Tilt-Compensated Magnetic Field Sensor

Adam N. Bingaman

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

Master of Science
In
Electrical Engineering

William T. Baumann, Chair
Aloysius A. Beex
William A. Davis
Bradley A. Davis

13 May 2010
Blacksburg, Virginia

Keywords: magnetometer, inclinometer, compass, tilt-compensation, median filter, digital filter, CORDIC

Tilt-Compensated Magnetic Field Sensor

Adam N. Bingaman

(Abstract)

Motion and tilt have long hindered the accuracy, reliability, and response of magnetic detection systems. Perturbations in the magnetic field reading resulting from motion cause degradation of the output signal, compromising the performance and reliability of the magnetometer system.

The purpose of this document is to describe the development, construction, and testing of a tilt-stabilized three-axis magnetic field sensor. The sensor is implemented as a three-axis general-purpose magnetic field sensor, with the additional capability of being implemented as a compass. Design and construction of system hardware is discussed, along with software development and implementation.

Finite impulse response filters are designed and implemented in hardware to filter the acquired magnetic signals. Various designs of median filters are simulated and tested for smoothing inclination signal irregularities and noise.

Trigonometric conversions necessary for tilt-compensation are calculated in software using traditional methods, as well as the Coordinate Rotation Digital Computer (CORDIC) algorithm. Both calculation methods are compared for execution time and efficiency.

Successful incorporation of all design aspects leads to detection and output of stable earth magnetic fields, sinusoidal signals, and aperiodic signatures while the magnetometer system is subject to significant tilt motion. Optimized system execution time leads to a maximum detectable signal bandwidth of 410 Hz. Integration of azimuth angle calculation is incorporated and is successfully tested with minimal error, allowing the system to be used as a compass.

Results of the compensated system tests are compared to non-compensated results to display system performance, including tilt-compensation effectiveness, noise attenuation, and operational speed.

Table of Contents

LIST OF FIGURES

All photographs by author, 2010.

LIST OF TABLES

NOMENCLATURE

B	Magnetic Flux Density
H	Magnetic field intensity
D	Electric flux density
E	Electric field intensity
J	Current density
ρ_v	Electric charge density
ε_0	Vacuum permittivity
μ_0	Vacuum permeability
α	Alpha-trimmed median filter tuning coefficient
n	Digital sample index
N	Digital filter window width
k	Lower-upper-middle smoothing filter tuning coefficient
l	Lower-upper-middle sharpening filter tuning coefficient
m	Number of iterations considered in CORDIC algorithm
g	Number of samples per FIR section in FIR Median Hybrid filter

Chapter 1 Introduction

This thesis presents the design, construction, testing, and analysis of a tilt-stabilized magnetic sensor, or magnetometer. In this chapter, the problems associated with tilting a magnetometer system are explained. Additionally, the project research approach and organization are presented.

1.1 Problem Statement

Information about the magnetic fields within an environment can be extremely useful for many different purposes. The strength and orientation of a magnetic field allows for accurate navigation, as well as proximity detection of nearby ferrous objects or other stray magnetic sources.

Physical stability is crucial for reliable and accurate readings from any magnetometer or compass. Tilting motion causes the magnetometer to move throughout the surrounding magnetic field, causing output fluctuations in either field magnitude, direction, or both. Where a stable base is not possible, as in planes, jets, boats, automobiles, or any mounting device subject to motion, compensation for the magnetic sensor motion and tilt must be conducted.

For this thesis, a tilt-compensated magnetic sensor capable of reading the magnitude of the surrounding magnetic field in three Cartesian coordinate axes is designed, constructed, tested, and evaluated. The system's ability to compensate for tilt in magnetic field readings is analyzed after finite impulse response digital filtering, impulse-removing median filtering, tilt compensation and execution time optimization algorithms have been applied.

1.2 Review of Literature

Magnetic sensors have been the target of significant advances in technology in recent decades. However, simple magnetic sensors have been used by mankind for navigational and orienteering purposes for centuries in the form of the compass. Navigating by ship, flying aircraft, or driving a motorized vehicle requires navigational skills, and until the recent development of the global positioning system (GPS), the compass and map have been the tools of choice. Detection of the navigator's heading with respect to the Earth's magnetic field allows for insight into the direction and heading of the vehicle. A weakness of the compass or any magnetic sensor, however, has been the presence of motion. Tilt resulting from motion of the sensor within the surrounding magnetic field causes inaccurate and unreliable readings, compromising the magnetometer's performance. Compensation of tilt using accelerometers and inclinometers has been previously explored and using several different methods. Invariably, when not mechanically stabilized, the tilt of the magnetometer or compass system is found and the vector calculations to compensate for the tilt are then implemented. With the development and widespread use of the digital

microcontroller, calculation and compensation have become fast and accurate processes. Extensive work has been done using Field Programmable Gate Arrays (FPGA) to implement trigonometric calculation along with sensor interfacing to quickly and accurately compensate for magnetic sensor tilt. This has been the most desirable method for this calculation, due to the FPGA's principal advantage of being an application-specific integrated circuit, found by Laulainen et al [1]. Implementation of the digital microcontroller has been explored and evaluated as a formidable alternative to FPGA magnetometer systems.

To most accurately compensate for tilt error, noise in the magnetic field readings is maximally attenuated, and the tilt angle of the sensor itself must be calculated with minimal error. The algorithms that accomplish these tasks require additional microcontroller instruction cycles. Therefore a longer amount of time is required as the accuracy, reliability and effectiveness of the magnetic field reading calculations are improved. The goal of this project is to outline and present the findings of designing, building, and programming of a tilt-compensated magnetic field sensor with the added capability of use as a compass. The magnetometer system implements fast and effective filtering, compensation, and calculation algorithms on a simple and inexpensive microcontroller.

To further understand the requirements of a tilt-compensated system, the physical and electrical makeup of the magnetic sensors themselves are explored. Implemented in this project are anisotropic magnetoresistive (AMR) sensors from Honeywell International [2, 3]. AMR sensors are composed of permalloy thin films in a Wheatstone bridge setup; thus when magnetic fields are incident on the resistors, the direction of current flow is altered, changing the resistance of the sensors [2-4]. This characteristic of magnetoresistors makes them sensitive to magnetic fluctuation in one direction, and thus with multiple sensors, a multiple axis sensor system can be implemented [5-8]. Finer detail into the physical construction and characteristics of the AMR sensor as well as how to implement them as transducers and sensors is presented by Holman [4]. The dynamic detectable magnetic field range of an AMR sensor ranges from approximately 6 Gauss down to several microGauss (10^{-6}), dependent on manufacturer, specification, and construction [5], The bandwidth of AMR sensors is typically on the order of less than 1 kHz [2, 3]. One characteristic of the AMR thin film materials is degradation of the magnetic domains with time and outside interference from a magnetic field of large magnitude [6]. To ensure the sensitivity of the magnetoresistive sensors is maximized, occasional reversal and realignment of the magnetic domains is suggested. This is accomplished via several proposed transistor and capacitor switching circuits. The purpose of the switching circuit is to sink a short current pulse through resistive straps in the magnetic sensor chips, thus producing a large magnetic field pulse to reverse and realign the magnetic domains, increasing the sensitivity of the sensors [6].

Inclinometers output a voltage or current based on their inclination, based on the angle of inclination with respect to the force vector of gravity [7]. Calculation of the output angle of the inclinometer varies with manufacturer, model, construction, and damping of the output signal. To calculate the tilt of the magnetometer system for this project, inclinometers from VTI

Technologies are implemented. These inclinometers include an overdamped microelectromechanical system (MEMS) with a bandwidth of 18 Hz. Factory tolerance of ±0.0025 degrees of inclination is guaranteed [7].

Compensating for the tilt angle of the magnetic sensor requires readings from both the inclinometers in at least two axes of rotation (roll and pitch) [1, 10-13]. From the angle of inclination, the magnetic vector magnitude detected in each of the three Cartesian coordinate axes can be transformed back to the horizontal plane, relative to gravity, by several trigonometric calculations. These calculations include only sine and cosine calculations of the roll and pitch angles. With an arctangent calculation of the resultant 2-axis magnetic conversion, a heading, or azimuth angle, can be calculated in the 360 degree horizontal plane [1, 11-14]. Heading calculated from such compensated magnetic field vectors varies dependent on the sign of the two compensated horizontal axes' output [1, 8]. Readings of the magnetic field can be altered by many outside sources, including stray currents, ferrous objects, temperature drift, sensor bandwidth, or field distortions. The errors associated with all of these sources of magnetic interference are estimated and considered in the tolerance of system design [9, 10]. Noise filtering can be achieved by any digital or analog filter, and a sampling frequency of 250 Hz has proven sufficient for compass-specific noise filtering for the magnetic sensor's bandwidth [1].

To accurately calculate the tilt compensation of the magnetic field readings, the output of the inclinometers must be as noise free as possible. Upon a high-speed change in inclination angle, large acceleration or deceleration impulses result. When the high-frequency content of this step-like input reaches the limits of the bandwidth of the inclinometer, a high frequency output can be expected from the overdamped inclination sensors[11, 12]. As this high-frequency content exceeds the frequency of the sensor bandwidth, an impulse-like response is produced. Traditional linear filters are inadequate for the purpose of impulse attenuation, as a transient response is output with an impulse filter input [13].

To account for this high-frequency content of the possible input signal for the inclinometer, it has been documented that median filters of varying types prove sufficient for eliminating or attenuating the impulse-like response of the signal [14-16]. Providing high-frequency attenuation while still maintaining a low-noise and transient-free response is just one characteristic property of the non-linear median-based filter. Properties, characteristic tradeoffs, responses, and implementations have been documented for various filter architectures and specifications, including location in time of median values, weights, and filter structure [14-17]. During the last half-century, the median filter has been the target of significant research, and numerous variations of the first explored center-weight and alpha-trimmed filters have been developed and explored [14, 15]. Some of these variations can be implemented for the purpose of inclinometer signal filtering, to eliminate the possibility of high-frequency content of the inclination signal with their outlier rejection capabilities.

One such variation of the median filter is the alpha-trimmed median filter. One of the most basic median filter architectures, the α-trimmed median filter is a smoothing-type filter. Requiring only sorting of an input data array, the α-trimmed median filter is an efficient filter for smoothing impulse or outlier-laden data [15, 16].

Numerous variations of the median filter have been developed by Heinonen that incorporate FIR filter substructures. This FIR implementation allows for fast filter calculation with simple execution routines, and proves to be valuable in impulse attenuation while maintaining the sharpness of the desired signal corners. Output of the FIR-based median filters is calculated as the median of a varying number of FIR substructures. The substructures can be weighted, trimmed, or mean calculations, based on the FIR coefficient values of the system. Characteristics of the filter are reflected in the nature of the output, as well as the attenuation of components of the input signal, and have been explored and documented by Heinonen and Neuvo [18-20].

Several other variations of the median filter have been developed and implemented by Hardie and Broncelet called Lower-Upper-Middle (LUM) filters [21]. These median filters are highly versatile in their ability to alter their characteristics based on the required amount of smoothing or sharpening of the signal being filtered. In a LUM smoother, the data points considered in the output are closer to the center of the data array, and thus outliers are eliminated. In a LUM sharpener, data points considered in the output are taken from the end (chronologically newest and oldest) portions of the signal window, and thus high frequency perturbations are preserved and the signal is sharpened. LUM filters possess the useful characteristic of being capable of creating hybrid models of the two types of filter for the purpose of balancing smoothing and sharpening of the desired signal. These entire filter architectures are useful for one-dimensional signals as well as two-dimensional signals, and thus have significant relevance in the field of image processing and enhancement [21].

After the angle of inclination is calculated, tilt compensation can commence, as discussed above. Trigonometric calculations of sine, cosine, tangent can be easily executed with microcontroller programming. Due to the large amount of sine and cosine calculations needed for the tilt compensation [1, 8, 22-24], the bandwidth of the magnetometer system can be compromised. Developed and documented by Volder, the Coordinate Rotation Digital Computer (CORDIC) algorithm is capable of significantly reducing the instruction execution time of trigonometric and hyperbolic calculations versus floating point or traditional integer calculations [25]. Hence, CORDIC is explored as a possible alternative for calculating these trigonometric values. The CORDIC algorithm iteratively calculates trigonometric function results based on a set number of iterations and conditions, usually limited by the number of bits of the digital system. Investigation of the convergence, accuracy, and implementation of the CORDIC algorithm was conducted by Walther [26]. Numerous other explorations of the algorithm revealed analysis of the iteration process and precision of the iterative process [27], and expansions of the architecture of the algorithm and quantization analysis have also been explored [28-30].

Magnetic field readings for the tilt-compensated magnetometer are acquired by Honeywell International's HMC2003 module. A three-axis magnetometer system, the module contains onboard lowpass filters for noise attenuation. Additional filtering of the magnetic signals is achieved via finite impulse response (FIR) filters of various orders. Abundant amounts of literature covering FIR and other digital filters are available. Estimation on FIR filter order, structure, and characteristics can be found in Proakis and Manolakis [13, 31]. General FIR filter structure, as well as implementations and applications are outlined by Smith [31].

1.3 Approach and Results

This thesis develops a three-axis, tilt-compensated magnetometer system capable of stable and accurate magnetic field reading when perturbed with motion.

Low-cost anisotropic magnetoresistive sensors are implemented in system hardware, and tilt is acquired via two inclinometers positioned in the horizontal plane. Filtering of the inclination signal is accomplished via a median-based filter, determined from simulation and onboard implementation as the most suitable filter to smooth and eliminate high-frequency motion impulses in the signal. Finite impulse response filtering is implemented on the magnetic field reading of all three axes, due to the advantage of ease and flexibility of design while matching the relatively large phase delay of the inclinometer's median filter. The Coordinate Rotation Digital Computer (CORDIC) algorithm is explored as an alternative to traditional trigonometric calculation to quickly and efficiently calculate the trigonometric constants associated with the tilt of the system. Simple trigonometric equations are executed to compensate for tilt in the magnetic readings.

Test results, including system effectiveness and error calculation are analyzed in subsequent chapters.

1.4 Outline of Thesis

Chapter 2 presents the basics of electromagnetics, including magnetic fields. Effects of tilt within a magnetic field are discussed, and the sources of possible errors in field continuity are discussed in detail.

Chapter 3 discusses the hardware implemented in the magnetometer system. First, anisotropic magnetic sensors are explored, including theory, construction, implementation, and necessary interface procedures to maximize performance. Next, inclinometer theory, as well as operation, implementation, and error are discussed. Microcontroller and general hardware connections and specifications follow, with emphasis on data line connections, resolution, and processor speed. Finally, construction and testing of the tilt magnetometer platform is discussed.

Chapter 4 presents analysis of the signal processing aspects of the magnetometer system. First, the necessity of median filtering is discussed along with simulation, analysis, implementation, and testing of various architectures of the nonlinear-type filters. Next, simulation, implementation and testing of digital finite impulse response filters are explored with analysis of results following.

Chapter 5 discusses the trigonometric calculations, including exploring the CORDIC algorithm for fast and efficient execution of trigonometric calculations for the purpose of tilt compensation. Theory of the algorithm is presented with detailed explanation of advantages and disadvantages of each algorithm's execution. Error, resolution, and convergence of CORDIC are discussed with testing, results, and analysis closing the chapter.

Chapter 6 outlines the calculations required for tilt-compensation. Integration of the compensation equations is discussed, and calibration of the magnetometer system for maximum performance is discussed. Results and analysis of calibrated and compensated tests are provided.

Chapter 7 provides results of filtered and compensated readings of steady magnetic fields with varying types of motion stimulation. Tilt-stabilization effectiveness results are discussed with analysis of magnetic field readings with and without tilt-stabilization. System bandwidth from sampling frequency and execution time are discussed. Implementation of the magnetometer system as a compass is tested and accuracy is analyzed. In addition, varying magnetic fields are presented to the system displaying the capability of the magnetometer to detect fluctuations while in motion.

Chapter 8 closes with conclusion and recommended future work. Possible expansions, improvements, and changes necessary to improve the tilt-stabilized magnetometer are provided.

Chapter 2 Magnetic Fields

This chapter provides basic electromagnetic field theory with emphasis on magnetic field properties. Information about magnetic sources, including earth field, materials, and interference are discussed.

2.1 Origins and Equations

Iron oxide ore, also known as lodestone, was found to have magnetic properties many centuries ago in the region of Turkey, or as it was known then, Magnesia. It was found that magnets both attracted and repelled one another based on their orientation to each other. Several theories into the cause of this phenomenon arose amongst significant research between the 16^{th} and 18^{th} centuries [4]. Still not completely understood today, the movement of electrons within the magnetic material is widely accepted as the source of magnetic fields, whether it is from electric current, or the electrons making up the magnetic moment of natural or artificial magnets. Significant developments in electromagnetic theory were developed by Ampere, Faraday, and Gauss, among others [32-34].

The basis of all electromagnetic theory was developed from Ampere, Gauss, and Faraday's work by Maxwell with his four equations of static electric fields and steady magnetic fields.

$$\nabla \cdot \mathbf{D} = \rho_v \tag{2.1}$$

$$\nabla \times \boldsymbol{E} = 0 \tag{2.2}$$

$$\nabla \times \boldsymbol{H} = \boldsymbol{J} \tag{2.3}$$

$$\nabla \cdot \boldsymbol{B} = 0 \tag{2.4}$$

Relating **D** and **E** in a vaccuum, as well as **B** and **H** are the following equations:

$$\boldsymbol{D} = \varepsilon_0 \boldsymbol{E} \tag{2.5}$$

$$\boldsymbol{B} = \mu_0 \boldsymbol{H} \tag{2.6}$$

A magnetic field is described as a vector, and thus has direction and magnitude. Originating from the poles of a magnet, equipotential magnetic flux lines span from one pole to the other (north to south), as seen in Figure 2.1.

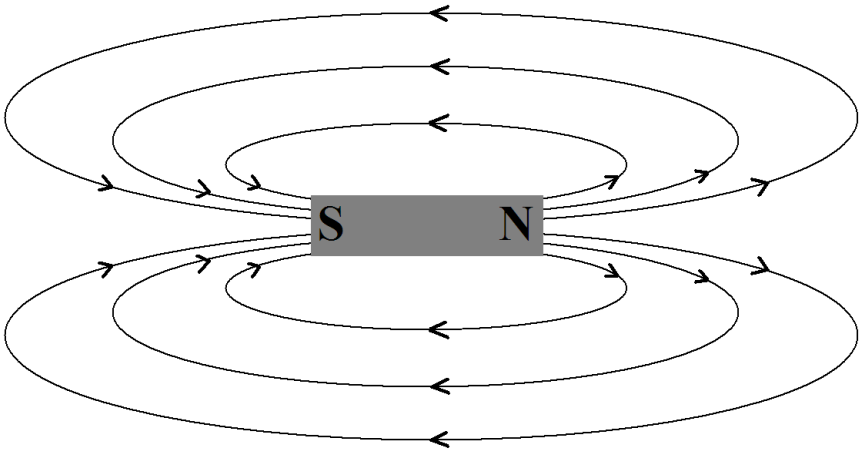


Figure 2.1: Magnetic field of a bar magnet.

As radial distance from the magnetic source increases, the magnitude of the magnetic field decreases. If distance from the magnetic source increases or decreases normal to the direction of polarization, the direction of the magnetic field lines is, ideally, unchanged.

Magnetic fields also result from electric current flowing through a conductor. Generated in a radial direction around the direction of current flow, magnetic field intensity is also inversely proportional to the distance from the conductor.

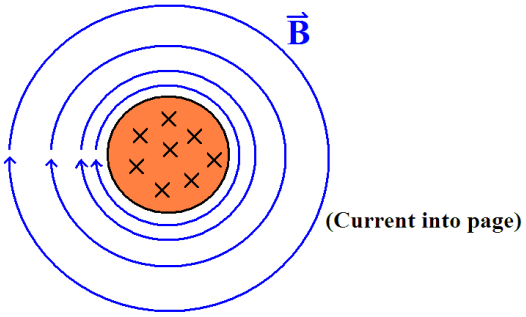


Figure 2.2: Magnetic field resulting from current flow.

2.2 Earth's Magnetic Field

The magnetic field of the Earth acts as a dipole field, with the north and south poles being the termination and source of the lines of magnetic flux. Tilted from true (rotational) north by 11.5 degrees [35], the earth field plays a vital role in deflecting particles from charged solar wind out of our atmosphere and into space.

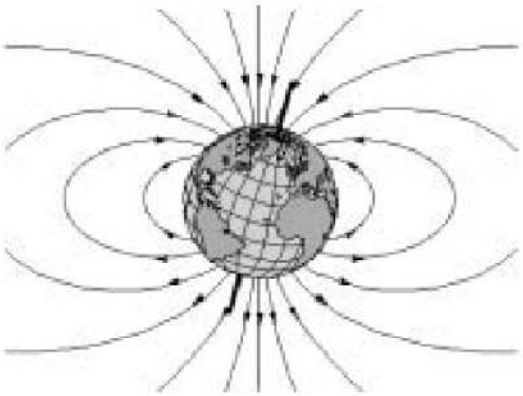


Figure 2.3: Magnetic field of the earth with permission from Honeywell International [9].

The source of earth's magnetic field is not 100% understood, even to this day. However, theory and experimentation by Elsasser, Bullard and Alfvén have supported Larmor's theory of a self-exciting dynamo process [36, 37]. This process theorizes that pressure changes and constant movement of the earth's crust induces electrical currents in magnetic materials contained in the core. This, as well as additional background information is discussed by Merrill, McElhinny and McFadden, as well as Gubbins and Herrero-Bervera [36, 37].

2.3 Tilt Effect

Magnetometers that detect fields in the three Cartesian axes detect the vector quantities of the surrounding magnetic field in these respective directions. In addition, by detecting the vector quantities of the two axes perpendicular to the vector force of gravity (horizontal plane), the heading, or azimuth angle of the magnetometer relative to the earth field can be calculated with the arctangent of the two magnitudes of the vector quantities. Assuming **x** and **y** are in the horizontal plane:

$$\text{Heading} = \tan^{-1}\left(\frac{\boldsymbol{y}}{\boldsymbol{x}}\right) \tag{2.7}$$

Upon tilting of the magnetometer, the direction of axial sensitivity has changed, and thus the vector quantities detected by the magnetometer change.

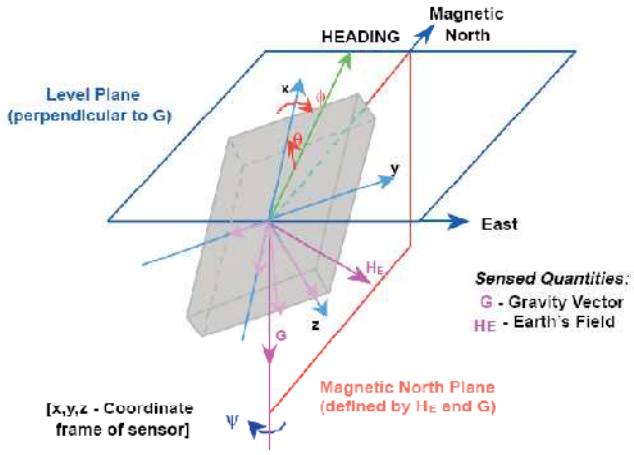


Figure 2.4: Tilted three-axis magnetic sensor and vector field quantities with permission from Honeywell International [9].

Trigonometric tilt compensation for motion is possible with proper tilt angle data. This calculation process allows for magnetic field detection and accurate heading calculation regardless of the tilt of the magnetometer itself.

2.4 Interference

Magnetic fields can be distorted, changed or subject to interference from many different sources. Nearby magnetic materials, such as iron, nickel, and cobalt cause distortions in the nearby magnetic field. The magnetic field lines bend toward the ferrous object, and thus a once uniform field now possesses directional distortion. This effect is the basis of the majority of electronic traffic detection systems, and has also been implemented in sensors that detect subterranean minerals in the fields of mining and petroleum extraction.

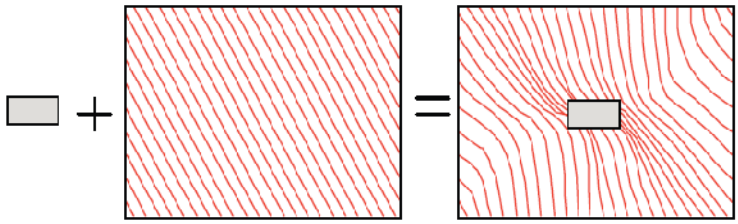


Figure 2.5: A ferrous object presented to a uniform magnetic field results in field distortion. With permission from Honeywell International [9].

External interference and distortion are also presented to Earth's magnetic field from cosmic sources. Solar flares eject charged particles into space, and upon colliding with the earth's outermost magnetic field, are deflected. This deflection distorts the shape of the magnetic field, and has been known to interfere with sensitive electronic devices.

Chapter 3 System Hardware

This chapter introduces the various pieces of hardware implemented in the magnetometer system. Including inclinometers, magnetic sensors, and the onboard microcontroller, interfacing of all components is discussed. Testing of the inclinometer and magnetic sensors is provided, with emphasis on directional sensitivity and possible sources of noise. In addition, design and construction of a wooden tilt-platform, on which the system hardware is mounted, is presented.

3.1 Magnetic Sensors

Magnetic sensors have been in use by the human race for centuries for navigation and orienteering. The earliest form of the magnetic sensor was the compass. In recent decades, numerous variations in materials and construction of magnetometers has allowed for not only direction readings, but highly sensitive vector magnetic field readings.

3.1.1 Origins

The simplest magnetic sensor is the needle compass, which is simply a small, well-balanced piece of magnetized material that lines up with surrounding magnetic fields. In 1857 William Thompson discovered magnetorestriction, or a resistive change in material when magnetic fields are presented. Discovered in an iron sample, this phenomenon was also found to be true in many other materials [4]. Further discovery and research led to numerous variations of the magnetic sensor.

Flux gate magnetometers are composed of a series of highly conductive windings, which produce a voltage when a magnetic flux is presented through the windings. Requiring little power to operate, flux gate sensors are widely used in many fields of study and industry [38].

Hall effect magnetic sensors utilize the Hall effect, which states that a torque is presented to a moving electron within a magnetic field. With the torque, a voltage is created, and the field magnitude is detected. Hall effect sensors typically produce low voltage-to-field outputs, and thus are widely used to detect magnetic fields of large magnitude [4, 38, 39].

Superconducting Quantum Interference Device (SQUID) magnetometers work using a thin insulating layer to separate two superconductors. Perturbations in the magnetic field produce a voltage across the insulated gap, making SQUID magnetometers some of the most sensitive devices available today [38, 39].

Several other more complicated magnetometers with varying degrees of sensitivity have also been developed and used successfully. Further discussion of these various magnetometers is available from Makovec and Ripka [35, 38].

3.1.2 Magnetoresistive Sensors

Magnetoresistive sensors are a simple and inexpensive form of the magnetic sensor. Magnetoresistive sensors are typically made out of permalloy, a nickel-iron alloy, which was found to have almost zero magnetorestriction [4]. Nanometers to microns thick, permalloy thin films have been found to have very useful sensor properties based on resistive changes when magnetic fields are applied.

3.1.2.1 Origin and Construction

Thin film permalloy magnetoresistive sensors are deposited on a substrate on a nanometer scale by any number of known methods (such as sputtering) [4, 38]. In the case of the anisotropic magnetoresistive sensors (AMR) used in this project, the substrate on which the film is deposited is the silicon wafer of the sensor chip. For a magnetic sensor to be sensitive in one direction, or axis, a magnetic field is presented at the time of deposition to align the magnetic domains [4, 32, 38, 39]. This magnetic alignment, or easy axis, is typically parallel to the lengthwise direction of the magnetoresistor. Current flowing through the thin film at a direction that creates angle Θ between the current flow and the magnetization of the material is presented with a resistance that is proportional to the angle Θ. When a magnetic field is applied, as in Figure 3.1, the magnetization is shifted from nominal, and the angle Θ between the current flow and the magnetization is changed. Thus, a change in resistance is the result.

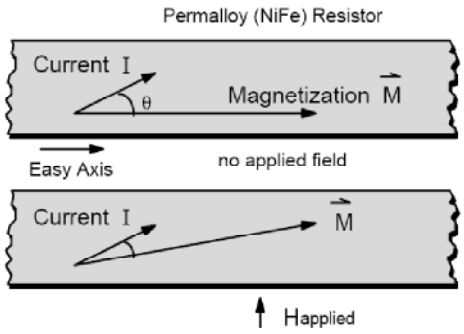


Figure 3.1: Magnetization direction change from applied magnetic field alters sensor resistance. With permission from Honeywell International [6].

To direct current in a non-parallel direction as described, a barber pole setup is implemented on the thin film resistor. Set at 45º angle to the lengthwise direction of the resistor are several parallel conductors. Thus, the shortest and least resistive path for current to flow lengthwise down the resistor is at a 45 º angle to the magnetization. In addition, to increase sensitivity, a Wheatstone bridge setup is employed on the chip, implementing four AMR sensors, and thus a differential output voltage can be detected proportional to this change in resistance.

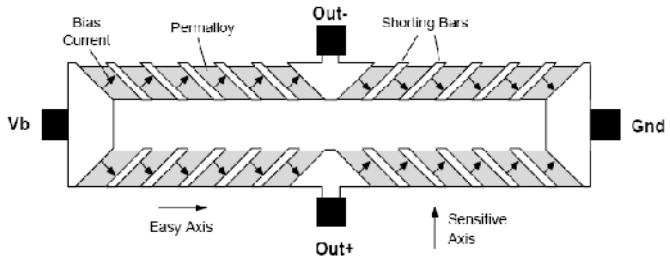


Figure 3.2: Wheatstone bridge architecture displaying the barber pole biasing of the AMR sensors with permission from Honeywell International [9].

3.1.2.2 Set/Reset Pulse

After initial factory magnetization of the easy (sensitive) axis of the AMR sensors, the magnetic domains that were aligned to create the magnetization can drift, or demagnetize. Upon strong external magnetic fields, or over time due to thermal drift, the magnetization of the domains becomes misaligned, and the easy axis sensitivity of the magnetoresistive sensors decreases.

To alleviate this problem, a strong magnetic field is applied in a known orientation to realign the domains and reestablish the axis sensitivity of the AMR sensors.

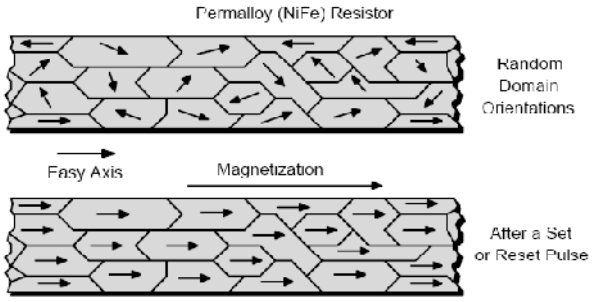


Figure 3.3: Random domain orientation versus domain orientation after a set or reset magnetic pulse. With permission from Honeywell International [6].

In the Honeywell HMC2003 three-axis magnetometer module (includes one HMC1001 and one HMC1002 sensor – see Section 3.1.3) implemented in this project, this is accomplished via a set/reset resistive strap that is oriented below the AMR sensors in their sensitive direction. Upon a high-voltage pulse, current flows through the resistive strap for several (typically less than 3) microseconds and the magnetic pulse resulting from the current realigns the magnetic domains.

The set/reset circuit used to create the magnetic/current pulse can be seen in the system schematic in Appendix A and in the HMC1001/1002 datasheet [2].

3.1.3 Implemented hardware

The magnetometers implemented in this project are the Honeywell International HMC1001 and HMC1002 AMR sensors. Both sensors are sensitive to ±2.0 Gauss in any sensitive direction, with the HMC1001 being a one-axis sensor in a single inline package (SIP), and the HMC1002 being a two-axis sensor in a 20-pin SOIC package. The HMC1002's two sensors are oriented perpendicular to one another, allowing bidirectional sensitivity. Mounted together, the HMC1001 and HMC1002 sensors create a three-axis magnetometer module called the HMC2003. Containing signal amplifying circuits and 1 kHz lowpass filters, the HMC2003, seen in Figure 3.4, is also sensitive to ±2.0 Gauss of magnetic field magnitude.

Figure 3.4: Honeywell International HMC2003 three-axis magnetoresistive module [3]. With permission from Honeywell International [3].

The HMC2003 module sensors output a nominal +2.5 volts for a null field reading, with field variations causing a ±1.0 Volt/Gauss change in output [3].

3.1.4 Testing and Results

To display the easy axis sensitivity of the Honeywell HMC2003 module implemented in this project, several experiments are performed.

To begin, a three-axis magnetic reading is recorded while the magnetometer hardware is rotated 360 degrees in space relative to the horizontal (X and Y axes) plane. The readings reflect the raw magnetic field detected by the sensor. It can be seen in Figure 3.5 that, as expected, the X and Y axes are 90 degrees offset from one another, as they are oriented orthogonal to one another. In this case, there is a magnetic component in the Z (vertical) direction with a larger magnitude than that reflected in the X and Y axes.

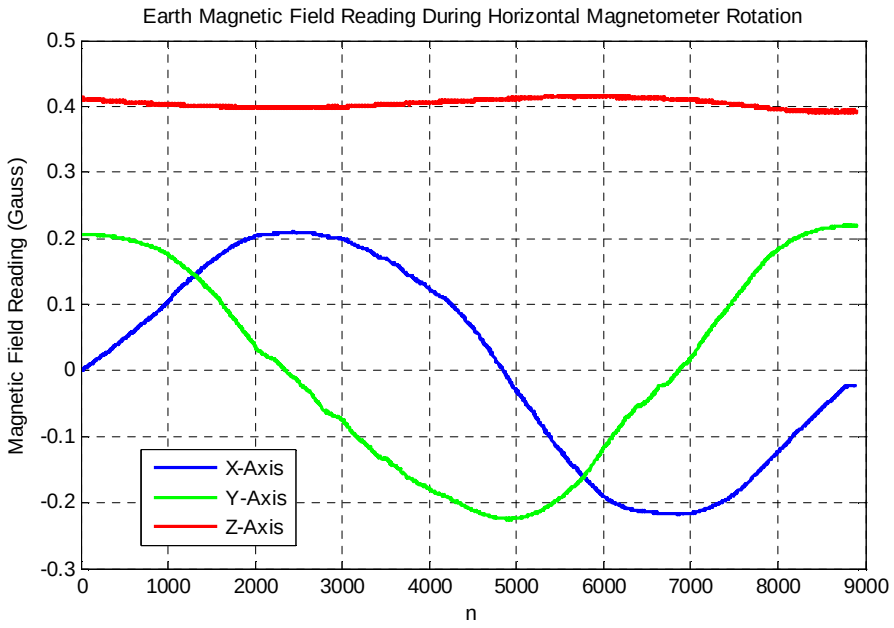


Figure 3.5: Magnetic field reading while magnetometer sensor is rotated in the horizontal X-Y plane.

The next experiment is very similar to the previous, with an alnico (AlNiCo) magnet rotated 360 degrees in the X-Y plane one meter above the magnetometer. Rotation of the dipole magnetic field, similar to rotating the magnetic sensor within the steady magnetic field in the previous experiment, can be seen in Figure 3.6. This test output reflects an initial offset calibration of the magnet's dipole aligned in the Y-axis direction, zeroing out all readings with the magnet present. Thus, when the magnet is parallel to its original position, a zero reading is observed.

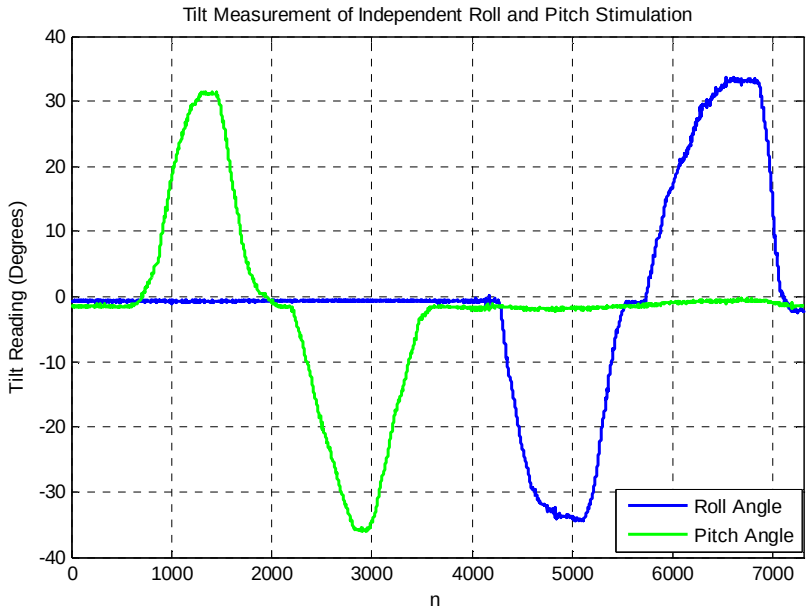


Figure 3.6: Magnetic field reading of X and Y axis sensors while alnico magnet is rotated one meter above magnetic sensors in the horizontal X-Y plane.

Finally, to further examine the sensitivity of the magnetic sensor's easy axis sensitivity, an initial data value of the surrounding magnetic field is taken and implemented as an offset, thus zeroing the readings of the magnetometer. An alnico bar magnet is then moved closer to the magnetometer from a significant distance to within 1 meter of the circuit board. The magnet is moved to this position from a great distance (greater than 20 meters in this experiment) to ensure the initial magnetometer readings are unaffected by the magnet's initial position. When moved closer to the sensor, the magnetic dipole is aligned with the X or Y axis, and then again removed to an undetectable distance. This process is repeated in 90 degree rotational intervals. It can be observed in Figure 3.7 that each axis is sensitive in its own respective direction. In addition, small nonzero perturbations in the magnetic readings of the cross-axes are present. This is a result of cross-axis error due to imperfections in magnet alignment relative to sensor axes, as well as very small imperfections in the magnetic sensors.

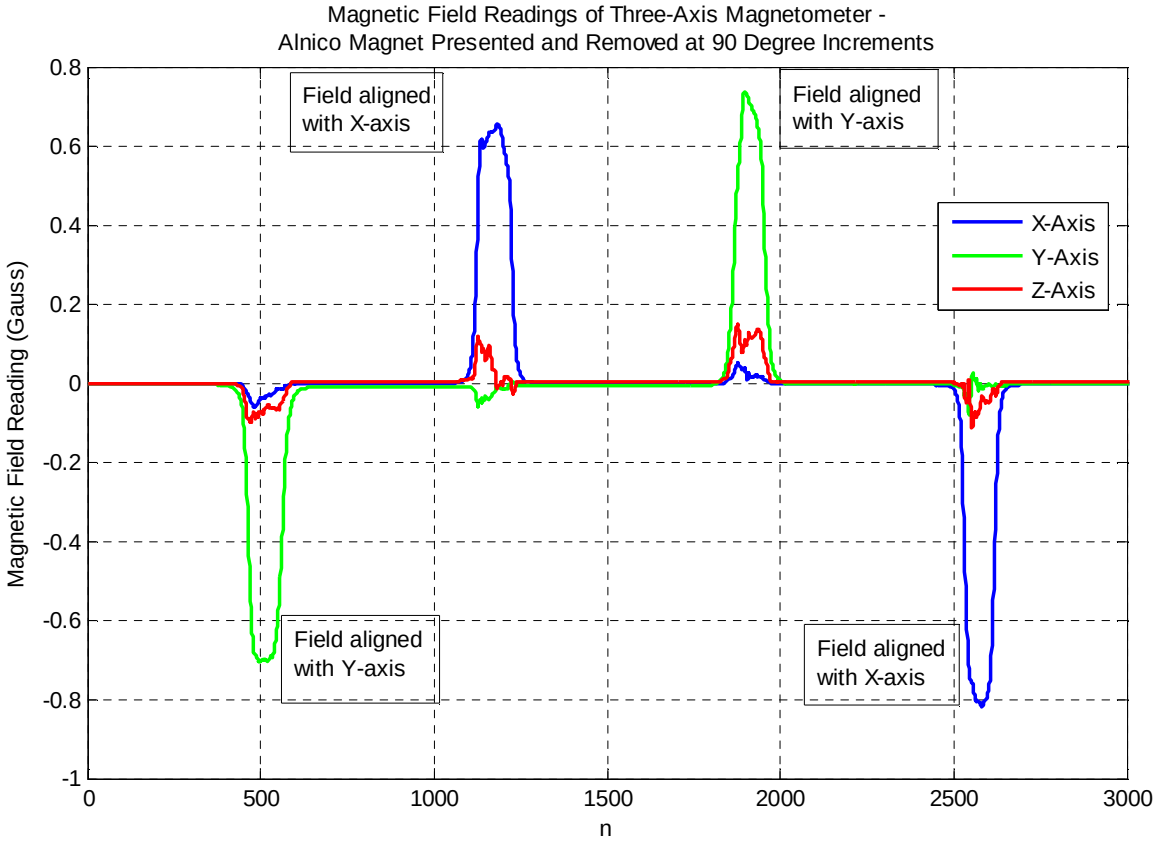


Figure 3.7: Alnico magnet presented and removed from magnetic sensor in 90 degree rotation intervals.

3.2 Inclination Sensors

Inclination can be measured by many varieties of sensors, including accelerometers, inclinometers, levels, or gradiometers. Any of these devices are used to detect the angle of inclination of a plane relative to another plane or force, usually the surface of the earth, or the pull of gravity.

3.2.1 Variations

There are many variations of devices that are capable of measuring inclination angle. These devices all present different levels of sensitivity. Used primarily in the field of geophysical studies, some variations of inclination sensors can detect perturbations in inclination down to 10^{-9} radians [40]. Other sensors, depending on materials and construction, present lower levels of inclination sensitivity, as well as cost.

Several different construction methods are incorporated into inclination sensors. The most common types of sensor transducers are spirit bubble, horizontal pendulum, vertical pendulum, or MEMS [40].

Spirit bubble type sensors implement a liquid with a gas bubble present within a container, much like a traditional carpentry level. With gravity working upon the liquid, the location of the bubble within the vessel is detected by any number of electrical or mechanical transducers to decode the inclination of the sensor itself relative to the vector of gravitational force [40].

Pendulum devices track the position of a gravity-driven pendulum within the sensor, and the inclination of the sensor is decoded by any means of the user [40]. MEMS devices fall under this category of sensor, differing only in size, as the MEMS are typically etched or printed on the micrometer scale within chips or embedded systems.

3.2.2 Implemented Hardware

To measure the inclination of the tilt-stabilized magnetometer system, low-cost inclinometer sensors from VTI Technologies are implemented in hardware. The SCA61T-FA1H1G inclinometers are sensitive from ±90º of inclination, with an output of 0 to +5 volts of inclusive full scale range, and are contained in a 8-SMD package [7].

The SCA61T series of inclinometers are MEMS-based sensors, with an overdamped sensing element for vibration attenuation. Calculation of the inclination angle of the sensor is most accurately accomplished via inverse sine calculation:

$$\mathrm{Angle_{OUT}} = \sin^{-1}\left(\frac{\mathrm{V_{OUT}} - \mathrm{V_{OFFSET}}}{2\ \mathrm{V/g}}\right) \tag{3.1}$$

where V_{OUT} is the current output voltage of the inclinometer, V_{OFFSET} is the initial reading, or zero calibration output, and 2V/g is the output sensitivity of the SCA61T sensor.

Two inclinometers are implemented in hardware for this project, with one sensor aligned in the x-axis direction, and the other aligned in the y axis direction, creating horizontal roll and tilt measurements relative to the vector of gravity. These sensors, when interfaced simultaneously provide complete two-axis data on the inclination of the circuit board. The convention used for

this project will entail pitch angle is measured as rotation about the Y-axis, and roll angle is measured about the X-axis.

3.2.3 Error

The inclinometers implemented in this project are subject to error in accuracy and precision due to several different electrical and physical mounting causes.

First, mounting of the inclinometer on the circuit board itself presents physical error in inclination angle measurement. To eliminate this error, the digital resolution of the inclination system must be known. For example, in this project 12-bit analog-to-digital converters are used to sample the inclination angle readings. At a full-scale range of ±90º, the resolution of the system is 0.044º. To ensure that the error due to mounting is minimized, the chip must be positioned within less than 0.044º of alignment with the easy axis of the magnetometer system. Due to surface-mount soldering and mounting precision error, this is attempted, but not necessarily accomplished in the construction of the circuit board (also discussed in Section 3.3 and Chapter 6).

In addition, slight sensitivity errors are present in the system. The tolerance of the inclinometers is ±1.5% at high degrees of inclination [41]. Cross-axis error, or the effect of a perpendicular axis tilt causing output in a non-tilted sensor, is less than 5% [41].

Finally, due to the overdamped transducer system onboard the SCA61T inclinometers, the bandwidth of vibration frequency attenuated by the system spans from 0 Hz to approximately 10 Hz [41]. In the case of extremely high-force perturbations (greater than 5g) from large acceleration conditions, the overdamped system may cause a perturbation from a zero point shift. This is discussed further in Section 3.2.4.

3.2.4 Testing and Results

To test and display the output of the inclinometer system for this project, two tests are performed.

First, a simple test of inclination angle is executed. Inclination of approximately 30º to 35 º in both the positive and negative direction is performed sequentially for both the X-axis and Y-axis inclinometers. It can be seen in Figure 3.8 that each sensor responds as expected, and a very minimal cross-axis error is seen due to possible mounting and electronic error, as mentioned in Section 3.2.3. The errors, displayed as deviation from zero inclination, are also the result of manual imperfection in initial and cross-axis alignment during the test. This error can be expected, as the tilt is created in the two tilt axes by hand.

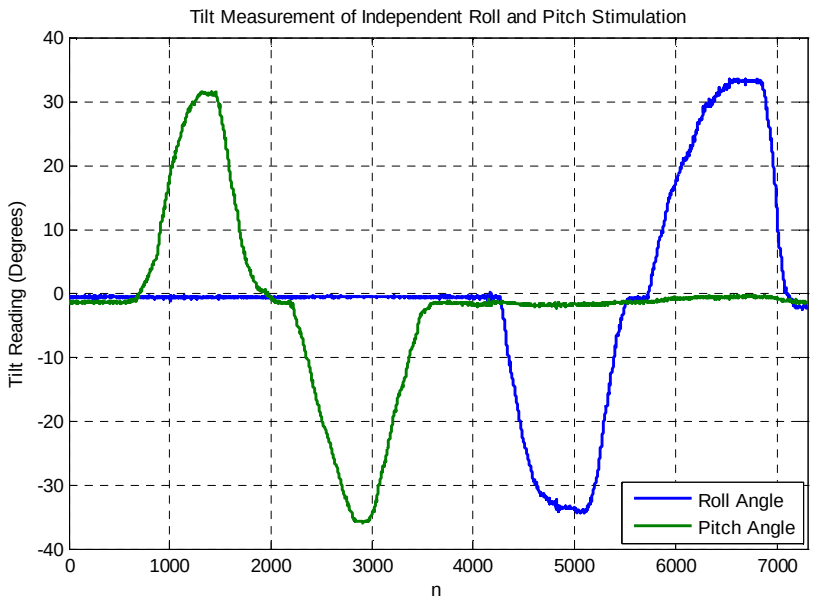


Figure 3.8: Inclination of magnetometer board to approximately ±30º.

To examine the performance of the inclinometers at high frequency, a large acceleration excitation is required. To accomplish this goal, a step-like input is presented to the magnetometer system by means of accelerating the rotation of the system using the force of gravity. This is accomplished via the step input platform designed for the magnetometer system (discussed in detail in Section 3.4). The step input is powered by a nonmagnetic weight that is released while attached to the platform. The weight inclines the platform at a rate of acceleration of 9.8 meters/second^2 at the edge of the platform where it is mechanically stopped at an angle of ±45º. This signal elicits a high-frequency response from the inclination system, and the effects of the overdamped MEMS system can be seen in Figure 3.9 below.

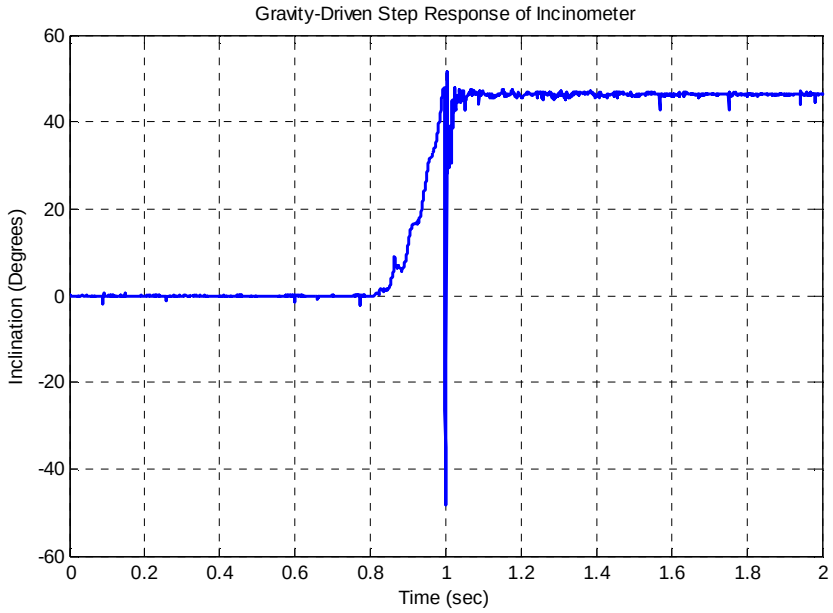


Figure 3.9: Sample gravitational inclination response of the SCA61T inclinometer at 9.8 m/s2 to inclination of 45º.

It is clear from the figure that the inclinometers produce an impulse-like undershoot. This is a result of a zero point shift and the overdamped system, or a change in the zero-reference point of the MEMS system as a result of the large amount of negative acceleration force [41]. This effect is reflected in step responses of all angle magnitudes and is seen consistently throughout all inclinometer devices tested (see Figure 4.1). Though not a true step response due to the controlled rotational acceleration of the platform, the acceleration of gravity is assumed to be the greatest acceleration presented to the magnetometer system. Thus, for the purpose of this report, the rotational signal produced from the gravity-driven acceleration (seen in Figure 3.9) will be referred to as the system's step response.

3.3 Circuit Design

Integration of all components of the circuit board, including the dsPIC microcontroller and all interface devices is accomplished using circuit design and printed circuit board (PCB) software. System specifications are primarily determined by component properties, with the focus being on instruction cycle, communication and execution speeds being maximized. The complete circuit layout can be seen in Appendix A.

3.3.1 Components and Specifications

Interfacing the magnetometer and inclinometer devices for this project is Microchip Technology's dsPIC™ microcontroller, dsPIC30F5011™. Operating at 16 bits and possessing 44 kilobytes of program memory, 4 kilobytes of data memory and 1 kilobyte of EEPROM, the dsPIC30F5011microcontroller is capable of a maximum of 30 million instructions per second (MIPS) [42]. On-chip are several communication modules, of which the universal asynchronous receiver transmitter (UART) and serial peripheral interface (SPI) are implemented. Sampling the analog inclinometer signals is the microcontroller's onboard 12-bit analog to digital converter (ADC). Chosen over communicating with the inclinometer's onboard SPI module, the 12-bit ADC allows for flexibility in sample speed and output format, in addition to requiring fewer instruction cycles, and thus execution time. Clocked by a 6.144 MHz crystal oscillator with phase locked loop (PLL) at 16x setting, the dsPIC microcontroller operates at 24.576 MIPS. System timing and sampling is controlled by one of the microcontroller's five onboard timer modules. The system will be programmed and debugged via Microchip's in-circuit debugger (ICD2).

Communicating through the SPI module at a clock frequency of 6.144 MHz, the magnetometer signal is sampled by Analog Devices' AD7734™ ADC. Set to 16-bit resolution, the ADC is capable of up to 15 kHz sampling rate, depending on several settings, including filter word length and chopping (see [43]), both of which affect the number of noise-free bits. This tradeoff in bit resolution versus sample rate provides flexibility in design and testing of the magnetometer system, and thus is chosen over utilizing the microcontroller's onboard ADC. These tradeoffs and specifications are covered in detail by the datasheet from Analog Devices [43]. For the purpose of this project, filter word is set to a value of 8 with chopping enabled. This results in a sampling time of 207 μsec, 4826 Hz output data rate, and a -3dB filter word cutoff frequency of 2500 Hz.

At higher sample rates, including the output data rate implemented, the effective resolution of the ADC is less than 16 bits, and thus approaches the resolution of the 12-bit readings of the inclinometers.

Powered by a 9V external source, a single 5V linear regulator powers the digital components of the system, while the 9V source is decoupled and powers the analog electronics onboard the HMC2003 module.

Communication to the PC is accomplished via a general purpose RS232 driver interfacing the UART module, which is set to a baud rate of 115,200 bits per second.

Accomplishing the task of creating a set and reset pulse for magnetic domain alignment is a simple switching transistor circuit, which can be seen in Appendix A, the HMC1001/1002 device datasheet, and application notes [6]. This circuit is controlled by one of the microcontroller's general purpose input/output (I/O) ports.

3.3.2 Layout

A printed circuit board layout is designed and constructed via surface mount reflow soldering techniques. The HMC2003 magnetometer is positioned at the center of the circuit board (within 0.02 mm), with the center of the HMC1002 chip being considered the center rotational point. The SCA61T inclinometers are aligned with the center axes of the HMC2003 module for maximally accurate inclination angles.

For external communications, a DB-9 port is present, with all communication and power signals interfaced to the pins.

The set/reset pulse circuit is off of center, with the shortest path to the resistive strap considered in its positioning. At the four corners of the PCB are 0.27" holes for the purpose of creating an anchor point for securing the circuit board to the tilt platform. Software layout and a photograph of the PCB can be seen below in Figures 3.10 and 3.11, respectively.

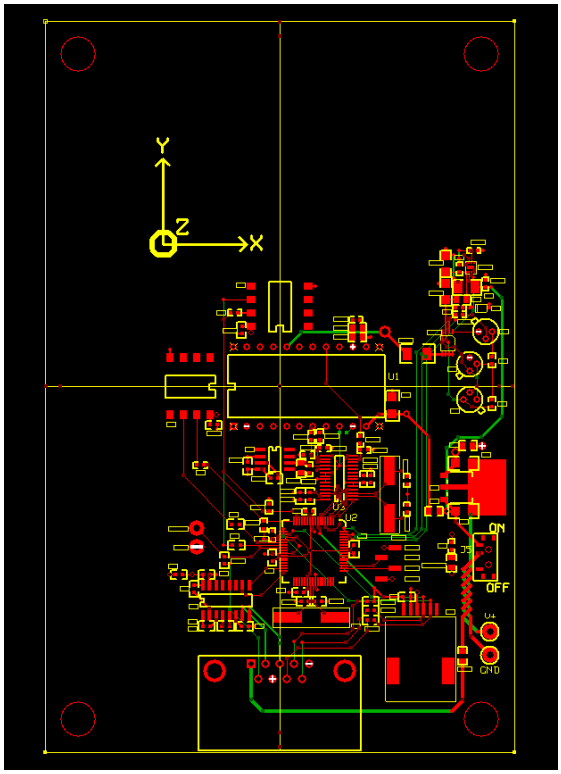


Figure 3.10: PCB layout of magnetometer/inclinometer circuit board.

Figure 3.11: Magnetometer/inclinometer circuit board.

3.4 Test Platform

Repeatable and consistent testing of the magnetometer system is required to analyze the system performance. To accomplish this goal, a tilt platform has been constructed with roll and pitch tilt capabilities.

3.4.1 Setup

Construction of the tilt platform is accomplished via a concentric frame design, with the magnetometer system circuit board mounted in the center platform.

With two axles protruding from the walls of this platform in the Y direction, pitch rotation is achieved. To allow for roll, these axles are inserted and free to move about a rectangular frame of which two more axes protrude in the X axis direction, and thus roll rotation is achieved.

The magnetometer circuit board is stabilized within the inner platform via four 0.25” thick stabilizer blocks with 0.25” wooden dowel rods protruding. These dowels fit tightly into the designed through holes in the circuit board, and with precise fitting, lateral motion of the circuit board relative to the platform itself is eliminated. The axles are positioned so that all rotation is centered exactly at the center of the HMC1002 chip onboard the magnetometer module.

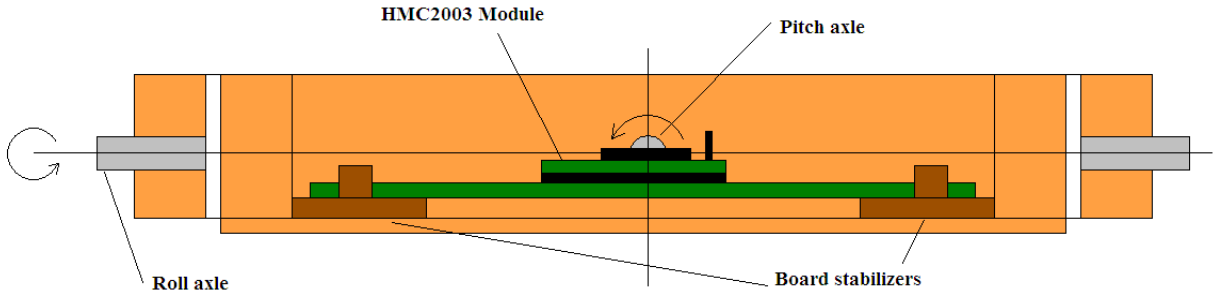


Figure 3.12: Cross-sectional view of tilt platform. Platform axles positioned for center of rotation at HMC1002 sensor.

3.4.2 Construction

Construction of the tilt platform must be void of any magnetic material to minimize interference, as discussed in Section 2.4. The platform itself is constructed from 0.5” CDX plywood, 0.25” mahogany plywood, and 0.375” wooden dowels as axles. Glued together, the system is void of any screws, nails or bolts. The circuit board is fastened in place with four tight rubber straps.

Nylon washers stabilize both the platform and the frame from lateral motion. Protractors are centered on both axes of rotation to estimate tilt, for use only in estimation of tilt, as these measurements are not accurate.

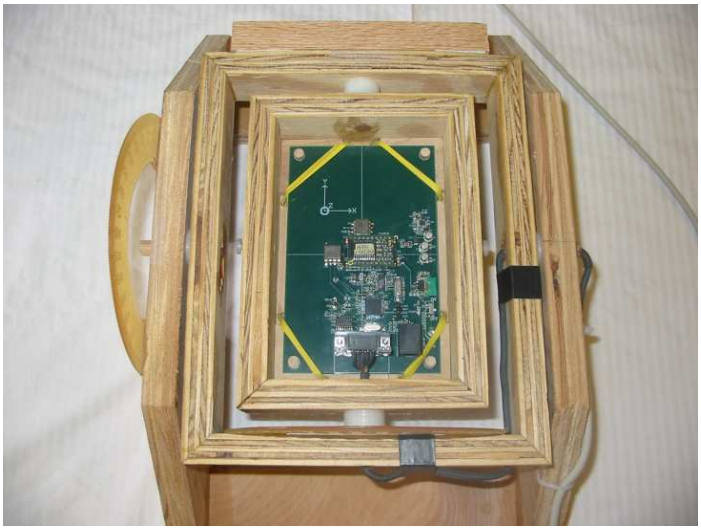


Figure 3.13: Top view of magnetometer/inclination system on tilt platform.

To eliminate magnetic self interference as a result of electrical current powering the board, wire motion is eliminated via the RS232 cable tracking through the axes of rotation. With 9 wires connected to the DB-9 connector onboard, the wires are run first through the inside of the 0.375" pitch angle dowel, fastened to the platform, and then run through the inside of the roll angle dowel. Eliminating tilt motion of the system wires, only a wire rotation is seen within the axles. Thus interference in the surrounding magnetic field from current flow is minimized to only the rotation of the wire bundle about itself, within a radius of less than 4 millimeters maximum.

Figure 3.14: System cable routed through pitch and roll axle.

Figure 3.15: Simultaneous roll and pitch tilt.

To test the response of the magnetometer inclination detection system at high frequencies (step input discussed in Chapter 3.2.4), a repeatable step stimulus platform is constructed. Conforming to the tilt platform, the step response platform consists of a frame stabilizing the tilt platform and a cable guide extension, at the end of which is a small nylon pulley wheel. Centered in the platform is a nylon cable guide.

To perform a step response at the acceleration of gravity (9.8 meters/second^2), a non-magnetic weight consisting of a plastic container filled with sand is attached to a nylon line, which is threaded over the pulley wheel, through the wire guide, and attached to one side of the internal tilt platform. After locking the outside platform in place (eliminating roll tilt capabilities), the weight is dropped from a stable position. With wooden stoppers halting the motion at ±45º of pitch rotation, a step input is created, and can be analyzed via the inclinometer system.

Figure 3.16: Step response platform with roll platform locked.

Figure 3.17: Pulley/wire guide combination (left) and wire guide without tilt platform (right).

Figure 3.18: Tilt platform after 45º step response.

Through the construction of the tilt platform, a non-magnetic setup is available for repeatable, stable readings of simultaneous roll and pitch tilt readings. When locking the roll platform, the system is capable of pitch-only readings, which includes the possibility of a repeatable step response of the system.

Chapter 4 Signal Filtering

With magnetic field and inclination signals being simultaneously obtained by the magnetometer system, integrating both signals within an accurate compensation system requires both signals to possess a high signal-to-noise ratio. With different filter specifications and requirements from the unique qualities of both the magnetic and inclination signals, both must be compatible in noise attenuation, phase delay, and execution time for the overall magnetometer system performance to be optimized.

In this chapter, median filters are designed for the purpose of attenuating high-frequency impulse-like noise in the inclination signal. In addition, finite impulse response digital filters are designed and tested to filter potentially unwanted noise in the magnetic signals. Various design specifications of each filter are discussed and presented with simulation and hardware testing.

4.1 Inclination Signal Filtering

As discussed in Section 3.2.4, as a high-frequency input is applied to the inclinometers of the magnetometer system, an inverted impulse response is output. This effect is due to the overdamped MEMS system onboard the sensors, causing a zero-point shift. This shift is a change in the point of zero-reference of the sensor due to the high acceleration impulse, thus skewing the output of the system. The effect is consistent and repeatable with multiple sensors and tests, as is seen in Figure 4.1 below, which shows a 0.8 second period of a full 2-second test. The impulse-like spike is present in the system response signal for approximately 10 milliseconds.

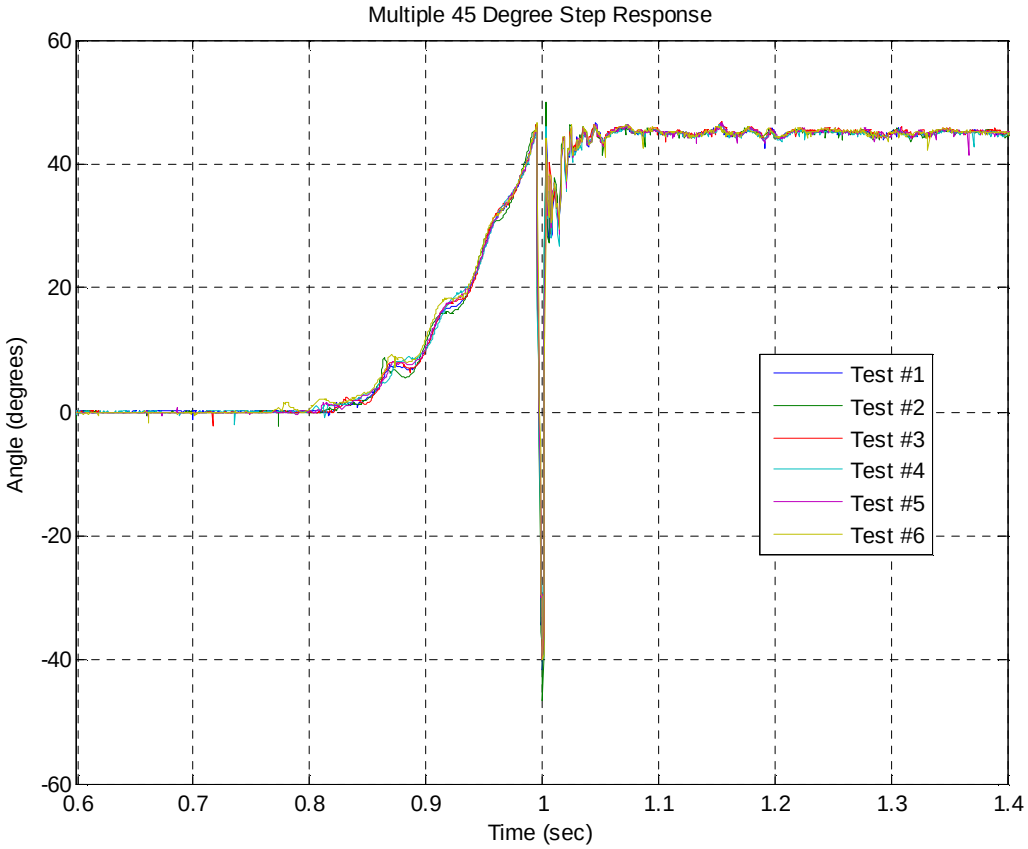


Figure 4.1: Six tests of 45° step response.

From several decades of research and development, median filters have become one of the most versatile and effective filter types for eliminating outliers [14-17, 19]. Median filters are rank-order based non-linear filters that output the median of the sorted data set, as seen in Equation 4.1. N represents the number of data points in the filter window.

$$y_n = \text{Median}(x_0, x_1, \ldots, x_{N-1}) \tag{4.1}$$

Significant advances in research have led to numerous variations of the median filter, all of which alter either the sample space considered, filter architecture, execution, or weights. Several of these filters are explored and considered for implementation in this project, for the purpose of smoothing the inclination signal. To test the individual median filters, a sampling frequency of 500 Hz is implemented. To effectively filter the impulse aspect of the high-frequency response of the inclinometers, a minimum filter length of approximately 10 times the number of samples containing the impulse portion of the high-frequency signal will be considered acceptable. At a sampling frequency of 500 Hz, this is equivalent to a 25^{th} order median filter. To display the magnitude and phase delay of the median filters, higher order filters will be simulated. With

higher order, more effective median filtering is achieved and the phase delay is clearly displayed. Window width of all median filter windows is denoted by N.

4.1.1 Alpha-Trimmed Median Filter

Alpha-trimmed (α-trimmed) filters have been the topic of significant research, with primary focus on alpha-trimmed mean calculation. For the purpose of this project, an α-trimmed median filter is simulated and tested to reveal potential for tunable outlier rejection [14,15].

The α-trimmed median filter is simply a median filter as described above, with portions of the input data set removed. The number of data points eliminated in this manner is determined by the coefficient α, where α·N samples are removed from each end of the data set, where N is the total number of points considered in the median window. Thus, after removing α·N newest and oldest points of the data set, the median of the remaining data points is output. Median filtering research has shown that filters considering a wider (larger N) data window, the smoothing and outlier rejection aspects of filters are emphasized [15,16]. Thus, as the window width decreases, the detail and corners of the input signal are preserved. α-trimming samples from the median window allows for a similar phase delay as a filter with the full window width, yet allows for more signal detail preservation in the output. This tradeoff while reflecting similar phase delay characteristics is explored in this section. It can also be noted that the α-trimmed phase and detail preservation characteristics of the filter are undesired by the user, the value of α can be set to zero, and the filter performs like a true median filter. Thus, when α is maximized, the filter acts as a time-delayed identity filter.

$$\alpha = 0.25 \qquad y = \text{median}(x_4, x_5, \ldots x_{10}, x_{11}) \tag{4.2}$$
$$N = 16$$
$$\alpha \cdot N = 4$$

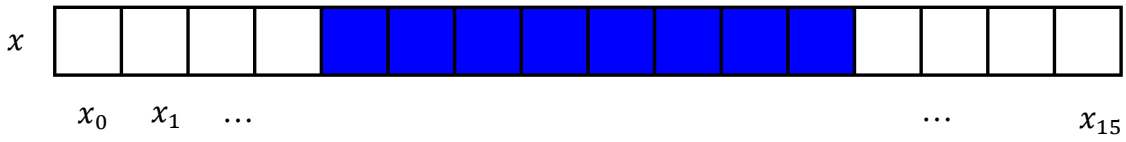


Figure 4.2: α-trimmed median filter with N = 16 and α = 0.25.

4.1.1.1 Simulation

To simulate the operation of the α–trimmed median filter, several variations are tested. Two degrees of freedom are exercised in this experiment, including the width of the median window (N) and magnitude of the filter's α coefficient. The MATLAB code, found in Appendix B, calculates the median of the α-trimmed window as described above. α-trimmed median filters will be denoted by 'ATM(N, α)', where N is the filter window width.

To begin, filters of different window lengths N are considered, all with the trim value remaining constant at $\alpha = 0.10$.

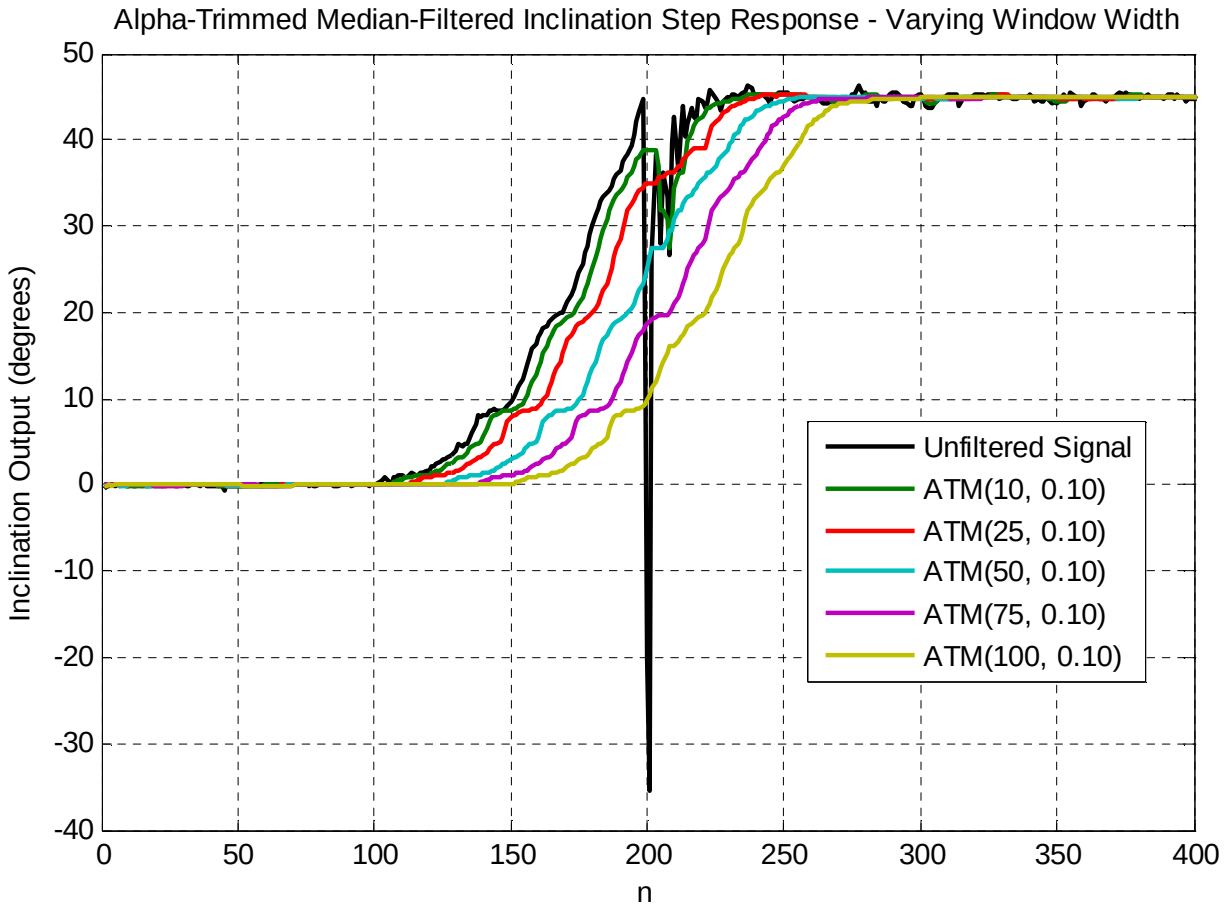


Figure 4.3: Alpha-trimmed median filter output with varying window width.

From Figure 4.3, it is clear that the α-trimmed median filter is effective at removing outliers, including the impulse-like signal that is present in the inclinometer step response. As the width of the median window is increased, the sharper artifacts of the signal are removed, as the number of data points considered for the median output is also increased. It is also shown that as window width increases, an increase in phase delay is created. Due to the non-linearity of the median filter, the exact phase delay is dependent only on the characteristics of the input signal contained within the median window, including frequency, corners, and rate of change. However with an ideal step response, the phase delay of a standard median filter is equivalent to half of the number of samples in the data window, or $N/2$ samples.

To explore the effect of the trimmed window width, a sample window of a fixed number of samples (N = 50) is considered, and the value of α is varied, altering the detail preservation characteristics of the filter.

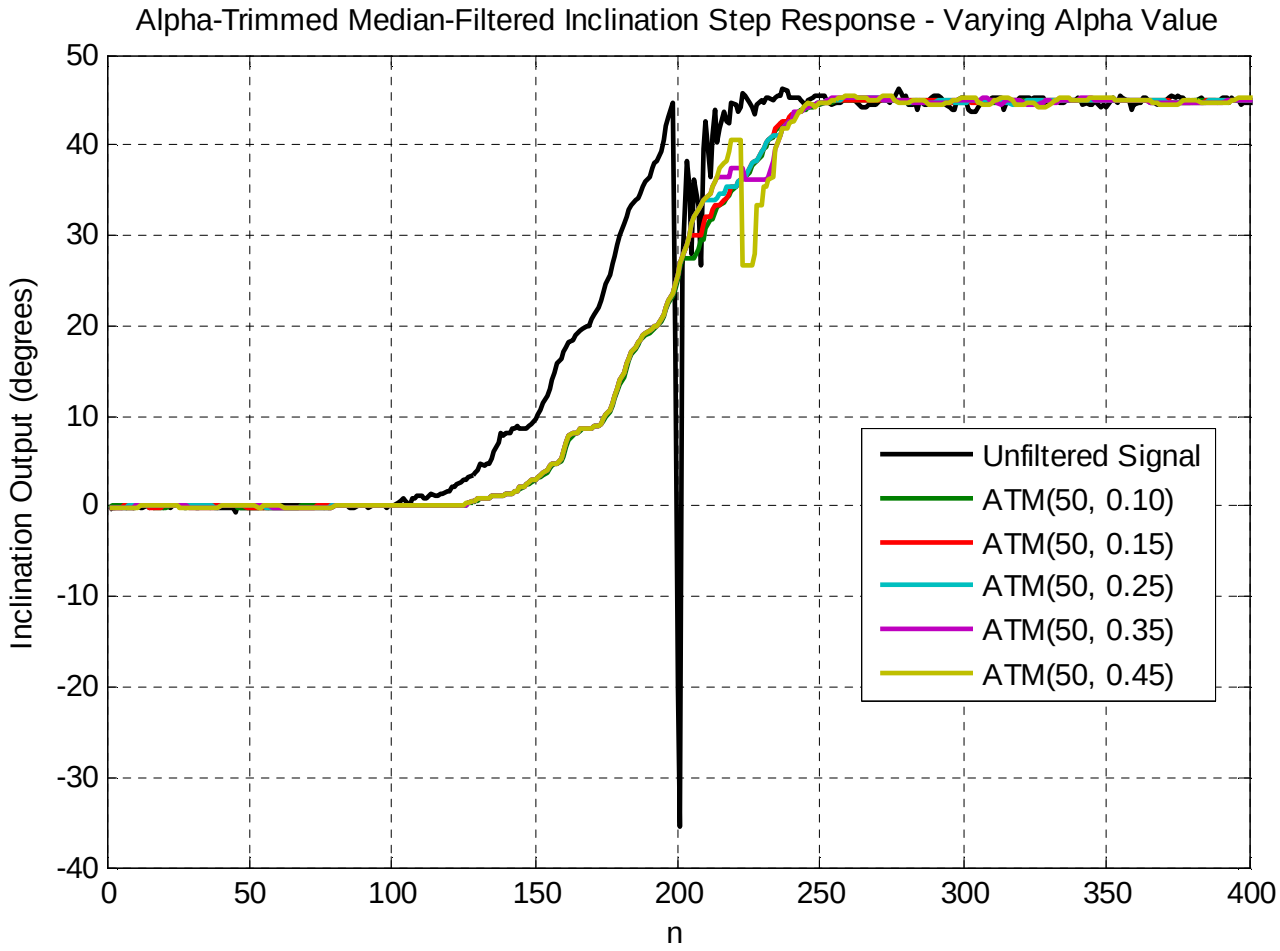


Figure 4.4: Alpha-trimmed median filter output with varying alpha and window width N = 50.

Figure 4.4 shows that the value of alpha for trimming the median window width has minimal effect on the smoothing of lower frequency aspects of the input signal. At the high-frequency impulse-like portion of the input signal, however, the alpha-value alters the smoothing performance of the filter. Figure 4.5 below shows the filter outputs in this region of the signal.

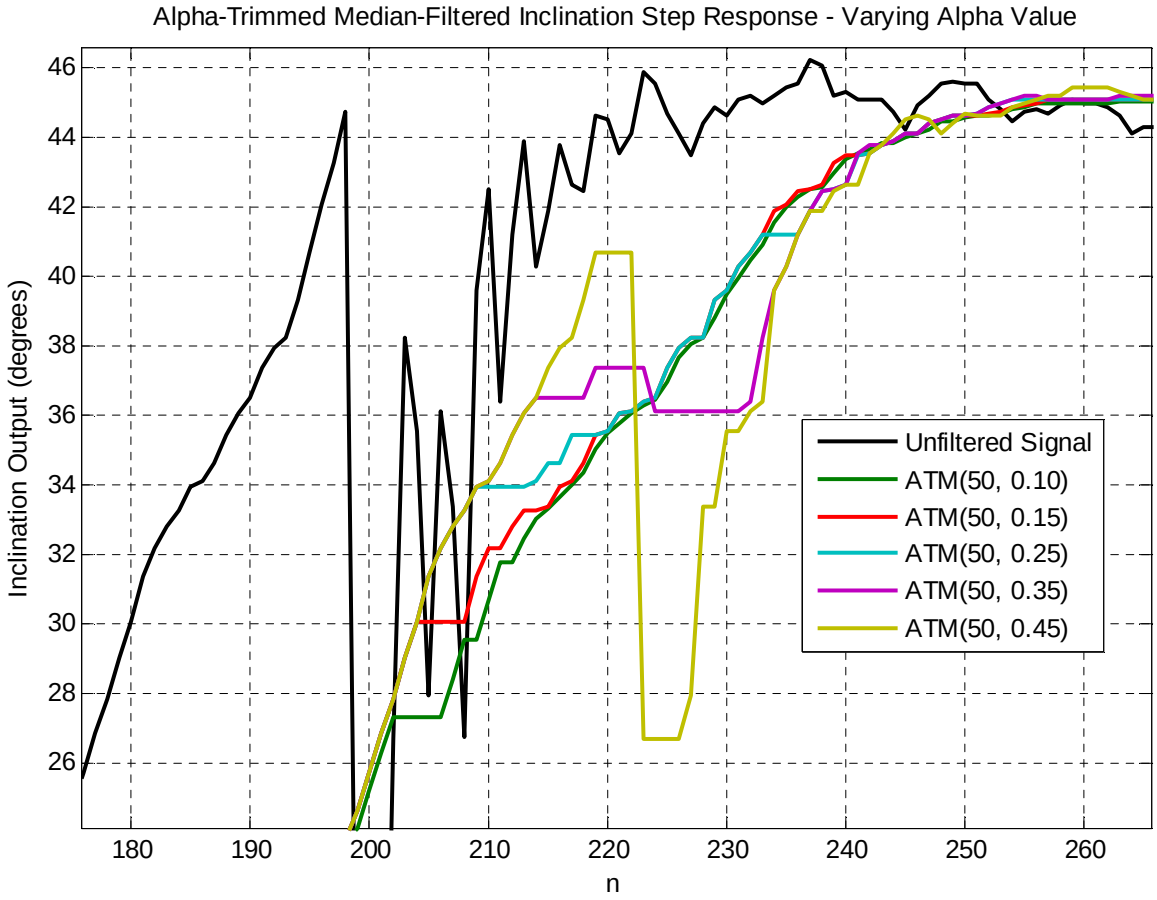


Figure 4.5: Zoomed view of Figure 4.4.

From Figure 4.5 it is clear that as the value of alpha increases, the filter approaches performing as an identity filter. This is seen in the samples of Figure 4.5 where alpha = 0.35 and alpha = 0.45, as the impulse aspect of the signal begins to reappear. As alpha value of the filter is decreased, the filter approaches the behavior of a true median filter, considering the entire input window as the α-trimmed data set. Nearly identical phase delay is presented, however, with varying values of α.

4.1.1.2 Implementation

Testing of the α-trimmed median filter in system hardware is most efficiently implemented via functions contained in the dsPIC DSP library. From the Microchip DSP library code, the function `VectorCopy` copies an array into another array with minimal cost in instruction cycles and `VectorMax` calculates the maximum value of an array. This microchip code can be viewed and downloaded from Microchip Technologies [45]. To find the median of an array, these two built-in functions are implemented to create the function `_Vector_Median`, which sequentially transfers the maximum array value into another vector. This vector is automatically sorted and returns the median. This function is called after obtaining every new sample from the inclinometer sensors. To incorporate the alpha-trimming aspects of the filter, the trimmed samples are excluded

from a vector buffer that is passed into `_Vector_Median`, eliminating them from the median sorting and calculation. Shifting of the samples in time within the filter window is achieved using function `VectorCopy` [45] which copies the window vector into itself with a one sample offset.

A filtered step response, as well as filtered low-frequency signals, all from the magnetometer system and dsPIC programming, can be seen in Figures 4.6 through 4.8 below. The filter window has width N = 75 and $\alpha = 0.20$ in all cases (ATM(75,0.20)).

From the figures, it is evident that the non-linear α-trimmed filters are effective at attenuating impulsive outliers, as well as removing small noise perturbations in low frequency signals. The phase delay is consistent through all frequencies of motion, and the transients from high acceleration conditions seen in Figure 4.8 are attenuated and smoothed.

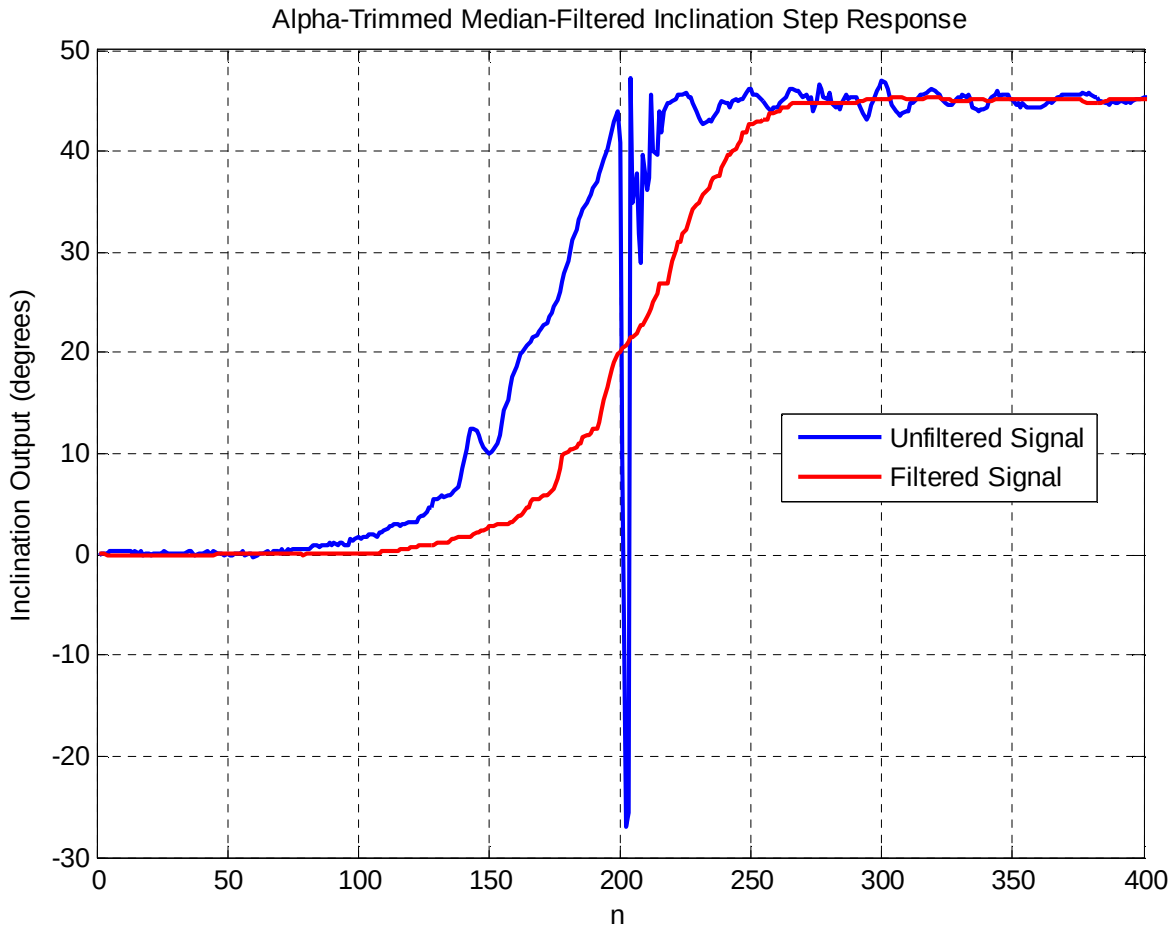


Figure 4.6: Filtered step response output from magnetometer/inclinometer system with ATM(75, 0.20) sampled at 500 Hz.

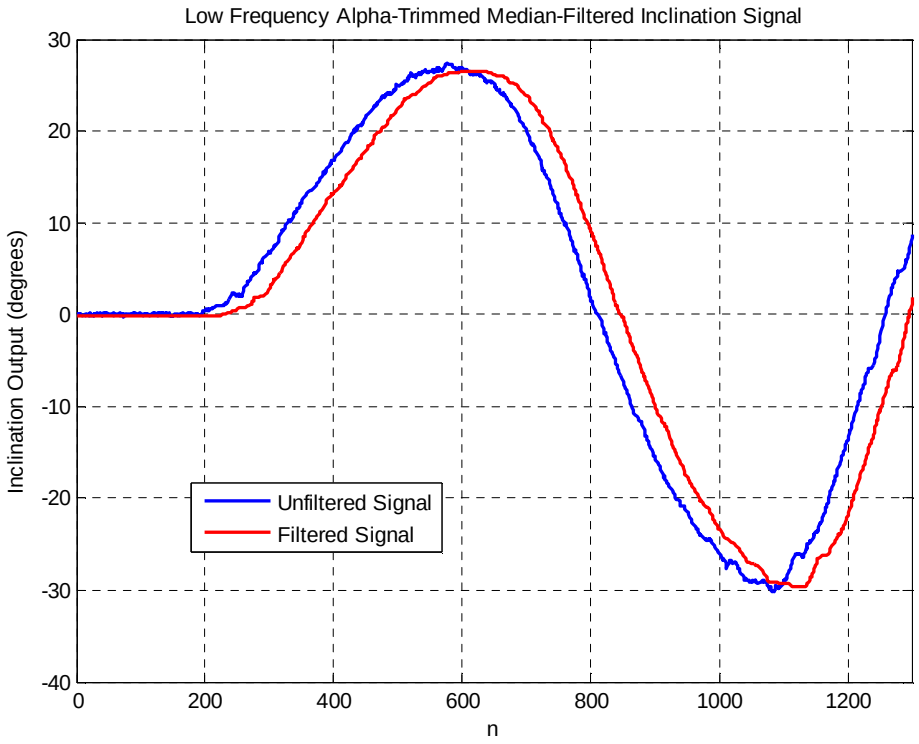


Figure 4.7: Filtered motion output from magnetometer/inclinometer system with ATM(75, 0.20) sampled at 500 Hz.

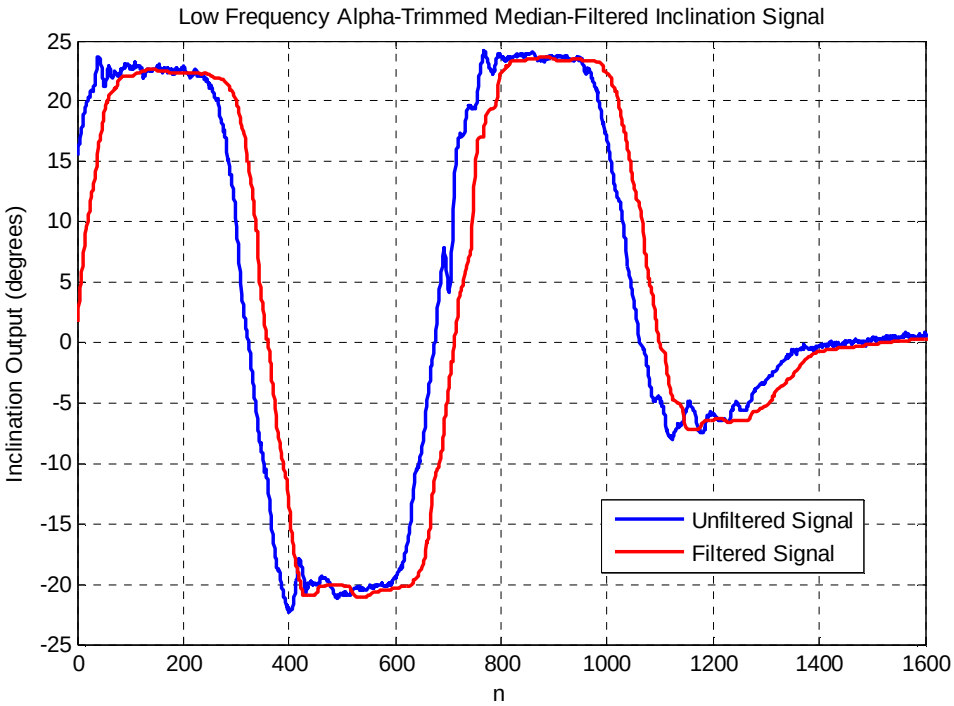


Figure 4.8: Filtered motion output with abrupt deceleration from magnetometer/inclinometer system with ATM(75, 0.20) sampled at 500 Hz.

4.1.2 Lower-Upper-Middle Median Filter

Lower-upper-middle (LUM) median filters, developed by Hardie and Broncelet, have proven versatile in their ability to be tuned to the specification of signal smoothing or sharpening, as necessary [21].

LUM filters are based on the theory that as the samples closer to the center of the median window become more heavily weighted compared to the ends of the window, the filter acts as a smoother [21]. Data points considered further towards the ends of the data result in a sharpening filter, preserving the edges of the input signal [21]. A LUM filter considers several different samples from the data set, including two or more samples spread evenly from the center of the data set, and the center sample itself. The spread of the two data points from the end of the data set is represented by the variable *l* or *k*, depending on the type of LUM filter.

4.1.2.1 LUM Sharpener

LUM sharpeners consider three data points; the center point and two data points offset from the ends of the data set by a spread factor *l*. The output of the filter is based on where the average of the two outside points are in relation to these two points and the center point, x^* [21].

$$t_l = \frac{(x_l + x_{N-l-1})}{2}$$

$$y_n = \begin{cases} x_l, & x_l < x^* < t_l \\ x_{N-l-1}, & t_l < x^* < x_{N-l-1} \\ x^*, & \text{otherwise} \end{cases} \tag{4.3}$$

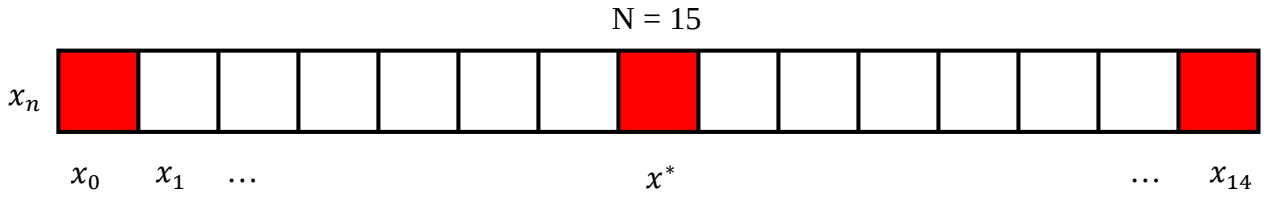


Figure 4.9: LUM sharpener with N = 15 and *l* = 0.

Simulation and implementation of the LUM sharpener are not required for the purpose of this project, due to the necessity of removing the sharp points from the inclination signal. However, inclusion of the sharpening aspects of the filter are incorporated in the LUM hybrid, which incorporates smoothing of the inclination signal while maintaining signal corners. This hybrid is examined in Section 4.1.2.3.

4.1.2.2 LUM Smoother

LUM smoother filters consider two data points at a known distance from the center of the data set, as well as the center point of the data set. Output of the filter is the median of these three points. The smoothing characteristic of the filter is a result of the data points considered in the median calculation to be the end points of a pseudo window that is spread about the center of the data set [21]

$$y_n = \text{Median}\left(X_{\frac{N-1}{2}-k}, X^*, X_{\frac{N-1}{2}+k}\right) \tag{4.4}$$

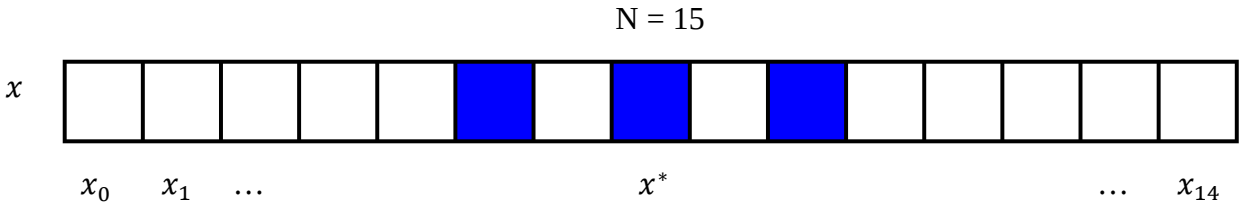


Figure 4.10: LUM smoother with N=15 and *k* = 1.

Simulation of the LUM smoother can be seen below in Figure 4.11 with varying values for median data point spread *k*. All LUM smoothing filters are denoted by 'LUMS(N, *k*)'. In this simulation, all data windows are 75 samples wide.

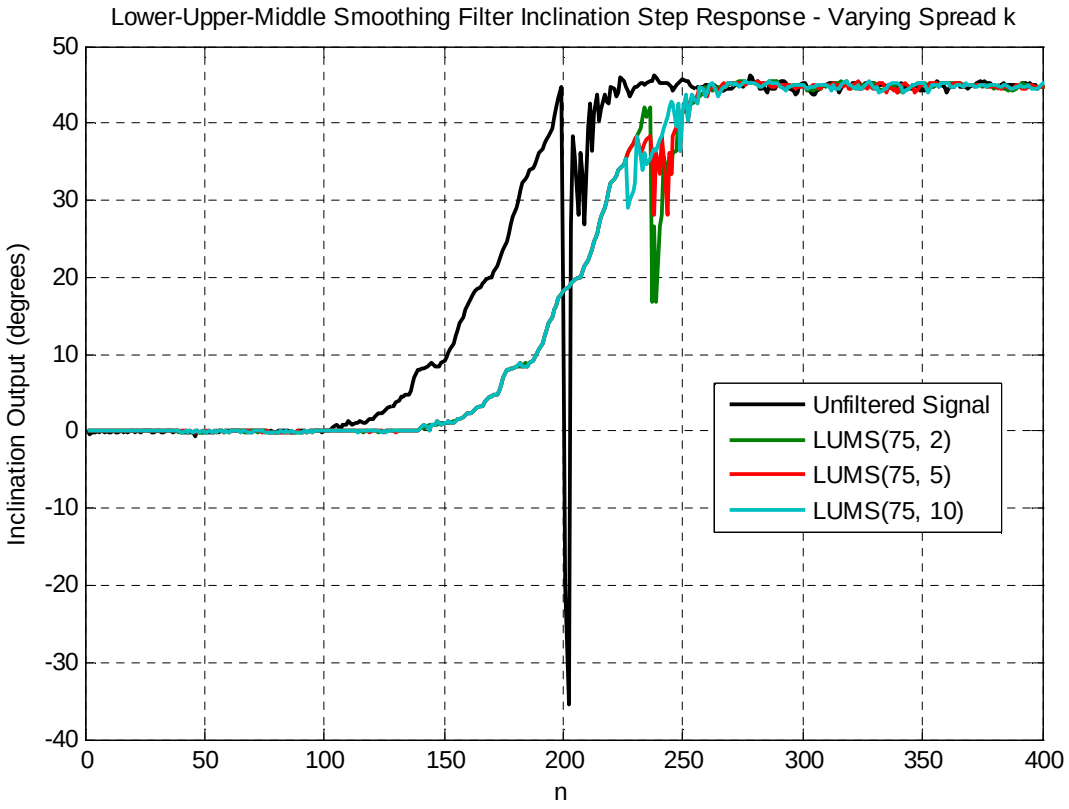


Figure 4.11: LUM median filter with varying spread values of *k*.

From the figure, the results show that as the spread decreases (lower *k* value), the smoothing effect of the filter diminishes as the filter tends toward simply turning into an N/2 phase-delayed identity filter. As the spread of the median points increases, the smoothing effect of the filter takes over, and the impulsive portion of the input signal is attenuated. The effect is not linear, however, in that as the spread increases further, the filter begins to act like a sharpener, as the points considered in the output are spread further to the ends of the data set. A zoomed view of Figure 4.11 is seen below in Figure 4.12.

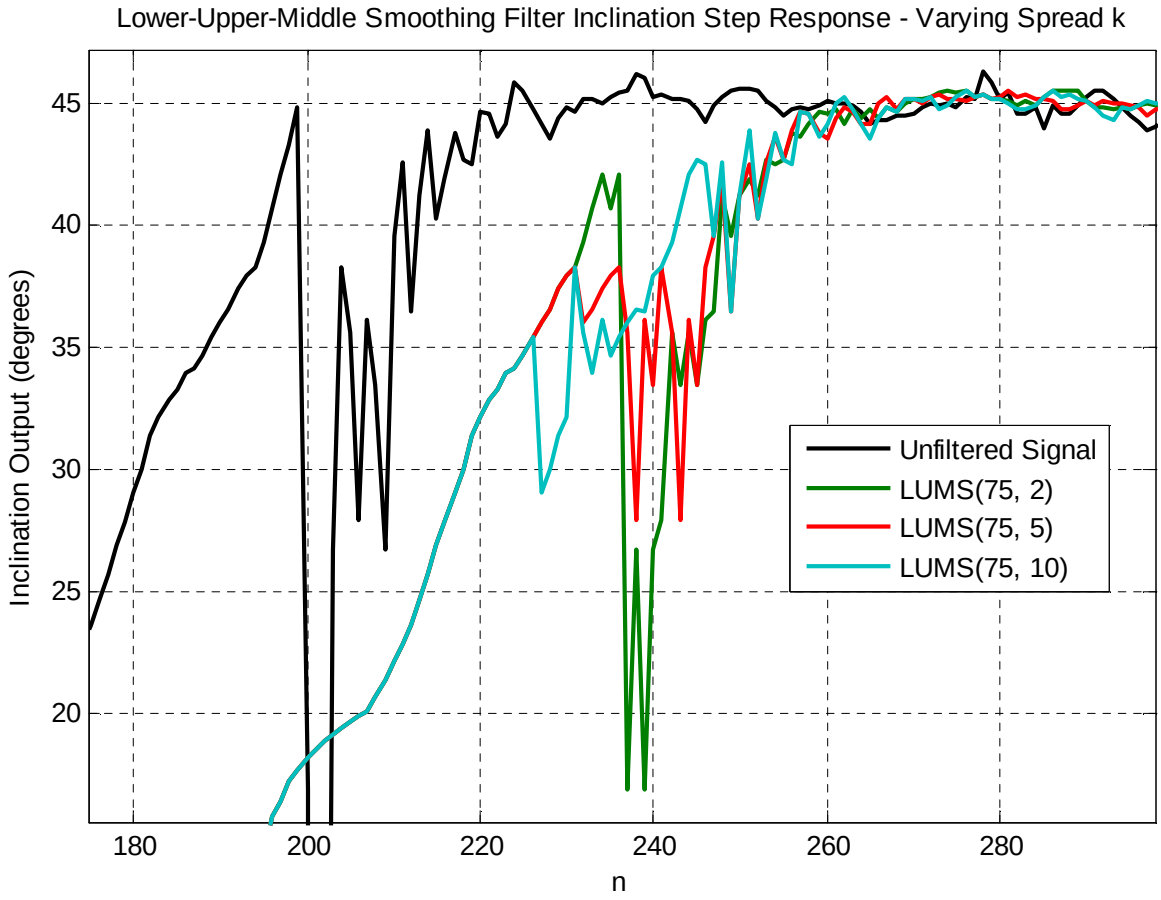


Figure 4.12: Zoomed view of Figure 4.11.

By examining the output of the filter, it is clear that the LUM smoother is capable of significant attenuation of the impulsive portion of the input signal. However, significant high-frequency artifacts remain through the implementation of this filter, due to the small number of points considered, as well as the spread of those points about the center of the data set.

4.1.2.3 LUM Hybrid

The LUM hybrid filter incorporates the aspects of smoothing and sharpening from both of the previously discussed filters [21]. By changing the ratio of the spread, more emphasis on either smoothing or corner preservation will be made. With smoothing of the input signal of highest priority, corner preservation is explored as a tradeoff for output signal filtering.

The LUM hybrid filter considers two medians, x_l and x_u, each calculated using three data points. Two of these points are on one side of the median, and the other is the center point itself. The spread of these points is determined by *k* and *l* [21]. The output of the filter is whichever point, x_l or x_u, is closer to the middle data point x^*.

$$x_{low} = \text{median}(x_l, x^*, x_{\frac{N-1}{2}-k})$$

$$x_{up} = \text{median}(x_{N-l-1}, x^*, x_{\frac{N-1}{2}+k}) \tag{4.5}$$

$$y_n = \begin{cases} x_{low}\,, & x^* \le \dfrac{(x_{low} + x_{up})}{2} \\ x_{up}\,, & \text{otherwise} \end{cases}$$

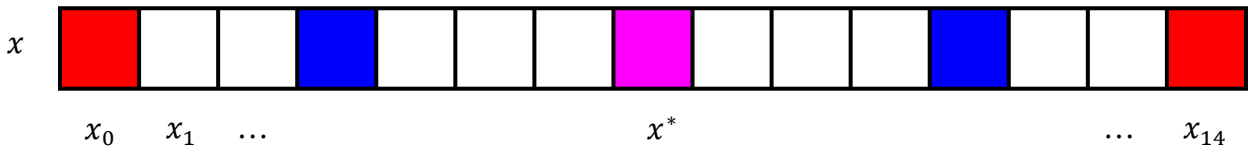


Figure 4.13: LUM hybrid filter structure.

To test the smoothing and sharpening aspects of the hybrid filter, two experiments are performed. First, emphasis on the sharpening aspects of the hybrid filter are explored with the value of *l* fixed and in a position towards the outer edge of the data set (*l* = 5), and the value of *k* is varied. The data window for all experiments is set at 75 samples in width. All LUM hybrid filters are denoted by 'LUMH(N, *k*, *l*)'. The results are seen in Figures 4.14 and 4.15 below.

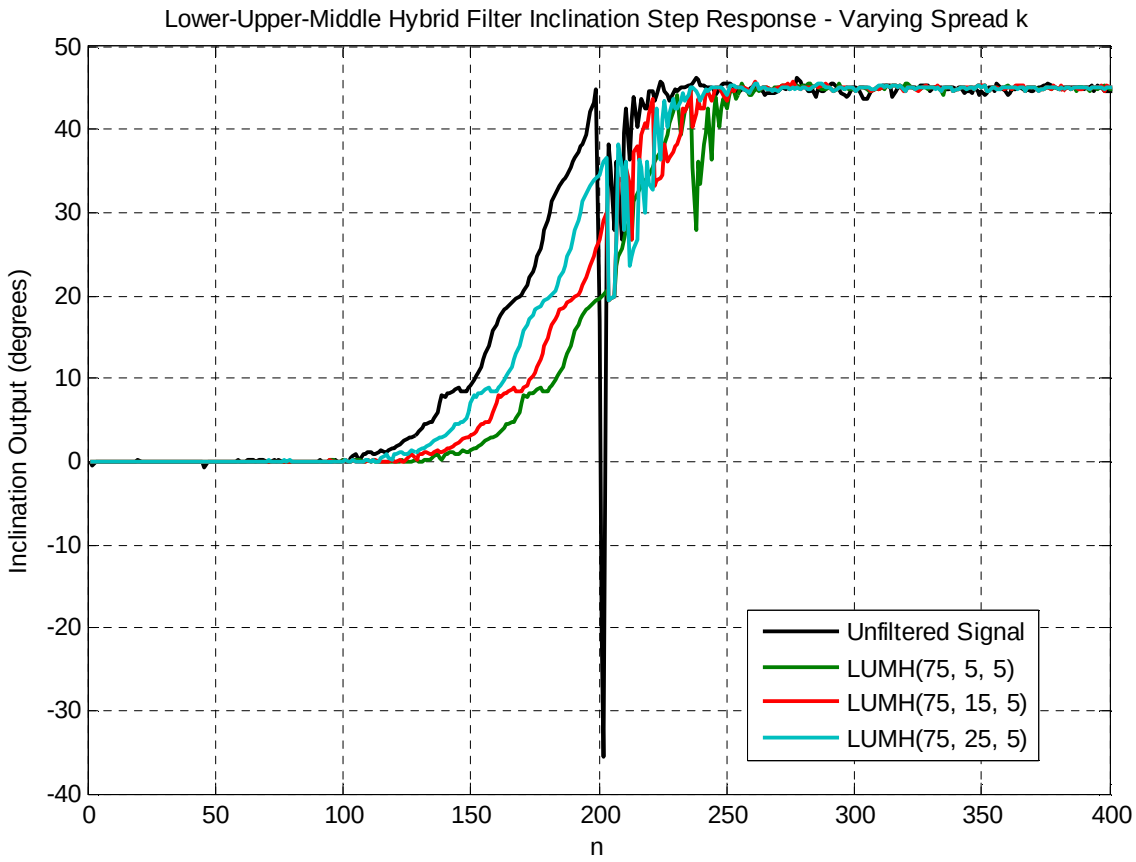


Figure 4.14: LUM hybrid filter with fixed *l* = 5 and varying *k*.

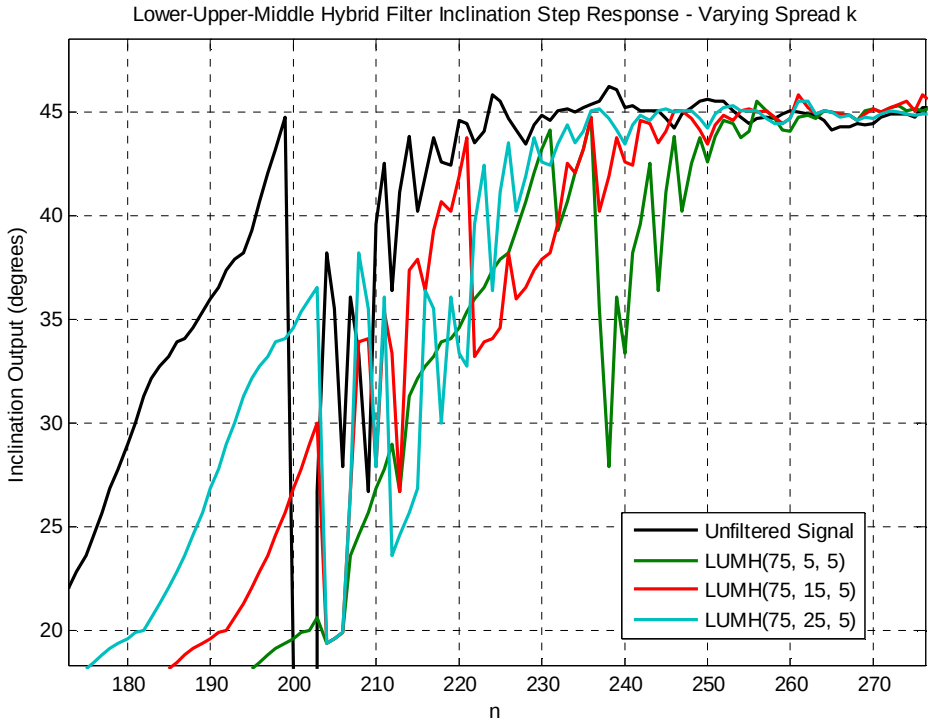


Figure 4.15: Zoomed view of Figure 4.14.

From the figures, it is clear that even as k is at lower values, the sharpening aspect of the filter is prevalent. As the smoothing aspect of the filter, *k*, is increased, the samples considered in the median of the filter are moved further from the center of the sample data, and the sharpening performance is increased.

In the second test the value of $k = 5$ is fixed and the value of l is altered (denoted by 'L' in figures). Results are shown below in Figures 4.16 and 4.17.

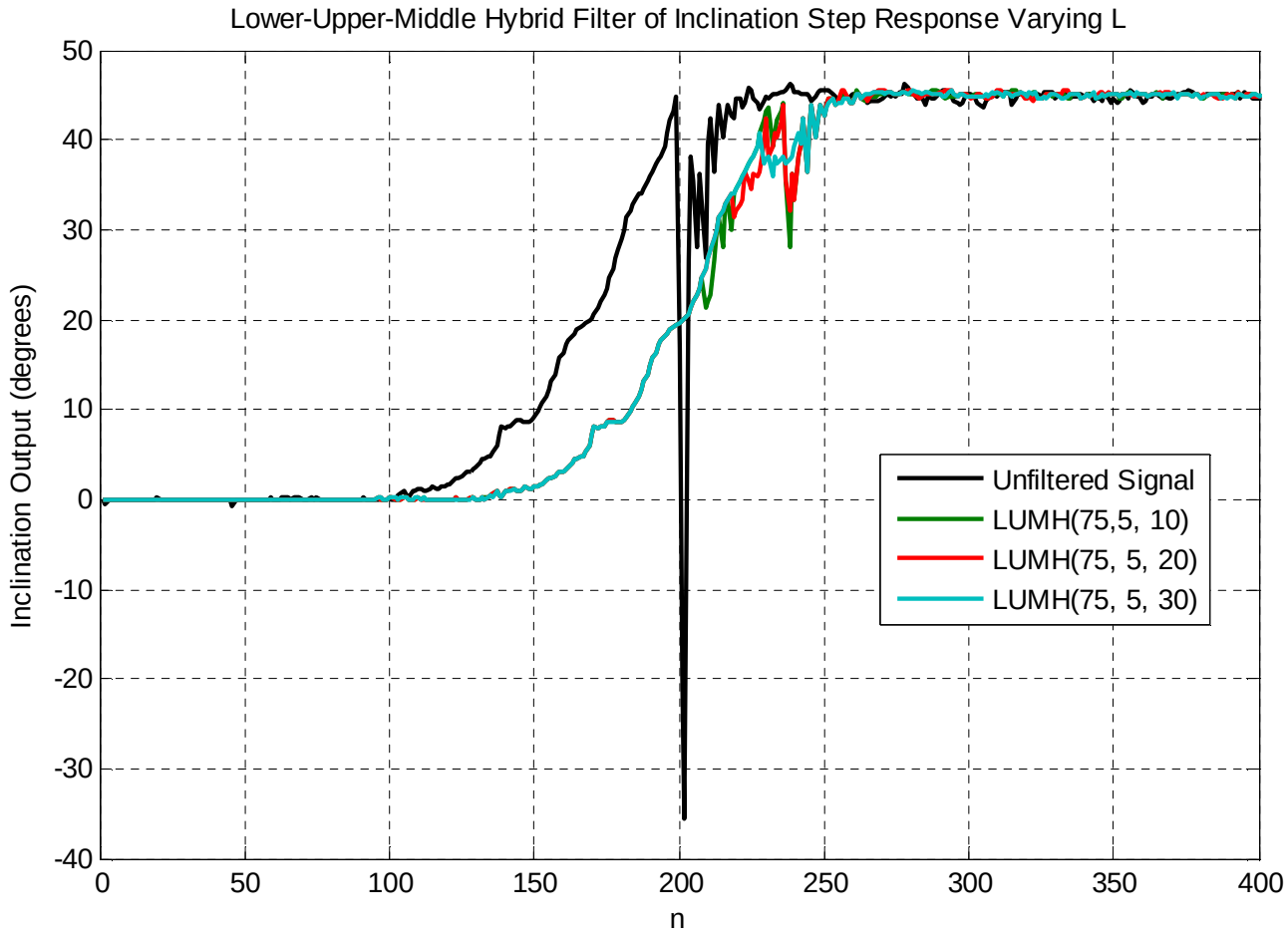


Figure 4.16: LUM filter with *k* = 5 with varying *l*.

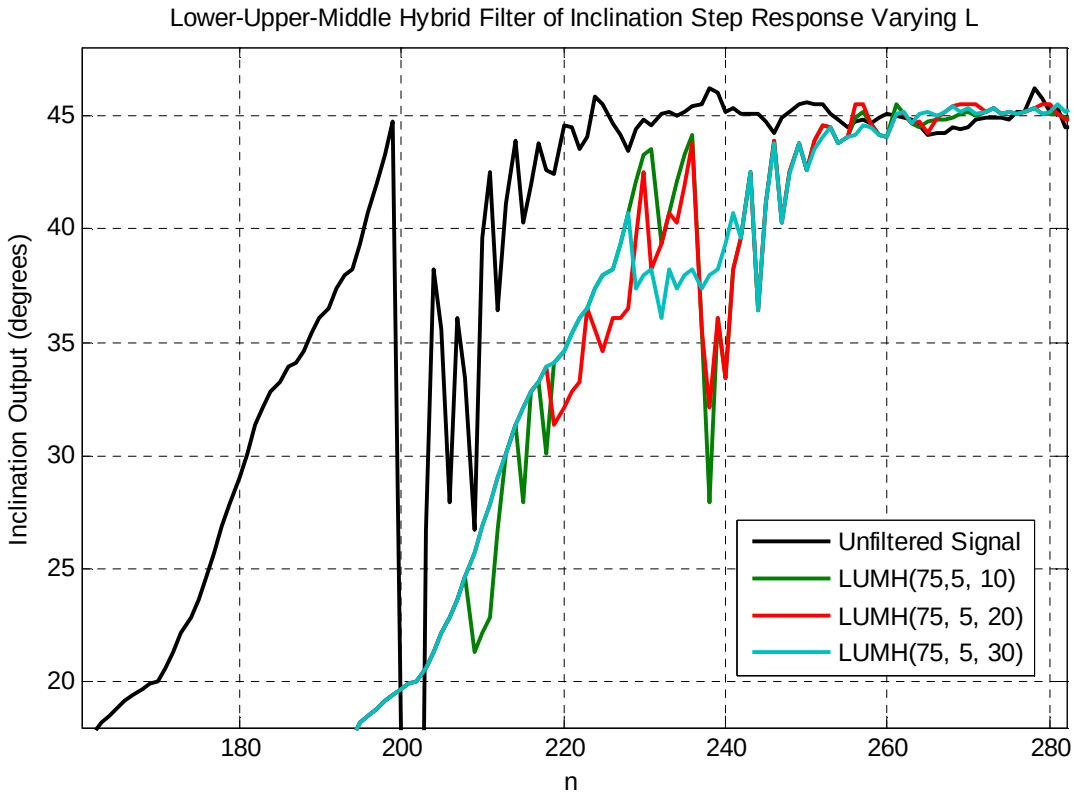


Figure 4.17: Zoomed view of Figure 4.16.

From the figures, the smoothing aspect of the filter is more effective at higher values of *l*, due to the points considered in the median being closer to the center of the data set, thus more closely mimicking a smoothing filter. This hybrid filter has the advantage of maintaining a sharp corner of the step response while still attenuating the high-frequency impulse aspects of the input. However, in this test, this characteristic of the filter is proven to be negligible, as the general transition from increasing to stable signal is approximately consistent through all tests.

4.1.2.4 Implementation

For the purpose of this project, only the LUM smoother is programmed and tested in hardware, as the smoothing aspect of the filter aspect is the desired response for the inclination signal. Further investigation and tuning of the LUM hybrid could prove useful in maintaining the corner portions of the inclination signal, though not investigated here.

Again using the function `VectorCopy` [45] to shift the inclination signal in time, the median of the three samples necessary for the LUM smoother are passed to `_Vector_Median` and the value returned is the output of the filter. Delay of the samples in time is completed when all filtering has taken place. Results calculated and output from the microcontroller can be seen in Figures 4.18 through 4.20 below.

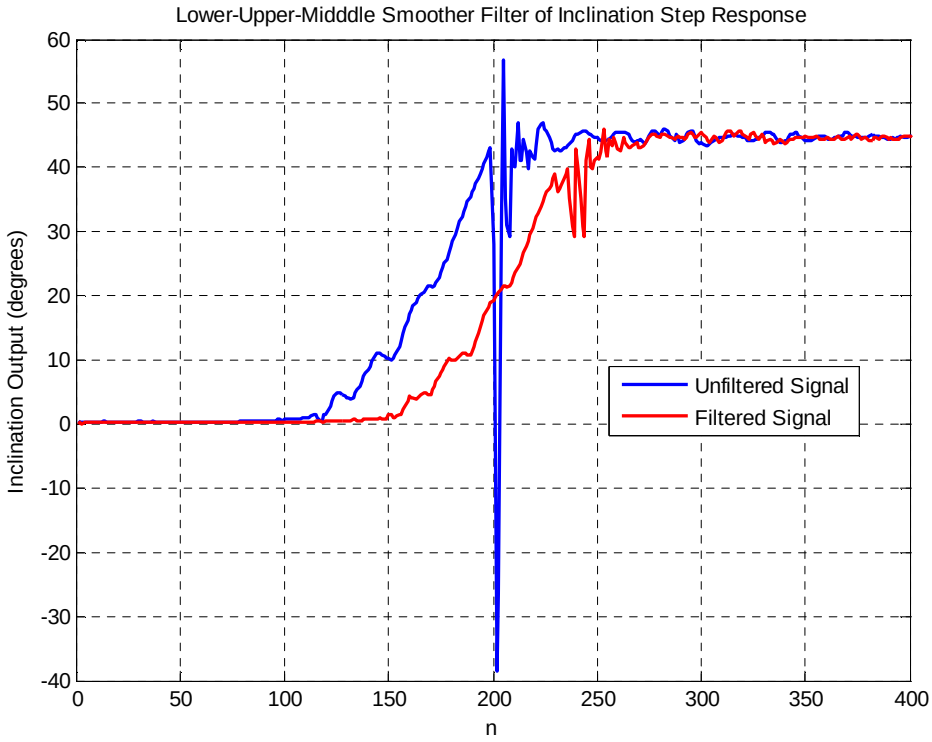


Figure 4.18: Filtered step response output from magnetometer/inclinometer system with LUMS(75, 5) sampled at 500 Hz.

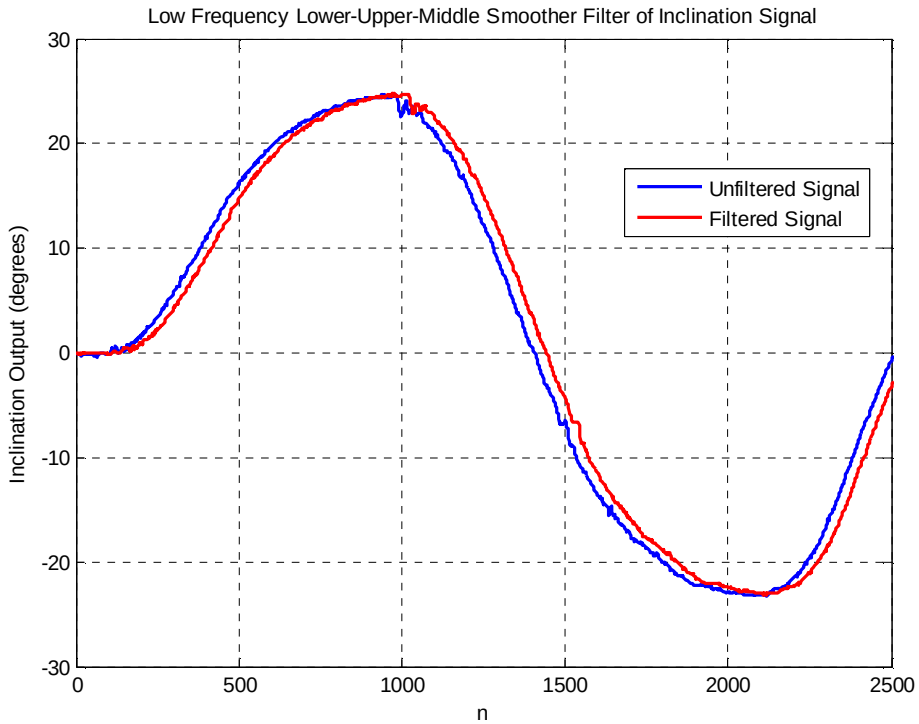


Figure 4.19: Filtered low frequency output from magnetometer/inclinometer system with LUMS(75, 5) sampled at 500 Hz.

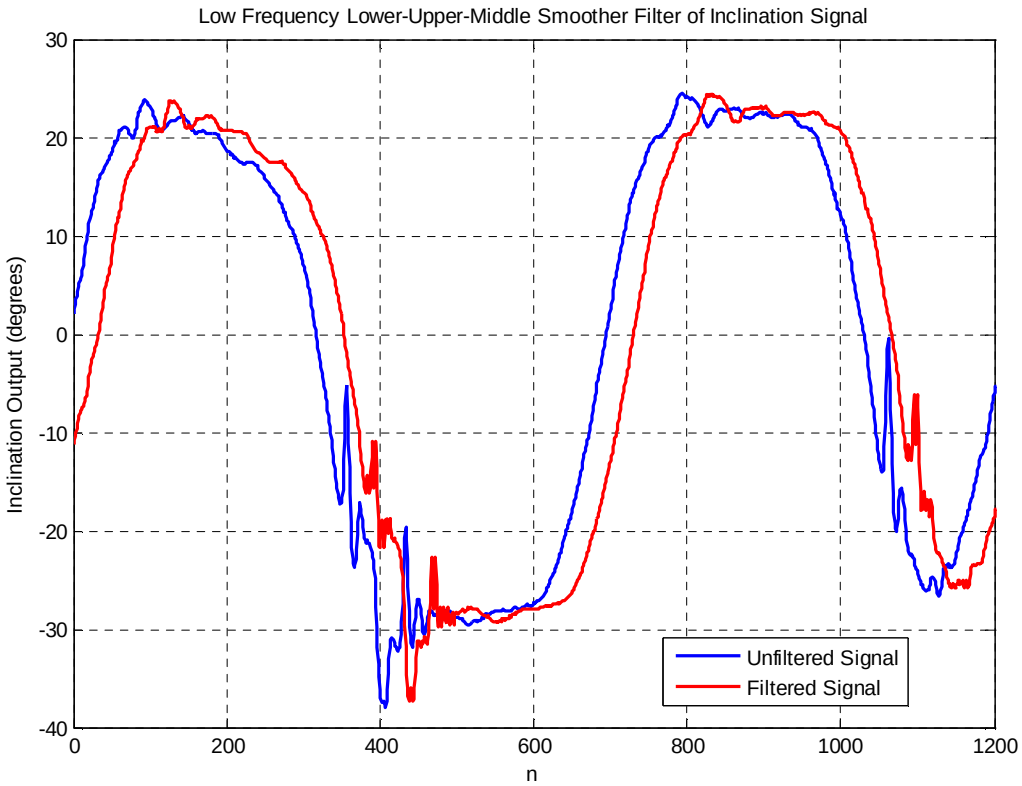


Figure 4.20: Filtered low frequency output with acceleration transients from magnetometer/inclinometer system with LUMS(75, 5) sampled at 500 Hz.

From the results, it is shown that for the same width filter window as the α-trimmed median filter, the LUM smoother is significantly less effective at attenuating the high-frequency noise associated with the inclination signal. Observation of the step response of the filter shows that the impulsive portion of the signal is attenuated, yet significant noise is still present, as was seen in the simulation of the filter.

4.1.3 FIR-Based Median Hybrid Filter

Finite impulse response median hybrid (FMH) filters have been developed, tested and analyzed in detail, including speed improvements and statistical properties, by Heinonen and Neuvo [18-20]. FMH filters have shown significant advantages in impulse and outlier rejection, while still maintaining crucial corner aspects of signals, and thus have been implemented in image processing for contrast improvement [18-20].

FMH filters operate by separating a data set into several different windows of equal or varying length, of which the mean of all contained points are calculated. The output of the filter is then calculated as the median of these several calculated mean values in addition to a substitute center point of the data set, denoted by x_c. The length of the mean-calculating sections can vary. However, for this project, all sections will remain of equal length with the outside, or newest and oldest sections (relative to time) being rounded up. Thus, for a filter window N samples long, each

mean-calculating section contains N/4 samples. All windows in possess widths of 4N+1 samples, resulting in similarly-sized FIR sections.

The FMH filters' advantages come from dsPIC microcontroller's ability to quickly and efficiently carry out FIR filter operations. With FIR coefficients equal to a value of $\frac{1}{g}$, where g is the number of data points in each section, the FIR structure will quickly and efficiently calculate the section mean of a data set. As an alternative to this process, a running sum of each FIR section can be calculated and updated with the incoming and outgoing sample. Continuing, by varying the value of the substitute center data point, x_c, to either the expected mean, maximum or minimum data value, emphasis can be placed on outlier rejection, sharpening corners, or outlier attenuation. For the purpose of this project, the value of x_c will be set as the mean (nominal value) of the expected inclination. This value being zero results in smoothing of the input signal. The formulas below show a FMH filter with five sections, including the center data point.

$$
\begin{aligned}
Y_0 &= \frac{1}{g}(x_0 + x_1 + \ldots + x_{g-1}) \\
Y_1 &= \frac{1}{g}(x_g + x_{g+1} + \ldots + x_{2g-1}) \\
Y_2 &= x^* = 0 \\
Y_3 &= \frac{1}{g}(x_{N-2g+1} + x_{N-2g+2} + \ldots + x_{N-g}) \\
Y_4 &= \frac{1}{g}(x_{N-g+1} + x_{g+1} + \ldots + x_N) \\
y &= \text{median}(Y_0, Y_1, Y_2, Y_3, Y_4) \\
N &= 17 \quad g = 4
\end{aligned}
\tag{4.6}
$$

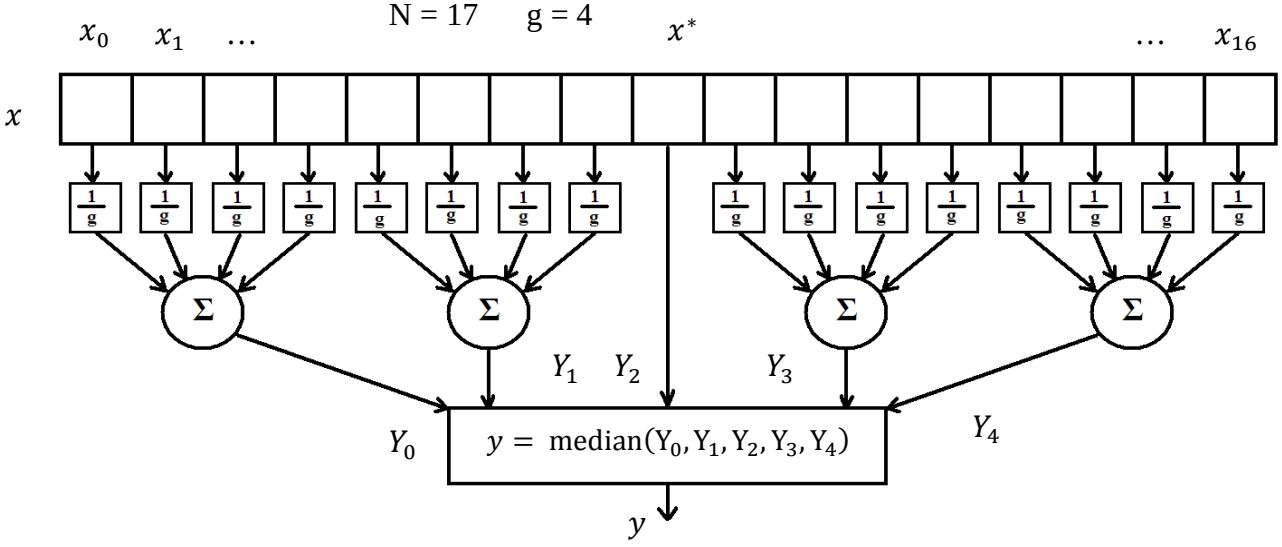


Figure 4.21: Finite impulse response median hybrid filter with N = 17 and g = 4.

4.1.3.1 Simulation

Simulation of the FIR median filter will contain 4 FIR sections and the center point, as pictured above. Larger number of sections allows for improved outlier rejection and flexibility in specification tuning. The width of the filter window N is varied in the simulation results seen in Figure 4.22 below, with the filters denoted by 'FMH(N)'.

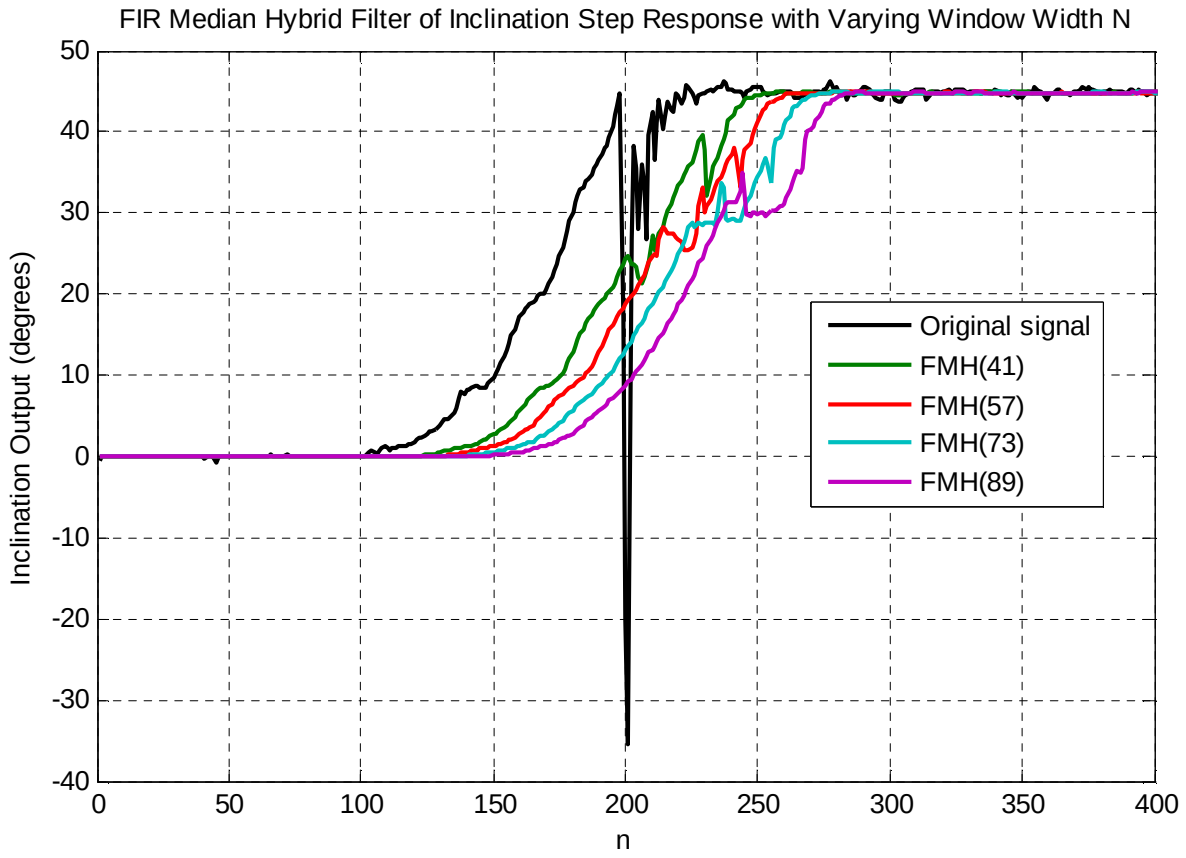


Figure 4.22: FMH-filtered inclination signal with varying filter window width N.

From the results, the FMH filter outperforms the LUM filters discussed in Section 4.1.2 in terms of both signal smoothing and impulse rejection. From observation, the ability to smooth the artifacts of the signal, such as the irregular "bump" that immediately follows the initiation of the step response outperforms both the α-trimmed median filter, as well as the LUM filters. This is due to the sections of the FMH filter acting as moving average filters, and thus slower frequency noise is attenuated with a higher order filter. However, it is also clear that impulsive noise is still evident in the filtered signal. The phase delay of the FMH filter follows the same pattern as for the other nonlinear median filters discussed in Chapter 3, specifically as the filter window becomes larger the phase delay of high-frequency signals also becomes larger, with an ideal step response creating a phase delay of exactly N/2 samples.

A closer view of the filtered outputs is seen below in Figure 4.23.

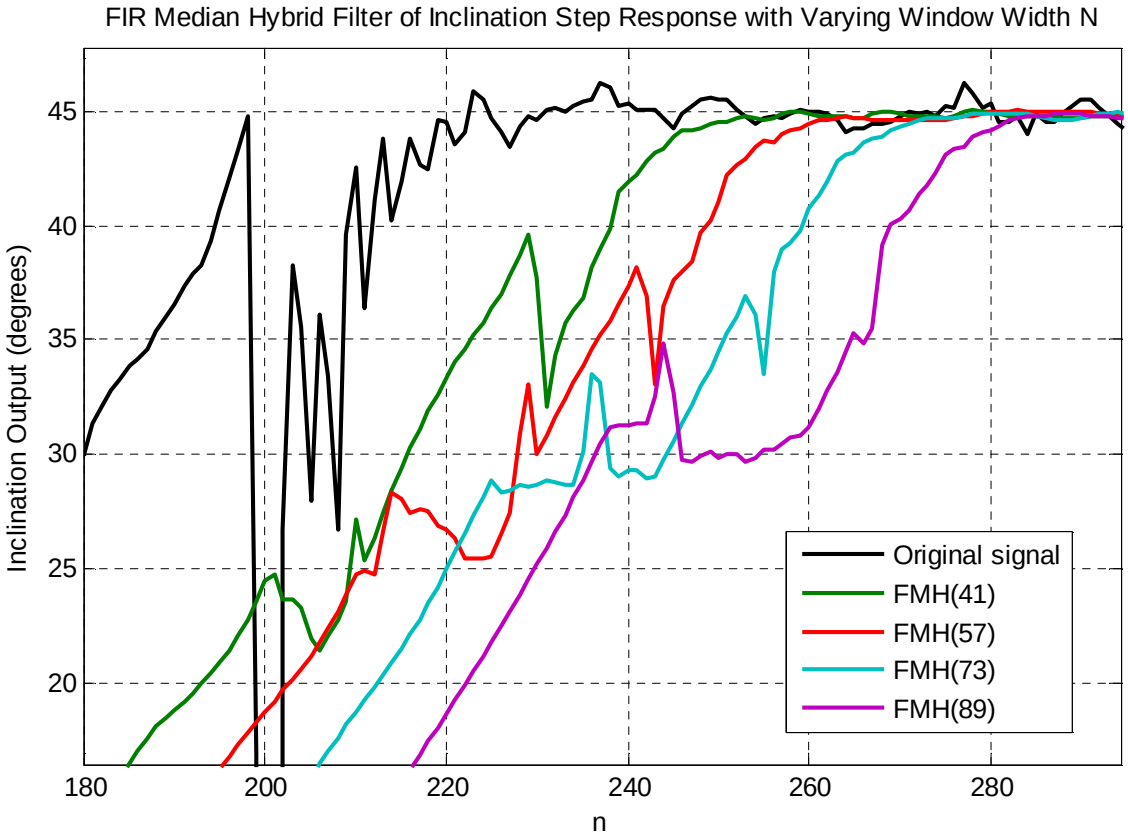


Figure 4.23: Zoomed view of Figure 4.22.

4.1.3.2 Implementation

Performing the FMH median hybrid filter in hardware can be accomplished quickly and efficiently using the DSP core of the dsPIC microcontroller.

Implementation of the DSP library FIR functions to find the median of each FIR window can be executed. The function implements FIR filters using the DSP core of the dsPIC processor's MAC command to multiply and prefetch the next instruction cycle. For the purpose of the FMH filter, the coefficients of the FIR sections (pre-calculated and stored) are inserted into a coefficient vector as $\frac{1}{g}$ where g is the number of samples in each FIR section. Thus, when the built-in DSP function `DotProduct` [45] is implemented on the signal vector and coefficient vector, the mean of the samples in each section is output. The four mean values along with the center substitute point are input into the `_Vector_Median` function, and the median is the filter output. This code is shown in Appendix C.

Hardware calculation and output of filtered signals with a total filter width of N = 73 samples (g = 18 samples per section for four sections) can be seen below in Figures 4.24 through 4.26. A step response, as well as two lower-frequency sample signals are presented. The number of instructions from call to return of the FMH filter is 1,519 instructions for a median window width of N = 73 samples.

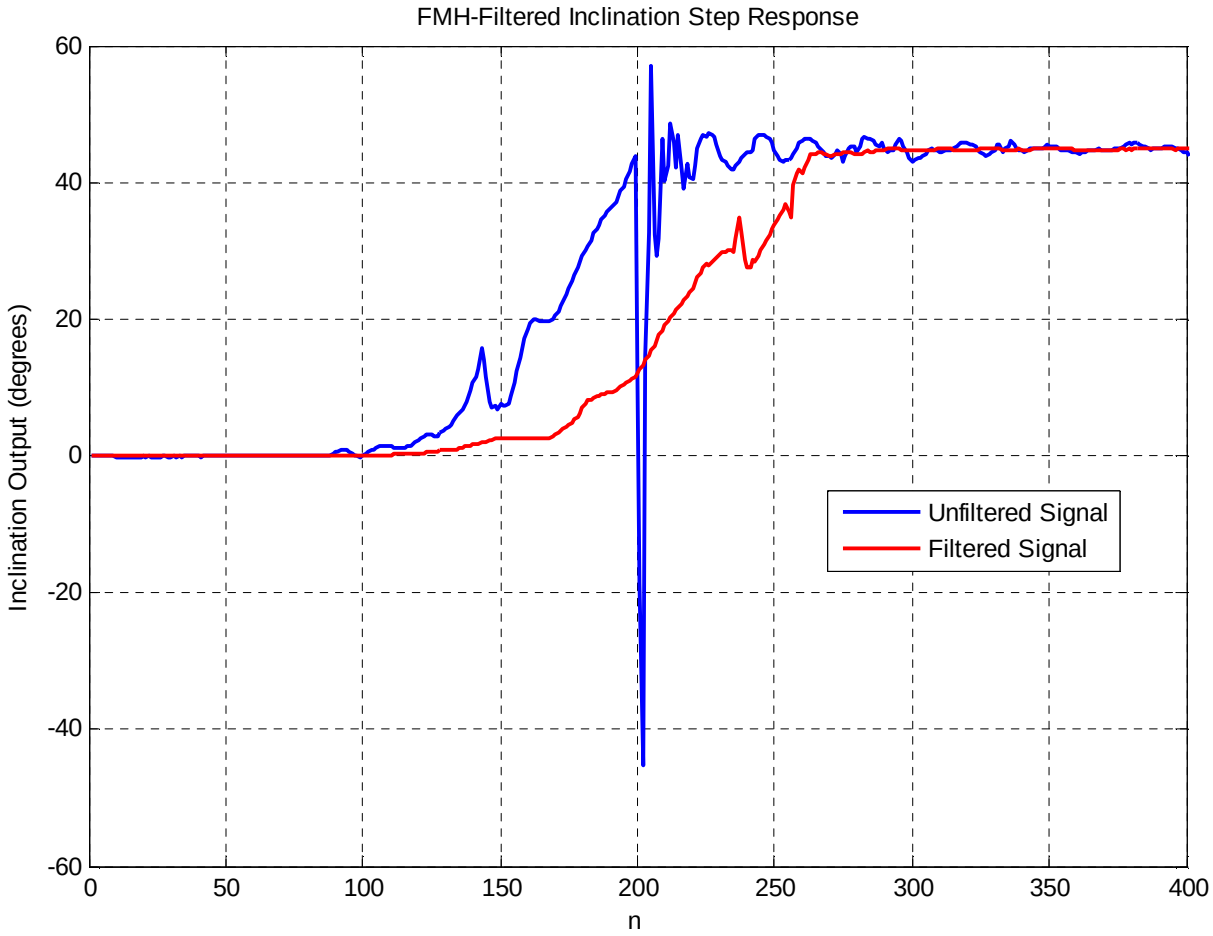


Figure 4.24: Filtered inclination step response with FMH(73) sampled at 500 Hz.

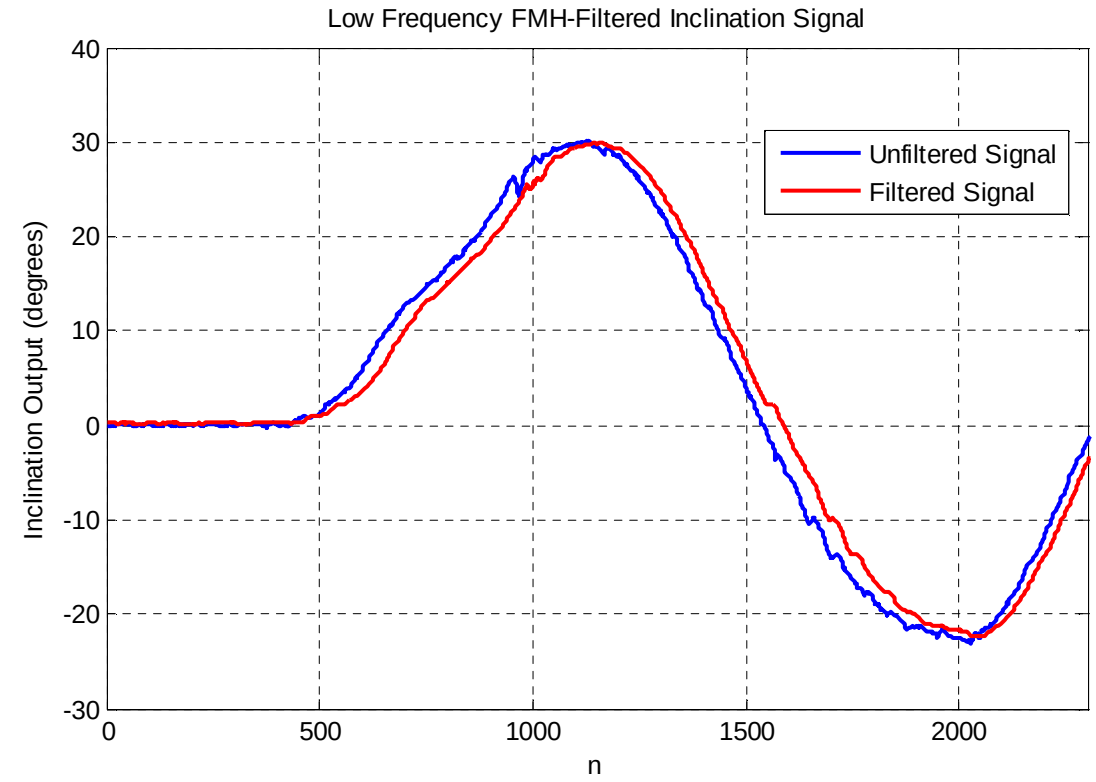


Figure 4.25: Filtered low frequency inclination signal with FMH(73) sampled at 500 Hz.

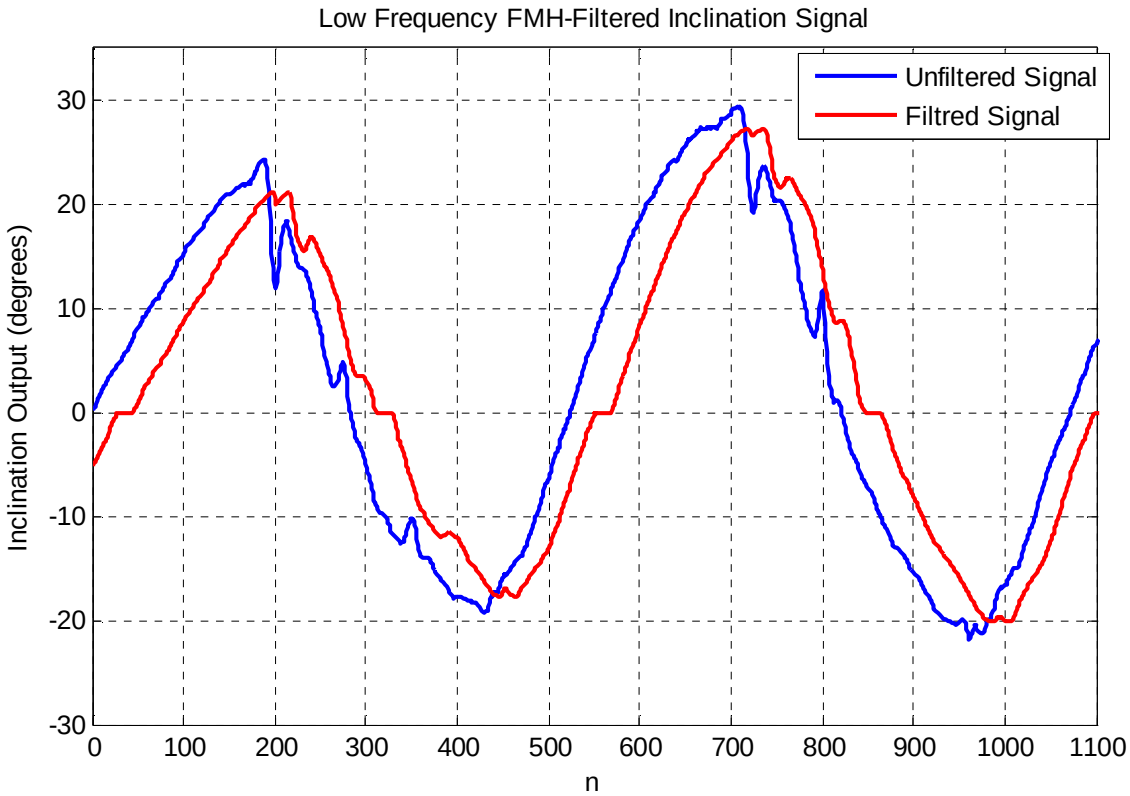


Figure 4.26: Filtered noisy low frequency inclination signal with FMH(73) at 500 Hz.

Hardware implementation reflects results seen in simulation, in that high frequency impulse attenuation is achieved, although remnants of the outliers remain as low frequency perturbations in the output signal. This is especially evident in Figure 4.26 where noise in a low frequency signal is output as smooth perturbations.

4.1.4 Results and Comparison

To compare the hardware-implemented output of the median filters in a side-by side manner, specifications for all of the filters are similarly set. Thus, filter window width N is fixed. This allows for comparable phase delay in the output signal across the filter tests. For the comparison, a window width of N= 49 is implemented. This number allows for definite window center data point identification, and also allows for even number of subsections in the FMH median filter (FMH(49)). The spread of the two data points of the LUM smoother is set at k = 5 samples (LUMS(49,5)), and the trimming value of the α-trimmed median filter is set at α = 0.10 (ATM(49, 0.10)). Three tests are run with the same step input excitations, and the output of each filter is plotted in Figure 4.27 below.

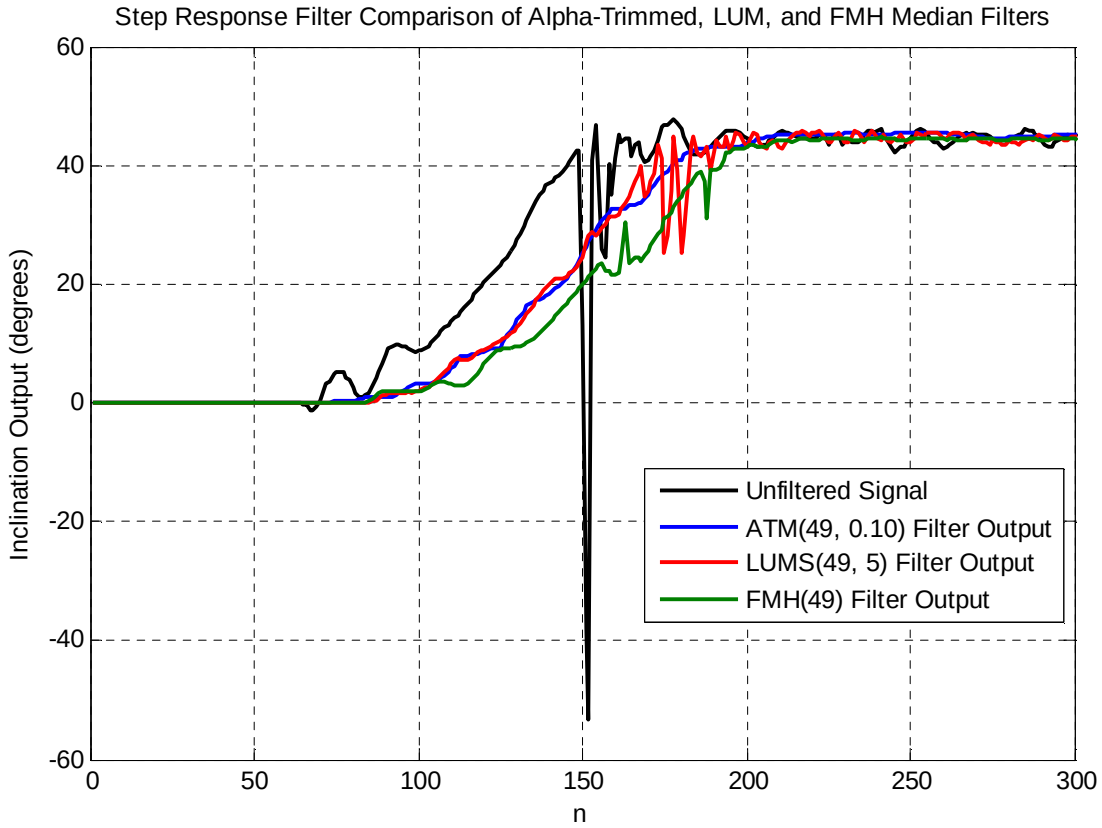


Figure 4.27: Comparison of ATM(49, 0.10), LUMS(49, 5), and FMH(49) median filters on inclination step response.

From the results, the α-trimmed median filter is the most effective method for smoothing the inclination signal, attenuating the impulse outliers, as well as maintaining the corner aspect of the step response. This filter is also the simplest to implement on the microcontroller due to the same median function being required for execution as the other filters. However, with the α-trimmed

median filter, preprocessing and preparation of the filter and input signal is not required, thus saving executed instruction cycles. Therefore, the α-trimmed median filter is the filter of choice for this project. The tuning of the window width of the filter is dependent on the design of the magnetic signal filtering, which is discussed in Section 4.2.

With all of the median filters tested in this chapter being non-linear rank-order filters, phase delay of the filters is also non-linear. However, by nature, median filter delay can be estimated as N/2 samples of delay for a perfect step response, where N is the number of samples in the median filter data set.

4.2 Magnetic Signal Filtering

The -3dB bandwidth of the magnetic sensor module is approximately 1kHz, thus the magnetic sensor is capable of detecting noise at frequencies higher than those of the low frequency fields of interest. The voltage signal output by the AMR magnetic sensors is susceptible to noise from stray magnetic sources, motion, or EMI interference. In addition, ADC noise is produced as sampling frequency increases, reducing the number of noise-free bits. A magnetic signal from the vertical (Z) magnetic axis of a reading taken from inside of a building is seen in Figure 4.27 below, and high frequency noise is shown. This noise is likely from ADC quantization, concluded from the consistent and repeated levels of the data points of the signal.

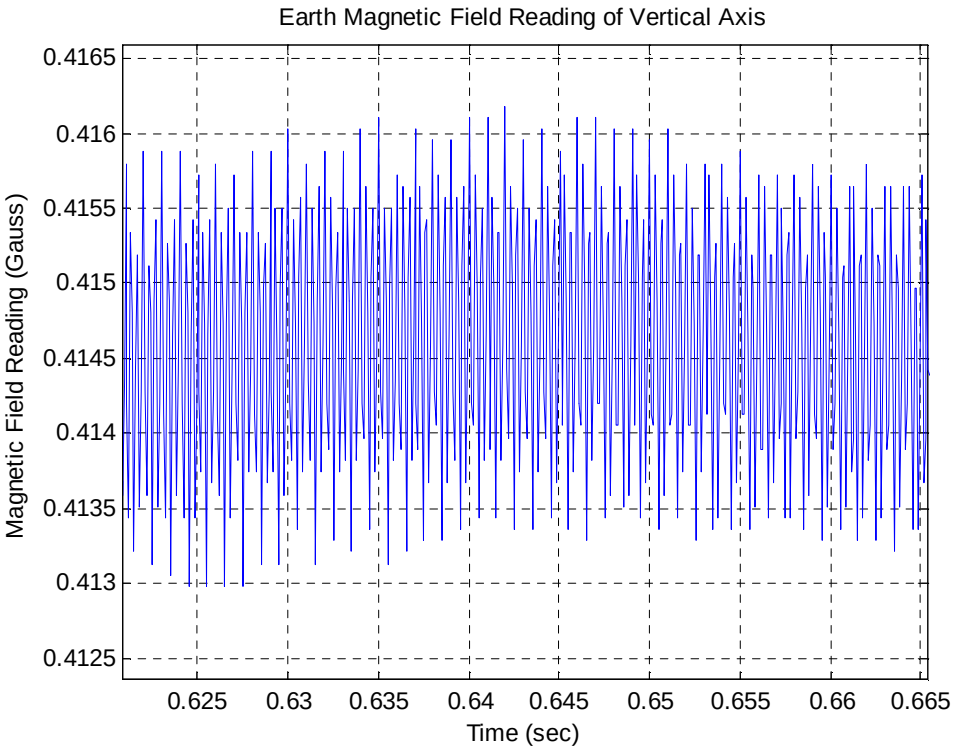


Figure 4.28: Sample magnetic field reading displaying high frequency noise.

The amplitude of the noise present in the signal is minimal (less than 3 milliGauss) relative to the earth field magnitude. However, filtering of the signal improves the signal to noise ratio of the system and the accuracy of the tilt-stabilized readings. In addition, for subsequent user-specific utilization of the tilt-stabilized results of the magnetometer system, filtering of the magnetic signal may be required. The presence of the FIR filter, as well as simple tuning of the filter allow for attenuation of stray noise or magnetic sources at the user's bandwidth of interest.

Filtering of the output signals of the magnetometers with a FIR filter incurs a phase delay that closely matches the delay caused by the median filters implemented on the inclinometer signal. With a similar delay designed for both the inclination signal as well as the magnetometer signal, tilt-compensation resulting from both signals will be more accurate. With similar delay, the output of the system is more reliable and accurate at high rates of acceleration, where phase delay is more prevalent.

4.2.1 Digital Filters

Since the incorporation of DSP cores within microprocessors, digital filters have been widely used for filtering and digital signal processing. Derived from the microcontroller's ability to execute the MAC (multiply and add to accumulator), MPY (multiply and store in accumulator) and MSC (multiply and subtract from accumulator), the digital filter executes instructions that process these signals at high rates of speed. The DSP instruction set derives its efficiency from multiplying two numbers that are stored in different banks of memory onboard the chip, and adding or subtracting them from a large (number of bits) accumulator. At the same time, the instruction prefetches, or preloads the variables multiplied in the next instruction, into memory for the same operation. This process takes significantly less time to execute than traditional sequential multiply and add (or subtract) instructions.

Significant work in the field of digital filtering has been accomplished in the last thirty years, and large amounts of documentation on various design estimation, processes, characteristics, and performance metrics are readily available.

4.2.1.1 Infinite Impulse Response (IIR) Filters

Infinite impulse response (IIR) filters are filters that incorporate feedback in their operation. IIR filters can have several different structures, including canonic, direct form, lattice or transposed, depending on the specifications of the filter and the signal being processed [13].

Despite the structure of the filter, typical IIR filters execute a simple difference equation, as seen below in Equation 4.7.

$$\sum_{i=0}^{M} a_i \, y_{n-i} = \sum_{j=0}^{N} b_i \, x_{n-j} \tag{4.7}$$

Advantages of IIR filters include steep drop-off while maintaining low filter order, as well as smaller hardware memory requirements. IIR filters, however, possess the possibility (if poorly implemented) to be unstable. Depending on the design, IIR filters have different phase delay characteristics. For the purpose of this project, IIR filters will not be implemented, instead using FIR filters, discussed in the next chapter.

4.2.1.2 Finite Impulse Response (FIR) Filters

Finite impulse response filters are digital filters that typically (most are non-recursive) incorporate no feedback in their calculation. A simple summation formula calculates the output of the filter, as seen below in Equation 4.9.

$$y_n = \sum_{i=0}^{N} b_i \, x_{n-i} \tag{4.8}$$

Advantages of the FIR filter are unconditional stability, as well as generalized linear phase delay, depending on design. Disadvantages of the FIR filter include larger memory requirements when compared to IIR filters, as well as larger phase delay, both due to the higher order of the filter.

For this project, however, the high order of the FIR filter (and thus larger delay) will be exploited to approximately match the phase delay caused by the high order of the median filters previously discussed. With simulation and testing, a generalized linear phase delay similar to the nonlinear phase delay of the median filters will be achieved, thus reducing error.

Estimation of the order of a FIR filter to obtain characteristics of stopband ripple and transition bandwidth has been approximated by Kaiser with the following formula:

$$\widehat{M} = \frac{-20\log(\delta_s) - 7.95}{14.36 * \left(\frac{\omega_s - \omega_p}{2\pi}\right)} \tag{4.9}$$

Where δ_s is the stopband ripple, and ω_s and ω_p are the stopband and passband cutoff frequencies, respectively. As an example for simulation, a desired passband fractional frequency of 0.25 and stopband frequency of 0.29 with -40 dB of stopband ripple would require a filter of 56th order.

$$\widehat{M} = \frac{-20\log(0.01) - 7.95}{14.36 * \left(\frac{2\pi * 0.25 - 2\pi * 0.29}{2\pi}\right)} = 55.8 \tag{4.10}$$

The order estimation changes with the windowing technique and design method, but this method is a reliable first-cut estimation of the required filter order.

4.2.2 FIR Filter Design and Simulation

To filter the high-frequency noise from the magnetometer signal, a lowpass filter design is implemented. Design of a generalized linear phase (GLP) FIR filter begins with the desired frequency response, in order to obtain the impulse response, or coefficients of the filter. The frequency response of a lowpass filter design is seen in Equation 4.11 below.

$$H_{LP,n}\left(e^{j\omega n}\right) = \frac{1}{2\pi}\int_{-\widehat{\omega}_c}^{\widehat{\omega}_c} e^{j\omega n}\, d\omega \tag{4.11}$$

This equation simplifies to

$$H_{dLP,n}\left(e^{j\widehat{\omega_c}n}\right) = \frac{\widehat{\omega}_c}{\pi}\frac{\sin\left(\widehat{\omega}_c n\right)}{\widehat{\omega}_c n} = \frac{\widehat{\omega}_c}{\pi}\operatorname{sinc}\left(2\widehat{\omega}_c n\right) \tag{4.12}$$

In order to center the frequency response within the filter window, the response is shifted by $\frac{N-1}{2}$ samples, where *N* is the order of the filter, an odd number. This number is kept odd to allow for a center point of the filter window.

To incorporate different filter characteristics and obtain the impulse response of the filter, this function can be weighted by different window designs. For the purpose of this project, the windows that are considered for weighting are the rectangular, Blackman, and Kaiser window (with β shaping factor = 5.0). All of these windows with a width of 51 samples can be seen in Figure 4.29. All code used to design and simulate FIR filters in this chapter can be found in Appendix B.

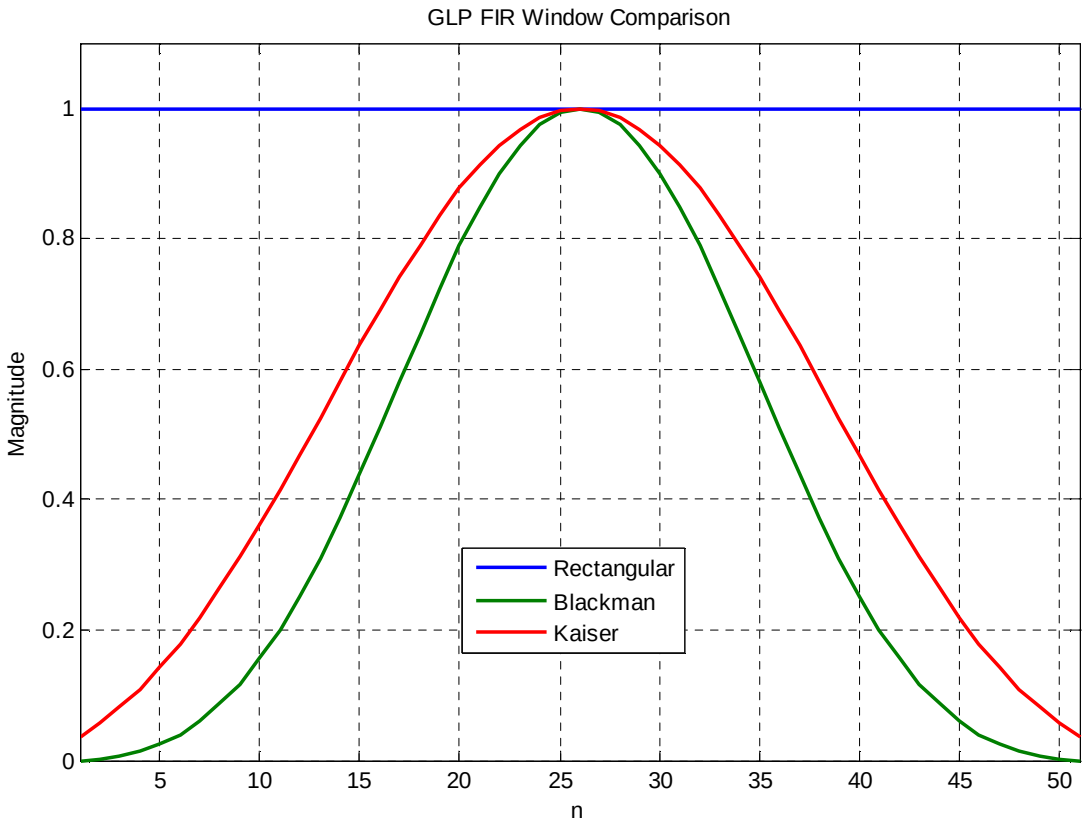


Figure 4.29: Plot comparison of FIR weighted windows with 51st order window.

The rectangular window maintains the frequency response as it is calculated. The Blackman window places emphasis on the center lobe of the window, where the Kaiser window places more emphasis on the outer edges of the window, in comparison. Weighting the frequency response by each window, the filter impulse response and coefficients are obtained. The three plots in Figure 4.29 to 4.31 below display the impulse responses of the three different filter designs. The responses displayed were designed to have a cutoff fractional frequency, or percent of the sampling frequency of 0.25. For example, a signal sampled at 1000 Hz would be cut off at the -3 dB frequency of 250 Hz.

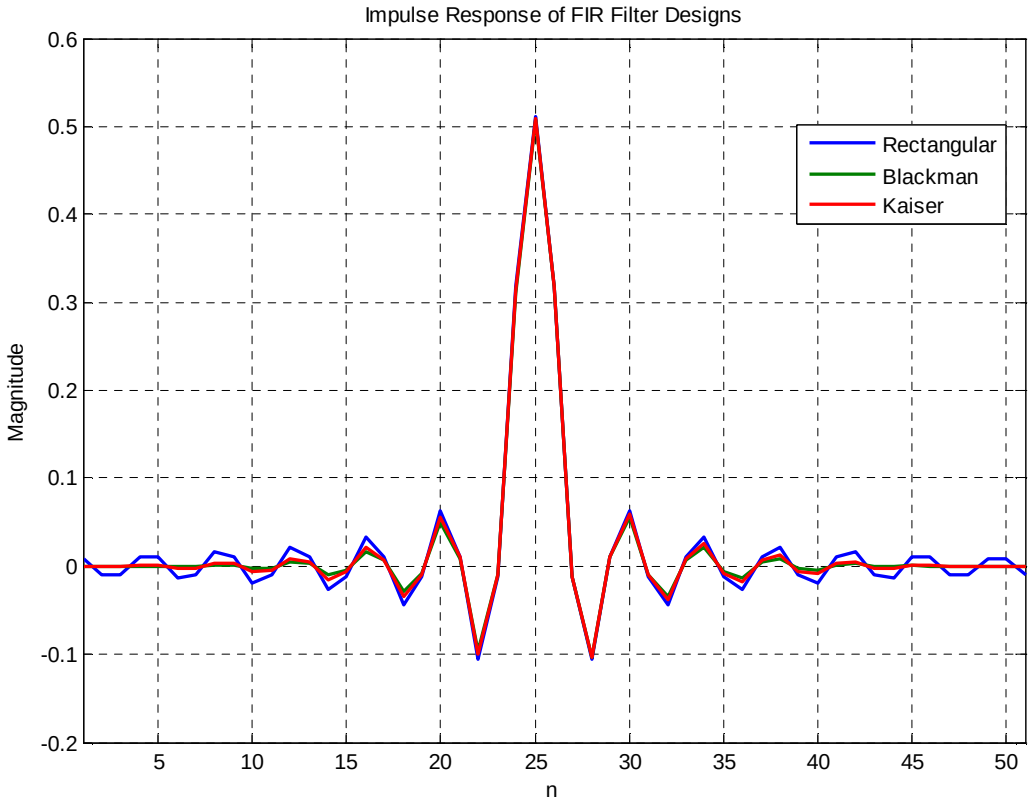


Figure 4.30: Impulse response of FIR filter designs.

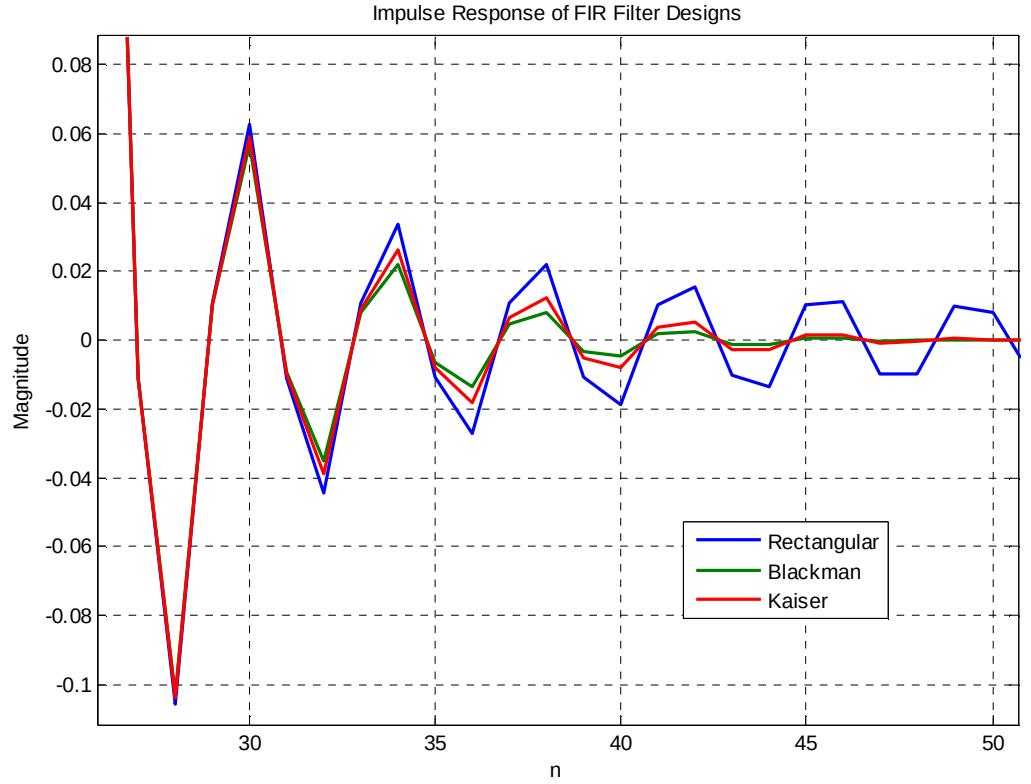


Figure 4.31: Zoomed view of impulse response of FIR filter designs.

From the impulse response functions, it is shown that the Blackman and Kaiser windows place less weight on the outer edges of the filter window, where the rectangular window, or the original desired frequency response has equal weight on all parts of the window. Implementing a Fourier transform on these impulse responses calculates the frequency response of the filters.

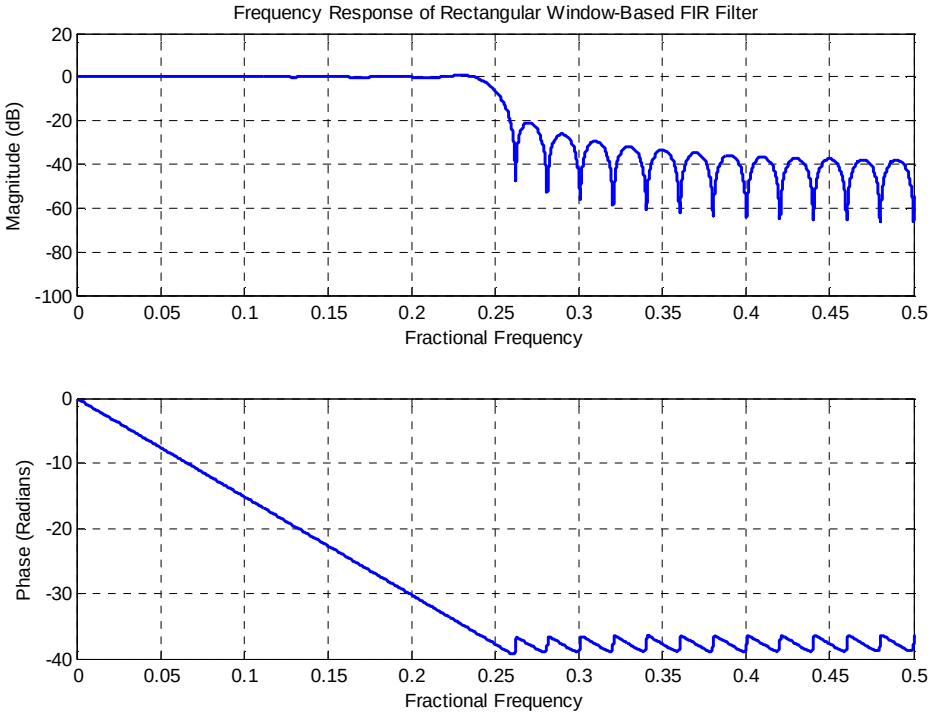


Figure 4.32: Frequency response of 51st-order rectangular window-based GLP FIR filter.

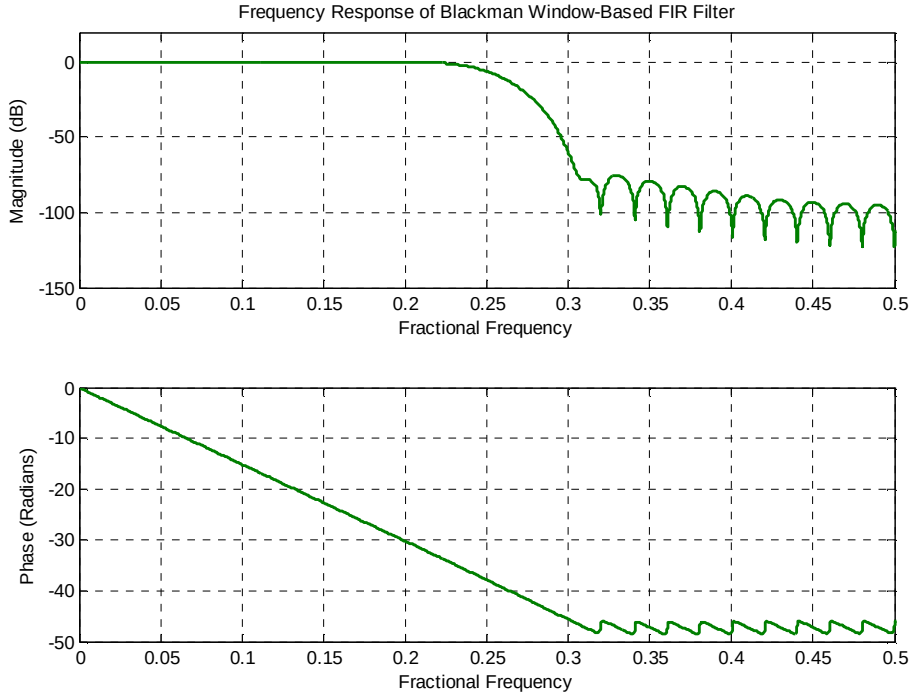


Figure 4.33: Frequency response of 51st-order Blackman window-based GLP FIR filter.

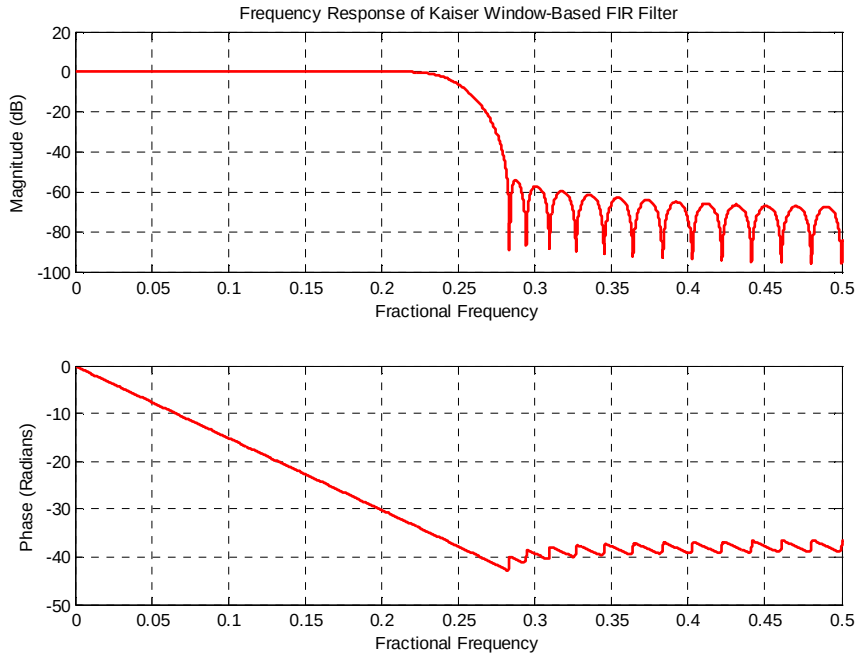


Figure 4.34: Frequency response of 51st-order Kaiser window-based GLP FIR filter.

From the frequency responses above, it is shown that designing for a 0.25 fractional cutoff frequency has yielded a lower cutoff frequency. Depending on the filters tested, the frequency varies at approximately 0.24. Tuning of this frequency is a simple process that can be incorporated in the design process. It can be noted that with higher order, the transition band of the filter becomes narrower, and the thus the drop-off at the cutoff frequency becomes steeper. This is shown in Figure 4.35, which is of the same Kaiser design as in Figure 4.34, but now of 151st order.

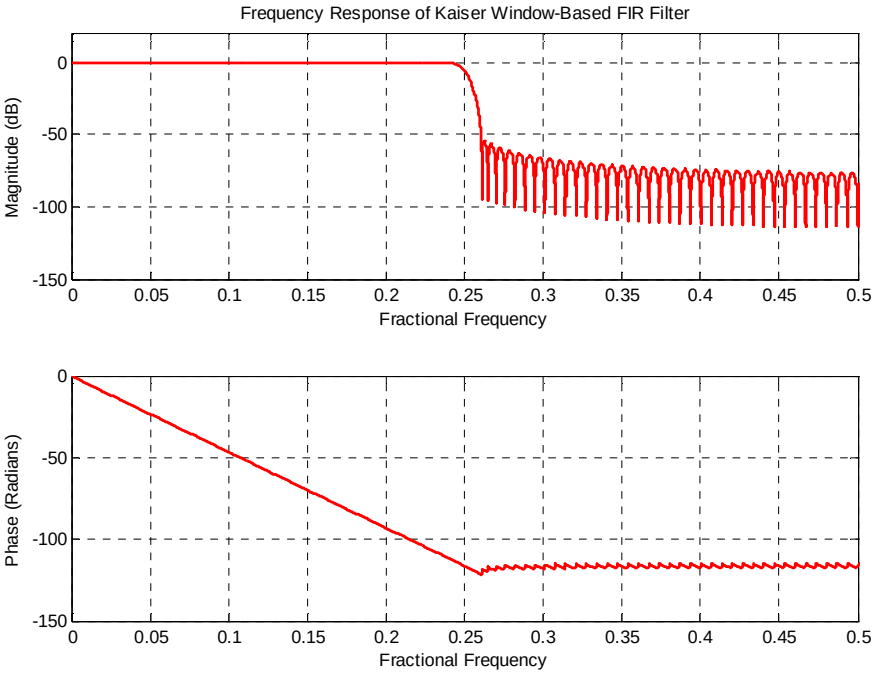


Figure 4.35: Frequency response of 151st-order Kaiser window-based GLP FIR filter.

4.2.3 Hardware Implementation

To implement the FIR digital filters simulated in Section 4.2.2 above, the coefficients of the filters are converted to fractional format. Using 16 bits, this 1.15 format represents the full scale value of -1.000 to +0.999695 (0x8000 to 0x7FFF, respectively). The task of signal filtering is accomplished implementing a built-in DSP function from Microchip that converts floating point numbers to fractional numbers called `Float2Fract` [45] . Upon this conversion, the 64-bit MATLAB coefficients are quantized to the microcontroller's 16 bit format.

To implement the filter, a FIR filter function contained in the DSP library [45] implements the MAC command along with a DO loop to quickly and efficiently calculate the output of the filter based on Equation 4.8.

To implement the MAC command (as well as MPY and MSC in IIR filter implementations), the coefficients of the filter must be stored in a bank of memory called the X-data memory, which represents half of the 4 kbytes of data memory onboard the microcontroller. The filter states must be allocated to Y-data memory, which occupies the other half of data memory. This allocation allows the DSP commands to access both pieces of filter data simultaneously, increasing efficiency and speed of operation.

To test the operation of the FIR filter code, the Blackman filter designed above, and seen in Figures 4.29, 4.30, 4.31 and 4.33 is tested. After storing the coefficients, allocating memory space for delays, and initializing the system by setting all delays to 0 (0x0000), a simple impulse function is input into the filter. This impulse function is represented by a value of "one" followed by 255 values of "zero" (0x7FFF, 0x0000, 0x0000, 0x0000,…). The output of the filter is then saved and tested once again in MATLAB by calculating the frequency response of the impulse function, or transfer function of the system [13]. The filter transfer function is now the hardware implemented, quantized and real-time output of the FIR filter.

Comparison of the simulated and hardware implementation of the 51st order Blackman filter is seen below in Figure 4.36.

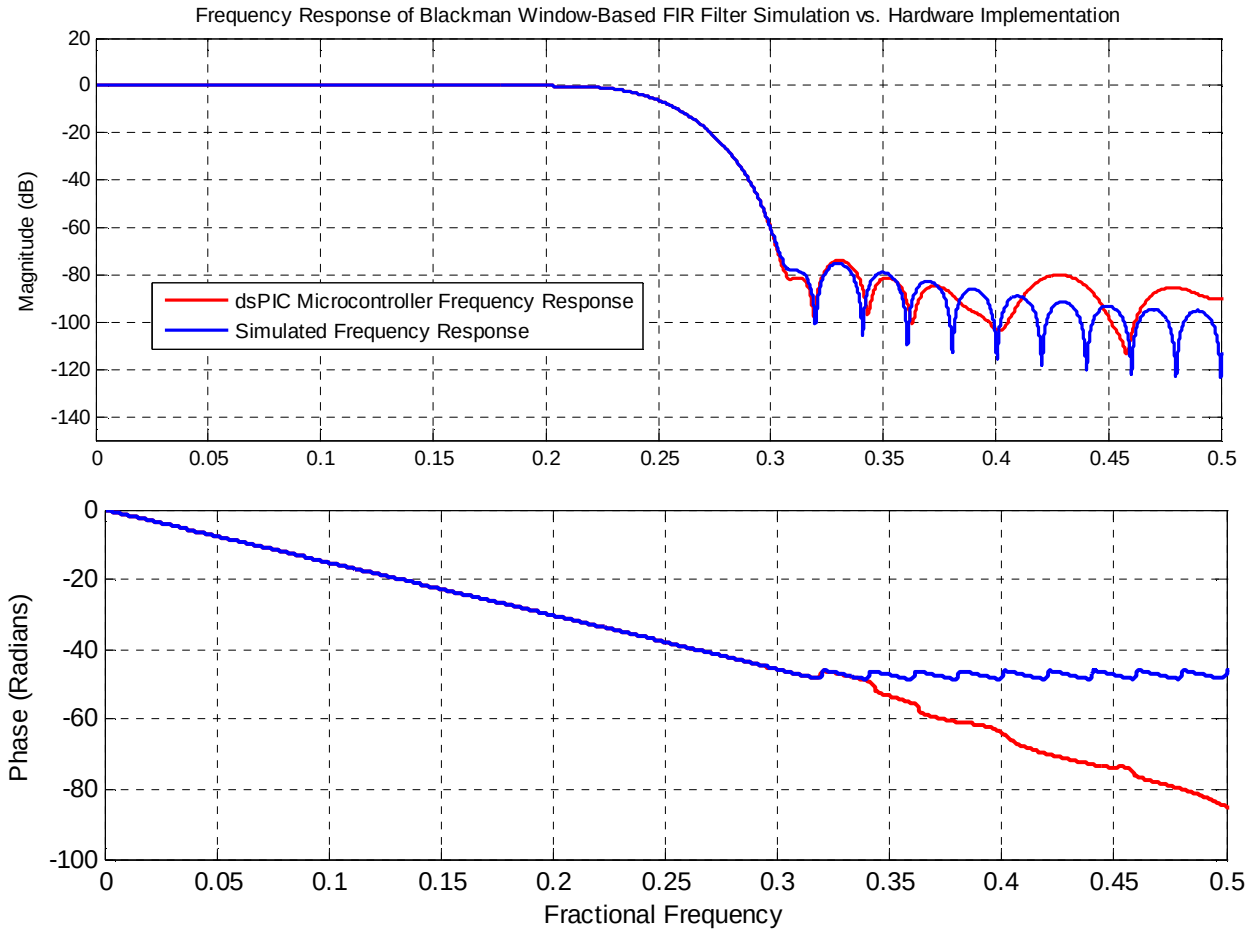


Figure 4.36: Simulated and dsPIC hardware implemented frequency responses of 51st-order Blackman window-based GLP FIR filter.

The hardware implemented filter response falls within 0.002 dB of the simulated filter, as seen below in Figure 4.37, a zoomed view of a portion of the magnitude output in the passband of the filter. The filter response begins to degrade from the simulated filter in magnitude and phase in the stopband, due to the quantization of the filter components to the 16-bit format of the dsPIC microcontroller.

Cutoff frequency is slightly lower (approximately 0.24) than designed (See Figure 4.37 below), as a result of the window design of the filter. This is also reflected in the simulations seen above. Tuning of the final cutoff frequency can be achieved by tuning the cutoff of Equation 4.11 in the early design steps.

In this passband, however, all performance characteristics are as designed. The phase is linear and the cutoff frequency, stopband frequency, as well as passband ripple are consistent with design parameters.

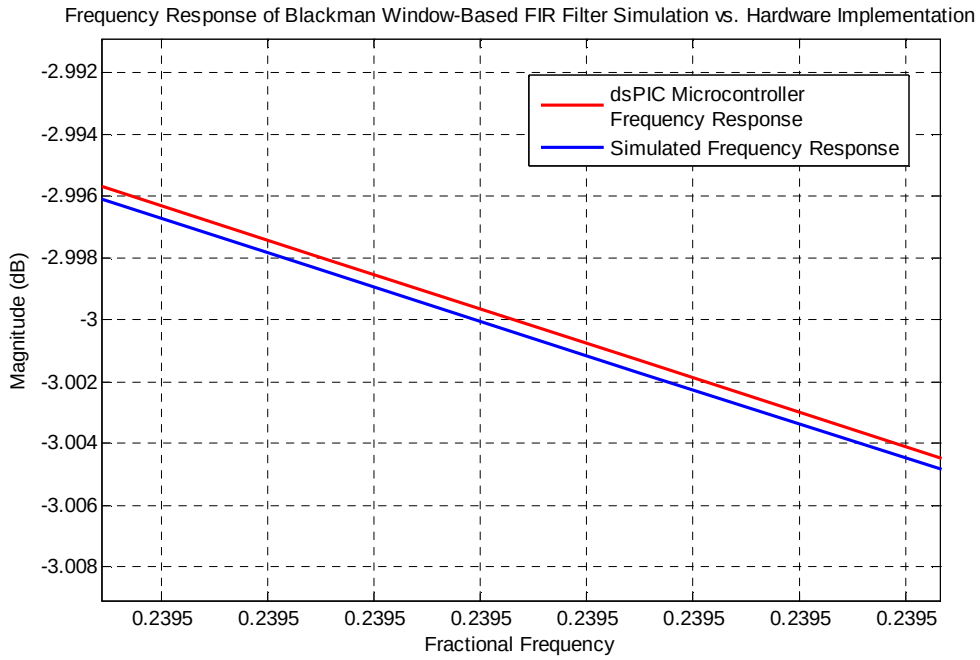


Figure 4.37: Zoomed view of passband portion of Blackman window-based GLP FIR filter.

Testing the filter on the same magnetic signal seen in Figure 4.28, noise attenuation is achieved, as is seen in Figure 4.38 below.

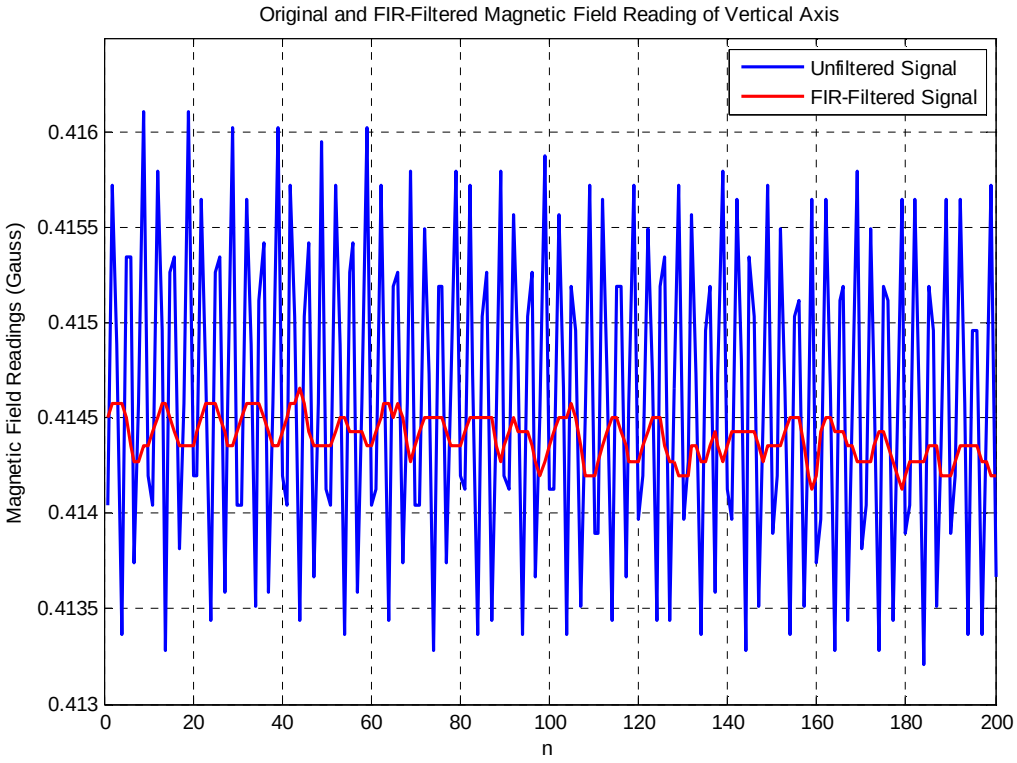


Figure 4.38: Filtered reading of Earth's magnetic field in the vertical axis of detection. Filtered with 51st order Blackman GLP FIR filter.

From Figure 4.38 the attenuation of the signal noise is even further apparent, although a lower cutoff frequency would reduce the noise remaining in the output signal. Adjustment of this cutoff frequency and the filter order will alter the output of the filter to produce more noise attenuation. Adjustment of these filter specifications is discussed further in Chapter 7.

To display the phase delay of the filter, a sample magnetic field step response from the tilt platform is performed, and the filter is implemented on the signal in hardware. The test is run at 500 Hz, similar to all other tests thus far.

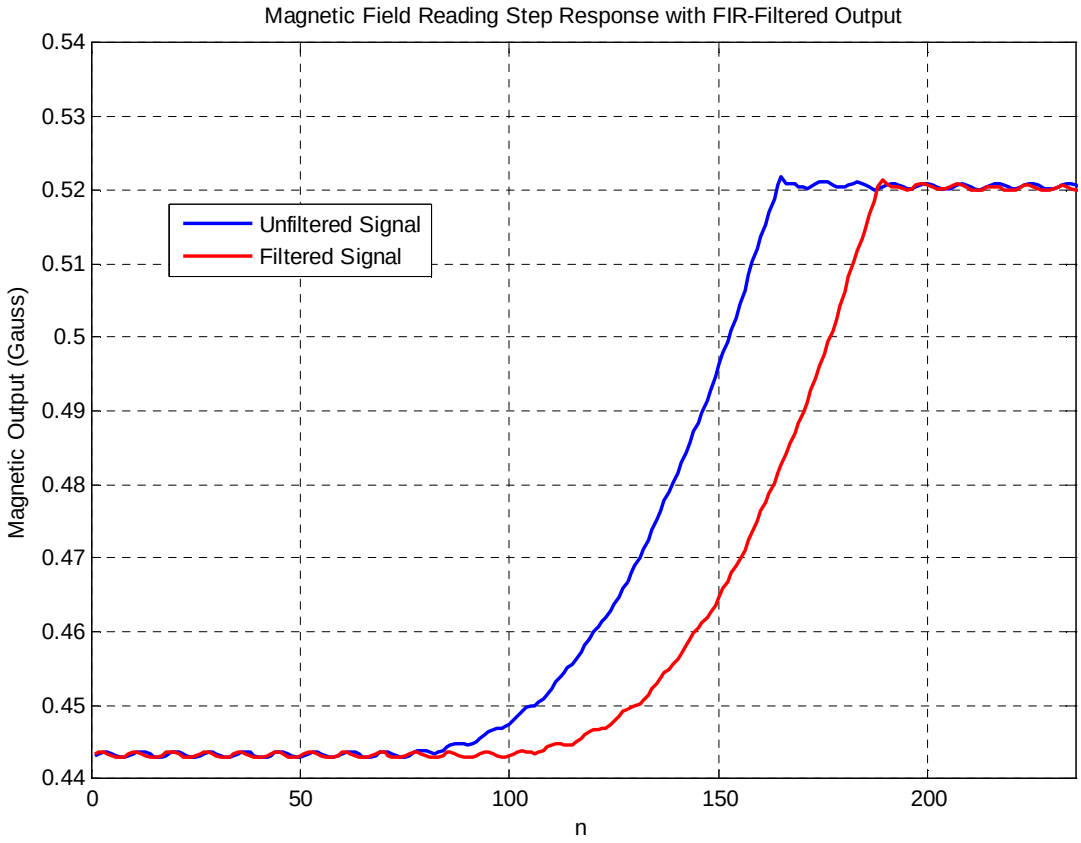


Figure 4.39: Step response of magnetometer and FIR-filtered output.

The phase delay of the FIR filter is approximately linear (as seen in Figure 4.36), as expected, and with the step input, the generalized linear phase delay is comparable to that of the non-linear median filters discussed in Section 4.1. Upon the high-frequency stimulation, the phase delay is approximated by N/2 samples, as expected from the frequency response and center-weighted FIR filter architecture. It is also of note that the noise filtered in Figure 4.38 is of such small amplitude that it is not easily visible in the step response signal. However, with the phase delay associated with the filter design, the inclination and magnetic signals can be similarly delayed, thereby reducing phase noise between the linear and non-linearly filtered signals.

Chapter 5 Trigonometry and CORDIC

Tilt-compensation of the magnetometer system output requires the sine and cosine transformation of inclinometer output angles. In this chapter, two methods for this calculation are tested. Traditional calculation of the sine and cosine of this angle is implemented in hardware and tested for operational speed. As an alternative, the CORDIC algorithm is simulated and tested in hardware for possible advantages in operational speed and bandwidth.

5.1 CORDIC

The Coordinate Rotation Digital Computer, or CORDIC, was developed and published in 1959 by Jack E. Volder. The computing technique was designed to be implemented on real-time navigational systems where trigonometric computing relationships were to be calculated quickly and efficiently [25]. Since its inception, CORDIC has been adapted to calculate Discrete Fourier Transforms, exponential, logarithm, forward and inverse circular and hyperbolic functions, ratios and square roots [1, 25, 28-30, 44].

5.1.1 Background

The CORDIC algorithm is a simple iterative process of shifts and adds whose result converges linearly by one more bit of accuracy for every additional iteration performed. Hence, integer mathematics can be incorporated in the functions listed above, rather than converting to floating point representation onboard a microcontroller. It must be noted that implementation of floating point arithmetic is an accurate method for the calculation of any of the above mentioned functions. However, floating point calculations are notorious for requiring an excessive number of instruction cycles per calculation compared to integer math, and thus execution time and processor data throughput speeds are significantly reduced. As another alternative, lookup tables for the trigonometric values of a given angle are a fast alternative to real-time calculation of the results. Lookup tables, however, require large amounts of valuable onboard memory.

For the purpose of this project, the CORDIC algorithm provides a balance of execution time versus memory savings. The algorithm is advantageous for implementation onboard the dsPIC, since calculation efficiency with minimal memory requirement is ideal. With faster execution time of onboard calculation, bandwidth can be increased with a higher sample rate. If bandwidth required is low, the additional instruction cycles available between samples can be used for subsequent tasks.

5.1.2 Algorithm

For the purpose of calculating the sine and cosine of an angle for the onboard inclination algorithm discussed in Chapter 6, vector rotation using CORDIC is explored.

Calculation of the sine and cosine of an angle Θ can be represented by the coordinates (x and y) of the unit vector at that angle, seen in Figure 5.1.

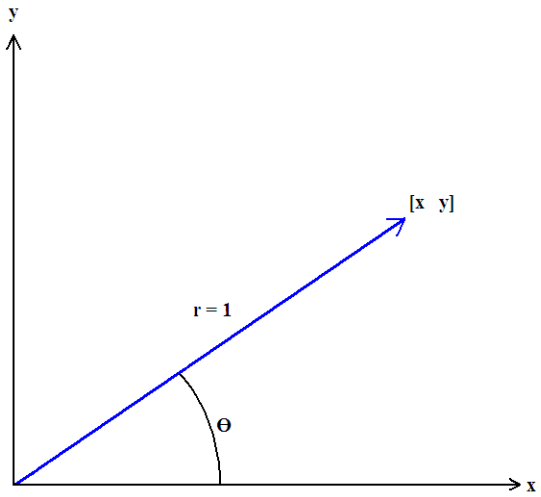


Figure 5.1: Vector [x y] at angle Θ on the unit circle.

Vector rotation occurs by transforming the vector using the rotation matrix:

$$\begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} \tag{5.1}$$

Thus, rotation through a change in angle Θ occurs via the following transformation:

$$[x' \quad y'] = [x \quad y] \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix} \tag{5.2}$$

Where $[x' \quad y']$ is the vector resulting from the transformation through angle Θ [1, 25, 28-30].

The transformation is accomplished via a series of *m* iterations, such that:

$$\Theta = \sum_{i=0}^{m} \Theta_i \tag{5.3}$$

The iterative process of the transformation is reflected as the product of the iterations, and by utilizing the definition of tangent:

$$[x' \quad y'] = \left[\prod_m \cos\Theta_m\right] [x \quad y] \prod_m \begin{bmatrix} 1 & \tan\Theta_m \\ -\tan\Theta_m & 1 \end{bmatrix} \tag{5.4}$$

The key to the effectiveness and efficiency of the CORDIC algorithm is based on restricting the angle of the $\tan\Theta_m$ term to a power of two, or a simple right bit shift in base-2 integer math:

$$\tan \Theta_m = \pm \frac{1}{2^m} \tag{5.5}$$

Thus:

$$\begin{aligned} [x' \quad y'] &= \left[\prod_m \cos\left(\tan^{-1} 2^{-m}\right)\right] [x \quad y] \prod_m \begin{bmatrix} 1 & \pm 2^{-m} \\ \mp 2^{-m} & 1 \end{bmatrix} \\ &= \left[\prod_m \frac{1}{\sqrt{1 + 2^{-2m}}}\right] [x \quad y] \prod_m \begin{bmatrix} 1 & \pm 2^{-m} \\ \mp 2^{-m} & 1 \end{bmatrix} \end{aligned} \tag{5.6}$$

$$\left[\prod_m \frac{1}{\sqrt{1 + 2^{-2m}}}\right] = Gain \tag{5.7}$$

The term represented in Equation 5.7 is simply a gain that changes based on the number of iterations, or bits, used in the calculation.

Execution of the algorithm requires adding or subtracting the angle of $\tan^{-1}(2^{-m})$, which is stored in a 16-word lookup table, to the desired angle which the calculation is being performed. The goal of this addition or subtraction is to bring the sum of the two angles closer to zero. Addition is performed on the next iteration of the algorithm if this sum is less than a value of zero, and subtraction is performed otherwise. Efficiency is achieved from the shift and add instructions implemented in this process which require only one instruction cycle each. After *m* iterations, the sine and cosine values of the desired angle are the y' and x' components of the vector $[x' \quad y']$, respectively [1, 25, 28-30].

For maximum accuracy of the calculations, the number of iterations is typically chosen based on the number of bits used to represent data in the processor. After *m* (*m* = number of microcontroller bits) shifts and adds, all bits of the data have been bit-shifted out from the CORDIC calculation, and no further accuracy is possible.

5.1.3 Simulation

To simulate the CORDIC algorithm, MATLAB code is written to implement the algorithm execution, including the adding and subtracting of the bit shifted values of x and y. This code can be found in Appendix B. Plotting the converging sine and cosine value of any angle versus the true (MATLAB's 64-bit) calculation of the same angle, it is clear that the CORDIC algorithm quickly and efficiently performs the calculation. Sample code executions are shown in Figures 5.2, 5.3, and 5.4 for the calculation of arbitrary angles 30º, -48.6º, and 84.65º, respectively.

Simulation is implemented with 15 bits, due to the on-chip implementation being executed on 1.15 fractional format values. The CORDIC gain at 15 bits (*m* = 15) is calculated in code to be

0.607252935385914, which is accurate to ten decimal places compared to the $m = \infty$ gain of 0.6072529350088813.

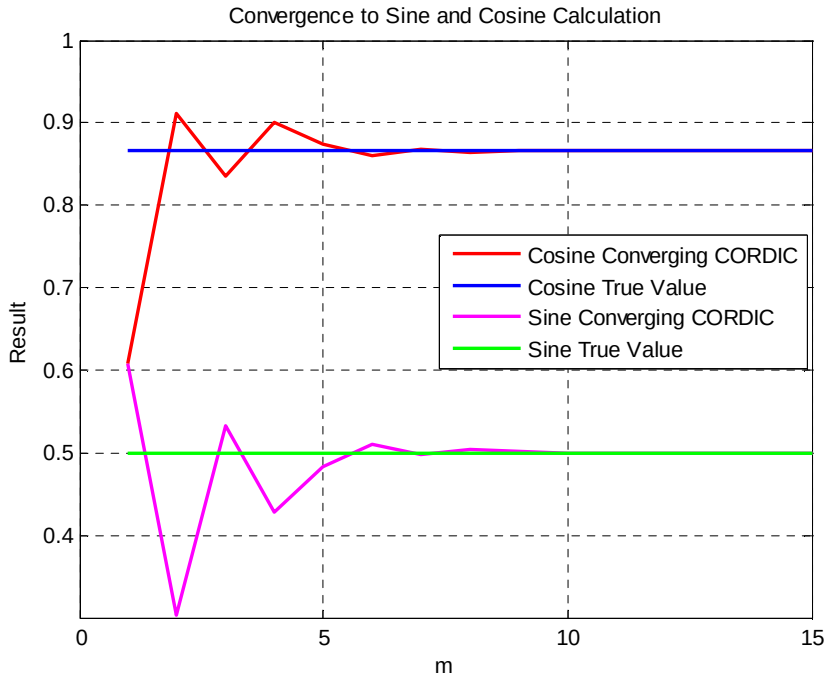


Figure 5.2: Converging CORDIC sine and cosine values for angle 30º.

CORDIC sin(30º) = 0.5000
CORDIC cos(30º) = 0.8660

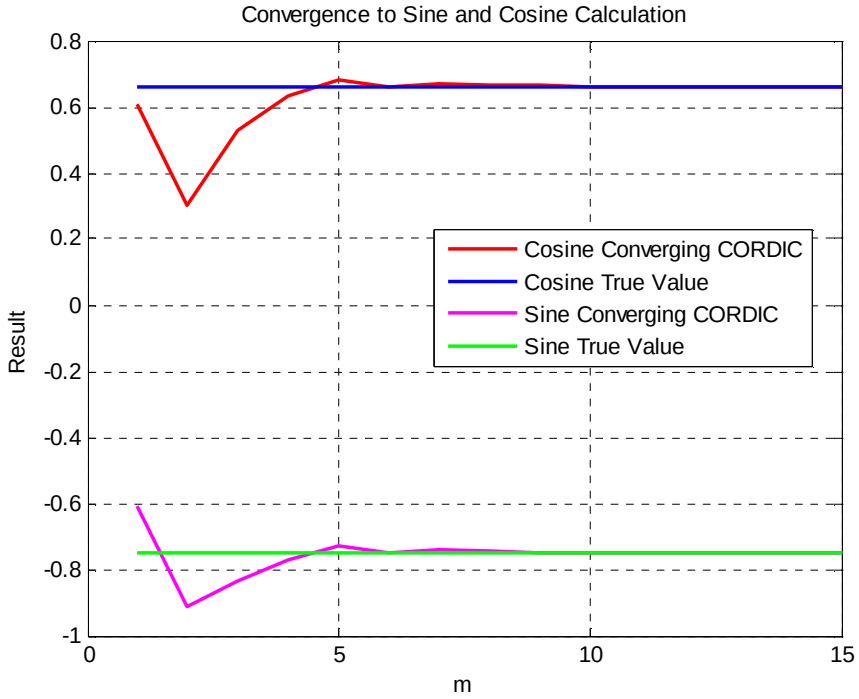


Figure 5.3: Converging CORDIC sine and cosine values for angle -48.6º.

CORDIC sin(-48.6º) = -0.7501
CORDIC cos(-48.6º) = 0.6613

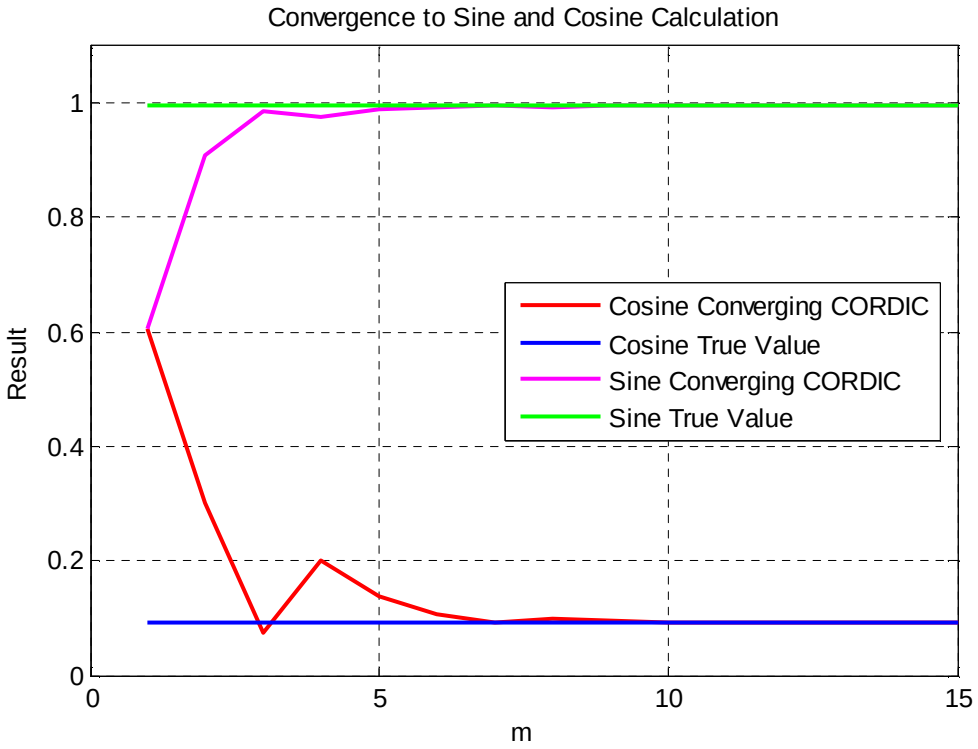


Figure 5.4: Converging CORDIC sine and cosine values for angle 84.65º.

CORDIC sin(84.65º) = 0.9956
CORDIC cos(84.65º) = 0.0932

5.1.4 Error and Resolution

As with any digital device with a finite number of operational bits, the error due to quantization becomes a factor in calculation. With the CORDIC algorithm, the number of iterations implemented in the trigonometric calculation determines the resolution.

For the purpose of this project, 12 bits are used to represent the angle of inclination. The data for inclination is obtained from the onboard ADC as the upper 12 bits of a signed fractional format. Thus, the resolution of the inclination angle for ±90º full scale range is 0.0439º. Hence, using 15 bits in the CORDIC calculation does not increase the resolution of the angle itself. Rather, the trigonometric sine and cosine calculations of the angle have 15 bit resolution based on the 12-bit angle.

To examine the resolution of an *m*-bit CORDIC calculation, a residual of the sine and cosine calculation for every angle (possible of representation with 16-bit resolution) from -90º to +90º

versus the 64-bit MATLAB calculation of that value is plotted. These residual plots can be seen below for 8-bit (Figure 5.5) and 15-bit (Figure 5.6) calculation. In addition, the standard deviation of the residuals at each number of bits is calculated and plotted, and can be seen in Figure 5.7.

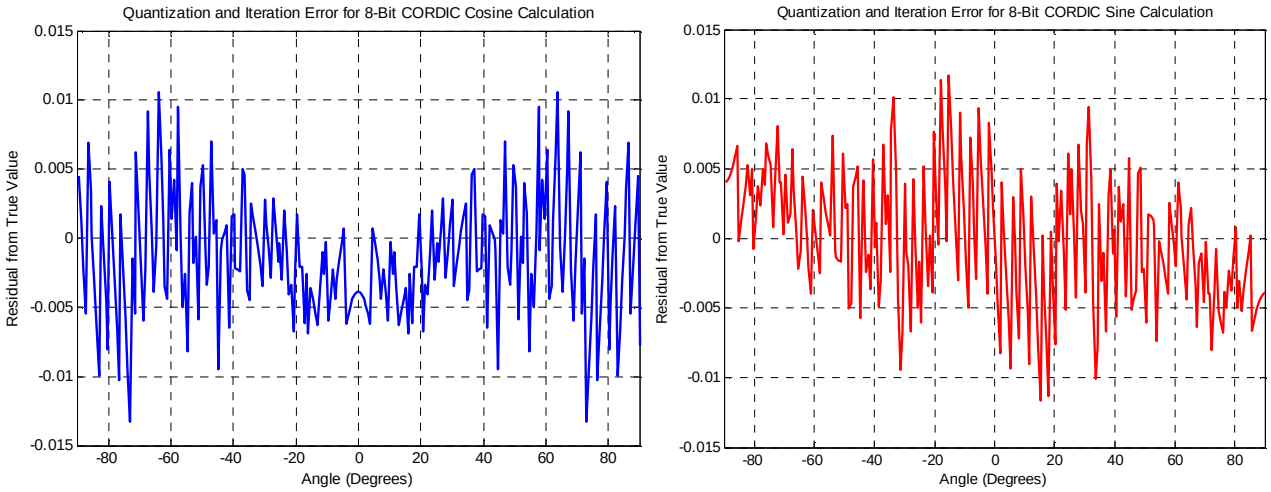


Figure 5.5: 8-bit CORDIC calculation residuals.

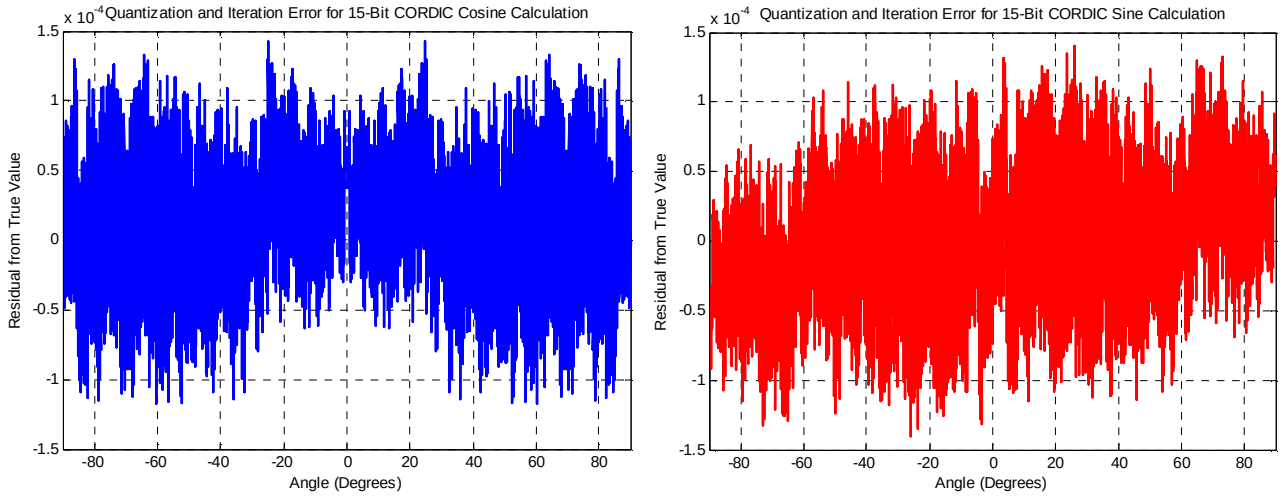


Figure 5.6: 15-bit CORDIC calculation residuals.

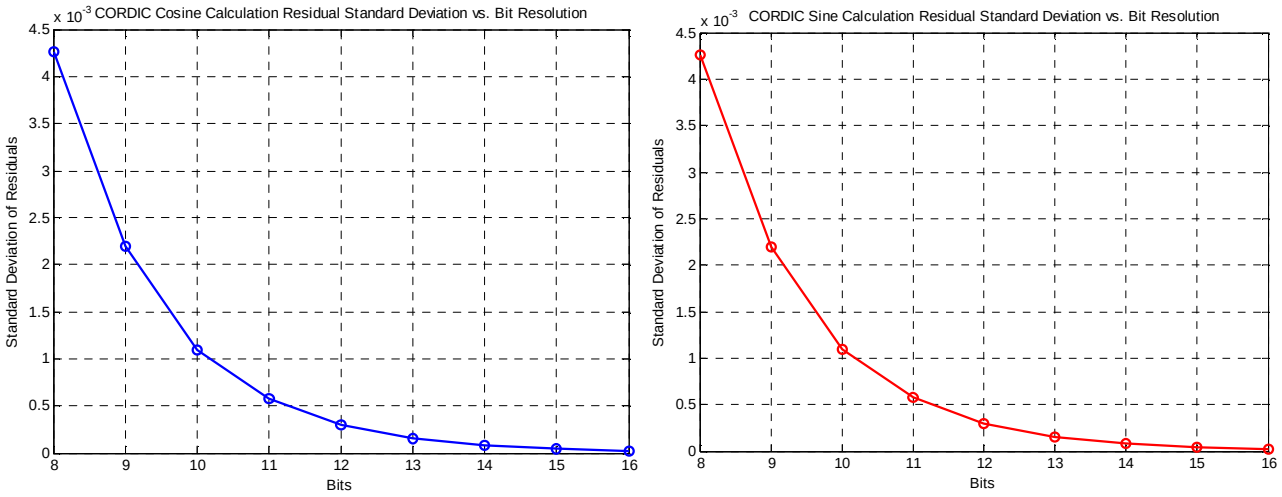


Figure 5.7: CORDIC-calculated vs. true MATLAB calculated residual standard deviation at 8 through 16 bits of resolution.

It is clear from all of the above figures that as the number of bits of resolution is increased, the calculation of the sine and cosine values resulting from the CORDIC algorithm approach the true unquantized value. From Figure 5.6, implementation of 15-bit CORDIC, as used in this project, results in values that fall within an envelope of 1.5×10^{-4} of the true value, which represents an angle value of less than 0.01º. Thus, any error on the order of 10^{-4} is considered sufficient in calculation.

5.1.5 Microcontroller Implementation and Results

Implementing the CORDIC algorithm on the dsPIC microcontroller requires efficient programming to reduce the number of instruction cycles required for calculation. In addition, mathematical calculations onboard the dsPIC are carried out as 1.15 signed fractional format to reduce conversion time by avoiding all possible floating point calculations. Converting the angle of inclination derived from Equation 3.1 from a floating point representation to a 16-bit fractional entails quantization error, and the same is true for the short 16 word deep arctangent lookup table required for the CORDIC algorithm.

The CORDIC function implemented onboard the magnetometer system can be found in Appendix C. For ease of execution, a unipolar calculation is made, meaning that all angles are converted to their absolute value and then the CORDIC algorithm is executed. Upon completion, the sign of the sine component of the vector is reestablished via a simple negation of the output value if necessary.

The CORDIC function receives the angle of inclination in degrees as a floating point number and converts it to a 1.15 fractional value representing a fraction of the 90º full scale range via the DSP library function `Float2Fract` [45]. Rounding of near-zero degree angles is required to avoid sign changes due to the quantized CORDIC lookup table. This rounding resulted in angles of less than 0.000061º in magnitude to be rounded to a magnitude of 0.005º.

Continuing, the CORDIC algorithm discussed in Section 5.2 is implemented, with simple shifts and addition or subtraction. These operations are executed on the appropriately-indexed values of the CORDIC lookup table according to the sign of the desired angle after each iteration.

After 15 iterations, the value of **x** is output as the cosine value of the angle calculation, and the value of **y** is negated if the input angle was negative, and then also output. These values are stored in a 1 x 2 vector that is utilized later in the magnetometer program to compensate for tilt.

Counting the number of microcontroller instructions required for the CORDIC algorithm to complete from the function call to the return value, it is found that 1684 cycles are required for both sine and cosine calculations combined. This number can be compared to the 5,489 instructions required for a both direct sine and cosine calculation using 32 bit floating point functions from the math library. This represents an execution time that is 3.25 times faster and utilizes only 30.6% of the instruction cycles. At 6.144 MIPS, the speed increase accounts for a time savings of 619 microseconds for every calculation made.

If a trigonometric lookup table is implemented for these same calculations, significant increase in memory requirements and instruction cycles are presented. Assuming a lookup table of 256 values for both sine and cosine, 512 bytes of memory are required. In addition, interpolation of these values for greater accuracy requires a significant increase in execution time due to the interpolation process. Using CORDIC, the single 16-word lookup table is implemented, using only 32 bytes of memory, or 6.25 % of that required by the interpolated lookup table method.

To test the resolution of the calculations, a small sample of angles is picked at random, and the results of the sine and cosine calculations versus the actual 64-bit calculation from MATLAB are compared.

Angle (degrees)	MATLAB Sine	dsPIC CORDIC Sine	Residual
35.8 º	0.5849576749	0.585025	6.73×10^{-5}
-24.7 º	-0.4178670738	-0.417907	3.99×10^{-5}
0.09 º	0.0015707956	0.001617	4.63×10^{-5}
-76 º	-0.9702957262	0.970153	1.42×10^{-4}
45 º	0.7071067811	0.707275	2.68×10^{-4}

Table 5.1: Residual calculation of dsPIC onboard sine calculations vs. 64-bit MATLAB calculations.

Angle (degrees)	MATLAB Cosine	dsPIC CORDIC Cosine	Residual
35.8 º	0.8110638189	0.810882	1.82×10^{-4}
-24.7 º	0.9085081775	0.908447	6.12×10^{-5}
0.09 º	0.9999987663	0.999939	5.97×10^{-5}
-76 º	0.2419218956	0.241760	1.62×10^{-4}
45 º	0.7071067811	0.706817	2.89×10^{-4}

Table 5.2: Residual calculation of dsPIC onboard cosine calculations vs. 64-bit MATLAB calculations.

From the above tables, it is clear that the CORDIC-calculated trigonometric conversions fall within the error envelope that is calculated in Section 5.4. The largest residual encountered in the random sampling of angle values is 2.89×10^{-4}, which when back-converted to an angle in degrees, represents an angle of 0.02º. This error falls below the 12-bit resolution of the 12-bit angle sample resolution of 0.044º. Thus, the error of the CORDIC calculations is negligible.

5.2 Trigonometric Identity Calculation

To determine if CORDIC is a suitable replacement for traditional methods of conversion using trigonometric identity, testing of execution time of this method is required.

From Equation 3.1, it can be determined that:

$$\sin(\text{Angle}_{\text{OUT}}) = \frac{\text{V}_{\text{OUT}} - \text{V}_{\text{OFFSET}}}{2\ \text{V/g}} \tag{5.8}$$

From the trigonometric identity presented in Equation 5.9 below, the following conversion can be made:

$$\sin(\Theta)^2 + \cos(\Theta)^2 = 1 \tag{5.9}$$

$$\cos(\Theta) = \sqrt{1 - \sin(\Theta)^2} \tag{5.10}$$

$$\cos(\text{Angle}_{\text{OUT}}) = \sqrt{1 - \left(\frac{\text{V}_{\text{OUT}} - \text{V}_{\text{OFFSET}}}{2}\right)^2} \tag{5.11}$$

Hence, calculation of the sine and cosine of the inclination angle requires no arcsine calculation. By squaring the inclination term and calculating a square root, the output sine and cosine values result. The function `_Trig_Trad`, which executes this calculation, is shown in Appendix C.

To implement the squaring of the inclination term (represented in fractional format), the built-in function `VectorScale` [45] is utilized. Shown in Equation 5.11, this value is converted to a floating point number, which is then subtracted from one. The square root of this resulting value is then calculated using floating point math, and the output is converted back to fractional format for output as the cosine term of the output angle.

Timing the function, it is found that 2,132 instruction cycles are required for the conversion and mathematics for each tilt angle. Thus, for both functions to execute, a total of 4,264 instructions are required. Despite using more instructions than the CORDIC algorithm, this process speeds up the total processing time of the system, in that the arcsine calculation and floating point mathematics required to convert the inclination reading to degrees in CORDIC are no longer required. This is covered in Section 5.3.

5.3 Discussion

By implementing FIR and median filters of minimum order in full magnetometer programming, comparison of the traditional floating point and CORDIC calculation processes reveals the advantages and disadvantages of each for application in this project. The free-running execution speed of the algorithms was tested to determine the maximum sampling rate, half of which is the maximum system bandwidth.

Method	Maximum Sampling Rate	Maximum System Bandwidth	Advantages	Disadvantages
Trigonometric Identity	820 sps	410 Hz	No arcsine calculation. No floating point preparation.	Square root calculation in floating point consumes instruction cycles.
CORDIC	692 sps	346 Hz	No floating point math in CORDIC function. Very efficient if angle already represented by degrees or fractional.	Arcsine calculation required prior to call. Arcsine requires floating point input.

Table 5.3: Comparison of CORDIC and traditional floating point trigonometric calculation.

Using CORDIC, floating point and an arcsine calculation are implemented in preparation for implementation of the CORDIC function itself. Thus despite speed of the CORDIC function being far superior to floating point or identity methods, the preparation of the necessary floating point values requires more time. Using this method, the maximum sampling frequency when

integrating all aspects of the tilt-compensated magnetometer system (discussed in Chapter 7) is found to be 692 samples per second, resulting in a maximum system bandwidth of 346 Hz.

When implementing the traditional method of sine and cosine calculation using the identity method (Equations 5.9 through 5.11), the floating point preparation of the inclination angle is not required, yet a floating point calculation is necessary for the square-root function. Within the complete system program, timing of the tilt-compensated algorithm using this method of trigonometric conversion yielded a maximum sample rate of 820 samples per second, resulting in a system bandwidth of 410 Hz.

It can be noted that for this project, CORDIC is a slower method for performing the magnetometer tasks, due to the floating point arcsine function used to calculate the angle output in degrees. This arcsine calculation is required for accurate inclination angle from the inclinometers [7]. However, it is possible for this arcsine calculation to be ignored and straight-line calculation of the output angle can be utilized, allowing for the output angle to be represented in fractional format. Using this method, significant reduction in execution time can be realized, thus drastically increasing system bandwidth. It is important to note, however, that with the straight line method, error increases with subsequent increase in inclination angle [7]. For example, at 45º of inclination (step response angle), a 10.2% error (4.6º) results, and at 70º of inclination, a 23.4% error (16.3º) is seen. For this project, and the potentially-large inclination angles that are tested, this amount of error is to be considered unacceptable, and this method is not utilized. Alternatively, a lookup table can be utilized to interpolate the value of the output angle, reducing execution time.

Both trigonometric methods discussed in this chapter proved successful at executing trigonometric functions, with each resulting in system bandwidths that are more than sufficient for the signals of interest. If further improvement in execution time and bandwidth are required in the future, further improvement in software is possible. This is discussed further in Section 8.2 (Future Work).

Chapter 6 Tilt Compensation

In this chapter we discuss the implementation of the filtered signals and subsequent calculated trigonometric values, discussed in previous chapters, being integrated in tilt compensation. Trigonometric equations that use vector mathematics are discussed and tested in the magnetometer system. In addition, a calibration routine is identified using simulation and testing of single and double-axis motion. Software integration of tilt and calibration data is implemented using DSP functions for fast and efficient execution.

6.1 Trigonometric Equations

The magnetic field vectors detected by each respective sensor in any one axis are found arithmetically by first using the roll-compensation equations seen in 6.1. These results are then transformed back to the horizontal plane with the pitch compensation equations seen in Equation set 6.2 (with roll angle Θ and pitch angle φ) [22]. The equations reflect the tilt convention discussed in section 3.2.2, also displayed in Figure 6.1. B_X, B_Y and B_Z represent the magnetic fields detected in the X, Y and Z axes of the magnetometer, and B_{Xroll}, B_{Yroll} and B_{Zroll} represent the roll-compensated output of the system in the X, Y and Z axes, respectively. $B_X{}'$, $B_Y{}'$ and $B_Z{}'$ then represent the full tilt-compensated output of the system in each axis direction, respectively.

$$\begin{aligned} B_{Xroll} &= B_X \\ B_{Yroll} &= B_Y \cos(\Theta) + B_Z \sin(\Theta) \\ B_{Zroll} &= -B_Y \sin(\Theta) + B_Z \cos(\Theta) \end{aligned} \tag{6.1}$$

$$\begin{aligned} B_X{}' &= B_{Xroll} \cos(\varphi) - B_{Zroll} \sin(\varphi) \\ B_Y{}' &= B_{Yroll} \\ B_Z{}' &= B_{Xroll} \sin(\varphi) + B_{Zroll} \cos(\varphi) \end{aligned} \tag{6.2}$$

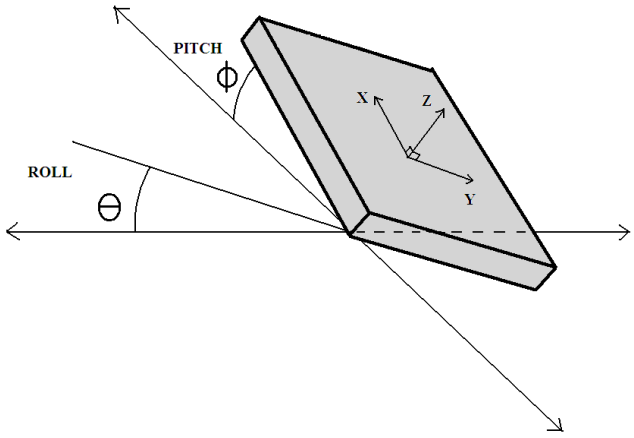


Figure 6.1: Axis tilt angle convention.

Combining the equations, the complete compensation equations that transform the tilted magnetic field readings back to the readings in the horizontal plane are seen in Equation set 6.3 below.

$$\begin{aligned} B_X' &= B_X \cos(\varphi) + B_Y \sin(\varphi) \sin(\Theta) - B_Z \sin(\varphi) \cos(\Theta) \\ B_Y' &= B_Y \cos(\Theta) + B_Z \sin(\Theta) \\ B_Z' &= B_X \sin(\varphi) - B_Y \sin(\Theta) \cos(\varphi) + B_Z \cos(\Theta) \cos(\varphi) \end{aligned} \tag{6.3}$$

From the equations, it is shown that if either angle, φ or Θ, posseses a value of zero (horizontal to gravity vector), one of the horizontal axes is unaffected by the tilt angle. If roll angle Θ is zero, the simultaneous pitch tilt does not affect the output of the Y axis. If pitch angle φ is zero, the simultaneous roll tilt does not affect the output of the X axis.

6.2 Initial System Integration

Integrating the tilt-compensation equations in software is accomplished by a single function called `_Tilt_Comp`, which can be seen in Appendix C.

The function is passed pointers to the start address of four vectors (represented in fractional format) stored in data memory onboard the dsPIC. The four vectors' dimensions and contents are seen in Table 6.1 below.

Table 6.1: Vectors passed to tilt-compensation function.

Vector Name	Dimension	Contents
`H_raw_xyz`	1 x 3	Raw (uncompensated) 3-axis magnetic readings
`H_comp_xyz`	1 x 3	Compensated 3-axis magnetic readings
`roll_sin_cos`	1 x 2	Sine and cosine result of roll angle, respectively
`pitch_sin_cos`	1 x 2	Sine and cosine result of pitch angle, respectively

After calling the function, the respective roll and pitch sine and cosine values are loaded into vectors that act as buffers for the vector operations that will soon follow. The tilt-compensation calculation is accomplished by implementing the built-in DSP functions `VectorMultiply` and `VectorDotProduct`, from Microchip technologies [45]. These functions utilize the MAC and MPY commands of the dsPIC core to quickly perform arithmetic vector operations. `VectorMultiply` [45] multiplies each element of a vector with the each similarly-indexed element of another vector. `VectorDotProduct` multiplies each element in the same fashion, but sums up the results of all multiply operations and outputs a single fractional value. By utilizing these functions with both the raw magnetic field data vector and the preloaded buffers containing sine and cosine calculation of the tilt angles, the X, Y and Z-axis results of Equation set 6.3 are calculated and stored in the vector `_H_comp_xyz`.

A sample test of the tilt-compensated magnetometer system is shown in Figure 6.2 below, where the platform is first subject to a positive roll angle motion, and then a positive pitch angle motion. From the figure, evidence of tilt-compensation is apparent, as the compensated magnetic readings are significantly more stable than the raw readings from the three individual axes. This is the desired result, as the field being detected is the steady earth field. Ideally, the tilt-compensated output of each axis in a steady earth field will be maximally flat, or possess zero standard deviation.

Closer inspection of the compensated readings show drift that is linearly related to the motion presented to the tilt platform in the X, Y and Z axes. In subsequent tests of the magnetometer system with both positive and negative pitch and roll angles presented in different chronological order, the drift changes magnitude and sign, relative to the original, uncompensated magnetic readings. Therefore, a correction, or calibration of the magnetometer system is required.

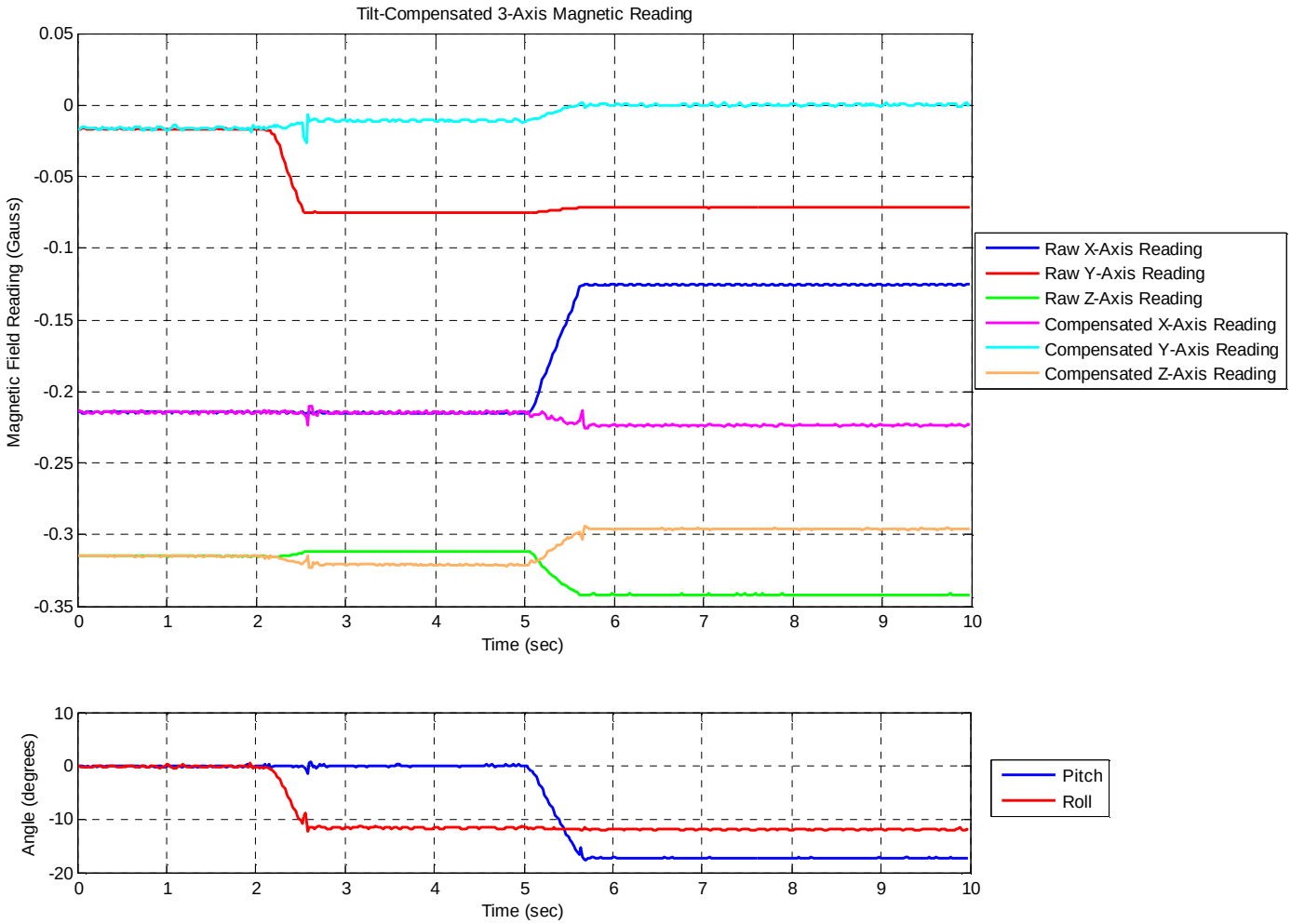


Figure 6.2: Output of tilt-compensated magnetometer system with roll and pitch angle tilt.

6.3 System Calibration

To calibrate the magnetometer system, it is necessary to first identify the possible sources of the interference, or drift, of the compensated output. The possible sources of the drifting compensated values are:

1. Magnetic interference (magnetic objects, stray magnetic fields, etc)
2. Misaligned inclinometer or magnetic sensors.
3. Sensor error (amplifier, sensor)

To isolate and identify the source of error, these factors are isolated and tested with calibration programs and alterations. To begin, the possible magnetic interference from outside magnetic sources is reduced by isolating the magnetometer system to a location lacking any man-made structures such as buildings, power lines, sewers, or metallic objects. Several 2-axis motion tests are carried out in this setting. One test can be seen below in Figure 6.3. All tests and resultant plots seen in this chapter reflect no signal filtering.

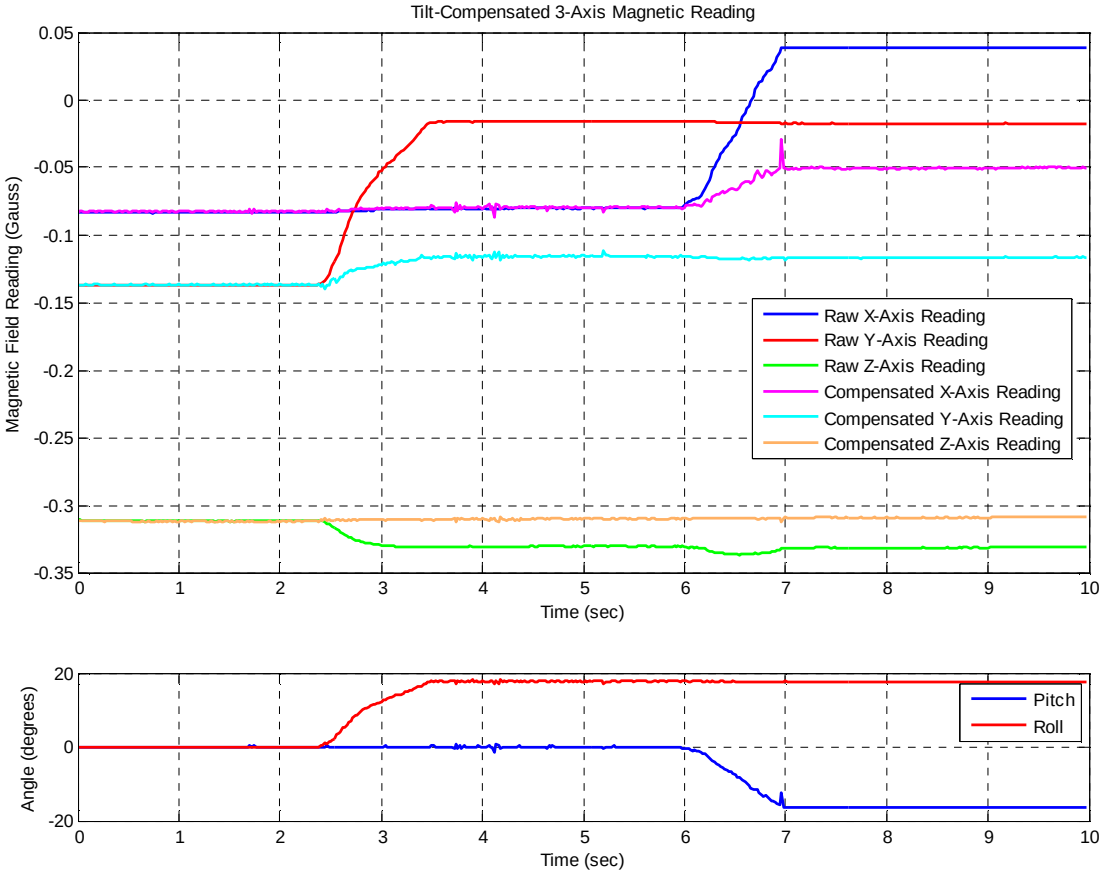


Figure 6.3: Output of tilt-compensated magnetometer system with roll and pitch angle tilt in an environment with minimal external interference.

From the figure, we seen that error in the Z-axis (vertical) direction has been reduced (compared to Figure 6.2), although the drift in the X and Y-axis readings is still present. Through all tests in this

environment, this reduction in Z-axis interference is reflected, and thus conclusions can be made. The change in the Z-axis reading reflects the removal of a field gradient in the vertical direction, likely the result of taking the initial readings in a building that was constructed with metallic roofing and steel-framed drop ceilings. Alteration of the direction of the magnetic field into these magnetic materials (discussed in Section 2.4) resulted in the interference seen in Figure 6.1, which is also reflected in the other two orthogonal axes when tilted.

The possibility of error in the output of the inclination sensors can be eliminated by observation of Figure 6.3 (also seen in Figure 6.8) and Table 6.5. With inclination angle error, deviation from a steady field reading can be expected in the vertical direction (Z-axis), by Equation set 6.3. From observation of the figures and table, the standard deviation of the Z axis is significantly less than the other two axes. Therefore, it can be concluded that error in the inclination angle is not the cause of the drift in compensated output.

The next test to determine the source of the interference analyzes the alignment of the inclinometers relative to the tilt platform. If the sensors are not aligned properly, significant output error can result in compensation with the error in inclination signal. To test if the inclinometers are improperly, a one-axis tilt is performed in the positive and negative direction in both roll and tilt. The output of the magnetometer system, including the compensated field readings is then plotted. If either sensor is out of alignment, the compensated output of the unaffected (X-axis in roll-only test, and Y-axis in pitch-only test) axis will deviate from the raw magnetic reading. If the inclinometers are, in fact, properly aligned on the PCB and tilt platform, the compensated field readings of the unaffected axes will stay constant when tilted. The plots of positive roll and positive pitch are plotted in Figures 6.4 and 6.5, respectively.

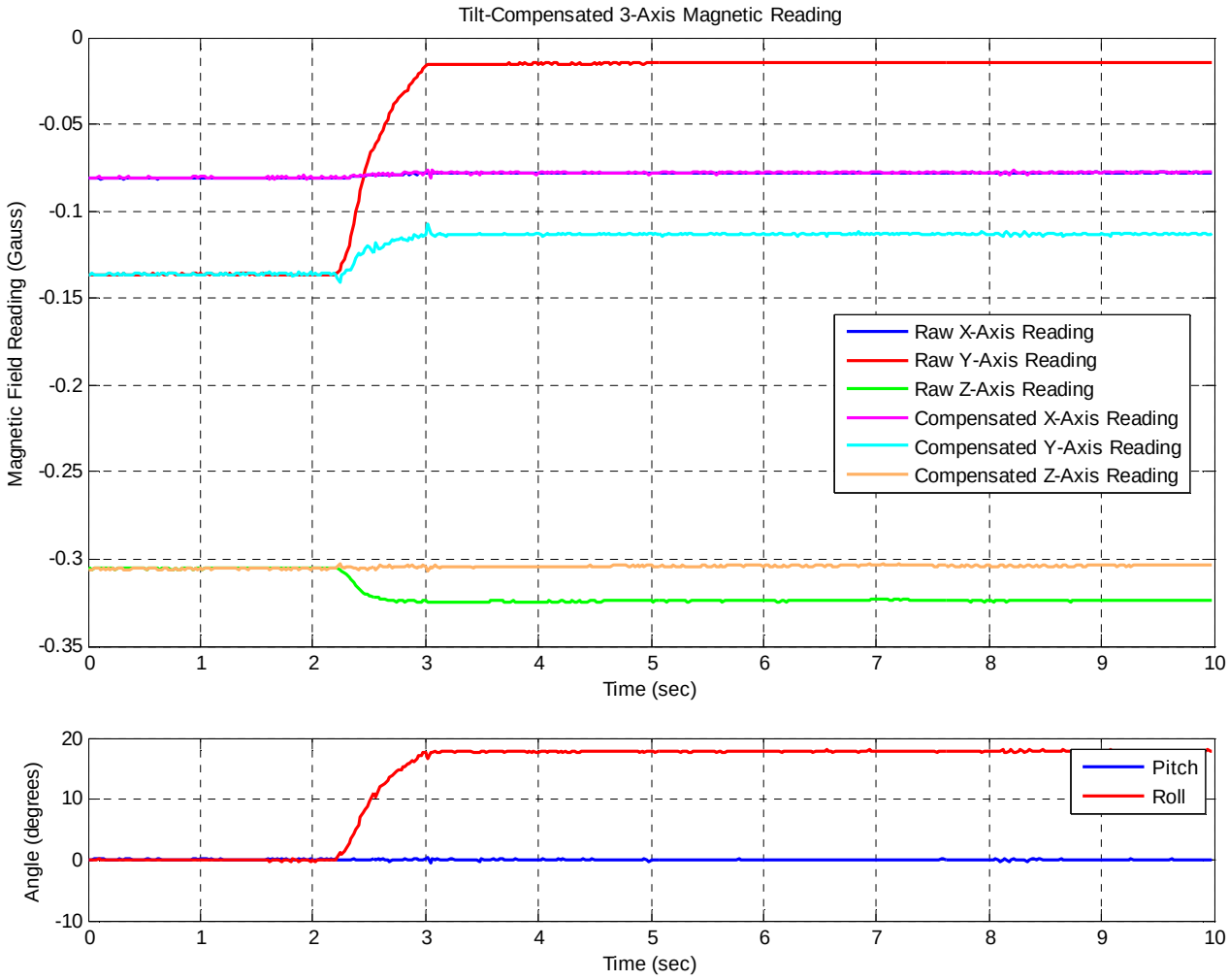


Figure 6.4: Three-axis reading and output with isolated roll tilt.

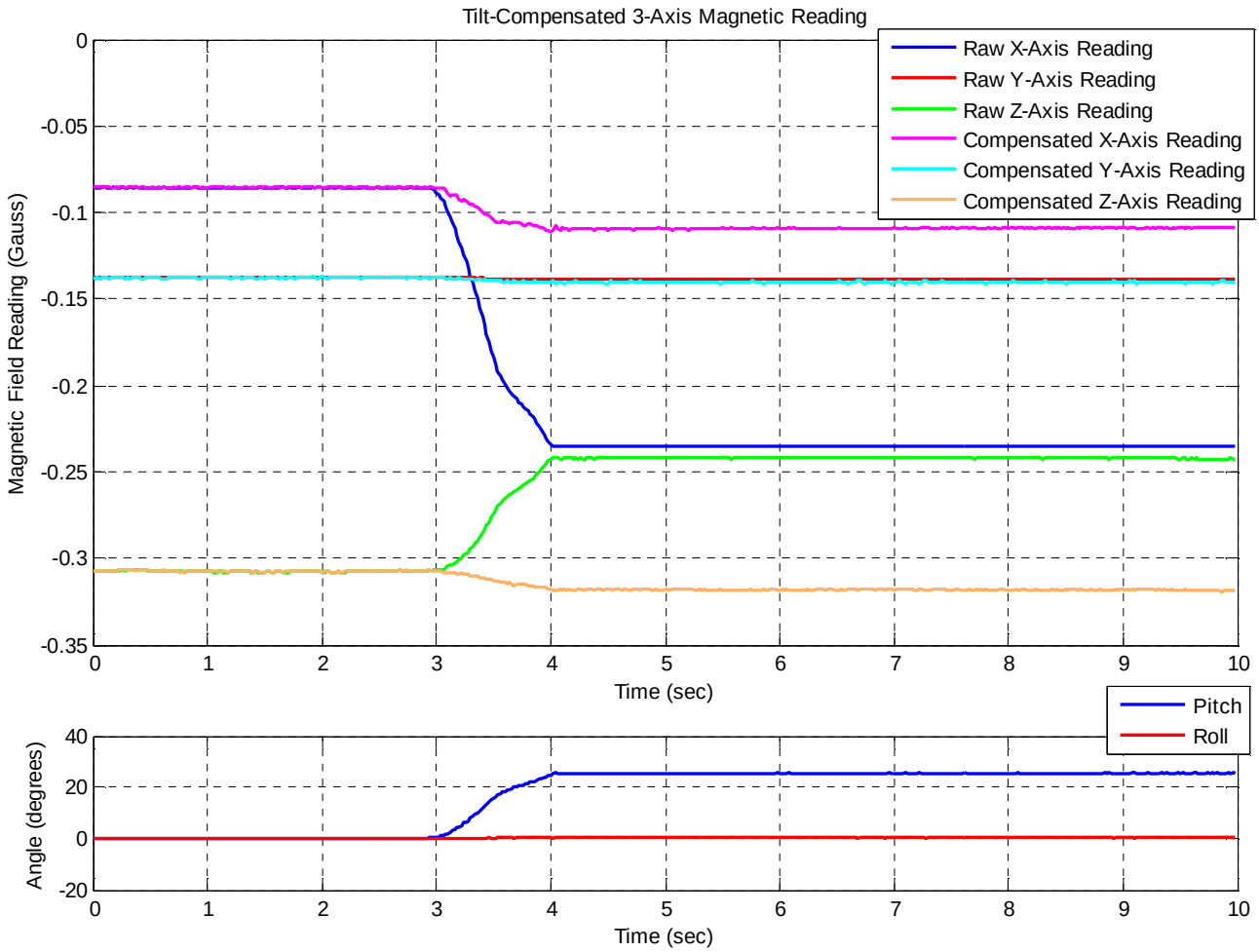


Figure 6.5: Three-axis reading and output with isolated pitch tilt.

Seen in the X-axis outputs of Figure 6.4 and the Y-axis output of Figure 6.5, the raw magnetometer reading does not diverge from the compensated output value, and therefore, the inclinometers and magnetometers are properly aligned with both one another as well as the platform itself. In addition, the error of the inclinometers is documented as +/-0.5% of the output value. In the extreme case of a 90º tilt, the error is maximized and calculated at 0.45º. Using Figure 6.5 above, this error represents approximately 62 microGauss error in the X axis output, and 2 milliGauss in the Z axis output. This error is significantly less than the deviation seen in the above figures. Therefore, it is concluded that sensor alignment is not the source of the output deviation.

Eliminating the possibility of inclinometer output error is the compensated Z-axis plot of Figure 6.3. This compensated output considers both the roll and pitch angles in calculation (See Equation Set 6.3). With the standard deviation of this signal being -29.2 dB, significantly less than the other two axes (see Table 6.5), it can be determined that inclinometer output is not responsible for the output error. This conclusion is supported by manual tuning of both inclination angles, resulting in no notable improvement in deviation.

Hence, the remaining deviation in the compensated output of the magnetometer is due to an error in magnetometer output or interfacing electronics. This error could be from sensor inaccuracies, amplifier or factory calibration error, or error in the analog electronics of the system.

Simple weighting of the magnetic readings (thus altering the compensated output) is eliminated as a solution to the deviation drift, as weighting any one of the axis readings will weight that axis' own compensated zero angle output. This creates a deviation from the raw magnetic field readings and the compensated magnetic field output will exist in the zero angle (initial) condition of the magnetometer system. It is therefore concluded that in the compensated output of the system, the change in the magnitude of the reading of one axis is adversely affecting tilt-compensated outputs.

To identify this relationship within the system, two calibration routines are designed that interface one-axis tilt readings like those in Figures 6.4 and 6.5. These MATLAB programs, designed for roll and tilt angle calibration, are seen in Appendix B and are called `Calibrate_Roll.m` and `Calibrate_Pitch.m`.

To interface these programs, a simple spreadsheet in Microsoft Excel is designed. The spreadsheet accepts a 500-sample long reading of a one-axis tilt run of the magnetometer. The spreadsheet incorporates eight pieces of data in eight columns, including the raw X, Y and Z axis readings, the compensated X, Y and Z axis readings, as well as the roll and pitch angle, all converted to units of Gauss and degrees, respectively. After conversion, the data can be cut from Excel and pasted into the data matrix that is created in the calibration M-file.

In a one axis tilt situation, the X and Y axes cannot affect the compensated output of the other, as one of the axis readings is not changing with either tilt scenario. Therefore, the relationship between the X and Z axis, as well as the Y and Z axis, are explored in the one-axis calibration programs.

The equations being explored by the calibration M-file for a roll-only scenario are as follows, including the calculated change in reading magnitude from zero-angle state, ΔB:

$$\begin{aligned} B_{Xroll} &= B_X \\ B_{Yroll} &= B_Y \cos(\Theta) + B_Z \sin(\Theta) + \Delta B_Z \\ B_{Zroll} &= -B_Y \sin(\Theta) + B_Z \cos(\Theta) + \Delta B_Z \end{aligned} \tag{6.4}$$

$$\begin{aligned} B_{Xroll} &= B_X \\ B_{Yroll} &= B_Y \cos(\Theta) + B_Z \sin(\Theta) + \Delta B_Y \\ B_{Zroll} &= -B_Y \sin(\Theta) + B_Z \cos(\Theta) + \Delta B_Y \end{aligned} \tag{6.5}$$

In Equation set 6.4, the net change in the output of the raw Z-axis reading, ΔB_Z, is assumed to affect both the Z-axis, and the Y-axis, as described above. The equations exploring the Y-axis change affecting both the Y and Z axes in the same way is as seen in Equation set 6.5.

It is seen from the figures presented in this chapter thus far that only a small percentage of the full scale net change of each axis is affecting the opposing axis. Therefore, coefficients must be included in the calibration equations, as in 6.6 and 6.7 below.

$$\begin{aligned} B_{Xroll} &= B_X \\ B_{Yroll} &= B_Y \cos(\Theta) + B_Z \sin(\Theta) + \rho_{YdZ}\Delta B_Z \\ B_{Zroll} &= -B_Y \sin(\Theta) + B_Z \cos(\Theta) + \rho_{ZdZ}\Delta B_Z \end{aligned} \tag{6.6}$$

$$\begin{aligned} B_{Xroll} &= B_X \\ B_{Yroll} &= B_Y \cos(\Theta) + B_Z \sin(\Theta) + \rho_{YdY}\Delta B_Y \\ B_{Zroll} &= -B_Y \sin(\Theta) + B_Z \cos(\Theta) + \rho_{ZdY}\Delta B_Y \end{aligned} \tag{6.7}$$

In these equations, the ρ coefficients are denoted by the axis they are altering and the axis whose net change is being scaled. For example, if the Y axis is being affected by a net change in magnitude of the Z-axis reading, the coefficient of the corresponding ΔB_Z is denoted as ρ_{YdZ}. If the error is linear in nature, the values of the ρ coefficients will be similar and repeatable in tests of both positive and negative angle excitation.

The calibration routine of the M-file considers the raw magnetic field and angle readings of a one-axis test, and alters and identifies the ρ coefficients that produce the most similar tilt-compensated steady-state result in the initial zero-angle and final tilted readings of the test. The test is performed four times for both roll and pitch, with positive and negative tilts of each axis while considering the ΔB_X, ΔB_Y, or ΔB_z components.

The pitch equations considered in this test (pitch equivalents of equations 6.6 and 6.7) are as follows:

$$\begin{aligned} B_{Xpitch} &= B_X \cos(\varphi) - B_Z \sin(\varphi) + \rho_{XdZ}\Delta B_Z \\ B_{Ypitch} &= B_Y \\ B_{Zpitch} &= B_X \sin(\varphi) + B_Z \cos(\varphi) + \rho_{ZdZ}\Delta B_Z \end{aligned} \tag{6.8}$$

$$
\begin{aligned}
B_{Xpitch} &= B_X \cos(\varphi) - B_Z \sin(\varphi) + \rho_{XdX} \Delta B_X \\
B_{Ypitch} &= B_Y \\
B_{Zpitch} &= B_X \sin(\varphi) + B_Z \cos(\varphi) \ + \rho_{ZdX} \Delta B_X
\end{aligned} \tag{6.9}
$$

Tables 6.2 and 6.3 display the calculated ρ coefficients for the roll and pitch calibration, respectively. The tests are carried out with inclination angles of approximately 20º.

Table 6.2: Roll angle coefficient calibration test results.

ρ	Positive Roll Angle	Negative Roll Angle
ρ_{YdZ} (from 6.6)	1.00	0.31
ρ_{ZdZ} (from 6.6)	0.11	0.08
ρ_{YdY} (from 6.7)	**-0.19**	**-0.17**
ρ_{ZdY} (from 6.7)	**-0.03**	**-0.04**

Table 6.3: Pitch angle coefficient calibration test results.

ρ	Positive Pitch Angle	Negative Pitch Angle
ρ_{XdZ} (from 6.8)	0.35	-1.00
ρ_{ZdZ} (from 6.6)	0.18	-0.56
ρ_{XdX} (from 6.9)	**-0.15**	**-0.14**
ρ_{ZdX} (from 6.9)	**-0.05**	**-0.04**

Examination of the tables shows a correlation in the ρ coefficients when the net change in the X-axis readings and Y-axis readings alters the Z-axis compensation. This coefficient relationship is maintained, regardless of the sign of the pitch and roll angle, and inspection of the other test conditions reveals no such correlation. This relationship concludes that a change in either the X or Y axes results in a linearly proportional change in both the compensation of its own output, as well as the compensation calculation of the Z-axis output.

Plotting the result of the calibration including $\rho_{XdX} = -0.145$ and ρ_{ZdX} = -0.045 (average of the positive and negative coefficients), Figure 6.6 is the calibrated result of the same isolated positive pitch test displayed in Figure 6.5.

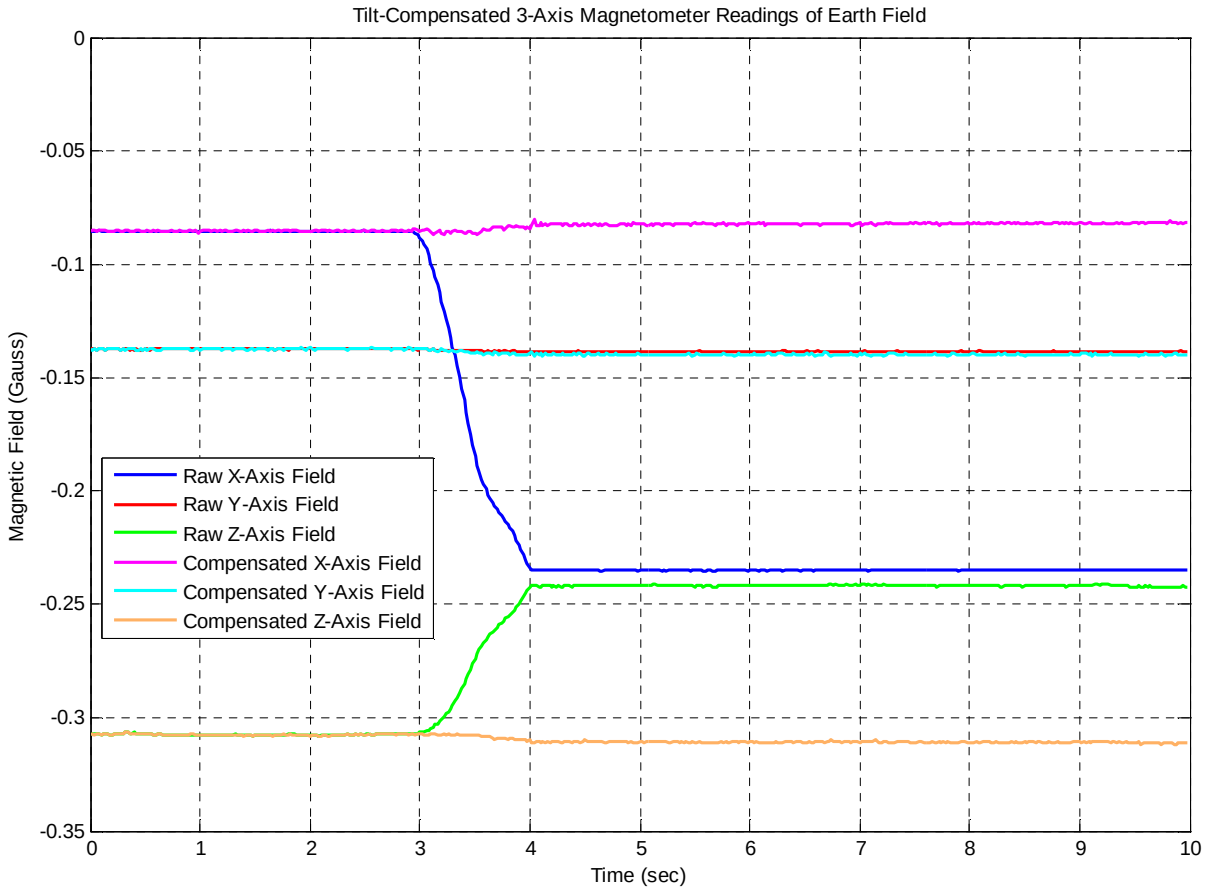


Figure 6.6: Test of calibration coefficients with positive pitch-isolated motion.

Resulting from the calibration, the compensated readings of the test clearly show improvement over the non-calibrated test. Including the ρ_{YdY} and ρ_{ZdY} in roll tests can show similar results.

Despite the improvement, there remains a small discrepancy of about 2.5 mGauss in the output of the Z axis in Figure 6.6. The signal calibration improvements are consistent in all tests, and small errors in the Z-axis are consistently reflected in these tests. This error is likely caused by either minute field irregularities or inaccurate calibration coefficients, as the coefficients calculated and implemented here were derived from a solitary one-axis test. From testing, repeated calibration tests with a mean-calculated calibration coefficient results in a more accurate calibration of the system.

Altering the position of the magnetometer system in the earth field being tested (turning the platform to another position) results in coefficient deviation (+/-0.05 or less from typical observed values), indicating that the error is rooted in either the magnetometer module or unforeseen magnetic field irregularities.

Assuming the magnetic fields incident on the magnetometer system are uniform in direction (dipole field), the root of the calibration coefficient values is theorized to be related to factory calibration of the magnetometer module. Similar to Figure 6.7, the magnetometer is factory calibrated using trim resistors that are connected in parallel to the Wheatstone bridge of the sensors themselves. The output of the bridge is then amplified with an active lowpass filter, presenting a 36 dB gain. With the trim resistors used on the module possessing a small tolerance, possible errors in calibration result. This error is subsequently amplified and output. Hence, after factory calibration, small errors still exists.

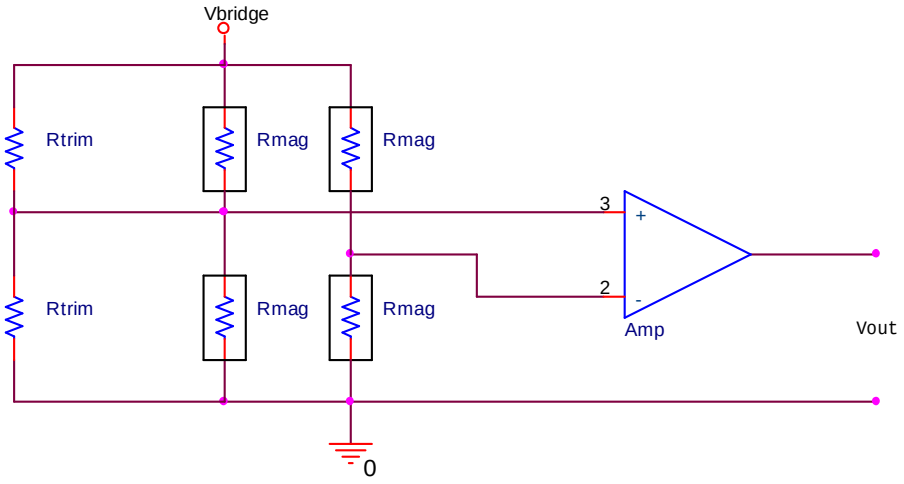


Figure 6.7: Trimming and amplification of magnetic sensor output.

Therefore, the calibration coefficients presented in this chapter represent a fine-tuning of the factory calibration of the magnetometer system, due this component tolerance. This fact is supported by the magnitude of the coefficients, in that the ρXdX and ρYdY coefficients are similar in magnitude and relatively larger than their ρZdX and ρZdY counterparts. With the X and Y magnetometer axes constructed in the same integrated circuit (IC), their calibration values are expected to be very similar, with the Z axis (on separate IC) being dissimilar. Further testing and investigation of the relationship between calibration coefficients and the factory testing can confirm this relationship. The remaining fluctuation in the calibration coefficients with varying orientation, mentioned above, is therefore assumed to be a result of non-uniformities in the surrounding magnetic field.

The accuracy of the calibration coefficients can be improved with several calibration tests resulting in a mean calibration coefficient. However, for the purpose of this project, a single roll and pitch calibration coefficient calculation prior to any set of tests will suffice.

By integrating the Equation sets 6.7 and 6.9, transformation of the tilted magnetometer readings are converted to the compensated horizontal plane with Equation set 6.10 below with roll angle Θ and pitch angle φ.

$$\begin{aligned} B_X' &= B_X\cos(\varphi) + B_Y\sin(\varphi)\sin(\Theta) - B_Z\sin(\varphi)\cos(\Theta) - \rho_{XdY}\Delta B_Y\sin(\varphi) + \rho_{XdX}\Delta B_X \\ B_Y' &= B_Y\cos(\Theta) + B_Z\sin(\Theta) + \rho_{YdY}\Delta B_Y \\ B_Z' &= B_X\sin(\varphi) - B_Y\sin(\Theta)\cos(\varphi) + B_Z\cos(\Theta)\cos(\varphi) + \rho_{ZdY}\Delta B_Y\cos(\varphi) + \rho_{XdX}\Delta B_X \end{aligned} \tag{6.10}$$

An uncalibrated test with both pitch and roll motion can be seen in Figure 6.8. The same test incorporating the calibration coefficients calculated above is seen in Figure 6.9 below.

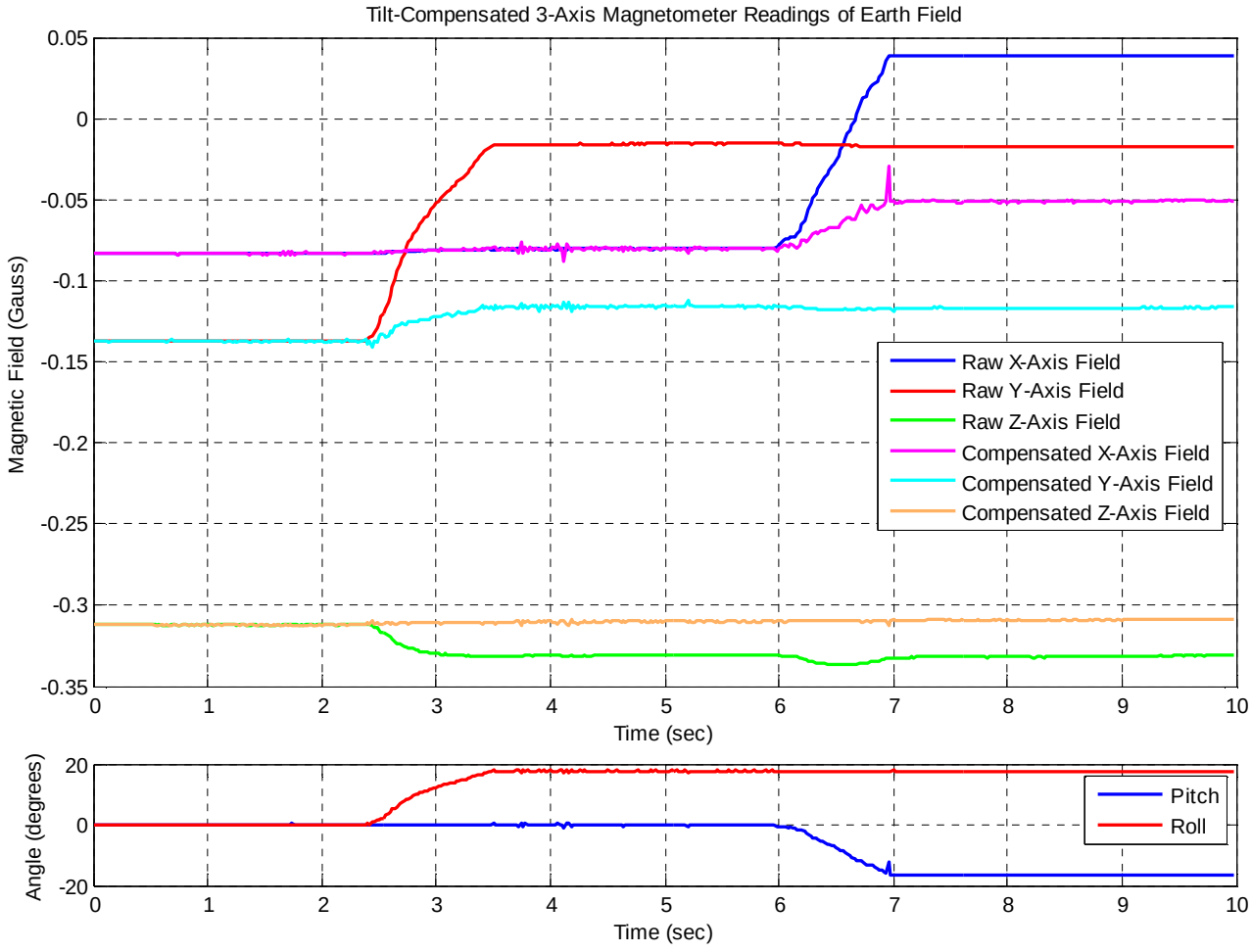


Figure 6.8: Non-calibrated 2-axis compensated earth field reading.

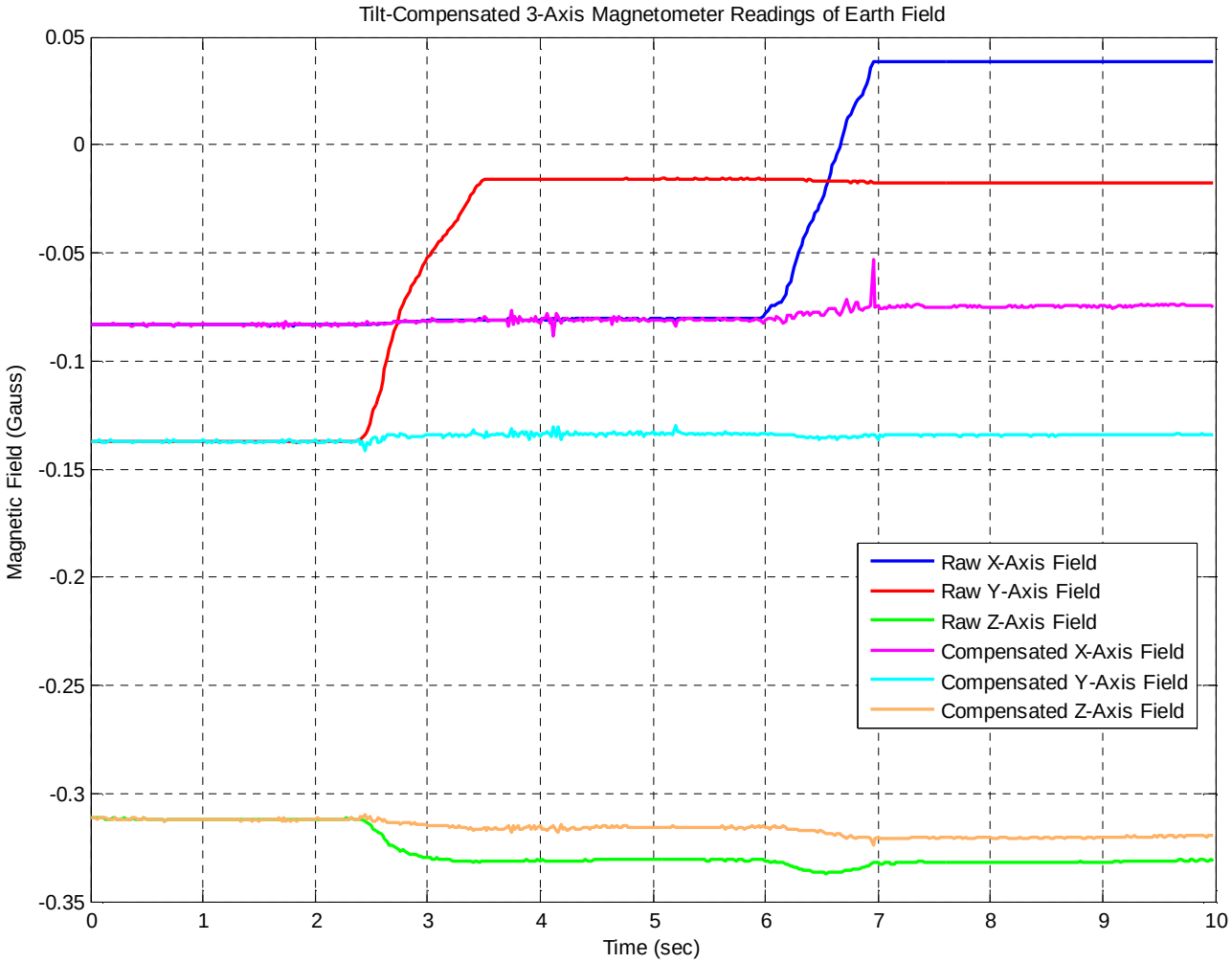


Figure 6.9: Calibrated 2-axis compensated earth field reading.

Calibration of the system clearly has improved the compensated output of the system in the X and Y axes, yet has produced a discrepancy in the compensated Z-axis output. A small discrepancy remains in the X-axis output as well. The ideal compensated output is a steady field with zero drift (ignoring extremely negligible earth drift). Standard deviation of these steady field signals is zero. Analysis of the standard deviation compensated signals of the one-axis and two-axis tilt experiments above before and after the calibration routine is performed is shown in Table 6.4 and 6.5, respectively. Reduction in standard deviation is a result of improvement in tilt-compensation resulting from calibration. With optimal compensation, standard deviation of the output signals approach those of ideal steady earth field, or a value of zero.

Table 6.4: Standard deviation of calibrated and uncalibrated magnetic output with single-axis negative pitch motion.

Axis Reading	Standard Deviation Uncalibrated (Figure 6.5)	Standard Deviation Calibrated (Figure 6.6)
X-axis	-60.6 dBGauss	-61.8 dBGauss
Y-axis	-42.4 dBGauss	-63.4 dBGauss
Z-axis	-54.2 dBGauss	-66.0 dBGauss

Table 6.5: Standard deviation of calibrated and uncalibrated magnetic output with two-axis motion.

Axis Reading	Standard Deviation Uncalibrated (Figure 6.8)	Standard Deviation Calibrated (Figure 6.9)
X-axis	-36.8 dBGauss	-44.4 dBGauss
Y-axis	-41.0 dBGauss	-61.4 dBGauss
Z-axis	-58.4 dBGauss	-50.8 dBGauss

The calibration routine has shown improvement of over 20 dBGauss of deviation reduction in the Y axis output in both tests, and 10 dBGauss of improvement in deviation in the two-axis test for the X-axis output. X is relatively unchanged in the roll-only test and Y is unchanged in the pitch-only test, as expected. In the Z direction, as mentioned above, there was an increase in deviation from the calibration of 7.4 dBGauss for two-axis stimulation. Likely from field inconsistencies, this tolerance in the output of the Z-axis is accepted for this project, as tests of other inclination combinations show improvement in the Z-axis output with the same calibration coefficients.

6.4 Hardware Implementation

To properly calculate the output of the magnetometer system with calibration coefficients and delta terms seen in Equation set 6.10, alteration of the tilt compensation function discussed in Section 6.2 is carried out. Seen in Appendix C, vector `_H_raw_xyz` is changed to a 1x6 vector that now contains the B_X, B_Y and B_Z axis information, as well as the ΔB_X, ΔB_Y, and ΔB_Z information calculated in the main function. The roll and pitch vectors are expanded to 1x6 with the sine and cosine values of the roll and pitch angles, respectively, loaded into the proper indexed location for equation set 6.10. Finally, the vectors `rho_x, rho_y` and `rho_z` contain the calibration coefficients for each respective axis, indexed as necessary to ensure the proper calculation of equation set 6.10. When these vectors are multiplied with one another and the output is calculated via the dot product operation, the output values of the calibrated and compensated X, Y, and Z-axis values result.

Timing this function using the simulation tool of the dsPIC debugger, it is found that calculation of Equation set 6.10 takes exactly 523 instruction cycles, or 21.2 microseconds at 6.144 MIPS. This calculation shows the true power of the dsPIC core of the microcontroller, as this same calculation would take several thousands of instruction cycles if implemented with simple accumulator multiplication, and many thousands of instructions if floating point calculations are implemented.

Chapter 7 Testing and Results

This chapter outlines several tests to display the operation and effectiveness of the tilt-compensation of the magnetometer system. Steady earth fields are detected and processed with several variations of motion stimulation. Time-varying magnetic fields are also detected, processed and displayed, exhibiting the system's ability to successfully detect low frequency signals and signatures. In addition, a compass algorithm, which outputs the heading of the magnetometer relative to the detected horizontal plane of the earth's magnetic field, is tested.

7.1 System Specifications

The performance and capabilities of the magnetometer system to detect low frequency magnetic fields depends on the design specifications of the various system tools. The sample rate, and thus bandwidth of the system, is limited by the execution speed of the necessary compensation and filtering algorithms. The order of the implemented FIR and median filter determine the level of noise and outlier attenuation, as well as phase delay.

7.1.1 Sampling Frequency

The process of integration of the various algorithms discussed in previous chapters into a tilt-compensated magnetometer system begins by first identifying the rate of data sampling.

Implementing the inclination filtering process by means of an 25th order α-trimmed median filter (concluded from Chapter 4) and use of the traditional algorithm for trigonometric calculation (from Chapter 5), as well as a 25th order FIR filter (from Chapter 4), a maximum sample frequency is calculated based on executed instruction cycles. The number of cycles used for these calculations, along with the data sampling and all overhead is calculated and tested at approximately 9,000 instruction cycles, resulting in a maximum sampling frequency of 680 sps at an operating speed of 6.144 MIPS.

When utilizing the CORDIC algorithm for trigonometric calculations, rather than the traditional algorithm, the maximum sampling frequency is calculated at 525 sps, or 11,700 instructions per sample.

In these calculations, the number of instruction cycles required to execute the FIR lowpass and median filter calculations is considered negligible, as the difference in execution time from a small order (approximately 25^{th} order) to a higher order filter (approximately 75^{th} order) is small relative to the overall execution time.

With either algorithm, the maximum sampling frequency is, for this project, undesired, as no time or instruction cycles remain for transmitting data for plotting. At a data rate of 115.2 kbps, transmitting the three axes' raw magnetic, three axes compensated, and two angle of inclination readings to the user at 16 bits each requires 1.11 milliseconds, or 6,826 instructions cycles. In

addition, in a realized application of the magnetometer system, utilization of the compensated magnetic signals for a desired purpose would require time. Thus, a sample rate that is slower than the maximum is used in the magnetometer programming. For the purpose of this project, a sample rate of 250 samples per second is implemented to allow for data transmission time, while still maintaining an acceptable bandwidth for field detection and compassing [1].

At this sample rate, the acceleration impulse of the inclination signal when the tilt platform is abruptly stopped consumes two to three samples, and thus the minimum order of the FIR and median filters will be 25 (see Section 4.1). Tests are also implemented utilizing 75th order filters, displaying the variation in output from the change in order.

7.1.2 Filter Specifications

As discussed in Chapters 4 and 5, the order of the median filter and FIR filter for the inclination and magnetic signals, respectively, are to possess the same order.

The FIR lowpass filters tested are of 25th (minimum order) and 75th order, both utilizing the Blackman window with the fractional cutoff frequency of 0.25. At a 250 sps sample rate, this equates to a 62.5 Hz cutoff frequency. This filter specification can easily be adjusted in design by the user for a target cutoff frequency of interest, based on the intended purpose of the magnetometer system and sources of magnetic noise to be attenuated.

Design of the 25th order α-trimmed median filter includes a value of $\alpha = 0.10$, or 3 samples removed from each end of the input data set (see Section 4.1.1).

The 25th-order FIR lowpass filter impulse and frequency response are shown below in Figure 7.1.

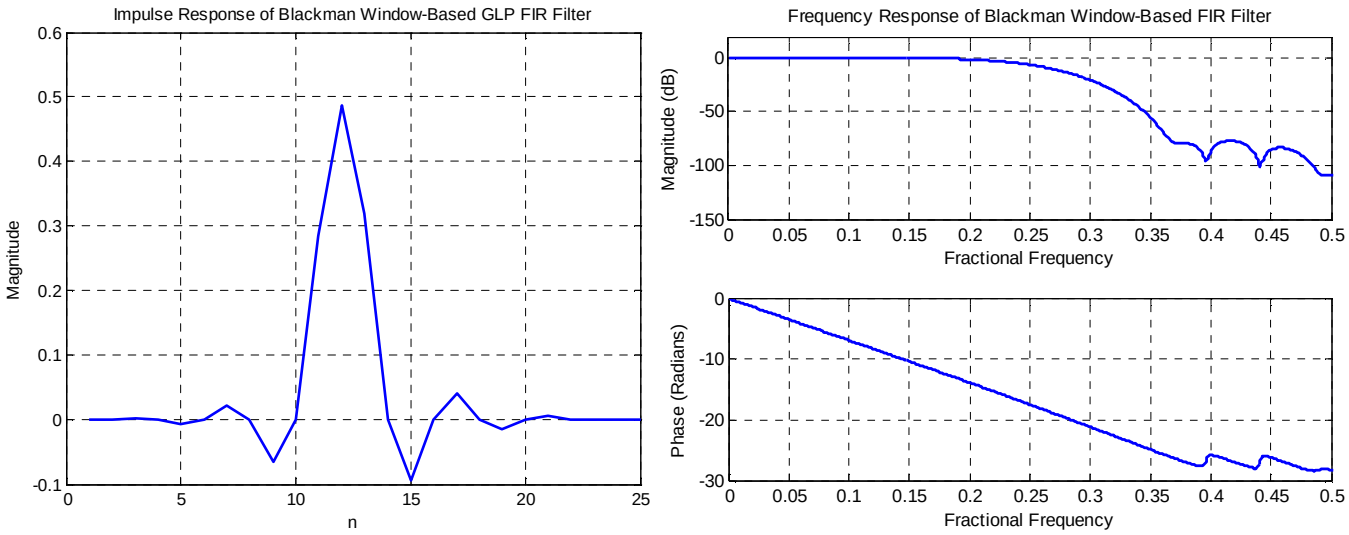


Figure 7.1: Impulse (left) and frequency (right) response of 25th-order Blackman window FIR lowpass filter.

The 75th-order implementation of the α-trimmed median filter includes a value of α = 0.20, or 15 trimmed samples.

The impulse and frequency response of the 75th order FIR lowpass filter with fractional frequency cutoff of 0.25 are shown below in Figure 7.2.

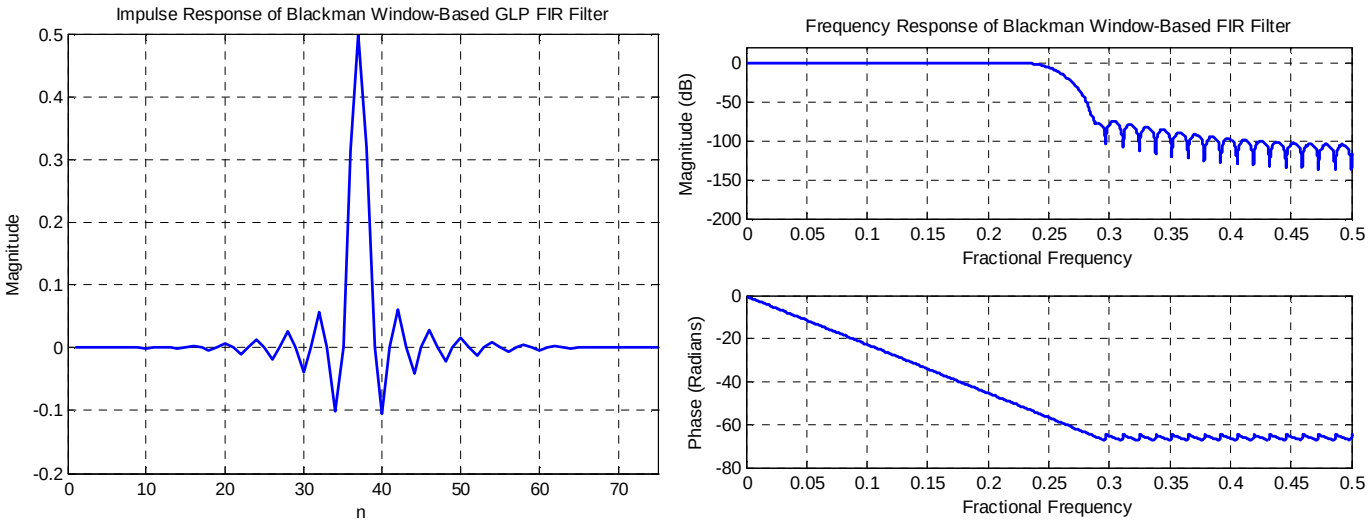


Figure 7.2: Impulse (left) and frequency (right) response of 75th -order Blackman window FIR lowpass filter.

7.2 Testing and Results

Testing of the tilt-compensated magnetometer system is accomplished by detecting several types of magnetic signals. Tests are implemented to detect earth's steady magnetic field, and the standard deviation of the steady fields is a benchmark for determining the performance of the system. Reflecting a steady field, minimal deviation of the output signals displays optimal performance. In addition, magnetic perturbations including simulated introduction of a passing metallic sphere and a sine wave field, both generated from a current-driven solenoid, are presented to the tilt-compensated system to display its capability of detecting varying magnetic signals.

All tests are performed for a standardized ten second time interval, and all tilt angle plots (subplots in figures) are maintained as unfiltered, raw angle output of the magnetometer system to allow for observation of inclination filtering effectiveness.

7.2.1 Earth Field Detection

Two tests are executed to examine the tilt-compensated magnetometer's performance. These tests include the gravity-driven step-like response of the inclinometers discussed in Chapters 3 and 4, as well as tests of random motion with abrupt stops to all platform tilt. This test is performed by tilting the platform until it contacts wooden stops, creating small acceleration impulses.

To examine the performance of the filtering algorithms effect on the high-frequency acceleration signals of the gravity-driven step response, three filter scenarios are tested. Unfiltered, 25thorder, and 75th-order FIR and median-filtered signals are displayed in Figures 7.3, 7.4 and 7.5, respectively.

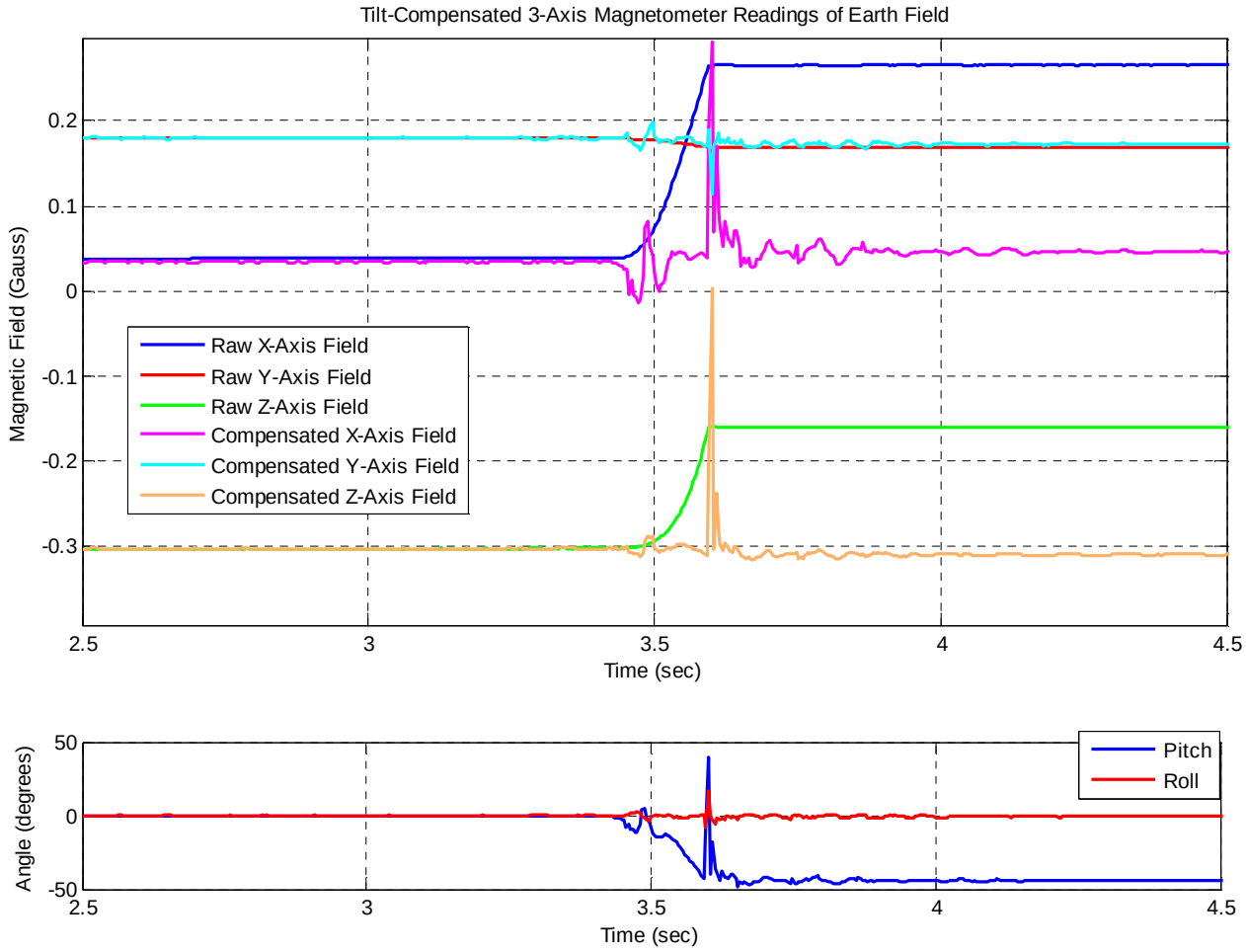


Figure 7.3: Gravity-driven step response of tilt-stabilized magnetometer system with no signal filtering.

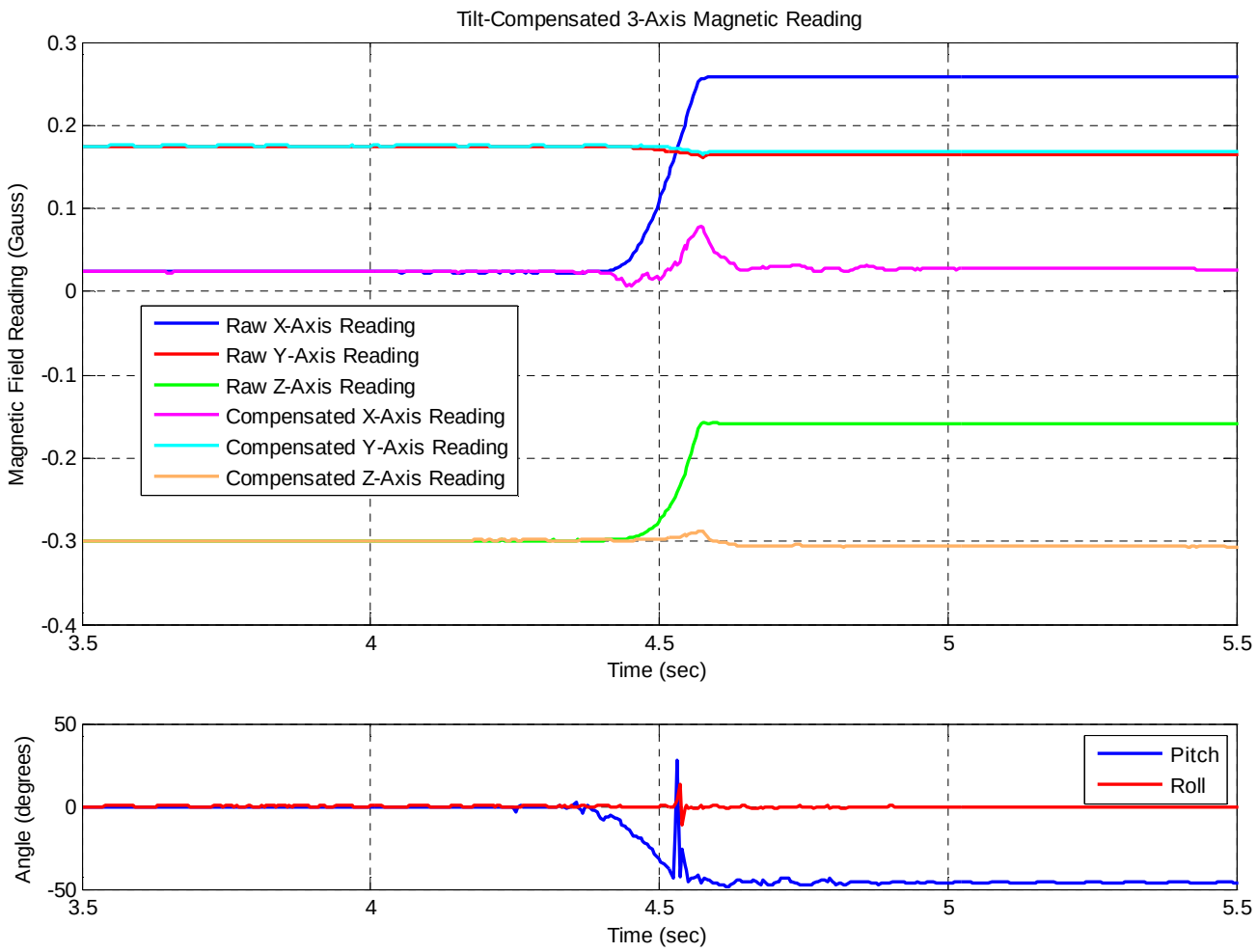


Figure 7.4: Gravity-driven step response of tilt-stabilized magnetometer system with 25th-order FIR and median filters.

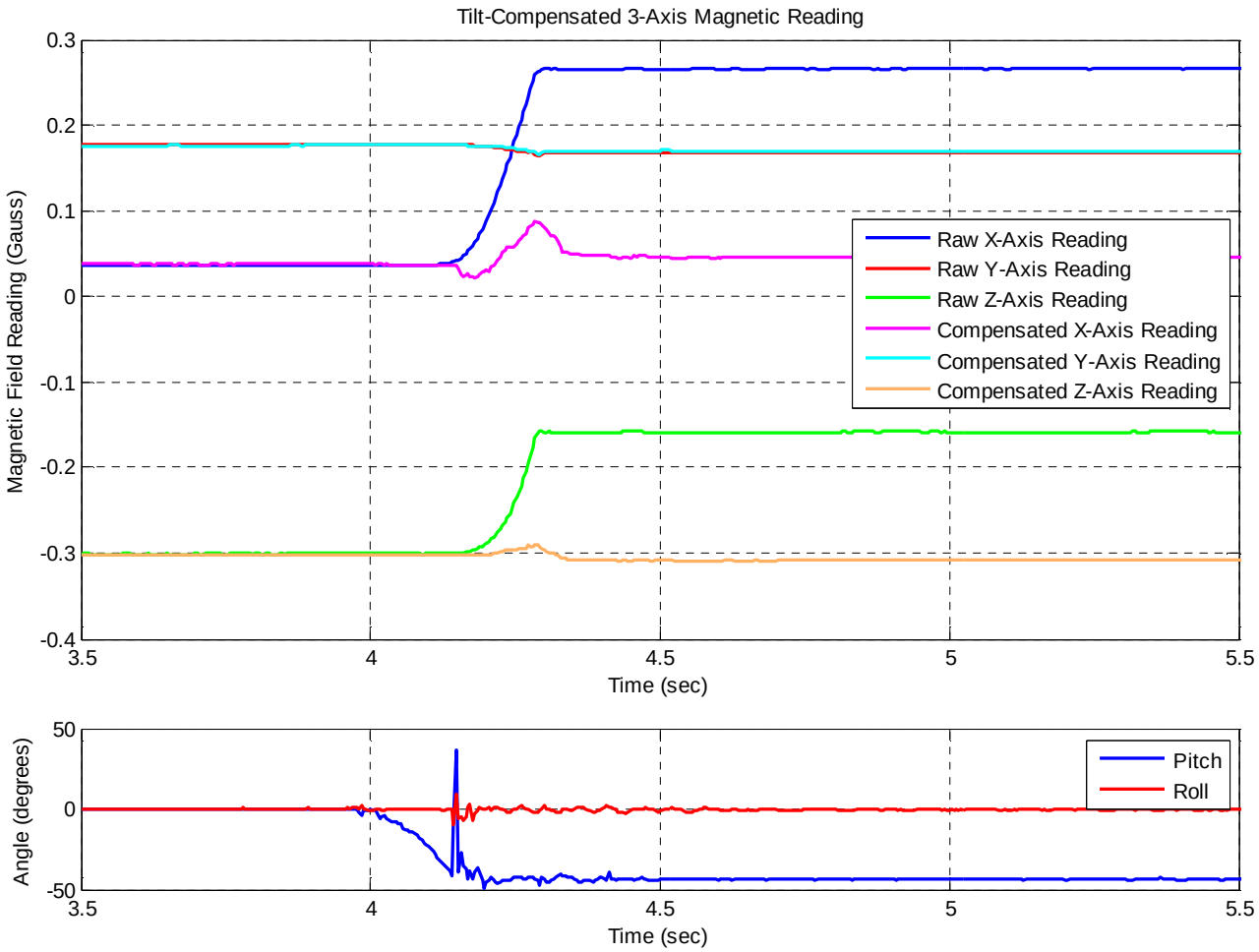


Figure 7.5: Gravity-driven step response of tilt-stabilized magnetometer system with 75th-order FIR and median filters.

Standard deviation of the steady earth field output of the system is calculated using a standardized time period spanning from 2 seconds prior to the step excitation to 2 seconds after.

Table 7.1: Standard deviation of unfiltered and filtered step response output signal with 25^{th} and 75^{th}-order FIR and α-trimmed median filters.

Axis Reading	Standard Deviation Unfiltered Output	Standard Deviation 25^{th}-Order Filtered Output	Standard Deviation 75^{th}-Order Filtered Output
X-axis	-34.2 dBGauss	-41.4 dBGauss	-40.6 dBGauss
Y-axis	-45.6 dBGauss	-49.6 dBGauss	-50.0 dBGauss
Z-axis	-35.2 dBGauss	-48.0 dBGauss	-48.0 dBGauss

The X-axis output displays a 3.6 dBGauss and 3.2 dBGauss drop in deviation when 25th, and 75th-order filters are implemented, respectively. In the Z-axis output, 6.4 dBGauss improvement is achieved with both filter orders. Improvement was also seen in the Y-axis, as the cross-axis effects of the inclinometer signal is reduced with both orders of filters. From visual inspection of the system output, there is significant steady field improvement when the FIR and median filters are implemented in the system. With similar output standard deviation, as well as visual improvement with both order of filters, it can be determined that 75th order filters are not necessary to achieve desirable results with high-acceleration conditions.

The next test that is performed is motion in both axes of tilt with abrupt stop of the directional changes. This is achieved by pitching, then rolling the tilt platform in both the positive and negative angle directions, stopping the motion with pre-sized wooden blocks. From this test, four acceleration impulses are produced in the inclination signals from each acceleration (abrupt stop) inclination impulse. Unfiltered outputs, along with the two filtered outputs at 25th and 75th-order filtering are seen in Figures 7.6, 7.7, and 7.8, respectively.

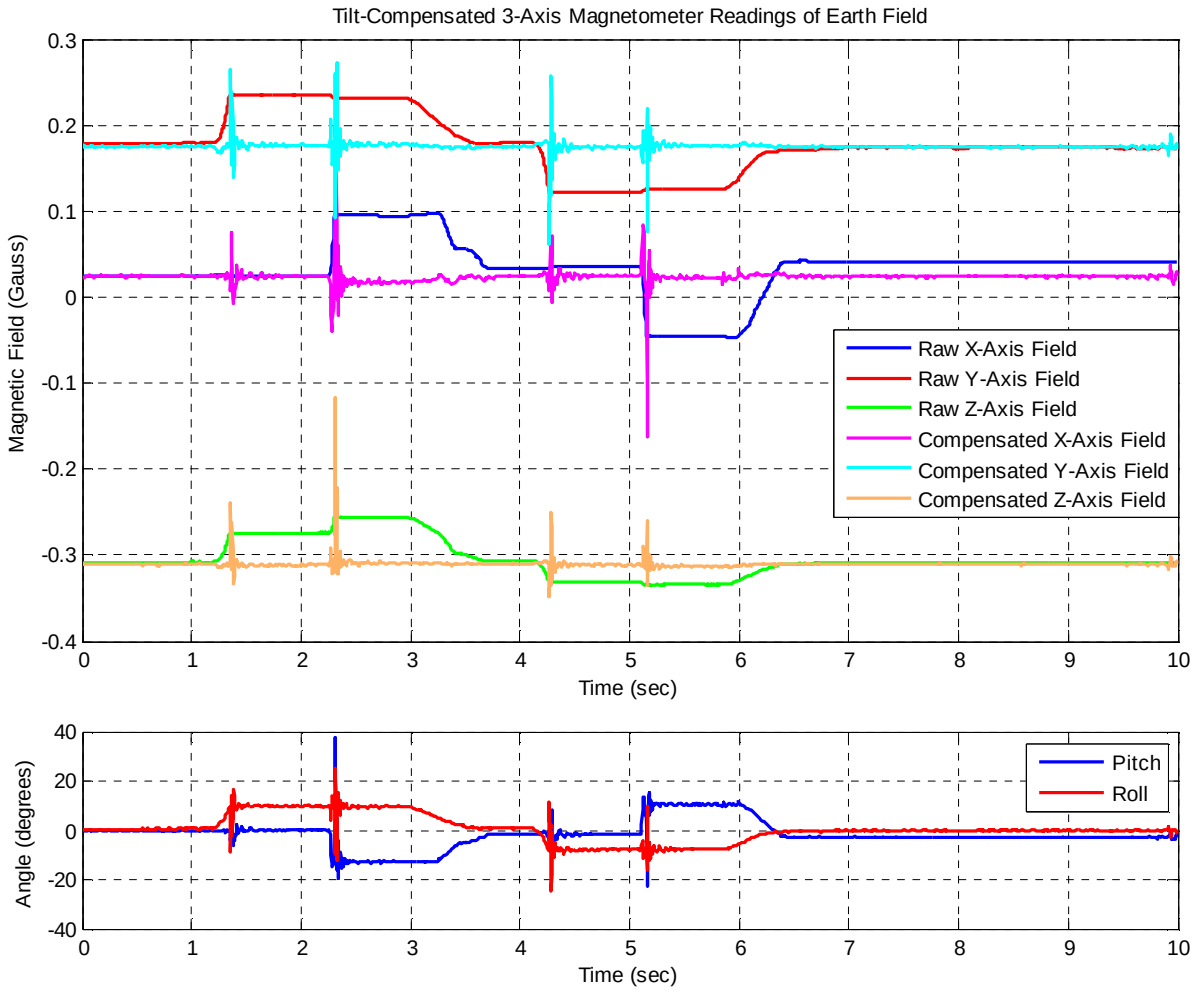


Figure 7.6: Unfiltered output of magnetometer with abrupt deceleration.

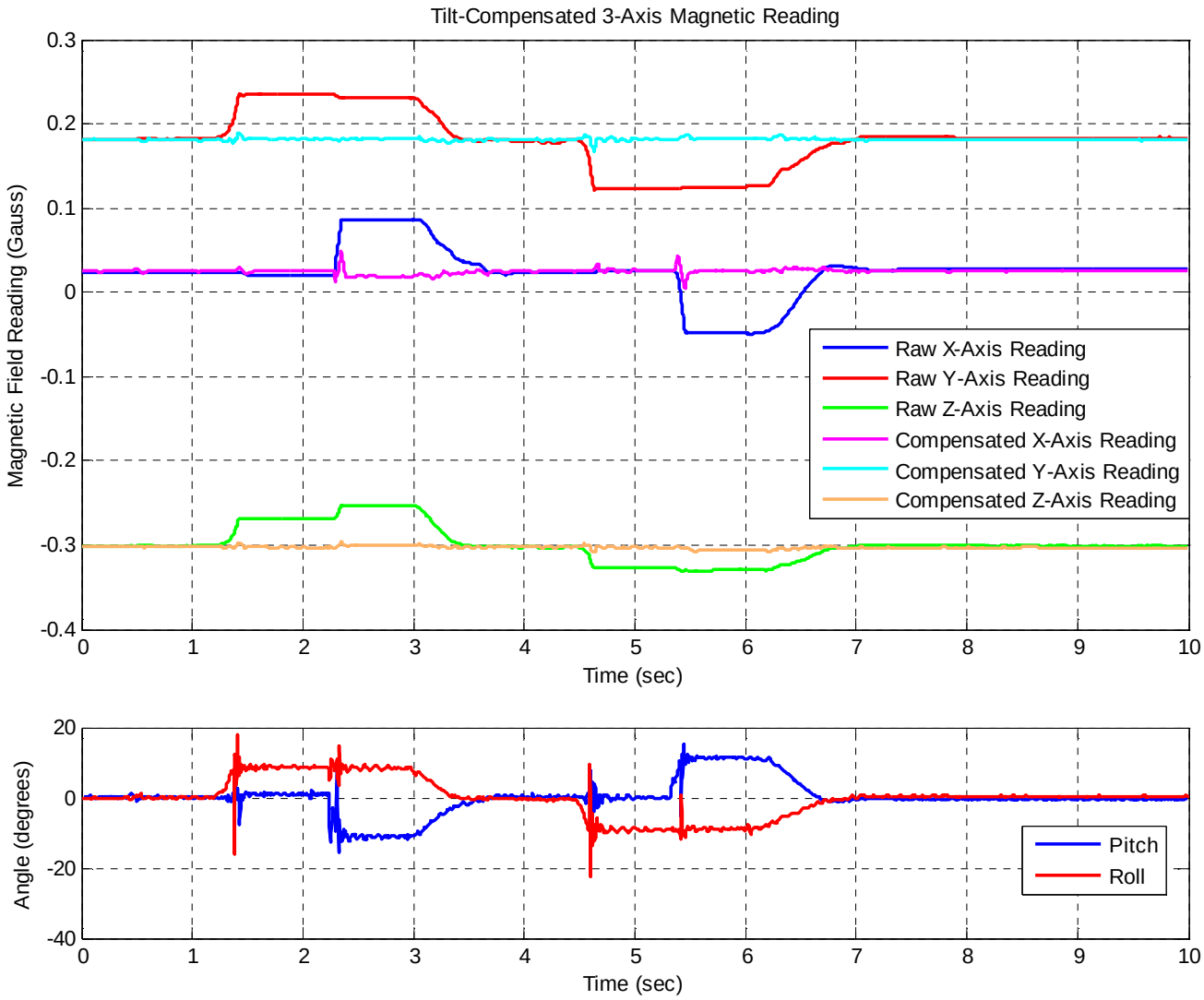


Figure 7.7: Magnetometer output with abrupt deceleration implementing 25th-order filters.

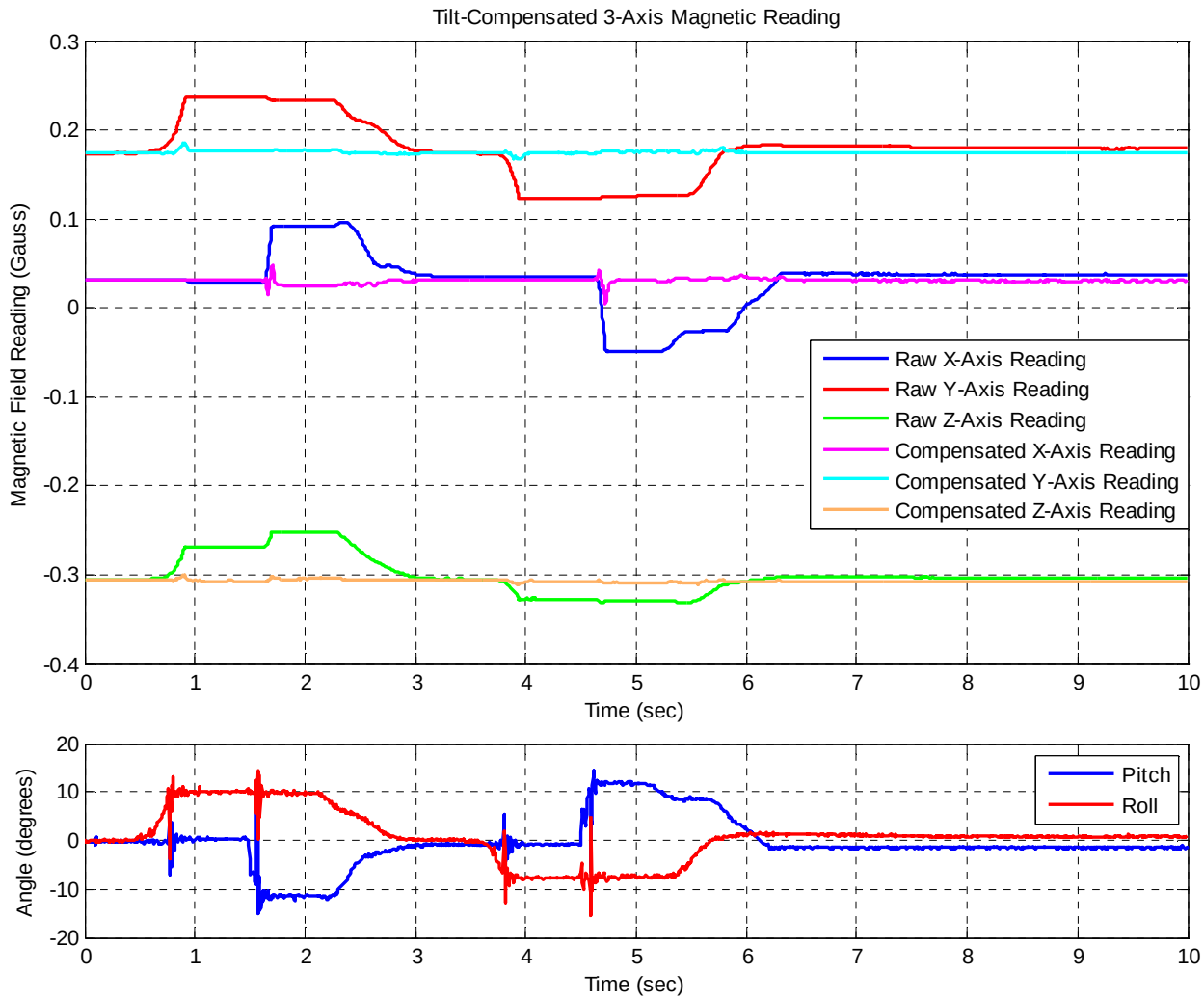


Figure 7.8: Magnetometer output with abrupt deceleration implementing 75th-order filters.

Table 7.2: Standard deviation of unfiltered and filtered output signal with 25^{th} and 75^{th}-order FIR and α-trimmed median filters.

Axis Reading	Standard Deviation Unfiltered Output	Standard Deviation 25^{th}-Order Filtered Output	Standard Deviation 75^{th}-Order Filtered Output
X-axis	-40.6 dBGauss	-51.2 dBGauss	-52.0 dBGauss
Y-axis	-43.4 dBGauss	-56.6 dBGauss	-56.8 dBGauss
Z-axis	-44.8 dBGauss	-56.6 dBGauss	-58.0 dBGauss

Standard deviation measurement of the steady output signals is not an exact measurement of the tilt-compensated results, due to the non-standardized motion of the tilt platform, However, this calculation can be used as an approximation of the performance improvement gained from the filtering of the inclination and magnetic signals. Improvements from 10.6 dBGauss to 13.2

dBGauss are presented when 25th order filters are implemented on the magnetometer system output signals. In addition, when the higher (75) order filter is used, improvement in the deviation of the compensated signals is produced in all three axes, seen in Table 7.2. This improvement is minimal in the Y-axis (additional 0.2 dBGauss), however the expected result of improved SNR is achieved.

Finally, slow rates of tilt are applied to the magnetometer platform to display the tilt-compensating capabilities of the system when no acceleration impulse is present in the input signals. Plots of just filtered outputs of the system can be seen in Figures 7.9 and 7.10 below. Standard deviation of the compensated output of these tests can be seen in Tables 7.3 and 7.4, respectively. Azimuth angle of the test seen in Figure 7.10 is also calculated in Section 7.2.2.

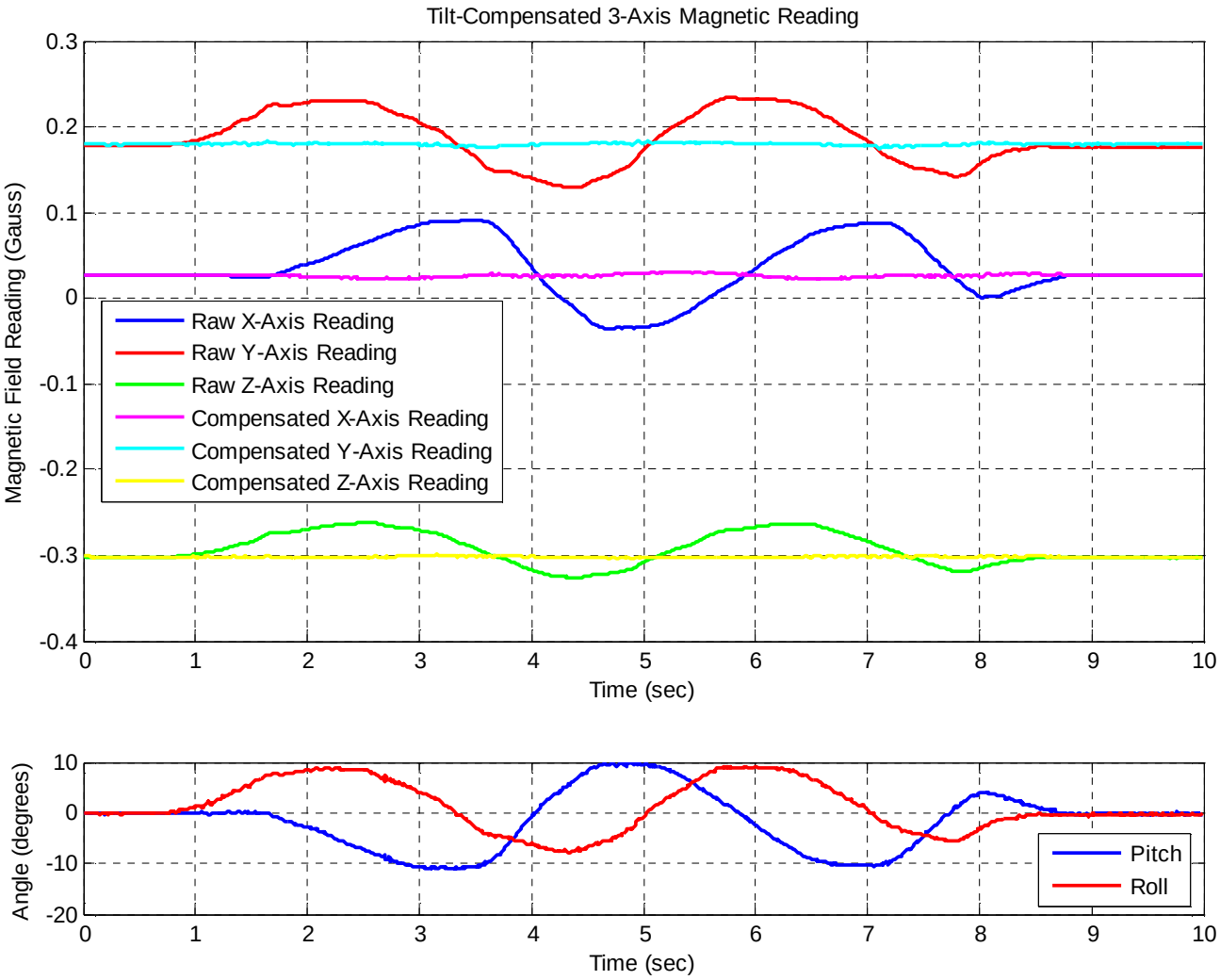


Figure 7.9: Random motion of magnetometer system with 25th-order filter implementation.

Table 7.3: Compensated output standard deviation from test seen in Figure 7.9.

	Standard Deviation of Uncompensated Output	Standard Deviation of Compensated Output	Maximum Field Deviation of Uncompensated Output	Maximum Field Deviation of Compensated Output	% Error Reduction Field Deviation
X-Axis	-29.4 dBGauss	-54.8 dBGauss	65.1 mGauss	5.2 mGauss	92.0%
Y-Axis	-30.4 dBGauss	-57.2 dBGauss	56.2 mGauss	4.7 mGauss	91.6%
Z-Axis	-34.6 dBGauss	-63.0 dBGauss	39.1 mGauss	3.7 mGauss	90.5%

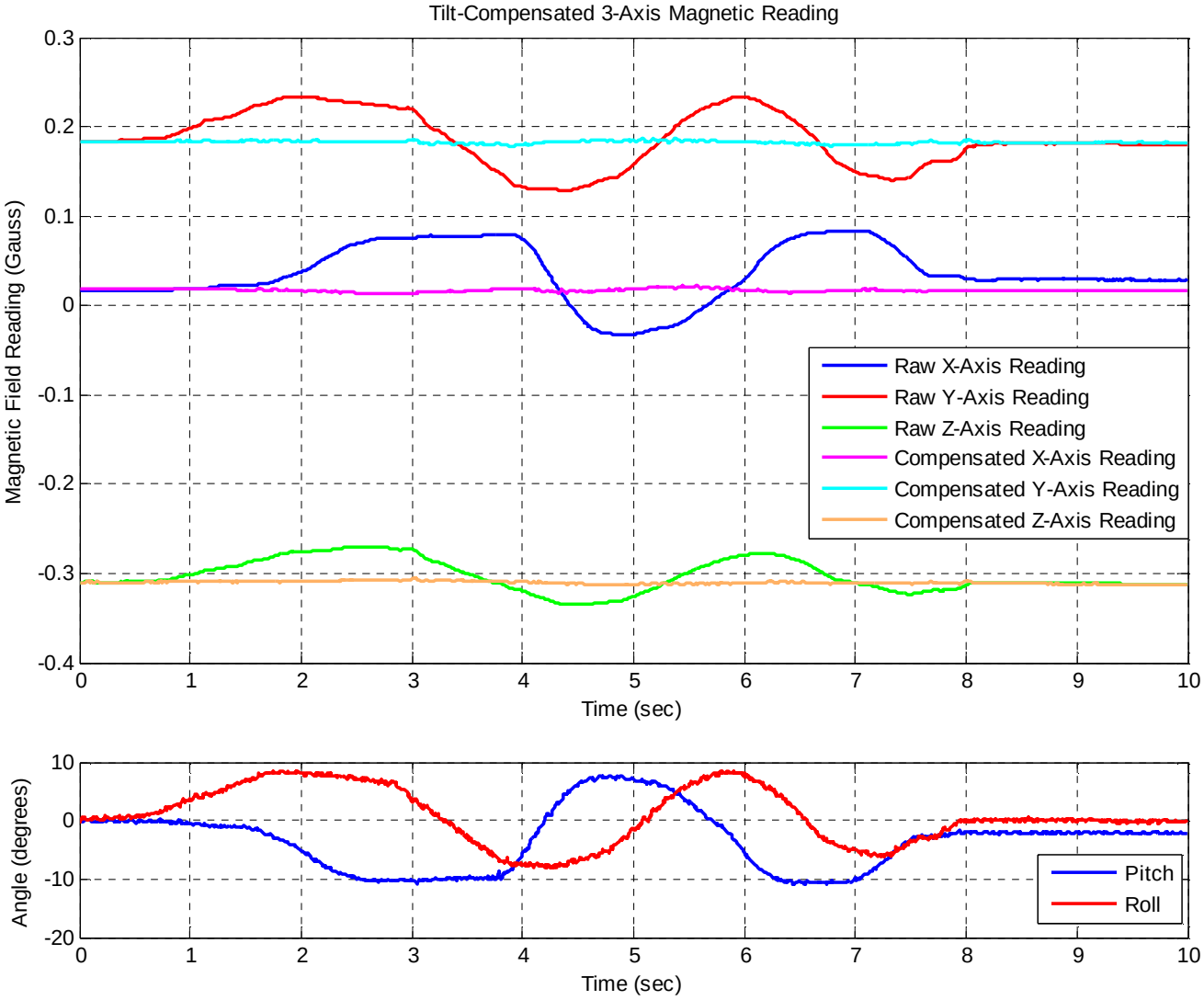


Figure 7.10: Random motion of magnetometer system with 75th-order filter implementation.

Table 7.4: Compensated output standard deviation from test seen in Figure 7.10.

	Standard Deviation of Uncompensated Output	Standard Deviation of Compensated Output	Maximum Field Deviation of Uncompensated Output	Maximum field Deviation of Compensated Output	% Error Reduction Field Deviation
X-Axis	-29.8 dBGauss	-55.8 dBGauss	66.2 mGauss	5.8 mGauss	91.2%
Y-Axis	-30.2 dBGauss	-55.8 dBGauss	55.2 mGauss	6.3 mGauss	88.6%
Z-Axis	-35.0 dBGauss	-56.8 dBGauss	39.8 mGauss	4.1 mGauss	89.7%

Results show significant improvement in both tests, with between 88% and 92% reduction in maximum field deviation resulting from tilt motion compensation. The system has reduced the standard deviation of the steady earth field output by between 21.8 dBGauss and 28.4 dBGauss when compared to the non-compensated output. These tests show that with low-frequency, steady motion, significant improvement in signal stability and reliability are achieved with the tilt-compensated magnetometer system.

7.2.2 Azimuth Calculation

The tilt-stabilized magnetometer has the additional function of utilizing the horizontal magnetic field calculations to determine its current azimuth angle. With 0º azimuth corresponding to a heading of true north and 180º corresponding to true south, the orientation of the tilt-compensated magnetometer (positive Y-axis reading relative to earth field north) is calculated via equation 7.1.

$$\boldsymbol{\theta_{azimuth}} = \tan^{-1}\left(\frac{\boldsymbol{Y_{horiz}}}{\boldsymbol{X_{horiz}}}\right) \tag{7.1}$$

Due to the limits of the arctangent function, several conditions are required to properly output the 0º to 360º azimuth angle (seen in Equation Set 7.2).

$$\boldsymbol{\theta_{azimuth}} = \begin{cases} \mathbf{180°} - \mathbf{\tan^{-1}}\left(\frac{\boldsymbol{Y_{horiz}}}{\boldsymbol{X_{horiz}}}\right) & , \quad \boldsymbol{X_{horiz}} < 0 \\ -\mathbf{\tan^{-1}}\left(\frac{\boldsymbol{Y_{horiz}}}{\boldsymbol{X_{horiz}}}\right) & , \quad \boldsymbol{X_{horiz}} > 0\,, \boldsymbol{Y_{horiz}} < 0 \\ \mathbf{360°} - \mathbf{\tan^{-1}}\left(\frac{\boldsymbol{Y_{horiz}}}{\boldsymbol{X_{horiz}}}\right), & \quad \boldsymbol{X_{horiz}} > 0, \boldsymbol{Y_{horiz}} > 0 \\ \mathbf{90°} & , \quad \boldsymbol{X_{horiz}} > 0, \boldsymbol{Y_{horiz}} > 0 \\ \mathbf{270°} & , \quad \boldsymbol{X_{horiz}} > 0, \boldsymbol{Y_{horiz}} > 0 \end{cases} \tag{7.2}$$

To calculate the azimuth angle based on the conditional statements, a function called `_Azimuth` is programmed for implementation on the dsPIC. Utilizing floating point operations for division and the arctangent function, the calculations are performed according to the conditions of the horizontal X and Y magnetic field readings. Due to the floating point calculations, this function is inefficient in execution time, requiring between 860 and 4877 instructions from call to return to perform the calculation in software, depending on the sign of the field readings. As an alternative, the function atan2 is used for the above calculation, with the notable drawback of requiring 3,206 instruction cycles for every instruction call. Improvements in the efficiency of this azimuth calculation are possible with alternative integer-based algorithms. However, for this project, the `atan2` function is programmed and used in software.

To examine the accuracy of the azimuth calculations resulting from the tilt-compensated system, two sample tests can be seen in Figures 7.11 and 7.12.

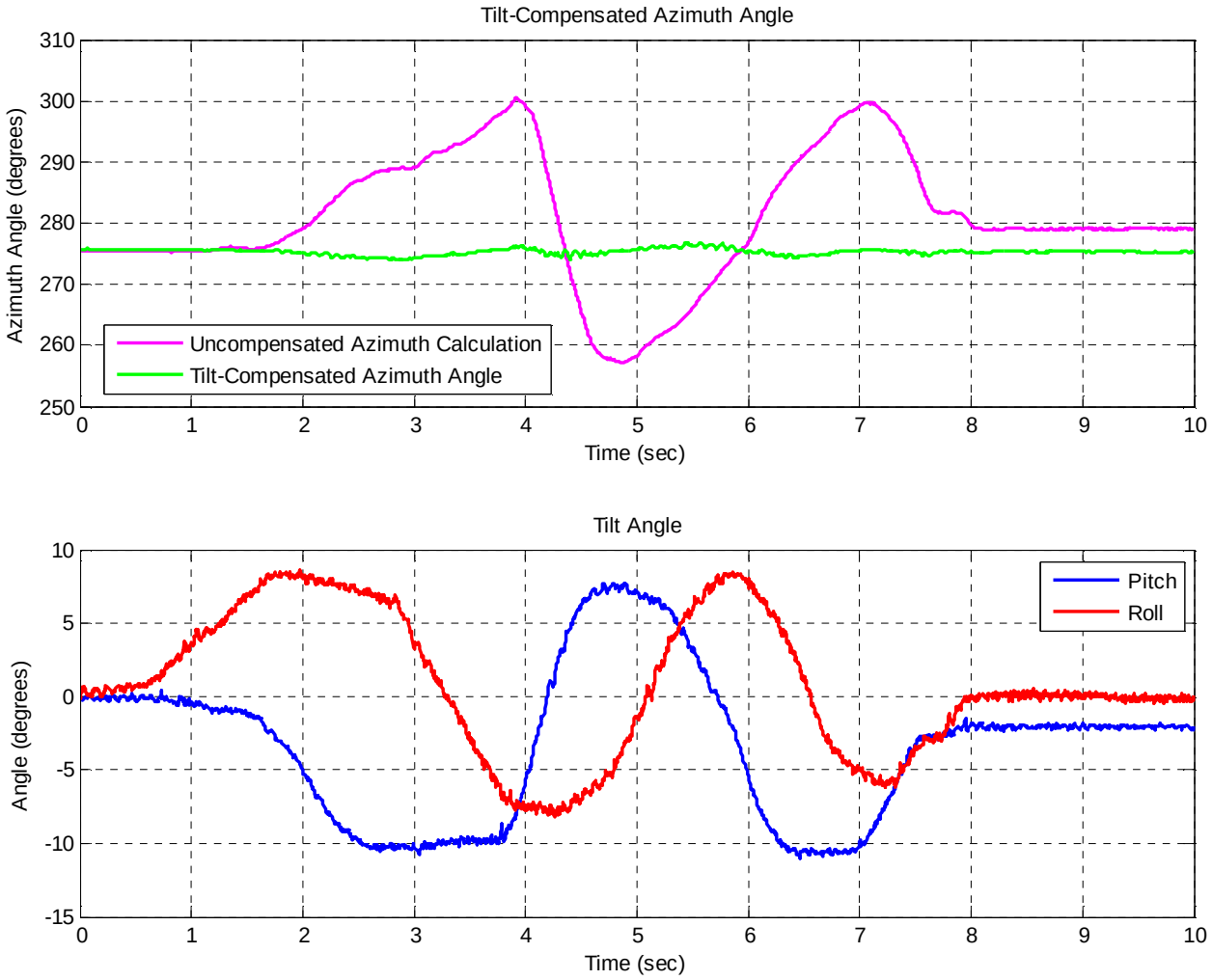


Figure 7.11: Azimuth angle of tilt-compensated magnetometer system implementing 75th-order filters (same test reflected in Figure 7.10).

Analyzing the azimuth output, it is calculated that if the magnetometer readings were left uncompensated, the azimuth reaches a maximum error of 24.8° from the steady state angle, with a standard deviation of 10.4°. Compensated, the maximum drift from the steady state is 1.7°, or a 93% reduction in error. Compensated standard deviation is calculated to be 0.50°.

Rotating the magnetometer, a second test is performed, seen in Figure 7.12 below.

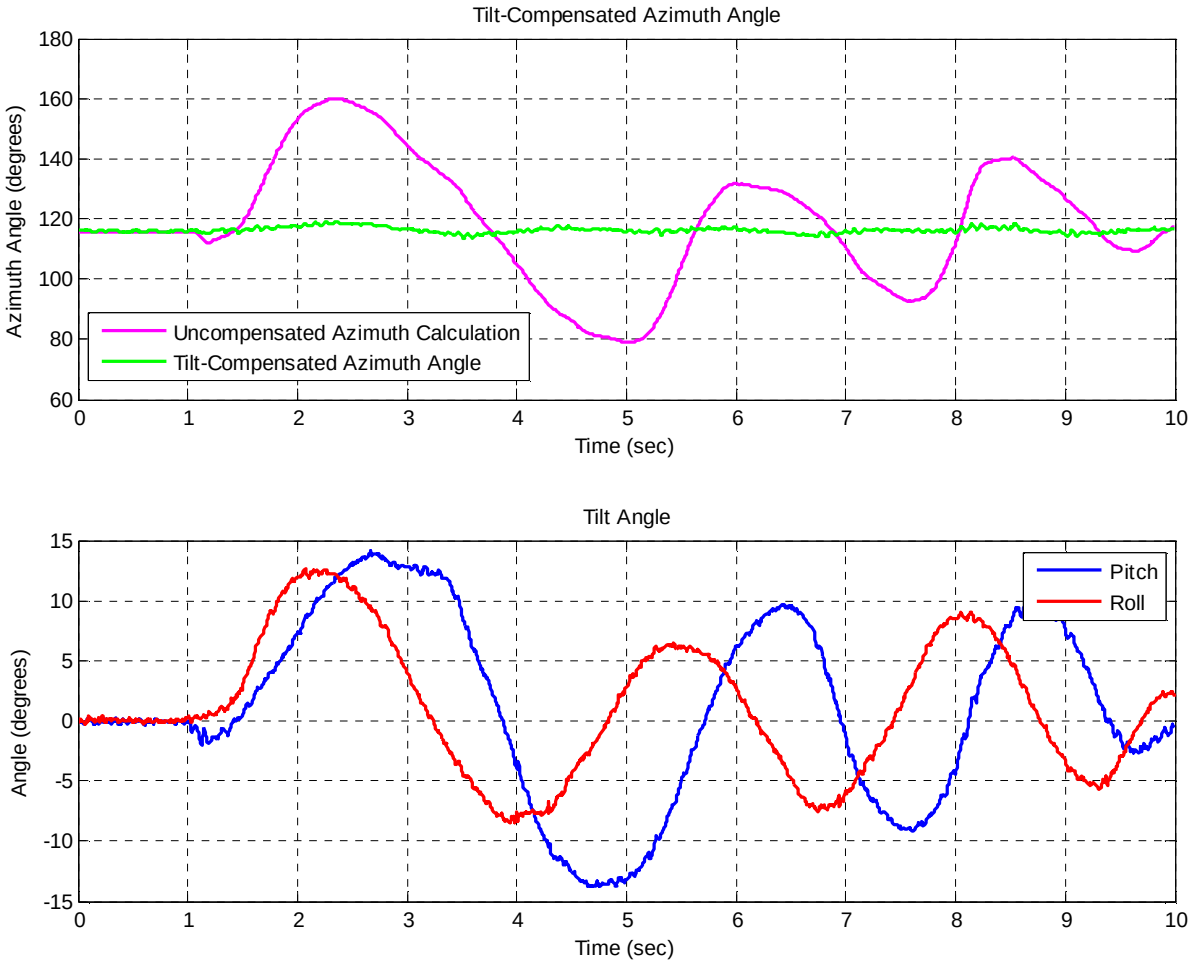


Figure 7.12: Azimuth angle of tilt-compensated magnetometer system implementing 25th-order filters.

This test displays up to 47° of error in the uncompensated azimuth output, and 2.9° of maximum error when compensated. This reflects a 93.8% reduction in azimuth angle error utilizing tilt-compensation. Standard deviation of the compensated and uncompensated outputs is calculated to be 1.0° and 40.5°, respectively.

Both tests present significant improvements in azimuth angle output when tilt-compensation is utilized. An overview of results from both tests can be seen in Table 7.3.

	True Azimuth Angle	Standard Deviation of Uncompensated Output	Standard Deviation of Tilt-Compensated Output	Maximum Deviation of Uncompensated Output	Maximum Deviation of Tilt-Compensated Output	Error Reduction from Tilt-Compensation
Test #1 (Figure 7.11)	275.7º	10.4 º	0.50 º	24.8 º	1.7 º	93.1%
Test #2 (Figure 7.12)	116.0º	40.5 º	1.0 º	44.0 º	2.9 º	93.8%

Table 7.5: Results comparison of azimuth angle tests seen in Figures 7.11 and 7.12.

With the added instruction cycles required for the execution of the azimuth calculation, the maximum sampling frequency drops to a maximum of 370 samples per second. Hence, a 250 sps sampling frequency remains adequate for azimuth calculation with several hundred microseconds to spare for further calculation or decision-making processes.

7.2.3 Varying Field Detection

In many field applications, magnetometers are required to detect varying fields. To demonstrate the tilt-compensated system's capability for this task, two tests are performed.

First, simulation of a magnetic object passing near the magnetometer is simulated by generating the magnetic signature of a slowly passing metallic sphere via a 60cm long (8cm diameter) solenoid. Using a current-controlled generator, a simulation of the fields reflecting a sphere passing the magnetometer system in the Y-axis direction is produced. The field is variable only in the Y-axis direction via the orientation of the solenoid parallel to this axis (see Figure 7.13) and a tilt-free reading of the generated field is seen in Figures 7.14 and 7.15 below. As expected, the generated signal has no effect on the cross-axes sensors (X and Z axes).

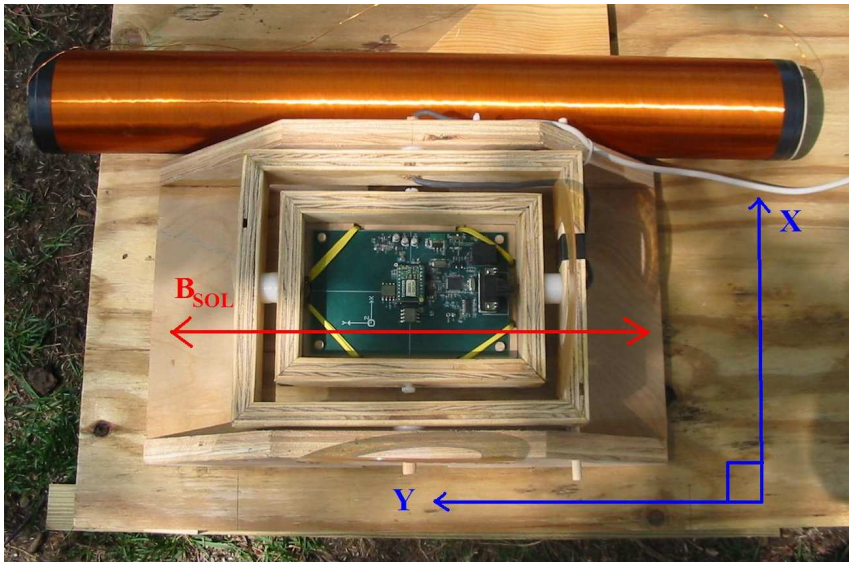


Figure 7.13: Electromagnetic solenoid positioned in Y-axis direction.

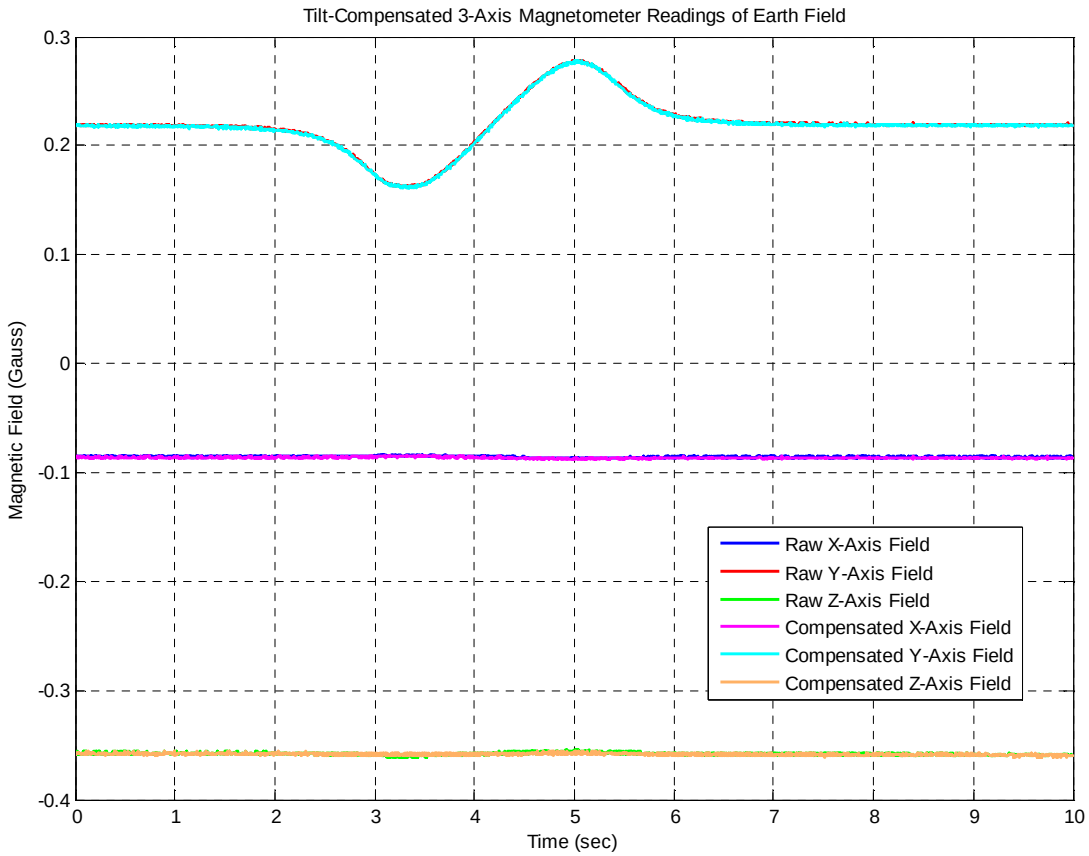


Figure 7.14: Output of motionless magnetometer system with generated sphere-proximity field presented in Y-axis.

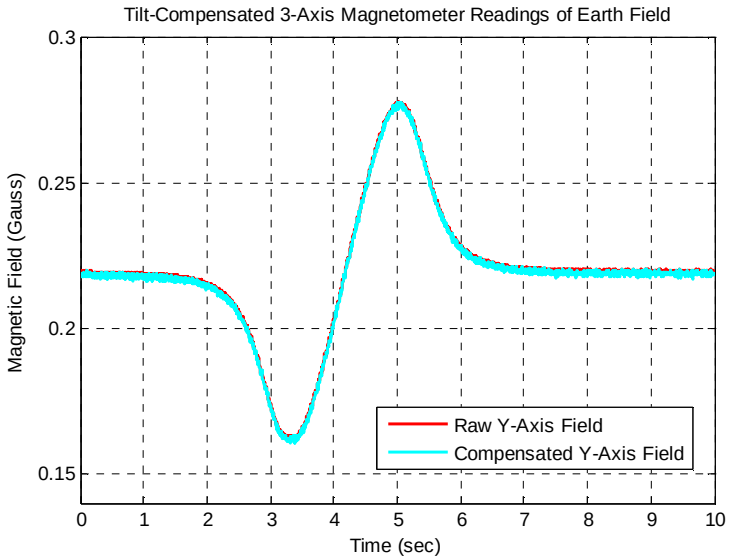


Figure 7.15: Isolated Y-axis reading of generated sphere signature seen in Figure 7.14.

Tilting of the magnetometer reveals the sphere signature in the compensated Y-axis output, seen in Figure 7.16.

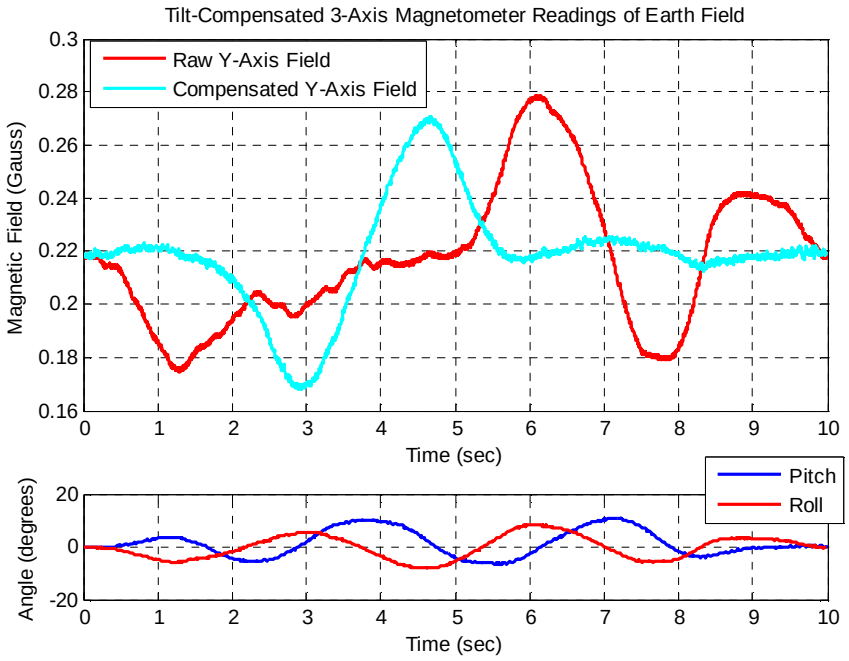


Figure 7.16: Isolated Y-axis tilt-compensated output of sphere signature detection.

The passing sphere is clearly visible, with approximate maximum magnitude maintained. Error in the steady state output of the system is due to the non-uniform nature of the field generated by the solenoid, as the field now possesses curvature (unlike the dipole earth field).

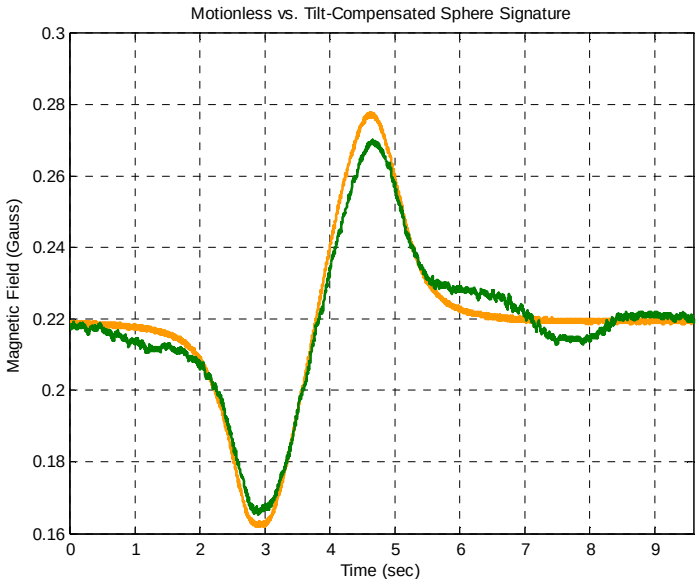


Figure 7.17: Comparison of motionless and tilt-compensated output of the magnetometer system.

Time-shifting the tilt-compensated signal to align peaks with the motionless signal (Figure 7.17), a signal to tilt-motion noise ratio (STNR) is capable of being calculated by equation set 7.3. ${\sigma_y}^2$ is the standard deviation of the output signal, and ${\sigma_t}^2$ is the standard deviation of the residuals of the motionless and tilt-compensated output.

$$STNR = \frac{{\sigma_y}^2}{{\sigma_t}^2} \tag{7.3}$$

For the test shown in the above figures, STNR is calculated to be 16.5 dB.

Improvement of this ratio can be achieved with more precise calibration of the system, as well as further signal amplification and filtering. This is discussed in Section 8.2, Recommended Future Work.

Detection of a sinusoid signal presented by the solenoid generator at an angle of 45º to the X and Y axes in the horizontal plane (see Figure 7.18) is presented in Figures 7.19 through 7.21 below. The sinusoid is programmed at 0.5 Hz.

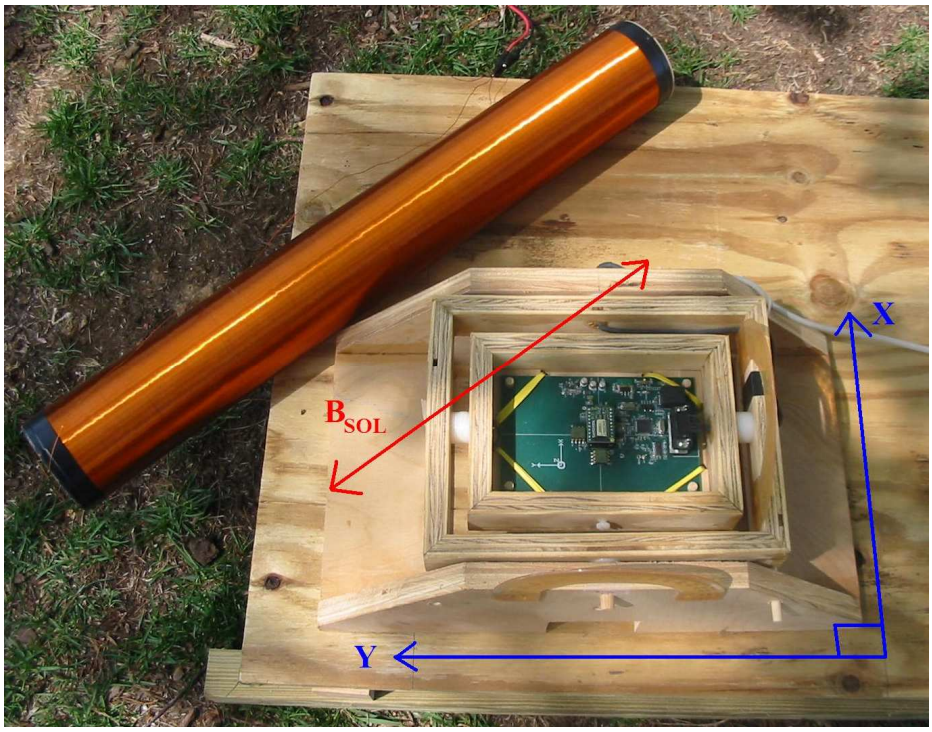


Figure 7.18: Positioning of electromagnetic solenoid at 45º angle to the X and Y axes in the horizontal plane.

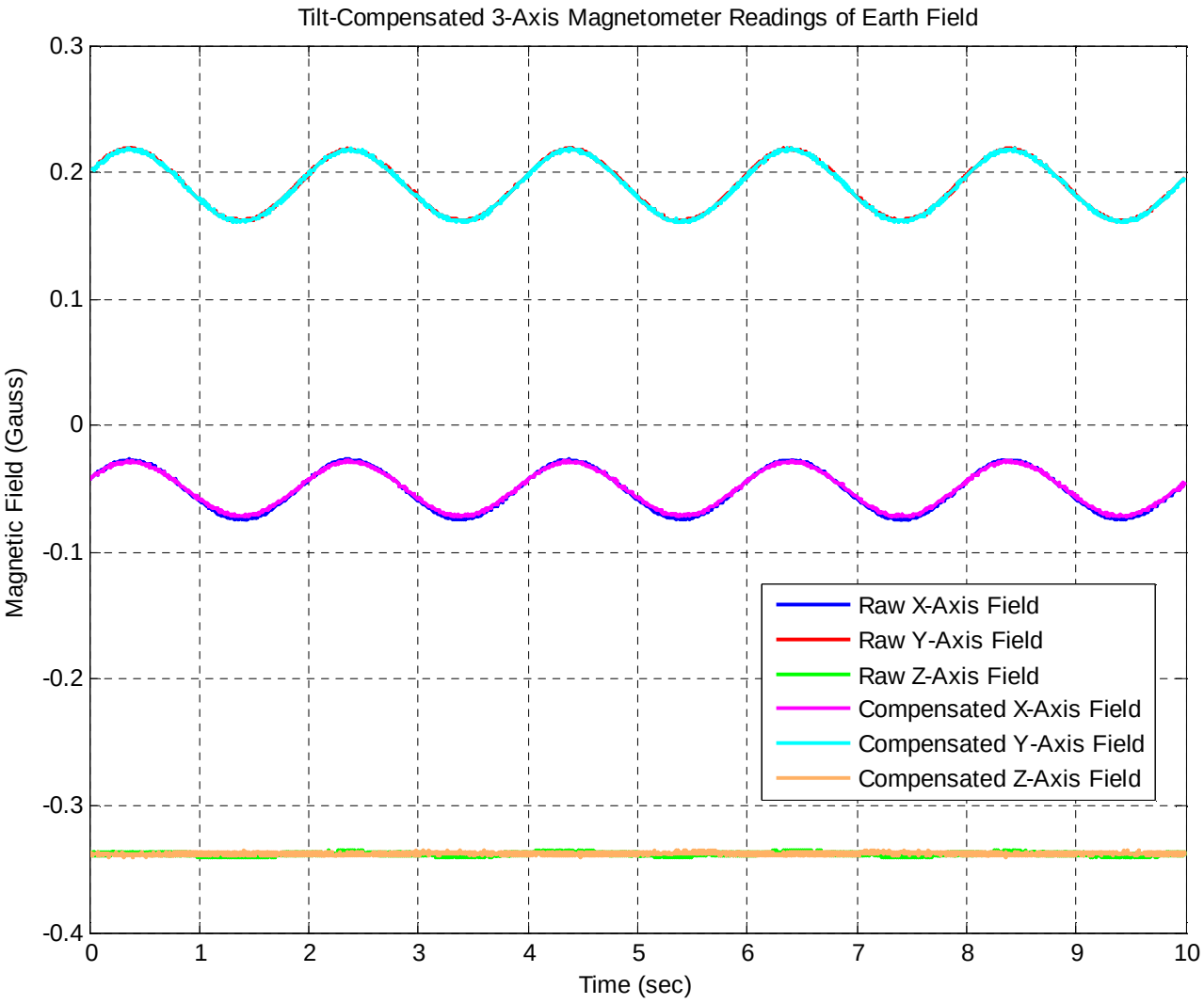


Figure 7.19: Generated sinusoid signal detected in the horizontal (X and Y axes) plane with no tilt.

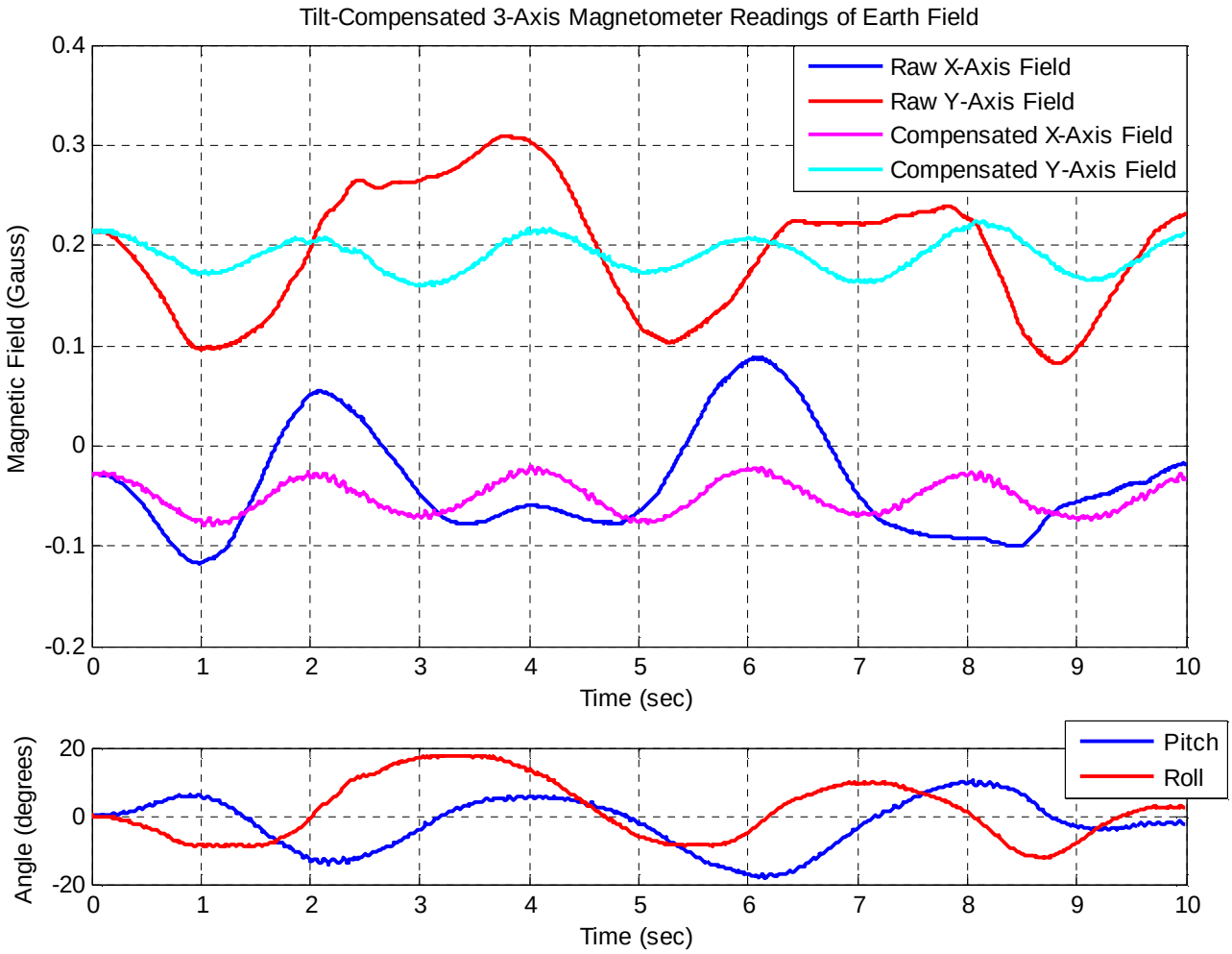


Figure 7.20: X and Y output of tilt-compensated magnetic field readings of generated sinusoid field.

Analyzing the Fourier Transform of the sinusoid (sans DC offsets) output including and excluding tilt motion reveals maintenance of the magnitude 0.5 Hz component of the tilt-compensated signal.

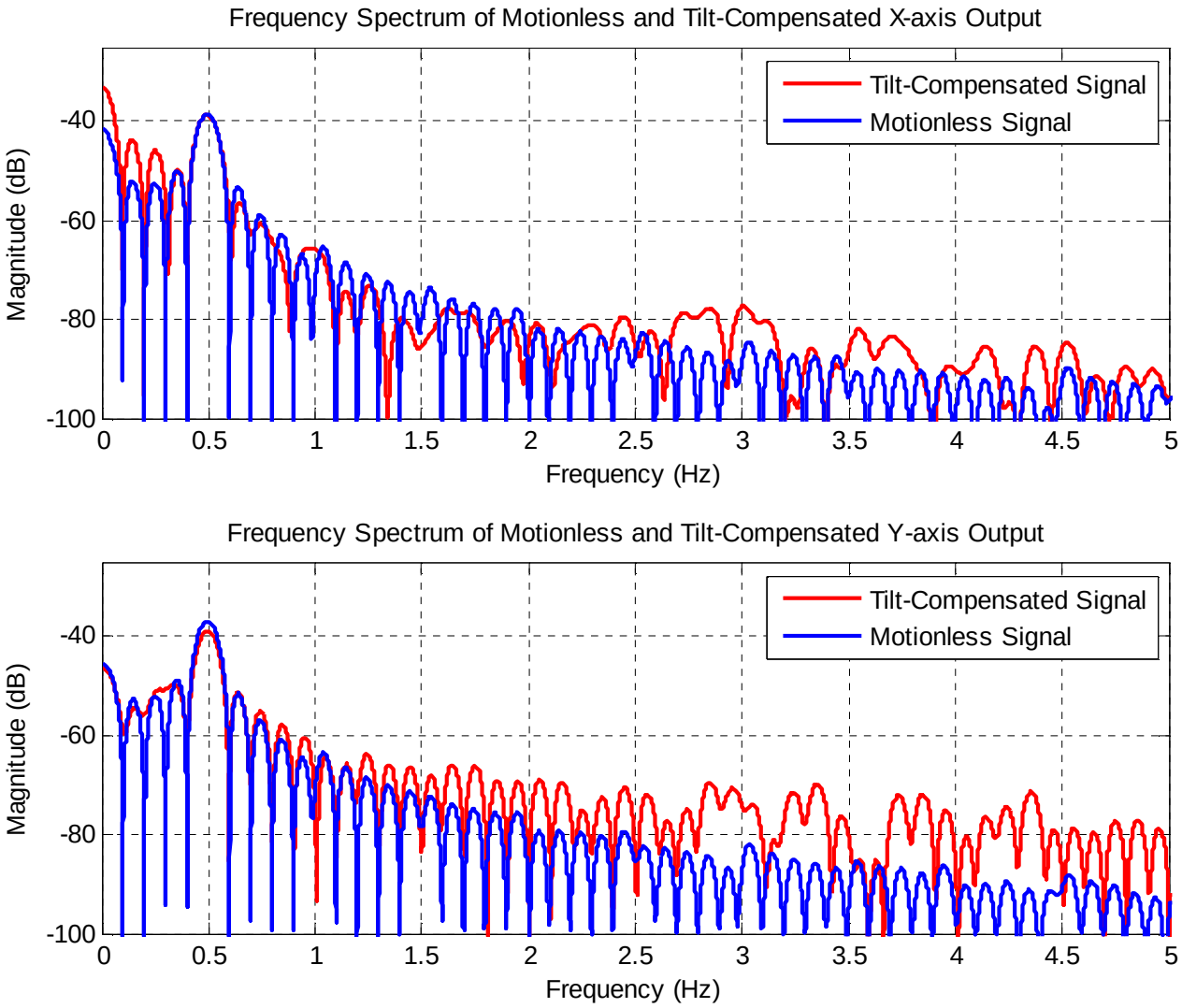


Figure 7.21: Frequency response of motionless and tilt-compensated magnetometer output.

Table 7.6: Comparison of 0.5 Hz component magnitude of motionless output versus tilt-compensated output.

Axis	0.5 Hz Component Magnitude Motionless	0.5 Hz Component Magnitude Including Tilt
X	-38.6 dBGauss	-38.8 dBGauss
Y	-37.0 dBGauss	-39.0 dBGauss

The 0.5 Hz component of the motionless outputs in the X and Y axes reflect magnitudes of -38.6 dBGauss and –37.0 dBGauss, respectively. The Y-axis output compensated for motion possesses a – 39.0 dBGauss magnitude, showing that 2 dBGauss of degradation of the sinusoid signal results

from the compensation calculations and error. In the X-axis, little degradation is seen, as only a 0.2 dBGauss discrepancy in the 0.5 Hz component magnitude is presented.

From these tests, the magnetometer has proven capable of detecting periodic and aperiodic varying magnetic fields. Hence, proximity detection and magnetic signature recognition are both suitable applications for the tilt-compensated magnetometer system.

Chapter 8 Conclusion

A tilt-compensated magnetometer has been designed, tested and analyzed. This integrated system solves the problem of magnetometer inaccuracy when tilting motion is presented. Successful testing has demonstrated the system's ability to significantly more accurately detect steady earth fields, sinusoidal fields, and aperiodic signatures when subjected to tilt motion.

8.1 Summary

Design and construction of a tilting platform designed for accurate roll and pitch motion allowed for repeatable and reliable testing of the system. The platform houses a printed circuit board that includes a digital signal processing microcontroller, a single three-axis magnetometer module, dual inclinometers, and various other necessary peripherals.

Testing of inclinometer sensors utilized to acquire inclination angle readings presented accurate output, but revealed an inverted high frequency impulse-like response component when presented with a large acceleration condition. Various topologies of non-linear median filters were simulated and explored to attenuate this high frequency content. Considering results of simulation and testing, an α-trimmed median filter is implemented in the final design.

Finite impulse response filtering is explored for the magnetic signals. In addition to improving the signal-to-noise ratio of the magnetic signals, the user can easily alter the design of the FIR filter for any subsequent magnetometer task or specification. The FIR filter also complements the median filtering of the inclination signals with similar phase delay.

Necessary for the trigonometric compensation, the CORDIC algorithm is explored as an alternative to traditional calculations of sine and cosine. For the final design, CORDIC was not found to be ideal due to execution speed. However, this fast and accurate method for trigonometric calculation can be the most efficient method if alternative inclinometers that output data in a different format are used in future work.

Testing of the magnetometer output revealed the necessity for a calibration routine. Coefficients are calculated for a method of magnitude correction, significantly improving the accuracy of the tilt-compensated system.

Final testing of the magnetometer system revealed stable readings of earth's magnetic field, and this steady-field data is optionally utilized as a compass system. In addition, varying magnetic fields are accurately and reliably detected in testing, demonstrating the magnetometer's capability to be used for signature or proximity detection.

8.2 Recommended Future Work

Further programming of the magnetometer system to incorporate automatic calibration when powering up drastically reduces time and data required to prepare the system for operation.

Considering nearby magnetic materials and stray or non-uniform fields, calibration routines for simpler preparation and user interfacing within these potentially noisy environments is recommended.

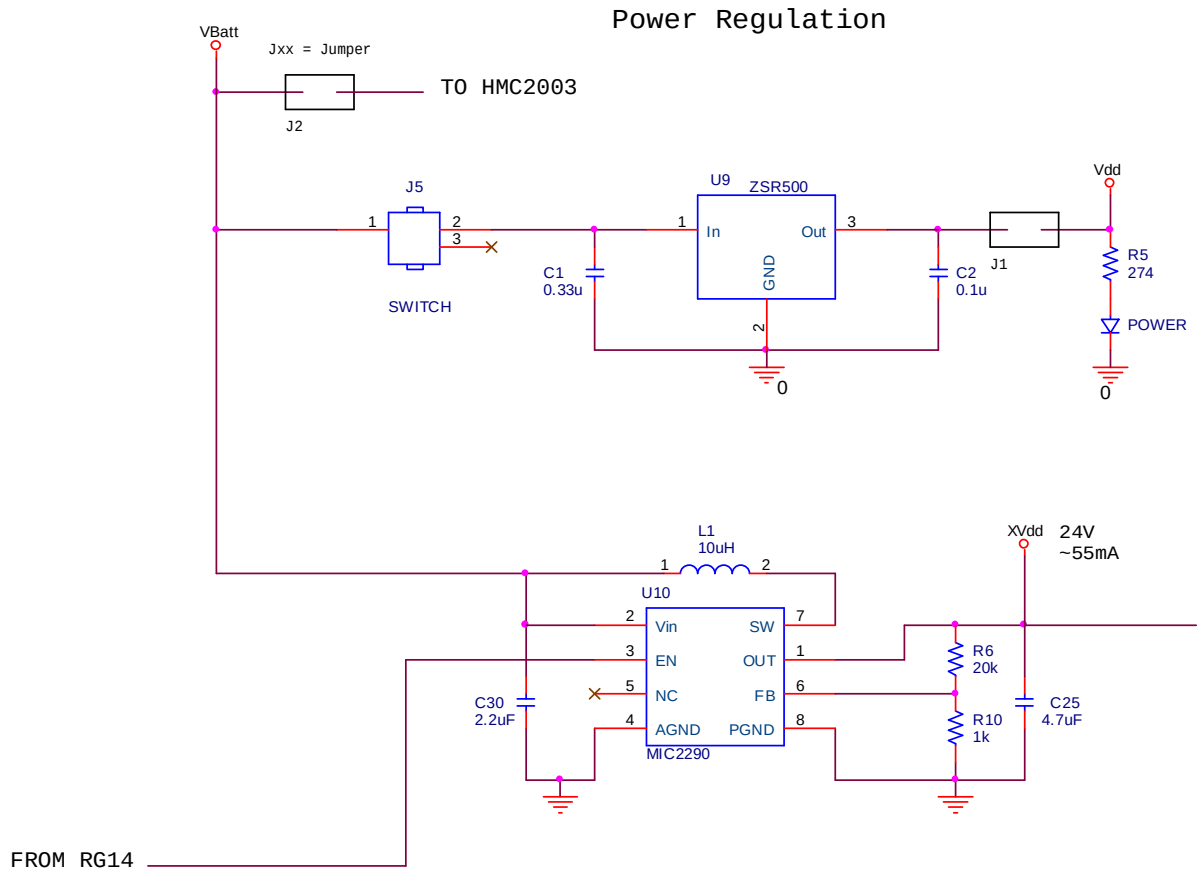
Documented algorithms have been developed that calculate the square-root of an integer using only integer-based mathematics. These algorithms are extremely efficient, and require only a small percentage of the instruction cycles used in this project for the same calculation. Discussed in Section 5.2, improvements in programming of trigonometric calculations via a similar algorithm will drastically improve the bandwidth of this magnetometer system, if such a larger bandwidth is required.

Signature detection for proximity and varying field presence can be optimized with matched filtering and maximum likelihood algorithms. Custom filtering of this nature will allow for task-specialization of the magnetometer system. Efficient programming of these tasks is required to maintain the maximum bandwidth of the system, but this process will greatly improve the versatility and breadth of the tilt-compensated magnetometer system.

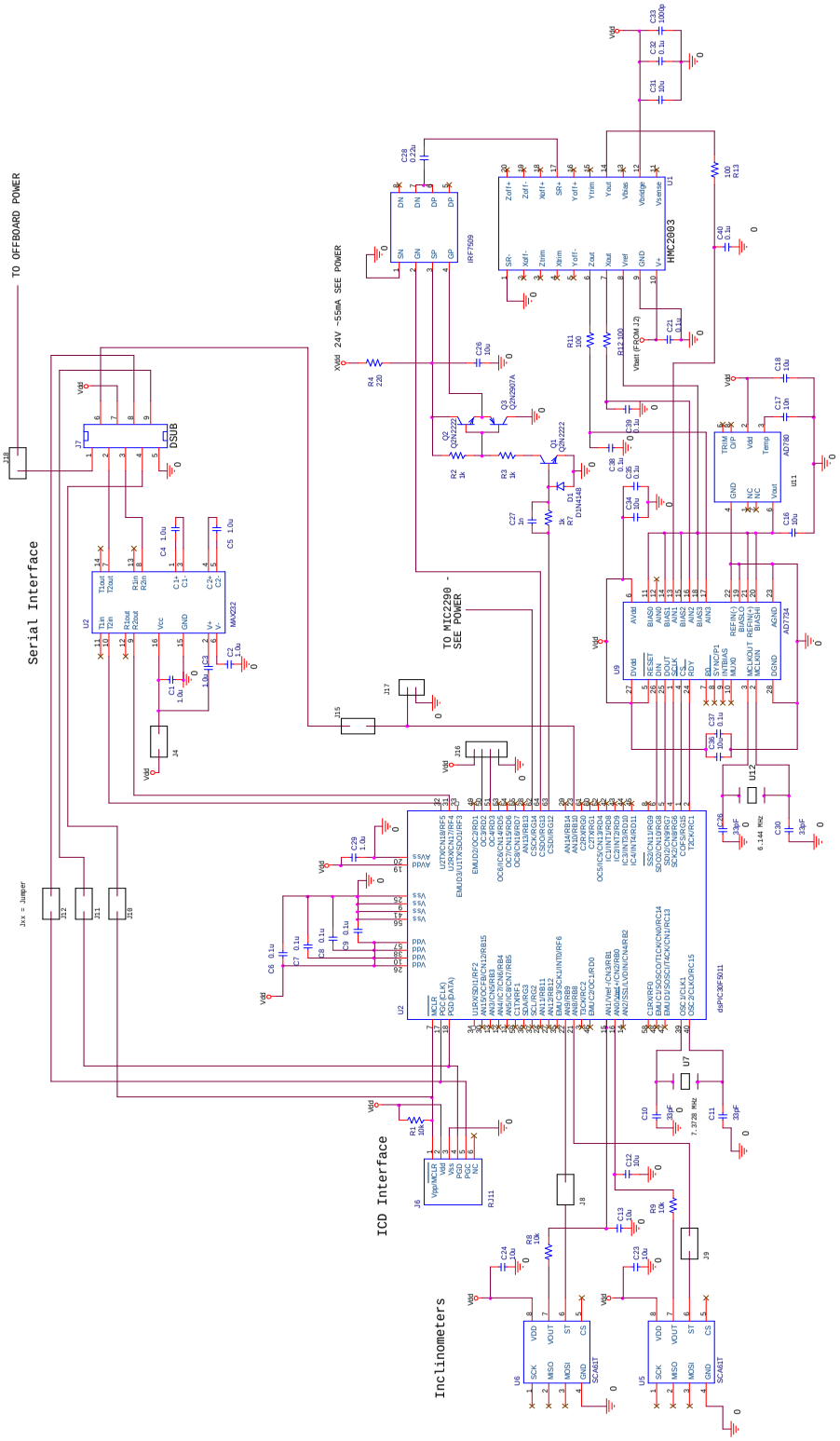
The tilt-compensated magnetometer system has been proven effective at detecting magnetic fields when presented with tilting motion. The tests implemented in this project are performed in uniform dipole fields, or close approximations. In the case where the magnetic field is not uniform, additional calibration and compensation for movement or tilt through this field is necessary to ensure the proper compensation of the magnetic readings.

Appendix A System Schematics

A.1 Power Circuit Schematic



A.2 dsPIC and Peripherals



Appendix B MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       Median_Filt_AlphaTrim.m
%   Author:     Adam Bingaman
%   Date:       1/14/10
%   Purpose:    Simulates alpha-trimmed median filter
%   Notes:      NONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
% Set system parameters
xn = [];            % Input signals
yn = [];
N = 100;             % Length of input window
alpha = 0.10;          % Trim value
alpha = alpha * N;
win_current = zeros(1,N);

for i = 1:length(xn)
    % Load current window
    win_current(N) = xn(i);
    % Find median of window and output
    yn(i) = median(win_current(alpha:(N-alpha+1)));
    % Shift window
    for j = 1:(N-1)
        win_current(j) = win_current(j+1);
    end
en
figure;     % Plot filtered result
plot(xn,'-','Linewidth',2);
grid on;
xlabel('Signal Magnitude');
ylabel('n');
title('Median Filtered Signal');
hold;
plot(yn,'-r','Linewidth',2);
legend('Original Signal','Filtered Signal');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       Median_Filt_FMH_5.m
%   Author:     Adam Bingaman
%   Date:       1/20/10
%   Purpose:    Simulates FIR-Median Hybrid Filters
%   Notes:      NONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
% Set system parameters
xn = [];            % Input signal
yn = [];
N = 41;            %
M = 5;
k_index_mid = (N+1)/2;    % Lower half of window
if( mod((N-1),4) == 0)
    k_inner = (N-1) / 4;
    k_outer = (N-1) / 4;
elseif mod((N-1),2) == 0
    k_inner = (N-3) / 4;
    k_outer = ((N-3) / 4) + 1;
end
win_current = zeros(1,N);
H = [];
for i = 1:length(xn)
    % Load current window
```

```
    win_current(N) = xn(i);
    % FMH filter process
    H(1) = mean(win_current( (k_index_mid + k_inner + 1) : N ) );
    H(2) = mean(win_current( (k_index_mid + 1) : (k_index_mid + k_inner)));
    H(3) = 1;
    H(4) = mean(win_current( (k_outer + 1) : (k_outer + k_inner)));
    H(5) = mean(win_current( 1 : k_outer ));
    % Find median of results
    yn(i) = median(H(1:5));
    % Shift window
    for j = 1:(N-1)
        win_current(j) = win_current(j+1);
    end
end
figure;         % Plot filtered result
plot(xn,'-','Linewidth',2);
grid on;
xlabel('Signal Magnitude');
ylabel('n');
title('FIR-Median Hybrid Filtered Signal with M = 5');
hold;
plot(yn,'-r','Linewidth',2);
legend('Original Signal','Filtered Signal');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       Median_Filt_LUM_Smooth.m
%   Author:     Adam Bingaman
%   Date:       2/7/10
%   Purpose:    This project simulates FIR-Median Hybrid Filters
%   Notes:      NONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
% Set system parameters
xn = [];            % Input signal
yn = [];
N = 75;            % Number of elements in median window
elem_buffer = [];
k = 2;  % Sample index counter for samples
index_mid = (N+1)/2;    % Lower half of window
dwin_current = zeros(1,N);
for i = 1:length(xn)
    % Load current window
    win_current(N) = xn(i);
    elem_buffer(1) = win_current(index_mid - k);
    elem_buffer(2) = win_current(index_mid);
    elem_buffer(3) = win_current(index_mid+ k);
    % Find median of window and output
    yn(i) = median(elem_buffer);
    % Shift window
    for j = 1:(N-1)
        win_current(j) = win_current(j+1);
    end
end
figure;         % Plot filtered result
plot(xn,'-','Linewidth',2);
grid on;
xlabel('Signal Magnitude');
ylabel('n');
title('Smooth LUM Filtered Signal');
hold;
plot(yn,'-r','Linewidth',2);
legend('Original Signal','Filtered Signal');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       Median_Filt.m
%   Author:     Adam Bingaman
%   Date:       7/7/09
%   Purpose:    This project simulates FIR-Median Hybrid Filters
%   Notes:      NONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
% Set system parameters
xn = [];            % Input signal
yn = [];            % Output signal
low_buffer = [];
hi_buffer = [];
N = 75;            % Number of elements in median window
index_mid = (N+1)/2;    % Lower half of window
% 1 <= k <= l <= index_mid
k = 5;             % Spread from center
l = 30;             % Spread from end
win_current = zeros(1,N);
for i = 1:length(xn)
    % Load current window
    win_current(N) = xn(i);

    low_buffer(1) = win_current(index_mid-k);
    low_buffer(2) = win_current(index_mid);
    low_buffer(3) = win_current(l);
    hi_buffer(1) = win_current(index_mid+k);
    hi_buffer(2) = win_current(index_mid);
    hi_buffer(3) = win_current(N-l+1);
    x_low = median(low_buffer);
    x_hi = median(hi_buffer);
    if ((x_low + x_hi) / 2) <= x_low
        yn(i) = x_low;
    else
        yn(i) = x_hi;
    end
    % Shift window
    for j = 1:(N-1)
        win_current(j) = win_current(j+1);
    end
end
figure;         % Plot filtered result
plot(xn,'-','Linewidth',2);
grid on;
xlabel('Signal Magnitude');
ylabel('n');
title('LUM Hybrid Filtered Signal');
hold;
plot(yn,'-r','Linewidth',2);
legend('Original Signal','Filtered Signal');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       FIR_Design
%   Author:     Adam Bingaman
%   Date:       4/21/10
%   Purpose:    This program implements window-based GLP FIR filter design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
ff_hat = 0.25;      % FRACTIONAL FREQUENCY OF BP HIGH SIDE CUTOFF
M = 75; % Length of frequency responses
for k = 1:M
    hBPn(k) =  2*pi*ff_hat/(pi) * sinc(2*ff_hat*(k-(M-1)/2));
end
```

```
% IMPLEMENT KAISER WINDOW FILTER DESIGN
hdn = [];
%win = kaiser(M,5.0);
%win = hanning(M);
%win = rectwin(M);
win = blackman(M);
hn = hBPn .* win';
hn = hn';
% Plot impulse response
figure;
plot(hn,'LineWidth',2);
title('Impulse Response of Blackman Window-Based GLP FIR Filter');
xlabel('n');
ylabel('Magnitude');
grid on;
% Calculate frequency response of system
[filter_RESP(1,:),ff] = freqz(hn,1,2^16,'whole',1);
figure;
subplot(2,1,1);
plot(ff,20*log10(abs(filter_RESP)),'LineWidth',2);
title('Frequency Response of Blackman Window-Based FIR Filter');
xlabel('Fractional Frequency');
ylabel('Magnitude (dB)');
grid on;
subplot(2,1,2);
plot(ff,unwrap(angle(filter_RESP)),'LineWidth',2);
xlabel('Fractional Frequency');
ylabel('Phase (Radians)');
grid on;
figure;
plot(win,'LineWidth',2);
xlabel('n');
ylabel('Magnitude');
title('FIR Window');
grid on;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       CORDIC.m
%   Author:     Adam Bingaman
%   Date:       2/2/10
%   Purpose:    This project implements the CORDIC algorithm to calculate
%                   the sin and cosine of the provided angle theta.
%   Notes:      NONE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
n = 15;                 % Number of iterations, AKA number of bits of accuracy
theta = 84.65;        % Angle in degrees  -90 <= theta <= 90
d_theta = theta;        % Initial rotation angle
theta_track = [];
% Initial coordinates of theta on unit circle
x = 1;
y = 0;
% Plot vectors
x_conv = [];        % Convergent values
y_conv = [];
x_true = [];        % True values
y_true = [];
for i = 1:n
    x_true(i) = cos(theta * pi/180);
    y_true(i) = sin(theta * pi/180);
end
% Create atan table
atan_table = [];
```

```
for i = 1:n
   atan_table(i) = atan(2^-(i-1)) * 180 /pi;
end
% Precompute gain
gain = 1;
for i = 1:n
    gain = gain * (1/sqrt(1+2^-(2*(i-1))));
end
x = x*gain;
% Compute sine and cosine of angle accordingly
for i = 1:n
    % If last iteration resulted in angle LESS than desired angle
    if d_theta > 0
        % Rotate counterclockwise
        xtemp = x - y * (2^-(i-1));
        y = y + (x * 2^-(i-1));
        x = xtemp;
        x_conv(i) = x;
        y_conv(i) = y;
        d_theta = d_theta - atan_table(i);
    % If last iteration resulted in angle MORE than desired angle
    else
        % Rotate clockwise
        xtemp = x + y * (2^-(i-1));
        y = y - (x * 2^-(i-1));
        x = xtemp;
        x_conv(i) = x;
        y_conv(i) = y;
        d_theta = d_theta + atan_table(i);
    end
end
% Plot convergence of both calculations
figure;
plot(x_conv,'r','LineWidth',2);
hold;
plot(x_true,'b','LineWidth',2);
plot(y_conv,'m','LineWidth',2);
plot(y_true,'g','LineWidth',2);
title('Convergence to Sine and Cosine Calculation');
xlabel('n');
ylabel('Result');
legend('Cosine Converging CORDIC','Cosine True Value', 'Sine Converging CORDIC', 'Sine
True Value');
grid on;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       Calibrate_Roll.m
%   Author:     Adam Bingaman
%   Date:       3/29/10
%   Purpose:    This project calibrates the roll angle and then the deltaZ
%                   component of the Y and Z readings of the magnetometer
%   Notes:      None
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
data= [];      % Data of non-calibrated system
%%% Incorporate roll calibration and calibrate deltaZ coefficient for Y %%%
calib_coeff = -1.00:0.01:1.00;
calib_data_deltaY = [];
mean_dev_deltaY = [];
for i = 1:(length(calib_coeff))
   for j = 1:length(data(:,1))
        calib_data_deltaY(j,i) = data(j,2)*cos(data(j,5)*pi/180) +
data(j,3)*sin(data(j,5)*pi/180) + calib_coeff(i)*(data(j,2)-data(1,2));
    end
```

```
end
% Calculate minimum error calibration
for i = 1:(length(calib_coeff))
    mean_dev_deltaY(i) = abs(mean(calib_data_deltaY((1:50),i)) -
mean(calib_data_deltaY((length(calib_data_deltaY(:,1)-
50):length(calib_data_deltaY(:,1))),i)));
end
% Identify calibration coefficient that results in minimum drift
for i = 1:(length(calib_coeff))
    if ((mean_dev_deltaY(i) - min(mean_dev_deltaY)) == 0)
        calib_coeff_deltaY = calib_coeff(i);
        deltaY_index = i;
    end
end
%%% Incorporate roll calibration and calibrate deltaZ coefficient for Y %%%
calib_coeff = -1.00:0.01:1.00;
calib_data_deltaZ = [];
mean_dev_deltaZ = [];
for i = 1:(length(calib_coeff))
   for j = 1:length(data(:,1))
        calib_data_deltaZ(j,i) = data(j,1)*sin(data(j,4)*pi/180) -
data(j,2)*sin(data(j,5)*pi/180)*cos(data(j,4)*pi/180) +
data(j,3)*cos(data(j,4)*pi/180)*cos(data(j,5)*pi/180) + calib_coeff(i)*(data(j,2)-
data(1,2));
    end
end
% Calculate minimum error calibration
for i = 1:(length(calib_coeff))
    mean_dev_deltaZ(i) = abs(mean(calib_data_deltaZ((1:50),i)) -
mean(calib_data_deltaZ((length(calib_data_deltaZ(:,1)-
50):length(calib_data_deltaZ(:,1))),i)));
end
% Identify calibration coefficient that results in minimum drift
for i = 1:(length(calib_coeff))
    if ((mean_dev_deltaZ(i) - min(mean_dev_deltaZ)) == 0)
        calib_coeff_deltaZ = calib_coeff(i);
        deltaZ_index = i;
    end
end
figure;
plot(data(:,1),'Linewidth',2);
hold;
plot(data(:,2),'r','Linewidth',2);
plot(data(:,3),'g','Linewidth',2);
plot(data(:,1),'c','Linewidth',2);
plot(calib_data_deltaY(:,deltaY_index),'m','Linewidth',2);
plot(calib_data_deltaZ(:,deltaZ_index),'y','Linewidth',2);
grid on;
legend('Raw X-Axis Field','Raw Y-Axis Field','Raw Z-Axis Field', 'Calibrated Compensated
X-Axis Field','Calibrated Compensated Y-Axis Field','Calibrated Compensated Z-Axis
Field');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Name:       Calibrate_Pitch.m
%   Author:     Adam Bingaman
%   Date:       3/29/10
%   Purpose:    This project calibrates the roll angle and then the deltaZ
%                   component of the Y and Z readings of the magnetometer
%   Notes:      None
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all;
data= [];      % Data of non-calibrated system
%%% Incorporate roll calibration and calibrate deltaZ coefficient for Y %%%
calib_coeff = -1.00:0.01:1.00;
```

```
calib_data_deltaX = [];
mean_dev_deltaX = [];
for i = 1:(length(calib_coeff))
   for j = 1:length(data(:,1))
        calib_data_deltaX(j,i) = data(j,1)*cos(data(j,4)*pi/180) +
data(j,2)*sin(data(j,5)*pi/180)*sin(data(j,4)*pi/180) -
data(j,3)*cos(data(j,5)*pi/180)*sin(data(j,4)*pi/180) + calib_coeff(i)*(data(j,1)-
data(1,1));
    end
end
% Calculate minimum error calibration
for i = 1:(length(calib_coeff))
    mean_dev_deltaX(i) = abs(mean(calib_data_deltaX((1:50),i)) -
mean(calib_data_deltaX((length(calib_data_deltaX(:,1)-
50):length(calib_data_deltaX(:,1))),i)));
end
% Identify calibration coefficient that results in minimum drift
for i = 1:(length(calib_coeff))
    if ((mean_dev_deltaX(i) - min(mean_dev_deltaX)) == 0)
        calib_coeff_deltaX = calib_coeff(i);
        deltaX_index = i;
    end
end
%%% Incorporate roll calibration and calibrate deltaZ coefficient for Y %%%
calib_coeff = -1.00:0.01:1.00;
calib_data_deltaZ = [];
mean_dev_deltaZ = [];
for i = 1:(length(calib_coeff))
   for j = 1:length(data(:,1))
        calib_data_deltaZ(j,i) = data(j,1)*sin(data(j,4)*pi/180) -
data(j,2)*sin(data(j,5)*pi/180)*cos(data(j,4)*pi/180) +
data(j,3)*cos(data(j,4)*pi/180)*cos(data(j,5)*pi/180) + calib_coeff(i)*(data(j,1)-
data(1,1));
    end
end
% Calculate minimum error calibration
for i = 1:(length(calib_coeff))
    mean_dev_deltaZ(i) = abs(mean(calib_data_deltaZ((1:50),i)) -
mean(calib_data_deltaZ((length(calib_data_deltaZ(:,1)-
50):length(calib_data_deltaZ(:,1))),i)));
end
% Identify calibration coefficient that results in minimum drift
for i = 1:(length(calib_coeff))
    if ((mean_dev_deltaZ(i) - min(mean_dev_deltaZ)) == 0)
        calib_coeff_deltaZ = calib_coeff(i);
        deltaZ_index = i;
    end
end
figure;
plot(data(:,1),'Linewidth',2);
hold;
plot(data(:,2),'r','Linewidth',2);
plot(data(:,3),'g','Linewidth',2);
plot(calib_data_deltaX(:,deltaX_index),'c','Linewidth',2);
plot(data(:,2),'m','Linewidth',2);
plot(calib_data_deltaZ(:,deltaZ_index),'y','Linewidth',2);
grid on;
legend('Raw X-Axis Field','Raw Y-Axis Field','Raw Z-Axis Field', 'Calibrated Compensated
X-Axis Field','Calibrated Compensated Y-Axis Field','Calibrated Compensated Z-Axis
Field');
```

Appendix C C-Programming Software Code

```
/*****************************************************************************************************
*****************************************************************************************************
*
*                   Author                              :                   Adam Bingaman
*                   Company                             :                   NanoSonic Inc.
*                   Filename                  :                 dsPIC_Mag_Inc_2.5.c
*                   Date                                :                   4/2010
*
*                   Purpose:  This file interfaces the SCA61T inclinometers, outputting two 12-bit
*                                       readings padded to 16-bits.
*
*                   Versions: 2.2       Combined magnetometer and inclinometer acquisition algorithms
*                                       2.3 Implemented calibration of delta components
*                                       2.4 Incorporate FIR and median filtering
*                                       2.5 Added floating point square root calculation of trig
*****************************************************************************************************
*****************************************************************************************************/
#include <p30f5011.h>
#include <stdio.h>
#include <math.h>
#include <libpic30.h>
#include "timer.h"
#include "math.h"
#include "uart.h"
#include "AD7734_16bit.h"
#include "dsp.h"
#include "FIR_Filter.h"

_FOSC(CSW_FSCM_OFF & XT_PLL16);                 // Set PLL
_FWDT(WDT_OFF);
_FBORPOR(PBOR_OFF & BORV_20 & PWRT_64 & MCLR_EN);
_FGS(CODE_PROT_OFF);

#define XTFREQ          6144000                 // On-board Crystal frequency
#define PLLMODE         16                              // On-chip PLL setting
#define FCY             XTFREQ*PLLMODE/4                // Instruction Cycle Frequency
#define BAUDRATE        115200                          // Desired Baud Rate
#define BRGVAL          ((FCY/BAUDRATE)/16)-1           // Formula for U1BRG register
#define MILLISEC                FCY/1000        // 1 millisecond worth of instructions

// ###################### PROGRAM CONTROL VARIABLES ############################
// ----- General system control -----
long F_SAMP = 250;                      // System sampling frequency
#define N25                             // Number of samples in tilt (median) and magnetic (FIR) filters
fractional pitch_angle_win[N];          // Window N samples wide of tilt readings
fractional roll_angle_win[N]; // Window N samples wide of tilt readings

// ----- CORDIC -----
fractional roll_sin_cos[2] = {0x0000, 0x0000};          // Sine and cosine values (respectfully) of roll angle
fractional pitch_sin_cos[2] = {0x0000, 0x0000};         // Sine and cosine values (respectfully) of roll angle
extern const fractional atan_table[32];                         // Arctangent lookup table for CORDIC

// ----- Magnetic vectors -----
fractional H_base_xyz[3] = {0x0000, 0x0000, 0x0000};    // Baseline magnetic readings
fractional          H_raw_xyz[6] = {0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000};        // Raw magnetic readings [ X Y Z
delta_X delta_Y delta_Z]
fractional H_comp_xyz[3] = {0x0000, 0x0000, 0x0000};    // Compensated magnetic readings
```

```
// ----- Calibration -----
float rho_XdX = -0.1700;
float rho_ZdX = -0.0500;
float rho_YdY = -0.1600;
float rho_ZdY = -0.0200;
//                [   1      1     -1    rho_XdX -rho_ZdX    0  ]
fractional x_rho[6] = {0x7FFF, 0x7FFF, 0x8001, 0x0000, 0x0000, 0x0000};
//                [   0      1      1      0   rho_YdY     0  ]
fractional y_rho[6] = {0x0000, 0x7FFF, 0x7FFF, 0x0000, 0x0000, 0x0000};
//                [   1     -1      1    rho_ZdX, rho_ZdY,  0  ]
fractional z_rho[6] = {0x7FFF, 0x8001, 0x7FFF, 0x0000, 0x0000, 0x0000};

// ----- FIR Filter -----
// Align coefficients memory at end of Y memory - Y memory occupies 0x1000 to 0x17FF
fractional _YDATA(0x0080) FIR_Delay_X[FIR_FILTER_NCOEFFS];          // Align complex signal memory at end of Y memory
fractional _YDATA(0x0080) FIR_Delay_Y[FIR_FILTER_NCOEFFS];          // Align complex signal memory at end of Y memory
fractional _YDATA(0x0080) FIR_Delay_Z[FIR_FILTER_NCOEFFS];          // Align complex signal memory at end of Y memory

/* Global Definitions */
extern const fractional FIR_Coeffs[FIR_FILTER_NCOEFFS*2]        /* IIR coefficients array in Program memory */
__attribute__ ((space(auto_psv), aligned (256)));

/* Extern definitions for the imported *.s files from dsPIC Filter Design */
extern FIRStruct FIR_PSV;  /*Contains filter structures for FIR-LPF*/

FIRStruct filter_X, filter_Y, filter_Z;

// ----- Alpha - Trimmed Median Filter -----
#define    median_N                                25          // Window length of median filter - odd number
#define    alpha                                   3           // Number of samples to trim from both "ends" of window
fractional vect_roll_buff[median_N-(2*alpha)];
fractional vect_pitch_buff[median_N-(2*alpha)];
fractional roll_filt, pitch_filt;

// ----- FMH Filter -----
// #define             FMH_N                          73          // Window length of FMH filter - odd number
// unsigned int FMH_k_inner, FMH_index_mid, FMH_k_outer;        // Size of internal and external buffers and middle index
// fractional _XDATA(0x0080)          FMH_coeffs_inner[(FMH_N-1)/2];      /* FIR coefficients array in X memory */
// fractional _XDATA(0x0080)          FMH_coeffs_outer[(FMH_N-1)/2];      /* FIR coefficients array in X memory */
// fractional ang_x_win[FMH_N];
// fractional ang_y_win[FMH_N];
// fractional vect_x_buff[FMH_N];
// fractional vect_y_buff[FMH_N];

// ----- LUM filter specifications -----
// #define             LUM_N                          49          // Window length of FLUM filter - odd number
// #define             LUM_spread                     5           // Number of samples from center of LUM smoother
// fractional ang_x_win[LUM_N];
// fractional ang_y_win[LUM_N];
// ##############################################################################

void _HMC100X_Reset(void);
void _HMC100X_Set(void);
void _ADC_Init(void);
void _Trans_Data_16bit(unsigned int data);
void _UART_Init(void);
void _SPI_Init(void);                                           // Initialize SPI module for communication
void _SPI_Data_Out(int data_out);                               // Transmits integer through SPI
int _SPI_Data_In(void);                                         // Receives data in receive module
void _Timer32_Init(void);
void _Trig(fractional theta, fractional* sin_cos[]);
fractional _Vector_Median(fractional vector_in[], unsigned int numSamps);
void _Vector_Shift(fractional vector_in[], unsigned int numSamps);
```

```
fractional _Int2Fract(unsigned int data);
void _Tilt_Comp(fractional* field_raw[], fractional* field_comp[], fractional* pitch_ang[], fractional* roll_ang[]);
void _Tilt_Comp_Float(fractional* field_raw[], fractional* field_comp[], fractional* pitch_ang[], fractional* roll_ang[]);
void _Filter_Define_X(void);
void _Filter_Define_Y(void);
void _Filter_Define_Z(void);
void _FMH_Init(void);                                                   // Initialize FMH filter coefficients and sections
fractional _FMH_Filter(fractional vector_in[]);                         // FMH filter function set up by _FMH_Init
fractional _LUM_Filter(fractional vector_in[], unsigned int numSamps);  // LUM smoother filter functions

int main()
{
        fractional pitch_angle, roll_angle;                     // Angle of inclination in degrees
        fractional base_pitch, base_roll, in_pitch, in_roll;    // Angles of inclination BASELINE
        fractional pitch_filt, roll_filt;
        fractional temp;
        fractional azimuth;
        int i;

        TRISB = 0x0003;                                         // AN0 and AN1 inputs
        TRISBbits.TRISB10 = 0;                          // Set as output
        LATBbits.LATB10 = 0;
        TRISD = 0x0000;
        TRISG = 0x0000;
        LATG = 0x0000;
        LATGbits.LATG12 = 1;    // Initial RESET condition for pulse generation
        LATGbits.LATG13 = 0;    // Initial RESET condition for pulse generation
        _Timer32_Init();        // Initialize timer
        _UART_Init();           // Initialize UART
        _ADC_Init();            // Initialize ADC for scanning
        _SPI_Init();            // Initialize SPI

        /* FILTER OPERATIONS */
        _Filter_Define_X();
        _Filter_Define_Y();
        _Filter_Define_Z();
        FIRDelayInit(&filter_X);
        FIRDelayInit(&filter_Y);
        FIRDelayInit(&filter_Z);

        // Initialize vectors
        x_rho[3] = Float2Fract(rho_XdX);
        x_rho[4] = Float2Fract(-rho_ZdY);
        y_rho[4] = Float2Fract(rho_YdY);
        z_rho[3] = Float2Fract(rho_ZdX);
        z_rho[4] = Float2Fract(rho_ZdY);

        LATGbits.LATG14 = 1;            // Turn ON boost converter
        __delay32(MILLISEC*250);
        _HMC100X_Reset();               // SET pulse for magnetic sensitivity
        _HMC100X_Set();                 // RESET pulse for magnetic sensitivity
        LATGbits.LATG14 = 0;            // Turn OFF boost converter
        __delay32(MILLISEC*250);

        // AD7734 initialization routines
        _AD7734_Reset();                // Reset AD7734 ADC to default
        _AD7734_Init_Ch1();             // Setup Channel 1
        _AD7734_Init_Ch2();             // Setup Channel 2
        _AD7734_Init_Ch3();             // Setup Channel 3

        ADCON1bits.ADON = 1;            // Turn on ADC
        T2CONbits.TON=1;                // Turn on timer
        // Acquire initial offset voltage of inclinometers
```

```
		while( !_T3IF)			// Wait for T3 interrupt to flag
		{}
		IFS0bits.T3IF = 0;		// Clear interrupt flag after flag detection
		ADCON1bits.ASAM = 1;		// Start sampling - Internal counter ENDS SAMPLING
		while(!ADCON1bits.DONE){}	// Wait for conversion to complete
		// Acquire offset value of inclinometer
		base_pitch = ~ADCBUF1+1;
		base_roll = ~ADCBUF0+1;
		// Acquire baseline magnetic field for error calculation
		H_base_xyz[0] = _Int2Fract(_AD7734_Samp_Ch2());
		H_base_xyz[1] = _Int2Fract(~_AD7734_Samp_Ch1()+1);		// Negated to match documented convention

		H_base_xyz[2] = _Int2Fract(_AD7734_Samp_Ch3());		// Negated to match documented convention

// Real time system operation
   while(1)
   {
		// Timer trigger sampling of inclination
		while( !_T3IF)			// Wait for T3 interrupt to flag
		{}
		IFS0bits.T3IF = 0;		// Clear interrupt flag after flag detection
		ADCON1bits.ASAM = 1;		// Start sampling - Internal counter ENDS SAMPLING
		while(!ADCON1bits.DONE){}	// Wait for conversion to complete

		// Acquire inclination in fractional format
		pitch_angle_win[N-1] = ~ADCBUF1 + 1; // Acquire x tilt reading - Negated to match documented convention
		roll_angle_win[N-1] = ~ADCBUF0 + 1;   // Acquire y tilt reading - Negated to match documented convention
		// Acquire X axis reading
		temp = _Int2Fract(_AD7734_Samp_Ch2());
		FIR(1,&H_raw_xyz[0],&temp,&filter_X);
		// Acquire Y axis reading
		temp = _Int2Fract(~_AD7734_Samp_Ch1()+1);
		FIR(1,&H_raw_xyz[1],&temp,&filter_Y);
		// Acquire Z axis reading
		temp = _Int2Fract(_AD7734_Samp_Ch3());
		FIR(1,&H_raw_xyz[2],&temp,&filter_Z);
		H_raw_xyz[3] = H_raw_xyz[0] - H_base_xyz[0];		// delta_X
		H_raw_xyz[4] = H_raw_xyz[1] - H_base_xyz[1];		// delta_Y
		H_raw_xyz[5] = H_raw_xyz[2] - H_base_xyz[2];		// delta_Z

		// Fill hold buffers with values for filtering
		VectorCopy(median_N-(2*alpha), &vect_roll_buff[0], &roll_angle_win[alpha]);
		VectorCopy(median_N-(2*alpha), &vect_pitch_buff[0], &pitch_angle_win[alpha]);

		// Filter angle values
		roll_filt = _Vector_Median(vect_roll_buff, median_N-(2*alpha));
		pitch_filt = _Vector_Median(vect_pitch_buff, median_N-(2*alpha));

		// Filter angle values
		// roll_filt = _FMH_Filter(vect_roll_buff);
		// pitch_filt = _FMH_Filter(vect_pitch_buff);

		// Filter angle values
		// roll_filt = _LUM_Filter(ang_roll_win, LUM_N);
		// pitch_filt = _LUM_Filter(ang_pitch_win, LUM_N);

		// Angle of tilt sensors in degrees
		 pitch_angle = (pitch_filt- base_pitch)  << 1;
		roll_angle = (roll_filt - base_roll) << 1;

		// Voltage reading of filtered inclination
		// Vin_pitch = (Fract2Float(pitch_filt - base_pitch))* 1.25;  // Voltage reading
		// Vin_roll =  (Fract2Float(roll_filt - base_roll))* 1.25;        //  Voltage reading
```

```
                    // Angle of tilt sensors in degrees
                    // pitch_angle_deg = asin(Vin_pitch) * 180.0/PI;  // FLOAT
                    // roll_angle_deg = asin(Vin_roll) * 180.0/PI;      // FLOAT

                    // Calculate sine and cosine values
                    _Trig(pitch_angle, &pitch_sin_cos[0]);
                    _Trig(roll_angle, &roll_sin_cos[0]);

                    // _CORDIC_16bit(pitch_angle_deg, pitch_sin_cos);
                    // _CORDIC_16bit(roll_angle_deg, roll_sin_cos);

                    _Tilt_Comp_Cal(&H_raw_xyz[0], &H_comp_xyz[0], &pitch_sin_cos[0], &roll_sin_cos[0]);

                    // temp = (Fract2Float(H_comp[1])  / Fract2Float(H_comp[0]) );
                    //azimuth = Float2Fract(temp0);

                    _Vector_Shift(pitch_angle_win, N-1);                    // Delay angle buffer one sample
                    _Vector_Shift(roll_angle_win, N-1);                     // Delay angle buffer one sample

            }
            while(1){}
            return 0;
}

/*****************************************************
Name:       _Trig
Purpose:    Calculate sine and cosine of given angle
Type:       void
Inputs:     fractional theta : Angle to calculate trig
                                fractional[] : Result vector holding sine and cosine, respectfully
Outputs:    NONE
References: NONE
Notes:      Normalizes readings with sine multiplication by 0.625
*****************************************************/
void _Trig(fractional theta, fractional* sin_cos[])
{
            float temp_0, temp_1;
            sin_cos[0] = VectorScale(1, &theta, &sin_cos[0], 0x4FFF);           // Normalize to voltage level reading
            temp_0 = Fract2Float(VectorPower(1, &sin_cos[0]));                  // Square the sine component
            temp_1 = sqrt(1.000 - temp_0);                      // sqrt (1-sin^2(x))
            sin_cos[1] = Float2Fract(temp_1);

}

/*****************************************************
Name:               _Vector_Median
Purpose:  Inject initial values in median filter windows (X & Y)
Type:               fractional
Inputs:             Vector to find median
Outputs:  Median of vector
References:         spi.h
                                dsPIC30F Family Reference Manual
Notes:              # instruction cycles for VectorMax = 19+7(N-2)
*****************************************************/
fractional _Vector_Median(fractional vector_in[], unsigned int numSamps)
{
            unsigned int k;                                 // Counter
            fractional temp[65];                    // Value of 64 is maximum value that can be evaluated
            unsigned int maxIndex;                  // Index of maximum vector value
            unsigned int midIndex;                  // Middle index of vector
            midIndex = (numSamps - 1) / 2;          // Calculate middle index of vector
            // Sort vector
```

```
	for(k = 0 ; k < numSamps ; k++)
	{
		temp[k] = VectorMax(numSamps, &vector_in[0], &maxIndex);
		vector_in[maxIndex] = 0x8001;			// Reduce to -1 for minimum ignorance
	}
	return temp[midIndex];
}

/*****************************************************
Name:		_Vector_Shift
Purpose:	Time delays both X and Y axis tilt information
Type:		void
Inputs:		array to be shifted, number of elemts in array minus one = numSamps
Outputs:	NONE
Notes:		N/A
*****************************************************/
void _Vector_Shift(fractional vector_in[], unsigned int numSamps)
{
	VectorCopy(numSamps, &vector_in[0], &vector_in[1]);
}

/*****************************************************
Name:		_Int2Fract
Purpose:	Converts from unsigned integer to fractional format 1.15
Type:		void
Inputs:		Data to be converted in unsigned int form
Outputs:	Fractional representation
Notes:		0x0000 to 0xFFFF unsigned int
			to
		0x8001 to 0x7FFF fractional
*****************************************************/
fractional _Int2Fract(unsigned int data)
{
	if(data == 0xFFFF)
		data = 0x7FFF;
	else
		data += 0x8001;
	return data;
}


/*****************************************************
Name:		_Tilt_Comp_Cal
Purpose:	Tilt-compensates with calibration coefficients
Type:		void
Inputs:		Pointers to magnetic vectors and angle vectors
Outputs:	NONE
Notes:
*****************************************************/
void _Tilt_Comp_Cal(fractional* field_raw[], fractional* field_comp[], fractional* pitch_ang[], fractional* roll_ang[])
{
	// Set up vectors for multiplication
	fractional x_pitch[6] = {pitch_ang[1], pitch_ang[0], pitch_ang[0], 0x7FFF, pitch_ang[0], 0x7FFF};
	fractional x_roll[6] = {0x7FFF, roll_ang[0], roll_ang[1], 0x7FFF, 0x7FFF, 0x7FFF};
	fractional y_roll[6] = {0x7FFF, roll_ang[1], roll_ang[0], 0x7FFF, 0x7FFF, 0x7FFF};
	fractional z_pitch[6] = {pitch_ang[0], pitch_ang[1], pitch_ang[1], 0x7FFF, pitch_ang[1], 0x7FFF};
	fractional z_roll[6] = {0x7FFF, roll_ang[0], roll_ang[1], 0x7FFF, 0x7FFF, 0x7FFF};
	// Arithmetic buffer
	fractional temp[6] = {0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000};
	// Transform to horizontal axis
	// X-axis
	VectorMultiply(6, &temp[0], &field_raw[0], &x_pitch[0]); // X pitch
	VectorMultiply(6, &temp[0], &temp[0], &x_roll[0]);		// X pitch
```

```
	field_comp[0] = VectorDotProduct(6, &temp[0], &x_rho[0]);		// Sum X, roll and pitch components
	// Y-axis
	VectorMultiply(6, &temp[0], &field_raw[0], &y_roll[0]);		// Y roll
	field_comp[1] = VectorDotProduct(6, &temp[0], &y_rho[0]);		// Sum Y, roll and pitch components
	// Z-axis
	VectorMultiply(6, &temp[0], &field_raw[0], &z_pitch[0]); // Z pitch
	VectorMultiply(6, &temp[0], &temp[0], &z_roll[0]);		// Z pitch
	field_comp[2] = VectorDotProduct(6, &temp[0], &z_rho[0]);		// Sum X, roll and pitch components
}

// Define X-axis FIR filter
void _Filter_Define_X(void)
{
	filter_X.numCoeffs = FIR_FILTER_NCOEFFS;
	filter_X.coeffsBase = __builtin_psvoffset(&FIR_Coeffs[0]);
	filter_X.coeffsEnd = __builtin_psvoffset(&FIR_Coeffs[0])+FIR_FILTER_NCOEFFS*2-1;
	filter_X.coeffsPage = __builtin_psvpage(&FIR_Coeffs[0]);
	filter_X.delayBase = &FIR_Delay_X[0];
	filter_X.delayEnd = &FIR_Delay_X[0]+FIR_FILTER_NCOEFFS-1;
	filter_X.delay = &FIR_Delay_X[0];
};
//Define Y-axis FIR filter
void _Filter_Define_Y(void)
{
	filter_Y.numCoeffs = FIR_FILTER_NCOEFFS;
	filter_Y.coeffsBase = __builtin_psvoffset(&FIR_Coeffs[0]);
	filter_Y.coeffsEnd = __builtin_psvoffset(&FIR_Coeffs[0])+FIR_FILTER_NCOEFFS*2-1;
	filter_Y.coeffsPage = __builtin_psvpage(&FIR_Coeffs[0]);
	filter_Y.delayBase = &FIR_Delay_Y[0];
	filter_Y.delayEnd = &FIR_Delay_Y[0]+FIR_FILTER_NCOEFFS-1;
	filter_Y.delay = &FIR_Delay_Y[0];
};
// Define Z-axis FIR filter
void _Filter_Define_Z(void)
{
	filter_Z.numCoeffs = FIR_FILTER_NCOEFFS;
	filter_Z.coeffsBase = __builtin_psvoffset(&FIR_Coeffs[0]);
	filter_Z.coeffsEnd = __builtin_psvoffset(&FIR_Coeffs[0])+FIR_FILTER_NCOEFFS*2-1;
	filter_Z.coeffsPage = __builtin_psvpage(&FIR_Coeffs[0]);
	filter_Z.delayBase = &FIR_Delay_Z[0];
	filter_Z.delayEnd = &FIR_Delay_Z[0]+FIR_FILTER_NCOEFFS-1;
	filter_Z.delay = &FIR_Delay_Z[0];
};

/*****************************************************
Name:		_FMH_Init
Purpose:	Initializes FHM filter coefficients and number of taps per FIR
Type:		void
Inputs:		NONE
Outputs:	NONE
Notes:		N/A
*****************************************************/
void _FMH_Init(void)
{
	int i;

	if( ((FMH_N-1) % 4) == 0 )		// All four FIR sections same length
	{
		FMH_index_mid = (FMH_N-1)/2;		// N middle index
		FMH_k_inner = FMH_index_mid / 2;	// Number of samples in inner FIR sections
		FMH_k_outer = FMH_index_mid / 2;	// Number of samples in outer FIR sections
	}
	else if( ((FMH_N-1) % 2) == 0 )		// Else if inner and outer will be of different length
```

```
	{
		FMH_index_mid = (FMH_N-1)/2;		// N middle index
		FMH_k_inner = (FMH_index_mid - 1) / 2;	// Number of samples in inner FIR sections
		FMH_k_outer = (FMH_index_mid + 1) / 2;	// Number of samples in outer FIR sections
	}
	for(i = 0 ; i < FMH_k_inner ; i++)
		FMH_coeffs_inner[i] = 0x7FFF / FMH_k_inner;
	for(i = 0 ; i < FMH_k_outer ; i++)
		FMH_coeffs_outer[i] = 0x7FFF / FMH_k_outer;
}

/*****************************************************
Name:		_FMH_Filter
Purpose:	Implements FMH filter with M=5 and any size N (assuming odd) using DSP functions
Type:		Fractional
Inputs:		Input vector containing time samples of length N
Outputs:	Output of FMH filter
Notes:		See _FMH_Init for setup
*****************************************************/
fractional _FMH_Filter(fractional vector_in[])
{
	int i;
	fractional y_buff[5];			// Buffer holding section results
	fractional out;				// Return value of filter

	// Calculate mean of each section and store in buffer
	y_buff[0] = VectorDotProduct(FMH_k_outer, &FMH_coeffs_outer, &vector_in[0]);
	y_buff[1] = VectorDotProduct(FMH_k_inner, &FMH_coeffs_inner, &vector_in[FMH_k_outer]);
	y_buff[2] = 0x0000;					// Center delay perpetually expected mean
	y_buff[3] = VectorDotProduct(FMH_k_inner, &FMH_coeffs_inner, &vector_in[FMH_index_mid+1]);
	y_buff[4] = VectorDotProduct(FMH_k_outer, &FMH_coeffs_outer, &vector_in[FMH_index_mid+FMH_k_inner+1]);
	// Calculate median of result
	out = _Vector_Median(y_buff, 5);
	return out;
}

/*****************************************************
Name:		_LUM_Filter
Purpose:	Implements FLUM filter with any size N (assuming odd)
Type:		Fractional
Inputs:		Input vector containing time samples of length N
Outputs:	Output of LUM filter
Notes:		NONE
*****************************************************/
fractional _LUM_Filter(fractional vector_in[], unsigned int numSamps)
{
	int i;
	fractional y_buff[3];			// Buffer holding section results
	fractional out;				// Return value of filter

	int LUM_midIndex = (numSamps - 1) / 2;
	// Calculate mean of each section and store in buffer
	y_buff[0] = vector_in[LUM_midIndex - LUM_spread];
	y_buff[1] = vector_in[LUM_midIndex];
	y_buff[2] = vector_in[LUM_midIndex + LUM_spread];
	// Calculate median of result
	out = _Vector_Median(y_buff, 3);
	return out;
}
```

References

[1] E. Laulainen, L. Koskinen, M. Kosunen *et al.*, "Compass tilt compensation algorithm using CORDIC." *ISCAS Proceedings*, 2008, pp. 1188-1191.

[2] Honeywell International, "1- and 2-Axis Magnetic Sensors," HMC1001 & HMC1002 datasheet, 2000, http://www.magneticsensors.com/datasheets/hmc1001-2_1021-2.pdf

[3] Honeywell International, "HMC2003 Three-Axis Magnetic Sensor Hybrid," HMC2003 datasheet, 2004, http://www.magneticsensors.com/datasheets/hmc2003.pdf

[4] P. A. Holman, *Magnetoresistance Transducers and How to Use Them as Sensors*, First ed.: Honeywell International, 2004.

[5] M. J. Caruso, Bratland, T., Smith, C.H., Schneider, R., "A new perspective on magnetic field sensing," *Sensors*, pp. 34-46, December, 1998

[6] Honeywell International, "Set/Reset Function for Magnetic Sensors," Application Note 213, 2002, www.ssec.honeywell.com

[7] VTI Technologies, "SCA61T Inclinometer Series," SCA61T datasheet, 2006, http://www.vti.fi/midcom-serveattachmentguid-2b9fadebe6fa4a0c24f144cd55fda22d/SCA61T_inclinometer_datasheet_8261900A.pdf

[8] Honeywell International, "Applications of magnetic sensors for low cost compass systems," Application Note 211, 2002 www.ssec.honeywell.com

[9] M. J. Caruso, and L. S. Withanawasam, "Vehicle detection and compass applications using AMR magnetic sensors," May, 1999.

[10] M. J. Caruso, "Applications of magnetoresistive sensors in navigation systems," *Sensors and Actuators*, vol. SAE SP-1220, pp. 15-21, February, 1998.

[11] N. S. Nise, *Control Systems Engineering*, Fourth ed., Hoboken, New Jersey: John Wiley and Sons, 2004.

[12] B. P. Lathi, *Principles of Linear Systems and Signals*, Second ed., New York, New York: Oxford Press, 2005.

[13] J. G. Proakis, and D. G. Manolakis, *Digital Signal Processing - Principles, Algorithms and Applications*, Fourth ed., Upper Saddle River, New Jersey: Pearson Prentice Hall, 2007.

[14] N. Gallagher, Jr., and G. Wise, “A theoretical analysis of the properties of median filters,” *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 29, no. 6, pp. 1136-1141, 1981.

[15] J. W. Tukey, *Exploratory Data Analysis*, Reading, Massachusetts: Addison-Wesley Publishing Company, 1977.

[16] N. C. Gallagher, Jr., "Median filters: a tutorial." pp. 1737-1744 vol.2.

[17] G. Arce, and M. McLoughlin, “Theoretical analysis of the max/Median filter,” *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 35, no. 1, pp. 60-69, 1987.

[18] P. Heinonen, and Y. Neuvo, “FIR-median hybrid filters with predictive FIR substructures,” *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 36, no. 6, pp. 892-899, 1988.

[19] P. Heinonen, and Y. Neuvo, "Smoothed median filters with FIR substructures." *ISCAS Proceedings*, 1991, pp. 49-52.

[20] P. Heinonen, and Y. Neuvo, “FIR-median hybrid filters,” *Acoustics, Speech and Signal Processing, IEEE Transactions on,* vol. 35, no. 6, pp. 832-838, 1987.

[21] R. C. Hardie, and C. Boncelet, “LUM filters: a class of rank-order-based filters for smoothing and sharpening,” *Signal Processing, IEEE Transactions on,* vol. 41, no. 3, pp. 1061-1076, 1993.

[22] X. Li, R. Kang, X. Shu *et al.*, "Tilt-Induced-Error Compensation for 2-Axis Magnetic Compass with 2-Axis Accelerometer." *WRI World Congress on Computer Science and Information Engineering*, 2009, pp. 122-125.

[23] S. Y. Cho, “Enhanced tilt compensation method for biaxial magnetic compass,” *Electronics Letters*, vol. 41, no. 24, pp. 1323-1325, 2005.

[24] G. Basile, M. Pierantoni, S. Pirani *et al.*, "Attitude compensated electronic compass for aircraft navigation." pp. 126-131.

[25] J. E. Volder, “The CORDIC Trigonometric Computing Technique,” *Electronic Computers, IEEE Transactions on,* vol. EC-8, no. 3, pp. 330-334, 1959.

[26] J. S. Walther, “A unified algorithm for elementary functions,” *Proc. Spring Joint Computer Conf.*, pp. 379-385, 1971.

[27] T. Lang, and E. Antelo, “CORDIC vectoring with arbitrary target value,” *Computers, IEEE Transactions on,* vol. 47, no. 7, pp. 736-749, 1998.

[28] N. Takagi, T. Asada, and S. Yajima, “Redundant CORDIC methods with a constant scale factor for sine and cosine computation,” *Computers, IEEE Transactions on,* vol. 40, no. 9, pp. 989-995, 1991.

[29] K. Turkowski, “Fixed-Point Trigonometry with CORDIC Iterations,” Apple Computers, Application Note, 1990.

[30] K. Maharatna, J. Valls, J. Tso-Bing *et al.*, “50 Years of CORDIC: Algorithms, Architectures, and Applications,” *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 56, no. 9, pp. 1893-1907, 2009.

[31] S. W. Smith, *Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed., San Diego: California Technical Publishing, 1999.

[32] S. A. Macintyre, "chapter Magnetic Field Measurement," *Measurement, Instrumentation and Sensor Handbook*: CRC Press LLC, 1999.

[33] H. D. Young, R. A. Freedman, F. W. Sears *et al.*, *University Physics*, Reading, Massachusetts: Addison-Wesley Publishing Company, 2000.

[34] W. J. Hayt, and J. A. Buck, *Engineering Electromagnetics*, Seventh ed., New York, New York: McGraw Hill Publishing, 2006.

[35] K. L. Makovec, “A Nonlinear Magnetic Controller for Three-Axis Stability of Nanosatellites,” Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2001.

[36] D. Gubbins, and E. Herrero-Bervera, *Encyclopedia of Geomagnetism and Paleomagnetism*, Springer, Netherlands, 2007.

[37] R. T. Merrill, M. W. McElhinny, and P. L. McFadden, *Magnetic Field of the Earth*, San Diego, California: Academic Press, 1996.

[38] P. Ripka, *Magnetic Sensors and Magnetometers*, Artech House, Boston, 2001.

[39] M. J. Caruso, Bratland, T., Smith, C.H., Schneider, R., “A new perspective on magnetic field sensing,” *Sensors*, pp. 34-46, December, 1998.

[40] A. Chrzanowski, and W. E. Baker, "chapter Tilt Measurement," *Measurement, Instrumentation and Sensor Handbook*, C. P. LLC, ed., 1999.

[41] VTI, “Tilt Sensing - Application Note 2,” vol. VTI Technologies, no. Application Note 2, 2006.

[42] Microchip Technology Inc., "dsPIC30F5011/5013 ," dsPIC30F5011 datasheet, 2004, http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010346

[43] Analog Devices, "AD7734," AD7734 datasheet, 2003, http://www.analog.com/en/analog-to-digital-converters/ad-converters/ad7734/products/product.html

[44] E. Laulainen, L. Koskinen, M. Kosunen *et al.*, "Noise filtering for tilt compensated compass." *Biennial Electronics Conference Proceedings*, 2008, pp. 337-340.

[45] Microchip Technology Inc. "dsPIC C Compiler forPIC24 MCUs and dsPIC DSCs", downloadable code and documentation, 2010, http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010065