2020-09

A MACHINE LEARNING APPROACH TO ENABLE MISSION PLANNING OF TIME-OPTIMAL ATTITUDE MANEUVERS

Smith, Reed R., Jr.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A MACHINE LEARNING APPROACH TO ENABLE MISSION PLANNING OF TIME-OPTIMAL ATTITUDE MANEUVERS

by

Reed Ransom Smith

September 2020

Co-Advisors: Mark Karpenko
Brian M. Wade

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
A MACHINE LEARNING APPROACH TO ENABLE MISSION PLANNING OF TIME-OPTIMAL ATTITUDE MANEUVERS

5. FUNDING NUMBERS

6. AUTHOR(S) Reed Ransom Smith

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

8. PERFORMING ORGANIZATION REPORT NUMBER

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

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

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

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

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Time-optimal spacecraft rotations have been developed and implemented on orbiting spacecraft, highlighting opportunities for improving slew performance. Double-digit reductions in the time required to slew from one attitude to another have been demonstrated. However, the ability to perform mission planning to make use of minimum time slewing maneuvers is largely precluded by the need to compute a numerical solution to find a single minimum time maneuver control trajectory. Machine learning approaches can eliminate the need to generate problem solutions by approximating time-optimal maneuver times with sufficient accuracy for planning using only the initial and final attitude requirements. The advantages of time-optimal spacecraft maneuvers, a planning construct for evaluating legacy and machine learning maneuver time generators, and the machine learning processes that enable this approach are outlined. Compared to legacy planning techniques, time-optimal slew approximations yield target collection increases of 3% to 24% for an example planning framework.

14. SUBJECT TERMS
optimal control, trajectory optimization, genetic algorithm, machine learning, supervised learning, neural network, linear regression, path planning, trajectory planning, optimal maneuvers, time-optimal rotations, spacecraft control, mission planning, remote sensing, remote sensing planning

15. NUMBER OF PAGES
171

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A MACHINE LEARNING APPROACH TO ENABLE MISSION PLANNING OF TIME-OPTIMAL ATTITUDE MANEUVERS

Reed Ransom Smith
Lieutenant Commander, United States Navy
BA, Duke University, 2008
M, Systems Analysis, Naval Postgraduate School, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ASTRONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2020

Approved by: Mark Karpenko
Co-Advisor

Brian M. Wade
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Time-optimal spacecraft rotations have been developed and implemented on orbiting spacecraft, highlighting opportunities for improving slew performance. Double-digit reductions in the time required to slew from one attitude to another have been demonstrated. However, the ability to perform mission planning to make use of minimum time slewing maneuvers is largely precluded by the need to compute a numerical solution to find a single minimum time maneuver control trajectory. Machine learning approaches can eliminate the need to generate problem solutions by approximating time-optimal maneuver times with sufficient accuracy for planning using only the initial and final attitude requirements. The advantages of time-optimal spacecraft maneuvers, a planning construct for evaluating legacy and machine learning maneuver time generators, and the machine learning processes that enable this approach are outlined. Compared to legacy planning techniques, time-optimal slew approximations yield target collection increases of 3% to 24% for an example planning framework.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

DCM	direction cosine matrix
DoE	Design of Experiments
DU	distance units
FLOP	floating-point operation
GA	genetic algorithm
GCI	Geocentric Inertial
LVLH	Local Vertical/Local Horizontal
RSIP	Remote Sensing Iterative Planner
TRACE	Transition Region and Coronal Explorer
TSP	Traveling Salesman Problem
TU	time units
VU	value units

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to express my gratitude for the considerable direction and support provided to me by my advisors, Professor Mark Karpenko and Professor Brian Wade. Professor Karpenko's extensive experience and ingenuity in this area offered me the opportunity to explore a critical and timely topic that can directly improve the efficiency of space systems. Professor Wade has been exceedingly generous with his time and expertise, which has introduced me and many of my colleagues to machine learning as a discipline, from which I expect to benefit for many years.

I would also like to thank my colleague and classmate Lieutenant Benjamin Diehl for his assistance and collaboration with the development of our example spacecraft's kinematics and graphics.

I have benefited immensely from the instruction and collaboration here at the Naval Postgraduate School, and for that, I am grateful.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Remote sensing from space provides a useful service to a growing number of customers, however the barriers to entry remain high. Satellite, launch, and mission operations costs are significant while the industry becomes increasingly crowded. To excel, satellite operators are well-incentivized to pursue increases in efficiency in their operations. This could mean more bandwidth, higher platform availability, or more sensing products. Increasing cost efficiency could be the result of taking pictures more quickly or decreasing overhead time where sensing is not occurring. This thesis focuses on reducing unproductive time by incorporating minimum time maneuvers into operations.

1.1 Remote Sensing Planning

Slewing a sensor from one target to the next target with traditional optics systems requires a sensor attitude change, assuming the field of view is fixed to a boresight axis. For spacecraft, this process is frequently accomplished by fixing the boresight to an axis of the vehicle and rotating the entire vehicle until the desired alignment is achieved.

An intuitive method to slew a spacecraft from one attitude to another as quickly as possible is to rotate about the axis that is perpendicular to both the starting and final boresight vectors. This effectively traces a direct arc with the sensor's boresight across a flat target surface; an example of this maneuver is depicted in Figure 1.1. However, such a maneuver (an *eigenaxis* maneuver) is not the fastest way to accomplish the rotation [1].

Real spacecraft have non-uniform mass properties and can maneuver more quickly around certain axes of rotation and less quickly around others. These characteristics are exploited by dynamic optimization to produce maneuvers that are more time-efficient. This type of maneuver was first demonstrated on the NASA Transition Region and Coronal Explorer (TRACE) spacecraft on August 10, 2010 [2]. To illustrate the potential gain, the same sample maneuver shown in Figure 1.1 was accomplished using a time-optimal control trajectory, depicted in Figure 1.2. The time-optimal maneuver is accomplished in 29.6 seconds, which is 16.6% faster than the original eigenaxis maneuver.

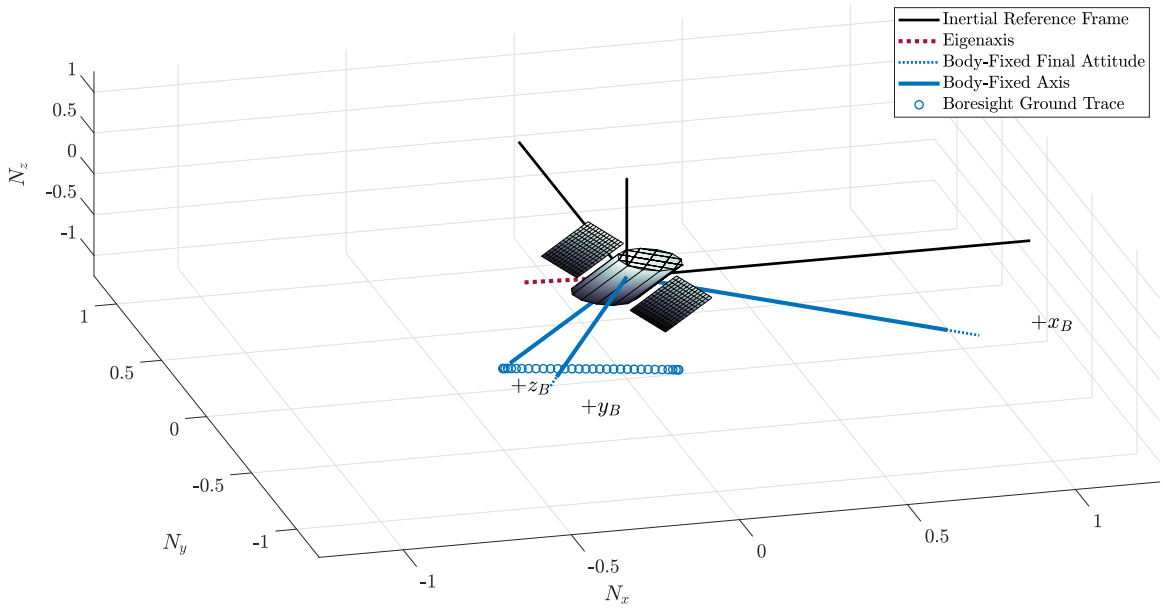


Figure 1.1. Eigenaxis rotation, duration = 35.5 sec.

While this seems like an obvious technique to exploit, developing and implementing a time-optimal slewing maneuver is only half the battle. To be useful, time savings from optimal maneuvers must be incorporated into planning algorithms, which presents a problem in the current state of practice, as spacecraft slew plans can include hundreds or thousands of attitudes and slews. Although computing a single time-optimal slew can be done quite quickly (tens of seconds), planning algorithms must solve a combinatorial optimization to obtain a high value plan. In such a system, the number of optimal slews that would need to be determined to produce a single mission plan would be on the order of millions or more because of the iterative nature of the planning algorithm. Thus, incorporating the solution of time-optimal maneuvers as part of the planning process is impractical for an operational system.

The TRACE demonstration produced maneuver time reductions ranging from 5% to 21% [2]. However, the ability to develop a realistic mission plan based on these capabilities remains out of reach because of the scale of the computations required to combine planning algorithms and optimal slewing maneuver calculations.

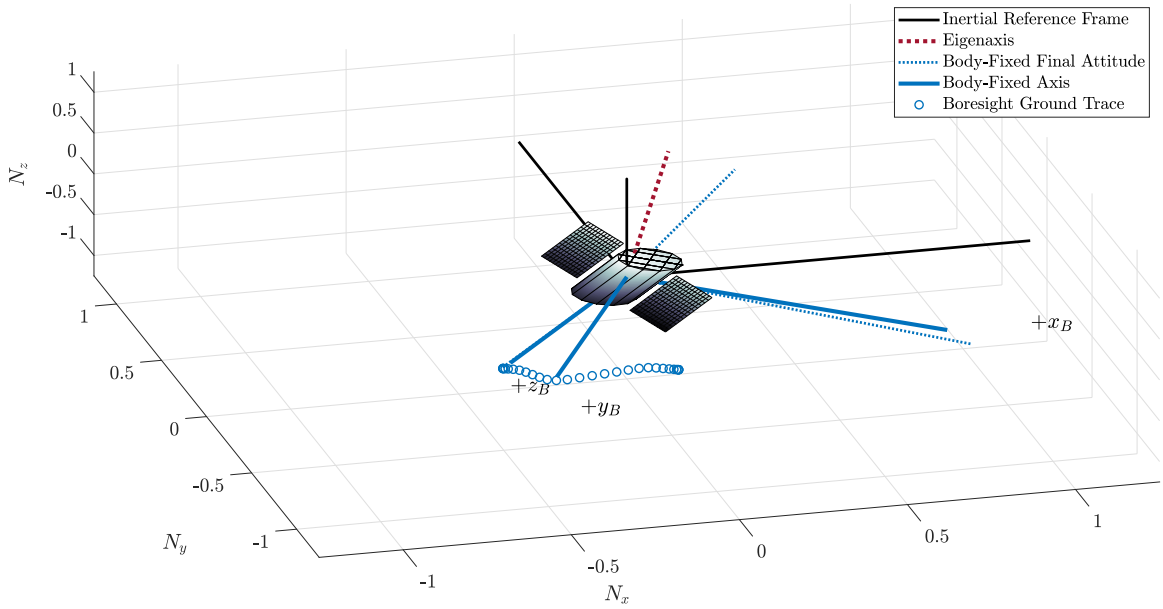


Figure 1.2. Time-optimal rotation, duration = 29.6 sec.

Machine learning can bridge this gap today. While the maneuvers discussed for the TRACE mission required high fidelity, flight quality solutions, a planning algorithm need only evaluate the cost of accomplishing a given maneuver. In the case of an imaging mission planner, a *fitness function* is used to evaluate how well a given parameter accomplishes its objective. For mission planning, the fitness of a given maneuver sequence will be evaluated by summing the individual maneuver times; faster sequences that accomplish the same purpose are favored.

To iterate a mission plan, the precise engineering solution for a time-optimal slew need not be available. The algorithm must provide maneuver times with sufficient accuracy for the fitness function to correctly distinguish the relative fitness of various candidate maneuvers.

Many mission planning routines use legacy maneuvers that are inefficient to fly but easy to model. If the optimal slew time could be efficiently approximated by a machine learning tool, then that approximated time may suffice to generate mission plans that incorporate efficiencies gained from optimal slewing maneuvers.

Machine learning models are lightweight (i.e., fast running) approximations of computationally intensive engineering models. These are called *metamodels* and may be the mechanism

required to sensibly combine iterative planners and optimal maneuvers.

Assuming such a machine learning model could be created, what kind of accuracy would be required for a useful product? If TRACE demonstrated maneuver time reductions of 5% to 21%, it is reasonable to infer that accuracy to within the low single digits would be sufficient to implement a mission plan that incorporates optimal slews and yields overall greater efficiency for a remote sensing enterprise, even if slight inaccuracies are introduced in the planning phase by the machine learning metamodel. Any remaining discrepancies between the approximated mission plan and flight-fidelity mission plan would be trivial compared to the efficiencies gained.

Once a mission plan is generated for time-optimal slews, the full engineering model provides the accuracy required to generate a flight-fidelity solution, without running computationally intensive calculations during the iterative planning process. Flight-quality solutions can be generated by the engineering model after the fact, with the broad mission plan in hand. Even with slight inaccuracies in the approximated slew times when compared to the full-fidelity engineering model slew times, the overall mission plan time is likely to approach the true optimal time as the number of included slews increases.

To determine whether or not machine learning models provide a sufficiently accurate optimal slew time, a common framework for comparison is required. Once this framework is created, mission plans can be generated using legacy techniques as well as approximated optimal slew techniques. Those approximated mission plans can then be evaluated against flight-fidelity mission plans for accuracy and potential time savings.

1.2 Current State of Practice

Dimensionality associated with these problems leads operators to simple, suboptimal planning routines augmented heuristically by analysts to improve performance or remove obviously detrimental results. "Traditional planning and scheduling relies on humans to pre-program a set of actions to accomplish a pre-planned task or series of tasks" [3]. These human-in-the-loop systems would naturally tend to slow or delay the planning process, and would preclude evaluating hundreds, thousands, or more potential options that are required to find an optimal or near-optimal solution to the problem. Frequently, there are insufficient

resources to collect all potential value for a given problem. For remote sensing missions, this simply means that there is not enough time and assets to collect all the imagery desired. This type of problem is called an *over-subscription* problem. One approach to over-subscription problems is the *greedy* approach, in which the planner simply schedules the highest priority collection first, and proceeds down the list in priority order. "If the observation being considered is still possible, it is added to the schedule; otherwise it is discarded. This approach can work reasonably well for problems in which the cost of achieving an objective does not depend on the order in which the objectives are achieved" [4]. This, of course, is not the case for an overhead collection vehicle attempting to find the most efficient path through a list of possible objects of varying value.

1.3 Enabling Greater Sensing Efficiency

The task at hand then is to develop a planning method that incorporates optimal slewing maneuvers into a mission plan while being executable with computing resources available today. The flow for such a planner is depicted in Figure 1.3. Currently, over-subscription planners are widely available [5] and engineering models that provide time-optimal slew control trajectories have been demonstrated [2]. The "Slew Time" function block that provides slew times to the "Path Planner" is currently being accomplished using legacy techniques. Two of those legacy techniques are the *taxicab* method and the *eigenaxis* method, both of which will be discussed further in Chapter 2.

Augmenting the Slew Time function block with the capability to provide optimal slew times can lead to significant gains in mission efficiency. However, given the limits of modern processors, instead of attempting to determine the precise optimal slew time for a candidate maneuver (which would be required millions of times for a relatively simple mission plan), a machine learning metamodel can be used to approximate the optimal slew time, which allows the iterative planner to quickly determine whether that maneuver should be saved for further consideration or discarded as relatively inefficient.

For early planning, control trajectories and maneuver details are irrelevant—only the maneuver time is required. If this metamodel provides approximate maneuver times with sufficient accuracy to generate feasible mission plans, then the demonstrated 5% to 21% gains in efficiency for a single maneuver can be extended to an enterprise scale.

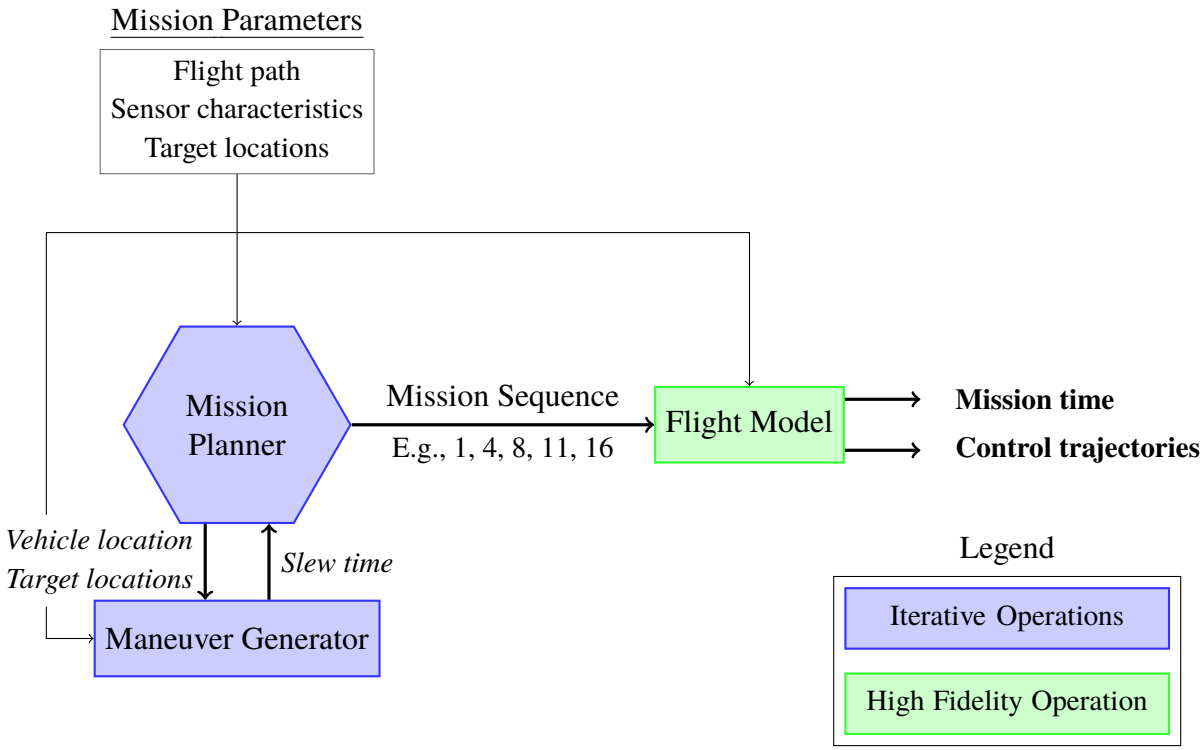


Figure 1.3. Over-subscription planning flow diagram.

1.4 Thesis Outline

A method for incorporating time-optimal rotations into the mission planning process will be developed in the following chapters. Chapter 2 will discuss the mechanics of remote sensing from space, including the limits of legacy techniques. Chapter 3 will describe the use of a genetic algorithm to solve the over-subscription problem for a remote sensing mission plan with targets of varying values. Chapter 4 will address the specifics of producing a mathematically optimal maneuver, which is used to generate labeled data for training machine learning models and for generating high fidelity control trajectories. Finally, Chapter 5 will describe the techniques used to generate fast-running machine learning metamodels to approximate optimal slew times at speeds that are practical for inclusion in an iterative mission planner.

CHAPTER 2:
Kinematics of Sensing from Space

This chapter will introduce and discuss the orbital mechanics and attitude rotations required to orient a spacecraft for remote sensing operations. First, the orbital mechanics required to predict a spacecraft's velocity relative to Earth as a function of orbital altitude are introduced. Next, multiple methods for determining a spacecraft's attitude will be discussed, including direction cosine matrices, Euler angles, and quaternions. Methods to describe kinematic rotations of a spacecraft are identified, including basic methods for determining the kinematics required to achieve a given orientation relative to Earth. Finally, the state space kinematics model used in the development of this planner will be defined for future inclusion in a mission planning program.

2.1 Orbital Mechanics

Newton's first law of motion states that a body will remain in its state of rest or motion until it is acted upon by an outside force to change that state. With knowledge of planetary motion and the existence of satellites in orbit around the Earth, it is apparent that some external force is acting upon orbiting bodies.

Law of Gravity The magnitude of that force is described by Newton's Law of Gravity

$$f = \frac{Gm_1m_2}{r^2} \tag{2.1}$$

where, G is the universal gravitational constant, m_1 and m_2 are the masses of the two objects (in our case, the mass of Earth and the mass of the orbiting satellite), and r is the radius between the two points. The gravitational acceleration of an object with respect to the Earth is then

$$g = \frac{GM_\oplus}{r^2} \tag{2.2}$$

where, $M_\oplus$ is the mass of the Earth. Equation (2.2) describes the acceleration acting radially on an object in Earth's orbit between that object and Earth. This acceleration varies as a function of the radius from the point mass of Earth to the orbiting object. With this

information, it is apparent that the speed with which an object accelerates *towards* the body that it is orbiting is a function only of its distance from that body, assuming that the mass of the orbited body is fixed.

Classical Orbital Elements The radius of an orbit need not be constant, in fact constant radius circular orbits are a special case of elliptical orbits. Very few orbits are truly circular, however circular orbits are useful and benefit from some simplified dynamics. From these basic principles, six classical orbital elements can be derived. The orbital elements completely describe any orbit and are shown in Figure 2.1. Further information can be found in Wie's textbook [6]. The six elements are

a = semimajor 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.

These elements provide all the information required to determine the radius and velocity of an Earth orbiting vehicle. Altitude above the Earth's surface can easily be inferred from the average radius to the center of the Earth.

2.1.1 Elliptical Orbits

The two cases of closed orbits are elliptical orbits and circular orbits, of which elliptical orbits are more generic. Circular orbits are a subset of elliptical orbits with eccentricity $e = 0$. Before considering circular orbits, consider the information necessary to obtain the altitude and velocity of a spacecraft at a given point in its elliptical orbit.

One method to define the position of the spacecraft is its *true anomaly*, θ, which is the angle between the argument of perigee and the current position of the spacecraft. The true anomaly vertex is coincident with the main focus of the ellipse (which is the position of the planet being orbited). Assuming that a spacecraft's true anomaly and its orbital elements are known, the spacecraft's radius at a given true anomaly can then be obtained by

$$r = \frac{a(1 - e^2)}{1 + e\cos\theta}. \tag{2.3}$$

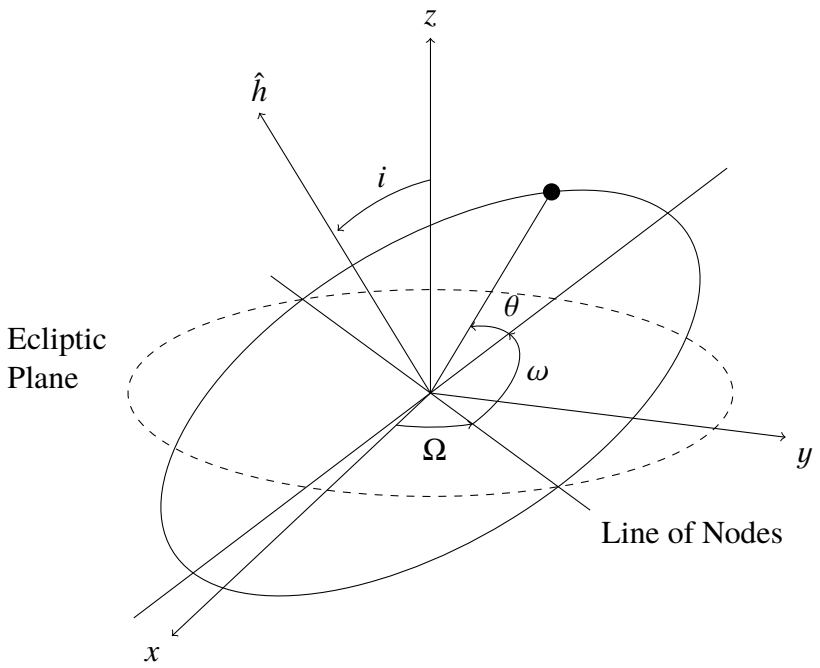


Figure 2.1. Selected classical orbital elements.

To determine the spacecraft's altitude above the surface, z, simply subtract the radius of the Earth.

$$z = r - R_{\oplus} \tag{2.4}$$

The magnitude of the spacecraft's velocity in an elliptical orbit is a function of the distance between the two objects. Where $\mu = GM_{\oplus}$, the velocity of Earth orbiting satellites is determined by

$$v = \sqrt{\mu \left(\frac{2}{r} - \frac{1}{a} \right)}. \tag{2.5}$$

With these relationships in hand, a known orbit's altitude and velocity information can be determined for a mission planning algorithm. However, accounting for the changing altitudes and velocities associated with elliptical orbits adds unnecessary complications for a concept demonstration. Circular orbits provide a simpler mission framework to exercise potential planning algorithm improvements.

2.1.2 Circular Orbits

For a spacecraft in a circular orbit, altitude and velocity determination is somewhat simplified. For a circular orbit, $e = 0$. The formula for the radius at a given point then becomes

$$r = a. \tag{2.6}$$

The constant orbital radius is a helpful new feature of circular orbits, and assuming a spherical Earth (which is not true, but is an acceptable simplification for this purpose), the spacecraft's altitude is now constant.

Now that $r = a$ for circular orbits, orbital velocity becomes

$$v = \sqrt{\frac{\mu}{a}}. \tag{2.7}$$

The length of the semimajor axis and the standard gravitational parameter μ are constant for a given orbit, therefore a constant velocity is also a feature of circular orbits.

Equations (2.6) and (2.7) highlight the advantages of circular orbits where they can be used. Many remote sensing spacecraft are in circular or nearly circular orbits to take advantage of fixed distance and velocity parameters for the sensor. This can be advantageous for uniformity of perspective or consistency in the resolution of an imaging product.

2.1.3 Flat Earth Model

For circular orbits in which vehicle altitude and velocity is constant, spacecraft position relative to Earth can be effectively modeled over sufficiently small areas using a flat Earth perspective for low earth orbit. This assumption significantly simplifies the planning problem by allowing the use of polar coordinates to be neglected. This approach was chosen for this thesis because it meets all the required attributes for determining relative geometry of a spacecraft and sensing targets on the surface.

2.2 Rotational Kinematics

In addition to a spacecraft's position relative to a target, information about the spacecraft's attitude compared to a known reference frame is essential for modeling and performing

remote sensing operations. There are multiple reference frames available for considering the attitude of orbiting vehicles. For Earth-centered applications, the most encompassing of these frames is the *inertial reference frame*, which is Earth-centered. The inertial reference frame is sometimes referred to as Geocentric Inertial (GCI). Its z-axis is defined as the Earth's pole, and the x-axis is the direction from Earth to the Sun on the first day of Spring, also referred to as the *vernal equinox*, Υ. While this is not truly an inertial frame thanks to the precession of the Earth's axis around the ecliptic pole, it is sufficient for most practical purposes.

While the GCI provides a useful reference frame for orbit problems, the *local horizontal* or Local Vertical/Local Horizontal (LVLH) reference frame is useful to consider a satellite's attitude relative to the Earth's surface. This reference frame fixes the z-axis to the *nadir*, or directly below the spacecraft. The x-axis is aligned in the direction of the velocity vector, perpendicular to nadir. This reference frame is sometimes referred to in terms of *roll*, *pitch*, and *yaw* for rotations around the x, y, and z axes, respectively.

"Typically, we choose the Earth's center as the origin for problems in orbit analysis or geometry on the Earth's surface and the spacecraft's position for problems concerning the apparent position and motion of objects as seen from the spacecraft or analysis of payload observations" [7].

2.2.1 Transformation Matrices

With multiple reference frames ready for use and an agile spacecraft ready to maneuver, a method of mathematically describing a vehicle's rotation relative to one or all of these frames is necessary. The mathematical mechanism that effects this change from one coordinate system to another is a *transformation matrix*.

Generally, one right-hand orthogonal reference frame can be described in terms of another right-hand orthogonal reference frame using the *direction cosines*, which are the cosines of the angles between the vector and the coordinate axes used to describe it.

Figure 2.2 shows a reference frame B with basis vectors $\{\vec{b}_1, \vec{b}_2, \vec{b}_3\}$ that can be described in terms of a reference frame A with basis vectors $\{\vec{a}_1, \vec{a}_2, \vec{a}_3\}$ by using the direction cosines

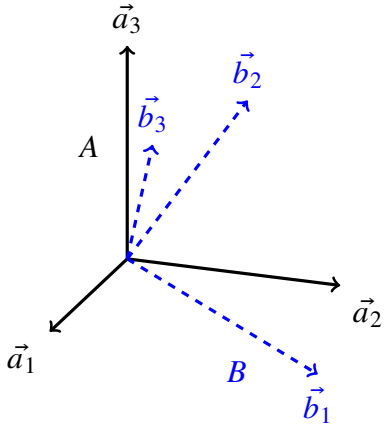


Figure 2.2. Two reference frames, A and B.

$C_{ij} = \vec{b}_i \cdot \vec{a}_j$, taking advantage of the inner product to derive the following relationships.

$$\vec{b}_1 = C_{11}\vec{a}_1 + C_{12}\vec{a}_2 + C_{13}\vec{a}_3 \tag{2.8a}$$

$$\vec{b}_2 = C_{21}\vec{a}_1 + C_{22}\vec{a}_2 + C_{23}\vec{a}_3 \tag{2.8b}$$

$$\vec{b}_3 = C_{31}\vec{a}_1 + C_{32}\vec{a}_2 + C_{33}\vec{a}_3 \tag{2.8c}$$

The direction cosines can also be presented in matrix form.

$$\begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vec{b}_3 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} = \mathbf{C}^{B/A} \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} \tag{2.9}$$

When these operations are applied in sequence, the transformation matrix that results is a *direction cosine matrix* (DCM), $\left[C_{ij}\right]$. $\left[C_{ij}\right]$ is similar to the form $\mathbf{C}^{B/A}$ which more specifically indicates the DCM for a transformation to B from A.

$$\left[\mathbf{C}_{ij}\right] = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{2.10}$$

Direction cosine matrices are orthonormal, meaning that $\left[C_{ij}\right]^{-1} = \left[C_{ij}\right]^T$, which allows

for the following inverse operation.

$$\begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} = \begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix} \begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vec{b}_3 \end{bmatrix} \tag{2.11}$$

Now, an arbitrary vector $\vec{H}$ can be expressed in terms of reference frame A and B.

$$\begin{aligned} \vec{H} &= H_1\vec{a}_1 + H_2\vec{a}_2 + H_3\vec{a}_3 \\ &= H'_1\vec{b}_1 + H'_2\vec{b}_2 + H'_3\vec{b}_3 \end{aligned} \tag{2.12}$$

Since $\vec{H}$ is known and expressed in terms of A, Wie expresses the same vector using reference frame B as $\vec{H}'$ [6].

$$H'_1 \equiv \vec{b}_1 \cdot \vec{H} = \vec{b}_1 \cdot (H_1\vec{a}_1 + H_2\vec{a}_2 + H_3\vec{a}_3) \tag{2.13a}$$

$$H'_2 \equiv \vec{b}_2 \cdot \vec{H} = \vec{b}_2 \cdot (H_1\vec{a}_1 + H_2\vec{a}_2 + H_3\vec{a}_3) \tag{2.13b}$$

$$H'_3 \equiv \vec{b}_3 \cdot \vec{H} = \vec{b}_3 \cdot (H_1\vec{a}_1 + H_2\vec{a}_2 + H_3\vec{a}_3) \tag{2.13c}$$

Algebraic reorganization yields the familiar direction cosine matrix, $\mathbf{C}^{B/A}$.

$$\begin{bmatrix} \vec{H}'_1 \\ \vec{H}'_2 \\ \vec{H}'_3 \end{bmatrix} = \begin{bmatrix} \vec{b}_1 \cdot \vec{a}_1 & \vec{b}_1 \cdot \vec{a}_2 & \vec{b}_1 \cdot \vec{a}_3 \\ \vec{b}_2 \cdot \vec{a}_1 & \vec{b}_2 \cdot \vec{a}_2 & \vec{b}_2 \cdot \vec{a}_3 \\ \vec{b}_3 \cdot \vec{a}_1 & \vec{b}_3 \cdot \vec{a}_2 & \vec{b}_3 \cdot \vec{a}_3 \end{bmatrix} \begin{bmatrix} \vec{H}_1 \\ \vec{H}_2 \\ \vec{H}_3 \end{bmatrix} = \mathbf{C}^{B/A} \begin{bmatrix} \vec{H}_1 \\ \vec{H}_2 \\ \vec{H}_3 \end{bmatrix} \tag{2.14}$$

The same DCM that transforms orthogonal basis vectors of different reference frames also transforms the representation of an arbitrary vector from one frame to another.

2.2.2 Euler Angles

Consider three special case DCMs that result from transforming a matrix by rotating it about a single axis. Three simplified and useful DCM cases describe the rotations about

the first, second, and third axes of reference frame A.

$$\mathbf{C}_1(\theta_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \tag{2.15a}$$

$$\mathbf{C}_2(\theta_2) = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \tag{2.15b}$$

$$\mathbf{C}_3(\theta_3) = \begin{bmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.15c}$$

One method for orienting a body is to rotate it in succession about the axes of the body-fixed reference frame; an example of such a rotation is provided in Figure 2.3. Wie calls this a *body-axis rotation* and explains that the first rotation can be about any axis, the second rotation is about either of the unrotated axes, and the third rotation is about either axis not used for the second rotation.

Consider a rotation from reference frame A to reference frame B using three successive rotations, first about the axis $\vec{a}_3$, then about $\vec{a}_2$, and finally about $\vec{a}_1$ (shown in Figure 2.4). Wie symbolically denotes this operation by $\mathbf{C}_1(\theta_1) \leftarrow \mathbf{C}_2(\theta_2) \leftarrow \mathbf{C}_3(\theta_3)$.

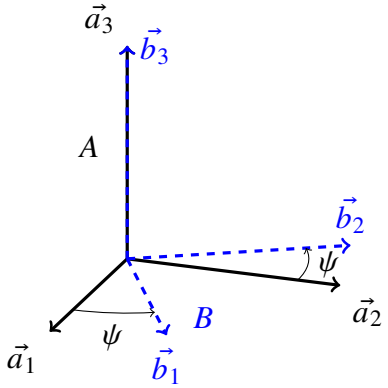


Figure 2.3. Principal axis rotation around $\vec{a}_3$.

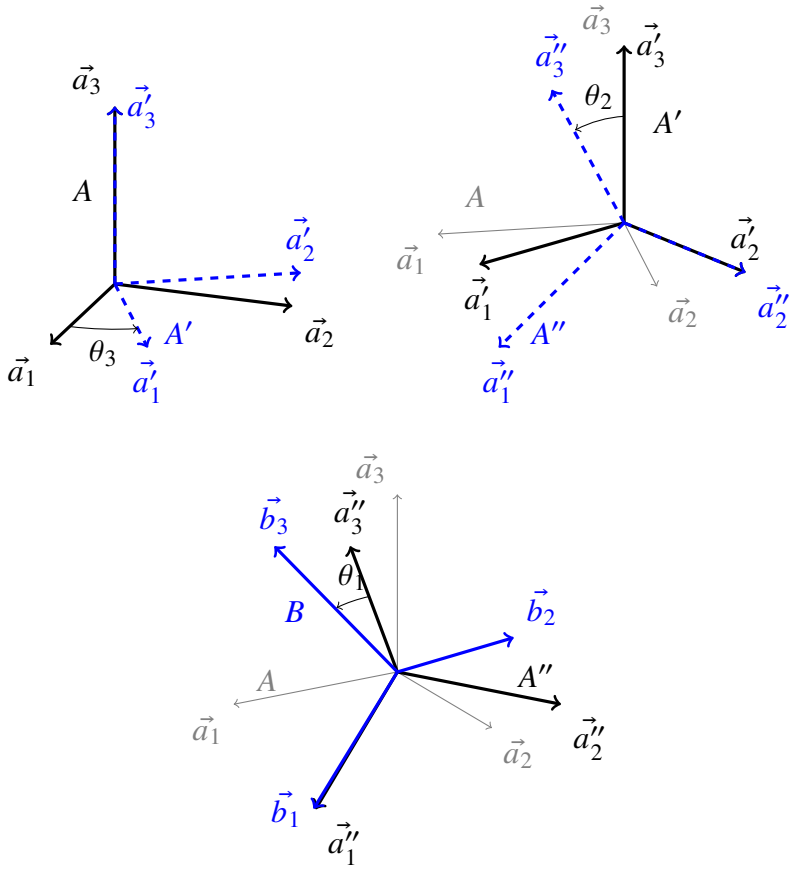


Figure 2.4. Body axis rotations, 3-2-1.

The intermediate rotations described by this symbolic notation are

$$\mathbf{C}_3(\theta_3) : A' \leftarrow A \tag{2.16a}$$

$$\mathbf{C}_2(\theta_2) : A'' \leftarrow A' \tag{2.16b}$$

$$\mathbf{C}_1(\theta_1) : B \leftarrow A''. \tag{2.16c}$$

The elements of each of these rotations are:

$$\begin{bmatrix} \vec{a}'_1 \\ \vec{a}'_2 \\ \vec{a}'_3 \end{bmatrix} = \begin{bmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} = \mathbf{C}_3(\theta_3) \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} \tag{2.17a}$$

$$\begin{bmatrix} \vec{a}''_1 \\ \vec{a}''_2 \\ \vec{a}''_3 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} \vec{a}'_1 \\ \vec{a}'_2 \\ \vec{a}'_3 \end{bmatrix} = \mathbf{C}_2(\theta_2) \begin{bmatrix} \vec{a}'_1 \\ \vec{a}'_2 \\ \vec{a}'_3 \end{bmatrix} \tag{2.17b}$$

$$\begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vec{b}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \vec{a}''_1 \\ \vec{a}''_2 \\ \vec{a}''_3 \end{bmatrix} = \mathbf{C}_1(\theta_1) \begin{bmatrix} \vec{a}''_1 \\ \vec{a}''_2 \\ \vec{a}''_3 \end{bmatrix} \tag{2.17c}$$

By algebraically combining these equations, the condensed mathematical transformation is expressed:

$$\begin{bmatrix} \vec{b}_1 \\ \vec{b}_2 \\ \vec{b}_3 \end{bmatrix} = \mathbf{C}_1(\theta_1)\mathbf{C}_2(\theta_2)\mathbf{C}_3(\theta_3) \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix} \tag{2.18}$$

And therefore,

$$\mathbf{C}^{B/A} = \mathbf{C}_1(\theta_1)\mathbf{C}_2(\theta_2)\mathbf{C}_3(\theta_3). \tag{2.19}$$

The three defining angles for this process, $\theta_1, \theta_2, \theta_3$, are called *Euler angles*. "In general, Euler angles have an advantage over direction cosines in that three Euler angles determine a unique orientation, although there is no unique set of Euler angles for a given orientation" [6].

2.2.3 Eigenaxis Rotation

Direction cosine matrices are particularly valuable for transforming a vector from one reference frame to another; however, to describe the rotation between two vectors (both within the same reference frame), Euler's *eigenaxis rotation* is a useful tool. This method makes use of an *eigenaxis*, $\vec{e}$ or $\hat{\mathbf{e}}$, which is a unit vector that is the axis of rotation, fixed both in the body frame and in the inertial frame. The angle of the rotation about the eigenaxis, ψ, is the *eigenangle*. An example eigenaxis rotation is depicted in Figure 2.5.

To calculate the axis around which a body must rotate for a given maneuver without changing reference frames, the cross product can be exploited to find the vector normal to the plane

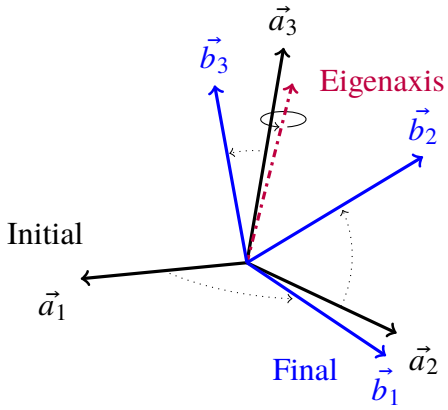


Figure 2.5. Example eigenaxis rotation.

that contains the current vector (e.g., **a**) and the desired vector (e.g., **b**). This will yield the eigenaxis ($\hat{\mathbf{e}}$), rotating about which will provide a direct path for the maneuver.

$$\hat{\mathbf{e}} = \mathbf{a} \times \mathbf{b} \tag{2.20}$$

With the axis of rotation obtained, the angle of rotation must also be calculated. This eigenangle is obtained by exploiting the inner product of the two attitudes. The eigenangle is indicated by ψ and the magnitude of a vector is indicated by $\| \|$.

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\psi \tag{2.21a}$$

$$\psi = \arccos \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|} \tag{2.21b}$$

In this manner, both the eigenaxis and eigenangle can be obtained from knowledge of the two original vectors.

2.2.4 Quaternions

Quaternions, invented by William Hamilton in the 19th century, are commonly used to describe spacecraft rotations [6]. Quaternions are defined as

$$\mathbf{q} = \begin{bmatrix} e_1 \sin(\psi/2) \\ e_2 \sin(\psi/2) \\ e_3 \sin(\psi/2) \\ \cos(\psi/2) \end{bmatrix}. \tag{2.22}$$

One parameterization of the direction cosine matrix in terms of quaternions is

$$\mathbf{C}^{B/A} = \mathbf{C}(\mathbf{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix}. \tag{2.23}$$

Quaternions have some advantageous properties. The 2-norm of all quaternions is 1 (i.e., $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$). Additionally, they do not have geometric singularities as Euler angles do, which makes them frequently preferable for aerospace applications. "Moreover, quaternions are well suited for onboard real-time computation because only products and no trigonometric relations exist in the quaternion kinematic differential equations. Thus, spacecraft orientation is now commonly described in terms of quaternions" [6].

2.2.5 GCI Transformation

Wie describes the process of representing a vector $\vec{r}$ with known GCI coordinates $X\vec{I} + Y\vec{J} + Z\vec{K}$ into a reference frame that is fixed to the orbit plane with its origin coincident with the body being orbited. Such a reference frame is called the *perifocal* frame (x, y, z) and has basis vectors $(\vec{i}, \vec{j}, \vec{k})$ [6]. To accomplish this, a series of three successive rotations are used. This method uses a 3–1–3 method, $\mathbf{C}_3(\omega) \leftarrow \mathbf{C}_1(i) \leftarrow \mathbf{C}_3(\Omega)$, taking advantage of the orbital elements right ascension (Ω), inclination (i), and argument of perigee (ω).

Wie defines the vector using intermediate rotations

$$\vec{r} = X'\vec{I}' + Y'\vec{J}' + Z'\vec{K}' = X''\vec{I}'' + Y''\vec{J}'' + Z''\vec{K}'' \tag{2.24}$$

where, single prime ($'$) indicates components of the position vector in the first intermediate reference frame and double prime ($''$) indicates the components in the second intermediate reference frame.

Using the nomenclature from Figure 2.1, Wie computes the three successive rotations in the following form.

$$\begin{bmatrix} \vec{I}' \\ \vec{J}' \\ \vec{K}' \end{bmatrix} = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{I} \\ \vec{J} \\ \vec{K} \end{bmatrix} \tag{2.25a}$$

$$\begin{bmatrix} \vec{I}'' \\ \vec{J}'' \\ \vec{K}'' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix} \begin{bmatrix} \vec{I}' \\ \vec{J}' \\ \vec{K}' \end{bmatrix} \tag{2.25b}$$

$$\begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{k} \end{bmatrix} = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{I}'' \\ \vec{J}'' \\ \vec{K}'' \end{bmatrix} \tag{2.25c}$$

Here, the DCM is defined with the following elements.

$$\begin{aligned} C_{11} &= \cos\Omega\cos\omega - \sin\Omega\sin\omega\cos i \\ C_{12} &= \sin\Omega\cos\omega + \cos\Omega\sin\omega\cos i \\ C_{13} &= \sin\omega\sin i \\ C_{21} &= -\cos\Omega\sin\omega - \sin\Omega\cos\omega\cos i \\ C_{22} &= -\sin\Omega\sin\omega + \cos\Omega\cos\omega\cos i \\ C_{23} &= \cos\omega\sin i \\ C_{31} &= \sin\Omega\sin i \\ C_{32} &= -\cos\Omega\sin i \\ C_{33} &= \cos i \end{aligned}$$

And therefore the total transformation is described by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{2.26}$$

2.3 Spacecraft Rotations

Ross, Proulx, and Karpenko [8] have developed a state space model with an elementary kinematics representation that is a suitable framework to model attitude changes for a remote sensing mission planner operating over a flat Earth. Their state space model is reproduced in Figure 2.6.

2.3.1 Kinematic State Space Model

As a basis for a three-dimensional spacecraft attitude model, tilt (θ) and pan (ϕ) angles from the vehicle's body-fixed Cartesian coordinate system must be defined. Tilt and pan are defined as the angles from the body-fixed triad to the target $(\ell^1, \ell^2, 0)$ along the b_1 and b_2 axes respectively. This assumes a staring-type sensor where the yaw of the boresight is not relevant. Certain spacecraft sensors (e.g., pushbroom-type) must control yaw as well, which is neglected for this purpose. The principle of superposition provides the following basic relationship.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} + r_{\text{LOS}} = \begin{pmatrix} \ell^1 \\ \ell^2 \\ 0 \end{pmatrix} \tag{2.27}$$

Applying kinematic principles, individual relationships are derived.

$$x + r_{\text{LOS}} \sin\theta = \ell^1 \tag{2.28a}$$

$$y + r_{\text{LOS}} \sin\phi \cos\theta = \ell^2 \tag{2.28b}$$

$$z - r_{\text{LOS}} \cos\phi \cos\theta = 0 \tag{2.28c}$$

Algebraic rearrangement of these equations allows tilt and pan angles to be derived from

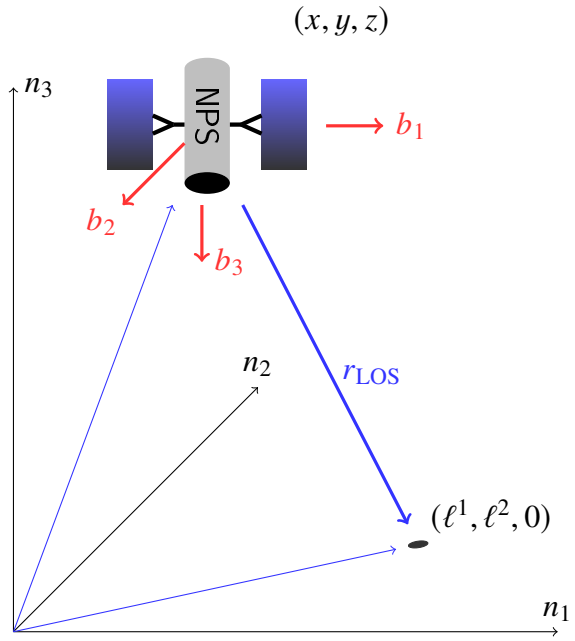


Figure 2.6. State space schematic for a remote sensing planner. Adapted from [8].

known target and vehicle locations in the inertial frame.

$$x + z\frac{\tan\theta}{\cos\phi} = \ell^1 \tag{2.29a}$$

$$y + z\tan\phi = \ell^2 \tag{2.29b}$$

2.3.2 Kinematic Slew Time Derivation

An individual rotation is assumed to occur according to a known acceleration rate (α) and a maximum slew rate ($\omega_{\max}$), beginning from rest, $\omega = 0\,\text{rad/s}$. The vehicle rotates over an angle, ψ.

There are two conditions that must be derived.

1. An acceleration-limited slew where coasting is not required.
2. A rate-limited slew where a coast phase at $\omega_{\max}$ is required.

Acceleration-Limited Slew

For a rest to rest maneuver, half of the maneuver will be accelerating and half will be decelerating. The maximum rotation for an acceleration limited maneuver occurs at the critical angle, ψ_{crit}, which corresponds to a peak angular velocity of ω_{max}, shown in Figure 2.7. The critical time, t_{crit}, is the midpoint of any acceleration-limited maneuver.

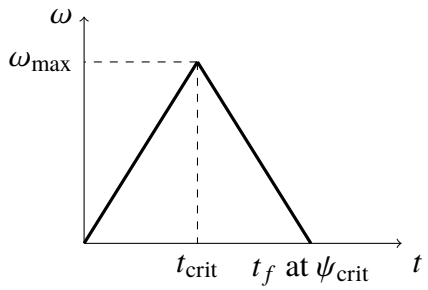


Figure 2.7. Acceleration-limited slew.

From rest, begin with the relationship

$$\psi = \frac{1}{2}\alpha t^2,$$

then, the rotation time can be determined.

$$\frac{1}{2}\psi = \frac{1}{2}\alpha t_{\text{crit}}^2 \tag{2.30a}$$

$$t_{\text{crit}} = \sqrt{\frac{\psi}{\alpha}} \tag{2.30b}$$

$$t_f = 2\sqrt{\frac{\psi}{\alpha}} = \sqrt{\frac{4\psi}{\alpha}} \tag{2.30c}$$

Rate-Limited Slew

In the event that $\psi > \psi_{\text{crit}}$, a coast period is required prior to deceleration (Figure 2.8).

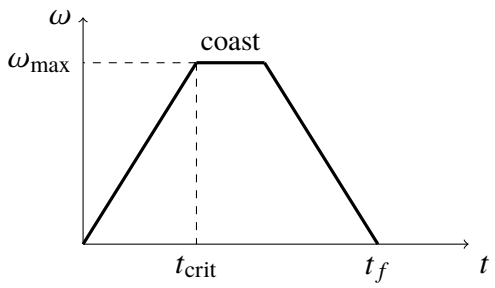


Figure 2.8. Rate-limited slew.

The total time required for the maneuver is the sum of the acceleration, coast, and deceleration phases, Equation (2.31).

$$t_f = 2t_{\text{crit}} + t_{\text{coast}} \tag{2.31}$$

$$t_{\text{coast}} = \frac{\psi - \psi_{\text{crit}}}{\omega_{\max}} \tag{2.32}$$

Apply the kinematic formulas relating angular acceleration and velocity to the critical conditions.

$$\omega_{\max} = \alpha t_{\text{crit}} \tag{2.33a}$$

$$t_{\text{crit}} = \frac{\omega_{\max}}{\alpha} \tag{2.33b}$$

Using equations (2.30a) and (2.33b), the critical angle can be derived from a vehicle's kinematic parameters.

$$\psi_{\text{crit}} = \alpha t_{\text{crit}}^2 \tag{2.34a}$$

$$\psi_{\text{crit}} = \alpha \left(\frac{\omega_{\max}}{\alpha}\right)^2 \tag{2.34b}$$

$$\psi_{\text{crit}} = \frac{\omega_{\max}^2}{\alpha} \tag{2.34c}$$

From equations (2.31), (2.32), (2.33b), and (2.34c), the total maneuver time can be obtained.

$$t_{\text{total}} = 2\frac{\omega_{\max}}{\alpha} + \frac{\psi - \psi_{\text{crit}}}{\omega_{\max}} \tag{2.35a}$$

$$t_{\text{total}} = 2\frac{\omega_{\max}}{\alpha} + \frac{\psi}{\omega_{\max}} - \frac{\psi_{\text{crit}}}{\omega_{\max}} \tag{2.35b}$$

$$t_{\text{total}} = 2\frac{\omega_{\max}}{\alpha} + \frac{\psi}{\omega_{\max}} - \frac{\frac{\omega_{\max}^2}{\alpha}}{\omega_{\max}} \tag{2.35c}$$

$$t_{\text{total}} = \frac{\omega_{\max}}{\alpha} + \frac{\psi}{\omega_{\max}} \tag{2.35d}$$

The resulting total time equation is a function of known elements and is used for computing slew times that are rate-limited. Kinematically determining slew times based on vehicle parameters will be used for two legacy planning techniques that serve as benchmarks for current practices in this concept demonstration.

2.3.3 Legacy Rotation Methods

Two methods are commonly used to change a spacecraft's attitude. The first method discussed will be the *taxicab* method. It minimizes the 1-norm of the slew, similar to driving through city blocks in a grid configuration. An example taxicab slew is provided in Figure 2.9. Using this method, the spacecraft rotates only about its principal axes, in sequence (i.e., moving in each axis direction in the surface plane sequentially). The second method is the most common for modern spacecraft and is the *eigenaxis* method, which is shown in Figure 1.1. An eigenaxis slew minimizes the 2-norm of the slew angle between two points (i.e., the sensor boresight follows a direct arc from the current position to the desired position).

For both of these methods, there is a nonlinear relationship between the moving vehicle and the pan and tilt angles to a given target. Therefore, to simplify the calculation the following process is used.

1. The time required to perform the slew is computed, assuming the slew is performed from a static position at (x, y, z).
2. Once the slew time is computed, the vehicle is advanced along the n_1 axis according to kinematics ($x_{\text{new}} = x_{\text{previous}} + v_{\text{vehicle}} t_{\text{slew}}$).

3. Vehicle velocity is constant and directed along the n_1 inertial axis, which is coincident with b_1 body-fixed axis.
4. Slew time used for the cost function is recomputed from the attitude at the previous vehicle position with the sensor boresight pointed at the previous target to the attitude at the new vehicle position with the sensor boresight pointed at the new target.

Taxicab Method

Absent other circumstantial benefits, the taxicab method is an inefficient method of slewing a sensor. This concept is extended to a spacecraft sensor by first maneuvering around one spacecraft principal axis and then another in sequence, ultimately arriving at the desired sensor orientation. In the event that there are power, design, or external limitations in a system, this may be a desirable or necessary practice. From an efficiency perspective, it is undesirable.

Eigenaxis Method

The eigenaxis slew is a simple method to improve slewing efficiency over taxicab slews by maneuvering directly to the desired attitude. This rotation takes its name from the Euler parameters that define a rotation [6]. While the eigenangle describes the minimum angular distance to accomplish the desired attitude change, it does not infer any guarantees of time-optimality [1]. Time-optimality of rotations will be explored in more detail in Chapter 4. A brief comparison of the time required to accomplish an identical three-axis reorientation by both legacy methods discussed here, as well as the time-optimal method introduced in Chapter 1 is provided in Table 2.1. In context, the relative inefficiency of a taxicab rotation and the potential advantages of a time-optimal reorientation are evident.

Table 2.1. Slew time comparison by maneuver for a representative rotation.

Method	Time per Maneuver
Taxicab	46.8 s
Eigenaxis	35.5 s
Time-optimal	29.6 s

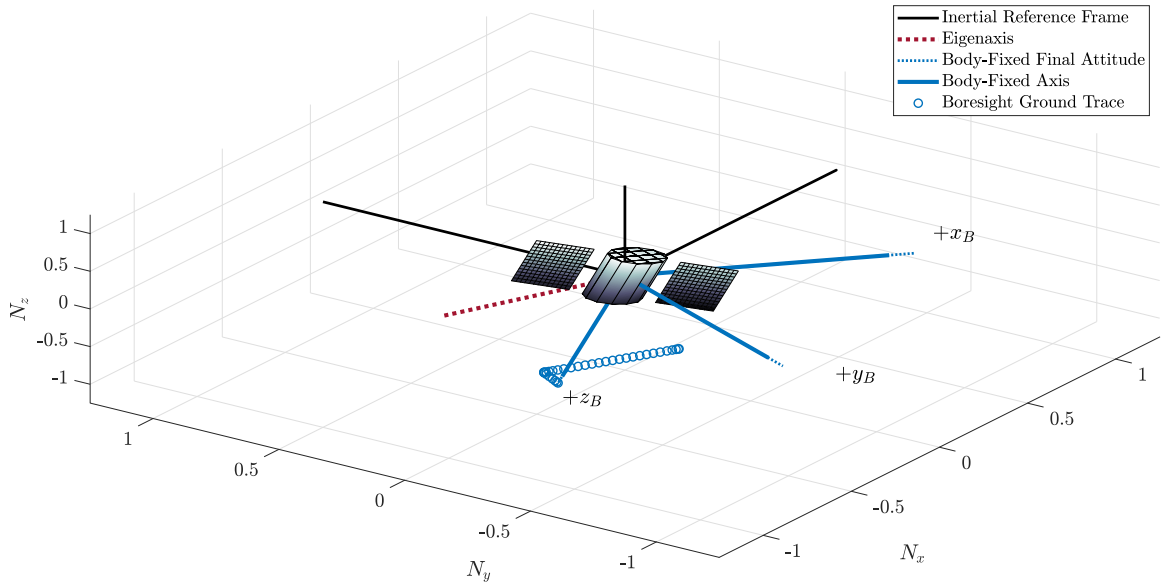


Figure 2.9. Taxicab rotation, duration = 46.8 sec.

2.4 Summary

This section introduced orbital mechanics, rotational kinematics, as well as a model for describing spacecraft rotations using multiple reference frames. These relationships are the basis for computing slew times needed by a mission planning system. That planning process will be developed in Chapter 3. Finally, this section also introduced two legacy slewing techniques, as well as the mathematics required to compute slew times for each.

The eigenaxis technique assumes a kinematics model in which the vehicle can be accelerated in any direction, up to a maximum angular velocity. This assumption is a simplification and neglects significant realities including the physical properties of the spacecraft, which time-optimal techniques will account for in Chapter 4. Taxicab rotations are further limited to slews about a spacecraft's principal body-axes, exacerbating the inefficiencies of the eigenaxis method.

To their advantage, eigenaxis maneuvers are simple and direct. They are intuitive, provide feasible results, and have been effectively used for decades for planning and on-orbit. However, as the quality and reliability of spacecraft increase, enterprise efficiencies are increasingly attainable and desirable.

Moving away from simple, suboptimal maneuvers is a necessary step for today's responsible stewards of satellite resources, given the availability of optimal control techniques. The following chapters will propose a method for implementing these optimal control approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Development of a Mission Path Planning Algorithm

To take advantage of the rotational agility of imaging spacecraft, a process is needed to select which targets to image and in what order. This planning process should be based on the slewing maneuvers that would be required, with the goal of selecting a sequence of operations that includes as much efficiency as possible. This would allow an organization to take as many pictures as possible, given a finite amount of the time in which the spacecraft may have access to an imaging target.

This chapter will classify the planning problem using combinatorial literature, describe the challenges associated with combinatorial problems of this nature, and present an approach to this planning problem that uses that uses genetic algorithm (GA) techniques. The interactions of the planner relative to the larger problem are highlighted in Figure 3.1.

3.1 Problem Classification

Finding an optimal solution in a set of possible solutions is referred to as *combinatorial optimization*. For mission planning problems, it is not feasible to simply try every combination because of the scale and computing limitations introduced in Chapter 1. Recall, for an imaging target list of 31 potential targets, there are 31 factorial possible sequences through the entire list without repetition. The formula for permutations quantifies this phenomenon

$$P(n, r) = \frac{n!}{(n - r)!} \tag{3.1}$$

where there are n distinct objects and r is the number of objects selected [9]. Note that this refers to a *permutation* where the order in which the objects are selected matters, as opposed to a *combination* where the order of selection does not matter. For a mission planning problem, the sensor will slew from one target to the next in a prescribed order; the correct approach here is to analyze permutations.

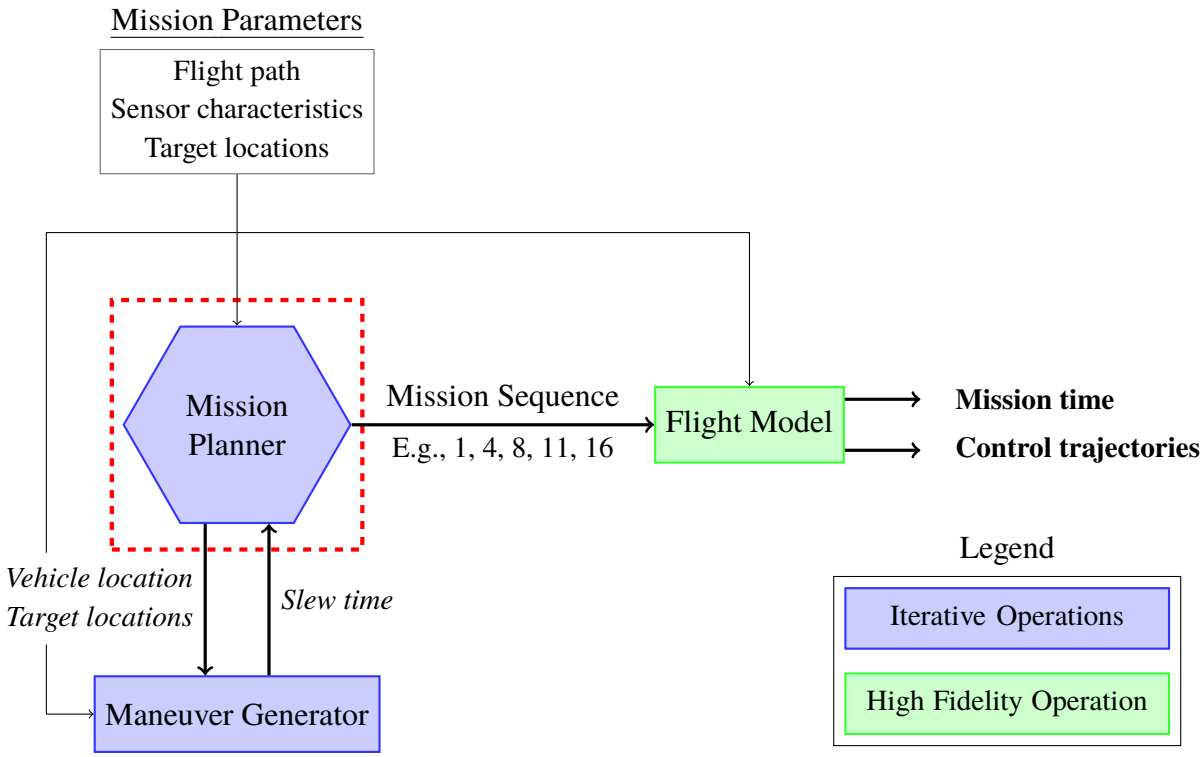


Figure 3.1. Over-subscription planning flow diagram (Mission Planner).

3.1.1 Traveling Salesman Problem

To determine the total number of sequences through all 31 targets, simply calculate the number of permutations of 31 objects while selecting 31:

$$P(31,31) = \frac{31!}{0!} = \frac{31!}{1} = 8 \times 10^{33} \tag{3.2}$$

Equation (3.2) refers to the solution space for a common combinatorial optimization problem called the Traveling Salesman Problem (TSP). TSP describes a traveling salesman with a list of cities to visit and who wishes to find the most efficient route through all of these cities. This problem is classically addressed using the two-dimensional distance between each city, with the solution being the shortest possible path that visits all cities.

Assume a situation in which a time or distance constraint prevents the traveling salesman from visiting all cities. For example, assume only 11 cities may be visited. The number of

possible solutions is then

$$P(31, 11) = \frac{31!}{20!} = 3 \times 10^{15}. \tag{3.3}$$

While this number of solutions is significantly less than the full permutation result, is is still a prohibitively large number of possibilities to solve naively; the author's personal computer has difficulty solving problems with permutations in excess of $11! = 4 \times 10^7$.

3.1.2 Orienteering Problem

To further develop the problem at hand, we must consider the value attached to various imaging targets. Assume some targets have greater value to the planner than others. With this concept incorporated, the problem begins to resemble an *orienteering problem*, which is a subset of the TSP. "In an orienteering problem, we are given a set of cities, a prize for each city (possibly zero), and a network of roads between cities. The objective is for a salesman to collect as much prize money as possible given a fixed amount of gas" [4]. So in an orienteering problem, two concepts are introduced: constraints on the number of cities that can be visited and non-uniform value associated with different cities. David Smith proposes a methodology for selecting an optimal path in such an *over-subscribed* scenario, which is a situation where the resources required to visit every city exceed the resources available [4]. However, the number of *nodes* (cities) in his example is small, so an exact solution can be computed.

In the spacecraft planning problem, the resource requirement for collecting against one target is the same as any other. This may contrast to other problems, for example a Mars rover planning problem where different science experiments may require different resource allocation. The planning techniques introduced in this thesis have been refined into a methodology the author refers to as the Remote Sensing Iterative Planner (RSIP).

Smith addresses the greedy approach to orienteering problems, in which the highest value targets are collected first regardless of resource requirement. Greedy approaches repeat this process until resources are exhausted, which is a simple and intuitive method. Unfortunately, the greedy approach often does not yield the best results. For the remote sensing applications, greedy planning would be an appealing approach, however the problem of dimensionality precludes naively solving every possible solution for comparison. Smith's orienteering problems have only a few nodes, so all possible paths can be explored. That is

not possible for mission planning problems with the number of potential targets that must be considered for remote sensing from space.

While the brute force approach will not provide insight, the principle of assigning values to targets is relevant. RSIP attempts to achieve some of the advantages of the greedy approach in a more advantageous way by incorporating the values of the targets in the style of an orienteering problem using a fitness function, $\mathcal{J}$. Fitness functions are the formulas for calculating the "goodness" of a candidate solution, accomplished by defining a metric to be minimized. Candidate solutions are evaluated against the fitness function throughout an iterative solver. Fitness functions for a solver can be refined to improve performance, resulting in an algorithm that converges on desirable solutions.

3.1.3 RSIP Fitness Functions

The spacecraft image planning problem has all the elements of an over-subscribed orienteering problem—a complex set of imaging targets with varying distances between them, time constraints due to a fixed orbital velocity and field of view, and differing values assigned to different imaging targets. Fitness functions are used to evaluate the goodness of each potential solution during the course of the GA. Each version of the planner uses only one fitness function; however, several versions were created to build to the capability of the final version of RSIP.

In the first version, the fitness function ($\mathcal{J}$) minimized distance between all targets in each target sequence

$$\mathcal{J}_1 = \sum_{i=1}^{n-1} d_i \tag{3.4}$$

where n is the number of cities and d is the distance between two sequential cities in the sequence.

Next, the approach was enhanced by changing the fitness function to minimize the time required to travel between one city and the next. This allows for different vehicle speeds in different directions.

$$\mathcal{J}_2 = \sum_{i=1}^{n-1} t_i \tag{3.5}$$

where t is the time required to travel between two sequential cities.

After accounting for travel time, the method of computing the travel time between cities was changed from modeling two-dimensional travel to the time required to slew a spacecraft sensor from one target to the next. This required an adjustment to the method of calculating t_i, however the fitness function, Equation (3.5), was unchanged.

The fitness function was next adjusted to account only for the targets that could be collected in the constrained environment, n_c.

$$\mathcal{J}_3 = \sum_{i=1}^{n-1} \alpha t_i + \sum_{j=1}^{n_c-1} \beta t_j \tag{3.6}$$

This function is a *multi-objective fitness function* in contrast to the single objective fitness functions in equations (3.4) and (3.5). To generate an efficient solution for a time-constrained problem, generally efficient paths through the entire target set must be achieved before time-constrained performance takes priority. Without this groundwork, the algorithm consistently converges prematurely. This effect is achieved with the scalarization method of multi-objective optimization [10]. The values of the scale factors α and β are pre-determined and changing over the total number of generations; this will be discussed in more detail in Section 3.2.5.

Finally, the multi-objective fitness function was adjusted to include value as a decision variable in lieu of constrained travel time in the final version of RSIP. The formulation for Equation (3.7) will be further developed in Section 3.2.6.

$$\mathcal{J}_4 = \sum_{i=1}^{n-1} \alpha t_i + \beta \left[v_{\max} - \sum_{j=1}^{n_c} v_j \right], \tag{3.7}$$

where $v_{\max}$ is the total value available in the target set, v_j is the value associated with a given target and α and β are again scale factors.

This sequence of planning process development began with a simple Traveling Salesman Problem construct that was solved with a genetic algorithm. From there, the use of travel time as a decision variable was introduced, and finally orienteering problem characteristics were

included in the form of a multi-objective fitness function accounting for constrained time and target collection value. The underlying genetic algorithm approach remained constant through each of these versions. The implications of these functions will be discussed in more detail in Section 3.2.

3.1.4 Genetic Algorithm Nomenclature

John Holland was one of the developers of the *genetic algorithm* in the 1960s and 1970s, in an effort to model evolutionary behaviors outlined in Charles Darwin's theory of natural selection [5]. A GA attempts to leverage evolutionary traits found in biology and apply those principles to other complex problems that seek to converge on an optimal solution. GAs borrow much of their nomenclature from biology as well. A group of possible solutions to a problem, or paths through a problem, is referred to as a *population*. The individual solutions are called *genomes* and are encoded into *chromosomes*; therefore, a population is made up of chromosomes.

A population evolves over *generations*, where each generation draws its population from some combination of chromosomes from the preceding generation, and the results of genetic modifications to the preceding generation's makeup. Chromosomes are evaluated by some fitness function, $\mathcal{J}$, wherein the best performing chromosomes are selected to pass their genetic makeup (their genomes) on to the next generation. The concept of passing high-performing genomes into the next generation is referred to as *elitism* and depending on the mechanisms chosen to build out future populations, elitism can be relatively weak or strong.

Some members of the next generation are generated through random changes to chromosomes. This occurs in two different ways: either *crossover* which combines portions of different parent chromosomes, or *mutation* which randomly manipulates some portion of a parent chromosome. The next generation is populated by way of selection, crossover, and mutation, and the process repeats over successive generations (frequently hundreds or thousands of generations) until an acceptable solution to the problem as been found, typically by converging on a nearly optimal solution.

3.1.5 Properties of Genetic Algorithms

"Genetic algorithms have some distinct advantages for certain classes of problems, notably their ability to handle complex problems and their use of *parallelism*—the ability to explore the search space in many directions simultaneously" [5]. They are able to pursue multiple solutions in parallel because individual chromosomes act independently from one another and can explore the search space independently as well. Additionally, GAs can deal with *stationary* fitness functions that remain static over time or with *non-stationary* fitness functions that change with time.

Some problem attributes such as genome sequence and relative fitness are produced and tracked by the algorithm. However, other elements of the algorithm must be specified by the developer. For example, the fitness function, the structure of chromosomes, and the rates of selection, crossover, and mutation must all be predetermined and coded into the algorithm. As with any selection, there is the risk of selecting disadvantageous parameters. For example, defining the correct fitness function is critical—a poor definition here may lead to an excellent solution to a problem that the designer did not intend to solve. Similarly, the exact mechanisms for crossover and mutation must be defined. The mechanism of combining parent chromosomes for crossover must be defined, as well as the frequency and extent of mutations. Additionally, the size of the population and the number of generations over which the population will be evolved must be determined; too small and an optimal solution may not be found, too large and excessive computing time may be required. Careful design as well as trial and error are frequently required to create an effective GA. With this in mind, it is an effective tool for efficiently generating solutions that converge on optimality.

3.1.6 Genetic Algorithm Design

In this section, the implications of genetic operators and population parameters will be explored in more detail.

Genetic Operators

Genetic algorithms typically use three mechanisms to pass genes from one generation to the next—selection, crossover, and mutation.

Selection New generations are populated with some portion of the highest performing chromosomes from the previous generation. This *selection* is determined by the fitness function where better performing chromosomes are preferred and it ensures that high fitness solutions are included in subsequent generations. The concept of selection of the fittest is referred to as *elitism* in the GA context. Algorithm designers decide to what degree selection of high performing solutions are flowed into subsequent generations, however very strong elitism could cause the algorithm to converge on a solution too soon, similar to finding a local extremal solution instead of the global extremal solution. Conversely, excessively weak elitism can delay converging on a solution. These concerns must be considered when assigning weights and probabilities in the GA design process.

Crossover Crossover is typically the primary method of populating a new generation. Crossover can be implemented in different ways. One option is to combine parent chromosomes at a single point (i.e., the beginning portion belongs to one parent chromosome and the ending portion belongs to the other). Another option is to use multiple swap points, where the child chromosome is a collection of multiple segments from the parent chromosomes [5].

Generally, crossover is the main method for making evolutionary progress. For a given crossover chromosome, genetic segments are chosen from parent chromosomes that have remained in the population due to their evolutionary progress. As the populations become more efficient with later generations, it follows that the chromosomes within the population as well as their progeny are also becoming more efficient. These actions enhance the chances of finding increasingly efficient paths from desirable building blocks. This principle is the basis for relying on the crossover component to aid convergence to a solution in a GA. More precisely, the crossover operation provides "mixing of the solutions and convergence in a subspace" [5].

Mutation While the goal of crossover is to aid convergence, Yang attributes increases in the genetic diversity of a population to the mutation operation [5]. Probability of mutation is typically much smaller than the crossover probability. Mutations commonly swap bits or elements of a chromosome, however that could happen at only two sites, multiple sites, or more complicated schemes could be developed. Mutation at multiple sites can add evolutionary efficiency to a routine, however Yang cautions that excessive mutation can

make it difficult for a population to converge. Thanks to these injections of randomness into genomes, mutation tends to find a solution outside of the subspace, and in that way expands genetic diversity [5].

Population Parameters

Population size must be considered as a balance between having a sufficiently large population to explore all of the relevant search space, and avoiding excessively large populations that require extensive computing resources to evaluate each chromosome's fitness. A population that is too small risks being caught in a premature solution if one chromosome shows significantly better fitness than the others and then overwhelms all other chromosomes in a small population. Yang recommends a population size of $n = 40$ to 200 for most problems [5].

Consideration must also be given to the appropriate number of generations. One simple method is to define a set number of generations and evaluate the performance over those generations. If performance improvements diminish well before the end of the routine (i.e., performance flattens out early), then fewer generations may be needed to arrive at an acceptable solution. Conversely, if significant improvements continue to be found through the final generation, then a convergence likely has not occurred and the routine may benefit from additional generations.

In some cases, a stopping criterion could also be used. If performance improvements drop below a certain threshold, or if a specified performance level has been reached, a routine could be instructed to terminate. These approaches require sufficient knowledge of the system to ensure that the stopping criteria is eventually reached. This may be feasible, however for initial development of a GA, some amount of trial and error should be expected.

3.2 Developing the Remote Sensing Iterative Planner

Based on the discussion to this point, a TSP construct is a reasonable starting framework for overhead remote sensing planning. For a set of targets or cities, the goal of this system is to develop a path that travels through the cities as efficiently as possible. Two significant differences from the classic TSP arise: there is no guarantee that an airborne collection plan

will be able to collect every potential target and there is no guarantee that all targets will be equally valued. Regardless, a TSP approach is a promising starting point.

A naive, brute force method of solving a TSP is only feasible for very small target sets. Generally, target sets of 10 to 12 targets can be brute forced with commonly available computers; each additional node adds significant complexity. A TSP solution for an 11-node mission plan where all possible paths are computed yields 4×10^7 possibilities. Add one additional target and the total number possible permutations increases by an order of magnitude to 5×10^8. Quickly this becomes prohibitively complex.

When choosing an approach for solving a TSP-like problem for mission planning purposes, we can take advantage of the fact that true optimality is not required. This opens the decision space up to feasible solvers that do not guarantee optimality, but likely converge on an optimal solution.

The technique employed should be well suited to the complexity resulting from the number of permutations in a likely problem, and be able to search in multiple directions of the search space simultaneously. A technique that begins with some subset of all possible paths, then selects the more efficient solutions, and finally iterates those selected paths to find further improvements is clearly advantageous. Ultimately, genetic algorithms excel at this type of problem and that is the primary approach for this remote sensing mission planner, RSIP.

3.2.1 Features of the RSIP Genetic Algorithm

Several features of the GAs are well suited to this planning problem. This section will explore these features and their specific implementation in RSIP.

Problem Complexity

The target list used for this thesis contains 31 elements. For context, the number of targets that can be collected by this model from low earth orbit is typically in the range 5 to 15 for a given mission. To begin unscrambling a randomly initialized permutation, the algorithm must consider the full permutation. From Equation (3.2), that search space contains nearly 10^{34} possibilities. Even if reduced to 11 targets out of the 31 total candidates, the number remains unreasonable at around 10^{15} possibilities, Equation (3.3). Assuming

this complexity precludes a brute force approach, a genetic algorithm continues to be an appealing approach to the problem.

Search Space Exploration and Mutation

To determine whether a randomly generated permutation might be broadly efficient by chance, the best performing genome in a randomly initialized population was plotted. Examining Figure 3.2, there is no apparent efficiency in this randomly generated permutation. Heuristically, this was the case consistently. From this, it was inferred that convergence to an efficient solution does not lie in the original randomly generated permutations. To the contrary, mutation is required to expand the search subspace outside of the genetic makeup of initial randomly generated genomes. It should be advantageous to rely heavily on mutation to expand the search space, particularly given a population size of 100 and reasonable solution spaces on the order of 10^{15} possibilities.

Crossover

Of the three genetic operators, crossover operations are problematic. Crossover relies on combining elements of two parent chromosomes. In binary or combination operations, this usually does not present a problem. However, for a image collection planning problem, each chromosome is a permutation, and therefore repetition or skipped elements are unacceptable. All chromosomes must be complete permutations of the range of cities. Therefore, crossover is particularly problematic in this case and was not used. This leaves only selection and mutation as the feasible genetic operators. Based on Yang's analysis, we should be wary of the algorithm failing to converge on a solution due to the lack of crossover. However, given the heavy reliance on mutation to expand the search space beyond the initial population, the infeasibility of crossover for this application may not be problematic. That is to say, convergence is not the biggest challenge here; expanding the search space is the more pressing concern.

Selection and Elitism

To balance elitism that preserves high performing chromosomes without stifling the benefits of parallelism, a strict 25% selection parameter was used, but not over the entire population. This was a design characteristic of Kirk's approach that was adopted here [11]. Early trials

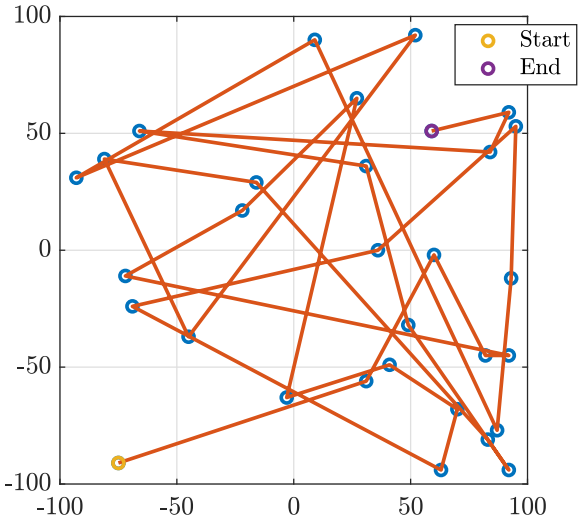


Figure 3.2. Best performing population member in the first generation.

demonstrated that selection from the best performing subset of the entire population (i.e., the top 25% of 100 genomes) quickly eliminated almost all diversity in the population; mutation was unable to provide sufficient balance.

In Kirk's approach, the population is broken into groups of four chromosomes and the highest performing chromosome within each group is selected. The other three are discarded, and their places are taken by mutations of the selected chromosome. In this way, the diversity associated with parallelism is enforced while still maintaining a strong sense of elitism to ensure that well-performing members are maintained and only replaced if a better version of that chromosome is created through mutation.

Parallelism

Structuring genetic competition for a place in subsequent generations into bouts of four chromosomes is a key design choice. In early iterations of the GA, the entire population consisted of very similar genomes within a few hundred generations. This early convergence consisted of genomes that outperformed the genomes formed randomly at initialization, but did not offer a path to a globally efficient solution and resulted in inefficient solutions overall. Enforcing parallelism by instituting bouts of four for selection eliminated this early convergence problem and allowed the algorithm to consistently progress toward a globally optimal solution.

Stationary and Non-stationary Fitness Functions

The flexibility of GAs to handle different types of fitness functions was a key enabler to developing a planner that applied GA concepts to a large scale orienteering problem. Equations (3.4) and (3.5) are stationary fitness functions in that the metric used to evaluate a chromosome's relative fitness does not change during the course of the algorithm (i.e., from one generation to the next).

Equations (3.6) and (3.7) are examples of multi-objective non-stationary fitness functions. Equation (3.7) is also the fitness function used in the final version of the mission planner.

$$\mathcal{J}_4 = \sum_{i=1}^{n-1} \alpha t_i + \beta \left[v_{\max} - \sum_{j=1}^{n_c} v_j \right] \tag{3.7}$$

Here, the scalarization method is implemented with scale factors α and β, which adjust over time to emphasize the total path time objective of the fitness function early in the process and the constrained value objective toward the end of the evolutionary process. To accomplish this effect, the total path time scale factor, α, decrements linearly from one to zero over the course of the GA, while the constrained value scale factor, β, increments linearly from zero to one over the same period.

This fitness function results from the need to generate modestly efficient choices early in the evolutionary process. Without generally efficient solutions in the population by the time that constrained choices take precedence, the algorithm will not be able to converge on a low time or high value solution. These relationships cause efficient paths through the entire target list to be favored early on and high value solutions that satisfy algorithm constraints to be favored in the later portion of the solver.

3.2.2 RSIP Design Choices

The first iteration of the Remote Sensing Iterative Planner relied on several simplifying assumptions. This simplified planner was used to develop the general technique and validate the genetic algorithm parameters chosen by the designer. The initial assumptions were

1. All targets are visited; time is not a constraint.
2. Travel from one target to the next occurs at the same speed in all directions.

3. Travel from one target to the next occurs in a straight, two-dimensional line.

None of these assumptions will carry through to the final model, but all are assumed in the first iteration of the GA. Several techniques for GA implementation were considered, however all shared a few attributes fundamental to genetic algorithms:

1. The first generation of the algorithm is initialized with a *population* of n samples of randomly generated, unique permutations that represent possible collection plans.
2. A certain portion of the best performing members (or *genomes*) of the population are selected for inclusion in the next generation.
3. Remaining members of the next generation are created through some combination of parts of well-performing genomes, selected in the previous step. This is accomplished by way of mutation of parent genomes.
4. Iterative selection of the best performers resulting from random genetic adjustments of preceding generations is repeated until a satisfactory solution to the planning problem is achieved.

This technique is flexible over many parameters chosen by the designer; however, typical population parameters used in this planner are listed in Table 3.1.

Table 3.1. Typical genetic algorithm population parameters.

Parameter	Value
Population Size	100
Iterations	1000
Imaging Targets	31

To guarantee the optimal solution using a brute force method, a 31-city mission plan would require evaluating 31 factorial (8×10^{33}) permutations, which is too many for any practical system. By contrast, a feasible result is achieved by evaluating 10^5 ($100 \cdot 1000$) different permutations using the genetic algorithm. Here the advantages of the GA become apparent.

After considering various implementations of a GA, the essential elements of the RSIP GA are:

1. Population genomes are broken into groups of four.
2. The best performing genome of each group of four is selected for inclusion in the subsequent generation, and the remaining three genomes are discarded.
3. The best performing genome is replicated and mutated to generate three child sequences to fill the remaining three spots in each group of four.
   (a) One child sequence has a randomly selected portion of the gene reversed.
   (b) One child sequence has two randomly selected elements swapped.
   (c) One child sequence has a randomly selected portion shifted by one position, keeping the elements' relative order intact.
4. This iterative process is repeated for the desired number of generations.

3.2.3 Results of Genetic Algorithm Minimizing Distance

Using total path length as the fitness function (i.e., minimizing the two dimensional distance over the permutation path through the set of targets presented in Figure 3.3), typical GA results are presented in Figure 3.4. Decaying exponential progress is indicated, implying that a near-optimal solution may have been obtained. This method does not guarantee optimality, nor does it preclude the routine from becoming trapped at a local minimum, however this result is feasible in that valid permutations are produced and those permutations become more advantageous as the routine progresses.

This first iteration of the algorithm does not address movement in different directions at different rates nor being time constrained and therefore unable to visit every city. These features are added in the next versions of the GA.

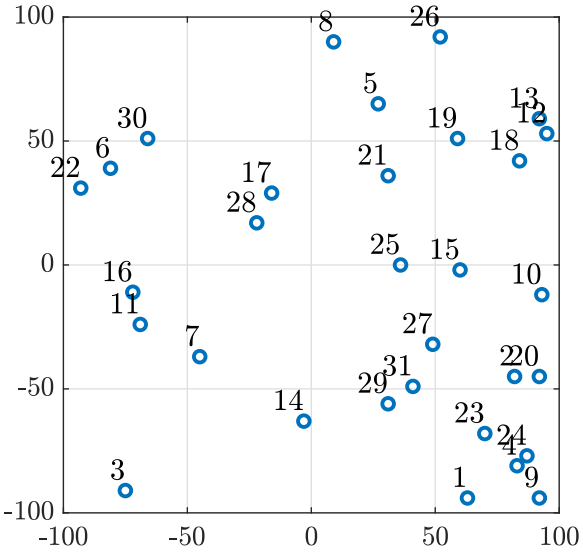
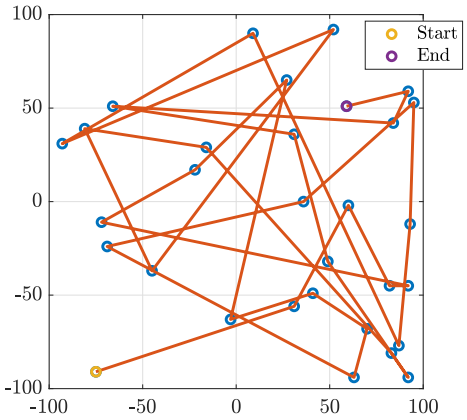
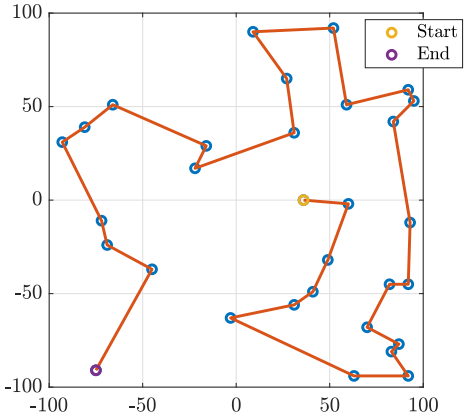


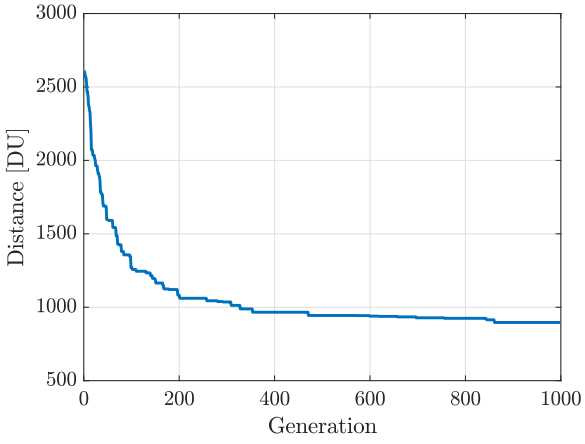
Figure 3.3. Labeled targets (cities in the traditional TSP discussion).



(a) Shortest path between all cities in the first generation.



(b) Shortest path between all cities in the 1000th generation.



(c) Best performing population member by generation.

Figure 3.4. Genetic algorithm performance, minimizing distance between cities.

3.2.4 Results of Genetic Algorithm Minimizing Time

Finding the shortest two-dimensional path through a set of targets is certainly advantageous, however in a spacecraft application a vehicle's sensor will not necessarily move at equal speeds in all directions. The GA can be modified to accommodate this.

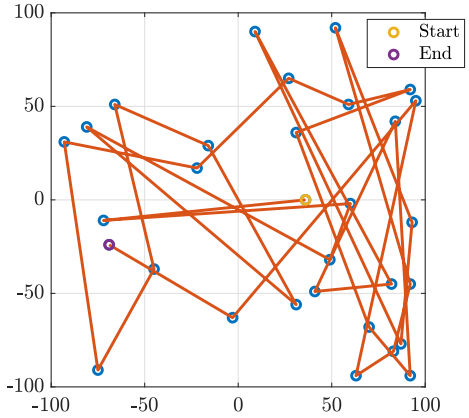
To demonstrate the concept, the scenario from Figure 3.4 is repeated, but instead of minimizing the total distance traveled, total time for the route is minimized. The case presented in Figure 3.5 assumes that the traveling salesman can move along the vertical axis at twice the speed that the salesman can move along the horizontal axis.

Compared to Figure 3.4, the time-minimizing parameters in Figure 3.5 show similar trends in performance improvement, as well as similar heuristic differences from the best randomly-generated genome in the first generation when compared to the best performer in the final generation. The first generation is of course unimpressive, as it is a randomly generated permutation. It is depicted in Figure 3.5a.

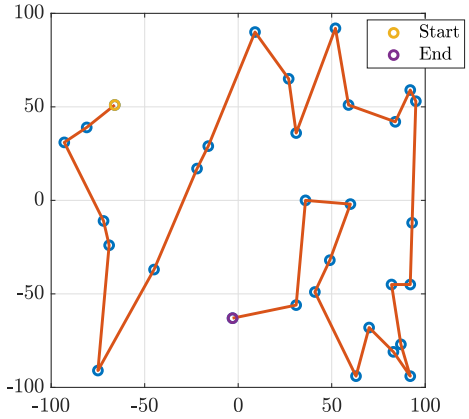
The total time minimizing solution is presented in Figure 3.5b. The fastest total time solution superimposed on the shortest distance solution is presented in Figure 3.6 for comparison. This figure highlights the tendency of the algorithm to favor travel in the vertical direction, as there is a speed advantage to be exploited. Table 3.2 lists the time and distances for the travel paths using generic distance units (DU) and time units (TU). The fastest time approach finds its best solution over a much longer ground path than the shortest distance approach.

Table 3.2. Comparison of shortest distance and fastest time methods.

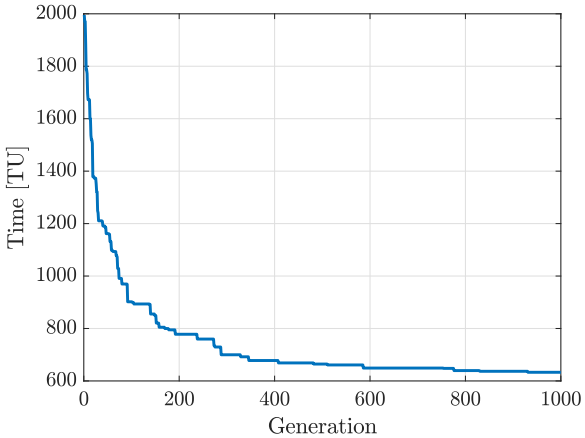
Optimization Method	Distance [DU]	Time [TU]
Shortest Path	896.3	696.5
Fastest Time	1009.7	633.0



(a) Shortest travel time between all cities in the first generation.



(b) Shortest travel time between all cities in the 1000th generation.



(c) Best performing population member by generation.

Figure 3.5. Genetic algorithm performance, minimizing travel time.

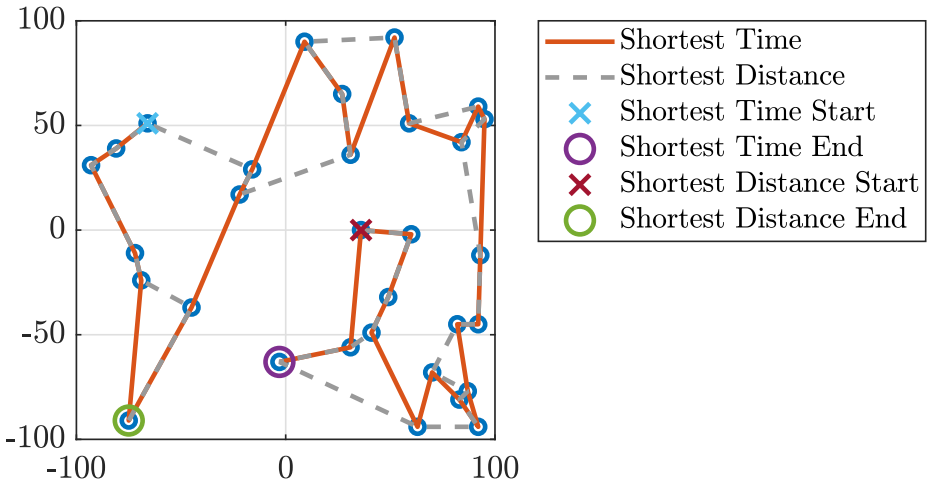


Figure 3.6. Fastest time and shortest distance solutions superimposed.

3.2.5 Results of Genetic Algorithm Minimizing Time with Time Constraints

Now that the genetic algorithm is configured to account for different rates along different axes, overall time constraints must be addressed. Spacecraft are constrained by orbital mechanics, therefore the altitude and access window for a remote sensing vehicle may not permit the vehicle to collect all the desired targets. In fact, this is likely to be the case. The mission planner must be able to account for time constrained scenarios in which only a portion of the desired targets can be collected.

Practically, the planning algorithm must be able to generate a mission plan for an n choose r permutation where the number of targets collected (r) is not strictly defined, rather enforced by a maximum allowable mission time. Later, the mission time will be enforced as a product of the spacecraft's orbital elements. This means that the final version of RSIP will determine maximum mission time by calculating the window in which the spacecraft falls within an acceptable field of regard based on look angle constraints. Before building up to that model, time is constrained as an operator input.

Figure 3.7 presents the results of a time constrained solution. In this case, the maximum allowable mission time is 300 TU, chosen by the designer. The GA for this scenario operates in largely the same manner as previously described, with an updated fitness function, Equation (3.6).

$$\mathcal{J}_3 = \sum_{i=1}^{n-1} \alpha t_i + \sum_{j=1}^{n_c-1} \beta t_j \tag{3.6}$$

In this scenario, 17 out of 31 targets are collected within the 300 TU constraint.

The different paths and travel times when minimizing time versus distance inform a key takeaway for mission planning programs. Adopting a simplifying assumption that travel distance is a reasonable proxy for image collection planning comes with significant costs. In Table 3.2, the time calculations assume identical velocity capabilities for the shortest distance and fastest time approaches, however the shortest distance approach is 10% less efficient than the approach that takes vehicle speed into account.

As spacecraft generally do not have uniform agility, extending the shortest distance assumption to spacecraft maneuvers would have similarly detrimental effects. This result reinforces

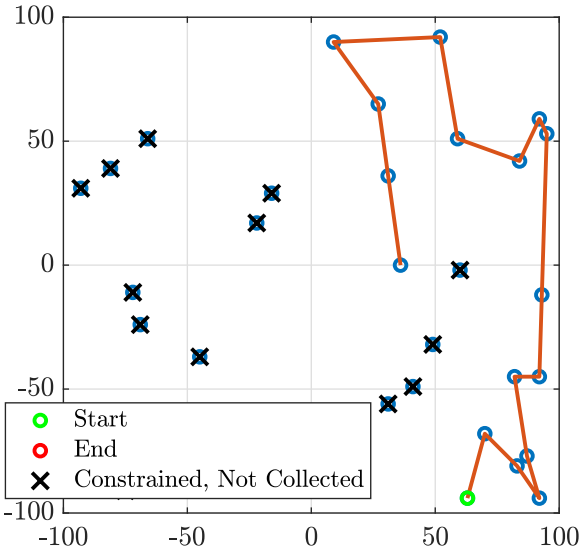


Figure 3.7. Best-time constrained collection plan, final generation by shortest travel time.

the need for path planning programs that account for the non-uniform maneuvering capabilities associated with spacecraft.

3.2.6 Results of Genetic Algorithm Maximizing Value with Time Constraints

Values of 1, 2, or 3 value units (VU) have been assigned to the targets, as depicted in Figure 3.8.

The value-based multi-objective fitness function was generated in a similar fashion as the time-constrained fitness function discussed in Section 3.2.5. An adjustment was made to the fitness function so that uncollected value is minimized in the later part of the algorithm.

$$\mathcal{J}_4 = \sum_{i=1}^{n-1} \alpha t_i + \beta \left[v_{\max} - \sum_{j=1}^{n_c} v_j \right] \tag{3.7}$$

Using the value-based approach, the value maximizing solution depicted in Figure 3.9a was generated. In this solution, the path tends to favor the right-hand side of the target

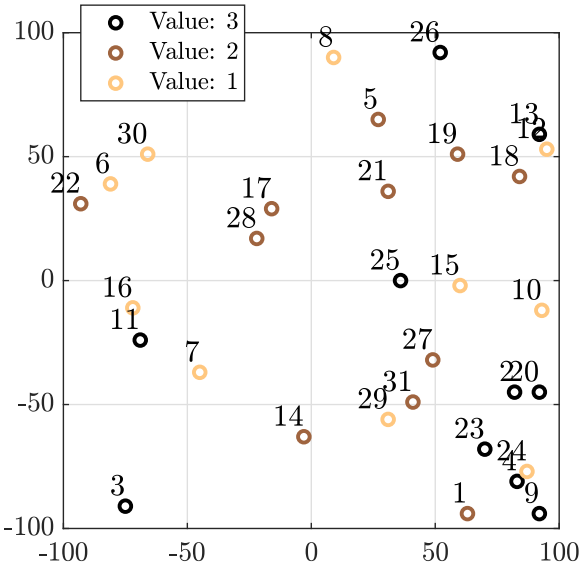
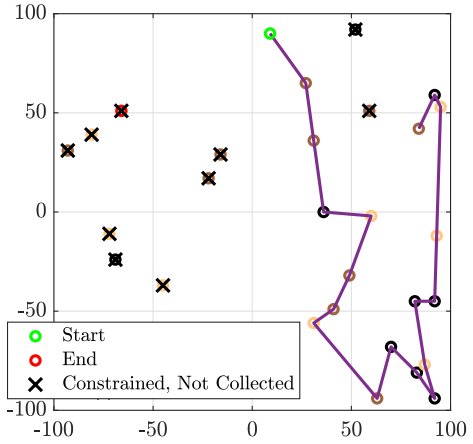


Figure 3.8. Targets with collection value assigned.

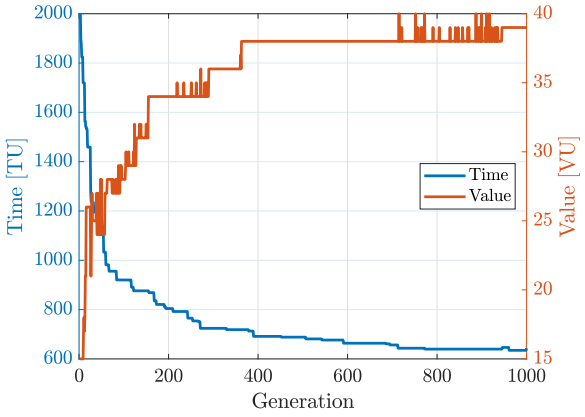
field where more high-value targets are located. Convergence on a solution occurs in the value-based version using the same tools and techniques as the time-minimizing approach.

Figure 3.9b illustrates progress in both time and value improvement over the evolution of this scenario, as a result of the multi-objective value-based fitness function. Some spikes over the final value collected are the result of scalarization causing some time-efficient paths to slightly out-perform value-efficient paths due to the scalar weights for those particular generations. Generally, this performance is indicative of the algorithm converging on an optimal solution.

This latest addition to RSIP is potentially open to criticism because of the manner in which the collected portion of the permutation is selected. In this planner, only the first portion of any given permutation is collected; therefore, if a more efficient or higher value path through the middle or end of the permutation was available, it would not be considered. Ultimately, the algorithm is designed to select for the most valuable sequences as they appear at the beginning of each sequence, and therefore that design is relied upon to ensure that the most advantageous paths evolve in that position.



(a) Time-constrained collection plan, final generation by most value.



(b) Population performance over entire path, by generation.

Figure 3.9. Time constrained, value maximizing genetic algorithm solution.

3.3 Remote Sensing Iterative Planner Mission Models

Section 3.2 has discussed several techniques for path finding for remote sensing purposes, culminating in the value-maximizing, time-constrained model. This model accounts for maneuvering abilities that are not uniform in all directions, includes the ability to assign different values to different targets, and selects for efficient plans that meet external time constraints that preserve maximum collection value in the constrained environment.

The next logical step is to extend these principles from a two-dimensional model to a remote sensing spacecraft model that incorporates orbital mechanics to derive the vehicle's position at a given time and vehicle kinematics to determine path times to reorient from one target to the subsequent target. Two legacy methods of accomplishing these calculations are the taxicab method which rotates about one principal axis of the vehicle, and then around a second, and the eigenaxis rotation, which slews the vehicle in a direct arc from one orientation to the next. Both of these methods were introduced in Chapter 2 and neither are the most efficient method of accomplishing an attitude change [1]. However, these approaches are commonly used today and are therefore useful to analyze.

Taxicab Method

A mission plan for an orbital altitude of 675 km over a target field described in Appendix A using the taxicab maneuvering approach is presented in Figure 3.10. This mission plan and all subsequent mission plans are value-maximizing and time-constrained, using Equation (3.7) as their fitness function.

In this example, five targets are collected for a value of 13 VU out of 62 available VU.

Eigenaxis Method

For the same scenario that was shown in Figure 3.10, changing only the slewing method from taxicab slews to eigenaxis slews yields value gains with no change in hardware. The taxicab and eigenaxis scenarios, along with all future spacecraft scenarios discussed, use identical vehicle and target parameters and can be directly compared. Figure 3.11 depicts such a mission plan, which when compared to the taxicab approach, collects two additional targets and achieves an additional five value units. Table 3.3 lists the average time per slewing maneuver from the same vehicle collecting on the same target field, albeit not

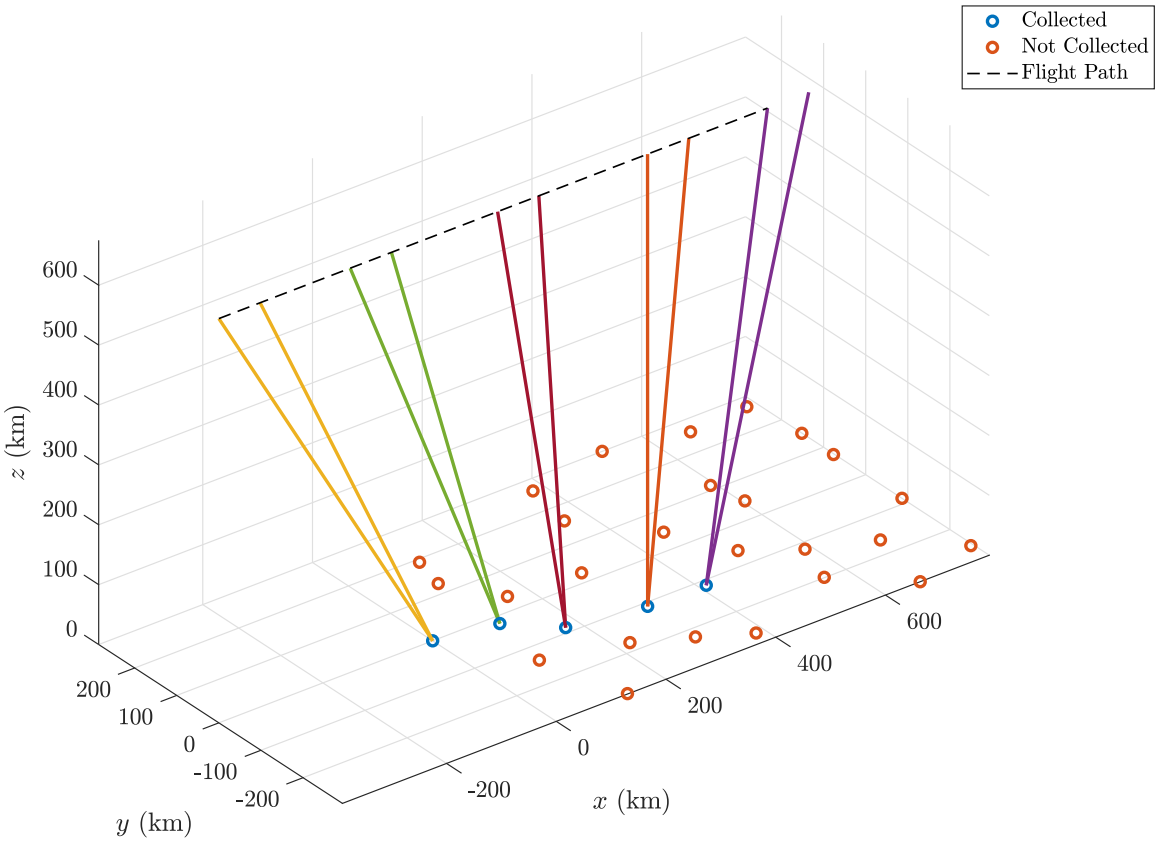


Figure 3.10. Spacecraft mission plan using taxicab slews, value maximizing.

identical targets. Based on this comparison, there is a clear advantage to eigenaxis slews over taxicab slews when possible. A simple Pythagorean inspection of the different paths also supports this conclusion.

3.4 Summary

This section introduced the path planning problem for spacecraft applications, including classifying the problem according to current combinatorial optimization literature. A genetic algorithm was offered as a candidate solution to this path planning problem, and such an algorithm was developed for remote sensing planning. Finally, two mission plans were presented based on this mission planner and using two different legacy spacecraft maneuvering techniques. The comparison of these two plans highlights the ability to gain value and path efficiency by adjusting the slewing method used. Chapter 4 will introduce the

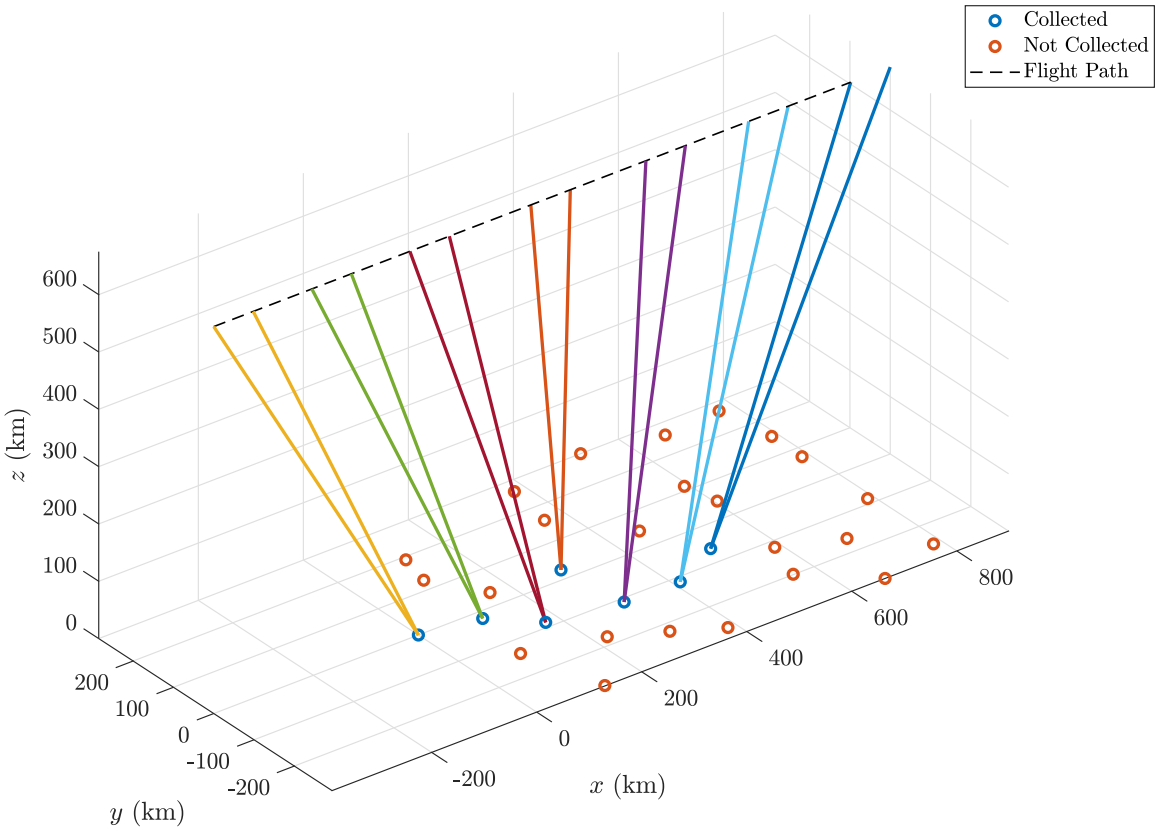


Figure 3.11. Spacecraft mission plan using eigenaxis slews, value maximizing.

concept of mathematically optimal attitude rotations that can achieve further efficiencies for remote sensing vehicle maneuvers, if such a maneuver can be incorporated into the planning process.

Table 3.3. Average slew time by maneuver.

Method	Time per Maneuver
Taxicab	20.73 s
Eigenaxis	13.23 s

CHAPTER 4:
Time-Optimal Maneuvers

Two path planning techniques for spacecraft reorientation that are simple to calculate and execute were introduced in Chapter 2, however these techniques are inefficient in terms of time. The least efficient technique addressed in this thesis is the taxicab technique, which rotates a vehicle around one principal axis and then another principal axis to achieve the final desired attitude. The eigenaxis technique is more efficient than the taxicab technique in that it accomplishes the reorientation in a single rotation along a direct arc, but it is also not time-optimal. Bilimoria and Wie showed decisively that even the direct arc reorientation of the vehicle will not be the optimal, minimum time reorientation in almost all cases [1]. The theory behind time-optimal reorientation maneuvers has been developed in literature for decades. Furthermore, flight implementation of truly time-optimal maneuvers has since been demonstrated on operational spacecraft, such as NASA's Transition Region and Coronal Explorer (TRACE) spacecraft [2].

This chapter develops the engineering model that produces a mathematically optimal minimum time reorientation maneuver of an example spacecraft. The relationship between the engineering model and the larger planning problem is illustrated in Figure 4.1. The approach relies on Pontryagin's principle to develop the necessary conditions for optimality by way of a cost function, dynamic modeling, endpoint constraints and path constraints. This implementation and problem solution is accomplished using the DIDO software package in MATLAB. Further details about DIDO and Pontryagin's principle for optimal control are available in two publications by Ross [12], [13].

This chapter will present the framework for minimum time reorientations of the example spacecraft, including scaling and balancing the problem, and verification and validation of the results for an example maneuver. Pontryagin's principle is briefly introduced, followed by the analytic and numerical implementation for this specific problem. The engineering model developed herein will produce a set of sample maneuvers that will be used to train machine learning models to approximate maneuver times based on the required slew, thereby enabling mission planning that incorporates minimum time maneuvers.

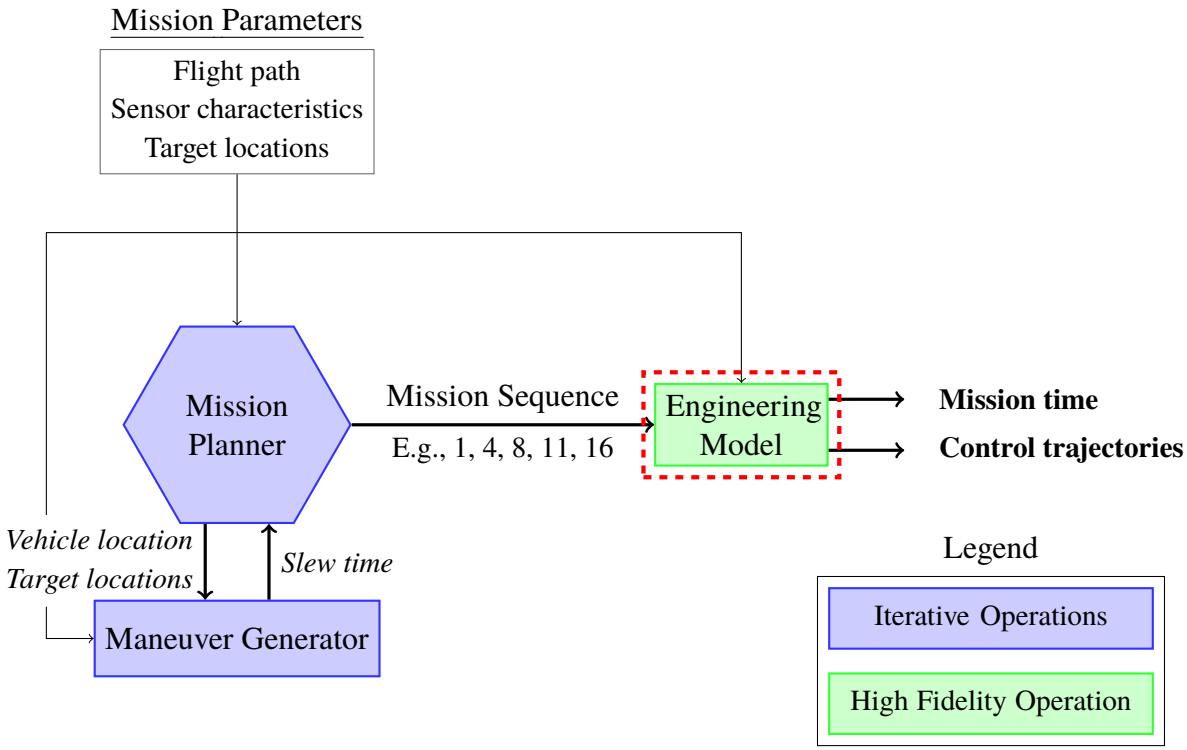


Figure 4.1. Over-subscription planning flow diagram (Engineering Model).

4.1 Parameters of Example Spacecraft

To investigate the viability of applying machine learning to a planning problem, an example reaction wheel spacecraft model was developed. The mass and agility parameters for this spacecraft are provided here. A summary of the notation used for optimal control problems is available in Appendix B.

The inertia matrix $\mathbf{I}$ and the reaction wheel configuration matrix $\mathbf{Z}$ for the spacecraft are given here.

$$\mathbf{I} = \begin{bmatrix} 222.17 & 2.58 & -9.10 \\ 2.58 & 264.51 & 2.42 \\ -9.10 & 2.42 & 171.91 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{4.1}$$

$$\mathbf{Z} = \begin{bmatrix} \cos(\alpha)\cos(\eta) & \sin(\alpha)\cos(\eta) & -\cos(\alpha)\cos(\eta) & -\sin(\alpha)\cos(\eta) \\ -\sin(\alpha)\cos(\eta) & \cos(\alpha)\cos(\eta) & \sin(\alpha)\cos(\eta) & -\cos(\alpha)\cos(\eta) \\ \sin(\eta) & \sin(\eta) & \sin(\eta) & \sin(\eta) \end{bmatrix} \tag{4.2}$$

$$\alpha = 45 \text{ deg} \qquad \eta = 30 \text{ deg}$$

The reaction wheel configuration matrix maps the reaction wheel cluster to the body principal axes. Torque exerted by an individual reaction wheel is represented by ${}^{rw}\tau$ and spacecraft angular velocity is given by ω. The maximum torque that can be exerted by a reaction wheel and the maximum angular velocity about any principal axis are given as

$$\begin{aligned} {}^{rw}\tau_{\max} &= 0.6 \text{ N} \cdot \text{m} \\ \omega_{\max} &= 1.00 \, {}^{\text{deg}}/_{\text{sec}} \, . \end{aligned}$$

4.2 Dynamic Modeling

Previous sections have addressed spacecraft rotations in terms of kinematics only; this chapter introduces spacecraft dynamic modeling. This model is derived from a high fidelity example developed by Lippman, Kaufmann, and Karpenko [14], which is reproduced in Appendix C.

4.2.1 Spacecraft Dynamics Model

The state vector for the spacecraft dynamics model is a four element quaternion representation for attitude and a three element vector for angular rates of the body. The control is a three element vector of the torque on the rigid body, $\boldsymbol{\tau}_b$, subject to torque constraints on each of four reaction wheels. The configuration matrix $\mathbf{Z}$ maps the contributions of the individual reaction wheels to the spacecraft body.

Spacecraft attitude kinematics are parameterized in terms of rotational velocity using quater-

nion representation, where $\mathbf{Q}(\omega)$ is defined as [6]:

$$\mathbf{Q}(\omega) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}. \tag{4.3}$$

The transformation matrix $\mathbf{Q}(\omega)$ enables modeling of quaternion dynamics of the spacecraft using the relationship $\dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}(\omega)\mathbf{q}$. In this model, spacecraft body torque is controlled directly, subject to agility and reaction wheel limitations. Spacecraft angular acceleration can be obtained from Euler's equations of motion, where external torques are assumed to be zero [6].

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau}_b - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \tag{4.4}$$

Torque on the body of the spacecraft is represented as the opposite of internal torque, which is generated by reaction wheel momentum exchange devices for the example spacecraft:

$$\boldsymbol{\tau}_b = -\left(\mathbf{Z}\boldsymbol{\tau}_{rw} + \boldsymbol{\omega} \times \mathbf{Z}\boldsymbol{h}_{rw}\right) \tag{4.5}$$

where $\boldsymbol{h}_{rw}$ represents the angular momentum of the reaction wheels. Substituting for $\boldsymbol{\tau}_b$, Equation (4.4) becomes

$$\mathbf{I}\dot{\boldsymbol{\omega}} = -\mathbf{Z}\boldsymbol{\tau}_{rw} - \boldsymbol{\omega} \times \left(\mathbf{Z}\boldsymbol{h}_{rw} + \mathbf{I}\boldsymbol{\omega}\right) \tag{4.6}$$

Angular momentum is conserved, therefore for a zero net-bias system, $\mathbf{Z}\boldsymbol{h}_{rw} + \mathbf{I}\boldsymbol{\omega} = 0$. This simplifies the dynamics function to

$$\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left[\boldsymbol{\tau}_b\right]. \tag{4.7}$$

In this form, angular acceleration on the spacecraft, $\dot{\boldsymbol{\omega}}$, is defined as a function of the control variable, which is torque on the spacecraft body.

The inertia matrix, $\mathbf{I}$, has been simplified from the true inertia tensor describing the non-aligned principal axes of inertia and the body-fixed axes. The eigenvalues are used to represent spacecraft inertia parameters in a more convenient manner.

$$\mathbf{I} = \begin{bmatrix} 223.66 & 0 & 0 \\ 0 & 264.71 & 0 \\ 0 & 0 & 170.23 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{4.8}$$

4.2.2 Minimum Time Reorientation Problem

The goal of this effort is to improve performance over legacy rotation techniques by implementing a solution that minimizes time. Problem $\mathcal{P}$ is a minimum time optimal control problem that produces the control trajectories required to implement such a solution. The cost function, dynamics, boundary conditions, and path constraints, collected here for ease of reference, are

$$\mathcal{P}: \begin{cases} \text{Minimize:} & J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = t_f \\ \text{Subject to:} & \dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}(\omega)\mathbf{q} \\ & \dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\left[\boldsymbol{\tau}_b\right] \\ & \boldsymbol{\tau}_{rw} = \mathbf{Z}^{+}\left[\boldsymbol{\tau}_b\right] \\ & t_0 = 0 \\ & \mathbf{q}_0 = (q_0^1, q_0^2, q_0^3, q_0^4)^T \\ & \mathbf{q}_f = (q_f^1, q_f^2, q_f^3, q_f^4)^T \\ & \boldsymbol{\omega}_0 = (0,0,0)^T \\ & \boldsymbol{\omega}_f = (0,0,0)^T \\ & \mathbf{h}_1(\mathbf{x}(t), \mathbf{u}(t)) \triangleq |^{rw}\tau_i| \leq \tau_{max},\ i = 1,2,3,4 \\ & \mathbf{h}_2(\mathbf{x}(t), \mathbf{u}(t)) \triangleq |\omega_j| \leq \omega_{\max},\ j = 1,2,3 \end{cases}$$

where $\mathbf{q} \in \mathbb{R}^4$, $\boldsymbol{\omega} \in \mathbb{R}^3$, and $\mathbf{u} := \{\boldsymbol{\tau}_b \in \mathbb{R}^3\}$. τ_{rw} represents the torque exerted by each of four reaction wheel momentum exchange devices and $\mathbf{Z}^+$ is the Moore-Penrose pseudoinverse of the reaction wheel configuration matrix, $\mathbf{Z}$. All external torques are assumed to be zero.

In this model, torque on the spacecraft body is the control variable, however torque limits are imposed upon the reaction wheels and not the body axes. This is enforced by path constraint

$\mathbf{h}_1$, which is the maximum allowable torque magnitude that can be applied by each reaction wheel ($|\tau_{rw}| \leq {}^{rw}\tau_{\max}$). The second path constraint, $\mathbf{h}_2$, restricts the rotational rate around each principal axis of the vehicle to less than or equal to $\omega_{\max}$, which is $\omega_{\max} = 1\ {}^{\text{deg}}/_{\text{sec}}$ in this problem.

The potential for saturating the angular momentum of a reaction wheel is implicitly addressed by the $\mathbf{h}_2$ path constraint. By controlling the maximum angular velocity of the spacecraft, reaction wheel saturation is also limited due to the conserved momentum relationship $\mathbf{I}\boldsymbol{\omega} = -\mathbf{Z}\boldsymbol{h}_{rw}$.

4.3 Trajectory Optimization Problem

Before any attempt to solve this problem using a numerical tool can be made, the full problem formulation must be developed. The first step of this process was to define the system dynamics ($\boldsymbol{f}(t) = \dot{\mathbf{x}}(t)$), system state space ($\mathbf{x} \in \mathbb{R}^7$), system control space ($\mathbf{u} \in \mathbb{R}^3$), and constraints ($\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t))$), which are collected in Problem $\mathcal{P}$. The problem formulation techniques and principles discussed in this chapter are available in more detail in Ross [13].

4.3.1 Optimal Control Cost Functional

A cost functional, $J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f]$, must also be identified.[1] This general form of a *functional* is a function of the entire state and control trajectories for the problem as well as time, indicated by $\mathbf{x}(\cdot)$, $\mathbf{u}(\cdot)$, and t_f; it is not a function of any individual value at a point in time, which would be annotated by the more common notation $\mathbf{x}(t)$ and $\mathbf{u}(t)$.

A cost functional comprises an *endpoint cost* ($E(\mathbf{x}_f, t_f)$) and a *running cost* ($F(\mathbf{x}(t), \mathbf{u}(t), t)$) and will define the metric that the solver will minimize; Equation (4.9) is the *standard cost functional* for an optimal control problem [13].

$$J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] := E(\mathbf{x}_f, t_f) + \int_{t_o}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t), t)\ dt \tag{4.9}$$

It is crucial that the cost functional be correctly defined, otherwise an analyst may find no solution, or a solution to an unintended problem. This problem finds the fastest maneuver

[1] This cost functional is specific to the optimal control problem and should not be confused with the fitness function for a genetic algorithm, $\mathcal{J}$, discussed in Chapter 3.

to accomplish a given rotation without regard to other parameters that might limit control effort, outside of $\mathbf{h}_1$ and $\mathbf{h}_2$. Accordingly, the only factor impacting the cost functional is the maneuver final time, which is represented as an endpoint cost.

$$E(\mathbf{x}_f, t_f) = t_f \tag{4.10}$$

As there are no other considerations, running cost for this problem is zero.

$$F(\mathbf{x}(t), \mathbf{u}(t), t) = 0 \tag{4.11}$$

The cost functional for Problem $\mathcal{P}$ is then:

$$J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = t_f \tag{4.12}$$

4.3.2 Necessary Conditions for Optimality

This section develops the *necessary conditions for optimality*, which are the set of conditions that must be satisfied for a solution to be optimal. Satisfying these conditions cannot guarantee global optimality, however such a guarantee is not needed to achieve a valuable result. The minimum time maneuvers produced by this solver will be shown to be feasible, and with that result, a feasible minimum time rotation is a valuable improvement over legacy techniques, even without a guarantee of global optimality.

Problem Variable Summary

$\mathbf{x} = [q_1, q_2, q_3, q_4, \omega_1, \omega_2, \omega_3]^T$
$\mathbf{u} = [{}^b\tau_1, {}^b\tau_2, {}^b\tau_3]^T$
$F(\mathbf{x}(t), \mathbf{u}(t)) = 0$
$E(\mathbf{x}_f) = t_f$

Pontryagin Hamiltonian

A full account of the significance of the *Hamiltonian* is not practical here, but some relevant concepts will be discussed. "For the control function $\boldsymbol{u}(\cdot)$ to be optimal, Pontryagin's Principle requires that ***u globally minimize the Hamiltonian for every*** $t \in [t_0, t_f]$" [13]. The general form of the Pontryagin Hamiltonian is provided in Equation (4.13). The

following steps implement the mathematical processes needed to define the Hamiltonian function for this problem and accomplish the control minimization.

For a minimum time optimal control problem, the Hamiltonian units are time-units divided by time-units (i.e., unitless). The covector $\boldsymbol{\lambda}$ is referred to as a *costate* for the state variable covector specifically, or alternatively the *adjoint covector*. The units of the adjoint covector allow the evaluation of the algebraic functions in Equation (4.13) to be mathematically legal. The costate also varies as needed for the optimization process. Equation (4.14) is the Hamiltonian for this problem.

$$H(\mathbf{x}, \lambda, \mathbf{u}, t) = F(\mathbf{x}(t), \mathbf{u}(t)) + \boldsymbol{\lambda} \cdot \mathbf{f}(x, u) \tag{4.13}$$

$$\begin{aligned} H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}, t) = & \frac{1}{2}[\lambda_{q_1}(q_2\omega_3 - q_3\omega_2 + q_4\omega_1) \\ & + \lambda_{q_2}(-q_1\omega_3 + q_3\omega_1 + q_4\omega_2) \\ & + \lambda_{q_3}(q_1\omega_2 - q_2\omega_1 + q_4\omega_3) \\ & - \lambda_{q_4}(q_1\omega_1 + q_2\omega_2 + q_3\omega_3)] \\ & + I^{-1}_{(1,1)}\lambda_{\omega_1}{}^{b}\tau_1 \\ & + I^{-1}_{(2,2)}\lambda_{\omega_2}{}^{b}\tau_2 \\ & + I^{-1}_{(3,3)}\lambda_{\omega_3}{}^{b}\tau_3 \end{aligned} \tag{4.14}$$

Lagrangian of the Hamiltonian

The Lagrangian of the Hamiltonian introduces path constraints to the problem. This is accomplished by adding the dot product of the vector containing constraints on problem variables and a covector for measuring the path constraints, $\boldsymbol{\mu}$, such that

$$\bar{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}, t) = H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}, t) + \boldsymbol{\mu} \cdot \mathbf{h}(\mathbf{u}) \tag{4.15}$$

$$
\begin{aligned}
\bar{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}, t) = & \frac{1}{2}[\lambda_{q_1}(q_2\omega_3 - q_3\omega_2 + q_4\omega_1) \\
& + \lambda_{q_2}(-q_1\omega_3 + q_3\omega_1 + q_4\omega_2) \\
& + \lambda_{q_3}(q_1\omega_2 - q_2\omega_1 + q_4\omega_3) \\
& - \lambda_{q_4}(q_1\omega_1 + q_2\omega_2 + q_3\omega_3)] \\
& + I^{-1}_{(1,1)}\lambda_{\omega_1}{}^{b}\tau_1 \\
& + I^{-1}_{(2,2)}\lambda_{\omega_2}{}^{b}\tau_2 \\
& + I^{-1}_{(3,3)}\lambda_{\omega_3}{}^{b}\tau_3 \\
& + \mu_1{}^{rw}\tau_1 + \mu_2{}^{rw}\tau_2 + \mu_3{}^{rw}\tau_3 + \mu_4{}^{rw}\tau_4 \\
& + \mu_5\omega_1 + \mu_6\omega_2 + \mu_7\omega_3
\end{aligned}
\quad , \tag{4.16}
$$

where reaction wheel torques are defined as

$$
\begin{bmatrix} {}^{rw}\tau_1 \\ {}^{rw}\tau_2 \\ {}^{rw}\tau_3 \\ {}^{rw}\tau_4 \end{bmatrix} = \mathbf{Z}^{+} \begin{bmatrix} {}^{b}\tau_1 \\ {}^{b}\tau_2 \\ {}^{b}\tau_3 \end{bmatrix}. \tag{4.17}
$$

Adjoint Equations

The partial derivative of the Hamiltonian with respect to the costate vector provides a powerful result: the system dynamics.

$$
\frac{\partial H}{\partial \boldsymbol{\lambda}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u}, t) = \dot{\mathbf{x}}
$$

Further analysis indicates that this result is related to the *adjoint covector*, Equation (4.18). This set of equations defines important relationships in an optimal control problem and enables the minimization that leads to an extremal result.

$$
\dot{\lambda} = -\frac{\partial \bar{H}}{\partial \mathbf{x}} \tag{4.18}
$$

$$\dot{\lambda}_{q_1} = -\frac{\partial \bar{H}}{\partial q_1} = \frac{1}{2}\left[\lambda_{q_2}\omega_3 - \lambda_{q_3}\omega_2 + \lambda_{q_4}\omega_1\right] \tag{4.19a}$$

$$\dot{\lambda}_{q_2} = -\frac{\partial \bar{H}}{\partial q_2} = \frac{1}{2}\left[-\lambda_{q_1}\omega_3 + \lambda_{q_3}\omega_1 + \lambda_{q_4}\omega_2\right] \tag{4.19b}$$

$$\dot{\lambda}_{q_3} = -\frac{\partial \bar{H}}{\partial q_3} = \frac{1}{2}\left[\lambda_{q_1}\omega_2 - \lambda_{q_2}\omega_1 + \lambda_{q_4}\omega_3\right] \tag{4.19c}$$

$$\dot{\lambda}_{q_4} = -\frac{\partial \bar{H}}{\partial q_4} = -\frac{1}{2}\left[\lambda_{q_1}\omega_1 + \lambda_{q_2}\omega_2 + \lambda_{q_3}\omega_3\right] \tag{4.19d}$$

$$\dot{\lambda}_{\omega_1} = -\frac{\partial \bar{H}}{\partial \omega_1} = \frac{1}{2}\left[-\lambda_{q_1}q_4 - \lambda_{q_2}q_3 + \lambda_{q_3}q_2 + \lambda_{q_4}q_1\right] - 2\mu_5\omega_1 \tag{4.19e}$$

$$\dot{\lambda}_{\omega_2} = -\frac{\partial \bar{H}}{\partial \omega_2} = \frac{1}{2}\left[\lambda_{q_1}q_3 - \lambda_{q_2}q_4 - \lambda_{q_3}q_1 + \lambda_{q_4}q_2\right] - 2\mu_6\omega_2 \tag{4.19f}$$

$$\dot{\lambda}_{\omega_3} = -\frac{\partial \bar{H}}{\partial \omega_3} = \frac{1}{2}\left[-\lambda_{q_1}q_2 + \lambda_{q_2}q_1 - \lambda_{q_3}q_4 + \lambda_{q_4}q_3\right] - 2\mu_7\omega_3 \tag{4.19g}$$

Hamiltonian Minimization As introduced previously, an optimal solution requires the minimization of the Hamiltonian with respect to the control vector. This is the *Hamiltonian Minimization Condition*, Equation (4.20).

$$\frac{\partial \bar{H}}{\partial \mathbf{u}} = 0 \tag{4.20}$$

$$\frac{\partial \bar{H}}{\partial^b \tau_1} = I^{-1}_{(1,1)}\lambda_{\omega_1} + \mu_1 Z^+_{(1,1)} + \mu_2 Z^+_{(2,1)} + \mu_3 Z^+_{(3,1)} + \mu_4 Z^+_{(4,1)} = 0 \tag{4.21a}$$

$$\frac{\partial \bar{H}}{\partial^b \tau_2} = I^{-1}_{(2,2)}\lambda_{\omega_2} + \mu_1 Z^+_{(1,2)} + \mu_2 Z^+_{(2,2)} + \mu_3 Z^+_{(3,2)} + \mu_4 Z^+_{(4,2)} = 0 \tag{4.21b}$$

$$\frac{\partial \bar{H}}{\partial^b \tau_3} = I^{-1}_{(3,3)}\lambda_{\omega_3} + \mu_1 Z^+_{(1,3)} + \mu_2 Z^+_{(2,3)} + \mu_3 Z^+_{(3,3)} + \mu_4 Z^+_{(4,3)} = 0 \tag{4.21c}$$

Hamiltonian Value Condition In this problem, the Endpoint Lagrangian is a function of the final time and therefore the value of the Hamiltonian at t_f is not zero.

$$\bar{E}(\mathbf{x}_f, \nu, t_f) = E(\mathbf{x}_f) + \nu \cdot \mathbf{e}(\mathbf{x}_f)$$

$$\begin{aligned}\bar{E}(\mathbf{x}_f, \nu, t_f) = t_f &+ \nu_{q_1}(q_1 - q_1^f) + \nu_{q_2}(q_2 - q_2^f) + \nu_{q_3}(q_3 - q_3^f) + \nu_{q_4}(q_4 - q_4^f) \\ &+ \nu_{\omega_1}(\omega_1) + \nu_{\omega_2}(\omega_2) + \nu_{\omega_3}(\omega_3)\end{aligned} \tag{4.22}$$

$$H[@t_f] = -\frac{\partial \bar{E}}{\partial t_f} = -1 \tag{4.23}$$

This result is expected for a minimum time problem.

Hamiltonian Evolution Equation For a properly minimized Hamiltonian and an optimal trajectory, the Hamiltonian Evolution Equation provides the following relationship [13].

$$\frac{d\mathcal{H}}{dt} = \frac{\partial H}{\partial t}$$

Given a value for the Hamiltonian at t_f obtained from the Hamiltonian Value Condition, and information about the derivative of the Hamiltonian with respect to time, a plot of the Hamiltonian can be used to verify optimality. In this problem, the Hamiltonian is not a function of time.

$$\frac{\partial H}{\partial t} = 0, \ \therefore \ \frac{d\mathcal{H}}{dt} = 0 \tag{4.24}$$

It is therefore expected to be constant as a function of time.

Transversality Equations Transversality allows the discovery of boundary conditions that may not be provided in the problem formulation by determining the final costate value as the partial derivative of the Endpoint Lagrangian with respect to the final state whose condition is not known, Equation (4.25).

Using the endpoint conditions and the Endpoint Lagrangian, where

$$\mathbf{e}(\mathbf{x}_f) := \begin{bmatrix} \mathbf{q}_f - \begin{bmatrix} q_1^f \\ q_2^f \\ q_3^f \\ q_4^f \end{bmatrix} \\ \boldsymbol{\omega}_f \end{bmatrix} = [0]$$

and

$$\begin{aligned}\bar{E}(\mathbf{x}_f, \nu, t_f) = t_f &+ \nu_{q_1}(q_1 - q_1^f) + \nu_{q_2}(q_2 - q_2^f) + \nu_{q_3}(q_3 - q_3^f) + \nu_{q_4}(q_4 - q_4^f) \\ &+ \nu_{\omega_1}(\omega_1) + \nu_{\omega_2}(\omega_2) + \nu_{\omega_3}(\omega_3)\end{aligned}, \tag{4.22}$$

the costate values λ_i at t_f are defined. In this problem, all required boundary conditions are provided, so transversality principles are not expected to provide new information. This is indicated by the final costate value being equal to the endpoint covector element for each state, which is also unknown.

$$\boldsymbol{\lambda}_{t_f} = \frac{\partial \bar{E}}{\partial \mathbf{x}_f} \tag{4.25}$$

$$\lambda_{q_{1_f}} = \frac{\partial \bar{E}}{\partial q_{1_f}} = \nu_{q_1} \tag{4.26a}$$

$$\lambda_{q_{2_f}} = \frac{\partial \bar{E}}{\partial q_{2_f}} = \nu_{q_2} \tag{4.26b}$$

$$\lambda_{q_{3_f}} = \frac{\partial \bar{E}}{\partial q_{3_f}} = \nu_{q_3} \tag{4.26c}$$

$$\lambda_{q_{4_f}} = \frac{\partial \bar{E}}{\partial q_{4_f}} = \nu_{q_4} \tag{4.26d}$$

$$\lambda_{\omega_{1_f}} = \frac{\partial \bar{E}}{\partial \omega_{1_f}} = \nu_{\omega_1} \tag{4.26e}$$

$$\lambda_{\omega_{2_f}} = \frac{\partial \bar{E}}{\partial \omega_{2_f}} = \nu_{\omega_2} \tag{4.26f}$$

$$\lambda_{\omega_{3_f}} = \frac{\partial \bar{E}}{\partial \omega_{3_f}} = \nu_{\omega_3} \tag{4.26g}$$

4.3.3 Scaling and Balancing

Problems that deal in orbital magnitudes have the potential to develop significant mismatches of magnitude between the states and controls and their corresponding covectors. Therefore, scaling and balancing techniques were employed to make the models more robust and better

conditioned for numerical analysis. Scaling and balancing techniques for optimal control problems are discussed at length by Ross et al. [15].

Canonical Scaling

First, canonical scaling was considered using common units for mass, length, time, and angle. To reduce run times and improve the ability of DIDO to solve otherwise poorly conditioned problems, the dynamics were considered according to the following relationships.

The variables must be defined in terms of their scaled states; a given variable's scaled state is indicated by an overbar and the selected scale factors are indicated by a capitalized constant (e.g., $x = \bar{x}\ X$).

$$q = \bar{q} \tag{4.27a}$$

$$\omega = \frac{\theta}{t} = \frac{\bar{\theta}\ A}{\bar{t}\ T} = \bar{\omega}\ \frac{A}{T} \tag{4.27b}$$

$$\tau = \text{Force} \cdot \text{radius} = F \cdot r = \bar{F}\ \frac{M\ L}{T^2} \cdot \bar{r}\ L = \bar{\tau}\ \frac{M\ L^2}{T^2} \tag{4.27c}$$

$$\mathbf{I} = \bar{\mathbf{I}}\ ML^2 \tag{4.27d}$$

The goal of canonical scaling is to ensure that states fall within a desirable range (e.g., $\mathbf{x} \in [-1, 1]$). Quaternions fall into this range by definition, and are therefore not adjusted by a scale factor in this problem.

Using the scaled variable relationships, scaled dynamics equations are derived.

$$\begin{aligned} \frac{d\mathbf{q}}{dt} = \frac{d\bar{\mathbf{q}}}{d\bar{t}\ T} &= \frac{1}{2}\mathbf{Q}(\bar{\omega})\left(\frac{A}{T}\right)\bar{\mathbf{q}} \\ \dot{\bar{\mathbf{q}}} &= \frac{A}{2}\mathbf{Q}(\bar{\omega})\bar{\mathbf{q}} \end{aligned} \tag{4.28}$$

Rotation rate dynamics are a function of torque.

$$\begin{aligned} \frac{d\boldsymbol{\omega}}{dt} = \frac{d\bar{\omega}}{d\bar{t}}\ \frac{A}{T^2} &= \left(\bar{\mathbf{I}}\ ML^2\right)^{-1}\ [\bar{\boldsymbol{\tau}}_b]\ \ \frac{ML^2}{T^2} \\ \dot{\bar{\omega}} &= \frac{1}{A}\bar{\mathbf{I}}^{-1}\ [\bar{\boldsymbol{\tau}}_b] \end{aligned} \tag{4.29}$$

For this model, the only canonical unit that remains for the given dynamics is angle, A. While the goal of canonical scaling is to ensure that states fall within a desirable range, it cannot also guarantee that the states and their covectors are properly *balanced*. Properly balanced variables have vectors and covectors that fall within similar orders of magnitude, which is desirable for numerical computation. With only one canonical unit to adjust and the inability to simultaneously ensure effective balancing, canonical scaling is not a particularly useful technique for this application.

Designer Units

Instead of canonical scaling, scaling with designer units was used because it allows for each state-costate, control-covector, or path-covector pair to be scaled individually, which also allows for balancing [15]. Using the selected scale factors, well-scaled and balanced states, constraints, and covectors were achieved, ensuring that well-conditioned parameters were provided to the optimization solver. Equation (4.30) lists the scale factors used.

$$\begin{aligned} K_{q_1} &= 0.2 & K_{h_{1,1}} &= 3 \\ K_{q_2} &= 0.2 & K_{h_{1,2}} &= 1 \\ K_{q_3} &= 0.2 & K_{h_{1,3}} &= 6 \\ K_{q_4} &= 0.1 & K_{h_{1,4}} &= 1 \\ K_{\omega_1} &= 0.01 & K_{h_{2,1}} &= 0.02 \\ K_{\omega_2} &= 0.01 & K_{h_{2,2}} &= 0.02 \\ K_{\omega_3} &= 0.05 & K_{h_{2,3}} &= 0.02 \end{aligned} \tag{4.30}$$

4.4 Example Solution

An example trajectory optimization problem was solved using DIDO in MATLAB. This maneuver is a rest-to-rest maneuver (i.e., zero rotational velocity at the beginning and end of the maneuver) with the quaternion boundary conditions $\mathbf{q}_0 = (0, 0, 0, 1)^T$ and $\mathbf{q}_f = (0.2660, 0.4234, 0.0472, 0.8647)^T$.

4.4.1 Results and Analysis

Figure 4.2 shows the scaled quaternion state trajectories and their respective scaled costates; Figure 4.3 shows the scaled rotational rate state trajectories and their respective scaled costates. The control trajectories are depicted in Figure 4.4. The minimum time required for this slewing maneuver is 49.04 seconds with this problem formulation. Figure 4.5 is a three-dimensional rendering of the rotation of this spacecraft and the boresight ground trace.

In Figure 4.3, the effects of the state path constraint can be seen early in the maneuver, when the angular rates plateau. This behavior is examined in more detail in Section 4.4.2.

The fluctuations in the control trajectories (Figure 4.4) that start before the ten second mark and end after the 40 second mark is a byproduct of the active velocity state constraint, which is preventing the steady application of torque so as not to exceed the path constraint defined by $\mathbf{h}_2$. Before and after this time frame, the path constraint is not active and traditional switching function behavior is observed. This solution is consistent with optimal control theory and the required conditions described by Karush, Kuhn, and Tucker [13], which will be addressed in more detail in the next section.

4.4.2 Verification and Validation

This problem formulation will be verified and validated along two lines of inquiry: investigating the feasibility of the solution and checking for adherence to the necessary conditions for optimality.

Model Feasibility Conditions

The feasibility of this solution will be addressed in two main ways: whether or not the model adheres to the problem dynamics while satisfying path constraints and whether or not the desired boundary conditions have been achieved. Visually inspecting Figure 4.4, it appears to be a *switching function* type controller as expected. This is evident from the control trajectories that are alternately at the upper bound or the lower bound, with the exception of fluctuating inputs in the control trajectories while the path constraint $\mathbf{h}_2$ is active.

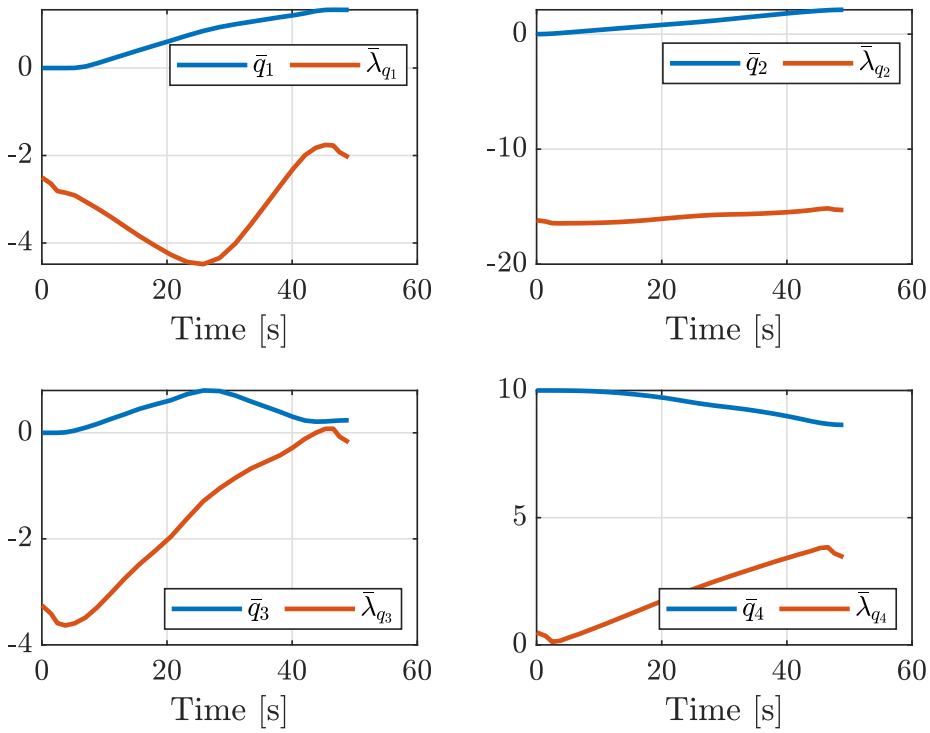


Figure 4.2. Scaled quaternion states and corresponding costates.

The control trajectory solution was propagated through the plant dynamics, and the resulting state trajectories are overlaid with the DIDO solver trajectories and displayed in Figure 4.6. From this figure, it is apparent that the plant dynamics have been satisfied and that the initial and final boundary conditions have also been met. Furthermore, this result supports the conclusion that the fluctuations in the control trajectories due to the active path constraint are not degrading the overall solution.

Model Optimality Conditions

While adhering to the necessary conditions for optimality developed for this model is a requirement for an optimal solution, that is distinct from satisfying the *sufficient conditions* for optimality which would guarantee an optimal solution [13]. Here, evidence is presented that supports the conclusion that the solution may be optimal, or even is likely to be optimal. However, this cannot guarantee that the solution is globally optimal. This process also seeks to uncover any evidence that the necessary conditions have not been satisfied. Under such conditions, the solution *cannot* be optimal.

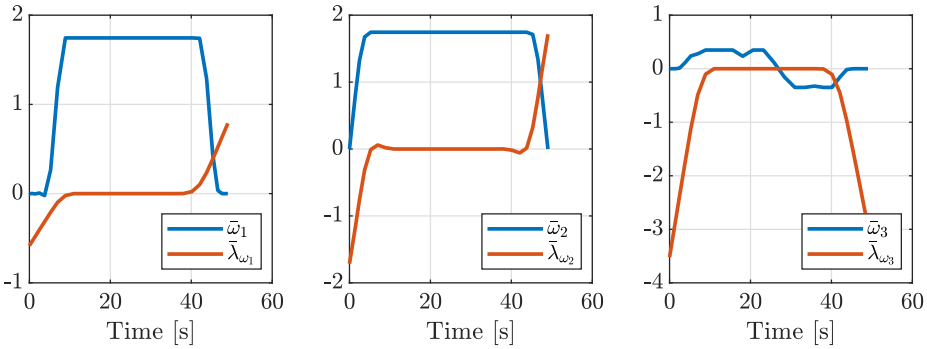


Figure 4.3. Scaled angular rate states and corresponding costates.

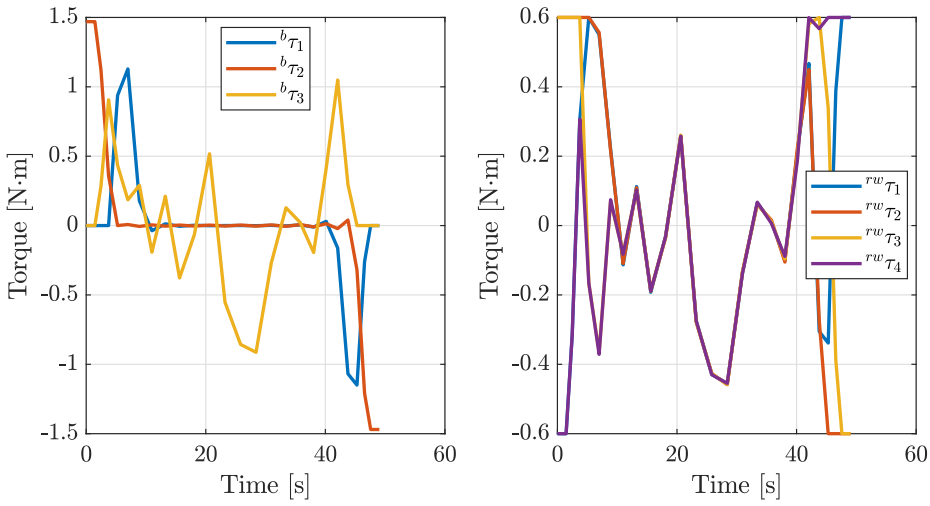
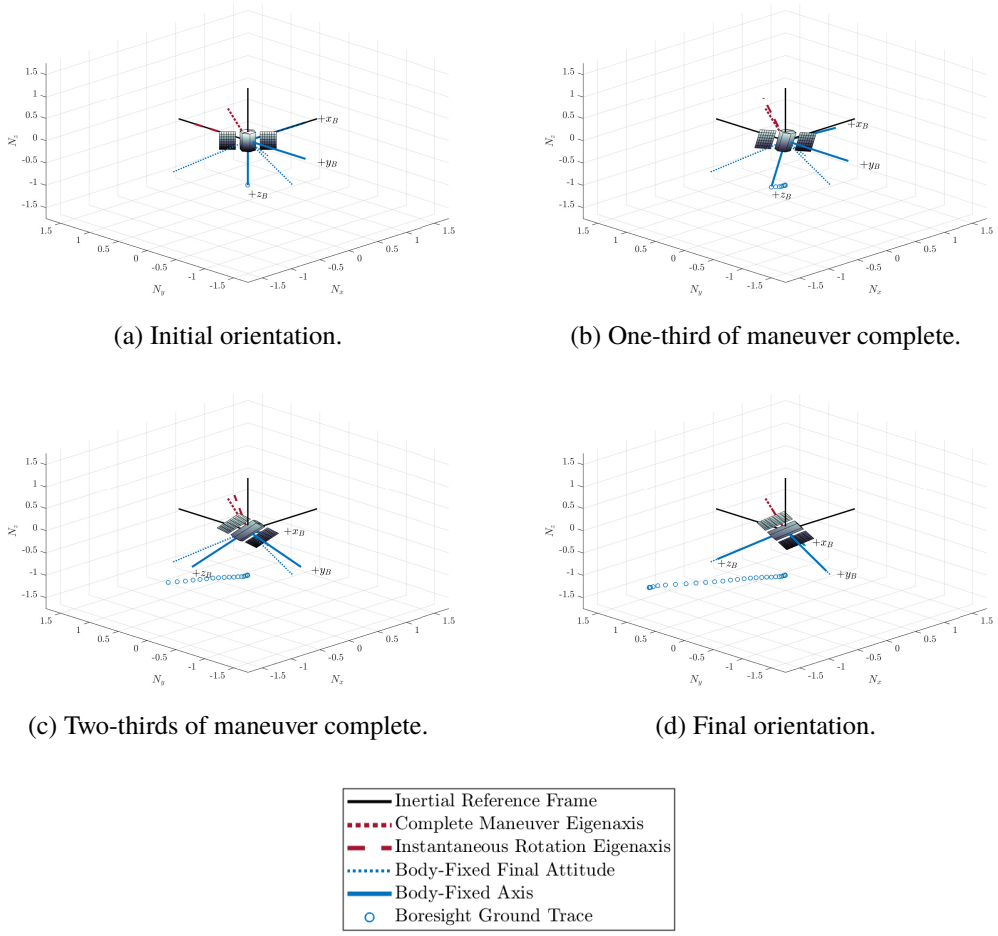


Figure 4.4. Control trajectories and control torques mapped to reaction wheels.

Costate Behavior From Equations (4.19a) through (4.19g), it can be inferred that the adjoint equations will likely be non-constant, but little else can be guaranteed. Figures 4.2 and 4.3 do indeed show non-constant costate behavior.

Hamiltonian Behavior According to the Hamiltonian Evolution Equation, Equation (4.24), the ordinary derivative of a properly minimized Pontryagin Hamiltonian with respect to time will be equal to the partial derivative of the Pontryagin Hamiltonian with respect to time, which in this case is zero, as the Hamiltonian is not an explicit function of time.



(a) Initial orientation.

(b) One-third of maneuver complete.

(c) Two-thirds of maneuver complete.

(d) Final orientation.

Figure 4.5. Time lapse of slewing maneuver.

It is further shown from the Hamiltonian Value Condition, Equation (4.23), that the Hamiltonian at the final time is equal to -1. As the derivative with respect to time is zero, the Hamiltonian should be constant with respect to time. This will allow the Hamiltonian resulting from the numerical solution to be inspected for constancy. While a constant Hamiltonian does not guarantee optimality, it is a requirement for optimality [13].

The Pontryagin Hamiltonian is plotted against time in Figure 4.7. Given the scaling of the vertical axis and the discretization of the problem by the DIDO solver, this behavior is consistent with a properly minimized Hamiltonian and is evidence that an optimal solution has been found.

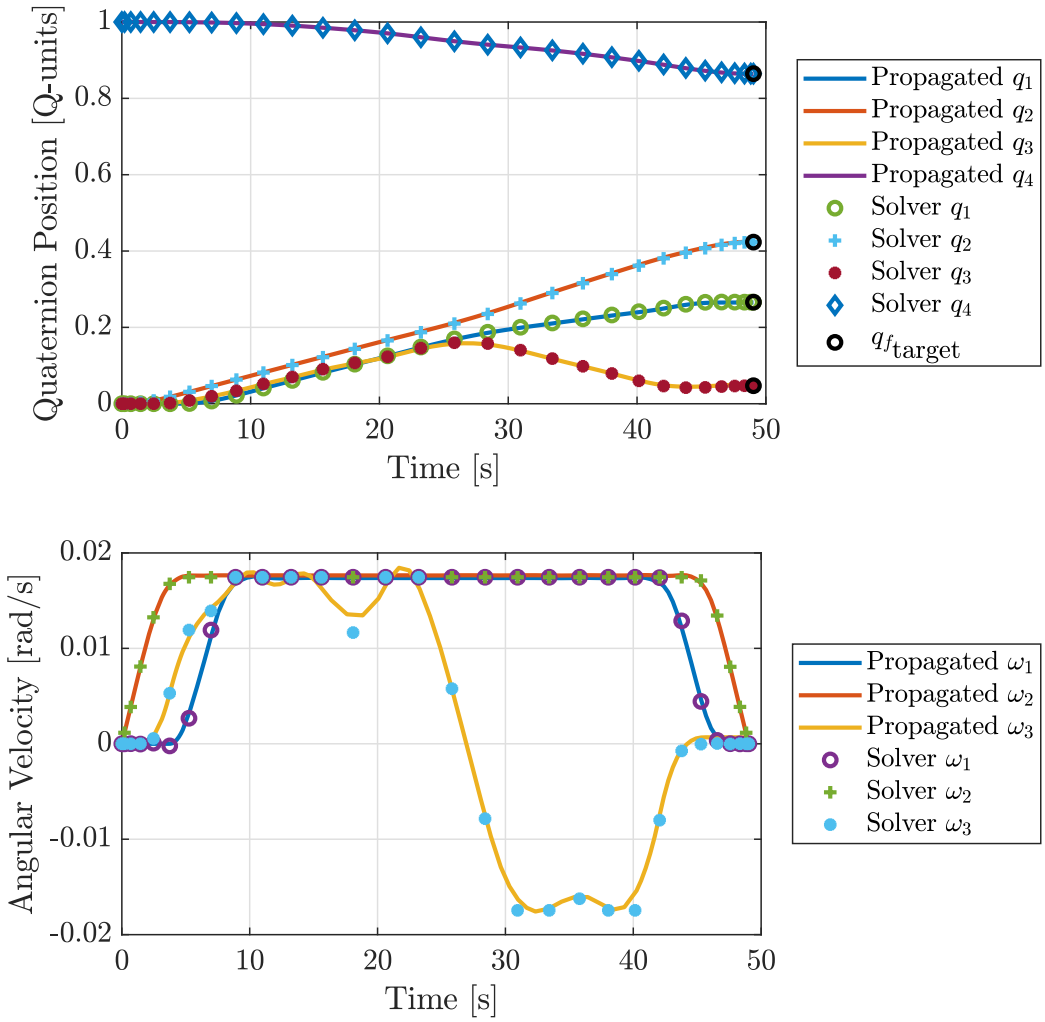


Figure 4.6. Verified state trajectories in engineering units.

Control Constraint Behavior Karush, Kuhn, and Tucker developed two conditions that describe control variable behavior: the *stationarity* condition and the *complementarity* condition [13]. These conditions rely upon the Lagrangian of the Hamiltonian developed for this problem, Equation (4.16). The Lagrangian of the Hamiltonian in its general form makes use of the Lagrange multiplier μ.

$$\bar{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}, t) = H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}, t) + \boldsymbol{\mu} \cdot \mathbf{h}(\mathbf{u})$$

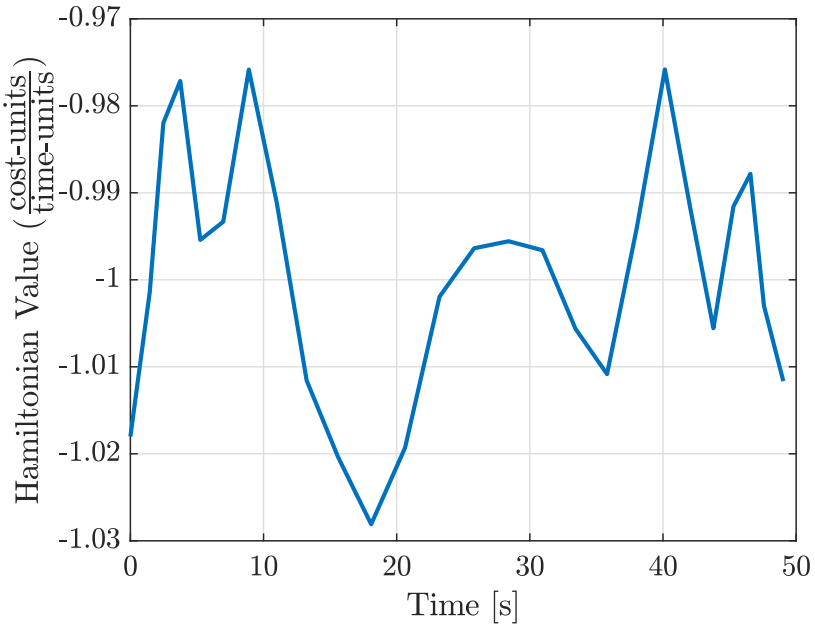


Figure 4.7. Hamiltonian as a function of time.

The μ multiplier is allowed to vary as necessary to ensure that $\dfrac{\partial \bar{H}}{\partial \mathbf{u}} = 0$ is always possible. This is the definition of the stationarity condition, and was used in the development of the necessary conditions for this problem [13].

The complementarity condition describes the behavior of the control covector, $\boldsymbol{\mu}$, which varies in response to the control constraint, $\mathbf{h}_1(\mathbf{u})$. The complementarity condition is defined [13] as

$$\mu_i \begin{cases} \leq 0 & \text{if} \quad u_i = u^{\text{lower}} \\ = 0 & \text{if} \quad u^{\text{lower}} < u_i < u^{\text{upper}} \\ \geq 0 & \text{if} \quad u_i = u^{\text{upper}} \end{cases} .$$

To check for adherence to the stationarity and complementarity conditions, the control trajectories are plotted over their corresponding control covector, μ_i in Figure 4.8. While there are slight deviations from strict adherence to the values of μ relative to the zero crossings of the corresponding control trajectory, these are artifacts of the numerical solution process. The effects of the binding path constraint $\mathbf{h}_1$ are again evident through the fluctuations in the middle portion of the reaction wheel torque trajectories. During this condition the values for μ are zero or approach zero because extremal control effort is not being applied.

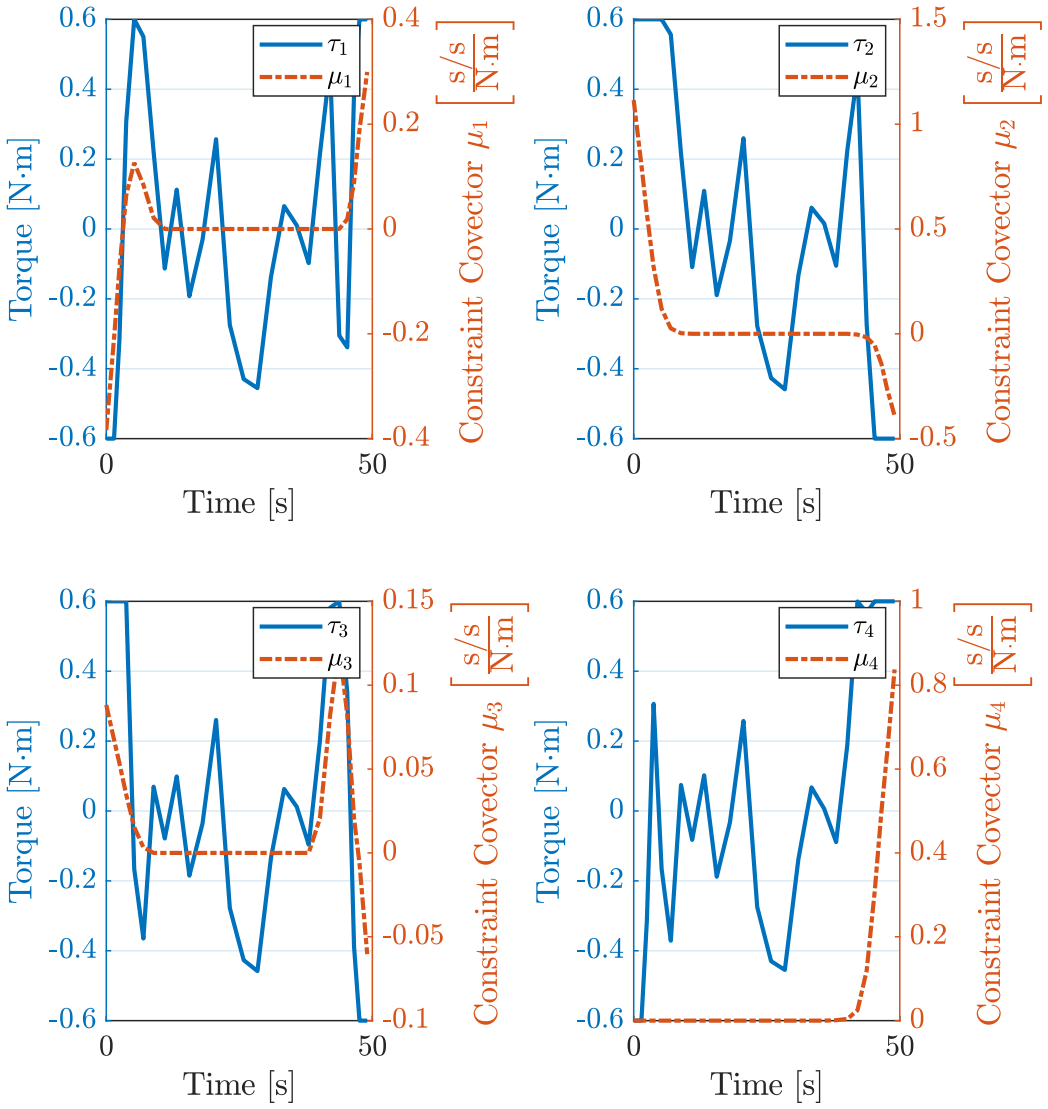


Figure 4.8. Reaction wheel torque trajectories and corresponding covectors.

Adherence to the complementarity principles is observed: μ values are negative when the constrained parameter is at its lower bound and the values are positive when it is at its upper bound.

Path Constraint Covector Behavior The second set of constraints, $\mathbf{h}_2$, is defined by the three angular rate state variables and ensures that the rotational rate about each principal axis of the vehicle is kept below the maximum allowable value. The explicit value of μ_j over

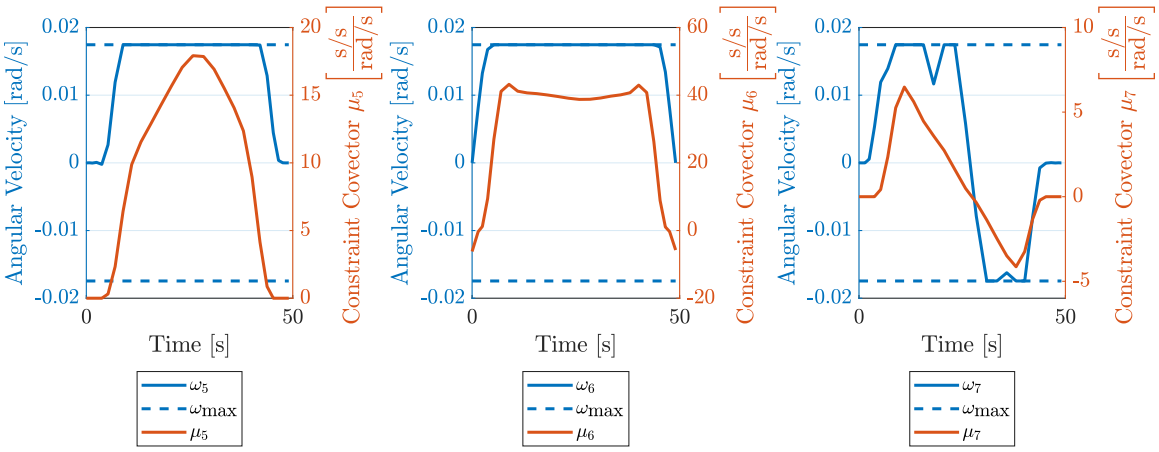


Figure 4.9. State variable path constraint plotted with its covector.

time is not readily verifiable; however, the behavior can be verified graphically. Figure 4.3 depicted each scaled state's behavior over time, and Figure 4.9 depicts the path constraint $\mathbf{h}_2$ using unscaled inputs, along with the control covector μ_j and the maximum allowable vehicle rotation rate, $\omega_{\max}$.

This plot demonstrates that the $\mathbf{h}_2$ path constraint becomes active for each rotational axis during the middle portion of the maneuver, where μ_j is non-zero and $|\omega_j| = \omega_{\max}$.

To check for effective scaling of the model constraints, the scaled constraints and their covectors are presented in Figure 4.10; here, it is evident that the constraints are properly balanced for efficient computation.

4.5 Summary

Performing the same example rotation described in this chapter using the legacy eigenaxis technique requires 68.0 seconds, compared to 49.0 seconds for the minimum time maneuver derived here. That is a 27.9% improvement over the status quo eigenaxis maneuver for a roughly 1 radian slew. A performance improvement of this magnitude will likely appeal to any spacecraft operations manager. As these maneuvers have been demonstrated on orbit, the last remaining hurdle to wide implementation of these techniques is the ability to incorporate them into mission planning processes. This optimal maneuver model will be the basis for fast-running machine learning models that can approximate optimal slew times

with enough accuracy to enable enterprise-wide implementation of minimum time satellite rotations.

This chapter has established the process needed to develop minimum time slewing maneuvers, which will enable significant gains if incorporated into mission planning. The development of the necessary conditions and extensive verification and validation make this model a reliable framework from which a library of slewing maneuvers and their associated time-optimal slew times can be developed. In Chapter 5, such a library will be generated, and that data will be used to train supervised machine learning models. These, in turn, will be used to approximate optimal slew times for path planning purposes.

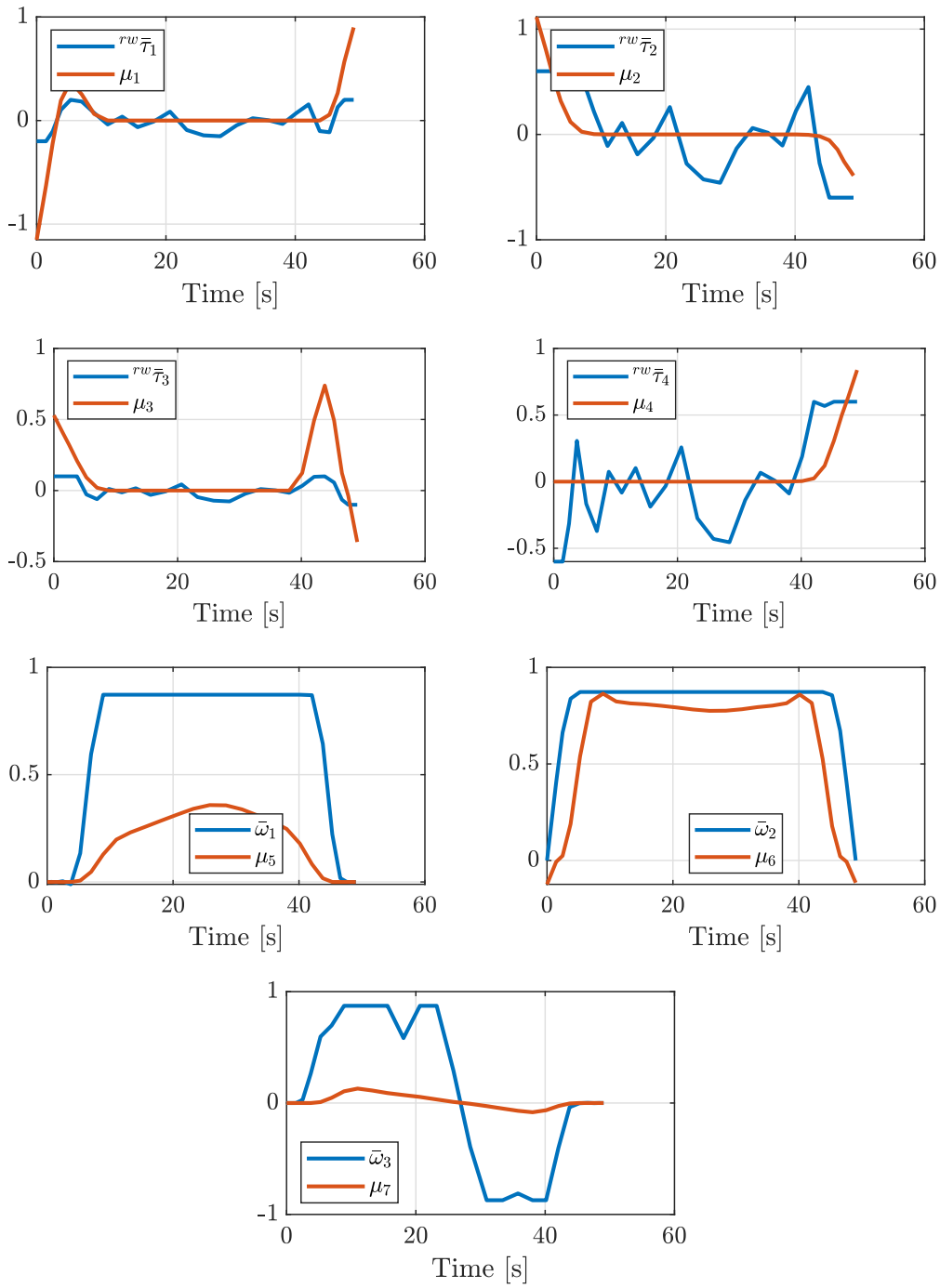


Figure 4.10. Scaled path constraints and constraint covectors.

CHAPTER 5:
Approximating Optimal Slew Times Using Machine Learning

The task at hand now is to incorporate minimum time maneuvers into a mission planner. Figure 5.1 highlights the function under consideration in this chapter as it relates to the larger system. As discussed previously, the computational resources required to produce a single minimum time maneuver make it impractical to incorporate optimal maneuver solvers directly into an iterative path planner such as Remote Sensing Iterative Planner (RSIP). To date, simple-to-calculate but physically inefficient solutions have been used to generate feasible mission plans (e.g., the taxicab and eigenaxis maneuvers). To enable improvements to the planning process, supervised machine learning concepts can be used to incorporate approximations of certain aspects of optimal spacecraft slews into mission planners without otherwise changing or modifying the algorithm.

Fortunately, the complete solution for an optimal maneuver it not required to generate a mission plan sequence. Principally, only those concepts required by the fitness function, Equation (3.7), need be produced. Externally, the fitness function requires maneuver time and the value of the target in question. The value of the target in question is defined by the planner, so only the maneuver slew time must be approximated.

$$\mathcal{J}_4 = \sum_{i=1}^{n-1} \alpha t_i + \beta \left[v_{\max} - \sum_{j=1}^{n_c} v_j \right] \tag{3.7}$$

A slew time can be derived from a starting orientation and a final orientation in the orbital reference frame, discussed in Chapter 2. The mission planner uses these attitude pairs to determine slew times for legacy techniques, just as the optimal solver uses these attitude inputs for the same purpose, as addressed in Chapter 4.

As RSIP already provides initial and final attitudes for a maneuver, a supervised learning model needs only to approximate the optimal maneuver time, t_i, from these attitudes and

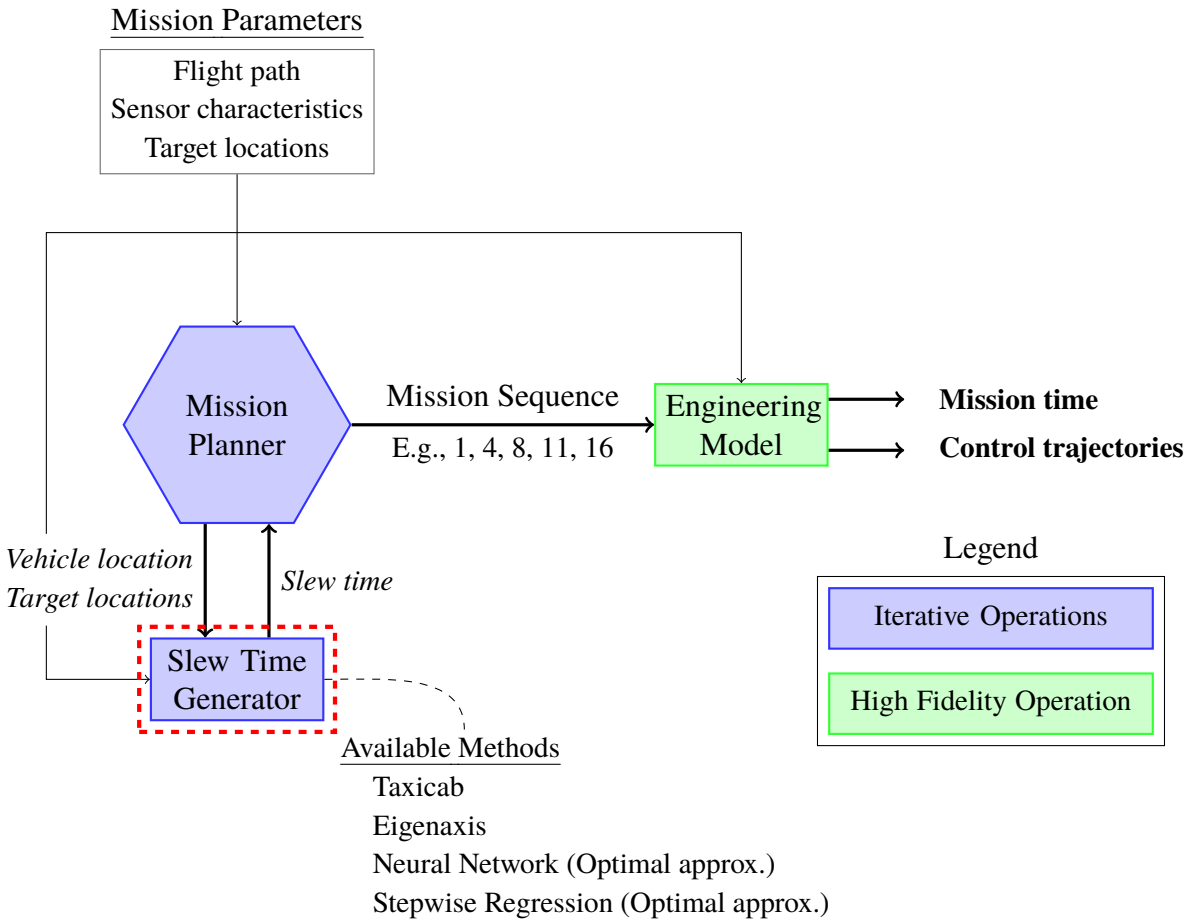


Figure 5.1. Over-subscription planning flow diagram (Slew Time Generator).

provide that input to the planner fitness function for evaluation, in the same fashion as the legacy approach.

This chapter will introduce supervised learning, the methods for generating a set of a training data that enables the supervised learning process, and the resulting surrogate models that approximate minimum time maneuvers. Two supervised learning methods will be discussed: neural networks and stepwise multiple linear regression. Mission planner results using both methods will be presented and the relative strengths of each will be discussed.

5.1 Supervised Learning for Maneuver Time Approximation

Supervised learning is a machine learning method that relies on a labeled data set of known inputs and outputs. This labeled data is used to initially fit a model by *training* model parameters to minimize prediction error based on the known true outputs, given a set of corresponding inputs. Supervised learning is the appropriate approach here because of the availability of a labeled dataset for training (i.e., a dataset of optimal maneuvers and their corresponding optimal slew times). This contrasts with the *unsupervised learning* approach, in which input data is not associated with a response variable, and is instead used to find patterns or to group inputs. This approach would be appropriate in cases where the goal of the model is to understand relationships between the input variables by grouping them into output classes. In this thesis, the labeled dataset of minimum time maneuvers for supervised learning was created using the engineering model discussed in Chapter 4.

5.1.1 Supervised Learning Approaches

Within supervised learning, models are typically either *regression* models, associated with continuous data, or *classification* models, which are appropriate in cases of discrete data. For maneuver time approximation, the input data is a vector of real data that represents a spacecraft orientation or rotation. The associated output data is the time required to perform the rotation using a minimum time maneuver. These data are continuous, and therefore regression approaches are employed here.

Two common frameworks for supervised learning problems with continuous data are neural networks and linear regression. These two techniques were explored for this problem, leading to two models for optimal maneuver time approximation, one based on each framework. These "models of models" are referred to as *surrogate models*, or *metamodels* of the system they are intended to approximate [16]. In this case, they are surrogate models of the engineering model from Chapter 4. Broadly speaking, metamodels are light weight, fast-running approximations of other models, which in this case is a high fidelity optimization model that requires significant computational resources to produce a minimum time maneuver solution. Of course, without the high fidelity model, accuracy is sacrificed; by definition, surrogate models are simply approximations of the high fidelity solution.

5.1.2 Surrogate Model Concept

A machine learning surrogate model is developed in three main parts. First, the training data to be used must be developed or obtained. The process of intelligently sampling the input space is called *Design of Experiments* (DoE) [17], and will be discussed further in the next section. Once the desired set of model inputs is generated, the corresponding model outputs must be generated. This is accomplished through a *wrapper function*, which is the application of the full fidelity engineering model to the input DoE to obtain the labeled target outputs. Once the inputs and outputs are associated, they are referred to as *labeled data* and then used for model training. Before a supervised learning model can be trained on the labeled data, it must be created in a software package, which necessitates decisions about how the model will operate. For this remote sensing application, some of these decisions have already been made (i.e., the decision to develop a neural network and a linear regression model). Additional decisions remain, however.

For the neural network, the type of network must be decided (e.g., a feed-forward network, a convolutional network, a recurrent network), the number of hidden layers must be assigned, the number of nodes in each layer must be decided, and the activation function used in each node must be chosen. These choices are made by the designer and are called *hyperparameters*, which are not adjusted by the model during training [18]. These options will be explored in detail in subsequent sections. For the linear regression model, choices must be made about the error function used, method for determining regression terms, and statistical confidence thresholds.

With these three broad steps completed—selection of input DoE, generation of labeled data with the engineering model, and selection of hyperparameters for the surrogate model—model training and evaluation can be accomplished. This thesis will evaluate multiple approaches and compare them for effectiveness.

5.2 Design of Experiments

An effective supervised learning model depends on its training data. A model with poorly designed or incomplete training data is likely to have decreased performance, therefore DoE theory for designing the data collection process governs this discipline. DoE aims to intelligently sample the input space so that maximum information can be gleaned from

the fewest number of trials. An effective DoE will also ensure that the entire relevant input space is sampled, allowing a model to interpolate future inputs, instead of having to extrapolate conclusions that are not included in the training data.

This model uses training data that is analytically generated and noiseless. The only concerns are developing the correct data and committing to the processing time to produce it. This is distinct from any physical experiments which may require limited resources, such as reagents, equipment, personnel support, etc. Furthermore, analytic results do not have to account for noise in the data, which would increase output error and could potentially mask important relationships between the input data and maneuver time. It remains important, however, to ensure than the entire sample space is covered and that the model is designed with enough flexibility to capture the relevant features that predict maneuver time.

Six significant principles that apply to DoE are assembled here for discussion [17]. These principles guided the development of the DoE used for RSIP.

1. *Interactions*. While individual factors can play a large role in the output of a process, the interactions between these factors can also have significant impacts.
2. *Scarcity of Effects.* Most outcome variability tends to be the result of a subset of design variables, rather than the full set of variables.
3. *Hierarchy of Factors*. Variability tends to result mostly from the main (first order) effects, followed by second order effects and third order effects, and so on.
4. *Heredity of Factors*. If a higher order effect is deemed important (e.g., a second order effect), then the lower order effects that make up the higher order effect (e.g., its composite first order effects) are also important and should be included in the model.
5. *Aliasing*. Also referred to as *confounding*, the source of some variability may not be identifiable due to a DoE that does not include every possible combination of input factors.
6. *Orthogonality*. Trials within a DoE should contain as little overlap in information as possible. This can be complicated by inputs which are not independent of one another.

Three of these principles are inherently respected by the structure of the models chosen for this thesis, as well as by the training algorithms used to determine final weights, biases, and coefficients. Scarcity of effects, hierarchy of factors, and heredity of factors principles

are all implemented by the algorithms that determine how to weight, include, or exclude different factors. The role of interactions, aliasing, and orthogonality are addressed in the creation of the DoE itself, which will be discussed further in the next section. Broadly speaking, the models employed here account for interactions and the DoE minimizes the potential for aliasing while maximizing orthogonality.

5.2.1 Space Filling Design

Maneuver time approximation requires a DoE that is appropriate for continuous data on the interior of a sample space. That is to say, there is no need for the model to extrapolate inputs that are outside the domain of the training set. The domain that covers rotations is limited by geometry (i.e., there is no need to model rotations in excess π radians for this planner). The class of approaches that satisfy these requirements is *Space Filling Design* [17], an example of which is a Latin Hypercube shown in Figure 5.2.

Latin Hypercube Sampling

A Latin Hypercube maximizes "the minimum distance between design points but requires even spacing of the levels of each factor. This method produces designs that mimic the uniform distribution" [17]. This occurs by dividing each input variable domain into n uniform subdivisions (where n is the number of data points) and placing a single data point within each subdivision in such a way that information overlap is minimized by maximizing orthogonality between the data points.

Recall that the inputs for a rotation can be expressed in multiple ways: as a pair of quaternions including an initial and a final quaternion ($\mathbf{q}_i$ and $\mathbf{q}_f$), as a relative change in quaternion units ($\Delta\mathbf{q}$), or as the Euler parameters that define the change in terms of an axis and angle of rotation ($\hat{\mathbf{e}}$ and ψ).

The domain of values for quaternion inputs is $q_i \in [-1, 1]$ subject to $\sum_i q_i^2 = 1$ and for Euler parameters the domain of inputs is $e_i \in [-1, 1]$ subject to $\sum_i e_i^2 = 1$ for the eigenaxis elements and $\psi \in [-\pi, \pi]$ for the rotation angle. The input domains include continuous data and the domain is bounded without the risk that future inputs may exceed the training domain, because all rotations can be represented by elements within these bounds.

In summary, a Latin Hypercube uses random sampling techniques that cover the interior

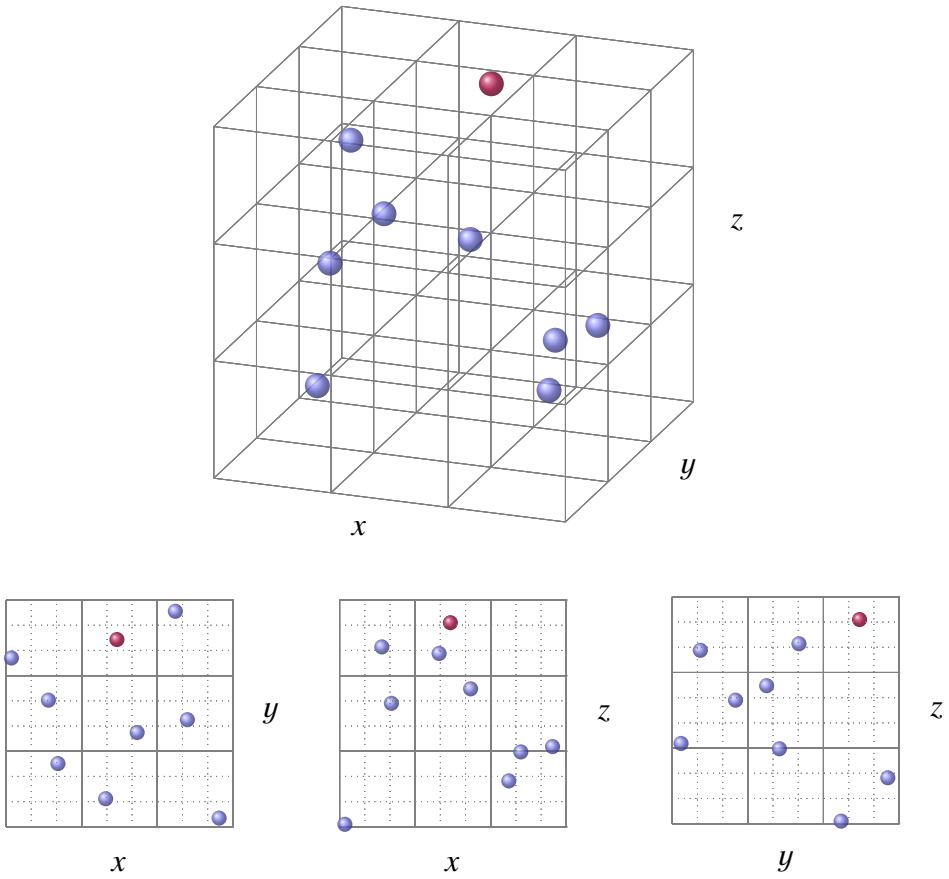


Figure 5.2. Example Latin Hypercube.

of the design space, and maximize information while minimizing overlap between samples over p variables. The n values of data points are distributed over equal sized intervals along each variable.

RSIP Latin Hypercube

Input sampling for RSIP was accomplished using a MATLAB generated Latin Hypercube with four variables and 2000 samples, however initial training data included only a subset of randomly selected vectors. These vectors represent sensor boresight orientations, Figure 5.3.

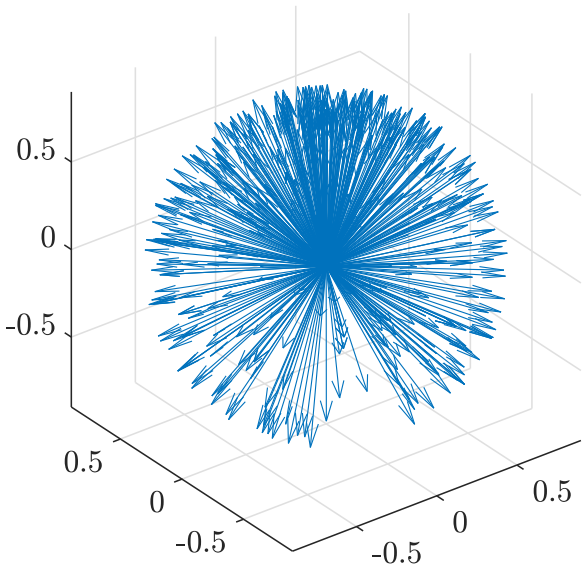


Figure 5.3. Input data set eigenaxes.

5.2.2 Input Parameterization

The hypercube generated for optimal maneuver time approximation consisted of four variables (e_1, e_2, e_3, ψ) and 500 samples ($n = 500$), chosen because Euler parameters can be used to define a rotation independently, or transformed into quaternion representation for the same purpose. Euler parameters are used as the starting point because the eigenaxis ($\hat{\mathbf{e}}$) is independent of the eigenangle (ψ); if quaternion samples had been chosen directly, random sampling would need to ensure that the quaternion definition is respected. Recall that a quaternion is

$$\mathbf{q} = \begin{bmatrix} e_1 \sin(\psi/2) \\ e_2 \sin(\psi/2) \\ e_3 \sin(\psi/2) \\ \cos(\psi/2) \end{bmatrix} \tag{2.22}$$

and therefore $q_{1,2,3}$ are not independent of q_4 due to the eigenangle, and any analysis would need to consider these relationships. A simpler approach is to simply generate the rotations using Euler parameters and convert to quaternion when convenient.

Three configurations were considered for input parameterization:

1. 8×1 vector of initial quaternion attitude ($\mathbf{q}_0$) and final quaternion attitude ($\mathbf{q}_f$) in the

orbital reference frame.

2. 4×1 vector of relative quaternion rotation ($\Delta\mathbf{q}$) between the initial and final quaternion attitudes.
3. 4×1 vector of Euler parameters (i.e., 3×1 eigenaxis of rotation ($\hat{\mathbf{e}}$) and the angle of rotation about that axis, the eigenangle [ψ]).

The first option consisting of the starting and stopping orientation of a rotation was discarded because of the superfluous information it injects into the model. For a spacecraft, the important information that affects the reorientation of the vehicle is the change of the body attitude in the body-fixed reference frame. The starting orientation of the spacecraft is relevant to the orientation of the vehicle in the inertial and orbital frames, but not relevant to the spacecraft itself, which only experiences rotations relative to the body frame. While orientation relative to the inertial or orbital frame is important to the planning problem, it is not required to compute a specific reorientation maneuver, as its information can easily be condensed into one of the other two forms.

The two remaining candidates are both 4×1 vectors: $\Delta\mathbf{q}$ and $[\hat{\mathbf{e}}, \psi]^T$. Given the risk of introducing undesired aliasing into the DoE, the Euler parameters are the more desirable choice as they are the more direct mathematical representation of a rotation. Quaternion representation risks aliasing the effects of the $\hat{\mathbf{e}}$ and ψ elements for $q_{1,2,3}$ because $q_{1,2,3}$ are a function of $\hat{\mathbf{e}}$ and ψ while q_4 is a function of ψ alone. Therefore in quaternion representation, there are no inputs that are functions of the eigenaxis exclusively. Figures 5.4 and 5.5 depict the fit of each factor by the target maneuver time; patterns resulting from these distinctions are apparent.

Relative Quaternion

The relative quaternion in the body frame ($\Delta\mathbf{q}$) can be computed from initial and final orientations in the orbital frame using the orthonormal quaternion transmuted matrix $\mathbf{Q}$ [6].

$$\Delta\mathbf{q} = \mathbf{Q}^T\mathbf{q}_f = \begin{bmatrix} q_0^4 & -q_0^3 & q_0^2 & q_0^1 \\ q_0^3 & q_0^4 & -q_0^1 & q_0^2 \\ -q_0^2 & q_0^1 & q_0^4 & q_0^3 \\ -q_0^1 & -q_0^2 & -q_0^3 & q_0^4 \end{bmatrix}^T \begin{bmatrix} q_f^1 \\ q_f^2 \\ q_f^3 \\ q_f^4 \end{bmatrix} \tag{5.1}$$

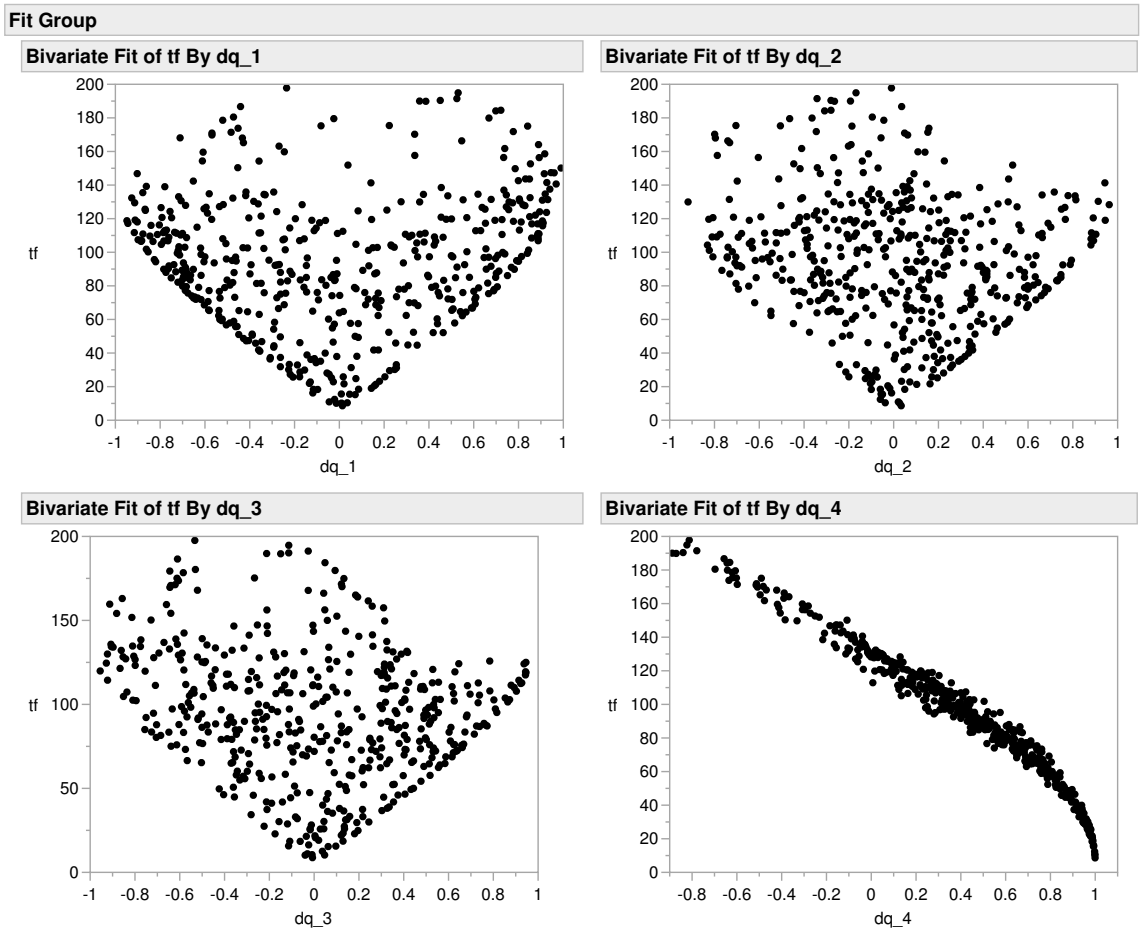


Figure 5.4. Fit of final maneuver time by relative quaternion input data.

An analysis of $\Delta\mathbf{q}$ inputs versus target time was performed by fitting target time (t_f) by the quaternion elements independently, shown in Figure 5.4. These plots illustrate the fit of target maneuver time by relative quaternion elements for the labeled data set. There is a strong correlation between maneuver time and q_4. Meaningful correlations or patterns are not visually evident between maneuver time and the other three quaternion elements, however, the shape of these three distributions is strongly influenced by the $e_i \cdot \sin \psi/2$ operation, which indicates potential masking and may make relationships harder to identify during the model training process. There is also a strong curvature indicated as q_4 nears 1.

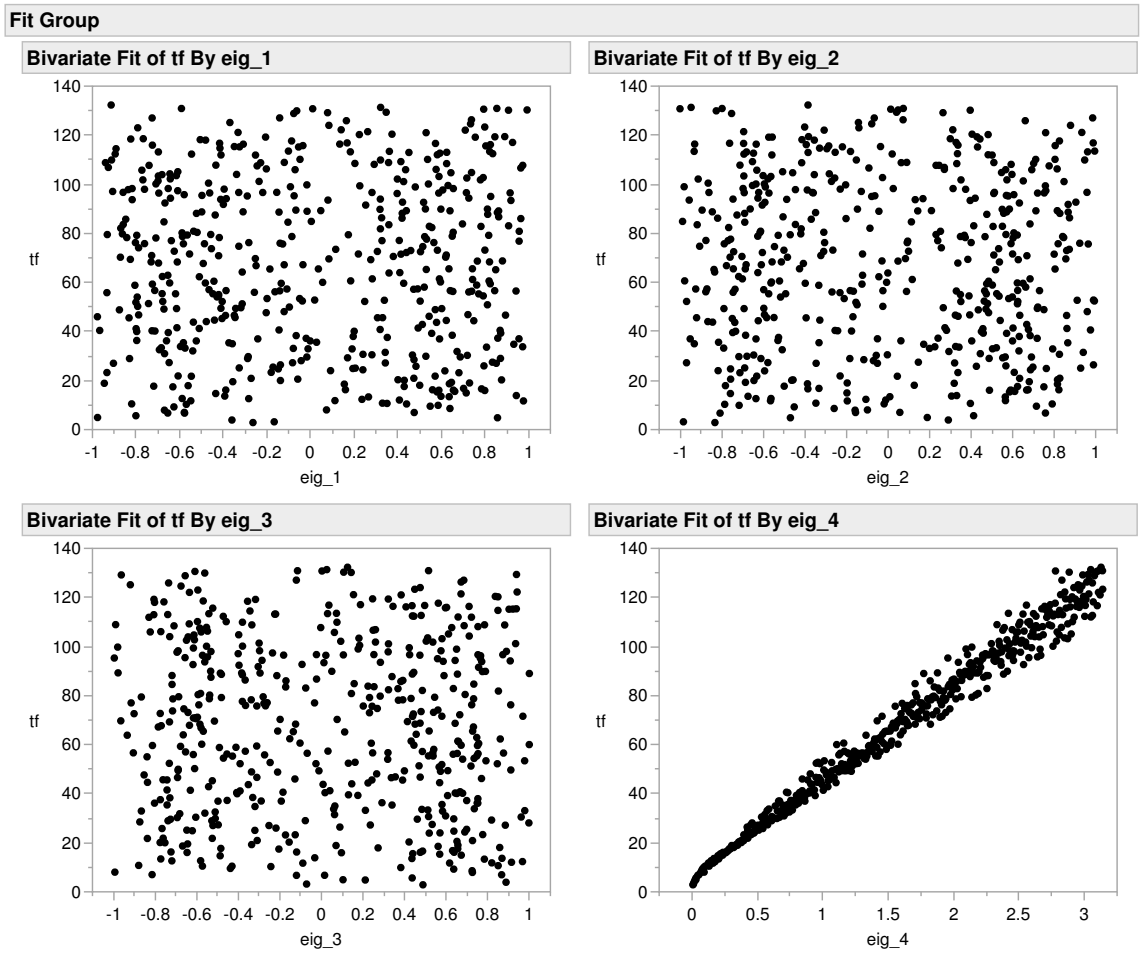


Figure 5.5. Fit of final maneuver time by Euler parameter input data.

Euler Parameters

The fit of the Euler parameter elements against maneuver time (Figure 5.5) shows similar strong correlation to maneuver time in the fourth element (the eigenangle). The other three elements lack the distinctive shaping from the $\Delta\mathbf{q}$ approach and resemble a much more visually random pattern. This implies that the triangularly shaped distributions of $q_{1,2,3}$ against t_f from Figure 5.4 are products of the mathematical mapping that creates quaternions, and not inherent in the input data itself. The curvature at small slew times is also less apparent in the Euler parameter representation than the relative quaternion representation, indicating that it may also be exaggerated by the quaternion transformation.

Building of DoE

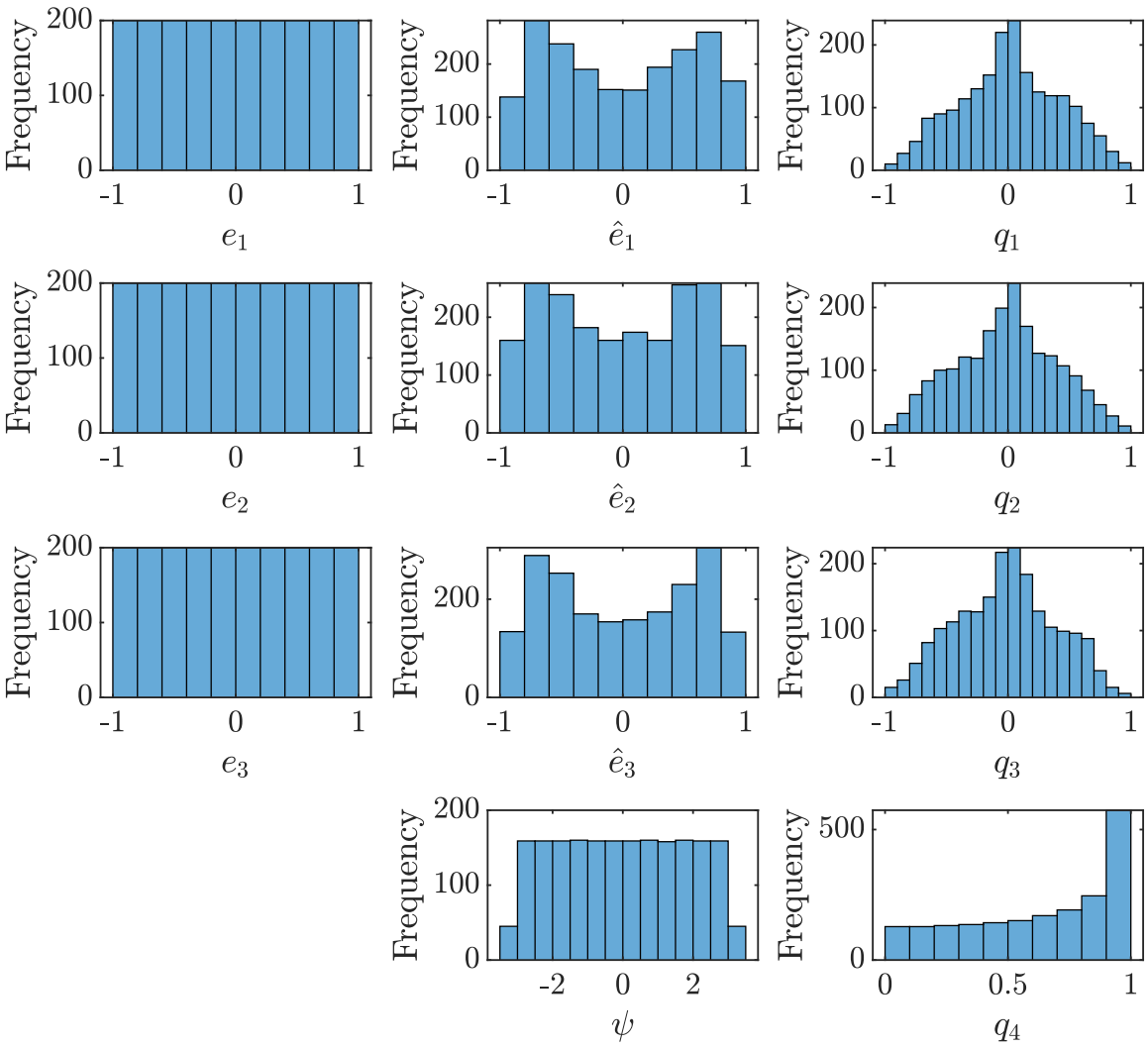


Figure 5.6. Input data set histograms.

DoE using both Euler parameters and the relative quaternion construction were generated from the same underlying data; the relative quaternion form is simply an extra transformation on the Euler parameters. A Latin Hypercube of four variables was created to represent the three elements of an eigenaxis and a fourth element for the eigenangle. This ensures that each set of Euler parameters represents a deliberate input point. Most Latin Hypercube generating programs create their elements on a normalized or uniform distribution of either $[0, 1]$ or $[-1, 1]$, so for this problem the elements of the eigenaxis were generated over uniformly distributed subdivisions from $[-1, 1]$, then normalized to a unit vector, and the

eigenangle was mapped to a domain of $[-\pi, \pi]$. For transformation into a relative quaternion vector, the quaternion definition from Equation (2.22) is applied. Distributions of the DoE elements are provided in Figure 5.6.

The DoE distributions show the effects of starting with a uniformly distributed DoE, normalizing the three eigenaxis elements to a unit vector, and then applying the quaternion definition. In Figure 5.6, smaller slews appear to be favored, indicated by the concentration of $q_{1,2,3}$ elements around zero and q_4 elements around one, however, this is somewhat of a product of applying trigonometric functions of half the eigenangle, per the quaternion definition. The $\cos \psi/2$ term yields more inputs closer to one over the domain $\psi \in [-\pi, \pi]$, while the $e_i \cdot \sin \psi/2$ terms do the opposite, yielding more terms close to zero.

To ensure that the DoE contained adequate representation over the whole input space, random subsets of starting and stopping quaternions were selected to form rotations in the orbital frame, and then transformed into relative quaternion maneuvers in the body-frame. This operation is largely redundant; however, it was done to validate the effectiveness of reducing orbital inputs into relative quaternion inputs as a simplification technique. Also, the engineering model discussed in Chapter 4 requires an 8×1 input vector consisting of a starting and stopping quaternion, so it is convenient to have a rotation expressed in all three manners. The distribution of relative quaternion elements is depicted in Figure 5.7; this operation had the effect of applying a more negative kurtosis to the $q_{1,2,3}$ elements and reducing the severity of the left skew in the q_4 element.

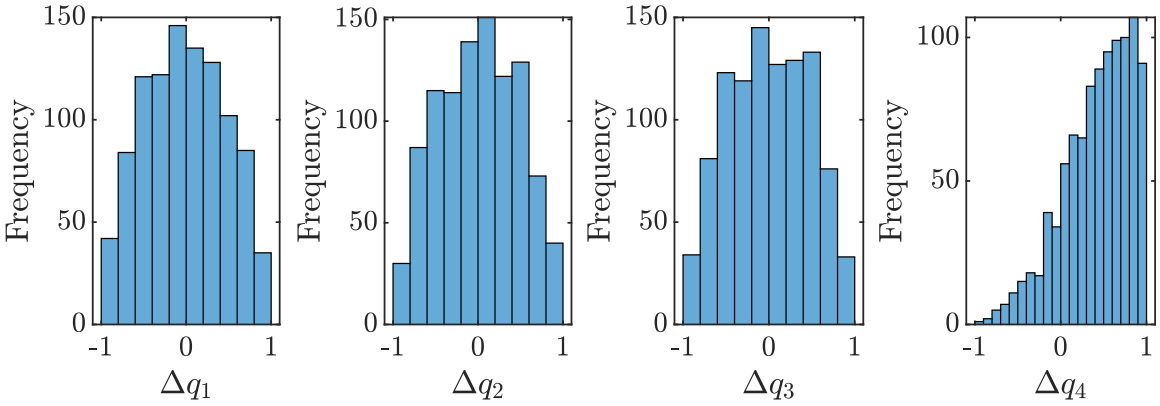


Figure 5.7. Relative quaternion DoE histograms.

Both of these methods were used to train candidate neural networks, where their comparative performance was evaluated. All input candidates had strong correlation with output times, however the Euler parameters had slightly better performance. As the basis of a quaternion rotation, but without the mathematical operations required to generate a quaternion, this result is intuitive, as it is the most direct expression of the physical rotation desired. Accordingly, Euler parameters were selected as the input form for this model. Specific configuration performances are addressed in more detail in the next section.

5.3 Minimum Time Rotation Surrogate Models

With the DoE available in both relative quaternion and Euler parameter form, a neural network was the first surrogate model built and trained.

5.3.1 Neural Networks

Broadly, neural networks are nonlinear models that rely on many linear modeling processes as well as some non-linear processes. "A neural network is a two-stage regression or classification model, typically represented by a network diagram.... This network applies both to regression [and] classification. For regression, typically...there is only one output unit y_1" [19]. For this thesis, the neural network is a regression model that has a single output. Figure 5.8 is an example of a small neural network of this type.

Feed-Forward Neural Networks

A common type of neural network is a *feed-forward network*, in which layers are positioned in sequence and inputs flow from the input layer sequentially through the output layer. Figure 5.8 is a fully-connected feed-forward network in that information flows in one direction through the network, and also each node in a layer is connected to each node in the adjacent layers.

In this example, the input layer consists of two nodes, which is analogous to two input variables. There are two hidden layers, each consisting of two nodes or *neurons*. Finally, the output layer is a single neuron representing one output, which is a typical configuration for a regression, according to Hastie [19].

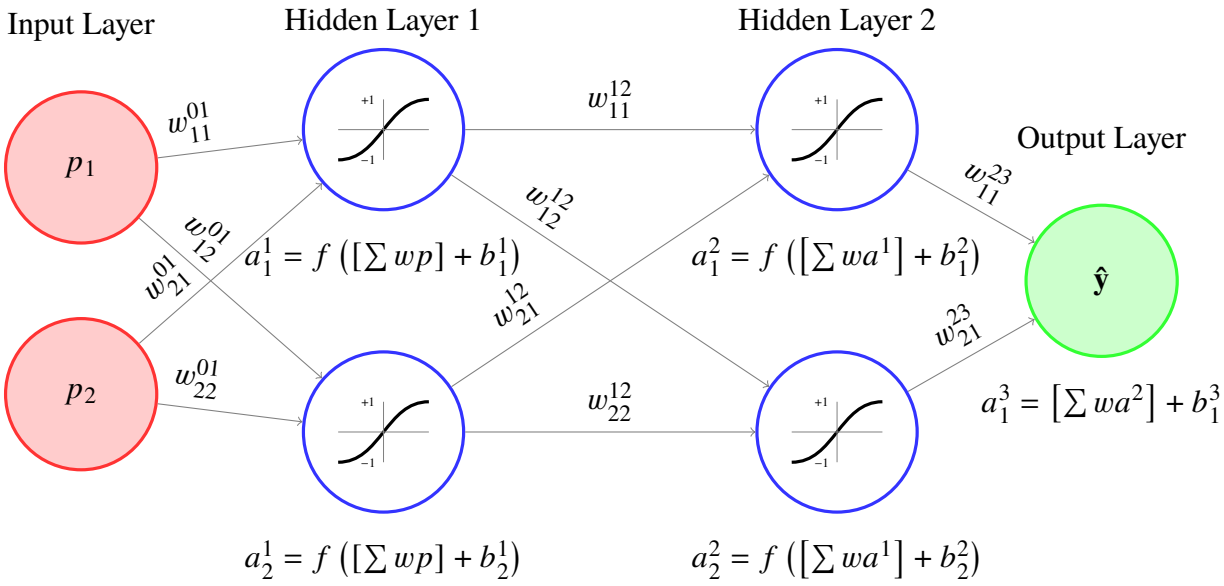


Figure 5.8. Example neural network with two hidden layers.

The input layer performs no operations on the network inputs, it merely passes them along to the next layer. Weights are assigned to all node edges. All neurons except the input layer receive all weighted inputs, sum them, and add a bias term for that neuron. The resulting scalar value is simply a linear combination of its inputs and is referred to as the *logit*. For example, the logit for the first node in the first hidden layer in the example network from Figure 5.8 would be $z_1^1 = w_{11}^{01} p_1 + w_{21}^{01} p_2 + b_1^1$. After calculating the logit, the non-linearity of the neuron is introduced by an *activation function*.

For each neuron, the non-linear activation function is applied to the logit, z, in the form $a = f(z)$ where a is the output of the neuron. Many different activation functions can be used; among those, sigmoid functions are common. In this model, the hyperbolic tangent sigmoid function is the activation function applied to all hidden and output layer neurons. Figure 5.9 plots this function, showing its domain of all real numbers, which maps to an output range between one and negative one, $a \in (-1, 1)$. It is differentiable and returns values close to zero when the input argument is close to zero; both of those traits aid in training network. The formula for a hyperbolic tangent sigmoid function is given in Equation (5.2).

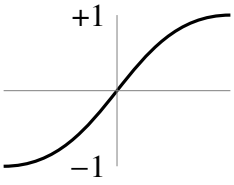


Figure 5.9. Hyperbolic tangent sigmoid activation function.

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{5.2}$$

The same process occurs in all neurons downstream of the input nodes until the output predictions are produced by the output layer. These predictions are typically referred to by the variable $\hat{y}$.

Figure 5.8 labels the relevant mathematical operations with their positions in the network, highlighting the relationships between the layers and the forward flow of information through the network used to make predictions. The scripting convention for network parameters is w_{jm}^{kn}, where k is the previous layer, n is the next layer, j is the previous neuron, and m is the next neuron. Similarly for activation functions and biases, a_m^n represents the mth neuron in layer n. All mathematical operations are linear with the exception of the activation function, Equation (5.2); superscripts represent layers and not exponential operations [18].

Neural Network Hyperparameters In the previous section, several features of neural networks were described, some aspects of which are adjusted by the model itself and some of which are set in advance by the designer. The aspects of the network that are set in advance are called *hyperparameters*. Hyperparameters are typically aspects of the model's structure. The number of hidden layers, input nodes, output nodes, and nodes per hidden layer are all hyperparameters, as is the activation function applied in neurons. The remaining parameters are adjusted by the model during the training process. Examples of parameters adjusted by the model are weights and biases.

Training Neural Networks After a supervised learning model is constructed, it must be *trained* to the labeled data. During this process the model makes adjustments to weight and bias parameters.

Neural networks are trained by randomly initializing model weights and biases, and then propagating training inputs through the model. The predicted values are evaluated against the target data set and adjustments are made to the weights and biases of each neuron based on the output error through a process called *backpropagation*. "The backpropagation algorithm leverages the chain rule of differential calculus, which computes the error gradients in terms of summations of local-gradient products over the various paths from a node to the output" [20]. Updates to the weights and biases are then computed based on these error gradients. The Levenberg-Marquardt backpropagation technique was used in this thesis; it is the recommended training method in MATLAB's Deep Learning Toolbox for supervised learning problems. Together, one forward and backward sweep through the network is an *epoch*. Training continues through multiple epochs until a stopping criterion is reached. Additional details and the mathematical derivations for backpropagation are available from Hastie [19] or Aggarwal [20].

RSIP Neural Network

When designing the neural network for the Remote Sensing Iterative Planner, the hyperparameters that have the greatest effect on model performance were found to be the number of hidden layers and the number of nodes in each hidden layer. "Generally speaking it is better to have too many hidden units than too few. With too few hidden units, the model might not have enough flexibility to capture the nonlinearities in the data; with too many hidden units, the extra weights can be shrunk toward zero" [19].

More than four nodes in the hidden layers are desired to ensure that features from each of the four element input vector can be captured. Several different layer and node combinations are presented in tables 5.1 and 5.2 for performance comparison. For a relatively simple problem, a shallow network should suffice. Accordingly, possible neural network configurations investigated here consisted of one or two hidden layers.

As the initial weights and biases are initiated with some degree of randomness, training the same model with different initializations or random number seeds typically results in a slightly different final model. In some cases, training multiple models and averaging their predictions yields a better result than relying on an individual model. This approach was explored here as well. Multiple configurations were investigated for the DoE in two forms: relative quaternion elements and Euler parameters.

Relative Quaternion Input Evaluation The relative quaternion results are provided in Table 5.1, which reports the predicted final time (t_f) mean absolute error (MAE) and standard deviation (σ) for different hidden layer configurations. This table presents results for relative quaternion input ($\Delta\mathbf{q}$) performance of the average of five network outputs and the expected performance of an individual network. Here, the performance achieved by averaging the outputs of five individually trained neural networks exceeds the performance the five networks evaluated independently. This chart implies that for a given model, better performance should be expected from the average of five different predictions when compared to the prediction of any one of those networks alone. This is the result of averaging the effects of random error from the individual neural networks that is the product of their random initializations during the training process.

A representative training performance plot is presented in Figure 5.10. MATLAB was configured to discontinue neural network training after ten consecutive increases in validation error, and then selected the epoch with the least validation error as the final model. For this network, training was stopped at the 138th epoch, and the 128th epoch had the best validation performance at 1.7804 s^2 mean squared error. The validation set was 15% of the labeled data while 85% of the data was used for training. An independent $n = 100$ test set was used evaluate performance with new data, depicted in Figure 5.11.

Searching for the overall best performer using a relative quaternion input, the networks that appear to perform best both contain two hidden layers, one of 13 nodes each and one of 20 nodes each. These two networks have similar performance, with each excelling in either mean absolute error or standard deviation.

Euler Parameter Input Evaluation The same analysis was conducted with Euler parameter input vectors. These results are presented in Table 5.2. Here, we see the same conclusion that averaging the prediction of five neural networks leads to a better overall result than should be expected from any individual network. Furthermore, the predictions using Euler parameter inputs tend to outperform relative quaternion inputs in both mean absolute error and standard deviation. This is an intuitive conclusion, given that both inputs provide the same information, however relative quaternion elements are masked behind an extra mathematical transformation which could confound variation due to the eigenaxis and variation due to the eigenangle.

Table 5.1. Neural network predicted time error with relative quaternion inputs.

Hidden Layer Configuration	Using 5 Networks		Average of Individual Networks	
	MAE (s)	σ (s)	MAE (s)	σ (s)
[50]	0.888	1.558	1.414	2.482
[13]	1.036	1.616	1.310	1.982
[120]	1.170	1.748	2.067	3.722
[13, 13]	0.892	1.467	1.415	2.453
[50, 50]	1.242	1.889	2.336	4.117
[50, 13]	1.072	1.727	1.562	2.736
[120, 50]	1.503	2.753	3.127	6.123
[120, 13]	1.104	1.865	1.961	3.346
[7, 7]	1.157	1.837	1.742	2.670
[20, 20]	0.817	1.536	1.373	2.803
[13, 7]	0.917	1.489	1.365	2.233
[20, 7]	1.007	1.624	1.482	2.326
[20, 13]	1.037	1.909	1.443	2.600

Notably, for both sets of inputs, the two best performing models both consisted of two hidden layers comprised of 13 or 20 nodes. In general, Euler parameter inputs tended to outperform relative quaternion inputs. Due to its performance and the reduced computational burden associated with 13-node hidden layers over 20-node hidden layers, the final neural network architecture comprises two hidden layers of 13 nodes each with Euler parameter inputs. The structure of the final neural network is graphically depicted in Figure 5.12.

Using the final neural network architecture, a 100 sample point test set was used to evaluate model performance with a very high coefficient of determination, adjusted for sample size (adjusted $R^2 = 0.998$, Figure 5.11).

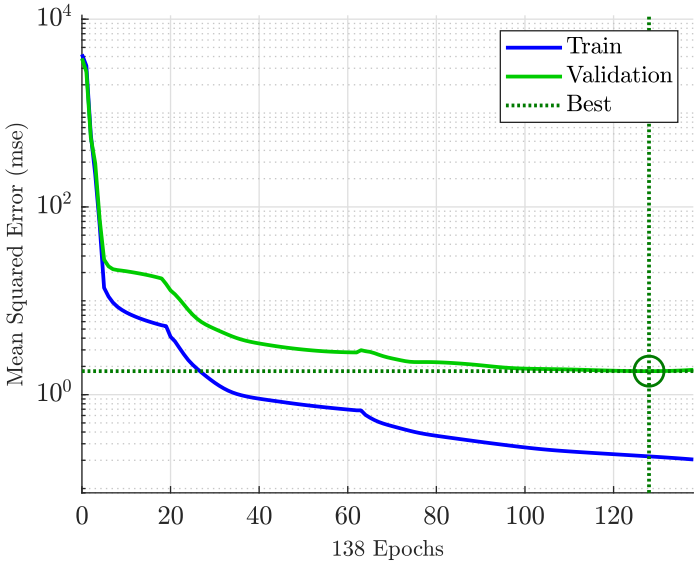


Figure 5.10. Neural network validation set performance.

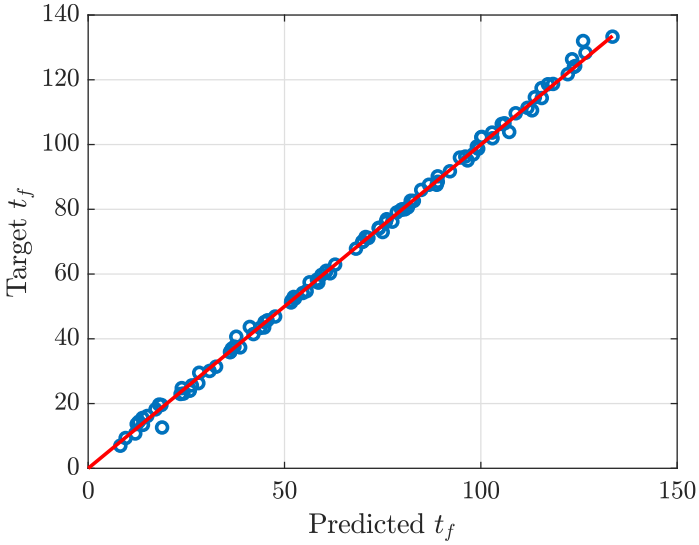


Figure 5.11. Neural network averaged predictions versus target outputs for test data set, $R^2 = 0.998$.

Table 5.2. Neural network predicted time error with Euler parameter inputs.

Hidden Layer Configuration	Using 5 Networks		Average of Individual Networks	
	MAE (s)	σ (s)	MAE (s)	σ (s)
[50]	0.800	1.454	1.126	1.938
[13]	0.782	1.316	1.187	1.814
[120]	1.050	1.580	1.840	2.824
[13, 13]	0.716	1.270	1.022	1.845
[50, 50]	1.270	2.309	2.295	4.272
[50, 13]	0.872	1.623	1.535	2.642
[7, 7]	1.079	1.628	1.606	2.364
[20, 20]	0.756	1.184	1.182	2.057
[13, 7]	0.785	1.353	1.157	1.996
[20, 7]	0.855	1.345	1.225	2.084
[20, 13]	0.769	1.296	1.114	1.948

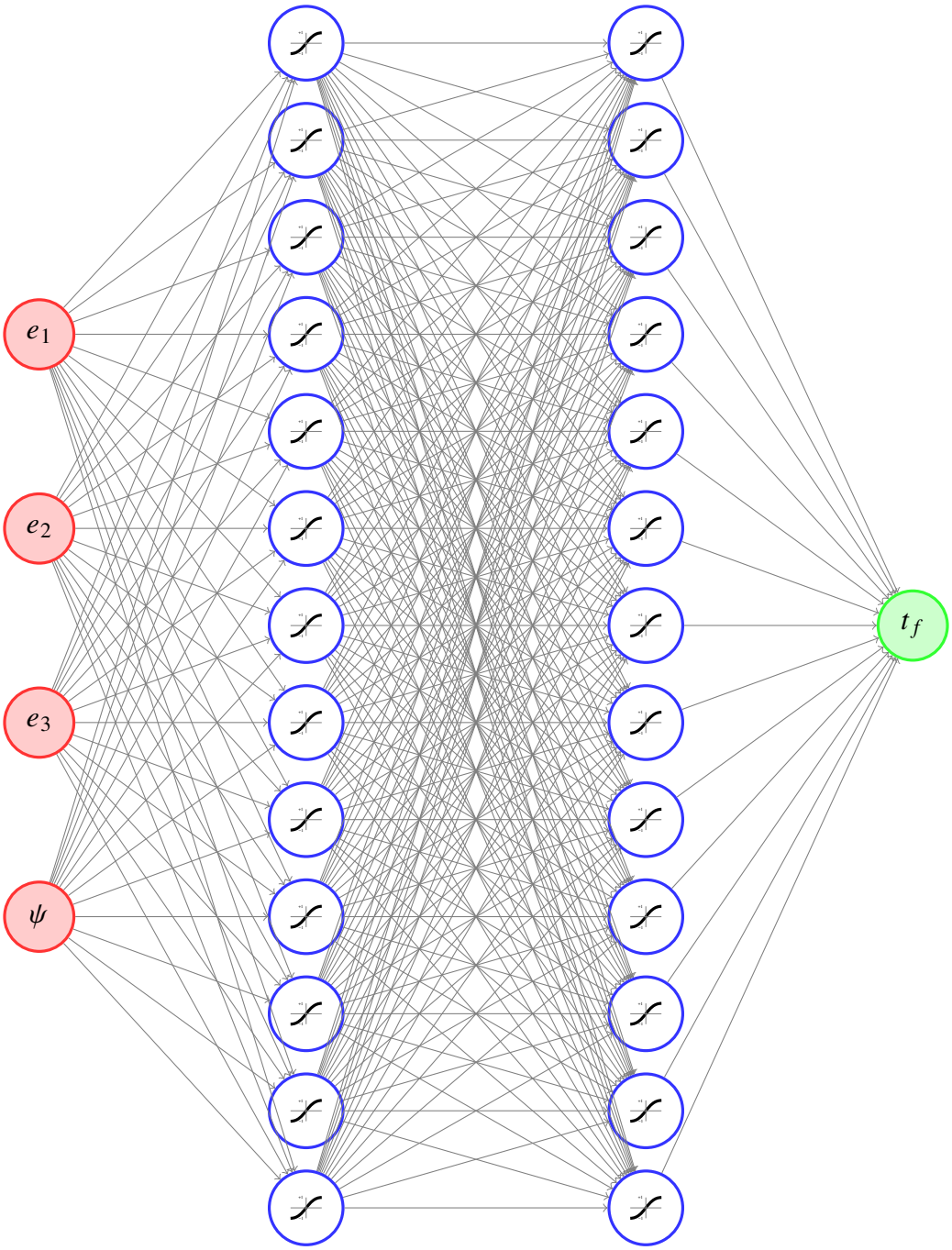


Figure 5.12. Feed-forward neural network with two, 13-node hidden layers.

5.3.2 Stepwise Multiple Linear Regression

Neural networks are an exciting and robust approach to non-linear approximations, and while they are significantly less computationally demanding than the optimal maneuver solver addressed in Chapter 4, they still require a significant number of mathematical operations to execute, depending on the number of nodes involved. The "linear model has distinct advantages in terms of inference and, on real-world problems, is often surprisingly competitive in relation to non-linear methods" [21]. In this case, there are multiple inputs that map to a single output, which is well suited to the linear regression approach. For those reasons, a linear regression technique was built using the same DoE as the neural network to compare the relative advantage of each model.

One potential advantage of linear regression models over neural networks is *model interpretability*. "It is often the case that some or many of the variables used in a multiple regression model are in fact not associated with the response. Including such *irrelevant* variables leads to unnecessary complexity in the resulting model" [21]. Analyzing a regression model after fitting for significant variables by their p-value and coefficient can lead to insights in system performance. A "p-value is the probability that, if the null hypothesis is true, the results from another randomly selected sample will be as extreme or more extreme as the results obtained from the given sample" [22]. Thus for low p-values, typically $p < 0.05$, an analyst can generally assume that the observed effect is statistically significant and not due to random chance. This kind of peak-under-the-hood is not available with neural networks, due to the complex nature of propagating neural network inputs.

The specific technique chosen for RSIP is *stepwise multiple linear regression* or *stepwise regression*, which is a variable selection technique in that it uses a statistical method to determine which terms in a linear regression are significant to include. "Variable selection is used when the analyst is faced with a series of potential predictors but does not have (or use) the necessary subject matter knowledge to enable her to prespecify the 'important' variables to include in the model" [23]. This allows the regression to select what combination of Euler parameter main effects, interactions, and second-order effects provide significant influence over the system output.

The basic form for a general linear regression is provided in Equation (5.3), where β_i are coefficients for input variables, x_i are the input variables, and ϵ is random error. Our model

is analytical, so while error between predictions and target values will remain, this is a noiseless system. Also, the input terms x_i apply to all combinations of inputs, and not simply the main effects (e_1, e_2, e_3, ψ); an x_i term also accounts for the interaction $e_1 \cdot e_2$ and another for the quadratic term ψ^2, etc.

$$\mathbf{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon \tag{5.3}$$

Note that linear regression refers to the linear relationship between the model coefficients, β_i and the model variables, x_i [21]. A term such as ψ^2 is treated as an independent input variable for the regression and maintains a linear relationship with its coefficient β; it does not imply a non-linear action.

Stepwise Multiple Linear Regression Model Fitting

This model was fitted using MATLAB. In general, stepwise multiple linear regression regression models use forward and backward selection to add or remove terms to the model one at a time. The addition or removal of a term is based on the statistical significance of its effect on the variance of the outcome, based on an F-test.

"Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model. In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model.... [Backward stepwise selection] begins with the full least squares model containing all... predictors, and then iteratively removes the least useful predictor, one-at-a-time" [21].

***F*-test** This model began with all quadratic terms, which includes the constant (β_0), main effect (e.g., p_1), interaction (e.g., $p_1 \cdot p_2$), and second-order effect (e.g., p_1^2) terms. The algorithm then fits the regression to the training data by determining the coefficients (β_i) and terms that minimize prediction error using a least squares method.

The F-statistic is determined by the relationship between the variance of the model before the addition or removal of a term and afterwards [22]:

$$F = \frac{(SS_{\text{total}} - SSE_{\text{between}})(k-1)}{SSE_{\text{between}}(n-k)} \tag{5.4}$$

The training algorithm uses this test to determine if the difference in variance between the model with the term included and with the term removed was significant. The criterion used for this model was the sum of squared errors (SSE). The threshold for adding terms is a p-value of 0.05 or less and 0.10 or greater for removing them. Thus, if the F-test determines that there is a greater than 0.10 probability that the true value for the term's coefficient is zero, the term is removed. This method is a type of linear regression and therefore requires linear regression assumptions: errors are normally distributed, the variance of errors is constant, and the errors are uncorrelated [22]. As the training and test data for this model come from analytical sources and not nature, the impacts of noise are negligible, though prediction error will not be eliminated.

If several terms are available to be added or removed from the model, MATLAB will only act on one term at a time, choosing the term with the lowest p-value. A higher order or interaction term will not be added unless both of the main effects are included as well. Similarly, a main effect term will not be considered for removal if significant higher order or interaction terms are present. These rules ensure adherence to DoE principles including scarcity of effects, hierarchy of factors, and heredity of factors.

The stepwise regression model fitted for this problem using the Euler parameter data set is presented in Table 5.3. The final stepwise linear regression is also presented in Equation (5.5).

$$\hat{\mathbf{y}} = \beta_0 + \beta_1 e_1 + \beta_2 e_2 + \beta_3 \psi + \beta_4 e_1 e_2 + \beta_5 e_2 \psi + \beta_6 e_2^2 + \beta_7 \psi^2 \tag{5.5}$$

The main effects that were found to be statistically significant based on the DoE are e_1, e_2 and ψ. Recall from the DoE analysis that the eigenangle of rotation has a strong impact on maneuver time. Its inclusion with an extremely low p-value and a relatively large coefficient is intuitive. For context, with all other terms held constant, an increase in the eigenangle

Table 5.3. Stepwise multiple linear regression terms.

Term (x_i)	Coefficient (β_i)	p-value
(Intercept)	4.1513	7.28×10^{-14}
e_1	0.40122	0.17733
e_2	−1.6608	0.00466
ψ	40.376	3.82×10^{-285}
$e_1 \cdot e_2$	1.416	0.017756
$e_2 \cdot \psi$	1.8547	2.09×10^{-10}
e_2^2	5.0066	1.22×10^{-17}
ψ^2	−0.52705	0.012203

of 1 rad would lead to a 40.376 second increase in maneuver time. Of course, given the interaction and higher order terms that also include the eigenangle, this thought experiment is not directly realizable, but it is a helpful way to gain intuition about the relative effects of the input variables based on their coefficients.

Recall that the inertia matrix for the spacecraft in question is

$$\mathbf{I} = \begin{bmatrix} 223.66 & 0 & 0 \\ 0 & 264.71 & 0 \\ 0 & 0 & 170.23 \end{bmatrix} \text{kg} \cdot \text{m}^2. \tag{4.8}$$

The inclusion of the e_1 and e_2 terms of the eigenaxis implies that those terms are more significant than the e_3 term, which is not included. This is perhaps the case because they map to the higher moments of inertia along those axes, shown in Equation (4.8). Rotations that require movement around principal axes with higher moments of inertia should be expected to take longer, given a minimum time bang-bang control solution. In this particular model, Equation (5.5) indicates that the time prediction for any rotation about the e_3 axis, for example, will be exclusively a function of ψ from the β_3 and β_7 terms, as well as the intercept, β_0.

Ultimately, the model defined in Table 5.3 and used here is a *Response Surface Model* because its input variables include main effects, interactions, and second-order (or "quadratic")

terms [16]. The terms give curvature and twist to the n-dimensional hyperplane that results from the regression. Here, the model represents a four-dimensional hyperplane after dropping the e_3 variable, leaving three independent input variables plus the prediction output. These characteristics allow the model to capture some non-linear features while still using linear regression to determine model coefficients.

Full Linear Regression Comparison

Stepwise regression can be a valuable tool for models that contain a large number of independent input variables. For example, in some cases it might be undesirable or infeasible to evaluate a complete quadratic model with 100 independent input variables and forward stepwise selection may be an appealing framework to find the interactions and quadratic terms that have a significant effect on the output's variance. However, some of the available literature is skeptical about stepwise regression in principal because the p-values used for selection of terms are based on the estimates of the term coefficients, as well as several assumptions that are explored in more detail in Harrell [23].

"The problems of p-value-based variable selection are exacerbated when the analyst (as she so often does) interprets the final model as if it were pre-specified. Copas and Long stated one of the most serious problems with stepwise modeling eloquently when they said, 'The choice of the variables to be included depends on estimated regression coefficients rather than their true values, and so X_j is more likely to be included if its regression coefficient is over-estimated than if its regression coefficient is underestimated"' [23].

The importance of the final input terms is relatively evident in this stepwise model, as they are directly mapped to a physical system. Longer rotations will take longer, as should rotations that require large reorientations around axes with larger moments of inertia. Only the smallest principal axis moment of inertia component (e_3) was deemed insignificant by the stepwise model.

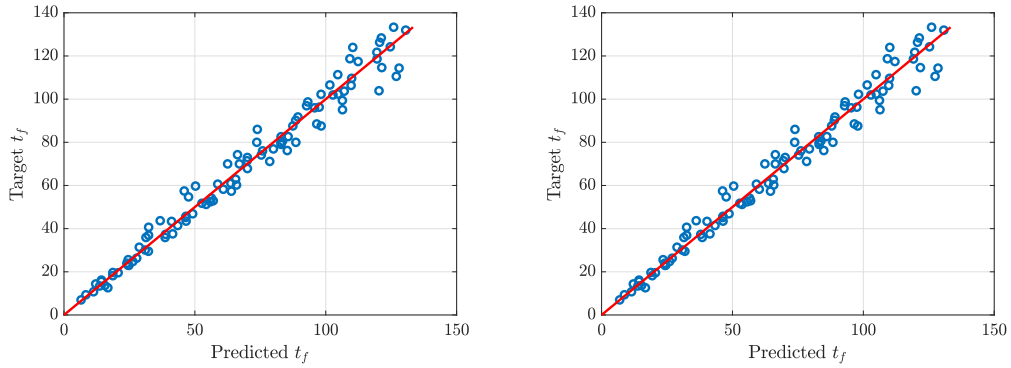
However, Harrell concluded that "[u]nless most predictors are either very significant or clearly unimportant, the full model usually outperforms the reduced model. Full model fits have the advantage of providing meaningful confidence intervals using standard formulas" [23].

Table 5.4. Linear regression terms.

Term (x_i)	Coefficient (β_i)	p-value
(Intercept)	4.0401	2.63×10^{-9}
e_1	0.29854	0.62412
e_2	-1.7053	0.003986
e_3	0.73392	0.2257
ψ	40.367	7.5×10^{-281}
$e_1 \cdot e_2$	1.3959	0.020321
$e_1 \cdot e_3$	0.55279	0.37967
$e_1 \cdot \psi$	0.042562	0.89433
$e_2 \cdot e_3$	-0.87833	0.1482
$e_2 \cdot \psi$	1.9047	1.21×10^{-10}
$e_3 \cdot \psi$	-0.34041	0.29706
e_1^2	0.17883	0.81871
e_2^2	5.1237	7.69×10^{-13}
e_3^2	0	N/A
ψ^2	-0.5163	0.01500

To explore the impact of this concern in this model, a full linear regression with no selection or removal of terms was fitted to compare performance with the stepwise model. This linear regression included all independent input terms as main effects, their interactions, and their quadratic terms. The resulting model is defined in Table 5.4. A graphical comparison of test set performance is presented in Figure 5.13.

The test sets show a model and fit that is very similar between the stepwise and full linear approaches for this particular case. Accordingly, there is no apparent detriment to removing some less significant terms from the regression for optimal maneuver time approximation. It remains for future work to further explore various linear regression implementations to find a desirable balance between computational efficiency and model accuracy.



(a) Stepwise regression, model in Table 5.3. (b) Linear Regression, model in Table 5.4.

Figure 5.13. Test data predicted vs. target outputs.

5.4 Mission Plan Analysis

The advantage of the two supervised models developed in this chapter over legacy techniques is their ability to output maneuver times that are close to the optimal time without the need to explicitly solve the optimal control problem. Chapter 2 discusses the taxicab and eigenaxis legacy maneuvers, including their suboptimality. Chapter 3 posited that the full maneuver profile is not needed to support a path planner; only the time required to execute the maneuver is needed. Chapter 4 provided the engineering model needed to produce time-optimal slewing maneuvers. Now that two supervised learning techniques are available, path planning programs should be able to take advantage of the approximated maneuver times to generate more efficient mission plans.

5.4.1 Minimum Time Approximation Mission Plans

RSIP was augmented to run with approximated optimal slew times using these new techniques; the results of this effort are presented in this section.

Neural Network Mission Model

A sample mission model using a neural network to approximate minimum time maneuver slew times is shown in Figure 5.14. This particular mission was flown at 675 km altitude and collected eight different targets for a total collection value of 20 VU.

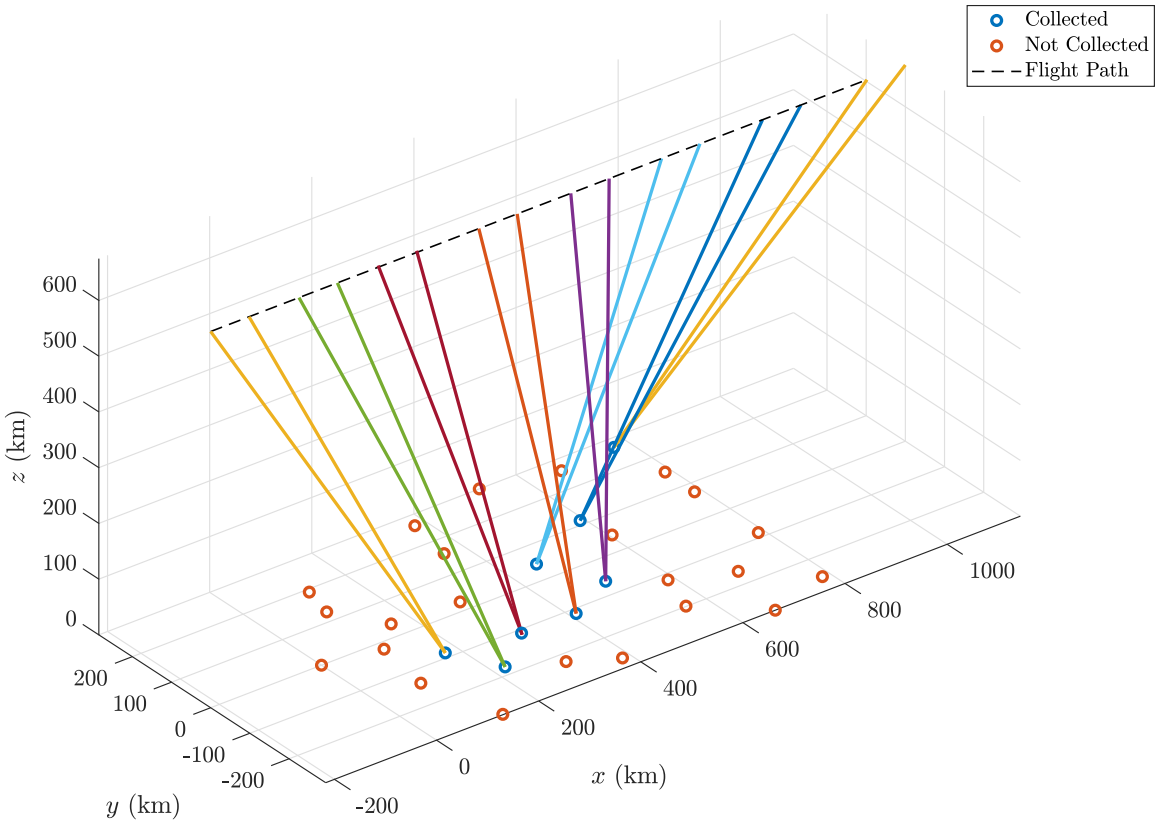


Figure 5.14. Mission plan using neural network optimal slew approximations, value maximizing.

Stepwise Regression Mission Model

A sample mission model using a stepwise regression function to approximate minimum time maneuver slew times is shown in Figure 5.15. This mission was flown at 675 km altitude and collected nine different targets for a total collection value of 19 VU.

5.4.2 Legacy Mission Plan Comparisons

The sample mission plans provided in Figures 5.14 and 5.15 are enticing results, but require further inspection. The feasibility of the mission plans must be investigated prior to being implemented on an actual system, however these plans were created in the exact same fashion as the plans discussed in Chapter 3 that used legacy slewing maneuvers (figures 3.10 and 3.11). Only the code used to generate the maneuver time, t_f, required adjustment. In the case of the legacy models, that code provided a simple analytical

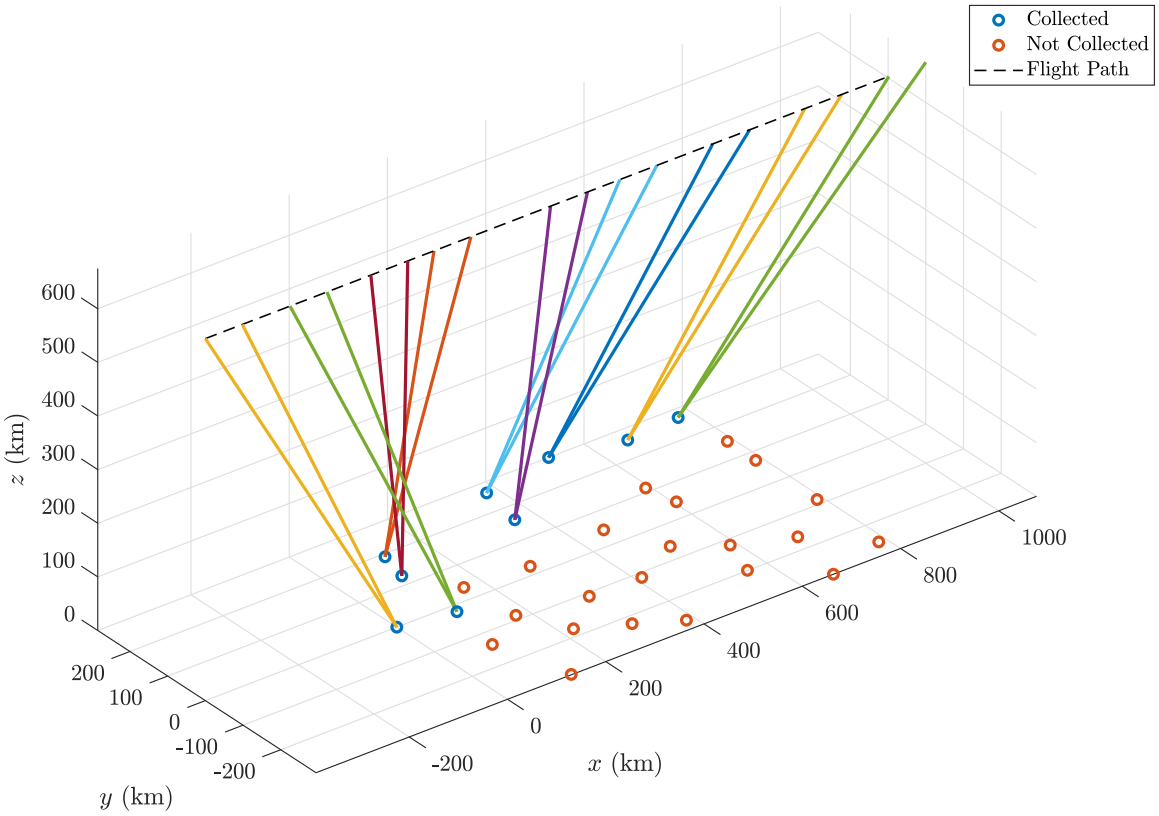


Figure 5.15. Mission plan using stepwise regression optimal slew approximations, value maximizing.

solution. However, the neural network calculates maneuver times through a sequence of algebraic scaling operations, matrix multiplications, closed-form combinations, and algebraic unscaling operations. The stepwise model produces a maneuver time through simple linear operations on input variables. With these distinctions, the path planner operates with the exact same mechanisms.

To facilitate comparisons, nine mission plans were generated for each of the four maneuver time calculation techniques previously discussed, and the actual optimal mission time required to execute each plan was also produced, using the engineering model from Chapter 4. For each method, mission plans were created from three different orbital altitudes: 675 km, 875 km, and 1200 km. For each altitude, three plans were created using different random number seeds for neural network initialization and genome permutations in the genetic algorithm. This led to distinct solutions of similar value and efficiency.

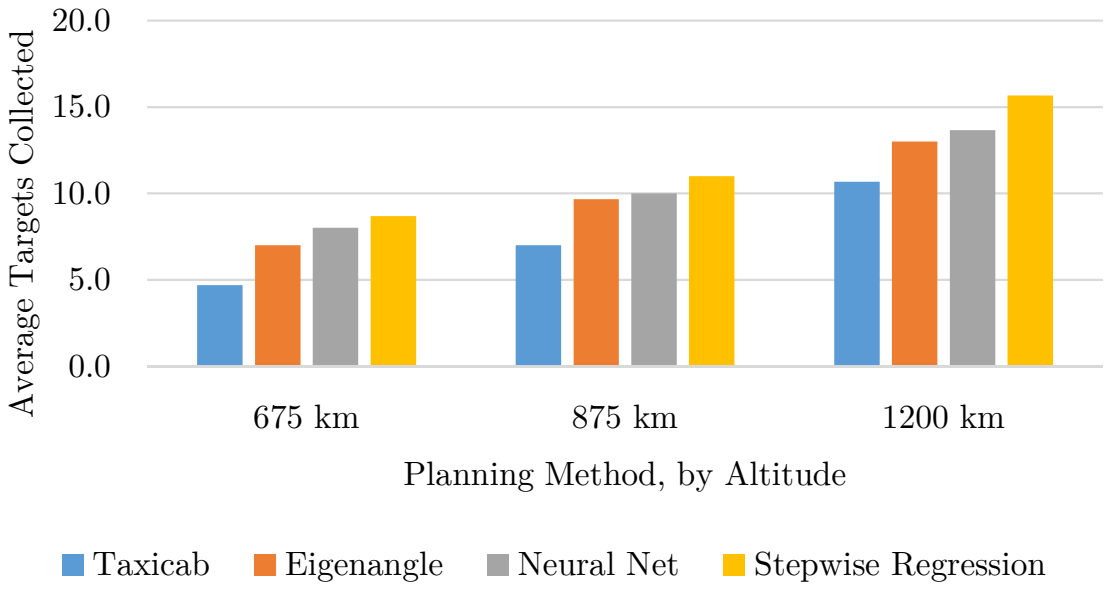


Figure 5.16. Average number of targets collected, by planning method and spacecraft altitude.

Figure 5.16 compares the average number of targets collected, broken down by altitude and maneuver method. Consistently, the optimal approximation techniques outperform legacy techniques in number of targets collected. Figure 5.17 makes the same comparison, but measures the value of the mission plan instead of the number of targets collected. In this form, the relative performance of the methods remains the same. For different orbital altitudes over the same target field, it is intuitive that a higher altitude would enable additional collections due to the slower orbital speed as well as a slightly larger access window thanks to additional slant range for a given look angle constraint, which in this model is 30 degrees.

Feasibility and Improvement

By design, the planning program is a low fidelity mission sequence planner, in that it provides merely a sequence of targets to collect and does not issue a high fidelity solution without the use of the full engineering model. RSIP attempts to achieve that function by integrating the engineering model into the final phase of the planner, where the promulgated mission sequence is input into the high fidelity engineering model to compute actual maneuver times, in lieu of approximations.

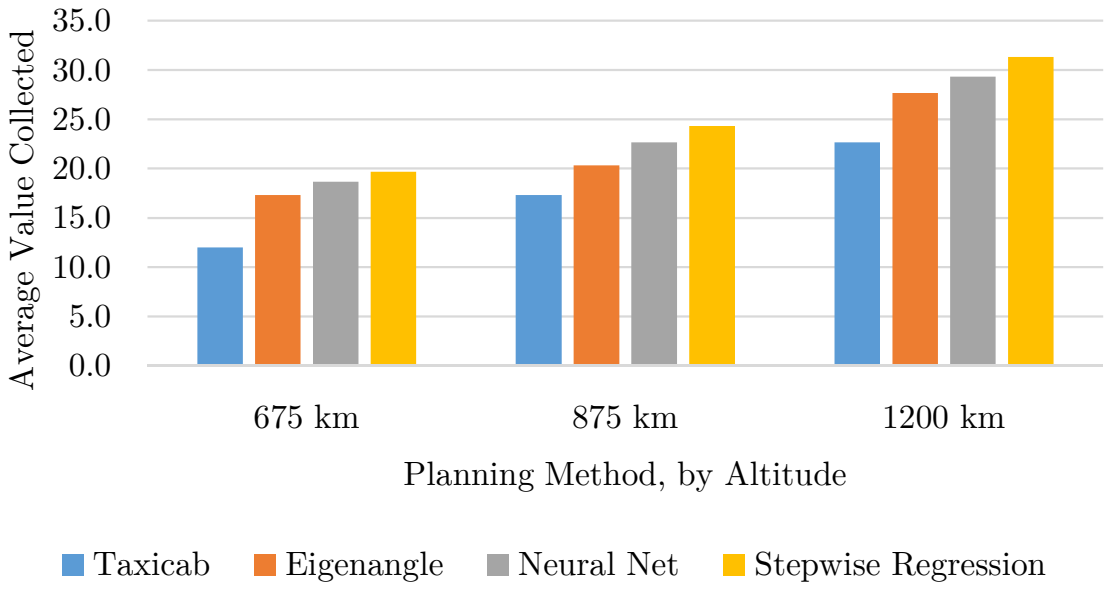


Figure 5.17. Average mission value collected, by planning method and spacecraft altitude.

A successful implementation of the planning principles proposed in this thesis does not require an exact approximation or a precise high fidelity mission plan. Rather, success is defined in two ways:

1. Does this mission planner produce a feasible mission sequence that assumes optimal maneuvers?
2. Does the mission plan allow more value to be obtained than with legacy techniques?

If the answers to both of these questions are "yes," then the model is successful in that it represents an improvement over the status quo of mission planning that can be achieved simply by incorporating proven techniques to increase mission efficiency.

While the individual maneuvers produced by an engineering model like the one presented in Chapter 4 are proven to be mathematically optimal, convergence on an efficient solution for a genetic algorithm-driven path planner like RSIP does not come with the same assurances of optimality. The possibility of suboptimality in the mission planner should not be considered a risk, however. For similar planning effort, more mission accomplishment can be obtained and that advance alone is valuable.

Slew Time Prediction Error

An important metric for the supervised learning methods is their accuracy in approximating slews. Much discussion has been made of accuracy in approximating individual slews, as well as test data and training datasets. A useful macro metric is the overall residual error from a mission plan generated using approximations of optimal slews with that same mission plan executed with the actual optimal slews. Those residual comparisons are presented in Figure 5.18.

The two optimal approximation techniques are considerably closer to the optimal slew, noting the logarithmic scale on the vertical axis. For this model, the stepwise regression technique yields smaller residuals than the neural network, though both are significant improvements over legacy techniques. This chart encompasses data from the same mission plans discussed herein, where the slews are relatively small. It is reasonable to infer that the gains from this method would be more advantageous for a plan that required generally longer slews.

Relative Value of this Approach

The amount of efficiency gained through this technique must be determined for each individual system to which it is applied, as it will be a function of the physical properties of the spacecraft, the planning techniques currently in use, and typical slews for the system. For example, systems with longer slews may see more significant efficiency gains than systems that generally make small slews. For the six mission sequences presented here for direct neural network–stepwise comparison, the median slew is 5.4 degrees and the maximum slew is 24.6 degrees. Whether or not the potential gains are worth the effort of reworking planning systems and incorporating optimal maneuvers will depend on individual satellite operators, however with demonstrated gains of 5% to 21% for individual maneuvers as previously discussed, the potential could be significant and should be explored [2].

The direct comparisons presented in this section make a compelling case for taking advantage of these techniques. Using the the results in Figures 5.16 and 5.17 and the eigenaxis technique as a baseline, tables 5.5 and 5.6 quantify these advantages. For example, using a neural network enabled optimal slew approximation technique for a system at 675 km, operators could expect an increase in targets imaged on the order of 14%. Using a stepwise regression enabled technique for the same mission could yield an increase on the order of

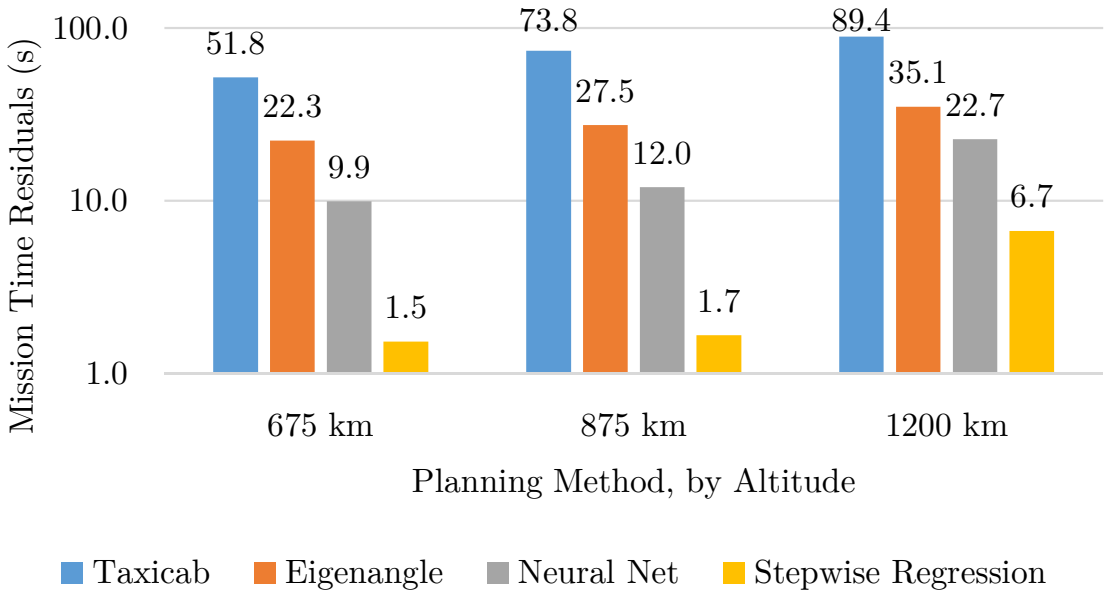


Figure 5.18. Mission time residual error, by planning method.

Table 5.5. Targeting gains resulting from optimal approximation techniques.

	Taxicab	Eigenaxis (baseline)	Neural Network	Stepwise Regression
675 km	4.7 (-33%)	7.0	8.0 (14%)	8.7 (24%)
875 km	7.0 (-28%)	9.7	10.0 (3%)	11.0 (14%)
1200 km	10.7 (-18%)	13.0	13.7 (5%)	15.7 (21%)

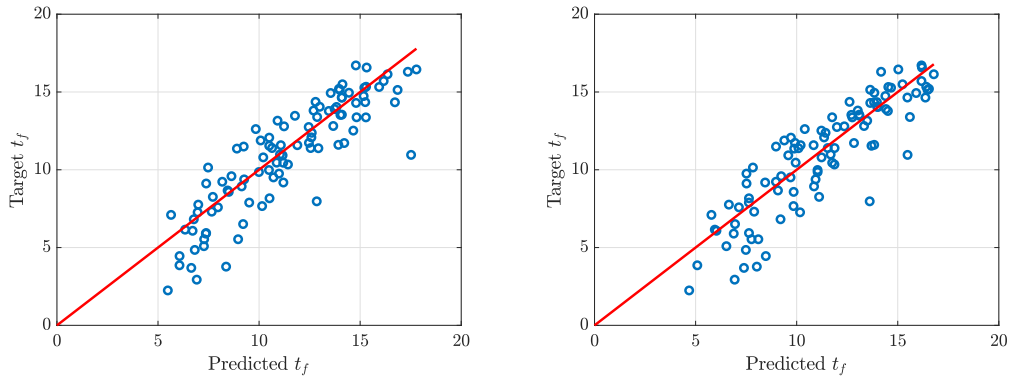
24% for a system similar to this example. For this concept demonstration, double digit efficiency gains are consistently achieved.

5.4.3 Small Angle Performance

Surprisingly, in this model the stepwise regression technique outperforms the neural network. An intuitive assumption may be that the neural network would have more opportunity to capture relationships and non-linearities in the data given the complex interactions associated with feed-forward networks and the number of hidden layer nodes. This expectation is also supported by coefficients of determination for test set performance. The adjusted R^2 for the neural network was 0.998, whereas for stepwise regression it was 0.974.

Table 5.6. Value gains resulting from optimal approximation techniques (VU).

	Taxicab	Eigenaxis (baseline)	Neural Network	Stepwise Regression
675 km	12.0 (-31%)	17.3	18.7 (8%)	19.7 (13%)
875 km	17.3 (-15%)	20.3	22.7 (11%)	24.3 (20%)
1200 km	22.7 (-18%)	27.7	29.3 (6%)	31.3 (13%)



(a) Neural network over small angles, architecture in Figure 5.12.

(b) Stepwise regression over small angles, model in Table 5.3.

Figure 5.19. Test data predicted vs. target outputs over small angles.

A possible explanation for this phenomenon is that while the neural network performs better over the full domain of possible slews ($\psi \in [-\pi, \pi]$), the vast majority of slews required for the missions modeled here are small slews, generally less than 25 degrees. While the neural network should be expected to perform better over the full input space, over small slews the neural network is biased slightly higher than the stepwise model, leading to a more consistent over-estimation of the true optimal slew time. A comparison of the performance of the trained models over small angle slews is presented in Figure 5.19. For this test set with eigenangles from zero to 0.26 rad (15°), bias in both models is visually evident. Table 5.7 provides summary statistics for small angle mission residuals from optimal performance. While both optimal approximation techniques have similar residual variances, the mean residual is considerably higher for the neural network. Finally, Figure 5.20 depicts this phenomenon graphically; while the variance for both techniques is similar, the mean error for the neural network is biased higher.

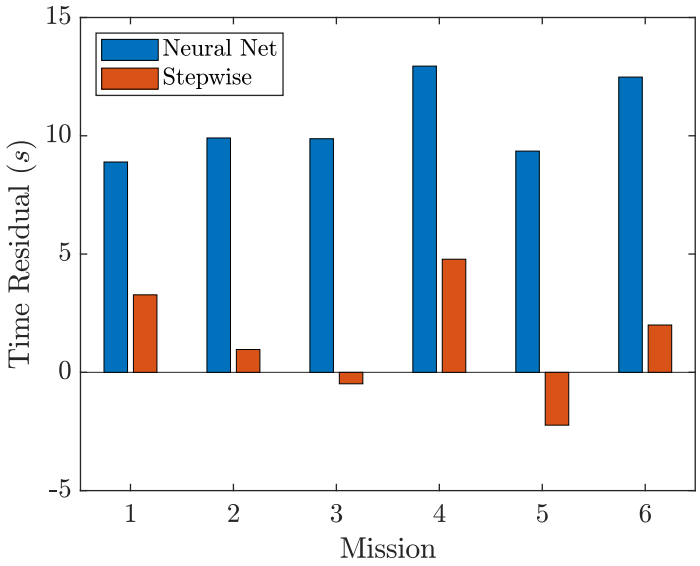


Figure 5.20. Mission residuals from optimal execution, by method.

Table 5.7. Summary statistics for small angle test set.

	Neural Network	Stepwise Regression
Mean (s)	1.44	0.19
Variance (s^2)	3.36	3.21

With a relatively small number of samples, one should not draw the conclusion that one technique is *better* than the other, rather that both yield feasible results, and for these two specific models with this training dataset, the stepwise regression model does tend to perform slightly better.

Subsequent investigations of these techniques and real-world implementation should not discount either method; however, further effort and feature engineering is needed to develop a DoE that achieves best model performance over the types of slews most likely to be required. For RSIP, the model does not achieve its best performance over small slews, which make up nearly all of its mission maneuvers. Subsequent versions of this planner will take that into account and design a more appropriately tailored DoE.

5.4.4 Computational Comparisons

Separate from performance comparisons, the simple nature of regressions may make them an appealing option when additional computational efficiency is desired. While both a neural network and a regression are significantly simpler algorithms than the full engineering model, stepwise regression will typically be simpler still than a neural network. In this case, the number of floating-point operations (FLOPs) to compute a maneuver time is orders of magnitude different for the different methods. A *FLOP* is a single arithmetic operation of one value on another. In computer memory, these values are stored as *floating-point numbers*, which is simply a protocol for storing non-binary numbers in a computer's memory. Determining the number of floating-point operations required to complete a set of code can be a useful proxy for the complexity of the code and the computing resources it requires. Table 5.8 compares the maneuver time methods used in RSIP to compute a single maneuver time.

Taxicab and eigenaxis methods are both very efficient computationally (and of course inefficient physically). The taxicab computation is simply two eigenaxis maneuvers around different principal axes, followed by their summation. The stepwise regression prediction is also relatively efficient. With seven terms and four interactions, the stepwise method requires 17 FLOPs to compute a maneuver time. For the neural network, two hidden layers of 13 nodes is a far more complex prediction scheme. Furthermore, to lower the prediction variance to the desired level, the average prediction of five networks is used in RSIP, which requires five times the floating-point operations as a single neural network.

Table 5.8. Floating-point operations required for maneuver time methods.

Method	FLOPs
Taxicab	7
Eigenaxis	3
Stepwise Regression	17
Neural Network	756
5 Neural Networks	3785

One conclusion that should be considered, is that even if a neural network is outperforming a regression technique, it may still be advantageous to consider a regression if its performance is adequate and the decreased computational requirements are significant. In this particular case, the the regression performs slightly better and runs considerably faster, but it is likely that many applications will have better performance with a neural network. For the current version of RSIP, stepwise regression for calculating maneuver times is clearly the most desirable approach.

5.5 Summary

The goal of this thesis is to explore the possibility of integrating time-optimal maneuvering as part of a mission planning process, which would enable widespread adoption of these techniques in space enterprises. This chapter addresses the final discipline required to achieve such a mission planning process using supervised machine learning.

Design of Experiments provided a useful sampling of the the input space for this problem, which was developed into labeled data using the optimal solver discussed in Chapter 4. With this labeled data, multiple formats for the input data were evaluated, as well as multiple approaches for supervised learning models. For this system, a neural network model and a stepwise multiple linear regression model were selected based on their performance, both using Euler parameter inputs.

When augmenting the mission planner with these metamodels, both minimum time approximation techniques performed significantly better than legacy techniques. This outcome reinforces the conclusion that with simple planning changes, space system operators can realize double-digit increases in mission efficiency using techniques already demonstrated on-orbit.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusion

This thesis addresses the absence of time-optimal attitude maneuvers in current space operations. Time-optimal rotations are significantly faster than legacy slewing techniques that are commonly used today. These minimum time maneuvers have been demonstrated on-orbit, first in 2010; the lack of wide-spread adoption of this maneuvering technique is remarkable. The inability to incorporate time-optimal slews into existing mission planning frameworks is the likely source of this arrested development, and this thesis offers an approach for minimum time maneuver implementation that leverages supervised learning.

Two viable supervised learning approaches that approximate the minimum maneuver time were developed: a neural network and a stepwise multiple linear regression. Both of these models were trained on a dataset of time-optimal slews, generated from the optimal control problem presented in Chapter 4. Using these metamodels in existing planners can improve mission efficiency without the need to solve optimal control (or hybrid optimal control) problems as part of the planning process. Both of these approaches yield feasible mission sequences that are significant improvements over the legacy mission planning approach. This thesis thus demonstrates that existing mission planners can be augmented to account for time-optimal spacecraft maneuvers by using supervised machine learning to approximate optimal maneuver times. To incorporate these ideas into a mission planner, all that is needed is to update the model used for slew time prediction.

6.1 Implications for Current and Future Missions

The potential added value from an approach such as this is significant and should be considered by any spacecraft operator with an over-subscription problem for mission planning. Any operator with more targets to collect than time to collect them could directly and immediately benefit from the incorporation of time-optimal maneuvers into their satellite slews, enabled by a supervised learning augmented planning process. Today, double-digit percentage increases in imaging target collection is an attainable goal with the implementation of the mission planning approach presented herein.

6.2 Future Work

The feasibility of the supervised learning approach has been demonstrated, however future work remains to iterate upon this approach and refine its efficacy. Some potential areas for further investigation follow.

Feature Engineering The relative superior performance of linear regression over neural networks in this example system is a reminder of the efficiency and utility of regressions; however, it also indicates that neural network performance could very likely be improved. Feature engineering should be undertaken to more precisely identify the drivers of neural network performance. Future investigation should identify whether training separate supervised learning models for acceleration-limited slews and rate-limited slews reduces overall variance in the minimum time prediction.

Constrained Mission Plans Real spacecraft frequently have maneuver constraints based on hardware limitations or bright bodies that may damage sensitive sensors inside certain look angles. The incorporation of look-angle constraints and keep-out zones has been demonstrated for optimal control problems previously [14], [24]. However, methods to introduce the constraints into the combinatorial mission planning problem remain to be demonstrated.

APPENDIX A:
Simulation Target Locations

Simulation targets used for the RSIP mission models are presented here.

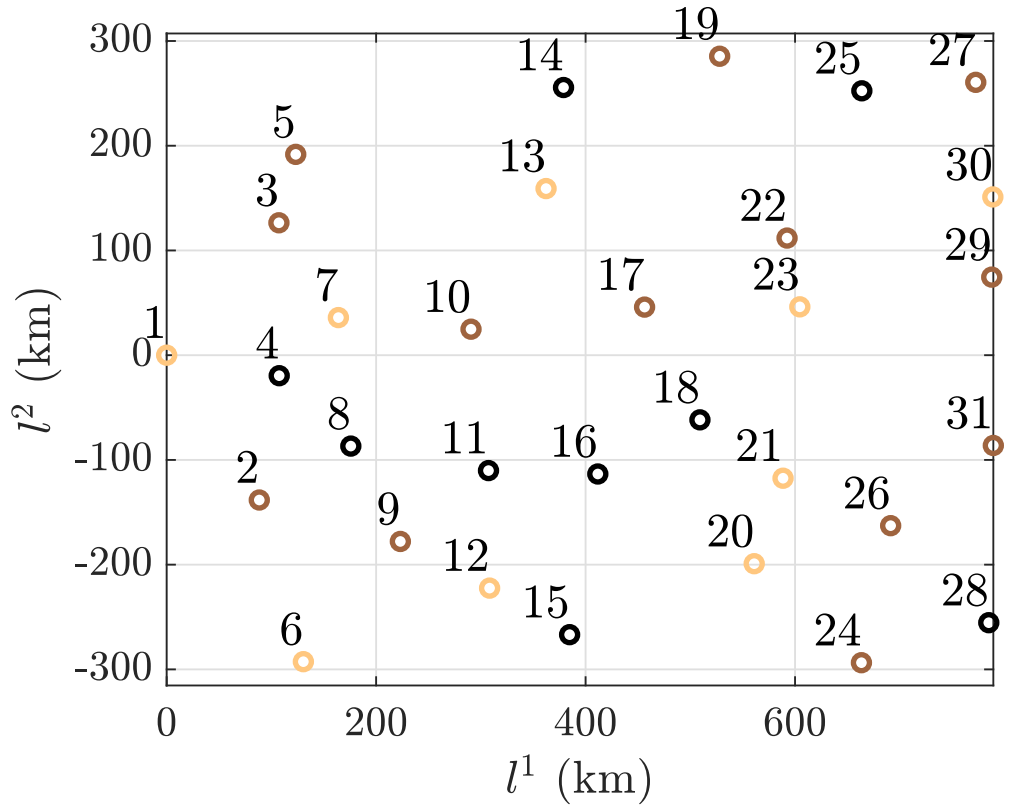


Figure A.1. Simulation targets.

Table A.1. Simulation targets.

Target	ℓ^1 (km)	ℓ^2 (km)	Value
1	0	0	1
2	88.44	-138.273	2
3	107.185	126.426	2
4	107.465	-19.578	3
5	123.24	191.754	2
6	130.535	-292.635	1
7	163.97	35.901	1
8	175.835	-86.796	3
9	223.145	-177.849	2
10	290.595	24.819	2
11	307.3	-110.184	3
12	308.345	-222.396	1
13	362.25	159.099	1
14	378.94	255.582	3
15	384.745	-266.841	3
16	411.665	-113.43	3
17	456.33	45.921	2
18	509.18	-61.731	3
19	528.025	285.438	2
20	561	-199.086	1
21	588.535	-117.477	1
22	592.365	111.837	2
23	604.555	46.176	1
24	663.385	-293.616	2
25	663.775	252.444	3
26	691.29	-162.771	2
27	772.55	260.529	2
28	784.98	-255.42	3
29	787.81	74.547	2
30	788.855	151.11	1
31	789.325	-86.208	2

APPENDIX B:
Optimal Control Nomenclature

The following nomenclature is used to develop the minimum time optimal control model used for this project. Ross' primer on optimal control theory is a useful guide for further information on this subject [13].

Trajectory Optimization Nomenclature

$\mathbf{x}$	=	State Vector
$\mathbf{u}$	=	Control Vector
$F\left(\mathbf{x}(t), \mathbf{u}(t)\right)$	=	Running Cost Function
$E(\mathbf{x}_f, t_f)$	=	Endpoint Cost Function
$\mathbf{e}(x_f, t_f)$	=	Endpoint Condition Vector
$\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t))$	=	Constraint Function
$\boldsymbol{\lambda}$	=	State Covector
$\boldsymbol{\nu}$	=	Endpoint Condition Covector
$\boldsymbol{\mu}$	=	Constraint Covector
$H(x, \lambda, u, t)$	=	Hamiltonian Functional
$\bar{H}(x, \lambda, \mu, u, t)$	=	Hamiltonian Lagrangian
$\bar{E}(\mathbf{x}_f, \boldsymbol{\nu}, t_f)$	=	Endpoint Lagrangian
$\mathbf{q}$	=	Quaternion Attitude Vector
ω	=	Spacecraft rotational velocity [$^{\text{rad}}/_{\text{s}}$]
α	=	Spacecraft rotational acceleration [$^{\text{rad}}/_{\text{s}^2}$]
$\mathbf{I}$	=	Inertia matrix of the spacecraft [$\text{kg} \cdot \text{m}^2$]
$\mathbf{Z}$	=	Reaction wheel configuration matrix
τ	=	torque [$\text{N} \cdot \text{m}$]
h	=	Reaction wheel angular momentum [$\text{N} \cdot \text{m} \cdot \text{s}$]
Ω_w	=	Reaction wheel angular rate [$^{\text{rad}}/_{\text{s}}$]
I_w	=	Reaction wheel moment of inertia [$\text{kg} \cdot \text{m}^2$]

Scaling Nomenclature

$\bar{\cdot}$	=	An overbar indicates that the variable or parameter has been scaled
M	=	Mass Units
L	=	Length Units
T	=	Time Units
A	=	Angle Units

APPENDIX C:
Spacecraft High Fidelity Model

The high fidelity model from which the optimization and eigenaxis models were derived is presented here. It can be found in Reference [14]. The model was modified to this spacecraft's characteristics and a path constraint that is not considered in this problem was removed.

$$
\text{High Fidelity:} \begin{cases}
\text{Minimize:} & J[\mathbf{x}(\cdot), \mathbf{u}(\cdot), t_f] = t_f \\
\text{Subject to:} & \dot{\mathbf{q}} = \frac{1}{2}\mathbf{Q}(\omega)\mathbf{q} \\
& \dot{\mathbf{h}}_{RW} = \boldsymbol{\tau}_{RW} \\
& \boldsymbol{\omega} = \mathbf{u_1} \\
& \boldsymbol{\tau}_{RW} = \mathbf{u_2} \\
& t_0 = 0 \\
& \mathbf{x}_0 = \left[\mathbf{q}^0, \mathbf{h}_{RW}^0\right]^T \\
& \mathbf{x}_f = \left[\mathbf{q}^f, \mathbf{h}_{RW}^f\right]^T \\
& -\tau_{\max} \leq \tau_{\mathrm{RW}} \leq \tau_{\max} \\
& -h_{\max} \leq h_{\mathrm{RW}} \leq h_{\max} \\
& h(t) \triangleq \mathbf{I}\boldsymbol{\omega} + \mathbf{Z}\mathbf{h}_{RW} = 0
\end{cases}
$$

This high fidelity model takes into account the capabilities of the reaction wheels, as well as torque and angular momentum saturation. It ensures that momentum is conserved by using a constraint that defines the sum of the momentum of the spacecraft and the angular momentum of the reaction wheel assembly in the body-fixed frame to be zero. The model represents the spacecraft's attitude with quaternion representation and includes the actual spacecraft moment of inertia.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D:
Supervised Learning Dataset

The labeled data for time-optimal rotations that was used to train the neural network and regression models is presented in this section.

Table D.1. Supervised learning labeled data, $n = 500$.

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
1	-0.6265	-0.4512	0.6355	0.0801	9.2476
2	0.1120	0.5636	0.8184	0.1489	11.8064
3	-0.5548	0.5979	0.5785	1.9122	71.5721
4	-0.9404	0.0629	-0.3343	2.4987	108.8106
5	0.4730	-0.7356	0.4850	1.3772	57.1958
6	0.0099	-0.9999	0.0027	2.7790	130.7753
7	0.2485	-0.9683	-0.0263	0.9796	52.2292
8	-0.2857	-0.7171	0.6358	0.6209	29.7438
9	0.6615	0.6082	0.4388	0.2231	15.5634
10	-0.2498	-0.7766	-0.5784	1.5555	66.8205
11	-0.6249	0.0981	-0.7745	1.6805	72.7039
12	-0.0708	0.5646	-0.8224	2.4904	105.9342
13	-0.5373	-0.8434	0.0078	0.9187	43.7938
14	0.5759	0.2701	-0.7716	2.6069	106.0559
15	0.8071	0.3403	0.4824	0.7062	33.6040
16	-0.7252	0.3355	0.6013	2.9610	115.9445
17	0.9728	0.2243	0.0586	0.5883	33.7513
18	0.4298	-0.4633	-0.7750	0.3784	20.0895
19	-0.9457	0.3175	0.0701	0.3043	18.8720
20	0.1549	0.8135	0.5606	0.3210	18.5293
21	0.8709	0.4852	-0.0774	0.5550	29.0070

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
22	-0.4988	0.5694	0.6534	2.3820	88.7381
23	0.8341	0.1452	0.5321	0.5381	27.4708
24	-0.6127	0.2728	-0.7417	0.9365	41.5137
25	-0.7924	-0.1405	-0.5936	2.9641	123.0128
26	0.5562	-0.7660	0.3223	1.9937	82.3595
27	-0.3391	0.5949	-0.7288	2.6207	104.0127
28	-0.1014	0.9922	0.0722	2.4193	113.5224
29	0.5818	-0.5006	-0.6410	1.9406	73.1156
30	-0.4348	-0.7578	-0.4865	0.1987	14.6915
31	-0.1549	0.9553	0.2520	2.1067	96.7004
32	-0.6548	0.7557	0.0152	0.0479	6.7436
33	-0.1211	0.9565	0.2654	0.9048	48.0866
34	0.9190	0.3942	0.0120	2.1301	93.3973
35	-0.6544	-0.1229	0.7461	1.4387	62.9082
36	0.3736	0.4033	0.8353	2.9938	120.4141
37	-0.7884	0.2556	0.5595	1.7552	74.1041
38	-0.0922	-0.9800	0.1763	2.0951	98.9804
39	0.3615	-0.9319	0.0295	2.6110	113.4042
40	0.7919	-0.0270	-0.6101	2.4079	102.7230
41	-0.6932	-0.5089	-0.5105	0.7458	32.5431
42	-0.3292	-0.3637	0.8714	2.2851	96.6590
43	-0.6957	0.7180	0.0213	2.2365	96.3992
44	0.5300	0.7566	0.3829	3.1200	121.0292
45	-0.0159	-0.0443	0.9989	1.7722	88.9949
46	0.3593	-0.9173	-0.1717	1.8523	83.5005
47	0.7167	-0.2566	-0.6484	2.5235	102.5456
48	0.8941	0.3229	-0.3102	0.9483	46.6506
49	0.5806	0.5832	0.5682	0.2152	15.3435
50	0.0277	-0.8147	0.5793	0.7326	35.5979
51	-0.7134	-0.1758	0.6783	0.8827	40.1768

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
52	-0.4107	-0.5585	0.7207	2.9219	111.9494
53	0.3425	0.9374	0.0631	0.5569	31.4791
54	-0.8958	0.0194	0.4441	2.6299	112.3392
55	0.4803	-0.5898	0.6492	2.6614	98.9055
56	-0.9144	-0.3849	0.1254	3.1273	132.3074
57	-0.4655	-0.5977	-0.6527	1.4386	57.0277
58	0.9265	0.3263	0.1875	2.7395	117.0523
59	0.6064	-0.3884	-0.6939	2.5118	97.6669
60	0.4393	0.8126	-0.3831	2.1609	89.2272
61	-0.7688	0.3500	0.5351	2.5503	101.7839
62	-0.6675	0.6306	-0.3960	2.3638	92.3673
63	-0.5756	-0.7900	0.2113	0.6885	33.2132
64	-0.9038	-0.3459	-0.2520	0.4903	27.0176
65	-0.2834	-0.6351	-0.7186	0.1540	12.6189
66	-0.2640	-0.8322	0.4877	0.0079	2.8657
67	0.4076	-0.0117	0.9131	0.1820	12.0439
68	-0.0499	-0.2111	0.9762	1.0117	53.4115
69	0.4296	0.6218	0.6548	0.2859	17.5679
70	0.8171	-0.5282	0.2311	1.8606	79.1884
71	0.8587	0.2397	-0.4530	0.7561	37.2574
72	0.9559	-0.0513	-0.2893	0.6702	37.0181
73	-0.3604	0.2892	0.8868	0.0163	3.9041
74	-0.4120	-0.4872	0.7700	2.3607	93.8414
75	0.9084	0.4113	-0.0755	1.8816	83.0699
76	-0.7092	0.6003	0.3698	1.9349	78.1048
77	-0.0970	0.7192	-0.6880	1.1003	49.9492
78	-0.3998	0.8648	-0.3039	2.1513	91.7879
79	0.0799	-0.9557	0.2833	2.0368	93.5405
80	0.5720	0.7484	-0.3358	1.1219	48.8914
81	-0.6189	-0.5737	0.5365	1.4229	55.3501

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
82	0.8625	0.1422	-0.4856	2.0795	89.1389
83	0.0036	-0.9876	0.1572	1.7058	84.8581
84	-0.3133	-0.8334	0.4552	1.4579	64.2187
85	0.9731	0.2302	-0.0034	2.3879	107.7040
86	0.5268	-0.6627	-0.5323	2.4115	89.9254
87	-0.8473	-0.3694	0.3815	2.0237	85.7438
88	0.1734	-0.4704	-0.8652	1.8366	79.4191
89	0.7844	-0.3856	0.4859	0.2687	16.6017
90	0.9643	-0.0924	0.2484	2.3762	106.6353
91	0.6242	0.3298	-0.7083	2.7487	108.4174
92	-0.5230	0.8250	-0.2142	2.0695	88.1255
93	-0.2575	-0.7448	-0.6156	1.2889	55.9517
94	0.5850	-0.7158	0.3813	0.0898	9.8856
95	0.4754	-0.1566	0.8657	0.0541	6.9618
96	-0.9125	-0.1528	-0.3794	2.5676	109.8956
97	-0.4214	0.6667	-0.6148	1.7908	70.8645
98	0.7260	-0.6256	-0.2855	2.3089	94.3566
99	0.4009	0.8137	0.4209	0.3726	20.3155
100	-0.7285	-0.3233	-0.6040	1.7224	70.9472
101	-0.1397	0.9206	-0.3647	1.6461	75.9606
102	-0.5098	-0.0445	0.8591	1.3971	62.2975
103	0.5449	0.5490	0.6338	2.2552	82.8640
104	0.5235	0.4654	-0.7137	2.0177	78.6454
105	-0.0659	0.6823	-0.7281	0.8125	38.3064
106	-0.0633	-0.8257	-0.5605	3.0811	129.9339
107	-0.1445	0.8650	-0.4806	2.0511	88.6411
108	-0.4287	0.5967	0.6783	1.3283	54.0628
109	0.6805	-0.7121	-0.1726	0.4376	23.1772
110	0.4095	-0.6258	-0.6638	1.5939	63.9836
111	-0.1279	0.9614	0.2436	1.5539	75.6464

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
112	0.3214	-0.9466	0.0256	3.0487	131.3356
113	-0.4184	0.8433	-0.3373	1.8161	78.2769
114	-0.9338	-0.2314	-0.2731	1.1233	55.8146
115	0.6347	-0.7437	-0.2100	1.5158	65.5296
116	-0.6777	0.6230	-0.3908	1.2550	52.2343
117	-0.2830	-0.3052	0.9093	1.6779	76.2182
118	-0.8008	0.4107	0.4359	0.0309	5.6504
119	-0.7734	-0.6308	0.0623	2.4794	105.7400
120	-0.4372	-0.7260	-0.5308	2.2298	87.4622
121	0.4398	-0.6042	0.6645	1.8419	72.1772
122	0.5027	0.4617	-0.7308	1.2710	52.3657
123	0.7712	0.4717	-0.4274	1.3034	55.3120
124	0.9152	-0.3596	-0.1816	2.2217	96.4844
125	-0.0167	-0.2042	-0.9788	2.1398	99.7532
126	0.8446	-0.0252	0.5349	1.2604	56.6161
127	-0.6383	-0.7691	-0.0331	1.7569	77.0035
128	0.4271	-0.3982	-0.8118	1.3770	59.6409
129	0.2762	0.6252	-0.7300	1.6436	69.2628
130	-0.4058	-0.7487	-0.5242	1.0413	45.2674
131	-0.8719	-0.4838	0.0754	1.5857	70.2619
132	0.3520	-0.5264	-0.7739	0.8335	37.5633
133	0.6548	0.3119	-0.6884	2.4570	98.2453
134	0.6550	0.5879	0.4747	0.2379	16.0910
135	0.6498	-0.6264	0.4306	2.0222	78.8245
136	-0.6154	-0.6007	0.5104	2.8486	103.5240
137	0.7996	-0.2656	0.5386	1.7302	73.1580
138	0.7952	0.5657	-0.2182	0.5195	25.9696
139	0.3430	0.3194	0.8834	0.5070	27.1615
140	0.5487	0.7124	-0.4376	0.0827	9.5313
141	0.4836	0.4240	-0.7658	0.5425	25.8022

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
142	-0.0673	-0.8937	0.4436	2.0467	89.4600
143	-0.9767	0.1880	0.1037	0.0308	4.9245
144	-0.6699	-0.1587	0.7253	0.0671	7.9076
145	-0.5872	0.3475	0.7310	1.9410	78.6907
146	-0.3565	0.6780	-0.6429	0.3487	19.5182
147	0.7815	0.5041	0.3677	2.0278	82.8166
148	0.3983	-0.9126	0.0927	2.3480	102.1257
149	-0.7969	-0.5295	0.2908	0.7808	36.3170
150	-0.6111	0.5347	-0.5837	1.8231	68.3126
151	0.6462	-0.6856	-0.3352	0.3386	19.2497
152	0.7432	0.1158	0.6589	1.3947	61.1877
153	-0.6524	-0.7466	0.1300	0.8410	39.5431
154	0.4912	-0.8705	0.0314	1.2410	57.3527
155	-0.6460	0.6206	0.4443	1.5033	59.7719
156	-0.4780	-0.8662	0.1457	0.9074	44.0205
157	-0.3495	-0.0365	0.9362	2.6746	115.3636
158	0.6527	0.0882	0.7525	1.2710	56.3158
159	0.6017	-0.6921	0.3986	1.1715	49.3155
160	-0.6978	-0.6802	-0.2243	2.7690	113.1750
161	0.8242	0.4734	-0.3109	1.3576	59.4262
162	-0.8919	-0.2943	-0.3434	2.7310	114.4796
163	-0.3962	-0.2302	0.8888	2.3051	98.0166
164	-0.8182	0.0246	0.5743	2.2969	98.2537
165	0.5007	0.1255	0.8565	0.3193	18.0664
166	-0.0754	-0.0639	-0.9951	1.9564	95.2973
167	0.2259	0.7554	-0.6150	2.3965	99.3658
168	-0.8627	-0.4197	0.2822	1.8482	79.7669
169	0.8248	-0.3933	0.4063	2.8914	116.1525
170	-0.2033	0.4825	-0.8520	1.0281	47.6554
171	0.7920	0.3885	-0.4709	1.2591	54.4961

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
172	-0.6806	0.5216	0.5145	1.4055	55.3970
173	0.3298	-0.3471	-0.8779	0.1222	10.7685
174	-0.7882	0.3920	0.4744	1.0579	46.7556
175	-0.2289	-0.4153	0.8804	2.5882	108.7767
176	-0.4545	0.5309	-0.7152	1.8715	73.7423
177	0.6393	-0.2593	0.7239	0.0751	8.6054
178	-0.6424	0.4182	0.6423	2.4581	94.3174
179	-0.6811	-0.3548	-0.6405	2.6549	103.9268
180	0.6498	-0.6847	-0.3300	0.8612	38.4553
181	0.7219	-0.5941	0.3549	1.0911	46.8568
182	0.7340	-0.0788	-0.6746	2.9806	124.6285
183	-0.2061	-0.5568	-0.8046	2.8900	119.2259
184	-0.2884	-0.6112	-0.7371	0.6096	28.9599
185	-0.8589	-0.4455	0.2526	1.9593	83.6064
186	-0.2418	-0.9510	-0.1929	0.6629	36.9803
187	0.4215	-0.6897	0.5888	2.2647	88.1292
188	-0.0428	-0.0708	0.9966	0.4604	28.0862
189	-0.0114	0.5016	-0.8650	0.6539	32.9341
190	0.4143	-0.6518	-0.6353	2.8732	109.2286
191	0.7650	0.0474	0.6423	1.5692	68.6832
192	-0.7926	-0.0254	-0.6092	1.1151	50.2273
193	-0.4479	-0.4923	0.7463	1.3373	55.3709
194	0.1679	0.9852	0.0359	2.5571	116.8595
195	0.5043	-0.7421	0.4416	1.7928	72.5008
196	-0.3084	-0.6339	0.7093	2.9391	116.3417
197	0.8569	-0.3486	-0.3799	2.0833	88.4566
198	0.3796	-0.7023	-0.6022	1.7408	70.5744
199	-0.9773	-0.1691	-0.1274	0.8438	45.9937
200	0.7990	0.5935	0.0961	0.2578	15.9284
201	0.3600	-0.0013	-0.9329	1.3309	63.8315

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
202	-0.4213	0.3473	-0.8378	1.2260	55.0332
203	-0.3461	-0.5484	-0.7612	1.0513	45.6728
204	0.3192	0.6741	0.6661	0.9241	40.7992
205	0.5655	0.3988	0.7219	1.9086	76.2377
206	-0.8244	0.4410	-0.3548	2.9449	118.4392
207	-0.6086	-0.5069	0.6105	2.8991	105.1095
208	-0.2901	0.5771	0.7634	2.3248	95.2194
209	-0.2360	-0.2722	0.9329	2.2892	101.1594
210	-0.6842	0.5201	0.5112	0.7681	33.3545
211	0.5127	0.6658	-0.5420	1.0397	42.8502
212	-0.3698	-0.1428	-0.9181	2.9522	125.1680
213	-0.5239	0.4016	0.7512	0.8149	36.1441
214	-0.2026	0.8120	-0.5474	2.5392	106.2665
215	0.8262	-0.0790	0.5578	2.2228	95.0289
216	-0.0483	0.8018	-0.5956	2.5802	110.1526
217	0.8215	-0.1807	-0.5408	2.7211	113.3682
218	-0.7145	0.6224	0.3196	1.3356	56.7144
219	0.2621	-0.6940	0.6705	3.0339	121.4384
220	-0.7090	-0.5947	-0.3791	2.5577	100.1420
221	-0.4979	0.5680	-0.6553	2.5755	95.4535
222	0.8187	-0.5219	0.2395	1.3886	60.5773
223	0.8113	0.5826	0.0498	2.0883	89.8867
224	0.5185	-0.5920	-0.6170	2.5351	92.6308
225	-0.1380	-0.6448	-0.7518	0.3624	20.0052
226	-0.6449	-0.2827	0.7100	2.5913	104.2599
227	-0.7670	0.2985	-0.5679	2.9781	118.5607
228	0.4269	0.8450	0.3221	2.4288	101.0851
229	0.9931	0.0480	0.1070	2.8839	130.2935
230	0.4744	-0.1169	-0.8725	0.5446	28.4960
231	-0.1082	-0.9244	-0.3657	1.1656	57.3082

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
232	0.6161	-0.5934	0.5179	0.4738	23.9351
233	0.3209	0.9442	0.0746	2.5077	109.8867
234	-0.1344	0.6480	0.7497	1.3549	59.6170
235	-0.0585	-0.7567	0.6511	0.3667	20.5626
236	-0.5829	-0.7873	0.2009	1.7616	75.7300
237	-0.1994	0.5793	0.7904	1.2126	53.4102
238	0.3972	0.5075	0.7646	2.4233	96.2314
239	0.8038	-0.2852	-0.5221	2.8599	115.6364
240	0.4197	-0.8191	0.3912	2.0620	86.1689
241	0.7096	-0.3073	-0.6341	3.1075	122.0555
242	-0.1631	-0.8760	-0.4539	1.2019	56.1692
243	0.6226	0.5384	0.5679	1.1017	44.4825
244	-0.4576	0.5659	-0.6858	2.0575	79.5192
245	-0.5653	0.6202	0.5439	2.1401	79.3096
246	-0.3865	0.1673	0.9070	2.1775	94.1351
247	-0.6978	-0.6933	0.1801	2.4772	103.3773
248	0.1155	0.8703	0.4787	1.6598	73.4760
249	0.4934	0.6000	-0.6298	1.0045	41.3965
250	-0.6694	0.4428	0.5965	0.7041	30.9433
251	-0.5892	0.6039	-0.5368	2.1693	80.1939
252	-0.4221	0.5329	0.7333	1.3166	54.3860
253	0.5322	-0.7567	-0.3798	1.8993	77.6134
254	0.7383	0.5449	-0.3974	2.6853	104.7808
255	0.2658	0.5627	0.7827	0.1759	13.3191
256	0.6438	-0.4586	-0.6125	2.8078	104.6939
257	-0.4865	0.8167	-0.3104	1.7834	75.7543
258	-0.1444	0.9721	-0.1847	0.7233	40.6073
259	0.6197	-0.7593	-0.1984	1.6292	70.2822
260	0.7190	-0.0572	-0.6927	0.6222	30.2583
261	-0.7968	0.3662	-0.4805	1.9215	79.1961

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
262	0.4219	0.9019	-0.0932	2.1207	92.7021
263	0.0470	0.2080	-0.9770	1.8773	89.2491
264	0.5779	-0.7367	-0.3512	0.9946	43.6996
265	-0.0348	-0.6822	0.7303	1.5254	67.2032
266	-0.3898	0.8575	0.3357	0.9710	46.5226
267	0.5815	0.4805	0.6565	3.0499	112.0223
268	0.9321	0.1709	-0.3192	0.6407	34.7803
269	-0.1479	-0.7571	0.6363	0.4745	24.7355
270	-0.2723	0.6317	0.7258	0.3963	20.8801
271	0.1698	0.5505	0.8174	0.5924	29.2569
272	-0.3933	0.3048	0.8674	0.1963	13.6574
273	-0.1208	-0.6092	-0.7837	1.5400	67.1866
274	-0.8202	-0.0192	-0.5717	0.1323	10.4784
275	0.1340	-0.9294	0.3438	2.6961	116.3694
276	-0.0547	0.4295	0.9014	2.6976	115.1974
277	0.8454	0.4267	0.3213	2.9307	119.3375
278	-0.5963	0.3242	0.7344	0.7521	33.8596
279	0.7258	-0.2697	0.6328	1.3124	56.0900
280	0.2035	0.6990	0.6855	2.6305	108.4877
281	-0.8263	0.3829	0.4131	1.6128	69.3864
282	-0.2449	0.7734	-0.5847	2.6164	107.4934
283	-0.6312	0.7482	-0.2044	0.2914	17.4238
284	0.5594	0.8203	-0.1192	0.2599	16.2197
285	0.4596	0.8576	0.2308	1.6678	72.9885
286	-0.8054	-0.3332	-0.4902	1.3637	58.8237
287	0.8543	0.0729	0.5147	3.1095	130.9123
288	-0.7265	-0.0667	0.6839	3.0350	127.0233
289	0.9765	-0.1963	-0.0894	0.1616	11.6864
290	-0.6090	0.7575	-0.2350	2.3456	97.3626
291	-0.6971	0.3350	-0.6340	1.6571	67.8851

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
292	0.3509	0.3951	0.8490	0.1162	10.6659
293	-0.9272	0.2699	0.2595	2.4481	106.8677
294	0.8693	0.2804	-0.4069	0.8228	40.4574
295	-0.5466	0.0442	-0.8362	0.4069	21.7447
296	0.6967	0.6815	0.2239	2.0000	83.8313
297	-0.3587	-0.9312	0.0645	0.6372	34.9868
298	-0.5886	-0.5627	-0.5804	0.1468	12.6679
299	-0.7191	-0.4993	-0.4833	0.2929	17.6925
300	0.3385	0.9081	-0.2465	1.0683	52.7094
301	-0.5200	-0.6153	-0.5925	2.1699	80.3768
302	0.4204	0.5551	-0.7177	2.2525	88.1652
303	-0.3193	0.2649	0.9099	1.0081	49.6737
304	-0.1416	0.9826	-0.1206	2.8280	127.0512
305	-0.8010	-0.1548	0.5783	1.1689	51.8777
306	0.5880	-0.5726	-0.5712	2.6664	96.4102
307	0.8126	0.4741	0.3390	2.1262	87.7401
308	-0.4341	-0.7118	-0.5521	2.8242	107.7893
309	-0.9074	-0.1290	-0.4000	2.2437	97.0697
310	-0.5776	-0.8075	-0.1199	0.0450	6.7107
311	0.1644	0.6587	-0.7342	3.0611	125.9021
312	0.0843	0.8536	-0.5140	1.5712	69.5848
313	0.3071	-0.9202	0.2426	0.8856	45.5908
314	0.6541	-0.3091	0.6904	1.9777	80.4332
315	-0.1380	0.9865	0.0886	0.4224	26.3884
316	0.2816	-0.0036	0.9595	0.5326	30.4882
317	-0.9668	-0.2549	0.0173	0.7335	40.4067
318	-0.1410	0.4637	0.8747	1.2353	56.5574
319	0.9423	0.3277	-0.0684	1.1348	56.4824
320	0.5406	-0.6349	-0.5520	2.7598	100.1047
321	0.3260	0.8714	-0.3665	1.9885	86.3280

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
322	0.5911	0.6818	0.4310	1.5267	61.4416
323	-0.7430	0.4776	0.4689	1.2267	51.3711
324	0.3186	0.5480	0.7734	2.1941	89.7797
325	-0.6605	0.0614	0.7483	0.7690	36.1813
326	0.3276	0.2384	-0.9142	1.6901	77.0111
327	-0.8426	0.3704	0.3909	1.8288	78.2656
328	0.7989	-0.4010	0.4483	3.1398	123.3288
329	0.5368	0.7089	-0.4575	2.9087	110.1251
330	0.7423	-0.4256	0.5175	1.6145	65.5544
331	-0.6072	-0.2307	-0.7603	0.4278	21.8651
332	0.5148	-0.5461	-0.6609	1.9718	74.5908
333	0.4674	-0.6458	-0.6037	0.6700	30.3981
334	-0.4625	0.7499	0.4730	2.4504	96.4543
335	0.1826	0.9582	-0.2201	2.5306	113.2183
336	-0.6720	0.1118	0.7320	1.9824	84.6699
337	-0.0823	-0.9685	0.2349	0.4489	27.2397
338	-0.0855	0.8384	-0.5383	1.8028	78.6974
339	-0.1864	0.8019	-0.5676	1.4957	65.4493
340	0.7380	0.0718	0.6710	3.0181	126.2793
341	0.4738	0.8248	-0.3086	0.3800	20.9217
342	0.4148	0.7977	0.4378	1.6921	71.5067
343	-0.1035	-0.6887	-0.7177	1.9683	84.5390
344	0.3305	-0.5834	-0.7419	1.0536	45.6153
345	-0.8179	0.1301	-0.5605	2.2022	93.7164
346	-0.4430	0.4828	0.7554	1.4661	60.4734
347	0.5214	0.6609	0.5397	1.4858	58.0635
348	-0.4170	-0.7616	0.4960	3.0070	116.7194
349	0.0842	0.8763	0.4744	2.9195	124.0089
350	0.7487	-0.6240	0.2239	0.9867	44.3659
351	-0.7970	0.4481	-0.4049	0.8742	39.9825

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
352	0.1406	0.3106	0.9401	2.8500	122.2074
353	0.5891	-0.5634	0.5793	2.8720	103.1727
354	-0.5613	-0.7816	0.2722	0.3026	17.9096
355	-0.0017	0.5976	-0.8018	0.7648	36.1410
356	0.9565	0.0940	-0.2763	1.6073	76.8111
357	0.5289	-0.6783	-0.5100	0.6500	29.8896
358	-0.5697	-0.7851	-0.2430	0.1012	10.2962
359	-0.1665	-0.9835	-0.0713	0.0117	3.1660
360	0.5438	0.3361	0.7689	1.0612	46.1803
361	-0.0200	0.7160	0.6979	0.7793	37.1952
362	-0.8300	-0.1841	-0.5266	0.5763	28.9494
363	-0.4110	-0.5973	-0.6887	1.1920	49.4398
364	-0.7404	0.2450	0.6259	2.6470	107.7090
365	-0.5830	-0.3320	-0.7415	0.7938	35.4174
366	0.7555	-0.5735	-0.3167	3.0121	119.1876
367	0.1906	0.7906	0.5819	0.6803	32.8837
368	0.6245	-0.2545	-0.7384	2.7981	113.0233
369	0.7391	-0.1669	0.6526	1.2876	56.4041
370	0.9644	-0.2023	-0.1704	1.8212	86.0065
371	-0.7947	-0.5168	0.3184	1.2455	54.0199
372	-0.6144	-0.7889	-0.0119	1.0718	49.3265
373	0.1841	-0.9775	-0.1028	1.1606	60.6229
374	0.2002	-0.7327	0.6504	0.8703	40.0829
375	0.6358	0.0083	-0.7718	0.2048	13.3986
376	-0.7999	0.4553	-0.3909	0.9015	41.1363
377	0.3579	0.5080	0.7835	2.7299	108.8463
378	0.8760	-0.4568	-0.1551	0.3568	20.2620
379	-0.6992	0.6492	-0.2994	2.5141	101.3692
380	-0.3094	-0.8448	0.4367	2.1024	88.7675
381	-0.4813	-0.3546	-0.8017	2.9706	117.9924

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
382	-0.7311	-0.5129	0.4499	1.7070	68.3442
383	0.7147	-0.6914	0.1054	2.2650	96.5689
384	-0.6722	0.4700	0.5721	1.2185	49.1196
385	-0.4823	-0.5629	0.6712	1.5194	59.7510
386	0.6408	0.7272	0.2460	1.3029	56.6352
387	-0.3743	0.8630	0.3391	0.6941	35.3739
388	0.2239	0.4803	0.8480	2.9059	120.2507
389	-0.8764	0.4331	-0.2105	1.0254	48.8056
390	0.2400	0.6841	0.6887	0.4793	24.3743
391	0.4405	0.7883	-0.4297	0.1011	10.4487
392	-0.7614	0.3723	-0.5308	2.4647	97.9495
393	0.6366	-0.3874	-0.6668	0.3318	18.6699
394	-0.1804	0.6058	0.7749	0.4534	23.3297
395	-0.3887	0.6654	0.6373	0.2419	16.1222
396	0.4360	0.6900	-0.5777	2.7917	105.9288
397	0.2054	0.9250	-0.3198	0.4263	25.0478
398	0.6866	0.5625	0.4606	0.6511	29.6090
399	0.5886	0.4867	-0.6455	0.2402	16.0363
400	0.5933	-0.6469	0.4791	3.0615	112.6652
401	0.3066	-0.3097	-0.9001	1.5984	72.6162
402	-0.2523	0.4996	-0.8287	2.7120	111.8933
403	-0.5400	0.6284	-0.5599	0.7185	32.0745
404	0.5308	-0.8367	-0.1351	1.4741	65.2720
405	-0.8702	-0.0874	0.4848	1.8933	82.0759
406	0.5557	0.5691	0.6061	0.9424	39.2280
407	-0.8030	0.0917	-0.5889	1.7641	76.2107
408	-0.5919	-0.7977	-0.1149	3.1382	130.8816
409	0.3204	-0.4176	0.8503	1.9236	81.8379
410	0.7018	-0.3322	0.6301	0.2774	16.7892
411	-0.4405	-0.3282	-0.8356	0.9609	44.6187

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
412	0.5983	-0.6731	0.4347	0.1682	13.5714
413	-0.8360	-0.4454	0.3206	2.3672	97.8113
414	0.9571	0.2514	0.1440	1.7094	80.8254
415	-0.6432	-0.5771	0.5032	2.7861	102.1828
416	-0.6527	-0.6969	-0.2972	2.4297	98.2710
417	0.6396	0.7098	-0.2950	2.1835	89.1791
418	-0.8323	-0.3940	0.3899	0.9832	45.5487
419	0.8574	-0.4696	0.2106	0.0237	4.8182
420	0.2405	0.8229	0.5148	0.6114	30.6809
421	0.6224	0.5077	-0.5957	1.8966	71.1564
422	0.5730	-0.6943	0.4353	3.0983	116.6390
423	-0.5787	0.0562	-0.8136	0.0576	7.0135
424	0.6719	0.2919	-0.6807	0.6940	31.8353
425	0.4512	-0.3846	-0.8053	2.8334	112.9630
426	-0.3433	0.7167	0.6070	1.1498	49.3957
427	0.8208	-0.5576	-0.1242	1.5632	68.3503
428	-0.3875	-0.1824	-0.9036	1.1902	56.6474
429	0.7791	0.5706	0.2596	2.4698	101.7948
430	-0.3497	0.6324	-0.6912	1.0872	46.5827
431	0.5978	-0.1474	0.7880	2.1177	89.8641
432	-0.7268	-0.2425	0.6426	0.8975	40.2780
433	0.5959	-0.5605	-0.5750	1.5757	60.1681
434	0.3469	0.0402	0.9370	3.0275	129.3556
435	0.2195	-0.1230	0.9678	1.4536	71.5702
436	-0.2530	0.0832	-0.9639	1.4163	69.6939
437	0.0918	-0.8168	0.5695	0.4519	24.0139
438	-0.0869	0.8229	0.5615	2.7770	117.6537
439	-0.0349	0.8024	0.5957	2.7171	115.7542
440	-0.0770	-0.7520	-0.6546	3.0766	128.8488
441	-0.0169	0.6885	-0.7251	0.5934	29.5817

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
442	-0.7612	0.5903	-0.2686	1.8008	75.8068
443	-0.6793	-0.7180	-0.1520	1.4217	62.3973
444	0.0555	-0.0074	0.9984	1.1068	60.0214
445	-0.0789	0.1990	0.9768	0.5749	33.0936
446	0.8801	-0.0295	0.4739	1.5387	68.7483
447	0.0732	-0.0994	-0.9923	0.0937	8.0210
448	-0.4919	-0.4071	-0.7696	1.6478	67.7155
449	0.0360	-0.6863	0.7265	1.4831	65.4921
450	0.8721	0.4569	0.1750	0.7992	39.3048
451	-0.9337	0.3353	-0.1259	0.3969	23.2480
452	-0.5059	-0.4141	-0.7567	3.0677	118.1862
453	-0.4677	-0.8646	0.1835	2.3525	100.4609
454	-0.7252	-0.1256	-0.6769	1.3507	59.2927
455	0.4069	-0.8755	-0.2607	1.7411	77.0860
456	-0.4154	-0.3292	0.8480	2.8103	114.7175
457	0.2920	0.5910	0.7520	2.6808	108.1053
458	0.8492	0.5162	0.1117	1.4744	65.1589
459	-0.5360	-0.6128	-0.5807	2.1884	80.8584
460	0.5917	-0.6349	-0.4968	0.8637	36.9615
461	-0.0375	0.9909	0.1294	0.9504	52.4169
462	0.6306	-0.6514	0.4219	2.5963	99.4997
463	0.2818	-0.8608	0.4237	2.2870	96.6534
464	0.2359	-0.1069	0.9659	0.1777	12.3261
465	-0.5491	-0.5650	0.6158	0.1261	11.7323
466	0.8354	0.3332	0.4372	2.7611	112.4692
467	-0.8595	0.5106	-0.0240	2.2483	96.6996
468	0.0782	0.2636	-0.9615	2.9874	129.1738
469	0.7620	0.5007	-0.4107	1.4137	58.9080
470	0.2831	-0.7003	0.6554	1.4497	61.4418
471	-0.3254	-0.2197	0.9197	2.7082	115.2481

Continued on next page

Table D.1 – *Continued from previous page*

Data Point	e_1	e_2	e_3	ψ (radians)	t_f (seconds)
472	0.9078	0.3943	0.1429	3.0839	130.1808
473	-0.3324	0.9305	0.1540	2.8183	121.1262
474	-0.5104	0.8590	0.0405	0.8434	41.2135
475	0.1600	0.6724	-0.7227	0.2651	16.2527
476	-0.3288	-0.9055	0.2683	1.6268	74.7352
477	-0.1203	0.0893	-0.9887	2.3321	108.9069
478	0.9437	0.3293	-0.0321	0.2080	13.8725
479	0.8518	-0.0515	-0.5213	0.5179	27.0215
480	-0.4050	0.3493	0.8450	1.1443	52.2048
481	0.3201	-0.7452	0.5849	2.3181	93.9448
482	-0.5794	0.6039	-0.5474	2.6418	95.7617
483	0.8907	-0.3825	0.2459	2.8563	119.4223
484	0.7941	0.0125	-0.6076	3.0954	130.5968
485	-0.9320	0.2301	-0.2800	1.7188	79.4973
486	0.1958	0.6300	0.7515	0.8225	37.9692
487	0.8113	-0.1491	0.5654	1.1407	50.8706
488	-0.5456	-0.6927	-0.4716	3.0026	112.2904
489	0.1943	0.7599	-0.6203	0.4823	24.8988
490	0.0190	0.9867	0.1616	0.9643	52.8040
491	0.7291	-0.1966	0.6556	0.3498	19.0036
492	0.2530	-0.6050	0.7550	2.2066	91.5131
493	0.8629	-0.5049	-0.0234	2.0030	86.7206
494	0.5714	-0.6312	-0.5245	0.4069	21.7857
495	-0.3269	-0.6891	-0.6467	0.5112	25.1393
496	-0.3133	0.6211	0.7183	1.1940	51.2675
497	0.7138	-0.1978	0.6718	0.2355	14.9402
498	-0.6484	-0.5777	-0.4958	0.5787	27.3095
499	-0.2192	-0.6217	0.7519	2.3217	96.6485
500	-0.1720	0.6955	0.6976	0.4993	25.4664

List of References

[1] 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, pp. 446–452, 1993. Available: http://arc.aiaa.org/doi/full/10.2514/3.21030

[2] M. Karpenko, S. Bhatt, N. Bedrossian, and I. M. Ross, "Flight implementation of shortest-time maneuvers for imaging satellites," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1069–1079, 2014. Available: https://doi.org/10.2514/1.62867

[3] K. Bollino, L. R. Lewis, P. Sekhavat, and I. M. Ross, "Pseudospectral optimal control: A clear road for autonomous intelligent path planning," in *AIAA Infotech@Aerospace 2007 Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 2007. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2007-2831

[4] D. E. Smith, "Choosing objectives in over-subscription planning," in *Proceedings of the Fourteenth International Conference on International Conference on Automated Planning and Scheduling* (ICAPS'04). Whistler, British Columbia: AAAI Press, 2004, pp. 393–401.

[5] X. Yang, "Genetic algorithms," in *Nature-Inspired Optimization Algorithms*, 1st ed. London: Elsevier, 2014, pp. 77–87.

[6] B. Wie, *Space Vehicle Dynamics and Control*, 2nd ed. Reston, VA: American Institute of Aeronautics and Astronautics, 2008. Available: http://arc.aiaa.org/doi/book/10.2514/4.860119

[7] J. R. Wertz, "Space mission geometry," in *Space Mission Engineering: the New SMAD* (Space Technology Library). Hawthorne, CA: Microcosm Press, 2011, pp. 149–196.

[8] I. M. Ross, R. J. Proulx, and M. Karpenko, "Autonomous UAV sensor planning, scheduling and maneuvering: An obstacle engagement technique," in *2019 American Control Conference (ACC)*. Philadelphia, PA, USA: IEEE, July 2019, pp. 65–70. Available: https://ieeexplore.ieee.org/document/8814474/

[9] J. P. Abramson, *College Algebra* (OpenStax). Houston, TX: Rice University, 2017. Available: https://openstax.org/details/books/college-algebra

[10] N. Gunantara, "A review of multi-objective optimization: Methods and its applications," *Cogent Engineering*, vol. 5, no. 1, July 2018. Available: https://www.cogentoa.com/article/10.1080/23311916.2018.1502242

[11] J. Kirk, "Traveling Salesman Problem—Genetic algorithm," May 2014. Available: https://www.mathworks.com/matlabcentral/fileexchange/13680

[12] I. M. Ross, "Enhancements to the DIDO Optimal Control Toolbox," *arXiv:2004.13112 [cs, math]*, Apr. 2020, arXiv: 2004.13112. Available: http://arxiv.org/abs/2004.13112

[13] I. M. Ross, *A Primer on Pontryagin's Principle in Optimal Control*, 2nd ed. San Francisco: Collegiate Publishers, Mar. 2015.

[14] T. Lippman, J. M. Kaufman, and M. Karpenko, "Autonomous planning of constrained spacecraft reorientation maneuvers" (Advances in the Astronautical Sciences), Stevenson, WA, 2017.

[15] I. M. Ross, Q. Gong, M. Karpenko, and R. J. Proulx, "Scaling and balancing for high-performance computation of optimal controls," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 10, pp. 2086–2097, 2018. Available: https://doi.org/10.2514/1.G003382

[16] D. C. Montgomery, *Design and Analysis of Experiments*, 8th ed. Hoboken, NJ: John Wiley & Sons Inc, 2013.

[17] SAS Institute Inc. 2017, *JMP® 13 Design of Experiments Guide*, 2nd ed. Cary, NC: SAS Institute Inc.

[18] S. Skansi, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence* (Undergraduate Topics in Computer Science). Cham: Springer International Publishing, 2018. Available: http://link.springer.com/10.1007/978-3-319-73004-2

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). New York, NY: Springer New York, 2009. Available: http://link.springer.com/10.1007/978-0-387-84858-7

[20] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018. Available: http://link.springer.com/10.1007/978-3-319-94463-0

[21] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning* (Springer Texts in Statistics). New York, NY: Springer New York, 2013, vol. 103. Available: http://link.springer.com/10.1007/978-1-4614-7138-7

[22] B. Illowsky and S. Dean, *Introductory Statistics* (OpenStax). Houston, TX: Rice University, 2018. Available: https://openstax.org/details/books/introductory-statistics

[23] F. E. Harrell, Jr., *Regression Modeling Strategies* (Springer Series in Statistics). Cham: Springer International Publishing, 2015. Available: http://link.springer.com/10.1007/978-3-319-19425-7

[24] A. Fleming, P. Sekhavat, and I. M. Ross, "Constrained, minimum-time maneuvers for CMG actuated spacecraft," *Advances in the Astronautical Sciences*, vol. 135, pp. 1009–1028, 2009.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California