, so on each iteration the CORDIC block must decide whether it will rotate by a positive angle or negative angle. It decides based on the value of the y_i register. If y_i is negative, then to get the angle closer to zero, it must rotate by a positive α_i, and conversely, if y_i is positive then it must rotate by a negative α_i. Therefore, the transformation on each iteration is

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & \pm 2^{-i} \\ \mp 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

As it makes this decision on each iteration, it also must update the register which tracks the total amount of rotation. To do this, it must have the actual value of each α_i stored in a lookup table, and then on each iteration it looks up the corresponding value of the α_i and then either adds or subtracts it from the running total depending on whether it rotated by a positive or negative angle.

Figure 2-5 provides an example of the CORDIC algorithm after it has run 5 iterations. The values of the x_i, y_i, and θ_i (where θ_i is the running total of the amount of rotation) are provided on each step. As the number of iterations increases, the value in the θ_i will get arbitrarily close to the actual angle of the system, and the x_i register will get arbitrarily close to the actual magnitude of the vector (or to within a fixed scale factor of the actual magnitude). See References [10] or [11] for a more detailed explanation of the CORDIC algorithm and variations of it which allow for other transcendental operations.

The precision of the CORDIC algorithm is of concern for this project, and there are several important articles that address this issue ([1], [3], and [5]). The angle measurement can be thought of as the sum of the true measurement and an error



x_i	y_i	θ_i
0.2	0.9	0°
0.9	−0.2	90°
1.1	0.7	45°
1.45	0.15	71.56°
1.4875	−0.2125	85.60°
1.5141	−0.0266	78.47°

Figure 2-5: Example of the CORDIC Algorithm Running 5 Iterations.

which is zero mean, uniformly distributed, white* noise. The range or endpoints of the uniformly distributed noise determine the variance, and these endpoints are set by the number of iterations that are performed in the CORDIC algorithm, and on the i^{th} iteration it is $\pm 2^{-i}$. This looks just like a B fractional bit A/D converter whose endpoints would be $\pm 2^{-B}$, and so the CORDIC algorithm basically produces as many bits of angle precision as iterations.

The error in the amplitude estimate of the CORDIC is related to the error in the angle measurement by $e_a = Ka(1-\cos\theta_e)$, where θ_e is the error in the angle estimate, K is the inherent amplitude gain of the CORDIC (which approaches 2.647), and a is the true amplitude. This can be approximated as $e_a \approx \frac{Ka\theta_e^2}{2}$, which shows that if the angle estimate is uniformly distributed and has a variance of σ^2 then the variance of the amplitude estimate will be $\frac{9}{5}K^2a^2\sigma^4$. Given that $\sigma \ll 1$, the amplitude estimate noise power will be less than that of the angle estimate, and so the angle estimate is the more noisy estimate for a fixed number of iterations. This means we will want to pick the number of iterations in the CORDIC based on the required precision of the angle estimate.

*See Section 3.2.1 for conditions when the CORDIC angle quantization noise can be approximated as white.

The CORDIC algorithm has a tradeoff between precision and the amount of hardware and delay. This tradeoff, however, is probably not significant in most applications because the Hilbert Transform filter will be the dominant source of area and delay. The CORDIC algorithm requires 3 registers (x_i, y_i, and θ), an adder/subtracter, a barrel shifter, 1 small ROM to store each α_i, and some control logic. The Hilbert Transform filter, on the other hand, requires at least one multiplier, a tap delay line, at least one adder, and some control logic. Tricks can be played so that the Hilbert Transform filter only requires one multiplier and adder, but even still, when one considers the size of the tap delay line and the multiplier, they will be much larger than the CORDIC hardware.

The delay of the CORDIC is also of negligible concern when it is compared to the delay of the Hilbert Transform filter. This assumes that the CORDIC block is being clocked by a system clock which is much faster than the sampling clock. The Hilbert Transform delay is always going to be $(N-1)/2$ sampling clock delays. So, for example, suppose the Hilbert Transform filter is 51 taps and the sampling frequency is f_s. Then the delay of the Hilbert Transform filter is $\frac{25}{f_s}$. Now suppose that the system clock is $50f_s$ and the CORDIC has 25 iterations, then the delay of the CORDIC would be $\frac{1}{2f_s}$, which is 50 times faster than that of the Hilbert Transform filter.

This means that CORDIC is an efficient algorithm in terms of area and delay when compared to the Hilbert Transform filter, which indicates that the Hilbert Transform filter is the place to start when one is optimizing for delay or area since it is the dominant source of both.

2.5 Implementation Details

There are several design decisions one must make when implementing the Vector Readout Method. Following is a summary of the constraints that have either been shown already or will be derived in later chapters. They are listed here as a reference.

Vector Readout Design Rules

A/D Sampling Frequency: $> 4f_c$ Hz

f_c is the carrier frequency of the signal to be demodulated. (See Section 2.2).

A/D Quantization Amount: B fractional bits.

The amount of acceptable A/D quantization is application specific, so here we say that the signal is quantized to B fractional bits. In Chapters 3 and 5 we find how this noise propagates through to the amplitude and frequency estimates so that one can find B for a given SNR, but here we use B to find the widths of the output registers of the Hilbert Transform and CORDIC block assuming that one wants to match quantization noise power of these blocks to that of the A/D converter.

Hilbert Transform Quantization Amount: B fractional bits.

If one implements the Hilbert Transform filter as an all-pass filter approximation, then the A/D converter noise will go through the filter and have the same power on the output. Furthermore, if the Hilbert Transform filter is implemented in Direct Form as shown in Figure 2-4, then the quantization of the Hilbert Transform filter can be localized to the output. This means quantizing the output to B bits will make the quantization noise power of the Hilbert Transform filter match that of the A/D converter.

CORDIC Angle Quantization Amount: $\lceil B + \log_2 2\pi a \rceil$ fractional bits.

Equation 3.14 in Chapter 3 is the result of a derivation that shows how the A/D quantization noise propagates through onto the angle estimate, and it follows from this equation that if the A/D conversion produces a noise signal with power σ^2 then it will cause a noise signal on the normalized[†] angle estimate with a power of

$$\sigma_\theta^2 = \frac{\sigma^2}{4\pi^2 a^2},$$

where a is the amplitude of the signal. The noise power that results on the angle

[†]The normalized angle estimate is obtained by dividing the angle estimate by 2π. The CORDIC algorithm inherently does this so that the angle can be thought of as a 2's complement number between -1 and 1.

estimate when it is quantized to B_θ fractional bits is $\sigma_\theta^2 = \frac{2^{-2B_\theta}}{12}$, so equating these two and solving for B_θ given that $\sigma^2 = \frac{2^{-2B}}{12}$ yields

$$B_\theta \quad = \quad B + \log_2 2\pi a. \tag{2.8}$$

This means that if the amplitude is $a = 1$, then $B_\theta = B + 2.65$, so the angle measurement would need three additional fractional bits of precision in order to match its quantization noise power to that of the A/D converter.

CORDIC Amplitude Quantization Amount: B fractional bits

If the A/D converter introduces quantization noise with power σ^2, then Equation 5.6 in Chapter 5 finds that the mean square error in the amplitude estimate due to A/D converter quantization is also σ^2. This means that the fractional bit width of the amplitude estimate can match the fractional bit width of the A/D converter.

CORDIC Iterations: $\geq B_\theta$

B_θ is the fractional bit width of the angle estimate register. Since the CORDIC produces 1 bit of angle precision for each iteration, it is most efficient to make the number of iterations at least match the width of the angle register.

Chapter 3

Frequency Demodulation Analysis

It was previously established that the first task in obtaining an estimation of the acceleration is frequency demodulation, and this chapter analyzes the capabilities of the Vector Readout method at FM demodulation. Then this method is compared to other methods of FM demodulation.

3.1 Noise Source Definitions and Metrics

A signal $x(t)$ that is both frequency and amplitude modulated can be expressed as

$$x(t) = a(t)\cos\theta(t) \tag{3.1}$$

where $a(t)$ is the amplitude modulation and $\theta(t)$ is the phase modulation. The goal in FM demodulation is to find the instantaneous frequency, which is the time derivative of the phase modulation:

$$\omega(t) = \frac{\mathrm{d}\theta(t)}{\mathrm{d}t}.$$

Figure 3-1 shows a block diagram representation of the ideal FM demodulator where the input is the signal $x(t)$ and the output is the instantaneous frequency $\omega(t)$. The

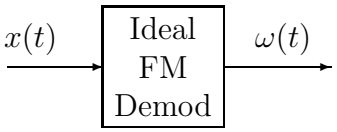


Figure 3-1: The Ideal FM Demodulator Outputs the Instantaneous Frequency.

Vector Readout method of FM demodulation can be represented as a combination of an ideal frequency demodulator and a noise source generator as depicted in Figure 3-2. Thus, if the noise source is called $e(t)$ and the estimate of the frequency is called

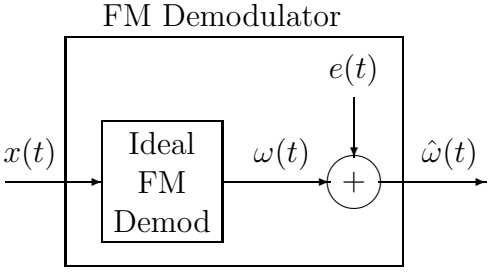


Figure 3-2: FM Demodulator Represented as an Ideal FM Demodulator and a Noise Source.

$\hat{\omega}(t)$, then the output of an estimator is

$$\hat{\omega}(t) = \omega(t) + e(t).$$

Because this noise is generated within the demodulator, it will be called the *Intrinsic Noise* of the estimator, and this chapter seeks to characterize it for the Vector Readout method of FM demodulation.

There is another source of noise that is of concern as well. This noise will be called the *Input Noise*. It occurs when the signal $x(t)$ is corrupted by an additive noise source $v(t)$ prior to demodulation as depicted in Figure 3-3. For this accelerometer this noise

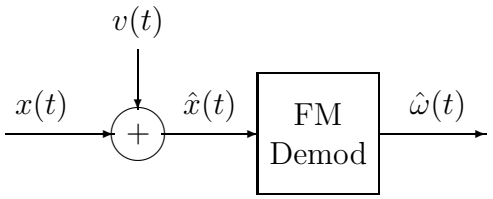


Figure 3-3: Input Noise $v(t)$ Corrupts the Signal Prior to Demodulation.

represents the fact that we cannot measure exactly the position of the oscillator but can only obtain an estimate of it. For this analysis we consider two cases of Input Noise, both of which model the actual Input Noise of the accelerometer. One is when $v(t)$ is a zero-mean, wide-sense stationary random process with variance σ_v^2. Since the position signal will be sent through an anti-aliasing filter prior to being sampled, we assume that this noise is white over the bandwidth of the A/D converter. This noise can come from the first gain stage of the analog electronics or from mechanical noise in

the sensor. This will be called *White Input Noise*. The second input noise of interest will be called *Harmonic Input Noise*, and it occurs when $v(t)$ is a deterministic sum of harmonics:

$$v(t) = a_1(t)\cos[k_1\theta(t) + \phi_1] \; + \; a_2(t)\cos[k_2\theta(t) + \phi_2] \; + \; \cdots \tag{3.2}$$

The k_x's, a_x's, and ϕ_x's in Equation 3.2 are unrestricted, but it is assumed that the amplitude of these harmonics is much smaller than the amplitude of the signal to be demodulated, $|a(t)| \gg |a_x(t)|$.

For all cases of Input Noise it is desirous to know how the noise propagates through the demodulator to effect the frequency estimate. Thus, it is helpful to move the input noise through the demodulator so that it is represented as additive output noise as shown in Figure 3-4. Therefore, as with the Intrinsic Noise, the effect of the Input

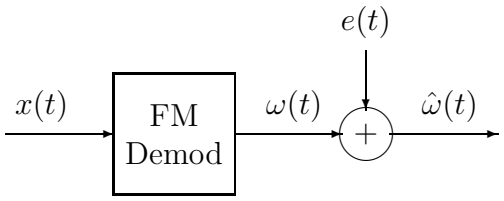


Figure 3-4: Input Noise Moved Through the FM Demodulator To Become Additive Output Noise.

Noise on the frequency estimate can be thought as the sum of the true instantaneous frequency and an error term:

$$\hat{\omega}(t) = \omega(t) + e(t).$$

With the noise sources defined, we can define the metrics by which the Vector Readout method of demodulation can be evaluated. The two metrics used here to characterize the noise $e(t)$ will be *mean error*, denoted as m_e, and *mean square error*, denoted as λ_e. The mean error is the time-averaged expected value of the error:

$$m_e = \frac{1}{2T}\int_{-T}^{T} \mathrm{E}[e(t)]\ \mathrm{d}t. \tag{3.3}$$

The mean square error is the time-averaged expected value of the square error:

$$\lambda_e = \frac{1}{2T}\int_{-T}^{T} \mathrm{E}[e^2(t)]\ \mathrm{d}t. \tag{3.4}$$

If $e(t)$ is periodic, then T can be set to the length of the period to obtain the average, and if $e(t)$ is not periodic, then the time length T needs to expand all time, so $T \to \infty$.

The function $\mathrm{E}[\cdot]$ in the above equations denotes the expected value function, and if $e(t)$ is deterministic, then it can be removed since it will not effect the result. On the other hand, if $e(t)$ is a wide-sense stationary random process then the time-averaging integral can be removed since the mean and mean square error are constant. Equations 3.3 and 3.4 cover the most general case when the error $e(t)$ is a non-stationary random process and needs averaged over both time and distribution.

3.2 Vector Readout FM Demodulation Analysis

The Vector Readout method of FM demodulation, as described in Chapter 2, is shown again in block diagram form in Figure 3-5. The delay block has been removed to simplify this analysis since it will not effect the mean or mean square error, and a downsampler has been added as will be explained later. The arctan() function of the CORDIC block is the only non-linear element in this demodulation method. Because the noise sources are assumed to be much smaller than the signal to be demodulated, the arctan() function can be linearized around the error sources to yield good approximations, and with this linearization, we can consider the error sources separately and then sum each of their contributions to the mean and mean square error to produce the final result.

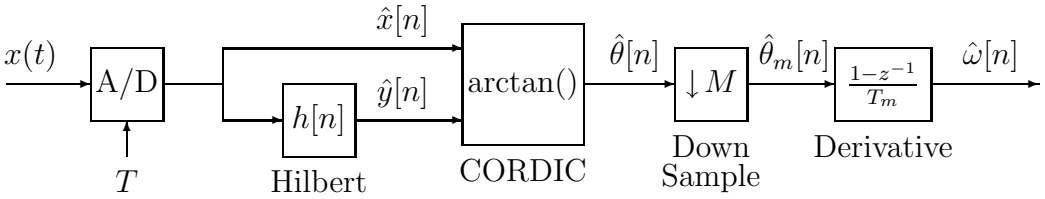


Figure 3-5: Block Diagram of Vector Readout Method of FM Demodulation.

3.2.1 Intrinsic Noise of Vector Readout Method

There are several intrinsic sources of error within the Vector Readout method as shown in Figure 3-5. The input signal $x(t)$ in this case is a pure sine wave that is both frequency and amplitude modulated $x(t) = a(t)\cos\theta(t)$. Initially we assume this signal is band limited so that no aliasing occurs in the A/D conversion, and later we will consider the effects of aliasing*. In addition we assume that the amplitude of $x(t)$ is within the range of the A/D converter, that numbers are stored digitally in fixed-point registers, and that whenever bit widths are given, they represent the number of fractional bits. With this setup, we are prepared to characterize the intrinsic noise sources of the Vector Readout method.

A/D Converter Quantization Noise

The first noise source comes from the quantization of the input signal. Here we use the classic quantization approach and assume that an infinite precision A/D converter maps the continuous-time input $x(t)$ into a discrete-time signal $x[n]$ so that

$$x[n] = x(nT) = a[n]\cos\theta[n],$$

where T is the sampling rate. Then an additive noise source $e_x[n]$ is used to account for quantization as shown in Figure 3-6. This noise source $e_x[n]$ is modeled as a

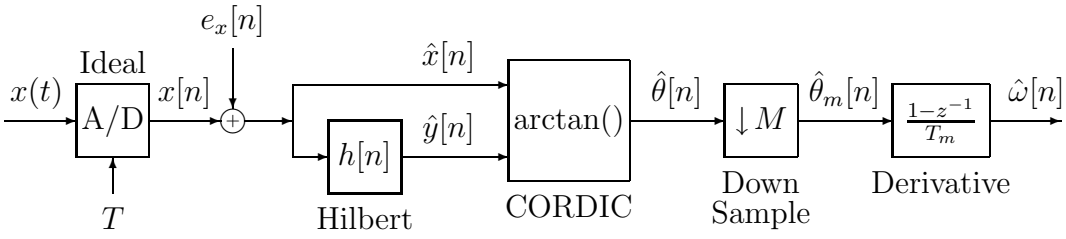


Figure 3-6: A/D Quantization Noise Represented as Additive Noise Source

*See Chapter 2 for a discussion of signal bandwidth and how to avoid aliasing using the Vector Readout method.

zero-mean, white, and uniformly distributed random process with variance

$$\sigma_x^2 = \frac{2^{-2B_x}}{12} \tag{3.5}$$

(where B_x is the number of fractional bits used to store $\hat{x}[n]$). Therefore, the estimate of the position signal, which is labeled $\hat{x}[n]$ in Figure 3-6, is

$$\hat{x}[n] = x[n] + e_x[n].$$

One thing to note is that this noise source looks the same as the White Input Noise case, so their effects will be the same.

The estimate $\hat{x}[n]$, which is the in-phase estimate, is passed through the Hilbert Transform filter to produce the quadrature estimate. If we assume the Hilbert Transform filter is an ideal $-90°$ phase shifter, then the true in-phase signal $x[n]$ goes through to produce a perfect quadrature signal $y[n] = a[n] \sin \theta[n]$. If the noise $e_x[n]$ goes through to produce an output noise $e_y[n]$, then the statistics of $e_y[n]$ can be obtained by considering the characteristics of the Hilbert Transform filter. The Hilbert Transform filter is assumed to be an FIR, anti-symmetric approximation of the ideal Hilbert Transform filter[†]. As a result, the unit sample response is an odd function, as the example Hilbert Transform filter in Figure 3-7 shows. Because the unit sample response is odd, and because all auto-correlation functions are even, when you convolve them, the sample at time zero will always be zero. Thus, the correlation between $e_x[n]$ and $e_y[n]$, which is simply

$$K_{e_x e_y}[m] = h[m] * K_{e_x e_x}[m], \tag{3.6}$$

will always be zero for $m = 0$. This means that samples of the noise in the in-phase channel are uncorrelated with their corresponding samples of the noise in the

[†]For simplicity the impulse response of the Hilbert Transform filter in this analysis is centered at the origin. In reality it is delayed so that the filter is causal. Since the in-phase signal will be delayed to match the delay of the Hilbert Transform filter, the delay is of no consequence to this analysis. The effects of the delay may be of concern for the specific application, but it does not effect the noise statistics.

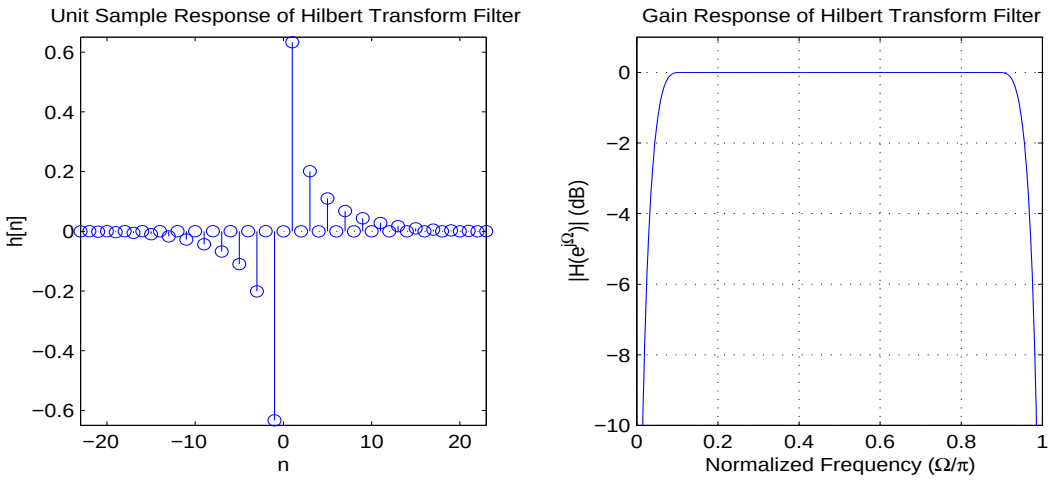


Figure 3-7: Unit Sample and Frequency Response of Hilbert Transform Filter.

quadrature channel. It does not mean they are completely uncorrelated, but samples at the same time are. This fact will be used later in finding the mean and mean square error of the estimator.

Another property of an FIR Hilbert Transform filter is that it approximates an all-pass filter. This can be seen in the frequency response of the sample Hilbert Transform filter shown in Figure 3-7. Because this is the case, the first and second order statistics of $e_y[n]$ will approximately match those of $e_x[n]$, including the fact that $\sigma_x^2 \approx \sigma_y^2$. With $e_y[n]$ characterized, we can express the estimate of the quadrature signal as the sum of the true quadrature signal $y[n]$ and the filtered noise $e_y[n]$,

$$\hat{y}[n] = y[n] + e_y[n],$$

and we can move the noise source $e_x[n]$ downstream as shown in Figure 3-8. This, however, is not a complete characterization of $\hat{y}[n]$ because in reality the Hilbert Transform filter is not ideal and introduces its own quantization noise, and it has gain error.

Hilbert Transformer Quantization Noise

The quantization noise of the Hilbert Transform filter can be isolated to a single place if the filter is implemented in FIR Direct Form (see Reference [7] p.367). When a filter

is implemented in Direct Form, all the arithmetic can be done without truncating or rounding, and it is only the final output which needs rounded if the word length needs reduced. If the output is rounded to B_h fractional bits, then this quantization can be represented as an additive noise source which is zero-mean, white, and uniformly distributed with variance

$$\sigma_h^2 = \frac{2^{-2B_h}}{12} \tag{3.7}$$

as shown in Figure 3-8. Here $e_h[n]$ is the Hilbert Transform filter quantization noise

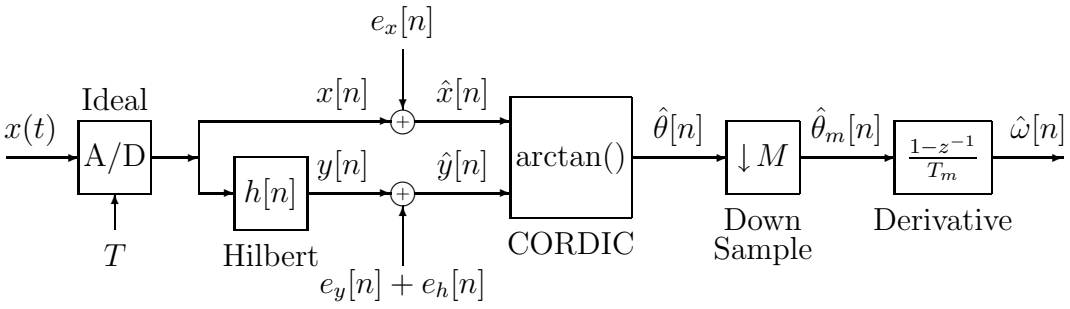


Figure 3-8: Vector Readout FM Demodulation with A/D and Hilbert Transformer Quantization Noise.

source, and so the quadrature signal estimate becomes

$$\hat{y}[n] = y[n] + e_y[n] + e_h[n].$$

Note that $e_h[n]$ is independent of $e_y[n]$ and $e_x[n]$.

Ripple in the Gain Response of Hilbert Transform Filter

The final effect to consider with the Hilbert Transform filter is that its gain response has ripple on it. This ripple can be made arbitrarily small by increasing the number of taps in the filter, but this adds delay which may be of concern. If the optimum or equiripple FIR filter (see Reference [7] p.486) is used to approximate the ideal Hilbert Transform filter, then the ripple will be bounded and the gain of the filter over the bandwidth of interest can be expressed as

$$\left|H(\mathrm{e}^{\mathrm{j}\Omega})\right| = 1 + \epsilon_h$$

where ϵ_h is the ripple. This ripple is not obvious in Figure 3-7 because it is so small, but in Figure 3-9, the pass band is zoomed to show this ripple. This ripple will have

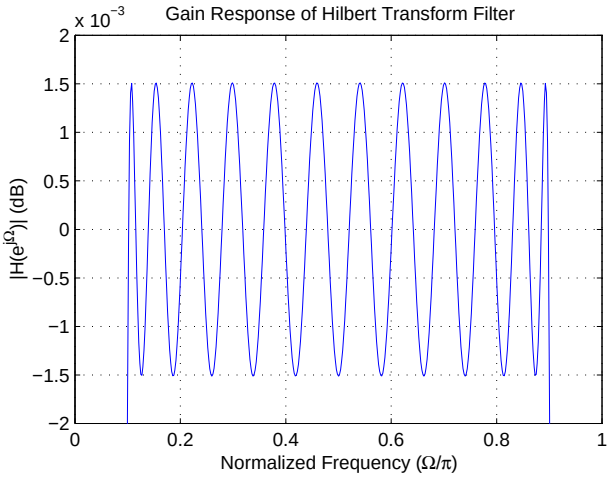


Figure 3-9: Frequency Response of Hilbert Transform Filter Showing Ripple in Pass Band.

negligible effects on the statistics of $e_y[n]$. However, it will effect $y[n]$, which is the true quadrature signal. The result is that our estimate of the quadrature signal is

$$\hat{y}[n] = (1 + \epsilon_h)y[n] + e_y[n] + e_h[n].$$

These errors fully represent the intrinsic errors on the quadrature signal estimate. It was shown in Chapter 2 that the phase response of the Hilbert Transform filter is exactly $-90°$, so the quadrature estimate will have no phase error on it.

To summarize, we have characterized $\hat{x}[n]$ and $\hat{y}[n]$, which are our estimates of the in-phase and quadrature signals. They are

$$\hat{x}[n] = a[n]\cos\theta[n] + e_x[n] \tag{3.8}$$

$$\hat{y}[n] = (1+\epsilon_h)\, a[n]\sin\theta[n] + e_y[n] + e_h[n]. \tag{3.9}$$

Now we are ready to consider how these noise sources propagate through the CORDIC block and onto the phase estimate.

Quantization Noise of CORDIC

As can be seen from Figure 3-5, the in-phase and quadrature estimates are passed into the CORDIC where it produces an angle estimate. The result is that

$$\hat{\theta}[n] = \arctan\left(\frac{\hat{y}[n]}{\hat{x}[n]}\right) + e_c[n] \tag{3.10}$$

where $e_c[n]$ is the quantization noise of the CORDIC. A rough estimate is that the CORDIC produces about 1 bit of fractional precision for each iteration. Thus, the size of $e_c[n]$ can be adjusted by changing the number of iterations in the CORDIC, but its characteristics change drastically depending on how large $e_c[n]$ is compared to the other noise sources on the phase estimate. To see this, consider the case when the noise on the phase measurement is small compared to the quantization noise of the CORDIC. When this is the case, the noise is not well represented as a white random process because it is like sending a saw tooth signal into a A/D converter. However, when the noise on the phase estimate is on the same order or larger than the quantization noise of the CORDIC, it helps to mix things up enough so that the quantization noise looks white. It will generally be desirable that the CORDIC not be the dominant noise source in the system, so for this analysis we assume that $e_c[n]$ is not large compared to the other noise, so it can be represented as a white, zero mean, uniformly distributed random process with variance

$$\sigma_c^2 = \frac{2^{-2N}}{12} \tag{3.11}$$

where N is the number of iterations in the CORDIC algorithm.

Noise Propagated Through the CORDIC

Equation 3.10 shows how the in-phase and quadrature estimates get propagated through the CORDIC. Substituting Equations 3.8 and 3.9 into Equation 3.10 yields

that

$$\hat{\theta}[n] = \arctan\left(\frac{(1+\epsilon_h)\, a[n]\sin\theta[n] \;+\; e_y[n] \;+\; e_h[n]}{a[n]\cos\theta[n] \;+\; e_x[n]}\right) \;+\; e_c[n] \tag{3.12}$$

This non-linear relation is not easy to work with, but using a first order Taylor Series expansion of Equation 3.12 to approximate $\hat{\theta}[n]$, we obtain

$$\hat{\theta}[n] \approx \theta[n] \;+\; \frac{\epsilon_h \sin 2\theta[n]}{2} \;+\; \frac{e_y[n]+e_h[n]}{a[n]}\cos\theta[n] \;-\; \frac{e_x[n]}{a[n]}\sin\theta[n] \;+\; e_c[n]. \tag{3.13}$$

In simulations this approximation has proved to be good (which makes sense because the true in-phase and quadrature signals are the dominant signals), and it is extremely useful because it enables us to treat the CORDIC block as a linear system with respect to the error sources. So we define a new noise source $e_\theta[n]$ which is the sum of all the noise sources in Equation 3.13:

$$e_\theta[n] = \frac{\epsilon_h \sin 2\theta[n]}{2} \;+\; \frac{e_y[n]+e_h[n]}{a[n]}\cos\theta[n] \;-\; \frac{e_x[n]}{a[n]}\sin\theta[n] \;+\; e_c[n] \tag{3.14}$$

This allows us to express the phase estimate as $\hat{\theta}[n] \approx \theta[n] + e_\theta[n]$, and so all the intrinsic noise sources have been moved through the CORDIC as shown in Figure 3-10.

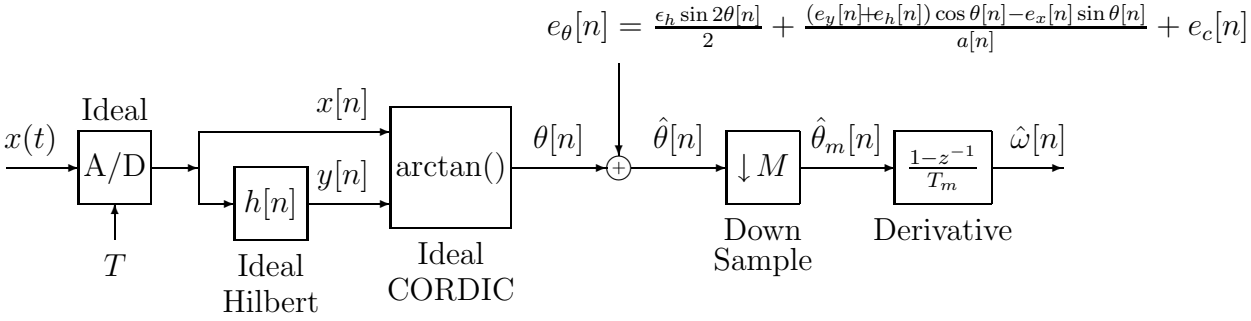


Figure 3-10: A/D Quantization, Hilbert Gain Error, Hilbert Quantization, and CORDIC Quantization All Represented as Single Additive Phase Error.

Noise Propagated Through Downsampler

The downsampler as shown in Figure 3-5 follows the CORDIC, and it reduces the sampling rate by a factor of M. We define a new sampling time T_m which is

$$T_m = MT.$$

The result is that our phase estimate $\hat{\theta}[n]$ becomes

$$\begin{aligned} \hat{\theta}_m[n] &= \hat{\theta}[nM] \\ &= \theta[nM] + e_\theta[nM] \\ &= \theta_m[n] \ + \ e_m[n] \end{aligned}$$

The downsampler will not change the mean or mean square error of the phase estimate, however, it does serve to decorrelate it. This is because the cross-correlation between $e_x[n]$ and $e_y[n]$ equals the unit sample response of the Hilbert Transform filter, as shown by Equation 3.6. So $e_x[n]$ and $e_y[n]$ become less and less correlated as the time difference between them increases because the unit sample response of Hilbert Transform filter (an example of which is shown in Figure 3-7) rolls off and eventually equals zero. Thus, as M increases, the noise sources on the phase estimate decorrelate, and so the rest of this analysis will assume that M is large enough so that all the random processes in $e_\theta[n]$ can be approximated as an uncorrelated.

Another effect of the downsampler is that if M is large enough then the sinusoidal terms in Equation 3.13 get aliased. Therefore the high frequency terms get mapped to low frequency terms, and where they end up will depend on both M and $\omega[n]$, and since $\omega[n]$ is moving by design, it is difficult to track where these high frequency terms will end up when they are aliased. As a result, the rest of this analysis will find the results under worst case scenarios.

Therefore, we express the error terms of the downsampled phase estimate as

$$e_m[n] = \frac{\epsilon_h \sin 2\omega_0 n}{2} \ + \ \frac{e_y[n] + e_h[n]}{a[n]} \cos \omega_0 n \ - \ \frac{e_x[n]}{a[n]} \sin \omega_0 n \ + \ e_c[n] \qquad (3.15)$$

where ω_0 is ω aliased (and $\theta[n] \approx \omega n$). Now it becomes useful to take advantage of linearity in characterizing each term of $e_m[n]$ individually. The mean error is zero, and Table 3.1 summarizes the mean square error for each of the terms in Equation 3.15.

Noise Source	**Mean Square Error** $(\text{rad})^2$	**Error Characterization**
A/D Quantization ($e_x[n]$ and $e_y[n]$)	$\frac{\sigma_x^2}{a^2}$	Noise is white and wide-sense stationary. σ_x^2 is noise variance of A/D (Eq 3.5).
Hilbert Filter Quantization ($e_h[n]$)	$\frac{\sigma_h^2}{2a^2}$	Noise is not wide-sense stationary. $e_m[n] = e_h[n]\cos\omega_0 n$ where ω_0 is ω aliased. σ_h^2 is noise variance of Hilbert filter quantization (Eq 3.7).
CORDIC Quantization ($e_c[n]$)	σ_h^2	Noise is white and wide-sense stationary. σ_h^2 is noise variance of CORDIC quantization (Eq 3.11).
Hilbert Gain Ripple (ϵ_h)	$\frac{\epsilon_h^2}{8}$	$e_m[n] = \frac{\epsilon_h \sin\omega_0 n}{2}$ where ω_0 is 2ω aliased.

Table 3.1: Summary of Intrinsic Noise Sources on the Downsampled Estimate of the Phase.

Differentiator

The output of differentiator as shown in Figure 3-5 is the frequency estimate. It can be expressed as

$$\hat{\omega}[n] = \frac{\hat{\theta}_m[n] - \hat{\theta}_m[n-1]}{T_m} \tag{3.16}$$

If $\omega[n]$ is slowly varying compared to the sampling rate T_m, then a good approximation of the true frequency is

$$\omega[n] \approx \frac{\theta_m[n] - \theta_m[n-1]}{T_m},$$

and we can write the frequency estimate as the sum of the true frequency and noise[‡]:

$$\begin{aligned} \hat{\omega}[n] &= \omega[n] + \frac{e_m[n] - e_m[n-1]}{T_m} \\ &= \omega[n] + e_\omega[n]. \end{aligned}$$

[‡]Here we assume that the differentiation can be performed without any quantization.

In characterizing the error in the frequency estimate, it is helpful to realize this differentiator is an FIR filter with a gain and phase response of

$$\begin{aligned} \left|H(\mathrm{e}^{\mathrm{j}\Omega})\right| &= \frac{2}{T_m}\sin\frac{\Omega}{2} \\ \angle H(\mathrm{e}^{\mathrm{j}\Omega}) &= \frac{\Omega}{2}-\frac{\pi}{2} \quad \text{for } 0<\Omega<\pi. \end{aligned} \tag{3.17}$$

So the power spectral density of the wide-sense stationary parts of the error is

$$\begin{aligned} S_{e_\omega e_\omega}(\mathrm{e}^{\mathrm{j}\Omega}) &= \left|H(\mathrm{e}^{\mathrm{j}\Omega})\right|^2 S_{e_m e_m}(\mathrm{e}^{\mathrm{j}\Omega}) \\ &= \frac{4\sigma_m^2}{T_m^2}\sin^2\frac{\Omega}{2} \\ &= \frac{4}{T_m^2}\left(\frac{\sigma_x^2}{a^2}+\sigma_c^2\right)\sin^2\frac{\Omega}{2} \end{aligned}$$

and substituting in for σ_x^2, and σ_c^2 from Equations 3.5, and 3.11, the the variance and the power spectral density of the frequency noise estimate due to A/D converter quantization and CORDIC quantization are:

$$\lambda_e = \frac{4^{-B_x}}{6a^2T_m^2} + \frac{4^{-N}}{6T_m^2}$$

$$S_{ee}(\mathrm{e}^{\mathrm{j}\Omega}) = \frac{1}{3T_m^2}\left(\frac{4^{-B_x}}{a^2} + 4^{-N}\right)\sin^2\frac{\Omega}{2}$$

These results are summarized in Table 3.2, and they have been verified by simulation by generating a pure cosine wave ($x[n]$) and a pure sine wave ($y[n]$) which were quantized to 11 fractional bits, passed through an ideal arctan() function, quantized to match the quantization of an $N = 16$ CORDIC algorithm, downsampled by 40, and differentiated according to Equation 3.16. The resultant power spectral density and calculated power spectral density are shown in Figure 3-11.

The Hilbert Transform filter quantization is not easily characterized since it is not wide-sense stationary. Using linearity we look at just the Hilbert Transform filter quantization and set $e_m[n] = \frac{e_h[n]}{a[n]}\cos\omega_0 n$. The differentiator will not change the mean, but it will double the mean square error since $e_h[n]$ is assumed to be white.

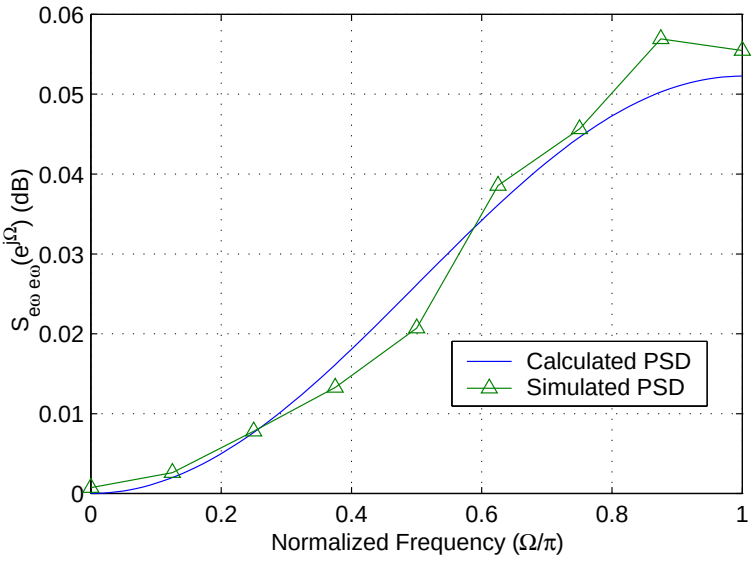


Figure 3-11: Calculated and Simulated Power Spectral Density of Noise Due to A/D and CORDIC Quantization on the Frequency Estimate.

The result is that

$$\begin{aligned} e_\omega[n] &= \frac{e_h[n]\cos\omega_0 n - e_h[n-1]\cos\omega(n-1)}{a[n]T_m} \\ \lambda_e &= \frac{\sigma_h^2}{2a^2T_m^2} \end{aligned}$$

This result has also been verified via simulation and is summarized in Table 3.2.

The last intrinsic error source to consider is the Hilbert gain ripple term of $e_m[n]$ of Equation 3.15. We assume $e_m[n] = \frac{\epsilon_h \sin\omega_0 n}{2}$ where ω_0 is 2ω aliased, and find how it effects $e_\omega[n]$. Using the frequency response of the differentiator as expressed in Equation 3.17, the error comes

$$e_\omega[n] = \frac{\epsilon_h}{T_m}\sin\frac{\omega_0}{2}\cos\omega_0\left[n-\frac{1}{2}\right].$$

This has a mean of zero, so $m_e = 0$, and the mean square error due to the Hilbert Transform filter gain ripple is bound by

$$\lambda_e \leq \frac{\epsilon_h^2}{2T_m^2}.$$

These results have also been verified by simulation and are summarized in Table 3.2.

Environmental Considerations

The only Intrinsic Noise source that depends on the environment is the A/D converter quantization noise. Temperature can affect the offset of the A/D converter. Suppose the offset of the A/D converter is a_0, then the in-phase signal estimate becomes $\hat{x}[n] = x[n] + a_0$. For this application the A/D converter is followed by a digital bandpass filter to help remove this offset, so it will not be of concern. Without this bandpass filter, it can be shown that this offset will cause a zero mean error source on the frequency estimate that has a mean square error that is bound by $\lambda_e \leq \frac{2a_0^2}{a^2 T_m^2}$. The gain of the A/D converter is also a function of temperature. This will not effect the frequency of the signal, though, so it will not have any effects on the acceleration estimate.

3.2.2 Input Noise Sensitivity

White Input Noise Sensitivity

White noise on the input signal behaves just like the A/D converter quantization noise, and since the arctan() function performed by the CORDIC algorithm was linearized, the effects of the white input noise add to those of the Intrinsic Noise Sources. Once again we assume an input with an amplitude a, and if zero-mean, white noise having variance σ_v^2 is added to that input prior to demodulation, then it will result as zero mean noise on the output having a spectral density of

$$S_{ee}(\mathrm{e}^{\mathrm{j}\Omega}) = \frac{4\sigma_v^2}{a^2 T_m^2} \sin^2 \frac{\Omega}{2}$$

and a mean square error of

$$\lambda_e = \frac{2\sigma_v^2}{a^2 T_m}.$$

This was derived in Section 3.2.1.

Harmonic Input Noise

To find the effects of harmonic input noise, once again we rely on the linearization of the arctan() function so that the harmonic input noise can simply be added to the other results.

After being processed by the Hilbert Transform filter, $\hat{x}[n]$ and $\hat{y}[n]$ are

$$\begin{aligned}
\hat{x}[n] &= a[n]\cos\theta[n] \;+\; a_1[n]\cos(k_1\theta[n]+\phi_1) \;+\; a_2[n]\cos(k_2\theta[n]+\phi_2) \;+\; \cdots \\
\hat{y}[n] &= a[n]\sin\theta[n] \;+\; a_1[n]\sin(k_1\theta[n]+\phi_1) \;+\; a_2[n]\sin(k_2\theta[n]+\phi_2) \;+\; \cdots
\end{aligned}$$

Then $\hat{\theta}[n]$ can be found as

$$\begin{aligned}
\hat{\theta}[n] &= \arctan\left(\frac{y[n]}{x[n]}\right) \\
&= \arctan\left(\frac{a[n]\cos\theta[n] \;+\; a_1[n]\cos(k_1\theta[n]+\phi_1) \;+\; a_2[n]\cos(k_2\theta[n]+\phi_2) \;+\; \cdots}{a[n]\sin\theta[n] \;+\; a_1[n]\sin(k_1\theta[n]+\phi_1) \;+\; a_2[n]\sin(k_2\theta[n]+\phi_2) \;+\; \cdots}\right)
\end{aligned}$$

This can be Taylor Series expanded about $a_1, a_2, \cdots$ to yield

$$\hat{\theta}[n] \approx \theta[n] \;+\; \frac{a_1}{a}\sin((k_1-1)\theta[n]+\phi_1) \;+\; \frac{a_2}{a}\sin((k_2-1)\theta[n]+\phi_2) \;+\; \cdots$$

So

$$e_\theta[n] = \frac{a_1}{a}\sin((k_1-1)\theta[n]+\phi_1) \;+\; \frac{a_2}{a}\sin((k_2-1)\theta[n]+\phi_2) \;+\; \cdots,$$

and then it is downsampled by M to produce

$$\begin{aligned}
e_m[n] &= e_\theta[nM] \\
&= \frac{a_1}{a}\sin((k_1-1)\theta[nM]+\phi_1) \;+\; \frac{a_2}{a}\sin((k_2-1)\theta[nM]+\phi_2) \;+\; \cdots \\
&= \frac{a_1}{a}\sin\omega_1 n \;+\; \frac{a_2}{a}\sin\omega_2 n \;+\; \cdots
\end{aligned}$$

where the phase terms ϕ_x's have been dropped because they do not affect the averages and where ω_1, ω_2, ... are the aliased frequencies. Because the true frequency ω moves, it is not easy to track where the harmonics get mapped when they are aliased, so we

will once again find the worst case result. When $e_m[n]$ goes through the digital differentiator, it becomes

$$e_\omega[n] = \frac{2a_1}{aT_m} \sin\frac{\omega_1}{2} \cos\omega_1 \left[n-\frac{1}{2}\right] \; + \; \frac{2a_2}{aT_m} \sin\frac{\omega_2}{2} \cos\omega_2 \left[n-\frac{1}{2}\right] \; + \; \cdots$$

Therefore, the mean error of the frequency estimate is

$$m_e = 0$$

and the mean square error is bound by

$$\lambda_e \; \leq \; \frac{2}{a^2 T_m^2} \left(a_1^2 \; + \; a_2^2 \; + \; \cdots\right).$$

Summary

The results of this analysis are summarized in Table 3.2. It reveals that the Vector Readout method of FM demodulation is capable of a precision FM demodulation that has no mean error due to either Intrinsic or Input Noise. The mean square error of the Intrinsic Noise sources can be made arbitrarily small by increasing the amount of hardware and delay, and the Input Noise sources can also be decreased significantly if one filters between the A/D converter and the Hilbert Transform filter. In light of these characteristics and the facts that this method can be efficiently implemented, that it has no significant temperature dependence, and that it simultaneously produces an amplitude demodulation, this method of FM demodulation is a great candidate for this application.

3.3 FM Demodulation Comparison

It is of interest to know how the Vector Readout method of FM demodulation compares to other methods that could be used. Following is a discussion of some of the problems associated with using other methods for this application.

Noise Source	**Conditions**	**Mean Error**	**Mean Square Error** $(\text{rad/s})^2$	**Error Characterization**
A/D Converter Quantization	Input has amplitude a and gets quantized to B_x fractional bits.	0	$\frac{4^{-B_x}}{6a^2T_m^2}$	$S_{ee}(\mathrm{e}^{\mathrm{j}\Omega}) = \frac{4^{-B_x}}{3a^2T_m^2}\sin^2\frac{\Omega}{2}$
Hilbert Transform Quantization	Output register of the Hilbert filter is quantized to B_h fractional bits.	0	$\frac{4^{-B_h}}{12a^2T_m^2}$	$e[n] = \frac{e_h[n]\cos\omega_0 n - e_h[n-1]\cos\omega_0(n-1)}{aT_m}$ where ω_0 is ω aliased and $e_h[n]$ is a B_h bit quantization noise process. $e[n]$ is not wide sense stationary.
CORDIC Quantization	The CORDIC algorithm is iterated N times.	0	$\frac{4^{-N}}{6T_m^2}$	$S_{ee}(\mathrm{e}^{\mathrm{j}\Omega}) = \frac{4^{-N}}{3T_m^2}\sin^2\frac{\Omega}{2}$
Hilbert Gain Error	Gain Response has worse case gain ripple of ϵ_h.	0	$\leq \frac{\epsilon_h^2}{2T_m^2}$	$e[n] = \frac{\epsilon_h}{T_m}\sin\frac{\omega_0}{2}\cos\omega_0\left[n-\frac{1}{2}\right]$ where ω_0 is 2ω aliased.
A/D Bias	Input has amplitude a and A/D has an offset a_0.	0	$\leq \frac{2a_0^2}{a^2T_m^2}$	$e[n] = -\frac{2a_0}{aT_m}\sin\frac{\omega_0}{2}\cos\omega_0\left[n-\frac{1}{2}\right]$ where ω_0 is ω aliased.
White Input Noise	Input has amplitude a and is corrupted with zero-mean white noise with power σ^2.	0	$\frac{2\sigma^2}{a^2T_m^2}$	$S_{ee}(\mathrm{e}^{\mathrm{j}\Omega}) = \frac{4\sigma^2}{a^2T_m^2}\sin^2\frac{\Omega}{2}$
Harmonic Input Noise	Input has amplitude a and is corrupted with harmonics: $a_1\cos(k_1\omega n + \phi_1) + a_2\cos(k_2\omega n + \phi_2) + \cdots$.	0	$\leq \frac{2}{a^2T_m^2}\left(a_1^2 + a_2^2 + \cdots\right)$	$e[n] = \frac{2}{aT_m}\left(a_1\sin\frac{\omega_1}{2}\cos\omega_1 n + a_2\sin\frac{\omega_2}{2}\cos\omega_2 n + \cdots\right)$ where ω_x is $(k_x-1)\omega$ aliased.

Table 3.2: Summary of Frequency Estimation Errors Using Vector Readout Method for FM Demodulation.

Phase-Locked Loop (PLL)

The input into the VCO (Voltage Controlled Oscillator) of a PLL contains the instantaneous frequency of the input signal, and this can be used as the FM demodulation. It is difficult, however, to make the VCO environmentally stable over temperature. This means that this is not a good method to use to perform a low bandwidth FM demodulation. In addition, for this accelerometer, the PLL needs to have a wide bandwidth to allow for the possibly large frequency swings which can result under extreme accelerations. These frequency shifts will cause dynamics in the FM demodulation as the PLL tries to track the frequency. Furthermore, the ability of the PLL to track a wide bandwidth signal is inversely related to its ability to reject noise.

Differentiate and Envelope Detect

This method relies on the fact that differentiating a sine wave amplitude modulates the frequency of the signal onto the signal. Then the frequency can be determined using an amplitude demodulation method. There are several problems with this method for this application. First is that differentiation is a high-pass filter, so high-frequency noise and harmonics get amplified. Second is that it requires a stable amplitude on the carrier signal, and for this accelerometer, this is not the case under a changing acceleration as is shown in Chapter 6. Third is that it is not as environmentally stable as the Vector Readout method in that a changing temperature will change the gain of both the amplitude signal and the differentiator.

Counter FM Demodulation Analysis

If one counts the zero crossings of the output signal and divides that by the time it takes to count them, then a first order approximation of the frequency can be obtained. This, however, requires a high sampling rate for precision since you can only count to within half a cycle of the carrier. This can be a good method for extremely low bandwidth signals because the count can be averaged over many cycles of the carrier and thus reducing the precision issue, but the signal must be of low

bandwidth for this to yield useful results.

Arcsine FM Demodulation Analysis

It is possible to improve the Counter method by sampling the output signal at a fixed rate and taking the arcsin() of the signal. Then adjacent samples can be subtracted and divided by the change in time to get a first order approximation of the frequency. In addition, for small values of x, $\arcsin(x) \approx x$, which makes this technique even easier. However, this requires a stable amplitude, and the amplitude is temperature dependent, which makes this method environmentally sensitive.

3.4 Conclusion

The Vector Readout method of FM demodulation is the most logical method to use for this application since it meets all the desired criteria of being a high-precision, environmentally stable demodulation method. Furthermore, as will be shown in Chapter 5, the Vector Readout method also yields a high-precision and stable AM demodulation at the same time as the FM demodulation, and since it requires no additional time or hardware, this gives the Vector Readout method one additional large advantage over other methods that could be used. Thus, for all of the above reasons, we chose to use the Vector Readout for both the FM and AM demodulation method for this application.

With the FM demodulation method selected and characterized, in the next Chapter we turn to finding the best way to turn the frequency estimate into an acceleration estimate. We will then use Table 3.2 to pick the bit widths, the amount of tolerable Hilbert Transform Filter gain ripple, and the amount of tolerable input noise.

Chapter 4

Mapping Frequency To Acceleration

In Chapter 1 we saw that the frequency estimate needs mapped to an acceleration estimate, and in Chapter 3 we characterized the noise of the frequency estimator using the Vector Readout method. This chapter determines the best method for mapping the frequency estimate to an acceleration estimate based on the characteristics of the noise and the sensor.

4.1 Compensation Methods

In Chapter 1 we showed that the oscillator frequency ω and the acceleration g are related by

$$m\omega^2 = k_1 + k_g g, \tag{4.1}$$

where m is the mass, k_1 is the spring stiffness, and k_g is the scale factor between acceleration and spring stiffness. This can be expressed in an alternate form

$$\omega^2 = \omega_0^2 + \gamma g, \tag{4.2}$$

where ω_0 is the zero input frequency and γ is the scale factor from acceleration to square frequency. Under this first order model if γ, ω_0, and ω are known exactly, then

we can obtain an exact estimate of the true acceleration g by building a compensator that takes in γ, ω_0, and ω and returns g according the relation obtained by solving Equation 4.2 for g.

In reality, though, obtaining an acceleration estimate is much more difficult for several reasons. One is that the relationship between the oscillation frequency and acceleration is determined by a very complicated mechanical and electrical system and is not completely captured by the first order model of Equation 4.2. Another is that we do not know the parameters ω_0 and γ exactly and if they change over time we have no way of estimating or knowing it. Lastly, we do not know the oscillation frequency ω exactly but only have an estimate of it which is obtained via frequency demodulation and is corrupted with noise.

Thus, a good method of turning the frequency estimate into an acceleration estimate needs developed. For the Vibratory Accelerometer, the compensator that turns the frequency estimate into an acceleration estimate will be a polynomial function which is parameterized in a calibration step. This calibration step will be performed individually on each accelerometer and involves stepping each accelerometer through known acceleration levels while obtaining and averaging the resultant frequency estimates for each step. Then coefficients from a least-squares polynomial curve fit can be obtained which map the acquired average frequency estimates back to the known acceleration levels, and these coefficients can then be set in each accelerometer. Thus the compensator is a generic polynomial function $f(\omega)$ of the form

$$\hat{g} = f(\hat{\omega}) \; = \; \nu_0 \; + \; \nu_1\hat{\omega} \; + \; \nu_2\hat{\omega}^2 \; + \; \cdots \; + \; \nu_N\hat{\omega}^N,$$

where the ν terms are determined for each accelerometer via calibration, $\hat{\omega}$ is the frequency estimate, N is the order of the compensator and is set by hardware constraints and precision limitations, and $\hat{g}$ is the acceleration estimate.

This method of compensation is good in that it can capture higher order relations between the frequency and acceleration not captured by Equation 4.2, but its limitation is that it assumes that g is the only changing variable that effects the fre-

quency. If ω_0, γ, or higher order parameters relating frequency and acceleration (such as oscillation amplitude) change over time then errors will result in the g estimate. Furthermore, this method assumes that the mean frequency estimate over a given g level captures the complete frequency estimate, and therefore the higher order noise characteristics are neglected. If these are not dealt with they can have undesirable effects as will be shown later.

The compensation procedure, however, is actually more complicated than previously stated, and the reason is that the Vibratory Accelerometer produces two frequency estimates as a differential signal and the acceleration estimate needs to be a single signal. As a result, the compensation can be performed in different places depending on when and how this differential signal is collapsed into a single signal. Figure 4-1 shows three different topologies which could be used to turn the differential frequency estimate into a single acceleration estimate, and following is a description of each method.

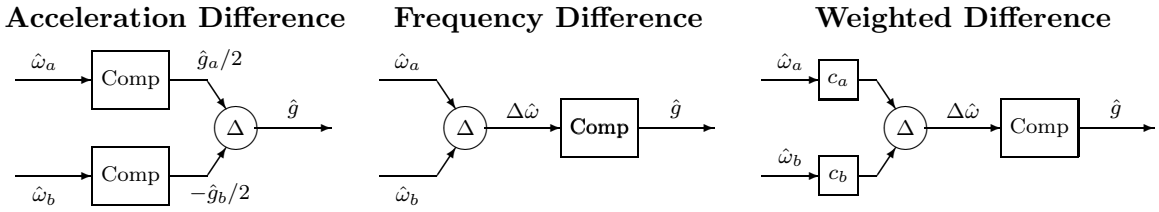


Figure 4-1: Different Compensation Methods

Acceleration Difference: This method obtains the differential acceleration estimates and then subtracts them to obtain a single acceleration estimate. This requires compensating both channels of the differential frequency signal.

Frequency Difference: This method subtracts the frequency estimates first to collapse the differential signal into a single signal and then compensates to obtain the acceleration estimate. This requires a single compensator.

Weighted Difference: Since the zero-input frequencies of each channel are separated to avoid frequency locking problems, it is possible to weight each frequency

estimate prior to differencing. The weights are chosen so that the differenced signal $\Delta\hat{\omega}$ is zero whenever the input g is zero.

4.1.1 First Order Compensators

In order to evaluate which compensation method to use, we must fix the compensator coefficients (which will be fixed in real life via calibration) and then we can see how sensitive each method is to the drawbacks listed earlier. In order to determine and fix the compensator coefficients, we assume that the first order model of Equation 4.2 is exact so that the frequency and acceleration relationship for each channel is

$$\omega_a^2 = \omega_{0a}^2 + \gamma g \tag{4.3}$$

$$\omega_b^2 = \omega_{0b}^2 - (1+\epsilon)\gamma g, \tag{4.4}$$

where the a and b subscripts denote channel A and channel B parameters and ϵ represents the difference in scale factor which will exist between the two channels (to ease the algebra in the future we will define and use $\epsilon_1 = 1+\epsilon$ and $\epsilon_2 = 2+\epsilon$). We then assume that ω_{0a}, ω_{0b}, ϵ, and γ do not change over time and that we can measure ω_a and ω_b exactly (no noise on the frequency estimates). Under these conditions, we can determine exactly the coefficients that would be obtained via calibration by solving the frequency and acceleration relations of Equations 4.3 and 4.4 under the constraints imposed by each of the three compensation methods. There are two reasons why these coefficients, which will be called the *First Order Coefficients*, are good approximations to the coefficients that will be obtained in reality when the accelerometer is calibrated. One is that the relations of Equations 4.3 and 4.4 are good approximations of the true system, and the other is the frequency estimate has a zero mean error. Thus we turn to finding the First Order Coefficients for the three compensation methods.

Acceleration Difference First Order Coefficients

The individual frequency estimates for each channel can be found by solving Equations 4.3 and 4.4 for g, which is

$$\begin{aligned} g_a &= \frac{\omega_a^2 - \omega_{0a}^2}{\gamma} \\ -g_b &= \frac{\omega_b^2 - \omega_{0b}^2}{\gamma\epsilon_1}. \end{aligned}$$

So the compensator coefficients for each channel are

$$\begin{aligned} \nu_{0a} &= -\omega_{0a}^2/\gamma & \qquad \nu_{0b} &= -\omega_{0b}^2/\gamma\epsilon_1 \\ \nu_{1a} &= 0 & \nu_{1b} &= 0 \\ \nu_{2a} &= 1/\gamma & \nu_{2b} &= 1/\gamma\epsilon_1 \\ \nu_{3a} &= 0 & \nu_{3b} &= 0 \\ \vdots & \vdots & \vdots & \vdots \end{aligned}$$

Frequency Difference First Order Coefficients

This method of compensation first subtracts the frequency estimates of each channel and then finds the resulting acceleration measurement. Mathematically, the frequency difference is

$$\begin{aligned} \Delta\omega &= \omega_a - \omega_b \\ &= \sqrt{\omega_{0a}^2 + \gamma g} - \sqrt{\omega_{0b}^2 - \epsilon_1\gamma g}. \end{aligned}$$

Solving this for g and defining $\Gamma = \epsilon_2(\omega_{0a}^2\epsilon_1 + \omega_{0b}^2)$ yields

$$g = \frac{\epsilon_2(\omega_{0b}^2 - \omega_{0a}^2) + 2\Delta\omega\sqrt{\Gamma - \epsilon_1\Delta\omega^2} - \epsilon\Delta\omega^2}{\gamma\epsilon_2^2}.$$

So Taylor Series expanding this about $\Delta\omega$ gives the first order coefficients as

$$\begin{aligned}
\nu_0 &= \frac{\omega_{0b}^2-\omega_{0a}^2}{\gamma\epsilon_2} & \nu_5 &= \frac{-\epsilon_1^2\Gamma^{-3/2}}{4\gamma\epsilon_2^2} \\
\nu_1 &= \frac{2\Gamma^{1/2}}{\gamma\epsilon_2^2} & \nu_6 &= 0 \\
\nu_2 &= \frac{-\epsilon}{\gamma\epsilon_2^2} & \nu_7 &= \frac{-\epsilon_1^3\Gamma^{-5/2}}{8\gamma\epsilon_2^2} \\
\nu_3 &= \frac{-\epsilon_1\Gamma^{-1/2}}{\gamma\epsilon_2^2} & \nu_8 &= 0 \\
\nu_4 &= 0 & \vdots & \vdots
\end{aligned}$$

Weighted Difference First Order Coefficients

This compensation method is similar to the Frequency Difference method except that it weights the frequency estimates before subtracting them. These weights are selected so that $\Delta\omega = 0$ for $g = 0$. Mathematically, this is

$$\Delta\omega = c_a\omega_a - c_b\omega_b$$

$$c_a\omega_{0a} - c_b\omega_{0b} = 0.$$

If we define $\omega_0 = c_a\omega_{0a} = c_b\omega_{0b}$, then the frequency difference can be expressed as*

$$\Delta\omega \;=\; \omega_0\sqrt{1+\frac{\gamma}{\omega_{0a}^2}g} \;-\; \omega_0\sqrt{1-\frac{\gamma}{\omega_{0b}^2}g}.$$

So we see that weighting the frequencies has the effect of giving both channels the same zero-input frequency and of giving them different scale factors. Solving for g to obtain an estimate of the acceleration yields

$$g = \frac{2\omega_{0a}^2\omega_{0b}^2\Delta\omega\sqrt{\omega_0^2(\omega_{0a}^2+\omega_{0b}^2)^2-\omega_{0a}^2\omega_{0b}^2\Delta\omega^2} \;+\; \omega_{0a}^2\omega_{0b}^2(\omega_{0b}^2-\omega_{0a}^2)\Delta\omega^2}{\gamma\omega_0^2(\omega_{0a}^2+\omega_{0b}^2)^2}$$

*ϵ has been dropped because it can be absorbed into ω_{ob}.

Taylor Series expanding this about $\Delta\omega$ yields the first order coefficients

$$\begin{aligned}
\nu_0 &= 0 & \nu_5 &= \frac{-\omega_{0a}^6\omega_{0b}^6}{4\gamma\omega_0^5(\omega_{0a}^2+\omega_{0b}^2)^5} \\
\nu_1 &= \frac{2\omega_{0a}^2\omega_{0b}^2}{\gamma\omega_0(\omega_{0a}^2+\omega_{0b}^2)} & \nu_6 &= 0 \\
\nu_2 &= \frac{\omega_{0a}^2\omega_{0b}^2(\omega_{0b}^2-\omega_{0a}^2)}{\gamma\omega_0^2(\omega_{0a}^2+\omega_{0b}^2)^2} & \nu_7 &= \frac{-\omega_{0a}^8\omega_{0b}^8}{8\gamma\omega_0^7(\omega_{0a}^2+\omega_{0b}^2)^7} \\
\nu_3 &= \frac{-\omega_{0a}^4\omega_{0b}^4}{\gamma\omega_0^3(\omega_{0a}^2+\omega_{0b}^2)^3} & \nu_8 &= 0 \\
\nu_4 &= 0 & \vdots & \quad \vdots
\end{aligned}$$

4.1.2 Spring Stiffness Variation

With the first order coefficients defined for the three compensation methods, we turn to evaluating each method to various types of parameter variation. We start by considering spring stiffness variation by supposing that after calibrating, the zero input spring stiffness changes such that

$$m\omega_a^2 = k_{1a} + k_1 + k_g g \tag{4.5}$$

$$m\omega_b^2 = k_{1b} \pm k_1 - \epsilon_1 k_g g. \tag{4.6}$$

where k_1 is the change in spring stiffness such that when the $\pm$ sign in Equation 4.6 is '+' it corresponds to a common mode spring stiffness change and when it is '$-$' it corresponds to differential mode change. By defining $\omega_1^2 = k_1/m$, Equations 4.5 and 4.6 can be expressed as

$$\omega_a^2 = \omega_{0a}^2 + \omega_1^2 + \gamma g \tag{4.7}$$

$$\omega_b^2 = \omega_{0b}^2 \pm \omega_1^2 - \epsilon_1\gamma g. \tag{4.8}$$

This changing spring stiffness results in a new zero input frequency which is $\sqrt{\omega_{0a}^2+\omega_1^2}$ for channel A and $\sqrt{\omega_{0b}^2 \pm \omega_1^2}$ for channel B. Solving Equations 4.7 and 4.8 for ω_a and ω_b enables us to send the new constraints imposed by the spring stiffness variation through the compensators to see how each one handles it. The results are summarized in Table 4.1 where they have been approximated when necessary by Taylor Series

Acceleration Estimate With Spring Stiffness Variation

	Common Mode	Differential Mode
Acceleration Difference	$g + \frac{k_1 \epsilon}{2 k_g \epsilon_1}$	$g + \frac{k_1}{k_g}$
Frequency Difference	$g + \frac{k_1(\omega_{0b} - \omega_{0a})}{k_g(\omega_{0b} + \omega_{0a})}$	$g + \frac{k_1}{k_g}$
Weighted Difference	$g + \frac{k_1(\omega_{0b}^2 - \omega_{0a}^2)}{k_g(\omega_{0b}^2 + \omega_{0a}^2)}$	$g + \frac{k_1}{k_g}$

Table 4.1: Summary of Spring Stiffness Variation Results.

expanding about g and k_1.

The results of Table 4.1 show that both a common and differential mode change spring stiffness results in a bias on the acceleration estimate. In Figure 4-2 the bias resulting from a common mode spring stiffness change is plotted using the nominal

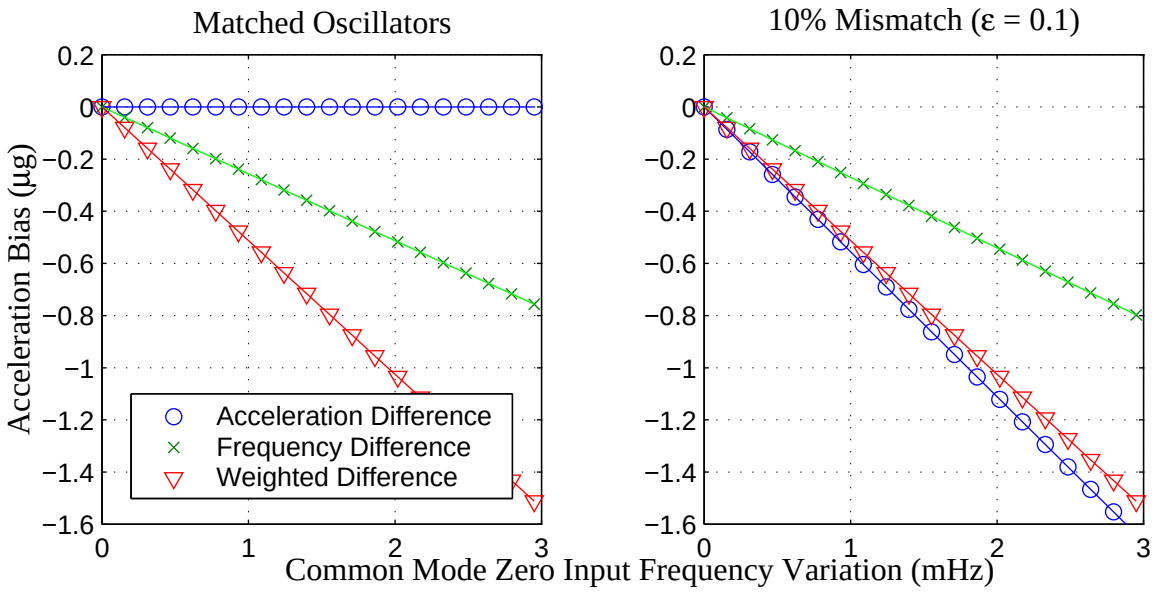


Figure 4-2: Spring Stiffness Common Mode Variation with Matched Parameters.

values[†] of the VA for the cases when the oscillators are matched ($\epsilon = 0$) and when they are matched to 10% ($\epsilon = 0.1$). The zero input frequency variation $\left(\sqrt{\omega_{0a}^2 + \omega_1^2} - \omega_{0a}\right)$ is plotted on the x-axis. Notice that the Acceleration Difference method is the only compensation method that is sensitive to mismatches, thus the Acceleration Difference method is the only method that will perform better as the sensor is built better. Note that for the Weighted Difference method if the zero input frequency drifts in a common mode by 2 mHz it will cause a 1 μg bias. On the other hand, the Frequency Difference method can tolerate a drift of about 3.5 mHz before a 1 μg bias results. Notice that all three methods have approximately the same Differential Mode sensitivity and that it is not related to how well the oscillators are matched.

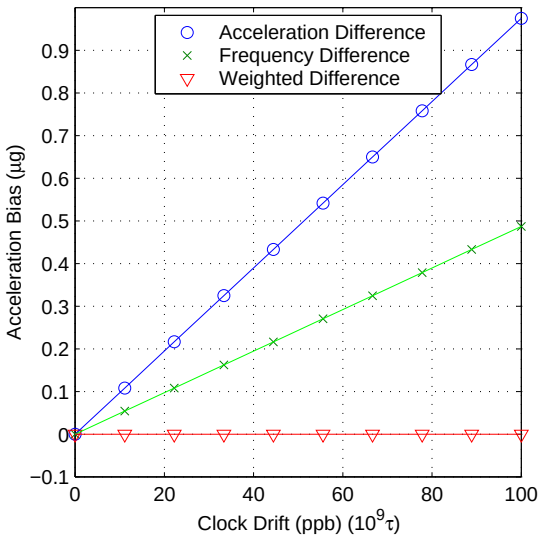
4.1.3 Sampling Clock Drift

Suppose the sampling clock drifts after calibration such that the sampling time becomes $T' = (1+\tau)T$ where T is the sampling time used during calibration. Since the frequency estimate is generated using this clock as a reference, the frequency estimate from each channel will also change proportionally to $(1+\tau)$ so that $\hat{\omega}' = (1+\tau)\hat{\omega}$. If $g = 0$ then sending these frequency estimates into each compensator results in a bias in the acceleration estimate. The resulting acceleration bias for each method is listed in Table 4.2. The corresponding figure shows that if the Acceleration Difference method is used to compensate then the clock must be held with 100 ppb in order that the acceleration bias not exceed 1 μg. The Weighted Difference method is clearly the least sensitive to clock drift in that it causes no bias error.

[†]The nominal values used throughout this analysis are $\omega_{0a} = 2\pi \cdot 20,000$ rad/s, $\omega_{0b} = 2\pi \cdot 19,000$ rad/s. The scale factor for the channel A is assumed to be 100 Hz/g, and this makes $\gamma = 157.9 \times 10^6$ rad^2/s^2/g.

Acceleration Difference	$\frac{\tau(\omega_{0a}^2 - \omega_{0b}^2)}{\gamma\epsilon_1}$
Frequency Difference	$\frac{2\tau\omega_{0a}\omega_{0b}(\omega_{0a} - \omega_{0b})}{\gamma(\omega_{0a} + \omega_{0b})}$
Weighted Difference	0

Table 4.2: Acceleration Bias Resulting From Sampling Clock Drift.



4.1.4 Noisy Frequency Estimates

Suppose that the frequency estimates from each channel are corrupted with noise such that

$$\hat{\omega}_a[n] = \omega_a[n] + e_{\omega a}[n] \tag{4.9}$$

$$\hat{\omega}_b[n] = \omega_b[n] + e_{\omega b}[n] \tag{4.10}$$

where $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$ are zero-mean random processes with auto-covariance functions $R_{\omega_a\omega_a}[m]$ and $R_{\omega_b\omega_b}[m]$ and variances σ_a^2 and σ_b^2. We now turn to see how this noise effects the acceleration estimate and the velocity estimate (which will be obtained by integrating the acceleration estimate). The two metrics used to compare the compensation methods are bias on the acceleration estimate and random walk on the velocity estimate.

To do this, we first find the error on the acceleration estimate to second order for each compensation method. Using Taylor Series expansions and setting $g = 0$, we find:

Acceleration Difference Acceleration Estimation Error:

$$e_g[n] \approx \frac{1}{\gamma}\left(\omega_{0a}e_{\omega_a}[n] - \omega_{0b}e_{\omega_b}[n]\frac{1}{\epsilon_1}\right) + \frac{1}{2\gamma}\left(e_{\omega_a}^2[n] - e_{\omega_b}^2[n]\frac{1}{\epsilon_1}\right) \tag{4.11}$$

Frequency Difference Acceleration Estimation Error:

$$e_g[n] \approx \frac{2\omega_{0a}\omega_{0b}}{\gamma(\omega_{0a}\epsilon_1 + \omega_{0b})}(e_{\omega_a}[n] - e_{\omega_b}[n]) \; + \; \frac{\omega_{0b}^3 - \omega_{0a}^3\epsilon_1^2}{\gamma(\omega_{0a}\epsilon_1 + \omega_{0b})^3}(e_{\omega_a}[n] - e_{\omega_b}[n])^2 \tag{4.12}$$

Weighted Difference Acceleration Estimation Error:

$$e_g[n] \approx \frac{2\omega_{0a}\omega_{0b}}{\gamma(\omega_{0a}^2 + \omega_{0b}^2)}(\omega_{0b}e_{\omega_a}[n] - \omega_{0a}e_{\omega_b}[n]) + \frac{\omega_{0b}^2 - \omega_{0a}^2}{\gamma(\omega_{0b}^2 + \omega_{0a}^2)^2}(\omega_{0b}e_{\omega_a}[n] - \omega_{0a}e_{\omega_b}[n])^2 \tag{4.13}$$

Acceleration Bias Comparison and Sensitivity

From Equations 4.11, 4.12, and 4.13 it is straight forward to find the mean error, and they are listed and compared in Table 4.3 and Figure 4-3. These show that if the noise characteristics of $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$ are constant then a fixed bias can result in the acceleration estimate. A fixed bias, however, is not of concern because it can be corrected either by adjusting the coefficients of the compensator to have an equivalent second stage or by making the application correct for it. If the noise characteristics of $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$ change after calibration, however, the resulting change in bias cannot be corrected. Thus in Table 4.3 and Figure 4-3 the common and differential mode bias sensitivity are presented. These are defined by considering the effects when the noise power of $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$ move commonly and differentially. Suppose the noise power in channel A becomes $\sigma_a^2 + \sigma^2$ and noise power in channel B becomes $\sigma_b^2 + \sigma^2$. These changes will change the mean error of the acceleration estimate m_e, and the *Common Mode Bias Sensitivity* is defined as $\left|\frac{\mathrm{d}m_e}{\mathrm{d}\sigma^2}\right|$ and tells how sensitive the bias is to common mode changes in the noise power. The *Differential Mode Bias Sensitivity* is defined similarly except the noise power changes differentially.

In Table 4-3 and Figure 4-3 nominal values for the VA and a mismatch of 10% ($\epsilon = \pm 0.1$) are used to provide actual worst-case scenario numbers for the bias sensitivities. Notice that for the common mode case both Acceleration Difference and Frequency Difference compensation methods are sensitive to the matching between channels while the Weighted Difference is not, and with a 10% mismatch, all three methods

	Acceleration Difference		Frequency Difference		Weighted Difference	
Bias Description	$\frac{1}{2\gamma}\left[\sigma_a^2 - \sigma_b^2\frac{1}{\epsilon_1}\right]$		$\frac{(\sigma_a^2+\sigma_b^2)(\omega_{0b}^3-\omega_{0a}^3\epsilon_1^2)}{\gamma(\omega_{0b}+\omega_{0a}\epsilon_1)^3}$		$\frac{(\omega_{0b}^2\sigma_a^2+\omega_{0a}^2\sigma_b^2)(\omega_{0b}^2-\omega_{0a}^2)}{\gamma(\omega_{0b}^2+\omega_{0a}^2)^2}$	
Matched Bias ($\epsilon_1 = 1$, $\sigma^2=\sigma_a^2=\sigma_b^2$)	0	0	$\frac{2\sigma^2(\omega_{0b}^3-\omega_{0a}^3)}{\gamma(\omega_{0b}+\omega_{0a})^3}$	$-2.4\times10^{-10}\sigma^2$	$\frac{\sigma^2(\omega_{0b}^2-\omega_{0a}^2)}{\gamma(\omega_{0b}^2+\omega_{0a}^2)}$	$-3.2\times10^{-10}\sigma^2$
Common Mode Bias Sensitivity	$\frac{\epsilon}{2\gamma}$	$1.25\times10^{-8}\left(\frac{g}{Hz^2}\right)$	$\frac{2(\omega_{0b}^3-\omega_{0a}^3\epsilon_1^2)}{\gamma(\omega_{0b}+\omega_{0a}\epsilon_1)^3}$	$2.0\times10^{-8}\left(\frac{g}{Hz^2}\right)$	$\frac{\omega_{0b}^2-\omega_{0a}^2}{\gamma(\omega_{0b}^2+\omega_{0a}^2)}$	$1.3\times10^{-8}\left(\frac{g}{Hz^2}\right)$
Differential Mode Bias Sensitivity	$\frac{1}{\gamma}$	$2.5\times10^{-7}\left(\frac{g}{Hz^2}\right)$	0	0	$\frac{(\omega_{0b}^2-\omega_{0a}^2)^2}{\gamma(\omega_{0b}^2+\omega_{0a}^2)^2}$	$6.5\times10^{-10}\left(\frac{g}{Hz^2}\right)$

Table 4.3: Comparison of Acceleration Estimation Bias under different compensation methods when the frequency estimates from each channel are corrupted with zero-mean noise having power σ_a^2 and σ_b^2. The nominal values of the VA and an assumed 10% mismatch ($\epsilon = \pm 0.1$) are used to provide actual numbers for worst-case bias sensitivities.

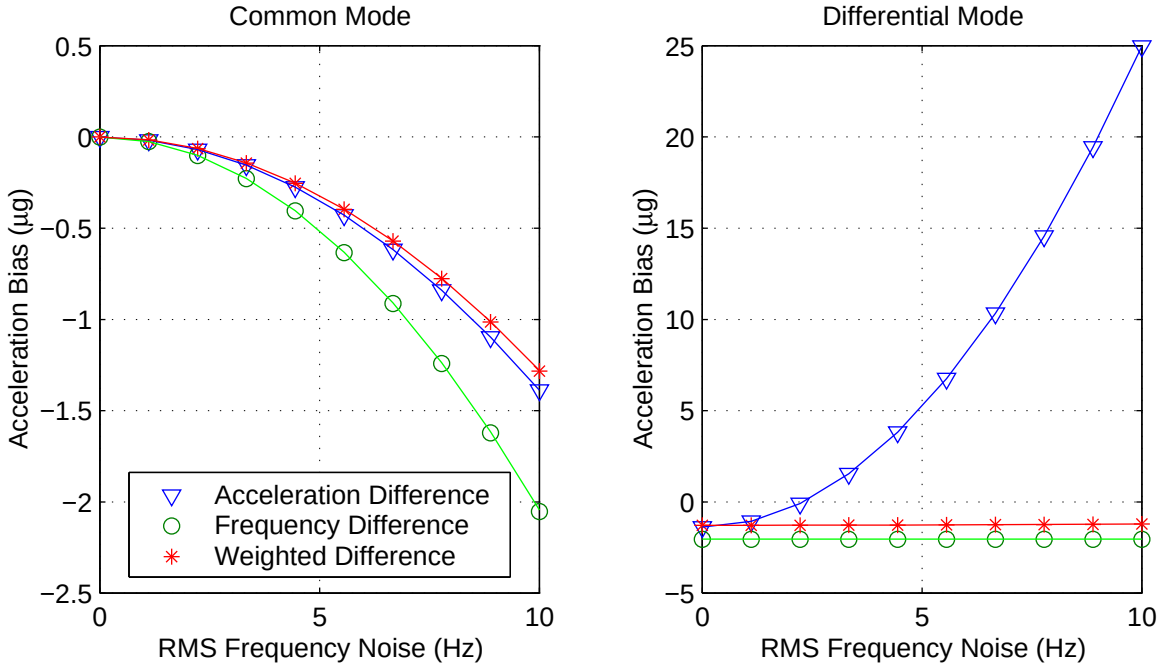


Figure 4-3: Acceleration Bias resulting from common and differential mode frequency noise power changes.

have comparable common mode bias sensitivity. Also notice that both the Frequency Difference and Weighted Difference have very similar differential mode bias sensitivity while the Acceleration Difference is orders of magnitude worse.

Random Walk

In most applications the acceleration estimate will be integrated to obtain a velocity estimate. This integration can result in random walk in the velocity estimate—meaning that the variance of the velocity estimate error grows as a function of time. If the acceleration estimate is integrated with a first order discrete integrator, the error in the velocity estimate will be

$$e_v[n] = T\sum_{i=0}^{n} e_g[i].$$

This has a variance of

$$\sigma_v^2[n] = nT^2 R_{gg}[0] \; + \; 2T^2 \sum_{i=0}^{n-1} (n-i) R_{gg}[i], \tag{4.14}$$

and if this grows as a function of time, then the velocity estimate is said to have a random walk.

For this application, the nature of the velocity estimate random walk depends on the nature of the acceleration estimation noise, and this depends on the both the method of demodulation and the method of compensation. Here we are concerned with choosing which compensator results in the smallest amount of random walk, and to do this we consider two different types of noise. One is when the noise on the frequency estimates are zero-mean, white, uncorrelated random processes, and the other is when they are differentiated zero-mean, white, uncorrelated random processes.

When $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$ are zero-mean, white, uncorrelated random processes with variances of σ_a^2 and σ_b^2, then $e_g[n]$ will also be white. If the variance of $e_g[n]$ is σ_g^2, then from Equation 4.14 the variance of the velocity estimation error is

$$\sigma_v^2[n] = nT^2\sigma_g^2.$$

The value of σ_g^2 depends on the method of compensation and can be approximated well by the linear terms of Equations 4.11, 4.12, and 4.13, and the resulting value of $\sigma_v^2[n]$ for each method is presented in Table 4.4. The random walk column of

	Velocity Variance $\sigma_v^2[n]$	**Random Walk**
Acceleration Difference	$\frac{nT^2}{\gamma^2}\left(\omega_{0a}^2\sigma_a^2 + \omega_{0b}^2\sigma_b^2\right)$	$4.9673 \times 10^{-4}\sqrt{t}$ ft/s
Frequency Difference	$\frac{nT^2}{\gamma^2}\left[\frac{2\omega_{0a}\omega_{0b}}{\omega_{0a}+\omega_{0b}}\right]^2(\sigma_a^2+\sigma_b^2)$	$4.9624 \times 10^{-4}\sqrt{t}$ ft/s
Weighted Difference	$\frac{nT^2}{\gamma^2}\left[\frac{2\omega_{0a}\omega_{0b}}{\omega_{0a}^2+\omega_{0b}^2}\right]^2(\omega_{0b}^2\sigma_a^2+\omega_{0a}^2\sigma_b^2)$	$4.9607 \times 10^{-4}\sqrt{t}$ ft/s

Table 4.4: Variance and random walk of velocity estimate due to **white frequency noise**. The values in the random walk column assume $\sigma_a^2 = \sigma_b^2 = 1$ rad^2/s^2, $T = 200$ μs, and nominal VA values.

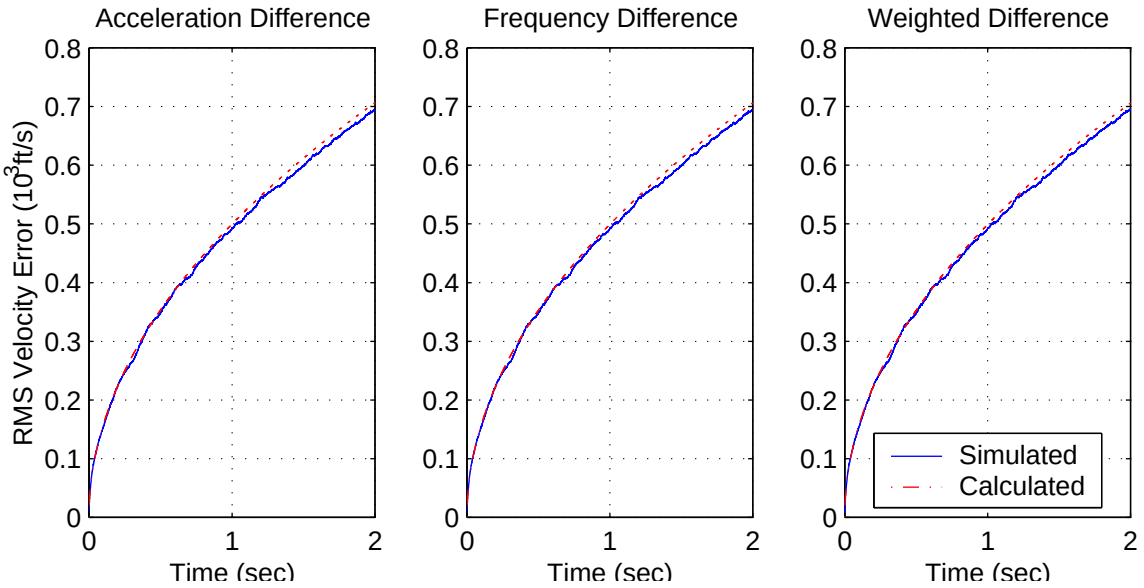


Figure 4-4: White Noise Random Walk

Table 4.4 was determined using the nominal values of the Vibratory Accelerometer. These results have been simulated and are shown in Figure 4-4. These simulated results were obtained by creating 2 seconds of white Gaussian noise of unity variance (rad^2/sec^2) to represent $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$. The noise was added to an ideal frequency

estimate and sent through each compensator. The resulting acceleration estimate was integrated. This sample path was then saved and more sample paths were generated in a similar way. Then the corresponding samples were squared, averaged, and square-rooted to produce the simulated random walk lines shown in Figure 4-4 where 1000 sample paths are averaged. The results of Table 4.4 and Figure 4-4 show that for the nominal values of the VA, the method of compensation does not significantly effect the amount of the random walk, thus any method is as good as the others.

When the noise on the frequency estimate, however, is differentiated white noise, then the results are quite different. To do this analysis we make the approximation that $e_{\omega_a}[n]$ and $e_{\omega_b}[n]$ are Gaussian. This approximation is justified by the Central Limit Theorem in that the frequency noise for the VA is the sum of many independent noise sources and thus approaches Gaussian conditions. This means we can use the relation that if $x[n]$ is a zero-mean Gaussian random process with auto-covariance $R_{xx}[m]$ and $y[n] = x^2[n]$, then

$$\begin{aligned} R_{yy}[n,m] &= \mathrm{E}(x^2[n]x^2[m]) - \mathrm{E}(x^2[n])\mathrm{E}(x^2[m]) \\ &= \sigma_x^4 + 2R_{xx}^2[n,m] - \sigma_x^4 \\ R_{yy}[m] &= 2R_{xx}^2[m] \end{aligned} \tag{4.15}$$

Furthermore, if the differentiation is a single order discrete filter, then the variance can be expressed as

$$\sigma_v^2[n] = \left(n+\frac{1}{2}\right) T^2 R_{gg}[0] \; + \; 2nT^2 R_{gg}[1].$$

Therefore, with these relations and the error sources of Equations 4.11, 4.12, and 4.13, $\sigma_v^2[n]$ can be found for each compensation method and is expressed in Table 4.5. The second order approximations of Equations 4.12 and 4.13 did not prove sufficient in simulations in capturing the random walk, and so Table 4.5 actually provides the results to a third order error expansion. Once again this result was simulated and is plotted in Figure 4-5. From this figure it is obvious that the Acceleration Difference

	Velocity Variance $\sigma_v^2[n]$	Random Walk
Acceleration Difference	$\frac{3nT^2}{4\gamma^2}\left(\sigma_a^4+\sigma_b^4\right)+\frac{T^2}{2\gamma^2}\left(\omega_{0a}^2\sigma_a^2+\omega_{0b}^2\sigma_b^2\right)$	$0.1751\sqrt{t}$ ft/s
Frequency Difference	$\frac{9nT^2}{2\gamma^2}\left[\frac{(\omega_{0a}^2+\omega_{0b}^2)^2}{(\omega_{0a}+\omega_{0b})^5}\right]^2(\sigma_a^2+\sigma_b^2)^3+$ $\frac{3nT^2}{\gamma^2}\left[\frac{\omega_{0b}^3-\omega_{0a}^3}{(\omega_{0a}+\omega_{0b})^3}\right]^2(\sigma_a^2+\sigma_b^2)^2+$ $\frac{T^2}{\gamma^2}\left[\frac{2\omega_{0a}\omega_{0b}}{\omega_{0a}+\omega_{0b}}\right]^2(\sigma_a^2+\sigma_b^2)$	$0.0115\sqrt{t}$ ft/s
Weighted Difference	$\frac{9nT^2}{2\gamma^2}\left[\frac{\omega_{0a}\omega_{0b}}{(\omega_{0a}^2+\omega_{0b}^2)^3}\right]^2(\omega_{0b}^2\sigma_a^2+\omega_{0a}^2\sigma_b^2)^3+$ $\frac{3nT^2}{\gamma^2}\left[\frac{\omega_{0b}^2-\omega_{0a}^2}{(\omega_{0b}^2+\omega_{0a}^2)^2}\right]^2(\omega_{0b}^2\sigma_a^2+\omega_{0a}^2\sigma_b^2)^2+$ $\frac{T^2}{\gamma^2}\left[\frac{2\omega_{0a}\omega_{0b}}{\omega_{0a}^2+\omega_{0b}^2}\right]^2(\omega_{0b}^2\sigma_a^2+\omega_{0a}^2\sigma_b^2)$	$0.0143\sqrt{t}$ ft/s

Table 4.5: Variance and random walk of velocity estimate due to **differentiated white frequency noise**. The values in the random walk column assume $\sigma_a^2 = \sigma_b^2 = 2/T^2$ rad^2/s^2, $T = 200$ μs, and nominal VA values.

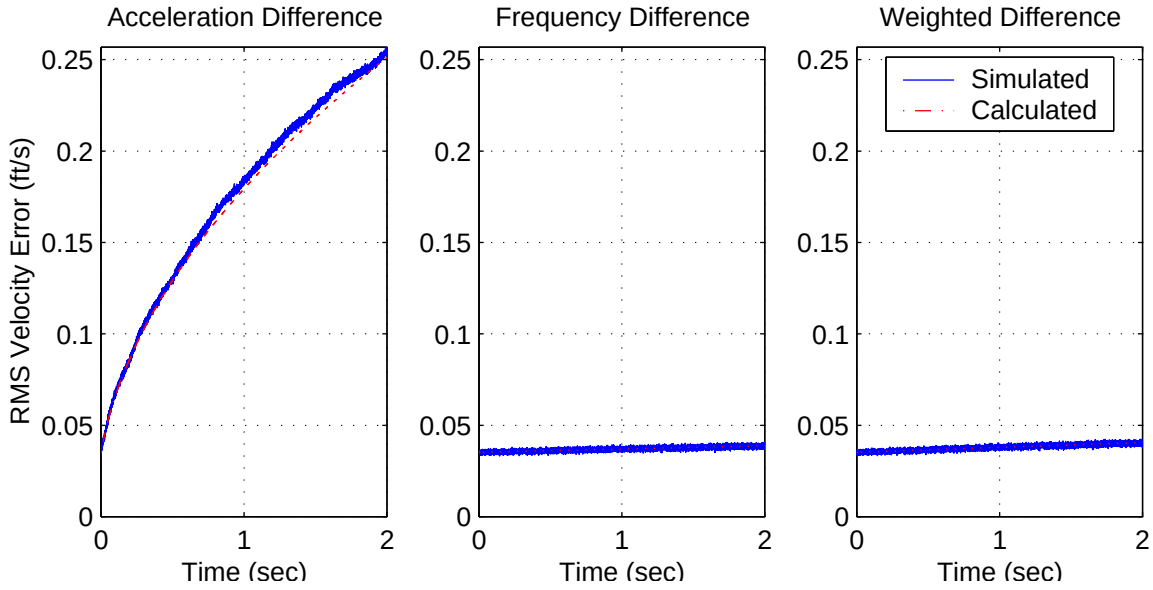


Figure 4-5: Differentiated White Noise Random Walk

has a dramatically worse random walk. The reason is that the second order term in the error in Equations 4.11, 4.12, and 4.13 is what gives rise the random walk when

the noise is differentiated, and the second order term is considerably larger in the Acceleration Difference method.

4.1.5 Compensation Method Selection

Given the above analysis, the method of compensation which is the best for the Vibratory Accelerometer is the Weighted Difference method. While the other methods have the better common mode spring stiffness variation tolerance, it is equal to or better than the others in all the other areas. It matches the others in differential spring stiffness variation, it has zero sensitivity to sampling clock drift, it is almost equal to the other methods in terms of common mode noise power sensitivity and random walk from white frequency noise, it has very small differential mode noise power sensitivity, and it has an order of magnitude less random walk from differentiated white frequency noise.

Notice that this analysis does not suggest an optimum weighted zero input frequency ω_0. Thus, it can be set at a value which allows the smallest quantization error.

4.1.6 Parameter Selection

With the frequency demodulation selected to be the Vector Readout method and the compensation method selected to be the Weighted Difference method, we are in a position to pick some of the design parameters.

A/D Converter Bit Width

The A/D converter bit width was selected to be 12 bits. This makes it several orders of magnitude lower in noise power than the white noise on the position signal due to the analog gain stage which precedes the A/D converter. A 12 bit converter is not too aggressive to build, and it does not result in registers that are too large for the amount of available gates. Therefore, it is a reasonable bit width.

Sampling Rate

Table 3.2 shows that in order to reduce the noise power on the frequency estimate we must reduce the sampling rate. On the other hand, we know from Chapter 2 that we cannot reduce the sampling rate less than 4 times the carrier frequency or else aliasing can occur. For this application the nominal carrier frequency is 20 kHz, which means that we do not want to sample below 80 kHz. We chose to sample at 200 kHz so that we did not have to implement an aggressive anti-aliasing filter. Instead we over sample a little and then use a digital filter to attenuate the harmonics and out of band noise. A digital filter is much easier to make stable over environmental conditions.

After converting the signal into phase, we downsample as was discussed in Chapter 3. This lower sampling rate can be as low as the carrier frequency since the demodulation has occurred and all the signal information must be below the carrier frequency. Once again it is advantageous to sample as slow as possible to reduce noise power and increase frequency estimate resolution. We chose to sample at 5 kHz here. This is 4 times lower than the bandwidth of our carrier, so it is possible that if the phase modulation has bandwidth above 5 kHz that it will get aliased, but for this application we are constraining the signal to have a bandwidth below 5 kHz.

Chapter 5

Amplitude Demodulation Analysis

In Chapter 1 it was established that the controller of the Vibratory Accelerometer needs an estimate of the amplitude in order to control it, and so this chapter presents an analysis of the amplitude demodulation ability of the Vector Readout method. This chapter will parallel Chapter 3 in that it will characterize the Intrinsic Noise of the method as well as characterize how Input Noise propagates through the system. This chapter also compares the Vector Readout method to other methods that could have been used to AM demodulate.

5.1 Noise Source Definitions

Given a signal that is both frequency and amplitude modulated

$$x(t) = a(t)\cos\theta(t),$$

the goal in AM demodulation is to find the instantaneous amplitude, which is $a(t)$, as shown in Figure 5-1. Just as with FM demodulators, AM demodulators produce estimates of $a(t)$ which are corrupted with the both the Intrinsic Noise and the Input

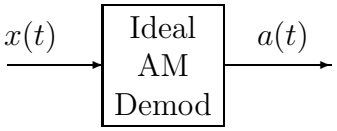


Figure 5-1: The Ideal AM Demodulator Outputs the Instantaneous Amplitude.

Noise of the demodulator (see Section 3.1 for a detailed explanation of Intrinsic and Input Noise). For this analysis we consider the same two cases of Input Noise—White Input Noise and Harmonic Input Noise. Thus, we seek to characterize the error $e(t)$ in the estimate of amplitude, which can be expressed as

$$\hat{a}(t) = a(t) + e(t).$$

The metrics used to characterize the ability of the Vector Readout method to AM demodulate will be the same as those used for FM demodulation as defined in Section 3.1, namely the *mean* error m_e and *mean square* error λ_e.

5.2 Vector Readout AM Demodulation Analysis

The Vector Readout method of amplitude demodulation is described in Chapter 2 and is shown in block diagram form in Figure 5-2. The square root function of

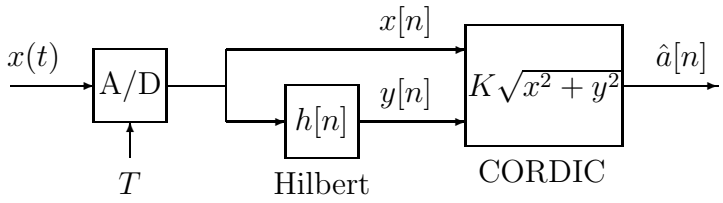


Figure 5-2: Amplitude Demodulation Using Vector Readout method

the CORDIC block is the only non-linear element of concern in the demodulation method, and because the noise sources are small compared to the signal, the square root function can be linearized around the error sources to yield good approximations. This allows us to consider each noise source individually and then add up the results to obtain the final error.

5.2.1 Intrinsic Noise

A/D, Hilbert, and CORDIC Quantization and Hilbert Gain Error

It was shown in Section 3.2.1 that the in-phase and quadrature estimates due to the A/D converter quantization are

$$\hat{x}[n] = a[n]\cos\theta[n] + e_x[n] \tag{5.1}$$

$$\hat{y}[n] = (1+\epsilon_h)\, a[n]\sin\theta[n] + e_y[n] + e_h[n]. \tag{5.2}$$

Furthermore, the CORDIC output is

$$\hat{a}[n] = \sqrt{\hat{x}^2[n] + \hat{y}^2[n]} + e_c[n] \tag{5.3}$$

where $e_c[n]$ is the quantization noise of the CORDIC. So plugging Equations 5.1 and 5.2 into Equation 5.3 gives

$$\hat{a}[n] = \sqrt{\left((1+\epsilon_h)a[n]\sin\theta[n] + e_y[n] + e_h[n]\right)^2 + \left(a[n]\cos\theta[n] + e_x[n]\right)^2} + e_c[n].$$

If we linearize this it becomes

$$\hat{a}[n] \approx a[n] + \epsilon_h a[n]\sin^2\theta[n] + (e_h[n] + e_y[n])\sin\theta[n] + e_x[n]\cos\theta[n] + e_c[n]. \tag{5.4}$$

So to first order, the mean error will be $m_e = \frac{\epsilon_h a[n]}{2}$. If we take a second order expansion, the cross terms between all the noise sources will have a mean of zero because they are uncorrelated. However, the squared terms are not zero mean, and they add to the mean as follows

$$m_e = \frac{\epsilon_h a}{2} + \frac{2\sigma_x^2 + \sigma_h^2}{4a}. \tag{5.5}$$

Thus, the gain error of the Hilbert Transform filter ϵ_h, which causes a signal at 2ω on the amplitude estimate, causes a first order bias that depends on the frequency (remember that ϵ_h is a function of the frequency), but since errors that are a func-

tion of frequency can be calibrated out (see Chapter 4), this is not a problem for our application. In addition, Equation 5.5 shows that the quantization of the A/D converter and the Hilbert Transform filter quantization cause a second order bias on the amplitude estimate. This bias is also of little concern for this application since it will be a fixed bias over all frequencies and thus will be calibrated out.

The mean square error of the amplitude estimate due to Intrinsic Noise can be found from Equation 5.4, and it is

$$\lambda_e = \frac{3}{8}\epsilon_h^2 a^2 \; + \; \sigma_x^2 \; + \; \frac{1}{2}\sigma_h^2 \tag{5.6}$$

We will see in more detail in Chapter 6 how these noise sources propagate through the closed-loop system to effect the acceleration estimate.

Environmental Stability

The only environmentally sensitive element in the Vector Readout method of amplitude demodulation is the A/D converter. The A/D converter will have an offset that is temperature dependent. For our application, however, a bandpass filter between the A/D converter and the Hilbert Transform filter will remove this offset, so it will not be of concern.

The gain of the A/D converter will also change as a function of the temperature, and this will feed right through onto the amplitude estimate. The sensitivity of the A/D converter to temperature will largely be a function of the sensitivity of the voltage reference to temperature. So if the reference is stable over temperature, then this method can also be stable over temperature.

5.2.2 Input Noise

It is of interest to know how the Vector Readout method of AM demodulation handles an input signal that is corrupted with additive noise.

White Input Noise

When the position signal is corrupted with additive white noise with power σ^2, it behaves exactly the same as the A/D converter quantization noise, so it produces a second order mean error in the amplitude estimate:

$$m_e = \frac{\sigma^2}{2a}.$$

The mean square error is unaffected by the system, so it is

$$\lambda_e = \sigma^2.$$

Harmonics Input Noise

When the input is corrupted with harmonics as given in Equation 3.2, then the in-phase and quadrature estimates will be

$$\begin{aligned}
\hat{x}[n] &= a[n]\cos\theta[n] + a_1[n]\cos(k_1\theta[n]+\phi_1) + a_2[n]\cos(k_2\theta[n]+\phi_2) + \cdots \\
\hat{y}[n] &= a[n]\sin\theta[n] + a_1[n]\sin(k_1\theta[n]+\phi_1) + a_2[n]\sin(k_2\theta[n]+\phi_2) + \cdots
\end{aligned}$$

and the amplitude estimate will be $\hat{a}[n] = \sqrt{\hat{x}^2[n] + \hat{y}^2[n]}$. These harmonics do not result in a first order mean error, however, the second order expansion shows that the mean error is

$$m_e = \frac{1}{4a}\left(a_1^2 + a_2^2 + \cdots\right).$$

A more intuitive reason why these noise sources all cause a second order mean error on the amplitude estimate can be gained from Figure 5-3.

5.2.3 Summary

Thus we see that the Vector Readout method can perform a high-precision, stable amplitude demodulation. We see we have a slight second-order bias in the presence of noise, so we will turn to comparing this method to other methods now to see if

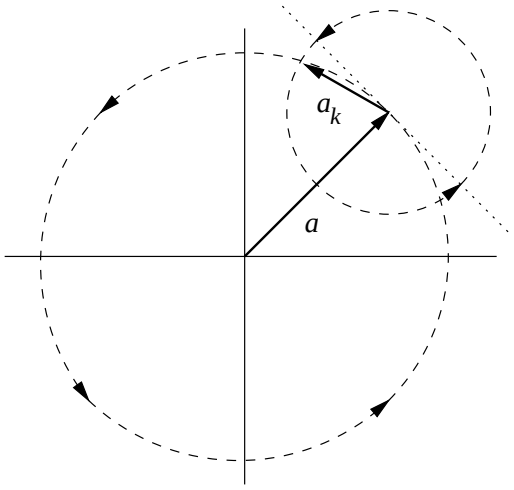


Figure 5-3: **Nature of Bias From Vector Readout Amplitude Demodulation From Harmonic Input Noise:** In this figure the vector a is the signal which is corrupted by a harmonic a_k. When a_k is in the position shown, the two vectors sum to have a magnitude that is the same as a. So if we sweep a_k a complete turn, we can see that it will result in an average magnitude slightly larger than a because the perimeter that a_k sweeps is outside the circle of a a slight bit more than inside.

other alternatives are better.

5.3 AM Demodulation Comparison

Full Wave Rectification

Full wave rectification involves taking the absolute value of the signal and then low-pass filtering it. This yields the average value of the magnitude of the signal, and this is proportional to the amplitude of the signal. While this is a simple method, it suffers in that any noise and out of band signals (such as harmonics) also get rectified and add a bias to the amplitude measurement.

Synchronous Modulation

If the frequency and phase of the carrier signal are known, then multiplying the output signal by the carrier signal and then low-pass filtering gives the amplitude of the output signal. This technique is similar to the Full Wave Rectification except that it only rectifies signals in phase with the carrier, so it does not rectify out of band noise. The problem with this method, however, is that it requires an accurate knowledge of the phase and frequency of the carrier signal, and the VA output signal may have by design a rapidly moving frequency. Thus, even though it has no mean error, it is difficult to implement for this application.

Furthermore, both the Full Wave Rectification and Synchronous Modulation have ripple that require low frequency filtering. This makes it difficult to measure fast amplitude changes.

Peak Detect With Quadrature Signal

Sampling the output signal on the rising edge of a quadrature signal results in sampling the output signal at its peak. Its limitations are that an analog comparator, whose offset is not environmentally stable, makes the decision of when to sample. Furthermore, it also has a second order bias.

5.3.1 Conclusion

Thus, compared to other methods that could be used, the bias of the Vector Readout method is small, and as we will see in the next chapter, this bias is quite inconsequential in how much it will effect the acceleration estimate. Thus, we choose to use the Vector Readout method of amplitude demodulation.

Chapter 6

Controller Design

The fact that the amplitude of the resonators within the accelerometer must be controlled was introduced in Chapter 1. The reason is that the amplitude couples into the frequency, and so any changes in the frequency due to changes in the amplitude of the resonators will be falsely interpreted as changes in acceleration.

6.1 Amplitude Disturbances

The amplitude of the resonator can be disturbed by several sources. One notable disturbance is caused by acceleration. This can be seen by modeling the resonator as a second order system with a changing spring constant:

$$\ddot{x}(t) + k(t)x(t) = 0,$$

where the spring constant $k(t)$ changes as a function of the acceleration. The solution to this differential equation is not trivial unless $k(t)$ is constant, however, consider an approximate solution for the case when $k(t)$ is a step change such that $k(t) = k_1 + \Delta k u(t)$. Suppose that for $t < 0$ the solution to the system is $x(t) = a\sin(\sqrt{k_1}t + \phi)$. Then the state of the system at time $t = 0$ can be approximated as $x(0) = a\sin\phi$ and $\dot{x}(0) = a\sqrt{k_1}\cos\phi$ (this is an approximation because $\dot{x}(0)$ will be effected by the change in $k(t)$). So if we assume that the conditions at $t = 0$ are the initial conditions

to the solution of the system with the new spring constant, then the new amplitude will be

$$\alpha' = \alpha\sqrt{\frac{k_1}{k_1 + \Delta k}\cos^2\phi + \sin^2\phi}.$$

This shows that the new amplitude depends on the phase of the position signal when the step occurs and that the amount of change in the amplitude is maximized when $\phi = 0$, so the maximal change in amplitude is

$$\alpha' = \alpha\sqrt{\frac{k_1}{k_1 + \Delta k}}. \tag{6.1}$$

Furthermore it is minimized when $\phi = \frac{\pi}{2}$ which results in $\alpha' = \alpha$.

This approximation for the change in amplitude agrees well with simulation, and simulation also shows that if you distribute the change in $k(t)$ over many cycles of the position signal, then the amount of change in the amplitude does not depend on the phase of the signal and that it is less than the amount of change that would result from a step change of the same amount. Notice that the amplitude disturbance is not symmetric, which means that a step of -100g will not produce the opposite amplitude effect when compared to a step of 100g even if they occur at the same phase. This lack of symmetry in the amplitude disturbance is also noticeable in simulation when the spring constant $k(t)$ is slowly changed. This means that the common mode rejection of the system to amplitude disturbances caused by acceleration will not be ideal.

Thus, for the purposes of this analysis, the worst case scenario is assumed such that no common mode rejection occurs and that the error in the acceleration estimate due to a step change of 100g can account for 10% of the allowed error. The allowed error is 1 μg. Furthermore the amplitude disturbance due to changes in acceleration is also assumed to be the major cause of amplitude disturbance so that other effects neglected.

6.2 Sensitivity of Acceleration Measurement To Amplitude

Before determining how to control the oscillator, we need to establish how changes in amplitude affect the frequency. The relation between the amplitude and frequency can be found when one considers a more accurate model of the oscillator. This is:

$$m\ddot{x} + b\dot{x} + k_1 x + k_2 x^2 + k_3 x^3 = f \tag{6.2}$$

The addition of the square and cubic term in this differential equation changes the natural frequency. Reference [6] uses perturbation methods to approximate the natural frequency of the system described by Equation 6.2, but the same result can also be found using describing functions as follows. Consider the case when

$$f(t) = \beta \cos \omega_n t.$$

Since the plant causes a -90° phase shift at its natural frequency, we assume that the output will be

$$x(t) = \alpha \sin \omega_n t.$$

This means that $\dot{x}(t)$, $\ddot{x}(t)$, $x^2(t)$, and $x^3(t)$ will be

$$\begin{aligned}
\dot{x}(t) &= \alpha\omega_n \cos \omega_n t \\
\ddot{x}(t) &= -\alpha\omega_n^2 \sin \omega_n t \\
x^2(t) &= \frac{\alpha^2}{2}(1 - \cos 2\omega_n t) \\
x^3(t) &= \frac{\alpha^3}{4}(3 \sin \omega_n t - \sin 3\omega_n t)
\end{aligned}$$

Plugging these all back into Equation 6.2 and equating the $\sin \omega_n t$ terms gives

$$m\omega_n^2 = k_1 + \frac{3k_3}{4}\alpha^2. \tag{6.3}$$

In this approximation we use the describing function technique of neglecting any of the frequency terms in the non-linearity that are not at the fundamental frequency. This shows that the natural frequency is a function of the amplitude α. This approximation has been verified in simulation as shown in Figure 6-1 where both the natural frequency relation of Equation 6.3 and the simulated frequency of the resonator are plotted as a function of the amplitude. Solving Equation 6.3 for ω_n and taking the

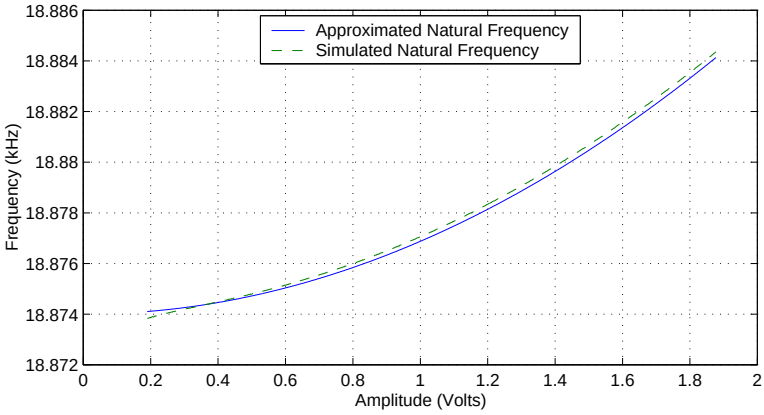


Figure 6-1: Comparison of Approximated (Equation 6.3) and Simulated Natural Frequency.

partial derivative with respect to α yields

$$\frac{\partial \omega_n}{\partial \alpha} = \frac{3k_3\alpha}{4m\omega_n}.$$

This is the scale factor that relates changes in amplitude to changes in frequency. If we say that $\frac{\partial \omega_n}{\partial g} = \gamma$, where $\frac{\partial \omega_n}{\partial g}$ is often called the scale factor of the accelerometer, the sensitivity becomes $\frac{\partial g}{\partial \alpha} = \frac{3k_3\alpha}{4m\omega_n\gamma}$. Finally, knowing that there is a gain K from the position $\alpha(t)$ (in meters) to the position signal $a(t)$ (in volts), the sensitivity of the acceleration measurement to changes in measured amplitude, which we call ν, is

$$\nu = \frac{\partial g}{\partial a} = \frac{3k_3 a}{4m\omega_n K^2\gamma}. \tag{6.4}$$

The nominal values of each of the variables in the above equation are provided in Table 6.1. If we assume that $r(t)$ is the reference or desired amplitude, then the total

m	1.325×10^{-9} kg	The effective mass.
k_3	$61.38 \times 10^9 \frac{\text{N}}{\text{m}^3}$	The effective cubic spring constant.
ω_n	$2\pi \cdot 20000 \frac{\text{rad}}{\text{s}}$	The nominal natural frequency.
a	1.0 V	The nominal measured amplitude of the resonator.
K	$4 \times 10^6 \frac{\text{V}}{\text{m}}$	The gain of the electronics which convert the position of resonator in meters to a voltage.
γ	$2\pi \cdot 100 \frac{\text{rad}}{\text{g}}$	The scale factor of the accelerometer.
ν	$0.0275 \frac{\text{g}}{\text{V}}$	The sensitivity of the acceleration measurement to changes in amplitude.

Table 6.1: Nominal Values For Sensitivity of Accelerometer to Changes in Amplitude.

integrated error due to amplitude deviation is

$$e_v(t) = \nu \int_0^t a(t) - r(t)\ \mathrm{d}t. \tag{6.5}$$

This is the velocity error, and after T_o seconds, it must be less than the error budget of the controller. This error budget has been defined to be 10% of the velocity error which would result from a 1 μg bias.

6.3 Controller Overview

With the acceptable error defined, a controller can be designed to meet these specifications. The biggest difficulty in designing the controller, however, is that the force needs to be applied at the resonant frequency of the oscillator, and that frequency is moving as a function of the acceleration. Thus, the controller needs to determine both the frequency and amplitude of the force to be applied to the resonator. To do this we use an Automatic Gain Controller (AGC) as shown in Figure 6-2. Notice that there are really two loops in the system. The inner loop is known as the Phase Regeneration Loop, and the outer is the Automatic Gain Control Loop. In Figure 6-2, $P(s)$ is the accelerometer, and the input into the accelerometer is the force $f(t)$. Notice that this force is the product of the AGC signal $c(t)$ and the carrier signal $v(t)$. The goal of the Phase Regeneration Loop is to set the frequency and phase of the carrier $v(t)$. The goal of the amplitude controller $G(s)$ is to control the amplitude

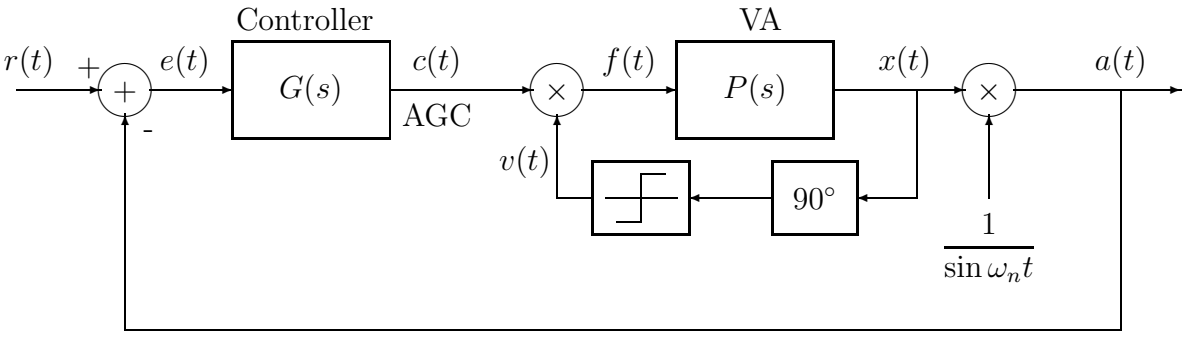


Figure 6-2: AGC Controller Block Diagram

of the oscillation.

Figure 6-2 shows that this system is really in a positive feedback configuration. The reason is that the outer loop can be considered a variable gain block, so the system can be collapsed to that shown in Figure 6-3 where the outer loop is expressed as a variable gain AGC. The first block in the feedback path of the inner loop causes a

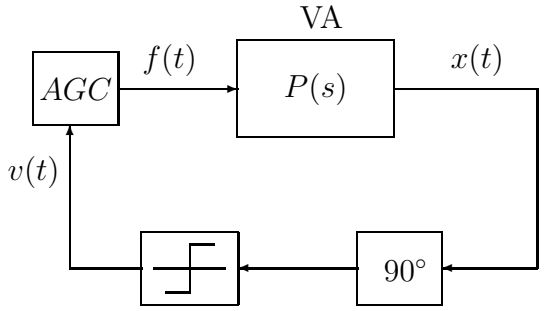


Figure 6-3: Outer Loop Collapsed To A Variable Gain Block Labeled AGC.

90° phase shift. This has the effect of "undoing" the phase introduced by the plant. The plant has a transfer function (see Equation 1.1) of

$$P(s) = \frac{1}{ms^2 + bs + k},$$

so the phase of the plant at ω_n is $-90°$. Therefore, the 90° phase shifter ensures that the phase around the loop at ω_n is 0°.

Following the phase shifter is the limiter block. This limiter performs several useful functions. One is that it normalizes the amplitude so that the gain of the 90° phase shifter does not have to be unity. Therefore, a simple inverting integrator or

differentiator can be used to perform the phase shift.

The other useful function of the limiter is that it adjusts the gain of the system to ensure the open loop gain at ω_n is one. A positive feedback system will oscillate at a frequency where the open loop transfer function equals one [2]. Therefore, the limiter could perhaps more appropriately be called the automatic gain control block. To see this consider the case when the system is oscillating at ω_n and AGC is fixed. The gain around the loop is exactly one since the system is oscillating. Now suppose that AGC doubles. This means that output of the plant also doubles since it is linear. However, the amplitude of the output of the limiter stays the same, so its gain halves. Therefore, the AGC block caused the amplitude of the oscillators to double, and the limiter automatically adjusted for this to keep the open loop gain exactly one to ensure that oscillation continues.

One potential problem with the limiter is that it turns the 90° phase-shifted sine wave into a square wave, so it also introduces harmonics into the loop. However, for this application, the plant has a Q on the order of 100,000, so these harmonics are severely attenuated by the plant. Therefore, they will be neglected in this analysis.

6.4 Amplitude Control

6.4.1 The Amplitude Transfer Function of the Plant

The outer or AGC loop of Figure 6-2 shows that the controller $G(s)$ is what adjusts the amplitude of the resonators. To design this controller, it is necessary to know how changes in the amplitude of the drive signal effect the amplitude of the resonator. In other words, as Figure 6-4 shows, we must collapse the amplitude modulation, the plant, the amplitude demodulation, and the phase regeneration blocks into a single block and find the transfer function through it. Knowing this transfer function, which is called $H(s)$ in Figure 6-4, enables us to design the controller $G(s)$ to hold the amplitude $a(t)$ constant.

The first point to make in finding $H(s)$ is that since modulation and demodulation

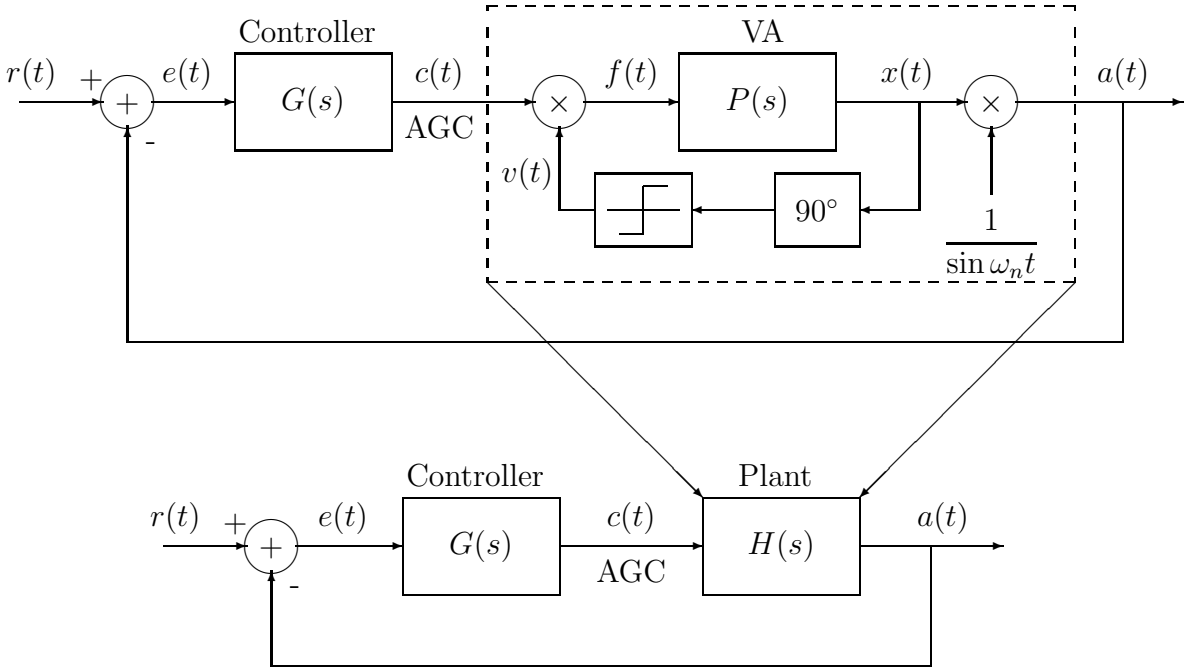


Figure 6-4: Amplitude Controller Block Diagram Reduction

are non-linear, $H(s)$ in general does not exist. However, when the accelerometer is operating at its natural frequency, it is possible to make an approximation that characterizes the plant well and enables us to define an $H(s)$.

To find $H(s)$ we assume the resonator is oscillating at its natural frequency so that the position signal is $x(t) = a(t)\sin\omega_n t$. This means $v(t) = \cos\omega_n t$ (here we neglect the harmonics introduced by the limiter). If we send in a complex exponential at frequency Ω as the AGC signal $c(t)$ and follow it through each of the blocks to see how it effects the amplitude $a(t)$, then we can find $H(\mathrm{j}\omega)$. Thus, we define $c(t)$ as

$$c(t) = \mathrm{e}^{\mathrm{j}\Omega t},$$

and after it is modulated with the phase regeneration signal, it becomes the force signal $f(t)$ that drives the resonator:

$$\begin{aligned} f(t) &= c(t)v(t) = \mathrm{e}^{\mathrm{j}\Omega t}\cos(\omega_n t) \\ &= \frac{1}{2}\left(\mathrm{e}^{\mathrm{j}(\Omega+\omega_n)t} + \mathrm{e}^{\mathrm{j}(\Omega-\omega_n)t}\right). \end{aligned}$$

Thus, $f(t)$ is the sum of two complex exponentials, and the resonator is a linear, time invariant system of the form

$$P(s) = \frac{P_0}{s^2 + \frac{\omega_n}{Q}s + \omega_n^2}.$$

Since the complex exponentials of $f(t)$ are eigenfunctions of a linear, time-invariant system, the output of the oscillators due to the input $f(t)$ is

$$x(t) = \frac{1}{2}\left(|P(\Omega+\omega_n)|\mathrm{e}^{\mathrm{j}(\Omega+\omega_n)t+\angle P(\Omega+\omega_n)} + |P(\Omega-\omega_n)|\mathrm{e}^{\mathrm{j}(\Omega-\omega_n)t+\angle P(\Omega-\omega_n)}\right). \quad (6.6)$$

The equation for $x(t)$ can be simplified considerably when we make the approximation that Ω is much smaller than ω_n. Then the magnitude terms of Equation 6.6 become

$$\begin{aligned} |P(\Omega+\omega_n)| &= \frac{P_0}{\sqrt{(\omega_n^2-(\Omega+\omega_n)^2)^2 + \left(\frac{\omega_n(\Omega+\omega_n)}{Q}\right)^2}} \\ &= \frac{QP_0}{\sqrt{Q^2(\Omega^2+2\omega_n\Omega)^2 + \omega_n^2(\Omega+\omega_n)^2}} \\ &\approx \frac{QP_0}{\omega_n}\frac{1}{\sqrt{(2Q\Omega)^2+\omega_n^2}} \end{aligned} \qquad \begin{aligned} |P(\Omega-\omega_n)| &= \frac{P_0}{\sqrt{(\omega_n^2-(\Omega-\omega_n)^2)^2 + \left(\frac{\omega_n(\Omega-\omega_n)}{Q}\right)^2}} \\ &= \frac{QP_0}{\sqrt{Q^2(\Omega^2-2\omega_n\Omega)^2 + \omega_n^2(\Omega-\omega_n)^2}} \\ &\approx \frac{QP_0}{\omega_n}\frac{1}{\sqrt{(2Q\Omega)^2+\omega_n^2}}. \end{aligned}$$

Since these are equal, we define

$$P_1 = |P(\Omega+\omega_n)| = |P(\Omega-\omega_n)|.$$

Now we turn to evaluating the phase terms of Equation 6.6 using the same approximation that $\Omega \ll \omega_n$:

$$\begin{aligned} \angle P(\Omega+\omega_n) &= -\arctan\left(\frac{\omega_n(\Omega+\omega_n)}{Q(\omega_n^2-(\Omega+\omega_n)^2)}\right) \\ &= -\arctan\left(\frac{\omega_n(\Omega+\omega_n)}{-Q\Omega(\Omega+2\omega_n)}\right) \\ &\approx -\arctan\frac{\omega_n}{-2Q\Omega} \\ &= -\frac{\pi}{2} - \arctan\frac{2Q\Omega}{\omega_n} \end{aligned} \qquad \begin{aligned} \angle P(\Omega-\omega_n) &= -\arctan\left(\frac{\omega_n(\Omega-\omega_n)}{Q(\omega_n^2-(\Omega-\omega_n)^2)}\right) \\ &= -\arctan\left(\frac{\omega_n(\Omega-\omega_n)}{-Q\Omega(\Omega-2\omega_n)}\right) \\ &\approx -\arctan\frac{-\omega_n}{2Q\Omega} \\ &= \frac{\pi}{2} - \arctan\frac{2Q\Omega}{\omega_n}. \end{aligned}$$

This means that if we define

$$\psi = \arctan\frac{2Q\Omega}{\omega_n},$$

then $x(t)$ of Equation 6.6 simplifies considerably to:

$$\begin{aligned} x(t) &= \frac{1}{2}\left(P_1 e^{j(\Omega+\omega_n)t-\frac{\pi}{2}-\psi} + P_1 e^{j(\Omega-\omega_n)t+\frac{\pi}{2}-\psi}\right) \\ &= P_1 e^{j\Omega t-\psi}\cos\left(\omega_n t - \frac{\pi}{2}\right) \\ &= P_1 e^{j\Omega t-\psi}\sin\omega_n t. \end{aligned}$$

The output of the resonator $x(t)$ is then demodulated with $\sin\omega_n t$, so the amplitude $a(t)$ is

$$a(t) = P_1 e^{j\Omega t-\psi}.$$

So we see that if we send an AGC signal $c(t) = e^{j\Omega t}$ into the system, then the resulting amplitude is $a(t) = P_1 e^{j\Omega t-\psi}$.

$$c(t) = e^{j\Omega t} \longrightarrow \boxed{\text{H(s)}} \longrightarrow a(t) = P_1 e^{j\Omega t-\psi}$$

Thus, the systems looks linear and time invariant under the approximation that $\Omega \ll \omega_n$. So the system $H(j\Omega)$ has a magnitude of P_1 and a phase of ψ, and so $H(s)$ can easily be found to be a single pole system:

$$\left.\begin{aligned} |H(j\Omega)| &= \tfrac{QP_0}{\omega_n}\tfrac{1}{\sqrt{(2Q\Omega)^2+\omega_n^2}} \\ \angle H(j\Omega) &= -\arctan\tfrac{2Q\Omega}{\omega_n} \end{aligned}\right\} \quad H(s) = \frac{QP_0}{\omega_n^2}\left[\frac{1}{\frac{2Q}{\omega_n}s+1}\right]. \tag{6.7}$$

With the amplitude dynamics of the plant determined and expressed as $H(s)$ in Equation 6.7, we can begin to design a controller which generates the AGC signal $c(t)$. The nominal values of the parameters in $H(s)$ that will be used in designing the controller are provided in Table 6.2.*

*The parameters such as Q, P_0, and ω_n within $H(s)$ are assumed fixed, but in reality they may change over time or from sensor to sensor. These effects will be considered later in the chapter.

6.4.2 The Transfer Function of the Demodulator

In determining the amplitude transfer function of the plant, $H(s)$ in Equation 6.7, we assumed that we are able to perform a perfect amplitude demodulation. As Chapter 5 shows, this is not really the case, so in order to make the model of the system as shown in Figure 6-4 more accurate, we need to account for the dynamics of the demodulator. These dynamics are introduced in the feedback path as block $D(s)$ in Figure 6-5.

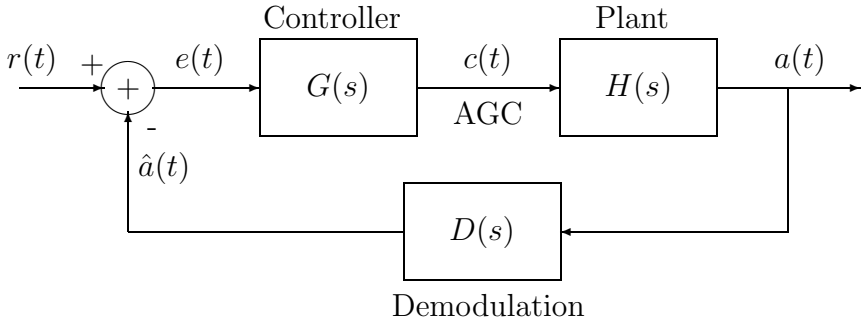


Figure 6-5: Controller With Amplitude Demodulation Dynamics Included

The nature of $D(s)$ depends on the method of amplitude demodulation, but using the Vector Readout approach, $D(s)$ is composed of a gain K_d and a delay T_d.[†]

$$D(s) = K_d \mathrm{e}^{-sT_d} \tag{6.8}$$

The gain of the demodulation K_d is fixed by the demodulation algorithm and is provided in Table 6.2. The value of the delay T_d is also set by the demodulation algorithm, but it is adjustable and can be made as short as needed at the expense of adding more noise to both the amplitude and frequency demodulation.

The delay T_d is the result of processing done by the digital filters, and it introduces a very significant constraint in designing the controller. In the absence of this delay, the system is a simple first order system as defined by $H(s)$. With the delay, however, the open loop transfer function has a linear phase term which will pose a practical limit on how fast the controller can be and still remain stable.

[†]The effects of noise from the amplitude demodulation are important and will be considered later.

6.4.3 Defining Nominal System Parameters

The nominal values of $H(s)$ and $D(s)$ are provided in Table 6.2.

P_0	6.0×10^7	The unit-less gain relating the AGC signal $c(t)$ to the position signal $x(t)$. This assumes that the gain of the D/A conversion is 1.
Q	$70,000$	The Quality Factor of the resonator. It can vary significantly from 10,000 to 100,000, and so the controller should be designed to be insensitive to the exact value of Q.
ω_n	$2\pi \cdot 20,000 \frac{\text{rad}}{\text{s}}$	The zero input natural frequency of the resonator. Nominally it will be 20 kHz but can be as low as 15 kHz.
T_d	$267.5\ \mu$s	The time delay through the signal processing blocks of the demodulator.
K_d	$0.3294\ \frac{1}{\text{V}}$	The gain of the demodulation. The CORDIC has a gain of $\frac{1.647}{2}$ and the A/D conversion has a gain of $\frac{1}{2.5}$.

Table 6.2: Nominal Accelerometer System Values

The product of $H(s)$ and $D(s)$ is plotted in Figure 6-6. Notice the effects of the

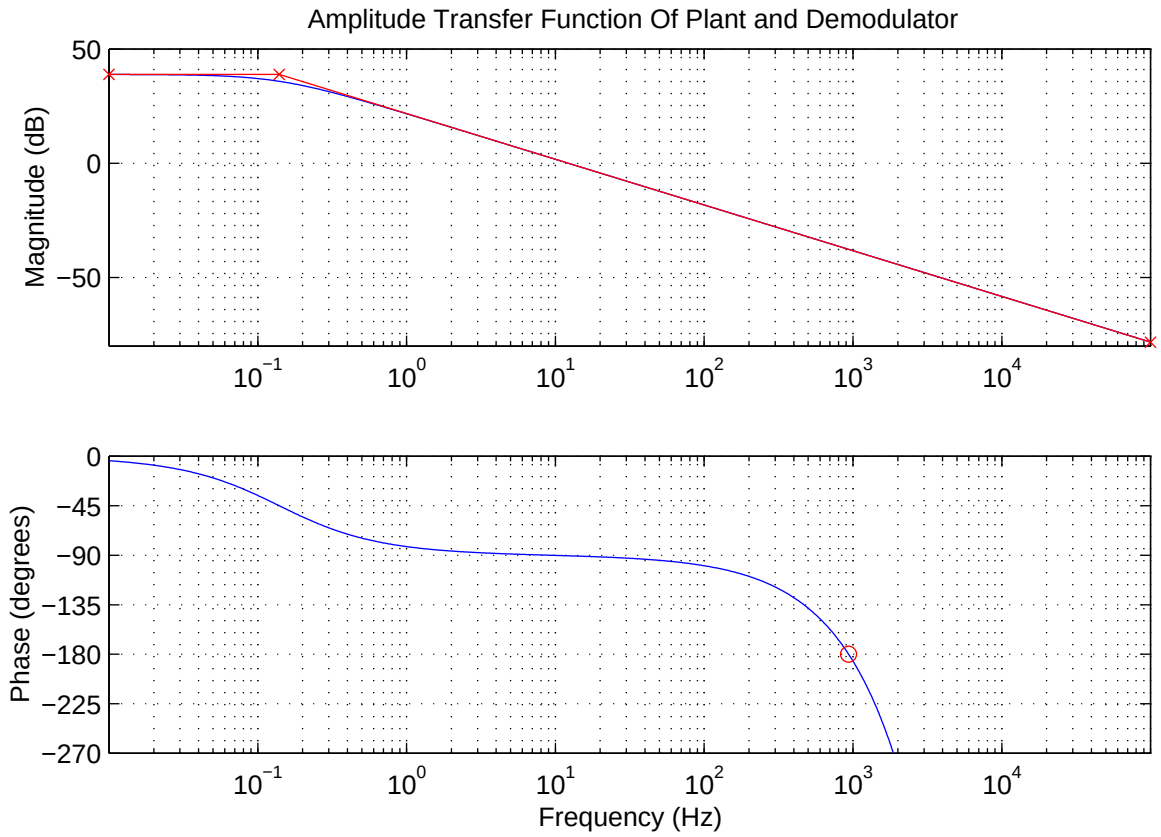


Figure 6-6: Plant and AM Demodulation Transfer Function.

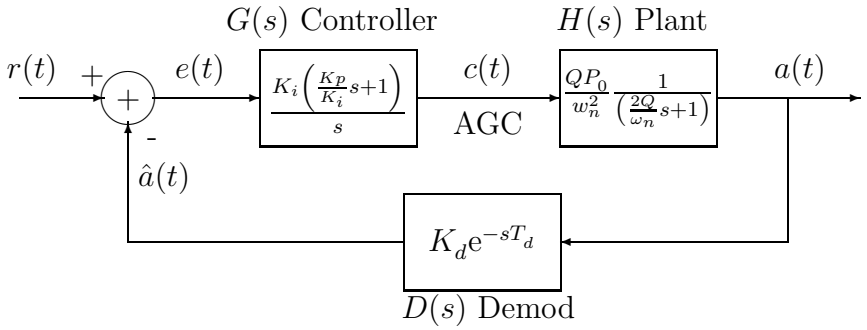


Figure 6-7: Amplitude Controller Block Diagram

delay on the phase. The single pole of the plant causes $-90°$ of phase by 10 Hz, and then the delay begins to get significant and introduces an additional $-90°$ of phase by 1000 Hz. When the phase caused by the delay is plotted on a log scale, it rolls off sharply above 100 Hz. Because the phase rolls off sharply while the gain roll off remains fixed at -10 dB per decade, it makes pushing the unity crossover frequency above 1000 Hz difficult and presents a limitation on the bandwidth of the controller. The easiest way to combat this limitation is decrease the amount of the delay.

The sharp roll off of phase due to the delay makes the Gain Margin an important measure of stability if unity crossover occurs in the region where the phase is steep. The reason is that slight variations in the open loop gain can cause significant changes in the unity crossover phase, and these changes will greatly effect the stability of the system.

6.4.4 Designing The Controller

Using $H(s)$ and $D(s)$ defined in Equations 6.7 and 6.8, we now turn to designing the controller $G(s)$ which will hold the amplitude $a(t)$ constant. The control loop is shown again in Figure 6-7 showing $G(s)$ as a PI (Proportional plus Integral) controller. This controller has several advantages for this system. One is that it has zero steady state error, which is necessary in keeping the acceleration error bias to a minimum. The zero steady state error results because the PI controller has a pole at zero. The second advantage of the PI controller is that it compensates for the addition of a pole into the system by putting a zero at a high frequency to allow one to increase the bandwidth

of the system. Thus we can define $G(s)$ to have the form

$$G(s) = \frac{K_i}{s} + K_p, \tag{6.9}$$

and it remains to find controller coefficients which can meet the design specification.

With $H(s)$, $D(s)$, and $G(s)$ specified, we can write the open loop gain of the system as

$$L(s) = \frac{QP_0K_iK_d}{\omega_n^2} \frac{\left(\frac{K_p}{K_i}s + 1\right)}{s\left(\frac{2Q}{\omega_n}s + 1\right)} \mathrm{e}^{-sT_d} \tag{6.10}$$

We can see from this loop equation that we have control over both the gain of the system and the location of the zero.

Disturbance Rejection

The most important task of the controller is to reject amplitude disturbances to an acceptable level. An amplitude disturbance $b(t)$ due to acceleration changes can be modeled as an additional input signal in the system block diagram as shown in Figure 6-8. The transfer function from the disturbance to the amplitude is

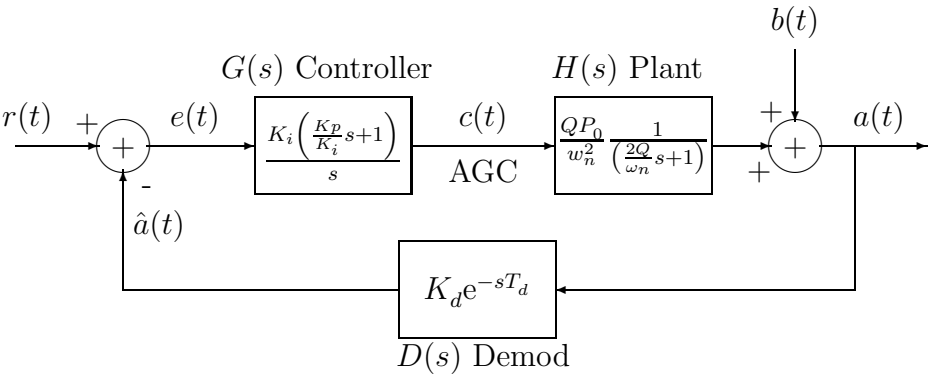


Figure 6-8: Control Loop With Additive Disturbance

$$\begin{aligned} \frac{A(s)}{B(s)} &= \frac{1}{1 + H(s)G(s)D(s)} \\ &= \frac{\omega_n^2 s\left(\frac{2Q}{\omega_n}s + 1\right)}{\omega_n^2 s\left(\frac{2Q}{\omega_n}s + 1\right) + K_iQP_0K_d\left(\frac{K_p}{K_i}s + 1\right)\mathrm{e}^{-sT_d}}. \end{aligned}$$

We are interested in minimizing the velocity error $e_v(t)$ as defined in Equation 6.5, which is

$$e_v(t) = \nu \int_0^t a(t) - r(t)\ \mathrm{d}t. \tag{6.11}$$

Using linearity, we can find how much each input individually contributes and add them to find the total error. Therefore, if we assume that the set point $r(t)$ is held constant for all time and if $b(t)$ is shut off, then the output $a(t)$ will also be constant for all time and $r(t)$ will contribute zero error to $e_v(t)$. Then if we shut $r(t)$ off and turn $b(t)$ on, we can find how much error the disturbance will cause. The integral of Equation 6.11 is not easily evaluated, but we can find the total velocity by letting $t \to \infty$. This is just the total area under $a(t)$, and if we assume that $b(t) = 0$ for $t < 0$, then this can easily be found by using the transform property that the area under the time domain signal is the same as the Laplace Transform of the signal evaluated at zero. Thus, if the input disturbance is a step of amplitude β, then $B(s) = \frac{\beta}{s}$, so the total velocity error is

$$\lim_{t\to\infty} e_v(t) = \nu A(0) = \frac{\nu\omega_n^2\beta}{K_i Q P_0 K_d} \tag{6.12}$$

Notice that the velocity error does not depend on the proportional coefficient, but only on the integral coefficient. It also shows that the error is minimized as K_i is maximized. If the location of the zero is kept fixed so that the ratio of K_p to K_i is fixed, then increasing K_i increases the open loop gain and pushes unity crossover out. This increase in bandwidth makes the controller faster at responding to disturbances, but it also makes it more responsive to noise. Thus, before we design this controller to maximize K_i, we need to look at the noise/bandwidth trade off to see if there are any issues with having a fast controller.

If we say that the total velocity error of Equation 6.12 must be less than the error budget, then we can find a bound on the integral coefficient. This is

$$K_i > \frac{\nu\omega_n^2\beta}{EQP_0K_d},$$

where E is the error budget and is assumed to be 2.45×10^{-4} $\frac{\text{m}}{\text{s}}$. If we assume a worst case step input of a step of 100g, then by using the nominal values of Table 6.1 and 6.2 and the relation of Equation 6.1, then $\beta = 0.3$, and the bound on K_i is

$$K_i > 0.384. \tag{6.13}$$

Noise Rejection

There are three sources of noise that can enter into the loop. The first is noise picked up in the analog gain circuitry. The second is the noise introduced by the amplitude demodulation. Both of these noise sources where characterized in Chapter 5, and the third noise source is the noise of the D/A converter. The controller is really a discrete-time controller, and its output is converted to an analog voltage, and this D/A conversion will have noise on it. All of these noise sources can be characterized as additive noise sources which inject noise at various spots in the system. We will first focus on the effect of the first two noise sources which can be combined into a single noise source that corrupts our estimate of the amplitude. In Figure 6-9, this noise is shown as the signal $e_a(t)$, where $a(t)$ and $e_a(t)$ are summed together to produce the signal $\hat{a}(t)$, which is the amplitude measurement. The transfer function

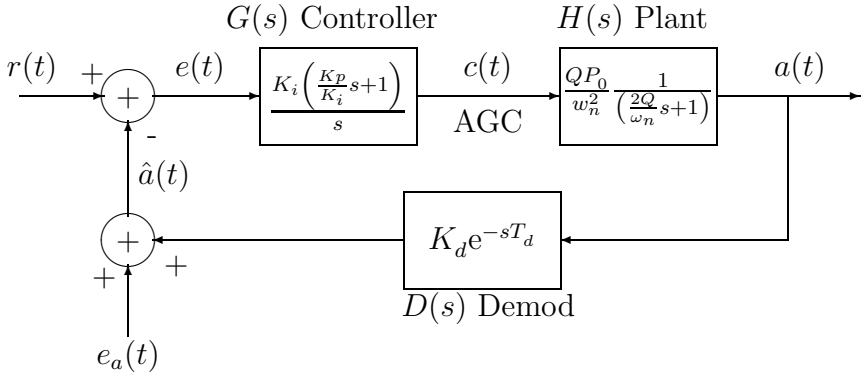


Figure 6-9: Amplitude Control Loop With Additive Noise

from the noise to the amplitude is

$$\frac{A(s)}{E_a(s)} = \frac{-H(s)G(s)}{1 + H(s)G(s)D(s)}$$

$$= \frac{-K_i Q P_0 \left(\frac{K_p}{K_i}s + 1\right)}{\omega_n^2 s \left(\frac{2Q}{\omega_n}s + 1\right) + K_i Q P_0 K_d \left(\frac{K_p}{K_i}s + 1\right) \mathrm{e}^{-sT_d}}.$$

This is a cumbersome response, but as Figure 6-10 shows, this frequency response can be approximated well with a single pole of the form

$$\frac{A(s)}{E_a(s)} = \frac{1/K_d}{\frac{2\omega_n}{P_0 K_d K_p - 2\omega_n K_i/K_p}s + 1}.$$

This approximation was obtained by dropping the delay term, dividing by the zero to find the approximate single pole location, and then creating the single pole system with the correct DC gain.

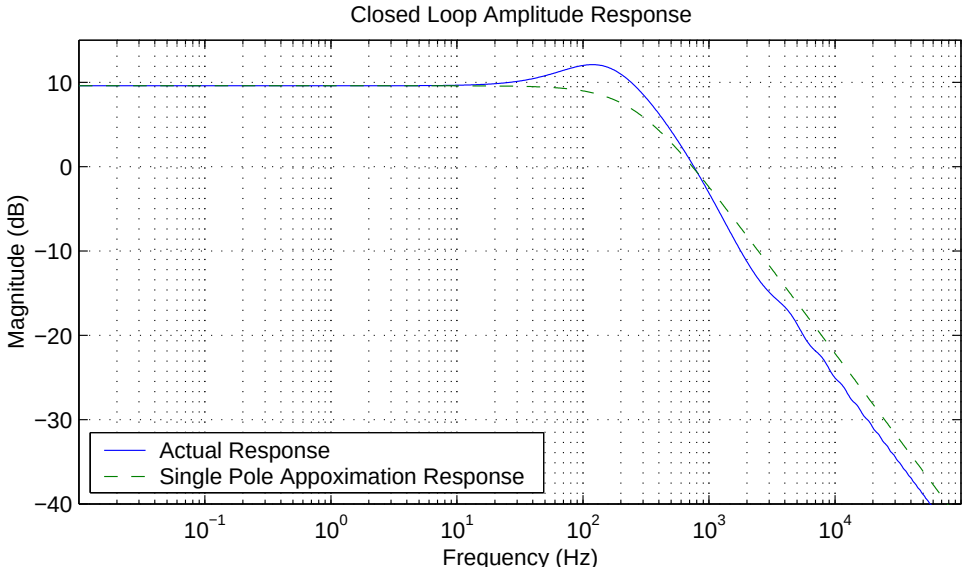


Figure 6-10: Closed Loop Amplitude Demodulation Noise Response.

In Chapter 5 we characterized $e_a(t)$ as a white noise random process. We will assume it has a noise power of σ^2. Since the noise gets filtered by a single pole, any noise that has power above the break point will be attenuated by the system. Bringing this break point down would result in more noise filtering, but the cost would be a reduction in bandwidth of the controller, which results in less disturbance rejection and more velocity error.

It turns out for this system that the amount of noise attenuation is of negligible

consequence because if the noise is zero mean, it will not cause an acceleration estimation error bias. It will, however, cause random walk on the velocity estimate because amplitude movement feeds through with a scale factor of ν onto the frequency, and we saw in Chapter 4 that error on the frequency estimate causes a random walk on the velocity estimate. The amount of random walk, however, is effected little by the location of the break point of the filter because the DC component of the noise is what an integrator gains the most.

To see that it is of negligible advantage to reduce the breakpoint of the filter, consider the following derivation. If the injected noise is white and filtered by a first order system with a pole at τ, then the autocorrelation of the noise after the filtering is

$$K_{aa}(t) = \frac{1}{2\tau} \mathrm{e}^{\frac{-|t|}{\tau}}.$$

The signal is then sampled at rate T, so the autocorrelation becomes

$$K_{aa}[n] = \frac{1}{2\tau} \mathrm{e}^{\frac{-|n|T}{\tau}}.$$

This signal then feeds through onto the frequency with scale factor ν and is then integrated with a first order accumulator of the form

$$\hat{v}[n] = \hat{v}[n-1] + T\hat{g}[n],$$

so the variance of the velocity estimate will be

$$\sigma_v^2[n] = \nu^2\sigma^2T^2 \left[n \sum_{i=-n+1}^{n-1} K_{aa}[i] - 2\sum_{i=0}^{n-1} iK_{aa}[i] \right].$$

When nT gets large compared to τ, this approximates to

$$\begin{aligned} \sigma_v^2[n] &\approx \nu^2\sigma^2nT^2 - \nu^2\sigma^2T^2 \frac{\mathrm{e}^{\frac{-T}{\tau}}}{\tau(-1+\mathrm{e}^{\frac{-T}{\tau}})^2} \\ &\approx \nu^2\sigma^2nT^2. \end{aligned}$$

In the absence of filtering, the variance is exactly equal to $\nu^2\sigma^2 nT^2$, so it means that the low pass filter has little effect in reducing the random walk when the time of the measurement is much larger than the time constant of the filter. This shows there is no real advantage to making the controller slower to increase the amount of noise attenuation, and since there is an advantage to making it faster, the controller will be optimized for speed.

Σ-Δ D/A Conversion

The third noise source injects its noise on the AGC signal $c(t)$, and this noise is the result of the D/A conversion. This conversion is being performed using a first order Σ-Δ modulator. For this application a Σ-Δ modulator has a unique advantage over traditional converters.

The input into the Σ-Δ modulator is a digital word which represents the AGC value, and it is running at 200 kHz. The output is a 1 bit sequence $\hat{c}[n]$ as shown in Figure 6-11. This sequence is converted from a digital bit into a digital logic level of

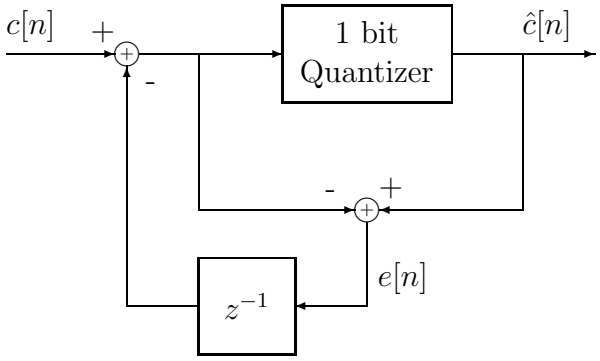


Figure 6-11: First Order Σ-Δ Modulator.

either 0 V or 5 V. It is then filtered by a double pole low pass analog filter and level shifted to be centered between -2.5 V and 2.5 V.

The whole D/A process is shown in Figure 6-12. The quantizer within the Σ-Δ modulator is represented as an additive noise source which is zero-mean, white, and uniformly distributed with variance $\sigma_q^2 = \frac{1}{3}$ (assuming that $c[n]$ represents a fractional number between -1 and 1). The process of converting the digital sequence into an

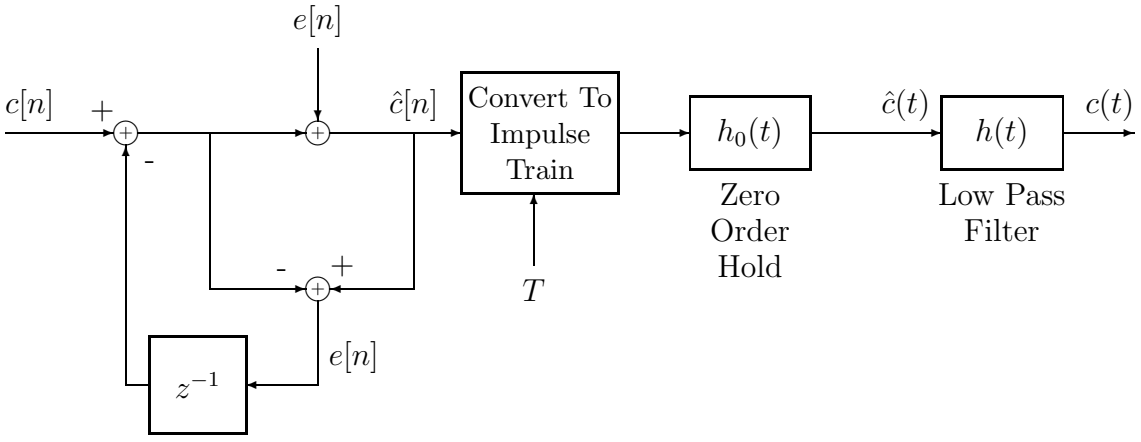
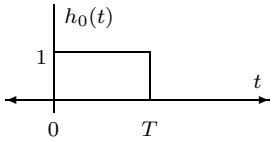


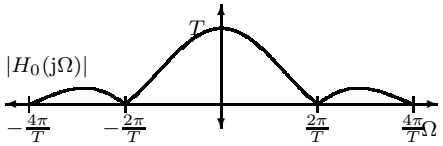
Figure 6-12: D/A Conversion With Σ-Δ Modulator.

analog voltage can be viewed as converting the sequence to an impulse train and then filtering it with a rectangular box filter $h_0(t)$ where

$$h_0(t) = \begin{cases} 1, & 0 \le t \le T \\ 0, & \text{otherwise.} \end{cases}$$



$$H_0(\mathrm{j}\Omega) = \frac{2\sin \Omega T/2}{\Omega} \mathrm{e}^{-\mathrm{j}\Omega T/2}$$



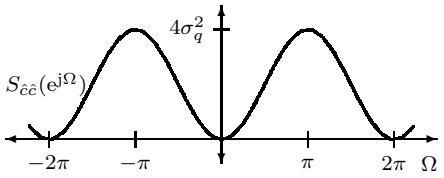
This is a zero-order hold D/A conversion. At this point the signal is an analog signal which is the composite of the true AGC signal and the quantization noise, so it is low-pass filtered by $h(t)$ to cut the quantization noise. The transfer function of the filter is

$$H(s) = \frac{1}{(\tau s + 1)^2}$$

where $\tau = \frac{1}{2\pi \cdot 30\mathrm{e}3}$.

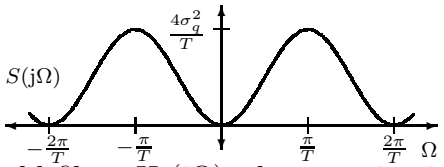
We are concerned with how the quantization noise effects the acceleration measurement, and since this system is linear, we can analyze how the error source effects it independently of the true AGC signal. The error source $e[n]$ (shown in Figure 6-12) is shaped by the Σ-Δ modulator to push the noise into high frequency bands. The noise on the output of the Σ-Δ modulator has a power spectral density of [7]

$$S_{\hat{c}\hat{c}}(\mathrm{e}^{\mathrm{j}\Omega}) = 4\sigma_q^2 \sin^2 \frac{\Omega}{2}.$$



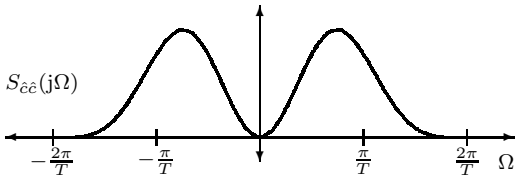
It is then converted to a continuous time impulse train, where it becomes

$$S(\mathrm{j}\Omega) = \frac{4\sigma_q^2}{T} \sin^2 \frac{\Omega T}{2}.$$



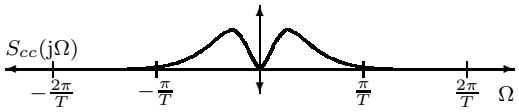
Then after being filtered by the zero-order hold filter $H_0(\mathrm{j}\Omega)$, the quantization noise becomes

$$S_{\hat{c}\hat{c}}(\mathrm{j}\Omega) = \frac{16\sigma_q^2 \sin^4 \Omega T/2}{T\Omega^2}.$$



This shows the noise is shaped as desired in that the noise power is minimized at low bandwidths, which is where the AGC signal $c(t)$ is. The low-pass filter $H(\mathrm{j}\Omega)$ serves to attenuate the noise and pass the AGC signal. After the low pass filter the noise becomes

$$S_{cc}(\mathrm{j}\Omega) = \frac{16\sigma_q^2 \sin^4 \Omega T/2}{T\Omega^2(\tau^2\Omega^2 + 1)^2}.$$



So you can see that as you decrease T (increase the sampling rate) it pushes the quantization noise power higher and higher. Since the AGC signal and the poles of the low-pass filter are independent of the sampling rate, the effect is that more of the quantization noise is attenuated. This means that the Σ-Δ modulator should run as fast as the hardware will allow. The only problem with this is that the input $c[n]$ must be upsampled to match the sampling rate of the Σ-Δ modulator. If this upsampling is a simple zero-order hold, however, the signal portion of the output $c(t)$ will be exactly the same as it would have been otherwise since the actual D/A conversion is also a zero-order hold converter. More complicated methods (such as first-order hold) could be employed if $c[n]$ is of sufficient bandwidth where the droop of $H_0(\mathrm{j}\Omega)$ is of concern, but for our application it is fine.

Thus, for this application we upsample $c[n]$ with a simple zero-order hold to 10 MHz and run the Σ-Δ modulator at this rate. The resulting power spectrum is shown in Figure 6-13. If this noise spectrum is integrated and the resulting variance

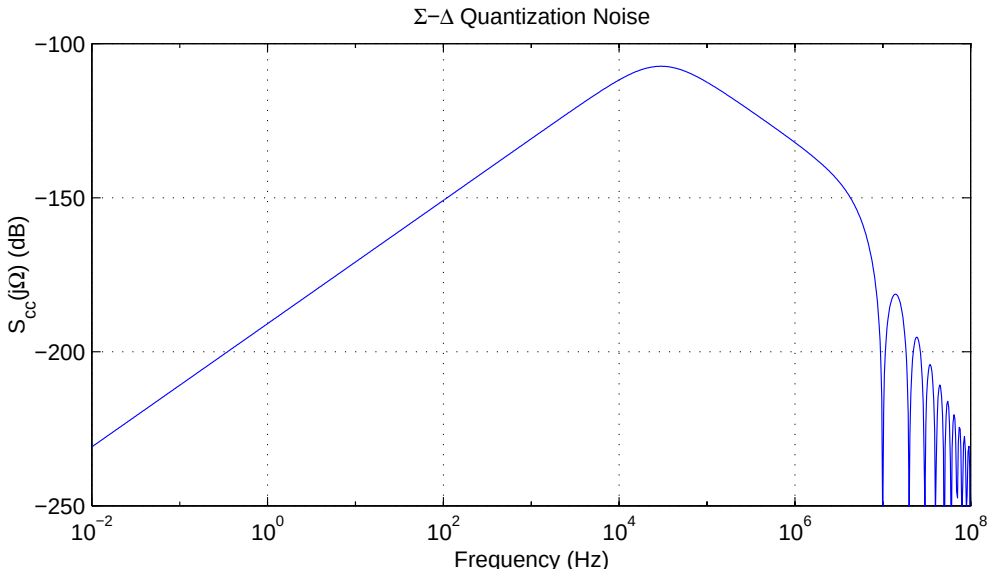


Figure 6-13: Quantization Noise From Σ-Δ D/A AGC Conversion.

is converted into an equivalent number of bits, the result is about 7 bits. This is not a very good converter, and it could easily be made better by decreasing the bandwidth of the analog lowpass filter. The problem with doing that, however, is that these poles start adding phase to the open loop system and reduce the bandwidth of the controller. It turns out, however, that this converter does not necessarily need to be any better than it is because the noise gets filtered further by the plant and the closed loop system. To see this, we first model the noise source as shown in Figure 6-14 where the Σ-Δ modulator noise is called $e_c(t)$. The transfer function from this noise source to the amplitude is

$$\frac{A(s)}{E_c(s)} = \frac{H(s)}{1 + G(s)H(s)D(s)},$$

which is plotted in Figure 6-15. This is a bandpass filter which has a zero at zero and further attenuates the higher order noise. Thus, the quantization noise of the Σ-Δ modulator that appears on the amplitude of the resonator is shaped such that

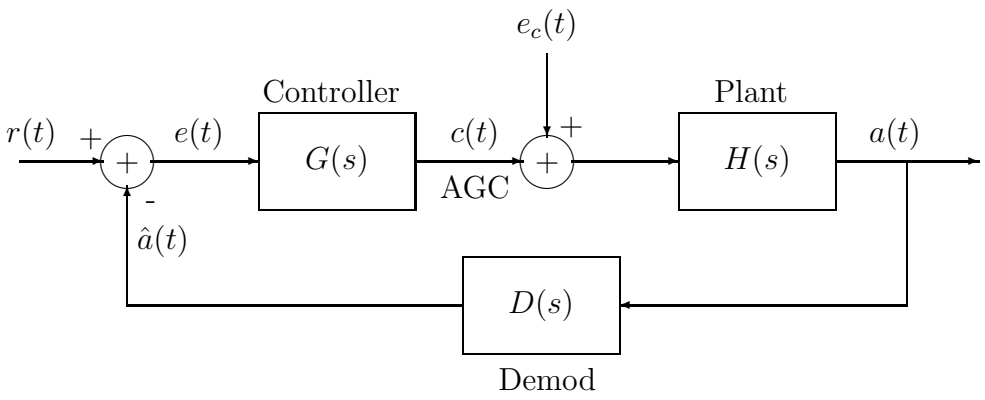


Figure 6-14: Block Diagram Depicting Additive Σ-Δ Quantization Noise

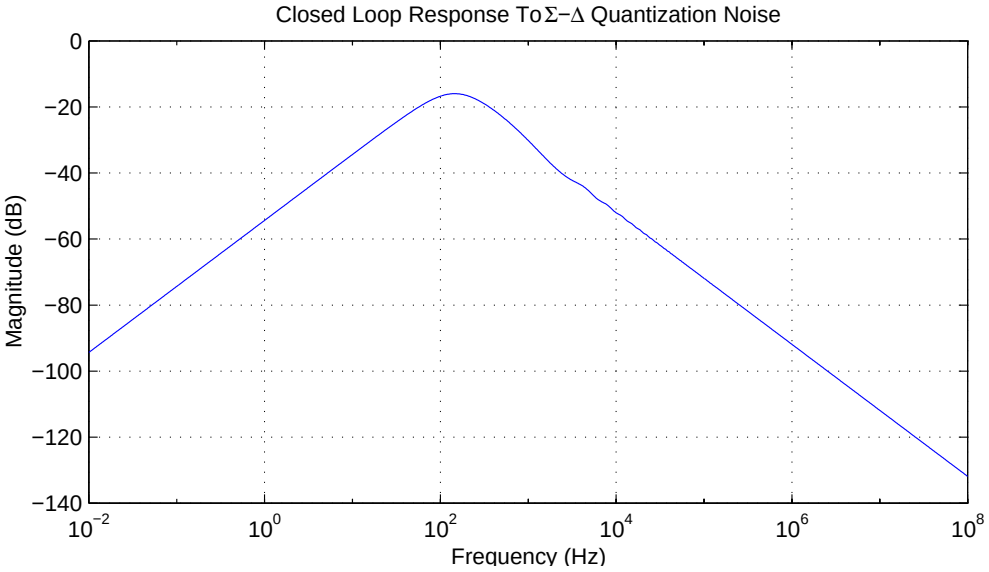


Figure 6-15: Σ-Δ Quantization Noise To Amplitude Transfer Function.

it has two zeros at zero and four poles at higher order frequencies. The amplitude noise feeds through to become frequency noise with a scale factor of ν, and since its spectrum has a zero at zero frequency, it does not result in a substantial random walk after it is converted to an acceleration estimate and then integrated to become a velocity estimate (see Chapter 4). Thus, the noise due to the D/A conversion is actually of little consequence compared to the noise that gets injected prior to and during AM demodulation.

6.4.5 Designing Around the Open Loop Transfer Function

We have established that the controller needs to be as fast as possible for two reasons. One is that it increases the DC gain of the closed loop system and the other is that the controller is faster at compensating for a wider range of amplitude disturbances. The open loop system is plotted in Figure 6-16. This shows that the delay introduced by the digital filters is what constrains the bandwidth of the controller because pushing the bandwidth out too far will cause the system to be unstable.

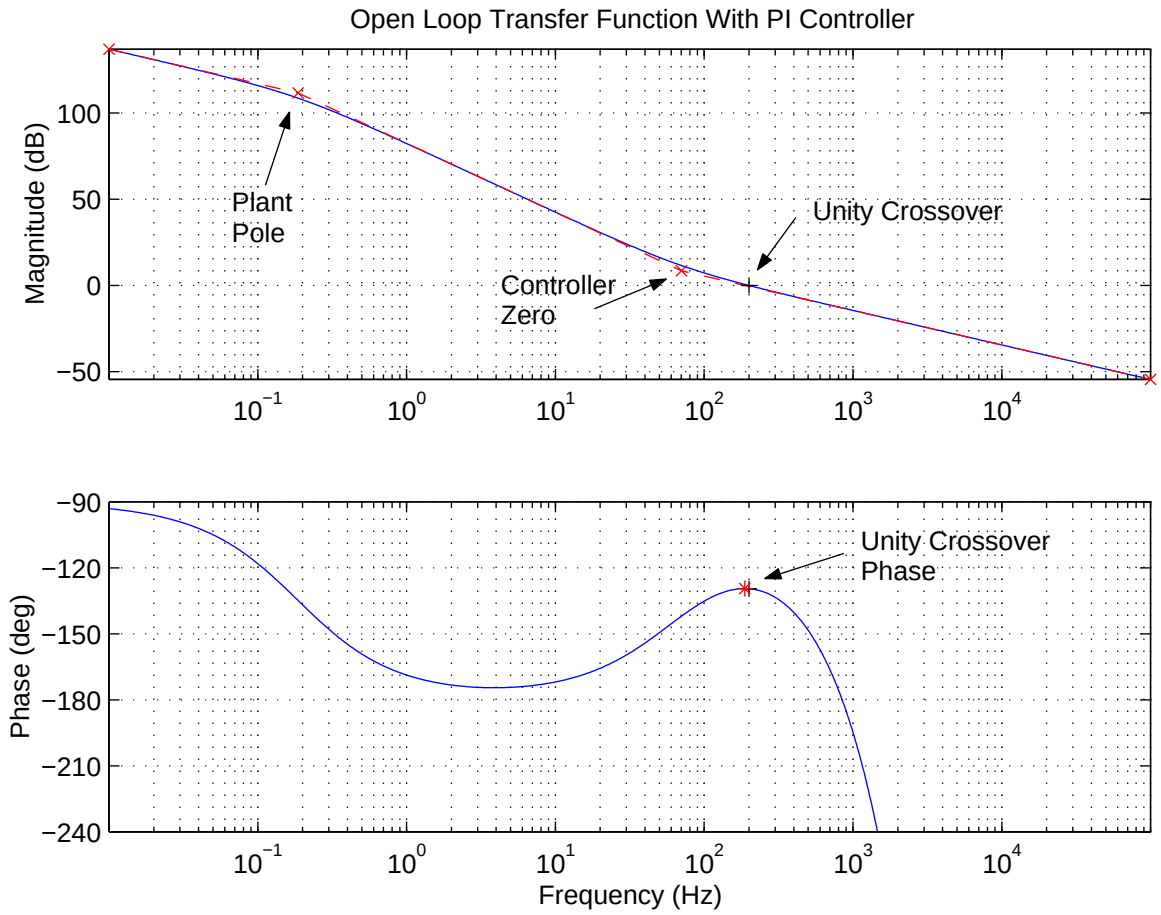


Figure 6-16: Open Loop Transfer Function of System.

Thus, in order to design this controller to maximize the DC gain and bandwidth under the constraints of stability, we use the following approach. As shown in Figure 6-16, the phase of the open loop system has a local maximum which results from the addition of the positive slope of the controller zero and the negative slope of the

delay. This phase maximum makes a great position to crossover because the system will be optimally stable at that point. The frequency of this local maximum is

$$\omega_c \approx \sqrt{\frac{K_i}{T_d K_p}}.$$

With this set as the crossover frequency, we have constrained the system enough to uniquely determine K_i and K_p, which are

$$\begin{aligned} K_p &= \frac{\omega_c \tau_p \sqrt{T_d^2 \omega_c^2 + 1}}{K_0} \\ K_i &= \frac{-T_d K_p^3 K_0^2}{T_d^2 K_p^2 K_0^2 - \tau_p^2}. \end{aligned}$$

Using the nominal values of the accelerometer as provided in Table 6.2 and forcing unity crossover at 200 Hz gives $K_p = 22.36$ and $K_i = 988.8$. The open loop transfer function for the system with these controller values is plotted in Figure 6-16. You can see that with these values, K_i is about three orders of magnitude larger than it must be in order to ensure that a 100g disturbance does not cause more than 10% of the error budget in the mean acceleration estimation error as stated in Equation 6.13.

One may notice and argue then that this controller could stand to be a lot slower and still meet the specification, and while this is true, we established that there is no real harm in adding more noise to the system by speeding it up since the random walk is dominated by the low bandwidth noise. Furthermore, in practice we found that a crossover of around 200 Hz was necessary for the servo to adequately null the amplitude error due to ambient room temperature variation.

6.4.6 Stability Margins

Since not all sensors will have exactly the same parameters and since these parameters may change over time for a given sensor, it is important to characterize the stability margins of the controller and consider worst case parameter variation to ensure that

the system will be stable. The open loop transfer function found in Equation 6.10 is

$$L(s) \;=\; \frac{QP_0K_iK_d}{\omega_n^2}\,\frac{\left(\frac{K_p}{K_i}s+1\right)}{s\left(\frac{2Q}{\omega_n}s+1\right)}\,\mathrm{e}^{-sT_d}.$$

The only parameters that may drift over time or be different from sensor to sensor are Q, ω_n, and P_0. The other parameters, which are K_i, K_p, and T_d can be changed, but once they are set by the designer they are fixed and will not drift.

The phase margin for the values of K_p and K_i found in the previous section is 50.6°, and the gain margin is 4.40. These margins offer more than adequate stability as can be seen from the following cases of parameter variation:

Q: The variation in the quality factor Q of the resonator is from 10,000 to 100,000. The stability of the system, however, is not significantly affected by the value of Q. Changes in the value of Q will change the location of the plant pole and scale the DC gain, but the unity crossover frequency and phase are not affected. This is shown in the plot in Figure 6-17.

ω_n: Variations in the natural frequency can range from 10 kHz to 30 kHz, and these changes actually change the gain of the system as can be seen in Figure 6-18. The worst case scenario is when the natural frequency is at 10 kHz, and Figure 6-18 shows that the system is still stable and has around 40° of phase stability.

P_0: The variation in the DC gain of the plant can be caused by differences in mass or capacitance from sensor to sensor. Under these conditions, the gain margin is 4.40, which is more than enough for such parameter variation. One area that may be of concern, however, is if the mechanical designers change these parameters in future revisions of the mechanical structure. If this is ever the case the controller coefficients can be scaled accordingly and everything will still work the same.

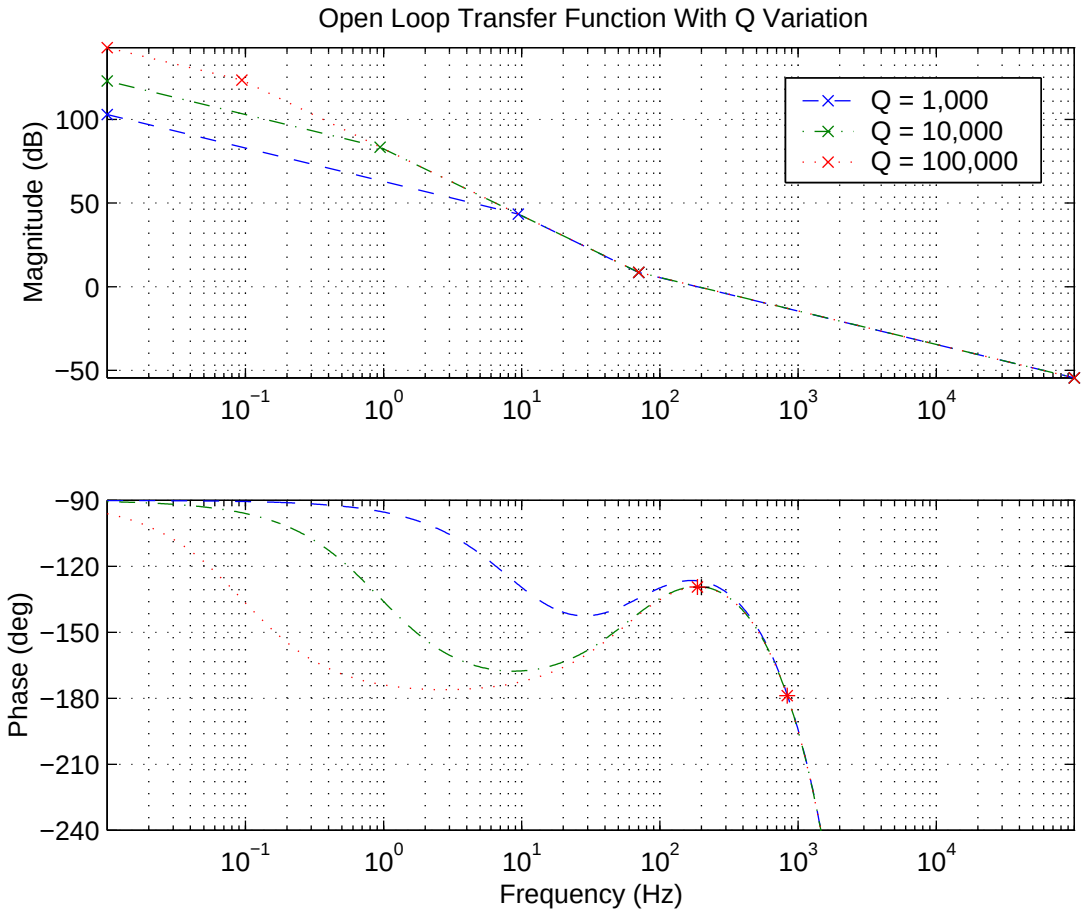


Figure 6-17: Open Loop Transfer Function With Q Variation

6.5 Conclusion

Therefore, we have found a controller that is more than adequate at meeting the design constraints. Furthermore we have shown it has sensible stability margins and that it behaves well under disturbances and noise.

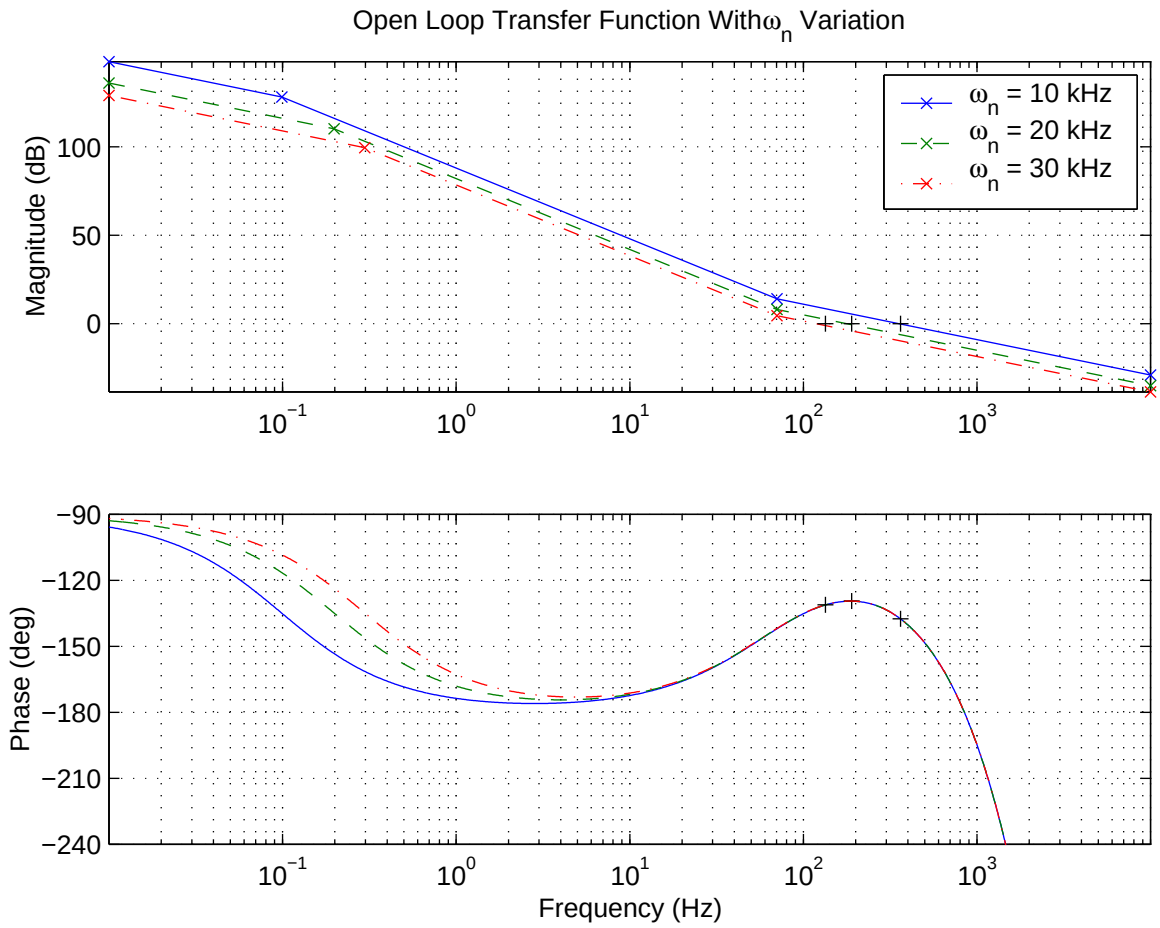


Figure 6-18: Open Loop Transfer Function With ω_n Variation

Chapter 7

Conclusion

7.1 Design Specification Check

In the preceding chapters each component of the system used to control the Vibratory Accelerometer was analyzed and characterized in great detail. Putting all of this together and plugging in the nominal values of the accelerometer enables us to see how well the system meets the design specifications stated in Chapter 1.

7.1.1 Dynamic Range (1μg to 100g)

The upper limit of the dynamic range is interpreted for this application to be the point of hard saturation. Under extreme conditions, the accelerometer can experience $\pm$100g, which corresponds to a frequency swing between 10 kHz and 30 kHz. This swing in frequency does not pose saturation issues to the A/D converter, the Hilbert Transform filter, or the CORDIC. The saturation issue occurs when adjacent angle samples coming out of the CORDIC are subtracted. Thus, the register that performs the subtraction must be large enough to handle a frequency of 30 kHz. This has been ensured.

The lower limit of the dynamic range is interpreted to be the RMS value of the noise on the acceleration estimate over a 1 Hz bandwidth. All the noise sources on the acceleration estimate as characterized in previous chapters fit into three categories. The first is the stochastic noise sources in Table 3.2. These noise sources, however,

are differentiated white noise, so as one reduces the bandwidth from 5 kHz to 1 Hz, these noises virtually disappear and become negligible. The other type of noise in Table 3.2 is that due to harmonic input, and the 1 Hz filter will also remove these. The last noise source is that which feeds through the amplitude loop onto the frequency. Figure 6-10 shows the spectrum of this noise. If we temporarily assume that this noise is actually white over the full 100 kHz bandwidth and that it has power σ^2, then the same white noise will appear on the frequency estimate with power

$$\sigma_{\hat{g}}^2 = 2\nu^2\sigma^2$$

where ν is the scale factor from amplitude to acceleration as found in Equation 6.4 and the factor of 2 results from the fact that there are two channels with independent noise sources of the same power. If this noise is filtered by a 1 Hz filter, then it will reduce the noise by a factor of 10^{-5}, and so the RMS value becomes

$$\sigma_{\hat{g}} = 0.307\ \mu\text{g}.$$

This is 3 times better than the specification of 1μg. In fact, we can obtain the 1μg specification if the acceleration signal is filtered to any frequency below 10 Hz.

Acceleration Estimation Bias ($< 1\ \mu$g)

In Equation 4.13 we found the acceleration estimation error for the Weighted Difference method when the frequency estimates of channels A and B are corrupted with noise. As mentioned, this noise produces a fixed bias that can be calibrated out, but here we assume a worst case scenario that it is not calibrated out. From Equation 4.13 we can see that if channel A has noise power of σ_a^2 and channel B has noise power of σ_b^2, then the mean error in the acceleration estimate will be

$$m_e = \frac{\omega_{0b}^2 - \omega_{0a}^2}{\gamma(\omega_{0b}^2 + \omega_{0a}^2)^2}\left(\omega_{0b}^2\sigma_a^2 + \omega_{0a}^2\sigma_{0b}^2\right)$$

Using Table 3.2 and the nominal values of the accelerometer, namely, $\omega_{0a} = 2\pi \cdot 20,000\ \frac{\text{rad}}{\text{s}}$, $\omega_{0b} = 2\pi \cdot 19,000\ \frac{\text{rad}}{\text{s}}$, and $\gamma = 157.9 \times 10^6\ \frac{\text{rad}^2}{\text{s}^2\text{g}}$, we can find the total bias error that will result in the acceleration error. The results for each noise source are included in Table 7.1.

Velocity Estimate Random Walk ($< 0.014\ \frac{\text{ft}}{\text{s}\sqrt{\text{hr}}}$)

The velocity estimate random walk for the Weighted Difference method of compensation was characterized in Tables 4.4 and 4.5. Once again we consider the three types of noise that corrupt the acceleration estimate. Table 3.2 provides the description of the differentiated white noise and deterministic noise sources, and Section 6.4.4 describes the velocity estimation random walk which comes from noise in the amplitude loop. The amount of velocity estimate random walk that results from each of these noise sources has been calculated and provided in Table 7.1. Table 3.2 shows that when the noise sent into the compensator is differentiated white noise or a deterministic harmonic, it is of negligible consequence to the random walk. The reason that the differentiated white noise case is so negligible is because we chose to use the Weighted Difference method of compensation (see Chapter 4). The reason that the amplitude loop noise causes the majority of the random walk can be seen by comparing Tables 4.4 and 4.5. If the position signal has a noise dominated by the white noise of the first analog gain stage, which is σ^2, then it feeds through onto the actual amplitude with unity gain, meaning the actual amplitude of the resonator has noise power σ^2. The amplitude affects the velocity noise power as $\sigma_v^2[n] = 2\nu^2\sigma^2 nT^2$, where ν is a scale factor determined by Equation 6.4. Therefore, the velocity random walk is $\nu\sigma\sqrt{2T}$. Using nominal values gives the results in Table 7.1. This shows the random walk is an order of magnitude better than the design specification.

7.2 Implementation Details

This project was a large success in terms of finding a method to both amplitude and frequency demodulate the position signal of the Vibratory Accelerometer and in

Noise Source	**Conditions**	**Acceleration Estimate Mean Error** (μg)	**Velocity Estimate Random Walk** $\left(\frac{\text{ft}}{\text{s}\sqrt{\text{hr}}}\right)$
A/D Converter Quantization	Input has amplitude $a = 1$V and gets quantized to $B_x = 11$ fractional bits.	3.22×10^{-4}	1.52×10^{-8}
Hilbert Transform Quantization	Output register of the Hilbert filter is quantized to $B_h = 15$ fractional bits.	6.30×10^{-7}	2.98×10^{-11}
CORDIC Quantization	The CORDIC algorithm is iterated $N = 15$ times.	1.26×10^{-6}	5.95×10^{-11}
Hilbert Gain Error	Gain Response has worse case gain ripple of $\epsilon_h = 1.8 \times 10^{-4}$.	1.21×10^{-4}	0
A/D Bias	A/D offset removed by filtering, so $a_0 = 0$.	0	0
White Input Noise	Input has amplitude $a = 1$V and is corrupted with zero-mean white noise with power $\sigma^2 = 6.25 \times 10^{-6}\text{V}^2$ prior to being sampled.	0.1014	4.79×10^{-6}
Harmonics Input Noise	Input has amplitude $a = 1$V and is corrupted with 3 harmonics at 40 kHz, 60 kHz, & 80 kHz, all with an amplitude of -60 dB.	0.0487	0
AM Demod/ Controller Noise	Each position signal has amplitude $a = 1$V and noise $\sigma^2 = 6.25 \times 10^{-6}\text{V}^2$.	≈ 0	2.65×10^{-3}
	Totals	0.15 μg	2.7×10^{-3} $\frac{\text{ft}}{\text{s}\sqrt{\text{hr}}}$
	Design Specification	1.0 μg	1.4×10^{-2} $\frac{\text{ft}}{\text{s}\sqrt{\text{hr}}}$

Table 7.1: Acceleration Estimation Bias Errors and Velocity Estimation Random Walk.

terms of the implementation. Amazingly, within hours of the first-time programming of the FPGA which implements all the digital blocks discussed in this thesis, the system was working in its ability to both amplitude and frequency demodulate test

signals. Even more amazingly, within minutes of the first time we put a sensor in the system and turned on the power, the loop closed and the controller maintained the oscillation of the two resonators.

One reason for this ease in implementation was that the complete system was modeled and simulated prior to actually testing it in the lab. All the analog circuitry and mechanical systems were modeled in VHDL (VHSIC Hardware Description Language), and then the digital logic, which was also implemented in VHDL, was tested via simulation. This enabled most of the debugging to be done in simulation prior to actually testing it in the lab.

We are using a Xilinx XCV800 FPGA for this design, and because of the modularity that VHDL allows, each block within the gate array can be synthesized individually. This allows one to see how large various parts of the design are. The Xilinx part we used contains 9408 slices, and two slices compose a Xilinx CLB (Combinatorial Logic Block). This FPGA has 888,439 system gates, which means the slice count which the Xilinx software reports can be converted to an equivalent number of gates. In Table 7.2 a summary of the slice count, the equivalent gate count, and the size percentage of each block is presented. The synthesis tool that we used was Synplicity, and it really did a great job at efficiently synthesizing this design.

This shows that the filter block is only twice as large as the CORDIC block, which means the filters block is quite small considering its functionality. The reason is that the Xilinx gate array has on-chip RAM which is where the tap-delay lines of the filters are stored.

This table shows that we are using approximately half a million gates. It also shows that if area ever becomes a concern, one of the first places to simplify is the Interfaces block. This block is a very versatile block which allows the user to control the functionality and the state of the system through a computer, and it also contains several communication layers that allow data to be sent to a computer. It has been designed for versatility in testing and debugging the system, and so it can be simplified greatly as the need for its functionality decreases.

Most of the blocks in Table 7.2 have been highly optimized for size. Some of the

Component	Description	Slice Count	Gate Equivalence	Percent
Filters	• 2 - Folded Hilbert Transform Filters (47 Taps, 16 bit delay line, 16 bit coefficients) • 2 - Folded Direct Form Bandpass Filters (62 Taps, 12 bit delay line, 16 bit coefficients)	1099	103,783	11%
CORDIC	• 2 - CORDIC Implementations (20 bit internal registers, 20 bit arctan() lookup table, 16 iterations)	457	43,157	4%
Controller	• 2 - PI Controllers (24 bit floating point coeff, 16 bit input) • 2 - Σ-Δ Modulators (1^{st} order)	768	72,526	8%
Weighted Difference	• 2 - 9 bit $\times$ 24 bit multiplier • 1 - 24 bit adder • 1 - 24 bit subtracter	313	29,557	3%
Non-Linear Compensator	• 9^{th} order polynomial (24 bit input, 24 bit floating point coeff)	786	74,225	8%
Interfaces	• 8 bit UART • Hybrid XModem Comm. Protocol • Control Logic • 24 bit Bus • Decimation Filters	1044	98,895	11%
Complete System	• Everything Mentioned Above • Timing Logic • A/D Control Logic • Output Decimation Filter • Pseudo Random Noise Generator • Minor Glue Logic	5895	556,691	62%

Table 7.2: Summary of Resource Usage and Total Gate Count.

features used to ensure a small design are the following:

- Since each channel of the accelerometer has both a bandpass and a Hilbert Transform filter, and since multiplication is large in terms of combinatorial logic, these filters have been implemented in a processing engine which uses a single MAC (Multiply/Accumulate) unit. Therefore, the filter block contains only 2 MAC units, one for each channel. The system clock is not fast enough to allow a single MAC unit to be shared between all the filters, but if in the future the size of this design becomes an issue, this is one area where savings could be accrued by simply doubling the system clock or halving the sampling rate and sharing a single MAC between all four filters.

- We saw in Chapter 2 that the in-phase channel needs to be delayed to remain synchronous with the quadrature channel, and since the Hilbert Transform filter

already contains a delay line, it has been shared with in-phase channel. Thus, no extra hardware is used to delay the in-phase channel, and this results in substantial savings in the amount of RAM used.

- Since both channels have the exact same bandpass and Hilbert Transform filters, the filter coefficients are shared between the channels.
- The two CORDIC blocks share the small ROM which stores the angle lookup table.
- The Non-Linear Compensator implements a ninth order floating point polynomial. Real savings have been obtained by implementing all the multiplications within this block using an add/shift technique which works in much the same way a human would multiply large numbers by hand. The reason that a multiplication of this form is possible here is that the signals have been downsampled prior to being compensated, and so there are enough system clock ticks between samples to perform the multiplication this way. The filter block does not have sufficient time to use an add/shift multiplication scheme, and so it cannot benefit from the savings of not using a combinatorial multiplier.

If the size of this design ever needs to be reduced in the future, one area that could greatly reduce size is to reduce register widths. This, of course, reduces the precision of the system, but as shown in Chapter 4, these digital blocks are several orders of magnitude more precise than they need to be. Since area is not a concern now, there is no reason to try to reduce the size of one area of the system which will not need improved and redesigned even as other areas of the system are.

7.3 Data Collection Software

Part of this project involved building an RS-232 serial communication interface to a computer. The purpose of this interface is for collecting data and controlling the state of the system. I implemented a UART in the FPGA which took care of the low-level communication issues. Then at the data level I implemented a hybrid XModem

protocol which had 6 data channels. Two of these channels are 24 bits wide and run at 500 Hz. The other four are 16 bits wide and run at 125 Hz. This enables us to collect data at a rather high rate.

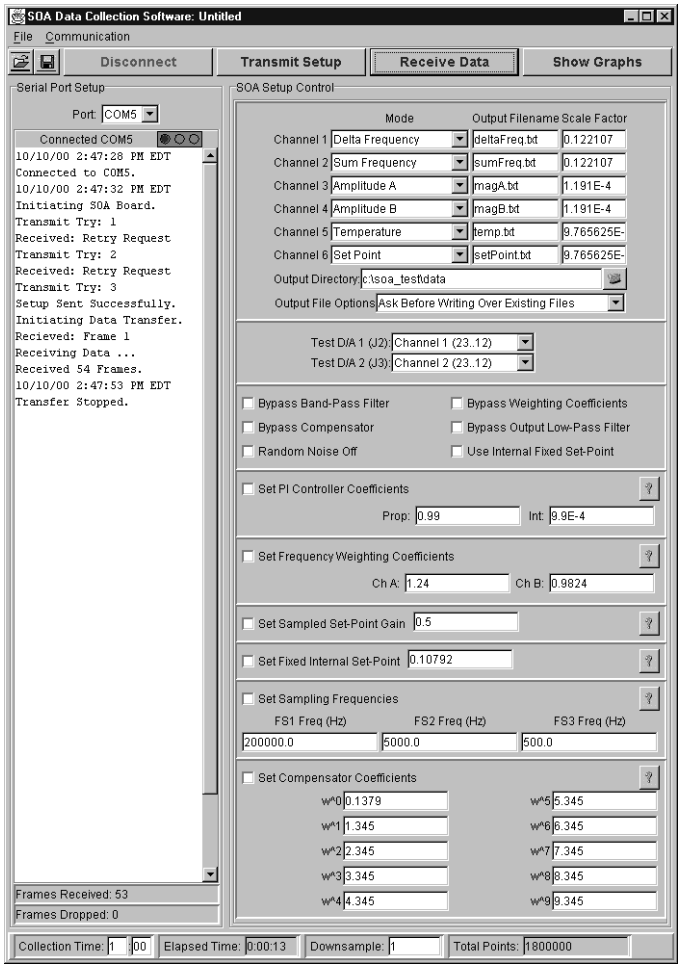


Figure 7-1: Data Collection Software

I also wrote the software on the PC that both controls the state of the system as well as receives the data. A picture of the GUI of this software is provided in Figure 7-1. I wrote it in Java and added features such as real time display of the data. This proved very useful in the debug stages and will be used by future test engineers as they rigorously test the system. This software allows the user to set the controller coefficients, frequency weighting coefficients, controller set point, controller set point gain, sampling rates, and compensator coefficients. Furthermore, it lets the user select to collect data on over 16 different internal registers including such signals

as amplitude, frequency, frequency difference, frequency sum, acceleration estimate, AGC, temperature, and set point.

7.4 Real Data

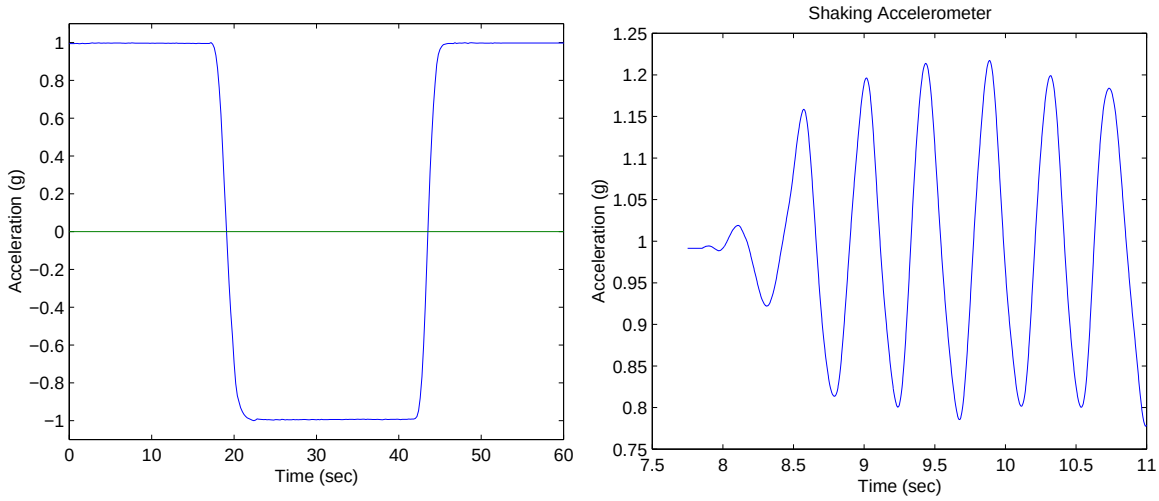


Figure 7-2: Data Collected From Accelerometer (a) Accelerometer Rotated 180° In Earth's Gravitational Field, (b) Accelerometer Manually Shaken.

The electronics have been tested and debugged at this point. The amount of acceleration data taken at this point is limited as others are working on packaging issues to make it possible to mount the sensor on a dividing head and test it more thoroughly. However, as shown in Figure 7-2, we have run simple functional tests of the accelerometer. The first graph starts with the accelerometer measuring the gravitational force of the earth, which is 1g. The sensor is then rotated 180° to measure -1g, and finally it is rotated back to measure 1g. The second graph shows the accelerometer initially positioned to measure the earth gravitational field (1g), and then it is manually shaken. Both of these functional tests verify that the sensor is working.

7.5 Next Stage

This project is just one part of a large effort at Draper Laboratory to build a high precision MEMS accelerometer. There has been a lot of work done by others prior to this project, and there will be much work to do after this. This includes rigorous testing of the instrument to find its limitations in terms of environmental stability and resolution, continued research and improvements in fabrication, and revising the mechanical structure to improve the deficiencies found during testing.

The electronics design as they stand now are very solid and stable and should not need any major revisions. The next step will be to migrate them to an ASIC. The digital electronics designed here are not the dominant source of noise and error in the electronics, so if future improvements are required, they will probably be to find ways of reducing noise in the charge amplifier which is the first stage analog amplifier of the position signal. It is the largest electrical noise source in the system.

Thus far, this project has been a large success, and I consider myself fortunate to have been involved with it. This has been a very interesting and aggressive design problem that has solidified and expanded much of my classroom education. Once again I would like to thank all those that made it possible.

Bibliography

[1] E. Antelo, J.D. Bruguera, T. Lang, and E.L. Zapata. Error Analysis and Reduction for Angle Calculation Using the CORDIC Algorithm. *IEEE Transactions on Computers*, 46(11):1264–1271, September 1997.

[2] L.A. Gould, W.R. Markey, J.K. Roberge, and D.L. Trumper. *Control Systems Theory*. 1997.

[3] Yu Hen Hu. The Quantization Effects of the CORDIC Algorithm. *IEEE Transactions on Signal Processing*, 40(4):834–844, April 1992.

[4] Clyde F. Coombs Jr. *Electronic Instrument Handbook*. McGraw-Hill, Inc., 1994.

[5] Kishore Kota and Joseph Cavallaro. Numerical Accuracy and Hardware Tradeoffs for CORDIC Arithmeic for Special-Purpose Processors. *IEEE Transactions on Computers*, 42(7):769–779, July 1993.

[6] Ali Hasan Nayfeh. *Nonlinear Oscillations*. Wiley, New York, 1979.

[7] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Signal Processing Series. Prentice Hall, New Jersey, second edition, 1999.

[8] L.R. Rabiner and R.W. Schafer. On the Behavior of Minimax FIR Digital Hilbert Transformers. *The Bell System Technical Journal*, pages 363–390, February 1974.

[9] Martin S. Roden. *Analog and Digital Communication Systems*. Prentice Hall, Inc., New Jersey, fourth edition, 1996.

[10] Jack E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Transactions On Electronic Computers*, pages 330–334, September 1959.

[11] J.S. Walther. A unified algorithm for elementary functions. *Spring Joint Computer Conference Proceedings*, 38(7):379–385, 1971.

[12] Conversations with Paul Ward of Draper Laboratory.