Development of an attitude estimator for the PEARL Cubesat

Alexander Nucera

June 11, 2008

Abstract

The object of this thesis is to illustrate the objectives, development methodology and results of xNooch, a software tool designed and developed by this writer, to estimate the obtainable accuracy of the attitude determination of PEARL, a project of the Space Dynamics Laboratory in Logan, Utah, in collaboration with Utah State University. The software was later developed into a more flexible tool, evolving to become a valid pre-mission simulator, and both an on-board and post-mission estimator. PEARL is the prototype for a constellation of satellites, whose mission would be a more accurate sounding of Earth's time-varying magnetosphere, and Earth's electric field. xNooch will help determine if PEARL meets the minimum requirements for its mission, with its current sensing equipment, or if any redesigning must be applied; it will guide the designing of the communication protocol, and more specifically the telemetry rates; it will provide accuracy information on the collected data, after the flight is completed and the data is collected; it might be utilized for guidance during the flight. This student explored several techniques for accomplishing the development of such a tool, and successfully implemented it after careful selection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This document is a master level thesis in the framework of the Spacemaster program, a joint European master in space science and technology. This thesis shall be presented to a committee of faculty members of the Czech University of Technology in Prague, and the LuleåUniversity of Technology. It illustrates the objectives, methodology and results obtained in a project which is being developed at Utah State University, one of the Spacemaster partner universities. In detail, the project is a collaboration of USU and its Space Dynamics Laboratory.

Background The student project, codename: 'xNooch', is being developed for the Space Dynamics Laboratory's PEARL cubesat in collaboration with USU. USU, as a partnership university of the Spacemaster program, has hosted this student and inserted him into this project on the account of the Czech Technical University (CTU) in Prague, Czech Republic, and the Luleå Technical University (LTU) in Luleå, Sweden.

The thesis work started on March 1st, 2008, and lasted until May 28th , 2008, at USU, and while it is now completed it may be upgraded in CTU by the end of June, 2008. This thesis is to be presented during the last week of

June, 2008.

Along with individual work, this student attended weekly meetings with the PEARL development team at SDL, Attitude Determination lectures held by Dr. Fullmer, Space Navigation lectures held by Dr. Geller, and Linear Covariance meetings hosted by Dr. Geller.

Task The student's project above mentioned is complementary to PEARL, an SDL satellite scheduled for launch in 2009. In more detail, it entails the implementation of a software called xNooch, which will be able to estimate the attitude[1] of PEARL and the accuracy of such estimation, based on the measurements received from the sensors on board, and their relative expected accuracy. Such a tool will be helpful in determining, in simulation, if PEARL meets the minimum requirements for its mission, and if any modifications need to be done, in meeting either sufficiency or optimality. As PEARL is a work in progress, recent information dating the last week of May, 2008, says there might be extra funds which would allow the installment of hardware capable of supporting an on board Kalman filter. xNooch is clearly a candidate for that role. Nevertheless, its original purpose is a post-mission processing passage. Attitude estimation accuracy knowledge lends a hand to the science group, in determining the quality of the data and, consequently, of its scientific value.

The original problem that xNooch attempts to solve is essentially divided into two parts, strongly interlaced. The determination of spacecraft attitude is a problem treated widely [1]. For the specifics of PEARL, attitude will be determined with a set of measurements that each carry a certain amount of inaccuracy. The xNooch needs to cleverly collect that information, extract the inaccuracies in it, estimate the resulting attitude; it then needs to deter-

[1]in quaternion form.

mine how inaccurate that value[2] is, and what is the influence of each sensor inaccuracy on that value. The most realistic influence xNooch can have, in the design phase, is on the telemetry protocol, for most of the equipment has been pre-selected and is being currently approved.

Techniques The post-processed attitude determination accuracy estimation, which is the original task of xNooch and the focus of attention of this project, is achievable in a number of ways. This student's attention is focused on a limited amount of possibilities, given the constraints of this project. Techniques that are known to accomplish this task, and will be investigated, are: the Monte Carlo simulation; the linear covariance analysis; the application of a batch estimator (such as the Kalman smoothing filter).

Of the three techniques listed above, the Monte Carlo simulation [2] is surely the simplest one. It consists of a brute force approach, for it simulates the mission (with all of its modeled dynamics) a large amount of times, varying the input variables (the errors on our sensors, and how they would propagate). The results are therefore quite reliable, although this solution is certainly inelegant and quite costly in terms of resources. It is essentially useful in an a-priori simulation.

The strongest attention will be put on the attitude Kalman filter [3]. A topic treated widely, the Kalman filter is a cleverly designed low-pass filter which offers an optimal estimate, given the available data. In this paper an extended form of it will be derived, equipping it to treat the highly non-linear model we are confronted with. Further, the Kalman smoothing algorithm will be investigated. It shall utilize the principles of a Kalman filter applied to an entire batch of information, rather than on single incoming real time measurements. The ready possession of the entire batch of measurements provides better information on the true value of the stimuli on our sensors at

[2]which will take the form of a quaternion.

all considered time instants, or steps, diminishing the effect of measurement noise.

The linear covariance approach, as devised by Geller [4], allows to obtain Monte Carlo like results in a fraction of the time, by developing a linearized model of the system being analyzed, and running a single simulation in which the error covariances are determined by directly propagating, updating and correcting an augmented state covariance matrix.

Chapter 2

The Framework

2.1 PEARL and its mission

As mentioned, xNooch is a tool complementary to the PEARL satellite.

The PEARL is just one of an endless list of projects that fall under the CubeSat category. Such a trend in modern satellite design is easily traceable to the relatively small costs involved with this class of satellites, designed to be very small and more affordable to organizations such as universities, lacking the extremely large funds required for larger, more complex spacecraft. In particular, PEARL is a project of the Space Dynamics Laboratory based in Logan, Utah, with the collaboration of Utah State University.

A space mission is indeed a mighty task. Any reader to whom this document is directed might have a ballpark idea of what figures are required when taking on such an endeavour[1]. Even where funds are prosperous, and research support is generous, financial obstacles come in the way and obscure regions of the planning process. So it comes that PEARL is currently under design, and may or may not be equipped with hardware that may greatly increase its maneuvering capability, through sensing, processing, or actuation.

[1] *mica pizza e fichi*, my former roman colleagues would say.

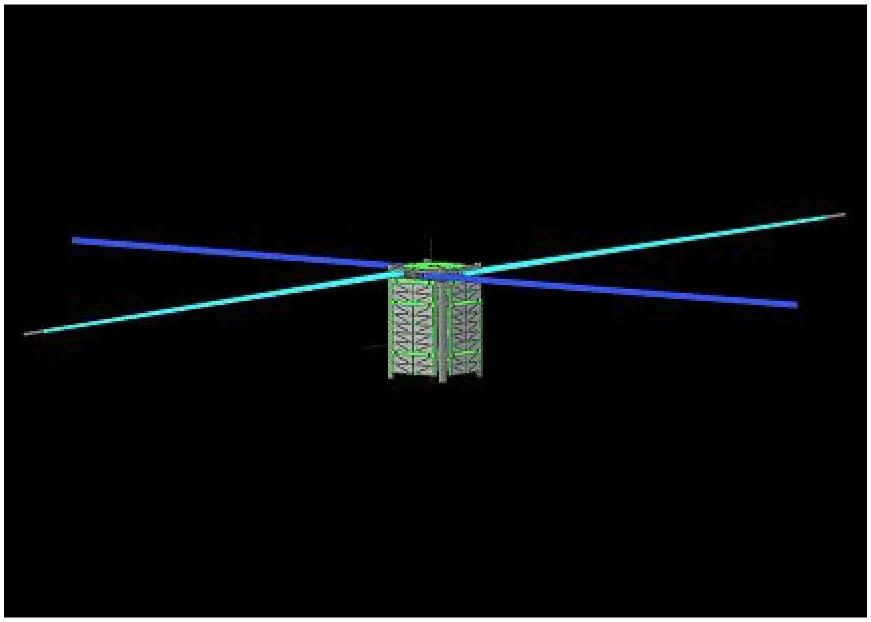

Figure 2.1: A drawing of the PEARL, with its booms deployed

The financial aspect, for the duration of this project, has been brought up more than once. While it did not influence this writer's work, it remains one of the issues being dealt with (and may bring up setbacks).

PEARL is intended to be a prototype for a much larger mission, involving approximately 100 satellites of identical design. The mission of this constellation of satellites would be the sounding of the magnetosphere, with the objective of creating a more detailed model of Earth's electric field, rich in irregularities which are nowadays often disconsidered[2]. The coordination of such a large satellite fleet would allow the synchronous measurement of several different points in space, which would be key in determining the *time-varying* characteristic of this planet's electric field.

While the large fleet mission is still in proposal phase, with SMEX as its temporary name, PEARL is currently being designed, and the launch is

[2]which really means they are modeled as error, and dealt with that way.

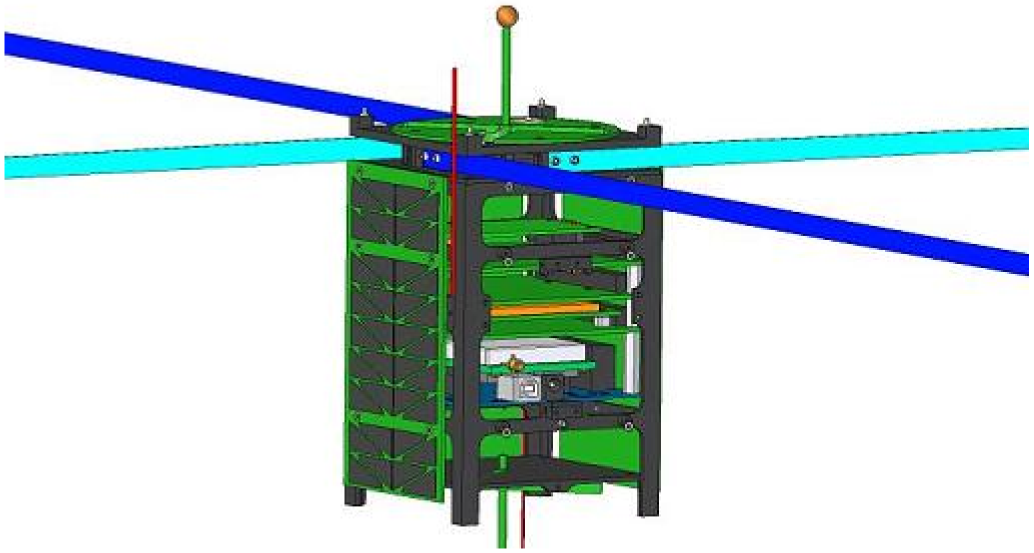

Figure 2.2: A zoom into PEARL after removing one of the solar panel

hypothesized to be sometime in 2009. Thus the need for xNooch: it would be an important tool in determining the effective capability of the satellite to acquire data that is sufficiently reliable, in time for modifications in its design and, more importantly, telemetry protocols. xNooch's results are key in the approval for SMEX.

The information contained in this chapter was collected by this student while attending weekly meetings at SDL with the rest of the development team of PEARL, lead by Dr. Chad Fish. The attitude determination sub-team comprised, apart from myself, Dr. Rees Fullmer, Bryan Bingham, and Anders Forslund, fellow Spacemaster student and dear friend[3].

[3]and quite the guitar player and comedian as well.

2.2 Sensors on board PEARL and the error model

PEARL, because of its limited resources, must and will be extremely lightweight, and for that reason it will employ the highly passive spin stabilization technique [5]. It will be also equipped with magnetic torquer coils, for attitude control. For attitude determination, which is the main concern of this writer, PEARL is currently expected to be equipped with 3 gyro rate sensors; two quad-cell sun sensors; a global positioning system (GPS) receiver; two infrared earth horizon crossing sensors; three magnetometers, which will be also employed in the mission's scientific measurements; various temperature sensors[4], attached onto most of the other sensors listed above.

All of these sensors provide measurements that inevitably carry a certain unknown error. In the earlier stages of software development, simple random computer generated white noise might be employed for testing of the software. It will be developed following the incremental paradigm, therefore solving the essential problem of attitude determination and consequently incrementing the complexity in the problem. This choice is primarily due to the extremely short time assigned to a project like this. It felt wise and recommendable to develop working prototypes step by step, to make sure a minimum result would be met in good time.

Other members of the team this writer is inserted in are developing accurate noise models, which are created with in mind the inner dynamics of the sensors immersed in the expected environment of the mission. A preliminary analysis of the expected nature and sources of these errors has been done in collaboration with, but mainly by, Dr. Rees Fullmer and Anders Forslund. In particular, Anders' thesis, currently in progress, documents that part of

[4]potentially useful for bias drift estimation, but disconsidered in our treatment due to lack of specification.

the project quite well. That analysis did not involve much of this writer's participation.

To determine how much these errors influence the estimation of the spacecraft's attitude is precisely the objective of xNooch.

2.3 The satellite model

Along with the error model, a satellite model is being prepared by this writer's team members. That model is actually broken up in three complementary parts:

The physics model describes information about the universe and its laws, as far as our simulation is concerned; it includes the truth values of time, location and orientation of the satellite, in relation to earth and the sun;

The sensors and actuators model, which corroborate the simulation with the inaccuracies brought in by the sensing of the devices on board, through the use of the error model;

The attitude determination and control (ADC) model, which describes the software being used on board the spacecraft computer.

Any deeper analysis of this model is not necessary for this report.

2.4 The development environment

This model is being developed in MatLab's Simulink tool/interface. This implementation choice is supported by the scalability and modularity of that programming environment, which helps managing a team project like this one. xNooch represents a portion of the simulation, and a tool, which in the original conception works entirely off-line. That means that it was intended to run after the satellite's mission simulation. It originally appeared too expensive in terms of processing resource to have a Kalman filter run on the satellite, in real time. xNooch's purpose, as a reminder, is not pure attitude determination. For that reason, xNooch is not required to be implemented in Simulink, and is in fact developed with pure MatLab code. The reason behind this choice is the complexity of the algorithms xNooch will be implementing, and the support offered by MatLab code will ease the programming burden.

The possibility of it being installed on the satellite is still very viable. xNooch does not require large interfacing with the model that the ADC team has developed, and therefore the choice of having it implemented in Matlab remains adequate. The required translation to make it run on board PEARL is out of this project's scope.

Chapter 3

Preliminary Approach

3.1 xNooch's objective

xNooch is a software tool developed onto the previously mentioned framework. Its objective is to estimate the attitude of PEARL during the entire course of its mission, after it being completely completed[1]. Thus, it was designed as a post-processing tool. The attitude determination of the entire flight time will be obtained by processing the data acquired by the sensing instruments. Along with that information, xNooch is required to estimate the relative error carried by that attitude estimate, at each time step, which is a function of the error in the measurement (unknown), and the accuracy of the sensors.

3.2 Techniques

As mentioned in the introduction, a few different ways will be examined for the implementation of xNooch.

[1]or simulated.

3.2.1 Monte Carlo analysis

The Monte Carlo approach is by definition a brute force approach. In general it is appropriate for problems where it is impossible or impractical to find results using a deterministic algorithm. The method consists of defining a set of variable inputs, for a certain problem. Subsequently, once a model is prepared and data is collected, it is executed a large number of times, varying the input errors. After hundreds or thousands of simulations, the results are taken and analyzed, according to the problem.

A simpler problem than the PEARL's mission will be illustrated. This example is taken from [6]. Let's assume a spacecraft equipped with an inertial navigation system that includes accelerometers and gyros. The equations that describe the motion or trajectory of this spacecraft, error free, are

$$\begin{aligned}
\dot{\mathbf{r}}^I &= \mathbf{v}_I^I && (3.1)\\
\dot{\mathbf{v}}_I^I &= T_I^B \mathbf{a}_{I,non-grav}^B + \mathbf{g}^I(\mathbf{r}^I) && (3.2)\\
\dot{T}_I^B &= T_I^B \Omega_{B/I}^B && (3.3)\\
\dot{q}_B^I &= \frac{1}{2} q_B^I \otimes \omega_{B/I}^B && (3.4)
\end{aligned}$$

where the accelerometer and gyro measurements, $\mathbf{a}_{I,non-grav}^B$ and $\Omega_{B/I}^B$, are error free, the superscript indicates the coordinate frame, the subscript indicates the observation frame, and the double subscript indicates the measurement to one frame with respect to the other; $\mathbf{r}$ is the position vector, $\mathbf{v}$ the velocity vector, $\mathbf{g(r)}$ the acceleration due to gravity in that position, $\mathbf{a}$ is the accelerometer measurement and Ω the gyro measurement, and q the quaternion describing the attitude.

Next, we introduce error sources. The initial position ($\delta\mathbf{r}^I$), velocity ($\delta\mathbf{v}_I^I$) and attitude error ($\delta\theta$) are such that

$$\hat{\mathbf{r}}^I = \mathbf{r}^I + \delta\mathbf{r} \tag{3.5}$$

$$\hat{\mathbf{v}}^I = \mathbf{v}^I + \delta\mathbf{v} \tag{3.6}$$

$$\hat{q}_B^I = \delta q \otimes q_B^I, \qquad \text{where} \qquad \delta q \approx \begin{pmatrix} \delta\theta^B/2 \\ 1 \end{pmatrix} \tag{3.7}$$

The gyro errors can be noise ($\boldsymbol{\nu}_{gyro}$), bias ($\mathbf{b}_{gyro}$), and scale factor (s_{gyro}),

$$\hat{\omega}_{B/I}^B = [I + Diag(s_{gyro})\omega_{B/I}^B + \mathbf{b}_{gyro} + \boldsymbol{\nu}_{gyro} \tag{3.8}$$

$$= [I + S_{gyro}]\omega_{B/I}^B + \mathbf{b}_{gyro} + \boldsymbol{\nu}_{gyro} \tag{3.9}$$

The accelerometer errors may be due to noise ($\boldsymbol{\nu}_{accel}$), bias ($\mathbf{b}_{accel}$), scale factor ($\mathbf{s}_{accel}$), misalignment ($\boldsymbol{\epsilon}$), g^2-sensitivity ($\mathbf{k}_{g^2}$), and g^3-sensitivity ($\mathbf{k}_{g^3}$).

$$\tilde{\mathbf{a}}_{I,non-grav}^B = [I - (\boldsymbol{\epsilon}\times)][I + Diag(s_{accel})]\mathbf{a}_{I,non-grav}^B + \mathbf{b}_{accel} + \cdots + \mathbf{k}_{g^2}\frac{a^2}{g_e} + \mathbf{k}_{g^3}\frac{a^3}{g_e^2} + \boldsymbol{\nu}_{accel} \tag{3.10}$$

$$= [I + E_{accel}][I + S_{accel})]\mathbf{a}_{I,non-grav}^B + \mathbf{b}_{accel} + \mathbf{k}_{g^2}\frac{a^2}{g_e} + \mathbf{k}_{g^3}\frac{a^3}{g_e^2} + \boldsymbol{\nu}_{accel} \tag{3.11}$$

where $a =\mid \mathbf{a}_{I,non-grav}^B \mid$ and $g_e = 9.81 m/s^2$.

Proceeding with a Monte Carlo simulation, all these error sources can be given random input values and the describing equations integrated each time, and after a very large number of simulations (in the order of a thousand) the variance of the result will be the propagated error covariance. This simulation requires an enormous amount of processing, for the simulation needs to be run thousands of times for the result to be reliable.

As previously stated, the Monte Carlo simulation represents a coarse approach to the solution of this problem. Moreover, the determination of the error covariance associated with the estimation of the satellite at all times requires an enormous amount of simulations. The inelegance of this technique and the high processing cost represented a strong motivation to defer this simulation to a later part/phase of the project. Keeping in mind the objectives of this thesis, the Monte Carlo simulation adds very little to the quality of this document and of this student's work in general, representing *de facto* a large and costly waste of time. While it remains a useful tool, this approach is of little academic value; perhaps just for comparison. As we will see, the results obtained with the Kalman filter are of stunning quality, thus not requiring the need for further investigation. Therefore, this student preferred to concentrate his resources on the remaining options. The Monte Carlo simulation was never attempted.

3.2.2 Linear Covariance Analysis

In contrast, Linear Covariance analysis guarantees the same result with much less computational effort, and consequent time. Linear Covariance, or LinCov, is a technique to specifically propagate the covariance of an input to an output. In case of a linear function, such as $y = a \cdot x + b$, it is fairly intuitive that if x follows a distribution defined by a certain mean and variance, then y will follow the same distribution, with mean and variance proportional to x's, according to that function. Though, if the equation is not linear, then that is in general not valid.

Applied to our sample problem, the strategy is to linearize the equations about the true trajectory, without any errors. Once they are linearized, simple stochastic linear system theory can be applied to determine the covariance of the navigation error.

By selecting a state vector

$$\mathbf{x} = (\mathbf{r}^I \quad \mathbf{v}_I^I \quad \boldsymbol{\theta}^B \quad \mathbf{s}_{gyro} \quad \mathbf{b}_{gyro} \quad \boldsymbol{\epsilon} \quad \mathbf{s}_{accel} \quad \mathbf{b}_{accel} \quad \mathbf{k}_{g^2} \quad \mathbf{k}_{g^3})^T$$

the linearized equations of motion are

$$\begin{aligned}
\delta\dot{\mathbf{r}}^I &= \delta\mathbf{v}_I^I && (3.12)\\
\delta\dot{\mathbf{v}}_I^I &= \frac{\partial \mathbf{g}^I}{\partial \mathbf{r}^I}|_{nom}\delta\mathbf{r}^I + T_I^B[\mathbf{a}_{I,non-grav}^B \times]\delta\boldsymbol{\theta}^B + \\
&\quad + T_I^B\Big(- [\boldsymbol{\epsilon}\times]\mathbf{a}_{I,non-grav}^B + Diag(\mathbf{s}_{accel})\mathbf{a}_{I,non-grav}^B + \\
&\quad + \mathbf{b}_{accel} + \mathbf{k}_{g^2}\frac{a^2}{g_e} + \mathbf{k}_{g^3}\frac{a^3}{g_{e^2}}\Big) + \boldsymbol{\nu}_{accel} && (3.13)\\
\delta\dot{\boldsymbol{\theta}}^B &= -\boldsymbol{\omega}_{B/I}^B \times \delta\dot{\boldsymbol{\theta}}^B + Diag(\mathbf{s}_{gyro})\boldsymbol{\omega}_{B/I}^B + \mathbf{b}_{gyro} + \boldsymbol{\nu}_{gyro} && (3.14)\\
\mathbf{s}_{gyro} &= 0_{3\times1} && (3.15)\\
\mathbf{b}_{gyro} &= 0_{3\times1} && (3.16)\\
\boldsymbol{\epsilon} &= 0_{3\times1} && (3.17)\\
\mathbf{s}_{accel} &= 0_{3\times1} && (3.18)\\
\mathbf{b}_{accel} &= 0_{3\times1} && (3.19)\\
\mathbf{k}_{g^2} &= 0_{3\times1} && (3.20)\\
\mathbf{k}_{g^3} &= 0_{3\times1} && (3.21)
\end{aligned}$$

which can be written in state space format as

$$\dot{x} = Fx + G\begin{pmatrix} \boldsymbol{\nu}_{accel} \\ \boldsymbol{\nu}_{gyro} \end{pmatrix} \tag{3.22}$$

where

$$F = \begin{pmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ F_{v,r} & 0_{3\times3} & F_{v,\theta} & 0_{3\times3} & 0_{3\times3} & F_{v,\epsilon} & F_{v,s_{accel}} & F_{v,b_{accel}} & F_{v,k_{g^2}} & F_{v,k_{g^3}} \\ 0_{3\times3} & 0_{3\times3} & F_{\theta,\theta} & F_{\theta,s_{gyro}} & F_{\theta,b_{gyro}} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} & 0_{21\times3} \end{pmatrix} \tag{3.23}$$

$$G = \begin{pmatrix} 0_{3\times3} & 0_{3\times3} \\ I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \\ 0_{21\times3} & I_{21\times3} \end{pmatrix} \tag{3.24}$$

$$F_{v,r} = \frac{\partial \mathbf{g}^I}{\partial \mathbf{r}^I}|_{nom}, F_{v,\theta} = T_I^B[\mathbf{a}_{I,non-grav}^B \times], F_{v,\epsilon} = T_I^B[\mathbf{a}_{I,non-grav}^B \times] \tag{3.25}$$

$$F_{v,s_{accel}} = T_I^B Diag(\mathbf{a}_{I,non-grav}^B), F_{v,b_{accel}} = T_I^B, F_{v,k_{g^2}} = T_I^B \frac{a^2}{g_e}, F_{v,k_{g^3}} = T_I^B \frac{a^3}{g_{e^2}} \tag{3.26}$$

and

$$F_{\theta,\theta} = -[\boldsymbol{\omega}_{B/I}^B \times], F_{\theta,s_{gyro}} = Diag(\boldsymbol{\omega}_{B/I}^B), F_{\theta,b_{gyro}} = I_{3\times3} \tag{3.27}$$

and

$$Q = E\Big[\begin{pmatrix} \boldsymbol{\nu}_{accel} \\ \boldsymbol{\nu}_{gyro} \end{pmatrix}\begin{pmatrix} \boldsymbol{\nu}_{accel}^T \boldsymbol{\nu}_{gyro}^T \end{pmatrix}\Big] = \begin{pmatrix} Q_{accel}\delta(t-\tau) & 0 \\ 0 & Q_{gyro}\delta(T-\tau) \end{pmatrix} \tag{3.28}$$

To propagate the state covariance matrix we use

$$P_{i+1} = \Phi_i P_i \Phi_i^T + GQG^T \Delta t \tag{3.29}$$

where

$$\Phi_i = \exp F\Delta t \approx I + F\Delta t + F^2\Delta t^2/2 \tag{3.30}$$

This approach would deliver the same results as the Monte Carlo simulation in just one execution of the algorithm. The possible caveat is the linearization of the equations of motions, or for more complex problems the development of a linear model. Nonetheless, the elegance of the method and its efficiency made it worthy of consideration.

In exploring the possibilities of LinCov, both Dr. Geller and Anders Forslund were of great value. In an initial phase, enthusiasm towards a new technique directed most of this students attention to the research of the potential of this instrument. While gathering information, and acquiring initial experience, a follow up meeting with both Dr. Geller and Dr. Fullmer resulted in abandoning this approach, for a few reasons. Primarily, it seemed that an experimental technique was less appreciated by the SDL staff. The estimation of PEARL's attitude was a matter to be assigned preferably to a tool that delivers, reliably and consistently. Along that line, xNooch had strong time boundaries. Researching a new technique and risking a setback, or even worse, a failure, was not a chance the development team was ready to take. As a side issue, the thesis being prepared by Anders Forslund does cover in great deal the use of LinCov for spin axis estimation, a problem similar to the one xNooch solves. For that reason, USU found that it would be recommendable to invest part of its research force[2] in a new and potentially ground breaking technique, and another part in a solid, well known and tested method. This partitioning should guarantee results of an exquisitely scientific mark on one side, and of practical engineering matter on the other. Anders' much longer time at USU, paired with our different areas of expertise, quickly advised the division of tasks.
Qualitatively, LinCov does not outclass a Kalman filter. In essence, what is being processed is quite similar, and the mathematics supporting LinCov are

[2]its European contingent, as Dr.Fullmer introduced Anders and myself to the SDL team, one freezing early March Friday morning.

based on the derivations of the Kalman filter[3]. Therefore, to not pursue the usage of LinCov does not make the value of this thesis any lesser. Proof of this may be found in [3] and [4]. At a much later stage, my colleague Anders discovered that MatLab's linearization tool removes the heaviest burden off the application of LinCov, which is the linearization of the system or process being simulated. That possibility opens a number of opportunities yet to be explored. Please refer to his work (currently under development) for further discussions.

3.2.3 The Kalman filter and the Kalman smoother

The following method turned out to be the one xNooch implemented. It was chosen essentially because it's the method with which this writer is and was most comfortable with, along with the reasons discussed above. Again, here, time constraints played a key role.

The standard Kalman filter is subject of many engineering courses, and its widespread reaches many different sorts of applications. It's a very efficient recursive algorithm used to estimate the state(s) of a system after processing a series of noisy measurements taken from whatever sensing equipment is being used. It uses a model of the system to predict the evolution of it, and then corrects that prediction with a noisy measurement[4]. It then corrects its successive prediction, by propagating the related measurement error covariance. Extensive discussion on its application to spacecraft may be found in [1]. A general form of the Kalman filter equations may be found in table 3.1. An interesting application of the same principles governing the Kalman filter has been explored in [3], and called the Kalman smoothing filter.

[3]a long talk with Dr. Geller convinced me of this equipollence, as he himself admitted his own tool would not outperform Kalman's classic.

[4]Alternatively, like in our approach, where there is only the measurement equation, it may figure out a realistic model for the system and follow it, without ever expliciting it.

Table 3.1: A simple discrete-time Kalman filter, as illustrated in [1]

Model	$\mathbf{x}_{k+1} = \Phi_k \mathbf{x}_k + \Gamma_k \mathbf{u}_k + \Upsilon_k \mathbf{w}_k, \quad \mathbf{w}_k \approx N(0, Q_k)$ $\tilde{\mathbf{y}}_k = H_k \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k \approx N(0, R_k)$
Initialize	$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0$ $P_0 = E\{\tilde{\mathbf{x}}(t_0)\tilde{\mathbf{x}}^T(t_0)\}$
Gain	$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1}$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k[\tilde{\mathbf{y}}_k - H_k \hat{\mathbf{x}}_k^-]$ $P_k^+ = [I - K_k H_k] P_k^-$
Propagation	$\hat{\mathbf{x}}_{k+1}^- = \Phi_k \hat{\mathbf{x}}_k^+ + \Gamma_k \hat{\mathbf{u}}_k$ $P_{k+1}^- = \Phi_k P_k^+ \Phi_k^T + \Upsilon_k Q_k \Upsilon_k^T$

The so called Kalman smoothing filter is an application of the filter to a batch of data. The standard version of the filter is in fact recursive, and only ever keeps track of one state vector and one measurement, and the related error covariances. For the PEARL problem, the entire collection of data is available, and thus a batch estimation technique is surely advisable for it can perform better than a recursive technique. The Kalman smoothing filter does exactly that.

The idea behind it is simple: it takes a standard version of the filter, and runs it through the entire set of data, recording the new filtered data and the related error covariances. It then flips the data over, hence inverting time, and runs the filter again, recording the results. The two result sets are then aligned in time, and weight averaged, based on the associated error covariance. The resulting set is the final result set of data.

> If we call our state $\mathbf{x}(t)$, then the smoothed estimate based on all measurements between 0 and T is denoted by $\hat{\mathbf{x}}(t|T)$. An optimal smoother can be thought of as a suitable combination of two optimal filters. One of the filters, called a "forward filter", operates on all the data before time t and produces the estimate $\hat{\mathbf{x}}(t)$; the other filter, called a "backward filter", operates on all the data after time t and produces the estimate $\hat{\mathbf{x}}(t)_b$. Together these two filters utilize *all* the available information. . . . This suggests that the optimal combination of $\hat{\mathbf{x}}(t)$ and $\hat{\mathbf{x}}(t)_b$ will, indeed, yield the optimal smoother; proof of this assertion can be found in [7]. . . . In *fixed-interval smoothing*, the initial and final times 0 and T are fixed and the estimate $\hat{\mathbf{x}}(t|T)$ is sought, where t varies from 0 to T. [3]

Refer to [3] for the derivation of the smoother. Table 3.2 shows the smoother itself, in general form.

Table 3.2: A standard Kalman smoother, as illustrated in [3]

Model	$\dot{\mathbf{x}}(t) = F(t)\mathbf{x}(t) + G(t)\mathbf{w}(t), \quad \mathbf{w}(T) \approx N(0, Q(t))$ $\mathbf{z}(t) = H(t)\mathbf{x}(t) + \mathbf{v}(t), \quad \mathbf{v}(t) \approx N(0, R(t))$
Initialize other assumptions	$E[\mathbf{x}(0)] = \hat{\mathbf{x}}_0, \quad E[(\mathbf{x}(0) - \hat{\mathbf{x}}_0)(\mathbf{x}(0) - \hat{\mathbf{x}}_0)^T] = P_0$ $E[\mathbf{w}(t_1) \cdot \mathbf{v}^T(t)] = 0$ for all t; $\quad R^{-1}(t)$ exists
Forward Filter **Error Covariance Propagation**	$\dot{\hat{\mathbf{x}}}(t) = F(t)\hat{\mathbf{x}}(t) + P(t)H^T(t)R^{-1}(t)[\mathbf{z}(t) - H(t)\hat{\mathbf{x}}(t)],$ $\hat{\mathbf{x}}(0) = \hat{\mathbf{x}}_0$ $\dot{P} = F(t)P(t) + P(t)F(t)^T + G(t)Q(t)G(t)^T -$ $-P(t)H^T(t)R^{-1}(t)H(t)P(t),$ $P(0) = P_0$
Backward Filter $(\tau = T - t)$ **Error Covariance Propagation** $(\tau = T - t)$	$\frac{d}{d\tau}\mathbf{s}(T-\tau) = [F^T(T-\tau) - P_b^{-1}(T-\tau)G(T-\tau)\cdot$ $\cdot Q(T-\tau)G^T(T-\tau)]\mathbf{s}(T-\tau) +$ $+H^T(T-\tau)R^{-1}(T-\tau)\mathbf{z}(T-\tau),$ $\mathbf{s}(0) = 0$ $\frac{d}{d\tau}P_b^{-1}(T-\tau) = P_b^{-1}(T-\tau)F(T-\tau) - F^T(T-\tau)\cdot$ $\cdot P_b^{-1}(T-\tau) - P_b^{-1}(T-\tau)G(T-\tau)\cdot$ $\cdot Q(T-\tau)G^T(T-\tau)P_b^{-1}(T-\tau) +$ $+H^T(T-\tau)R^{-1}(T-\tau)H(T-\tau),$ $P_b^{-1}(0) = 0$
Optimal Smoother **Error Covariance Propagation**	$\hat{\mathbf{x}}(t\|T) = P(t\|T)[P^{-1}(t)\hat{\mathbf{x}}(t) + \mathbf{s}(t)]$ $= [I + P(t)P_b^-1(t)]\hat{\mathbf{x}}(t) + P(t\|T)\mathbf{s}(t)$ $P(t\|T) = [P^-1(t) + P_b^{-1}(t)]^{-1}$ $= P(t) - P(t)P_b^{-1}(t)[I + P(t)P_b^{-1}(t)]^{-1}P(t)$

Chapter 4

Implementation and results

As previously explained, the problem at hand has been tackled choosing an incremental approach: devising at first a solution to a simpler problem, and iteratively adding complexity to the problem and adjusting the solution. The first step in this approach, having chosen the Kalman smoother as a primary solution, was to derive a Kalman filter capable of estimating the attitude of PEARL based on the nature of the measurements we knew we were going to obtain. Again, as a reminder, xNooch shall take as inputs the measurements and the associated error covariances, and output an attitude estimate along with an accuracy of that estimate.

4.1 Derivation of the Kalman Filter

This process has been aided greatly by [1] and by Dr. Fullmer's course on Attitude Control, attended while at USU, and the related course notes [8]. While attitude may be expressed in a number of ways (Euler angles, Rodrigues parameters, rotation vector...) it seemed appropriate to direct focus on the *quaternion* description. Quaternions, as known, do not present singularities and hence rid us of a potential problem. Quaternions though follow tremendously non-linear operations, for they must obey a normalization constraint:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = \mathbf{q}^T\mathbf{q} = 1 \tag{4.1}$$

and this constraint may be violated (and most likely is) by the linear measurement updates of the filter. This obstacle may be overcome by utilizing a multiplicative error quaternion, which we will introduce next. This approach, along with a normalization step in the algorithm, is very effective with a small angle approximation.

4.1.1 Dealing with quaternions

As we know, a quaternion product may be written as $R(q'') = R(q')R(q)$, or

$$\begin{bmatrix} q_1'' \\ q_2'' \\ q_3'' \\ q_4'' \end{bmatrix} = \begin{bmatrix} q_4' & q_3' & -q_2' & q_1' \\ -q_3' & q_4' & q_1' & q_2' \\ q_2' & -q_1' & q_4' & q_3' \\ -q_1' & -q_2' & -q_3' & q_4' \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{4.2}$$

also definable as $q'' = q' \otimes q$.

Recalling that $q = \begin{bmatrix} \vec{q} \\ q_4 \end{bmatrix}$ and $q^{-1} = \begin{bmatrix} \vec{-q} \\ q_4 \end{bmatrix}$.

According to Rodrigues' formula,

$$R(q) = [q_4^2 - q^2]I_{3\times3} + 2\vec{q}\vec{q}^T - 2q_4 Q \tag{4.3}$$

where

$$Q = q^\times = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{4.4}$$

These formulations allow for both $R^T(q) = R^{-1}(q)$ and $R^{-1}(q) = R(q^{-1})$.

We may then define our measurement error as

$$R(q) = R(\delta q)R(\hat{q}) \tag{4.5}$$

$$R_I^T = R_M^T R_I^M \tag{4.6}$$

where $\begin{matrix} T & = & \textit{true} \\ M & = & \textit{measured} \\ I & = & \textit{inertial} \end{matrix}$ and $\begin{matrix} q & = & \textit{true quaternion} \\ \hat{q} & = & \textit{measured quaternion} \\ \delta q & = & \textit{quaternion error} \end{matrix}$ so

$$R(\delta q) = R(q)R^{-1}(\hat{q}) = R(q)R(\hat{q}^{-1}) \tag{4.7}$$

or

$$\delta q = q \otimes \hat{q}^{-1} \tag{4.8}$$

and by differentiation

$$\delta \dot{q} = \dot{q} \otimes \hat{q}^{-1} + q \otimes \dot{\hat{q}}^{-1} \tag{4.9}$$

.

Since $R(q)R^{-1} = I \Rightarrow I(q'') = R(q)R(q^{-1}) \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = q \otimes q^{-1}$ and substituting back into 4.9 we get

$$0 = \dot{q} \otimes q^{-1} + q \otimes \dot{q}^{-1} \tag{4.10}$$

$$0 = \dot{\hat{q}} \otimes \hat{q}^{-1} + \hat{q} \otimes \dot{\hat{q}}^{-1} \tag{4.11}$$

From [5] we know that $\dot{q} = \frac{1}{2}\Omega(\omega)q$ where

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} = \begin{bmatrix} -\omega^\times & \omega \\ -\omega^T & 0 \end{bmatrix}$$

and if we're using measured values then $\dot{\hat{q}} = \frac{1}{2}\Omega(\hat{\omega})\hat{q}$.

Substituting into 4.11:

$$\frac{1}{2}\Omega(\hat{\omega})\hat{q} \otimes \hat{q}^{-1} + \hat{q} \otimes \dot{\hat{q}}^{-1} = 0 \quad (4.12)$$

$$\frac{1}{2}\begin{bmatrix} -\omega^\times & \omega \\ -\omega^T & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \hat{q} \otimes \dot{\hat{q}}^{-1} = 0 \quad (4.13)$$

$$\frac{1}{2}\begin{bmatrix} \hat{\omega} \\ 1 \end{bmatrix} + \hat{q} \otimes \dot{\hat{q}}^{-1} = 0 \quad (4.14)$$

$$\underbrace{-\frac{1}{2}\begin{bmatrix} \hat{\omega} \\ 1 \end{bmatrix}}_{R(q'')} = \underbrace{\hat{q} \otimes \dot{\hat{q}}^{-1}}_{R(q')R(q)} \quad (4.15)$$

$$R(q) = R^{-1}(q')R(q'') \quad (4.16)$$

$$R(q) = R(q^{-1})R(q'') \quad (4.17)$$

$$q = q'^{-1} \otimes q'' \quad (4.18)$$

which, with $q \Rightarrow \dot{\hat{q}}^{-1}, \quad q' \Rightarrow \hat{q}, \quad q'' \Rightarrow -\frac{1}{2}\begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix}$ becomes

$$\dot{\hat{q}}^{-1} = \hat{q}^{-1} \otimes \left(-\frac{1}{2}\begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix}\right) \quad (4.19)$$

Now, remembering:

$$\dot{q} = \frac{1}{2}\Omega(\omega)q \tag{4.20}$$

if we define $q'' = q' \otimes q = Q(q)$ then matrices $Q(q)$ and $\Omega\left(\begin{bmatrix}\omega\\0\end{bmatrix}\right)$ are identical.

$$\Omega(\omega)q = \begin{bmatrix}\omega\\0\end{bmatrix} \otimes q \tag{4.21}$$

$$\dot{q} = \frac{1}{2}\begin{bmatrix}0 & \omega_z & -\omega_y & \omega_x\\ -\omega_z & 0 & \omega_x & \omega_y\\ \omega_y & -\omega_x & 0 & \omega_z\\ -\omega_x & -\omega_y & -\omega_z & 0\end{bmatrix}\begin{bmatrix}q_1\\q_2\\q_3\\q_4\end{bmatrix} = \frac{1}{2}\begin{bmatrix}\omega\\0\end{bmatrix} \otimes q \tag{4.22}$$

Remembering 4.9, we obtain

$$\delta\dot{q} = \frac{1}{2}\begin{bmatrix}\omega\\0\end{bmatrix} \otimes q \otimes \hat{q}^{-1} + q \otimes \left\{-\frac{1}{2}\hat{q}^{-1}\begin{bmatrix}\hat{\omega}\\0\end{bmatrix}\right\} \tag{4.23}$$

Remembering $q \otimes \hat{q}^{-1} = \delta q$

$$\delta\dot{q} = \frac{1}{2}\left\{\begin{bmatrix}\omega\\0\end{bmatrix}\delta q - \delta q\begin{bmatrix}\hat{\omega}\\0\end{bmatrix}\right\} \tag{4.24}$$

On a note, as we will use this equality later:

$$\delta q \otimes \begin{bmatrix}\hat{\omega}\\0\end{bmatrix} = \begin{bmatrix}\delta q_4 & \delta q_3 & -\delta q_2 & \delta q_1\\ -\delta q_3 & \delta q_4 & \delta q_1 & \delta q_2\\ \delta q_2 & -\delta q_1 & \delta q_4 & \delta q_3\\ -\delta q_1 & -\delta q_2 & -\delta q_3 & \delta q_4\end{bmatrix}\begin{bmatrix}\hat{\omega}_x\\\hat{\omega}_y\\\hat{\omega}_z\\0\end{bmatrix} \tag{4.25}$$

$$= \begin{bmatrix} 0 & -\hat{\omega}_z & \hat{\omega}_y & \hat{\omega}_x \\ \hat{\omega}_z & 0 & -\hat{\omega}_x & \hat{\omega}_y \\ -\hat{\omega}_y & \hat{\omega}_x & 0 & \hat{\omega}_z \\ -\hat{\omega}_x & -\hat{\omega}_y & -\hat{\omega}_z & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ \delta q_4 \end{bmatrix} \tag{4.26}$$

$$= \begin{bmatrix} \hat{\omega}^\times & \hat{\omega} \\ \hat{-}\omega^T & 0 \end{bmatrix} \delta q = \Gamma(\hat{\omega})\delta q \tag{4.27}$$

Let $\omega = \hat{\omega} + \delta\omega$; then equation 4.24 becomes:

$$\delta\dot{q} = \frac{1}{2}\left\{ \begin{bmatrix} \hat{\omega} + \delta\omega \\ 0 \end{bmatrix} \otimes \delta q - \delta q \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \right\} \tag{4.28}$$

$$\delta\dot{q} = \frac{1}{2}\left\{ \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \otimes \delta q - \delta q \otimes \begin{bmatrix} \hat{\omega} \\ 0 \end{bmatrix} \right\} + \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \otimes \delta q \tag{4.29}$$

$$\begin{aligned} \delta\dot{q} &= \frac{1}{2}\left\{ \Omega(\hat{\omega})\delta q - \Gamma(\hat{\omega})\delta q \right\} + \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \otimes \delta q \\ &= \frac{1}{2}\left\{ \begin{bmatrix} -\hat{\omega}^\times & \hat{\omega} \\ -\hat{\omega}^T & 0 \end{bmatrix} - \begin{bmatrix} \hat{\omega}^\times & \hat{\omega} \\ -\hat{\omega}^T & 0 \end{bmatrix} \right\} \delta q + \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \otimes \delta q \\ &= \begin{bmatrix} -\hat{\omega}^\times & 0 \\ 0 & 0 \end{bmatrix} \delta q + \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \otimes \delta q \end{aligned} \tag{4.30}$$

Let $\delta q = \begin{bmatrix} \delta\rho \\ \delta q_4 \end{bmatrix}$; then

$$\delta\dot{q} = -\hat{\omega}^\times \delta\rho + \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \otimes \delta q \tag{4.31}$$

Linearization of the last term

$$N - \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \otimes \delta q = \Omega(\delta\omega) \begin{bmatrix} \rho \\ \delta q_4 \end{bmatrix} \tag{4.32}$$

where $\delta q_4 = \sqrt{(1 - \delta q_1^2 - \delta q_2^2 - \delta q_3^2)}$

$$N = \frac{1}{2} \begin{bmatrix} 0 & \delta\omega_z & -\delta\omega_y & \delta\omega_x \\ -\delta\omega_z & 0 & \delta\omega_x & \delta\omega_y \\ \delta\omega_y & -\delta\omega_x & 0 & \delta\omega_z \\ -\delta\omega_x & -\delta\omega_y & -\delta\omega_z & 0 \end{bmatrix} \begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \\ (1 - \delta q_1^2 - \delta q_2^2 - \delta q_3^2)^{1/2} \end{bmatrix} \tag{4.33}$$

Careful observation shows how, for small angle variation and slow rotation, the N matrix greatly reduces. Without going through trivial math, in can be demonstrated that,

$$\lim_{\delta q, \delta\omega \to 0,0} N = \frac{1}{2} \begin{bmatrix} \delta\omega_x \\ \delta\omega_y \\ \delta\omega_z \\ 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix} \tag{4.34}$$

$$\delta\dot{q} = -\begin{bmatrix} \hat{\omega}^\times \delta\rho \\ 0 \end{bmatrix} + \begin{bmatrix} \delta\omega \\ 0 \end{bmatrix}$$

$$\left.\begin{aligned} &\delta\dot{\rho} = -\hat{\omega}^\times \delta\rho + 1/2\delta\omega \\ &\delta\dot{q}_4 = 0 \end{aligned}\right\} \tag{4.35}$$

4.1.2 Rate Sensor Model

In this section we will show how the derivation of the rate sensor integrates into the extended version of the Kalman filter we will be using. The derivation is once again guided by [1] and was rather surprising to this writer. Nevertheless, it revealed its worthiness in the testing phase. We will start by analyzing the model of the sensor.

$$\tilde{\omega} = \omega + \beta + \eta_\nu \tag{4.36}$$

Here, $\tilde{\omega}$ is the measured angular velocity, and it is decomposed into the true angular velocity ω, the bias effect β, and the measurement noise η_ν. We will also be considering the bias drift effect, namely $\dot{\beta} = \eta_u$. As these are vectorial dimensions, we will define the standard deviations of the two noises as σ_ν and σ_u respectively, like so: $\eta_\nu = \sigma_n u^2 I_{3\times3}$ and $\eta_u = \sigma_u^2 I_{3\times3}$. As usual we'll be using a hat to indicate estimated values, and we will design the filter to estimate the bias drift, assuming it will not vary in time. This is, of course, a simplification.

$$\delta\omega = \omega - \hat{\omega} = \tilde{\omega} - \beta - \eta_\nu - \tilde{\omega} + \hat{\beta} = -\underbrace{(\beta - \hat{\beta})}_{\Delta\beta} - \eta_\nu = -(\Delta\beta + \eta_\nu) \tag{4.37}$$

Substituting this result into 4.35 yields

$$\delta\dot{\rho} = -[\hat{\omega}^\times]\delta\rho\frac{1}{2}(\Delta\beta + \eta_\nu) \tag{4.38}$$

From [5] we know that $q = \begin{bmatrix} \rho \\ q_4 \end{bmatrix} = \begin{bmatrix} e_x \sin(\alpha/2) \\ e_y \sin(\alpha/2) \\ e_z \sin(\alpha/2) \\ \cos(\alpha/2) \end{bmatrix}$ and linearizing this brings:

$$\begin{aligned} \rho_x &= e_x \sin\frac{\alpha}{2} \\ \delta\rho_x &= \rho_x(\alpha + \delta\alpha) - \rho_x(\alpha) \\ &= \frac{\partial\rho_x}{\partial\alpha}|_{\alpha=0}\delta\alpha \\ &= \frac{1}{2}e_x \cos\frac{\alpha}{2}|_{\alpha=0}\delta\alpha \end{aligned} \tag{4.39}$$

$$= \frac{e_x}{2}\delta\alpha \tag{4.40}$$

and so on for the y and z directions. A different transformation shows that:

$$\delta\rho = \frac{1}{2}\begin{bmatrix} e_x\delta\alpha \\ e_y\delta\alpha \\ e_z\delta\alpha \end{bmatrix} = \frac{1}{2}\delta\vec{\alpha} = \frac{1}{2}\begin{bmatrix} \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix} \tag{4.41}$$

We have therefore found a direct relation with the Euler Angles.

So:

$$\delta\dot{\rho} = \frac{1}{2}\delta\dot{\alpha} = -\frac{1}{2}[\hat{\omega}^{\times}]\delta\alpha - \frac{1}{2}(\Delta\beta + \eta_\nu) \qquad \text{or} \qquad \delta\dot{\alpha} = -[\hat{\omega}^{\times}]\delta\alpha - (\Delta\beta + \eta_\nu) \tag{4.42}$$

which has finally brought us to our choice of a state vector, for the Kalman filter. It resulted in a fairly straight-forward choice, for the literature is not abundant and considering our hardware constraints this was the only suitable implementation. The derivation, however, differs in several points from reference [1], which is the inspirational source. The differences are due to the fact that the derivation was redone by this student, with the objective of
a) understanding the material;
b) looking for ways to improve it, or make more adequate for our particular mission.
While task a was accomplished, task b proved to be too difficult of an achievement. Nevertheless, the final result of the derivation matches Crassidi's, and is therefore theoretically reliable. The testing proves it is indeed effective.

$$\begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} \Rightarrow \begin{bmatrix} \text{``eulerangles''} \\ \text{``bias''} \end{bmatrix}$$

$$\begin{bmatrix} \delta\dot{\alpha} \\ \delta\dot{\beta} \end{bmatrix} = \begin{bmatrix} -\hat{\omega}^{\times} & -I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} \begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} + \begin{bmatrix} -I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \begin{bmatrix} \eta_{\nu} \\ \eta_{u} \end{bmatrix} \tag{4.43}$$

equivalent to the more traditional

$$\delta\dot{\mathbf{x}} = F(x)\delta x + Gw \tag{4.44}$$

.

4.1.3 The measurement equation

From the very beginning we knew we could rely on at least more than one vectorial measurements (sun sensor and magnetometer), and therefore planned accordingly. The measurement equation is:

$$s^B = R(q)s^I \tag{4.45}$$

where B stands for body coordinates, I for inertial, and *R(q)* is the rotation matrix obtained from the attitude quaternion that relates the two coordinates systems. Our measurement clearly comes in body frame coordinates. The inertial vector, or, more precisely, the earth centered inertial coordinate system vector, is a function of the universe and mission models. It was not in this project's scope to generate the ECI vector, but rather just to know how to deal with it and its significance. The measurement equation defines its relevance.

In general:

$$\left.\begin{array}{rcl} b_1 & = & R(q)r_1 \\ b_2 & = & R(q)r_2 \\ & \vdots & \\ b_m & = & R(q)r_m \end{array}\right\} \text{m vectors observed} \tag{4.46}$$

or, including measurement errors

$$\left.\begin{array}{rcl} b_{1meas} & = & R(q)r_1 + \nu_1 \\ b_{2meas} & = & R(q)r_2 + \nu_2 \\ & \vdots & \\ b_{Kmeas} & = & R(q)r_m + \nu_m \end{array}\right\} \tag{4.47}$$

or, changing notation:

$$\tilde{y} = h(\tilde{x}) + \nu \tag{4.48}$$

Now, back to the quaternion. This filter is designed precisely for mission attitude determination, but not for the detumbling phase (PEARL will utilize another algorithm for detumbling, which is not a part of this report). This means that it is appropriate to make the small angle assumption, and simplify our model and, therefore, our filter and implementation.

Pulling back Rodrigues' formula (4.3), but adapting it to small angles ($\delta q_4 \approx 1, \vec{\rho}^2 \approx 0, \vec{\rho} \cdot \vec{\rho}^T \approx 0$), we obtain

$$R(\delta q) = I_{3\times3} - \delta\alpha^{\times} = \begin{bmatrix} 1 & \delta\psi & -\delta\theta \\ -\delta\psi & 1 & \delta\phi \\ \delta\theta & -\delta\phi & 1 \end{bmatrix} \tag{4.49}$$

where $2\delta\rho = \delta\alpha = \begin{bmatrix} \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix}$ and these are the approximated Euler angles, valid only for small rotation scenarios.

$$\begin{array}{rrcl} \text{Truth} & b & = & R(q)r \\ \text{Predicted} & \tilde{b}^- & = & R(\hat{q}^-)r \end{array} \tag{4.50}$$

$$\begin{array}{rrcl} \text{Error in the observed vector} & \delta b & = & b - \hat{b}^- \\ & & = & (R(q) - R(\hat{q}^-))r \\ & & = & [I_{3\times3} - \delta\alpha^\times - I_{3\times3}]R(\hat{q}^-)r \\ & & = & -\delta\alpha^\times R(\hat{q}^-)r \\ & & = & [R(\hat{q}^-)r]^\times \delta\alpha \end{array} \tag{4.51}$$

The measurement equation therefore becomes $\tilde{y}_m = \underbrace{\hat{b}_m(\hat{q}^-) + [R(\hat{q}^-)r]^\times \delta\alpha}_{h(\hat{q}^-,\alpha)} + \nu_m$, or more in general $\tilde{y} = h(\delta\alpha, \delta\beta) + \nu$. By linearizing vector h about $\delta\alpha_0 = 0$ and $\delta\beta_0 = 0$ we get

$$h(\delta\alpha, \delta\beta) = \hat{b}^-(\hat{q}^-) + [R(\hat{q}^-)]^\times \delta\alpha \tag{4.52}$$

$$\tilde{y} - \hat{b}^-(\hat{q}^-) = \underbrace{\begin{bmatrix} (R(\hat{q}^-)r)^\times & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{3m\times6} \begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} + \nu \tag{4.53}$$

$$\tilde{y} - \hat{b}^-(\hat{q}^-) = H(\hat{q}^-)\mathbf{X} + \nu \tag{4.54}$$

which is our final measurement equation[1], fitted to our problem.

4.1.4 Discrete time form

This section briefly illustrates the derivation of the $\bar{\Omega}(\omega)$ matrix, which allows a smooth conversion to discrete time measurements. It is voluntarily extremely synthetic.

$$\dot{\hat{q}} = \frac{1}{2}\Omega(\hat{\omega})\hat{q} \iff \dot{x} = Ax \tag{4.55}$$

Assuming

$$x(t) = e^{At}x_0 \tag{4.56}$$

We attempt to apply this type of solution to the quaternion form

$$\dot{x}(t) = Ax(t) \tag{4.57}$$

We choose T as our sampling rate

$$x(T) = e^{AT}x(0) \tag{4.58}$$

$$x(2T) = e^{2AT}x(0) = e^{AT}x(0)\ ecc\ldots ecc\ldots \tag{4.59}$$

[1]almost a month of tests were spent trying to understand why the filter presented an unstable behavior. It seemed to work, but the higher the spin rate, the quicker the estimation would be knocked off course, sometimes to return, but usually to just go bananas. A few PhDs and a couple of tenured professors scratched their heads for quite a while, in the attempt of solving this mysterious abnormality. At times it seemed a specific instant would be key to instability. Others, the estimation would slowly drift away. But if simulated non-spinning, it worked just fine. Every conceivable alteration was done, both in tuning and in simplifying. An embarrassing amount of data sets had to be developed, to see if the problem lay somewhere in between. The morning before departing from USU, heading back to CTU, was when we were able to identify a very small error in this equation, missing brackets to be precise. With the correction, the problem disappeared.

So, for any k,

$$x_{k+1} = e^{AT} x_k \tag{4.60}$$

Applied to the quaternion

$$\hat{q}_k + 1 = exp[\frac{1}{2}\Omega(\hat{\omega}T]\hat{q}_k \tag{4.61}$$

Matrix exponential form:

$$exp[\frac{1}{2}\Omega(\hat{\omega}T] = \sum_{j=0}^{\infty}[\frac{1}{2}\Omega(\hat{\omega}T]^j/j! \tag{4.62}$$

which could be then separated in odd and even terms. But:

$$\Omega(\hat{\omega})\Omega(\hat{\omega}) = \begin{bmatrix} 0 & \hat{\omega}_z & -\hat{\omega}_y & \hat{\omega}_x \\ -\hat{\omega}_z & 0 & \hat{\omega}_x & \hat{\omega}_y \\ \hat{\omega}_y & -\hat{\omega}_x & 0 & \hat{\omega}_z \\ -\hat{\omega}_x & -\hat{\omega}_y & -\hat{\omega}_z & 0 \end{bmatrix} \begin{bmatrix} 0 & \hat{\omega}_z & -\hat{\omega}_y & \hat{\omega}_x \\ -\hat{\omega}_z & 0 & \hat{\omega}_x & \hat{\omega}_y \\ \hat{\omega}_y & -\hat{\omega}_x & 0 & \hat{\omega}_z \\ -\hat{\omega}_x & -\hat{\omega}_y & -\hat{\omega}_z & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} -\hat{\omega}^2 & 0 & 0 & 0 \\ 0 & -\hat{\omega}^2 & 0 & 0 \\ 0 & 0 & -\hat{\omega}^2 & 0 \\ 0 & 0 & 0 & -\hat{\omega}^2 \end{bmatrix} = -||\hat{\omega}||^2 I_{4\times4}$$

$$\Omega(\hat{\omega})^{2k} = (-1)^k ||\hat{\omega}||^{2k} I_{4\times4} \tag{4.63}$$

$$\Omega(\hat{\omega})^{2k+1} = (-1)^k ||\hat{\omega}||^{2k} \Omega(\hat{\omega}) \tag{4.64}$$

which is also found in [1]We identified an infinte series of sines and cosines

$$exp[1/2\ \Omega(\hat{\omega}T] = \cos\left(\frac{||\hat{w}||T}{2}\right) I_{4\times4} + \frac{\Omega(\hat{\omega})}{||\hat{w}||} sin\left(\frac{||\hat{w}||T}{2}\right) \tag{4.65}$$

Defining:

$$C = \cos\left(\frac{||\hat{w}||T}{2}\right) \qquad \text{and} \qquad \hat{\psi} = \frac{\hat{w}}{||\hat{w}||}\sin\left(\frac{||\hat{w}||T}{2}\right) \tag{4.66}$$

$$exp[1/2\ \Omega(\hat{\omega}T] = CI + \begin{bmatrix} -\hat{\psi}^{\times} & \hat{\psi} \\ -\hat{\psi} & 0 \end{bmatrix} = \bar{\Omega}(\hat{\omega}) \tag{4.67}$$

or in more compact form

$$\hat{q}_{k+1} = \bar{\Omega}(\hat{\omega}_k)\hat{q}_k = \begin{bmatrix} CI_{3\times3} - \hat{\psi}_k^{\times} & \hat{\psi}_k \\ -\hat{\psi}_k & C \end{bmatrix}\hat{q}_k \tag{4.68}$$

which is the discrete time propagation of the quaternion estimate

4.1.5 Measurement updates

Quaternion

Assuming a measurement update[2] comes in, $\tilde{y}$: then

$$\hat{x}_k^+ - \hat{x}_k^- = \Delta x_k = \begin{bmatrix} \delta\hat{\alpha}_k^+ \\ \delta\hat{\beta}_k^+ \end{bmatrix} = K_k(y_k - h(\hat{x}_k)) \tag{4.69}$$

$$R(\hat{q}_k^+) = R(\delta\hat{q}_k^+)R(\hat{q}_k^-) \tag{4.70}$$

or

$$\hat{q}_k^+ = \begin{bmatrix} \frac{1}{2}\delta\hat{\alpha}_k^+ \\ 1 \end{bmatrix} \otimes \hat{q}_k^- = \frac{1}{2}\begin{bmatrix} 2 & \delta\alpha_3 & -\delta\alpha_2 & \delta\alpha_1 \\ -\delta\alpha_3 & 2 & \delta\alpha_1 & \delta\alpha_2 \\ \delta\alpha_2 & -\delta\alpha_1 & 2 & \delta\alpha_3 \\ -\delta\alpha_1 & -\delta\alpha_2 & -\delta\alpha_3 & 2 \end{bmatrix}\begin{bmatrix} q_1^- \\ q_2^- \\ q_3^- \\ q_4^- \end{bmatrix} \tag{4.71}$$

[2]in this section, all values are vectorial. To keep the writing slimmer, none of them will be marked as such

$$= \begin{bmatrix} q_1^- \\ q_2^- \\ q_3^- \\ q_4^- \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \delta\alpha_3 q_2 & -\delta\alpha_2 q_3 & \delta\alpha_1 q_4 \\ -\delta\alpha_3 q_1 & \delta\alpha_1 q_3 & \delta\alpha_2 q_4 \\ \delta\alpha_2 q_1 & -\delta\alpha_1 q_2 & \delta\alpha_3 q_4 \\ -\delta\alpha_1 q_1 & -\delta\alpha_2 q_2 & -\delta\alpha_3 q_3 \end{bmatrix} \tag{4.72}$$

or

$$\hat{q}_k^+ = \hat{q}_k^- + \frac{1}{2}\begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_2 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}\begin{bmatrix} \delta\alpha_1 \\ \delta\alpha_2 \\ \delta\alpha_3 \end{bmatrix} \tag{4.73}$$

$$\hat{q}_k^+ = \hat{q}_k^- + \frac{1}{2}\Xi(\hat{q}_k^-)\delta\hat{\alpha}_k^+ \tag{4.74}$$

Bias

$$\hat{\beta}_k^+ = \hat{\beta}_k^- + \delta\hat{\beta}_k^+ \tag{4.75}$$

Rate ω

$$\hat{\omega}_k^+ = \underbrace{\tilde{\omega}_k}_{\text{Measured rate}} - \hat{\beta}_k^+ \tag{4.76}$$

4.1.6 Propagation of the state covariance

Looking back at the rate sensor model, these were our describing equations:

$$\tilde{\omega} = \omega + \beta + \eta_\nu \tag{4.77}$$

$$\dot{\beta} = \eta_u \tag{4.78}$$

Now let's assume that

$$E[\eta_u \eta_\nu^T] = \sigma_u^2 I_{3\times 3} \quad (4.79)$$
$$E[\eta_\nu \eta_u^T] = \sigma_\nu^2 I_{3\times 3} \quad (4.80)$$

Then, [1] tells us that

$$P_k + 1^- = \Phi_k P_k^+ \Phi_k^T + G_k Q_k G_k^T \quad (4.81)$$

where

$$\Phi = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \quad (4.82)$$
$$\Phi_{11} = I_{3\times 3} - \hat{w}^\times \frac{\sin(||\hat{w}||T)}{||\hat{w}||} + \hat{w}^\times \hat{w}^\times \left[\frac{1-\cos(||\hat{w}||T)}{||\hat{w}||^2}\right] \quad (4.83)$$
$$\Phi_{12} = \hat{w}^\times \left[\frac{1-\cos(||\hat{w}||T)}{||\hat{w}||^2}\right] - I_{3\times 3}T \quad (4.84)$$
$$\Phi_{21} = 0_{3\times 3} \quad (4.85)$$
$$\Phi_{22} = I_{3\times 3} \quad (4.86)$$
$$G_k = \begin{bmatrix} -I_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & I_{3\times 3} \end{bmatrix} \quad (4.87)$$
$$Q_k = \begin{bmatrix} (\sigma_\nu^2 + \frac{1}{3}\sigma_u^2 T^3) I_{3\times 3} & -\frac{1}{2}\sigma_u^2 T^2 I_{3\times 3} \\ -\frac{1}{2}\sigma_u^2 T^2 I_{3\times 3} & \sigma_u^2 T I_{3\times 3} \end{bmatrix} \quad (4.88)$$
$$(4.89)$$

4.1.7 The final derivation of the extended quaternion attitude Kalman filter

Here is in brief a collection of the entire algorithm we have just derived, and which will be shown at work in the following chapter.

The three initial guesses of quaternion, bias and the related error covariance (presumably a relatively high value) are $\hat{q}_k^- \qquad \hat{\beta}_k^- \qquad P_k^-$.

1. Kalman gain calculation:

a)

$$R(\hat{q}_k^-) \tag{4.90}$$

b)

$$H_k(\hat{q}_k^-) = \begin{bmatrix} (R(\hat{q}_k^-)r_1^I)^\times & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ (R(\hat{q}_k^-)r_m^I)^\times & 0 & 0 & 0 \end{bmatrix} \tag{4.91}$$

c)

$$K_k = P_k^- H_k^T [H_k P_k^- H_k^T + R_k]^{-1} \tag{4.92}$$

where R is none other than the measurement noise covariance;

2. Measurement update: y_k, $\tilde{w}_k$.

$$\begin{aligned} \Delta\hat{x}_k^+ &= K_k(y_k - h(\hat{q}^-)) && (4.93) \\ \Delta\hat{x}_k^+ &= \begin{bmatrix} \delta\hat{\alpha}_k^+ \\ \delta\hat{\beta}_k^+ \end{bmatrix} && (4.94) \\ \hat{q}_k^+ &= \hat{q}_k^- + \frac{1}{2}\Xi(\hat{q}_k^-)\delta\hat{\alpha}_k^+ && (4.95) \\ \hat{\beta}_k^+ &= \hat{\beta}_k^- + \delta\hat{\beta}_k^+ && (4.96) \\ \hat{\omega}_k^+ &= \tilde{\omega}_k - \hat{\beta}_k^+ && (4.97) \\ P_k^+ &= [I - K_k H_k(\hat{q}_k^-)]P_k^- && (4.98) \end{aligned}$$

3. Propagation

$$\begin{aligned} \hat{q}_{k+1}^- &= \bar{\Omega}(\hat{\omega}_k^+)\hat{q}_k^+ && (4.99) \\ \hat{\beta}_{k+1}^- &= \hat{\beta}_k^+ && (4.100) \\ P_{k+1}^- &= \Phi_k P_k^+ \Phi_k^T + G_k Q_k G_k^T && (4.101) \end{aligned}$$

Chapter 5

Results and discussions

5.1 General comments

xNooch proved to be a much harder task than what was expected. As commented previously, the ridding of annoying and very disruptive bugs was the cause of a major setback that kept not just this writer, but a large part of the department scratching their heads for quite a while. The final solution came only just in time before the project time was over, and quite luckily I might add.

In the attempt of getting xNooch to work, its code was cleaned up of any source of disturbance or uncertainty. In removing uncertainty, we removed complexity, attempting to make the problem simpler to solve and find just where the threshold of its robustness lay. For that reason, the simulation ran for last and that I show in this document is not realistic for a space mission. First and foremost, we adopted a simple white noise that is generated by the same code that runs the filter. The error model, in this simulation, is not used. While white noise does not make the filter's job any easier, it would have been satisfactory to observe the performance with realistic mission data. Second, the telemetry that is expected currently will have an 88 second window of transmission, approximately each 92 minutes. Consequently, the filter is required to collect data while it can, and then predict and estimate for a time much longer than the duration of the previous transmission. Unfortunately, the development team has not yet completed an entire batch of "truth" data past 88 seconds. Thus, the simulation shown in this report documents 88 seconds of flight.

The data that is analyzed is still quite challenging, and not far from realistic. Even though the noise is not modeled, the spacecraft dynamics are, and so is the trajectory that is being followed and the spin motion applied to it. This means the values of the ECI vectors are appropriate, and in line with the signals the sensors are receiving. The sensors do thus follow the mission model, the universe model, and the spacecraft model as well. The filter is tuned to the sensitivity of the sensors, while the parameters for what we could call "process noise", which is whichever random disturbance our flight path may encounter, are not known. One of the aids we were attempting to give the filter, a magical sun sensor which may see the sun at all times (even though the s/c is spinning) has not been removed, for it is part of another not accessible simulation. In more technical terms, the sun vector in body frame coordinates is always available and always known. The filter is obviously devised to cope with sun sensor data switching on and off, but has not been tested for it, for all the reasons explained above.

5.2 The graphs

The results shown below apply the filter we derived, in its entirety, and 88 seconds worth of data provided by a simulation built by other team members. That simulation provides a simple model of the universe (relative positions of the main celestial bodies), the satellite (its structure and sensor placement), and the mission (1 Hertz spin frequency, along with orbit trajectory), with all of the related physics, dynamics and disturbances. It does not model sensor noise, for the reasons explained in the previous section.

Therefore, sensor noise has been added following the parameters here described: the standard deviation σ_ν of the random measurement noise acting on the rate sensors is 0.01 rad/sec^2. The standard deviation of the bias drift acting on the rate sensors is $\sigma_u = 0.01 rad^3/sec^2$. For the sun sensor, the standard deviation is $\sigma_{sun} = 0.04 rad$, and $\sigma_{mag} = 0.2 tesla$ for the magnetometer.

The following graphs plot the result of xNooch. Figure 5.1 shows the

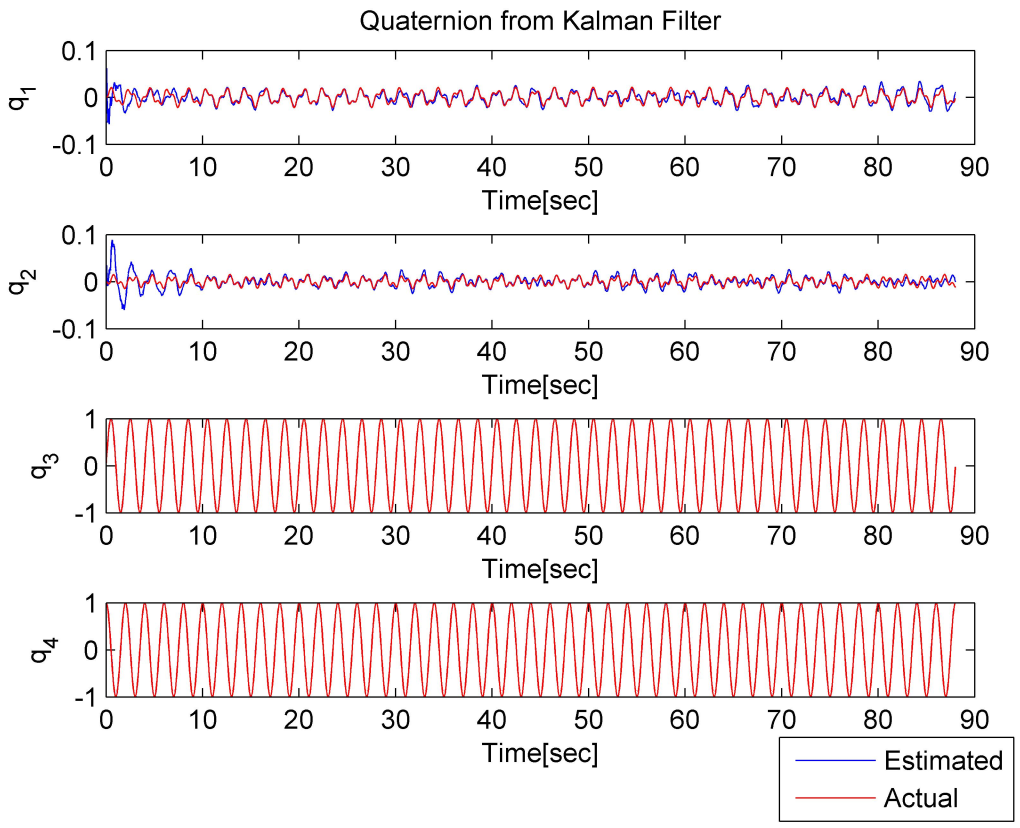

Figure 5.1: The quaternion graphs of an 88 seconds simulation

estimation of the quaternion parameters in blue, against the red line indicat-

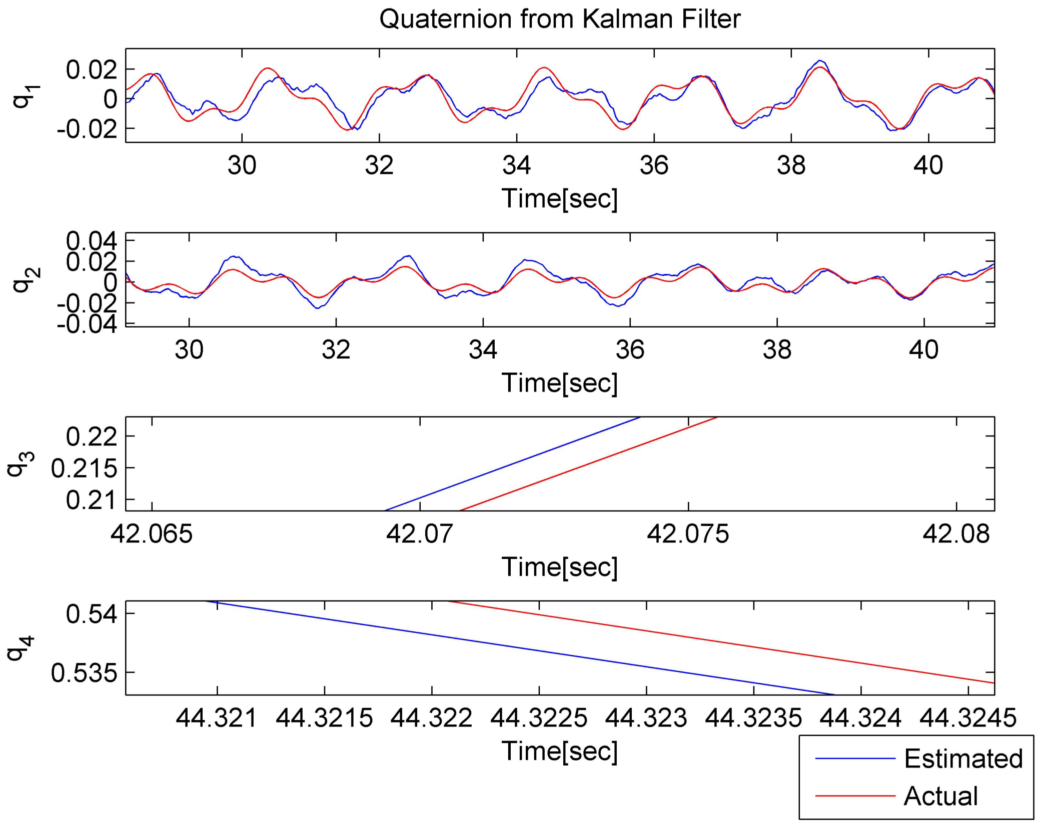

Figure 5.2: A close up of the quaternion graphs

ing the truth value. The entire 88 seconds show an extremely close match of the values, and so does the close up in figure 5.2. The values match almost identically. The estimation does describe, very closely, the true state of the simulation. The only severe bias to the estimation is the possibility to scan the position of the sun at all times, and not just at 1 Hertz short intervals. The average quaternion error values are ($q_1 = 7.1842e-004$, $q_2 = -3.3155e-004$, $q_3 = 0.0011$, $q_4 = 0.9997$)[1]. Similar conclusions may be

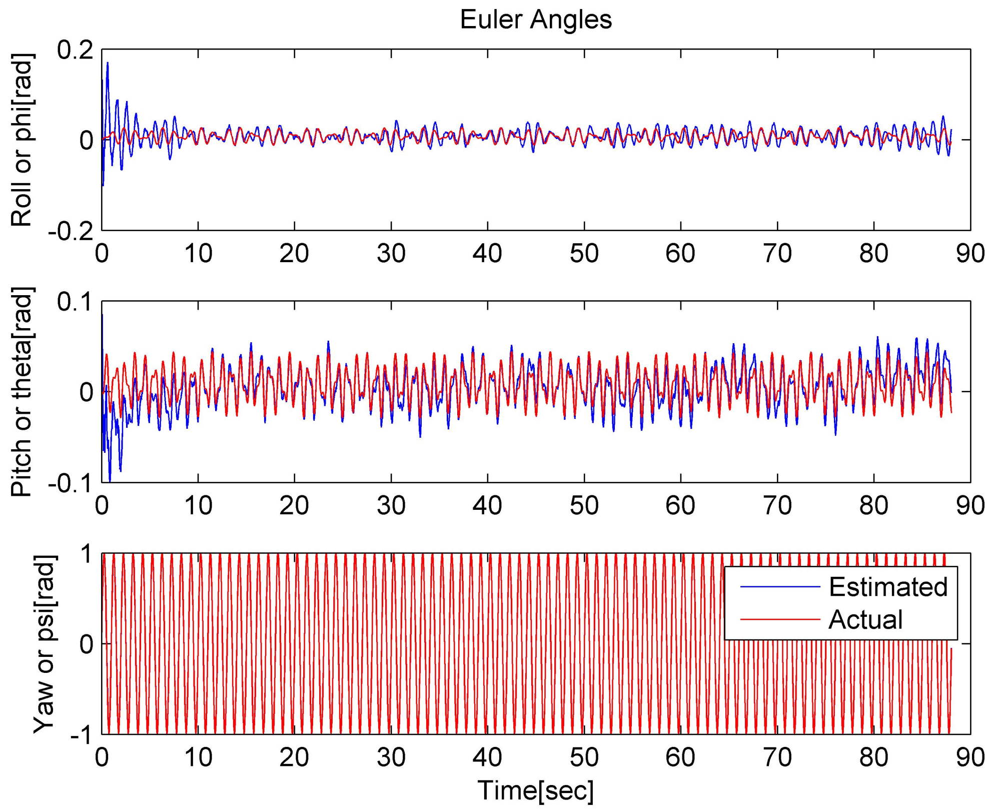

Figure 5.3: The Euler angles calculated from the simulated data

drawn by looking at the Euler angle graphs, 5.3 and 5.4 both in full size and zoomed. Here, especially for the spin axis, it was quite challenging to find

[1]these values clearly cannot form a quaternion themselves.

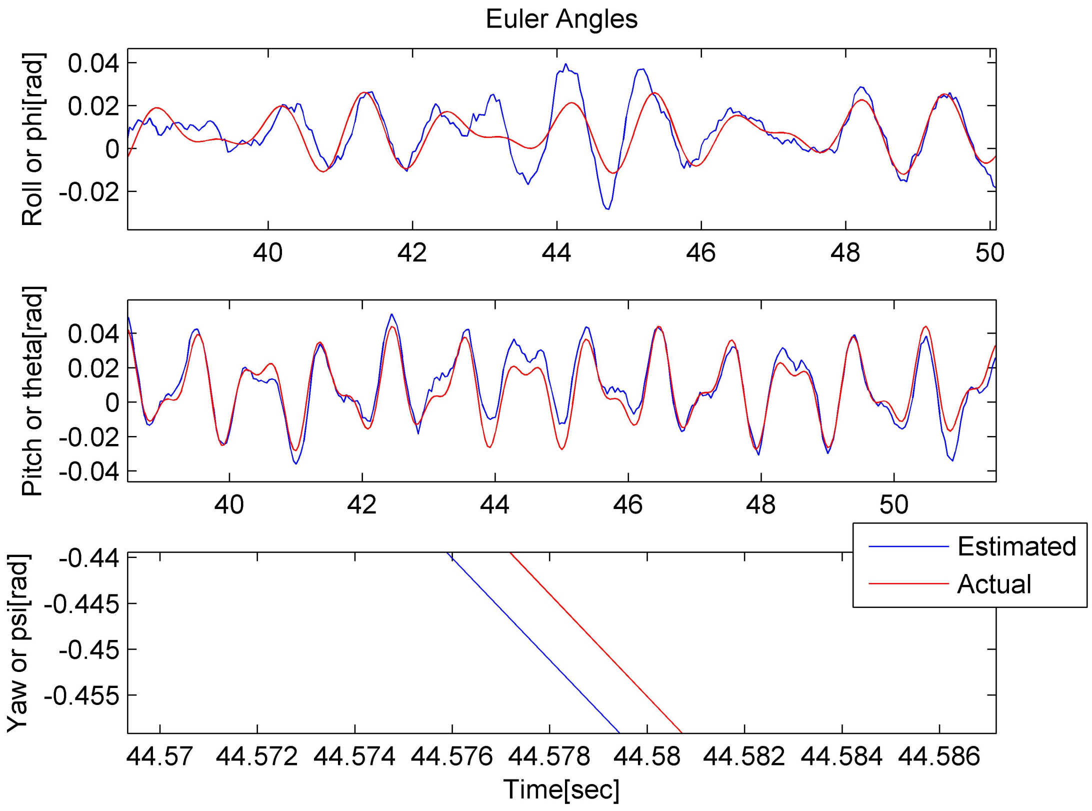

Figure 5.4: A closer look at the Euler angles graphs

the estimation to be at all different from the truth. The average error in ϕ is 0.0013 radians, while it is 0.004 in θ and 3.7192×10^{-4} in ψ. It is extremely accurate, as these numbers, graphs and the scale of the zoomed version show. It sounds quite natural that the spin axis is the one that is estimated the best, for it is the most stable one. This is surely evidence of the effectiveness of spin stabilization. Figure 5.5 shows for a short instant the matching

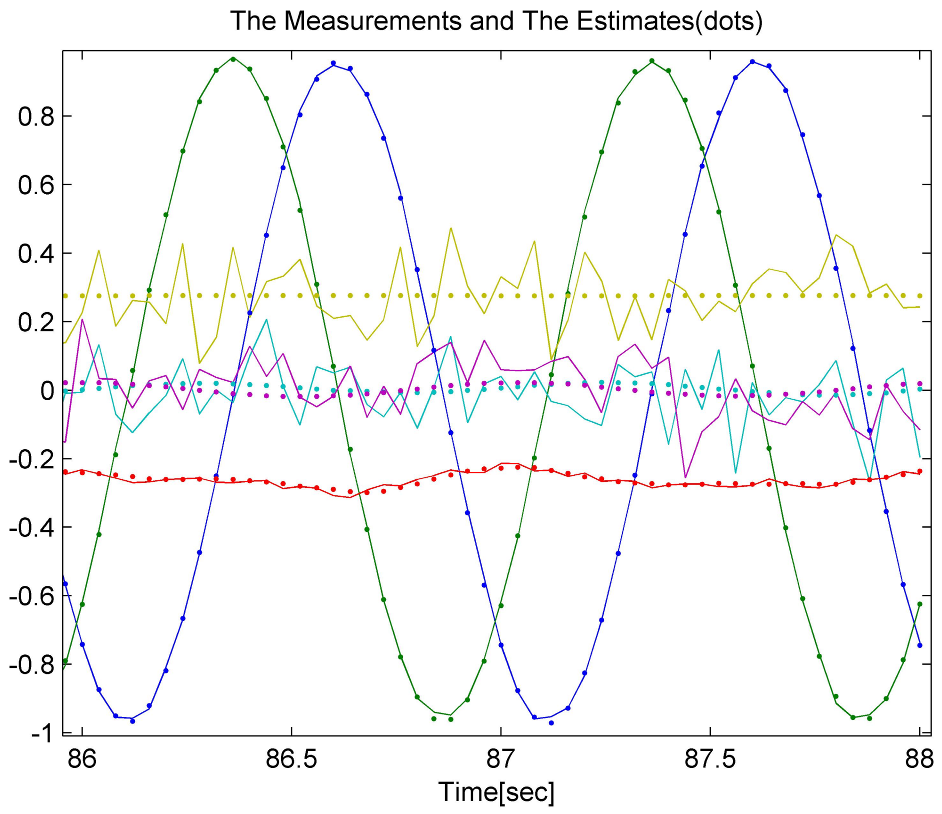

Figure 5.5: The measured data versus the estimated prediction of the filter

of measured data, and predicted data. The difference between these two measures is called the residual. The graph shows how the prediction itself is quite good, and much smoother than the noisy measurements. Attempts reaching up to 100x the standard deviations used here have shown results still vaguely acceptable. Estimation was poor, but not worthless. Figures

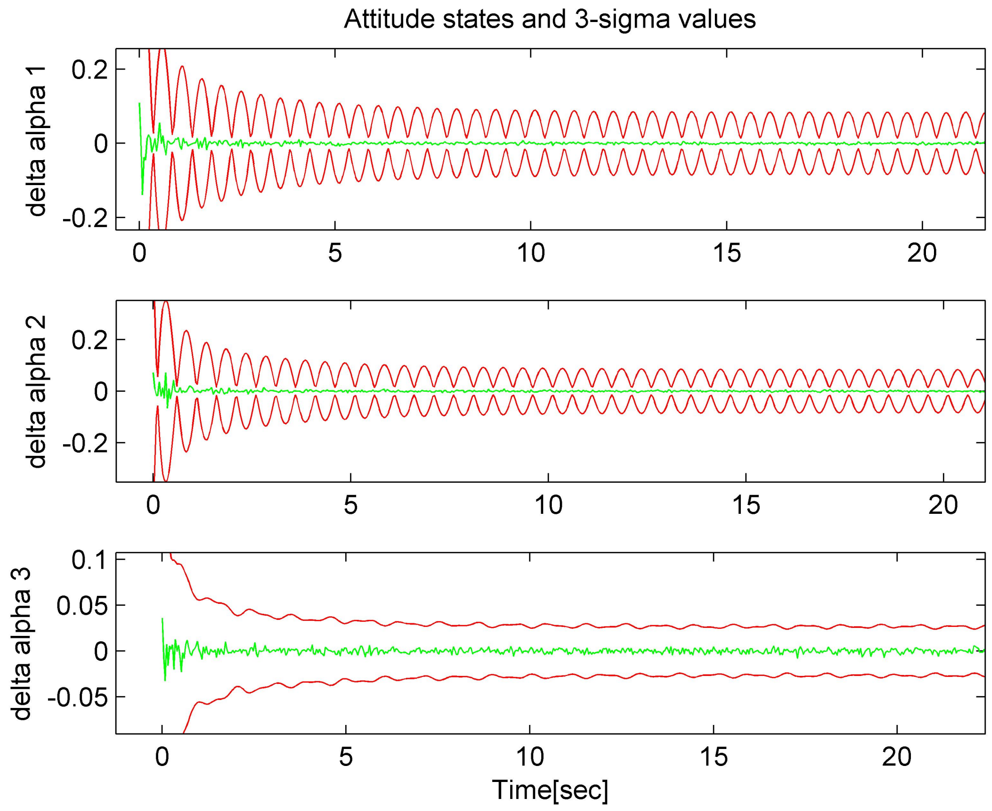

Figure 5.6: The first three estimated states and their $3 - \sigma$ values

5.6 and 5.7 show the oscillations in the actual values of the states, in the filter's inner workings, and the respective 3-sigma values. The graphs show not only how the actual value of the state falls quite well in the predicted 3-sigma path, but also how each measurement greatly contributes to shrink the expected standard deviation of our states. These graphs, especially the three pertaining to the $\delta\alpha$ estimation, are perhaps the best indication that the filter is working correctly. The $\delta\beta$s do not contribute much, for the values converge quite quickly and stay constant, as they are designed to be in these simulations. Figure 5.8 shows the α angle error. It is an indicator of

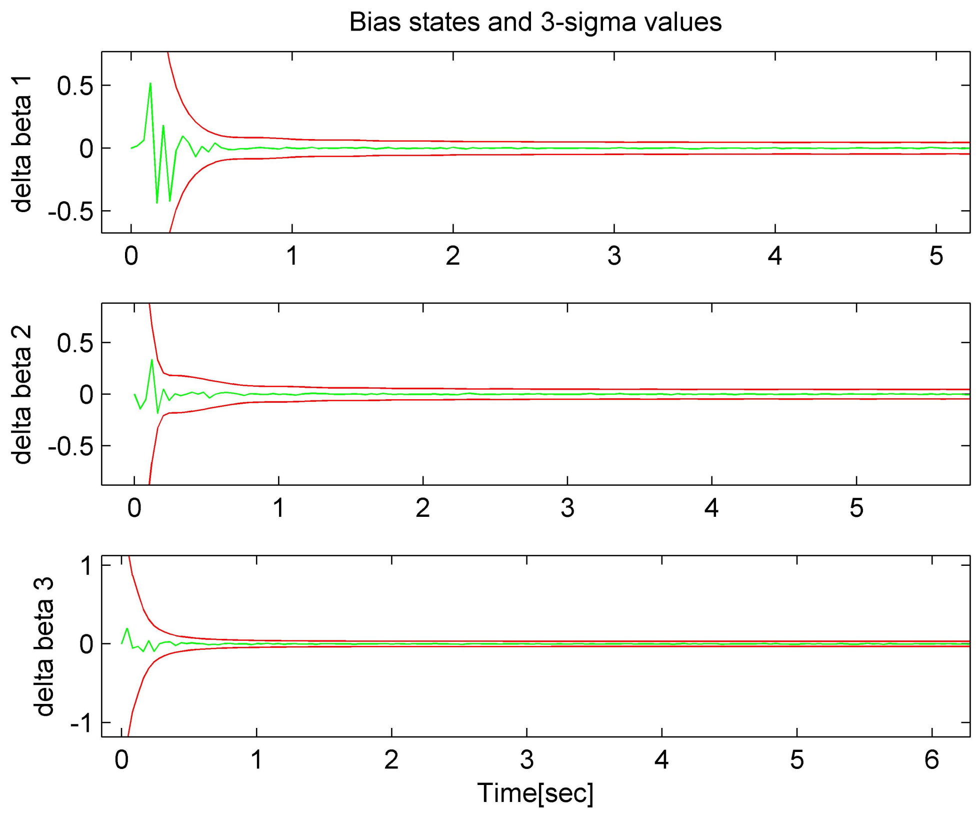

Figure 5.7: The second three states, and the $3-\sigma$ values

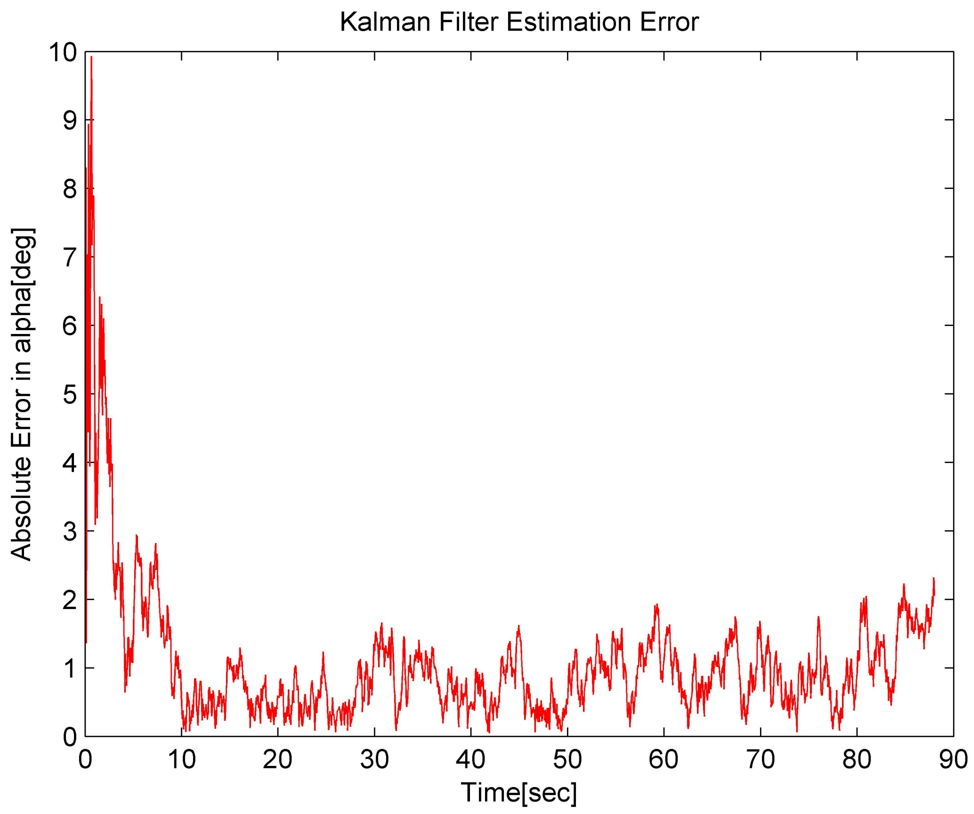

Figure 5.8: The error in the angle α

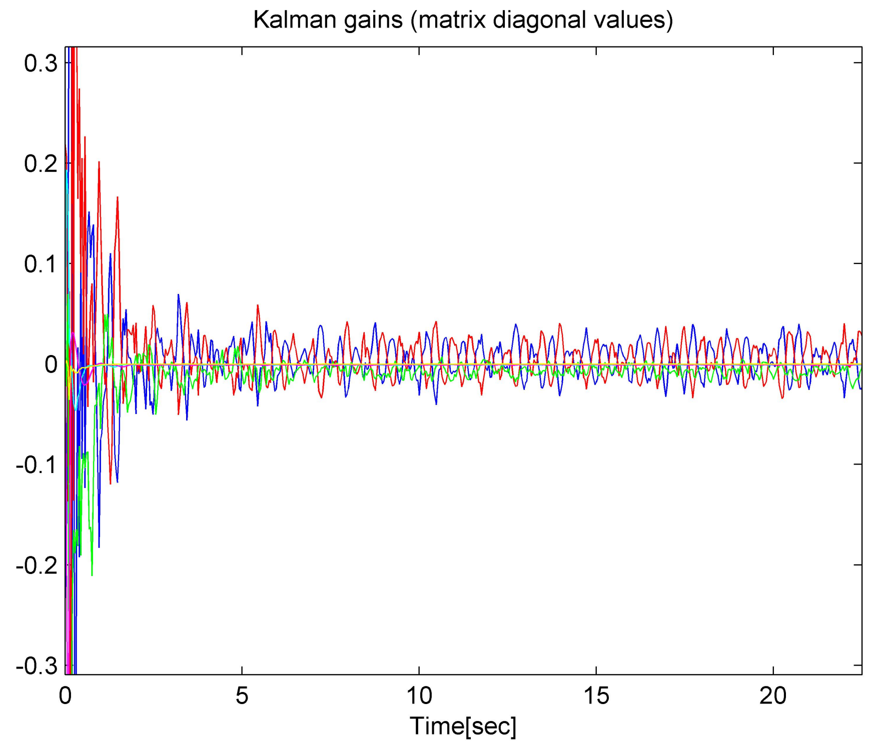

Figure 5.9: A close up of the Kalman gain matrix diagonal values

how accurate we are estimating. It is the difference in the absolute value of the estimated α and the true. As a reminder, α is the angle that drives the quaternion. The average alpha angle error is 1.1795 deg. As we have said, we are introducing little error in our system. Nevertheless, such a small error reveals that the filter works, and it works quite well indeed. The last figure is posted just to show the peculiarity of the filter's reaction to a spinning satellite. This filter could obviously work with a non-spinning satellite [2], but it adapts to the spin motion by fluctuating its Kalman gain, which has in fact an oscillatory behavior. It appeared surprising at first, but quite obvious after a brief observation of equation 4.92 on page 39.

[2]but not with a tumbling one!

5.3 Conclusions and Further Work

These graphs show a simulation with quite favorable conditions. Nevertheless, they demonstrate the behavior of the filter and show its working state. The filter is adaptable to a vast range of applications and satellites, and the cure in its design and coding make it suitable to a teaching environment as well. It has satisfied the development team at USU, and for the way it is designed it can aid in both pre-mission design, simulation, on board attitude estimation, and post-mission attitude accuracy estimation.

Looking back to the objective of xNooch, it is though only partly fulfilled. While the filter does record the error covariance matrices at each step, and therefore allows to know what is the accuracy of the current estimation, this filter does not take advantage of batch estimation. Even though time was limited, the smoothing option was investigated, and found an immediate halt when analyzed more in depth. As explained earlier, the smoothing principle is based on the combination of two estimates, suitably merged. *This particular filter adapts poorly to such an approach, for the states calculated by it are variations in non-linear physical parameters. In principle, the variation $\delta\alpha$ (or $\delta\beta$) from step* k *to step* k+1 *has little to do with the corresponding variation estimated by the so called "backward filter", from step* k+1 *to* k*, or any other step for that matter. A quickly discarded idea has been to use the negative of the backward variation, but that approach does not seem to be a correct solution, for the states we have chosen are rather non-linear in nature.* The case requires therefore more investigation, as this seems to be a non-trivial problem. This work thus allows for further research, and may be a module of a larger project.

In conclusion, the xNooch set out to be an optimal batch estimator, but settled for being an effective optimal attitude Kalman filter, designed for the PEARL cubesat but adaptable to other applications.

Bibliography

[1] Crassidis, J.L., Junkins, J.L.: *Optimal Estimation of Dynamic Systems*, Chapman & Hall/CRC (2004)

[2] Paul Coddington, *Monte Carlo Simulation for Statistical Physics* Northeast Parallel Architectures Center at Syracuse University

[3] Gelb, A.: *Applied Optimal Estimation*, Analytic Science Corporation Technical Staff, (1974)

[4] Geller, D.K: *Linear Covariance Techniques for Orbital Rendezvous Analysis and Autonomous Onboard Mission Planning*, *Journal of Guidance, Control and Dynamics*, American Institute of Aeronautics and Astronautics, (2006)

[5] Sidi, M.J.: *Spacecraft Dynamics and Control*, Cambridge, (1997)

[6] Geller, D.K.: *lecture notes from the 'Space Navigation' course, held at USU*, April 11th 2008

[7] Rauch, H.E., Tung, F., and Striebel, C.T.: *Maximum likelihood Estimates of Linear Dynamic Systems*, AIAA Journal, Vol. 3, No. 8, August 1965

[8] Fullmer, R.: *lecture notes from the 'Attitude Control' course, held at USU*, March 2008