12-2015

Implementing a Linear Quadratic Spacecraft Attitude Control System

Daniel Kolosa

IMPLEMENTING A LINEAR QUADRATIC SPACECRAFT ATTITUDE CONTROL SYSTEM

by

Daniel Kolosa

A thesis submitted to the Graduate College
in partial fulfillment of the requirements
for the Degree of Master of Science
Mechanical and Aerospace Engineering
Western Michigan University
December 2015

Thesis Committee:

Jennifer Hudson, Chair Ph.D
Kristina Lemmer, Ph.D
Kapsong Ro, Ph.D

IMPLEMENTING A LINEAR QUADRATIC SPACECRAFT ATTITUDE CONTROL SYSTEM

Daniel Kolosa, M.S.E.

Western Michigan University

This thesis implements a linear quadratic attitude control system for a low-thrust spacecraft. The goal is to maintain spacecraft alignment with a time-varying thrust vector needed for trajectory change maneuvers. A linear quadratic attitude control approach is used to maintain spacecraft pointing throughout flight. This attitude control strategy uses the thrust-acceleration input obtained from a linear quadratic optimal trajectory control model that simulates the trajectory of a spacecraft in orbit maneuvers. This attitude model simulates a CubeSat, a small satellite that is equipped with a low-thrust propulsion and attitude control system. An orbit raising and a plane change scenario is modelled for this spacecraft. The results of the attitude model show that for the orbit raising maneuver, the attitude controller exhibits periodic behavior with the same frequency as the calculated spacecraft thrust acceleration.

ACKNOWLEDGMENTS

Although my name shows up on the cover of this thesis, it is a number of people whom I owe my gratitude to that have contributed to its finishing. Because of them, my thesis production has been a remarkable experience that I will always be grateful.

First and foremost I wish to thank Dr. Jennifer Hudson whom I have been amazingly fortunate to have as my advisor since the day I began my master studying. She is the one professor who truly made a difference in my life. Dr. Hudson has been providing me with numerous opportunities which greatly enhance my knowledge level in orbital mechanics and engineering. I doubt that I will ever be able to fully express my appreciation, but I owe her my eternal gratitude.

A very special appreciation goes out to Dr. Kristina Lemmer, whom I have been blessed to have as advisory committee member. Her insightful comments and constructive criticism of my thesis made me focus on my ideas. I am very thankful for her contribution to my development as a researcher. Her strict attitude towards research will be always my criteria in the future.

I must also acknowledge Dr. Kapsong Ro, for his suggestions, of the subject of the thesis. His vast knowledge and skills in control theory aided me to form my ambition through my undergraduate and graduate time. Thanks to his encouragement, I was able to enrich my ideas.

I recognize that this research would not have been possible without the financial assistance of teaching assistantships and scholarships, and for that I express my deep appreciation.

Finally, I would like to thank my parents Andrew and Jadwiga, my brother Chris, my sister Martha, and my beloved Yihan who always cares and support me through the good times and bad.

Daniel Kolosa

TABLE OF CONTENTS

Table of Contents-Continued

LIST OF TABLES

LIST OF FIGURES

List of Figures—Continued

NOMENCLATURE

A = state-vector cost matrix

a = width

a = semi-major axis

B = input cost matrix

b = height

C = direction cosine

E = eccentric anomaly

e = eccentricity

F = thrust acceleration

F_R = radial thrust acceleration

$\hat{F}_R$ = radial thrust acceleration direction

F_S = normal thrust

$\hat{F}_S$ = normal thrust acceleration angle

F_W = circumferential thrust acceleration

$\hat{F}_W$ = circumferential thrust acceleration direction

G = gravitational constant

$\boldsymbol{H}$ = angular momentum

I = identity matrix

I_{sp} = specific impulse

i = inclination

J = inertia matrix

$\boldsymbol{K}$ = unit vector

K_f = final value weight matrix

L = cost function

l = length

M = mean anomaly

m = mass

m_1 = mass of central body

$\dot{m}$ = mass flow rate

$\boldsymbol{N}$ = unit vector points to direction of the ascending node

P = riccati equation

Q = state weight matrix

$\boldsymbol{q}$ = quaternion

R = input weight matrix

$\boldsymbol{r}$ = position vector

T = euler to direction cosine matrix

t_P = time of perigee passage

u = input control law

$\boldsymbol{v}$ = velocity vector

x = state-vector

α^{RSW} = cosine coefficient of Fourier series

β^{RSW} = sine coefficient of Fourier series

Δv = velocity change

ε = specific orbital energy

ϕ = angle between thrust for orbit transfers

Ω =right ascension longitude of the ascending mode

ω = argument of perigee

μ = standard gravitational parameter of the Earth

θ = euler angle

$\dot{\theta}$ = angular velocity

Chapter 1

INTRODUCTION

The dynamics describing the non-linear trajectory and attitude of a spacecraft implementing low-thrust maneuvers can be computationally intensive. This thesis proposes a trajectory and attitude control model that linearizes spacecraft dynamics using Fourier thrust-acceleration components.

A linear quadratic optimal control model was created to design trajectories for low-thrust spacecraft for different orbit maneuvers. This model approximates the trajectory of a spacecraft given an initial and a target state. Then a linear quadratic optimal control attitude controller was designed to be used on low-thrust spacecraft for modeling an approximation of its orientation. The objective of this attitude controller was to align the body frame of the spacecraft with the input thrust-acceleration angles of the trajectory control law. Both of the linear quadratic models were tested in an orbit raising and a plane change maneuver. This thesis is focused on implementing these optimal control models for missions that could be potentially feasible on small satellites.

The motivation for creating a linear quadratic trajectory controller was to express non-linear spacecraft dynamics as linear secular averaged equations. By creating this linear controller it is possible to simulate orbits that could be computationally inexpensive relative to a non-linear trajectory model and yield accurate simulations. After creating a liner quadratic trajectory controller, a linear quadratic attitude controller was created to model

the attitude of a spacecraft. The motivation to creating an attitude model is to determine whether it is possible for a low-thrust spacecraft to execute the necessary orientation.

This thesis will begin by first introducing low-thrust spacecraft and specifically CubeSats. Some key concepts of orbital mechanics are introduced, including the coordinate systems used and how to transform from one coordinate system to another, as well as a few simple orbit transfer maneuvers. Also the topic of low-thrust maneuvers such as spiral, plane change and eccentricity change maneuvers will be discussed.

In section 2, A model simulating the trajectory of a spacecraft in low-thrust spiral orbit using linear quadratic optimal control is discussed in section six. The non-linear Newtonian dynamics are linearized using linear quadratic optimal control. The linearized model was tested using a plane change and orbit raising maneuver.

In section 3, the fundamentals of attitude dynamics will be discussed as well as Euler and quaternion coordinate systems. Also various attitude control mechanisms will be reviewed.

In sections 4, 9, and 10, the theory of the linear quadratic attitude control model is discussed. This includes the parameters that were chosen for the linear quadratic system and any necessary coordinate transformations. The attitude control model was implemented on an orbit raising and a plane change orbit maneuver and the results are discussed.

Low-Thrust Spacecraft and CubeSats

This thesis will focus on the targeting maneuvers of a type of satellite called a CubeSat.

CubeSats are small 10cm by 10cm by 10cm satellites, as shown in Figure 1.1.

Figure 1.1: 1U CubeSat[1]

CubeSats have become popular with universities and start up companies because they are relatively inexpensive to design, build, and launch. Most They have been designed for short scientific missions like weather or atmospheric monitoring and recreational projects like photography and recording videos. Because of the size of these satellites, the propulsion and attitude systems must also be quite small and therefore produce little thrust and moment compared to large satellites. It is uncommon for CubeSats to have an attitude control or propulsion system because it is difficult to fit these components in a small form factor or they tend to fly simple missions that do not require sophisticated control. There have been investigations into implementing a propulsion system to allow far more dynamic missions and even the possibility for interplanetary travel as proposed by Ames, NASA, University of Michigan using their CubeSat Ambipolar Thruster[2], and other organizations.

These small satellites can be equipped with small propulsion systems. These propulsion systems like to one shown in figure 1.2, produce thrust in the range of micro-Newtons, and

on the power level of Watts. Generally, low-thrust propulsion use cold gas, nitrogen, or butane for fuel. Many new propulsion systems for small satellites are in development such as ion thrusters, electrospray thrusters, and micro-resistojet thrusters[3]. These propulsion systems have not yet been flown on any CubeSat mission yet.

Figure 1.2: Busek BIT-1 Ion Thruster[4]

These low-thrust propulsion systems and all other propulsion systems use the metric of specific impulse to measure their performance. The specific impulse, (I_{sp}) in seconds, is the unit of measurement used to determine the efficiency of a thruster and can be calculated based on equation (1.1).

$$I_{sp} = \frac{F}{\dot{m}} \tag{1.1}$$

Where F is the thrust of the thruster and $\dot{m}$ is the mass flow rate of the rocket's propellant. The higher a propulsion system's I_{sp} the more efficient it is.

Trajectory Dynamics

Trajectory control is control of a spacecraft's orbital path. Many orbital mechanics problems can be modelled as restricted two-body problems, where an object of negligibly small mass orbits a large central body. This kind of problem assumes that there are no other bodies affecting this system and other phenomena like air drag are neglected. The orbital dynamics can be described by Newtonian equations of motion. The orbital dynamics of a two-body system can be represented by the differential equation:

$$\ddot{\mathbf{r}} = \frac{\mu}{r^3}\mathbf{r} \tag{1.2}$$

$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}$, is the position in Cartesian coordinates, and μ is the gravitation constant of the central body.

Coordinate Systems

Orbital mechanics can be expressed using different coordinate systems. One method to express an orbit state is the keplarian orbital elements. Orbital elements describe the size, shape, and orientation of an orbit in terms of six elements. The orbital elements are shown in figure[5].

a = semi-major axis

e = eccentricity

t_p = time of perigee passage

Ω = right ascension longitude of the ascending node

i = inclination of the orbit plane

ω = argument of the perigee

The semi-major axis is the average sum of the periapsis and apoapsis distances. The eccentricity describes the shape of the orbit, and ranges from zero to one, with one being a circular orbit and approaching zero is a more elliptical orbit. The time of perigee passage is the time when an orbit body is at the closest passing point in the orbit. The ascending node, is the intersection point between the reference plane and the inclination. The inclination is the vertical lift of the orbit. The argument of the perigee is the orientation of the orbit in the orbital plane. These orbital elements are visualized in the figure below:

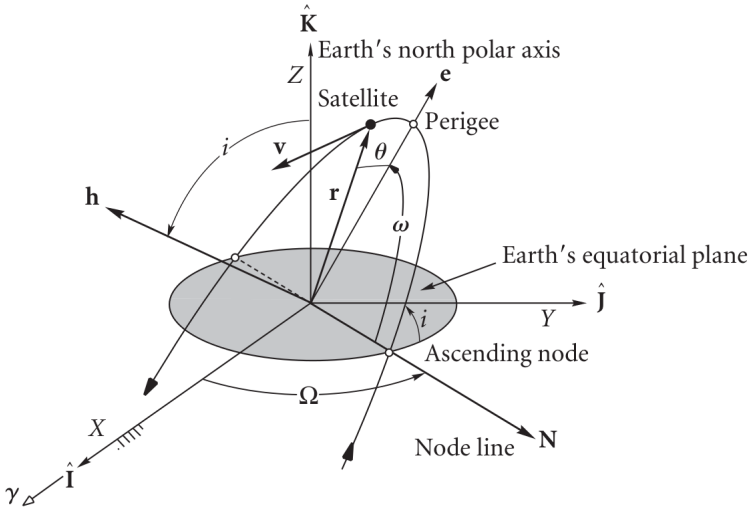

Figure 1.3: Orbital Elements[6]

An orbit can be transformed from a Cartesian coordinate system to orbital elements using a set of transformations shown in equation (1.3) below. These transformations may be nec-

essary when plotting trajectories.

$$
\begin{aligned}
r &= \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}^T \\
v &= \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T \\
\mathbf{h} &= \mathbf{r} \times \mathbf{v} \\
h &= \sqrt{\mathbf{h} \cdot \mathbf{h}} \\
\hat{\mathbf{K}} &= [0 \quad 0 \quad 1] \\
\mathbf{N} &= \hat{\mathbf{K}} \times \mathbf{h} \\
N &= \sqrt{\mathbf{N} \cdot \mathbf{N}} \\
i &= \cos^{-1}(\tfrac{h_z}{h}) \\
\Omega &= cos^{-1}(N_x/N) \\
e &= \tfrac{1}{\mu}[\mathbf{v} \times \mathbf{h} - \mu \tfrac{\mathbf{r}}{r}] \\
\omega &= \cos^{-1}(\mathbf{N} \cdot \mathbf{e}/Ne) \\
\theta &= \cos^{-1}(\tfrac{\mathbf{e} \cdot \mathbf{r}}{er}) \\
E &= \cos^{-1}(\tfrac{e+\cos\theta}{1+e\cos\theta}) \\
M &= E - e\sin E \\
\varepsilon &= \tfrac{\mu}{-2a}
\end{aligned}
\tag{1.3}
$$

The position and the velocity vectors can be obtained by solving the differential equation for the orbital dynamics of a two-body problem, equation 1.2. To obtain the orbital elements from the position and velocity vector the angular momentum h, is calculated. The

cross product for the pointing vecotr K and the angular momentum is used to determine the node line N. Using the node line, angular momentum, and position and velocity vectors, the eccentricity, inclination, perigee, ascending node, eccentric, and mean anamoly can be determined.

Orbit Transfers

Orbit transfers are ways in which a spacecraft can change the shape, size, or orientation of its orbit. Completing an orbit transfer requires energy, whether that be electrical systems or using chemical combustion. Δv is the velocity change required for an orbit maneuver. For a spacecraft with a chemical propulsion system, orbit change maneuvers can be approximated with instantaneous velocity changes, the Δv is given in (1.4) below[5].

$$\begin{aligned} \Delta v &= v_2 - v_1 \\ \Delta v &= \sqrt{v_1^2 + v_2^2 - 2v_1 v_2 \cos\phi} \end{aligned} \tag{1.4}$$

Where v_1 and v_2 are the velocities of the spacecraft before and after the thrust impulse, respectively and ϕ is the angle between v_1 and v_2. The higher the Δv for an orbit maneuver, the more energy it costs to implement that orbital maneuver. There are many different types of orbit transfers and depending on the target destination, some orbit transfers are more efficient than others.

A Hohmann transfer is one of the most energy efficient orbit transfers. In an orbit-

raising Hohmann transfer, a spacecraft starts in a circular orbit, then thrusts to create an elliptical orbit where the periapsis is the radius of the initial orbit, and the apoapsis of the elliptical orbit is the radius of the destination orbit. The spacecraft executes a second thrust impulse at apoapsis to re-circularize its orbit. A diagram of a Hohmann is pictured below.

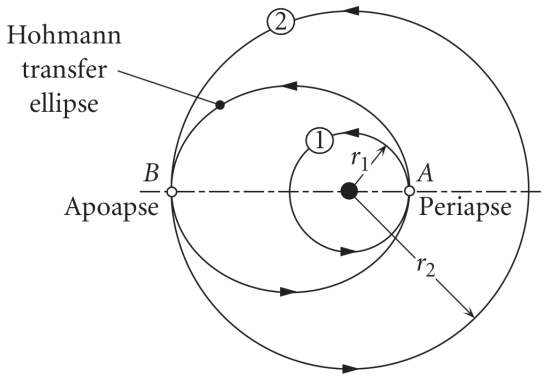

Figure 1.4: Hohmann Transfer[6]

Fundamentally, determining the Δ v for a Hohmann transfer is similar to an impulse maneuver[5].

$$\begin{aligned} \Delta v &= \Delta v_1 + \Delta v_2 \\ \Delta v_1 &= \sqrt{\frac{2\mu}{r_1} - \frac{\mu}{a}} - \sqrt{\frac{\mu}{r_1}} \\ \Delta v_2 &= \sqrt{\frac{\mu}{r_2}} - \sqrt{\frac{2\mu}{r_2} - \frac{\mu}{a}} \end{aligned} \tag{1.5}$$

Low-thrust Maneuvers

High-thrust maneuvers use chemical propulsion to make ΔV maneuvers. Electric propulsion can not produce as high a thrust as chemical propulsion giving rise to the development of low-thrust maneuvers. Low-thrust maneuvers assume that the orbit transfer time is short relative to the orbital period, and the thrust is modelled as continuous over long periods of time. To model the trajectory of low-thrust maneuvers, Gauss's Variational equations(1.6)can be used [7].

$$\begin{aligned}
\frac{da}{dt} &= 2\sqrt{\frac{a}{\mu}}[F_R \frac{ae}{\sqrt{1-e^2}} \sin\nu + F_S \frac{a^2\sqrt{1-e^2}}{a(1-e\cos E}] \\
\frac{de}{dt} &= \sqrt{\frac{a}{\mu}}\sqrt{1-e^2}[F_R \sin\nu + F_S(\cos\nu + \cos E)] \\
\frac{di}{dt} &= \sqrt{\frac{a}{\mu}} \frac{(1-e\cos E)}{\sqrt{1-e^2}} F_W \cos(\nu+\omega) \\
\frac{d\Omega}{dt} &= \sqrt{\frac{a}{\mu}} \frac{(1-e\cos E)}{\sqrt{1-e^2}} F_W \sin(\nu+\omega) \\
\frac{d\omega}{dt} &= \sqrt{\frac{a}{\mu}} \frac{(1-e^2)}{e}[-F_r \cos\nu + F_s(1+\frac{1-e\cos E}{1-e^2})\sin\nu] - \cos i \frac{d\Omega}{dt} \\
\frac{d\varepsilon}{dt} &= -2\sqrt{\frac{a}{\mu}}(1-e\cos E)F_R + (1-\sqrt{1-e^2})(\dot{\omega}+\dot{\Omega}) + 2\sqrt{1-e^2}\sin^2(\frac{i}{2})\dot{\Omega}
\end{aligned} \tag{1.6}$$

Gauss's Variational equations provide a convenient way to express non-linear dynamics of a spacecraft. They are linearized about orbital elements which allow for inexpensive computation time.

When optimizing low-thrust maneuvers researchers typically optimize for minimum propellant use or minimum time of flight. Because low-thrust maneuvers have a long transfer time, they are sensitive to small changes in the magnitude and direction of thrust.

Spiral Maneuvers

This thesis focuses on low-thrust spiral orbit maneuvers. These orbit maneuvers require

a long duration of thrust rather than impulsive. One type of spiral maneuver is an increase or decrease of the semi-major axis while maintaining a constant eccentricity and inclination.

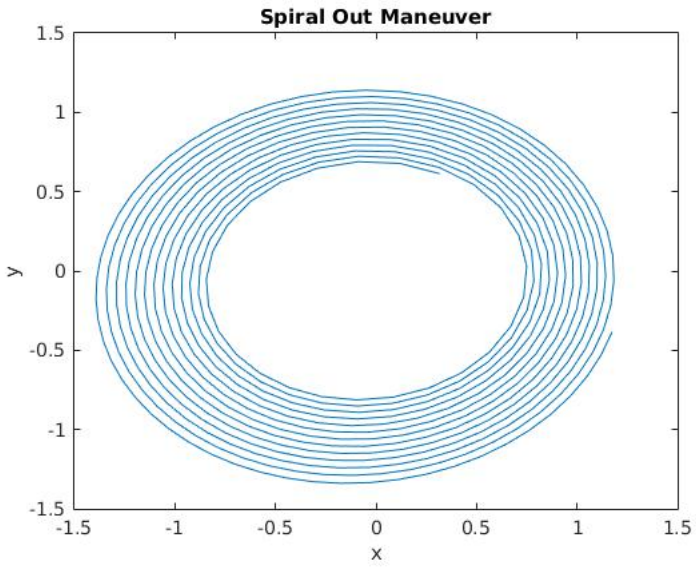

Figure 1.5: Spiral Out Manuever Normalized About The Radius Of The Earth

From figure 1.5 above, it can be seen that the spacecraft undergoes a large number of orbits compared to a Hohmann transfer. For the spiral maneuver, the spacecraft starts at a circular low-earth orbit (LEO). As the spacecraft is orbiting the Earth, it begins to spiral out by thrusting in small increments, until it reaches its target orbit. A trade off to low-thrust spiral orbit maneuvers is that they can take a long time to implement.

Plane Change Maneuvers

A plane change maneuver is an orbit maneuver where the inclination of the orbit is changed without changing the eccentricity or semi major axis as shown in figure 1.6.

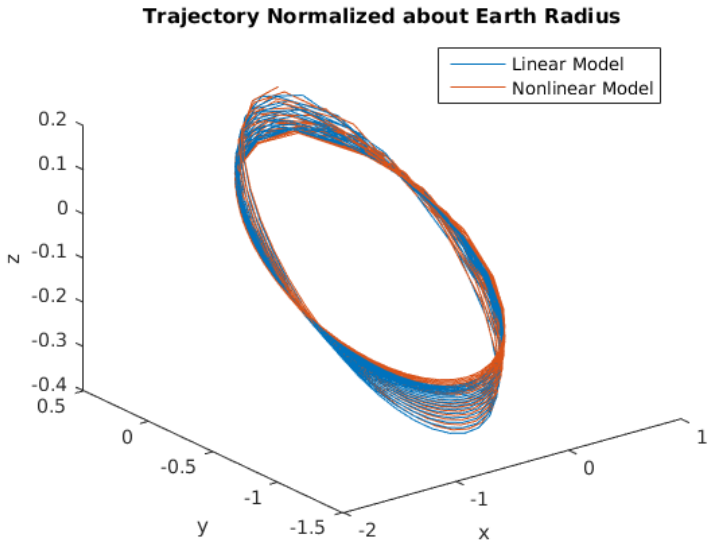

Figure 1.6: A Plane Change Maneuver

A plane change maneuver is made by pointing the thrust vector orthogonality to the angular momentum vector. For a minimal transfer time, the change in inclination and the delta v can be expressed as equation (1.7).

$$\begin{aligned} \Delta i &= \frac{2\Delta i}{\pi v_1} \\ \Delta v &= \frac{2\sin(\frac{\Delta i}{2})\sqrt{1-e^2}\cos(\omega+E)Ma}{1+e\cos E} \end{aligned} \tag{1.7}$$

The most efficient way to make a plane change maneuver is to thrust at the apogee of the current orbit along the intersection of the initial and target plane.

Eccentricity Change Maneuvers

The eccentricity change maneuver involves changing the shape of an orbit, either making it more elliptical or circular while maintaining the semi major axis a, constant, as shown

in the figure below.

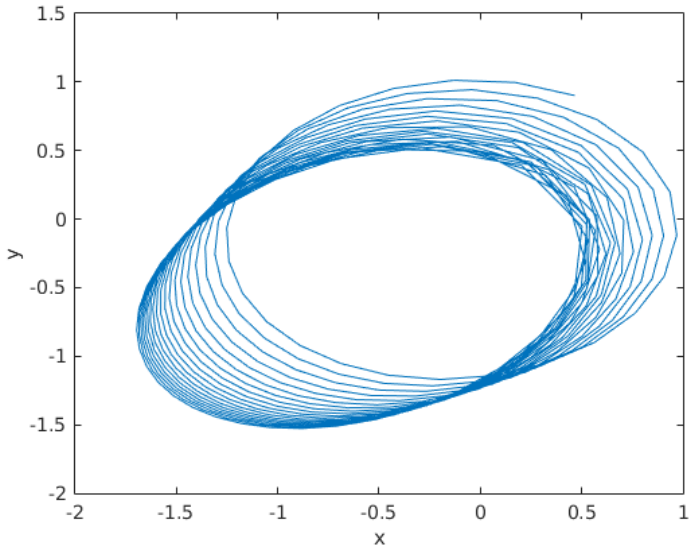

Figure 1.7: Eccentricity Change Maneuver

The change in eccentricity can be done by aligning the thrust vector in the direction of the angular momentum, orthogonal to the velocity and position vectors. The eccentricity change for a minimum transfer time and small thrust can be expressed as (1.8).

$$\frac{de}{dt} = 2\sqrt{\frac{a}{\mu}} f \cos\theta \tag{1.8}$$

Where θ is the thrust vector. There are many other types of orbit maneuvers that are outside the scope of this paper. The orbit maneuver that will be discussed later in this paper is based on low-thrust propulsion.

Chapter 2

LINEAR TRAJECTORY MODEL

The linear quadratic attitude control is based on a previously described trajectory control method[8]. The trajectory model is a linear approximation of two-body orbital dynamics with low-thrust acceleration. They are found by averaging for orbital motion with thrust expressed as Gauss's equations (1.6) for orbital motion of a perturbing thrust acceleration[8].

$$\bar{\dot{a}} = 2\sqrt{\frac{a^3}{\mu}}\left[\frac{1}{2}e\,\beta_1^R + \sqrt{1-e^2}\,\alpha_0^S\right] \tag{2.1}$$

$$\bar{\dot{e}} = \sqrt{\frac{a}{\mu}}\sqrt{1-e^2}\left[\frac{1}{2}\sqrt{1-e^2}\,\beta_1^R + \alpha_1^S - \frac{3}{2}e\,\alpha_0^S - \frac{1}{4}e\,\alpha_2^S\right] \tag{2.2}$$

$$\begin{aligned}\bar{\dot{i}} = {} & \sqrt{\frac{a}{\mu}}\frac{1}{\sqrt{1-e^2}}\left[\frac{1}{2}\left(1+e^2\right)\cos\omega\,\alpha_1^W - \frac{3}{2}e\cos\omega\,\alpha_0^W - \frac{1}{2}\sqrt{1-e^2}\sin\omega\,\beta_1^W\right. \\ & \left. -\frac{1}{4}e\cos\omega\,\alpha_2^W + \frac{1}{4}e\sqrt{1-e^2}\sin\omega\,\beta_2^W\right]\end{aligned} \tag{2.3}$$

$$\begin{aligned}\overline{\dot{\Omega}} = {} & \sqrt{\frac{a}{\mu}}\frac{\csc i}{\sqrt{1-e^2}}\left[\frac{1}{2}\sqrt{1-e^2}\cos\omega\,\beta_1^W + \frac{1}{2}\left(1+e^2\right)\sin\omega\,\alpha_1^W - \frac{3}{2}e\sin\omega\alpha_0^W\right. \\ & \left. -\frac{1}{4}e\sqrt{1-e^2}\cos\omega\,\beta_2^W - \frac{1}{4}e\sin\omega\,\alpha_2^W\right]\end{aligned} \tag{2.4}$$

$$\overline{\dot{\omega}} = \sqrt{\frac{a}{\mu}}\frac{1}{e}\left[-\frac{1}{2}\sqrt{1-e^2}\,\alpha_1^R + e\sqrt{1-e^2}\,\alpha_0^R + \frac{1}{2}\left(2-e^2\right)\beta_1^S - \frac{1}{4}e\beta_2^S\right] - \cos i\,\overline{\dot{\Omega}} \tag{2.5}$$

$$\begin{aligned}\overline{\dot{\varepsilon}_1} = {} & \sqrt{\frac{a}{\mu}}\left[\left(-2-e^2\right)\alpha_0^R + 2e\alpha_1^R - \frac{1}{2}e^2\alpha_2^R\right] + \\ & \left(1-\sqrt{1-e^2}\right)\left(\overline{\dot{\omega}} + \overline{\dot{\Omega}}\right) + 2\sqrt{1-e^2}\sin^2\left(\frac{i}{2}\right)\overline{\dot{\Omega}}.\end{aligned} \tag{2.6}$$

The Fourier coefficients are a representation of periodic thrust laws.

$$\vec{\mathbf{F}} = F_R\hat{\mathbf{r}} + F_W\hat{\mathbf{w}} + F_S\left(\hat{\mathbf{w}} \times \hat{\mathbf{r}}\right) \tag{2.7}$$

$$F_{R,W,S} = \sum_{k=0}^{\infty}\left[\alpha_k^{R,W,S}\cos kE + \beta_k^{R,W,S}\sin kE\right]. \tag{2.8}$$

In [7], it was found that only a finite set of the Fourier coefficients $\alpha_k^{R,W,S}, \beta_k^{R,W,S}$ affect the average trajectory dynamics. This set is used as the control input u. Since we are implementing linear quadratic trajectory control, the system dynamics take the state-space linear form:

$$\dot{x} = Ax + Bu \tag{2.9}$$

u is a thrust acceleration vector in terms of Fourier coefficients, A is a six-by-six zero

matrix, and B is expressed as:

$$\mathbf{B} = \begin{bmatrix} 0_{2,3} & B_1 & 0_{2,7} \\ 0_{2,3} & 0_{2,4} & B_3 \\ B_4 & 0_{2,4} & B_5 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} alpha_{\beta_{1R}} & \alpha_{\alpha_{0s}} & 0 & 0 \\ e_{\beta_{1R}} & e_{\alpha_{0s}} & e_{\alpha_{1s}} & e_{\alpha_{2s}} \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 0 & 0 & i_{\alpha_{0w}} & i_{\alpha_{1w}} & i_{\alpha_{2w}} & i_{\beta_{1w}} & i_{\beta_{2w}} \\ 0 & 0 & \Omega_{\alpha_{0w}} & \Omega_{\alpha_{1w}} & \Omega_{\alpha_{2w}} & \Omega_{\beta_{1w}} & \Omega_{\beta_{2w}} \end{bmatrix} \tag{2.10}$$

$$B_4 = \begin{bmatrix} \omega_{\alpha_{0r}} & \omega_{\alpha_{1r}} & 0 \\ M_{\alpha_{0r}} & M_{\alpha_{1r}} & M_{\alpha_{2r}} \end{bmatrix}$$

$$B_5 = \begin{bmatrix} \omega_{\beta_{1s}} & \omega_{\beta_{2s}} & \omega_{\alpha_{0w}} & \omega_{\alpha_{1w}} & \omega_{\alpha_{2w}} & \omega_{\beta_{1w}} & \omega_{\beta_{2w}} \\ M_{\beta_{2s}} & M_{\beta_{2s}} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The state vector x is in terms of six orbital elements and the input control law u is in terms of the fourteen Fourier thrust-acceleration components:

$$\mathbf{x} = \begin{bmatrix} a & e & i & \omega & \Omega & M \end{bmatrix}^T$$

$$\mathbf{u} = \begin{bmatrix} \alpha_0^R & \alpha_1^R & \alpha_2^R & \beta_1^R & \alpha_0^S & \alpha_1^S & \alpha_2^S & \beta_1^S & \beta_2^S & \alpha_0^W & \alpha_1^W & \alpha_2^W & \beta_1^W & \beta_2^W \end{bmatrix}^T \tag{2.11}$$

Linear Quadratic Optimal Control

LQ Optimal Control is a subset of optimal control. As the name implies, linear quadratic optimal control minimizes a quadratic cost function for a linear system. In some cases, this method can be used for non-linear systems when the system dynamics are linearized about an operating point.

The state equation for a linear system is modelled as:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ x(0) &= x_0 \end{aligned} \tag{2.12}$$

and the control law is defined as:

$$u = \phi(x) \tag{2.13}$$

This control law must minimize a quadratic cost function(2.14). The cost function represents the total penalty incurred over the entire time span of the simulation. The goal of using a linear quadratic controller is to minimize the code function.

$$J(x_0, \phi) = \frac{1}{2}x(t)^T K_f x(t) + \frac{1}{2}\int_0^\infty (x^T(t)Qx(t) + u(t)^T Ru(t))dt \tag{2.14}$$

Where Q is a positive semidefinite matrix and R is a positive definite matrix.

$$Q = \begin{bmatrix} q_1 & 0 & 0 & \dots & 0 \\ 0 & q_2 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & q_n \end{bmatrix} \tag{2.15}$$

$$R = \begin{bmatrix} r_1 & 0 & 0 & \dots & 0 \\ 0 & r_2 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \dots & \vdots \\ \vdots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & r_n \end{bmatrix}$$

Q and R are weight factor matrices whose values are determined by using trial and error, and insight. These weight factors are used to incur a penalty on the state x(t), input u(t), or final value

The principle of linear quadratic optimal control can be stated as:

Assume (A,B) is stabilizable, and (Q,A) is detectable. Then there exists a unique solution S, in the class of positive semidefinite matrices, to the objective Riccati equation. Furthermore, the closed-loop system matrix $A - BR^{-1}B^{T}S$ is stable.[11]

The control law that minimizes the cost function can be represented in (2.16)

$$u(t) = -R(t)^T B_T(t) P(t) x(t) \tag{2.16}$$

where P is the solution to the differential Riccati equation over a finite horizon continuous time, given in (2.17).

$$\begin{gathered} \dot{P} = A^T P(t) + P(t)A + R - P(t)BR^{-1}B^T P(t) \\ P(t_f) = 0 \end{gathered} \tag{2.17}$$

The differential Riccati equation is solved in backwards time $t_f \rightarrow t_0$. The fundamentals of trajectory dynamics and low-thrust maneuvers can be demonstrated in a trajectory control simulation.

Trajectory Control

The LQ trajectory control was implemented with two different orbit scenarios. The first orbit scenario is an orbit raising maneuver and the second is a plane change maneuver. Through trial and error, both orbit maneuvers use the weight factors values given below:

$$K_f = 1000 I_6$$

$$\mathbf{Q} = \begin{bmatrix} .1 & 0 & 0 & 0 & 0 & 0 \\ 0 & .1 & 0 & 0 & 0 & 0 \\ 0 & 0 & .1 & 0 & 0 & 0 \\ 0 & 0 & 0 & .1 & 0 & 0 \\ 0 & 0 & 0 & 0 & .1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.18}$$

For **Q** matrix (2.18), the mean anamoly **Q**(6,6) is not being targeted. The first scenario is an orbit raising scenario over a period of five orbits. The initial and target parameters of the trajectory are given in the tables below.

Table 2.1: Initial Orbit State For Raising Maneuver

Orbital elements		Units
a_0	6678	km
e_0	0.67	
i_0	20	degrees
Ω_0	20	degrees
ω_0	20	degrees
M_0	20	degrees

Table 2.2: Target Orbit State For Raising Maneuver

Orbital elements		Units
a_{targ}	7345	km
e_{targ}	0.7370	
i_{targ}	22	degrees
Ω_{targ}	22	degrees
ω_{targ}	22	degrees
M_{targ}	20	degrees

The result from the trajectory model is shown below in terms of orbital elements and a 3D plot.

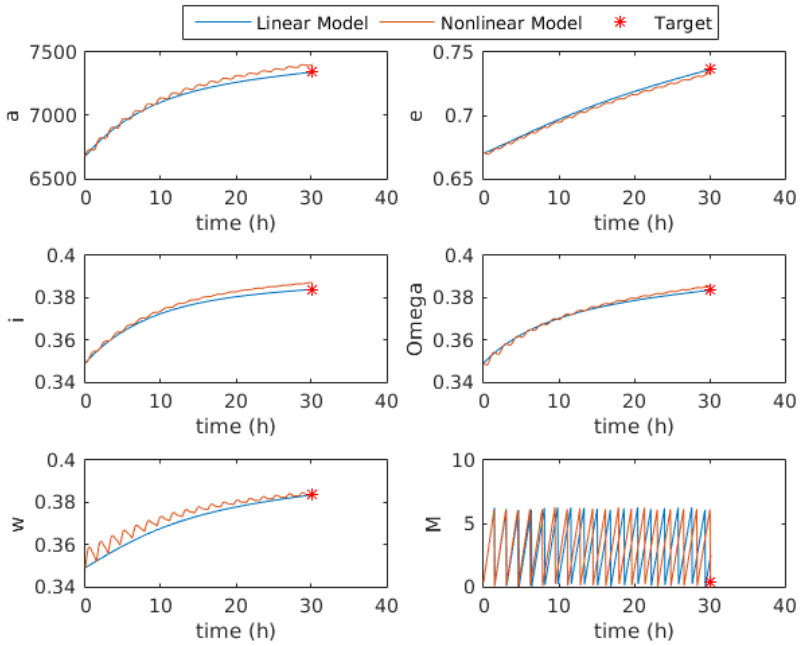

Figure 2.1: Trajectory In Terms Orbital Elements

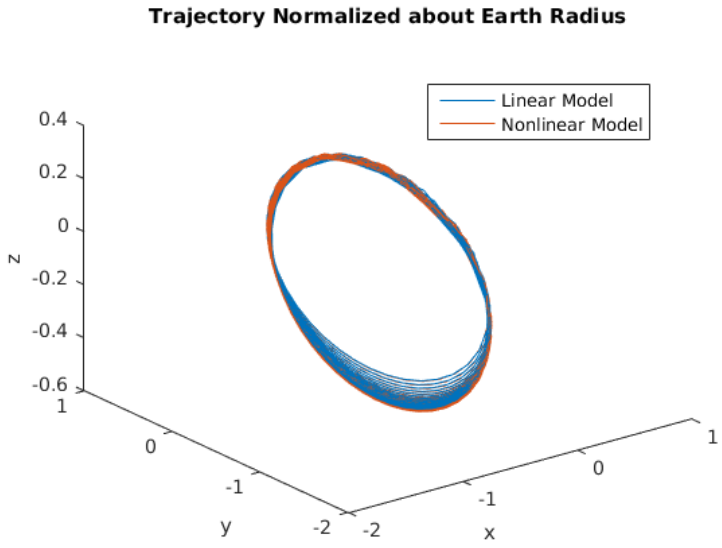

Figure 2.2: 3D Trajectory Of Orbits

Figure 2.1 and figure 2.2 show the trajectory of the spacecraft in orbital elements and a 3D plot in Cartesian coordinates, respectively. The periodic motion in figure 2.1 represents the orbital element states using a non-linear model, the blue line represents the orbital element states using averaged secular equations(2.1-2.6), and the star is the target state. The solutions have the form of time-varying thrust acceleration vectors. The direction of the thrust-acceleration vectors define attitude angles that will be used in the attitude controller.

The second scenario is a plane change maneuver. In the plane change maneuver the initial state inclination is twenty degrees and the target state is ten degrees, the initial and target states are shown in the tables below.

Table 2.3: Initial Plane Change Maneuver

Orbital elements		Units
a_0	6678	km
e_0	0.67	
i_0	20	degrees
Ω_0	20	degrees
ω_0	20	degrees
M_0	20	degrees

Table 2.4: Target Plane Change Maneuver

Orbital elements		Units
a_{targ}	7345	km
e_{targ}	0.67	
i_{targ}	10	degrees
Ω_{targ}	20	degrees
ω_{targ}	20	degrees
M_{targ}	20	degrees

The results for this orbit maneuver is shown in the plots below.

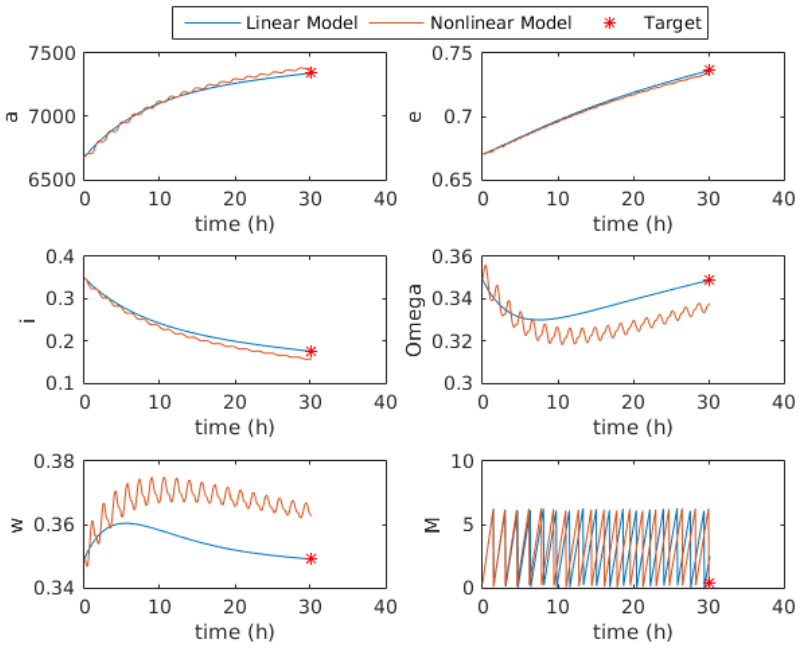

Figure 2.3: Orbital Elements Plane Change Maneuver

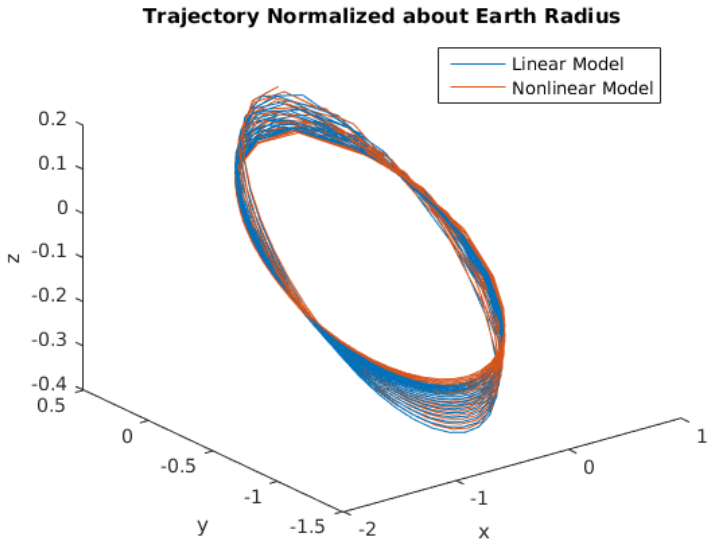

Figure 2.4: 3-D Trajectory Plane Change

It can be seen in Figure 2.3 that the LQ control model does reach the target state but the non-linear model misses the target state. This issue may be due to any singularities present in the non-linear function. To find a solution, would require looking outside the scope of this thesis. Now that the trajectory control has been discussed an attitude model can be built using the same principles.

Chapter 3

ATTITUDE DYNAMICS

Attitude control is necessary to control the orientation and the spin of a satellite. In space, the overwhelming force that acts on a spacecraft is gravity and thrust (neglecting drag forces). Attitude control is necessary to successfully navigate a spacecraft because it allows a spacecraft to point to a target orientation for thrusting maneuvers or to point sensors or cameras at an object.

A basic way to think of attitude control is to first consider that gravity is the only force present. Another assumption that can be made is that the center of mass is at the center of the spacecraft, referred to as torque-free motion. In torque-free motion the external disturbance torques on the spacecraft are zero.

A key component of attitude control is the inertia of the spacecraft and the angular momentum. The inertia matrix is dependent on the geometry and mass of the spacecraft, its general form is given in the matrix(3.1).

$$\mathbf{J} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yx} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{bmatrix} \tag{3.1}$$

The angular momentum about the body axis of the spacecraft is given as

$$\mathbf{H} = \mathbf{J}\dot{\theta}$$

$$\dot{\theta} = \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} \tag{3.2}$$

where $\dot{\theta}$ is the angular rotation of the spacecraft body. Spacecraft attitude control systems typically use Euler angles and quaternions to represent their orientation.

Coordinate Systems

When modeling attitude control typically two major types of coordinate systems are used. The easier to understand and visualize are Euler angles. Similar to an aircraft, a spacecraft also can execute roll, pitch, and yaw maneuvers. Euler angles can be expressed in terms of radians or degrees of rotation about the roll, pitch, and yaw axes as shown in Figure 3.1 below.

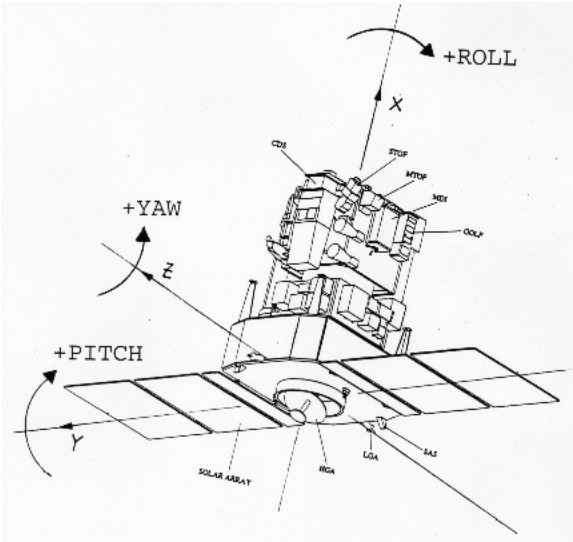

Figure 3.1: Euler Angle Representation On A Satellite[9]

θ x,y, and z represent the roll, pitch, and yaw respectively

$$\theta = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \tag{3.3}$$

Euler angles are not very computationally intensive as compared to quaternions, but can have singularities or points where the value is undefined at certain angles.

Quaternions are another commonly used coordinate system. Unlike Euler angels, which can be visualized as roll, pitch, and yaw, quaternions can not be intuitively visualized. However, quaternions provide a useful mathematical representation of spacecraft attitude without singularities. Quaternions are more robust than Euler angles because they allow for more computationally intensive simulations. Quaternions can be expressed as:

$$\mathbf{q} = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{3.4}$$

where q_x, q_y, q_z are quaternions along the x, y, and z axis and q_s is a scaling factor.

The direction cosine transformation matrix from a vector A to B can be expressed as [5]:

$$C^{B/A} = \begin{bmatrix} \vec{b}_1 \cdot \vec{a}_1 & \vec{b}_1 \cdot \vec{a}_2 & \vec{b}_1 \cdot \vec{a}_3 \\ \vec{b}_2 \cdot \vec{a}_1 & \vec{b}_2 \cdot \vec{a}_2 & \vec{b}_2 \cdot \vec{a}_3 \\ \vec{b}_3 \cdot \vec{a}_1 & \vec{b}_3 \cdot \vec{a}_2 & \vec{b}_3 \cdot \vec{a}_3 \end{bmatrix} \tag{3.5}$$

To convert Euler angles to quaternions, the Euler angles must first be converted into a direction cosine matrix (DCM) [10].

$$T = \begin{bmatrix} \sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta z & \sin\theta_x \sin\theta_y \cos\theta_z + \cos\theta_x \sin\theta z & -\sin\theta_x \cos\theta_y \\ -\cos\theta_y \sin\theta_z & -\cos\theta_y \cos\theta_z & \sin\theta_z \\ \cos\theta_x \sin\theta_y \sin\theta_z + \sin\theta_x \cos\theta z & -\cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x \sin\theta z & -\cos\theta_x \cos\theta_y \end{bmatrix} \tag{3.6}$$

From direction cosine matrix the values can then be converted into quaternions.

$$
\begin{aligned}
q_s &= \sqrt{\tfrac{1}{4}(1+T_{11}+T_{22}+T_{33})} \\
q_x &= \sqrt{\tfrac{1}{4}(1+T_{11}-T_{22}-T_{33})} \\
q_y &= \sqrt{\tfrac{1}{4}(1-T_{11}+T_{22}-T_{33})} \\
q_z &= \sqrt{\tfrac{1}{4}(1-T_{11}-T_{22}+T_{33})}
\end{aligned}
\tag{3.7}
$$

The spacecraft attitude representation in the Earth-centered inertial reference frame is dependent upon the coordinate location of the spacecraft relative to the center of the Earth as shown in figure 3.2 below.

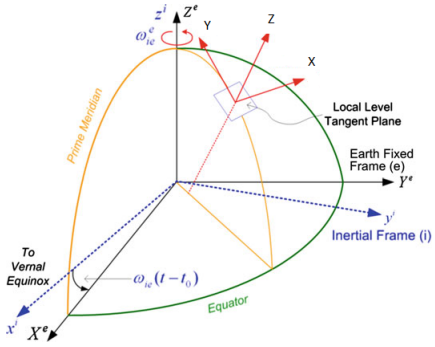

Figure 3.2: Earth Inertial Reference Frame [19]

This coordinate system can be derived from the x,y,z coordinates of the spacecraft and the rotational velocity of the Earth.

In attitude dynamics using the Earth-centered inertial frame may not as be useful. It is typical to represent the attitude of a spacecraft in the body frame because of on board sensors and attitude systems. The orientation angles must be transformed from the inertial frame to the body frame. Putting the orientation Euler angles in the transformation matrix

(3.8), the orientation is given as a direction cosine matrix.

$$R_{bi} = \begin{bmatrix} \cos\theta_s \cos\theta_w & \cos\theta_w \sin\theta_s & -\sin\theta_w \\ \cos\theta_s \sin\theta_r \sin\theta_w - \cos\theta_r \sin\theta_s & \cos\theta_r \cos\theta_s + \sin\theta_r \sin\theta_s \sin\theta_w & \cos\theta_w \sin\theta_r \\ \sin\theta_s \sin\theta_r + \cos\theta_s \cos\theta_r \sin\theta_w & \cos\theta_r \sin\theta_s \sin\theta_w - \cos\theta_s \sin\theta_r & \cos\theta_r \cos\theta_w \end{bmatrix} \tag{3.8}$$

The transformation matrix may be converted into quaternions by using(3.9).

$$\begin{aligned} q_s &= \sqrt{\tfrac{1}{4}(1 + R_{bi11} + R_{bi22} + R_{bi33})} \\ q_x &= \sqrt{\tfrac{1}{4}(1 + R_{bi11} - R_{bi22} - R_{bi33})} \\ q_y &= \sqrt{\tfrac{1}{4}(1 - R_{bi11} + R_{bi22} - R_{bi33})} \\ q_z &= \sqrt{\tfrac{1}{4}(1 - R_{bi11} - R_{bi22} + R_{bi33})} \end{aligned} \tag{3.9}$$

This paper uses an alternative method to represent the attitude by using the a transformation on the thrust-acceleration vectors of the spacecraft rather than using the body frame.

Attitude Control Systems

There are various ways to control the attitude of a spacecraft. One way is to use reaction wheels. Reaction wheels use spinning wheels, generating rotational momentum for a spacecraft. One reaction wheel is used to spin a spacecraft about one axis, so for two or three axis control, two or three reaction wheels are necessary. The reaction wheels are contained within an enclosure and can be placed anywhere within a spacecraft. These reaction

wheels can build up momentum over time and become over saturated. Therefore, reaction wheels are often paired with a magnetorquer to reduce the built up momentum from the reaction wheels.

The magnetorquer is a component that generates a magnetic field when a voltage is applied and can be used to decrease a reaction wheel's momentum. One caveat with using a reaction wheel and magnetorquer is that they must work with another magnetic field, (Earth's magnetic field) so this combination will not be effective at all outside of the range of planetary magnetic fields, for example during interplanetary missions. In the case where reaction wheels and magnetorquers can not be used, a thruster can be used as an attitude control system.

Attitude control thrusters can be used for high orbit and interplanetary missions. These systems are more complex than using reaction wheels and magnetorquers, and require fuel tanks, fuel system accessories like valves or pumps, and may require the spacecraft to go though more rigorous ground testing.

Another attitude controller is a nutation damper. A spacecraft can undergo wobbling, or rocking motions in the axis of rotation, this is known as nutation. Nutation is dampened by means of fluid frictional force, and stretching and compression of non-rigid components which dissipates energy causing the spacecraft to stabilize[6]. Nutation dampers use a tube filled with viscous fluid, a mass, and a spring as shown in figure 3.3 below.

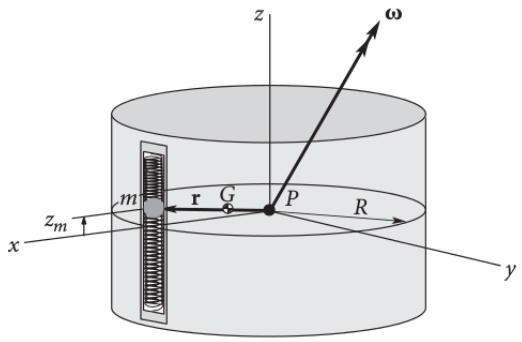

Figure 3.3: Diagram Of A Nutation Damper[6]

Another attitude control mechanism is gravity-gradient stabilization. Gravity-gradient stabilization uses a tethered mass or a boom and a planet's gravitational field to stabilize itself, as seen in figure 3.4. The benefits to using this system is that it requires little power but the spacecraft is limited in mobility.

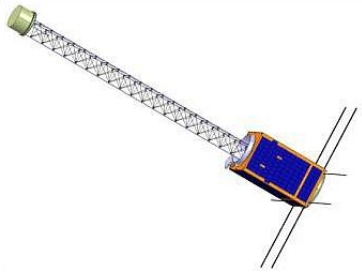

Figure 3.4: Gravity-Gradient Stabilization On A CubeSat[13]

All of the attitude control mechanisms mentioned above can be implemented on low-thrust spacecraft. The attitude control laws demonstrated in this thesis could be implemented by any attitude control system that can vary its torque output, such as reaction wheels or thrusters.

Chapter 4

ATTITUDE MODEL

Before a LQ attitude control model is discussed, literature on attitude control will be discussed. Other works use a reduced quaternion model to model the attitude[14]. Yang also illustrated that an attitude controller can be modelled by linearizing the equations for attitude mechanics[15]. By using only three quaternion components, Yang was able to accurately show an analytic model for an attitude controller.

Caulbert and Biggs proposed a singularity-free controller based on a special set of attitude relations both globally and uniquely described on the special orthogonal group[16]. Their attitude control model uses a rotation matrix which does not have some of the pitfalls that Euler angles, or quaternions have but requires more computational overhead[16].

There are some assumptions that must be made for creating an attitude model. The spacecraft is assumed to have a rigid body, and the spacecraft's center of mass is located in the center of the spacecraft[17]. The non-linear dynamics of spacecraft attitude in terms of quaternions can be described as [17]:

$$\begin{aligned} \dot{q} &= \tfrac{1}{2}(S(q) + q_4 I_{3x3})\boldsymbol{\omega} \\ \dot{q_4} &= -\tfrac{1}{2}\boldsymbol{\omega}^T q \end{aligned} \tag{4.1}$$

linearizing these non-linear equations we can then implement LQ optimal control to find

an approximate solution to the targeting problem.

$$\begin{bmatrix} \dot{\omega} \\ \\ \dot{q} \end{bmatrix} = \mathbf{A}x + \mathbf{B}u$$

$$x = \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 & q_1 & q_2 & q_3 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 0_3 & 0_3 \\ \\ .5I_3 & 0_3 \end{bmatrix} \tag{4.2}$$

$$\mathbf{B} = \begin{bmatrix} J^{-1} \\ \\ 0_3 \end{bmatrix}$$

where A and B are the state matrix and input matrix, respectively and, 0_3 and I_3 are three by three zero and identity matrices, respectively. For a linear quadratic system, the cost function that must be minimized is given as :

$$L = \frac{1}{2}x(t)^T K_f x(t) + \frac{1}{2}\int_0^\infty (x^T Q x + u(t)^T R u)dt \tag{4.3}$$

The weight factors for the state, input, and final parameters, respectively are given in the matrices below:

$$\mathbf{Q} = I_6$$
$$\mathbf{R} = I_3 \tag{4.4}$$
$$\mathbf{K}_f = I_6$$

A finite-horizon Riccati equation and an input matrix are used to determine the LQ optimal control:

$$-\dot{P} = A^T P + PA - PBR^{-1}B^T P + Q \tag{4.5}$$
$$P(T) = K_f \tag{4.6}$$
$$u_t = -R^{-1}B^T Px \tag{4.7}$$

where equation (4.5) is the differential Riccati equation used in optimal control to minimize cost. P is integrated in discrete backwards time starting from the final time to the initial time, (4.7) represents the control torque of the spacecraft.

Linear Quadratic Attitude Control

LQ model was simulated on a CubeSat. The dimensions of the CubeSat that was used are (10 cm x 10 cm x 30 cm) with a mass of three kilograms. It is assumed that the center of mass is located at the center of the spacecraft. Assuming that a CubeSat takes a rectangular shape (non-deployable solar panels) the mass moment of inertia for the x, y, and z axes respectively can be described as [6, p.417]:

$$
\begin{aligned}
J_{xx} &= \tfrac{1}{12} m(a^2 + l^2) \\
J_{yy} &= \tfrac{1}{12} m(b^2 + l^2) \\
J_{zz} &= \tfrac{1}{12} m(a^2 + b^2)
\end{aligned}
\tag{4.8}
$$

where m is the mass of the spacecraft, a, b, and l are the spacecraft width (10 cm), height (10cm), and length (30cm), respectively. An inertia matrix J, can be defined where the diagonals J_{xx}, J_{yy}, and J_{zz} represent the inertia of a 3-U CubeSat and the off-diagonals are assumed to be zero.

$$
\mathbf{J} = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}
\tag{4.9}
$$

The attitude control model uses the direction of the thrust-acceleration components (F_r, F_w, F_s) and the resultant force vector (F_t) as its target state. The thrust input for this system is used in the attitude control model and is expressed in an inertial frame[12].

$$
\begin{aligned}
F_R &= \alpha_0^R + \alpha_1^R \cos(E) + \alpha_2^R \cos(2E) + \beta_1^R \sin(E) \\
F_S &= \alpha_0^S + \alpha_1^S \cos(E) + \alpha_2^S \cos(2E) + \beta_1^S \sin(E) + \beta_2^S \sin(2E) \\
F_W &= \alpha_0^W + \alpha_1^W \cos(E) + \alpha_2^W \cos(2E) + \beta_1^W \sin(E) + \beta_2^W \sin(2E) \\
F_t &= \sqrt{F_R^2 + F_S^2 + F_W^2}
\end{aligned}
\tag{4.10}
$$

The equations in (4.10) give the direction vectors that the attitude controller will use for solving the attitude targeting problem.

Direction Vector Transformations

The objective of the linear quadratic attitude control is to track the direction of the thrust acceleration vector throughout the orbit change maneuver. The attitude is tracked by solving a set of boundary value problems over a continuous time span. The time span is split into intervals with initial and target conditions. The linear quadratic controller solves for the final value of the attitude for the interval. The final values are then used as the initial values for the next interval. The parameters of the boundary value problem are the direction vector of the resultant thrust-acceleration components. The thrust-acceleration vectors are expressed as (F_r, F_w, F_s) as expressed in equation (4.10). Each of these vectors have corresponding direction vectors ($\hat{r}$, $\hat{w}$, $\hat{s}$) and a resultant $\hat{t}$ as defined in equation 4.11. These direction vectors are the initial and target states for the attitude controller.

$$\begin{aligned} \hat{r} &= \frac{\vec{r}}{|\vec{r}|} \\ \hat{w} &= \frac{\vec{r}\times\vec{v}}{|\vec{r}\times\vec{v}|} \\ \hat{s} &= \hat{w}\times\hat{r} \\ \hat{t} &= \frac{\hat{r}+\hat{w}+\hat{s}}{|\hat{r}+\hat{w}+\hat{s}|} \end{aligned} \tag{4.11}$$

The direction vectors are expressed as x, y, and z coordinates in the Earth inertial frame. To determine the angle between the resultant vector and the pointing direction of the space-

craft, the Euler's Eigenaxis Rotation Theorem is used. For each interval of the thrust-acceleration tracking problem the Euler Eigenaxis Rotation is expressed as:

$$\hat{F}_t = \mathbf{R}\hat{a}\hat{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{4.12}$$

where $\hat{a}$ is the initial pointing direction. We assume that at the beginning of each time interval, the spacecraft is oriented with the body y-axis along the initial direction of the thrust acceleration vector. At the end of each interval, the spacecraft must align with the new direction of the thrust vector. The transformation vector R represents the required rotation and both $\hat{F}_t$ and $\hat{a}$ must be unit vectors. By selecting equation (4.12) as the pointing vector, it can be assumed that the y component is zero. The spin of the spacecraft is not being tracked, only yawing and pitching movements are. To align to the resultant vector, a direction cosine matrix **R** must be determined[18].

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + s(\hat{a} \times \hat{F}_t) + (s(\hat{a} \times \hat{F}_t)^2 (\frac{1 - \hat{a} \cdot \hat{F}_t}{|\hat{a} \times \hat{F}_t|}) \tag{4.13}$$

where is defined as the skew symmetric matrix. The skew symmetric matrix is a square matrix where the transpose of the matrix is also its negative. A one-by-three matrix can be expressed as a skew symmetric matrix in equation (4.14).

$$\mathbf{A} = \begin{bmatrix} a & b & c \end{bmatrix}$$

$$s(\mathbf{A}) = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \tag{4.14}$$

When the rotation matrix is known, the quaternions can be determined by the Eigenaxis e, and the pointing angle θ:

$$\begin{aligned} \theta &= \cos^{-1}\left(\tfrac{trace(R)-1}{2}\right) \\ e &= \tfrac{1}{2\sin\theta}\begin{bmatrix} R_{23}-R_{32} \\ R_{31}-R_{13} \\ R_{12}-R_{21} \end{bmatrix} \\ q_1 &= e_1 \sin\tfrac{\theta}{2} \\ q_2 &= e_2 \sin\tfrac{\theta}{2} \\ q_3 &= e_3 \sin\tfrac{\theta}{2} \\ q_4 &= \cos\tfrac{\theta}{2} \end{aligned} \tag{4.15}$$

Where q_4 is the scale value. These quaternions give the angles between the resultant direction and the desired direction to point the satellite for the attitude controller.

Attitude Control Implementation

The attitude control model is implemented using the quaternions of the thrust-acceleration vector components in the initial and target states of the attitude. The trajectory for the orbit raising maneuver was modelled first, followed by the plane change maneuver. The time span used for the attitude controller is the time from one angle to the other. The spacecraft will perform twenty orbits. The initial state for the angular velocity is assumed to be zero (spacecraft is not rotating). The initial orientation state is the initial value of the thrust direction. The script file of the trajectory and attitude controller is given in the appendix.

The thrust-acceleration of the orbit raising manuever is given in figure 4.1 below.

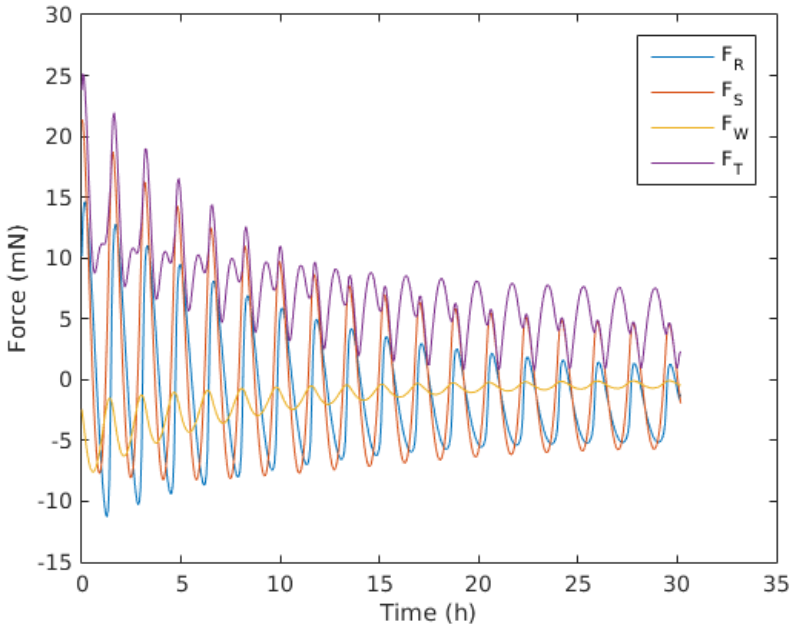

Figure 4.1: Orbit Raising Thrust-Acceleration

It can be seen from figure 4.1, the thrust acceleration is periodic. The direction vectors of the thrust-acceleration components F_R, F_S, F_W, are implemented into the attitude controller.

Figure 4.2 shows the output of the attitude transformations in terms of quaternions as well as the angular velocity and torque input to the spacecraft, and the thrust-acceleration.

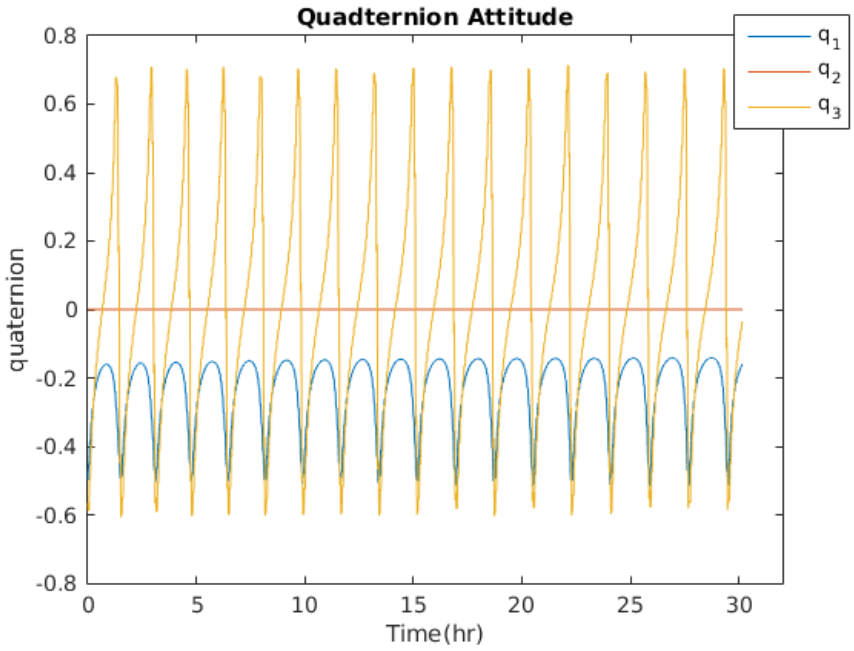

Figure 4.2: Quaternion Output Of Orbit Raising Maneuver

Figure 4.2 shows that the quaternion transformation follows a periodic motion. This periodic behavior of the orientation is expected because the spacecraft is moving along a periodic orbit. For the quaternion q_2 it can be seen that throughout the flight the value is zero, assuming the spacecraft is not spinning along its axis. Observing the quaternion attitude, q_3 seems to be affected the most by an orbit-raising maneuver.

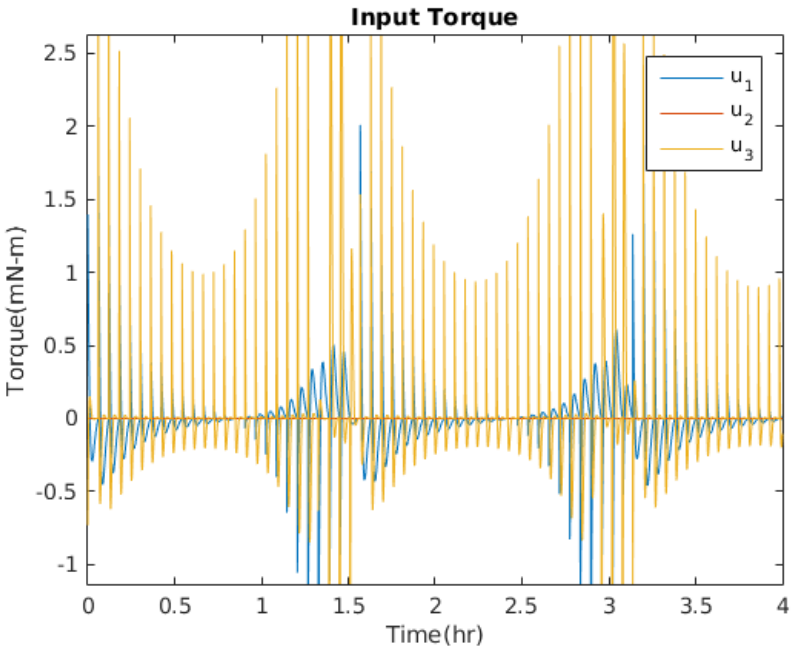

Figure 4.3: Torque Input Of Orbit Raising Maneuver

A snippet of the input torque u shown in figure 4.3. Similarly to theattitude output, q_3 appears to experience the highest input torque and the input torque also has periodic behavior.

The thrust-acceleration of the plane change manuever is given in figure 4.4 below.

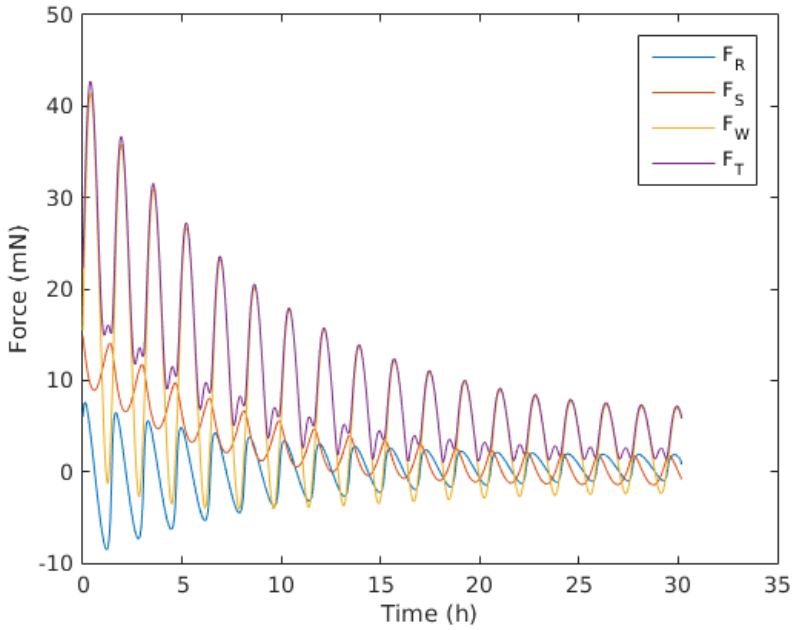

Figure 4.4: Plane Change Thrust-Acceleration

Figure 4.4 above, shows the thrust-acceleration components for the plane change maneuver. It can be seen that the thrust-acceleration for the plane change maneuver has a higher magnitude companed to the orbit raising maneuver. Also the thrust acceleration gets smaller over time, faster than the orbit raising maneuver. The results of the attitude controller for the plane change maneuver is shown in the figure below.

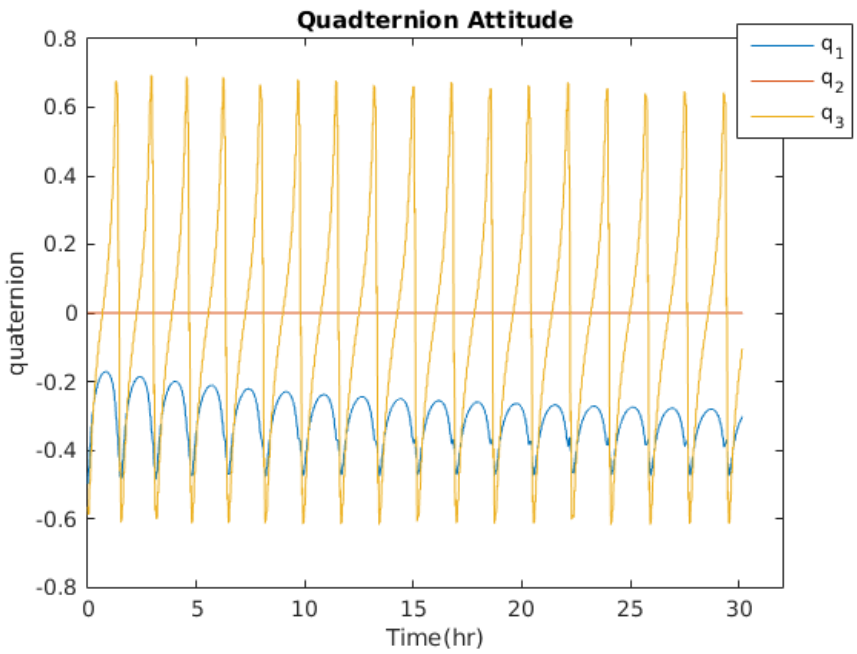

Figure 4.5: Attitude Plane Change

Looking at figure 4.5, the output for the attitude of the plane change, the attitude seems to have similar behavior to the orbit raising maneuver. As expected q_2 remains zero for the entire time span. It can be seen from figure 4.5 that over the time span the magnitude ofq_1 decreases. The magnitude decrease may be due to the plane change.

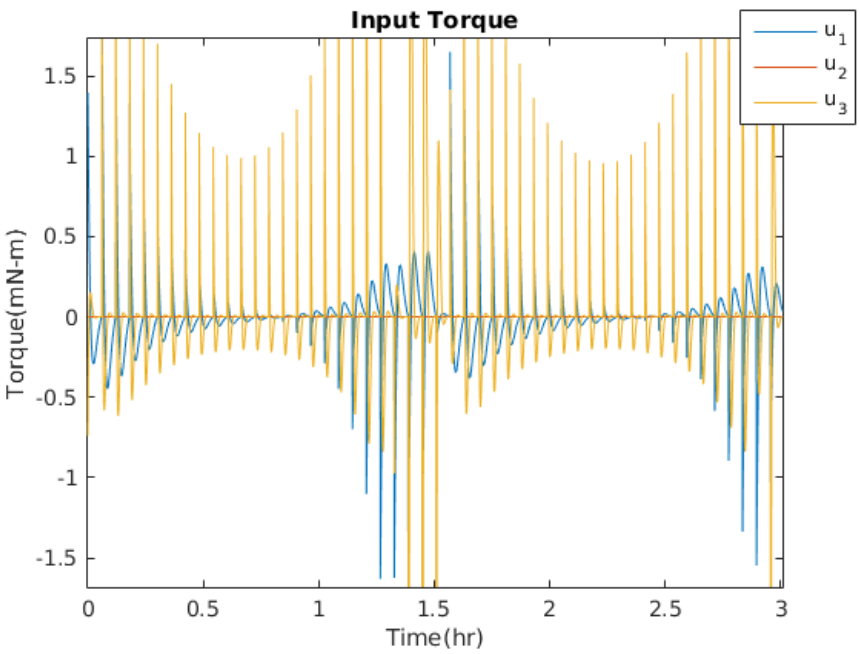

Figure 4.6: Input Torque For Plane Change

Observing figure 4.6, the input torque for the plane change is similar to the orbit raising maneuver both in magnitude and periodic behavior.

Chapter 5

CONCLUSION

A linear quadratic trajectory controller was created to determine the required thrust accelerations and control torques to execute orbit maneuvers. This trajectory model was used in two orbit scenarios, an orbit-raising and a plane change maneuver. Then a linear quadratic attitude controller was developed to control the attitude of a spacecraft to track the required thrust vector during flight. The attitude controller tracked the direction of the thrust-acceleration direction vector. The attitude was modelled by dividing the trajectory output of the spacecraft as individual boundary value problems with initial and target states, and then solving each of these problems using linear quadratic optimal control. The attitude controller exhibits periodic behavior with the same frequency as the calculated spacecraft thrust acceleration.

The LQ trajectory controller models the trajectory of a low-thrust maneuver and then the attitude controller uses the results from the trajectory controller to model the orientation of the spacecraft. Possible future work can include integrating the attitude controller directly with the trajectory controller. The attitude controller can serve as a constraint to the trajectory controller by forcing the trajectory controller to find a new solution if the attitude controller can not achieve a given attitude rate.

REFERENCES

[1] "NASA's CubeSat Launch Initiative", *nasa.gov/directorates/somd/home/CubeSats¡nitiative.html*, accessed November 19,2015

[2] "PEPEL Thrusters: CubeSat Ambipolar Thruster", University of Michigan, Department of Aerospace Engineering *http://pepl.engin.umich.edu/thrusters/CAT.html* date accessed, November 19, 2015

[3] Kolosa, D., Spangelo, S., Lemmer, K., Hudson, J., "em Mission Analysis for a Micro RF Ion Thruster for CubeSat Orbital Maneuvers", Western Michigan University, Kalamazoo Mi., Jet Propulsion Laboratory, Pasedena, CA., 2014

[4] "ION Thrusters", Busek Space Propusion and Systems, *http://busek.com*, date accessed November 19, 2015

[5] Wig, B., "Space Vehicle Dynamics and Control" ed. 2nd, AIAA Education Series, 2008

[6] Curtis, H., D., "Orbital Mechanics for Engineering Students", Ed.1, Elsevier Butterworth-Heinemann, January 10, 2005

[7] Hudson, J. S., Scheeres, D., J., "Reduction of Low-Thrust Continuous Controls for

Trajectory Dynamics, Journal of Guidance, Control, and Dynamics, Vol.32, No.3, May-June 2009

[8] Hudson, J. S., Scheeres, D., J., "em Orbital Targeting Using Reduced Eccentric Anomaly Low-Thrust Coefficients", Journal of Guidance, Control, and Dynamics, Vol.34, No.3, May-June 2011

[9] "The Last Minutes on June 25, 1998 Before the Loss of Trajectory", sohowww.nascom.nasa.gov/about/Recovery/ESR7/

[10] Grofsekatthofer, K., Zizung, Y., "*Introduction into quaternions for spacecraft attitude representation*", Technical University of Berlin, Department of Astronautics and Aeronautics, Berlin, Germany, May 31, 2012

[11] Kwong, R.,H., "Linear Quadratic Optimal Control",chapter 6, ECE 410 Control Sytems, Systems Control Group, University of Toronto

[12] Kolosa, D., Hudson, J., "Linear Model For Low-Thrust Spiral Orbits and Optimal Control", AIAA Guidance, Navigation, and Control Conference, doi: 10.2514/6.2015-2013

[13] BUAA-Sat (Beihang University Student Microsatellite), Earth Observation Portal, $directory.eoportal.org/web/eoportal/-/buaa-sat$, date accessed: November 19, 2015

[14] Yang, Y., "*Attitude control in spacecraft orbit-raising using a reduced quaternion model*" *Advances in Aircraft and Spacecraft Science, Vol.1, No. 4*, 2014, pg.427-441

[15] Yang, Y., "Analytic LQR Design for Spacecraft Control System Based on Quaternion Model", ascelibrary.org, American Society for Civil Engineers

[16] Caulbert A., Biggs, J.,D., "An efficient, singularity free attitude controller on SO(3)"

[17] McClamroch N.,H., "AE 573 Spacecraft Dynamics and Control [lecture notes]", University of Michigan, 2006

[18] De Ruiter, A., j., Damaren, C., J., Forbes, J., R., "Spacecraft Dynamics and Control An Introduction", Whiley, 2012

[19] Noureldin, A., Karamat, T.B., Georgy, J., " Fundementals of Inertial Navigation, Satellite-based Positioning and their integration", ISBN: 978-3-642-30465-1

APPENDIX

Description of Matlab Code

Below is the Matlab code that was created to produce the results given in the paper. The main file is $driver_LQRASE_JHudson.m$, this file contains the script for the LQ trajectory control and the function of the LQ attitude controller. The code below initializes the initial state and target state or the orbit as well as the constants. The semi-major axis a0 and atarg are normalized with the radius of the Earth. The transfer time indicates the number of orbits, and the target and initial states are converted from orbital elements to Cartesian coordinates using the oe_rv function.

```
%Reframing ASE targeting as an LQ optimal control problem
global mu A B Q R Pbig tspan r xT dist u  tspan_bk
% Constants
mu = 398600*60^4/RE^3;   %RE^3/h^2
RE=6378;            % Earth radius to normalize distances

% Initial State
a0 = 6678/RE;         %km
e0 = 0.67;
i0 = 20*pi/180;     %rad
Omega0 = 20*pi/180; %rad
```

```
12 w0 = 20*pi/180;      %rad
13 M0 = 20*pi/180;      %rad
14 n0 = sqrt(mu/a0^3);
15 x0 = [a0; e0; i0; Omega0; w0; M0];
16
17 % Transfer Time
18 t0 = 0;              % start time
19 ttarg = 2*pi*sqrt(a0^3/mu)*20;      % target time;
20 dt = ttarg/500;
21 tspan = 0:dt:ttarg;
22 \end{minted}
23 \begin{minted}[linenos=true,bgcolor=bg]{matlab}
24 tspan_bk = ttarg:-dt:0;
25 [r0,v0]=oe_to_rv(x0, t0);
26 % Target State
27 %atarg = 1.357;
28 atarg = 7345/RE;
29 etarg = .7370;
30 itarg = 22*pi/180;
31 Omegatarg = 22*pi/180;
32 wtarg = 22*pi/180;
33 Mtarg = 22*pi/180;
34 xT = [atarg; etarg; itarg; Omegatarg; wtarg; Mtarg];
```

The script below declares the state matrix A, input matrices B and u. Input B uses the

function findGM to convert the orbit elements using Kelper's laws.

```
icl=x0-xT;
Kf = 100*eye(6);
Q = .1*eye(6); Q(6,6)=0;
R = 1*eye(14);
%LF Model Parameters
A = zeros(6,6);
B = find_G_M(a0,e0,i0,w0);
u = zeros(length(tspan),14);
```

The script below calculates the Riccati equation using the ode45 function in backwards time, and then the P matrix is flipped into forward time. Within the for loop the linear quadratic trajectory model is calculated in ode45 using the ASE2 function and the parameters are the time span and the initial state. The function returns the position and velocity in Cartesian coordinates. The target state is converted from orbital elements to Cartesian coordinates.

```
%1. Optimize LF Model
y0l=[icl; 0];
[~,Pbig]=ode45(@findP,tspan_bk,Kf(:)); %integrate from tf to t0
Pbig=flipud(Pbig); %flip to forward time
```

```
for n=1:1
    [~,Yl]=ode45('ASE2',tspan,y0l);
    Jl=Yl(end,end);
    al = Yl(:,1)+atarg;
    el = Yl(:,2)+etarg;
    il = Yl(:,3)+itarg;
    Omegal = Yl(:,4)+Omegatarg;
    wl = Yl(:,5)+wtarg;
    Ml=zeros(length(tspan),1);
    rl = zeros(length(al),3);
    vl = zeros(length(al),3);
    for j = 1:length(tspan)
        nt=sqrt(mu/al(j)^3);
        Ml(j) = Yl(j,6)+Mtarg+nt*tspan(j); %note fake-out here
        Ml(j) = rem(Ml(j),2*pi);
        xl = [al(j) el(j) il(j) Omegal(j) wl(j) Ml(j)];
        [rl(j,1:3),vl(j,1:3)]=oe_to_rv(xl,tspan(j));
    end
```

The script below calculates the direction vectors of the thrust-acceleration vectors. The direction vectors and then summed and the resultant direction vector is calculated. The resultant direction is then used as the input for direction vector transformation.

```
    %calculate the directioqun in terms of the x,y,z
    rhat = zeros(length(tspan),3);
    what = zeros(length(tspan),3);
    shat = zeros(length(tspan),3);
    that = zeros(length(tspan),3);
    for k = 1:length(rl)
        rcv = cross(rl(k,:),vl(k,:));
        rhat(k,:) = rl(k,:)/norm(rl(k,:));
        what(k,:) = rcv/norm(rcv);
        shat(k,:) = cross(what(k,:),rhat(k,:));
        fthat = rhat(k,:) + what(k,:) + shat(k,:);
        that(k,:) = fthat/norm(fthat);
    end
%Rotate the direction vector to align with the inertial frame to determine the angle
quadDir = dirVecTrans(that)
```

The script below calculates the magnitude of the Fourier thrust-acceleration vectors. The true anomaly is calculated using the Kepler equation in a function called myKepler that takes the inputs of the trajectory target state. Then the true anomaly and the input control law u, is used to calculate the thrust acceleration components and their resultant vector F_r, F_w, F_S, F_T.

```
% alpha = [a0R a1R a2R b1R a0S a1S a2S b1S b2S a0W a1W a2W b1W b2W]'; %RSW
E=zeros(length(tspan),1);
FR=zeros(length(tspan),1);
FS=zeros(length(tspan),1);
FW=zeros(length(tspan),1);
FT=zeros(length(tspan),1);
for j=1:length(tspan)
    Pvec = Pbig(j,:);
    P=zeros(6,6);
    P(:)=Pvec;
    u(j,:) = (-inv(R)*B'*P*(Yl(j,1:6)'))';
    E(j) = my_keplerE([al(j); el(j); il(j); Omegal(j); wl(j); Ml(j)], tspan(j), mu);
    FR(j)=u(j,1)+u(j,2)*cos(E(j))+u(j,3)*cos(2*E(j))+u(j,4)*sin(E(j));
    FS(j)=u(j,5)+u(j,6)*cos(E(j))+u(j,7)*cos(2*E(j))+u(j,8)*sin(E(j))+u(j,9)*sin(2*E(j));
    FW(j)=u(j,10)+u(j,11)*cos(E(j))+u(j,12)*cos(2*E(j))+u(j,13)*sin(E(j))+u(j,14)*sin(2*E(
    FT(j)=sqrt(FR(j)^2+FS(j)^2+FW(j)^2); % total force
end

%Attitude LQ control
[attitude] = AttMain(quadDir);
```

The LQ attitude and the LQ trajectory are computed. The LQ trajectory control is then compared to the non-linear model of the trajectory. The script below begins by setting the initial conditions for the non-linear model and then using ode45. The non-linear model is

computed using the function NewtEOM2, setting the time interval as the trajectory time span, and the initial conditions and the initial state in Cartesian coordinates. The non-linear model outputs the position and velocity of the trajectory over the entire time span and then is converted into orbital elements using the rv_to_Oe function.

```
% Run Trajectory using Newtonian Equations of Motion
y0Newt=[r0 v0 0];
options = odeset('RelTol',1e-8);
[~,YNewt]=ode45('Newt_EOM2',tspan,y0Newt,options);
J=YNewt(end,end);
aNewt = zeros(length(tspan),1);
eNewt = zeros(length(tspan),1);
iNewt = zeros(length(tspan),1);
OmegaNewt = zeros(length(tspan),1);
wNewt = zeros(length(tspan),1);
thetaNewt = zeros(length(tspan),1);
ENewt = zeros(length(tspan),1);
MNewt = zeros(length(tspan),1);
for j = 1:length(tspan)
    [aNewt(j),eNewt(j),iNewt(j),OmegaNewt(j),wNewt(j),thetaNewt(j),ENewt(j),MNewt(j)] = rv
end
```

The outputs for the LQ model and the non-linear model in orbital elements are placed in subplots. Also the thrust-acceleration is converted to milli-Newtons and displayed in a plot, as well as a 3-d plot showing the trajectory of the spacecraft using the Linear Quadratic model and the non-linear model.

```
% Plot initial trajectories
figure(2);
    subplot(3,2,1); plot(tspan,al*RE); ylabel('a'); hold all;
    plot(tspan,aNewt*RE);
    plot(ttarg,atarg*RE,'r*');
    subplot(3,2,2); plot(tspan,el); ylabel('e'); hold all;
    plot(tspan,eNewt);
    plot(ttarg,etarg,'r*');
    subplot(3,2,3); plot(tspan,il); ylabel('i'); hold all;
    plot(tspan,iNewt);
    plot(ttarg,itarg,'r*');
    subplot(3,2,4); plot(tspan,Omegal); ylabel('Omega'); hold all;
    plot(tspan,OmegaNewt);
    plot(ttarg,Omegatarg,'r*');
    subplot(3,2,5); plot(tspan,wl); ylabel('w'); hold all;
    plot(tspan,wNewt);
```

```
    plot(ttarg,wtarg,'r*');
    subplot(3,2,6); plot(tspan,Ml); ylabel('M'); hold all;
    plot(tspan,MNewt);
    plot(ttarg,Mtarg,'r*');


figure(1);
    plot3(rl(:,1),rl(:,2),rl(:,3)); hold all;
    plot3(YNewt(:,1),YNewt(:,2),YNewt(:,3));
    legend('Linear Model','Nonlinear Model');

figure(2);
    legend('Linear Model','Nonlinear Model','Target');
```

The script below shows the output for the thrust-acceleration.

```
FRmN=FR*RE*1000/60^4*1000; % Convert to mN
FSmN=FS*RE*1000/60^4*1000;
FWmN=FW*RE*1000/60^4*1000;
FTmN=FT*RE*1000/60^4*1000;

figure(3);hold off
    plot(tspan,FRmN,tspan,FSmN,tspan,FWmN,tspan,FTmN);
    xlabel('Time (h)'); ylabel('Force (mN)'); hold all
```

```
9       legend('F_R','F_S','F_W','F_{total}');
```

This part of the scripts details how the output data from the attitude controller is parsed and place into plots accordingly

```
1       %atttiude time(s),w1,w2,w3,q1,q2,q3,J
2   attTime = attitude(:,1)/60^2;
3   q1Att = attitude(:,5); q2Att = attitude(:,6); q3Att = attitude(:,7);
4   u1Att = attitude(:,9)*1000; u2Att = attitude(:,10)*1000; u3Att = attitude(:,11)*1000;
5
6   figure(4)
7       plot(attTime,q1Att,attTime,q2Att,attTime,q3Att)
8       legend('q_1','q_2','q_3')
9       ylabel('quaternion'), xlabel('Time(hr)')
10      xlim([0 32])
11      title('Quadternion Attitude')
12
13  figure(5)
14      plot(attTime,u1Att,attTime,u2Att,attTime,u3Att)
15      legend('u_1','u_2','u_3')
16      ylabel('Torque(mN-m)'), xlabel('Time(hr)')
17      xlim([0 32])
18      title('Input Torque')
```

Attitude Controller Function

The following functions do not come packaged with Matlab and will have to be discussed. The first file that will be discussed is the attitude controller. This function has some similarities to the main script because they are both using similar methods to generate an output.

The function takes an input called bodyorientation, this input is a matrix of quaternions of the direction transformation vector. The first values of this matrix is the initial state of the first iteration of the attitude controller. The first target state is the second row of the body-orientation matrix. The inertia matrix is created and an initial angular velocity is initialized.

```
1
2  function [ output ] = AttMain(bodyorientation)
3  global tspan AAt BAt QAt RAt PbigAt tspanAt xTAt
4  targq1 = bodyorientation(:,1);    %roll
5  targq2 = bodyorientation(:,2);    %pitch
6  targq3 = bodyorientation(:,3);    %yaw
7
8  Jxx= 25.0; Jyy= 25.0; Jzz= 5;
9  wx= 0; wy= 0; wz= 0;
10
```

```
%initial state inputs
J = [Jxx 0 0; 0 Jyy 0; 0 0 Jzz];  %inertia matrix
omega0= [wx ; wy ; wz]*(pi/180);

%initial State
[inistate] = [0;0;0;targq1(1);targq2(1);targq3(1)];
```

At this point the initial parameters for the trajectory are initialized. The next step is to set the parameter for the LQ controllers. The weight factors Q, K, and R are initialized, as well as the state and input vectors.

```
%Target State
QAt = .001*eye(6);
Kf = eye(6);
RAt = 1*eye(3);
%Mode  Parameters
AAt = [zeros(3,3), zeros(3,3); .5*eye(3), zeros(3,3)];
BAt = [inv(J) ; zeros(3,3)];
```

The section of script below initializes the time span, initial, and target states of the attitude controller. The initial time and final time used here break up the time span that is used in the LQ trajectory controller into a smaller interval. This interval will then be evaluated by the attitude controller

```
%target inputs
for (i = 1 : length(tspan))
    %time
    ttargAt = tspan(i+1);   % target time (seconds)
    dt = (ttargAt-tspan(i))/1000;
    tspanAt = (tspan(i):dt:ttargAt)*3600;
    tspan_bkAt = (ttargAt:-dt:tspan(i))*3600;
    omega_target = [0;0;0].*(pi/180);
    euler_target = [targR(i+1),targS(i+1),targW(i+1)];
    [targstate] = [euler_target(1);euler_target(2);euler_target(3)];
    x0 = [inistate];
    xTAt = [omega_target ; targstate];
    icl = x0-xTAt; %initial conditions
```

The following script calculates the Riccati equation for the given set of the initial and target states for the time span. The ode45 function is called on state which is the state-space model of the attitude controller.

When the LQ controller finishes, the torque input is then calculated. When all of the calculations are completed the new initial state of the next time span is set to the last values of the LQ controller output

```
%optimize LF model
    [~,PbigAt] = ode45('findPAt',tspan_bkAt,Kf(:));
```

```
    PbigAt = flipud(PbigAt);
%Run the LQR model
    %y0ini = [omega0;1;1;1]; %[w1;w2;w3;q1;q2;q3]
    [~,Y1] = ode45('state',tspanAt,[inistate ; 0]);
%calculate the input
    for (j = 1 : length(tspanAt))
        Pvec = PbigAt(j,:);
        P=zeros(6,6);
        P(:)=Pvec;
        u(j,:) = (-inv(RAt)*BAt'*P*(Y1(j,1:6)'-xTAt))';
    end
```

Now the data from the Linear quadratic controller will be concatinated and stroed for returning back to the main function

```
 %Save the input torque and attitude to return to main
    if (i == 1)
        Yold = Y1;
        uold = u;
        tspanAtOld = tspanAt;
    else
        Yfinal = vertcat(Yold,Y1);
        Yold = Yfinal;
```

```
        ufinal = vertcat(uold,u);
        uold = ufinal;
        tspanAtFinal = [tspanAtOld,tspanAt];
        tspanAtOld = tspanAtFinal;
    end
```

Now the for loop will repeat itself and sets up the next initial values for the linuear controller. When the for loop finishes, the linear quadratic output is returned to the main function.

```
    %set initial state for next iteration
    %w1,w2,w3,q1,q2,q3
    [inistate] = [Y1(end,1),Y1(end,2),Y1(end,3),Y1(end,4),Y1(end,5),Y1(end,6)]';

end

output = [tspanAtFinal' Yfinal ufinal];

end
```

The calculation for the LQ attitude controller continues until it reaches the end of the time span of the trajectory controller. At the end of each set of attitude time spans the LQ controller, and to torque input are outputted into plots.

Newton's Equations of Motion

This function is used to calculate the trajectory using Newton's Equations of Motion for circular orbits.

```
function dy = Newt_EOM2(t,y)

global mu u tspan

rx=y(1);
ry=y(2);
rz=y(3);
vx=y(4);
vy=y(5);
vz=y(6);
% F=y(7);   %integrated thrust acceleration

% k = length(FC_Ra);

r=[rx ry rz];
v=[vx vy vz];
rmag=norm(r);
rhat=1/rmag*r;
h=cross(r,v);
```

```
hmag=norm(h);
hhat=1/hmag*h;
hrhat=cross(hhat,rhat);
evec=1/mu*cross(v,h)-rhat;
e=norm(evec);
if ((1-e)/(1+e))<0
    t
    e
end
theta=atan2(dot(hhat,cross(evec,r)),dot(evec,r));
E=2*atan2(sqrt(1-e)*tan(theta/2),sqrt(1+e));

if E<0
    E=E+2*pi;
end


alpha=interp1(tspan,u,t,'linear');
R=alpha(1)+alpha(2)*cos(E)+alpha(3)*cos(2*E)+alpha(4)*sin(E);
S=alpha(5)+alpha(6)*cos(E)+alpha(7)*cos(2*E)+alpha(8)*sin(E)+alpha(9)*sin(2*E);
W=alpha(10)+alpha(11)*cos(E)+alpha(12)*cos(2*E)+alpha(13)*sin(E)+alpha(14)*sin(2*E);

% R=interp1(tspan,FR,t,'linear');
% S=interp1(tspan,FS,t,'linear');
```

```
% W=interp1(tspan,FW,t,'linear');


thrust=R*rhat+S*hrhat+W*hhat;

c=-mu/(rmag)^3;

dy=zeros(7,1);

dy(1)=vx;
dy(2)=vy;
dy(3)=vz;
dy(4)=c*rx+thrust(1);
dy(5)=c*ry+thrust(2);
dy(6)=c*rz+thrust(3);

dy(7)=sqrt(R^2+W^2+S^2);
```

Linear Quadratic Function

This function is used to calculate the linear quadratic trajectory for a spacecraft. The state-space model uses the input control law u.

```

```

```
%LF Model

function dxl=ASE2(t,y)

global A B Q R Pbig tspan xT mu

x = y(1:6)+xT;
% J = y(7);

Pvec = interp1(tspan,Pbig,t);

P=zeros(6,6);
P(:)=Pvec;

nt=sqrt(mu/x(1)^3);
F=[zeros(5,1); nt];

u_t = -inv(R)*B'*P*y(1:6);

dx = A*x + B *u_t;
dJ = y(1:6)'*Q*y(1:6) + u_t'*R*u_t;

dxl = [dx; dJ];
```

Gaussian Matrix

This function calculates the input matrix for the linear quadratic state-space model. The Gaussian function consists of fourteen low-thrust Fourier coefficients.

```
%calculates G such that xdot = G*alpha + F, where the 6th element of x is M
function G=find_G_M(a,e,i,w)
global mu

G=zeros(6,14);

G(1,4)=sqrt(a^3/mu)*e; %b1R
G(1,5)=2*sqrt(a^3/mu)*sqrt(1-e^2); %a0S

%e
G(2,4)=.5*sqrt(1-e^2); %b1R
G(2,5)=-1.5*e; %a0S
G(2,6)=1; %a1S
G(2,7)=-.25*e; %a2S
G(2,:)=G(2,:)*sqrt(a/mu)*sqrt(1-e^2);

%i
G(3,10)=-1.5*e*cos(w); %a0W
G(3,11)=.5*(1+e^2)*cos(w); %a1W
```

```
20 G(3,12)=-.25*e*cos(w); %a2W
21 G(3,13)=-.5*sqrt(1-e^2)*sin(w); %b1W
22 G(3,14)=.25*e*sqrt(1-e^2)*sin(w); %b2W
23 G(3,:)=G(3,:)*sqrt(a/mu)/sqrt(1-e^2);
24
25 %Omega
26 G(4,10)=-1.5*e*sin(w); %a0W
27 G(4,11)=.5*(1+e^2)*sin(w); %a1W
28 G(4,12)=-.25*e*sin(w); %a2W
29 G(4,13)=.5*sqrt(1-e^2)*cos(w); %b1W
30 G(4,14)=-.25*e*sqrt(1-e^2)*cos(w); %b2W
31 G(4,:)=G(4,:)*sqrt(a/mu)*csc(i)/sqrt(1-e^2);
32
33 %w
34 G(5,1)=e*sqrt(1-e^2); %a0R
35 G(5,2)=-.5*sqrt(1-e^2); %a1R
36 G(5,8)=.5*(2-e^2); %b1S
37 G(5,9)=-.25*e; %b2S
38 G(5,:)=G(5,:)*sqrt(a/mu)/e;
39 G(5,:)=G(5,:)-cos(i)*G(4,:);
40 %M
41 G(6,1)=-2-e^2; %a0R
42 G(6,2)=2*e; %a1R
43 G(6,3)=-.5*e^2; %a2R
```

```
44 G(6,:)=G(6,:)*sqrt(a/mu);
45 G(6,:)=G(6,:)+(1-sqrt(1-e^2))*(G(5,:)+G(4,:))+2*sqrt(1-e^2)*(sin(i/2))^2*G(4,:)-(G(5,:)+G(
```

Ricatti Equation

The function inputs are time and an empty vector. The Riccati equation is then computed using the ode45 function and the output is populated.

```
1  function Pvecdot = findP(t,Pvec)
2
3  global A B Q R
4
5  P=zeros(6,6);
6  P(:)=Pvec;
7
8  Pdot = -(A'*P+P*A-P*B*inv(R)*B'*P+Q);
9
10 Pvecdot=Pdot(:);
```

Kepler's Equations

This function takes the inputs of orbital elements, time, and gravitational constant and outputs the true anomaly.

```
function nu = my_kepler(oe, t, mu)
a=oe(1);
e=oe(2);
i=oe(3);
Omega=oe(4);
w=oe(5);
M=oe(6);
% Tau=t-M/sqrt(mu/a^3);
dx=0;
k=.85;
del=1e-14;
Mstar=M-floor(M/(2*pi))*2*pi;
if abs(sin(Mstar))>1e-10 %check that nu~=0
   sigma=sin(Mstar)/abs(sin(Mstar)); %sgn(sin(Mstar))
    x=Mstar+sigma*k*e;
    for count=1:10
        es=e*sin(x);
        f=x-es-Mstar;
        if abs(f)<del
            E=x;
            nu=2*atan2(sqrt((1+e)/(1-e))*tan(E/2),1);
            break
        else
            ec=e*cos(x);
```

```
            fp=1-ec;
            fpp=es;
            fppp=ec;
            dx=-f/(fp+dx*fpp/2+dx^2*fppp/6);
            x=x+dx;
        end
    end
    if count==10 %check that Newton's method converges
       nu='undefined';
    end
else
    nu=0;
    E=0;
end
```

Orbital Elements to Cartesian

This function transforms keplarian orbital elements into Cartesian coordinates.

```
function [r,v] = oe_to_rv(oe, t)
global mu
a=oe(1);
e=oe(2);
i=oe(3);
```

```
Omega=oe(4);
w=oe(5);
M=oe(6);
%Determine orbit type
if a<0 || e<0 || e>1 || abs(i)>2*pi || abs(Omega)>2*pi || abs(w)>2*pi %problem
    a
    e
    i
    Omega
    w
   error('Invalid orbital element(s)');
end

xhat=[1 0 0];
yhat=[0 1 0];
zhat=[0 0 1];
nu=my_kepler(oe, t, mu);
nhat=cos(Omega)*xhat+sin(Omega)*yhat;
% hhat=sin(i)*sin(Omega)*xhat-sin(i)*cos(Omega)*yhat+cos(i)*zhat;
rhatT=-cos(i)*sin(Omega)*xhat+cos(i)*cos(Omega)*yhat+sin(i)*zhat;
rmag=a*(1-e^2)/(1+e*cos(nu));
vmag=sqrt(mu/rmag*(2-rmag/a));
gamma=atan2(e*sin(nu),1+e*cos(nu));
u=w+nu;
```

```
30 rhat=cos(u)*nhat+sin(u)*rhatT;
31 vhat=sin(gamma-u)*nhat+cos(gamma-u)*rhatT;
32 r=rmag*rhat;
33 v=vmag*vhat;
```

Cartesian to Orbital Elements

The following function converts the given set of Cartesian coordinates into orbital elements.

```
1  function [a, e, i, Omega, w, theta, E, M] = rv_to_oe(r,v)
2  global mu
3  %Input position and velocity vectors, output orbital elements
4  rmag=norm(r);
5  rhat=1/rmag*r;
6  h=cross(r,v);
7  hmag=norm(h);
8  hhat=1/hmag*h;
9  z=[0 0 1];
10 n=cross(z,h);
11
12 if norm(n)<1e-8; %equatorial orbit
13    evec=1/mu*cross(v,h)-rhat;
14    Energy=.5*dot(v,v)-mu/rmag;
```

```
15      a=-mu/(2*Energy);
16      e=norm(evec);
17      i=0;
18      Omega=0;
19
20      if abs(e)<1e-8
21          w=0;
22      else
23          w=acos(evec(1)/norm(evec));
24      end
25  else
26      nhat=n/norm(n);
27      evec=1/mu*cross(v,h)-rhat;
28      Energy=.5*dot(v,v)-mu/rmag;
29      a=-mu/(2*Energy);
30      e=norm(evec);
31      i=acos(dot(z,hhat));
32      Omega=atan2(nhat(2),nhat(1));
33      if abs(e)<1e-8
34          w=0;
35      else
36          w=atan2(dot(hhat,cross(nhat,evec)),dot(nhat,evec));
37      end
38  end
```

```
39
40 if w<0
41     w=w+2*pi; %define w on (0,2*pi)
42 end
43 if Omega<0
44     Omega=Omega+2*pi; %define Omega on (0,2*pi)
45 end
46
47 theta=atan2(dot(hhat,cross(evec,r)),dot(evec,r));
48 E=2*atan2(sqrt(1-e)*tan(theta/2),sqrt(1+e));
49 if E<0
50     E=2*pi+E;
51 end
52 M=E-e*sin(E);
53 if M<0
54     M=M+2*pi; %define M on (0,2*pi)
55 end
```

Direction Transformation

The input of this function is the resultant direction vector of the thrust-acceleration. The rotation matrix R, is calculated from the pointing direction vector a, and resultant direction vector.

```
function quadDir = dirVecTrans(dirVector)

q1=zeros(length(dirVector),1);
q2=zeros(length(dirVector),1);
q3=zeros(length(dirVector),1);
q4=zeros(length(dirVector),1);
a= [0,1,0];  %assume pointing in the correct direction
\end{minted}

for i = 1 : length(dirVector(:,1))
    %Calculate the rotation matrix
    %S is a scaling factor
    R=eye(3)+skew(cross(a,dirVector(i,:)))+skew(cross(a,dirVector(i,:)))^2*...
        ((1-dot(a,dirVector(i,:)))/norm(cross(a,dirVector(i,:))));
    theta=acos((trace(R)-1)/2);
    e=1/(2*sin(theta))*[R(2,3)-R(3,2),R(3,1)-R(1,3),R(1,2)-R(2,1)];

%Convert into quaternions
    q1(i)=e(1)*sin(theta/2);
    q2(i)=e(2)*sin(theta/2);
    q3(i)=e(3)*sin(theta/2);
    q4(i)=cos(theta/2);
end
quadDir = [q1, q2, q3, q4];
```

```
25 end
```

Skew Matrix Function

The skew symmetric matrix function first verifies that the matrix is a 1-by-3 matrix. Then a 3-by-3 zero matrix is created and the matrix is populated with the input. The output is a 3-by-3 matrix.

```
1  %given a 3x1 matrix create a 3x3 skew symmetric
2  function S = skew(a)
3  %check if the size of the matrix is 1x3
4  if size(a) == [1, 3]
5      %write a skew matrix
6      S = zeros(3,3);
7      S(1,2) = -a(3);
8      s(1,3) = a(2);
9      S(2,1) = a(3);
10     S(2,3) = -a(1);
11     S(3,1) = -a(2);
12     S(3,2) = a(1);
13 else
14     error('Matrix must be of size 1 x 3')
15 end
```