Control of a Fast Steering Mirror for Laser-Based Satellite Communication

by

Larry Edward Hawe II

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2006

Author .....
Department of Mechanical Engineering
January 20, 2006

Certified by .....
David L. Trumper
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by .....
Lallit Anand
Chair, Departmental Committee on Graduate Students

Control of a Fast Steering Mirror for Laser-Based Satellite Communication

by

Larry Edward Hawe II

Submitted to the Department of Mechanical Engineering
on January 20, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

MIT Lincoln Laboratory has been contracted by NASA to test and build a platform capable of sending and receiving laser communication signals in space from Mars. The two main components of the pointing system on the spacecraft include an inertial reference frame to provide coarse laser control and a Fast Steering Mirror to remove any spacecraft jitter from the optical path. The optical path in the satellite must have no more that 400 nanoradians RMS motion from 1 Hz to 1 kHz. This thesis focuses on the feedback control of this Fast Steering Mirror (FSM).

The feedback on the FSM comes from two different set of sensors. On power up, the FSM's angular position is controlled with feedback from local position sensors (KAMAN eddy current sensors). Optical feedback is accomplished with a laser beam and quad cell optical sensor. The optical sensor has an extremely small range of operation, and the mirror must first be pointed onto the active area of the quad cell before the optical feedback can be activated. This thesis investigates the controller being used for this FSM, the feedback loops for the different sensors and the pointing algorithms used to switch between feedback sensors.

The analog control system in use has a crossover frequency of approximately 1 kHz. MIT Lincoln Laboratory and NASA would like to use an FSM with a closed loop bandwidth as high as possible to lower noise restrictions on other parts of the spacecraft. This thesis investigates the FSM dynamics in detail and applies a different control system to push out the bandwidth as far as possible.

Thesis Supervisor: David L. Trumper
Title: Professor of Mechanical Engineering

Acknowledgments

Please do not be fooled into thinking that this body of work is mine alone. It might take a village to raise a child, but it takes a small army to finish a Master's thesis. Without the help and support of the following people, this thesis simply would not exist.

Working at Lincoln Laboratory over the last year has been a great experience. Not only did I get to work on an interesting project, but I also has access to incredible resources. However, all of the resources at my disposal would mean very little without the people who taught me how to use them. Thanks to Jason LaPenta for getting me up to speed with his quad cell and the supporting electronics. To Cathy DeVoe for walking me through the subtleties of optics (which still seems like magic to me). Thanks to Al Pillsbury for his help with the mechanical aspects of the FSM. Chad Ware helped with the hardware for the MIRU demonstration for NASA, which while having little to do with the FSM, was still incredibly important for me keeping my job. Thanks to David Baron for putting up with me turning out the lights in the lab everyday, forcing him to work in the dark. Thanks to Rick and Karen down in Electronic Stock for taking care of all my electronic hardware needs, which were very large. Special thanks to Paula Ward for maintaining the ESD Lab where I spent 99% of my time, and for taking the photos of the hardware. Very big thanks go to Professor James K. Roberge, who splits his time evenly between Lincoln and MIT. He sat down with me and went through his entire compensation circuitry, as well as taught (along with Dr. Kent Lundburg) the most useful electronics class I ever took. Special thanks also to my supervisor, Jamie Burnside. Jamie showed me the ropes at Lincoln, honestly appraised my work and even wrote most of the first spiral code. It has been a privilege working for him. Thanks also to the Group 76 Leader, Ed Corbett for giving me the opportunity to work in Group 76, and thanks to Annmarie Gorton for taking care of all my administrative needs, especially getting this thesis through the release process (without which no one would ever read this).

When I wasn't working at Lincoln, I was probably working at the Precision Motion Control Laboratory at MIT. The guys at the PMC Lab made my work seem less stressful and added the necessary degree of fun and entertainment to my life, preventing (or at least delaying) the onset of insanity. They all have also helped me with almost every aspect of my graduate career, from class selection to LaTeX typesetting. For all this and more, my thanks to the PMC students: Emre Armagan, Marty Byl, Augusto Barton, David Cuff, Dan Kluk, Xiaodong Lu, Dave Otten, Aaron Mazzeo, and Rick Montesanti.

My entrance into the PMC lab was facilitated by Professor David L. Trumper, my academic advisor for the last 4 and a half years, as well as my thesis advisor. Professor Trumper gave me the support necessary to complete this degree, providing lab space and funding. He has taught me more in the meetings we have than I have learned in a month in some of my classes. He also made typesetting this document much easier by exposing me to LaTeX during my TA days, making the writing part of my thesis go exponentially faster. It has been a privilege to work for him, and I

thank him for the opportunities he has given me during my academic career. Thanks also to Laura Zaganjori, Denise Moody and Maggie Sullivan, three wonderful women who have taken care of the administrative needs of the PMC and made my life that much easier.

Graduate students have a certain number of hoops to jump through during their tenure at MIT. The ease of jumping through those hoops is often controlled by the personnel of the Graduate Office. Leslie Regan has been remarkable by making the entire hoop jumping process virtually painless. She has constantly reminded me of deadlines and generally made sure that I would graduate, no matter how hard I tried to not do so. Without her, my graduate experience would have been much rougher, and I thank her for all of her hard work taking care of the ME grad students.

With all of this academic support, one would think I had no problem with this thesis. However, my family has kept me grounded throughout these years and they deserve special thanks. When I was stressed out, my Mom's laugh always picked up my spirits. My Dad has been my inspiration and my ultimate measuring stick. Thanks to Diana Schroeder, my third maternal figure, who helped me with my dreadful entrance essays (single handedly saving my entrance application). And to the rest of my family: Jamie Hawe, Alisha Ledbetter, Karla Dornhecker, Tony Shelton, Angelina Stowe, Christina Mullan, the Utter family, the Langfords, Viola Lucas and all the rest: I say thank you so much for supporting me during this journey.

There is one member of the PMC I have failed to mention. The "me" and "I" in this document is actually not me. A doctoral student has worked with me extensively over the past year. He has helped my in this project in more ways than I can count, from bringing me up to speed on the project to tutoring me in the finer points of control system theory. He has shared his knowledge of control system theory with me and helped me apply it to this document. He has provided a gentle nudging in regards to doing the work, and he is big reason for this document's existence. This has truly been a team effort. There is a special bond between us that could only be forged after countless hours of debugging circuitry at 4 in the morning. His help has been invaluable, and it truly has been an honor working with him. My sincere thanks to my colleague and friend, Joe Cattell.

Finally, I have to thank my girlfriend of the last four plus years, Darlene Utter. Darlene has supported me in so many ways while I have been in school that listing them would be an exercise in futility. She has taken care of our day to day needs for the last year and a half while I have been in school. She has also been severely neglected, especially during these last few months while I finished this project. I look forward finally spending some time with her, and to helping her get through her own Master's program at MIT. Above all, I look forward to sharing my life with her.

For Dennis Dean Dillon

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 MarsComm Summary

In an effort to better understand the solar system, NASA and the United States government have decided to send several unmanned vehicles to Mars in the next decade. Current communication between the Earth and the orbiters takes place via high frequency radio transmission. This technology is several decades old, and has the advantage of being space proven and relatively easy to implement. However, even with directional antennas, much of the power used in radio communication is wasted as the waves radiate in many directions, with only a small percentage of the transmitted power reaching the receiver. NASA has decided to implement a laser-based communication system designed to achieve higher bandwidth data transfers with much less wasted power. The Mars Laser Communication Demonstration (MarsComm) program is the implementation of this laser-based communication onboard an orbiter.

Unlike radio waves, lasers can be highly focused at the transmission point. This is due to the difference in relative wavelengths. Deep space radio transmissions operate near the GHz range, giving them a wavelengths between 10 m and 10 cm, where lasers operate in the infrared or visible spectrums with wavelengths between 100 μm and 400 nm. The shorter wavelength lasers can be better focused, giving them a tighter angular dispersion pattern and greater power densities (W/m^2) for the same transmission power and distance. This allows for much less power to be used in

communication, but introduces a problem of pointing. The less tightly confined radio waves do not have to be pointed at the receiver more accurately than a few degrees, but the tighter confined laser must be pointed with much greater accuracy. The lasers used in MarsComm disperse to a diameter of approximately $1/6^{th}$ the Earth's radius over the trip from Mars to Earth. To hit the communication ground station on Earth, the laser must be pointed with no more than 400 nanoradians RMS error between 1 Hz and 1 kHz.

This pointing accuracy requirement one of the key challenges for making MarsComm a viable technology. Several pieces are required to make the system work. A system diagram is shown in Figure 1-1. This schematic shows the signal path of a laser beam emitted from the Earth. The main parts of the system include a Magnetohydrodynamic Inertial Reference Unit (MIRU), a Fast Steering Mirror (FSM), a quad cell optical detector and a focal plane array (FPA).

1.1.1 The MIRU

The MIRU provides an inertial reference unit for the spacecraft. This is accomplished by using Magnetohydrodynamic (MHD) sensors. The MHD sensors used, Part # ARS-12B, are produced by Applied Technology Associates (ATA)[1]. These sensors provide angular rate information from 2 Hz to about 1 kHz. Because the MHDs are rate sensors, the system is AC coupled. Angular position information below 2 Hz is provided by the FPA, an IR sensitive imaging unit. This low frequency information is blended with the MHD sensors to provide feedback from DC to about 1 kHz. Using this feedback information, the MIRU is able to provide an inertially stable reference frame aboard the spacecraft with the crossover frequency of about 300 Hz.

1.1.2 The Fast Steering Mirror

The MIRU provides an inertial reference on board the spacecraft, but it does not correct for the error between the spacecraft motion ("jitter") and its inertially stable motion. This correction is accomplished with the Fast Steering Mirror. The MIRU

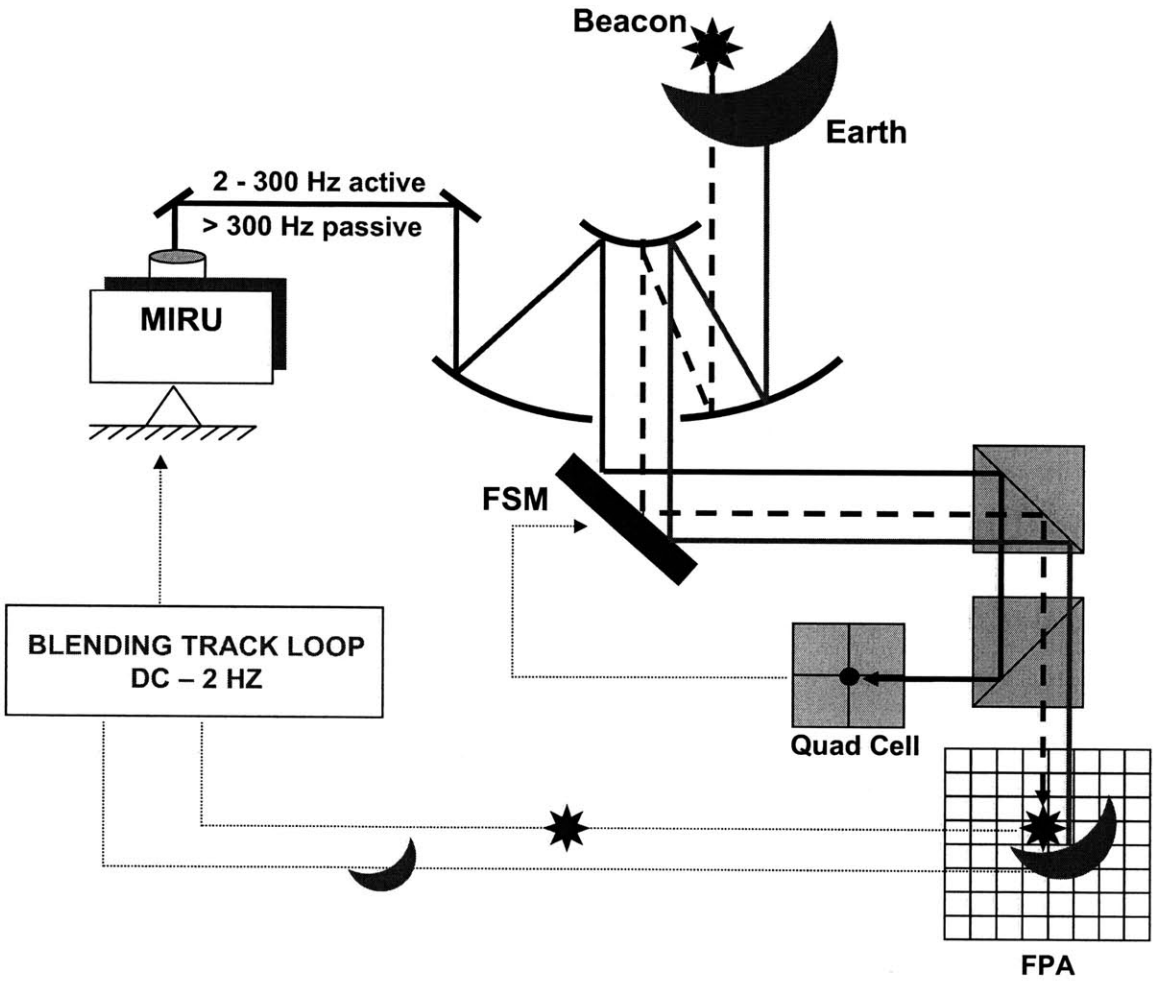

Figure 1-1: Simplified MarsComm spacecraft signal path schematic, courtesy of Jamie Burnside.

beams a laser along the same optical path as the Earth bound laser beams. This beam bounces off of the FSM and hits a quad cell detector. At the zero state with no relative motion between the Earth and the spacecraft, the MIRU will not be active, the FSM will not be active and the laser will hit the center of the quad cell. As the spacecraft jitter introduces angular error in the pointing of the beam, the MIRU stays inertially stable, and the FSM uses this inertially stable reference beam to zero out the quad cell and restore the optical path as if the spacecraft had not moved. The current control system for the FSM has a crossover frequency of 1 kHz and a closed loop bandwidth of 2 kHz. If the bandwidth of the FSM could be pushed up, it would decrease the RMS angular error of the system and make the other system component noise requirements lower, leading to lower costs. Before redesigning the FSM hardware, it is prudent to investigate if the performance can be improved via a better controller design that can be easily integrated into the system.

1.2 Thesis Overview

The goal of this thesis is to find the best controller for the FSM, given its hardware characteristics.

Chapter 2 describes the rest of the hardware used in the experiments. Several testing configurations are presented along with an overview of the integration of the hardware with the software control system.

Chapter 3 focuses on the actual FSM hardware. Prior research and design of the FSM is presented, including individual component descriptions along with schematics. Detailed open loop frequency analysis of each axis is presented along with parametric models for computer simulations.

Chapter 4 details the design of a spiral acquisition algorithm used to initially point the FSM onto the active region of the quad cell, used to switch between KAMAN and quad cell control.

Chapter 5 details the control theory behind the current analog compensator in use. The design of an original compensator to push the FSM to its hardware limits

is presented along with experimental results.

Chapter 6 details the conclusions of the controller design and recommendations for future work.

Chapter 2

Hardware Overview

This chapter describes the hardware used during the experimentation process. Section 2.1 describes each piece of hardware used during experimentation in detail, including electronic schematics where appropriate. Section 2.2 describes the physical layout of the hardware during experimentation, including wiring diagrams and pictures of the test setups. Finally, Section 2.3 describes how xPC was used to control the experiments, including the design and implementation of graphical user interfaces (GUIs) and the basic components of the Simulink models used.

2.1 Hardware Components

This section details all of the hardware used in the experiments controlling the FSM. Figure 2-1 shows the general layout of the experiment with all the relevant hardware. The ordering of this section follows the beam path of the laser in the experiment.

2.1.1 Laser Sources

The laser sources used in the experiment simulate the laser beam produced by the MIRU. Two separate laser sources were used in these experiments, as described below.

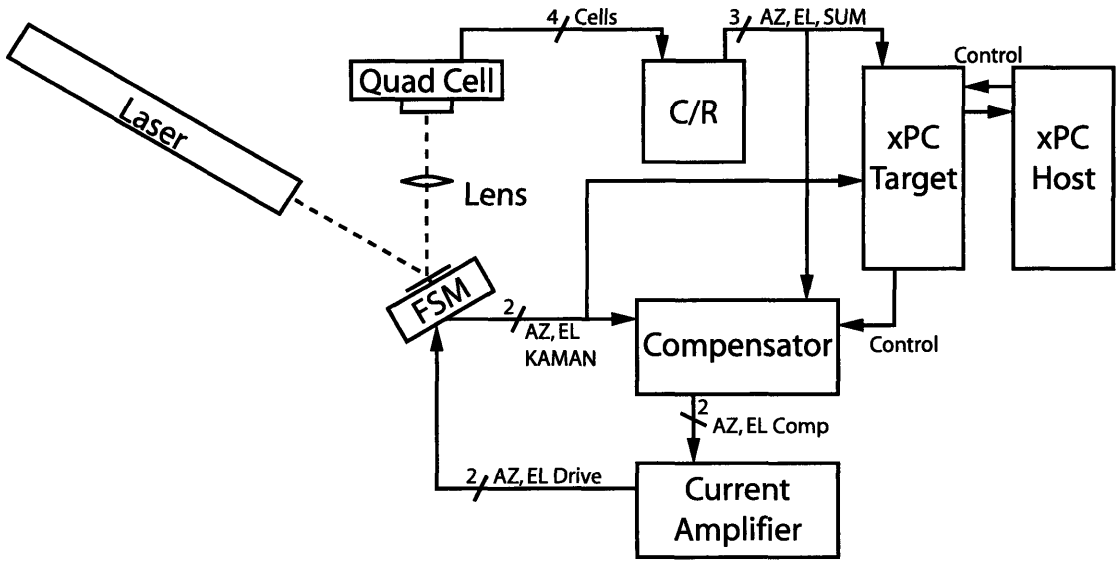

Figure 2-1: Diagram of the experimental hardware configuration with signal paths.

Fiber-Coupled Source

The Thorlabs S1FC675 fiber-coupled laser source was used in early experimentation. The MIRU is designed to use a fiber-coupled source, and thus using the source is a good approximation to the actual flight hardware. The laser outputs a red gaussian laser beam at 675 nm into a fiber. The output power of the laser source is controllable between 0 and 2.5 mW. Further attenuation of laser power is accomplished with a neutral density optical filter. The power level of the laser must be controlled so as not to saturate the quad cell and position indicator electronics (see Section 2.1.5). The output of the fiber is fed into a collimator, shown in Figure 2-3. The final output at the collimator is a circular gaussian beam with a diameter of approximately 5 mm.

Unfortunately, during experimentation, the fiber-coupled laser source introduced a variety of problems. The laser output cross sectional power (the spot size on the quad cell and the distribution of power on that spot) varied greatly when the fiber was moved or jostled. This movement was not extreme in the short term; however, the drift was noticeable after several minutes. The spot often drifted off the active region of the quad cell within 30 minutes. The cause of this drifting has not been

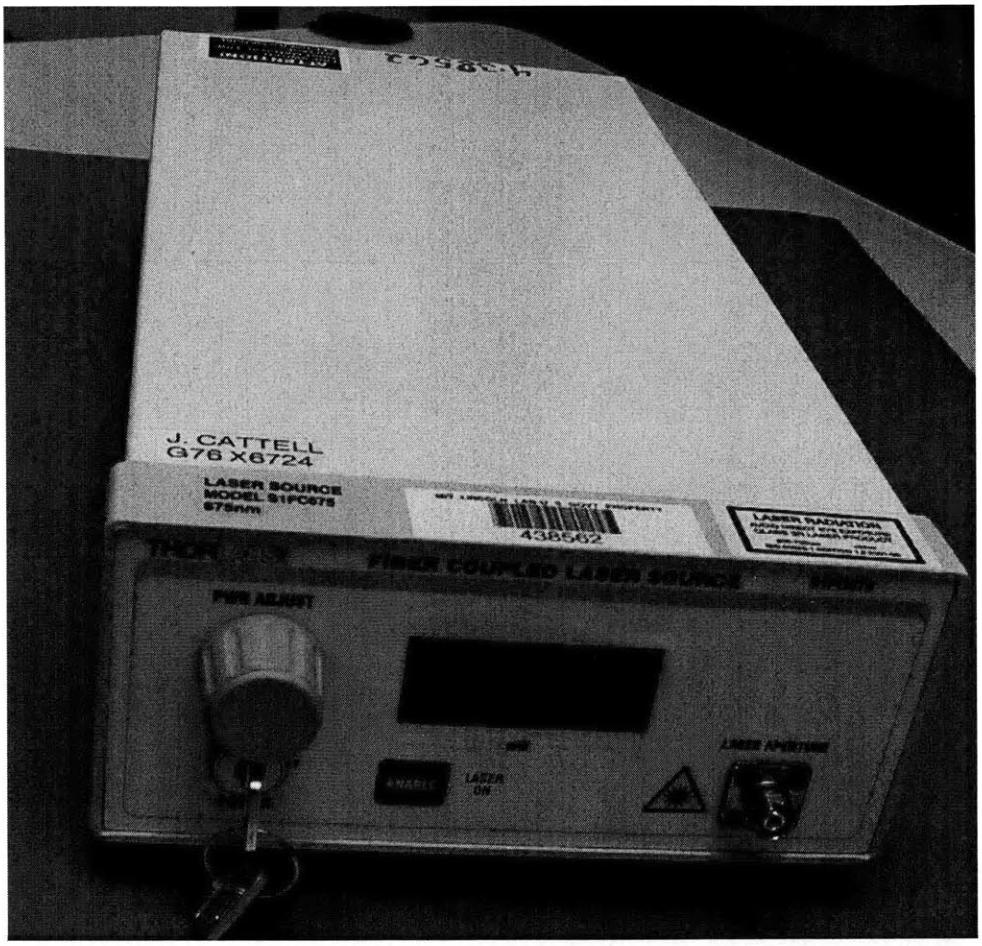

Figure 2-2: Picture of the Thorlabs fiber-coupled laser source, model S1FC675.

Figure 2-3: Picture of the collimator used with the fiber-coupled laser source in mount, courtesy of Paula Ward.

identified; however, due to time constraints with the project, a different laser source was implemented with much better results.

Helium-Neon (HeNe) Laser Source

The Helium-Neon (HeNe) Laser used during the experiments is shown in Figure 2-4. The laser is approximately 40 cm long with a tube diameter of approximately 7 cm. The laser is held in a mount at its midpoint and attached to the table. The mount allows the laser to be moved in two angular axes. This speeds up alignments procedures, but is not necessary in the final configuration of the hardware. With the mount in place, the natural frequency of the laser system is approximately 66 Hz (which was discovered after trying to debug 60 Hz noise). This is due to the fact that the mounting of the laser creates a large cantilever system, much like a see-saw. Placing the laser on blocks would eliminate this natural frequency, but would not allow adjustment of the laser orientation. The laser outputs a green (535 nm) gaussian circular spot approximately 1 mm in diameter at 20 cm from the laser aperture. The maximum power output of the laser is 5 mW, but has been measured to be no more than 3 mW nominal power. The output of the laser is attenuated by a neutral density optical filter directly after the laser aperture. The output power level is set by the quad cell limits (see Section 2.1.5).

2.1.2 The FSM

The laser source beam exits and reflects off of the Fast Steering Mirror. The mirror is used to control where the spot hits the quad cell. The FSM is discussed in detail in Chapter 3.

2.1.3 Focusing Lens

A lens is necessary to focus the laser beam onto the quad cell. The lens used in the experiments is shown in Figure 2-5.

The lens used is a standard convex lens with a focal length of approximately 15

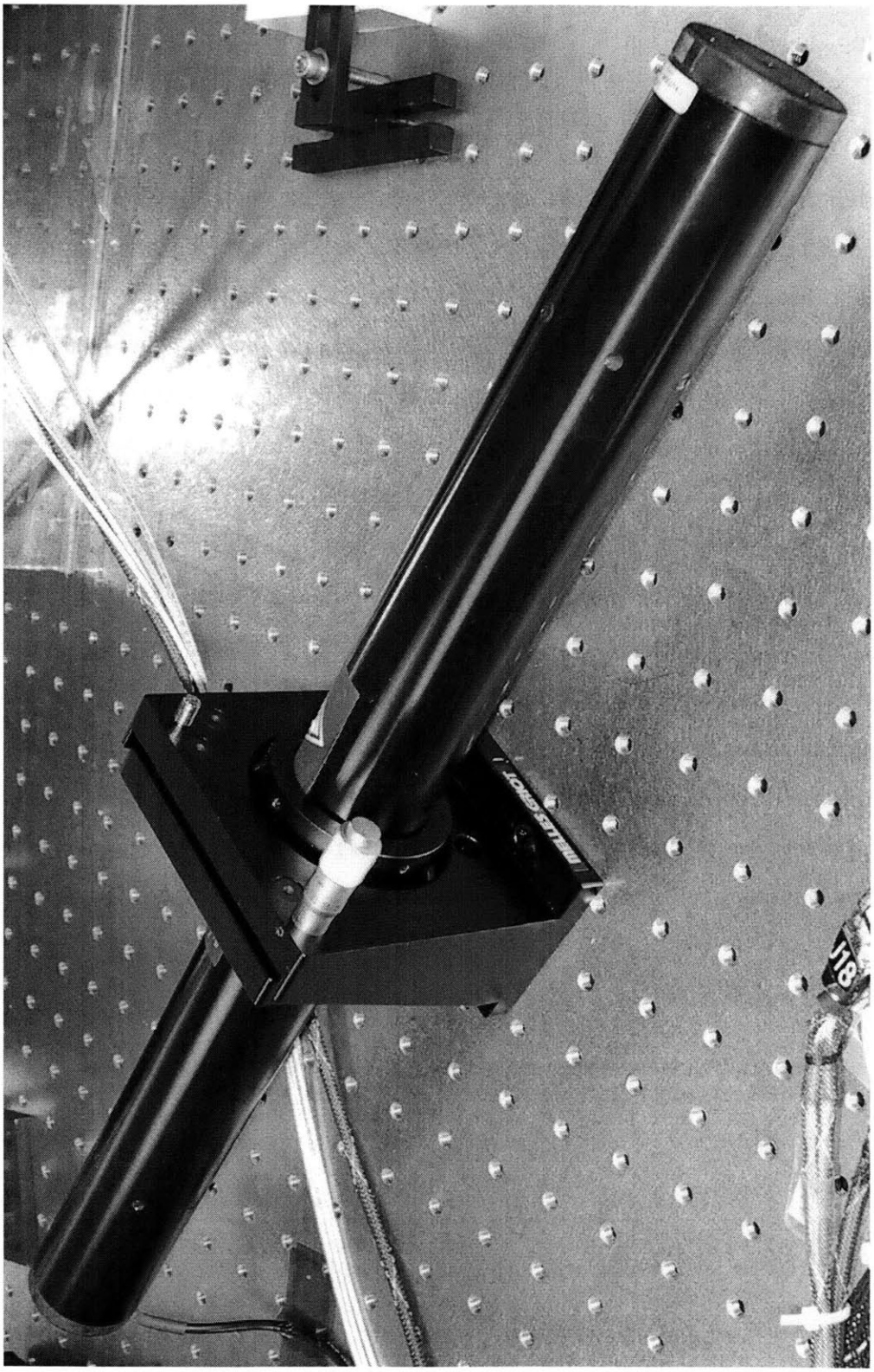

Figure 2-4: Picture of the Helium-Neon (HeNe) laser source with 2 axis adjustable mount.

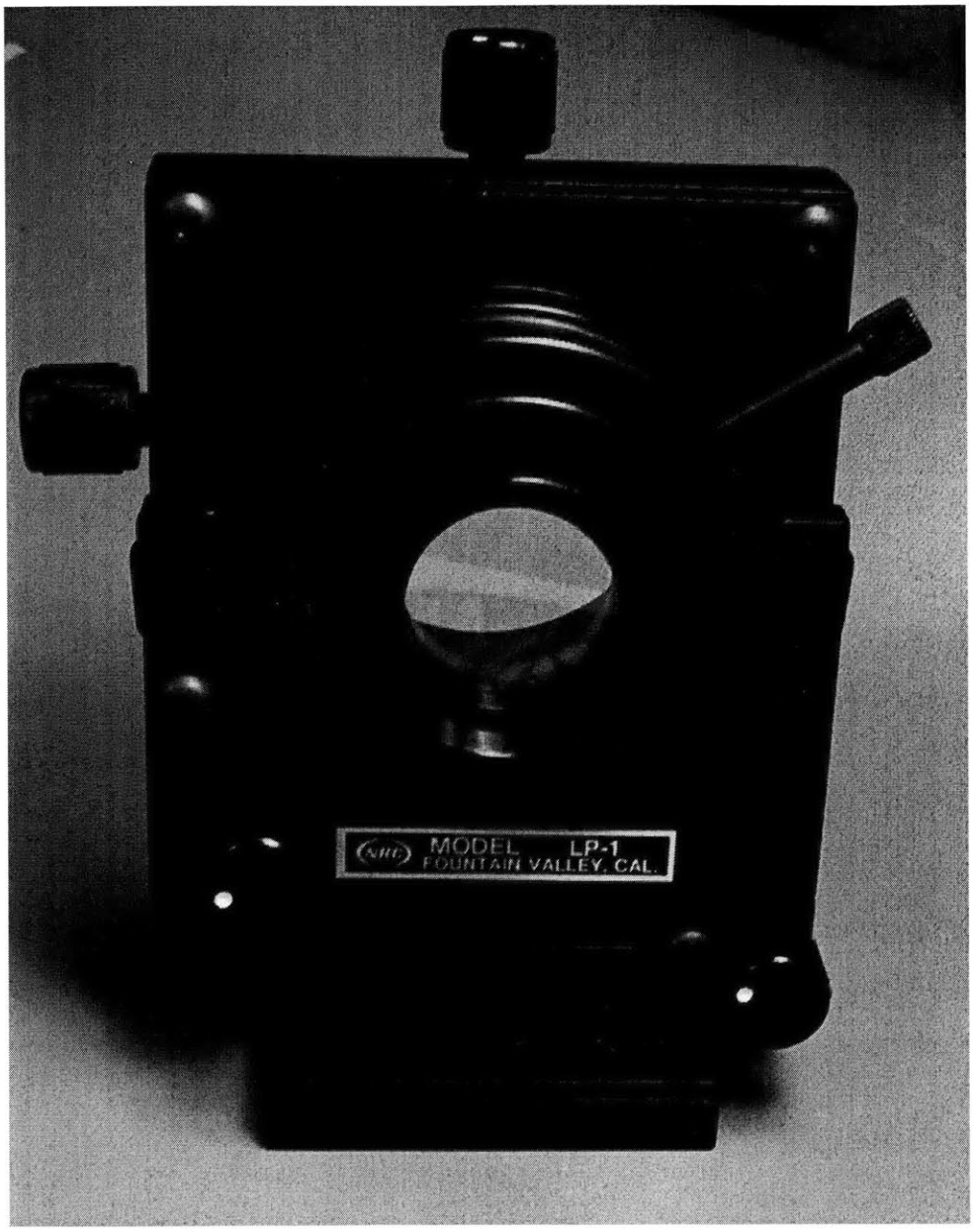

Figure 2-5: Picture of the focusing lens with 5 axis adjustable mount.

cm. This lens focuses the laser beam onto the active area of the quad cell (which is approximately 1 mm^2). The mount for the lens allows manipulation of the lens in 5 different axes. This mount was necessary in the testing phase to keep the FSM near the center of its operating region without having to realign the optical hardware. However, skewing can occur if the lens must be placed at gross angles relative to the quad cell. This was not a problem during experimentation because the HeNe laser was adjusted first to eliminate gross angular errors and the lens mount was only used for fine adjustments.

2.1.4 xPC Control

Control of the hardware in the experiments is handled by a part of MATLAB called xPC. xPC is actually run on two separate computers during an experiment.

xPC Target

One computer, known as the xPC Target, runs a very basic kernel of xPC, bypassing all the software complexities of Windows and other operating systems. This bare kernel allows the computer to run programs at very high sampling rate. The programs that run on the Target are created in Simulink. The Simulink models run on the Target PC, simulating the desired system, such as a basic controller or a data acquisition device. Input and outputs are passed through between 1 and 4 General Standards PMC-ADADIO cards. Each card has 8 Analog Inputs, 4 Analog Outputs and 8 Digital I/O channels. Standard operating frequency of the xPC Target Simulink model is 10 kHz, but simple models can be run as high as 50 kHz.

xPC Host

While the xPC Target is running, parameters in the Simulink model can be changed in real time to alter the functionality of the model. This control is accomplished via an ethernet link between the xPC Target PC and an additional PC, called the xPC Host. MATLAB is running on the xPC Host machine, and parameters can be

changed on the Target machine via the MATLAB command line. The model running on the Target PC can also be started, stopped or reloaded via the xPC Host PC. For ease of use, graphical user interfaces (GUIs) can be written in MATLAB and run on the Host PC to control the Target PC.

2.1.5 The Quad Cell

The quad cell is the heart of the optical tracking loop. The quad cells used in these experiments are all SPOT-4DMI quad cells made by UDT Sensors, Inc. The quad cell itself is simply 4 very small, very sensitive photodiodes arranged in a small grid. Figure 2-6 shows the schematic of the quad cell.

Each active element of the quad cell (A, B, C or D) has an active area of 0.25 mm^2, giving a combined total active area of 1 mm^2. Of this active area, only the middle 25% is linear. Non-linearities occur past the inner 25%, becoming extreme past about 75%. The gaps between the elements are 13 μm wide. Each of these 4 elements outputs a voltage proportional to the amount of light hitting it. There are two packages used in the experiments. These two packages are shown in Figures 2-7 and 2-8. Each package contains the same SPOT-4DMI quad cell, but outputs to a different conversion board. In order to calculate the position of the laser beam, the 4 element voltages outputted from either package must be converted into Azimuth and Elevation signals. This task is accomplished via conversion hardware.

The quad cell in Figure 2-7 is mounted at a 45° angle in the flight box (due to packaging constraints). Figure 2-9 shows the orientation of the quad cell. Because of this packaging, the converted Azimuth and Elevation signals must be rotated by 45°.

2.1.6 Conversion/Rotation Hardware

The quad cell packages shown in Figures 2-7 and 2-8 both output 4 element voltages corresponding to the intensity of light on each photodiode. This information must be converted into Azimuth and Elevation position information to be useful to the controller hardware. Each package requires a different method of conversion, and if

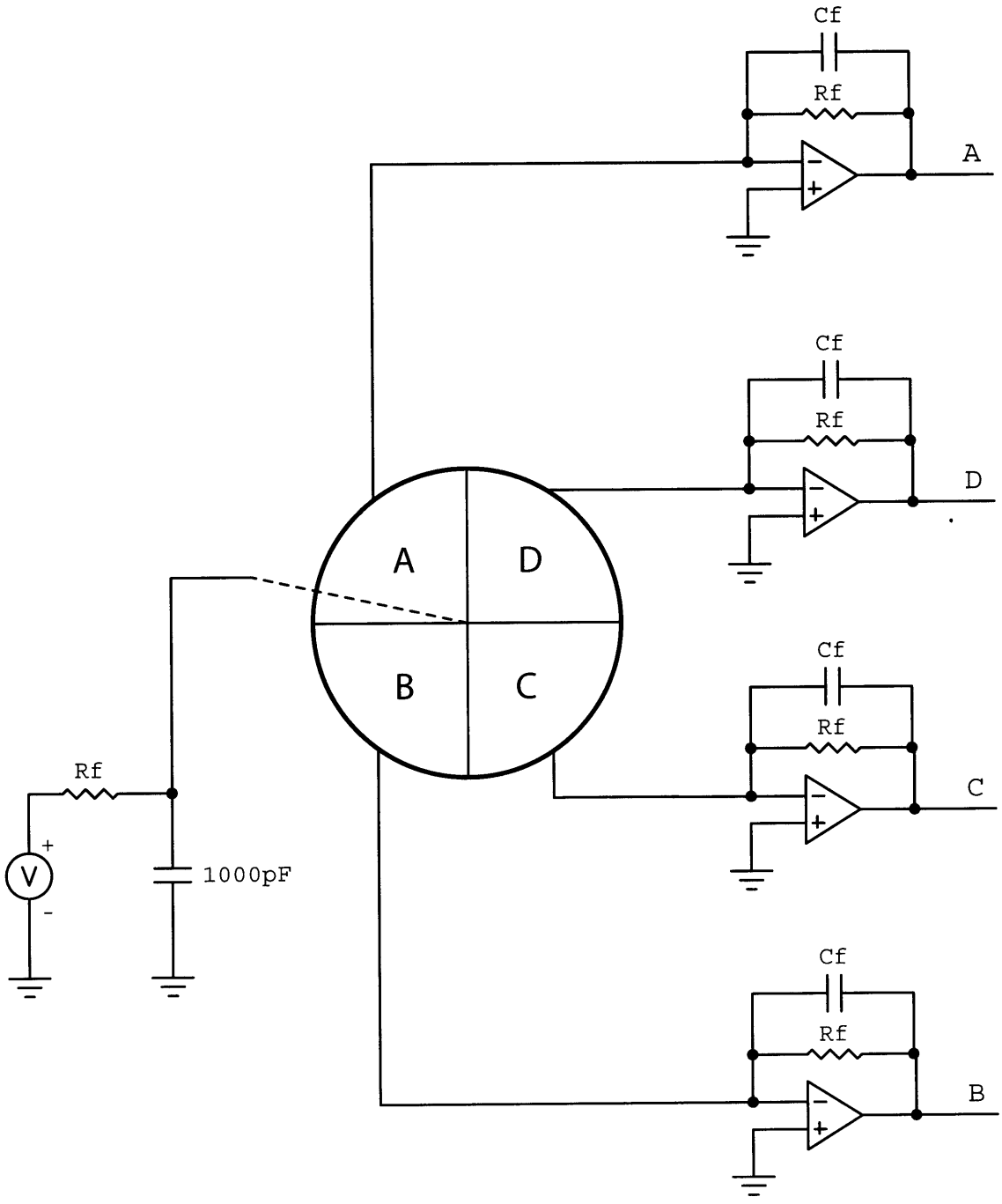

Figure 2-6: Generalized schematic of SPOT-4DMI quad cell showing photodiode elements and output signals.

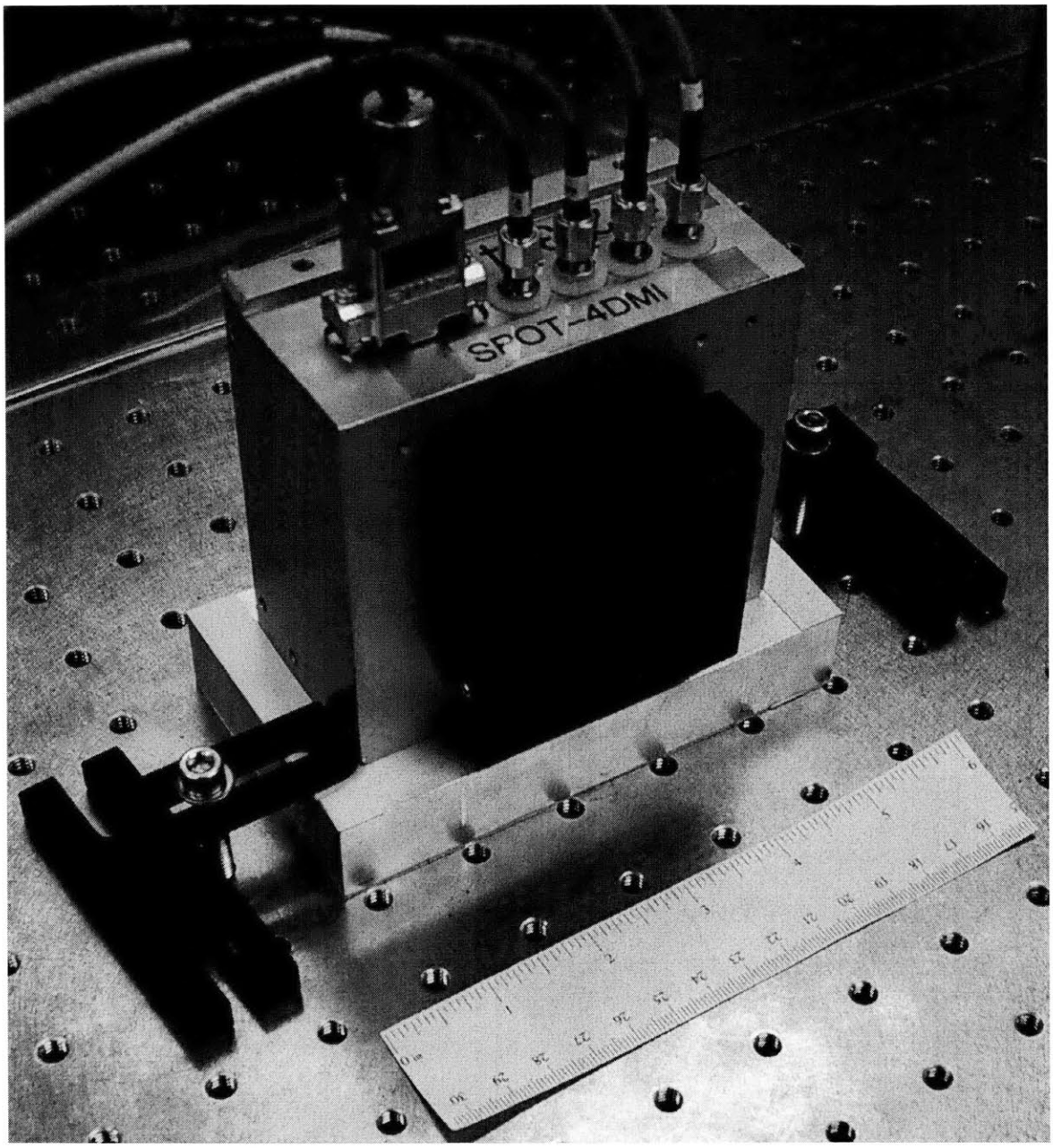

Figure 2-7: Picture of the SPOT-4DMI quad cell, mounted in flight box.

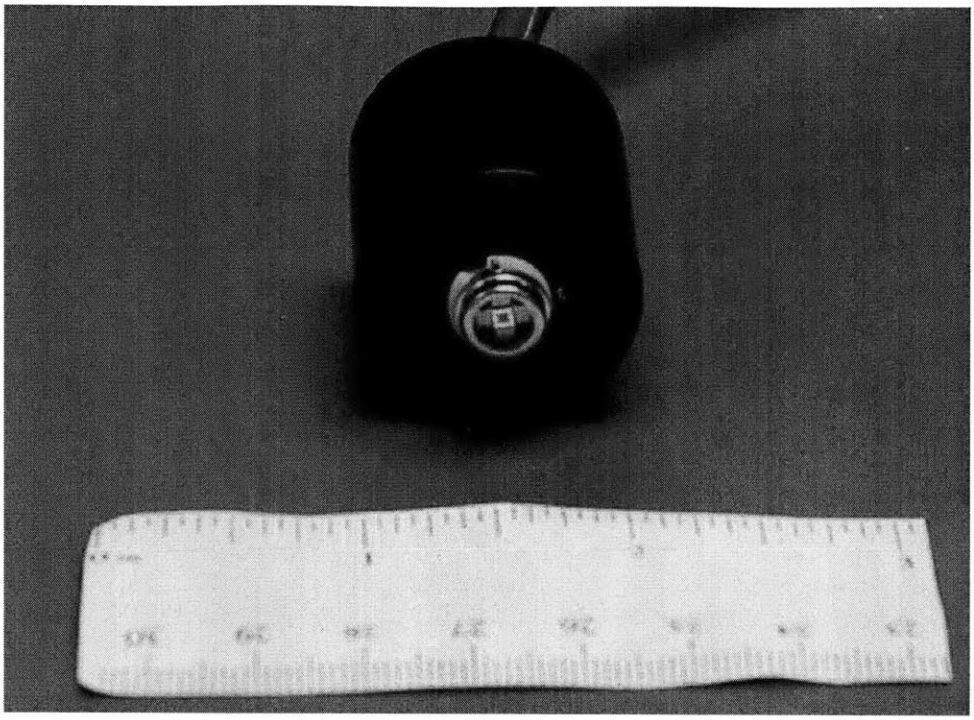

Figure 2-8: Picture of the SPOT-4DMI quad cell, mounted on plastic housing for use with the UDT Model 431 X-Y Position Indicator.

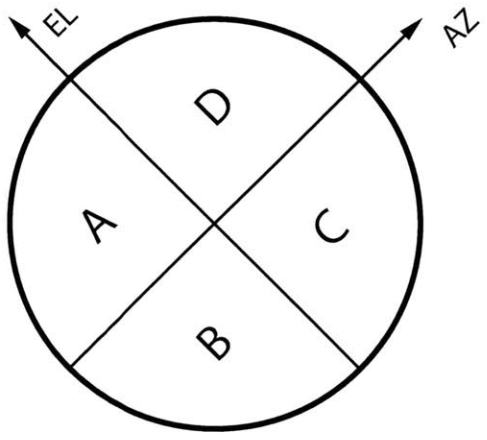

Figure 2-9: Orientation of SPOT-4DMI quad cell in flight box of Figure 2-7.

necessary, rotation.

Conversion Algorithm

The basic math required to obtain Azimuth and Elevation from the quad cell is straightforward. Using the orientation of the quad cell elements as shown in Figure 2-10, the basic Azimuth and Elevation voltages are as follows:

$$AZ = \frac{(C+D)-(A+B)}{(A+B+C+D)} \tag{2.1}$$

$$EL = \frac{(A+D)-(B+C)}{(A+B+C+D)} \tag{2.2}$$

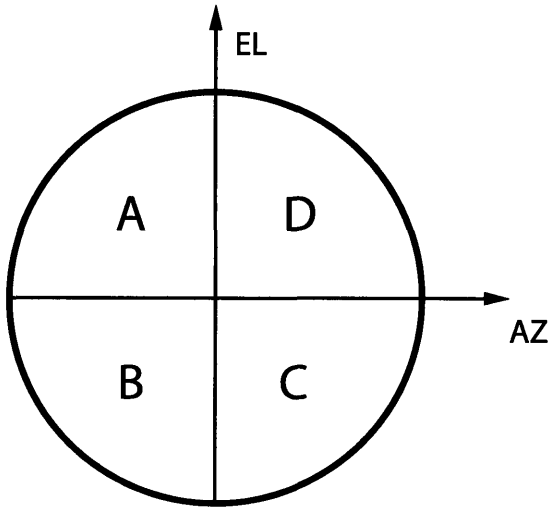

Figure 2-10: Diagram of the SPOT-4DMI quad cell's active elements with labeled axes.

Equations 2.1 and 2.2 are easy to understand. If the laser spot was circular and focused directly at the center of the quad cell, the voltages on each element would be the same and the Azimuth and Elevation voltages would both be zero, as expected. This reasoning would hold no matter what the size of the spot was (as long as it was all contained in the active region of the quad cell). Furthermore, the reasoning still holds if the spot is not circular, but rather a skewed ellipse. This works because

the Azimuth voltage is essentially the left side of the quad cell minus the right, and the Elevation is the top minus the bottom. A centered ellipse would have equal components left and right, as well as top to bottom, and thus would still register zero volts for both Azimuth and Elevation. For this reason, quad cells are said to be "centroid" friendly, as they tend to output the centroid of a spot, no matter its specific shape.

UDT X-Y Position Indicator

Figure 2-11 shows the UDT Instruments Model 431 X-Y Position Indicator used with the quad cell package shown in Figure 2-8. This position indicator is a single box solution for quad cell signal conversion. The quad cell is plugged into the position indicator, and the hardware outputs the Azimuth, Elevation, and Optical Sum voltages. The Optical Sum voltage is the sum of the voltages on each of the 4 photodiodes, and is useful in determining if the beam is on the active region of the quad cell. Azimuth and Elevation are outputted analog voltages, ± 1 Vpk. The Optical Sum voltage is always positive, ranging up to 10 Vpk.

The 431 Position Indicator has a front end amplifier designed to allow adjustment for different incident beam strengths. This amplifier allows the use of different lasers or different laser powers without adjusting the neutral density filtering. This amplifier should be set so that the Optical Sum voltage is between 300 and 1000 on the Sum display.

Rotation is not possible with the Model 431. The quad cell package must be aligned to the FSM in experiment. This is more difficult than it would seem, and the mounting hardware for this quad cell Package was very cumbersome. The original mounting had 5 degrees of freedom, leading to the addition of several low frequency modes that interfered with data collection. This setup was used only very early on in testing, giving way to the quad cell flight package shown in Figure 2-7. However, the flight package is not compatible with the Model 431 and a Conversion/Rotation Board had to be built to interface with the quad cell.

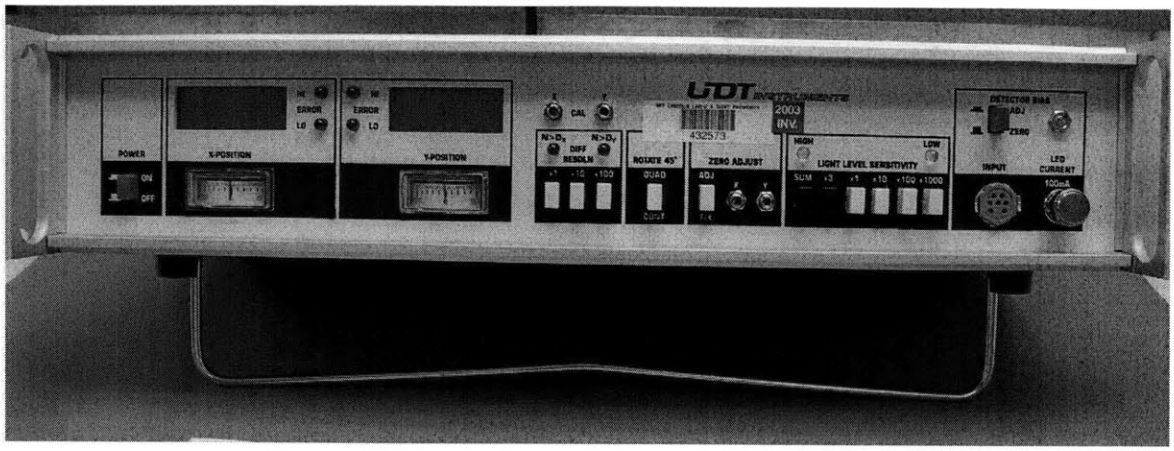

Figure 2-11: Picture of the UDT Instruments Model 431 X-Y Position Indicator.

C/R Board

As was stated before, there is no commercial solution to convert the 4 element outputs of the quad cell flight package shown in Figure 2-7. However, Lincoln Laboratory has designed hardware for this exact purpose. The hardware is flight approved, but was not available for use during this project. Instead, the schematics were made available, and after small modifications, an analog Conversion/Rotation board was built. This board is shown in Figure 2-12.

The bottom half of the C/R Board handles the conversion from 4 element voltages to non-rotated Azimuth and Elevation voltages. The schematic for the conversion is shown in Figure 2-13. The inputs to this part of the board are the 4 element voltages A, B, C, and D. The outputs are the two non-rotated Azimuth and Elevation voltages, AZ_NR and EL_NR respectively. The Optical Sum voltage, Q_SUM, is outputted at the top of the board as well.

The top half of the C/R Board handles the rotation of the Azimuth and Elevation signals from the conversion half of the board. As stated before, this rotation is necessary because the SPOT-4DMI quad cell is mounted at a 45° angle in the flight box. A simple rotation matrix is used to rotate the two signals. To rotate a set of

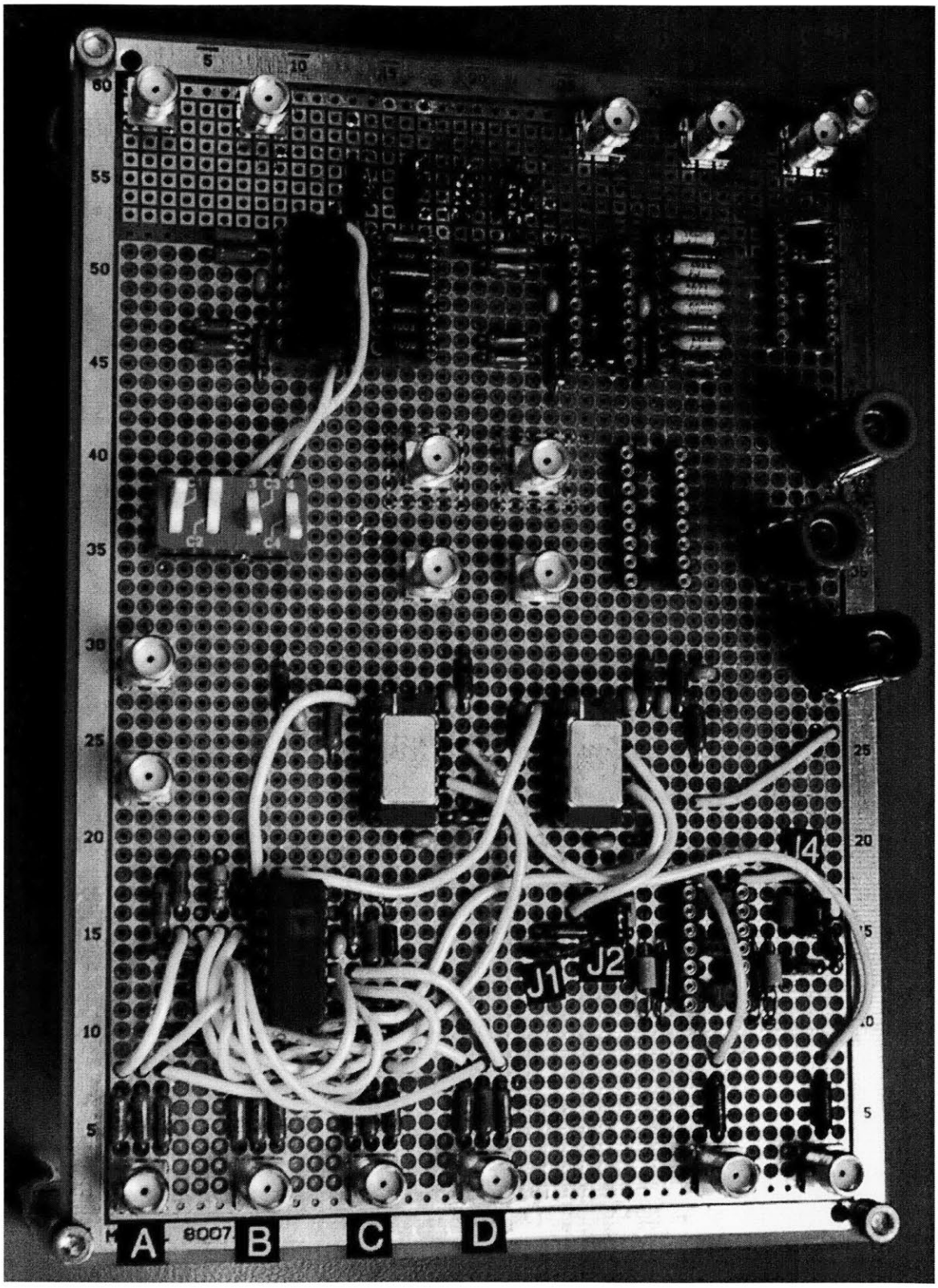

Figure 2-12: Picture of the Conversion/Rotation Board as configured for use with digital compensator, courtesy of Paula Ward.

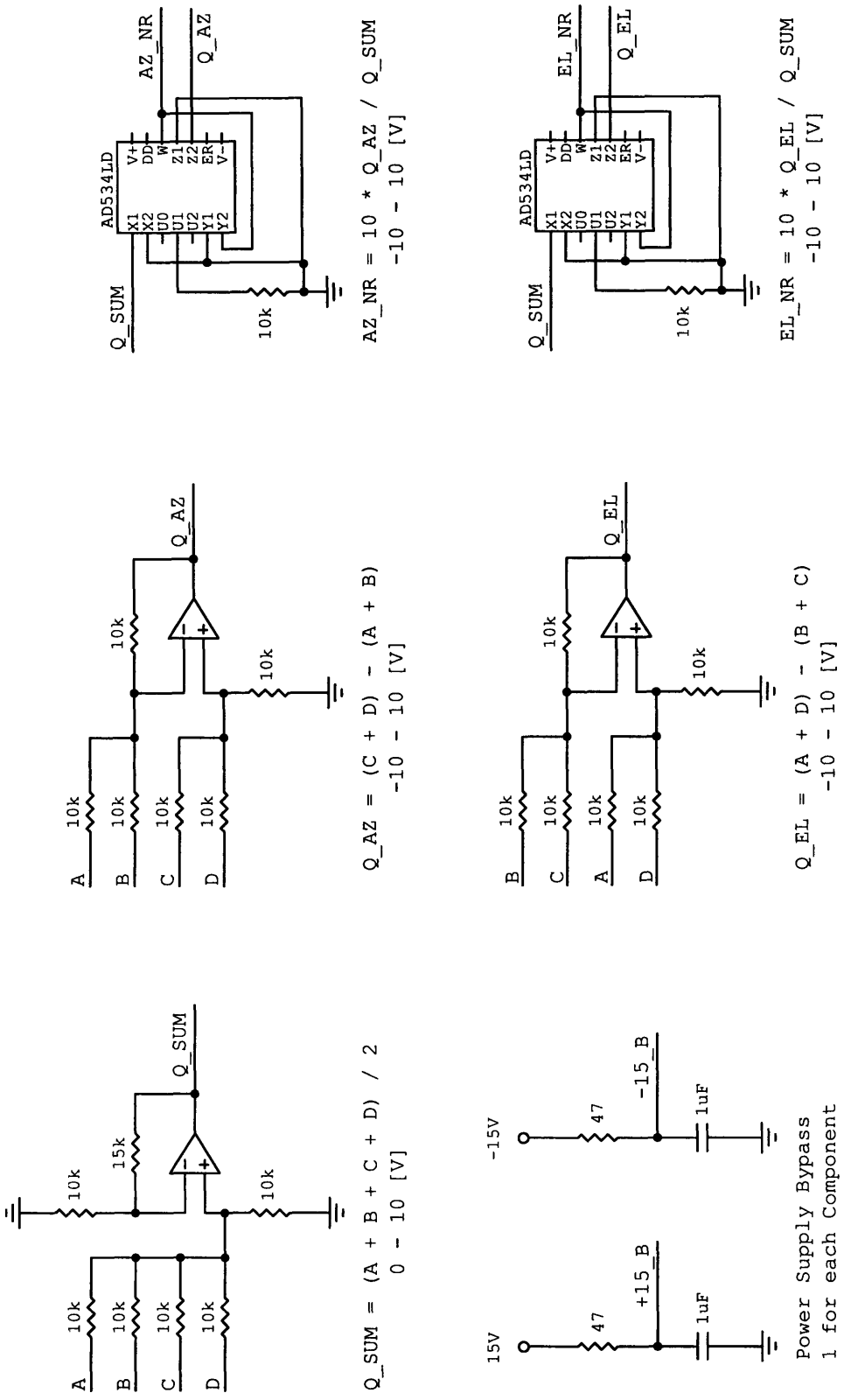

Figure 2-13: Schematic of the conversion circuitry of the C/R Board showing op-amp math functions and output voltage ranges, originally designed at Lincoln Laboratory.

orthogonal axes clockwise by an angle θ, the signal vector is multiplied as follows:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} AZ \\ EL \end{bmatrix} = \begin{bmatrix} AZ_{rot} \\ EL_{rot} \end{bmatrix} \tag{2.3}$$

When rotating 45° clockwise, equation 2.3 simplifies to

$$AZ_{rot} = \frac{\sqrt{2}}{2}(AZ - EL) \tag{2.4}$$

$$EL_{rot} = \frac{\sqrt{2}}{2}(AZ + EL) \tag{2.5}$$

Ignoring the $\sqrt{2}/2$ factor for the moment, rotating the axes 45°is as simple as either adding or differencing the Azimuth and Elevation signals. In fact, to rotate any multiple of 45° involves just a combination of adding and differencing the Azimuth and Elevation signals. This is precisely what the upper half of the C/R accomplishes, using two inverting summers. The schematic for the rotation hardware is shown in Figure 2-14. The input and output voltage ranges for the rotation hardware is ±10 V. The $\sqrt{2}/2$ gain factor is applied by adding a resistive voltage divider network to the end of the output. In this way, the gain of the quad cell voltage can be lowered as necessary. The $\sqrt{2}/2$ gain factor cannot be implemented directly into the op-amp math functions, as it would introduce a gain of less than 1, causing the op-amp circuitry to be unstable (the op-amps used are unity-gain stable).

The switches in Figure 2-14 are actually jumper blocks that select either the positive or negative version of a signal. These jumpers allow the quick changing of the rotation behavior as necessary.

2.1.7 Analog Compensator

Before this work began, the FSM was controlled by an analog compensator originally designed by MIT professor James K. Roberge at Lincoln Laboratory. Figure 2-15 shows a picture of the entire Controller hardware rack. The Controller hardware rack houses the Analog Compensator, the current amplifiers and the analog and digital

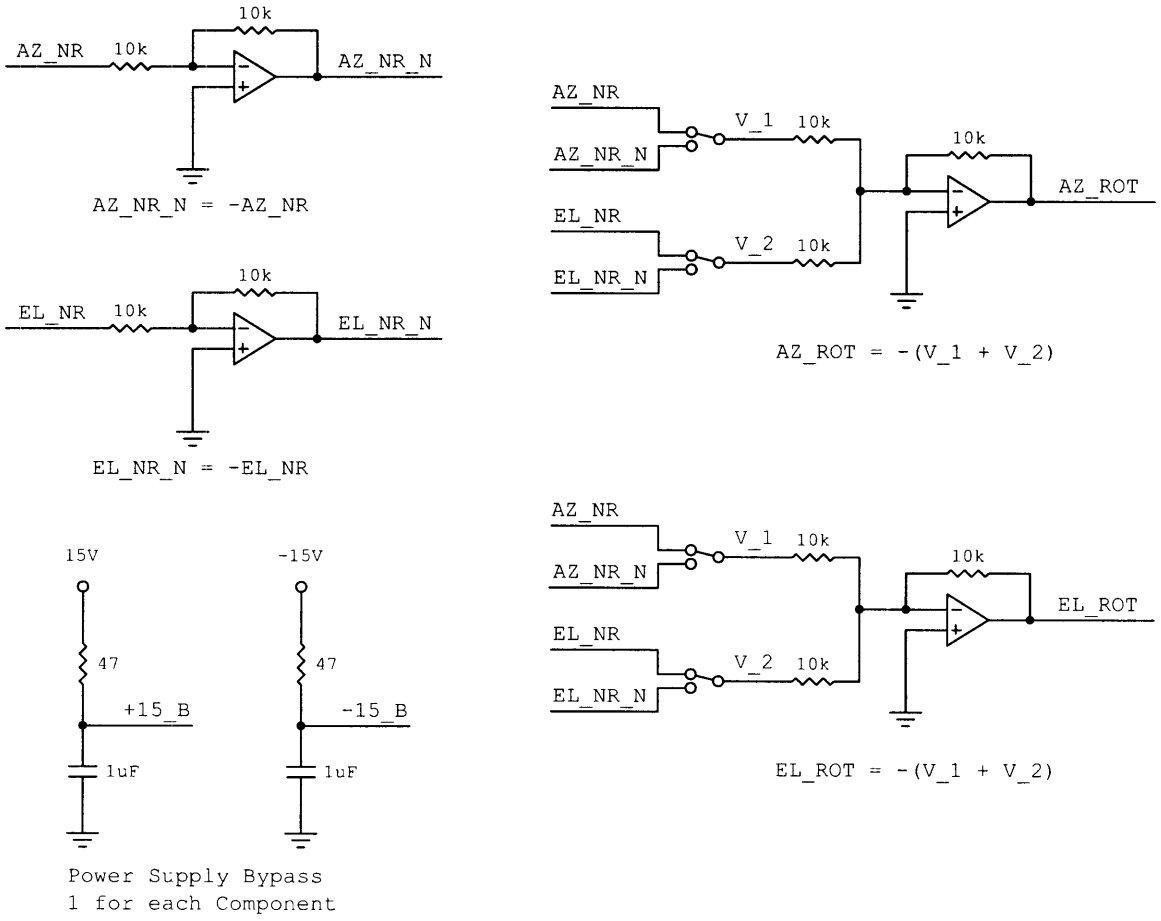

Figure 2-14: Schematic of rotation circuitry of the C/R Board, including op-amp math functions.

power supplies.

Figure 2-15: Picture of the original controller hardware box mounted in rack, housing the analog compensator, current amplifiers and power supplies.

The compensator was designed to control the FSM in either KAMAN or quad cell feedback mode with the flip of a digital switch. Both sensors use the same generic feedback control. The block diagram for the control of the FSM is shown in Figure 2-16.

Included before the compensator are electronics used to rotate the KAMAN sensors 45° for use in the feedback path. The rotation is accomplished much in the same way as the rotation of the quad cell Azimuth and Elevation. The schematic for these electronics is shown in Figure 2-17. This schematic also contains the KAMAN feedback summing junctions. The overall circuit shown in Figure 2-17 takes in both the KAMAN sensor voltages and the commanded voltages, and outputs the difference between the two, just as shown in Figure 2-16. The quad cell Azimuth and Elevation channels are also added together with the appropriate command voltages in the same

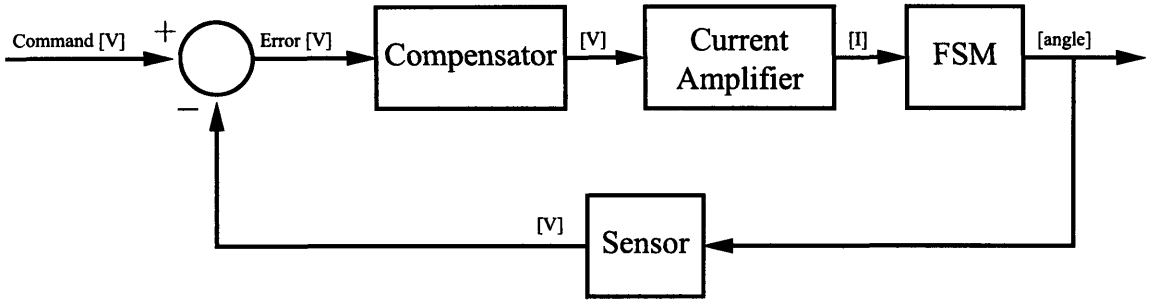

Figure 2-16: Generalized feedback control block diagram for control of the FSM.

manner. However, no rotation circuitry for the quad cell is present.

The schematic for the compensator is shown in Figure 2-18. The compensator takes in output of the KAMAN or quad cell summing junctions (also known as the Error term), and outputs a voltage intended for a current amplifier. This compensator is used for both feedback sensors, but a digital switch allows only one sensor to be active at a time. Thus, the first op-amps in Figure 2-18 only see either the KAMAN voltage or the quad cell voltage, but never both. For a more detailed review of this original compensator with experimental results, see Section 5.1.

2.1.8 Current Amplifier

The final elements of the control system are the current amplifiers. The current amplifiers take a voltage command from the compensator and drive the voice coil actuators of the FSM as appropriate. The bandwidth of these current amplifiers is sufficiently high as to not introduce any noticeable dynamic behavior in the frequency ranges of interest. The schematic for one of the current amplifiers is broken up between Figures 2-19 and 2-20.

2.2 Testing Configurations

Two main testing configurations were used in benchmarking the performance of the FSM. The first configuration is the baseline configuration designed to work with the

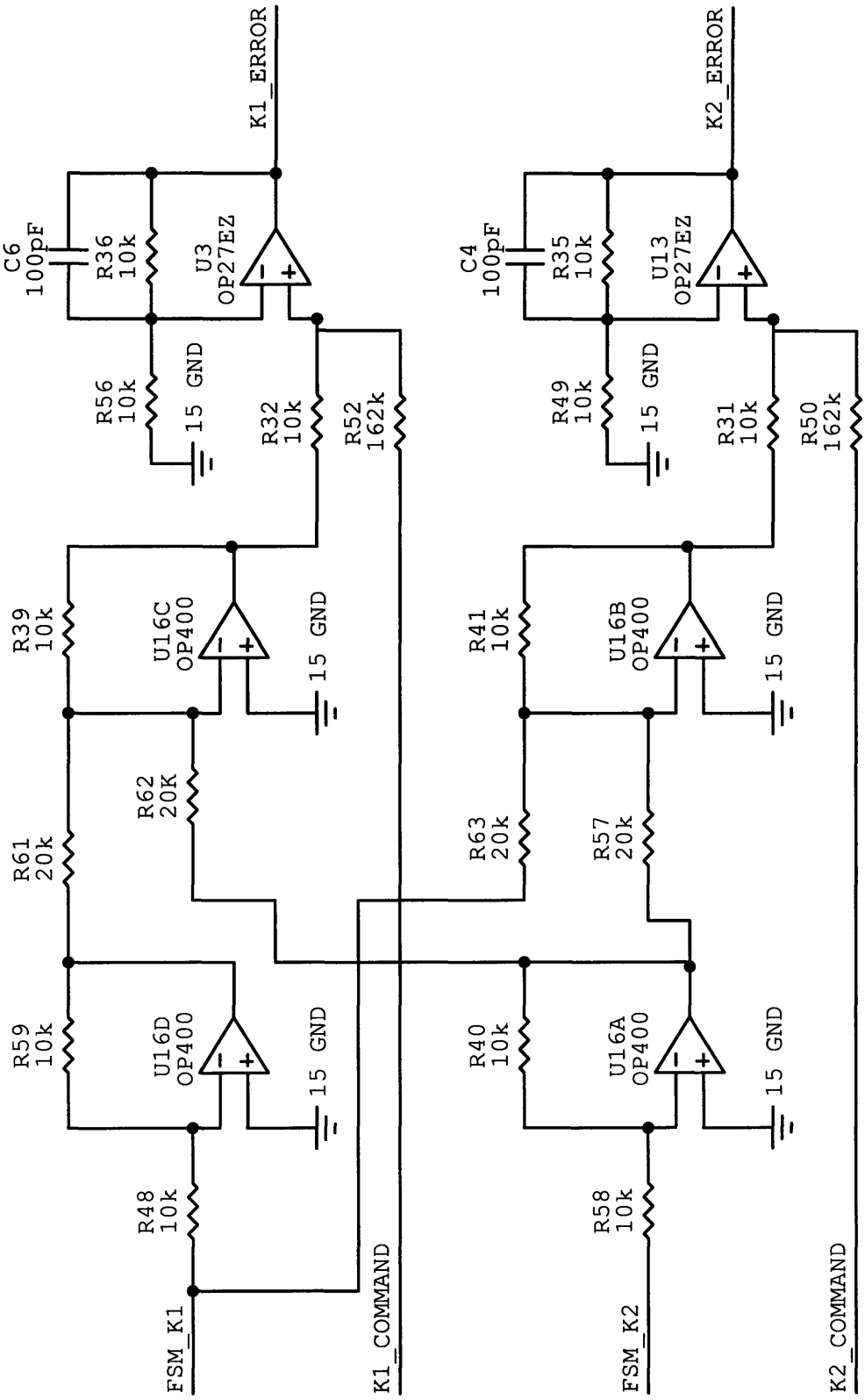

Figure 2-17: Schematic of the KAMAN rotation circuitry and summing amplifier used in feedback control, designed at Lincoln Laboratory.

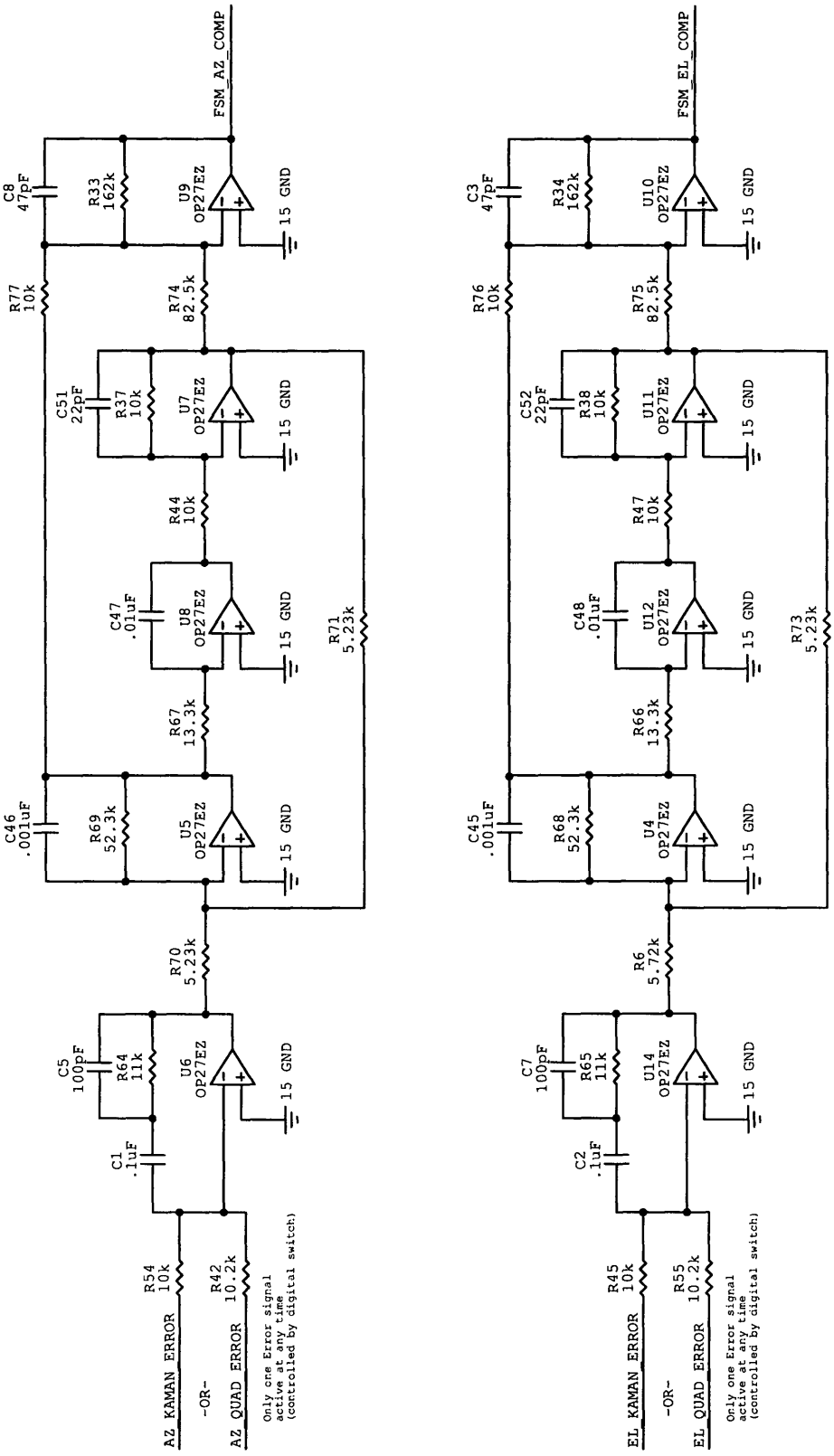

Figure 2-18: Schematic of the original analog compensator used for both KAMAN and quad cell feedback control, designed at Lincoln Laboratory.

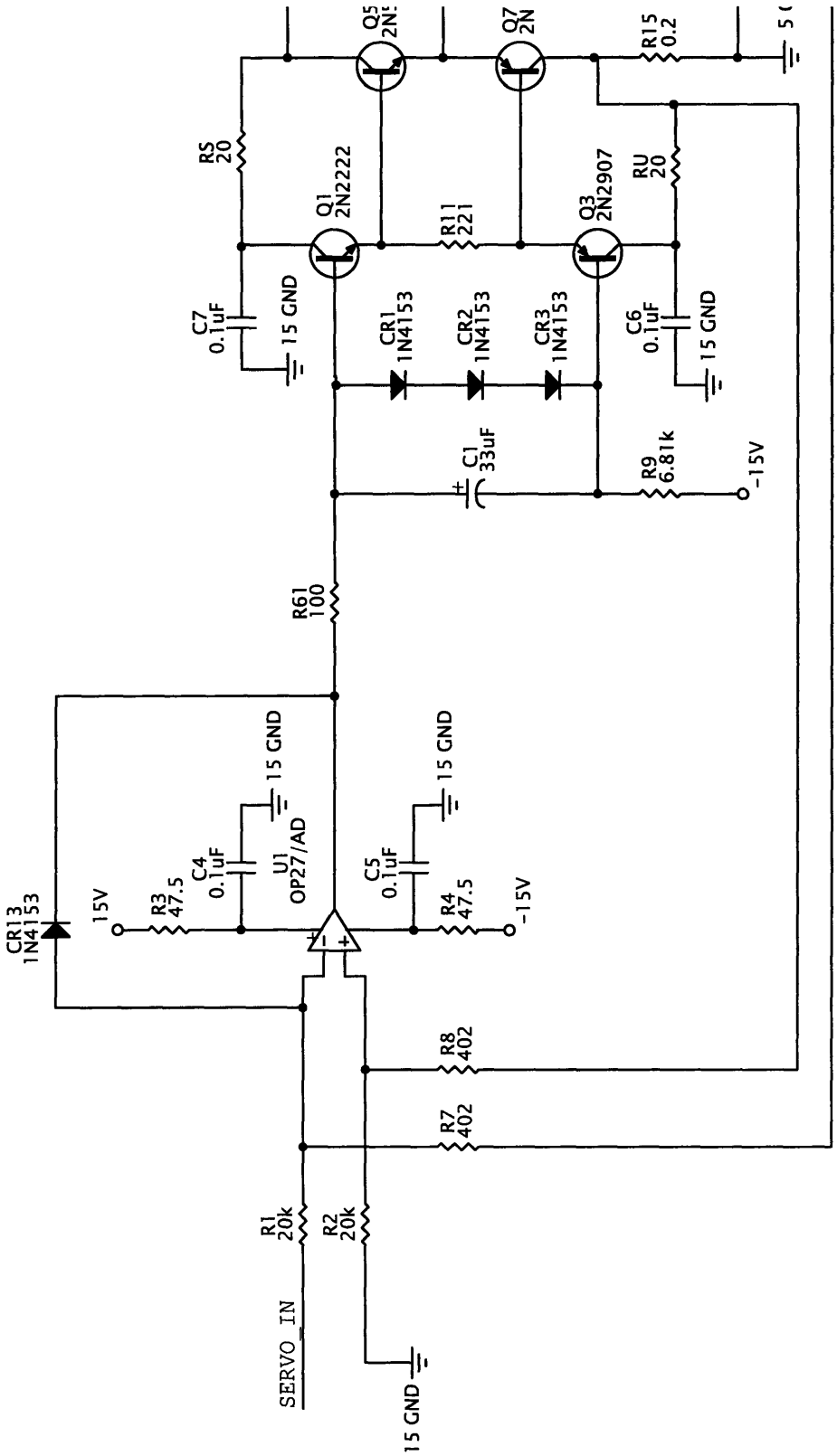

Figure 2-19: Schematic of the high bandwidth current amplifier designed at Lincoln Laboratory (1/2).

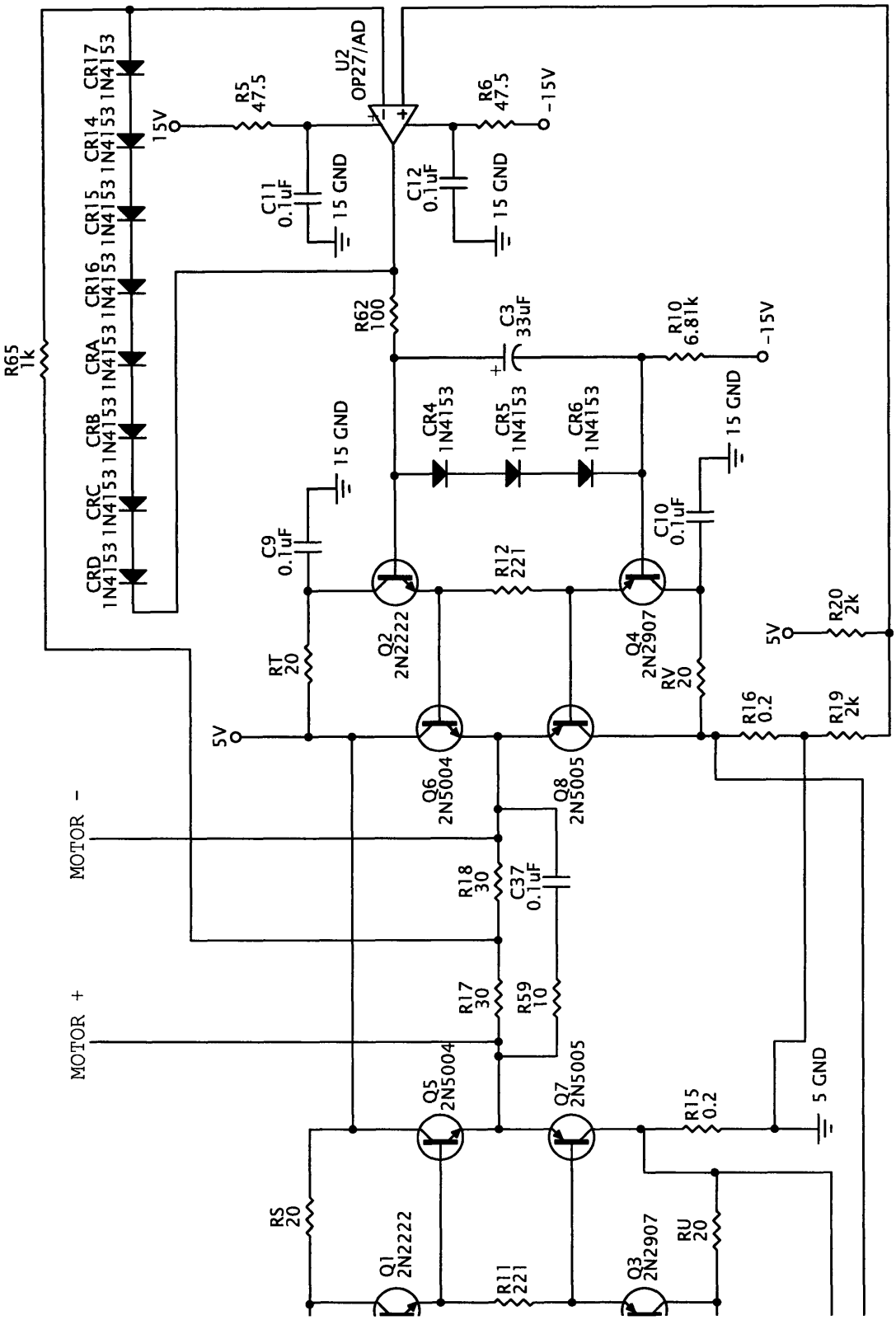

Figure 2-20: Schematic of the high bandwidth current amplifier designed at Lincoln Laboratory (2/2).

analog compensator already in use at Lincoln Laboratory. The second configuration implements an original compensator design using xPC as a digital compensator. These configurations differ in their compensation methods only and the optical setup is the same. Figure 2-21 shows a picture of the hardware configuration.

The quad cell is sensitive enough to register atmospheric disturbances interfering the laser beam path. For this reason, all data was collected with the optical beam path and relevant hardware contained in a Plexiglass shroud. The testing configuration with the shroud in place is shown in Figure 2-22.

2.2.1 Analog Compensator Implementation

The baseline testing configuration is shown in Figure 2-23. This configuration uses the original analog compensator designed by Professor Roberge for feedback control. xPC is used only to control the analog compensator functions. Different models used with xPC provide digital switching between KAMAN and quad cell mode, allow spiral acquisition of the laser spot on the quad cell (see Chapter 4) or take data while an experiment is running. In this configuration, xPC runs the models at a 10 kHz sampling rate.

2.2.2 Digital Compensator Implementation

The testing configuration used to test the original digital compensator is shown in Figure 2-24. The only difference between this testing configuration and the configuration used with the original analog compensator is that the compensation now occurs on the xPC Target PC digitally instead of on separate hardware. In this configuration, the xPC Target must run the models as fast as possible to keep phase loss down to a minimum, typically running between 20 and 50 kHz.

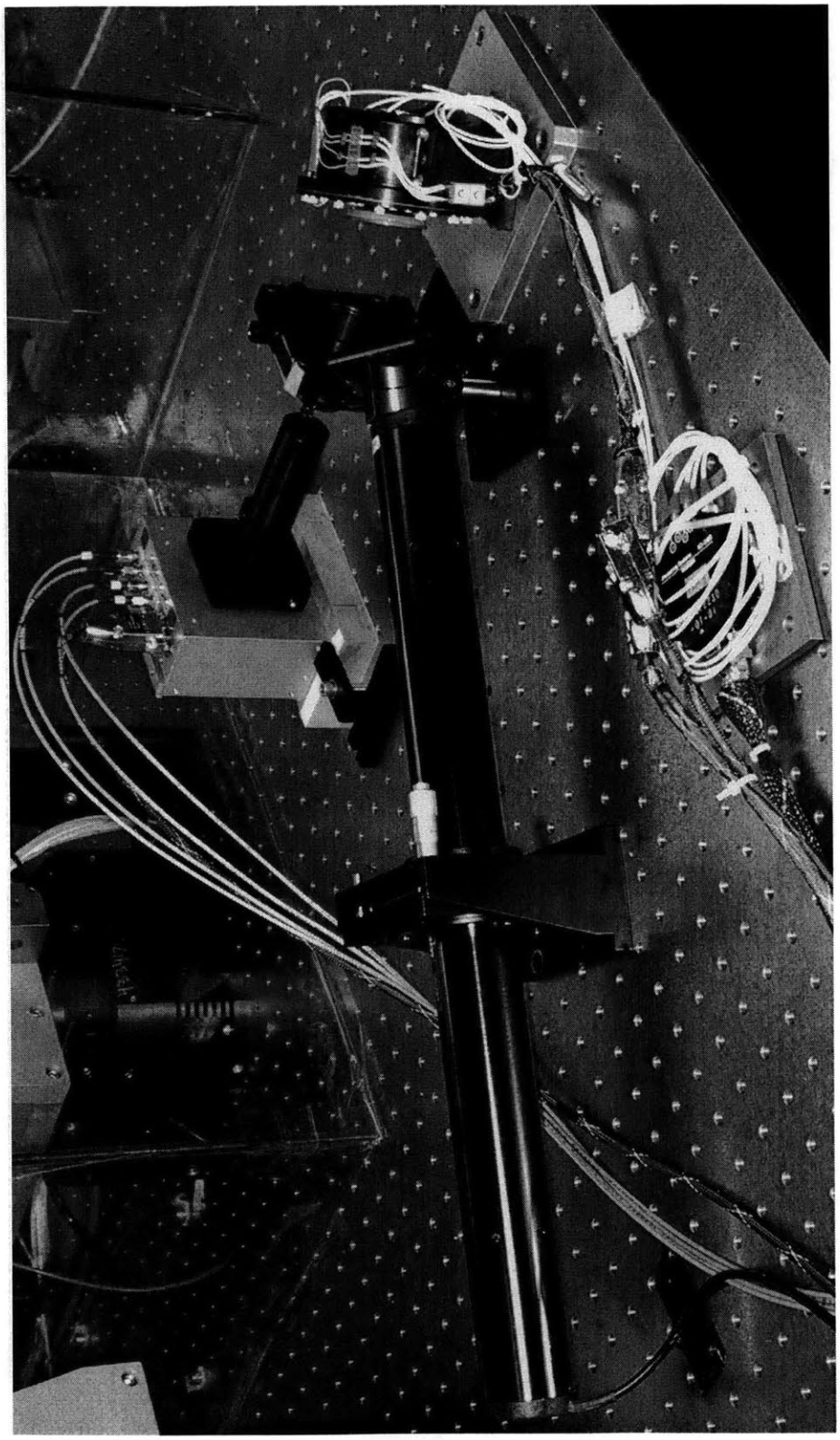

Figure 2-21: Picture of the experimental hardware configuration.

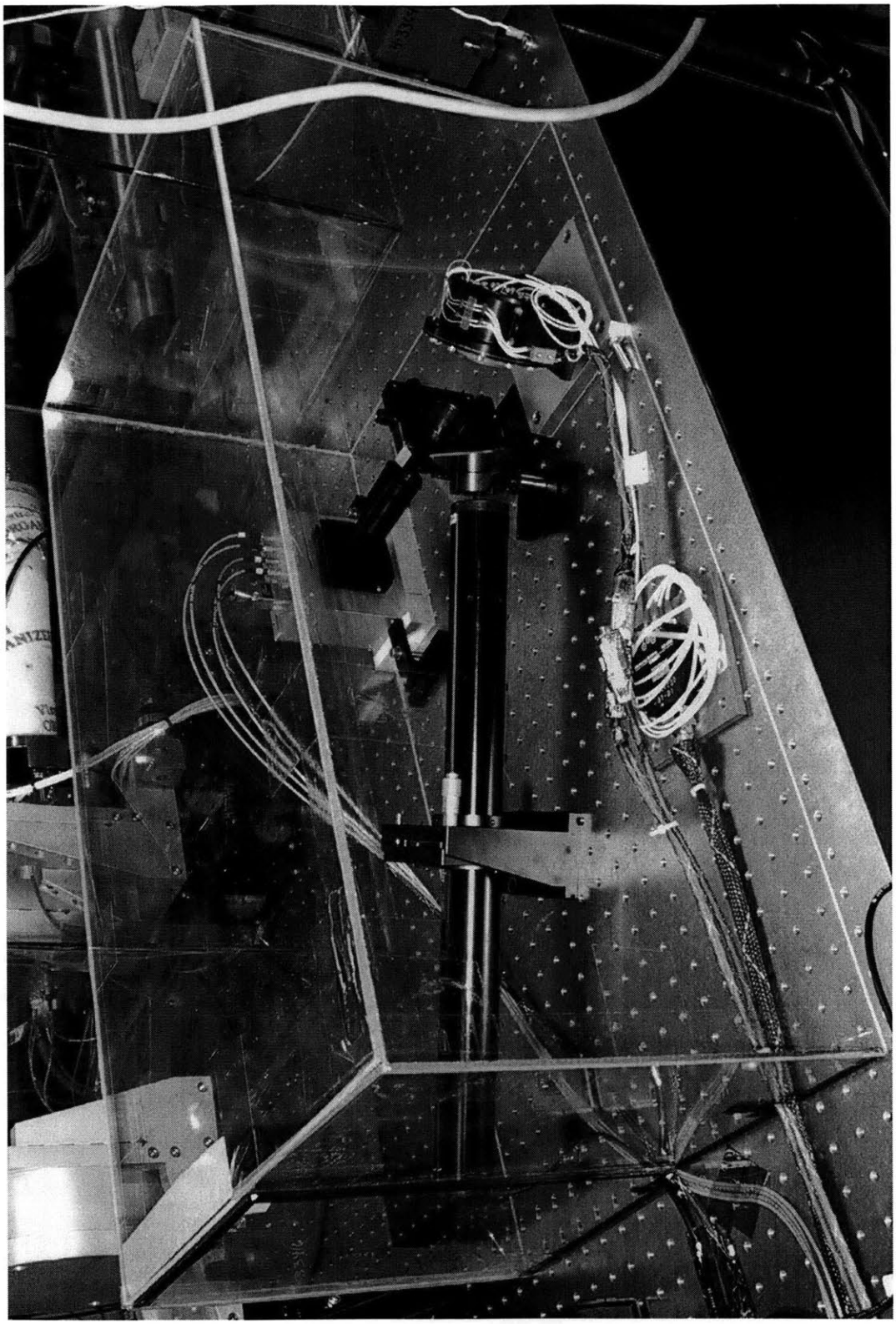

Figure 2-22: Picture of the experimental hardware setup with Plexiglas shroud.

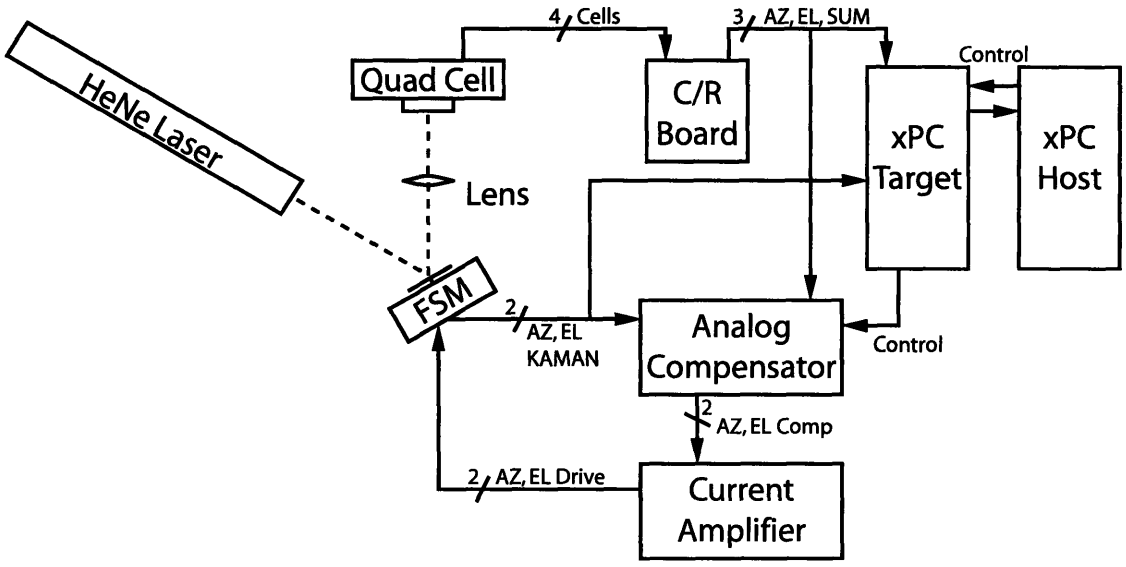

Figure 2-23: Schematic of the hardware configuration using the original analog compensator with xPC.

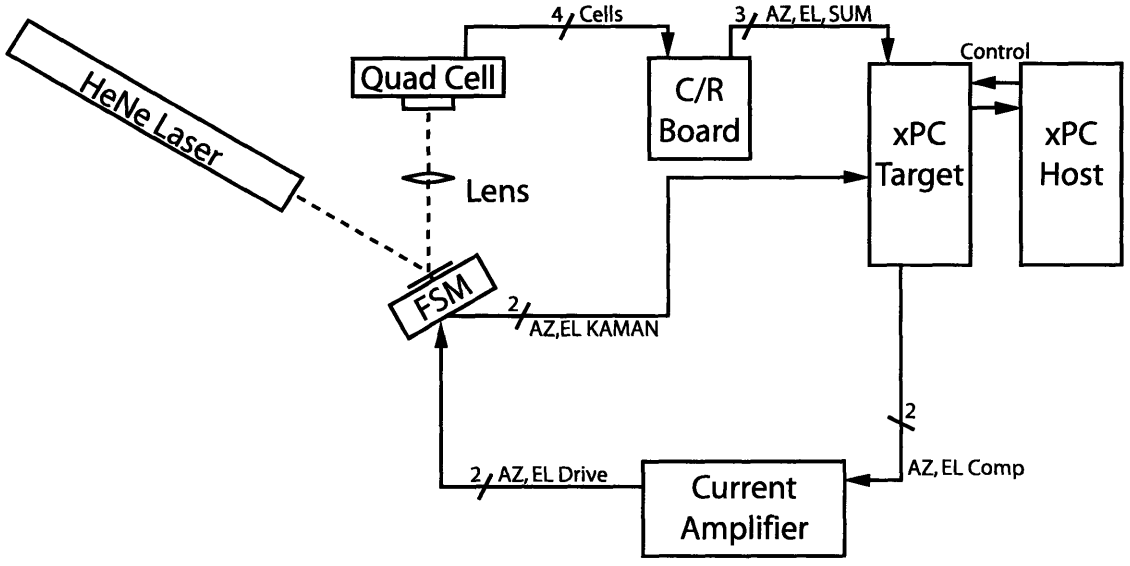

Figure 2-24: Schematic of the hardware configuration using the digital compensator with xPC.

2.3 xPC Implementation

As stated before, xPC is used in the experiments both as a control device for the analog controller and as a compensator as well. This section describes how to implement this type of xPC control or compensation.

2.3.1 Simulink Models

xPC is simply another set of blocks that Simulink can use in its models. The only major difference is that the xPC Target machine runs these Simulink models instead of the computer running MATLAB. As was stated in Section 2.1.4, the xPC Target uses the General Standards PMC-ADADIO cards for communication between the Target computer and the experiment. Several blocks are provided in Simulink to control the PMC-ADADIO cards. Figure 2-25 shows a very basic model illustrating the A/D, D/A and Digital I/O blocks available. The model will run at a specified frequency (not shown in the figure).

Simulation of the model in Figure 2-25 begins with the small block on the upper left of Figure 2-25. This block starts the A/D conversion on the PMC-ADADIO card. Once that is finished, it drives its 'Enable' bit high, telling the next block to start. The next block reads the A/D channels into the Target machine. When this finishes, the next block in the chain (in this case, the D/A write block) evaluates. The D/A block writes its current values into memory, and after this is done, the final block in the chain tells the xPC Target to update its D/A channels with these new values. In the meantime, while or slightly after all of this reading and writing is occurring, the rest of the model evaluates. The first D/A channel is simply the discrete time integral of the sum of the two A/D channels. These two A/D channels also go toward the xPC Scope block. This block produces a display showing the signals routed to it on the monitor of the xPC Target PC. This is useful as a second oscilloscope to monitor signals. The second D/A channel is simply a MATLAB generated pulse generator. The amplitude, pulse width and frequency of this pulse generator can be changed in real time in MATLAB on the xPC Host PC while the model is running. The Digital

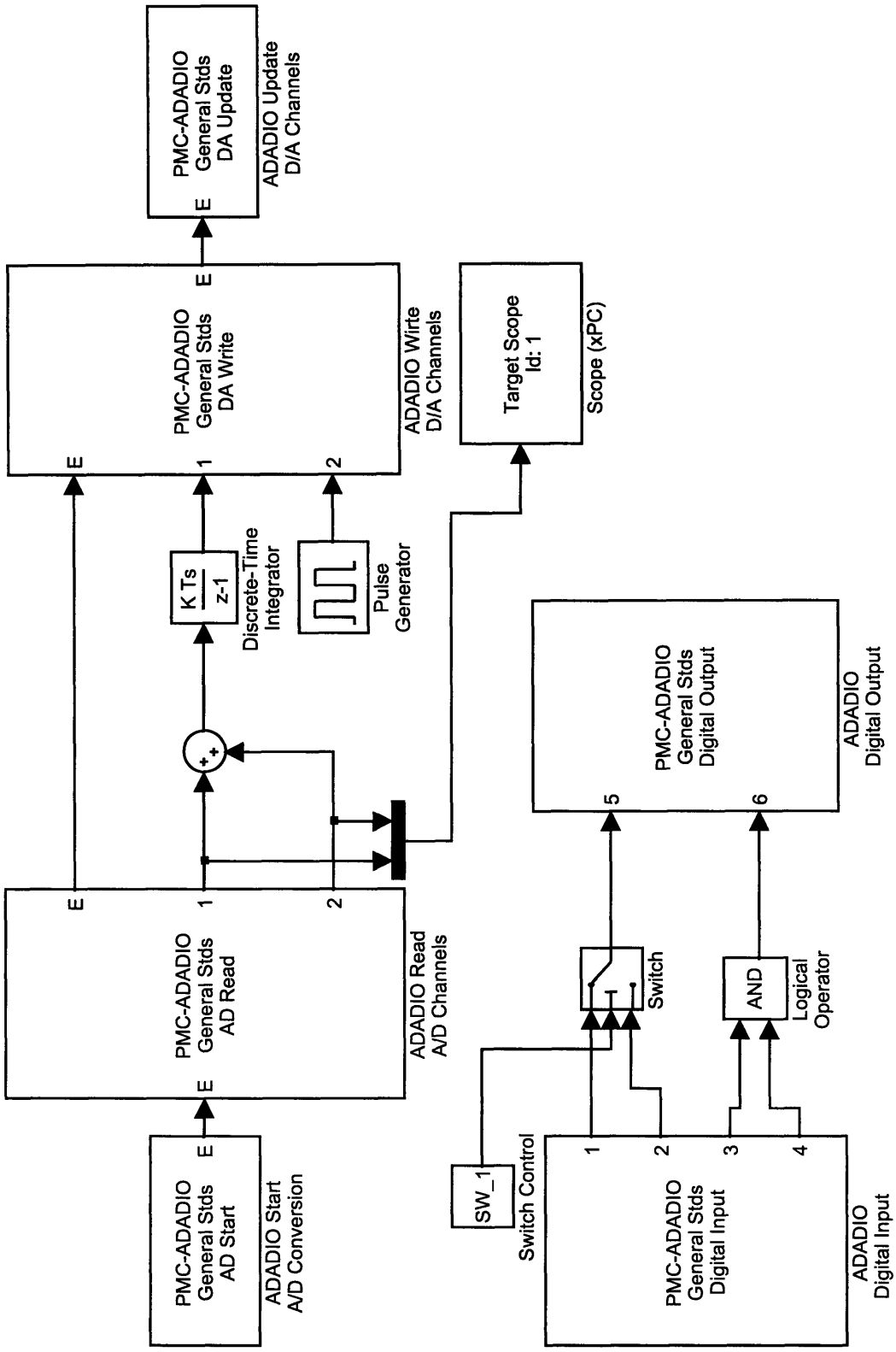

Figure 2-25: Diagram of the Simulink model illustrating the A/D, D/A and Digital I/O capabilities of the General Standards PMC-ADADIO card with common connection methodology.

Input and Output blocks each evaluate once a sampling period. The can be placed in the chain to control exactly when they evaluate if wanted, but this is not necessary. Looking at the digital part of model now, SW_1 controls the output to Channel 5 of the Digital I/0, switching it between input Channel 1 or input Channel 2. SW_1 can be controlled on the xPC Host PC in real time as well. The output to Channel 6 of the Digital I/O is simply the logical AND of I/O input Channels 3 and 4.

2.3.2 Graphical User Interface

The great part about xPC is that parameters can be changed while the Simulink model is running. The not so great thing is that the commands to do this are quite cumbersome. Fortunately, it is possible to write graphical user interfaces (GUIs) in MATLAB to make the control a little easier. Figure 2-26 shows the GUI used to control the digital compensator. The section labeled "FSM Model Control" provides for control over the Simulink model being used. A text box allows for changing the active model. Buttons provide control to build the model on the xPC Target, and to start or stop the model simulation. The section labeled "Scope Control" allows control of the xPC Target scopes in the model (as mentioned in Section 2.3.1). Finally, the section labeled "FSM Compensator Control" allows parameter changes in the model itself. The xPC section of the MATLAB help file is very useful in explaining how these parameters are changed. For reference, the code used to generate this GUI is available in Appendix F.

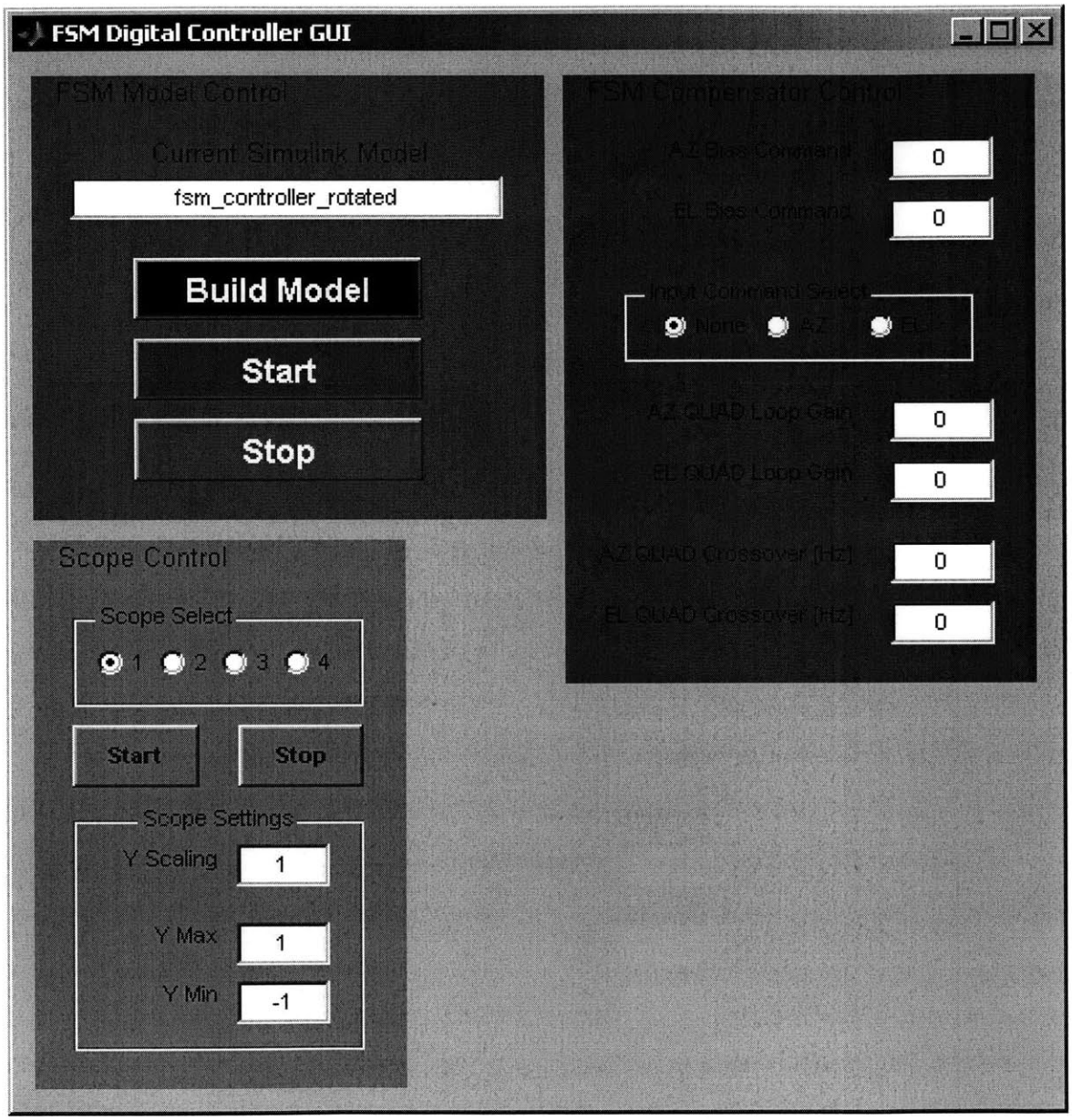

Figure 2-26: Picture of the GUI used to control the digital compensator running on the xPC Target PC.

Chapter 3

The Fast Steering Mirror

This chapter describes the Fast Steering Mirror (FSM). Section 3.1 reviews the applications of FSMs and the origins of the FSM used in these experiments, including its predecessors. Section 3.2 describes the FSM in terms of its components, providing technical data and assembly drawings for reference. Section 3.3 reviews the open loop frequency response of the FSM as taken with the local angle sensors. The frequency responses are examined in detail and parametric models are provided for simulation purposes. Section 3.4 provides the same information, but with the transfer functions taken in respect to the optical angle detector.

3.1 FSM Background Information

This section provides background information on the FSM, including military and commercial applications, as well as the history of FSMs at Lincoln Laboratory.

3.1.1 FSM Applications

A Fast Steering Mirror is a mirror whose angular position can be controlled at a relatively high bandwidth. FSMs are used in many applications and their mirror diameters vary between less than 1 cm to over 30 cm. Fast Steering Mirrors have been used in military and aerospace applications since at least the mid 1980's. These

two industries were the only which could afford the early FSMs, as most were prohibitively costly for commercial applications. Most applications of the FSM in these fields involve target acquisition, alignment, line of sight stabilization and laser beam steering. As the price of FSMs has come down over the last two decades, their usage in industry has increased. Typical commercial applications include beam steering for photolithography during computer chip manufacturing, beam steering for laser eye surgery, astronomy applications, laser-based communication and other optical scanning applications. Most FSMs are similar in design, using flexure mounted mirrors driven with linear voice coil actuators. Piezoelectric actuators can be used when the total angular stroke requirements are low.

3.1.2 FSMs at Lincoln Laboratory

FSMs have been used at MIT for the better part of 15 years. In the 1980's, Lincoln Laboratory began running projects requiring laser beam pointing accuracies in the microradian range, including laser communication. In 1992, G.C. Loney of Group 71 published Project Report IRP-15 titled "*High Bandwidth Steering Mirror Research*". The report details the design of the High Bandwidth Steering Mirror (HBSM) built at Lincoln Laboratory to meet the acquisition, tracking and pointing (ATP) requirements of several experiments. Several prototype mirrors were built using the HBSM design as a template, including the FSM described in this document. These prototype FSMs were used in many different capacities, including in experiments studying optical compensation of thermal blooming, as the mirror used in the Laser Intersatellite Transmission Experiment (LITE), and as a mirror used in the laser communications payload of The National Reconnaissance Office's Geosynchronous Lightweight Technology Experiment (GeoLITE), launched May 18^{th}, 2001.

As MarsComm was commissioned, a decision was made to use this particular FSM as the mirror slated to be in the optical path of the MIRU hardware aboard the MarsComm satellite, both because its mirror diameter was large enough to accommodate the MIRU laser (25 mm in diameter), and the mirror was available "off the shelf". The FSM had been used in other projects before and had performed well

enough to be used in this project. The purpose of this thesis is to document the dynamic characteristics of the FSM, and to design high bandwidth controllers for the device.

3.2 FSM Components

The FSM is a mirror controllable in two axes (Azimuth and Elevation). A picture of the mirror is shown in Figure 3-1, and a exploded component view is shown in Figure 3-2.

3.2.1 Voice Coil Actuators

The FSM mirror is driven by four electromagnetic voice coil actuators. The voice coils are manufactured by BEI Kimco, part # LA05A-05, with a maximum peak force of 0.7 N. The actuators are used in pairs, two to an axis in a push-pull configuration, and thus the maximum peak force on any axis will be 1.4 N. The voice coils are electrically modeled as having a resistance of 4 Ω in parallel with an inductance of 25 μH.

3.2.2 KAMAN Sensors

Local positioning sensors are included in the FSM chassis. The position sensors are KAMAN KD-5100 differential eddy current sensors. These sensors use small coils to drive a magnetic field into the conductive target plate. The strength of the eddy currents produced in a target plate is sensed to give position with resolution on to a nanometer scale. Two sensors are placed a fixed distance apart, and the differential position between the sensors divided by the distances between them gives the angle of deflection (valid only for small angles). The distance between the KAMAN sensors is approximately 80 mm, and assuming a detectible differential length of 80 nanometers, the KAMAN sensors have 1 microradian resolution. The output scale factor of the KAMAN sensors is approximately 1.6 μradians / millivolt. The KAMAN sensors are

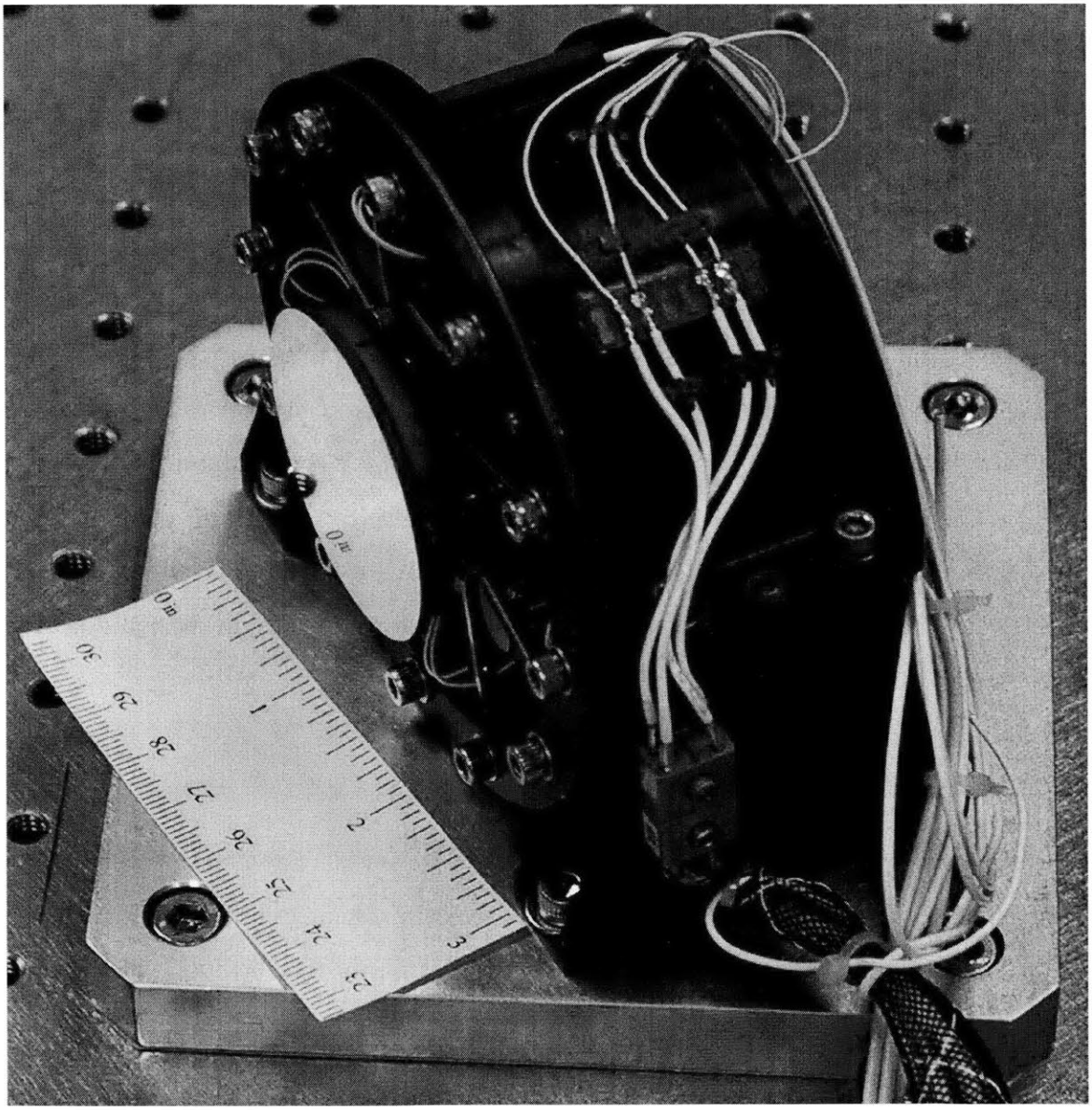

Figure 3-1: Picture of the Fast Steering Mirror mounted on an optical table.

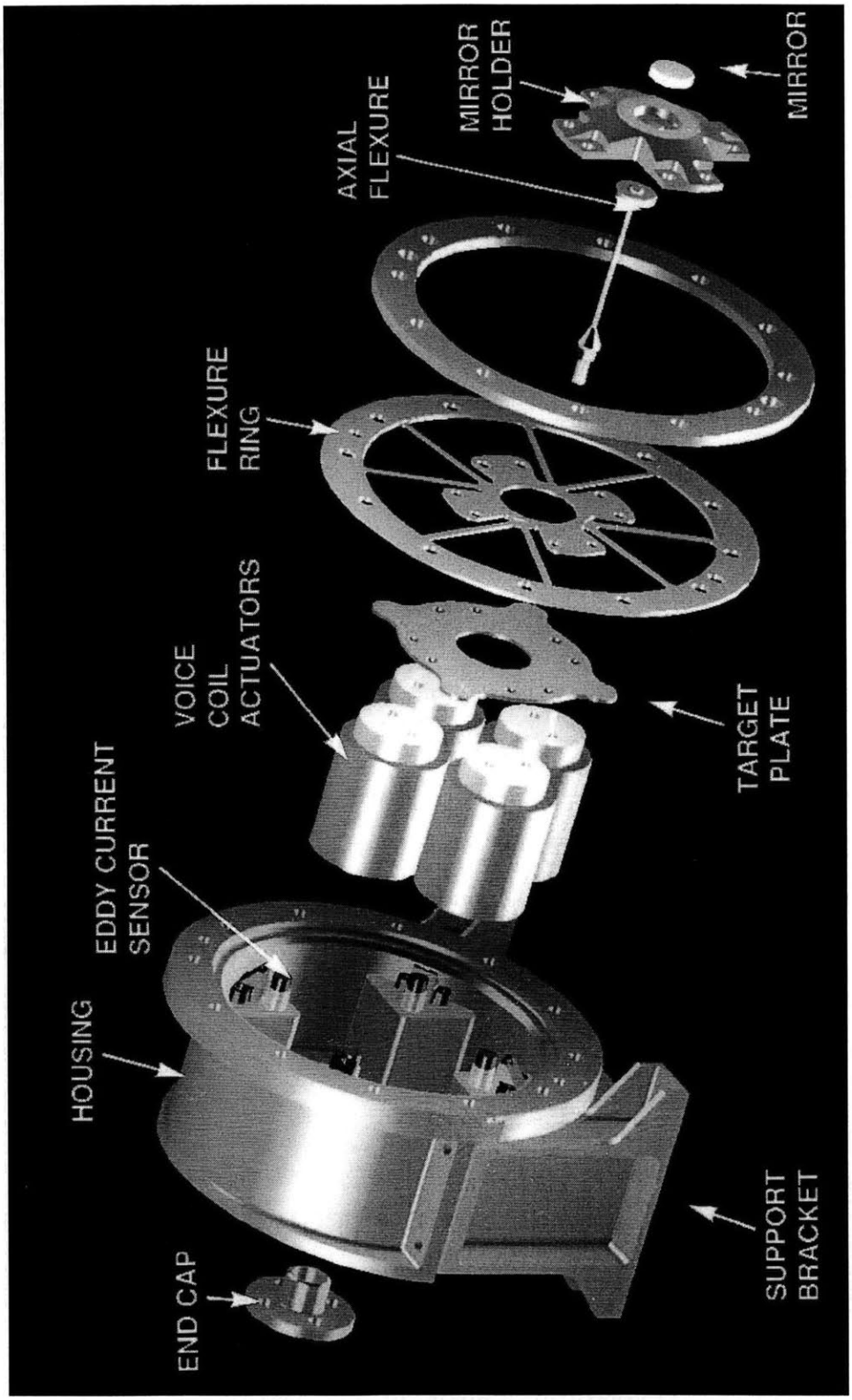

Figure 3-2: Exploded component diagram of the FSM (shown with a smaller mirror), courtesy of James Roberge.

mounted at 45° angles relative to the Azimuth and Elevation axes, and therefore need to be rotated before use in the feedback path. The KAMAN (pronounced KAY-MEN or KUH-MON depending on your preference) sensors are used mainly to control the mirror before going into optical feedback.

3.2.3 The Mirror

The mirror on this FSM is larger than previous mirrors have been on other FSM projects at Lincoln Laboratory. The large size is necessary to accommodate the MIRU laser diameter of 25 mm. The mirror is approximately 45 mm in diameter. The mirror is made of beryllium, giving it a high stiffness to weight ratio. The mirror is coated with aluminum protected by SiO_2.

3.2.4 Flexure Assemblies

The flexure assembly provides the flexible mounting for the mirror. Their are two separate flexures in the body of the FSM. The first flexure is an axial flexure fixing the mirror to the housing assembly. This flexure provides high axial stiffness perpendicular to the surface of the mirror while providing low bending stiffness, allowing the mirror to rotate in the Azimuth and Elevation directions. This flexures is made out of T6 aluminum. The second flexure is the flexure ring. This ring, made from stainless steel (and possibly layered with other materials to provide additional damping), provides lateral support for the mirror and allows it to rotate in the Azimuth and Elevation axes as well.

3.3 KAMAN Open Loop Frequency Responses

The FSM can be analyzed without any feedback control in place by using two different sensors. The KAMAN sensors are built in and provide angle measurements referenced to the mirror body. Alternatively, a quad cell gives angle measurements with respect to the mirror surface, and is used for optical tracking (see Section 2.1.5). This section

deals with the open loop frequency responses of the FSM using the KAMAN sensors.

In a simple model, the FSM can be thought of as a mass-spring-damper system. The main mass is the mirror at the end of the flexure assembly. The flexure acts both as a spring and a very weak damper. The resulting system is a standard underdamped 2^{nd} order system. The frequency response of Figure 3-3 shows that between 10 Hz and 1.5 kHz, the FSM Azimuth axis behaves almost exactly like an underdamped 2^{nd} order system, with a resonant peak at about 112 Hz. This data was gathered with the Agilent 35670A Dynamic Signal Analyzer, applying a voltage of up to 10 Vpk to the current driver and differentially amplifying the KAMAN output voltage as necessary. Data was taken with a resolution of 1600 points / kHz between 10 Hz and 12 kHz, and 1600 points between 12 and 20 kHz.

Interesting (and slightly annoying) things start occurring after about 1.5 kHz. Several pole-zero "doublets" occur between 1.5 and 4 kHz. The doublets are pairs of complex zeros and poles. They occur when part of the FSM "decouples" from what the KAMAN sensors register. A simple mass-spring-damper system can illustrate what is occurring at these doublets.

Figure 3-4 shows a simple model of the FSM near where one of these doublets occurs. Both masses are part of the FSM; however, m_1 represents the part of the FSM that is directly monitored by the KAMAN sensors, and m_2 represents the part of the FSM that is decoupling from the system. For example, the movements of the mirror might differ from the movements of the KAMAN target plate at higher frequencies and the two are "decoupled" from each other (and the KAMAN sensors are no longer measuring true mirror angle).

The force balance equations for the model in Figure 3-4 are

$$m_1\ddot{x}_1 + (b_1 + b_2)\dot{x}_1 + (k_1 + k_2)x_1 = b_2\dot{x}_2 + k_2x_2 + F \tag{3.1}$$

$$m_2\ddot{x}_2 + b_2\dot{x}_2 + k_2x_2 = b_2\dot{x}_1 + k_2x_1 \tag{3.2}$$

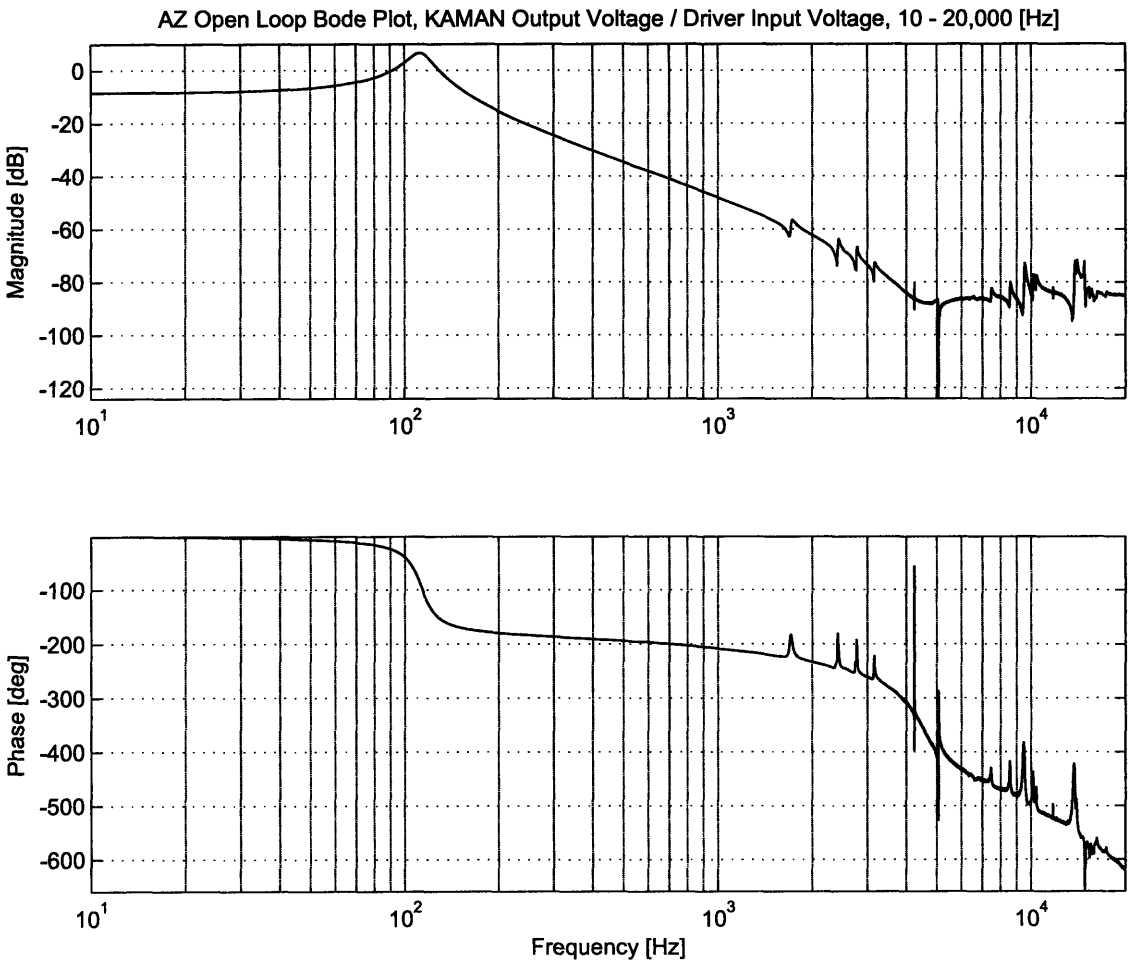

Figure 3-3: FSM Azimuth open loop frequency response, KAMAN Voltage Out / Driver Voltage In, 10 - 20,000 Hz. Data taken with a resolution of 1600 points / kHz between 10 Hz and 12 kHz, and 1600 points between 12 and 20 kHz.

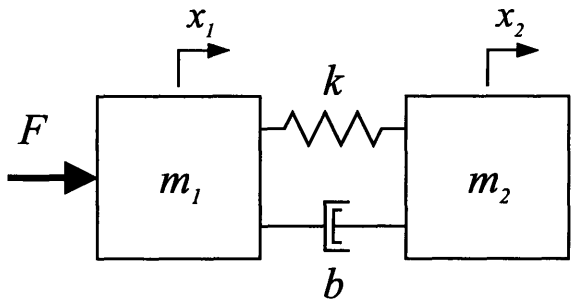

Figure 3-4: Simple model of one axis of FSM. m_1 represents the part of the FSM that is directly monitored by the KAMAN sensors, and m_2 represents the decoupling mass.

Taking the Laplace Transform of each equation yields

$$X_1(s)\left(m_1 s^2 + bs + k\right) = X_2(s)\left(bs + k\right) + F(s) \tag{3.3}$$

$$X_2(s)\left(m_2 s^2 + bs + k\right) = X_1(s)\left(bs + k\right) \tag{3.4}$$

Finally, rearrange to obtain the two transfer functions $X_1(s)/F(s)$ and $X_2(s)/F(s)$.

$$\frac{X_1(s)}{F(s)} = \frac{m_2 s^2 + bs + k}{\left(m_1 m_2 s^2 + b\left(m_1 + m_2\right)s + k\left(m_1 + m_2\right)\right)s^2} \tag{3.5}$$

$$\frac{X_2(s)}{F(s)} = \frac{bs + k}{\left(m_1 m_2 s^2 + b\left(m_1 + m_2\right)s + k\left(m_1 + m_2\right)\right)s^2} \tag{3.6}$$

The poles and zeros of Equations 3.5 and 3.6 are found using the quadratic formula. The zeros for $X_1(s)/F(s)$ are

$$z_{1,2} = \frac{-b \pm \sqrt{b^2 - 4m_2 k}}{2m_2} \tag{3.7}$$

The zero of $X_2(s)/F(s)$ is

$$z = -\frac{k}{b} \tag{3.8}$$

Both transfer functions have the same poles. These poles are

$$p_{1,2} = \frac{-b(m_1+m_2) \pm \sqrt{(b(m_1+m_2))^2 - 4m_1m_2k(m_1+m_2)}}{2m_1m_2} \tag{3.9}$$

$$p_{3,4} = 0 \tag{3.10}$$

The transfer function in Equation 3.5 provides a basic model for each "doublet" of the FSM. The state-space representation of this model is described in Appendix A. The Bode diagram of the first doublet of the Azimuth axis is shown in Figure 3-5.

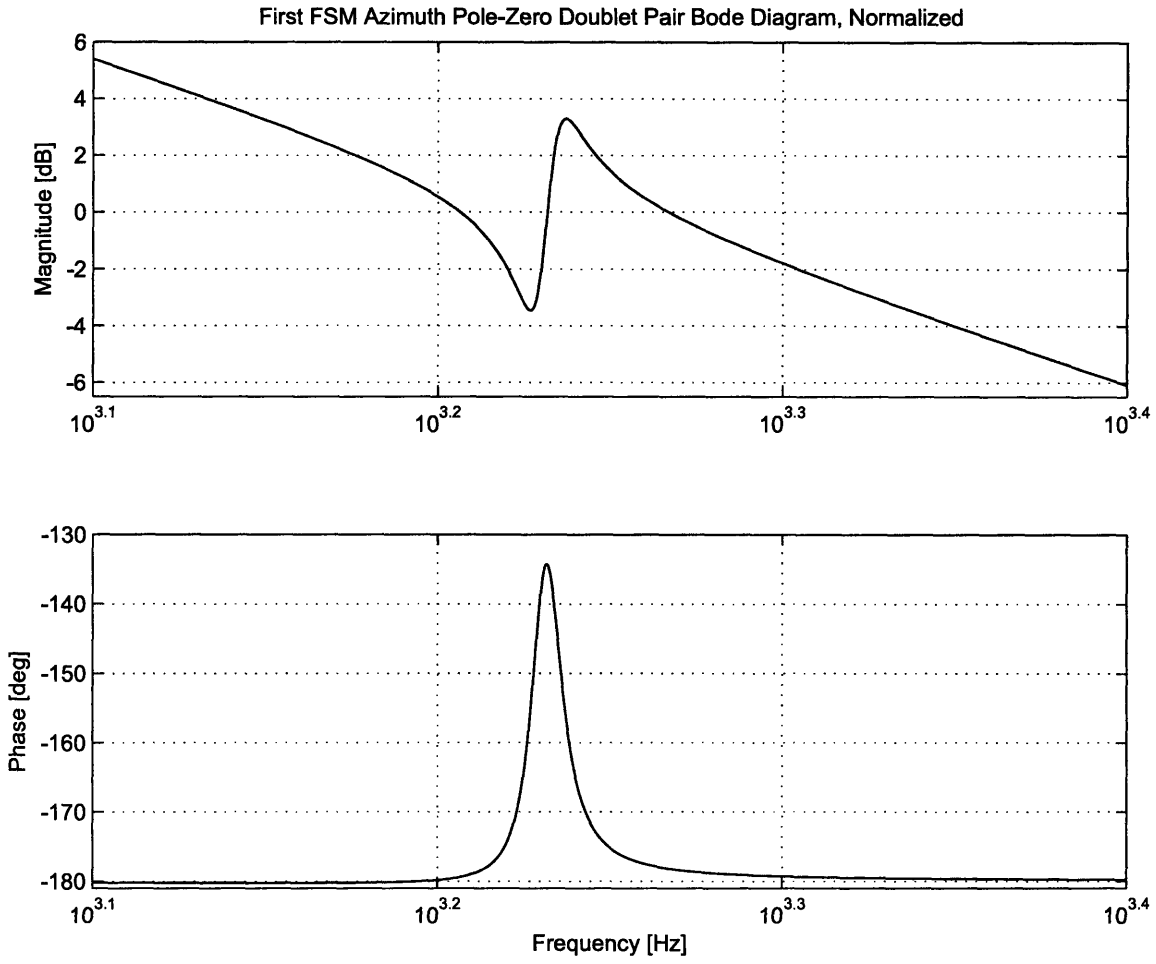

Figure 3-5: Normalized Bode diagram of first "doublet" of the FSM Azimuth axis. Bode diagram is Position / Input Force.

Figure 3-5 is a normalized Bode diagram of the first doublet that occurs on the

FSM Azimuth axis near 1.6 kHz. Equation 3.5 provides a basic description of this doublet; however, the interactions are not confined to two masses. For this reason, this doublet is modeled as

$$\frac{X_1(s)}{F(s)} = \frac{m_z s^2 + b_z s + k_z}{(m_p s^2 + b_p s + k_p)\, s^2} \tag{3.11}$$

Modeling each doublet in this manner allows control of each individual peak of the doublet. The parameter values used in Figure 3-5 are

$$m_z = .0001$$

$$b_z = .015$$

$$k_z = 23710$$

$$m_p = .0001$$

$$b_p = .0148$$

$$k_p = 23880$$

Although the model does not exactly follow the form of Equation 3.5, Figure 3-5 still illustrates the dynamics of m_1 in Figure 3-4. Before the doublet occurs, both m_1 and m_2 are moving together with little dynamic interaction, and the transfer function rolls off with a -2 decade per decade slope. As the frequency of the input force moves toward the doublet, the magnitude hits a complex zero pair. In physical terms, m_2 is moving almost completely out of phase with m_1, and thus opposing the force on m_1. Moving past the complex zeros, the magnitude rises sharply to a set of complex poles. This peak represents the natural frequency of the system. m_2 is no longer directly opposing the motion of m_1, but is in fact helping. Moving past this point, the frequency response continues down with a -2 decade per decade slope as if nothing happened. However, the magnitude of the continuing line has actually been pushed up slightly. This is because m_2 has decoupled from the system, and the effective mass of the system has been slightly reduced. Control issues related to these doublets are discussed in Chapter 5.

Looking past the first few doublets, the transfer function of Figure 3-3 starts to go crazy. After about 5 kHz, the -2 decade per decade roll off disappears and the mirror stops behaving like a 2^{nd} order system. However, the data is unreliable past 5 kHz. As was mentioned before, a DSA was used and the KAMAN sensors were differentially amplified. The DSA used has a noise floor of about -80 dB, requiring substantial amplification of the KAMAN output voltage. This amplification introduces a lot of high frequency noise. The current driver cannot supply enough current at higher frequencies to produce enough angular motion for the KAMAN sensors to register. In order to get better high frequency information with the KAMAN sensors, the current driver would have to be upgraded to provide more current or the voice coil actuators would have to be upgraded to provide more force.

Using the data in Figure 3-3 as a baseline, a parametric model was created in MATLAB. Development of the model was as follows. The first step was to provide the equation for the basic second order system describing each axis. This equation is of the form $K/\left(ms^2+bs+k\right)$, with $\sqrt{k/m}$ representing the natural frequency and $b/2\sqrt{km}$ representing the damping ratio of the system. K is the term used to adjust the DC gain of the system. This equation represents the main 2^{nd} order system describing the axis. Each doublet of the system is added by multiplying the main equation by an equation of the same form as Equation 3.11, except the two poles at the origin are removed. These two poles provided the -2 decade per decade roll off of each doublet, but this roll off is accomplished with the main 2^{nd} order equation. In practice, both m_z and m_p are kept constant, and the frequency of the doublet peaks is controlled with k_z and k_p. The difference between k_z and k_p controls if the doublet goes up or down first, and the total width of the doublet. The heights of the doublet peaks are controlled by b_z and b_p. The difference between b_z and b_p controls the relative "strength" of each half of the doublet pair, giving one peak more amplitude than the other. While each doublet was added, additional poles and zeros were added to maintain the overall slope and phase of the transfer function.

The state-space representation of the parametric model for the FSM Azimuth axis is provided in Appendix B. This model relates the KAMAN output voltage to the

current driver input voltage. This model is a reduced order model of the FSM Azimuth axis and is valid to about 5 kHz, which is adequate for preliminary simulations. The model consists of the main 2^{nd} order system, 3 doublets, and 3 higher frequency poles with 1 higher frequency zero for minor slope corrections. The comparison between the model and the experimental data is shown in Figure 3-6. The model fits fairly well, with minor digression of phase after 300 Hz (about 10°).

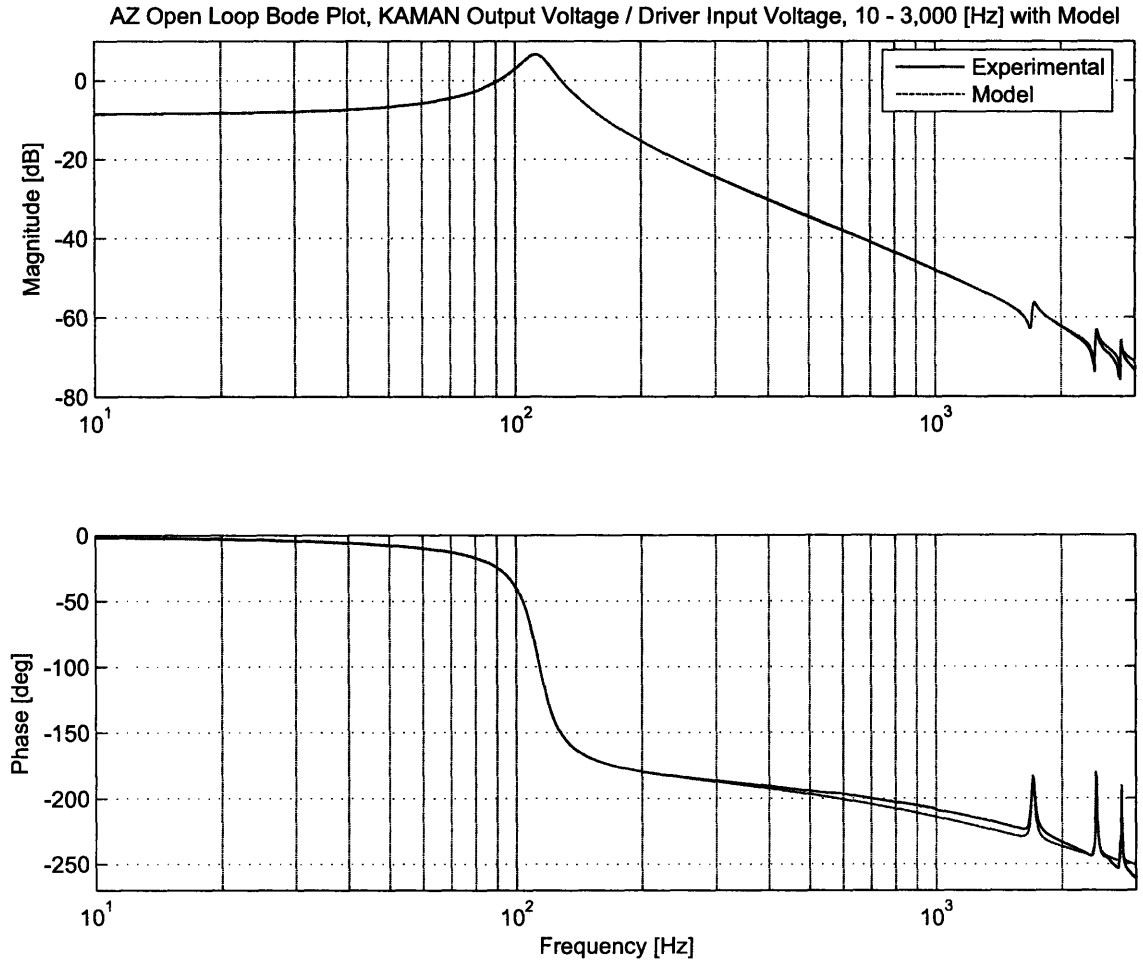

Figure 3-6: FSM Azimuth open loop KAMAN frequency response with parametric model, 10 - 3,000 Hz.

The Elevation axis of the FSM is almost identical to the Azimuth axis. Data was gathered the exact same way, and the frequency response is shown in Figure 3-7. A parametric model was also created to represent the Elevation axis. The state-space

representation of this model is shown in Appendix C. This model is slightly simpler than the Azimuth model, and consists of the main 2^{nd} order equation, 3 doublets and 1 higher frequency pole for minor slope correction. The comparison between this model and the experimental data is shown in Figure 3-8.

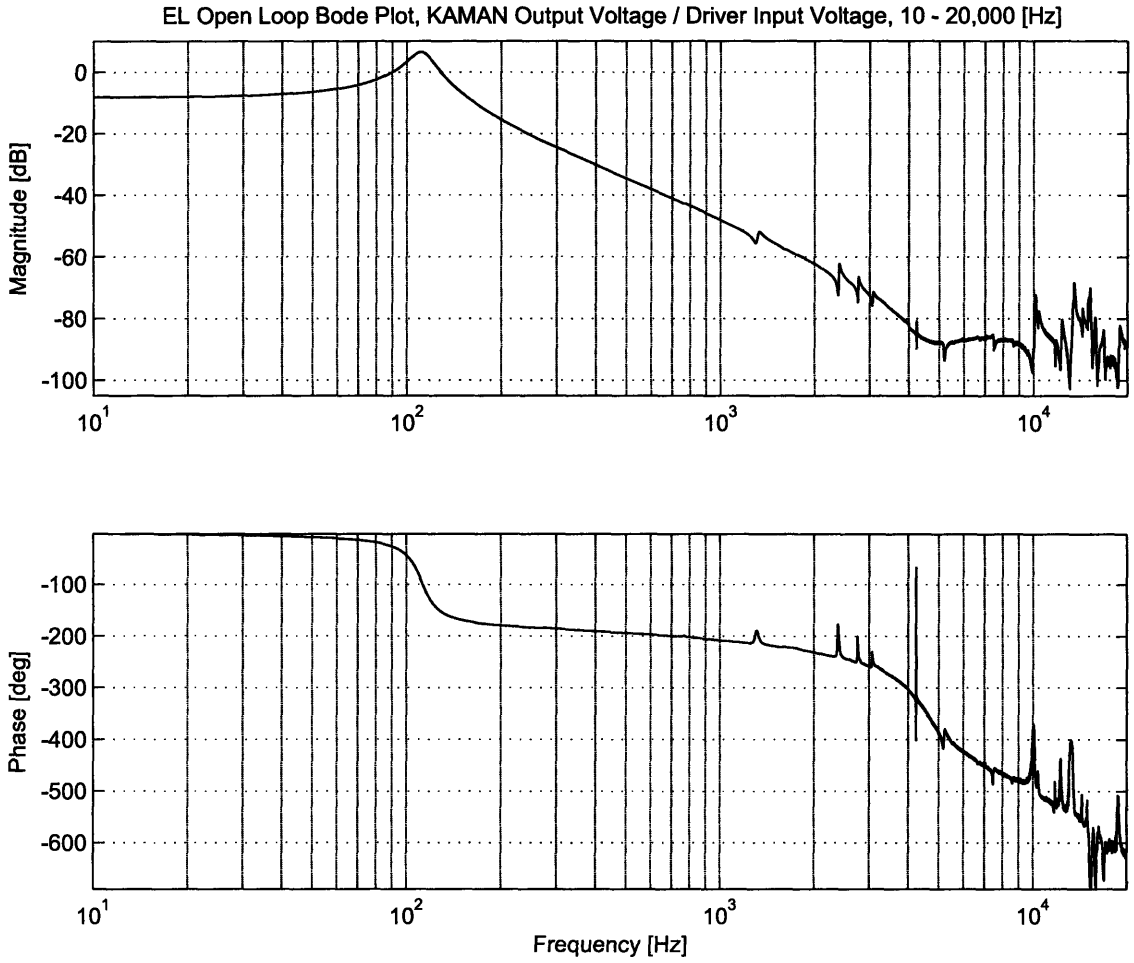

Figure 3-7: FSM Elevation open loop frequency response, KAMAN Voltage Out / Driver Voltage In, 10 - 20,000 Hz. Data taken with a resolution of 1600 points / kHz between 10 Hz and 12 kHz, and 1600 points between 12 and 20 kHz.

Finally, a comparison between the two axes is shown in Figure 3-9. The two axes seem to meet pretty well, except for the first doublet of each axis. On the Azimuth axis, the doublet occurs near 1.6 kHz. However, the first doublet on the Elevation axis occurs at about 1.3 kHz. The reasons for this difference are unknown at the present time and warrant further study. The rest of the model seems to match fairly

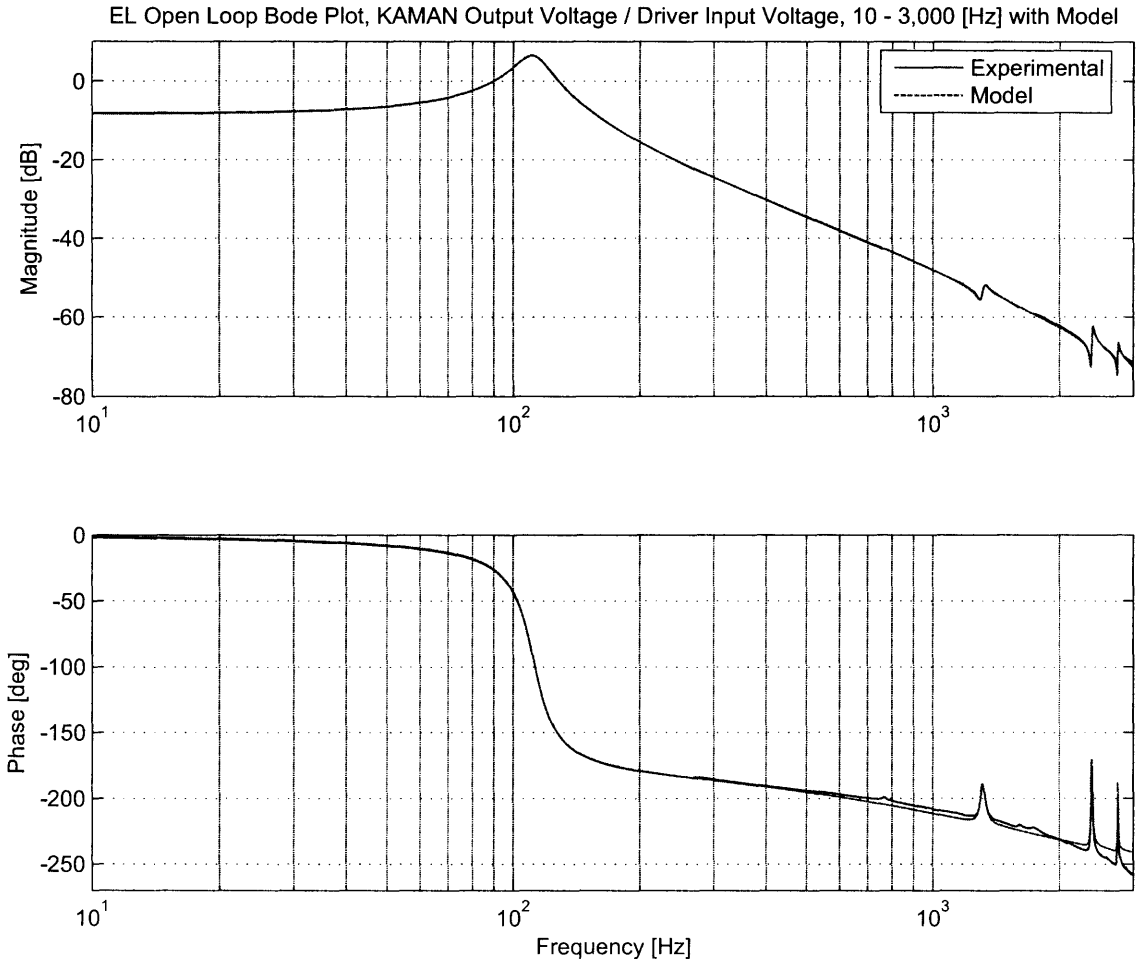

Figure 3-8: FSM Elevation open loop KAMAN frequency response with parametric model, 10 - 3,000 Hz.

well out to 5 kHz. Beyond that, the data is questionable at best and a comparison would yield no interesting or reliable information.

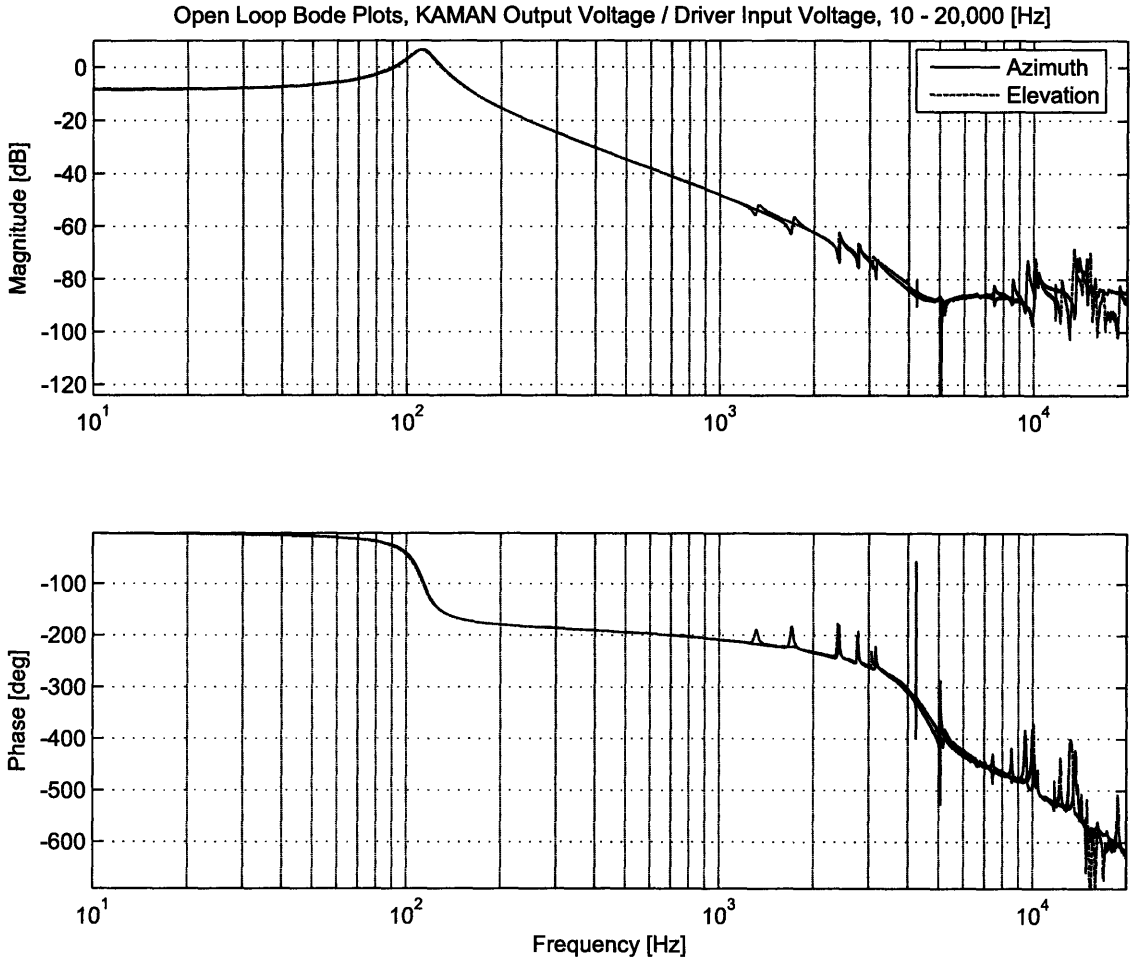

Figure 3-9: FSM Azimuth and Elevation open loop frequency response overlay, KAMAN Voltage Out / Driver Voltage In, 10 - 20,000 Hz.

3.4 Quad Cell Open Loop Frequency Responses

Characterizing the open loop frequency responses of the FSM with only its KAMAN sensors is not enough to fully describe the system nor attempt any sort of optical feedback control. In order to fully understand the FSM, open loop frequency responses using the quad cell were obtained. One might assume that the open loop frequency

responses should look almost identical independent of what position sensor was being used. However, analyzing the differences between the KAMAN sensor frequency responses and the quad cell frequency responses yields a great deal of information about the FSM hardware and its limitations.

Open loop frequency responses were taken in much the same was as with the KAMAN frequency responses. However, unlike the KAMAN sensors, the FSM must be pointed so that the beam falls on the active region of the quad cell before taking data. The initial pointing of the FSM onto the quad cell was accomplished in one of three ways:

1. Turn on all the hardware, and position the laser beam onto the active region of the quad cell by adjusting the Laser Source or Focusing Lens. Often, after taking a small amount of data, the laser spot would move off of the active region of the quad cell, requiring realignment of the optical hardware. This is undesirable because aiming the laser spot on the active region of the quad cell in this manner could take over 5 minutes.

2. Use a summing voltage amplifier to add in a bias voltage from the Spiral Acquisition Algorithm (see Chapter 4). The bias voltage would center the laser beam on the quad cell, and allow data to be taken. Unfortunately, this often added in noise sources (the bias voltage came from a computer, and the DSA supplied the driving voltage, creating unavoidable ground loops).

3. Use a controller with an extremely low crossover frequency (< 0.5 Hz) and use the DSA as a disturbance input. This controller keeps the laser beam pointed on the quad cell for low frequencies, but has no command authority for frequencies of interest. This method was not available at the time of data collection and remains untested.

In the end, method 1 worked the best, even though it was the most laborious. Also, the quad cell is much more sensitive than the KAMAN sensors at detecting angular position. For this reason, the quad cell output voltage did not need to be amplified to produce acceptable results. Moreover, the data is more reliable in the higher

frequencies than the KAMAN data. The experimental frequency response for the Azimuth axis with quad cell sensing is shown in Figure 3-10. As expected, the frequency response looks very similar to the frequency response with KAMAN sensors at lower frequencies. There are however a few important differences. Figure 3-11 shows both the KAMAN and quad cell frequency responses, normalized to the same DC magnitude for easy comparison.

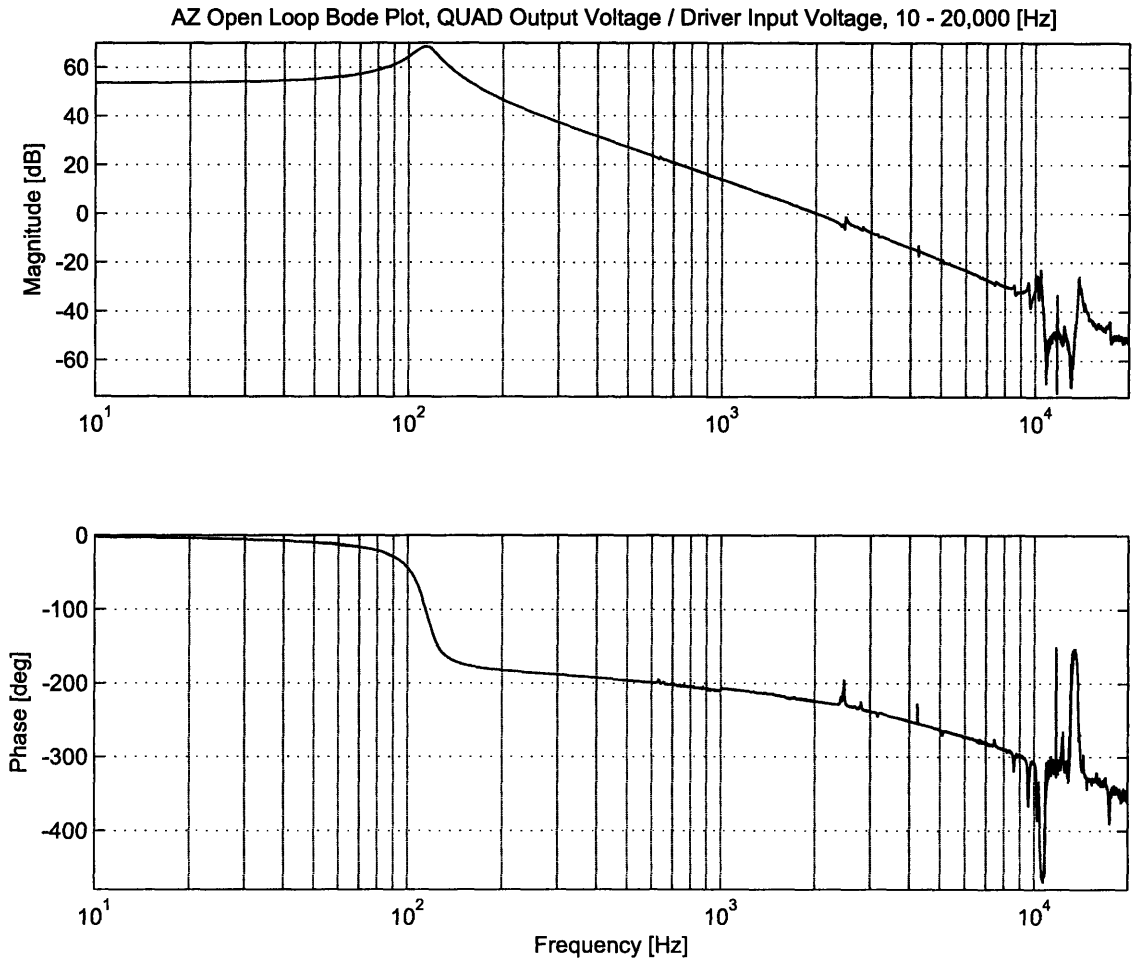

Figure 3-10: FSM Azimuth open loop frequency response, quad cell Voltage Out / Driver Voltage In, 10 - 20,000 Hz. Data taken with a resolution of 1600 points / kHz between 10 Hz and 12 kHz, and 1600 points between 12 and 20 kHz.

Upon initial inspection, several key differences stand out between the KAMAN sensor frequency response and the quad cell frequency response. First, the first doublet of the quad cell transfer function seems to be missing! Figure 3-12 is a zoomed

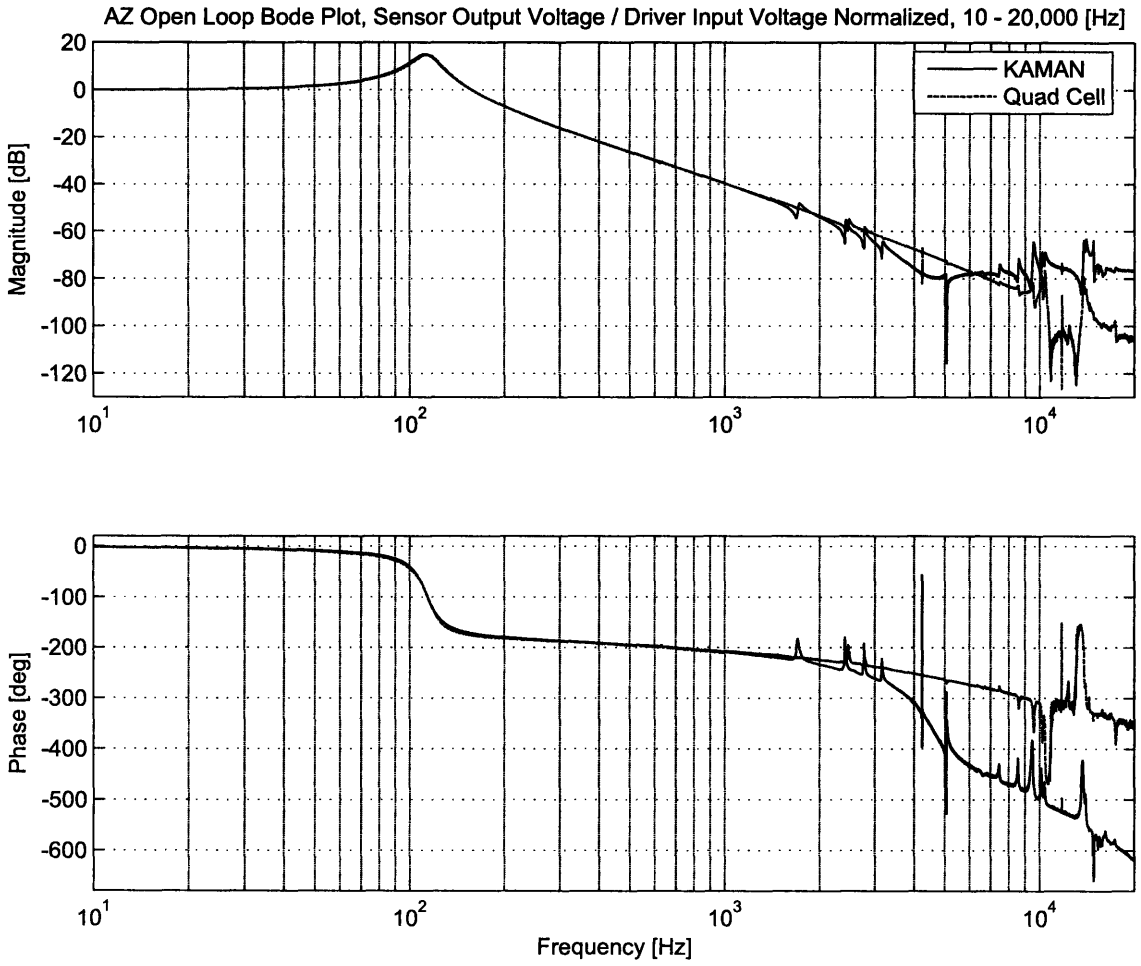

Figure 3-11: Normalized comparison of the FSM Azimuth KAMAN open loop frequency response with the quad cell open loop frequency response, 10 - 20,000 Hz.

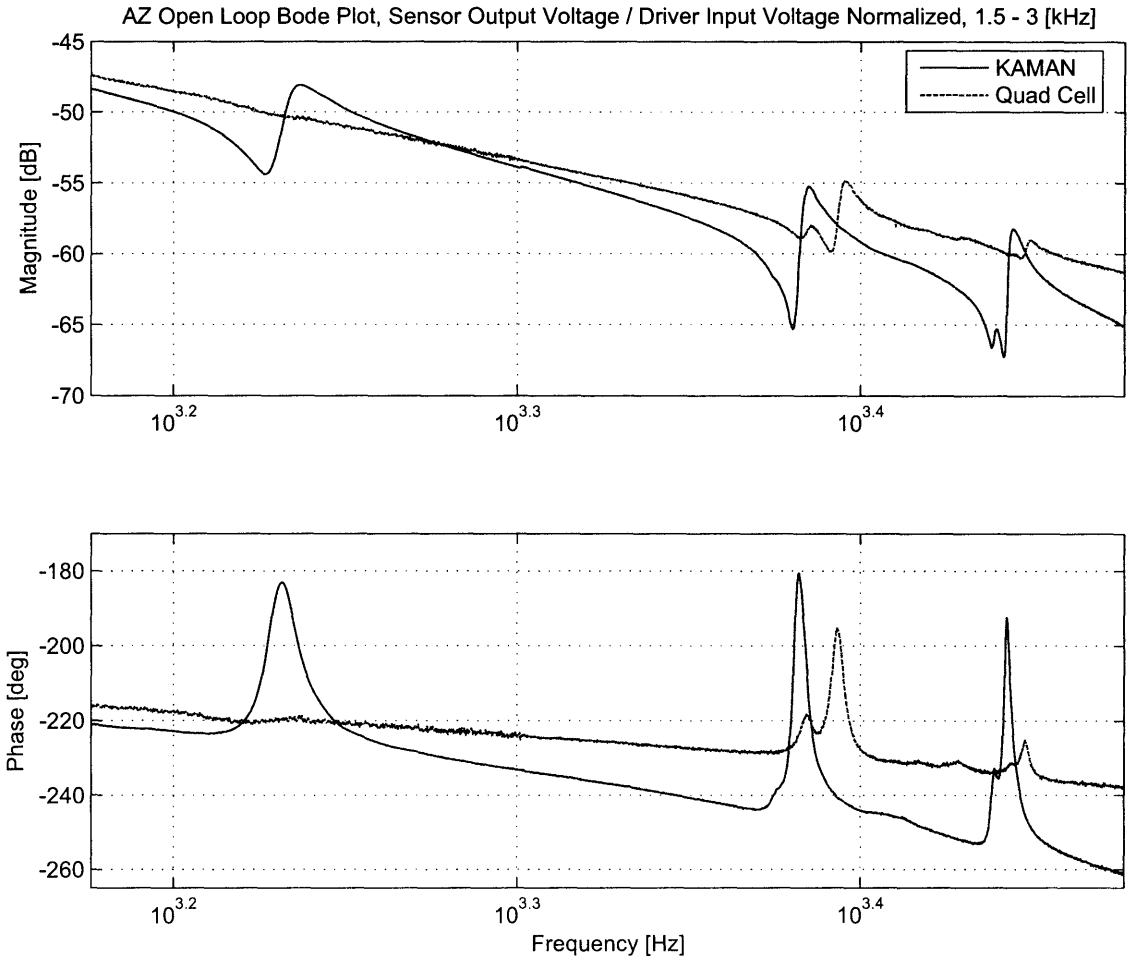

Figure 3-12: Normalized comparison of the FSM Azimuth KAMAN open loop frequency response with the quad cell open loop frequency response, 1.5 - 3 kHz.

version of Figure 3-11 showing the detail around the first few doublets. This "missing" doublet is helpful from a control point of view (a nice -2 decade per decade roll off would be preferable the entire way down), and this also gives valuable information about the FSM. Because the first doublet is missing (or at the very least barely noticeable), it is reasonable to assume that the first doublet of the KAMAN transfer function has to do with some part of the KAMAN target plate assembly. This is feasible because the quad cell measures actual mirror angle, where the KAMAN sensors measure the angle of the KAMAN target plate / mirror / flexure assembly. The next two doublets are at least similar in frequency position, but the relative amplitudes between the two sensors differ significantly. This too gives a little more information on the possible location of these modes, but further investigation is beyond the scope of this thesis.

Continuing to examine Figure 3-11, it is also apparent that the general -2 slope of the quad cell transfer function continues throughout the frequency range, whereas the KAMAN transfer function seems to "level off" after about 5 kHz. This suggests that the earlier hypothesis of unreliable KAMAN data past 5 kHz is correct. The quad cell is much more sensitive than the KAMAN sensors (difference in DC magnitude is about 60 dB). This means that for the given test setup, with the FSM rolling off at -40 dB a decade, the quad cell can gather reliable data about one and a half decades past the KAMAN sensors. Therefore, this transfer function is assumed accurate all the way out to 20 kHz.

Focusing on the quad cell transfer function of Figure 3-10 past 5 kHz, even more dramatic things start to happen. Several modes appear to excite and decouple from the system, leading to a series of dramatic peaks past 10 kHz. Most of these resonances can be ignored, but some cannot (see Section 5.2.3 for further analysis). Finally, in order to effectively simulate the FSM Azimuth axis in MATLAB, a parametric model must be built. The building of this model follows the same procedure outlined in Section 3.3, except the number of doublets is much higher. In all, the parametric model for the Azimuth axis includes the main 2^{nd} order equation, 19 doublets, and 4 higher frequency poles with 2 higher frequency zeros to control overall slope and

phase. This model overlays the experimental data in Figure 3-13.

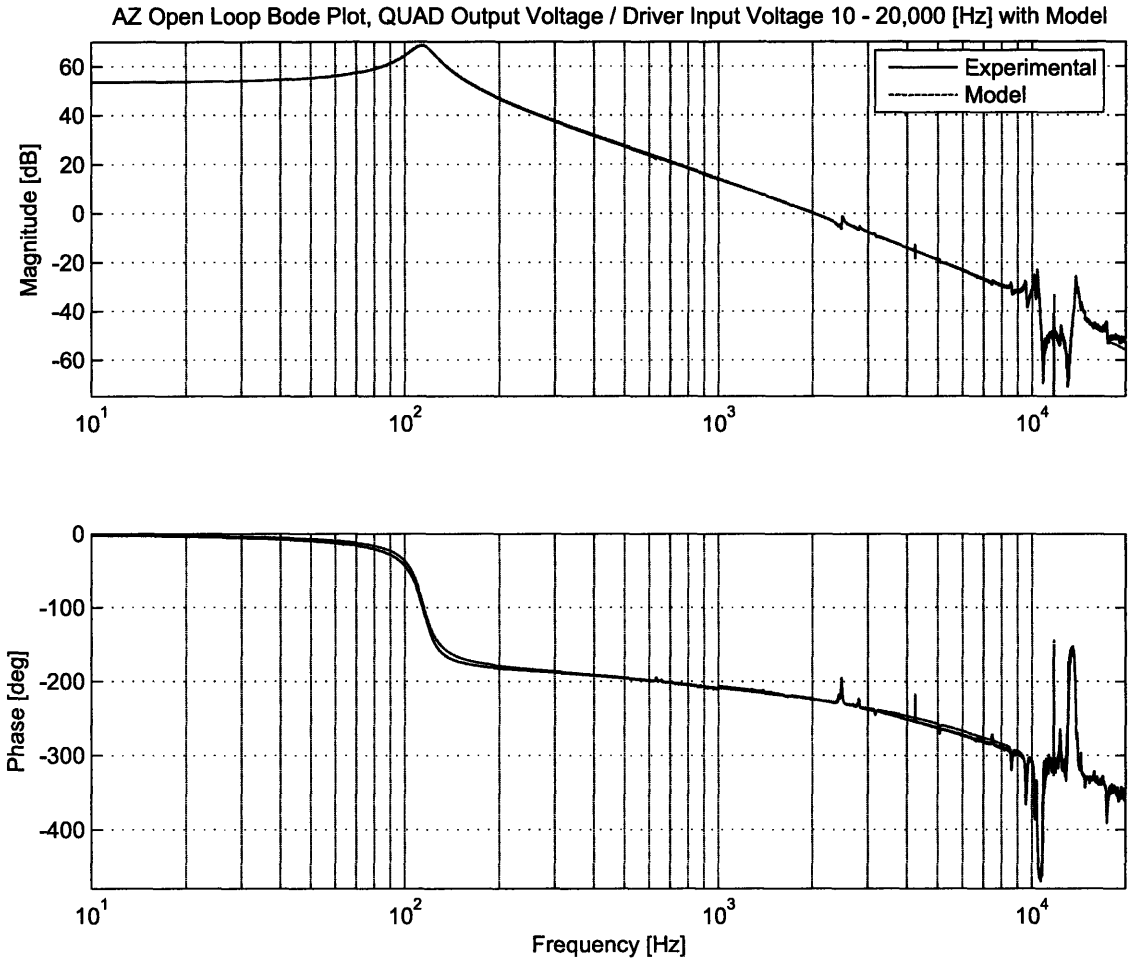

Figure 3-13: FSM Azimuth open loop quad cell frequency response with parametric model, 10 - 20,000 Hz.

The state-space representation of this model is documented in Appendix D. This model can be used to simulate the FSM Azimuth axis in MATLAB; however, a reduced order model would work in the frequencies of interest.

As before with the KAMAN responses, the quad cell frequency responses of the Azimuth and Elevation axes are virtually identical to each other. All of the Elevation counterpart figures are shown in Figures 3-14 through 3-17. Also as before, a state-space model was created to represent the Elevation axis in MATLAB and is presented in Appendix E. In all, this parametric model includes the main 2^{nd} order equation,

21 doublets, and 4 higher frequency poles with 2 higher frequency zeros to control overall slope and phase.

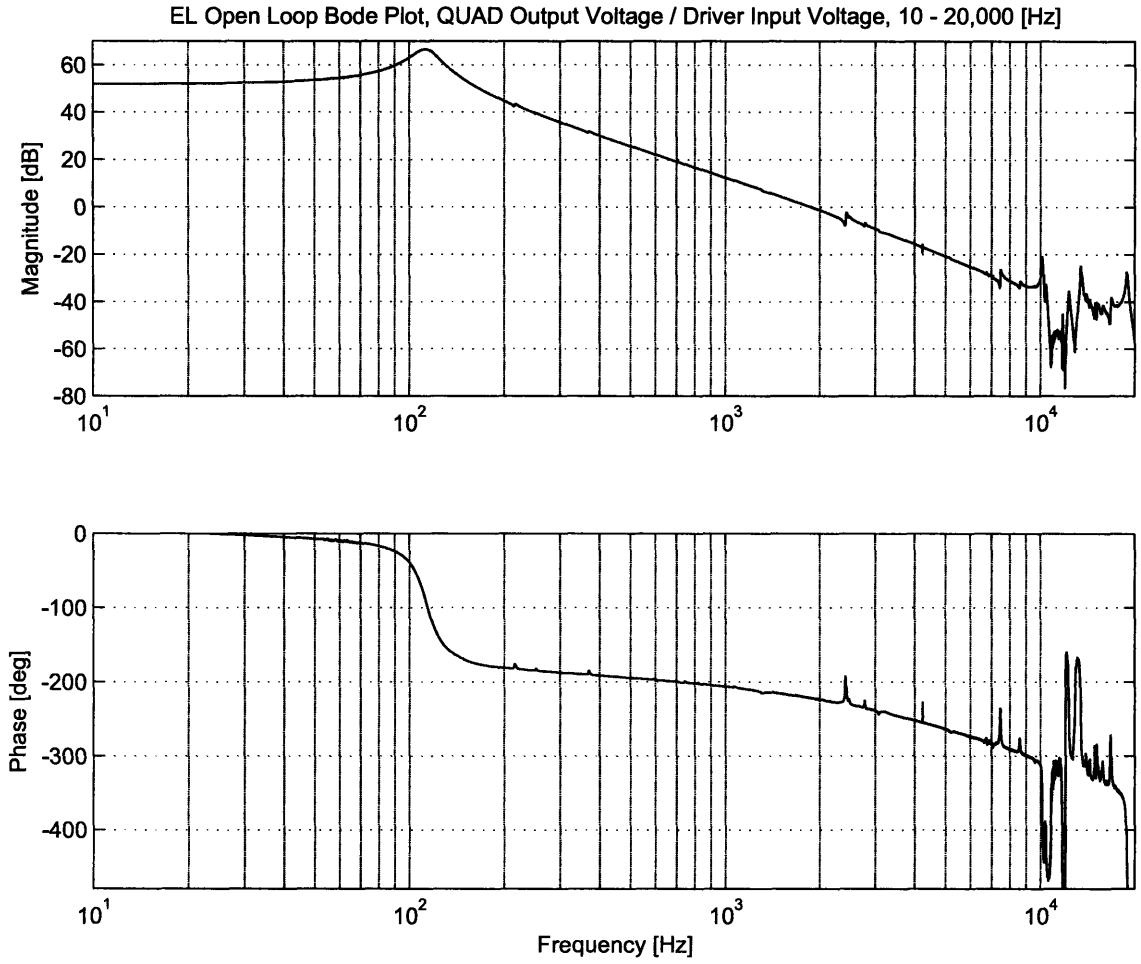

Figure 3-14: FSM Elevation open loop frequency response, quad cell Voltage Out / Driver Voltage In, 10 - 20,000 Hz. Data taken with a resolution of 1600 points / kHz between 10 Hz and 12 kHz, and 1600 points between 12 and 20 kHz.

Figure 3-18 shows the comparison between the Azimuth and Elevation axes. The plots are virtually identical before 5 kHz except for a small gain shift (probably a difference in the current drivers or voice coil actuators) and small shifts in frequency of the doublets (probably small differences in manufacturing and assembly of the axes). After 5 kHz the only differences are very small and will not be considered in the application of the controller. For all intents and purposes, the axes can be thought of as identical. Most of the original controller design in Chapter 5 focuses on

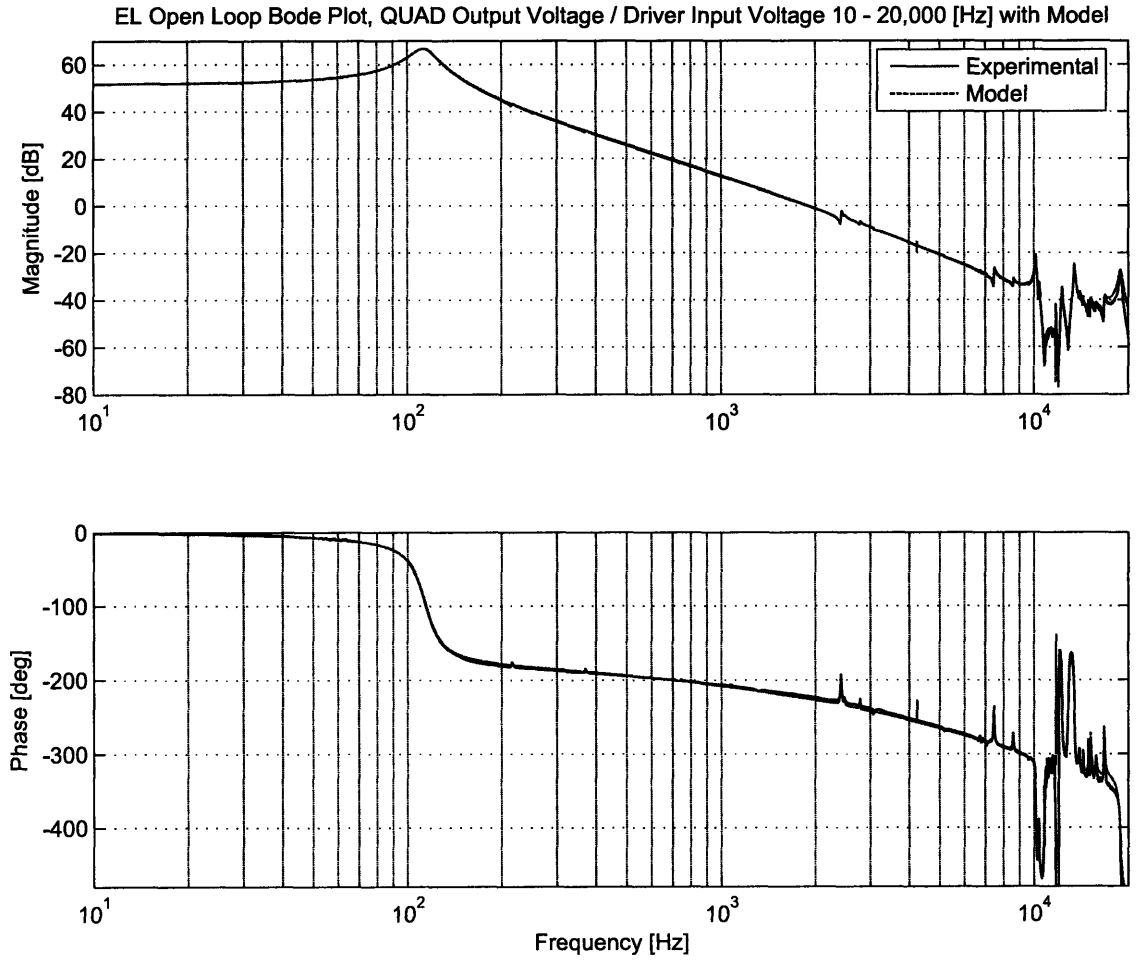

Figure 3-15: FSM Elevation open loop quad cell frequency response with parametric model, 10 - 20,000 Hz.

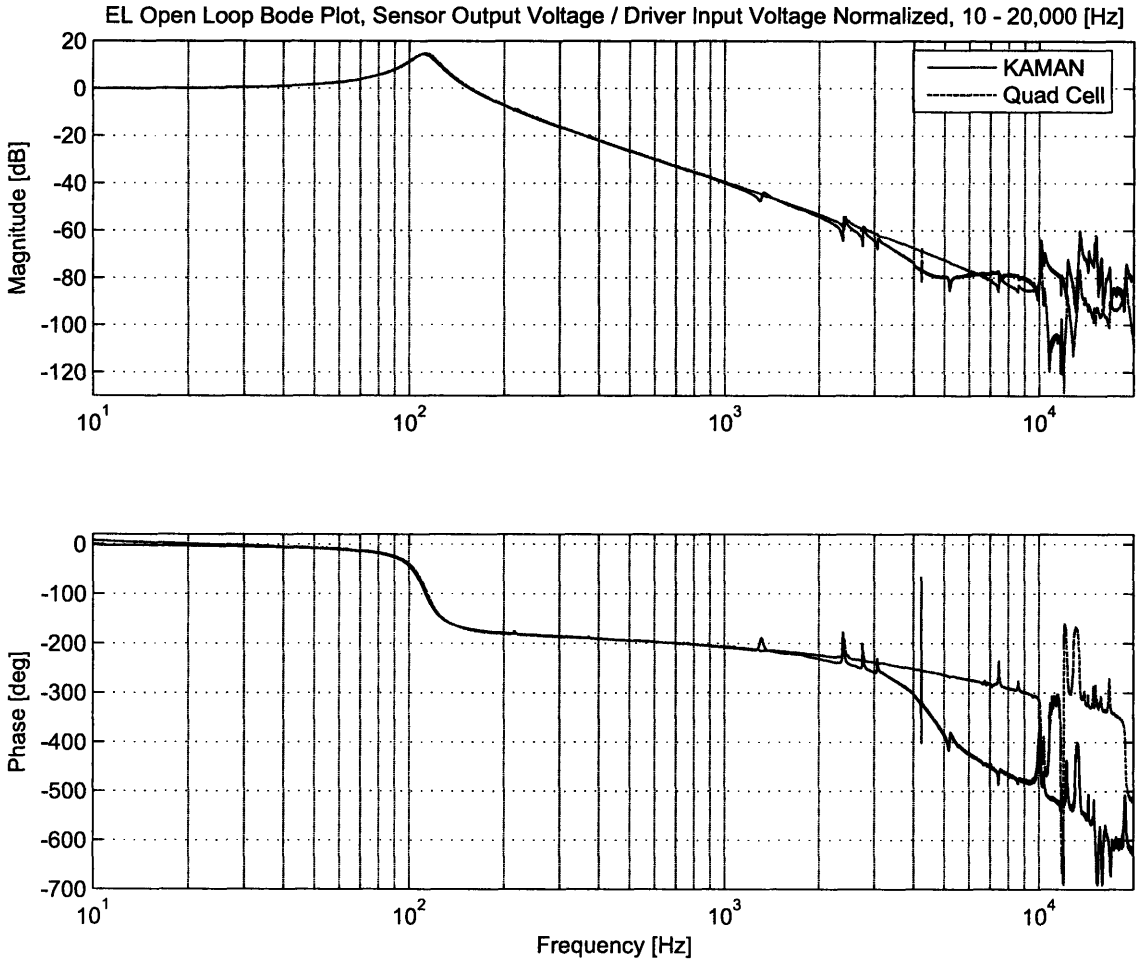

Figure 3-16: Normalized comparison of the FSM Elevation KAMAN open loop frequency response with quad cell open loop frequency response, 10 - 20,000 Hz.

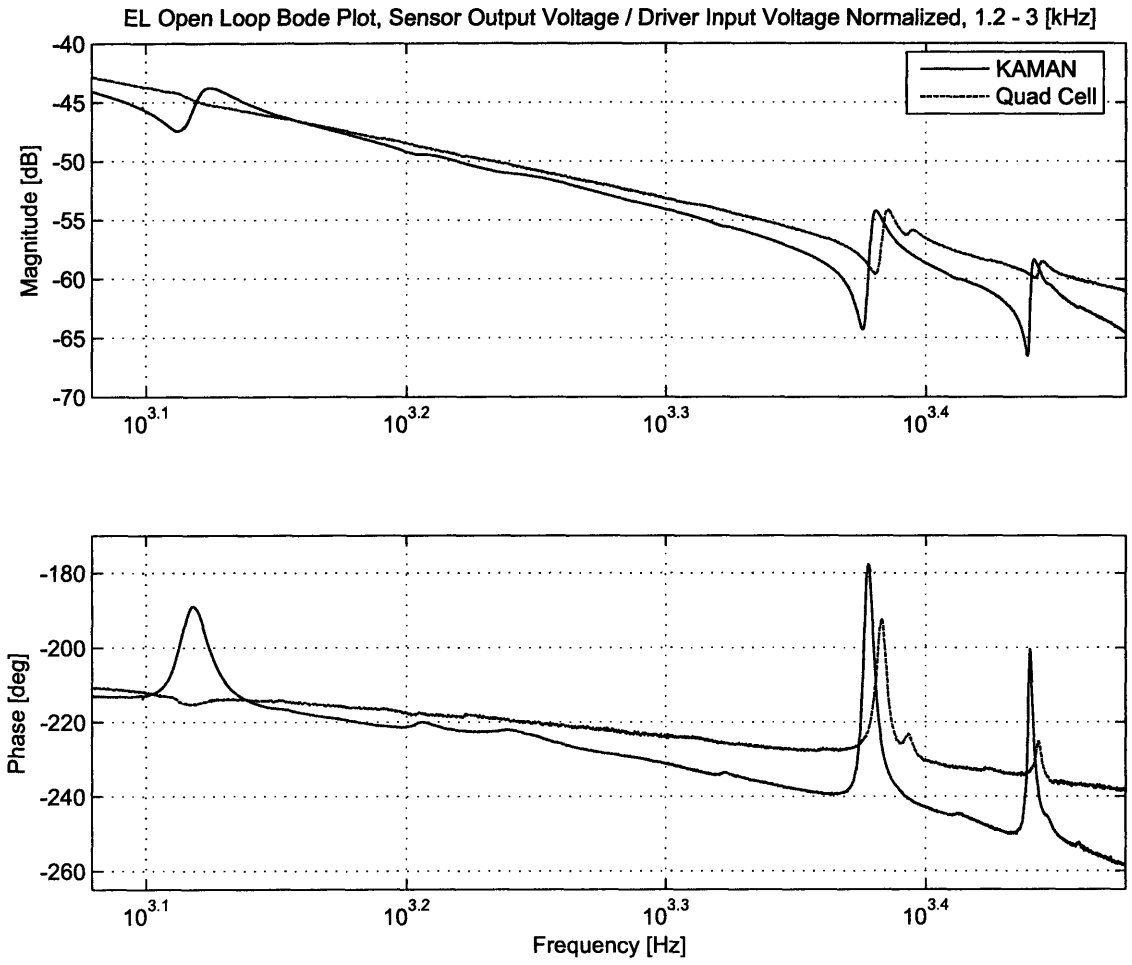

Figure 3-17: Normalized comparison of the FSM Elevation KAMAN open loop frequency response with quad cell open loop frequency response, 1.2 - 3 kHz

controlling the Azimuth axis only, although the controller can be applied directly to the Elevation as well.

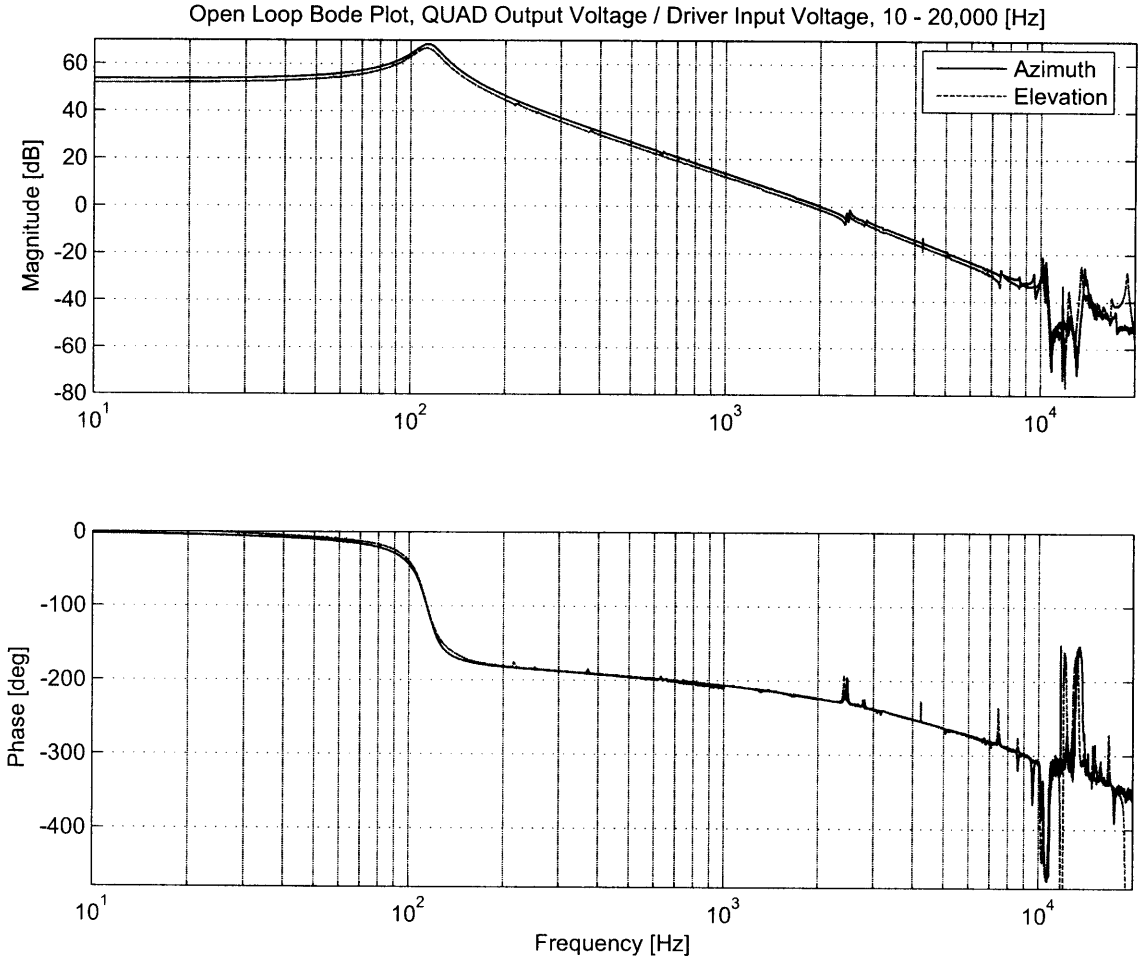

Figure 3-18: FSM Azimuth and Elevation Open Loop Frequency Response Overlay, KAMAN Voltage Out / Driver Voltage In, 10 - 20,000 Hz

There are several reasons for taking such extremely fine resolution data of the FSM. The first is to examine the differences between the KAMAN and quad cell sensor outputs for the same hardware. Examining these differences provides clues on mode locations that are not apparent looking at a low resolution frequency response (even at 1600 points / kHz, some doublets only have 20 data points describing them). Moreover, examining the differences provide clues on mode locations (modes that occur in KAMAN sensing and not with the quad cell point to modes in the KAMAN

target plate). Developing a high order quad cell model, while time consuming, allows very accurate simulation of the hardware with an infinite number of controllers. Designing in MATLAB with code is much faster than implementing each controller and taking experimental frequency responses. By using the high order model, the design of the compensator in Chapter 5 can be done where it should be, on paper, and then experimentally verified.

Chapter 4

Spiral Acquisition

This chapter investigates how xPC and Simulink are used to find the active area of the quad cell by steering the FSM while the feedback loop is closed using the KAMAN sensors (known as closed loop KAMAN mode). Section 4.1 discusses the reasons for needing this capability. Section 4.2 presents the basic algorithm requirements to complete the acquisition process. Section 4.3 lists and describes the different ways of steering the FSM to point the laser beam onto the quad cell. Finally, Section 4.4 details the algorithm that allows the seamless transition from closed loop KAMAN mode to optical tracking mode once the FSM has steered the laser beam onto the quad cell.

4.1 Motivation

The motivation for choosing an acquisition algorithm is to provide quick method for initiating optical feedback mode. The FSM cannot enter optical tracking mode until the laser spot is pointed onto the active region of the quad cell. In the most basic case, without a spiral acquisition algorithm, one would have to type in KAMAN position commands at random trying to point the laser beam onto the quad cell. It is also possible to use the mounting hardware to point the laser beam onto the quad cell, but this is extremely time consuming and not available during spaceflight. Furthermore, the KAMAN sensors are known to drift. Depending on the feedback configurations, a

1 mV change in the KAMAN sensor output can lead to a 1 V change in the quad cell output. With the quad cell having an active region of about ± 10 V (when using C/R board and the boxed quad cell), the beam can move from dead center to completely off the quad cell with a KAMAN drift of only 5 mV, requiring constant correction.

The main reason for using spiral acquisition is that by doing so, the resulting system is robust and self-correcting. By using an algorithm to find the quad cell, it is insured that if for any reason the quad cell is lost, it can be found again relatively quickly. This feature is incredibly important for space-based systems as there is no hardware adjustment possible in space. Implementing a robust, programmable algorithm to find the quad cell and enter optical tracking mode is essential to the success of any laser-based communications system.

4.2 Basic Algorithm Requirements

On the most basic level, the spiral algorithm must be able to do each of the following things:

1. Command the FSM in a controlled two dimensional pattern.
2. Take in sensor voltages of the quad cell.
3. Apply basic math to the quad cell sensor inputs to determine if the laser beam is in the linear region of the quad cell, preferably close to the center.
4. If the beam is on the quad cell, smoothly switch over to quad cell feedback mode.

The first two items are easily handled by the A/D and D/A PMC-ADADIO blocks. These blocks allow communication between Simulink and the analog world. All that it left is to manipulate those input and output voltages to achieve the second two desired functions. These functions can be accomplished by using Stateflow.

4.2.1 Stateflow Implementation with xPC

Stateflow[1] allows the creation of a Finite State Machine in Simulink. A Finite State Machine is any device with a collection of states that are traversed in a specific order dependent on the values of certain parameters. A basic example Stateflow diagram is shown in Figure 4-1.

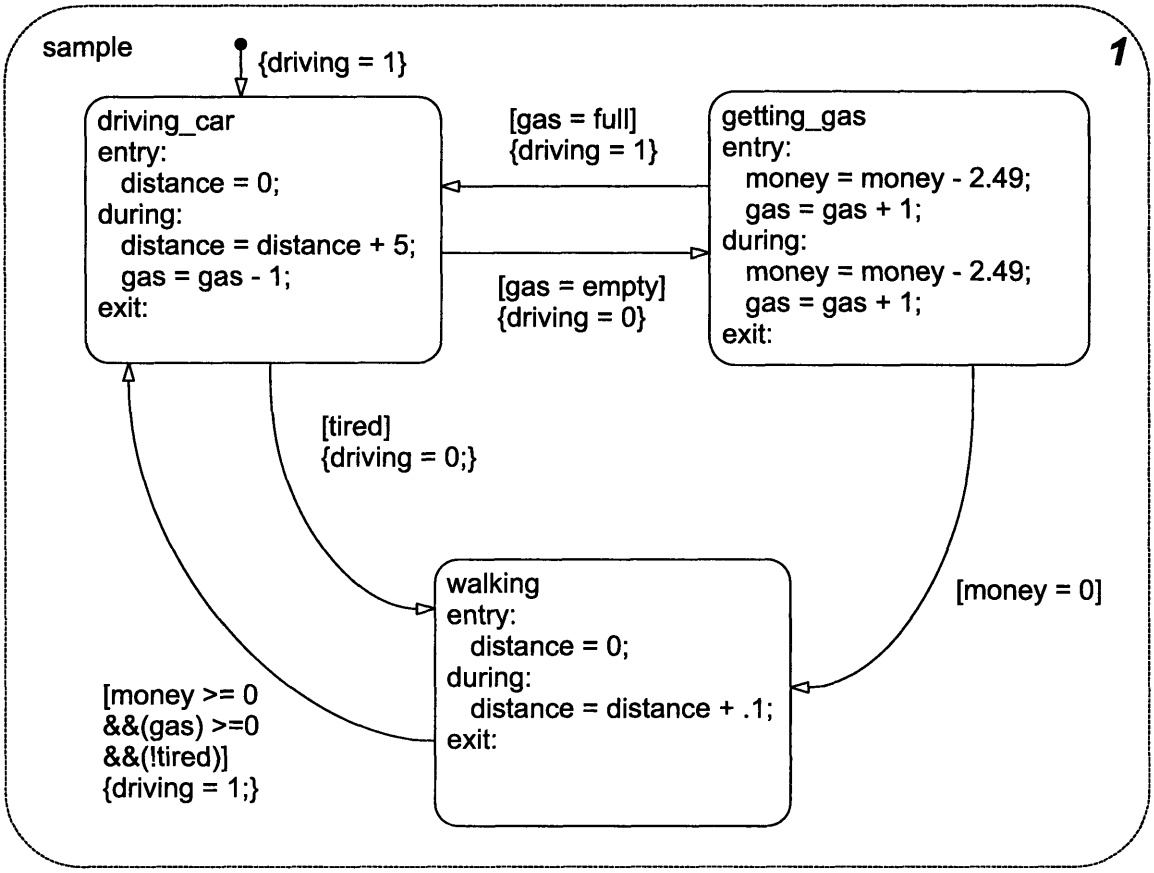

Figure 4-1: Example of a Stateflow diagram.

The three states in this Stateflow diagram are *driving_car*, *getting_gas* and *walking*. The dot with the arrow signifies the starting point of the diagram, which is *driving_car* in this case. The variables in this diagram are *distance*, *gas*, *full*, *empty*, *money*, *tired*, and *driving*. Assume that all variables are initialized (they usually would be initialized

[1]For more information about Stateflow including documentation and examples, see `http://www.mathworks.com/access/helpdesk/help/toolbox/stateflow/`.

in a different state. At startup, *driving* is set to 1 and the *driving_car* state is entered, setting *distance* to 0. One of three things happens at the next state transition. If *tired* is non zero, the *walking* state is entered, and both *driving* and *distance* are set to zero. If $gas = empty$, then the *getting_gas* state is entered, setting *driving* to 0, decreasing *money* by 2.49 and increasing *gas* by 1. Or, if neither of those conditions was met, then it remains in the *driving_car* state, and *distance* is increased by 5 and *gas* is decreased by 1. Then, the process repeats.

If the *getting_gas state* is the current state, one of 3 things happens on the next transition. If $money = 0$, the state changes to *walking* and *distance* is set to 0. If $gas = full$, then *driving* is set to 1, the *driving_car* state is entered and *distance* is set to 0. Otherwise, *money* decreases by 2.49, *gas* increments by 1 and the state remains *getting_gas*.

If *walking* is the current state, one of 2 things can occur. If $money \geq 0$, $gas \geq 0$ and $tired = 0$ (*!tired* is the logical NOT of *tired*), then *driving* is set to 1, the *driving_car* state is entered and *distance* is set to 0. Otherwise, *distance* is increased by .1 and the state remains *walking*.

Any of these parameters could be outputted to the next level of Simulink and used as inputs to other sections or as outputs via D/A channels on the PMC-ADADIO cards. This is the basic concept behind using a Stateflow algorithm. It's easy to take the 3 quad cell inputs, perform some math on them and change states if they satisfy the conditions set in the acquisition loop (see Section 4.4) It's also easy to output variables to represent the commands to each of the 4 feedback loops (Azimuth and Elevation for both KAMAN and quad cell Feedback). The Stateflow implementation of each steering algorithm is described in Section 4.3.

4.3 Steering The Beam

The first step in the acquisition algorithm is steering the beam path in a pattern. There are many different ways to steer the FSM in a patterned search. This section details the three different methods used and lists out the benefits and complications

with using each of them.

4.3.1 Square Grid

The first method utilized is the simplest form of searching: a square grid of evenly spaced search points. The grid is searched in a spiral, working from the middle out to the edges.

Square Spiral Algorithm

The Stateflow diagram for this algorithm is shown in Figures 4-2 and 4-3. Figure 4-2 shows the blocks responsible for steering the beam and Figure 4-3 shows the block responsible for the acquisition mode of the algorithm (see Section 4.4). Stateflow input, output and local variables used in Figures 4-2 and 4-3 are listed and described in Tables 4.1, 4.2 and 4.3 respectively.

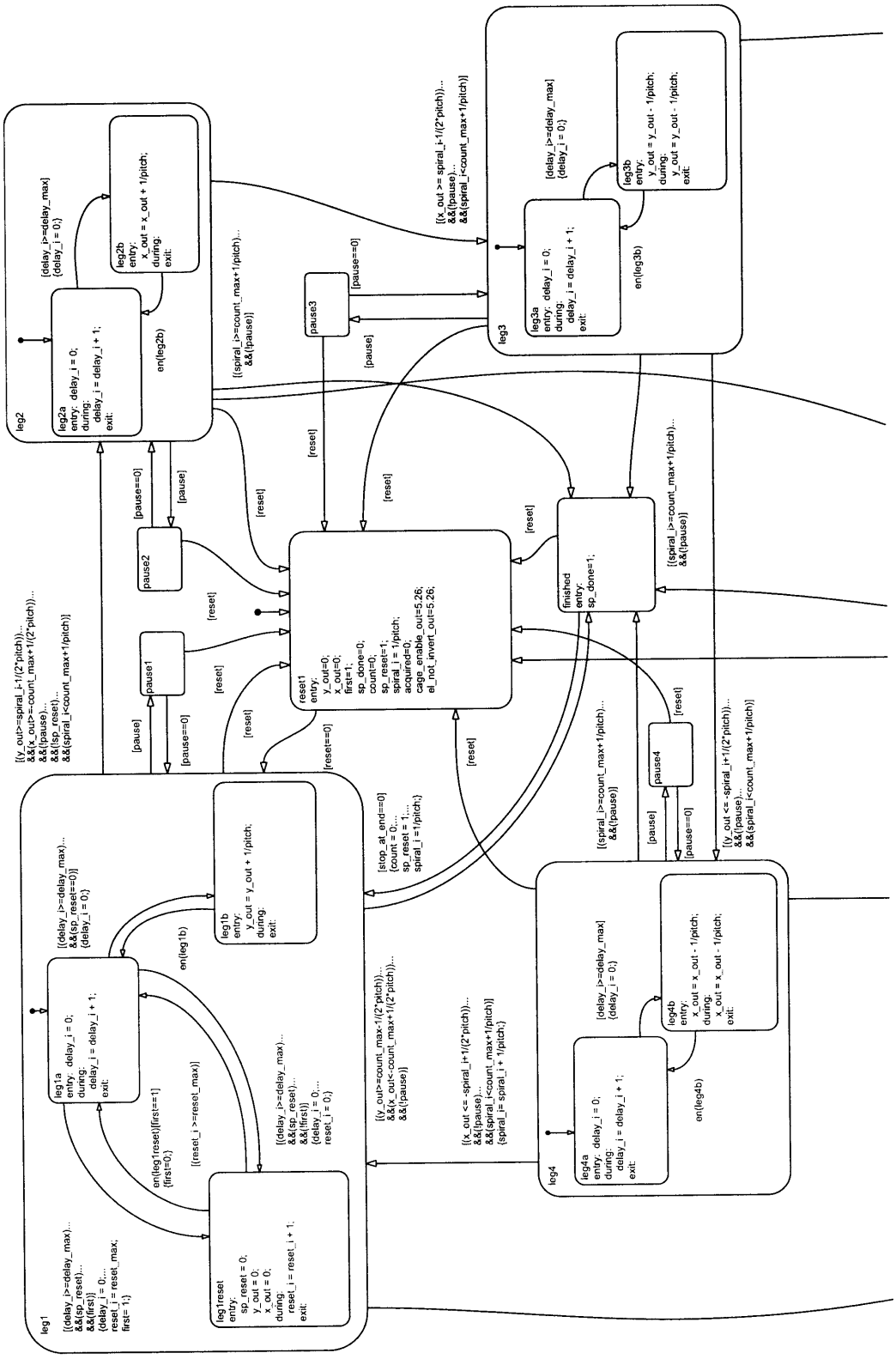

Figure 4-2: Stateflow diagram of the square spiral algorithm (1/2).

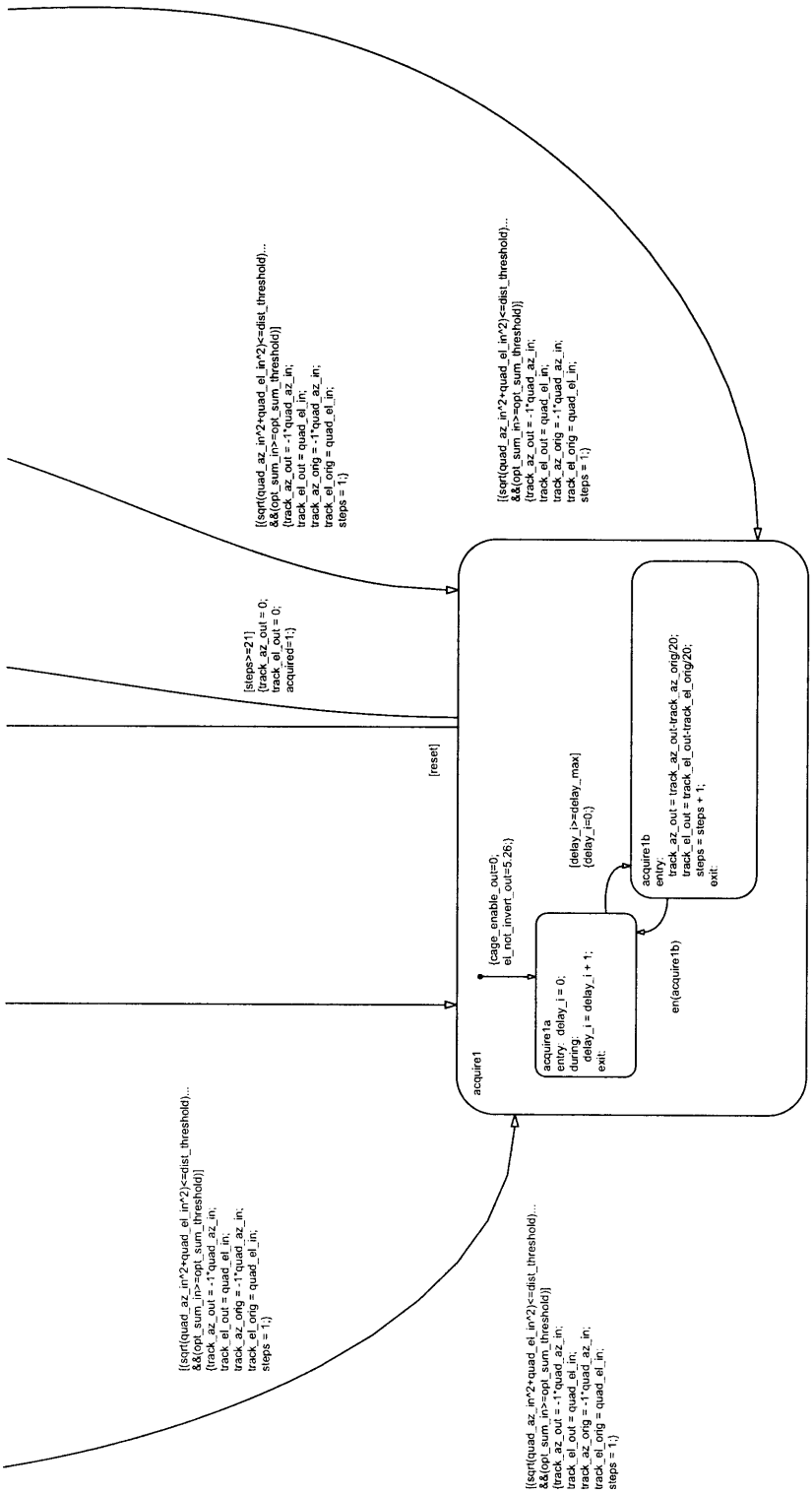

Figure 4-3: Stateflow diagram of the square spiral algorithm (2/2).

Name	Description
step_in	Trigger port. Stateflow advances to the next state on rising edge.
pause	When set to 1, pauses the spiral algorithm. Resumes when set to 0.
count_max	Spiral stops when either $\|x\_out\|$ or $\|y\_out\|$ reaches this value.
delay_max	Time (in cycles) before evaluating to next state.
reset_max	Time (in cycles) after reset or finishing to start algorithm over.
pitch	Controls step size. The difference between subsequent commanded voltages *x_out* and *y_out* is $pitch^{-1}$.
reset	Resets algorithm when 1. Initializes and starts algorithm when set back to 0.
stop_at_end	When 1, will not repeat spiral if acquisition did not occur. When 0, repeats until acquisition occurs.
quad_az_in	Azimuth input from the quad cell. Used to determine distance from center of quad cell.
quad_el_in	Elevation input from the quad cell. Used to determine distance from center of quad cell.
opt_sum_in	Optical Sum input from the quad cell. Used to determine if on active region quad cell.
dist_threshold	Sets how close the beam must be to the origin of the quad cell before entering acquisition mode. If $dist\_threshold \geq \sqrt{quad\_az\_in^2 + quad\_el\_in^2}$, algorithm switches to acquisition mode (provided that $opt\_sum\_threshold \leq opt\_sum\_in$).
opt_sum_threshold	Sets how high the Optical Sum input from the quad cell must be before entering acquisition mode. If $opt\_sum\_threshold \leq opt\_sum\_in$, algorithm switches to acquisition mode (provided that $dist\_threshold \geq \sqrt{quad\_az\_in^2 + quad\_el\_in^2}$).

Table 4.1: Table of Stateflow input variables used in the square spiral algorithm.

Name	Description
x_out	Command to KAMAN Azimuth axis.
y_out	Command to KAMAN Elevation axis.
track_az_out	Command to quad cell Azimuth axis.
track_el_out	Command to quad cell Elevation axis.
cage_enable_out	Digital signal controlling feedback mode (switching between KAMAN and quad cell) of FSM when using the original analog controller.
el_not_invert	Digital signal controlling feedback mode (switching between KAMAN and quad cell) of FSM when using the original analog controller.
acquired	Is 1 if the algorithm found the quad cell and entered acquisition mode, 0 otherwise.
sp_done	Is 1 if algorithm is in *finished* state, 0 otherwise.

Table 4.2: Table of Stateflow output variables used in the square spiral algorithm.

Name	Description
delay_i	Counter for *delay_max*.
first	Is 1 if coming out of *reset1* state, otherwise 0. Used to avoid waiting *reset_delay* before starting spiral.
reset_i	Counter for *reset_max*.
sp_reset	Set to 1 when algorithm is to wait *reset_delay* (if $first = 0$). Is set to 0 only in *leg1reset*.
spiral_i	Counter for allowable *x_out* and *y_out* values. Output changes directions when *x_out* or *y_out* is greater than $\lvert spiral\_i - pitch^{-1}/2 \rvert$. Is incremented by $pitch^{-1}$ after exiting *leg4*. Spiral algorithm finishes without entering acquisition mode when $spiral\_i \geq count\_max + pitch^{-1}$.
steps	Counter used for bumpless transfer.
track_az_orig	Stores original quad cell Azimuth command used when switching to quad cell tracking mode. Used for bumpless transfer.
track_az_orig	Stores original quad cell Elevation command used when switching to quad cell tracking mode. Used for bumpless transfer.

Table 4.3: Table of Stateflow local variables used in the square spiral algorithm.

Figure 4-2 shows the relevant Stateflow block diagram controlling the actual steering of the beam. This main steering code was originally written by Jamie Burnside and modified for use in xPC with the analog controller. The output commands controlling the position of the FSM always move in one of four directions: up, right, down, or left. The Stateflow states *leg1*, *leg2*, *leg3*, and *leg4* in Figure 4-2 each handle moving the commands in each direction: *leg1* only moves the command output *y_out* position up, *leg2* moves the *x_out* position up (thus moving right), *leg3* moves down and *leg4* moves left. The algorithm itself is very simple. For illustration purposes, define $pitch = 2$, $count\_max = 3$. The Stateflow diagram initializes in the *reset1* state, then moves to *leg1*. If *delay_max* has a positive value, the algorithm stays in *leg1a* for *delay_max* steps, then evaluates *leg1b* (this behavior occurs in all states and allows control over the speed of the spiral, but will be ignored for the rest of this section). The first decision is now made, as *y_out* has now been increased one step to

a value of .5 V. Moving to the next state occurs when

$$x\_out \text{ OR } y\_out > \left| spiral\_i - pitch^{-1}/2 \right| \tag{4.1}$$

The transition does occur because $|spiral\_i - pitch^{-1}/2|$ is .25 V in this example. The algorithm moves to *leg2*. In this state, *x_out* is increased until it passes .25 V. It does this in one step and reaches .5 V. After this, the state changes again to *leg3*. *leg3* decreases the value of *y_out* in .5 V increments until its absolute value passes .25 V. This occurs after 2 steps, stopping with $y\_out = -.5$ V. Finally, *leg4* is entered, and the same process occurs, with *x_out* decreasing to -.5 V after 2 steps. The cycle is completed with a transition back to *leg1*. However, the counter *spiral_i* is incremented by $pitch^{-1}$, so the commands will increment 2 additional times in each state on the next rotation. This means that *leg1* and *leg2* will increment 3 times, and *leg3* and *leg4* will increment 4 times before *spiral_i* is incremented again. This process continues, producing an ever increasing spiral, as shown in Figure 4-4 with $pitch = 2$ and $count\_max = 3$.

The spiral will leave these steering states when one of two conditions are met: either the quad cell is found and the algorithm enters acquisition mode (see Section 4.4) or $spiral\_i > count\_max + pitch^{-1}$. If this occurs, the algorithm has not found the quad cell and resets itself. If $stop\_at\_end = 1$, the algorithm enters the $finished$ state and is dormant until reset. If $stop\_at\_end = 0$, the algorithm enters *leg1* to start another spiral. The first state visited is *leg1reset*, where the *x_out* and *y_out* values are reset and the algorithm stays for *reset_max* cycles. Then, the entire process starts again.

xPC Implementation

For this steering algorithm to be useful, it must be integrated into the Simulink model running on the xPC Target machine. Figure 4-5 shows the overall block diagram of this implementation with appropriate A/D, D/A and Digital I/O blocks. This implementation is the same for all three spiral algorithms. Figure 4-6 shows the

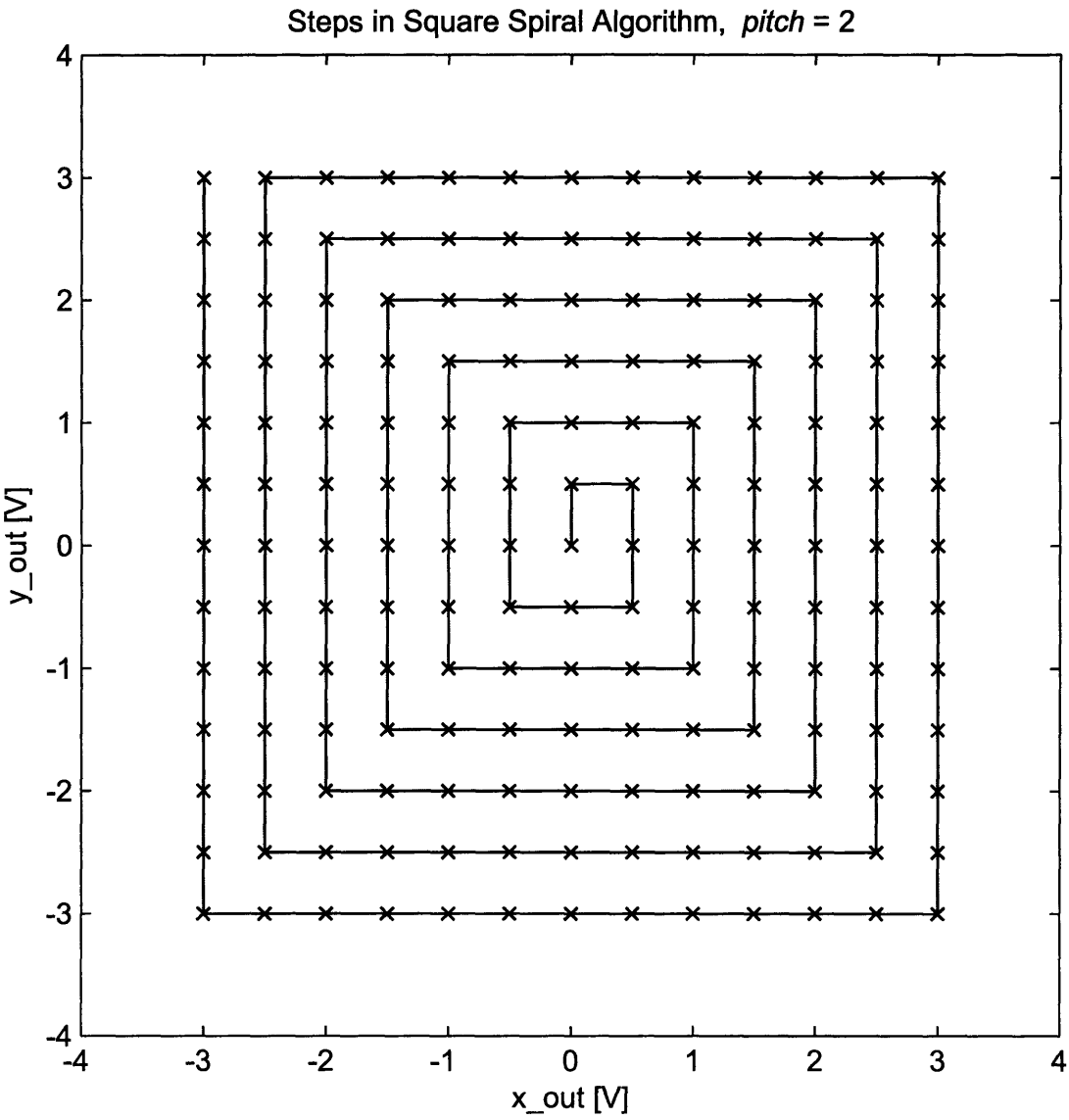

Figure 4-4: X-Y plot of the steering path of the square spiral algorithm with $pitch = 2$ and $count\_max = 3$.

model contained in the main 'Spiral Locator' block.

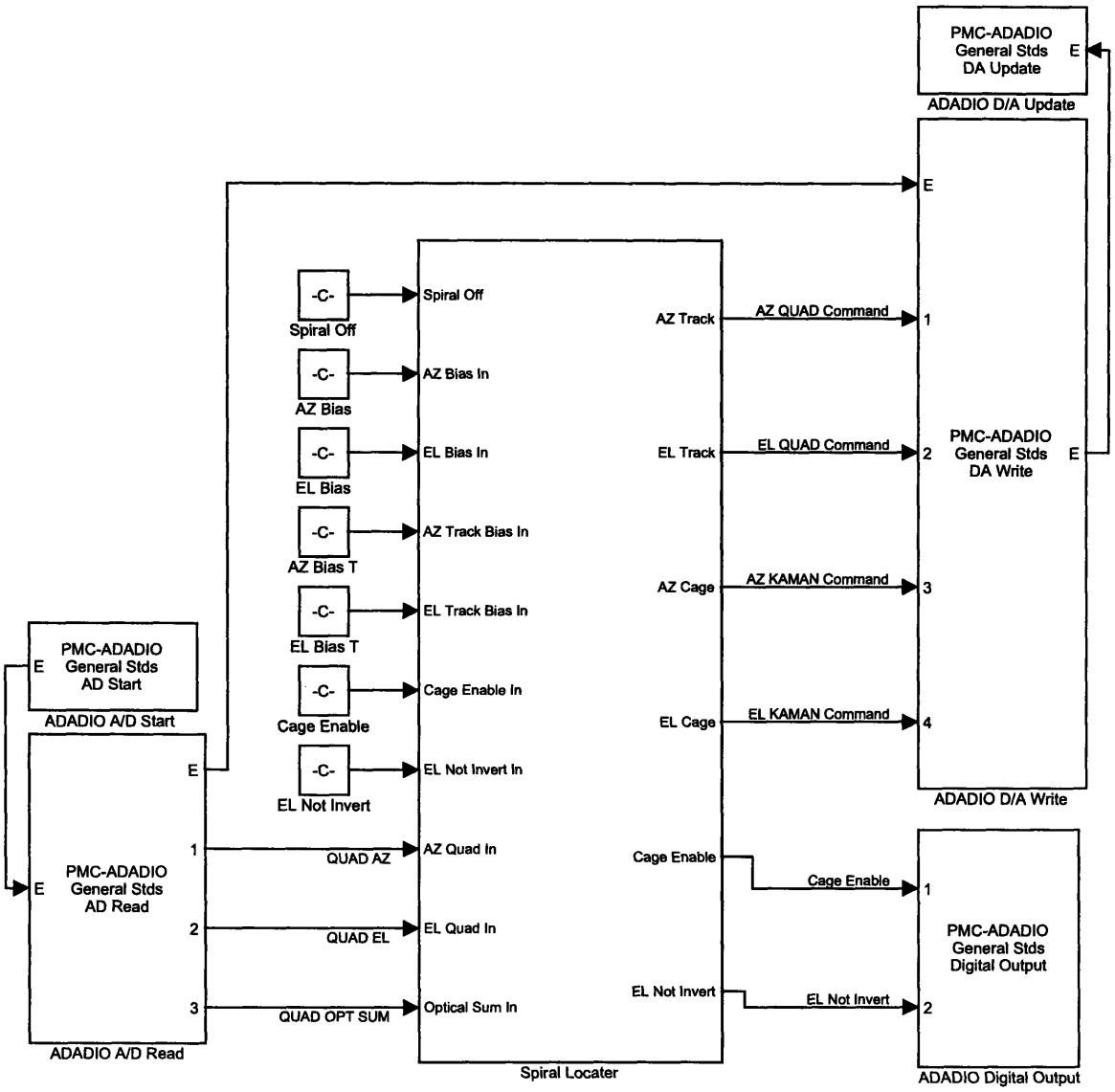

Figure 4-5: Simulink model of the top level xPC implementation for all spiral algorithms.

Figures 4-5 and 4-6 show that for this algorithm to work, it must take in 3 analog signals (the quad cell Azimuth, Elevation and Optical Sum voltages), 8 different parameter values besides the values listed in Tables 4.1, 4.2 and 4.3, and it must output 4 analog signals (command voltages to each of the 4 FSM feedback loops) and 2 digital outputs (controlling which feedback loop to use for each axis). The

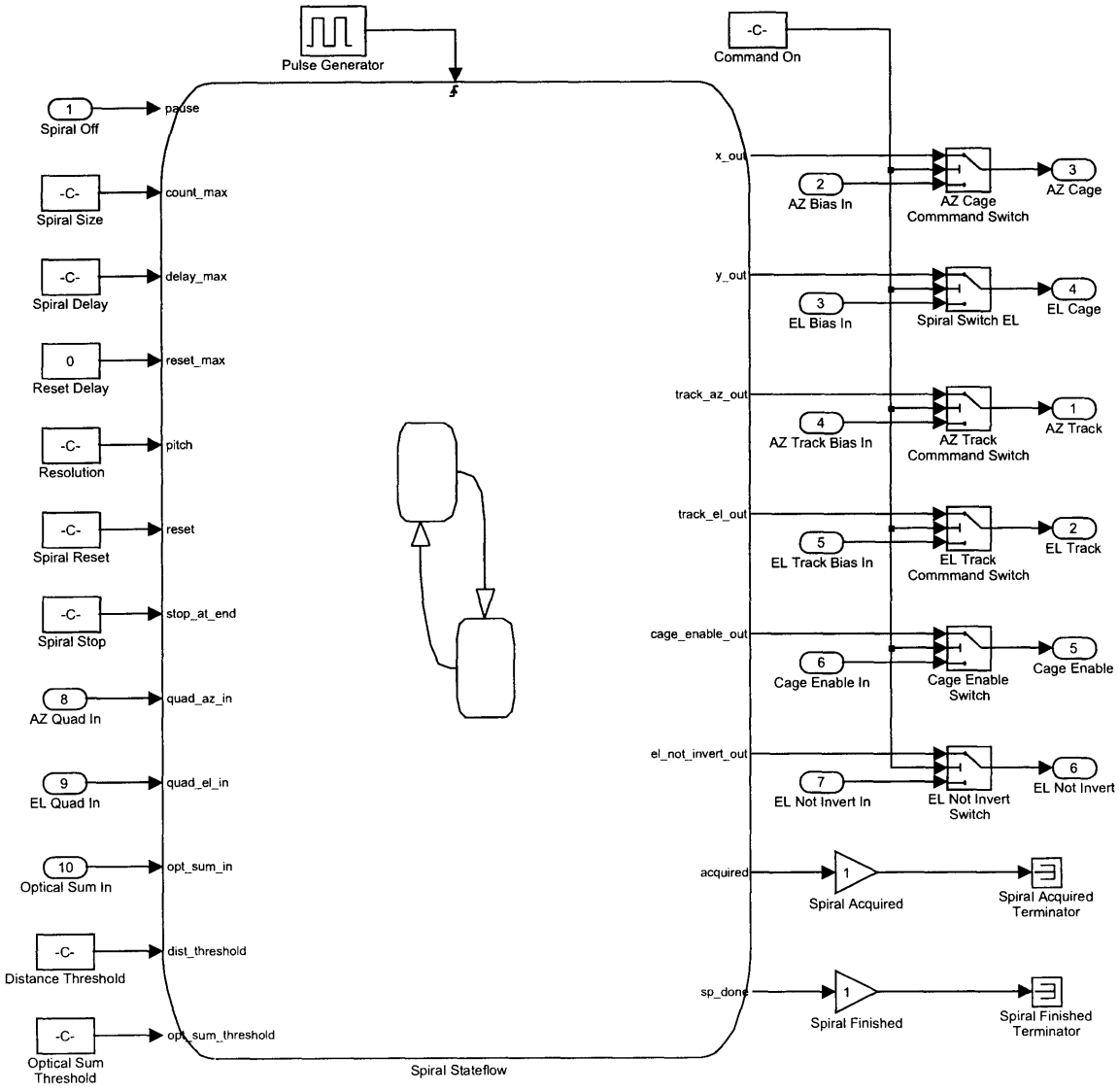

Figure 4-6: X-Y plot of the steering path using the CLV spiral algorithm,

parameters in Figures 4-5 and 4-6 shown with a '-C-' in them are all parameters that can be controlled and changed in real time on the xPC Host Machine. Figure 4-7 shows the GUI used with the square spiral algorithm.

The orange 'Spiral Locater' box controls the square spiral algorithm. The associations between parameter names in Figure 4-7 and the parameters in Figures 4-5 and 4-6 are fairly self evident and a detailed description will not be given. However, there are a few things of note. The FSM 'Zero Bias' commands are added to the 'Command Cage' commands to make up the 'Bias' blocks of Figure 4-5. The 'Command Track' boxes link directly to the 'Bias T' parameters of Figure 4-5. The 'Max Distance' box links to the 'Distance Threshold' box of Figure 4-6. Looking back at Figure 4-6, there is a pulse generator controlling the Stateflow block. This generator has a controllable frequency, so the intervals between state transitions can be controlled. The frequency of the generator was usually left at 100 [Hz], meaning that the states would transition 100 times a second and search 100 steering points for the quad cell. The 'Spiralize' button controls the initiation of the square spiral algorithm. The code for this button press is shown in Appendix G.

There are only two parameters available in Figure 4-7 that directly control the steering pattern: 'Spiral Size' and 'Resolution'. Changing the 'Spiral Size' parameter only affects the maximum voltage commanded by the square spiral algorithm. The grid pattern is unchanged. Changing the 'Resolution' parameter doesn't affect the maximum voltage commanded, but does change the grid pattern. Increasing the 'Resolution' parameter leads to a much denser grid with more complete revolutions to reach the maximum command voltage. The effects of changing these parameters are shown in Figures 4-8 and 4-9 respectively.

Why Not Use The Square Spiral?

The square spiral algorithm is attractive because of its simplicity. The grid pattern is very predictable and the steering pattern is the perfect search pattern. Unfortunately, there is a reason not to use it. The steering path consists of nothing but straight lines and 90° turns. This means that there are very high accelerations at the corners of

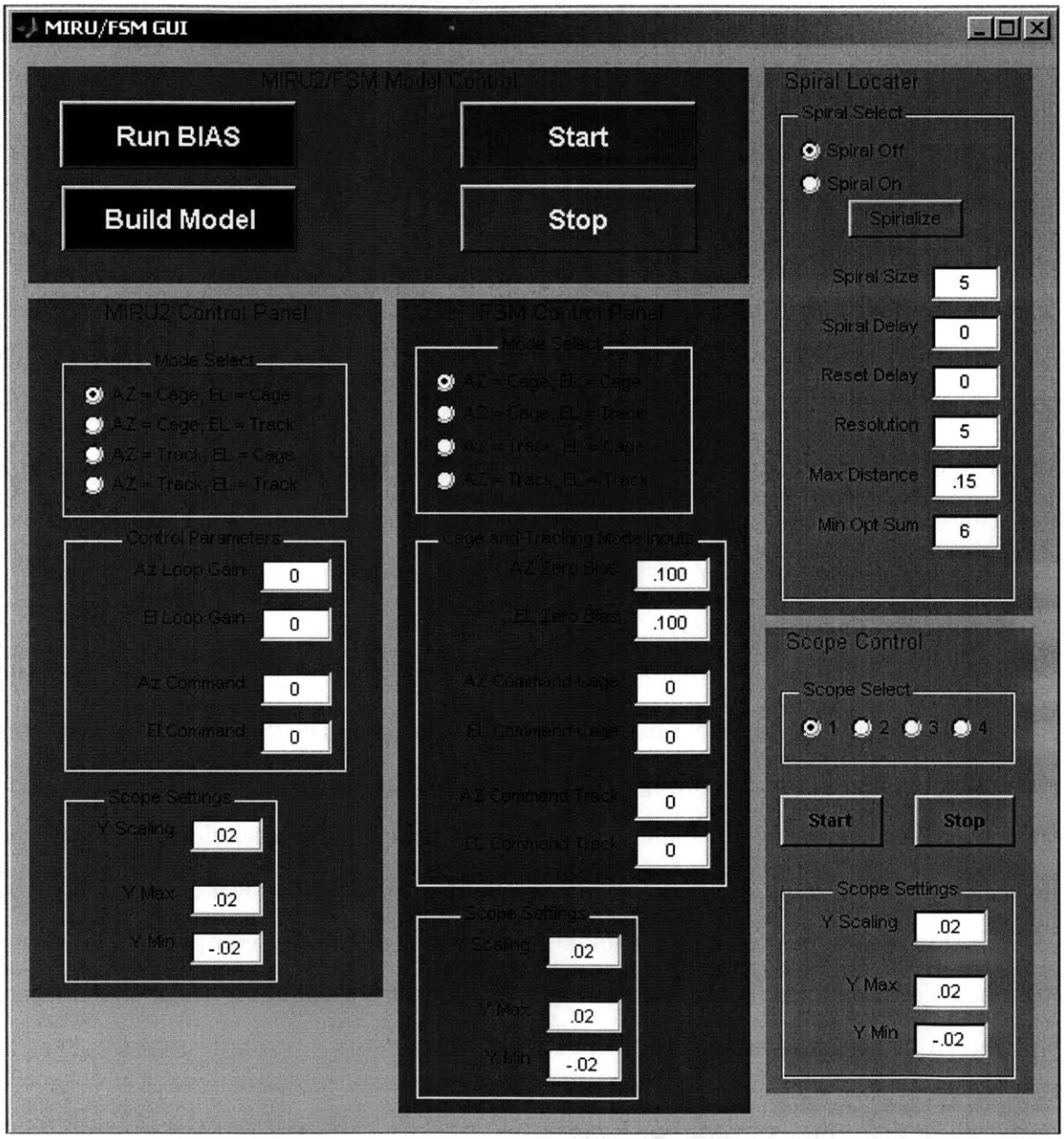

Figure 4-7: MATLAB graphical user interface used to control square spiral algorithm.

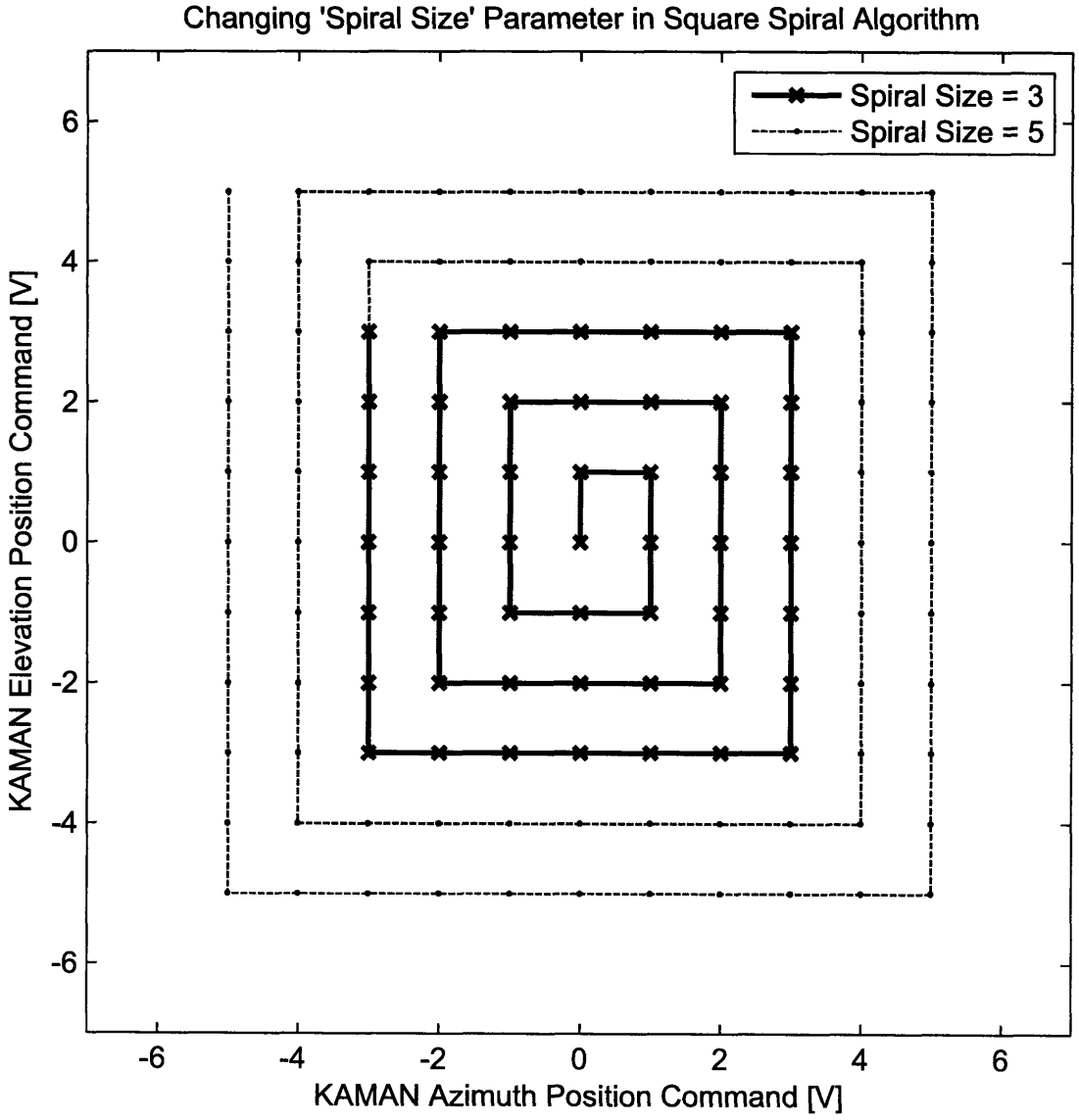

Figure 4-8: X-Y plot of steering path showing the effect of changing the 'Spiral Size' value in the square spiral algorithm.

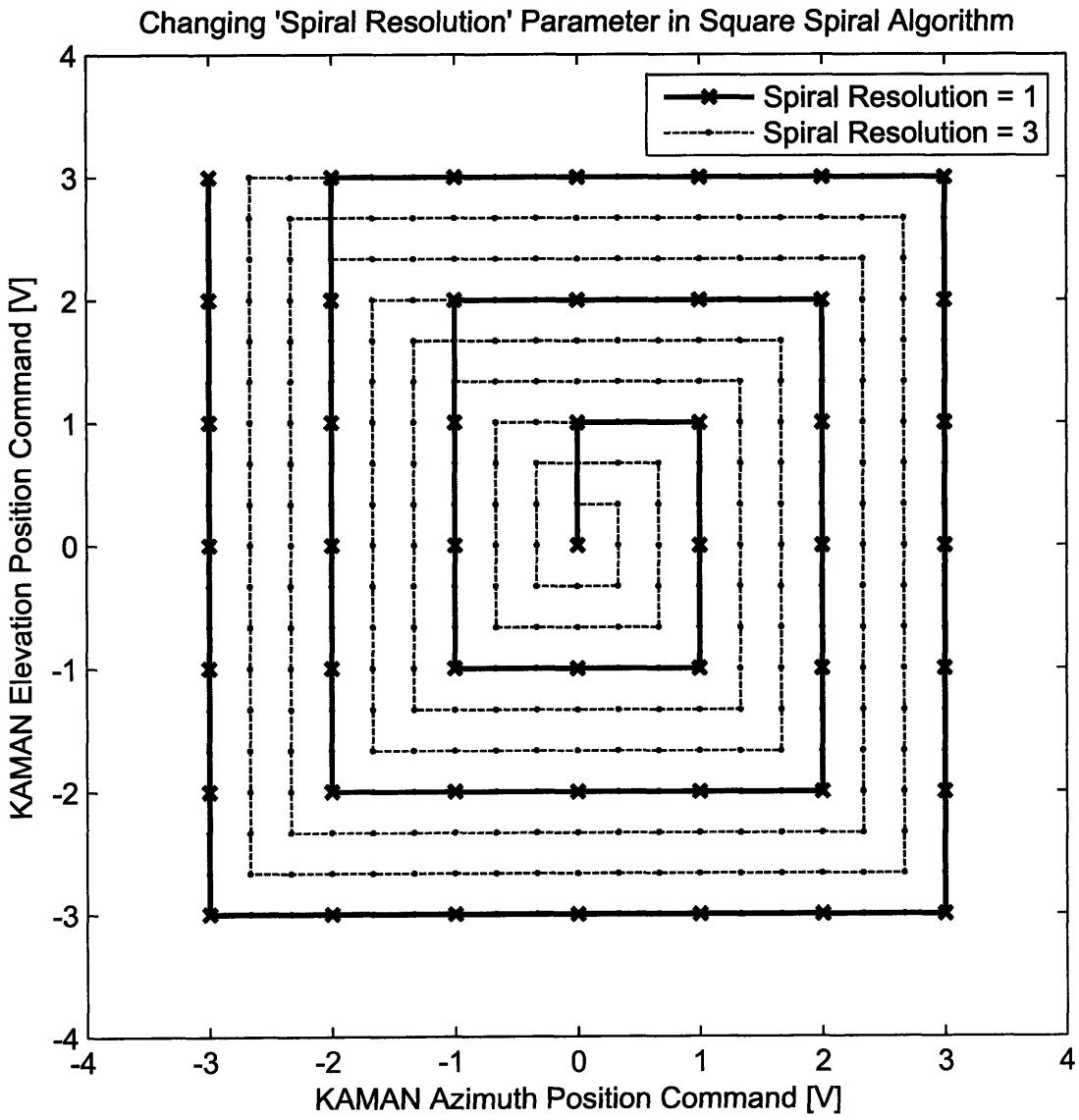

Figure 4-9: X-Y plot of steering path showing the effect of changing the 'Spiral Resolution' value in the square spiral algorithm.

the pattern, with one axis stopping completely and the other starting instantly. This can lead to "rounding" at the corners when the FSM cannot accelerate fast enough with the commands. For this reason, a true spiral algorithm was chosen to eliminate these sudden accelerations.

4.3.2 Constant Angular Velocity Spiral

Avoiding the sudden accelerations imparted on the FSM when using the square spiral algorithm is easily avoidable by using a steering pattern with no straight lines and no corners. The easiest way to implement a steering pattern with those specifications is to use a constant angular velocity (CAV) spiral. This type of spiral is best thought of in polar coordinates. Each step in the pattern of a constant angular velocity spiral will have the same change in angle and the same increase in radius. Thus, implementing the algorithm is as easy as calculating these two values and converting back into cartesian commands.

CAV Spiral Algorithm

The Stateflow representation of this algorithm is shown in Figure 4-10. Stateflow input, output and local variables used in Figure 4-10 are listed and described in Tables 4.4, 4.5 and 4.6 respectively.

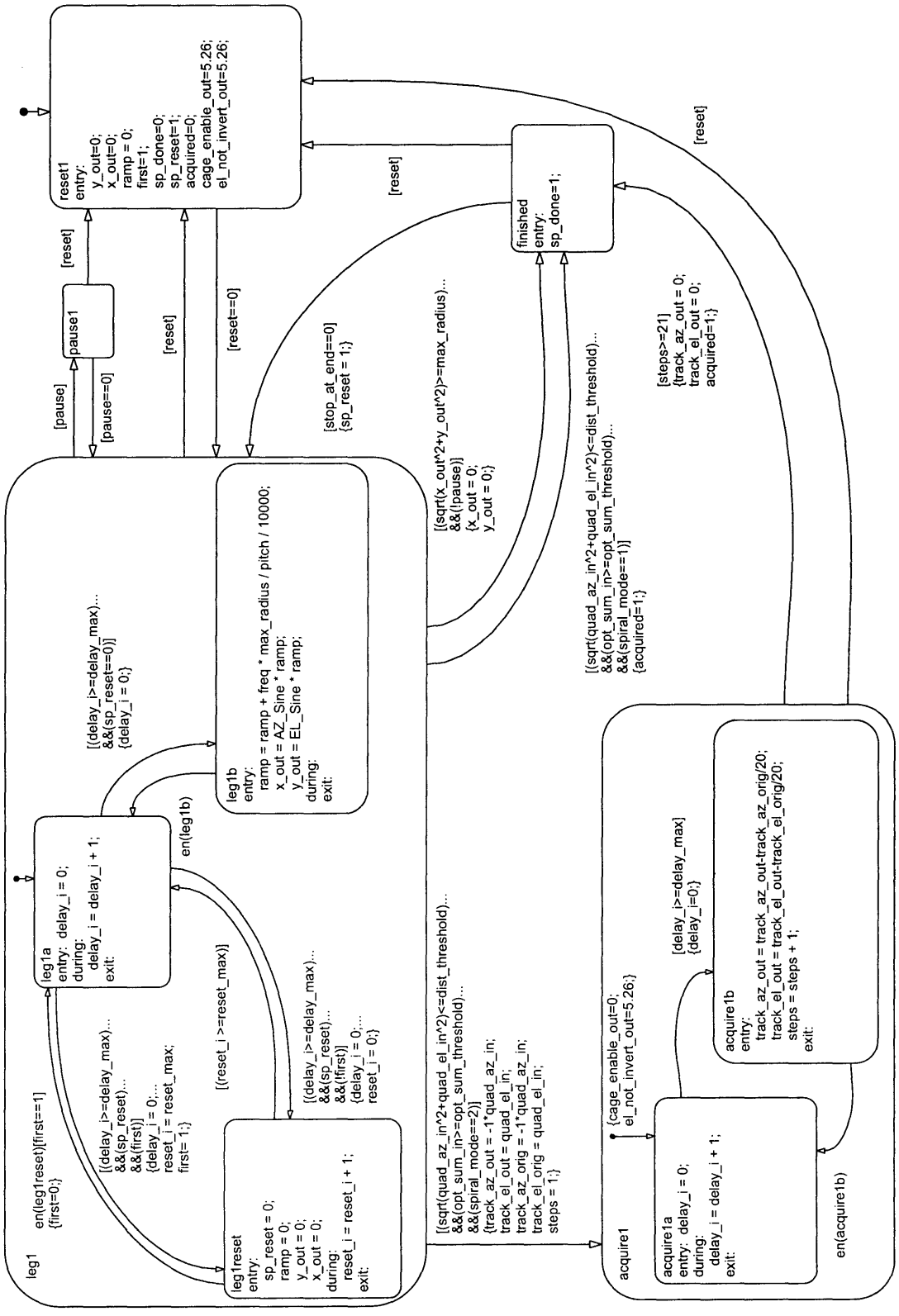

Figure 4-10: Stateflow diagram of the constant angular velocity (CAV) spiral algorithm.

Name	Description
pause	When set to 1, pauses the spiral algorithm. Resumes when set to 0.
max_radius	Maximum radius of the commanded spiral.
freq	Frequency of spiral rotation.
reset_max	Time (in cycles) to wait before restarting algorithm.
pitch	Number of complete revolution per full spiral.
reset	Resets algorithm when 1. Restarts when switched back to 0.
stop_at_end	When 1, will not repeat spiral if acquisition did not occur. When 0, repeats until acquisition occurs.
quad_az_in	Azimuth input from the quad cell. Used to determine distance from center of quad cell.
quad_el_in	Elevation input from the quad cell. Used to determine distance from center of quad cell.
opt_sum_in	Optical Sum input from the quad cell. Used to determine if on active region quad cell.
dist_threshold	Sets how close the beam must be to the origin of the quad cell before entering acquisition mode. If $dist\_threshold \geq \sqrt{quad\_az\_in^2 + quad\_el\_in^2}$, algorithm switches to acquisition mode (provided that $opt\_sum\_threshold \leq opt\_sum\_in$).
opt_sum_threshold	Sets how high the Optical Sum input from the quad cell must be before entering acquisition mode. If $opt\_sum\_threshold \leq opt\_sum\_in$, algorithm switches to acquisition mode (provided that $dist\_threshold \geq \sqrt{quad\_az\_in^2 + quad\_el\_in^2}$).
AZ_Sine	Sine wave input. $AZ\_Sine = \sin(2\pi freq \cdot t)$, $x\_out = ramp \cdot AZ\_Sine$.
EL_Sine	Sine wave input. $EL\_Sine = \sin(2\pi freq \cdot t - \pi/2)$, $y\_out = ramp \cdot EL\_Sine$.
spiral_mode	When 1, finds quad cell but does not acquire. When 2, finds quad cell and enters acquisition mode.

Table 4.4: Table of Stateflow input variables used in the CAV spiral algorithm.

Name	Description
x_out	Command to KAMAN Azimuth axis.
y_out	Command to KAMAN Elevation axis.
track_az_out	Command to quad cell Azimuth axis.
track_el_out	Command to quad cell Elevation axis.
cage_enable_out	Digital signal controlling feedback mode (switching between KAMAN and quad cell) of FSM when using the original analog controller.
el_not_invert	Digital signal controlling feedback mode (switching between KAMAN and quad cell) of FSM when using the original analog controller.
acquired	Is 1 if the algorithm found the quad cell and entered acquisition mode, 0 otherwise.
sp_done	Is 1 if algorithm is in *finished* state, 0 otherwise.

Table 4.5: Table of Stateflow output variables used in the CAV spiral algorithm.

Name	Description
delay_i	Counter for *delay_max*.
delay_max	Time (in cycles) before evaluating to next state. Usually set to zero.
first	Is 1 if coming out of *reset1* state, otherwise 0. Used to avoid waiting *reset_delay* before starting spiral.
ramp	Stores current value of command radius from center. $x\_out = ramp \cdot AZ\_Sine$ and $y\_out = ramp \cdot EL\_Sine$.
reset_i	Counter for *reset_max*.
sp_reset	Set to 1 when algorithm is to wait *reset_delay* during first state. Is set to 0 only in *leg1reset*.
steps	Counter used for bumpless transfer.
track_az_orig	Stores original quad cell Azimuth command used when switching to quad cell tracking mode. Used for bumpless transfer.
track_az_orig	Stores original quad cell Elevation command used when switching to quad cell tracking mode. Used for bumpless transfer.

Table 4.6: Table of Stateflow local variables used in the CAV spiral algorithm.

The Stateflow diagram shown in Figure 4-10 has only 1 state dedicated to the steering of the beam, *leg1*, compared to the 4 used in the square spiral algorithm. The implementation of the algorithm is equally less complex. As stated before, the next point in the CAV spiral pattern is always just a fixed increase in angle combined with a fixed increase in the radius. Fortunately, the change in angle is accomplished automatically by the *AZ_Sine* and *EL_Sine* parameters which are time based sine wave signals from Simulink, and do not need to be implemented in Stateflow. The only parameter the Stateflow has to change is the current radius of the spiral, represented by the *ramp* parameter. The change in radius is calculated as a function of *freq*,

pitch, *max_radius* and the sampling frequency of the model F (usually 10 kHz).

$$\Delta ramp \cdot \frac{F \cdot pitch}{freq} = max\_radius$$
$$\Delta ramp = \frac{max\_radius \cdot freq}{F \cdot pitch} \tag{4.2}$$

The first equation states that change in *ramp* times the number of points in the entire spiral algorithm equals the final radius of the spiral. Simply rearranging this equation yields $\Delta ramp$ as a function of input parameters. *leg1b* handles the incrementing of the *ramp* parameter per Equation 4.2 (with $F = 10^4$). This ramp is multiplied by the *AZ_Sine* and *EL_Sine* parameters to convert the polar representation of the spiral into the cartesian commands *x_out* and *y_out*. The state changes out of *leg1* only when the quad cell is acquired (see Section 4.4) or the radius of the spiral reaches *max_radius*. Finally, note that the behavior of the model relating to the rest of the parameters (such as *stop_at_end* and *reset_max*) is precisely as described in Section 4.3.1. Figure 4-11 shows the outputted spiral path with $pitch = 5$, $max\_radius = 10$ and $freq = 250$.

xPC Implementation

The xPC integration for the algorithm is the same the other two algorithms, and is shown in Figure 4-5. Figure 4-12 shows the Simulink block diagram used with the CAV algorithm and Figure 4-13 shows the MATLAB GUI used to control the CAV algorithm.

Figure 4-12 is almost identical the Block Diagram of the square spiral algorithm shown in Figure 4-6. The new parameter *spiral_mode* allows changing between two modes of the algorithm. When $spiral\_mode = 1$, the algorithm will find the quad cell but not switch into the *acquire1* state and enter quad cell feedback mode. When $spiral\_mode = 2$, the algorithm will find the quad cell and switch into the *acquire1* to complete acquisition of the quad cell and enter quad cell feedback mode. Also, the *AZ_Sine* and *EL_Sine* blocks provide two sinusoidal signals for use in the Stateflow block. The frequency of both sine waves is controlled by *freq*.

The code ran when the 'Spiralize' button is pressed on the GUI of Figure 4-13 is

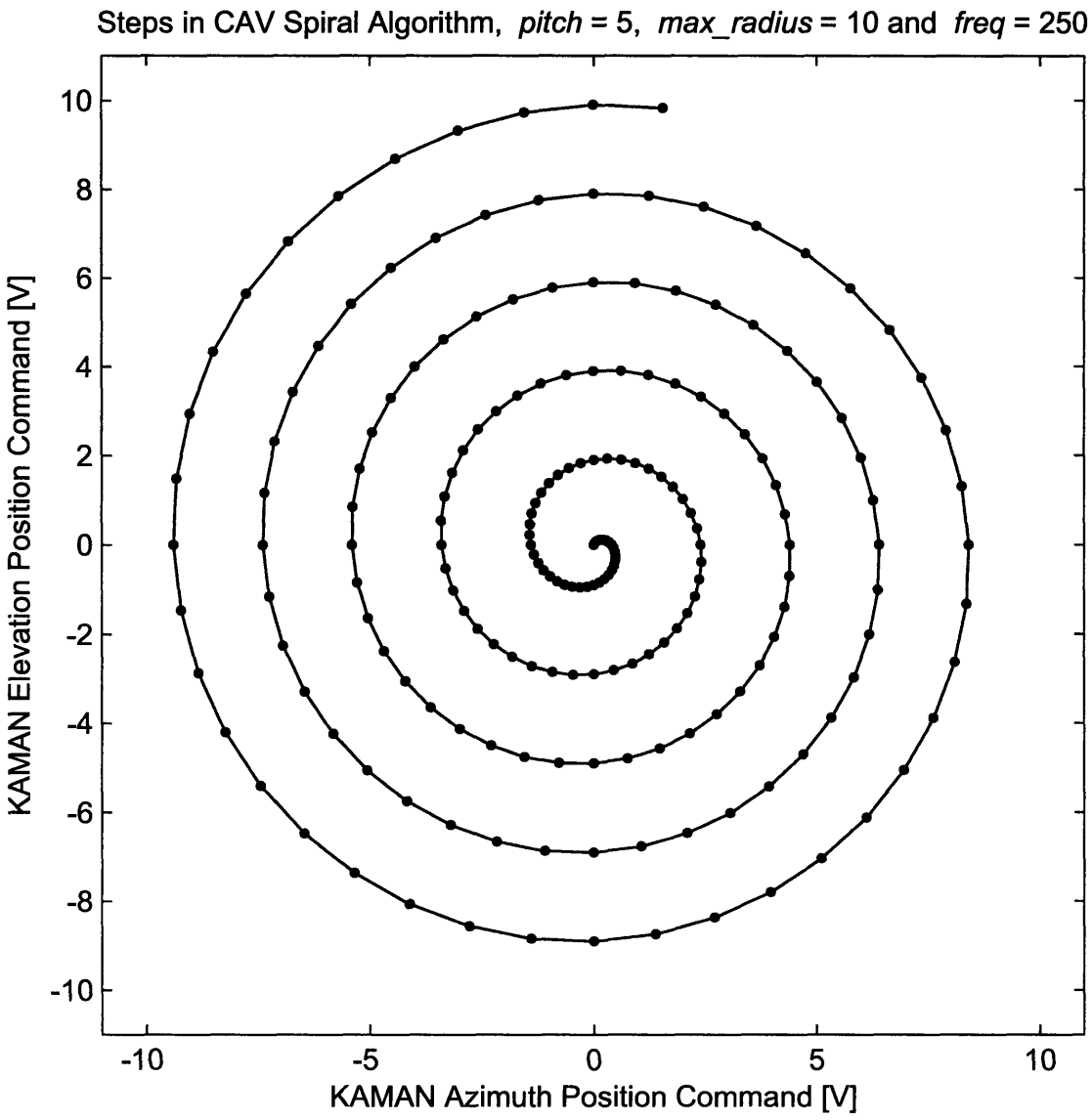

Figure 4-11: X-Y plot of the steering path of the CAV spiral algorithm with $pitch = 5$, $max\_radius = 10$ and $freq = 250$.

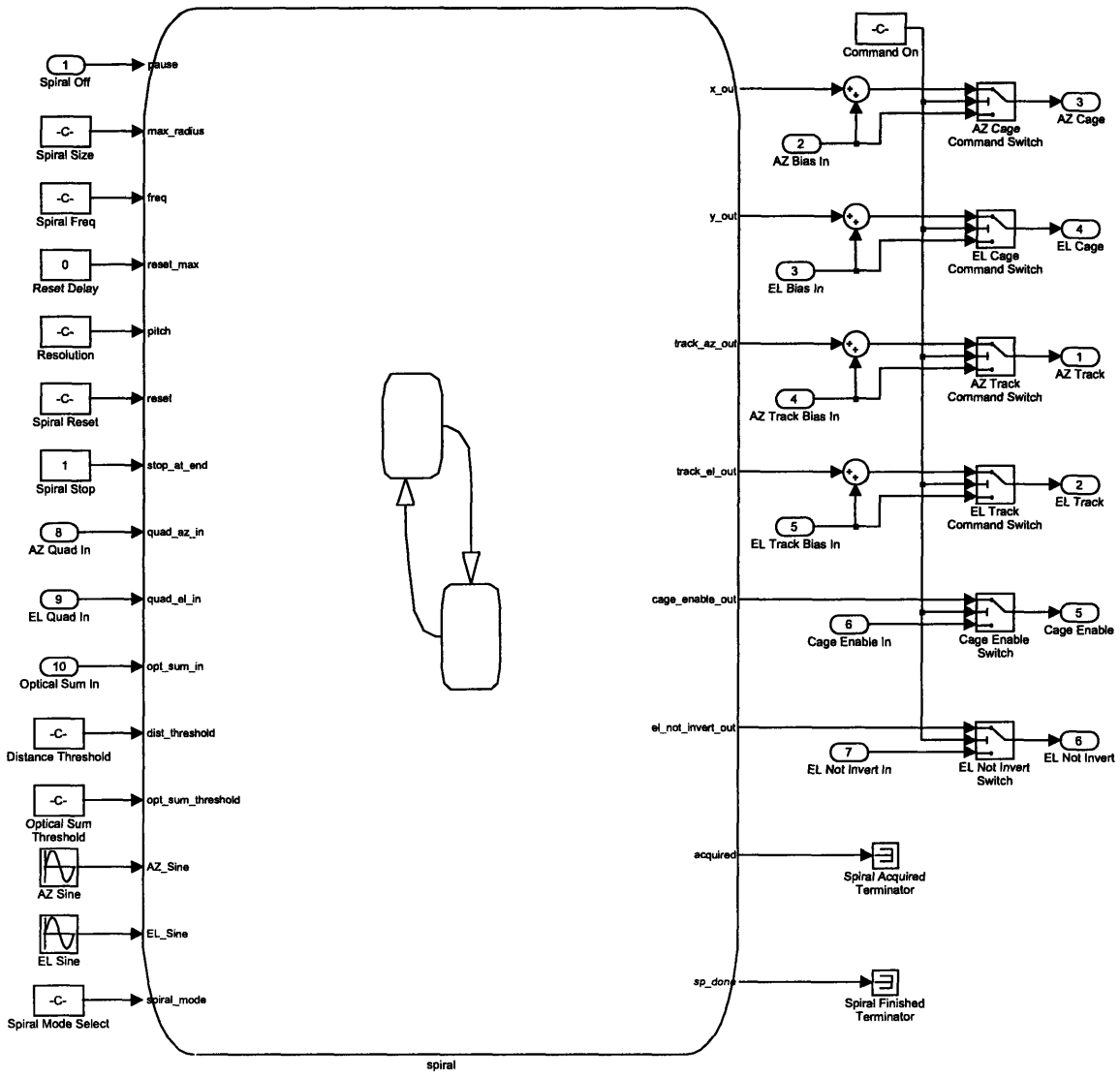

Figure 4-12: Second level Simulink model of the CAV spiral algorithm.

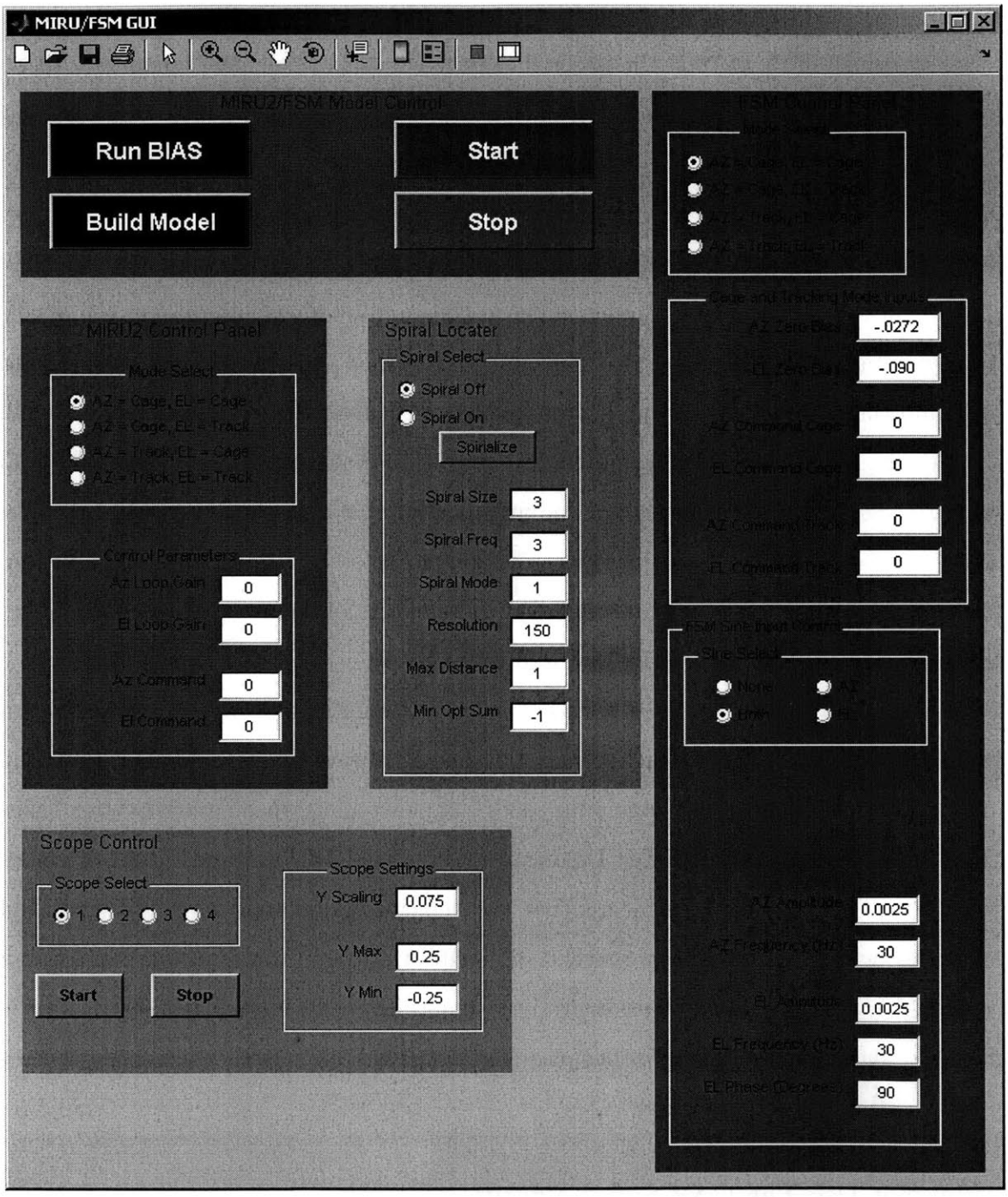

Figure 4-13: MATLAB graphical user interface used to control the CAV spiral algorithm.

slightly different from the code from the square spiral algorithm in Appendix G. This new code is available in Appendix H. The methods are similar, except that the code accounts for the fact that the algorithm may not go into acquisition mode. If it does not (*spiral_mode* = 1), but the algorithm finds the quad cell, the current position commands are saved and used as the new starting point for another spiral. This time, the spiral is reduced in magnitude by a factor of 10 (making it 10 times more dense) and the *dist_threshold* is reduced by a factor of 10. A new spiral is run to get closer to the center of the quad cell. This second spiral is necessary because the grid spacing of the CAV spiral get much wider as the spiral diameter increases. Using this implementation, a coarse spiral is run first just to find the general area of the quad cell, and then a fine spiral is run automatically to get near the center of the quad cell.

Finally, there are three parameters in Figure 4-13 that control the outputted spiral pattern. 'Spiral Size' controls the maximum allowable radius of the spiral, 'Spiral Freq' controls how many points per rotation are plotted, and 'Spiral Resolution' controls how many rotations are included. The effects of changing these parameters are shown in Figures 4-14, 4-15 and 4-16 respectively.

Figure 4-15 is not exactly what one expects. Theoretically, the two spirals shown should be lying on top of each other, but one is always slightly smaller than the other. This is because the algorithm does not start in *leg1b* at startup, but takes 2 state transitions to get there. During this time, the *AZ_Sine* and *EL_Sine* values are already initialized. This means that the departure angle will be slightly rotated clockwise when *leg1b* is finally reached. The effect is more pronounced when using a very high $freq$ value, and that's why the two spirals do not match perfectly. This is not an issue during acquisition because the departure angle from the center doesn't have an effect on the grid itself.

Why Not Use The CAV Spiral algorithm?

The CAV Spiral algorithm solves the problem or accelerating around sharp corners, but introduces a few new problems as well. First, as the spiral diameter increases, the grid points get wider and wider apart (meaning the FSM is moving faster and

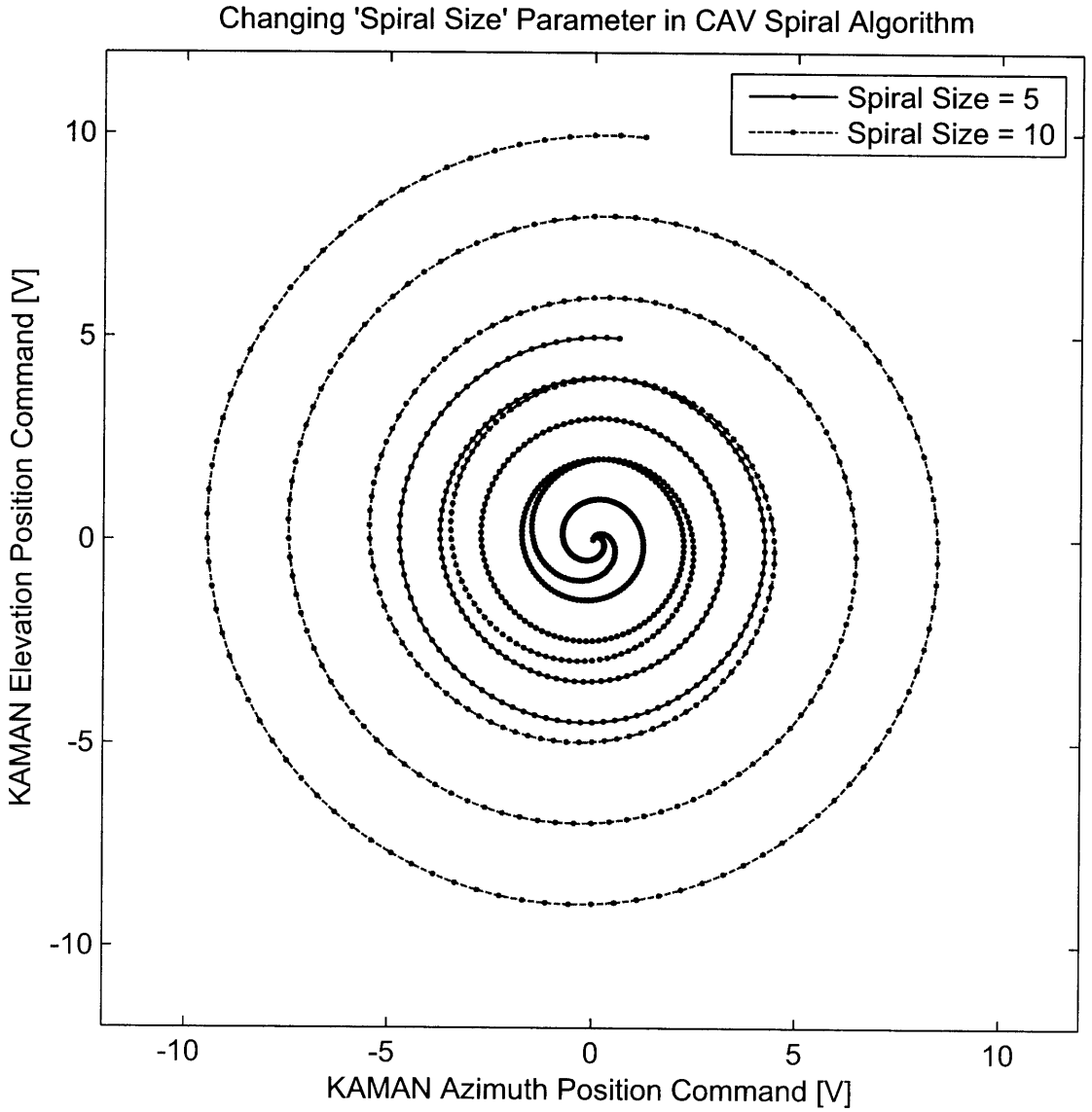

Figure 4-14: X-Y plot of the steering path showing the effect of changing the 'Spiral Size' value in the CAV spiral algorithm.

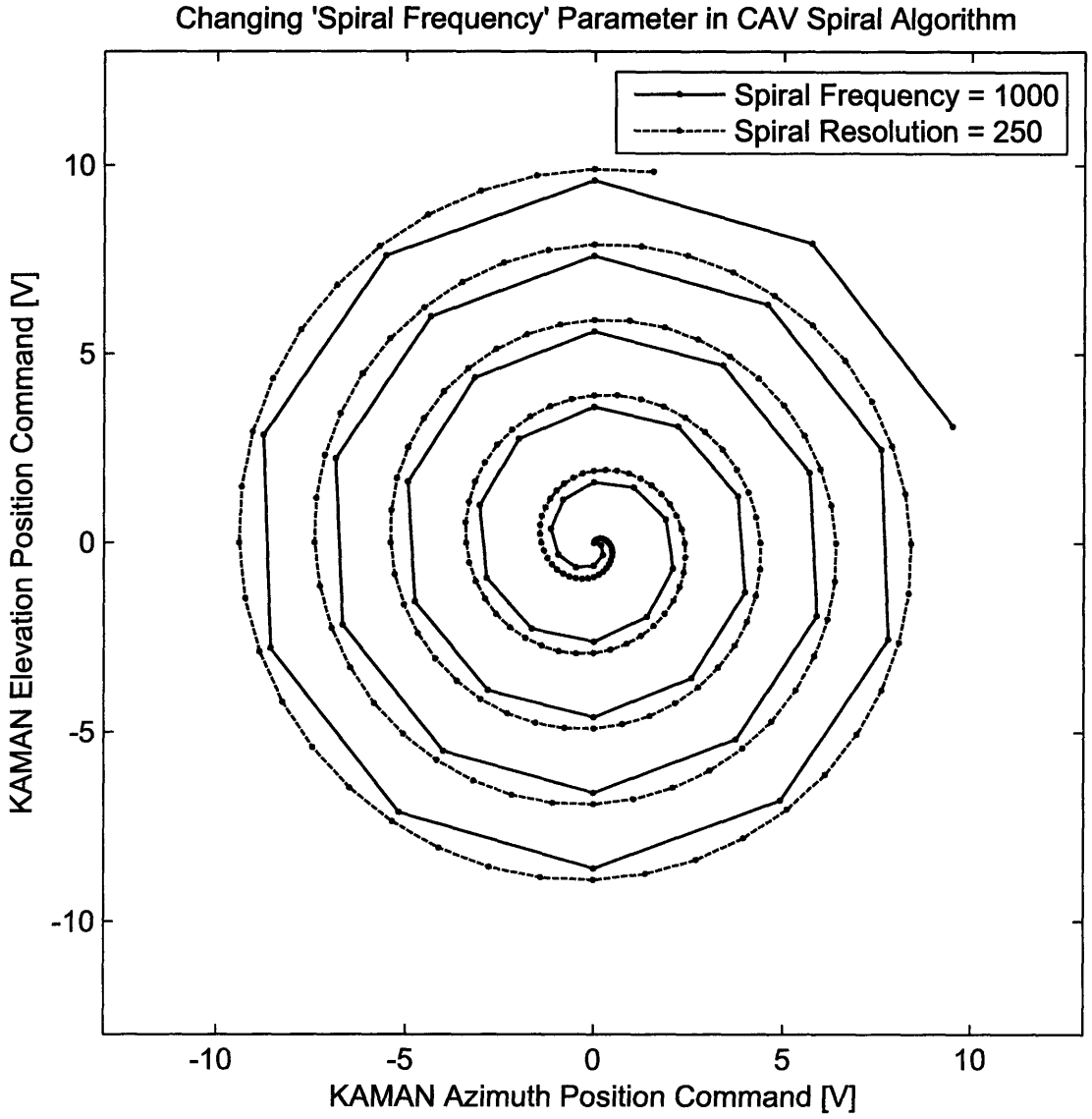

Figure 4-15: X-Y plot of the steering path showing the effect of changing the 'Spiral Frequency' value in the CAV spiral algorithm.

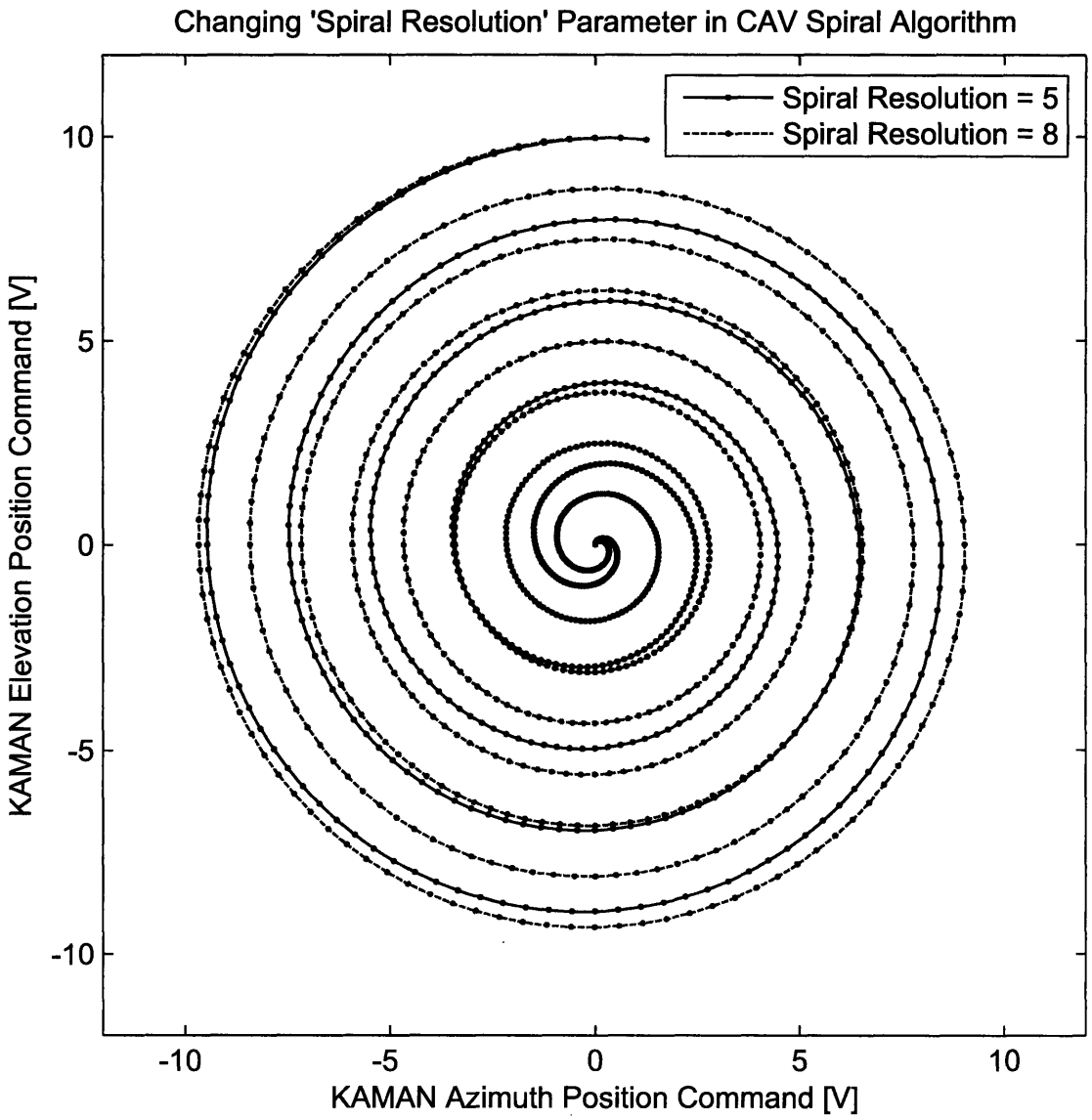

Figure 4-16: X-Y plot of the steering path showing the effect of changing the 'Spiral Resolution' value in the CAV spiral algorithm.

faster and undergoing harsher accelerations). In order to guarantee that the quad cell will be found, the $freq$ value must be low enough so that the outer spiral points are close enough to not miss the quad cell. This means that a lot of redundant points are wasted in the inner part of the spiral, wasting time and being very inefficient. The optimum solution would be to combine the path of the CAV spiral with the fixed step length and uniform grid pattern of the square spiral. Fortunately, the constant linear velocity spiral does just that.

4.3.3 Constant Linear Velocity Spiral

The constant linear velocity (CLV) spiral combines the best features from the square spiral and CAV spiral. The CLV algorithm has a smooth beam path like the CAV Spiral combined with the grid uniformity of the square spiral.

Constant Linear Velocity Spiral Algorithm

Figure 4-17 shows the Stateflow implementation of the CLV Spiral algorithm. Stateflow input, output and local variables used in Figure 4-17 are listed and described in Tables 4.7, 4.8 and 4.9 respectively.

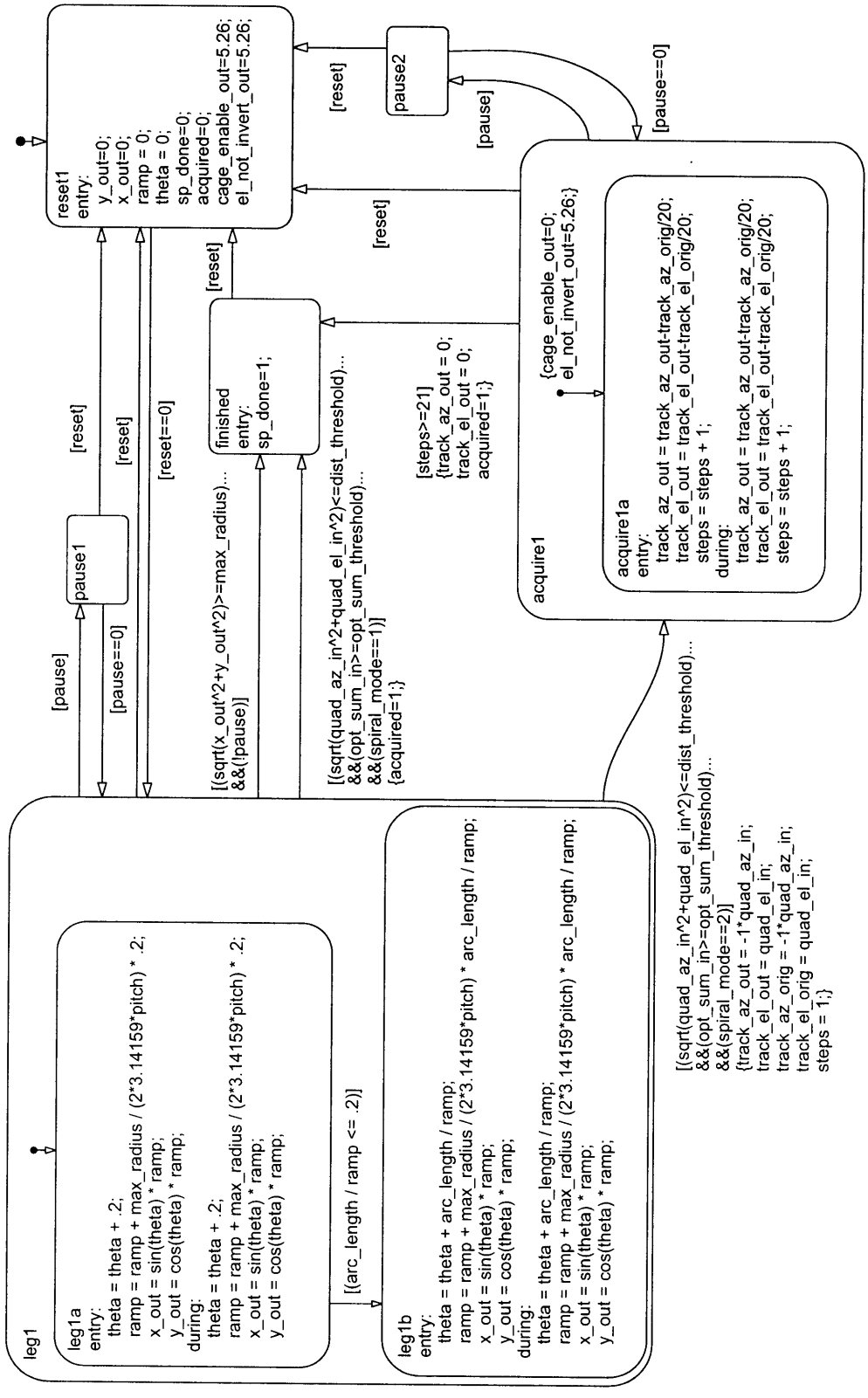

Figure 4-17: Stateflow diagram of the constant linear velocity (CLV) spiral algorithm.

Name	Description
pause	When set to 1, pauses the spiral algorithm. Resumes when set to 0.
max_radius	Maximum radius of the commanded spiral.
arc_length	Maximum step size between grid point.
pitch	Number of complete revolutions per full spiral.
reset	Resets algorithm when 1. Restarts when switched back to 0.
stop_at_end	When 1, will not repeat spiral if acquisition did not occur. When 0, repeats until acquisition occurs.
quad_az_in	Azimuth input from the quad cell. Used to determine distance from center of quad cell.
quad_el_in	Elevation input from the quad cell. Used to determine distance from center of quad cell.
opt_sum_in	Optical Sum input from the quad cell. Used to determine if on active region quad cell.
dist_threshold	Sets how close the beam must be to the origin of the quad cell before entering acquisition mode. If $dist\_threshold \geq \sqrt{quad\_az\_in^2 + quad\_el\_in^2}$, algorithm switches to acquisition mode (provided that $opt\_sum\_threshold \leq opt\_sum\_in$).
opt_sum_threshold	Sets how high the Optical Sum input from the quad cell must be before entering acquisition mode. If $opt\_sum\_threshold \leq opt\_sum\_in$, algorithm switches to acquisition mode (provided that $dist\_threshold \geq \sqrt{quad\_az\_in^2 + quad\_el\_in^2}$).
spiral_mode	When 1, finds quad cell but does not acquire. When 2, finds quad cell and enters Acquisition mode.

Table 4.7: Table of Stateflow input variables used in the CLV spiral algorithm.

Name	Description
x_out	Command to KAMAN Azimuth axis.
y_out	Command to KAMAN Elevation axis.
track_az_out	Command to quad cell Azimuth axis.
track_el_out	Command to quad cell Elevation axis.
cage_enable_out	Digital signal controlling feedback mode (switching between KAMAN and quad cell) of FSM using original analog controller.
el_not_invert	Digital signal controlling feedback mode (switching between KAMAN and quad cell) of FSM using original analog controller.
acquired	Is 1 if the algorithm found the quad cell and entered acquisition mode, 0 otherwise.
sp_done	Is 1 if algorithm is in *finished* state, 0 otherwise.

Table 4.8: Table of Stateflow output variables used in the CLV spiral algorithm.

Name	Description
ramp	Stores current value of command radius from center V.
steps	Counter used for bumpless transfer.
theta	Stores current spiral angle [rad].
track_az_orig	Stores original quad cell Azimuth command used when switching to quad cell tracking mode. Used for bumpless transfer.
track_az_orig	Stores original quad cell Elevation command used when switching to quad cell tracking mode. Used for bumpless transfer.

Table 4.9: Table of Stateflow local variables used in the CLV spiral algorithm.

The Stateflow code generating the steering commands is contained entirely in *leg1*. The main parameter of interest is *arc_length*. This parameter sets the step size of the individual spiral points. With the step size being constant, instead of the step angle as with the CAV Spiral, each subsequent spiral has more points than the previous one, creating a spiral that is formed with constant linear velocity.

In the beginning of the algorithm, the *arc_length* is too large to approximate a

spiral shape. To avoid this problem, *leg1a* starts the algorithm with a CAV spiral until

$$\frac{ramp \cdot \pi}{20} \geq arc\_length \tag{4.3}$$

meaning until there would be 40 points per revolution of the spiral using *arc_length* as the step size. Once this is true, the steering pattern continues with constant linear velocity. For each new step, the algorithm must calculate the change in angle and the change is radius. The diagram in Figure 4-18 shows how these two values are approximated using *arc_length* as a guide (a in Figure 4-18).

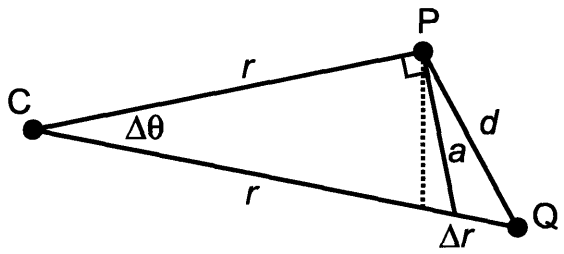

Figure 4-18: Diagram showing step progression of CLV spiral algorithm from point P to point Q.

The change in angle $\Delta\theta$ is calculated using the small angle approximation.

$$\Delta\theta = \tan\left(\frac{a}{r}\right) \approx \frac{a}{r} \tag{4.4}$$

Note that this is just an approximation to the correct change in angle. However, when small angles are used, the error is negligible. Next, the change in radius is calculated as a function of the change in angle:

$$\Delta r = \frac{max\_radius \cdot \Delta\theta}{2 \cdot pitch} \tag{4.5}$$

Next, the polar coordinate representation calculated is transformed into *x_out* and *y_out* commands. The algorithm continues this way until the *max_radius* is reached or the algorithm enters acquisition mode. Figure 4-19 shows an outputted spiral,

illustrating the switch between CAV and CLV mode. Normally, the CAV mode would not last more than a few revolutions.

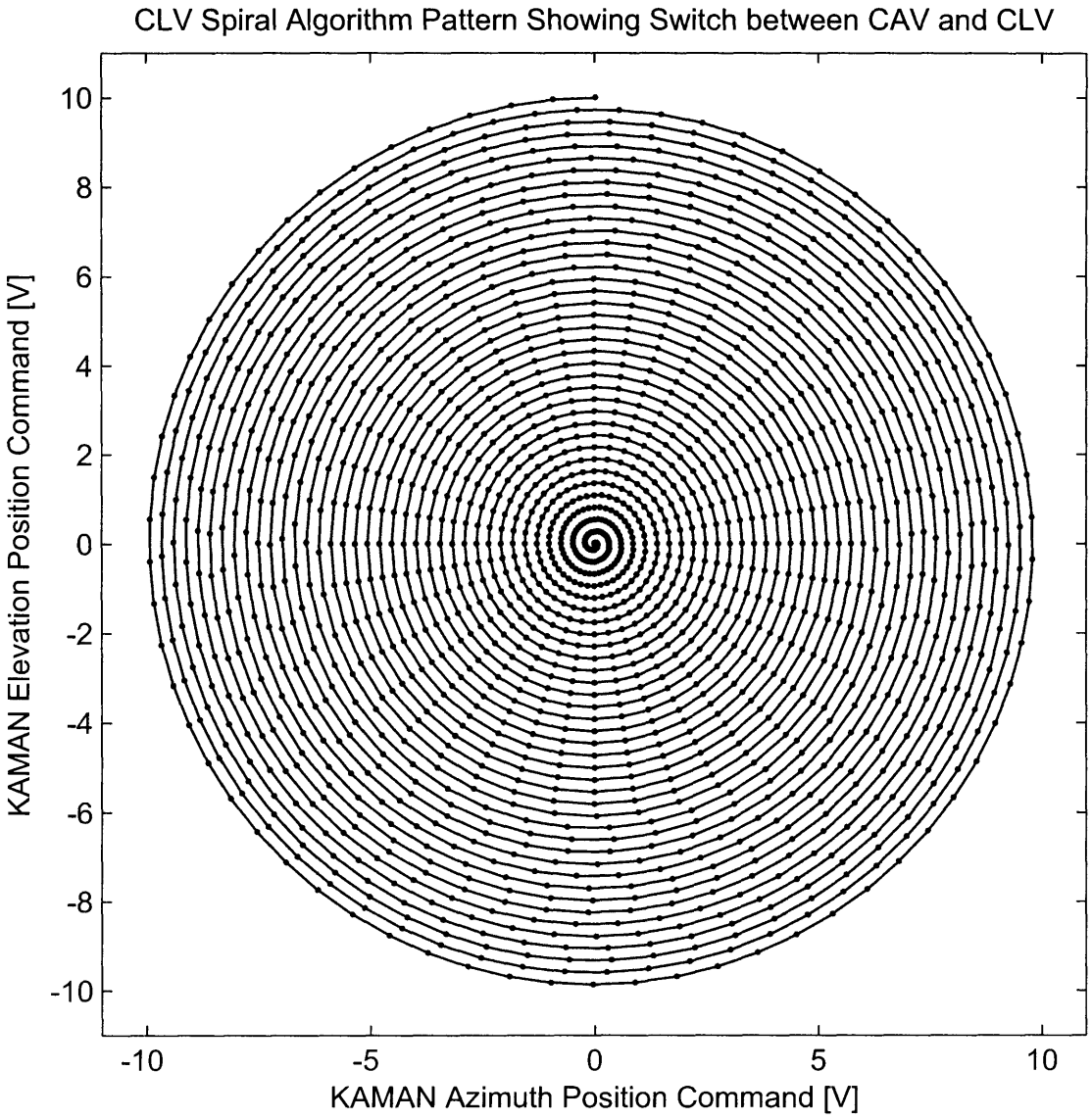

Figure 4-19: X-Y plot of the steering path using the CLV spiral algorithm, showing the transition from CAV to CLV.

Figure 4-20 shows a comparison between the CLV algorithm and a CAV spiral calculated in MATLAB to highlight any errors.

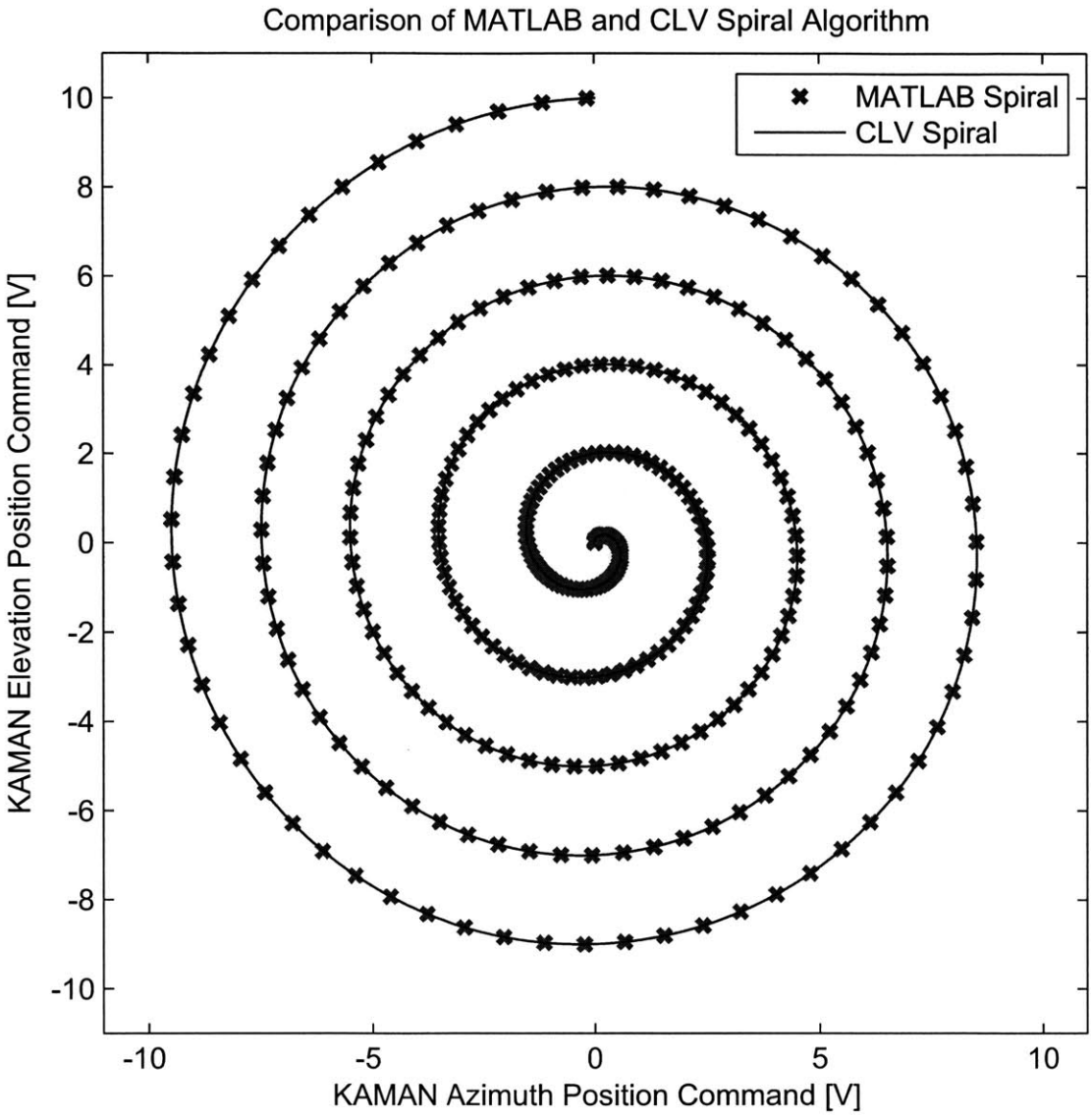

Figure 4-20: X-Y plot of the steering path using the CLV spiral algorithm compared with a MATLAB generated CAV spiral.

xPC Implementation

The xPC integration for the algorithm is the same the other two algorithms, and is shown in Figure 4-5. Figure 4-21 shows the Simulink block diagram used with the CLV algorithm and Figure 4-22 shows the MATLAB GUI used to control the CLV algorithm.

Figure 4-21 is basically the same as the CAV Simulink block diagram shown in Figure 4-12. The notable differences include the inclusion of the *arc_length* parameter and the removal of unnecessary parameters. The code used in the GUI to call 'Spiralize' is the exact same and included in Appendix H. The user changeable parameters that affect the CLV Spiral shape are the 'Spiral Size', the 'Spiral Arc Length', and the 'Spiral Resolution' parameters. Figures 4-23, 4-24 and 4-25 show the effects of changing each of these parameters, respectively.

Finally, it is important to note that the 'Spiral Size', 'Spiral Arc Length', and 'Spiral Resolution' should be set to a specific ratio for optimal coverage. To approximate a square grid with a spiral, the change between successive spirals should be the same as the space between points on the spiral curve. This means that using

$$pitch = \frac{max\_radius}{arc\_length} \tag{4.6}$$

will produce optimal results. This optimal pattern is shown in Figure 4-26.

Imagine that the active area of the quad cell was centered between the 4 points in Figure 4-26. The rightmost grid points do not sit on the intersections of the dotted lines representing a CAV grid the points are formed with the CLV spiral. The spacing shown is a 'worst case scenario'; when the points visited line up in this fashion, the effective coverage of each point is at a minimum. In order to acquire the quad cell, the *dist_threshold* would have to be approximately $arc\_length/\sqrt{2}$. Thus obeying

$$dist\_threshold \geq \frac{\sqrt{2}}{2} arc\_length \tag{4.7}$$

guarantees that the grid pattern will not miss the quad cell.

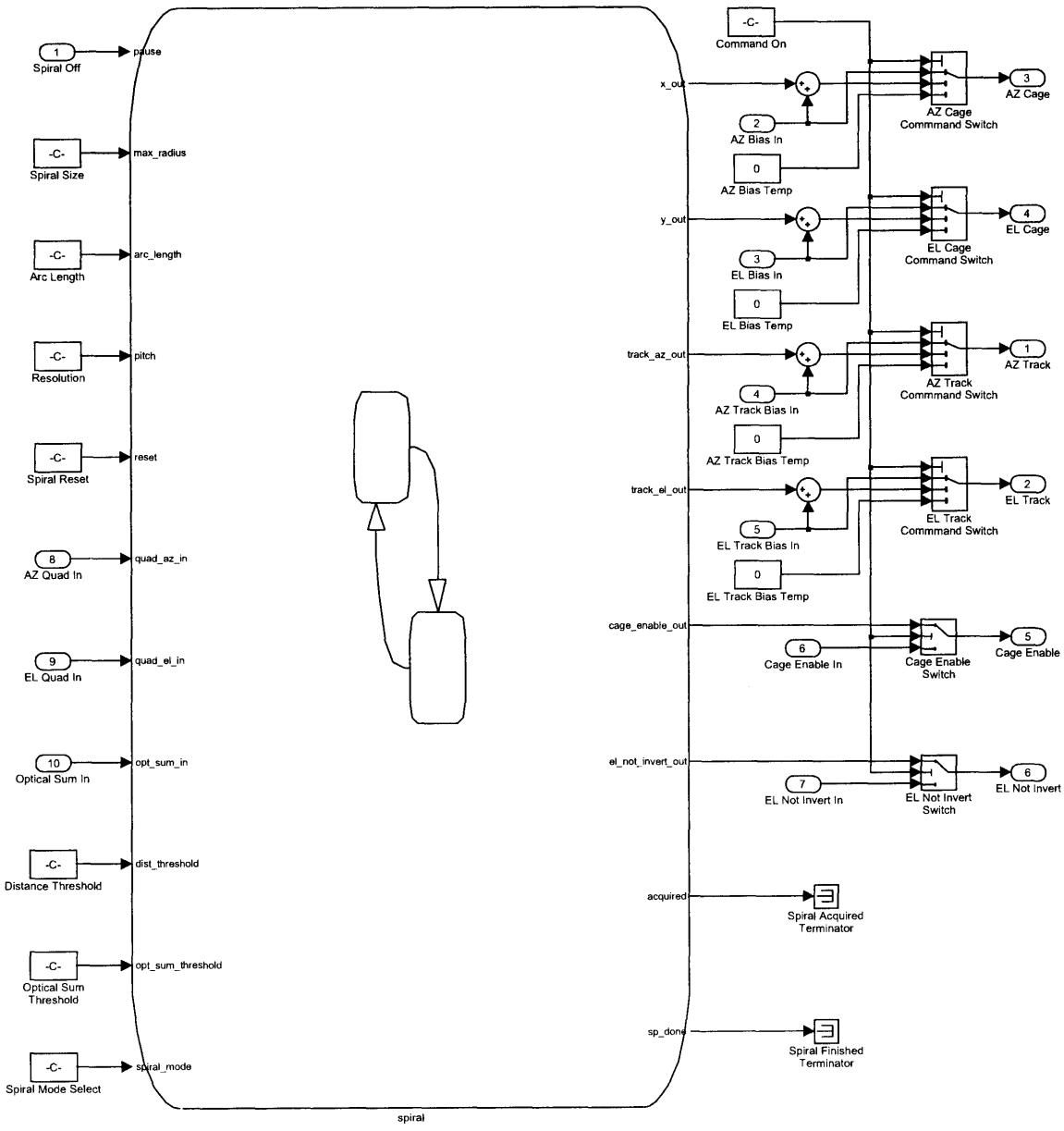

Figure 4-21: Second level Simulink model of the CLV spiral algorithm.

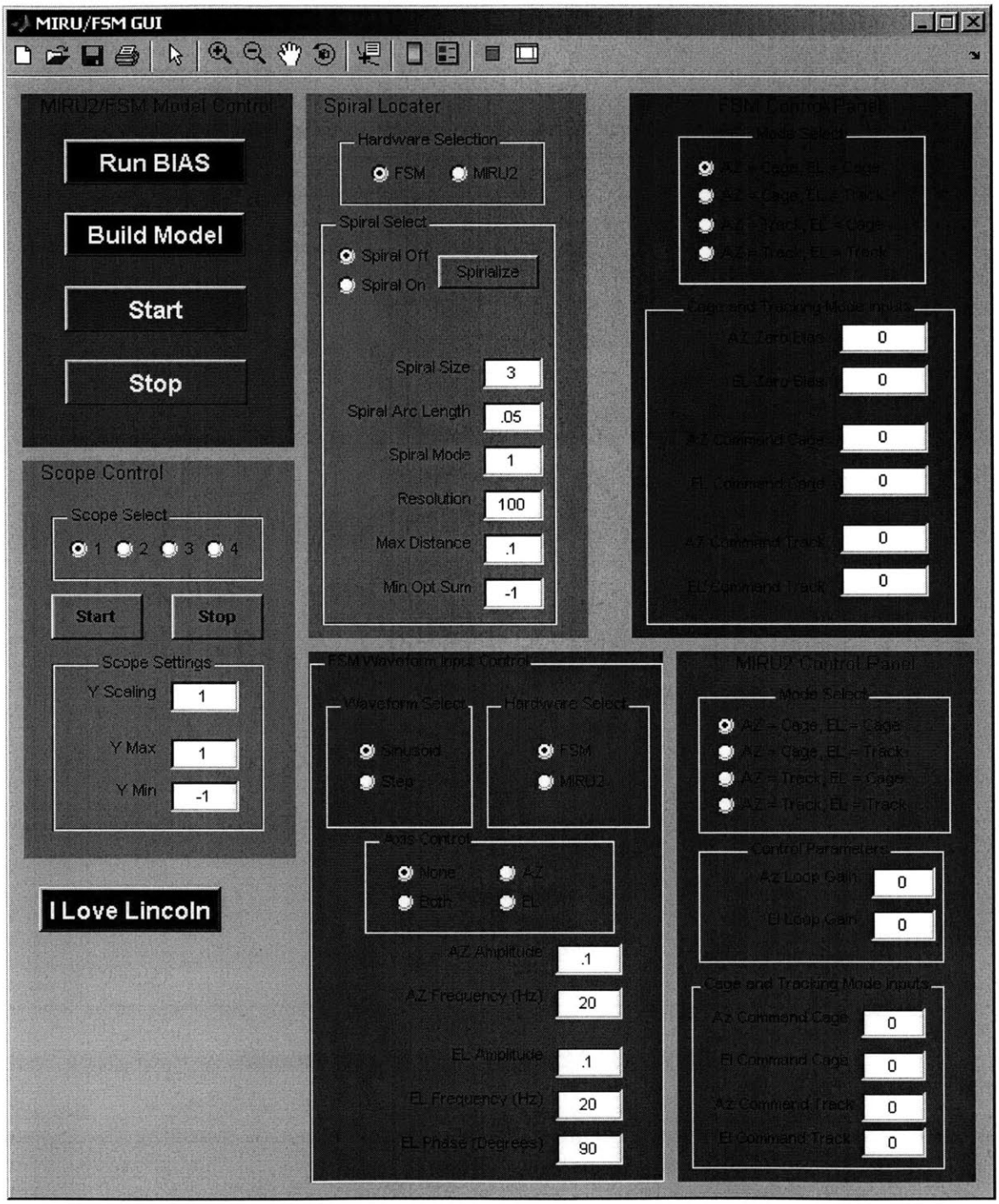

Figure 4-22: MATLAB graphical user interface used to control the CLV spiral algorithm.

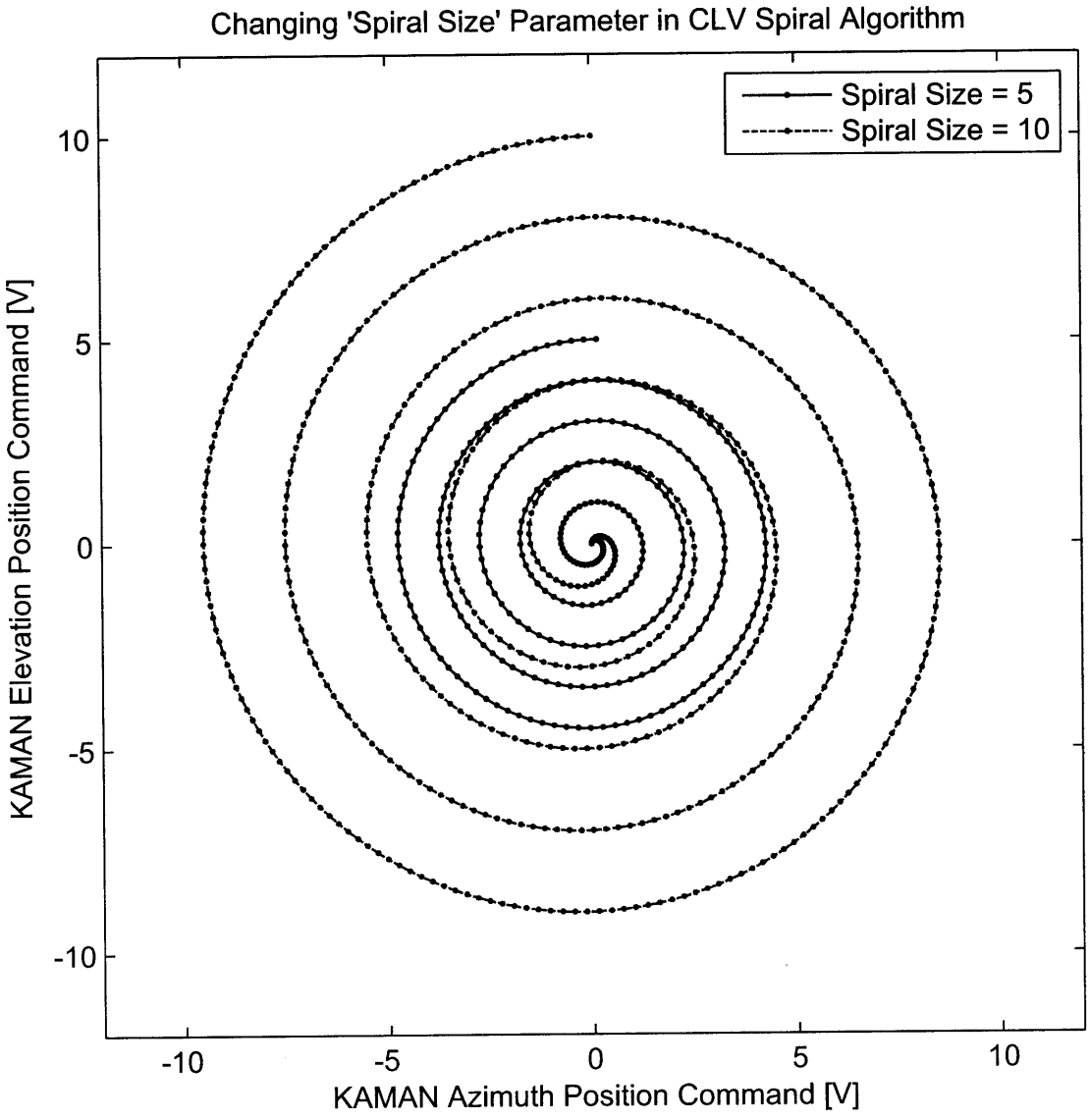

Figure 4-23: X-Y plot of the steering path showing the effect of changing the 'Spiral Size' value in the CLV spiral algorithm.

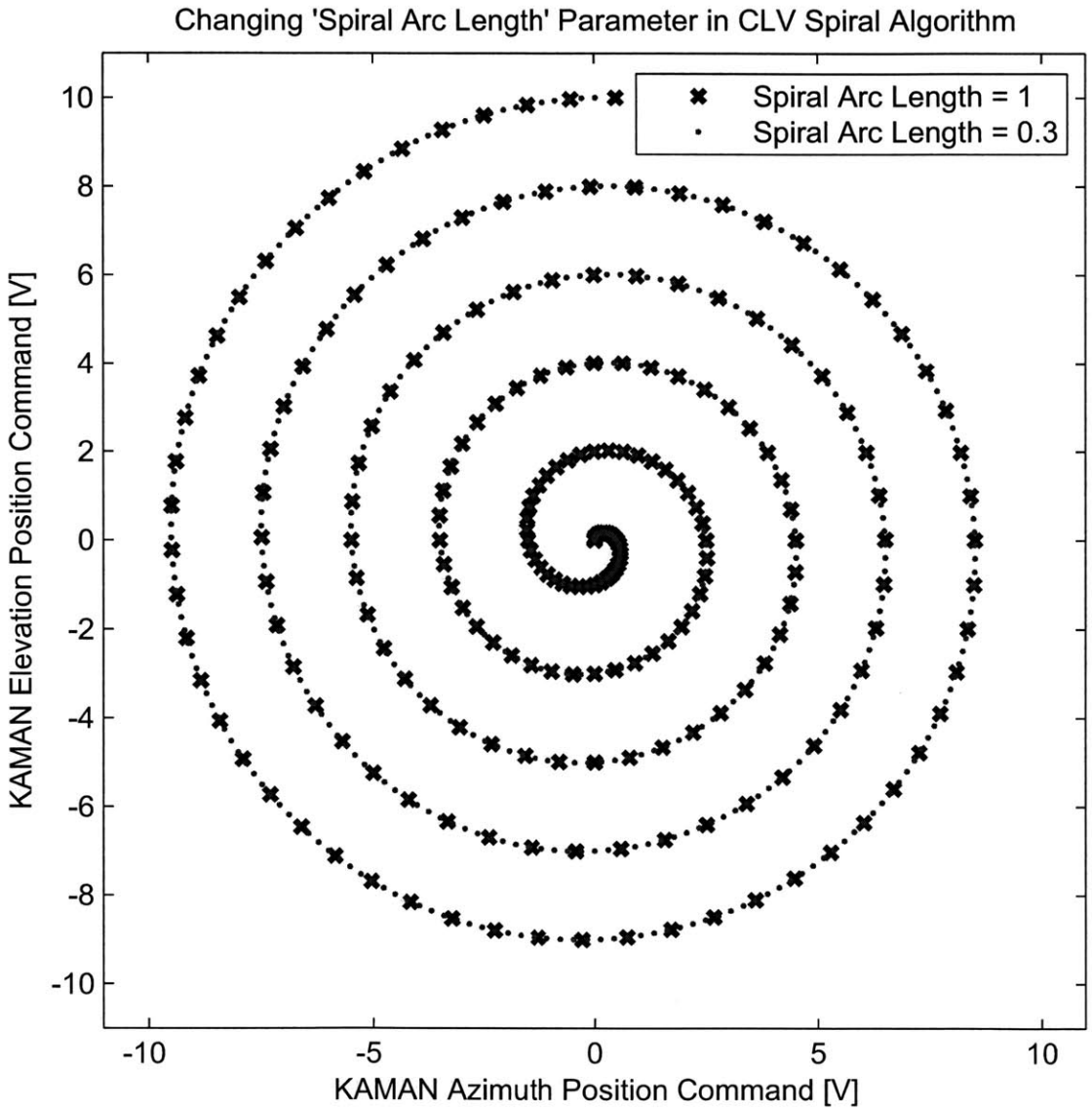

Figure 4-24: X-Y plot of the steering path showing the effect of changing the 'Spiral Arc Length' value in the CLV spiral algorithm.

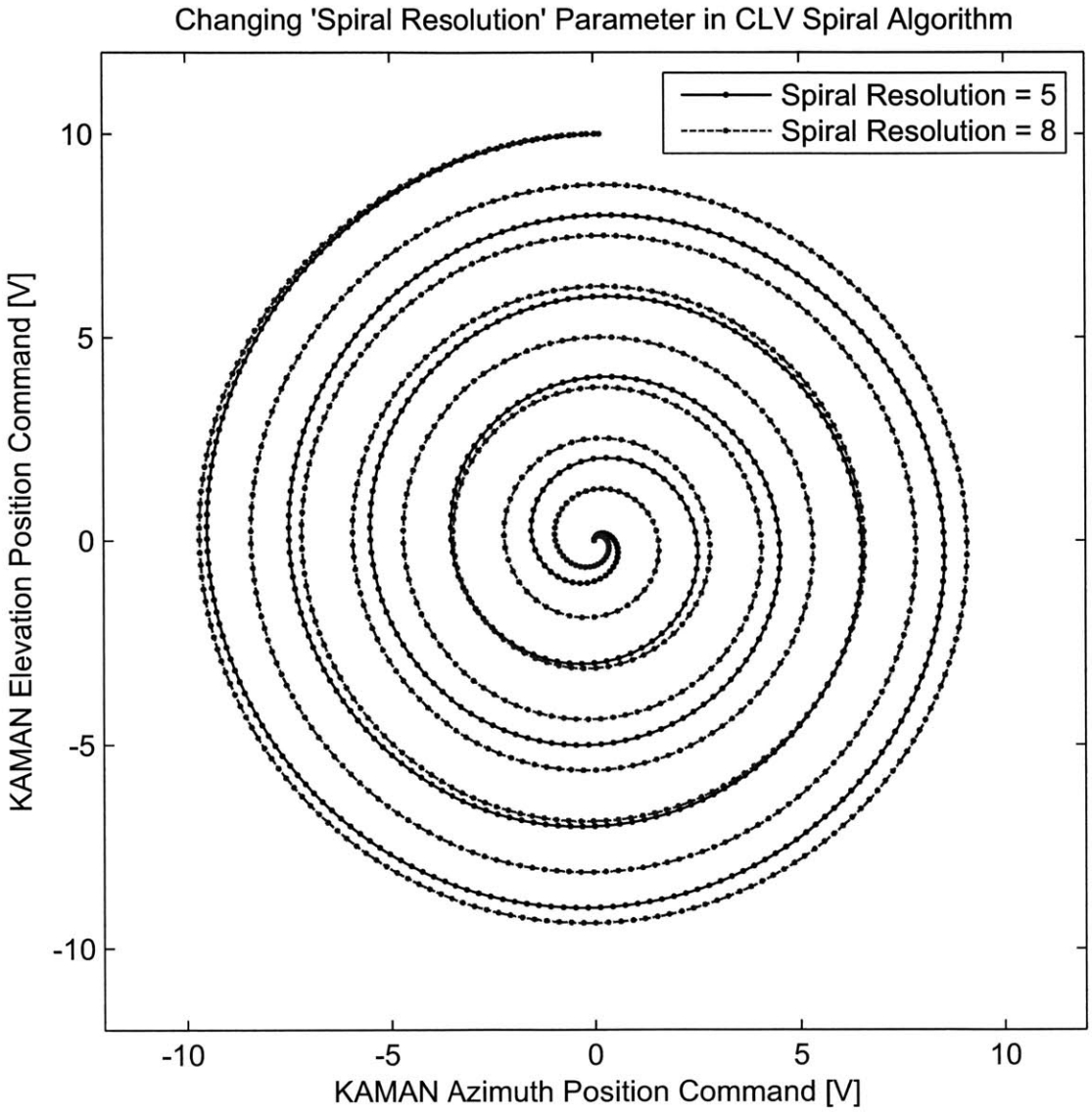

Figure 4-25: X-Y plot of the steering path showing the effect of changing the 'Spiral Resolution' value in the CLV spiral algorithm.

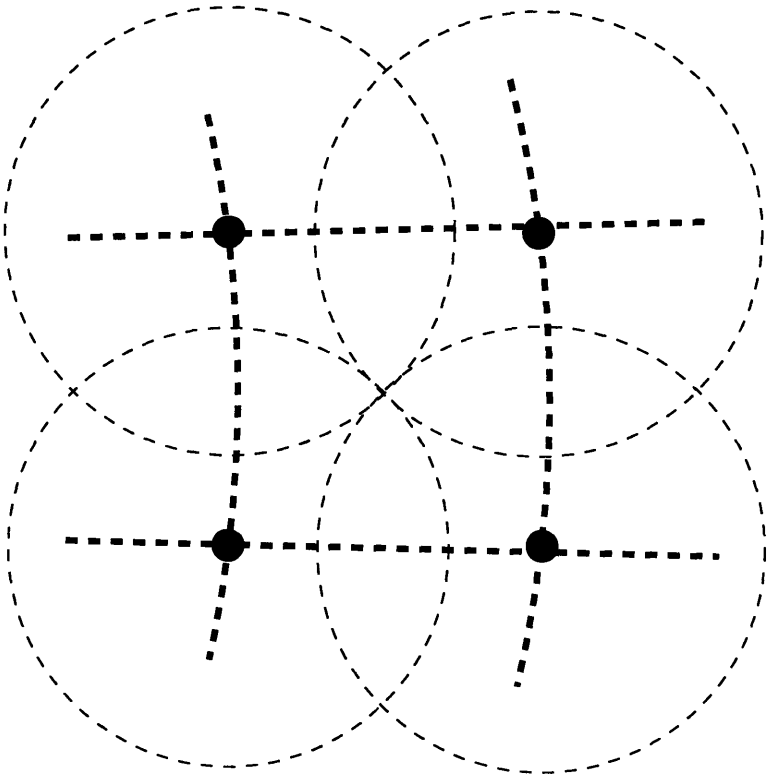

Figure 4-26: Diagram showing grid points visited during CLV spiral algorithm with effective coverage of each point.

4.4 Acquisition and Bumpless Transfer to Quad Cell Control

Section 4.3 reviewed the three methods for steering the beam in a spiral pattern. Each of these three steering methods use essentially the same Stateflow code to perform the actual search for the quad cell and switch into quad cell tracking mode. The Stateflow diagram representing acquisition mode is shown in Figure 4-27.

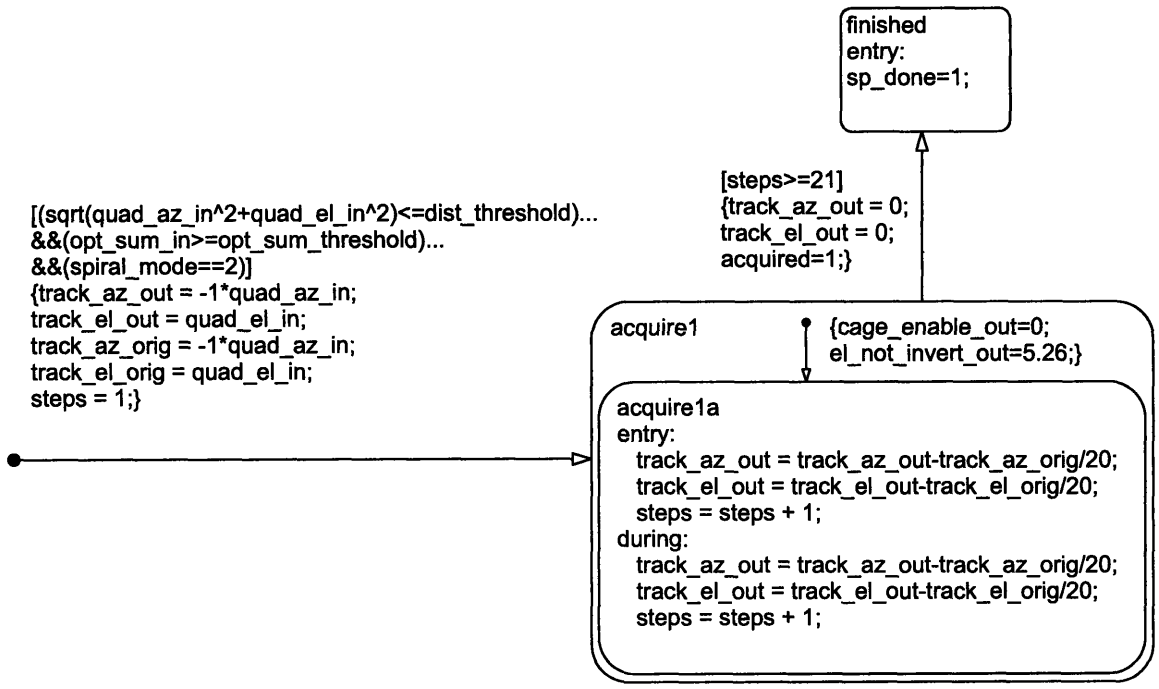

Figure 4-27: Stateflow diagram showing acquisition algorithm used with all spiral algorithms.

The Stateflow code, small as it is, controls both the parameters governing the searching algorithm as well as the bumpless transfer.

4.4.1 Searching Algorithm

The following code snippet is present (with only very minor differences) in all three algorithms.

```
[(sqrt(quad_az_in^2+quad_el_in^2)<=dist_threshold)...
&&(opt_sum_in>=opt_sum_threshold)...
&&(spiral_mode==2)]
{track_az_out = -1*quad_az_in;
track_el_out = quad_el_in;
track_az_orig = -1*quad_az_in;
track_el_orig = quad_el_in;
steps = 1;}
```

This code snippet defines the transition between every state controlling the spiral pattern and the *acquire1* state. The top two conditionals define the searching algorithm. In order for the state to switch to the *acquire1* leg, both of the following conditions must be true:

$$\sqrt{quad\_az\_in^2 + quad\_el\_in^2} \leq dist\_threshold \tag{4.8}$$

$$opt\_sum\_in \geq opt\_sum\_threshold \tag{4.9}$$

where *dist_threshold* and *opt_sum_threshold* are set by the user. These equations say that in order for the switch to acquire mode to occur, the beam spot must be within a certain distance of the center of the quad cell and the value of the Optical Sum input must be above a certain value. That's all there is to it. Finding the correct values of *dist_threshold* and *opt_sum_threshold* is the tricky part. Figures 4-28 through 4-31 address this problem. These figures show experimental data gathered using the square spiral algorithm with the SPOT-4DMI quad cell connected to the UDT Model 431 X-Y Position Indicator. Figure 4-28 shows the commanded voltages to the FSM while it was under KAMAN feedback control and the resulting quad cell position. The spiral starts at the center of the quad cell (after a successful acquisition) and set not to enter acquisition mode. Figure 4-29 shows the distance from the center of the quad cell and the Optical Sum as a function of the commanded KAMAN position during the same spiral.

Looking at Figure 4-28, it appears that the quad cell position outputs are extremely noisy. This is not true. The "noise" in this figure is corrupted data from the

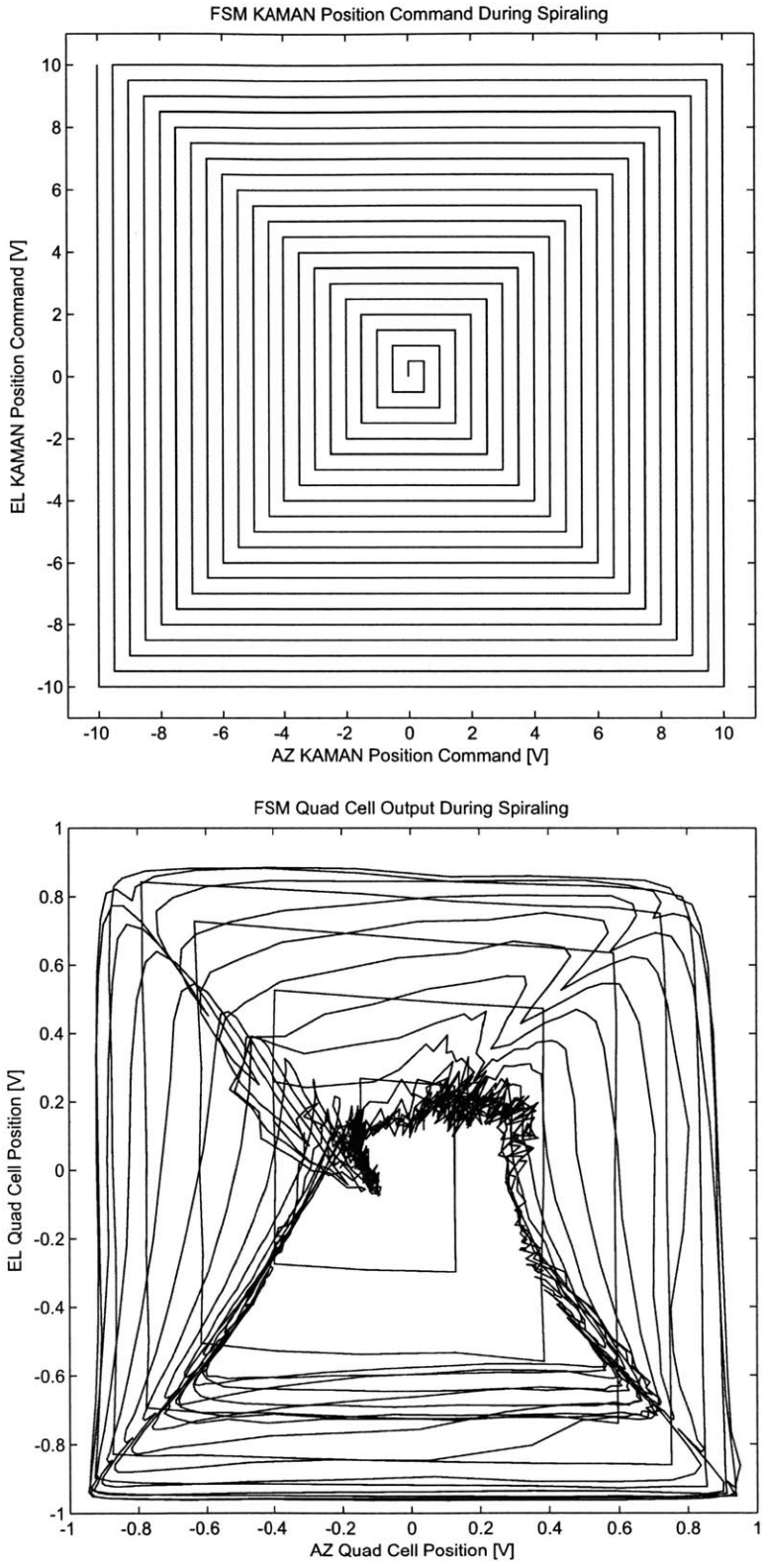

Figure 4-28: Experimental plots showing the FSM steering path and quad cell position over a 10 V range using the square spiral algorithm.

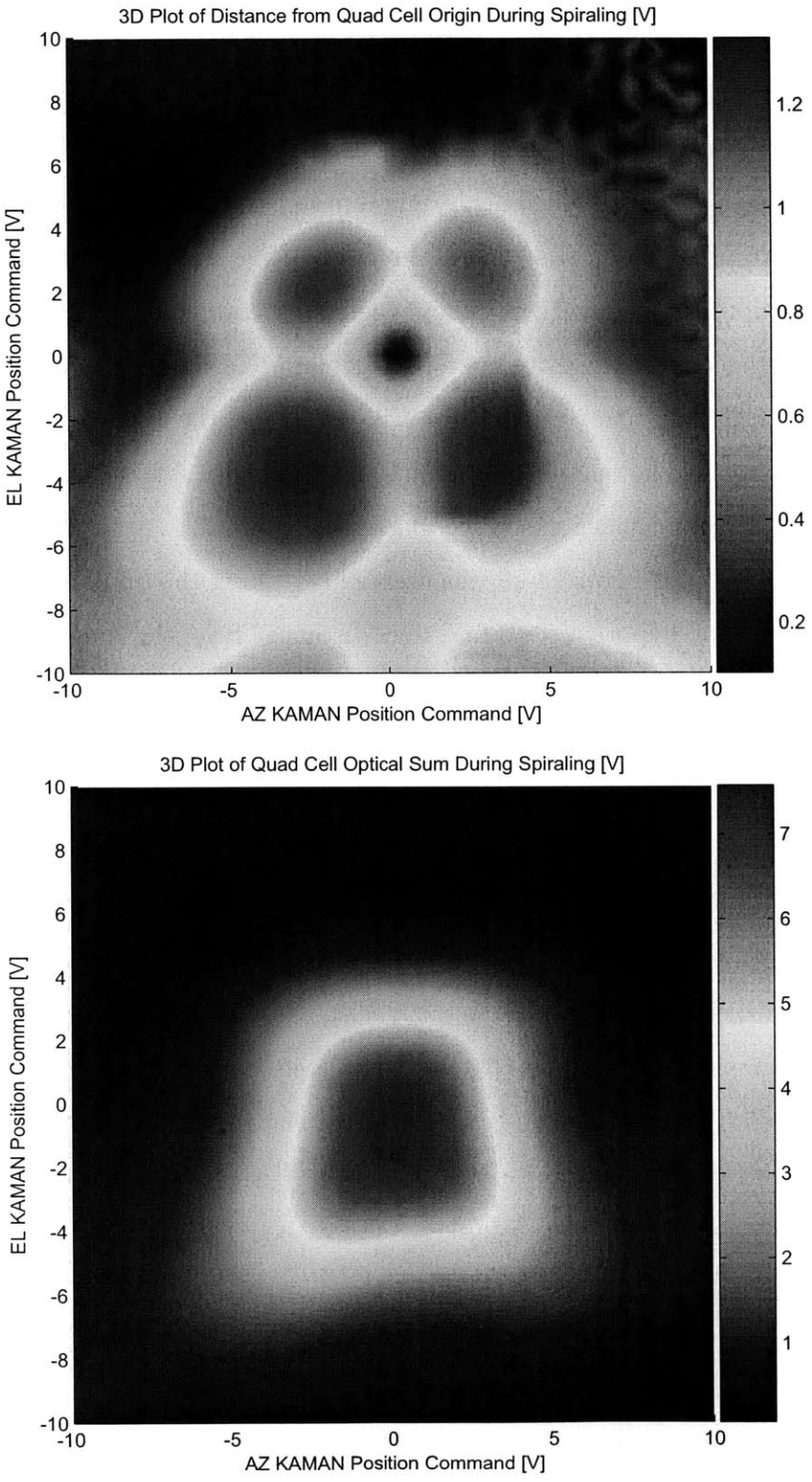

Figure 4-29: Experimental plots showing the distance from the origin of the quad cell and quad cell Optical Sum over a 10 V range using the square spiral algorithm.

Model 431 X-Y Position Indicator. When the laser moves off of the active region of the quad cell, the Model 431 has a hard time displaying the correct position. Instead of pegging off scale, the outputs tend to go back to zero. Several LEDs on the Model 431 light up when this error condition occurs, alerting the operator that the quad cell position data is corrupted. This is only a problem with the Model 431. The C/R Board tends to peg scale in the general direction of the spot when the laser is not on the active region of the quad cell, unless the spot is *completely* off of the quad cell. This is advantageous because optical feedback can be initiated when the beam is not on the quad cell, and the compensator will command the FSM to steer the beam toward the center of the quad cell (the quad cell looks like it is saturated rather than corrupted, and still provides useful feedback data).

The top portion of Figure 4-29 was created by taking the quad cell position voltages and taking the hypotenuse to find the "distance" from the quad cell origin. The "distance" voltage is $\sqrt{quad\_az\_in^2 + quad\_el\_in^2}$, which is one side of the inequality of Equation 4.8. This shows that if *dist_threshold* was set to .4 [V], the spiral algorithm would enter acquisition mode when the commanded position was in one of the darker blue regions (provided that Equation 4.9 was also satisfied). This also shows why both Equations 4.8 and 4.9 need to be satisfied to enter the *acquire1* state. Keeping $dist\_threshold = .4$ V, Equation 4.8 is satisfied near the center the origin of Figure 4-29 (which is the origin of the quad cell), and toward the outer fringes as well. The outer regions of the Figure 4-29 are completely off the quad cell, but Equation 4.9 is satisfied! This is because of the corrupted data gathered with the Model 431 when the beam is not pointed on the quad cell. If there was no *opt_sum_threshold* parameter, the spiral algorithm would switch into acquisition mode if it commanded the FSM completely off of the quad cell. Unfortunately, the quad cell feedback loops would go unstable after the FSM rammed into its mechanical stops (something to avoid). The top portion of Figure 4-29 is the Optical Sum output voltage during the same commanded spiral. The Optical Sum voltage is high near the center of the quad cell and falls off quickly as the spot moves off of the active region of the quad cell. By requiring both Equations 4.8 and 4.9 to be satisfied, the spiral algorithm

will not enter acquisition the beam is not pointed on the active region of the quad cell and when the Model 431 is outputting corrupted data. The user must correctly set both *dist_threshold* and *opt_sum_threshold* so that acquisition occurs only on the linear region of the quad cell. The central linear regions of Figures 4-28 and 4-29 are shown in Figure 4-30 and Figure 4-31.

Looking at Figure 4-31, the center of the quad cell is at the darkest blue point. A reasonable value for *dist_threshold* might be .3 V which would put the beam very close to the center of the quad cell. A reasonable value for *opt_sum_threshold* might be 6 V, insuring that acquisition only takes place in this linear region. These values only apply when using the Model 431 X-Y Position Indicator.

4.4.2 Bumpless Transfer

The rest of the Stateflow code in Figure 4-27 governs the bumpless transfer behavior of the algorithms. The idea behind the bumpless transfer is simple. In theory, once the algorithm is certain that it has found the active linear region of the quad cell, switching to quad cell feedback mode would simply cause the FSM to rapidly center the beam on the quad cell. However, this sudden centering of the beam creates a "bump" in the transfer. Avoiding this large sudden change in the FSM angular command ensures that the FSM will not go unstable during the transfer to quad cell feedback mode.

The implementation of the bumpless transfer is quite simple. After the searching algorithm has verified that the beam is pointed on the quad cell, the Azimuth and Elevation quad cell voltages are stored and the commands for the quad cell feedback path are set to these voltages (with the appropriate sign conventions if needed). For example, assume the FSM has found the quad cell and the current position output from the quad cell is [.12 -.17] (in terms of [*AZ_voltage EL_voltage*]). These sensor voltages are used to control the amplitude of the command voltages in the feedback path, in this case setting the Azimuth quad cell command to -.12 V (the quad cell Azimuth axis is inverted compared to the FSM Azimuth axis) and the Elevation quad cell command is set to -.17 V. When the feedback sensors are switched from KAMAN

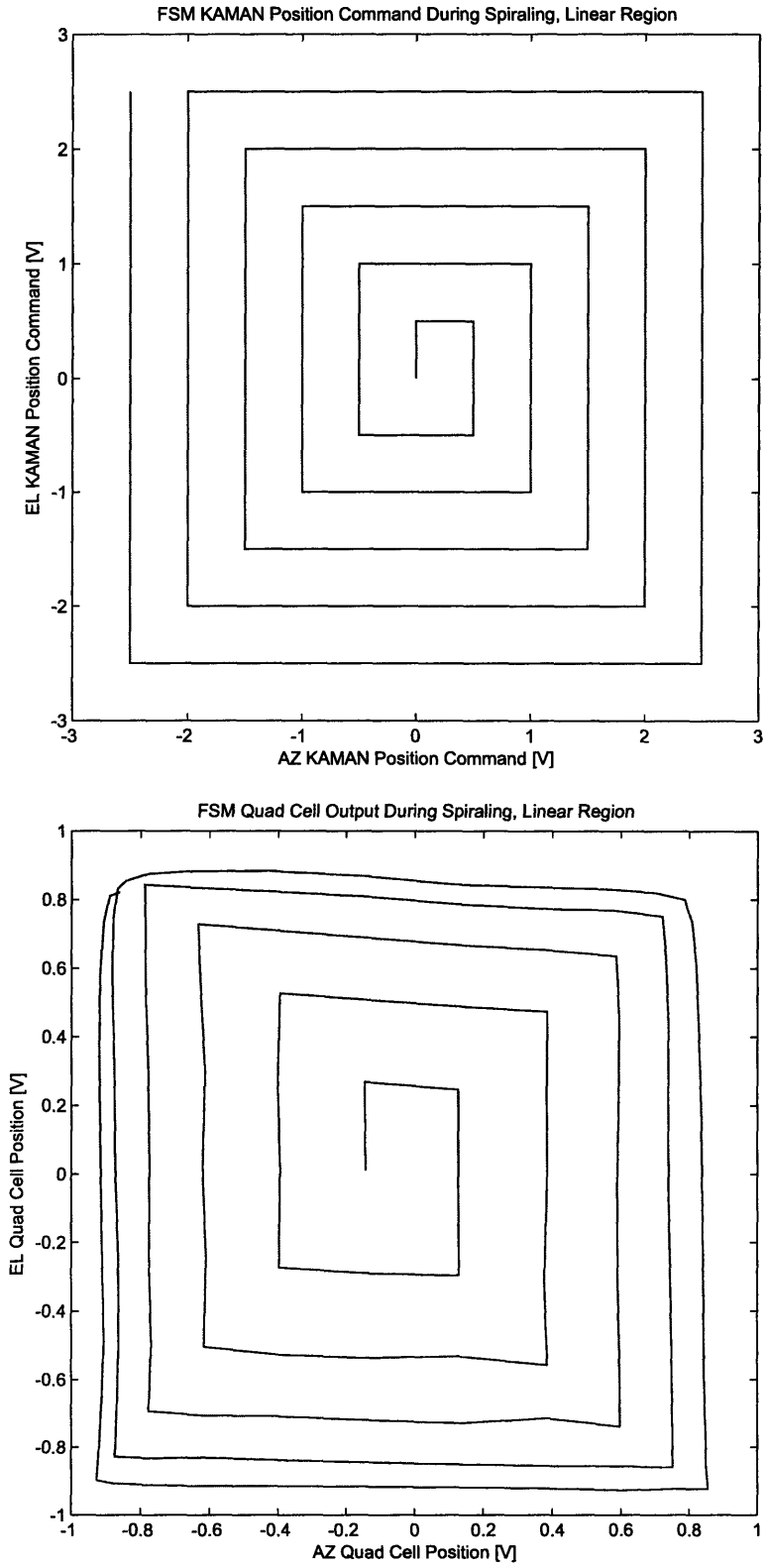

Figure 4-30: Experimental plots showing the FSM steering path and observed quad cell position in the linear range using the square spiral algorithm.

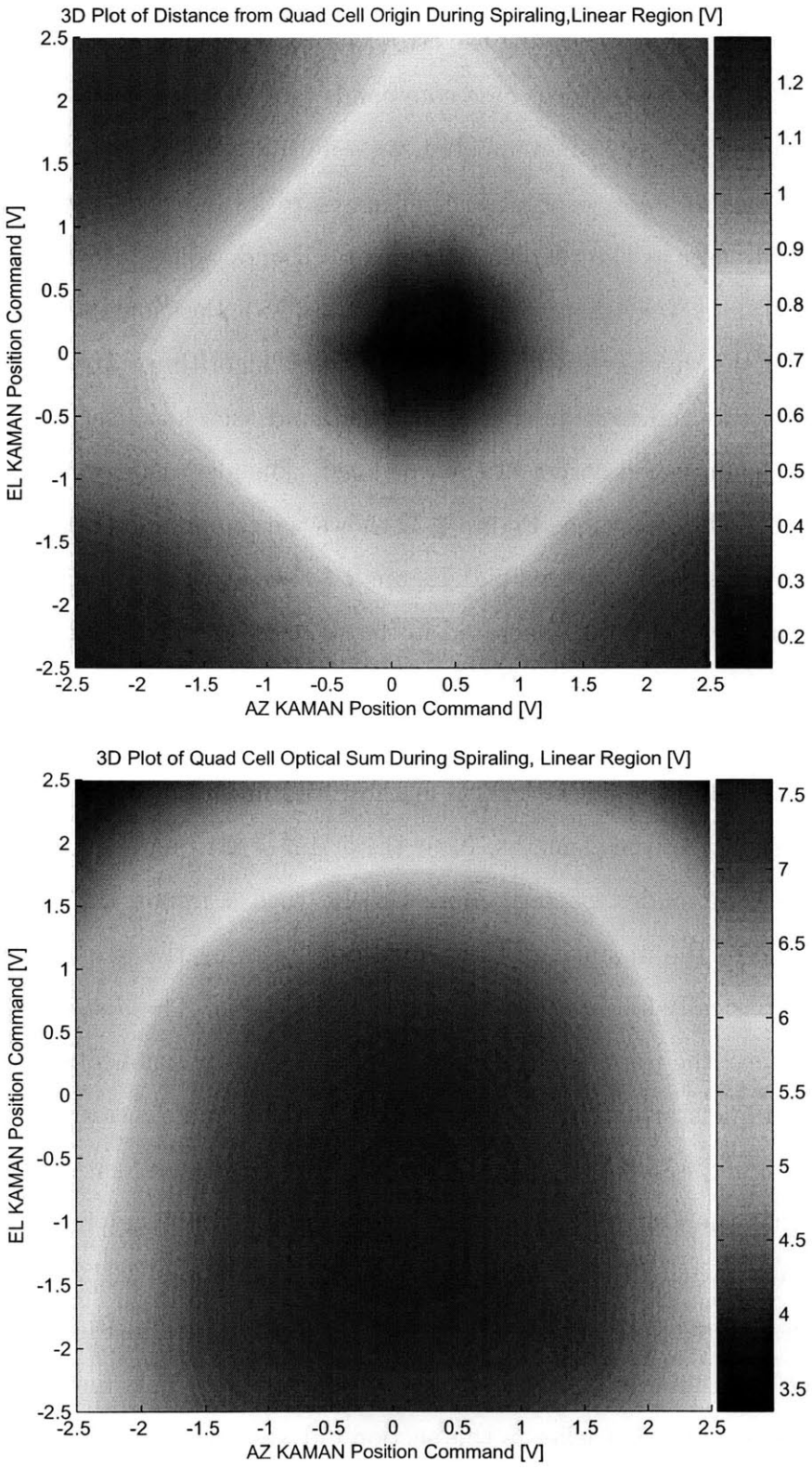

Figure 4-31: Experimental plots showing the distance from the origin of the quad cell and quad cell Optical Sum in the linear range using the square spiral algorithm.

to quad cell, the beam will not move because the commanded voltages are the same as the actual output voltages, driving the error to zero. Now in quad cell feedback mode, the last step is to slowly reduce these commands to [0 0] (the center of the quad cell). This is accomplished in 20 incremented steps during the evaluation of the *acquire1* state. After the 20 steps, each algorithm moves to its finished state, and the FSM is pointing the laser at the center of the quad cell under quad cell feedback control. Experimental data was taken with the Model 431 X-Y Position Indicator during an acquisition of the quad cell using the square spiral algorithm. To obtain this data, the algorithm was run to acquire the quad cell, and then an offset was commanded to steer the beam near a corner of the quad cell. The algorithm was then run again starting at this offset position. Figure 4-32 shows the commanded beam path and the actual quad cell output voltages and Figure 4-33 shows the distance from the quad cell origin and Optical Sum, generated in the same manner as Figure 4-29.

The first thing to notice in Figure 4-32 is that the quad cell voltage does not produce the uniform grid that was commanded, but rather a grid that is compressed on the left and bottom sides. This is because the linear region of the quad cell is only the inner 25% of the elements. Near the origin of the quad cell, the grid pattern matches the commanded pattern more closely, with a minor angular discrepancy (probably due to the mounting of the quad cell). During the acquisition, the quad cell voltages basically follow the commanded grid pattern. The *dist_threshold* was set near .15 V. Near the upper right corner of the quad cell position plot, the grid stops and a slanted line starts. This is the point where the bumpless transition occurs. At this point, Figure 4-33 shows that the distance from the quad cell origin is at a low point and that the Optical Sum is very near its maximum, implying that the beam is pointed very near to the origin of the quad cell. Equations 4.8 and 4.9 have been satisfied, and acquisition begins. The quad cell position plot in Figure 4-32 shows a straight line from this point to the origin of the quad cell. This line was made in 20 discrete steps as outlined earlier in this section (20 was chosen arbitrarily, the number could be much higher for a smoother transition). At this point, the beam is in optical feedback mode and pointed at the center of the quad cell, and the spiral acquisition

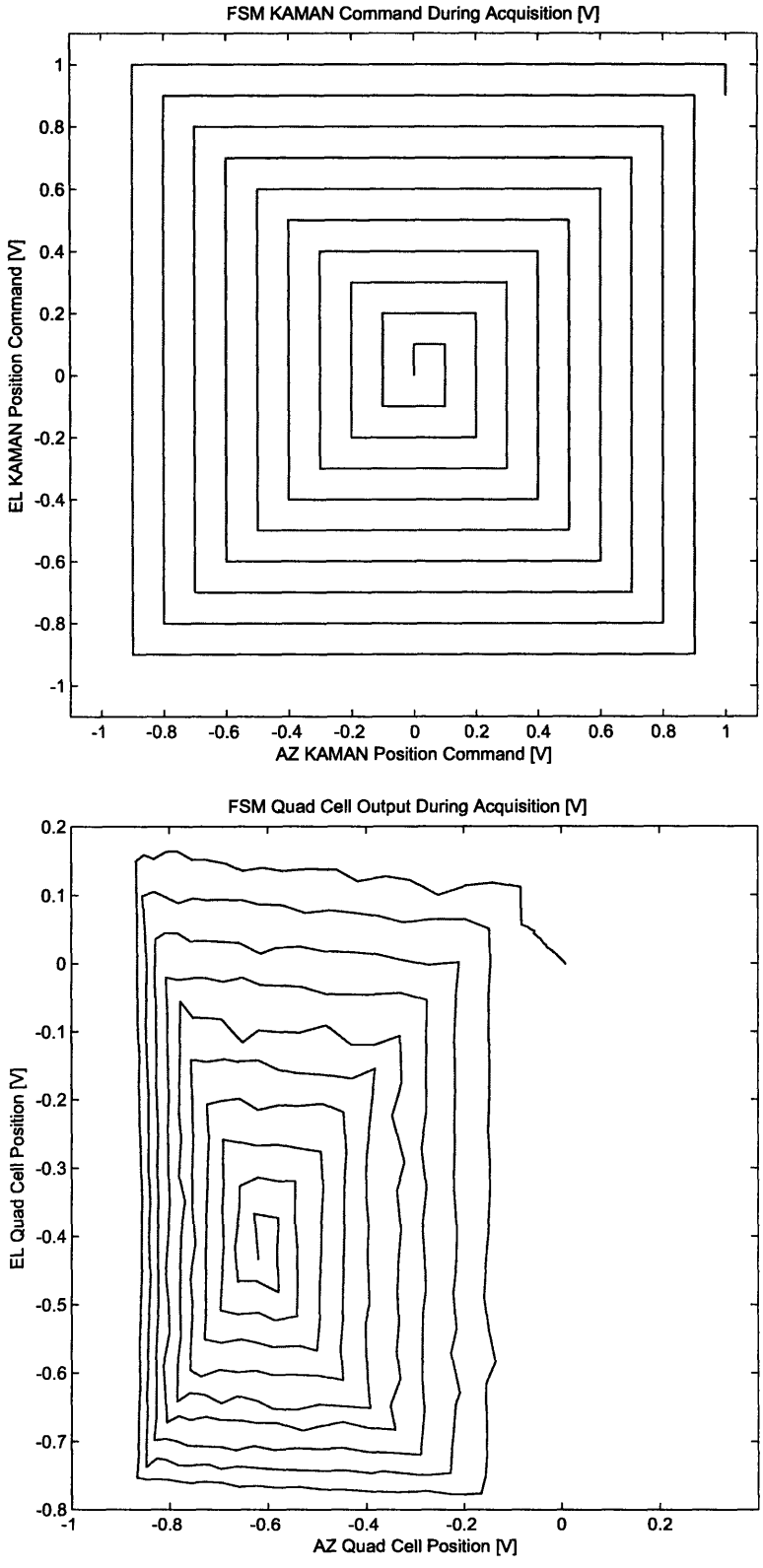

Figure 4-32: Experimental plots showing the FSM steering path and observed quad cell position during acquisition using the square spiral algorithm.

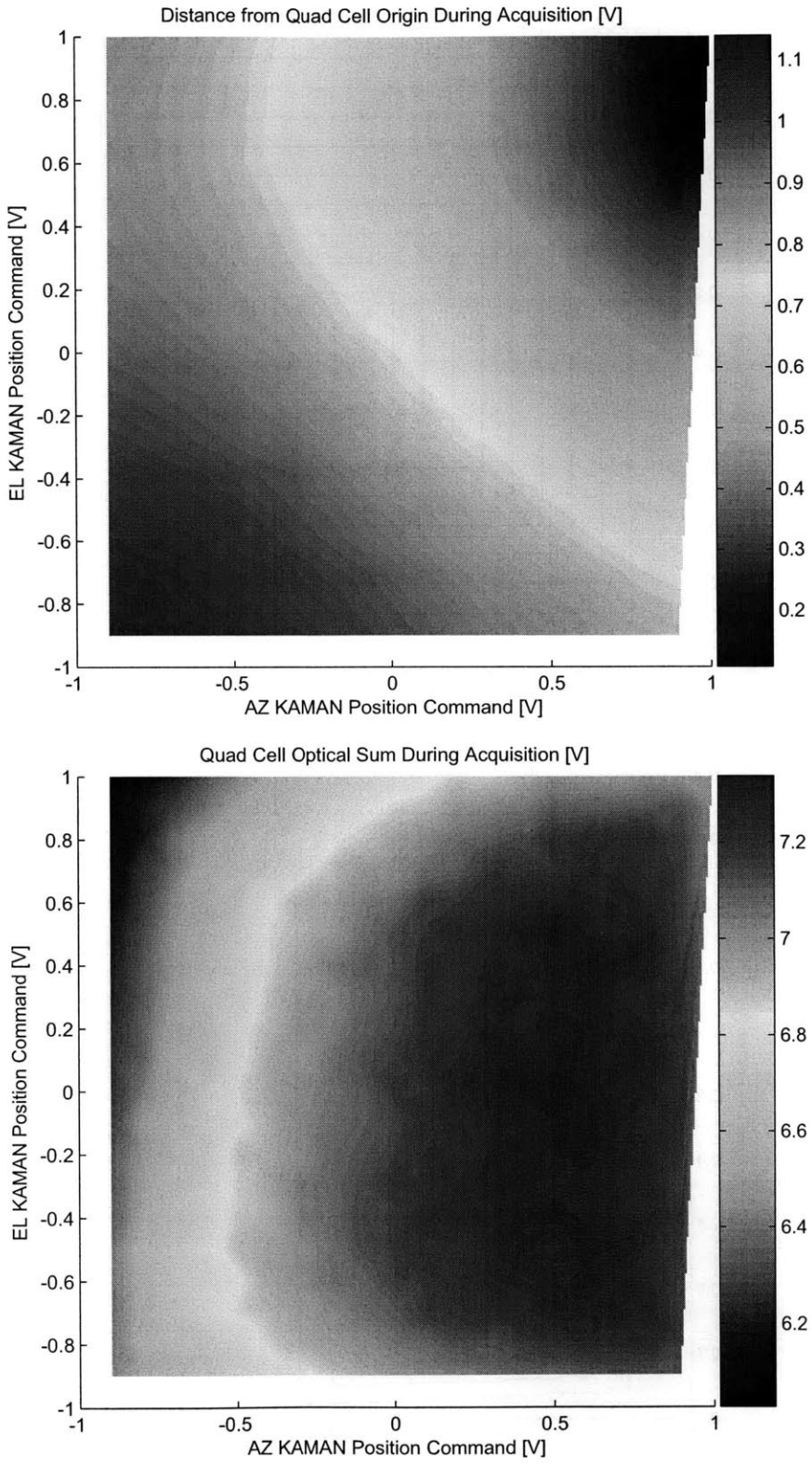

Figure 4-33: Experimental plots showing the distance from the origin of the quad cell and quad cell Optical Sum during acquisition using the square spiral algorithm.

has accomplished its task.

Chapter 5

FSM Compensator Design

The previous sections of this document have built up the tools necessary to control the FSM in a laboratory environment. This chapter focuses on the control of the FSM. Section 5.1 describes the original control hardware designed by Professor James K. Roberge. Section 5.1.1 describes the software control of this hardware in xPC, including the graphical user interface built to control the hardware. Section 5.1.2 looks at the analog implementation of the compensator, including the implementation of rate feedback. Section 5.1.3 includes the experimental closed loop responses of this compensator.

With the baseline performance established, Section 5.2 lays the groundwork for the new (and possibly improved) digital compensator. Sections 5.2.1 and 5.2.2 describe the implementation of the digital controller in xPC including the usage of the relevant graphical user interface. Section 5.2.3 describes the design process and the relevant requirements for the closed loop response. Section 5.2.4 describes the implementation of different compensators with their simulated responses. Finally, Section 5.2.5 reviews the experimental results obtained when these compensators were implemented.

5.1 Original Compensator Overview

The FSM used in these experiments has existed at Lincoln Labs for a number of years. James K. Roberge, a Professor in the Electrical Engineering and Computer Science

department at MIT designed the analog compensator used previous to this work. This section describes the xPC interface used to control the analog compensator during experimentation. The compensator is analyzed as well, and experimental results taken with the FSM under quad cell feedback using this analog compensator are included as a baseline performance reference.

5.1.1 xPC Integration

The analog controller shown in Figure 2-15 is a stand-alone controller, containing all of the components needed to control the FSM including the analog compensator, differential input channels for the feedback sensors (KAMAN and QUAD), a current amplifier to drive the FSM voice coils and the necessary power supplies for the system. However, a method is needed to properly command this stand-alone controller. The command of the analog controller is accomplished with Simulink and xPC. Figure 5-1 shows the Simulink model ran on the xPC Target computer during experimentation.

The graphical user interface used is the same as shown in Figure 4-22. Note that the MIRU controls are not implemented in Figure 5-1. In normal operation of the laboratory, xPC must be controlling both the FSM and the MIRU, and a more complicated model is used. This model is run at a constant sampling rate of 10 kHz, and contains the compensation for the MIRU hardware. However, the MIRU implementation will not be discussed in this document. Each block in Figure 4-22 controls a different part of the model in Figure 5-1. The 'Spiral Locater' panel is covered extensively in Chapter 4.

The 'FSM Control Panel' panel provides a user interface to control which feedback sensors are used with the FSM. 'Cage' refers to using the local KAMAN sensors as the feedback sensors. 'Track' refers to optical tracking mode using the quad cell as the feedback sensor. The 'Mode Select' button group selects between four available modes:

1. Both axes in Cage (KAMAN) mode.

2. Azimuth axis in Cage mode, Elevation axis in Track (quad cell) mode.

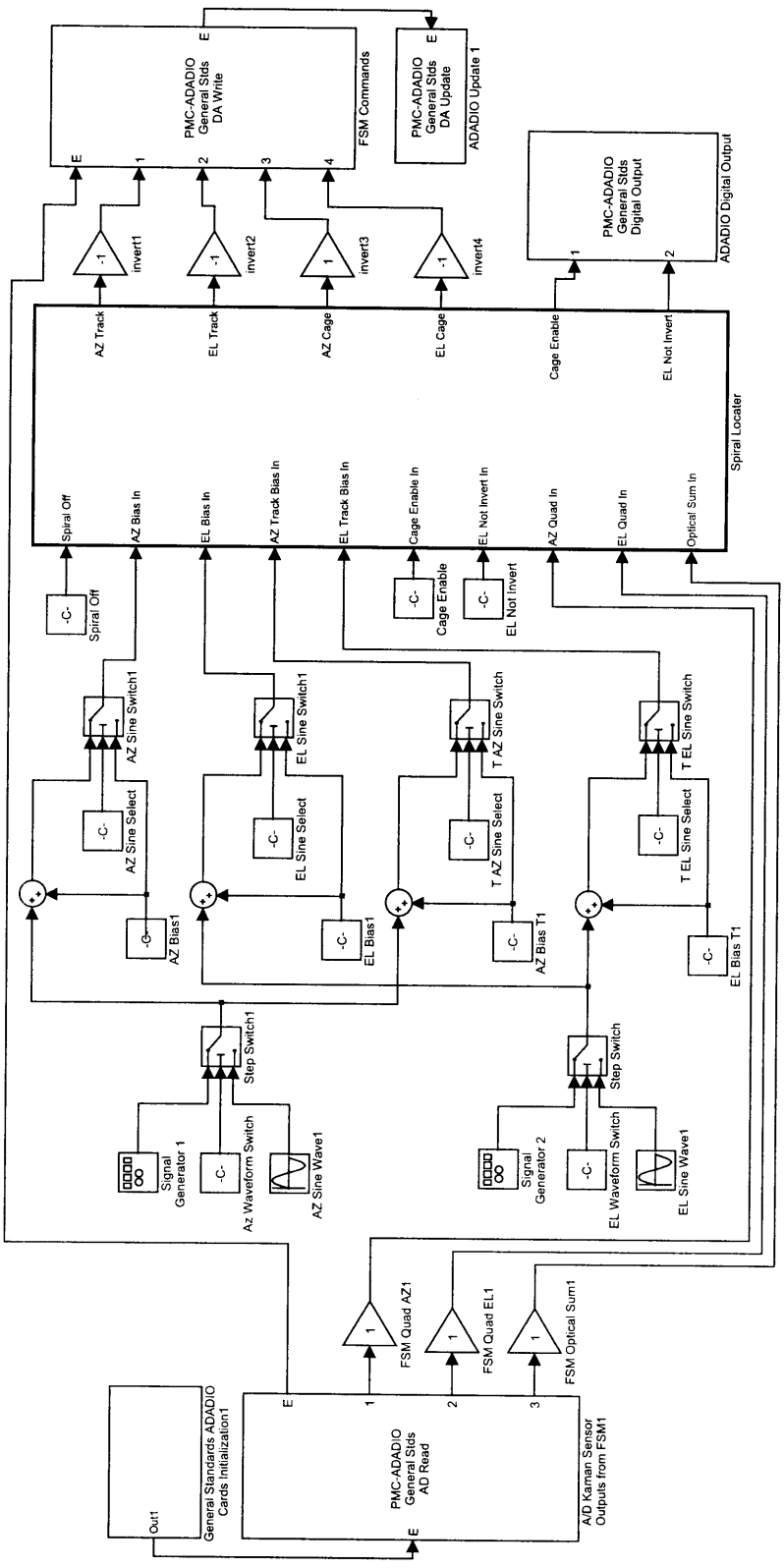

Figure 5-1: Simulink model used with analog compensator.

3. Elevation axis in Cage mode, Azimuth axis in Track mode.

4. Both axes in Track mode.

The digital signals 'Cage Enable' and 'EL Not Invert' are set according to which mode is selected. These digital signals control digital switches in the Analog Controller, choosing which feedback sensors to use. The 'Cage and Tracking Mode Inputs' group allows manual input of commands to each of the four feedback loops. The 'Bias' Commands and the 'Cage Commands' are added together and applied to the appropriate axis. This allows zeroing out the KAMAN sensors if there is any offset present. The 'Track' commands are outputted to the quad cell loops of the Analog Compensator.

The 'FSM Waveform Input Control' panel allows the selection of different signals to be used as commands to the feedback loops of the Analog Compensator. These signals can be sine waves or step commands, set to either or both axes of the FSM. The commands are sent to each feedback loop (KAMAN or quad cell). This panel also allows control of the frequency, amplitude and phase of these signals. The 'I Love Lincoln' button sets these parameters to values that produce the Lincoln Laboratory logo, a Lissajous figure.

5.1.2 Analog Compensator

The theory behind Professor Roberge's Analog Compensator is simple. He begins by representing the FSM as an underdamped second order system. The equation for such a system is usually written as

$$\frac{1}{\frac{1}{\omega_n^2}s^2 + \frac{2\zeta}{\omega_n}s + 1} \tag{5.1}$$

This system has a significant peak near its natural frequency. To reduce this peak, the Analog Compensator implements a minor loop using rate feedback. Rate feedback is accomplished by adding a derivative term into the feedback path of the mirror. The block diagram implementation is shown in Figure 5-2.

Applying Black's formula to the feedback loop, the new mirror transfer function

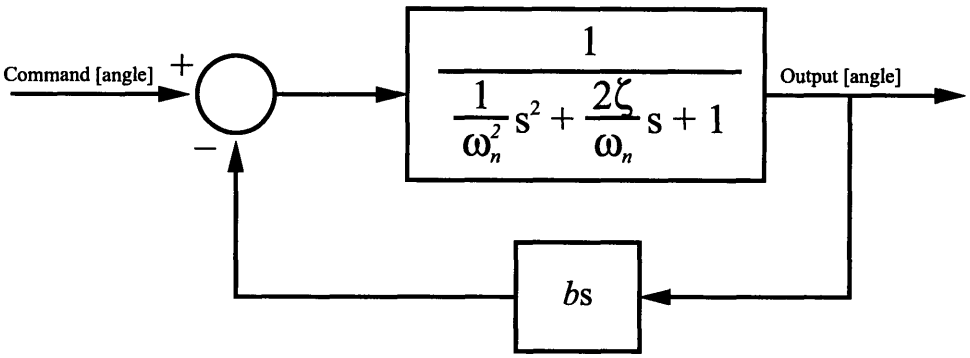

Figure 5-2: Block diagram showing rate feedback applied to an underdamped 2^{nd} order system.

becomes

$$\frac{1}{\frac{1}{\omega_n^2}s^2 + \left(b + \frac{2\zeta}{\omega_n}\right)s + 1} \tag{5.2}$$

This means that the rate feedback loop provides electronic damping to the mirror. With the proper amount of damping, the mirror's transfer function is made to be critically damped. Figure 5-3 shows a reduced order model for the Azimuth axis under quad cell control, with and without rate feedback.

The mirror with rate feedback can now be represented as a system with two real poles at the natural frequency of the mirror, ω_n. The idea now is to implement the compensator as shown in Figure 5-4.

By adding the inverse of the mirror under feedback control (the inverse of Equation 5.2) into the forward path, the dynamics of the mirror are effectively canceled out. With this in place, $G_c(s)$ can be made to be whatever is desired, and the negative loop transmission will be $G_c(s)H(s)$, where $H(s)$ represents the dynamics of the feedback sensor used (either KAMAN or quad cell). The compensator $G_c(s)$ chosen is a simple integrator $G_c(s) = K/s$. The design of this compensator is independent of the feedback sensor used. The only "cancelation" of the system is that of the major resonant peak near 120 Hz. Thus, the bandwidth is limited by the first resonant peak, which differs between the KAMAN and quad cell. For the same compensator, the crossover frequency will be limited by the KAMAN loop, as its resonant peaks

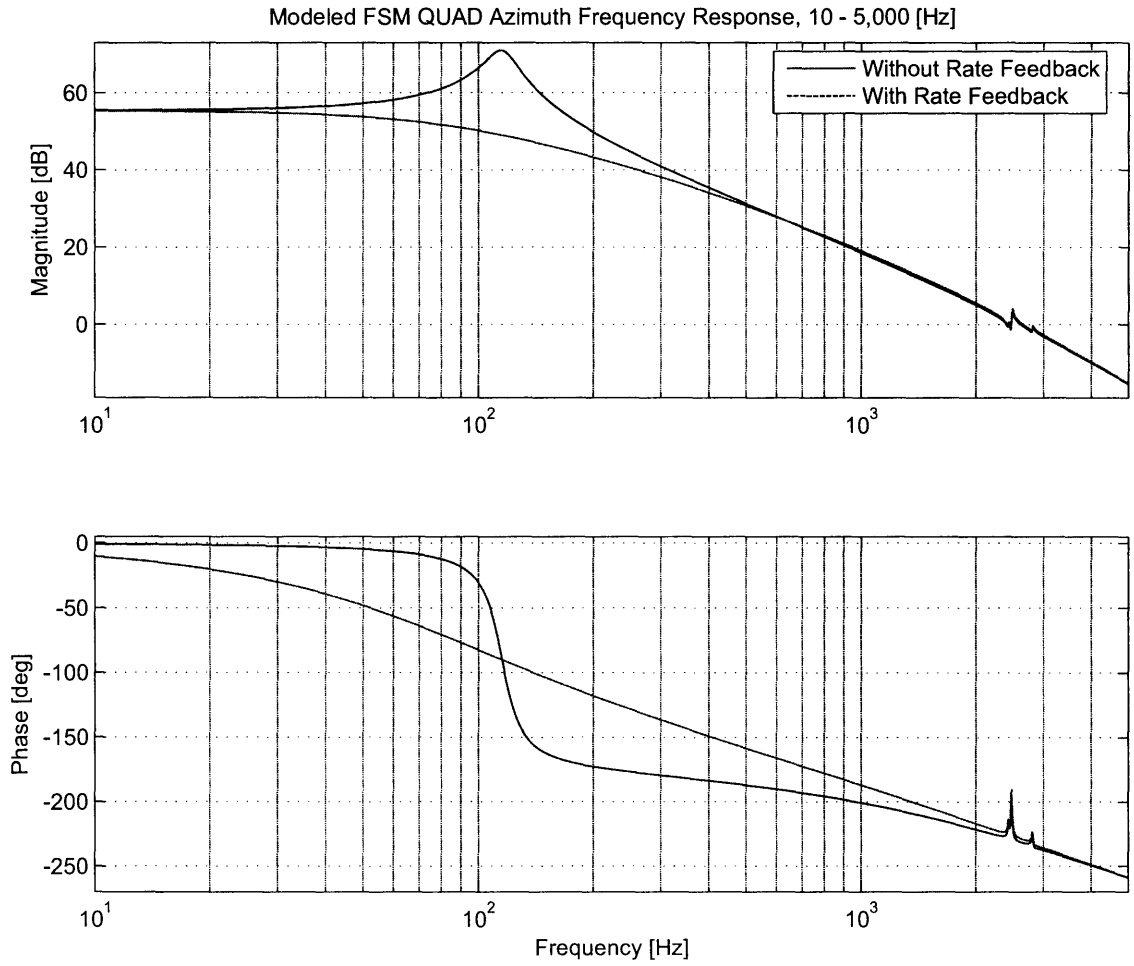

Figure 5-3: Comparison of the modeled open loop frequency responses of the FSM Azimuth axis using the quad cell, with and without rate feedback.

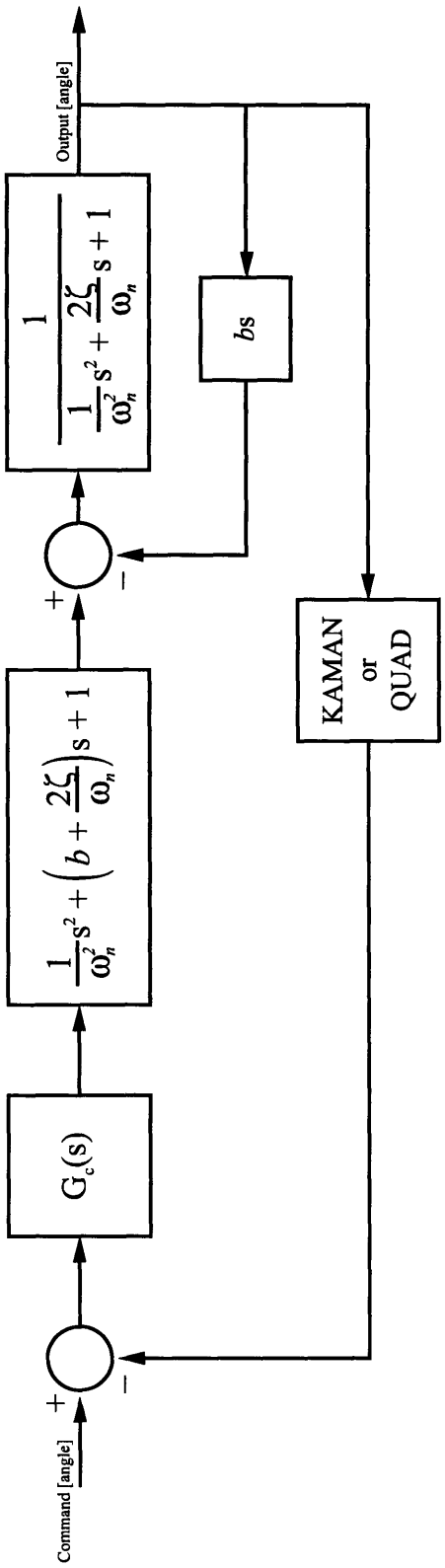

Figure 5-4: Block diagram of the feedback control system using the analog compensator with rate feedback, with the FSM represented as an underdamped 2^{nd} order system.

occur much earlier in the frequency response. Resistors can be set to give different loop gains between the KAMAN and quad cell feedback loops, allowing the quad cell loops to achieve higher bandwidth than the KAMAN loops.

Compensator Model

The schematic for the analog implementation of the compensator is shown in Figure 2-18. The transfer function of the total compensator function (excluding the rate feedback term) realized by this circuitry is shown in Figure 5-5.

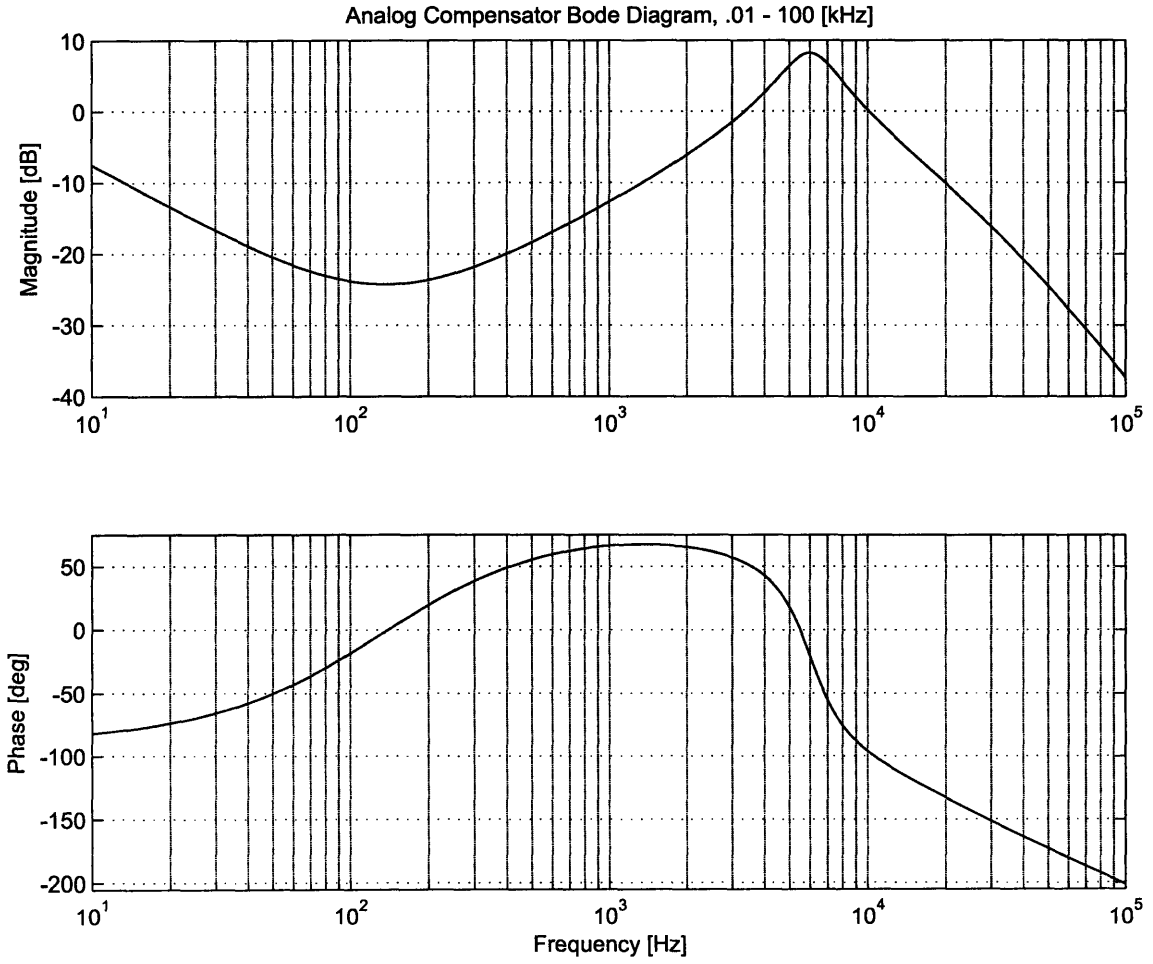

Figure 5-5: Modeled frequency response of the original analog compensator, V_{in}/V_{out}, from 10 Hz to 100 kHz.

5.1.3 Baseline Experimental Results

The data collected and presented for baseline results was for the Elevation Axis; the Azimuth responses are virtually identical. To determine the baseline results with which to compare the digital compensator, the gain of the compensator was set as high as possible while maintaining a gain margin $G_m \geq 6$ dB and phase margin $\Phi_m \geq 30$ degrees. Using these conditions and playing with the gain of the compensator, the crossover frequency of the negative loop transmission was pushed to $f_c \approx 1$ kHz. The loop transmission, along with the simulated model loop transmission is shown in Figure 5-6. Figure 5-7 shows the experimental data, but zoomed in after crossover near the first "doublet" of the mirror. This plot shows that the gain margin is $G_m = 6$ dB. The phase margin for this system is $\Phi_m = 38$ degrees.

To prove the closed loop stability of this system[1], Figures 5-8 and 5-9 show Nyquist plots of the negative loop transmission of the system.

Figure 5-10 shows the complete closed loop frequency response.

The overall performance specifications for the Elevation Axis in quad cell feedback mode using the analog compensator are summarized in Table 5.1.

Parameter	**Value**
f_c [kHz]	.997
G_m [dB]	6
Φ_m [deg]	37
$D_r(s)$ @ 10 Hz [dB]	44.8
-3 dB Frequency [kHz]	1.9

Table 5.1: Experimental performance specifications of the FSM quad cell Elevation axis using the original analog compensator.

[1]The famous Catch-22 of drawing a Nyquist diagram to prove a system stable with data that could only be obtained if the system was stable.

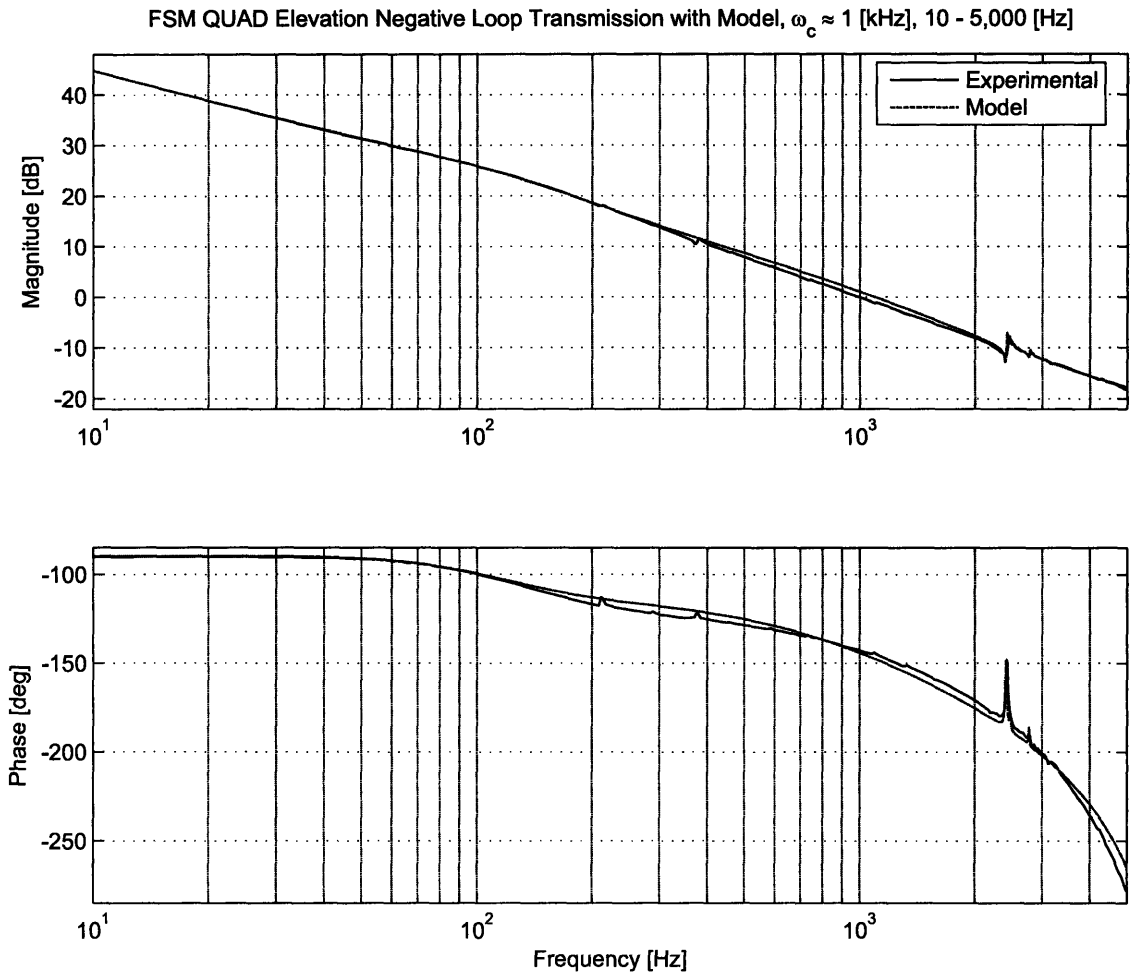

Figure 5-6: Comparison between the experimental and modeled negative loop transmissions of the FSM in quad cell feedback mode using the original analog compensator with $f_c \approx 1$ kHz.

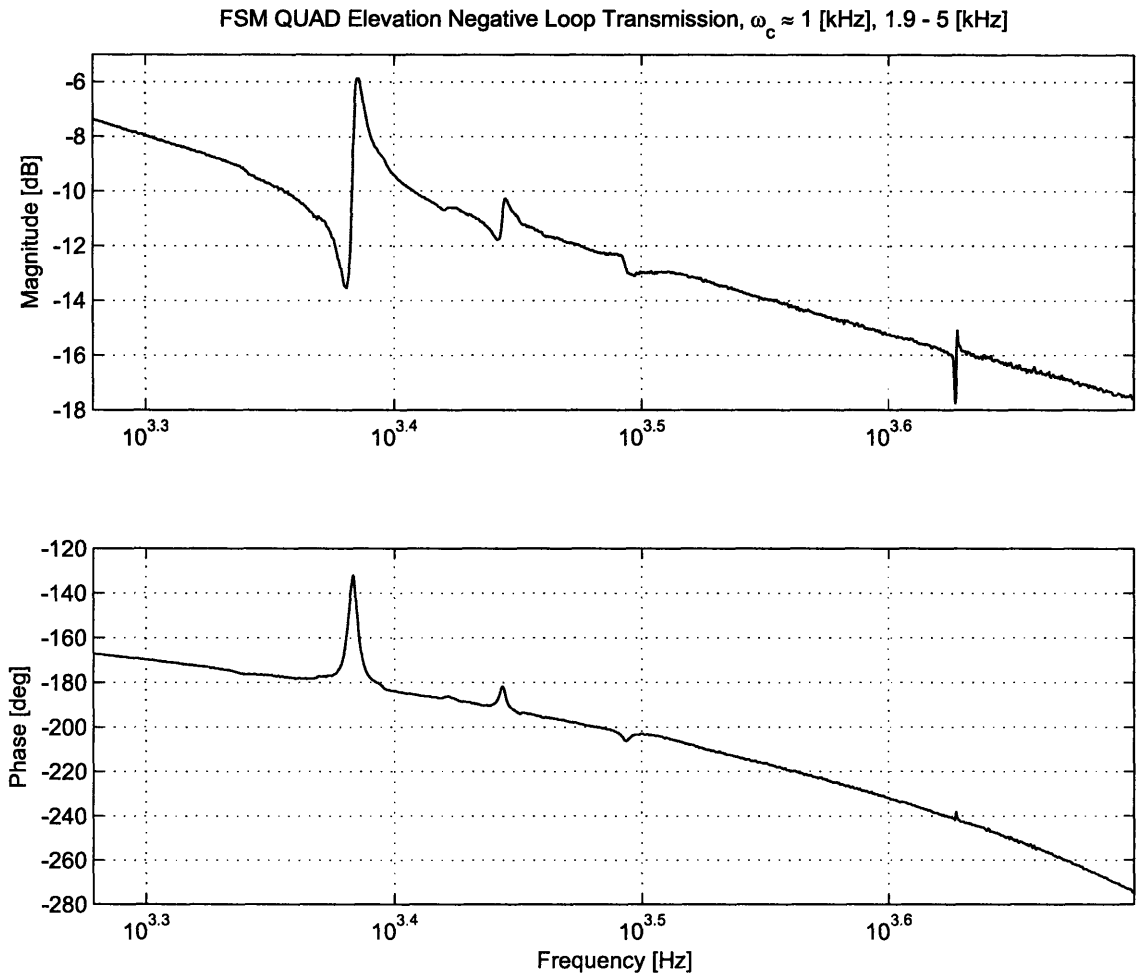

Figure 5-7: Negative of the loop transmission of FSM Elevation axis in quad cell feedback mode using the original analog compensator with $f_c \approx 1$ kHz, 1.9 kHz To 5 kHz.

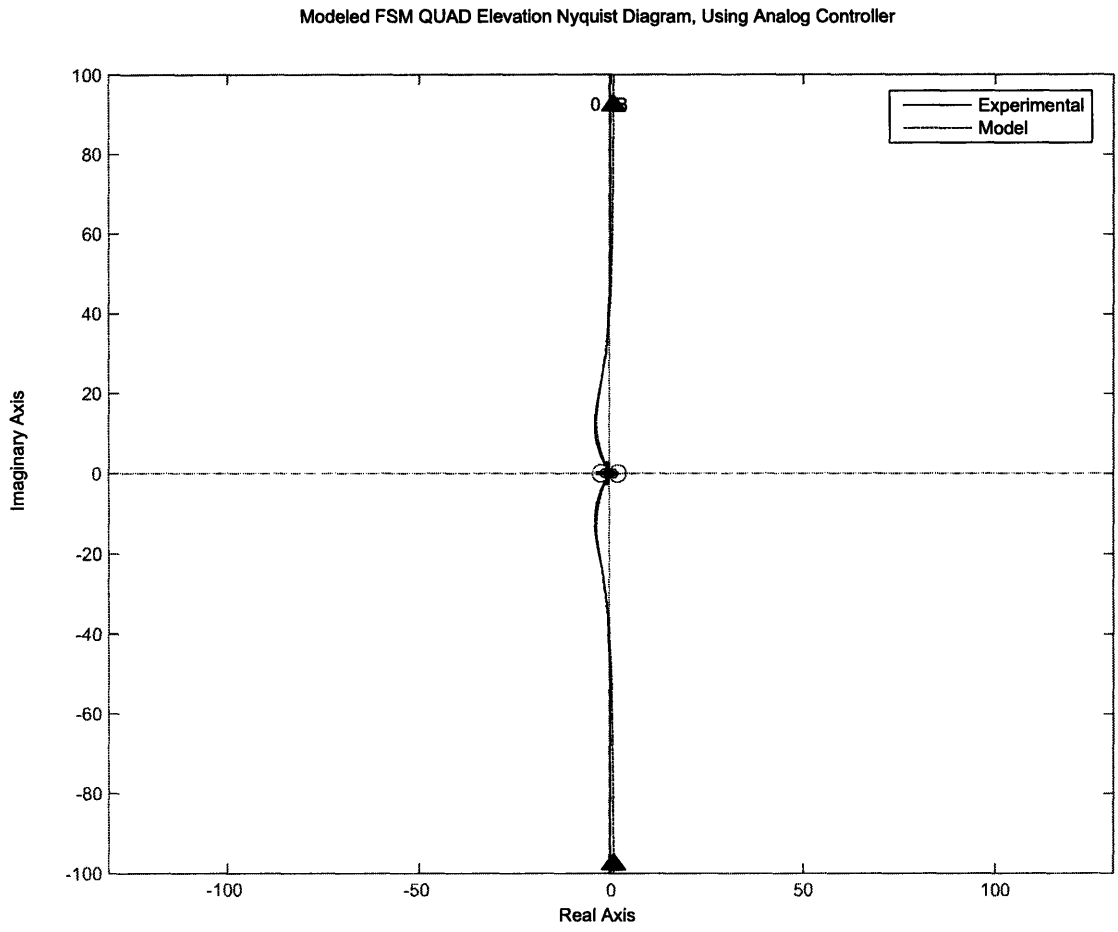

Figure 5-8: Simulated and experimental Nyquist plots of the negative loop transmission of the FSM Elevation axis in quad cell feedback mode using the original analog compensator with $f_c \approx 1$ kHz. The region near the -1 point shown in detail in Figure 5-9.

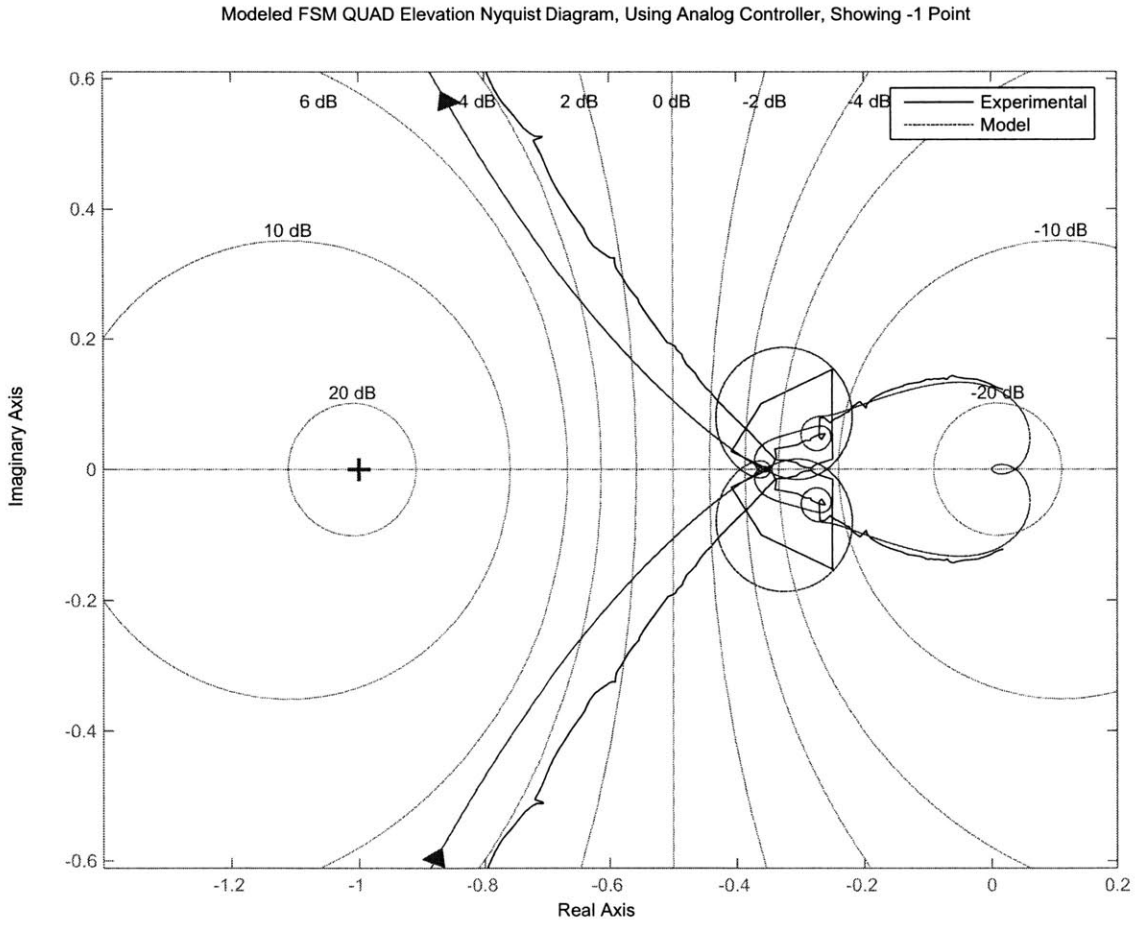

Figure 5-9: Simulated and experimental Nyquist plots of the negative loop transmission of the FSM Elevation axis in quad cell feedback mode using the original analog compensator with $f_c \approx 1$ kHz, zoomed in to show the -1 point.

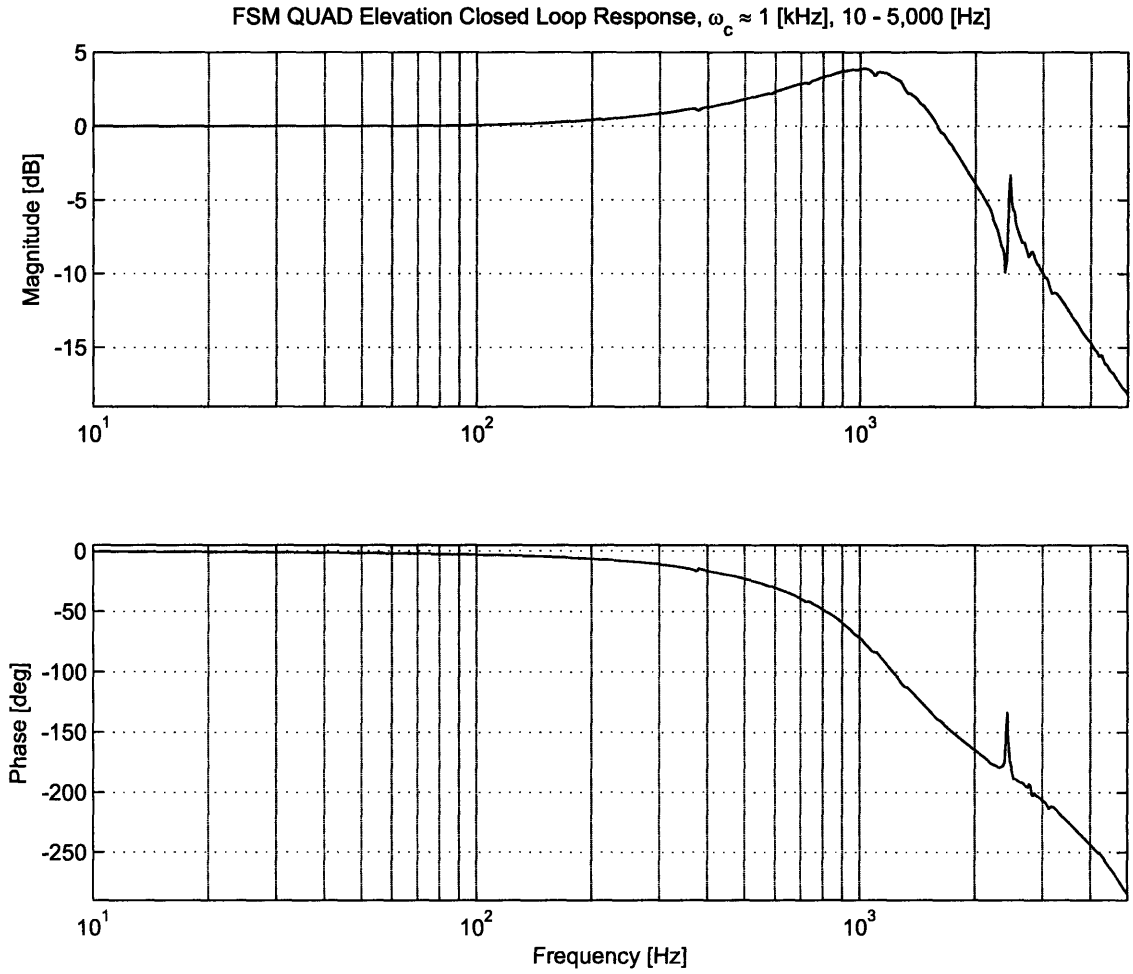

Figure 5-10: Closed loop frequency response of the FSM Elevation axis in quad cell feedback mode using the original analog compensator with $f_c \approx 1$ kHz, 10 Hz To 5 kHz.

5.2 New Compensator Overview

It is obvious from the plots in Section 5.1.3 that the Original Analog Controller, as built, cannot achieve a higher bandwidth. The question is if any controller can achieve a higher bandwidth given the hardware characteristics of the FSM. This section seeks to answer that question.

5.2.1 xPC Digital Control

A block diagram of the feedback system implemented is shown in Figure 5-11.

Referring to Figure 5-11, $C(s)$ is the command voltage into the feedback loop, $G_c(s)$ is the digital compensator, $E(s)$ is the summing junction error, $M(s)$ is the model of one FSM axis, including the current amplifier, the mirror and the dynamics associated with the sensor (for the Quad cell, the dynamics can be ignored in the frequencies of interest, and the block just becomes a gain), bs is the rate feedback damping, $Q(s)$ is the quad cell voltage out (or the angle out with a simple gain term), and $D(s)$ is the disturbance input. The minor loop containing the rate feedback can be simplified to

$$M_r(s) = \frac{M(s)}{1 + M(s) \cdot bs} \tag{5.3}$$

where $M_r(s) = M(s)$ if $b = 0$ and rate feedback is not used. Therefore, the loop transmission of the system will be $LT = G_c(s)M_r(s)$. The disturbance response is $Q(s)/D(s) \approx 1/LT$. The disturbance rejection can be thought of as the inverse of the disturbance response, and thus is $D_r(s) = LT$. The two blocks $G_c(s)$ and bs are the controllable user inputs.

The design of the new compensator will go through several design iterations. It would not be practical to implement this compensator in the analog world until the final design is chosen. For this reason, a digital compensator is used. Fortunately, xPC is the perfect platform to run a high bandwidth digital compensator that can be easily modified. Figure 5-12 shows the xPC block diagram of the digital controller.

The controller is initialized by creating the discrete time state-space matrices associated with the desired continuous time compensation function $G_c(s)$. The dis-

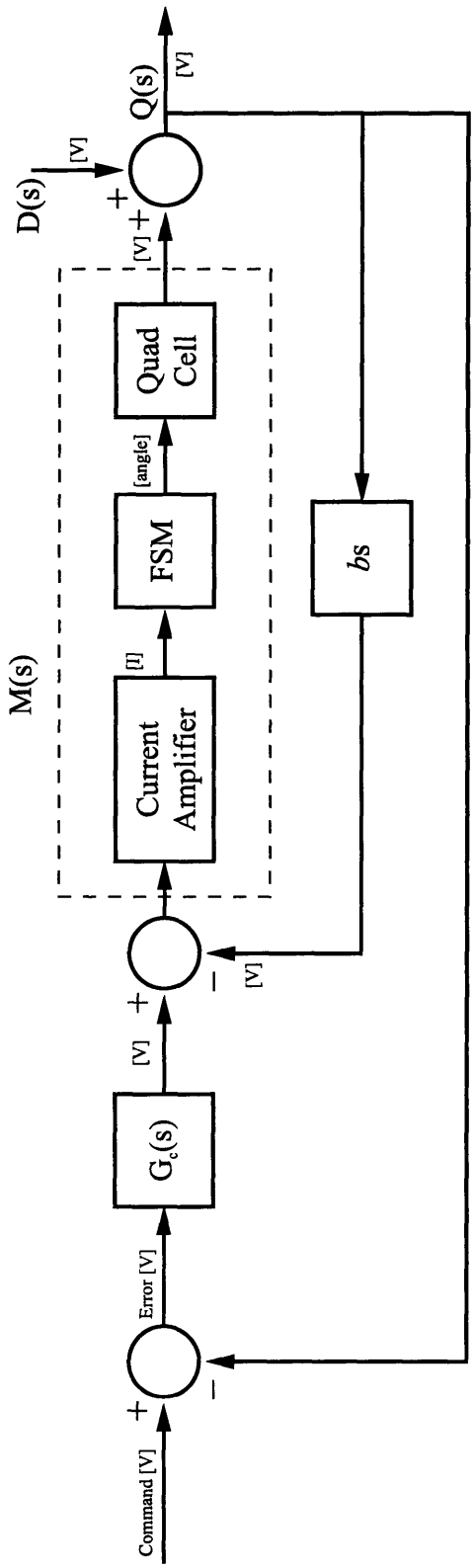

Figure 5-11: Block diagram of the feedback control system implemented with the digital compensators, including rate feedback.

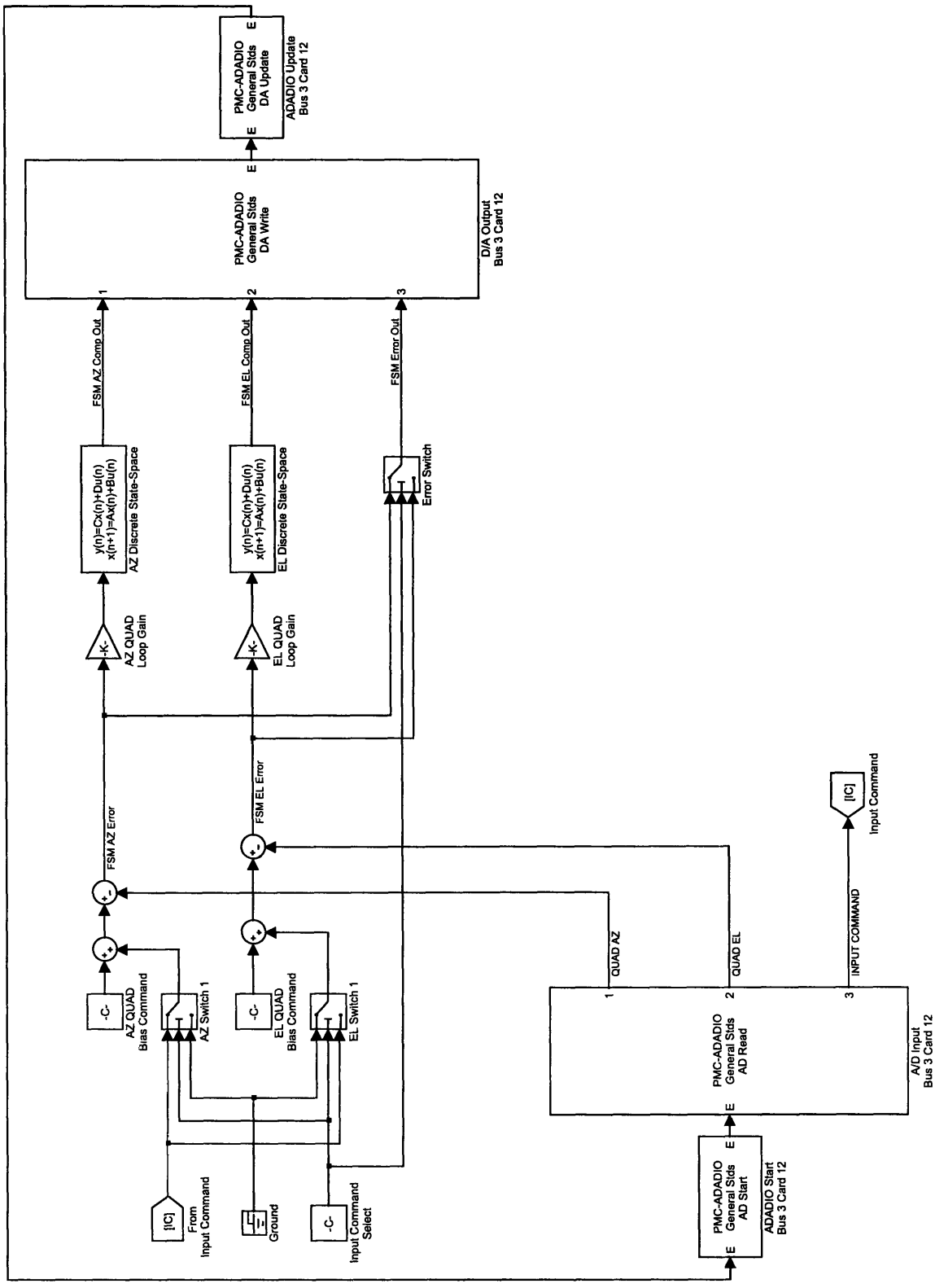

Figure 5-12: Simulink model of the digital compensator.

cretization of these matrices is performed by MATLAB using the c2d function with the zero-order hold option. The zero-order hold option was chosen to reduce computation time, but the other methods can be used and remain untested. These parameters are then loaded into the xPC Target machine when the model is built. The model built on the xPC Target machine is ran as fast as possible to minimize the half sample phase delay characteristics of digital controllers. xPC can run in one of two modes. In the normal mode, the xPC Host has control over the xPC Target machine, allowing real time access to model parameters. Unfortunately, in this configuration, the digital compensator will not run much faster than 20 kHz. At this sampling rate, the phase delay is 9° at 1 kHz and 13.5° at 1.5 kHz. Having this much extra phase delay would greatly impact the design of the compensator. Fortunately, xPC has a high speed mode known as 'polling mode'. In polling mode, the xPC Host has no control over the xPC Target Machine. To run as efficiently as possible, the xPC Target machine will not allow the model to be altered in real time. Thus, once a compensator has been tested in non polling mode, data is gathered using polling mode. Sometimes, the additional phase delay caused by non-polling mode will cause the system to become unstable, even though the system will work in polling mode. To address problems like this, the compensators are tested in short time bursts to verify that they are working properly. In polling mode, the controller can run at about 45 kHz, reducing the phase delay by over a factor of 2. The digital compensator will only be used for quad cell feedback. Including the A/D and D/A channels necessary for KAMAN control would have roughly doubled the necessary computation time and forced the xPC Target machine to run at half the sampling rate, introducing additional phase delay.

Figure 5-13 shows the xPC block diagram of the digital controller including a rotation matrix for the quad cell inputs. This matrix is not designed to perform the full 45° rotation (although it could, but the C/R board still handles this main rotation). Instead, this rotation matrix corrects small errors in the orientation of the quad cell. In practice, there was up to 7° extra degrees of rotation present. If left unchecked, this extra rotation would couple the axes of the FSM together (the

FSM would see movement in two axes and try to stop it, even if the movement was really only occurring in one axis. This rotation matrix corrects these errors for better performance.

Figure 5-14 is the xPC block diagram of the digital controller when used to integrate rate feedback into the system (see Section 5.1.2). The model differs from Figure 5-13 only by the inclusion of two differentiators and the appropriate summing junctions.

5.2.2 MATLAB Graphical User Interface

Figure 2-26 shows the graphical user interface used to control the digital compensator. The 'FSM Model Control' panels allows the changing of the current Simulink model, as well as controls for building the model on the xPC Target machine and starting and stoping the controller. The 'Build' button also serves to initialize the state-space matrices used in the model. The 'FSM Compensator Control' panel offers very basic functionality, allowing the command voltages to be changed, as well as the loop gains. The 'Input Command Select' group selects which axis will get commands from the third A/D channel (used for taking frequency responses). The 'Crossover' boxes do not accept inputs; these boxes display the approximate crossover frequency of the current model with the current loop gain.

5.2.3 Design Methodology

The design methodology used is an iteration method. The idea is to start with the most basic compensator possible to have a point of reference, and modify this initial controller until the performance has been improved as much as possible. The goal is to have the crossover of the negative loop transmission be as high as possible. Looking at the open loop frequency response plots, it becomes apparent that at the very least, a lead compensator will have to be used to bring the phase up at crossover. This is the starting point of the design. Hopefully, after all of the iterations from starting at this point, the final result will be a negative loop transmission plot that crosses over

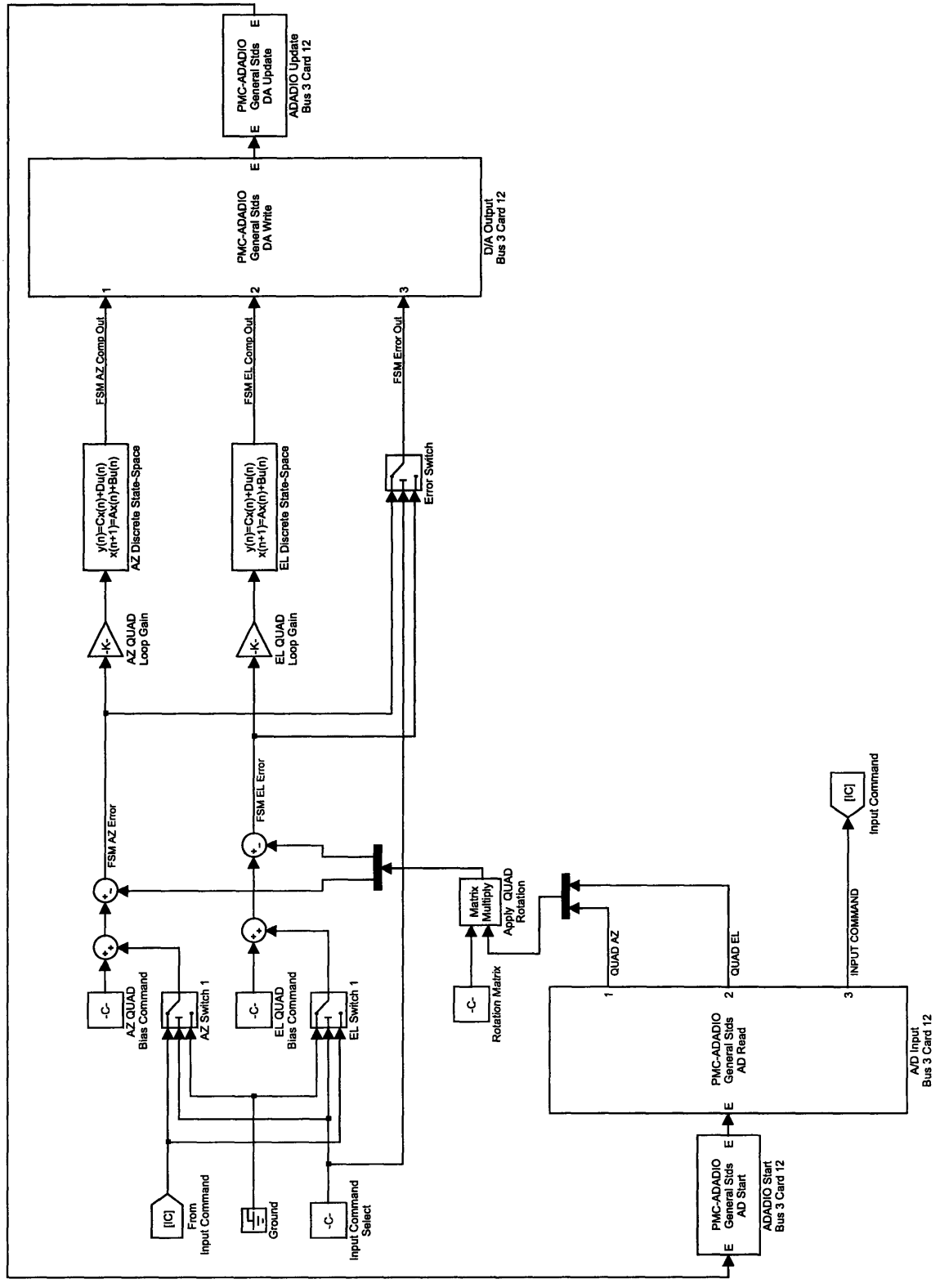

Figure 5-13: Simulink model of the digital compensator with the quad cell rotation stage.

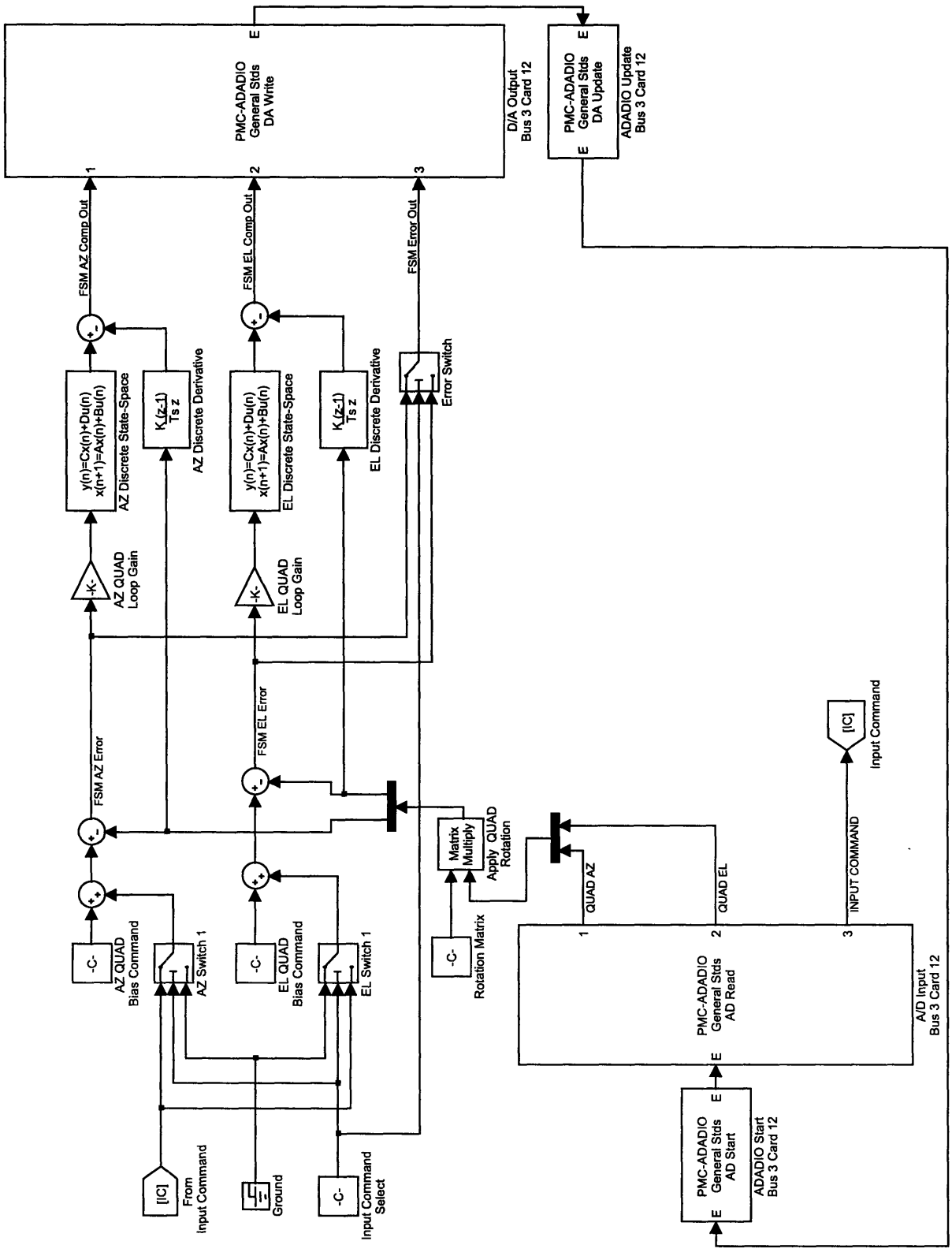

Figure 5-14: Simulink model of the digital compensator with quad cell rotation stage and integrated rate feedback.

beyond 1 kHz.

Before beginning the simulations, it is imperative to have a model of the hardware as that is as accurate as possible in the frequency ranges of interest. Models were fit to each FSM axis in quad cell mode in Section 3.4. Unfortunately, using the digital controller adds phase delay to the model. The electronics are also slightly different, and these differences need to be taken into account. Figure 5-15 shows the model used for the Azimuth axis under quad cell control with the digital phase delay incorporated. This model was fit using data taken with the digital electronics.

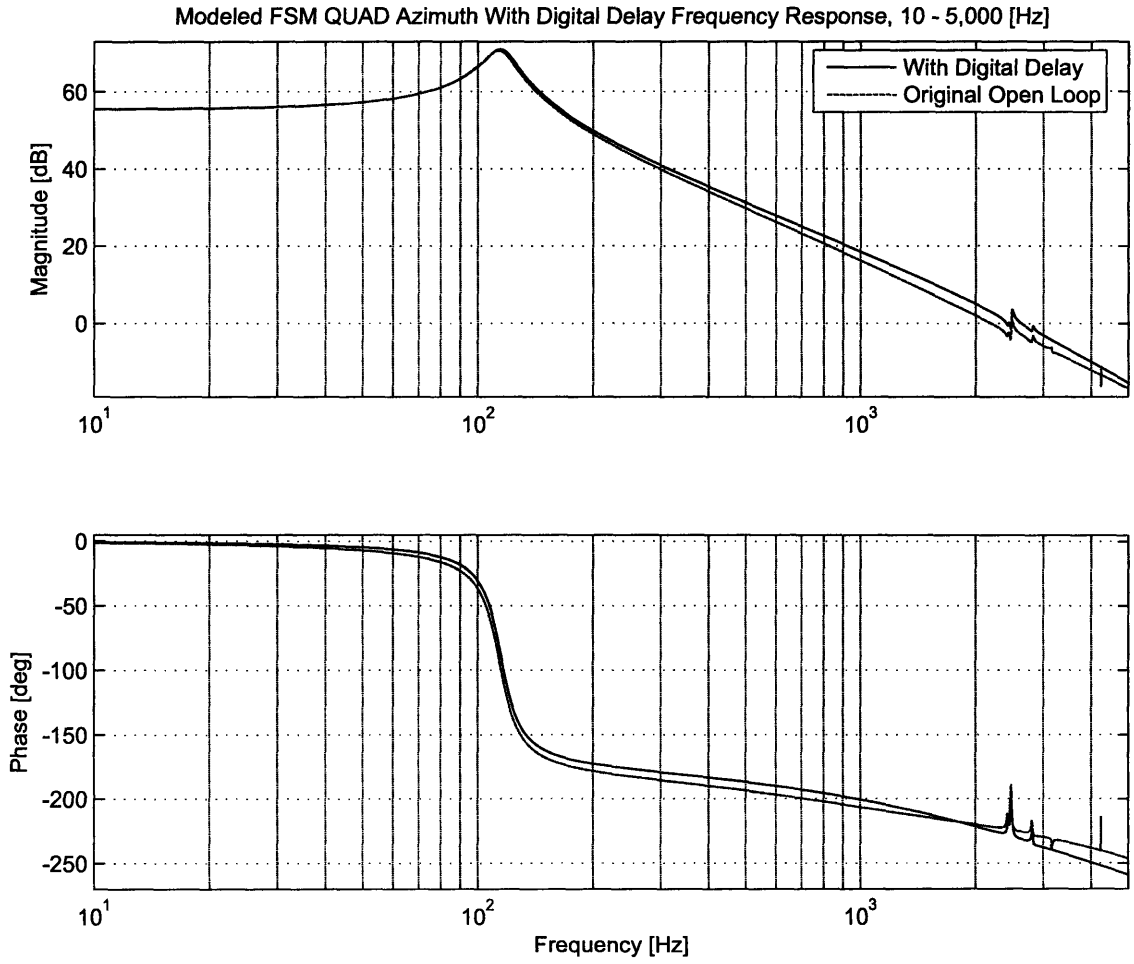

Figure 5-15: Modeled FSM quad cell azimuth axis with incorporated digital phase delay shown with original open loop model.

Finally, compensators will be designed with both the original mirror and the

mirror with rate feedback in mind (see Figure 5-3). It is of interest to consider the rate feedback system because of the phase gain near the desired crossover frequencies. Figure 5-16 shows the phase gained between 900 Hz and 3 kHz when rate feedback is used, compared to the original model. If the crossover frequency is 1.2 kHz, the rate feedback system will have about 10° more phase margin compared to the mirror without rate feedback.

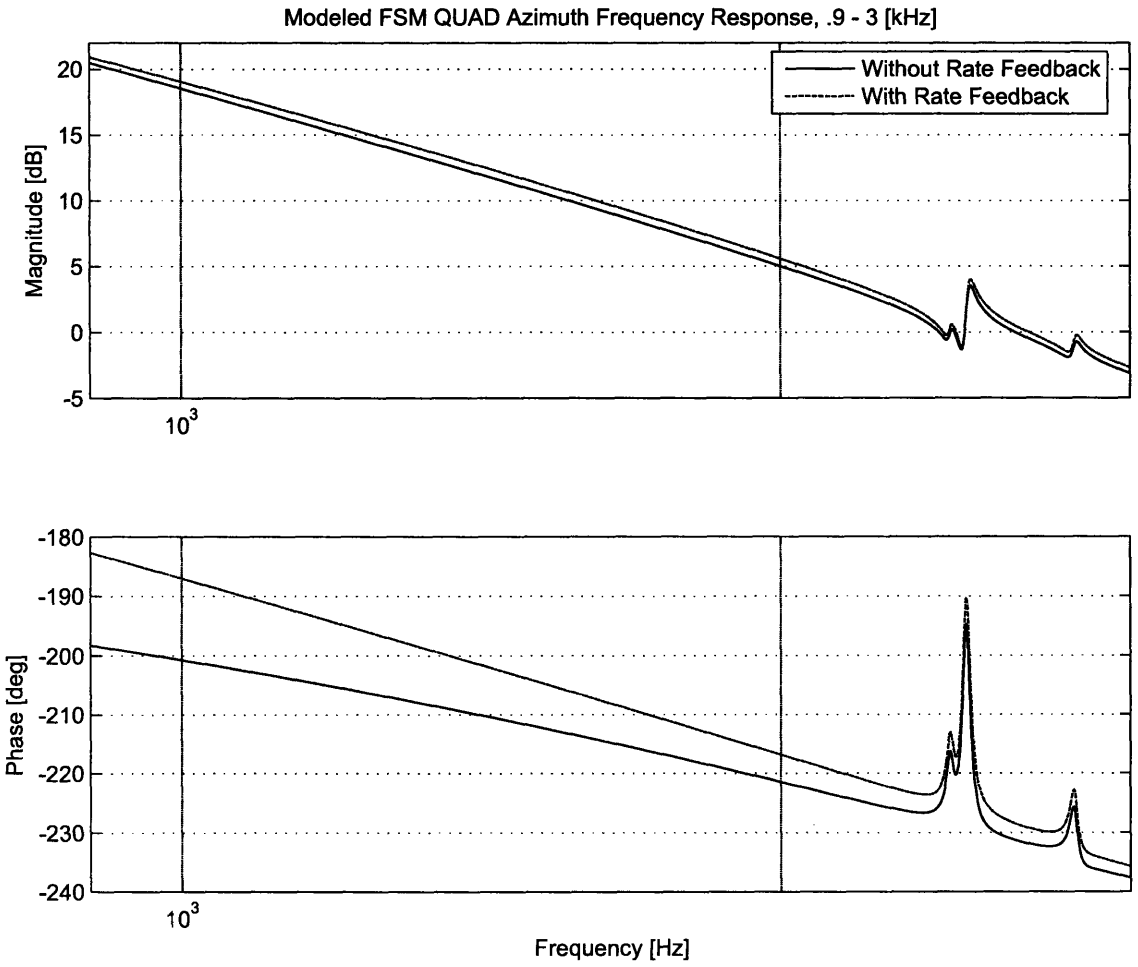

Figure 5-16: comparison of the FSM quad cell Azimuth axis modeled open loop frequency responses with and without rate feedback near 1 kHz.

5.2.4 Digital Compensators

This section contains the transfer functions of each digital compensator along with their simulated responses and the experimental data gathered during experimental testing.

Simple Lead Compensator

The first step in attacking this problem is getting the most basic controller to work. The first controller is a simple lead controller with the following transfer function.

$$G_{c1}(s) = K\frac{\alpha\tau s + 1}{\tau s + 1} = K\frac{(2\pi \cdot 380)^{-1} s + 1}{(2\pi \cdot 3800)^{-1} s + 1} \tag{5.4}$$

where K is the gain factor controlling the crossover frequency ωc, $1/\alpha\tau$ is the frequency of the zero and $1/\tau$ is the frequency of the pole. In this case, the frequency of the zero has be set at 380 Hz, and the frequency of the pole is one decade higher at 3800 Hz. Using an α value of 10 sets the maximum phase bump of the compensator to 55° at the logarithmic center between the zero and the pole, which in this case is approximately 1.2 kHz. Figure 5-17 shows the frequency response of $G_{c1}(s)$ with $K = .0535$.

Figure 5-18 shows the modeled Negative loop transmission of the Azimuth axis when $G_{c1}(s)$ is used. The values of K used are different between the normal model and the model with rate feedback. Values of K were chosen to push the crossover frequency f_c as far as possible, while maintaining a gain margin $G_m \geq 6$ dB and a phase margin $\Phi_m \geq 30°$. Usually, this meant that f_c was very near the maximum phase bump provided by $G_{c1}(s)$. Figure 5-19 shows the modeled Negative loop transmission near f_c.

Figure 5-19 is helpful in pointing out where the limitations of the compensator G_{c1} are. Looking at the loop transmission without rate feedback, f_c cannot be push up any higher because of the phase margin specification. The only way to push f_c will be to create a bigger phase bump with the lead compensator. This will have the effect of pushing up the amplitude of the first "doublet" peak near 2.4 kHz, and

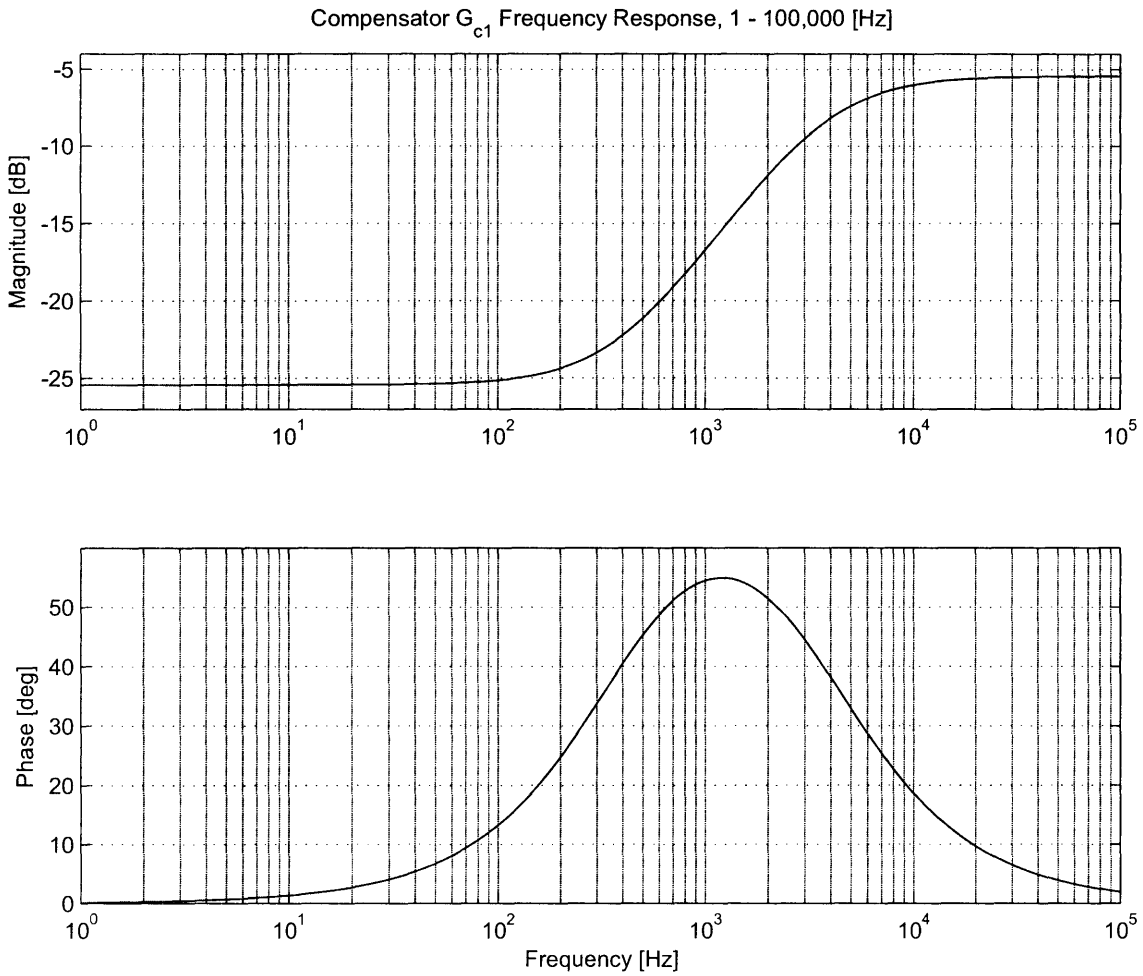

Figure 5-17: Modeled frequency response of $G_{c1}(s)$ with $K = .0535$, 1 Hz to 100 kHz.

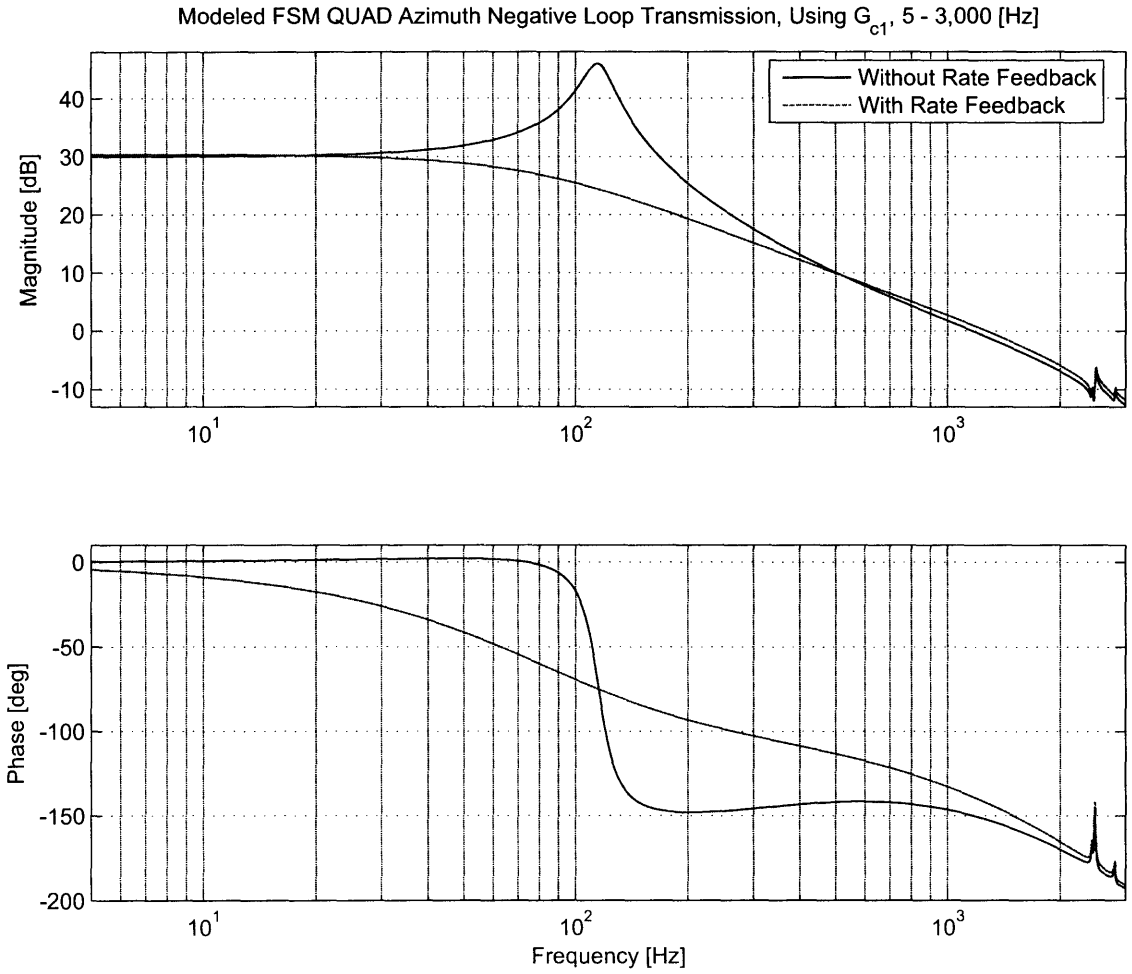

Figure 5-18: Modeled FSM quad cell Azimuth axis negative loop transmission using $G_{c1}(s)$, 5 Hz to 3 kHz.

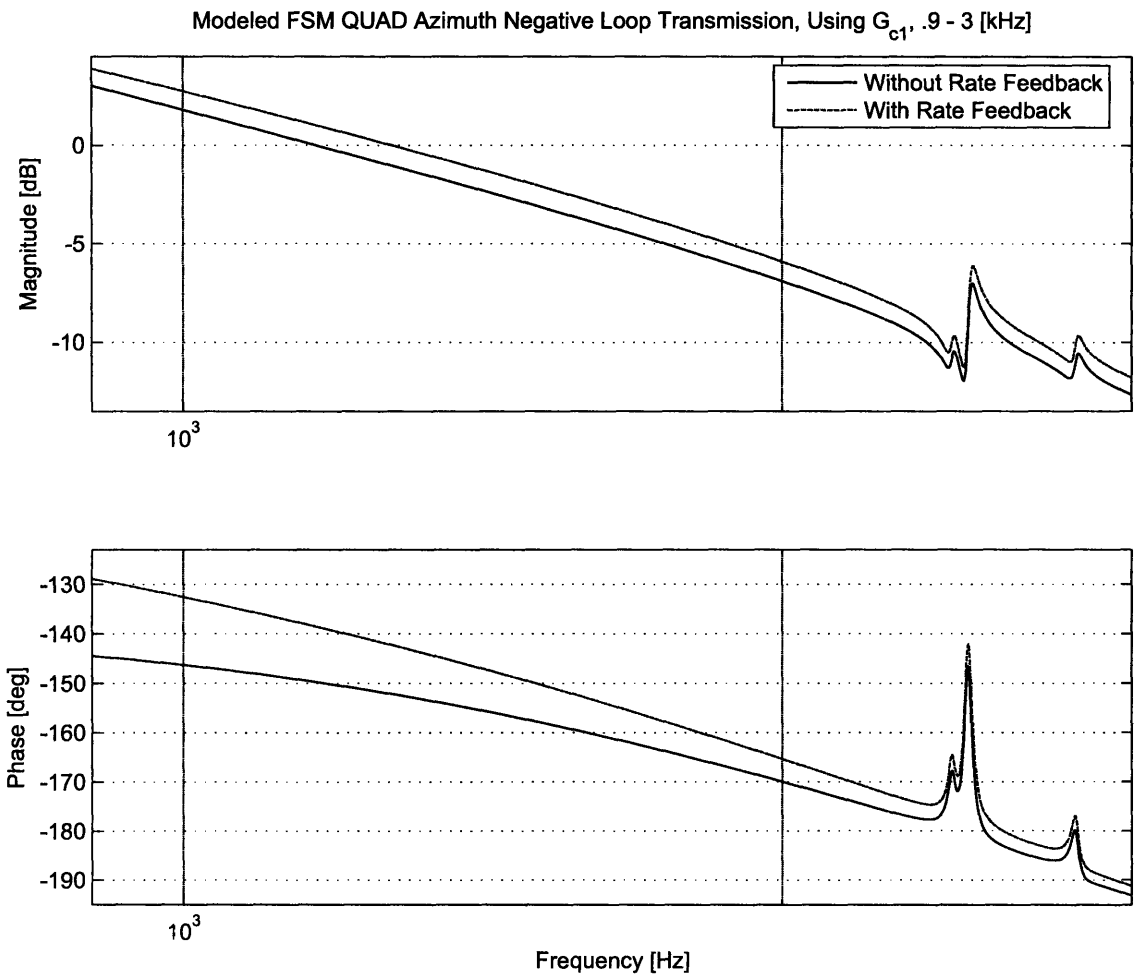

Figure 5-19: Modeled FSM quad cell azimuth axis negative loop transmission using $G_{c1}(s)$ near crossover.

thus must be done with care. Looking at the loop transmission with rate feedback used, the phase margin is not the limiting factor in pushing up f_c, because the open loop model with rate feedback has about 10° less phase loss compared to the open loop model without rate feedback. This allows f_c to be pushed up until the doublet peak hits -6 dB, running into the gain margin spec. To deal with this problem, it is possible to move the maximum phase bump of the compensator to the left of f_c, instead of crossing over in the exact middle of the phase bump. This has the effect of reducing the phase margin, but also knocking down the "doublet" peak, effectively trading phase margin for gain margin.

Figures 5-20 and 5-21 show the Nyquist diagram of the system. Because the open loop system is stable and there are no encirclements of the -1 point, the closed loop system will be stable. Figure 5-21 shows that for the system without rate feedback, the maximum closed loop peak will be just under 6 dB. The system with rate feedback will have a maximum closed loop peak about 1.5 dB less than this. Figure 5-22 shows the closed loop responses for both systems.

Looking at Figure 5-22, The closed loop -3 dB points are virtually identical for both systems with $\omega_{-3dB} \approx 2.08$ kHz. The rate feedback system has less peaking as predicted by the Nyquist diagram in Figure 5-21. However, both systems show lower than unity low frequency behavior because of the relatively low loop gain at low frequencies. Figure 5-23 shows the step response of this closed loop system.

Both systems in Figure 5-23 show overshooting responses. However, the overshoot percentage for the system without rate feedback is almost 60%, where the overshoot is about 30% on the system with rate feedback. Both systems have similar settling times. Table 5.2 summarizes the performance of each system using $G_{c1}(s)$ as the feedback compensator.

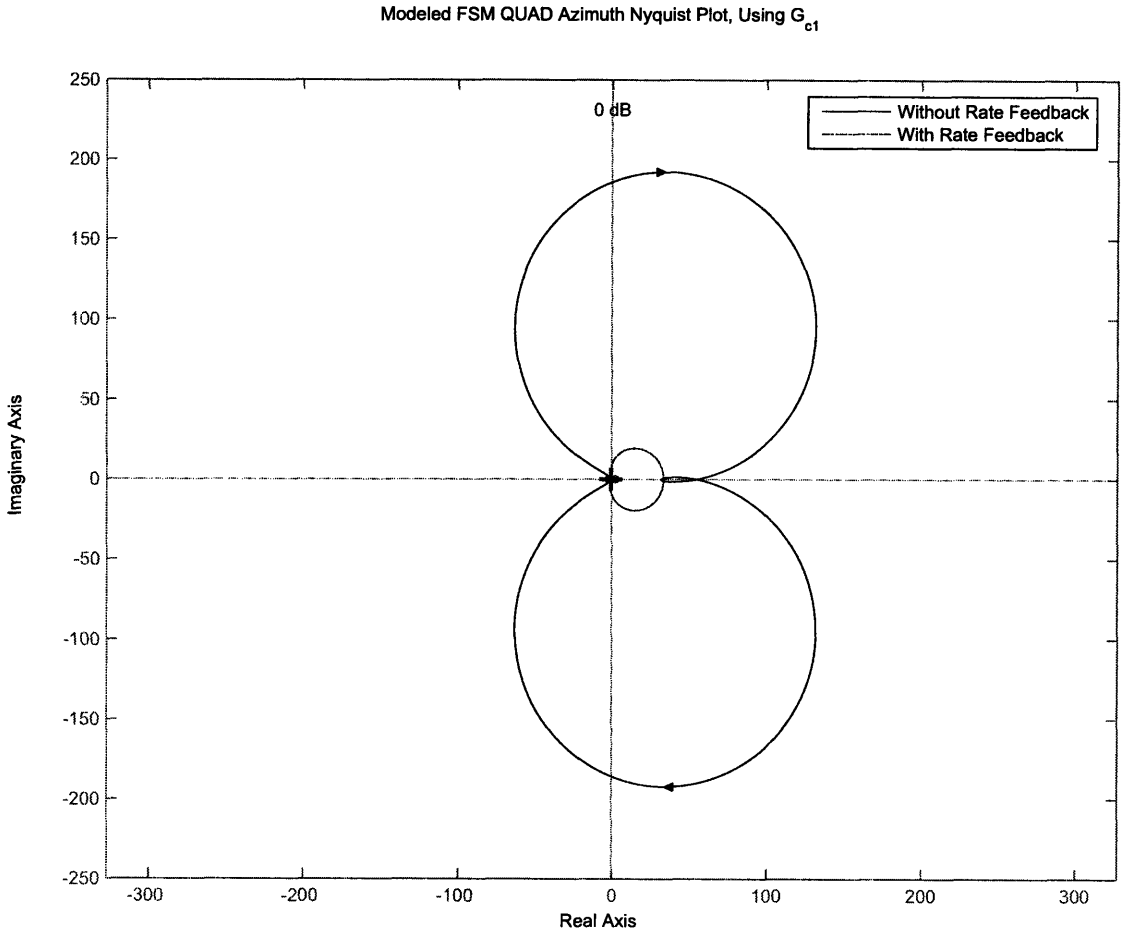

Figure 5-20: Modeled FSM quad cell Azimuth axis Nyquist diagram using $G_{c1}(s)$.

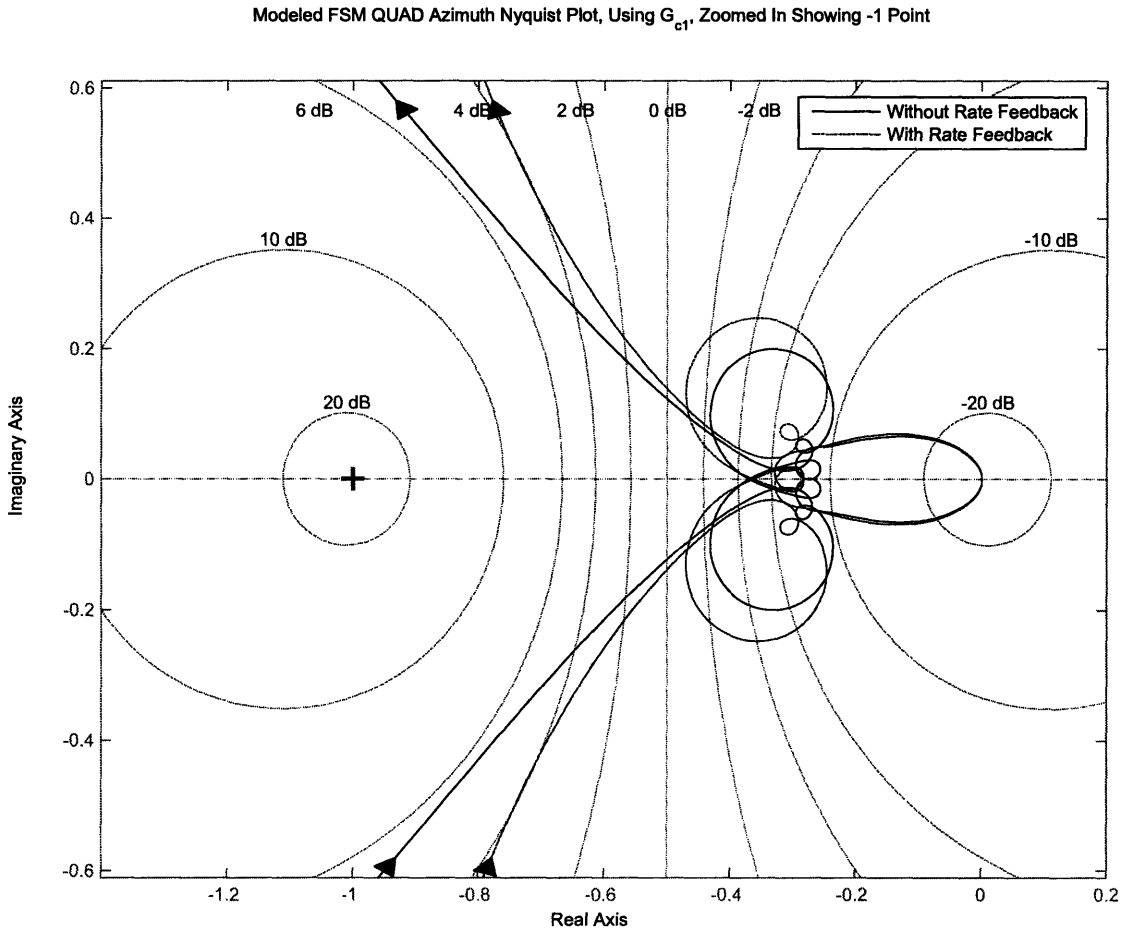

Figure 5-21: Modeled FSM quad cell Azimuth axis Nyquist diagram using $G_{c1}(s)$ near the -1 point.

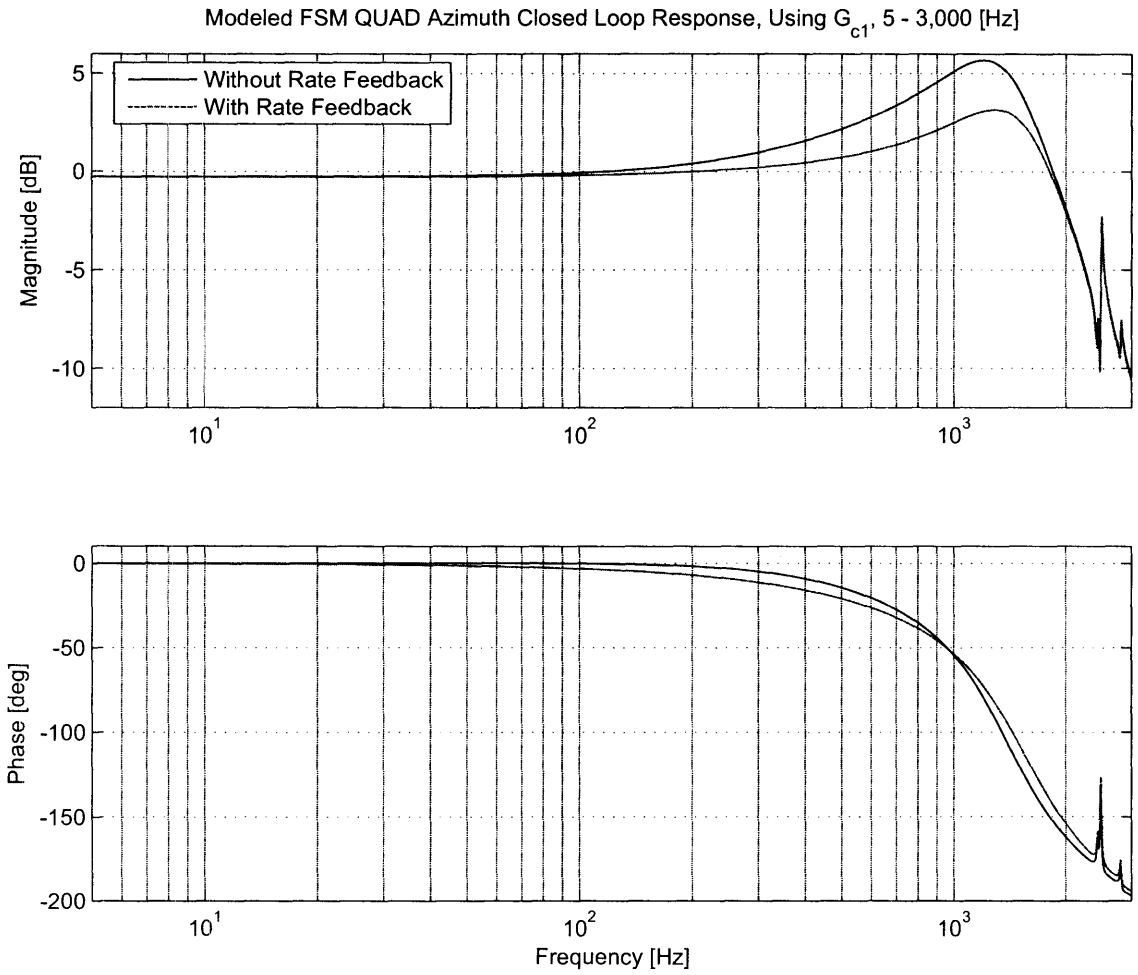

Figure 5-22: Modeled FSM quad cell Azimuth axis closed loop frequency response using $G_{c1}(s)$.

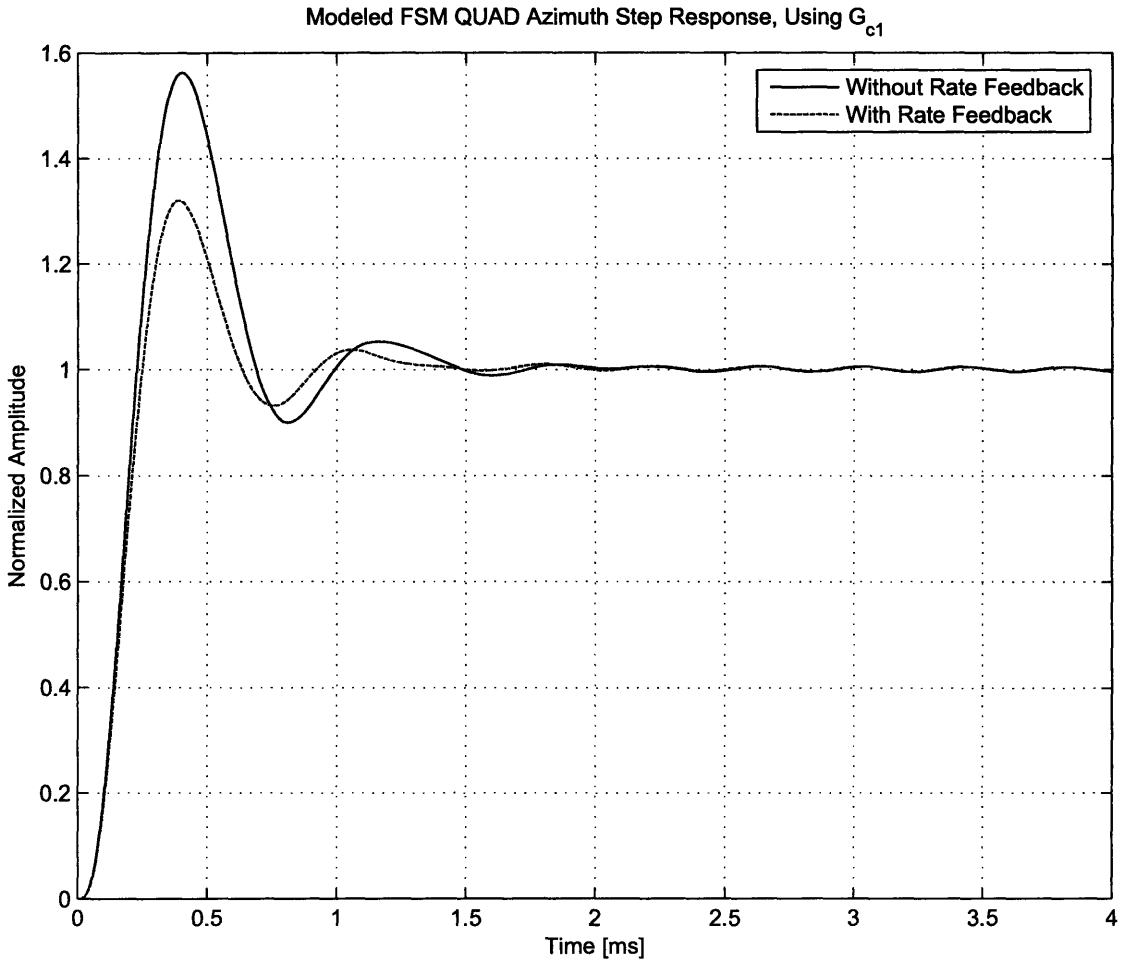

Figure 5-23: Modeled FSM quad cell Azimuth axis step response using $G_{c1}(s)$.

Parameter	Without RFB	With RFB
K	0.0535	0.0562
f_c [kHz]	1.17	1.27
G_m [dB]	7	6.1
Φ_m [deg]	30	38
$D_r(s)$ @ 10 Hz [dB]	30	30
-3 dB Frequency [kHz]	2.07	2.09

Table 5.2: Modeled performance specifications of the FSM quad cell Azimuth axis using $G_{c1}(s)$, both with and without rate feedback.

Lead With Low Frequency Integrator And Increased α

The simple lead compensator of the previous section is a good place to start. This next compensator focuses on getting the most performance out of the mirror without using rate feedback. As mentioned in the last section, $G_{c1}(s)$ did not produce enough of a phase bump to push the crossover frequency any higher. This can be fixed by increasing the parameter α of Equation 5.4. To increase the maximum phase bump to about 60°, $\alpha = 15.85$. Keeping the phase bump centered at 1.2 kHz, the lead compensation function becomes

$$L(s) = K\frac{\alpha\tau s + 1}{\tau s + 1} = K\frac{(2\pi \cdot 302)^{-1} s + 1}{(2\pi \cdot 4786)^{-1} s + 1} \tag{5.5}$$

where K is the gain parameter that controls the crossover frequency f_c This transfer function can be used to produce similar results to $G_{c1}(s)$, except with more phase margin and a corresponding decline in gain margin (the first "doublet peak" is brought up higher in magnitude as the pole of the lead compensator is pushed higher in frequency). However, the low frequency performance of $G_{c1}(s)$ is unsatisfactory. Ideally, there should be one or more integrators added at low frequencies to increase disturbance rejection and eliminate steady state errors. One integrator of the form

$$I(s) = \frac{s + 2\pi f}{s} \tag{5.6}$$

is added to the lead compensator. $I(s)$ is a pole at the origin with a zero at f Hz. $I(s)$ is multiplied with the lead compensator $L(s)$ to produce the overall compensator $G_{c2}(s)$.

$$G_{c2}(s) = K\left(\frac{s+2\pi f}{s}\right)\left(\frac{\alpha\tau s+1}{\tau s+1}\right) = K\left(\frac{s+2\pi\cdot 50}{s}\right)\left(\frac{(2\pi\cdot 302)^{-1}s+1}{(2\pi\cdot 4786)^{-1}s+1}\right) \tag{5.7}$$

The gain is set to $K = .0435$ to set $f_c = 1.19$ kHz. The frequency response for the compensator $G_{c2}(s)$ is shown in Figure 5-24. The performance of the compensator is

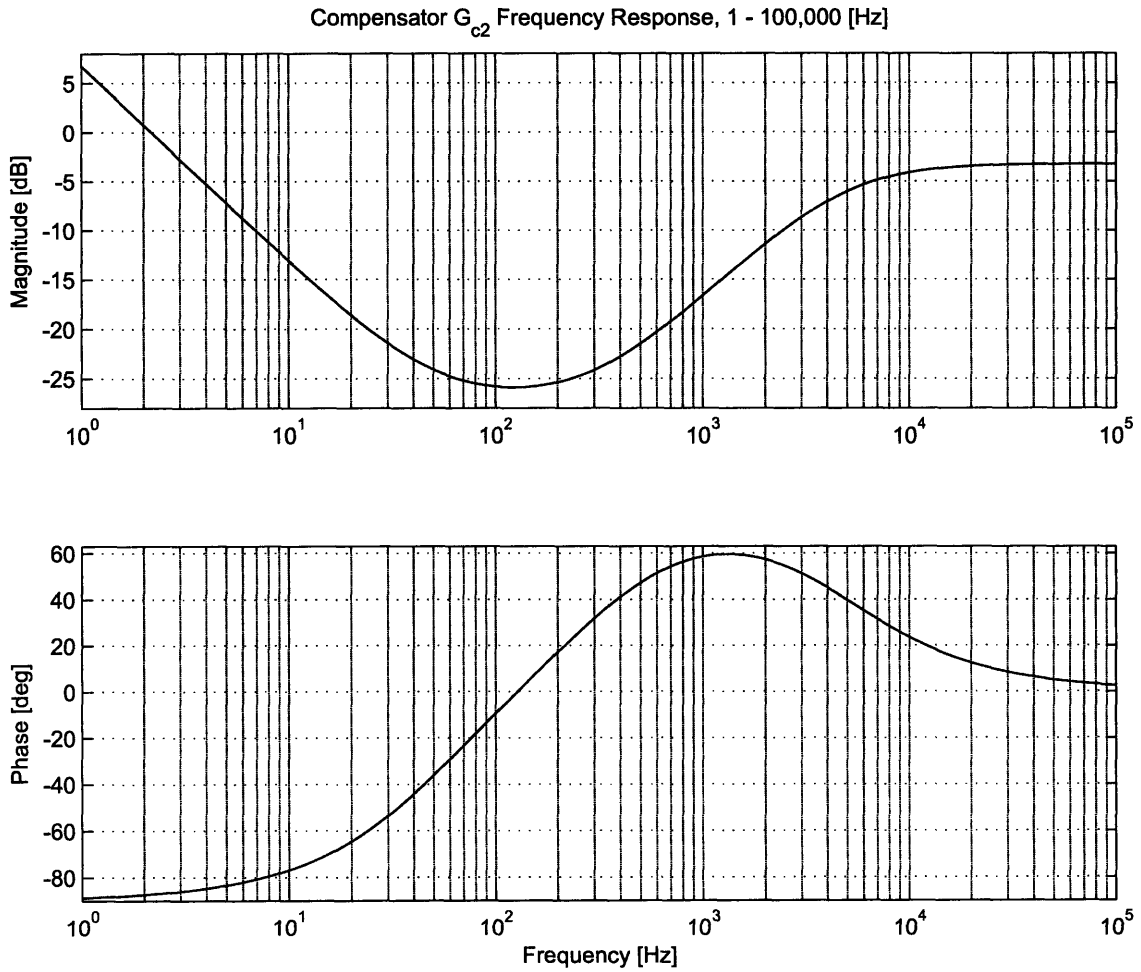

Figure 5-24: Modeled frequency response of the digital compensator $G_{c2}(s)$.

best revealed by examining the negative loop transmission, plotted in Figures 5-25

and 5-26.

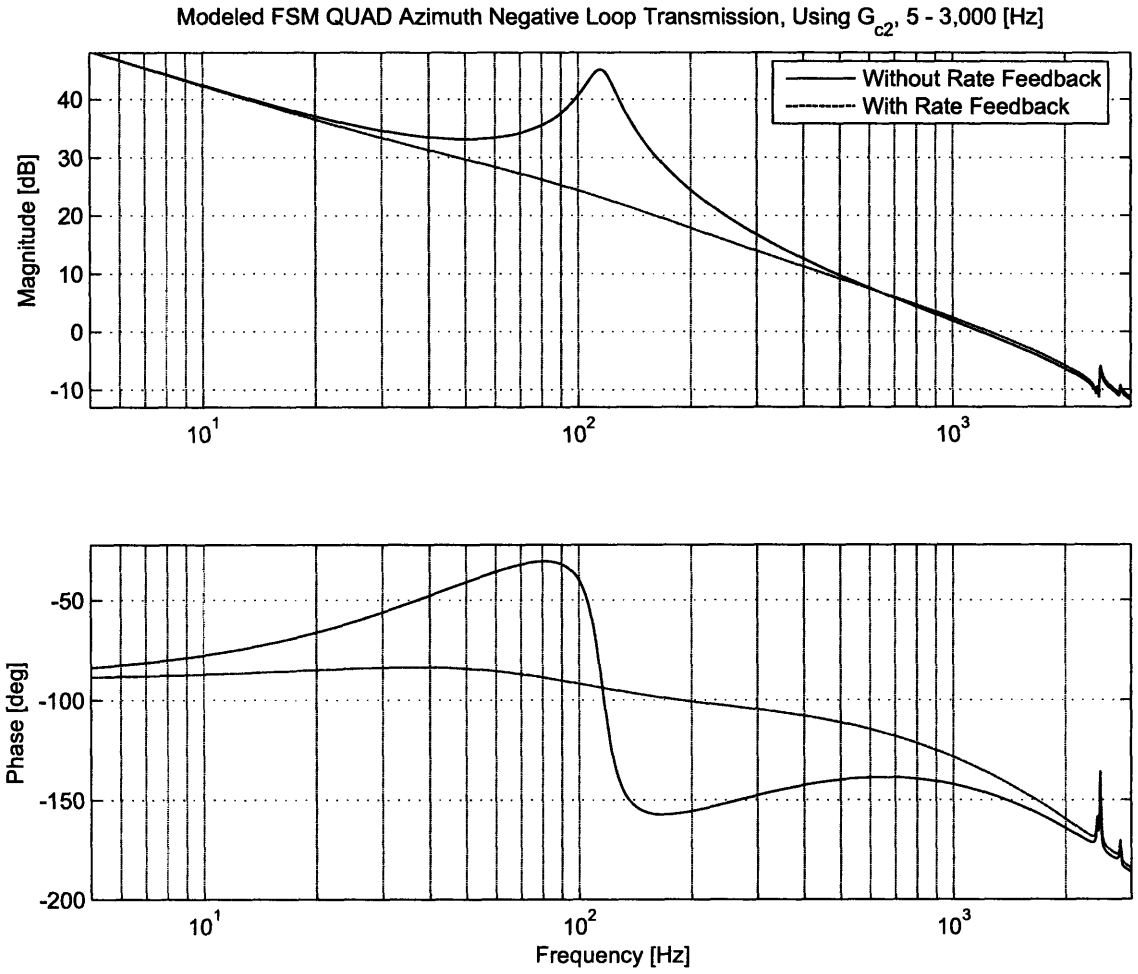

Figure 5-25: Modeled FSM quad cell Azimuth axis negative loop transmission using $G_{c2}(s)$ with $K = .0435$.

The negative loop transmissions in Figure 5-25 provide great insight on the performance of the compensator. This compensator was designed to be used for the mirror without rate feedback, so these plots will be ignored until the next section. The negative loop transmission of the Azimuth axis appears as one would expect, with the integrator present at the lower frequencies, with peaking in the loop transmission at the natural frequency of the axis, then rolling off again with a -1 slope until after crossover. It is important to note that the stability of the system depends on the crossover frequency f_c. If crossover occurs between approximately 150 - 200 Hz, the

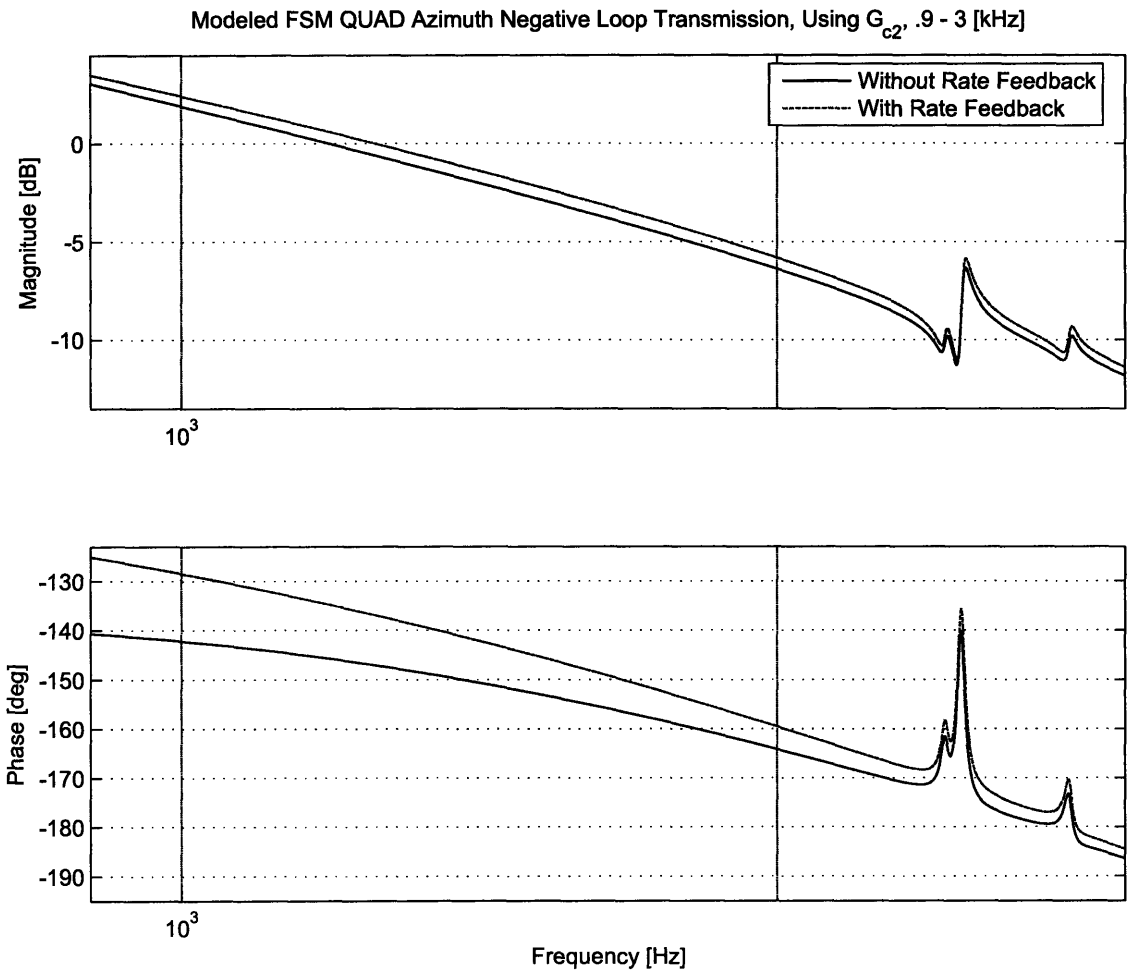

Figure 5-26: Modeled FSM quad cell Azimuth axis negative loop transmission near crossover using $G_{c2}(s)$ with $K = .0435$.

phase margin of the system will be less than 30°. Figure 5-26 shows the negative loop transmission near the crossover frequencies. The system has a crossover frequency of approximately 1.2 kHz, phase margin $\Phi_m \approx 34°$ and a gain margin $G_m = 6.2$ dB. Because of the high loop gain, the disturbance rejection is equal to the negative loop transmission, and in this case is over 42 dB at 10 Hz, which is on the same order as the original analog compensator. This compensator is reaching the limit of the specifications set in the beginning of the chapter. Crossing over at a higher frequency is not possible with this controller, as the gain margin would become too low. Moving the position of the lead phase bump is not advisable because too much phase margin would be lost. This compensator pushes the FSM as far as it will go without using rate feedback. Figures 5-27 and 5-28 show the Nyquist diagram for this system.

There are no encirclements of the -1 Point, which means that the system will be closed loop stable. The maximum peaking of the closed loop system should be just under 5 dB. Figure 5-29 shows the modeled closed loop response of the system and Figure 5-30 shows the modeled step response of the system. The closed loop frequency response shown in Figure 5-29 is very similar to the closed loop response of the system when using $G_{c1}(s)$ as shown in Figure 5-22. Their are several small differences between the plots. The low frequency values of the closed loop response are set an unity when $G_{c2}(s)$ is used. $G_{c1}(s)$ produces low frequency values that are below unity, implying steady state errors. Second, the maximum peak of the closed loop response is about 1 dB lower when $G_{c2}(s)$ is used as compared to $G_{c1}(s)$. This is because of the increased phase margin from the lead compensator of $G_{c2}(s)$. The step response of the system, shown in Figure 5-30, is also very similar to the step response when $G_{c1}(s)$ is used. However, the lag compensator used in $G_{c2}(s)$ introduces a long-tail transient into the step response that doesn't die out until about 13 ms. This long-tail transient introduces about 2% error until it dies out, which is acceptable given the increased low frequency disturbance rejection. Table 5.3 summarizes the predicted performance of $G_{c2}(s)$.

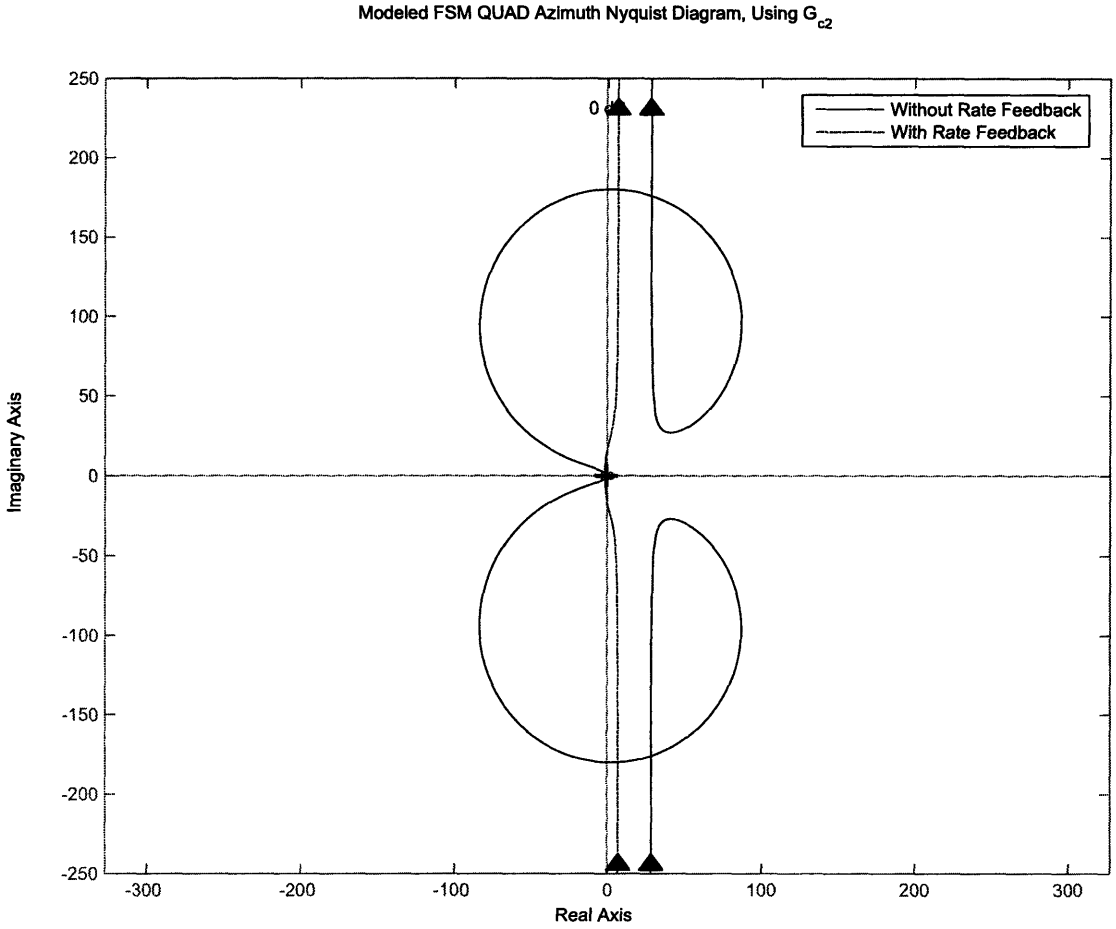

Figure 5-27: Modeled FSM quad cell Azimuth axis Nyquist diagram using $G_{c2}(s)$ with $K = .0435$.

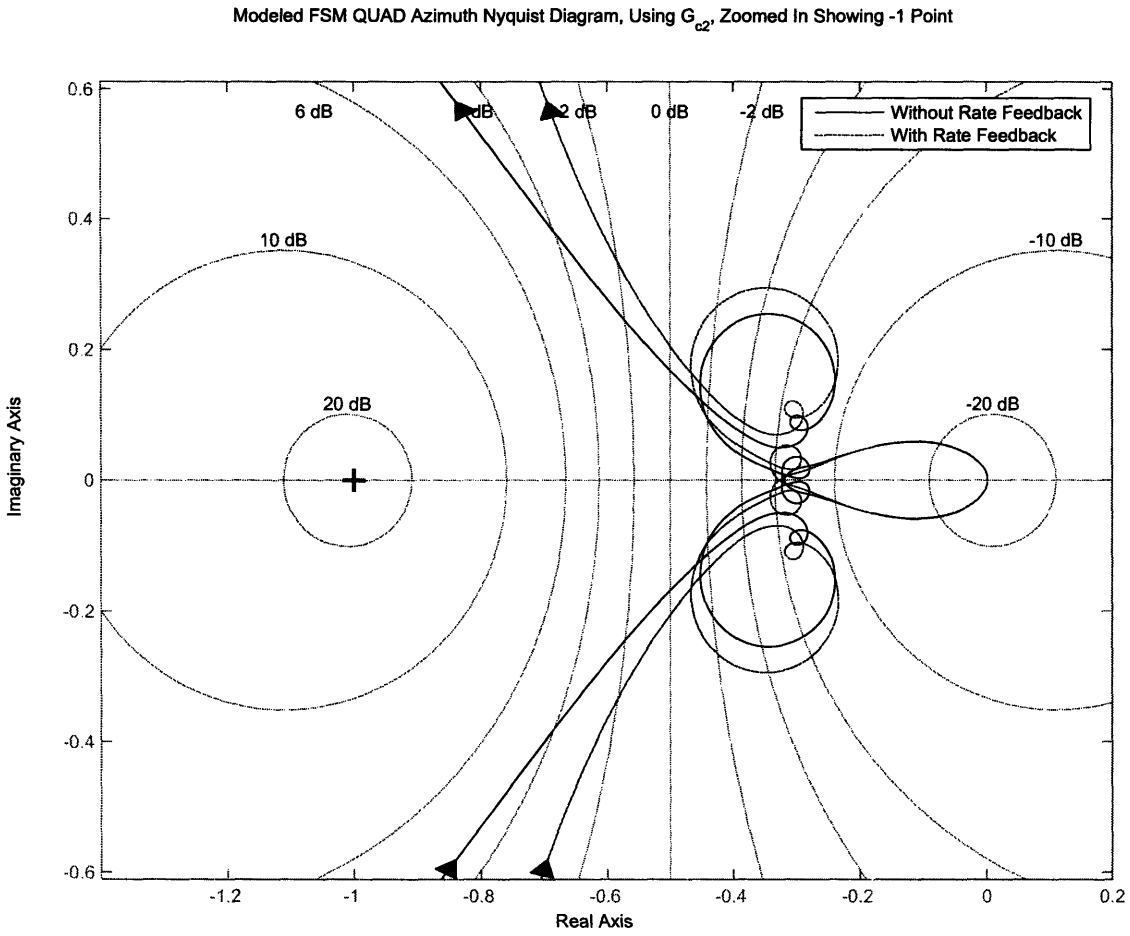

Figure 5-28: Modeled FSM quad cell Azimuth Axis Nyquist diagram near the -1 point using $G_{c2}(s)$ with $K = .0435$.

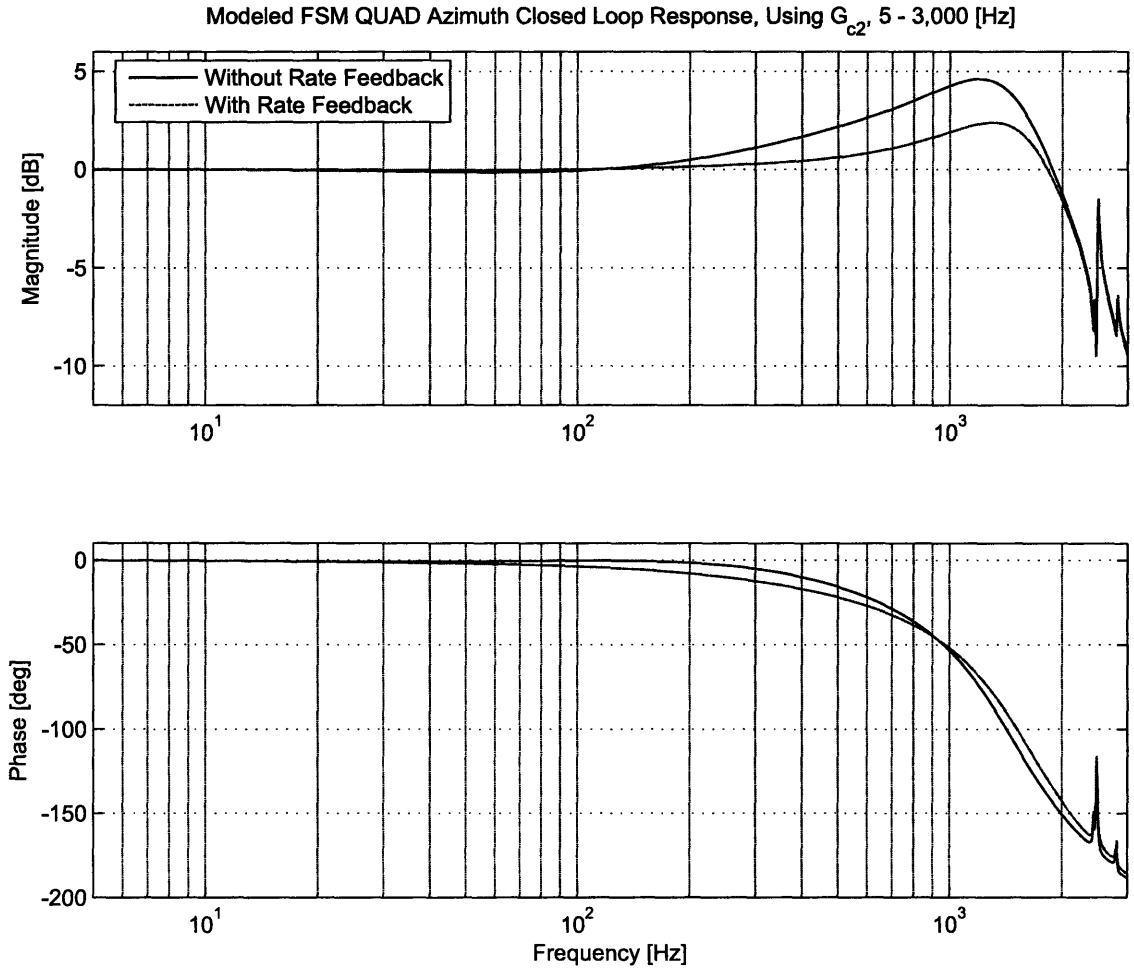

Figure 5-29: Modeled FSM quad cell Azimuth axis closed loop frequency response using $G_{c2}(s)$ with $K = .0435$.

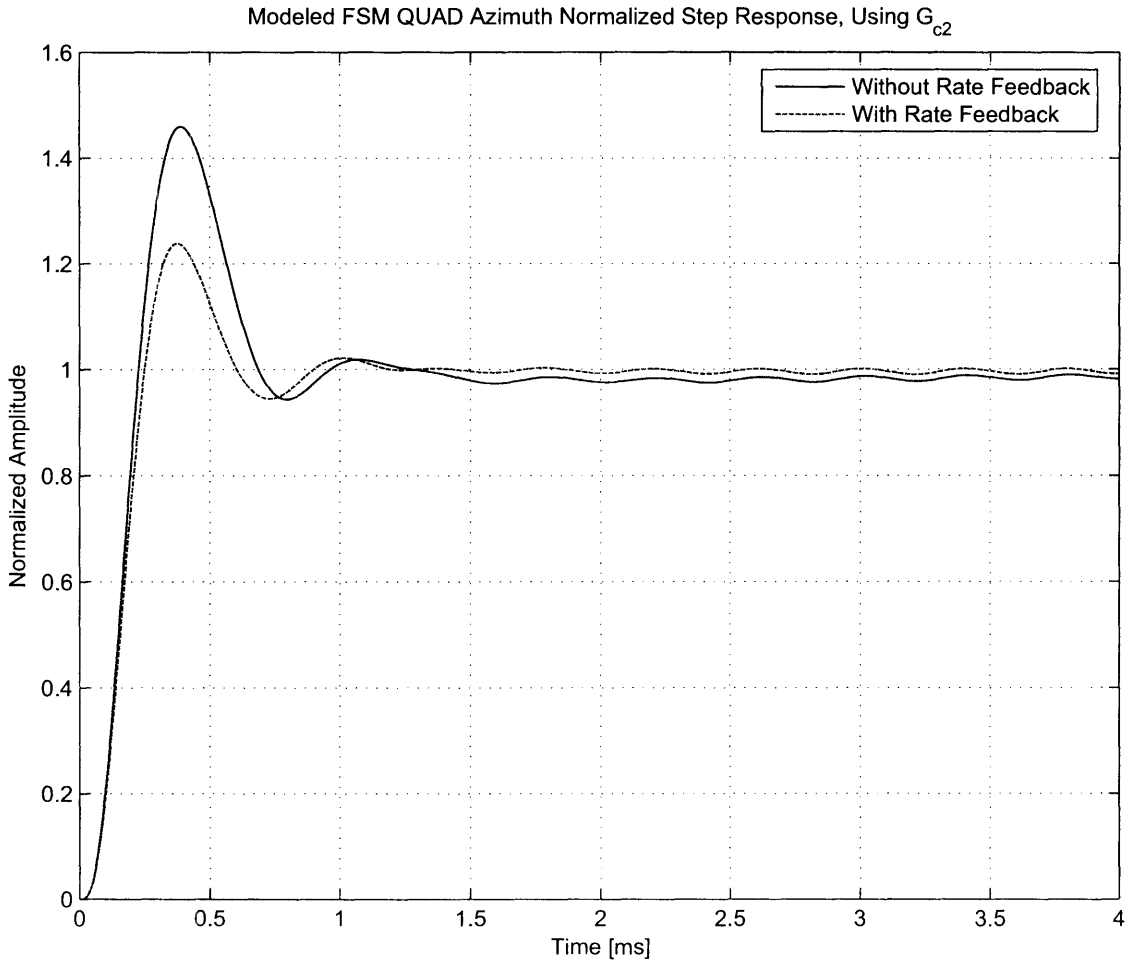

Figure 5-30: Modeled FSM quad cell Azimuth axis step response using $G_{c2}(s)$ with $K = .0435$.

Parameter	Without RFB	With RFB
K	0.0435	0.0435
f_c [kHz]	1.19	1.25
G_m [dB]	6.3	5.9
Φ_m [deg]	34	43
$D_r(s)$ @ 10 Hz [dB]	42	42
-3 dB Frequency [kHz]	2.16	2.13

Table 5.3: Modeled performance specifications of the FSM quad cell Azimuth axis using $G_{c2}(s)$, both with and without rate feedback.

Dual Lead With Low Frequency Integrator And Decreased α

$G_{c2}(s)$ was designed to provide the highest bandwidth without using rate feedback. However, rate feedback provides a substantial phase jump from the original system around the crossover frequencies. For this reason, a new compensator will be designed with rate feedback in mind. Using $G_{c2}(s)$ as a starting point and then manipulating to improve performance for the rate feedback system, the new compensator obtained is

$$G_{c3}(s) = K\left(\frac{s+2\pi\cdot 50}{s}\right)\left(\frac{(2\pi\cdot 250)^{-1}s+1}{(2\pi\cdot 2500)^{-1}s+1}\right)\left(\frac{(2\pi\cdot 10000)^{-1}s+1}{(2\pi\cdot 31623)^{-1}s+1}\right) \quad (5.8)$$

The second lead acts to boost the phase margin more than it subtracts gain margin. The frequency response of $G_{c3}(s)$ is shown in Figure 5-31.

The performance of the compensator is best revealed by examining the negative loop transmission, plotted in Figures 5-32 and 5-33.

The negative loop transmissions in Figure 5-32 provide great insight on the performance of the compensator. This compensator was designed to be used for the mirror with rate feedback, and the non rate feedback curves serve only as a reference. The negative loop transmission of the Azimuth axis appears very similar to the loop transmission of the original analog controller. Figure 5-33 shows the negative loop transmission near the crossover frequencies. The system has a phase margin $\Phi_m \approx 36°$

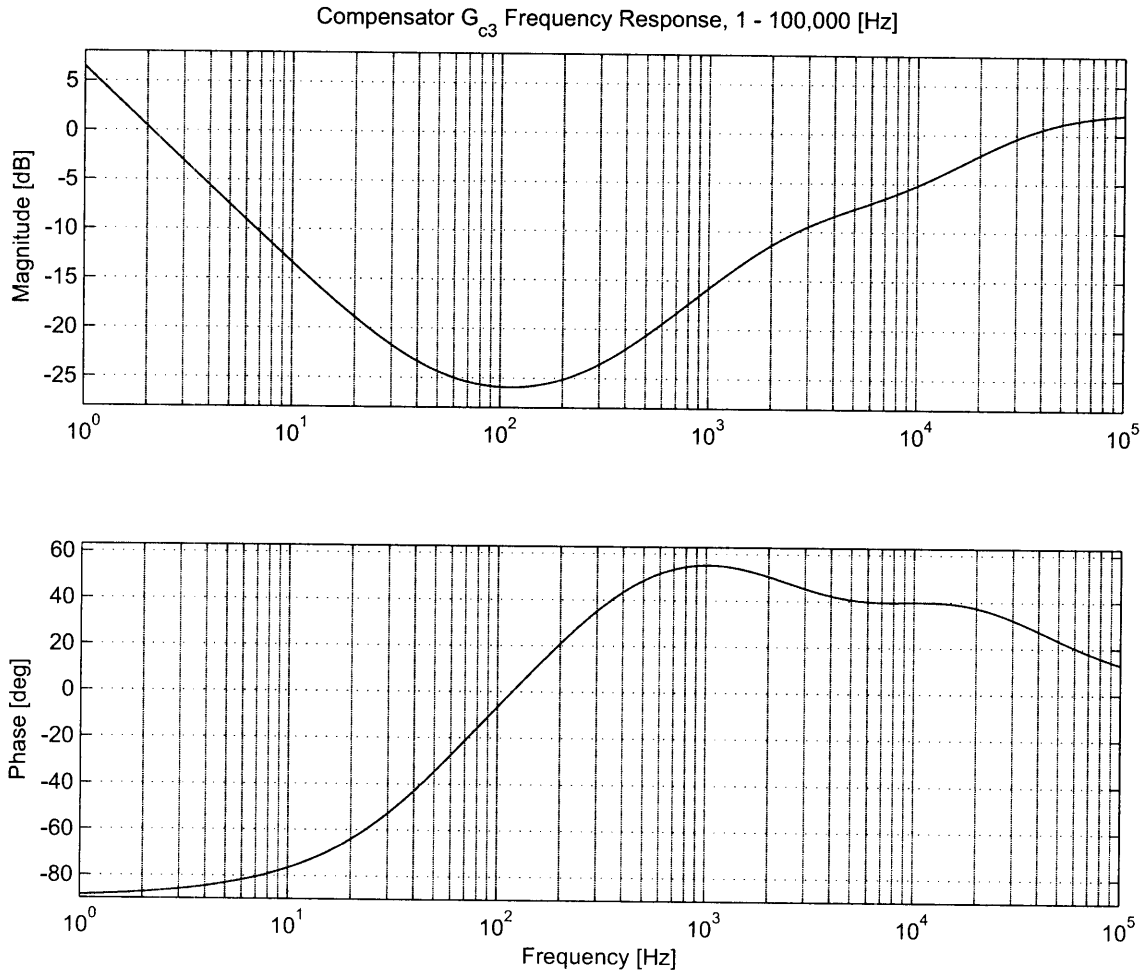

Figure 5-31: Modeled frequency response of the digital compensator $G_{c3}(s)$.

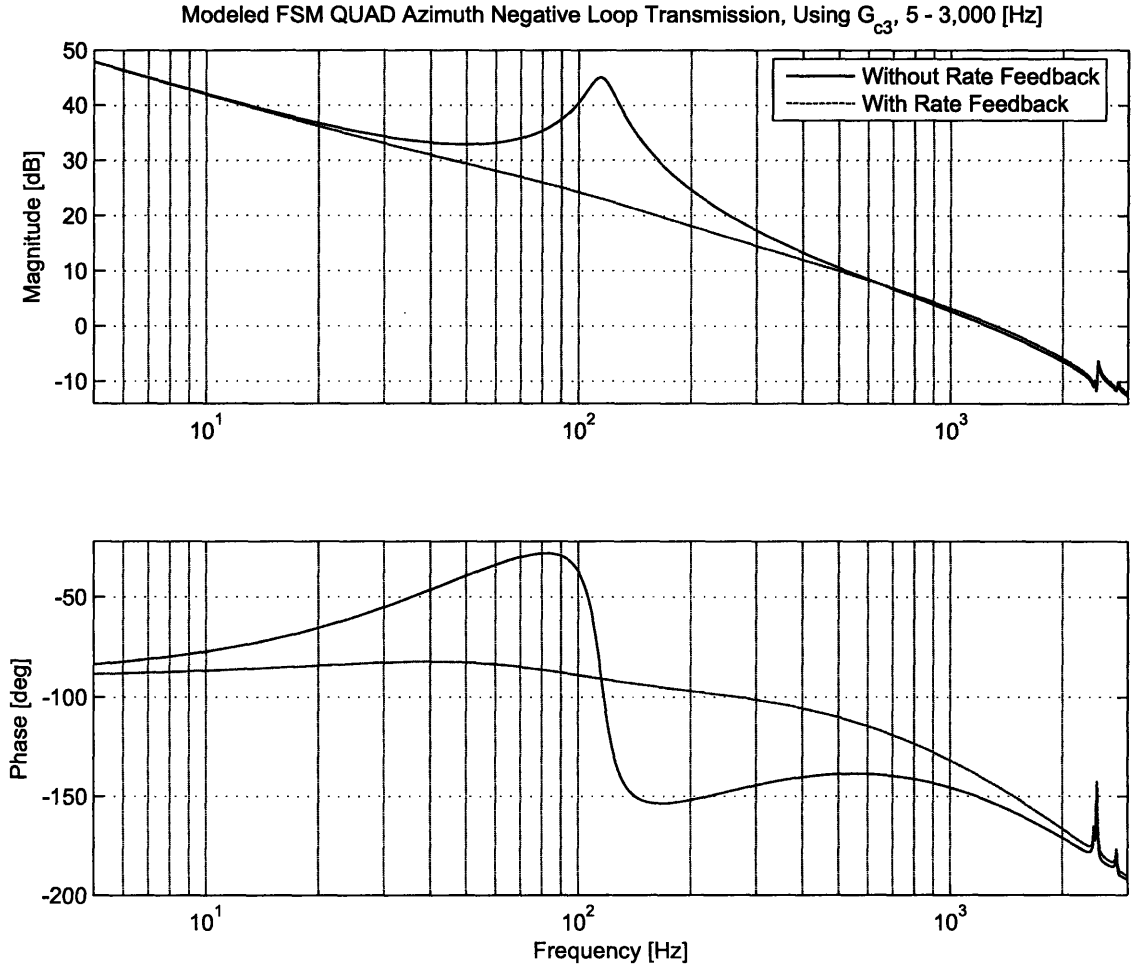

Figure 5-32: Modeled FSM quad cell Azimuth axis negative loop transmission using $G_{c3}(s)$ with $K = .043$.

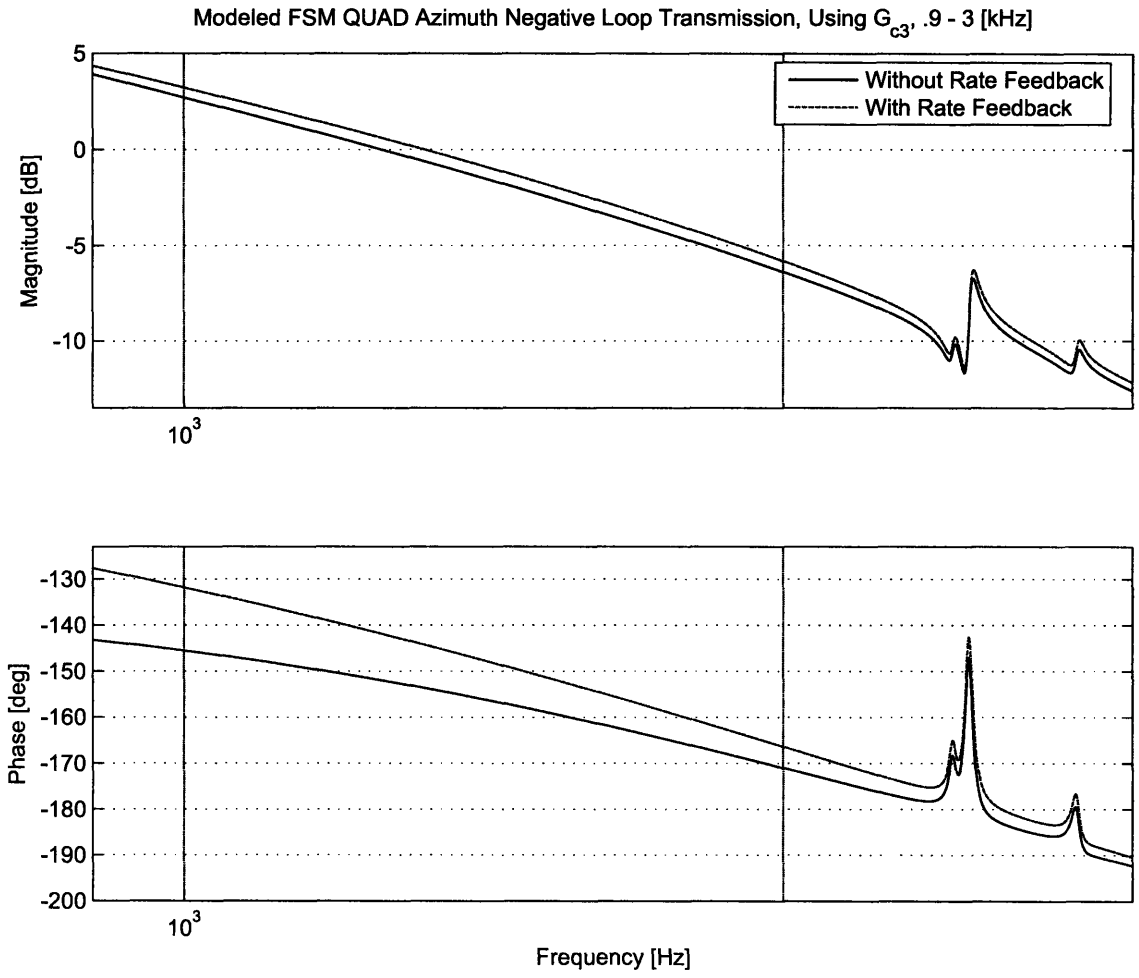

Figure 5-33: Modeled FSM quad cell Azimuth axis negative loop transmission near crossover using $G_{c3}(s)$ with $K = .043$.

and a gain margin $G_m = 6.2$ dB, crossing over at $f_c \approx 1.32$ kHz, a 32% improvement over the original analog compensator. Because of the high loop gain, the disturbance rejection is equal to the negative loop transmission, and in this case is over 42 dB at 10 Hz, which is on the same order as the original analog compensator. Figures 5-34 and 5-35 show the Nyquist diagram for this system. There are no encirclements

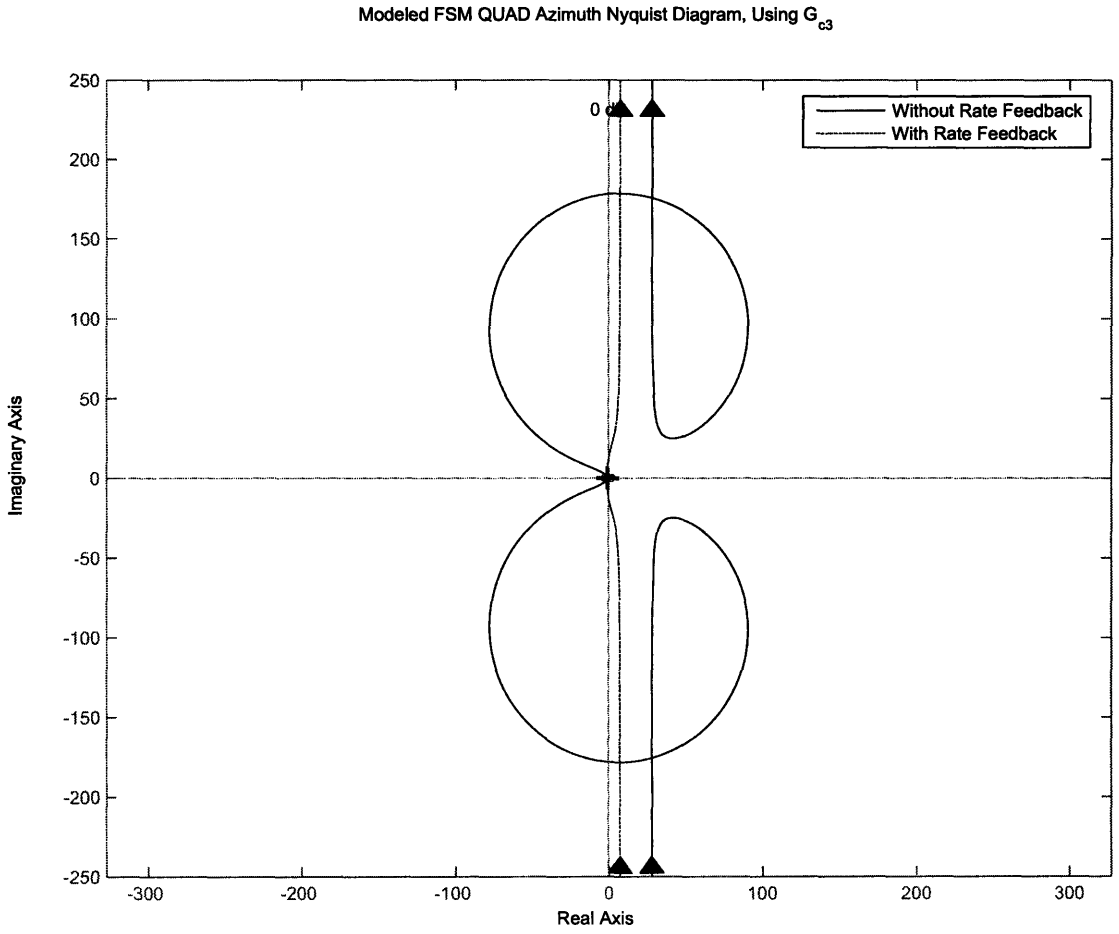

Figure 5-34: Modeled FSM quad cell Azimuth axis Nyquist diagram using $G_{c3}(s)$ with $K = .043$.

of the -1 Point, meaning that the system will be closed loop stable. The maximum peaking of the closed loop system should be just under 5 dB. Figure 5-36 shows the modeled closed loop response of the system and Figure 5-37 shows the modeled step response of the system. The closed loop frequency response shown in Figure 5-36 is

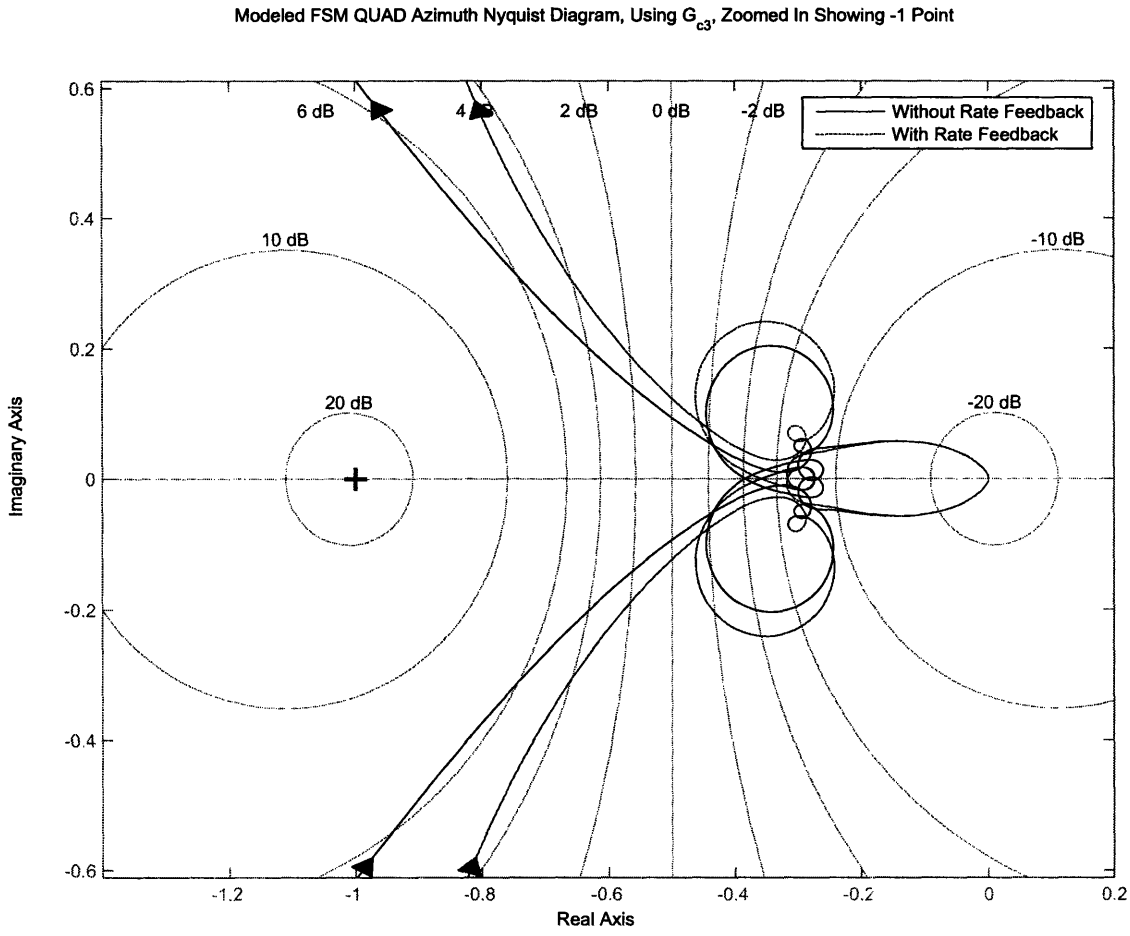

Figure 5-35: Modeled FSM quad cell Azimuth axis Nyquist diagram near the -1 point using $G_{c3}(s)$ with $K = .043$.

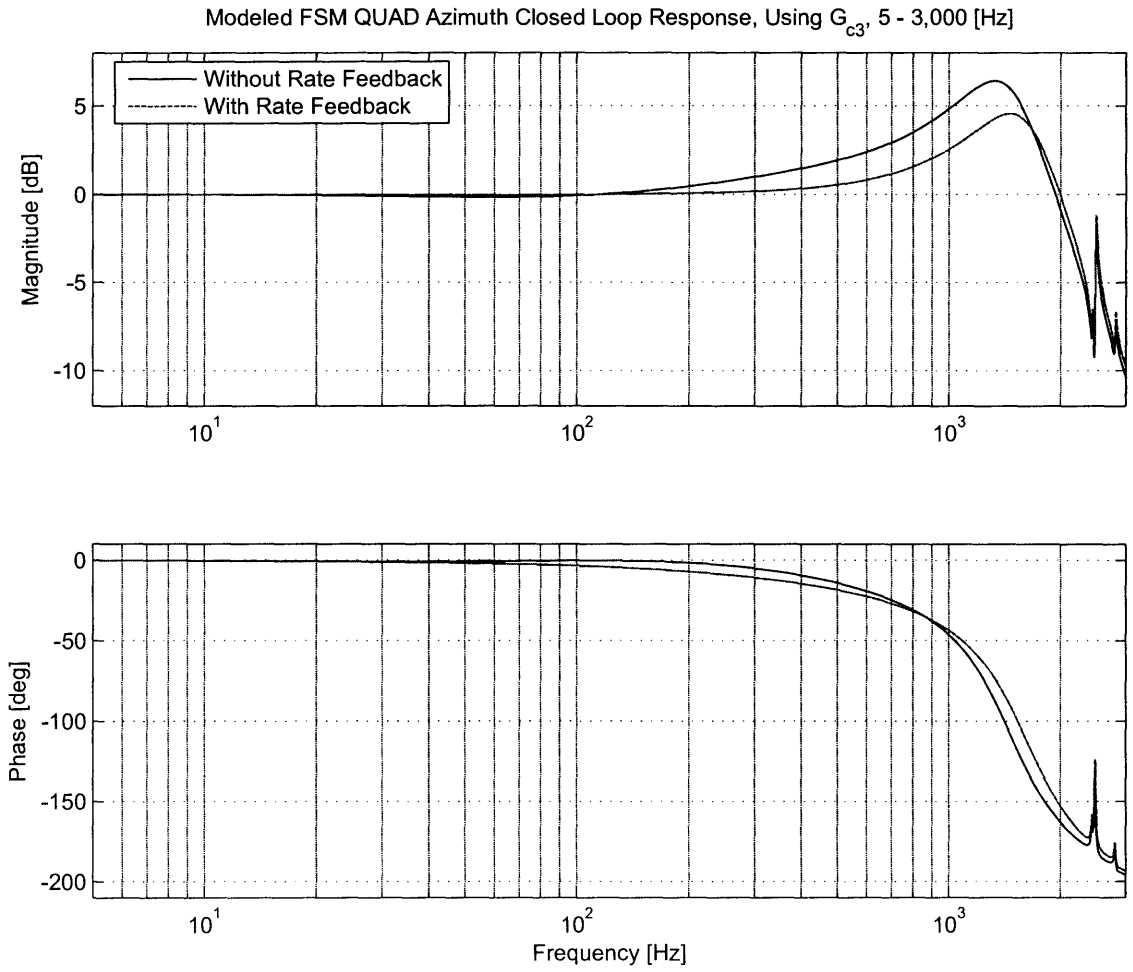

Figure 5-36: Modeled FSM quad cell Azimuth axis closed loop response using $G_{c3}(s)$ with $K = .043$.

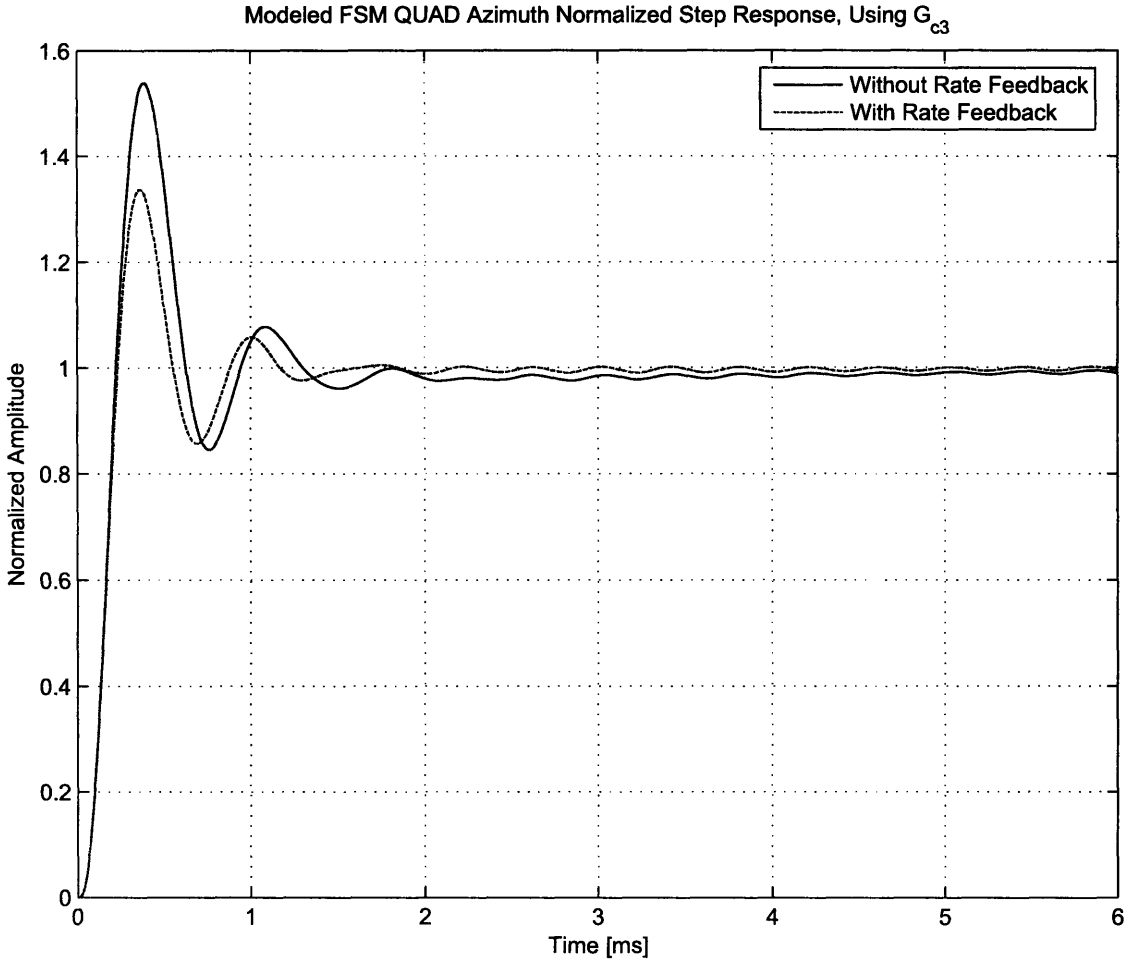

Figure 5-37: Modeled FSM quad cell Azimuth axis step response using $G_{c3}(s)$ with $K = .043$.

very similar to the closed loop response of the system when using $G_{c2}(s)$ as shown in Figure 5-29. Figure 5-37 showing the step response of the system is also very similar to the step response when $G_{c2}(s)$ is used. The same long-tail transient is present in Figure 5-37 because of the lag compensator Table 5.4 summarizes the predicted performance of $G_{c3}(s)$.

Parameter	Without RFB	With RFB
K	0.042	0.042
f_c [kHz]	1.26	1.32
G_m [dB]	6.8	6.2
Φ_m [deg]	28	36
$D_r(s)$ @ 10 Hz [dB]	42	42
-3 dB Frequency [kHz]	2.16	2.21

Table 5.4: Modeled performance specifications of the FSM quad cell Azimuth axis using $G_{c3}(s)$, both with and without rate feedback.

5.2.5 Experimental Results

This section presents the experimental results from implementing compensators $G_{c2}(s)$ and $G_{c3}(s)$ digitally. Each compensator was tested on the Azimuth axis of the FSM with the Elevation axis under low bandwidth control to avoid coupling issues.

Experimental Results Using $G_{c2}(s)$ Without Rate Feedback

The $G_{c2}(s)$ compensator is designed to push the bandwidth of the FSM as high as possible without using rate feedback. The experimental negative loop transmission is shown in Figure 5-38.

The experimental data matches up with the simulated response very well throughout the plot. Figure 5-39 shows the experimental negative loop transmission near the crossover frequency.

There are some small discrepancies in the experimental data just after crossover. Figure 5-39 shows several mini "bumps" in the response right after crossover. The

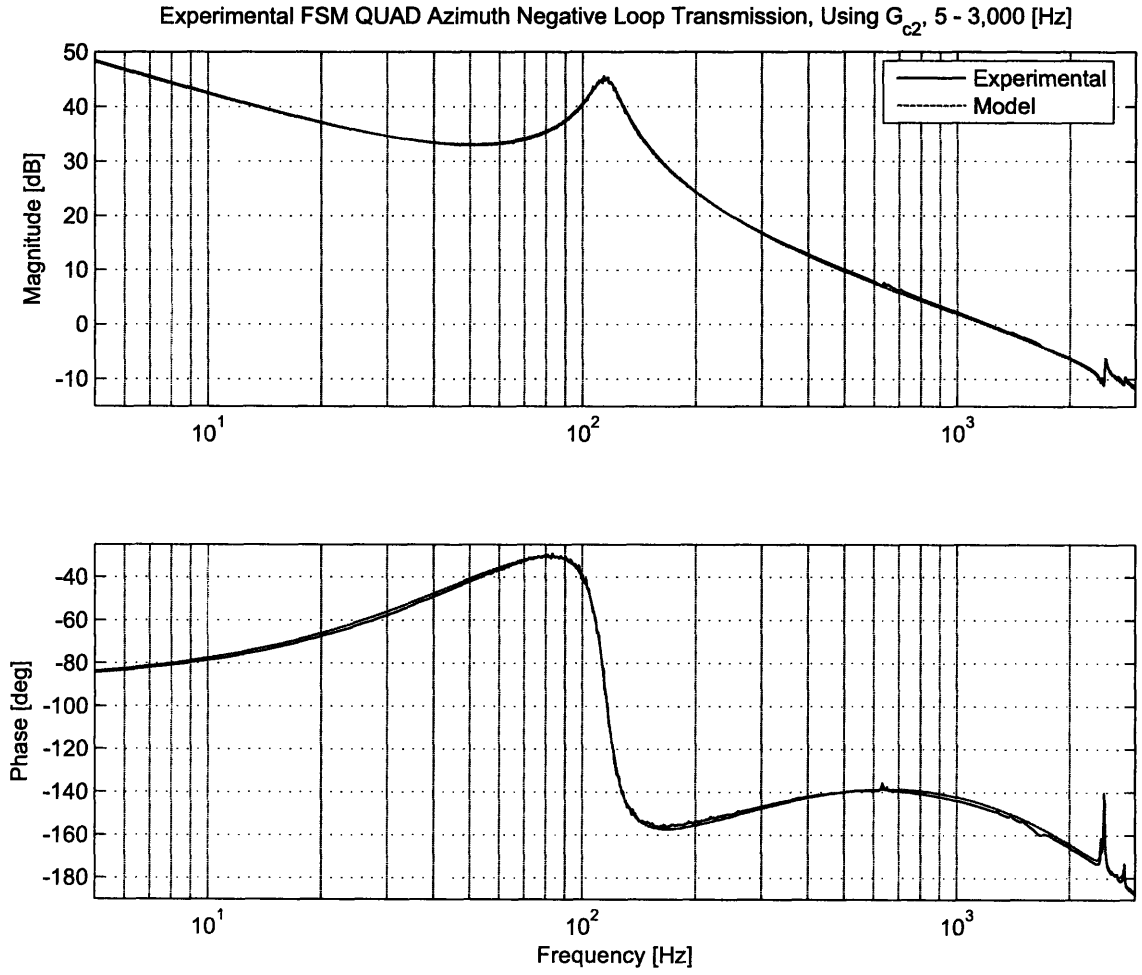

Figure 5-38: Experimental FSM quad cell Azimuth axis negative loop transmission using $G_{c2}(s)$ with $K = .0435$, 5 Hz - 3 kHz.

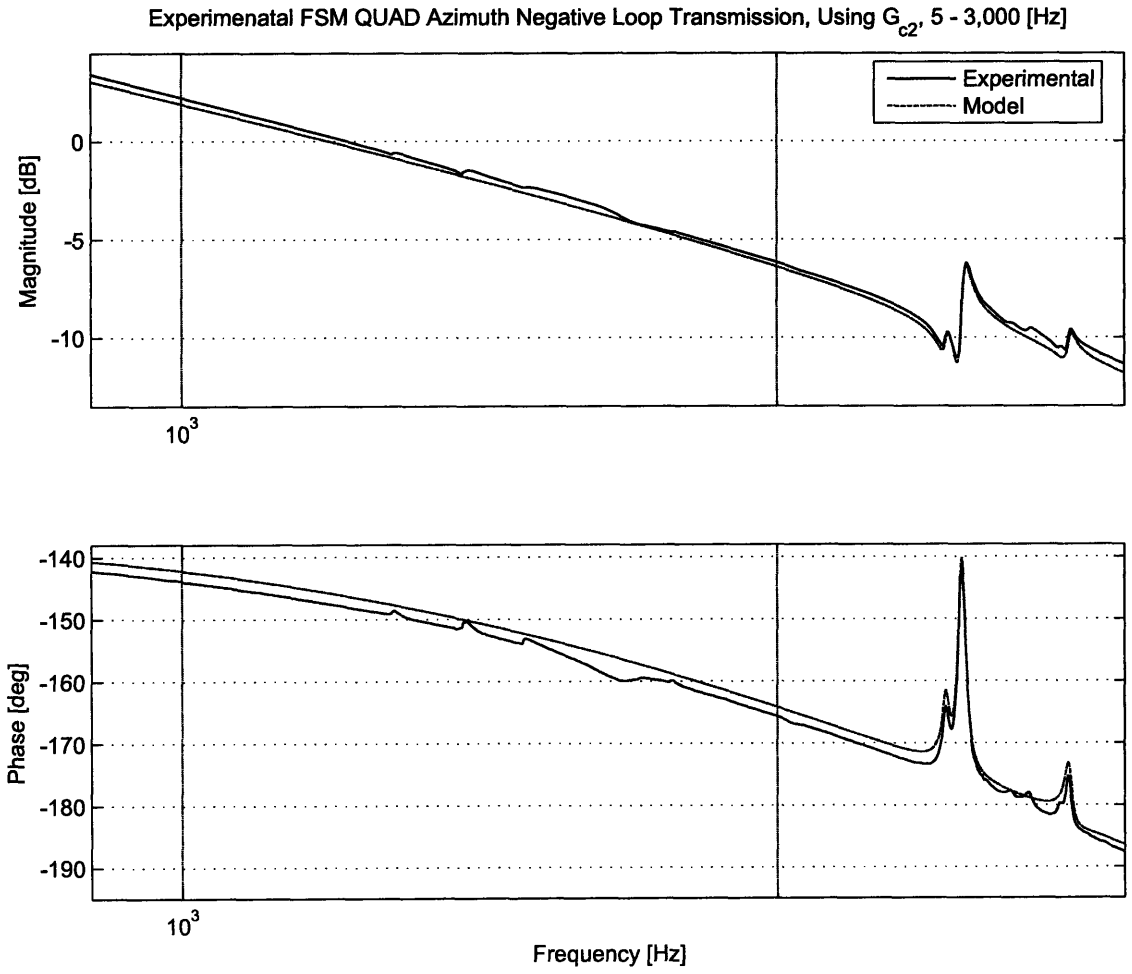

Figure 5-39: Experimental FSM quad cell Azimuth axis negative loop transmission near crossover using $G_{c2}(s)$ with $K = .0435$, 2.25 - 3 kHz.

reason for these bumps is not apparent. Ignoring them for the moment, Figures 5-40 and 5-41 show the experimental Nyquist diagram.

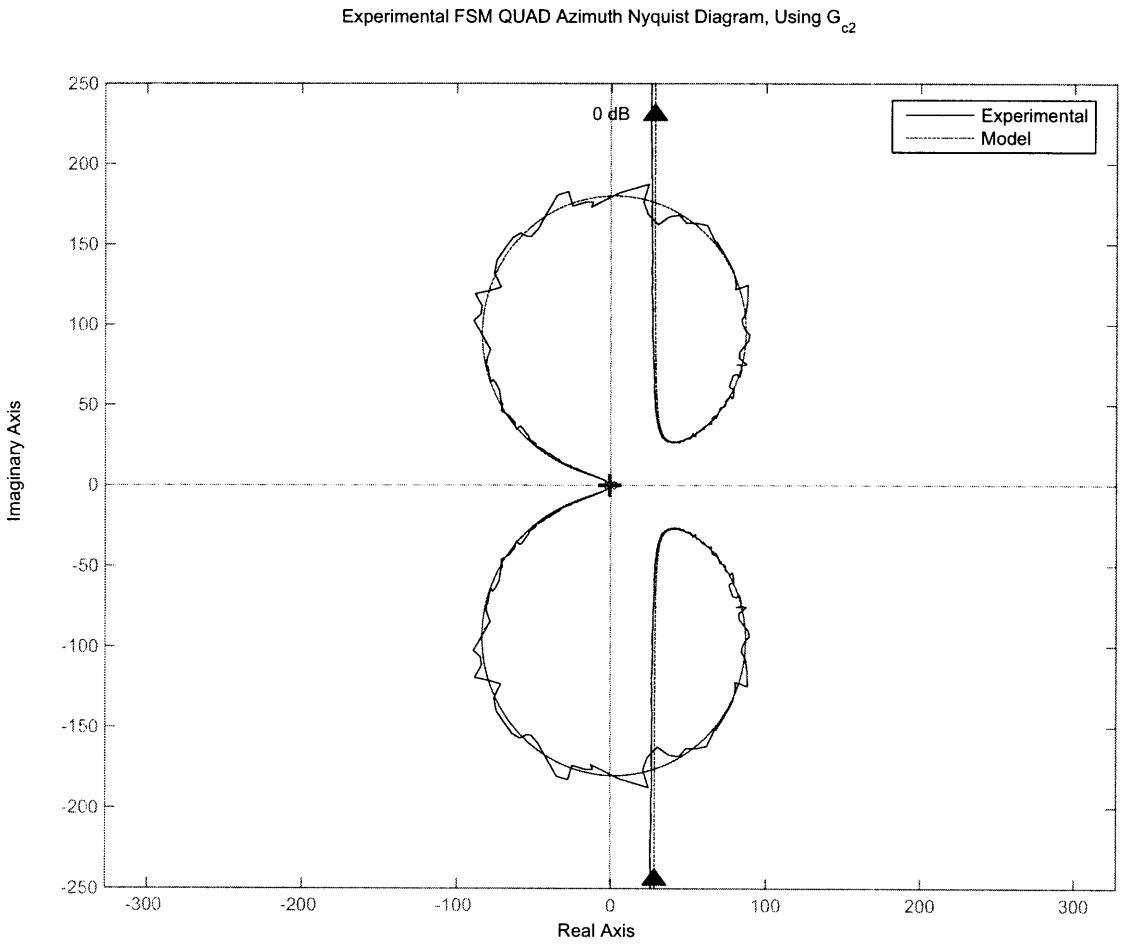

Figure 5-40: Experimental FSM quad cell Azimuth axis Nyquist diagram using $G_{c2}(s)$ with $K = .0435$.

As expected, there are no encirclements of the -1 point, so the system is closed loop stable. From this data, the maximum peak of the closed loop system is expected to be no more than 6 dB. The experimental closed loop response is shown in Figure 5-42, and the step response is shown in Figure 5-43.

Whoa! The closed loop response diverges away from the predicted response near the crossover frequency. In fact, the maximum peak seems to occur at the same place where the "mini-bumps" were in the negative loop transmission shown in Figure 5-39.

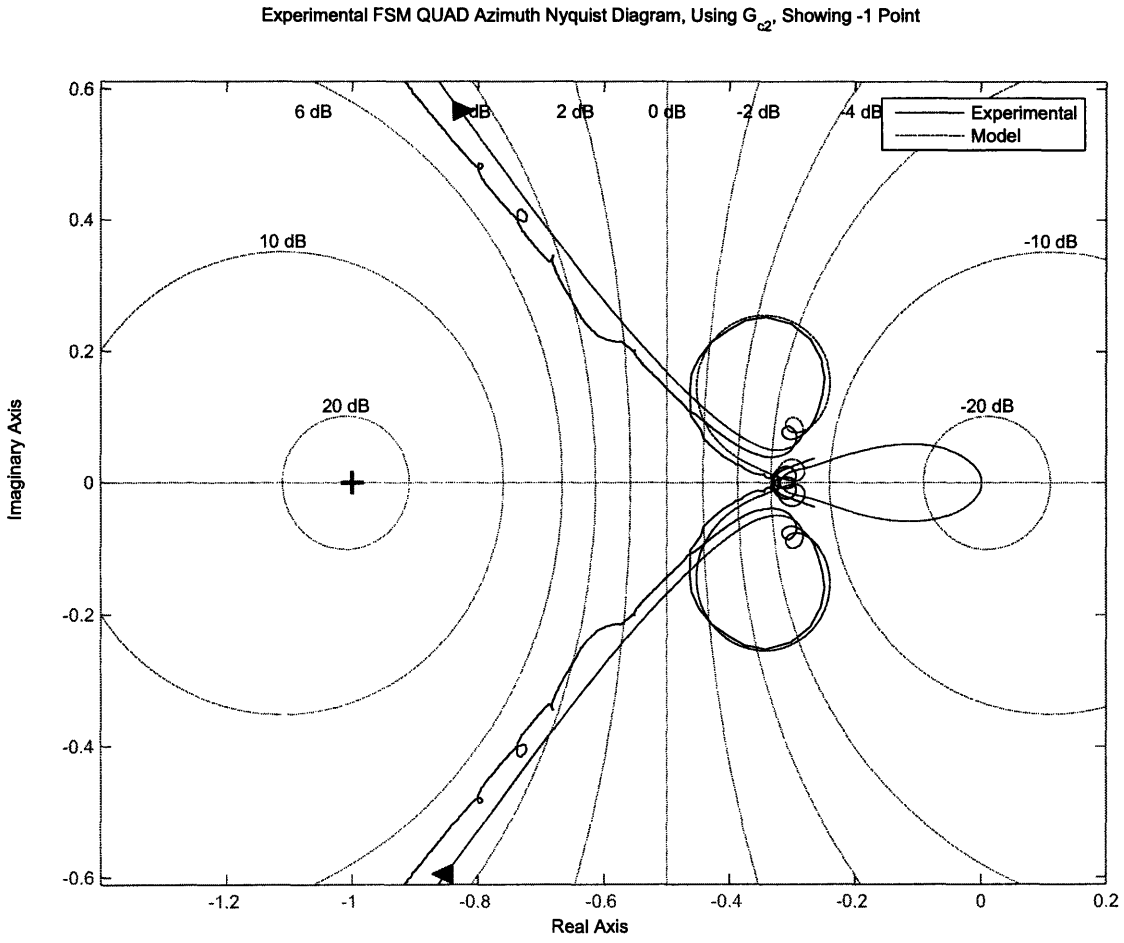

Figure 5-41: Experimental FSM quad cell Azimuth axis Nyquist diagram near the -1 point using $G_{c2}(s)$ with $K = .0435$.

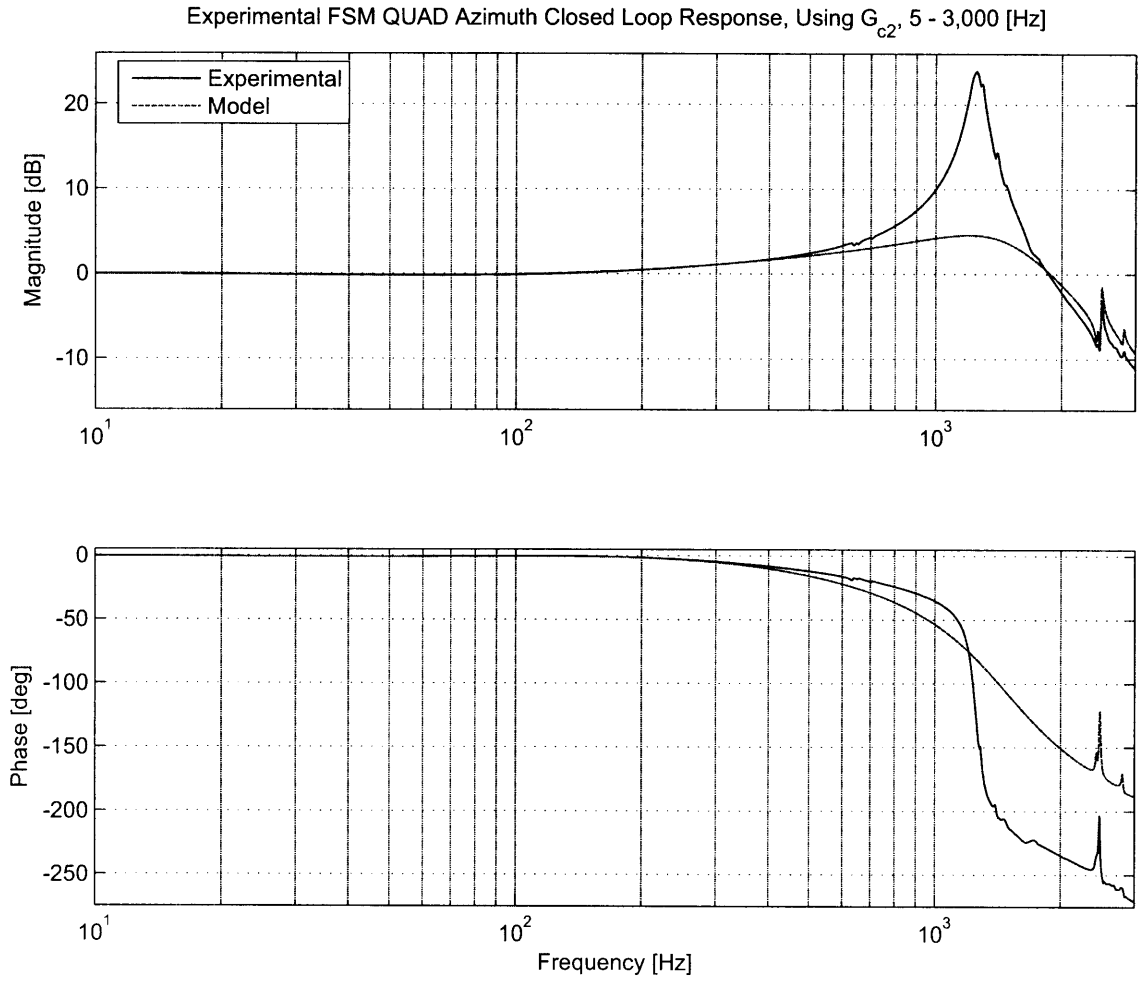

Figure 5-42: Experimental FSM quad cell Azimuth axis closed loop frequency response using $G_{c2}(s)$ with $K = .0435$.

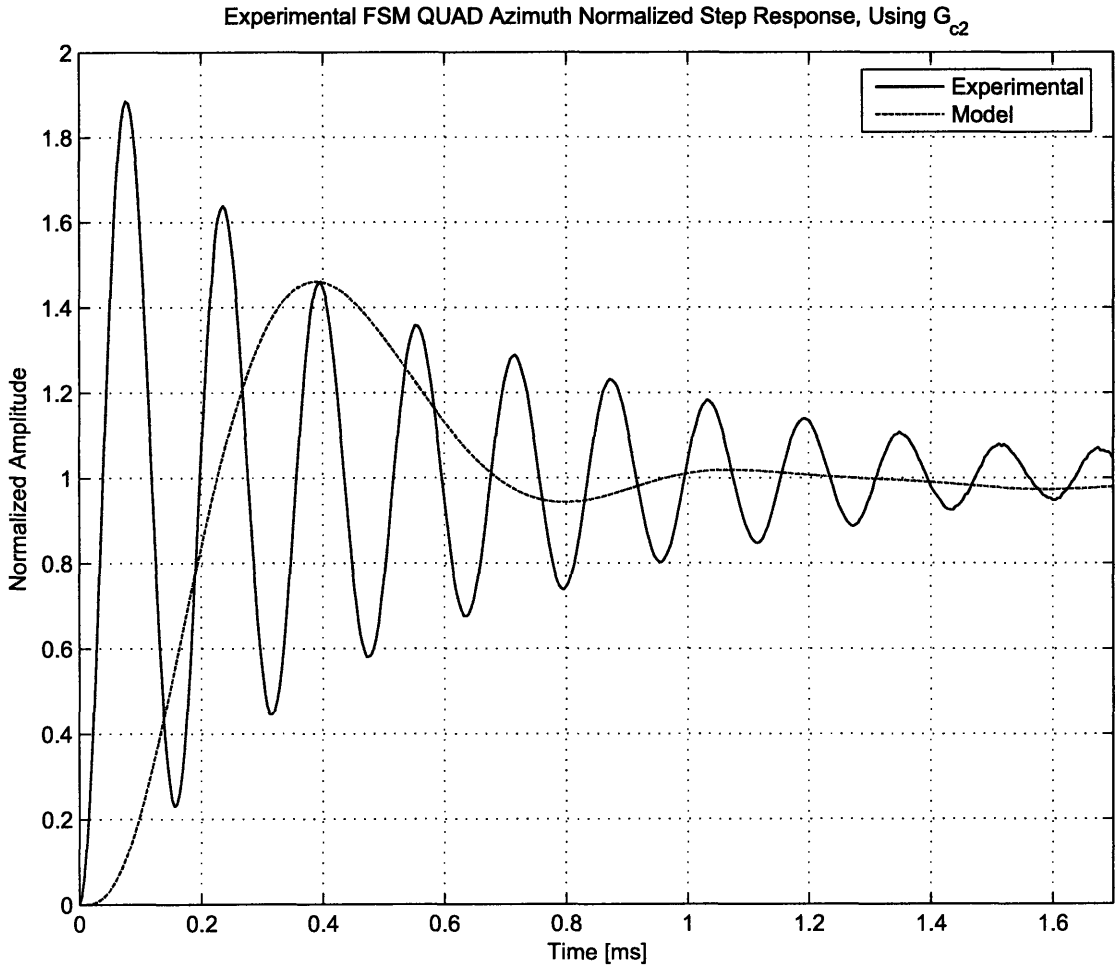

Figure 5-43: Experimental FSM quad cell Azimuth axis normalized step response using $G_{c2}(s)$ with $K = .0435$.

The step response in Figure 5-43 also diverges from the predicted response to match the experimental closed loop response. The step response also shows that there is a constant, steady state oscillation present in the system. Upon further investigation, it was determined that the oscillation frequency was gain dependent: the frequency of the oscillation changed when the compensator gain was changed. This leads to a hypothesis that the digital controller is introducing these oscillations as a limit cycle in the digital control. Technically, the compensator still meets the specifications, but should not be used with these oscillations. Their are 3 ways to remedy the situation: decrease the gain of the compensator until the oscillation disappears, use a faster sampling rate or implement the compensator with analog components. Decreasing the gain of the system helps, but then the maximum achievable bandwidth is reduced. The digital compensator is running as fast as the hardware will allow already, and building an analog compensator is beyond the scope of this thesis. Hopefully implementing the rate feedback will help eliminate these oscillations near crossover. Table 5.5 summarizes the performance of the compensator.

Parameter	**Modeled**	**Experimental**
f_c [kHz]	1.19	1.21
G_m [dB]	6.3	6.2
Φ_m [deg]	34	32
$D_r(s)$ @ 10 Hz [dB]	42	42
-3 dB Frequency [kHz]	2.16	2.05

Table 5.5: Experimental performance specifications of the FSM quad cell Azimuth axis using $G_{c2}(s)$ without rate feedback and $K = 0.0435$.

Experimental Results Using $G_{c3}(s)$ With rate feedback

The $G_{c3}(s)$ compensator is designed to push the bandwidth of the FSM as high as possible using rate feedback. Before attempting to implement the compensator with maximum bandwidth, it is prudent to crossover at a lower frequency to avoid the oscillations seen with $G_{c2}(s)$. Using $K = .0210$ puts the crossover frequency near 1

kHz. The experimental negative loop transmission with this gain is shown in Figure 5-44.

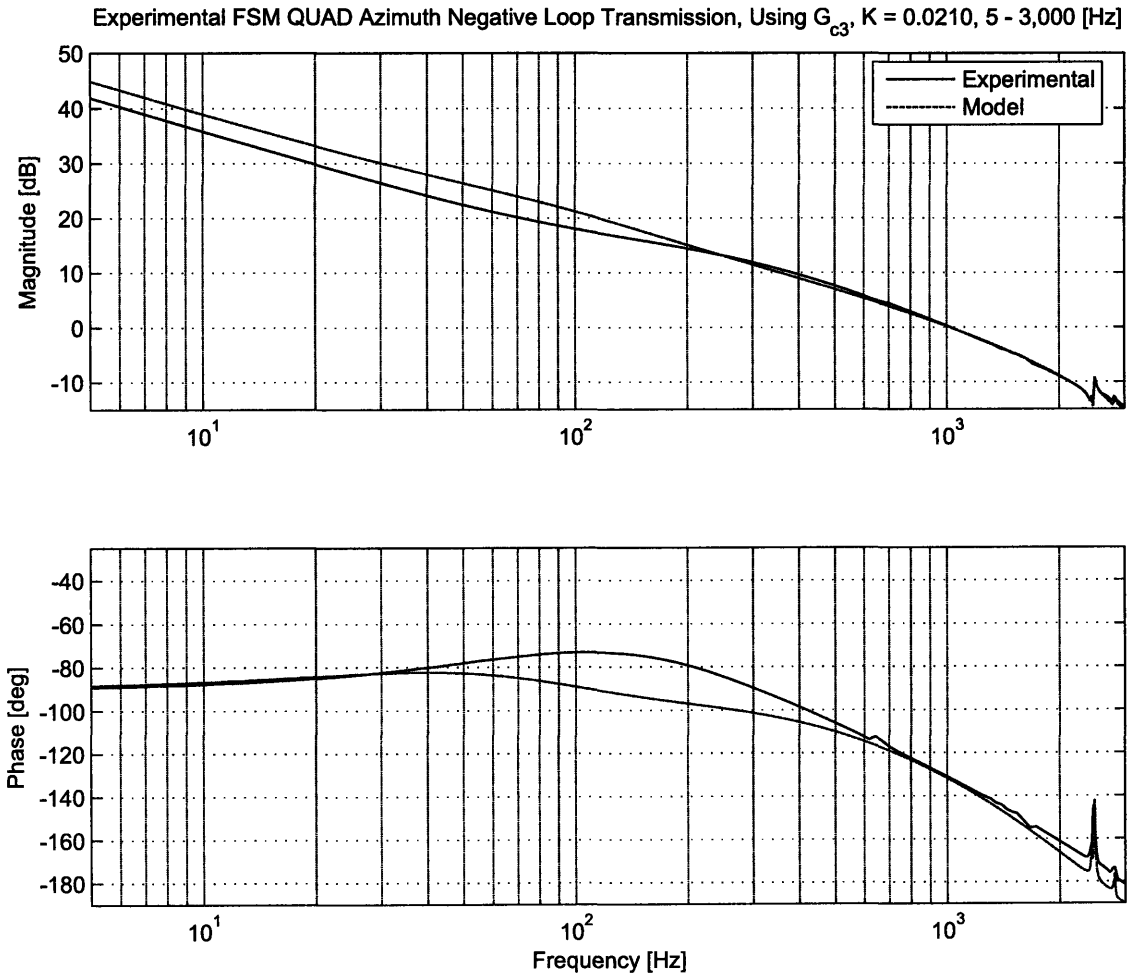

Figure 5-44: Experimental FSM quad cell Azimuth axis negative loop transmission using $G_{c3}(s)$ with $K = .0210$, 5 Hz - 3 kHz.

The experimental data does not match up as well with the model as the data gathered with G_{c2}. There appears to be an extra pole zero pair in the negative loop transmission near 100 Hz. This discrepancy does not affect the predicted crossover frequency, predicted gain or phase margins, or predicted responses, and thus can be ignored (a thorough investigation of this discrepancy was not performed because of time restrictions). The data matches well outside of this discrepancy. The crossover frequency is about 1 kHz, with about 10 dB of gain margin and about 50° of phase

margin. The disturbance rejection at 10 Hz is approximately 36 dB. Figure 5-45 shows the experimental Nyquist diagram near the -1 point.

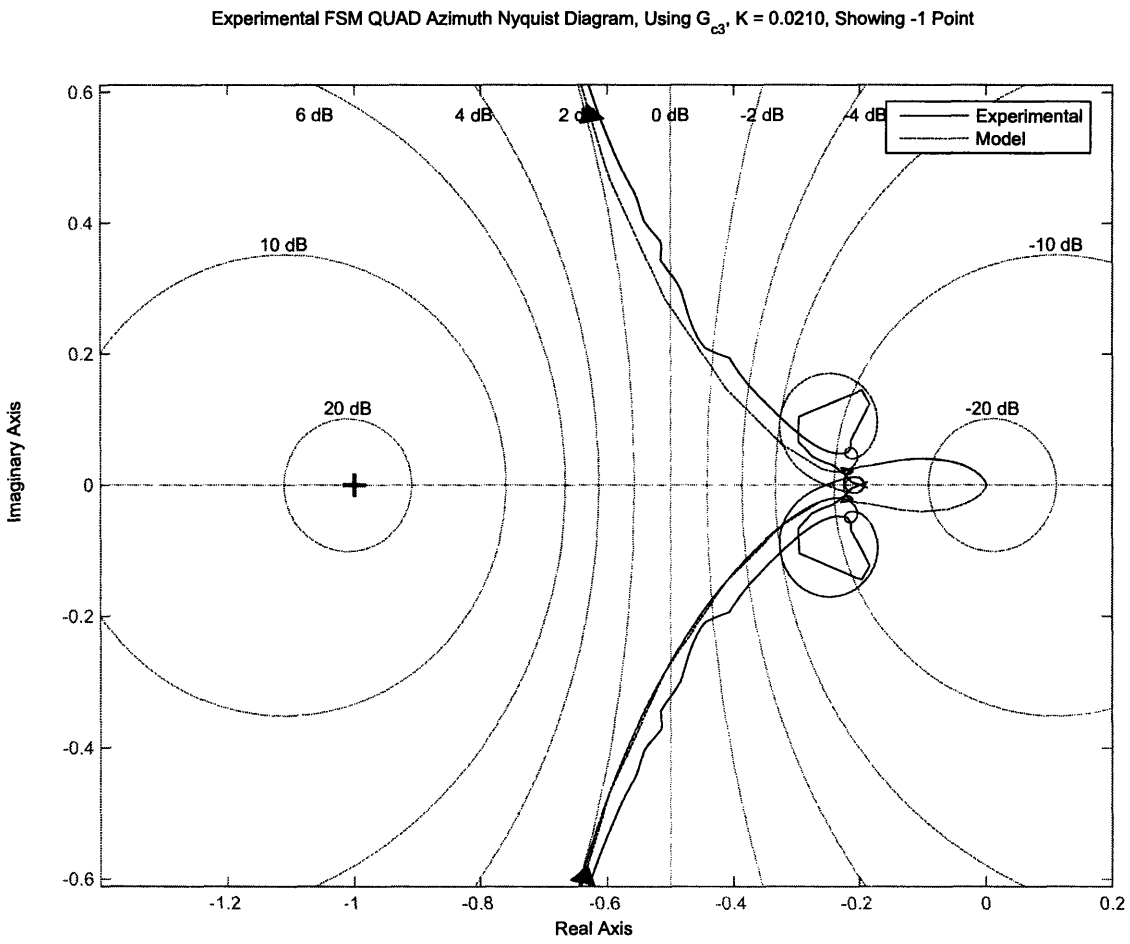

Figure 5-45: Experimental FSM quad cell Azimuth axis Nyquist diagram near the -1 point using $G_{c3}(s)$ with $K = .0210$.

As expected, there are no encirclements of the -1 point, so the system is closed loop stable. From this data, the maximum peak of the closed loop system is expected to be no more than 2 dB. The experimental closed loop response is shown in Figure 5-46, and the step response is shown in Figure 5-47.

As with the G_{c2} compensator, the experimental closed loop and step responses of the system diverge slightly from the predictions. However, there are no oscillations present in the step response. The maximum overshoot of the closed loop response is

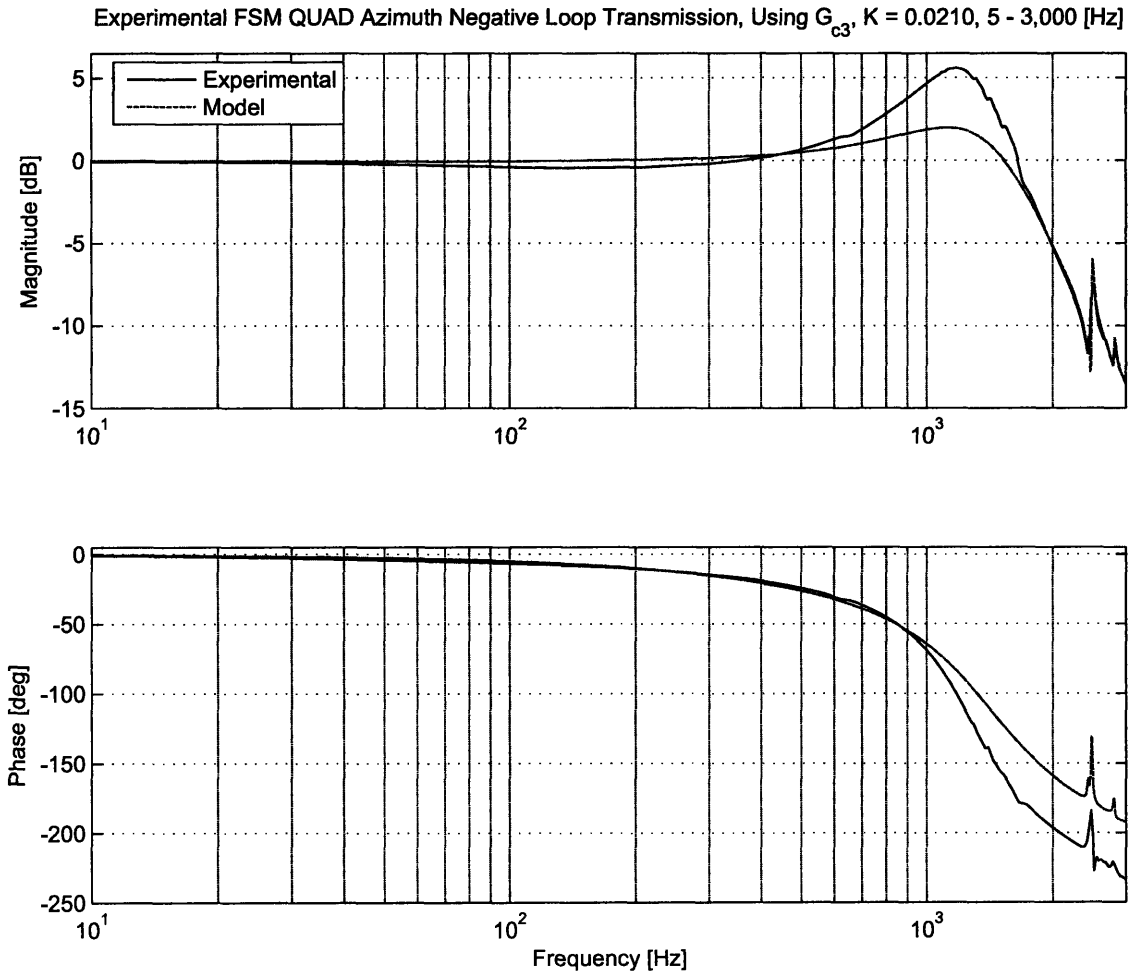

Figure 5-46: Experimental FSM quad cell Azimuth axis closed loop frequency response using $G_{c3}(s)$ with $K = .0210$.

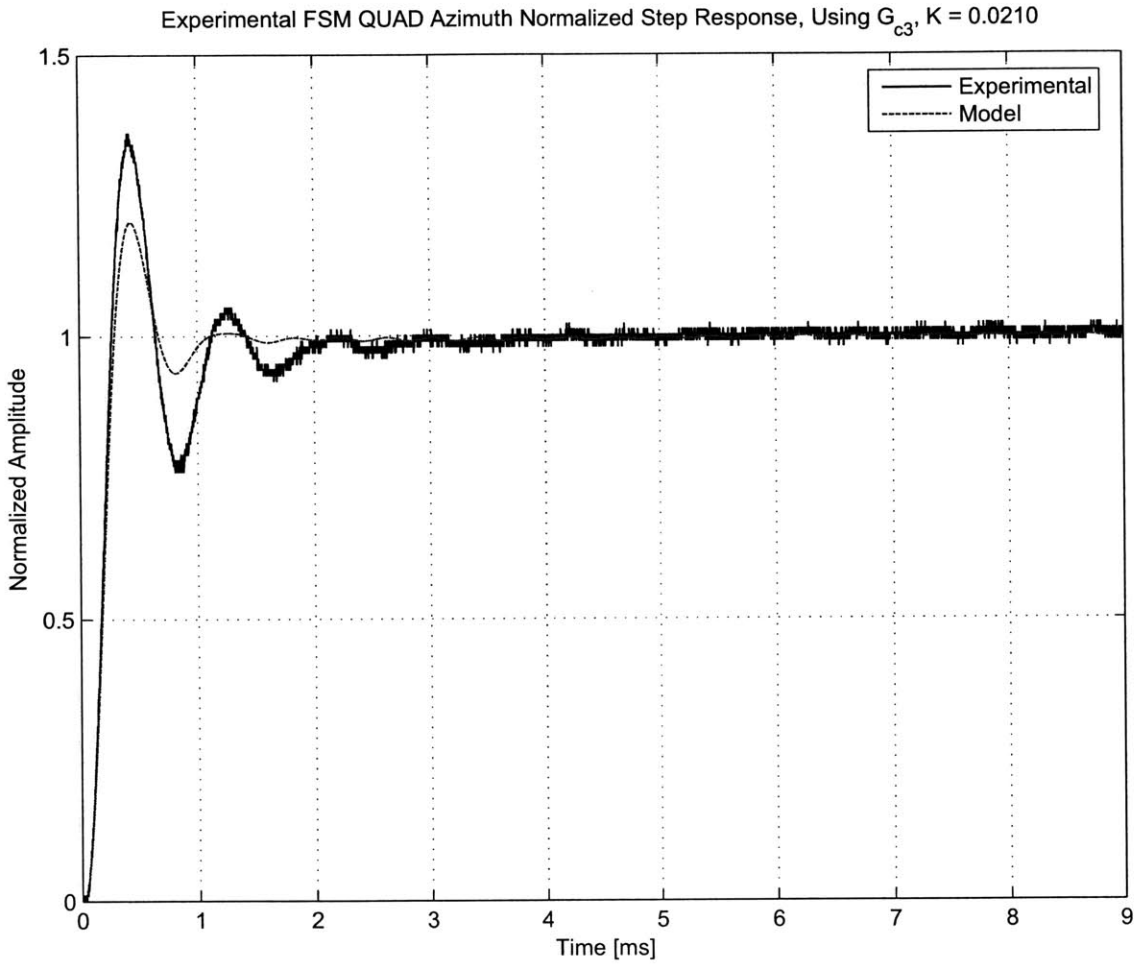

Figure 5-47: Experimental FSM quad cell Azimuth axis normalized step response using $G_{c3}(s)$ with $K = .0210$.

over 5 dB compared with the predicted 2 dB. The step response has a bigger overshoot and longer settling time than predicted. The reasons for these discrepancies is not apparent and warrants further study (for more discussion, see Section 6.2). However, the steady state oscillations observed with $G_{c2}(s)$ have basically disappeared. Table 5.6 summarizes the performance of the compensator with $K = .0210$.

Parameter	Model	Experimental
K	0.021	0.021
f_c [kHz]	1.02	1.02
G_m [dB]	9.3	10
Φ_m [deg]	48	49
$D_r(s)$ @ 10 Hz [dB]	39	36
-3 dB Frequency [kHz]	1.81	1.83

Table 5.6: Experimental performance specifications of the FSM quad cell Azimuth axis using $G_{c3}(s)$ with rate feedback and $K = .0210$.

Proving that the digital compensator can provide oscillation free control, the bandwidth can now be pushed up further. The experimental negative of the loop transmission is shown in Figure 5-48 with $K = .0336$, producing $f_c = 1.42$ kHz.

The experimental data matches the modeled negative loop transmission as well as the previous value of K did. Again, the discrepancy does not affect the response near crossover and can be mostly ignored. The crossover frequency is about 1.42 kHz, with about 6 dB of gain margin and about 35° of phase margin. The disturbance rejection at 10 Hz is approximately 40 dB. Figure 5-49 shows the experimental Nyquist diagram near the -1 point.

As expected, there are no encirclements of the -1 point, so the system is closed loop stable. From this data, the maximum peak of the closed loop system is expected to be no more than 6 dB. However, if the data matches what has been observed thus far, the maximum closed loop peak will be higher than expected, and the step response will have more overshoot and a longer settling time. The experimental closed loop response is shown in Figure 5-50, and the step response is shown in Figure 5-51.

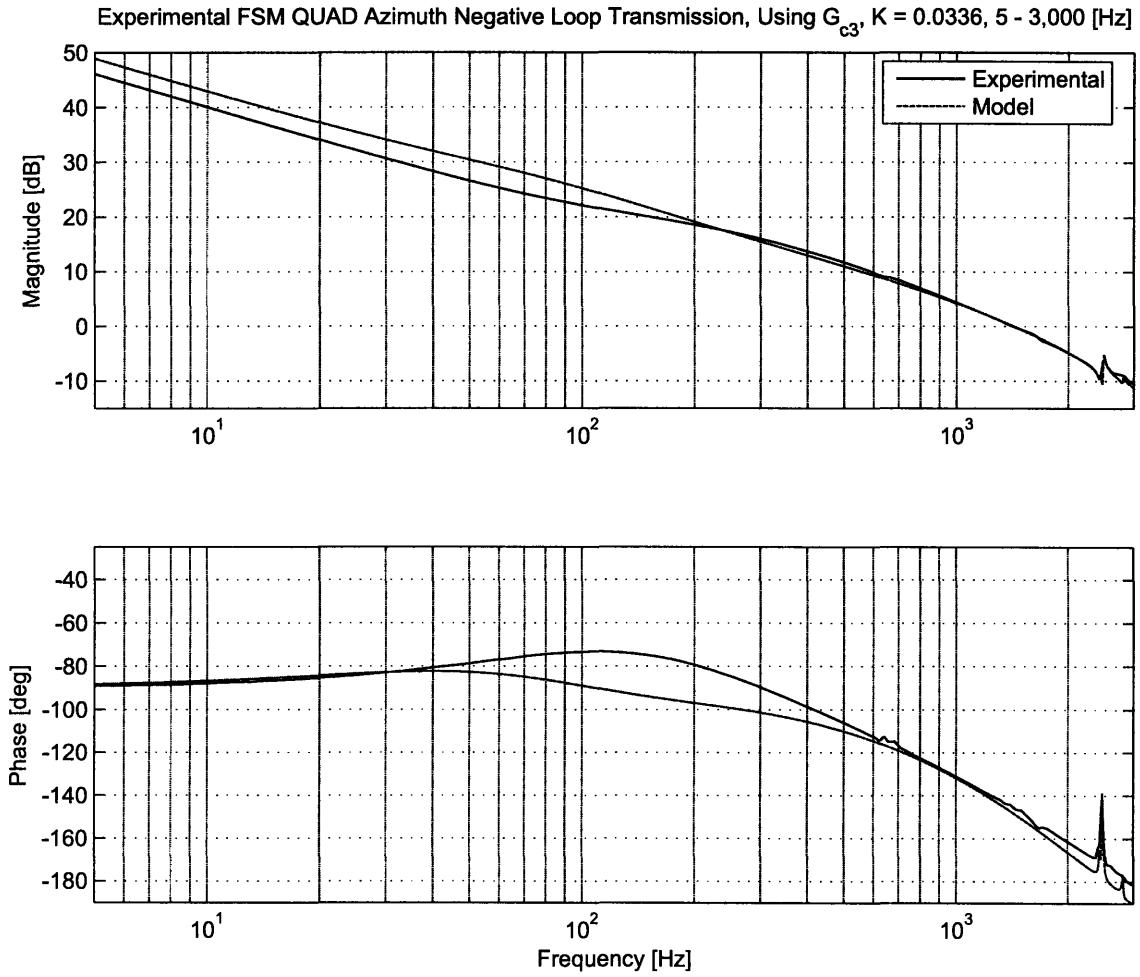

Figure 5-48: Experimental FSM quad cell Azimuth axis negative loop transmission using $G_{c3}(s)$ with $K = .0336$, 5 Hz - 3 kHz.

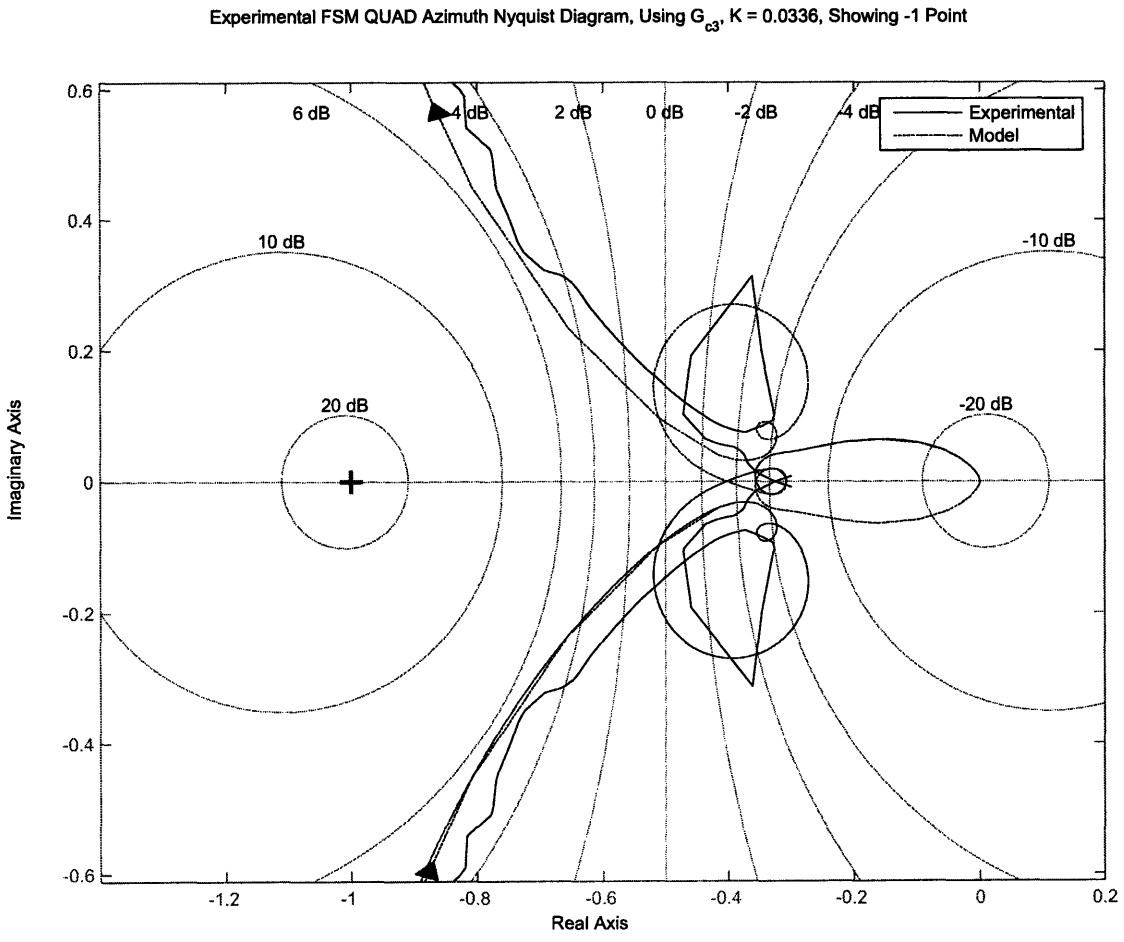

Figure 5-49: Experimental FSM quad cell Azimuth axis Nyquist diagram near the -1 point using $G_{c3}(s)$ with $K = .0336$.

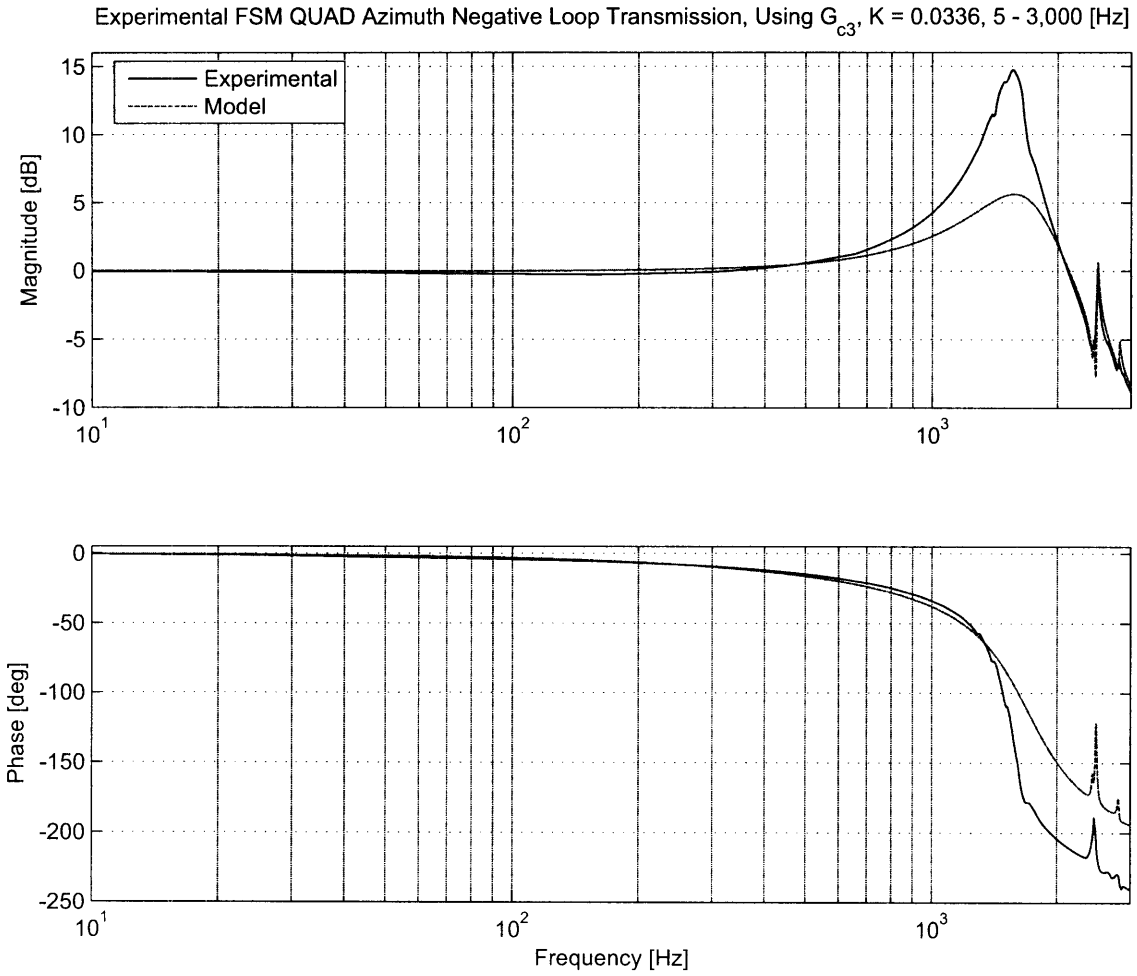

Figure 5-50: Experimental FSM quad cell Azimuth axis closed loop frequency response using $G_{c3}(s)$ with $K = .0336$.

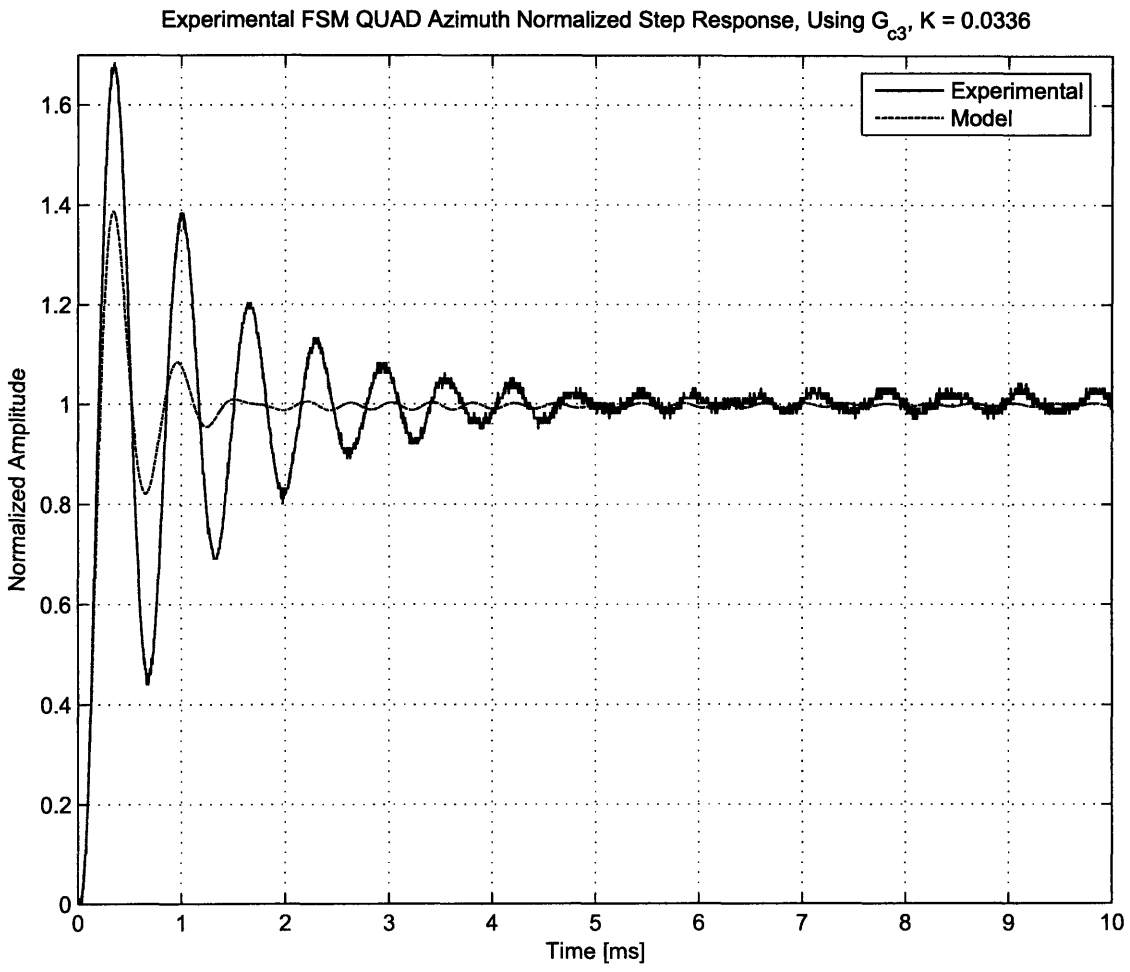

Figure 5-51: Experimental FSM quad cell Azimuth axis normalized step response using $G_{c3}(s)$ with $K = .0336$.

As expected, the closed loop response shows much higher peaking than the model predicts. Furthermore, the step response shows higher overshoot, an increased settling time and the steady state oscillation. When the gain is push up further, the closed loop peak continues to grow, but also shifts out in frequency. The steady state oscillation also grows in amplitude and increases in frequency as the gain is pushed out further, just as observed when using the G_{c2} compensator. Because the frequency of the oscillation changes with gain, the resonance is probably not coming from the FSM or any other piece of hardware. No conclusions can be drawn from the data at this time, and the issue warrants further study (for more discussion, see Section 6.2). Aside from the oscillation, this compensator pushes the bandwidth of the FSM Azimuth axis near its limits. Table 5.7 summarizes the performance of the compensator with $K = .0336$.

Parameter	**Model**	**Experimental**
K	0.0336	0.0336
f_c [kHz]	1.43	1.42
G_m [dB]	5.3	6
Φ_m [deg]	32	35
$D_r(s)$ @ 10 Hz [dB]	43	40
-3 dB Frequency [kHz]	2.32	2.29

Table 5.7: Experimental performance specifications of the FSM quad cell Azimuth axis using $G_{c3}(s)$ with rate feedback and $K = .0336$.

Chapter 6

Conclusions and Suggestions for Future Work

This Chapter presents a summary of this work and suggestions for future efforts. Section 6.1 summarizes the work accomplished and highlights the possible application of methods and techniques used in this thesis in other hardware setups. It also discusses the hardware limitations imposed on the closed loop response by the FSM. Section 6.2 provides suggestions for future work to continue to improve the closed loop bandwidth of the FSM. Finally, Section 6.3 presents the overall conclusions from the work completed during the thesis.

6.1 Summary

The performance of the Fast Steering Mirror in quad cell feedback mode can be improved by implementing a new compensator in the feedback path. The bandwidth of the system can be improved by 20% without using rate feedback, but can also be improved by 40% by using the same rate feedback technique of the original analog compensator. The mirror, as built, can be used in applications requiring a -3 dB frequency of 2.2 kHz, approximately 10% higher than the original analog compensator. The compensation methods are not only applicable to this particular FSM. The designed compensators can be implemented in any FSM system, with any type

of sensor. The spiral acquisition algorithm is also not limited to use in this specific setup. In a basic form, the spiral location block in Simulink only needs two input sensors to determine position and can output steering commands to any hardware with cartesian axis control. This can be useful for other work in the laboratory requiring the acquisition of an optical sensor.

6.1.1 FSM Hardware Limits

There comes a point where electrical control cannot solve inherently mechanical problems. The main obstacle to improving bandwidth further in this system is the first dominant "doublet" in the open loop frequency response of the FSM. Crossing over beyond this doublet would lead to further problems, as there are several more of these doublets beyond 2.5 kHz. These doublets a are mechanical problem that must be dealt with in the mechanical realm. Eliminating the first doublet could push the closed loop bandwidth of the system beyond 2 kHz. Of course, this would require a complete redesign of the FSM, which makes this discussion moot.

6.2 Suggestions for Future Work

This thesis has tried to implement a digital controller with the goal of increasing the closed loop bandwidth as high as possible given the inherent hardware limitations of the FSM. While these improvements will not be incorporated into the MarsComm program, they can be included in any new programs using FSMs at Lincoln Laboratory. This section suggests possible future work to complete both on the compensator and on the FSM itself if the goal is to continue to improve the closed loop performance of the FSM in optical tracking mode.

6.2.1 Track Down Steady State Oscillations Source

The experimental results presented in Section 5.2.5 show steady state oscillations when the gain of the compensator is pushed up. The frequency of these oscillations is

gain-dependent (and with low enough gain these oscillations are not present). Little more than conjecture is prudent with the data collected, but the hardware is probably not causing a gain dependent oscillation. The best guess with the data available is that the digital controller is introducing some sort of limit cycle, either due to the A/D or D/A introducing quantization or as an artifact due to the phase lag of the digital system. If a digital controller will be used in the future, it is prudent to locate the source of these steady state oscillations and take the necessary steps to eliminate them. Most likely, a higher resolution A/D or D/A subsystem will be required.

6.2.2 Faster Digital Control

One limitation of the hardware used in these experiments outside of the inherent limitations of the FSM is the speed of the digital compensator. The digital compensator can be run reliably at 40 kHz (and slightly less reliably at 45 - 50 kHz), which introduces a digital delay into the system. This delay can be reduced by improving the sampling frequency of the digital controller by upgrading the hardware running the compensator or using a different interface besides xPC.

6.2.3 Controller Implementation in Analog

A major benefit of using a digital controller was the ability to test different compensators in real time by changing the parameters digitally. This flexibility leads to the undesired effect of phase delay, effectively reducing the phase margin of the system. By implementing any of these compensators in (sufficiently fast) analog circuitry, this phase delay is not an issue and the phase margins off all systems will rise by about 5°. For the systems with only 30° of phase margin, this is an almost 20% gain at relatively low cost (building analog circuits is a bit harder than writing code in MATLAB, and after the circuitry is built, it is more difficult to modify).

6.2.4 FSM Modifications

After exhausting all control options to improve the performance of the FSM, it might be necessary to modify the hardware directly. There are two main hardware changes that could be implemented in the FSM without a complete redesign of the hardware or control methods.

Remove the KAMAN Sensors

The KAMAN sensors in the FSM are very practical to have, as they provide a direct measurement of the mirror angle and allow feedback without the use of external sensors. However, they do add weight and complexity to the mirror. The KAMAN target plate is attached to the underside of the mirror, and might introduce several more modes into the system than would be present without them. One possible remedy would be to remove the sensors completely, eliminating several vibrational modes of the system. The FSM would then have to be driven open-loop before acquiring the quad cell, but the spiral acquisition algorithm should be able to accomplish this easily (though at a much lower frequency than when using the algorithm in closed loop KAMAN mode).

Improved Electromagnetic Actuation

The voice coils in the FSM work well enough in the system, but like any component, an upgrade wouldn't hurt. The voice coil actuators could be replaced with a similar type of non contact linear motor. Development of such motors for the use in ultra fast tool servos has taken place in the Precision Motion Control Laboratory at MIT. These actuators provide extraordinarily high accelerations, and integrating them into the FSM would greatly improve stiffness and performance.

6.3 Conclusions

The main contribution of this thesis is the characterization of the FSM hardware and the implementation of a digital compensator to achieve the highest bandwidth possible given the hardware limitations. The results presented serve as a baseline performance report on this particular FSM for Lincoln Laboratory personnel wishing to incorporate the FSM into a future project. The thesis also serves as a starting point for hardware modifications to the FSM, including complete redesigns.

Appendix A

State-Space Representation of a "Doublet"

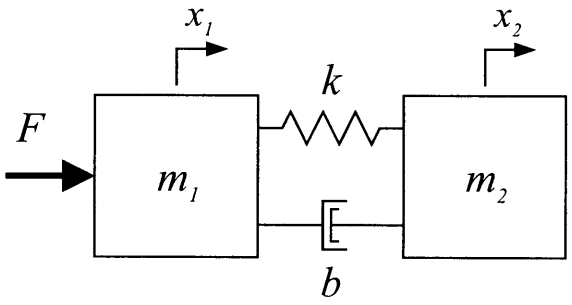

Figure A-1: Simple model of one axis of FSM. m_1 represents the part of the FSM that is directly monitored by the KAMAN sensors, and m_2 represents the decoupling mass.

State-space representation of the position x_1 of m_1, with a force input.

$$A = \begin{bmatrix} -\frac{b(m_1+m_2)}{m_1 m_2} & -\frac{k(m_1+m_2)}{m_1 m_2} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & \frac{1}{m_1} & \frac{b}{m_1 m_2} & \frac{k}{m_1 m_2} \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

State-space representation of the position x_2 of m_2, with a force input.

$$A = \begin{bmatrix} -\frac{b(m_1+m_2)}{m_1 m_2} & -\frac{k(m_1+m_2)}{m_1 m_2} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & \frac{b}{m_1 m_2} & \frac{k}{m_1 m_2} \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Appendix B

Azimuth KAMAN State-Space Model

The following are the state-space matrices describing the model derived from the FSM Azimuth KAMAN frequency response data shown in Figure 3-3.

$A =$

$$\begin{bmatrix} -124 & -5.04 \cdot 10^5 & -30 & -2.2 \cdot 10^6 & -45 & -2.6 \cdot 10^6 & -13 & -1.5 \cdot 10^6 & 2769 & 1.108 \cdot 10^4 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -260 & -1.158 \cdot 10^8 & -45 & -2.6 \cdot 10^6 & -13 & -1.5 \cdot 10^6 & 2769 & 1.108 \cdot 10^4 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -165 & -2.308 \cdot 10^8 & -13 & -1.5 \cdot 10^6 & 2769 & 1.108 \cdot 10^4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -85 & -3.043 \cdot 10^8 & 2769 & 1.108 \cdot 10^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9000 & 1.6 \cdot 10^4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.6 \cdot 10^4 & 8 \cdot 10^4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 \cdot 10^4 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1.939 \cdot 10^5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Appendix C

Elevation KAMAN State-Space Model

The following are the state-space matrices describing the model derived from the FSM Elevation KAMAN frequency response data shown in Figure 3-7.

$$A = \begin{bmatrix} -127.5 & -4.925 \cdot 10^5 & -20 & -1.2 \cdot 10^6 & -30 & -2.8 \cdot 10^6 & -20 & -1.3 \cdot 10^6 & 9900 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -290 & -6.86 \cdot 10^7 & -30 & -2.8 \cdot 10^6 & -20 & -1.3 \cdot 10^6 & 9900 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -160 & -2.265 \cdot 10^8 & -20 & -1.3 \cdot 10^6 & 9900 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -90 & -3.003 \cdot 10^8 & 9900 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9900 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1.942 \cdot 10^5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Appendix D

Azimuth Quad Cell State-Space Model

The following are the state-space matrices describing the model derived from the FSM Azimuth quad cell frequency response shown in Figure 3-10. Due to size constraints, some matrices have been broken into several sections. The subscripts reference which columns are included in each piece.

$A_{1:10} =$

$$
\begin{bmatrix}
-125 & -5.15 \cdot 10^5 & 5 & -4 \cdot 10^5 & 5 & -1.5 \cdot 10^6 & 15 & -5 \cdot 10^5 & 5 & 4 \cdot 10^5 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -152 & -2.321 \cdot 10^8 & 5 & -1.5 \cdot 10^6 & 15 & -5 \cdot 10^5 & 5 & 4 \cdot 10^5 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -152 & -2.424 \cdot 10^8 & 15 & -5 \cdot 10^5 & 5 & 4 \cdot 10^5 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -150 & -3.119 \cdot 10^8 & 5 & 4 \cdot 10^5 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -150 & -3.946 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$A_{11:20} =$

$$
\begin{bmatrix}
-0.8 & -1\cdot10^{5} & 0 & 1\cdot10^{6} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.8 & -1\cdot10^{5} & 0 & 1\cdot10^{6} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.8 & -1\cdot10^{5} & 0 & 1\cdot10^{6} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.8 & -1\cdot10^{5} & 0 & 1\cdot10^{6} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.8 & -1\cdot10^{5} & 0 & 1\cdot10^{6} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-5.8 & -7.098\cdot10^{8} & 0 & 1\cdot10^{6} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -250 & -1.019\cdot10^{9} & 0 & -3\cdot10^{6} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -400 & -2.202\cdot10^{9} & 40 & 1\cdot10^{7} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -410 & -2.912\cdot10^{9} & -100 & 5\cdot10^{7} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -700 & -3.605\cdot10^{9} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$A_{21:30} =$

$$
\begin{bmatrix}
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-150 & 6\cdot10^7 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-650 & -4.1\cdot10^9 & 270 & 3.9\cdot10^8 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -530 & -4.275\cdot10^9 & -34.5 & -1.4\cdot10^7 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -35 & -5.464\cdot10^9 & 0 & -5.4\cdot10^7 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -800 & -6.044\cdot10^9 & 50 & -8\cdot10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -750 & -7.55\cdot10^9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$A_{31:40} =$

$$\begin{bmatrix}
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
170 & -4 \cdot 10^7 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-630 & -7.765 \cdot 10^9 & 70 & -1.5 \cdot 10^7 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -380 & -8.18 \cdot 10^9 & 100 & 1.5 \cdot 10^7 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -600 & -8.645 \cdot 10^9 & 100 & -1 \cdot 10^7 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -600 & -9.96 \cdot 10^9 & 500 & 1 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1000 & -1.195e+010 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$A_{41:44} = \begin{bmatrix}
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-1.5 \cdot 10^4 & 5624 & 2.22 \cdot 10^4 & 0 \\
0 & 0 & 0 & 0 \\
-3 \cdot 10^4 & 3749 & 1.48 \cdot 10^4 & 0 \\
0 & -7400 & 3 \cdot 10^4 & 0 \\
0 & 0 & -3 \cdot 10^4 & 9 \cdot 10^4 \\
0 & 0 & 0 & -9 \cdot 10^4
\end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C_{1:22} = \begin{bmatrix} 0 & 2.456 \cdot 10^{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_{23:44} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Appendix E

Elevation Quad Cell State-Space Model

The following are the state-space matrices describing the model derived from the FSM Elevation quad cell frequency response shown in Figure 3-14. Due to size constraints, some matrices have been broken into several sections. The subscripts reference which columns are included in each piece.

$A_{1:10} =$

$$
\begin{bmatrix}
-127.5 & -5.1 \cdot 10^5 & 0 & -3 \cdot 10^5 & 6 & -1.7 \cdot 10^6 & 5 & -3 \cdot 10^5 & -8 & 4 \cdot 10^5 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -160 & -2.422 \cdot 10^8 & 6 & -1.7 \cdot 10^6 & 5 & -3 \cdot 10^5 & -8 & 4 \cdot 10^5 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -162 & -2.313 \cdot 10^8 & 5 & -3 \cdot 10^5 & -8 & 4 \cdot 10^5 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -100 & -3.051 \cdot 10^8 & -8 & 4 \cdot 10^5 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -258 & -3.736 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$A_{11:20} =$

$$
\begin{bmatrix}
-1 & -1.5 \cdot 10^5 & 0 & 5 \cdot 10^5 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1.5 \cdot 10^5 & 0 & 5 \cdot 10^5 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1.5 \cdot 10^5 & 0 & 5 \cdot 10^5 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1.5 \cdot 10^5 & 0 & 5 \cdot 10^5 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1.5 \cdot 10^5 & 0 & 5 \cdot 10^5 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-12 & -7.086 \cdot 10^8 & 0 & 5 \cdot 10^5 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -400 & -1.068 \cdot 10^9 & 0 & -1.8 \cdot 10^7 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -400 & -2.205 \cdot 10^9 & 40 & -1 \cdot 10^7 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -410 & -2.922 \cdot 10^9 & 0 & 5 \cdot 10^6 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -800 & -3.625 \cdot 10^9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$A_{21:30} =$$

$$\begin{bmatrix}
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-250 & 1.5 \cdot 10^8 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-650 & -4.065 \cdot 10^9 & 350 & 3.4 \cdot 10^8 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -450 & -4.255 \cdot 10^9 & -34.5 & -1.4 \cdot 10^7 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -35 & -5.464 \cdot 10^9 & -100 & -2.94 \cdot 10^8 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -600 & -5.964 \cdot 10^9 & 200 & -6 \cdot 10^8 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -800 & -7.15 \cdot 10^9 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$A_{31:40} =$

$$
\begin{bmatrix}
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
100 & -2\cdot10^7 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-500 & -7.72\cdot10^9 & 30 & -1.5\cdot10^7 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -350 & -8.14\cdot10^9 & 30 & -2.5\cdot10^7 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -320 & -8.78\cdot10^9 & -150 & -2.5\cdot10^7 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -750 & -9.815\cdot10^9 & -350 & -9\cdot10^7 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -900 & -1.112\cdot10^{10} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

$$
A_{41:48} = \begin{bmatrix}
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.409 \cdot 10^{10} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3000 & -1.409 \cdot 10^{10} & -250 & -7.5 \cdot 10^{7} & 4500 & -1.511 \cdot 10^{4} & 2.833 \cdot 10^{4} & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -950 & -9.1 \cdot 10^{9} & 4500 & -1.511 \cdot 10^{4} & 2.833 \cdot 10^{4} & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -9000 & -3.022 \cdot 10^{4} & 5.667 \cdot 10^{4} & 0 \\
0 & 0 & 0 & 0 & 0 & -3.4 \cdot 10^{4} & 3 \cdot 10^{4} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -3 \cdot 10^{4} & 3.4 \cdot 10^{4} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -3.4 \cdot 10^{4}
\end{bmatrix}
$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C_{1:24} = \begin{bmatrix} 0 & 2.125 \cdot 10^{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_{25:48} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \end{bmatrix}$$

Appendix F

Code Used to Generate GUI for Control of the Digital Compensator

```
function varargout = fsm_comp(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @fsm_comp_OpeningFcn, ...
                   'gui_OutputFcn',  @fsm_comp_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before fsm_comp_model is made visible.
function fsm_comp_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
handles.output = hObject;

% Checks to see if a model has been built in MATLAB already. If so,
% restores the current model name. Otherwise, the default model name is
% set. The model name is then written to the fsm_comp.
if evalin('base','exist(''CURRENT_MODEL'');')==1
    handles.MODEL_NAME = evalin('base','CURRENT_MODEL;');
```

```
else
    handles.MODEL_NAME = 'fsm_controller_rotated';
end
set(handles.MODEL_NAME_EDIT,'String',handles.MODEL_NAME);
evalin('base','fsm_comp_init;'); % Evaluates inital variable parameters used in model
guidata(hObject,handles); % Update handles structure


function varargout = fsm_comp_OutputFcn(hObject, eventdata, handles)
% --- Outputs from this function are returned to the command line.
varargout{1} = handles.output;


function BuildFSM_Callback(hObject, eventdata, handles)
evalin('base','fsm_comp_init;'); % Evaluates inital variable parameters used in model
evalin('base',['rtwbuild(''',handles.MODEL_NAME,''');'])
evalin('base','fsm_comp_model = xpctarget.xpc;'); % Sets name for xPC Communication
evalin('base','built=1'); % Set so other parts of script know model has been built
evalin('base','clear current_scope;') % Clears scope out on rebuild. Otherwise scope becomes structure for some reason.
% Get all useful parameter IDs (PID) for use in script
% Save current values of all parameters and set them in GUI
% only do this if the PIDs are valid
% Allows the same GUI to control multiple models with different parameters

handles.PID_AZ_SUM_THRESHOLD = evalin('base','getparamid(fsm_comp_model,'''',''AZ_SUM_THRESHOLD'')');
if ~isempty(handles.PID_AZ_SUM_THRESHOLD) % if it is not empty, IE it exists
    handles.AZ_SUM_THRESHOLD = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_AZ_SUM_THRESHOLD),');']);
    set(handles.AZ_SUM_THRESHOLD_EDIT,'String',num2str(handles.AZ_SUM_THRESHOLD));
end

handles.PID_EL_SUM_THRESHOLD = evalin('base','getparamid(fsm_comp_model,'''',''EL_SUM_THRESHOLD'')');
if ~isempty(handles.PID_EL_SUM_THRESHOLD)
    handles.EL_SUM_THRESHOLD = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_EL_SUM_THRESHOLD),');']);
    set(handles.EL_SUM_THRESHOLD_EDIT,'String',num2str(handles.EL_SUM_THRESHOLD));
end

handles.PID_AZ_QUAD_LOOP_GAIN = evalin('base','getparamid(fsm_comp_model,'''',''AZ_QUAD_LOOP_GAIN'')');
if ~isempty(handles.PID_AZ_QUAD_LOOP_GAIN)
    handles.AZ_QUAD_LOOP_GAIN = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_AZ_QUAD_LOOP_GAIN),');']);
    set(handles.AZ_QUAD_LOOP_GAIN_EDIT,'String',num2str(handles.AZ_QUAD_LOOP_GAIN));
    AZ_MODEL = FSM_MODEL_QUAD_AZ;
    A_AZ_QUAD_CONTROLLER = evalin('base','A_AZ_QUAD_CONTROLLER;');
    B_AZ_QUAD_CONTROLLER = evalin('base','B_AZ_QUAD_CONTROLLER;');
    C_AZ_QUAD_CONTROLLER = evalin('base','C_AZ_QUAD_CONTROLLER;');
    D_AZ_QUAD_CONTROLLER = evalin('base','D_AZ_QUAD_CONTROLLER;');
    SS_MODEL = ss(A_EL_QUAD_CONTROLLER, B_EL_QUAD_CONTROLLER, C_EL_QUAD_CONTROLLER, D_EL_QUAD_CONTROLLER);
    LT = handles.EL_QUAD_LOOP_GAIN * EL_MODEL * SS_MODEL;
    %crossover_freq = round(crossover(LT,0,5,5000)/(2*pi));
    %set(handles.AZ_QUAD_CROSSOVER_EDIT,'String',num2str(crossover_freq));
end

handles.PID_EL_QUAD_LOOP_GAIN = evalin('base','getparamid(fsm_comp_model,'''',''EL_QUAD_LOOP_GAIN'')');
if ~isempty(handles.PID_EL_QUAD_LOOP_GAIN)
    handles.EL_QUAD_LOOP_GAIN = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_EL_QUAD_LOOP_GAIN),');']);
    set(handles.EL_QUAD_LOOP_GAIN_EDIT,'String',num2str(handles.EL_QUAD_LOOP_GAIN));
    EL_MODEL = FSM_MODEL_QUAD_EL;
    A_EL_QUAD_CONTROLLER = evalin('base','A_EL_QUAD_CONTROLLER;');
    B_EL_QUAD_CONTROLLER = evalin('base','B_EL_QUAD_CONTROLLER;');
    C_EL_QUAD_CONTROLLER = evalin('base','C_EL_QUAD_CONTROLLER;');
    D_EL_QUAD_CONTROLLER = evalin('base','D_EL_QUAD_CONTROLLER;');
    SS_MODEL = ss(A_EL_QUAD_CONTROLLER, B_EL_QUAD_CONTROLLER, C_EL_QUAD_CONTROLLER, D_EL_QUAD_CONTROLLER);
```

```
    LT = handles.EL_QUAD_LOOP_GAIN * EL_MODEL * SS_MODEL;
    %crossover_freq = round(crossover(LT,0,5,5000)/(2*pi));
    %set(handles.EL_QUAD_CROSSOVER_EDIT,'String',num2str(crossover_freq));
end

handles.PID_AZ_QUAD_BIAS_COMMAND = evalin('base','getparamid(fsm_comp_model,'''',''AZ_QUAD_BIAS_COMMAND'')');
if ~isempty(handles.PID_AZ_QUAD_BIAS_COMMAND)
    handles.AZ_QUAD_BIAS_COMMAND = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_AZ_QUAD_BIAS_COMMAND),');']);
    set(handles.AZ_QUAD_BIAS_COMMAND_EDIT,'String',num2str(handles.AZ_QUAD_BIAS_COMMAND));
end

handles.PID_EL_QUAD_BIAS_COMMAND = evalin('base','getparamid(fsm_comp_model,'''',''EL_QUAD_BIAS_COMMAND'')');
if ~isempty(handles.PID_EL_QUAD_BIAS_COMMAND)
    handles.EL_QUAD_BIAS_COMMAND = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_EL_QUAD_BIAS_COMMAND),');']);
    set(handles.EL_QUAD_BIAS_COMMAND_EDIT,'String',num2str(handles.EL_QUAD_BIAS_COMMAND));
end

handles.PID_INPUT_COMMAND_SELECT = evalin('base','getparamid(fsm_comp_model,'''',''INPUT_COMMAND_SELECT'')');
if ~isempty(handles.PID_INPUT_COMMAND_SELECT)
    handles.INPUT_COMMAND_SELECT = evalin('base',['fsm_comp_model.getparam(',num2str(handles.PID_INPUT_COMMAND_SELECT),');']);
    switch handles.INPUT_COMMAND_SELECT
        case 1
            set(handles.INPUT_COMMAND_SELECT_NONE,'Value',1);
        case 2
            set(handles.INPUT_COMMAND_SELECT_AZ,'Value',1);
        case 0
            set(handles.INPUT_COMMAND_SELECT_EL,'Value',1);
    end
end

guidata(gcbo,handles); % Update handles structure


function StartFSM_Callback(hObject, eventdata, handles)
evalin('base','fsm_comp_model.start;'); % Starts model


function StopFSM_Callback(hObject, eventdata, handles)
evalin('base','fsm_comp_model.stop;'); % Stops the model


function AZ_QUAD_BIAS_COMMAND_EDIT_Callback(hObject, eventdata, handles)
handles.AZ_QUAD_BIAS_COMMAND = eval(get(hObject,'string'));
evalin('base',['fsm_comp_model.setparam(',num2str(handles.PID_AZ_QUAD_BIAS_COMMAND),',',num2str(handles.AZ_QUAD_BIAS_COMMAND),')']);
guidata(gcbo,handles); % Update handles structure


function EL_QUAD_BIAS_COMMAND_EDIT_Callback(hObject, eventdata, handles)
handles.EL_QUAD_BIAS_COMMAND = eval(get(hObject,'string'));
evalin('base',['fsm_comp_model.setparam(',num2str(handles.PID_EL_QUAD_BIAS_COMMAND),',',num2str(handles.EL_QUAD_BIAS_COMMAND),')']);
guidata(gcbo,handles); % Update handles structure


function scope_start_Callback(hObject, eventdata, handles)
evalin('base','current_scope.start;')


function scope_stop_Callback(hObject, eventdata, handles)
evalin('base','current_scope.stop;')
```

```
function scope_select_SelectionChangeFcn(hObject, eventdata, handles)
selection = get(handles.scope_select, 'SelectedObject');
switch get(selection,'Tag')
    case 'scope_1'
        handles.current_scope = 1;
    case 'scope_2'
        handles.current_scope = 2;
    case 'scope_3'
        handles.current_scope = 3;
    case 'scope_4'
        handles.current_scope = 4;
end

evalin('base',['current_scope = fsm_comp_model.getscope(',num2str(handles.current_scope),');'])
A = evalin('base','current_scope.YLimit;');
handles.scope_min = A(1);
handles.scope_max = A(2);
set(handles.scope_scaling_edit,'String',num2str(handles.scope_max))
set(handles.scope_max_edit,'String',num2str(handles.scope_max))
set(handles.scope_min_edit,'String',num2str(handles.scope_min))
guidata(gcbo,handles);


function scope_scaling_edit_Callback(hObject, eventdata, handles)
handles.scope_scaling = eval(get(hObject,'string'));
handles.scope_min = -1 * handles.scope_scaling;
handles.scope_max = handles.scope_scaling;
evalin('base',['current_scope.set(''YLimit'', [',num2str(handles.scope_min),',',num2str(handles.scope_max),']);']);
set(handles.scope_max_edit,'String',num2str(handles.scope_max));
set(handles.scope_min_edit,'String',num2str(handles.scope_min));
guidata(gcbo,handles);


function scope_max_edit_Callback(hObject, eventdata, handles)
handles.scope_max = eval(get(hObject,'string'));
evalin('base',['current_scope.set(''YLimit'', [',num2str(handles.scope_min),',',num2str(handles.scope_max),']);']);
guidata(gcbo,handles);


function scope_min_edit_Callback(hObject, eventdata, handles)
handles.scope_min = eval(get(hObject,'string'));
evalin('base',['current_scope.set(''YLimit'', [',num2str(handles.scope_min),',',num2str(handles.scope_max),']);']);
guidata(gcbo,handles);



function Input_Command_Select_Panel_SelectionChangeFcn(hObject, eventdata, handles)
% Selects which input will be tied to the INPUT_COMMAND A/D channel. Used
% to take transfer functions using DSA
selection = get(handles.Input_Command_Select_Panel, 'SelectedObject');
switch get(selection,'Tag')
    case 'INPUT_COMMAND_SELECT_NONE'
        handles.INPUT_COMMAND_SELECT = 1;
    case 'INPUT_COMMAND_SELECT_AZ'
        handles.INPUT_COMMAND_SELECT = 2;
    case 'INPUT_COMMAND_SELECT_EL'
        handles.INPUT_COMMAND_SELECT = 0;
end
evalin('base',['fsm_comp_model.setparam(',num2str(handles.PID_INPUT_COMMAND_SELECT),',',num2str(handles.INPUT_COMMAND_SELECT),')']);
guidata(gcbo,handles); % Update handles structure
```

```
function AZ_QUAD_LOOP_GAIN_EDIT_Callback(hObject, eventdata, handles)
% Evaluates AZ Loop Gain input and saves to handles structure
% Also calculates new crossover frequency of Compensator and displays in
% fsm_comp
handles.AZ_QUAD_LOOP_GAIN = eval(get(hObject,'string'));
evalin('base',['fsm_comp_model.setparam(',num2str(handles.PID_AZ_QUAD_LOOP_GAIN),',',num2str(handles.AZ_QUAD_LOOP_GAIN),')']);
AZ_MODEL = FSM_AZ_QUAD_MODEL;
A_AZ_QUAD_CONTROLLER = evalin('base','A_AZ_QUAD_CONTROLLER;');
B_AZ_QUAD_CONTROLLER = evalin('base','B_AZ_QUAD_CONTROLLER;');
C_AZ_QUAD_CONTROLLER = evalin('base','C_AZ_QUAD_CONTROLLER;');
D_AZ_QUAD_CONTROLLER = evalin('base','D_AZ_QUAD_CONTROLLER;');
LT = handles.AZ_QUAD_LOOP_GAIN * AZ_MODEL * ss(A_AZ_QUAD_CONTROLLER, B_AZ_QUAD_CONTROLLER, C_AZ_QUAD_CONTROLLER, D_AZ_QUAD_CONTROLLER);
%crossover_freq = round(crossover(LT,0,5,5000)/(2*pi));
%set(handles.AZ_QUAD_CROSSOVER_EDIT,'String',num2str(crossover_freq));
guidata(gcbo,handles); % Update handles structure


function EL_QUAD_LOOP_GAIN_EDIT_Callback(hObject, eventdata, handles)
% Evaluates EL Loop Gain input and saves to handles structure
% Also calculates new crossover frequency of Compensator and displays in
% fsm_comp
handles.EL_QUAD_LOOP_GAIN = eval(get(hObject,'string'));
evalin('base',['fsm_comp_model.setparam(',num2str(handles.PID_EL_QUAD_LOOP_GAIN),',',num2str(handles.EL_QUAD_LOOP_GAIN),')']);
EL_MODEL = FSM_EL_QUAD_MODEL;
A_EL_QUAD_CONTROLLER = evalin('base','A_EL_QUAD_CONTROLLER;');
B_EL_QUAD_CONTROLLER = evalin('base','B_EL_QUAD_CONTROLLER;');
C_EL_QUAD_CONTROLLER = evalin('base','C_EL_QUAD_CONTROLLER;');
D_EL_QUAD_CONTROLLER = evalin('base','D_EL_QUAD_CONTROLLER;');
LT = handles.EL_QUAD_LOOP_GAIN * EL_MODEL * ss(A_EL_QUAD_CONTROLLER, B_EL_QUAD_CONTROLLER, C_EL_QUAD_CONTROLLER, D_EL_QUAD_CONTROLLER);
%crossover_freq = round(crossover(LT,0,5,5000)/(2*pi));
%set(handles.EL_QUAD_CROSSOVER_EDIT,'String',num2str(crossover_freq));
guidata(gcbo,handles); % Update handles structure


function MODEL_NAME_EDIT_Callback(hObject, eventdata, handles)
% Evaluates input string for model name, saves to handle structure
% Also saves name to MATLAB base environment for recovery if fsm_comp is
% restarted
handles.MODEL_NAME = get(hObject,'string');
evalin('base',['CURRENT_MODEL = ''',handles.MODEL_NAME,''';']);
guidata(gcbo,handles); % Update handles structure
```

Appendix G

MATLAB GUI Code for Running the Square Spiral Algorithm

```
% --- Executes on button press in spiralize.
function spiralize_Callback(hObject, eventdata, handles)

% Get xPC Parameter IDs
pid_off = evalin('base','tg.getparamid(''Spiral Off'',''Value'')');
pid_reset = evalin('base','tg.getparamid(''Spiral Locater/Spiral Reset'',''Value'')');
pid_finished = evalin('base','tg.getsignalid(''Spiral Locater/spiral/p8'');'); %Spiral Finished
pid_acquired = evalin('base','tg.getsignalid(''Spiral Locater/Spiral Acquired'');');
pid_stop = evalin('base','tg.getparamid(''Spiral Locater/Spiral Stop'',''Value'');');
pid_step = evalin('base','tg.getparamid(''Spiral Locater/Spiral Step'',''Value'');');
pid_spAZ = evalin('base','tg.getsignalid(''Spiral AZ'')');
pid_spEL = evalin('base','tg.getsignalid(''Spiral EL'')');
pid_spAZq = evalin('base','tg.getsignalid(''Quad AZ'')');
pid_spELq = evalin('base','tg.getsignalid(''Quad EL'')');
pid_spOS = evalin('base','tg.getsignalid(''Optical Sum'')');
pid_Cage = evalin('base','tg.getparamid(''Cage Enable'',''Value'')');
pid_EL = evalin('base','tg.getparamid(''EL Not Invert'',''Value'')');
pid_command = evalin('base','tg.getparamid(''Spiral Locater/Command On'',''Value'')');

% Update GUI to show Spiraling
set(handles.spiral_on,'Value',1)
set(handles.spiral_off,'Value',0)

% Put Spiral in reset state, turn on Commands
% so that Spiral now controls the outputs to the controller
% Start by taking out of reset state
evalin('base',['tg.setparam(',num2str(pid_reset),',1)'])
evalin('base',['tg.setparam(',num2str(pid_off),',0)'])
evalin('base',['tg.setparam(',num2str(pid_command),',1)'])
evalin('base',['tg.setparam(',num2str(pid_reset),',0)'])

% wait until the finished signal comes back or
% user presses the spiral off button on the GUI to stop
while (evalin('base',['tg.getsignal(',num2str(pid_finished),');'])==0)
    evalin('base','pause(.1);');
    if get(handles.spiral_off,'Value')==1
        break
```

```
    end
end

% If the bumpless transfer was successful, set digital
% switches to quad cell tracking mode
% otherwise set to kaman tracking mode
if evalin('base',['tg.getsignal(',num2str(pid_acquired),');'])==1
        evalin('base',['tg.setparam(',num2str(pid_Cage),',0)']);
        evalin('base',['tg.setparam(',num2str(pid_EL),',5.26)']);
        set(handles.AZTrack_ELTrack,'Value',1);
        set(handles.AZCommandT_edit,'String',num2str(0));
        set(handles.ELCommandT_edit,'String',num2str(0));
else
        evalin('base',['tg.setparam(',num2str(pid_Cage),',5.26)']);
        evalin('base',['tg.setparam(',num2str(pid_EL),',5.26)']);
        set(handles.AZCage_ELCage,'Value',1);
end

% turn off the spiral command control
% output the acquired signal to the user
% update the GUI to show spiral is off
evalin('base',['tg.setparam(',num2str(pid_command),',0);']);
evalin('base',['tg.getsignal(',num2str(pid_acquired),')'])
set(handles.spiral_off,'Value',1)
guidata(gcbo,handles);
```

Appendix H

MATLAB GUI Code for Running the CAV and CLV Spiral Algorithms

```
% --- Executes on button press in spiralize.
function spiralize_Callback(hObject, eventdata, handles)
% hObject    handle to spiralize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get xPC Parameter IDs
pid_off = evalin('base','getparamid(tg,'''','''Spiral_Off''')');   %ID to turn spiral off
pid_reset = evalin('base','getparamid(tg,'''','''Spiral_Reset''')');
pid_finished = evalin('base','tg.getsignalid(''Spiral Locater 2/spiral/p8'');');
pid_acquired = evalin('base','tg.getsignalid(''Spiral Locater 2/spiral/p7'');');
pid_AZ_cage_command = evalin('base','tg.getsignalid(''Spiral Locater 2/AZ Cage Command Switch'');');
pid_EL_cage_command = evalin('base','tg.getsignalid(''Spiral Locater 2/EL Cage Command Switch'');');
pid_stop = evalin('base','getparamid(tg,'''','''Spiral Locater 2/Spiral Stop'');');
pid_step = evalin('base','getparamid(tg,'''','''Spiral Locater 2/Spiral Step'');');
pid_spAZ = evalin('base','tg.getsignalid(''Spiral AZ'')');
pid_spEL = evalin('base','tg.getsignalid(''Spiral EL'')');
pid_spAZq = evalin('base','tg.getsignalid(''Quad AZ'')');
pid_spELq = evalin('base','tg.getsignalid(''Quad EL'')');
pid_spOS = evalin('base','tg.getsignalid(''Optical Sum'')');
pid_Cage = evalin('base','getparamid(tg,'''','''Cage_Enable''')');
pid_EL = evalin('base','getparamid(tg,'''','''EL_Not_Invert''')');
pid_command = evalin('base','getparamid(tg,'''','''Spiral_Command_On''')');

% Update GUI to show Spiraling
set(handles.spiral_on,'Value',1)
set(handles.spiral_off,'Value',0)

% Put Spiral in reset state, turn on Commands
% so that Spiral now controls the outputs to the controller
% Start by taking out of reset state
evalin('base',['tg.setparam(',num2str(pid_reset),',1)'])
evalin('base',['tg.setparam(',num2str(pid_off),',0)'])
```

```
evalin('base',['tg.setparam(',num2str(pid_command),',1)'])
evalin('base',['tg.setparam(',num2str(pid_reset),',0)'])

wait until the finished signal comes back or
user presses the spiral off button on the GUI to stop
while (evalin('base',['tg.getsignal(',num2str(pid_finished),');'])==0)
    evalin('base','pause(.05);');
    if get(handles.spiral_off,'Value')==1
        break
    end
end

% If in Acquisition mode, set digital switches as appropiate and end
if handles.spiral_mode == 2
    if evalin('base',['tg.getsignal(',num2str(pid_acquired),');'])==1
        evalin('base',['tg.setparam(',num2str(pid_Cage),',0)']);
        evalin('base',['tg.setparam(',num2str(pid_EL),',5.26)']);
        set(handles.AZTrack_ELTrack,'Value',1);
        set(handles.AZCommandT_edit,'String',num2str(0));
        set(handles.ELCommandT_edit,'String',num2str(0));
    else
        evalin('base',['tg.setparam(',num2str(pid_Cage),',5.26)']);
        evalin('base',['tg.setparam(',num2str(pid_EL),',5.26)']);
        set(handles.AZCage_ELCage,'Value',1);
    end
% If not in Acquisition Mode but found the Quad Cell,
% Set the new starting place at where you are, then
% re-spiralize setting the spiral_size and dist_threshold to
% 1/10th their original values. This allows a course spiral to find the
% Quad Cell, followed by a tighter spiral to get as close to center as
% possible
% If didn't find Quad Cell, just end
elseif handles.spiral_mode == 1
    if evalin('base',['tg.getsignal(',num2str(pid_acquired),');'])==1
        AZ_cage = evalin('base',['tg.getsignal(',num2str(pid_AZ_cage_command),');']);
        EL_cage = evalin('base',['tg.getsignal(',num2str(pid_EL_cage_command),');']);
        evalin('base',['tg.setparam(',num2str(pid_command),',0);']);
        pid_AZ_bias = evalin('base','getparamid(tg,''',''AZ_Bias'')');
        evalin('base',['tg.setparam(',num2str(pid_AZ_bias),',',num2str(AZ_cage),')']);
        pid_EL_bias = evalin('base','getparamid(tg,''',''EL_Bias'')');
        evalin('base',['tg.setparam(',num2str(pid_EL_bias),',',num2str(EL_cage),')']);
        handles.ELBias = EL_cage;
        handles.AZBias = AZ_cage;
        handles.AZCommand = 0;
        evalin('base',['tg.setparam(',num2str(pid_AZ_bias),',',num2str(handles.AZBias),')']);
        set(handles.AZBias_edit,'String',num2str(handles.AZBias));
        set(handles.AZCommand_edit,'String',num2str(handles.AZCommand));
        handles.ELCommand = 0;
        evalin('base',['tg.setparam(',num2str(pid_EL_bias),',',num2str(handles.ELBias),')']);
        set(handles.ELBias_edit,'String',num2str(handles.ELBias));
        set(handles.ELCommand_edit,'String',num2str(handles.ELCommand));

        % setting spiral_size and distance_threshold parameters
        pid_Spiral_Size = evalin('base','getparamid(tg,''',''Spiral_Size'');')
        evalin('base',['tg.setparam(',num2str(pid_Spiral_Size),',',num2str(handles.spiral_size/10),');']);
        pid_Spiral_Dist = evalin('base','getparamid(tg,''',''Spiral_Dist_Threshold'');');
        evalin('base',['tg.setparam(',num2str(pid_Spiral_Dist),',',num2str(handles.dist/10),')']);

        % restarting spiral
        evalin('base',['tg.setparam(',num2str(pid_reset),',1)'])
        evalin('base',['tg.setparam(',num2str(pid_off),',0)'])
```

```
        evalin('base',['tg.setparam(',num2str(pid_command),',1)'])
        evalin('base','Location_Scope = getscope(tg,5);');
        evalin('base',['tg.setparam(',num2str(pid_reset),',0)'])

        while (evalin('base',['tg.getsignal(',num2str(pid_finished),');'])==0)
            evalin('base','pause(.05);');
            if get(handles.spiral_off,'Value')==1
            break
            end
        end

        if evalin('base',['tg.getsignal(',num2str(pid_acquired),');'])==1
            AZ_cage = evalin('base',['tg.getsignal(',num2str(pid_AZ_cage_command),');']);
            EL_cage = evalin('base',['tg.getsignal(',num2str(pid_EL_cage_command),');']);
            evalin('base',['tg.setparam(',num2str(pid_command),',0);']);
            pid_AZ_bias = evalin('base','getparamid(tg,'''','''AZ_Bias'')');
            evalin('base',['tg.setparam(',num2str(pid_AZ_bias),',',num2str(AZ_cage),')']);
            pid_EL_bias = evalin('base','getparamid(tg,'''','''EL_Bias'')');
            evalin('base',['tg.setparam(',num2str(pid_EL_bias),',',num2str(EL_cage),')']);
            handles.ELBias = EL_cage;
            handles.AZBias = AZ_cage;
            handles.AZCommand = 0;
            evalin('base',['tg.setparam(',num2str(pid_AZ_bias),',',num2str(handles.AZBias),')']);
            set(handles.AZBias_edit,'String',num2str(handles.AZBias));
            set(handles.AZCommand_edit,'String',num2str(handles.AZCommand));
            handles.ELCommand = 0;
            evalin('base',['tg.setparam(',num2str(pid_EL_bias),',',num2str(handles.ELBias),')']);
            set(handles.ELBias_edit,'String',num2str(handles.ELBias));
            set(handles.ELCommand_edit,'String',num2str(handles.ELCommand));
        end
        % set parameters back to original values
        evalin('base',['tg.setparam(',num2str(pid_Spiral_Size),',',num2str(handles.spiral_size),');']);
        evalin('base',['tg.setparam(',num2str(pid_Spiral_Dist),',',num2str(handles.dist),')']);
    end
end

% Show User if Acquisition was sucessful
evalin('base',['tg.setparam(',num2str(pid_command),',0);']);
evalin('base',['tg.getsignal(',num2str(pid_acquired),')'])
set(handles.spiral_off,'Value',1)
guidata(gcbo,handles);
```

Appendix I

Vendors

Agilent Technologies, Inc.
395 Page Mill Rd.
Palo Alto, CA 94306 Telephone – (877) 424-4536
Web address – http://www.home.agilent.com
Product Used: 35670A Dynamic Signal Analyzer

General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Telephone – (800) 653-9970
Web address – http://www.generalstandards.com
Product Used: PMC-ADADIO PCI Card

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098
Telephone – (508) 647-7000
Web address – http://www.mathworks.com
Products Used: MATLAB, Simulink, xPC

Tektronix Texas, LLC
1500 North Greenville Avenue
Richardson, TX 75081
Telephone – (800) 969-4638
Web address – http://www.tektronix.com
Product Used: TDS5104B Digital Phosphor Oscilloscope

UDT Instruments

727 South Wolfe Street

Baltimore, MD 21231

Telephone – (410) 342-2626

Web address – http://www.udtinstruments.com

Product Used: Model 431 X-Y Optical Position Indicator

UDT Sensors, Inc.

12525 Chadron Avenue

Hawthorne, CA 90250

Telephone – (310) 978-0516

Web address – http://www.udt.com

Product Used: SPOT 4DMI Quad Cell

Bibliography

[1] Applied Technology Associates, *ARS-12B Datasheet*, http://www.atasensors.com/Sensors2/pdf_files/2005 New Data Sheets/ARS-12B/ARS-12B MHD Angular Rate Sensor.pdf

[2] Agarwal, Anant and Lang, Jefferey H., *Foundations of Analog and Digital Electronic Circuits*, (MIT 6.002 Reference, 2002).

[3] The Mathworks, Inc., *Stateflow User's Guide*, `http://www.mathworks.com/access/helpdesk/help/pdf_doc/stateflow/sf_ug.pdf`

[4] The Mathworks, Inc., *Using Simulink*, `http://www.mathworks.com/access/helpdesk/help/pdf_doc/simulink/sl_using.pdf`

[5] The Mathworks, Inc., *xPC Target User's Guide*, `http://www.mathworks.com/access/helpdesk/help/pdf_doc/xpc/xpc_target_ug.pdf`

[6] Palm III, William J., *Modeling, Analysis and Control of Dynamic Systems*, (New York, New York: John Wiley & Sons, 2000).

[7] Loney, G.C., *High Bandwidth Steering Mirror Research*, (Lincoln Laboratory Project Report IRP-15, 1992).

[8] Van de Vegte, John, *Feedback Control Systems*, (Englewood Cliffs, New Jersey: Prentice Hall, 1994).