ECE 167/L: Sensing and Sensor Technologies
Lab 3 Report

Jose Santiago

Table of Contents

Lab Introduction and Overview

The purpose of this lab is to familiarize ourselves with a 9DOF IMU sensor and the errors in the sensing elements. We learn to communicate with the IMU, understand error sources, and apply linear algebra techniques to calibrate the errors out of the sensor. We will learn about bias drift and the basics of gyro integration to get an angle from the raw angular rate of the sensor. The required hardware for this lab are the UNO32, Sparkfun IMU Module, and Earth's Gravity and Magnetic Field.

Low cost MEMs accelerometers, gyros, and magnetometers can be used in combination to give good high bandwidth information about the orientation of the device in space. These miniaturized devices have several error sources, such as temperature drift, non-linearity, hysteresis, and cross coupling. In this lab we focus on scale factor and bias error for the sensors, along with methods to measure and calibrate out the biases. We specifically take interest in gyro bias drift and use naive methods for removing it, while testing the effectiveness of this method.

Ellipsoid Calibration using Simulated Data

Before calibrating the 3-axis sensors, we should understand the geometry of the problem. We start by doing a problem in 2D first. We will be using MATLAB to calibrate the data from a hypothetical 2D sensor. This sensor measures the x and y components of a fixed vector. As the sensor is rotated, the axes of the sensor measures the components of the fixed vector. Plotting the points from the sensor depicts a circle of radius equal to the length of the vector. The sensor has a null shift on each of the axes, and the scale factors are different on both axes. Therefore, rotating the sensor and the points gives us an ellipse that is centered around the biases. The semi-major and semi-minor axes lengths correspond to the scale factors, and there is a wideband noise on the measurements.

Given the noisy points on the ellipse, we want to find the scale factor, cross-coupling, and null shifts so that we can reconstruct the circle centered at the origin with radius corresponding to the vector. We can start by expanding the ellipse equation. Expanding this equation gives us the following equation:

$$x^2 = [a^2R^2 - x_o^2 - \frac{a^2y_o^2}{b^2}] - y^2[\frac{a^2}{b^2}] + y[2\frac{a^2y_o}{b^2}] + x[2x_o]$$

We can then put this equation in vector form to be able to use in MATLAB to solve for the Least Squares. We can define a vector A with the provided x and y values from MATLAB file EllipseXYData.m as follows:

$$A = [x \; -y^2 \; y \; 1]$$

And then we set the R vector equal to the square of the x measurements provided by the same file used in the A vector as follows:

$$R = x^2$$

With this information we can perform the Least Squares of the data with the \ operation on MATLAB (A\B). This gives us the values necessary to solve for our x_o, y_o, a, and b values. The

result from this operation are four values:

$$[2x_o \ \frac{a^2}{b^2} \ 2\frac{a^2 y_o}{b^2} \ a^2R^2 - x_o^2 - \frac{a^2 y_o^2}{b^2}] \ = \ [0.7923 \ 0.2658 \ -0.1801 \ 1.0349]$$

Then, solving for the respective values, we get:

$$x_o \ = \ 0.3962$$

$$y_o \ = \ -0.3388$$

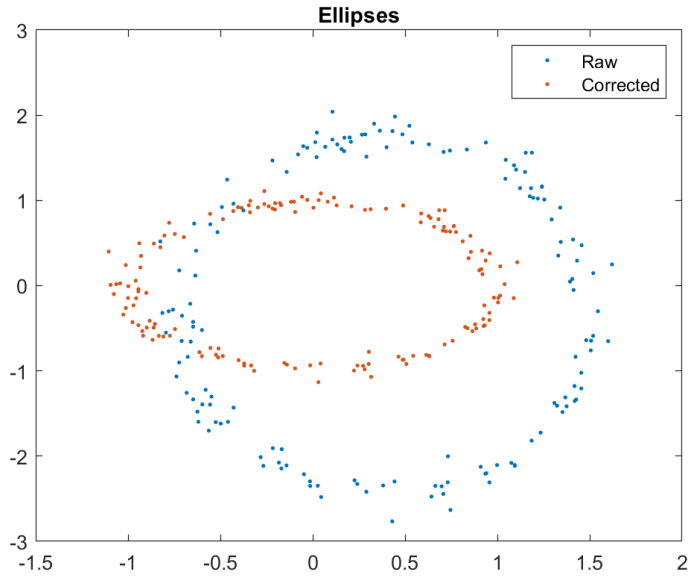
$$a \ = \ 1.1056$$

$$b \ = \ 2.1445$$

Using these values we can find the x and y measurements corrected via our Least Squares:

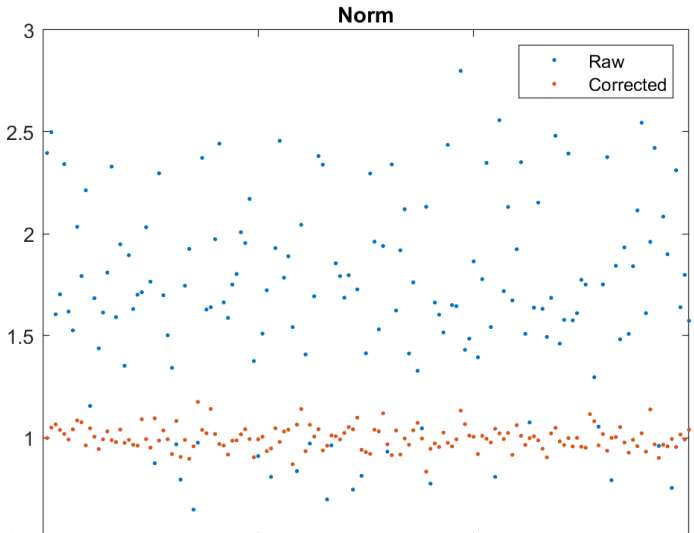
$x_h = \frac{x - x_o}{a}$ and $y_h = \frac{y - y_o}{b}$

With the values we solved for, and the provided x and y values from the EllipsesXYData.m file, we can find the corrected x and y values, x_h and y_h. Plotting both the corrected and uncorrected



x and y values we can see both ellipses as follows:

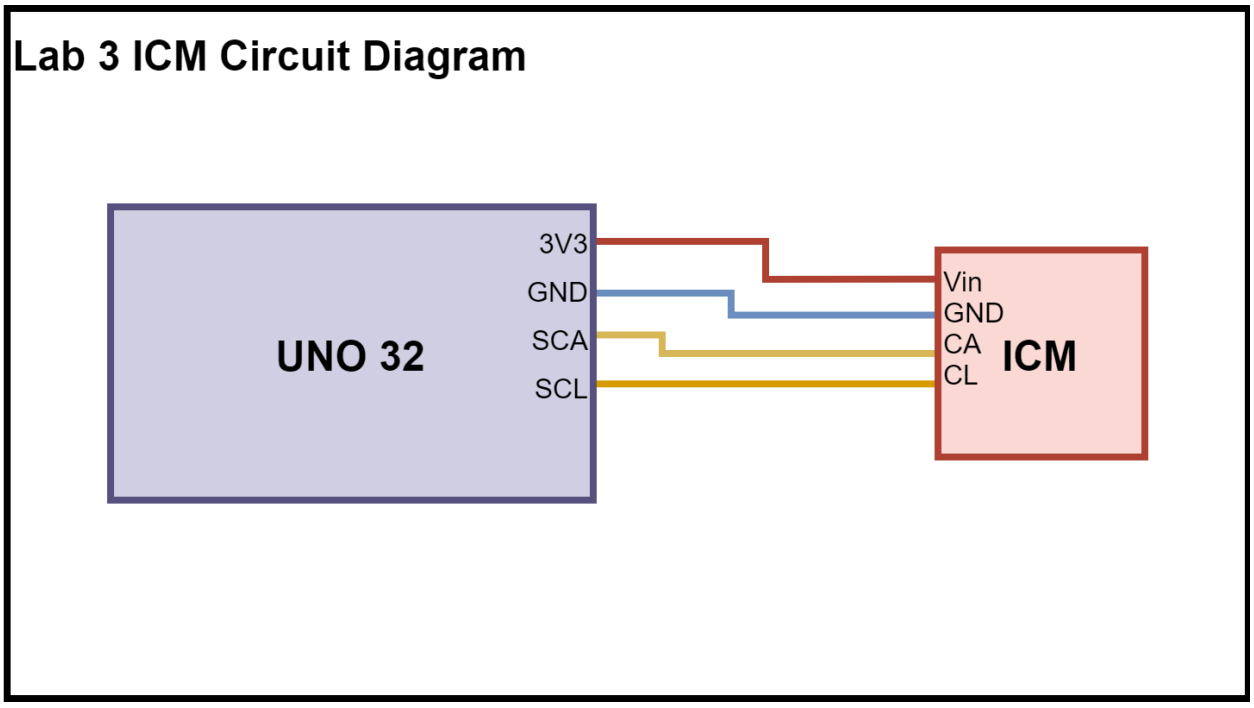
We can see in the figure how the ellipse from the corrected data has a radius of one, while the



ellipse from the raw data does not. If we take a look at the 2-norm of the pre- and post-calibrated data we can confirm that our data has been calibrated:
In the plot of the 2-norm we can see that the raw data is very spread out, while the calibrated data congregates around 1.

Naive Calibration of Magnetometer and Accelerometer

To begin calibrating the Magnetometer and Accelerometer we must first wire up the IMU to the UNO32. The circuit is fairly simple, but some changes have to be made to the UNO32. We must take the IO-shield off to access jumpers underneath. The jumpers for JP6 and JP8 on the micro must be shifted so that the jumpers are closest to the labels RG3 and RG2 respectively. In other words the pins for A4 and A5 will be exposed. Then the IO-shield can be put back on and we can connect our IMU to the UNO32 as shown in the following diagram:



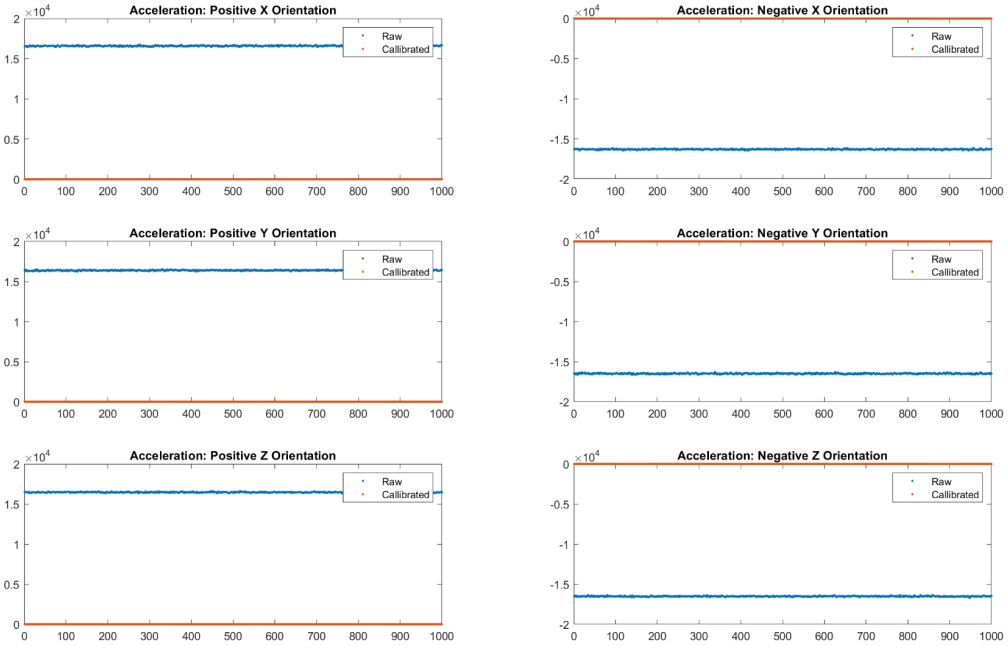
After setting up our circuit we can begin calibrating the Magnetometer and Accelerometer naively. To do this we must find the bias and scale factor of our magnetometer and accelerometer in the x, y, and z axes. We start with the Accelerometer as our expected values are more easily identifiable as the acceleration due to gravity when the IMU is at rest. We can start with the Z axis as the IMU lays flat on a desk. In this orientation we expect our reading to be 1g, or as the datasheet describes 16,384 as the sensor's raw value. We take about a thousand samples in this axis and average the samples to get a good estimate of the actual reading of the accelerometer due to gravity. We invert the orientation of the IMU in the Z axis (we flip it over) and then take another thousand samples. We take an average of the samples taken by our sensor again to find an estimate for the actual readings of the accelerometer due to gravity when the Z axis is inverted. With these two readings we can get the full range of values the sensor outputs for acceleration in the Z axis. The values I recorded were 16,472 for g and -16,514 for -g. With this data we can find the scaling factor. We sum the plus and minus g

values recorded and divide them by two, giving us 16,493 as the scaling factor. To find the bias or null shift, we can simply add the scaling factor to -g. This gives us our new midpoint and null shift, -21. We can repeat this process for the x and y axes to find the scaling factors and null shifts of:

$sf_x = 16,435;\ ns_x = 158$

$sf_y = 16,439;\ ns_y = -48$

With this information we can shift and scale our data to +/- 1g. We do this by subtracting the respective bias to every axis and then dividing the new maximum and minimum of the range of values from the sensor by the scale factor. This leaves us with a scale of +/- 1g as shown in the following figure:



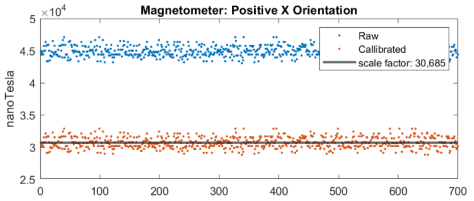
We can see that the raw readings are about +/- 16,500 and the calibrated values are +/- 1.

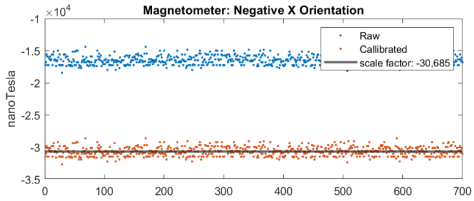
To calibrate the Magnetometer we had to hold the sensor in the air and turn it around in different orientations until two out of the three axes were zero or close to zero. I started with the Z axis. When I found the position that made the sensor values for X and Y go to zero and the Z values go positive I started logging my data. Then I changed the orientation of the sensor so that the sensor values for X and Y go to zero, but the Z values go negative and recorded this data. After collecting this data I averaged the values of the respective positive and negative orientations of the magnetometer in the Z axis. After converting to nanoTesla (by multiplying by 150) I found that the range of values for the magnetometer in the Z axis were about -75,400 to 10,000. This gives us a scale factor of 42,580. Adding this to our negative Z axis value we get a bias of 32,568. We repeat this process for the X and Y axes. To find the following scale factors and biases:

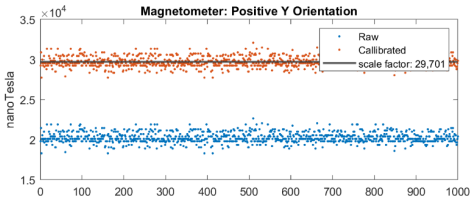
$sf_x = 30,685;\ ns_x = 14,231$

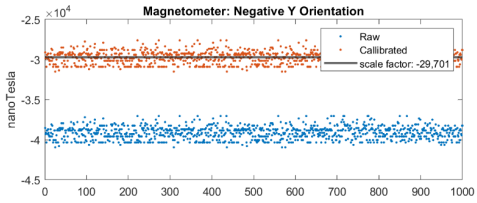
$sf_y = 29,700;\ ns_y = -9,4439$

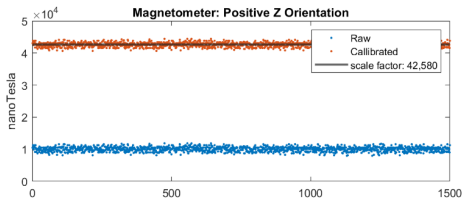
When we scale these by subtracting the bias by the average of our samples in the positive and negative orientations and then dividing by the scale factor we get the following figures:

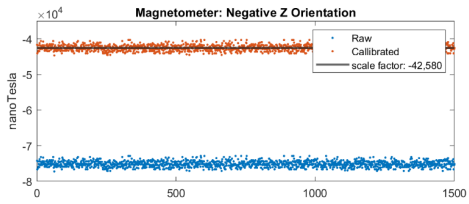












Gyro Bias and Bias Drift

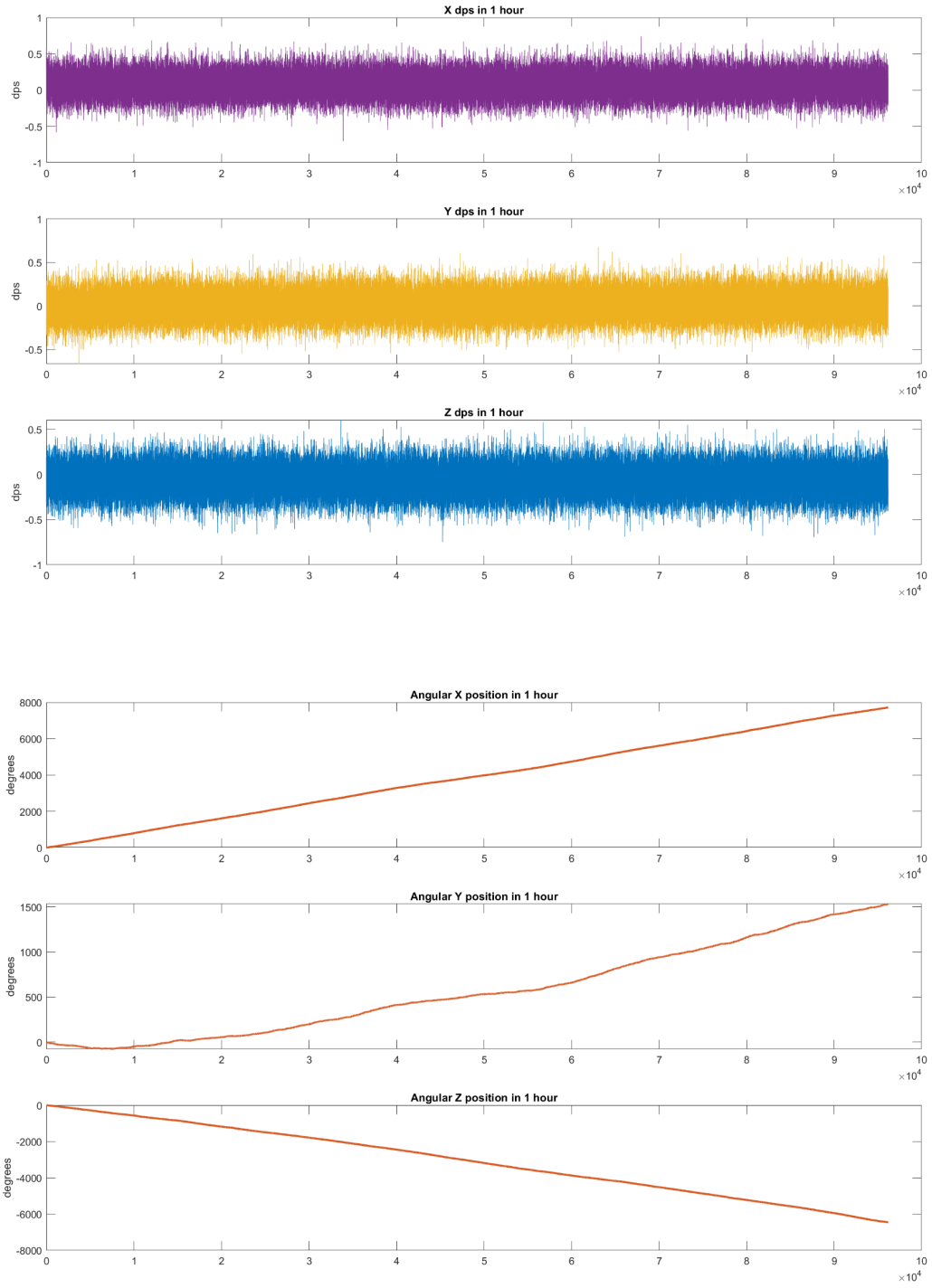
Next we want to start working with our gyroscope. We start by analyzing the bias and bias drift. We take 10 seconds of data as the gyroscope lay at rest on a desk and average the samples from every axis to get an initial estimate of our bias drift for every axis. We also have to convert our data to degrees per second by dividing the raw values by 131. The initial biases from my 10 second samples were:

$b_x = -0.5651$

$b_y = 0.3342$

$b_z = 0.0913$

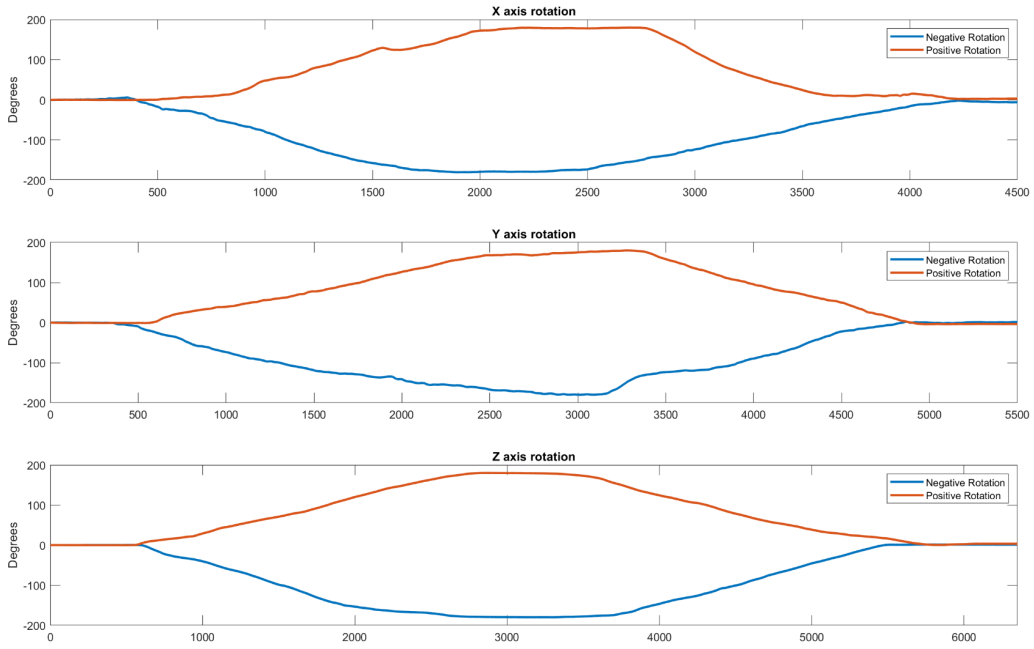
After this we record data for an hour long to find the bias and bias drift accrued by the gyroscope. We subtract the bias estimated from the 10 second data to this 1 hour data to try and calibrate it. After recording the value for an hour we can see the gyro bias in degrees per second in the figure below. If we integrate the gyro's bias in degrees per second with the MATLAB function cumtrapz(), we can find the change in degrees recorded by the gyroscope according to the degrees per second of the gyroscope. We can see how the degrees change as the gyroscope sits for an hour.



Above we can see how the gyroscope has a drift in degrees per second, and how that corresponds to a perceived change in degrees of the gyroscope's position. With this we can conclude that we cannot calibrate the gyroscope naively by subtracting the bias.

Gyro Scale Factor via Angle Integration

The proper method for calibrating the gyroscope is via something called a “rate table.” This “rate table” spins the sensor at a constant rate on each axis and records the data. They are expensive so we will not be using this. Instead we calibrate the scale factor by integrating the rotation rate through a constant set of angles. We will spin the gyroscope in every axis by 180 degrees and back to 0 degrees. We will also do it in the negative direction to go from -180 degrees and back to 0 degrees. After logging this data we convert the data into degrees per second in MATLAB. We also take 10 seconds of data of the gyroscope at rest again to estimate the bias for every axis. We subtract this bias from the degrees per second data of the gyroscope when it is rotated to and from +/- 180 degrees. We then change these degrees per second readings of the gyroscope to degrees in every axis. We also store the maximum and minimum of these degree readings and scale them to 180 degrees. We do this by multiplying the raw degree value by 180 and dividing by either the maximum of the readings, or the minimum of the readings, depending on whether the degrees are positive or negative. After this we can see that our gyroscope is working as intended via the figure below:



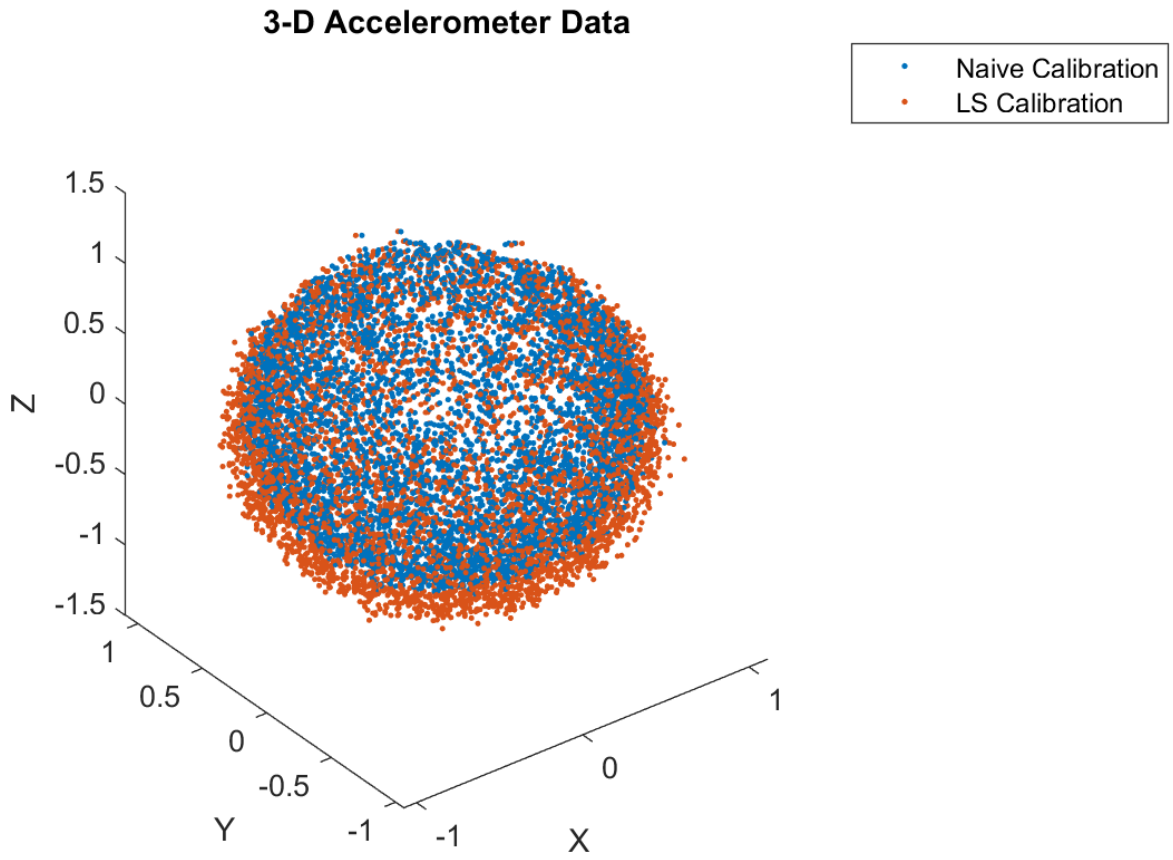
It is clear how the gyroscope is turned back and forth from both +/- 180 degrees in every axis.

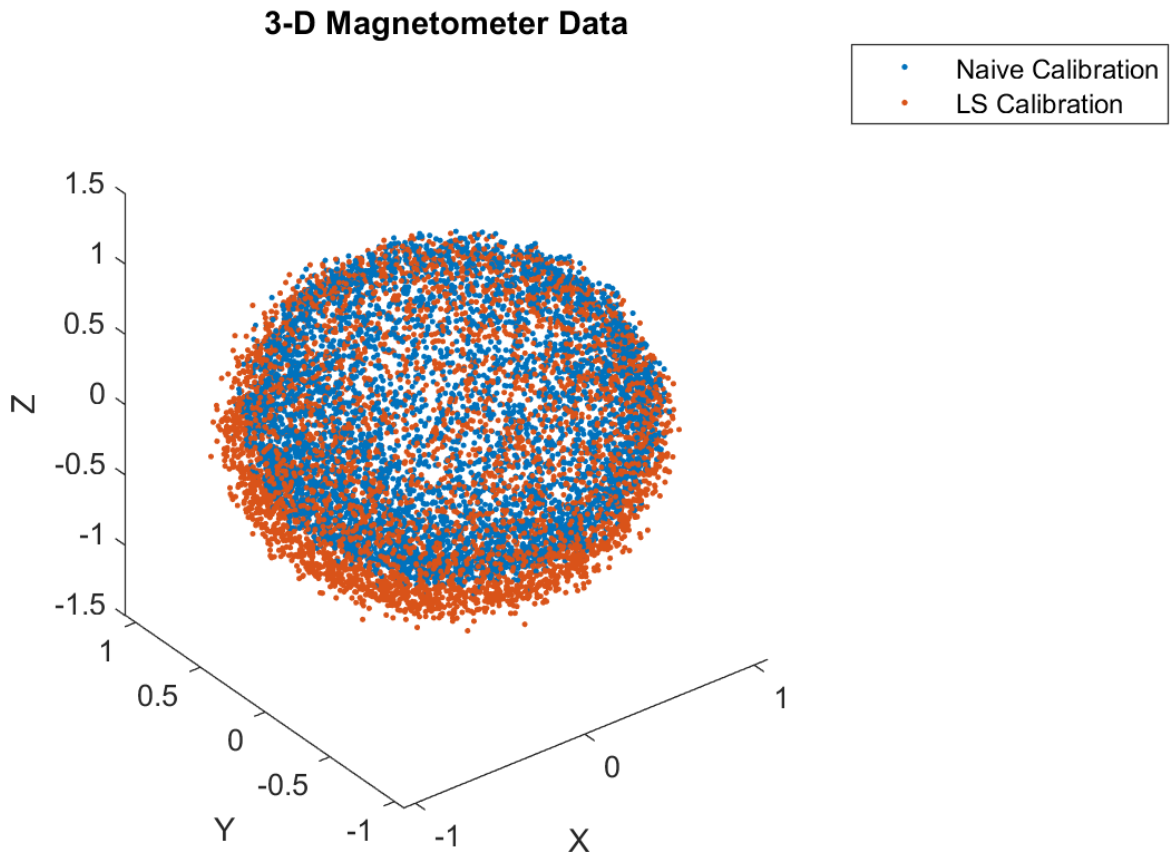
Tumble Test Simulation

Next we run a simulation of the tumble test prior to running the actual tumble test. We are given noise values of our accelerometer and magnetometer sensors and given the task to calibrate the data naively and plot the 3D data. We are then given MATLAB equations to perform a Least Squares calibration of the simulated tumble data. Before calibrating the data we

convert to engineering units, so we divide the tumble data of the accelerometer by 16,384 to get it in terms of gravity and multiply the magnetometer data by 150 to convert to nanoTesla. To calibrate this data naively I recorded the maximum and minimum values of the tumble data. I chose these as the maximum and minimum values of the ranges for the accelerometer and magnetometer for their respective x y and z axes. I found the scale factor as I did in every other section, by taking the sum of the maximum and absolute value of the minimum and dividing it by two for every axis of both the accelerometer and magnetometer. Then I took the mean of the data for every axis to be used as the bias. With this I would be able to scale the tumble data to +/- the scale factor of the tumble data by subtracting the converted data in engineering units by the mean/bias and dividing by the scale factor we calculated.

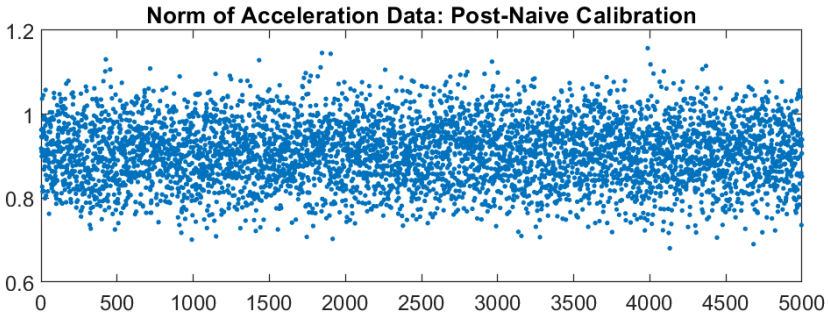
After completing the naive calibration of this data we could move on to implementing the Least Squares. To do this we simply call CalibrateEllipsoidData3D, provided to us, followed by the CorrectEllipsoidData3D. This data gives us 3D plots of the accelerometer and magnetometer data corrected to be an ellipsoid with a radius of 1. We can also plot the norm of both the naive calibration and Least Squares calibration to compare the accuracy of each method. We can see the 3D plots as well as the calculated norms below:

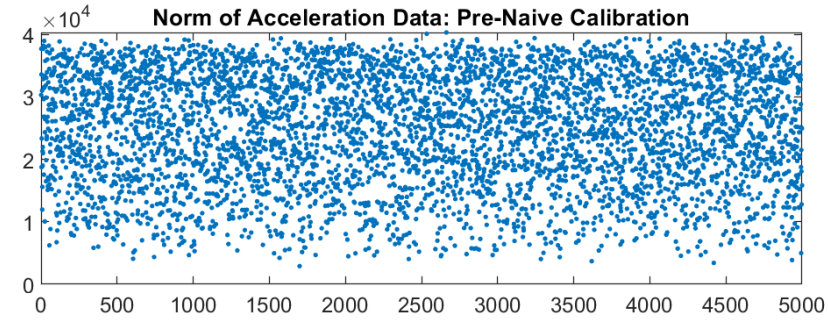


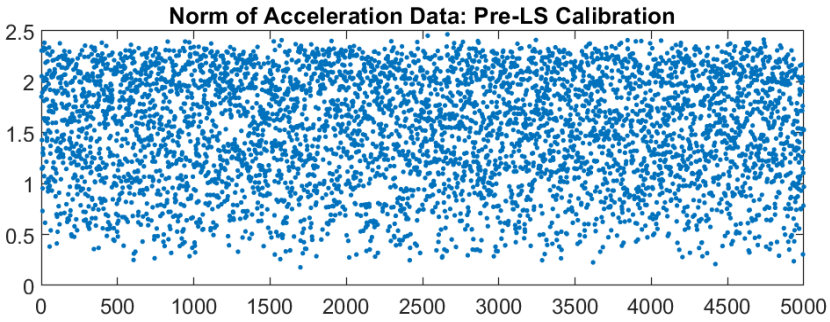


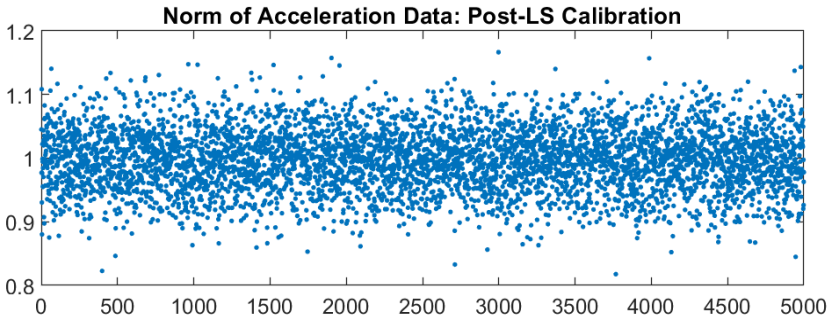
In these 3D plots we can see that the LS calibration is more accurate than the naive calibration.

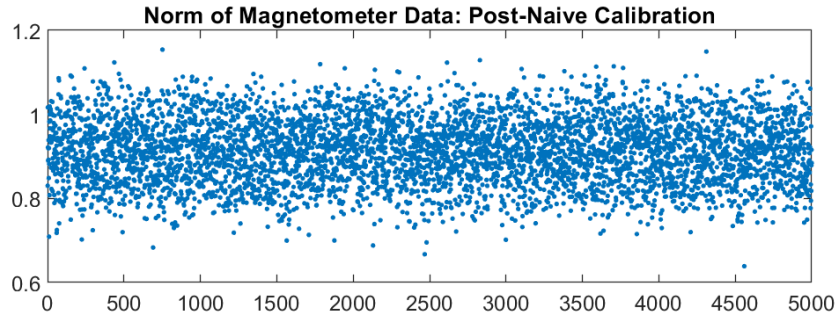
It may be difficult to see here, but if we look at the norms we can see the clear difference:

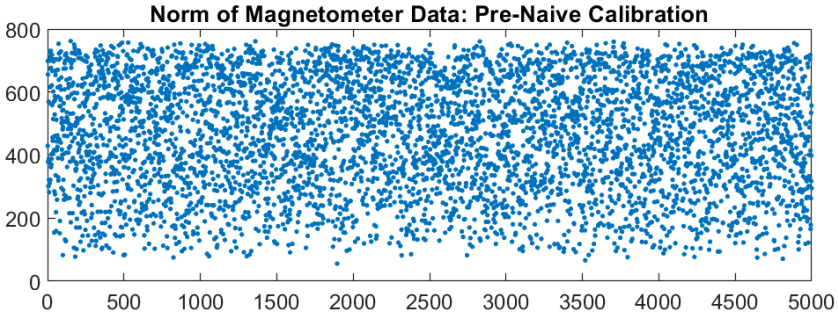


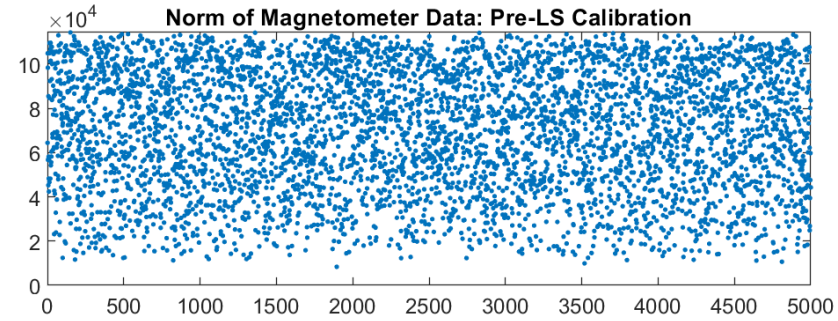


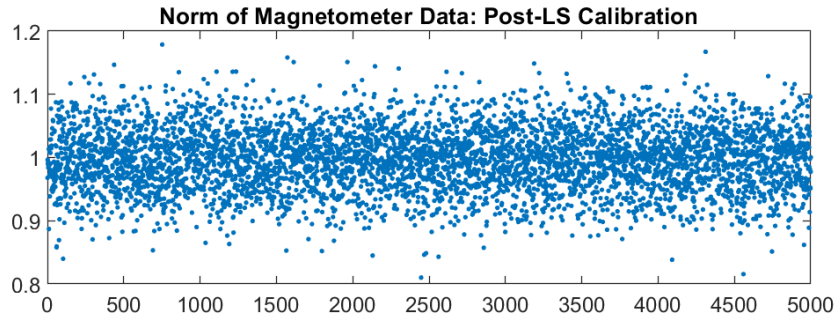





Norm of Magnetometer Data: Post-Naive Calibration
1.2
1
0.8
0.6
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

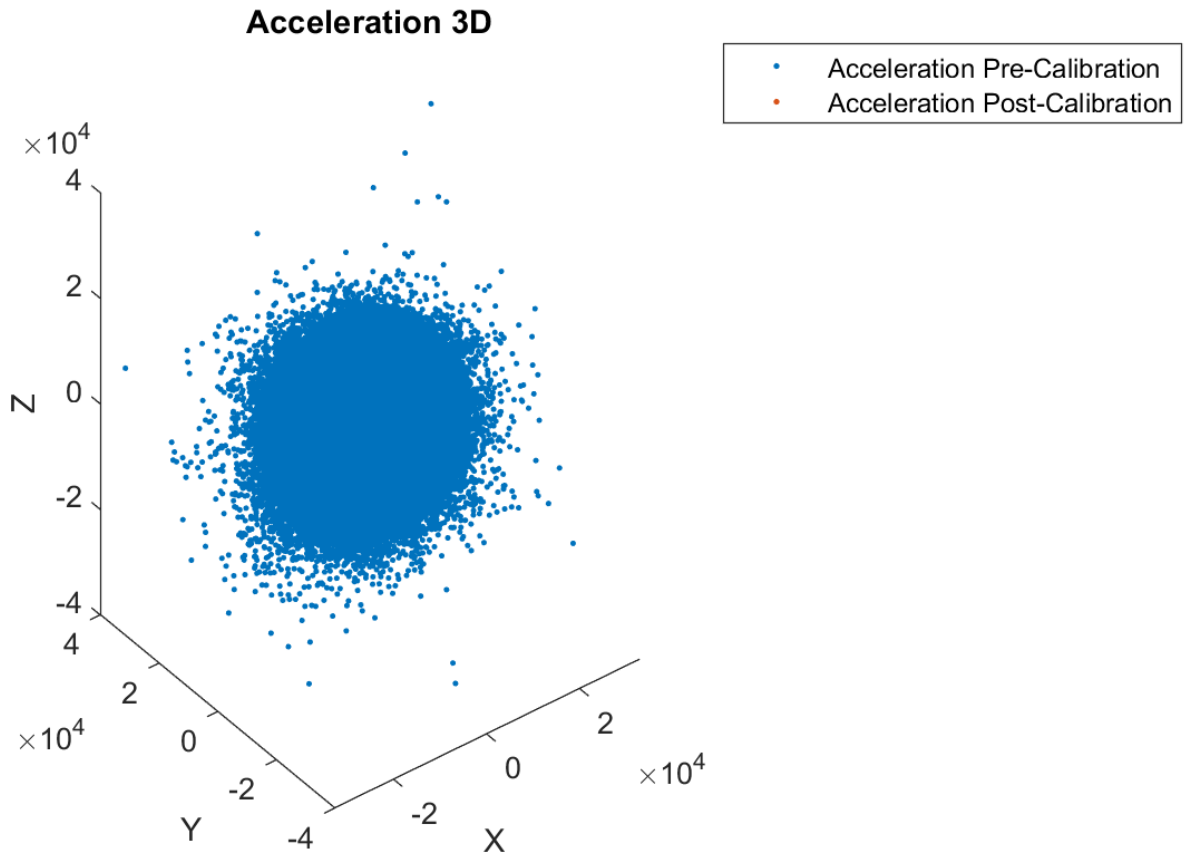

Norm of Magnetometer Data: Pre-Naive Calibration
800
600
400
200
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


Norm of Magnetometer Data: Pre-LS Calibration
×10^4
10
8
6
4
2
0
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

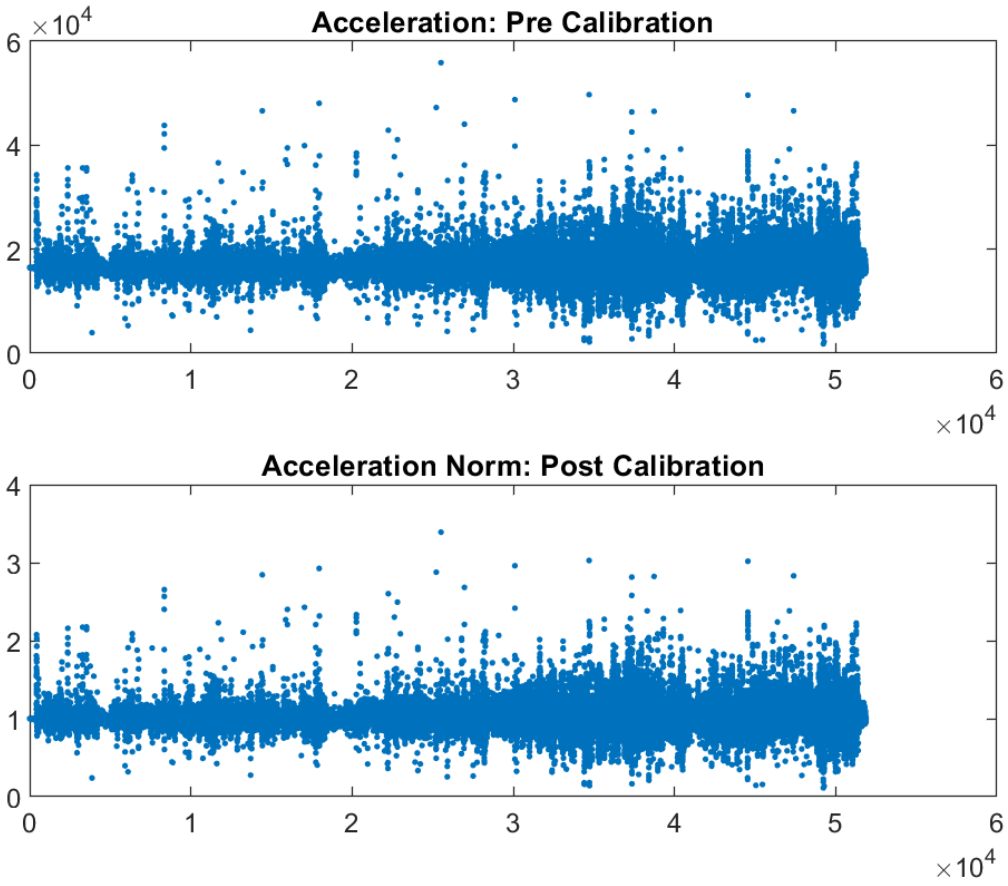

Norm of Magnetometer Data: Post-LS Calibration
1.2
1.1
1
0.9
0.8
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

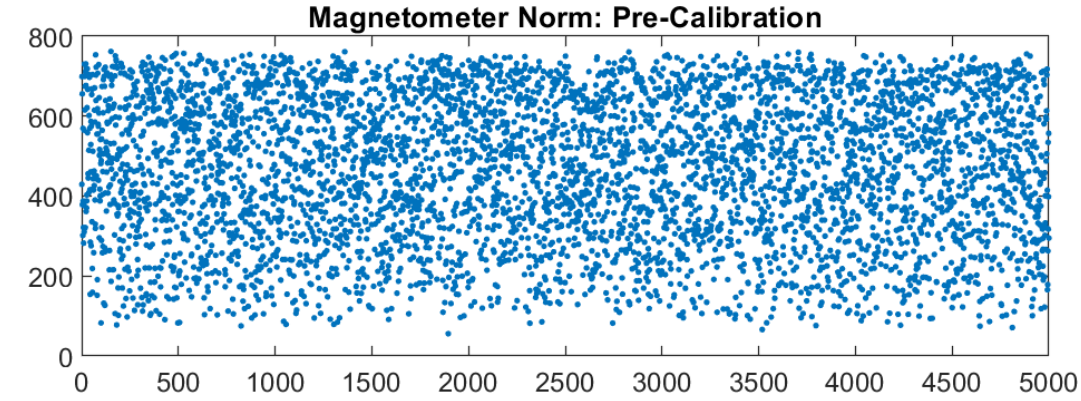
Tumble Test for Accelerometer and Magnetometer

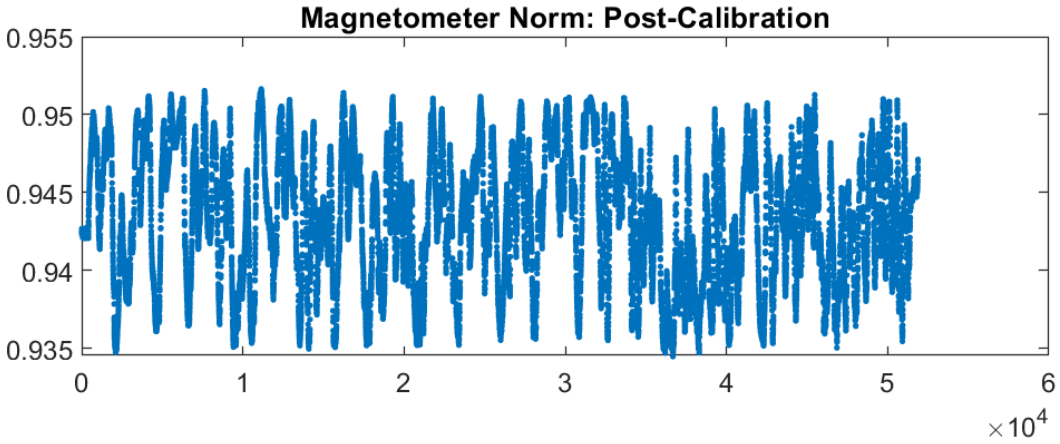
After successfully calibrating the simulated data in the previous section, we can collect our own data from our sensor and calibrate it naively according to the data we collected from the naive calibration earlier in the lab. I tumbled the sensor in my hands for 5 minutes and used this data to perform my naive calibration. I used the values of the scale factor and bias from the previous experiment I performed for both the magnetometer and accelerometer naive calibration. I used this data to calibrate my raw readings from my tumble data as I did in previous sections. I subtracted the bias from every respective axis and divided it by the scale factor. Below we can see what I got for the 3D plots of data before and after calibrating the data.



Here we have the 3D plots of both the pre and post calibrated data, but you can't really see the calibrated data due to the difference in scale. We can have a better look at what is going on with the norm data.


Acceleration: Pre Calibration
×10^4
6
4
2
0
0
1
2
3
4
5
6
×10^4
Acceleration Norm: Post Calibration
4
3
2
1
0
0
1
2
3
4
5
6
×10^4





With the norm plots we can see how we scaled the data to about 1 for both the accelerometer and magnetometer.

Conclusion

In conclusion this lab was very different from others. We spent the whole lab calibrating our sensors in MATLAB. It was a very experimental lab in learning exactly how the sensors worked and how they didn't. I think this lab is useful and can be applicable to several different sensors. I think it really showed the power of MATLAB in understanding and calibrating our sensor data. Although I think the math was complicated at parts, it was a nice refresher in Linear and Matrix math. These aspects of my major curriculum had yet to be fully utilized but I'm glad they were for this lab. Understanding the 3D space using these sensors has been really informative. I look forward to working with this sensor in the next lab and seeing how we can utilize it to its full potential.