MAGNETIC ATTITUDE ESTIMATION OF A TUMBLING SPACECRAFT

A Thesis

presented to

the Faculty of

California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Aerospace Engineering

by

Erick Jonathan Sturm II

July 2005

APPROVAL PAGE

TITLE:	Magnetic Attitude Estimation of a Tumbling Spacecraft
AUTHOR:	Erick Jonathan Sturm II
DATE SUBMITTED:	7/21/2005

Dr. Jordi Puig-Suari	
Advisor	Signature
Dr. John Mottmann	
Committee Member	Signature
Dr. Eric Mehiel	
Committee Member	Signature

ABSTRACT

MAGNETIC ATTITUDE ESTIMATION OF A TUMBLING SPACECRAFT

Erick Jonathan Sturm II

This thesis is a compilation of the work done to arrive at a ground-based attitude estimation routine for a tumbling spacecraft using only data from its magnetometers. While the essential component for attitude estimation is the extended Kalman filter, development of the supporting components, an orbit propagator and a satellite emulator, is also provided in an attempt to provide a comprehensive understanding of the entire routine. Each component was coded in Matlab, which was used to run simulations of the entire attitude estimation routine. The results of each simulation guided the tuning of the Kalman filter, ultimately the Kalman filter obtained 100% convergence to the spacecraft's attitude in less than 1.5 orbits to within ±10º.

ACKNOWLEDGEMENTS

Foremost, my parents, thank you for your support throughout my years at Cal Poly and all my life, I would not have been able to accomplish any of this without you. Spencer, had you not kept pestering me about PolySat and the coolness of Kalman filters I would still be searching for a thesis topic; thank you for being my boss and my friend. Dr. P, thank you for taking the time to always sit down with me and a white board to do some real engineering work, without which there would be no PolySat Kalman filter. Erin & Ken, my ADC protégés and friends, our extended lunch breaks kept me sane in that lab with no windows, thank you for all the little thankless jobs you did for me. Finally, Ellen, your patience, your support, and your love kept me going, I cannot thank you enough.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

ADC	Attitude Determination and Control
ADS	Attitude Determination System
ECEF	Earth-Centered Earth-Fixed
ECI	Earth-Centered Inertial
EKF	Extended Kalman Filter
EOM	Equation of Motion
GMST	Greenwich Mean Sidereal Time
IGRF	International Geomagnetic Reference Field
LNED	Local North East Down
MBMV	Magnetic Body Measurement Vector
MBRV	Magnetic Body Reference Vector
MIRV	Magnetic Inertial Reference Vector
MME	Minimum Model Error
MMSE	Minimum Mean Square Error
MSTL	Multidisciplinary Space Technology Laboratory
NASA	National Aeronautics and Space Administration
PEKF	PolySat Extended Kalman Filter
P-POD	Poly-Picosatellite Orbital Deployer
RAAN	Right Ascension of the Ascending Node
SBEV	Solar Body Estimated Vector
SBRV	Solar Body Reference Vector
SCO	Spacecraft-Centered Orbital
SIRV	Solar Inertial Reference Vector
SPRP	Solar Panel Reference Power
SSDL	Space Systems Development Laboratory
TLE	Two Line Element
UT1	Universal Time

LIST OF SYMBOLS

Upper Case	Definition	Units
E	Eccentric Anomaly	Radians
F	Linear Dynamics Matrix	N/A
H	Linear Measurement Matrix	N/A
I	Identity Matrix	N/A
J	Moment of Inertia Matrix	kg-m^2
JD	Julian Date	Days
K	Kalman Gain Matrix	N/A
P	Covariance Matrix	N/A
Q	Process Noise Matrix	N/A
R	Measurement Noise Matrix	N/A
T_s	Sampling Time	Seconds
T_U	Time from 1/1/00	Julian Centuries
Lower Case		
b	Magnetic Field	Gauss
c( )	Cosine	N/A
e	Eccentricity	N/A
h	Specific Angular Momentum	km^2/sec
n	Mean Motion	rad/sec
q	Quaternion	N/A
s( )	Sine	N/A
t	Time	sec
v	Measurment Noise	N/A
w	Process Noise	N/A
x	State Vector	N/A
z	Measurement Vector	N/A
Greek		
ω	Angular Rate	rad/sec
Ω	Right Ascension of the Ascending Node	Degrees
ν	True Anomaly	Radians
θ	An Angle	Radians or
ϕ	Longitude	Degrees
η	Noise	N/A
Φ	State Transition Matix	N/A
Δ	A Change In	N/A
δ	A Small Change In	N/A
σ	Standard Deviation	N/A

Overscript		
$\rightarrow$	Vector	N/A
$\sim$	Reduced State	N/A
$-$	Four Component Vector	N/A
$\wedge$	Estimate	N/A
Superscript		
err	Error	N/A
Body	In the Body Frame	N/A
T	Transpose	N/A
+	A Posteriori	N/A
-	A Priori	N/A
Subscript		
k	The kth element of the time vector	N/A
ve	Vernal Equinox	N/A
1	First Vector Component	N/A
2	Second Vector Component	N/A
3	Third Vector Component	N/A
4	Fourth Vector Component	N/A
$\oplus$	Earth	N/A

CHAPTER 1: INTRODUCTION

1.1 Background

1.1.1 CubeSat

Established as a collaborative effort between the Multidiciplinary Space Technology Laboratory (MSTL) of California Polytechnic State University (Cal Poly) and the Space Systems Development Laboratory (SSDL) of Stanford University in 1999, the CubeSat Project developed a new satellite standard for the design of research and educational picosatellites. The goals of the standard were to reduce development time of a satellite to within a student's college career, as well as the development and launch costs of the satellite. The resulting design concept specified small cubic satellites, 10cm on a side, weighing less than a kilogram, to become affectionately know as cubesats. CubeSat is also the developer of the Poly-Picosatellite Orbital Deployer (P-POD). The P-POD holds three cubesats and serves as the interface between them and the launch vehicle. The P-POD eliminates the need for cubesat developers to design custom interfaces to the launch vehicles, helping CubeSat meet its standard of lower launch costs and shorter development timelines.

1.1.2 PolySat

Founded in 2000, PolySat is Cal Poly's cubesat developer and is also based in the MSTL. PolySat began in order to better understand cubesat development and to help CubeSat better serve other cubesat developers by having one based in their own lab. Currently, PolySat has built two satellites, CP-1 and CP-2. Both are awaiting a launch on a Russian launch vehicle, and will hopefully be in orbit sometime late 2005 or early 2006. This thesis focuses on the attitude determination system (ADS) of CP-2, and as such, more details on CP-2 will be provided in the following section.

1.1.3 CP-2

CP-2 is the result of PolySat's desire to provide a generic cubesat bus that could allocate one third of its mass, power, and volume to a nonspecific payload. The bus is also to provide minimal attitude determination and control (ADC) capability for the payload. CP-2 seeks to provide this minimal ADC through the use of five two-axis Honeywell HMC1052 magnetometers and five magnetorquers; however the magnetorquers are used simply to slow the satellites tumble and not as controller actuators. This thesis is the product of an attempt to quantify what "minimal" means from an attitude determination stand point, when the spacecraft is tumbling.

1.2 Thesis Overview

1.2.1 Thesis Motivation

In addition to quantifying the attitude determination system of CP-2 by predicting its performance, the work presented in this thesis will be used with real flight data for attitude determination of CP-2 while in orbit. Also, keeping the work as generic as possible to allow for use in other non-linear, single vector measurement spacecraft systems shaped the development of this thesis. Thus, the attitude estimation routine presented herein has been kept as generic as possible.

1.2.2 Thesis Framework

The attitude estimation routine presented herein is ground-based. CP-2 lacks the processing power or the need for on-board real-time attitude information. The ground-based routine makes both batch and recursive filtering a possibility for attitude estimation, though recursive is chosen here. Processing power is not a limiting factor in a ground-based routine. Another benefit of having the attitude estimates on the ground is that they can be combined with other downloaded data for different subsystem model validation.

All the components of the routine are coded in Matlab. Matlab is not the most time efficient medium in which to run code, yet there is no need for extremely short running times with a ground-based system. Matlab has the benefits of an integrated differential equation solver and built-in graphics. Matlab has also become a fairly universal simulation tool in the aerospace industry. The aerospace engineering students in PolySat are also very familiar with Matlab, and writing code that is easier for others in the lab to understand is essential to project success. Spencer Studley goes into more depth on essential practices for a successful cubesat development program in his thesis, "Development of a Long-Term Student-Run Picosatellite Program."

1.2.3 Thesis Flow

This thesis is a compilation of the work done to arrive at an attitude estimation routine for a tumbling spacecraft equipped with magnetometers. While the essential component of attitude estimation is the extended Kalman filter, development of the supporting components is also provided in an attempt to provide a deeper understanding of the entire routine. Chapter 2 discusses the different reference frames used in the routine and the matrices necessary to transform from one frame to the next. From there the chapters try to mimic the flow of the actual routine, as pictured in Figure 1.1.

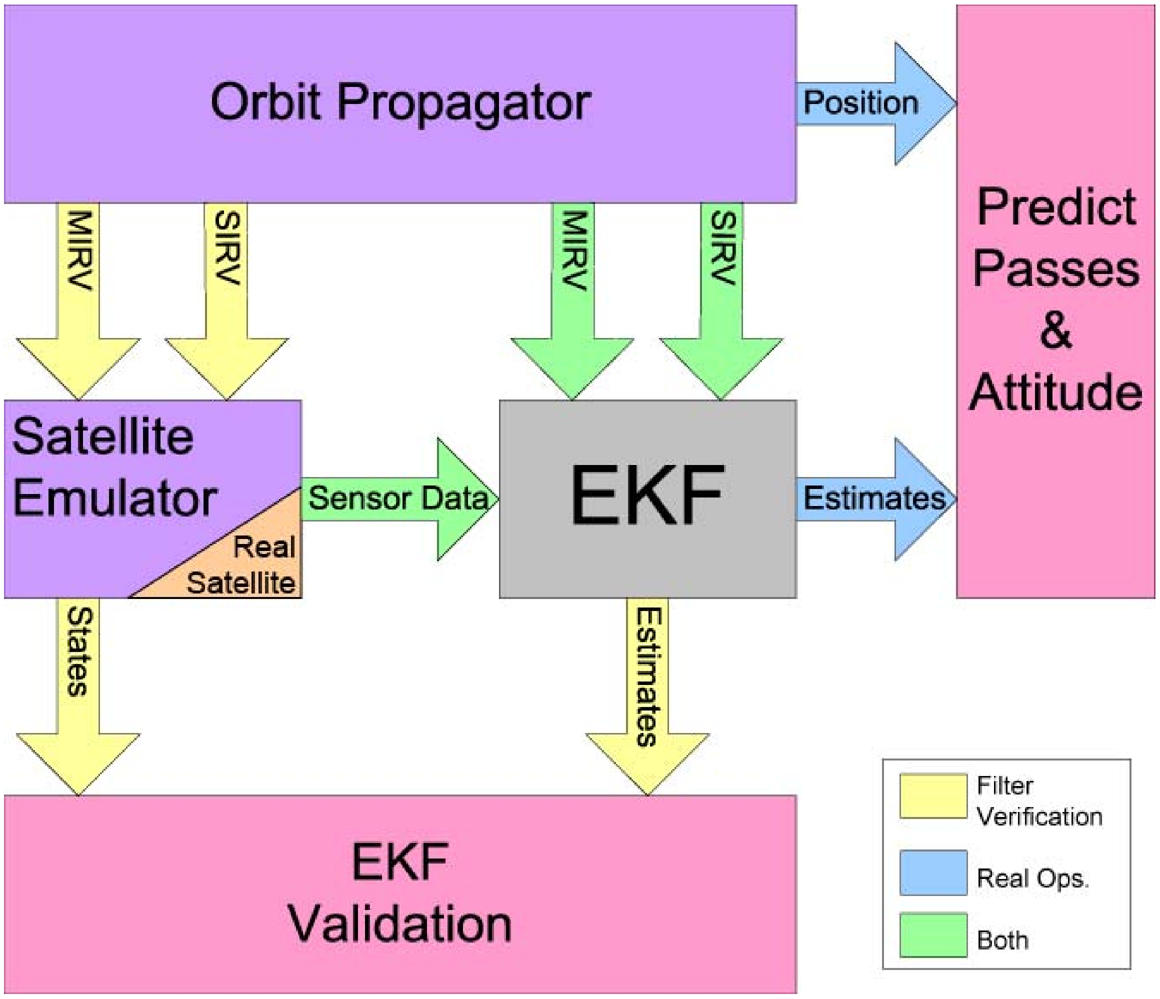

Figure 1.1: Attitude Estimation Routine Flow Chart

One should note that Figure 1.1 shows the flow for both simulated and real spacecraft sensor data. This thesis uses the "Filter Verification" path for the obvious reasons of checking to make sure the components of the routine are functioning properly and the fact that CP-2 is not yet in orbit. Following that path, Chapter 3 covers the development of the orbit propagator and the generation of the both the magnetic and solar inertial reference vectors, MIRV and SIRV respectively. Chapter 4 discusses the ADS hardware and how it was characterized and modeled in Matlab for use in the satellite emulator. Chapter 5 covers the satellite emulator and how it creates the necessary sensor data for the extended Kalman filter. Chapter 6 goes in depth on the theory, development, and performance of the extended Kalman filter. Chapter 7 summarized the work discussed and suggests future work that can be done with the routine.

CHAPTER 2: REFERENCE FRAME DEFINITIONS

2.1 Coordinate Frames

2.1.1 Space-Craft Orbital

The spacecraft orbital (SCO) frame is formed at the location of the spacecraft with the x-axis aligned with the radial vector, the z-axis aligned with the orbital angular momentum vector, and y-axis formed from the right-hand rule from the x- and z- axes. Figure 2.1 shows a drawing of the SCO frame.

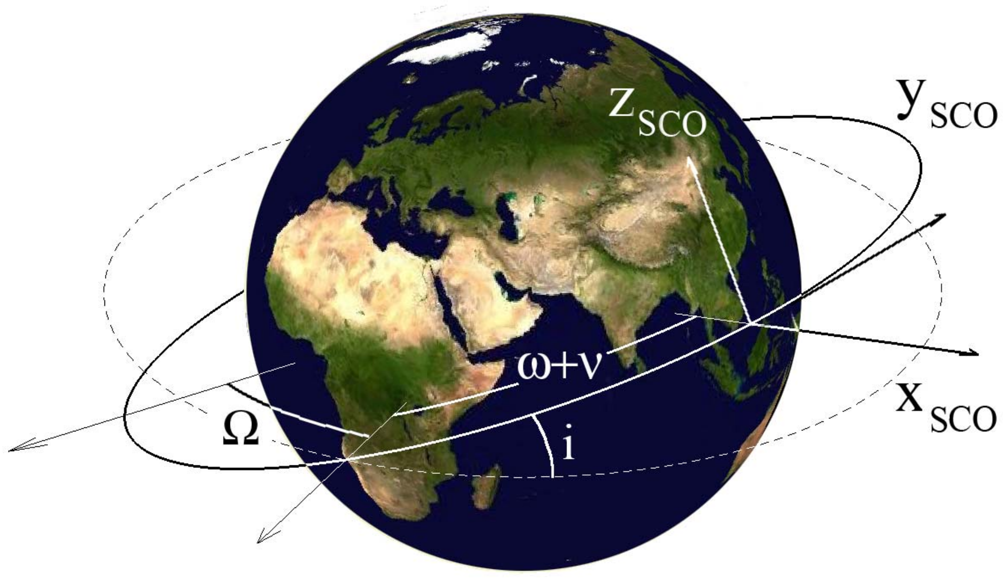

Figure 2.1: Spacecraft-Centered Orbital Frame

2.1.2 Earth-Centered Inertial

The Earth-Centered Inertial (ECI) frame is centered on the Earth and fixed in inertial space. The x-axis is aligned with the radial vector from the Sun to the Earth on the vernal equinox, also known as the line of Aries. The z-axis is aligned with the Earth's orbital angular momentum vector. Again the y-axis is formed from the right-hand rule between the x- and z- axes. Figure 2.2 shows a drawing of the ECI frame.

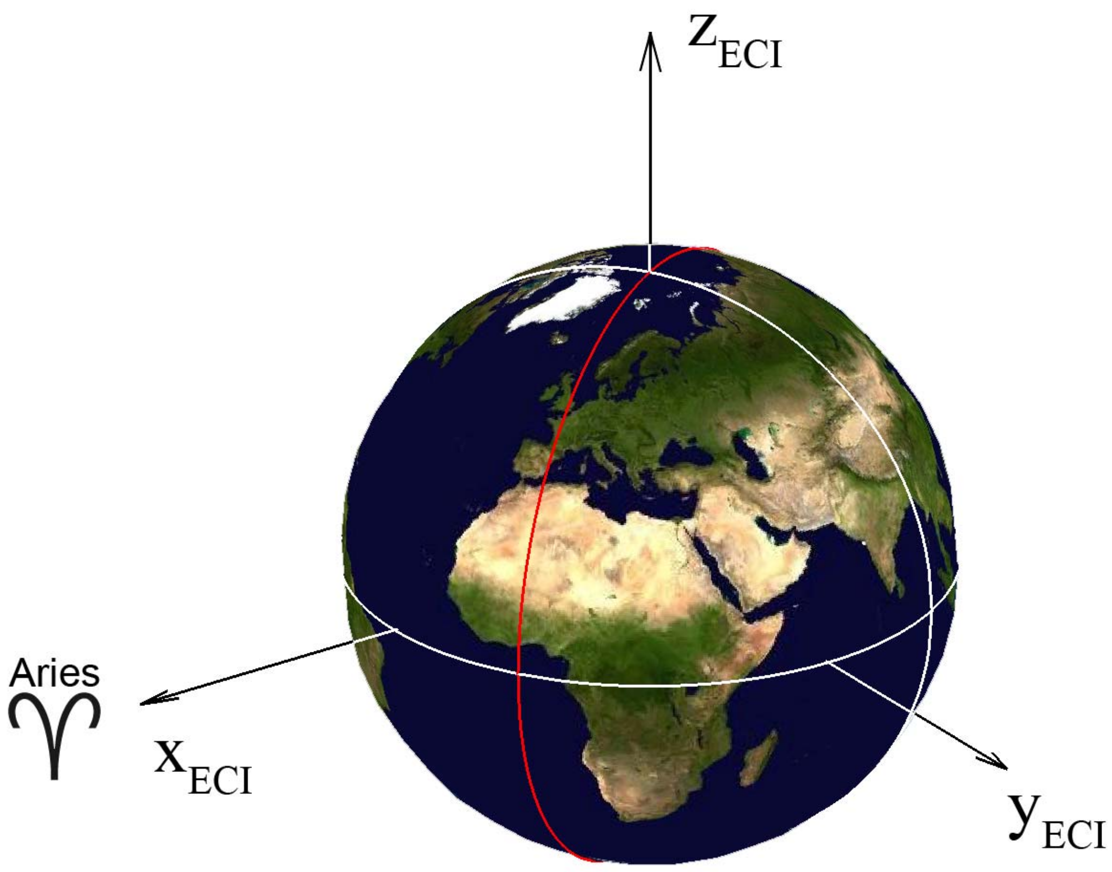

Figure 2.2: Earth-Centered Inertial Frame

2.1.3 Earth-Centered Earth-Fixed

The Earth-Centered Earth-Fixed (ECEF) frame rotates at the sidereal period of the Earth with respect to the ECI frame. The x-axis is aligned with the intersection of the prime meridian and the equator, the z-axis is aligned with the North Pole, and the y-axis is aligned with the intersection of the equator and 90ºE Longitude. Figure 2.3 shows a drawing of the ECEF frame.

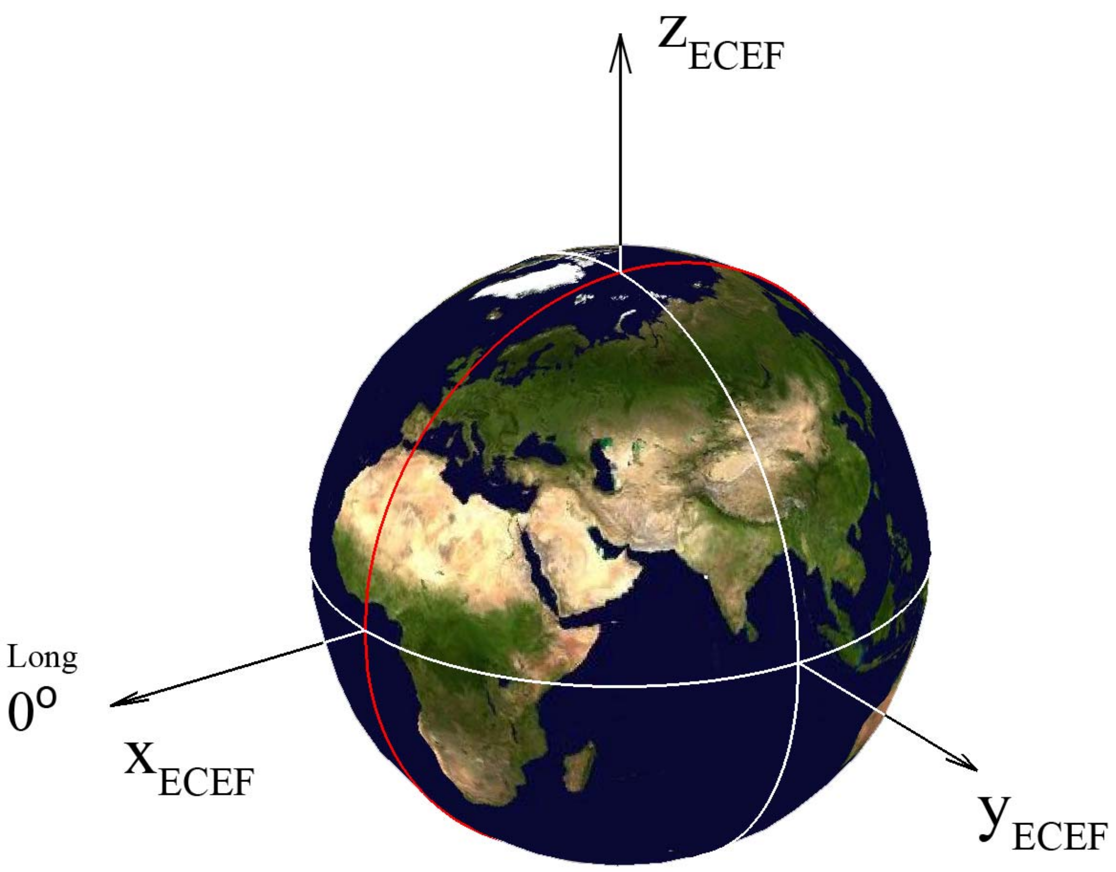

Figure 2.3: Earth-Centered Earth-Fixed Frame

2.1.4 Local North East Down

The Local North East Down (LNED) frame is fairly self-explanatory. The x-axis is aligned with local north, the y-axis with local east, and the z-axis aligned against the local radius vector. Figure 2.4 shows a drawing of the LNED frame.

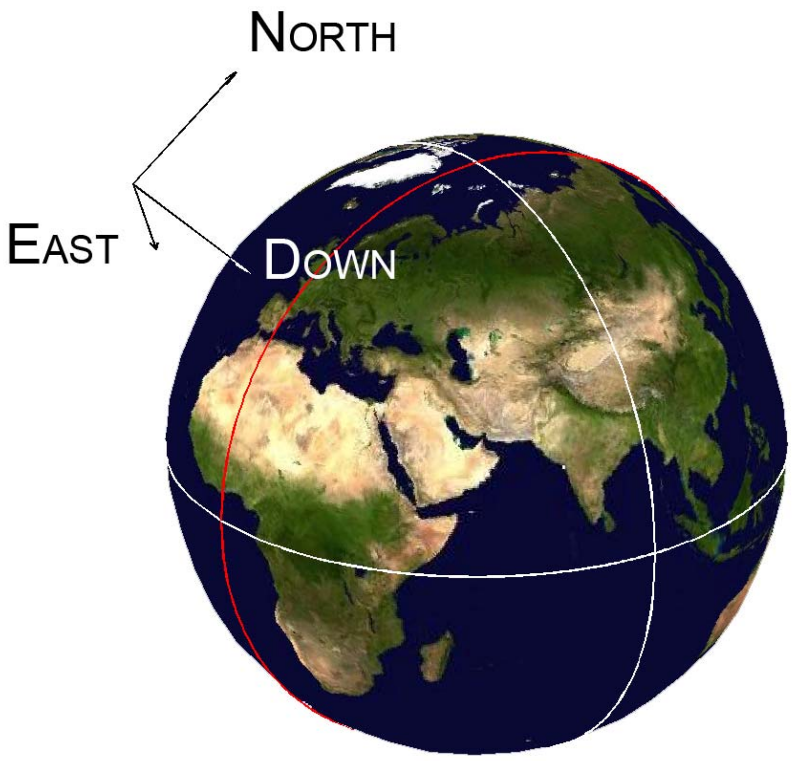

Figure 2.4: Local North East Down Frame

2.1.5 Body-Fixed

The Body-Fixed (Body) frame is aligned geometrically with the spacecraft. The axes are an orthogonal set of unit vectors aligned with three of the unit normals of the sides of CP-2. Figure 2.5 shows a drawing of the Body frame.

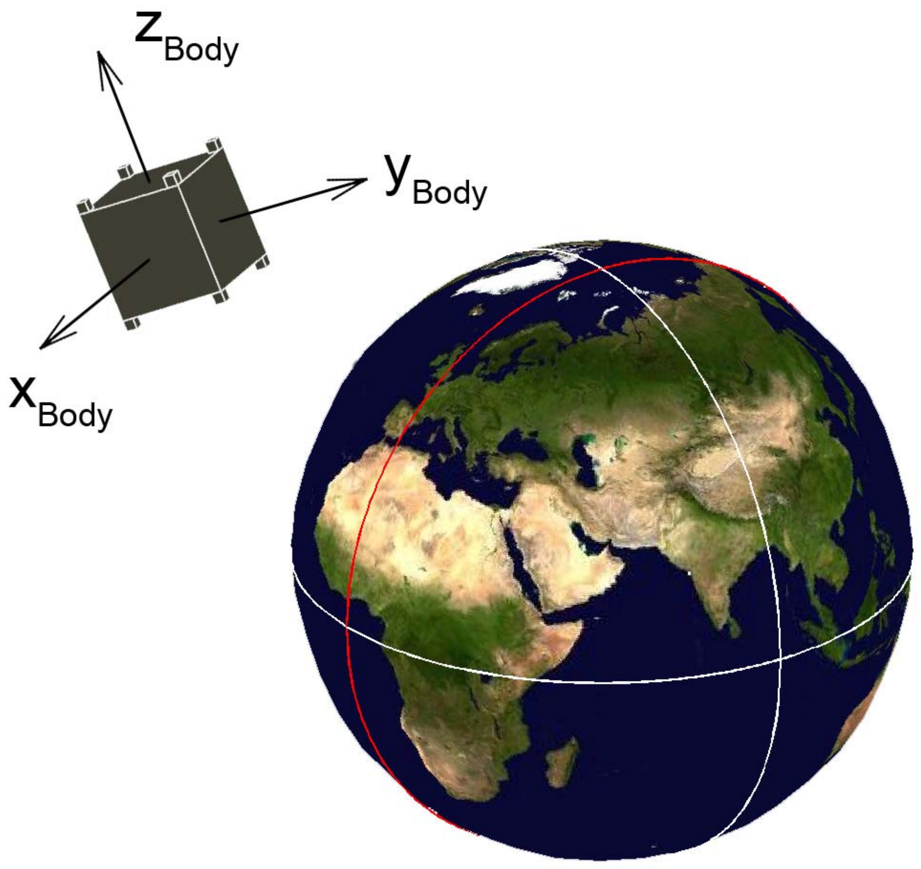

Figure 2.5: Body-Fixed Frame

2.2 Frame Transformations

2.2.1 Neighboring Frames

Neighboring frames are not a technically defined term and are used in this thesis solely as a way to more easily understand the different coordinate frames and their relationships to each other. Neighboring frames are defined as two coordinate frames that one can transform between without first transforming into another defined coordinate frame. Obviously, mathematical technicalities are set aside in this definition as there are an infinite number of ways to transform between two frames, thus, while one way of transforming goes through an intermediary frame that is defined in the previous section, another way would not. Neighboring frames are just the frames that are easiest to transform into without first transforming into another defined frame. For example, converting from the ECI frame to the LNED frame is most easily done by first converting to the ECEF frame; therefore the ECEF frame is a neighbor of the ECI frame as well as the LNED frame as can be seen in Figure 2.6.

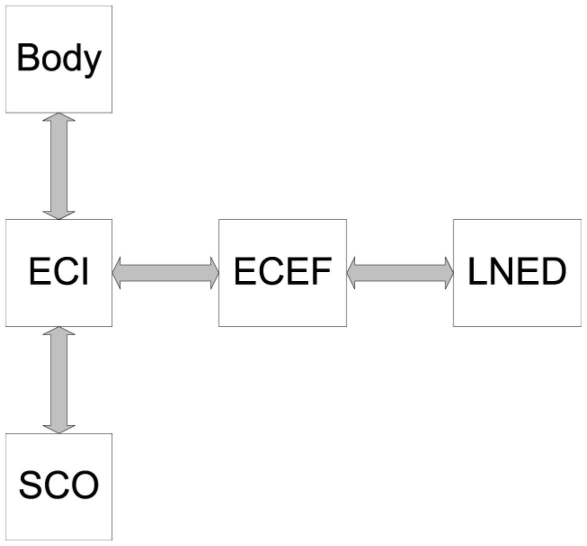

Figure 2.6: Neighboring Frame Definitions

Figure 2.6 shows the neighbors of each of the frames as defined in this thesis, any frames connected by arrows are neighbors. In this thesis, transformation matrices are defined for each arrow in Figure 2.6, and transforming to a non-neighboring frame is done by simply multiplying it by another transformation matrix. This process eliminated the need to derive transformation matrices for all frame transformation combinations, though the end result is the same.

2.2.2 Transformation Matrices

Frame transformations are done through the identification of angles of rotation between frames and the proper application of trigonometric functions to those angles. The end result is arranged into a 3x3 matrix of sines and cosines that can take a vector expressed in one coordinate frame and transform it into another. The following four sections define the angles of rotation to be used for each transformation and derive the corresponding transformation matrix. Though each matrix is derived for transforming a certain direction, the reverse direction's matrix is obtained by simply taking the transpose of the original transformation matrix.

2.2.2.1 ECO to ECI

Figure 2.1 shows the angles of rotation between the ECO and ECI frames. Knowing these angles, the transformation matrix can be shown to be:

$$\vec{x}_{ECI} = \begin{bmatrix} c(\nu+\omega)c(\Omega) - s(\nu+\omega)c(i)s(\Omega) & -s(\nu+\omega)c(\Omega) - c(\nu+\omega)c(i)s(\Omega) & s(i)s(\Omega) \\ c(\nu+\omega)s(\Omega) + s(\nu+\omega)c(i)c(\Omega) & -s(\nu+\omega)s(\Omega) + c(\nu+\omega)c(i)c(\Omega) & -s(i)c(\Omega) \\ s(\nu+\omega)s(i) & c(\nu+\omega)s(i) & c(i) \end{bmatrix} \vec{x}_{ECO}$$

where Ω is the right ascension of the ascending node (RAAN), i is the inclination, ω is the argument of perigee, and ν is the true anomaly. Note that sine and cosine functions have been abbreviated as s( ) and c( ), respectively.

2.2.2.2 ECEF to ECI

The angle between the line of Aries and 0º Longitude relates the ECI frame to the ECEF frame. This angle can be split into the sum of two angles. The first angle θ^{0hUT} is the angle between the vernal equinox and 0º Longitude at midnight UT1. This angle can be found using the equation from the Astronomical Almanac.

$$GMST^{0hUT} = 24110.54841 + 8640184.812866 T_U + 0.093104 T_U^2 - 6.2\times 10^{-6} T_U^3$$

where

$$T_U = \frac{(JD - 2451545.0)}{36525}$$

The second angle is the product of the time since midnight and the angular rotational rate of the earth. Using these angles the transformation can be written as:

$$\vec{x}_{ECI} = \begin{bmatrix} c\left(\theta^{0hUT} + \omega_{\oplus}\Delta t\right) & s\left(\theta^{0hUT} + \omega_{\oplus}\Delta t\right) & 0 \\ -s\left(\theta^{0hUT} + \omega_{\oplus}\Delta t\right) & c\left(\theta^{0hUT} + \omega_{\oplus}\Delta t\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{x}_{ECEF}$$

2.2.2.3 LNED to ECEF

Latitude θ and longitude ϕ are the rotation angles relating these two frames.

$$\vec{x}_{ECEF} = \begin{bmatrix} -s(\theta)c(\phi) & -s(\phi) & -c(\theta)c(\phi) \\ -s(\theta)s(\phi) & c(\phi) & -c(\theta)s(\phi) \\ c(\theta) & 0 & -s(\theta) \end{bmatrix} \vec{x}_{LNED}$$

2.2.2.4 ECI to Body

The Body frame uses quaternions to represent the rotation to the ECI frame. The quaternion is a 4x1 matrix that defines a unit vector in space and the angle to rotate about said unit vector to transform from one frame to the other. The quaternion can be written as:

$$\bar{q} = \begin{bmatrix} \vec{q} \\ q_4 \end{bmatrix} \quad \text{where} \quad \vec{q} = \hat{e}\sin\left(\frac{\theta}{2}\right) \quad \text{and} \quad q_4 = \cos\left(\frac{\theta}{2}\right)$$

In these equations, $\hat{e}$ is the unit vector and θ is the angle. The transformation matrix A, from the ECI frame to the Body frame, is given by the following expression.

$$A(\bar{q}) = \left(q_4^2 - \|\vec{q}\|^2\right) I_{3\times3} + 2\vec{q}\vec{q}^{\,T} - 2q_4\left[\vec{q}\,\times\right]$$

CHAPTER 3: ORBIT PROPAGATOR

3.1 Functional Flow

The orbit propagator is the first step in the attitude estimation routine. The NASA two line elements (TLEs) and a time vector specifying the initial time, sampling time interval, and final time are the orbit propagator's inputs. It reads in the NASA two line elements (TLEs) from a text file and propagates the orbit for the time specified in the time vector. The orbit is propagated in the ECI frame and the position is stored in those coordinates at each sampling time interval. The position is then converted from the ECI frame to ECEF frame and expressed as radius, latitude, and longitude. These values are used with the International Geomagnetic Reference Field (IGRF) model to generate the magnetic inertial reference vector (MIRV). A solar inertial reference vector (SIRV) is also calculated in ECI coordinates by using the time since the last vernal equinox. The orbit propagator passes the MIRV and the SIRV to both the satellite emulator and Kalman filter, as can be seen in Figure 3.1.

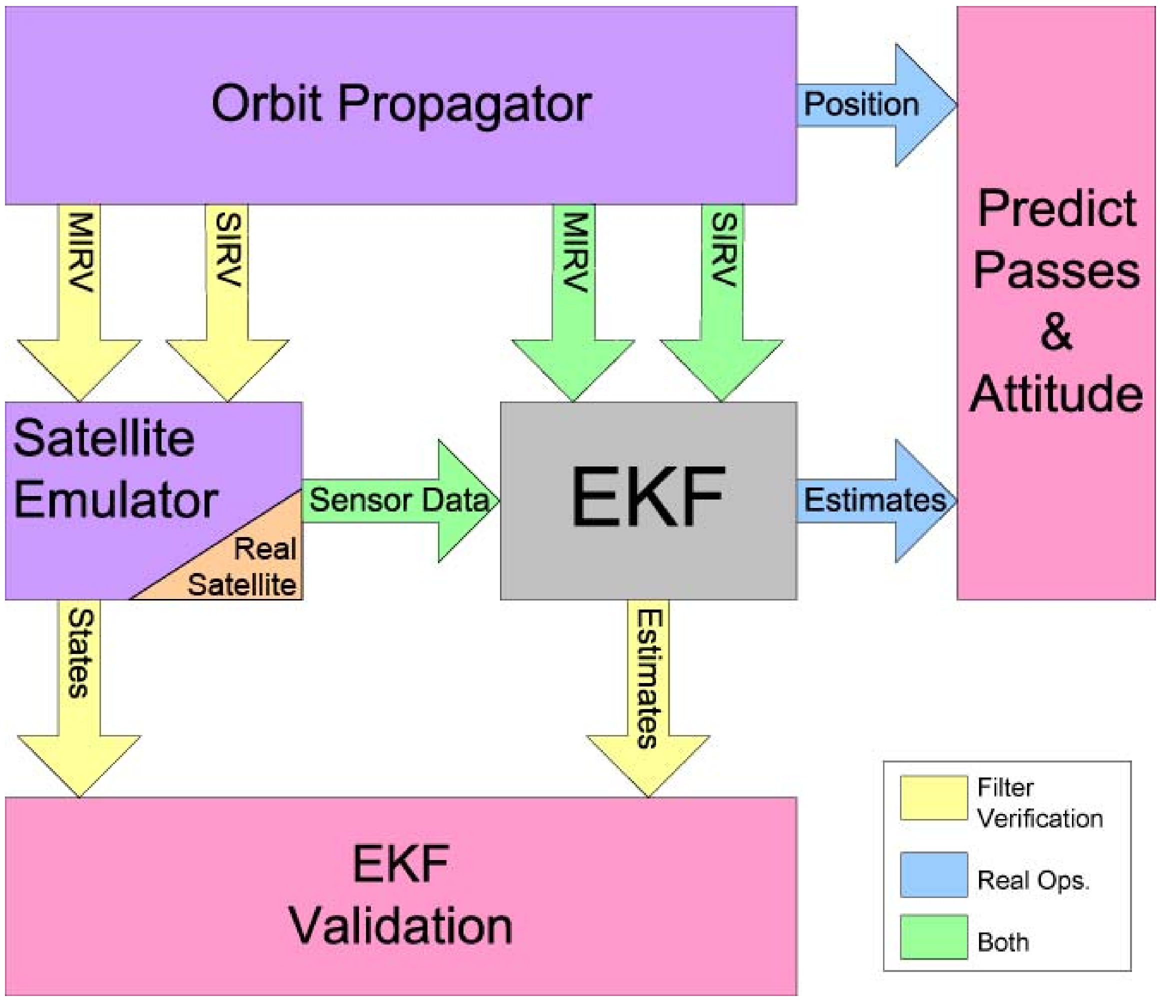

Figure 3.1: Attitude Estimation Routine Flow Chart

3.2 Orbital Mechanics

3.2.1 Equations of Motion

The orbit propagator uses the equations of motion (EOMs) for a body in a gravitational field generated by a point mass and expands these equations to include the J_2 through J_6 terms to account for the effects of the Earth's oblateness.

3.2.2 NASA Two Line Element Conversion

The TLEs give the spacecraft's orbit and its position in that orbit in the SCO frame. In order for the orbit propagator to simulate the orbit, this data is converted into the spacecraft's position and velocity in the ECI frame.

3.3 Performance

The only perturbation used in the orbit propagator's EOMs is the Earth's oblateness. Other perturbations exist that could be modeled to improve the performance of the orbit propagator such as: drag forces, 3rd body effects, and solar pressure. Yet, even with the exclusion of these perturbations the orbit propagator appears to be accurate for around one week, when compared with others.

3.4 Magnetic Inertial Reference Vector

3.4.1 International Geomagnetic Reference Field

The magnetic inertial reference vectors used in this thesis are calculated from the 10th generation Inertial Geomagnetic Reference Field (IGRF-10) model. The IGRF model is good for five years and revised every five years. The current model, IGRF-10, was released in 2005 and is valid until 2010. For 2005 to 2010, the model has a precision of 0.1 nano-Teslas per year.

3.4.2 magfd.m

The IGRF-10 model used in this thesis is stored in several MatLab dat-files and called by the magfd m-file. These files were created by Maurice A. Tivey of the Woods Hole Oceanographic Institution.

3.4.3 Vector Generation

The orbit propagator gives the magfd m-file the required inputs: time, altitude, longitude, and colatitude (90°-latitude). Then the magfd m-file uses the dat-file specified by the inputted time and generates the magnetic field vector and its magnitude at that location in nano-Tesla in the LNED frame, and outputs it in the following format.

$$\begin{bmatrix} b_N & b_E & b_D & \|\bar{b}\| \end{bmatrix}^T$$

The orbit propagator stores the vectors from each time step in a matrix, as shown below.

$$\begin{bmatrix} b_N^1 & b_E^1 & b_D^1 \\ b_N^2 & b_E^2 & b_D^3 \\ b_N^3 & b_E^3 & b_D^3 \\ \vdots & \vdots & \vdots \end{bmatrix}^T$$

3.4.4 Frame Conversion

Once the orbit propagator has the magnetic field vector matrix, it puts it through a frame transformation to take the vectors from the LNED frame to the ECEF frame and then to the ECI frame. Once in the ECI frame, the matrix is stored as the MIRV matrix and passed to the EKF, and, in the case of filter verification, to the satellite emulator.

3.5 Solar Inertial Reference Vector

3.5.1 Vector Definition

The solar inertial reference vector (SIRV) is the unit vector in the direction of the radius vector from the sun to the spacecraft expressed in the ECI frame. Figure 3.2 shows a drawing of the SIRV, though the vectors are not drawn to scale.

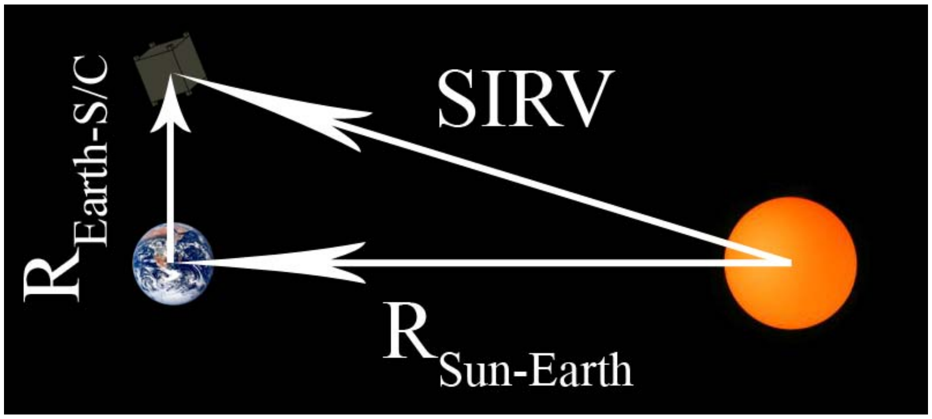

Figure 3.2: The Solar Inertial Reference Vector

Since the distance from the Earth to the spacecraft is many orders of magnitude smaller than the distance from the Sun to the Earth, especially for the satellite in LEO, the SIRV can be approximated as the distance vector from the Sun to the Earth.

3.5.2 Vector Generation

Throughout the year the SIRV changes in the ECI frame. Using the mean motion of the Earth, the obliquity of the ecliptic, and the time since the last vernal equinox, the SIRV at a given time step can be approximated in ECI coordinates.

$$SIRV_k = \begin{bmatrix} \cos[n_\oplus(t_k - t_{ve})] \\ \sin[n_\oplus(t_k - t_{ve})]\cos(\phi_\oplus) \\ -\sin[n_\oplus(t_k - t_{ve})]\sin(\phi_\oplus) \end{bmatrix}$$

The above expression is only an approximation because the angle from the vernal equinox is calculated using time and mean motion, instead of solving the transcendental equation:

$$E - e\sin(E) = n_\oplus \Delta t$$

and then solving for the θ in the following:

$$\tan\left(\frac{E}{2}\right) = \sqrt{\frac{1-e}{1+e}}\tan\left(\frac{\theta}{2}\right)$$

This approximation is acceptable because the eccentricity of the Earth's orbit is near zero, setting the eccentricity to zero in the previous two equations yields:

$$\theta = E = n\Delta t$$

The SIRV, like the MIRV, is calculated at each time step and stored in a matrix.

$$SIRV = \begin{bmatrix} SIRV_1^T \\ SIRV_2^T \\ SIRV_3^T \\ \vdots \end{bmatrix}$$

The SIRV matrix is then passed to the satellite emulator. The SIRV matrix is also passed to the EKF; however, not for use as a measurement reference, but instead to be transformed into the Body frame by the quaternions estimated by the EKF. This solar body estimated vector (SBEV) matrix is then used to estimate the power produced by each solar panel in an attempt to identify if the EKF converged to the wrong state by comparison to the solar body reference vector (SBRV) generated by the satellite emulator. The SIRV could be used as a measurement reference; however the power produced by the solar panel is affected by more than the angle of the incident solar vector. The temperature, Earth's albedo, and powered spacecraft components all affect the solar panel power production. All these processes are difficult to characterize, thus making it difficult to use solar panels as sun sensors.

CHAPTER 4: ATTITUDE SENSORS

4.1 Magnetometers

The magnetometers used on CP-2 are Honeywell HMC1052 two-axis magnetometers. Figure 4.1 shows the magnetometer's location on one of the side boards of CP-2.

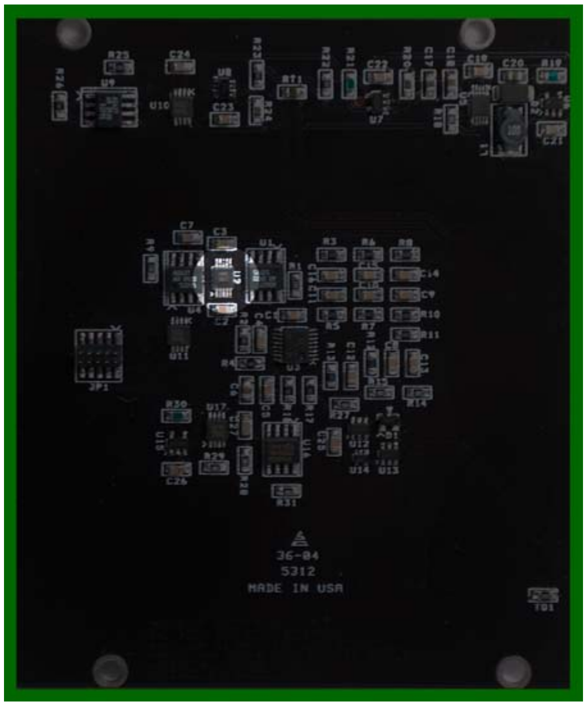

Figure 4.1: HMC1052 on a CP-2 Side Board

Five of the six sides of CP-2 have one magnetometer, resulting in ten magnetometer readings. These ten readings are combined into three measurements along the three axes of the spacecraft; therefore, two axes are double redundant and one axis is triple redundant.

4.1.1 Characterization

Each magnetometer outputs two 8-bit numbers corresponding to the strength of the magnetic field in each direction of the magnetometer's axes. The characterization process was designed to quantify how these 8-bit numbers related to field strength in Gauss, as well as to make the magnetometers behave properly. Proper behavior means that each axis reads near 128 when no field is present and that the strongest field expected does not result in magnetometer saturation.

The characterization process involved taking each side panel and aligning one magnetometer axis with magnetic North and recording the 8-bit number generated by each axis in this field, as well as recording the value in Gauss as read from a FVM-400 vector magnetometer. The process is then repeated for different angles from magnetic North. Figure 4.2 shows the side panel being rotated, the transparency underneath is aligned with magnetic North and has markings for the different angles to which to rotate the side panel.

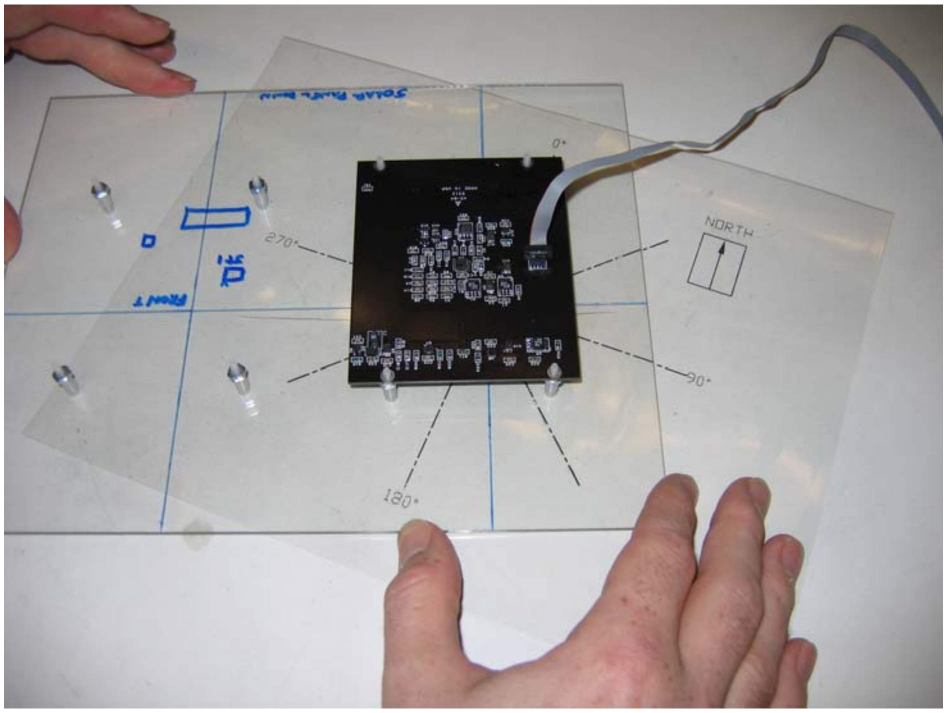

Figure 4.2: Magnetometer Characterization Hardware

The side panel is attached to a piece of acrylic that has two orthogonal lines drawn on it that intersect directly below the magnetometer. These lines make sure that the magnetometer does as little translational motion as possible by ensuring that rotation takes place about its axis. The values generated by the above procedure are recorded in Excel and plotted on the plot show in Figure 4.3.

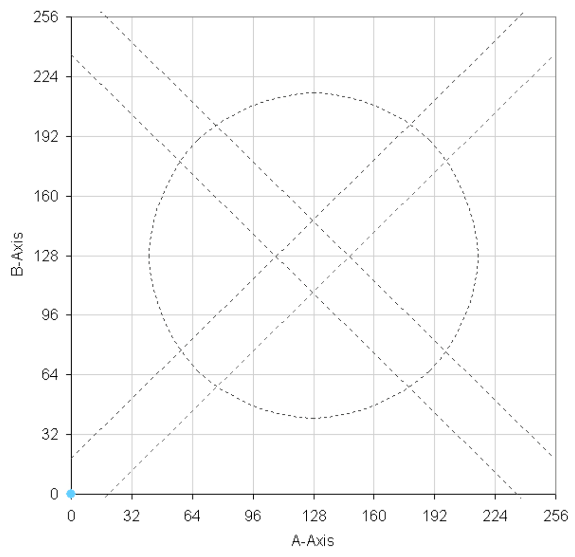

Figure 4.3: Magnetometer Characterization Plot

The axes of the plot are the axes of the magnetometer, ideally the magnetometer would produce points that lie on the circle and centered in the diamond in the middle. The circle is sized such that the difference in magnetic field strength on the ground and in orbit is taken into account; therefore, a value lying on that circle will not result in saturated values during in-orbit operation. Three resistors can be changed on the side board to ensure that the magnetometer behaves as desired. Two of the resistors adjust the gain of the magnetometer's axes and one adjusts the tare value for the axes. The resistors are changed in a certain order while trying to obtain the ideal behavior because the single resistor to change the tare point restricts which magnetometers can be made acceptable. The single tare point resistor means that the tare point can only be changed by an equal amount on both magnetometer axes. Thus the tare point can only be moved on a 45º line in the positive or negative directions of both axes. The two parallel lines going from the lower left to the upper right represent the bounds that the magnetometer's tare point must be within before any resistor changes, if that magnetometer can be made flight acceptable. Once a magnetometer is found that it has a tare point within those bounds, the tare point resistor is changed to move the tare point to within the diamond created by the parallel lines sloping from the upper left to the lower right. After this change has been made the two gain resistors are changed such that the magnetometer outputs points that lie on the circle. Figure 4.4 shows the progression of a typical side board's magnetometer behavior.

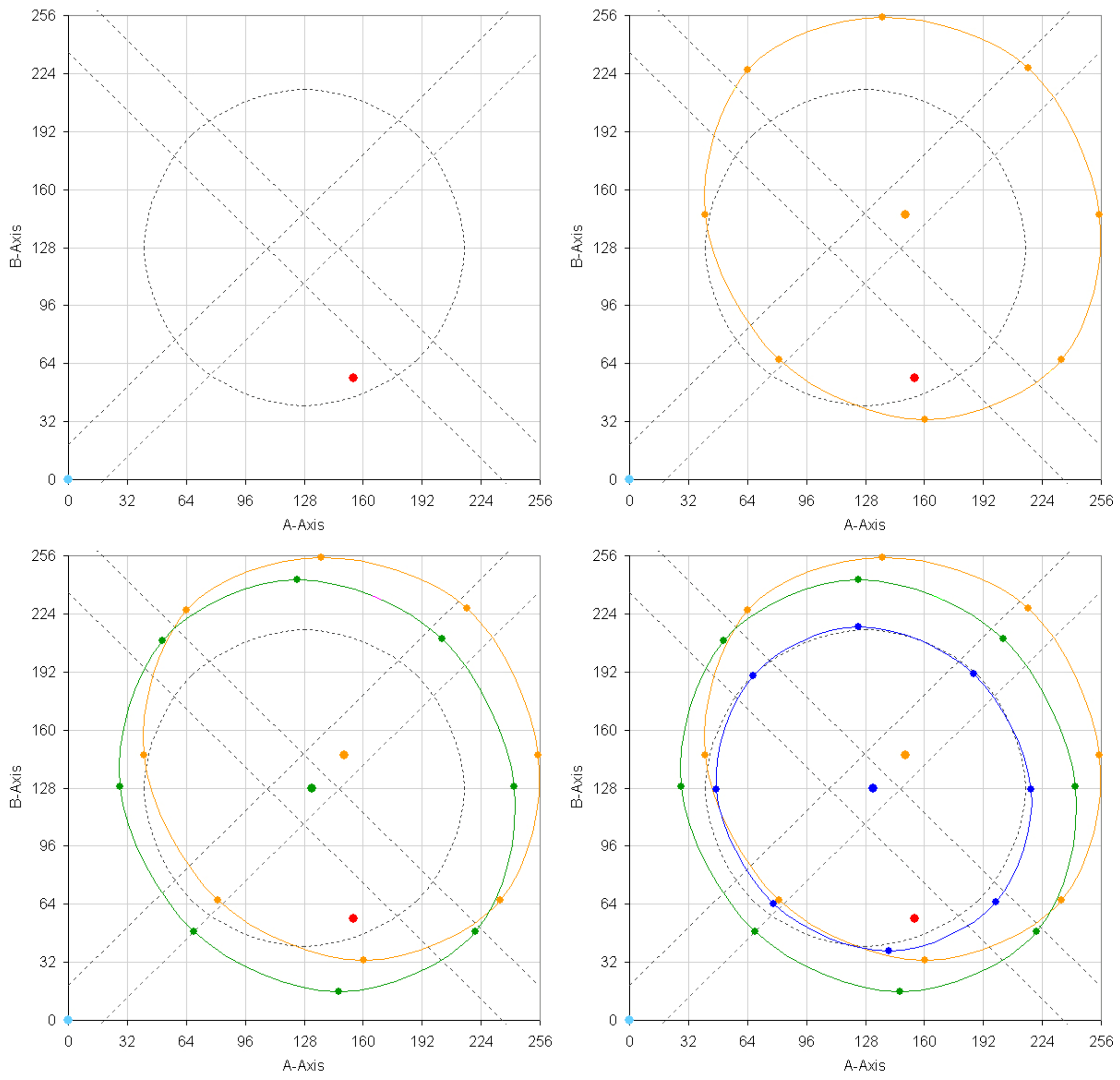

Figure 4.4: Magnetometer Behavior Progression

The upper left plot of Figure 4.4 shows a magnetometer with a tare point that is unacceptable. The magnetometer is replaced with a new magnetometer that produces the orange points in the upper right plot. This magnetometer has a tare point that can be put into the diamond. The tare point resistor is changed to produce this result, and the magnetometer outputs the green points in the lower left plot. Finally the gain resistors are changed to make the diameter of the circle shrink to the ideal circle as can be seen by the blue points in the lower right plot. This entire process is repeated for each side panel and the results are stored for eventual reduction of the raw magnetometer data downloaded from the satellite.

4.1.2 Modeling

The magnetometers are modeled in the satellite emulator as a single three axis magnetometer with an 8-bit resolution with noise corruption. This model assumes that the magnetometer data coming out of the satellite has already been reduced, which basically means the averaging and conversion to Gauss of the magnetometer readings is completed before going to the extended Kalman filter. A simple algorithm is being developed to do the required arithmetic. But for now, this assumption does not affect the performance of the filter, and therefore, is an acceptable one.

4.2 Solar Panels

The solar panels can be used as extremely coarse sun sensors, as the power produced by each panel varies based on the angle of incident light. However, the filter presented in this thesis does not use the solar panels as an attitude sensor. The incident light vector can be difficult to define in the ECI frame due to corruption of the SIRV from the Earth's albedo; however, the true difficulty lays in the variation in the maximum solar panel power production with temperature, battery voltage, powered spacecraft components, etc. During the solar panel characterization process each of those effects was found to be significant and have the potential to greatly corrupt the solar panel power production, leading to a large noise in the calculated angle. With all this in mind, using the solar panels as attitude sensors is still possible, and would be a fun challenge, but the challenge will become significantly less difficult with solar panel flight data and attitude data to compare it to. Also, during eclipse, the spacecraft has no solar panel power production, so it would rely solely on magnetometers. Thus, only using magnetometer's as sensors is not the author's attempt to shy away from a difficult problem, but can be viewed as a necessary portion of the solution to the problem. Ultimately, an extended Kalman filter using both magnetometers and solar panels could produce superior results to a magnetometer-only filter. For now, the solar panels will simply be used to see if a pattern can be found between the filter converging to the correct satellite state and the solar panel power; thus, in this thesis the solar panels power is presented as a possible way to check for filter divergence. To this end, the solar panel power production was characterized with respect to incident light angle.

4.2.1 Characterization

The solar panel power characterization used a lamp at varying angles to the solar panel normal. The resulting relationship between power and angle was not clear cut. While most published papers suggested a sine/cosine relationship, these panels are producing a cubic sine/cosine relationship. Also, this relationship is without a highly variable temperature or active spacecraft processes. Thus for this thesis, the cubic cosine function was assumed for the variation of power production with angle. Even if this particular relationship is a bad assumption, which it undoubtedly is, it still shows that if the filter converges to the incorrect state the solar panels will know, as will be shown in Chapter 6.

4.2.2 Modeling

The five solar panels are each modeled as having a power production that varies with the cosine cubed of the angle between the panel normal, defined positive outwards, and the incident light. While this modeling is done in the satellite emulator as a means to check for filter divergence, it also serves as a rough prediction of spacecraft power production.

CHAPTER 5: SATELLITE EMULATOR

5.1 Functional Flow

The satellite emulator is the next step in the routine after the orbit propagator. The emulator's job is to take its inputs, the MIRV and SIRV matrices, and convert them to its outputs, the magnetic body measurement vector (MBMV) matrix and the solar panel reference power (SPRP) matrix. The MIRV and SIRV are converted from the ECI frame into the body frame at the different time steps through the quaternions which are propagated by the spacecraft attitude equations of motion.

5.2 Equations of Motion

The homogeneous form of Euler's moment equations are used to propagate the angular velocity of the spacecraft from one time vector entry to the next. The equations are shown below in vector form as a single equation.

$$\frac{d\vec{h}}{dt} = -\vec{\omega} \times \vec{h}$$

In the previous equation, h is the specific angular momentum vector and ω is the angular rate vector. The previous equation can be solved for the first derivative of ω with respect to time by using the fact that the specific angular momentum is angular momentum vector multiplied by the inertia matrix. The resulting equation follows, and is used to propagate the angular rates from one time step to the next.

$$\dot{\vec{\omega}} = -J^{-1}\left(\vec{\omega} \times J\vec{\omega}\right)$$

If the spacecraft body frame is assumed to be the principal axis frame the above equation simplifies to the following three equations:

$$\dot{\omega}_1 = \frac{(J_2 - J_3)}{I_1}\omega_2\omega_3$$
$$\dot{\omega}_2 = \frac{(J_3 - J_1)}{I_2}\omega_3\omega_1$$
$$\dot{\omega}_3 = \frac{(J_1 - J_2)}{J_3}\omega_1\omega_2$$

The constants made up of the spacecraft inertias are often rewritten as K_x, K_y, and K_z. The angular rates are then used to propagate the quaternions in the following kinematic equation.

$$\dot{\overline{q}} = \frac{1}{2}\vec{\omega} \otimes \overline{q}$$

The ⊗ symbol stands for quaternion multiplication. The above equation can also be written as:

$$\dot{\overline{q}} = \frac{1}{2}\Omega(\vec{\omega})\overline{q}$$

The Ω matrix is defined as:

$$\Omega(\vec{\omega}) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}$$

This matrix is the same as the quaternion multiplication matrix except that since the rate vector has no fourth component the diagonal terms are set to zero.

5.3 Magnetic Body Measured Vector Creation

The magnetic body measured vector (MBMV) is created from the MIRV matrix. Each row of the MIRV matrix is transposed back to a vector and multiplied by the quaternion transformation matrix to create a magnetic body reference vector (MBRV).

$$MBRV_k = A(\bar{q}_k)MIRV_k$$

The MBRV is then transposed to become a row in the MBRV matrix and stored; it is also corrupted with white noise and given an 8-bit resolution of one Gauss to behave like the magnetometers as characterized in 4.1.1.

$$MBMV_k = round2\left(MBRV_k + \eta_k , \frac{1Gauss}{2^8} \right)$$

The round2 function rounds the first entry to the nearest multiple of the second entry. This corrupted vector is the MBMV; it is calculated for each step in the time vector and stored in the same matrix form as the MIRV as shown below.

$$[MBMV] = \begin{bmatrix} MBMV_1^T \\ MBMV_2^T \\ \vdots \\ MBMV_{t_{final}}^T \end{bmatrix}$$

5.4 Solar Panel Reference Power Production

The solar panel reference power (SPRP) is created from the SIRV matrix. Each row of the SIRV matrix is transposed back to a vector and multiplied by the quaternion transformation matrix to create the solar body reference vector (SBRV) in the same way the MBRV is created. The SBRV is then dotted with the unit normals of each solar panel (defined positive outward), at each time step. Given that the SBRV is a unit vector, the dot product of it and another unit vector produces the cosine of the angle between the two vectors. Thus, the result of this dot product is cubed, in compliance with the characterization done in 4.2.1, and then multiplied by the maximum solar panel power production, approximately one Watt.

$$SPRP_k^i = P_{\max}\left(SBRV_k \circ n_{panel}\right)$$

In this thesis, the hollow circle dot product represents a dot product that is set to zero if positive, also referred to as the solar dot product. The dot product only takes into account angle and not direction, only if the two vectors are pointed in opposite directions is the SBRV actually incident on the panel, as shown in Figure 5.1.

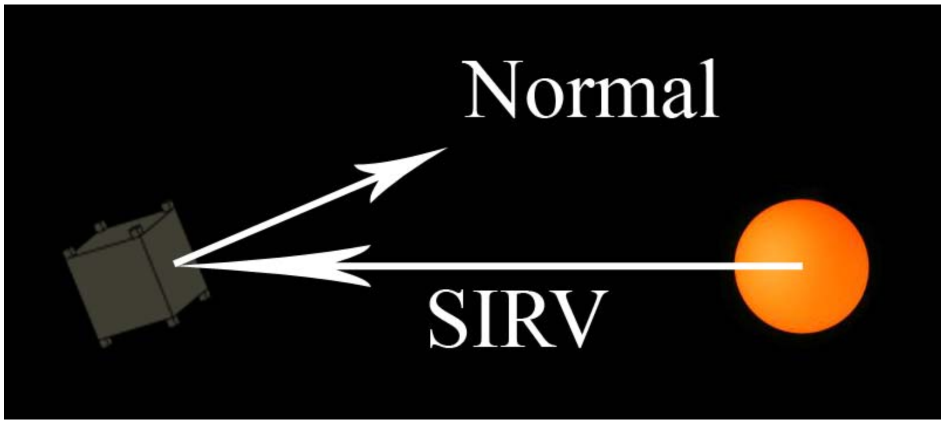

Figure 5.1: Solar Panel Normal and Solar Inertial Reference Vector

The resulting power is the SPRP and is stored at each time step of the time vector. The five SPRPs are combined, each as a column of the single SPRP matrix.

$$\left[SPRP\right] = \left[SPRP^1 \quad \cdots \quad SPRP^5\right]$$

CHAPTER 6: ATTITUDE ESTIMATION

6.1 Attitude Determination Methods

6.1.1 Deterministic Solution

Deterministic solutions are available whenever the spacecraft can observe two or more vectors. Algorithms such as QUEST, FOAM, or TRIAD can be used to determine the Euler angles or quaternions between the ECI and Body frames. When only vector measurement is available, a filtering algorithm must be used to determine the spacecraft attitude.

6.1.2 Batch Filtering

Batch filtering is non-real-time filtering that looks at a set of data and forms estimates by minimizing a cost function. A common example of batch filtering is the minimum model error (MME) approach. This method of filtering can be used with only one vector measurement; however, it can never be made to operate in real-time. Given that one of the eventual goals of this routine is to have it implemented on-board a spacecraft for use with magnetic control, a filter that could be used in real time is essential.

6.1.3 Recursive Filtering

Recursive filtering can work with one vector measurement and in real-time. The Kalman filter is the most prominent form of a recursive filter. Recursive filters use the current measurement in combination with all past measurements to make an estimate; and therefore, do not rely on the knowledge of future measurements. The Kalman filter was chosen as the basis for the recursive filter developed in this thesis since it uses a state-space system model and can be implemented in real-time, though that is not done in this thesis.

6.2 Kalman Filter

6.2.1 Brief History

In 1960, R.E. Kalman used state space models to answer the minimum mean square error (MMSE) filtering problem. The two primary attributes of the Kalman filter solution are vector modeling of the random processes considered and recursive processing of the measurement data. Soon after Kalman published his paper, other papers were published with extensions and variations to the filter presented in Kalman's original paper. One variation described a way to extend the filter to non-linear systems. This variation became known as the extended Kalman filter (EKF) and is very prominent in the aerospace industry today. The following describes the original, linear Kalman filter equations; the EKF is covered in Section 6.3.

6.2.2 Equations

6.2.2.1 System Model Equations

The system to be modeled must be described in state space form and then discretized. The standard state space form is shown below.

$$\begin{aligned}\dot{\vec{x}} &= F\vec{x} + G\vec{u} + \vec{w} \\ \vec{z} &= H\vec{x} + \vec{v}\end{aligned}$$

The last term in each equation is the noise vector. The second term in the dynamic equation can be omitted if there aren't any control or disturbance inputs being modeled. The discretized equations are shown below.

$$\vec{x}_{k+1} = \Phi\vec{x}_k + \vec{w}_k$$
$$\vec{z}_k = H\vec{x}_k + \vec{v}_k$$

The state transition matrix Φ can be found with the following equation.

$$\Phi(T_s) = e^{Ft}\Big|_{T_s} \approx I + FT_s$$

In general, the Φ and H matrices have a subscript k on them; however, most systems describe by the linear state space model have non-time-varying Φ and H matrices. Knowing the dimension of all the vectors and matrices is very helpful in reducing confusion when developing the filter equations. The state vector $\vec{x}$ is an nx1 vector. The state transition matrix Φ is an nxn matrix. The process noise vector $\vec{w}$ is an nx1 vector. The measurement vector $\vec{z}$ is an mx1 vector. The measurement matrix H is an mxn matrix, and the measurement noise vector $\vec{v}$ is an mx1 vector.

6.2.2.2 Filter Equations

The following equations are used in the Kalman filter loop. The first is the Kalman gain matrix equation. The Kalman gain matrix K_k is an nxm matrix defined by the following equation.

$$K_k = P_k^- H^T \left(H \; P_k^- H^T + R \right)^{-1}$$

The R matrix is the mxm measurement noise matrix defined as the variance of the current measurement noise vector. Technically the measurement noise can change in time, if the same sensor is being use to take a measurement at each time step then in general the variance of the noise is constant.

$$R = E\left[v_k v_k^T\right]$$

This definition means little to an engineer; so here is another, in general the measurement noise matrix is a diagonal matrix with each element of the diagonal set to the square of the standard deviation, variance, of the measurement vector entry in the corresponding row. The state covariance matrix P is an nxn matrix corresponding to the expected error in the state vector. The superscript minus means signifies an a priori, or before a measurement, estimate. The next equations are the state update equation and the covariance update equation.

$$\hat{\vec{x}}_k^+ = \hat{\vec{x}}_k^- + K_k\left(\vec{z}_k - H\hat{\vec{x}}_k^-\right)$$
$$P_k^+ = \left(I - K_k H_k\right)P_k^-$$

The only new expressions here are the superscript plus, which signifies an updated, a posteriori, estimate, and the carat which signifies an estimate. The P matrix technically is not an estimate, it is simply a variable quantifying the accuracy of the current state estimate. The first equation in the previous set can be rewritten as the equation below.

$$\hat{\vec{x}}_k^+ = \hat{\vec{x}}_k^- + \Delta\hat{\vec{x}}_k$$
$$\Delta\hat{\vec{x}}_k = K_k\left(\vec{z}_k - H\hat{\vec{x}}_k^-\right)$$

The $\Delta\hat{\vec{x}}_k$ term is called the state error vector and it will become useful in the CP2 extended Kalman filter. The last equations are the state and covariance propagation equations.

$$\hat{\vec{x}}_{k+1}^- = \Phi\hat{\vec{x}}_k^+$$
$$P_{k+1}^- = \Phi P_k^+ \Phi^T + Q_k$$

The process noise matrix Q_k is an nxn matrix describing the expected error in the dynamic equations being used. The following equation shows how to calculate Q_k.

$$Q_k = \int_0^{Ts} \Phi(t) Q \Phi^T(t) dt$$
$$Q = E\left[w_k w_k^T\right]$$

The matrix Q is similar to the R matrix except it corresponds to the variance in the first derivative of the state vector. Here the subscript k shows that Q could vary with time; however, because it is derived from the state transition matrix, Q_k only varies when Φ varies.

6.2.3 Filter Loop

The equations in the previous section are combined in a loop to estimate the current state of the modeled system based on the current and all previous measurements. Figure 6.1 shows a typical loop for the original, linear Kalman filter.

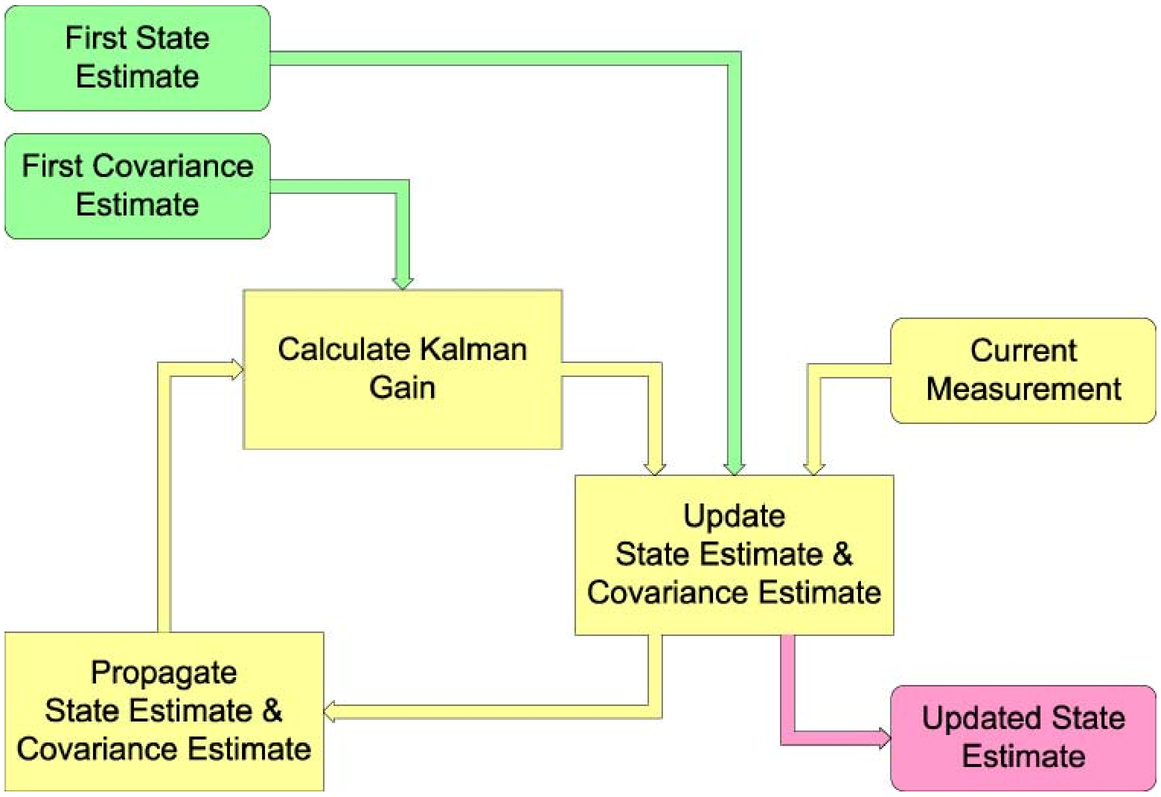
First State Estimate
First Covariance Estimate
Calculate Kalman Gain
Current Measurement
Update State Estimate & Covariance Estimate
Propagate State Estimate & Covariance Estimate
Updated State Estimate

Figure 6.1: Typical Kalman Filter Loop

The filter is initialized with the first state vector estimate $\hat{x}_0^-$ and the first covariance matrix P_0^-, which corresponds to the estimated variance in the initial state vector estimate. Once initialized, the loop is completed as many times as there are measurements; during each loop the updated state vector estimate is saved.

6.2.4 Inadequacy

The standard Kalman filter does not work as an attitude estimator for the system presented in this thesis because it cannot handle non-linear dynamic systems, caused by a tumbling spacecraft, nor can it handle non-linear state-measurement relationships caused by not directly measuring the spacecraft state but instead the magnetic field. For these reasons the EKF was necessary.

6.3 Extended Kalman Filter

As mentioned in Section 6.2.1 the EKF can handle non-linear systems. This section will develop the EKF equations and discuss the corresponding changes made in the loop.

6.3.1 Equations

A non-linear system can be described by the following equations, neglecting the inputs.

$$\begin{aligned}\dot{\vec{x}} &= f(\vec{x},t)+\vec{w}(t)\\ \vec{z} &= h(\vec{x},t)+\vec{v}(t)\end{aligned}$$

If the F and H matrices from the linear Kalman filter are now defined by the following:

$$\begin{aligned}F_k &= \frac{d}{d\vec{x}} f(\vec{x}_k,t)\bigg|_{\hat{\vec{x}}_k^+}\\ H_k &= \frac{d}{d\vec{x}} h(\vec{x}_k,t)\bigg|_{\hat{\vec{x}}_k^-}\end{aligned}$$

then the filter is linearized about the current state vector estimate and can be treated as the linear Kalman filter. Now all the linear Kalman filter equations can be used with only slight modifications to the loop. The linearization about the current state estimate implicitly assumes that the actual state is close to the current state estimate. When this is true the filter works properly; however, a large error could lead to filter divergence. This assumption leads to the need for filter tuning to prevent divergence. EKF tuning will be discussed in detail in Section 6.4.4.

6.3.2 Extended Loop

The EKF loop can be seen in Figure 6.2.

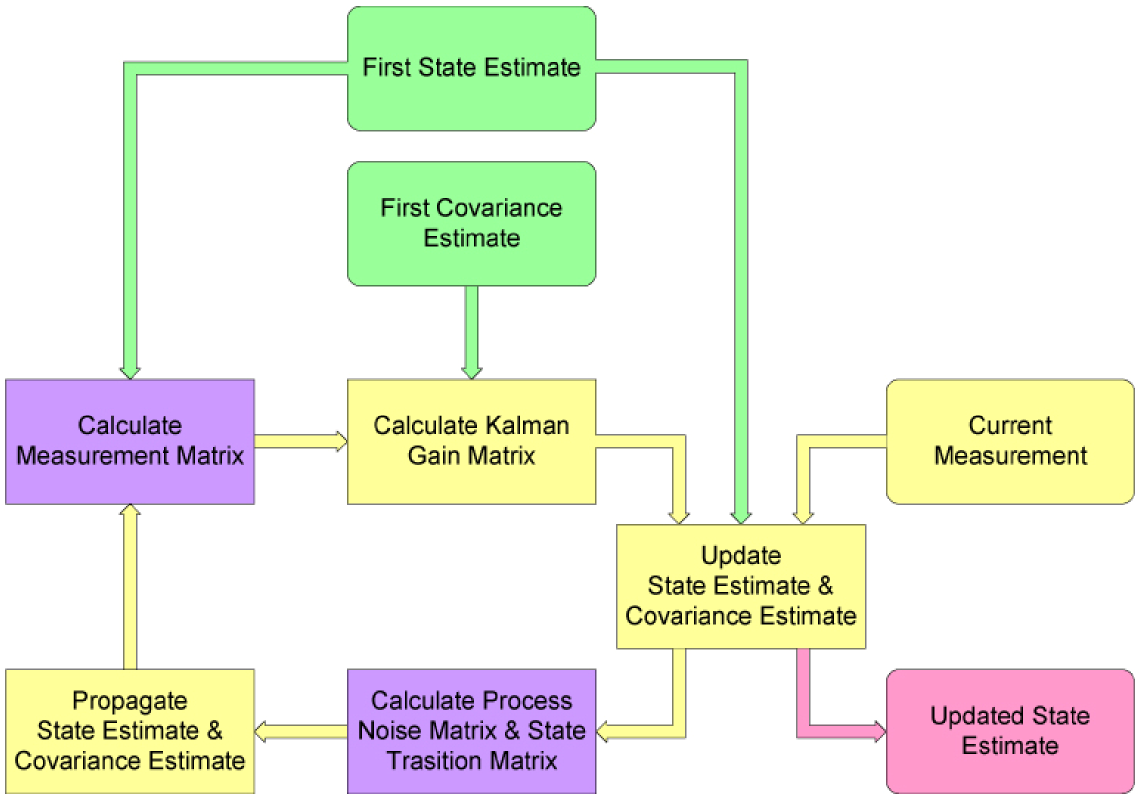
First State Estimate
First Covariance Estimate
Calculate Measurement Matrix
Calculate Kalman Gain Matrix
Current Measurement
Update State Estimate & Covariance Estimate
Propagate State Estimate & Covariance Estimate
Calculate Process Noise Matrix & State Trasition Matrix
Updated State Estimate

Figure 6.2: Typical Extended Kalman Filter Loop

The purple boxes show the changes to the loop. The state transition matrix Φ, measurement matrix H, and process noise matrix Q are no longer constant during the loop. Therefore, they are recalculated each time through the loop. Other than those calculations the loop is essentially the same. The next section derives the specific equations for the PolySat EKF (PEKF), discusses the modifications to the loop, and presents the filter performance.

6.4 PolySat Extended Kalman Filter

6.4.1 Functional Flow

The PolySat extended Kalman filter (PEKF) uses the MBMV from the satellite emulator and the MIRV from the orbit propagator to estimate the spacecraft's attitude at each time step in the time vector. The PEKF also uses the estimated quaternions to generate a magnetic body estimate vector (MBEV) as a check to make sure the updated quaternion estimates transform the MIRV into a vector that is similar to the MBRV. As another check, the PEKF takes the SIRV from the orbit propagator and transforms it into the solar body estimated vector (SBEV) using the quaternion estimates. The SBEV is then used to find the solar panel estimated power (SPEP). The SPEP is then compared to the SPRP, and the solar panel characterization used to find the SPEP is different than how the solar panels actually generate their power, it is assumed that large differences between the SPEP and SPRP will be able to be seen and attributed to filter divergence. In the verification mode the filter outputs plots of the error between actual and estimated quaternion and rates, as well as the error between the MBRV and the MBEV, and the SPRP and the SPEP, all of which will be shown in detail later on in this chapter.

6.4.2 Equation & Matrix Derivations

6.4.2.1 State Vector

The full state vector to be used for the PEKF will be:

$$\vec{x} = \begin{bmatrix} \bar{q} \\ \vec{\omega} \end{bmatrix} \quad \text{where} \quad \bar{q} = \begin{bmatrix} \vec{q} \\ q_4 \end{bmatrix}$$

The system model equations were defined in Section 5.2 and are repeated here for convenience.

$$\dot{\bar{q}} = \frac{1}{2}\Omega(\vec{\omega})\bar{q} \quad \& \quad \begin{aligned} \dot{\omega}_1 &= K_1\omega_2\omega_3 \\ \dot{\omega}_2 &= K_2\omega_3\omega_1 \\ \dot{\omega}_3 &= K_3\omega_1\omega_2 \end{aligned}$$

These equations are used to propagate the state vector from one time step to the next in the PEKF. The propagation is done using a numerical differential equation solver in Matlab, this is acceptable for a ground-based routine; however, this propagation method will have to be changed for this routine to be implemented onboard a satellite.

6.4.2.2 Body-Fixed (Reduced) State Vector

A body-fixed state vector is also used in the PEKF in order to remove the built-in quaternion redundancy that comes from representing a frame transformation with four variables. The removal of this redundancy eliminates the need to deal with quaternion normalization and associated issues with the covariance matrix P in the PEKF. The body-fixed, or reduced, state vector is defined as:

$$\tilde{x} = \begin{bmatrix} \delta\vec{q} \\ \vec{\omega} \end{bmatrix}$$

The $\delta\vec{q}$ vector is the vector component of the error quaternion that transforms the estimated full quaternion to the actual quaternion, shown below.

$$\bar{q} = \delta\bar{q} \otimes \hat{\bar{q}}$$

Assuming the filter is converging, the error quaternion will start to represent a smaller and smaller rotation; and as such, the magnitude of the vector component will be less than one and will be approaching zero. Thus, the fourth component of the error quaternion can be computed as:

$$\delta q_4 = \sqrt{1 - \left\|\delta\vec{q}\right\|^2}$$

However, during initial convergence the magnitude of the vector component could be greater than one, leading to an imaginary fourth quaternion component. During this situation the error quaternion is calculated by the method developed by Humphreys of Utah State University, shown below.

$$\delta\overline{q} = \frac{1}{\sqrt{1+\left\|\delta\tilde{q}\right\|^2}}\begin{bmatrix}\delta\tilde{q}\\1\end{bmatrix}$$

A relationship between the reduced state vector and the full state vector can be derived. First the quaternion product is expressed as matrix multiplication

$$\overline{q} = \delta\overline{q} \otimes \hat{\overline{q}} = \left\lfloor \Xi\left(\hat{\overline{q}}\right) \middle| \hat{\overline{q}} \right\rfloor \delta\overline{q}$$

$$\Xi\left(\hat{\overline{q}}\right) \equiv \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}$$

Knowing that quaternions are normalized, the following can be shown.

$$\delta\tilde{q} = \Xi^T\left(\hat{\overline{q}}\right)\overline{q}$$

Now the relationship can be shown mathematically as:

$$\begin{bmatrix}\delta\tilde{q}\\ \vec{\omega}\end{bmatrix} = \begin{bmatrix}\Xi^T\left(\hat{\overline{q}}\right) & 0_{3\times3}\\ 0_{3\times4} & I_{3\times3}\end{bmatrix}\begin{bmatrix}\overline{q}\\ \vec{\omega}\end{bmatrix}$$

Using properties of quaternion multiplication, the reduced state dynamic equation is found to be:

$$\delta\dot{\overline{q}} = \frac{1}{2}\Omega(\vec{\omega})\delta\overline{q} - \frac{1}{2}\delta\overline{q} \otimes \hat{\overline{\omega}} \qquad \text{where} \qquad \overline{\omega} = \begin{bmatrix}\vec{\omega}\\0\end{bmatrix}$$

6.4.2.3 State Error Vector

The state error vector $\Delta\tilde{x}$ is defined as:

$$\Delta\tilde{x} \equiv \tilde{x} - \hat{\tilde{x}}$$

Using the fact

$$\Xi^T\left(\hat{\bar{q}}\right)\hat{\bar{q}} = 0$$

The state error vector becomes

$$\Delta\tilde{x} = \begin{bmatrix} \delta\bar{q} \\ \Delta\bar{\omega} \end{bmatrix}$$

6.4.2.4 State Transition Matrix

The state transition matrix is approximated as:

$$\Phi_k \approx I + F_k T_s$$

The linear system dynamics matrix F$_k$ can be found be linearizing the dynamics equations in Section 6.4.2.1 about the current state vector estimate. The linearization results in the following matrix.

$$F_k = \begin{bmatrix} -[\hat{\omega}\times] & \frac{1}{2}I_{3\times3} \\ 0_{3\times3} & \Theta(\hat{\omega}) \end{bmatrix} \quad \text{where} \quad \Theta(\hat{\omega}) = \begin{bmatrix} 0 & K_1\omega_3 & K_1\omega_2 \\ K_2\omega_3 & 0 & K_2\omega_1 \\ K_3\omega_2 & K_3\omega_1 & 0 \end{bmatrix}$$

6.4.2.5 Measurement Matrix

The measurement matrix is derived from the following equation.

$$\vec{b}^{Body} = A(\delta\bar{q})A\left(\hat{\bar{q}}\right)MIRV$$

This equation can be rewritten as:

$$\vec{b}_k^{Body} = A(\delta\bar{q}_k)\hat{\vec{b}}_k^{Body} \quad \text{where} \quad \hat{\vec{b}}_k^{Body} = A\left(\hat{\bar{q}}_k\right)MIRV_k$$

The transformation matrix $A(\bar{q})$ can be written as:

$$A(\bar{q}) = \left(q_4^2 - \|\bar{q}\|^2\right) I_{3\times3} + 2\bar{q}\bar{q}^T - 2q_4[\bar{q}\times]$$

The rotation created from $\delta\bar{q}$ is small, and reduces the previous equation to:

$$A(\delta\bar{q}) = I_{3\times3} - 2[\delta\bar{q}\times]$$

Now, using the cross product commutative relationship, $h(\tilde{x})$ can be written as:

$$h(\tilde{x}) = I_{3\times3} + 2\left[\hat{\bar{b}}^{Body}\times\right]\delta\bar{q}$$

The discrete measurement matrix easily follows from the above equation.

$$H_k = \left[2\left[\hat{\bar{b}}_k^{Body}\times\right] \quad 0_{3\times3}\right]$$

6.4.2.6 The Noise Matrices

The measurement noise matrix is defined as:

$$R_k = \sigma_{magnetometer}^2 I_{3\times3}$$

The constant in front of the identity matrix is the square of the standard deviation of the magnetometers.

The process noise matrix is found from the equation in Section 6.2.2.2 using the previously defined state transition matrix. The non-discrete process noise matrix gets factored into an identity matrix and scalar as shown below.

$$[Q] = QI_{6\times6}$$

Physically the scalar Q is the square of the error of the first derivatives of the state vector. This parameter is set during the filter tuning process.

6.4.2.7 Loop Initialization Equations

The initial quaternion estimate $\hat{q}_0^-$ is just taken to be aligned with the ECI frame. The initial rate estimate $\hat{\vec{\omega}}_0^-$ is found be using the first two magnetic field measurements, which will be shown below. The change in the magnetic field measured by the satellite is given by the following equation.

$$\dot{\vec{b}}_{Measured}^{Body} = \dot{\vec{b}}_{IGRF}^{Body} + \vec{\omega} \times \vec{b}_{Measured}^{Body}$$

For short sampling times the inertial change of the magnetic field is negligible and the measured change can be approximated by the difference between two consecutive measurements divided by the sampling time.

$$\frac{\vec{b}_2^{Body} - \vec{b}_1^{Body}}{T_s} = \vec{\omega} \times \vec{b}_1^{Body}$$

A pseudo-inverse cross product is then used to yield the initial rate estimate, as shown below.

$$\hat{\vec{\omega}}_0^- = \frac{\vec{b}_1^{Body} \times \dfrac{\vec{b}_2^{Body} - \vec{b}_1^{Body}}{T_s}}{\left\| \vec{b}_1^{Body} \right\|^2}$$

The initial covariance estimate P_0^- is defined as the 6x6 identity matrix multiplied by a constant.

$$P_0^- = PI_{6\times6}$$

The scalar parameter P is set during the filter tuning process along with Q.

6.4.3 Modified Loop

The modified loop for the PEKF is show in Figure 6.3. The purple boxes are the additions to the standard EKF. This loop is similar to the loop developed by Humphreys of Utah State University.

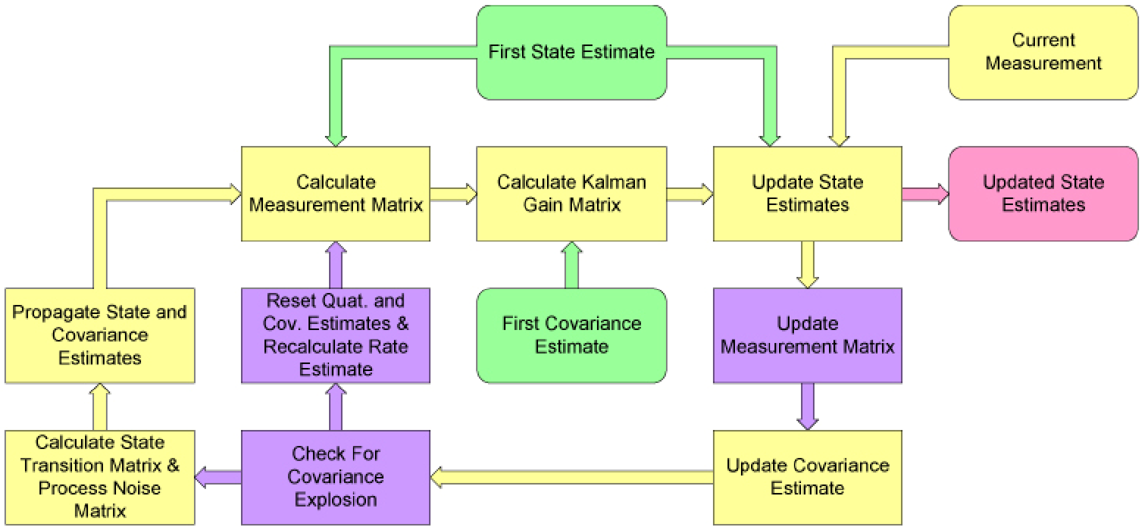
First State Estimate
Current Measurement
Calculate Measurement Matrix
Calculate Kalman Gain Matrix
Update State Estimates
Updated State Estimates
Propagate State and Covariance Estimates
Reset Quat. and Cov. Estimates & Recalculate Rate Estimate
First Covariance Estimate
Update Measurement Matrix
Calculate State Transition Matrix & Process Noise Matrix
Check For Covariance Explosion
Update Covariance Estimate

Figure 6.3: The PolySat Extended Kalman Filter Loop

The purple block that updates the measurement matrix recalculates H_k with the updated state estimate. This process helps keep the covariance matrix from blowing up and causing filter divergence. The possibility of the covariance matrix exploding still exists, which is why the other two purple blocks are implemented in the loop. The first block checks to see if the covariance matrix is growing too large, and if it does, the second block resets the quaternion estimate and covariance error and recalculates the rate estimate based on the current and previous magnetic field estimate. This method of recalculating the rate estimate is a variation from the loop presented by Humphreys, where he simply reset the entire state vector to its initial estimate.

6.4.4 Filter Tuning and Performance

6.4.4.1 Quantifying Performance

Tuning the PEKF consisted of changing the values of Q and P to improve filter performance. The percentage of the trials that the filter converged, and the residual error after convergence, quantified the filter's performance, the higher the convergence the higher the performance. Convergence time was also considered; however, being a ground-based filter, the time to convergence was not as important as convergence itself.

6.4.4.2 Performance Plot Explanation

Each performance plot consists of four separate axes, as can be seen in Figure 6.4. This section will explain how each axis is generated.

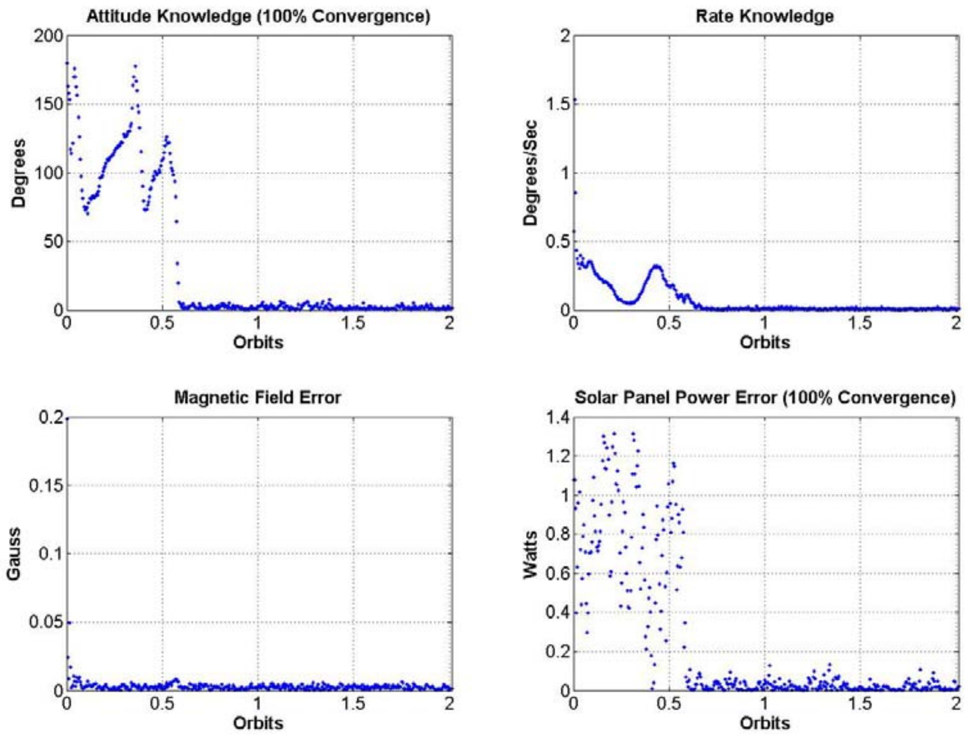

Figure 6.4: Performance Plot

The Attitude Knowledge graph shows a plot of the angle of rotation specified by the error quaternion between the estimated and actual quaternions. The error quaternion is found from the following equation.

$$\bar{q}_4^{err} = \begin{bmatrix} \Xi^T(\bar{q}) \\ \bar{q}^T \end{bmatrix} \hat{\bar{q}}$$

The angle specified by the error quaternion is found from the fourth component.

$$\theta = 2\cos^{-1}\left(q_4^{err}\right)$$

The angle θ is taken as the pointing accuracy and plotted at each time step as the attitude knowledge.

The Rate Knowledge plot shows a graph of the arithmetic difference between the actual and estimated spacecraft rates.

$$\vec{\omega}^{err} = \vec{\omega} - \hat{\vec{\omega}}$$

The Magnetic Field Error graph is a plot of the magnitude of the error between the MBEV and the MBRV at each entry in the time vector.

$$\left\|\vec{b}^{err}\right\| = \left\|MBEV - MBRV\right\|$$

The Solar Panel Power Error graph is a plot of the magnitude of the error between the SPEP and the SPRP, a calculation very similar to the Magnetic Field Error graph. The Solar panel Power Error will go to zero when the PEKF converges, simply because the SPEP and SPRP are generated using the same solar panel characterization; however, when the SPRP is coming from the downloaded data of the satellite, the cosine cubed characterization will no longer be a perfect model. In this case, even when the PEKF converges the Solar Panel Power Error will not go to zero; yet, the error should be smaller than when the filter diverges. For now, this plot is included simply to show that the solar panel power error does recognize filter divergence.

6.4.4.3 Covariance Error Matrix Initialization

The initialization of the covariance error matrix, P plays a role in the performance of the filter. Mostly, P effects the time to convergence and not the residual error after convergence. Theoretically, if P is initially guessed to small, the filter thinks the system is in a state that it's not actually in and may not pay attention to the first measurement; while if P is initially guessed to large the filter has no idea what state the system is in and may over compensate with the first measurement. To tune the filter for P, the process noise gain Q was held constant and P was varied to find the optimum filter performance, which for P is quantified by settling time. The series of performance plots below in Figure 6.5 to Figure 6.11 show the filter's response to different initial P values.

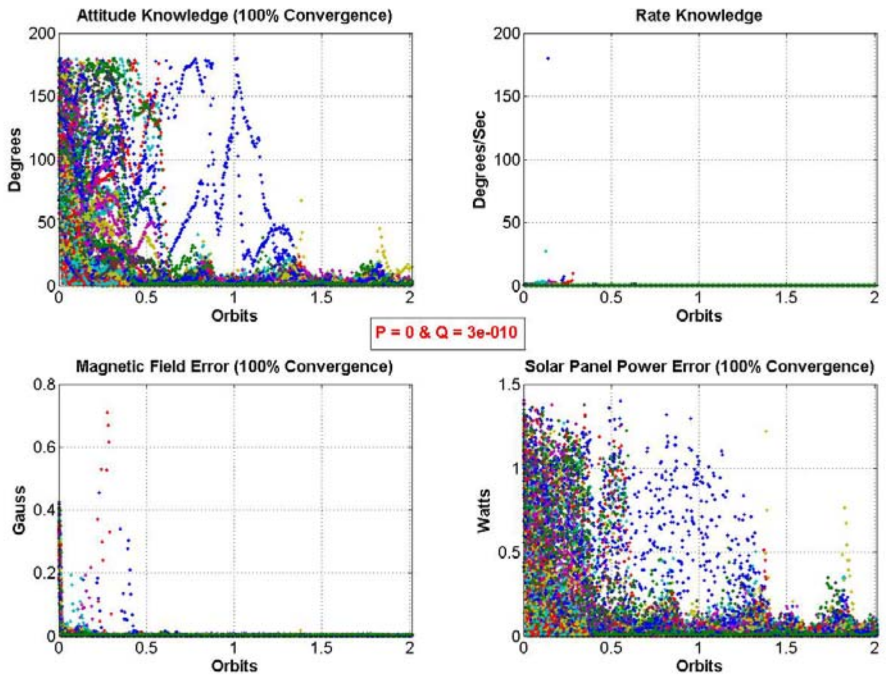

Figure 6.5: Tuning P - P = 0 & Q = 3e-10

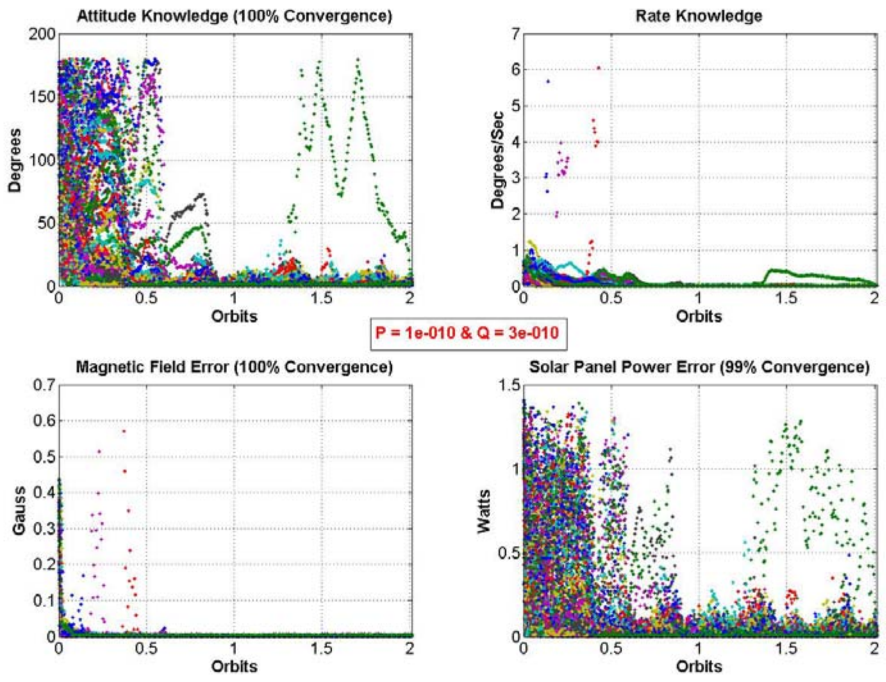

Figure 6.6: Tuning P - P = 1e-10 & Q = 3e-10

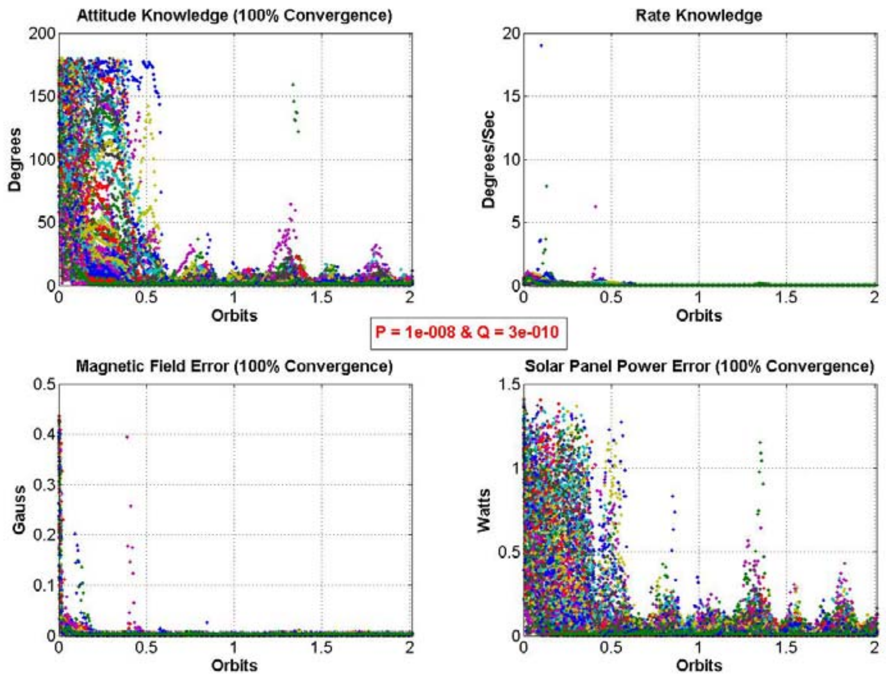

Figure 6.7: Tuning P - P = 1e-8 & Q = 3e-10

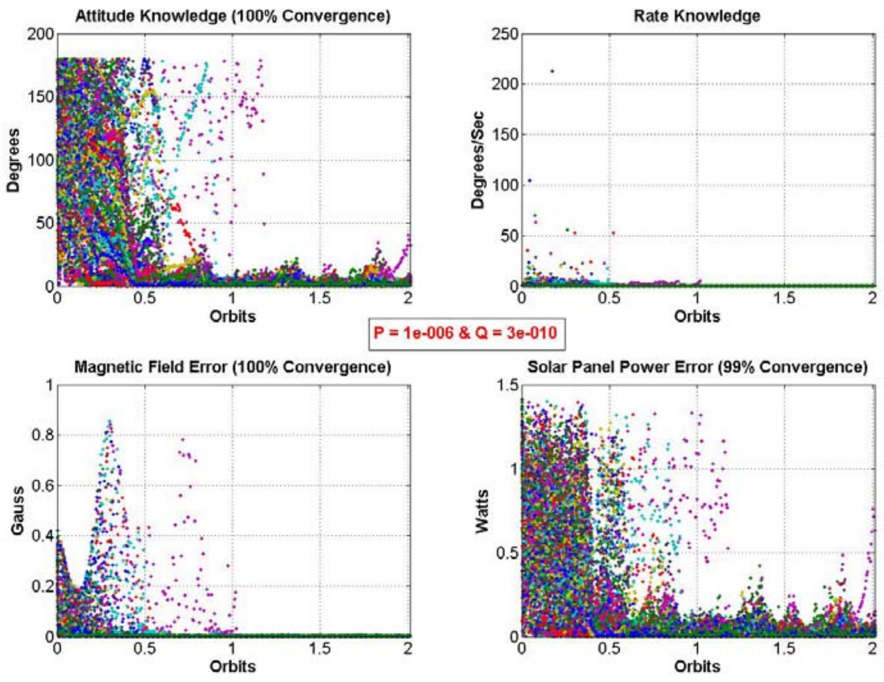

Figure 6.8: Tuning P - P = 1e-6 & Q = 3e-10

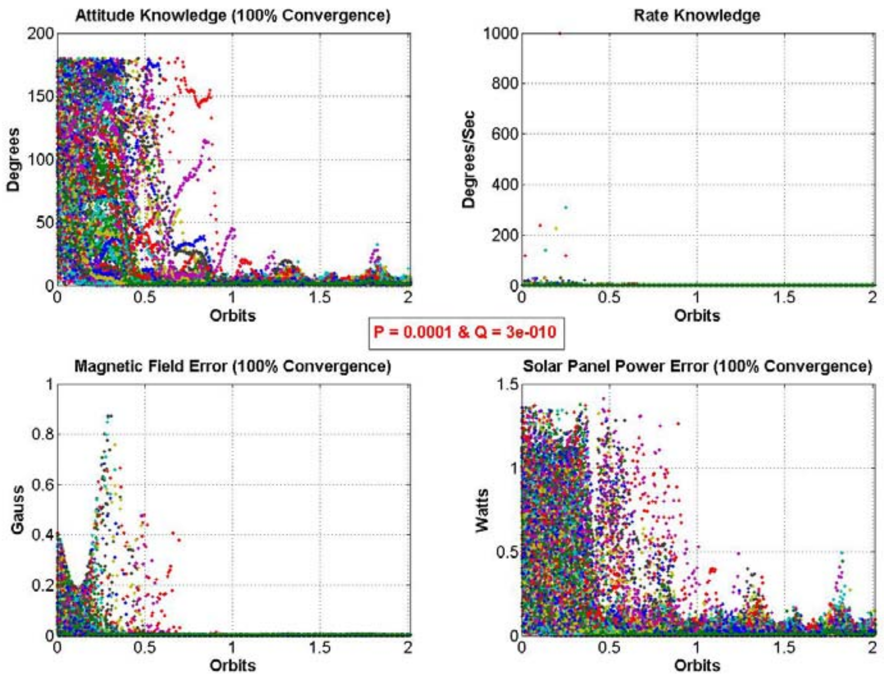

Figure 6.9: Tuning P - P = 1e-4 & Q = 3e-10

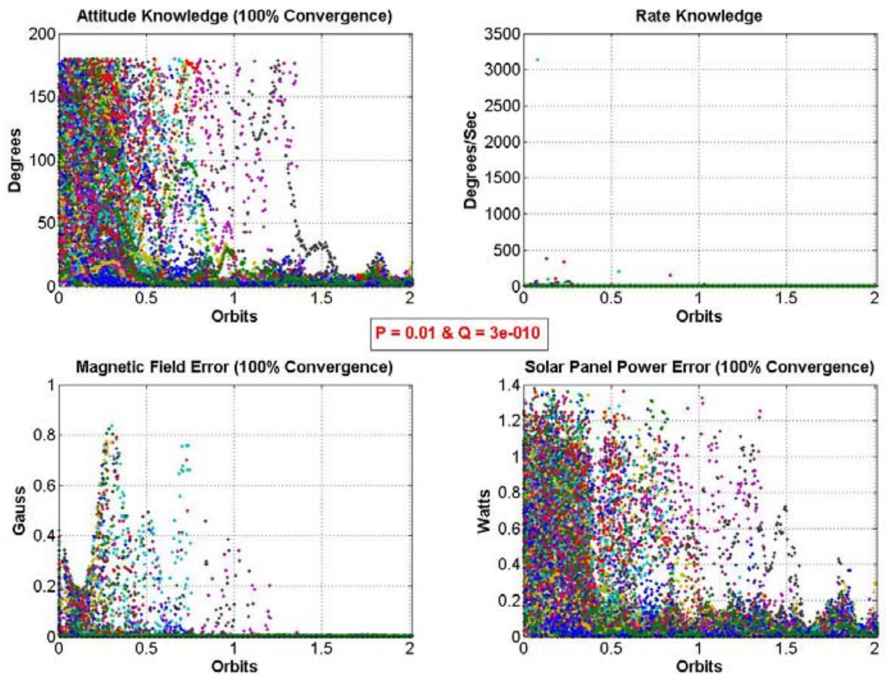

Figure 6.10: Tuning P - P = 1e-2 & Q = 3e-10

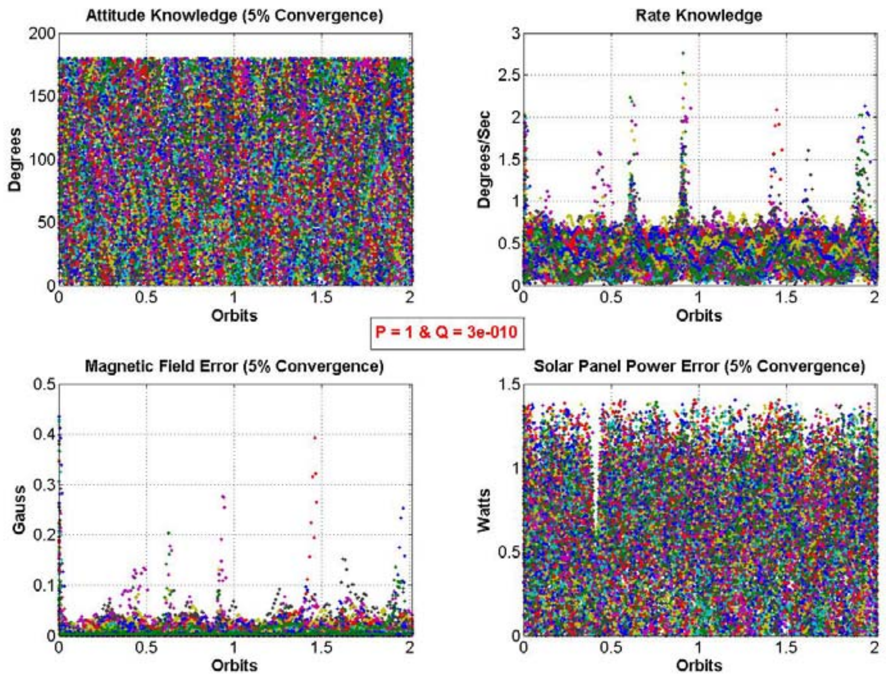

Figure 6.11: Tuning P - P = 1 & Q = 3e-10

P was varied from 0 to 1, with the best performance occurring at 1e-8. The above figures show that the filter is fairly robust about overcoming small initial guess of P; however, if P is guessed to large the filter diverges.

6.4.4.4 Process Noise Matrix Gain

The process noise gain, Q, effects both the filter convergence time and the residual error value. The filter was tuned for Q by holding P at its optimum value of 1e-8 and varying Q until optimum filter performance was found. Figure 6.12 to Figure 6.17 show the results.

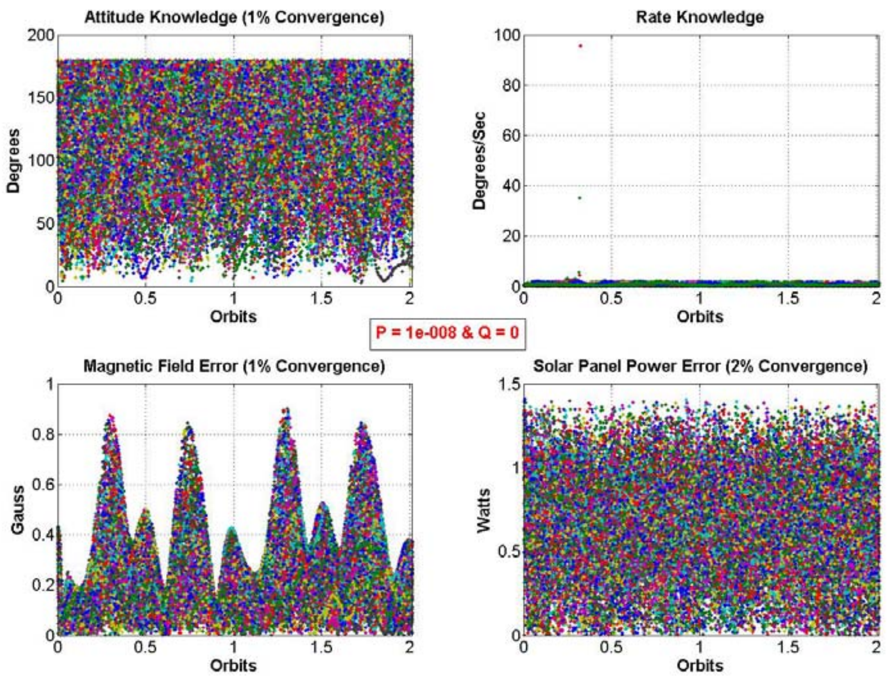

Figure 6.12: Tuning Q - P = 1e-8 & Q = 0

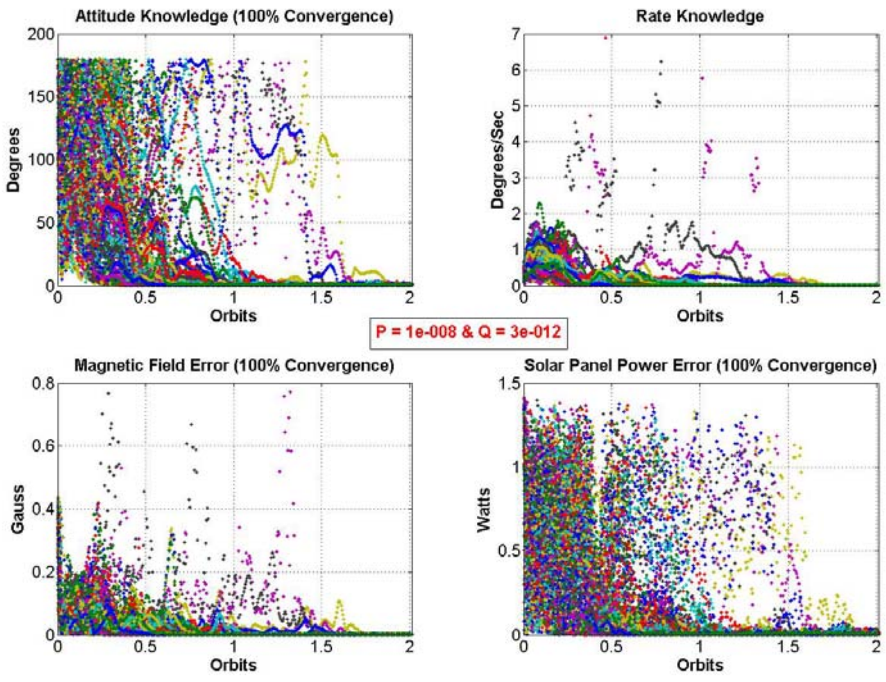

Figure 6.13: Tuning Q - P = 1e-8 & Q =3e-12

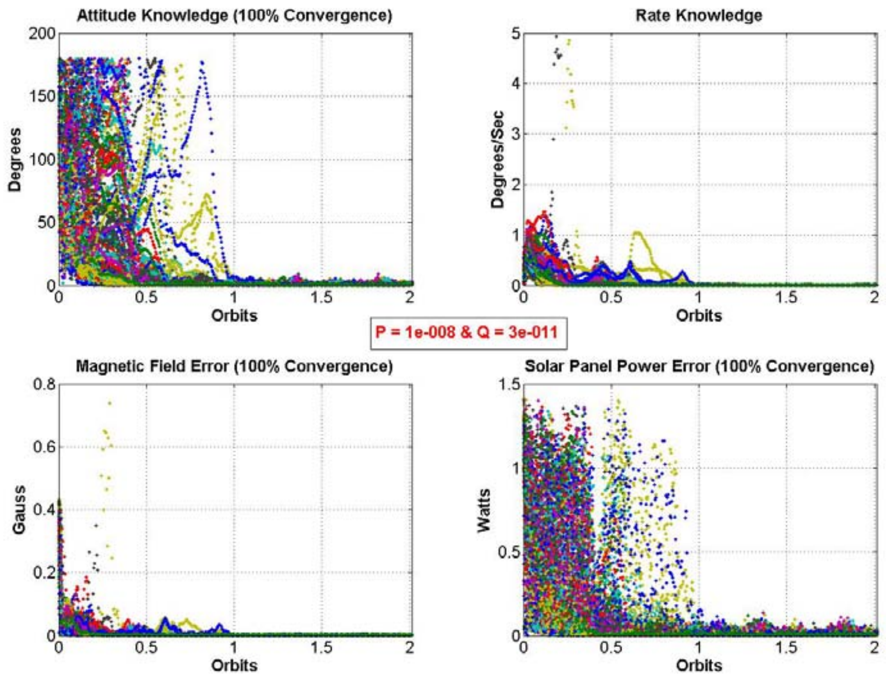

Figure 6.14: Tuning Q - P = 1e-8 & Q = 1e-11

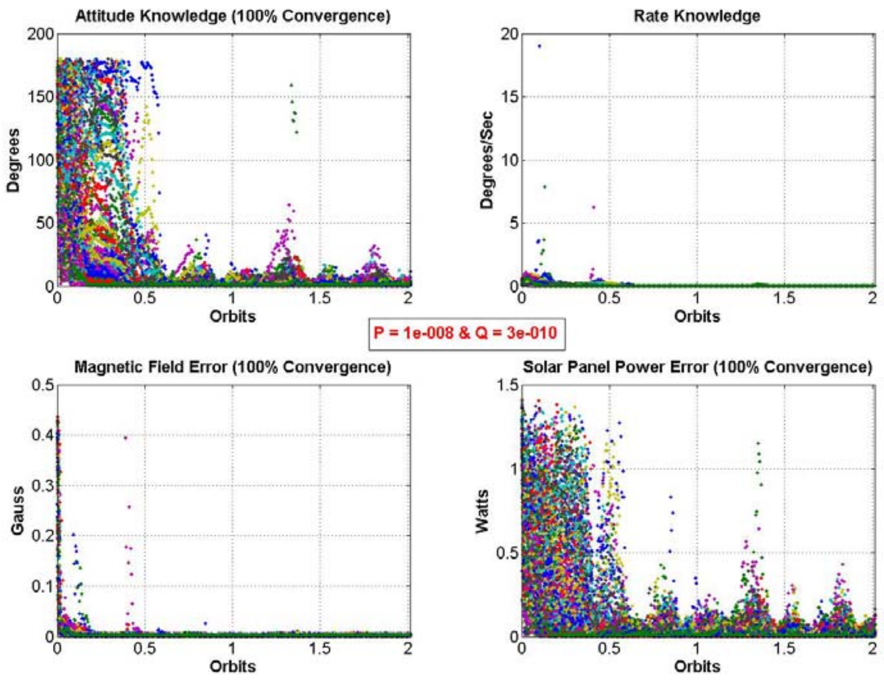

Figure 6.15: Tuning Q - P = 1e-8 & Q = 3e-10

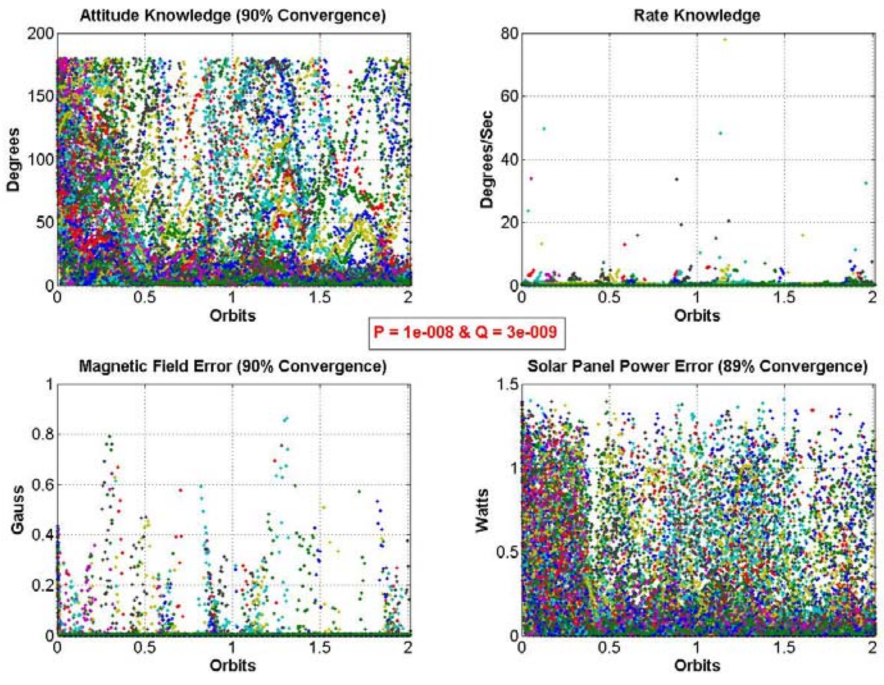

Figure 6.16: Tuning Q - P = 1e-8 & Q = 3e-9

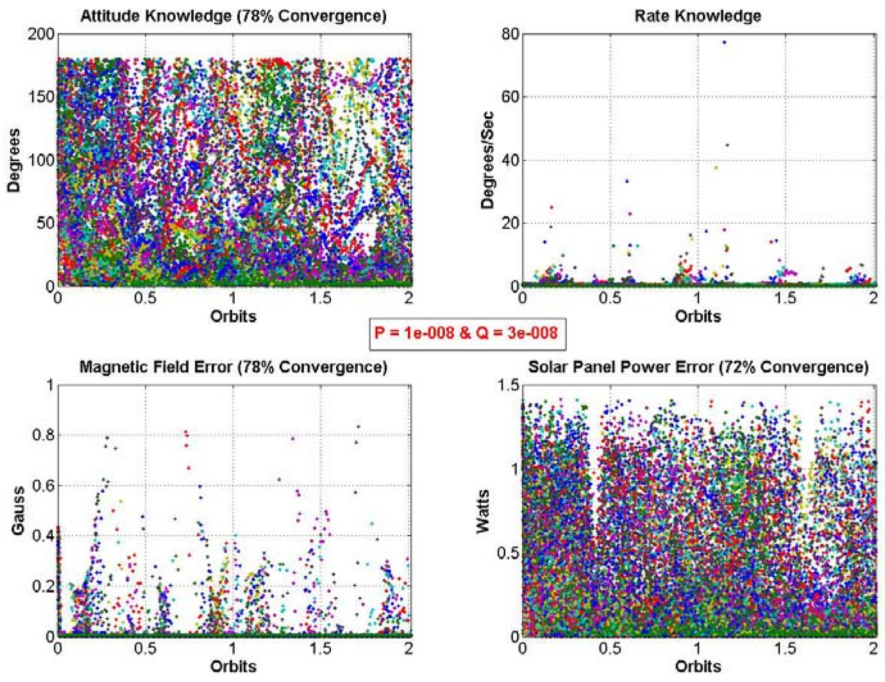

Figure 6.17: Tuning Q - P = 1e-8 & Q = 3e-8

As can be seen from the above figures, when Q is too small the filter cannot converge because it assumes it knows the dynamics of the system perfectly; and when Q is to large the filter diverges because it changes the dynamics of the system too drastically. The best filter performance occurs when Q is set to 3e-11, here the filter converges within 1.5 orbits and has a very low residual error value of less than 10º.

6.4.4.5 Filter Divergence

With proper tuning the filter can achieve 100% convergence; however, the possibility of divergence still exists though it could probably be identified using solar panel data. Figure 6.18 shows the filter performance before the state and covariance resets were introduced into the filter loop and divergence was not rare.

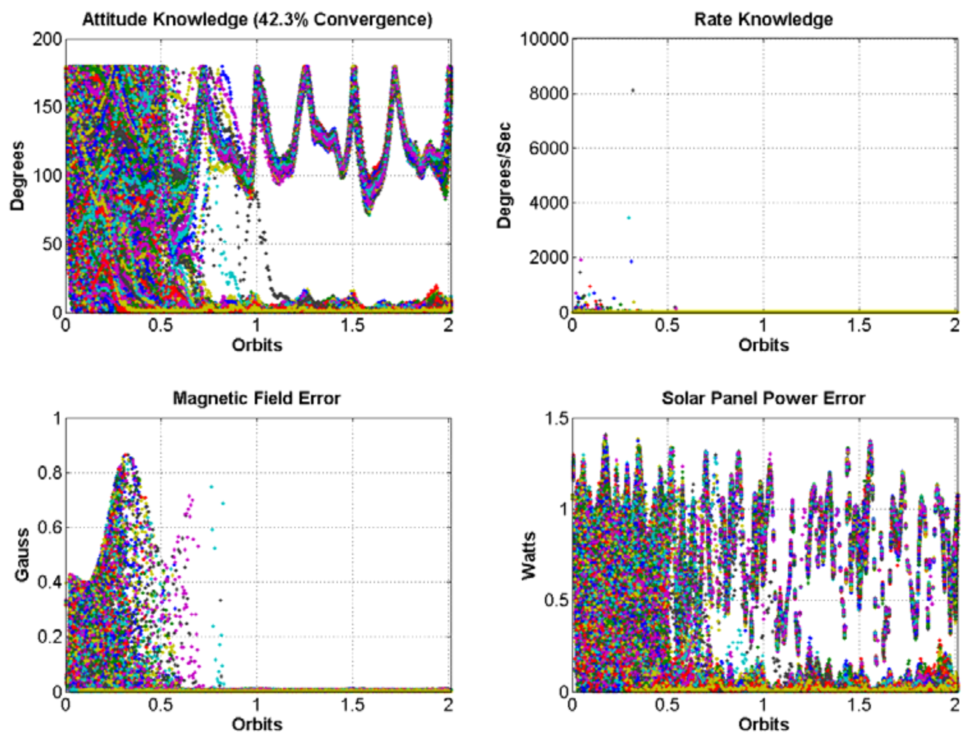

Figure 6.18: Filter Diverges, but Solar Panels Know

When the filter diverges, the Magnetic Error doesn't show the divergence. The solar panel error, however, does catch the divergence, suggesting that understanding the solar panel response to incident light angle is the way to identify filter divergence.

CHAPTER 7: CONCLUSION

7.1 Summary

This thesis presented an attitude estimation routine that can provide spacecraft attitude from measurements of the Earth's magnetic field. An orbit propagator is used to find the magnetic inertial reference vector. A satellite emulator then uses this vector create a magnetic body measured vector. An extended Kalman filter then takes this vector and compares it the expected magnetic inertial reference vector in order to estimate the spacecraft's attitude. Solar panel power production is also estimated by the filter in an effort to provide a means of determining filter divergence.

After tuning the initial covariance error gain and process noise gain the filter can achieve 100% convergence within 1.5 orbits to a residual error value of less than 10º. Even if the filter were to diverge, the simulation results suggest the divergence could be seen by looking at the solar panels data.

7.2 Recommendations for Future Work

7.2.1 Orbit Propagator

Currently, the orbit propagator only takes the Earth's oblateness into account when modeling the orbit. Expanding the propagator to include more perturbations would increase the accuracy of the orbit model when propagated for longer times. Perturbations could include 3rd body effects, drag, and solar pressure. Further tests of this propagator against a commercial one, such as STK or SOAP, would help to validate the code being used.

7.2.2 Satellite Emulator

The satellite emulator does not have any disturbance torques in its EOMs. Adding gravity gradient, drag, magnetic, and solar disturbance torques would increase the accuracy of the dynamic behavior of the satellite while in orbit. Also, having the emulator output raw sensor data and creating another m-file to reduce the raw data for input into the Kalman filter, instead of doing it all in the emulator, would make the verification mode of this routine run more like its operational mode. The satellite emulator computes the spacecraft’s angular rates, which could be corrupted with noise to model rate gyros fairly easily. Expansion to include even more sensor models and outputs would make the attitude estimation routing even more generic and customizable to different spacecraft.

7.2.3 Kalman Filter

Expanding the Kalman filter to include estimates of the spacecraft inertia matrix and the applied disturbance torques would surely increase the accuracy of the estimated states. Also, expanding the measurement matrix to include the solar panels, once some in-orbit data has been collected to better characterize them, would decrease convergence time and nearly eliminate the possibility of filter divergence.

7.2.4 Attitude Estimation Routine

The next steps of this entire routine include: making it less computationally burdensome for implementation onboard a satellite and then combining the routine with a magnetic controller for a fully magnetic ADC system. The ground-based routine can also be combined with the simulations of the magnetic controller presented by Daniel Guerrant in his paper, “Design and Analysis of Fully Magnetic Control for Picosatellite Stabilization.” Also, once flight data have been downloaded this routine could be used to validate spacecraft subsystem models, such as power, thermal, and especially ADC.

LIST OF REFERENCES

Astronomical Almanac for 2005 U.S. G.P.O.

Brown, R.G., and Hwang, P.Y.C., *Introduction to Random Signals and Applied Kalman Filtering 3rd Ed*, John Wiley & Sons, Inc., 1997.

Challa, M., Natanson, G., and Wheeler, C., “Simultaneous Determination of Spacecraft Attitude and Rates Using Only a Magnetometer”, AIAA paper, AIAA-96-3630-CP

Humphreys, T.E., “Attitude Determination for Small Satellites with Modest Pointing Contraints”, Utah State University, Small Sat, 2002.

Krogh, K., “Attitude Determination for the AAU CubeSat”, Aalborg University, June 2002.

Lefferts, E.J., Markley, F.L., and Shuster, M.D., “Kalman Filtering for Spacecraft Attitude Estimation”, *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, 1982, pp. 417-429.

Psiaki, M.L., Martel, F., and Pal, P.K., “Three-Axis Attitude Determination via Kalman Filtering of Magnetometer Data”, *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 3, 1989, pp. 506-514.

Sidi, M.J., *Spacecraft Dynamics & Control: A Practical Engineering Approach*, Cambridge University Press, 2002.

Svartveit, K., “Attitude Determination of the NCUBE Satellite”, NTNU, June 2003.

Wertz, J.R., (ed.), *Spacecraft Attitude Determination and Control*, Kluwer, 2005.

Zarchan, P., Musoff, H., *Fundamental of Kalman Filtering: A Practical Approach*, AIAA, 2000.