

2016

An Alternative Sensor Fusion Method For Object Orientation Using Low-Cost Mems Inertial Sensors

Joshua Lee Bouffard
University of Vermont

Follow this and additional works at: <http://scholarworks.uvm.edu/graddis>

 Part of the [Aerospace Engineering Commons](#), [Electrical and Electronics Commons](#), and the [Mathematics Commons](#)

Recommended Citation

Bouffard, Joshua Lee, "An Alternative Sensor Fusion Method For Object Orientation Using Low-Cost Mems Inertial Sensors" (2016). *Graduate College Dissertations and Theses*. Paper 537.

This Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks @ UVM. It has been accepted for inclusion in Graduate College Dissertations and Theses by an authorized administrator of ScholarWorks @ UVM. For more information, please contact donna.omalley@uvm.edu.

AN ALTERNATIVE SENSOR FUSION METHOD FOR OBJECT ORIENTATION
USING LOW-COST MEMS INERTIAL SENSORS

A Thesis Presented

by

Joshua L. Bouffard

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Master of Science
Specializing in Electrical Engineering

May 2016

Defense Date: April 4, 2016
Thesis Examination Committee:

Stephen Titcomb, Ph.D., Advisor
Walter Varhue, Ph.D., Chairperson
Tian Xia, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of the Graduate College

ABSTRACT

This thesis develops an alternative sensor fusion approach for object orientation using low-cost MEMS inertial sensors. The alternative approach focuses on the unique challenges of small UAVs. Such challenges include the vibrational induced noise onto the accelerometer and bias offset errors of the rate gyroscope. To overcome these challenges, a sensor fusion algorithm combines the measured data from the accelerometer and rate gyroscope to achieve a single output free from vibrational noise and bias offset errors.

One of the most prevalent sensor fusion algorithms used for orientation estimation is the Extended Kalman filter (EKF). The EKF filter performs the fusion process by first creating the process model using the nonlinear equations of motion and then establishing a measurement model. With the process and measurement models established, the filter operates by propagating the mean and covariance of the states through time.

The success of EKF relies on the ability to establish a representative process and measurement model of the system. In most applications, the EKF measurement model utilizes the accelerometer and GPS-derived accelerations to determine an estimate of the orientation. However, if the GPS-derived accelerations are not available then the measurement model becomes less reliable when subjected to harsh vibrational environments. This situation led to the alternative approach, which focuses on the correlation between the rate gyroscope and accelerometer-derived angle. The correlation between the two sensors then determines how much the algorithm will use one sensor over the other. The result is a measurement that does not suffer from the vibrational noise or from bias offset errors.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to Stephen Titcomb, Ph.D. As my advisor, you have offered a tremendous amount of help, support, and encouragement. Without you, this thesis would not have been possible... Thank you!

I would like to thank my committee members, Walter Varhue, Ph.D. and Tian Xia, Ph.D. for your time, interest, and patience.

To my parents, Alan and Priscilla Bouffard, thank you for all your love and support. Your emphasis on educational excellence and self-improvement gave me the confidence to pursue my Master's degree in engineering at UVM. Your life lessons have proven invaluable, and I truly appreciate all you have done and continue to do for me.

To Alanna Bouffard, thank you for your love and patience. You have been an invaluable resource for guidance and support. The numerous sacrifices you made allowing me the time and energy to complete this work were incredible, and I thank you for every one of them. I am blessed to have you in my life as my wife and best friend...I love you!

To my son Isaac, thank you for the sleepless nights and extra motivation to complete this thesis. You have changed my life forever, and I hope to return the favor one day of giving you all the love and motivation to pursue your dreams.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
LIST OF TABLES	v
LIST OF FIGURES	vi
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Goal of the Thesis	3
1.3 Structure of the Thesis.....	4
Chapter 2 Unmanned Aerial Vehicle (UAV).....	6
2.1 Introduction	6
2.2 Summary	11
Chapter 3 An Introduction to MEMS Inertial Sensors.....	12
3.1 The Accelerometer	12
3.1.1 Accelerometer Sensor Measurements.....	18
3.1.2 ADC Conversion Overview	20
3.1.3 Computing Angles from Accelerometers	22
3.2 The Rate Gyroscope	26
3.2.1 Rate Gyroscope MEMS Sensor	35
3.2.2 ADC Conversion Overview	37
3.3 Euler Angles & Elementary Rotations.....	37
3.3.1 Numerical Corrections	43
3.3.2 Euler Angle Computation and Singularity Avoidance	44
3.4 Summary	47
3.5 Lab Exercise.....	49
3.5.1 Part A	49
3.5.2 Part B	51
3.5.3 Part C	53
Chapter 4 An Introduction to Sensor Fusion.....	54
4.1 Introduction	54
4.2 Sensor Fusion Implementation.....	55
4.3 The Kalman Filter	57

4.3.1	Kalman Filter Summary.....	67
4.4	Lab Exercise.....	70
4.4.1	Part A	71
4.4.2	Part B	74
4.4.3	Part C	76
Chapter 5	An Alternative Sensor Fusion Approach.....	81
5.1	Introduction	81
5.2	Related Work.....	83
5.3	Proposed Algorithm	84
5.3.1	Development of the Algorithm	86
5.4	Lab Exercise & Experimental Results	91
5.4.1	Lab Exercise Part 1 – Programming the MCU	94
5.4.2	Lab Exercise Part 2 – Data Collection	97
5.4.3	Lab Exercise Part 3 – Data Analysis.....	99
5.4.4	Experimental Test Results	105
5.5	Summary	107
5.6	Future Work	108
	BIBLIOGRAPHY.....	109

LIST OF TABLES

Table 1 ADXL335 Accelerometer Datasheet [24]	19
Table 2 InvenSense IDG-500 Datasheet [25]	36

LIST OF FIGURES

Figure 1 Multiplex “Easy Star” with wireless camera system.....	7
Figure 2 Example of an OSD Camera System.....	8
Figure 3 Accelerometer Diagram, After [9]	12
Figure 4 Amplitude of a mechanical 2 nd order system with varying damping	14
Figure 5 Parallel Plate Capacitance Sensing [9]	16
Figure 6 Wheatstone Bridge [27].....	17
Figure 7 Capacitance Measurement Block Diagram [28].....	18
Figure 8 MEMS Accelerometer Sensors	18
Figure 9 NanoCore12 Evaluation Kit [23].....	20
Figure 10 Circuit Interface Diagram.....	20
Figure 11 Accelerometer Axis Definition [2].....	22
Figure 12 Accelerometer Tilt Angles After [2]	23
Figure 13 Pythagorean Theorem in Graphical Form	24
Figure 14 Mechanical Gyroscope [30]	26
Figure 15 Spinning “top” Precession Example [29].....	27
Figure 16 Gyroscope Precession [31]	29
Figure 17 Artificial Horizon Indicator [32]	30
Figure 18 Coriolis Effect of the Earth.....	31
Figure 19 Coriolis Effect Diagram [12].....	31
Figure 20 Force caused by the Coriolis Effect.....	34
Figure 21 Coriolis force Right Hand Rule	35
Figure 22 IDG-500 Functional Block Diagram [25]	36
Figure 23 Euler Elementary Rotations.....	38
Figure 24 Prediction and Measurement Distributions [16]	56
Figure 25 New Position Estimate [16].....	56
Figure 26 Kalman Filter Process [18].....	68
Figure 27 Accelerometer Derivative & Angular Rate	85

Figure 28 YZ-Accelerometer Derivative vs. Rate Gyroscope	89
Figure 29 NanoCore12 module MCU.....	92
Figure 30 SparkFun 5 DOF IMU.....	93
Figure 31 MCU and IMU Connection Diagram.....	93
Figure 32 Accelerometer YZ-Angle Derivative vs. Rate Gyroscope	106
Figure 33 Sensor Fusion Results	106

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The availability of low-cost, commercially available, sensors developed using the micro-electromechanical systems (MEMS) manufacturing process, enables advances in inertial sensing. Two of the most prevalent MEMS sensors are the accelerometer and rate gyroscope. The accelerometer measures the specific force (units in of g-force) relative to free fall [1]. The rate gyroscope measures the angular velocity through the Coriolis Effect. The ideal rate gyroscope determines the orientation of an object by integrating the angular rates starting from a known position. Together, the accelerometer provides information needed for the initial starting position while the rate gyroscope updates the orientation by integrating the angular rates.

Current research in object orientation using MEMS inertial sensors addresses the challenges of non-ideal rate gyroscopes. Non-ideal gyroscopes have a non-zero offset term that becomes part of the integration cycle. Since the offset can vary during typical operation, it becomes difficult to compensate.

The work of [4] implements the highly successful Extended Kalman Filter (EKF) state estimator in a manner that combines the rate gyroscope with the accelerometer and GPS-derived accelerations. The conclusion of their work demonstrated the importance of

accurate sensor data from the MEMS sensors and was evident when trying to compensate the accelerometer to reflect only the effects of gravity and not acceleration due to turning.

The significance for the accelerometer to reflect only the effects of gravity come from a technique developed to help stabilize the rate gyroscope. Since the majority of small UAV's remain relatively close to the earth's surface, the gravitational force becomes a constant vector quantity. Typically, in level flight, and at a steady cruising speed, the net forces equal that of gravity. This provides a stable reference for the rate gyroscope (as they tend to drift with time). However, when the net acceleration is not equal to the force of gravity, it becomes more difficult to track the gravitational force; leading to gyroscopic drift. These difficulties (tracking gravitational force) led to the development of my alternative approach. The approach I created also relies on the force of gravity as a measurement input, but does not compute the orientation based on that vector to correct rate gyroscope drift. Instead, I start by assuming that the net acceleration is equal to the gravitational vector and then calculate an equivalent angle. However, instead of trying to compensate for external forces, I take the derivative of the equivalent angular. This produces an angular rate estimate to be compared with the rate gyroscope's data.

The idea behind this method is to determine a correlation between the accelerometer and the rate gyroscope and produces one of three outcomes. The first is when the accelerometer undergoes linear acceleration not associated with any rotations. The correlation between the accelerometer and rate gyroscope will be weak signifying no change in orientation. The second is when the accelerometer measures both rotational

and linear accelerations. This produces a relatively strong correlation between the accelerometer and rate gyroscope (allowing for more deterministic orientation estimations). The last is when the accelerometer measures a net force equal to the gravitational force vector while the rate gyroscope measures small rotational rates. In this condition, the accelerometer and rate gyroscope will produce a weak correlation indicating gyroscopic drift.

The final step of my accelerometer and rate gyroscope correlation calculation is the “weighting” factor. The weighting factor is what allows the orientation estimation portion of the algorithm to rely on either the accelerometer, rate gyroscope, or both. The result is a method, by which the accelerometer is used to:

- a) Significantly reduce the effects of gyroscope drift during periods of stable non-rotating flight.
- b) Increase the accuracy of rotational measurements.
- c) Significantly reduce accelerometer measurement errors caused by vibrational/turbulent forces.

1.2 GOAL OF THE THESIS

The goal of this thesis is to introduce an alternative method of acquiring more accurate data fusion results from an accelerometer and rate gyroscope. In this thesis, I will also address the basic building blocks for object orientation estimation by use of MEMS sensors, rotation matrices, and the Kalman filter through a series of laboratory experiments.

The first experiment provides an intuitive illustration of matrix rotations using Euler angles. At first, this experiment performs a simple single rotation, then transitions to include a mathematical formulation for continuous rotations using the small angle approximation and integration technique. Finally, the matrix rotation experiment ends by providing a means for “Gimbal Lock” prevention.

The second experiment introduces the fundamental concepts of the Kalman filter. The experiment, due to its complexity, was broken down into three main parts. The first part provides the mathematical structure for modeling the dynamics of a simple linear system (or linearized system). The second part builds onto the first by converting a discrete time differential equation into the necessary state-space format used by the Kalman filter algorithm. The final part of the experiment introduces the mathematical formulation of the Kalman filter and applies it to the state-space equation of the linear system from the second exercise.

1.3 STRUCTURE OF THE THESIS

This thesis introduces the concepts necessary for object orientation. When applying these concepts to small UAV's, unique challenges arise. One of the challenges is the use of MEMS based sensors, such as accelerometers and rate gyroscopes. Although these sensors are ideal due to their low-cost and small package size, neither the accelerometer, or rate gyroscope, can be used independently when determining orientation. This is where the concept of sensor fusion is introduced.

The concepts of object orientation and MEMS sensor fusion are introduced as follows:

- **Chapter 2** describes the small UAV and the adaptation from simple line-of-sight radio control, to advance flight capabilities using systems such as the wireless on-screen-display and navigation system. It then goes on to introduce basic autopilot systems and concepts of estimation theory used for object orientation.
- **Chapter 3** introduces the MEMS based accelerometer and rate gyroscope sensors. First, a theory of operation is described for each sensor, followed by the conversion process needed to convert the analog signals into digital signals. Next, an introduction to rotation matrices using Euler angles. The last section is a lab exercise developed to apply the concepts discussed throughout the chapter.
- **Chapter 4** is the core of this thesis. It starts with an introduction to sensor fusion and follows with the derivation of the Kalman filter. Although there are many ways to derive the Kalman filter equations, this chapter uses the derivation of the linear recursive estimator for its foundation. The chapter then ends with a lab exercise developed to introduce the basic linear Kalman filter using Matlab.
- **Chapter 5** is the alternative sensor fusion approach being proposed by this thesis. The alternative sensor fusion approach was based on observations made while working with MEMS accelerometers and rate gyroscopes. The goal of the alternative approach is to fuse the accelerometer and rate gyroscope's data prior to the use of Kalman filtering. The last section is a lab exercise, which is performed using real hardware using MEMS sensors to capture measurement data. The data is then analyzed using the algorithms introduced to obtain the orientation estimates.

CHAPTER 2

UNMANNED AERIAL VEHICLE (UAV)

2.1 INTRODUCTION

An Unmanned Aerial Vehicle (UAV) is any aircraft flown without an actual person on board. For the purpose of the thesis, we will focus our attention on small electric powered, propeller driven aircraft, with a fixed wingspan of less than 2 meters. This particular aircraft can be hand launched and operates from a ground controller within visual range. The payload portion of a UAV includes a variety of systems, such as a radio transceiver, servo actuation controller, and a flight stabilization / autopilot system. The flight stabilization / autopilot systems are further divide down into two primary functions: the inertial navigation sensors (accelerometer, rate gyroscope, magnetometer and global positioning system (GPS)), the main processing unit, and pressure sensors for wind speed and altitude measurements.

Small UAV's may appear to be a modern technical advancement in aerospace science, but actually the early pioneers of these radio controlled (RC) aircraft date back to the late 1940's and early 1950's [5]. Today there are numerous types of radio controlled UAV's on the market, ranging from electric to jet powered in ready-to-fly (RTF) and almost ready-to-fly (ARTF) kits.

RTF and ARTF UAV's can be easily assembled and essentially comprise the same components, except for the fact that the ARTF kits do not include any radio electronics. Both RTF and ARTF aircraft are typically constructed using expanded

polyolefin (EPO) or expanded Polypropylene (EEP) injection molding, and are reinforced using carbon fiber rods for the spars of the wing. This type of construction is very robust, and often referred to as “crash proof” when compared to traditional aircraft made of balsa wood. One of the most popular RTF/ ARTF aircraft utilizing EEP is the Multiplex electric powered “Easy Star.” This aircraft is constructed using proprietary EEP foam called ELAPOR®, which is high-tech particle foam ideal for injection-molded RC components, such as the fuselage and wings [6]. The Easy Star is a very popular UAV due to its unique design, having a pusher prop mounted just behind the main wing. The following picture is the Easy Star aircraft modified with a wireless camera and landing gear.



Figure 1 Multiplex “Easy Star” with wireless camera system

A wireless camera system installed in the front of the aircraft allows the pilot to see through the “eyes” of the aircraft instead of watching it from the ground, creating a first person view (FPV). Controlling a UAV in this way provides a more natural feel (similar to an aircraft flight simulator). Pilots looking for this realistic experience can also incorporate an on-screen display (OSD) camera system designed to provide

information such as navigation, altitude, reference horizon, flight time, and battery life (Figure 2). One of the key benefits to having a FPV OSD system is the ability to fly higher and farther, freeing the aircraft from being within direct line of sight to the person controlling the aircraft.



Figure 2 Example of an OSD Camera System

With the ability to fly higher and farther, the next evolutionary step is to employ an autopilot system. The first autopilot systems were developed to assist pilots in maintaining level flight while traveling long distances. These systems were limited to using mechanical directional gyroscopes and altitude indicator to control the elevator and rudder in level flight. Today, autopilot systems are capable of more than level flight such as coordinated turns and flight path tracking. In essence, a modern autopilot system can perform all aspects of flight from the moment the aircraft lifts off the runway until the aircraft touches back down onto the runway.

The autopilot system is considered to have two independent systems connected together to work as one [7]. The first system is the attitude heading and reference system

(AHRS) with the second being the global positioning system (GPS). The AHRS system provides information relating the orientation of the aircraft to the inertial frame while the GPS relates the aircraft's position relative to Earth. The information generated from the two inputs feed into the flight director (FD) which has the navigational waypoints stored into memory. The FD is responsible for processing the input data, relating the input data to the desired course, and then calculates the control outputs necessary to achieve a certain heading. The outputs of the autopilot system are control signals, which move the control surfaces and throttle controls as necessary. Deflection of the control surfaces will cause a change in the orientation, speed, or both.

Since small UAV's lack the payload capacity for heavier mechanical sensors, lightweight low-cost MEMS sensors provide a great alternative solution. However, by using low-cost MEMS sensor there is a significant increase in the complexity of the AHRS system because each sensor lacks the ability to output reliable data for orientation estimation. For instance, a low cost MEMS gyroscope will suffer from gyro bias errors, which, if not dealt with, will cause integration errors to build over time. The errors can be severe enough to cause the aircraft to rotate about its axis several degrees per second. MEMS based accelerometers used on small aircraft can suffer from high noise levels due to vibrational forces caused by the propeller.

Estimation theory is a form of statistical analysis used to estimate the value of a parameter (assuming the measurement data has normally distributed error sources). The use of estimation has many advantages over digital low-pass filtering. For instance, depending on the sample size, a low pass filter can respond quickly to changes in input

signals or respond very slowly (several seconds). Sensors with large sensitivity to noise typically result in low pass filters with relatively large sample sizes, and thus slow output responses.

The significant impact of time delays in the loop response of a system can be demonstrated through a cruise control example. To begin, let us assume that a cruise control system is designed to maintain a constant speed for all input conditions. One method is to apply a low-pass filter using a large sample size. In this case, small changes in the road will not immediately affect the output response of the cruise control. Instead, the input response will be averaged out over a given amount of time. The drawback to this system is apparent if the car begins to travel up a long hill. Instead of maintaining a constant speed, the car will begin slowing down until the engine starts increasing speed. The engine will continue increasing speed as the time along the hill increases. However, when the car starts traveling downhill, the engine will continue averaging data from when it was going uphill resulting in a continued increase in speed. At this point, the car will be racing downhill until the averaged sample size catches up, slowing the engine speed. Now, let us assume that the cruise control has a small sample size and can make quick changes to the output. This will provide a more appropriate response for the road environment but may become excessive in some situations.

In the above cruise control example, we see how excessive time delays can affect the response of the engine's speed to changes in road conditions. If we instead used estimation theory, we would first describe the state of the car using the classic equations of motion and then predict where the car might be during the next time step. Since the

car will travel on various road conditions, we can use a sensor to monitor the acceleration of the car as it travels down the road along with a global positioning system (GPS), helping us determine the car's location. The sensor data from the acceleration will be used to update the equations of motion to reflect changes as the car travels and the data from the GPS will be used to compare against the predicted location. The errors from the predicted location and the location of the GPS will be used to update the new predictions for the next data sample. The above process of predicting the location, and updating with the GPS, is then repeated continuously as the car travels down the road. The benefit of this process is the ability to perform quick output changes with long-term accuracy.

2.2 SUMMARY

In this chapter, we defined a small UAV as being an electrically powered, propeller driven aircraft, with a fixed wingspan of less than 2 meters. This particular aircraft can be hand launched and operates from a ground controller within visual range. The payload portion of a UAV includes a variety of systems, such as a radio transceiver, servo actuation controller, and a flight stabilization / autopilot system. One particular aspect of the flight stabilization / autopilot system that we will focus our attention on is the inertial sensors and the orientation estimation algorithms.

The inertial sensors we are interested in is a MEMS based accelerometer and rate gyroscope. The two sensors are used together through a sensor fusion process in an effort to overcome each sensor's weakness. The method used by most is the Extended Kalman Filter (nonlinear version of the Kalman filter) which is further discussed in Chapter 4.

CHAPTER 3

AN INTRODUCTION TO MEMS INERTIAL SENSORS

3.1 THE ACCELEROMETER

An accelerometer is a sensor used to measure the specific force (in units of g force) of an object relative to free fall. The accelerometer accomplishes this by measuring a force proportional to the rate of acceleration as provided by Isaac Newton's second law of motion, " $F = ma$." Referencing [9], the most intuitive model used to measure a force proportional to acceleration is to analyze the spring-mass-damper system shown below (Figure 3). This system is often chosen because the acceleration can be solved in terms of the spring constant and displacement.

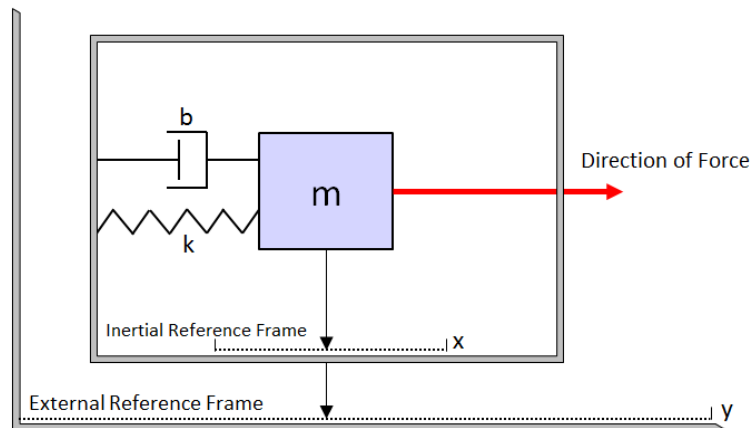


Figure 3 Accelerometer Diagram, After [9]

For the system to work we must be able to measure the relative position between a movable mass and a fixed frame. By doing this we can define the displacement x ,

relative to the frame, as being equal to $(z-y)$. This now gives us the ability to define a dynamic equation as follows,

$$k(z - y) + b(\dot{z} - \dot{y}) + m\ddot{z} = 0 \quad (3.1)$$

Realizing that $(z - y) = x$ we come up with,

$$\ddot{x} + \frac{b}{m}\dot{x} + \frac{k}{m}x = -\ddot{y} = -a(t) \quad (3.2)$$

In this representation, a force in the positive “x” direction will cause the dampener to produce a force in the negative “x” direction equal to the product of the dampener constant, and the rate of change denoted as “ $b\dot{x}$ ”, where \dot{x} represents velocity. A second force is also created in the negative “x” direction equal to the product of the spring constant and the displacement of the spring denoted by “ kx ”.

Since we want to express the acceleration in terms of the spring constant and displacement, we need to further simplify the equation by analyzing the frequency response characteristics of the system [8]. To find the frequency response we’ll solve for the transfer function by taking the Laplace transform of $x(t)$ and $a(t)$ and then evaluate

$$\left(\frac{Lx(s)}{La(s)}\right).$$

$$s^2X(s) - sx(0) - \dot{x}(0) + \frac{b}{m}sX(s) - x(0) + \frac{k}{m}X(s) = -A(s) \quad (3.3)$$

Here we will set the initial conditions to zero, ($\dot{x}(0) = 0$ and $x(0) = 0$) yielding the following simplified equation.

$$s^2X(s) + \frac{b}{m}sX(s) + \frac{k}{m}X(s) = -A(s) \quad (3.4)$$

$$H(s) = \frac{X(s)}{A(s)} = -\frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}} \quad (3.5)$$

An example for the frequency response of the mass-spring-damper system is shown in Figure 4.

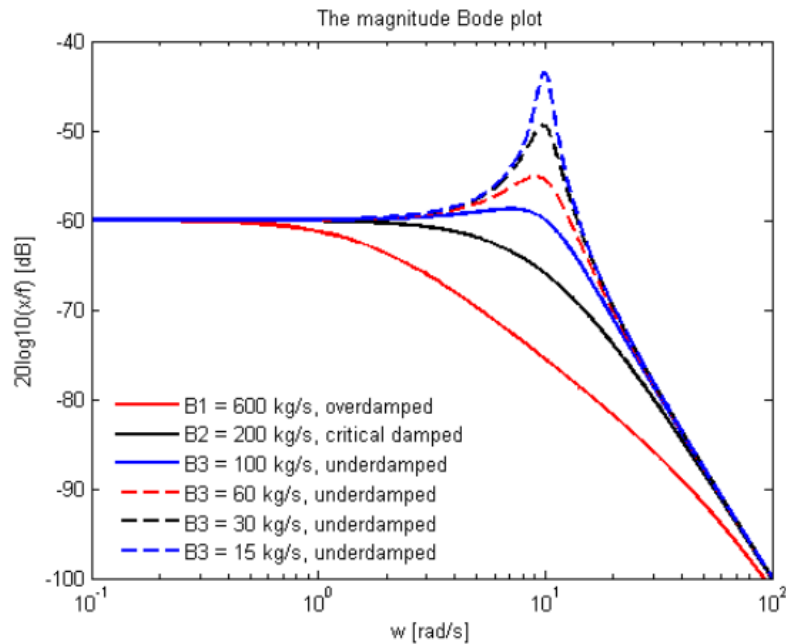


Figure 4 Amplitude of a mechanical 2nd order system with varying damping coefficient B [8]

The frequency response is characteristic to that of a low-pass filter where a resonance is depicted by a sudden rise of the gain at the cut-off frequency. To avoid resonance, an accelerometer is designed to operate in the flat region to the left of the resonant frequency. If we call the knee of the frequency response ω_n , and then constrain

the system such that $\omega_o \ll \omega_n$ then the system's magnitude will not have any frequency dependence. Therefore, the system can be simplified to;

$$H(0) \approx H(s) = -\frac{1}{0 + 0 + \frac{k}{m}} = -\frac{m}{k} \quad (3.6)$$

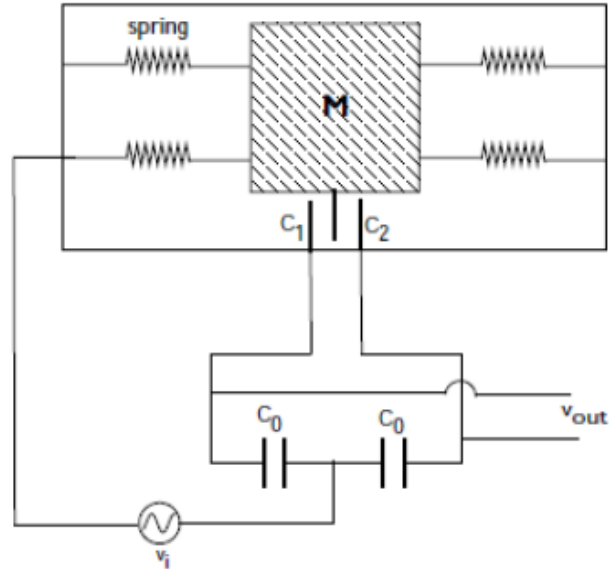
At this point, we can now transform the equation back into the time domain resulting in the desired form where the acceleration is a function of the mass, spring constant, and displacement.

$$a(t) = -\frac{m}{k}x(t) \quad (3.7)$$

Now that the accelerometer is a simplified function of displacement $x(t)$, we need to find a way to measure that displacement. In most low-cost MEMs accelerometers, the capacitive sensing approach is used due to a simpler manufactured design. In this topology, the geometry of the capacitor plate changes when undergoing acceleration. The equation for the parallel-plate capacitor is presented as;

$$C_0 = \epsilon_o \epsilon \frac{A}{x} = \epsilon_r \frac{A}{x} \quad (3.8)$$

where $\epsilon_r = \epsilon_o \epsilon$ is the relative permittivity of the dielectric, "A" is the area of the electrode and "x" is the distance between the two plates. As the proof mass undergoes acceleration, the mass will move relative to the frame, causing a change in displacement and therefore a change in capacitance.



The common plate is connected to the voltage supply through the mass and the spring.

Figure 5 Parallel Plate Capacitance Sensing [9]

The capacitance for the parallel plate sensing topology (Figure 5) is calculated by;

$$C_1 = \epsilon_r \frac{A}{x_o + x} = \epsilon_r \frac{A}{x_o(1 + \frac{x}{x_o})} = \frac{C_0}{1 + \delta} \quad (3.9)$$

$$C_2 = \epsilon_r \frac{A}{x_o - x} = \epsilon_r \frac{A}{x_o(1 - \frac{x}{x_o})} = \frac{C_0}{1 - \delta} \quad (3.10)$$

where $\delta = \frac{x}{x_o}$,

For this example, the Wheatstone bridge circuit is used to represent the parallel plate sensing topology. Referencing the diagram below (Figure 6), we will set C_1 as R_1 , C_2 as R_3 , and both R_2 and R_4 as C_0 . In this example, we assume that the circuit is balanced when no forces are present.

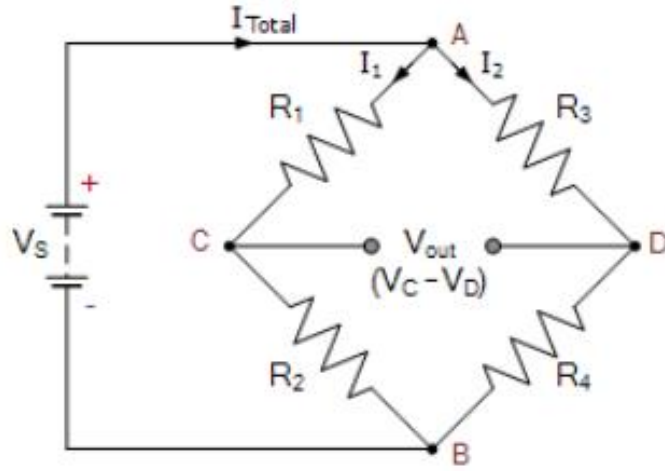


Figure 6 Wheatstone Bridge [27]

If we set $\delta \ll 1$, then the input to output equation becomes;

$$V_{out} = \frac{1}{2} \delta V_{in} \quad (3.11)$$

The equation for the acceleration now becomes a function of voltage as shown below;

$$a(t) = -\frac{m}{k} x_o \delta \quad (3.12)$$

$$a(t) = -\frac{m}{k} x_o \left(2 \frac{V_{out}}{V_{in}} \right) = -2x_o \frac{m}{k} A \quad (3.13)$$

where “A” is the voltage gain $\left(\frac{V_{out}}{V_{in}} \right)$.

This equation represents the acceleration as a function of time, equal to the displacement given as a function of voltage.

Other capacitive sensing topologies, such as those incorporated into the ADXL05 by Analog Devices, use an oscillator circuit and demodulator to measure capacitance.

The oscillator's function is to excite the capacitors where a change in capacitance is detected by the demodulator. The output of the demodulator results in an output voltage proportional to the change in capacitance (Figure 7).

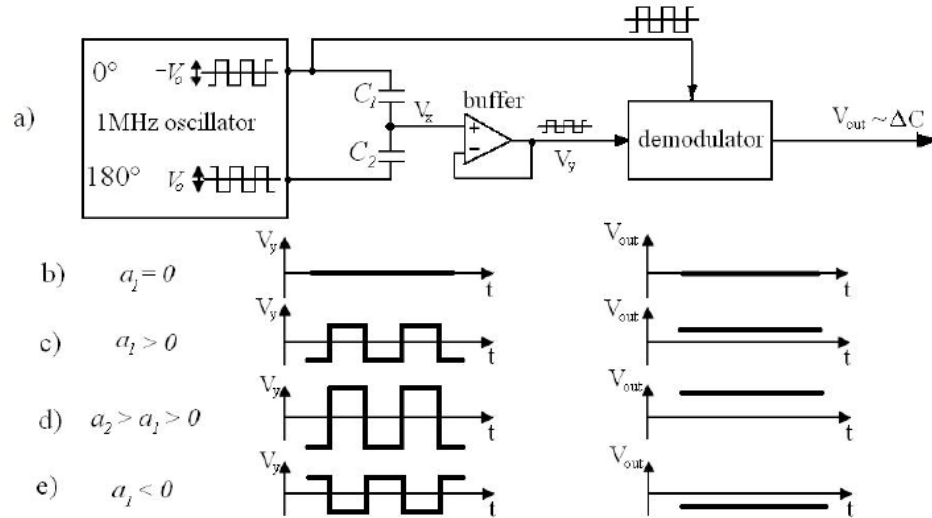


Figure 7 Capacitance Measurement Block Diagram [28]

3.1.1 ACCELEROMETER SENSOR MEASUREMENTS

Accelerometers are manufactured by a variety of vendors such as InvenSense, Analog Devices, and STMicroelectronics (Figure 8). These sensors are constructed on silicon wafers and then wire bonded to the signal conditioning circuitry producing a single sensor package solution.

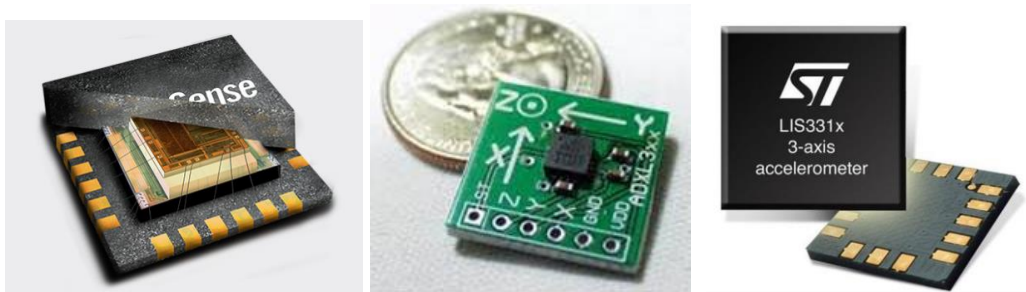


Figure 8 MEMS Accelerometer Sensors

The Analog Devices' ADXL335 accelerometer was chosen for this thesis because the output signals are a ratio metric analog voltage proportional to the measured acceleration. The analog signals were desired over the digital output version (Serial Communication Interface) due to the lack of SCI channels. In addition, this sensor was available with an adapter board allowing for instant experimentation. Table 1 (below) lists the functional characteristics of the ADXL335 accelerometer [24].

Table 1 ADXL335 Accelerometer Datasheet [24]

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		g
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity ¹			±1		%
SENSITIVITY (RATIOMETRIC)²	Each axis				
Sensitivity at X _{OUT} , Y _{OUT} , Z _{OUT}	V _S = 3 V	270	300	330	mV/g
Sensitivity Change Due to Temperature ³	V _S = 3 V		±0.01		%/°C
ZERO g BIAS LEVEL (RATIOMETRIC)					
0 g Voltage at X _{OUT} , Y _{OUT}	V _S = 3 V	1.35	1.5	1.65	V
0 g Voltage at Z _{OUT}	V _S = 3 V	1.2	1.5	1.8	V
0 g Offset vs. Temperature			±1		mg/°C
NOISE PERFORMANCE					
Noise Density X _{OUT} , Y _{OUT}			150		µg/√Hz rms
Noise Density Z _{OUT}			300		µg/√Hz rms
FREQUENCY RESPONSE⁴					
Bandwidth X _{OUT} , Y _{OUT} ⁵	No external filter		1600		Hz
Bandwidth Z _{OUT} ⁵	No external filter		550		Hz
R _{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz

$$Bandwidth = \frac{1}{2\pi \cdot 32k\Omega \cdot C}$$

Experimentation with the ADXL335 sensor was performed using the Nanocore12 microcontroller (MCU) developed by Technological Arts [23]. This particular MCU was chosen due to its small size, low cost, and easy to use Integrated Development Environment (IDE) software.



Figure 9 NanoCore12 Evaluation Kit [23]

The circuit interface diagram for the NanoCore12 and ADXL335 is shown in Figure 10.

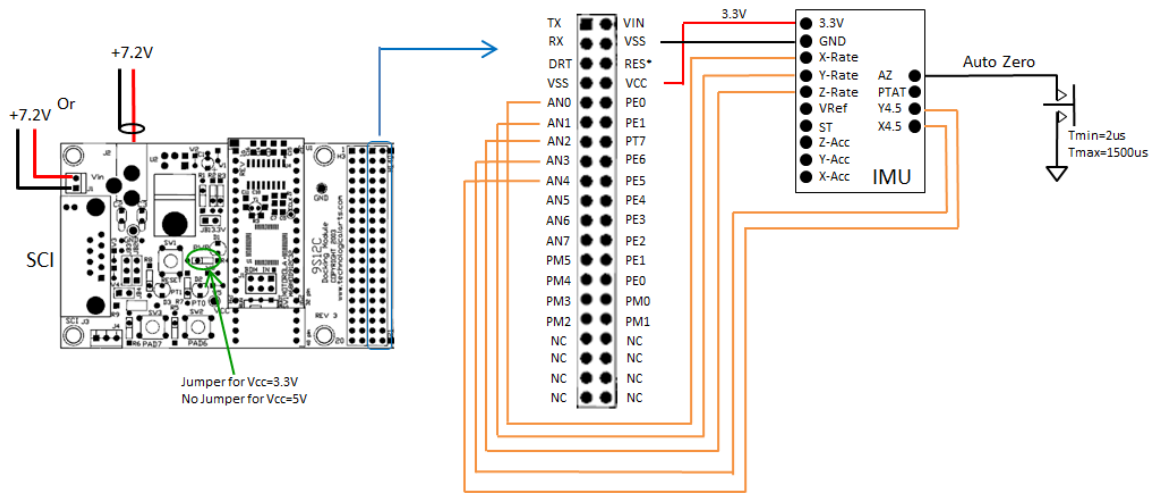


Figure 10 Circuit Interface Diagram

3.1.2 ADC CONVERSION OVERVIEW

The output of the accelerometer's analog signal is measured in units of mV/g, while the digital output signal from the MCU is in units of g's (one times the force of gravity). In order to convert the analog signal into a digital signal, an analog-to-digital converter

(ADC) is used. The ADC's function is to sample the voltage at specific intervals to produce a digital representation of the change in sensor output. To begin, we start with an ADC conversion equation, which converts the ADC result into units of volts.

$$\text{Analog Voltage Measured} = \frac{ADC_{Results}(ADC_{Voltage Scale})}{ADC_{Resolution}} \quad (3.14)$$

Next, we remove any bias offset errors from the sensor data.

$$\text{Analog Voltage Measured} = \frac{ADC_{Results}(ADC_{Voltage Scale})}{ADC_{Resolution}} - (Zero_g \text{ Bias}) \quad (3.15)$$

Lastly, we convert to the digital representation of the output signal into the desired units of force. The result is then divided by the sensitivity parameter outlined in the datasheet.

$$\text{Force} = \frac{\left(\frac{ADC_{Results}(ADC_{Voltage Scale})}{ADC_{Resolution}}\right) - (Zero_g \text{ Bias})}{Device \text{ Sensitivity}} \quad (\text{units in } g') \quad (3.16)$$

Since we are typically interested in more than one accelerometer measurement, we can repeat the ADC conversion for each input signal.

$$Accel_x = \frac{\left(\frac{ADC_{R_x}(ADC_{Voltage Scale})}{ADC_{Resolution}}\right) - (Zero_g \text{ Bias})}{Device \text{ Sensitivity}} + Null_{offset}$$

$$Accel_y = \frac{\left(\frac{ADC_{R_y}(ADC_{Voltage Scale})}{ADC_{Resolution}}\right) - (Zero_g \text{ Bias})}{Device \text{ Sensitivity}} + Null_{offset}$$

$$Accel_z = \frac{\left(\frac{ADC_{R_z}(ADC_{Voltage Scale})}{ADC_{Resolution}}\right) - (Zero_g \text{ Bias})}{Device \text{ Sensitivity}} + Null_{offset}$$

Example:

Determine the force acting on the accelerometer when the input ADC converted voltage signal is 508 (decimal) given the following parameters: ADC = 10-bit resolution, $V_{ADC} = V_s = 3V$, Zero g Bias Level = 1.35V, Sensitivity = 300mV/g.

$$Force = \frac{\left(\frac{508 (3.0v)}{1023}\right) - (1.35v)}{0.300 \text{ g/v}} = 0.96g's$$

3.1.3 COMPUTING ANGLES FROM ACCELEROMETERS

Accelerometers are designed with their sensing elements “modes” positioned orthogonal to one another. This arrangement allows measurements to be taken along the X, Y, and Z sensing axis to determine the orientation of the accelerometer. To standardize components produced by different manufacturers, the typical convention is to have the Z-axis point upward as shown in Figure 11.

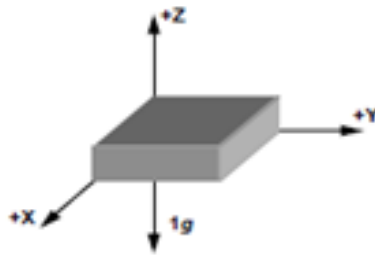


Figure 11 Accelerometer Axis Definition [2]

When placed on a level surface, the Z-axis produces a +1g measurement while the X and Y-axis produce a 0g measurement. Now, if the accelerometer were to rotate 90 degrees about the Y-axis, then the Y and Z-axis will produce a 0g measurement and the

X-axis will output a $\pm 1g$ measurement. Similarly, a rotation about the X-axis will cause the X and Z-axis to produce a $0g$ measurement while the Y-axis will have a $\pm 1g$ measurement. If we were to look at just a two axis accelerometer (no Z-axis) and rotate the sensor about either one of the sensing axes by 180 degrees, then the sensors would output $0g$ measurements even though the sensor package is upside-down. Therefore, a dual axis accelerometer is not suitable for orientation applications.

Although two-dimensional accelerometers are not well suited for orientation measurements, we can still use them to determine tilt angles as shown in Figure 12.

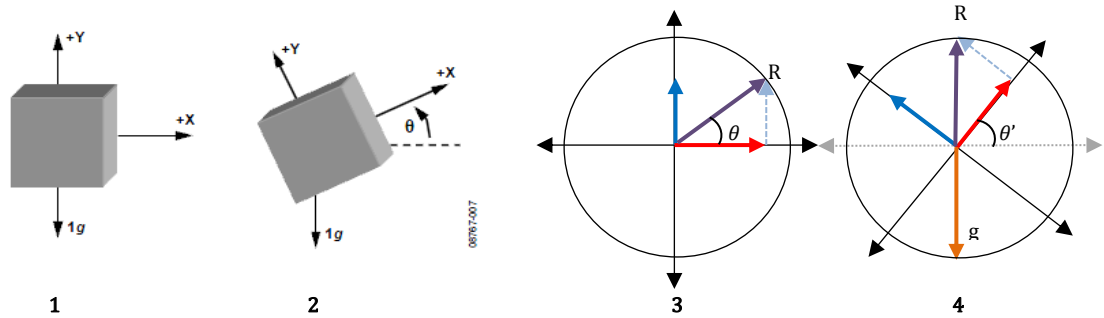


Figure 12 Accelerometer Tilt Angles After [2]

Now, imagine the accelerometer positioned such that the Y-axis was pointing upwards with the X-axis pointing to the right. If a counter clockwise (CCW) rotation was about the Z-axis by an angle of θ degrees then the measurement vectors of the accelerometer will also result in an angle of θ as shown in Figure 12 drawing No. 2. The angle theta (θ) can then be determined using the simple trigonometric equation,

$$\theta = \tan^{-1} \left(\frac{Ay}{Ax} \right) \quad (3.17)$$

where,

$A_y = Y$ Axis Accelerometer Output

$A_x = X$ Axis Accelerometer Output

Following the work provided in [2], a three dimensional orientation follows the same principle by utilizing Pythagoras' theorem:

$$C = \sqrt{x^2 + y^2}$$

$$Z = Z$$

$$\therefore \theta_z = \tan^{-1} \left(\frac{\sqrt{A_x^2 + A_y^2}}{A_z} \right) \quad (\text{Radians}) \quad (3.18)$$

The Pythagorean Theorem can be expressed graphically as shown in Figure 13.

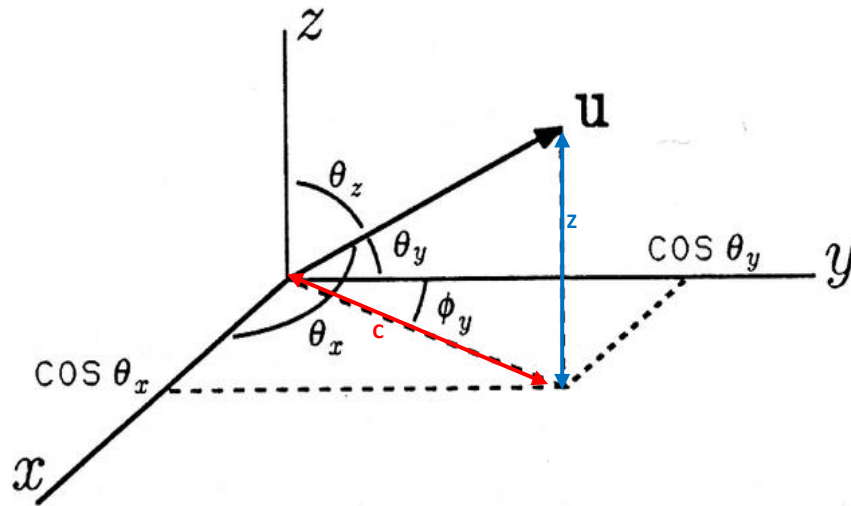


Figure 13 Pythagorean Theorem in Graphical Form

Following the same mathematical process yields the angles for the remaining two axes.

Note: The two remaining angles may not be obvious; however, if you flip the paper on its side you may be able to better visualize those angles.

$$\theta_x = \tan^{-1}\left(\frac{A_{x,out}}{\sqrt{A_{y,out}^2 + A_{z,out}^2}}\right) \quad (\text{Radians}) \quad (3.19)$$

$$\theta_y = \tan^{-1}\left(\frac{A_{y,out}}{\sqrt{A_{x,out}^2 + A_{z,out}^2}}\right) \quad (\text{Radians}) \quad (3.20)$$

At this point it's important to note that the atan2 function may instead be used in place of the arctangent function allowing the angle θ to span the interval $(-\pi, \pi]$. The angle can then be found using the atan2 with two input arguments (Y and X instead of Y/X) as follows,

$$\theta_x = \text{atan2}(-A_z, A_x) \quad (\text{Radians})$$

$$\theta_y = \text{atan2}(-A_z, A_y) \quad (\text{Radians})$$

The relationship below defines the criteria for the atan2 function:

$$\text{atan2}(y, x) = \begin{cases} \text{arctan}\left(\frac{y}{x}\right) & x > 0 \\ \text{arctan}\left(\frac{y}{x}\right) + \pi & y \geq 0, x < 0 \\ \text{arctan}\left(\frac{y}{x}\right) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ \text{undefined} & y = 0, x = 0 \end{cases} \quad (3.21)$$

Note: Since the accelerometer measure a positive 1g that actually points down, we need to set the A_z terms to a positive quantity. Ex: $\theta_x = \text{atan2}(A_{z_{Accel}}, A_x)$ and $\theta_y = \text{atan2}(A_{z_{Accel}}, A_y)$.

3.2 THE RATE GYROSCOPE

A traditional gyroscope is a mechanical disk designed to rotate at a high angular velocity. The properties of rotational inertia and angular momentum are utilized to maintain rigidity in the inertial frame. We can apply this concept to a rotating disk mounted on gimbals such that it can freely rotate about any of the three principle axes. By doing so, the disk is allowed to remain fixed in space due to the conservation of angular momentum. The gimbaled frame is then mounted to the instrument platform as shown in Figure 14.

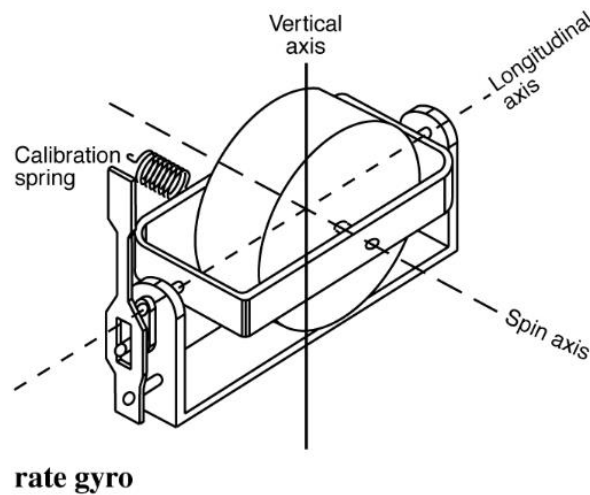


Figure 14 Mechanical Gyroscope [30]

When the gyroscope is rotated about its vertical axis, a torque occurs causing the disk to rotate about its longitudinal axis. This is called “precession torque” and is directly proportional to the applied rotational velocity.

Gyroscopic precession described best by analyzing the behavior of a “top” spinning on a flat surface. If the top experiences an external force, such as the force of gravity, it will begin to rotate about the vertical axis as shown in Figure 15.

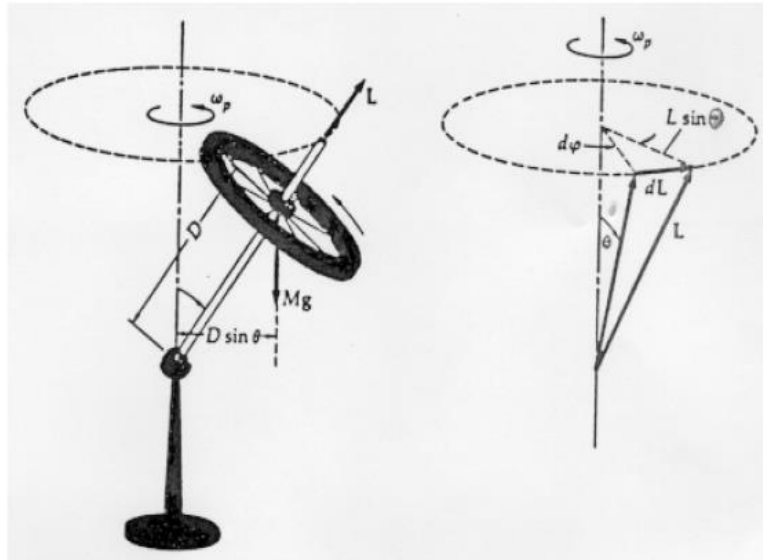


Figure 15 Spinning “top” Precession Example [29]

Using Figure 15 as our reference, we can derive the equation for precession angular velocity in terms of the force of gravity (Mg), the length of the rod (D), and angular momentum (L) [11]. First, we need to find the equation for torque as follows,

$$\tau = \frac{dL}{dt} = (Mg)D\sin\theta \quad (3.22)$$

Next, we will want to define precession angular velocity as;

$$\omega_p = \frac{d\phi}{dt} \quad (3.23)$$

where $d\phi$ is;

$$d\phi = \frac{dL}{L\sin\theta} \quad (3.24)$$

If we substitute the value for $d\phi$ into ω_p we get the equation for torque divided by $L\sin\theta$.

Further simplification yields the desired result in terms of force, distance, and angular momentum.

$$\omega_p = \frac{d\phi}{dt} = \frac{dL}{dt(L\sin\theta)} = \frac{\tau}{L\sin\theta} = \frac{(Mg)D\sin\theta}{L\sin\theta} = \frac{(Mg)D}{I\omega} \quad (3.25)$$

The precession torque acts as a force rotating the gyroscope about its longitudinal axis. Using Figure 16 as a reference, if torque is applied to the spinning disk causing it to rotate about the vertical (DE) axis (in the direction shown by ω), then the precession torque vector \mathbf{P} will act orthogonal to the angular momentum vector (BA).

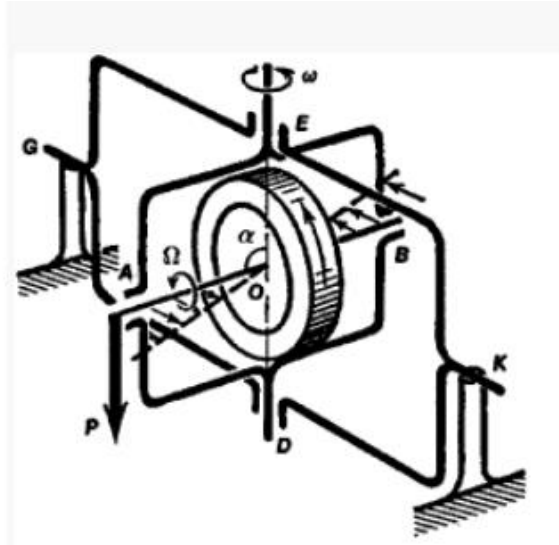


Figure 16 Gyroscope Precession [31]

The mechanical gyroscope is found in many avionic instruments, such as the gyrocompass and turn coordinator. The precession torque of the gyroscope is used either to dampen fluctuations (typically associated in magnetic compasses) or to indicate the rate-of-turn for the aircraft. Other instruments, such as the artificial horizon, have a gyro mounted on gimbals so that it can rotate freely about any axis. This configuration takes advantage of the conservation of angular momentum, which causes the gyro to remain fixed within the inertial reference frame.

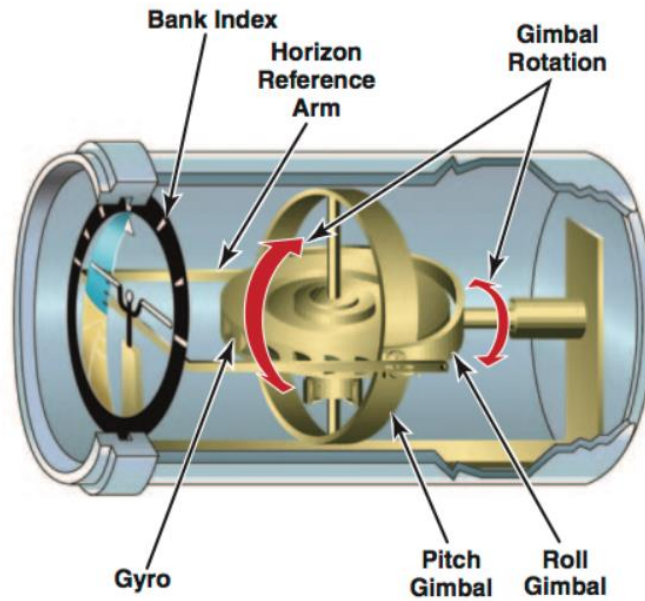


Figure 17 Artificial Horizon Indicator [32]

An alternative to the classic mechanical gyroscope is a MEMS rate gyroscope, which is based on the Coriolis Effect. The Coriolis Effect is considered a fictitious force arising from the choice of a rotating framework of reference. For example, a small commuter airplane leaving Burlington Vermont, heading to Orlando Florida, will take-off leaving the Earth's reference frame. Assuming a perfectly straight flight path, the airplane will drift off course and eventually end up somewhere over the Gulf of Mexico.



Figure 18 Coriolis Effect of the Earth

Keeping the Coriolis Effect in mind, the reason for ending up over the Gulf is that as the airplane traveled south in the inertial frame, the earth continued to spin relative to the inertial frame. Using Figure 19, to explain the Coriolis Effect more thoroughly, we can represent the earth using the coordinates X' , Y' and Z' and the airplane as x , y and z .

The vector “R” shows the airplane as it travels relative to a fixed reference point on earth.

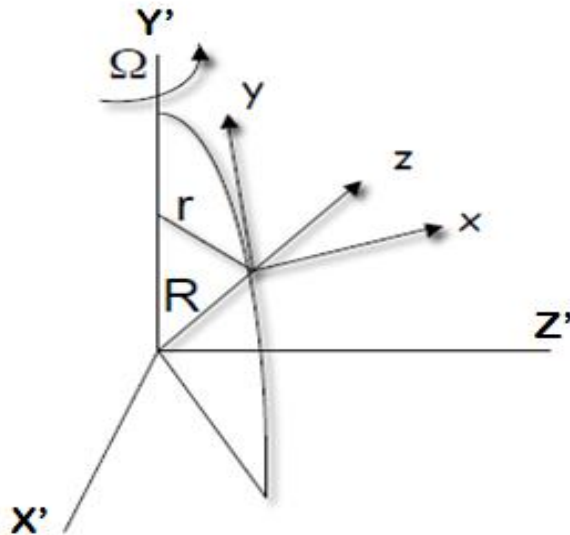


Figure 19 Coriolis Effect Diagram [12]

The equation to describe the force due to the Coriolis Effect is determined by applying rotational kinematics and the vector cross product [12]. To begin we first define a vector “R” as;

$$R = \hat{i}x + \hat{j}y + \hat{k}z \quad (3.26)$$

Next we taking the derivative of each component with respect to time,

$$\frac{dR}{dt} = \underbrace{\left(\hat{i} \frac{dx}{dt} + \hat{j} \frac{dy}{dt} + \hat{k} \frac{dz}{dt} \right)}_{\text{Rate of change relative to the fixed reference point on earth}} + \underbrace{\left(\frac{d\hat{i}}{dt}x + \frac{d\hat{j}}{dt}y + \frac{d\hat{k}}{dt}z \right)}_{\text{Rate of change of the fixed reference relative to the earth's center}} \quad (3.27)$$

Using the notation,

$$\dot{R} = \left(\hat{i} \frac{dx}{dt} + \hat{j} \frac{dy}{dt} + \hat{k} \frac{dz}{dt} \right) \quad (3.28)$$

\dot{R} is set equal to the rate of change relative to a fixed reference point on earth. The term $\left(\frac{d\hat{i}}{dt}x + \frac{d\hat{j}}{dt}y + \frac{d\hat{k}}{dt}z \right)$ represents the rate of change of the fixed reference frame relative to the earth’s center. This can be expressed more compactly as $\Omega \times R$. Note: $\Omega \times R$ is another way of saying that the linear velocity due to rotation is simply the radius multiplied by angular velocity. Substituting in \dot{R} and $\Omega \times R$ yields;

$$\left(\frac{dR}{dt} \right)_{\text{Space}} = \dot{R}_{\text{Earth}} + (\Omega \times R) \quad (3.29)$$

Next, we would like to determine an expression for the observed acceleration and the acceleration due to the rotation of the earth. Let us define the operator:

$$\frac{d(\)}{dt} = (\dot{\ }) + \Omega \times (\) \quad (3.30)$$

Plugging $\left(\frac{dR}{dt}\right)_{Space}$ into the parentheses results in the following,

$$\begin{aligned} \frac{d(\)}{dt} &= (\dot{\ }) + \Omega \times (\) \\ \frac{d}{dt} \left(\frac{R}{dt}\right)_{Space} &= \left(\dot{R} + (\dot{\Omega} \times R)\right) + \Omega \times \left(\dot{R} + (\Omega \times R)\right) \\ \frac{d}{dt} \left(\frac{R}{dt}\right) &= \left(\ddot{R} + (\dot{\Omega} \times R) + (\Omega \times \dot{R})\right) + (\Omega \times \dot{R}) + [\Omega \times (\Omega \times R)] \\ \frac{d}{dt} \left(\frac{R}{dt}\right) &= \left(\ddot{R} + (\dot{\Omega} \times R) + (\Omega \times \dot{R})\right) + (\Omega \times \dot{R}) + [\Omega \times (\Omega \times R)] \end{aligned} \quad (3.31)$$

This equation can then be re-arranged into the following representation;

$$A_{Space} = a_{Earth} + \underbrace{2(\Omega \times v_{Earth})}_{\text{Coriolis Term}} + \underbrace{[\Omega \times (\Omega \times R)]}_{\text{Centripetal Term}} + \underbrace{(\dot{\Omega} \times R)}_{\text{Change in Rotation Rate}} \quad (3.32)$$

Since we are trying to solve for the Coriolis force acting on an object, the acceleration of the earth, centripetal acceleration, and the change in rotation rate terms can be ignored. Therefore, the force equation simplifies to;

$$A_{Space} = \frac{F_{Space}}{M} = 2(\Omega \times v_{Earth}) \quad (3.33)$$

$$F_{Space} = 2M(\Omega \times v_{Earth}) \quad (3.34)$$

To a better understand Coriolis Effect; imagine a ball at the center of a spinning disk (Figure 20). As the disk spins, the ball moves from its original center position towards the outer edge. If you look at the ball from the inertial reference frame, as shown

in picture A, it will appear as though the ball travels in a straight line. If, on the other hand, you were to look at the ball from the disk's reference frame, the ball would travel in a curved trajectory as shown in picture B.

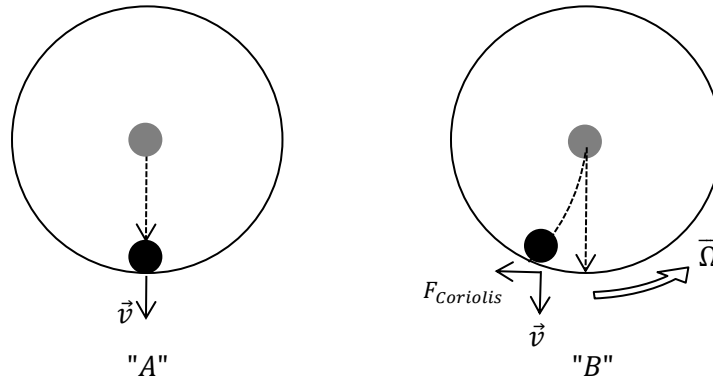


Figure 20 Force caused by the Coriolis Effect

The force caused by the Coriolis Effect appears to move the ball along the curved trajectory and is the essential operating principle of the MEMS vibrating structure rate gyroscope. Another way to look at Coriolis force would be to use the right hand rule. Referencing Figure 21, if you point your thumb in the direction of the spin axis, your index finger in the direction of the drive mode (velocity) then your middle finger will be pointing in the direction opposite the Coriolis force.

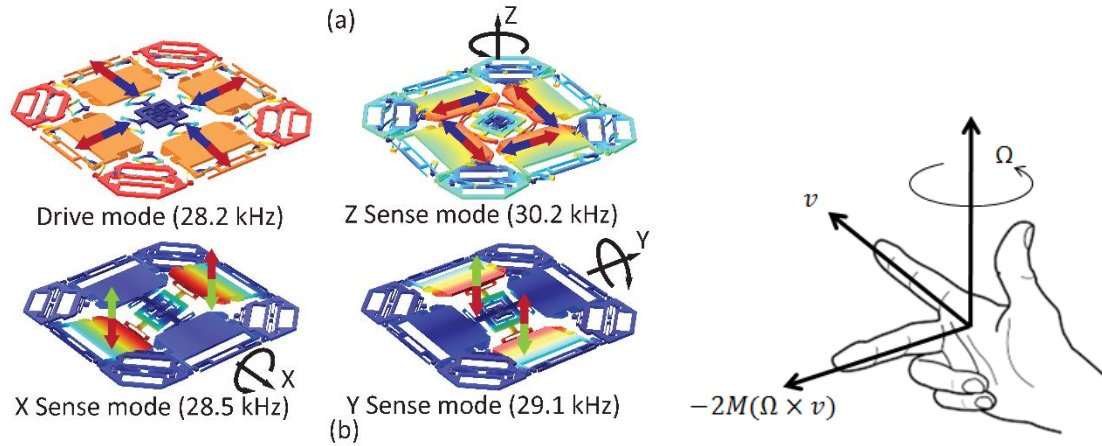


Figure 21 Coriolis force Right Hand Rule

3.2.1 RATE GYROSCOPE MEMS SENSOR

The rate gyroscope used for this thesis was the InvenSense IDG-500 [25]. This specific rate gyro uses a proprietary MEMS technology with vertically driven dual-mass bulk silicon configurations that sense the rate of rotation about the X- and Y-axis (in-plane sensing). The benefit of this dual-axis gyro is the guaranteed-by-design vibration rejection and high cross-axis isolation.

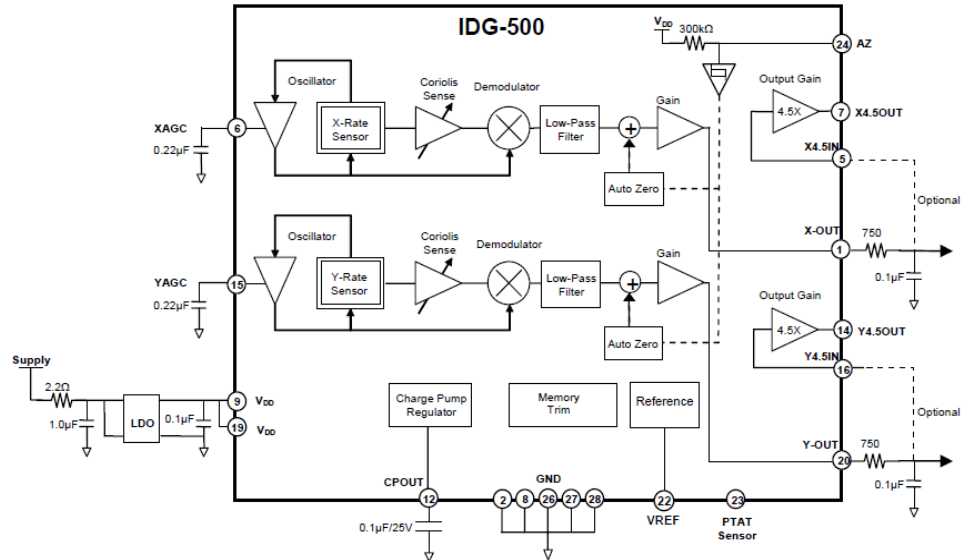


Figure 22 IDG-500 Functional Block Diagram [25]

Performance Summary:

Table 2 InvenSense IDG-500 Datasheet [25]

SPECIFICATIONS

All parameters specified are @ VDD = 3.0 V and Ta = 25°C. External LPF @ 2kHz. All specifications apply to both axes.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	
SENSITIVITY Full-Scale Range	At X-OUT and Y-OUT		±500		°/s	
	At X4.5Out and Y4.5Out		±110		°/s	
	Sensitivity	At X-OUT and Y-OUT		2.0		mV/°/s
		At X4.5Out and Y4.5Out		9.1		mV/°/s
		Initial Calibration Tolerance	At X-OUT and Y-OUT		±6	%
Over Specified Temperature	At X-OUT and Y-OUT		±10	%		
Nonlinearity	At X-OUT and Y-OUT, Best Fit Straight Line		<1		% of FS	
Cross-axis Sensitivity			±1		%	
REFERENCE Voltage (VREF)			1.35		V	
	Tolerance		±50		mV	
	Load Drive		100		μA	
	Capacitive Load Drive	Load directly connected to VREF		100		pF
	Power Supply Rejection	VDD= 2.7V to 3.3V		1		mV/V
Over Specified Temperature			±5		mV	
ZERO-RATE OUTPUT Static Output (Bias)	Factory Set		1.35		V	
	Initial Calibration Tolerance	Relative to VREF	With Auto Zero	±20		mV
		Without Auto Zero		±250		mV
Over Specified Temperature	Relative to VREF	Without Auto Zero	±50		mV	
Power Supply Sensitivity	@ 50 Hz		10		°/sec/V	

3.2.2 ADC CONVERSION OVERVIEW

The output of the rate gyro is an analog signal measured in units of mV/°/s. However, we need to convert the signal into units of °/s. The analog to digital conversion used for the rate gyroscope is the same method used for the accelerometer seen earlier in Sec. 3.1.2 (ADC Conversion Overview).

$$\text{Angular Rate} = \frac{\left(\frac{ADC_{Resuls}(ADC_{Voltage\ Scale})}{ADC_{Resolution}} \right) - (Zero_g\ Bias)}{Device\ Sensitivity} \quad (3.35)$$

Example:

Determine the force acting on the accelerometer when the input ADC converted voltage signal is 508 (decimal) given the following parameters: 10-bit ADC, $V_{ADC} = V_{dd} = 3V$, Zero Rate Output Level = 1.35V, Sensitivity = 9.1 mV/°/s.

$$\text{Result} = \frac{\left(\frac{508 (3.0v)}{1023} \right) - (1.35v)}{0.0091 \text{ v}/\text{°}/\text{sec}} = 15.35 \text{ °}/\text{sec}$$

3.3 EULER ANGLES & ELEMENTARY ROTATIONS

The orientation of an object can be described using a sequence of three elementary rotations (one about each principle axis). The body is fixed to one coordinate system, which is defined as $x_{Body}, y_{Body}, z_{Body}$, with each elementary rotation occurring about the body's axis. Initially, the body is aligned with the fixed reference frame. It then modifies its orientation after each elemental rotation; these are referred to as “intrinsic” rotations [13]. The sequence of the elementary rotations is shown using Figure 23.

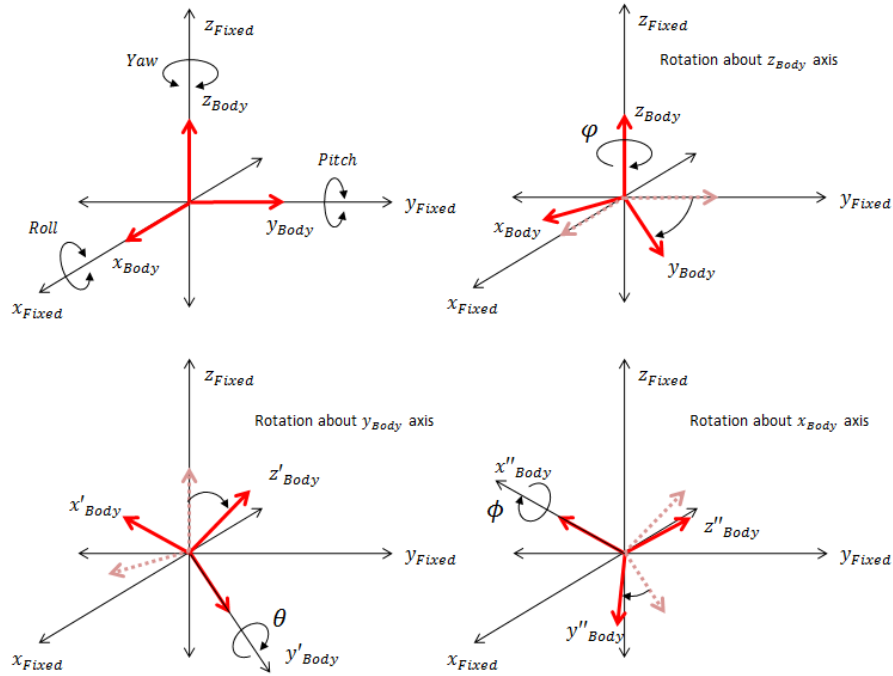


Figure 23 Euler Elementary Rotations

The rotation of an object with respect to a fixed or “global” reference frame is expressed mathematically with an orthogonal matrix with the determinant equal to one. In a 3-dimensional space, rotations about the z, y, and x axes (yaw-pitch-roll) are achieved using the following rotation matrices:

$$R_{z_g}^{v1}(\psi) = \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (3.36)$$

$$R_{y_{v1}}^{v2}(\theta) = \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix} \quad (3.37)$$

$$R_{x_{v2}}^b(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \quad (3.38)$$

Any rotation can be expressed as the product of the above three matrices. It is very important to note the specific order in which the rotations are applied, since the rotations are non-commutative. For UAV's, the typical convention is to first rotate about the z-axis (Yaw), then the y-axis (Pitch), and finally the x-axis (Roll). This rotation sequence is represented as the matrix product;

$$R = R_{z_g}^{v1}(\psi) \cdot R_{y_{v1}}^{v2}(\theta) \cdot R_{x_{v2}}^b(\phi) \quad (3.39)$$

The complete rotation from the body frame to the fixed reference frame is represented as;

$$R = \begin{bmatrix} \cos \psi \cos \theta & \cos \psi \sin \phi \sin \theta - \cos \phi \sin \psi & \sin \phi \sin \psi + \cos \phi \cos \psi \sin \theta \\ \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \psi \sin \theta & \cos \phi \sin \psi \sin \theta - \cos \psi \sin \phi \\ -\sin \theta & \cos \theta \sin \phi & \cos \phi \cos \theta \end{bmatrix} \quad (3.40)$$

The rotation matrix “R” can now be used to map a vector quantity v_b (defined in the body frame) to the fixed “global” frame v_g .

$$v_g = R v_b \quad (3.41)$$

Similarly, the inverse transformation is given by the matrix transpose.

$$v_b = R^T v_g \quad (3.42)$$

To track the orientation of an object through time we start with the rotation matrix at time “t” and then track the rate of change at time “ δt ”. The expression for the derivative of the matrix R is written as,

$$\dot{R} = \lim_{\delta t \rightarrow 0} \frac{R(t + \delta t) - R(t)}{\delta t} \quad (3.43)$$

Following the mathematical work of [14], $R(t + \delta t)$ can be written as the product of two matrices;

$$R(t + \delta t) = R(t)A(t) \quad (3.44)$$

where $A(t)$ is the rotation matrix which relates the body frame at time “t” to the body frame at time “t+ δt .” Now if we assume that the data measurement sample rate of the MEMS rate gyroscope fast enough, then we can apply the small angle approximation such that $\sin(\phi) \rightarrow \phi$, $\sin(\theta) \rightarrow \theta$, $\sin(\psi) \rightarrow \psi$, and the cosines of ϕ , θ , and ψ become one. By ignoring the products of angles, the “R” matrix becomes;

$$\Psi = \begin{pmatrix} 1 & -\psi & \theta \\ \psi & 1 & -\phi \\ -\theta & \phi & 1 \end{pmatrix} \quad (3.45)$$

Therefore, $A(t)$ can be written as,

$$A(t) = I + \delta\Psi \quad (3.46)$$

where,

$$\delta\Psi = \begin{pmatrix} 0 & -\delta\psi & \delta\theta \\ \delta\psi & 0 & -\delta\phi \\ -\delta\theta & \delta\phi & 0 \end{pmatrix} \quad (3.47)$$

We can now substitute equation (3.46) into equation (3.43) to give,

$$\begin{aligned}
\dot{R} &= \lim_{\delta t \rightarrow 0} \frac{R(t + \delta t) - R(t)}{\delta t} \\
&= \lim_{\delta t \rightarrow 0} \frac{R(t)(I + \delta\Psi) - R(t)}{\delta t} \\
&= \lim_{\delta t \rightarrow 0} \frac{R(t) + R(t)\delta\Psi - R(t)}{\delta t} \\
&= R(t) \cdot \lim_{\delta t \rightarrow 0} \frac{\delta\Psi}{\delta t}
\end{aligned} \tag{3.48}$$

For the limit $\delta t \rightarrow 0$, the small angle approximation is valid and equation (3.48)

becomes,

$$\dot{R} = R(t) \cdot \Omega(t) \tag{3.49}$$

where,

$$\Omega(t) = \begin{pmatrix} 0 & -\omega_{bz}(t) & \omega_{by}(t) \\ \omega_{bz}(t) & 0 & -\omega_{bx}(t) \\ -\omega_{by}(t) & \omega_{bx}(t) & 0 \end{pmatrix} \tag{3.50}$$

which is the skew symmetric form of the angular matrix $\omega_b(t)$. Since the attitude determination algorithm provides samples of the angular velocity, we can solve the differential equation \dot{R} for a single period $[t, t + \delta t]$.

$$R(t + \delta t) = R(t) \cdot \exp \int_t^{t+\delta t} \Omega(t) dt \tag{3.51}$$

Since we're integrating over a small change in "t," we can approximate the integral using the rectangular rule.

$$\text{Rule} \quad I_R = f(x) \cdot (b - a) \quad (3.52)$$

$$B = \int_t^{t+\delta t} \Omega(t) dt = \Omega(t) \cdot (t - t + \delta t)$$

$$B = \Omega(t) \delta t \quad (3.53)$$

We can now solve the differential equation by using Taylor's series expansion. Let $\sigma = |\omega_b \delta t|$, where $\omega_b = [\omega_{bx} \ \omega_{by} \ \omega_{bz}]^T$ and apply the trigonometric functions for $\sin(x)$ and $\cos(x)$.

$$\begin{aligned} R(t + \delta t) &= R(t) \cdot \exp(B) \\ &= R(t) \cdot \left(I + B + \frac{B^2}{2!} + \frac{B^3}{3!} + \frac{B^4}{4!} + \dots \right) \\ &= R(t) \cdot \left(I + B + \frac{B^2}{2!} - \frac{\sigma^2 B}{3!} - \frac{\sigma^3 B^2}{4!} + \dots \right) \\ &= R(t) \cdot \left(I + \left(1 - \frac{\sigma^2}{3!} + \frac{\sigma^4}{5!} \dots \right) B + \left(\frac{1}{2!} - \frac{\sigma^2}{4!} + \frac{\sigma^4}{6!} \dots \right) B^2 \right) \\ &= R(t) \cdot \left(I + \frac{\sin \sigma}{\sigma} B + \frac{1 - \cos \sigma}{\sigma} B^2 \right) \end{aligned} \quad (3.54)$$

Since we're using the small angle approximation, $\sin(\sigma) \rightarrow \sigma$ and $\cos(\sigma) \rightarrow 1$ resulting in the final solution

$$R(t + \delta t) = R(t) \cdot (I + B) \quad (3.55)$$

3.3.1 NUMERICAL CORRECTIONS

As mentioned in the previous section, the rotation matrix can be expressed mathematically with an orthogonal matrix whose determinant is equal to one. This means that any pair of columns (or rows) of the matrix are perpendicular, and that the sum of squares of the elements in each column (or row) is equal to one. However, numerical errors can accumulate over time due to sensor measurement errors. This accumulation of errors can cause the determinant to become greater than, or less than, one. In addition, the elements in each column (or row) can violate the perpendicularity property of the matrix.

To maintain the orthogonality property of the matrix, we first express the rotation matrix as a general matrix “R” with “r” elements. We can then compute the dot product of the R_1 and R_2 rows of the matrix, which should be zero.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (3.56)$$

$$R_1 = \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} \quad (3.57)$$

$$R_2 = \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} \quad (3.58)$$

The dot product is used to compute the error term;

$$Error = R_1^T \cdot R_2 = [r_{11} \quad r_{12} \quad r_{13}] \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \end{bmatrix} \quad (3.59)$$

Now the error term is used to re-calculate the R_1 and R_2 rows;

$$R'_1 = R_1 - \left(\frac{Error}{2}\right) R_2 \quad (3.60)$$

$$R'_2 = R_2 - \left(\frac{Error}{2}\right) R_1 \quad (3.61)$$

The next step is to adjust the R_3 row to be orthogonal to R_1 and R_2 by taking the cross product of R'_1 and R'_2 .

$$R'_3 = R'_1 \times R'_2 \quad (3.62)$$

Now that rows R_1 , R_2 and R_3 are orthogonal, we need to re-normalize the rows such that the magnitude is equal to one. This can be accomplished by computing the Taylor's series expansion, as described in [3].

$$R_{1,norm} = \frac{1}{2} (3 - R'_1 \cdot R'_1) R'_1 \quad (3.63)$$

$$R_{2,norm} = \frac{1}{2} (3 - R'_2 \cdot R'_2) R'_2 \quad (3.64)$$

$$R_{3,norm} = \frac{1}{2} (3 - R'_3 \cdot R'_3) R'_3 \quad (3.65)$$

3.3.2 EULER ANGLE COMPUTATION AND SINGULARITY AVOIDANCE

This section will solve for the angles θ , ϕ , and ψ , which correspond to the pitch, roll, and yaw angles respectively. Solving for the Euler angles is achieved using the rotation matrix "R" along with the generalized matrix as shown below.

$$R = \begin{bmatrix} \cos \theta \cos \psi & \sin \phi \sin \theta \cos \psi - \cos \phi \sin \psi & \cos \phi \sin \theta \cos \psi + \sin \phi \sin \psi \\ \cos \theta \sin \psi & \sin \phi \sin \theta \sin \psi + \cos \phi \cos \psi & \cos \phi \sin \theta \sin \psi - \sin \phi \cos \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \quad (3.66)$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (3.67)$$

Determining the angle for θ :

The angle for θ is most easily solved using r_{31} .

$$r_{31} = -\sin \theta \quad (3.68)$$

Inverting the equation yields,

$$\theta = -\sin^{-1}(r_{31}) \quad (3.69)$$

Determining the angle for ϕ :

The angle for ϕ can be solved using r_{32} and r_{33} .

$$r_{32} = \sin \phi \cos \theta \quad r_{33} = \cos \phi \cos \theta \quad (3.70)$$

$$\frac{r_{32}}{r_{33}} = \frac{\sin \phi \cos \theta}{\cos \phi \cos \theta} = \tan \phi \quad (3.71)$$

Note that when determining ϕ there are two possibilities for the above equation. If $\cos \theta > 0$, then $\phi = \text{atan}(+r_{32}/+r_{33})$. However, if $\cos \theta < 0$, then $\phi = \text{atan}(-r_{32}/-r_{33})$ producing the same result as $\text{atan}(+r_{32}/+r_{33})$. One simple way to handle this is to use the `atan2` function;

$$\phi = \text{atan2}(r_{32}, r_{33}) \quad (3.72)$$

Determining the angle for ψ :

The angle for ψ can be solved using r_{11} and r_{21} .

$$r_{21} = \cos \theta \sin \psi \quad r_{11} = \cos \theta \cos \psi \quad (3.73)$$

$$\frac{r_{21}}{r_{11}} = \frac{\cos \theta \sin \psi}{\cos \theta \cos \psi} = \tan \psi \quad (3.74)$$

The atan2 function can then be used to solve for ψ as,

$$\psi = \text{atan2}(r_{21}, r_{11}) \quad (3.75)$$

Singularity Avoidance

Up to this point we have developed a way to solve for θ when $\theta \neq 0$. So what happens when $\cos \theta = 0$? In this condition (corresponding to $\theta = \pm \pi/2$) the elements

r_{11}, r_{21}, r_{32} and r_{33} become zero and the following results;

$$\phi = \text{atan2}(0,0) \quad (3.76)$$

$$\psi = \text{atan2}(0,0) \quad (3.77)$$

For this situation, the elements r_{11}, r_{21}, r_{32} and r_{33} , are not able to determine the values of ϕ and ψ . One way to solve this is to use the remaining elements of “R.”

For the case when $\theta = -\pi/2$ we can solve the remaining elements using;

$$r_{12} = -\sin \phi \cos \psi - \cos \phi \sin \psi = -\sin(\phi + \psi)$$

$$r_{13} = -\cos \phi \cos \psi + \sin \phi \sin \psi = -\cos(\phi + \psi) \quad (3.78)$$

$$r_{22} = -\sin \phi \sin \psi + \cos \phi \cos \psi = \cos(\phi + \psi) = -r_{13}$$

$$r_{23} = \cos \phi \sin \psi - \sin \phi \cos \psi = -\sin(\phi + \psi) = r_{12}$$

Using the equations for r_{12} and r_{13} , we can solve for ϕ as [15],

$$(\phi - \psi) = \text{atan2}(r_{12}, r_{13})$$

$$\phi = \psi + \text{atan2}(r_{12}, r_{13}) \quad (3.79)$$

Now consider the case when $\theta = \pi/2$. After applying the sum and difference angle identity, the remaining elements become;

$$r_{12} = \sin \phi \cos \psi - \cos \phi \sin \psi = \sin(\phi - \psi)$$

$$r_{13} = \cos \phi \cos \psi + \sin \phi \sin \psi = \cos(\phi - \psi)$$

$$r_{22} = \sin \phi \sin \psi + \cos \phi \cos \psi = \cos(\phi - \psi) = r_{13}$$

$$r_{23} = \cos \phi \sin \psi - \sin \phi \cos \psi = -\sin(\phi - \psi) = -r_{12}$$

(3.80)

Using the equation for r_{12} and r_{13} , we can solve for ϕ as [22];

$$(\phi + \psi) = \text{atan2}(-r_{12}, -r_{13})$$

$$\phi = -\psi + \text{atan2}(-r_{12}, -r_{13}) \quad (3.81)$$

3.4 SUMMARY

This chapter introduced the mathematical concepts for object orientation. The first section introduced the MEMS accelerometer and how it can be used for basic tilt angle calculations. The second section introduced the MEMS rate gyroscope based on the Coriolis Effect. Both sensors are developed using capacitive sensing technology and

provide analog output signals. The analog signals are then converted into digital form using the same conversion process.

The second half of this chapter introduced a rotation matrix used to map a vector in the body frame to the global frame. Using intrinsic rotations, we performed rotations by first rotating about the yaw axis, then about the pitch axis and finally about the roll axis. Once we could relate rotations about the body axis to the global frame, we were able to discuss a method to avoid the singularity problem. The singularity problem happens when angles reach $\pm 90^\circ$ causing θ to be zero. We have now developed the concepts necessary to perform basic object orientation using the accelerometer and rate gyroscope MEMS sensors. In addition, we developed the mathematical equations to track the orientation through time. This chapter concludes with a lab exercise that will apply the fundamental concepts of object orientation.

3.5 LAB EXERCISE

The objective of this lab exercise is to introduce the Euler angle rotation matrix. The lab will begin by asking the student to create three matrices representing the rotations about each of the principle axes. The students are then asked to rotate the body axis, using two different rotation sequences, to observe how the final orientation changes. The second exercise will introduce the student to object tracking by using the small angle approximation of the rotational rate. The final exercise will build on the object tracking process to include singularity avoidance. At the end, the student will need to demonstrate the code for angles exceeding 90° of rotation (about any axis).

Lab Environment

To perform this lab, the student will need a computer with MATLAB software.

Layout

- a) The first exercise will require the student to generate the code necessary to perform a single rotation.
- b) The second exercise will elaborate on the first by performing successive rotations based on data generated by a rate gyroscope.
- c) The third exercise will implement a singularity avoidance technique.

3.5.1 PART A

- a) Open a new '.m' file
- b) Write a script that will produce a rotation matrix based on the Euler angle parameters $R_x(\phi)$, $R_y(\theta)$, and $R_z(\psi)$.
- c) Determine an arbitrary rotation sequence for;
 - a. Yaw-Pitch-Roll (ψ, θ, ϕ)

Example

```
%Create variables for the Cosine and Sine terms to simplify the math
Cr = cos(roll);   Sr = sin(roll);
Cp = cos(pitch); Sp = sin(pitch);
Cy = cos(yaw);   Sy = sin(yaw);

Rx = [1   0   0;
      0  cos(pitch)  sin(pitch);
      0 -sin(pitch)  cos(pitch)];

Ry = [cos(roll)  0  -sin(roll);
      0          1   0;
      sin(roll)  0  cos(roll)];

Rz = [cos(yaw)  sin(yaw)  0;
      -sin(yaw) cos(yaw)  0;
      0         0         1];

Rotation_matrix = Rz*Rx*Ry
```

- d) Using two other rotation sequences, determine the angles needed to produce the same final orientation.

Questions:

- 1) Explain the method used to determine the angles needed.

(Optional) Creating Plots:

You can create plots as a visual reference by using the MATLAB “quiver3” function.

Example

```
%First define the unit vectors in the vehicle's frame of reference
x_v = [1;0;0];
y_v = [0;1;0];
z_v = [0;0;1];
o_v = [x_v';y_v';z_v']; %Convert to quiver3 compliant form
```

```

%Create the initial plot
figure(1)
quiver3(zeros(3,1),zeros(3,1),zeros(3,1),o_v(:,1),o_v(:,2),o_v(:,3)) %Draw
the frame

%Enter in any initial conditions here
%Note that units must be converted to radians
yaw = 40*pi/180;
pitch = -10*pi/180;
roll = 10*pi/180;

%Perform the rotation using the rotation matrix
x_v1 = XYZ_2Rotation_Matix (yaw,pitch,roll)*x_v;
y_v1 = XYZ_2Rotation_Matix (yaw,pitch,roll)*y_v;
z_v1 = XYZ_2Rotation_Matix (yaw,pitch,roll)*z_v;
o_v1 = [x_v1';y_v1';z_v1'];

%Plot the response
figure(3)
quiver3(zeros(3,1),zeros(3,1),zeros(3,1),o_v1(:,1),o_v1(:,2),o_v1(:,3))

```

3.5.2 PART B

- a) Open a new “.m” file
- b) Write a script to perform the following tasks;
 - a. Initialize the orientation of the body using the yaw-pitch-roll sequence used for “Part A.”
 - b. Rotate the body using a constant angular velocity about a single axis. The student must make use of the small angle approximation and must rotate the body at least 10° about the rotating axis.

Example

```

I = [1 0 0;
     0 1 0;
     0 0 1];

yaw = 0*pi/180;

```

```

pitch = 0*pi/180;
roll = 0*pi/180;

Cr = cos(roll); Sr = sin(roll);
Cp = cos(pitch); Sp = sin(pitch);
Cy = cos(yaw); Sy = sin(yaw);

R = [Cr*Cy+Sr*Sp*Sy Cp*Sy -Sr*Cy+Cr*Sp*Sy;
     -Cr*Sy+Sr*Sp*Cy Cp*Cy Sr*Sy+Cr*Sp*Cy;
     Sr*Cp -Sp Cr*Cp];

Wx = 0.07; %Gyro Readings (rad/sec) /Pitch
Wy = 0.035; %Gyro Readings (rad/sec) /Roll
Wz = 0; %Gyro Readings (rad/sec) /Yaw

Omega = [0 Wz -Wy;
         -Wz 0 -Wx;
         Wy Wx 0];

dt = 0.5; %Update Rate (50Hz)

i = 0;
while i < 20
    dR = I + Omega*dt;
    R = R*dR;

    %Numerical Corrections
    R1 = R(:,1);
    R2 = R(:,2);
    Error = R1'*R2;

    R1_prime = R1 - (Error/2)*R2;
    R2_prime = R2 - (Error/2)*R1;
    R3_prime = cross(R1_prime,R2_prime);

    %Re-Normalize
    R(:,1) = 0.5*(3-(R1_prime'*R1_prime))*R1_prime;
    R(:,2) = 0.5*(3-(R2_prime'*R2_prime))*R2_prime;
    R(:,3) = 0.5*(3-(R3_prime'*R3_prime))*R3_prime;

    i = i+1;
end

%Euler Angles

```

$$\begin{aligned}\text{Pitch} &= (\text{asin}(\text{R}(3,2))*180)/\pi \\ \text{Roll} &= (\text{atan2}(\text{R}(3,1),\text{R}(3,3))*180)/\pi \\ \text{Yaw} &= (\text{atan2}(\text{R}(1,2),\text{R}(2,2))*180)/\pi\end{aligned}$$

- c) Rotate the body using two constant velocities about two axes.

Questions:

- 1) Explain the result of the two rotations.
- 2) If you were to keep the body axis rotation, what would the rotations look like?
- 3) Using all possible combinations of constant angular velocities, describe (and sketch) four possible continuous rotations. *Hint: Think of typical aircraft maneuvers.*

3.5.3 PART C

- a) Copy the code from “Part B” into a new “.m” file
- b) Verify the singularity problem by rotating the body axis y-axis (pitch) to an angle to 90°
- c) Modify the script to avoid the singularity problem
- d) Verify the script changes by repeating step “b”

Analysis Questions:

- 1) From what we have learned in this chapter, describe a scenario where it is appropriate to use only an accelerometer for determining orientation.
- 2) Under what conditions will the accelerometer not work?
- 3) Is it possible to use only the rate gyroscope for determining the orientation of an object? Describe why, or why not.
- 4) Using the accelerometer and rate gyroscope, is it possible to track accurately the position of an object over an extended period? Describe why, or why not.

CHAPTER 4

AN INTRODUCTION TO SENSOR FUSION

4.1 INTRODUCTION

Sensor fusion is a method to combine different sensors in order to produce a better result than any one individual sensor's output. For example, a magnetic compass and directional gyroscope are two different sensors used for navigation. When fused together, they provide a navigational heading that is more reliable and more accurate than what either would produce individually. To understand the technique of using sensor fusion for the magnetic compass and directional gyroscope, one must recognize the specific properties of each device. The magnetic compass produces an accurate heading when determining north, but suffers from needle 'wandering' and sensitivity to metal surfaces. The directional gyroscope is used to achieve rigidity in free space (the ability to maintain a specific orientation while the platform it is mounted to changes in direction and orientation). However, a drawback of the gyroscope is the property of precession, which causes the gyroscope to drift slowly over time due to small internal frictional forces. We can now see how the magnetic compass suffers from short term wandering, but benefits from long-term directional stability. The gyroscope suffers from long-term drift, but benefits from short-term stability in free space. Together, the sensors can produce a result that has both short-term stability and long-term stability.

4.2 SENSOR FUSION IMPLEMENTATION

One of the most widely implemented sensor fusion techniques used in both robotics and navigation, is the Kalman filter. The Kalman filter is described as an “optimal estimator” for linear dynamic systems. Its recursive properties and ability to create predictions of the system make it an extremely popular mathematical technique. The predictions of the system are possible when the error covariances are zero mean, Gaussian distributed and assumed stationary over time.

The Kalman filter’s ability to make predictions can be broken down into two primary phases; the “Prediction” and “Measurement Update.” The prediction phase uses the system’s dynamic model and process covariance error to make a prediction of the system. The measurement update phase, along with the measurement covariance error, are used to read the system’s state (specific characteristics such as; location, velocity, acceleration, etc.). If we compare the prediction with the measurement update, we can determine the changes necessary for adjustment in the next prediction cycle. Let us assume that a linear system can be modeled using state space equations with some amount of process noise error (e.g. a car traveling along an imperfect road). The measurements taken will also have some amount of noise (sensor errors). Let us look at an example of a car driving down the road. The Kalman filter’s goal is to make the best possible estimate of the car’s location at each time interval. The estimate uses a prediction based on the car’s last known position and speed, as well as the current measurements of position and speed. By describing the prediction and measurement

errors as having a zero mean Gaussian distribution, we can show the car's previous prediction and current measurement as follows [16];

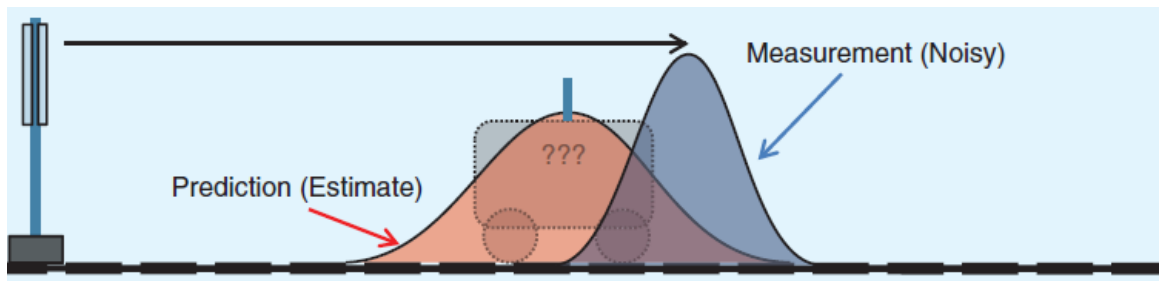


Figure 24 Prediction and Measurement Distributions [16]

To determine the best possible estimate of the current position, we can multiply the two Gaussian distributions to form a new Gaussian distribution as follows;

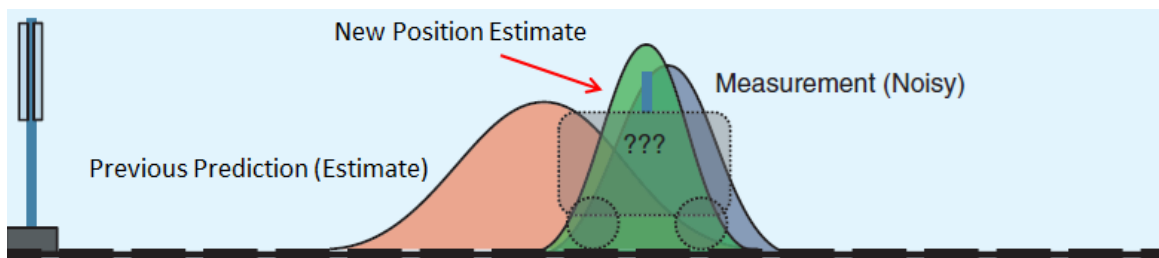


Figure 25 New Position Estimate [16]

The above example simply demonstrates the first cycle of the Kalman filter's recursive property; a continuous process of making predictions, performing measurement updates, and establishing a new position estimate.

4.3 THE KALMAN FILTER

To understand the Kalman filter more formally, we will follow the work of [17] and [18]. The following equation were developed using a more simplistic approach where the majority of the derivations are from the Least Squares Estimation section of [17] and later combined with the derivation process of [18].

The Kalman filter algorithm starts by describing the value of a variable within a discrete-time system in the form;

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \quad (4.1)$$

Here x_k is the state vector of a process at time k , F_{k-1} is the state transition matrix (equations of motion), G_{k-1} is the system input matrix, and w_{k-1} is the associated process noise (assumed to be zero-mean, uncorrelated white noise with known covariance matrix Q_k). Note that the expression “ $k-1$ ” simply means the previous “state” of the system. A more familiar version of this equation is;

$$\dot{x} = Ax + Bu \quad (4.2)$$

The observations of the state variable x_k are given in the form;

$$y_k = H_k x_k + v_k \quad (4.3)$$

where y_k is the actual measurement of x_k at time k , H_k is the connection matrix between the state vector and the measurement vector, and v_k is the associated measurement noise (assumed to be zero-mean, uncorrelated white noise with known covariance matrix R_k).

For the Kalman filter to be an optimal estimator, it must correctly model the system and measurement errors using Gaussian distributions. Therefore, the covariances of the two noise models are given by;

$$Q = E[w_k w_k^T] \quad (4.4)$$

$$R = E[v_k v_k^T] \quad (4.5)$$

where the function $E[x_y x_y^T]$ is the “Expected” value of x , which is also the center of the probability distribution. In mathematical terms, given a random variable X with values $x_1, x_2, x_3, \dots, x_n$, with probabilities $p_1, p_2, p_3, \dots, p_n$, the expected value of X is given by;

$$X = x_1 p_1 + x_2 p_2 + x_3 p_3 + \dots + x_n p_n \quad (4.6)$$

Since the Kalman filter is based on linear recursive least squares estimation, we can simplify the derivation process by analyzing the linear recursive estimator as follows;

$$y_k = H_k x + v_k \quad (4.7)$$

$$\hat{x}_k = \hat{x}_{k-1} + K_k (y_k - H_k \hat{x}_{k-1}) \quad (4.8)$$

where \hat{x}_k is computed based off the previous estimate \hat{x}_{k-1} and the new measurement y_k . K_k is the estimator gain matrix (Kalman gain) and is what we are trying to determine.

Before we compute the gain matrix K_k , it is worth looking at the mean of the estimation error of the linear recursive estimator [17]. This can be computed as;

$$E(\epsilon_{x,k}) = E(x_k - \hat{x}_k) \quad (4.9)$$

where $E(\epsilon_{x,k})$ describes the error between the true value of the system and the estimated value of the system. To carry out this expression we will substitute \hat{x}_k into $E(\epsilon_{x,k})$ as follows;

$$\begin{aligned}
E(\epsilon_{x,k}) &= E(x - \hat{x}_k) \\
&= E(x - \hat{x}_{k-1} - K_k(y_k - H_k \hat{x}_{k-1})) \\
&= E(\epsilon_{x,k-1} - K_k(H_k x + v_k - H_k \hat{x}_{k-1})) \\
&= E(\epsilon_{x,k-1} - K_k H_k (x - \hat{x}_{k-1}) - K_k v_k) \\
&= E(\epsilon_{x,k-1} - K_k H_k (\epsilon_{x,k-1}) - K_k v_k) \\
&= (I - K_k H_k) E(\epsilon_{x,k-1}) - K_k E(v_k)
\end{aligned} \tag{4.10}$$

Equation (4.10) shows that if $E(v_k) = 0$ and $E(\epsilon_{x,k-1}) = 0$, then $E(\epsilon_{x,k}) = 0$.

Therefore, if the measurement error is zero mean and the initial estimate of x is equal to the expected value of x , then the expected value of \hat{x}_k will also be equal to x . In other words, the estimate of \hat{x} will, on average, is equal to the true value of x . This is why equation (4.8) is called an unbiased estimator, regardless of the value for K_k .

At this point we can turn our attention back onto determining the gain matrix K_k . Since equation (4.8) is considered unbiased, regardless of the value for K_k . Therefore, we must choose the optimality criterion (to minimize) as the sum of the variances of the

estimation errors at time k. This is accomplished using the cost function J, where J is defined as;

$$\begin{aligned}
 J &= e_{y1}^2 + \cdots + e_{yk}^2 \\
 &= e_y^T e_y
 \end{aligned} \tag{4.11}$$

We can now substitute $(\epsilon_{x,k}) = E(x - \hat{x}_k)$ into the cost function as;

$$\begin{aligned}
 J &= E[(x_1 - \hat{x}_1)^2] + \cdots + E[(x_n - \hat{x}_n)^2] \\
 &= E(e_{x1,k}^2 + \cdots + e_{xn,k}^2) \\
 &= E(e_{x,k}^T e_{x,k}) \\
 &= E[\text{Tr}(e_{x,k}^T e_{x,k})] \\
 &= \text{Tr}P_k
 \end{aligned} \tag{4.12}$$

where P_k is the estimation-error covariance. Note that the diagonal of the covariance matrix contains the mean squared errors;

$$P_{kk} = \begin{bmatrix} E[e_{k-1}e_{k-1}^T] & E[e_k e_{k-1}^T] & E[e_{k+1}e_{k-1}^T] \\ E[e_{k-1}e_k^T] & E[e_k e_k^T] & E[e_{k+1}e_k^T] \\ E[e_{k-1}e_{k+1}^T] & E[e_k e_{k+1}^T] & E[e_{k+1}e_{k+1}^T] \end{bmatrix} \tag{4.13}$$

Since the trace of a matrix is the sum of the diagonal elements, we can see that the trace of the error covariance matrix is the sum of the mean squared errors. Therefore, the mean squared error may be minimized by minimizing the trace of P_k , which will also result in minimizing the trace of P_{kk} . We can use a process similar to the one for the cost function

J to obtain a recursive formula for the calculation of P_k . Using the results from above, we can write P_k as;

$$P_k = E(e_{x,k}e_{x,k}^T) \quad (4.14)$$

Substituting in the results for $E(\epsilon_{x,k})$ gives;

$$\begin{aligned} P_k &= E\{(I - K_k H_k) \epsilon_{x,k-1} - K_k v_k [\dots]^T\} \\ &= (I - K_k H_k) E(\epsilon_{x,k-1} \epsilon_{x,k-1}^T) (I - K_k H_k)^T - \\ &\quad K_k E(v_k \epsilon_{x,k-1}^T) (I - K_k H_k)^T - (I - K_k H_k) E(\epsilon_{x,k-1} v_k^T) K_k^T + \\ &\quad K_k E(v_k v_k^T) K_k^T \end{aligned} \quad (4.15)$$

Looking at the equation for P_k , we note that $\epsilon_{x,k-1}$ is independent of v_k . In other words, the measurement noise is uncorrelated with the error of the prior estimate as shown,

$$E(c_k \epsilon_{x,k-1}^T) = E(v_k) E(\epsilon_{x,k-1}) = 0 \quad (4.16)$$

A similar result holds for $E(\epsilon_{x,k-1} v_k^T)$, since both expected values are zero. Therefore, the equation for P_k simplifies to;

$$P_k = (I - K_k H_k) P_{k-1} (I - K_k H_k)^T + K_k R_k K_k^T \quad (4.17)$$

where R_k is the covariance of v_k . This equation is now the recursive formula for the covariance of the least squares estimation error.

The last step is to find a value for K_k that makes the cost function as small as possible. We first need to introduce a few properties of Matrix Calculus. Suppose that

“A” is a $m \times n$ matrix and $f(x)$ is a scalar. Then the partial derivative of the scalar, with respect to the matrix, can be computed as follows;

$$\frac{\partial f}{\partial A} = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix} \quad (4.18)$$

Next, we can show the partial derivative of a dot product to be computed as;

$$\begin{aligned} x^T y &= x_1 y_1 + \cdots + x_n y_n \\ \frac{\partial(x^T y)}{\partial x} &= \left[\frac{\partial(x^T y)}{\partial x_1} + \cdots + \frac{\partial(x^T y)}{\partial x_n} \right] \\ &= y_1 + \cdots + y_n \\ &= y^T \end{aligned} \quad (4.19)$$

The partial derivative of a quadratic with respect to a vector can be computed as;

$$\begin{aligned} x^T A x &= [x_1 \quad \cdots \quad x_n] \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= \left[\sum_i x_i A_{i1} \quad \cdots \quad \sum_i x_i A_{in} \right] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= \sum_{i,j} x_i x_j A_{ij} \end{aligned} \quad (4.20)$$

With the partial derivative calculation as shown;

$$\begin{aligned}
\frac{\partial(x^T Ax)}{\partial x} &= \left[\frac{\partial(x^T Ax)}{\partial x_1} \quad \cdots \quad \frac{\partial(x^T Ax)}{\partial x_n} \right] \\
&= \left[\sum_j x_j A_{1j} + \sum_i x_i A_{i1} \quad \cdots \quad \sum_j x_j A_{nj} + \sum_i x_i A_{in} \right] \\
&= \left[\sum_j x_j A_{1j} \quad \cdots \quad \sum_j x_j A_{nj} \right] + \left[\sum_i x_i A_{i1} \quad \cdots \quad \sum_i x_i A_{in} \right] \\
&= x^T A^T + x^T A
\end{aligned} \tag{4.21}$$

It is important to note that if “A” is symmetric, then $A = A^T$ and the above expression simplifies to;

$$\frac{\partial(x^T Ax)}{\partial x} = 2x^T A \tag{4.22}$$

The last partial derivative needed before deriving K_k is the partial derivative of $\text{Tr}(ABA^T)$ with respect to “A”. To begin we start by first computing ABA^T as follows:

$$\begin{aligned}
ABA^T &= \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} \begin{bmatrix} B_{11} & \cdots & B_{1n} \\ \vdots & \ddots & \vdots \\ B_{n1} & \cdots & B_{nn} \end{bmatrix} \begin{bmatrix} A_{11} & \cdots & A_{m1} \\ \vdots & \ddots & \vdots \\ A_{n1} & \cdots & A_{mn} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{j,k} A_{1k} B_{kj} A_{1j} & \cdots & \sum_{j,k} A_{1k} B_{kj} A_{mj} \\ \vdots & \ddots & \vdots \\ \sum_{j,k} A_{mk} B_{kj} A_{1j} & \cdots & \sum_{j,k} A_{mk} B_{kj} A_{mj} \end{bmatrix}
\end{aligned} \tag{4.23}$$

We can see that the trace of ABA^T is;

$$\text{Tr}(ABA^T) = \sum_{i,j,k} A_{ik}B_{kj}A_{ij} \quad (4.24)$$

The partial derivative of the $\text{Tr}(ABA^T)$ with respect to “A” can be computed as;

$$\begin{aligned} \frac{\partial \text{Tr}(ABA^T)}{\partial A} &= \begin{bmatrix} \frac{\partial \text{Tr}(ABA^T)}{\partial A_{11}} & \cdots & \frac{\partial \text{Tr}(ABA^T)}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \text{Tr}(ABA^T)}{\partial A_{m1}} & \cdots & \frac{\partial \text{Tr}(ABA^T)}{\partial A_{mn}} \end{bmatrix} \\ &= \begin{bmatrix} \sum_j A_{1j}B_{1j} + \sum_k A_{1k}B_{k1} & \cdots & \sum_j A_{1j}B_{nj} + \sum_k A_{1k}B_{kn} \\ \vdots & \ddots & \vdots \\ \sum_j A_{mj}B_{1j} + \sum_k A_{mk}B_{k1} & \cdots & \sum_j A_{mj}B_{nj} + \sum_k A_{mk}B_{kn} \end{bmatrix} \\ &= \begin{bmatrix} \sum_j A_{1j}B_{1j} & \cdots & \sum_j A_{1j}B_{nj} \\ \vdots & \ddots & \vdots \\ \sum_j A_{mj}B_{1j} & \cdots & \sum_j A_{mj}B_{nj} \end{bmatrix} + \begin{bmatrix} \sum_k A_{1k}B_{k1} & \cdots & \sum_k A_{1k}B_{kn} \\ \vdots & \ddots & \vdots \\ \sum_k A_{mk}B_{k1} & \cdots & \sum_k A_{mk}B_{kn} \end{bmatrix} \\ &= AB^T + AB \end{aligned} \quad (4.25)$$

If “B” is symmetric then the partial derivative can be simplified to,

$$\frac{\partial \text{Tr}(ABA^T)}{\partial A} = 2AB \quad \text{if } B = B^T \quad (4.26)$$

At this point we can now solve for K_k . To help simplify the problem, I expanded out the terms of P_k as follows;

$$\begin{aligned}
P_k &= (I - K_k H_k) P_{k-1} (I - K_k H_k)^T + K_k R_k K_k^T \\
&= (P_{k-1} - P_{k-1} K_k H_k) (I - K_k H_k)^T + K_k R_k K_k^T \\
&= P_{k-1} - P_{k-1} K_k H_k - P_{k-1} K_k^T H_k^T + K_k H_k P_{k-1} K_k^T H_k^T + K_k R_k K_k^T \\
&= P_{k-1} - P_{k-1} K_k H_k - P_{k-1} K_k^T H_k^T + K_k (H_k P_{k-1} H_k^T + R_k) K_k^T
\end{aligned} \tag{4.27}$$

Note that the trace of a matrix is equal to the trace of its transpose. Therefore, the expression for P_k can be written as;

$$\begin{aligned}
Tr[P_k] &= Tr[P_{k-1}] - Tr[P_{k-1} K_k H_k] - Tr[P_{k-1} K_k^T H_k^T] + Tr[K_k (H_k P_{k-1} H_k^T + R_k) K_k^T] \\
&= Tr[P_{k-1}] - 2Tr[P_{k-1} K_k H_k] + Tr[K_k (H_k P_{k-1} H_k^T + R_k) K_k^T]
\end{aligned} \tag{4.28}$$

Now take the partial derivative with respect to K_k and recall that $\frac{\partial Tr(ABA^T)}{\partial A} =$

$2AB$ if $B = B^T$;

$$\begin{aligned}
\frac{\partial Tr[P_k]}{\partial K_k} &= Tr[P_{k-1}] - 2Tr[K_k H_k P_{k-1}] + Tr[K_k (H_k P_{k-1} H_k^T + R_k) K_k^T] \\
&= -2[H_k P_{k-1}]^T + 2[K_k (H_k P_{k-1} H_k^T + R_k)]
\end{aligned} \tag{4.29}$$

Re-arranging the terms results in;

$$\frac{\partial Tr[P_k]}{\partial K_k} = 2(I - K_k H_k) P_{k-1} (-H_k^T) + 2K_k R_k \tag{4.30}$$

where $(H_k P_{k-1})^T = P_{k-1} H_k^T$.

The final step is to find the value of K_k that will also minimize the cost function J_k . This can be accomplished by setting the above derivative equal to zero and then solve for K_k ;

$$\begin{aligned}
0 &= 2(I - K_k H_k) P_{k-1} (-H_k^T) + 2K_k R_k \\
2K_k R_k &= 2(I - K_k H_k) P_{k-1} H_k^T \\
K_k R_k &= P_{k-1} H_k^T - K_k H_k P_{k-1} H_k^T \\
K_k H_k P_{k-1} H_k^T + K_k R_k &= P_{k-1} H_k^T \\
K_k (R_k + H_k P_{k-1} H_k^T) &= P_{k-1} H_k^T \\
K_k &= P_{k-1} H_k^T (H_k P_{k-1} H_k^T + R_k)^{-1} \tag{4.31}
\end{aligned}$$

In this section, we will now describe the state projection property used by the Kalman filter. Let us describe the state projection of \hat{x}_k using the following;

$$\hat{x}'_{k+1} = F \hat{x}_k \tag{4.32}$$

Recall that F is the system transition matrix. Next, we will define an equation to project the error covariance matrix into the next time interval, k+1. This can be achieved by forming an expression for the previous error;

$$e'_{k+1} = x_{k+1} - \hat{x}_{k+1} \tag{4.33}$$

Substituting in the expressions for x_{k+1} and \hat{x}_{k+1} yields;

$$\begin{aligned}
e'_{k+1} &= (F x_k + w_k) - F \hat{x}_k \\
&= F(x_k - \hat{x}_k) + w_k
\end{aligned}$$

$$= Fe_k + w_k \quad (4.34)$$

Extending the estimation error covariance to time $k+1$ produces the following;

$$\begin{aligned} P'_{k+1} &= E(\epsilon_{x,k+1}\epsilon_{x,k+1}^T) \\ &= E[(Fe_k + w_k)(Fe_k + w_k)^T] \end{aligned} \quad (4.35)$$

Since e_k and w_k have zero cross-correlation, the covariance simplifies to;

$$\begin{aligned} P'_{k+1} &= E[(Fe_k + w_k)(Fe_k + w_k)^T] \\ &= E[Fe_k(Fe_k)^T] + E[w_k w_k^T] \\ &= FP_k F^T + Q \end{aligned} \quad (4.36)$$

At this point, we have completed the basic derivation for the recursive Kalman filter.

The next section will provide a summary of the Kalman filter along with the process flow diagram.

4.3.1 KALMAN FILTER SUMMARY

Now that we have expressions for the state estimate \hat{x}_k , the estimation error covariance P_k , the estimator gain K_k , and the state prediction, we are able to summarize the recursive Kalman filter as follows:

- A. Initialize the state estimate and estimation-error covariance;

$$\hat{x}_0 = E(0)$$

$$P_0 = E[(x - \hat{x}_0)(x - \hat{x}_0)^T] \quad (4.37)$$

If no knowledge about \hat{x}_0 is given when the first measurement is taken, then $P_0 = \infty I$.

On the other hand, if the exact value of \hat{x}_0 is known when the first measurement is taken, then $P_0 = 0$.

B. After the initialization, and when $k > 0$, we perform the following:

- a) Obtain the measurement y_k (assuming y_k is given by the equation);

$$y_k = H_k x + v_k \quad (4.38)$$

- b) Update the estimate of \hat{x}_k and the estimation error covariance P_k by following the process diagram.

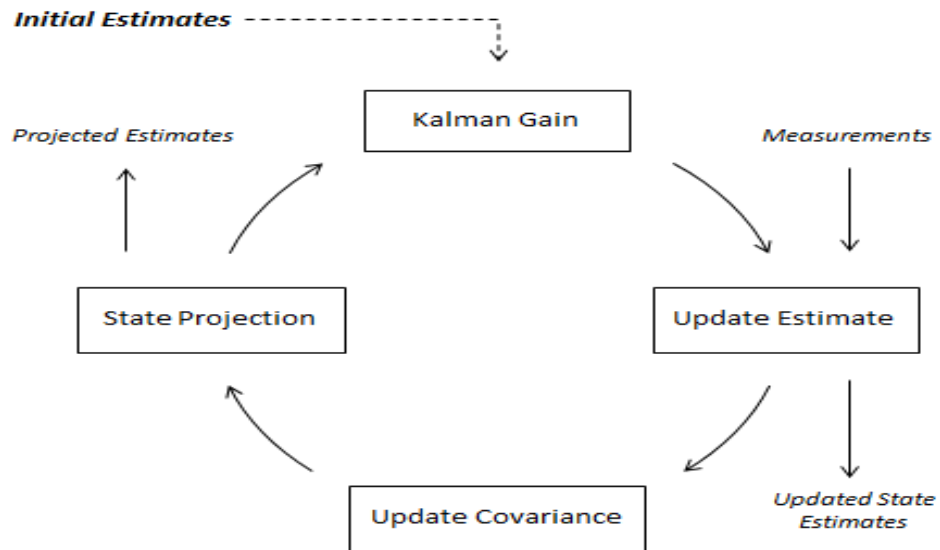


Figure 26 Kalman Filter Process [18]

Each box is given by the following equations;

Kalman Gain	$K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1}$
Update Estimate	$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1})$
Update Covariance	$P_k = (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T$
State Projection	$\hat{x}'_{k+1} = \Phi\hat{x}_k$ $P'_{k+1} = \Phi P_k \Phi^T + Q$

Note that the Kalman gain and the update covariance can have alternate forms as described in [18], and is summarized as follows;

Kalman Gain	$K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1}$ $= P_kH_k^TR_k^{-1}$
Update Covariance	$P_k = (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T$ $= (P_{k-1}^{-1} + H_k^TR_k^{-1}H_k)^{-1}$ $= (I - K_kH_k)P_{k-1}$

4.4 LAB EXERCISE

This lab exercise is designed to apply the equations of the Kalman filter to a classic physics problem using Matlab. The lab will start by modeling the classic problem of a falling body to verify the equations of motion [19, 20]. Next, we will assume that a sensor is mounted to the falling object that records position. To make the sensor realistic we will apply random noise to the measurements and then employ the Kalman filter to provide an estimate of the falling object's position as it falls. The true state (measured data and the filtered data) will later be plotted to see how well the Kalman filter tracks the true state of the system. The last part of the lab will be to adjust different parameters to see how they affect the performance of the filter.

Lab Environment

To perform this lab exercise the student will need a computer with MATLAB.

Layout

- a) The first exercise will be generating the code necessary to model the dynamics of a falling object.
- b) The second exercise will elaborate on the first by converting the discrete time differential equations into a single state-space format. This will be used later for the Kalman filter.
- c) The third exercise will be to implement the Kalman filter equations.

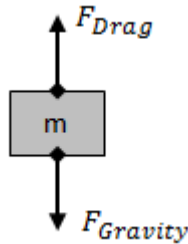
4.4.1 PART A

a) Determine the dynamic equation for an object of mass 'm' falling in air.

Start with Newton's second law of motion:

$$F = ma = m \cdot \left(\frac{dv}{dt}\right)$$

Now create the free body diagram;



Where,

$$F_{Drag} = \frac{1}{2} \rho C_D A v^2$$

$$F_{Gravity} = -gm$$

The total forces acting on the object becomes;

$$F_{Total} = F_{Drag} - F_{Gravity}$$

Since the Kalman filter requires the equations of motion to be linear, we must assume that the force of drag is directly proportional to the velocity [20]. Therefore, we will simplify the drag equation using the following form;

$$F_{Drag} = -kv$$

Note: The negative term comes from the fact that the velocity is negative (falling downward).

The equation of motion now becomes;

$$F_{Total} = -kv - gm$$

$$F_{Total} = m \cdot \left(\frac{dv}{dt}\right) = -kv - gm$$

$$\frac{dv}{dt} = -\frac{k}{m}v - g$$

Next, we need to create the discrete time differential equations. To calculate position we use;

$$v = \frac{dx}{dt} = \frac{(x_t - x_{t-1})}{dt}$$

$$x_t = x_{t-1} + v_{t-1} \cdot dt$$

To calculate velocity we use;

$$a = \frac{dv}{dt} = \frac{(v_t - v_{t-1})}{dt}$$

$$\frac{(v_t - v_{t-1})}{dt} = -\frac{k}{m}v - g$$

$$v_t = v_{t-1} - \left(\frac{k}{m}v_{t-1} + g\right) \cdot dt$$

- b) Open a new '.m' file
- c) Test the equations of motion by creating a simulation with the following parameters

$$K = 0.25, m = 10, TMAX = 200, dt = 0.01$$

Example

```

x0=200; %initial position
v0=0;   %initial velocity

TMAX=200;      %Total Time
dt=0.01;      %time sample
time=1:dt:TMAX; %Number of Iterations

%global g m k
g=9.8;        %gravitational constant
m=10;         %mass of the object
k=0.25;       %Drag equation coefficient

```

```

X=zeros(1,TMAX);      %Create X History Matrix
V=zeros(1,TMAX);      % Create Y History Matrix
X(1)=x0;              %Initialize position with initial position
V(1)=v0;              %Initialize position with initial velocity
%Note Matlab does not use '0' indexing

%Create the Simulation by solving the equations of motion
%and recording the results for each time step 'dt'.
for t=2:TMAX
    X(t)=X(t-1)+(V(t-1))*dt;
    V(t)=V(t-1)+(-(k/m)*(V(t-1))-g)*dt;
end

%Plot the results for position and velocity
figure();
plot(X,'b'); hold on;
title(['Falling object k/m = ' num2str(k/m)]);
plot(V,'r')
legend('x','v'); hold off

```

d) Change the values for both “k” and “m,” using three different values.

Questions:

1) The equations of motion were calculated assuming that the drag equation is linear.

Using the standard drag equation ($D = \frac{1}{2} \rho A v^2$), re-calculate the equations of motion.

2) Using the new equations of motion, solve the ordinary differential equations using the MATLAB function “ode45.”

4.4.2 PART B

- a) Open a new '.m' file
- b) Part B is an extension of Part A where the same concepts will be applied, but in a slightly different format, allowing the Kalman filter to be applied.
- c) The first step is to convert the differential equations used in Part A to their equivalent State-Space form.

Recall that the Kalman filter assumes a discrete time system of the form;

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1}$$

where x_k is the state vector of the process at time k, F is the state transition matrix (equations of motion), and w_{k-1} is the associated process noise. The observations of the state variable x_k can be given in the form;

$$y_k = H_k x_k + v_k$$

where y_k is the actual measurement of x_k at time k, H_k is the connection matrix between the state vector and the measurement vector, and v_k is the associated measurement noise. Therefore, the discrete time system for the falling object becomes;

$$x_t = \begin{bmatrix} x_t \\ v_t \end{bmatrix}$$

Giving,

$$x_t = \begin{bmatrix} x_t \\ v_t \end{bmatrix} = A \begin{bmatrix} x_{t-1} \\ v_{t-1} \end{bmatrix}$$

Where 'A' is the state transition matrix.

Since,

$$x_t = x_{t-1} + v_{t-1} \cdot dt$$

$$v_t = v_{t-1} - \left(\frac{k}{m} v_{t-1} + g \right) \cdot dt$$

The state space form becomes;

$$x_t = \begin{bmatrix} 1 & dt \\ 0 & \left(1 - \frac{k}{m} \cdot dt \right) \end{bmatrix} \begin{bmatrix} x_{t-1} \\ v_{t-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & -g \cdot dt \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

d) Verify the state space equation.

Example

```
x0=200; %initial position
v0=0;   %initial velocity

TMAX=200;      %Total Time
dt=0.01;      %time sample
time=1:dt:TMAX; %Number of Iterations

%global g m k
g=9.8;        %gravitational constant
m=10;        %mass of the object
k=0.25;      %Drag equation coefficient

x=zeros(2,TMAX); %Create x History Matrix
x(1,1)=x0;      %initialize position with initial position
x(2,1)=v0;      %Initialize position with initial position

u=[0 1]';      %Control Matrix

for t=2:TMAX
    A = [1    dt ;
         0 (1-(k/m)*dt)];
    B = [1    0 ;
         0 -g*dt];
    x(:,t) = A*x(:,t-1)+B*u;
```

```

end

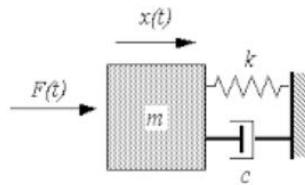
figure();
plot(x(1,:), 'b');
title(['Falling object k/m = ' num2str(k/m)]);

```

- e) Provide a printout of the plot using the three different parameters for ‘k’ and ‘m’ from Part A.

Questions:

- 1) Determine the state-space equations for the mass-spring-damper system shown below. The output of the system is the displacement (x).



4.4.3 PART C

- a) Open a new ‘.m’ file
- b) Create an object “s” whose members are all the important data structures implemented by the Kalman filter.

- a. Create the transition matrix called “s.A”

$$s.A = \begin{bmatrix} 1 & dt \\ 0 & \left(1 - \frac{k}{m} \cdot dt\right) \end{bmatrix}$$

- b. Create the input control called “s.B” and “s.u”

$$s.B = \begin{bmatrix} 1 & 0 \\ 0 & -g \cdot dt \end{bmatrix}$$

$$s.u = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- c. Create the variables for the measurement noise variance and standard deviation.

Example

$$MNstd = 12;$$

$$MNV = MNstd^2;$$

- d. Create the matrix for the measurement noise error;

$$s.R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot MNV;$$

Note: The Matlab function “eye(n)” returns an $n \times n$ matrix with 1’s along the diagonal and 0’s everywhere else.

- e. Create the variables for the process noise variance and standard deviation.

Example

$$PNstd = 0.4;$$

$$PNV = PNstd^2;$$

- f. Create the matrix for the process noise covariance matrix;

$$s.Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot PNV;$$

- g. Create the matrix for the measurement connection matrix;

$$s.H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix};$$

h. Initialize the states using the following;

$$s.x = [x_0 \quad v_0];$$

$$s.P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot MNV;$$

$$s.detP = \det(s.P);$$

$$s.z = [0 \quad 0];$$

c) Open a new '.m' file. This new '.m' file will contain the Kalman filter function.

a. Create a function for the Kalman filter update equations

Example

Function [s] = Kalman_Filter (s)

b. Implement the following equations;

Description	Theory	Matlab
Kalman Gain	$K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1}$	$K=(s.P)*(s.H)'*inv((s.H)*(s.P)*(s.H)'+(s.R));$
Update Estimate	$\hat{x}_k = \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1})$	$s.x = (s.x) + K*((s.z)-(s.H)*(s.x));$
Update Covariance	$P_k = (I - K_kH_k)P_{k-1}(I - K_kH_k)^T + K_kR_kK_k^T$	$s.P = (s.P) - K*(s.H)*(s.P);$
State Projection	$\hat{x}'_{k+1} = \Phi\hat{x}_k$ $P'_{k+1} = \Phi P_k \Phi^T + Q$	$s.P = (s.A)*(s.P)*(s.A)' + (s.Q);$

d) The simulation can be created by using the following example;

Example

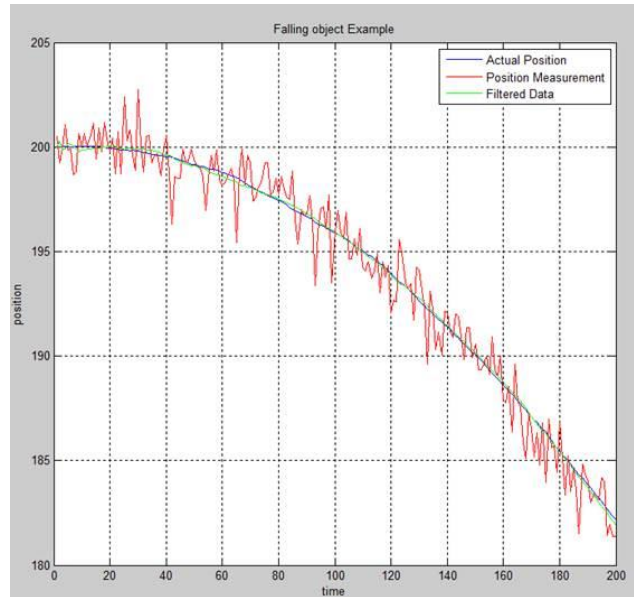
```
for t=2:TMAX
    %Start simulation process
    tru(t,:) = s(t-1).A*tru(t-1,:) + s(t-1).B*s(t-1).u + PNstd*randn(2,1); % True state +
noise
    s(t-1).z = s(t-1).H*tru(t,:) + MNstd*randn(2,1); %Create a
measurement
    s(t) = Falling_Object_Kalman_Filter(s(t-1)); %Perform a
Kalman filter iteration
    detP(t) = s(t).detP; %Keep track of the "net" uncertainty
end
```

e) Plot the results.

Example

```
Measurement = [s.z]';
Filter = [s.x]';

figure(1)
plot(tru(:,1),'b'); hold on;
plot(Measurement(:,1),'r');
plot(Filter(:,1),'g');
title('Falling object Example');
grid
xlabel('time');
ylabel('position');
legend('Actual Position','Position Measurement','Filtered Data'); hold off;
```

- f) Assume that your initial guess was not the actual starting position, such as 185 instead of 200. Change the initial state for “s.x” and plot the results.

Conceptual Questions:

- 1) Explain how to determine the variance and standard deviation of a measurement.
Why are these parameters important for the Kalman filter?
- 2) Explain how the Kalman gain is calculated.
- 3) How does the Kalman gain affect the update estimate?
- 4) What does the update covariance matrix do?
- 5) Explain what the state projection matrix is doing.

CHAPTER 5

AN ALTERNATIVE SENSOR FUSION APPROACH

5.1 INTRODUCTION

In the previous section, the Kalman filter was described as being an “optimal estimator” for linear dynamic systems. When applying the Kalman filter in sensor fusion applications, the filter takes each sensor as an input to the dynamics of the system, and produces an optimal output for those sensors. The question that I asked myself was to see if the accelerometer and rate gyroscope sensors could be combined prior to applying this optimal estimation algorithm.

The basis for proposing that the accelerometer and gyroscope be combined prior to the Kalman filter stems from my personal observations while characterizing and understanding the output responses of the accelerometer and rate gyroscope.

Observation 1: Starting with a three axis accelerometer, place it onto a fixed level surface (such as a table), such that the X and Y plane is parallel to the surface. If a rotation were to occur about either the X or Y-axis, then the accelerometer would become affected by the gravitational force acting against the internal sensing elements. The result is an output measurement proportional to the angle rotated. Now, if I assume that the gravitational force is constant, with no other accelerations acting on the surface, then I can extend this observation to a surface traveling at a constant velocity.

Observation 2: If an accelerometer is not under free fall, but subjected to random accelerations, such that the gravitational force is not easily extractable, then the

orientation of the accelerometer is not solvable. For example, it is not possible to determine the sensor's orientation while applying random accelerations simultaneously along all three axes of the device.

Observation 3: If a rate gyroscope were mounted to the accelerometer such that they share the same X, Y, and Z axes, then there becomes a possibility to separate out the gravitational force vector from the applied accelerations (provided that the sensors are not in free fall). Now, if we also limit the accelerations to approximately three times the force of gravity (range of accelerometer used), then rotations will cause a change in the sensed acceleration.

Problem Statement: Using the above observations, I realized that a possibility exists to relate the derivative of the accelerometer with the corresponding rate of change from the gyroscope. In other words, if the two sensors are correlated (meaning that the behavior of the accelerometer's derivative is similar to the behavior of the gyroscope), then the combination of the two sensor measurements can accurately reflect the change in orientation. However, if the two sensors are weakly correlated (assuming either that a constant acceleration occurs with no change from the rate gyroscope or that the gyroscope with no corresponding measures a constant rotational rate changes from the accelerometer), then the two sensor measurements will not accurately reflect the change in orientation. In this case, the two sensors are compared through a new algorithm, resulting in a weighted average of the two sensors. Where the weight is based on the magnitude of error observed.

5.2 RELATED WORK

Most research related to inertial sensor fusion techniques focus on either Euler or Quaternion based orientation estimation using the Extended Kalman filter [1, 4, 14, and 21]. However, the mathematical concepts are complex with most research papers not providing any intermediary steps to the computations. To address the advanced concepts used in most papers, [22] had developed an alternative algorithm using basic mathematics and trigonometry. This simplified approach focused on the accelerometer and rate gyroscope's relationship, rather than system dynamics.

In his work, [22] used the concept of initializing the orientation of an IMU with an accelerometer. He then produced estimates according to subsequent changes of orientation based on a weighted average of the accelerometer and rate gyroscope. To accomplish this task, the algorithm was broken down into three main parts;

- a. Estimate a new orientation by taking the previous estimate and applying the integrated result from the rate gyroscope between its last measurement and the current measurement.
- b. Using trigonometric relationships, determine the orientation of the IMU using the estimate found in part "a."
- c. Calculate a weighted average of the orientation provided by the accelerometer with that from part "b."

The final estimate should produce a better result than what the accelerometer or rate gyroscope could produce independently.

In contrast to the observations and problem statement provided above, the work of [22] assumes that the accelerometer produces results affected by noise and cannot be trusted. In addition, the rate gyroscope is considered the primary sensor. Therefore, its results are heavily weighted when compared to the accelerometer. Thus, this algorithm is primarily aimed at reducing the effects of integration errors due to bias errors of the gyroscope.

5.3 PROPOSED ALGORITHM

The algorithm that I am proposing was developed to “relate the rate of change of acceleration from the accelerometer with the corresponding rate of change of the rate gyroscope,” as described by the problem statement.

Prior to developing the full algorithm, I had conducted an initial feasibility study to analyze the behavior of the accelerometer’s derivative to that of the rate gyroscope. Referencing Figure 27, the results helped me determine that the problem statement could be supported. The data shown represents a test condition where the accelerometer and rate gyroscope were mounted on a test board and subject to random motion along the X-Axis. This is seen by the large spikes in the accelerometer’s data (showing heavy accelerations with very little rotations about the Y-Axis). However, there were times when the test board was subject to rotations about the Y-Axis and data from both the accelerometer and rate gyroscope showed strong correlation. Note that I had implemented a 12-point moving average filter to the derivative results of the accelerometer. The rate gyroscope’s results, however, are presented with no filtering.

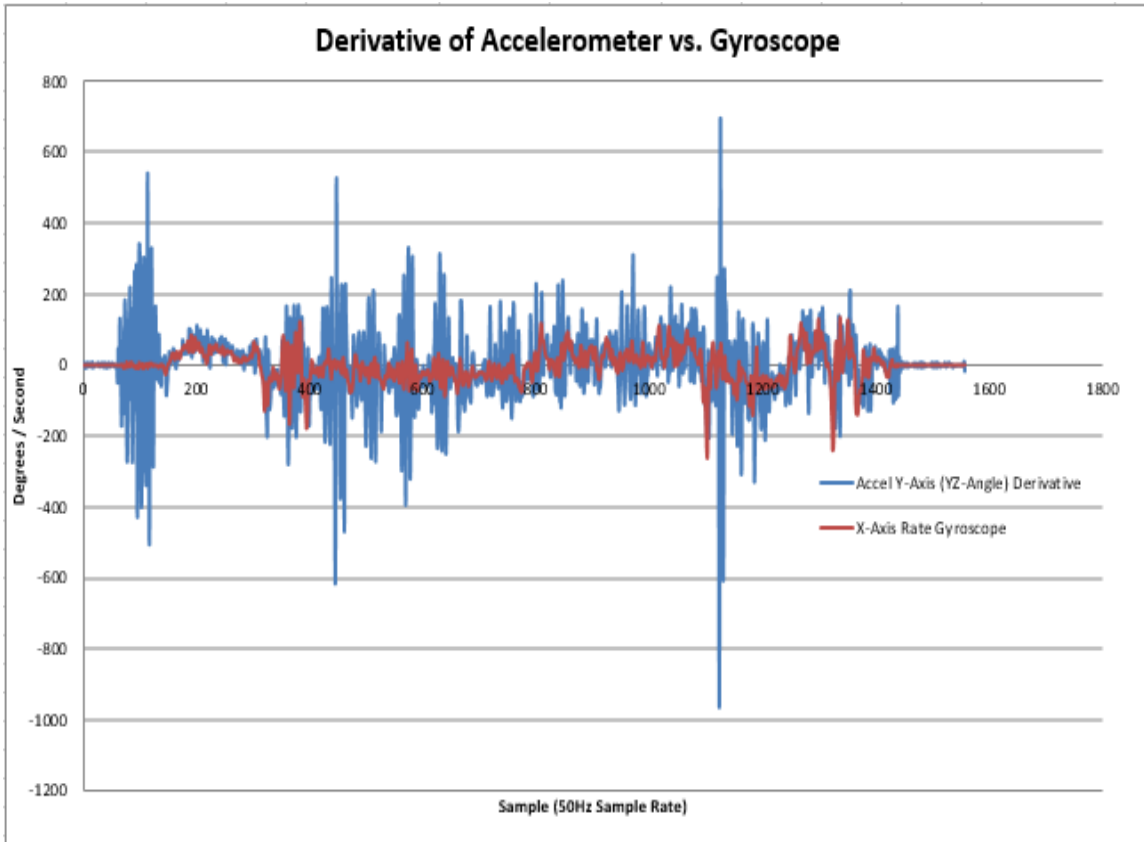


Figure 27 Accelerometer Derivative & Angular Rate

The result of the feasibility study provided cases where the accelerometer’s derivative matched the behavior of the rate gyroscope. The periods of strong correlation represented actual rotation while weakly correlated results represented periods of little rotational motion.

5.3.1 DEVELOPMENT OF THE ALGORITHM

The algorithm begins by initializing both the accelerometer and rate gyroscope while resting on a flat level surface (local reference frame). The conversion from the analog-to-digital converter (ADC) for the accelerometer is as follows;

$$R_{X_{ACC}} = R_X = \frac{\left(\frac{ADC_{R_X}(ADC_{Voltage\ Scale})}{ADC_{Resolution}}\right) - (Zero_g\ Bias)}{Device\ Sensitivity} + Null_{Offset} \quad (5.1)$$

$$R_{Y_{ACC}} = R_Y = \frac{\left(\frac{ADC_{R_Y}(ADC_{Voltage\ Scale})}{ADC_{Resolution}}\right) - (Zero_g\ Bias)}{Device\ Sensitivity} + Null_{Offset} \quad (5.2)$$

$$R_{Z_{ACC}} = R_Z = \frac{\left(\frac{ADC_{R_Z}(ADC_{Voltage\ Scale})}{ADC_{Resolution}}\right) - (Zero_g\ Bias)}{Device\ Sensitivity} + Null_{Offset} \quad (5.3)$$

Example C-Code Conversion:

```

-----
float IMU_AIN(Byte Channel){ //(Ax, Ay, Az, Gx, Gy)
//float Null_Offset[5] = {0, 0, 0, 0, 0}; //External!
float Bias_Point[5] = {1.6, 1.6, 2, 1.40, 1.34};
float Sensitivity[5] = {0.33, 0.33, 0.22, -0.092, -0.092};
return (((ADC_Read(Channel)*3.3)/1023)-Bias_Point[Channel])/Sensitivity[Channel]+Null_Offset[Channel];
}

```

Next, we would like to obtain the angles between the X-Z Axis and the Y-Z Axis as follows,

$$A_{XZ_{Accel}} = atan2(z, x) \quad (\text{Radians}) \quad (5.4)$$

$$A_{YZ_{Accel}} = atan2(z, y) \quad (\text{Radians}) \quad (5.5)$$

Note: The results of 'atan2' are in radians and will need to be converted into units of degrees.

$$\text{Degrees} = \text{Radians} \left(\frac{180^\circ}{\pi} \right) \quad (5.6)$$

Example C-Code Conversion:

```

void Get_ATAN2_Angles(void){
int Index=0;
int Position_Variable=0;
int Sequence=0;

//Shift Data
//Positions = 2 (X & Y)
//Buffer Size = 3
for(Position_Variable = 0; Position_Variable <= 1; Position_Variable++){
for(Index=0; Index < (Buffer_Size-1); Index++){
ATAN2_Data[Position_Variable][Index]=ATAN2_Data[Position_Variable][Index+1];
}
}

//Update
for(Sequence=0; Sequence <= 1; Sequence++){
ATAN2_Data[Sequence][Buffer_Size-1] = atan2(-1*IMU_Sensor_Data[Z_Axis][Buffer_Size-1],IMU_Sensor_Data[Sequence][Buffer_Size-1]);
}

//Radians to Degrees
Sequence=0;
for(Sequence=0; Sequence <= 1; Sequence++){
Radians_to_Degrees(ATAN2_Data[Sequence][Buffer_Size-1]);
}
}

#define M_PI 3.141593
//---- Convert Radians to Degrees----
float Radians_To_Degrees(float R){
return (R*(180/M_PI));
}

```

Since the accelerometer data may be corrupted by noise cause from mechanical vibrations, we can apply a moving average filter as shown;

$$A_{XZ_{Accel_{AVG}}}[m] = \frac{1}{N} \sum_{n=0}^{N-1} A_{XZ_{Accel}}[m - n] \quad \text{where } \{N \leq m\} \quad (5.7)$$

$$A_{YZ_{Accel_{AVG}}}[m] = \frac{1}{N} \sum_{n=0}^{N-1} A_{YZ_{Accel}}[m - n] \quad \text{where } \{N \leq m\} \quad (5.8)$$

Now that the angles for A_{xz} and A_{yz} are known, we are going to take the derivative to relate the rate of change of the angle between the X-Z and Y-Z Axes. This can be accomplished as follows;

$$\frac{dA_{XZ_{Accel}}}{dt} [n] = \frac{A_{XZ_{Accel}}[n] - A_{XZ_{Accel}}[n - 1]}{dt} \quad (5.9)$$

$$\frac{dA_{YZ_{Accel}}}{dt} [n] = \frac{A_{YZ_{Accel}}[n] - A_{YZ_{Accel}}[n - 1]}{dt} \quad (5.10)$$

One important aspect of taking the derivative of the accelerometer-derived angle is the magnitude of the noise. The noise is actually how the algorithm determines if the accelerometer data is usable. In the ideal situation there would be very little to no noise after taking the derivative. This would provide an accurate correlation between the accelerometer and the gyroscope. However, if there were a lot of noise then the rate gyroscope would take over, as it is not as sensitive to vibrational noise.

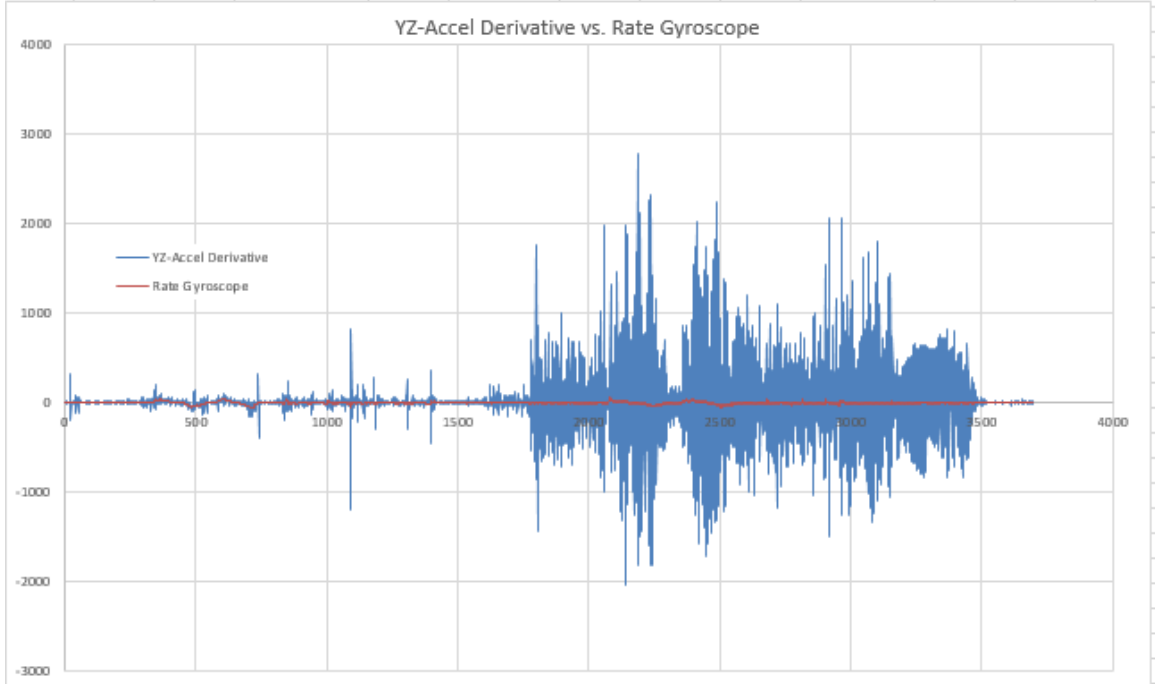


Figure 28 YZ-Accelerometer Derivative vs. Rate Gyroscope

The gyroscope readings are converted from ADC values into degrees/second in the same way the ADC values for the accelerometer were converted to units of force (only the X and Y-Axes are needed).

$$R_{X_{Gyro}} = R_X = \frac{\left(\frac{ADC_{R_X}(ADC_{Voltage\ Scale})}{ADC_{Resolution}} \right) - (Zero_g\ Bias)}{Device\ Sensitivity} + Null_{Offset} \quad (5.11)$$

$$R_{Y_{Gyro}} = R_Y = \frac{\left(\frac{ADC_{R_Y}(ADC_{Voltage\ Scale})}{ADC_{Resolution}} \right) - (Zero_g\ Bias)}{Device\ Sensitivity} + Null_{Offset} \quad (5.12)$$

The presented algorithm uses the direct integrated results from the previous and current measurement as follows;

$$I_{X_{Gyro}} = \left(R_{X_{Gyro}}[n] - R_{X_{Gyro}}[n - 1] \right) \cdot \Delta t \quad (5.13)$$

$$I_{Y_{Gyro}} = \left(R_{Y_{Gyro}}[n] - R_{Y_{Gyro}}[n - 1] \right) \cdot \Delta t \quad (5.14)$$

The final steps for the algorithm are to relate the rate of change of the X-Z and Y-Z angles from the accelerometer to the rate of change from the rate gyroscope. The first step is to produce a ratio based on the rate gyroscope's measurement. This will determine how well the accelerometer angle measurements are correlated. For instance, when the gyroscope readings match that of the accelerometer, the ratio will become one half, and the final output will have an equal contribution from both sensors. If the accelerometer readings are much smaller than the gyroscope, then the ratio will become one, and the algorithm will rely more heavily on the accelerometer results. If, however, the accelerometer results are much larger than the gyroscope, then the ratio will tend towards zero, causing the algorithm to rely more heavily on the gyroscope results. Note: since the Y-Z angle is the angle rotated about the X-axis, the Y-Z angle will be combined with the X-axis rate gyroscope reading. Similarly, the X-Z angle will be combined with the Y-axis rate gyroscope reading. The ratios are calculated as follows;

$$F_{X_{Ratio}} = \frac{|R_{X_{Gyro}}|}{|R_{X_{Gyro}}| + |A_{YZ_{Accel}}|} \quad (5.15)$$

$$F_{Y_{Ratio}} = \frac{|R_{Y_{Gyro}}|}{|R_{Y_{Gyro}}| + |A_{XZ_{Accel}}|} \quad (5.16)$$

The two sensors are now related using;

$$F_X[n] = (F_{X_{Ratio}} \cdot A_{YZ_{Accel}}) + \left[(1 - F_{X_{Ratio}}) \cdot (F_X[n - 1] + I_{X_{Gyro}}) \right] \quad (5.17)$$

$$F_Y[n] = (F_{Y_{Ratio}} \cdot A_{XZ_{Accel}}) + \left[(1 - F_{Y_{Ratio}}) \cdot (F_Y[n - 1] + I_{Y_{Gyro}}) \right] \quad (5.18)$$

The output will still have some high frequency noise, and should be filtered to represent smooth transitions in orientation. A simple moving average filter can be applied to adequately reduce system noise without causing a considerable lag in the system's response.

$$F_X[m] = \frac{1}{N} \sum_{n=0}^{N-1} F_X[m - n] \quad \text{where } \{N \leq m\} \quad (5.19)$$

$$F_Y[m] = \frac{1}{N} \sum_{n=0}^{N-1} F_Y[m - n] \quad \text{where } \{N \leq m\} \quad (5.20)$$

5.4 LAB EXERCISE & EXPERIMENTAL RESULTS

The alternative sensor fusion approach described above was based on observations while experimenting with the inertial sensors. This section describes how the experiments were created, and how the described algorithm was first implemented.

As mentioned in 3.1.1, the hardware used was the Technological Arts NanoCore12 module utilizing the Freescale 9S12C microcontroller (MCU) [23]. The code developed, was written using the C programming language. The integrated development software used to program the MCU was CodeWarrior.



Figure 29 NanoCore12 module MCU

The inertial measurement unit (IMU) used was a five degree-of-freedom (5 DOF) breakout board developed from SparkFun Electronics [24]. This board consists of the ADXL335 triple axis accelerometer and dual axis IDG500 rate gyroscope. This specific board was particularly useful because the output signals were ratio-metric analog voltages and allowed changes to be made to the filtering circuit. For instance, the first IMU used for experimenting had an incorrectly designed filter circuit for the rate gyroscope. The original configuration produces angular rate signals that were very difficult to integrate, leading to false assumptions about how the rate gyroscope operated. It was not until another IMU was implemented that a clear distinction was made between the two rate gyroscopes. Research of the original IMU's rate gyroscope revealed a design error of the filter circuit. This explained the differences between the two.

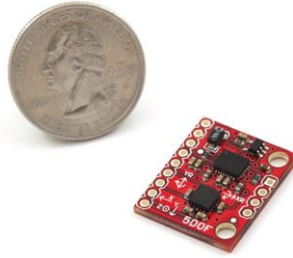


Figure 30 SparkFun 5 DOF IMU

The IMU was connected to the MCU’s docking module’s signal extension connector (as shown in the diagram below). With the IMU connected to the MCU, code could then be written to test for proper operation of the IMU. To help determine that the hardware was working properly, the MCU was used extensively in a special mode of operation called “Real Time Background Debug Mode.” This allowed the user to step through parts of the code and analyze the data result registers.

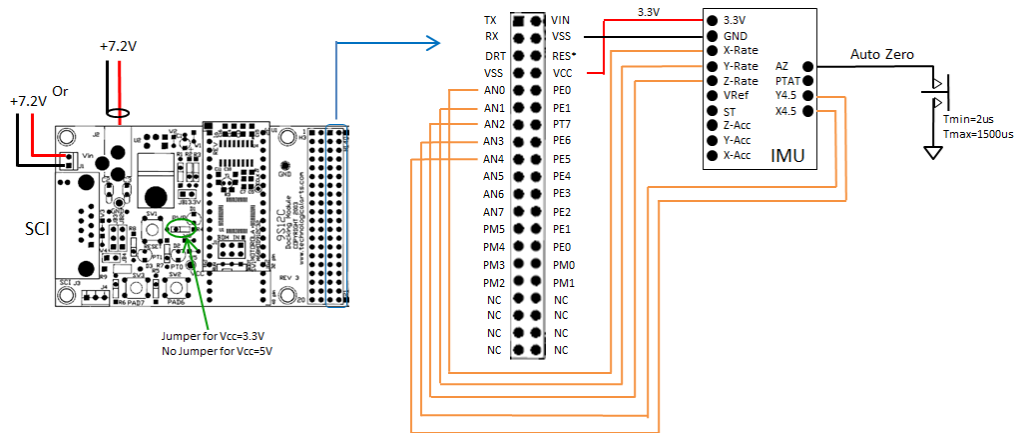


Figure 31 MCU and IMU Connection Diagram

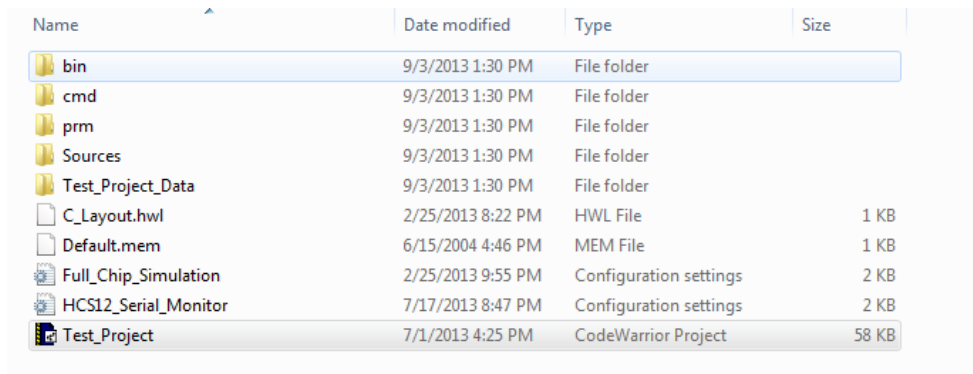
After the initial functional checks (to ensure that the MCU was able to monitor the IMU) the next step is to run the MCU in “Real Time” mode. This allows the MCU to send data

along the RS-232 serial communication interface (SCI) to the host computer. The data is then sent continuously to the computer's hyper-terminal to record information from the IMU to be analyzed later.

5.4.1 LAB EXERCISE PART 1 – PROGRAMMING THE MCU

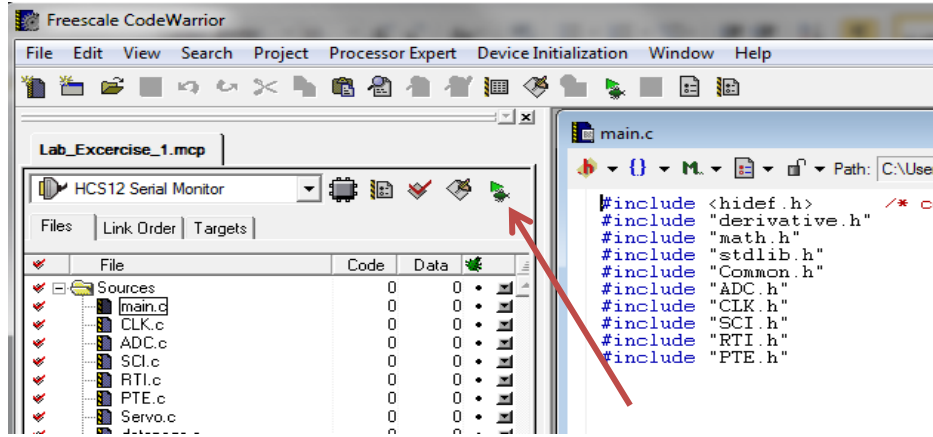
1. Navigate to the directory with the lab exercise folder.
2. Click on the “Code Warrior Project” (under the “Type” heading) to launch the application.

Example: The figure below shows the Code Warrior Project to be “Test_Project.”

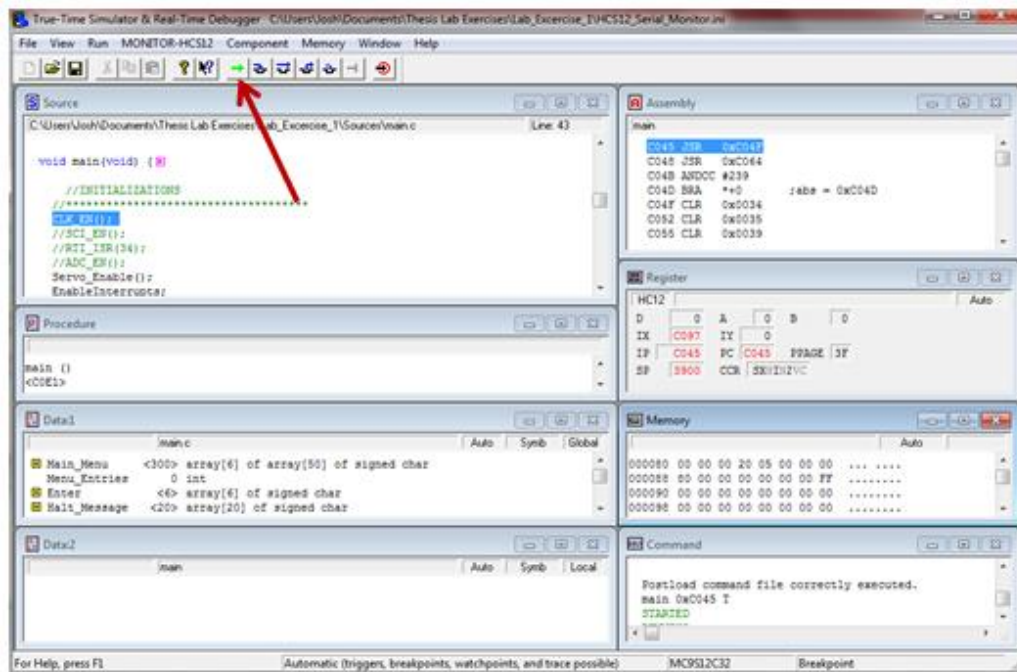


Name	Date modified	Type	Size
bin	9/3/2013 1:30 PM	File folder	
cmd	9/3/2013 1:30 PM	File folder	
prm	9/3/2013 1:30 PM	File folder	
Sources	9/3/2013 1:30 PM	File folder	
Test_Project_Data	9/3/2013 1:30 PM	File folder	
C_Layout.hwl	2/25/2013 8:22 PM	HWL File	1 KB
Default.mem	6/15/2004 4:46 PM	MEM File	1 KB
Full_Chip_Simulation	2/25/2013 9:55 PM	Configuration settings	2 KB
HCS12_Serial_Monitor	7/17/2013 8:47 PM	Configuration settings	2 KB
Test_Project	7/1/2013 4:25 PM	CodeWarrior Project	58 KB

3. Once CodeWarrior has opened up the project, click on the “Debug” button to program the microcontroller.

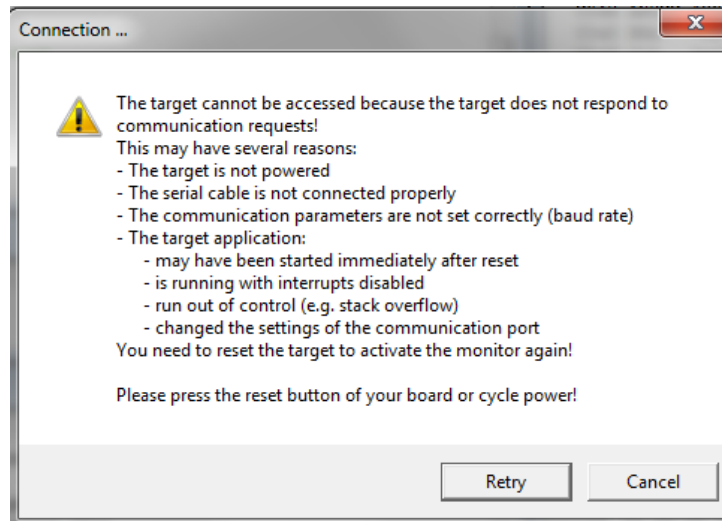


4. The “True-Time Simulator & Real-Time Debugger” should now be open. Press the “GREEN” arrow to start the program.



True-Time Simulator & Real-Time Debugger

Note: A “Connection...” warning window may appear if the target (MCU) has code to re-configure the clock speed and thus is no longer synchronized to the computer. If this occurs simply position the “Load/Run” switch to “RUN” and then press “RESET” on the project board.



5. Position the “Load/Run” switch to “RUN” and then press “RESET” on the project board.
6. Close out of the “True-Time Simulator & Real-Time Debugger” window.
7. Open the serial communication program installed on the computer.

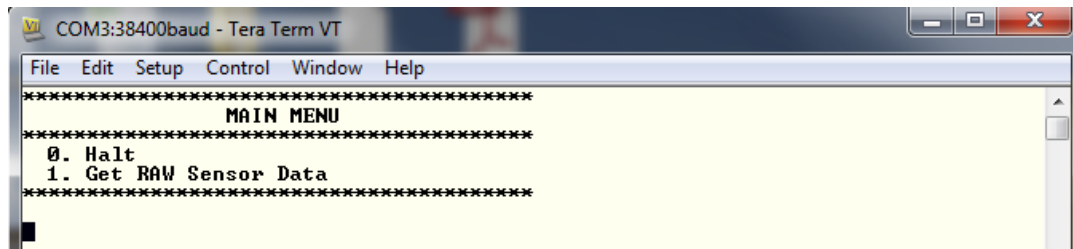
Note: *The Tera Term terminal emulator was used to communicate to the serial communication interface port, as Windows 7 doesn't have a HyperTerminal application installed.*

8. Configure the Serial Port as follows:
 - a. Locate the serial port connected to the project board
 - b. Under “Set-Up”, select “Serial Port” and set the following

- i. *Baud Rate: 38400*
- ii. *Data: 8 bit*
- iii. *Parity: none*
- iv. *Stop: 1 bit*
- v. *Flow control: none*

c. Press “OK”

9. Press “RESET” on the project board to restart communication. Once the board starts communicating, a “Main Menu” dialog should show up similar to the screenshot below;

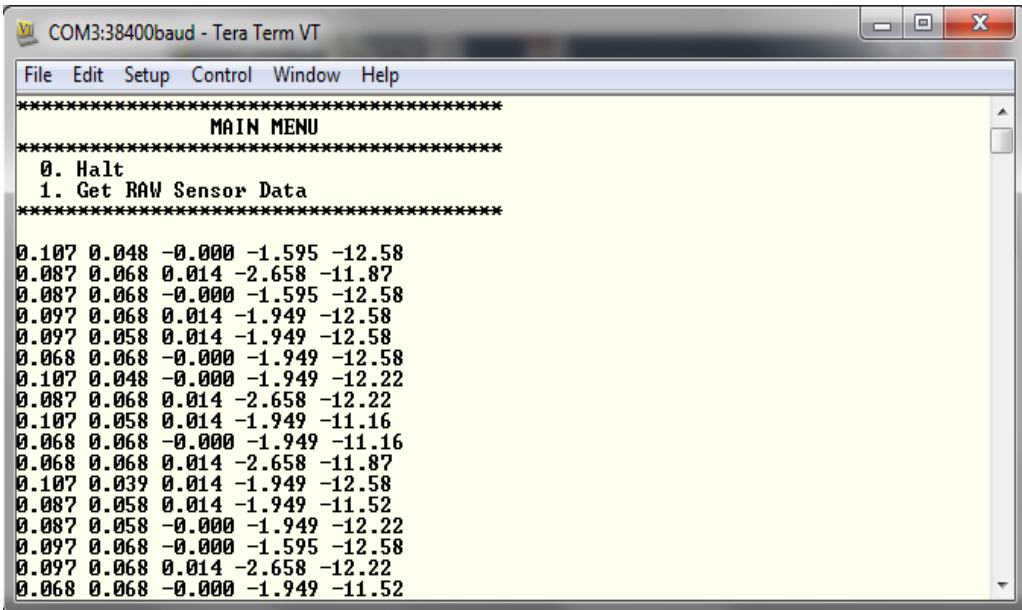


Example Main Menu Dialog

5.4.2 LAB EXERCISE PART 2 – DATA COLLECTION

1. At this point, the “Main Menu” should be displayed on the serial terminal. Before collecting data, the sensor will need to first be “zeroed” using a fixed level surface.
2. With the IMU sensor positioned on a flat level surface (not subjected to any movements), select the option to “Initialize”, “Null”, or “Zero” the Sensors.
3. Once the sensors have been initialized, select the option “Get RAW Sensor Data.”
The MCU should immediately start filling the terminal screen with data. Allow

the data to accumulate for approximately 10 seconds. The data will appear similar to the screenshot shown below.



Example Data

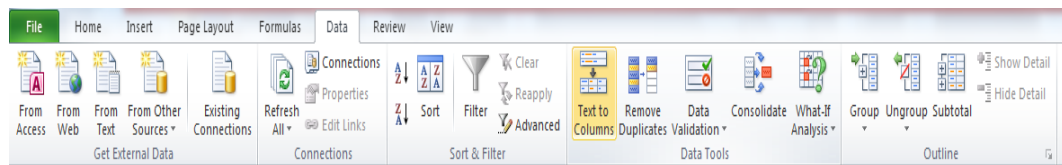
4. After the 10 seconds have passed, press "0" to stop updating the sensor values.
5. Since the data should have all similar values, we would like to record data of the IMU performing some kind of motion or rotation. Clear the screen on the serial terminal (typically under the "Edit" option at the top). Next, click on the option "Get RAW Sensor Data." After the data begins to update, wait approximately one second and then rotate the IMU board along any axis between ± 70 degrees of rotation. After about 8 seconds, position the IMU back to the level surface and avoid any further movement for an additional 2 seconds.
6. Press "0" to halt the data.

7. Go to “File” and then “Disconnect” to avoid forgetting to disconnect the SCI port when re-downloading code onto the microcontroller.

5.4.3 LAB EXERCISE PART 3 – DATA ANALYSIS

The data analysis portion of the lab experiment originally used Microsoft Excel; however, Matlab can instead be used. For this portion of the lab, we will be using Excel.

1. Copy the data from the HyperTerminal and paste into MS Excel.
2. To separate the data into their respective columns, click on the “Data” tab and then click on “Text to Columns.” Follow the instructions using space delimited.



3. Next, space out the columns for equations to be added.

Example

For the Accelerometer Data, Fusion Data and Rate Gyroscope Data;

Yacc							FUSION				Zacc		Xgyro				
Data	Unit Conversion	ATAN2	Rad-Deg	AVG	Derivative	AVG	Integral	Acc-Gyro		AVG	Data	Unit Conversion	Data	Unit Conversion	Integral	PW	Integral
502	-0.0929	-0.09967	-5.7106	-5.7106	0	0	-5.71059	1.0000	0.0000	-5.71059	-5.71059	622	0.9286	422	-2.508960573	-2.50896	7.8853E-07
502	-0.0929	-0.09967	-5.7106	-5.7106	0	0	-5.71059	1.0000	0.0000	-5.71059	-5.71059	622	0.9286	421	-1.792114695	-2.49319	0.014337706
503	-0.0831	-0.09018	-5.1669	-5.1669	27.182211	27.182211	-5.16695	0.0619	0.9381	-5.66352	-5.66352	621	0.9189	421	-1.792114695	-2.47742	0.014337706
502	-0.0929	-0.1018	-5.8326	-5.8326	-33.28074	-33.28074	-5.83256	0.0511	0.9489	-5.65855	-5.65855	620	0.9091	421	-1.792114695	-2.46165	0.014337706
502	-0.0929	-0.10072	-5.7709	-5.7709	3.0813762	3.0813762	-5.77094	0.3677	0.6323	-5.69081	-5.68087	621	0.9189	421	-1.792114695	-2.44587	0.014337706
501	-0.1026	-0.11243	-6.4416	-6.4416	-33.53319	-33.53319	-6.4416	0.0311	0.9689	-5.68635	-5.67481	620	0.9091	420	-1.075268817	-2.41433	0.028674624
502	-0.0929	-0.1029	-5.8955	-5.8955	27.304154	27.304154	-5.89552	0.0616	0.9384	-5.68578	-5.68037	619	0.8993	421	-1.792114695	-2.39856	0.014337706
503	-0.0831	-0.09114	-5.2222	-5.2222	33.665753	33.665753	-5.2222	0.0505	0.9495	-5.67384	-5.6842	620	0.9091	421	-1.792114695	-2.38279	0.014337706
502	-0.0929	-0.09967	-5.7106	-5.7106	-24.41956	-24.41956	-5.71059	0.0684	0.9316	-5.66356	-5.67739	622	0.9286	421	-1.792114695	-2.36702	0.014337706
503	-0.0831	-0.09114	-5.2222	-5.6578	24.41956	24.41956	-5.2222	0.0684	0.9316	-5.64981	-5.66825	620	0.9091	421	-1.792114695	-2.35125	0.014337706
502	-0.0929	-0.10072	-5.7709	-5.7333	-27.43671	-27.43671	-5.77094	0.0613	0.9387	-5.64147	-5.65717	621	0.9189	421	-1.792114695	-2.33548	0.014337706

Note: The gyroscope data has a column to plot the integral, where the algorithm only uses the integration results for each time step.

4. Apply the equations described in the proposed algorithm section (section 5.3.1).

Example

- a) Convert the ADC Hex number into units of force (Y-Accel and Z-Accel):

Clipboard Font Alignment

$=(((F3223*3.3)/1023)-1.65)/0.33$

	E	F	G	H	I	J	K	L	M	N
Yacc										
Data		Unit Conversion	ATAN2	Rad-Deg	AVG	Derivative	AVG	Integral		
	508	$=(((F3223*3.3)/1023)-1.65)/0.33$	-0.038	-2.1787	-2.1787	0	0	0		
	508	-0.0342	-0.038	-2.2026	-2.2026	-0.85422	-0.85422	-0.024		
	508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024		
	508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024		
	508	-0.0342	-0.038	-2.1553	-2.1553	1.69009	1.69009	0.0234		
	508	-0.0342	-0.038	-2.1553	-2.1553	0	0	0.0234		

Font Alignment Number Formatting

$=(((T3223*3.3)/1023)-1.7)/0.33$

	K	L	M	N	O	P	Q	R	S	T	U	V
					FUSION					Zacc		Xgy
Derivative	AVG	Integral		Acc+Gyro		AVG				Data	Unit Conversion	Dat
	0	0	0	1.0000	0.0000	-2.17868	-2.17868			619	$=(((T3223*3.3)/1023)-1.7)/0.33$	
	0.8542174	-0.8542174	-0.02392	0.8219	0.1781	-2.20191	-2.20191			618	0.8895	
	0	0	-0.02392	1.0000	0.0000	-2.2026	-2.2026			618	0.8895	
	0	0	-0.02392	1.0000	0.0000	-2.2026	-2.19645			618	0.8895	
	1.69009075	1.69009075	0.023404	0.6562	0.3438	-2.17154	-2.19466			620	0.9091	

b) Calculate the angle using ATAN2:

The screenshot shows the Excel formula bar with the formula `=ATAN2(U3223,G3223)` entered. The spreadsheet below shows a table with columns for Yacc Data, Unit Conversion, ATAN2, Rad-Deg, AVG, Derivative, AVG, Integral, FUSION Acc+Gyro, and Zacc Data. The ATAN2 column contains values calculated from the Unit Conversion and Derivative columns.

Yacc Data	Unit Conversion	ATAN2	Rad-Deg	AVG	Derivative	AVG	Integral	FUSION Acc+Gyro	Zacc Data
508	-0.0342	=ATAN2(U3223,G3223)	-2.1787	-2.1787	0	0	0	1.0000	619
508	-0.0342	-0.03844	-2.2026	-2.2026	-0.8542174	-0.8542174	-0.02392	0.8219	618
508	-0.0342	-0.03844	-2.2026	-2.2026	0	0	-0.02392	1.0000	618
508	-0.0342	-0.03844	-2.2026	-2.2026	0	0	-0.02392	1.0000	618
508	-0.0342	-0.03844	-2.2026	-2.2026	1.6900975	1.6900975	0.0234	0.6563	618

c) Convert from radians to degrees:

The screenshot shows the Excel formula bar with the formula `=DEGREES(H3223)` entered. The spreadsheet below shows the same table as in part b, but with the ATAN2 column values converted to degrees in the Rad-Deg column.

Yacc Data	Unit Conversion	ATAN2	Rad-Deg	AVG	Derivative	AVG	Integral
508	-0.0342	-0.038	=DEGREES(H3223)	-2.1787	0	0	0
508	-0.0342	-0.038	-2.2026	-2.2026	-0.85422	-0.85422	-0.024
508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024
508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024
508	-0.0342	-0.038	-2.1553	-2.1553	1.69009	1.69009	0.0234

d) Average the accelerometer derived angle:

The screenshot shows the Excel formula bar with the formula `=SUM(I3223:I3234)/12` entered. The spreadsheet below shows the same table as in part c, but with the average of the Rad-Deg column values calculated in the AVG column.

Yacc Data	Unit Conversion	ATAN2	Rad-Deg	AVG	Derivative	AVG	Integral
508	-0.0342	-0.038	-2.1787	-2.1787	0	0	0
508	-0.0342	-0.038	-2.2026	-2.2026	-0.85422	-0.85422	-0.024
508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024
508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024
508	-0.0342	-0.038	-2.1553	-2.1553	1.69009	1.69009	0.0234
508	-0.0342	-0.038	-2.1553	-2.1553	0	0	0.0234
508	-0.0342	-0.038	-2.1787	-2.1787	-0.83587	-0.83587	0
507	-0.0440	-0.043	-2.8310	-2.8310	-23.2972	-23.2972	-0.652
508	-0.0342	-0.038	-2.2026	-2.2026	22.443	22.443	-0.024
508	-0.0342	-0.038	-2.2026	-2.2026	0	0	-0.024
508	-0.0342	-0.038	-2.1787	-2.1787	0.85422	0.85422	0
508	-0.0342	-0.038	-2.1787	=SUM(I3223:I3234)/12	0	0	0
508	-0.0342	-0.038	-2.1553	-2.2372	0.83587	0.83587	0.0234
508	-0.0342	-0.038	-2.1787	-2.2352	-0.83587	-0.83587	0
508	-0.0342	-0.038	-2.1787	-2.2332	0	0	0

e) Compute the derivative of the data:

fx = $=(I1660-I1659)/0.02$

	E	F	G	H	I	J	K	L	M	N

f) Convert the ADC Hex number into units of angular rate:

X fx = $=(((W1659*3.3)/1023)-1.35)/-0.0045$

P	Q	R	S	T	U	V	W	X	Y	Z	AA

g) Calculate the integral of the angular rate:

Font Alignment Number
Y fx = $=Y1658+(X1659+2.509)*0.022$

	Q	R	S	T	U	V	W	X	Y	Z	AA

h) Calculate the integral of each time step (no accumulation of data):

Font		Alignment		Number					
fx		=(X1659+2.509)*0.02							
Q	R	S	T	U	V	W	X	Y	Z
			Zacc			Xgyro			
	AVG		Data	Unit Conversion		Data	Unit Conversion	Integral	PW Integral
71059	-5.71059		622	0.9286		422	-2.508960573	-2.50896	7.8853E-07
71059	-5.71059		622	0.9286		421	-1.792114695	-2.49319	=(X1659+2.50
66352	-5.66352		621	0.9189		421	-1.792114695	-2.47742	0.014337706

i) Perform the sensor fusion steps by calculating the ratio:

Font		Alignment		Number		Style		Cells					
X ✓ fx		=(ABS(X1659))/((ABS(X1659))+ABS(L1659))											
K	L	M	N	O	P	Q	R	S	T				
				FUSION			Zacc		Xgyro				
	Derivative	AVG	Integral	Acc+Gyro		AVG	Data	Unit Conversion	Data				
								Unit Conversion	Integral	PW Integral			
7106	0	0	-5.71059	1.0000	0.0000	-5.71059	-5.71059	622	0.9286	422	-2.508960573	-2.50896	7.8853E-07
7106	0	0	-5.71059	=(L1659)	0.0000	-5.71059	-5.71059	622	0.9286	421	-1.792114695	-2.49319	0.014337706
1669	27.182211	27.182211	-5.16695	0.0619	0.9381	-5.66352	-5.66352	621	0.9189	421	-1.792114695	-2.47742	0.014337706
8326	-33.28074	-33.28074	-5.83256	0.0511	0.9489	-5.65855	-5.65851	620	0.9091	421	-1.792114695	-2.46165	0.014337706
7709	3.0813762	3.0813762	-5.77094	0.3677	0.6323	-5.69081	-5.68087	621	0.9189	421	-1.792114695	-2.44587	0.014337706
4416	-33.53319	-33.53319	-6.4416	0.0311	0.9689	-5.68635	-5.67481	620	0.9091	420	-1.075268817	-2.41433	0.028674624

j) The accelerometer weight is 1-Ratio:

Font		Alignment		Number	
X ✓ fx		=1-O1659			
K	L	M	N	O	P
				FUSION	
	Derivative	AVG	Integral	Acc+Gyro	AVG
					Zacc
					Data
7106	0	0	-5.71059	1.0000	0.0000
7106	0	0	-5.71059	1.0000	=1-O1659
1669	27.182211	27.182211	-5.16695	0.0619	0.9381
8326	-33.28074	-33.28074	-5.83256	0.0511	0.9489

k) Fuse the data:

X ✓ fx $= (O1659*J1659)+(P1659*(Q1658+Z1659))$

	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
V2	Rad-Deg	AVG	Derivative	AVG	Integral		FUSION			Zacc			Xgyro						
							Acc+Gyro		AVG	Data	Unit Conversion		Data	Unit Conversion	Integral	PW	Integral		
9967	-5.7106	-5.7106	0	0	-5.71059		1.0000	0.0000	-5.71059	-5.71059	622	0.9286	422	-2.508960573	-2.50896	7.8853E-07			
9967	-5.7106	-5.7106	0	0	-5.71059		1.0000	0.0000	(1659)	-5.71059	622	0.9286	421	-1.792114695	-2.49319	0.014337706			
9018	-5.1669	-5.1669	27.182211	27.182211	-5.16695		0.0619	0.9381	-5.66352	-5.66352	621	0.9189	421	-1.792114695	-2.47742	0.014337706			
1018	-5.8326	-5.8326	-33.28074	-33.28074	-5.83256		0.0511	0.9489	-5.65855	-5.65851	620	0.9091	421	-1.792114695	-2.46165	0.014337706			
0072	-5.7709	-5.7709	3.0813762	3.0813762	-5.77094		0.3677	0.6323	-5.68087	-5.68087	621	0.9189	421	-1.792114695	-2.44587	0.014337706			
1243	-6.4416	-6.4416	-33.53319	-33.53319	-6.4416		0.0311	0.9689	-5.68635	-5.67481	620	0.9091	420	-1.075268817	-2.41433	0.028674624			

l) Calculate the average of the fused set:

X ✓ fx $= \text{SUM}(Q1658:Q1661)/4$

	I	J	K	L	M	N	O	P	Q	R	S	T	U
ATAN2	Rad-Deg	AVG	Derivative	AVG	Integral		FUSION				Zacc		
							Acc+Gyro		AVG	Data	Unit Cor		
-0.09967	-5.7106	-5.7106	0	0	-5.71059		1.0000	0.0000	-5.71059	-5.71059	622		
-0.09967	-5.7106	-5.7106	0	0	-5.71059		1.0000	0.0000	-5.71059	-5.71059	622		
-0.09018	-5.1669	-5.1669	27.182211	27.182211	-5.16695		0.0619	0.9381	-5.66352	-5.66352	621		
-0.1018	-5.8326	-5.8326	-33.28074	-33.28074	-5.83256		0.0511	0.9489	-5.65855	=SUM(Q1	620		
-0.10072	-5.7709	-5.7709	3.0813762	3.0813762	-5.77094		0.3677	0.6323	-5.68087	-5.68087	621		

5. Plot the following results onto a single graph;

- Y-Axis (or X-Axis) accelerometer data
- X-Axis (or Y-Axis) rate gyroscope integration data
- Y-Axis (or X-Axis) data fusion results

Lab Exercise Questions:

1) Describe the changes in orientation for the accelerometer and rate gyroscope while recording the data on the serial port. After applying the algorithms of the

spreadsheet to your dataset, did the results match the applied orientations?

Describe your results.

- 2) Apply the spreadsheet equations, from Part 3, to the x-axis (pitch) measurements.

Plot the results and describe if they match the applied orientations?

- 3) Describe how to incorporate the results of the x and y-axis into a rotation matrix?
- 4) How would you avoid Euler angle singularity problems?
- 5) Describe how accumulation errors from the rate gyroscope are eliminated.

5.4.4 EXPERIMENTAL TEST RESULTS

Figure 32 and Figure 33 show the results of data gathered while performing the above experiment. The data represents the IMU being rotated about the Y-Axis (pitch axis) while mounted to an apparatus. The apparatus included an unbalanced motor to simulate vibrational noise. These results demonstrate the effectiveness of my alternative proposed sensor fusion algorithm.

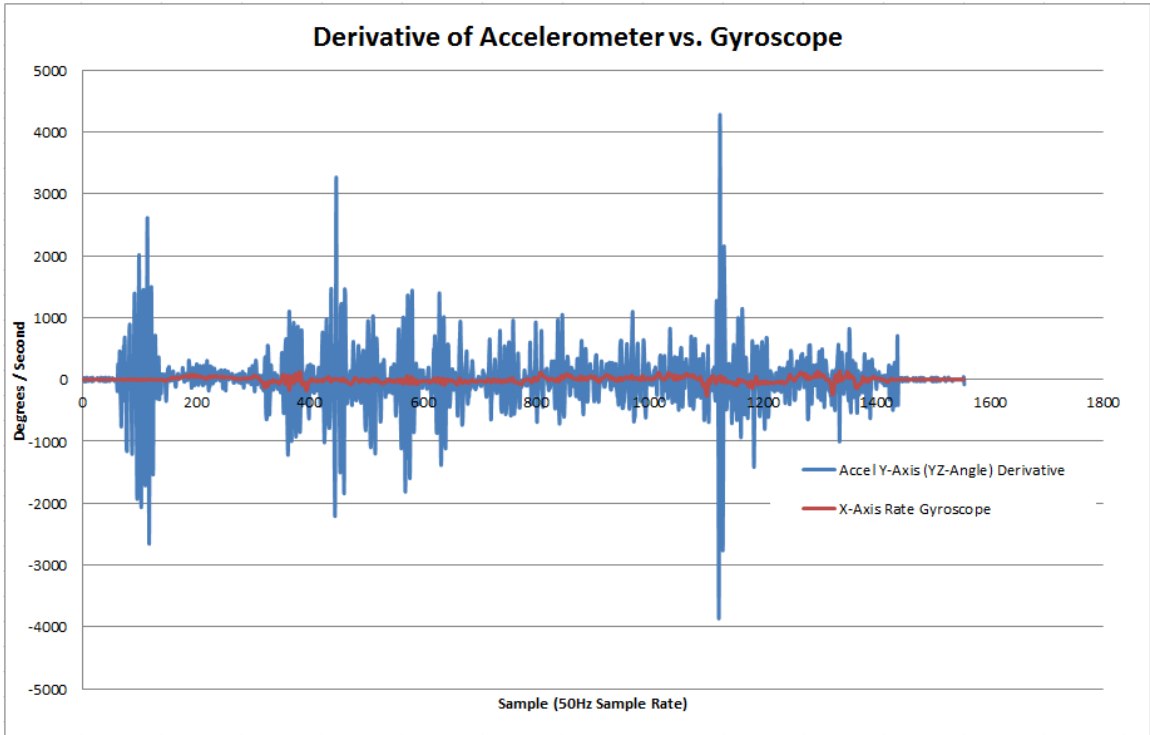


Figure 32 Accelerometer YZ-Angle Derivative vs. Rate Gyroscope

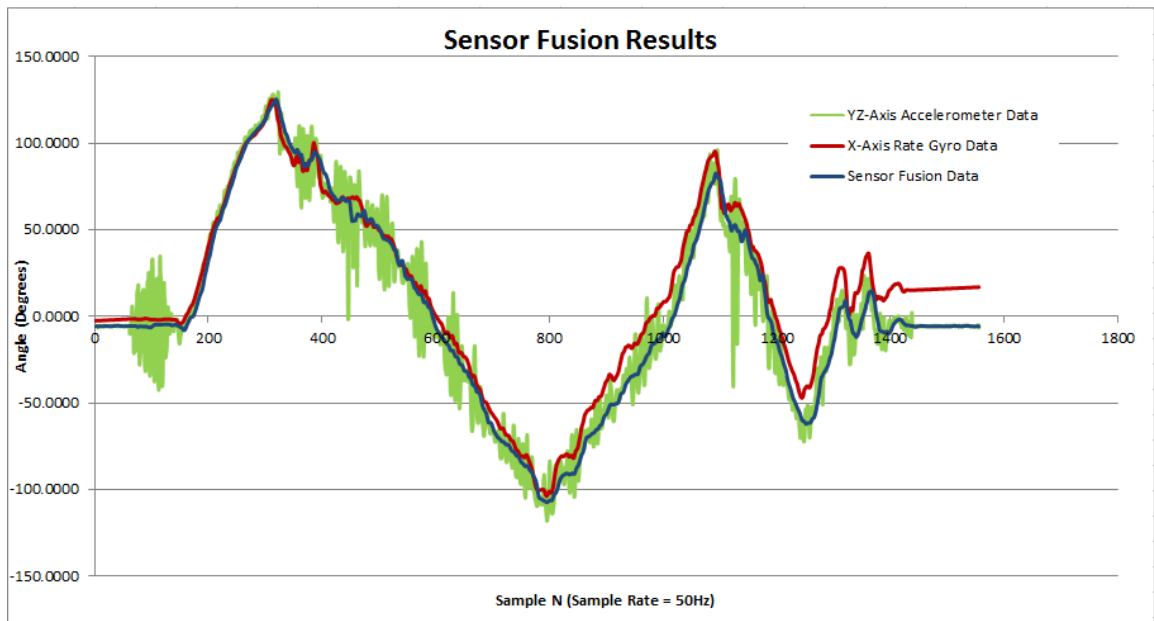


Figure 33 Sensor Fusion Results

5.5 SUMMARY

The proposed alternative sensor fusion method was a way to combine data from both an accelerometer and rate gyroscope to produce a single output representative of the actual orientation. This method was based on observations made while experimenting with the independent behavior of the accelerometer and rate gyroscope sensors. The primary environment considered was onboard a small UAV where vibrational noise (caused by the motor and propeller) can create significant noise errors from the accelerometers.

While working with a UAV (similar to that of the Multiplex EasyStar), I had observed that the aircraft would reach a steady cruising speed when flying straight and level. In this condition, the net acceleration measured would be the gravitational force vector. This led me to the realization that the accelerometer can be used to correct bias offset accumulation errors from the gyroscope. Similarly, when the aircraft undergoes maneuvers, such as coordinated turns or sudden changes in attitude, the accelerometer will provide orientation data mixed with translational accelerations. Since the accelerometer cannot differentiate between the two forms of acceleration, the algorithm uses the gyroscope to provide reliable orientation information (separating out changes in orientation from other external forces). This characteristic could be beneficial as another algorithm can use the information about the external forces to detect other properties about flight, such as slipping or skidding through turns.

In summary, the fusion properties of the proposed algorithm helped remove noise from the accelerometer's measurements by relying on the noise immunity properties of

the rate gyroscope. In addition, the algorithm helped remove integration errors, caused by the rate gyroscope, by relying on the bias stability of the accelerometer.

5.6 FUTURE WORK

The alternative sensor fusion approach produces output angles representative of the orientation. Since the Kalman filter also provides an output orientation representation, it would be a good exercise to compare the results of the two methods with known “true” data. This would provide useful insight into the performance of the alternative approach. Once the performance characteristics are known, then the alternative approach could be implemented into the Kalman filter.

The development of the state transition matrix for the alternative approach will most likely be different from what is typically implemented by the Kalman filter. Therefore, it may be possible to model the dynamics using standard equations of motion, taking into account the dynamics of the aircraft. By modeling the dynamics of the aircraft, the fusion algorithm will lead to results that are more accurate. Further development of the Kalman filter can include other navigational information (such as data from a GPS or magnetic compass) leading to the beginning stages of an autopilot system.

BIBLIOGRAPHY

- [1] J. D. Barton, “Fundamentals of Small Unmanned Aircraft Flight,” in John Hopkins APL Technical Digest, Vol. 31, No 2, 2012. [Online] Available: http://www.jhuapl.edu/techdigest/TD/td3102/31_02-Barton.pdf
- [2] C. J. Fisher, “Using an Accelerometer for Inclination Sensing,” in Tech. Rep., AN-1057, [Online]. Available: <http://www.analog.com/media/en/technical-documentation/application-notes/AN-1057.pdf>
- [3] S. Ayub *et al.*, “A Sensor Fusion Method for Smart phone Orientation Estimation,” 2012. [Online] Available: <http://www.cms.livjm.ac.uk/pgnet2012/Proceedings/Papers/1569603133.pdf>
- [4] K. W. Eure *et al.*, “An Application of UAV Attitude Estimation Using a Low-Cost Inertial Navigation System,” NASA, Washington, DC, Tech. Rep., TM-2013-218144, Dec. 2013. [Online] Available: <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20140002398.pdf>
- [5] Wikipedia, the free encyclopedia, *Radio-Controlled Aircraft*, [Online]. Available: https://en.wikipedia.org/wiki/Radio-controlled_aircraft
- [6] Wikipedia, the free encyclopedia, *Polypropylene*, [Online]. Available: <https://en.wikipedia.org/wiki/Polypropylene>
- [7] Advanced Avionics Handbook, US DoT FAA, Washington, DC, FAA-H-8083-6, 2009. [Online] Available: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/advanced_avionics_handbook/media/aah_ch04.pdf
- [8] Serendi-CDI, *Mass-Spring-Damper System – Physics*, [Online]. Available: http://serendi-cdi.org/serendipedia/index.php?title=Mass-Spring-Damper_System_-_Physics
- [9] A. Nistico, “Working Principle of a Capacitive Accelerometer,” Eng. Sci., Univ. of Roma Tor Vergata, 2013. [Online] Available: http://engineering-sciences.uniroma2.it/MENU/DOWNLOAD/TESI/2013/2013_tesi%20NISTICO%20Andrea.pdf
- [10] Wikipedia, the free encyclopedia, *atan2*, [Online]. Available: <https://en.wikipedia.org/wiki/Atan2>

- [11] The Simple Free Gyroscope, Dept. of Phys., Univ. of Guelph, [Online]. Available: <http://www.physics.uoguelph.ca/~orbax/phys2440/Gyroscope.pdf>
- [12] Coriolis Force, [Online]. Available: http://www.geo.cornell.edu/ocean/p_ocean/ppt_notes/13_Coriolis.pdf
- [13] Wikipedia, the free encyclopedia, *Euler Angle*, [Online]. Available: https://en.wikipedia.org/wiki/Euler_angles#Intrinsic_rotations
- [14] O. J. Woodman, "An introduction to inertial navigation," Comp. Lab, Univ. of Cambridge, Cambridge, UK, Tech. Rept. UCAM-CL-TR-696, Aug. 2007. [Online] Available: <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf>
- [15] G. G. Slabaugh, "Computing Euler angles from a rotation matrix," [Online] Available: <http://staff.city.ac.uk/~sbbh653/publications/euler.pdf>
- [16] F. Ramsey, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation," in IEEE SIGNAL PROCESSING MAGAZINE, Sept. 2012. [Online] Available: <http://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf>
- [17] D. Simon, Optimal State Estimation, Hoboken, New Jersey: John Wiley & Sons, Inc., 2006.
- [18] T. Lacey, "Tutorial: The Kalman Filter," [Online] Available: <http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf>
- [19] State Estimation with a Kalman Filter, [Online] Available: <https://courses.cs.washington.edu/courses/cse466/11au/calendar/14-StateEstimation-posted.pdf>
- [20] The Physics Hypertextbook, "Aerodynamic Drag," [Online] Available: <http://physics.info/drag/>
- [21] N. H. Q. Phuong *et al.*, "A DCM Based Orientation Estimation Algorithm with an Inertial Measurement Unit and a Magnetic Compass," in Journal of Universal Computer Science, Vol. 15, No 4, 2009. [Online] Available: http://www.jucs.org/jucs_15_4/a_dcm_based_orientation/jucs_15_04_0859_0876_phuong.pdf

- [22] S. Baluta, (2009). *A Guide to using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications*, [Online]. Available: http://www.starlino.com/imu_guide.html
- [23] Technological Arts, NC12DXC32 Kit 1, Product Details, [Online] Available: <http://www.technologicalarts.ca/shop/store/details/394/100/9s12/nanocore12/packages/nc12dxc32-kit-1.html>
- [24] SparkFun Electronics, *IMU Analog Combo Board – 5 Degrees of Freedom IDG500 / ADXL 335*, Datasheet, [Online]. Available: <https://www.sparkfun.com/products/11072>
- [25] Analog Devices, ADXL335 3-Axis $\pm 3g$ Accelerometer, Datasheet, [Online]. Available: <https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf>
- [26] InvenSense, IDG500 2-Axis Rate Gyro, Datasheet, [Online]. Available: https://www.sparkfun.com/datasheets/Components/SMD/Datasheet_IDG500.pdf
- [27] Electronic-Tutorials, “Wheatstone Bridge,” [Online] Available: <http://www.electronics-tutorials.ws/blog/wheatstone-bridge.html>
- [28] M. Andrejašič, “MEMS ACCELEROMETERS,” Seminar, Dept. of Physics, Univ. of Ljubljana, Mar. 2008. [Online] Available: http://mafija.fmf.uni-lj.si/seminar/files/2007_2008/MEMS_accelerometers-koncna.pdf
- [29] Ryono, “Gyroscopic Motion,” Lab Exercise, Dept. of Physics, Univ. of Guelph. [Online] Available: <http://ryono.net/apphysics/rotation/nutations.pdf>
- [30] DatWiki, “Rate Gyro,” [Online] Available: <http://www.datwiki.net/page.php?id=6532&find=rate%20gyro&searching=yes>
- [31] encyclopedia2.thefreedictionary, “Gyroscope,” [Online] Available: <http://encyclopedia2.thefreedictionary.com/Gyroscopic+intertia>
- [32] Nashvillecfi, “Attitude Indicator,” [Online] Available: <http://www.nashvillecfi.com/instrument/instruments.html>