The Pennsylvania State University
The Graduate School
College of Engineering

TIME-OPTIMAL, CONSTRAINED, SATELLITE REORIENTATION MANEUVER USING INVERSE DYNAMICS

A Thesis in
Aerospace Engineering
by
Michael Nino

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2019

The thesis of Michael Nino was reviewed and approved* by the following:

Robert G. Melton
Professor of Aerospace Engineering
Thesis Advisor

Puneet Singla
Associate Professor of Aerospace Engineering

Amy R. Pritchett
Professor and Head of Aerospace Engineering

*Signatures are on file in the Graduate School.

Abstract

Being able to quickly reorient certain satellites, such as orbiting astronomical observatories, is critical to mission objectives. If an event of interest occurs, the goal is to measure that event before it disappears. This can be difficult when the path of reorientation between the sensor and the event is constrained. An electromagnetic (EM) sensor is designed to record EM radiation within certain intensity tolerances. Celestial bodies, such as the Earth, Moon, and Sun, can output EM radiation well above those tolerances depending on the objective of the satellite. This requires that the sensor not only be reoriented from its current pointing to the direction of the event, but also avoid the EM output of those bodies in the range that exceeds the tolerances of the sensor. This leads to an optimal control problem that needs to be solved computationally. For a rigid body satellite with three-axis control capabilities, the use of Euler's rotation equations can lead to different attitude parameterizations, such as the use of a quaternion representation or Rodrigues parameters, to solve the system of equations. In this thesis, an inverse-dynamics representation is used, which involves parameterizing the attitude of the satellite into three separate orientation angles constituting the 3-2-1 Euler angle sequence. This is converted to a quaternion formulation to achieve compatibility with on board satellite control systems. Genetic algorithms, also called hybrid or heuristic methods, provide a framework for solving optimal control problems quickly. Particle Swarm Optimization (PSO) is a method that invokes a relationship between a particle (a single solution element) and a swarm (a group of solution elements) to search the solution space for an optimal solution. Combining PSO and inverse-dynamics has been shown in previous analyses to be a contender for a way to produce solutions to the reorientation problem efficiently. The method in this analysis determined to be the best contender for producing real-time solutions to this problem involves modeling the orientation angles of the inverse-dynamics problem using 5^{th} order polynomials that exactly meet the end point conditions of the problem. Using a normalized time unit, a relationship between the control profile and the final time can be used to determine a final time that guarantees the constraints associated with the satellite's design. This allows

PSO to address just the maneuver path that avoids the constraint bodies. This implementation was found to consistently produce results that are both closer to optimality and more computationally efficient than those in the literature.

Table of Contents

List of Figures

List of Symbols

A	matrix of least squares polynomial elements
$\vec{a}$	vector of polynomial coefficients
a	arbitrary weights used for polynomial definitions
BL_p	array of lower bounds for particle elements
BL_v	array of lower bounds for particle element velocities
BU_p	array of upper bounds for particle elements
BU_v	array of upper bounds for particle element velocities
b	arbitrary weights used for polynomial definitions
c	arbitrary weights used for polynomial definitions
c_C	constant associated with best elements of this particle and current particle
c_I	constant associated with previous velocity
c_S	constant associated with best elements ever seen and current particle
$\mathbf{E}$	Quaternion matrix
F	function associated with change of variables from t to τ
G_{best}	best particle
GG	best cost value
GG^*	array of best costs for each PSO iteration

I	Principal Moment of Inertia
J	array of all current costs
J_{best}	array of best cost seen by a particle
J_k	cost of current particle in current iteration
J_M	cost function associated with torques
J_{t_f}	cost function associated with final time
J_{UC}	cost function associated with unconstrained problem
J_ϵ	cost function associated with quaternion angles
J_ω	cost function associated with angular momenta
k	integer used for PSO element outline
l	integer used for PSO element outline
m	integer used for PSO element outline
M	Torque
max	maximum value associated with this element
min	minimum value associated with this element
N_e	number of elements per particle
N_i	number of iterations to run
N_p	number of particles per iteration
n_{coe}	order of the polynomial
n_{sec}	number of sections
P	array of all current particles
P_{best}	array of best elements seen by a particle
P_k	current particle
$rand$	random number between 0 and 1 using a Gaussian distribution of

T	Chebyshev Polynomials
t	time (in TU)
t_f	final time (in TU)
u	unit step function
V	array of all current particle velocities
V_k	velocity of current particle elements in current iteration
$\vec{y}$	vector of function values
α	Orientation angle
α_{num}	order of the α polynomial
β	Orientation angle
β_{num}	order of the β polynomial
ϵ	Quaternion element
λ	Component for quaternion calculation
ω	Angular Momentum
σ	Inertial frame orientation
τ	time normalization variable (differs from TU normalization)
θ	Orientation angle
θ_{num}	order of the θ polynomial
$\angle\sigma(t)\gamma_i$	angle between sensor axis and constraint body central axis

Acknowledgments

This research would not have been possible without the guidance of my thesis advisor, Dr. Robert G. Melton. I was provided with just enough assistance to not lose all momentum with my progress, but not so much that I avoided every potential roadblock and shortcoming that a research endeavor entails. He was there to slow me down when I was overzealous with results before I confirmed their validity, and to make sure I didn't enter rabbit holes leading to dead-ends.

Additionally, I would like to thank Dr. Puneet Singla for his indirect contribution to my thesis work. The method I used to produce my final results were formalized during one of his lectures on collocation, which has a very similar structure to the polynomial fitting used in this research.

I would also like to thank those at Penn State that may not have directly contributed to my research efforts, but were contributing members to my graduate experience here. I do not have sufficient room in this acknowledgment section to elaborate on everyone individually, but I will say a special thank you to Damien Guého. All of the lunches and the friendly banter made time fly by, perhaps too quickly, but it made my time here enjoyable.

And to those that were not here at Penn State during my attendance, but provided mental and emotional support from afar. Family and friends whom I talked to often, who spoke with me during long walks to and from class, who spent late nights and weekends keeping me company, thank you!

Last but not least, thank you Joe.

Chapter 1
Introduction

1.1 Problem Outline

1.1.1 Problem Significance

In observational astronomy it is critical that, upon detection of interesting energy signatures, a sensor is able to be reoriented toward the location of origin and take measurements. Relevant to this topic, the Neil Gehrels Swift Observatory satellite can be taken as an example in which this type of attitude adjustment is important, and an example that can seek its origin in astrophysical research at the Pennsylvania State University [1]. While the dynamics of the attitude control of a satellite are known well, both physical and path constraints on the system make it difficult to find control profiles that are optimal with respect to time, propellant, or both. While there can be much difficulty in finding an optimal solution to such a problem, it is still critical to obtain a control profile fast. Exotic cosmic phenomena can last for very short periods of time, from an average of 0.3 seconds for short duration phenomena to an average of 30 seconds for long-duration phenomena [2].Thus, attempting to find near-optimal solutions to this problem in real-time is much more useful than finding real optimal solutions using an expensive computational load.

Sensors on a satellite such as the Swift satellite can be very sensitive to the intensity of electromagnetic radiation [3]. They are built to detect low-intensity, high-frequency radiation that is produced by cosmic events such as supernovae, colliding black holes, or even events whose origins are not yet known. Due to this sensitive nature, a constant flux of high-intensity EM radiation can reduce

sensitivity, disturb calibrations, or even destroy a sensor altogether. Examples of such sources of EM radiation are: the Sun, Earth, and Moon. The effect this produces on the attitude control problem are essentially a path constraint. The flux of EM radiation produced by these sources create "keep-out" cones within which the sensor would see degradation if pointed in these directions. Additionally, the body of the satellite has both torque and rotation limitations depending on its structural design. This provides additional constraints that must be satisfied in order to produce a feasible solution. Lastly, longevity of the satellite must also be taken into consideration; thus, while a specific path may be more time-efficient, it may be necessary to impose a propellant constraint as well.

The approach in this analysis is to assume a rigid-body satellite with three-axis control of its rotational motion. Using these assumptions, a hybrid method known as Particle Swarm Optimization (PSO) is implemented to solve this optimal control problem. An inverse-dynamics approach to the kinematics is explored to reduce computational load, and many different implementations of the kinematics are discussed. Both an unconstrained and constrained case are investigated and compared to similar research.

1.1.2 Previous Research

One of the first papers to investigate the time-optimal, three-axis reorientation of a rigid spacecraft using a mathematically rigorous optimal-control approach was a publication 1993 by Karl D. Bilimoria and Bong Wie [5]. They investigated the unconstrained case to the problem presented above. To solve this problem, they used Euler's rotation equations for a rigid body spacecraft in conjunction with a Hamiltonian formulation for optimal time and bounded control inputs. The purpose of this paper was to solve, exactly, the time required to complete reorientation maneuvers of different angular distances, as well as to show what this path looked like, and how the maneuver time differs between the optimal and eigenaxis solutions. They were the first to show that the optimal maneuver path involves some precession out of plane. Additionally, their results concluded that there can be a significant difference between the optimal reorientation time and the eigenaxis variant - a difference of up to 8.5% [5]. A paper by John L. Junkins and James D. Turner in 1980 investigated the special cases of single-axis, continuous

torque, attitude maneuvers for large angle reorientations using a similar formulation with analytic representations, and those can be found in [4].

In 2013, Robert G. Melton published a paper than sought to solve the constrained version of this problem using hybrid methods, such as PSO, Differential Evolution, and Bacteria Foraging Optimization [15]. This approach involved modeling the control torque's of the spacecraft as Chebyshev polynomials to solve Euler's rotational equations directly. The objective of the hybrid methods were to solve for the coefficients of the control polynomials associated with a time-efficient solution that also met constraint conditions. The results of the hybrid methods were then used as initial conditions for a commercial psuedospectral optimizer, DIDO, to determine whether a combination of methods could produce results quicker than a psuedospectral method alone. This was the first study to attempt to use these hybrid methods to solve the constrained reorientation problem.

Then, in 2016, Ko Basu and Robert G. Melton attacked this same problem using the inverse-dynamics formulation. Rather than model the control torques to find a time-optimal solution, they modeled the reorientation path instead. The significant benefit of this approach is that Euler's rotational equations didn't need to be integrated - instead, the reorientation path history, represented by a quaternion, and their derivatives, can be used to directly solve for the control profile, significantly saving on computational effort. In their analysis, they modeled the elements of the reorientation path using basis spline (B-spline) polynomials. Using this approach, feasible solutions were produced, and these will be compared to the results in this thesis.

The last paper analyzed was by Dario Spiller, Luigi Ansalone, and Fabio Curti, published in 2016 [8]. They also approached the problem using an inverse-dynamics method; however, instead of using Euler parameters (quaternion elements), they used modified Rodrigues parameters to describe the attitude control. Their solutions are associated with a satellite whose three principal moments of inertia are not equal, and also with unnormalized units. The conclusion their paper presents is that inverse-dynamics, in conjunction with PSO, can produce near time-optimal solutions.

The research presented in this thesis attempts to expand on the work done in these papers to provide a method for computing near time-optimal solutions to the reorientation such that computational efficiency is also taken into account.

Chapter 2
Attitude Dynamics and Particle Swarm Optimization

2.1 Direct Attitude Dynamics

Assuming the satellite being maneuvered can be approximated as a rigid body, the rotational dynamics of the orientation of the satellite can be described using Euler's rotation equations, further detailed in [6].

$$\begin{aligned} \dot{\omega}_1 &= \left[M_1 - \omega_2\omega_3 \left(I_3 - I_2\right)\right] / I_1 \\ \dot{\omega}_2 &= \left[M_2 - \omega_3\omega_1 \left(I_1 - I_3\right)\right] / I_2 \\ \dot{\omega}_3 &= \left[M_3 - \omega_1\omega_2 \left(I_2 - I_1\right)\right] / I_3 \end{aligned} \tag{2.1}$$

The method for determining attitude control using the direct attitude dynamics would be to take the control inputs, in this case the torques M_i, as having a known time-history representation, and using these in conjunction with the initial conditions for the angular velocities and accelerations, ω_i, $\dot{\omega}_i$ and the principal moments of inertia, I_i, to solve for the time history of the angular velocities. This is not to be confused with a direct/indirect optimization method, as was used by [5]. While those can be a method used to solve this problem, the direct/inverse attitude dynamics and the direct/indirect optimization methods are different. In this thesis, direct refers to direct attitude dynamics, and the inverse method involves inverse dynamics.

With a known time-history for the angular velocities, and a known initial orientation, a quaternion representation of the problem can be used, and then

the following set of differential equations can be solved for the time-history of the orientation of the satellite.

$$\begin{bmatrix} \dot{\epsilon}_1 \\ \dot{\epsilon}_2 \\ \dot{\epsilon}_3 \\ \dot{\epsilon}_4 \end{bmatrix} = \begin{bmatrix} \epsilon_4 & -\epsilon_3 & \epsilon_2 & \epsilon_1 \\ \epsilon_3 & \epsilon_4 & -\epsilon_1 & \epsilon_2 \\ -\epsilon_2 & \epsilon_1 & \epsilon_4 & \epsilon_3 \\ -\epsilon_1 & -\epsilon_2 & -\epsilon_3 & \epsilon_4 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} \tag{2.2}$$

where the ϵ_i terms are the four elements of the quaternion vector used for the parameterization of the attitude of the satellite.

Determining a control profile that will produce a time-optimal solution to the unconstrained variant of this problem is a straightforward optimal-control problem, but the requirement of numerical integration of the governing equations, as well as the solution of a two-point boundary value problem, makes it computationally taxing. Adding onto this the path constraints, it becomes very difficult to find an optimal solution at all, and a near-optimal solution cannot be found in real-time using this method. More on the direct method can be found in [7].

2.2 Inverse Attitude Dynamics

Rather than taking the torque control profiles as the known quantities in the formulation and calculating the orientation time-history, the orientation time-histories can be taken to be known instead, and the control profile required to produce that time-history can be calculated. This approach to the dynamics is referred to as inverse-dynamics. Euler's rotation equations will be written as Eq. (2.3) for this formulation as the torques will be determined rather than the orientation. This inverse-dynamics formulation can be found in [8].

$$\begin{aligned} M_1 &= I_1\dot{\omega}_1 + \omega_2\omega_3\left(I_3 - I_2\right) \\ M_2 &= I_2\dot{\omega}_2 + \omega_3\omega_1\left(I_1 - I_3\right) \\ M_3 &= I_3\dot{\omega}_3 + \omega_1\omega_2\left(I_2 - I_1\right) \end{aligned} \tag{2.3}$$

Given that the orientation of the satellite is written in terms of a quaternion, the following relationship must be applied:

$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \epsilon_4^2 = 1 \tag{2.4}$$

If a guess at the time-history of each individual quaternion is the approach to be taken, it would be difficult to produce simple functions that guarantee the relationship in Eq. (2.4). Instead, an alternative route is taken that guarantees this constraint, while simplifying the necessary apriori information to three angles rather than four independent coordinate values. The elements of the quaternion can be rewritten in terms of three angles, α, β, and θ, as seen in Fig. 2.1.

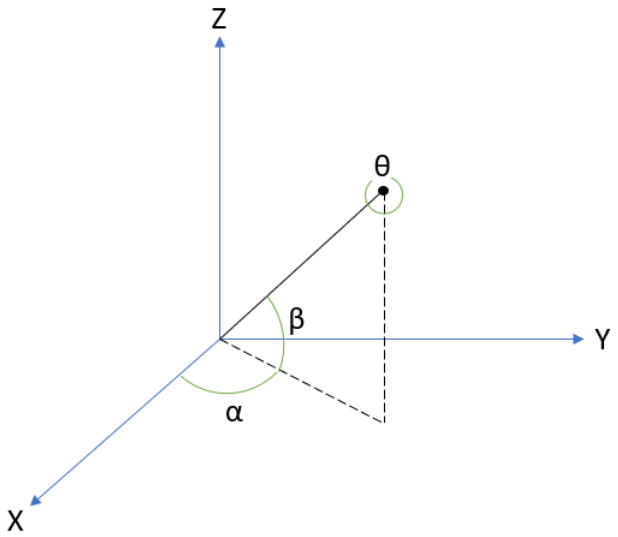

Figure 2.1. The three orientation angles related to the satellite.

The four quaternion elements in this formulation are

$$\begin{aligned} e_i &= \lambda_i sin(\tfrac{\theta}{2}),\ i \in (1,2,3) \\ e_4 &= cos(\tfrac{\theta}{2}) \end{aligned} \tag{2.5}$$

where the λ_i are given by

$$\begin{aligned} \lambda_1 &= \cos(\beta)\cos(\alpha) \\ \lambda_2 &= \cos(\beta)\sin(\alpha) \\ \lambda_3 &= \sin(\beta) \end{aligned} \tag{2.6}$$

These λ_i components correspond to the principal axis of rotation, also known as the eigenaxis, and the angle θ is the principle angle of rotation. Together, α, β, and θ constitute the 3-2-1 rotation Euler angles. This formulation does contain ambiguity at $\beta = \frac{\pi}{2}$, but this will be addressed later on.

This representation guarantees that, regardless of the particular time-histories of α, β, and θ, Eq. (2.4) is satisfied. While using these Euler angles does not directly require the need for quaternions, in real satellite attitude control, quaternion elements are often used to denote orientation, so having the flexibility to move between the two in this thesis can be beneficial for real applications. Additionally, using Euler angles for the direct attitude method creates two potential angle singularities when moving from the quaternion to the Euler angle parameterization, the inverse dynamics eliminates one of these, leaving only the ambiguity mentioned above.

Taking the inverse of the matrix composed of the quaternion elements in Eq. (2.2), the angular velocities of the satellite can be determined using the quaternion elements and their derivatives.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix} = 2 \begin{bmatrix} \epsilon_4 & \epsilon_3 & -\epsilon_2 & -\epsilon_1 \\ -\epsilon_3 & \epsilon_4 & \epsilon_1 & -\epsilon_2 \\ \epsilon_2 & -\epsilon_1 & \epsilon_4 & -\epsilon_3 \\ \epsilon_1 & \epsilon_2 & \epsilon_3 & \epsilon_4 \end{bmatrix} \begin{bmatrix} \dot{\epsilon}_1 \\ \dot{\epsilon}_2 \\ \dot{\epsilon}_3 \\ \dot{\epsilon}_4 \end{bmatrix}$$

This matrix representation can be rewritten as the following for ease of computation:

$$\omega = 2\mathbf{E}\dot{\epsilon} \tag{2.7}$$

where

$$w = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ 0 \end{bmatrix}, \ \mathbf{E} = \begin{bmatrix} \epsilon_4 & \epsilon_3 & -\epsilon_2 & -\epsilon_1 \\ -\epsilon_3 & \epsilon_4 & \epsilon_1 & -\epsilon_2 \\ \epsilon_2 & -\epsilon_1 & \epsilon_4 & -\epsilon_3 \\ \epsilon_1 & \epsilon_2 & \epsilon_3 & \epsilon_4 \end{bmatrix}, \ \dot{\epsilon} = \begin{bmatrix} \dot{\epsilon}_1 \\ \dot{\epsilon}_2 \\ \dot{\epsilon}_3 \\ \dot{\epsilon}_4 \end{bmatrix}$$

To compute the torques in Eq. (2.3), the angular accelerations are also needed. To

get these values, a derivative of Eq. (2.7) can be taken. This results in

$$\dot{\omega} = 2\dot{\mathbf{E}}\dot{\epsilon} + 2\mathbf{E}\ddot{\epsilon} \tag{2.8}$$

where

$$\dot{\mathbf{E}} = \begin{bmatrix} \dot{\epsilon_4} & \dot{\epsilon_3} & -\dot{\epsilon_2} & -\dot{\epsilon_1} \\ -\dot{\epsilon_3} & \dot{\epsilon_4} & \dot{\epsilon_1} & -\dot{\epsilon_2} \\ \dot{\epsilon_2} & -\dot{\epsilon_1} & \dot{\epsilon_4} & -\dot{\epsilon_3} \\ \dot{\epsilon_1} & \dot{\epsilon_2} & \dot{\epsilon_3} & \dot{\epsilon_4} \end{bmatrix}, \ \ddot{\epsilon} = \begin{bmatrix} \ddot{\epsilon_1} \\ \ddot{\epsilon_2} \\ \ddot{\epsilon_3} \\ \ddot{\epsilon_4} \end{bmatrix}$$

To solve all of the corresponding equations seen above, the first and second order derivatives of the quaternion elements are necessary, with respect to time. As there exists a function representation for α, β, and θ, the derivatives of the quaternion elements can be determined using the derivatives of those functions. Stepping through the problem, first, the derivatives of the quaternion elements can be represented as

$$\begin{aligned} \dot{\epsilon}_i &= \dot{\lambda}_i \sin\tfrac{\theta}{2} + \tfrac{\dot{\theta}}{2}\lambda_i \cos\tfrac{\theta}{2}, \ i \in (1,2,3) \\ \dot{\epsilon}_4 &= -\tfrac{\dot{\theta}}{2}\sin\tfrac{\theta}{2} \\ \ddot{\epsilon}_i &= \ddot{\lambda}_i \sin\tfrac{\theta}{2} + 2\tfrac{\dot{\theta}}{2}\dot{\lambda}_i \cos\tfrac{\theta}{2} + \tfrac{\ddot{\theta}}{2}\lambda_i \cos\tfrac{\theta}{2} - \tfrac{\theta^2}{4}\lambda_i \sin\tfrac{\theta}{2}, \ i \in (1,2,3) \\ \ddot{\epsilon}_4 &= -\tfrac{\dot{\theta}}{2}\sin\tfrac{\theta}{2} - \tfrac{\dot{\theta}^2}{4}\cos\tfrac{\theta}{2} \end{aligned} \tag{2.9}$$

Next, the derivatives of the λ's can be represented as

$$\begin{aligned} \dot{\lambda}_1 &= -\dot{\beta}\sin(\beta)\cos(\alpha) - \dot{\alpha}\cos(\beta)\sin(\alpha) \\ \dot{\lambda}_2 &= -\dot{\beta}\sin(\beta)\sin(\alpha) + \dot{\alpha}\cos(\alpha)\cos(\beta) \\ \dot{\lambda}_3 &= \dot{\beta}\cos(\beta) \\ \ddot{\lambda}_1 &= -\ddot{\beta}\sin(\beta)\cos(\alpha) - \dot{\beta}^2\cos(\beta)\cos(\alpha) + 2\dot{\alpha}\dot{\beta}\sin(\beta)\sin(\alpha) \\ &\quad -\dot{\alpha}^2\cos(\beta)\cos(\alpha) - \ddot{\alpha}\sin(\alpha)\cos(\beta) \\ \ddot{\lambda}_2 &= -\ddot{\beta}\sin(\beta)\sin(\alpha) - \dot{\beta}^2\cos(\beta)\sin(\alpha) - 2\dot{\alpha}\dot{\beta}\sin(\beta)\cos(\alpha) \\ &\quad -\dot{\alpha}^2\cos(\beta)\sin(\alpha) + \ddot{\alpha}\cos(\alpha)\cos(\beta) \\ \ddot{\lambda}_3 &= \ddot{\beta}\cos(\beta) - \dot{\beta}^2\sin(\beta) \end{aligned} \tag{2.10}$$

Depending on how α, β, and θ are modeled, their derivatives can be calculated from their function representations.

Lastly, a useful relationship that will be used for viewing the trajectory of the satellite, and accounting for the path constraints that will be discussed in more detail later, is the conversion from quaternion elements back to a unit vector representing the orientation of the sensor relative to the inertial frame. As the sensor on the satellite will always be aligned with the body-fixed 1-axis, the quaternion elements of the sensor-axis can be represented as a unit vector in the inertial frame as follows:

$$\begin{aligned} \sigma_1 &= \epsilon_1^2 + \epsilon_4^2 - \epsilon_3^2 - \epsilon_2^2 \\ \sigma_2 &= 2\left(\epsilon_1\epsilon_2 + \epsilon_3\epsilon_4\right) \\ \sigma_3 &= 2\left(\epsilon_3\epsilon_1 - \epsilon_2\epsilon_4\right) \end{aligned} \tag{2.11}$$

This result comes from the direction cosine matrix that transforms the body-fixed frame to the inertial frame in terms of the quaternion elements.

This concludes the necessary mathematics for the dynamics of the attitude maneuver under a rigid body approximation, using an inverse dynamics approach.

2.3 Particle Swarm Optimization

Particle Swarm Optimization is a metaheuritic optimization algorithm that imposes swarm theory to solve nonlinear optimization problems [9]. This method of solving nonlinear optimization problems is one of many evolutionary genetic methods that attempt to replicate the social behaviors of different ecosystems seen in nature. The algorithm is not difficult to define and implement, and compared to other optimization techniques, it can be less computationally taxing while providing similar results. Use of this algorithm for solving spacecraft trajectory problems was investigated in [10], and some of the methods used were applied to this research.

There are two main entities in the PSO algorithm; a particle, and the swarm. A particle in PSO is an object that holds all of the values necessary to construct a solution for the problem. In addition to this, the particle also contains an iteration history of previous solutions generated for that particle, as well as the relative optimality (or cost value) of those solutions. The swarm is a coalition of all particles that are searching for an optimal solution. The number of particles in a swarm is defined as part of the problem. There is a relationship between a given particle,

its iteration history, and the swarm that provides information on how the particle should move its elements to converge toward a solution closer to the optimality. Figure 1 in [8] is a diagram that represents the relationship very succinctly - this diagram can be seen in Fig. 2.2.

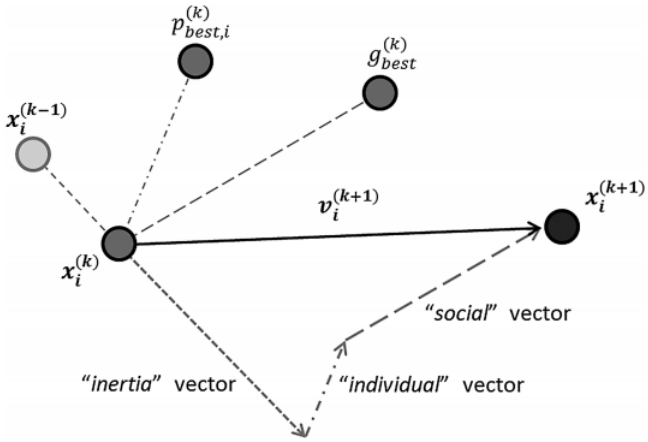

Figure 2.2. Figure 1 from [8] demonstrating the relationship between a particle's updates, itself, and the swarm.

The i subscripts represent the i^{th} element in the swarm, and the superscripts represent the iteration of PSO associated with that element. The x are the particles, v is the velocity update, p represents the iteration history of the particle, and g is the best particle seen by the swarm in all iterations. There are three components to the velocity update that brings the particle from its current iteration to the next. The first of these is the inertia vector. This is not reliant on any previous search information related to the swarm, as will be seen in the algorithm shortly. The individual vector (also known as the cognitive vector) is related to the location of the best iteration of the current particle in the solution space. A portion of the velocity vector will work to move the current iteration of the particle toward the best individual value of it. Lastly, the social vector pushes the current iteration of the particle toward the location of the best particle ever seen by the swarm. Together, the individual particles in conjunction with the swarm of particles search the solution space in search of an optimal solution. Relative to Fig. 2.2, this update looks like the following

$$x_i^{(k+1)} = c_I * x_i^{(k)} + c_C * p_{best,i}^{(k)} + c_S * g_{best}^{(k)} \tag{2.12}$$

where $c_I,\ c_C,\ c_S$ are weighting coefficients related to the significance of each factor to the update.

For ease of formulation, a few of the variables being used will be defined below:

P = array of all current particles;
J = array of all current costs;
V = array of all current particle velocities;
P_k = current particle;
J_k = cost of current particle in current iteration;
V_k = velocity of current particle elements in current iteration;
N_e = number of elements per particle;
N_p = number of particles per iteration;
N_i = number of iterations to run;
$f()$ = cost function;
J_{best} = array of best cost seen by a particle;
P_{best} = array of best elements seen by a particle;
GG = best cost value;
G_{best} = best particle;
GG^* = array of best costs for each PSO iteration;
BL_v = array of lower bounds for particle element velocities;
BU_v = array of upper bounds for particle element velocities;
BL_p = array of lower bounds for particle elements;
BU_p = array of upper bounds for particle elements;
$rand$ = random number between 0 and 1;
c_I = inertial weighting coefficient;
c_C = cognitive weighting coefficient;
c_S = social weighting coefficient;

To instantiate PSO, a few variables must be defined. N_i should be set to the number of iterations that are to be completed, N_p to the number of particles being used, and N_e to the number of elements corresponding to each particle. Then, the elements of each particle much be initialized for the first iteration of PSO - this will yield the initial array of particles, P. The next two arrays are BL_p and BU_p. PSO assumes there is a pretty good understanding of the dynamics of the elements

of the particles with respect to the system, and thus upper and lower bounds can be approximated to limit the search region. The initialization of P should be done with respect to these bounds. BL_v and BU_v are the max distance a particle element could travel up or down. This can be written as $BU_v = BU_p - BL_p$ and $BL_v = -BU_v$. P_{best}, J, and V arrays should all be initialized to 0, and J_{best}, GG, and GG^* should all be initialized to infinity.

There are four main functions that compose the body of PSO. The first of these is the cost function. For every particle in PSO during an iteration, we seek to determine the value of the cost function we are trying to minimize (or maximize if we impose a negative sign). This can be written as the following:

procedure EVALUATECOST(P)
  for k=1:N_p **do**
    $J_k = f(P_k)$
  end for
end procedure

The next function is responsible for keeping track of the best particle elements each particle has had, the cost corresponding to those elements, and whether or not a new overall best set of elements and cost has been seen.

procedure EVALUATEPARTICLEBEST( )
  for k=1:N_p **do**
    if $J_k < J_{best_k}$ **then**
      $P_{best_k} = P_k$
      $J_{best_k} = J_k$
    end if
    if $J_k < GG$ **then**
      $GG = J_k$
      $G_{best} = P_k$
    end if
  end for
end procedure

The third function is the velocity updating function. The sole responsibility of this function is to use the known information from G_{best}, P_{best}, V, and P to update the velocity value for each particle. The origin of the algorithm associated with

this function, and the constant values c_I, c_C, and c_S, can be found in Ref. [7].

procedure UPDATEVELOCITIES( )
  $c_I = (1 + rand)/2$
  $c_C = 1.49445 * rand$
  $c_S = 1.49445 * rand$
  for k=1:N_p **do**
    $V_k = c_I * V_k + c_C * (P_{best_k} - P_k) + c_S * (G_{best} - P_i)$
    for i=1:N_e **do**
      if $V_{k_i} < BL_{v_i}$ **then**
        $V_{k_i} = BL_{v_i}$
      end if
      if $V_{k_i} > BU_{v_i}$ **then**
        $V_{k_i} = BU_{v_i}$
      end if
    end for
  end for
end procedure

The last function necessary for PSO is the particle updating function. This takes into account the updated velocities that were just determined, as well as the bounds for the particle elements, and moves the particle elements to their new values. This can be written as follows:

procedure UPDATEPARTICLES( )
  for k=1:N_p **do**
    $P_k = P_k + V_k$
    for i=1:N_e **do**
      if $P_{k_i} < BL_{p_i}$ **then**
        $P_{k_i} = BL_{p_i}$
        $V_{k_i} = 0$
      end if
      if $P_{k_i} > BU_{p_i}$ **then**
        $P_{k_i} = BU_{p_i}$
        $V_{k_i} = 0$
      end if
    end for

end for

end procedure

Both UpdateVelocities and UpdateParticles take into account the upper and lower bounds for the particle elements and velocities. If a new velocity has been calculated that exceeds the maximum or minimum velocity possible, the new velocity is set to that bound. Similarly, if one of an updated particle's elements hit an upper or lower bound, the element is set to that bound, and the velocity of that element is set to 0.

This leads to the main loop that runs PSO. Once all of the above functions have been implemented, and the initial values of the above variables have been set, the PSO algorithm is:

procedure ParticleSwarmOptimization( )

for k=1:N_i **do**

EvaluateCost()

EvaluateParticleBest()

UpdateVelocities()

UpdateParticles()

$GG_k^* = GG$

end for

end procedure

Chapter 3
Approach Formulations and Algorithms

3.1 Understanding Constraints

The maneuver being considered is a rest-to-rest maneuver in which the initial angles for the three parameterized quaternion components are all 0. This yields the initial conditions for this problem:

$$\alpha(0) = 0 \tag{3.1}$$
$$\beta(0) = 0 \tag{3.2}$$
$$\theta(0) = 0 \tag{3.3}$$
$$\epsilon(0) = [0\ 0\ 0\ 1]^T \tag{3.4}$$
$$\omega(0) = [0\ 0\ 0]^T \tag{3.5}$$

These yield additional initial conditions for other parameters, namely:

$$\lambda_1(0) = 1 \tag{3.6}$$
$$\lambda_2(0) = 0 \tag{3.7}$$
$$\lambda_3(0) = 0 \tag{3.8}$$

However, from Eq. (2.7), this is not enough to satisfy Eq. (3.5). To meet that initial condition, $\dot{\epsilon}$ also must equal zero. This is satisfied automatically for $\dot{\epsilon}_2$, $\dot{\epsilon}_3$,

and $\dot{\epsilon}_4$, based on the formulation represented in Eq. (2.9), but not for $\dot{\epsilon}_1$. Therefore

$$\begin{aligned}\dot{\epsilon}_1 &= \dot{\lambda}_1(0)\sin\frac{\theta(0)}{2} + \frac{\dot{\theta}(0)}{2}\lambda_1(0)\cos\frac{\theta(0)}{2} \\ &= \frac{\dot{\theta}(0)}{2}\end{aligned}$$

The last condition that needs to be imposed is that

$$\dot{\theta}(0) = 0 \tag{3.9}$$

To satisfy the end point conditions, the following will be necessary:

$$\epsilon(t_f) = [\epsilon_{1_f}\ \epsilon_{2_f}\ \epsilon_{3_f}\ \epsilon_{4_f}]^T \tag{3.10}$$

$$\omega(t_f) = [0\ 0\ 0]^T \tag{3.11}$$

While the formulation uses α_f, β_f, and θ_f, the addition or subtraction of an integer multiple of 2π from these variables will yield the same orientation. With this understanding, it is more appropriate to use Eq. (3.10) for the boundary condition.

A corollary to Eq. (3.11) are the following conditions

$$\dot{\alpha}(t_f) = 0 \tag{3.12}$$

$$\dot{\beta}(t_f) = 0 \tag{3.13}$$

$$\dot{\theta}(t_f) = 0 \tag{3.14}$$

There are also constraints on the angular velocity and torque vectors. The magnitude of any component of the torque vector must be less than one to meet the constraint in this formulation. As the principal moment of inertias and maximal torque output of a satellite can vary, a time normalization is necessary to generalize these constraint bounds. The following normalized time unit (TU) and torque or moment unit (MU) will be used to insure the constraints outlined in this problem:

$$1TU = \sqrt{I/M_{max}} \tag{3.15}$$

$$1MU = M_{max} \tag{3.16}$$

The magnitude of the angular velocities are taken to always be sufficiently under

the constraint values given a torque profile that meets the previous criteria.

Lastly, there are path constraints on the system. Aligning the initial sensor axis with both the 1-axis of the body-fixed frame and the inertial frame, the relationship between sensor-axis in the body-fixed frame and the inertial axis is given by Eq. (2.11). In vector form, the time-history of this orientation for a given problem is:

$$\sigma(t) = [\sigma_1(t)\ \sigma_2(t)\ \sigma_3(t)] \tag{3.17}$$

Another assumption made is that the maneuver is being completed in a time-frame short enough such that the translational motion of the satellite, as well as constraint-bodies, is negligible. Now, for each object that has a constraint cone, the alignment of the central axis of the body can be generalized as

$$\gamma_i = [\gamma_{1_i}\ \gamma_{2_i}\ \gamma_{3_i}] \tag{3.18}$$

which does not change with time for the given body.

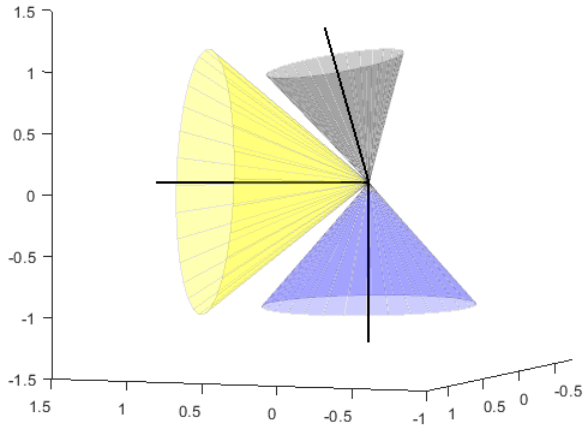

Figure 3.1. Example of constraint cones imposing path constraints on a maneuver space.

The angle between the sensor axis and the constraint-body central axis will be:

$$\angle\sigma(t)\gamma_i = \arccos(\sigma(t) \cdot \gamma_i) \tag{3.19}$$

Given this information, checking to determine if the path constraint is met is equivalent to checking that the angle from Eq. (3.19) is greater than the angle of the keep-out-cone corresponding to body i. An example of a three constraint body scenario can be seen in Fig. 3.1.

3.2 Cost Function

As mentioned in Section (2.3), a cost function is required to evaluate the particles with respect to optimality and constraint conditions. Many different versions of the cost function were tested for this analysis depending on the approach and the resulting performance. The cost function will first be considered without the addition of cone-constraints as those produce additional terms that are added later.

Given an unconstrained reorientation maneuver that is being optimized with respect to final time, the base cost function that is to be optimized is

$$J_{t_f} = \int_0^{t_f} 1dt$$

or, more simply,

$$J_{t_f} = t_f \tag{3.20}$$

The optimum solution is the maneuver that minimizes J_{t_f}. However, there are other constraint criteria that are not always directly satisfied by the approach to the problem. These are the boundary conditions - the initial and final orientations and angular velocities. Taking Eqs. (3.4), (3.5), (3.10), (3.11) to be the boundary conditions on the problem, a few additional terms can be added to the cost function. If ϵ_0, ω_0 are the initial conditions based on the problem formulation, and ϵ_{t_f}, ω_{t_f} are the final conditions, then two additional terms for the cost function can be introduced:

$$\begin{aligned} J_\omega &= ||\omega_0 - \omega(0)|| + ||\omega_{t_f} - \omega(t_f)|| \\ J_\epsilon &= min(||\epsilon_0 - \epsilon(0)||, ||\epsilon_0 + \epsilon(0)||) + min(||\epsilon_f - \epsilon(t_f)||, ||\epsilon_f + \epsilon(t_f)||) \end{aligned}$$

For J_ϵ, the reason to consider the minimum relationship between the quaternion at the beginning and final times is a consequence of Eq. (2.11). Multiplying the

quaternion vector by -1 produces an identical orientation axis when transformed back to the inertial frame.

Simplifying the above equations, if the initial quaternion conditions and angular velocities can be enforced, and the final condition for the angular velocities is zero, the equations become:

$$J_\omega = ||\omega(t_f)|| \tag{3.21}$$

$$J_\epsilon = min(||\epsilon_{t_f} - \epsilon(t_F)||, ||\epsilon_{t_f} + \epsilon(t_F)||) \tag{3.22}$$

Another constraint that needs to be satisfied is the maximum torque that can be produced about each axis. For this consideration, the torque for each body-axis rotation is allowed to reside in the domain $[-1,\ 1]$. Thus, if the solution approach is not able to guarantee this constraint, an additional term to the cost function must be added. There are many different approaches that could be taken when developing this term. For this analysis, a linear term composed of the maximal value of any individual torque component during the maneuver was used for simplicity.

$$J_M = max(max(0, ||M_1(t) - 1||), max(0, ||M_2(t) - 1||), max(0, ||M_3(t) - 1||) \tag{3.23}$$

For the conditions used, the cost function for the unconstrained maneuver becomes:

$$J_{UC} = a_1 J_{t_f} + a_2 J_\omega + a_3 J_\epsilon + a_4 J_M \tag{3.24}$$

where the a_i are weighting factors, whose values are chosen based upon the significance of the constraints relative to one another.

For the constrained maneuver, additional terms are to be added for each constraint-body. Three will be considered here, however this can be easily extrapolated for additional bodies. Given the orientation of the central axis of the bodies as shown in Eq. (3.18), and the orientation of the sensor over the timespan of the maneuver, given by Eq. (3.17), the angle between any individual body's central axis and the satellite sensor axis is determined from Eq. (3.19). Each body has an angle associated with the field-of-view of sensor and the apparent size and brightness of the body. These half-angles will be denoted as $\angle i$. So, for each constraint body, an additional term will be added to the cost function as follows:

$$J_{\sigma_i} = \begin{cases} \frac{1}{(\angle\sigma(t)\gamma_i + \in)}, & \text{if } \angle i \geq \angle\sigma(t)\gamma_i \\ 0, & \text{otherwise} \end{cases}$$

where $\in$ is a small number to ensure the denominator is never 0. The above condition is calculated at the time t during the maneuver that results in the closest pass of the sensor axis to the constraint body orientation axis.

The constrained cost function, to be denoted by J, is then

$$J = J_{UC} + \sum_i b_i J_{\sigma_i} \tag{3.25}$$

with the b_i being additional weighting factors. The exact formulation for the terms of J can change depending on what is and isn't enforced by the solution approach.

3.3 Chebyshev Polynomials

The first approach taken in this analysis was to model the angles α, β, and θ as a linear combination of Chebyshev polynomials of the first kind (these will simply be referred to as Chebyshev polynomials). Chebyshev polynomials have a useful property that on the interval of [-1, 1], the maximum value any individual Chebyshev polynomial, T_n, takes is 1. More properties and uses of Chebyshev polynomials can be found in [11]. It was seen that this magnitude limitation may be beneficial when modeling the time-histories of the orientation angles.

Chebyshev polynomials can be generated using the recurrence relationship:

$$T_0(t) = 1 \tag{3.26}$$

$$T_1(t) = t \tag{3.27}$$

$$T_{n+1}(t) = 2t * T_n(t) - T_{n-1}(t) \tag{3.28}$$

MATLAB provides a built-in function for producing these polynomials [12], however the simple relationship shown above would allow for a straightforward implementation in any coding language, an algorithmic benefit when considering real use cases.

A linear combination of Chebyshev polynomials has the form

$$\begin{aligned} f(t) &= a_0(T_0(t)) + a_1(T_1(t)) + \textstyle\sum_{i=2}^{n} a_i * (2t * T_{i-1}(t) - T_{i-2}(t)) \\ &= a_0 + a_1 t + \textstyle\sum_{i=2}^{n} a_i * (2t * T_{i-1}(t) - T_{i-2}(t)) \end{aligned} \tag{3.29}$$

As part of the boundary conditions of the system, it will be necessary to construct a way to generalize the constant term and the linear term of Eq. (3.29) to be 0. That is not as simple as setting a_0 and a_1 equal to 0 as the summation also includes constant and linear terms. To demonstrate some patterns in these terms, the first 8 Chebshev polynomials will be listed.

$$\begin{aligned} T_0 &= 1 \\ T_1 &= t \\ T_2 &= 2t^2 - 1 \\ T_3 &= 4t^3 - 3t \\ T_4 &= 8t^4 - 8t^2 + 1 \\ T_5 &= 16t^5 - 20t^3 + 5t \\ T_6 &= 32t^6 - 48t^4 + 18t^2 - 1 \\ T_7 &= 64t^7 - 112t^5 + 56t^3 - 7t \end{aligned}$$

If all that matters is the constant and linear terms of these polynomials, the following relationships can be extracted:

$$T_{2n} = (\sum_{i=1}^{n} c_i t^{2i}) + (-1)^{n+2} \tag{3.30}$$

$$T_{2n+1} = (\sum_{i=1}^{n} c_i t^{2i+1}) + (2n+1)(-1)^{n+2} t \tag{3.31}$$

where the c_i are not necessary for this consideration.

Thus, if the constant term is to be equal to 0, the constants in Eq. (3.29) can be manipulated as:

$$a_0 = -\sum_{i=1}^{n} a_{2n}(-1)^{n+2} \tag{3.32}$$

and for the linear term:

$$a_1 = -\sum_{i=1}^{n} a_{2n+1}(2n+1)(-1)^{n+2} \tag{3.33}$$

In order to meet the initial condition constraint of the problem, these are the only two relationships needed.

While the above is not a rigorous proof of this relationship for an arbitrary Chebyshev polynomial of order m, no order greater than 7 was used during this analysis, so the above is sufficient.

The PSO implementation of these polynomials was made modular so that the maximum order of the linear combination of Chebyshev polynomials taken could arbitrarily be chosen. The particle elements that needed to be defined were the coefficients a_i in Eq. (3.29). For α and β, a_0 is defined using Eq. (3.32), and the elements that PSO is solving for are the additional a_i. For θ, a_1 is also defined independently of the PSO initialization using Eq. (3.33). If α, β, and θ are linear combinations of Chebyshev polynomials up to the order α_{num}, β_{num}, and θ_{num} respectively, the total number of elements PSO needs to solve for is $\alpha_{num} + \beta_{num} + \theta_{num} - 1$, where the one element being removed is the linear term known for θ.

Additionally, an extra element to each particle can be added which holds the information for the final time. If this is done, the final time is set for any evaluation of a particle, and the cost function will be calculated with this information in mind. The alternative to this, a method that was also attempted, was to leave the final time ambiguous for any particle definition, and to evaluate the cost function over a set bounded time interval for every particle. The final time associated with the best cost value for a given particle was saved and associated with the particle.

3.4 Finite Power Series

3.4.1 Single Polynomial Optimization

The second approach taken was to model the time-histories of α, β, and θ using a power-series representation with PSO solving for the coefficients of each order element. Taking an arbitrary function $f(t)$, this power-series representation has

the form

$$f(t) = \sum_{i=0}^{n} a_i t^i \tag{3.34}$$

The main benefit of this formulation over Chebyshev polynomials is that to meet the constraints of the system, setting the constant and linear terms to 0 is as simple as letting a_0, a_1 equal 0. The direct downside versus Chebyshev polynomials is that there is no bounding on the value of $f(t)$ between any time-range that is explicitly the result of the summation of different ordered terms.

Given an order of α_{num}, β_{num}, and θ_{num} for each respective polynomial, the number of elements PSO is solving for related to the modeling is the same as for the Chebyshev polynomials, i.e. $\alpha_{num} + \beta_{num} + \theta_{num} - 1$, however the constant and linear term constraints are no longer linear combinations of the other coefficients, they are just 0 where applicable. This makes implementation much simpler.

Depending on the approach, there is an optional time element added to the PSO particles if it is also trying to solve for the final time rather than using the cost-function for this determination. The nature of this is exactly as was mentioned in the previous section.

3.4.2 Segmented Timespan Polynomial Optimization

Continuing on from the last section, instead of a single polynomial representation for the orientation angles, it is preferential to segment the time-span of the maneuver into multiple sections and create a polynomial representation for the time-histories of the angles in those domains. Due to the dynamical model, the orientation and angular velocities of these domains must be continuous. However, the torques in the system are allowed to change instantaneously to any value in the interval [-1, 1] for any of the three torques, so, relative to these values, the time-histories do not need to be continuous. From Eq. (2.5) and (2.7), this means that the power-series representations of these sections must be at least first-derivative continuous. This will be elaborated on briefly, but first consider how this could be represented.

If the maneuver is to last a time-span t_f and it is sought to represent this maneuver in n_{sec} sections, one representation of such a system would be the

following:

$$f(t) = \sum_{j=0}^{n_{sec}} \sum_{i=0}^{n_{coe}} (u(t - \frac{t_f}{n_{sec}} j) - u(t - \frac{t_f}{n_{sec}}(j+1))) * a_{ij}(t - \frac{t_f}{n_{sec}} j)^i \tag{3.35}$$

where n_{coe} is the order of the polynomial used in each section, $u(t-a)$ is the unit step-function where $u(t-a)$ is 0 when $t < a$ and 1 when $t \geq a$. This function assumes that the time-span of the maneuver is equally segmented in n_{sec} ways, and the polynomial used in each time-segment is of order n_{coe}. At each node of the function, the location at which one section ends and another begins, the previous polynomial representation is shut off by the unit step-function, and the new representation is activated. The time represented by the new polynomials is shifted so that the initial value of the polynomial is with respect to a zero time value.

To guarantee the initial conditions of the system, for α, β, and θ, a_{00} can be set equal to 0. For θ, a_{10} also needs to be set to 0. To provide up to first-derivative continuity, at each intersection point,

$$a_{0j} = \sum_{i=0}^{n_{coe}} a_{i(j-1)} (\frac{t_f}{n_{sec}})^i \tag{3.36}$$

$$a_{1j} = \sum_{i=0}^{n_{coe}} i a_{i(j-1)} (\frac{t_f}{n_{sec}})^{i-1} \tag{3.37}$$

For this implementation to work, it requires that the final time t_f be known for a given particle, so that value is given as a parameter for PSO. Therefore, if the order of the polynomials of each time-segment are α_{num}, β_{num}, and θ_{num}, respectively, the number of PSO particles needed just for the coefficients will be $n_{sec}(\alpha_{num} + \beta_{num} + \theta_{num} - 3) + 2$. The -3 term corresponds to the 'known' linear term components in each time-segment (due to the necessary continuity), and the $+2$ is because the polynomial corresponding to the first time-segment does not have a 'known' $\dot{\alpha}_0$, $\dot{\beta}_0$. In addition to these elements, one more element that denotes the final time must be included.

3.5 Higher-order Derivative Optimization

3.5.1 Single Polynomial Optimization

The main difficulty with the polynomial representations used in the previous approaches is that there is no clear way to:

- Guarantee the final orientation of the satellite for every single particle PSO checks (moreover, it is extremely difficult to find a solution that even yields the correct final orientation);
- Guarantee the final angular momentums of the maneuver to be equal to 0 (again, extremely difficult to find a solution that meets this criteria, let alone this and the previous remark);
- Guarantee that the torques associated with the maneuver never leave the feasibility interval of [-1, 1] for any individual component.

Ignoring the cone-constraints that also pose difficulty, the above three conditions are the constraints that the previous methods cannot directly enforce. They must be implemented as soft constraints in the cost function so that they are met within a certain tolerance.

This led to a new approach to this problem, still looking at inverse dynamics. Given what is known about the boundary conditions of the problem related to the orientation and angular velocities, it is sought instead to fit a polynomial directly to these boundary conditions on the interval of [0, 1] and use ambiguous values for higher-order derivatives at the boundaries of the maneuver. A relationship between the torques and the final time can then be used to determine the lowest value for the final time that guarantees not only the initial conditions, but all of the boundary conditions and constraints mentioned above.

First, consider a polynomial fit between two points where the value of those points, and the first n-derivatives of those points are known. A least squares fitting

method can be used to determine the coefficients of that polynomial [13]. I.e:

$$\begin{bmatrix} f(t_i) \\ f'(t_i) \\ \vdots \\ f^{(n)}(t_i) \\ f(t_f) \\ f'(t_f) \\ \vdots \\ f^{(n)}(t_f) \end{bmatrix} = \begin{bmatrix} 1 & t_i & t_i^2 & t_i^3 & t_i^4 & \dots & t_i^{2n} & t_i^{2n+1} \\ 0 & 1 & 2t_i & 3t_i^2 & 4t_i^3 & \dots & 2nt_i^{2n-1} & (2n+1)t_i^{2n} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & \frac{2n!}{n!}t_i^n & \frac{2n+1!}{(n+1)!}t_i^{n+1} \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & \dots & t_f^{2n} & t_f^{2n+1} \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & \dots & 2nt_f^{2n-1} & (2n+1)t_f^{2n} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & \frac{2n!}{n!}t_f^n & \frac{2n+1!}{(n+1)!}t_f^{n+1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \\ a_{n+1} \\ a_{n+2} \\ \vdots \\ a_{2n+1} \end{bmatrix}$$

written more simply as:

$$\vec{y} = A\vec{a} \tag{3.38}$$

Eq. (3.38) can be used to solve for the coefficient vector $\vec{a}$ given known t_i, t_f, and point components associated with $\vec{y}$. A is a $(2n+2) \times (2n+2)$ matrix and is invertible as the elements correspond to independent values of the two points being fit.

For the maneuvers being evaluated, t_i is set to 0 and t_f is set to 1 for all samplings. For α, β, and θ, $f(0) = 0$. For only θ, $f'(0) = 0$. Additionally, $f(1)$ equals $\alpha + 2k\pi$, $\beta + 2m\pi$, $\theta + 2l\pi$ for each respective fit, and $f'(1) = 0$ for all three angles. As the final orientation won't change for any integer addition/subtraction of 2π, per Eq. (2.5), k, m, and l allow for a more diverse sampling of potential attitude paths.

The above approach guarantees

1. The initial orientation of the satellite;
2. The final orientation of the satellite;
3. The initial angular velocity of the satellite;
4. The final angular velocity of the satellite;

The two constraints that it does not implicitly satisfy are the torque constraints and the keep-out cone constraints. Using Eqs. (2.3), (2.7), and (2.8), a relationship

can be extracted between the torques and time. Consider a time normalization of $\tau = \frac{t}{t_f}$. Given the time-span of t is [0, t_f], the time-span of τ is [0, 1]. Thus, the following can be considered:

$$\tau = \frac{t}{t_f} \tag{3.39}$$

$$\frac{d}{dt} = \frac{d}{d\tau}\frac{d\tau}{dt} = \frac{d}{d\tau}\frac{1}{t_f} \tag{3.40}$$

Since the polynomial fits are being considered on an interval of [0, 1], that can be associated with respect to the τ-time normalization. Thus, to determine a final time that meets the torque constraints imposed, the equations referenced above can be rewritten in terms of the τ time-variable with a t_f normalization component. Letting the $'$-notation imply a τ-derivative and $\dot{}$-notation imply a time-derivative, the angular momentum can be written as

$$\begin{aligned}
\omega(t) &= 2\mathbf{E}(t)\dot{\epsilon}(t) \\
\omega(\tau) &= 2\mathbf{E}(\tau)\epsilon'(\tau)\frac{d\tau}{dt} \\
&= 2\frac{1}{t_f}\mathbf{E}(\tau)\epsilon'(\tau)
\end{aligned}$$

Next, the angular acceleration can be written as:

$$\begin{aligned}
\dot{\omega}(t) &= 2\dot{\mathbf{E}}(t)\dot{\epsilon}(t) + 2\mathbf{E}(t)\ddot{\epsilon}(t) \\
\omega'(\tau) &= 2\frac{d\tau}{dt}\mathbf{E}'(\tau)\frac{d\tau}{dt}\epsilon'(\tau) + 2\mathbf{E}(\tau)\frac{d^2\tau}{dt^2}\epsilon''(\tau) \\
&= 2\frac{d^2\tau}{dt^2}(\mathbf{E}'(\tau)\epsilon'(\tau) + \mathbf{E}(\tau)\epsilon''(\tau)) \\
&= 2\frac{1}{t_f^2}(\mathbf{E}'(\tau)\epsilon'(\tau) + \mathbf{E}(\tau)\epsilon''(\tau))
\end{aligned}$$

Lastly, the torque equations in terms of t_f (M_1 can be used without loss of generality):

$$\begin{aligned}
M_1(t) &= I_1\dot{\omega}_1(t) + \omega_2(t)\omega_3(t)(I_3 - I_2) \\
M_1(\tau) &= I_1\omega'_1(\tau) + \omega_2(\tau)\omega_3(\tau)(I_3 - I_2)
\end{aligned}$$

$$= \frac{1}{t_f^2}(F(\epsilon(\tau), \dot{\epsilon}(\tau), \ddot{\epsilon}(\tau)))$$

where $F(\epsilon(\tau), \dot{\epsilon}(\tau), \ddot{\epsilon}(\tau))$ is a function of the quaternion elements and their derivatives in the τ-frame after converting from the t-frame.

Essentially, the angular velocities have a linear factor of $\frac{1}{t_f}$ that separates the adjusted τ-time and the real-time values, and the torques have a factor of $\frac{1}{t_f^2}$. The benefit of this normalization is that the maneuver can be analyzed in τ-space, and then the torque values can be normalized.

This means that the magnitude of the torque can be completely accounted for using the $\frac{1}{t_f^2}$ coefficient. So, to determine the final time that corresponds to a maximum torque magnitude of 1 for any component, all that needs to be done is to take the maximum magnitude any of the torque components sees during the maneuver in the τ-frame, and set the real total maneuver time equal to

$$t_f = \sqrt{||M_i||_{max}} \tag{3.41}$$

for whichever component M_i has the highest magnitude value.

The only constraint this approach does not directly satisfy is the keep-out cone constraints, but a solution that satisfies those can be obtained using PSO. Additionally, fitting with just the boundary conditions does not yield a time-optimal solution, so PSO seeks to solve that problem as well.

If the order of the polynomial fits are to be α_{num}, β_{num}, and θ_{num} respectively, the number of elements needed per particle is $\alpha_{num} + \beta_{num} + \theta_{num} - 4$. For a given angle, say α, $\alpha_{num} - 2$ elements are needed. The -2 corresponds to the total number of known parameters at the endpoints (three), minus one for the constant term of the polynomial. For θ, one less is needed as four total parameters are known. In addition to this, three additional elements are added to the particle that are associated with the k, m, and l integers mentioned above that allow the final angles to be an integer multiple of 2π from their expected values.

Results of this approach will be discussed later.

3.5.2 Segmented Timespan Polynomial Optimization

The last approach taken was to expand on the process in the previous section. PSO is given the option to guess at the value of the three angles at intermediate points throughout the maneuver, and fit a polynomial between each consecutive set of points. For example, if PSO is allowed to solve for three intermediate points in the maneuver, including the endpoints, there would be five total points, and four polynomial fit To do this. Eq. (3.38) is used between every two consecutive points in the system - this includes the initial and final points, and all the intermediate points PSO is guessing for. The polynomial fits need to be first-order continuous to guarantee that the angular velocities are continuous.

If there are n-points in the system, $n-2$ points PSO is calculating and the two endpoints known, and the order of the polynomial fits are to be α_{num}, β_{num}, and θ_{num} for each angle, respectively, between every two points, there will be $\frac{n}{2}(\alpha_{num}+\beta_{num}+\theta_{num}+3)-7$ total elements in each particle. This comes from needing $x+1$ elements to fit a polynomial of order x between two points. In this procedure, the number of elements is split evenly between each point - the consequence of this is that it limits the polynomial fit to odd-ordered polynomials. The -7 is composed of the 10 particle elements already known between α, β, and θ at the endpoints, and the 3 additional elements corresponding to the k, m, and l integers mentioned in the previous section.

Outside of the differences in the way the particles are set up, the rest of the procedure is identical to that in Section (3.5.1). These polynomial fits are connected at the equidistant nodes, and the polynomials are evaluated to determine the control-profile and final time that meets the constraints.

Chapter 4
Results

4.1 Case Study

For each approach, the initial case that was used to determine feasibility was that of the unconstrained reorientation maneuver. If the method could not produce results for the unconstrained case, it was deemed ineffective for producing results to the more difficult optimization problem.

To best evaluate the results produced by a given method, the conditions from [14] were used to provide a direct comparison to previous research throughout the process. Once an effective method was found, other endpoint conditions were used as well on a testing basis, but in this thesis, only the results produced as a comparison to previous literature will be presented. The effective method to be mentioned has been seen to work well under any example endpoint conditions tested.

4.2 Ineffective Approaches

4.2.1 Chebyshev Polynomials

While the Chebyshev polynomial approach was the anticipated path for this research, it was not an effective one for solving this problem. Guaranteeing the final-time constraints was very difficult given that the maneuver time for any tested maneuver would be well above one time-unit, and thus the polynomial approximations for the angle paths would blow up. Additionally, it was determined that the magnitude bounds on Chebyshev polynomials in the interval of [-1, 1] would be ineffective once linear combinations of different order Chebyshev polynomials was taken.

Many different variants of the cost function were used and different order linear combinations were taken for each angle path, but not a single run yielded a solution that met the endpoint conditions. There were two main approaches to accounting for the final time constraint. The first of these was the time-step method, where the cost function for a given particle would be evaluated at many time-values (between a set bounded interval), and the lowest cost time would be saved. The problem with this was that during the first few iterations of PSO, every particle generated had its lowest value at the lower bound. The second method was to set the final time as a particle in PSO. While this method showed more variation in the earlier iterations of PSO, the same fundamental problem existed, and the particles slowly converged to the lower time bound, or all particle elements associated with the polynomial coefficients converged to zero.

Example results of what was produced by this method can be seen in Fig 4.1, 4.2, and 4.3. Notice that theta is zero. All of the coefficients of this Euler parameter converged to 0, and the maneuver never left its initial orientation. This is confirmed in the torques and angular velocity plots, where all of the magnitudes are 10^{-3}. Plots like this were similarly produced for the other ineffective methods, and will not be represented for each.

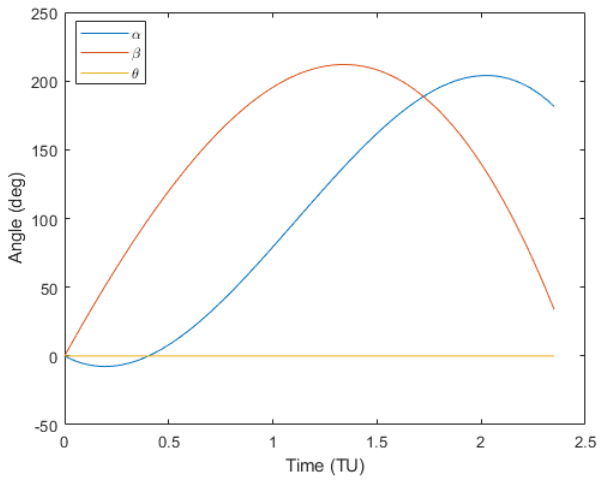

Figure 4.1. Orientation angle polynomial fits over maneuver time.

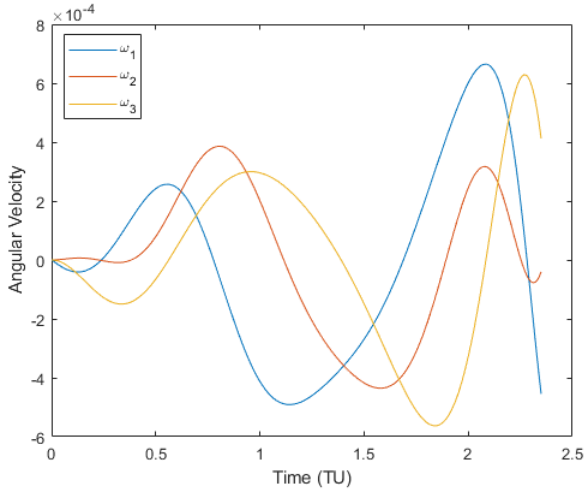

Figure 4.2. Angular velocities of the three axes associated with the satellite body-frame.

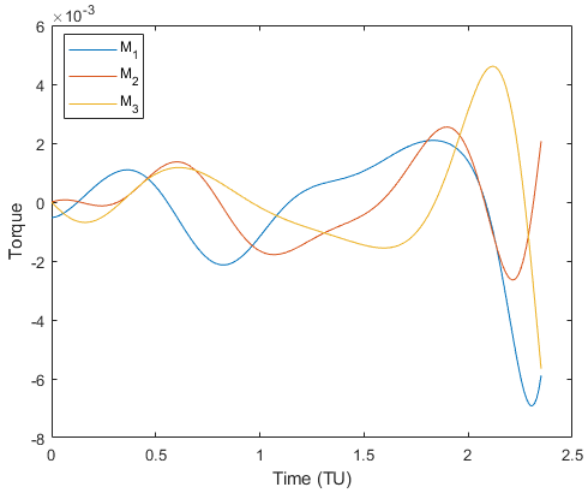

Figure 4.3. Torque of the three axes associated with the satellite body-frame.

To potentially rectify some of the issues with this method, it might be appropriate to impose a similar time normalization as in the higher-order derivative approach. This would allow for the bounding property of Chebyshev polynomials to be used advantageously.

4.2.2 Finite Power Series Single Polynomial Optimization

Once the Chebyshev polynomial approach was determined to be ineffective, moving to a simple finite power series representation for the approximation of the angle

paths was attempted. This appeared to be a more straightforward method, and would allow for better troubleshooting.

It was found that this method had the same issues as with the Chebyshev polynomial method, being its inability to converge on a solution that met the endpoint conditions. Attempting a time-normalization on this method could provide better results.

4.2.3 Finite Power Series Segmented Timespan Polynomial Optimization

As a consistent issue that seemed to appear had to do with the time-span of the solution and the polynomials values exploding, the next approach that was attempted was to divide the solution space into equidistant sections and connect polynomials between them using the power series approach.

Implementation was relatively straightforward, but additional computation time was necessary. One benefit this method provided was that the time-range for any given section could be reduced to less than one time-unit. Dramatic benefits were seen with this as it appeared solutions were no longer converging to zeros or the lower time bound. However, problems arose in two areas. The first involved the number of particle elements necessary for the implementation. To segment the time span into five sections, nearly three times the number of elements per particle were needed as compared to the other methods. Secondly, PSO failed to converge on polynomial connections that eventually would lead to the end-point constraints. The maneuver would wildly vary, and no feasible solution would be produced.

4.2.4 Higher-order Derivative Segmented Timespan Polynomial Optimization

Implementing this approach created a large degree of complexity that sprung up many inefficiencies. While this method appeared to be converging on a solution, the computation time required was well beyond what was already set as the benchmark based on the unsegmented version of this approach. This method was deemed ineffective because of this, although plausible for producing solutions without the computation constraint.

4.3 Effective Approach

When discussing computational efficiency, it is not as simple as comparing the CPU times between solution approaches as there may also be variations in software or other hardware. However, for the solutions generated for this thesis, a core i5-8600K processor was used running at 4.15 GHz with all code written MATLAB R2018a. These are being represented to demonstrate that the time benefits seen when generating a solution exceed that which would be expected purely from hardware or software differences.

4.3.1 Higher-order Derivative Optimization

This method worked exceptionally well for generating solutions to this problem for both the unconstrained and constrained variants in a very short period of time. This can be attested by how the initial and final constraints and torque constraints are perfectly satisfied for every particle PSO initializes. This simplifies PSO's responsibilities to determine, in the unconstrained case, which element set produces the minimal time for the normalization, and for the constrained case, both the minimal time and the solution that avoids the cones.

For the cost function, the only terms that were fully needed were the term associated with the time-normalization to satisfy the torque constraints, as well as an arbitrarily large magnitude step-function associated with the cone constraints. If the maneuver path entered the cones in the constrained case, the cost-function would essentially be infinite, and as soon as a solution was found that avoided these constraints, PSO would start converging. For a better sampling of the space, PSO was allowed to randomly generate particles for a number of iterations so a large enough pool of particles could be sampled prior to utilizing the velocity function of PSO. While this was not necessary, it was found that PSO would consistently converge to slightly lower times than without this functionality, in the same amount of processing time.

4.3.1.1 Conditions

The following conditions were experimentally determined to yield good solutions for any test-cases:

- Order of α, β, θ polynomials being fit is 5;
- Number of particles per iteration is 150;
- Number of prior particle generations is 100;
- Number of iterations of PSO per run is 200;
- The upper-bound on particle elements associated with the derivatives is 45;
- The lower-bound on particle elements associated with the derivatives is -45;
- The upper-bound on particle elements associated with the 2π multiplier is 1;
- The lower-bound on particle elements associated with the 2π multiplier is -1.

Additionally, if the final condition for β is $\frac{\pi}{2}$, the final value of α is arbitrary. Allowing for PSO to solve for the final value of α can reduce the final time of the maneuver to a small extent. In this case, two more conditions are imposed:

- The upper-bound on the particle element associated with α_f is 2π;
- The lower-bound on the particle element associated with α_f is 0.

This value of α will be included in the point component vector when doing the least-squares fit.

4.3.1.2 Unconstrained Solutions

For the unconstrained case, a single example maneuver was tested. The specific details of this test-case were taken from [14] so a direct comparison can be made between the results observed in this research and in prior research. The end point constraints associated with the maneuver tested are as follows:

1. $\beta_f = \frac{\pi}{2}$;
2. $\theta_f = \frac{3\pi}{4}$.

Provided these details, PSO is allowed to solve for α_f. All of the other conditions are consistent with the problem formulation, which is a rest-to-rest maneuver with zero-valued initial angle orientations. The angle between the initial and final sensor axis orientation is $\frac{3\pi}{4}$.

From [5], it is known that the eigenaxis rotation time associated with a rotation angle of $\frac{3\pi}{4}$ with unit-torque limits is $3.07TU$, and the optimal maneuver time associated with this reorientation is $2.8845TU$. These are bang-bang solutions that are produced directly from the optimal-control formulation for the dynamics.

From this research, a solution with final time close to that in the eigenaxis solution was not able to be replicated for the unconstrained case. However, a solution time of $3.2741TU$ can be seen in the figures below. The final α value associated with this maneuver was determined to be 0.23412 radians. This solution took 13.521 seconds of computation time to complete the 200 iterations of PSO and obtain the solution presented. Fig. 4.4 shows the attitude orientation path the satellite takes. Notice, this does not look like an eigenaxis rotation, but rather has some precession out of plane, as is expected from previous research such as [5].

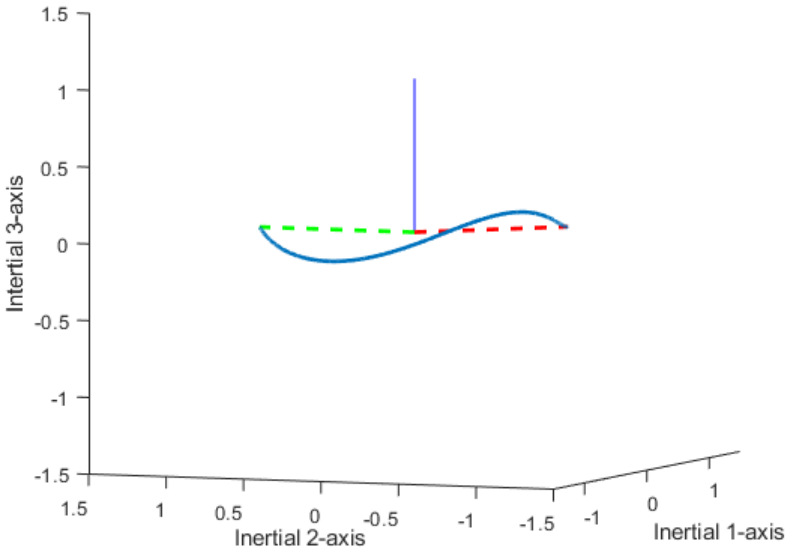

Figure 4.4. Final maneuver path corresponding to a final time of 3.2741 TU (computation time 13.521 seconds).

Fig. 4.5 has the three orientation angle time-histories plotted as a function of true time. There is no 2π multiplier for the final orientation angles as the quickest

maneuver time associated with the unconstrained case involves a reorientation in the path of shortest angular distance.

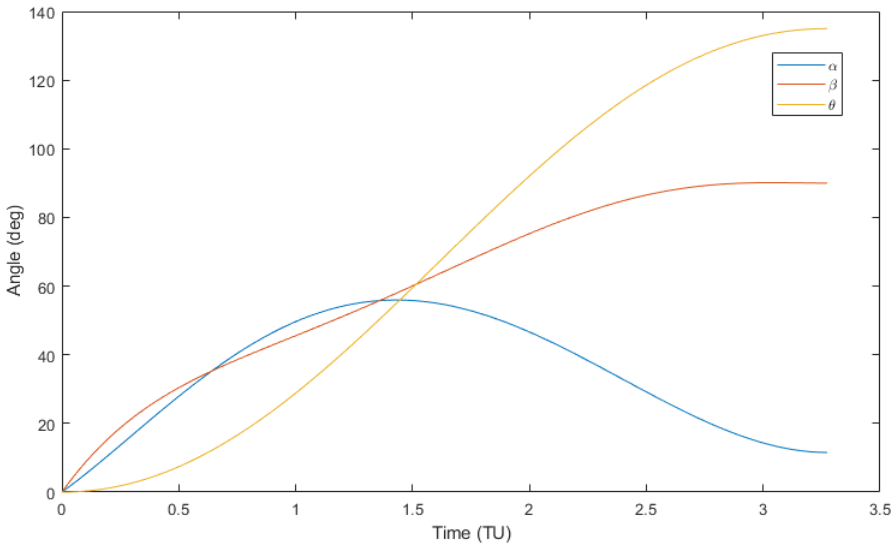

Figure 4.5. Orientation angle polynomial fits over maneuver time.

Fig. 4.6 are the angular velocities associated with each body-axis. While this maneuver looks like it is solely within the 1-2 plane, and thus would be a three rotation if the eigenaxis maneuver were used, it instead shows signs of the precession mentioned earlier.

Fig. 4.7 are the torques associated with this maneuver. They are not bang-bang as they are the time-histories approximated with polynomials, but there are signs of control switching and the different torques appear to resemble approximations of bang-bang control.

The last figure for this maneuver, Fig. 4.8, shows the convergence of the solution over the iteration number of PSO. The first 100 iterations of PSO are completely random sampled particles, so the gain over that period is random. After 100 iterations, there is a quick drop off where the time of the maneuver is decreasing, and it slowly tapers off as the benefit from the velocity function decreases.

From [14], for the unconstrained case, using a B-spline method with the inverse dynamic PSO approach, a final maneuver time of 4 TU was found that had satisfied all of the problem constraints. In this investigation, a more time-efficient maneuver, with final time 3.2741 TU, was found that also met these constraints. Not only

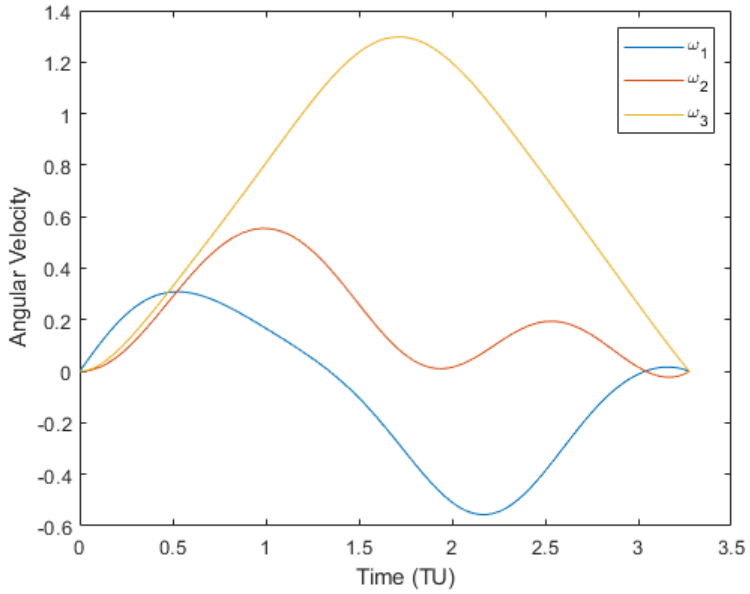

Figure 4.6. Angular momentum of the three axes associated with the satellite body-frame.

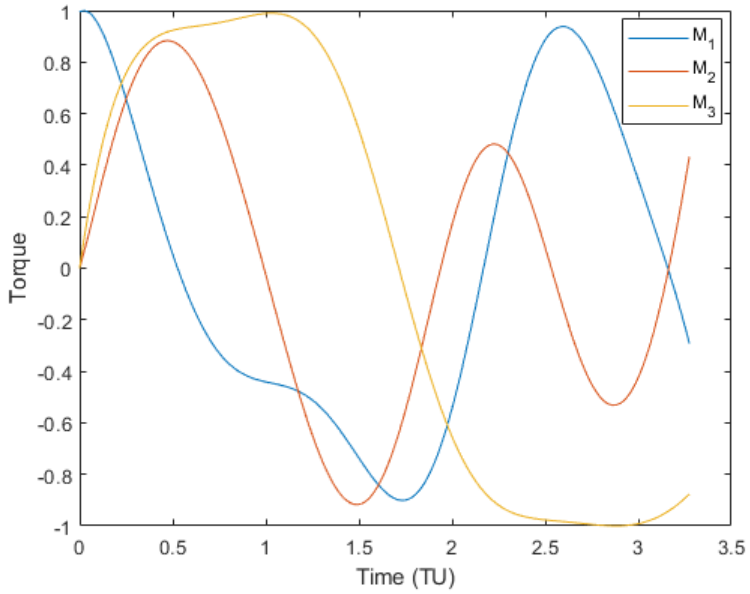

Figure 4.7. Torque of the three axes associated with the satellite body-frame.

was the solution found better than that of the previous research, it was also found much quicker, computationally. In [14], it took 3000 iteration of PSO and 2 hours of computation time to produce the 4 TU solution. In this investigation, it took

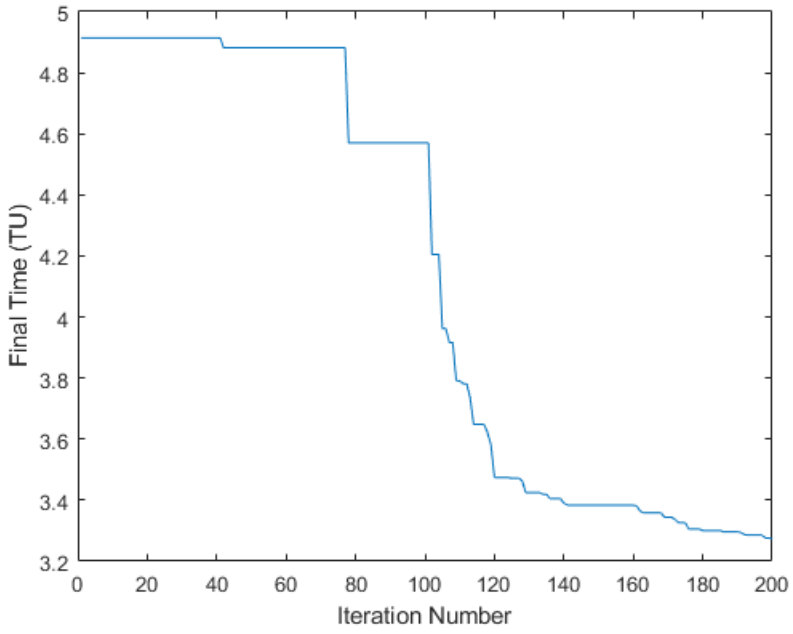

Figure 4.8. Convergence of cost function over PSO iterations.

200 iterations of PSO and 13.521 seconds of computation time. While there is a significant speed difference between the CPU's in each analysis, [14] using an Intel Core i3 2.27 Ghz processor versus the Intel Core i5 4.15 Ghz processor in this analysis, this does not completely account for the 530 times increase in speed of the algorithm. Provided this information, the method used in this research provides both a quicker, and better solution to the problem being investigated.

4.3.1.3 Constrained Solutions

The first constrained case attempted had the same end angle conditions as the unconstrained case. For this problem, there were three cones that were considered - the Earth, the Sun, and the Moon. The angles associated with the keep-out-cones are as follows:

- $\gamma_{Earth} = 33^o$;
- $\gamma_{Sun} = 47^o$;
- $\gamma_{Moon} = 23^o$;

These angles correspond approximately to those used for the Swift satellite's sensors.

A maneuver was found in 200 iterations of PSO with a total maneuver time of 3.9659 TU. The final value of α determined by PSO was 0.41564 radians. This solution took 15.968 seconds of computation time to obtain. In Fig. 4.9, the path that this maneuver takes around the cones can be seen. The reorientation path takes the sensor in the opposite direction of the smallest angular distance. It was found that taking this path converged onto a solution quicker than forcing the maneuver to take a path around the cones. That does not mean this is the path that the true optimal solution would take.

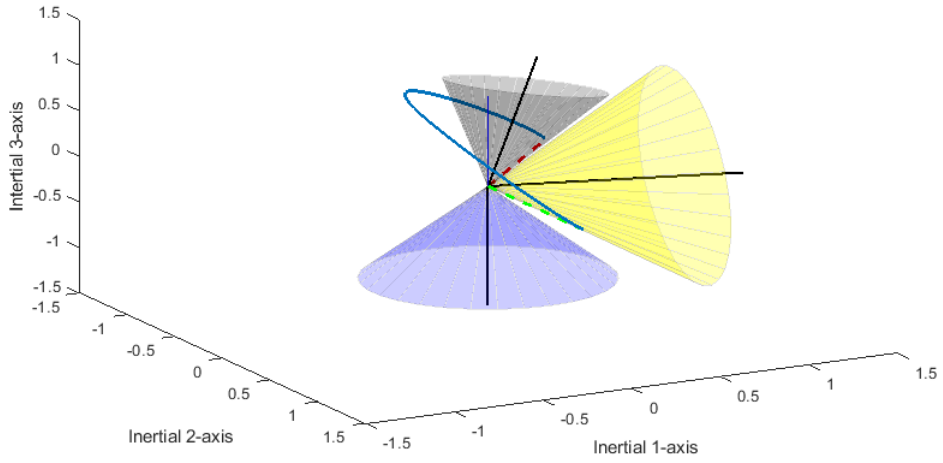

Figure 4.9. Final maneuver path corresponding to a final time of 3.9659 TU (computation time 15.968 seconds).

If the satellite took the path opposing the shortest eigenaxis rotation, the angle of rotation necessary would be $\frac{5\pi}{4}$ radians. The time of an eigenaxis rotation to cover this angular distance would be $3.87TU$ for a unit torque magnitude. The maneuver shown appears to be somewhere in between the shortest and longest angular distance. The solution found does not meet either of the eigenaxis rotation times, but compared to other solutions from previous research, the time of the maneuver is quite good. This will be discussed in more detail briefly.

Fig. 4.10 shows the time-histories of the orientation angles. Both α and β at the end points don't take on a factor of 2π, however θ takes on a factor of -2π from its given angle. This is the cause of the reorientation traveling opposed to the

minimum angular distance.

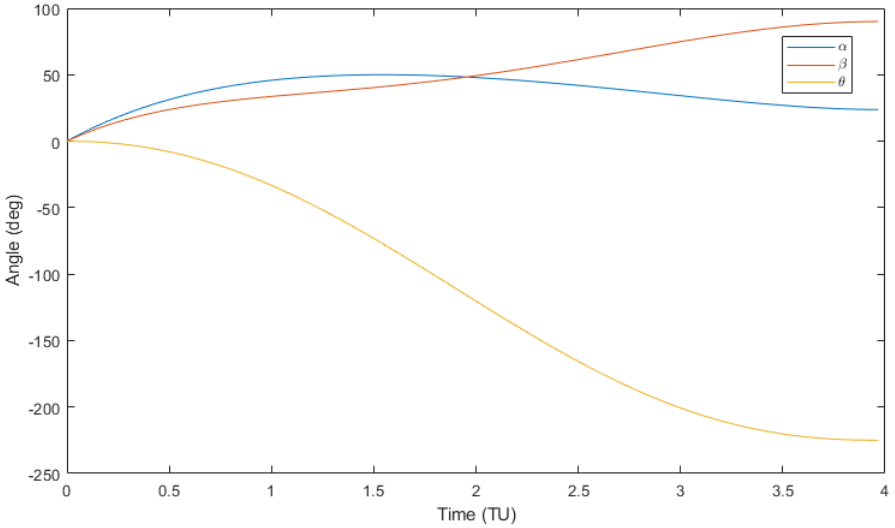

Figure 4.10. Orientation angle polynomial fits over maneuver time.

The angular velocities seen in Fig. 4.6 again see a large magnitude in the 3-component, but also in the 1-component. The torques are shown in Fig. 4.12 and again have what approximately looks like control switching between the maximal and minimal torque values for each component, however the 1-component of the torque has some additional oscillations.

A plot of the cone-constraints being satisfied can be seen in Fig. 4.13. The dashed lines represent the $\sigma's$ associated with each cone, and the solid lines represent the time history of the angle between the cones and the sensor axis. They are color coded for ease of reading. From this plot, it can be seen that sensor axis never comes into contact with cones, and actually maintains an angular distance of $> 10^o$ with every cone at all times.

The final figure for this solution is Fig. 4.14. This figure shows the convergence of PSO as a function of iterations. While the solution value was taken after 200 iterations, for this plot, PSO was allowed to continue to show the reduced benefit from additional computational effort. The first 100 iterations of PSO were randomly sampled particles, and then PSO rapidly converges to the solution until is slowly tapers of. This stagnation can sometimes be remedied depending on the problem with some additional modifications to PSO's algorithm, but those were not explored in this analysis.

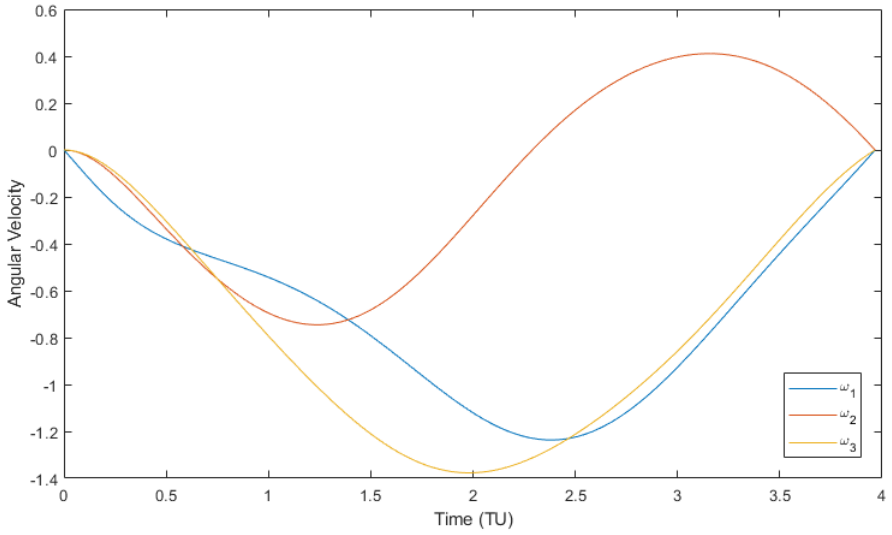

Figure 4.11. Angular momentum of the three axes associated with the satellite body-frame.

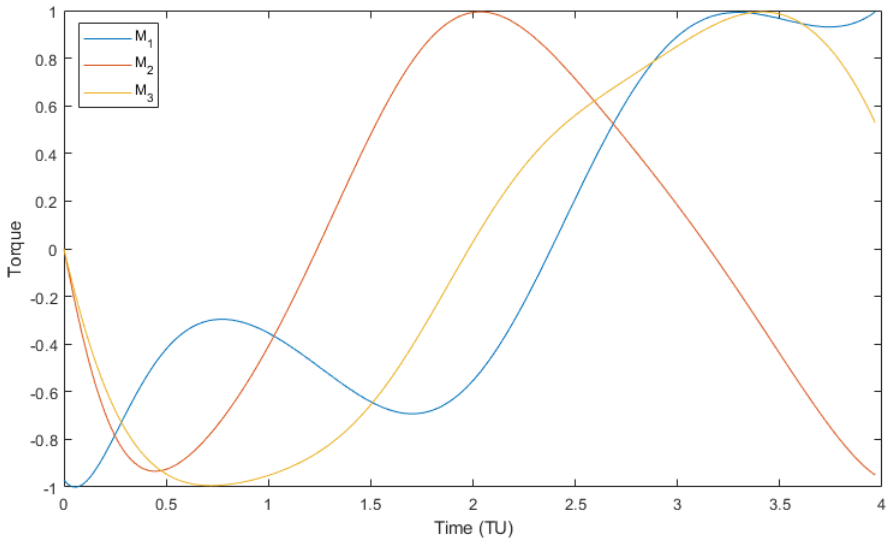

Figure 4.12. Torque of the three axes associated with the satellite body-frame.

Just as with the unconstrained case, all of the conditions, and in this case constraint cones, were taken to be identical to [14] so a comparison can be made. For the comparison, the best result from that paper will be taken.

Fig. 4.15 shows the maneuver path from [14] corresponding to the best solution presented that satisfies all of the problem constraints. The maneuver time associated

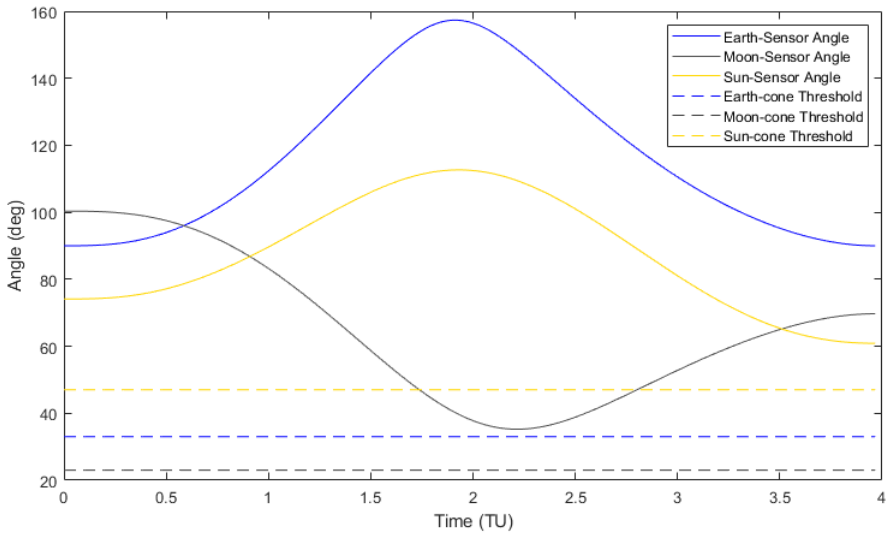

Figure 4.13. Angles of the central sensor axis relative to constraint body central axes. Cone threshold angles plotted for completeness.

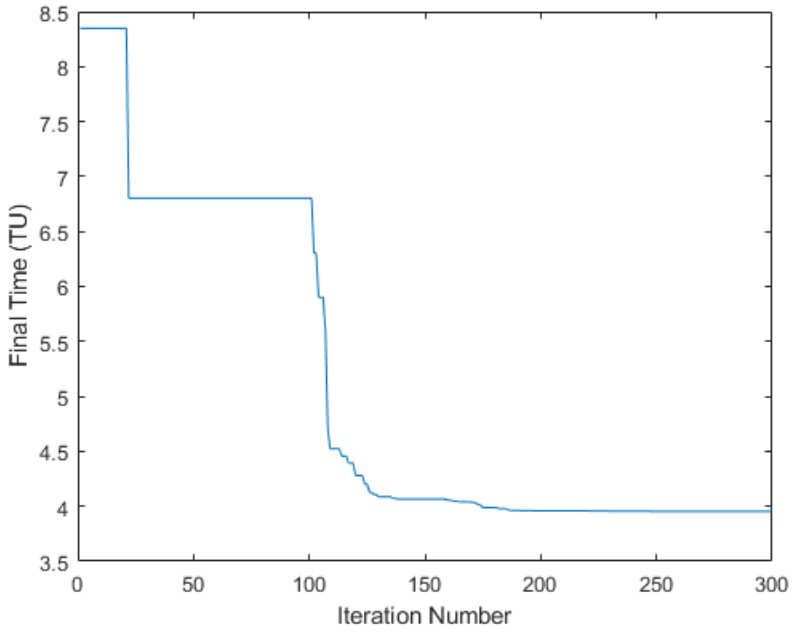

Figure 4.14. Convergence of cost function over PSO iterations.

with this solution is 4.3 TU and took 6 hours of computation time to obtain. That computation time is over 1300x greater than needed for the solution provided in Fig. 4.9, and has a time of roughly 0.3 TU greater.

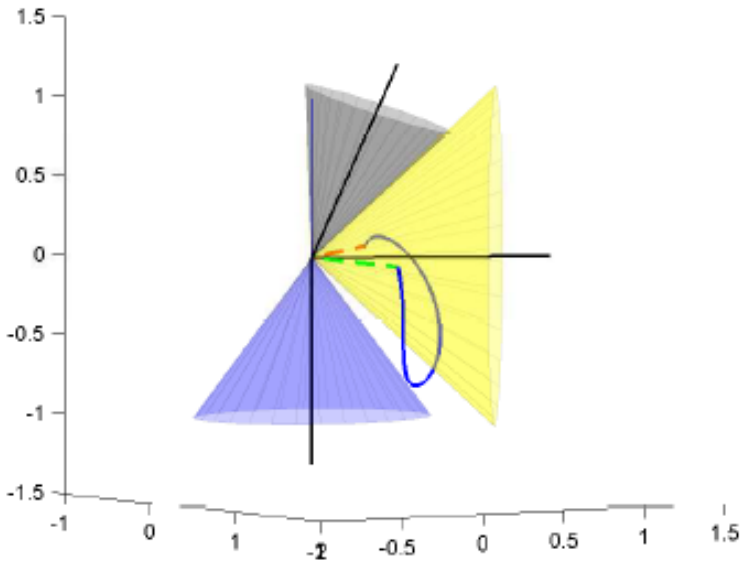

Figure 4.15. Path of solution with final time 4.3 TU from Fig. 18 in [14].

Chapter 5
Conclusions

5.1 Conclusion

There have been multiple studies on producing results to the constrained satellite reorientation problem. Using a direct optimal control method can produce near time-optimal results, but the trade off is computational efficiency. On the other hand, heuristic methods, such as PSO, are able to produce results further from optimality, but with less computation necessary. Some studies investigated a combination of these to produce solutions quicker than a single direct method, but with a solution that is more optimal than the hybrid methods.

All methods evaluated in this thesis used PSO as the sole optimization algorithm. While both maneuver time and needed propellant could be used as optimality constraints for the satellite reorientation maneuver, only the maneuver time was used in this analysis. Several different approaches were formulated and implemented to determine their feasibility as real-time, time-optimal, path-constrained maneuver generators for a rigid-body satellite with three-axis control. For all approaches, an inverse-dynamics formulation for the Euler rotation equations was used. Each method models the 3-2-1 Euler angles' time-histories differently to determine both the attitude and control profile during the course of the maneuver.

The first approach used to model the Euler angles was by approximating their time-histories as a linear combination of Cheybshev polynomials. In solving the direct attitude dynamics variant of this problem, [16] used Chebyshev polynomials to model the control profiles. For the inverse-dynamics case, using Chebyshev polynomials did not work. As the optimal maneuver time for the endpoint conditions

outlined in [14] exceeds 1 TU, the linear combination of polynomials was not able to take advantage of the properties of the individual Chebyshev polynomials. Thus, the Euler angle time-histories being modeled would exceed feasible values. Since the final time of the maneuver was unknown for the modeling of the angles, the endpoint conditions could not be satisfied implicitly. The polynomial representations of the Euler angles never converged on the endpoint conditions for the unconstrained case.

The second approach considered involved modeling the Euler angles using a power series rather than Chebyshev polynomials. This approach did not work for the same reasons the Chebyshev polynomials did not work. For any run of this implementation, PSO was unable to converge on the endpoints of the problem for the unconstrained case.

The third approach used the same power series formulation as the second method, but segmented the time-span of the maneuver such that each segment had a total time of less than 1 TU. This was done in an attempt to reduce the growth of the Euler angles as the maneuver time increased. However, this method never converged on the endpoint conditions for the unconstrained case.

The fourth approach will be mentioned after the fifth as it was the best method found for producing solutions.

The fifth approach incorporated a time-normalization such that the total maneuver time was equal to 1. This method involved segmenting the time-span of the maneuver into different sections, just like the third approach, however a polynomial was fit between every two concurrent points such that the connections were continuous up to the first derivative. First derivative continuity is necessary to ensure the angular velocities were continuous (the torque profiles do not need to be continuous). PSO was allowed to guess at the intermediate point's values for each of the Euler angles, as well as their higher-order derivatives to increase variation in the polynomial shape. The final time of the maneuver was determined based on a relationship between the maximum torque value seen during the maneuver and the time of the maneuver, outlined in Eq. (3.41). While this method was able to produce solutions that met the endpoint conditions for both the unconstrained and constrained cases, the maneuver times were so large for any feasible computation time that this method was dropped before it produced satisfactory solutions (ones with maneuver times near what was seen in approach four).

The best approach evaluated in this thesis, approach four, which will be called

the higher-order derivative approach, used a time-normalization just as approach five did. Rather than segmenting the time-span of the maneuver, each Euler angle was fit with a polynomial directly between the endpoints outlined by the problem. PSO was implemented to optimize the higher-order derivative components of the endpoints for the polynomial fit. By allowing PSO to vary these values, the path the maneuver takes can encompass the entire solution space. The benefit of this approach is that every particle evaluation implicitly satisfies the endpoint conditions and torque constraints for the unconstrained problem. It was also found that this method easily finds a path around any cone-constraints when these constraints are imposed on the cost function. Not only does it find solutions to the path constrained reorientation problem, but it finds solutions significantly quicker than previous literature, and with maneuver-times shorter than previous literature. The solution found for the comparison in Section [4.3.1.3] was computed 1300 times quicker, with a maneuver time 7.77% faster.

The objective of this research was to determine if there could be an algorithm to solving the path constrained satellite reorientation maneuver with computation time feasible for a real mission. As a result of this research, it appears that modifications to the higher-order derivative algorithm presented here may be a suitable candidate for a mission reorientation control system.

5.2 Future Investigations

While a method has been presented that could be a candidate for a real-time reorientation system, there are a few more avenues that could be beneficial to explore.

In this research, a rigid body satellite with three-axis control capabilities was used. While approximations like this can be useful for theoretical work, it may be necessary to further generalize the problem outline to account for other satellite properties. The maneuver used was also taken to be a rest-to-rest maneuver. Creating a generalized model for any set of end point conditions (orientation and angular velocities) would be a more realistic model for mission use.

In the implementation of the higher-order derivative method, the bounds on those higher-order derivatives were not rigorously determined. Some work on producing more refined bounds on these values based on the maneuver could

drastically benefit computation time. Also, in this research, while computational efficiency was important, there were many mathematical definitions in the code that could be adjusted to provide more efficiency.

As MATLAB was used for this analysis, moving to a more efficient coding language would definitely provide significant speed increases.

Bibliography

[1] S. Potter, "NASA Missions Catch First Light from a Gravitational-Wave Event." NASA–Solar System and Beyond. Accessed February 10th, 2019. https://www.nasa.gov/press-release/nasa-missions-catch-first-light-from-a-gravitational-wave-event

[2] "Gamma-ray Bursts." NASA– Goddard Space Flight Center. Accessed February 10th, 2019. https://imagine.gsfc.nasa.gov/science/objects/bursts1.html

[3] "About the Swift Gamma-Ray Burst Mission." NASA–Goddard Space Flight Center. Accessed February 10th, 2019. https://swift.gsfc.nasa.gov/about_swift/

[4] J. L. Junkins and J. D. Turner, "Optimal Continuous Torque Attitude Maneuvers," *AIAA Journal of Guidance and Control,* Vol. 3, 1980, pp. 210–217.

[5] K. D. Bilimoria and B.Wie, "Time-optimal three-axis reorientation of a rigid spacecraft," *Journal of Guidance, Control, and Dynamics,* Vol. 16, No. 3, 1993, pp. 446–452.

[6] T. R. Kane, P. W. Likins and D. A. Levinson, *Spacecraft Dynamics*, The Internet-First University Press, New York, 2005.

[7] P. Ghosh and B. A. Conway, "A direct method for trajectory optimization using the particle swarm approach", *Advances in the Astronautical Sciences*, Vol. 140, 2011 pp.775–794.

[8] D. Spiller, L. Ansalone and F. Curti, "Particle Swarm Optimization for Time-Optimal Spacecraft Reorientation with Keep-Out Cones," *Journal of Guidance Control and Dynamics*, Vol. 39, No. 2, 2015 pp.1–14.

[9] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, Vol. 4 pp. 1942–1948.

[10] M. Potani and B. A. Conway, "Particle Swarm Optimization Applied to Space Trajectories", Journal of Guidance, Control, and Dynamics, Vol. 33, No. 5 (2010), pp. 1429–1441.

[11] G. B. Arfken and H. J. Weber, *Mathematical Methods for Physicists, 6th Edition*, Academic Press, 2005.

[12] "Chebyshev polynomials of the first kind." From MathWorks. Accessed February 10th, 2019. https://www.mathworks.com/help/symbolic/chebyshevt.html

[13] E. W. Weisstein "Least Squares Fitting–Polynomial." From MathWorld–A Wolfram Web Resource. Accessed February 10th, 2019. http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html

[14] K. Basu and R. G. Melton, "Time-optimal reorientation via inverse dynamics: A quaternion and particle swarm formulation," *Advances in the Astronautical Sciences,* Vol. 156, 2016 pp. 1631–1646.

[15] R. G. Melton, "Hybrid methods for determining time-optimal, constrained spacecraft reorientation maneuvers," *Acta Astronautica*, Vol. 94, No. 1, 2014 pp.294–301.

[16] R. G. Melton, "Constrained time-optimal slewing maneuvers for rigid spacecraft," *Advances in the Astronautical Sciences,* Vol. 135, 2010 pp. 106–126.