Spring 2015

Calibration of Deterministic IMU Errors

Jeff Ferguson
Embry-Riddle Aeronautical University, fergusj5@my.erau.edu

Prescott, Arizona Campus
Department of Electrical and Computer Engineering

EE 495 Modern Navigation Systems: Theory & Practice

Spring Semester 2015

Instructor:
Dr. Stephen Bruder

Final Report
"Calibration of Deterministic IMU Errors"

Date of Presentation: Friday, April 17, 2014	**Instructor's Comments:**
Date Report Submitted: Monday, April 28, 2014	
Contact Information: Jeff Ferguson fergusj5@my.erau.edu	**Grade:**

TABLE OF CONTENTS PAGE

LIST OF TABLES PAGE

LIST OF FIGURES PAGE

List of Symbols

$\vec{b}$	Measurement bias vector
$\vec{b}_{FB}$	Measurement fixed bias vector
$\vec{f}_{ib}^{\,b}$	Inertial specific force vector in the body frame
g_l	Local gravity
G_g	G-dependent bias
M	Scale-factor and misalignment error matrix
$m_{A,B}$	Single misalignment term. A onto B.
s_A	Single scale factor term. Axis A.
$\vec{\omega}_{ib}^{b}$	Inertial angular rate vector in the body frame
$\vec{\omega}$	Angular rate vector
$\vec{w}$	White noise vector

List of Notation Standards

x	True x
$\tilde{x}$	Measurement of x
$\hat{x}$	Estimate of x
$\overline{x}$	Average of x
x_a	Corresponds to accelerometer
x_g	Corresponds to gyroscope

1. ABSTRACT

The VectorNav VN-200 is a MEMS navigation system that includes an accelerometer and gyroscope, as well as other navigation sensors. The accelerometer and gyroscope alone make up a commercial grade inertial measurement unit (IMU). Commercial grade IMUs produce poor performance on their own, and when used in navigation systems they can accumulate significant errors in less than a minute of operation. However, large increases in IMU accuracy can be achieved by properly calibrating out some of the IMUs deterministic errors. In this paper, methods are discussed for removing fixed biases, scale factor errors, and misalignment errors in both accelerometers and gyroscopes using a rate table. After discussing the methods for calibration, data collected from a VN-200 is used to calibrate the IMU. With the VN-200 calibrated, two separate tests using data collected from the sensor are performed to measure the increase in performance of the IMU as a dead-reckoning navigation system. In the two tests, the calibration reduced the calculated position error by 94.4% and 92.7% and the calculated attitude error by 96.6% and 94.7%. This substantial increase in accuracy by performing a simple calibration is so beneficial, it should performed for all accelerometers and gyroscopes whenever possible. At the very least, sensors should have their fixed biases removed before being used in navigation.

1.i. Objectives

- Calibrate the VN-200 to find the measurement errors due to fixed biases, scale factor errors, and misalignment errors.
- Run an IMU mechanization using data collected from the VN-200 following a known course. Confirm that calibrating the VN-200 improves the mechanization's estimation of position, velocity, and attitude.

2. INTRODUCTION & BACKGROUND

2.i. Deterministic Error Characteristics

Errors in an IMU's measurements can be broken down into three main categories: biases, scale-factor errors, and misalignment errors. IMU biases, $\vec{b}_a$ and $\vec{b}_g$ for the accelerometer and gyroscope respectively, are errors in the measurement that are present regardless of the forces or rates induced on the sensor. Biases can be further broken down into three types: fixed, stability, and instability. The fixed biases, or offsets, in the IMU measurements, $\vec{b}_{a,FB}$ and $\vec{b}_{g,FB}$, are the only biases that are deterministic in nature and can be calibrated out. Stability biases change randomly from run-to-run of the IMU, and instability biases change as a random process that is a function of time. Fixed biases can create one of the largest errors in an IMU mechanization; the errors increase at as a function of time squared for the accelerometer and time cubed for the gyroscope.

Scale factor errors describe how well the output of the sensor corresponds to a force or rate input. It allows for a way of modeling the difference between the ideal input-output slope of one and the linear measurement slope produced by the IMU. IMUs are often modeled with a linear response to simplify the calculation of measurements, but non-linearities can add further error to a measurement. The concept of scale factor error is illustrated in Figure 1: Scale Factor Error.

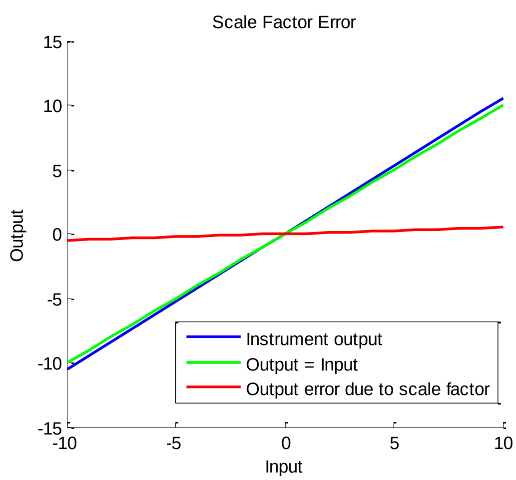

Figure 1: Scale Factor Error

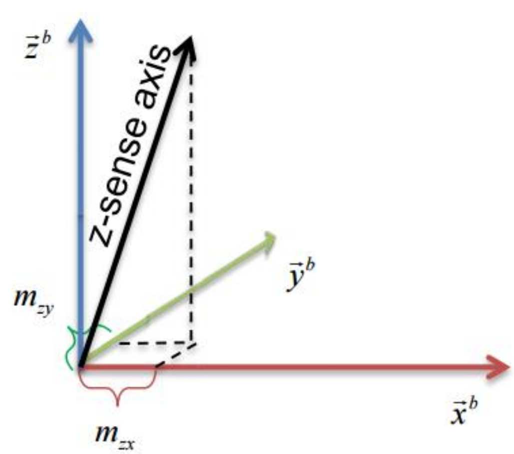

Figure 2: Misalignment Error [1]

Misalignment, or cross-coupling errors are due to imperfect construction or alignment of the three sensing axes in the accelerometer or gyroscope. As seen in Figure 2: Misalignment Error [1], if the sense-axis is not perfectly aligned with the body axis it is supposed to measure, forces on the other two axes will create false readings on the sense-axis. Scale factor and misalignment errors for an accelerometer or gyroscope are combined into a single matrix as shown in Equation (2.1) [2].

$$M_a = \begin{bmatrix} s_{a,x} & m_{a,xy} & m_{a,xz} \\ m_{a,yx} & s_{a,y} & m_{a,yz} \\ m_{a,zx} & m_{a,zy} & s_{a,z} \end{bmatrix} \qquad M_g = \begin{bmatrix} s_{g,x} & m_{g,xy} & m_{g,xz} \\ m_{g,yx} & s_{g,y} & m_{g,yz} \\ m_{g,zx} & m_{g,zy} & s_{g,z} \end{bmatrix} \tag{2.1}$$

2.ii. IMU Measurement Error Model

Equation (2.2) [2] shows how different errors sources each contribute skewing the accelerometer measurements away from the true specific forces incident on the IMU.

$$\tilde{\vec{f}}_{ib}^{b} = \vec{b}_a + (I_{3x3} + M_a)\,\vec{f}_{ib}^{b} + \vec{w}_a \tag{2.2}$$

Ignoring the white noise and two of the bias terms from Equation (2.2) reduces the measurement model to error terms that are deterministic. Equation (2.3) estimates the measurement model using just the terms that can be calibrated out.

$$\tilde{\vec{f}}_{ib}^{b} \approx \vec{b}_{a,FB} + (I_{3x3} + M_a)\,\vec{f}_{ib}^{b} \tag{2.3}$$

An estimate of the actual specific force can be calculated by rearranging Equation (2.3) to solve for the specific force in terms of the measurement and error terms. The estimate in Equation (2.4) is the calibration model for the accelerometer.

$$\hat{\vec{f}}_{ib}^{b} = (I_{3x3} + \hat{M}_a)^{-1}(\tilde{\vec{f}}_{ib}^{b} - \hat{\vec{b}}_{a,FB}) \tag{2.4}$$

A similar process can be taken for finding the calibration model of the gyroscope. Equation (2.5) [2] shows how different errors sources each contribute to skewing the gyroscope measurements away from the true angular rates incident on the IMU.

$$\tilde{\vec{\omega}}_{ib}^{b} = \vec{b}_g + (I_{3x3} + M_g)\tilde{\vec{\omega}}_{ib}^{b} + G_g\,\vec{f}_{ib}^{b} + \vec{w}_g \tag{2.5}$$

Eliminating the non-deterministic terms and the g-sensitivity term in Equation (2.5) produces Equation (2.6). The g-sensitivity term is deterministic and can be calibrated on some IMUs. However, VectorNav does not list an expected g-sensitivity for the VN-200 and does not offer any direction for calibrating the g-sensitivity of their gyroscopes.

$$\tilde{\vec{\omega}}_{ib}^{b} \approx \vec{b}_{g,FB} + (I_{3x3} + M_g)\vec{\omega}_{ib}^{b} \tag{2.6}$$

Because the profiles for testing the calibration and mechanization do not include large specific forces, the g-sensitivity term would, at most, create small errors, and little harm will come from ignoring it.

$$\hat{\tilde{\omega}}_{ib}^{b} = (I_{3x3} + \hat{M}_g)^{-1}(\tilde{\tilde{\omega}}_{ib}^{b} - \hat{\tilde{b}}_{g,FB}) \tag{2.7}$$

2.iii. IMU Mechanization

With the two calibration models complete, mechanization of captured data can be done to test the calibration models and methods. Measurements directly from the VN-200 will be input into the calibration models to create estimates of the specific force and angular rates. These values will then be input into a mechanization that will create a navigation profile from the inertial measurements. Data from the VN-200 is collected in a known inertial environment so that the generated navigation profile can be compared to the truth. The two known environments are: sitting still for 20 seconds, and rotating at 30 degrees per second in a fixed location for 20 seconds. Mechanization was done in the earth-centered-earth-fixed frame because the VN-200 does not translate relative to the frame, and thus the true position and velocity are constant.

3. METHODS AND APPROACH

For both the accelerometer and the gyroscope, measurements were taken to determine twelve calibration terms: three fixed bias values, three scale factor error values, and six misalignment error values. The measurements were collected from the VN-200 while it was on a rate table. The rate table allowed the VN-200 to be aligned at very precise orientations and rotated at very precise rates. The precision of the rate table was necessary to find the sensitive parameters. Data from the VN-200 was collected using the MATLAB script file in Appendix D: main_data_collection.m

3.i. Calibrating the Accelerometer

The twelve calibration variables can be found for the accelerometer using a single six-point tumble test. A tumble test aligns one of the sense-axes of the accelerometer with gravity and records measurements for 50-200 samples [4]. After recording the samples, the IMU is rotated so that the sense-axis now points in the opposite direction of gravity, and the measurements are taken again. Differences in the samples are compared to determine calibration. It is important to not take more than 200 samples because significant non-deterministic errors begin to enter IMU measurements after a short period of time. Below 50 samples, white noise can skew the average away from its true value.

3.i.a. Accelerometer Fixed Bias

The accelerometer fixed bias can be calculated using Equation (3.1). The + and – superscripts denote the positively and negatively aligned subscript axis, respectively. Just the average value of the measurements on the aligned axes are used in this calculation.

$$\hat{\bar{b}}_{a,FB} = \begin{bmatrix} \bar{\tilde{f}}_{ib,x}^{b,+} + \bar{\tilde{f}}_{ib,x}^{b,-} \\ \bar{\tilde{f}}_{ib,y}^{b,+} + \bar{\tilde{f}}_{ib,y}^{b,-} \\ \bar{\tilde{f}}_{ib,z}^{b,+} + \bar{\tilde{f}}_{ib,z}^{b,-} \end{bmatrix} \frac{1}{2} \tag{3.1}$$

This test works with the assumption that the fixed bias is the same regardless of the orientation or forces on the sensor. By adding the two opposing measurements together, the opposite gravity measurements will cancel each other out and twice the fixed bias will be left behind.

3.i.b. Accelerometer Scale Factor

Scale factor for the accelerometer is calculated in almost the opposite way from the fixed bias, as shown in Equation (3.2) [3]. This test, once again, only uses the measurements taken on the aligned sense-axis.

$$\begin{bmatrix} \hat{s}_{a,x} & 0 & 0 \\ 0 & \hat{s}_{a,y} & 0 \\ 0 & 0 & \hat{s}_{a,z} \end{bmatrix} = \begin{bmatrix} \bar{\tilde{f}}_{ib,x}^{b,+} - \bar{\tilde{f}}_{ib,x}^{b,-} & 0 & 0 \\ 0 & \bar{\tilde{f}}_{ib,y}^{b,+} - \bar{\tilde{f}}_{ib,y}^{b,-} & 0 \\ 0 & 0 & \bar{\tilde{f}}_{ib,z}^{b,+} - \bar{\tilde{f}}_{ib,z}^{b,-} \end{bmatrix} \frac{1}{2g_l} - I_{3x3} \tag{3.2}$$

By subtracting the two measurement averages, the effects of the fixed bias are removed, and the sensor’s output due to gravity are doubled. Because the input force, local gravity, is known, the input-to-output ratio can be calculated. The scale factor error is derived from this ratio.

3.i.c. Accelerometer Misalignment

Accelerometer misalignment is calculated in a similar way to the scale factor, but instead of measuring along the gravity aligned axis, the two non-gravity axes are measured to find the cross-coupling. These calculations are shown in Equation (3.3).

$$\begin{bmatrix} 0 & m_{a,xy} & m_{a,xz} \\ m_{a,yx} & 0 & m_{a,yz} \\ m_{a,zx} & m_{a,zy} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \bar{\tilde{f}}_{ib,y}^{b,+} - \bar{\tilde{f}}_{ib,y}^{b,-} & \bar{\tilde{f}}_{ib,z}^{b,+} - \bar{\tilde{f}}_{ib,z}^{b,-} \\ \bar{\tilde{f}}_{ib,x}^{b,+} - \bar{\tilde{f}}_{ib,x}^{b,-} & 0 & \bar{\tilde{f}}_{ib,z}^{b,+} - \bar{\tilde{f}}_{ib,z}^{b,-} \\ \bar{\tilde{f}}_{ib,x}^{b,+} - \bar{\tilde{f}}_{ib,x}^{b,-} & \bar{\tilde{f}}_{ib,y}^{b,+} - \bar{\tilde{f}}_{ib,y}^{b,-} & 0 \end{bmatrix} \frac{1}{2g_l} \tag{3.3}$$

The difference between the two measurements averages removes the fixed bias from the calculation and leaves twice the real measurement. Because the force incident on the cross-axis has a magnitude of local gravity, each element of the matrix is divided by $2g_l$. The arctangent of each element in the matrix is also taken to find the misalignment angle in radians.

3.ii. Calibrating the Gyroscope

Calibration of the gyroscope is done using equations that are essentially the same as the equations used in calibrating the accelerometer. The only element that changes is the input to find the scale factor and misalignment errors because gravity cannot be used.

3.ii.a. Gyroscope Fixed Bias

Intuitively, it seems a gyroscope's fixed bias could be calculated by fastening it in a fixed orientation and averaging the values measured on each sense axis, because the gyroscope is not rotating. This, however, does not work because gyroscopes are inertial sensors and are rotating inertially due to earth's rotation. High-resolution, low noise, gyroscopes can be used to measure the earth's rotation rate while sitting still. To offset the effect of earth rate, even if it cannot be measured on the VN-200, the sense-axis is once again aligned in opposite orientations to find the bias, Equation (3.4).

$$\hat{\vec{b}}_{g,FB} = \begin{bmatrix} \bar{\tilde{\omega}}_{ib,x}^{b,+} + \bar{\tilde{\omega}}_{ib,x}^{b,-} \\ \bar{\tilde{\omega}}_{ib,y}^{b,+} + \bar{\tilde{\omega}}_{ib,y}^{b,-} \\ \bar{\tilde{\omega}}_{ib,y}^{b,+} + \bar{\tilde{\omega}}_{ib,y}^{b,-} \end{bmatrix} \frac{1}{2} \tag{3.4}$$

3.ii.b. Gyroscope Scale Factor

Equation (3.5) shows the calculation for the gyroscope scale factor error.

$$\begin{bmatrix} \hat{s}_{g,x} & 0 & 0 \\ 0 & \hat{s}_{g,y} & 0 \\ 0 & 0 & \hat{s}_{g,z} \end{bmatrix} = \begin{bmatrix} \bar{\tilde{\omega}}_{ib,x}^{b,+} - \bar{\tilde{\omega}}_{ib,x}^{b,-} & 0 & 0 \\ 0 & \bar{\tilde{\omega}}_{ib,y}^{b,+} - \bar{\tilde{\omega}}_{ib,y}^{b,-} & 0 \\ 0 & 0 & \bar{\tilde{\omega}}_{ib,z}^{b,+} - \bar{\tilde{\omega}}_{ib,z}^{b,-} \end{bmatrix} \frac{1}{2\vec{\omega}} - I_{3x3} \tag{3.5}$$

This set of measurements requires setting the rate-table to rotate at a known rate around each of the sense-axes. This known rate, $\vec{\omega}$, is then divided out of the measurements to calculate the input-to-output error.

3.ii.c. Gyroscope Misalignment

The equation for calculating gyroscope misalignment, Equation (3.6), adapts the accelerometer misalignment equation in the same way as the scale factor error equations.

$$\begin{bmatrix} 0 & m_{g,xy} & m_{g,xz} \\ m_{g,yx} & 0 & m_{g,yz} \\ m_{g,zx} & m_{g,zy} & 0 \end{bmatrix} = \begin{bmatrix} 0 & \bar{\tilde{\omega}}_{ib,y}^{b,+} - \bar{\tilde{\omega}}_{ib,y}^{b,-} & \bar{\tilde{\omega}}_{ib,z}^{b,+} - \bar{\tilde{\omega}}_{ib,z}^{b,-} \\ \bar{\tilde{\omega}}_{ib,x}^{b,+} - \bar{\tilde{\omega}}_{ib,x}^{b,-} & 0 & \bar{\tilde{\omega}}_{ib,z}^{b,+} - \bar{\tilde{\omega}}_{ib,z}^{b,-} \\ \bar{\tilde{\omega}}_{ib,x}^{b,+} - \bar{\tilde{\omega}}_{ib,x}^{b,-} & \bar{\tilde{\omega}}_{ib,y}^{b,+} - \bar{\tilde{\omega}}_{ib,y}^{b,-} & 0 \end{bmatrix} \frac{1}{2\vec{\omega}} \tag{3.6}$$

The IMU is rotated at a known rate around the one axis, and the measurements on the other axes are averaged to determine the misalignment angles. Like the accelerometer, the arctangent of each term is taken to determine the misalignment in radians, rather than as a ratio.

4. REVIEW OF RESULTS

Using the rate table in the space systems lab at Embry-Riddle Aeronautical University, data was collected to calibrate the gyroscope and accelerometer in the VN-200. The accelerometer measurements from the 6-point tumble test are plotted in Appendix A: Accelerometer Calibration Data. The gyroscope measurements from the 6-point tumble test and the known rate rotation test are plotted in Appendix B: Gyroscope Calibration Data. The MATLAB file used to calculated all of the deterministic errors from the IMU data is included in Appendix E: main_calc_plot.m

4.i. Calibration Calculations

4.i.a. Accelerometer Calibration

The averaged accelerometer data from the 6-point tumble test is presented below in Table 1: 6-Point Tumble Test Accelerometer Average Data. The value calculated for local gravity was $g_l = 9.77561 \text{ m/s}^2$ based on the geodetic coordinates of the space systems lab. The data was used to calculate each of the twelve calibration variables below using the equations presented in Section 3.i. The fixed bias data for the accelerometer was converted to units of mg using the standard gravity constant $g = 9.80665 \text{ m/s}^2$.

Table 1: 6-Point Tumble Test Accelerometer Average Data

	Specific Force (m/s^2) (aligned with positive gravity)			Specific Force (m/s^2) (aligned with negative gravity)		
	x-aligned	y-aligned	z-aligned	x-aligned	y-aligned	z-aligned
x-sense	9.7785	0.0651	-0.0258	-9.8030	-0.0699	-0.0005
y-sense	-0.0897	9.7894	0.0187	0.0895	-9.7938	0.0070
z-sense	0.0460	0.0786	9.8431	0.0547	0.1001	-9.7123

$$\hat{\vec{b}}_{a,FB} = \begin{bmatrix} -1.2471 \\ -0.2228 \\ 6.6710 \end{bmatrix} \text{mg}$$

The scale factor error is presented below in parts per million. A linear extrapolation of the scale factor error is plotted in Figure 3: Accelerometer Scale Factor. Across the range of the sensor, the maximum error due to scale factor error is a quarter of a meter per second squared.

$$\hat{s}_a = \begin{bmatrix} 1,551 & 0 & 0 \\ 0 & 1,633 & 0 \\ 0 & 0 & 214 \end{bmatrix} \text{ppm}$$

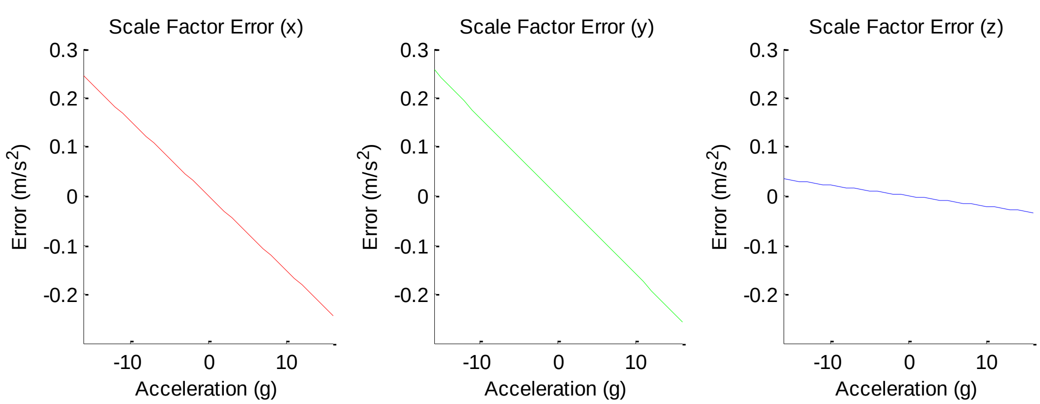

Figure 3: Accelerometer Scale Factor

The misalignment values required one more calculation beyond that given in Equation (3.3), because the misalignment matrix should be a skew symmetric matrix. The values for the alignment of axis A to axis B should be the negative of the values for the alignment of axis B to A. This property was ensured in the calculation by averaging the measurement for the crosscoupled axes. A plot of the estimated axes (RGB) versus the body frame (black) is shown in Figure 4: Accelerometer Misalignment.

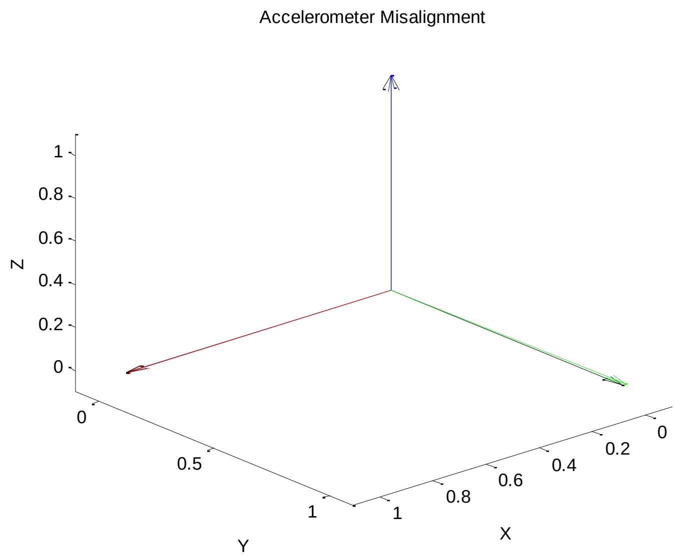

Figure 4: Accelerometer Misalignment

According to the VectorNav datasheet, the accelerometer should have a scale factor error between ±0.05° [5]. As seen below, the calculated misalignment was slightly greater than the range listed in the VN-200 specification. This could be due to incorrect assumptions and methods used for ensuring the misalignment matrix was skew symmetric.

$$\hat{m}_a = \begin{bmatrix} 0 & 0.4604 & -0.0243 \\ -0.4604 & 0 & 0.0485 \\ 0.0243 & -0.0485 & 0 \end{bmatrix} \text{ degrees}$$

4.i.b. Gyroscope Calibration

The data to calibrate the fixed bias on the gyroscope was taken from the 6-point tumble test.

$$\hat{\vec{b}}_{g,FB} = \begin{bmatrix} 1{,}826 \\ 1{,}714 \\ 2{,}791 \end{bmatrix} \text{ degrees/hour}$$

For the known rate tests to calibrate the scale factor and misalignment errors, the VN-200 was rotated at ±30 degrees per second around the x- and y-axes, and at ±60 degrees around the z-axis. The different rates were chosen due to the different axes utilized on the rate table. The z-axis could be rotated on the first-axis rotational table, while the x- and y-axes had to be rotated using the second-axis yoke. The first axis put less stress on the data-collection cable, so the VN-200 could be rotated at a higher rate safely.

The scale factor error is presented below in parts per million. A linear extrapolation of the scale factor error is plotted in Figure 5: Gyroscope Scale Factor. Across the range of the sensor, the maximum error due to scale factor error is three degrees per second

$$\hat{s}_a = \begin{bmatrix} -1,691 & 0 & 0 \\ 0 & -979 & 0 \\ 0 & 0 & 227 \end{bmatrix} \text{ppm}$$

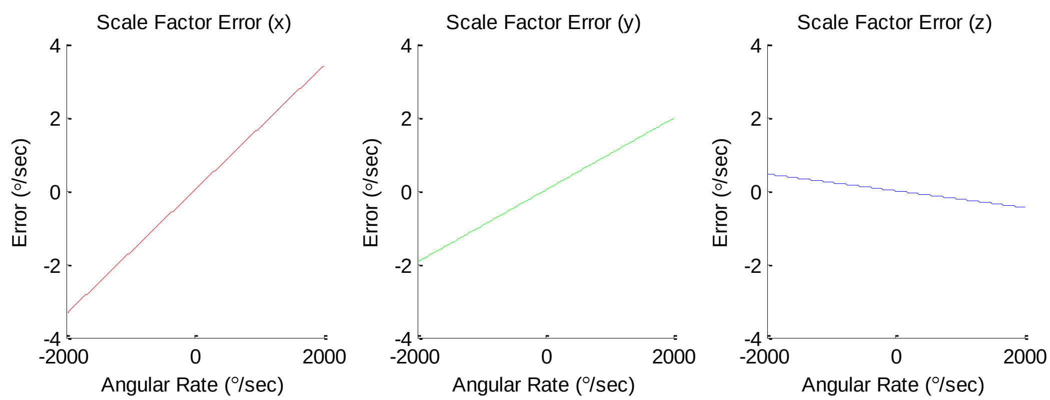

Figure 5: Gyroscope Scale Factor

The misalignment for the gyroscope was calculated the same way it was calculated for the accelerometer. A plot of the estimated axes (RGB) versus the body frame (black) is shown in Figure 6: Gyroscope Misalignment. According to the VectorNav datasheet, the gyroscope should have a misalignment error between ±0.05˚ [5]. Once again, the calculated misalignment was slightly greater than the range listed in the VN-200 specification, especially in the x-y coupling. This consistent result between the gyroscope and accelerometer could indicate a flaw in the alignment of the test setup.

$$\hat{m}_a = \begin{bmatrix} 0 & 0.4579 & 0.0374 \\ -0.4579 & 0 & 0.0581 \\ -0.0374 & -0.0581 & 0 \end{bmatrix} \text{degrees}$$

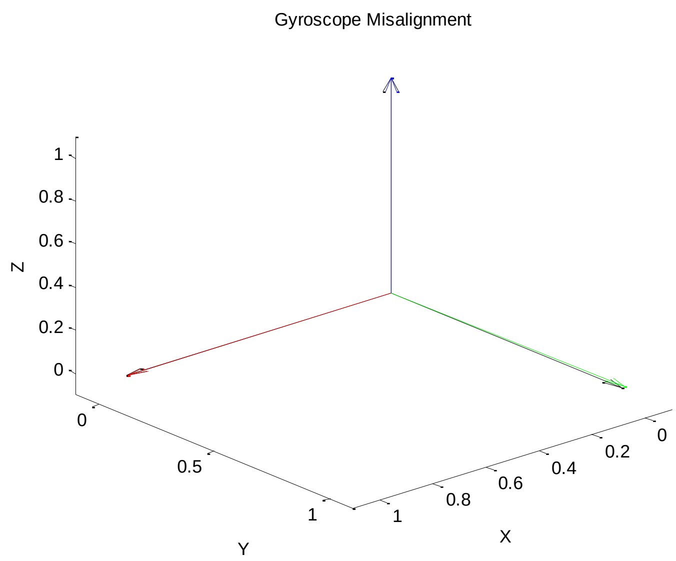

Figure 6: Gyroscope Misalignment

For both the accelerometer and the gyroscope, the scale factor and misalignment errors are relatively small compared to the fixed bias errors. This result indicates that the biggest increase in performance will come from calibrating out the fixed bias, not the misalignment and scale factor.

4.ii. Comparing Calibrated and Non-Calibrated Performance

Two test profiles were created using the rate table to collect data from the VN-200 over a known course for 20 seconds. The first oriented the VN-200 at a known position, and held it at a fixed position and orientation. The second was held the VN-200 in the same position as the first test, but rotating it at 30 degrees per second around its z-axis. These tests were used to create independent data sets to test the VN-200 calibration with. The data collected from these two test is plotted in Appendix C: Known Stimulus Test Data. The collected data was then used to drive an inertial navigation mechanization with different levels of calibration. With the known path, the data from the mechanization could be compared to the values the IMU measurements should have produced. The MATLAB script in Appendix F: proj_final.m was modified from EE495 Project 1 to create the new test profiles and plot the measured data versus truth with different levels of calibration.

The results from the first test profile are listed in Table 2: Errors after 20 Seconds (Fixed Position/Fixed Attitude). As seen in the table, the error with complete calibration of deterministic errors had the smallest errors after 20 seconds. It can also be seen that the fixed bias calibration had the largest effect on correcting the measurements from the IMU. The scale factor and misalignment matrices had little effect on the overall error in the mechanization. In the complete mechanization, they improved the error, but without removing the fixed bias, they actually made the error worse.

Table 2: Errors after 20 Seconds (Fixed Position/Fixed Attitude)

	Position Error (m)			Velocity Error (m/s)			Attitude Error (˚)		
Calibration Type	X	Y	Z	X	Y	Z	X	Y	Z
None	49	-92	-125	7	-13	-19	-8	-17	-10
Fixed Bias	-1.9	2.3	-8	-0.17	0.21	-0.87	0.65	0.33	0.09
M Matrix	49	-93	-127	7	-14	-19	-7.5	-17	10
Complete	-1.2	0.8	-9	-0.10	0.06	-0.95	0.65	0.32	0.09

The results from the second test profile are listed in Table 3: Errors after 20 Seconds (Fixed Position/Rotating Attitude). These results are similar to the results observed in the first test profile. Like the first test, the fixed bias had the most significant role in improving the errors in the mechanization and the scale factor and misalignment calibration were only positive when the fixed bias had been removed.

Table 3: Errors after 20 Seconds (Fixed Position/Rotating Attitude)

	Position Error (m)			Velocity Error (m/s)			Attitude Error (˚)		
Calibration Type	X	Y	Z	X	Y	Z	X	Y	Z
None	2	-31	42	0.51	-3.5	4	-13	-9	1.4
Fixed Bias	0.34	-1.8	4.9	0.03	-0.16	0.39	0.28	0.05	-0.62
M Matrix	0.8	-30	40	0.41	-3.4	4	-9.2	-12.9	1.3
Complete	-0.85	-0.83	3.6	-0.10	-0.05	0.24	0.40	0.14	-0.73

The next two sections show the error plots for each of the two trial runs before and after calibration. For both trial runs it can clearly be seen that the full calibration aided the IMU mechanization immensely.

4.ii.a. Plotted errors for fixed position, fixed orientation test

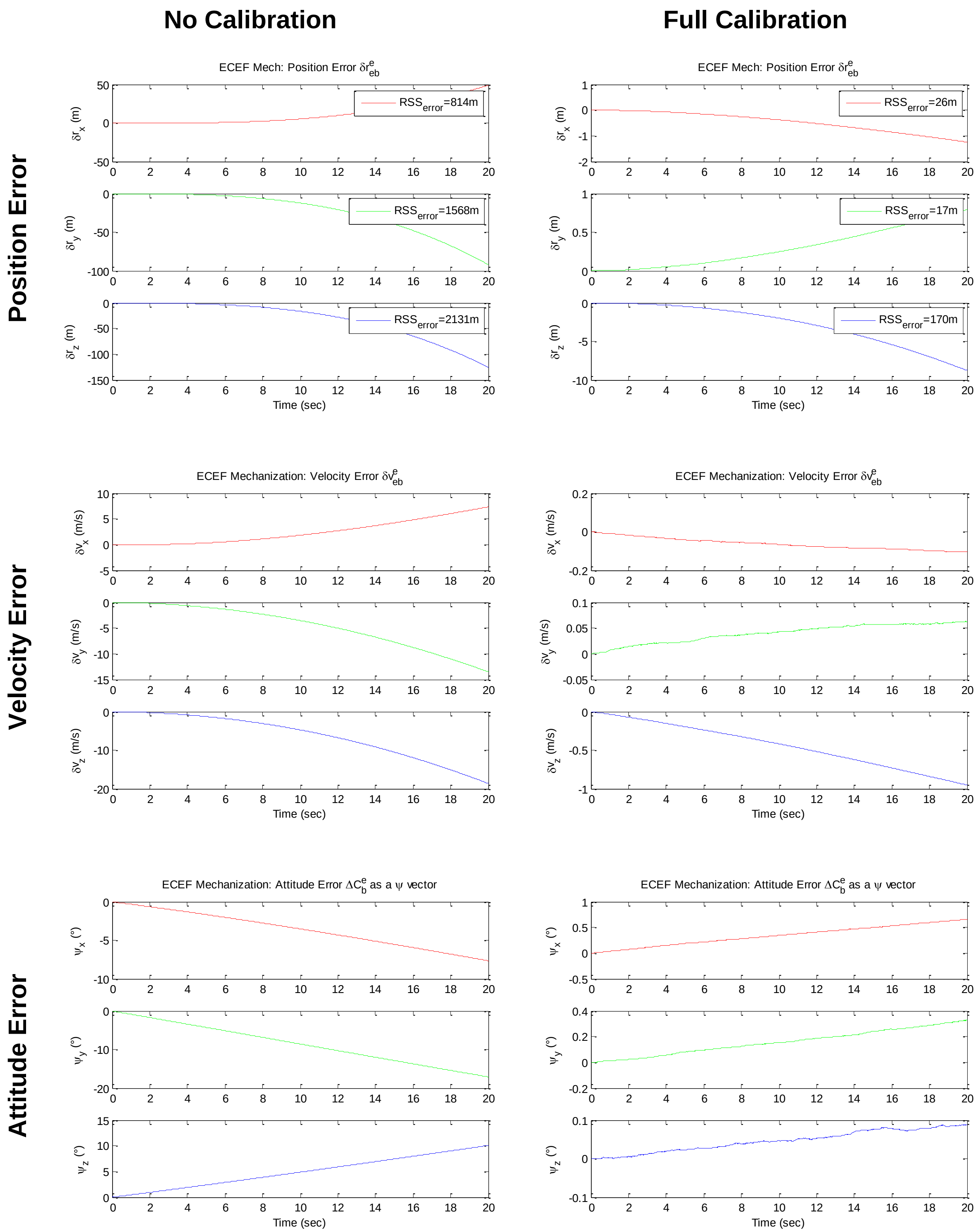

Figure 7: Plotted errors for fixed position, fixed orientation test

4.ii.b. Plotted errors, for fixed position, known rotation test

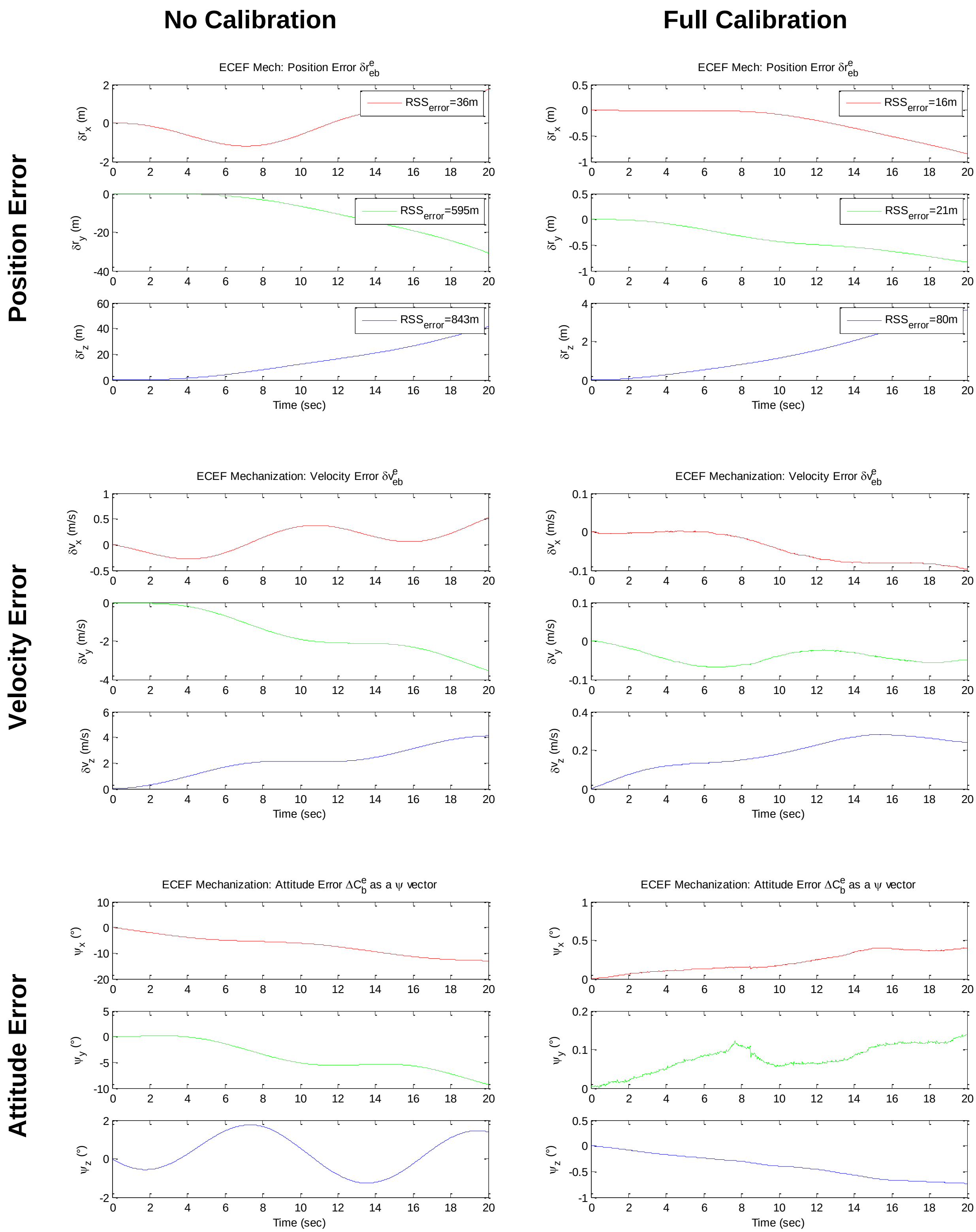

Figure 8: Plotted errors, for fixed position, known rotation test

5. CONCLUSION/SUMMARY

Calibration of the VectorNav VN-200 produced massive increases in performance over the short 20 second test runs of the IMU. The position, velocity, and attitude improvements for each of the tests are listed in Table 4: Improvements in PVA after Calibration.

Table 4: Improvements in PVA after Calibration

	Fixed Position, Fixed Attitude			Fixed Position, Rotating Attitude		
	Position Error (m)	Velocity Error (m)	Attitude Error (˚)	Position Error (m)	Velocity Error (m)	Attitude Error (˚)
Before Calibration	162.8 m	24.1 m/s	21.3˚	52.2 m	5.34 m/s	15.9˚
After Calibration	9.11 m	0.96 m/s	0.73˚	3.79 m	0.26 m/s	0.84˚
Improvement	94.4%	96.0%	96.6%	92.7%	95.0%	94.7%

The greatest improvement in the errors calculated by the mechanization came from removing the fixed biases in the sensors. Removing the scale-factor and misalignment terms did little to improve the accuracy of the measurements. This could be due to several factors. First, the calculated scale-factor and misalignment terms were relatively insignificant compared to the errors in fixed bias. The axes were fairly well aligned and scaled to begin with. Also, the two sample profiles did not test many of the situations where the scale-factor and misalignment would have a large effect. If the VN-200 were subjected to greater, and more varying, forces and angular rates than it was, the terms in the M matrix might have played a large role in improving the IMU measurements.

The improvements due to calibration are not quite as good as the 95-98% improvements advertised on VectorNav’s calibration page [4], but they are in a very similar range. Further improvements in the calibration could be achieved by performing the measurements at different regulated temperatures and over shorter periods of time. In the twenty minutes that were spent collecting all the data, random errors could have found their way into the IMU measurements. The calibration, however, did produce satisfactory results in decreasing the IMU measurement errors.

6. APPENDIX A: ACCELEROMETER CALIBRATION DATA

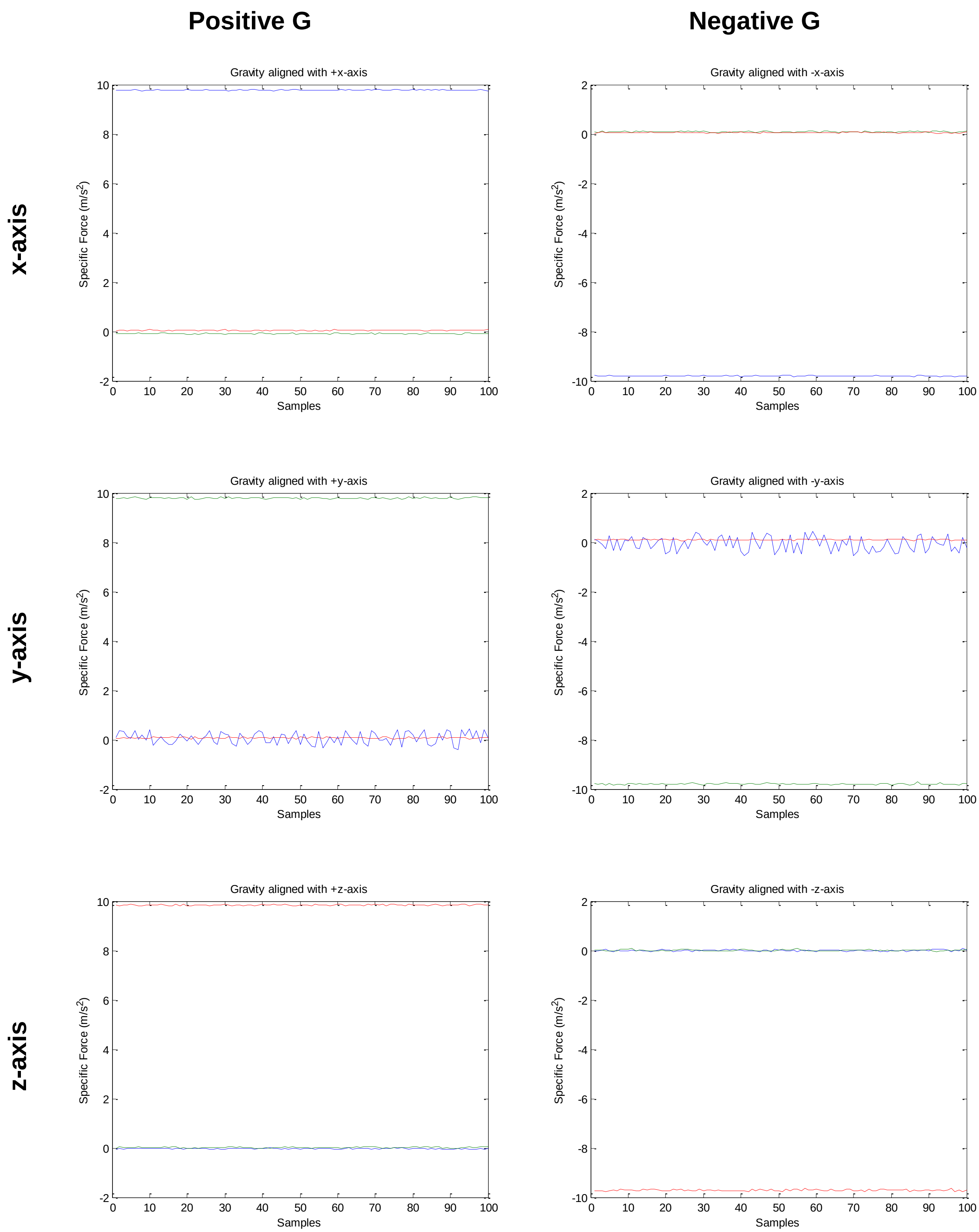

Figure 9: Accelerometer Calibration Data

7. APPENDIX B: GYROSCOPE CALIBRATION DATA

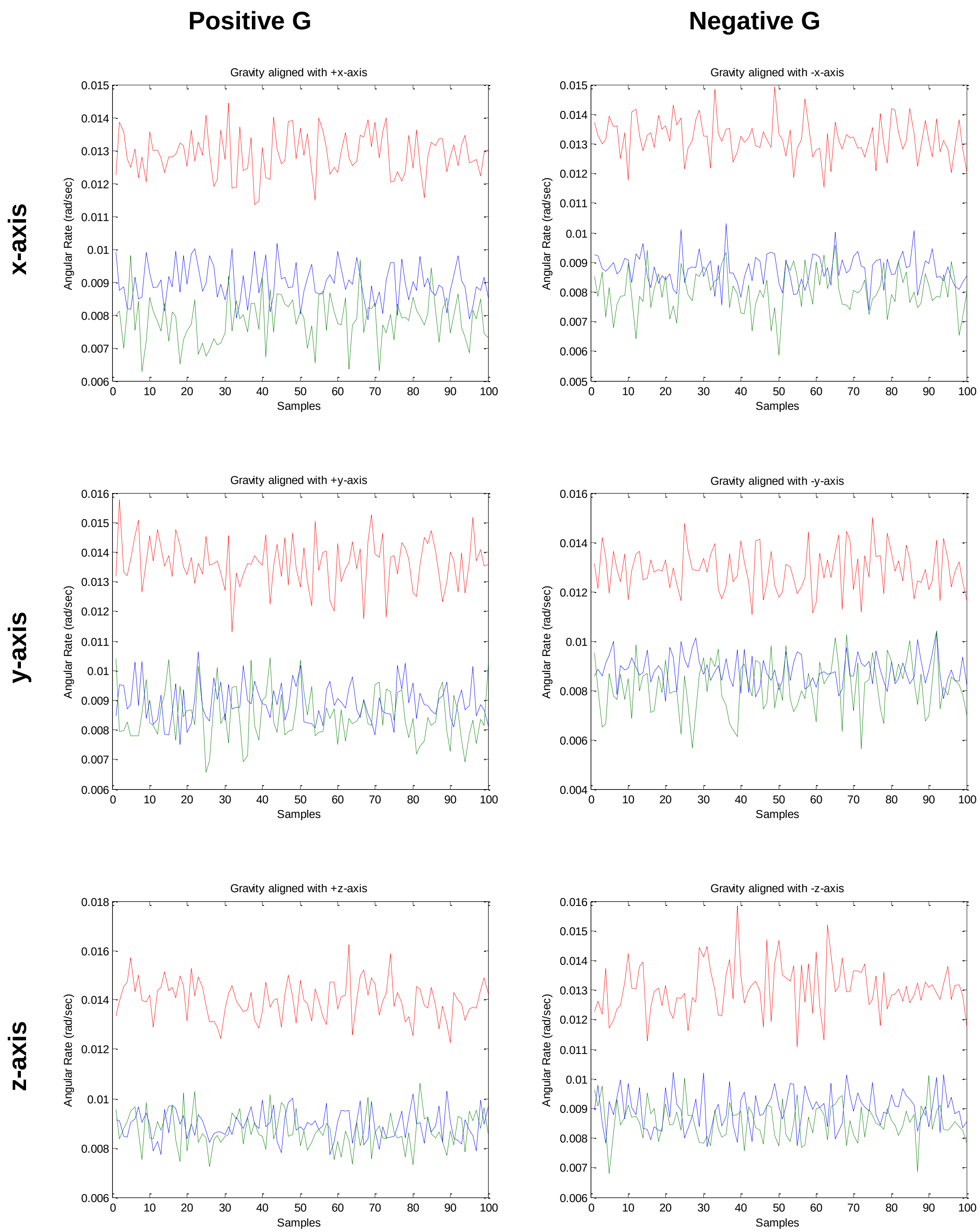

Figure 10: Gyro Fixed Bias Calibration Data

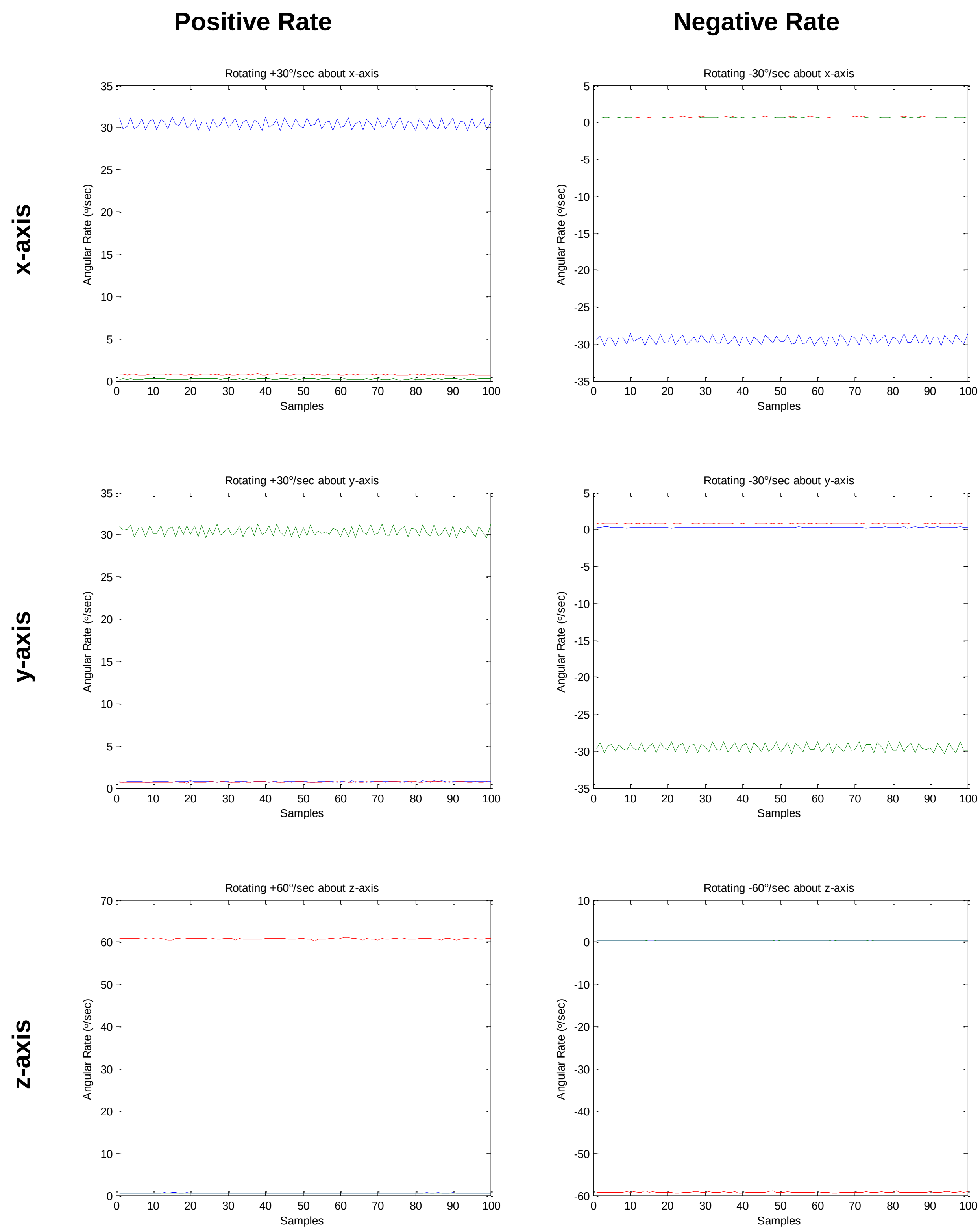

Figure 11: Gyro Scale Factor and Misalignment Calibration Data

8. APPENDIX C: KNOWN STIMULUS TEST DATA

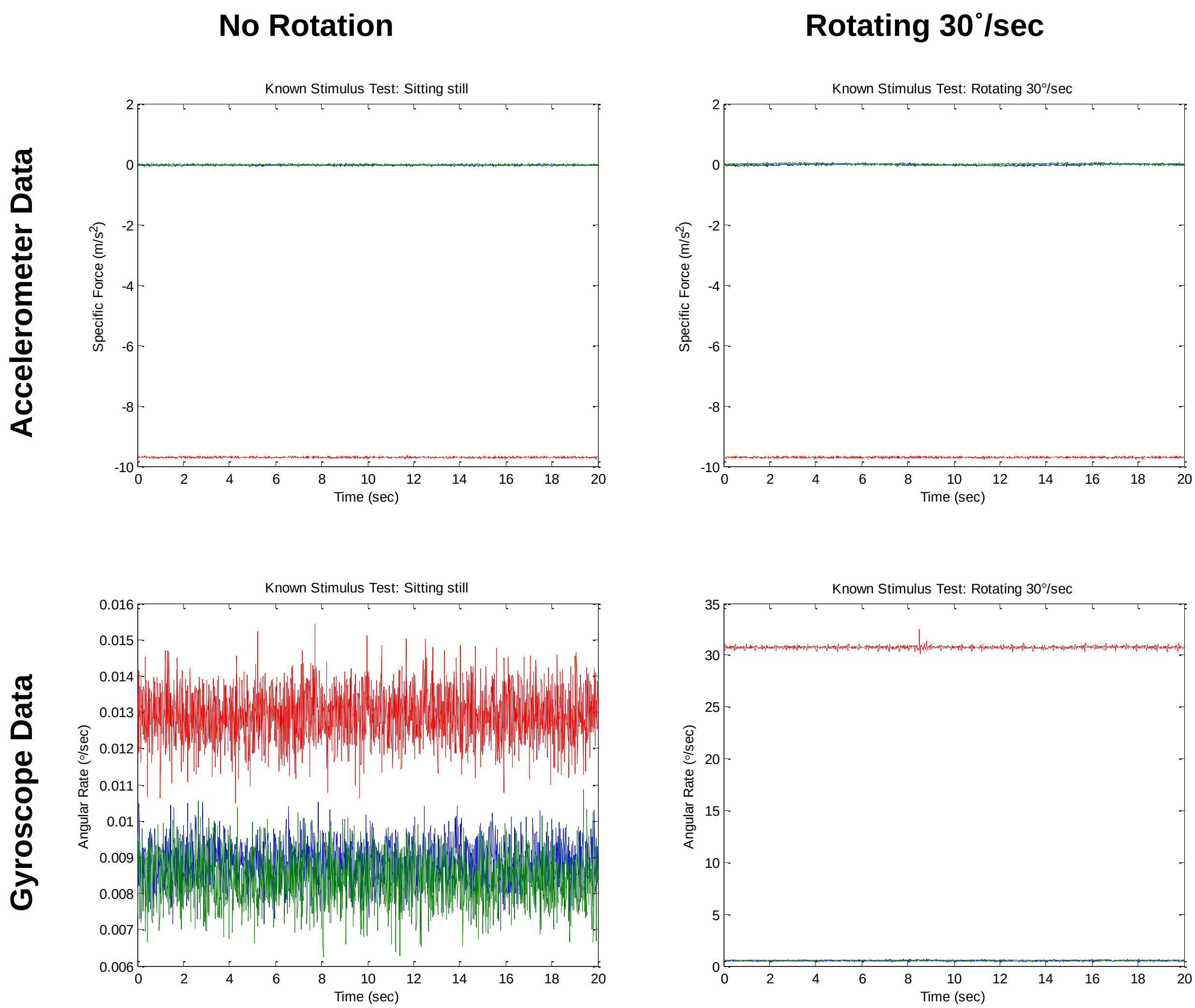

Figure 12: Known Stimulus Test Data

9. APPENDIX D: MAIN_DATA_COLLECTION.M

```
% Procesing the VN200 IMU data
clear all;          % Clear all variables from the workspace
close all;          % Close all windows
clc;                % "Clean" the command window

%% Configure the serial port
BaudRate = 115200;  % BaudRate: 115200, 128000, 230400,460800, 921600
s = VNserial('COM3', BaudRate); % Connect to the serial PORT COM?
N = 100; % Set number of samples

%% z-axis data
display('Collect data: z-axis aligned with positive G (space to continue)');
pause % Wait for user input to continue
d_z_pos_g = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_z_pos_g(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_z_pos_g.mat','d_z_pos_g');
fprintf('  Data written to ''d_z_pos_g.mat''\n\n');

display('Collect data: z-axis aligned with negative G (space to continue)');
pause % Wait for user input to continue
d_z_neg_g = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_z_neg_g(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_z_neg_g.mat','d_z_neg_g');
fprintf('  Data written to ''d_z_neg_g.mat''\n\n');

display('Collect data: rotating positively about z-axis (space to continue)');
pause % Wait for user input to continue
d_rot_pos_z = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_rot_pos_z(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_rot_pos_z.mat','d_rot_pos_z');
fprintf('  Data written to ''d_rot_pos_z.mat''\n\n');

display('Collect data: rotating negatively about z-axis (space to continue)');
pause % Wait for user input to continue
d_rot_neg_z = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_rot_neg_z(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_rot_neg_z.mat','d_rot_neg_z');
fprintf('  Data written to ''d_rot_neg_z.mat''\n\n');

%% x-axis data
display('Collect data: x-axis aligned with positive G (space to continue)');
pause % Wait for user input to continue
d_x_pos_g = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_x_pos_g(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
```

```
end
save('d_x_pos_g.mat','d_x_pos_g');
fprintf('  Data written to ''d_x_pos_g.mat''\n\n');

display('Collect data: x-axis aligned with negative G (space to continue)');
pause % Wait for user input to continue
d_x_neg_g = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_x_neg_g(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_x_neg_g.mat','d_x_neg_g');
fprintf('  Data written to ''d_x_neg_g.mat''\n\n');

display('Collect data: rotating positively about x-axis (space to continue)');
pause % Wait for user input to continue
d_rot_pos_x = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_rot_pos_x(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_rot_pos_x.mat','d_rot_pos_x');
fprintf('  Data written to ''d_rot_pos_x.mat''\n\n');

display('Collect data: rotating negatively about x-axis (space to continue)');
pause % Wait for user input to continue
d_rot_neg_x = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_rot_neg_x(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_rot_neg_x.mat','d_rot_neg_x');
fprintf('  Data written to ''d_rot_neg_x.mat''\n\n');

%% y-axis data
display('Collect data: y-axis aligned with positive G (space to continue)');
pause % Wait for user input to continue
d_y_pos_g = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_y_pos_g(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_y_pos_g.mat','d_y_pos_g');
fprintf('  Data written to ''d_y_pos_g.mat''\n\n');

display('Collect data: y-axis aligned with negative G (space to continue)');
pause % Wait for user input to continue
d_y_neg_g = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_y_neg_g(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_y_neg_g.mat','d_y_neg_g');
fprintf('  Data written to ''d_y_neg_g.mat''\n\n');

display('Collect data: rotating positively about y-axis (space to continue)');
pause % Wait for user input to continue
d_rot_pos_y = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_rot_pos_y(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
```

```
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_rot_pos_y.mat','d_rot_pos_y');
fprintf('  Data written to ''d_rot_pos_y.mat''\n\n');

display('Collect data: rotating negatively about y-axis (space to continue)');
pause % Wait for user input to continue
d_rot_neg_y = zeros(N,11); % Initialize space for the data-set
for i=1:N % Collect N number of samples
    d_rot_neg_y(i,:) = VNreadregister(s, 'IMU'); % Read the IMU data
    if mod(i,25) == 0
        disp(['    Samples = ', num2str(i)]);
    end
end
save('d_rot_neg_y.mat','d_rot_neg_y');
fprintf('  Data written to ''d_rot_neg_y.mat''\n\n');

Fs = 100;            % Default Sample frequency
dT = 1/Fs;           % Sample interval
nSamples = 2000;     % Number of samples to collect
numSec = nSamples/Fs; % Number of secondsd of data collection

display('Collect data: time dependant, sitting still (space to continue)');
pause % Wait for user input to continue
d_t_still = VNrecordADOR(s, 'IMU', Fs, numSec);
save('d_t_still.mat','d_t_still');
fprintf('  Data written to ''d_t_still.mat''\n\n');

display('Collect data: time dependant, rotating (space to continue)');
pause % Wait for user input to continue
d_t_rot = VNrecordADOR(s, 'IMU', Fs, numSec);
save('d_t_rot.mat','d_t_rot');
fprintf('  Data written to ''d_t_rot''\n\n');
```

10. APPENDIX E: MAIN_CALC_PLOT.M

```
% Procesing the VN200 IMU data for calibration

clear all;          % Clear all variables from the workspace
close all;          % Close all windows
clc;                % "Clean" the command window

%% Define variables
local_g = gravity(34.61453654,1582); % Calculate local gravity (m/s^2)
g = 9.80665; % Standard gravity constant (m/s^2)
fixed_rate_30 = 30*pi/180; % Rate table rate in (rad/sec)
fixed_rate_60 = 60*pi/180; % Rate table rate in (rad/sec)

%% Process starting data
% Import captured IMU data
load('data/d_rot_neg_x.mat')
load('data/d_rot_neg_y.mat')
load('data/d_rot_neg_z.mat')
load('data/d_rot_pos_x.mat')
load('data/d_rot_pos_y.mat')
load('data/d_rot_pos_z.mat')
load('data/d_x_neg_g.mat')
load('data/d_x_pos_g.mat')
load('data/d_y_neg_g.mat')
load('data/d_y_pos_g.mat')
load('data/d_z_neg_g.mat')
load('data/d_z_pos_g.mat')
load('data/d_t_still.mat')
load('data/d_t_rot.mat')

% Find mean values for the data columns (filter noise)
rot_neg_x = mean(d_rot_neg_x);
rot_neg_y = mean(d_rot_neg_y);
rot_neg_z = mean(d_rot_neg_z);
rot_pos_x = mean(d_rot_pos_x);
rot_pos_y = mean(d_rot_pos_y);
rot_pos_z = mean(d_rot_pos_z);
x_neg_g = mean(d_x_neg_g);
x_pos_g = mean(d_x_pos_g);
y_neg_g = mean(d_y_neg_g);
y_pos_g = mean(d_y_pos_g);
z_neg_g = mean(d_z_neg_g);
z_pos_g = mean(d_z_pos_g);

%% Accelerometer Fixed Bias
accel_b_cf = 1000 * 1/g; % Converstion factor from m/s^2 to mg

% Calculate accelerometer bias (mg)
accel_b_x = 1/2 * (x_pos_g(:,4) + x_neg_g(:,4)) * accel_b_cf;
accel_b_y = 1/2 * (y_pos_g(:,5) + y_neg_g(:,5)) * accel_b_cf;
accel_b_z = 1/2 * (z_pos_g(:,6) + z_neg_g(:,6)) * accel_b_cf;

% Display results
display(accel_b_x);
display(accel_b_y);
display(accel_b_z);

%% Accelerometer Scale Factor
accel_sf_cf = 1e6; % Convertion to ppm

% Calculate scale percentage
accel_sp_x = (x_pos_g(:,4) - x_neg_g(:,4)) / (2*local_g);
accel_sp_y = (y_pos_g(:,5) - y_neg_g(:,5)) / (2*local_g);
accel_sp_z = (z_pos_g(:,6) - z_neg_g(:,6)) / (2*local_g);

% Calculate scale factor (ppm)
accel_sf_x = (accel_sp_x-1) * accel_sf_cf;
accel_sf_y = (accel_sp_y-1) * accel_sf_cf;
accel_sf_z = (accel_sp_z-1) * accel_sf_cf;
```

```
% Display results
display(accel_sf_x);
display(accel_sf_y);
display(accel_sf_z);

% Plot Results
asfe = figure;
set(asfe,'Position',[10 200 800 250]);
xlim_asfe = [-16 16];
xlabel_asfe = 'Acceleration (g)';
ylim_asfe = [-0.3 0.3];
ylabel_asfe = 'Error (m/s^2)';
stdg = 9.80665;
x = xlim_asfe(1):1:xlim_asfe(2);
y = x*stdg;
ys_x = y*accel_sp_x;
ys_y = y*accel_sp_y;
ys_z = y*accel_sp_z;

subplot(1,3,1);
hold on;
plot(x,y-ys_x,'r');
xlim(xlim_asfe);
xlabel(xlabel_asfe);
ylim(ylim_asfe);
ylabel(ylabel_asfe);
title('Scale Factor Error (x)');

subplot(1,3,2);
hold on;
plot(x,y-ys_y,'g');
xlim(xlim_asfe);
xlabel(xlabel_asfe);
ylim(ylim_asfe);
ylabel(ylabel_asfe);
title('Scale Factor Error (y)');

subplot(1,3,3);
hold on;
plot(x,y-ys_z,'b');
xlim(xlim_asfe);
xlabel(xlabel_asfe);
ylim(ylim_asfe);
ylabel(ylabel_asfe);
title('Scale Factor Error (z)');

%% Accelerometer Misalignment
a_m_yx = atand((x_pos_g(:,5) - x_neg_g(:,5)) / (2*local_g));
a_m_zx = atand((x_pos_g(:,6) - x_neg_g(:,6)) / (2*local_g));
a_m_xy = atand((y_pos_g(:,4) - y_neg_g(:,4)) / (2*local_g));
a_m_zy = atand((y_pos_g(:,6) - y_neg_g(:,6)) / (2*local_g));
a_m_xz = atand((z_pos_g(:,4) - z_neg_g(:,4)) / (2*local_g));
a_m_yz = atand((z_pos_g(:,5) - z_neg_g(:,5)) / (2*local_g));
accel_m_yx = (a_m_yx - a_m_xy)/2;
accel_m_zx = (a_m_zx - a_m_xz)/2;
accel_m_xy = (a_m_xy - a_m_yx)/2;
accel_m_zy = (a_m_zy - a_m_yz)/2;
accel_m_xz = (a_m_xz - a_m_zx)/2;
accel_m_yz = (a_m_yz - a_m_zy)/2;

% Display results
display(accel_m_xy)
display(accel_m_xz)
display(accel_m_yx)
display(accel_m_yz)
display(accel_m_zx)
display(accel_m_zy)

% Plot figure
figure;
```

```
hold on;
title('Accelerometer Misalignment');

% Set limits and labels for the figure
xlim([-0.1,1.1]); ylim([-0.1,1.1]); zlim([-0.1,1.1]);
xlabel('X'); ylabel('Y'); zlabel('Z');

% Set the default orientation for the plot view
az = 139;
el = 30;
view(az, el);

% Plot the ideal and misaligned axes
quiver3(0,0,0,1,0,0,1,'color','k');
quiver3(0,0,0,0,1,0,1,'color','k');
quiver3(0,0,0,0,0,1,1,'color','k');
quiver3(0,0,0,cosd(accel_m_xy)*cosd(accel_m_xz),sind(accel_m_xy),sind(accel_m_xz),1,'color','r');
quiver3(0,0,0,sind(accel_m_yx),cosd(accel_m_yx)*cosd(accel_m_yz),sind(accel_m_yz),1,'color','g');
quiver3(0,0,0,sind(accel_m_zx),sind(accel_m_zy),cosd(accel_m_zx)*cosd(accel_m_zy),1,'color','b');

%% Gyroscope Fixed Bias
gyro_b_cf = 180/pi * 3600; % Converstion factor from rad/sec  to deg/hour

% Calculate accelerometer bias (mg)
gyro_b_x = 1/2 .* (x_pos_g(:,7) + x_neg_g(:,7)) .* gyro_b_cf;
gyro_b_y = 1/2 .* (y_pos_g(:,8) + y_neg_g(:,8)) .* gyro_b_cf;
gyro_b_z = 1/2 .* (z_pos_g(:,9) + z_neg_g(:,9)) .* gyro_b_cf;

% Display results
display(gyro_b_x);
display(gyro_b_y);
display(gyro_b_z);

%% Gyroscope Scale Factor
gyro_sf_cf = 1e6; % Convertion to ppm

% Calculate scale percentage
gyro_sp_x = (rot_pos_x(:,7) - rot_neg_x(:,7)) / (2*fixed_rate_30);
gyro_sp_y = (rot_pos_y(:,8) - rot_neg_y(:,8)) / (2*fixed_rate_30);
gyro_sp_z = (rot_pos_z(:,9)- rot_neg_z(:,9)) / (2*fixed_rate_60);

% Calculate scale factor (ppm)
gyro_sf_x = (gyro_sp_x-1) * gyro_sf_cf;
gyro_sf_y = (gyro_sp_y-1) * gyro_sf_cf;
gyro_sf_z = (gyro_sp_z-1) * gyro_sf_cf;

% Display results
display(gyro_sf_x);
display(gyro_sf_y);
display(gyro_sf_z);

% Plot Results
gsfe = figure;
set(gsfe,'Position',[10 200 800 250]);
xlim_gsfe = [-2000 2000];
xlabel_gsfe = 'Angular Rate (\circ/sec)';
ylim_gsfe = [-4 4];
ylabel_gsfe = 'Error (\circ/sec)';
xg = xlim_gsfe(1):1:xlim_gsfe(2);
ygs_x = xg*gyro_sp_x;
ygs_y = xg*gyro_sp_y;
ygs_z = xg*gyro_sp_z;

subplot(1,3,1);
hold on;
plot(xg,xg-ygs_x,'r');
xlim(xlim_gsfe);
xlabel(xlabel_gsfe);
ylim(ylim_gsfe);
ylabel(ylabel_gsfe);
title('Scale Factor Error (x)');
```

```
subplot(1,3,2);
hold on;
plot(xg,xg-ygs_y,'g');
xlim(xlim_gsfe);
xlabel(xlabel_gsfe);
ylim(ylim_gsfe);
ylabel(ylabel_gsfe);
title('Scale Factor Error (y)');

subplot(1,3,3);
hold on;
plot(xg,xg-ygs_z,'b');
xlim(xlim_gsfe);
xlabel(xlabel_gsfe);
ylim(ylim_gsfe);
ylabel(ylabel_gsfe);
title('Scale Factor Error (z)');

%% Gyroscope Misalignment
g_m_yx = atand((rot_pos_x(:,8) - rot_neg_x(:,8)) / (2*fixed_rate_30));
g_m_zx = atand((rot_pos_x(:,9) - rot_neg_x(:,9)) / (2*fixed_rate_30));
g_m_xy = atand((rot_pos_y(:,7) - rot_neg_y(:,7)) / (2*fixed_rate_30));
g_m_zy = atand((rot_pos_y(:,9) - rot_neg_y(:,9)) / (2*fixed_rate_30));
g_m_xz = atand((rot_pos_z(:,7) - rot_neg_z(:,7)) / (2*fixed_rate_60));
g_m_yz = atand((rot_pos_z(:,8) - rot_neg_z(:,8)) / (2*fixed_rate_60));
gyro_m_yx = (g_m_yx - g_m_xy)/2;
gyro_m_zx = (g_m_zx - g_m_xz)/2;
gyro_m_xy = (g_m_xy - g_m_yx)/2;
gyro_m_zy = (g_m_zy - g_m_yz)/2;
gyro_m_xz = (g_m_xz - g_m_zx)/2;
gyro_m_yz = (g_m_yz - g_m_zy)/2;

% Display results
display(gyro_m_xy)
display(gyro_m_xz)
display(gyro_m_yx)
display(gyro_m_yz)
display(gyro_m_zx)
display(gyro_m_zy)

% Plot figure
figure;
hold on;
title('Gyroscope Misalignment');

% Set limits and labels for the figure
xlim([-0.1,1.1]); ylim([-0.1,1.1]); zlim([-0.1,1.1]);
xlabel('X'); ylabel('Y'); zlabel('Z');

% Set the default orientation for the plot view
az = 139;
el = 30;
view(az, el);

% Plot the ideal and misaligned axes
quiver3(0,0,0,1,0,0,1,'color','k');
quiver3(0,0,0,0,1,0,1,'color','k');
quiver3(0,0,0,0,0,1,1,'color','k');
quiver3(0,0,0,cosd(gyro_m_xy)*cosd(gyro_m_xz),sind(gyro_m_xy),sind(gyro_m_xz),1,'color','r');
quiver3(0,0,0,sind(gyro_m_yx),cosd(gyro_m_yx)*cosd(gyro_m_yz),sind(gyro_m_yz),1,'color','g');
quiver3(0,0,0,sind(gyro_m_zx),sind(gyro_m_zy),cosd(gyro_m_zx)*cosd(gyro_m_zy),1,'color','b');

%% Collected Data Plots

% figure;
% plot(d_rot_neg_x(:,7:9).*180./pi)
% title('Rotating -30\circ/sec about x-axis');
% xlabel('Samples');
% ylabel('Angular Rate (\circ/sec)');
%
```

```
% figure;
% plot(d_rot_neg_y(:,7:9).*180./pi)
% title('Rotating -30\circ/sec about y-axis');
% xlabel('Samples');
% ylabel('Angular Rate (\circ/sec)');
%
% figure;
% plot(d_rot_neg_z(:,7:9).*180./pi)
% title('Rotating -60\circ/sec about z-axis');
% xlabel('Samples');
% ylabel('Angular Rate (\circ/sec)');
%
% figure;
% plot(d_rot_pos_x(:,7:9).*180./pi)
% title('Rotating +30\circ/sec about x-axis');
% xlabel('Samples');
% ylabel('Angular Rate (\circ/sec)');
%
% figure;
% plot(d_rot_pos_y(:,7:9).*180./pi)
% title('Rotating +30\circ/sec about y-axis');
% xlabel('Samples');
% ylabel('Angular Rate (\circ/sec)');
%
% figure;
% plot(d_rot_pos_z(:,7:9).*180./pi)
% title('Rotating +60\circ/sec about z-axis');
% xlabel('Samples');
% ylabel('Angular Rate (\circ/sec)');
%
% figure;
% plot(d_x_neg_g(:,4:6))
% title('dxnegg');
% title('Gravity aligned with -x-axis');
% xlabel('Samples');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(d_y_neg_g(:,4:6))
% title('Gravity aligned with -y-axis');
% xlabel('Samples');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(d_z_neg_g(:,4:6))
% title('Gravity aligned with -z-axis');
% xlabel('Samples');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(d_x_pos_g(:,4:6))
% title('Gravity aligned with +x-axis');
% xlabel('Samples');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(d_y_pos_g(:,4:6))
% title('Gravity aligned with +y-axis');
% xlabel('Samples');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(d_z_pos_g(:,4:6))
% title('Gravity aligned with +z-axis');
% xlabel('Samples');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(d_x_neg_g(:,7:9))
% title('dxnegg');
% title('Gravity aligned with -x-axis');
```

```
% xlabel('Samples');
% ylabel('Angular Rate (rad/sec)');
%
% figure;
% plot(d_y_neg_g(:,7:9))
% title('Gravity aligned with -y-axis');
% xlabel('Samples');
% ylabel('Angular Rate (rad/sec)');
%
% figure;
% plot(d_z_neg_g(:,7:9))
% title('Gravity aligned with -z-axis');
% xlabel('Samples');
% ylabel('Angular Rate (rad/sec)');
%
% figure;
% plot(d_x_pos_g(:,7:9))
% title('Gravity aligned with +x-axis');
% xlabel('Samples');
% ylabel('Angular Rate (rad/sec)');
%
% figure;
% plot(d_y_pos_g(:,7:9))
% title('Gravity aligned with +y-axis');
% xlabel('Samples');
% ylabel('Angular Rate (rad/sec)');
%
% figure;
% plot(d_z_pos_g(:,7:9))
% title('Gravity aligned with +z-axis');
% xlabel('Samples');
% ylabel('Angular Rate (rad/sec)');
%
% figure;
% plot(0.01:0.01:20,d_t_still(:,4:6))
% title('Known Stimulus Test: Sitting still');
% xlabel('Time (sec)');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(0.01:0.01:20,d_t_still(:,7:9))
% title('Known Stimulus Test: Sitting still');
% xlabel('Time (sec)');
% ylabel('Angular Rate (\circ/sec)');
%
% figure;
% plot(0.01:0.01:20,d_t_rot(:,4:6))
% title('Known Stimulus Test: Rotating 30\circ/sec');
% xlabel('Time (sec)');
% ylabel('Specific Force (m/s^2)');
%
% figure;
% plot(0.01:0.01:20,d_t_rot(:,7:9).*180./pi)
% title('Known Stimulus Test: Rotating 30\circ/sec');
% xlabel('Time (sec)');
% ylabel('Angular Rate (\circ/sec)');

%% Complile Calibration Variables

% Accelerometer fixed bias
b_a_FB = [accel_b_x * g/1000   % Bias - Fixed Bias x-axis term (m/s^2)
          accel_b_y * g/1000   % Bias - Fixed Bias y-axis term
          accel_b_z * g/1000]; % Bias - Fixed Bias z-axis term

% Accelerometer scale factor error & misalignment terms
s_a_x = accel_sf_x * 1e-6;     % x-axis scale factor error (ppm * 1e-6)
s_a_y = accel_sf_y * 1e-6;     % y-axis scale factor error
s_a_z = accel_sf_z * 1e-6;     % z-axis scale factor error
m_a_xy = accel_m_xy * pi/180;  % Misalignment of y-axis into x-axis (in rad)
m_a_xz = accel_m_xz * pi/180;  % Misalignment of z-axis into x-axis
m_a_yx = accel_m_yx * pi/180;  % Misalignment of x-axis into y-axis
```

```
m_a_yz = accel_m_yz * pi/180;  % Misalignment of z-axis into y-axis
m_a_zx = accel_m_zx * pi/180;  % Misalignment of x-axis into z-axis
m_a_zy = accel_m_zy * pi/180;  % Misalignment of y-axis into z-axis
M_a = [s_a_x  m_a_xy m_a_xz    % The combined Misalignment / Scale Factor matrix (dimensionless)
       m_a_yx s_a_y  m_a_yz
       m_a_zx m_a_zy s_a_z ];

% Gyroscope fixed bias
b_g_FB = [gyro_b_x / gyro_b_cf   % Bias - Fixed Bias x-axis term (m/s^2)
          gyro_b_y / gyro_b_cf   % Bias - Fixed Bias y-axis term
          gyro_b_z / gyro_b_cf]; % Bias - Fixed Bias z-axis term

% Gyroscope scale factor error & misalignment terms
s_g_x = gyro_sf_x * 1e-6;     % x-axis scale factor error (ppm * 1e-6)
s_g_y = gyro_sf_y * 1e-6;     % y-axis scale factor error
s_g_z = gyro_sf_z * 1e-6;     % z-axis scale factor error
m_g_xy = gyro_m_xy * pi/180;  % Misalignment of y-axis into x-axis (in rad)
m_g_xz = gyro_m_xz * pi/180;  % Misalignment of z-axis into x-axis
m_g_yx = gyro_m_yx * pi/180;  % Misalignment of x-axis into y-axis
m_g_yz = gyro_m_yz * pi/180;  % Misalignment of z-axis into y-axis
m_g_zx = gyro_m_zx * pi/180;  % Misalignment of x-axis into z-axis
m_g_zy = gyro_m_zy * pi/180;  % Misalignment of y-axis into z-axis
M_g = [s_g_x  m_g_xy m_g_xz   % The combined Misalignment / Scale Factor matrix (dimensionless)
       m_g_yx s_g_y  m_g_yz
       m_g_zx m_g_zy s_g_z ];

save('calibration.mat','b_a_FB','M_a','b_g_FB','M_g')
```

11. APPENDIX F: PROJ_FINAL.M

```
% Generating true PVA vs PVA derived from VN-200 data
%
% Two test cases
%   Fixed position, fixed orientation
%   Fixed position, rotatating 30 deg/sec about z-axis
%
% Four calibration options
%   No calibration
%   Fixed bias removed
%   Misalignment/Scale-factor calibrated
%   Full calibration
%

clear all   % Remove all items from workspace - release system memory
close all   % Close all open figures (i.e. plots)
clc         % Clear the command window

fidelity = 0;                        % Set to 0/1 for low/high fidelity mechanization eqns
global constants;                    % Global variables
load_constants;                      % Load a file of constants

Fs = constants.Fs;                   % Sample frequency (Hz)
dt = constants.dt;                   % Sample interval (sec)
constants.plot_title = 'Mech Eqn';   % A title used in all plots
R0 = constants.R0;                   % Earth's equatorial radius (meters)
e = constants.e;                     % Eccentricity
N = 2000;
constants.N = N;                     % Number of time steps in the simulation
t_sec = 0.01:0.01:20;

%***************************************************************************
% ******   Generate the ground truth data   ******************************
%***************************************************************************
%---------------------------------------------------------------------------
%% ECI to ECEF Coordinate Frame
%---------------------------------------------------------------------------
% Generate Angular Velocity
w_ie = constants.w_ie;               % WGS84 Earth rate (rad/s)
w_i__i_e = [0; 0; w_ie];             % Angular velocity of {e} wrt {i} resolved in {i} (rad/s)
Ohm_i__i_e = [ 0    , -w_ie, 0; ...  % Skew symmetric version of w_i__i_e (rad/s)
    w_ie,  0    , 0; ...
    0    ,  0    , 0];
constants.Ohm_i__i_e = Ohm_i__i_e;  % will need later (rad/s)

% Generate Orientation (i.e. Attitude)
C_i__e = zeros(3,3,N);               % Initialize to zero
theta_GMST = 0;                      % Set theta GMST to be zero
theta_e = w_ie * t_sec + theta_GMST;% Rotational angle theta_e = w_ie t + theta_GMST
for i=1:N
    C_i__e(:,:,i) = [ cos(theta_e(i)), -sin(theta_e(i)), 0;     % Orientation of {e} wrt {i}
        sin(theta_e(i)), cos(theta_e(i)) , 0;
        0               , 0               , 1];
end

% Generate PVA
r_i__i_e = [0; 0; 0];  % Position of the origin of {e} wrt origin of {i} resolved in {i} (m)
v_i__i_e = [0; 0; 0];  % Velocity of e-frame wrt i-frame (m/s)
a_i__i_e = [0; 0; 0];  % Acceleration of e-frame wrt i-frame (m/s^2)

%---------------------------------------------------------------------------
%% ECEF to Navigation Coordinate Frame
%---------------------------------------------------------------------------
% The curvlinear coordinates of the origin of the fixed frame
L_f       = 34.61453654*ones(1,N); % Geodetic Latitude  of the fixed frame (rad)
lambda_f  = -112.4502933*ones(1,N); % Geodetic Longitude of the fixed frame (rad)
h_f       = 1582*ones(1,N); % Geodetic height of the fixed frame (m)

% Generate Position - Need to determine the ECEF position of the fixed
```

```
% frame
r_e__e_f = zeros(3,N);
for i=1:N
    r_e__e_f(:,i) = llh2xyz(L_f(i), lambda_f(i), h_f(i)); % Position of the origin of the radar
tracking frame wrt e-frame org resolved in the e-frame (m)
end
r_f__f_b = zeros(3,N); % Position of the origin of {b} wrt {f} org resolved in {f} (m)

% Orientation of the Nav frame of the ship at the start wrt the ECEF frame
C_e__n_IMU_start = [ -cos(lambda_f(1))*sin(L_f(1)), -sin(lambda_f(1)) , -
cos(lambda_f(1))*cos(L_f(1));
    -sin(lambda_f(1))*sin(L_f(1)),  cos(lambda_f(1)) , -sin(lambda_f(1))*cos(L_f(1));
    cos(L_f(1))                   ,  0                , -sin(L_f(1))];

C_e__f = C_e__n_IMU_start; % The orientation of the tracker frame stays constant wrt ECEF
r_e__e_b = r_e__e_f + C_e__f * r_f__f_b; % Position of the origin of the b-frame wrt e-frame org
resolved in the e-frame (m)
r_e__e_n = r_e__e_b; % The origin of the Nav frame = origin of the body frame (m)

% The curvlinear coordinates of the Navigation/Body frame
L_b      = zeros(1,N);  % Initialize the arrays
lambda_b = zeros(1,N);
h_b      = zeros(1,N);
for i=1:N
    [L_b(i), lambda_b(i), h_b(i)] = xyz2llh(r_e__e_b(:,i));    % Geodetic Lat (rad), Lon (rad),
height(m) of Body/Nav frame
end

% The differential of curvlinear coordinates of the Navigation/Body frame
L_b_dot = zeros(1,N);
lambda_b_dot = zeros(1,N);
h_b_dot = zeros(1,N);

% Generate Angular Velocity
w_e__e_n = zeros(3,N);                           % Angular velocity of {n} wrt {e} resolved in {e}
w_i__i_n = zeros(3,N);                           % Angular velocity of {n} wrt {i} resolved in {i}
for i=1:N
    w_e__e_n(:,i) = [ sin(lambda_b(i)) * L_b_dot(i); ...
        -cos(lambda_b(i)) * L_b_dot(i); ...
        lambda_b_dot(i)];
    w_i__i_n(:,i) =   w_i__i_e +  C_i__e(:,:,i) *  w_e__e_n(:,i); % Angular velocity of {n} wrt
{i} in {i}
end

% Generate Orientation
C_e__n = zeros(3,3,N);              % Orientation of n-frame wrt e-frame
C_i__n = zeros(3,3,N);              % Orientation of n-frame wrt i-frame
for i=1:N
    C_e__n(:,:,i) = [ -cos(lambda_b(i))*sin(L_b(i)), -sin(lambda_b(i)) , -
cos(lambda_b(i))*cos(L_b(i));
        -sin(lambda_b(i))*sin(L_b(i)),  cos(lambda_b(i)) , -sin(lambda_b(i))*cos(L_b(i));
        cos(L_b(i)),  0                 ,                  -sin(L_b(i))];
    C_i__n(:,:,i) = C_i__e(:,:,i) * C_e__n(:,:,i);
end

% Generate Position
r_i__i_n = zeros(3,N);                          % Position of n-frame wrt i-frame resolved in {i}
for i=1:N
    r_i__i_n(:,i) = C_i__e(:,:,i) * r_e__e_n(:,i); % r_i__i_n = r_i__i_e + C_i__e * r_e__e_n ->
Eqn 2.96
end

% Generate Velocity
v_e__e_n = zeros(3,N);                          % Velocity of n-frame wrt e-frame resolved in the e-
frame
for i=1:N
    RN = (1 - e^2) * R0 /      (1 - e^2*sin(L_b(i))^2)^(3/2);    % Meridian radius of curvature
(m)
    RE =              R0 / sqrt(1 - e^2*sin(L_b(i))^2);          % Transverse radius of curvature
(m)
    v_n__e_n = [              (RN + h_b(i)) * L_b_dot(i);
```

```
        cos(L_b(i)) * (RE + h_b(i)) * lambda_b_dot(i);
        -h_b_dot(i)];
    v_e__e_n(:,i) = C_e__n(:,:,i) * v_n__e_n;
end

v_i__i_n = zeros(3,N);                          % Velocity of n-frame wrt i-frame in the i-frame
for i=1:N
    v_i__i_n(:,i) = C_i__e(:,:,i) * (v_e__e_n(:,i) + Ohm_i__i_e * r_e__e_n(:,i) );
end

% Generate Acceleration
a_e__e_n = (v_e__e_n(:,2:end) - v_e__e_n(:,1:end-1))/dt;  % Acceleration (numerically) of n-frame
wrt e-frame in the e-frame
a_e__e_n = [a_e__e_n, a_e__e_n(:,end)];
a_i__i_n = zeros(3,N);                          % Acceleration of n-frame wrt i-frame in the i-frame
for i=1:N
    a_i__i_n(:,i) = C_i__e(:,:,i)*(a_e__e_n(:,i) + 2*Ohm_i__i_e*v_e__e_n(:,i) +
Ohm_i__i_e*Ohm_i__i_e*r_e__e_n(:,i)); % Eqn 2.96
end

%--------------------------------------------------------------------------
%% Navigation to Body frame
%--------------------------------------------------------------------------

% Generate Orientation
C_i__b = zeros(3,3,N);              % Orientation of b-frame wrt i-frame
C_e__b = zeros(3,3,N);              % Orientation of b-frame wrt e-frame
C_n__b = zeros(3,3,N);              % Orientation of b-frame wrt n-frame

test = menu('Test Sample','Fixed Position, No Rotation','Fixed Position, Rotate about Z');

switch test
    case 1 % Fixed Position, No Rotation
        load('data/d_t_still.mat');  % Load in IMU data
        w_b__i_b = d_t_still(:,7:9)';
        f_b__i_b = d_t_still(:,4:6)';

        for i=1:N % Orientation of the b-frame wrt the n-frame
            C_n__b(:,:,i) = rotate_z(pi);
            C_i__b(:,:,i) = C_i__n(:,:,i) * C_n__b(:,:,i);
            C_e__b(:,:,i) = C_e__n(:,:,i) * C_n__b(:,:,i);
        end

        % Generate Angular Velocity
        w_n__n_b = zeros(3,N); % Angular velocity of b-frame wrt n-frame in the n-frame
        w_i__i_b = zeros(3,N); % Angular velocity of b-frame wrt i-frame in i-frame
        w_e__e_b = zeros(3,N); % Angular velocity of b-frame wrt i-frame in i-frame
        for i=1:N
            w_n__n_b(:,i) = [ 0; 0; 0];
            w_i__i_b(:,i) = w_i__i_n(:,i) + C_i__n(:,:,i) * w_n__n_b(:,i);
            w_e__e_b(:,i) = w_e__e_n(:,i) + C_e__n(:,:,i) * w_n__n_b(:,i);
        end

    case 2 % Fixed bias removed
        load('data/d_t_rot.mat');  % Load in IMU data
        w_b__i_b = d_t_rot(:,7:9)';
        f_b__i_b = d_t_rot(:,4:6)';

        for i=1:N % Orientation of the b-frame wrt the n-frame
            C_n__b(:,:,i) = rotate_z(30*i/100*pi/180);
            C_i__b(:,:,i) = C_i__n(:,:,i) * C_n__b(:,:,i);
            C_e__b(:,:,i) = C_e__n(:,:,i) * C_n__b(:,:,i);
        end

        % Generate Angular Velocity
        w_n__n_b = zeros(3,N); % Angular velocity of b-frame wrt n-frame in the n-frame
        w_i__i_b = zeros(3,N); % Angular velocity of b-frame wrt i-frame in i-frame
        w_e__e_b = zeros(3,N); % Angular velocity of b-frame wrt i-frame in i-frame
        for i=1:N
            w_n__n_b(:,i) = [ 0; 0; 30*pi/180];
            w_i__i_b(:,i) = w_i__i_n(:,i) + C_i__n(:,:,i) * w_n__n_b(:,i);
```

```
            w_e__e_b(:,i) = w_e__e_n(:,i) + C_e__n(:,:,i) * w_n__n_b(:,i);
        end
end

% Recall that the Navigation and body frames have the same origins
% Generate Position
r_n__n_b = [0; 0; 0];  % Position of b-frame wrt n-frame
v_n__n_b = [0; 0; 0];  % Velocity of b-frame wrt n-frame
a_n__n_b = [0; 0; 0];  % Acceleration of b-frame wrt n-frame


%**************************************************************************
%% *********   IMU measurements  ******************************************
%**************************************************************************

calibration = menu('Calibration Type','None','Fixed Bias','Misalignment/Scale','Complete');

load('calibration.mat');

switch calibration
    case 1 % No calibration case

    case 2 % Fixed bias removed
        for i = 1:N
            w_b__i_b(:,i) = w_b__i_b(:,i) - b_g_FB;
            f_b__i_b(:,i) = f_b__i_b(:,i) - b_a_FB;
        end

    case 3 % Misalignment/scale factor calibrated
        for i = 1:N
            w_b__i_b(:,i) = inv(eye(3) + M_g) * w_b__i_b(:,i);
            f_b__i_b(:,i) = inv(eye(3) + M_a) * f_b__i_b(:,i);
        end

    case 4 % Complete calibration
        for i = 1:N
            w_b__i_b(:,i) = inv(eye(3) + M_g) * (w_b__i_b(:,i) - b_g_FB);
            f_b__i_b(:,i) = inv(eye(3) + M_a) * (f_b__i_b(:,i) - b_a_FB);
        end
end


%--------------------------------------------------------------------------
%% ECEF Mechanization
%--------------------------------------------------------------------------
r_e__e_b_INS = zeros(3,N);
v_e__e_b_INS = zeros(3,N);
C_e__b_INS   = zeros(3,3,N);

% Initialize the INS mechanization (Using ground truth)
C_e__b_INS(:,:,1) = C_e__b(:,:,1);      % No errors in the initialization
v_e__e_b_INS(:,1) = v_e__e_n(:,1);      % Remember: the origin of b-frame = origin of n-frame
r_e__e_b_INS(:,1) = r_e__e_b(:,1);

for i=2:N  % Call the mechanization for each iteration
    [r_e__e_b_INS(:,i)  , v_e__e_b_INS(:,i)  , C_e__b_INS(:,:,i)] = ECEF_mech(...
        r_e__e_b_INS(:,i-1), v_e__e_b_INS(:,i-1), C_e__b_INS(:,:,i-1), ...
        w_b__i_b(:,i), f_b__i_b(:,i), fidelity); % Error free IMU
end

% % Plot the ECEF PVA Ground truth, INS derived PVA, & Error betw the two
plot_PVA_Jeff(r_e__e_n, v_e__e_n, C_e__b, r_e__e_b_INS, v_e__e_b_INS, C_e__b_INS, 'ECEF',
fidelity,  N, t_sec)
```

12. REFERENCES

[1] Bruder, S. (n.d.). Inertial Sensors & Errors. Retrieved February 11, 2015, from http://mercury.pr.erau.edu/~bruders/teaching/2015_spring/EE495/lectures/lecture 14_Inertial_Sensor_Errors.pdf

[2] Groves, P. (2013). Principles of GNSS, inertial, and multisensor integrated navigation systems (Second ed.).

[3] Gulf Coast Data Concepts, LLC. (2014). Calibrating 3-Axis Accelerometers. Retrieved April 15, 2015, from http://www.gcdataconcepts.com/calibration.html

[4] VectorNav. (n.d.). Calibration - VectorNav Library. Retrieved April 15, 2015, from http://www.vectornav.com/support/library/calibration

[5] VectorNav. (n.d.). VN-200 Rugged Specifications. Retrieved April 15, 2015, from http://www.vectornav.com/products/vn200-rugged/specifications