# SIMULATION TESTBED FOR ENTRY ANALYSIS AND DESIGN (STEAD)

A Thesis

by

MATTHEW JAMES HOLUB

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2019

Major Subject: Aerospace Engineering

## ABSTRACT

The simulation of a spacecraft in an accurately modeled environment gives engineers and researchers important data on the feasibility of designs. The Simulation Testbed for Entry Analysis and Design (STEAD) is a modular simulation analysis tool for entry vehicles, capable of testing any spacecraft on any planetary body. Although there are existing software packages in use to perform these simulations, STEAD is structured to reduce the workload for the user to generate new environments for each simulation and provide a starting point from which to build, test, and analyze systems.

STEAD is built on the SpaceCRAFT platform, allowing the user to connect new models and simulate engineering designs in a virtual reality (VR) environment. Although SpaceCRAFT can be used for many different space mission scenarios, this thesis focuses on the simulation and analysis of a lifting body entry vehicle. The integration of existing environmental models, such as NASA GRAMs and JPL SPICE, provide an accurate simulation environment for STEAD. Additionally, the first iteration of a generic atmospheric model was developed to accurately simulate the physical processes on any planetary body. The thesis adds functionality to the core SpaceCRAFT platform through the implementation of a dynamics propagation system and creation of interfaces for MATLAB and FORTRAN mathematical models. To demonstrate a use case for the software, STEAD was used to simulated the trajectory of a test vehicle using guidance commands provided by both an open-loop and closed-loop algorithm based on the Theory of Connections. The simulation tool was used to analyse the performance of the real-time guidance algorithm in an accurately modeled environment that provided realistic perturbations, such as density variation and wind speeds.

## DEDICATION

To my friends and family for their love and support.

## ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Dr. Gregory Chamitoff for his encouragement, insightful discussions, and financial support during my graduate studies. I would also like to thank Dr. Srinivas Vadali for his technical advice and assistance with the optimal guidance problem. I thank Dr. Ann McNamara for her participation on my committee and her insight into the advantages virtual reality can bring to a human-in-the-loop simulation.

I want to thank all the undergradute and graduate students of the ASTRO Center for their contributions to the SpaceCRAFT platform, especially Connor Jakubik and Eric Munoz for their work on the core systems, Sournov Bhattacharya for his assistance with the TCP server-client architecture, and Nicholas Anderson and Colton Marik for their work on the Generic Atmospheric Model.

I would also like to thank Daniel Green and Michelle Gilbert for their support during this project, and for allowing me to discuss issues and possible pathways for solving difficult problems. I only understand the Theory of Connections to my current level due to the time and effort of Hunter Johnston, who helped me learn the intricacies of a new optimal control method when few published papers were available.

Finally, I would like to thank my family and friends for their patience throughout this long and difficult processes. Without them, I would not be where I am today.

## CONTRIBUTORS AND FUNDING SOURCES

### Contributors

This work was supported by a thesis committee consisting of Dr. Gregory Chamitoff and Dr. Srinivas Vadali of the Department of Aerospace Engineering and Dr. Ann McNamara of the Department of Visualization.

The Generic Atmospheric Model was developed with assistance from the Environmental Modeling team in the SpaceCRAFT AggiE_Challenge course. The test vehicle model was created by Alex Sein of the SpaceCRAFT Tech Art team. The SpaceCRAFT platform has been continually developed by the ASTRO Lab at Texas A&M, with the assistance of the author and numerous undergraduate and graduate students.

All other work conducted for the thesis was completed by the author independently.

### Funding Sources

Graduate study was supported by a Graduate Assistance Research position from Texas A&M University. SpaceCRAFT development in the ASTRO Lab has been supported by a Space Act Agreement with NASA.

# NOMENCLATURE

| | |
|---|---|
| EDL | Entry, Descent, and Landing |
| STEAD | Simulation Testbed for Entry Analysis and Design |
| GAM | Generic Atmospheric Model |
| TPBVP | Two Point Boundary-Value Problem |
| ToC | Theory of Connections |
| NASA | National Aeronautics and Space Administration |
| POST | Program to Optimize Simulated Trajectories |
| POST2 | Program to Optimize Simulated Trajectories II |
| 6-DOF | Six degree of freedom |
| VR | Virtual Reality |
| GRAM | Global Reference Atmospheric Model |
| TAEM | Terminal Area Energy Management System |
| ASTRO | AeroSpace, Technology Research and Operations |
| JPL | Jet Propulsion Laboratory |
| UE4 | Unreal Engine 4 |
| TCP | Transmission Control Protocol |
| System | mathematical model that provides information to Entities in SpaceCRAFT environment |
| Entity | object simulated in the SpaceCRAFT environment |
| LVLH | Local-Vertical Local Horizontal |
| p | pressure |
| $\rho$ | density |

| | |
|---|---|
| $S$ | reference area |
| $m$ | mass |
| $C_L$ | coefficient of lift |
| $C_D$ | coefficient of drag |
| $L$ | Lift |
| $D$ | Drag |
| $r$ | radius |
| $\theta$ | longitude |
| $\phi$ | latitude |
| $V$ | velocity |
| $\gamma$ | flight path angle |
| $\psi$ | heading angle |
| $q$ | heating rate |
| $\beta$ | bank angle |
| $\alpha$ | angle of attack |
| $T$ | thrust |
| $\xi$ | basis function coefficients |

TABLE OF CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

## 1.1 Motivation

The growing interest in interplanetary exploration has created many engineering challenges. When sending humans or valuable scientific equipment to the surface of planetary bodies, particularly those with atmospheres, one of the most difficult phases of the voyage is the entry, descent, and landing (EDL) of the vehicle. For planets with low-density atmospheres, such as Mars, the task is made more difficult due to the small aerodynamic forces available for drag and lift. Significant computational resources are needed to simulate the complex processes associated with each entry vehicle configuration in multiple flight regimes. At extreme velocities, many complex interactions occur with the atmospheric environment which can cause issues with different aspects of the system including thermal and structural loads, and dynamics and controls, the latter serving as the focus of this project. It is also necessary to develop an accurate simulation environment in which to design, test, and evaluate entry-vehicle dynamics and controls.

This thesis is a culmination of work on environmental modeling and development of the SpaceCRAFT platform. SpaceCRAFT is a collaborative engineering simulation platform that provides the user with a framework for designing and implementing engineering models. The work in this thesis both progresses the platform capabilities, and develops the Simulation Testbed for Entry Analysis and Design (STEAD). STEAD is a stand-alone simulation tool built on the SpaceCRAFT platform that allows for the testing and analysis of lifting body entry vehicles. The development of this system included the integration and creation of environmental models including, atmospheric, gravity, and rigid-body dynamics propagation. STEAD is designed as a modular system, with the ability for any model to be replaced by a user defined mathematical or 3D model. Of particular interest to this thesis is the development of a Generic Atmospheric Model (GAM) which can be used to accurately simulate the atmospheric properties of any planet, and the implementation of an optimal control system. The project will use STEAD to test an optimal guidance solution to atmospheric entry using a new method of solving the two-point boundary value problem (TPBVP), a least-squares solution using the Theory of Connections (ToC) proposed by Dr. Daniele Mortari.

## 1.2 Literature Review

Previous work in the field of simulation engineering has produced several tools to model entry profiles. Although these tools are well developed, they lack functions that this project addresses including modularity, ease-of-use, pre-integrated high-fidelity models, and virtual reality interaction. Additionally, most of these programs are restricted to government use only, removing a large base of potential users.

In this report, "simulation" refers to a software package that approximates the behavior of a real-world environment or system. NASA's manned spaceflight missions of the 1960's relied heavily on vehicle simulators to provide the crew with an environment in which to practice procedures and operations. Many of these simulators used large mock up facilities to provided the crew with realistic interactable flight hardware and software. As technology has progressed, the fidelity of simulation capabilities has also grown. In the 1970's, NASA, in partnership with Martin Marietta Co, began development on the Program to Optimize Simulated Trajectories (POST) as a space shuttle simulation program. Since POST was restricted to simulating a single body at a time, users who needed to test a multi-stage vehicle were required to run several independent runs and piece the results together. In 1997, Lockheed Martin released POST2 which had the ability to simultaneously test multiple bodies in a high-fidelity six degree of freedom (6-DOF) simulation, and is today an industry-standard tool for the function of trajectory optimization[1,2,3]. The POST2 core software provides the minimum set of algorithms and communication structure to simulate a trajectory, but relies on the user to provide more detailed models such as sensors, flight software, and high fidelity environmental models.

Most software simulation tools are built on a core code base that controls the communication of models and storage of object variables. An existing system that accomplishes these tasks is Trick [4,5]. The Trick Simulation Development toolkit has become a standard simulation environment at NASA at the Johnson Space Center. The now open-source software began development in the

1980's as the need for a more generic simulation environment grew. Trick allows for real-time, human-in-the-loop simulation to support astronaut training and mission architecture design. Most recently, the Trick toolkit has been used to test complex processes aboard the newly-proposed Deep-Space Gateway. These applications include the analysis of power and thermal systems, dynamics and controls, and the effect of various human processes on the overall mission design. The communication framework of Trick serves as inspiration for the structure of SpaceCRAFT, the virtual reality (VR) engineering simulation environment utilized by this thesis.

NASA has also developed high-fidelity Global Reference Atmospheric Models (GRAMs). GRAMs are atmospheric models of various planetary bodies in the solar system including Earth, Mars, and Titan. These models are based on data recorded by past observations and scientific missions, and used to create a system that accepts time, longitude/latitude, and altitude and outputs the atmospheric parameters [6]. These high-fidelity models will be used to verify the capability of the generic atmospheric model developed in this thesis.

EDL methods vary depending on the vehicle, atmospheric characteristics, and mission objective. The two main types of vehicles are capsules and lifting bodies. The Pathfinder capsule spacecraft performed an atmospheric entry at Mars utilizing a variety of technologies to successfully land its payload, including a parachute, rocket motors for powered descent, and airbags for the impact [7]. Depending on the payload, it may be necessary to perform a soft landing. Perhaps the most recognized lifting body entry vehicle is the Space Shuttle. The shuttle guidance system is separated into two parts, the Entry guidance and the Terminal Area Energy Management (TAEM) system. The entry system flies a low lift-to-drag, high angle-of-attack profile to a mach number of 2.5 at an altitude of approximately 85,000 ft. The TAEM system guides the vehicle to a final runway approach using angle-of-attack to control the glide slope, bank angle for lateral track control, and speedbreaks to reduce airspeed [8]. The lifting body vehicle relies on aerodynamic forces for control, which makes entry on low density planets, such as Mars, more difficult.

Currently, most optimal control space guidance problems are solved using direct methods, where the states and controls are discretized and converted into a convex optimization problem to ensure convergence. These solutions have been applied to spacecraft problems including planetary landings [9] and low thrust orbit transfers [10]. Another method of solving optimal control problems is the indirect method, which applies optimal control theory to derive the necessary conditions that must be satisfied by the solution. For nonlinear problems, this results in a set of complex equations that are sensitive to the initial guess, making the problem hard to solve. Indirect methods tend to result in more accurate solutions, but are difficult to implement in a real-world controller. This thesis will also attempt to implement a control system design based off of previous work by Dr. Daniele Mortari and Dr. Roberto Furfaro. Mortari has developed a new method to solve boundary value problems for differential equations called the Theory of Connections. ToC method finds a constrained expression for a function subject to linear constraints on the function and its derivative [11]. An application of ToC to optimal control was presented by Furfaro and Mortari, where a least-squares solution of differential equations using the constraint expressions found from ToC are used to solve basic space guidance problems. Furfaro demonstrated the ability to use a ToC approach to solve two point boundary value problems, and noted a computational time on the order of $10^{-3}$ sec [12]. He concluded that ToC might be able to solve optimal guidance problems in real-time, a statement that will be investigated by this project.

The motivation of this thesis is to create a simulation tool that allows for the testing and analysis of entry vehicles. Although there are existing systems, such as POST2, that are often used by NASA, this tool would be an open-access application that is not restricted for government-use only.

## 1.3 Research Objectives

This thesis is an extension of past work in the development of the SpaceCRAFT platform by the AeroSpace, Technology Research and Operations (ASTRO) Lab [13,14]. SpaceCRAFT is an engineering simulation tool that contains the framework for the handling of variables, communication between mathematical models, and rendering of the simulation in real-time. The platform is structured to reduce the user workload when generate new environments for each simulation, and give a starting point of which to build, test, and analyze systems. SpaceCRAFT uniquely aims to be a scale model of the entire solar system, adapting the existing physics engine, PhysX, to use 64-bit coordinates to maintain numerical precision. The work in this thesis provides one of the first use-cases for SpaceCRAFT, a modular simulation environment for the testing and analysis of entry vehicles and control systems, as well as adding functionality to the overall platform.

STEAD is an end-to-end software application for the testing of lifting body spacecraft during EDL. The software aims to have capabilities equivalent to POST2, but with flexible environmental models and a connection interface that reduces user workload. Although this project focuses on the entry and descent phases, the platform is designed with a human-controlled landing phase in mind. Past work on the SpaceCRAFT platform has incorporated virtual displays and controls into human-in-the-loop simulations that would allow the user to manually land the spacecraft in VR. The research objectives of this thesis are as follows:

- To advance the SpaceCRAFT platform by adding new functionality and providing the first full use case with which to test the system.
- To integrate the NASA GRAMs into a simulation environment, which requires adding the ability to compile FORTRAN code with the core SpaceCRAFT platform.
- To apply the NASA Jet Propulsion Laboratory (JPL) SPICE toolkit, allowing SpaceCRAFT to accurately represent the states of all planetary bodies and provide functions for reference frame transformations.
- To investigate whether it is possible to use a ToC formulation to solve for the optimal guidance solution continuously during a real-time simulation.
- To develop a computer program that models the dynamics of an entry vehicle using the rotational and translational equations of motion integrated with a Runge-Kutta fourth-order method.
- To verify the GAM by comparing the atmospheric parameters such as temperature, density, and pressure against the NASA GRAM model for Mars and Titan.

- To perform test cases of a lifting body entry vehicle on different planetary bodies with STEAD to demonstrate the capabilities of the guidance system and the usability of the tool.

## 1.4 Contributions

- The development of a modular simulation tool for the testing and analysis of atmospheric entry vehicles, including physical vehicle characteristics and control system designs.
- The development of an atmospheric model, GAM, that provides atmospheric parameters of any planet based on ephemeris data and basic atmospheric characteristics.
- Implementation of a real-time optimal guidance solution using a ToC formulation applied to an entry TPBVP.

## 1.5 Organization of Thesis

Section 2 of this thesis discusses the framework of the SpaceCRAFT platform. This includes information on the structure of the core code where the data handling and model communication occurs. The current capabilities of the platform, as well as features currently in development will be introduced in terms of how they apply to this thesis. The sections following will identify the relevant models utilized in the entry simulation tool and discuss their derivation and implementation into the platform.

Section 3 is dedicated to the application of the SPICE toolkit from NASA JPL to the SpaceCRAFT platform, giving the system the ability to accurately place the planetary bodies of the solar system. The addition of the SPICE toolkit to the platform also provides functionality, such as coordinate frame transformations, sensor measurements, and ray-tracing methods.

Section 4 provides information regarding the atmospheric modeling. For this thesis, two atmospheric models are used: an in-house developed GAM and NASA GRAM for Mars. The generic model is compared to the GRAM model, providing a verification and validation test case.

Section 5 shows the derivation and implementation of the dynamics and guidance system of an entry vehicle. The dynamics propagation aspect of the simulation is applicable for all lifting body vehicles, but the guidance solution is specific to the example vehicle used to demonstrate the capabilities of the tool in the test cases. The dynamics of the vehicle are calculated by stepping through the translational and rotational equations of motion. The ToC methodology is explained and the algorithm for solving the optimal guidance problem is developed.

Section 6 discusses two test cases for the simulation tool: a lifting body entry on Mars and Titan. Each test case will provide an opportunity to validate the GAM by comparing the resulting atmospheric parameters to the NASA GRAMs. Since Mars and Titan have drastically different atmospheres, the test cases will show the performance of the optimal control system designed in Section 5. The testing of the ToC guidance algorithm showcases an example of how STEAD would be used as an analysis tool during the development of entry vehicles.

The thesis will conclude with an analysis of the test case results and a discussion on the capabilities of STEAD, and by extension SpaceCRAFT, to serve as an important tool in the development of engineering systems. Possible pathways for future work on the project will also be discussed including implementation of a landing sequence using VR controls and interfaces.

# 2. PLATFORM DEVELOPMENT

STEAD is built on the SpaceCRAFT platform currently in development by the ASTRO Lab at Texas A&M University []. SpaceCRAFT was proposed by former astronaut and ASTRO Center Director, Dr. Greg Chamitoff. His vision was to create an open-source virtual reality engineering simulation environment that could serve as the next major tool for developing space mission systems and architectures. With the platform available to the entire world, it would be possible for users to integrate their designs in an accurately modeled environment to analyze a fully integrated system and ensure compatibility between models.

Development of SpaceCRAFT began in 2016 with a small group of graduate students and undergraduate students in a Space System Design course in the Aerospace Engineering department at Texas A&M University. Teams of students were tasked with creating mission scenarios in Unreal Engine 4 (UE4), the main external software utilized by the platform. From this course, the backend of SpaceCRAFT branched off of UE4 to form the serverclient model that is currently in use. The ASTRO Lab currently coordinates a yearly undergraduate research course, AggiE_Challenge, that allows students from all engineering disciplines to contribute to the continued progression of the SpaceCRAFT platform, including environmental modeling, 3-D modeling, machine learning, and website development. To date, the SpaceCRAFT platform has been utilized for various projects including:

- Mars mission architecture senior design project (2017)
- NASA Big Idea Challenge solar panel lander (2017)
- Mars VR Desert Research Station for astronaut training (2018)
- Lunar surface operations mission design (2018)
- Artificial gravity quadrotor masters thesis (2018)
- Trappist-1d machine learning and AI robot programming challenge at SXSW (2019)

- NASA SUITS competition - AR head-up display for EVAs

One of the outcomes of this thesis was the advancement of various functions and models used by the core system of the platform. The architecture of the platform and its utilization for this thesis will be discussed in this section.

## 2.1 Platform Structure

The SpaceCRAFT platform consists of the core computer program that handles the data storage and communication between mathematical models and the distribution to the graphics engine. Every component of a SpaceCRAFT simulation can be classified into two categories: entities and systems. A "System" is a mathematical model that provides information to the objects being simulated, for example, an atmospheric model that provides an airplane with air density. An "Entity" is any object simulated in the virtual environment. The most common Entities in SpaceCRAFT are objects such as planets, spacecraft, robots, humans, and user-interfaces.

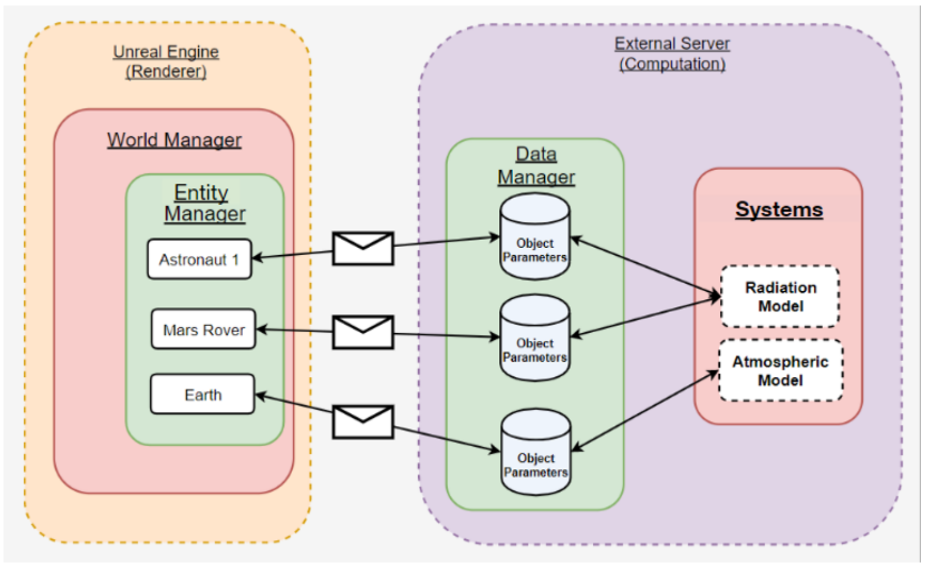


Figure 2.1: SpaceCRAFT platform communication overview.

As shown in Figure 2.1, the platform consists of two main sections: an external computational server run in a Linux environment and Unreal Engine 4 which is used as the rendering software.

The server holds the System Manager and Data Manager, which handle the simulation process. The System Manager controls the execution of the simulation Systems and sends parameter updates to the necessary Entities. The Data Manager is in constant communication with the UE4 Entity Manager to maintain the correct Entity parameters between the two platform sections. Each instance of a System is, by default, running in parallel on its own thread. Modern CPU development has shifted focus from increasing single thread performance to increasing thread count and multi-threading efficiency. SpaceCRAFT's as-multi-threaded-as-possible approach to processing lets the platform take full advantage of additional threads for performance. Interactions between the Systems and Entities are done by the *GetParameter* and *SetParameter* commands. To avoid rare conditions where parameters are accessed while still changing value from a set command, mutual exclusion locks are used to ensure only one operation is occurring at once. This protocal is in place for System commands as well as the client-server synchronization commands.

Although UE4 is mainly used as a rendering tool, the software also includes PhysX, a physics library that can be utilized to handle complex interactions, particularly collisions. A major issue with PhysX is its use of 32-bit precision floating-point numbers. Normally, this level of precision would be acceptable, but when simulating objects in the solar system at true-scale, the errors become significant. To combat this limitation, the SpaceCRAFT team created a coordinate-rebasing system called Overview. The Overview system, visualized in Figure 2.2, creates an independent "Overview Box", a 12 km cube in which PhysX solutions are valid, around relevant entities. As the entity moves outside of this region, the cube absorbs the velocity and position of the Entity and relocates the origin of the simulation to the current position. This allows the PhysX system to always be valid while maintaining 64-bit precision of the location and velocity of the entity at true-scale.

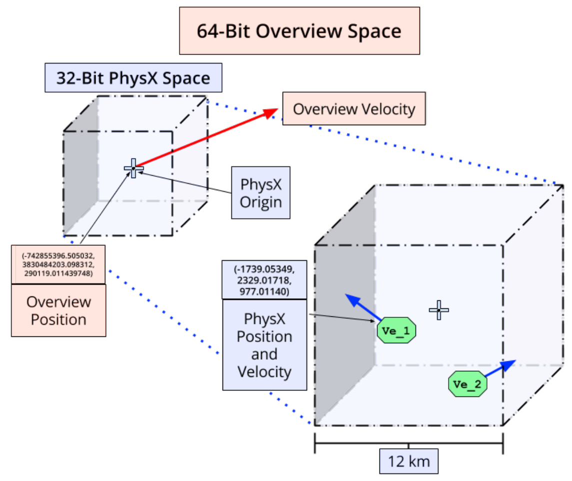


Figure 2.2: Diagram of Overview coordinate-rebasing system.

## 2.2 3rd Party Software

A key characteristic of the SpaceCRAFT platform is the ability to link 3rd party software through custom application programming interfaces (APIs) or other network protocols. Prior to this thesis, the team had created and implemented an API to allow the passing of variables and commands between Python scripts and the server using a transmission control protocol (TCP). TCP allows two nodes, in this case the source, Python, and the destination, SpaceCRAFT server, to connect and pass streams of data between them. A key benefit of using TCP versus other protocol types is the guarantee that the data is delivered and each packet is received in the order it was sent. This is useful for the simulation platform because it ensures that the values passed to the server and Unreal Engine are consistent with time.

The major 3rd party software used in this thesis is MATLAB, which was chosen due to its abundance of control system toolboxes that will be used in the optimal control design to provide control inputs to the vehicle. Following the model of the Python API, a TCP connection will allow the vehicle control System on the server to call functions from a continuously running MATLAB script. The server-client connection is established to allow for commands and data to pass between C++ and MATLAB using a request-reply architecture. The C++ client, in this case the dynamics and control system, requests the MATLAB server with a query string. The MATLAB server responds by sending a string on the same port. It is important to note that although the MATLAB server must be running prior to receiving requests from the client C++ program, the C++ program can terminate and re-connect as many times as necessary during the course of server execution. The use of MATLAB also lends the ability to visualize the simulation data in a graphical manner during a run. Since the vehicle controls will be generated in real-time, it is necessary to ensure that the speed of the TCP connection is adequate. To test this, a script for propagating the six equations of motion for an entry vehicle was called by a C++ system 10,000 times.

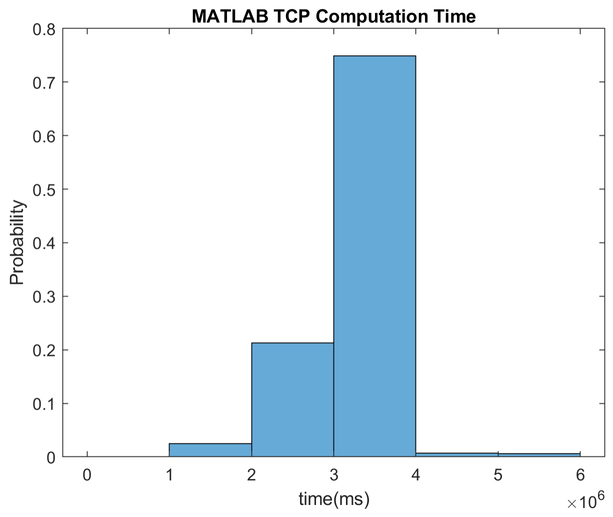


Figure 2.3: Histogram of time required to send a command from a C++ system and receive result from an external MATLAB server.

The histogram in Figure 2.3 shows that the return times are fairly consistent, but there are some random spikes in the computation time that could cause a backup in the TCP port if the System was called at a high enough frequency. To combat this, the frequency of the System is kept low enough to ensure calculations are completed before sending the next request.

## 2.3 Simulation Setup

For the STEAD simulation, the user does not need to understand the intricacies of the SpaceCRAFT platform. This thesis sets up the simulation in a way that creates a "plug-and-play" type interaction, where the user can directly replace components of the simulation with custom models or systems and adjust variables related to the basic setup of the environment. The overall structure of the simulation is easily visualized as a puzzle diagram, Figure 2.4, where each part can be replaced or modified to fit the needs of the user.

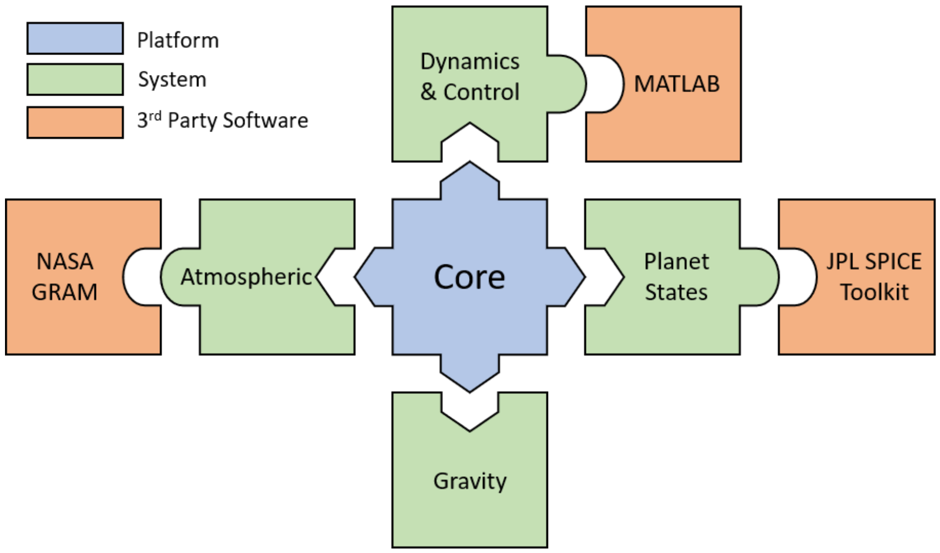
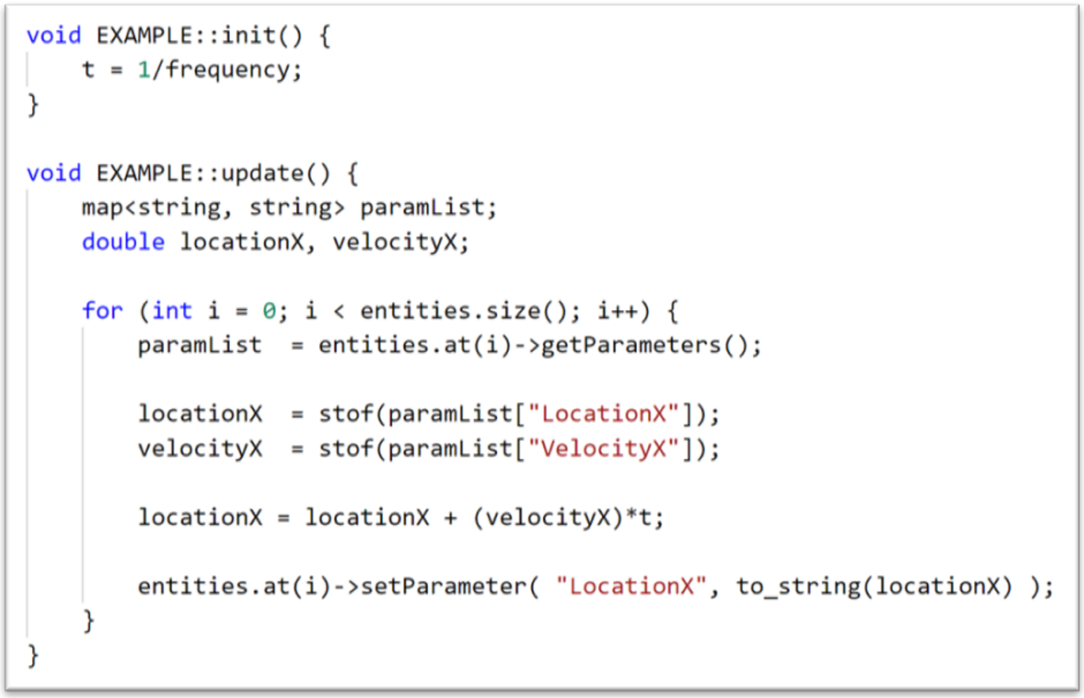


Figure 2.4: Representation of the system blocks needed for the simulation tool and how they connect to the core system.

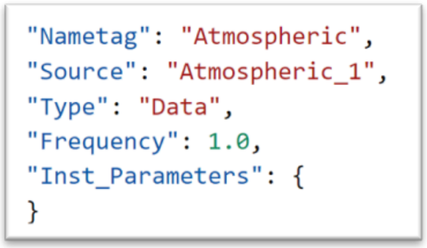
As with most platforms, there are some functions required in the code of each system to allow it to communicate with the core system. The *init()* function is an initializer that executes at a specific time before the start of the simulation. The initializer is called simultaneously from all Systems by the core system directly before the first timestep, which allows the initializer to have knowledge of all other Entities and Systems in the simulation. The *update()* function is the operation called by the core system at each iteration of the specific System. It is within this function that the System updates the parameters of each entity. A segment of the System C++ code is shown in Figure 2.5.



```cpp
void EXAMPLE::init() {
    t = 1/frequency;
}

void EXAMPLE::update() {
    map<string, string> paramList;
    double locationX, velocityX;

    for (int i = 0; i < entities.size(); i++) {
        paramList  = entities.at(i)->getParameters();

        locationX  = stof(paramList["LocationX"]);
        velocityX  = stof(paramList["VelocityX"]);

        locationX = locationX + (velocityX)*t;

        entities.at(i)->setParameter( "LocationX", to_string(locationX) );
    }
}
```

Figure 2.5: Screenshot of *init()* and *update()* functions necessary in every system.

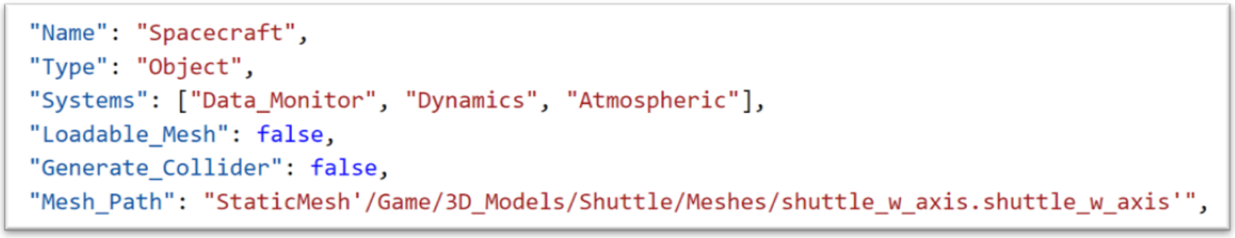
The frequency at which the System's update function is called depends on the user set value in the configuration file. From the viewpoint of resource utilization, it is advantageous to set lower frequencies on systems that do not necessarily vary on a given timescale, allowing the user to allocate computational power to other models. For example, if a user is testing a rover on the surface of Mars, the System that updates the temperature of the surface may be set to update 10 times a second, while the dynamics of the vehicle are updated 1000 times a second. The declaration of the atmospheric model is shown in Figure 2.6, with a frequency of one update per second.



```json
"Nametag": "Atmospheric",
"Source": "Atmospheric_1",
"Type": "Data",
"Frequency": 1.0,
"Inst_Parameters": {
}
```

Figure 2.6: The declaration of the system in the configuration file.

Although a graphic user interface is currently in development, at the time of this thesis the user is still required to modify the configuration file in a text editor to make changes to the simulation. Once the user has made the necessary adjustments to connect their model to the platform, the configuration file, given in Figure 2.7, is edited to use the correct vehicle mesh and Systems. The *Mesh_Path* command accepts the identifier and mesh path of the 3-D vehicle model while adding System tags to the *Systems* line to specify which Systems affect the vehicle.



```
"Name": "Spacecraft",
"Type": "Object",
"Systems": ["Data_Monitor", "Dynamics", "Atmospheric"],
"Loadable_Mesh": false,
"Generate_Collider": false,
"Mesh_Path": "StaticMesh'/Game/3D_Models/Shuttle/Meshes/shuttle_w_axis.shuttle_w_axis'",
```

Figure 2.7: Initial declaration of the vehicle entity

## 2.4 Future Software Updates

The SpaceCRAFT platform is constantly adding new features and improving capabilities. The team is currently working toward a set of universal propagation types utilizing the PhysX engine, which would reduce the user's workload when modeling object dynamics. Additionally, the newest release of Unreal Engine 4.22 contains a plugin for the visualization of atmospheres. This is of particular interest for this project due to the difficulty associated with accurately modeling the spectral properties of the atmosphere during an entry simulation. Since STEAD is built on top of SpaceCRAFT, any new updates produced by the SpaceCRAFT team in the future will be directly applied to the simulation tool with minimal effort.

## 2.5 STEAD Outline

STEAD uses multiple Systems to simulate the trajectory and control inputs of an entry vehicle, as shown in Figure 1.4. The SPICE system is used to produce the states of the target planet and provide inputs into the atmospheric model, which provides the atmospheric characteristics used to perform the aerodynamic calculations. STEAD uses these inputs to propagate the dynamics of the entry vehicle. To provide vehicle guidance, the MATLAB control system receives state vectors from dynamics System, and returns the guidance commands, such as thrust and bank angle. Each of these systems will be discussed in the following chapters.

# 3. PLANETARY STATES

The ability to have accurate locations and rotations of the planetary bodies is an important aspect of the simulation tool. To accomplish this, the project implements the NASA JPL SPICE toolkit, an information system designed to assist scientists and engineers with mission design and post-mission analysis of instrument data.

In 1982 The National Research Council's Committee on Data Management and Computation issued a report that detailed the need for a system to properly collect and archive the information needed to correctly interpret data from space science instruments [15]. As a result, NASA hosted a Planetary Data Workshop that resulted in the early SPICE concept, a method of archiving the fundamental data sets needed to back-trace the geometric parameters of the space instruments in order to understand the acquired data. The first proof of concept of the tool was performed with the instrument teams of the Voyager spacecraft, which many other teams utilizing the tool since, including the Magellan, Mars Observer, and Galileo projects. Although the tool was initially designed to assist scientists with data analysis, the space exploration community has made use of the tools for mission design and operations.

## 3.1 SPICE toolkit

The SPICE toolkit is designed to compute observational geometric parameters, such as states of bodies, and geometric events, such as occlusion. The ability to produce the position, velocity, and rotation of planetary bodies is of great benefit to STEAD. A system called *Spicy* was created to integrate the SPICE toolkit into the SpaceCRAFT core system. Inputs into the system include a target, observer, and reference frame. The target is the body of interest, the observer is the location or body that serves as the location of the "origin" of the simulation, and the reference frame defines the coordinate system with which the state exists.

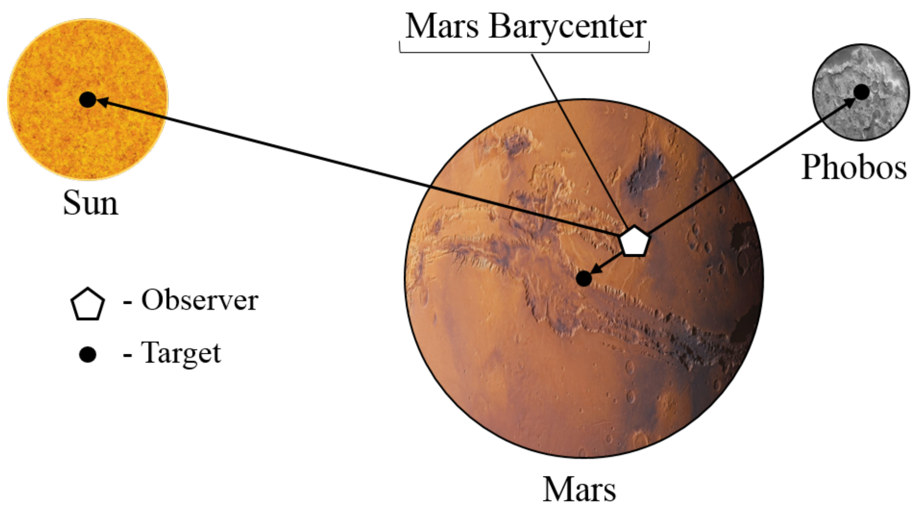


Figure 3.1: Illustration of a SPICE representation for the Mars system.

The basis of the simulation environment is with respect to the J2000 reference frame, an inertial frame based on the Earth's equator and equinox. From Figure 3.1, a basic STEAD setup includes the planet to be visited and the Sun, as targets, and the barycenter of the planetary system, as the observer, in the J2000 frame.

## 3.2 Ephemeris Data

In order to provide accurate planetary states the SPICE toolkit relies on data stored in binary kernel files. The DE430 kernel created by JPL's Solar System Dynamics Group contains the ephemeris data for each planetary body from Mercury to Pluto plus the Sun and the Moon. The orbits of the inner planets, Jupiter, and Saturn are determined by fitting spacecraft tracking data, whereas the orbits of Uranus, Neptune, and Pluto are found through a less accurate astrometric observation method. The planetary positions and velocities are found by numerically integrating dynamic models, and are stored as Chebyshev polynomial coefficients fit into 32-day segments. Although the binary files store information at discrete epochs, due to the effective modeling and storage of the ephemeris data, the position and velocity of planetary objects are seen as continuous by the SPICE toolkit. Additional kernels are also available to describe the positions of planetary satellites, such as Saturn's moon Titan.

## 3.3 Validation of the SPICE System

In order to validate the SPICE system, a simulation of the Sun, Earth, and Moon was created to check for astrological events that would show that the planetary states are accurate. For this test, the simulation time was set for the total solar eclipse on August 21, 2017 (18:14:00 UTC). The plot in Figure 3.2 shows the dot product of the vectors from the Sun to the Earth and the Sun to the Moon, where two parallel vectors would result in a value of one. A screenshot of the simulation is included from an observer standing in Missouri, United States. It is expected that during an eclipse, the two vectors should be parallel.

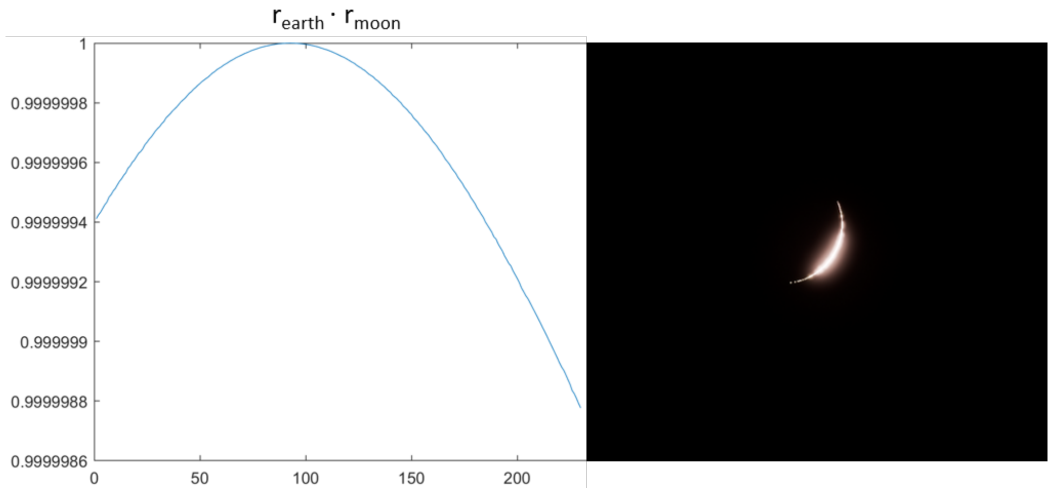


Figure 3.2: Plot of the dot product of a Sun-Earth and Sun-Moon vector during a solar eclipse (left). A screenshot of the simulated eclipse from a location on Earth's surface (right).

# 4. ATMOSPHERIC MODELING

For entry simulations, the atmospheric model is of high importance. Driven by the need to test future space exploration missions, NASA has developed atmospheric models for various planetary bodies in the Solar System. These models fit a set of functions to data obtained from observations and previous missions. This produces a good approximation of the atmospheric properties with a small computation time, since the model is essentially a set of functions with a table of coefficients.

Since the SPICE toolkit provides the simulation environment with accurate planet states at scale, it is proposed that a generic atmospheric model with the ability to represent any planet's atmospheric properties could be developed. By using the energy received from the Sun and atmospheric composition of the planet, the natural processes of the system are simulated. Although this model will be more computationally intensive, the result is a high fidelity model based on physical processes.

## 4.1 Mars Global Reference Atmospheric Model

Mars-GRAM utilizes input data tables from the NASA Ames Mars General Circulation Model and the University of Michigan Mars Thermospheric General Circulation Model to give the variation of atmospheric parameters with height, latitude, and time of day, as well as provide boundary layer data at the topographic surface of the planet. To calculate each atmospheric parameter, the data tables provide a daily mean value and the amplitude and phases of the diurnal and semi-diurnal tidal components to the following equation.

$$Parameter = A_0 + A_1 cos[\frac{\pi(t-\phi_1)}{12}] + A_2 cos[\frac{\pi(t-\phi_2)}{6}] \tag{4.1}$$

These values are provided at height increments of 5 km up to a maximum of 170 km from the reference ellipsoid. Therefore, to access the data for a certain height $z$, the model interpolates between two points just above and below. Temperature and wind components are found from a linear interpolation between the two points, but pressure is found by first computing the pressure scale height, the vertical distance over which pressure and density fall by a factor of 1/e,

$$H = \frac{(z_2 - z_1)}{ln[\frac{p(z_1)}{p(z_2)}]} \tag{4.2}$$

and evaluating pressure from the hydrostatic relationship:

$$p(z) = p(z_1)e^{\frac{z_1 - z}{H}} \tag{4.3}$$

The gas law ’constant’ (R) is linearly interpolated from the following calculation at each point.

$$R(z_i) = \frac{p(z_i)}{\rho(z_i)T(z_i)} \tag{4.4}$$

Finally, density at the desired height is found using the gas law relation:

$$\rho(z) = \frac{p(z)}{R(z)T(z)} \tag{4.5}$$

Integrating the NASA GRAM model into the platform required creating a System, written in C++, to interface with the FORTRAN source code. This System now gives the SpaceCRAFT platform the capability to utilize FORTRAN models, which are heavily used by the scientific community. A simple test of the Mars GRAM was performed to ensure the System was functioning correctly by simulating the model values at 1 km intervals and comparing to results to data recorded by the Viking 1 and Pathfinder missions, shown in Figure 4.1. These runs showed that the density and temperature values generated by Mars-GRAM are comparable to the measured values from past Mars missions. The density more closely follows the expected data than the temperature. The difference in the temperature range for Pathfinder and Viking from 50 km to 90 km is reportedly caused by the condensation of $CO_2$ on the vehicle, reducing the temperature. Additionally, the temperature is dependent on the time of day and year in which the data was recorded, in this case the GRAM model was simulated when the atmosphere from 0 km to 50 km was occluded from the

Sun by Mars itself.

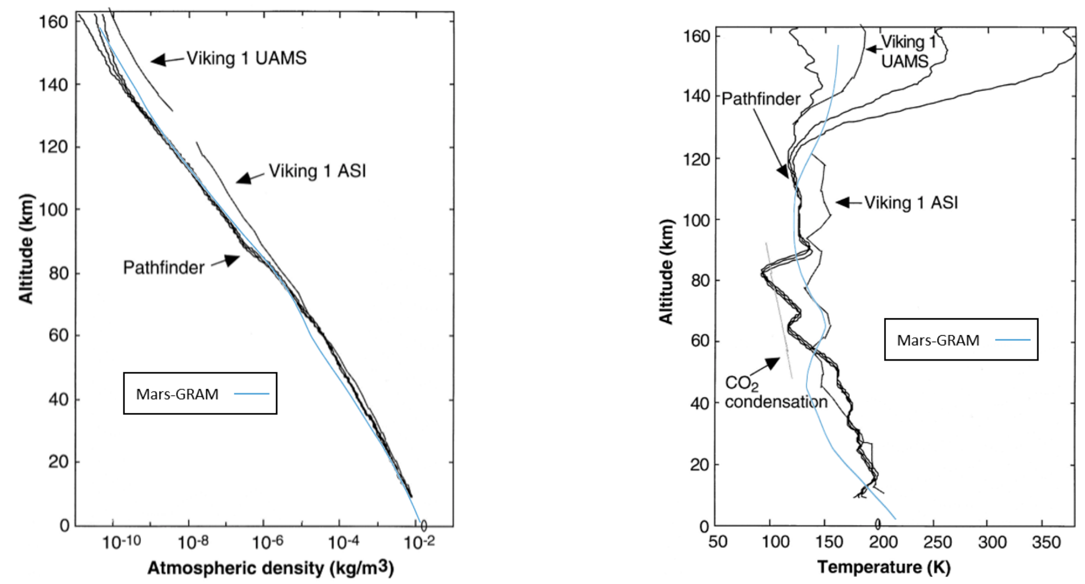


Figure 4.1: Output of Mars-GRAM 2010 with measured values of density and temperature from the Viking 1 and Pathfinder Mars missions [4].

To verify the Mars GRAM system was dynamically changing with time, the temperature time history at $0^o$ latitude/$0^o$ longitude was simulated at height intervals of 10 km for 24 hours at 30 minute intervals.

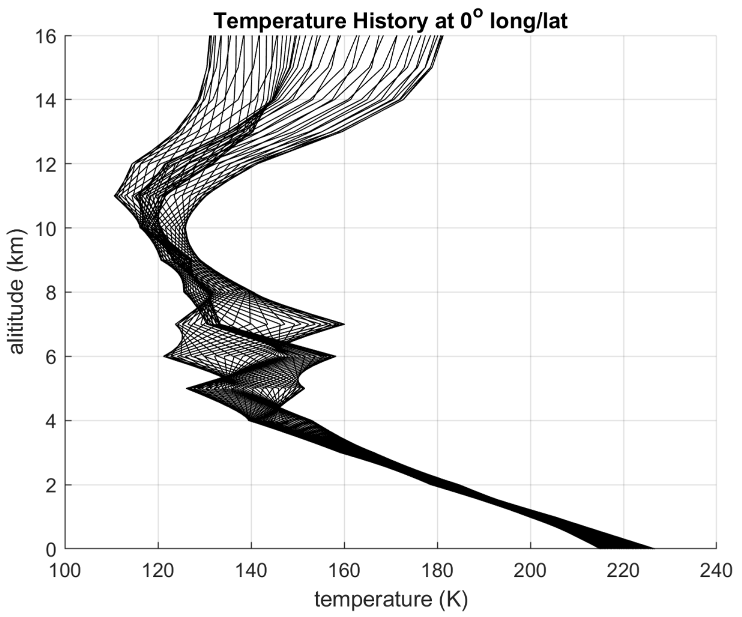


Figure 4.2: Temperature output of Mars-GRAM at $0^o$ latitude/$0^o$ longitude.

As seen in Figure 4.2, the temperature is close to linear at the boundary from 0 km to 40 km, with the sinusoidal function portion of Eq. 4.1 dominating the outer atmosphere.

The integration of the Mars GRAM provides the simulation with accurate values for the atmospheric parameters necessary to calculate the aerodynamic forces on the vehicle. There are additional settings in the configuration of the model that allow for user defined perturbations in both the atmospheric parameters and wind speeds. This input will give a disturbance to the simulation, requiring the guidance and control algorithms to compensate. Although one goal of this thesis is to develop a generic model that applies to any planet, the ability to incorporate existing high-fidelity models is necessary for the simulation tool to be flexible to the user.

## 4.2 Generic Atmospheric Model

One of the goals of this thesis was to develop a single atmospheric model that could simulate the atmosphere of any planet. To accomplish this, the GAM aims to simulate the physical processes that drive the dynamic atmospheric environment. The user inputs to this system include commonly known characteristics of the planet and its atmosphere, such as atmospheric composition, scale height, planet radius, and surface albedo. Additionally, the SPICE System provides the location and rotation of the planet relative to the Sun, which is the main actor in determining the model behavior. The energy from the Sun is used to calculate the temperature change, which is then related to pressure and density using Eqs. 4.3 and 4.5, with the gas constant assumed as constant. The model outputs the atmospheric conditions, temperature, density, and pressure, at the vehicle's current location.

### 4.2.1 Modeling Method

The GAM simulates the surface and atmospheric conditions at specific points called nodes. These nodes are generated at run-time with a user defined grid density. Each node is evenly spaced on a longitude and latitude line, forming a grid over the surface as shown in Figure 4.3. Due to the spherical nature of a planet, there is a higher concentration of nodes at the poles. Although this is sub-optimal for performance, it was determined that the simplicity of keeping the nodes in a row/column structure would make the interpolation between nodes easier. From each surface node, a new set of atmospheric nodes are placed radially at specified intervals, giving the model coverage of the entire atmosphere.

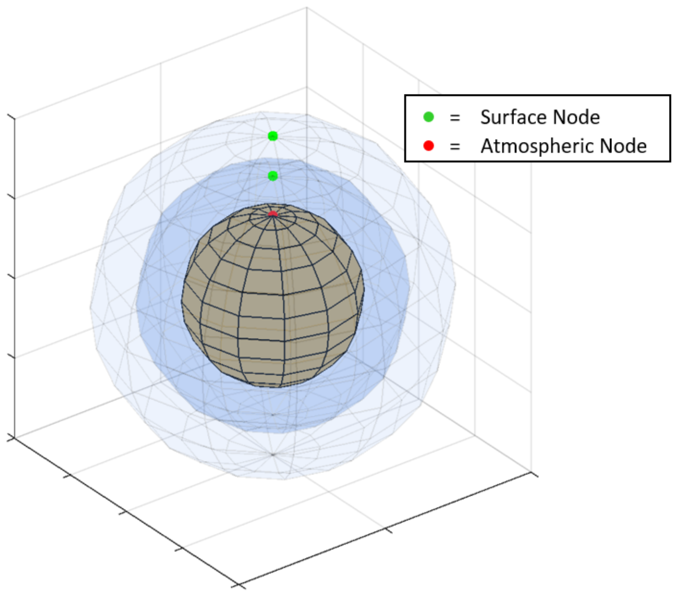


Figure 4.3: Visualization of GAM node distribution

Each node structure stores its atmospheric conditions, physical composition, and location in a body-fixed frame. Including the physical composition into each node allows the model to account for variations in the chemical makeup of the atmosphere due to altitude.

The physical processes modeled in the GAM include radiation, transmittance, absorption, and reflection. These processes occur sequentially through the vertical nodes above each surface node, illustrated in Figure 4.4. The Sun is treated as a blackbody with an intensity of

$$L = \frac{\sigma}{\pi} T^4 \tag{4.6}$$

where $\sigma$ is the Stefan$-$Boltzmann constant and $T$ is the temperature of the Sun. Ignoring the atmospheric nodes, a ray is traced from each surface node to the center of the Sun. This ray provides the distance from the node to the Sun ($r$) and the incidence angle ($\theta$) measured from the surface normal vector. The incident flux received by the surface node is calculated from the inverse-square law multiplied by the cosine of the incident angle.

$$\Phi_0 = L \left( \frac{r_{Sun}}{r} \right)^2 cos\theta \tag{4.7}$$

This flux value is now used as the energy source entering the atmosphere at the top most level.

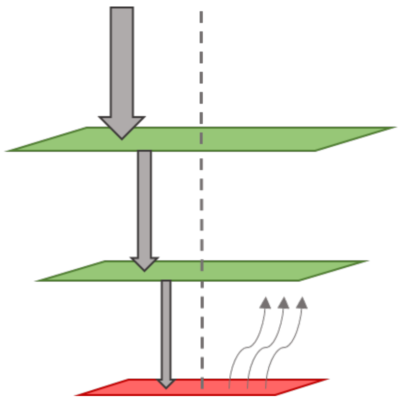


Figure 4.4: Direction of energy flow in a radial set of nodes.

As this energy passes through the atmospheric nodes, the transmittance of the node is calculated to find the amount of energy absorbed by each node.

$$T_r = e^{\frac{-sec\theta\rho_0 z}{\bar{m}(2-\frac{\Delta T}{T})}(\sqrt{2}\sigma_m)} \tag{4.8}$$

As seen in Eq. 4.8, the transmittance is dependent on a number of factors, including the wavelength ($\lambda$) of the flux. Instead of integrating over a spectral density equation, a simple Riemann sum was used to approximate the distribution. Since scattered light was assumed to be negligible, it can be inferred that any energy not transmitted was absorbed by the surface, resulting in an increase in temperature.

Once the energy reaches the surface node, there are two possibilities: reflection or absorption. The amount of energy reflected from the surface is dependent on the albedo ($\alpha_{surf}$), which varies depending on the material.

$$\Phi_{reflected} = \alpha_{surf}\Phi \tag{4.9}$$

For this model, the average bond albedo of the surface was used. This assumption will cause inaccuracies in the model where regions differ greatly from the average albedo, such as oceans and icy landscapes. The energy not reflected is absorbed by the surface node, increasing the temperature with the relationship shown in Eq. 4.10.

$$\begin{gathered}\Phi_{absorbed} = (1-\alpha_{surf})\Phi \\ \Delta T = \frac{mC_p}{\Phi_{absorbed}}\end{gathered} \tag{4.10}$$

During the simulation, the surface is constantly radiating its stored energy. This radiation process provides an energy input into the node system in the direction normal to the surface.

### 4.2.2 Preliminary Results

Modeling the atmosphere of a planet is a non-trivial task, which is made more challenging by the attempt to create a model that will work for any planet. Using only physical parameters and constants, the GAM was able to accurately simulate the temperature on the surface of Mars. To start the simulation, the model uses Eq. 4.12 to set the temperature of each node to an approximate mean temperature.

$$T \approx T_{sun}(1-\alpha_{surf})^{\frac{1}{4}}\left(\frac{r_{sun}}{2r}\right)^{\frac{1}{2}} \tag{4.11}$$

The model performs a run-up simulation for approximately six months before the designated simulation start time, allowing the model to naturally reach a state of equilibrium.

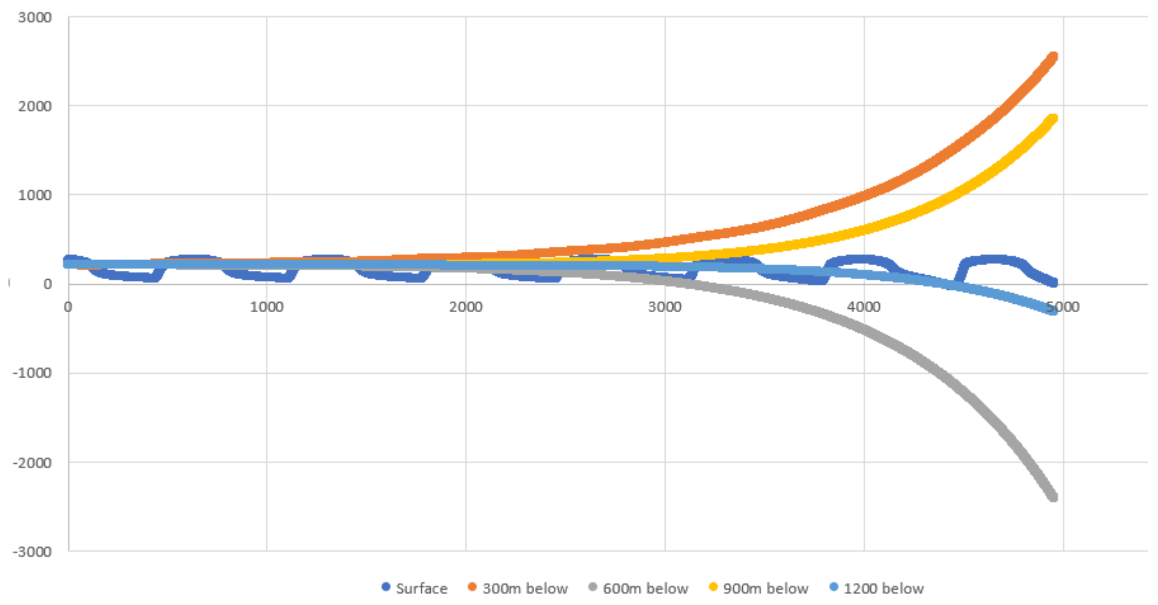


Figure 4.5: Mars temperature output from preliminary GAM

As seen in Figure 4.5, the surface temperature holds a stable solution, but the atmospheric node temperatures diverge exponentially. This is likely due to the use of the same absorption process for the atmospheric nodes as the surface nodes. The main limitation with the model is the inability to model the process of convection, which is the dominant drive in the motion of temperature through atmospheric nodes. From the accuracy of the surface temperatures, it is plausible to say that the transmittance portion, and by extension the absorption, of the atmospheric nodes is correct, but the transfer of energy from one atmospheric node to another is erroneous. Part of this issue is due to the restriction of energy flow to the radial direction, when true air flow is three-dimensional.

Although the model does not perfectly simulate the entire atmosphere, it is still possible to make use of the parts of the model that are accurate, and form relationships to obtain realistic results. Since the surface temperatures are accurately modeled, the relationship between surface temperature and atmospheric temperature was investigated. By comparing the surface temperatures of the Mars GRAM to its atmospheric results, the temperature of the region between the surface and 50 km (five times the atmospheric scale height) follows the relationship

$$T = T_{surf} e^{-kh} \tag{4.12}$$

where $k$ is a function of the scale height, and the region from 50 km and above follows

$$T_{50km<h} = T_{50km} - cos\theta(h-50) - 20sin\Big(\frac{k}{h}(x-50)\Big) \tag{4.13}$$

Using the GAM surface temperature output with these relationships produces accurate data for the lower lever of the atmosphere as shown in Figure 4.6.

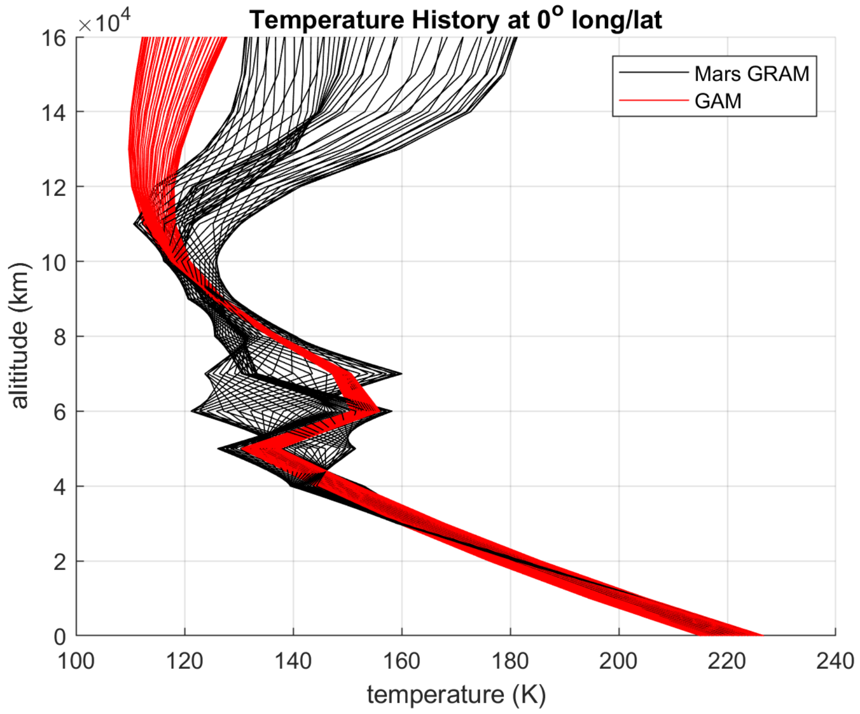


Figure 4.6: Comparison of Mars GRAM with the GAM

Although there is room for improvement, the development of the GAM node structure and the processes for transmittance, reflection, surface absorption, and radiation has created an atmospheric model that can give accurate low altitude atmospheric parameters, with a simple relationship to describe the upper atmosphere.

# 5. DYNAMICS AND CONTROL

With the accurate environment set up in the previous sections, the last component of the simulation tool is the dynamics modeling and control system design. Propagation of rigid body dynamics of lifting bodies is a well studied topic with a common method of solution. The application of the rigid body equations of motion and the representation of the system in the simulation tool is discussed throughout this section. Additionally, the section will present a new solution to the entry optimal guidance problem. The guidance system design is based on previous work by Dr. Daniele Mortari on a newly developed method to solve boundary value problems for differential equations called the Theory of Connections. An application of ToC to optimal control was presented by Roberto Furfaro and Mortari, where a least-squares solution of differential equations using the constraint expressions found from ToC are used to solve basic space guidance problems. This methodology will be expanded to the non-linear entry guidance problem which will provide input to the inner control loop of the system.

## 5.1 Vehicle Dynamics

Modeling the dynamics of the entry vehicle is a challenging task that requires the simultaneous calculation of the body states. The states of a rigid body include its velocity and rotational rates in the body frame as well as its current orientation and location. The orientation of the body is expressed in quaternions, although the conversion from quaternion to Euler angles and Euler angles to quaternion are needed to provide the correct inputs for the equations of motion. A quaternion is a mathematical notation for representing the orientation in a way that avoids the singularity issues present in rotation matrices. The use of quaternions is more numerically stable and more computationally efficient, particularly when used to transform the position, velocity, and orientation from the body frame into the inertial frame when rendering the simulation.

The modeling of a rigid body in six degrees of freedom requires the propagation of the six equations of motion and four quaternion differential equations. A Fourth Order-Runge Kutta method is used to calculate the propagation of these differential equations. The rigid body equations of motion are derived in the body frame, but the SpaceCRAFT platform requires the states to be represented in the inertial frame, thus requiring the definition of an intermediate frame and the relationships between the body, intermediate, and inertial frames. For this tool, the user will provide look-up tables for the aerodynamic derivatives of the vehicle. These derivatives, along with the vehicle state and control inputs, allow for the calculation of the external forces acting on the system. The dynamics propagation of a vehicle in STEAD is outlined in Figure 5.1.

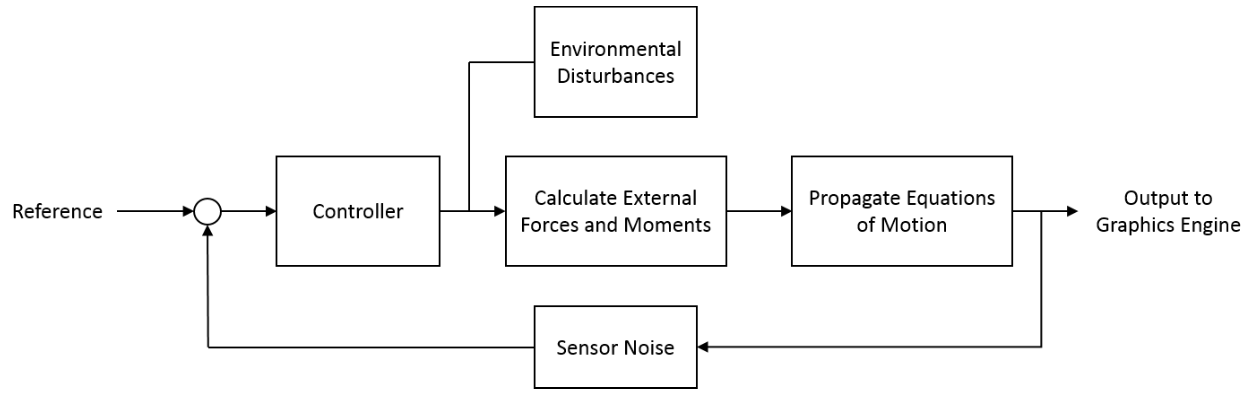

Environmental Disturbances
Reference
Controller
Calculate External Forces and Moments
Propagate Equations of Motion
Output to Graphics Engine
Sensor Noise

Figure 5.1: Block diagram of dynamic and control system implementation.

### 5.1.1 Establishment of Coordinate System

When calculating the aerodynamic forces on a body, it is necessary to define a set of reference frames by which to relate the internal and external forces and moments. The simulation environment contains an inertial reference frame at the center of the planetary body and a body-fixed frame defined with the $\hat{z}$-axis along the spin axis, and $\hat{x}$-axis along the intersection of zero longitude and the equator. To maintain a straight-forward relationship between the dynamics of the vehicle in the body frame to the inertial frame a Local Vertical, Local Horizontal (LVLH) frame is used as an intermediate frame.

The LVLH frame is related to the inertial frame through the current location and velocity of the vehicle. The $\hat{z}$-axis is aligned in the opposite direction of the location vector, giving a constant nadir pointing unit vector.

$$\hat{z} = -\frac{\vec{r}}{\|\vec{r}\|} \tag{5.1}$$

The forward x̂ is the projection of the inertial velocity vector onto the plane orthogonal to the LVLH ẑ-axis.

$$\hat{x} = proj_{Plane}(\vec{v_I}) = \vec{v_I} - \frac{\vec{v_I} \cdot \hat{z}}{\|\hat{z}\|^2} \tag{5.2}$$

The ŷ-axis is the orthogonal vector found through the cross product of the ẑ and x̂ axes.

$$\hat{y} = \hat{z} \times \hat{x} \tag{5.3}$$

These axes can be used to generate a rotation matrix to transform a vector from the inertial frame to the LVLH frame.

$$C = \begin{bmatrix} \hat{x}^T \\ \hat{y}^T \\ \hat{z}^T \end{bmatrix} \tag{5.4}$$

The rotation matrix is converted to a quaternion using Shepperd's Algorithm. First, each of the follow equations are computed, where $\zeta$ equals the trace of $[C]$.

$$\beta_0^2 = (1+\zeta)/4 \quad ; \quad \beta_k^2 = (1 + 2C_{kk} - \zeta)/4, k = 1, 2, 3 \tag{5.5}$$

The positive square root of max$(\beta_0^2, \beta_1^2, \beta_2^2, \beta_3^2)$ is retained and the other three parameters are recomputed by dividing the appropriate three of the following equations.

$$\begin{aligned} \beta_0\beta_1 &= (C_{23} - C_{32})/4 \quad ; \quad \beta_2\beta_3 = (C_{23} + C_{32})/4 \\ \beta_0\beta_2 &= (C_{31} - C_{13})/4 \quad ; \quad \beta_3\beta_1 = (C_{31} + C_{13})/4 \\ \beta_0\beta_3 &= (C_{12} - C_{21})/4 \quad ; \quad \beta_1\beta_2 = (C_{12} + C_{21})/4 \end{aligned} \tag{5.6}$$

The resulting values are used to form the quaternion from the inertial frame to LVLH frame.

$$q_{I\to LVLH} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \tag{5.7}$$

For aircraft, the body frame is generally related to the LVLH frame through the roll($\phi$), pitch($\theta$), and yaw($\psi$) Euler angle sequence. However, to avoid singularities when describing the orientation of the vehicle and to improve computational performance, the project utilizes quaternions to define orientation. The SpaceCRAFT core system has been adapted to render the vehicle using quaternions rather than the usual Euler angle rotations [16].

$$\begin{aligned} q_0 &= cos\frac{\psi}{2}cos\frac{\theta}{2}cos\frac{\phi}{2} + sin\frac{\psi}{2}sin\frac{\theta}{2}sin\frac{\phi}{2} \\ q_1 &= cos\frac{\psi}{2}cos\frac{\theta}{2}sin\frac{\phi}{2} - sin\frac{\psi}{2}sin\frac{\theta}{2}cos\frac{\phi}{2} \\ q_2 &= cos\frac{\psi}{2}sin\frac{\theta}{2}cos\frac{\phi}{2} + sin\frac{\psi}{2}cos\frac{\theta}{2}sin\frac{\phi}{2} \\ q_3 &= sin\frac{\psi}{2}cos\frac{\theta}{2}cos\frac{\phi}{2} - cos\frac{\psi}{2}sin\frac{\theta}{2}sin\frac{\phi}{2} \end{aligned} \tag{5.8}$$

Although the propagation of the vehicle orientation is calculated in quaternions, the angular rates in the translational and rotational equations of motion use Euler angles, requiring the need for a function to convert a quaternion to an Euler angle set.

$$\begin{aligned} \phi &= atan2(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ \theta &= asin(2(q_0q_2 - q_3q_1)) \\ \psi &= atan2(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{aligned} \tag{5.9}$$

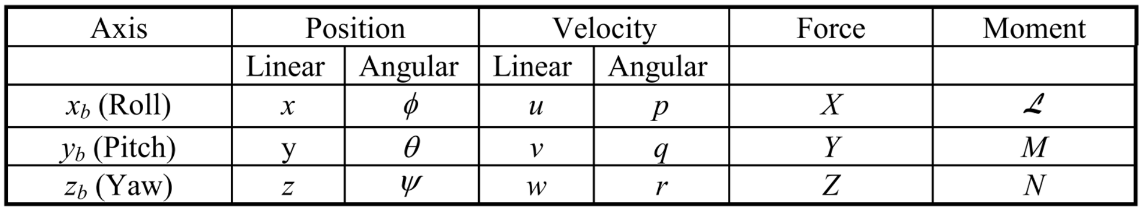
Since the SpaceCRAFT core system needs the location, velocity, and orientation data in the inertial frame, the system must perform multiple rotations during each update. To simplify the process, the quaternions from inertial to LVLH and LVLH to body can be combined through use of the Hamilton product.

### 5.1.2 Equations of Motion

The rigid body dynamic equations used to simulate a six degrees of freedom system contain three moment equations and three force equations. The equations of motion are defined in the body frame of the vehicle, because aerodynamic data and angular rates are measured with respect to this frame. It is also convenient to align the body axis with the principal axes of the system to make all products of inertia equal to zero.

$$\begin{aligned}
I_x\dot{p} + (I_z - I_y)qr &= \mathcal{L}(u, v, w, p, q, r, \rho, S, b) \\
I_y\dot{q} + (I_x - I_z)pr &= \mathcal{M}(u, v, w, p, q, r, \rho, S, c) \\
I_z\dot{r} + (I_y - I_x)pq &= \mathcal{N}(u, v, w, p, q, r, \rho, S, b) \\
m(\dot{u} + qw - rv) + mgsin\theta &= \mathcal{X}(u, v, w, p, q, r, \rho, S) \\
m(\dot{v} + ru - pw) + mgsin\phi cos\theta &= \mathcal{Y}(u, v, w, p, q, r, \rho, S) \\
m(\dot{w} + pv - qu) + mgcos\phi cos\theta &= \mathcal{Z}(u, v, w, p, q, r, \rho, S)
\end{aligned} \tag{5.10}$$

The relationship between the body frame and the variables used in the equations of motion are defined in Figure 5.2.



| Axis | Position | | Velocity | | Force | Moment |
|---|---|---|---|---|---|---|
| | Linear | Angular | Linear | Angular | | |
| $x_b$ (Roll) | $x$ | $\phi$ | $u$ | $p$ | $X$ | $\mathcal{L}$ |
| $y_b$ (Pitch) | y | $\theta$ | $v$ | $q$ | $Y$ | $M$ |
| $z_b$ (Yaw) | $z$ | $\psi$ | $w$ | $r$ | $Z$ | $N$ |

Figure 5.2: Table describing the definition of state variables with respect to the body axis.

As seen in Eq. 5.10, the Euler angles ($\phi$,$\theta$,$\psi$) appear in the equations of motion as well as the angular rates ($p$,$q$,$r$), which are described by the differential equations:

$$\begin{aligned}
\dot{\psi} &= (q sin\phi + r cos\phi) sec\theta \\
\dot{\theta} &= q cos\phi - r sin\phi \\
\dot{\phi} &= p + \dot{\psi} sin\theta
\end{aligned} \tag{5.11}$$

It is easily seen that issues arise when $\theta$ is equal to plus or minus 90°. To overcome this, the quaternion is propagated using the angular rates,

$$\begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{5.12}$$

then converted to Euler angles using Eq 5.9. This process is repeated during each update of the dynamics system at the frequency specified by the user.

### 5.1.3 Calculation of Aerodynamic Forces

For the simulation tool, the aerodynamic characteristics and mass properties of the vehicle are assumed to be given by the user. Aerodynamic characteristics include the aerodynamic flight derivatives and the span and chord of the lifting body. The mass properties include the inertia matrix and mass of the system. The simulation tool calculates the aerodynamic forces on the vehicle based on the current state and uses this information to propagate the dynamics of the vehicle.

The forcing function in the equations of motion represent the external forces and moments acting on the body. One approach to calculating these forces and moments is by using computational fluid dynamics (CFD) which uses numerical analysis to simulate the fluid flow-field. A large research area is the improvement of CFD methods to increase accuracy and computational speed. As such, it is not feasible to use CFD to find the aerodynamic forces. For the simulation tool, it is assumed that the user provides data tables that contain information on the aerodynamic coefficients at varying

Mach numbers, such as coefficient of lift ($C_L$) and drag ($C_D$), with respect to the state of the vehicle.

During a simulation, the Mach number is calculated by first finding the speed of sound

$$c = \sqrt{\gamma * \frac{p}{\rho}} \tag{5.13}$$

where the pressure $p$, density $\rho$, and adiabatic index $\gamma$ are received from the atmospheric model. The adiabatic index is a property of the atmosphere, defined as the ratio of specifics heats of a gas at constant pressure to a gas at constant volume. The Mach number is the velocity divided by the speed of sound.

$$M = \frac{u}{c} \tag{5.14}$$

Now, with the Mach number and current state of the vehicle, the simulation can select the corresponding coefficient values and calculate the forces and moments in the stability frame (LVLH frame) using the generic equation (with $a = X, Y, Z$).

$$\begin{aligned} F_a &= C_a S q \\ M_a &= C_a S q z \end{aligned} \tag{5.15}$$

where $z$ is the moment arm and dynamic pressure $q$ is

$$q = \frac{1}{2}\rho v^2 \tag{5.16}$$

The forces are transformed to the body axis using a quaternion, and applied to the equations of motion.

### 5.1.4 Propagation Method

In the previous sections, we have defined the relationship between reference frames using quaternions, set up the differential equations of motion, and approximated the aerodynamic forces and moments on the vehicle. These components are used to propagate the state of the vehicle during entry using a temporal discretization to find an approximate solution to the ordinary differential equations (ODEs). A fourth order Runge-Kutta method (RK4) was used. Given the initial state and governing differential equations

$$\begin{aligned} y' &= f(t, y) \\ y(t_0) &= \alpha \end{aligned} \tag{5.17}$$

a timestep size $h$ is defined such that $t_i = t_0 + ih$ and the following algorithm is used to compute the approximate solution:

$$\begin{aligned} w_0 &= \alpha \\ k_1 &= hf(t_i, w_i) \\ k_2 &= hf(t_i + \frac{h}{2}, w_i + \frac{k_1}{2}) \\ k_3 &= hf(t_i + \frac{h}{2}, w_i + \frac{k_2}{2}) \\ k_4 &= hf(t_i + h, w_i + k_3) \\ w_{i+1} &= w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{5.18}$$

This method is used to propagate the equations of motion and orientation, in quaternion form, simultaneously. By simultaneously solving the orientation and state, each $k_x$ step uses the intermediate rotation to maintain an accurate application of external forces and moments.

## 5.2 Optimal Guidance/Control Design

The vehicle control system contains and outer guidance loop and an inner control loop. The guidance system provides the reference trajectory and state of the vehicle for the entire entry profile. The inner control loop commands the control surfaces of the vehicle to follow the reference trajectory provided by the guidance loop. In order to ensure that the control loop can follow the guidance path, rate constraints must be applied to the optimal guidance solution. This project attempts to utilize ToC to update the guidance loop at a rate that would improve the overall performance of the vehicle during an entry where perturbations, such as wind, could push the vehicle off course.

The entry vehicle system is stated as an optimal control problem with defined state and control constraints. Typically, there are two main methods used to solve the minimization of the cost function, direct and indirect. The indirect method applies optimal control theory to derive the first-order necessary conditions that the solution must satisfy, but are difficult to implement and used less often in practice than direct methods. ToC is a proposed approach to solving both linear and nonlinear boundary value problems [11]. The following sections will discuss traditional optimal control theory and the ToC methodology in the context of solving a generic two-point boundary value problem (TPVBP), replicating the results obtained by Furfaro. This process will then be used to solve the lifting body entry problem.

### 5.2.1 Optimal Control Theory

An optimal control probolem (OCP) is defined by a cost function, which is minimized and meets a given set of constraints.

$$J = \int_{t_0}^{t_f} \boldsymbol{L}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \tag{5.19}$$

In general, constraints on the optimization problem include the dynamics of the system, boundary conditions, and control/state constraints. Optimal control theory is used to derive the first-order necessary conditions that must be met by the solution. To illustrate the traditional methodology used for the control system design, a generic optimal landing problem is solved analytically. Given the following free final time control minimization problem with state variables $\boldsymbol{r}$ and $\boldsymbol{v}$, thrust control variable $\boldsymbol{a_c}$, and gravity vector $\boldsymbol{a_g}$

$$J = \frac{1}{2} \int_{t_0}^{t_f} \boldsymbol{a_c}^T \boldsymbol{a_c} dt \tag{5.20}$$

$$\begin{aligned} \dot{\boldsymbol{r}} &= \boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{a_c} + \boldsymbol{a_g} \end{aligned} \tag{5.21}$$

with initial and final conditions

$$\begin{aligned}
\boldsymbol{r}(t_0) &= \boldsymbol{r_0} \\
\boldsymbol{v}(t_0) &= \boldsymbol{v_0} \\
\boldsymbol{r}(t_f) &= \boldsymbol{r_f} \\
\boldsymbol{v}(t_f) &= \boldsymbol{v_f} \\
H(t_f) &= 0
\end{aligned} \tag{5.22}$$

the Hamiltonian of the system is defined as

$$\begin{aligned}
H &= L + \boldsymbol{\lambda_r}^T f_r + \boldsymbol{\lambda_v}^T f_v \\
&= \frac{1}{2}\boldsymbol{a_c}^T \boldsymbol{a_c} + \boldsymbol{\lambda_r}^T \boldsymbol{v} + \boldsymbol{\lambda_v}^T (\boldsymbol{a_c} + \boldsymbol{a_g})
\end{aligned} \tag{5.23}$$

where $\lambda_r$ and $\lambda_v$ are the costates for position and velocity. The costate variables are interpreted as Lagrange multipliers associated with the state equations. Next, the partials of the Hamiltonian are used to derive the first-order necessary conditions

$$control: \quad \frac{\partial H}{\partial \boldsymbol{a_c}} = \boldsymbol{0} = \boldsymbol{a_c} + \boldsymbol{\lambda_v} \tag{5.24}$$

$$costate: \quad \begin{aligned} -\frac{\partial H}{\partial \boldsymbol{r}} &= \dot{\boldsymbol{\lambda}}_{\boldsymbol{r}} = \boldsymbol{0} \\ -\frac{\partial H}{\partial \boldsymbol{v}} &= -\dot{\boldsymbol{\lambda}}_{\boldsymbol{v}} = \boldsymbol{\lambda_r} \end{aligned} \qquad state: \quad \begin{aligned} \frac{\partial H}{\partial \boldsymbol{\lambda_r}} &= \dot{\boldsymbol{r}} = \boldsymbol{v} \\ \frac{\partial H}{\partial \boldsymbol{\lambda_v}} &= \dot{\boldsymbol{v}} = \boldsymbol{a_c} + \boldsymbol{a_g} \end{aligned} \tag{5.25}$$

The gravitational acceleration of the Moon, $a_g$ is assumed to be a constant value. With the costate equations, we can assign terminal constraints and integrate to find $\boldsymbol{\lambda_r}(t)$ and $\boldsymbol{\lambda_v}(t)$, where $t_{go} = t_f - t$. The relationship from Eq. 5.24 is used to find the control variable $\boldsymbol{a_c}$.

$$\begin{aligned}
\boldsymbol{\lambda_r} &= \boldsymbol{\nu_r} \\
\boldsymbol{\lambda_v} &= \boldsymbol{\nu_r} t_{go} + \boldsymbol{\nu_v}
\end{aligned} \tag{5.26}$$

$$\boldsymbol{a_c} = -\boldsymbol{\nu_r} t_{go} - \boldsymbol{\nu_v} \tag{5.27}$$

Eq 5.27 can be integrated twice using the terminal state constraints to find the state equations as a function of the constants $\boldsymbol{\nu_r}$ and $\boldsymbol{\nu_r}$. These constants can now be solved in terms of the state variables and substituted into Eq 5.28 to find the analytical expression for the control acceleration [17].

$$\boldsymbol{a_c} = -\frac{4\boldsymbol{v}}{t_{go}} - \frac{6\boldsymbol{r}}{t_{go}^2} - \boldsymbol{a_g} \tag{5.28}$$

The equations derived can be integrated over time using an ODE solver in MATLAB to produce a time history of the states and costates. Using the following boundary conditions and gravity vector for the Moon,

$$\begin{aligned} r_0 &= [0; -1000; 1500] \\ r_f &= [0; 0; 0] \\ v_0 &= [0; 100; -75] \\ v_f &= [0; 0; 0] \\ a_g &= [0; 0; -1.62] \end{aligned} \tag{5.29}$$

Figure 5.3 shows the output of the system using MATLAB's ode45 function and fsolve using 100 time steps on the interval $t = [0, 50]$. As expected from the relationships in Eq. 5.26, the $\boldsymbol{\lambda_r}$ costate is constant and the $\boldsymbol{\lambda_v}$ costate is linear with a slope equal to $-\boldsymbol{\lambda_r}$.

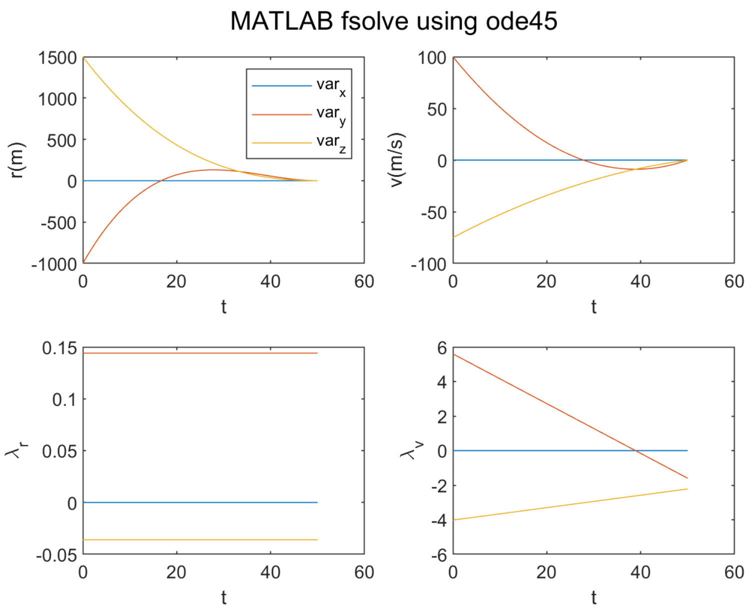


Figure 5.3: Plot of the time histories of the state and costates of the OCP using MATLAB fsolve and ode45.

### 5.2.2 Application of Theory of Connections

ToC provides general interpolation functions with an embedded set of linear constraints on the boundary conditions to obtain a fast and accurate least-squares solution [11]. Simply put, the method replaces the state and costate equations with a general function $g(t)$ and $n$ constraint functions $C(t)$, where $\boldsymbol{\beta}$ is a special function of time and the constraint times.

$$y(t) = g(t) + \sum_{k=1}^{n} \eta_k C_k(t) = g(t) + \boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{C}(t) \tag{5.30}$$

To demonstrate the accuracy of the ToC, the problem presented in Section 5.2.1 will be solved and compared with the analytical result. The ToC methodology starts with the first-order necessary conditions derived in Eqs 5.24 and 5.25. The state and costate equations are expressed as constrained expressions with embedded initial and terminal conditions. The active nature of each constraint is determined by multiplying each condition with a time-dependent function. The $\boldsymbol{g}_*$ function is now free, meaning it is no longer subject to any constraints.

$$\begin{aligned}
\boldsymbol{r}(t) &= \boldsymbol{g_r}(t) + \frac{t-t_f}{t_0-t_f}[\boldsymbol{r_0} - \boldsymbol{g_r}(t_0)] + \frac{t-t_0}{t_f-t_0}[\boldsymbol{r_f} - \boldsymbol{g_r}(t_f)] \\
\boldsymbol{v}(t) &= \boldsymbol{g_v}(t) + \frac{t-t_f}{t_0-t_f}[\boldsymbol{v_0} - \boldsymbol{g_v}(t_0)] + \frac{t-t_0}{t_f-t_0}[\boldsymbol{v_f} - \boldsymbol{g_v}(t_f)] \\
\boldsymbol{\lambda_r}(t) &= \boldsymbol{g_{\lambda_r}}(t) \\
\boldsymbol{\lambda_v}(t) &= \boldsymbol{g_{\lambda_v}}(t)
\end{aligned} \tag{5.31}$$

With the following relationships,

$$\Delta t = t_f - t_0 \qquad \boldsymbol{\beta_1} = \frac{\boldsymbol{t} - t_f}{t_0 - t_f} \qquad \boldsymbol{\beta_2} = \frac{\boldsymbol{t} - t_0}{t_f - t_0} \tag{5.32}$$

Eq. 5.30 is substituted into Eq. 5.26 to find the state and costate equations in terms of the general function $\boldsymbol{g_*}(t)$.

$$\begin{aligned}
&\dot{\boldsymbol{g}}_{\boldsymbol{r}} + \frac{1}{\Delta t}[\boldsymbol{g_r}(t_0) - \boldsymbol{g_r}(t_f)] + \beta_1 \boldsymbol{g_v}(t_0) + \beta_2 \boldsymbol{g_v}(t_f) - \boldsymbol{g_v} = \frac{1}{\Delta t}[\boldsymbol{r_0} - \boldsymbol{r_f}] + \beta_1 \boldsymbol{v_0} + \beta_2 \boldsymbol{v_f} \\
&\dot{\boldsymbol{g}}_{\boldsymbol{v}} + \frac{1}{\Delta t}[\boldsymbol{g_v}(t_0) - \boldsymbol{g_v}(t_f)] + \boldsymbol{g_{\lambda_v}} = \boldsymbol{a_g} + \frac{1}{\Delta t}[\boldsymbol{v_0} - \boldsymbol{v_f}] \\
&\dot{\boldsymbol{g}}_{\boldsymbol{\lambda_r}} = \boldsymbol{0} \\
&\dot{\boldsymbol{g}}_{\boldsymbol{\lambda_v}} + \boldsymbol{g_{\lambda_r}} = \boldsymbol{0}
\end{aligned} \tag{5.33}$$

The goal is now to find the $\boldsymbol{g_*}$ functions that best fit the system of equations in Eq 5.31. Each function is expanded using a set of $m$ orthogonal polynomial basis functions $\boldsymbol{h}$ with the coefficients for each basis function given in the vector $\boldsymbol{\xi}$.

$$\boldsymbol{g} = \boldsymbol{\xi^T h} \tag{5.34}$$

It is important to note that the basis function vector $\boldsymbol{h}$ is dependent on the state and costate equations in Eq. 5.30. The structure of the state constrained expressions, $r$ and $v$, inherently contain information in the $0^{th}$ and $1^{st}$ orders, therefore, to maintain linear independence, the basis functions start at the $2^{nd}$ order polynomial. For this problem, Chebyshev polynomials of the first kind were used as the basis functions. The Chebyshev polynomials are defined on the interval $[-1,+1]$, therefore we can make a variable change from $t \in [t_0, t_f]$ to $x \in [-1,+1]$.

$$\frac{t-t_0}{t_f-t_0} = \frac{x+1}{2} \tag{5.35}$$

The time derivative of is now expressed as

$$\dot{f} = \frac{df}{dt} = \frac{df}{dx}\frac{dx}{dt} = \frac{2}{dt}f' = cf' \tag{5.36}$$

With the variable change and expanding the solution with Eq. 5.33 gives a linear system of equations which can be assembled into the form $\mathbb{A}\boldsymbol{\xi} = \boldsymbol{b}$, where

$$\boldsymbol{p} = c\boldsymbol{h}'^T + \frac{\boldsymbol{h_0}^T - \boldsymbol{h_f}^T}{\Delta t} \qquad \boldsymbol{q} = -\boldsymbol{h}^T + \frac{(1+x)\boldsymbol{h_f}^T + (1-x)\boldsymbol{h_0}^T}{2} \qquad \boldsymbol{0} = \boldsymbol{0}_{m\times 1} \tag{5.37}$$

$$\mathbb{A} = \begin{bmatrix}
\boldsymbol{p} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{q} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{p} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{q} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{p} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{q} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{p} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{h}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{p} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{h}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{p} & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{h}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & c\boldsymbol{h}'^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & c\boldsymbol{h}'^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & c\boldsymbol{h}'^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{h}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & c\boldsymbol{h}'^T & \boldsymbol{0}^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{h}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & c\boldsymbol{h}'^T & \boldsymbol{0}^T \\
\boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{h}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & c\boldsymbol{h}'^T
\end{bmatrix} \tag{5.38}$$

$$\boldsymbol{b}_{12\times 1} = \begin{bmatrix} \frac{1}{2}((1-x)\boldsymbol{v_0} + (1+x)\boldsymbol{v_f}) - \frac{1}{\Delta t}(\boldsymbol{r_f} - \boldsymbol{r_0}) \\ \boldsymbol{a_g} + \frac{1}{\Delta t}(\boldsymbol{v_0} - \boldsymbol{v_f}) \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{5.39}$$

To find the solution to $\boldsymbol{g_*}$, $\mathbb{A}$ and $\boldsymbol{b}$ are evaluated at $N$ timesteps on the interval $[t_0, t_f]$ to create the augmented matrix $\mathbb{H}$ and vector $\boldsymbol{c}$.

$$\mathbb{H}_{12N\times 12m} = \begin{bmatrix} \mathbb{A}_1 \\ \mathbb{A}_2 \\ \vdots \\ \mathbb{A}_N \end{bmatrix} \qquad \boldsymbol{c}_{12N\times 1} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_N \end{bmatrix} \tag{5.40}$$

The resulting system $\mathbb{H}\boldsymbol{\xi} = \boldsymbol{c}$ can be solved for the vector of coefficients using linear least-squares:

$$\boldsymbol{\xi} = (\mathbb{H}^T\mathbb{H})^{-1}\mathbb{H}^T\boldsymbol{c} \tag{5.41}$$

These coefficients are used in Eq. 5.33 to find the generic functions, $\boldsymbol{g_*}$, which are substituted into Eq. 5.30 to find the time history of optimal state and costate equations. Using an expansion of four Chebyshev polynomials, the result of the ToC OCP and the errors between the analytical solution from section 5.2.1 are shown in Figures 5.4 and 5.5 respectively. The errors in the states and costates of the system are on the order of $10^{-13}$.

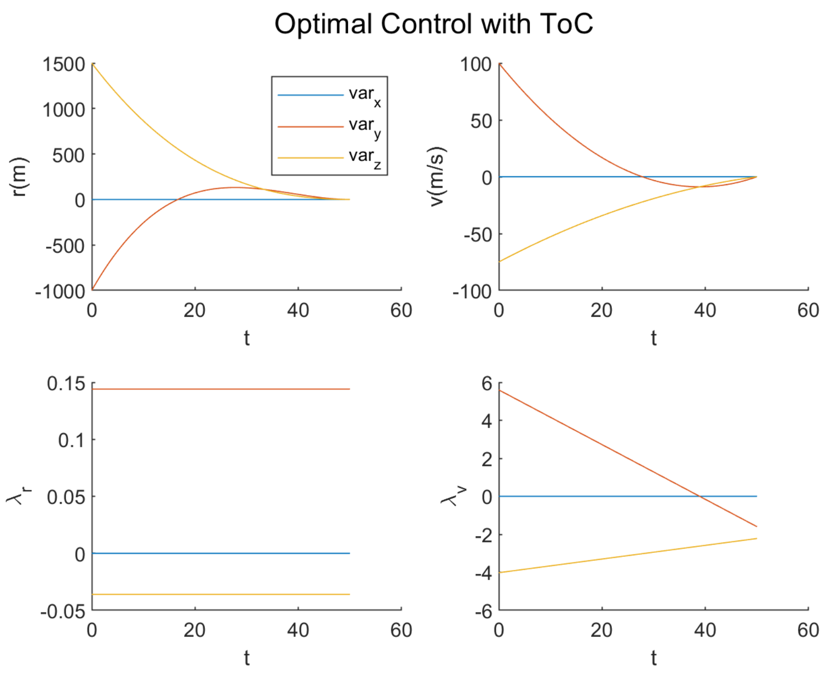


Figure 5.4: Time histories of the state and costates of the OCP using the ToC method.

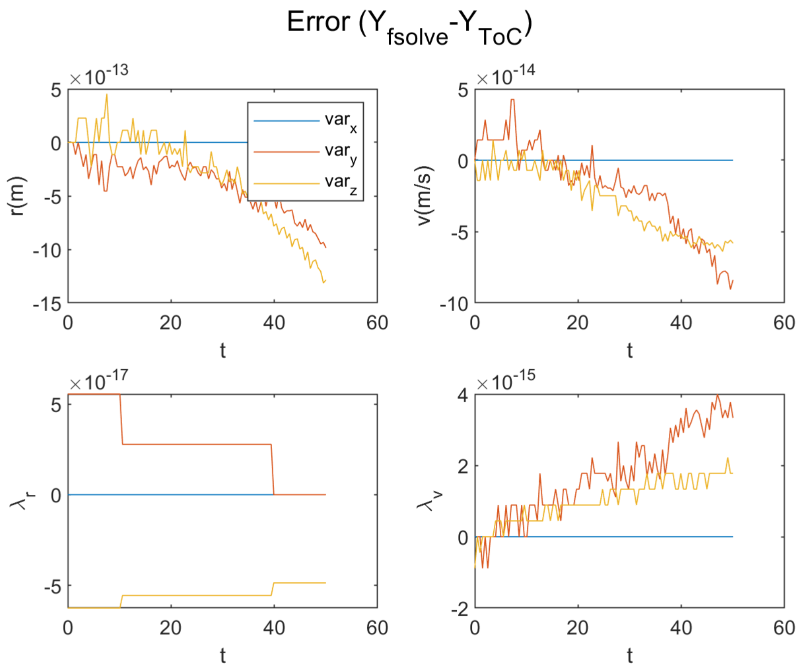


Figure 5.5: Error between the analytical results from Figure 5.3 and ToC results from Figure 5.4.

### 5.2.3 Optimal Guidance for a Lifting Body Entry Vehicle

The outer control loop of the test entry vehicle used for this thesis will use optimal control theory and ToC to solve for the optimal guidance path. Current research in the area of ToC has not determined a method with which to apply state or control constraints to the optimal control problem, therefore, this solution will be unconstrained. Different the previous example, the equations of motion for a lifting body entry are nonlinear, requiring a nonlinear least-squares algorithm to converge on the optimal solution. To demonstrate the nonlinear case, the Apollo atmospheric reentry problem presented in Reference [19] is solved. The vehicle is a lifting body with bank angle ($\beta$) as the control variable.

$$\dot{r} = V sin\gamma \tag{5.42}$$

$$\dot{\theta} = \frac{V cos\gamma cos\psi}{r cos\phi} \tag{5.43}$$

$$\dot{\phi} = \frac{V cos\gamma sin\psi}{r} \tag{5.44}$$

$$\dot{V} = -D - \frac{\mu sin\gamma}{r^2} \tag{5.45}$$

$$\dot{\gamma} = \frac{L cos\beta}{V} + (\frac{V}{r} - \frac{\mu}{r^2 V}) cos\gamma \tag{5.46}$$

$$\dot{\psi} = -\frac{L sin\beta}{V cos\gamma} - \frac{V cos\gamma cos\psi sin\phi}{r cos\phi} \tag{5.47}$$

with radius ($r$), longitude ($\theta$), latitude ($\phi$), velocity ($v$), flight path angle ($\gamma$), and heading angle ($\psi$). The lift ($L$) and drag ($D$) are defined as

$$L = \frac{1}{2} c_L S \rho V^2 \tag{5.48}$$

$$D = \frac{1}{2} c_D S \rho V^2 \tag{5.49}$$

where $c_L$ and $c_D$ are dependent on the angle of attack

$$c_L = c_{L_0} + c_{L_1} \alpha \tag{5.50}$$

$$c_D = c_{D_0} + c_{D_1}\alpha + c_{D_2}\alpha^2 \tag{5.51}$$

and atmospheric density ($\rho$) is modeled by the equation

$$\rho = \rho_0 e^{-k(r-r_E)} \tag{5.52}$$

From the reference problem statement, $\alpha$ is held constant.

Two important factors of an entry profile are the accelerations experienced by the vehicle and the aerodynamic heating, both of which are addressed by the optimal guidance solution in the cost function. For the heating rate ($q$), the relationship used is

$$q = \epsilon\sqrt{\rho}V^3 \tag{5.53}$$

where $\epsilon$ is a weighting variable. The vehicle acceleration can be stated as the magnitude of the forces acting on the body, in this case, lift and drag. As such, the cost function will minimize the magnitude of these forces and the vehicle heating.

$$J = \int_{t_0}^{t_f} \sqrt{L^2 + D^2} + q\,dt \tag{5.54}$$

Optimal control theory starts with formulating the Hamiltonian of the system.

$$\begin{aligned} H = {} & \sqrt{L^2 + D^2} + \epsilon\sqrt{\rho}V^3 + \lambda_r\Big[V sin\gamma\Big] + \lambda_\theta\Big[\frac{V cos\gamma cos\psi}{r cos\phi}\Big] + \lambda_\phi\Big[\frac{V cos\gamma sin\psi}{r}\Big] \\ & + \lambda_V\Big[-D - \frac{\mu sin\gamma}{r^2}\Big] + \lambda_\gamma\Big[\frac{L cos\beta}{V} + (\frac{V}{r} - \frac{\mu}{r^2 V})cos\gamma\Big] \\ & + \lambda_\psi\Big[-\frac{L sin\beta}{V cos\gamma} - \frac{V cos\gamma cos\psi sin\phi}{r cos\phi}\Big] \end{aligned} \tag{5.55}$$

Next, take the partials of the Hamiltonian with respect to the state variables to find the differential equations for the costates.

$$\begin{aligned}-\frac{\partial H}{\partial r} = \dot{\lambda_r} = \; & -Q_r + \lambda_\theta\Big[\frac{Vcos\gamma cos\psi}{r^2cos\phi}\Big] + \lambda_\phi\Big[\frac{Vcos\gamma sin\psi}{r^2}\Big] \\ & - \lambda_V\Big[-D_r + \frac{2\mu sin\gamma}{r^3}\Big] - \lambda_\gamma\Big[\frac{L_rcos\beta}{V} + (-\frac{V}{r^2} + \frac{2\mu}{r^3V})cos\gamma\Big] \\ & + \lambda_\psi\Big[\frac{L_rsin\beta}{Vcos\gamma} - \frac{Vcos\gamma cos\psi sin\phi}{r^2cos\phi}\Big]\end{aligned} \tag{5.56}$$

$$-\frac{\partial H}{\partial \theta} = \dot{\lambda_\theta} = \; 0 \tag{5.57}$$

$$-\frac{\partial H}{\partial \phi} = \dot{\lambda_\phi} = \; -\lambda_\theta\Big[\frac{Vcos\gamma cos\psi sin\phi}{rcos^2\phi}\Big] + \lambda_\psi\Big[\frac{Vcos\gamma cos\psi}{rcos^2\phi}\Big] \tag{5.58}$$

$$\begin{aligned}-\frac{\partial H}{\partial V} = \dot{\lambda_V} = \; & -Q_V - \lambda_r\Big[sin\gamma\Big] - \lambda_\theta\Big[\frac{cos\gamma cos\psi}{rcos\phi}\Big] - \lambda_\phi\Big[\frac{cos\gamma sin\psi}{r}\Big] \\ & + \lambda_V\Big[D_V\Big] - \lambda_\gamma\Big[\frac{(VL_V - L)cos\beta}{V^2} + (\frac{1}{r} + \frac{\mu}{r^2V^2})cos\gamma)\Big] \\ & + \lambda_\psi\Big[\frac{(VL_V - L)sin\beta}{V^2cos\gamma} + \frac{cos\gamma cos\psi sin\phi}{rcos\phi}\Big]\end{aligned} \tag{5.59}$$

$$\begin{aligned}-\frac{\partial H}{\partial \gamma} = \dot{\lambda_\gamma} = \; & -\lambda_r\Big[Vcos\gamma\Big] + \lambda_\theta\Big[\frac{Vsin\gamma cos\psi}{rcos\phi}\Big] + \lambda_\phi\Big[\frac{Vsin\gamma sin\psi}{r}\Big] \\ & + \lambda_V\Big[\frac{\mu cos\gamma}{r^2}\Big] + \lambda_\gamma\Big[(\frac{V}{r} - \frac{\mu}{r^2V})sin\gamma)\Big] \\ & + \lambda_\psi\Big[\frac{Lsin\beta sin\gamma}{Vcos^2\gamma} - \frac{Vsin\gamma cos\psi sin\phi}{rcos\phi}\Big]\end{aligned} \tag{5.60}$$

$$-\frac{\partial H}{\partial \psi} = \dot{\lambda_\psi} = \; \lambda_\theta\Big[\frac{Vcos\gamma sin\psi}{rcos\phi}\Big] - \lambda_\phi\Big[\frac{Vcos\gamma cos\psi}{r}\Big] - \lambda_\psi\Big[\frac{Vcos\gamma sin\psi sin\phi}{rcos\phi}\Big] \tag{5.61}$$

where,

$$\begin{aligned}
Q_r &= -\frac{1}{2}k\rho V^2 S\sqrt{C_L^2 + C_D^2} - \frac{1}{2}k\epsilon V^3\sqrt{\rho} \\
Q_V &= \rho S V\sqrt{C_L^2 + C_D^2} + 3\epsilon V^2\sqrt{\rho} \\
L_r &= -\frac{1}{2}k\rho V^2 S C_L \\
L_V &= \rho V S C_L \\
D_r &= -\frac{1}{2}k\rho V^2 S C_D \\
D_V &= \rho V S C_D
\end{aligned}$$

The partial of the Hamiltonian with respect to the control variable produces the optimality condition.

$$\frac{\partial H}{\partial \beta} = 0 = \; -\lambda_\gamma\Big[\frac{Lsin\beta}{mV}\Big] + \lambda_\psi\Big[\frac{Lcos\beta}{mVcos\gamma}\Big] \tag{5.62}$$

From here, it is necessary to derive expressions for the control in terms of the states and costates, which are then substituted into the state and costate differential equations. Using $H_\beta = 0$ and $H_{\beta\beta} \geq 0$, the following relationship can be derived.

$$cos\beta = -\frac{\lambda_\gamma cos\gamma}{\sqrt{\lambda_\psi^2 + \lambda_\gamma^2 cos^2\gamma}} \qquad\qquad sin\beta = \frac{\lambda_\psi}{\sqrt{\lambda_\psi^2 + \lambda_\gamma^2 cos^2\gamma}} \tag{5.63}$$

At this stage it is possible to express the state and costate differential equations in terms of the states, costates, and constant variables.

As seen above, the resulting system of differential equations are nonlinear, therefore, a nonlinear least-squares approach is posed. The approach starts with the constrained expressions from ToC with an initial guess for the basis function coefficients. The ability for the solution to converge is highly dependent on the initial guess of these coefficients. At the start of the trajectory, the equations of motion are propagated forward with MATLAB's ode45 function. The resulting state and costate history is used in a linear least-squares to find the starting values for $\xi$. The resulting output from this guess is used in a nonlinear least-squares iteration to find the change in the coefficient values, where the loss function $L$ for each differential equation $\dot{y}(t) = f(y,t)$ is defined on the new interval $x \in [-1,+1]$ as

$$L = cy'(x) - f(y,x,t) = 0 \tag{5.64}$$

This results in a vector of loss values at each timestep for each differential equation. The $L_2$-norm of these vector is then evaluated to determine if the current solution has converged. Figure 5.6 provides an illustration of the nonlinear least-squares algorithm.

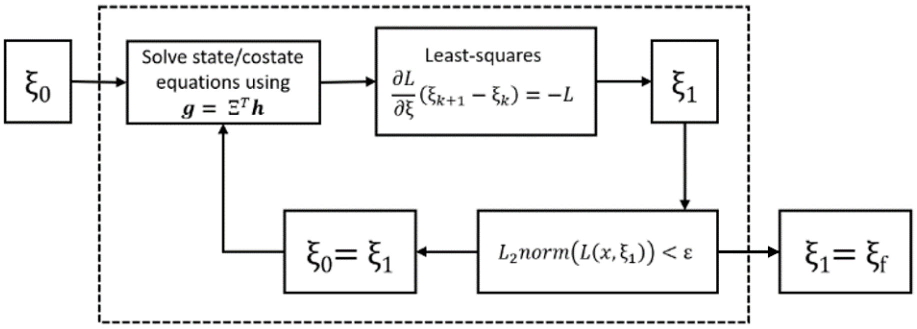

$\xi_0$
Solve state/costate equations using $\boldsymbol{g} = \Xi^T \boldsymbol{h}$
Least-squares $\frac{\partial L}{\partial \xi}(\xi_{k+1} - \xi_k) = -L$
$\xi_1$
$L_2 norm(L(x,\xi_1)) < \varepsilon$
$\xi_0 = \xi_1$
$\xi_1 = \xi_f$

Figure 5.6: Nonlinear least-squares process for solving optimal control problem with ToC

For the guidance problem, we have the following boundary conditions at initial and final time:

$$\begin{aligned}
r(t_0) &= r_0 & \lambda_r(t_f) &= 0 \\
\theta(t_0) &= \theta_0 & \theta(t_f) &= \theta_f \\
\phi(t_0) &= \phi_0 & \phi(t_f) &= \phi_f \\
V(t_0) &= V_0 & V(t_f) &= V_f \\
\gamma(t_0) &= \gamma_0 & \lambda_\gamma(t_f) &= 0 \\
\psi(t_0) &= \psi_0 & \lambda_\psi(t_f) &= 0
\end{aligned}$$

From ToC, the constrained expressions for the states and costates are

$$\begin{aligned}
r(t) &= g_r(t) + \Big[r_0 - g_r(t_0)\Big] \\
\theta(t) &= g_\theta(t) - \frac{t-t_f}{\Delta t}\Big[\theta_0 - g_\theta(t_0)\Big] + \frac{t-t_0}{\Delta t}\Big[\theta_f - g_\theta(t_f)\Big] \\
\phi(t) &= g_\phi(t) - \frac{t-t_f}{\Delta t}\Big[\phi_0 - g_\phi(t_0)\Big] + \frac{t-t_0}{\Delta t}\Big[\phi_f - g_\phi(t_f)\Big] \\
V(t) &= g_V(t) - \frac{t-t_f}{\Delta t}\Big[V_0 - g_V(t_0)\Big] + \frac{t-t_0}{\Delta t}\Big[V_f - g_V(t_f)\Big] \\
\gamma(t) &= g_\gamma(t) + \Big[\gamma_0 - g_\gamma(t_0)\Big] \\
\psi(t) &= g_\psi(t) + \Big[\psi_0 - g_\psi(t_0)\Big] \\
\lambda_r(t) &= g_{\lambda_r}(t) + \Big[\lambda_{r_f} - g_{\lambda_r}(t_f)\Big] \\
\lambda_\theta(t) &= g_{\lambda_\theta}(t) \\
\lambda_\phi(t) &= g_{\lambda_\phi}(t) \\
\lambda_V(t) &= g_{\lambda_V}(t) \\
\lambda_\gamma(t) &= g_{\lambda_\gamma}(t) + \Big[\lambda_{\gamma_f} - g_{\lambda_\gamma}(t_f)\Big] \\
\lambda_\psi(t) &= g_{\lambda_\psi}(t) + \Big[\lambda_{\psi_f} - g_{\lambda_\psi}(t_f)\Big]
\end{aligned} \tag{5.65}$$

The constrained functions are kept in the time domain by converting the basis functions with the $c$ value from Eq. 5.36.

$$\begin{aligned}
r &= \Big[h - h_0\Big]^T \xi_r + r_0 & \lambda_r &= \Big[h - h_f\Big]^T \xi_{\lambda_r} + \lambda_{r_f} \\
\theta &= p^T \xi_\theta + q_\theta & \lambda_\theta &= h^T \xi_{\lambda_\theta} \\
\phi &= p^T \xi_\phi + q_\phi & \lambda_\phi &= h^T \xi_{\lambda_\phi} \\
V &= p^T \xi_V + q_V & \lambda_V &= h^T \xi_{\lambda_V} \\
\gamma &= \Big[h - h_0\Big]^T \xi_\gamma + \gamma_0 & \lambda_\gamma &= \Big[h - h_f\Big]^T \xi_{\lambda_\gamma} + \lambda_{\gamma_f} \\
\psi &= \Big[h - h_0\Big]^T \xi_\psi + \psi_0 & \lambda_\psi &= \Big[h - h_f\Big]^T \xi_{\lambda_\psi} + \lambda_{\psi f}
\end{aligned} \tag{5.66}$$

$$\begin{aligned}
p^T &= \Big[h + \frac{1}{\Delta t}[(t - t_f)h_0 + (t - t_0)h_f]\Big]^T \\
q_* &= \frac{1}{\Delta t}[(t - t_0) *_f - (t - t_f) *_0]
\end{aligned}$$

The derivatives of the constrained functions are obtained using the relationship in Eq. 5.36.

$$\begin{aligned}
\dot{r} &= \Big[ch'\Big]^T \xi_r & \dot{\lambda}_r &= \Big[ch'\Big]^T \xi_{\lambda_r} \\
\dot{\theta} &= m^T \xi_\theta + n_\theta & \dot{\lambda}_\theta &= \Big[ch'\Big]^T \xi_{\lambda_\theta} \\
\dot{\phi} &= m^T \xi_\phi + n_\phi & \dot{\lambda}_\phi &= \Big[ch'\Big]^T \xi_{\lambda_\phi} \\
\dot{V} &= m^T \xi_V + n_V & \dot{\lambda}_V &= \Big[ch'\Big]^T \xi_{\lambda_V} \\
\dot{\gamma} &= \Big[ch'\Big]^T \xi_\gamma & \dot{\lambda}_\gamma &= \Big[ch'\Big]^T \xi_{\lambda_\gamma} \\
\dot{\psi} &= \Big[ch'\Big]^T \xi_\psi & \dot{\lambda}_\psi &= \Big[ch'\Big]^T \xi_{\lambda_\psi}
\end{aligned} \tag{5.67}$$

$$\begin{aligned}
m^T &= \Big[ch' + \frac{1}{\Delta t}[h_0 - h_f]\Big]^T \\
n_* &= \frac{1}{\Delta t}[*_f - *_0]
\end{aligned}$$

These functions are substituted into the $L$ functions found from the differential equations in Eqs. 5.42-47 and Eqs. 5.56-61.

$$L_1 = \dot{r} - Vsin\gamma$$

$$L_2 = \dot{\theta} - \frac{Vcos\gamma cos\psi}{rcos\phi}$$

$$L_3 = \dot{\phi} - \frac{Vcos\gamma sin\psi}{r}$$

$$L_4 = \dot{V} + D + \frac{\mu sin\gamma}{r^2}$$

$$L_5 = \dot{\gamma} - \frac{Lcos\beta}{V} - (\frac{V}{r} - \frac{\mu}{r^2 V})cos\gamma$$

$$L_6 = \dot{\psi} + \frac{Lsin\beta}{Vcos\gamma} + \frac{Vcos\gamma cos\psi sin\phi}{rcos\phi}$$

$$L_7 = \dot{\lambda_r} - \Bigg( - Q_r + \lambda_\theta\Big[\frac{Vcos\gamma cos\psi}{r^2 cos\phi}\Big] + \lambda_\phi\Big[\frac{Vcos\gamma sin\psi}{r^2}\Big] - \lambda_V\Big[ - D_r + \frac{2\mu sin\gamma}{r^3}\Big]$$
$$- \lambda_\gamma\Big[\frac{L_r cos\beta}{V} + (-\frac{V}{r^2} + \frac{2\mu}{r^3 V})cos\gamma\Big] + \lambda_\psi\Big[\frac{L_r sin\beta}{Vcos\gamma} - \frac{Vcos\gamma cos\psi sin\phi}{r^2 cos\phi}\Big]\Bigg)$$

$$L_8 = \dot{\lambda_\theta}$$

$$L_9 = \dot{\lambda_\phi} - \Bigg( - \lambda_\theta\Big[\frac{Vcos\gamma cos\psi sin\phi}{rcos^2\phi}\Big] + \lambda_\psi\Big[\frac{Vcos\gamma cos\psi}{rcos^2\phi}\Big]\Bigg)$$

$$L_{10} = \dot{\lambda_V} - \Bigg( - Q_V - \lambda_r\Big[sin\gamma\Big] - \lambda_\theta\Big[\frac{cos\gamma cos\psi}{rcos\phi}\Big] - \lambda_\phi\Big[\frac{cos\gamma sin\psi}{r}\Big] + \lambda_V\Big[D_V\Big]$$
$$- \lambda_\gamma\Big[\frac{(VL_V - L)cos\beta}{V^2} + (\frac{1}{r} + \frac{\mu}{r^2 V^2})cos\gamma)\Big] + \lambda_\psi\Big[\frac{(VL_V - L)sin\beta}{V^2 cos\gamma} + \frac{cos\gamma cos\psi sin\phi}{rcos\phi}\Big]\Bigg)$$

$$L_{11} = \dot{\lambda_\gamma} - \Bigg( - \lambda_r\Big[Vcos\gamma\Big] + \lambda_\theta\Big[\frac{Vsin\gamma cos\psi}{rcos\phi}\Big] + \lambda_\phi\Big[\frac{Vsin\gamma sin\psi}{r}\Big] + \lambda_V\Big[\frac{\mu cos\gamma}{r^2}\Big]$$
$$+ \lambda_\gamma\Big[(\frac{V}{r} - \frac{\mu}{r^2 V})sin\gamma)\Big] + \lambda_\psi\Big[\frac{Lsin\beta sin\gamma}{Vcos^2\gamma} - \frac{Vsin\gamma cos\psi sin\phi}{rcos\phi}\Big]\Bigg)$$

$$L_{12} = \dot{\lambda_\psi} - \Bigg(\lambda_\theta\Big[\frac{Vcos\gamma sin\psi}{rcos\phi}\Big] - \lambda_\phi\Big[\frac{Vcos\gamma cos\psi}{r}\Big] - \lambda_\psi\Big[\frac{Vcos\gamma sin\psi sin\phi}{rcos\phi}\Big]\Bigg)$$

With the new set of 12 differential $L$ equations, a Jacobian matrix is formed with respect to the $\boldsymbol{\xi}_*$ vector. Since calculation of these partials are time-intensive, and accuracy is critical, the symbolic equations are found using the Wolfram Mathematica 12.0 solver. The matrix $\mathbb{A}$ is expressed as:

$$\begin{bmatrix}
\frac{\partial \boldsymbol{L_1}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_1}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_1}}{\partial \boldsymbol{\xi_\gamma}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\
\frac{\partial \boldsymbol{L_2}}{\partial \boldsymbol{\xi_r}}^T & \frac{\partial \boldsymbol{L_2}}{\partial \boldsymbol{\xi_\theta}}^T & \frac{\partial \boldsymbol{L_2}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_2}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_2}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_2}}{\partial \boldsymbol{\xi_\psi}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\
\frac{\partial \boldsymbol{L_3}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_3}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_3}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_3}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_3}}{\partial \boldsymbol{\xi_\psi}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\
\frac{\partial \boldsymbol{L_4}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_4}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_4}}{\partial \boldsymbol{\xi_\gamma}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\
\frac{\partial \boldsymbol{L_5}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_5}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_5}}{\partial \boldsymbol{\xi_\gamma}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_5}}{\partial \boldsymbol{\xi_{\lambda_V}}}^T & \frac{\partial \boldsymbol{L_5}}{\partial \boldsymbol{\xi_{\lambda_\gamma}}}^T & \frac{\partial \boldsymbol{L_5}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T \\
\frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_\psi}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_{\lambda_V}}}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_{\lambda_\gamma}}}^T & \frac{\partial \boldsymbol{L_6}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T \\
\frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_\psi}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_{\lambda_r}}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_{\lambda_\theta}}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_{\lambda_\phi}}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_{\lambda_V}}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_{\lambda_\gamma}}}^T & \frac{\partial \boldsymbol{L_7}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T \\
\mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_8}}{\partial \boldsymbol{\xi_{\lambda_\theta}}}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T & \mathbf{0}^T \\
\frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_\psi}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_{\lambda_\theta}}}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_{\lambda_\phi}}}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_9}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T \\
\frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_\psi}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_{\lambda_r}}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_{\lambda_\theta}}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_{\lambda_\phi}}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_{\lambda_V}}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_{\lambda_\gamma}}}^T & \frac{\partial \boldsymbol{L_{10}}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T \\
\frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_\psi}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_{\lambda_r}}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_{\lambda_\theta}}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_{\lambda_\phi}}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_{\lambda_V}}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_{\lambda_\gamma}}}^T & \frac{\partial \boldsymbol{L_{11}}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T \\
\frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_r}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_\phi}}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_V}}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_\gamma}}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_\psi}}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_{\lambda_\theta}}}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_{\lambda_\phi}}}^T & \mathbf{0}^T & \mathbf{0}^T & \frac{\partial \boldsymbol{L_{12}}}{\partial \boldsymbol{\xi_{\lambda_\psi}}}^T
\end{bmatrix}$$

The resulting system of equations is of the form

$$\left[\frac{d\boldsymbol{L}}{d\boldsymbol{\xi}}(\boldsymbol{x_k}, \boldsymbol{\lambda_k}, \boldsymbol{\xi_k})\right]^T (\boldsymbol{\xi_{k+1}} - \boldsymbol{\xi_k}) = -\boldsymbol{L}(\boldsymbol{x_k}, \boldsymbol{\lambda_k}, \boldsymbol{\xi_k}) \tag{5.68}$$

which will act as the least-squares block shown in Figure 5.6.

Starting from the initial guess of the $\boldsymbol{\xi}$, the states, costates, and their derivatives are created from the constrained expressions in Eqs. 5.68 and 5.69. These values are applied to the Jacobian matrix and $L$ functions. The Jacobian matrix is of the size $N \times M$, where $N$ is the number of timesteps and $M$ is the number of basis functions used in the expansion. The value of the $d\boldsymbol{\xi}$ vector is found using MATLAB's psuedo-inverse command $A\backslash b$. The change in $\boldsymbol{\xi}$ is then added to the original vector of coefficients. With the new coefficients, the states and costates are reconstructed and used to find the residuals of the 12 $L$ functions. This process is repeated until the 2-norm of the each set of residuals converges. To improve the stability of the convergence, the algorithm takes half-steps when the 2-norm passes a sufficiently small threshold.

For this problem, initial guesses for the costates of the system are given in the reference, allowing for the propagation of the states and costates using ode45. The output of this initial propagation is provided in Figures 5.7-5.9.

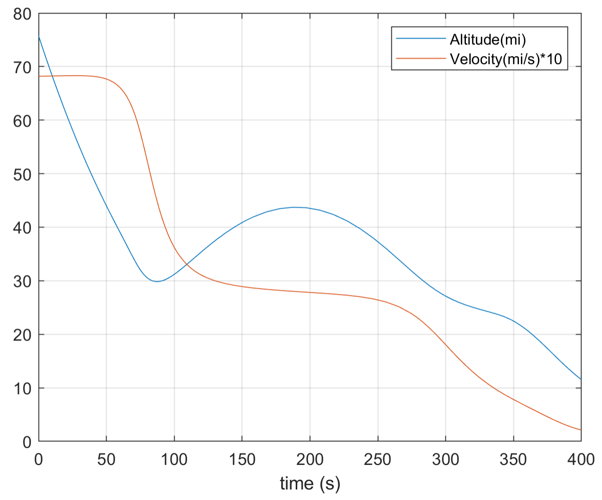


Figure 5.7: Initial propagation of states and costates (altitude/velocity).

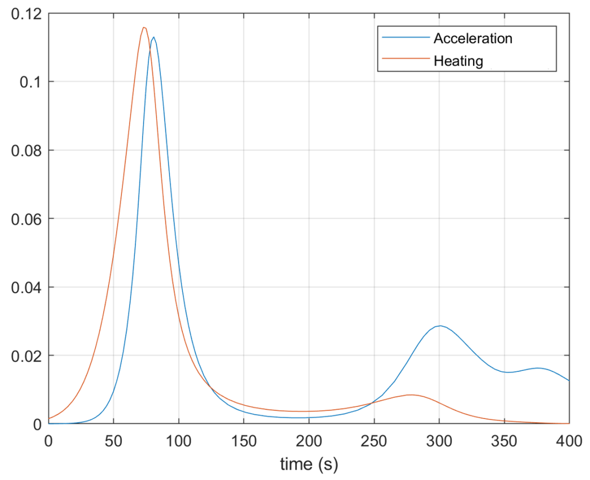


Figure 5.8: Initial propagation of states and costates (Cost function components).

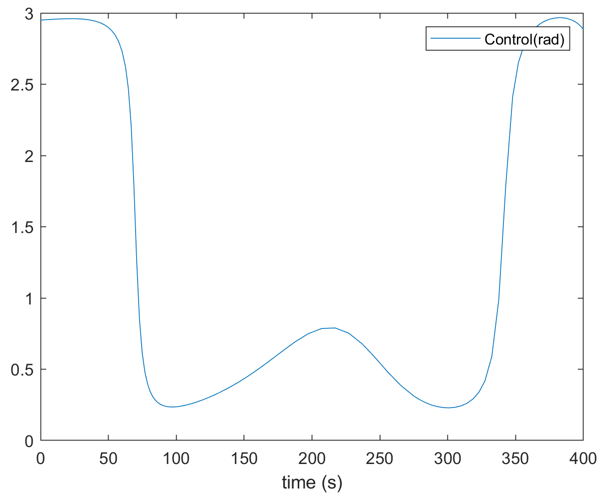
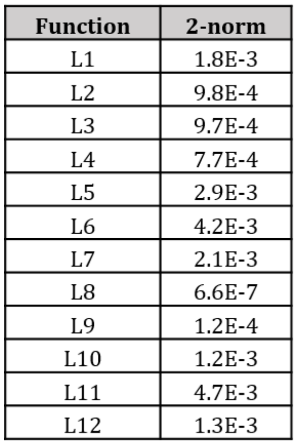


Figure 5.9: Initial propagation of states and costates (Control).

These time histories are used in a least-square to derive the initial guess for the coefficients $\boldsymbol{\xi_0}$. Since this action is only performed at the start of the simulation, the computation time of these values does not effect the speed of the real-time guidance update. For this example, a set of 20 basis functions are used with time discretized into 1000 timesteps. With this starting vector, the ToC solution converged in 2.9025 seconds, with the residuals shown in Figure 5.10. The final time was set at 400 seconds in conjunction with the final conditions in Reference [19].



| Function | 2-norm |
|---|---|
| L1 | 1.8E-3 |
| L2 | 9.8E-4 |
| L3 | 9.7E-4 |
| L4 | 7.7E-4 |
| L5 | 2.9E-3 |
| L6 | 4.2E-3 |
| L7 | 2.1E-3 |
| L8 | 6.6E-7 |
| L9 | 1.2E-4 |
| L10 | 1.2E-3 |
| L11 | 4.7E-3 |
| L12 | 1.3E-3 |

Figure 5.10: Residual values for the ToC converged solution.

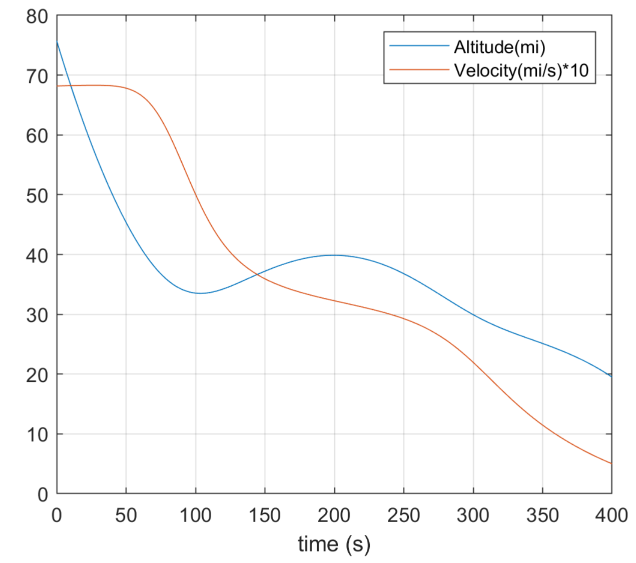


Figure 5.11: Solution to the ToC optimal guidance problem (altitude/velocity).

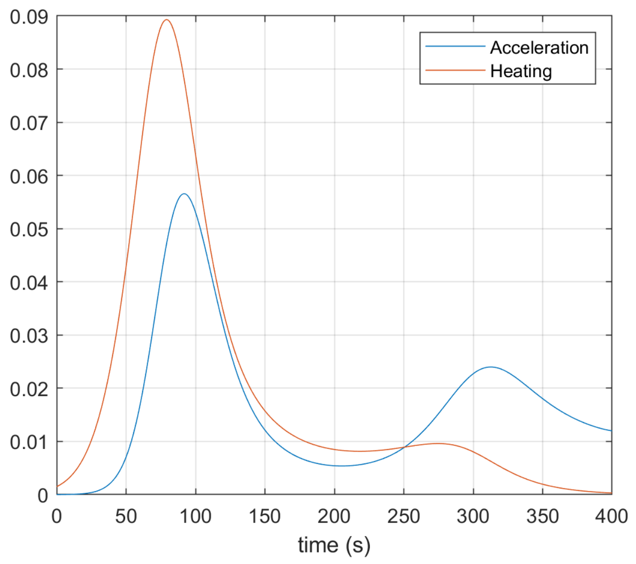


Figure 5.12: Solution to the ToC optimal guidance problem (Cost function components).

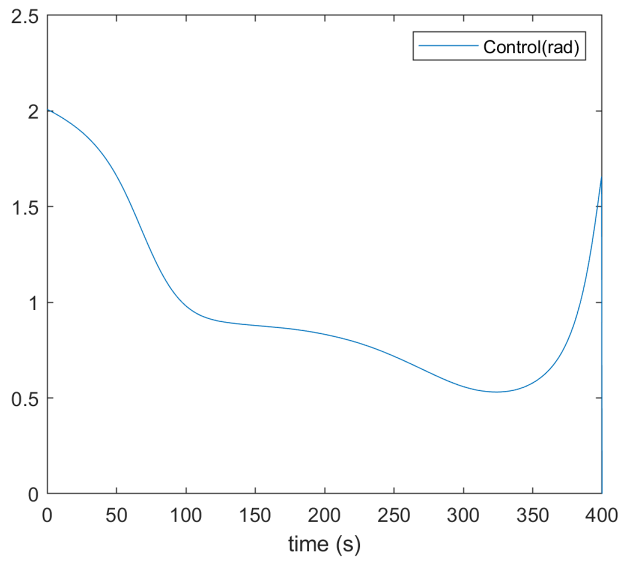


Figure 5.13: Solution to the ToC optimal guidance problem (Control).

As seen in Figures 5.11-5.13, the flight trajectory correlates well with the initial ode45 propagation. The residual equations were weighted with respect to their magnitudes to ensure that convergence was achieved at approximately the same magnitude for each $L$ function. Additionally, increasing the number of basis functions improves the accuracy, but requires an increase in discretization, resulting in a longer computation time. The reduction in computation time is necessary when attempting to resolve the nonlinear least-squares solution for a real-time guidance update.

### 5.2.4 Real-time Updating with ToC

In the following chapter, the ToC process will be used in a real-time guidance simulation. The initial trajectory calculation will be found offline, with subsequent updates found by treating the current state as the new initial state, and initializing the optimization sequence from the previous $\boldsymbol{\xi}$. These updates will occur continuously, providing a new guidance path for the vehicle in response to perturbations caused by the environment. Figure 5.14 shows a simple example of the recalculation of the trajectory derived in Section 5.2.2, where the vehicle state is perturbed from the original trajectory by 50 meters in the y and z directions.

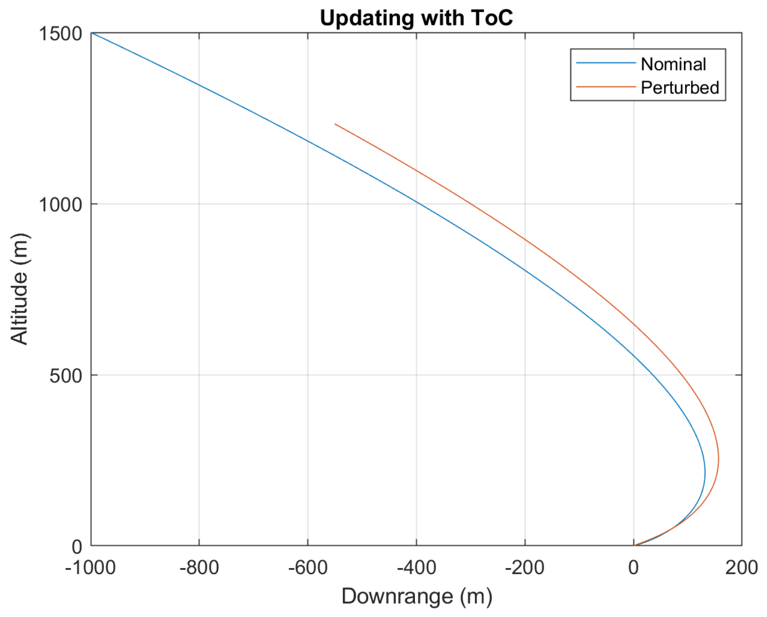


Figure 5.14: Example of a real-time trajectory update using ToC.

# 6. TEST CASES

Thus far, this thesis has discussed the structure of the simulation environment and the development of various mathematical models used to simulate the entry vehicle dynamics. Two main contributions of this thesis are the development of the GAM and the application of ToC to solve an optimal control problem. The test cases discussed in this section will provide a verification of these contributions as well as a presentation of STEAD as a stand-alone simulation and analysis tool.

## 6.1 Test Case Setup

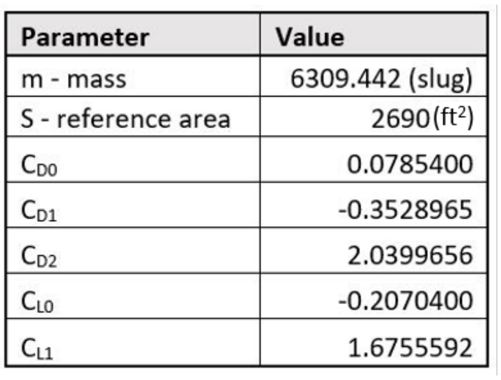
The project will perform two test cases: a Martian entry and a Titan entry. The atmospheric models will provide the perturbations to the vehicle dynamics, resulting in a need for a guidance update. The main method of perturbing the trajectory, wind gusts, are provided by the atmospheric models. Each test case will also be used to show the performance of the guidance system in the different atmospheres associated with each planet. The test case will compare the nominal optimal guidance solution with the open-loop simulation of the solution in STEAD. Additionally, the real-time guidance controller will be applied to the vehicle and compared to both the nominal and open-loop results. Of particular interest is the comparison of the position error for the open-loop and closed-loop systems.

Each simulation is run on an HP Omen X computer with an Intel i7-7820HK 2.90GHz processor and 32GB of RAM. The graphics system and MATLAB are run in a Windows environment, with the SpaceCRAFT server executing in a Linux terminal, Bash on Ubuntu on Windows.

### 6.1.1 Test Vehicle

The design of a new lifting body entry vehicle concept was outside the scope of this thesis. Therefore, the test cases will use the Space Shuttle as the base vehicle. Although not designed for the atmospheres of Mars and Titan, the vehicle serves as a place holder for a user defined model. The vehicle characteristics, such as mass, wing surface area, and aerodynamic coefficients, given in

Figure 6.1, are used in the dynamics propagation System. The optimal guidance solution used the following vehicle parameters, including the approximated $C_L$ and $C_D$ equations in Eqs 5.50 and 5.51:



| Parameter | Value |
|---|---|
| m - mass | 6309.442 (slug) |
| S - reference area | 2690 ($ft^2$) |
| $C_{D0}$ | 0.0785400 |
| $C_{D1}$ | -0.3528965 |
| $C_{D2}$ | 2.0399656 |
| $C_{L0}$ | -0.2070400 |
| $C_{L1}$ | 1.6755592 |

Figure 6.1: Test vehicle (Space Shuttle) parameters.

### 6.1.2 Optimal Guidance Setup

The lifting body test cases use a modified version of the equations of motion described in Eqs. 5.42-5.47, with the addition of angle of attack ($\alpha$) and thrust ($T$) as control variables as illustrated in Figure 6.2.

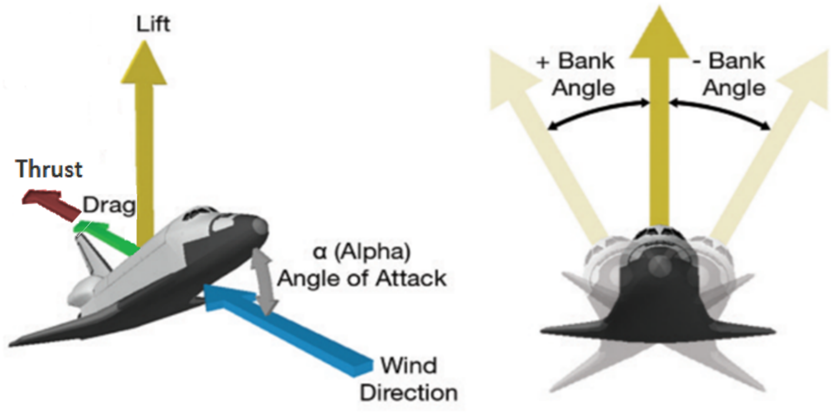


Figure 6.2: Diagram of test vehicle control variables.

These modifications to the optimal control problem change the velocity differential equation to

$$\dot{V} = -(D+T) - \frac{\mu sin\gamma}{r^2} \tag{6.1}$$

with $C_L$ and $C_D$ varying with the relationship in Eqs. 5.50 and 5.51. The "Thrust" variable is modeled as an applied acceleration in the same direction as the drag force. In theory, this acceleration could be achieved through a propulsion module or high drag device, such as a parachute.

For the test cases, the optimal guidance problem will seek to minimize the amount of thrust, or fuel, required by the entry vehicle:

$$J = \frac{1}{2}\int_{t_0}^{t_f} T^2 dt \tag{6.2}$$

As with bank angle control, the angle of attack and thrust control variables can be expressed in terms of the states and costates of the system:

$$T = \lambda_V \tag{6.3}$$

$$\alpha = \frac{c_{L1} cos\beta \lambda_\gamma - c_{D1}\lambda_V V - c_{L1}\lambda_\psi sin\beta sec\gamma}{2c_{D2}\lambda_V V} \tag{6.4}$$

These relationships are substituted into the equations of motion as shown in the previous chapter.

### 6.1.3 Initial Guess of Trajectory

As discussed in Chapter 5, the ToC guidance solution is initialized with an offline propagation of the system. To find this first candidate solution, the project uses DIDO, a 3$^{rd}$ party MATLAB toolbox developed by Elissar Global. DIDO implements an algorithm based on pseudospectral optimal control theory to find a solution to the OCP that meets the necessary conditions [20]. This software has been used successfully for real-world control problems, such as the ISS zero-propellant maneuver [21]. The toolbox requires the state differential equations, cost function, and boundary conditions as inputs, with the variables scaled and balanced to ensure the problem is well-posed.

To validate the DIDO output, the toolbox was used to provide a solution to the Space Shuttle entry problem discussed in Reference [22]. The results of this successful validation are shown in Appendix A.

For each test case, the equations of motion and cost function are consistent, with boundary conditions dependent upon the orbital radius and velocity of the entry vehicle at the start of the trajectory. The goal of the test vehicle guidance system is to reach the designated point where the astronaut would take over manual control for the landing sequence. The terminal boundary conditions for each trajectory will include a longitude/latitude location at a specified altitude, velocity, and flight path angle.

## 6.2 Mars Entry

The Martian atmospheric environment is much less dense than that of Earth. As such, the Space Shuttle would not be the optimal choice for an entry vehicle. To increase the maneuverability of the test vehicle, the reference area was increased by 50 percent, with the new thrust control ability. It is expected that the lack of atmospheric drag will require the guidance method to rely heavily on the deceleration provided by the thrust control. For the STEAD simulation, the Mars GRAM was used to provide the atmospheric properties. The initial conditions were based on the entry problem defined in Reference [22], scaling the starting altitude by the radius of Mars and setting the velocity equal to slightly less than orbital speed. The following initial and final conditions were applied to the guidance problem:

$$\begin{aligned}
r(t_0) &= 140000 \ ft & r(t_f) &= 5000 \ ft \\
\theta(t_0) &= 0^o & \theta(t_f) &= 40^o \\
\phi(t_0) &= 0^o & \phi(t_f) &= 2.5^o \\
V(t_0) &= 12000 \ ft/s & V(t_f) &= 250 \ ft/s \\
\gamma(t_0) &= -1^o & \gamma(t_f) &= -5^o \\
\psi(t_0) &= 0^o & \lambda_\psi(t_f) &= 0
\end{aligned}$$

with planetary constants for Mars shown in Figure 6.3.

| Parameter | Value |
|---|---|
| μ | 1.51246961e15 $(ft^3/s^2)$ |
| R | 11076771.7 (ft) |
| k | 2.74595e-5 (1/ft) |
| $\rho_0$ | 3.88102e-5 $(slug/ft^3)$ |

Figure 6.3: Mars planetary and atmospheric parameters.

Using the candidate solution from the DIDO toolbox, the following nominal optimal trajectory, Figure 6.4, was calculated. The nominal trajectory is defined as the vehicle path when no perturbing influences are applied.

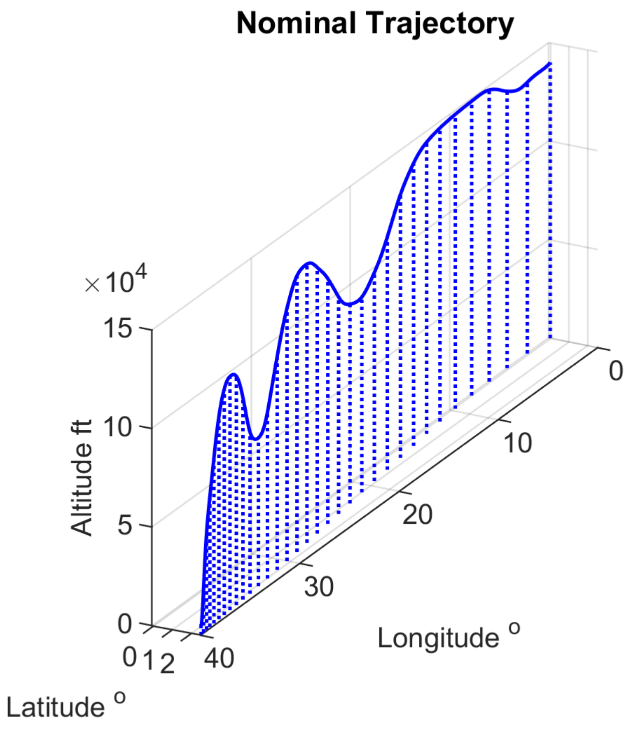


Figure 6.4: Mars - Nominal entry trajectory.

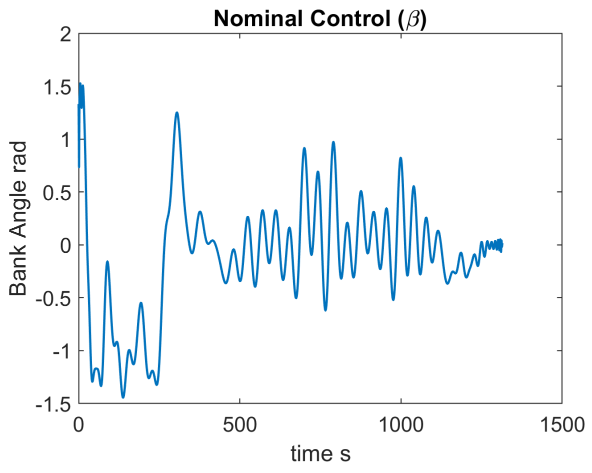


Figure 6.5: Mars - Nominal Control ($\beta$).

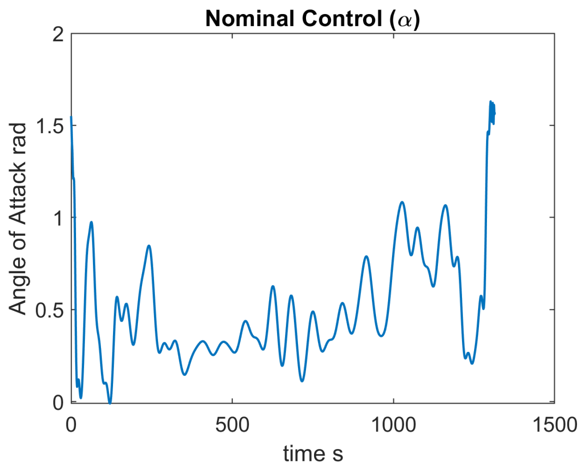


Figure 6.6: Mars - Nominal Control ($\alpha$).

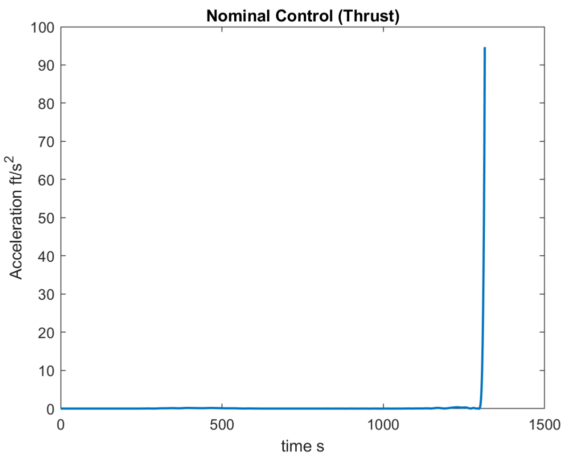


Figure 6.7: Mars - Nominal Control ($T$).

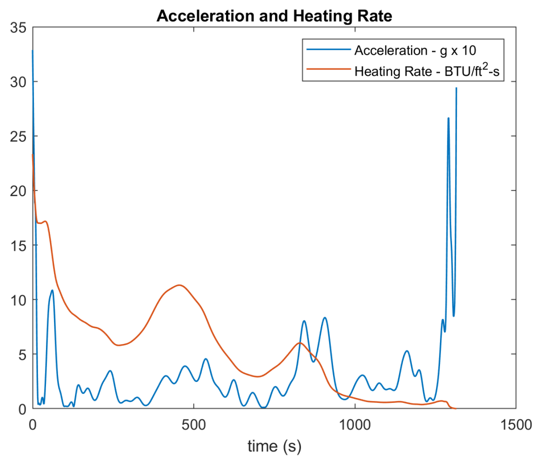


Figure 6.8: Mars - g-load and heating rate.

From the guidance output in Figures 6.5-6.7, it can be seen that the vehicle bank angle oscillates with a high frequency in an attempt to bleed off energy during the entry. The angle of attack remains large throughout most of the trajectory to produce a greater drag force. As expected, the vehicle commands a large thrust application to reduce the vehicle speed to the desired final condition, although it is interesting to note that the majority of the thrust control is applied in the last few seconds of the trajectory. Figure 6.8 gives the acceleration and heating experience by the vehicle throughout the trajectory. The individual state plots are shown in Appendix B.

Next, the vehicle was simulated in STEAD with the open-loop guidance commands. The perturbing forces, Figures 6.9 and 6.10, in the simulation environment will be the crosswind (EW wind), headwind (NS wind), and density fluctuations. The wind components and simulation density are outputs of the Mars GRAM, where as the model output is described by the density function used in the optimal guidance formulation (Eq. 5.52).

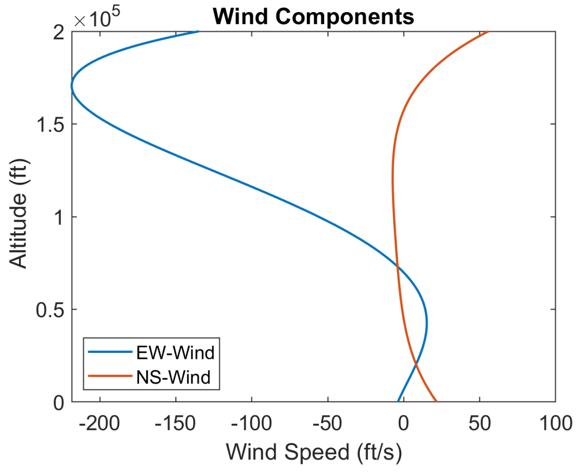


Figure 6.9: Mars GRAM wind speeds.

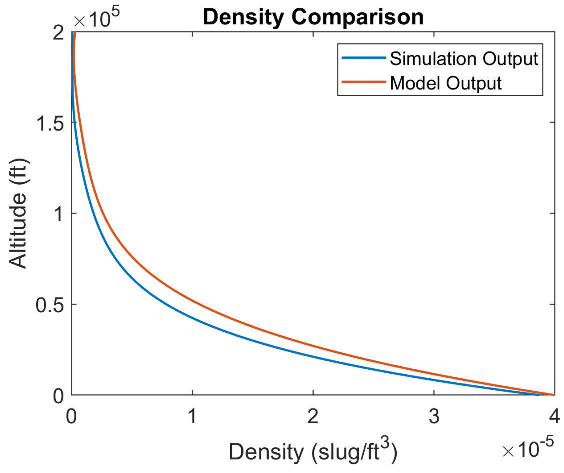


Figure 6.10: Mars GRAM density comparison.

The final 20 seconds of the trajectory are shown in Figure 6.11 to visualize the error in the final position of the vehicle.

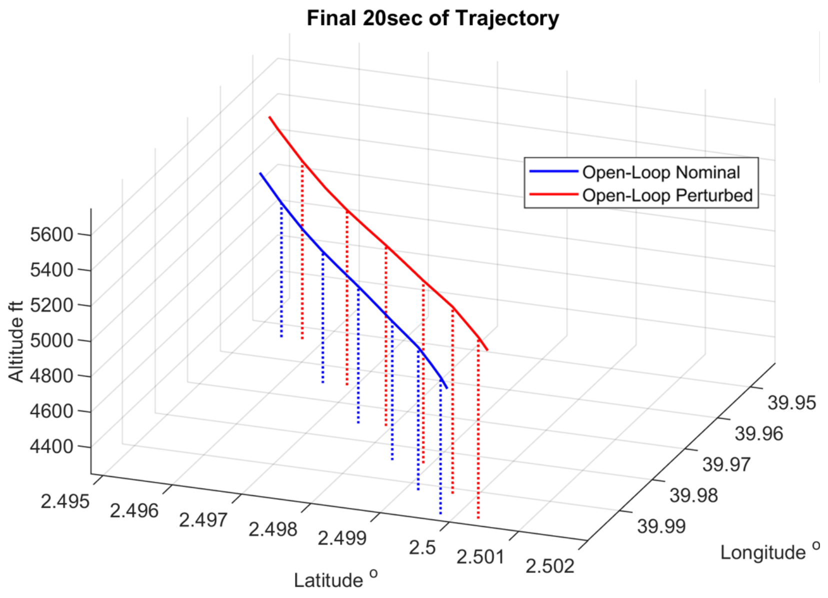


Figure 6.11: Mars - Nominal and perturbed trajectories.

Due to the low density of the Martian atmosphere, the wind speeds have a small effect on the trajectory of the vehicle. With the same atmospheric conditions, the test vehicle was simulated with the ToC methodology providing a real-time guidance update. The final 20 seconds of the trajectory were again plotted in Figure 6.12 to show the difference between the nominal, open-loop, and closed-loop solutions.

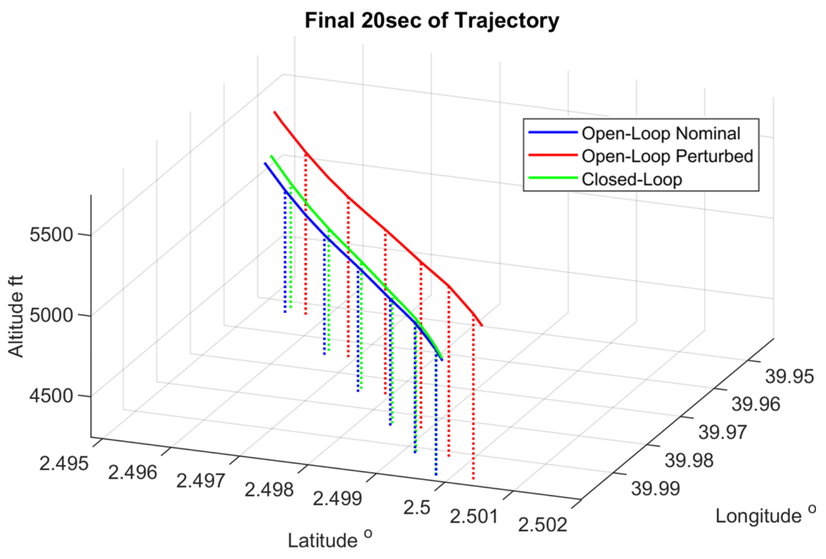
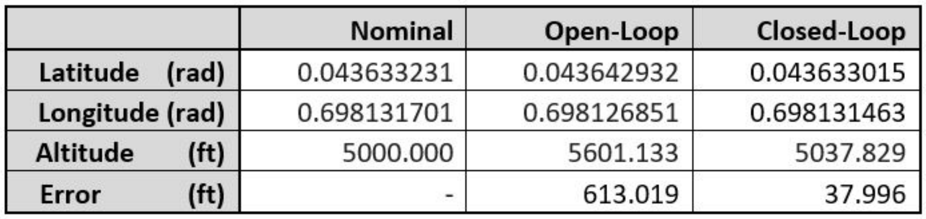


Figure 6.12: Mars - Nominal, open-loop, and closed-loop trajectories.

By inspection, the closed-loop system reduces the error in the final state of the vehicle through the guidance updates. The numerical results for the latitude, longitude, and altitude are shown in Figure 6.13. The error is calculated by finding the distance from the nominal solution.



| | Nominal | Open-Loop | Closed-Loop |
|---|---|---|---|
| **Latitude (rad)** | 0.043633231 | 0.043642932 | 0.043633015 |
| **Longitude (rad)** | 0.698131701 | 0.698126851 | 0.698131463 |
| **Altitude (ft)** | 5000.000 | 5601.133 | 5037.829 |
| **Error (ft)** | - | 613.019 | 37.996 |

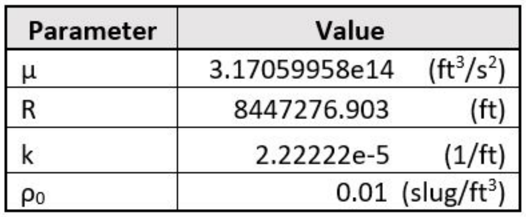
Figure 6.13: Mars - Final state of nominal, open-loop, and closed-loop simulations.

## 6.3 Titan Entry

The Titan entry test case utilizes the GAM model developed by this project. The Titan atmosphere is approximately five times as dense as Earth's, providing larger aerodynamic forces. With this increase in density comes a larger perturbation due to crosswinds, allowing for a demonstration of the real-time guidance update performed by the ToC algorithm. It would be expected that the thrust control required for the trajectory would be very small due to the magnitude of the aerodynamic forces. Since the atmosphere of Titan extends far from the surface, the initial altitude was set higher than the scaled value used in the Mars test case. The following initial and final conditions were applied to the guidance problem.

$$
\begin{aligned}
r(t_0) &= 200000 \ ft & \qquad r(t_f) &= 5000 \ ft \\
\theta(t_0) &= 0^o & \qquad \theta(t_f) &= 30^o \\
\phi(t_0) &= 0^o & \qquad \phi(t_f) &= 5^o \\
V(t_0) &= 6000 \ ft/s & \qquad V(t_f) &= 250 \ ft/s \\
\gamma(t_0) &= -1^o & \qquad \gamma(t_f) &= -5^o \\
\psi(t_0) &= 0^o & \qquad \lambda_\psi(t_f) &= 0
\end{aligned}
$$

with planetary constants for Mars shown in Figure 6.14.



| Parameter | Value |
|---|---|
| μ | 3.17059958e14 ($ft^3/s^2$) |
| R | 8447276.903 (ft) |
| k | 2.22222e-5 (1/ft) |
| $\rho_0$ | 0.01 (slug/$ft^3$) |

Figure 6.14: Titan planetary and atmospheric parameters.

Using the candidate solution from DIDO, the following nominal trajectory was found and plotted in Figure 6.15.

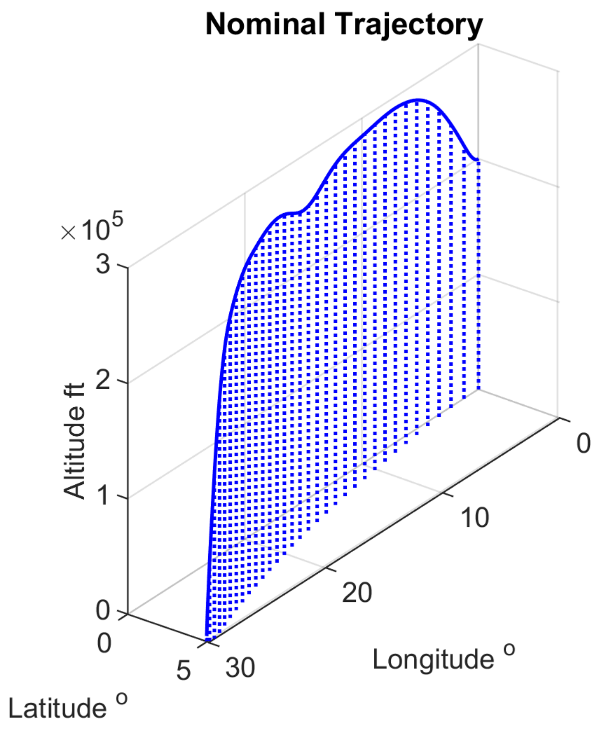


Figure 6.15: Titan - Nominal entry trajectory.

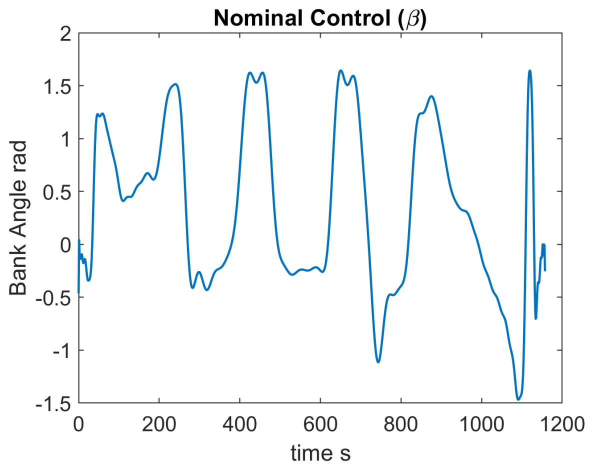


Figure 6.16: Titan - Nominal Control ($\beta$).

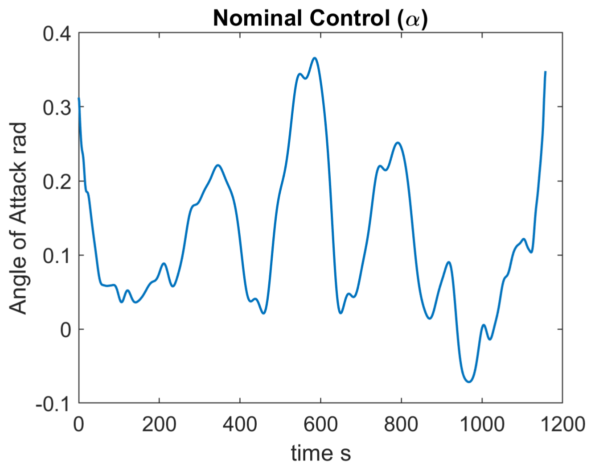


Figure 6.17: Titan - Nominal Control ($\alpha$).

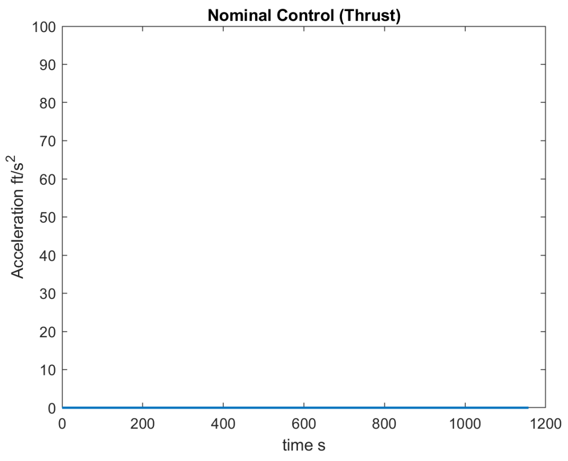


Figure 6.18: Titan - Nominal Control ($T$).

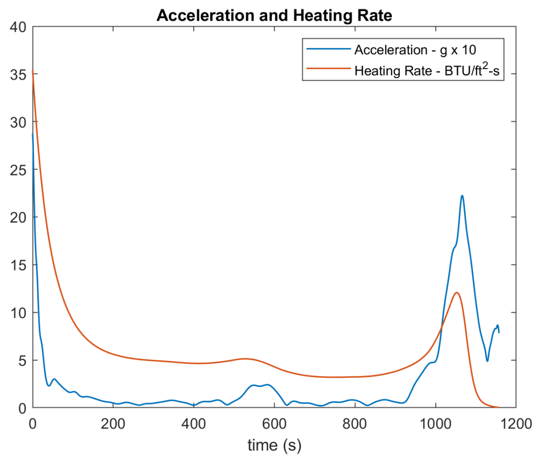


Figure 6.19: Titan - g-load and heating rate.

From the guidance output in Figures 6.16-6.18, the vehicle bank angle oscillates with a much lower frequency than the Mars test case, due to most of the energy bleeding off through the higher drag forces caused by the dense atmosphere. The angle of attack remains relatively small throughout most of the trajectory. As expected, the vehicle uses no thrust acceleration to meet the final conditions of the optimal guidance problem. Figure 6.19 gives the acceleration and heating experience by the vehicle throughout the trajectory. The individual state plots are shown in Appendix C.

The vehicle was then simulated in STEAD with the open-loop guidance commands. The perturbing forces in the simulation environment will be the crosswind and density fluctuations, both of which should have a larger effect on the trajectory due to the denser atmosphere.

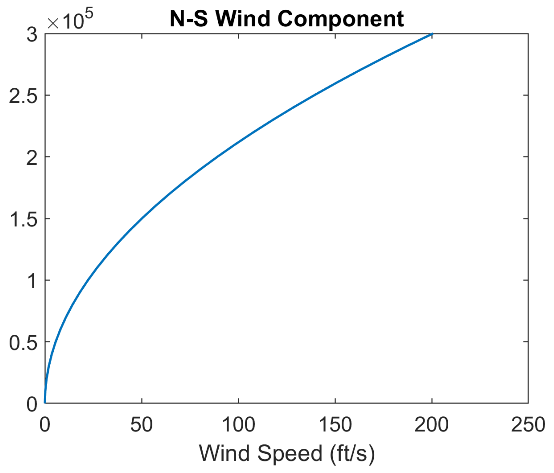


Figure 6.20: Titan GAM wind speed.

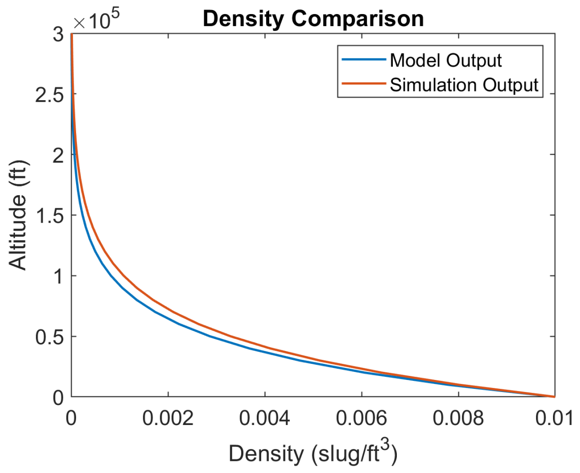


Figure 6.21: Titan GAM density comparison.

The final 20 seconds of the nominal and open-loop trajectories are shown in Figure 6.22.

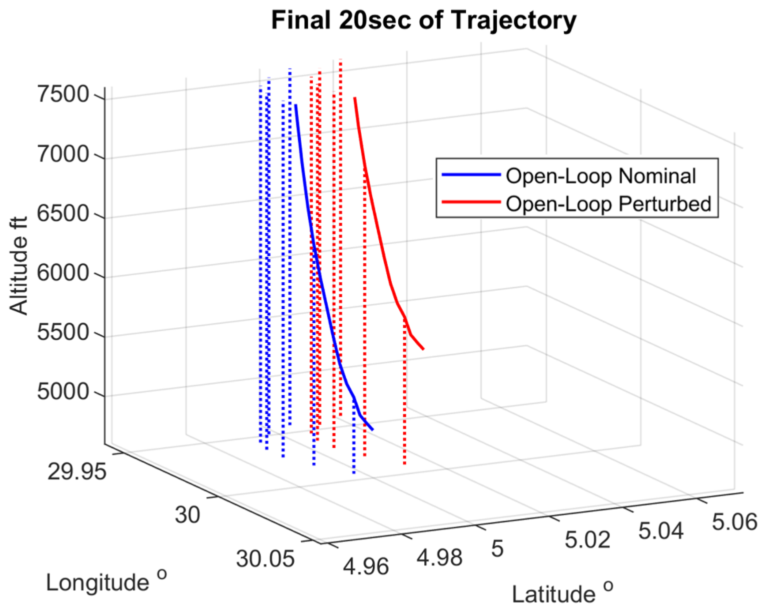


Figure 6.22: Titan - Nominal and perturbed trajectories.

The open-loop simulation resulted in a much larger error than the Martian test case due to the magnitude of the aerodynamic forces. The closed-loop solution was simulated in STEAD with the ToC methodology providing the guidance updates. The final section of the trajectory is shown in Figure 6.23 to compare the ending state of the closed-loop to the previous solutions.

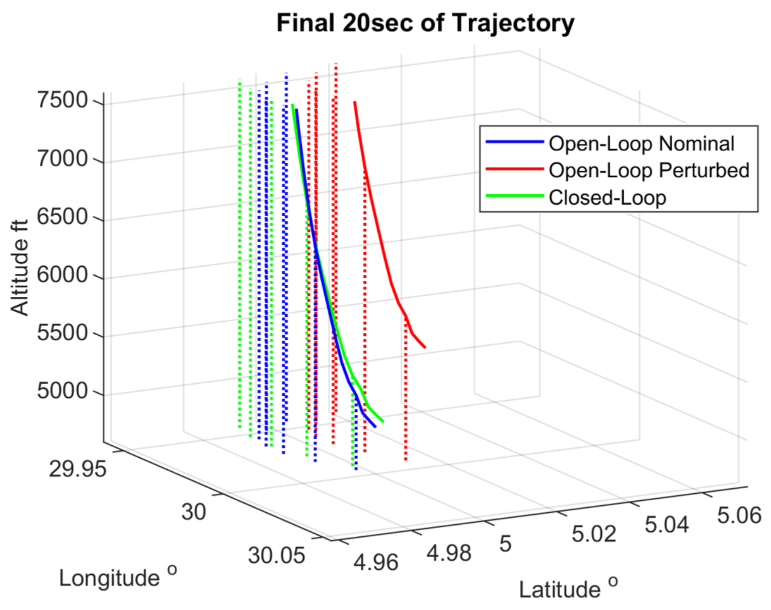
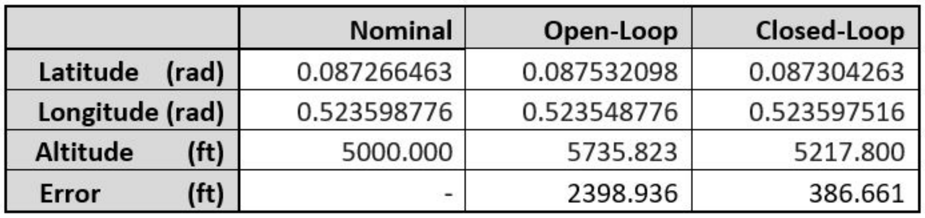


Figure 6.23: Titan - Nominal, open-loop, and closed-loop trajectories.

The numerical results for the latitude, longitude, altitude, and absolute error are shown in Figure 6.24.



| | Nominal | Open-Loop | Closed-Loop |
|---|---|---|---|
| Latitude (rad) | 0.087266463 | 0.087532098 | 0.087304263 |
| Longitude (rad) | 0.523598776 | 0.523548776 | 0.523597516 |
| Altitude (ft) | 5000.000 | 5735.823 | 5217.800 |
| Error (ft) | - | 2398.936 | 386.661 |

Figure 6.24: Titan - Final state of nominal, open-loop, and closed-loop simulations.

## 6.4 Discussion

The goal of STEAD is to serve as a simulation platform for the testing of lifting body entry vehicles. The Mars and Titan test cases provided two unique environments for the test vehicle, showcasing the ability of the simulation tool to accurately represent different environments. Additionally, the application of the open-loop guidance and ToC closed-loop guidance algorithms demonstrate the ability to simulate the dynamics and control of a spacecraft.

In both test cases, the ToC method improved the accuracy of the entry trajectory when perturbations were applied. The Mars test case had a very small open-loop error due to the low density of the atmosphere, therefore, the real-time updating only decreased the final position error by 575.023 ft. The Titan test case had a much larger open-loop error, providing a greater need for a real-time guidance algorithm. The ToC methodology reduced the error of the final position by 2012.275 ft.

To maintain a high level of accuracy, the ToC method used a set of 50 Chebyshev basis functions with the total flight time discretized into 1000 timesteps. Due to the increase in basis functions, the nonlinear least-squares algorithm converged in an average time of 12.249 seconds. With the current setup of the SpaceCRAFT platform, the decision was made to return an updated guidance command every 15 seconds. Since the update computations take a non-negligible amount of time, the resulting output contains outdated guidance commands. These commands are neglected, but when switching from the previous command to the updated, a discontinuity can occur in the controls. To alleviate this issue, it is necessary to develop an interpolation function that can transition the guidance commands to the updated values.

# 7. SUMMARY AND CONCLUSION

## 7.1 Conclusion

The objectives of this thesis were the development of a software platform to accurately simulate the entry, descent, and landing of a spacecraft. The project added key functionality to the core SpaceCRAFT platform by creating an interface for integrating FORTRAN models into a simulation and connecting a live MATLAB script, providing real-time inputs to the entry vehicle. The addition of the NASA GRAMs will give future users access to an accurate atmospheric environment for testing new mission concepts.

The SPICE system outlined in Chapter 3 provides the simulation environment with accurate planetary states, as shown by the solar eclipse verification case. SPICE has many capabilities that can be utilized by the SpaceCRAFT platform, including coordinate frame definitions, which would allow the placement of user defined reference frames in the simulation. This would be particularly useful for placing a surface fixed frame at the landing site of the entry vehicle. The Generic Atmospheric Model (GAM) developed by this thesis was successful in calculating the surface temperature of Mars using real-world processes. The simulation of the atmospheric temperature, however, was not accurately reproduced. By using the relationships discussed in Chapter 4, the GAM was able to produce realistic parameter values, but this method is not adequate for reproducing atmospheric properties, including mixing of a dynamic atmosphere, at high altitudes.

From the Mars and Titan test cases, the Theory of Connections (ToC) method has the mathematical ability to provide real-time guidance, however, there are aspects of the current state of the method that make it infeasible to use in practice. The initial ToC solution is highly sensitive to the initial candidate solution provided to the nonlinear least-squares algorithm. It was proposed that a decreasing gradient vector for the coefficient vector would be satisfactory for convergence, but in this thesis, the solution would not converge unless given a candidate solution. Currently, there is not a proposed method for applying constraints on the states and controls of the system.

Without these constraints on the bank angle or fuel consumption, the ToC method could produce infeasible results. It was discovered that by proposing a constrained candidate solution from DIDO, the ToC method became psuedo-constrained, as it converged to a trajectory near the initial input. This psuedo-constraint was somewhat maintained throughout the trajectories for the Mars and Titan test cases.

Current research is in progress to develop the method of adding state and control constraints into the ToC methodology. Once this is achieved, ToC can be used as a closed-loop guidance algorithm. Since each successive iteration of the least-squares process produces a new set of basis function coefficients, the method is able to use the previous result as the new initial guess. As such, the DIDO software is only required for the initial proposed solution. Although the solutions provided in this thesis focus on correcting the trajectory when the current state is perturbed, it could be possible to also change the final state, to produce a new landing target.

The STEAD system has successfully shown its ability to accurately model the real-world environment, and simulate the dynamics and guidance of a lifting body entry vehicle. The structure of the tool meets the requirements set forth by the project, particularly the modularity of the simulation environment. There are many areas of the system that would benefit from more complex models and further testing to improve the user experience.

## 7.2 Future Work

This thesis has demonstrated the core capabilities of STEAD and the SpaceCRAFT platform, but there are still many areas in which the simulation tool can be improved. The main focus of this project was the entry and descent phases of EDL, but the landing sequence simulation is the area that can benefit most from the use of virtual reality. Previous work in the ASTRO Lab has contributed to the development of various user control interfaces including the translational and rotational hand controllers used on-board the ISS, shown in Figure 7.1, and different types of displays, including computer monitors and tablets shown in Figure 7.2.

displays.

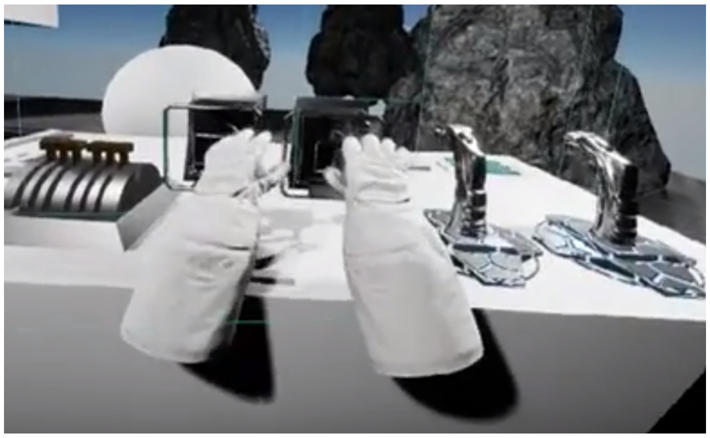


Figure 7.1: VR hand controllers to be integrated into the landing simulation.

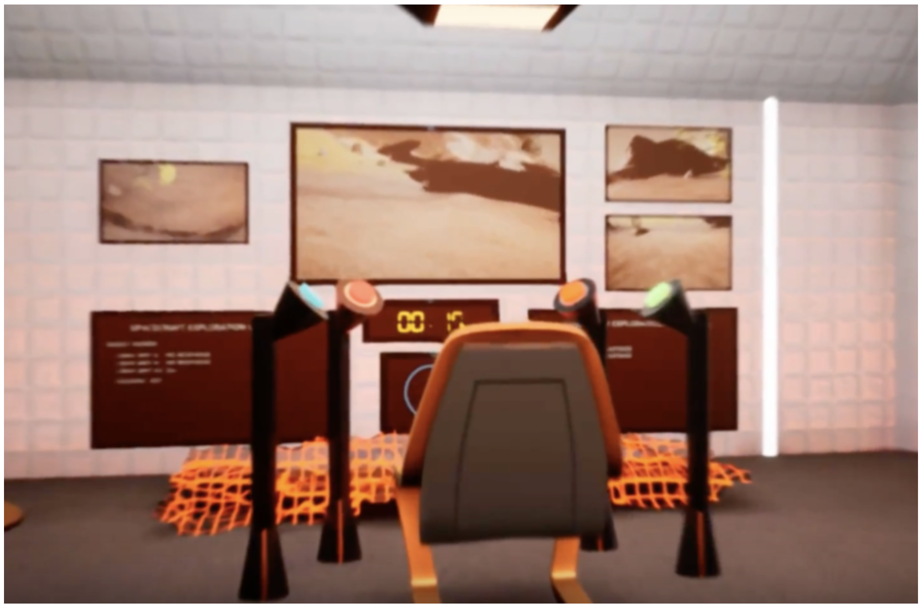


Figure 7.2: VR displays to be integrated into the landing simulation.

The landing simulation could be used to improve the cockpit layout and visual aids for the pilot. The computer-aided design team has previously designed interior layouts with integrated displays which would be a starting point for developing the cockpit of the entry vehicle. Visual aids could include targeting boxes or spline paths for the pilot to follow, with relevant data applied to the displays.

## REFERENCES

[1] M.R. Zwack and P.D. Dees. "Program to Optimize Simulated Trajectories II (POST2) Surrogate Models for Mars Ascent Vehicle (MAV) Performance Assessment." NASA/TM-2017-219842.

[2] Jody Davis, Scott Striepe, Robert Maddock, Andrew Johnson, Stephen Paschall II. "POST2 End-to-End Descent and Landing Simulation for ALHAT Design Analysis Cycle 2."

[3] Rafael A. Lugo, Jeremy D. Shidner, Richard W. Powell. "Launch Vehicle Ascent Trajectory Simulation Using the Program to Optimize SImulated Trajectories II (POST2)." AAS 17-274.

[4] John M. Penn, Alexander S. Lin, "The Trick Simulation Toolkit: A NASA/Open source Framework for Running Time Based Physics Models."

[5] Eddie J. Paddock, Alexander Lin, Keith Vetter. "Trick: A Simulation Development Toolkit." AIAA Modeling and Simulation Technologies Conference and Exhibit, Aug 2003.

[6] H. L. Justh. "Mars Global Reference Atmospheric Model 2010 Version: Users Guide." 2014.

[7] Spencer, David, "Mars Pathfinder Entry, Descent, and Landing Reconstruction", Journal of Spacecraft and Rockets, vol. 36-3, 1999. pp. 357-366.

[8] Thomas E. Moore. "Space Shuttle Entry Terminal Area Energy Management." NASA Technical Memorandum 104744, November 1991.

[9] Wang, Z., and Grant, M. "Constrained trajectory optimization for planetary entry via sequential convex programming." Journal of Guidance, Control, and Dynamics, 1-13, 2017.

[10] Wang, Z., and Grant, M. "Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach. IEEE Transactions on Aerospace and Electronic Systems." 2018

[11] Mortari, D. "The Theory of Connections. Part 1: Connecting Points." AAS 17-255, 2017 AAS/AIAA Space Flight Mechanics Meeting Conference, San Antonio, TX, USA, 5-9 February 2017

[12] Furfaro, R. and Mortari, D. "Least-squares Solution of a Class of Optimal Guidance Problems," AAS 18-362, 2018 AAS/AIAA Astrodynamics Specialist Conference, Snowbird, UT, August 19-23, 2018.

[13] N. McHenry, R. Hogan, M. Coen, M. Holub, B. Morrell, G. James, G. Chamitoff, "SpaceCRAFT: A Virtual Reality Sandbox and Open-Source Mission Simulation Platform." SpaceOps 2018, Marseilles, France, May 2018.

[14] N. McHenry, R. Hogan, E. Abdou, M. Coen, B. Morrell, G. Chamitoff, "Virtual Reality Multi-User Space System Mission Design and Simulation: Engagint the Public Through Open-Source Collaboration." International Astronautical Congress, Adelaide, Australia, Sept 2017.

[15] Acton, C.H. "Ancillary Data Services of NASA's Navigation and Ancillary Information Facility." Planetary and Space Science, Vol. 44, No. 1, pp. 65-70, 1996.

[16] Wilbur L. Hankey, L. Earl Miller, Stephen J. Scherr. "Use of Quaternions in FLight Mechanics." Flight Dynamics Laboratory, Air Force Wright Aeronautical Laboratories. Mar 1984.

[17] D'Souza, C. and D'Souza, C. "An optimal guidance law for planetary landing." Guidance, Navigation, and Control Conference, 1997, p. 3709.

[18] Mortari, D. "Least-squares Solutions of Linear Differential Equations." In Proceedings of 27th AAS/AIAA Space Flight Mechanics Meeting Conference, San Antonio, TX, USA, 5-9 February 2017.

[19] Walton E. Williamson, "Optimal Three Dimensional Reentry Trajectories for Apollo-type Vehicles." University of Texas, Austin, TX. January 1970.

[20] Ross, I. M. and Karpenko, M., "A Review of Pseudospectral Optimal Control: From Theory to Flight," Annual Reviews in Control, Vol.36, 2012, pp.182-197.

[21] N. Bedrossian, S. Bhatt, M. Lammers, L. Nguyen, and Y. Zhang, "First Ever Flight Demonstration of Zero Propellant Maneuver Attitude Control Concept," Proceedings of the AIAA Guidance, Navigation, and Control Conference, 2007, AIAA 2007-6734.

[22] Betts, John T., "Pracital Methods for Optimal Control and Estimation Using Nonlinear Programming," Society for Industrial and Applied Mathematics, 2010. pp. 603.

[23] Wilbur L. Hankey, "Flight Mechanics", Re-Entry Aerodynamics, AIAA Education Series.

[24] Wilbur L. Hankey, "Re-Entry Flight Regimes", Re-Entry Aerodynamics, AIAA Education Series, pp. 3-22.

[25] D'Souza, C. and D'Souza, C., "An optimal guidance law for planetary landing". In Guidance, Navigation, and Control Conference, 1997, (p. 3709).

[26] Christopher J. Cerimele, Edward A. Robertson, and Joseph A. Garcia. "A Rigid Mid Lift-to-Drag Ratio Approach to Human Mars Entry, Descent, and Landing", AIAA Guidance, Navigation, and Control Conference, AIAA SciTech Forum, (AIAA 2017-1898)

[27] Dwyer Cianciolo, Alicia M. "Entry, Descent and Landing Systems Analysis Study: Phase 1 Report." July 2010.

[28] Linden, Van der. "DASMAT-Delft University Aircraft Simulation Model and Analysis Tool: A Matlab/Simulink Environment for Flight Dynamics and Control Analysis." Dec 1998.

[29] Barton, Jeffery D. "Use of a High-Fidelity UAS Simulation for Design, Testing, Training, and Mission Planning for Operation in Complex Environments." Jan 2011.

[30] Jill L. Prince, Prasun N. Desai, and Eric M. Queen. "Mars Phoenix Entry, Descent, and Landing Simulation Design and Modeling Analysis." Journal of Spacecraft and Rockets, Vol. 48, No. 5, September-October 2011.

[31] K. P. Bollino and I. M. Ross, "A pseudospectral feedback method for real-time optimal guidance of reentry vehicles," 2007 American Control Conference, New York, NY, 2007, pp. 3861-3867.

# APPENDIX A

## DIDO Output Validation from Reference [20]

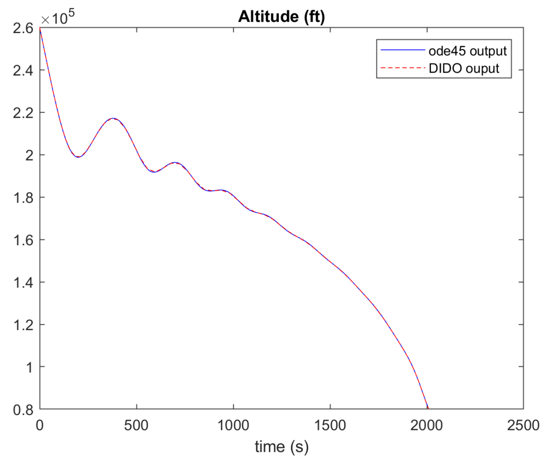


Figure A.1: DIDO verification (Altitude).

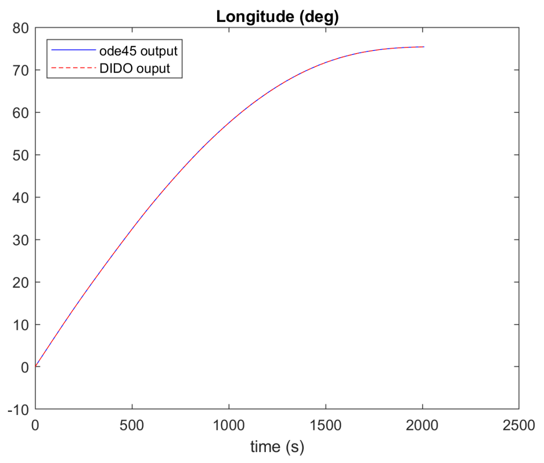


Figure A.2: DIDO verification (Longitude).

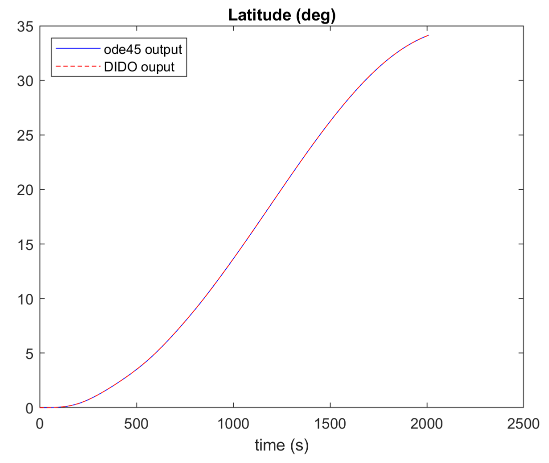


Figure A.3: DIDO verification (Latitude).

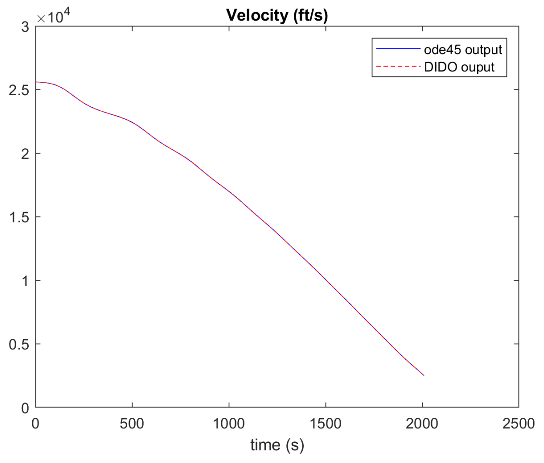


Figure A.4: DIDO verification (Velocity).

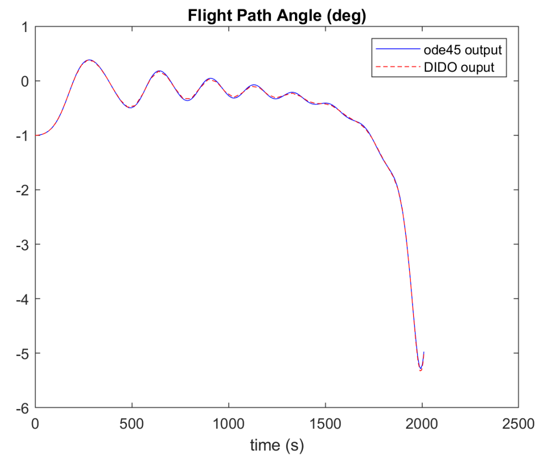


Figure A.5: DIDO verification (Flight Path).

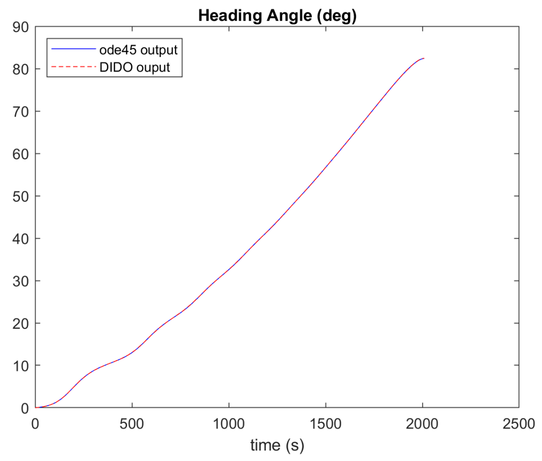


Figure A.6: DIDO verification (Heading).

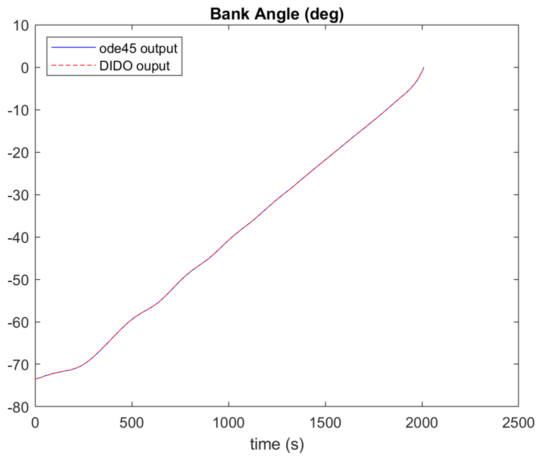


Figure A.7: DIDO verification (Control - $\beta$).

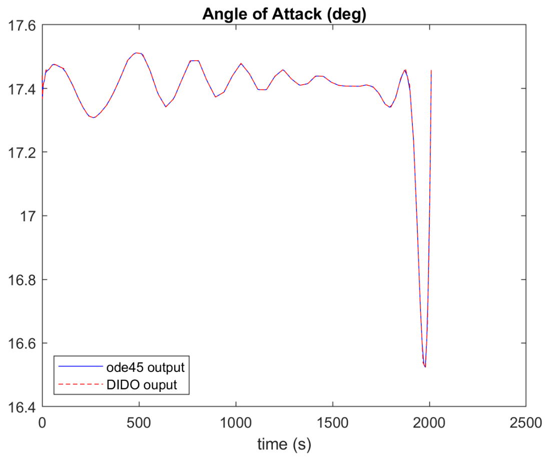


Figure A.8: DIDO verification (Control - $\alpha$).

Appendix A shows the results from implementing the DIDO toolbox for the entry problem defined in Reference [20]. The output matches the results given in the reference chapter, showing the feasibility of using DIDO to find the initial candidate solution for the Theory of Connections algorithm.

# APPENDIX B

## Nominal State and Control History for Mars Test Case

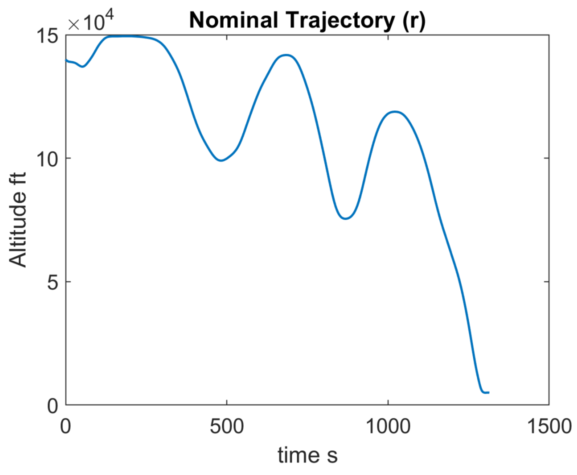


Figure B.1: Mars Trajectory (Altitude).

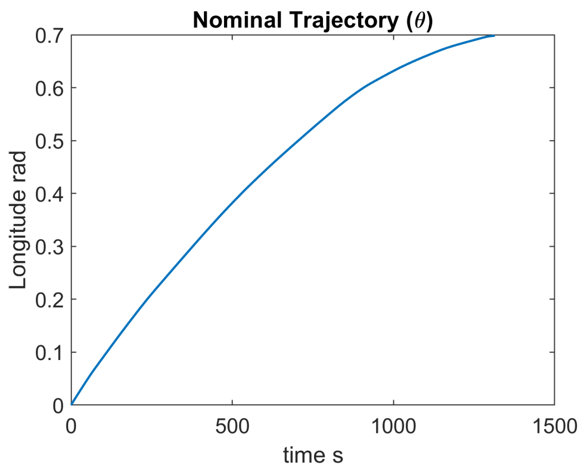


Figure B.2: Mars Trajectory (Longitude).

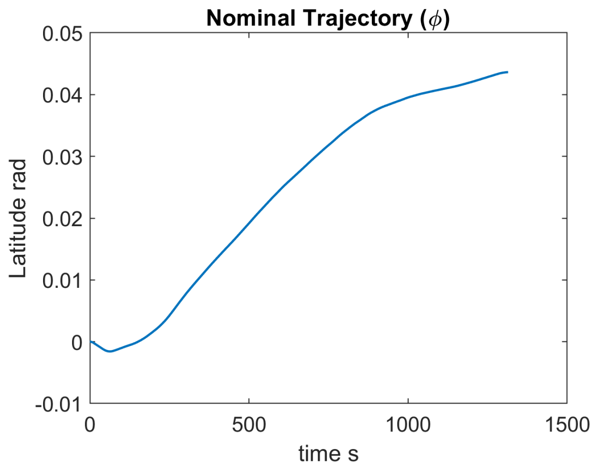


Figure B.3: Mars Trajectory (Latitude).

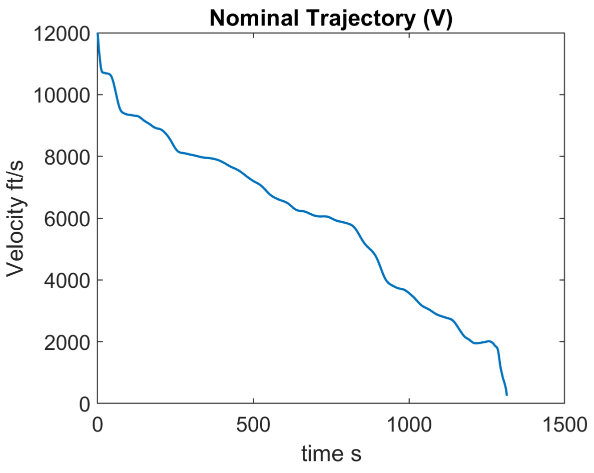


Figure B.4: Mars Trajectory (Velocity).

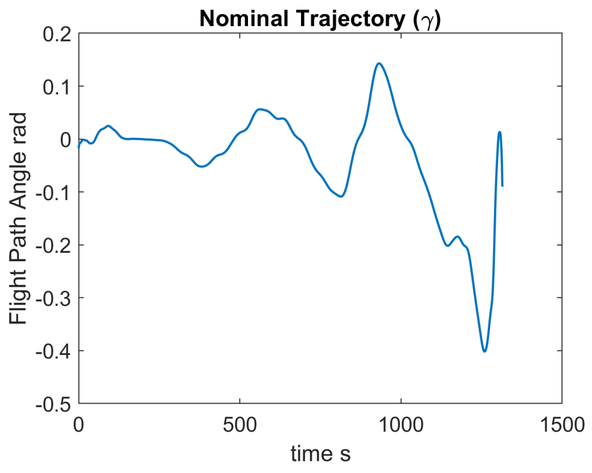


Figure B.5: Mars Trajectory (Flight Path).

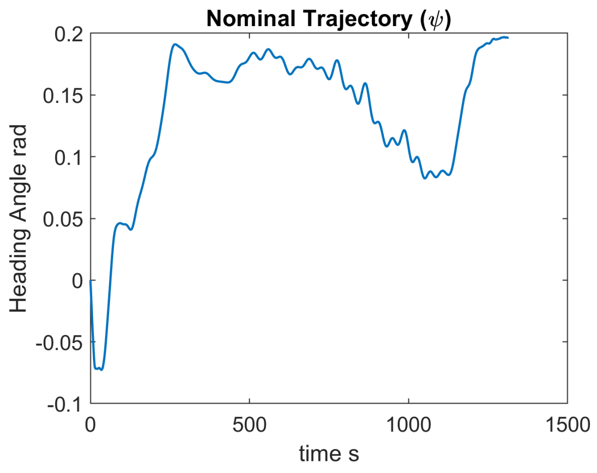


Figure B.6: Mars Trajectory (Heading).

Appendix B shows the initial trajectory solved by the Theory of Connections algorithm for the Mars test case. This initial trajectory is considered the nominal, as it does not take into account environmental disturbances. The control history shown are the inputs in the open-loop simulation performed in Chapter 6.

# APPENDIX C

## Nominal State and Control History for Titan Test Case

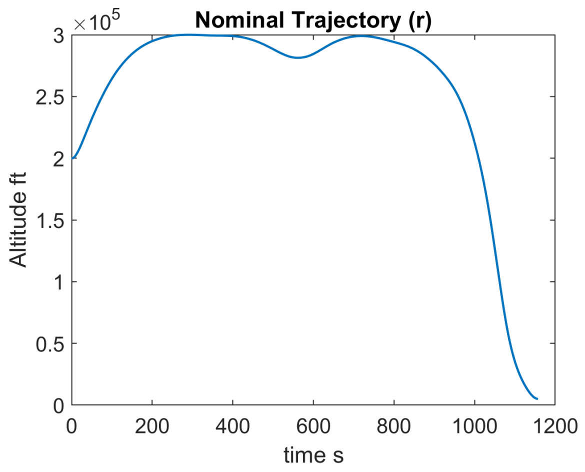


Figure C.1: Titan Trajectory (Altitude).

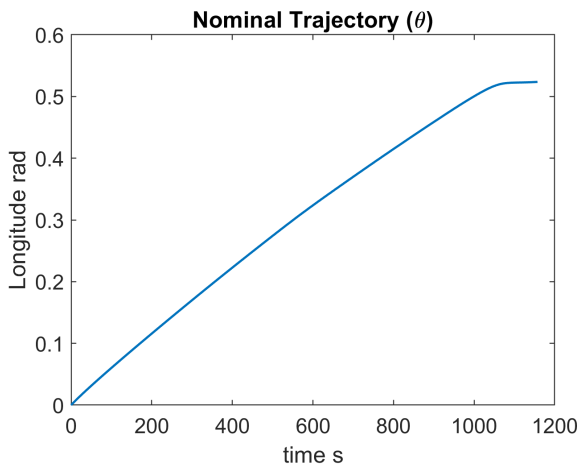


Figure C.2: Titan Trajectory (Longitude).

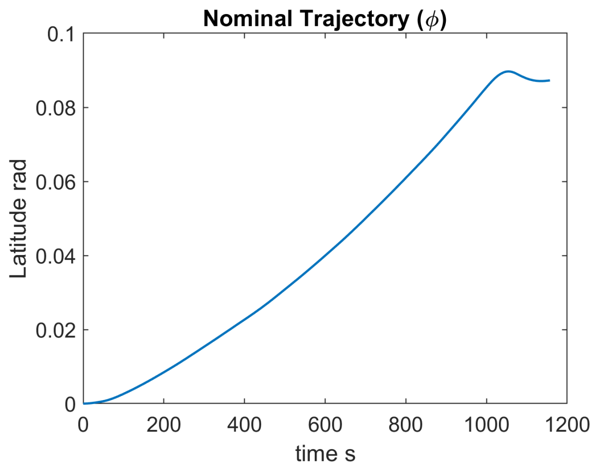


Figure C.3: Titan Trajectory (Latitude).

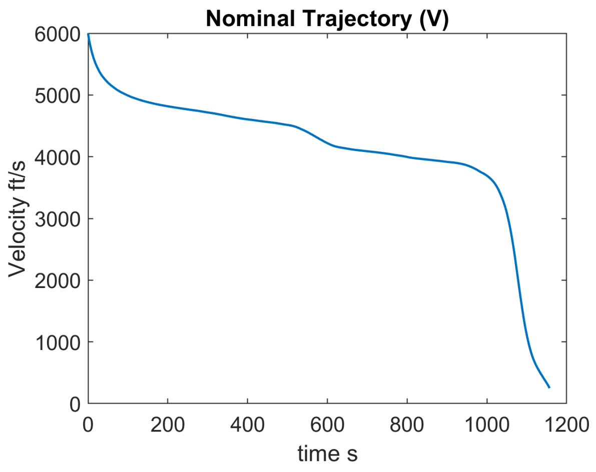


Figure C.4: Titan Trajectory (Velocity).

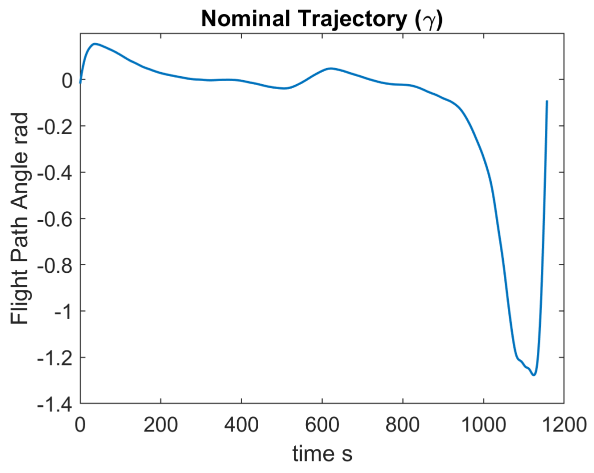


Figure C.5: Titan Trajectory (Flight Path).

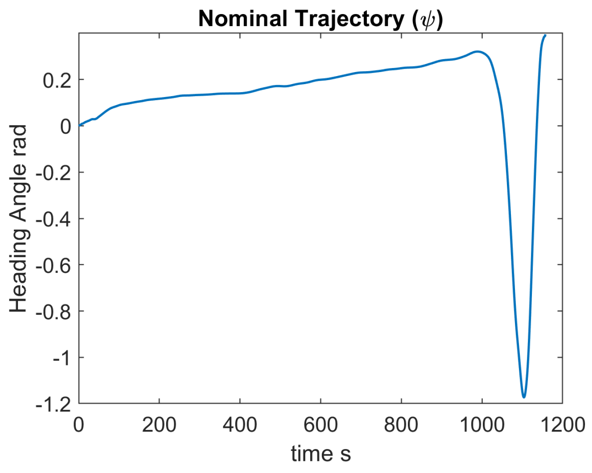


Figure C.6: Titan Trajectory (Heading).

Appendix C shows the initial trajectory solved by the Theory of Connections algorithm for the Titan test case. This initial trajectory is considered the nominal, as it does not take into account environmental disturbances. The control history shown are the inputs in the open-loop simulation performed in Chapter 6.