2015

SPACECRAFT ATTITUDE CONTROL USING MAGNETIC ACTUATORS

Karthik Mysore Srinivasa
Michigan Technological University

SPACECRAFT ATTITUDE CONTROL USING MAGNETIC ACTUATORS

By

Karthik Mysore Srinivasa

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Mechanical Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2015

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Mechanical Engineering

Department of Mechanical Engineering-Engineering Mechanics

Report Advisor: *Dr. Ossama Abdelkhalik*

Committee Member: *Dr. Mohammad Rastgaar-Aagaah*

Committee Member: *Dr. Nilufer Onder*

Department Chair: *Dr. William W. Predebon*

“Each of us has the cause to think with deep gratitude of those who have lighted the flame within us.”

Albert Schweitzer

Dedicated to those who made this possible:

Mother

Invisible means of power

Acknowledgements

First and foremost, I'd like to thank my advisor Dr. Ossama Abdelkhalik for his support. I am blessed to be acquainted with a person whose help, support and knowledge has made me a better person and help better position myself. His technical insights and adept guidance have been of immense help in my research work. He has been of great support to me in accommodating my schedule. I have also been deeply inspired by his perseverance and commitment to work.

I would like to thank Dr. Mo Rastgaar and Dr. Nilufer Onder for serving on my committee, and for their invaluable guidance for completion of my report.

I would also like to thank Michigan Tech Aerospace Enterprise, for having given me an opportunity for being a part of the project from which I was inspired to pursue my report.

I'd like to thank my relatives for their love and support.

I would like to acknowledge Mr. Shangyan Zou's help in my understanding of the satellite kinematics code. Thanks to Mr. Brandon Jackson, the report and work of whom, I often referred, throughout the course of my research.

I'd like to thank Sudhir Chandramouli for his help in my transition from India to the United States of America and his support throughout my stay so far.

Finally thanks to my friends Subramanya Datta for fuelling the thought of pursuing an advanced degree in the United States of America and Karthik N.R for his logistical help in obtaining necessary forms and documentation for me back in India.

Table of Contents

Abstract

This report presents a study on the problem of spacecraft attitude control using magnetic actuators. Several existing approaches are reviewed and one control strategy is implemented and simulated. A time-varying feedback control law achieving inertial pointing for magnetically actuated spacecraft is implemented. The report explains the modeling of the spacecraft rigid body dynamics, kinematics and attitude control in detail.

Besides the fact that control laws have been established for stabilization around local equilibrium, this report presents the results of a control law that yields a generic, global solution for attitude stabilization of a magnetically actuated spacecraft. The report also involves the use MATLAB as a tool for both modeling and simulation of the spacecraft and controller. In conclusion, the simulation outlines the performance of the controller in independently stabilizing the spacecraft in three mutually perpendicular directions.

1. Introduction

The primary objective of this paper is to design a control algorithm based on a time-varying feedback control law achieving inertial pointing for magnetically actuated spacecraft. In recent years, much work has been dedicated towards the problem of attitude control of rigid bodies using only magnetic actuators. In particular, the feasibility of periodic, time-varying actuators has, only recently, become a topic of increasing research interest. It is better to know the literature already present.

The problem of three-axis Attitude Control using only magnetic actuators was addressed by Wisniewski and Blanke (1998) [12] for a small satellite. The problem is approached by illustrating the loss of controllability with the assumption of linear time invariant model of the satellite. Subsequently, the problem is approached as a nonlinear time-varying problem where two controllers guarantee local stability; one for angular velocity assuming the equilibria around the unit vectors of the orbital frame and the other for attitude. An attempt is made to extend the effectiveness of the controller to guarantee global attitude stability based on the periodicity of both the system and the control law for Torque-Free Rigid Body Motion. A proportional gain is used to accomplish attitude stabilization. Finally, a time-varying gain is introduced along with the proportional gain in an attempt to demonstrate global asymptotic stability in the case of boom upside-down i.e. pointing towards the center of the Earth.

In Satellite Attitude Control using only Magnetorquers (1998) [13], Wang, Shtessel and Yang approach the attitude control problem in two stages. First designing an outer loop within the nonlinear periodic framework using Backstepping for virtual control. Second, designing an inner loop for detumbling control and attitude acquisition, that is, to track virtual signal using Sliding Mode Controller. The attitude parameters are linearized around the origin to prove the closed loop linearity of the system and thereby guaranteeing the local asymptotic stability according to Floquet's Theorems. The saturation is also taken into account in the control torque and fairly good results have been established in terms of performance of the controller for an isoinertial spacecraft in the Lower Earth Orbit (LEO).

Another significant work in the field of Attitude Control using only magnetic torque rods as actuators has been done by Lovera and Astolfi in Global Attitude Regulation using Magnetic Control (2001) [14]. The problem is approached by using a low-gain proportional-derivative-like control law to prove the global asymptotic stability of the system. The control law has also been extended for the magnetic control of Earth pointing satellite. The control law yields (almost) global solution to demonstrate asymptotic stability.

Magnetic Attitude Control problem has been approached in Attitude Stabilization of a Satellite by Magnetic Coils (2002) [15] by Bushenkov, Ovchinnikov and Smirnov where in the stabilization problem of a small satellite is addressed using only magnetic coils based on the fundamental assumption that geomagnetic field is periodic, the motion is periodic and the magnetic field at any instant is known from the three-axis magnetometer measurements. The problem is divide into two stages; One the magnetic stabilization and the other gravitational stabilization. The magnetic stabilization is approached by proving that the attitude approaches the equilibrium which is a constant vector to maintain the asymptotic stability of the system whereas the gravitation stabilization problem is addressed by proving that coincidence of magnetic field vector and the constant vector relating magnetic field vector and the quaternion vector in their respective cross products which gives rise to the fact that the magnetic field transformation matrix is an Identity Matrix. These approaches however, are designed for stabilization from the initial conditions and don't take into account, the disturbance torque as time progresses. In this way, it guarantees an (almost) global solution to the magnetic field stabilization problem.

Lovera and Astolfi in Spacecraft Attitude Control using Magnetic Actuators (2004) [1] built on the work done in Global Attitude Regulation using Magnetic Control (2001) approached the problem of inertial pointing of spacecraft with only magnetic coils as actuators using static attitude with rate feedback and dynamic attitude feedback. A global solution is guaranteed for the former in the case of static attitude and rate feedback with the assumption that the quaternion vector equilibria lies around zero target attitudes. This is accomplished by extending the PD-like Control

Law considered in Lovera and Astolfi (2001) with the introduction of position and velocity error constants and using the Lyapunov Stability criterion with appropriate scaling. Whereas, the latter yields an (almost) global solution ONLY in the case of isoinertial spacecraft. This is demonstrated with the assumption that the inertial magnetic field transformation matrix having values less than unity.

A more recent work by Silani and Lovera in Magnetic Spacecraft Attitude Control: A survey and some new results (2005) [16] explore various linear and nonlinear attitude control methods and corresponding results. The former encapsulates classical control, optimal periodic control and robust control methods while the latter is explored from the Lyapunov Stability perspective.

a) The first gives insight towards control via periodicity assumption of geomagnetic field and the assumption of time-invariant approximations of slow closed-loop system dynamics and the results of which are similar to Spacecraft Attitude Control using Magnetic Actuators (2004).

b) Optimal Control is studied from the Linear Quadratic perspective with state or output feedback. Again, assuming the periodicity of geomagnetic field, the marginal stability issue is addressed by incorporating 'J' secular effects and cyclic external disturbances into the plant model which acts as a time-periodic filter and provides a time-periodic state feedback.

c) Robust Control is proposed based on the H∞ approach with periodic state feedback assumption and by placing appropriate constraints on the control torque.

Additionally, it offers a new approach towards pursuing the attitude control problem using only magnetic actuators based on prediction of parameters in discrete time intervals. This is accomplished by assuming the spacecraft to be a time-invariant and modeling it using state-space representation with its actuators interacting with the time-varying geomagnetic magnetic field to regulate the attitude. A time delay property is used to predict the values at subsequent instants every time the sample period has elapsed while only the first time instant in every series is used to predict the

values at subsequent instants. Actuator saturation is placed on the coils by placing constraints on the magnetic dipole moment. Stability Analysis is done using either LQ or H∞ by identifying the appropriate weighting matrices. This approach has been used in Psiaki (2001). An (almost) global solution is obtained at the expense of performance of the controlled variables due to actuator constraints.

2. Spacecraft Model

The kinematic model of the rigid spacecraft simulates the behavior of the spacecraft in the orbit.

2.1 Coordinate Frames

For proper attitude estimation and control, the rigid body under consideration, the spacecraft, can be described in various reference frames. The figure below describes the reference frames used in this report.

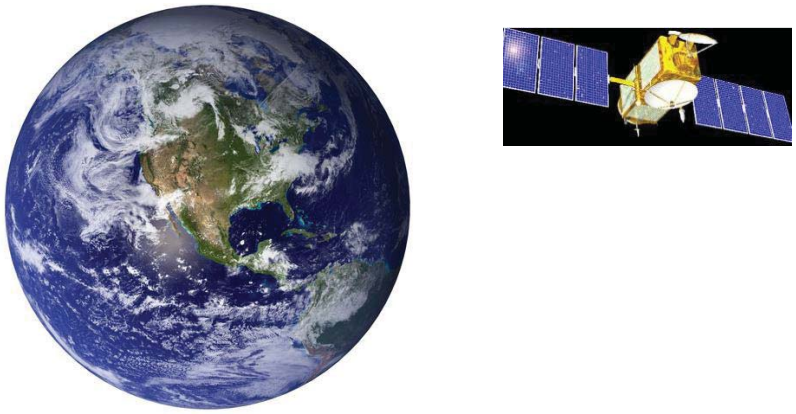

Figure: Coordinate Reference Frames [9],[11]

The three reference frames adopted in this report are as follows:

a) Inertial Frame

The Earth-centered inertial reference (ECI) frame: The origin of the axes in this coordinate reference frame is fixed at the center of the earth.

- The X-axis is parallel to the line of nodes and is positive in the Vernal Equinox direction.
- The Z-axis is parallel to the earth's geographic north-south axis and pointing north.
- The *y*-axis satisfies the right-handed orthogonal triad.

b) Non-inertial Frame

The Earth-centered earth-fixed (ECEF) frame: The origin of the axes in this coordinate reference frame is also fixed at the center of the earth but the coordinate system rotates with the rotation of the earth.

c) Body Frame

The origin of the axes in the body reference frame is fixed at the satellite center of mass and these are axes are assumed to coincide with the body's principal inertia axes; the Z-axis always pointing inertially.

d) Controller Frame

For simplicity, the axes of the controller frame are also assumed to be coinciding with the satellite body axes and therefore the principal inertia axes. The origin of the axes in the controller reference frame always coincides with the satellite body axes but the direction in which each axis is aligned can be modified, depending on the requirements, in which case the attitude propagation has to be transformed to the controller frame with respect to body frame. This is accomplished by a small piece of code in the control algorithm.

2.2 Attitude Parameters

2.2.1 Euler Angles

A commonly use set of attitude parameters are Euler angles. The spacecraft attitude (orientation) is commonly described through the yaw, pitch and roll Euler angles and are usually measured in the satellite body frame relative to the earth-centered inertial frame. There are 12 possible sets of Euler angles; the sequence of which can be used to describe the rotation of the spacecraft around respective axes in the sequence.

However, in any corresponding set, a 180 degrees change in the Euler angle results in a singularity as determined by the rotation matrix for that particular set sequence. For example, rotation matrix for 3-2-1 Euler angle sequence is as shown below.

$$\begin{bmatrix} cos\theta cos\Psi & cos\theta sin\Psi & -sin\theta \\ (sin\emptyset sin\theta cos\Psi - cos\emptyset sin\Psi) & (sin\emptyset sin\theta sin\Psi + cos\emptyset cos\Psi) & sin\emptyset cos\theta \\ (cos\emptyset sin\theta cos\Psi + sin\emptyset sin\Psi) & (cos\emptyset sin\theta sin\Psi - sin\emptyset cos\Psi) & cos\emptyset cos\theta \end{bmatrix}$$

The matrix above shows the rotation matrix for 3-2-1 Euler angle sequence. Here, Ψ, θ and Φ are yaw, pitch and roll Euler angles respectively.

It can be noticed that, the inverse transformation of the third element in the first row yields the pitch angle 'θ', which clearly varies between -1 and +1 for values in multiples of -90 degrees and +90 degrees respectively. Therefore, there exists a geometric singularity when pitching up or down 90 degrees. This geometric singularity manifests itself in the kinematic differential equation which will be presented later. Therefore, we use another set of attitude parameters called Euler Parameters (also called Quaternions).

2.2.2 Quaternions[5]

The Quaternions are another popular set of attitude coordinates which are used to describe arbitrary, large rotations. The Euler angles are on the other hand, easy to compute/develop and easy to visualize, but computationally intense. There is also a singularity problem when describing the attitude in terms of Euler angles for reasons discussed above. Therefore, Euler parameters (also called Quaternions), offers an effective method for describing the attitude coordinates. These Quaternions are based on Euler's Rotational theorem which says the relative orientation of two coordinate systems can be described by only one rotation about a fixed-axis.

The Euler parameter vector 'q' is defined in terms of principal rotation elements as follows:

$$q_0 = \cos\left(\frac{\emptyset}{2}\right)$$

$$q_1 = \mathrm{e}_1 \sin\left(\frac{\emptyset}{2}\right)$$

$$q_2 = \mathrm{e}_2 \sin\left(\frac{\emptyset}{2}\right)$$

$$q_3 = \mathrm{e}_3 \sin\left(\frac{\emptyset}{2}\right)$$

It is evident that, since the sum of squares of principal rotation elements is equal to '1' because of it being a unit vector, the 'q_i' satisfies the holonomic constraint

$$q_0^2 + q_1^2 + q_3^2 + q_4^2 = 1$$

This constraint geometrically describes a four-dimensional unit-sphere. The Direction Cosine Matrix (DCM) can be written in terms of Quaternions as follows:

$$[C] = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

It can be observed that 'q' and '-q' yields the same Direction Cosine Matrix. Given a direction Cosine Matrix, the inverse transformation can be performed to determine the Quaternions as follows:

$$q_0 = \pm\frac{1}{2}\sqrt{C_{11} + C_{22} + C_{33} + 1}$$

$$q_1 = \frac{C_{23} - C_{32}}{4q_0}$$

$$q_2 = \frac{C_{31} - C_{13}}{4q_0}$$

$$q_3 = \frac{C_{12} - C_{21}}{4q_0}$$

It can be noticed that the above 3 equations have a mathematical singularity whenever $q_0 \to 0$. This corresponds to the 'q' vector describing a 180° principal rotation.

It can be noticed that, the Quaternion vector consists of a scalar part (which is the first element) and a vector part. The transformation of quaternion from one reference frame to another can be incorporated as follows:

$$\vec{v}_C = {}^{C}_{S}q^{-1} \otimes [{}^{0}_{v_S}] \otimes {}^{C}_{S}q$$

The above equation demonstrates the rotation of vector by Quaternion from satellite body frame to controller frame.

2.3 Kinematic Equations

The parameterization of the four Euler parameters that describe the attitude kinematics can be represented by the kinematic differential equations as described below.

$$\dot{\boldsymbol{q}} = W(w)\boldsymbol{q}$$

Where, $\boldsymbol{q} = [q_1\ q_2\ q_3\ q_4]^T = [q_1\ \boldsymbol{q}^T]$ is a vector of unit norm ($\boldsymbol{q}^T\boldsymbol{q} = 1$) Euler parameters in which $'q_1'$ is a scalar and the remaining 3 quaternions for a vector.

Also,

$$W(\omega) = \frac{1}{2} * \begin{bmatrix} 0 & +\omega_Z & -\omega_Y & +\omega_X \\ -\omega_Z & 0 & +\omega_X & +\omega_Y \\ +\omega_Y & -\omega_X & 0 & +\omega_Z \\ -\omega_X & -\omega_Y & -\omega_Z & 0 \end{bmatrix}$$

This matrix can be obtained by differentiating 'q_i' from the following equation:

$$q_0 = \pm\frac{1}{2}\sqrt{C_{11} + C_{22} + C_{33} + 1}$$

By transmutation the above matrix can be equivalently written as,

$$\widetilde{W}(q) = \frac{1}{2} * \begin{bmatrix} +q_4 & -q_3 & +q_2 \\ +q_3 & +q_4 & -q_1 \\ -q_2 & +q_1 & +q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix}$$

This matrix helps in expressing the attitude kinematic differential equation in terms of quaternions and angular velocity.

2.4 Control Torque

The attitude control of spacecraft using magnetic actuators uses three magnetic torque rods aligned with spacecraft principal inertia axes. The torque produced by these magnetic torque rods in each direction is governed by the following equation.

$$\boldsymbol{T_{coils} = m_{coils}\, x\, \tilde{b}(t)}$$

It can be noticed that the torque produced is a cross product of magnetic dipole moment and the time-varying magnetic field. The SI unit of magnetic dipole moment is in Amperes-meters^2.

The strength of earth's magnetic field is time-varying as the spacecraft moves along the orbit and is expressed in the spacecraft Body Frame. The SI unit of earth's magnetic field is Tesla.

The computation of magnetic field from the World Magnetic Model (WMM) is discussed in the next section.

The control torque can also be computed using the following equation

$$T_{coils} = S\left(\tilde{b}(t)\right) m_{coils}$$

Where, S($\tilde{b}(t)$) is a skew-symmetric matrix depending on the earth's magnetic field.

$$S\left(\tilde{b}(t)\right) = \begin{bmatrix} 0 & +b_3 & -b_2 \\ -b_3 & 0 & +b_1 \\ +b_2 & -b_1 & 0 \end{bmatrix}$$

It can computed from the above equation and can also be seen from the above matrix that, the rank of the skew-symmetric matrix $S\left(\tilde{b}(t)\right) = 2$ (since $b_0(t) \neq 0$ along all orbits of practical interest), and the kernel of $S\left(\tilde{b}(t)\right)$ is given by the vector $\tilde{b}(t)$ itself, it is not possible control torques along the direction of $\tilde{b}(t)$.

2.5 Magnetic Field Computation

The strength of earth's magnetic field is computed using the World Magnetic Model (WMM).

2.5.1 World Magnetic Model (WMM) [6]

2.5.1.1 Overview

The World magnetic Model (WMM) is provided by the National Oceanographic and Atmospheric Administration's National Geophysical Data Center (NOAA/NGDC) and is updated and released every 5 years.

The World Magnetic Model (WMM) is a 12^{th} order and degree spherical Harmonic function. The magnetic field modeled in the World Magnetic Model (WMM) is represented by the term "main field" which refers to the portion of the earth's magnetic field at Epoch 2010.0. The Secular Variation (SV) is also taken into account since the earth's liquid-iron outer core which contributes to the majority of the earth's magnetic field intensity used in our computation called "core field", since the core field changes perceptibly from year to year. This Secular Variation (SV) is accounted for by a linear SV model in the World Magnetic Model (WMM). But, due to non-linear variations, the WMM has to be updated every 5 years. The Epoch year considered for the purpose of this report is Epoch 2010; although the magnetic field was approximated during the most recent execution which was accomplished using a MATLAB function which will be discussed in the following sections.

The World Magnetic Model (WMM) representing the earth's magnetic field at Epoch 2010.0 is shown below.

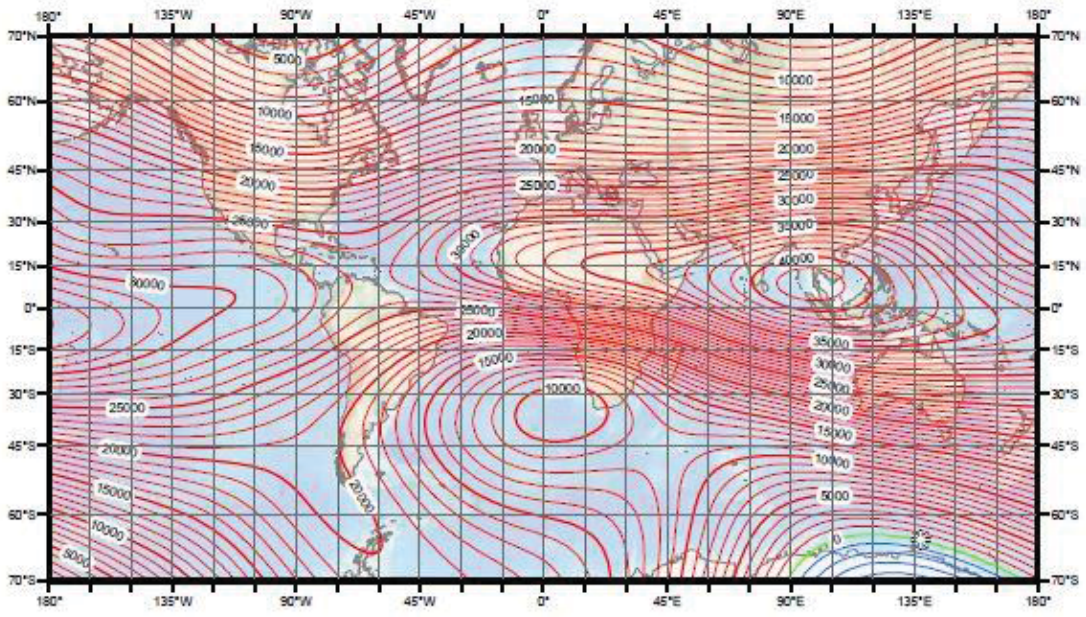

Figure: World Magnetic Model at Epoch 2010.0 [6]

The magnetic field is expressed as a negative spatial gradient of the scalar potential 'V' as follow:

$$B_m(\boldsymbol{\lambda}, \boldsymbol{\varphi'}, \boldsymbol{r}, \boldsymbol{t}) = -\boldsymbol{\nabla V}(\boldsymbol{\lambda}, \boldsymbol{\varphi'}, \boldsymbol{r't})$$

The magnetic field in terms of special harmonics of the scalar potential as:

$$V(\boldsymbol{\lambda}, \boldsymbol{\varphi'}, \boldsymbol{r}, \boldsymbol{t}) = \boldsymbol{a}\{\sum_{\boldsymbol{n=1}}^{\boldsymbol{N=12}} \sum_{\boldsymbol{m=0}}^{\boldsymbol{n}} (\boldsymbol{g_n^m(t)cos(m\lambda)} + \boldsymbol{h_n^m(t)sin(m\lambda)}) \left(\frac{\boldsymbol{a}}{\boldsymbol{r}}\right)^{\boldsymbol{n+1}} \breve{\boldsymbol{P}}_{\boldsymbol{n}}^{\boldsymbol{m}}(\boldsymbol{sin\varphi'})\}$$

Where,

N = 12 is the degree of expansion of the WMM

a = 6371200 m is the geomagnetic reference radius

$(\boldsymbol{\lambda}, \boldsymbol{\varphi'}, \boldsymbol{r})$ are the longitude, latitude and the radius in spherical geocentric reference frame and

$\boldsymbol{g_n^m(t)}$ $\boldsymbol{and}$ $\boldsymbol{h_n^m(t)}$ are the time-dependent Gaussian coefficients of degree 'n' and order 'm' describing the earth's magnetic field

$\breve{\boldsymbol{P}}_{\boldsymbol{n}}^{\boldsymbol{m}}(\boldsymbol{x})$ are the Schmidt semi-normalized associated Legendre's function

2.5.1.2 Usage of World Magnetic Model

The magnetic field necessary for the attitude control of spacecraft using magnetic actuators is provided by the following MATLAB function [17].

$$[\mathrm{b}_0(t)] = \mathrm{wrldmagm}\big(\mathrm{Altitude}, \mathrm{Latitude}, \mathrm{Longitude}, \mathrm{decyear}(2014{,}12{,}8)\big) * 1\mathrm{E} - 9;$$

This function takes Altitude, Latitude and Longitude in the Geodetic Frame as its inputs and also the day and month of the year to account for the secular variations and return the magnetic field in the Inertial Frame.

The factor 1E-9 converts the magnetic field from nanoTesla which is the default unit of the magnetic field output form the World Magnetic Model (WMM) to Tesla.

For example, the day on which the last execution was run is 8th December 2014. Therefore, the syntax is the year in which the execution is run '2014' and the month '12' and day '8'. This date vector is converted into a decimal year suitable for the WMM by the function **"decyear"**.

It is very important to note that, when the spacecraft is motion in the orbit, the altitude, latitude and longitude changes according to the motion of the spacecraft. Therefore, there has to be a mechanism in place to update the altitude, latitude and longitude dynamically. In simulation, this is accomplished by the SGP4 Orbit Propagator [3].

2.5.1.3 Simplified Perturbation Model (SGP4)

The SGP4 is a Simplified Perturbation Model that is used with the Two-Line element sets (TLEs) produced by NORAD and NASA. The SGP4 model outputs the orbital state vectors relative to Earth-Centered Inertial Frame (ECI Frame).

Note: It is very important to note that the SGP4 Orbit Propagator requires the Greenwich Mean Time (GMT) to be used to calculate the time since the Epoch Date. But, for the purpose of simulation, we shall stick with time elapsed since the start of simulation on the Decimal Year (decyear).

An example of the Two-Line Element (TLEs) is as follows:

```
ISS (ZARYA)
1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0 2927
2 25544  51.6416 247.4627  0006703 130.5360 325.0288
15.72125391563537
```

The syntax[3] of the SGP4 setup is as follows:

```
SGP4_Setup(longstr1, longstr2) % Orbit Propagator Setup
```

Orbital State Vectors are then expressed in the Geodetic Frame relative to ECI Frame by using appropriate rotations from ECEF w.r.t to ECI Frame to Geodetic Frame w.r.t ECI Frame. Finally, the corresponding elements from the Orbital State Vector are fed into the World Magnetic Model (WMM) to obtain the intensity of earth's magnetic field relative to the ECI Frame.

```
ECEF_Init = sgp4(T(i))*1000; %Position Vect in ECEF w.r.t ECI Frame
LLA_Init = ecef2lla(ECEF_Init'); %Position Vect in Geodetic w.r.t ECI
Latitude = LLA_Init(1); % Real-Time Latitude
Longitude = LLA_Init(2); % Real-Time Longitude
Altitude = LLA_Init(3)/1000; % Real-Time Altitude
```

TLE Lines 1 and 2 meaning attached in the Appendix

2.5.2 Magnetic Field Usage

Since, in orbital dynamics we mostly deal with Spacecraft Body Frame coordinates, we can express the magnetic field vector in Body Frame in terms of Attitude Matrix 'A(q)' and magnetic field vector expressed in ECI coordinates 'b_{0_t}' as follows:

$$b(t) = A(q)b_0(t)$$

The orthogonality of 'A(q)' implies that $\left|\left|b(t)\right|\right| = \left|\left|b_0(t)\right|\right|$

The Attitude Matrix ‘A(q)’ is obtained using the following syntax in MATLAB:

```
A_q = quat2dcm(q_init_s');
```

Where, the function ‘quat2dcm’ generates a direction cosine matrix using the quaternions and ‘q_init_s’ is the equilibrium to be stabilized and is dealt in further detail in the section pertaining to the Control Law.

This magnetic field ‘b(t)’, which is in the Spacecraft Body Frame is used in the computation of magnetic dipole moment and also the control torque (as discussed earlier).

2.6 Dipole Moment Calculation

The magnetic moment determines the torque, the spacecraft will experience in an external magnetic field, in our case earth’s magnetic field. The spacecraft dipole moment occurs due a current loop across the torque rods interacting with the earth’s magnetic field. The residual dipole moment, on the other hand, is of paramount importance in determining the disturbances that might/will eventually arise due to the interaction with earth’s magnetic field.

It is very important to note that the value of the control and residual dipole moment depends on the size of the spacecraft (which will be taken into account in the Inertia matrix in the control law discussed in further detail in the following section) and whether the on-board compensation is provided or not. The SI unit of dipole moment is Ampere-meter^2 and is in the range of 0.1 Amp-m^2 to 20 Amp-m^2, although it is not uncommon to exceed this range. This is frequently monitored in a feedback form,

$$m_{coils} = \frac{1}{\left|\left|b_0(t)\right|\right|^2} S^T\left(\tilde{b}(t)\right) u$$

Where, m_{coils} = Control Magnetic Dipole Moment in Amp-m^2

$b_0(t)$ = Magnetic Field expressed in Spacecraft Body Frame in Tesla

u = Control Input dealt in the following section

$S^T(b(t))$ = The transpose of the skew-symmetric matrix which is a function of magnetic field discussed in the Magnetic Field section.

$$S^T\left(\tilde{b}(t)\right) = \begin{bmatrix} 0 & -b_3 & +b_2 \\ +b_3 & 0 & -b_1 \\ -b_2 & +b_1 & 0 \end{bmatrix}$$

3. Stabilization using State Feedback Control

The type of control [1] used is a full state feedback with both attitude and angular rate fed back into the spacecraft kinematics and control is calculated for the next time instant for the dynamics (dynamic behavior) of the system.

The control law [1] that serves the above purpose is a PD like control law as follows:

$$u = -\left(\varepsilon^2 k_p \boldsymbol{q} + \varepsilon k_v I\omega\right)$$

Where, u = Control Input feedback for stabilization

k_p = Proportional Constant

q = 4x1 vector with both scalar and a vector component of the quaternion

k_v = Velocity Constant

I = Inertia Matrix (Inertia of Spacecraft with Principal Inertia Axes)

ω = Angular Velocity vector of the Spacecraft

It is very important to note that, without loss of generality, we are assuming [1] that the stabilizable equilibrium is given by $(0, \bar{\boldsymbol{q}})$ where $\bar{\boldsymbol{q}} = [1\ 0\ 0\ 0]^T$

4. Spacecraft Attitude Dynamics[5]

It is a very well know fact that the time derivative of the Angular Momentum is Torque which is the famous Euler's equation. In Satellite Body Frame, this can be written using Transport Theorem as, [4]

$$\dot{H} = \frac{d}{dt}(H) + \omega \times H = Torque$$

The Angular Momentum of a body about its center of mass is given by [4]

$$H = [I]\omega$$

The first equation becomes,

$$\frac{d}{dt}(H) = \frac{d}{dt}([I]\omega) + [I]\frac{d}{dt}(\omega)$$

We know that in Satellite Body Frame, using Transport Theorem,

$$\dot{\omega} = \frac{d}{dt}(\omega) + \omega \times \omega = \frac{d}{dt}(\omega)$$

Therefore, the derivative of Angular Momentum becomes,

$$\frac{d}{dt}(H) = \frac{d}{dt}([I]\omega) + [I]\frac{d}{dt}(\omega) = [I]\dot{\omega}$$

Substituting the above in the first equation, the Torque can be expressed as,

$$Torque = [I]\dot{\omega} + \omega \times ([I]\omega)$$

This leads to the famous Euler Rotational Equations of Motion,

$$\boldsymbol{[I]\dot{\omega} = -[\tilde{\omega}][I]\omega + Torque}$$

The $-[\tilde{\omega}]$ term is introduced when the cross product is removed. The tilde operator serves the purpose of being able to express $[\omega]$ in the form of a more easy to work with, skew symmetric matrix as follows:

$$[\tilde{\omega}] = -S(\omega) = \begin{bmatrix} 0 & -\omega_3 & +\omega_2 \\ +\omega_3 & 0 & -\omega_1 \\ -\omega_2 & +\omega_1 & 0 \end{bmatrix}$$

$$-[\tilde{\omega}] = S(\omega) = \begin{bmatrix} 0 & +\omega_3 & -\omega_2 \\ -\omega_3 & 0 & +\omega_1 \\ +\omega_2 & -\omega_1 & 0 \end{bmatrix}$$

The above mentioned Euler Rotation Equations of Motion can be used to express the spacecraft attitude dynamics as,

$$\boldsymbol{[I]\dot{\omega} = S(\omega)[I]\omega + Torque}$$

5. Simulation Results

The performance of the state feedback control law has been assessed for variety of parameters. Specifically, the iterations carried out have been focused on the effect of control law on the spacecraft model [2] for varying moments of inertia and initial quaternion or the initial orientation of the spacecraft.

The simulation results presented have been obtained using MATLAB as a tool for both designing the state feedback control and the spacecraft model [5]. The simulation results have also been carried out for different periods of time the satellite is in the orbit to demonstrate the consistency of the results obtained.

The two Inertia Matrices considered for spacecraft attitude control using magnetic actuators are I = diag [27, 17, 25] and I = diag [10, 10, 10]. The orbit is defined in the Two-Line element (TLE) Set presented below

```
longstr1 = '1 25544U 14067A   14342.51579142  .00005418  00000-0
10235-3 0  3046';
longstr2 = '2 25544  51.6494 243.7352 0003674 255.5105 239.9716
15.50141065841738';
```

The proportional and velocity constants in the PD- like control law are the controller parameters and are chosen to be $k_p = 50\ and\ k_v = 50\ and\ \varepsilon = 0.001$

5.1 Simulation Results for Standard Inertia Matrix = [27, 17, 25] kg- m^2

5.1.1 Angular Velocity simulation results

Simulation Time = 100,000 seconds

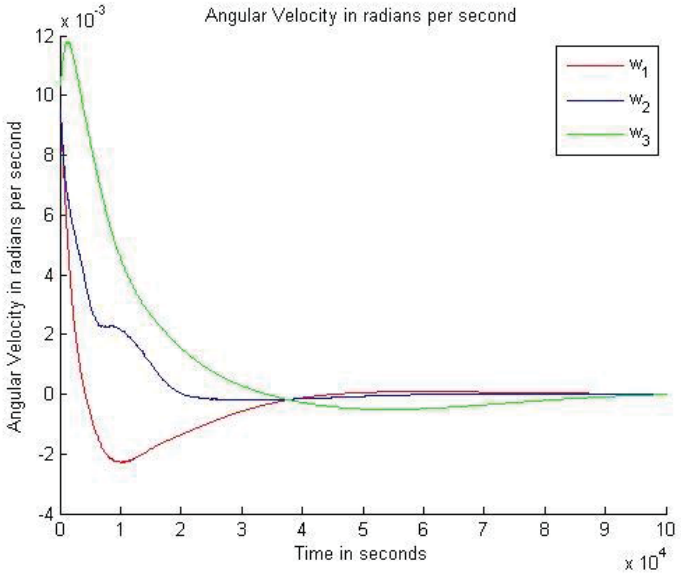

Figure1: Angular Velocity in radians per second

Simulation Time = 150,000 seconds

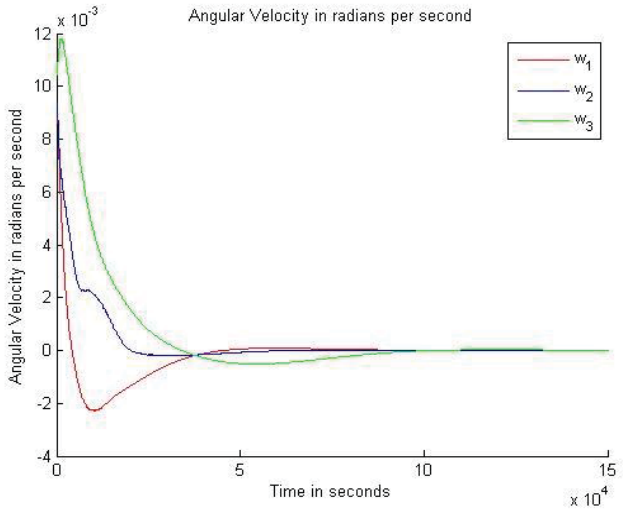

Figure2: Angular Velocity in radians per second

It can be observed from the above figures that, the initial angular velocity was ω = [0.01, 0.01, 0.01] radians/second and eventually the angular velocity is negligible or driven towards ω = [0, 0, 0]. This is very much in conformance with the control law due to the kinematics and dynamics of the system that all the trajectories of the system are such that $q \rightarrow 0$ and $\omega \rightarrow 0$. The consistency of the results has also been demonstrated in the figure (2) above where the angular velocity maintains the trend at 150,000 sec. For the sake of comparison, the results from literature [1], the control law of which has been used to obtain simulation results indicate the same trend.

Simulation Time = 100,000 seconds

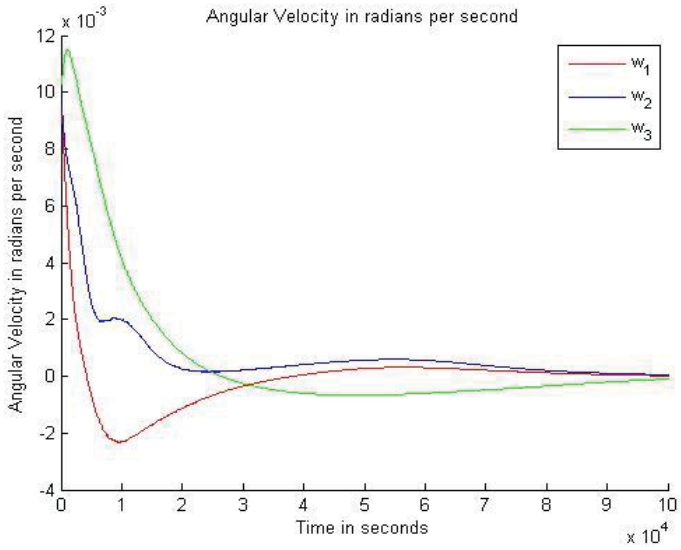

Figure3: Angular Velocity in radians per second

It can be observed from the above figure that the initial quaternion is NOT q = [1;0;0;0]. Therefore, it can be concluded that as long as the desired quaternions define a stabilizable equilibrium, the angular velocity is stabilized about that stabilizable equilibrium at least in the case where the spacecraft has small perturbations.

5.1.2 Quaternion simulation results

Simulation Time = 100,000 seconds

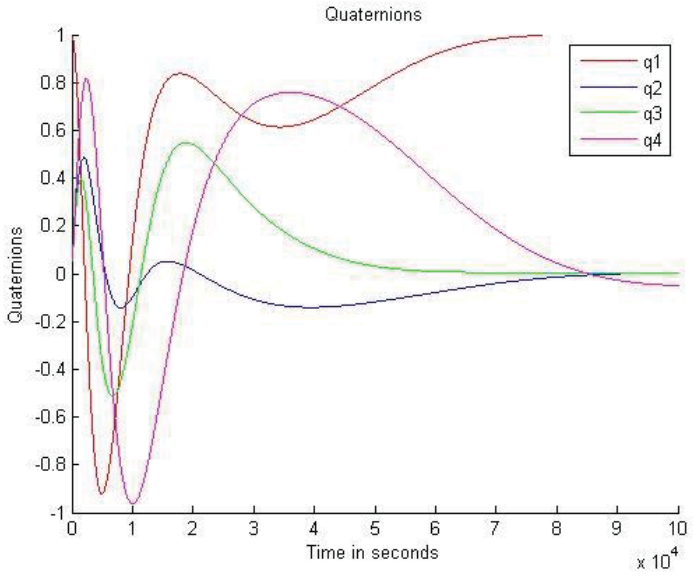

Figure4: Quaternions

Simulation Time = 150,000 seconds

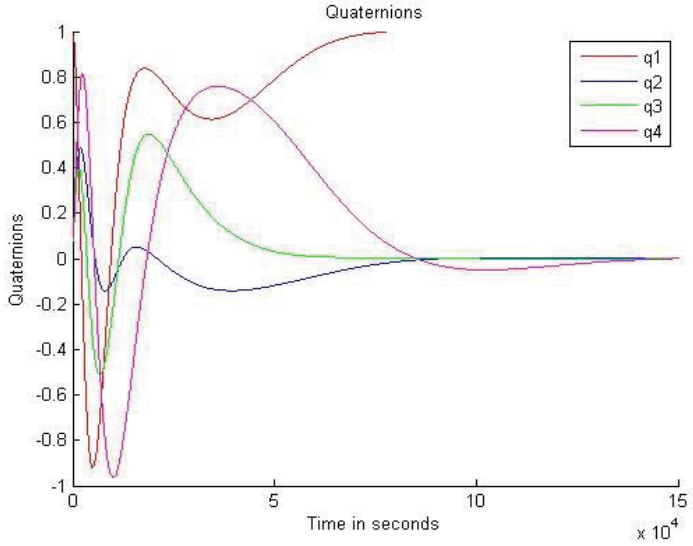

Figure5: Quaternions

It can be observed from the above figures that, the initial quaternion test cases were q = [1, 0, 0, 0]. Eventually the quaternions are driven towards q = [1, 0, 0, 0] which is the stabilizable equilibrium. This consistency has also been demonstrated in the figure (6) above where the quaternions maintain the trend at 150,000 sec.

Simulation Time = 100,000 seconds

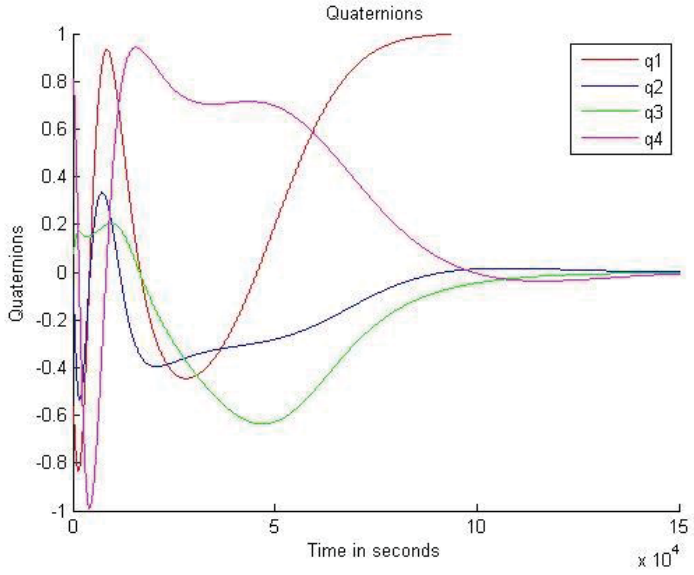

Figure6: Quaternions

Similar to angular velocity, it can be observed from the above figure that the initial quaternion is **NOT q** = [1;0;0;0]. It can be concluded from the above results that as long as the desired quaternions define a stabilizable equilibrium, the closed loop linear time- varying control law ensure that the spacecraft is oriented with respect to the desired quaternion at least in the case where the spacecraft experiences small perturbations. It is also very important to note that the above results can be **globally** achieved only when the proportional and velocity gains are greater than zero [1].

For the case where the spacecraft experiences large perturbations, an appropriate analysis of non-linear control closed loop system and kinematic equations is necessary to establish the domain of perturbations for which closed-loop stabilization is achieved and can guarantee only an almost global (definitely local) solution for stabilization [8].

5.1.3 Control Torque simulation results

Simulation Time = 100,000 seconds

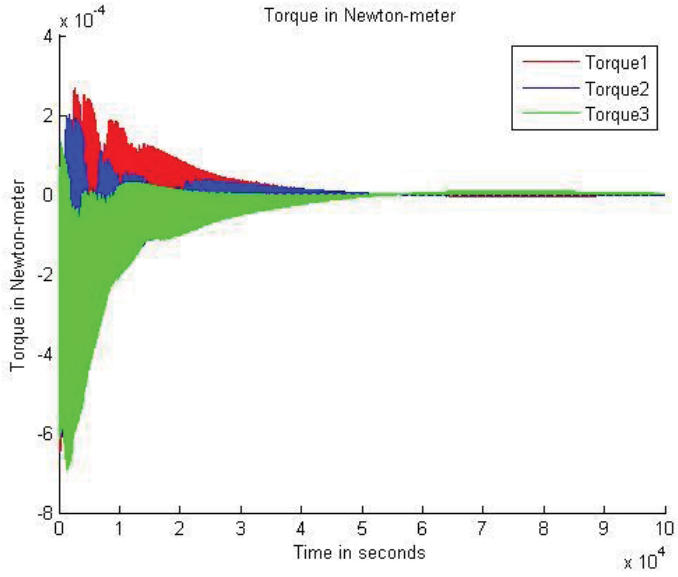

Figure7: Control Torque in N-m

Simulation Time = 150,000 seconds

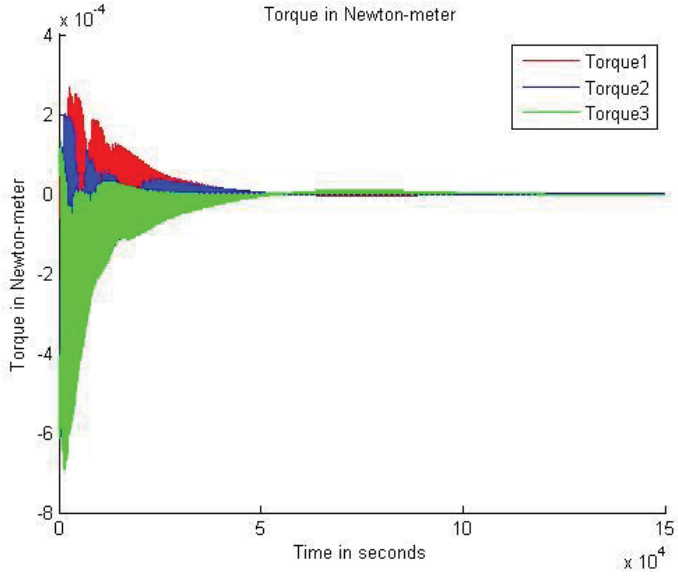

Figure8: Control Torque in N-m

The simulation results of the control torque in N-m shows the good performance of a state feedback controller in stabilizing the high angular rate without taking actuator saturation into account.

Simulation Time = 100,000 seconds

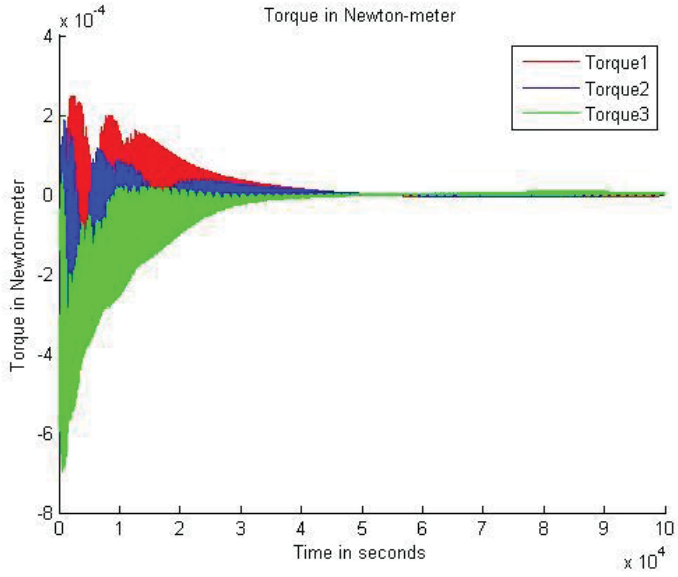

Figure9: Control Torque in N-m

5.1.4 Control Dipole Moment Simulation Results

Simulation Time = 100,000 seconds

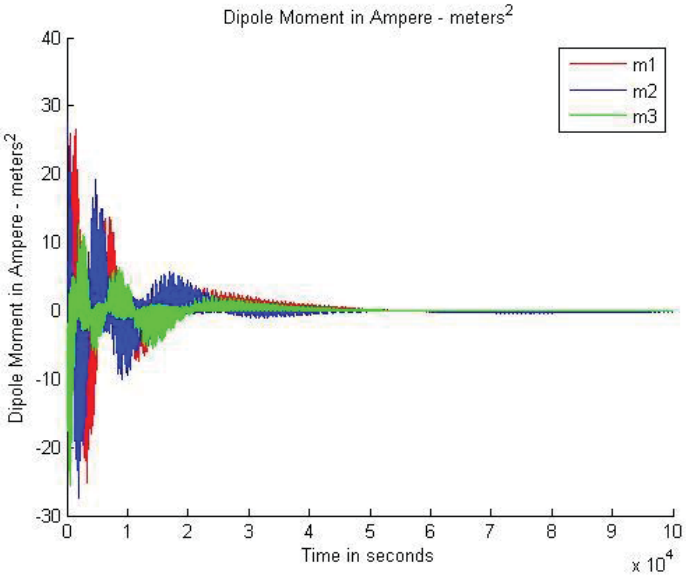

Figure10: Control Dipole Moment in A-m^2

Simulation Time = 150,000 seconds

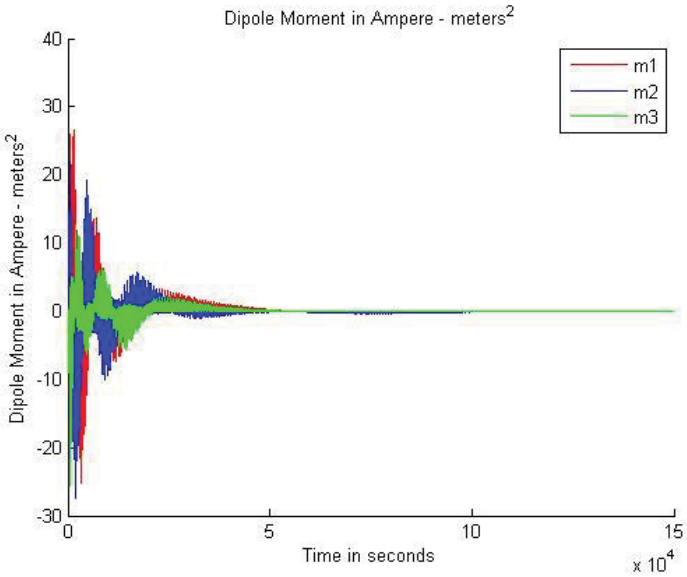

Figure11: Control Dipole Moment in A-m^2

The simulation results obtained can be compared to that presented in literature [1] and the difference with the initial dipole moment is due to the residual dipole moment of m_0 = [0.5; 0.5; 0.5] A-m^2 taken into consideration in the literature [1]. The fact that the control dipole moments approach m_coils = [0; 0; 0] demonstrates the consistency between the quaternions around the stabilizable equilibrium and the corresponding angular rate around the stabilizable equilibrium.

Simulation Time = 100,000 seconds

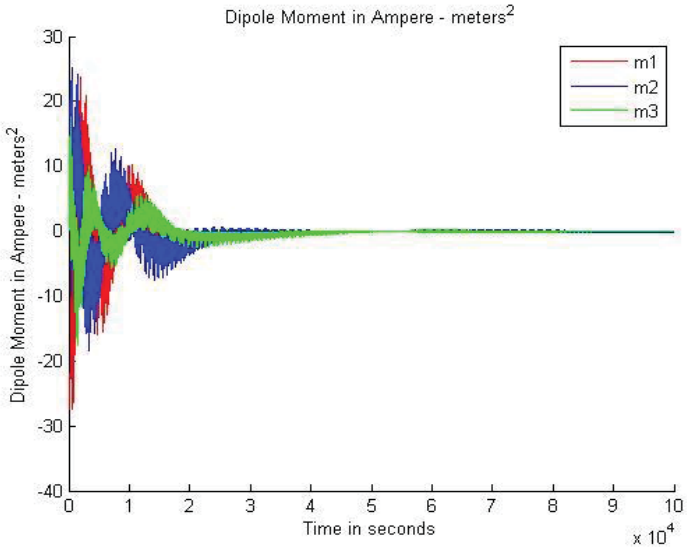

Figure12: Control Dipole Moment in A-m^2

5.2 Simulation Results for Inertia Matrix = [10, 10, 10] kg-m^2

5.2.1 Angular Velocity simulation results

Simulation Time = 100,000 seconds

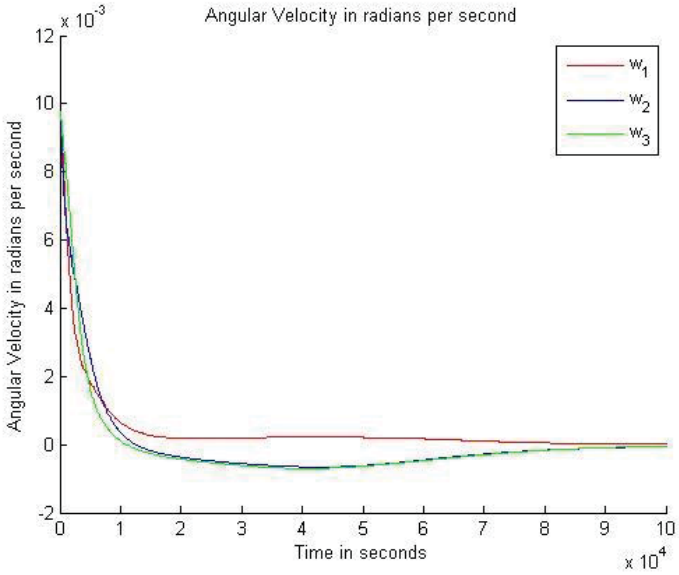

Figure13: Angular Velocity in radians/second

Simulation Time = 150,000 seconds

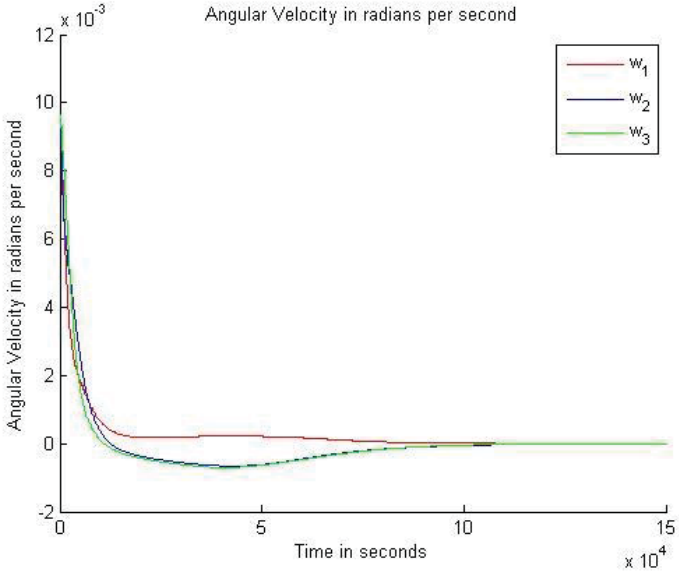

Figure14: Angular Velocity in radians/second

It can be observed from the above figures that, the initial angular velocity was ω = [0.01, 0.01, 0.01] radians/second and eventually the angular velocity is negligible or driven towards ω = [0, 0, 0]. The consistency of the results has also been demonstrated in the figure (2) above where the angular velocity maintains the trend at 150,000 sec.

5.2.2 Quaternions Simulation Results

Simulation Time = 100,000 seconds

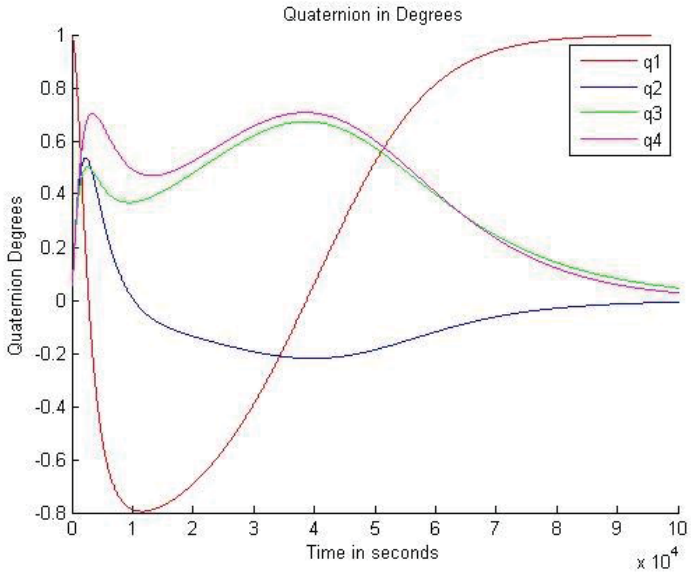

Figure15: Quaternions

Simulation Time = 150,000 seconds

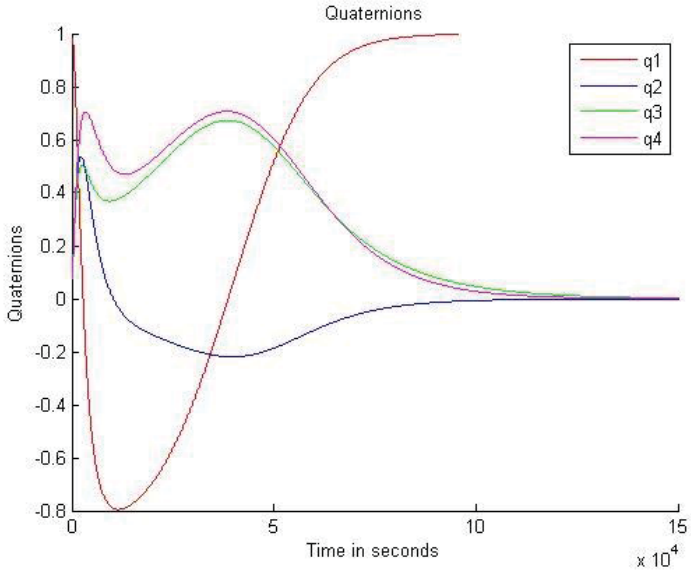

Figure16: Quaternions

It can be observed from the above figures that, the initial quaternion test cases were q = [1, 0, 0, 0]. Eventually the quaternions are driven towards q = [1, 0, 0, 0] which is the stabilizable equilibrium. This consistency has also been demonstrated in the figure (6) above where the quaternions maintain the trend at 150,000 sec.

5.2.3 Control Torque Simulation Results

Simulation Time = 100,000 seconds

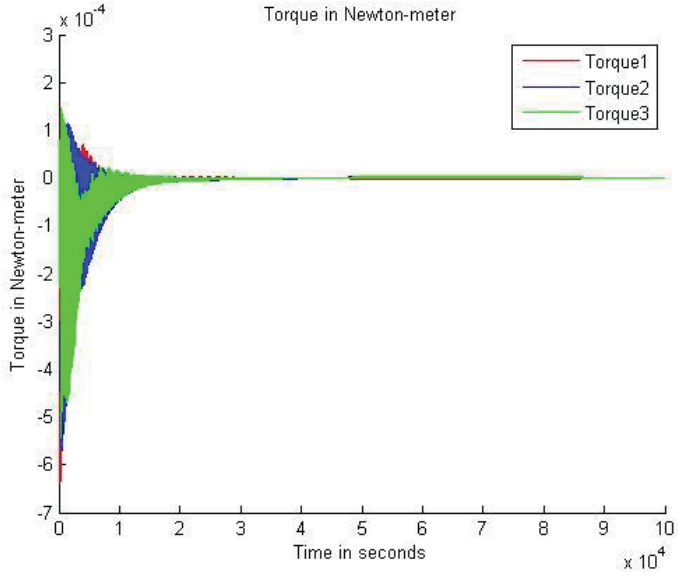

Figure17: Control Torque in N-m

Simulation Time = 150,000 seconds

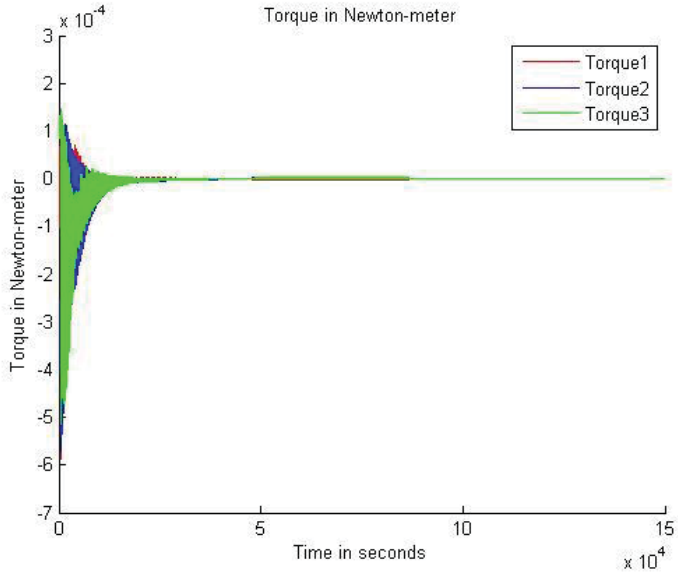

Figure18: Control Torque in N-m

5.2.4 Dipole Moment Simulation Results

Simulation Time = 100,000 seconds

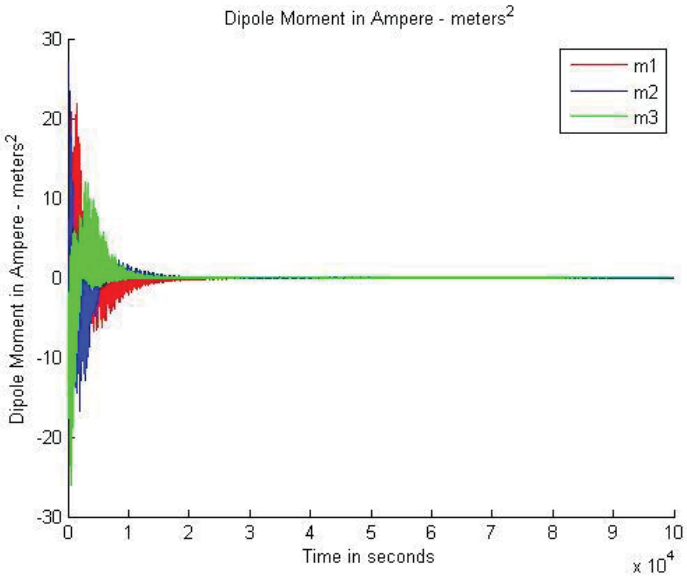

Figure19: Residual Dipole Moment in A-m^2

Simulation Time = 150,000 seconds

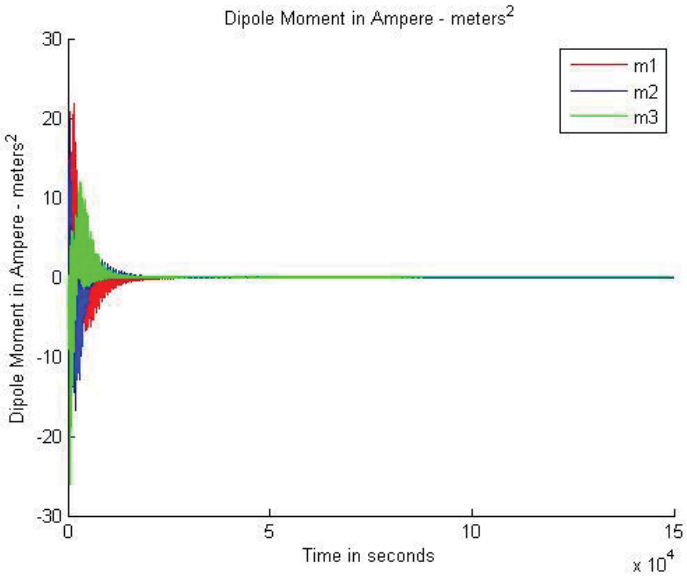

Figure20: Residual Dipole Moment in A-m^2

The simulation results for the dipole moment attests the statement [1] that due to the kinematics and dynamics of the system that all the trajectories of the system are such that $q \rightarrow 0$ and $\omega \rightarrow 0$. This can be seen from the control dipole moment vector being m_coils = [0; 0; 0]. This state has been reached even faster than the non iso-inertial spacecraft because of the availability of quaternion (Attitude) and angular velocity (Rate) feedback, at all times (together called the full state feedback).

Since, the stabilization problem we are dealing with, is based on state feedback stabilization, the performance of the state feedback controller is very good for both iso-inertial and non iso-inertial spacecraft.

For the case of partial state feedback, however, an almost global solution can be guaranteed only in the case of iso-inertial spacecraft because of the non-availability of rate feedback [10]. Also the necessity of complete quaternion information requires that the attitude information is available at all times from the sensors placed in the spacecraft.

6. Conclusion

The simulations results have demonstrated the feasibility of the control law with full state feedback in addressing the problem of attitude regulation of the spacecraft using only magnetic actuators. It has also been demonstrated by various iterations that the global solution can be established using the control law with full state feedback mentioned in section 3, in case of both non iso-inertial and iso-inertial spacecraft and the results appear to be fairly robust.

This report discussed the problem of inertial pointing of spacecraft using only magnetic actuators with a review of several existing approaches, the spacecraft model, the design of control algorithm for the purpose of inertial pointing of spacecraft using only magnetic actuators and finally presented results to document the results of the control law with full state feedback to justify the claim[3] that a global solution is available for the spacecraft to be controlled independently in three mutually perpendicular directions using only magnetic actuators and that attitude regulation can be achieved even in the absence of other active and/or passive control devices, at least for the spacecraft with small angular velocities.

The fact that an almost global solution is established for the case of full state feedback, a generic global solution for controllability of a spacecraft with large angular velocities for the case of full state feedback, partial feedback and output feedback provides a scope for significant theoretical research.

7. References

[1] Lovera, M & Astolfi, A (2004). Spacecraft Attitude Control using magnetic actuators

[2] Jackson, Brandon & Zou, Shangyan (2013). Satellite Kinematics.m - Attached in the Appendix

[3] Jackson, Brandon. SGP4 Propagator Setup, SGP4 Analyzer – Attached in the Appendix

[4] Jackson, Brandon. OCULUS-ASR ADAC Overview (2013)

[5] Analytical Mechanics of Space Systems – 2nd Edition

[6] World Magnetic Model

http://www.ngdc.noaa.gov/geomag/WMM/DoDWMM.shtml

[7] Two-Line Element Set

http://en.wikipedia.org/wiki/Two-line_element_set

[8] The Control Handbook (1996), Page 1311 –Levine, William S

[9] Earth Observatory– NASA

http://earthobservatory.nasa.gov/Features/10thAnniversary/

[10] Akella, Maruthi R (2001). Rigid Body Attitude Tracking without angular velocity feedback

[11] Jet Propulsion Laboratory – NASA

http://www.jpl.nasa.gov/news/news.php?release=2013-213

[12] Wisniewski, Rafael & Blanke, Mogens (1998). Fully Magnetic Attitude Control for spacecraft subject to gravity gradient

[13] Wang, Ping & Shtessel, Yuri B & Wang Yong-qian (1998). Satellite Attitude Control suing only Magnetorquers

[14] Lovera, Marco & Astolfi, Alessandro (2001). Global Attitude Regulation using Magnetic Control

[15] Bushenkov, Vladimir A, Ovchinnikov, Michael Yu & Smirnov, Georgi V (2002). Attitude Stabilization of a satellite by magnetic coils

[16] Silani, Enrico & Lovera, Marco (2005). Magnetic Spacecraft Attitude Control: A survey and some new results

[17] wrldmagm-Mathworks.
http://www.mathworks.com/help/aerotbx/ug/wrldmagm.html

Appendix

A. Control Algorithm for Spacecraft Attitude Control using Magnetic Actuators

```
% Karthik Mysore Srinivasa
% December 8th 2014
%function Torque_s = Controller(w_s,q_i_s)
clear all
clc

I = [27 17 25];% In Satellite Frame[kg-m^2]; Default values are [27 17
25]
I_Mat = diag(I);% Principal Inertia Matrix
eps = 0.001; % Epsilon
kp = 50;% Propotional Constant
kv = 50;% Velocity Constant
Tf = 100000;% Enter Simulation Time seconds
Ts = 0.1;% Step Time in seconds
K = zeros(3,Tf);% Dummy matrix to store Angular Velocity in rad/sec at
every instant
G = zeros(4,Tf);% Dummy Matrix to store Quaternions at every instant
U = zeros(3, Tf);% Dummy Matrix to store Control values
m_coils = zeros(3,Tf);% Dummy Matrix to store Residual Dipole Moments
in A-m^2 at every instant
Torque_s = zeros(3,Tf);% Dummy Matrix to store Torque in N-m at every
instant
Y = zeros(3,Tf);% dummy Matrix to store Euler Angles in Degrees at
every instant
q_s_c = [1;0;0;0];% Quaternion Rotation from Satellite to Controller
Frame
q_init_s = [1;0;0;0];...[-0.4873; 0.0345; 0.0542; 0.8709];% Initial
Quaternions
w_init_s = [0.01; 0.01; 0.01];% Initial Angular Velocity in rad/sec

T = zeros(1, Tf); % Dummy Vector to store Time in seconds as it
increments

%  %% Compute Magnetic Field vector using World Magnetic Model

for i = 1:Tf
    T(i) = i-1;

if T(i) == 0

% References: SGP4_Setup by Brandon Jackson
% Orbit (TLE)
longstr1 = '1 25544U 14067A   14342.51579142  .00005418  00000-0
10235-3 0  3046';
longstr2 = '2 25544  51.6494 243.7352 0003674 255.5105 239.9716
15.50141065841738';

% Initialize and Start the SGP4 propagator

SGP4_Setup(longstr1, longstr2) % Orbit Propagator Setup
ECEF_Init = sgp4(T(i))*1000; % Position Vector for each Time Instant in
ECEF Frame w.r.t ECI Frame
```

```
LLA_Init = ecef2lla(ECEF_Init'); % Transform to Geodetic Frame w.r.t
ECI Frame
Latitude = LLA_Init(1); % Compute Real-Time Latitude
Longitude = LLA_Init(2); % Compute Real-Time Longitude
Altitude = LLA_Init(3)/1000; % Compute Real-Time Altitude

%% Compute Magnetic Field vector using World Magnetic Model
    [b_0_t] = wrldmagm(Altitude, Latitude, Longitude,
decyear(2014,12,8))*1E-9;   % Convert nT to T;
    NED_Init = dcmecef2ned(Latitude, Longitude)*ECEF_Init; % Direction
Cosine Matrix for transformation from ECEF to NED Frame



    %% First time when Attitude and Rate are NOT available

    u = -((eps^2*kp*q_init_s(2:4))+(eps*kv*w_init_s)); % Control Law

    U(:,i) = u;

    A_q = quat2dcm(q_init_s'); % Attitude Matrix
     b_t = (A_q)*b_0_t; % Magnetic Field in Satellite Body Frame
          S_b_t = [0, b_t(3), -b_t(2);...
            -b_t(3), 0, b_t(1);...
            b_t(2), -b_t(1), 0]; % Skew-Symmetric Matrix
        Transpose = S_b_t';

    Abs = (norm(b_0_t)^2)^(-1);
    m = Abs*Transpose*u;
    m_coils(:, i) = m; % Residual Dipole Moment in A-m^2
    T_coils = cross(m_coils(:, i), b_t); % Control Torque in N-m
    %T_coils = S_b_t*m_coils(:, i);



    Torque_s(:, i) = T_coils; % Control Torque in N-m
    [w_s, q_i_s]= Satellite_Kinematics(I, w_init_s...
                                    , Torque_s(:, i), q_init_s...
                                    , q_s_c...
                                    , Ts); % Angular Velocity and
Quaternions at the next time instant

   G(:, 1) = q_init_s; % Quaternions at the first instant
   G(:, i+1) = q_i_s(:);
   K(:, i+1) = w_s; % Angular Velocity at the first instant
   K(:, 1) = w_init_s;

   [Y(1,i), Y(2,i), Y(3,i)] = quat2angle(G(:,i)', 'ZXZ'); % Euler
Angles in Radians
    Y(1,i) = Y(1,i)*(180/pi);Y(2,i) = Y(2,i)*(180/pi);Y(3,i) =
Y(3,i)*(180/pi); % Euler Angles in Degrees

else

% References: SGP4_Setup by Brandon Jackson
% Orbit (TLE)
```

```
longstr1 = '1 25544U 14067A   14342.51579142  .00005418  00000-0
10235-3 0  3046';
longstr2 = '2 25544  51.6494 243.7352 0003674 255.5105 239.9716
15.50141065841738';

% Initialize and Start the SGP4 propagator
% References: SGP4_Setup by Brandon Jackson
SGP4_Setup(longstr1, longstr2) % Orbit Propagator Setup
ECEF_Init = sgp4(T(i))*1000; % Position Vector for each Time Instant in
ECEF Frame w.r.t ECI Frame
LLA_Init = ecef2lla(ECEF_Init'); % Transform to Geodetic Frame w.r.t
ECI Frame
Latitude = LLA_Init(1); % Compute Real-Time Latitude
Longitude = LLA_Init(2); % Compute Real-Time Longitude
Altitude = LLA_Init(3)/1000; % Compute Real-Time Altitude

%% Compute Magnetic Field vector using World Magnetic Model
    [b_0_t] = wrldmagm(Altitude, Latitude, Longitude,
decyear(2014,12,8))*1E-9;   % Convert nT to T;
    NED_Init = dcmecef2ned(Latitude, Longitude)*ECEF_Init;

    %% If Attitude and Rate Feedback available, compute Control
Input'u'

    u = -((eps^2*kp*((G((2:4), (i)))))+(eps*kv*((K(:, (i)))))); %
Control Law


     A_q = quat2dcm(G(:,i)'); % Attitude Matrix
     b_t = (A_q)*b_0_t; % Magnetic Field in Satellite Body Frame
          S_b_t = [0, b_t(3), -b_t(2);...
            -b_t(3), 0, b_t(1);...
            b_t(2), -b_t(1), 0]; % Skew-Symmetric Matrix
        Transpose = S_b_t';

 U(:,i) = u;


 Abs = (norm(b_0_t)^2)^(-1);
 m = Abs*Transpose*u;
 m_coils(:, i) = m; % Residual Dipole Moment in A-m^2
 T_coils = cross(m_coils(:, i), b_t); % Control Torque in N-m
 %T_coils = S_b_t*m_coils(:, i);


 Torque_s(:, i) = T_coils; % Control Torque in N-m

 [w_s, q_i_s]= Satellite_Kinematics(I, w_init_s...
                                    , Torque_s(:, i), q_init_s...
                                    , q_s_c...
                                    , Ts); % Angular Velocity and
Quaternions at the next instant

 G(:, i+1) = q_i_s(:);
```

```
 K(:, i+1) = w_s;

 [Y(1,i), Y(2,i), Y(3,i)] = quat2angle(G(:,i)', 'ZXZ'); % Euler Angles
in Radians
 Y(1,i) = Y(1,i)*(180/pi);Y(2,i) = Y(2,i)*(180/pi);Y(3,i) =
Y(3,i)*(180/pi); % Euler Angles in Degrees

end

end

p = 0:(Tf);
figure(1);
hold on
plot(p, K(1,:),'r');
plot(p, K(2,:),'b');
plot(p, K(3,:),'g');
xlabel('Time in seconds');
ylabel('Angular Velocity in radians per second');
title('Angular Velocity in radians per second');
legend('w_1','w_2','w_3');
set(gca,'XLim',[0 Tf]);
hold off

figure(2);
hold on
plot(p, G(1,:),'r');
plot(p, G(2,:),'b');
plot(p, G(3,:),'g');
plot(p, G(4,:),'m');
xlabel('Time in seconds');
ylabel('Quaternions');
title('Quaternions');
legend('q1','q2','q3','q4');
set(gca,'XLim',[0 Tf]);
hold off

figure(3);
hold on
plot(T, Torque_s(1,:),'r');
plot(T, Torque_s(2,:),'b');
plot(T, Torque_s(3,:),'g');
xlabel('Time in seconds');
ylabel('Torque in Newton-meter');
title('Torque in Newton-meter');
legend('Torque1','Torque2','Torque3');
hold off

figure(4);
hold on
plot(T, m_coils(1,:),'r');
plot(T, m_coils(2,:),'b');
plot(T, m_coils(3,:),'g');
xlabel('Time in seconds');
ylabel('Dipole Moment in Ampere - meters^2');
```

```
title('Dipole Moment in Ampere - meters^2');
legend('m1','m2','m3');
hold off

figure(5);
hold on
plot(T, Y(1,:),'r');
plot(T, Y(2,:),'b');
plot(T, Y(3,:),'g');
xlabel('Time in seconds');
ylabel('Euler Angles in Degrees');
title('Euler Angles in Degrees');
legend('theta1','theta2','theta3');
hold off
```

B. Satellite Attitude Kinematics and Dynamics – SatelliteKinematics.m [2]

```
function [w_s, q_i_s]= Satellite_Kinematics(I_c, w_init_s, Torque_s,
q0_i_s, q_s_c, Ts)
%#codegen

persistent w_c_km1 q_i_c_km1

% Convert torque from satellite to controller frame
Torque_c_temp = quatmultiply(quatmultiply(quatinv(q_s_c), [0;
Torque_s]), q_s_c);
Torque_c = Torque_c_temp(2:4,1);

if  isempty(w_c_km1) || isempty(q_i_c_km1)
    % First time this function is run, pass through initial conditions
    w_s = w_init_s;
    q_i_s = q0_i_s;

    % Store for next time
    w_c_temp = quatmultiply(quatmultiply(quatinv(q_s_c), [0; w_s]),
q_s_c);
    w_c_km1 = w_c_temp(2:4);
    q_i_c_km1 = quatmultiply(q_i_s, q_s_c);
else
    % Second time this function is run, time to integrate

    x_km1 = [q_i_c_km1; w_c_km1];

    % Integrate
    k1 = Kinematics(x_km1, I_c, Torque_c);
    k2 = Kinematics(x_km1 + 0.5*k1*Ts, I_c, Torque_c);
    k3 = Kinematics(x_km1 + 0.5*k2*Ts, I_c, Torque_c);
    k4 = Kinematics(x_km1 + k3*Ts, I_c, Torque_c);

    x_k = x_km1 + (k1 + 2*k2 + 2*k3 + k4)*Ts/6;

    % Store for next time
    q_i_c_km1 = x_k(1:4);
    w_c_km1 = x_k(5:7);

    % Rotate and return
    q_i_s = quatmultiply(q_i_c_km1, quatinv(q_s_c));
    w_s_temp = quatmultiply(quatmultiply(q_s_c, [0; w_c_km1]),
quatinv(q_s_c));
    w_s = w_s_temp(2:4);
end

end

function output = skew_matrix(x)
% Returns the skew symmetric matrix of the input vector

output = [0 -x(3) x(2);...
    x(3) 0 -x(1);...
```

```
    -x(2) x(1) 0];
end

function results = Kinematics(x, I, Torque_c)
q = x(1:4);
w = x(5:7);
I_Mat = diag(I);

q_dot = 0.5.*[0 -w'; w -skew_matrix(w)]*q;
w_dot = I_Mat^(-1)*(-skew_matrix(w)*(I_Mat*w) + Torque_c);
% w_dot = [0; 0; 0];
% w_dot(1,1) = ( Torque_c(1) - w(2)*w(3)*(I(3,3) - I(2,2)) )/I(1,1);
% w_dot(2,1) = ( Torque_c(2) - w(1)*w(3)*(I(1,1) - I(3,3)) )/I(2,2);
% w_dot(3,1) = ( Torque_c(3) - w(1)*w(2)*(I(2,2) - I(1,1)) )/I(3,3);

results = [q_dot; w_dot];
end

function qres = quatmultiply(q, r)
q = q';
r = r';
% Calculate vector portion of quaternion product
% vec = s1*v2 + s2*v1 + cross(v1,v2)
vec = [q(:,1).*r(:,2) q(:,1).*r(:,3) q(:,1).*r(:,4)] + ...
        [r(:,1).*q(:,2) r(:,1).*q(:,3) r(:,1).*q(:,4)]+...
        [ q(:,3).*r(:,4)-q(:,4).*r(:,3) ...
          q(:,4).*r(:,2)-q(:,2).*r(:,4) ...
          q(:,2).*r(:,3)-q(:,3).*r(:,2)];

% Calculate scalar portion of quaternion product
% scalar = s1*s2 - dot(v1,v2)
scalar = q(:,1).*r(:,1) - q(:,2).*r(:,2) - ...
            q(:,3).*r(:,3) - q(:,4).*r(:,4);

qres = [scalar  vec]';
end

function qinv = quatinv(qin)

q_conj = [qin(1); -qin(2:4)];
qinv = q_conj./sqrt(sum(qin.^2));

end
```

C. SGP4 Setup[3]

```
function SGP4_Setup(longstr1, longstr2)
% SGP4_Setup
% Brandon Jackson
% bajackso@mtu.edu
% 9th July 2013
%
%
% Inputs:
% longstr1: TLE character String Line 1
% longstr2: TLE character Sring Line 2
%
% Outputs:
% satrec: structure containing all of the SGP4 satellite information
%
% Coupling:
% getgravconst
% days2mdhms
% jday
% sgp4init
%
% References:
% Norad Spacetrack Report #3
% Vallado, Crawford, Hujsak, Kelso 2006

%% Define Global Variables
global satrec gravc
% gravc = struct('mu', 0.0,...
%     'radiusearthkm', 0.0,...
%     'xke', 0.0,...
%     'tumin', 0.0,...
%     'j2', 0.0,...
%     'j3', 0.0,...
%     'j4', 0.0,...
%     'j3oj2', 0.0);

%% Include extrinsic functions
% coder.extrinsic('custom_str2double');

%% WGS-72 Earth Constants
% sgp4fix identify constants and allow alternate values
% Options 721 72 84
getgravc( 721 );

%% Define Constants
deg2rad  =   pi / 180.0;         %  0.01745329251994330;  % [deg/rad]
    xpdotp   =  1440.0 / (2.0*pi);   % 229.1831180523293;  %
[rev/day]/[rad/min]

    revnum = 0;
    elnum  = 0;
    year   = 0;
    satrec.error = 0;
```

```
%% Parse TLE
% Set the implied decimal points since doing a formated read fixes for
bad
% input data values (missing, ...)
    for (j = 11:16)
        if (longstr1(j) == ' ')
            longstr1(j) = '_';
        end
    end

    if (longstr1(45) ~= ' ')
        longstr1(44) = longstr1(45);
    end
    longstr1(45) = '.';

    if (longstr1(8) == ' ')
        longstr1(8) = 'U';
    end

    if (longstr1(10) == ' ')
        longstr1(10) = '.';
    end

    for (j = 46:50)
        if (longstr1(j) == ' ')
            longstr1(j) = '0';
        end
    end
    if (longstr1(52) == ' ')
        longstr1(52) = '0';
    end
    if (longstr1(54) ~= ' ')
        longstr1(53) = longstr1(54);
    end
    longstr1(54) = '.';

    longstr2(26) = '.';

    for (j = 27:33)
        if (longstr2(j) == ' ')
            longstr2(j) = '0';
        end
    end

    if (longstr1(63) == ' ')
        longstr1(63) = '0';
    end

    if ((length(longstr1) < 68) || (longstr1(68) == ' '))
        longstr1(68) = '0';
    end

    % parse first line
    carnumb = custom_str2double(longstr1(1));
    satrec.satnum = custom_str2double(longstr1(3:7));
```

```
    classification = longstr1(8);
    intldesg = longstr1(10:17);
    satrec.epochyr = custom_str2double(longstr1(19:20));
    satrec.epochdays = custom_str2double(longstr1(21:32));
    satrec.ndot = custom_str2double(longstr1(34:43));
    satrec.nddot = custom_str2double(longstr1(44:50));
    nexp = custom_str2double(longstr1(51:52));
    satrec.bstar = custom_str2double(longstr1(53:59));
    ibexp = custom_str2double(longstr1(60:61));
    numb = custom_str2double(longstr1(63));
    elnum = custom_str2double(longstr1(65:68));

    % parse second line
    cardnumb = custom_str2double(longstr2(1));
    satrec.satnum = custom_str2double(longstr2(3:7));
    satrec.inclo = custom_str2double(longstr2(8:16));
    satrec.nodeo = custom_str2double(longstr2(17:25));
    satrec.ecco = custom_str2double(longstr2(26:33));
    satrec.argpo = custom_str2double(longstr2(34:42));
    satrec.mo = custom_str2double(longstr2(43:51));
    satrec.no = custom_str2double(longstr2(52:63));
    revnum = custom_str2double(longstr2(64:68));

% find no, ndot, nddot
    satrec.no   = satrec.no / xpdotp; %//* rad/min
    satrec.nddot= satrec.nddot * 10.0^nexp;
    satrec.bstar= satrec.bstar * 10.0^ibexp;

% convert to sgp4 units
    satrec.a    = (satrec.no*gravc.tumin)^(-2/3);                % [er]
    satrec.ndot = satrec.ndot  / (xpdotp*1440.0);          %
[rad/min^2]
    satrec.nddot= satrec.nddot / (xpdotp*1440.0*1440);     %
[rad/min^3]

% find standard orbital elements
    satrec.inclo = satrec.inclo  * deg2rad;
    satrec.nodeo = satrec.nodeo * deg2rad;
    satrec.argpo = satrec.argpo  * deg2rad;
    satrec.mo    = satrec.mo     *deg2rad;

    satrec.alta = satrec.a*(1.0 + satrec.ecco) - 1.0;
    satrec.altp = satrec.a*(1.0 - satrec.ecco) - 1.0;

%% Find SGP4 Epoch Time of element set
%  Remember that sgp4 uses units of days from 0 jan 1950 (sgp4epoch)
and
%  minutes from the epoch (time)

% Temp fix for years 1957-2056
% Correct fix will occur when year is 7-digits in 21e
     if (satrec.epochyr < 57)
         year= satrec.epochyr + 2000;
       else
         year= satrec.epochyr + 1900;
```

```
     end;

     [mon,day,hr,minute,sec] = days2mdh ( year,satrec.epochdays );
     satrec.jdsatepoch = jday( year,mon,day,hr,minute,sec );

 %% Initialize the orbit at SGP4 Epoch
 sgp4epoch = satrec.jdsatepoch - 2433281.5; % days since 0 Jan 1950
 sgp4init(sgp4epoch);
end

    function output_double = custom_str2double(input_str)
        i = 0.0;
        output_double = 0;
        npast = 0;
        num_sign = 1;
        input_str = input_str(input_str ~= ' ');
        for i = 1:length(input_str)
            if input_str(i) == '-'
                num_sign = -1;
            elseif input_str(i) == '.' && npast == 0
                npast = 1;
            elseif npast >= 1;
                output_double = output_double + (input_str(i) -
'0')*10^-npast;
                npast = npast+1;
            else
                % Value is greater than 1
                output_double = output_double*10 + (input_str(i) -
'0');
            end
        end
        output_double = num_sign*output_double;
    end
```