NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN EXTENDED KALMAN FILTER FOR QUATERNION-BASED ATTITUDE ESTIMATION

by

João L. Marins

September 2000

Chaiman of Committee and Supervisor:	Xiaoping Yun
Committee Member:	Eric R. Bachmann
Committee Member:	Robert G. Hutchins

Approved for public release; distribution is unlimited.

20001130 055

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 2000	3. REPORT TYPE AND DATES COVERED Engineer's Thesis

4. TITLE AND SUBTITLE:
An Extended Kalman Filter For Quaternion-Based Attitude Estimation

5. FUNDING NUMBERS

6. AUTHOR(S)
Marins, João Luís.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. PONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed here are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(maximum 200 words)***

This thesis develops an extended Kalman filter for real-time estimation of rigid body motion attitude. The filter represents rotations using quaternions rather than Euler angles, which eliminates the long-standing problem of singularities associated with those angles. A process model for rigid body angular motions and angular rate measurements is defined. The process model converts angular rates into quaternion rates, which are in turn integrated to obtain quaternions. The outputs of the model are values of three-dimensional angular rates, three-dimensional linear accelerations, and three-dimensional magnetic field vector. Gauss-Newton iteration is utilized to find the best quaternion that relates the measured linear accelerations and earth magnetic field in the body coordinate frame to calculated values in the earth coordinate frame. The quaternion obtained from the optimization algorithm is used as part of the observations for the Kalman filter. As a result, the measurement equations become linear.

A new approach to attitude estimation is introduced in this thesis. The computational requirements related to the extended Kalman filter developed using this approach are significantly reduced, making it possible to estimate attitude in real-time. Extensive static and dynamic simulation of the filter using Matlab proved it to be robust. Test cases included the presence of large initial errors as well as high noise levels. In all cases the filter was able to converge and accurately track attitude.

14. SUBJECT TERMS
Inertial Navigation, Extended Kalman Filter, Quaternion

15. NUMBER OF PAGES 114

16. PRICE CODE

17.SECURITY CLASSIFICATION OF REPORT	18.SECURITYCLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AN EXTENDED KALMAN FILTER FOR QUATERNION-BASED ATTITUDE ESTIMATION

João Luís Marins
Lieutenant Commander, Brazilian Navy
B.S., Escola Naval (Brazil), December 1985
B.S.E.E., Universidade de São Paulo (Brazil), December 1991

Submitted in partial fulfillment of the
requirements for the degrees of

ELECTRICAL ENGINEER
and
MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author:

João Luís Marins

Approved by:

Xiaoping Yun, Chairman of Committee and Supervisor

Eric R. Bachmann, Committee Member

Robert G. Hutchins, Committee Member

Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis develops an extended Kalman filter for real-time estimation of rigid body motion attitude. The filter represents rotations using quaternions rather than Euler angles, which eliminates the long-standing problem of singularities associated with those angles. A process model for rigid body angular motions and angular rate measurements is defined. The process model converts angular rates into quaternion rates, which are in turn integrated to obtain quaternions. The outputs of the model are values of three-dimensional angular rates, three-dimensional linear accelerations, and three-dimensional magnetic field vector. Gauss-Newton iteration is utilized to find the best quaternion that relates the measured linear accelerations and earth magnetic field in the body coordinate frame to calculated values in the earth coordinate frame. The quaternion obtained from the optimization algorithm is used as part of the observations for the Kalman filter. As a result, the measurement equations become linear.

A new approach to attitude estimation is introduced in this thesis. The computational requirements related to the extended Kalman filter developed using this approach are significantly reduced, making it possible to estimate attitude in real-time. Extensive static and dynamic simulation of the filter using Matlab proved it to be robust. Test cases included the presence of large initial errors as well as high noise levels. In all cases the filter was able to converge and accurately track attitude.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
- A. MOTIVATION 1
- B. RESEARCH QUESTIONS 3
- C. THESIS OUTLINE 4

II. BACKGROUND 7
- A. INTRODUCTION 7
- B. ROTATING FRAMES 7
  - 1. Frames 7
    - a. Rotation Matrices 8
    - b. Euler Angles 8
    - c. Axis and Angle Rotation 10
  - 2. Quaternions 10
    - a. Motivation 10
    - b. Quaternions 11
    - c. Quaternions Representing Rotations 13
- C. THE KALMAN FILTER 16
  - 1. Basics 16
  - 2. The Kalman Filter 17
  - 3. The Filter Structure 17
  - 4. The Discrete Kalman Filter Algorithm 18

III. THE MODEL 21
- A. INTRODUCTION 21
- B. THE PROCESS MODEL 21
  - 1. Defining the State and Output Variables 22
  - 2. The Process Model 23
  - 3. Relating Angular Rates and Quaternions 24

IV. THE KALMAN FILTER DESIGN 29
- A. OVERVIEW 29
- B. FIRST APPROACH 29

C. ALGORITHMS FOR QUATERNION CONVERGENCE..... 32
1. Stating the Problem..... 32
2. Minimizing the Error..... 33
D. SECOND APPROACH..... 36

V. MODELING SYSTEM NOISE..... 39
A. INTRODUCTION..... 39
B. PROCESS NOISE AND TIME CONSTANTS..... 40
C. MEASUREMENT NOISE..... 43
D. COVARIANCE MATRIX OF THE MEASUREMENT..... 46

VI. TESTS AND SIMULATIONS..... 49
A. INTRODUCTION..... 49
B. TESTS OF THE CONVERGENCE ALGORITHM..... 49
C. STATIC TESTS OF THE COMPLETE FILTER..... 53
1. Static Test 1 54
2. Static Test 2 56
3. Static Test 3..... 59
D. DYNAMIC TESTS OF THE COMPLETE FILTER..... 61
1. Generating Data..... 61
2. Dynamic Test 1..... 62
3. Dynamic Test 2..... 65

VII. CONCLUSIONS..... 69
A. SUMMARY..... 69
B. ANALYSIS OF RESULTS..... 70
C. SUGGESTIONS FOR FUTURE WORK..... 70

APPENDIX A – ROUTINES FOR CONVERGENCE TESTING 73
APPENDIX B – EXTENDED KALMAN FILTER..... 77
APPENDIX C – DATA GENERATION FOR SIMULATION..... 87
LIST OF REFERENCES..... 91
INITIAL DISTRIBUTION LIST..... 93

LIST OF FIGURES

Figure 2.1. Rotation Using Quaternions. 15

Figure 2.2. Kalman Loop. From [ref. 9]. 20

Figure 3.1. Process Model for Angular Rates. 23

Figure 3.2. Determining the Quaternion Rates. 25

Figure 3.3. Process Model for Angular Rates and Quaternions. 27

Figure 4.5. Final Diagram for the Filter. 36

Figure 5.1. Simulation of the Process Model for *p*, *q*, and *r*. 41

Figure 5.2. Comparison Between Real and Modeled Processes. 42

Figure 6.1. Quaternion Convergence Using Gauss-Newton (no Noise). 50

Figure 6.2. Quaternion Convergence Using Newton (no Noise). 51

Figure 6.3. Quaternion Convergence Using Gauss-Newton (Measurement Noise). .. 52

Figure 6.4. Static Test 1 - Two Coincident Frames. 54

Figure 6.5. Static Test 1 - Convergence of Angular Rates. 55

Figure 6.6. Static Test 1 – Convergence of Quaternion Components. 56

Figure 6.7. Static Test 2 – Rotated Frame. 57

Figure 6.8. Static Test 2 - Convergence of Angular Rates. 58

Figure 6.9. Static Test 2 – Convergence of Quaternion Components. 58

Figure 6.10. Static Test 3 – Frame Rotated About One of Its Axis. 59

Figure 6.11. Static Test 3 - Convergence of Angular Rates. 60

Figure 6.12. Static Test 3 – Convergence of Quaternion Components. 60

Figure 6.13. Data Generation for Dynamic Tests. 62

Figure 6.14. Dynamic Test 1 – Angular Rates Profile. 63

Figure 6.15. Dynamic Test 1 – Final Position After Rotation. 63

Figure 6.16. Dynamic Test 1 – Convergence of Angular Rates. 64

Figure 6.17. Dynamic Test 1 – Convergence of Quaternion Components. 65

Figure 6.18. Dynamic Test 2 – Angular Rates Profile. 66

Figure 6.19. Dynamic Test 2 – Final Position After Rotations. 66

Figure 6.20. Dynamic Test 2 – Convergence of Angular Rates. 67

Figure 6.21. Dynamic Test 2 – Convergence of Quaternion Components. 67

LIST OF TABLES

Table 5.1. Values of Variance and Time Constant for the Process Model. 41

Table 5.2. Values of Variance for the Measurement Noise. 44

Table 5.3. Values of Variances for Angular Rates and Acceleration. 44

Table 6.1. Experiment to Verify the Convergence Using Gauss-Newton Method. .. 53

Table 6.2. Static Test 1 – Initial and Steady-state Values. 55

Table 6.3. Static Test 2 – Initial and Steady-state Values. 57

Table 6.4. Static Test 3 – Initial and Steady-state Values. 59

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

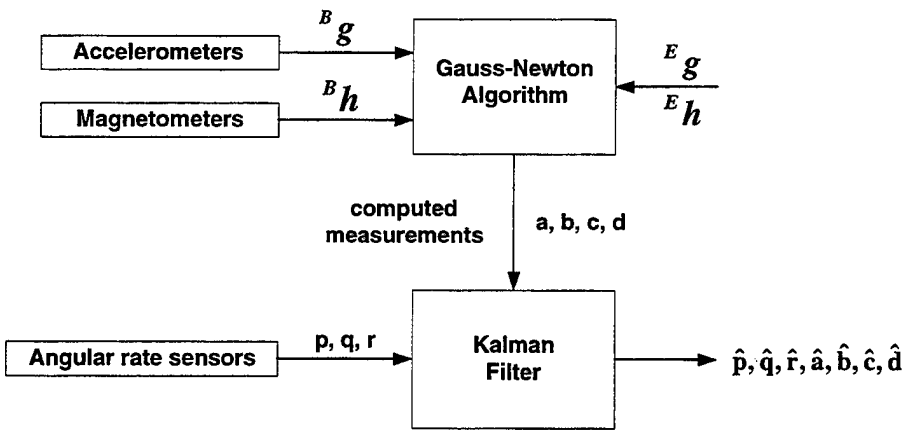
This thesis presented a complete design of an Extended Kalman filter for real-time estimation of rigid body motion attitude. The use of quaternions to represent rotations, instead of Euler angles, eliminated the long-standing problem of singularities called "gimbal lock".

A simple first-order process model for rigid body angular rate measurements was defined, which closely matched real process data obtained from angular rate sensors.

The quaternion rates were nonlinearly related to the current quaternion and the angular rates. The process model converted angular rates into quaternion rates, which were integrated to obtain quaternions.

Two approaches to the Kalman filter design were investigated. The first approach used nine output equations: three angular rates, three components of linear acceleration, and three components of the earth magnetic field. Since these output equations were highly nonlinear functions of the process state variables, the partial derivatives needed for the Kalman filter design were very complicated. As a result, the Kalman filter was not feasible for real-time implementation.

The second approach utilized Gauss-Newton algorithm to find the optimal quaternion that related the values of linear accelerations and earth magnetic field in the body coordinate frame and in the earth coordinate frame. The optimal quaternion was used as part of the measurement for the Kalman filter, which had seven output equations: three angular rates and four components of quaternions. Furthermore, the output equations were linear and the output matrix was the identity matrix. As a result, the partial derivatives were total derivatives and had constant values. The convergence algorithm replaced the computation of the partial derivatives of the nine nonlinear measurement equations. The following diagram represents the approach.



The computational requirements for the Kalman filter developed using this approach were significantly reduced, making it possible to estimate attitude in real time.

Extensive static and dynamic simulation of the filter using Matlab proved it to be robust. Test cases included the presence of large initial errors as well as high noise levels. In all cases the filter was able to converge and accurately track attitude.

ACKNOWLEDGMENTS

I would like to thank Dr. Xiaoping Yun, who gave me the opportunity to work on this difficult and interesting topic. His assistance and patience enabled me to present difficult concepts in this thesis in a meaningful manner.

I wish to express my sincere appreciation to Prof. Eric Bachman for his advice and suggestions during the process of writing and presenting my thesis. I know how important time is for a doctor's degree candidate.

I express my gratitude to Dr. Hutchins, the first person who taught me how quaternions work. In his class I developed my first Extended Kalman filter and learned all the background needed for this thesis.

I specially thank Dr. Robert McGhee for the original vision of the quaternion convergence. His idea is the core of this thesis and led us all to great discoveries.

Finally, I would like to thank my wife, Nice, for her love, support, patience and faith throughout this time. I would also like to thank my children, André and Amanda, whose love and very presence were a continual source of happiness and strength.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Navigation is the process of computing the position, orientation and speed of a body or vehicle. When this computation is based solely on inputs from self-contained inertial sensing instruments it is called inertial navigation. Angular rate sensors and accelerometers are typical inertial sensors.

Inertial navigation systems integrate the output of accelerometers to obtain velocity, and integrate velocity to obtain distance traveled. In order to accomplish this, the system should know the orientation of the accelerometer axes relative to a fixed reference frame. Whether this orientation is fixed or not, leads to two different types of mechanizations for inertial systems. These mechanizations establish how the sensors are attached to the system.

In the first type, the inertial sensors are mounted on a stable platform, which is connected to gimbals. Torque motors receive information from the angular rate sensors and drive the gimbals in order to keep the platform leveled and headed in a fixed direction (usually north, east, and down - NED). The position is calculated using the output of accelerometers directly. This kind of mechanization is called "gimbaled".

Alternatively, the inertial sensors can be fixed to the vehicle. This mechanization is called "strapdown". In this case, the accelerations are measured with respect to body-fixed axes and they need to be converted into integration axes, normally earth-fixed axes.

Thus, it is important to keep track of the orientation of the body with respect to the earth coordinates.

Orientation can be defined as a set of parameters that relates the angular position of a frame to another reference frame. There are numerous methods for describing this relation. Among them, rotation matrices, Euler angles and quaternions are commonly used. In "strapdown" systems, they must be calculated at each step of the integration, so the coordinate transformation between frames can be carried out. Orientation or attitude tracking systems perform this task. They are used in navigation systems, mobile robots, head tracking and other applications.

Attitude tracking systems might be implemented in different forms. [Ref. 1] presents a solution using GPS and inertial sensors used for aircraft. The difference among the GPS signals received by three antennas gives attitude information. [Ref. 2] replaces the antenna information with information coming from celestial observations. [Ref 3] describes an attitude package, which combines the outputs of inclinometers, gyros, and compasses to obtain attitude estimation. All three examples utilize Euler angles to represent orientation and a Kalman filtering algorithm to integrate the information.

The system in [Ref. 4] has a different approach. It uses what is called Magnetic Angular Rate Gravity (MARG) sensors. Each MARG sensor consists of three orthogonal rate sensors, three orthogonal accelerometers, and three orthogonal magnetometers mounted in a "strapdown" configuration. Quaternions represent the orientation between frames. The use of quaternions avoids the singularity problems characteristic of filters that use Euler angles. Integration of angular rate outputs provides an angular orientation,

which is corrected with information coming from the accelerometers and magnetometers. A complementary filter estimates the attitude of a rigid body that the sensor is attached to. Complementary filters are used when the system deals with redundant measurement of same quantity corrupted with noises separated in frequency (low-frequency and high-frequency). They have fixed gains and normally are designed in the frequency-domain.

This thesis intends to follow the same approach described above, but replaces the complementary filter with a Kalman filter. The Kalman filter blends information from the sensors to provide an optimal estimate from all sources combined. Instead of fixed gains, the gains of an Kalman filter are adjusted at each step.

The use of quaternions is maintained because of characteristics that make them very suitable for this approach. They can easily be transformed into a matrix and can be integrated easily due to their dependence on the angular rates. They are quadratic functions, which guarantees good convergence properties. These characteristics may be used to reduce the order of the output vector as well as the computational effort needed to run the filter.

B. RESEARCH QUESTIONS

This thesis will examine the following research topics:

- Evaluate the use of quaternions for representing attitude in conjunction with attitude filter design;
- Design a Kalman Filter that estimates attitude based on information provided by accelerometers, angular rates, and magnetometers;

- Verify the viability of using the quadratic properties of quaternions to reduce the filter complexity and computational requirements; and
- Test the filter performance using simulated data.

C. THESIS OUTLINE

The purpose of this thesis is to present a complete study and implementation in Matlab of a Kalman Filter that estimates the attitude of a body with respect to a fixed reference frame. The parameters that represent attitude are quaternions. The thesis is organized as follows:

- Chapter II reviews the concepts of frames and compares several methods for representing frame rotations. It introduces the notion of quaternions and shows how they can represent rotations. It also reviews the basics of Kalman Filters.
- Chapter III presents the process model used to develop the Kalman filter. As the quaternion rate depends on the values of angular rates, the model utilized in this thesis is based on the relationship between angular rates and quaternions.
- Chapter IV is the core of this thesis. It starts with the usual formulation of the Kalman Filter. Subsequently, a more efficient method is presented, which reduces the order of the output vector and results in a linear state-output relationship. The link between these two approaches is the convergence

properties of quaternions. Solutions based on the Gauss and Gauss-Newton convergence algorithms are presented.

- Chapter V describes the procedure applied to obtain the statistical properties of the system.
- Chapter VI presents tests of the convergence properties and describes the filter simulation. It explains how simulated data were created and discusses the results of applying that data to the filter. Static and dynamic tests were used to evaluate the filter performance.
- Chapter VII presents a short summary along with an analysis of results and conclusions. It ends with suggestions for the future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. INTRODUCTION

There are several conventions for describing orientation and rotation in 3D space. Some are easier to visualize than others are. Each has some kind of limitation. This chapter presents the notion of a frame and gives an overview on three well-known orientation or rotation methods. Quaternions are introduced as an alternative method.

It also presents the basics of the Kalman filter by discussing the formulation and structure of the discrete Kalman filter, which will be used in this thesis.

B. ROTATING FRAMES

1. Frames

In order to describe the orientation of a body, a coordinate system is attached to the body and then it is referenced to another fixed coordinate system. If the Cartesian coordinate system is adopted, a set of three orthonormal vectors is used. This set of vectors is called a frame.

Two frames are used in this thesis. The earth frame is fixed and has its x, y, and z-axes pointing North, East, and Down respectively. The body frame moves with the body being tracked and has its x, y, and z-axes parallel to the main axes of the body. The body axes point respectively to the front, right, and down.

Any vector can be represented in a frame by three components. These components are the projections of the vector along each of the frame axes. Given the components of a vector in one frame, the components in another frame can be calculated provided that the relationship between the frames is known.

a. Rotation Matrices

Let ${}^{B}\vec{v}$ and ${}^{E}\vec{v}$ be 3 x 1 vectors containing the components of a reference vector in frames *B* and *E*. There exists a 3 x 3 matrix *R* such that

$$ {}^{E}\vec{v} = R\ {}^{B}\vec{v}, \tag{2.1} $$

where each column of *R* is the projection of each one of the frame-*B* axes on the frame-*E* axes.

Rotation matrices must be orthogonal (both frames with orthogonal axes) and orthonormal (all axes are orthonormal). Thus, although the matrix has nine elements these constraints reduce the number of degrees-of-freedom (DOF) to three.

Using rotation matrices has some drawbacks [Ref. 5]. Violation of the above constraints during the execution of the algorithm may make it difficult to keep the matrix orthonormalized. Another problem is that it is very hard to interpolate rotations between two orientations.

b. Euler Angles

Suppose that a sequence of rotations is applied to a frame *B* about its own axes. The Euler Theorem states that three rotations are needed (no two rotations about the

same axis) to make the frame B coincide with another frame E [Ref. 6]. The three angles are called Euler Angles.

It is worth noting that each rotation is about the new coordinate frame obtained after the previous rotation. For example, a rotation sequence xyz means a rotation about x-axis, followed by a rotation about the new y-axis, followed by a rotation about the newer z-axis.

There is one rotation matrix that represents each rotation described above. This compound rotation is the product of those three matrices and it is the same rotation matrix mentioned in the previous section.

This method of describing orientation is efficient because it uses only three angles to represent three degrees-of-freedom. Euler angles are not constrained, so they don't have to be adjusted (see quaternion normalization). However, although only three angles represent the orientation, it may be necessary to construct a rotation matrix for each angle before performing a rotation.

Other problems arise when using Euler angles [Ref. 5]. It is difficult to find a series of rotations that represent a single rotation. Furthermore, Euler angles have a singularity problem named "gimbal lock". It happens when the coordinate frame is rotated 90 degrees consecutively.

For instance, suppose a frame with three orthonormal axes is rotated 90 degrees about the x-axis and 90 degrees about the new y-axis. The old x axis and the newer z axis are now aligned and one degree-of-freedom is lost. In this case, the rotations about the x and z axes are no longer unique and only their sum or difference can be described.

c. *Axis and Angle Rotation*

This method is based on the concept of the Euler axis of rotation. The Euler axis is defined as the axis of rotation about which one coordinate frame can be rotated into another. In other words, there exists a pair composed of an axis and an angle $({}^{B}\bar{v}, \theta)$ such that if a frame *B* is rotated about this axis through an angle θ the final position of frame *B* coincides with the position of frame *E*. Note that the axis is represented in the original frame *B*.

This method also has the problems cited in the previous sections with the exception of "gimbal lock". It is a step in the direction of another way to represent rotations, namely, quaternions.

2. Quaternions

The attitude reference quaternion is a different way of representing the Euler axis and angle. It uses a set of four parameters: three represent the components of a vector directed along the Euler axis; the fourth is a scalar quantity.

The discussion to follow describes how quaternions are used as an alternative to Euler angles in strapdown attitude systems.

a. *Motivation*

Understanding how quaternions perform rotation is quite easy if the reasoning starts in two dimensions [Ref. 7]. Let *v* be a complex number

$$v = e\,i + h\,, \tag{2.2}$$

where i represents the imaginary number so that

$$i\,i = -1 \qquad (2.3).$$

The complex number v can be thought of as a vector with two components e and h in the complex plane.

Define another complex number $u = a\,i + d$ with components

$$\begin{aligned} a &= \sin\theta \\ d &= \cos\theta \end{aligned} \qquad (2.4).$$

The product of u and v is given by

$$w = u\,v = (e\cos\theta + h\sin\theta)\,i + (h\cos\theta - e\sin\theta) \qquad (2.5).$$

From the latter expression it is easy to notice that the product vector represents the vector v rotated by θ in the complex plane [Ref. 7].

b. Quaternions

In 1843, Hamilton invented the so-called hyper-complex number of rank 4, to which he gave the name quaternion. Quaternions are a four-dimensional extension to complex numbers. Based on Equation 2.3, Hamilton defined a similar rule for dealing with the operations on the vector part of the quaternion. He stated that

$$i^2 = j^2 = k^2 = i\,j\,k = -1 \qquad (2.6).$$

A quaternion can be regarded as an element of $\Re^4$. In this thesis, quaternions will be represented using the notation from [Ref. 7], which defines the quaternions as

$$n = a\,i + b\,j + c\,k + d\,, \qquad (2.7)$$

where a, b, c, d are real numbers and i, j, k are unit vectors directed along the x, y, and z axis respectively.

An alternative way of representing a quaternion is as the sum of a scalar and a 3-dimensional vector [Ref. 6]. The Equation 2.7 can be rewritten as follows

$$n = n_0 + \vec{n}\,, \tag{2.8}$$

where the vector

$$\vec{n} = (a,b,c) = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \quad and \quad n_0 = d = \begin{bmatrix} 0 \\ 0 \\ 0 \\ d \end{bmatrix} \tag{2.9}.$$

When the scalar part of a quaternion is zero the quaternion is called a pure quaternion.

The product of two quaternions is defined as follows.

Let n and m be two quaternions such that

$$n = a\,i + b\,j + c\,k + d = n_0 + \vec{n} \tag{2.10}$$

$$m = e\,i + f\,j + g\,k + h = m_0 + \vec{m} \tag{2.11}.$$

The quaternion product of n and m is defined so that the following special products are satisfied [Ref. 6]:

$$i^2 = j^2 = k^2 = i\,j\,k = -1 \tag{2.12}$$

$$i\,j = k = -j\,i \tag{2.13}$$

$$j\,k = i = -k\,j \tag{2.14}$$

$$k\,i = j = -i\,k \tag{2.15}.$$

Note that Equations 2.12 to 2.15 are the usual vector product for tri-orthogonal vectors. Then, applying all the above rules and the associative and commutative laws of scalar multiplication, the product of m and n is

$$\begin{aligned} n\,m = {} & d\,h-(a\,e+b\,f+c\,g)+ \\ & +d\,(e\,i+f\,j+g\,k)+h\,(a\,i+b\,j+c\,k)+ \\ & +(b\,g-c\,f)\,i+(c\,e-a\,g)\,j+(a\,f-b\,e)\,k \end{aligned} \qquad (2.16).$$

After separating the scalar and vector products, the above equation can be written

$$n\,m = n_0\,m_0 - \vec{n}\,\vec{m} + n_0\,m + m_0\,n + \vec{n}\times\vec{m}. \qquad (2.17).$$

Note that the vector cross product that appears in the last term makes the quaternion product, in general, non-commutative.

Another application for the notation "scalar plus vector" is the rotation using quaternions, which is focused on the next section.

c. *Quaternions Representing Rotations*

A quaternion

$$n = a\,i+b\,j+c\,k+d = n_0+\vec{n}, \qquad (2.18)$$

such that

$$n_0 = \cos\theta \qquad \text{and} \qquad |\vec{n}| = \sin\theta, \qquad (2.19)$$

can be used to rotate a vector $\vec{u}$. Note that these two constraints make n a unit quaternion.

In order to do that, $\vec{u}$ must be written as a pure quaternion

$$u = u_0+\vec{u} = 0+\vec{u} \qquad (2.20).$$

The rotation is performed through the double quaternion multiplication

$$n\,u\,n^*, \tag{2.21}$$

where n^* is the complex conjugate of the quaternion n defined as

$$n^* = -a\,i - b\,j - c\,k + d = n_0 - \vec{n} \tag{2.22}.$$

This operation rotates the vector $\vec{u}$ through an angle 2θ about the axis defined by $\vec{n}$ or, similarly, rotates the frame through an angle -2θ about the axis defined by $\vec{n}$.

The operation using Equation 2.21 is equivalent to a matrix multiplication

$$n\,u\,n^* = R\,\vec{u}, \tag{2.23}$$

where

$$R = \begin{bmatrix} d^2 + a^2 - b^2 - c^2 & 2(ab - cd) & 2(ac + bd) \\ 2(ab + cd) & d^2 + b^2 - a^2 - c^2 & 2(bc - ad) \\ 2(ac - bd) & 2(bc + ad) & d^2 + c^2 - b^2 - a^2 \end{bmatrix} \tag{2.24}.$$

In both cases it can be proved [Ref. 3] that the result is again a pure quaternion, which means it represents a vector. Furthermore, the final vector has the same length as the initial one.

As a further example, suppose that a vector $\vec{v}$ pointing in the x-axis direction is to be rotated 120 degrees about the axis defined by $(i, j, k) = (1, 1, 1)$.

First, $\vec{v}$ should be written as a pure quaternion v

$$v = (1, 0, 0, 0).$$

The unit vector representing the Euler axis is

$$\vec{u} = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right).$$

This unit vector and the rotation angle are used to calculate the unit quaternion and its complex conjugate

$$n = \left(\sin 60° \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}} \right), \cos 60° \right) = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2} \right)$$

$$n^* = \left(-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, \frac{1}{2} \right).$$

Finally, the vector $\vec{v}$ is rotated using Equation 2.24

$$^R\vec{v} = n\, u\, n^* = (0, 1, 0, 0).$$

As can be seen in Figure 2.1 a rotation of 120 degrees about the referred axis moves the vector $\vec{v}$ in x-axis to a new position in y-axis.

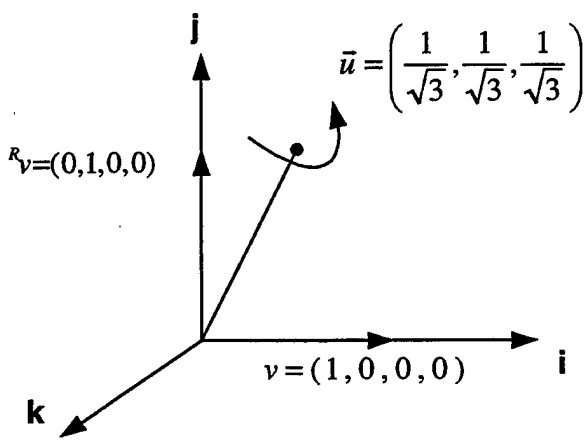


Figure 2.1. Rotation Using Quaternions.

Quaternion rotation operators have no singularities and they can relate the components of a vector when it is represented in any two independent coordinate frames in $\Re^3$.

C. THE KALMAN FILTER

1. Basics

The attitude of a body can be determined by either deterministic or non-deterministic methods [Ref. 8].

Deterministic models use two vectors representing the measurement in two different frames and calculate the matrix that transforms one coordinate system into another. For one measurement in each frame and no noise, the problem can be represented by a system of equations and easily solved. If there exists more than one measurement in one of the frames, which includes measurement noise, the solution is not unique and optimization methods must be used in order to find the best solution with minimum error.

Non-deterministic methods combine dynamic and/or kinematics models with sensor data. Estimation algorithms utilize a time series of measurements in order to estimate the attitude of the body. Among them, the Kalman Filter has been proven to be very useful for attitude estimation using vector measurements.

2. The Kalman Filter

The general Kalman filter structure can be applied to a variety of problems. One of the principal advantages of the filter is that it allows the blending of information from several different sources to develop the optimum estimate. The resulting estimate is statistically optimal with respect to the input data.

Kalman filters compute the solution recursively. In particular, each updated estimate of the state is computed from the previous estimate and the new input data, so only the previous estimate requires storage. Thus, the Kalman Filter is computationally more efficient than others filters that require all previous input data at each step of the filtering process and it becomes ideal for implementation on a digital computer.

3. The Filter Structure

The Kalman Filter is a filter that can optimally estimate, in real time, the states of a system based on noise corrupted inputs. State variables describe system behavior as a function of time. Errors in modeling the system are treated as process. The noise is assumed to be a Gaussian random process with a known covariance matrix [Ref. 8].

The process noise is used to focus on the measurements more than on the model itself. The filter minimizes the covariance of the estimated error between the model output and the measurements.

4. The Discrete Kalman Filter Algorithm

The following description of the Kalman filter is based on [Ref. 9]. It assumes a process model with n states. The equation describing how the process evolves in time is

$$x_{k+1} = F_k \; x_k + w_k \qquad (2.25).$$

The measurement equations define how the outputs are related to the states. For a system with m outputs the measurement equation is

$$z_k = H_k \; x_k + v_k \qquad (2.26).$$

Each term of Equations (2.25) and (2.26) an be defined as follows.

x_k : (n x 1) process state vector at time t_k

$F(x_k)$: (n x n) matrix relating the states x_k to x_{k+1} (state transition matrix)

w_k : (n x 1) process noise vector with known covariance structure at time t_k

z_k : (m x 1) measurement vector at time t_k

$H(x_k)$: (m x n) matrix relating the states x_k to the measurement y_k

v_k : (m x 1) measurement noise vector with known covariance structure at time t_k

The following covariance matrices are assumed for the noise vectors

$$E[w_k w_i^T] = \begin{cases} Q_k, & i = k \\ 0, & i \neq k \end{cases} \qquad (2.27)$$

$$E[v_k v_i^T] = \begin{cases} R_k, & i = k \\ 0, & i \neq k \end{cases} \qquad (2.28)$$

$$E[w_k v_i^T] = 0, \qquad \textit{for all } i,k \qquad (2.29).$$

The discrete Kalman Filter contains five recursive equations. A prior estimate of the state is updated using a correction proportional to the difference between the estimated output and the observation

$$\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H_k\ \hat{x}_k^- \right) \qquad (2.30).$$

The proportionality factor K_k is known as Kalman gain. This gain is calculated such that the trace of the covariance matrix for the states is minimized is given by

$$K_k = P_k^-\ H_k^T\ \left(H_k\ P_k^-\ H_k^T + R_k\right)^{-1}, \qquad (2.31)$$

where P_k^- is the covariance matrix for the state $\hat{x}_k^-$.

For the updated estimate of the state $\hat{x}_k$, a new covariance matrix can be computed

$$P_k = \left(I - K_k\ H_k \right) P_k^- \qquad (2.32).$$

The updated estimated is then projected ahead using the state transition matrix

$$\hat{x}_{k+1}^- = F_k\ \hat{x}_k \qquad (2.33).$$

Finally, the covariance matrix for the projected state is calculated as

$$P_{k+1}^- = F_k\ P_k\ F_k^T + Q_k \qquad (2.34).$$

The complete loop of the algorithm can be seen in Figure 2.2.

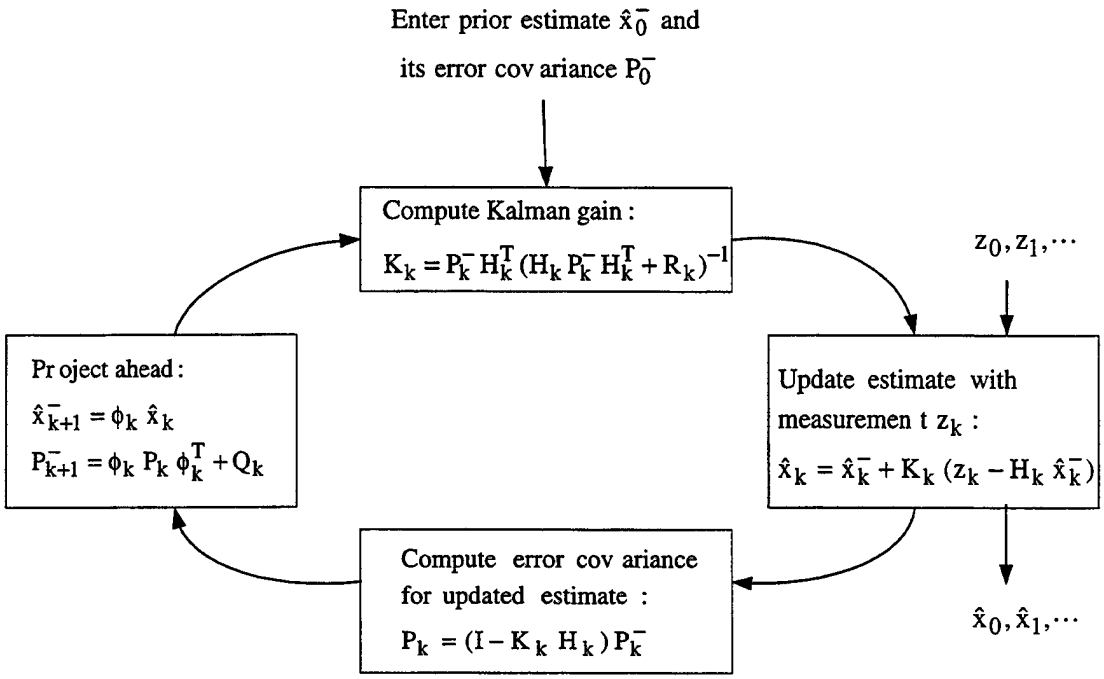


Figure 2.2. Kalman Loop. From [Ref. 9].

III. THE MODEL

A. INTRODUCTION

This chapter presents the steps one needs to carry out before designing a Kalman filter. It starts by defining the state variables. Characteristics of the sensor used make it necessary to include quaternion components as states. The relationship between quaternions and angular rates is derived, which allows the development of a process model for the complete system.

B. THE PROCESS MODEL

In order to start designing a Kalman filter, three steps must be completed:

- Choose the states of the system. The states or state variables are variables representing the dynamics of the system. The Kalman filter is designed to estimate these variables;
- Choose the output of the system. The outputs are variables related to the states, which are compared to observed values;
- Define a process model for the state variables. A process model is a mathematical model that has white noise as an input and a signal with the same characteristics as the state variables in the output. In general, this model is nonlinear.

The next sections explain how these three steps are conducted for that specific system.

1. Defining the State and Output Variables

The angular rates determine how the body behaves dynamically in space. Suppose the real values of angular rates were known. Given an initial position, integrating the angular rates through Euler equations would lead us to determine the orientation at each time step. Therefore, the angular rates can be used to represent the body dynamics and are included in the system state vector.

The angular rates are defined as

p: body angular velocity around the x axis (roll)

q: body angular velocity around the y axis (pitch)

r: body angular velocity around the z axis (yaw).

The second step is to define the outputs of the system. The outputs are values that must be compared to observed values or measurements coming from the sensors. The measurements, in this case, are angular rates, gravity vector, and local magnetic field vector obtained respectively from the rate sensors, accelerometers, and magnetometers. All measurements are taken in body frame.

Since the angular rates are included in state vector of the system and are also outputs, the first three elements of the output vector are identical to those of the state. This simplifies the filter design.

However, the other six observations (in body frame) need to be compared to the real values of gravity and magnetic field (in earth frame). This comparison is performed after the coordinate transformation, which is based on the orientation parameters

estimated by the filter (quaternions). In other words, known values of gravity and earth magnetic field are obtained from gravimetric and magnetometric charts in earth coordinates. They are converted into body coordinates using the quaternions. Finally, they are compared to the observed values in body coordinates coming from the sensors.

The last six elements of the output vector are related to constant values through the values of quaternion components, which are unknown. Thus, the quaternion components are included in the state vector as states to be estimated. The Extended Kalman Filter becomes a seven-state filter in which the states are three angular rates and four quaternion components.

Having defined the state and the output vectors, a process model for the state variables can be determined.

2. The Process Model

The state vector as defined in the previous section has two sets of variables. The first set is the three angular rates and the second is the four components of the quaternion.

The simplest process model for angular rates is one for generating those rates from white noise. Figure 3.1 shows a diagram of this process model.

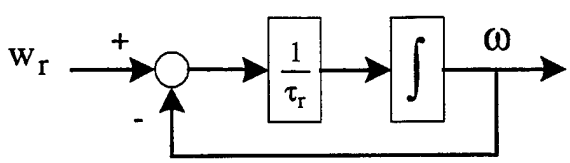


Figure 3.1. Process Model for Angular Rates.

In the diagram

- w_r is a 3-dimensional vector representing white noise sequences with known covariance structure that generate p, q, and r ;
- τ_r is the time constant for p, q, and r ; and
- ω represents the angular rates p, q, and r.

The variances of the white sequences and the time constants were adjusted so that the spectral characteristics of the signal generated by the model match those of the angular rates under normal operational conditions. Chapter IV will discuss how those values were chosen.

The above model represents the derivative of the angular rates as a function of the angular rates. Similar representation is expected for the quaternion components. The derivative of the quaternions should be a function of the states (angular rates and/or quaternions). The next section derives the dynamics of quaternion components, so that they can fit in the state space representation.

3. Relating Angular Rates and Quaternions

As it was seen in the beginning of this chapter, the quaternion components were included as states to be estimated. Therefore, a process model for quaternions must be defined. This means it is necessary to find out how the quaternion rates are related to the state (angular rates and qauternion components). This relationship can be obtained as follows [Ref. 7].

Imagine three quaternions that perform the following rotations on a vector $\bar{v}$

$m_{B(t)}^{E(t)}$: rotates a vector in body frame into earth frame at time t

$m_{B(t+\Delta t)}^{E(t+\Delta t)}$: rotates a vector in body frame into earth frame at time $t+\Delta t$

$n_{B(t+\Delta t)}^{B(t)}$: rotates a vector in body frame at time $t+\Delta t$ to body frame at time t.

Figure 3.2 shows the reference frame, the vector v in two different instances and the quaternions involved.

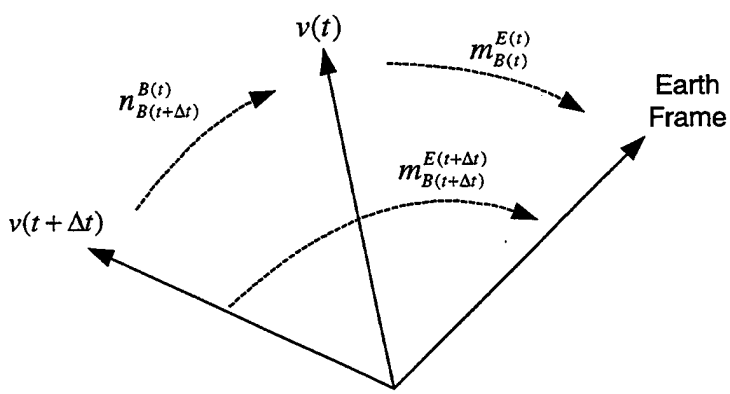


Figure 3.2. Determining the Quaternion Rates.

The variation of the quaternion m in the interval $t+\Delta t$ is

$$dm_B^E = m_{B(t+\Delta t)}^{E(t+\Delta t)} - m_{B(t)}^{E(t)} \qquad (3.1).$$

Using the chain rule

$$m_{B(t+\Delta t)}^{E(t+\Delta t)} = m_{B(t)}^{E(t)}\, n_{B(t+\Delta t)}^{B(t)} \qquad (3.2).$$

Substituting Equation (3.2) into (3.1) leads to

$$dm_B^E = m_{B(t)}^{E(t)}\, n_{B(t+\Delta t)}^{B(t)} - m_{B(t)}^{E(t)} = m_{B(t)}^{E(t)} \left(n_{B(t+\Delta t)}^{B(t)} - 1\right) \qquad (3.3).$$

Notice that $n_{B(t+\Delta t)}^{B(t)}$ is the quaternion rotating from the position at time $t+\Delta t$ to the position at time t. If the rotation is through an angle α about an axis $\vec{u}$, the corresponding quaternion can be represented as

$$n_{B(t+\Delta t)}^{B(t)} = \frac{\vec{u}}{|\vec{u}|}\sin\left(\frac{\alpha}{2}\right) + \cos\left(\frac{\alpha}{2}\right) \approx \frac{1}{2}\vec{u} + 1 \qquad (3.4).$$

Substituting (3.4) in (3.3) produces

$$dm_B^E = m_{B(t)}^{E(t)} \left(\frac{1}{2}\vec{u} + 1 - 1\right) = \frac{1}{2}\, m_{B(t)}^{E(t)}\, \vec{u} \qquad (3.5).$$

The above expression is divided by the time interval and evaluated in the limit when the interval goes to zero. As $\vec{u}$ is the direction of the Euler axis for the rotation between two positions in the same frame, $\vec{u}$ is stationary in this frame. Thus, the derivative in time of $\vec{u}$ is the angular rate of the body frame.

$$\dot{m}_B^E = \frac{1}{2}\, m_B^E\, \omega_B \qquad (3.6).$$

This equation indicates that the quaternion rates are a function of the body angular rates. Therefore, they can be included in the process model as if they were generated from the angular rates.

Unlike the angular rates, the quaternion componentes are generated from a mathematical expression and not from a process driven by white noise. This part of the model can be appended to the first one to complete the process model.

Using Equation 3.6, the process model becomes

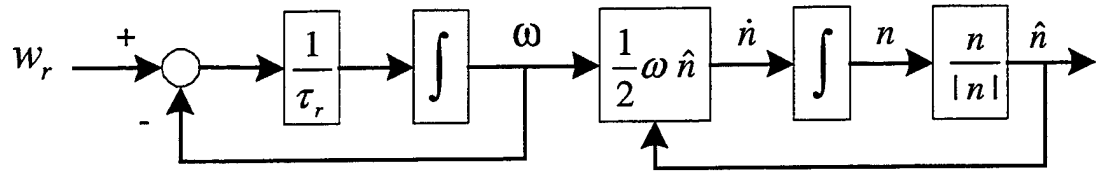


Figure 3.3. Process Model for Angular Rates and Quaternions.

where

$$\omega = \begin{bmatrix} 0 & r & -q & p \\ -r & 0 & p & q \\ q & -p & 0 & r \\ -p & -q & -r & 0 \end{bmatrix} \tag{3.7}$$

is a matrix whose elements are the angular rates and

$$\hat{n} = \frac{n}{|n|} \tag{3.8}$$

normalizes the quaternion to unit length.

Since unit quaternions are needed to perform the necessary rotation, the normalization is the constraint to be applied in this case.

The states and outputs of the system are defined as well as the process model. The design of the Kalman filter starts in the next Chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. THE KALMAN FILTER DESIGN

A. OVERVIEW

The Kalman filter solves the minimum mean-square error (MMSE) filtering problem using state space methods. It has great applicability in situations with nonlinear measurement relationships [Ref. 9]. The Extended Kalman filter linearizes the equations about a trajectory that is continually updated using states estimates.

This chapter starts presenting the state and output equations used for the Kalman filter design. Two approaches are discussed. The first approach works with a set of nonlinear equations, but is extremely complicated. The second approach is based on the convergence properties of quaternions. It leads to another way to solve the problem without dealing with nonlinear measurement equations. This approach is used for the design in this thesis.

B. FIRST APPROACH

As was discussed in Chapter III, the filter states are:

x_1 : angular rate p

x_2 : angular rate q

x_3 : angular rate r

x_4 : quaternion component a

x_5 : quaternion component b

x_6 : quaternion component c

x_7 : quaternion component d (scalar component)

Based on the model from Figure 3.3 the state equations can be written as

$$\dot{x}_1 = -\frac{1}{\tau_{rx}} x_1 + \frac{1}{\tau_{rx}} w_{rx} \tag{4.1}$$

$$\dot{x}_2 = -\frac{1}{\tau_{ry}} x_2 + \frac{1}{\tau_{ry}} w_{ry} \tag{4.2}$$

$$\dot{x}_3 = -\frac{1}{\tau_{rz}} x_3 + \frac{1}{\tau_{rz}} w_{rz} \tag{4.3}$$

$$\dot{x}_4 = \frac{1}{2\sqrt{x_4^2 + x_5^2 + x_6^2 + x_7^2}} (x_3 x_5 - x_2 x_6 + x_1 x_7) \tag{4.4}$$

$$\dot{x}_5 = \frac{1}{2\sqrt{x_4^2 + x_5^2 + x_6^2 + x_7^2}} (-x_3 x_4 + x_1 x_6 + x_2 x_7) \tag{4.5}$$

$$\dot{x}_6 = \frac{1}{2\sqrt{x_4^2 + x_5^2 + x_6^2 + x_7^2}} (x_2 x_4 - x_1 x_5 + x_3 x_7) \tag{4.6}$$

$$\dot{x}_7 = \frac{1}{2\sqrt{x_4^2 + x_5^2 + x_6^2 + x_7^2}} (-x_1 x_4 - x_2 x_5 - x_3 x_6) \tag{4.7}$$

It should be noted that besides the non-linearity in the parenthesis introduced by quaternion integration, square-rooted terms appear in the denominator due to the quaternion normalization.

The outputs are the angular rates, and the known values of gravity and local magnetic field converted to body coordinates. They are compared to the measurements. The first three measurements come from the rate gyros. The last six are readings coming from the accelerometers (low frequency components) and from the magnetometers.

The outputs are defined as:

z_1 : angular rate p

z_2 : angular rate q

z_3 : angular rate r

z_4 : component of gravity on the x-axis of the body frame

z_5 : component of gravity on the y-axis of the body frame

z_6 : component of gravity on the z-axis of the body frame

z_7 : component of the local magnetic field on the x-axis of the body frame

z_8 : component of the local magnetic field on the y-axis of the body frame

z_9 : component of the local magnetic field on the z-axis of the body frame

The major difficult with this approach is that the coordinate transformation involves the computation of a matrix to perform this transformation. This matrix is the same as that presented in equation 2.24, but uses the values of normalized quaternions.

$$R = \begin{bmatrix} \left(\hat{d}^2 + \hat{a}^2 - \hat{b}^2 - \hat{c}^2\right) & 2\left(\hat{a}\hat{b} - \hat{c}\hat{d}\right) & 2\left(\hat{a}\hat{c} + \hat{b}\hat{d}\right) \\ 2\left(\hat{a}\hat{b} + \hat{c}\hat{d}\right) & \left(\hat{d}^2 + \hat{b}^2 - \hat{a}^2 - \hat{c}^2\right) & 2\left(\hat{b}\hat{c} - \hat{a}\hat{d}\right) \\ 2\left(\hat{a}\hat{c} - \hat{b}\hat{d}\right) & 2\left(\hat{b}\hat{c} + \hat{a}\hat{d}\right) & \left(\hat{d}^2 + \hat{c}^2 - \hat{b}^2 - \hat{a}^2\right) \end{bmatrix} \tag{4.8}$$

The filter design was started using this approach but the output equations become quite complicated. As the design of the Extended Kalman filter involves partial derivatives, the final expression seemed to be mathematically too complex to allow real-time applications.

Equation 4.9 shows one of the six output equations for this approach

$$z_4 = \frac{(x_4{}^2 + x_7{}^2 - x_5{}^2 - x_6{}^2)\, h_1 + 2(x_4\, x_5 - x_6 x_7)\, h_2 + 2(x_4 x_6 + x_5 x_7)\, h_3}{x_4^2 + x_5^2 + x_6^2 + x_7^2}, \tag{4.9}$$

where

$h_1, h_2, and\ \ h_3$ are values of earth magnetic field measured in earth coordinates.

A second approach is described in the next section. It has two key advantages. It not only reduces the dimension of the output vector but also requires less computational effort.

C. ALGORITHMS FOR QUATERNION CONVERGENCE

1. Stating the Problem

Imagine a body in which a tri-orthogonal frame is placed at its center of gravity. If three accelerometers and three magnetometers are fixed to the origin of the frame, they start measuring components of the gravity and of the earth magnetic field in the axis of the frame. As these values are known and constant for a limited geographic area, one can expect that there exists a quaternion relating the measurements (values in body frame) to the real magnetic and gravity fields (values in earth frame).

Obviously there are several sources of errors, including:

- misalignments between pairs of axes in each sensor;
- impossibility of placing both sensors at the center of the body;
- variation of both gravity and magnetic field; and
- errors inherent to the sensors.

That means there is no quaternion that exactly converts what is measured (body frame) into the known values (earth frame). The solution is to determine the best quaternion such that, after the conversion, some criterion is satisfied. This chapter examines this problem using the minimum-squared-error (MSE) criterion. The approach is similar to that applied in [Ref. 4], but there a complementary filter was used. Another difference is that the error is minimized in earth frame in this derivation. Two different algorithms are evaluated and some discussions about the convergence properties are presented.

2. Minimizing the Error

Let Q be the error function defined as

$$^{E}Q = \varepsilon^{T}\varepsilon = (^{E}y_1 - M\ ^{B}y_0)^{T}\ (^{E}y_1 - M\ ^{B}y_0), \tag{4.10}$$

where

$^{E}y_1$: is a 6x1 vector with values of gravity and magnetic field in the earth frame

$^{B}y_0$: is a 6x1 vector with the measurements in the body frame

and

$$M = \begin{bmatrix} R & 0 \\ 0 & R \end{bmatrix}$$

where R is the matrix defined by Equation 2.24.

Because y_0 and y_1 are known, the error is a function of the matrix M, and thus it depends on the four components of the quaternion. The objective is to find iteratively the values of quaternion components that give the minimum error.

Several optimization algorithms exist in the literature. Among them the Gauss and Gauss-Newton are the most used ones. The main difference between them is that the former uses the first and second derivatives of the error function (gradient and Hessian) and the latter uses only the first derivative (Jacobian), which is related to the gradient. The different properties of convergence will be addressed at the end of this chapter.

The formulation for the iterative algorithm can be found in [Ref. 10]. For the Gauss-Newton method it is given as

$$\hat{n}_{k+1} = \hat{n}_k - \left[J^T(\hat{n}_k)\, J(\hat{n}_k) \right]^{-1} J^T(\hat{n}_k)\ {}^E\varepsilon(\hat{n}_k) \tag{4.11}$$

where

$\hat{n}$ is a vector with the four components of the quaternion and

J is the Jacobian matrix defined as

$$J = -\left[\left(\frac{\partial R}{\partial a}\ {}^B y_0 \right) \left(\frac{\partial R}{\partial b}\ {}^B y_0 \right) \left(\frac{\partial R}{\partial c}\ {}^B y_0 \right) \left(\frac{\partial R}{\partial d}\ {}^B y_0 \right) \right] \tag{4.12}.$$

For the Newton method

$$\hat{n}_{k+1} = \hat{n}_k - \left[\nabla^2{}_n\, {}^E Q(\hat{n}_k) \right]^{-1} \left[\nabla_n\, {}^E Q(\hat{n}_k) \right] \tag{4.13}$$

where

$\nabla_n {}^E Q$ is the gradient of the error function Q calculated in earth coordinates with respect to each one of the quaternion components. The gradient is calculated using the formula

$$\nabla_n \, {}^E Q = -2 \begin{bmatrix} \left(\frac{\partial R}{\partial a} \, {}^B y_0 \right)^T \\ \left(\frac{\partial R}{\partial b} \, {}^B y_0 \right)^T \\ \left(\frac{\partial R}{\partial c} \, {}^B y_0 \right)^T \\ \left(\frac{\partial R}{\partial d} \, {}^B y_0 \right)^T \end{bmatrix} \left({}^E y_1 - R \, {}^B y_0 \right) \qquad (4.14).$$

In the same way, the Hessian is the second derivative of the error function Q calculated in earth coordinates with respect to each one of the quaternion components. It is calculated by the formula

$$\nabla^2{}_n \, {}^E Q = \left\{ \begin{bmatrix} \frac{\partial^2 R}{\partial a^2} & \cdots & \frac{\partial^2 R}{\partial a \partial d} \\ \vdots & & \vdots \\ \frac{\partial^2 R}{\partial d \partial a} & \cdots & \frac{\partial^2 R}{\partial d^2} \end{bmatrix} {}^B y_0 \left({}^E y_1 - R \, {}^B y_0 \right) - \begin{bmatrix} \left(\frac{\partial R}{\partial a} \, {}^B y_0 \right)^T \\ \vdots \\ \vdots \\ \left(\frac{\partial R}{\partial d} \, {}^B y_0 \right)^T \end{bmatrix} \begin{bmatrix} \left(\frac{\partial R}{\partial a} \, {}^B y_0 \right)^T \\ \vdots \\ \vdots \\ \left(\frac{\partial R}{\partial d} \, {}^B y_0 \right)^T \end{bmatrix}^T \right\} \qquad (4.15).$$

So, the main goal is to find the best values for a, b, c, d such that when the matrix R is used to convert the measurements to earth frame the error is minimized.

It is worth noting that, as the matrix R represents a coordinate transformation, it is orthonormal. Thus, the only difference when minimizing the error in body coordinates is that the transpose of R is applied to the known values in earth frame. This leads to similar expressions and does not affect the convergence properties.

D. SECOND APPROACH

With the introduction of the convergence algorithm as an external loop to the Kalman filter, the quaternion components are now computed measurements. No equations involving coordinates transformations need to be evaluated. The algorithm is initialized using the last value of quaternion, which, for small step sizes is very close to the next value. Figure 4.5 shows how simple the filter becomes and gives a general idea of the data flow.

Once again, it should be noted that regardless of where the error is minimized (body or earth coordinates), the convergence is not affected. However, it is important that the Kalman filter knows which one is being used.

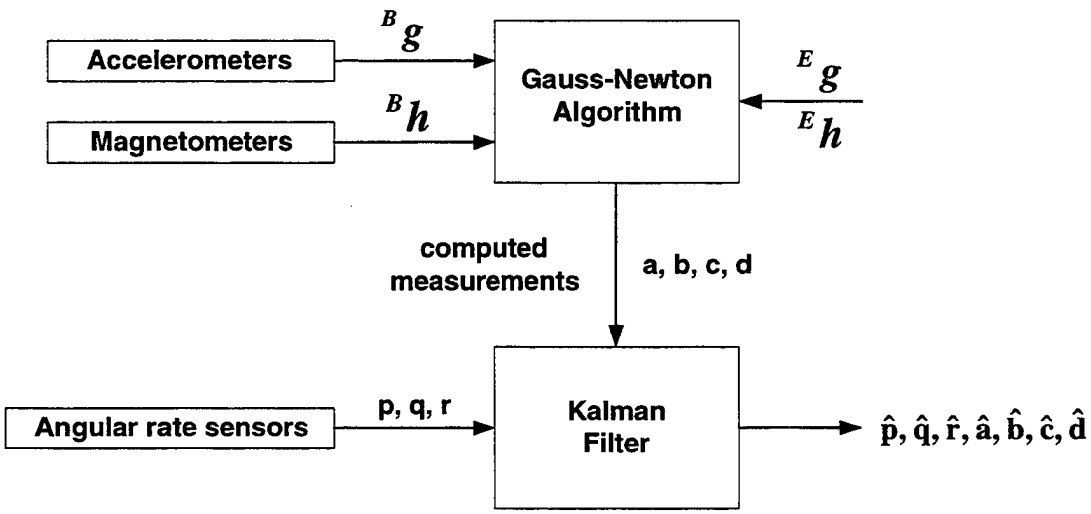


Figure 4.5. Final Diagram for the Filter.

For the final system the inputs are the same as before. However, the new outputs are

z_1 : angular rate p

z_2 : angular rate q

z_3 : angular rate r

z_4 : quaternion component a

z_5 : quaternion component b

z_6 : quaternion component c

z_7 : quaternion component d.

As can be seen the outputs are identical to the states. They are no longer related to the states by the nonlinear equations.

THIS PAGE INTENTIONALLY LEFT BLANK

V. MODELING SYSTEM NOISE

A. INTRODUCTION

One important assumption in the Kalman Filter is that any errors in the control input vector and the measurement vector are zero-mean and follow a Gaussian distribution. The Kalman Filter uses those two vectors as well as their covariance matrices. They are determined based on the motion dynamics of the body or vehicle (input vector) and on the characteristics of the sensors used (measurement noise). In order for the filter to achieve its best performance, the statistical properties of those vectors must be determined

It happened that, at the time this thesis was written, neither MARG sensors nor data from rigid body experiments were available. The only data set available was from experiments with the Small Autonomous Underwater Vehicle Navigation System (SANS) [Ref. 11]. That set includes readings of angular rates and gravity for one experiment and was used as basis for determining the statistical characteristics of the process to be estimated and of the sensors utilized. The following sections describe the procedures applied to obtain the statistical properties of both the input and measurement vectors.

B. PROCESS NOISE AND TIME CONSTANTS

The process noise w_k is the noise that when applied as an input to the process model gives the desired process as the output. Figure 3.2 is repeated here. It represents the process model for the angular rates and quaternions.

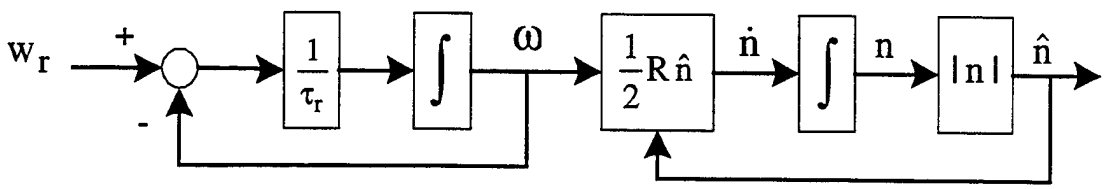


Figure 3.2. Process Model for Angular Rates and Quaternions.

The goal is to determine the values of noise variance and time constants that generate values of *p*, *q*, and *r* matching with real ones. It was assumed that the operation conditions of the vehicle were known (in this case the underwater vehicle). Values of angular rates from the SANS test were used. The process model was simulated with Simulink. Figure 5.1 shows the corresponding diagram. The S-function is a Matlab ODE45 routine that integrates Equations 4.1 through 4.7.

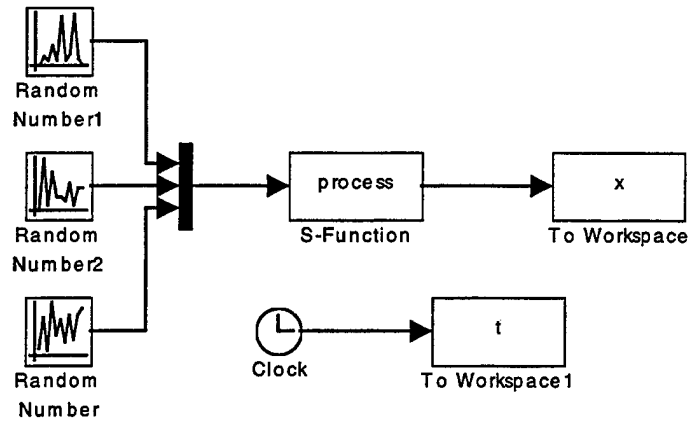


Figure 5.1. Simulation of the Process Model for *p*, *q*, and *r*.

The values of noise variance and the time constants were adjusted until the best match was obtained. Table 5.1 presents the best values for the parameters.

	Variance (rad^2/sec^2)	Time constant (sec)
p	$1x10^{-14}$	0.002
q	$1x10^{-4}$	0.15
r	$9x10^{-2}$	1.0

Table 5.1. Values of Variance and Time Constant for the Process Model.

The results of the simulation using the Simulink and the values of Table 5.1 were compared to the real values of angular rates. As the model is a simple first order model, the results can be considered satisfactory. Figure 5.2 shows a comparison between the real and the simulated processes.

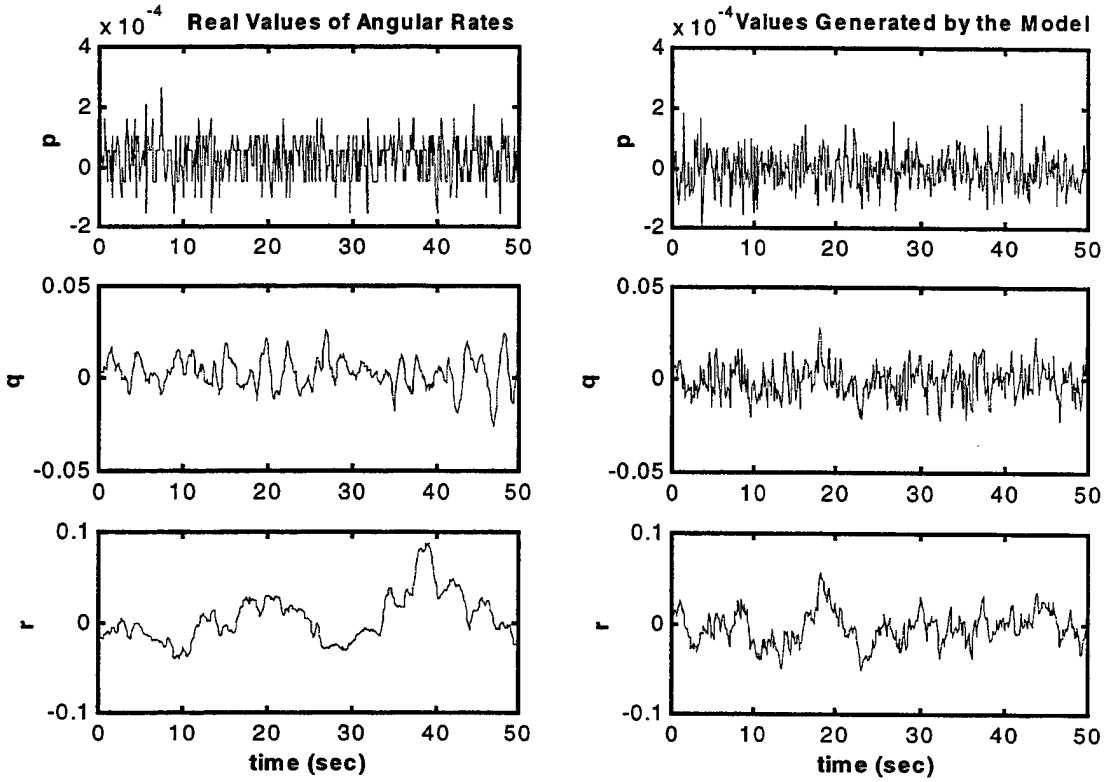


Figure 5.2. Comparison Between Real and Modeled Processes.

Two facts should be mentioned. First, the process model as depicted in Figure 3.2 does not have noise generating quaternions. Therefore only three variances are needed to describe the statistics of the process. Despite that, one has to keep in mind that the Equations 4.1 through 4.7 rule the entire process. So slight changes on the values of variance or time constant of the variable p can affect the processes of the other six variables.

Second, there is no means of checking the values of the quaternions generated by the model. However, as that part of the model involves only mathematical equations of deterministiv nature, it is expected that the model fits reality.

Based on the above assumptions, the covariance matrix of the process noise is defined as

$$Q = \begin{bmatrix} 1x10^{-14} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1x10^{-4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 9x10^{-2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (5.1).$$

Note that the cross correlation terms are all zero.

C. MEASUREMENT NOISE

The values of measurement noise for the rate sensors and accelerometers were determined from the same data set. Small intervals in which the values of the angular rate and gravity components seem to be stationary were used to estimate the values of variances. Table 4.2 presents the values obtained.

The values above can be compared to those from [Ref. 3] and [Ref. 8]. They both use MotionPak inertial measurement units made by Systron Donner, which include Sundstrand Q-Flex accelerometers and Systron Donner GyroChips. The values used in those references are presented in Table 5.3. As can be seen the estimated values from

Table 5.2 are quite reasonable and thus, they are be used for noise variances of the angular rate sensors and accelerometers.

	Variance
p	5.4×10^{-9} (rad/sec)^2
q	2.2×10^{-4} (rad/sec)^2
r	8.4×10^{-4} (rad/sec)^2
g_x	0.012 (m/sec^2)^2
g_y	0.028 (m/sec^2)^2
g_z	0.002 (m/sec^2)^2

Table 5.2. Values of Variance for the Measurement Noise.

	Angular rates	Acceleration
[Ref. 8]	7.6×10^{-5} (rad/sec)^2	0.01 (m/sec^2)^2
[Ref. 9]	6.4×10^{-3} (rad/sec)^2	3×10^{-5} (m/sec^2)^2

Table 5.3. Values of Variances for Angular Rates and Acceleration.

The data set does not include measurements of the earth's magnetic field. Therefore, no estimate can be made. However, [Ref. 9] mentions the variance for a two-axis magnetometer made by Precision Navigation Inc. as 2 degrees. That value

corresponds to a variance of approximately 10^{-3} if one considers a unitary normalized value for earth magnetic field.

Given the above discussion, the measurement noise covariance matrix can be written as

$$Q = \begin{bmatrix} 5.4x10^{-9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2.2x10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8.4x10^{-4} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.012 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.028 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.002 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix} \quad (5.2)$$

Note that all cross correlation terms in Q are also zero.

Another important observation is related to the dimension of Q. The matrix is 9 x 9 because it reflects the noises of nine sensors that furnish the measurements. However, the state space was defined as having six states and seven outputs. None of those accelerations or magnetic fields appear directly in the measurements. As it was discussed, the measurements are converted through the convergence algorithm into quaternions measurements. That conversion transforms the 9 x 9 Q matrix into a new 7 x 7 Q matrix explained in the next section.

D. COVARIANCE MATRIX OF THE MEASUREMENT

From the convergence algorithm (Equation 4.11), the following relationship is determined between the measurement, which has known covariance matrix, and the quaternion components:

$$n_{k+1} = n_k - F(n_k)\left[{}^{E}y - R(n_k)\ {}^{B}y\right] \tag{5.3}$$

$$n_{k+1} = n_k - F(n_k)\ {}^{E}y + F(n_k)\,R(n_k)\ {}^{B}y \tag{5.4}$$

That means the values for the quaternions that the convergence algorithm calculates are a function of the previous value or the guess. $F(n_k)$ depends on which type of algorithm is being used (Newton or Gauss-Newton).

The above expression can be rewritten as

$$y = A\,x + B, \tag{5.5}$$

where

$$B = n_k - F(n_k)\ {}^{E}y \tag{5.6}$$

and

$$A = F(n_k)\,R(n_k) \tag{5.7}$$

Assume that after one step the optimization algorithm converges to a reasonable value and that the initial guess always has the same value. Then, the parameters A and B are constant for any iteration of the Kaman filter. Using the same procedure as [Ref. 9], the covariance matrix for the quaternion components is given by

$$C_n = A\,C_y\,A^T \tag{5.8}$$

Two observations should be made. If the initial guess is not the same, the values of A and B change at each of the Kalman filter iteration. This is not a problem because it is easy to implement a covariance matrix that depends on the guess. Second, even if one considers that the convergence takes more than one iteration, the values of A and B are still functions of the initial guess.

In this thesis the first option is used and the covariance matrix is updated on each step with the value of $A = F(n_k)\,R(n_k)$ coming from the convergence algorithm.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. TESTS AND SIMULATIONS

A. INTRODUCTION

The Extended Kalman filter design based on the second approach achieves good results provided the algorithm that calculates the computed quaternion converges in few steps.

This chapter begins describing how the Kalman filter algorithm was tested and presents some results of the convergence process. It also presents all static and dynamic tests performed on the complete filter.

As no MARG sensor was available at the time this thesis was being completed, all data used here is generated mathematically.

B. TESTS OF THE CONVERGENCE ALGORITHM

The objective of the test is to check the convergence for different rotations, initial estimates, and noise levels. A six-element vector is chosen, which contains the components of gravity and local magnetic field vectors. An arbitrary quaternion is selected and used to rotate the initial vector. Gaussian noise is added to the rotated vector in order to simulate the measurement noise. The initial guesses are always values around the real value of the quaternion. The following examples illustrate the procedure.

Suppose the following values are measured in body frame:

$$^{B}y = [1 \quad 3 \quad 5 \quad 2 \quad -4 \quad 0]$$

and the values in earth frame as known as

$^{E}y = [\,5 \quad 1 \quad 3 \quad 0 \quad 2 \quad -4\,]$.

The initial guess is taken to be the quaternion

$\hat{n} = [\,0.3 \quad 0.4 \quad 0.6 \quad 0.7\,]$,

which after normalization is

$\hat{n} = [\,0.2020 \quad 0.3030 \quad 0.6061 \quad 0.7071\,]$.

Figure 6.1 shows the convergence when using the Gauss-Newton method. As can be seen, the algorithm converges in only three iterations. Figure 6.2 shows the convergence for the Newton algorithm. In both cases, no noise was added.

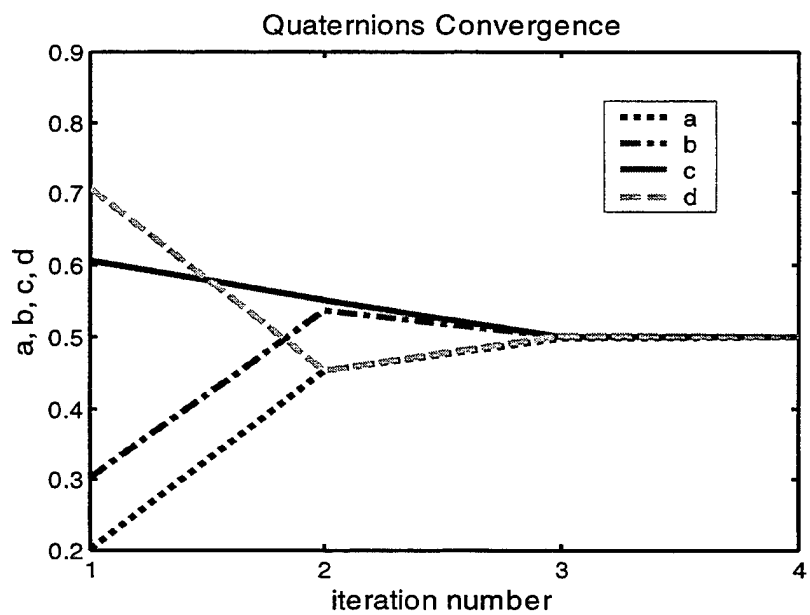


Figure 6.1. Quaternion Convergence Using Gauss-Newton (no Noise).

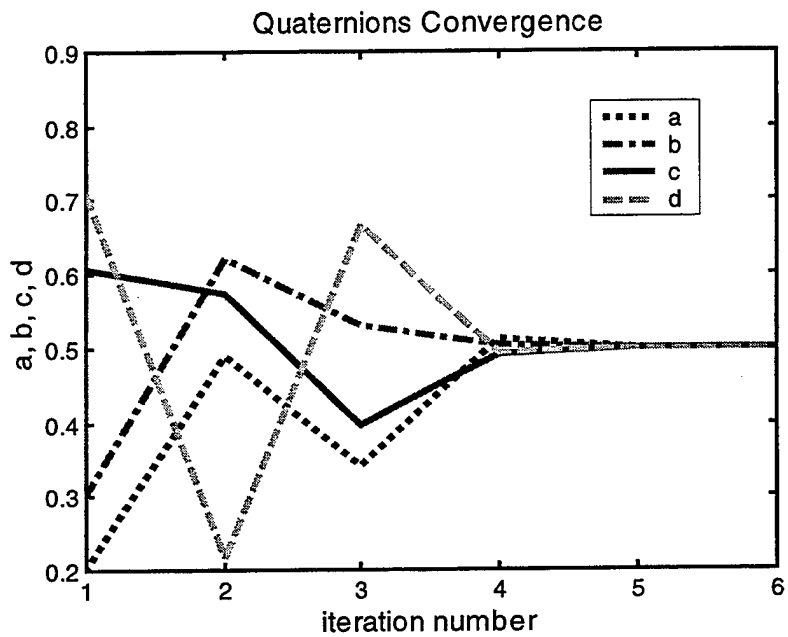


Figure 6.2. Quaternion Convergence Using Newton (no Noise).

In the experiments above, the quaternion converges to $\hat{n} = [\,0.5 \quad 0.5 \quad 0.5 \quad 0.5\,]$.

As it was shown in Chapter II, this quaternion corresponds to a rotation of 120 degrees about the axis $[\,1 \quad 1 \quad 1\,]$. As there is no measurement noise, the values are the same in both frames but with the order changed. The error in this case is zero.

Suppose noise is added to the measurement such that

${}^{B}y = [\,1.2 \quad 3.1 \quad 4.8 \quad 1.9 \quad -3.7 \quad 0.3\,]$

The algorithm now converges to the quaternion

$\hat{n} = [\,0.5291 \quad 0.5127 \quad 0.4912 \quad 0.4989\,]$.

This is the best value that can be found and the error is different from zero. Figure 6.3 shows the convergence using the Gauss-Newton algorithm.

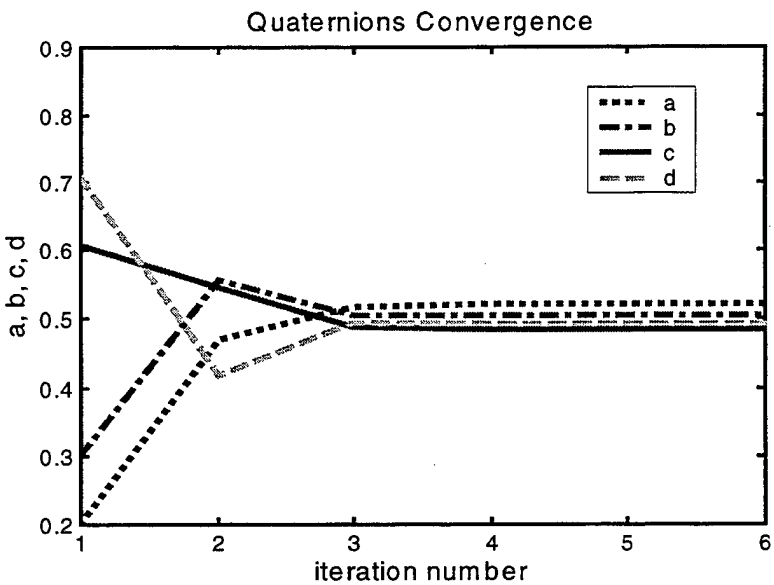


Figure 6.3. Quaternion Convergence Using Gauss-Newton (with Measurement Noise).

Extensive simulation was carried out with the Matlab routine from Appendix A. In this routine two unit vectors represent gravity and local magnetic field. The following elements are set in the routine:

- a vector representing the axis of rotation
- the variance of the measurement noise
- distance between the real value of quaternion and initial estimates.

The program rotates the unit vectors about the axis of rotation through several angles, adds noise, and solves the convergence problem for each case.

Table 6.1 shows how the values converge after three iterations of the Gauss-Newton algorithm. The vectors are rotated about the axis $\vec{m} = [1 \quad 3 \quad 2]$ through different angles and the noise measurement has standard deviation 0.05.

Angle	a	a_hat	b	b_hat	c	c_hat	d	d_hat
20	0.0464	0.0465	0.1392	0.1379	0.0928	0.0939	0.9848	0.9849
40	0.0914	0.0905	0.2742	0.2765	0.1828	0.1816	0.9397	0.9394
60	0.1336	0.1347	0.4009	0.3992	0.2673	0.2680	0.8660	0.8664
80	0.1718	0.1676	0.5154	0.5205	0.3436	0.3413	0.7660	0.7645
100	0.2047	0.2046	0.6142	0.6143	0.4095	0.4094	0.6428	0.6428
120	0.2315	0.2311	0.6944	0.6946	0.4629	0.4628	0.5000	0.4999
140	0.2511	0.2511	0.7534	0.7534	0.5023	0.5023	0.3420	0.3420
160	0.2632	0.2605	0.7896	0.7911	0.5264	0.5256	0.1736	0.1734
180	0.2673	0.2768	0.8018	0.7970	0.5345	0.5369	0.0000	0.0000
200	0.2632	0.2460	0.7896	0.7976	0.5264	0.5232	-0.1736	-0.1721
220	0.2511	0.2548	0.7534	0.7517	0.5023	0.5027	-0.3420	-0.3426
240	0.2315	0.2384	0.6944	0.6909	0.4629	0.4630	-0.5000	-0.5015
260	0.2047	0.2097	0.6142	0.6115	0.4095	0.4090	-0.6428	-0.6440
280	0.1718	0.1751	0.5154	0.5135	0.3436	0.3428	-0.7660	-0.7669
300	0.1336	0.1338	0.4009	0.4008	0.2673	0.2672	-0.8660	-0.8661
320	0.0914	0.0937	0.2742	0.2732	0.1828	0.1815	-0.9397	-0.9400
340	0.0464	0.0476	0.1392	0.1390	0.0928	0.0917	-0.9848	-0.9849
360	0.0000	-0.0003	0.0000	-0.0003	0.0000	0.0005	-1.0000	-1.0000

Table 6.1. Experiment to Verify the Convergence Using Gauss-Newton Method.

C. STATIC TESTS OF THE COMPLETE FILTER

The purpose of the static tests is to verify that the filter converges to the correct steady-state values after a single rotation. The body frame of the sensors is at rest, but with a fixed orientation relative to the earth frame. Among the ten different tests performed, three are presented in this thesis as described below.

1. Static Test 1 - Two Coincident Frames

Figure 6.4 shows the position of the vectors involved in both body and earth frames.

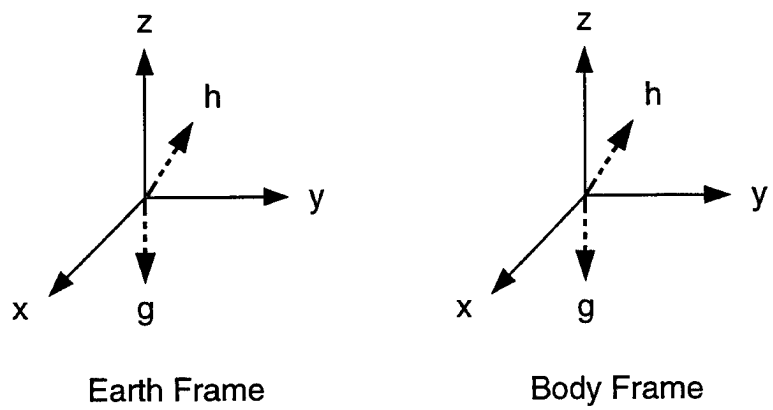


Figure 6.4. Static Test 1- Two Coincident Frames.

Unitary values of g (gravity) and h (magnetic induction) were used. The measurements were generated as

$$^{B}g = (0\,,0\,,-1) \qquad \text{and} \qquad ^{B}h = (-0.5\,,0\,,0.866)\,.$$

The measurement noises specified in Chapter V were added to those values that represent the sensor readings in the body frame.

Table 6.2 shows the initial values utilized for the states and the expected steady-state values. Figures 6.5 and 6.6 present the simulation results.

	p	q	r	a	b	c	d
Initial Values	1.0	1.0	1.0	0.5	0.5	0.5	0.5
Steady-state Values	0	0	0	0	0	0	1.0

Table 6.2. Static Test 1 – Initial and Steady-state Values.

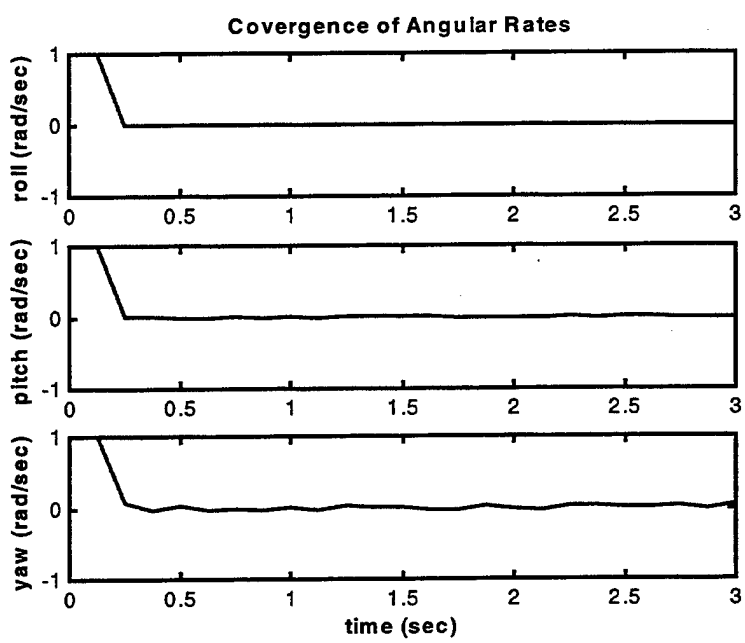


Figure 6.5. Static Test 1 - Convergence of Angular Rates.

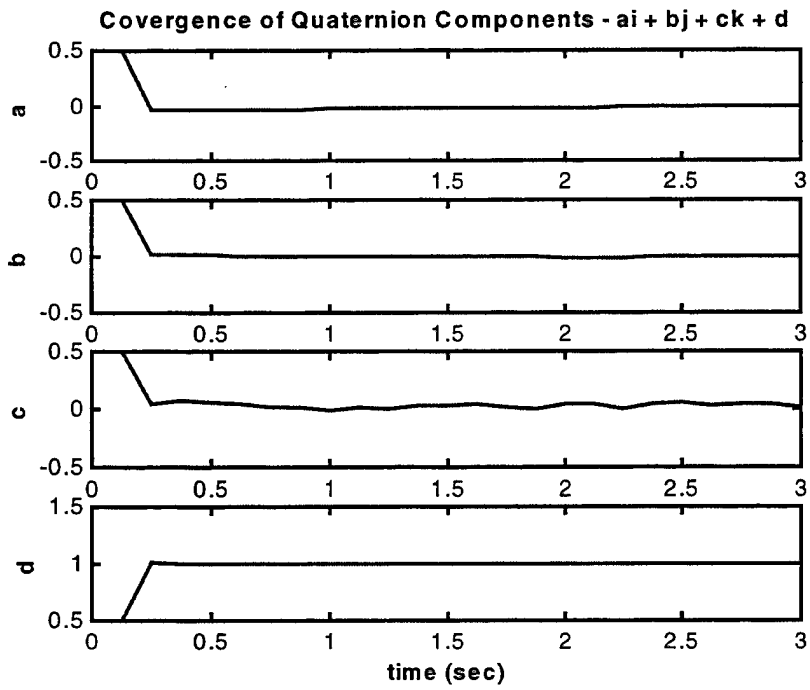


Figure 6.6. Static Test1 – Convergence of Quaternion Components.

2. Static Test 2 - Frames Related by the Quaternion (0.5 , 0.5 , 0.5 , 0.5)

Figure 6.7 shows the position of the vectors involved in both frames. This case corresponds to that discussed in Chapter II. Both frames are coincident and then the body frame is rotated 120 degrees about the axis $\vec{m} = [1 \quad 1 \quad 1]$.

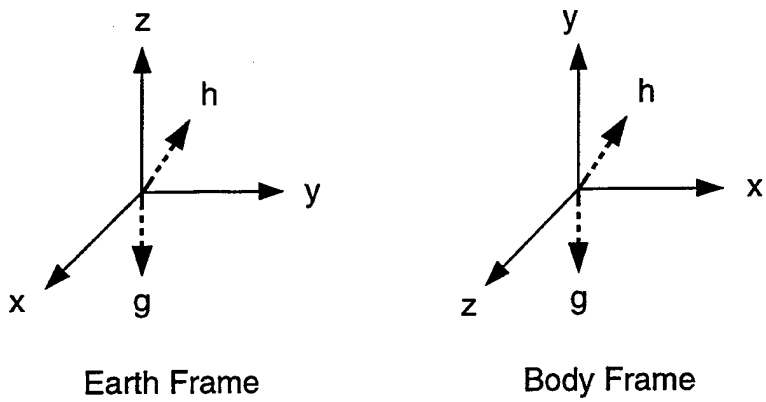


Figure 6.7. Static Test 2 – Rotated Frame.

Table 6.3 shows the initial values utilized for the states and the expected steady-state values. Figures 6.8 and 6.9 present the simulation results.

	p	q	r	a	b	c	d
Initial Values	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Steady-state Values	0	0	0	0.5	0.5	0.5	0.5

Table 6.3. Static Test 2 – Initial and Steady-state Values.

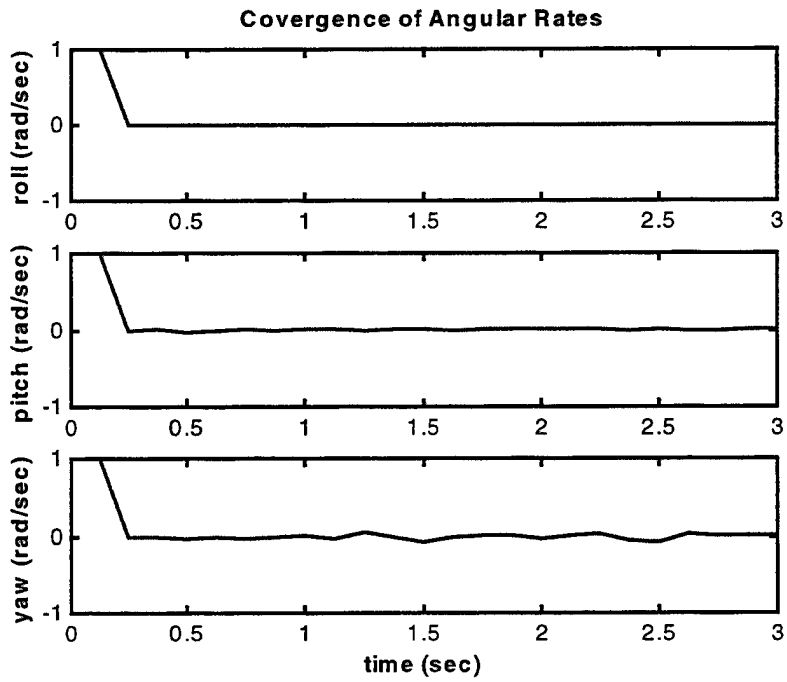


Figure 6.8. Static Test 2 – Convergence of Angular Rates.

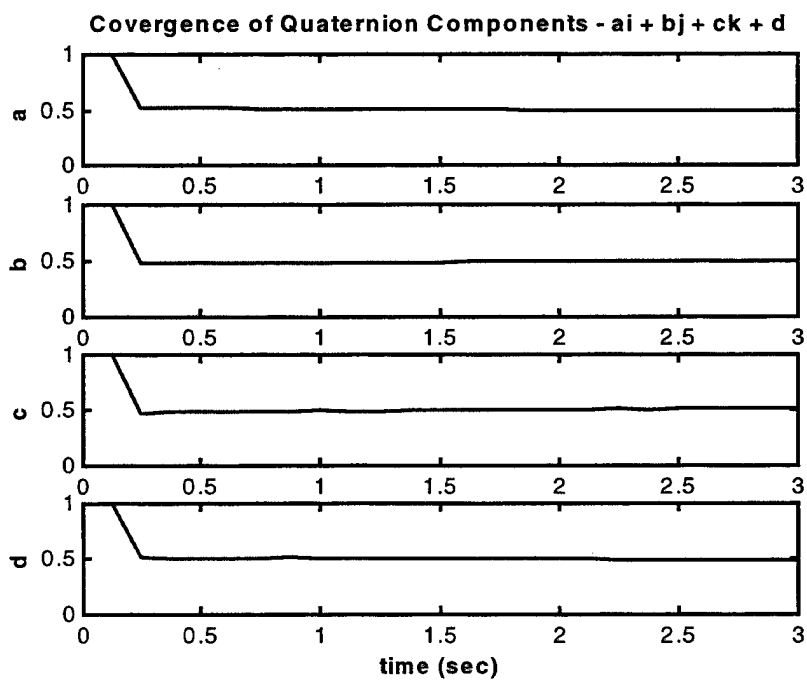


Figure 6.9. Static Test 2 – Convergence of Quaternion Components.

3. Static Test 3 - Frames Related by the Quaternion (0 , 0.707 , 0 , 0.707)

Figure 6.10 shows the position of the vectors involved in the measurements in both frames. This case corresponds to rotating the body frame about the y-axis through an angle of 90 degrees.

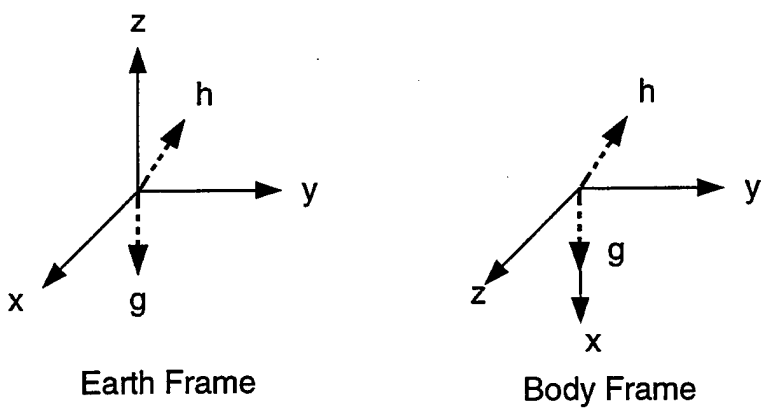


Figure 6.10. Static Test 3– Frame Rotated About One of Its Axis.

Table 6.4 shows the initial values utilized for the states and the expected steady-state values. Figures 6.11 and 6.12 present the simulation results.

	p	q	r	a	b	c	d
Initial Values	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Steady-state Values	0	0	0	0	0.71	0	0.71

Table 6.4. Static Test 3 – Initial and Steady-state Values.

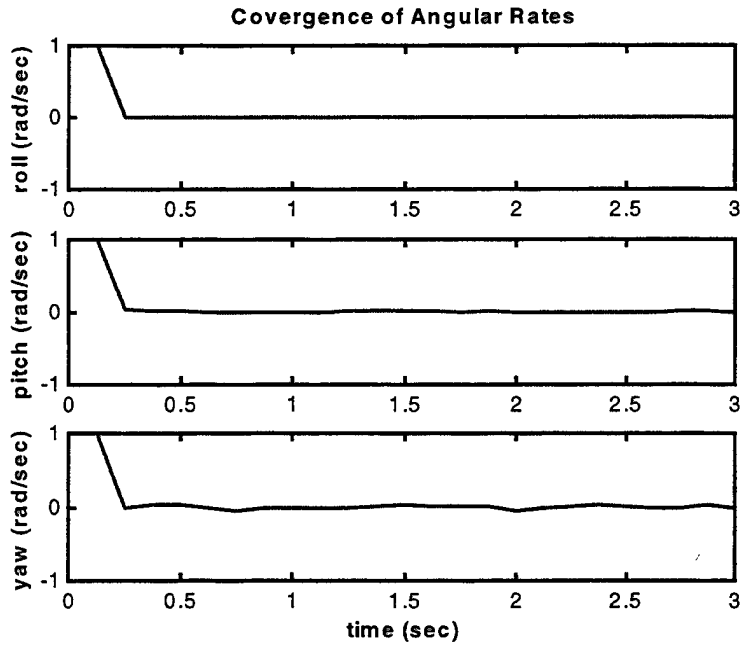


Figure 6.11. Static Test 3 – Convergence of Angular Rates

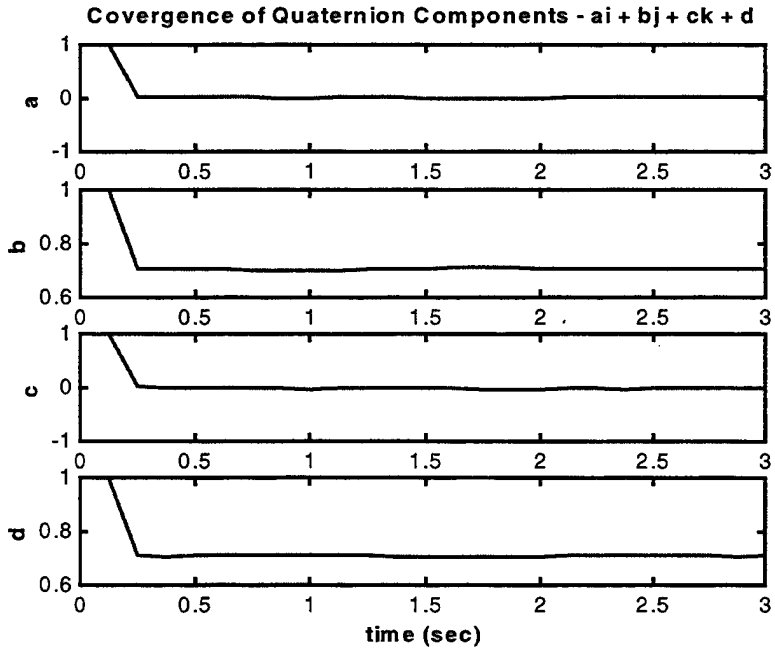


Figure 6.12. Static Test 3 – Convergence of Quaternion Components.

B. DYNAMIC TESTS OF THE COMPLETE FILTER

The dynamic tests intend to verify if the filter correctly estimates the attitude parameters when the body is moving.

Unlike the convergence and static tests, where the data is easily generated by one rotation of the values of gravity and local magnetic field vectors, in the dynamic tests the attitude is changing continuously. In this case a procedure to generate the data is more complex and involves computing, at each step, the body orientation. The procedure is described as follows.

1. Generating Data

Suppose body and earth frame are coincident such that the initial attitude parameters (Euler angles or quaternions) are known. If a rotation is applied to the body, one can easily determine how the parameters evolve in time by integrating Equation 3.6 or the Euler equations [Ref. 12]. In the first case, the quaternion components are obtained directly whereas in the second Euler angles need to be converted into direct cosine matrix and then quaternions.

As the goal is to check the quaternion filter output, the first option would not work with quaternion during the data generation. Therefore, the process of generating data for the experiment has four steps. An angular rate profile is defined. The integration of that profile using Euler equations leads to the Euler angles. The initial conditions are always null Euler angles corresponding to the position with both frames aligned. Those angles are used to make up the rotation matrix that transforms value of gravity and local magnetic field vectors from earth frame to body frame. Gaussian noise is added

representing the measurement noise. Figure 6.13 shows the diagram of the process. Appendix C includes the code for the process.

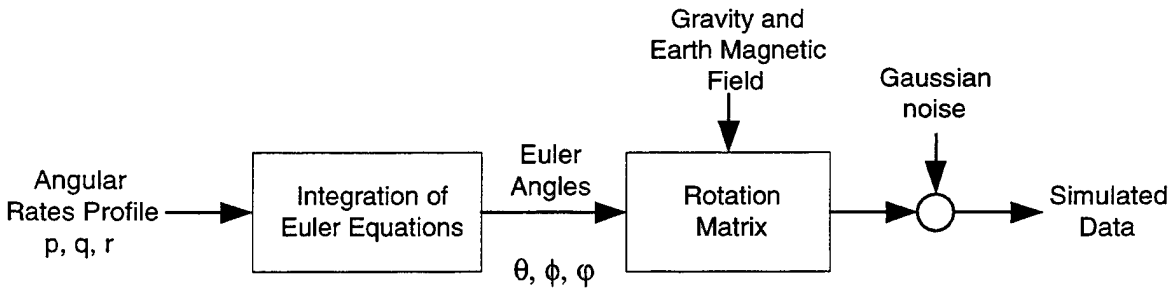


Figure 6.13. Data Generation for Dynamic Tests.

Two dynamic tests are presented in this thesis. The angular rates are always applied as pulses that rotate the body exactly 90 degrees. Thus, it is easy to check if the filter converges to the right values.

2. Dynamic Test 1 – Gaussian Pulse and Rotation About One Axis

For this test only rotation about the x-axis is performed. The profile for that rotation is a Gaussian pulse as seen in Figure 6.14. When integrated by Euler equations the final position of the body frame corresponds to a 90-degree rotation about that axis. Figure 6.15 shows both frames after the rotation is finished.

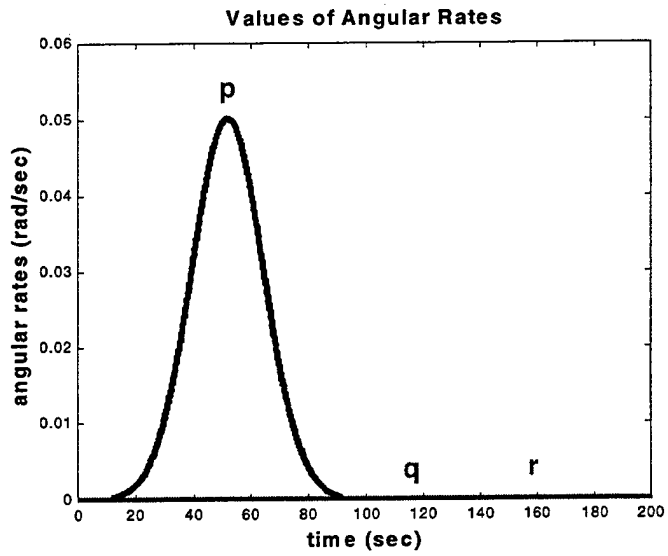


Figure 6.14. Dynamic Test 1 – Angular Rates Profile.

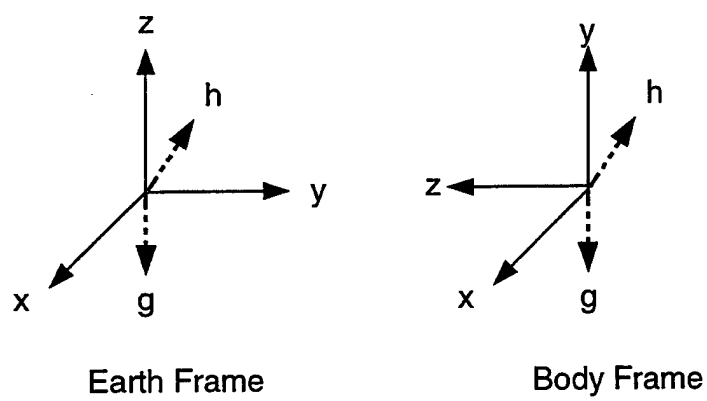


Figure 6.15. Dynamic Test1 – Final Position After Rotation.

The quaternion relating these two positions can be calculated by applying a rotation of 90 degrees about the x-axis. The value of that quaternion is

$$n = (\sin 45° (1, 0, 0), \cos 45°) = \left(\frac{\sqrt{2}}{2}, 0, 0, \frac{\sqrt{2}}{2} \right) = (0.71, 0, 0, 0.71).$$

Figure 6.16 shows the estimates for of angular rates. The Gaussian pulse is correctly estimated as well as the absence of rotation about the y and z-axes.

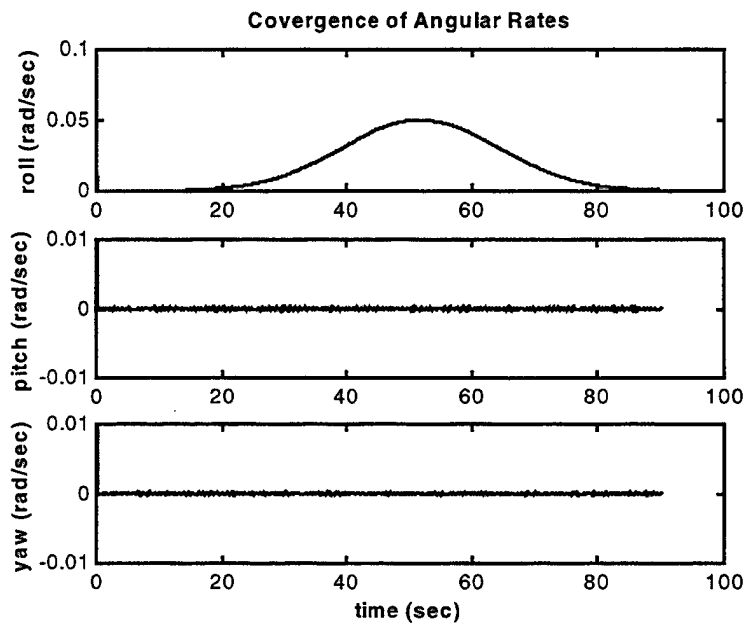


Figure 6.16. Dynamic Test 1 – Convergence of Angular Rates.

Figure 6.17 shows the convergence of the quaternion components. At the beginning, they converge almost immediately to the initial values (both frames coincident). The final values are exactly the ones calculated above.

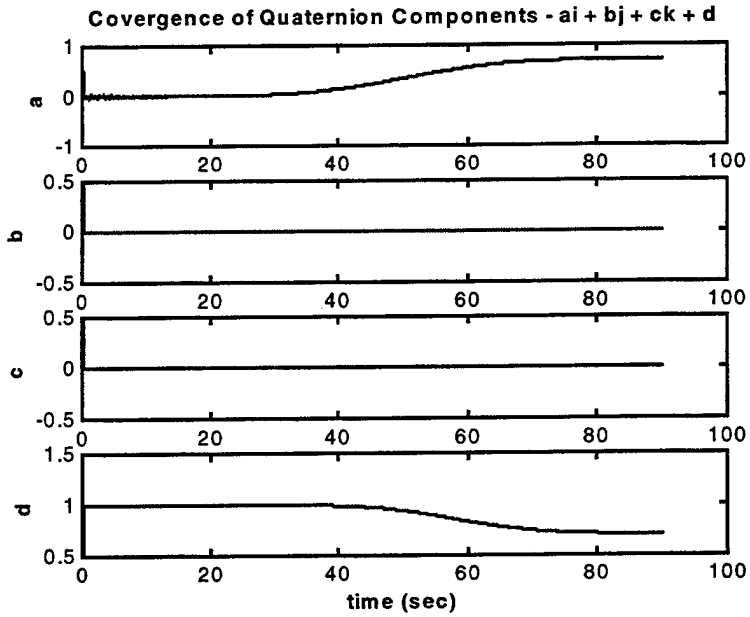


Figure 6.17. Dynamic Test 1 – Convergence of Quaternion Components.

3. Dynamic Test 2 – Square Pulse and Two Consecutive rotations

In this second test two rotations are performed. The profile for those rotations can be seen in Figure 6.18. The two square pulses rotate the body consecutively 90 degrees about x-axis and 90 degrees about y-axis. In this case, the final position after the first rotation is the same of the dynamic test 1 and the compound rotation is equivalent to a single rotation through 120 degrees about the axis $\vec{m} = [1 \quad 1 \quad 1]$, which is the same case of the static test 2. Therefore, the final quaternion realting both frames is

$$n = (\,0.5\,,0.5\,,0.5\,,0.5\,).$$

Figure 6.19 shows both frames after the rotations are complete.

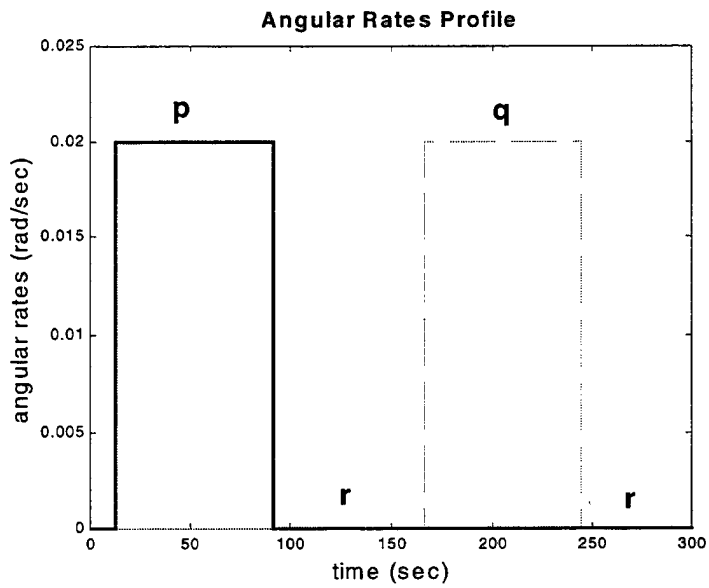


Figure 6.18. Dynamic Test 2 – Angular Rates Profile.

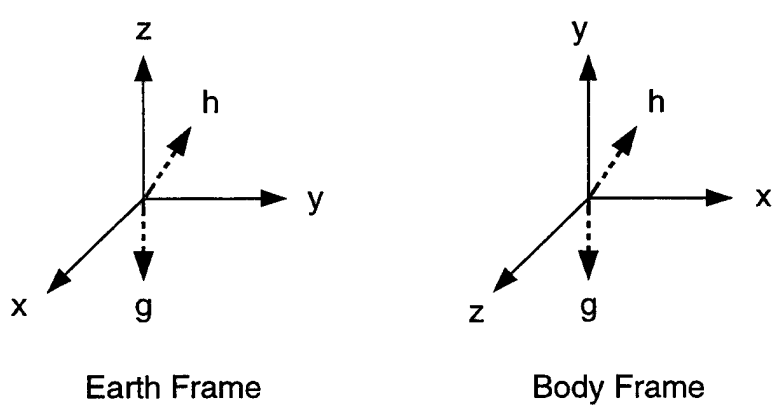


Figure 6.19. Dynamic Test 2 – Final Position After Rotations.

As expected, the filter estimates the angular rates and the quaternion components even in the pulse transitions. Figures 6.20 and 6.21 present the results of the estimates.

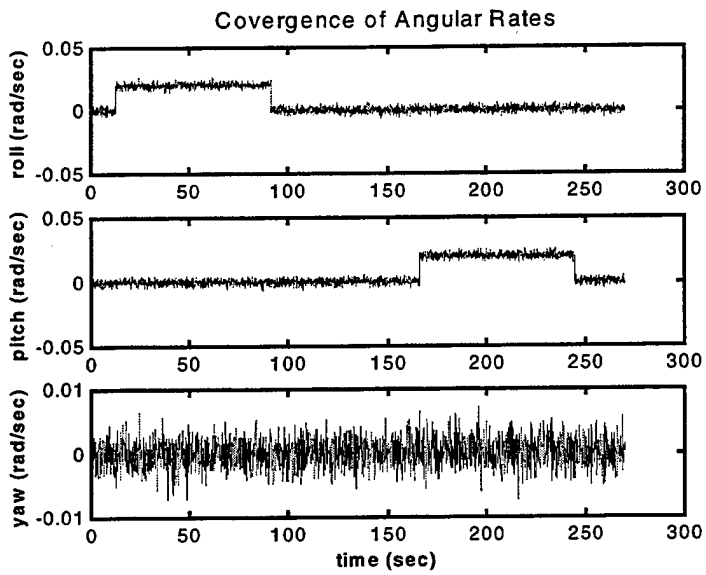


Figure 6.20. Dynamic Test 2 – Convergence of Angular Rates.

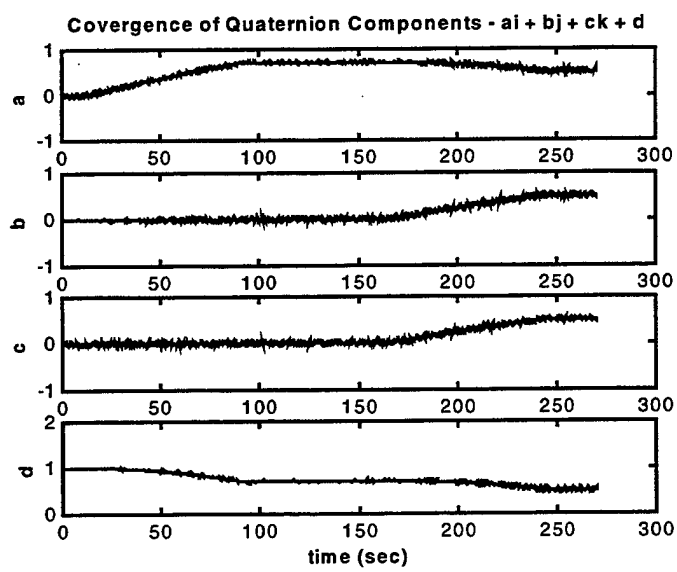


Figure 6.21. Dynamic Test 2 – Convergence of Quaternion Components.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS

A. SUMMARY

This thesis presented a complete design of an Extended Kalman filter for real-time estimation of rigid body motion attitude. The use of quaternions to represent rotations, instead of Euler angles, eliminated the long-standing problem of singularities called "gimbal lock" defined in Chapter II.

A simple first-order process model for rigid body angular rate measurements was defined, which closely matched real process data obtained from angular rate sensors. The quaternion rates were nonlinearly related to the current quaternion and the angular rates. The process model converted angular rates into quaternion rates, which were integrated to obtain quaternions.

Two approaches to the Kalman filter design were investigated. The first approach used nine output equations: three angular rates, three components of linear acceleration, and three components of the earth magnetic field. Since these output equations were highly nonlinear functions of the process state variables, the partial derivatives needed for the Kalman filter design were very complicated. As a result, it was decided that a filter formulated with these equations would not be useful for real-time applications.

The second approach utilized Gauss-Newton iteration to find the optimal quaternion that related the values of linear accelerations and earth magnetic field in the body coordinate frame and in the earth coordinate frame. The optimal quaternion was used as part of the measurement for the Kalman filter, which had seven output equations: three angular rates and four components of quaternions. Furthermore, the output equations were linear and the output matrix was the identity matrix. As a result, the partial derivatives were total derivatives and had constant values. The convergence algorithm replaced the computation of the partial derivatives of the nine nonlinear measurement equations. The computational requirements for the Kalman filter developed using this approach were significantly reduced, making it possible to estimate attitude in real time.

B. ANALYSIS OF RESULTS

Extensive tests were conducted to verify the convergence of Gauss-Newton and Newton algorithms, and the performance of the Kalman filter. In almost all cases and for both Gauss-Newton and Newton algorithms, the convergence to acceptable values occurred in less than three iterations. In very few cases (about 0.5%) where the algorithm diverged, the Kalman filter did not use the quaternion information. The filter implementation used only the Gauss-Newton algorithm because it does not involve second partial derivatives.

The filter achieved excellent results for all static and dynamic tests. The convergence to steady-state values took only two or three iteration steps. The two examples presented in this thesis showed that the filter was able to track angular rates either for smooth inputs (Gaussian pulse) or for abrupt ones (square pulses). Similar results were observed for the quaternion estimates. They quickly converged to the correct values and tracked the orientation of the body frame. No singularities were noticed, even when the two consecutive rotations described in Chapter II were applied (static test 2 and dynamic test 2).

C. SUGGESTIONS FOR FUTURE WORK

Although the current filter worked well, there is much work that could be done to further improve its performance. A few possibilities are as follows.

First, one can adjust the statistic parameters of the process model such that the filter design corresponds to the process to be estimated and sensors to be used.

Second, to make this filter faster, further work needs to be carried out on the convergence algorithm. The fact that it optimizes a function that includes a block diagonal matrix can be used to simplify the algorithm, mainly the steps involving matrix inversion.

Third, the filter should be tested using real data obtained from the MARG sensors currently under development at the NPS.

Finally, the best test would be the application of the attitude estimation into a filter that estimates position of a body or vehicle in real-time.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A – ROUTINES FOR CONVERGENCE TESTING

A1. GAUSS-NEWTON METHOD

```
%
%  Gauss-Newton Method - Normalized Guess
%
%  Evaluate the convergence for angles of rotation between 10 and 350 %
%  degrees
%
%     By João Luís Marins, Monterey, July 2000.
%



clc;  clear;

yb = input('Enter the a six-element vector in body coordinates : ');
h = input('Enter the axis of rotation (three-element row vector) : ');

res = [];

for theta = 10:20:350

theta = theta*pi/180;

%  Define the measurement noise
var = 0;
noise = sqrt(var)*randn(6,1);

%  Calculate the quaternion for the angle and axis chosen
h = h/(sqrt(h*h'));
n = [ ((sin(theta/2))*h)' ; cos(theta/2)]';

%  Rotate the known vector and add measurement noise
a=n(1);  b=n(2);  c=n(3);  d=n(4);
M = [d^2+a^2-b^2-c^2   2*(a*b-c*d)      2*(a*c+b*d);
      2*(a*b+c*d)    d^2+b^2-c^2-a^2    2*(b*c-a*d);
      2*(a*c-b*d)      2*(b*c+a*d)    d^2+c^2-a^2-b^2];
R = [M zeros(3) ; zeros(3) M];
ye = (R*yb');
ye = ye + noise;

%  Distance between correct value and guess
n_hat = (n - [0.02 0.02 0.02 0.02])';

ac_n =[];

%  Three iterations using the normalized quaternion
for i=1:3

   n_hat = n_hat/(sqrt(n_hat'*n_hat));
```

```
   if i==1
      n_guess = n_hat;
   end

ac_n = [ac_n n_hat];

a=n_hat(1); b=n_hat(2); c=n_hat(3); d=n_hat(4);

%  Calculate the partial derivatives needed for the Jacobian
   R_a = [ a  b c ; b -a -d ;  c  d -a];
   d_R_a = 2*[R_a zeros(3) ; zeros(3) R_a];
   R_b = [-b  a d ; a  b  c ; -d  c -b];
   d_R_b = 2*[R_b zeros(3) ; zeros(3) R_b];
   R_c = [-c -d a ; d -c  b ;  a  b  c];
   d_R_c = 2*[R_c zeros(3) ; zeros(3) R_c];
   R_d = [ d -c b ; c  d -a ; -b  a  d];
   d_R_d = 2*[R_d zeros(3) ; zeros(3) R_d];

jacobian = -[  (d_R_a*yb')' ; (d_R_b*yb')' ;
               (d_R_c*yb')' ; (d_R_d*yb')']';

%  Rotate the measured vector using the quaternion guess
%  and calculate the error between real value and rotated one
M_hat = [d^2+a^2-b^2-c^2    2*(a*b-c*d)      2*(a*c+b*d);
         2*(a*b+c*d)      d^2+b^2-c^2-a^2    2*(b*c-a*d);
         2*(a*c-b*d)       2*(b*c+a*d)      d^2+c^2-a^2-b^2];

R_hat = [M_hat zeros(3) ; zeros(3) M_hat];

err = ye - R_hat*yb';

%  Calculate next quaternion using previous one and the Jacobian
n_hat = n_hat - inv(jacobian'*jacobian)*jacobian'*err;

end

n_hat = n_hat/(sqrt(n_hat'*n_hat));
res = [res ; round(theta*180/pi) n n_guess' n_hat'];

end

result = [res(:,1:2) res(:,10) res(:,3) res(:,11) res(:,4) res(:,12)
res(:,5) res(:,13)];

%  Write the results to an excel file,  which must be opened
channel = ddeinit('excel', 'Converge.xls');
rc = ddepoke(channel,'r2c1:r19c9', result);
rc = ddeterm(channel);
```

A2. NEWTON-METHOD

```
%
%  In order to use the Newton Method the only change is to replace
%  the part inside the loop with the code below
%

ac_n = [ac_n n_hat];

a=n_hat(1); b=n_hat(2); c=n_hat(3); d=n_hat(4);

   dM_a2 = [ 2 0 0 ; 0 -2 0 ; 0 0 -2];
   dR_a2 = [dM_a2 zeros(3);zeros(3) dM_a2];
   dM_b2 = [-2 0 0 ; 0  2 0 ; 0 0 -2];
   dR_b2 = [dM_b2 zeros(3);zeros(3) dM_b2];
   dM_c2 = [-2 0 0 ; 0 -2 0 ; 0 0  2];
   dR_c2 = [dM_c2 zeros(3);zeros(3) dM_c2];
   dM_d2 = [ 2 0 0 ; 0  2 0 ; 0 0  2];
   dR_d2 = [dM_d2 zeros(3);zeros(3) dM_d2];

   dM_ab = [0 2 0 ; 2  0  0 ; 0  0  0];
   dR_ab = [dM_ab zeros(3);zeros(3) dM_ab];
   dM_ac = [0 0 2 ; 0  0  0 ; 2  0  0];
   dR_ac = [dM_ac zeros(3);zeros(3) dM_ac];
   dM_ad = [0 0 0 ; 0  0 -2 ; 0  2  0];
   dR_ad = [dM_ad zeros(3);zeros(3) dM_ad];
   dM_bc = [0 0 0 ; 0  0  2 ; 0  2  0];
   dR_bc = [dM_bc zeros(3);zeros(3) dM_bc];
   dM_bd = [0 0 2 ; 0  0  0 ; -2 0  0];
   dR_bd = [dM_bd zeros(3);zeros(3) dM_bd];
   dM_cd = [0 -2 0 ; 2  0 0 ;  0 0  0];
   dR_cd = [dM_cd zeros(3);zeros(3) dM_cd];

   aux1 = [yb'*dR_a2*err  yb'*dR_ab*err yb'*dR_ac*err yb'*dR_ad*err;
           yb'*dR_ab*err  yb'*dR_b2*err yb'*dR_bc*err yb'*dR_bd*err;
           yb'*dR_ac*err  yb'*dR_ab*err yb'*dR_c2*err yb'*dR_cd*err;
           yb'*dR_ad*err  yb'*dR_bd*err yb'*dR_cd*err yb'*dR_d2*err];

   R_a = [ a  b c ; b -a -d ;  c  d -a];
   d_R_a = 2*[R_a zeros(3) ; zeros(3) R_a];
   R_b = [-b  a d ; a  b  c ; -d  c -b];
   d_R_b = 2*[R_b zeros(3) ; zeros(3) R_b];
   R_c = [-c -d a ; d -c  b ;  a  b  c];
   d_R_c = 2*[R_c zeros(3) ; zeros(3) R_c];
   R_d = [ d -c b ; c  d -a ; -b  a  d];
   d_R_d = 2*[R_d zeros(3) ; zeros(3) R_d];

   aux2 = [(d_R_a*yb)' ; (d_R_b*yb)' ;(d_R_c*yb)' ;(d_R_d*yb)' ];

%  Rotate the measured vector using the quaternion guess
%  and calculate the error between real value and rotated one
M_hat = [d^2+a^2-b^2-c^2    2*(a*b-c*d)      2*(a*c+b*d);
         2*(a*b+c*d)      d^2+b^2-c^2-a^2    2*(b*c-a*d);
         2*(a*c-b*d)       2*(b*c+a*d)      d^2+c^2-a^2-b^2];
```

```
R_hat = [M_hat zeros(3) ; zeros(3) M_hat];

err = ye - R_hat*yb';

%  Calculate next quaternion using previous one, the gradient
%  and the Hessian

   grad = -2*aux2*err;
   hess = -2*(aux1-aux2*aux2');
   n_hat = n_hat - inv(hess)*grad;
```

APPENDIX B – EXTENDED KALMAN FILTER

B1. MAIN PROGRAM

```
%  This is the main program of the Extended Kalman filter
%  designed for estimating attitude of a body. The attitude
%  parameters are quaternions.
%
%  The measurements should be genrated in advance or collected
%  in real experiment and should be written to an excel file.
%  This file must be opened when the filter runs.
%
%  This program calls three other routines:
%     - converg : the convergence algorithm, can be Newton or
%                 Gauss-Newton;
%     - transition : allows the projection ahead through the
%                    linearization of the state transition matrix;
%     - projection : allows the projection ahead using the ODE45
%                    integration.
%
%     By João Luís Marins, Monterey, July 2000.
%

clc;  clear;

%  Reading measurements (p,q,r,gx,gy,gz,hx,hy,hz)
%  The data is read from an excel file, which must be opened
sim_data = [];
channel = ddeinit('excel', 'dynamic_2.xls');
sim_data = ddereq(channel,'r1c1:r2400c9');
rc = ddeterm(channel);

%  Definitions
run_time = 4.50;                  %  simulation time in minutes
T = 0.125;                        %  resolution time in seconds
samp = run_time*60/T;             %  number of samples

%  Time constants used indide the routines transition and projection
tau_rx = 0.002;
tau_ry = 0.15;
tau_rz = 1.0;

%  Values of std dev used in the Q matrix - E[w'*w]
std1 = 1e-7;
std2 = 0.01;
std3 = 0.30;

%  Process model - noise vector for angular rates
randn('state',sum(100*clock));
wrx = std1*randn(1,samp);
wry = std2*randn(1,samp);
wrz = std3*randn(1,samp);
```

```
% Process model - noise vector for quaternions - no noise
wna = zeros(1,samp);
wnb = zeros(1,samp);
wnc = zeros(1,samp);
wnd = zeros(1,samp);

% System noise vector
w = [ wrx ; wry ; wrz ; wna ; wnb ; wnc ; wnd ];

% Measurements - noise vector for angular rates
vrx = randn(1,samp);
vry = randn(1,samp);
vrz = randn(1,samp);

% Measurements - noise vector for quaternions
vna = randn(1,samp);
vnb = randn(1,samp);
vnc = randn(1,samp);
vnd = randn(1,samp);

% Measurement noise vector
v = [ vrx ; vry ; vrz ; vna ; vnb ; vnc ; vnd ];

% Step 0 : Initial conditions
x_hat_minus = [ 0.01 ; 0.01 ; 0.01 ; 0.5 ; 0.5 ; 0.5 ; 0.5 ];

% Initial covariance matrix
P_minus = diag([ 0.5 0.5 0.5 0.5 0.5 0.5 0.5]);

% Matrices for flag = 0 (quaternions do not converge)
R_0 = diag([(7.36e-5)^2 (0.015)^2 (0.029)^2]);
H_0 = [eye(3)  zeros(3,4)];

% Matrices for flag = 1 (quaternions converge)
R_aux = diag([(0.110^2) (0.167^2) (0.045^2)(0.03^2)(0.03^2) (0.03^2)]);
H_1 = eye(7);

% Process noise covariance matrix
Q = diag([ std1^2 std2^2 std3^2 0 0 0 0 ]);

x_hat_plus_cum = [x_hat_minus];

% Accumulate all the estimates
n_acum = [];

% Registry number of times the convergence algorithm diverges
count_flag = 0;
flag_cum = [];

for k = 2:samp

% Step 1 : Read sensors (angular rates, gravity, and magnetic field)

ye = [0 ; 0 ; -1 ; -0.5 ; 0 ; sqrt(3)/2 ];
g_norm = (sim_data(k,4:6))'/sqrt((sim_data(k,4:6))*(sim_data(k,4:6))');
```

```
h_norm = (sim_data(k,7:9))'/sqrt((sim_data(k,7:9))*(sim_data(k,7:9))');

yb = [ g_norm ; h_norm ];

%  First guess is the current quaternion
guess = x_hat_minus(4:7);

%  Step 2 : Converge the quaternions using x_hat(k|k-1)

[n,flag,A] = Converg(ye,yb,guess,k);
flag_cum = [flag_cum flag];
P_minus = diag([ 0.5 0.5 0.5 0.5 0.5 0.5 0.5]);
n_acum = [n_acum n];

if flag == 1   %  quaternion converge

   z = [(sim_data(k,1:3))' ; n];

   %  The A matrix relates the covariance of the measurement
   %  noise (6 by 6) to the covariance of the quaternions
   %   after convergence (4 by 4)
   R_1 = [ R_0 zeros(3,4) ; zeros(4,3) A*R_aux*A' ];

   %  Step 3 : Determine the Kalman gain
   K = P_minus*H_1'*inv(H_1*P_minus*H_1'+ R_1);

   %  Step 4 : Update estimate with measurement
   x_hat_plus = x_hat_minus + K*[z - H_1*x_hat_minus];

   %  Step 5 : Compute error covariance for updated estimate
   P_plus = (eye(7) - K*H_1)*P_minus;

else  %  quaternion diverge

   z = [(sim_data(k,1:3))'];

   %  Step 3 : Determine the Kalman gain
   K = P_minus*H_0'* inv(H_0*P_minus*H_0'+ R_0);

   %  Step 4 : Update estimate with measurement
   x_hat_plus = x_hat_minus + K*[z - H_0*x_hat_minus];

   %  Step 5 : Compute error covariance for updated estimate
   P_plus = (eye(7) - K*H_0)*P_minus;

   count_flag = count_flag + 1;

  end

x_hat_plus_cum = [x_hat_plus_cum  x_hat_plus];

%  Step 6 : Project ahead
phi = transition(x_hat_plus);
P_minus = phi*P_plus*phi'+ Q;
[ti , x_aux] = ode45('projection',[0 T],x_hat_plus);
```

```
x_hat_minus = (x_aux(find(ti==T),:))';

end

m = length(x_hat_plus_cum(1,:));

figure(1);
subplot(311);plot((1:m)*0.125,x_hat_plus_cum(1,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
title('Covergence of Angular Rates'); ylabel('roll (rad/sec)');
subplot(312);plot((1:m)*0.125,x_hat_plus_cum(2,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
ylabel('pitch (rad/sec)');
   subplot(313);plot((1:m)*0.125,x_hat_plus_cum(3,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
ylabel('yaw (rad/sec)');   xlabel('time (sec)');

figure(2);
subplot(411);plot((1:m)*0.125,x_hat_plus_cum(4,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
title('Covergence of Quaternion Components - ai + bj + ck + d');
   ylabel('a');
subplot(412);plot((1:m)*0.125,x_hat_plus_cum(5,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
ylabel('b');
subplot(413);plot((1:m)*0.125,x_hat_plus_cum(6,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
ylabel('c');
subplot(414);plot((1:m)*0.125,x_hat_plus_cum(7,:),'b');
set (gca,'LineWidth',1.5,'FontSize',12);hold;
xlabel('time (sec)');
ylabel('d');
```

B2. CONVERGENCE ROUTINE

```
function    [n,flag,A] = converg(ye,yb,guess,k)

%     This function uses the Gauss-Newton method to find the best
%     quaternion that rotates the measurements of gravity and magnetic
%     field from body coordinates to earth coordinates. The first guess
%     is the value calculated from the state equations. The calculated
%     values are normalized at each iteration.
%     The Gauss-Newton is a first order approximation where the
%     Jacobian is used instead of the gradient and hessian of the
%     Newton method.
%     The flag indicates if the quaternion converged to a value close
%     to the initial guess or not (1 or 0). The threshold can be
%     adjusted.
%
%     By João Luís Marins, Monterey, June 2000.
%

n_hat = guess;
ac_n = [];

for i=1:1

%     Normalize the quaternion
n_hat = n_hat/(sqrt(n_hat'*n_hat));
ac_n = [ac_n n_hat];

a=n_hat(1);       b=n_hat(2);       c=n_hat(3);       d=n_hat(4);

%  Calculate the partial derivatives needed for the Jacobian
   R_a = [ a  b c ; b -a -d ;  c  d -a];
   d_R_a = 2*[R_a zeros(3) ; zeros(3) R_a];
   R_b = [-b  a d ; a  b  c ; -d  c -b];
   d_R_b = 2*[R_b zeros(3) ; zeros(3) R_b];
   R_c = [-c -d a ; d -c  b ;  a  b  c];
   d_R_c = 2*[R_c zeros(3) ; zeros(3) R_c];
   R_d = [ d -c b ; c  d -a ; -b  a  d];
   d_R_d = 2*[R_d zeros(3) ; zeros(3) R_d];

jacobian = -[  (d_R_a*yb')' ; (d_R_b*yb')' ;
               (d_R_c*yb')' ; (d_R_d*yb')']';

%  Rotate the measured vector using the quaternion guess
%  and calculate the error between real value and rotated one
M_hat = [d^2+a^2-b^2-c^2    2*(a*b-c*d)      2*(a*c+b*d);
         2*(a*b+c*d)     d^2+b^2-c^2-a^2    2*(b*c-a*d);
         2*(a*c-b*d)      2*(b*c+a*d)     d^2+c^2-a^2-b^2];

R_hat = [M_hat zeros(3) ; zeros(3) M_hat];

err = ye - R_hat*yb';
```

```
%  Calculate next quaternion using previous one and the Jacobian
n_hat = n_hat - inv(jacobian'*jacobian)*jacobian'*err;

end

n_hat = n_hat/(sqrt(n_hat'*n_hat));

%     Calculate the matrix that converts the measurement errors
%     of the six sensors into the calculated measurement error
%     of the four computed quaternions
A = inv(jacobian'*jacobian)*jacobian'*R_hat;

%     Check if the algorithm converges
distance = sqrt(sum((n_hat-guess).^2));
n = n_hat;

if    (k == 2)
   flag = 1;
elseif (distance <= 0.5)
   flag = 1;
else
   flag = 0;
end
```

B3. LINEARIZATION OF THE STATE TRANSITION MATRIX

```
function    xprime = projection(t,x);

%     This function is an ODE file with the differential equations
%     describing the state equations for the filter. The filter calls
%     this routine using The ODE45 function when projecting ahead
%     (calculating x_hat_minus).
%
%     By João Luís Marins, Monterey, July 2000.
%

%     time constants for angular rates
tau_rx = 0.002;
tau_ry = 0.15;
tau_rz = 1.00;

xprime = [-x(1)/tau_rx;
          -x(2)/tau_ry;
          -x(3)/tau_rz;
          ( x(3)*x(5)-x(2)*x(6)+x(1)*x(7))/
/(2*sqrt(x(4)^2+x(5)^2+x(6)^2+x(6)^2));
          (-    x(3)*x(4)+x(1)*x(6)+x(2)*x(7))/
/(2*sqrt(x(4)^2+x(5)^2+x(6)^2+x(6)^2));
          ( x(2)*x(4)-x(1)*x(5)+x(3)*x(7))/
(2*sqrt(x(4)^2+x(5)^2+x(6)^2+x(6)^2));
          (-x(1)*x(4)-x(2)*x(5)-x(3)*x(6))/
(2*sqrt(x(4)^2+x(5)^2+x(6)^2+x(6)^2))];
```

B4. PROJECT AHEAD USING ODE45

```
function    [phi] = transition(x_hat_plus)

%     This function calculates the state transition matrix phi,
%     which is the linearization of the F matrix.
%     The filter calls this routine when projecting ahead
%     (calculating P_minus).
%

%     resolution time in seconds
T = 0.01;

%     time constants for angular rates
tau_rx = 0.002;
tau_ry = 0.15;
tau_rz = 1.0;

x1 = x_hat_plus(1);
x2 = x_hat_plus(2);
x3 = x_hat_plus(3);
x4 = x_hat_plus(4);
x5 = x_hat_plus(5);
x6 = x_hat_plus(6);
x7 = x_hat_plus(7);
n2 = sqrt(x4^2 + x5^2 + x6^2 + x7^2);

f1 = -x1/tau_rx;
f2 = -x2/tau_ry;
f3 = -x3/tau_rz;
f4 = 0.5*( x3*x5 - x2*x6 + x1*x7)/n2;
f5 = 0.5*(-x3*x4 + x1*x6 + x2*x7)/n2;
f6 = 0.5*( x2*x4 - x1*x5 + x3*x7)/n2;
f7 = 0.5*(-x1*x4 - x2*x5 - x3*x6)/n2;

%     Derivatives for linearization

df1_x1 = 1 - T/tau_rx;
df2_x2 = 1 - T/tau_ry;
df3_x3 = 1 - T/tau_rz;

df4_x1 = 0.5*( x7/n2);
df4_x2 = 0.5*(-x6/n2);
df4_x3 = 0.5*( x5/n2);
df4_x4 = 1 + 0.5*(-f4*x4/n2^3);
df4_x5 = 0.5*( x3/n2 - f4*x5/n2^3);
df4_x6 = 0.5*(-x2/n2 - f4*x6/n2^3);
df4_x7 = 0.5*( x1/n2 - f4*x7/n2^3);

df5_x1 = 0.5*( x6/n2);
df5_x2 = 0.5*( x7/n2);
df5_x3 = 0.5*(-x4/n2);
df5_x4 = 0.5*(-x3/n2 - f5*x4/n2^3);
```

```
df5_x5 = 1 + 0.5*(-f5*x5/n2^3);
df5_x6 = 0.5*( x1/n2 - f5*x6/n2^3);
df5_x7 = 0.5*( x2/n2 - f5*x7/n2^3);

df6_x1 = 0.5*(-x5/n2);
df6_x2 = 0.5*( x4/n2);
df6_x3 = 0.5*( x7/n2);
df6_x4 = 0.5*( x2/n2 - f6*x4/n2^3);
df6_x5 = 0.5*(-x1/n2 - f6*x5/n2^3);
df6_x6 = 1 + 0.5*(-f6*x6/n2^3);
df6_x7 = 0.5*( x3/n2 - f6*x7/n2^3);

df7_x1 = 0.5*(-x4/n2);
df7_x2 = 0.5*(-x5/n2);
df7_x3 = 0.5*(-x6/n2);
df7_x4 = 0.5*(-x1/n2 - f7*x4/n2^3);
df7_x5 = 0.5*(-x2/n2 - f7*x5/n2^3);
df7_x6 = 0.5*(-f7*x7/n2^3);
df7_x7 = 1 + 0.5*(-x3/n2 - f7*x6/n2^3);

phi = [ df1_x1     0        0        0        0        0        0    ;
          0      df2_x2     0        0        0        0        0    ;
          0        0      df3_x3     0        0        0        0    ;
       df4_x1    df4_x2   df4_x3   df4_x4   df4_x5   df4_x6   df4_x7  ;
       df5_x1    df5_x2   df5_x3   df5_x4   df5_x5   df5_x6   df5_x7  ;
       df6_x1    df6_x2   df6_x3   df6_x4   df6_x5   df6_x6   df6_x7  ;
       df7_x1    df7_x2   df7_x3   df7_x4   df7_x5   df7_x6   df7_x7  ];
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C – DATA GENERATION FOR SIMULATION

C1. MAIN PROGRAM

```
%  This program generates measured values of gravity and local
%  magnetic field by integrating Euler equations with a profile for
%  the angular rates.
%  The Euler angles obtained from the euler equations are
%  used to make up a direct cosine matrix and then to rotate
%  values in earth frame into body frame.
%  The measurement noise variance can be chosen.
%  The final values are written to a excel file so that they can be
%  read later by the Extended Kalman filter.
%
%  It calls the routine gene_euler which contains the Euler equations
%  to be used by the ODE45 integration routine.
%
%     By João Luís Marins, Monterey, May 2000.
%

clear; clc;

x = [];

%  Creating angular rates profile (p,q,r)

p1 = [zeros(100,1) ; 0.02*ones(629,1) ; zeros(500,1)];
q1 = [zeros(1229,1)];
r1 = [zeros(1229,1)];

p2 = [zeros(1229,1)];
q2 = [ zeros(100,1) ; 0.02*ones(629,1) ; zeros(500,1)];
r2 = [zeros(1229,1)];

sim_pqr = [p1 q1 r1 ; p2 q2 r2];

%  Define the initial values for the Euler angles
x0 = [0  0  0];

%  Integrate Euler equations
for i=1:2400
   [t1,x1] = ode45('gene_euler', [0;0.125],[ x0';sim_pqr(i,:)']);
   [a,b] = size(x1);
   x0 = x1(a,1:3);
   x = [x ; x0];
end

angles = x;

%  Gravity and local magnetic field in earth frame
g0 = [0 ; 0 ; -1];       h0 = [-0.5 ; 0 ; sqrt(3)/2];

result = [];
```

```
%  Calculate the direct cosine matrix to rotate the vector from
%  earth frame into body frame

for i=1:2400
   phi = angles(i,1);   theta = angles(i,2);    psi = angles(i,3);

   C_phi =   [    1          0           0     ;
                  0       cos(phi)    sin(phi) ;
                  0      -sin(phi)    cos(phi) ];

   C_theta = [ cos(theta)   0      -sin(theta) ;
                  0         1           0     ;
               sin(theta)   0       cos(theta) ];

   C_psi =   [ cos(psi)  sin(psi)        0 ;
              -sin(psi)  cos(psi)        0 ;
                  0         0            1 ];

%  Rotating the vectors
   g1 = C_phi*C_theta*C_psi*g0;
   h1 = C_phi*C_theta*C_psi*h0;

   result = [result ; [ g1' h1' ]];
end

%  Adding measurement noise
meas = [sim_pqr(1:2400,:) result(1:2400,:)];

noise = randn(2400,9);

noise(:,1) = 2e-3*noise(:,1);
noise(:,2) = 2e-3*noise(:,2);
noise(:,3) = 2e-3*noise(:,3);
noise(:,4) = 0.02*noise(:,4);
noise(:,5) = 0.02*noise(:,5);
noise(:,6) = 0.02*noise(:,6);
noise(:,7:9) = 0.02*noise(:,7:9);

meas_noise = meas + noise;

%  Writing data to an excel file
channel = ddeinit('excel', 'dynamic_2.xls');
rc = ddepoke(channel,'r1c1:r2400c9', meas_noise);
rc = ddeterm(channel);

figure(1);plot((1:2400)/8,angles(1:2400,:));
figure(2);plot((1:2400)/8,sim_pqr(1:2400,:));
figure(3);plot((1:2400)/8,result(1:2400,:));
```

C2. EULER EQUATIONS

```
function xprime = gene_euler1(t,x);

%  This function is an ODE file with the Euler equations.
%
%  The main program calls this routine using the ODE45 function
%  in order to integrate the Euler equations to find the three
%  Euler angles.
%
%
%     By João Luís Marins, Monterey, May 2000.


%  Euler equations
xprime = [x(4) + x(5)*sin(x(1))*tan(x(2)) + x(6)*cos(x(1))*tan(x(2));
          x(5)*cos(x(1)) - x(6)*sin(x(1));
          x(5)*(sin(x(1))/cos(x(2))) + x(6)*(cos(x(1))/cos(x(2)));
          0;
          0;
          0];
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Hayward, Roger C., Gebre-Egziabher, Demoz, Powell, J. David. *GPS-Based Attitute for Aircraft.* http://einstein.stanford.edu/gps/ABS/att_for_aircraft_rch1998.html (15 April 2000).
2. Quine, Ben. *Attitude Determination Subsystem.* http://www.atmosp.physics.utoronto.ca/people/ben/pages/spacecraft/Attitude.html (03 June 2000).
3. Leader, D.E., *Kalman Filter Estimation of Underwater Vehicle Position and Attitude Using a Doppler Velocity Aided Inertial Motion Unit*, Engineer Degree Thesis, Massachusets Institute of Technology and Woods Hole Oceanographic Institution, Massachusetts, September 1994.
4. Bachmann, E. R., Duman, I., Usta, U. Y., McGhee, R.B., Yun, X. P., Zyda, M. J., "Orientation Tracking for Humans and Robots Using Inertial Sensors," *Proc. of 1999 Symposium on Computational Intelligence in Robotics & Automation.*
5. Bobick, Nick. *Rotating Objects Using Quaternions.* http://www.gamasutra.com/features/programming/19980703. (26 December 1999).
6. Kuipers, J.B., *Quaternions and Rotation Sequences*, Princeton University Press, Princeton, New Jersey 1999.
7. Savage, P. G., *Strapdown Inertial Navigation Lecture Notes*, Strapdown Associates, Inc., Minnetonka, Minesota 1985.
8. Crassidis, J. L., Markley, F. L., "Predictive Filtering for Attitude Estimation Without Rate Sensors," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 3, May-June 1997, pp. 522-527.
9. Brown, R.G. and Hwang, P.Y.C., *Introduction to Random Signals and Applied Kalman Filtering*, 3rd Edition, John Wiley and Sons, New York 1997.
10. Hagan, Martin T., Demuth, Howard B., Beale, Mark, *Neural Network Design*, PWS Publishing Company, Boston 1995.
11. Arslan, S., *Testing and Evaluation of the Small Autonomous Underwater Vehicle Navigation system (SANS)*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 2000.
12. Stovall, Sherryl L., *Basic Inertial Navigation*, Naval Air Warfare Center Weapons Division, China Lake, California, September 1997.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center..... 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library..... 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code EC..... 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

4. Dr. Xiaoping Yun, Code EC/Yx..... 3
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

5. Eric Bachmann, Instructor, Code CS/Bc..... 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5118

6. Dr. Robert G. Hutchins..... 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5121

7. Dr. Robert B. McGhee, Code CS/Mz..... 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5118

8. LCDR João Luís Marins..... 1
   Rua Salvador de Mendonça, 104 / 303
   Rio Comprido, Rio de Janeiro, RJ
   Brazil, CEP: 20261-030

9. Instituto de Pesquisas da Marinha..... 1
Rua Ipiru, 2
Jardim Guanabara, Ilha, Rio de Janeiro, RJ
Brazil, CEP: 21931-090

10. Coronel Osny Lisboa..... 1
Instituto de Estudos Avançados
Rodovia dos Tamoios, Km 5,5
Caixa Postal 6044
São José dos Campos, SP
Brazil, CEP: 12231

11. Major Carlos Kasemodel..... 1
Instituto de Aeronáutica e Espaço
Praça Mal. Eduardo Gomes, 50
São José dos Campos, SP
Brazil, CEP: 12228-904