2010:016

MASTER'S THESIS

Design and Implementation of Attitude Determination Algorithm for the Cubesat UWE-3

Sajid Ghuffar

Luleå University of Technology

Master Thesis, Continuation Courses
Space Science and Technology
Department of Space Science, Kiruna

2010:016 - ISSN: 1653-0187 - ISRN: LTU-PB-EX--10/016--SE

Julius-Maximilians-Univesität Würzburg

Lehrstuhl für Informatik VII

Institut für Robotik und Telematik

Luleå Tekniska Universitet

Institutionen för rymdvetenskap

Kiruna

Design and Implementation of Attitude Determination Algorithm for the Cubesat UWE-3

Master Thesis in the subject

MSc. Space Science and Technology

Presented By

Sajid Ghuffar

Julius-Maximilians-Univesität Würzburg

Lehrstuhl für Informatik VII

Institut für Robotik und Telematik

Luleå Tekniska Universitet

Institutionen för rymdvetenskap

Kiruna

Design and Implementation of Attitude Determination Algorithm for the Cubesat UWE-3

Master Thesis in the subject

MSc. Space Science and Technology

Presented By

Sajid Ghuffar

Born on 23.10.1984, Rawalpindi, Pakistan

Completed at

University of Wuerzburg

Faculty of Computer Science

Institute of Robotics and Telematics

Supervisors:

Prof. Dr. Klaus. Schilling

Dr. Thomas Kuhn

Delivery Date of Thesis:

12th, October 2009

Acknowledgments

First of all I want to thank Prof Dr. Klaus Schilling for providing me the possibility to do a very interesting Master thesis work. This project allowed me to have a good insight in the development of a Picosatellite while gaining a lot of knowledge and skill in the area of small satellites.

Sincere thanks to Dr Thomas Kuhn from Luleå University of Technology for supervising the thesis work remotely and assisting me in every aspect of the project.

Furthermore a lot of sincere thanks to my instructors Karthik Ravandoor and Stephan Busch for helping me in every moment of the project. Without their supervision, help and technical support I would not have been able to achieve the goals of this thesis work. Special thanks to Adam Cseh for his continued support and guidance through the course of the project. I like to pay special gratitude to Stephan Busch and Adam Cseh for the support they provided me during my second year of Master studies. Also sincere thanks to Prof Dr. Hakan Kayal for being so helpful and kind to me.

I like to thank Impulse Accelerated Technologies and Mr. Hans-Jürgen Schwender for providing their software and support, which was vital in the success of this thesis project.

I also like to thank all my friends especially Kashif Gulzar, Hamza Baig, Chen Xi and Muhammad Zubair for their assistance during my Master studies and give my appreciation for the Erasmus Mundus programme through which I got the opportunity to study SpaceMaster course.

Finally I thank my parents and my family for giving me a lot of support and advice while I was studying abroad and guiding me all the way in my life.

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Würzburg, the 12th, October 2009 _____

(Sajid Ghuffar)

Abstract

This master thesis project describes the design, simulation and implementation of Attitude Determination System for University of Würzburg Experimental Satellite UWE-3. Following the success of the Cubesat program at University Würzburg and considering the objectives of future space missions there is an imminent need for a highly precise attitude determination system. The objective of this thesis work is the realization of such a highly precise attitude determination system which can lay a foundation for accomplishing complex space oriented tasks for future missions at University of Würzburg.

This report investigates various attitude estimation methods and presents a software model for the implementation of these attitude estimation methods. These software models are then simulated and verified and based on the simulation results, the best attitude estimation method is selected which is ideally suited to the project needs and requirements.

After choosing the best attitude estimation method an analysis on potential hardware platforms is done. Several hardware platforms and design techniques are studied, simulated and evaluated for most feasible hardware platform, which is best suited for a Picosatellite mission. After the analysis a single hardware platform is recommended for the on board implementation of the attitude determination algorithm. Finally some future work and extension to the work is outlined.

Contents

1 Introduction

The faculty of Computer Science at University Würzburg is actively taking part in projects related to Pico and Nano satellites and space exploration. University of Würzburg Experimental Satellite UWE-1 was the first in series of Picosatellite missions undertaken in University Würzburg. UWE-1 was the first Picosatellite from a German University, which was successfully launched on the 27th of October 2005 from Plesetsk, Russia. Main objective of the mission was to test communication protocols like TCP/IP in space where major problems of path delays and low communication bandwidth typically arrive.

UWE-2 is the second Picosatellite developed at University Würzburg which was launched on 23rd September, 2009 from Sriharikota, India. The main objective of the UWE-2 was the demonstration of an attitude determination system on a Picosatellite. Signals from UWE-2 have been decoded all over the world.

UWE-3 is the third Satellite in the series of UWE missions undertaken at department of Computer Science University Würzburg. The probable launch date for UWE-3 mission is in End of year 2010. This project work is related to the attitude determination system of UWE-3.

1.1 Thesis Definition

The attitude determination system is a vital part of any satellite. Tasks like communication, Earth observation, space exploration all require a very precise attitude determination system on board. Developing an accurate attitude determination system on a Picosatellite is quite a challenge considering limited amount of resources available on board a Picosatellite.

The aim of this thesis work is to develop a precise attitude determination system (ADS) for UWE-3 Picosatellite. This project is in continuation of the work done by Karthik Ravandoor [1] and Oliver Kurz [2], who developed the attitude determination system of UWE-2. In the following step by step objectives and advancement of the thesis work is presented.

- Development and simulation of the reference mathematical models for an ADS and simulation of ADS sensors data.
- Investigation, design and simulation of attitude estimation algorithms.
- Investigation of hardware platforms for on board implementation of the ADS algorithm.

- Implementation, simulation and analysis of the ADS algorithm on the selected hardware platforms.

Figure 1: View of UWE-3 (Courtesy: UWE-3 Team)

1.2 Background

Small satellites have opened a new era of space research with numerous universities and institutions vibrantly taking part in exploring newer grounds related to space explorations. Due to lower development and launch cost of Pico, Nano and Microsatellites students now have the opportunity to work and realize their own space related projects within the scope of a small satellite project. One of the main objectives of small satellite missions is to allow students to learn while promoting research activities within the universities.

The UWE-3 project is the continuation of the small satellite projects undertaken at department of Computer Science, University Würzburg. In UWE-3 predecessor i.e. UWE-2 data from attitude

determination sensors were sent to ground station for attitude estimation of the satellite because there was no on board implementation of the attitude determination algorithm. Mission objectives of UWE-3 include attitude control by a micro reaction wheel. Therefore it is imperative to have precise attitude determination software on board.

The sensors on board the UWE-3 satellite will be 3-axis gyros, Magnetometers and the Sun Sensors. New sensor technologies are in the process of evaluation at the department to achieve best possible attitude estimation. Following is little description of the attitude determination sensors:

Gyroscopes are essential parts of an ADS providing feedback of satellite turn rates. State of the art MEMS (Micro Electro Mechanical Systems) technology and miniaturized integrated circuit technology has resulted in small size and decreased power consumption in gyros and is therefore feasible for integration into Picosatellite. While gyroscopes don't suffer from problems related to orbit or external dependencies e.g. Sun sensors, star sensors, gyro bias induces significant error in measurements of satellite angular velocity. Therefore in a highly accurate ADS gyro bias needs to be estimated.

Sun Sensors are used to measure the angle of the Sun with respect to the satellite to obtain the unit vector of Sun position in satellite body frame. Sun sensors will be placed on each of six sides of the Cubesat to measure angle in two directions. By combining the output of the Sun sensors on each side, Sun position vector can be calculated.

Magnetometers measure the Earth magnetic field and output 3-axis information about the direction of the magnetic field vector. Due to noise sources present in the satellite magnetometer measurements will be less accurate than the Sun sensors measurements. Noise sources in the satellite come from permanent magnets which are used to passively stabilize the satellite and the electronics used in the satellite. The noise from the permanent magnets is static and therefore it is easy to determine and compensate for, while the noise from on board electronics is unpredictable and changes over time, hence it is difficult to estimate this noise.

1.3 Report Outline

This report is segmented into the following chapters.

Introduction provides project definition, objectives and the scope of the thesis project.

Definitions and Notations provides descriptions about the fundamentals of ADS which are necessary in understanding the concept of the ADS algorithm.

Reference Mathematical Models and Orbit Propagation covers the mathematical models used for the Sun and magnetic field models and the algorithm used for orbit generation.

Attitude Representation discusses the various methods of representing the attitude of a satellite which is the basic building block in understanding the attitude estimation algorithms.

Attitude Estimation investigates different attitude estimation algorithms and techniques and their Matlab based design.

Simulation and Results first discusses the simulation of the first the reference mathematical models, then the simulation of various attitude estimation techniques and finally the results obtained from implementation and analysis on hardware platforms.

Software Implementation discusses various modules and functions of the C implementation.

Hardware Implementation discusses different hardware platforms and there analysis on the basis of project requirements.

Conclusion concludes the thesis work with final results of the best attitude determination algorithm and the most feasible hardware platform along with the future work.

2 Reference Frames

2.1 Earth Centered Inertial Frame (ECI)

Earth Centered Inertial Frame is a group of reference frames whose origin is fixed with the center of the Earth and the frame remains fixed with respect to inertial space. In ECI frame the equations that describe the orbital motion are simpler. ECI is also useful for specifying the direction of the celestial objects [3].

The intersection of the Earths equatorial plane and Earth's orbit plane around Sun is used as principal direction in ECI frames. This point of intersection is known as vernal equinox. The origin of the ECI frame is in the center of the Earth, the X-axis is in the direction of vernal equinox while the Z-axis is along the Earth's rotation axis pointing toward North Pole as shown in Figure 2: Earth Centered Inertial Frame. Gravitational force from moon and other celestial objects causes Earth's spin axis to wobble as a result Earth equatorial plane is not fixed. The orbit of the Earth around the Sun is also changing; hence the vernal equinox also changes. Therefore when defining the ECI frame a particular epoch date is specified. When the short-term periodic oscillations of this motion are averaged out, they are considered "mean" as opposed to "true" values [3]. On the basis of true and mean values of equator and equinox following ECI frames are defined and will be used further in the report. More details about these reference frames can be found in [4].

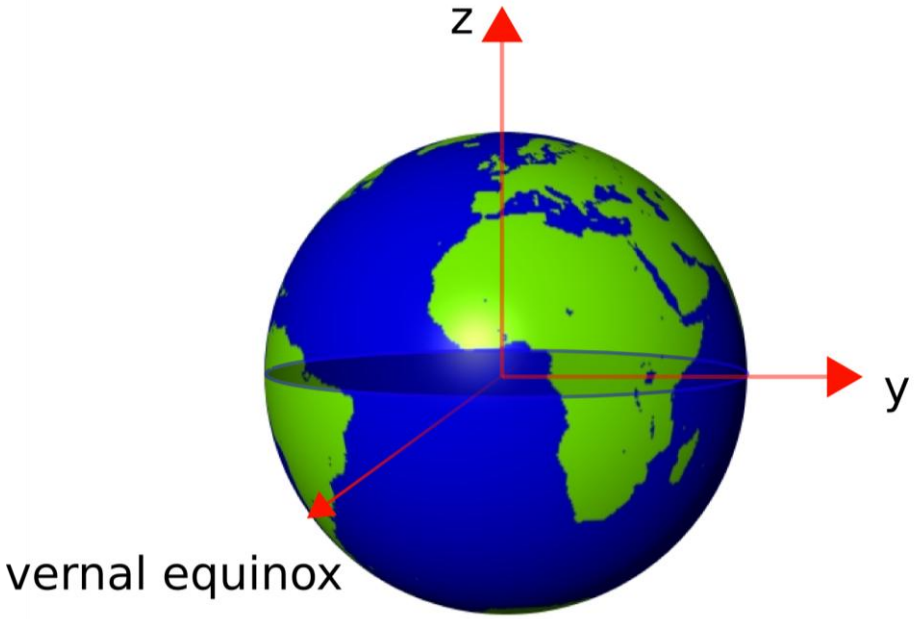

Figure 2: Earth Centered Inertial Frame [1]

2.1.1 J2000

One commonly used ECI frame is defined with the Earth's mean equator and equinox at 12:00 Terrestrial Time on 1 January 2000. It can be referred to as J2000. The X-axis is aligned with the mean equinox. The Z-axis is aligned with the Earth's spin axis pointing towards celestial North Pole. The Y-axis is rotated by 90^{o} East on the Celestial equator about Z-axis.

2.1.2 MOD

A Mean of Date (MOD) frame is defined using the mean equator and equinox on a particular date. The direction of the X axis is defined by the mean vernal equinox and the Z axis is defined by the mean spin axis of the Earth at the time of the state vector. The term 'mean' indicates that precession is accounted for but nutation is not [5].

2.1.3 TEME

The ECI frame used for the NORAD Two-Line elements is sometimes called True Equator, Mean Equinox (TEME) although it does not use the conventional mean equinox.

2.2 Earth Centered Earth Fixed Frame (ECEF)

Earth Centered Earth Fixed is a Cartesian coordinate system whose Z-axis is along Earth rotation axis and X-axis is at 0° latitude, 0° longitude. Hence ECEF rotates with the Earth and therefore have its name fixed because all the points on Earth surface remain fixed. The ECEF frame rotates along with the Earth with an angular velocity of 7.2921×10^{-5} rad/sec.

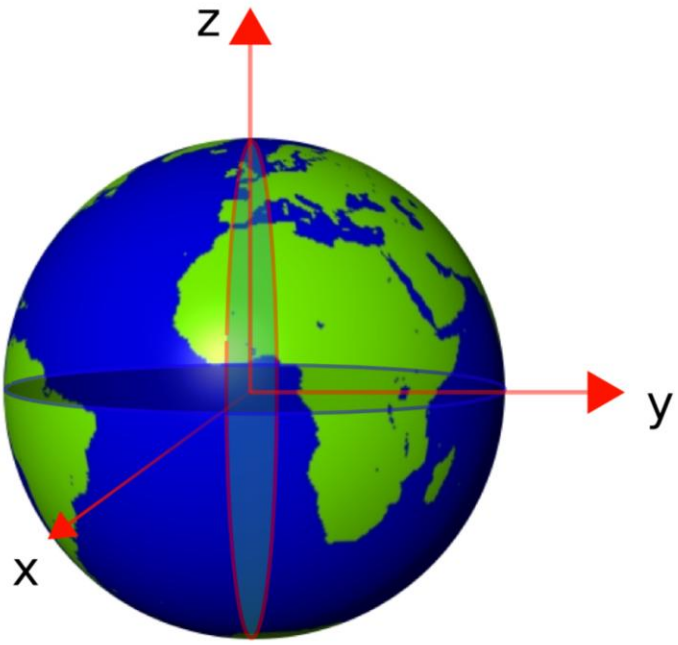

Figure 3: ECEF Frame

2.3 Body Frame

This frame is fixed with the body of the satellite and its origin is placed at the center of mass of the satellite. It is this frame which determines the attitude of the satellite. The ADS of the UWE-3 Picosatellite computes quaternion between the body frame and the ECI frame. This frame is denoted as “b” and the diagram below shows the body frame with respect to different sides of the satellite.

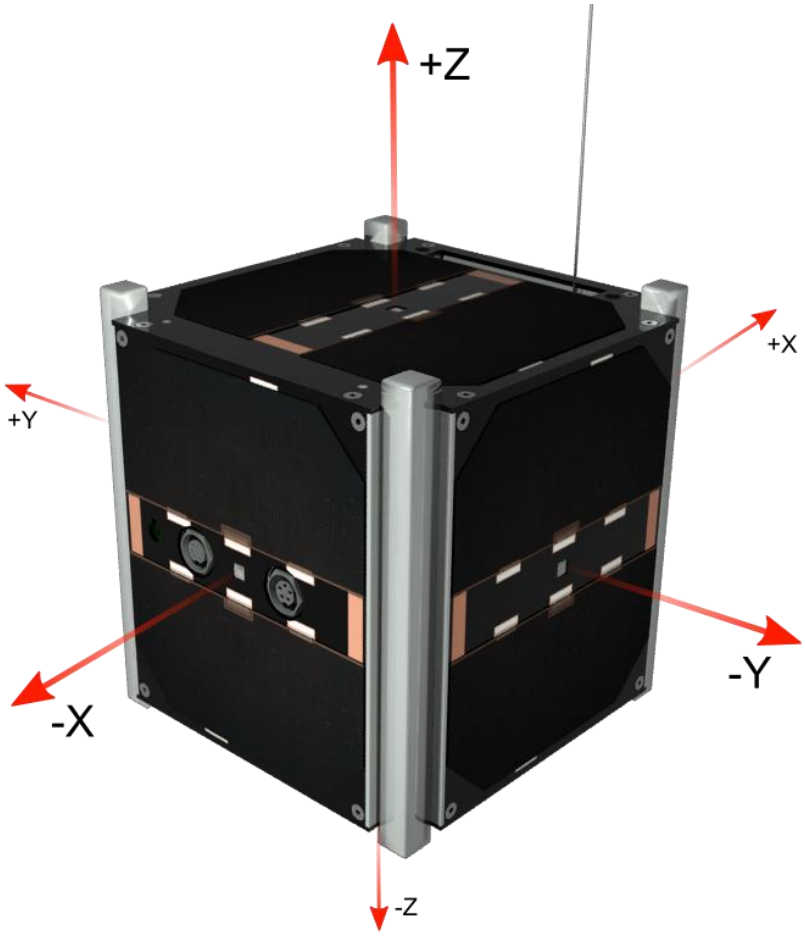

Figure 4: Satellite Body Frame

3 Attitude Representation

This chapter briefly defines and discusses various methods of representing satellite attitude which are fundamental concepts in understanding the attitude determination algorithms.

3.1 Rotation Matrix

Consider a rigid body in space and let an orthogonal frame be attached to the rigid body, the problem of attitude determination is to represent the orientation of this orthogonal frame with respect to some reference orthogonal frame. In UWE-3 attitude determination system reference frame will be Earth Centered Inertial frame and the orthogonal frame attached to the satellite is the satellite body frame.

A rotation matrix is a 3×3 matrix which specifies the rotation between the reference frame and the body frame. It is also known as direction cosine matrix because it is the direction cosine of the one axis in other frame [6].

$$\vec{\boldsymbol{r}}_{\boldsymbol{2}} = R\,\vec{\boldsymbol{r}}_{\boldsymbol{1}} \tag{1}$$

$$\boldsymbol{R} = \begin{bmatrix} x_2.x_1 & x_2.y_1 & x_2.z_1 \\ y_2.x_1 & y_2.y_1 & y_2.z_1 \\ z_2.x_1 & z_2.y_1 & z_2.z_1 \end{bmatrix} \tag{2}$$

where $\vec{\boldsymbol{r}}_{\boldsymbol{1}} = [x_1, y_1, z_1]^T$ and $\vec{\boldsymbol{r}}_{\boldsymbol{2}} = [x_1, y_1, z_1]^T$ An example of rotation $\propto$ around X-axis is:

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos \propto & sin \propto \\ 0 & -sin \propto & cos \propto \end{bmatrix} \tag{3}$$

The rotation matrix is orthogonal which means $\boldsymbol{A}\boldsymbol{A}^T = 1$. This implies that $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$. For an orthogonal matrix the columns (or rows) of the matrix are orthogonal unit vectors.

3.1.1 Rotation from ECEF to ECI

The rotation from ECEF to ECI frame is along the z-axis and its equal to angle $\propto = \omega_e t$, where ω_e is the Earth's angular rate and t is time elapsed since the two frames coincided. This transformation is used to transform the Earth Magnetic Field vector from ECEF to ECI, because

the IGRF model used gives magnetic field vector in ECEF coordinates. This rotation is expressed as following rotation matrix.

$$\boldsymbol{R}_E^I = \begin{bmatrix} cos \propto & -sin \propto & 0 \\ sin \propto & cos \propto & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

The inverse of this rotation matrix is also to transform output of SGP4 algorithm from ECI to ECEF which is then used as input to IGRF model to calculate magnetic field vector.

3.1.2 Rotation from MOD to ECI

This rotation is used to transform a vector in Mean of Date coordinate system to ECI coordinate system. In Mean of Date reference frame precession effects are accounted but the nutation effects are not taken into account. The Sun reference model outputs vector in MOD which is then transformed into ECI. This rotation matrix is referenced from Vallado [4].

3.2 Quaternion

Quaternions provide a convenient way of representing orientation and rotation of objects in three dimensions. Compared to rotation matrices they are more efficient because they do not involve trigonometric operations and only involve algebraic manipulation. Quaternions have been used in a lot of application e.g. robotics, satellite, graphics, navigation etc. A Quaternion is represented as:

$$\boldsymbol{q} = q_4 + iq_1 + jq_2 + kq_3 \quad (5)$$

$$i^2 = j^2 = k^2 = ijk = -1 \quad (6)$$

$$ij = -ji = k \quad (7)$$

$$jk = -kj = i \quad (8)$$

$$ki = -ik = j \quad (9)$$

Quaternion represents a very easy way of representing rotations. Any rotation in three dimensions can be realized as a single rotation about some axis.

$$\boldsymbol{q} = \cos\left(\alpha/2\right) + \vec{\boldsymbol{u}}\sin\left(\alpha/2\right) \quad (10)$$

Where $\vec{\boldsymbol{u}}$ is a unit axis and α is the angle of rotation along this axis.

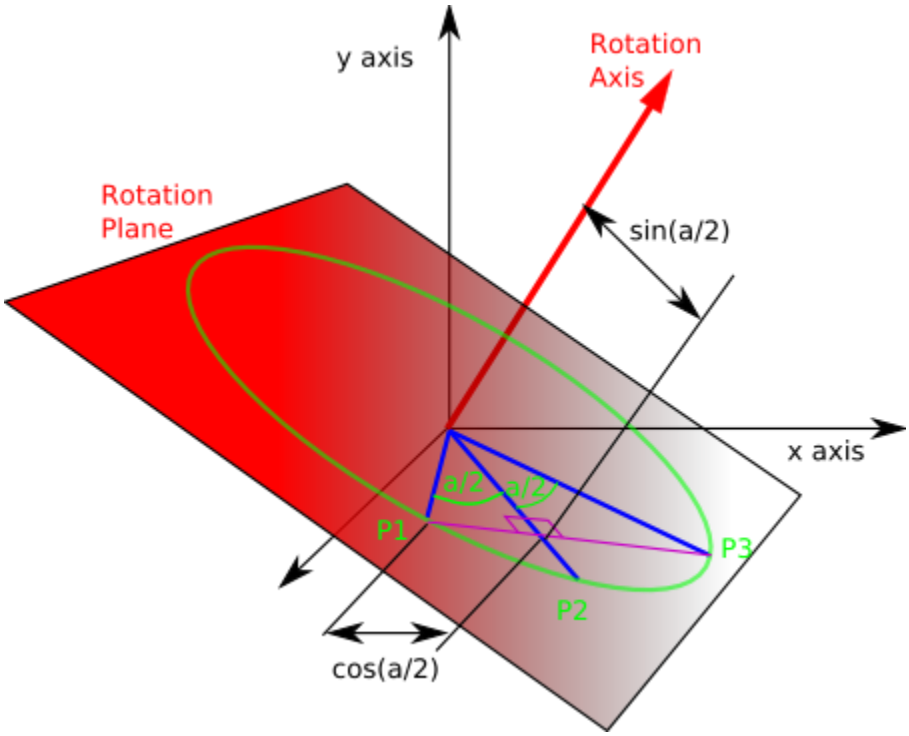

Figure 5: Graphical representation of quaternion [7]

Quaternion to rotation matrix conversion is given by following equation:

$$\boldsymbol{R} = \begin{bmatrix} {q_1}^2 - {q_2}^2 - {q_3}^2 + {q_4}^2 & 2(q_1q_2 + q_4q_3) & 2(q_1q_3 - q_4q_2) \\ 2(q_1q_2 - q_4q_3) & -{q_1}^2 + {q_2}^2 - {q_3}^2 + {q_4}^2 & 2(q_2q_3 + q_4q_1) \\ 2(q_1q_3 + q_4q_2) & 2(q_2q_3 - q_4q_1) & -{q_1}^2 - {q_2}^2 + {q_3}^2 + {q_4}^2 \end{bmatrix} \quad (11)$$

If **q** is the current satellite quaternion and $\mathbf{q}'$ is the desired quaternion then quaternion error is calculated as follows:

$$\boldsymbol{q_e} = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_2 & q_4 & q_3 \\ -q_1 & -q_1 & -q_3 & q_4 \end{bmatrix} \cdot \begin{bmatrix} -q_1{}' \\ -q_2{}' \\ -q_3{}' \\ q_4{}' \end{bmatrix} \quad (12)$$

Given the quaternions of two successive rotations **q** and $\mathbf{q}'$ the resultant quaternion is calculated as follows [8]:

$$\boldsymbol{q}'' = \begin{bmatrix} q_4' & q_3' & -q_2' & q_1' \\ -q_2' & q_4' & q_1' & q_2' \\ q_2' & -q_1' & q_4' & q_2' \\ -q_1' & -q_2' & -q_3' & q_4' \end{bmatrix} . \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{13}$$

The above mentioned formula will be used in Kalman filter update step to update the estimated quaternion.

4 Reference Mathematical Models and Orbit Propagator

This chapter presents the mathematical models which have been used in the Attitude determination system for UWE-3. These models are magnetic field model and the Sun vector model.

4.1 Magnetic Field Model

UWE-3 will have magnetometers on board to give 3-axis magnetic field information. To calculate the required attitude quaternion of the satellite a reference magnetic field vector in inertial frame needs to be estimated. This estimation is done on board using the model for Earth magnetic field described below. The input to this model is the satellite's position which is obtained by the satellite orbit and position estimator which is also running on board the satellite.

The International Geomagnetic Reference Field **(**IGRF) is a global model of the geomagnetic field. It allows spot values of the geomagnetic field vector to be calculated anywhere from the Earth's core out into space. The IGRF is generally revised once every five years by a group of modelers associated with the International Association of Geomagnetism and Aeronomy (IAGA) [9]. The coefficients used in our model are of year 2005 (valid up to year 2010) according to the 10^{th} generation IGRF model valid from 1900 till 2015.

The IGRF is a series of mathematical models of the Earth's main field and its annual rate of change (secular variation). In source-free regions at the Earth's surface and above, the main field, with sources internal to the Earth, is the negative gradient of a scalar potential *V* which can be represented by a truncated series expansion [10,11].

$$V(r,\varphi,\theta) = a\sum_{n=1}^{L}\sum_{m=0}^{n}\left(\frac{a}{r}\right)^{l+1}\left(g_n^m cosm\varphi + h_n^m sinm\varphi\right)P_n^m(cos\theta) \qquad (14)$$

where *r* is radial distance from the Earth's center, *L* is the maximum degree of the expansion, φ is East longitude, θ is colatitude (the polar angle), a is the Earth's radius, g_n^m and h_n^m are Gauss coefficients, and $P_n^m(cos\theta)$ are the Schmidt normalized associated Legendre functions of degree n and order *m* [3]. The magnetic field calculated according to the IGFR is in ECEF coordinate system which is then transformed to ECI coordinate system.

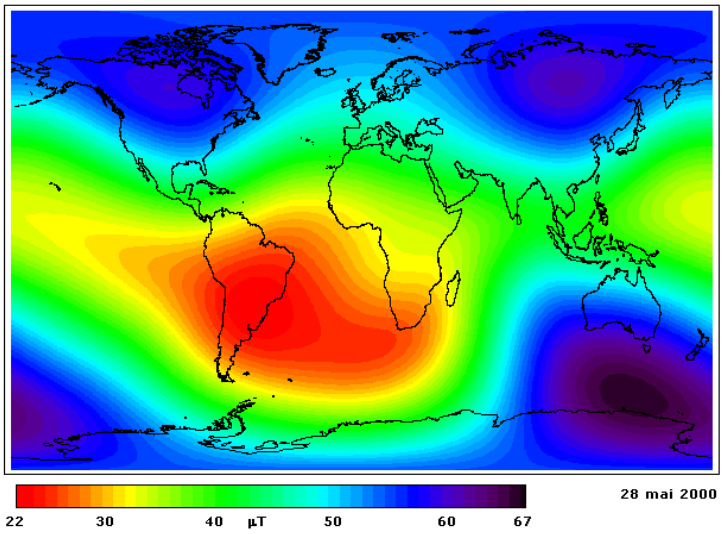

Figure 6: Earth's Magnetic Field [Source: www.astrosurf.com]

4.2 Sun Vector Model

Just like the attitude estimator requires a reference magnetic field vector along with the measured magnetic field from magnetometer, it also needs a corresponding Sun vector along with the measured Sun vector from the on board Sun sensors. The Sun vector model used in our algorithm is referenced from [4]. The resultant Sun position vector is in ECI frame of reference with an accuracy of 0.01^{o} and it is valid from 1950 to 2050 because of the truncation of the expansions [4]. The input required for the Sun position vector is the time in Julian date. More details of the model can be found in [4]. Matlab and C algorithm for this model can be found on celestrak website [12].

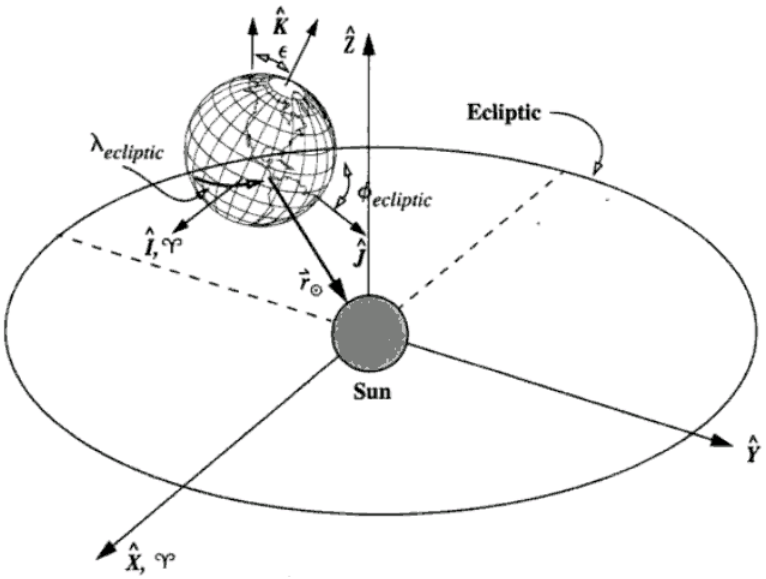

Figure 7: Geometry for the Sun Position Vector [Source: David Vallado]

4.3 Orbit Propagation

North American Aerospace Defense Command (NORAD) maintains general perturbation element set on all the resident space objects. These element sets are periodically updated so as to maintain a reasonable prediction capability on all space objects. The NORAD element sets are "mean" values obtained by removing periodic variations in a particular way. In order to obtain good predictions, these periodic variations must be reconstructed (by the prediction model) in exactly the same way they were removed by NORAD [13].

SGP4 (Simplified General Perturbations Satellite Orbit Model 4) is a NASA/NORAD algorithm of calculating near Earth satellites. A satellite with an orbital period less than 225 minutes is termed as near Earth satellite. NORAD two line element sets are generated by using SGP4 algorithm for near Earth satellites [3].

SGP4 algorithm is outlined in SpaceTrack report published by NORAD. Input for this algorithm is the Two Line Element (TLE) dataset and the propagation time (in minutes). The output of the algorithm is the position and velocity vector in TEME (True Equator Mean Equinox) reference frame.

5 Attitude Estimation

Three different attitude determination models have been studied namely Enhanced Triad Algorithm (ETA), Enhanced q-method Algorithm (EQA) and Kalman Filter [14]. All of these models use the combination of the 3 on board sensors i.e. Sun sensors, magnetometer and the 3-Axis gyros. All these models are explained with detail in next sections. Results from these different methods have been compared with the results from Satellite Tool Kit (STK) to compute the accuracy of each algorithm. These methods have been referenced from Crassidis [14] and detailed explanation can be found in [14,17].

5.1 Enhanced Triad Algorithm (ETA)

The attitude determination problem can be considered as determining the direction cosine matrix or the rotation matrix or equivalently the quaternion based on comparing the vectors measured in the body coordinate system with the known vectors in a reference frame e.g. ECI coordinate system. In this thesis work the known reference vectors are expressed in ECI coordinate system.

The TRIAD algorithms determines attitude based on two vector measurements

$$\vec{\boldsymbol{b}}_1 = \boldsymbol{A}\vec{\boldsymbol{r}}_1 + \vec{\boldsymbol{v}}_1 \tag{15}$$

$$\vec{\boldsymbol{b}}_2 = \boldsymbol{A}\vec{\boldsymbol{r}}_2 + \vec{\boldsymbol{v}}_2 \tag{16}$$

where,

$\vec{\boldsymbol{b}}_1$ and $\vec{\boldsymbol{b}}_2$ are the measured unit vectors in the BFCS

$\vec{\boldsymbol{r}}_1$ and $\vec{\boldsymbol{r}}_2$ are the known unit vectors in the ECI

$\vec{\boldsymbol{v}}_1$and $\vec{\boldsymbol{v}}_1$ are the error vectors from the measurements

A is the rotation matrix from ECI to body frame

Two triads constructed from pair of orthonormal vector are **R** and **S** as given below [15]:

$$\boldsymbol{R} = \left[\vec{\boldsymbol{r}}_1 \quad \frac{\vec{\boldsymbol{r}}_1 \times \vec{\boldsymbol{r}}_2}{|\vec{\boldsymbol{r}}_1 \times \vec{\boldsymbol{r}}_2|} \quad \vec{\boldsymbol{r}}_1 \times \frac{\vec{\boldsymbol{r}}_1 \times \vec{\boldsymbol{r}}_2}{|\vec{\boldsymbol{r}}_1 \times \vec{\boldsymbol{r}}_2|}\right] \tag{17}$$

$$\boldsymbol{S} = \left[\vec{\boldsymbol{b}}_1 \quad \frac{\vec{\boldsymbol{b}}_1 \times \vec{\boldsymbol{b}}_2}{|\vec{\boldsymbol{b}}_1 \times \vec{\boldsymbol{b}}_2|} \quad \vec{\boldsymbol{b}}_1 \times \frac{\vec{\boldsymbol{b}}_1 \times \vec{\boldsymbol{b}}_2}{|\vec{\boldsymbol{b}}_1 \times \vec{\boldsymbol{b}}_2|}\right] \tag{18}$$

Then the estimate of the rotation matrix **A** by the TRIAD algorithm is given by

$$\boldsymbol{A}^{TRIAD} = \boldsymbol{S}R^T \tag{19}$$

The estimated attitude matrix is always exact for the first vector measurement; hence the first vector component should always be the most accurate sensor. The TRIAD method fails when there is only one sensor measurement available e.g. the Sun sensor reading (during Eclipse period). Also when the two vector measurements are co-aligned the TRIAD algorithm fails.
The Enhanced TRIAD method as described in [14] makes use of the gyro reading to overcome the problems of co-alignment and one sensor reading by weighted quaternion combination. Following equation shows the Enhanced TRIAD method.

$$\boldsymbol{q} = (1-\propto)\boldsymbol{q}_p + \propto \boldsymbol{q}_{TRIAD} \tag{20}$$

where $\boldsymbol{q}$ is the quaternion estimate by ETA, $\boldsymbol{q}_p$ is computed from the quaternion propagation equation by using gyro measurements, $\boldsymbol{q}_{TRIAD}$ is the quaternion determined by TRIAD algorithm and $\propto$ is the weighing factor given by:

$$\propto = (1 - |\vec{\boldsymbol{r}}_{\boldsymbol{1}}.\vec{\boldsymbol{r}}_{\boldsymbol{2}}|^2)\propto_0 \tag{21}$$

The constant $\propto_0$ is a gain factor. The filter gain $\propto$ in above Equation is automatically adjusted to accommodate periods of vector co-alignment, as the vectors become co-aligned, the gain approaches zero. Also, $\propto_0$ is set to zero when only one measurement set is available [14].
The quaternion q_p is calculated using the zeroth order quaternion propagation equation [16]:

$$\boldsymbol{q_p} = \left(\cos\left(\frac{|\boldsymbol{\omega}|}{2}\Delta t\right).\boldsymbol{I}_{4\times4} + \frac{1}{\boldsymbol{\omega}}sin\left(\frac{|\boldsymbol{\omega}|}{2}\Delta t\right).\boldsymbol{\Omega}(\boldsymbol{\omega})\right).\boldsymbol{q}^{-} \tag{22}$$

where $\boldsymbol{q}^{-}$ is the quaternion computed in previous step and the matrix $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is given by

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{23}$$

The Enhanced TRIAD algorithm is still prone to errors like sensors biases which the algorithm cannot estimate. So the sensor biases need to be estimated prior to have the best attitude solution.

5.2 Enhanced q-method Algorithm (EQA)

Principle of Enhanced q-method Algorithm is same as ETA except that instead of TRIAD q-Method is used for quaternion estimation. Davenport q-Method provided the first useful solution of Wahba's problem for spacecraft attitude determination [14]. The Wahba's problem is given as

$$L(A) = \frac{1}{2}\sum_i a_i \left|\vec{\boldsymbol{b}}_{\boldsymbol{i}} - \boldsymbol{A}\vec{\boldsymbol{r}}_{\boldsymbol{i}}\right|^2 \qquad \text{i=1....N} \tag{24}$$

where $\vec{\boldsymbol{b}}_{\boldsymbol{i}}$ is the vector in body fixed coordinate system, $\vec{\boldsymbol{r}}_{\boldsymbol{i}}$ is the known vector in reference frame, L(A) is the cost function and **A** is the estimate of the rotation matrix which minimizes the cost function L(A).

The full derivation of the q-method can be found in Crassidis [17]. The q-method identifies the best estimate for the quaternion as the normalized eigenvector associated with the maximum eigen value for the matrix **K**.

$$\boldsymbol{K}\boldsymbol{q} = \lambda \boldsymbol{q} \tag{25}$$

A representation for K, occasionally referred to as the Davenport matrix, is:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{S} - s\boldsymbol{I}_{3\times 3} & \boldsymbol{Z} \\ \boldsymbol{Z} & s \end{bmatrix} \tag{26}$$

where the matrix **S,** matrix **Z** and the scalar quantity s is given in the equations below:

$$\boldsymbol{Z} = \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{12} \end{bmatrix} \tag{27}$$

where B_{ij} (i,j=1..3) are the components of the matrix B given by

$$\boldsymbol{B} = \sum_{i=1}^{N} {\sigma_i}^{-2} \vec{\boldsymbol{b}}_i \vec{\boldsymbol{r}}_i^{\,T} \tag{28}$$

where σ_i is the noise covariance of the vector measurements.

$$s = Tr(\boldsymbol{B}) \tag{29}$$

$$\boldsymbol{S} = \boldsymbol{B} + \boldsymbol{B}^T \tag{30}$$

The Enhanced q-method makes use of the gyro reading to overcome the problems of co-alignment and one sensor reading by weighted quaternion combination. Following equation shows the Enhanced q-method.

$$\boldsymbol{q} = (1-\propto)\boldsymbol{q}_p + \propto \boldsymbol{q}_q \tag{31}$$

where $\boldsymbol{q}$ is the quaternion estimate by EQA, $\boldsymbol{q}_p$ is computed from the quaternion propagation equation by using gyro measurements, $\boldsymbol{q}_q$ is the quaternion determined by q-method. $\propto$ and $\boldsymbol{q}_p$ are computed in the same manner as given in equations [21 and 22].

5.3 Kalman Filter

The Kalman Filter utilizes a dynamic model for the time development of the system and a model of sensor measurements to compute the best estimate of the system state using a linear estimator based on present and past measurements. Kalman filter is highly suited for on board estimation of the attitude. More information about Kalman Filtering related to satellites can be found in Lefferts [18].

In the following paragraph, general Equations for a Kalman Filter are given. **P** is the covariance matrix for the states of the Kalman filter, **K** is the Kalman gain, **R** is the measurement noise, **Q** is the process noise covariance, **H** is the sensitivity matrix, Φ is the state transition matrix, $\mathbf{\Gamma}_k$ is termed as control matrix, $\mathbf{u}_k$ is the control vector $\hat{\boldsymbol{x}}$ is the state of the filter and $\tilde{\boldsymbol{y}}$ is the measurement [14].

Filter starts with an initial estimation of the state and the covariance matrix $\hat{\boldsymbol{x}}_0^+$and $\boldsymbol{P}_0^+$ and estimate the state and covariance to time step 1. Then the following steps follow:

- Kalman Gain calculation

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_K{}^T \left[\boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_K{}^T + \boldsymbol{R}_k\right]^{-1} \tag{32}$$

- Covariance update

$$\boldsymbol{P}_k^+ = [I - \boldsymbol{K}_k \boldsymbol{H}_K]\boldsymbol{P}_k^- \tag{33}$$

- State estimate update from the measurement $\tilde{y}$

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\tilde{\boldsymbol{y}} - \boldsymbol{H}_K\hat{\boldsymbol{x}}_k^-) \quad (34)$$

- State propagation and the covariance estimate for the next time step.

$$\hat{\boldsymbol{x}}_{k+1}^- = \boldsymbol{\Phi}_{\mathrm{k}}\hat{\boldsymbol{x}}_k^+ + \boldsymbol{\Gamma}_{\mathrm{k}}\mathbf{u}_{\mathrm{k}} \quad (35)$$

$$\boldsymbol{P}_{k+1}^- = \boldsymbol{\Phi}_{\mathrm{k}}\boldsymbol{P}_k^+{\boldsymbol{\Phi}_{\mathrm{k}}}^{\mathrm{T}} + \mathbf{Q}_{\mathrm{k}} \quad (36)$$

- Loop continued

5.4 Isotropic Kalman Filter (IKF):

In this section, the equations for the IKF are shown. The state vector consists of an incremental quaternion and gyro bias, hence the gyro bias is also estimated along with the attitude quaternion. This is shown in equation below where q is the unit quaternion with 4 components and b is the gyro bias for 3-axis.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ -- \\ \boldsymbol{b} \end{bmatrix} \quad (37)$$

The true angular velocity is modeled by following equation

$$\boldsymbol{\omega} = \boldsymbol{\omega}_g + \boldsymbol{b} + \boldsymbol{\eta}_1 \quad (38)$$

Where ω is the true angular velocity, ω_g is the velocity given by the on board gyros and b is the gyro drift. The bias drift noise is modeled as

$$\dot{\boldsymbol{b}} = \boldsymbol{\eta}_2 \quad (39)$$

where η_1 and η_2 are modeled as guassian white noise process. The assumed measurement (residual) in the IKF is given by:

$$\tilde{\vec{\boldsymbol{z}}} = \tilde{\vec{\boldsymbol{u}}} \times \hat{\vec{\boldsymbol{u}}} \quad (40)$$

Where $\hat{u}$ is the expected magnetometer or the Sun Sensor vector, calculated by rotating the inertial reference vectors defined by estimated quaternion and $\tilde{u}$ is the measured unit vector in

body frame [19]. UWE-3 has both the Sun sensor and the magnetometer on board so the residual is calculated by weighing the residuals from both sensors based on sensor noise covariance [20].

$$resd = \frac{{\sigma_t}^2}{{\sigma_{sun}}^2} \times resd_{sun} + \frac{{\sigma_t}^2}{{\sigma_{mag}}^2} \times resd_{mag} \tag{41}$$

where σ_t is total noise covariance, given by

$$\frac{1}{{\sigma_t}^2} = \frac{1}{{\sigma_{sun}}^2} + \frac{1}{{\sigma_{mag}}^2} \tag{42}$$

When only one sensor reading is available then noise covariance value can be set to infinity in the above mentioned formula hence only the available sensor measurement will be used to calculate measurement residual. The Isotropic Kalman Filter is derived by making an approximation of the sensitivity matrix **H**. Matrix **H** is approximated as rank 3 identity matrix which also leads to:

$$\boldsymbol{H} = \boldsymbol{I}_{3\times3} \tag{43}$$

$$\boldsymbol{R} = r\,\boldsymbol{I}_{3\times3} \tag{44}$$

This approximation leads to attitude and gyro bias covariance which are equal in all directions in space, or isotropic. Therefore, the covariance matrix has the form given by:

$$\boldsymbol{P} = \begin{bmatrix} P_q\,\boldsymbol{I}_{3\times3} & P_c\,\boldsymbol{I}_{3\times3} \\ P_c\,\boldsymbol{I}_{3\times3} & P_b\,\boldsymbol{I}_{3\times3} \end{bmatrix} \tag{45}$$

where P_q is the quaternion variance, P_b is the bias variance and P_c is the covariance between quaternion and the bias. Also the state transition matrix is approximated as:

$$\boldsymbol{\Phi} = \begin{bmatrix} \mathbf{I}_{3\times3} & -\Delta t\,\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{46}$$

The Kalman Gain is calculated as follows

$$K_q = \frac{P_{q_{k+1}}^-}{P_{q_{k+1}}^- + r} \tag{47}$$

$$K_b = \frac{P_{c_{k+1}}^-}{P_{q_{k+1}}^- + r} \tag{48}$$

Where K_q is Kalman gain for quaternion and K_b is the Kalman gain for bias estimate and *r* is the measurement noise which is approximated as equal in all the directions. The state update equations are:

$$\begin{aligned} P_{q_k}^+ &= K_q r \\ P_{c_k}^+ &= K_b r \\ P_{b_k}^+ &= P_{b_{k+1}}^- - K_b * P_{c_{k+1}}^- \end{aligned} \tag{49}$$

The next cycle Kalman Update is:

$$\begin{aligned} P_{q_{k+1}}^- &= P_{q_k}^+ - 2P_{c_k}^+\Delta t + P_{b_k}^+\Delta t^2 + \Delta t{\sigma_v}^2 + \frac{{\sigma_u}^2\Delta t^3}{3} \\ P_{c_{k+1}}^- &= P_{c_k}^+ - P_{b_k}^+\Delta t - \frac{{\sigma_u}^2\Delta t^2}{2} \\ P_{b_{k+1}}^- &= P_{b_k}^+ + {\sigma_u}^2\Delta t \end{aligned} \tag{50}$$

where ${\sigma_u}^2$ and ${\sigma_v}^2$ are the scalar covariances of the gyro-drift ramp measurement noise, respectively and Δt is the sampling time. The quaternion state update is done by quaternion multiplication of the estimated quaternion and the weighted residual as follows:

$$\begin{aligned} \boldsymbol{q}_k^+ &= \begin{bmatrix} K_q \dfrac{\tilde{\vec{\boldsymbol{u}}} \times \hat{\vec{\boldsymbol{u}}}}{2} \\ 1 \end{bmatrix} \otimes \boldsymbol{q}_{k+1}^- \\ \boldsymbol{b}_k^+ &= \boldsymbol{b}_{k+1}^- + K_b(\tilde{\vec{\boldsymbol{u}}} \times \hat{\vec{\boldsymbol{u}}}) \end{aligned} \tag{51}$$

The operator ⊗ specifies a quaternion product which is computed by equation [13]. Next cycle quaternion update is done by using quaternion propagation equation of first order, which uses gyro measurements to calculate quaternion [16], while the next cycles update for the bias is same as the updated bias value.

$$\boldsymbol{q}_{k+1}^{-} = \left(\cos\left(\frac{|\boldsymbol{\omega}|}{2}\Delta t\right).\boldsymbol{I}_{4\times4} + \frac{1}{\boldsymbol{\omega}}sin\left(\frac{|\boldsymbol{\omega}|}{2}\Delta t\right).\boldsymbol{\Omega}(\boldsymbol{\omega})\right).\boldsymbol{q}_{k}^{+}$$
$$\boldsymbol{b}_{k+1}^{-} = \boldsymbol{b}_{k}^{-} \tag{52}$$

where the matrix $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is given in equation [23]. As Kalman filter requires an initial estimate of the state values i.e. quaternion and the bias values, the initial estimate of quaternion is done through TRIAD algorithm which has been explained in previous section. The initial gyro bias estimate is set to zero but before launch of the satellite the gyro bias can be measured and set as the initial value of gyro bias in the Kalman filter.

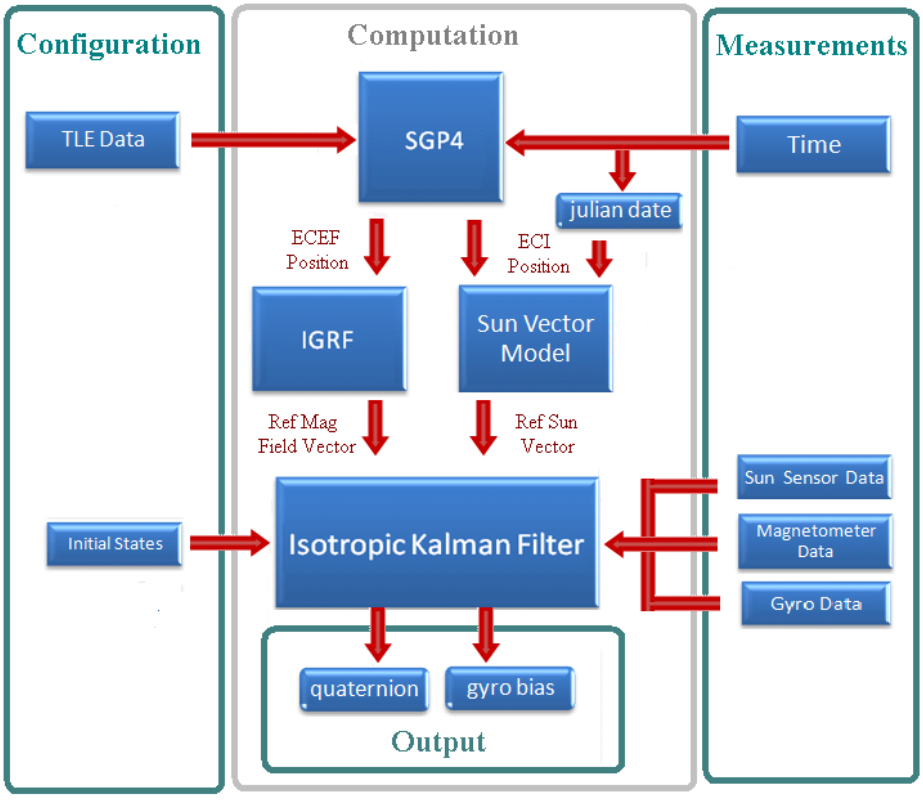

Figure 8: Attitude Determination System Model

6 Simulation and Results

This chapter discusses the simulation and results of the mathematical reference models and the attitude estimation algorithm described in the previous sections. These models and algorithms were simulated in Matlab. The results of the simulation from Matlab were tested against the vectors and attitude generated from Satellite Tool Kit (STK). A sample TLE dataset of a Picosatellite was used for simulations. Satellite attitude quaternion and reference vectors reports were generated from STK and in the following sections the results of this comparison are stated.

6.1 Sun Reference Vector

The Sun position vector software model was referenced from [4]. The output of this function is in J2000 ECI frame. As stated in [4] this model can provide Sun position vector accuracy of about 0.01^{o}. The simulations results indeed show the correct implementation with following results compared with the vectors from STK.

Table 1: Sun Reference Model Simulation Results

	Angle (Deg)
Maximum Error	0.0085
Average Error	0.0033

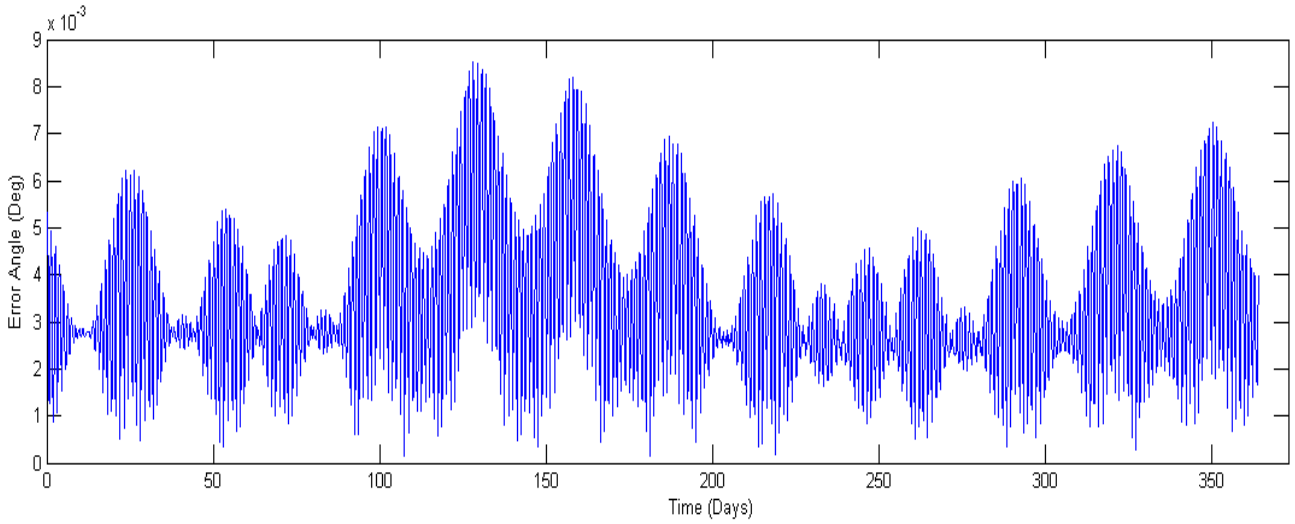

Figure 9: Simulation Graph showing error in Sun vector for 1 year

6.2 Magnetic Field Reference Vector

IGRF model was used to calculate the magnetic field vectors and the output of the IGRF model was then compared with STK generated IGRF vectors in ECEF coordinate systems. The average error was 0.13^{o}. The problem in the software model which was leading to this deviation could not be identified. The simulation results are as follows:

Table 2: Magnetic Field Reference Vector Simulation Results

Property	Result
Maximum Error	0.33 deg
Average Error	0.13 deg
Max. Magnetic Field Magnitude Error	0.79 %
Mean Magnetic Field Magnitude Error	0.37 %

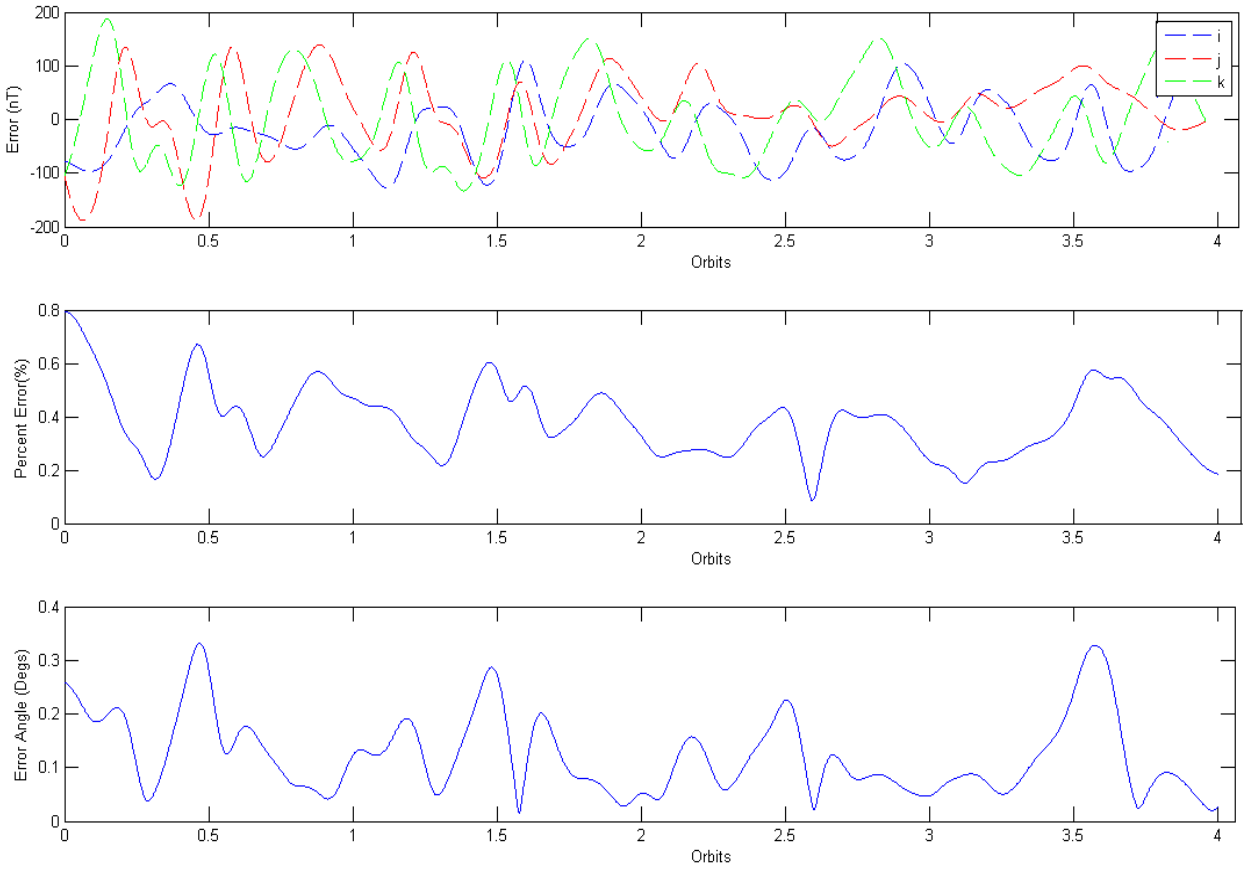

Figure 10: Magnetic Field Reference Vector simulation results

The figure above shows the accuracy results of the reference magnetic field model. The error in magnitude is less than 200nT which results in less than 1% error in the magnetic field magnitude (approx. 21000nT) at the sampled satellite orbit.

6.3 SGP4 Propagator

The SGP4 algorithm used for the ADS was also simulated to verify the accuracy of the algorithm. The output position vector from the SGP4 propagator was compared with position vector from STK. The results are shown in next table and figure.

Table 3: SGP4 position vector simulation results

Property	Result
Maximum Error	0.0000128 deg
Average Error	0.0000124 deg

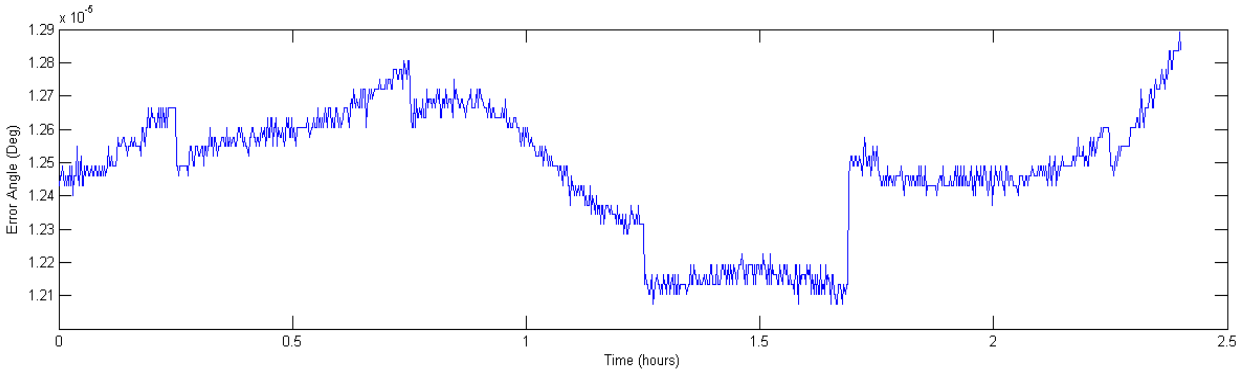

Figure 11: SGP4 comparison results

The results of the SGP4 simulation show high accuracy when compared with the STK generated position vector. The table above shows an average error of 0.000012^{o}, hence the SGP4 algorithm implementation is correct.

6.4 Enhanced TRIAD Algorithm

Enhanced TRIAD method requires careful selection of the Gain $\propto_0$ to achieve good results. Given below are the test conditions and the simulations results.

Table 4: Simulation cases for ETA

	Case 1 $\propto_0 = 1$	Case 2 $\propto_0 = 5$
Magnetometer Noise Covariance	0	0.05 Unit Vector
Sun Sensor Noise Covariance	0	0.05 Unit Vector
Gyro Noise	0	5*1.e-6
Bias drift Noise	0	2*1.e-8

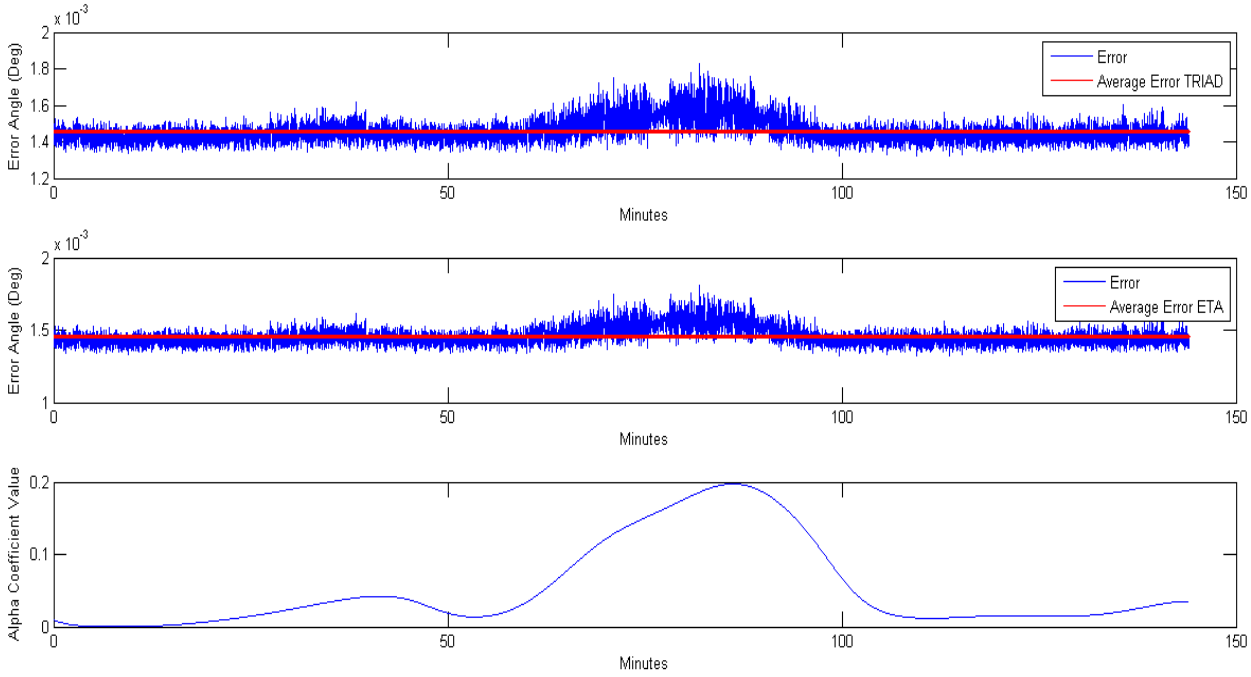

Figure 12: Enhanced TRIAD simulation result Case 1

TRIAD error determines best attitude when the two sensor measurement are orthogonal. As the vector measurement start to become collinear error increases. Depending on the gain of $\propto_0$ the weight of gyro readings increases. Because the maximum angle between Sun sensor and

magnetometer is only 120^{o} for the sampled satellite orbit the error difference between TRIAD and ETA is not significant. In case 2 ETA shows a drop in error as $\propto$ rises but this is also due to the fact that the noise covariance for the gyro in the test condition is very small. Hence to achieve better results the gyro bias needs to be estimated and the gain $\propto$ should not only depend on the angle between the reference vectors but also on the noise covariance of sensors.

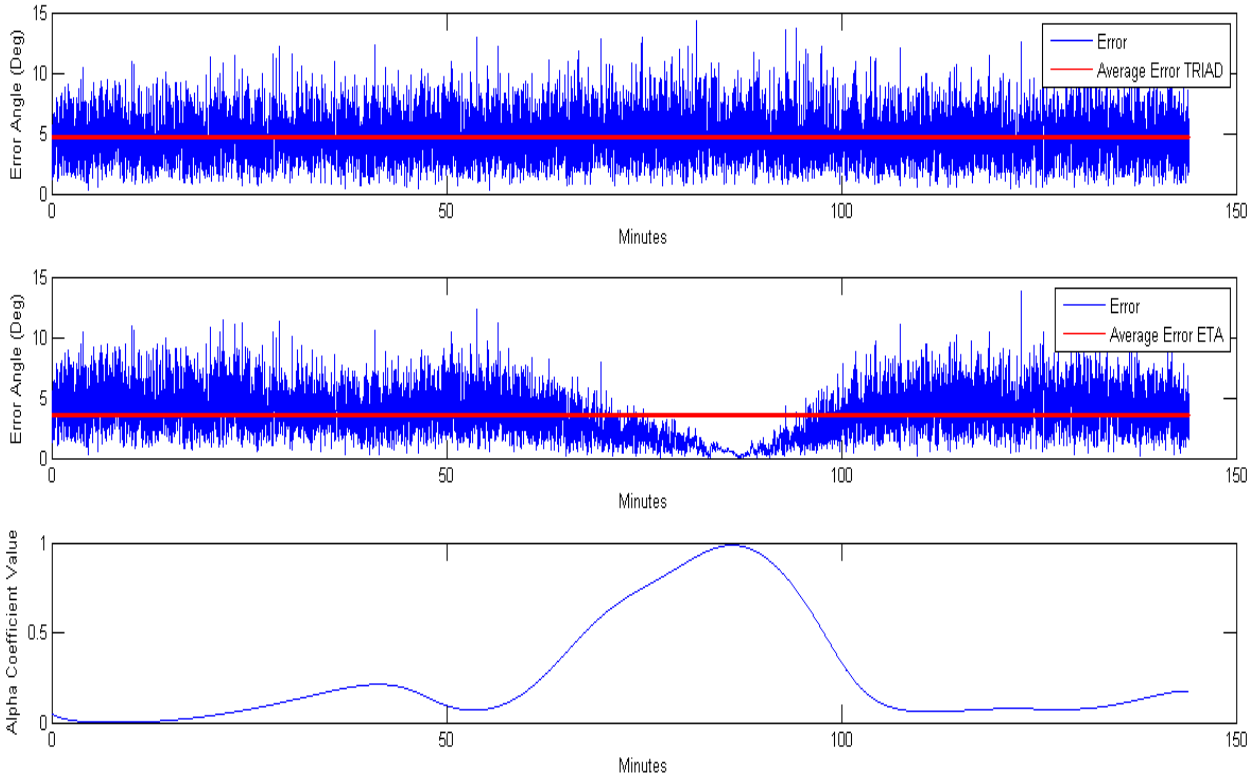

Figure 13: ETA Simulation graphs for case 2

Table 5 : Results of ETA accuracy for two cases

	Case 1	Case 2
ETA Error	0.0085 deg	3.58 deg
TRIAD Error	0.0033 deg	4.68 deg

6.5 Enhanced q-method

Simulation results for EQA are almost the same as that of ETA. Given below are the test conditions and the simulations results.

Table 6: Simulation cases for EQA

	Case 1 $\propto_0 = 1$	Case 2 $\propto_0 = 1$
Magnetometer Noise Covariance	0	0.05 Unit Vector
Sun Sensor Noise Covariance	0	0.05 Unit Vector
Gyro Noise	0	5*1.e-6
Bias drift Noise	0	2*1.e-8

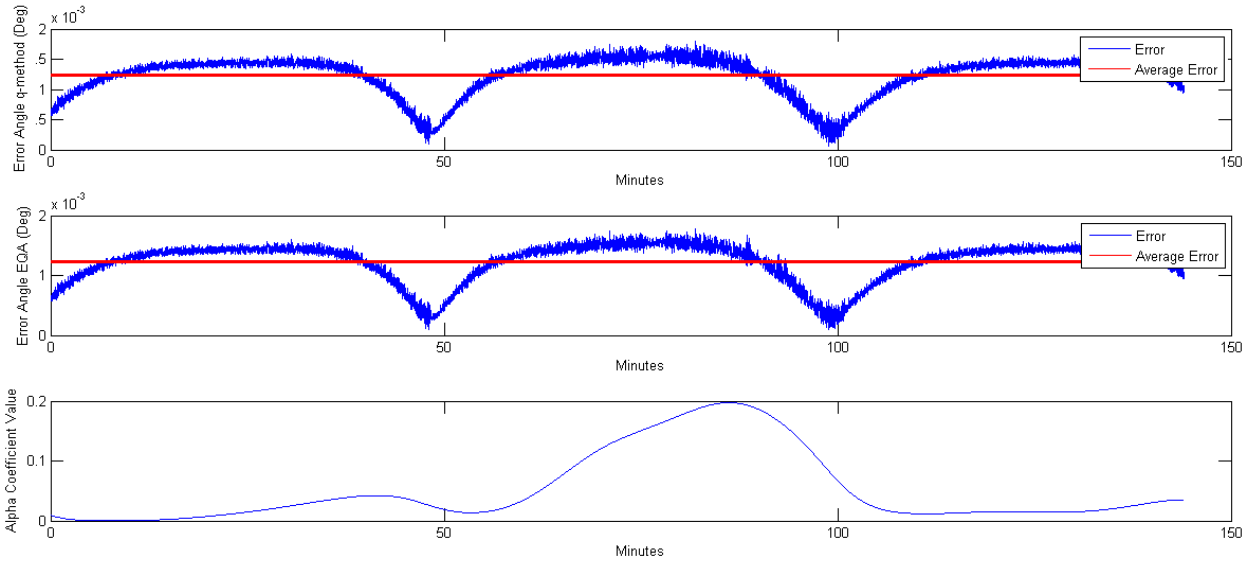

Figure 14: EQA simulation result Case 1

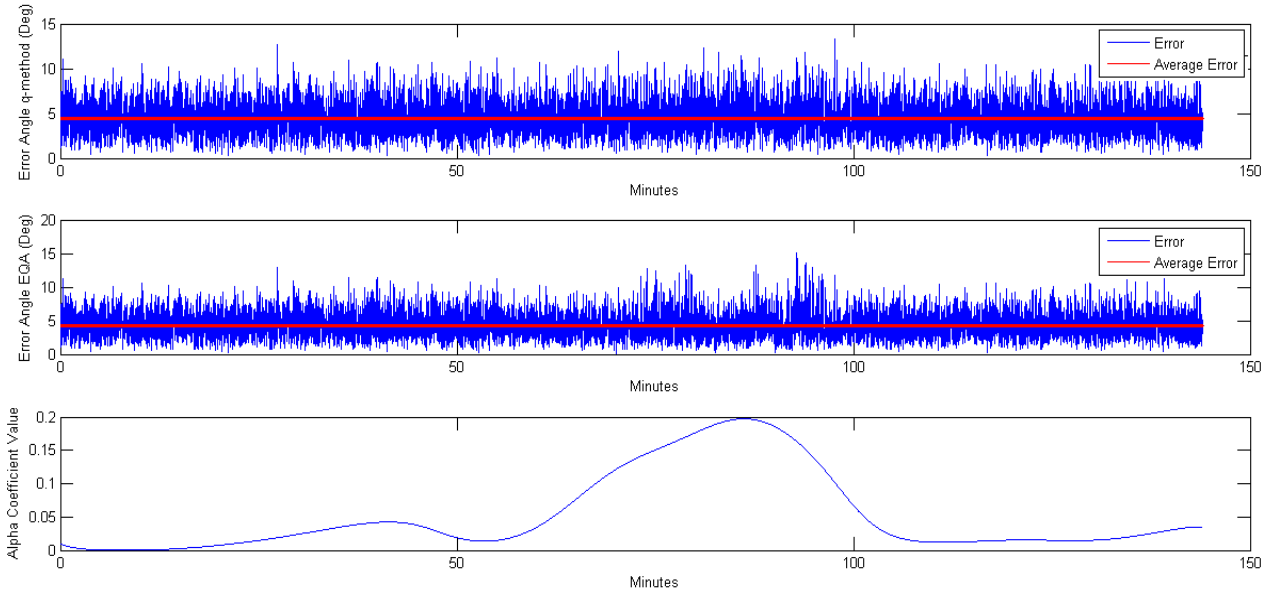

Figure 15: EQA simulation result for Case 2

Table 7 : Results of EQA accuracy for two cases

	Case 1	Case 2
Enhanced q-method Error	0.00122 deg	4.29 deg
q-method Error	0.00123 deg	4.36 deg

The results of Enhanced q-method are similar to ETA. Both of them achieve a high accuracy when both or one of the sensor measurements is very accurate, but as the noise increases the error increases significantly. Both of the these algorithms use gyros reading on the basis of the angle of two measurement vectors and do not depend on sensor noise levels for weighted quaternion estimate. Thus the accuracy of these methods highly depends on the accuracy of the measurements.

6.6 Kalman Filter

This section discusses the simulation results obtained from the Kalman filter model. The Isotropic Kalman filter described in the previous section estimates attitude quaternion as well as the gyro bias. The output of the Kalman filter depends heavily on the input conditions and the noise covariance; hence several test cases have been considered to simulate the output of the Kalman filter with varying conditions. The sampling time is 0.864 sec which corresponds to 0.00001 increase in Julian date, hence simplifies the incrementing time during the course of simulation. Following are the test cases input parameters and results.

Test Case 1:

Table 8: Kalman Filter Test Case 1 parameters

Parameters	Values
Initial Attitude Estimate	TRIAD
Initial Bias	0 (deg/hr)
Magnetometer Noise	$0.001 \times \mathbf{I}_{3\times3}$
Sun Sensor Noise	$0.001 \times \mathbf{I}_{3\times3}$
Gyro Noise	5*1.e-6 (rad/sec^(1/2))
Bias drift Noise	2*1.e-8 (rad/sec^(3/2)

In this test case initial state estimation for the Kalman Filter is given by the TRIAD algorithm, From the previous section it is seen that the TRAID algorithm can determine very accurately the attitude if sensors are also accurate but even with noise covariance of 0.05 (unit vector) the error in determination was less than 5^{o} which is good enough to start the Kalman Filter. With the given noise parameters of the gyro it is assumed to be a very accurate sensor. The initial bias on the gyro for this test case is 0 deg. The measurement covariance matrix **R** is calculated from noise on magnetometer and Sun sensor reading while the process noise **Q** is based on noise from the gyros which predicts the next cycle estimate.

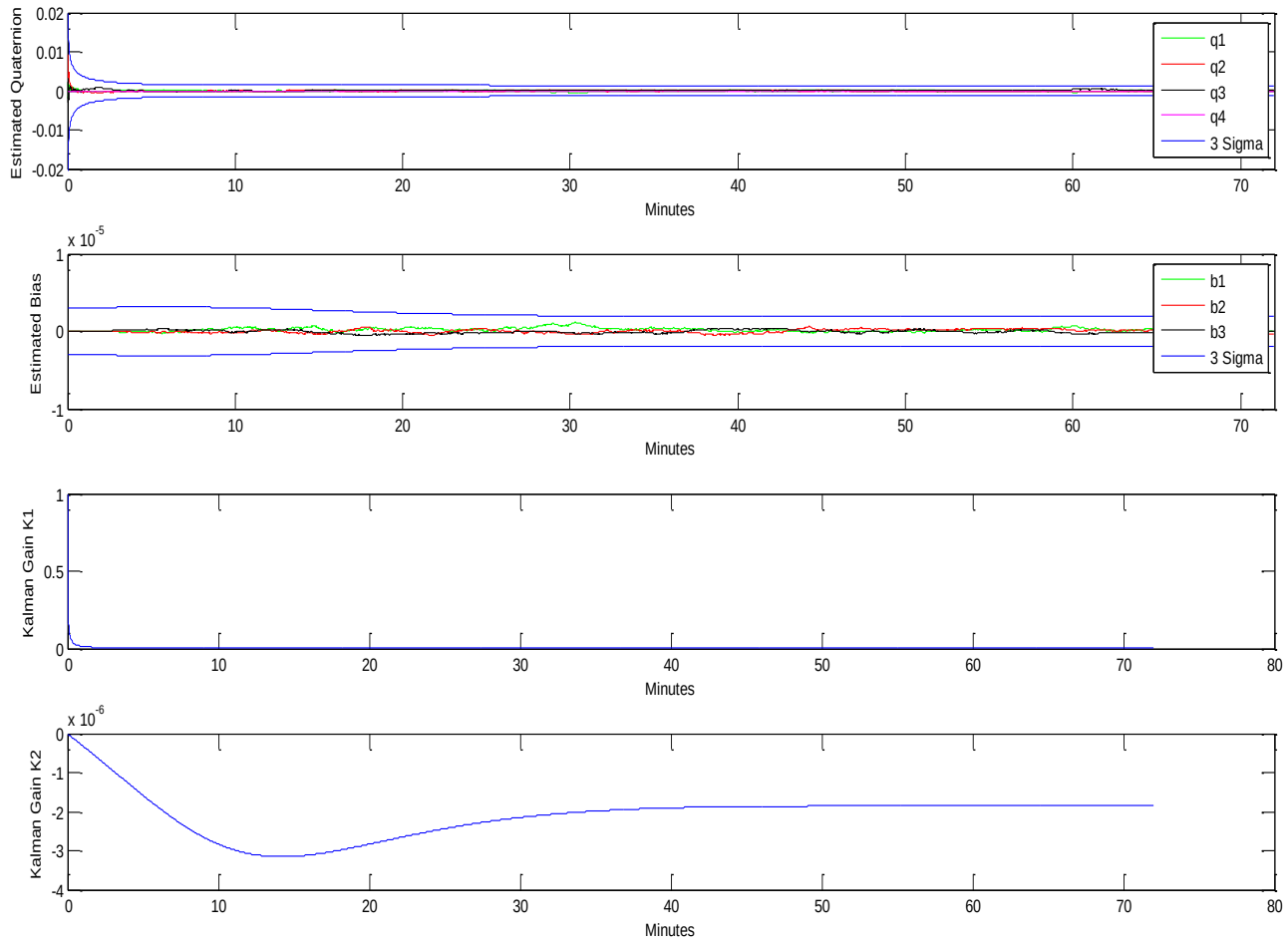

Figure 16: Estimated Quaternion Bias and Error for Case 1

Table 9: Kalman Filter Test Case 1 results

Parameters	Results
Steady State Covariance Matrix	$\begin{bmatrix} 7.3566.e^{-9} & 1.8522.e^{-11} \\ 1.8522.e^{-11} & 1.5754.e^{-13} \end{bmatrix}$
Quaternion 3-Sigma Value	$2.57.e^{-04}$
Bias 3-Sigma Value	$1.19.e^{-06}$
Steady State Error Angle	<0.014 deg

The result shows that the errors for both the quaternion and the bias estimate are within the 3-sigma error bounds calculated from the final covariance matrix. This is the significance of the steady state covariance matrix that the error will be inside this 3-sigma limit. The simulation

graphs also show that the Kalman Gain for the quaternion stabilizes very quickly compared to the Kalman Gain of the Bias Estimate.

Test Case 2:

Table 10: Kalman Filter Test Case 2 Input parameters

Parameters	Values
Initial Attitude Estimate	TRIAD
Initial Bias	0 (deg/hr)
Magnetometer Noise	$0.05\times\mathbf{I}_{3\times3}$
Sun Sensor Noise	$0.05\times\mathbf{I}_{3\times3}$
Gyro Noise	5*1.e-6 (rad/sec^(1/2))
Bias drift Noise	2*1.e-8 (rad/sec^(3/2)

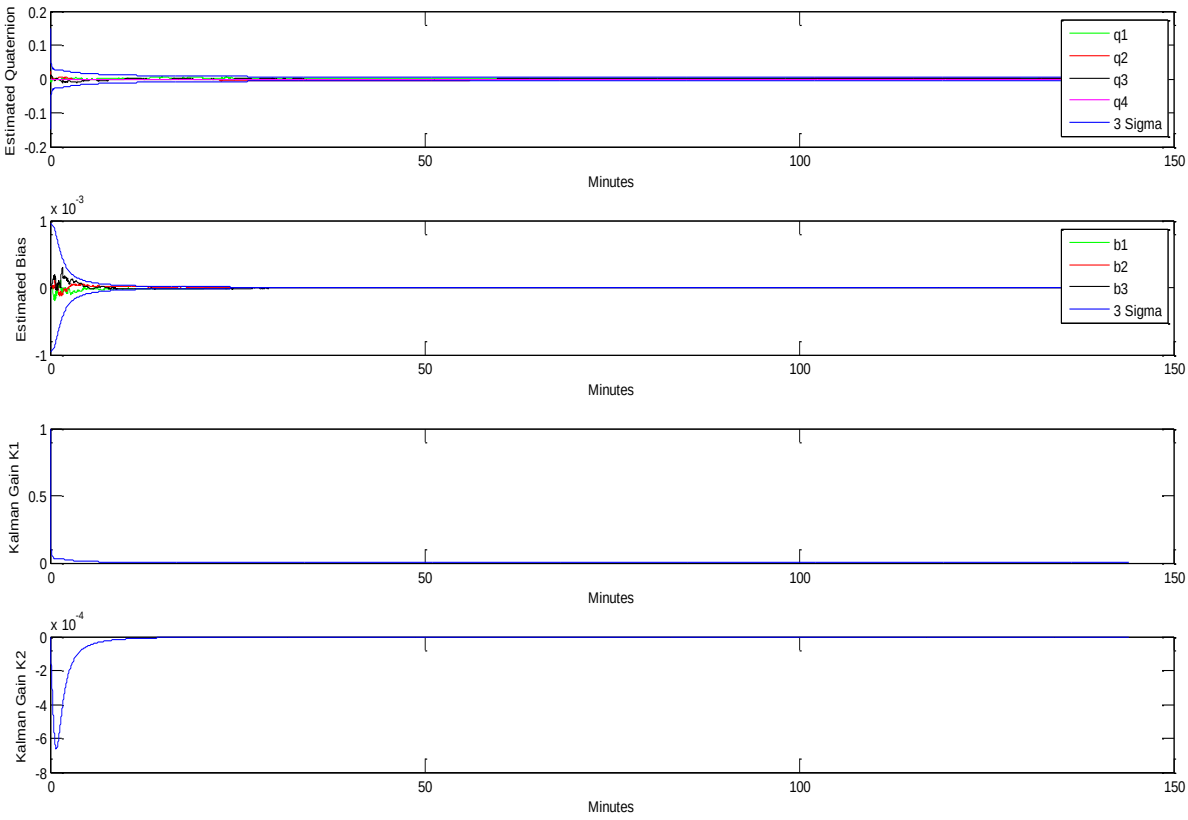

Figure 17: Test Case 2 quaternion and bias error

Table 11: Kalman Filter Test Case 2 results

Parameters	Results
Steady State Covariance Matrix	$= \begin{bmatrix} 2.0179.e^{-6} & 9.3047.e^{-10} \\ 9.3047.e^{-10} & 8.6965.e^{-13} \end{bmatrix}$
Quaternion 3-Sigma Value	$4.26.e^{-03}$
Bias 3-Sigma Value	$2.79.e^{-06}$
Steady state Error Angle	< 0.14 deg

The results from this test case signify the importance of input noise parameters for the Sun sensor and the magnetometer. With respect to test case 1 in test case 2 the noise levels for the Sun sensor and magnetometer are now 0.05 (unit vector) which results in high measurement noise and consequently the final covariance matrix, quaternion and Bias 3-sigma values increase. Hence for a higher measurement noise covariance, the attitude estimation accuracy decreases.

Test Case 3:

Table 12: Kalman Filter Test Case 3 Input parameters

Parameters	Values
Initial Attitude Estimate	80 degrees error
Initial Bias	0 (deg/hr)
Magnetometer Noise	$0.05 \times \mathbf{I}_{3\times3}$
Sun Sensor Noise	$0.05 \times \mathbf{I}_{3\times3}$
Gyro Noise	5*1.e-6 (rad/sec^(1/2))
Bias drift Noise	2*1.e-8 (rad/sec^(3/2)

For the test case 3 the initial quaternion was chosen as **q**=[0 0 0 1], instead of determining it from TRIAD algorithm. Rests of the conditions are the same.

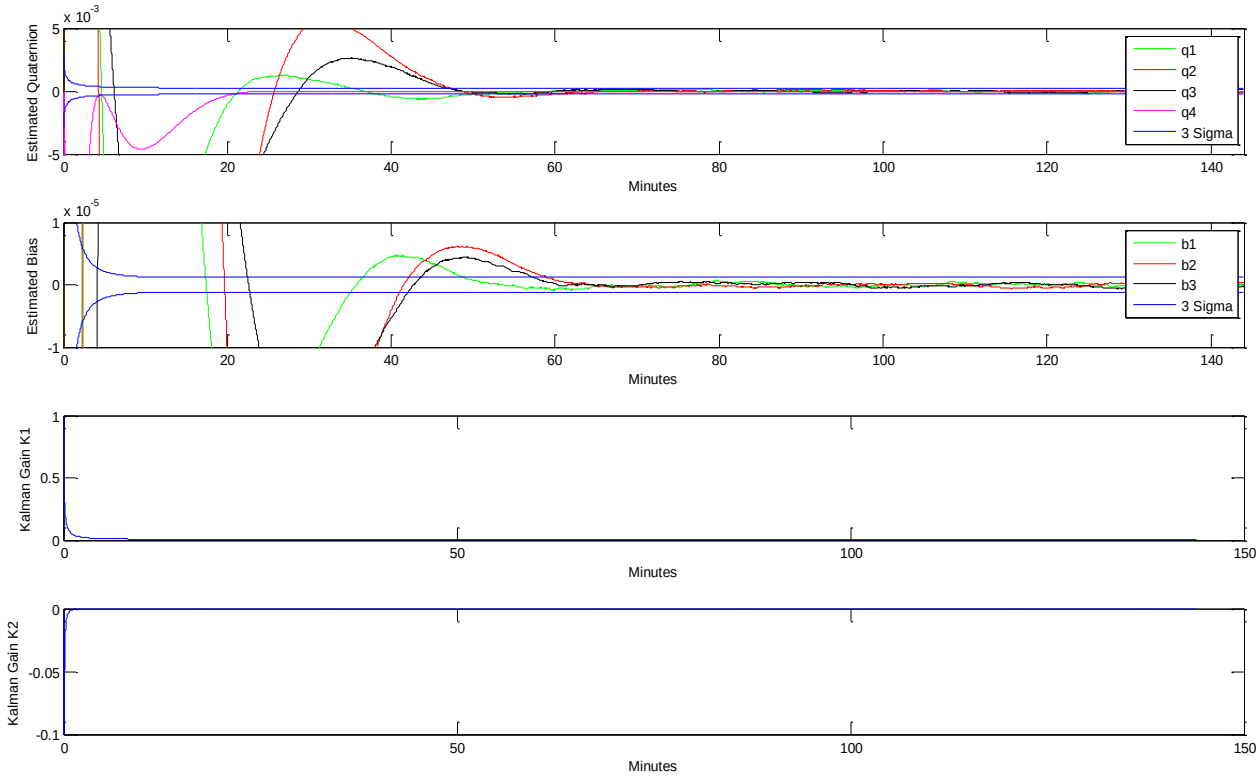

Figure 18: Test Case 3 quaternion and bias estimates

Table 13: Kalman Filter Test Case 3 results

Parameters	Results
Steady State Covariance Matrix	$= \begin{bmatrix} 2.0179.e^{-6} & 9.3047.e^{-10} \\ 9.3047.e^{-10} & 8.6965.e^{-13} \end{bmatrix}$
Quaternion 3-Sigma Value	$4.26.e^{-03}$
Bias 3-Sigma Value	$2.79.e^{-06}$
Steady State Error Angle	< 0.14 deg

Due to very high initial error of 80^{o} the quaternion and bias error exceeds the 3 σ sigma limits but converges to the steady state covariance matrix of the previous case within 60 minutes.

Test Case 4:

Significance of this test case is to show the ability of Kalman filter to accurately estimate the bias on each axis. Hence for this test case an initial gyro bias of 0.1 deg/hr is added to gyro readings. Table below shows the input parameters for this test case.

Table 14: Kalman Filter Test Case 4 Input parameters

Parameters	Values
Initial Attitude Estimate	TRIAD
Initial Bias	0.1 deg/hr on each axis
Magnetometer Noise	$0.05 \times \mathbf{I}_{3\times3}$
Sun Sensor Noise	$0.05 \times \mathbf{I}_{3\times3}$
Gyro Noise	5*1.e-6 (rad/sec^(1/2))
Bias drift Noise	2*1.e-8 (rad/sec^(3/2)

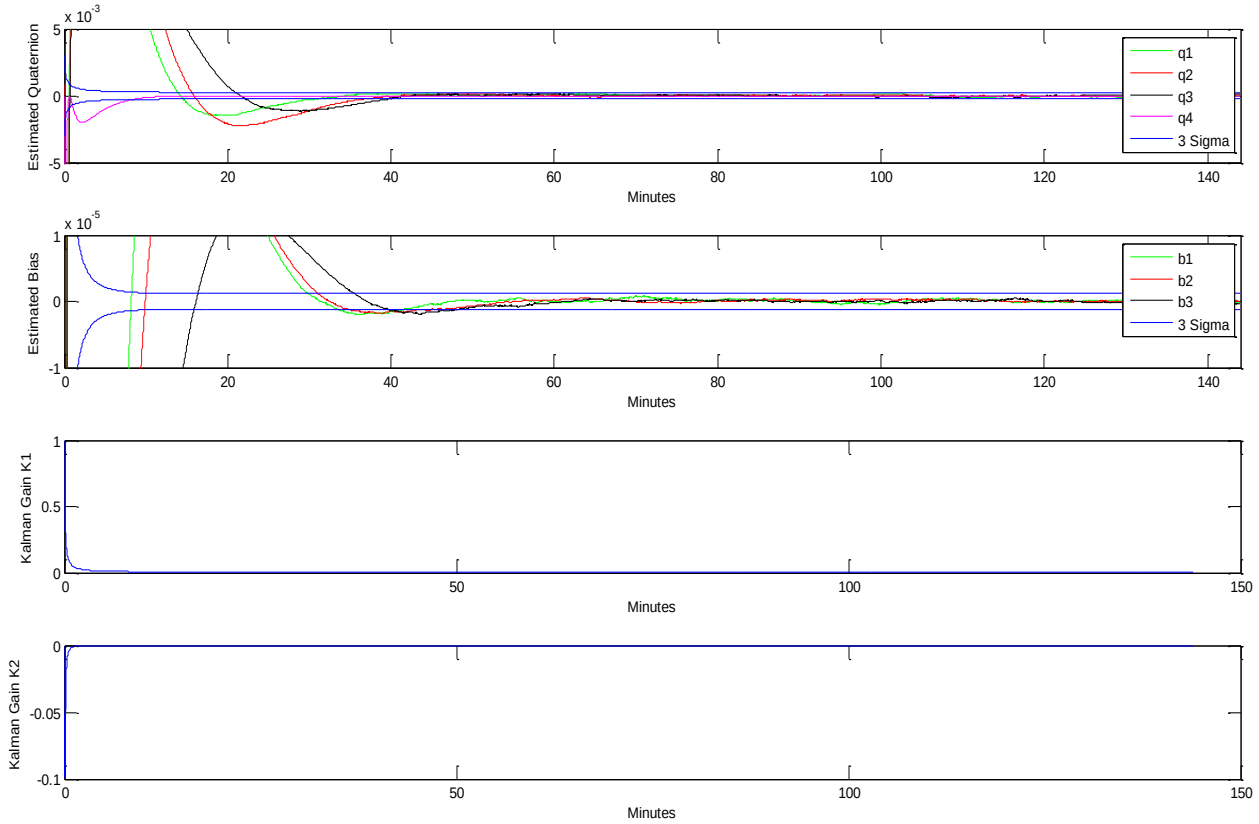

Figure 19: Test Case 4 quaternion and bias error and Kalman Gain

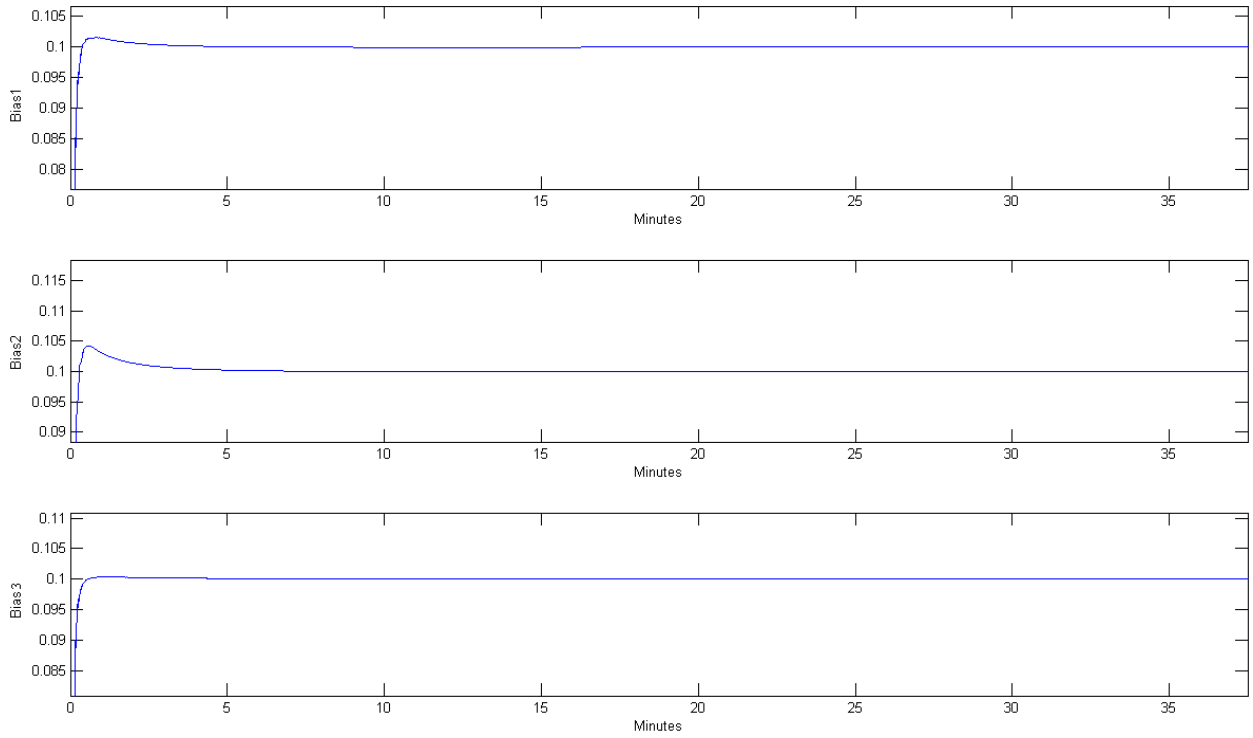

Figure 20: Kalman Filter Bias Estimates Test Case 4

Figure above shows the estimation of the bias for the three axes, as noticeable in the graph that Kalman Filter very quickly converges to the initial bias value. While both ETA and EQA are unable to estimate bias, Kalman Filter bias estimates help predict the satellite attitude very accurately.

Table 15: Kalman Filter Test Case 4 results

Parameters	Results
Steady State Covariance Matrix	$= \begin{bmatrix} 2.0179.e^{-6} & 9.3047.e^{-10} \\ 9.3047.e^{-10} & 8.6965.e^{-13} \end{bmatrix}$
Quaternion 3-Sigma Value	$4.26.e^{-03}$
Bias 3-Sigma Value	$2.79.e^{-06}$
Steady State Error Angle	< 0.14 deg

Test Case 5:

This test case scenario visualizes worst case initial conditions with an initial attitude error of 80 degrees and initial bias on each axis.

Table 16: Kalman Filter Test Case 5 Input parameters

Parameters	Values
Initial Attitude Estimate	80 deg
Initial Bias	0.1 deg/hr on each axis
Magnetometer Noise	$0.001 \times \mathbf{I}_{3\times 3}$
Sun Sensor Noise	$0.001 \times \mathbf{I}_{3\times 3}$
Gyro Noise	5*1.e-6 (rad/sec^(1/2))
Bias drift Noise	2*1.e-8 (rad/sec^(3/2)

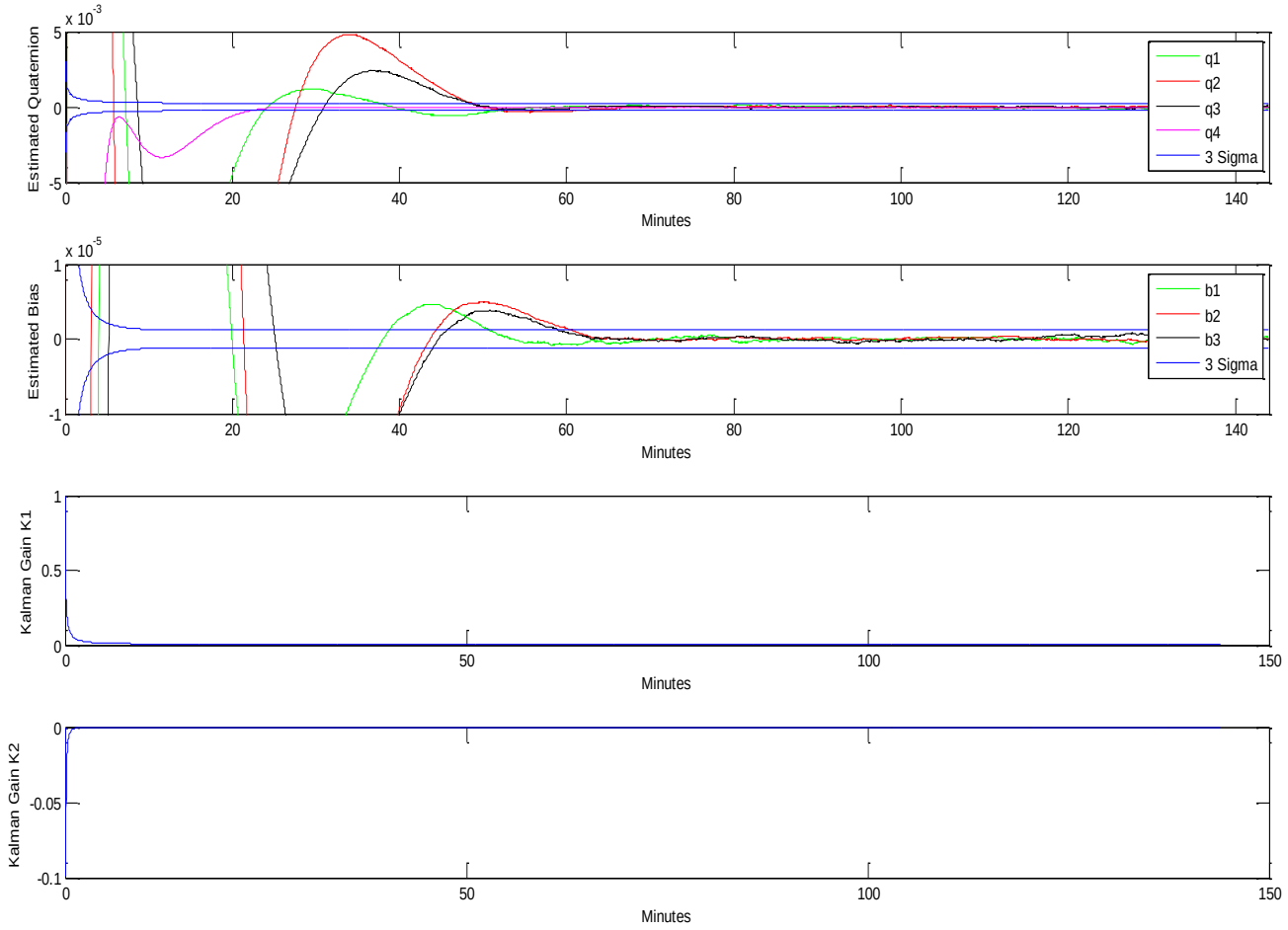

Figure 21: Kalman Filter Test Case 5 quaternion and bias estimation

Table 17: Kalman Filter Test Case 5 results

Parameters	Results
Steady State Covariance Matrix	$= \begin{bmatrix} 7.3566.e^{-9} & 1.8522.e^{-11} \\ 1.8522.e^{-11} & 1.5754.e^{-13} \end{bmatrix}$
Quaternion 3-Sigma Value	$2.57.e^{-04}$
Bias 3-Sigma Value	$1.19.e^{-06}$
Steady State Error Angle	< 0.014 deg

The simulation result shows that the Kalman filter error converges to the estimated 3 σ bounds in 60 minutes which is less than the orbital time period of a satellite in low Earth orbit. To achieve better accuracy and lower steady state error it is imperative that highly accurate sensors are used because it will result in less measurement noise covariance **R** and consequently the response of the Kalman filter improves rapidly.

After the simulation of the three attitude estimation methods IKF is found to be the best in terms of accuracy of attitude estimation. In the test conditions with high measurement noise and high initial gyro bias IKF estimate attitude quaternion very accurately and is thus deemed to be the most suitable method for attitude estimation.

7 Software Implementation

Following the simulation of the attitude determination software in Matlab, the ADS algorithm was ported to C for hardware implementation. Software that resembled closely with the ADS software in Matlab was developed. The results generated were then verified against result the results generated by the Matlab software. This chapter discusses the details of the individual software element and some results of implementation.

7.1 Modules

The main modules of the ADS software are given in the table below and their software dependencies are discussed in next sections.

Table 18: Main ADS modules

Modules	Description
ads	The main module of the ADS software
kalman filter	This module implements Isotropic Kalman Filter
sgp4	Orbit propagator model for spacecraft with orbit period less than 225 minutes
sun vector	Computes the reference Sun vector in ECI coordinates.
mag vector	Computes the reference magnetic field vector in ECI coordinated using the IGRF model
times	Utilities to calculate time conversions
matop	Utilities for matrix, vector and quaternion operations

7.1.1 Main Module

Module Name	Dependencies		
ads	*kalmanfilter*	*sgp4*	*times*

This is the top level module of the ADS software. It accepts input in the form of TLE data and the orbit propagation time. This module is responsible for setting up timing utilities and initializations for the Kalman filter and the magnetic field and sun vectors.

7.1.2 Kalman filter

Module Name	Dependencies			
kalmanfilter	*sgp4*	*sunvector*	*magvector*	*times*

This module is the implementation of the IKF. It uses time utilities to propagate time in required format to the dependant modules. It uses matop for the matrix, vector and quaternion operations.

7.1.3 Orbit Propagator

Module Name	Dependencies
sgp4	*None*

This module is the implementation of the sgp4 algorithm described in chapter 4. Input to the sgp4 module is TLE data and time in minutes. *readTLE*() and *init_sgp4constants()* reads TLE data and accordingly initializes the sgp4 constants respectively.

7.1.4 Reference Sun vector

Module Name	Dependencies
sunvector	*matop*

It calculates the reference Sun vector based on the model described in chapter 4. Input to this module is time in Julian date format. Function *precess()* computes the transformation from MOD to J2000 frame.

7.1.5 Reference Magnetic Field vector

Module Name	Dependencies
magvector	*None*

Computes the reference magnetic field vector based on the IGRF model. Input to this module is the position vector of the satellite and time in Julian date. Function *igrf2005()* initializes the IGRF coefficients for year 2005, *Schmidt()* computes coefficients that relate Schmidt functions to associated Legendre functions, *recursion()* recursively calculates derived Legendre polynomials and *bfield()* computes the magnetic field vector at given point. The function *GSTvernalequinox()* calculates the angle between vernal equinox and prime meridian to compute ECI to ECEF transformation matrix.

7.1.6 Time Utilities

Module Name	Dependencies
times	*None*

This module calculates time utility functions. The function *days2ymdhms()* converts time from TLE data year, month, days, hours, minutes and seconds. This information is then used to calculate the Julian date.

7.1.7 Matrix operations

Module Name	Dependencies
matop	*None*

This module performs operations related to matrix, vectors and quaternions. As vectors and quaternion are also represented in matrix form, therefore vector and quaternion multiplication corresponds to matrix operations which are implemented in this module.

7.2 Results of C Implementation

The results of the C implementation were compared against the simulation results from the Matlab. The results showed correct implementation in C. Following graph shows the comparison between the sgp4 algorithm in C and sgp4 in Matlab. The output difference between the two implementations is negligible. The error angle between the vector from Matlab and the one from the C implementation is of the order of 10^{-6} (deg).

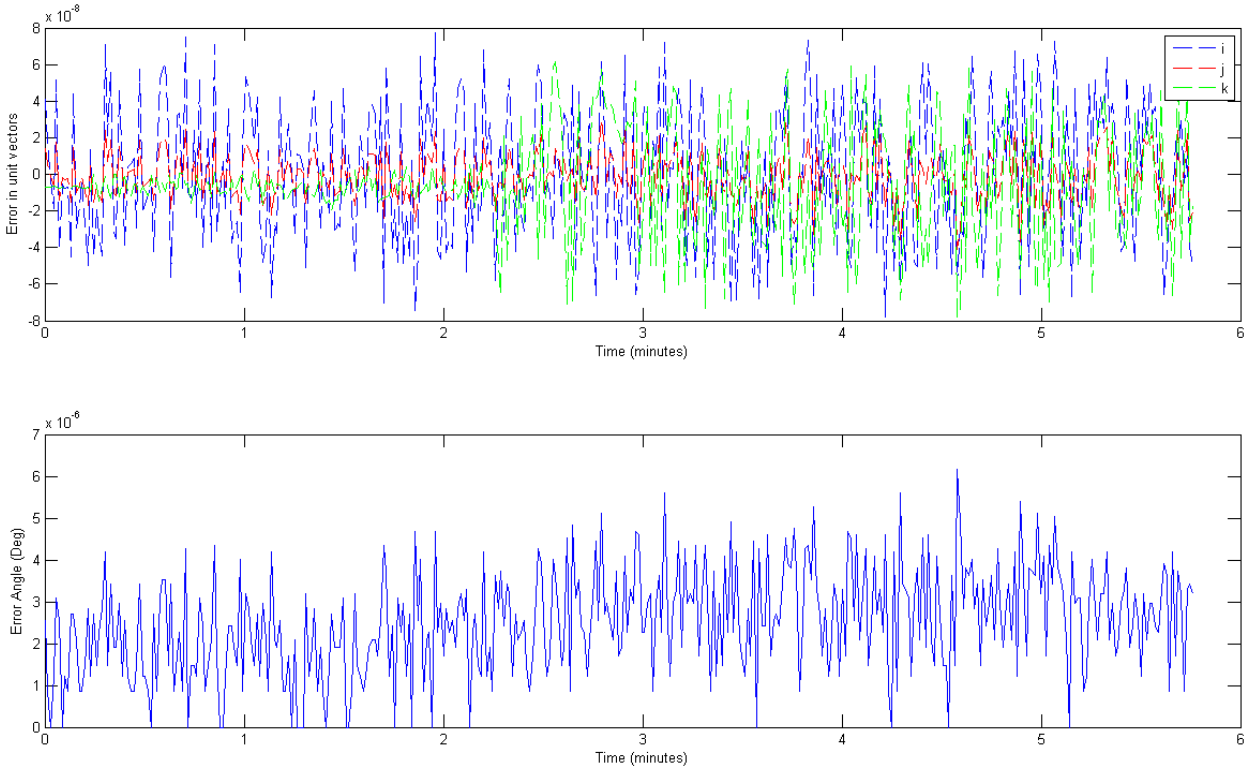

Figure 22: Matlab and C implementation comparison

Similarly the results from all other modules of the C code were compared to Matlab implementation. The results verified that the C implementation is correct and there is no significant loss in accuracy.

8 Hardware Selection

For the hardware implementation of the attitude determination algorithm research, analysis and evaluation on different possible hardware architectures was done. While searching for hardware platform one of the requirements is that the hardware device must have the SPI (Serial Peripheral Interface) and I^2C interface which is required for interface with the onboard sensors. The hardware platform selected during this research will not only be used to implement attitude determination algorithm but also to implement attitude control algorithm. The attitude control algorithm is not in the scope of this thesis work but a general idea of the amount of complexity of Attitude determination algorithm is known. Therefore enough resources should be available on the finally selected hardware so that it can incorporate attitude control algorithm as well. The attitude control algorithm will most probably be a PID (Proportional Integral Derivative) controller for one axis micro reaction wheel. This algorithm will require a high update rate as compared to the Kalman filter but will require less amount of code memory considering attitude determination algorithm also needs to compute reference magnetic field, Sun vector and SGP4 algorithm. The following characteristics are most important when doing a survey on possible hardware platforms [21].

8.1 Selection Criterion

1. Performance
2. Power Consumption
3. Flexibility and Ease of use

8.1.1 Performance

System performance is an important criterion when designing a computationally demanding application. The performance of the system can be measured in the form of Million Instructions per Cycle (MIPS), Million Multiply Accumulates per Second (MMACS) etc. A simpler way of measuring the performance of the system can just be done by counting the number of system clock cycles required to compute an algorithm.

8.1.2 Power

In a space based application power is one of the most important requirements. Given limited resources of a Picosatellite power efficiency becomes even more important. Due to current growth in the market of battery powered devices a lot of research is being done to produce power efficient hardware which can result in longer battery life. Hardware vendors are targeting new family of devices with low power consumption and increased performance. This has led to COTS low power devices which are very suitable for small satellite missions.

8.1.3 Flexibility and Ease of Use

Flexibility is also an important issue when dealing with space related projects. Flexibility is the capability of the system to be modified at any stage. With changing requirements, conditions and uncertainties involved in the Space related projects device flexibility is a vital requirement.
Ease of use is also a significant characteristic in the selection of the device. Considering the time constraints and the deadline of the project it was not possible to learn and use a completely new tool which require significant amount of prior knowledge and experience.

8.2 Architecture Exploration

On the basis of the above mentioned characteristics following three devices were further analyzed to choose the best hardware platform.

1. Microcontroller/General Purpose Processor
2. Field Programmable Gate Array (FPGA)
3. Digital Signal Processor (DSP)

8.2.1 Microcontroller/General Purpose Processor

A microcontroller is usually a general purpose CPU with IO interfaces, memory, timers and other peripheral units like SPI (Serial Peripheral Interface), I^2C (Inter Integrated Circuit Wire), ADC (Analog to Digital Converter) etc. A general purpose processor can be used for wide variety of tasks but it might not be optimized for certain applications. Next generation microcontrollers contain RISC (Reduced Instruction Set Controller), pipelined and superscalar architectures and dedicated MAC (Multiply Accumulate) units which has vastly increased the performance level of the modern day microcontrollers. Hence the boundary between

microcontrollers and DSP is diminishing. Modern day 32-bit pipelined RISC architectures provide a higher degree of performance level but still they cannot compete with the performance of DSP or FPGA.

The On board computer chosen for UWE-3 is a MSP430F5438 device from Texas Instruments. MSP430 is a 16-Bit RISC ultra low power microcontroller with 165 µA/MIPS. Some important characteristics of the on board computer are given below:

- Up to 25 MHz system clock
- Hardware Multiplier Supporting 32-Bit Operations
- Active Mode : 165 µA/MHz at 8MHz
- Standby Mode 2.60 µA

As MSP430 device and software tools were already available, the attitude determination algorithm was first simulated for the MSP430F5438 device. The IAR embedded workbench was used for profiling of the attitude determination algorithm. During profiling IAR embedded workbench outputs the number of floating point operations performed within each function of the algorithm. This information was very important in finding the bottleneck of the algorithm, which can later be optimized to make algorithm as efficient as possible. The multiply, add and divide operation performed by each function are shown below:

Table 19: Number of different floating point operations

Functions	Division	Multiplication	Addition	Trigonometric
Sun Vector	1	63	27	15
Magnetic Field Vector	98	2660	1090	0
SGP4	28	270	126	21
Kalman Filter	15	68	24	2

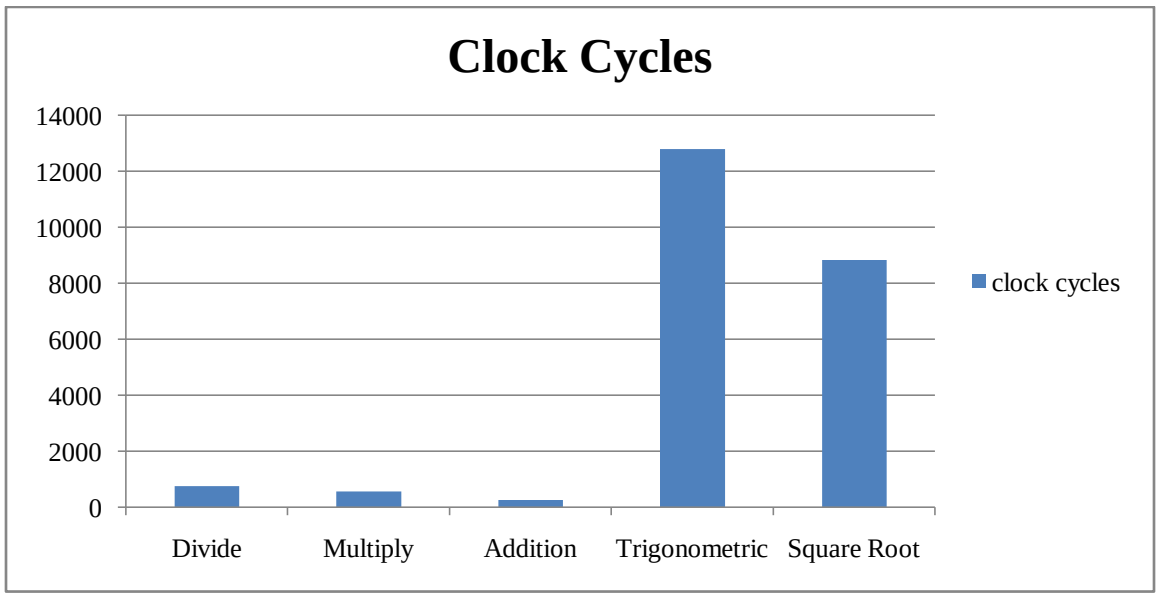

Figure 23: Floating point operations comparison

Table 20: Comparison of ADS Modules

Functions	Clock Cycles	Program Memory Kbytes	Data Memory bytes
Sun Vector	253,024	12,434	258
Magnetic Field Vector	2,722,771	31,164	7,186
SGP4	785,026	43,870	1,638
Kalman Filter	372,144	24,056	1,066

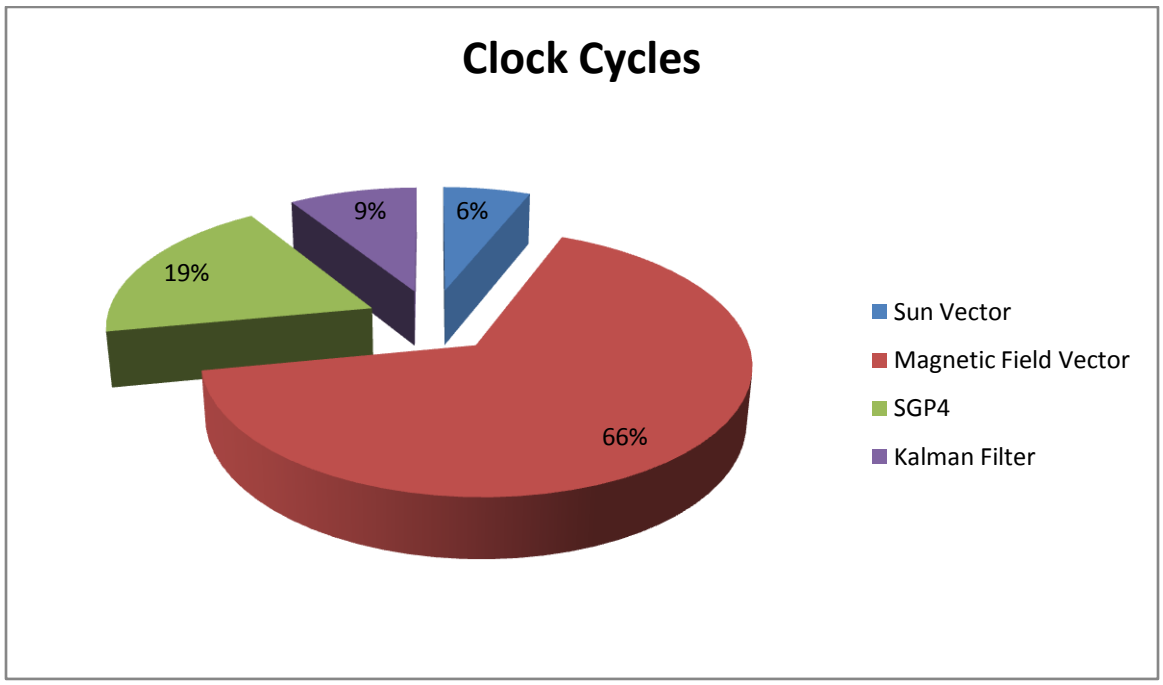

Figure 24: Clock Cycles Comaparison

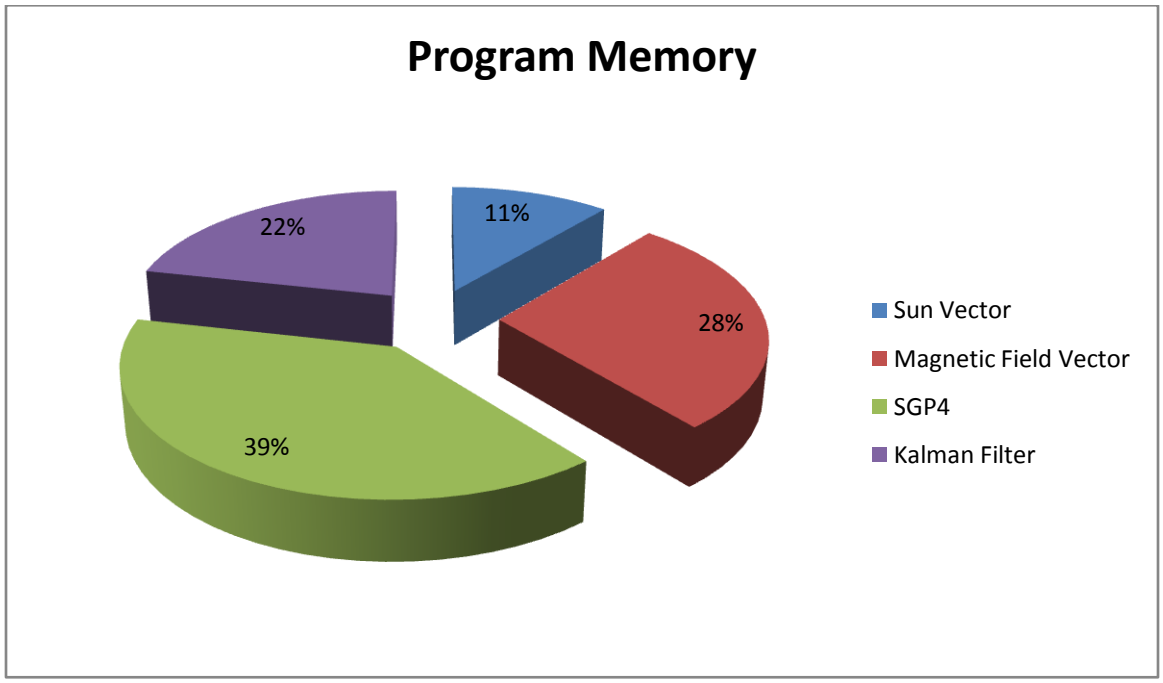

Figure 25: Program Memory Size comparison

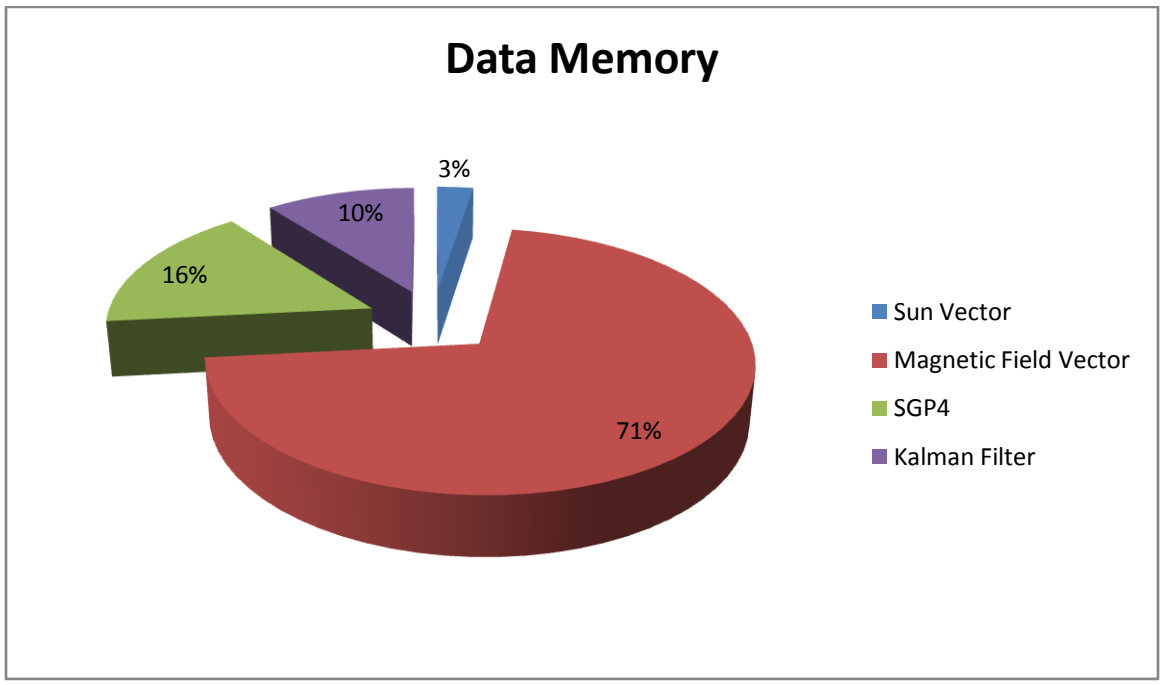

Figure 26: Data Memory Size Comparison

8.2.2 Field Programmable Gate Array (FPGA)

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by the end user after manufacturing hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) [3].

The most common FPGA architecture consists of an array of configurable logic blocks (CLBs), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.

A classic FPGA logic block consists of a 4-input lookup table (LUT), and a flip-flop, as shown below. In recent years, manufacturers have started moving to 6-input LUTs in their high performance parts, claiming increased performance.

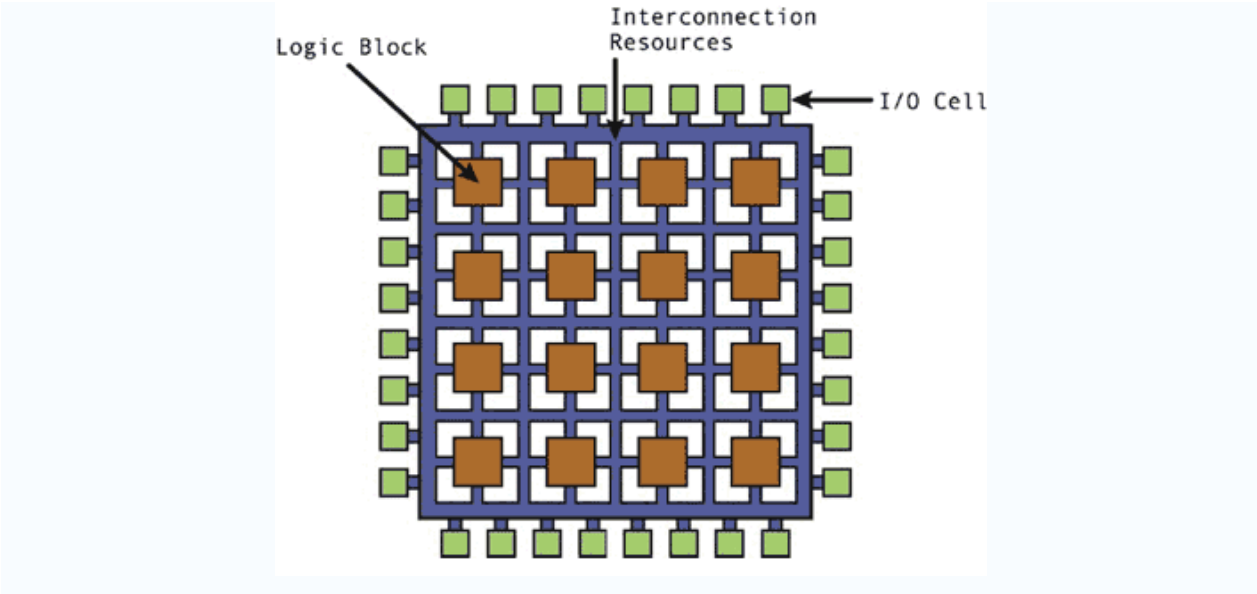

Figure 27: FPGA block Diagram

8.2.2.1 FPGA Coding Techniques

Programming of an FPGA is usually done in a Hardware Descriptive Language (HDL). HDL language syntax as compared to other higher level programming languages includes notations which define concurrent and timed execution which is an attribute of a hardware design. The two commonly used HDL languages for hardware design are:

1. Verilog HDL
2. VHDL (**V**ery High Speed Integrated Circuits **HDL)**

VHDL has a higher learning curve than Verilog but offers more flexibility and control to the design. VHDL is modeled after Ada while Verilog is based on C for syntax. VHDL is stricter in syntax than Verilog. In Verilog a "Module" is the basic building block while in VHDL a "Entity" is the basic building block.

8.2.2.2 FPGA Design Synthesis and Implementation

Steps involved in the generation of the final bit file from the HDL code are highlighted here and shown in figure 27.

- The system is designed in a HDL code or a schematic diagram

- The design is then synthesized to create a Netlist. The V, VHD, SCH files are translated into an industry standard format EDIF (Electronic Design Interchange Format) file. A Netlist is a list of nets which define connection between flip flops and gates etc.
- After Netlist generation the steps Translate, Map, Place & Route are performed. This process is called Implementation. In this process the Netlist elements are mapped to FPGA physical resources. This process usually can only be done using FPGA vendor provided software tools as it requires information about the device internal architecture [22].
- After Place & Route a bit file is generated which is the then used to configure the FPGA.

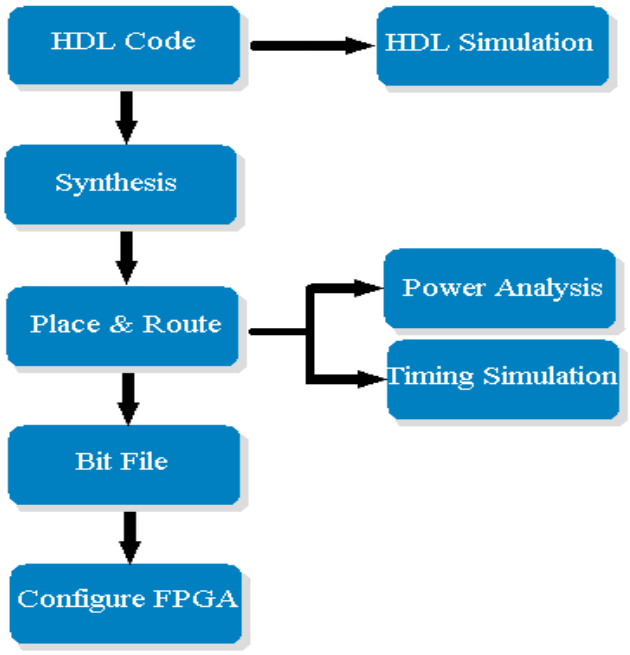

Figure 28: FPGA design Flow

8.2.2.3 Processor Based FPGA Design

The presence of processor core in a FPGA has enabled software/hardware co-design application to be built in a single device. In high performance embedded application the best utilization of the resources can be achieved by a mixed software/hardware mixed design. The most time critical part of the process can be implemented in hardware which would otherwise be the

bottleneck in the standalone software solution. While less performance and time critical process of the application can be executed on a processor.

Modern day FPGAs provide both the softcore and hardcore processor solutions for embedded design. A softcore processor is usually provided in form of an IP (Intellectual Property) which is entirely implemented on the FPGA fabric. A hard processor core is a dedicated part of the integrated circuit of an FPGA [23].

The examples of the softcore processor are

- Microblaze, Picoblaze for Xilinx FPGAs
- Nios II, Leon for Altera FPGAs
- Cortex-M1, 8051 for Actel FPGAs.

While the examples for of hard processor core are

- Power PC in Xilinx Virtex Family FPGAs
- AVR core in Atmel FPSLIC FPGA

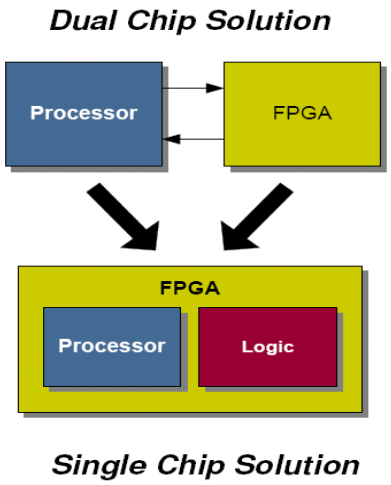

Figure 29: Processor based FPGA design Concept [24]

Using a softcore processor provides additional flexibility to the design, because the softcore processor can be configured to have as much resources as required. The program and data memory resides in the block RAMs of the FPGA and the amount of block RAM used by the processor can be configured. Another advantage of the embedded processor design in an FPGA

is use of multiple processors on a single FPGA fabric. This is shown in Figure 30: Xilinx Virtex-II Pro Die Showing PowerPC Hard Processors [26]. Newer versions of vendors software tools have made multi processor design for FPGA very simplified. Multiple processors can be configured in different topologies e.g. symmetric, dataflow, distributed multiprocessor configuration [25].

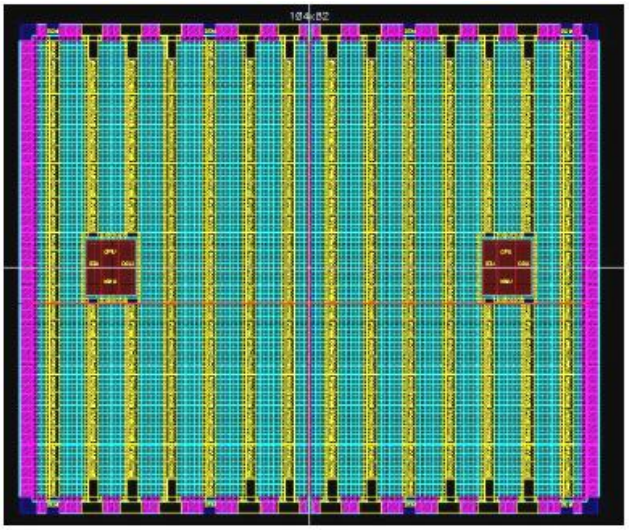

Figure 30: Xilinx Virtex-II Pro Die Showing PowerPC Hard Processors [26]

8.2.2.4 FPGA Based Implementation

To implement ADS Algorithm on FPGA there are two options

1. Hardware only implementation
2. Implementation on an embedded softcore processor

8.2.2.4.1 Hardware Implementation of the Attitude Determination Algorithm

The design of the ADS algorithm on hardware can be done either using a Hardware Descriptive Language or designing at a higher level of abstraction e.g. C/C++. The first option of writing HDL code is a time consuming process hence the second option was deemed feasible considering the project timeline.

As C language Implementation of the ADS algorithm has already been completed before, a search was done to find suitable C to HDL converters. Several such compilers were found which allows the user to build FPGA based application in C language or other high level language like

C. Some of these tools have limitations on the usage of floating point variables, two dimensional arrays and pointers etc. For the ADS algorithm floating point usage is a necessity therefore the choice of such tools were then restricted to a few. Evaluation versions of a few such tools were requested from different vendors. Impulse Accelerated Technologies were the first to offer an evaluation version of their tool "Impulse CoDeveloper" for our project. Hence it was decided that Impulse CoDeveloper will be used for analysis of Hardware implementation. Following sections are referenced from Impulse CoDeveloper user guide.

8.2.2.4.1.1 Impulse CoDeveloper

Impulse CoDeveloper includes the Impulse C software-to-hardware compiler, interactive parallel optimizer, and Platform Support Packages supporting a wide range of FPGA-based systems. Impulse tools are compatible with all popular FPGA platforms [27]. The complete CoDeveloper environment consists of a set of libraries allowing Impulse C applications to be executed in a standard C/C++ desktop compiler (for simulation and debugging purposes) as well as cross-compiler and translation tools allowing Impulse C applications to be implemented on selected programmable hardware platforms. Additional tools for application profiling and co-simulation with other environments (including links to EDA (Electronic Design Automation) tools for hardware simulation) are provided.

A lot of useful documentation, tutorials and examples are provided along with Impulse CoDeveloper which were extremely useful in understanding the principles and techniques for efficient hardware designing using C language. The figure below shows FPGA design flow using Impulse CoDeveloper.

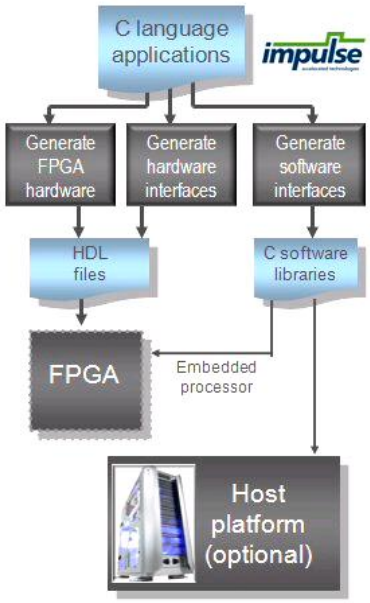

Figure 31: Impulse CoDeveloper C-to-FPGA Tools [Source: Impulse CoDeveloper User Guide]

8.2.2.4.1.1.1 Writing Applications in Impulse CoDeveloper

Programming for FPGA using multiple processes is different from programming applications which target traditional processors. Therefore some new concepts of processes and streams are used in Impulse CoDeveloper. This section is referenced from Impulse CoDeveloper documentation and further details can be found on ImpulseC website and software help.

8.2.2.4.1.1.2 Process

Processes are the fundamental units of computation in an Impulse C application. Once they have been created, assigned and started, the processes in an application execute as independently synchronized units of code on the target hardware. Programming with Impulse C processes is conceptually very similar to programming with threads. As with thread programming, each Impulse C process has its own control flow, is independently synchronized and has access to its own local memory resources (which will vary depending on the target platform).

8.2.2.4.1.1.3 Streams

Data processed by the application flows between different processes in the form of Streams. Stream-oriented programming is conceptually similar to dataflow programming, but is less restrictive in that it more easily supports process synchronization (through data buffering and message passing) and non-dataflow concepts such as shared memories.

Impulse C defines several distinct mechanisms for communicating among processes:

- Streams — Buffered, fixed-width data streams
- Signals — One-to-one synchronization with optional data
- Registers — Unbuffered data or control lines
- Shared memories — Memory shared between hardware and software

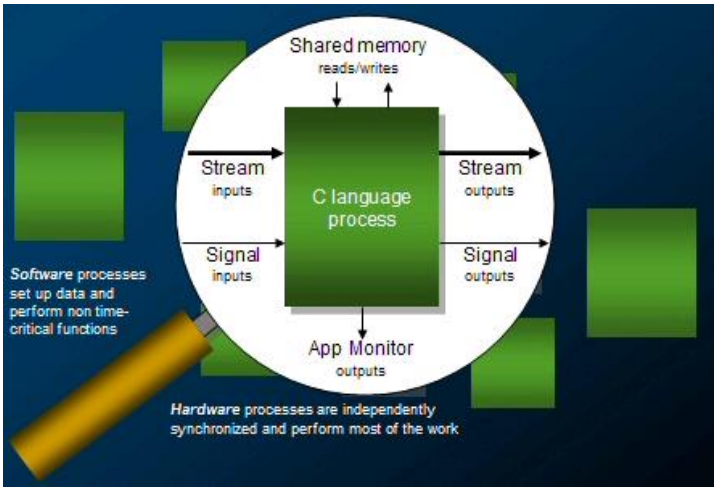

Figure 32: Impulse C Programming Model [Source: Impulse CoDeveloper User Guide]

Impulse C pragmas are used to control features like pipelining and scheduling at the level of C source code. While writing Impulse C processes, it is imperative to understand how the C code is parallelized to achieve the best results in terms of execution and size of the resulting logic. Following example shows the advantage of using PIPELINE pragma for increased speed [27].

$$for\ (i = 0; i < 10; i++)$$

$$sum += i \ll 1;$$

When this loop is not pipelined it will result in one stage consisting of an adder and one shifter executed ten times and the computational time will be at least ten times the shifter and the adder delays. When the loop is pipelined it will result in two stages and consisting of an adder and

shifter executed concurrently. The computational time now will be ten times the delay of either the adder or shifter whichever is greater as given below [27]:

10 × (delay (shifter) + delay (adder)) without pipeline

10 × max ((delay (shifter), delay (adder)) with pipeline

More examples of Impulse C pragmas can be found in Impulse C user guide.

8.2.2.4.1.2 HDL Implementation of Attitude Determination Algorithm

To start with C to HDL implementation using Impulse CoDeveloper Sun Vector reference model was first programmed in Impulse C. A process "Sun_vector" was created with an input stream of Julian Date and output stream of a 3 dimensional Sun vector. Impulse C provides floating point support for Xilinx and Altera FPGAs only. Hence to build an application which involves floating point operations, only Xilinx and Altera FPGAs can be targeted.

To implement trigonometric functions in hardware Cordic functions were used. Cordic function examples were given with Impulse CoDeveloper for $90° \leq \theta \geq 0°$, which was extended for $360° \leq \theta \geq 0°$. CORDIC (for COordinate Rotation DIgital Computer) is a simple and efficient algorithm to calculate hyperbolic and trigonometric functions. It is commonly used when no hardware multiplier is available (e.g., simple microcontrollers and FPGAs) as the only operations it requires are addition, subtraction, bitshift and table lookup [3]. Similarly implementation for power function was also written in Impulse C. This implementation was referenced from Verilog implementation of the power function in [28].

$$pow(x, y) = \exp(y * \log(x))$$

Xilinx Generic option was selected for HDL code generation by Impulse CoDeveloper. Impulse C has software processes to test the functionality of the application. It was used to verify the output of the Hardware process "Sun_Vector". The HDL code generation tool of the Impulse CoDeveloper shows following characteristics of the HDL code.

Table 21: HDL characteristics of Sun vector implementation

Characteristic	Value
Total Stages	163
Estimated DSPs	288
Max Unit Delay	64

The total no. of stages represents the number of clock cycles to execute the code. It shows that FPGA can run reference Sun vector algorithm in only 163 cycles. Estimated DSPs show the DSP blocks that would be utilized to execute the code. Max unit delay is the combinational circuit delay in the FPGA. The speed of the execution on FPGA depends upon this delay.

After generating HDL code from Impulse CoDeveloper Xilinx ISE tool was used to synthesize the code for further analysis. Xilinx Virtex-5 xc5vlx50 was used as target device for HDL code synthesis in Xilinx ISE tool. The synthesis report below shows the utilization of the available resources in FPGA. It shows that our application exceeds the resources available on Virtex-5 device.

Table 22: Synthesis Report from Xilinx ISE

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	27118	28800	94%
Number of Slice LUTs:	43956	28800	152%
Number of fully used LUT-FF pairs	19467	51607	37%
Number of bonded IOBs	74	440	16%
Number of Block RAM/FIFO	3	48	3%
Number of DSP48Es	232	48	483%

The utilization column in the above shows that Sun vector reference model, which require least amount of processor resources, cannot fit into a high density FPGA using hardware implementation.

8.2.2.4.2 Implementation on an Embedded Softcore Processor

The results shown in the previous sections depict the hardware only solution is not practically possible for our algorithm. One more significance of this synthesis report is that even a Processor based embedded solution is not feasible. The basic idea of processor based FPGA design is to implement software bottleneck in hardware while the rest of the code executes sequentially like on a normal processor. In our attitude determination algorithm the computational burden is homogenously distributed over different parts of the algorithm. Hence there is no identifiable bottleneck in the algorithm and the resources available on FPGA are not adequate enough to have a hardware implementation for even a smaller portion of the algorithm.

8.2.3 Digital Signal Processor (DSP)

A Digital Signal Processor is a specialized processor which can efficiently perform digital signal processing tasks [3]. Most of the digital signal processing algorithms involves computation of FFT (Fast Fourier Transform) and FIR (Finite Impulse Response) filters which depend heavily on the performance of the MAC unit of the processor therefore DSP have special arithmetic units.

Some important features of a DSP are:

- Special Multiply Accumulate unit (MAC)
- Separate program and data memories
- Pipelined Architecture
- Fast memory access

Some of the high end DSP also have hardware floating point unit which can perform very fast floating point operations. Next generation DSP use SIMD (Single Instruction Multiple Data), VLIW (Very Large Instruction Word) and superscalar architecture to increase parallelism.

- SIMD (Single Instruction Multiple Data) architecture is very effective when same operation is to be performed on different data sets. Examples of such algorithm are found

commonly in Low Level Image processing applications. For example basic image filtering involves same operation on each pixel and when the size of image is very large it will require a lot of time to complete the process.

- VLIW (Very Large Instruction Word) refers to a processor architecture which is designed for Instruction Level Parallelism. A VLIW instruction encodes multiple operations hence one instruction can use multiple execution units at the same time. To accommodate this VLIW architectures usually are at least 64 bit wide and some architectures have even wider instructions [3].

- A superscalar CPU architecture implements a form of parallelism called instruction-level parallelism within a single processor. It thereby allows faster CPU throughput than would otherwise be possible at the same clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier [3].

8.2.3.1 DSP Selection

After survey of available DSP devices two DSP's were selected for further evaluation. Initial selection was based on the power consumption of the devices. Two low power digital signal processors from Texas Instruments were selected.

- TMS320C55x™ DSPs
- TMS320C674x Low Power DSPs

8.2.3.1.1 TMS320C55x DSPs

The C55x DSP architecture achieves high performance and low power through increased parallelism and with focus on power savings. The CPU supports an internal bus structure that is composed of one program bus, one 32-bit data read bus and two 16-bit data read buses, two 16-bit data write buses, and additional buses dedicated to peripheral and DMA activity. These buses provide the abilities to perform up to four 16-bit data reads and two 16-bit data writes in a single cycle.

The C55x CPU provides two multiply-accumulate (MAC) units, each capable of 17-bit x 17-bit multiplication and a 32-bit add in a single cycle. A central 40-bit arithmetic/logic unit (ALU) is supported by an additional 16-bit ALU. Use of the ALUs is under instruction set control, providing the ability to optimize parallel activity and power consumption. These resources are managed in the Address Unit (AU) and Data Unit (DU) of the C55x CPU.

The C55x CPU supports a variable byte width instruction set for improved code density. The Instruction Unit (IU) performs 32-bit program fetches from internal or external memory and queues instructions for the Program Unit (PU). The Program Unit decodes the instructions, directs tasks to the Address Unit (AU) and Data Unit (DU) resources, and manages the fully protected pipeline [29].

8.2.3.1.2 TMS320C674x DSPs

TMS320C674x VLIW DSP is from the low power floating point family of Texas Instruments requiring 20 times lower standby power and 1/3 the power consumption of the existing floating point DSP processors available today. With maximum processor operating frequency up to 300 MHz. The C674x DSP core can perform 2400/1800 MIPS/MFLOPS (Million Floating Point Operations) [30].

The power consumption of the C674x CPUs is in the range of 6mW in idle mode and 420mW maximum in Active mode (70% max load of CPU running at 300 MHz at 1.2V, 25° C). C674x has two multiplier units with Mixed-Precision IEEE Floating-Point Multiply Supported up to:

- 2 ×SP × SP -> SP Per Clock
- 2× SP × SP -> DP Every Two Clocks
- 2× SP × DP -> DP Every Three Clocks
- 2× DP × DP -> DP Every Four Clocks

where SP is the32-bit single precision floating point, while DP is 64-bit double precision floating point. C674x DSP core also has Six ALU (32-/40-Bit) Functional Units

- Supports 32-Bit Integer, SP (IEEE Single Precision/32-Bit) and DP (IEEE Double Precision/64-Bit) Floating-Point
- Supports up to four SP Additions Per Clock, four DP Additions Every 2 Clocks

- Supports up to two floating-Point (SP or DP) approximate reciprocal or square root operations per cycle

The above mentioned floating point operations characteristics of C674x core are vital in comparing relative processing speed to the C55x DSP or the MSP430 microcontroller.

8.2.3.2 DSP Power Estimation

TMS320VC5505 and TMS320VC5504 low power DSPs have standby power as low as 0.12mW and performance up to 600 MIPS. Texas Instrument provides a spreadsheet for power estimation but this estimation spreadsheet is not available for C5505 or C5504 DSP. Spreadsheets for C5506, C5509 are available which can give an approximated power consumption of C5505 or C5504 DSP. The inputs to the spreadsheet are Core and IO voltage, frequency, temperature and utilization of different modules present in the device.

The following table shows power consumption in a scenario where 100MHz operating frequency, 1.6V core voltage, 75% CPU utilization, I^2C in active mode and one DMA channel in active mode are utilized for C5509 DSP.

Table 23: Power consumption summary for C5509 DSP

Module	Power Consumption (mW)
CLKGEN	21.520
CPU	75.200
DMA	7.511
I^2C	2.432
Total	106.6

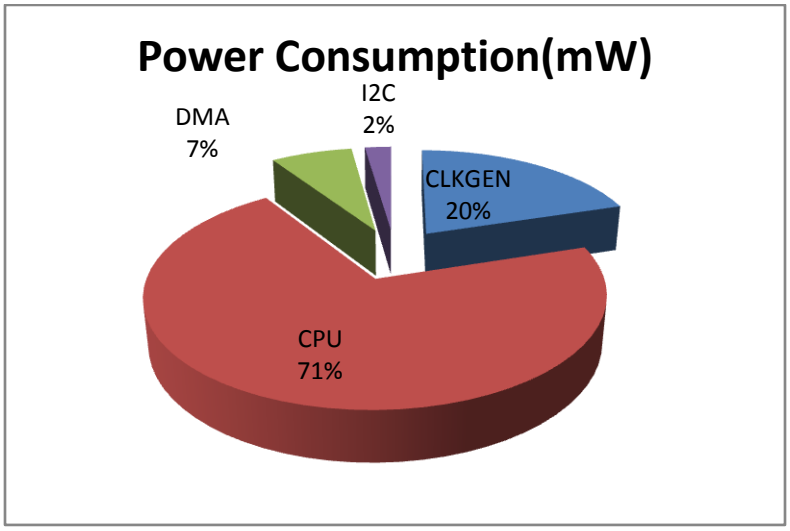

Figure 33: Power Consumption C5509 DSP

Unfortunately power estimation spreadsheet for C674x low power floating point DSP is also not available so further analysis of C674x power cannot be done in simulation.

8.2.3.3 Profiling

Profiling of the Attitude Determination Algorithm was done in Code Composer Studio (CCS) by Texas Instruments. A C55x Cycle Accurate simulator was used for code debugging and code profiling for C55x DSPs and C674x Cycle Accurate simulator was used for profiling of Low Power Floating Point DSPs family C674x. Profiling was also done for Texas Instrument MSP430 device which is the On board Computer (OBC) of the UWE-3.

Table 24: Speed Comparison of DP C55x and C674x

Functions	Clock Cycles C674x	Clock Cycles C55x
Sun Vector	8,652	66,653
Magnetic Field Vector	82,853	1,161,269
SGP4	70,574	507,329
Kalman Filter	54,768	259,815

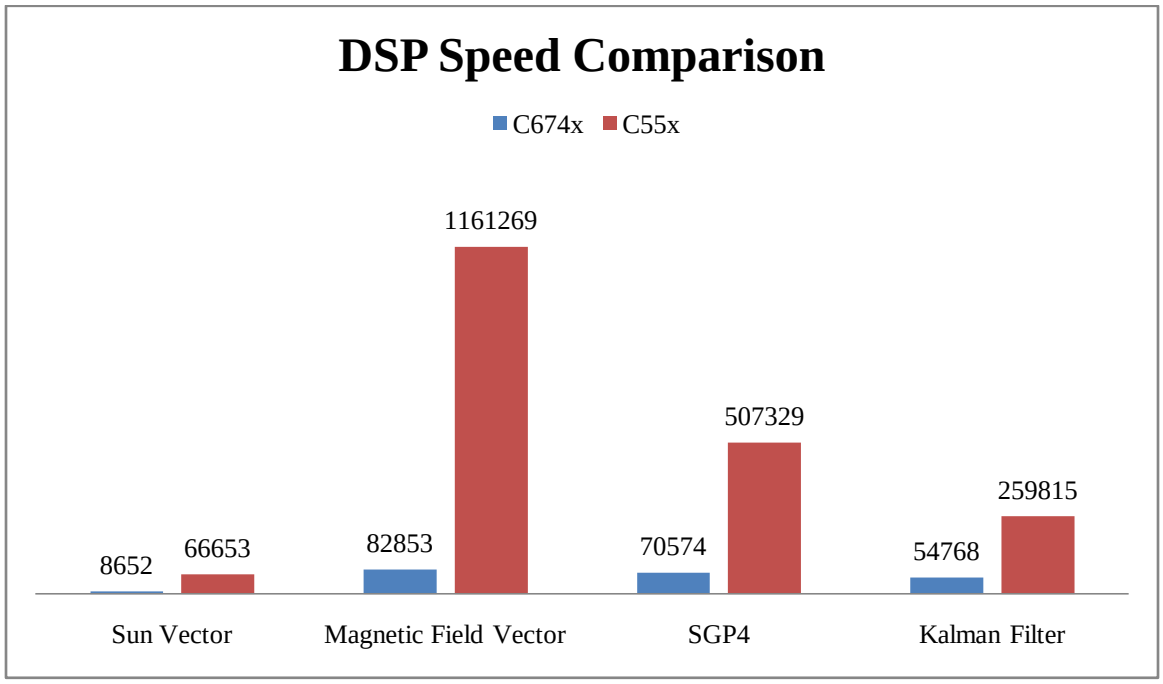

Figure 34: Speed Comparison for C55x and C674x DSP

8.2.3.3.1 Magnetic Field Reference Model

The profiling of ADS algorithm shows that magnetic field reference vector calculation vector is the bottleneck of the code. It takes highest amount of clock cycles to compute the reference vector. Therefore this model was further analyzed to increase the speed of the algorithm.

The formula to calculate magnetic field at a given point is given in equation [14]. The speed and accuracy of the magnetic field reference model depends upon the number of spherical harmonics of degree n and order m being computed. Therefore the magnetic field model was analyzed by simultaneously changing the degree and order of the spherical harmonics used for magnetic field reference vector calculation. The profiling of the code was done in Code Composer Studio for C674x DSP to measure the clock cycles while error was measured by simulation in Matlab. The following table and the graphs show the result of these analyses.

Table 25: Magnetic Field Model Analysis

n,m	Clock Cycles C674x DSP	Max Magnitude Error (%)	Max. Error Angle (Deg)	Average error Angle (Deg)
13	82,853	1.51	0.84	0.14
12	71,936	1.50	0.84	0.15
11	62,196	1.50	0.83	0.15
10	52,295	1.47	0.82	0.15
9	44,018	1.57	0.88	0.15
8	35,857	1.51	0.84	0.15
7	28,312	1.74	1.00	0.17
6	22,498	2.10	1.20	0.24
5	16,860	3.42	1.48	0.42
4	12,216	6.43	3.44	0.79
3	8,180	12.26	5.64	2.04
2	5,097	34.87	17.76	4.53
1	2,673	59.42	30.04	7.55

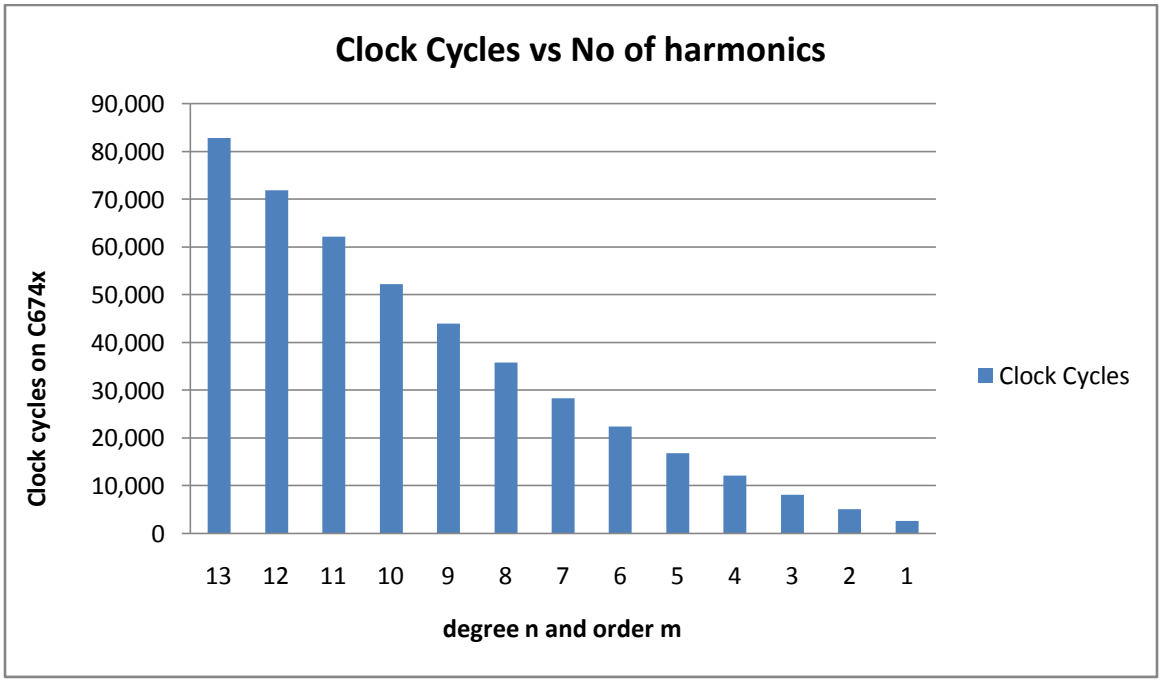

Figure 35: Magnetic Field Model Speed Analysis

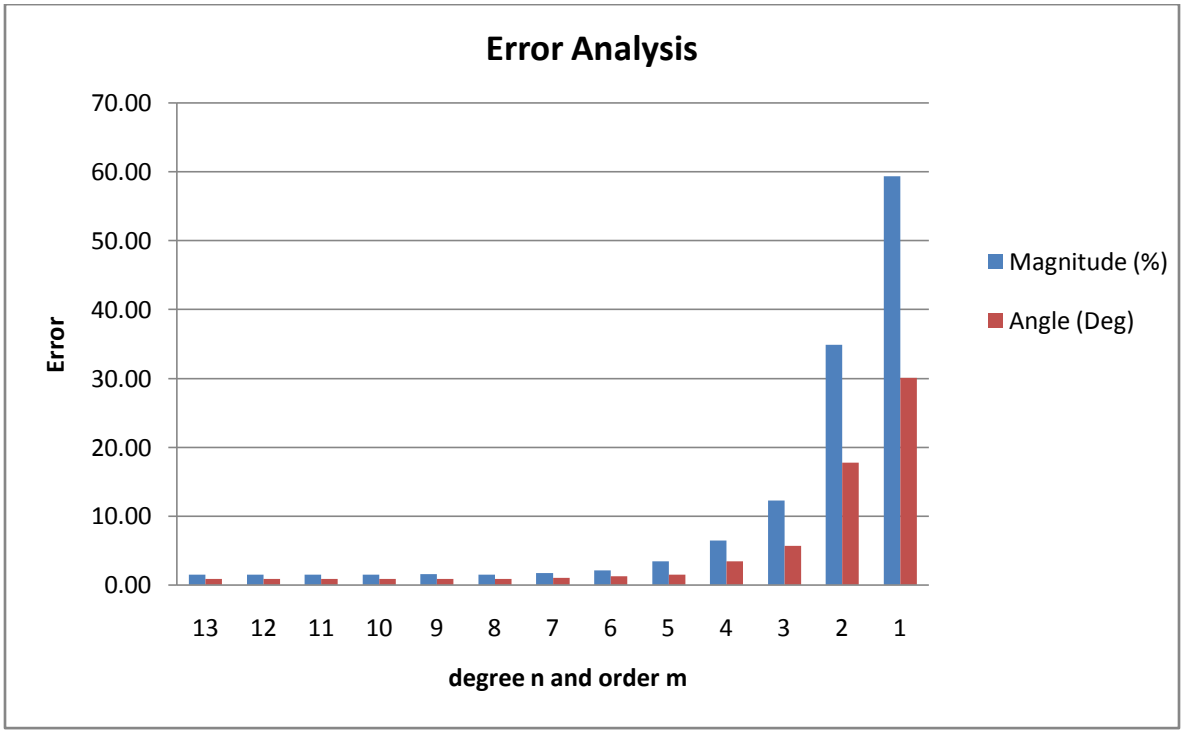

Figure 36: Magnetic Field Model error analysis with varying harmonics

The above analysis shows that the contribution of the higher harmonics is not significant and only induces small deviation in the magnetic field vector computation. Although effect of higher harmonics varies with position but still the overall contribution to the magnetic field is very less, hence the harmonics of degree 8 and order m can provide good accuracy depending on the requirements of ADS of the satellite. So when spherical harmonics of 8 degree and order are used, the computational demand is reduced by approx. 50% with no significant loss in accuracy.

8.3 Verification

The C implantation of the attitude determination algorithm was simulated on a target processor. This was done to verify the accuracy of the algorithm. For this purpose simulation was done for Texas Instruments MSP430 microcontroller. The results show no significant loss in accuracy with 64-bit double precision floating usage, but when algorithm is simulated with 32-bit floating point number there is slight decrease in accuracy. The following graphs show simulation of the sun reference vector model on a target MSP430 device with 32-bit and 64-bit implementation.

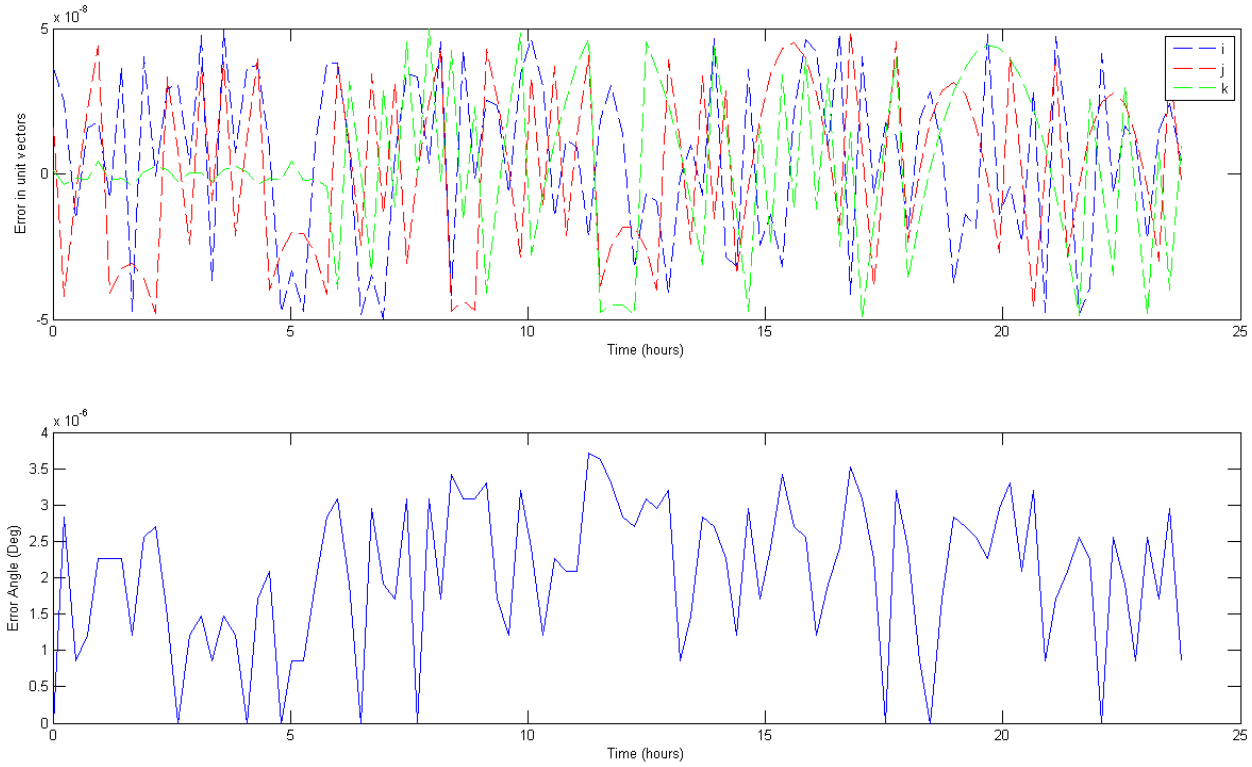

Figure 37: 64-bit Sun reference model implementation results

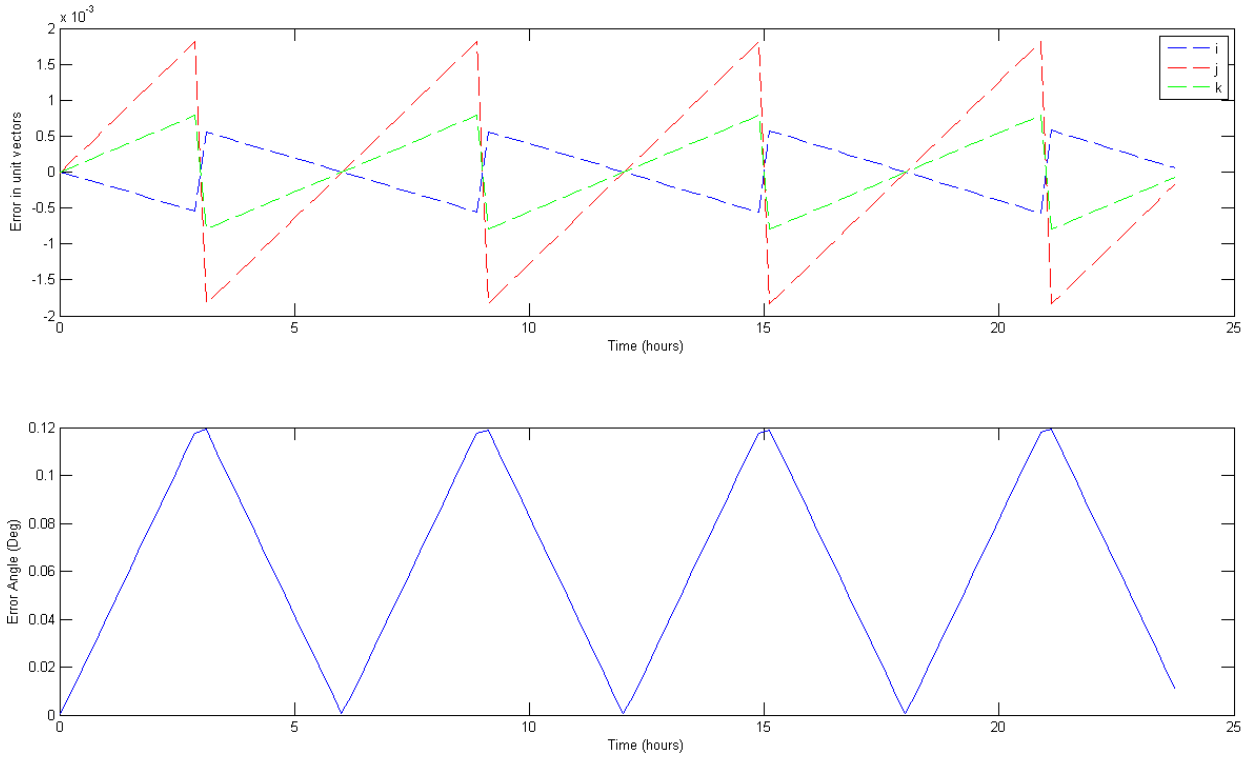

Figure 38: 32-bit Sun reference model implementation results

The comparison of Matlab simulation against MSP430 targeted simulations show that with 64-bit implementation there is no significant difference between the two. While in 32-bit implementation the error in unit vector is of the order of 10^{-3} which leads to an error of about 0.1 in the sun vector. By looking closely in the 32-bit implementation the effects of rounding are easily visible. After further analysis on the cause of this deviation, it was found that the Julian date which is input for reference Sun vector computation is causing this significant error in the sun vector. To verify this source of deviation, 32-bit floating point precision was used in Matlab simulation and the Matlab results also show the same deviation as was in MSP430 implementation. The exact cause of this deviation in Julian date calculation couldn't be found. Hence it was decided to use 64-bit floating point precision which gives comparable results to Matlab simulation.

9 Conclusion

This section revisits the thesis work to summarize the experiences and results achieved during the course of the project. Furthermore some suggestions and future work is indicated which can lead to a more robust and accurate attitude determination system.

9.1 Attitude Estimation Method

Following the simulation results of the Attitude estimation techniques IKF is found most suitable for implementation. First ETA and EQA cannot estimate gyro biases, second both ETA and EQA are vulnerable in the presence of sensor noises and sensor biases. Due to initial concern about the computational requirements of Kalman Filter an Isotropic Kalman Filter was implemented and analyzed. Profiling results have shown that IKF require even less computational power than the sgp4 and the magnetic field reference model.

9.2 Hardware Platform

9.2.1 Microcontroller

Texas instrument ultra low power MSP430 microcontroller can implement attitude determination algorithm with up to 3 Hz Kalman filter update rate, but MSP430 doesn't have enough resources to accommodate attitude control algorithm and some future extensions in the attitude determination algorithm.

9.2.2 FPGA

The results from the hardware implementation on a FPGA show that FPGA based implementation of the attitude determination algorithm is not feasible. This is due to the fact that 64-bit double precision implementation requires more logic cells than are available in current FPGAs.

9.2.3 Digital Signal Processor

Texas Instruments low power DSPs provide the best solution for implementation of the attitude determination algorithm. Fixed point C55x DSP consumes less power than Floating Point C674x DSP but C674x is 20 times faster than C55x due to floating point support. Therefore C674x can run at 20 times less frequency than the C55x to achieve similar results related to Kalman filter

update rate. The power consumption than will be lower in C674x. Idle power for C55x is less than 1mW and approximately 6mW for C674x DSP. Active power for both C55x and C674x is approx. less than 150mW.

Considering some future extensions in the attitude determination and control algorithm Texas Instrument's TMS320C674x low power floating point family is chosen as the best candidate for hardware implementation of the attitude determination algorithm. When using 64-bit double precision floating point operations there is no loss in accuracy compared with the results from Matlab simulation. The ADS algorithm only requires 0.2 Million clock cycles for one cycle computation of the Kalman filter. TMS320C674x DSPs can run at 300 MHz hence there is a lot of room for future extension.

9.3 Future work

This section summarizes some improvement and work future work for implementation of the attitude determination algorithm.

9.3.1 Attitude Estimation

In the implementation of the IKF for attitude determination some improvements can be performed. The calibration of the Kalman filter requires sensor characteristics which have not been done yet. This step can be performed after the testing and verification of the on board sensors which are still in the process of evaluation.

The sun eclipse condition and slowly varying magnetic field at poles have not been simulated. These conditions can be negated by changing the noise covariance of the Sun sensor and the magnetometer.

9.3.2 On board Implementation

In this thesis work, currently only software based simulations have been done. Testing, debugging and verification of the attitude determination software on real hardware are still pending. The sensors data has been simulated from STK and real sensors have not been used. The calibration and testing of the actual on board sensors is a very challenging and time consuming process. The data collected from the testing of the on board sensors will be used to

set the process and noise covariance in the Kalman filter. Achieving high precision and accuracy from the on board sensors will require a lot of testing and verification.

List of Figures

List of Tables

References

[1] Ravandoor, K. (December, 2007). *Design and simulation of software based attitude determination system for the Cubesat UWE-2.* Master Thesis, University Würzburg.

[2] Kurz, O. (December, 2007). *Design and Implementation of an Attitude Determination System For the Cubesat UWE-2.* Master Thesis, University Würzburg.

[3] *Wikipedia, The Free Encyclopedia*. Retrieved 10 5, 2009, from www.wikipedia.com

[4] Vallado, D. A. *Fundamentals of Astrodynamics and Applications* (2nd Edition ed.). The Space Technology Library.

[5] *AGI*. . Retrieved 9. 30. 2009, from https://www.stk.com/resources/help/stk613/helpSystem/extfile/gator/ab-axes.htm

[6] Wiley J Larson, J. W. (2005). *Space Mission Analysis and Design.* Space Technology Library.

[7] Maths - Quaternion Notations - As a quantity similar to axis-angle, Retrieved 10.5.2009 from EuclideanSpace - building a 3D world. http://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/geometric/axisAngle/index.htm

[8] Sidi, M. J. (1997). *Spacecraft Dynamics and Control.* Cambridge University Press.

[9] *The IGRF Model*. Retrieved 10. 3. 2009, from British Geological Survey: http://www.geomag.bgs.ac.uk/gifs/igrf.html

[10] Roithmayr, C. M. *Contributions of Spherical Harmonics, to Magnetic and Gravitational Fields.* Langley Research Center, Hampton, Virginia: NASA/TM–2004–213007

[11] *International Geomagnetic Reference Field*. Retrieved 10.1.2009, from National Oceanic and Atmospheric Administration: http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html

[12] *Celestrak*. Retrieved 10.10.2009, from http://www.celestrak.com/

[13] Roehrich, F. R. (December 1980.). *SpaceTrack Report No. 3, Models for Propagation of NORAD Element Sets*,

[14] John L. Crassidis, S. F. *Contingency Designs for Attitude Determination of TRMM.* Goddard Space Flight Center, Code 712, Greenbelt, MD 20771.

[15] Wertz, J. (1984). *Spacecraft Attitude Determination and Control.* Dordreeht, The Netherlands: D. Reidel Publishing Co.

[16] Roumeliotis, N. T. *Indirect Kalman Filter for 3D Attitude Estimation, A Tutorial for Quaternion Algebra.* Department of Computer Science & Engineering, University of Minnesota.

[17] Crassidis, J. a. (2004). *Optimal Estimation of Dynamic Systems.* Chapman and Hall, ISBN 1-58488-391-X.

[18] Lefferts, E. M. (Sept.-Oct. 1982). Kalman Filtering for Spacecraft Attitude Estimation. *Journal of Guidance, Control and Dynamics, Vol. 5, ,* 417-429.

[19] Mark L Psiaki, M. F., & Pal, P. K. (1989). Three Axis Attitude Determination, via Kalman Filter of Magnetometer data. *J Guidance, VOL. 13, NO. 3* , 506-514.

[20] Itzhack Y. Bar-Itzhack, Richard R. Harman, Optimized Triad Algorithm for Attitude Determination, Flight Dynamics Support Branch, Code 553 NASA Ooddard Space Flight Center Greenbelt,MID 20771

[21] Diwan, M. (2007). *Methodology for Analyzing Complex Algorithms for Small Satellites.* Electrical and Computer Engineering Raleigh, North Carolina.

[22] *fpga4fun-where FPGAs are fun*. Retrieved 9. 21. 2009, from Synthesis and place-and-route (P&R): http://www.fpga4fun.com/FPGAsoftware5.html

[23] *Soft CPU Cores for FPGA*. Retrieved 10. 1. 2009, from 1-Core Technologies: http://www.1-core.com/library/digital/soft-cpu-cores/

[24] *DSP System Design on FPGAs Using the ISE Design Suite 10.1.* Xilinx High-Performance System Design Seminars.

[25] Archide, R. (2009. 10 1). Xilinx Dual Processor Webcast.

[26] B. Fletcher, M. (2005). FPGA embedded processors: Revealing True Systems Performance. *Embedded Systems Conference San Francisco.*

[27] *Impulse CoDeveloper*. Retrieved 10. 9. 2009, from Impulse Accelerated

[28] Mark G. Arnold, C. W. *Verilog Transcendental Functions for Numerical Testbenches.* University of Manchester, Institute of Science & Technology, UK.

[29] *TMS320674x Fixed/Floating point DSP, Product Review.* Texas Instruments, SPRS591A, (June 2009–Revised August 2009).

[30] *TMS320VC5505, Low-Power Fixed-Point Digital Signal Processor* . Retrieved 10. 10. 2009, from Texas Instruments: http://focus.ti.com/docs/prod/folders/print/tms320vc5505.html

A Acronyms

ADS Attitude Determination System

DSP Digital Signal Processor

ECI Earth Centered Inertial Frame

ECEF Earth Centered Earth Fixed

EDA Electronic Design Automation

EDIF Electronic Design Interchange Format

ETA Enhanced TRIAD Algorithm

EQA Enhanced q-method Algorithm

FPGA Field programmable Gate Array

IAGA International Association of Geomagnetism and Aeronomy

IEEE Institute of Electrical and Electronics Engineer

IKF Isotropic Kalman Filter

IGRF International Geomagnetic Reference Field

MEMS Micro Electro Mechanical Systems

MOD Mean Equinox of Date

NORAD North American Aerospace Defense Command

SGP4 Simplified General Perturbations Model

SPI Serial Peripheral Interface

TLE Two Line Element

UWE University of Würzburg Experimental Satellite

B Softwares

MATLAB used for designing and simulating the software

STK Satellite Tool Kit, for simulation and comparison of reference models and sensor data

Impulse CoDeveloper Software and tool for C to FPGA implementation

Xilinx ISE Synthesis and Implementation software for Xilinx FPGAs

Code Composer Studio Software for design, simulation and debugging of algorithm on Texas Instruments DSPs

IAR Embedded Workbench Software for simulation and debugging of Texas Instruments microcontroller

SVN Subversion, a version control system for managing the source code and documents

C Contents of CD

The report is supplemented by a CD containing the source code along with other related documents.

Doc directory contains the Thesis report

Literature directory contains documents related to the thesis in following subdirectories

Datasheets Contains datasheets of the hardware analyzed

Misc contains miscellaneous material related to thesis work

Codes directory contains software of the project divided into two sub directories

Matlab contains Matlab based attitude determination software

C implementation Contains C code of the attitude determination algorithm