NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DYNAMIC ACCURACY OF INERTIAL MAGNETIC SENSOR MODULES

by

Heather L. Pelachick

December 2016

Thesis Advisor:	Xiaoping Yun
Co-Advisor	Zachary Staples
Second Reader	James Calusdian

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE DYNAMIC ACCURACY OF INERTIAL MAGNETIC SENSOR MODULES	**5. FUNDING NUMBERS**
6. AUTHOR(S) Heather L. Pelachick	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Magnetic, angular rate, and gravity (MARG) sensor modules have extensive applications in inertial navigation and motion tracking. A wide assortment of these sensor modules exist at varying cost points. In the current fiscal environment, affordable devices are needed; however, performance cannot be sacrificed.

The main purpose of the study is to devise a series of tests to evaluate the dynamic accuracy of the LORD MicroStrain® 3DM-GX4-25 Attitude Heading Reference System (AHRS). Previous NPS theses have been constrained to static tests and dynamic, rotational tests in a single axis of motion. Utilizing the OptiTrack Motion Capture System, we examine the dynamic accuracy of the 3DM-GX4-25 for rotations in a single axis of motion and arbitrary motions in three-dimensional space, compare the dynamic accuracy against other MARG sensor modules used in previous NPS theses, and provide a cost analysis of the 3DM-GX4-25. Although the dynamic accuracy of the 3DM-GX4-25 met performance specifications for most cases, another MARG sensor module existed with better performance and lower cost.

14. SUBJECT TERMS magnetic, angular rate, and gravity, MARG, micro-electro-mechanical systems, MEMS, quaternion	**15. NUMBER OF PAGES** 147
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

DYNAMIC ACCURACY OF INERTIAL MAGNETIC SENSOR MODULES

Heather L. Pelachick
Captain, United States Marine Corps
B.S., The Pennsylvania State University, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2016

Approved by: Xiaoping Yun
Thesis Advisor

Zachary Staples
Co-Advisor

James Calusdian
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Magnetic, angular rate, and gravity (MARG) sensor modules have extensive applications in inertial navigation and motion tracking. A wide assortment of these sensor modules exist at varying cost points. In the current fiscal environment, affordable devices are needed; however, performance cannot be sacrificed.

The main purpose of the study is to devise a series of tests to evaluate the dynamic accuracy of the LORD MicroStrain® 3DM-GX4-25 Attitude Heading Reference System (AHRS). Previous NPS theses have been constrained to static tests and dynamic, rotational tests in a single axis of motion. Utilizing the OptiTrack Motion Capture System, we examine the dynamic accuracy of the 3DM-GX4-25 for rotations in a single axis of motion and arbitrary motions in three-dimensional space, compare the dynamic accuracy against other MARG sensor modules used in previous NPS theses, and provide a cost analysis of the 3DM-GX4-25. Although the dynamic accuracy of the 3DM-GX4-25 met performance specifications for most cases, another MARG sensor module existed with better performance and lower cost.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AHRS	attitude and heading reference system
AKF	adaptive Kalman filter
AUV	autonomous underwater vehicle
CF	complementary filter
EF	estimation filter
GPS	global positioning system
MARG	magnetic, angular rate, and gravity
MEMS	micro-electro-mechanical systems
NED	north east down
NPS	Naval Postgraduate School
QUARC	Quanser real-time control
R^3	three-dimensional space
RMS	root-mean square
UAV	unmanned aerial vehicle
U.S.	United States
YEI	Yost Engineering Incorporated

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

For all who have influenced and shaped my career…

To my parents, you taught me so much and instilled in me the values to persevere and never give up. To my godparents, your unconditional love has meant the world. To my godfather, thank you for setting the example and showing the path to become more. To my husband, thank you for being by my side through it all. I love you.

A special thanks to my advisor, Xiaoping Yun, co-advisor, Zachary Staples, and second reader, James Calusdian. William Butler Yeats said, “Education is not the simple filling of a pail, but the lighting of a fire.” Thank you for lighting the fire. Your expertise, guidance, patience, and encouragement kept me moving forward. Without you, this could not have been accomplished. Finally, an additional thanks to the lab technicians, Albert Jordan and Rushen Dal.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The magnetic, angular rate, and gravity (MARG) sensor module is designed to measure the three-dimensional angular motion of a rigid body for posture detection. The sensor module is comprised of three angular rate sensors or gyroscopes, three accelerometers, and three magnetometers arranged on the orthogonal axes of the sensor module for a total of nine micro-electro-mechanical systems (MEMS) [1]. The measurements obtained from the MEMS are outputted as raw data and/or processed through a filtering algorithm to achieve the full attitude (roll, pitch, and yaw) solution.

The technology of the MEMS allows for a sensor module that is small, power-efficient, and suitable in a wide range of applications [1]. MARG sensor modules are used for, but not limited to, navigation, human motion tracking, and consumer electronic devices such as cellular telephones. Recent developments include incorporation into global positioning system (GPS) devices, in which MARG sensor modules supplement the devices when a GPS signal is unavailable.

B. MOTIVATION

The United States (U.S.) military has also recognized the resourcefulness of MARG sensor modules and has incorporated the technology in applications such as unmanned aerial vehicles (UAVs), autonomous underwater vehicles (AUVs), and guided missiles. The U.S. military continues to look for innovative ways to adopt the technology in ongoing research. As recently as June 2016, the United States Marine Corps issued a source-sought notice for wearable MARG sensor modules for the purpose of measuring infantry fatigue and performance [2]. With the notice came specific performance specifications; however, performance specifications such as dynamic accuracy can be ambiguous from manufacturers. Since arbitrary motion in three-dimensional space is unconstrained, manufacturers often use a root-mean square (RMS) metric with the caveat that the accuracy may exceed the performance specifications; however, no further

explanation is provided. Furthermore, instantaneous errors, which can be quite large for a specific RMS value, are not addressed.

Concurrent research with MARG sensor modules has also been conducted at the Naval Postgraduate School (NPS). In particular, a project known as Reticle seeks to create a battlefield network for dismounted infantry, with the capability of deconflicting geometries-of-fire, especially during situations in which line-of-sight is obscured [3], [4]. The network is also designed to alert commanders when the warfighter fires his or her direct-fire weapon and provide the direction of that fire [5]. Lastly, a posture-detection algorithm was developed to indicate if a warfighter is in a kneeling or prone position and uses this information to reduce drift when a MARG sensor module is stationary for long periods of time [6]. Throughout the design of Reticle, the Yost Engineering Incorporated (YEI) 3-space data-logging sensor was utilized at a cost of approximately $255.00 per unit. Prior to adoption into Reticle, the static and dynamic performance specifications were validated in [4], and the cost of the YEI 3-space data-logging sensor was justified.

C. PREVIOUS WORK

In [7], Jeremy Cookson built a low-cost pendulum with an optical encoder to test the dynamic accuracy of MARG sensor modules. The pendulum was designed in order to execute dynamic, repeatable tests in a single axis. Each axis of the MARG sensor was tested independently by aligning it to the direction of swing. Cookson applied his research to the MicroStrain® 3DM-GX1 and 3DM-GX3-25 sensors. In [8], Leslie Landry developed similar repeatable tests and utilized the pendulum to test the dynamic accuracy of the Xsens MTx sensor. In both theses, the motion of the pendulum was constrained to one axis; therefore, a limitation existed in which the tests were only able to measure the dynamic accuracy one axis at a time. For that reason, it was not possible to explore the three-dimensional dynamic accuracy simultaneously.

D. GOALS

The objective of this thesis is trifold. First, a series of dynamic accuracy tests for arbitrary motions in three-dimensional space must be developed. Furthermore, these tests cannot be limited under test to a single axis. A ground truth reference instrument that is at

least one order of magnitude more accurate than the MARG sensor module under test must be utilized, and a way to compare those results must be developed. Second, the performance of the MARG sensor module under test needs to be compared with other MARG sensor modules used in previous NPS theses. Finally, in the current fiscal environment, affordable devices are needed; however, performance cannot be sacrificed. Consequently, a cost analysis of the MARG sensor module under test must be performed and a determination be made to adopt the MARG sensor module into Reticle.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ROTATIONS IN THREE-DIMENSIONAL SPACE

Rotational representations in three-dimensional space, such as Euler angles, rotation matrices, and quaternions are introduced in this chapter. Algebraic properties of quaternions and equivalent relationships between rotations in three-dimensional space are also defined.

Representation of a point in three-dimensional space, denoted as R^3, is accomplished through the selection of an origin and three coordinate axes. The origin, denoted O, is a fixed point in which the three coordinate axes, denoted the x-, y-, and z-axes, pass through O and are mutually perpendicular to one another [9]. A representation of R^3 is illustrated in Figure 1.

The three-dimensional coordinate system is defined as either a right-handed coordinate system or left-handed coordinate system, based on the direction of the positive z-axis with respect to the positive x- and y-axis. To determine the orientation of a three-dimensional coordinate system, one uses the right hand and aligns it positively along the x-axis. Then, the fingers are curled toward the positive y-axis. If the thumb points in the direction of the positive z-axis, the coordinate system is a right-handed coordinate system. If the thumb points in the direction of the negative z-axis, the coordinate system is a left-handed coordinate system [10]. This can also be verified by repeating the steps above with the left hand. Applying this method to Figure 1 reveals that the coordinate system follows the right-hand rule.

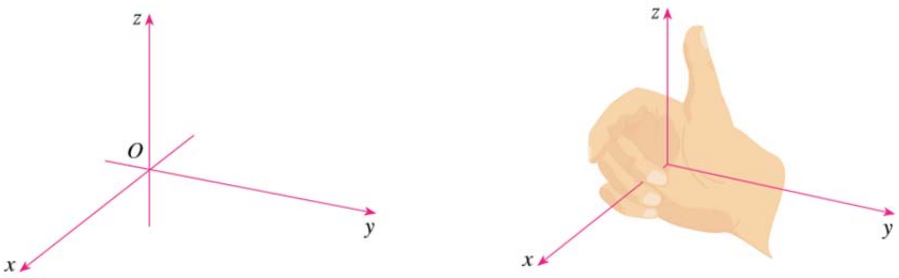

Figure 1. R^3 and Right-Hand Rule. Source: [9].

A three-dimensional coordinate system can be defined for numerous applications. Standard coordinate systems also exist such as the north, east, down (NED) coordinate frame. The NED coordinate frame is formed utilizing a plane tangent to the Earth's surface. The x- and y-axes lie in this plane with the positive x-axis pointing North and the positive y-axis pointing East. The positive z-axis points downward toward the center of the Earth, defining a right-handed coordinate system [10].

A. ROLL, PITCH, AND YAW

Consider an object located in the coordinate system of Figure 1. The orientation of this object can be defined with reference to its rotation about its coordinate axes. A rotation through an angle ϕ about the x-axis is defined as roll. Subsequently, a rotation through an angle θ about the y-axis is defined as pitch. Finally, a rotation through an angle ψ about the z-axis is defined as yaw.

Furthermore, the direction of positive rotation is defined through the use of the right-hand or left-hand rule. For a right-handed coordinate system, a positive rotation about any of its axes is a counter-clockwise rotation. If one is unsure of the direction of a positive rotation, align the right thumb along one of the positive axes and curl the fingers. The positive direction is in the direction of the curled fingers. For a left-handed coordinate system, a positive rotation about any of its axes is a clockwise rotation. The direction of positive rotation can be verified by repeating the steps above with the left hand.

B. EULER ANGLES AND ROTATION MATRICES

In the instance of a second coordinate system, the second coordinate system can be rotated into the first, given both have adopted the same right-handed or left-handed orientation, via a rotation matrix. The angle through which a coordinate axis is rotated is called an Euler angle. Let the x-, y-, or z-axis be rotated through an angle ϕ, θ, or ψ, respectively. Through trigonometric properties, the rotation matrices R_ϕ, R_θ, and R_ψ are defined in [10] as

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$
$$R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.1}$$
$$R_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

C. QUATERNIONS

In 1843, William Rowan Hamilton devised a method for representing orientation in a three-dimensional space which he called a quaternion [10]. The quaternion is expressed as

$$q = q_0 + \vec{i}q_1 + \vec{j}q_2 + \vec{k}q_3 = q_0 + \vec{q} \tag{2.2}$$

where q_0 is a scalar, $\vec{q}$ is a vector in R^3, and $\vec{i}$, $\vec{j}$, and $\vec{k}$ are unit vectors.

The multiplication of two quaternions

$$\begin{aligned} p &= p_0 + \vec{i}p_1 + \vec{j}p_2 + \vec{k}p_3 \\ q &= q_0 + \vec{i}q_1 + \vec{j}q_2 + \vec{k}q_3 \end{aligned} \tag{2.3}$$

is defined in [10] as

$$\begin{aligned} pq = {} & p_0q_0 - \left(p_1q_1 + p_2q_2 + p_3q_3\right) \\ & + p_0\left(iq_1 + jq_2 + kq_3\right) + q_0\left(ip_1 + jp_2 + kp_3\right) \\ & + i\left(p_2q_3 - p_3q_2\right) + j\left(p_3q_1 - p_1q_3\right) + k\left(p_1q_2 - p_2q_1\right). \end{aligned} \tag{2.4}$$

Rotations in a three-dimensional coordinate system can be represented by either Euler angles, rotation matrices, or quaternions and can be interchanged through mathematical equations. Given the quaternion in Equation (2.2), the equivalent Euler angles are defined in [10] as

$$\begin{aligned}\phi &= \arctan\left(\frac{2q_2q_3 + 2q_0q_1}{2q_0^2 + 2q_3^2 - 1}\right) \\ \theta &= \arcsin\left(-2q_1q_3 + 2q_0q_2\right) \\ \psi &= \arctan\left(\frac{2q_1q_2 + 2q_0q_3}{2q_0^2 + 2q_1^2 - 1}\right).\end{aligned} \tag{2.5}$$

Given the rotation matrix,

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.6}$$

the scalars of the equivalent quaternion are defined in [10] as

$$\begin{aligned} q_0 &= \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1} \\ q_1 &= \frac{r_{23} - r_{32}}{4q_0} \\ q_2 &= \frac{r_{31} - r_{13}}{4q_0} \\ q_3 &= \frac{r_{12} - r_{21}}{4q_0}. \end{aligned} \tag{2.7}$$

The rotational mathematics discussed here are used to align the coordinate systems of the MARG sensor module under test and the ground truth reference instrument described in Chapter III.

III. MARG SENSOR MODULE AND MOTION CAPTURE SYSTEM

The hardware and associated software of the MARG sensor module under test and ground truth reference instrument used in this thesis are described in this chapter. First, the LORD MicroStrain® 3DM-GX4-25 MARG sensor module, which was evaluated for its dynamic accuracy in order to justify its cost and determine if it is a viable option for Reticle, is described. Then, the NaturalPoint OptiTrack 2.3.3 Motion Capture System, used as the ground truth reference instrument, is described.

A. LORD MICROSTRAIN® 3DM-GX4-25

The LORD MicroStrain® 3DM-GX4-25, shown in Figure 2, is a MARG sensor module that provides direct sensor measurements and computed outputs. The 3DM-GX4-25 can be used in a wide range of applications such as unmanned vehicle navigation, robotic control, platform stabilization, motion tracking and analysis, vehicle health monitoring, and device aiming [11]. It is available separately or contained within two types of starter kits. The starter kits contain the 3DM-GX4-25, a communications cable (USB or RS232), a power supply with country plug adapters, Lord MicroStrain® MIP Monitor software suite, User Manual, Quick Start Guide, and Calibration Certificate [11]. The purchase price of the 3DM-GX4-25 with starter kit used in this thesis was approximately $1,895.00.

Figure 2. LORD MicroStrain® 3DM-GX4-25. Source: [11].

1. Configuration

The 3DM-GX4-25 contains tri-axial gyroscopes (gyros), tri-axial accelerometers, tri-axial magnetometers, temperature sensors, and a pressure sensor. The tri-axial sensors are aligned along the 3DM-GX4-25's orthogonal coordinate system and are temperature compensated to provide direct sensor measurements. Computed outputs are obtained once the temperature compensated measurements are processed through an attitude and heading reference system (AHRS) estimation filter (EF) microprocessor with an adaptive Kalman filter (AKF). The 3DM-GX4-25 houses a separate processor in which temperature compensated measurements are processed through a complementary filter (CF) in order to maintain backwards compatibility with the Lord MicroStrain® 3DM-GX3 sensor modules [11]. A block diagram of the 3DM-GX4-25 is shown in Figure 3.

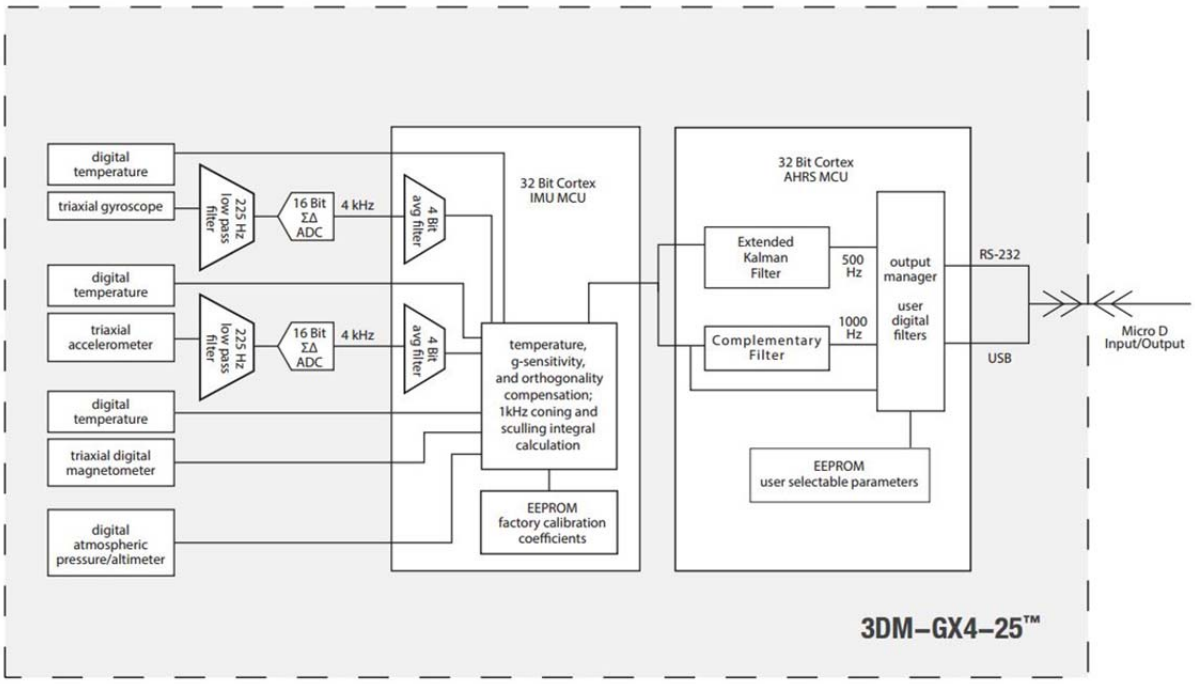

Figure 3. 3DM-GX4-25 Block Diagram. Source: [11].

a. Direct Sensor Measurements

The direct sensor measurements of the 3DM-GX4-25 include angular rate, acceleration, and the magnetic field. Angular rate is a $n \times 3$ matrix where n is the

number of time samples, and the columns of the matrix are the readings corresponding to the x-, y-, and z-axes in units of radians per second. Acceleration is a $n \times 3$ matrix where n is the number of time samples, and the columns of the matrix are the readings corresponding to the x-, y-, and z-axes in units of gravitational force (g). The magnetic field, including the Earth's magnetic field and local magnetic anomalies, is a $n \times 3$ matrix where n is the number of time samples, and the columns of the matrix are the readings corresponding to the x-, y-, and z-axes in units of Gauss (G) [11].

b. Computed Outputs

The computed outputs of the 3DM-GX4-25 include the GPS correlation timestamp and the full attitude solution. Although the 3DM-GX4-25 did not contain the GPS functionality, the timestamp was provided from the processor time and measured in weeks and seconds; therefore, the time does not start at zero when the 3DM-GX4-25 is initialized. To synchronize time for this thesis, the initial value from the GPS correlation timestamp was manually subtracted from all time samples in the MATLAB code. The full attitude solution can be represented in Euler angles, quaternions, or rotation matrices. For this thesis, the quaternion was the selected output. The quaternion is represented as a $n \times 4$ matrix where n is the number of time samples, and the columns of the matrix are the scalars of the quaternion in the order q_0, q_1, q_2, and q_3.

2. Software

The Lord MicroStrain® MIP Monitor software suite is required for configuration of the 3DM-GX4-25, real-time measurement monitoring, and data recording. The software is included with the starter kit on a flash drive or is available as a free download from the Lord MicroStrain® website. It was easily installed and operated from a Windows® 7 host computer containing a 2.20-GHz processor and 8 GB of RAM by running the *Autorun.exe* file from the software directory in Windows® Explorer and following the on screen prompts [11].

3. Coordinate System

a. Sensor Coordinate System

The sensor coordinate system is illustrated on the housing of the 3DM-GX4-25 as seen in Figure 4. The positive x-axis points toward the communication and power port of the 3DM-GX4-25 and is parallel with the long side of the housing. The positive y-axis points toward the mounting hole of the 3DM-GX4-25 and is offset clockwise from the positive x-axis by ninety degrees. The positive z-axis points directly through the bottom of the 3DM-GX4-25 [11]. This orientation defines a right-handed coordinate system.

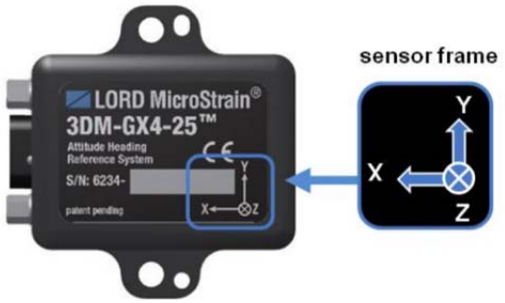

Figure 4. 3DM-GX4-25 Sensor Coordinate Frame. Source: [11].

b. NED Coordinate Frame

The 3DM-GX4-25 reports the full attitude solution in the NED coordinate frame which is defined as a right-handed coordinate system.

4. Performance Specifications

Lord MicroStrain® Sensing Systems provides performance specifications, as seen in Figure 5, for the direct sensor measurements and computed outputs. For this thesis, the dynamic accuracy of the CF output was scrutinized. The CF output was chosen over the AKF output due to the fact that values from the AKF were irregularly outputted when a (.csv) file was selected as the data file. The dynamic accuracy for the CF is given in RMS and indicates “typ.” The RMS value and “typ,” or “typical,” can be vague to the user. Since the RMS value only measures the magnitude of a set of data points, instantaneous error may be greater than the specified RMS value. Additionally, “typical” suggests that

the dynamic accuracy may exceed the specified RMS value; however, no indication for when this is acceptable is provided.

General			
Integrated sensors	Triaxial accelerometer, triaxial gyroscope, triaxial magnetometer, temperature sensors, and pressure altimeter,		
Data outputs	**Inertial Measurement Unit (IMU) outputs:** acceleration, angular rate, magnetic field , ambient pressure, deltaTheta, deltaVelocity **Computed outputs:** **Adaptive Kalman Filter (AKF):** filter status, timestamp, attitude estimates (in Euler angles, quaternion, orientation matrix), bias compensated angular rate, pressure altitude, gravity-free linear acceleration, attitude uncertainties, gyroscope and accelerometer bias, scale factors and uncertainties, gravity and magnetic models, and more. **Complementary Filter (CF):** attitude estimates (in Euler angles, quaternion, orientation matrix), stabilized north and gravity vectors, correlation timestamp		
Inertial Measurement Unit (IMU) Sensor Outputs			
	Accelerometer	**Gyroscope**	**Magnetometer**
Measurement range	±5 *g* (standard) ±16*g* (option)	300°/sec (standard) ±75, ±150, ±900 °/sec (options)	±2.5 Gauss
Non-linearity	±0.03 % fs	±0.03 % fs	±0.4% fs
Resolution	<0.1 m*g*	<0.008°/sec	--
Bias instability	±0.04 m*g*	10°/hr	--
Initial bias error	±0.002 *g*	±0.05°/sec	±0.003 Gauss
Scale factor stability	±0.05 %	±0.05 %	±0.1 %
Noise density	100 µg/√Hz	0.005°/sec/√Hz	100 µGauss/√Hz
Alignment error	±0.05°	±0.05°	±0.05°
Adjustable bandwidth	225 Hz (max)	250 Hz (max)	-
Offset error over temperature	0.06% (typ)	0.05 % (typ)	--
Gain error over temperature	0.05% (typ)	0.05% (typ)	--
Scale factor non-linearity (@ 25° C)	0.02% (typ) 0.06% (max)	0.02% (typ) 0.06% (max)	±0.0015 Gauss
Vibration induced noise	--	0.072°/s RMS/*g* RMS	--
Vibration rectification error (VRE)	--	0.001°/s/g^2 RMS	--
IMU filtering	4 stage filtering: analog bandwidth filter to digital sigma-delta wide band anti-aliasing filter to (user adjustable) digital averaging filter sampled at 4 kHz and scaled into physical units; coning and sculling integrals computed at 1 kHz		
Sampling rate	4 kHz	4 kHz	50 Hz
IMU data output rate	1 Hz to 1000 Hz		
Pressure Altimeter			
Range	-1800 m to 10,000 m		
Resolution	< 0.1 m		
Noise density	0.01 hPa RMS		
Sampling rate	25 Hz		

Computed Outputs	
Attitude accuracy	AKF outputs: ±0.25° RMS roll & pitch, ±0.8° RMS heading (typ) CF outputs: ±0.5° roll, pitch, and heading (static, typ), ±2.0° roll, pitch, and heading (dynamic, typ)
Attitude heading range	360° about all axes
Attitude resolution	< 0.01°
Attitude repeatability	0.3° (typ)
Calculation update rate	500 Hz
Computed data output rate	AKF outputs: 1 Hz to 500 Hz CF outputs: 1 Hz to 1000 Hz
Operating Parameters	
Communication	USB 2.0 (full speed) RS232 (9,600 bps to 921,600 bps, default 115,200)
Power source	+ 3.2 to + 36 V dc
Power consumption	100 mA (typ), 120 mA (max) with Vpri = 3.2 V dc to 5.5 V dc 550 mW (typ), 800 mW (max) with Vaux = 5.2 V dc to 36 V dc
Operating temperature	-40 °C to +85 °C
Mechanical shock limit	500 *g* (calibration unaffected) 1000 *g* (bias may change), 5000 *g* (survivability)
MTBF	1.2 million hours (Telcordia method I, GL/35C) 0.45 million hours (Telcordia method I, GM/35C)
Physical Specifications	
Dimensions	36.0 mm x 24.4 mm x 36.6 mm
Weight	16.5 grams
Enclosure material	Aluminum
Regulatory compliance	ROHS, CE
Integration	
Connectors	Data/power output: micro-DB9
Software	MIP™ Monitor, MIP™ Hard and Soft Iron Calibration, Windows XP/Vista/7/8 compatible
Compatibility	Protocol compatibility across 3DM-GX3, GX4, RQ1, GQ1, and GX5 product families
Software development kit (SDK)	MIP™ data communications protocol with sample code available (OS and platform independent)

LORD SENSING

Figure 5. 3DM-GX4-25 Performance Specifications. Source: [12].

B. NATURALPOINT OPTITRACK

1. Configuration

The NaturalPoint OptiTrack 2.3.3 system is a motion capture system that uses infrared cameras to track the posture of a defined object. The workspace was configured with twelve infrared cameras mounted near the ceiling of a 3.5×5.5 meter room and served to maximize the required overlapping fields of view for the cameras. Additionally, the floor was covered with interlocking rubber mats to reduce any glare or reflections that may affect camera performance. The system was operated from a ground control station with a Windows® 7 host computer containing a 3.20-GHz processor and 12 GB of RAM. A model of the workspace with the ground control station is shown in Figure 6.

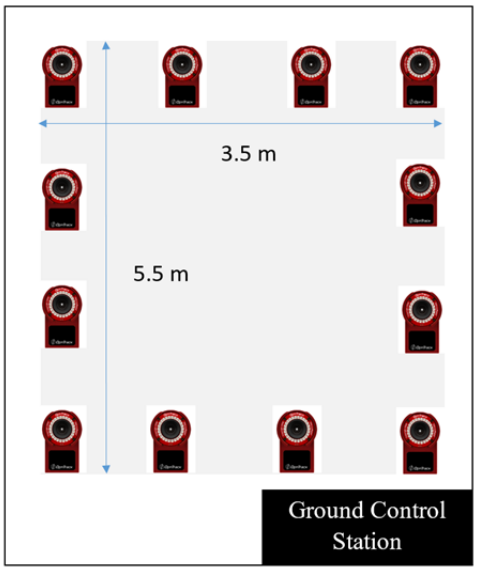

Figure 6. Workspace and Ground Control Station Configuration. Adapted from [13].

2. Software

The OptiTrack Tracking Tools software is required for performing camera calibrations, defining the workspace coordinate system, and defining an object for posture detection (called a trackable) [14]. With a camera calibration completed and workspace coordinate system and trackable defined, the time-sampled posture of the

trackable was captured using the Quanser real-time control (QUARC) 2.2 software and exported via MATLAB/Simulink R2011b. QUARC eliminates the need for computer code by adding several libraries of custom blocks to Simulink for use in building a Simulink model and streaming data to the MATLAB workspace [15].

3. Coordinate System

a. OptiTrack Coordinate System

The NaturalPoint OptiTrack 2.3.3 coordinate system is seen in Figure 7(a). The positive z-axis points toward the ground control station. The positive x-axis points to the left of the workspace and is offset clockwise from the positive z-axis by ninety degrees. The positive y-axis points up. This orientation defines a left-handed coordinate system (For newer versions of OptiTrack, the standard coordinate system is defined as a right-handed coordinate system.).

b. QUARC Coordinate System

With the OptiTrack coordinate system defined as a left-handed coordinate system, QUARC is advertised to undergo a transformation to convert the coordinate system into a right-handed coordinate system; however, engineers have observed a discrepancy with certain versions. QUARC 2.2 seemed to inherit this discrepancy when paired with NaturalPoint OptiTrack 2.3.3; therefore, the coordinate system initially obtained followed the same left-hand rule orientation as in 7(a) but followed the right-hand rule for rotations about the axes. To correct this discrepancy, a negative sign was added to the data corresponding to the y-axis. The modified QUARC coordinate system is seen in Figure 7(b), where the positive z-axis points toward the ground station, the positive x-axis points left, and positive y-axis points down.

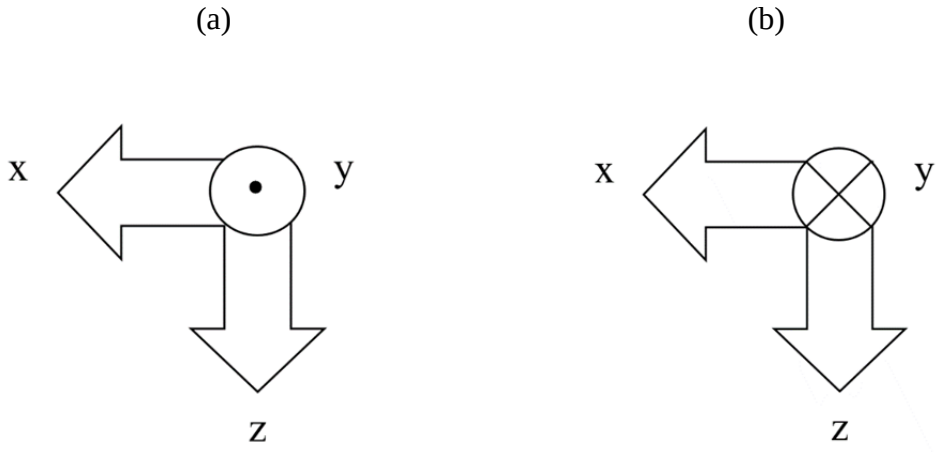

Figure 7. Coordinate System Diagram of OptiTrack (a) and Modified QUARC (b)

In summary, the 3DM-GX4-25, OptiTrack, and associated software were introduced in this chapter. The 3DM-GX4-25 was tested, and its direct sensor measurements and computed outputs were extracted by means that are discussed in Chapter IV. The OptiTrack Motion Capture System was described which provides the posture of a trackable that is made available in MATLAB/Simulink via the QUARC software library. It is this combination of OptiTrack and QUARC that provide the ground truth data for calculating the dynamic accuracy of the 3DM-GX4-25. Finally, a discussion of the differing coordinate systems of the 3DM-GX4-25, OptiTrack, and QUARC were introduced to highlight data pre-processing that is discussed in Chapter V.

IV. TEST DESIGN AND METHODOLOGY

The test setup procedures and methodology used in this thesis are described in this chapter. First, the fabrication process to convert the 3DM-GX4-25 into a trackable is described. Next, the steps to extract direct sensor measurements and computed outputs from the 3DM-GX4-25 with the LORD MicroStrain® MIP Monitor software suite are presented. Then, the setup and operation of OptiTrack via OptiTrack Tracking Tools software and QUARC are described. Finally, the testing methodology used to investigate the performance of the 3DM-GX4-25 is presented.

A. TEST SETUP

1. 3DM-GX4-25

The 3DM-GX4-25 sensor housing is designed with two mounting tabs that have holes for easy installment. Utilizing brass screws to avoid interference with the 3DM-GX4-25's magnetometers, the sensor was mounted to a wooden block. Five 0.125-inch holes were drilled into the front face of the wooden block asymmetrically. This was done to avoid ambiguous posture solutions in which OptiTrack could not discern a rotation [14]. Five infrared markers were attached to wooden dowel rods and installed to the wooden block via the drilled holes. The mounted 3DM-GX4-25 with infrared markers is seen in Figure 8.

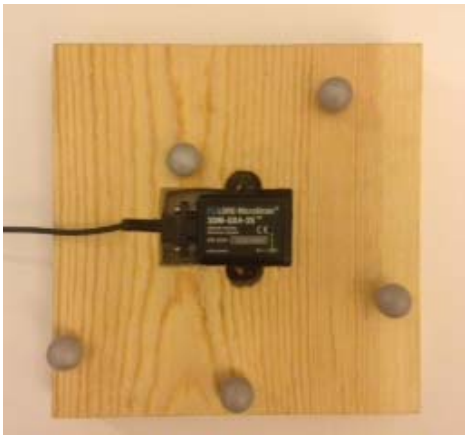

Figure 8. Mounted 3DM-GX4-25 with Infrared Markers for Posture Detection

2. LORD MicroStrain® MIP Monitor Software Suite

The LORD MicroStrain® MIP Monitor software suite is required for configuration and obtainment of direct sensor measurements and computed outputs from the 3DM-GX4-25 as described in [11].

LORD MicroStrain® MIP Monitor Software Suite Setup

1. Launch the MIP Monitor software. The MIP Monitor window is loaded with the 3DM-GX4-25 displayed in the device pane as seen in Figure 9.

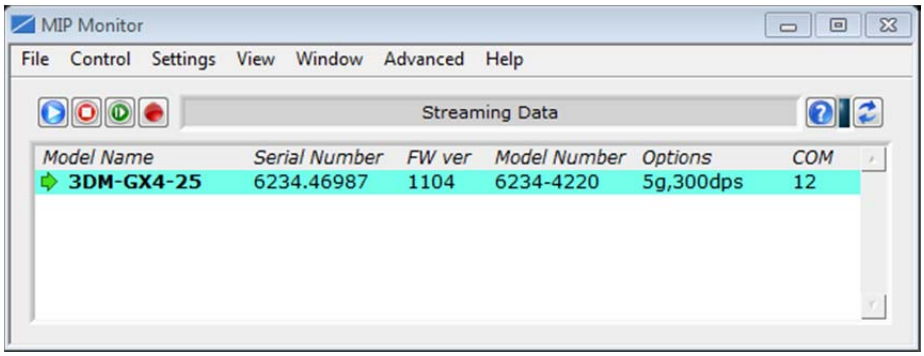

Figure 9. MIP Monitor Window

2. Right click on the device name and select Device Settings. The Device Setup window is loaded as seen in Figure 10.

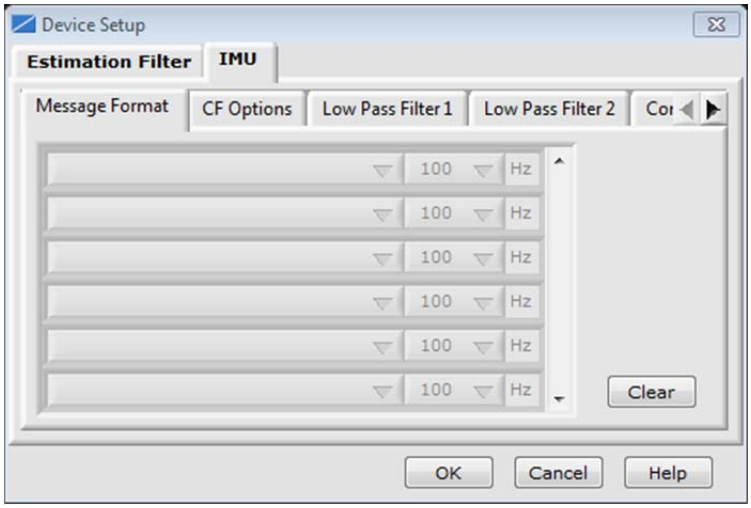

Figure 10. Device Setup Window

3. On the IMU tab, click the top dropdown arrow and select Accelerometer Vector. This outputs the raw x-, y-, and z-axis accelerations in units of gravitational force (g). Click the top dropdown to the right and select 100. This is the sampling rate of 100 samples per second or hertz.

4. Repeat the procedures in Step 3 for subsequent rows, selecting Angular Rate Vector, Magnetometer Vector, GPS Correlation Timestamp, and CF Quaternion along with the same sampling rate. The Angular Rate Vector outputs the raw x-, y-, and z-axis angular rate in units of radians per second. The Magnetometer Vector outputs the raw x-, y-, and z-axis magnetic field in units of Gauss (G). The GPS Correlation Timestamp outputs the time metric from the processor measured in weeks and seconds. The CF Quaternion outputs the computed scalars of the quaternion described in Equation (2.2).

5. On the MIP Monitor window, select View > Sensor Data Monitor. The Data Monitoring and Streaming window is loaded as seen in Figure 11.

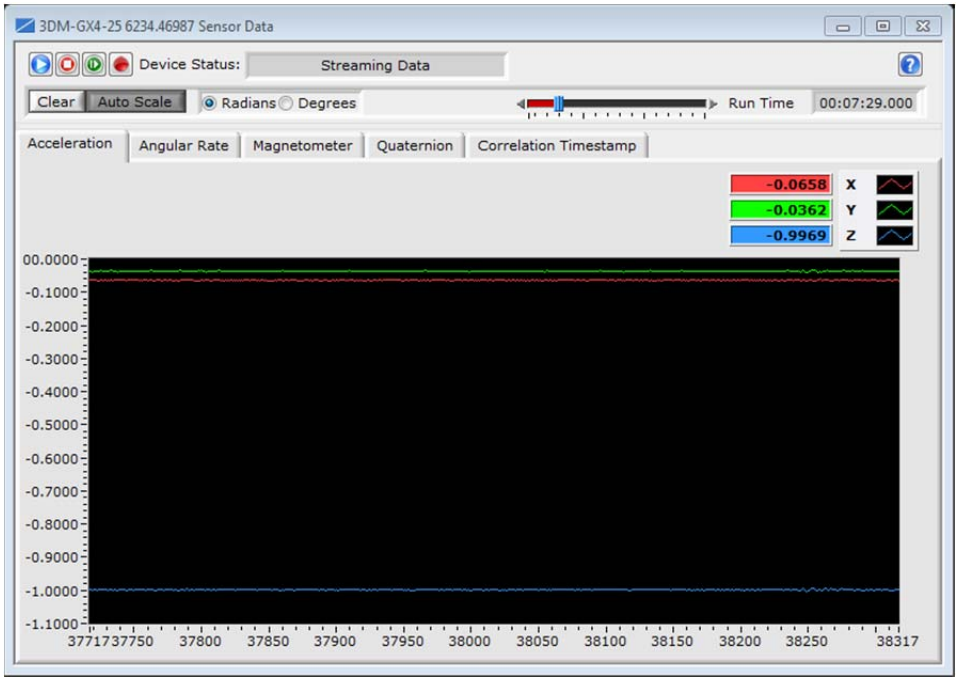

Figure 11. Data Monitoring and Streaming Window

6. Click the Start Streaming Data icon (blue arrow). The selected outputs can now be observed in the window.

7. To record data, click the Arm Recording icon (red dot). The Log File Format window is loaded. Click the dropdown arrow and select

Spreadsheet File (csv). Click OK. Select the file path and file name, then save the file as a (.csv) file. Click OK.

8. To end recording, click the Arm Recording icon again and click OK.

To initiate follow-on simulations, repeat steps seven through eight.

3. OptiTrack Tracking Tools Software

Prior to operation of OptiTrack, a camera calibration must be performed, the coordinate system of the workspace must be defined, and an object for posture detection must be defined as a trackable.

a. Performing a Camera Calibration

Camera calibrations, as described in [14], are recommended to be performed by Quanser once per week. For this thesis, all data was extracted following a camera calibration to achieve the best results.

Camera Calibration

1. Launch the NaturalPoint Tracking Tools Software. The Tracking Tools Quick Start menu is loaded as seen in Figure 12.

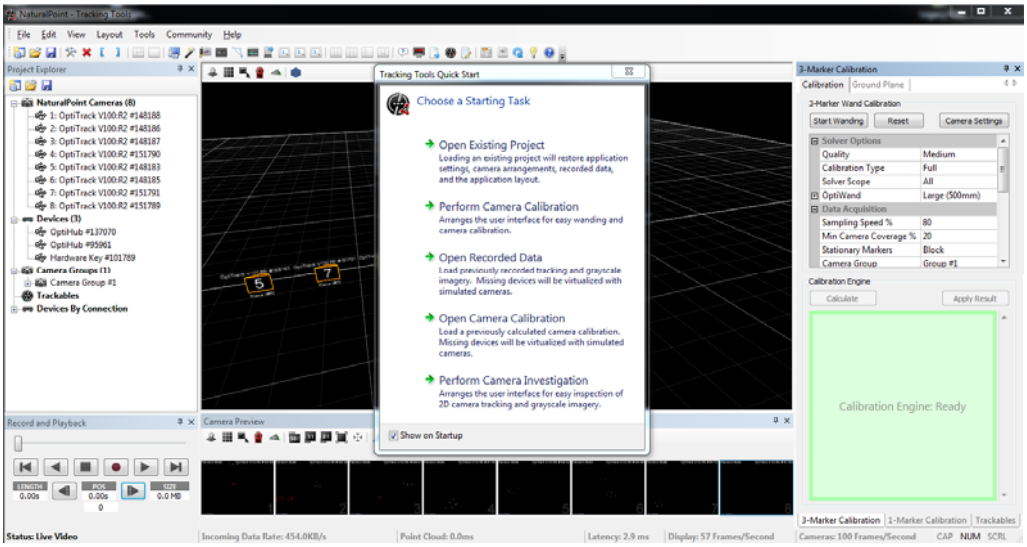

Figure 12. Tracking Tools Quick Start Menu with Initial NaturalPoint Tracking Tools Backdrop

2. Click on Perform Camera Calibration. The Tracking Tools calibration screen is loaded.

3. Ensure all infrared markers have been removed from the workspace. For other infrared sources that cannot be removed, block the disturbances from each camera view. On the Tracking Tools calibration screen, right click each camera window and select Hardware Mask > Block Visible Markers as seen in Figure 13.

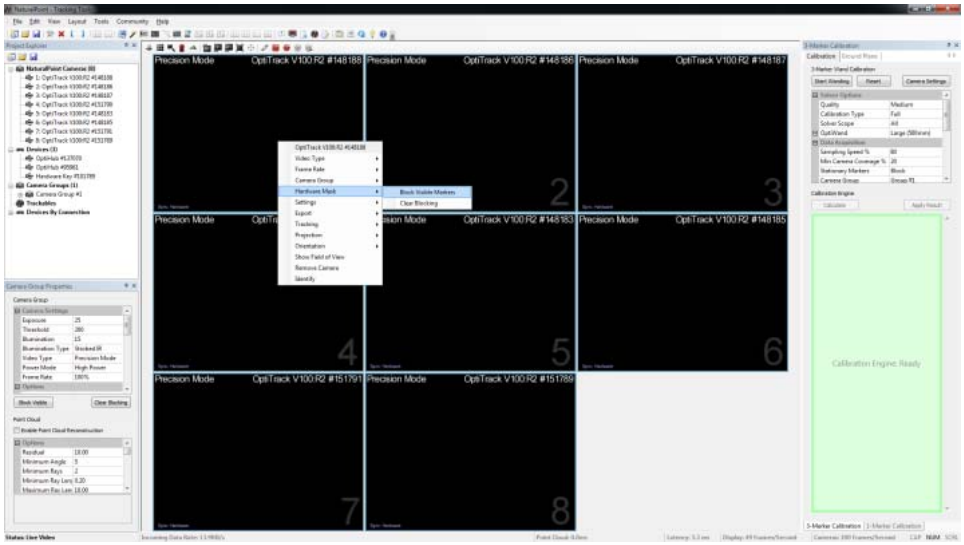

Figure 13. Blocking Visible Markers on Tracking Tools Calibration Screen

4. Set the solver quality. On the 3-Marker Calibration tab located on the bottom, right corner of the Tracking Tools calibration screen, click the dropdown arrow to the left of Quality under Solver Options. Select Very High (slow), which corresponds to the most accurate calibration.

5. Click on Start Wanding.

6. Begin swaying the 3-Marker Calibration Wand, as seen in Figure 14, slowly and evenly through the entire workspace ensuring all elevations are covered. Each camera view on the Tracking Tools calibration screen displays the captured samples. Collect between 2,000 and 10,000 samples, ensuring complete coverage. Once a sufficient number of samples are collected, the background color of the Calibration Engine task pane turns green. Click on Calculate. The calibration process begins, and the solver window appears with the overall result rating at the top and each camera listed with the number of samples and mean error. The overall result rating is based on the lowest rating of any one camera and is tiered into six

categories: Poor, Fair, Good, Great, Excellent, and Exceptional. Allow the calibration process to run until the result rating reaches Exceptional.

Figure 14. 3-Marker Calibration Wand

7. After receiving the message Ready to Apply and obtaining the desired overall result rating, click on Apply Result. The Calibration Result Report screen is loaded as seen in Figure 15.

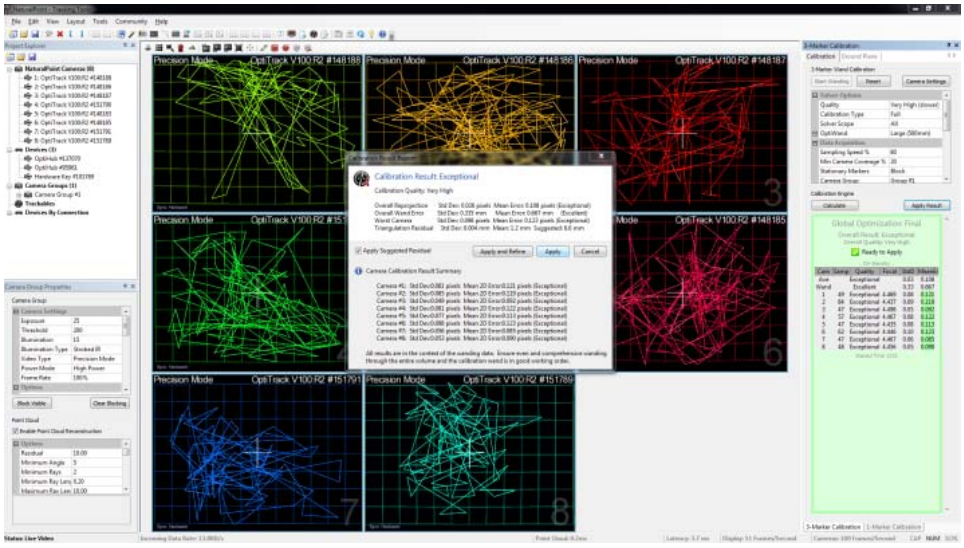

Figure 15. Calibration Result Report Screen

8. Click Apply. The Save As menu will be loaded.

9. Save the camera calibration as a (.cal) file.

b. Defining the Workspace Coordinate System

Once the camera calibration has been performed, the workspace coordinate system must be defined through the groundplane calibration procedures described in [14].

Groundplane Calibration

1. After saving the camera calibration, the Groundplane calibration screen is loaded.

2. Place the calibration square with infrared markers, as seen in Figure 16, in the field of view of OptiTrack with the z-axis of the square pointing to the operator's station and the x-axis pointing to the left. Ensure the maximum number of cameras have the calibration square in their fields of view. Click on Set Ground Plane as seen in Figure 17.

Figure 16. Calibration Square

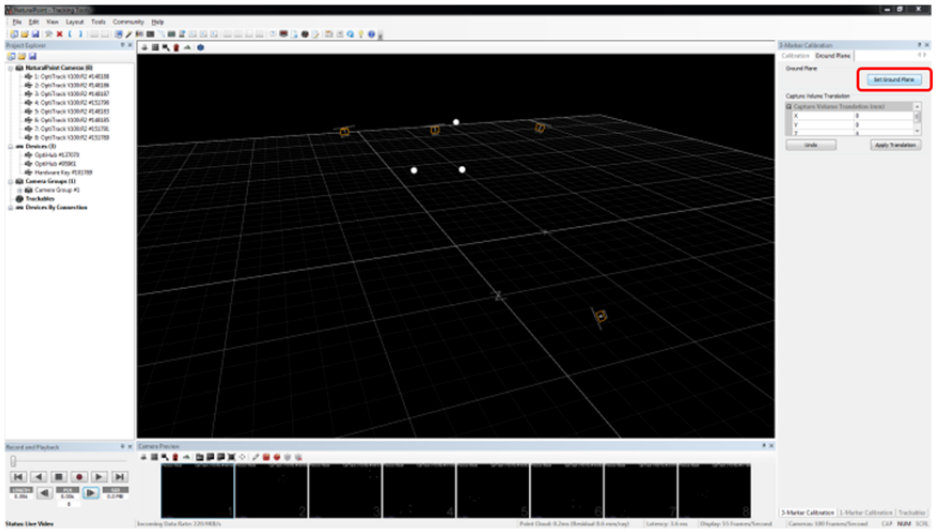

Figure 17. Groundplane Calibration Screen

3. The Save As menu will be loaded. Save the groundplane as a (.cal) file.

c. Defining a Trackable Object

In order for OptiTrack to detect the posture of an object in the workspace, the object must be defined as a trackable as described in [14].

Defining a Trackable

1. Place the object with a minimum of three asymmetrically oriented infrared markers in the workspace. Select the Trackables tab located on the bottom, right corner of the Tracking Tools calibration screen. While holding the shift key, select the white orbs corresponding to the infrared markers of the object to be tracked. Field of view lines between the cameras and the infrared markers appear as seen in Figure 18.

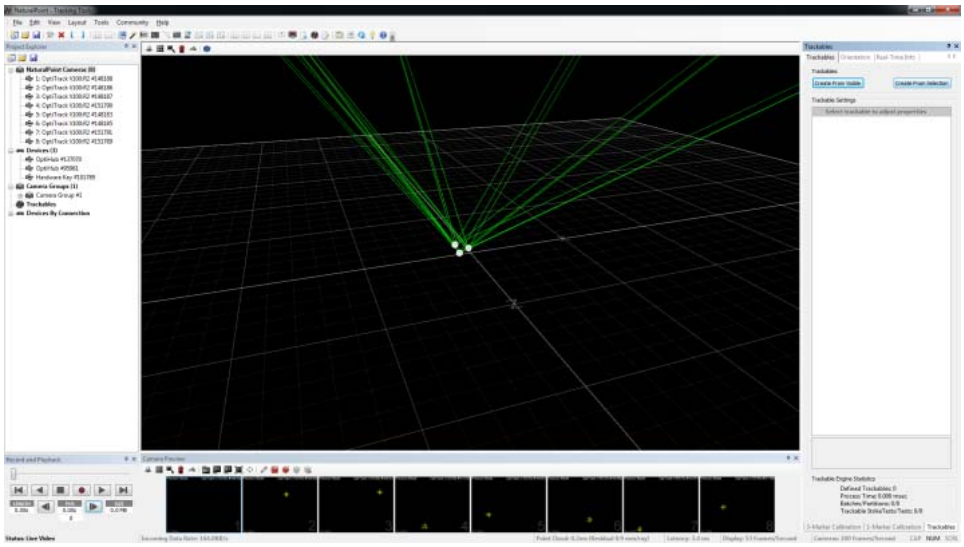

Figure 18. Selection of Infrared Markers to Define a Trackable

2. Once all markers have been selected, click on Create From Selection. The trackable is created and assigned an identification number under Trackable ID in General Properties of Trackable Settings.

3. Save the trackable as a (.tra) file, by selecting Save Trackables under the File menu.

4. QUARC

Once the camera calibration with groundplane calibration is completed and a trackable is defined, a Simulink model must be created to extract the ground truth data via QUARC as described in [14].

QUARC Setup

1. In Simulink, create a new Simulink model and add the OptiTrack Trackables block, as seen in Figure 19, located in the QUARC library under QUARC Targets > Devices > Third-Party > NaturalPoint > OptiTrack.

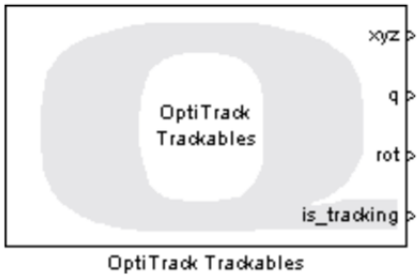

Figure 19. OptiTrack Trackables Block

2. Double-click the OptiTrack Trackables block. In the Source Block Parameters, reference the groundplane calibration file and trackables file generated during the groundplane calibration and defining a trackable procedures. Enter the assigned Trackable ID from the trackable settings. Enter the sampling time of 0.01 seconds, corresponding to the sampling rate of 100 hertz.

3. In the model, add a Clock block found in the QUARC library under Simulink > Sources.

4. In the model, add two To Workspace blocks found in the QUARC library under Simulink > Sinks. Connect one block to the Clock. Connect the second block to the "q" of the OptiTrack Trackables block. The "q" outputs a $n \times 4$ matrix where n is the number of samples, and the columns of the matrix are the scalars of a quaternion in the order q_1, q_2, q_3, and q_0.

5. Select Simulation > Model Configuration Parameters and enter the values from Table 1.

Table 1. Configuration Parameters for Simulink Model

Start time	0.0
Stop time	Inf
Type	Fixed-step
Solver	ode1(Euler)
Fixed-step size (fundamental sample time)	0.01
Periodic sample time constraint	Unconstrained
Tasking mode for periodic sample times	Auto

6. Build the model by clicking the Incremental Build icon.

7. Click the Connect to Target icon.

8. Capture the position of the trackable by clicking the Start Real-Time code icon.

9. To stop the simulation, click the Stop icon.

To initiate follow-on simulations, repeat steps six through nine.

B. TEST METHODOLOGY

1. Slow Motion in a Single Axis

For the first group of tests, the 3DM-GX4-25 was constrained to one positive rotation through a single axis at an angular rate no greater than one radian per second. Three trials for each angle of rotation were conducted. In order to perform a rotation and capture its time-sampled positions, the mounted 3DM-GX4-25 with infrared markers was placed on the ground near the origin of OptiTrack set during the groundplane calibration. The positive x-axis of the 3DM-GX4-25 was placed in line with the defined x-axis of OptiTrack. The data collection was started for the 3DM-GX4-25 and QUARC. After a few seconds of no disturbance, the mounted 3DM-GX4-25 with infrared markers was rotated by hand about the selected axis to the appearance of forty-five degrees. For a rotation about the x- or y-axis, or through the roll or pitch angle, respectively, the angle of rotation was held constant by resting the mounted 3DM-GX4-25 with infrared markers against a nonmetallic rigid source. For a rotation about the z-axis, or though the yaw angle, the mounted 3DM-GX4-25 with infrared markers could remain horizontally on the ground to execute and hold its rotation. After a few seconds of no disturbance, the data collection was stopped for the 3DM-GX4-25 and QUARC.

2. Fast Motion in a Single Axis

For the second group of tests, the 3DM-GX4-25 was constrained to one positive rotation through a single axis at an angular rate greater than one radian per second. Three trials for each angle of rotation were conducted utilizing the same steps described in Section IV.B.1.

3. Arbitrary Slow Motion

For the third group of tests, the 3DM-GX4-25 was subjected to arbitrary movement in all three axes at an angular rate of no greater than 2.5 radians per second. Three trials were conducted. In order to perform the arbitrary movement and capture its time-sampled positions, the mounted 3DM-GX4-25 with infrared markers was placed on the ground near the origin of OptiTrack set during the groundplane calibration. The positive x-axis of the 3DM-GX4-25 was placed in line with the defined x-axis of OptiTrack. The data collection was started for the 3DM-GX4-25 and QUARC. After a few seconds of no disturbance, the mounted 3DM-GX4-25 with infrared markers was picked up from its horizontal position and arbitrarily rotated by hand through the workspace. Then, the mounted 3DM-GX4-25 with infrared markers was returned to its original position. After a few seconds of no disturbance, the data collection was stopped for the 3DM-GX4-25 and QUARC.

4. Arbitrary Fast Motion

For the fourth group of tests, the 3DM-GX4-25 was subjected to arbitrary movement in all three axes at an angular rate greater than 2.5 radians per second. Three trials were conducted utilizing the same steps described in Section IV.B.3.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS

The results from the testing methodology described in Chapter IV are presented in this chapter. First, the calculations for variation and error are defined. Then, the RMS values of the 3DM-GX4-25 for each test are presented in a table listing each trial. For select trials of each test, the time-sampled angles of rotation for the 3DM-GX4-25 and QUARC are overlaid in a plot and the difference of those angles is presented in a plot. Lastly, observations are discussed.

In order to formulate the results for comparison, the modified coordinate system of QUARC was rotated into the 3DM-GX4-25 coordinate system through a positive ninety degree rotation about the x-axis. This was accomplished by utilizing the rotation matrix from Equation (2.1) and calculating R_ϕ as

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos 90^\circ & \sin 90^\circ \\ 0 & -\sin 90^\circ & \cos 90^\circ \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}. \tag{5.1}$$

Next, R_ϕ was converted into a quaternion with a function written in MATLAB called *rotm2quat* utilizing Equation (2.7) and calculating the scalars as

$$\begin{aligned} q_0 &= \frac{1}{2}\sqrt{r_{11}+r_{22}+r_{33}+1} = \frac{1}{2}\sqrt{1+0+0+1} = \frac{\sqrt{2}}{2} \\ q_1 &= \frac{r_{23}-r_{32}}{4q_0} = \frac{1-(-1)}{4\cdot\left(\frac{\sqrt{2}}{2}\right)} = \frac{\sqrt{2}}{2} \\ q_2 &= \frac{r_{31}-r_{13}}{4q_0} = \frac{0-0}{4\cdot\left(\frac{\sqrt{2}}{2}\right)} = 0 \\ q_3 &= \frac{r_{12}-r_{21}}{4q_0} = \frac{0-0}{4\cdot\left(\frac{\sqrt{2}}{2}\right)} = 0. \end{aligned} \tag{5.2}$$

Finally, the calculated quaternion was multiplied with the quaternions outputted from QUARC utilizing the quaternion multiplication (*quatmultiply*) function in MATLAB.

With the quaternions of QUARC in the same coordinate system of the 3DM-GX4-25, all quaternions were converted into the respective angles of rotation utilizing the quaternion to rotation angles (*quat2angle*) function in MATLAB. Next, the angles of rotation from QUARC were shifted into the NED coordinate frame with the exact initial angle of the 3DM-GX4-25. This was accomplished by taking the difference of the initial angle from QUARC and the initial angle from the 3DM-GX4-25 and subtracting that computation from all the angles outputted from QUARC such that

$$angle_{QUARC,NED} = angle_{QUARC} - \left(initial\ angle_{QUARC} - initial\ angle_{3DM-GX\,4-25} \right). \quad (5.3)$$

Lastly, the data points from the 3DM-GX4-25 and QUARC were aligned manually.

The 3DM-GX4-25 and QUARC both experienced data drops but in different ways. For the 3DM-GX4-25, data drops were represented by a gap in the data in which no value was registered for that particular time sample; therefore, the interpolate (*interpl*) function was used in MATLAB to generate the missing data points of the 3DM-GX4-25 required to perform a comparison against the ground truth data in the MATLAB code. For QUARC, data drops were represented by the previous value repeating itself for that particular time sample. These values were utilized in the generation of plots but were excluded in error calculations. This method was selected over using the interpolate function to replace the repeating values to ensure the RMS value was only calculated with the ground truth data obtained directly from QUARC.

A. VARIATION AND ERROR CALCULATIONS

The variation and error were calculated utilizing standard deviation and the RMS value. Standard deviation, denoted as σ, measures the variation of data from its mean, μ. It is expressed by

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \mu\right)^2} \quad (5.4)$$

where n is the number of samples and x_i represents the data points. The mean is expressed by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (5.5)$$

where, once again, n is the number of samples and x_i represents the data points. The RMS value measures the magnitude of a set of data. It is expressed by

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \tag{5.6}$$

where n is the number of samples and x_i represents the data points.

1. No Disturbance

The 3DM-GX4-25 and OptiTrack experienced inherent variation despite no outside disturbance. This variation was captured by placing the mounted 3DM-GX4-25 with infrared markers into the field of view of OptiTrack. Without disturbing either system, the variation inherent to both the 3DM-GX4-25 and OptiTrack are seen in Figure 20 with the 3DM-GX4-25 on the left and OptiTrack on the right.

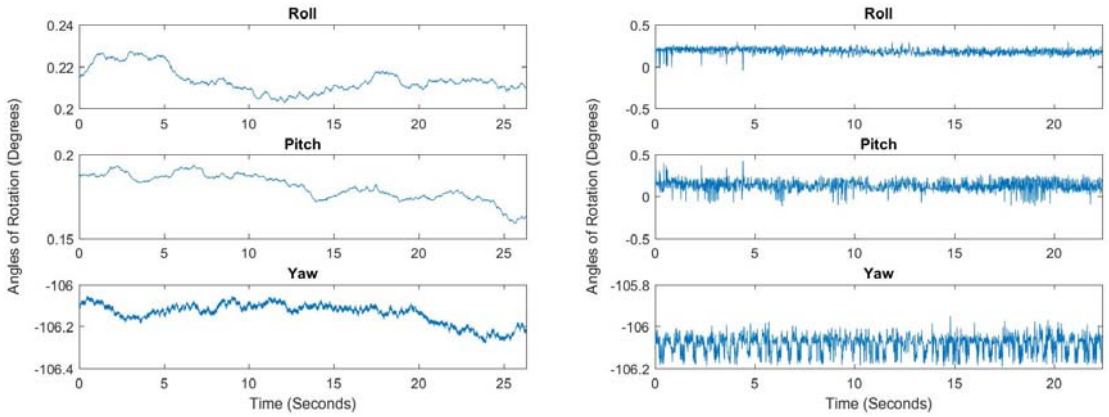

Figure 20. Inherent Variation of 3DM-GX4-25 and OptiTrack

With this variation, the standard deviation for each axis of rotation was calculated as minimal and is presented in Table 2.

Table 2. Standard Deviation of 3DM-GX4-25 and OptiTrack for No Disturbance

No Disturbance	3DM-GX4-25			OptiTrack		
	Roll	Pitch	Yaw	Roll	Pitch	Yaw
Trial 1	0.0061°	0.0078°	0.0499°	0.0366°	0.0638°	0.0438°

2. Disturbance

Since it was known that OptiTrack requires line-of-sight to the infrared markers, it was necessary to learn the sensitivity of blocking the field of view as well as other unavoidable factors while working in close proximity of the 3DM-GX4-25. These type of disturbances were captured by placing the mounted 3DM-GX4-25 with infrared markers on the ground in the field of view of OptiTrack. Following a few seconds of no disturbance, the field of view was entered. A tight circle was executed around the 3DM-GX4-25, and the field of view was exited. The simulation was stopped following a few seconds of no disturbance. Three trials were conducted. The vibration from walking in close proximity of the 3DM-GX4-25 produced a slightly higher standard deviation than in the case of no disturbance as seen in the three trials presented in Table 3. In addition, walking past each camera of OptiTrack and restricting the field of view increased its standard deviation compared to the case of no disturbance as seen in the three trials presented in Table 3.

Table 3. Standard Deviation of 3DM-GX4-25 and OptiTrack for a Disturbance

Disturbance	3DM-GX4-25			OptiTrack		
	Roll	Pitch	Yaw	Roll	Pitch	Yaw
Trial 1	0.0081°	0.0339°	0.0254°	0.0509°	0.1508°	0.0835°
Trial 2	0.0249°	0.0283°	0.0342°	0.1092°	0.1830°	0.0604°
Trial 3	0.0253°	0.0477°	0.0314°	0.1522°	0.1863°	0.0632°

Trial 3 is shown in Figure 21 where the disturbance of the 3DM-GX4-25 is shown on the left and OptiTrack is shown on the right.

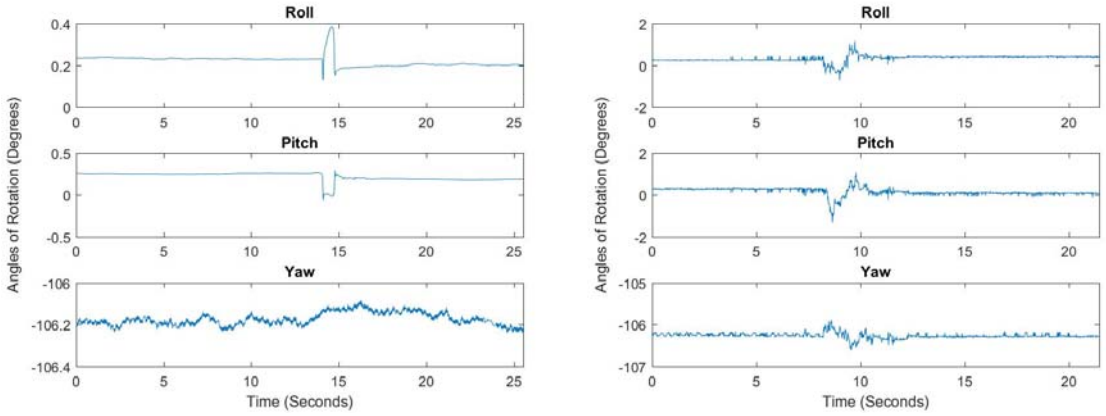

Figure 21. 3DM-GX4-25 and OptiTrack with a Disturbance

3. Masking

The last variation studied, relating only to OptiTrack, occurred when an infrared marker was completely obscured from all of the cameras' fields of view, which we call "masking." This variation was captured by placing the mounted 3DM-GX4-25 with infrared markers on the ground in the field of view of OptiTrack. After a few seconds of no disturbance, three of the five infrared markers were masked consecutively. Three trials were conducted. Masking resulted in a slightly higher standard deviation than in the case of no disturbance as seen in the three trials presented in Table 4.

Table 4. Standard Deviation of OptiTrack for Masking

Masking	OptiTrack		
	Roll	Pitch	Yaw
Trial 1	0.1778°	0.1753°	0.1235°
Trial 2	0.1314°	0.1038°	0.0713°
Trial 3	0.2958°	0.2701°	0.1113°

Trial 3 is shown in Figure 22. Looking at each angle of rotation, we see distinct peaks corresponding to an infrared marker being masked.

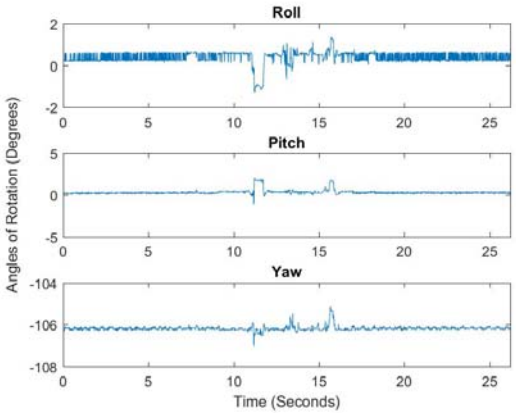

Figure 22. OptiTrack with Masking

B. RESULTS

The results of the testing methodology previously described in Chapter IV are contained in this section. The RMS values are presented in a table. For select trials, the angles of rotation from the 3DM-GX4-25 and OptiTrack are overlaid in a plot and the difference of those angles is presented in a plot.

1. Slow Motion in a Single Axis

The results of slow motion in a single axis of rotation for roll, pitch, and yaw as described in Chapter IV are contained in this subsection.

a. Roll

The RMS values of a slow rotation about the x-axis are presented in Table 5. The maximum angular rate for each trial was 0.6294 radians per second, 0.4601 radians per second, and 0.7397 radians per second, respectively. Recall from Chapter III, the performance specifications for dynamic accuracy was defined as ±2 degrees RMS. Based on the values in Table 5, the 3DM-GX4-25 meets the manufacturer's performance specifications.

Table 5. RMS Values for Slow Rotation about the x-axis

Roll Rotation	Roll	Pitch	Yaw
Trial 1	0.4962°	1.2691°	0.4137°
Trial 2	0.3958°	1.0856°	0.7240°
Trial 3	0.1645°	1.3395°	0.5231°

Trial 3 is shown in Figure 23, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right.

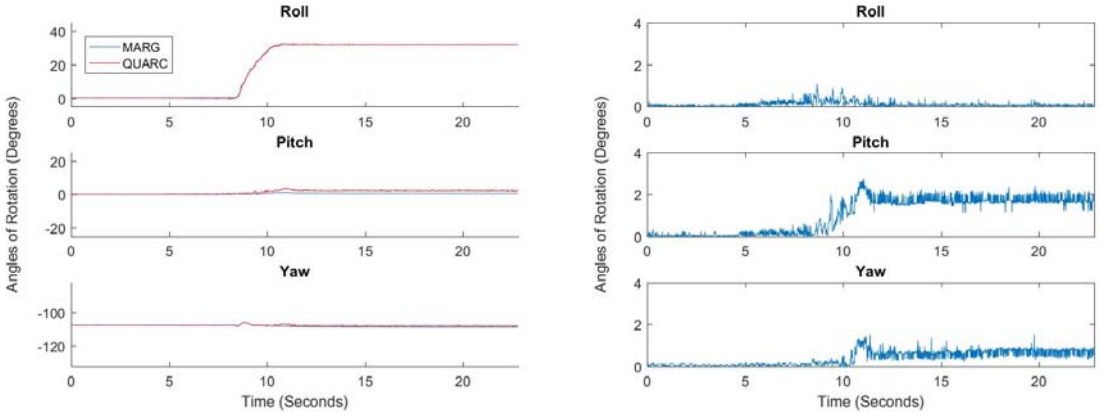

Figure 23. Trial 3 of a Slow Rotation about the x-axis

b. Pitch

The RMS values of a slow rotation about the y-axis are presented in Table 6. The maximum angular rate for each trial was 0.4868 radians per second, 0.8195 radians per second, and 0.6050 radians per second, respectively. Based on the values in Table 6, the 3DM-GX4-25 does not meet the manufacturer's performance specifications for the first trial. Yaw exceeded the performance specifications with a value of 2.0154 degrees.

Table 6. RMS Values for Slow Rotation about the y-axis

Pitch Rotation	Roll	Pitch	Yaw
Trial 1	0.8962°	0.6251°	2.0154°
Trial 2	1.7001°	0.2445°	1.9249°
Trial 3	1.2505°	0.4791°	1.9136°

Trial 1 is shown in Figure 24, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right. During this trial, OptiTrack experienced a data drop between 12.67 seconds and 12.71 seconds. The data points during this timeframe were excluded in the calculations of RMS.

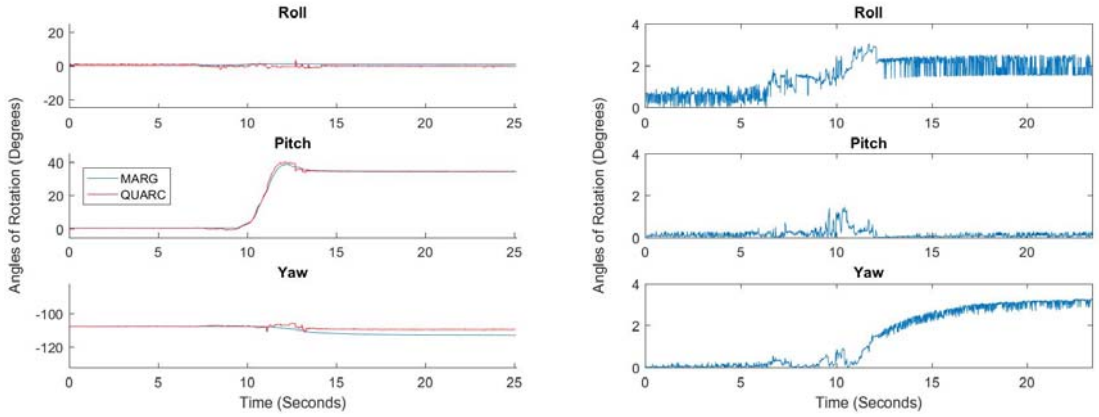

Figure 24. Trial 1 of a Slow Rotation about the y-axis

c. Yaw

The RMS values of a slow rotation about the z-axis are presented in Table 7. The maximum angular rate for each trial was 0.4186 radians per second, 0.4394 radians per second, and 0.3529 radians per second, respectively. Based on the values in Table 7, the 3DM-GX4-25 meets the manufacturer's performance specifications.

Table 7. RMS Values for Slow Rotation about the z-axis

Yaw Rotation	Roll	Pitch	Yaw
Trial 1	1.1264°	0.9209°	1.7158°
Trial 2	0.4355°	0.7235°	1.3363°
Trial 3	1.1039°	1.0676°	1.3425°

Trial 1 is shown in Figure 25, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right.

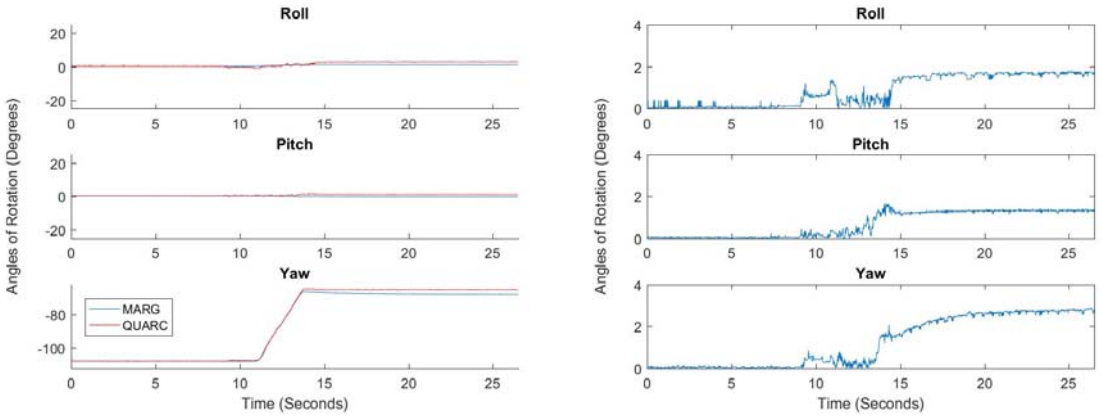

Figure 25. Trial 1 of a Slow Rotation about the y-axis

2. Fast Rotation in a Single Axis

The results of fast motion in a single axis of rotation for roll, pitch, and yaw as described in Chapter IV are contained in this subsection.

a. Roll

The RMS values of a fast rotation about the x-axis are presented in Table 8. The maximum angular rate for each trial was 3.1699 radians per second, 3.3616 radians per second, and 2.5993 radians per second, respectively. Based on the values in Table 8, the 3DM-GX4-25 meets the manufacturer's performance specifications.

Table 8. RMS Values for Fast Rotation about the x-axis

Roll Rotation	Roll	Pitch	Yaw
Trial 1	0.6269°	1.2985°	0.4241°
Trial 2	0.2370°	1.5546°	0.5456°
Trial 3	0.5028°	1.4252°	0.6020°

Trial 1 is shown in Figure 26, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right.

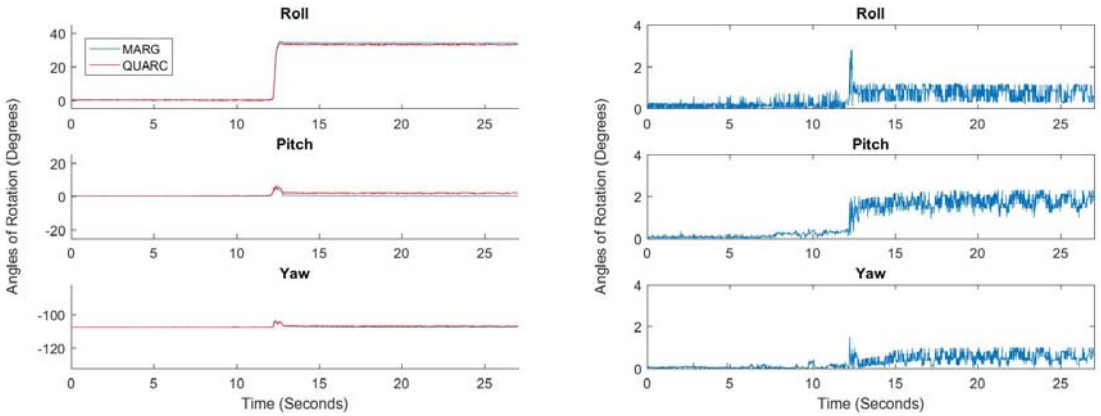

Figure 26. Trial 1 of a Fast Rotation about the x-axis

b. Pitch

The RMS values of a fast rotation about the y-axis are presented in Table 9. The maximum angular rate for each trial was 3.4578 radians per second, 2.7673 radians per second, and 4.8021 radians per second, respectively. Based on the values in Table 9, the 3DM-GX4-25 meets the manufacturer's performance specifications.

Table 9. RMS Values for Fast Rotation about the y-axis

Pitch Rota	Roll	Pitch	Yaw
Trial 1	1.2414°	0.5935°	1.7882°
Trial 2	1.0707°	0.6050°	1.6012°
Trial 3	1.5314°	0.6350°	1.5596°

Trial 3 is shown in Figure 27, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right.

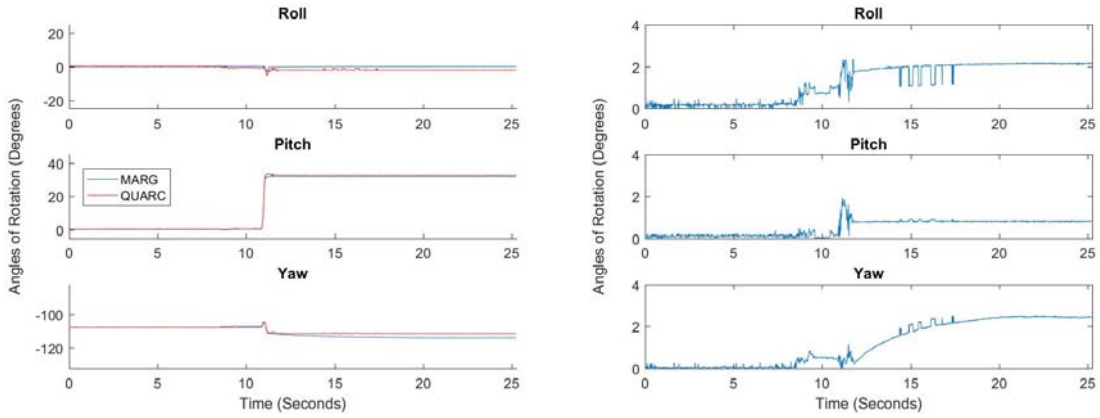

Figure 27. Trial 3 of a Fast Rotation about the y-axis

c. Yaw

The RMS values of a fast rotation about the z-axis are presented in Table 10. The maximum angular rate for each trial was 4.0985 radians per second, 5.3000 radians per second, and 4.9538 radians per second, respectively. Based on the values in Table 10, the 3DM-GX4-25 meets the manufacturer's performance specifications.

Table 10. RMS Values for Fast Rotation about the z-axis

Yaw Rotation	Roll	Pitch	Yaw
Trial 1	0.7118°	0.9507°	1.3776°
Trial 2	0.9410°	0.6218°	0.7619°
Trial 3	0.8834°	0.9526°	0.7227°

Trial 1 is shown in Figure 28, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right.

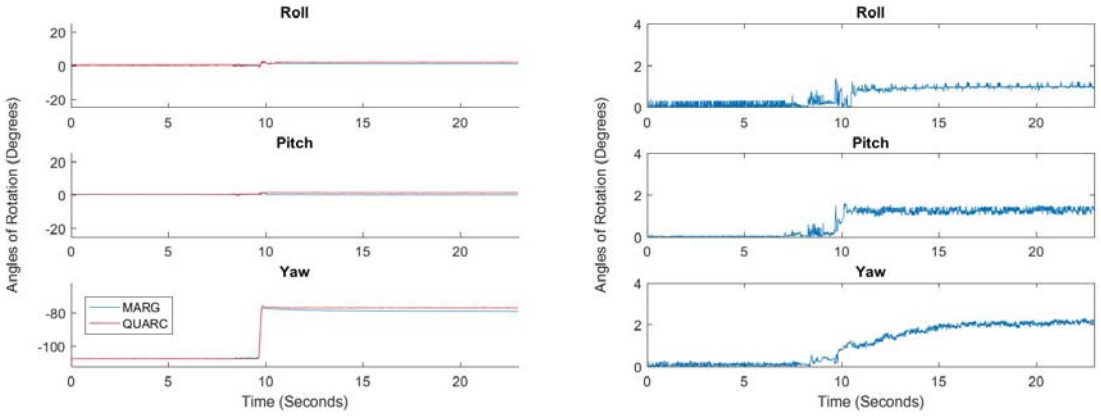

Figure 28. Trial 1 of a Fast Rotation about the z-axis

3. Arbitrary Slow Motion

The results of the arbitrary slow rotations as described in Chapter IV are contained in this subsection. The RMS values for arbitrary slow rotations are presented in Table 11. The maximum angular rate for each trial was 2.3528 radians per second, 1.6555 radians per second, and 2.3025 radians per second, respectively. Based on the values in Table 11, the 3DM-GX4-25 does not meet the manufacturer's performance specifications for all trials. In each instance, yaw more than doubled the performance specifications.

Table 11. RMS Values for Arbitrary Slow Rotations

Arbitrary	Roll	Pitch	Yaw
Trial 1	1.5666°	1.5689°	4.1489°
Trial 2	1.1306°	1.9698°	4.7443°
Trial 3	0.7729°	1.2588°	4.0237°

Trial 2 is shown in Figure 29, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right.

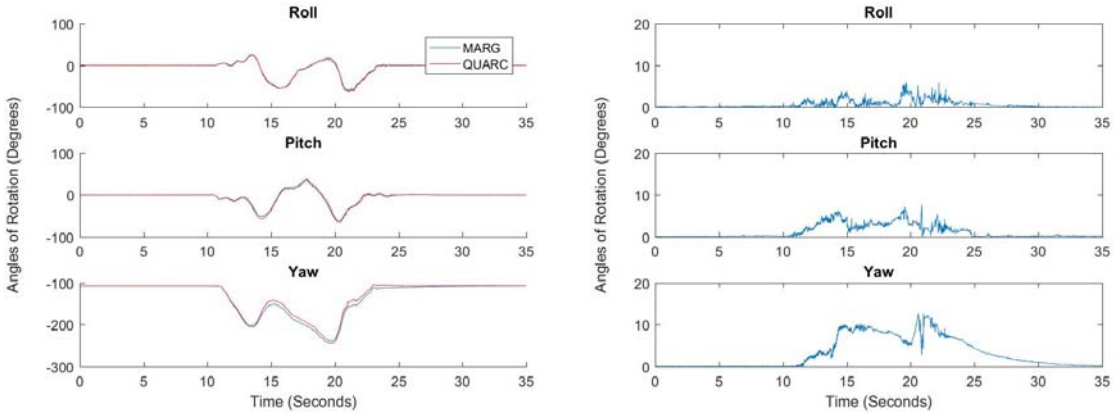

Figure 29. Trial 2 of Arbitrary Slow Rotations

4. Arbitrary Fast Motion

The results of the arbitrary fast rotations as described in Chapter IV are contained in this subsection. The RMS value for arbitrary fast rotations are presented in Table 12. The maximum angular rate for each trial was 7.1996 radians per second, 9.1706 radians per second, and 8.7432 radians per second, respectively. Based on the values in Table 12, the 3DM-GX4-25 does not meet the manufacturer's performance specifications for the third trial. Yaw exceeded the performance specifications with a value of 2.5852 degrees.

Table 12. RMS Values for Arbitrary Fast Rotations

Arbitrary	Roll	Pitch	Yaw
Trial 1	0.7509°	0.5129°	1.7630°
Trial 2	0.9118°	0.6937°	1.6343°
Trial 3	1.3719°	0.8641°	2.5852°

Trial 2 is shown in Figure 30, where the overlaid angles of rotation from the 3DM-GX4-25 and OptiTrack are on the left and the difference of those angles are on the right. In this trial, OptiTrack experienced a data drop between 8.24 seconds and 8.33 seconds. The data drop is the cause for the spike seen in the difference plot. A zoomed-in view of the angle of rotation for roll is presented in Figure 31 which highlights the data drop. The data points during this timeframe were excluded in the calculations of RMS.

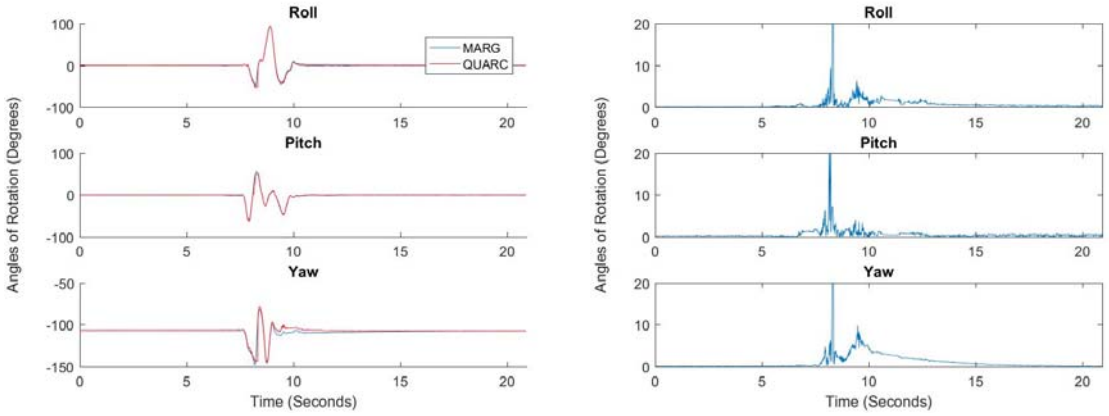

Figure 30. Trial 2 of Arbitrary Fast Rotations

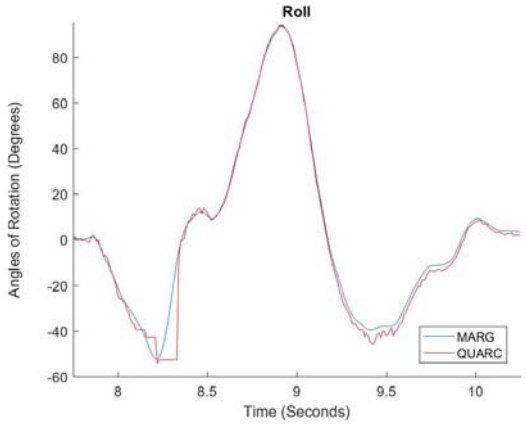

Figure 31. Roll Angle of Rotation with Data Drop

C. OBSERVATIONS

Based on the collected data from the CF, the 3DM-GX4-25 often met the performance specifications for dynamic accuracy; however, instances occurred in which the RMS value exceeded the manufacturer's performance specifications. Additionally, instantaneous errors grew larger based on the complexity of the motion.

For all trials, roll and pitch remained under the manufacturer's performance specification of ± 2 degrees RMS; however, yaw exceeded this value for some of the trials. Furthermore, yaw typically experienced the largest RMS values when compared to roll and pitch. These errors in yaw were especially prevalent during the arbitrary slow

rotations where the RMS values were more than double the manufacturer's performance specifications. This error is attributed to the design of the 3DM-GX4-25 in which yaw depends on sensor measurements taken from the magnetometer. These values are then passed through the 3DM-GX4-25's filters, which creates a lag [16]. This lag became prevalent once a disturbance stopped. A zoomed-in plot of the yaw angle of rotation taken from trial 1 of a fast rotation about the z-axis is shown in Figure 32. The lag begins at 9.94 seconds following the cessation of the rotation.

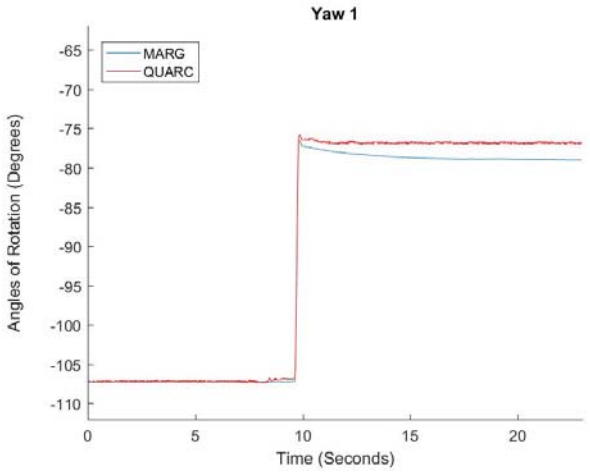

Figure 32. Yaw Angle of Rotation from Trial 1 of a Fast Rotation about the z-axis

Although instantaneous error exceeded the RMS value in each trial, the magnitude of the instantaneous error was based on the complexity of motion. For the slow and fast rotations in a single axis, the maximum instantaneous error was 3.6041 degrees and 3.3608 degrees, respectively; however, for the slow and fast arbitrary motions, the maximum instantaneous error was 20.0944 degrees and 22.6795 degrees, respectively. Furthermore, this larger instantaneous error typically occurred during a transition between a positive rotation and negative rotation. A zoomed-in plot of the yaw angle of rotation taken from trial 1 of the arbitrary slow motion is seen in Figure 32. This occurrence is especially prevalent at 14.34 seconds where the instantaneous error reached 14.9478 degrees and at 21.86 seconds where the instantaneous error reached 20.0944 degrees.

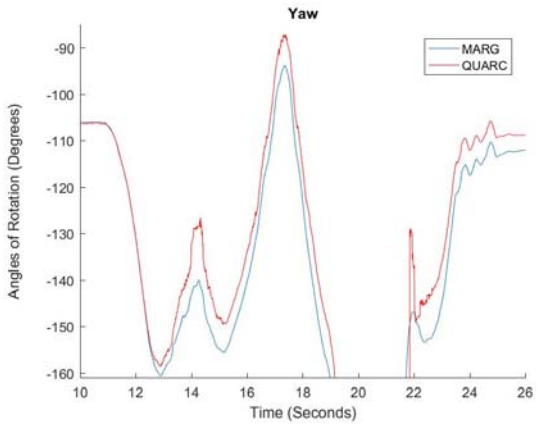

Figure 33. Yaw Angle of Rotation from Trial 1 of Arbitrary Slow Rotations

In the rotations about a single axis for both slow and fast motion, the difference of the angles between the 3DM-GX4-25 and OptiTrack were predominantly greater following the completion of a rotation compared to before the rotation despite the 3DM-GX4-25 and OptiTrack no longer experiencing a disturbance. More specifically, for seventeen of the eighteen trials the difference was larger in the two axes not experiencing a rotation. Examples can be seen in Figures 22 through 27. In fifteen of the trials, the difference was larger for all three angles of rotation. Examples can be seen in Figures 24 through 27. Lastly, in four of the six trials of a rotation about the x-axis for both slow and fast motion, the RMS value for roll was the smallest compared to the other angles of rotation as seen in Table 5 and Table 8. In all six cases of a rotation about the y-axis for both slow and fast motion, the RMS value for pitch was smallest compared to the other angles of rotation as seen in Table 6 and Table 9. However, in four of the six cases of a rotation about the z-axis for both slow and fast motion, the RMS value for yaw was the largest compared to its other angles of rotation as seen in Table 7 and Table 10.

Finally, data drops in OptiTrack were more prevalent with increased speed and complexity of motion. Only one data drop occurred during rotations in a single axis as previously shown in Section V.B.1.b; however, in the six trials of arbitrary motion, four experienced multiple data drops.

VI. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

A. CONCLUSIONS

In this thesis, a series of dynamic accuracy tests for arbitrary motions in three-dimensional space were developed and implemented. The NaturalPoint OptiTrack 2.3.3 Motion Capture System was used as a system of reference, and those measurements were compared to those of the LORD MicroStrain® 3DM-GX4-25 via MATLAB/Simulink R2011b. From this comparison, the dynamic accuracy of the 3DM-GX4-25 was determined and compared to other MARG sensor modules used in the aforementioned NPS theses.

Although the tests performed on other MARG sensor modules in the previous NPS theses differ from the tests performed in this thesis, the conclusions from those tests can be compared. The 3DM-GX1 and 3DM-GX3-25 performed similarly to the 3DM-GX4-25. All three MARG sensor modules met the dynamic accuracy performance specifications of ± 2 degrees RMS for most cases but experienced large instantaneous errors. Furthermore, the 3DM-GX3-25 and 3DM-GX4-25 both experienced larger RMS values for yaw in conjunction with a lag. The 3DM-GX1 does not experience this lag following the cessation of dynamic motion. The Xsens MTx sensor module performed similarly to the 3DM-GX4-25 as well. Both sensor modules met the dynamic accuracy performance specifications of ± 2 degrees RMS for roll and pitch; however, yaw failed in certain instances. In addition, larger instantaneous errors in the Xsens MTx occurred for faster motion, whereas larger instantaneous errors in the 3DM-GX4-25 occurred for more complex motions. Finally, the YEI-3-space data-logging sensor was validated to ± 1 degrees utilizing a VICON motion capture system.

Based on the results from the tests performed on all the MARG sensor modules, the YEI 3-space data-logging sensor exceeded the dynamic accuracy of the 3DM-GX4-25. Furthermore, the cost of the YEI 3-space data-logging sensor was significantly lower as compared to the 3DM-GX4-25. In consideration of the performance and cost of the 3DM-GX4-25, the dynamic accuracy of the computed outputs from the CF is not

justified; however, if the dynamic accuracy performance specifications of the computed outputs from the AKF can be validated, use of the 3DM-GX4-25 may be justified. Until that happens, Reticle should continue to use the YEI 3-space data-logging sensor.

B. FUTURE WORK

Although the goals set forth in this thesis were accomplished, there is still work that can be done pertaining to this topic. Despite developing a series of dynamic accuracy tests for arbitrary motions in three-dimensional space, the exact motion of a trial could not be replicated because rotations were executed by hand; therefore, a physical rotation system with the ability to replicate exact motions in three-dimensional space must be developed that does not introduce magnetic anomalies.

Due to the limitations noted in Chapter III, only the quaternions outputted from the CF were studied; however, the manufacturer's performance specifications claim a lower RMS value of ± 0.25 degrees RMS for roll and pitch and ± 0.8 for yaw when the AKF outputs are utilized. A method needs to be devised to adequately separate the outputted data from the AKF for analysis, after which the manufacturer's performance specifications need validation.

Finally, the versions of OptiTrack and QUARC utilized for this thesis presented significant challenges in aligning the coordinate system to the 3DM-GX4-25 coordinate system and required much communication with Quanser engineers. In addition, OptiTrack experienced inherent error that increased due to its reliance on line-of-sight as discussed in Chapter V. Finally, data drops occurred that seemed to increase with the complexity of the motion of the trackable; therefore, the tests described in this thesis should be repeated with another motion capture system of better caliber.

APPENDIX A. MATLAB CODE FOR NO DISTURBANCE/DISTURBANCE

A. CODE

The code used to capture inherent error and errors induced due to unavoidable factors is contained in this section.

```
%No Disturbance/Disturbance
%By:  Heather Pelachick

%This code utilizies sensor measurements from the 3DM-GX4-25 and QUARC
%to determine the inherent error, error due to a disturbance,
%and angles of rotation plot.

clear all;
close all;

%----------------------------- IMU ----------------------------------------

%Load Data - Quaternion
RNG='M17..P1768';
disturb_1_IMU=csvread('Disturb_IMU_1.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(disturb_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_disturb_1=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C1768';
t_IMU_disturb_1=csvread('Disturb_IMU_1.csv',16,2,RNG);
t_IMU_disturb_1=t_IMU_disturb_1-8277.115; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I1768';
gyro=csvread('Disturb_IMU_1.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_disturb_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_disturb_1,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_disturb_1(length(t_IMU_disturb_1))])
subplot(3,1,2),plot(t_IMU_disturb_1,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_disturb_1(length(t_IMU_disturb_1))])
subplot(3,1,3),plot(t_IMU_disturb_1,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_disturb_1(length(t_IMU_disturb_1))])
```

```
%Rename variable
disturb_1_IMU_deg=RPY_deg;

std(disturb_1_IMU_deg(1,:))
std(disturb_1_IMU_deg(2,:))
std(disturb_1_IMU_deg(3,:))

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2438';
disturb_2_IMU=csvread('Disturb_IMU_2.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(disturb_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_disturb_2=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C2438';
t_IMU_disturb_2=csvread('Disturb_IMU_2.csv',16,2,RNG);
t_IMU_disturb_2=t_IMU_disturb_2-9204.535; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2438';
gyro=csvread('Disturb_IMU_2.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_disturb_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_disturb_2,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_disturb_2(length(t_IMU_disturb_2))])
subplot(3,1,2),plot(t_IMU_disturb_2,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_disturb_2(length(t_IMU_disturb_2))])
subplot(3,1,3),plot(t_IMU_disturb_2,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_disturb_2(length(t_IMU_disturb_2))])

%Rename variable
disturb_2_IMU_deg=RPY_deg;

std(disturb_2_IMU_deg(1,:))
std(disturb_2_IMU_deg(2,:))
std(disturb_2_IMU_deg(3,:))

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P1213';
disturb_3_IMU=csvread('Disturb_IMU_3.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(disturb_3_IMU);
```

```
%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_disturb_3=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C1213';
t_IMU_disturb_3=csvread('Disturb_IMU_3.csv',16,2,RNG);
t_IMU_disturb_3=t_IMU_disturb_3-9556.181; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I1213';
gyro=csvread('Disturb_IMU_3.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_disturb_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_disturb_3,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_disturb_3(length(t_IMU_disturb_3))])
subplot(3,1,2),plot(t_IMU_disturb_3,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_disturb_3(length(t_IMU_disturb_3))])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(t_IMU_disturb_3,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_disturb_3(length(t_IMU_disturb_3))])
xlabel('Time (Seconds)')

saveas(gcf,'IMU_Disturb.jpeg');

%Rename variable
disturb_3_IMU_deg=RPY_deg;

std(disturb_3_IMU_deg(1,:))
std(disturb_3_IMU_deg(2,:))
std(disturb_3_IMU_deg(3,:))

%--------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3143';
mask_1_IMU=csvread('Mask_IMU_1.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(mask_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_mask_1=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C3143';
```

```
t_IMU_mask_1=csvread('Mask_IMU_1.csv',16,2,RNG);
t_IMU_mask_1=t_IMU_mask_1-11100.777; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3143';
gyro=csvread('Mask_IMU_1.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_mask_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_mask_1,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_mask_1(length(t_IMU_mask_1))])
subplot(3,1,2),plot(t_IMU_mask_1,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_mask_1(length(t_IMU_mask_1))])
subplot(3,1,3),plot(t_IMU_mask_1,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_mask_1(length(t_IMU_mask_1))])

%Rename variable
mask_1_IMU_deg=RPY_deg;

std(mask_1_IMU_deg(1,:))
std(mask_1_IMU_deg(2,:))
std(mask_1_IMU_deg(3,:))

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3098';
mask_2_IMU=csvread('Mask_IMU_2.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(mask_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_mask_2=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C3098';
t_IMU_mask_2=csvread('Mask_IMU_2.csv',16,2,RNG);
t_IMU_mask_2=t_IMU_mask_2-10330.159; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3098';
gyro=csvread('Mask_IMU_2.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_mask_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_mask_2,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_mask_2(length(t_IMU_mask_2))])
subplot(3,1,2),plot(t_IMU_mask_2,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_mask_2(length(t_IMU_mask_2))])
```

```
subplot(3,1,3),plot(t_IMU_mask_2,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_mask_2(length(t_IMU_mask_2))])

%Rename variable
mask_2_IMU_deg=RPY_deg;

std(mask_2_IMU_deg(1,:))
std(mask_2_IMU_deg(2,:))
std(mask_2_IMU_deg(3,:))

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2034';
mask_3_IMU=csvread('Mask_IMU_3.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(mask_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_mask_3=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C2034';
t_IMU_mask_3=csvread('Mask_IMU_3.csv',16,2,RNG);
t_IMU_mask_3=t_IMU_mask_3-10609.191; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2034';
gyro=csvread('Mask_IMU_3.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_mask_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_mask_3,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_mask_3(length(t_IMU_mask_3))])
%ylabel('Angle (Degrees)')
subplot(3,1,2),plot(t_IMU_mask_3,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_mask_3(length(t_IMU_mask_3))])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(t_IMU_mask_3,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_mask_3(length(t_IMU_mask_3))])
%ylabel('Angle (Degrees)')
xlabel('Time (Seconds)')

%Rename variable
mask_3_IMU_deg=RPY_deg;

std(mask_3_IMU_deg(1,:))
std(mask_3_IMU_deg(2,:))
std(mask_3_IMU_deg(3,:))

%-------------------------------------------------------------------------
```

```
%Load Data - Quaternion
RNG='M17..P2511';
no_disturb_IMU=csvread('No_Disturb.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(no_disturb_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_no_disturb=RPY_deg(:,1);

%Load Data - Time
RNG='C17..C2511';
t_IMU_no_disturb=csvread('No_Disturb.csv',16,2,RNG);
t_IMU_no_disturb=t_IMU_no_disturb-11423.933; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2511';
gyro=csvread('No_Disturb.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_no_disturb(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
figure
subplot(3,1,1),plot(t_IMU_no_disturb,RPY_deg(1,:)),title('Roll')
xlim([0 t_IMU_no_disturb(length(t_IMU_no_disturb))])
subplot(3,1,2),plot(t_IMU_no_disturb,RPY_deg(2,:)),title('Pitch')
xlim([0 t_IMU_no_disturb(length(t_IMU_no_disturb))])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(t_IMU_no_disturb,RPY_deg(3,:)),title('Yaw')
xlim([0 t_IMU_no_disturb(length(t_IMU_no_disturb))])
xlabel('Time (Seconds)')

saveas(gcf,'IMU_NoDisturb.jpeg');

%Rename variable
no_disturb_IMU_deg=RPY_deg;

std(no_disturb_IMU_deg(1,:))
std(no_disturb_IMU_deg(2,:))
std(no_disturb_IMU_deg(3,:))

%------------------------------ QUARC -------------------------------------

%Rotation Matrix
rotm=[1 0 0; 0 0 1; 0 -1 0];

%Function to convert rotation matrix to quaternion
quat=rotm2quat(rotm);

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('Disturb_quarc_1.mat')

%Change variable name of loaded data
```

```
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_disturb_1=RPY_deg(:,1);

%Load Data - Time
load('t_disturb_1');
t_quarc_disturb_1=mytime;

%Plot
figure
subplot(3,1,1),plot(t_quarc_disturb_1,RPY_deg(1,:)-(deg1_disturb_1(1)-...
    deg_disturb_1(1))),title('Roll')
xlim([0 length(t_quarc_disturb_1)/100])
subplot(3,1,2),plot(t_quarc_disturb_1,RPY_deg(2,:)-(deg1_disturb_1(2)-...
    deg_disturb_1(2))),title('Pitch')
xlim([0 length(t_quarc_disturb_1)/100])
subplot(3,1,3),plot(t_quarc_disturb_1,RPY_deg(3,:)-...
    (deg1_disturb_1(3)-deg_disturb_1(3))),title('Yaw')
xlim([0 length(t_quarc_disturb_1)/100])

%Rename variable
disturb_1_quarc_deg=RPY_deg;

std(disturb_1_quarc_deg(1,:))
std(disturb_1_quarc_deg(2,:))
std(disturb_1_quarc_deg(3,:))

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('Disturb_quarc_2.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];
```

```
%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_disturb_2=RPY_deg(:,1);

%Load Data - Time
load('t_disturb_2');
t_quarc_disturb_2=mytime;

%Plot
figure
subplot(3,1,1),plot(t_quarc_disturb_2,RPY_deg(1,:)-...
    (deg1_disturb_2(1)-deg_disturb_2(1))),title('Roll')
xlim([0 length(t_quarc_disturb_2)/100])
subplot(3,1,2),plot(t_quarc_disturb_2,RPY_deg(2,:)-...
    (deg1_disturb_2(2)-deg_disturb_2(2))),title('Pitch')
xlim([0 length(t_quarc_disturb_2)/100])
subplot(3,1,3),plot(t_quarc_disturb_2,RPY_deg(3,:)-...
    (deg1_disturb_2(3)-deg_disturb_2(3))),title('Yaw')
xlim([0 length(t_quarc_disturb_2)/100])

%Rename variable
disturb_2_quarc_deg=RPY_deg;

std(disturb_2_quarc_deg(1,:))
std(disturb_2_quarc_deg(2,:))
std(disturb_2_quarc_deg(3,:))

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('Disturb_quarc_3.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_disturb_3=RPY_deg(:,1);

%Load Data - Time
load('t_disturb_3');
t_quarc_disturb_3=mytime;

%Plot
figure
```

```
subplot(3,1,1),plot(t_quarc_disturb_3,RPY_deg(1,:)-...
    (deg1_disturb_3(1)-deg_disturb_3(1))),title('Roll')
xlim([0 length(t_quarc_disturb_3)/100])
subplot(3,1,2),plot(t_quarc_disturb_3,RPY_deg(2,:)-...
    (deg1_disturb_3(2)-deg_disturb_3(2))),title('Pitch')
xlim([0 length(t_quarc_disturb_3)/100])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(t_quarc_disturb_3,RPY_deg(3,:)-...
    (deg1_disturb_3(3)-deg_disturb_3(3))),title('Yaw')
xlim([0 length(t_quarc_disturb_3)/100])
xlabel('Time (Seconds)')

saveas(gcf,'Quarc_Disturb.jpeg');

%Rename variable
disturb_3_quarc_deg=RPY_deg;

std(disturb_3_quarc_deg(1,:))
std(disturb_3_quarc_deg(2,:))
std(disturb_3_quarc_deg(3,:))

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('Mask_quarc_1.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_mask_1=RPY_deg(:,1);

%Load Data - Time
load('t_mask_1');
t_quarc_mask_1=mytime;

%Plot
figure
subplot(3,1,1),plot(t_quarc_mask_1,RPY_deg(1,:)-...
    (deg1_mask_1(1)-deg_mask_1(1))),title('Roll')
xlim([0 length(t_quarc_mask_1)/100])
subplot(3,1,2),plot(t_quarc_mask_1,RPY_deg(2,:)-...
    (deg1_mask_1(2)-deg_mask_1(2))),title('Pitch')
xlim([0 length(t_quarc_mask_1)/100])
subplot(3,1,3),plot(t_quarc_mask_1,RPY_deg(3,:)-...
    (deg1_mask_1(3)-deg_mask_1(3))),title('Yaw')
xlim([0 length(t_quarc_mask_1)/100])
```

```
%Rename variable
mask_1_quarc_deg=RPY_deg;

std(mask_1_quarc_deg(1,:))
std(mask_1_quarc_deg(2,:))
std(mask_1_quarc_deg(3,:))

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('Mask_quarc_2.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_mask_2=RPY_deg(:,1);

%Load Data - Time
load('t_mask_2');
t_quarc_mask_2=mytime;

%Plot
figure
subplot(3,1,1),plot(t_quarc_mask_2,RPY_deg(1,:)-...
    (deg1_mask_2(1)-deg_mask_2(1))),title('Roll')
xlim([0 length(t_quarc_mask_2)/100])
subplot(3,1,2),plot(t_quarc_mask_2,RPY_deg(2,:)-...
    (deg1_mask_2(2)-deg_mask_2(2))),title('Pitch')
xlim([0 length(t_quarc_mask_2)/100])
subplot(3,1,3),plot(t_quarc_mask_2,RPY_deg(3,:)-...
    (deg1_mask_2(3)-deg_mask_2(3))),title('Yaw')
xlim([0 length(t_quarc_mask_2)/100])

%Rename variable
mask_2_quarc_deg=RPY_deg;

std(mask_2_quarc_deg(1,:))
std(mask_2_quarc_deg(2,:))
std(mask_2_quarc_deg(3,:))

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('Mask_quarc_3.mat')
```

```
%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_mask_3=RPY_deg(:,1);

%Load Data - Time
load('t_mask_3');
t_quarc_mask_3=mytime;

%Plot
figure
subplot(3,1,1),plot(t_quarc_mask_3,RPY_deg(1,:)-...
    (deg1_mask_3(1)-deg_mask_3(1))),title('Roll')
xlim([0 length(t_quarc_mask_3)/100])
subplot(3,1,2),plot(t_quarc_mask_3,RPY_deg(2,:)-...
    (deg1_mask_3(2)-deg_mask_3(2))),title('Pitch')
xlim([0 length(t_quarc_mask_3)/100])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(t_quarc_mask_3,RPY_deg(3,:)-...
    (deg1_mask_3(3)-deg_mask_3(3))),title('Yaw')
xlim([0 length(t_quarc_mask_3)/100])
xlabel('Time (Seconds)')

saveas(gcf,'Quarc_Mask.jpeg');

%Rename variable
mask_3_quarc_deg=RPY_deg;

std(mask_3_quarc_deg(1,:))
std(mask_3_quarc_deg(2,:))
std(mask_3_quarc_deg(3,:))

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('No_Disturb_quarc.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);
```

```
%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_no_disturb=RPY_deg(:,1);

%Load Data - Time
load('t_No_Disturb');
t_quarc_no_disturb=mytime;

%Plot
figure
subplot(3,1,1),plot(t_quarc_no_disturb,RPY_deg(1,:)-...
    (deg1_no_disturb(1)-deg_no_disturb(1))),title('Roll')
xlim([0 length(t_quarc_no_disturb)/100])
subplot(3,1,2),plot(t_quarc_no_disturb,RPY_deg(2,:)-...
    (deg1_no_disturb(2)-deg_no_disturb(2))),title('Pitch')
xlim([0 length(t_quarc_no_disturb)/100])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(t_quarc_no_disturb,RPY_deg(3,:)-...
    (deg1_no_disturb(3)-deg_no_disturb(3))),title('Yaw')
xlim([0 length(t_quarc_no_disturb)/100])
xlabel('Time (Seconds)')

saveas(gcf,'Quarc_NoDisturb.jpeg');

%Rename variable
no_disturb_quarc_deg=RPY_deg;

std(no_disturb_quarc_deg(1,:))
std(no_disturb_quarc_deg(2,:))
std(no_disturb_quarc_deg(3,:))
```

B. FUNCTION

The function used in conjuction with the code for no disturbance/disturbance is contained in this section.

```
%Rotation Matrix to Quaternion
%By:  Heather Pelachick

function [quat]=rotm2quat(r)

q0=(1/2)*sqrt(r(1,1)+r(2,2)+r(3,3)+1);
q1=-(r(3,2)-r(2,3))/(4*q0);
q2=-(r(1,3)-r(3,1))/(4*q0);
q3=-(r(2,1)-r(1,2))/(4*q0);

quat=[q0 q1 q2 q3];
```

APPENDIX B. MATLAB CODE FOR SLOW MOTION IN A SINGLE AXIS

A. CODE

The code used during the slow motion in a single axis is contained in this section.

```
%Slow Motion in a Singe Axis
%By:  Heather Pelachick


%This code utilizes sensor measurements from the 3DM-GX4-25 and QUARC
%to determine the dynamic accuracy, angular rate, peak instantaneous error,
%angles of rotation plot, and difference plot for slow motions in a single
%axis.

clear all;
close all;

%---------------------------- IMU ---------------------------------------

%Load Data - Quaternion
RNG='M17..P1978';
roll_1_IMU=csvread('roll_1_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_roll_1=RPY_deg(:,520);

%Load Data - Time
RNG='C17..C1978';
t_IMU_roll_1=csvread('roll_1_IMU_7Sep.csv',16,2,RNG);
t_IMU_roll_1=t_IMU_roll_1-14760.721; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I1978';
gyro=csvread('roll_1_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_roll_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
roll_1_IMU_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2442';
roll_2_IMU=csvread('roll_2_IMU_7Sep.csv',16,12,RNG);
```

```
%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_roll_2=RPY_deg(:,487);

%Load Data - Time
RNG='C17..C2442';
t_IMU_roll_2=csvread('roll_2_IMU_7Sep.csv',16,2,RNG);
t_IMU_roll_2=t_IMU_roll_2-15132.487; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2442';
gyro=csvread('roll_2_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_roll_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
roll_2_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2697';
roll_3_IMU=csvread('roll_3_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_roll_3=RPY_deg(:,495);

%Load Data - Time
RNG='C17..C2697';
t_IMU_roll_3=csvread('roll_3_IMU_7Sep.csv',16,2,RNG);
t_IMU_roll_3=t_IMU_roll_3-15472.463; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2697';
gyro=csvread('roll_3_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_roll_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
roll_3_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------
```

```
%Load Data - Quaternion
RNG='M17..P2869';
pitch_1_IMU=csvread('pitch_1_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(pitch_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_pitch_1=RPY_deg(:,477);

%Load Data - Time
RNG='C17..C2869';
t_IMU_pitch_1=csvread('pitch_1_IMU_7Sep.csv',16,2,RNG);
t_IMU_pitch_1=t_IMU_pitch_1-15965.669; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2869';
gyro=csvread('pitch_1_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_pitch_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
pitch_1_IMU_deg=RPY_deg;

%------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2770';
pitch_2_IMU=csvread('pitch_2_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(pitch_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_pitch_2=RPY_deg(:,495);

%Load Data - Time
RNG='C17..C2770';
t_IMU_pitch_2=csvread('pitch_2_IMU_7Sep.csv',16,2,RNG);
t_IMU_pitch_2=t_IMU_pitch_2-16314.175; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2770';
gyro=csvread('pitch_2_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_pitch_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
```

```
pitch_2_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3078';
pitch_3_IMU=csvread('pitch_3_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(pitch_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_pitch_3=RPY_deg(:,488);

%Load Data - Time
RNG='C17..C3078';
t_IMU_pitch_3=csvread('pitch_3_IMU_7Sep.csv',16,2,RNG);
t_IMU_pitch_3=t_IMU_pitch_3-16717.611; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3078';
gyro=csvread('pitch_3_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_pitch_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
pitch_3_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3042';
yaw_1_IMU=csvread('yaw_1_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(yaw_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_yaw_1=RPY_deg(:,530);

%Load Data - Time
RNG='C17..C3042';
t_IMU_yaw_1=csvread('yaw_1_IMU_7Sep.csv',16,2,RNG);
t_IMU_yaw_1=t_IMU_yaw_1-16968.787; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3042';
gyro=csvread('yaw_1_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
```

```
    gyro_yaw_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Plot
%Rename variable
yaw_1_IMU_deg=RPY_deg;

%------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2858';
yaw_2_IMU=csvread('yaw_2_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(yaw_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_yaw_2=RPY_deg(:,514);

%Load Data - Time
RNG='C17..C2858';
t_IMU_yaw_2=csvread('yaw_2_IMU_7Sep.csv',16,2,RNG);
t_IMU_yaw_2=t_IMU_yaw_2-17332.203; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2858';
gyro=csvread('yaw_2_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_yaw_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
yaw_2_IMU_deg=RPY_deg;

%------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3143';
yaw_3_IMU=csvread('yaw_3_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(yaw_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_yaw_3=RPY_deg(:,461);

%Load Data - Time
RNG='C17..C3143';
t_IMU_yaw_3=csvread('yaw_3_IMU_7Sep.csv',16,2,RNG);
t_IMU_yaw_3=t_IMU_yaw_3-17784.159; %Subtract timestamp
```

```
%Load Data - Gyro
RNG='G17..I3143';
gyro=csvread('yaw_3_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_yaw_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
yaw_3_IMU_deg=RPY_deg;

%------------------------------ QUARC -------------------------------------

%Rotation Matrix
rotm=[1 0 0; 0 0 1; 0 -1 0];

%Function to convert rotation matrix to quaternion
quat=rotm2quat(rotm);

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('roll_1_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_roll_1=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_roll_1_7Sep');
t_quarc_roll_1=mytime;

%Rename variable
roll_1_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('roll_2_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];
```

```
%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_roll_2=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_roll_2_7Sep');
t_quarc_roll_2=mytime;

%Rename variable
roll_2_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('roll_3_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_roll_3=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_roll_3_7Sep');
t_quarc_roll_3=mytime;

%Rename variable
roll_3_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('pitch_1_quarc_7Sep.mat')

%Change variable name of loaded data
```

```
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_pitch_1=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_pitch_1_7Sep');
t_quarc_pitch_1=mytime;

%Rename variable
pitch_1_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('pitch_2_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_pitch_2=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_pitch_2_7Sep');
t_quarc_pitch_2=mytime;

%Rename variable
pitch_2_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------
```

```
%Load Data - Quaternion
load('pitch_3_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_pitch_3=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_pitch_3_7Sep');
t_quarc_pitch_3=mytime;

%Rename variable
pitch_3_quarc_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('yaw_1_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_yaw_1=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_yaw_1_7Sep');
t_quarc_yaw_1=mytime;

%Rename variable
```

```
yaw_1_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('yaw_2_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_yaw_2=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_yaw_2_7Sep');
t_quarc_yaw_2=mytime;

%Rename variable
yaw_2_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('yaw_3_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_yaw_3=RPY_deg(:,1);

%Load Data - Time
```

```
load('t_quarc_yaw_3_7Sep');
t_quarc_yaw_3=mytime;

%Rename variable
yaw_3_quarc_deg=RPY_deg;

%-------------------------- Separate Comparison --------------------------

%Compare Roll
figure
hold on
plot(t_IMU_roll_1(520:1962)-t_IMU_roll_1(520),roll_1_IMU_deg(1,520:1962))
plot(t_quarc_roll_1(1:1563),roll_1_quarc_deg(1,1:1563)-...
    (deg1_roll_1(1)-deg_roll_1(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Roll 1')
ylim([-5 45])
xlim([0 15.63])

figure
hold on
plot(t_IMU_roll_2(487:2426)-t_IMU_roll_2(487),roll_2_IMU_deg(1,487:2426))
plot(t_quarc_roll_2(1:2071),roll_2_quarc_deg(1,1:2071)-...
    (deg1_roll_2(1)-deg_roll_2(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Roll 2')
ylim([-5 45])
xlim([0 20.71])

figure
hold on
plot(t_IMU_roll_3(495:2681)-t_IMU_roll_3(495),roll_3_IMU_deg(1,495:2681))
plot(t_quarc_roll_3(1:2281),roll_3_quarc_deg(1,1:2281)-...
    (deg1_roll_3(1)-deg_roll_3(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Roll 3')
ylim([-5 45])
xlim([0 22.81])

%Compare Pitch
figure
hold on
plot(t_IMU_pitch_1(477:2853)-t_IMU_pitch_1(477),...
    pitch_1_IMU_deg(2,477:2853))
plot(t_quarc_pitch_1(1:2511),pitch_1_quarc_deg(2,1:2511)-...
    (deg1_pitch_1(2)-deg_pitch_1(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Pitch 1')
ylim([-5 45])
xlim([0 25.11])

figure
hold on
plot(t_IMU_pitch_2(495:2754)-t_IMU_pitch_2(495),...
    pitch_2_IMU_deg(2,495:2754))
plot(t_quarc_pitch_2(1:2340),pitch_2_quarc_deg(2,1:2340)-...
    (deg1_pitch_2(2)-deg_pitch_2(2)),'r')
hold off
```

```
legend('MARG','QUARC','Location','NorthWest')
title('Pitch 2')
ylim([-5 45])
xlim([0 23.40])

figure
hold on
plot(t_IMU_pitch_3(488:3062)-t_IMU_pitch_3(488),...
    pitch_3_IMU_deg(2,488:3062))
plot(t_quarc_pitch_3(1:2720),pitch_3_quarc_deg(2,1:2720)-...
    (deg1_pitch_3(2)-deg_pitch_3(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Pitch 3')
ylim([-5 45])
xlim([0 27.20])

%Compare Yaw
figure
hold on
plot(t_IMU_yaw_1(530:3026)-t_IMU_yaw_1(530),yaw_1_IMU_deg(3,530:3026))
plot(t_quarc_yaw_1(1:2651),yaw_1_quarc_deg(3,1:2651)-...
    (deg1_yaw_1(3)-deg_yaw_1(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Yaw 1')
ylim([-112 -62])
xlim([0 26.51])

figure
hold on
plot(t_IMU_yaw_2(514:2842)-t_IMU_yaw_2(514),yaw_2_IMU_deg(3,514:2842))
plot(t_quarc_yaw_2(1:2465),yaw_2_quarc_deg(3,1:2465)-...
    (deg1_yaw_2(3)-deg_yaw_2(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Yaw 2')
ylim([-112 -62])
xlim([0 24.65])

figure
hold on
plot(t_IMU_yaw_3(461:3127)-t_IMU_yaw_3(461),yaw_3_IMU_deg(3,461:3127))
plot(t_quarc_yaw_3(1:2841),yaw_3_quarc_deg(3,1:2841)-...
    (deg1_yaw_3(3)-deg_yaw_3(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Yaw 3')
ylim([-112 -62])
xlim([0 28.41])

%-------------------------------------------------------------------------

%Compare Roll 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_roll_1(520:1962)-t_IMU_roll_1(520),roll_1_IMU_deg(1,520:1962))
plot(t_quarc_roll_1(1:1563),roll_1_quarc_deg(1,1:1563)-...
    (deg1_roll_1(1)-deg_roll_1(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
```

```
ylim([-5 45])
xlim([0 15.63])

subplot(3,1,2), title('Pitch')
hold on
plot(t_IMU_roll_1(520:1962)-t_IMU_roll_1(520),roll_1_IMU_deg(2,520:1962))
plot(t_quarc_roll_1(1:1563),roll_1_quarc_deg(2,1:1563)-...
    (deg1_roll_1(2)-deg_roll_1(2)),'r')
hold off
ylim([-25 25])
xlim([0 15.63])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_roll_1(520:1962)-t_IMU_roll_1(520),roll_1_IMU_deg(3,520:1962))
plot(t_quarc_roll_1(1:1563),roll_1_quarc_deg(3,1:1563)-...
    (deg1_roll_1(3)-deg_roll_1(3)),'r')
hold off
ylim([-132 -82])
xlim([0 15.63])

%Compare Roll 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_roll_2(487:2426)-t_IMU_roll_2(487),roll_2_IMU_deg(1,487:2426))
plot(t_quarc_roll_2(1:2071),roll_2_quarc_deg(1,1:2071)-...
    (deg1_roll_2(1)-deg_roll_2(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 20.71])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_roll_2(487:2426)-t_IMU_roll_2(487),roll_2_IMU_deg(2,487:2426))
plot(t_quarc_roll_2(1:2071),roll_2_quarc_deg(2,1:2071)-...
    (deg1_roll_2(2)-deg_roll_2(2)),'r')
hold off
ylim([-25 25])
xlim([0 20.71])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_roll_2(487:2426)-t_IMU_roll_2(487),roll_2_IMU_deg(3,487:2426))
plot(t_quarc_roll_2(1:2071),roll_2_quarc_deg(3,1:2071)-...
    (deg1_roll_2(3)-deg_roll_2(3)),'r')
hold off
ylim([-132 -82])
xlim([0 20.71])

%Compare Roll 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_roll_3(495:2681)-t_IMU_roll_3(495),roll_3_IMU_deg(1,495:2681))
plot(t_quarc_roll_3(1:2281),roll_3_quarc_deg(1,1:2281)-...
    (deg1_roll_3(1)-deg_roll_3(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 22.81])
```

```
subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_roll_3(495:2681)-t_IMU_roll_3(495),roll_3_IMU_deg(2,495:2681))
plot(t_quarc_roll_3(1:2281),roll_3_quarc_deg(2,1:2281)-...
    (deg1_roll_3(2)-deg_roll_3(2)),'r')
hold off
ylim([-25 25])
xlim([0 22.81])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_roll_3(495:2681)-t_IMU_roll_3(495),roll_3_IMU_deg(3,495:2681))
plot(t_quarc_roll_3(1:2281),roll_3_quarc_deg(3,1:2281)-...
    (deg1_roll_3(3)-deg_roll_3(3)),'r')
hold off
ylim([-132 -82])
xlim([0 22.81])
xlabel('Time (Seconds)')

saveas(gcf,'Roll_Angles.jpeg')

%Compare Pitch 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_pitch_1(477:2853)-t_IMU_pitch_1(477),pitch_1_IMU_deg(1,477:2853))
plot(t_quarc_pitch_1(1:2511),pitch_1_quarc_deg(1,1:2511)-...
    (deg1_pitch_1(1)-deg_pitch_1(1)),'r')
hold off
ylim([-25 25])
xlim([0 25.11])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_pitch_1(477:2853)-t_IMU_pitch_1(477),...
    pitch_1_IMU_deg(2,477:2853))
plot(t_quarc_pitch_1(1:2511),pitch_1_quarc_deg(2,1:2511)-...
    (deg1_pitch_1(2)-deg_pitch_1(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 25.11])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_pitch_1(477:2853)-t_IMU_pitch_1(477),...
    pitch_1_IMU_deg(3,477:2853))
plot(t_quarc_pitch_1(1:2511),pitch_1_quarc_deg(3,1:2511)-...
    (deg1_pitch_1(3)-deg_pitch_1(3)),'r')
hold off
ylim([-132 -82])
xlim([0 25.11])
xlabel('Time (Seconds)')

saveas(gcf,'Pitch_Angles_1.jpeg')

%Compare Pitch 2
figure
subplot(3,1,1), title('Roll')
```

```
hold on
plot(t_IMU_pitch_2(495:2754)-t_IMU_pitch_2(495),...
    pitch_2_IMU_deg(1,495:2754))
plot(t_quarc_pitch_2(1:2340),pitch_2_quarc_deg(1,1:2340)-...
    (deg1_pitch_2(1)-deg_pitch_2(1)),'r')
hold off
ylim([-25 25])
xlim([0 23.40])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_pitch_2(495:2754)-t_IMU_pitch_2(495),...
    pitch_2_IMU_deg(2,495:2754))
plot(t_quarc_pitch_2(1:2340),pitch_2_quarc_deg(2,1:2340)-...
    (deg1_pitch_2(2)-deg_pitch_2(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 23.40])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_pitch_2(495:2754)-t_IMU_pitch_2(495),...
    pitch_2_IMU_deg(3,495:2754))
plot(t_quarc_pitch_2(1:2340),pitch_2_quarc_deg(3,1:2340)-...
    (deg1_pitch_2(3)-deg_pitch_2(3)),'r')
hold off
ylim([-132 -82])
xlim([0 23.40])
xlabel('Time (Seconds)')

saveas(gcf,'Pitch_Angles_2.jpeg')

%Compare Pitch 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_pitch_3(488:3062)-t_IMU_pitch_3(488),...
    pitch_3_IMU_deg(1,488:3062))
plot(t_quarc_pitch_3(1:2720),pitch_3_quarc_deg(1,1:2720)-...
    (deg1_pitch_3(1)-deg_pitch_3(1)),'r')
hold off
ylim([-25 25])
xlim([0 27.20])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_pitch_3(488:3062)-t_IMU_pitch_3(488),...
    pitch_3_IMU_deg(2,488:3062))
plot(t_quarc_pitch_3(1:2720),pitch_3_quarc_deg(2,1:2720)-...
    (deg1_pitch_3(2)-deg_pitch_3(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 27.20])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_pitch_3(488:3062)-t_IMU_pitch_3(488),...
    pitch_3_IMU_deg(3,488:3062))
plot(t_quarc_pitch_3(1:2720),pitch_3_quarc_deg(3,1:2720)-...
```

```
    (deg1_pitch_3(3)-deg_pitch_3(3)),'r')
hold off
ylim([-132 -82])
xlim([0 27.20])

%Compare Yaw 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_yaw_1(530:3026)-t_IMU_yaw_1(530),yaw_1_IMU_deg(1,530:3026))
plot(t_quarc_yaw_1(1:2651),yaw_1_quarc_deg(1,1:2651)-...
    (deg1_yaw_1(1)-deg_yaw_1(1)),'r')
hold off
ylim([-25 25])
xlim([0 26.51])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_yaw_1(530:3026)-t_IMU_yaw_1(530),yaw_1_IMU_deg(2,530:3026))
plot(t_quarc_yaw_1(1:2651),yaw_1_quarc_deg(2,1:2651)-...
    (deg1_yaw_1(2)-deg_yaw_1(2)),'r')
hold off
ylim([-25 25])
xlim([0 26.51])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_yaw_1(530:3026)-t_IMU_yaw_1(530),yaw_1_IMU_deg(3,530:3026))
plot(t_quarc_yaw_1(1:2651),yaw_1_quarc_deg(3,1:2651)-...
    (deg1_yaw_1(3)-deg_yaw_1(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-112 -62])
xlim([0 26.51])
xlabel('Time (Seconds)')

saveas(gcf,'Yaw_Angles.jpeg')

%Compare Yaw 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_yaw_2(514:2842)-t_IMU_yaw_2(514),yaw_2_IMU_deg(1,514:2842))
plot(t_quarc_yaw_2(1:2465),yaw_2_quarc_deg(1,1:2465)-...
    (deg1_yaw_2(1)-deg_yaw_2(1)),'r')
hold off
ylim([-25 25])
xlim([0 24.65])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_yaw_2(514:2842)-t_IMU_yaw_2(514),yaw_2_IMU_deg(2,514:2842))
plot(t_quarc_yaw_2(1:2465),yaw_2_quarc_deg(2,1:2465)-...
    (deg1_yaw_2(2)-deg_yaw_2(2)),'r')
hold off
ylim([-25 25])
xlim([0 24.65])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_yaw_2(514:2842)-t_IMU_yaw_2(514),yaw_2_IMU_deg(3,514:2842))
```

```
plot(t_quarc_yaw_2(1:2465),yaw_2_quarc_deg(3,1:2465)-...
    (deg1_yaw_2(3)-deg_yaw_2(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-112 -62])
xlim([0 24.65])

%Compare Yaw 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_yaw_3(461:3127)-t_IMU_yaw_3(461),yaw_3_IMU_deg(1,461:3127))
plot(t_quarc_yaw_3(1:2841),yaw_3_quarc_deg(1,1:2841)-...
    (deg1_yaw_3(1)-deg_yaw_3(1)),'r')
hold off
ylim([-25 25])
xlim([0 28.41])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_yaw_3(461:3127)-t_IMU_yaw_3(461),yaw_3_IMU_deg(2,461:3127))
plot(t_quarc_yaw_3(1:2841),yaw_3_quarc_deg(2,1:2841)-...
    (deg1_yaw_3(2)-deg_yaw_3(2)),'r')
hold off
ylim([-25 25])
xlim([0 28.41])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_yaw_3(461:3127)-t_IMU_yaw_3(461),yaw_3_IMU_deg(3,461:3127))
plot(t_quarc_yaw_3(1:2841),yaw_3_quarc_deg(3,1:2841)-...
    (deg1_yaw_3(3)-deg_yaw_3(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-112 -62])
xlim([0 28.41])

%--------------------------Difference-------------------------------------

%Roll 1
ti=t_quarc_roll_1(1:1563);
t=t_IMU_roll_1(520:1962)-t_IMU_roll_1(520);
x=roll_1_IMU_deg(1,520:1962);
roll_i=interp1(t,x,ti);
x=roll_1_IMU_deg(2,520:1962);
pitch_i=interp1(t,x,ti);
x=roll_1_IMU_deg(3,520:1962);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(roll_1_quarc_deg(1,1:1563)-...
    (deg1_roll_1(1)-deg_roll_1(1)))))
ylim([0 4]), title('Roll')
xlim([0 15.63])
subplot(3,1,2),plot(ti,abs(pitch_i'-(roll_1_quarc_deg(2,1:1563)-...
    (deg1_roll_1(2)-deg_roll_1(2)))))
ylim([0 4]), title('Pitch')
xlim([0 15.63])
subplot(3,1,3),plot(ti,abs(yaw_i'-(roll_1_quarc_deg(3,1:1563)-...
    (deg1_roll_1(3)-deg_roll_1(3)))))
ylim([0 4]), title('Yaw')
xlim([0 15.63])
```

```
roll_diff=abs(roll_i'-(roll_1_quarc_deg(1,1:1563)-...
    (deg1_roll_1(1)-deg_roll_1(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:1562).^2))

pitch_diff=abs(pitch_i'-(roll_1_quarc_deg(2,1:1563)-...
    (deg1_roll_1(2)-deg_roll_1(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:1562).^2))

yaw_diff=abs(yaw_i'-(roll_1_quarc_deg(3,1:1563)-...
    (deg1_roll_1(3)-deg_roll_1(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:1562).^2))

%Roll 2
ti=t_quarc_roll_2(1:2071);
t=t_IMU_roll_2(487:2426)-t_IMU_roll_2(487);
x=roll_2_IMU_deg(1,487:2426);
roll_i=interp1(t,x,ti);
x=roll_2_IMU_deg(2,487:2426);
pitch_i=interp1(t,x,ti);
x=roll_2_IMU_deg(3,487:2426);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(roll_2_quarc_deg(1,1:2071)-...
    (deg1_roll_2(1)-deg_roll_2(1)))))
ylim([0 4]), title('Roll')
xlim([0 20.71])
subplot(3,1,2),plot(ti,abs(pitch_i'-(roll_2_quarc_deg(2,1:2071)-...
    (deg1_roll_2(2)-deg_roll_2(2)))))
ylim([0 4]), title('Pitch')
xlim([0 20.71])
subplot(3,1,3),plot(ti,abs(yaw_i'-(roll_2_quarc_deg(3,1:2071)-...
    (deg1_roll_2(3)-deg_roll_2(3)))))
ylim([0 4]), title('Yaw')
xlim([0 20.71])

roll_diff=abs(roll_i'-(roll_2_quarc_deg(1,1:2071)-...
    (deg1_roll_2(1)-deg_roll_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2070).^2))

pitch_diff=abs(pitch_i'-(roll_2_quarc_deg(2,1:2071)-...
    (deg1_roll_2(2)-deg_roll_2(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2070).^2))

yaw_diff=abs(yaw_i'-(roll_2_quarc_deg(3,1:2071)-...
    (deg1_roll_2(3)-deg_roll_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2070).^2))

%Roll 3
ti=t_quarc_roll_3(1:2281);
t=t_IMU_roll_3(495:2681)-t_IMU_roll_3(495);
x=roll_3_IMU_deg(1,495:2681);
roll_i=interp1(t,x,ti);
x=roll_3_IMU_deg(2,495:2681);
pitch_i=interp1(t,x,ti);
```

```
x=roll_3_IMU_deg(3,495:2681);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(roll_3_quarc_deg(1,1:2281)-...
    (deg1_roll_3(1)-deg_roll_3(1)))))
ylim([0 4]), title('Roll')
xlim([0 22.81])
subplot(3,1,2),plot(ti,abs(pitch_i'-(roll_3_quarc_deg(2,1:2281)-...
    (deg1_roll_3(2)-deg_roll_3(2)))))
ylim([0 4]), title('Pitch')
xlim([0 22.81])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(roll_3_quarc_deg(3,1:2281)-...
    (deg1_roll_3(3)-deg_roll_3(3)))))
ylim([0 4]), title('Yaw')
xlim([0 22.81])
xlabel('Time (Seconds)')

saveas(gcf,'Roll_Diff.jpeg')

roll_diff=abs(roll_i'-(roll_3_quarc_deg(1,1:2281)-...
    (deg1_roll_3(1)-deg_roll_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2280).^2))

pitch_diff=abs(pitch_i'-(roll_3_quarc_deg(2,1:2281)-...
    (deg1_roll_3(2)-deg_roll_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2280).^2))

yaw_diff=abs(yaw_i'-(roll_3_quarc_deg(3,1:2281)-...
    (deg1_roll_3(3)-deg_roll_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2280).^2))

%Pitch 1
ti=t_quarc_pitch_1(1:2511);
t=t_IMU_pitch_1(477:2853)-t_IMU_pitch_1(477);
x=pitch_1_IMU_deg(1,477:2853);
roll_i=interp1(t,x,ti);
x=pitch_1_IMU_deg(2,477:2853);
pitch_i=interp1(t,x,ti);
x=pitch_1_IMU_deg(3,477:2853);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(pitch_1_quarc_deg(1,1:2511)-...
    (deg1_pitch_1(1)-deg_pitch_1(1)))))
ylim([0 4]), title('Roll')
xlim([0 25.11])
subplot(3,1,2),plot(ti,abs(pitch_i'-(pitch_1_quarc_deg(2,1:2511)-...
    (deg1_pitch_1(2)-deg_pitch_1(2)))))
ylim([0 4]), title('Pitch')
xlim([0 25.11])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(pitch_1_quarc_deg(3,1:2511)-...
    (deg1_pitch_1(3)-deg_pitch_1(3)))))
ylim([0 4]), title('Yaw')
xlim([0 25.11])
xlabel('Time (Seconds)')
```

```
saveas(gcf,'Pitch_Diff_1.jpeg')

roll_diff=abs(roll_i'-(pitch_1_quarc_deg(1,1:2511)-...
    (deg1_pitch_1(1)-deg_pitch_1(1))));
roll_diff=[roll_diff(1:1266) roll_diff(1272:2511)];
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2505).^2))

pitch_diff=abs(pitch_i'-(pitch_1_quarc_deg(2,1:2511)-...
    (deg1_pitch_1(2)-deg_pitch_1(2))));
pitch_diff=[pitch_diff(1:1266) pitch_diff(1272:2511)];
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2505).^2))

yaw_diff=abs(yaw_i'-(pitch_1_quarc_deg(3,1:2511)-...
    (deg1_pitch_1(3)-deg_pitch_1(3))));
yaw_diff=[yaw_diff(1:1266) yaw_diff(1272:2511)];
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2505).^2))

%Pitch 2
ti=t_quarc_pitch_2(1:2340);
t=t_IMU_pitch_2(495:2754)-t_IMU_pitch_2(495);
x=pitch_2_IMU_deg(1,495:2754);
roll_i=interp1(t,x,ti);
x=pitch_2_IMU_deg(2,495:2754);
pitch_i=interp1(t,x,ti);
x=pitch_2_IMU_deg(3,495:2754);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(pitch_2_quarc_deg(1,1:2340)-...
    (deg1_pitch_2(1)-deg_pitch_2(1)))))
ylim([0 4]), title('Roll')
xlim([0 23.40])
subplot(3,1,2),plot(ti,abs(pitch_i'-(pitch_2_quarc_deg(2,1:2340)-...
    (deg1_pitch_2(2)-deg_pitch_2(2)))))
ylim([0 4]), title('Pitch')
xlim([0 23.40])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(pitch_2_quarc_deg(3,1:2340)-...
    (deg1_pitch_2(3)-deg_pitch_2(3)))))
ylim([0 4]), title('Yaw')
xlim([0 23.40])
xlabel('Time (Seconds)')

saveas(gcf,'Pitch_Diff_2.jpeg')

roll_diff=abs(roll_i'-(pitch_2_quarc_deg(1,1:2340)-...
    (deg1_pitch_2(1)-deg_pitch_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2339).^2))

pitch_diff=abs(pitch_i'-(pitch_2_quarc_deg(2,1:2340)-...
    (deg1_pitch_2(2)-deg_pitch_2(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2339).^2))

yaw_diff=abs(yaw_i'-(pitch_2_quarc_deg(3,1:2340)-...
    (deg1_pitch_2(3)-deg_pitch_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2339).^2))
```

```
%Pitch 3
ti=t_quarc_pitch_3(1:2720);
t=t_IMU_pitch_3(488:3062)-t_IMU_pitch_3(488);
x=pitch_3_IMU_deg(1,488:3062);
roll_i=interp1(t,x,ti);
x=pitch_3_IMU_deg(2,488:3062);
pitch_i=interp1(t,x,ti);
x=pitch_3_IMU_deg(3,488:3062);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(pitch_3_quarc_deg(1,1:2720)-...
    (deg1_pitch_3(1)-deg_pitch_3(1)))))
ylim([0 4]), title('Roll')
xlim([0 27.20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(pitch_3_quarc_deg(2,1:2720)-...
    (deg1_pitch_3(2)-deg_pitch_3(2)))))
ylim([0 4]), title('Pitch')
xlim([0 27.20])
subplot(3,1,3),plot(ti,abs(yaw_i'-(pitch_3_quarc_deg(3,1:2720)-...
    (deg1_pitch_3(3)-deg_pitch_3(3)))))
ylim([0 4]), title('Yaw')
xlim([0 27.20])

roll_diff=abs(roll_i'-(pitch_3_quarc_deg(1,1:2720)-...
    (deg1_pitch_3(1)-deg_pitch_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2719).^2))

pitch_diff=abs(pitch_i'-(pitch_3_quarc_deg(2,1:2720)-...
    (deg1_pitch_3(2)-deg_pitch_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2719).^2))

yaw_diff=abs(yaw_i'-(pitch_3_quarc_deg(3,1:2720)-...
    (deg1_pitch_3(3)-deg_pitch_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2719).^2))

%Yaw 1
ti=t_quarc_yaw_1(1:2651);
t=t_IMU_yaw_1(530:3026)-t_IMU_yaw_1(530);
x=yaw_1_IMU_deg(1,530:3026);
roll_i=interp1(t,x,ti);
x=yaw_1_IMU_deg(2,530:3026);
pitch_i=interp1(t,x,ti);
x=yaw_1_IMU_deg(3,530:3026);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(yaw_1_quarc_deg(1,1:2651)-...
    (deg1_yaw_1(1)-deg_yaw_1(1)))))
ylim([0 4]), title('Roll')
xlim([0 26.51])
subplot(3,1,2),plot(ti,abs(pitch_i'-(yaw_1_quarc_deg(2,1:2651)-...
    (deg1_yaw_1(2)-deg_yaw_1(2)))))
ylim([0 4]), title('Pitch')
xlim([0 26.51])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(yaw_1_quarc_deg(3,1:2651)-...
    (deg1_yaw_1(3)-deg_yaw_1(3)))))
```

```
ylim([0 4]), title('Yaw')
xlim([0 26.51])
xlabel('Time (Seconds)')

saveas(gcf,'Yaw_Diff.jpeg')

roll_diff=abs(roll_i'-(yaw_1_quarc_deg(1,1:2651)-...
    (deg1_yaw_1(1)-deg_yaw_1(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2650).^2))

pitch_diff=abs(pitch_i'-(yaw_1_quarc_deg(2,1:2651)-...
    (deg1_yaw_1(2)-deg_yaw_1(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2650).^2))

yaw_diff=abs(yaw_i'-(yaw_1_quarc_deg(3,1:2651)-...
    (deg1_yaw_1(3)-deg_yaw_1(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2650).^2))

%Yaw 2
ti=t_quarc_yaw_2(1:2465);
t=t_IMU_yaw_2(514:2842)-t_IMU_yaw_2(514);
x=yaw_2_IMU_deg(1,514:2842);
roll_i=interp1(t,x,ti);
x=yaw_2_IMU_deg(2,514:2842);
pitch_i=interp1(t,x,ti);
x=yaw_2_IMU_deg(3,514:2842);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(yaw_2_quarc_deg(1,1:2465)-...
    (deg1_yaw_2(1)-deg_yaw_2(1)))))
ylim([0 4]), title('Roll')
xlim([0 24.65])
subplot(3,1,2),plot(ti,abs(pitch_i'-(yaw_2_quarc_deg(2,1:2465)-...
    (deg1_yaw_2(2)-deg_yaw_2(2)))))
ylim([0 4]), title('Pitch')
xlim([0 24.65])
subplot(3,1,3),plot(ti,abs(yaw_i'-(yaw_2_quarc_deg(3,1:2465)-...
    (deg1_yaw_2(3)-deg_yaw_2(3)))))
ylim([0 4]), title('Yaw')
xlim([0 24.65])

roll_diff=abs(roll_i'-(yaw_2_quarc_deg(1,1:2465)-...
    (deg1_yaw_2(1)-deg_yaw_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2464).^2))

pitch_diff=abs(pitch_i'-(yaw_2_quarc_deg(2,1:2465)-...
    (deg1_yaw_2(2)-deg_yaw_2(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2464).^2))

yaw_diff=abs(yaw_i'-(yaw_2_quarc_deg(3,1:2465)-...
    (deg1_yaw_2(3)-deg_yaw_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2464).^2))

%Yaw 3
ti=t_quarc_yaw_3(1:2841);
```

```
t=t_IMU_yaw_3(461:3127)-t_IMU_yaw_3(461);
x=yaw_3_IMU_deg(1,461:3127);
roll_i=interp1(t,x,ti);
x=yaw_3_IMU_deg(2,461:3127);
pitch_i=interp1(t,x,ti);
x=yaw_3_IMU_deg(3,461:3127);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(yaw_3_quarc_deg(1,1:2841)-...
    (deg1_yaw_3(1)-deg_yaw_3(1)))))
ylim([0 4]), title('Roll')
xlim([0 28.41])
subplot(3,1,2),plot(ti,abs(pitch_i'-(yaw_3_quarc_deg(2,1:2841)-...
    (deg1_yaw_3(2)-deg_yaw_3(2)))))
ylim([0 4]), title('Pitch')
xlim([0 28.41])
subplot(3,1,3),plot(ti,abs(yaw_i'-(yaw_3_quarc_deg(3,1:2841)-...
    (deg1_yaw_3(3)-deg_yaw_3(3)))))
ylim([0 4]), title('Yaw')
xlim([0 28.41])

roll_diff=abs(roll_i'-(yaw_3_quarc_deg(1,1:2841)-...
    (deg1_yaw_3(1)-deg_yaw_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2840).^2))

pitch_diff=abs(pitch_i'-(yaw_3_quarc_deg(2,1:2841)-...
    (deg1_yaw_3(2)-deg_yaw_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2840).^2))

yaw_diff=abs(yaw_i'-(yaw_3_quarc_deg(3,1:2841)-...
    (deg1_yaw_3(3)-deg_yaw_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2840).^2))
```

B. FUNCTION

The function used in conjuction with the code for slow motion in a single axis is contained in this section.

```
%Rotation Matrix to Quaternion
%By:  Heather Pelachick

function [quat]=rotm2quat(r)

q0=(1/2)*sqrt(r(1,1)+r(2,2)+r(3,3)+1);
q1=-(r(3,2)-r(2,3))/(4*q0);
q2=-(r(1,3)-r(3,1))/(4*q0);
q3=-(r(2,1)-r(1,2))/(4*q0);

quat=[q0 q1 q2 q3];
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. MATLAB CODE FOR FAST MOTION IN A SINGLE AXIS

A. CODE

The code used during the fast motion in a single axis is contained in this section.

```
%Fast Motion in a Single Axis
%By:  Heather Pelachick

%This code utilizies sensor measurements from the 3DM-GX4-25 and QUARC
%to determine the dynamic accuracy, angular rate, peak instantaneous error,
%angles of rotation plot, and difference plot for fast motions in a single
%axis.

clear all;
close all;

%----------------------------- IMU ---------------------------------------

%Load Data - Quaternion
RNG='M17..P3038';
roll_1_IMU=csvread('fast_roll_1_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);


%Initial degree values
deg_roll_1=RPY_deg(:,492);

%Load Data - Time
RNG='C17..C3038';
t_IMU_roll_1=csvread('fast_roll_1_IMU_7Sep.csv',16,2,RNG);
t_IMU_roll_1=t_IMU_roll_1-18225.035; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3038';
gyro=csvread('fast_roll_1_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);
    gyro_roll_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end
```

```
%Rename variable
roll_1_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2305';
roll_2_IMU=csvread('fast_roll_2_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_roll_2=RPY_deg(:,465);

%Load Data - Time
RNG='C17..C2305';
t_IMU_roll_2=csvread('fast_roll_2_IMU_7Sep.csv',16,2,RNG);
t_IMU_roll_2=t_IMU_roll_2-18608.541; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2305';
gyro=csvread('fast_roll_2_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_roll_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
roll_2_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2718';
roll_3_IMU=csvread('fast_roll_3_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(roll_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_roll_3=RPY_deg(:,482);

%Load Data - Time
RNG='C17..C2718';
t_IMU_roll_3=csvread('fast_roll_3_IMU_7Sep.csv',16,2,RNG);
t_IMU_roll_3=t_IMU_roll_3-18853.897; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2718';
gyro=csvread('fast_roll_3_IMU_7Sep.csv',16,6,RNG);
```

```
for ii=1:length(gyro)
    gyro_roll_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
roll_3_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M18..P2553';
pitch_1_IMU=csvread('fast_pitch_1_IMU_7Sep.csv',17,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(pitch_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_pitch_1=RPY_deg(:,466);

%Load Data - Time
RNG='C18..C2553';
t_IMU_pitch_1=csvread('fast_pitch_1_IMU_7Sep.csv',17,2,RNG);
t_IMU_pitch_1=t_IMU_pitch_1-19107.033; %Subtract timestamp

%Load Data - Gyro
RNG='G18..I2553';
gyro=csvread('fast_pitch_1_IMU_7Sep.csv',17,6,RNG);
for ii=1:length(gyro)
    gyro_pitch_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
pitch_1_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2696';
pitch_2_IMU=csvread('fast_pitch_2_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(pitch_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_pitch_2=RPY_deg(:,488);

%Load Data - Time
RNG='C17..C2696';
t_IMU_pitch_2=csvread('fast_pitch_2_IMU_7Sep.csv',16,2,RNG);
t_IMU_pitch_2=t_IMU_pitch_2-19390.599; %Subtract timestamp
```

```
%Load Data - Gyro
RNG='G17..I2696';
gyro=csvread('fast_pitch_2_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_pitch_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
pitch_2_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2895';
pitch_3_IMU=csvread('fast_pitch_3_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(pitch_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_pitch_3=RPY_deg(:,507);

%Load Data - Time
RNG='C17..C2895';
t_IMU_pitch_3=csvread('fast_pitch_3_IMU_7Sep.csv',16,2,RNG);
t_IMU_pitch_3=t_IMU_pitch_3-19668.965; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2895';
gyro=csvread('fast_pitch_3_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_pitch_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
pitch_3_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2725';
yaw_1_IMU=csvread('fast_yaw_1_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(yaw_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_yaw_1=RPY_deg(:,528);
```

```
%Load Data - Time
RNG='C17..C2725';
t_IMU_yaw_1=csvread('fast_yaw_1_IMU_7Sep.csv',16,2,RNG);
t_IMU_yaw_1=t_IMU_yaw_1-19990.491; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2725';
gyro=csvread('fast_yaw_1_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_yaw_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
yaw_1_IMU_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2831';
yaw_2_IMU=csvread('fast_yaw_2_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(yaw_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_yaw_2=RPY_deg(:,512);

%Load Data - Time
RNG='C17..C2831';
t_IMU_yaw_2=csvread('fast_yaw_2_IMU_7Sep.csv',16,2,RNG);
t_IMU_yaw_2=t_IMU_yaw_2-20373.323; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2831';
gyro=csvread('fast_yaw_2_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_yaw_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
yaw_2_IMU_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2683';
yaw_3_IMU=csvread('fast_yaw_3_IMU_7Sep.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(yaw_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);
```

```
%Initial degree values
deg_yaw_3=RPY_deg(:,461);

%Load Data - Time
RNG='C17..C2683';
t_IMU_yaw_3=csvread('fast_yaw_3_IMU_7Sep.csv',16,2,RNG);
t_IMU_yaw_3=t_IMU_yaw_3-20641.969; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2683';
gyro=csvread('fast_yaw_3_IMU_7Sep.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_yaw_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
yaw_3_IMU_deg=RPY_deg;

%------------------------------ QUARC -------------------------------------

%Rotation Matrix
rotm=[1 0 0; 0 0 1; 0 -1 0];

%Function to convert rotation matrix to quaternion
quat=rotm2quat(rotm);

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_roll_1_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_roll_1=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_roll_1_7Sep');
t_quarc_fast_roll_1=mytime;

%Rename variable
fast_roll_1_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------
```

```
%Load Data - Quaternion
load('fast_roll_2_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_roll_2=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_roll_2_7Sep');
t_quarc_fast_roll_2=mytime;

%Rename variable
fast_roll_2_quarc_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_roll_3_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_roll_3=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_roll_3_7Sep');
t_quarc_fast_roll_3=mytime;

%Rename variable
```

```
fast_roll_3_quarc_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_pitch_1_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];
u=u(2:2280,:);

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_pitch_1=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_pitch_1_7Sep');
t_quarc_fast_pitch_1=mytime;

%Rename variable
fast_pitch_1_quarc_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_pitch_2_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_pitch_2=RPY_deg(:,1);
```

```
%Load Data - Time
load('t_quarc_fast_pitch_2_7Sep');
t_quarc_fast_pitch_2=mytime;

%Rename variable
fast_pitch_2_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_pitch_3_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_pitch_3=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_pitch_3_7Sep');
t_quarc_fast_pitch_3=mytime;

%Rename variable
fast_pitch_3_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_yaw_1_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);
```

```
%Initial degree values
deg1_yaw_1=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_yaw_1_7Sep');
t_quarc_fast_yaw_1=mytime;

%Rename variable
fast_yaw_1_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_yaw_2_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_yaw_2=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_yaw_2_7Sep');
t_quarc_fast_yaw_2=mytime;

%Rename variable
fast_yaw_2_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_yaw_3_quarc_7Sep.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
```

```
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_yaw_3=RPY_deg(:,1);

%Load Data - Time
load('t_quarc_fast_yaw_3_7Sep');
t_quarc_fast_yaw_3=mytime;

%Rename variable
fast_yaw_3_quarc_deg=RPY_deg;

%--------------------------------------------------------------------------

%Compare Roll
figure
hold on
plot(t_IMU_roll_1(492:3022)-t_IMU_roll_1(492),roll_1_IMU_deg(1,492:3022))
plot(t_quarc_fast_roll_1(1:2705),fast_roll_1_quarc_deg(1,1:2705)-...
    (deg1_roll_1(1)-deg_roll_1(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Roll 1')
ylim([-5 45])
xlim([0 27.05])

figure
hold on
plot(t_IMU_roll_2(465:2289)-t_IMU_roll_2(465),roll_2_IMU_deg(1,465:2289))
plot(t_quarc_fast_roll_2(1:1921),fast_roll_2_quarc_deg(1,1:1921)-...
    (deg1_roll_2(1)-deg_roll_2(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Roll 2')
ylim([-5 45])
xlim([0 19.21])

figure
hold on
plot(t_IMU_roll_3(482:2702)-t_IMU_roll_3(482),roll_3_IMU_deg(1,482:2702))
plot(t_quarc_fast_roll_3(1:2315),fast_roll_3_quarc_deg(1,1:2315)-...
    (deg1_roll_3(1)-deg_roll_3(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Roll 3')
ylim([-5 45])
xlim([0 23.15])

%Compare Pitch
figure
hold on
plot(t_IMU_pitch_1(466:2536)-t_IMU_pitch_1(466),...
    pitch_1_IMU_deg(2,466:2536))
plot(t_quarc_fast_pitch_1(1:2163),fast_pitch_1_quarc_deg(2,1:2163)-...
    (deg1_pitch_1(2)-deg_pitch_1(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Pitch 1')
ylim([-5 45])
```

```
xlim([0 21.64])

figure
hold on
plot(t_IMU_pitch_2(488:2680)-t_IMU_pitch_2(488),...
    pitch_2_IMU_deg(2,488:2680))
plot(t_quarc_fast_pitch_2(1:2368),fast_pitch_2_quarc_deg(2,1:2368)-...
    (deg1_pitch_2(2)-deg_pitch_2(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Pitch 2')
ylim([-5 45])
xlim([0 23.68])

figure
hold on
plot(t_IMU_pitch_3(507:2879)-t_IMU_pitch_3(507),...
    pitch_3_IMU_deg(2,507:2879))
plot(t_quarc_fast_pitch_3(1:2526),fast_pitch_3_quarc_deg(2,1:2526)-...
    (deg1_pitch_3(2)-deg_pitch_3(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Pitch 3')
ylim([-5 45])
xlim([0 25.26])

%Compare Yaw
figure
hold on
plot(t_IMU_yaw_1(528:2709)-t_IMU_yaw_1(528),yaw_1_IMU_deg(3,528:2709))
plot(t_quarc_fast_yaw_1(1:2299),fast_yaw_1_quarc_deg(3,1:2299)-...
    (deg1_yaw_1(3)-deg_yaw_1(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Yaw 1')
ylim([-112 -62])
xlim([0 22.99])
xlabel('Time (Seconds)')
ylabel({'Angles of Rotation (Degrees)';' '})

saveas (gcf, 'Yaw.jpeg')

figure
hold on
plot(t_IMU_yaw_2(512:2815)-t_IMU_yaw_2(512),yaw_2_IMU_deg(3,512:2815))
plot(t_quarc_fast_yaw_2(1:2411),fast_yaw_2_quarc_deg(3,1:2411)-...
    (deg1_yaw_2(3)-deg_yaw_2(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Yaw 2')
ylim([-112 -62])
xlim([0 24.11])

figure
hold on
plot(t_IMU_yaw_3(461:2667)-t_IMU_yaw_3(461),yaw_3_IMU_deg(3,461:2667))
plot(t_quarc_fast_yaw_3(1:2390),fast_yaw_3_quarc_deg(3,1:2390)-...
    (deg1_yaw_3(3)-deg_yaw_3(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
title('Yaw 3')
ylim([-112 -62])
```

```
xlim([0 23.90])

%-------------------------------------------------------------------------

%Compare Roll 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_roll_1(492:3022)-t_IMU_roll_1(492),roll_1_IMU_deg(1,492:3022))
plot(t_quarc_fast_roll_1(1:2705),fast_roll_1_quarc_deg(1,1:2705)-...
    (deg1_roll_1(1)-deg_roll_1(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 27.05])

subplot(3,1,2), title('Pitch')
hold on
plot(t_IMU_roll_1(492:3022)-t_IMU_roll_1(492),roll_1_IMU_deg(2,492:3022))
plot(t_quarc_fast_roll_1(1:2705),fast_roll_1_quarc_deg(2,1:2705)-...
    (deg1_roll_1(2)-deg_roll_1(2)),'r')
hold off
ylim([-25 25])
xlim([0 27.05])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_roll_1(492:3022)-t_IMU_roll_1(492),roll_1_IMU_deg(3,492:3022))
plot(t_quarc_fast_roll_1(1:2705),fast_roll_1_quarc_deg(3,1:2705)-...
    (deg1_roll_1(3)-deg_roll_1(3)),'r')
hold off
ylim([-132 -82])
xlim([0 27.05])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Roll_Angles.jpeg')

%Compare Roll 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_roll_2(465:2289)-t_IMU_roll_2(465),roll_2_IMU_deg(1,465:2289))
plot(t_quarc_fast_roll_2(1:1921),fast_roll_2_quarc_deg(1,1:1921)-...
    (deg1_roll_2(1)-deg_roll_2(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 19.21])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_roll_2(465:2289)-t_IMU_roll_2(465),roll_2_IMU_deg(2,465:2289))
plot(t_quarc_fast_roll_2(1:1921),fast_roll_2_quarc_deg(2,1:1921)-...
    (deg1_roll_2(2)-deg_roll_2(2)),'r')
hold off
ylim([-25 25])
xlim([0 19.21])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_roll_2(465:2289)-t_IMU_roll_2(465),roll_2_IMU_deg(3,465:2289))
```

```
plot(t_quarc_fast_roll_2(1:1921),fast_roll_2_quarc_deg(3,1:1921)-...
    (deg1_roll_2(3)-deg_roll_2(3)),'r')
hold off
ylim([-132 -82])
xlim([0 19.21])

%Compare Roll 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_roll_3(482:2702)-t_IMU_roll_3(482),roll_3_IMU_deg(1,482:2702))
plot(t_quarc_fast_roll_3(1:2315),fast_roll_3_quarc_deg(1,1:2315)-...
    (deg1_roll_3(1)-deg_roll_3(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 23.15])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_roll_3(482:2702)-t_IMU_roll_3(482),roll_3_IMU_deg(2,482:2702))
plot(t_quarc_fast_roll_3(1:2315),fast_roll_3_quarc_deg(2,1:2315)-...
    (deg1_roll_3(2)-deg_roll_3(2)),'r')
hold off
ylim([-25 25])
xlim([0 23.15])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_roll_3(482:2702)-t_IMU_roll_3(482),roll_3_IMU_deg(3,482:2702))
plot(t_quarc_fast_roll_3(1:2315),fast_roll_3_quarc_deg(3,1:2315)-...
    (deg1_roll_3(3)-deg_roll_3(3)),'r')
hold off
ylim([-132 -82])
xlim([0 23.15])

%Compare Pitch 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_pitch_1(466:2536)-t_IMU_pitch_1(466),...
    pitch_1_IMU_deg(1,466:2536))
plot(t_quarc_fast_pitch_1(1:2163),fast_pitch_1_quarc_deg(1,1:2163)-...
    (deg1_pitch_1(1)-deg_pitch_1(1)),'r')
hold off
ylim([-25 25])
xlim([0 21.64])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_pitch_1(466:2536)-t_IMU_pitch_1(466),...
    pitch_1_IMU_deg(2,466:2536))
plot(t_quarc_fast_pitch_1(1:2163),fast_pitch_1_quarc_deg(2,1:2163)-...
    (deg1_pitch_1(2)-deg_pitch_1(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 21.64])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_pitch_1(466:2536)-t_IMU_pitch_1(466),...
```

```
    pitch_1_IMU_deg(3,466:2536))
plot(t_quarc_fast_pitch_1(1:2163),fast_pitch_1_quarc_deg(3,1:2163)-...
    (deg1_pitch_1(3)-deg_pitch_1(3)),'r')
hold off
ylim([-132 -82])
xlim([0 21.64])

%Compare Pitch 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_pitch_2(488:2680)-t_IMU_pitch_2(488),...
    pitch_2_IMU_deg(1,488:2680))
plot(t_quarc_fast_pitch_2(1:2368),fast_pitch_2_quarc_deg(1,1:2368)-...
    (deg1_pitch_2(1)-deg_pitch_2(1)),'r')
hold off
ylim([-25 25])
xlim([0 23.68])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_pitch_2(488:2680)-t_IMU_pitch_2(488),...
    pitch_2_IMU_deg(2,488:2680))
plot(t_quarc_fast_pitch_2(1:2368),fast_pitch_2_quarc_deg(2,1:2368)-...
    (deg1_pitch_2(2)-deg_pitch_2(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 23.68])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_pitch_2(488:2680)-t_IMU_pitch_2(488),...
    pitch_2_IMU_deg(3,488:2680))
plot(t_quarc_fast_pitch_2(1:2368),fast_pitch_2_quarc_deg(3,1:2368)-...
    (deg1_pitch_2(3)-deg_pitch_2(3)),'r')
hold off
ylim([-132 -82])
xlim([0 23.68])

%Compare Pitch 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_pitch_3(507:2879)-t_IMU_pitch_3(507),...
    pitch_3_IMU_deg(1,507:2879))
plot(t_quarc_fast_pitch_3(1:2526),fast_pitch_3_quarc_deg(1,1:2526)-...
    (deg1_pitch_3(1)-deg_pitch_3(1)),'r')
hold off
ylim([-25 25])
xlim([0 25.26])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_pitch_3(507:2879)-t_IMU_pitch_3(507),...
    pitch_3_IMU_deg(2,507:2879))
plot(t_quarc_fast_pitch_3(1:2526),fast_pitch_3_quarc_deg(2,1:2526)-...
    (deg1_pitch_3(2)-deg_pitch_3(2)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-5 45])
xlim([0 25.26])
```

```
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_pitch_3(507:2879)-t_IMU_pitch_3(507),...
    pitch_3_IMU_deg(3,507:2879))
plot(t_quarc_fast_pitch_3(1:2526),fast_pitch_3_quarc_deg(3,1:2526)-...
    (deg1_pitch_3(3)-deg_pitch_3(3)),'r')
hold off
ylim([-132 -82])
xlim([0 25.26])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Pitch_Angles.jpeg')

%Compare Yaw 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_yaw_1(528:2709)-t_IMU_yaw_1(528),yaw_1_IMU_deg(1,528:2709))
plot(t_quarc_fast_yaw_1(1:2299),fast_yaw_1_quarc_deg(1,1:2299)-...
    (deg1_yaw_1(1)-deg_yaw_1(1)),'r')
hold off
ylim([-25 25])
xlim([0 22.99])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_yaw_1(528:2709)-t_IMU_yaw_1(528),yaw_1_IMU_deg(2,528:2709))
plot(t_quarc_fast_yaw_1(1:2299),fast_yaw_1_quarc_deg(2,1:2299)-...
    (deg1_yaw_1(2)-deg_yaw_1(2)),'r')
hold off
ylim([-25 25])
xlim([0 22.99])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_yaw_1(528:2709)-t_IMU_yaw_1(528),yaw_1_IMU_deg(3,528:2709))
plot(t_quarc_fast_yaw_1(1:2299),fast_yaw_1_quarc_deg(3,1:2299)-...
    (deg1_yaw_1(3)-deg_yaw_1(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-112 -62])
xlim([0 22.99])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Yaw_Angles.jpeg')

%Compare Yaw 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_yaw_2(512:2815)-t_IMU_yaw_2(512),yaw_2_IMU_deg(1,512:2815))
plot(t_quarc_fast_yaw_2(1:2411),fast_yaw_2_quarc_deg(1,1:2411)-...
    (deg1_yaw_2(1)-deg_yaw_2(1)),'r')
hold off
ylim([-25 25])
xlim([0 24.11])

subplot(3,1,2),title('Pitch')
hold on
```

```
plot(t_IMU_yaw_2(512:2815)-t_IMU_yaw_2(512),yaw_2_IMU_deg(2,512:2815))
plot(t_quarc_fast_yaw_2(1:2411),fast_yaw_2_quarc_deg(2,1:2411)-...
    (deg1_yaw_2(2)-deg_yaw_2(2)),'r')
hold off
ylim([-25 25])
xlim([0 24.11])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_yaw_2(512:2815)-t_IMU_yaw_2(512),yaw_2_IMU_deg(3,512:2815))
plot(t_quarc_fast_yaw_2(1:2411),fast_yaw_2_quarc_deg(3,1:2411)-...
    (deg1_yaw_2(3)-deg_yaw_2(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-112 -62])
xlim([0 24.11])

%Compare Yaw 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_yaw_3(461:2667)-t_IMU_yaw_3(461),yaw_3_IMU_deg(1,461:2667))
plot(t_quarc_fast_yaw_3(1:2390),fast_yaw_3_quarc_deg(1,1:2390)-...
    (deg1_yaw_3(1)-deg_yaw_3(1)),'r')
hold off
ylim([-25 25])
xlim([0 23.90])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_yaw_3(461:2667)-t_IMU_yaw_3(461),yaw_3_IMU_deg(2,461:2667))
plot(t_quarc_fast_yaw_3(1:2390),fast_yaw_3_quarc_deg(2,1:2390)-...
    (deg1_yaw_3(2)-deg_yaw_3(2)),'r')
hold off
ylim([-25 25])
xlim([0 23.90])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_yaw_3(461:2667)-t_IMU_yaw_3(461),yaw_3_IMU_deg(3,461:2667))
plot(t_quarc_fast_yaw_3(1:2390),fast_yaw_3_quarc_deg(3,1:2390)-...
    (deg1_yaw_3(3)-deg_yaw_3(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthWest')
ylim([-112 -62])
xlim([0 23.90])

%--------------------------Difference-------------------------------------

%Roll 1
ti=t_quarc_fast_roll_1(1:2705);
t=t_IMU_roll_1(492:3022)-t_IMU_roll_1(492);
x=roll_1_IMU_deg(1,492:3022);
roll_i=interp1(t,x,ti);
x=roll_1_IMU_deg(2,492:3022);
pitch_i=interp1(t,x,ti);
x=roll_1_IMU_deg(3,492:3022);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_roll_1_quarc_deg(1,1:2705)-...
    (deg1_roll_1(1)-deg_roll_1(1)))))
```

```
ylim([0 4]), title('Roll')
xlim([0 27.05])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_roll_1_quarc_deg(2,1:2705)-...
    (deg1_roll_1(2)-deg_roll_1(2)))))
ylim([0 4]), title('Pitch')
xlim([0 27.05])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_roll_1_quarc_deg(3,1:2705)-...
    (deg1_roll_1(3)-deg_roll_1(3)))))
ylim([0 4]), title('Yaw')
xlim([0 27.05])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Roll_Diff.jpeg')

roll_diff=abs(roll_i'-(fast_roll_1_quarc_deg(1,1:2705)-...
    (deg1_roll_1(1)-deg_roll_1(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2704).^2))

pitch_diff=abs(pitch_i'-(fast_roll_1_quarc_deg(2,1:2705)-...
    (deg1_roll_1(2)-deg_roll_1(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2704).^2))

yaw_diff=abs(yaw_i'-(fast_roll_1_quarc_deg(3,1:2705)-...
    (deg1_roll_1(3)-deg_roll_1(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2704).^2))

%Roll 2
ti=t_quarc_fast_roll_2(1:1921);
t=t_IMU_roll_2(465:2289)-t_IMU_roll_2(465);
x=roll_2_IMU_deg(1,465:2289);
roll_i=interp1(t,x,ti);
x=roll_2_IMU_deg(2,465:2289);
pitch_i=interp1(t,x,ti);
x=roll_2_IMU_deg(3,465:2289);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_roll_2_quarc_deg(1,1:1921)-...
    (deg1_roll_2(1)-deg_roll_2(1)))))
ylim([0 4]), title('Roll')
xlim([0 19.21])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_roll_2_quarc_deg(2,1:1921)-...
    (deg1_roll_2(2)-deg_roll_2(2)))))
ylim([0 4]), title('Pitch')
xlim([0 19.21])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_roll_2_quarc_deg(3,1:1921)-...
    (deg1_roll_2(3)-deg_roll_2(3)))))
ylim([0 4]), title('Yaw')
xlim([0 19.21])

roll_diff=abs(roll_i'-(fast_roll_2_quarc_deg(1,1:1921)-...
    (deg1_roll_2(1)-deg_roll_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:1921).^2))

pitch_diff=abs(pitch_i'-(fast_roll_2_quarc_deg(2,1:1921)-...
    (deg1_roll_2(2)-deg_roll_2(2))));
max_pitch_diff=max(pitch_diff)
```

```
rms_pitch_diff=sqrt(mean(pitch_diff(1:1921).^2))

yaw_diff=abs(yaw_i'-(fast_roll_2_quarc_deg(3,1:1921)-...
    (deg1_roll_2(3)-deg_roll_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:1921).^2))

%Roll 3
ti=t_quarc_fast_roll_3(1:2315);
t=t_IMU_roll_3(482:2702)-t_IMU_roll_3(482);
x=roll_3_IMU_deg(1,482:2702);
roll_i=interp1(t,x,ti);
x=roll_3_IMU_deg(2,482:2702);
pitch_i=interp1(t,x,ti);
x=roll_3_IMU_deg(3,482:2702);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_roll_3_quarc_deg(1,1:2315)-...
    (deg1_roll_3(1)-deg_roll_3(1)))))
ylim([0 4]), title('Roll')
xlim([0 23.15])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_roll_3_quarc_deg(2,1:2315)-...
    (deg1_roll_3(2)-deg_roll_3(2)))))
ylim([0 4]), title('Pitch')
xlim([0 23.15])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_roll_3_quarc_deg(3,1:2315)-...
    (deg1_roll_3(3)-deg_roll_3(3)))))
ylim([0 4]), title('Yaw')
xlim([0 23.15])

roll_diff=abs(roll_i'-(fast_roll_3_quarc_deg(1,1:2315)-...
    (deg1_roll_3(1)-deg_roll_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2314).^2))

pitch_diff=abs(pitch_i'-(fast_roll_3_quarc_deg(2,1:2315)-...
    (deg1_roll_3(2)-deg_roll_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2314).^2))

yaw_diff=abs(yaw_i'-(fast_roll_3_quarc_deg(3,1:2315)-...
    (deg1_roll_3(3)-deg_roll_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2314).^2))

%Pitch 1
ti=t_quarc_fast_pitch_1(1:2163);
t=t_IMU_pitch_1(466:2536)-t_IMU_pitch_1(466);
x=pitch_1_IMU_deg(1,466:2536);
roll_i=interp1(t,x,ti);
x=pitch_1_IMU_deg(2,466:2536);
pitch_i=interp1(t,x,ti);
x=pitch_1_IMU_deg(3,466:2536);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_pitch_1_quarc_deg(1,1:2163)-...
    (deg1_pitch_1(1)-deg_pitch_1(1)))))
ylim([0 4]), title('Roll')
xlim([0 21.64])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_pitch_1_quarc_deg(2,1:2163)-...
```

```
    (deg1_pitch_1(2)-deg_pitch_1(2)))))
ylim([0 4]), title('Pitch')
xlim([0 21.64])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_pitch_1_quarc_deg(3,1:2163)-...
    (deg1_pitch_1(3)-deg_pitch_1(3)))))
ylim([0 4]), title('Yaw')
xlim([0 21.64])

roll_diff=abs(roll_i'-(fast_pitch_1_quarc_deg(1,1:2163)-...
    (deg1_pitch_1(1)-deg_pitch_1(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2163).^2))

pitch_diff=abs(pitch_i'-(fast_pitch_1_quarc_deg(2,1:2163)-...
    (deg1_pitch_1(2)-deg_pitch_1(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2163).^2))

yaw_diff=abs(yaw_i'-(fast_pitch_1_quarc_deg(3,1:2163)-...
    (deg1_pitch_1(3)-deg_pitch_1(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2163).^2))

%Pitch 2
ti=t_quarc_fast_pitch_2(1:2368);
t=t_IMU_pitch_2(488:2680)-t_IMU_pitch_2(488);
x=pitch_2_IMU_deg(1,488:2680);
roll_i=interp1(t,x,ti);
x=pitch_2_IMU_deg(2,488:2680);
pitch_i=interp1(t,x,ti);
x=pitch_2_IMU_deg(3,488:2680);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_pitch_2_quarc_deg(1,1:2368)-...
    (deg1_pitch_2(1)-deg_pitch_2(1)))))
ylim([0 4]), title('Roll')
xlim([0 23.68])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_pitch_2_quarc_deg(2,1:2368)-...
    (deg1_pitch_2(2)-deg_pitch_2(2)))))
ylim([0 4]), title('Pitch')
xlim([0 23.68])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_pitch_2_quarc_deg(3,1:2368)-...
    (deg1_pitch_2(3)-deg_pitch_2(3)))))
ylim([0 4]), title('Yaw')
xlim([0 23.68])

roll_diff=abs(roll_i'-(fast_pitch_2_quarc_deg(1,1:2368)-...
    (deg1_pitch_2(1)-deg_pitch_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2367).^2))

pitch_diff=abs(pitch_i'-(fast_pitch_2_quarc_deg(2,1:2368)-...
    (deg1_pitch_2(2)-deg_pitch_2(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2367).^2))

yaw_diff=abs(yaw_i'-(fast_pitch_2_quarc_deg(3,1:2368)-...
    (deg1_pitch_2(3)-deg_pitch_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2367).^2))
```

```
%Pitch 3
ti=t_quarc_fast_pitch_3(1:2526);
t=t_IMU_pitch_3(507:2879)-t_IMU_pitch_3(507);
x=pitch_3_IMU_deg(1,507:2879);
roll_i=interp1(t,x,ti);
x=pitch_3_IMU_deg(2,507:2879);
pitch_i=interp1(t,x,ti);
x=pitch_3_IMU_deg(3,507:2879);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_pitch_3_quarc_deg(1,1:2526)-...
    (deg1_pitch_3(1)-deg_pitch_3(1)))))
ylim([0 4]), title('Roll')
xlim([0 25.26])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_pitch_3_quarc_deg(2,1:2526)-...
    (deg1_pitch_3(2)-deg_pitch_3(2)))))
ylim([0 4]), title('Pitch')
xlim([0 25.26])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_pitch_3_quarc_deg(3,1:2526)-...
    (deg1_pitch_3(3)-deg_pitch_3(3)))))
ylim([0 4]), title('Yaw')
xlim([0 25.26])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Pitch_Diff.jpeg')

roll_diff=abs(roll_i'-(fast_pitch_3_quarc_deg(1,1:2526)-...
    (deg1_pitch_3(1)-deg_pitch_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff.^2))

pitch_diff=abs(pitch_i'-(fast_pitch_3_quarc_deg(2,1:2526)-...
    (deg1_pitch_3(2)-deg_pitch_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff.^2))

yaw_diff=abs(yaw_i'-(fast_pitch_3_quarc_deg(3,1:2526)-...
    (deg1_pitch_3(3)-deg_pitch_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff.^2))

%Yaw 1
ti=t_quarc_fast_yaw_1(1:2299);
t=t_IMU_yaw_1(528:2709)-t_IMU_yaw_1(528);
x=yaw_1_IMU_deg(1,528:2709);
roll_i=interp1(t,x,ti);
x=yaw_1_IMU_deg(2,528:2709);
pitch_i=interp1(t,x,ti);
x=yaw_1_IMU_deg(3,528:2709);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_yaw_1_quarc_deg(1,1:2299)-...
    (deg1_yaw_1(1)-deg_yaw_1(1)))))
ylim([0 4]), title('Roll')
xlim([0 22.99])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_yaw_1_quarc_deg(2,1:2299)-...
    (deg1_yaw_1(2)-deg_yaw_1(2)))))
ylim([0 4]), title('Pitch')
xlim([0 22.99])
```

```
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_yaw_1_quarc_deg(3,1:2299)-...
    (deg1_yaw_1(3)-deg_yaw_1(3)))))
ylim([0 4]), title('Yaw')
xlim([0 22.99])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Yaw_Diff.jpeg')

roll_diff=abs(roll_i'-(fast_yaw_1_quarc_deg(1,1:2299)-...
    (deg1_yaw_1(1)-deg_yaw_1(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2298).^2))

pitch_diff=abs(pitch_i'-(fast_yaw_1_quarc_deg(2,1:2299)-...
    (deg1_yaw_1(2)-deg_yaw_1(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2298).^2))

yaw_diff=abs(yaw_i'-(fast_yaw_1_quarc_deg(3,1:2299)-...
    (deg1_yaw_1(3)-deg_yaw_1(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2298).^2))

%Yaw 2
ti=t_quarc_fast_yaw_2(1:2411);
t=t_IMU_yaw_2(512:2815)-t_IMU_yaw_2(512);
x=yaw_2_IMU_deg(1,512:2815);
roll_i=interp1(t,x,ti);
x=yaw_2_IMU_deg(2,512:2815);
pitch_i=interp1(t,x,ti);
x=yaw_2_IMU_deg(3,512:2815);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_yaw_2_quarc_deg(1,1:2411)-...
    (deg1_yaw_2(1)-deg_yaw_2(1)))))
ylim([0 4]), title('Roll')
xlim([0 24.11])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_yaw_2_quarc_deg(2,1:2411)-...
    (deg1_yaw_2(2)-deg_yaw_2(2)))))
ylim([0 4]), title('Pitch')
xlim([0 24.11])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_yaw_2_quarc_deg(3,1:2411)-...
    (deg1_yaw_2(3)-deg_yaw_2(3)))))
ylim([0 4]), title('Yaw')
xlim([0 24.11])

roll_diff=abs(roll_i'-(fast_yaw_2_quarc_deg(1,1:2411)-...
    (deg1_yaw_2(1)-deg_yaw_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2410).^2))

pitch_diff=abs(pitch_i'-(fast_yaw_2_quarc_deg(2,1:2411)-...
    (deg1_yaw_2(2)-deg_yaw_2(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2410).^2))

yaw_diff=abs(yaw_i'-(fast_yaw_2_quarc_deg(3,1:2411)-...
    (deg1_yaw_2(3)-deg_yaw_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2410).^2))
```

```
%Yaw 3
ti=t_quarc_fast_yaw_3(1:2390);
t=t_IMU_yaw_3(461:2667)-t_IMU_yaw_3(461);
x=yaw_3_IMU_deg(1,461:2667);
roll_i=interp1(t,x,ti);
x=yaw_3_IMU_deg(2,461:2667);
pitch_i=interp1(t,x,ti);
x=yaw_3_IMU_deg(3,461:2667);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_yaw_3_quarc_deg(1,1:2390)-...
    (deg1_yaw_3(1)-deg_yaw_3(1)))))
ylim([0 4]), title('Roll')
xlim([0 23.90])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_yaw_3_quarc_deg(2,1:2390)-...
    (deg1_yaw_3(2)-deg_yaw_3(2)))))
ylim([0 4]), title('Pitch')
xlim([0 23.90])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_yaw_3_quarc_deg(3,1:2390)-...
    (deg1_yaw_3(3)-deg_yaw_3(3)))))
ylim([0 4]), title('Yaw')
xlim([0 23.90])

roll_diff=abs(roll_i'-(fast_yaw_3_quarc_deg(1,1:2390)-...
    (deg1_yaw_3(1)-deg_yaw_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:2389).^2))

pitch_diff=abs(pitch_i'-(fast_yaw_3_quarc_deg(2,1:2390)-...
    (deg1_yaw_3(2)-deg_yaw_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:2389).^2))

yaw_diff=abs(yaw_i'-(fast_yaw_3_quarc_deg(3,1:2390)-...
    (deg1_yaw_3(3)-deg_yaw_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:2389).^2))
```

B. FUNCTION

The function used in conjuction with the code for fast motion in a single axis is contained in this section.

```
%Rotation Matrix to Quaternion
%By:  Heather Pelachick

function [quat]=rotm2quat(r)

q0=(1/2)*sqrt(r(1,1)+r(2,2)+r(3,3)+1);
q1=-(r(3,2)-r(2,3))/(4*q0);
q2=-(r(1,3)-r(3,1))/(4*q0);
q3=-(r(2,1)-r(1,2))/(4*q0);

quat=[q0 q1 q2 q3];
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MATLAB CODE FOR ARBITRARY MOTION

A. CODE

The code used during arbitrary motion is contained in this section.

```
%Arbitrary Motion
%By:  Heather Pelachick

%This code utilizies sensor measurements from the 3DM-GX4-25 and QUARC
%to determine the dynamic accuracy, angular rate, peak instantaneous error,
%angles of rotation plot, and difference plot for slow and fast arbitrary
%motions.

clear all;
close all;

%---------------------------- IMU ---------------------------------------

%Load Data - Quaternion
RNG='M17..P4131';
arb_1_IMU=csvread('Arb_1_IMU.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(arb_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Correct singularity
arb_deg=[RPY_deg(3,1:2309) RPY_deg(3,2310:2533)-360 RPY_deg(3,2534:4115)];
RPY_deg=[RPY_deg(1:2,:);arb_deg];

%Initial degree values
deg_arb_1=RPY_deg(:,505);

%Load Data - Time
RNG='C17..C4131';
t_IMU_arb_1=csvread('Arb_1_IMU.csv',16,2,RNG);
t_IMU_arb_1=t_IMU_arb_1-12393.525; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I4131';
gyro=csvread('Arb_1_IMU.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_arb_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
arb_1_IMU_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3800';
```

```
arb_2_IMU=csvread('Arb_2_IMU.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(arb_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Correct singularity
arb_deg=[RPY_deg(3,1:1683) RPY_deg(3,1684:1837)-360 RPY_deg(3,1838:2057)...
    RPY_deg(3,2058:2424)-360 RPY_deg(3,2425:3784)];
RPY_deg=[RPY_deg(1:2,:);arb_deg];

%Initial degree values
deg_arb_2=RPY_deg(:,490);

%Load Data - Time
RNG='C17..C3800';
t_IMU_arb_2=csvread('Arb_2_IMU.csv',16,2,RNG);
t_IMU_arb_2=t_IMU_arb_2-12804.191; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3800';
gyro=csvread('Arb_2_IMU.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_arb_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
arb_2_IMU_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3942';
arb_3_IMU=csvread('Arb_3_IMU.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(arb_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Correct singularity
arb_deg=[RPY_deg(3,1:2138) RPY_deg(3,2139:2538)-360 RPY_deg(3,2539:3926)];
RPY_deg=[RPY_deg(1:2,:);arb_deg];

%Initial degree values
deg_arb_3=RPY_deg(:,536);

%Load Data - Time
RNG='C17..C3942';
t_IMU_arb_3=csvread('Arb_3_IMU.csv',16,2,RNG);
t_IMU_arb_3=t_IMU_arb_3-13033.397; %Subtract timestamp

%Load Data - Gyro
```

```
RNG='G17..I3942';
gyro=csvread('Arb_3_IMU.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_arb_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
arb_3_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P3393';
fast_arb_1_IMU=csvread('Fast_Arb_1_IMU.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(fast_arb_1_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_fast_arb_1=RPY_deg(:,1455);

%Load Data - Time
RNG='C17..C3393';
t_IMU_fast_arb_1=csvread('Fast_Arb_1_IMU.csv',16,2,RNG);
t_IMU_fast_arb_1=t_IMU_fast_arb_1-13390.633; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I3393';
gyro=csvread('Fast_Arb_1_IMU.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_fast_arb_1(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
fast_arb_1_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2462';
fast_arb_2_IMU=csvread('Fast_Arb_2_IMU.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(fast_arb_2_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg_fast_arb_2=RPY_deg(:,465);

%Load Data - Time
RNG='C17..C2462';
```

```
t_IMU_fast_arb_2=csvread('Fast_Arb_2_IMU.csv',16,2,RNG);
t_IMU_fast_arb_2=t_IMU_fast_arb_2-13859.579; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2462';
gyro=csvread('Fast_Arb_2_IMU.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_fast_arb_2(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
fast_arb_2_IMU_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
RNG='M17..P2873';
fast_arb_3_IMU=csvread('Fast_Arb_3_IMU.csv',16,12,RNG);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(fast_arb_3_IMU);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Correct singularity
Fast_arb_deg=[RPY_deg(3,1:1566) RPY_deg(3,1567:1581)-360 ...
    RPY_deg(3,1582:2857)];
RPY_deg=[RPY_deg(1:2,:);Fast_arb_deg];

%Initial degree values
deg_fast_arb_3=RPY_deg(:,513);

%Load Data - Time
RNG='C17..C2873';
t_IMU_fast_arb_3=csvread('Fast_Arb_3_IMU.csv',16,2,RNG);
t_IMU_fast_arb_3=t_IMU_fast_arb_3-14145.225; %Subtract timestamp

%Load Data - Gyro
RNG='G17..I2873';
gyro=csvread('Fast_Arb_3_IMU.csv',16,6,RNG);
for ii=1:length(gyro)
    gyro_fast_arb_3(ii)=sqrt(gyro(ii,1)^2+gyro(ii,2)^2+gyro(ii,3)^2);
end

%Rename variable
fast_arb_3_IMU_deg=RPY_deg;

%------------------------------ QUARC -----------------------------------

%Rotation Matrix
rotm=[1 0 0; 0 0 1; 0 -1 0];

%Function to convert rotation matrix to quaternion
quat=rotm2quat(rotm);

%-------------------------------------------------------------------------

%Load Data - Quaternion
```

```
load('arb_quarc_1.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_arb_1=RPY_deg(:,1);

%Load Data - Time
load('t_arb_1');
t_quarc_arb_1=mytime;

%Rename variable
arb_1_quarc_deg=RPY_deg;

%--------------------------------------------------------------------------

%Load Data - Quaternion
load('arb_quarc_2.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_arb_2=RPY_deg(:,1);

%Load Data - Time
load('t_arb_2');
t_quarc_arb_2=mytime;

%Rename variable
arb_2_quarc_deg=RPY_deg;
```

```
%-------------------------------------------------------------------------

%Load Data - Quaternion
load('arb_quarc_3.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_arb_3=RPY_deg(:,1);

%Load Data - Time
load('t_arb_3');
t_quarc_arb_3=mytime;

%Rename variable
arb_3_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_arb_quarc_1.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_fast_arb_1=RPY_deg(:,1);

%Load Data - Time
load('t_fast_arb_1');
```

```
t_quarc_fast_arb_1=mytime;

%Rename variable
fast_arb_1_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_arb_quarc_2.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];

%Convert to degrees
RPY_deg=RPY*(180/pi);

%Correct singularity
fast_arb_deg=[RPY_deg(1,1:888) RPY_deg(1,889:896)+360 RPY_deg(1,897:2217)];
RPY_deg=[fast_arb_deg;RPY_deg(2:3,:)];

%Initial degree values
deg1_fast_arb_2=RPY_deg(:,1);

%Load Data - Time
load('t_fast_arb_2');
t_quarc_fast_arb_2=mytime;

%Rename variable
fast_arb_2_quarc_deg=RPY_deg;

%-------------------------------------------------------------------------

%Load Data - Quaternion
load('fast_arb_quarc_3.mat')

%Change variable name of loaded data
u=simout.signals.values;

%Correct Y axis
u=[u(:,4) u(:,1) -u(:,2) u(:,3)];

%Translate into IMU coordinates
u_c=quatmultiply(quat,u);

%Calculate Roll, Ptich, Yaw using MATLAB Function quat2angle
[yaw, pitch, roll]= quat2angle(u_c);

%Enter output from quat2angle into matrix
RPY=[roll';pitch';yaw'];
```

```
%Convert to degrees
RPY_deg=RPY*(180/pi);

%Initial degree values
deg1_fast_arb_3=RPY_deg(:,1);

%Load Data - Time
load('t_fast_arb_3');
t_quarc_fast_arb_3=mytime;

%Rename variable
fast_arb_3_quarc_deg=RPY_deg;

%--------------------Compare Angles of Rotation (Together)-----------------

%Compare Arb 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(1,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(1,1:3856)-...
    (deg1_arb_1(1)-deg_arb_1(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
xlim([0 38.56])

subplot(3,1,2), title('Pitch')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(2,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(2,1:3856)-...
    (deg1_arb_1(2)-deg_arb_1(2)),'r')
hold off
xlim([0 38.56])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(3,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(3,1:3856)-...
    (deg1_arb_1(3)-deg_arb_1(3)),'r')
hold off
xlim([0 38.56])

%Compare Arb 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_arb_2(490:3784)-t_IMU_arb_2(490),arb_2_IMU_deg(1,490:3784))
plot(t_quarc_arb_2(1:3500),arb_2_quarc_deg(1,1:3500)-...
    (deg1_arb_2(1)-deg_arb_2(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
xlim([0 35.00])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_arb_2(490:3784)-t_IMU_arb_2(490),arb_2_IMU_deg(2,490:3784))
plot(t_quarc_arb_2(1:3500),arb_2_quarc_deg(2,1:3500)-...
    (deg1_arb_2(2)-deg_arb_2(2)),'r')
hold off
xlim([0 35.00])
ylabel({'Angles of Rotation (Degrees)';' '})
```

```
subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_arb_2(490:3784)-t_IMU_arb_2(490),arb_2_IMU_deg(3,490:3784))
plot(t_quarc_arb_2(1:3500),arb_2_quarc_deg(3,1:3500)-...
    (deg1_arb_2(3)-deg_arb_2(3)),'r')
hold off
xlim([0 35.00])
xlabel('Time (Seconds)')

saveas(gcf,'Arb_Angles.jpeg')

%Compare Arb 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_arb_3(536:3926)-t_IMU_arb_3(536),arb_3_IMU_deg(1,536:3926))
plot(t_quarc_arb_3(1:3557),arb_3_quarc_deg(1,1:3557)-...
    (deg1_arb_3(1)-deg_arb_3(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
xlim([0 35.57])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_arb_3(536:3926)-t_IMU_arb_3(536),arb_3_IMU_deg(2,536:3926))
plot(t_quarc_arb_3(1:3557),arb_3_quarc_deg(2,1:3557)-...
    (deg1_arb_3(2)-deg_arb_3(2)),'r')
hold off
xlim([0 35.57])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_arb_3(536:3926)-t_IMU_arb_3(536),arb_3_IMU_deg(3,536:3926))
plot(t_quarc_arb_3(1:3557),arb_3_quarc_deg(3,1:3557)-...
    (deg1_arb_3(3)-deg_arb_3(3)),'r')
hold off
xlim([0 35.57])

%Compare Fast Arb 1
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455),...
    fast_arb_1_IMU_deg(1,1455:3377))
plot(t_quarc_fast_arb_1(1:1995),fast_arb_1_quarc_deg(1,1:1995)-...
    (deg1_fast_arb_1(1)-deg_fast_arb_1(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
xlim([0 19.95])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455),...
    fast_arb_1_IMU_deg(2,1455:3377))
plot(t_quarc_fast_arb_1(1:1995),fast_arb_1_quarc_deg(2,1:1995)-...
    (deg1_fast_arb_1(2)-deg_fast_arb_1(2)),'r')
hold off
xlim([0 19.95])

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455),...
```

```
    fast_arb_1_IMU_deg(3,1455:3377))
plot(t_quarc_fast_arb_1(1:1995),fast_arb_1_quarc_deg(3,1:1995)-...
    (deg1_fast_arb_1(3)-deg_fast_arb_1(3)),'r')
hold off
xlim([0 19.95])

%Compare Fast Arb 2
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(1,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(1,1:2091)-...
    (deg1_fast_arb_2(1)-deg_fast_arb_2(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
xlim([0 20.91])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(2,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(2,1:2091)-...
    (deg1_fast_arb_2(2)-deg_fast_arb_2(2)),'r')
hold off
xlim([0 20.91])
ylabel({'Angles of Rotation (Degrees)';' '})

subplot(3,1,3),title('Yaw')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(3,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(3,1:2091)-...
    (deg1_fast_arb_2(3)-deg_fast_arb_2(3)),'r')
hold off
xlim([0 20.91])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Arb_Angles.jpeg')

%Compare Fast Arb 3
figure
subplot(3,1,1), title('Roll')
hold on
plot(t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513),...
    fast_arb_3_IMU_deg(1,513:2857))
plot(t_quarc_fast_arb_3(1:2499),fast_arb_3_quarc_deg(1,1:2499)-...
    (deg1_fast_arb_3(1)-deg_fast_arb_3(1)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
xlim([0 24.99])

subplot(3,1,2),title('Pitch')
hold on
plot(t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513),...
    fast_arb_3_IMU_deg(2,513:2857))
plot(t_quarc_fast_arb_3(1:2499),fast_arb_3_quarc_deg(2,1:2499)-...
    (deg1_fast_arb_3(2)-deg_fast_arb_3(2)),'r')
hold off
xlim([0 24.99])

subplot(3,1,3),title('Yaw')
```

```
hold on
plot(t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513),...
    fast_arb_3_IMU_deg(3,513:2857))
plot(t_quarc_fast_arb_3(1:2499),fast_arb_3_quarc_deg(3,1:2499)-...
    (deg1_fast_arb_3(3)-deg_fast_arb_3(3)),'r')
hold off
xlim([0 24.99])

%--------------------Compare Angles of Rotation (Separate)--------------------

%Compare Arb 1
figure,title('Roll')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(1,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(1,1:3856)-(deg1_arb_1(1)-...
    deg_arb_1(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 38.56])

figure, title('Pitch')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(2,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(2,1:3856)-(deg1_arb_1(2)-...
    deg_arb_1(2)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 38.56])

figure,title('Yaw')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(3,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(3,1:3856)-(deg1_arb_1(3)-...
    deg_arb_1(3)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 38.56])

figure,title('Yaw')
hold on
plot(t_IMU_arb_1(505:4115)-t_IMU_arb_1(505),arb_1_IMU_deg(3,505:4115))
plot(t_quarc_arb_1(1:3856),arb_1_quarc_deg(3,1:3856)-(deg1_arb_1(3)-...
    deg_arb_1(3)),'r')
hold off
legend('MARG','QUARC','Location','NorthEast')
axis([10 26 -161 -85])
ylabel({'Angles of Rotation (Degrees)';' '})
xlabel('Time (Seconds)')
saveas(gcf,'Yaw_Trial 1.jpeg')

%Compare Arb 2
figure, title('Roll')
hold on
plot(t_IMU_arb_2(490:3784)-t_IMU_arb_2(490),arb_2_IMU_deg(1,490:3784))
plot(t_quarc_arb_2(1:3500),arb_2_quarc_deg(1,1:3500)-(deg1_arb_2(1)-...
    deg_arb_2(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 35.00])

figure,title('Pitch')
hold on
```

```
plot(t_IMU_arb_2(490:3784)-t_IMU_arb_2(490),arb_2_IMU_deg(2,490:3784))
plot(t_quarc_arb_2(1:3500),arb_2_quarc_deg(2,1:3500)-(deg1_arb_2(2)-...
    deg_arb_2(2)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 35.00])

figure,title('Yaw')
hold on
plot(t_IMU_arb_2(490:3784)-t_IMU_arb_2(490),arb_2_IMU_deg(3,490:3784))
plot(t_quarc_arb_2(1:3500),arb_2_quarc_deg(3,1:3500)-(deg1_arb_2(3)-...
    deg_arb_2(3)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 35.00])

%Compare Arb 3
figure, title('Roll')
hold on
plot(t_IMU_arb_3(536:3926)-t_IMU_arb_3(536),arb_3_IMU_deg(1,536:3926))
plot(t_quarc_arb_3(1:3557),arb_3_quarc_deg(1,1:3557)-(deg1_arb_3(1)-...
    deg_arb_3(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 35.57])

figure,title('Pitch')
hold on
plot(t_IMU_arb_3(536:3926)-t_IMU_arb_3(536),arb_3_IMU_deg(2,536:3926))
plot(t_quarc_arb_3(1:3557),arb_3_quarc_deg(2,1:3557)-(deg1_arb_3(2)-...
    deg_arb_3(2)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 35.57])

figure,title('Yaw')
hold on
plot(t_IMU_arb_3(536:3926)-t_IMU_arb_3(536),arb_3_IMU_deg(3,536:3926))
plot(t_quarc_arb_3(1:3557),arb_3_quarc_deg(3,1:3557)-(deg1_arb_3(3)-...
    deg_arb_3(3)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 35.57])

%Compare Fast Arb 1
figure, title('Roll')
hold on
plot(t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455),...
    fast_arb_1_IMU_deg(1,1455:3377))
plot(t_quarc_fast_arb_1(1:1995),fast_arb_1_quarc_deg(1,1:1995)-...
    (deg1_fast_arb_1(1)-deg_fast_arb_1(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 19.95])

figure,title('Pitch')
hold on
plot(t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455),...
    fast_arb_1_IMU_deg(2,1455:3377))
plot(t_quarc_fast_arb_1(1:1995),fast_arb_1_quarc_deg(2,1:1995)-...
    (deg1_fast_arb_1(2)-deg_fast_arb_1(2)),'r')
hold off
```

```
legend('MARG','QUARC','Location','SouthEast')
xlim([0 19.95])

figure,title('Yaw')
hold on
plot(t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455),...
    fast_arb_1_IMU_deg(3,1455:3377))
plot(t_quarc_fast_arb_1(1:1995),fast_arb_1_quarc_deg(3,1:1995)-...
    (deg1_fast_arb_1(3)-deg_fast_arb_1(3)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 19.95])

%Compare Fast Arb 2
figure, title('Roll')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(1,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(1,1:2091)-...
    (deg1_fast_arb_2(1)-deg_fast_arb_2(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 20.91])

figure, title('Roll')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(1,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(1,1:2091)-...
    (deg1_fast_arb_2(1)-deg_fast_arb_2(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
axis([7.75 10.25 -60 95])
xlabel('Time (Seconds)')
ylabel({'Angles of Rotation (Degrees)';' '})
saveas(gcf,'fast_arb_roll.jpeg')

figure,title('Pitch')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(2,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(2,1:2091)-...
    (deg1_fast_arb_2(2)-deg_fast_arb_2(2)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 20.91])

figure,title('Yaw')
hold on
plot(t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465),...
    fast_arb_2_IMU_deg(3,465:2446))
plot(t_quarc_fast_arb_2(1:2091),fast_arb_2_quarc_deg(3,1:2091)-...
    (deg1_fast_arb_2(3)-deg_fast_arb_2(3)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 20.91])

%Compare Fast Arb 3
figure, title('Roll')
hold on
plot(t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513),...
    fast_arb_3_IMU_deg(1,513:2857))
```

```
plot(t_quarc_fast_arb_3(1:2499),fast_arb_3_quarc_deg(1,1:2499)-...
    (deg1_fast_arb_3(1)-deg_fast_arb_3(1)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 24.99])

figure,title('Pitch')
hold on
plot(t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513),...
    fast_arb_3_IMU_deg(2,513:2857))
plot(t_quarc_fast_arb_3(1:2499),fast_arb_3_quarc_deg(2,1:2499)-...
    (deg1_fast_arb_3(2)-deg_fast_arb_3(2)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 24.99])

figure,title('Yaw')
hold on
plot(t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513),...
    fast_arb_3_IMU_deg(3,513:2857))
plot(t_quarc_fast_arb_3(1:2499),fast_arb_3_quarc_deg(3,1:2499)-...
    (deg1_fast_arb_3(3)-deg_fast_arb_3(3)),'r')
hold off
legend('MARG','QUARC','Location','SouthEast')
xlim([0 24.99])

%-------------------------Difference--------------------------------------

%Arb 1
ti=t_quarc_arb_1(1:3856);
t=t_IMU_arb_1(505:4115)-t_IMU_arb_1(505);
x=arb_1_IMU_deg(1,505:4115);
roll_i=interp1(t,x,ti);
x=arb_1_IMU_deg(2,505:4115);
pitch_i=interp1(t,x,ti);
x=arb_1_IMU_deg(3,505:4115);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(arb_1_quarc_deg(1,1:3856)-...
    (deg1_arb_1(1)-deg_arb_1(1)))))
title('Roll')
xlim([0 38.56])
ylim([0 20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(arb_1_quarc_deg(2,1:3856)-...
    (deg1_arb_1(2)-deg_arb_1(2)))))
title('Pitch')
xlim([0 38.56])
ylim([0 20])
subplot(3,1,3),plot(ti,abs(yaw_i'-(arb_1_quarc_deg(3,1:3856)-...
    (deg1_arb_1(3)-deg_arb_1(3)))))
title('Yaw')
xlim([0 38.56])
ylim([0 20])

roll_diff=abs(roll_i'-(arb_1_quarc_deg(1,1:3856)-...
    (deg1_arb_1(1)-deg_arb_1(1))));
roll_diff=[roll_diff(1:2140) roll_diff(2186:2210) roll_diff(2212:3856)];
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff.^2))

pitch_diff=abs(pitch_i'-(arb_1_quarc_deg(2,1:3856)-...
```

```
    (deg1_arb_1(2)-deg_arb_1(2))));
pitch_diff=[pitch_diff(1:2140) pitch_diff(2186:2210) ...
    pitch_diff(2212:3856)];
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff.^2))

yaw_diff=abs(yaw_i'-(arb_1_quarc_deg(3,1:3856)-...
    (deg1_arb_1(3)-deg_arb_1(3))));
yaw_diff=[yaw_diff(1:2140) yaw_diff(2186:2210) yaw_diff(2212:3856)];
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff.^2))

%Arb 2
ti=t_quarc_arb_2(1:3500);
t=t_IMU_arb_2(490:3784)-t_IMU_arb_2(490);
x=arb_2_IMU_deg(1,490:3784);
roll_i=interp1(t,x,ti);
x=arb_2_IMU_deg(2,490:3784);
pitch_i=interp1(t,x,ti);
x=arb_2_IMU_deg(3,490:3784);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(arb_2_quarc_deg(1,1:3500)-...
    (deg1_arb_2(1)-deg_arb_2(1)))))
title('Roll')
xlim([0 35.00])
ylim([0 20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(arb_2_quarc_deg(2,1:3500)-...
    (deg1_arb_2(2)-deg_arb_2(2)))))
title('Pitch')
xlim([0 35.00])
ylim([0 20])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(arb_2_quarc_deg(3,1:3500)-...
    (deg1_arb_2(3)-deg_arb_2(3)))))
title('Yaw')
xlim([0 35.00])
ylim([0 20])
xlabel('Time (Seconds)')

saveas(gcf,'Arb_Diff.jpeg')

roll_diff=abs(roll_i'-(arb_2_quarc_deg(1,1:3500)-...
    (deg1_arb_2(1)-deg_arb_2(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:3499).^2))

pitch_diff=abs(pitch_i'-(arb_2_quarc_deg(2,1:3500)-...
    (deg1_arb_2(2)-deg_arb_2(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:3499).^2))

yaw_diff=abs(yaw_i'-(arb_2_quarc_deg(3,1:3500)-...
    (deg1_arb_2(3)-deg_arb_2(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:3499).^2))

%Arb 3
ti=t_quarc_arb_3(1:3557);
t=t_IMU_arb_3(536:3926)-t_IMU_arb_3(536);
x=arb_3_IMU_deg(1,536:3926);
```

```
roll_i=interp1(t,x,ti);
x=arb_3_IMU_deg(2,536:3926);
pitch_i=interp1(t,x,ti);
x=arb_3_IMU_deg(3,536:3926);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(arb_3_quarc_deg(1,1:3557)-...
    (deg1_arb_3(1)-deg_arb_3(1)))))
title('Roll')
xlim([0 35.57])
ylim([0 20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(arb_3_quarc_deg(2,1:3557)-...
    (deg1_arb_3(2)-deg_arb_3(2)))))
title('Pitch')
xlim([0 35.57])
ylim([0 20])
subplot(3,1,3),plot(ti,abs(yaw_i'-(arb_3_quarc_deg(3,1:3557)-...
    (deg1_arb_3(3)-deg_arb_3(3)))))
title('Yaw')
xlim([0 35.57])
ylim([0 20])

roll_diff=abs(roll_i'-(arb_3_quarc_deg(1,1:3557)-...
    (deg1_arb_3(1)-deg_arb_3(1))));
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff(1:3556).^2))

pitch_diff=abs(pitch_i'-(arb_3_quarc_deg(2,1:3557)-...
    (deg1_arb_3(2)-deg_arb_3(2))));
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff(1:3556).^2))

yaw_diff=abs(yaw_i'-(arb_3_quarc_deg(3,1:3557)-...
    (deg1_arb_3(3)-deg_arb_3(3))));
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff(1:3556).^2))

%Fast Arb 1
ti=t_quarc_fast_arb_1(1:1995);
t=t_IMU_fast_arb_1(1455:3377)-t_IMU_fast_arb_1(1455);
x=fast_arb_1_IMU_deg(1,1455:3377);
roll_i=interp1(t,x,ti);
x=fast_arb_1_IMU_deg(2,1455:3377);
pitch_i=interp1(t,x,ti);
x=fast_arb_1_IMU_deg(3,1455:3377);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_arb_1_quarc_deg(1,1:1995)-...
    (deg1_fast_arb_1(1)-deg_fast_arb_1(1)))))
title('Roll')
xlim([0 19.95])
ylim([0 20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_arb_1_quarc_deg(2,1:1995)-...
    (deg1_fast_arb_1(2)-deg_fast_arb_1(2)))))
title('Pitch')
xlim([0 19.95])
ylim([0 20])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_arb_1_quarc_deg(3,1:1995)-...
    (deg1_fast_arb_1(3)-deg_fast_arb_1(3)))))
title('Yaw')
```

```
xlim([0 19.95])
ylim([0 20])

roll_diff=abs(roll_i'-(fast_arb_1_quarc_deg(1,1:1995)-...
    (deg1_fast_arb_1(1)-deg_fast_arb_1(1))));
roll_diff=[roll_diff(1:794) roll_diff(800:1994)];
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff.^2))

pitch_diff=abs(pitch_i'-(fast_arb_1_quarc_deg(2,1:1995)-...
    (deg1_fast_arb_1(2)-deg_fast_arb_1(2))));
pitch_diff=[pitch_diff(1:794) pitch_diff(800:1994)];
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff.^2))

yaw_diff=abs(yaw_i'-(fast_arb_1_quarc_deg(3,1:1995)-...
    (deg1_fast_arb_1(3)-deg_fast_arb_1(3))));
yaw_diff=[yaw_diff(1:794) yaw_diff(800:1994)];
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff.^2))

%Fast Arb 2
ti=t_quarc_fast_arb_2(1:2091);
t=t_IMU_fast_arb_2(465:2446)-t_IMU_fast_arb_2(465);
x=fast_arb_2_IMU_deg(1,465:2446);
roll_i=interp1(t,x,ti);
x=fast_arb_2_IMU_deg(2,465:2446);
pitch_i=interp1(t,x,ti);
x=fast_arb_2_IMU_deg(3,465:2446);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_arb_2_quarc_deg(1,1:2091)-...
    (deg1_fast_arb_2(1)-deg_fast_arb_2(1)))))
title('Roll')
xlim([0 20.91])
ylim([0 20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_arb_2_quarc_deg(2,1:2091)-...
    (deg1_fast_arb_2(2)-deg_fast_arb_2(2)))))
title('Pitch')
xlim([0 20.91])
ylim([0 20])
ylabel({'Angles of Rotation (Degrees)';' '})
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_arb_2_quarc_deg(3,1:2091)-...
    (deg1_fast_arb_2(3)-deg_fast_arb_2(3)))))
title('Yaw')
xlim([0 20.91])
ylim([0 20])
xlabel('Time (Seconds)')

saveas(gcf,'Fast_Arb_Diff.jpeg')

roll_diff=abs(roll_i'-(fast_arb_2_quarc_deg(1,1:2091)-...
    (deg1_fast_arb_2(1)-deg_fast_arb_2(1))));
roll_diff=[roll_diff(1:811) roll_diff(817) roll_diff(823:824) ...
    roll_diff(835:2090)];
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff.^2))

pitch_diff=abs(pitch_i'-(fast_arb_2_quarc_deg(2,1:2091)-...
    (deg1_fast_arb_2(2)-deg_fast_arb_2(2))));
pitch_diff=[pitch_diff(1:811) pitch_diff(817) pitch_diff(823:824) ...
```

```
    pitch_diff(835:2090)];
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff.^2))

yaw_diff=abs(yaw_i'-(fast_arb_2_quarc_deg(3,1:2091)-...
    (deg1_fast_arb_2(3)-deg_fast_arb_2(3))));
yaw_diff=[yaw_diff(1:811) yaw_diff(817) yaw_diff(823:824) ...
    yaw_diff(835:2090)];
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff.^2))

%Fast Arb 3
ti=t_quarc_fast_arb_3(1:2499);
t=t_IMU_fast_arb_3(513:2857)-t_IMU_fast_arb_3(513);
x=fast_arb_3_IMU_deg(1,513:2857);
roll_i=interp1(t,x,ti);
x=fast_arb_3_IMU_deg(2,513:2857);
pitch_i=interp1(t,x,ti);
x=fast_arb_3_IMU_deg(3,513:2857);
yaw_i=interp1(t,x,ti);

figure
subplot(3,1,1),plot(ti,abs(roll_i'-(fast_arb_3_quarc_deg(1,1:2499)-...
    (deg1_fast_arb_3(1)-deg_fast_arb_3(1)))))
title('Roll')
xlim([0 24.99])
ylim([0 20])
subplot(3,1,2),plot(ti,abs(pitch_i'-(fast_arb_3_quarc_deg(2,1:2499)-...
    (deg1_fast_arb_3(2)-deg_fast_arb_3(2)))))
title('Pitch')
xlim([0 24.99])
ylim([0 20])
subplot(3,1,3),plot(ti,abs(yaw_i'-(fast_arb_3_quarc_deg(3,1:2499)-...
    (deg1_fast_arb_3(3)-deg_fast_arb_3(3)))))
title('Yaw')
xlim([0 24.99])
ylim([0 20])

roll_diff=abs(roll_i'-(fast_arb_3_quarc_deg(1,1:2499)-...
    (deg1_fast_arb_3(1)-deg_fast_arb_3(1))));
roll_diff=[roll_diff(1:1117) roll_diff(1123:1144) roll_diff(1150:1200) ...
    roll_diff(1221:1224) roll_diff(1230:1301) roll_diff(1307:1327) ...
    roll_diff(1333:1360) roll_diff(1371:2498)];
max_roll_diff=max(roll_diff)
rms_roll_diff=sqrt(mean(roll_diff.^2))

pitch_diff=abs(pitch_i'-(fast_arb_3_quarc_deg(2,1:2499)-...
    (deg1_fast_arb_3(2)-deg_fast_arb_3(2))));
pitch_diff=[pitch_diff(1:1117) pitch_diff(1123:1144) ...
    pitch_diff(1150:1200) pitch_diff(1221:1224) pitch_diff(1230:1301) ...
    pitch_diff(1307:1327) pitch_diff(1333:1360) pitch_diff(1371:2498)];
max_pitch_diff=max(pitch_diff)
rms_pitch_diff=sqrt(mean(pitch_diff.^2))

yaw_diff=abs(yaw_i'-(fast_arb_3_quarc_deg(3,1:2499)-...
    (deg1_fast_arb_3(3)-deg_fast_arb_3(3))));
yaw_diff=[yaw_diff(1:1117) yaw_diff(1123:1144) yaw_diff(1150:1200) ...
    yaw_diff(1221:1224) yaw_diff(1230:1301) yaw_diff(1307:1327) ...
    yaw_diff(1333:1360) yaw_diff(1371:2498)];
max_yaw_diff=max(yaw_diff)
rms_yaw_diff=sqrt(mean(yaw_diff.^2))
```

B. FUNCTION

The function used in conjuction with the code for arbitrary motion is contained in this section.

```
%Rotation Matrix to Quaternion
%By:  Heather Pelachick

function [quat]=rotm2quat(r)

q0=(1/2)*sqrt(r(1,1)+r(2,2)+r(3,3)+1);
q1=-(r(3,2)-r(2,3))/(4*q0);
q2=-(r(1,3)-r(3,1))/(4*q0);
q3=-(r(2,1)-r(1,2))/(4*q0);

quat=[q0 q1 q2 q3];
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] X. Yun, E. R. Bachmann, A. Kavousanos-Kavousanakis, F. Yildiz and R. B. McGhee, "Design and implementation of the MARG human body motion tracking system," *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 625–630, vol. 1.

[2] J. Keller, Military & Aerospace Electronics (Jun. 9, 2016), "Marine Corps asks industry for wearable IMUs to help measure infantry fatigue and performance," Available: http://www.militaryaerospace.com/articles/2016/06/infantry-wearable-imus.html. Accessed Nov. 10, 2016.

[3] J. Driesslein, "Scalable mobile ad hoc network (MANET) to enhance situational awareness in distributed small unit operations," M.S. thesis, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2015.

[4] C. Khan, "Geometry-of-fire tracking algorithm for direct-fire weapon systems," M.S. thesis, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2015.

[5] K. Reese, "A situational-awareness system for networked infantry including an accelerometer-based shot-identification algorithm for direct-fire weapons," M.S. thesis, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2016.

[6] A. Foushee, "Using posture estimation to enhance personal inertial tracking," M.S. thesis, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2016.

[7] J. L. Cookson, "A method for testing the dynamic accuracy of micro-electro-mechanical systems (MEMS) magnetic, angular rate, and gravity (MARG) sensors for inertial naviagation systems (INS) and human motion tracking applications," M.S. thesis, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2010.

[8] L. M. Landry, "Inertial sensor characterization for inertial navigations and human motion tracking applications," M.S. thesis, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2012.

[9] J. Stewart, *Calculus: Early Transcendentals*, 7th ed. Belmont: Brooks/Cole, Cengage Learning, 2012.

[10] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton: Princeton University Press, 1999.

[11] LORD User Manual 3DM-GX4-25 Attitude Heading Reference System (AHRS). (2015). LORD Corp. [Online]. Available: http://files.microstrain.com/3DM-GX4-25_User_Manual_%288500-0047%29.pdf. Accessed Nov.15, 2016.

[12] LORD Data Sheet 3DM-GX4-25 Attitude Heading Reference System (AHRS). (2016). LORD Corp. [Online]. Available: http://www.microstrain.com/sites/default/files/3dm-gx4-25_datasheet_8400-0060.pdf. Accessed Nov. 15, 2016.

[13] OptiTrack Flex 3. (2016). NaturalPoint, Inc. [Online]. Available: http://optittrack.com/products/flex-3.buy.html. Accessed Nov. 16, 2016.

[14] *Multi-Vehicle Coordination User Manual*, Markham, Ontario, Canada: Quanser Inc., 2012.

[15] MATLAB/Simulink and QUARC Primer. (2011). Quanser, Inc. [Online]. Available: http://www.quanser.com/Content/files/MATLAB-Simulink-QUARC-Primer.pdf. Accessed Nov. 22, 2016.

[16] J. Calusdian, "A personal navigation system based on inertial and magnetic field measurements," Ph.D. dissertation, Dept. Elect. Comput. Eng., Naval Postgraduate School, Monterey, CA, 2010.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California