Pitch-Control Predictor-Corrector

and

Neural Network Ascent Guidance

Adam L. Cowling

Contents:

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Aerospace Engineering

Dr. Christopher D. Hall, Chair
Dr. Craig A. Woolsey
Jeffrey S. Robinson

May 4, 2009
Blacksburg, Virginia

Keywords: Ascent Guidance, Trajectory Optimization, Neural Network,
Predictor-Corrector

Pitch-Control Predictor-Corrector
and
Neural Network Ascent Guidance

Adam L. Cowling

(ABSTRACT)

A pitch-control predictor-corrector ascent guidance algorithm has been developed and evaluated for a rocket-based upper stage of a two-stage-to-orbit launch vehicle. Detailed descriptions of the predictor-corrector algorithm and a neural network loop modification are given. The mission requirement is insertion into a stable 50×100 nmi orbit at 375,000 ft altitude, coasting toward apogee at a positive inertial flight path angle. Three degree-of-freedom trajectory analysis is performed using the Program to Optimize Simulated Trajectories (POST2). Results of Monte Carlo simulations including uncertainties in atmosphere, thrust, aerodynamics and initial state are presented and compared to trajectories optimized for maximum injected weight.

This thesis was completed to fulfill a graduate research assistantship at the National Institute of Aerospace under the sponsorship of NASA Langley's Hypersonics Project and the Vehicle Analysis Branch.

Acknowledgments

I owe a tremendous debt of gratitude to Jeff Robinson of NASA Langley Research Center for having confidence in me and going out of his way to provide me with the opportunity of a lifetime. His mentorship has helped me grow as an aerospace engineer and as a person.

I would not be where I am today if not for the opportunities given to me by John Korte, Chuck Leonard and everyone in the Vehicle Analysis Branch. John and Chuck have offered a wealth of assistance and leadership while enduring a very difficult year. A special thanks goes to Jim Beatty, Paul Tartabini, and Scott Striepe for the painful grilling they gave me on my research.

I would like to thank Dr. Chris Hall and Dr. Craig Woolsey for serving on my committee and for being two of the best teachers I have ever had. I would also like to thank Dr. Robert Lindbergh, Dr. Bernie Grossman, and the faculty and staff of the National Institute of Aerospace for providing a truly unique educational experience.

I cannot express how much I thank my parents for supporting me in everything, both emotionally and financially. I thank my brother, Will, for always setting the bar high, and I thank my future sister, Katie, for the endless discussions of quantum physics and anti-gravity.

Finally, I would like to thank my fiancée, Nicky, for her love, support, and patience (and also for not killing me when I decided to get my Master's).

“**WIN** - **W**hats **I**mportant **N**ow: To make change; To influence; To use this moment to be better than the last; To achieve greatness in all aspects of your life;

WIN at home. **WIN** at school. **WIN** at business. **WIN** at life.”

— Bud Foster

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

3-DOF	3-Degree-Of-Freedom
CBAERO	Configuration Based Aerodynamics code
CG	Center of Gravity
ECI	Earth-Centered Inertial frame
ECR	Earth-Centered Rotating frame
GRAM99	1999 NASA/MSFC Global Reference Atmospheric Model
IGM	Iterative Guidance Mode
J2000	Epoch at noon January 1, 2000
LEO	Low Earth Orbit
LOX	Liquid Oxygen oxidizer
MECO	Main Engine Cut-Off
NASA	National Aeronautics and Space Administration
NN	Neural Network
NPSOL	Nonlinear Programming, Systems Optimization Laboratory
O/F	Oxidizer/Fuel ratio
OTIS	Optimal Trajectories by Implicit Simulation
P/C	Predictor-Corrector
PEG	Powered Explicit Guidance

POST2	Program to Optimize Simulated Trajectories 2
RP-1	Refined kerosene rocket fuel
T/O	Targeting and Optimization
TSTO	Two-Stage-To-Orbit

Subscripts

i	Generic counter
I	Measured in ECI frame
R	Measured in ECR frame

English

a_i	Layer i output
A_e	Nozzle exit area
$\underline{A}_{AI}$	Aerodynamic acceleration in the ECI frame
$\underline{A}_{GI}$	Gravitational acceleration in the ECI frame
$\underline{A}_{TI}$	Thrust acceleration in the ECI frame
b_i	Bias for layer i
C_A	Axial force coefficient
C_D	Drag Coefficient
C_L	Lift Coefficient
C_M	Moment Coefficient
C_N	Normal force coefficient
dtg	Guidance interval
e	Error between network output and target output
f_i	Transfer function for layer i
f_i'	Derivative of transfer function for layer i
F	Cost functional

g	Gradient
h_{gd}	Geodetic altitude
h_{ma}	Mean apogee altitude
h_{mp}	Mean perigee altitude
H	Hessian matrix
I	Identity matrix
I_{sp}	Specific impulse
IW_i	Input weights for layer i
J	Jacobian matrix, J^T denotes transpose
$J2 - J8$	Second through eighth harmonics in the gravity potential function
k	Total number of equality and inequality constraints
K_j	Direction field samples for Runge-Kutta integration
lb	Lower constraint bounds for NPSOL
$LW_{i,j}$	Layer weights for layer i, coming from layer j
M_θ	Moment in the pitch plane
n	Length of χ vector
P	Neural network input
P_t	Training set input
q	Size of input set P
$\bar{q}$	Mean value
$\dot{\underline{r}}_I$	Time derivative of the translational position in the ECI frame
R_{Ee}	Equatiorial radius for Earth
R_{Em}	Mean radius of Earth
R_{Ep}	Polar radius for Earth
$\mathbf{R}^i$	Real set of dimension i

t	Time
t^*	Time at optimization event
t_f	Final time
tgo	Time-to-go
T	Neural network output
T_t	Training set target output
T_{vac}	Vacuum thrust
u	Controls
u_L	Lower control bound
u_U	Upper control bound
ub	Upper constraint bounds for NPSOL
v	Velocity magnitude
$\underline{v}_I$	Translational velocity in the ECI frame
w	Individual neural network weight
x	State variables
$\dot{x}$	Time derivative of the state variables
z_i	Intermediate output for layer i

Greek

α	Angle of attack
β	Constraint set for NPSOL
γ	Flight path angle
δ_i	Sensitivity of layer i
δ_i^*	Sensitivity processed by layer i weights
ϵ	Parameter vector in the equations of motion
η	NPSOL variable set

θ_{GIM}	Gimbal angle
θ_R	Relative Euler pitch angle
λ	Longitude
μ	Levenberg-Marquardt performance parameter
μ_E	Gravitational constant for Earth
σ	Standard deviation
τ	Constraint tolerances for NPSOL
ϕ_{eq}	Equality end conditions
ϕ_{gd}	Geodetic latitude
ϕ_{in}	Inequality end conditions
χ	Concatenated vector of weights and bias
ψ	Velocity azimuth
ω_E	Rotation rate of Earth

Chapter 1

Introduction

1.1 Motivation and Problem Definition

The National Aeronautics and Space Administration (NASA) has a desire to employ Monte Carlo analysis methods on trajectory simulations for conceptual design of launch vehicles. Monte Carlo techniques for trajectory analysis require one of two options: targeting and optimization (T/O) of every Monte Carlo case or a guidance algorithm that steers the vehicle toward the exit conditions without significant performance degradation. While T/O provides optimal trajectories, it can be time intensive to produce a full set of optimized cases. A guidance algorithm provides a more real-world trajectory with guidance impacts. In order to reach the desired exit conditions, the guidance algorithm must be able to handle uncertainties in propulsion performance, mass properties, aerodynamics, and atmospheric properties.

The goals of this research are:

- Develop candidate guidance algorithms for use in conceptual design tools.
- Design the algorithms to utilize the Program to Optimize Simulated Trajectories 2

(POST2) with potential for implementation into the source code.

- Ensure that algorithms employ the framework of real vehicle guidance systems.
- Reduce the computation time associated with traditional Monte Carlo analysis.
- Compare the algorithms to T/O for a two-stage-to-orbit (TSTO) rocket-based upper stage trajectory simulation.

The trajectory of interest for comparing the algorithms is the ascent of a TSTO upper stage to a 50×100 nmi orbit about Earth, with an optimization objective of maximum injected weight at orbit insertion. The initial state is defined just after stage separation to be 28.5° latitude, 287° longitude, and 135,000 ft altitude with a velocity of 6,000 ft/s magnitude, 92.7° azimuth, and 7.5° flight path angle. Orbit insertion is to occur above a geodetic altitude of 375,000 ft and at a positive inertial flight path angle for a coast to apogee. The 50×100 nmi orbit is selected as a standard low earth orbit (LEO) phasing orbit, allowing for circularization or other maneuvers at apogee, though this study only addresses the trajectory until main engine cut-off (MECO) at insertion.

1.2 Software

Trajectory simulation is a data-intensive field requiring considerable computational ability and careful selection of software. A focus of this study was to use industry standard programs in developing a time-efficient simulation. Software was required for trajectory simulation, optimization, and data transport. In addition, it was desired to have dispersion in the atmosphere to provide a more realistic simulation. The programs selected for this study are detailed below.

1.2.1 Program to Optimize Simulated Trajectories 2 (POST2)

All trajectory simulations were generated using the Program to Optimize Simulated Trajectories 2[1,2,3,4] (POST2), Version 1.1.6G, a standard within the aerospace field with extensive use and validation. POST2 is a targeting and optimization (T/O) trajectory simulation program for generalized point masses in the vicinity of a single rotating, attracting planet. POST2 has capabilities for 3-degree-of-freedom (3-DOF) translational and 6-DOF rotational/translational simulations, as well as multiple vehicles. The original 3-DOF POST was first developed in the 1970s as a Space Shuttle trajectory simulation and optimization code. The addition of a 6-DOF capability as well as updates to vehicle modeling, targeting, optimization and simulation provided the remaining improvements leading to POST2. More than 1000 input and output variables are available, allowing for many options for the design trajectory. POST2 is a flexible code that can accommodate a wide range of problem definitions because the trajectory is decomposed into a sequence of events, or phases. This sequence can consist of any number of events, which are categorized as primary, secondary, or roving. Up to ten events are monitored at one time to allow more customization in the design. Every simulation in POST2 comes from the combination of five basic components: the planet module, the vehicle module, the trajectory simulation module, the trajectory auxiliary calculations module, and the T/O module.

The planet module contains both an oblate model with $J2 - J8$ harmonics and a spherical model allowing special input of spherical harmonic gravitational data. The atmosphere and wind models are defined by tables of pressure, density, temperature, speed of sound, and windspeeds. Several available standard atmospheres are programmed into POST2, as well as a spectral turbulence model.

The vehicle module consists of mass properties, propulsion, aerodynamics, aeroheating, and steering models. Mass properties can be defined in several different ways including propellant weights, multiple stage weights, and payload weights. The propulsion model can accommodate up to 25 rocket, jet, or ramjet engines per event. Flow-rate and thrust equations are included for added variability in propulsion design. POST2 defines aerodynamic characteristics through mono- and multi-variate table input of axial/normal, lift/drag, and side force coefficients. POST2 has an array of aeroheating calculation options including table lookup, Chapman, and maximum centerline. The steering (guidance) model provides a wide range of options for body rates, aerodynamic angles, inertial Euler angles, and relative Euler angles. Guidance computation can include table lookup, polynomials, piecewise linear functions and closed-loop linear feedback. In addition to standard routines, the user can code models directly into the general open-loop guidance subroutine.

The trajectory simulation module controls the integration of the equations of motion. This module monitors and executes the event sequence and contains multiple standard numerical integration methods, including Runge-Kutta and Krogh. The trajectory auxiliary calculations module includes calculation methods for a broad range of output variables. Keplerian conic parameters and crossrange/downrange are among the variables calculated in this module.

The T/O module contains several methods to perform targeting with or without optimization. More than 400 variables are available as independent (control) and dependent (target) variables. The module determines the set of controls that maximize or minimize an optimization variable such that the target is met. Target, control, and optimization variables are defined at events specified by the user. POST2 can handle both equality and inequality constraints. The optimization algorithms available in POST2 are a projected gradient algorithm, an accelerated projected gradient algorithm, and NPSOL, a nonlinear programming package developed at Stanford University[5].

The generality of the POST2 code provides a foundation for developing a wide range of problems, thus resulting in a large number of inputs and outputs. The user can control the level of POST2 output by choosing from a standard print block or specific variables, as well as echoing the input file and printing event blocks. Basic input/output is text-based, and a variety of output converters are available to provide compatability with standard programs such as Microsoft Excel[6] and MATLAB[7].

1.2.2 Nonlinear Programming Package (NPSOL)

The optimization method of choice is NPSOL[5], a nonlinear programming package from the Systems Optimization Laboratory at Stanford University. NPSOL is made up of a set of subroutines for minimizing an objective function subject to a set of constraints. Generically, these constraints may be variable bounds, linear constraints, or smooth nonlinear constraints. In the POST2 implementation, the constraints for NPSOL are all treated as nonlinear. NPSOL constraints are defined by lower and upper bounds, as well as a feasibility tolerance, therefore it is common practice in POST2 to define the target variables with both a range and a tolerance. The basis for NPSOL is a sequential quadratic programming method where each search direction is defined by the solution of a quadratic programming subproblem. The POST2 implementation is limited to a cold start, which means NPSOL derives the initial working set from the variables and constraints, not a user-defined estimate. NPSOL operates under the assumption that the objective function and all constraints are second-order smooth, which is often not the case with a trajectory simulation. However, if discontinuities exist in isolation at a distance away from the solution, NPSOL will typically succeed. As a result of this violation of the NPSOL definition, local and global minima are not readily distinguished. When this situation occurs, NPSOL often requires user intervention to redefine some parameters of the optimization to avoid non-optimal targeted solutions. Adjusting the initial values or bounds of independent variables in POST2 often helps correct such situations.

1.2.3 Global Reference Atmosphere Model 1999 (GRAM99)

The NASA Marshall Space Flight Center Global Reference Atmospheric Model (GRAM99) serves as a design reference atmosphere offering full global, altitude, monthly, and seasonal coverage. GRAM99 is a physics-based engineering tool designed to provide dispersions in typical atmospheric parameters. GRAM99 is a system of three individual models based on empirical data. Each model represents a different altitude range. For altitudes between 0 and 88,580 ft, the Global Upper Air Climatic Atlas is used. Several sources, including the Middle Atmosphere Program, are used for the data set from 65,600 ft to 393,700 ft. Lastly, the region above 295,000 ft is defined by the Marshall Engineering Thermosphere model. Extensive references regarding the individual models can be found in the GRAM99 manual[8]. In regions where two models overlap, GRAM99 utilizes fairing methods to ensure a smooth transition. GRAM99 allows the user to input desired initial latitude, longitude and altitude values, as well as increments for each. In addition, the user can input a specific date by year, month, day, hour, minute and second. Output of each randomly seeded GRAM99 run was automatically converted to POST2 table format by a module in written in Adaptive Modeling Language[9]. The output consisted of tables of temperature, density, pressure, speed of sound, and windspeeds as a function of geodetic altitude.

1.2.4 MATLAB and The Neural Network Toolbox

MATLAB[7] is used to drive the input/output functions of the guidance algorithms in developing POST2 input decks and searching POST2 output files. For all neural network architectures, training, and simulation, the MATLAB Neural Network Toolbox[10] is used. The Neural Network Toolbox allows the user to create custom networks or select from a variety of standard network architectures, such as feed-forward and cascade-forward. The implementation of neural network methodology into simple MATLAB functions provides the user with several options in selecting the number of layers and neurons, the training methods, the transfer functions, and the connections between layers. In addition, the toolbox features an array of training techniques including gradient descent backpropagation and Levenberg-Marquardt backpropagation. The generality of the toolbox allows simple sensitivity analyses regarding network structure to be performed.

1.3 Thesis Overview

The remainder of this thesis is organized as follows. First, a literature review is presented to explain the previous developments made in trajectory analysis, launch vehicle ascent, predictor-corrector guidance, and neural network algorithms. Next, an overview of the design trajectory with specific vehicle and simulations descriptions is provided. The simulation methodology for the optimization, predictor-corrector, and neural network algorithms is explained next, followed by Monte Carlo results for each.

Chapter 2

Literature Review

In this chapter, a review of the developments in ascent guidance, predictor-corrector-like algorithms and neural network control applications is presented. Some introduction to the possible options available for ascent guidance is given with specific interest in easily replicated, computationally fast methods. In addition, previous trajectory simulation developments involving predictive control loops are provided, followed by an in-depth review of neural network applications, architectures, and training methods.

2.1 Ascent Guidance

Ascent guidance has historically been broken down into endo- and exo-atmospheric flight regimes with the use of multiple guidance algorithms. In the interest of computation time and conceptual design, a method combining both regimes into one algorithm is preferrable. Several candidate advanced guidance methods with specific interest in time and cost have been reviewed by Hanson[11] for application in various flight phases. The methods are presented with special consideration for the X-33 reusable launch vehicle technology demonstrator. Hanson *et al.*[12] presented specific guidance designs for the X-33, including the Iterative Guidance Mode (IGM), or linear tangent steering, made popular by the Saturn rocket series. In addition, the prototype guidance methods for the X-33 were designed to account for vehicular and atmospheric dispersions. Another step was taken on the X-33 when Calise *et al.*[13] developed a guidance algorithm for 3-D optimal ascent that held applications for out of plane steering for optimization. This numerical/analytical hybrid algorithm was also developed to be customizable and not limited to just launch vehicle applications.

Most space missions are at some point compared to the Apollo missions or the Space Shuttle, especially when considering the future of launch vehicles. The Apollo program's Saturn V rocket utilized the IGM guidance system and proved its optimality and accuracy. Another major benefit to the IGM method was its generality with regard to other missions. Chandler and Smith[14] described the development of the algorithm with regard to various vehicles and applications, and Horn *et al.*[15] later applied IGM to more generalized missions. The Space Shuttle took a different path, utilizing an all encompassing algorithm called Powered Explicit Guidance[16,17] (PEG). The PEG system was designed to include provisions for all flight regimes of the Space Shuttle, and the ascent portion of PEG used a predictor-corrector guidance algorithm. In a Masters thesis by Su[18], PEG was implemented for comparison with some advanced optimal-control based algorithms. The trajectory simulations in the thesis are performed in the Optimal Trajectories by Implicit Simulation (OTIS) code. The results of the thesis suggest that while computer simulations are feasible, there remains a need for Monte Carlo simulation with complex atmospheric dispersions.

Selection of a simulation standard requires careful consideration of the techniques and methods behind the trajectory integration. Since one goal of this thesis is to provide options that are compatible with POST2, it was chosen as the trajectory simulator. Nelson[19] has performed a study comparing POST and OTIS and has found them to be in correlation on many trajectories. POST2 and its predecessors have a long history of use in Monte Carlo simulations and its generality make it an ideal code for many applications. Queen and O'Mara[20] observed satisfactory results in their implementation of GRAM atmospheres in Monte Carlo simulations performed with POST. Additionally, Hanson and Hall[21] have performed Monte Carlo simulations on Ares I using POST and provided a comprehensive list of uncertainties found in all phases of flight.

2.2 Predictor-Corrector Guidance

Predictor-Corrector (P/C) methods are a subset of closed-loop guidance algorithms, with the basic idea of a loop that predicts the remaining trajectory and corrects the control command accordingly. Aside from the P/C implementation in PEG, little has been done to expand the scope of such an algorithm. However, in the 1990s, Powell *et al.*[22,23,24] developed some algorithms in POST2 for P/C ascent guidance of launch vehicles on Earth and aerocapture and landing on Mars. The ascent guidance evaluations were for both vertical and horizontal takeoff launch vehicles. In each, trajectory analysis was performed including GRAM atmospheres. The predictor-corrector formulation in these papers does not target an orbit by apsides altitudes for sufficient geodetic altitude and positive flight path angle, but the basic concepts are readily applicable to different missions. The Mars aerocapture study was another implementation of a P/C algorithm showing the versatility in addressing multiple applications.

Additional closed-loop methods have been developed that utilize on-board computational power in calculating guidance commands. Closed-loop systems for endo-atmospheric flight have been shown by Lu *et al.*[25] to have significant advantages over typical open-loop systems. These advantages came from the use of on-board solvers, such as Lu's two-point boundary value problem solver for the inner guidance loop. Graesslin *et al.*[26] made use of nonlinear programming methods in the design of a predictive inner loop. Nonlinear programming methods are the basis for NPSOL, which solves the T/O problem using a sequential quadratic programming algorithm. In 2004, Jits and Walberg[27] presented a P/C algorithm using both inner and outer external POST2 loops for Mars aerocapture guidance. This P/C algorithm provided a basic framework for guidance that can be applied to multiple mission architectures, including ascent guidance.

2.3 Neural Networks

The theory behind neural networks (NN) has a broad range of training methods, architectures and transfer functions. Much of the research into justifying NN usage came from three papers by White *et al.*[28,29,30]. In the first, it was shown that "multilayer feedforward networks are universal approximators," suiting them to a wide range of problems. The second and third papers further asserted that such networks can learn arbitrary mappings, offering the potential to approximate a dynamic model given the right training set and network formulation. These powerful capabilities were the driving force behind the development of several NN architectures and methods.

The training of NN was born in 1986 from the work of Rumelhart *et al.*[31] in the form of backpropagation. With the backbone of the backpropagation concept formed, many variations and applications arose. Yann le Cun[32,33,34] helped develop a theoretical backpropagation framework that views backpropagation as an optimization problem. Additionally, an investigation into generalization and design strategies using backpropagation and other learning techniques was performed, resulting in a call for software and computer interface improvements. Battiti[35] presented a review on first- and second-order learning methods that showed similarities between many algorithms based on backpropagation and other methods. In 1998, le Cun *et al.*[36] revealed many tricks of the trade regarding backpropagation, including batch and stochastic training. Among the many options of NN components is the selection of transfer functions. Sigmoid transfer functions are the most popular because they are typically continuous and differentiable. Justification of the use of sigmoid transfer functions was presented by Harrington[37] and Duch and Jankowski[38]. In order to further increase computation speed, several methods were developed to promote faster learning. Among the methods was Riedmiller and Braun's[39] RPROP algorithm which evolves based on gradient sign changes. The RPROP algorithm showed better performance than standard steepest descent algorithms. Another fast-learning method is an application of the Levenberg-Marquardt algorithm designed by Hagan and Menhaj[40]. The Levenberg-Marquardt algorithm is an alternating approach that shifts between a classic Newton iteration and a gradient descent method.

The computational advantage and "intelligent" capabilities of artificial NN have made them a strong candidate for control systems. In 1992, Schaechter and Rauch[41] used NN for control applications in real-time software simulations, with an outlook for hardware implementation. As an extension of the basic NN control concept, Steinberg[42] performed an intensive literature and simulation review to investigate the potential applications of fuzzy logic and neural computing to flight control. The resulting paper provided a comprehensive outlook on future technologies, as well as an excellent bibliography. In 1997, Steck and Rokhsaz[43] advanced the idea further with the development of several examples of NN applications in the modeling of nonlinear aerodynamics and flight dynamics. While the examples showed excellent generalization and modeling results, control generalization degraded significantly beyond the training domain. Hagan and Demuth[44] later developed a tutorial on basic NN control system applications, with specific algorithms and concepts that would later be included in the MATLAB Neural Network Toolbox. The tutorial warns against "overfitting" a training set, which can cause the NN to fail to generalize future input. To avoid training set complications, Plutowski and White[45] developed a process for selecting a concise training set, and Weller and Avraam[46] provided a model for competitive selection of a training set.

Algorithms utilizing NN came to the forefront of flight control design in the early 2000s. Johnson *et al.*[47,48] produced two papers with applications of NN for launch vehicle control. The first paper presented a neural network augmentation in the design of a feedback linearization attitude controller for the X-33. The second expanded on this method developing a more robust system for reusable launch vehicles. An application specifically geared toward rapid computation and optimization for conceptual design was developed in 2005 by Qazi *et al.*[49]. The study used NN methods to incorporate trajectory optimization and near-real-time guidance. The results showed promising control characteristics with specific applications to launch vehicle closed-loop guidance.

Several advancements have been made in guidance and control, specifically in the fields of endo-atmospheric closed-loop control, predictive loop methods, and neural network applications. Much of the literature available on these topics was presented above with specific applications to the predictor-corrector and neural network guidance algorithms developed in this thesis.

Chapter 3

Trajectory Overview

In this chapter, an overview of the design trajectory is presented. First, the coordinate reference frames used in the simulation will be explained. Next, a description of the planet model, including gravitational and atmospheric properties is included. Then, the vehicle model and its aerodynamics, propulsion and other features are explained. Lastly, the nominal initial state and the final target orbit conditions are given.

3.1 Coordinate Frames

In trajectory analysis, the large number of trackable variables requires careful consideration of the frames in which variables are measured. This section presents the coordinate frames required for the trajectory simulations performed. Considerations for Earth rotation, geodesy, and body coordinates are included. All images in this section are adapted from the POST2 manuals[2,3].

3.1.1 Earth-Centered Inertial

The Earth-Centered Inertial (ECI) frame in Figure 3.1 is a Cartesian system where the Z-axis intersects the north pole, the X-axis lies in the equatorial plane and points toward the Greenwich meridian at time equal to zero, and the Y-axis completes the right-handed system. All values measured in the inertial frame are identified by the subscript I. The translational equations of motion are integrated in the ECI frame.

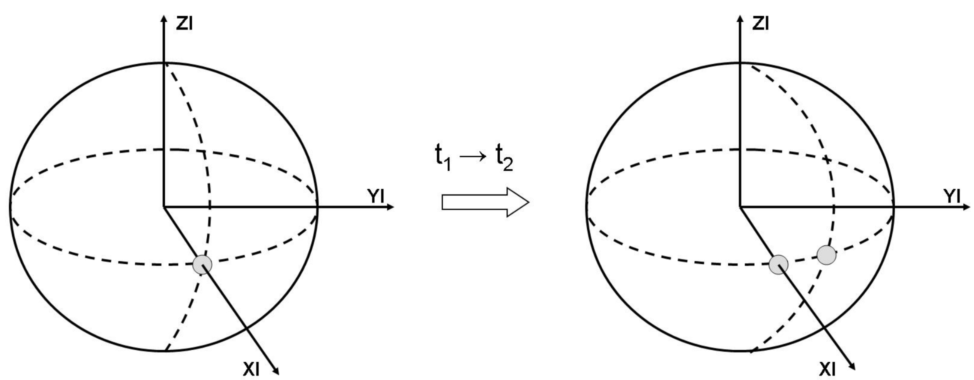


Figure 3.1: Earth-Centered Inertial frame.

3.1.2 Earth-Centered Rotating

The Earth-Centered Rotating (ECR) frame in Figure 3.2 is a Cartesian system where the Z-axis intersects the north pole, the X-axis lies in the equatorial plane and points toward the Greenwich meridian at all times, and the Y-axis completes the right-handed system. All values measured in the rotating frame are identified by the subscript R. For the state definition in this thesis, the longitude, λ_R, is measured east of the Greenwich Meridian.

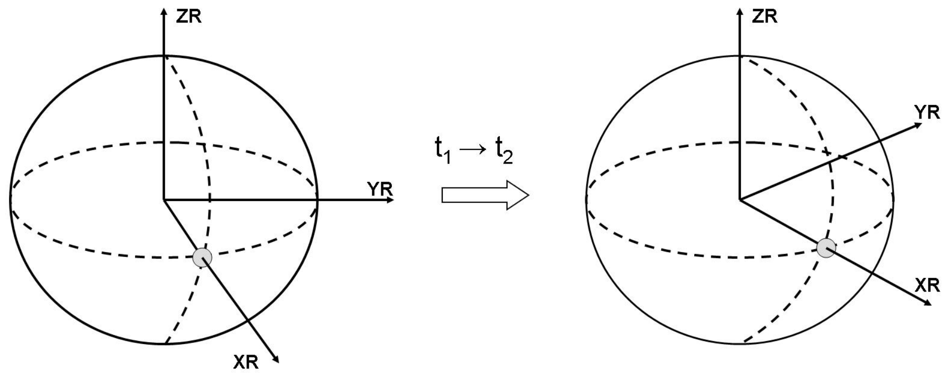


Figure 3.2: Earth-Centered Rotating frame.

3.1.3 Geodesy and the Local Horizontal

When considering an oblate planet, geodetic coordinates must be considered. The geodetic latitude and altitude are defined in Figure 3.3. The geodetic latitude, ϕ_{gd}, is the angle between the equatorial plane and a vector normal to the surface passing through the vehicle position. The geodetic altitude is, h_{gd}, is the distance between the surface and the vehicle along the vector normal to the surface. The local horizontal plane is the plane normal to the geodetic altitude vector. The local horizontal plane is illustrated in Figure 3.4. Within the

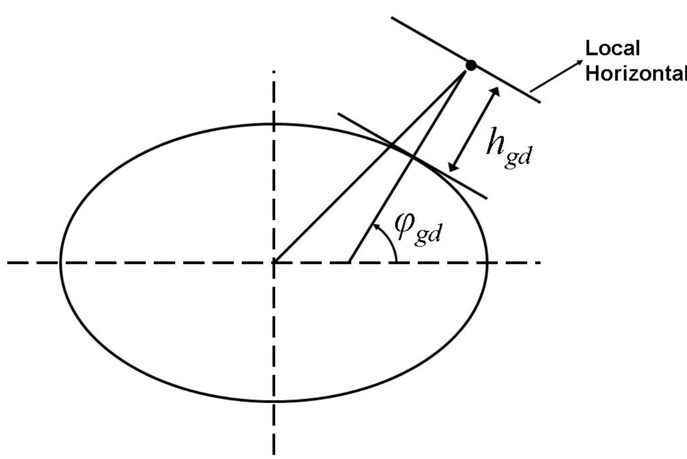


Figure 3.3: Geodetic latitude and altitude definitions.

local horizontal plane, the velocity vector is defined by the velocity magnitude, v, the velocity azimuth, ψ, and the flight path angle, γ. The velocity azimuth is measured positive clockwise from north to the projection of the velocity vector in the local horizontal plane. The flight path angle is the angle between the local horizontal plane and the velocity vector and is measured positive above the local horizontal plane. The local horizontal frame is dependent on the Earth frame, so corresponding subscripts accompany the variables where appropriate. Likewise, the phrasing "inertial" corresponds to values in the ECI local horizontal, and "relative" corresponds to values in the ECR local horizontal.

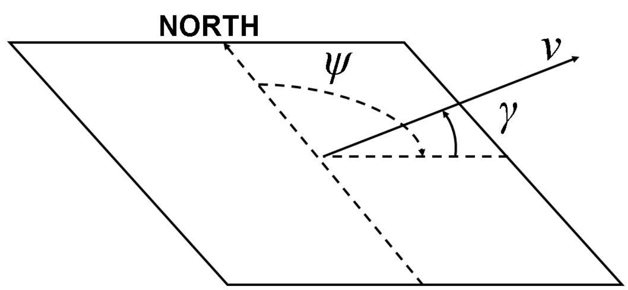


Figure 3.4: Local horizontal plane.

3.1.4 Body/Body Reference

Two sets of axes are associated with the vehicle itself. The body frame and the body refrence frame are placed on a generic model in Figure 3.5. The body reference frame is a Cartesian system where the X-axis points along the longitudinal axis of the vehicle, the Y-axis points out the wing, and the Z-axis completes the right-handed system. The origin is arbitrary, but for simplicity it is placed at the nose. Within the body reference frame, the center of gravity (CG), the aerodynamic reference point, and the engine gimbal points are measured. The body axes are defined with the CG as the origin. This Cartesian system is aligned with the

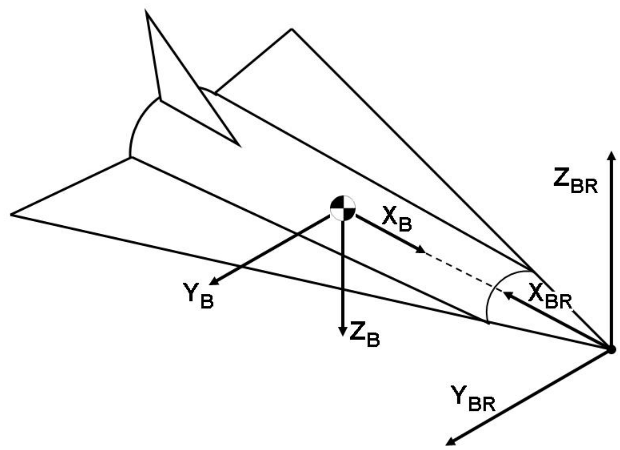


Figure 3.5: Body axes and body reference axes on a generic model.

X-axis forward along the longitudinal axis, the Y-axis pointing out the right wing and the Z-axis completing the right-handed system. The aerodynamic and thrust forces are initially measured in this frame before being rotated to the ECI frame for integration. The angle of attack, α, is the angle between the body X-axis and the atmospheric-relative velocity vector when sideslip and bank angle are both zero. The relative Euler pitch angle, θ_R is the elevation angle above the local horizontal frame.

3.2 Planet Model

The generality of POST2 provides for a broad range of gravitational and atmospheric inputs to define the planet model. This section describes the oblate Earth model and the atmospheres chosen for study.

3.2.1 Oblate Earth Model

This study makes use of an oblate Earth model defined in the POST2 Manuals[2,3]. The model consists of the second through eighth harmonics in the gravitational potential function defined by the WGS 84 Equipotential Ellipsoid model and given in Table 3.1.

Table 3.1: Second through eighth harmonics in the gravitational potential function.

$\boldsymbol{i}$	$\boldsymbol{J_i}$
2	1.0826393×10^{-3}
3	$-2.53215307 \times 10^{-6}$
4	$-1.61098761 \times 10^{-6}$
5	$-2.35785649 \times 10^{-7}$
6	$5.43169846 \times 10^{-7}$
7	$-3.32376398 \times 10^{-7}$
8	$-1.77210399 \times 10^{-7}$

The gravitational constant for the Earth is $\mu_E = 1.4076539 \times 10^{16}$ ft^3/s^2. The equatorial and polar radii for Earth are $R_{Ee} = 2.092564633 \times 10^7$ ft and $R_{Ep} = 2.085554866 \times 10^7$ ft, respectively. The mean perigee altitude, h_{mp}, and mean apogee altitude, h_{ma} are measured relative to the mean earth radius, calculated as

$$R_{Em} = \frac{R_{Ee} + R_{Ep}}{2} = 2.08905975 \times 10^7 ft \tag{3.1}$$

Lastly, the rotation rate for Earth is $\omega_E = 7.292115 \times 10^{-5}$ rad/s.

3.2.2 Atmosphere

For all nominal cases, the atmosphere is a 1976 US Standard Atmosphere[50] with extensions for the upper atmosphere and no winds. This model is a widely accepted industry standard for basic atmosphere calculations. The atmosphere is mathematics-based as a function of altitude. For all Monte Carlo cases, the atmosphere uses tables generated by GRAM99. These tables are monovariate functions of geodetic altitude. The four tables used by POST2 are atmospheric temperature, density, pressure and speed of sound. In addition to the GRAM99 atmosphere, GRAM99 winds are included as windspeed tables in a north-east-down frame. The reference date for GRAM99 was selected as the epoch J2000, or noon January 1, 2000. The reference location is 287° longitude and 28.5° latitude.

3.3 Vehicle Model

The vehicle for this study is the rocket-based upper stage of a TSTO launch vehicle similar to the model in Figure 3.6. This section gives detailed explanations of the vehicle weight, propulsion characteristics, aerodynamics, and static trim in the pitch plane.

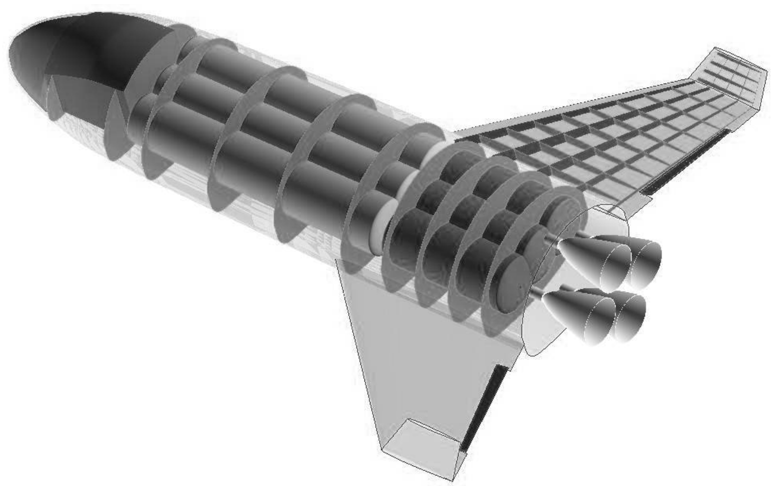


Figure 3.6: Two-stage-to-orbit upper stage model. Courtesy Jeff Robinson, NASA

3.3.1 Vehicle Mass Properties

The vehicle total weight is calculated as the sum of the dry weight, fuel weight, and oxidizer weight. The total vehicle weight for all simulations is 304,863 lb. The initial propellant weights are 67,781.2 lb and 176,231.1 lb. The propellant weights are allowed to go negative during the simulation because the optimization objective is to maximize the injected weight. The CG location was fixed, and fuel consumption and slosh were not considered.

3.3.2 Propulsion

The propulsion system for the vehicle consists of 4 RP-1/LOX rocket engines with typical[51,52,53] performance parameters. Each rocket engine has a specific impulse (I_{sp}) of 346.8 s, a nozzle exit area (A_e) of 6.644 ft^2, and vacuum thrust (T_{vac}) of 101,643 lb. The oxidizer/fuel (O/F) ratio is 2.6:1 and is input to POST2 as a mixture ratio of 0.278. The vehicle is initially set to 90% throttle, though engine throttling is used to limit the acceleration to $3g$ during flight.

3.3.3 Aerodynamics

The aerodynamics for the vehicle were developed at NASA Ames Research Center using the Configuration Based Aerodynamics (CBAERO) software package[54], courtesy of Jeff Bowles. CBAERO is an engineering-level software tool that predicts aero-thermodynamic environments of general vehicle configurations. The data has been converted to multivariate tables of C_D, C_L, and C_M. The coefficient values are determined as a function of dynamic pressure, Mach number and angle of attack. Linear interpolation is used between data points and linear extrapolation is used beyond the minimum and maximum values. The reference area for the vehicle is 1364.4 ft^2.

3.3.4 Static Trim

All of the vehicle properties described above culminate in the requirement of maintaining static trim in the pitch plane. Static trim is maintained in this study through engine gimabaling such that

$$\sum M_\theta = 0 \tag{3.2}$$

where M_θ is the moment in the pitch plane. The pitch plane in the body reference axes is shown in Figure 3.7. The reference length for the static trim calculations is 73.2 ft.

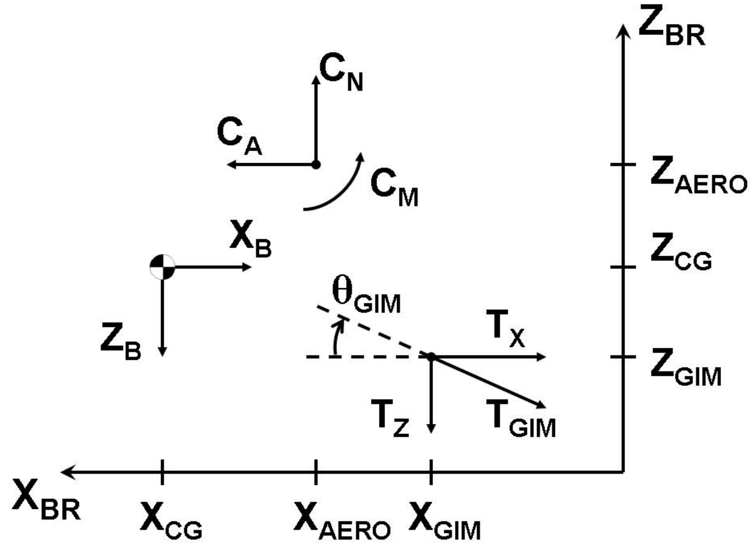


Figure 3.7: Pitch plane in the body reference axes for static trim.

The engine gimbal locations are given in Table 3.2. The engines gimbal about these points creating thrust at the gimbal angle, θ_{GIM}. The fixed CG location is at 60% X and 6% Z,

Table 3.2: Engine gimbal points [ft] for static trim in the pitch plane.

Engine	X_{GIM}	Y_{GIM}	Z_{GIM}
1	70.27	0.0	-3.093
2	69.89	6.312	4.368
3	69.89	0.0	10.73
4	69.89	-6.312	4.368

while the aerodynamic reference point is along the longitudinal axis at 67% X. About the aerodynamic reference point, the C_L and C_D forces are resolved into axial (C_A) and normal (C_N) forces.

3.4 Trajectory States

3.4.1 Initial State

The nominal initial state is shown in Figure 3.8. The initial state is assumed to be the point just after stage separation, without any remaining effects from the first stage. The full vehicle is launched from Kennedy Space Center at (28.5°N, 279.45°E)[51]. At the time of separation, the nominal initial geodetic latitude and relative longitude for the upper stage is (28.5°N, 287°E). The upper stage nominal initial velocity is 6,000 ft/s at an azimuth of 92.7° and a relative flight path angle of 7.5°. The azimuth assumes a launch along a great circle with an inclination of 28.5°; however further investigation of the inclination is beyond the scope of this thesis. The nominal initial geodetic altitude is 135,000 ft.

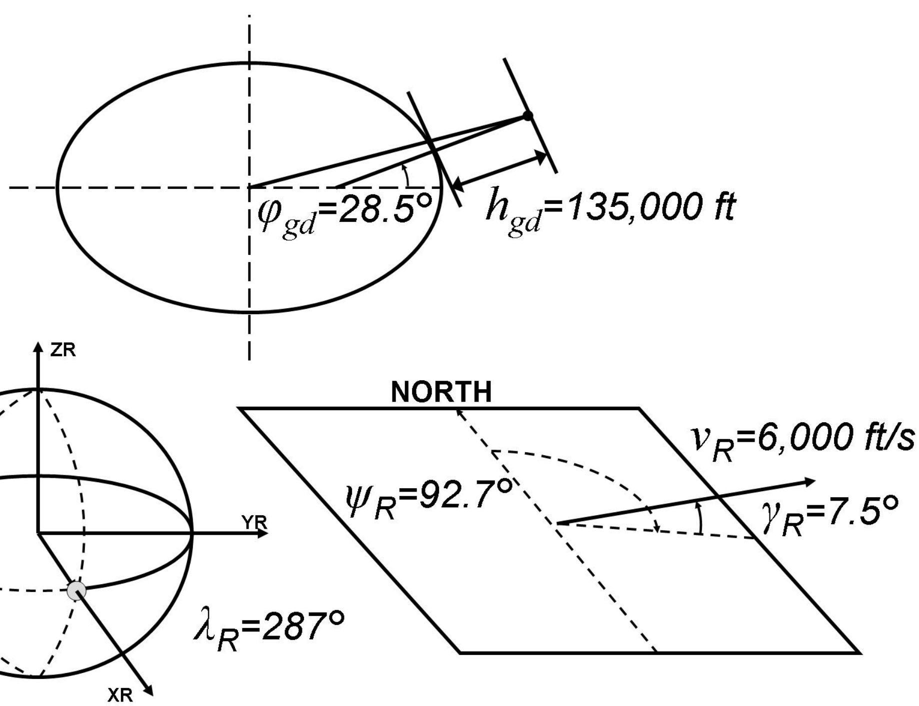


Figure 3.8: Nominal initial state.

3.4.2 Target Orbit

The target orbit is defined by mean perigee and apogee altitudes as 50×100 nmi. The orbit is chosen as it is a good phasing orbit for reaching a 100 nmi circular orbit by performing a burn at apogee. This thesis is only concerned with the trajectory from the upper stage nominal initial conditions described above to MECO. No consideration of the trajectory beyond MECO, such as circularization, is included. In addition to the basic size constraints, it is desired to reach the orbit above 375,000 ft to ensure atmospheric exit with a positive flight path angle for a coast toward apogee.

3.5 Summary

The trajectory overview has been presented including coordinate frames, an oblate Earth model, a detailed vehicle model, and initial and target states. The next chapter presents the methodology for the targeting and optimization problem in POST2 and NPSOL, the predictor-corrector algorithm, and the neural network modification. Each method is tested using a nominal case developed using the trajectory overview from Chapter 3.

Chapter 4

Simulation Methodology

Now that the vehicle, flight environment, initial conditions, and target orbit have been defined, the algorithm methodologies will be explained. This chapter first looks at the formulation of the trajectory optimization problem and a nominal optimal case. Next, a predictor-corrector algorithm is explained and a nominal case is developed. Lastly, an in-depth explanation of a neural network modification is given.

4.1 Optimization with POST2 and NPSOL

The objective of trajectory optimization is to find the path that minimizes or maximizes a cost function, such as minimum fuel, using the controls available to the system. For POST2, the basic translational equations of motion[4] are

$$\begin{aligned} \dot{\underline{r}}_I &= \underline{v}_I \\ \underline{v}_I &= \underline{A}_{TI} + \underline{A}_{AI} + \underline{A}_{GI} \end{aligned} \tag{4.1}$$

where $\underline{A}_{TI}$ is the thrust acceleration, $\underline{A}_{AI}$ is the aerodynamic acceleration, and $\underline{A}_{GI}$ is the gravitational acceleration. The I subscript denotes that the equations are solved in the ECI frame. The equations of motion fit the general form[55,56] of

$$\begin{aligned} &\dot{x}(t) = f\left(t,\, x(t),\, u(t),\, \epsilon\right) \\ &x(t) \in \mathbf{R}^m, \qquad u(t) \in \mathbf{R}^n, \qquad \epsilon \in \mathbf{R}^p \end{aligned} \tag{4.2}$$

where x denotes the state variables, u denotes the controls, and ϵ denotes the parameter vector. The equations of motion are integrated forward in time using the 4th Order (4-Stage) Runge-Kutta[57,58] method with the update formula

$$x_{i+1} = x_i + \frac{1}{6}[K_1 + 2K_2 + 2K_3 + K_4] \tag{4.3}$$

where the direction field samples K_j are given by

$$\begin{aligned} K_1 &= \Delta t \cdot f\left(t_i,\, x(t_i)\right) \\ K_2 &= \Delta t \cdot f\left(t_i + \tfrac{1}{2}\Delta t,\, x(t_i) + \tfrac{1}{2}K_1\right) \\ K_3 &= \Delta t \cdot f\left(t_i + \tfrac{1}{2}\Delta t,\, x(t_i) + \tfrac{1}{2}K_2\right) \\ K_4 &= \Delta t \cdot f\left(t_i + \Delta t,\, x(t_i) + K_3\right) \end{aligned} \tag{4.4}$$

The targeting problem consists of modification of the controls until the constraints are met. Next, the optimization objective is approached by changing the controls to improve the optimization variable without violating any constraints. The search method, NPSOL, progresses through a series of major iterations. Between blocks of major iterations, the problem is rescaled to set the maximum magnitude of the elements of the Jacobian to 1. The series of major iteration blocks used for this thesis was 25, 25, 100, 100, 250. For the T/O problem, the controls have simple lower and upper bounds

$$u_{i_L} \leq u_i(t) \leq u_{i_U}, \qquad i = 1, 2, ..., n \tag{4.5}$$

In addition to bounds, the control variables also have associated weights, which were initially set equal to the initial control guesses. The T/O problem is also defined by a combination of equality and inequality end conditions

$$\begin{aligned} \phi_{eq}\left(x(t_f),\ \epsilon\right) &= 0 \in \mathbf{R}^{k_{eq}} \\ \phi_{in}\left(x(t_f),\ \epsilon\right) &\geq 0 \in \mathbf{R}^{k_{in}} \end{aligned} \tag{4.6}$$

where k is the total number of inequality and equality constraints. The optimization variable is minimized (or maximized) by use of a cost functional

$$F\left(x(t^*),\ \epsilon\right) \tag{4.7}$$

where t^* denotes the event at which minimization should occur. For NPSOL to solve the sequential quadratic programming problem, the problem is formulated as

$$\begin{aligned} &minimize \qquad F(\eta), \qquad \eta \in \mathbf{R}^l \\ &for \qquad lb \leq \beta(\eta) \leq ub, \qquad \beta(\eta) \equiv \begin{pmatrix} \eta \\ A\eta \\ c(\eta) \end{pmatrix} \end{aligned} \tag{4.8}$$

where η is a variable set, $F(\eta)$ is a nonlinear function, A is a $k \times l$ matrix, and $c(\eta)$ is a k-vector of nonlinear functions. In addition to the bounds, the constraints also have tolerances τ. Figure 4.1 illustrates the regions of activity for the bounds on β. In region 1 the lower bound is violated and in region 5 the upper bound is violated. Within regions 2 and 4, the corresponding bounds are considered active. Region 3 is the ideal region where the bounds are satisfied and inactive. The T/O routine is exited when a solution is found that optimizes the optimization variable at the appropriate event while satisfying the problem constraints. However, it is important to note that the POST2 implementation of NPSOL does not distinguish between local and global minima. The responsibility of determining whether the converged solution is the optimal solution rests solely with the user.

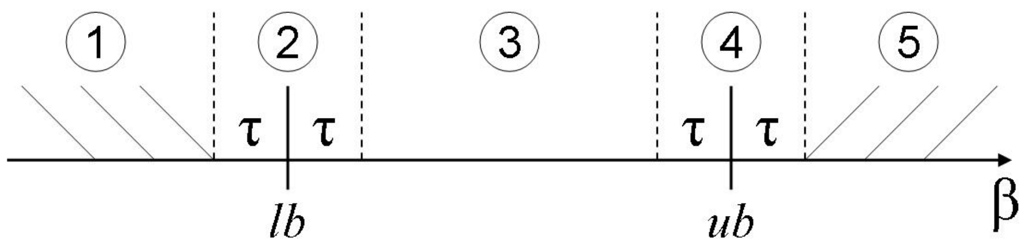


Figure 4.1: NPSOL bounds and tolerances.

4.1.1 Nominal Optimization Case

A nominal case was developed to maximize the injected weight. The target variables chosen for this trajectory were the final mean perigee and apogee altitudes, the final geodetic altitude, and the final inertial flight path angle. The ranges and tolerances for the constraints of T/O problem were

$$\begin{gathered} 50nmi \leq h_{mp} \leq 50.01nmi(\pm 0.01) \\ 100nmi \leq h_{ma} \leq 100.01nmi(\pm 0.01) \\ 375,000ft \leq h_{gd} \leq 380,000ft(\pm 1000) \\ 0.05^{\circ} \leq \gamma_I \leq 60^{\circ}(\pm 0.01) \end{gathered} \tag{4.9}$$

Steering was commanded by an angle of attack table as a function of the relative velocity

Table 4.1: Final angle of attack table for nominal optimization case.

	1	2	3	4	5	6	7	8	9	10
v_r [kft/s]	6.0	8.5	9.0	9.5	10.0	12.0	16.0	20.0	23.0	26.0
$\alpha(v_r)$ [deg]	30.24	26.93	23.37	22.13	20.84	14.81	4.70	-3.84	-9.83	-15.12

of the vehicle. The independent variables for NPSOL were 10 angle of attack table values with bounds $-25^\circ \leq \alpha(v_r) \leq 50^\circ$ at relative velocities of 6.0, 8.5, 9.0, 9.5, 10.0, 12.0, 16.0, 20.0, 23.0, and 26.0 kft/s, and the inertial MECO velocity with bounds 23.0 kft/s $\leq v_{I,meco} \leq$ 29.0 kft/s. The angle of attack table used linear interpolation between values and linear extrapolation beyond. The integration time step for the optimization was set to 10 seconds for the sake of computation time. While this reduction in time came at the cost of decreased accuracy, the accuracy for the chosen integration step was adequate for the requirements of this study. The nominal case started from the nominal initial conditions, and used a 1976 US Standard Atmosphere, 100% thrust, and 100% aerodynamics. The final angle of attack table is shown in Table 4.1 and the final MECO velocity was 25,752.26 ft/s. The nominal pitch profile, shown in Figure 4.2, shows a smooth curve with the only hitch ($\approx$70 sec) due to the higher concentration of control variables between 8.5 and 10.0 kft/s. Figure 4.3 shows that the geodetic altitude profile follows a smooth path toward the altitude constraint. The angle of attack table, pitch profile, and geodetic altitude show characteristic properties of optimal solutions in that there are few major fluctuations that would cause steering losses. The end conditions were successfully targeted within the tolerances with a 49.996 $\times$ 99.993 nmi orbit, geodetic altitude of 374,994 ft, and inertial flight path angle of 0.351°. The injected weight was 50,435 lb, corresponding to a propellant fraction of 0.835.

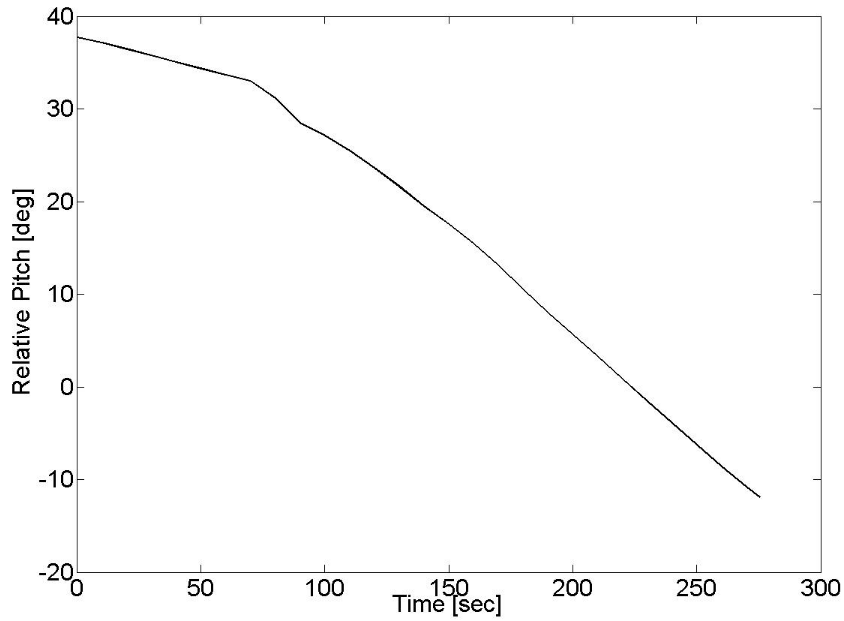


Figure 4.2: Relative pitch profile for nominal optimization case.

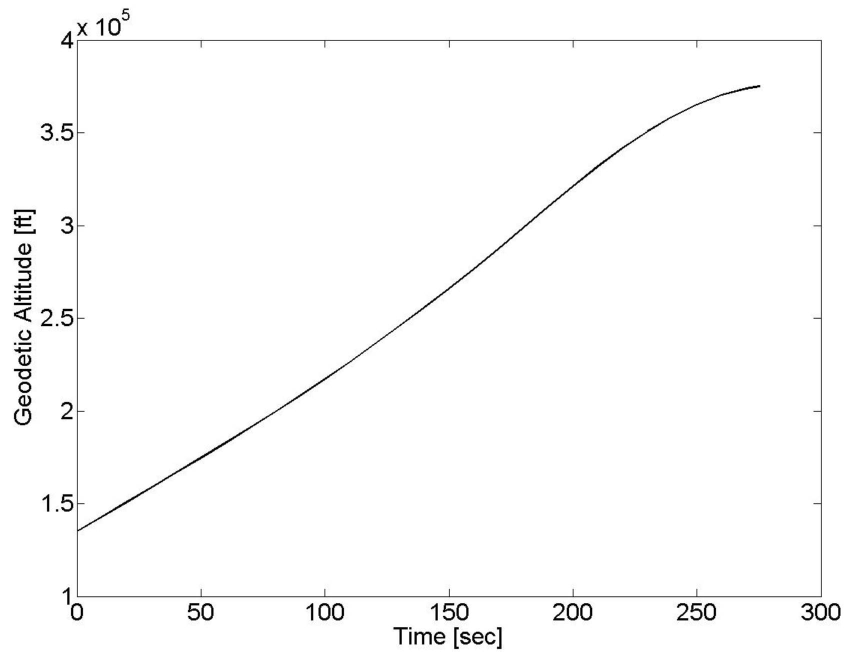


Figure 4.3: Geodetic altitude profile for nominal optimization case.

4.2 Predictor-Corrector

A predictor-corrector (P/C) is a guidance algorithm that uses knowledge of the vehicle state to project a path toward the exit conditions to determine the next pitch command. The algorithm "predicts" the path that the vehicle will continue on at the current state and "corrects" the control command for the vehicle to reach the desired end conditions. The benefit to a P/C algorithm is its use of computational power to determine the current path and find the appropriate control to guide the vehicle toward the appropriate end state. The concept for the P/C algorithm developed here comes largely from the POST2 framework created by Jits[27].

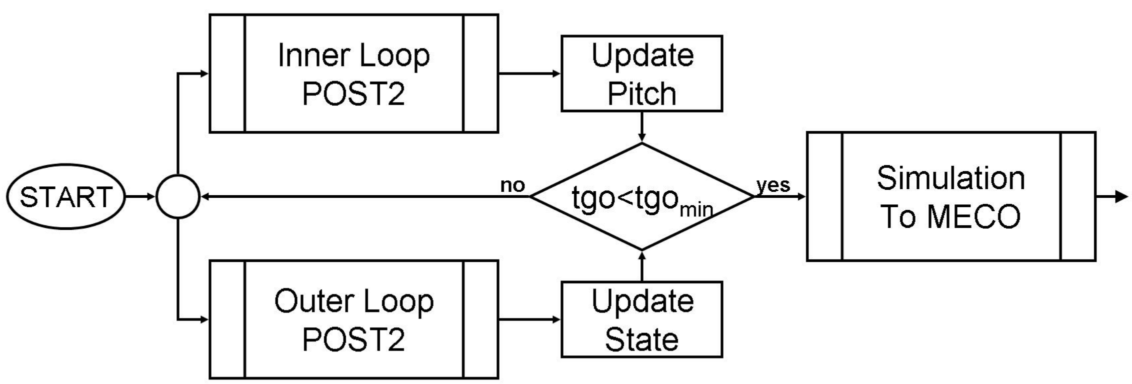


Figure 4.4: Predictor-corrector guidance architecture.

The basic flow of the algorithm can be seen in Figure 4.4. The outer loop of the P/C algorithm is the "vehicle" simulation, which propagates the trajectory at a given pitch command for a time equal to the guidance interval. The guidance interval, or dtg, is the time between the execution of consecutive pitch commands. The inner loop is the "guidance computer" simulation that propagates the trajectory toward the end conditions from the current state. NPSOL is used to target the final orbit conditions using a pitch control sequence. When NPSOL converges on the targeted orbit, the next pitch command for the outer loop is de-

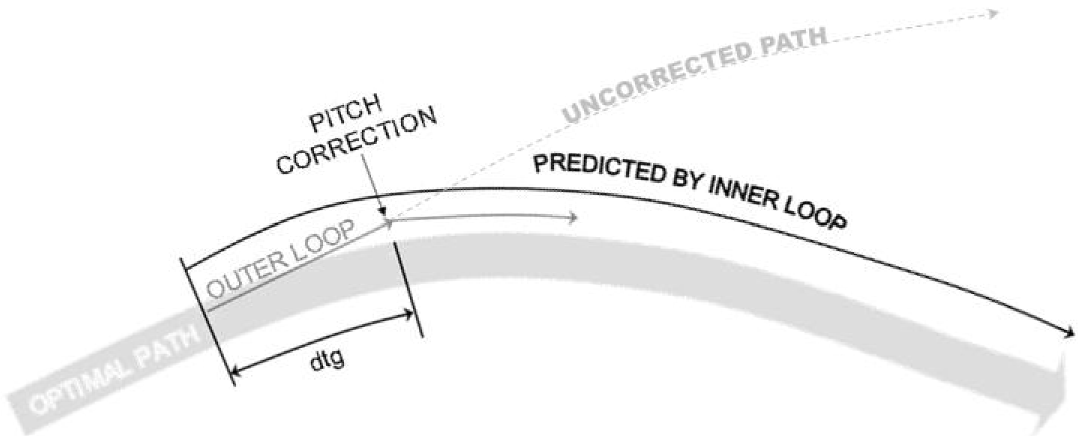


Figure 4.5: Predictor-corrector visual representation.

termined. At the end of each guidance interval, the pitch command is updated by the inner loop, and the vehicle state is updated by the outer loop. A visual representation of this process is given in Figure 4.5. The vehicle deviates from the desired path due to perturbations, and if left uncorrected, would fly toward an undesirable end state. Using pitch corrections from the inner loop, the vehicle maintains a path toward the correct end conditions. The algorithm continues to process until a user-defined time-to-go minimum is reached. The time-to-go, or tgo, is a guidance commanded variable that represents the remaining trajectory time predicted by the inner loop. After the minimum time-to-go is reached, the vehicle trajectory is simulated to MECO.

4.2.1 Predictor-Corrector Trajectory Simulation

The inner loop trajectory is a simulation from the current vehicle state toward the end conditions with NPSOL targeting. The current vehicle state is obtained from the outer loop in the previous guidance interval. Targeting without optimization is chosen to reduce the inner loop computation time. The assumption is made that the guidance system has no knowledge of the "actual" flight atmosphere, thrust performance, or aerodynamic forces. Therefore, the inner loop uses nominal information in the simulation. The atmosphere used is a 1976 US Standard Atmosphere, the thrust is assumed to be 100%, and the aerodynamic coefficients from the CBAERO tables are kept at 100%. These conditions are the "approximate" flight properties. In order to keep computation time down, the inner loop required a simple event sequence with few controls. The event sequence chosen provides NPSOL with four controls to use in targeting the four constraints of the target orbit. The controls are two pitch commands, the time between the two commands, and the MECO velocity. The event sequence can be described as

1. Fly at current pitch for *dtg* seconds.
2. Fly at **pitch = control 1** for a duration of **time = control 2**.
3. Fly at **pitch = control 3** until **velocity = control 4**.

The first event exists as an approximation to the outer loop trajectory during the current guidance interval. Unlike the optimization that steers with angle of attack, the inner loop uses the first term (constant) of a pitch polynomial as the control. Occasionally, some situations arose in which POST2 could not correctly target the orbit. A series of checks was developed to provide the predictor-corrector algorithm with automated solutions to any errors it may encounter. The flow of the inner loop through these checks is shown in Figure 4.6. When NPSOL is used for targeting, it receives several blocks of major iterations, between which a rescale of the problem is performed. The rescale is done to set the largest term in the Jacobian of the optimization problem equal to one. When the problem targets the correct orbit just before a rescale is set to occur, POST2 will sometimes rescale the problem before exiting NPSOL, resulting in an error. If the first check finds this error, it removes the previous

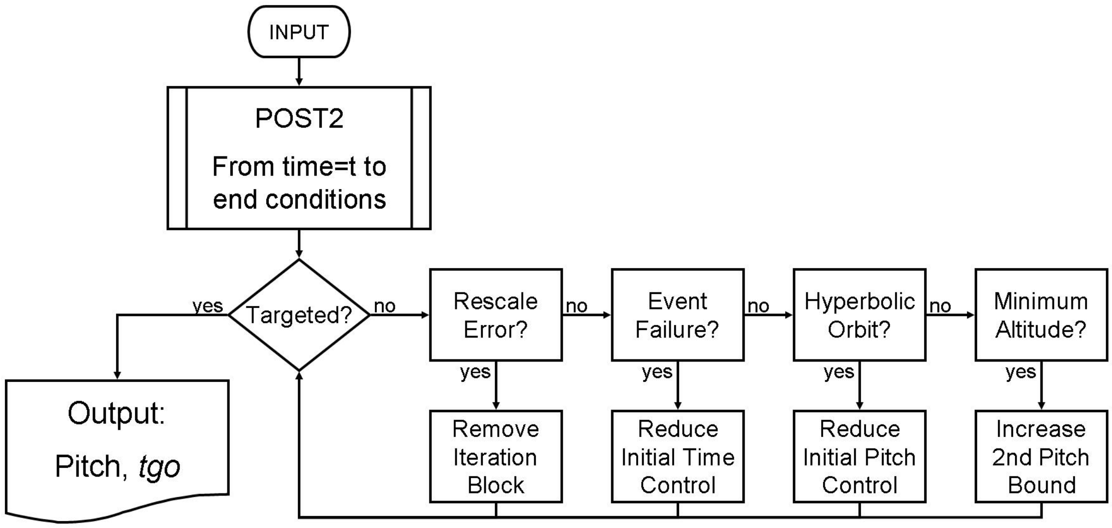


Figure 4.6: Predictor-corrector inner loop flow.

major iteration block to prevent the unwanted rescale. The second check determines if the trajectory failed to reach the third event in the sequence. This error occurs when the time control between the pitch controls is larger than the remaining trajectory time, causing the vehicle to miss the target. The solution for this error is to reduce the initial guess for the time control. In some cases, the initial guess for the first pitch control is too high, causing the vehicle to enter a hyperbolic orbit and burn all of its mass. The correction for this error is to reduce the initial guess. Lastly, the vehicle may reach a minimum altitude when the second pitch command cannot go high enough to reach the target. When a minimum altitude error is encountered, the upper bound on the control is increased. When the orbit is targeted, the inner loop outputs the time-to-go and the first pitch control as the pitch command. The initial guesses for the control variables are an important part of the targeting problem. The first set of initial values were predetermined by an optimal trajectory developed using the inner loop event sequence starting from the nominal initial conditions. Each guidance interval, the best initial values are extracted from the previous inner loop output and fed back as initial guesses for the next iteration. From one iteration to the next, the bounds on the pitch controls are updated to approximately $\pm 15°$ of the previous initial guesses for pitch, and the upper bound for the time control is set to the time-to-go less one guidance interval.

The outer loop trajectory propagates the vehicle from the current state to the end of the guidance interval at the given pitch command. The outer loop has unique atmospheric properties, thrust, and aerodynamics, which are the "actual" flight properties.

4.2.2 Nominal Predictor-Corrector Case

A nominal case was developed using the P/C algorithm, starting from the nominal initial conditions. A 1976 US Standard Atmosphere with no wind was used for the outer loop, as well as nominal thrust and aerodynamics. A guidance interval schedule of five seconds for the first 200 trajectory seconds and 3 seconds for the remaining trajectory was chosen based on analysis of inner loop computation times. As the remaining trajectory became smaller (i.e. as trajectory time passed) the computation time reduced as well. The nominal trajectory showed near-real-time guidance, meaning the inner loop computation time was approximately equal to the guidance interval. The nominal case proved to be sufficient in targeting the desired orbit, though the pitch profile is considerably different from the optimal path as shown in Figure 4.7. The initial pitch is considerably higher for the P/C algorithm due to the formation of the inner loop control design. With only two pitch controls, NPSOL tends to converge to a solution with one high pitch and one low pitch. The building of the nominal pitch profile is shown in Figure 4.8. Each iteration corresponds to one step of the pitch profile. Figures 4.8(a)-4.8(e) show the pitch profile for individual inner loop iterations while Figure 4.8(f) shows how they construct the entire profile. While this formation deviates

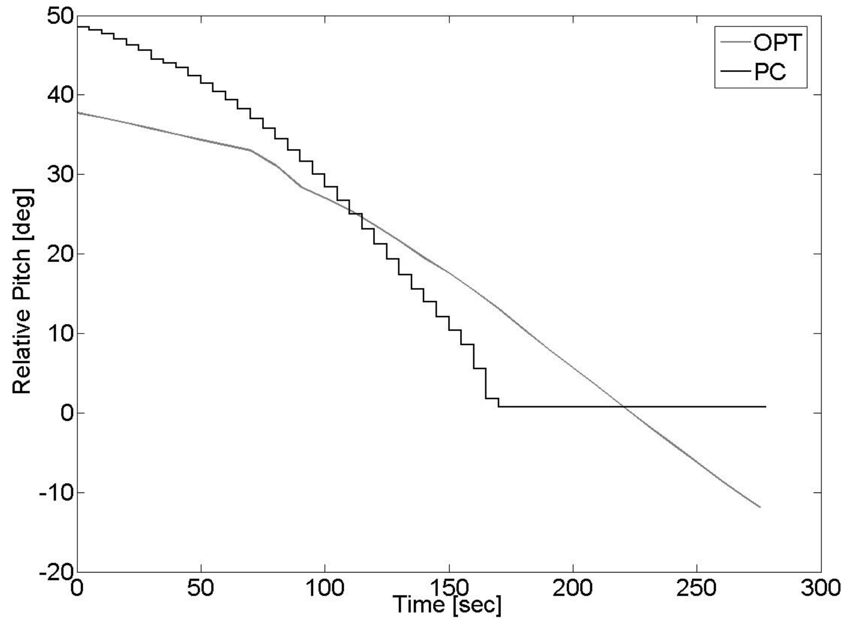


Figure 4.7: Relative pitch profile for nominal predictor-corrector case.

far from the optimal pitch profile, it was the only method that produced an inner loop that ran fast enough for real-time consideration. The step pattern of the pitch profile is a result of using individual pitch commands at each interval. An attempt was made to use the second (linear) coefficient of the pitch polynomial to simulate pitch rates. The inner loop computation time for this attempt was, on average, two to three times longer than the pitch command case. In addition, the pitch rate case did not target any orbit. The aerodynamic force changes resulting from the pitch rates caused the vehicle to veer too far off-target to recover.

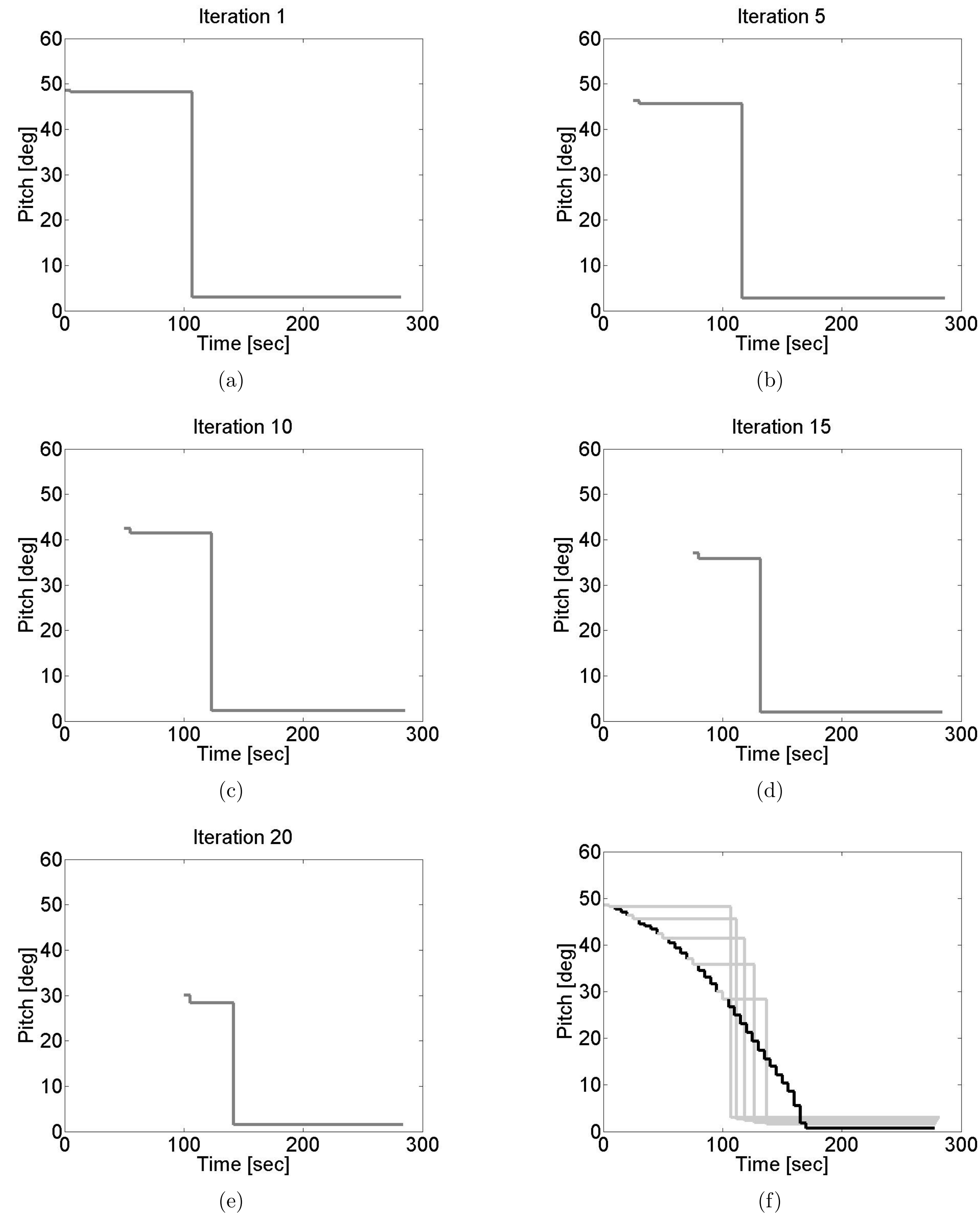


Figure 4.8: Individual inner loop iterations (a-e) make up the entire pitch profile (f) for the nominal predictor-corrector case.

The P/C algorithm showed high accuracy in targeting the appropriate orbit, largely due to the minimal perturbations by aerodynamics and atmosphere at the high altitudes near the end of the trajectory. The geodetic altitude profile in Figure 4.9 shows that while the trajectory deviates from the optimal path, it benefits from exiting the atmosphere earlier. The orbit was targeted at 50.010×100.000 nmi with a geodetic altitude of 379,999 ft and an inertial flight path angle of 0.357°. While the P/C algorithm targeted with high accuracy, the higher initial pitch and subsequent deviations from the optimal path resulted in a significant loss of injected weight. The final injected weight for the P/C was 49,463 lb, 972 lb less than the optimized case.

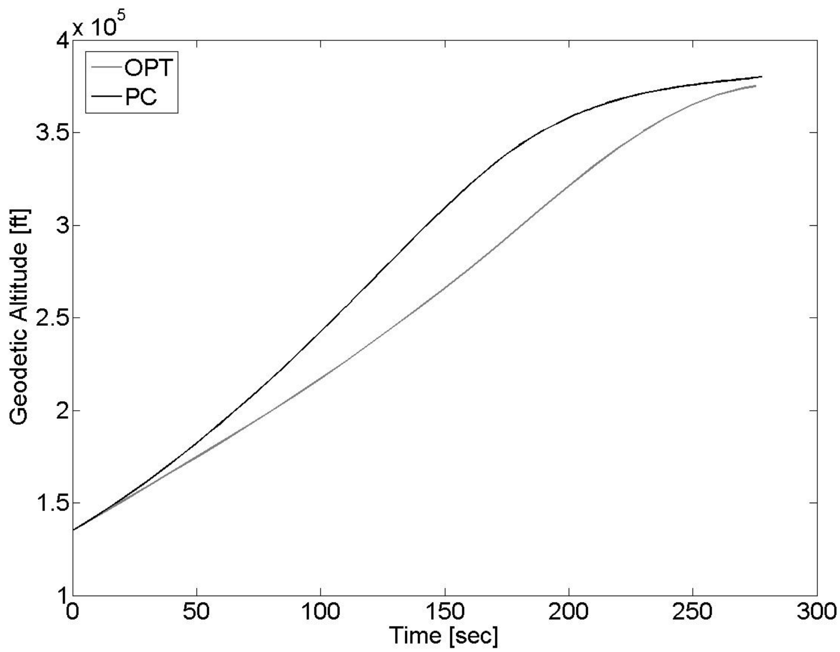


Figure 4.9: Geodetic altitude profile for nominal predictor-corrector case.

4.3 Neural Network Modification

A neural network (NN) is a set of interconnected artificial neurons that can mimic the response of a system without the need to design a model. Each neuron is an individual processing unit containing weights to modify the input. The most typical network architecture is the feedforward backpropagation network, which has an input layer, one or more hidden layers, and an output layer. Each layer receives input from the previous layer, and processes the data through a series of weights and a transfer function. The concept behind a neural network is that it may be trained using known data to approximate the output of a system. Schaechter[41] aptly describes a NN "as a special purpose computer which is configured to represent a nonlinear mapping implied by empirical data." According to White[28], "multilayer feedforward networks are universal approximators," making them a suitable fast alternative to an on-the-fly targeting method.

4.3.1 Neural Network Architecture Selection

A neural network modification for the previous guidance architecture has been developed. As shown in Figure 4.10, a NN module has replaced the inner loop POST2 run. The NN module is trained to compute the pitch command and time-to-go given some representation of the current vehicle state. The greatest benefit of having a NN module as the inner loop is the computation time of one pitch command from a given state is near-instantaneous, as it is merely a few steps of matrix multiplication. With a lower inner loop computation time, the guidance interval can be set smaller than for the predictor-corrector algorithm. Within the module, a set of separate networks are used to process the vehicle state. Each network is activated for a different range of time-to-go, as shown in Figure 4.11.

The first NN in the module is the main workhorse for the trajectory steering. This network

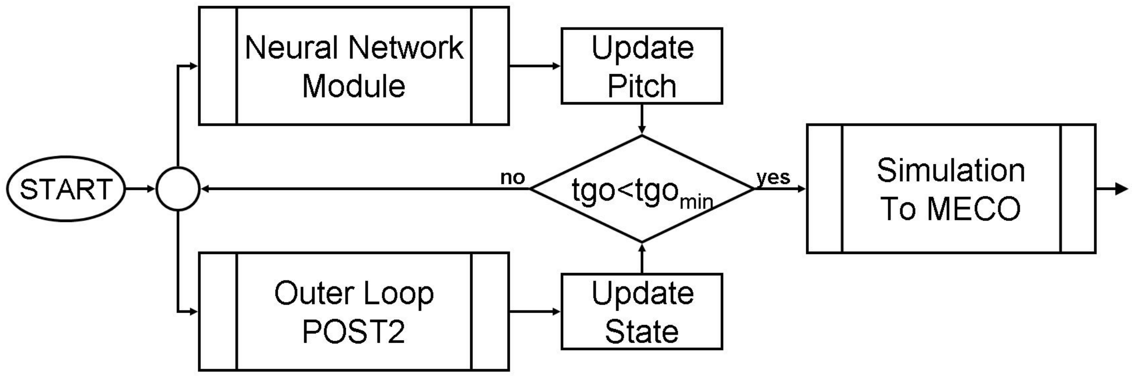


Figure 4.10: Guidance architecture with neural network module.

receives the vehicle's current relative velocity, geodetic altitude, relative flight path angle and total weight, and outputs the next pitch command. Selection of the four input variables came from extensive experimentation into which combination of inputs produced the least error. This network determines all pitch commands until the predicted time-to-go reaches ten seconds. While the first network is feasible for use up to the minimum time-to-go, attempts doing so resulted in final orbits with undesirable accuracy. When time-to-go reaches ten seconds, this network is only used for approximating the time-to-go, and the second network is engaged for pitch commands. This network converts the vehicle's current mean apsides altitudes into a pitch command. The apsides altitudes were chosen as inputs because they are the biggest constraints on the final orbit. The network switch time-to-go of ten seconds was chosen for the behavior of the mean perigee altitude. The vehicle approaches apogee during flight, so the computed perigee altitude rapidly changes from a few thousand nmi below zero to the target value of 50 nmi. In the last ten seconds, however, the perigee altitude values are significantly close to the target value, allowing smoother processing of the NN. The final network in the module serves to finalize the pitch control and set a MECO velocity for the remaining trajectory. Like the first network, the closing NN receives the current relative velocity, geodetic altitude, relative flight path angle and total weight for the vehicle.

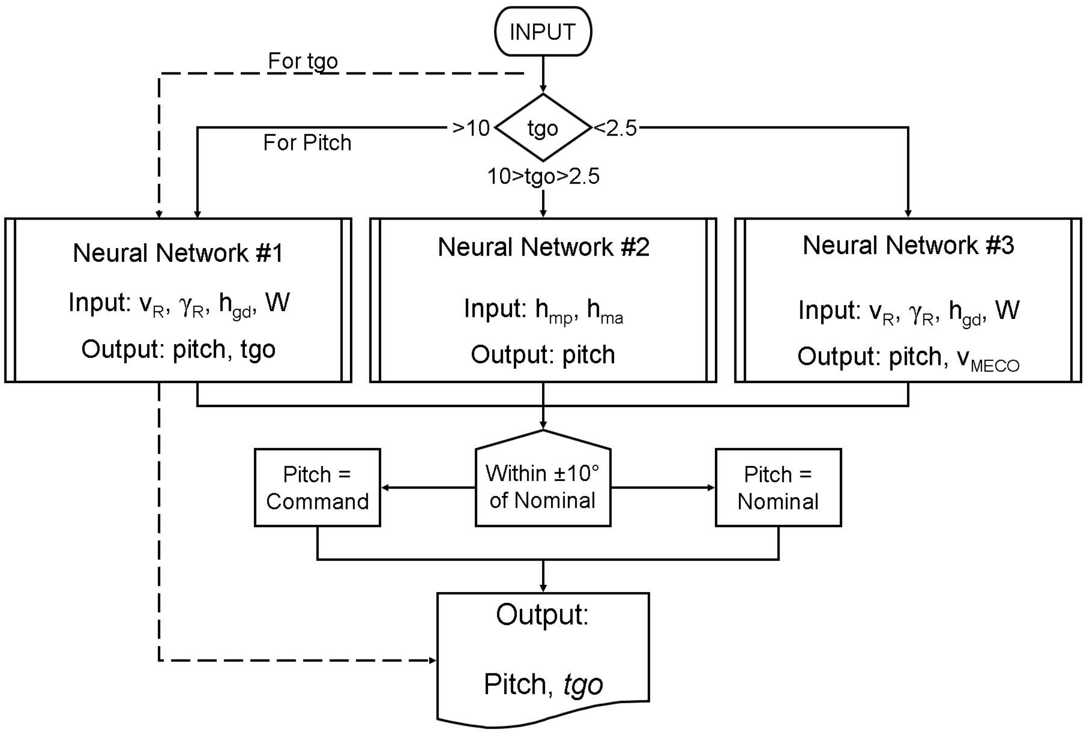


Figure 4.11: Neural network module flow detail.

After each pitch command is output, a feasibility check is made based on a nominal optimization of the trajectory, as shown at the bottom of Figure 4.11. If the command violates feasibility bounds, the network outputs an approximate nominal pitch value at the current trajectory time as the pitch command. The feasibility bounds were set to $\pm 10\%$ of a cubic approximation to the nominal pitch time history. These bounds are implemented to account for combinations of NN inputs that may be beyond the training domain of the network. Each network was trained to a specific set of inputs, P_t, and target outputs, T_t, that were developed from optimization cases. The training set consisted of 100 optimization cases that had nominal aerodynamics and thrust, a 1976 Standard Atmosphere with no winds, and a random initial state. The set came from selection of all succesfully optimized cases in a set

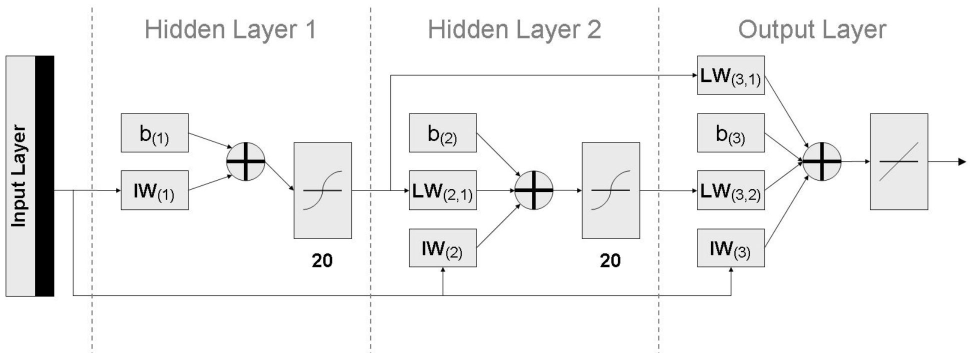
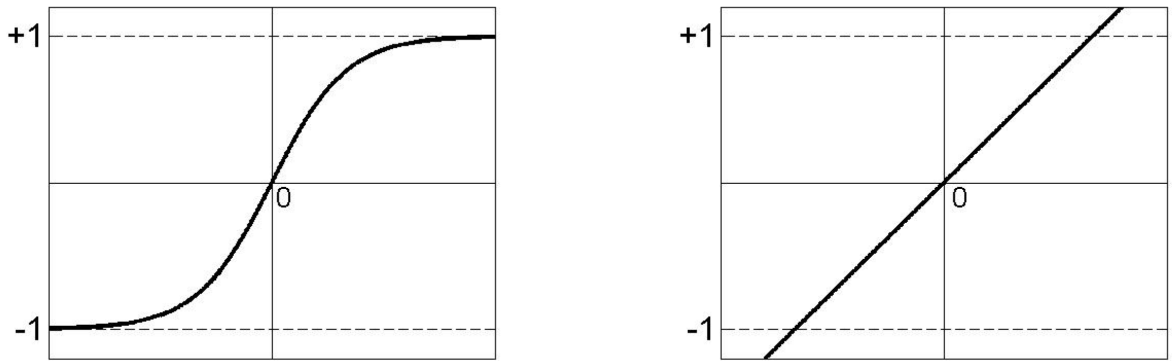


Figure 4.12: Neural network architecture.

of 125 attempts, based on feasibility of the control sequence chosen by the optimizer. For the first network, the training set consisted of all data points along the trajectories. This set consisted of approximately 3,000 input-output pairs. At each data point, the state is used as a training input and the pitch and time-to-go are used as the training targets. The second network was trained using only data points in the last 10 seconds of the trajectory. The linear nature of the pitch time histories in the last ten seconds allowed linear interpolation to boost the training set size. The training set for the second network consisted of approximately 1,000 pairs of apsides altitudes and pitch commands. The closing network was trained on the last 10 seconds, though the target outputs for this network were the final pitch and velocity at the end of each trajectory case. This training set also consisted of approximately 1,000 pairs.

While the inputs and outputs varied by network, the architecture of all three networks was kept identical, and can be seen in Figure 4.12, with notation adapted from the MATLAB Neural Network Toolbox User's Guide[10]. The basic template for the networks was a cascade-forward flow, a modification to typical feedforward logic. The cascading effect comes from the inclusion of terms in each layer's calculation from all previous layers, while a typical feedfor-



(a)

(b)

Figure 4.13: Tangent-sigmoid (a) and pure linear (b) transfer functions.

ward network has weights only from directly adjacent previous layers. The cascade-forward architecture provided higher accuracy over basic feedforward implementations. Selection of the number of hidden layers was critical in network design. The two-hidden-layer design showed accuracy improvements over a single-hidden-layer design without a major increase in training time. However, attempts at any more than two hidden layers proved to increase the training time without a significant improvement in accuracy. The number of neurons per hidden layer was chosen to be 20 for the same reasons. Below 20, the NN showed larger errors in calculation, whereas above 20, the network had a higher training cost. The transfer functions were chosen as tangent-sigmoid for the hidden layers and pure linear for the output layer. The transfer functions, shown in Figure 4.13, are typical choices for backpropagation trained networks because they are continuous and differentiable. The functions and their derivatives are

$$f_1(x) = f_2(x) = \frac{2}{1+e^{-2x}} - 1 \tag{4.10}$$

$$f_1'(x) = f_2'(x) = \frac{4e^{-2x}}{(1+e^{-2x})^2} \tag{4.11}$$

$$f_3(x) = x \tag{4.12}$$

$$f_3'(x) = 1 \tag{4.13}$$

Lastly, the Levenberg-Marquardt Algorithm was selected as the training method. This method uses an augmented Newton iteration and a modified backpropagation technique to significantly reduce training time without much loss of accuracy. All network development, simulation and training was performed using the MATLAB Neural Network Toolbox, and an explanation of the simulation and training algorithms is presented below.

4.3.2 Neural Network Simulation and Training

Network simulation is the process of passing q number of inputs P through the hidden layers and output layer to obtain q outputs T. The first layer processes the inputs, producing the intermediate output

$$z_1 = IW_1 \cdot P + b_1 \tag{4.14}$$

where IW_1 contains the input weights for the first hidden layer, and b_1 is the layer bias. This output is passed through the tangent-sigmoid transfer function to produce the layer output

$$a_1 = f_1(z1) \tag{4.15}$$

The output from Equation (4.15) becomes the input for the next hidden layer. This layer processes similarly to the first hidden layer, but a term is added in from the input layer for a cascading network. The intermediate output is given by

$$z_2 = IW_2 \cdot P + LW_{2,1} \cdot a_1 + b_2 \tag{4.16}$$

where $LW_{2,1}$ is the layer weight for the second hidden layer corresponding to the output of the first hidden layer. The output is again passed through the layer transfer function to yield

$$a_2 = f_2(z2) \tag{4.17}$$

The processes is repeated with the addition of another cascading term to the intermediate output for the output layer to produce

$$z_3 = IW_3 \cdot P + LW_{3,1} \cdot a_1 + LW_{3,2} \cdot a_2 + b_2 \tag{4.18}$$

which is passed through the output layer transfer function to give the final neural network output

$$a_3 = f_3(z3) \tag{4.19}$$

The training objective is to minimize the error between the training set target output and the neural network output given by

$$e = T_t - T \tag{4.20}$$

The basic training concept is backpropagation, which comes in a variety of algorithms. The backpropagation technique used here is based on the Levenberg-Marquardt algorithm[40]. The algorithm is based on a series of epochs, where each epoch consists of processing the entire training set and updating the weights. Let χ be a vector of length n consisting of all weights and biases for a given layer in the format $\chi = [w_1, w_2, ..., w_n, b]$, concatenating all input weights and layer weights in the given layer. The critical step in the Levenberg-Marquardt algorithm is the weight update method

$$\Delta\chi = -[H + \mu I]^{-1} g \tag{4.21}$$

where H is the Hessian matrix, μ is a performance parameter, I is the identity matrix, and g is the gradient. When μ is zero, Equation (4.21) becomes Newton's method. The reasoning behind μ is to approach Newton's method as fast as possible by incrementing at each epoch to improve performance. The Levenberg-Marquardt training algorithm is designed to offer improved training time without the need to calculate the exact Hessian matrix. The Hessian matrix is approximated by

$$H = J^T J \tag{4.22}$$

and the gradient is calculated as

$$g = J^T e \tag{4.23}$$

where J is the Jacobian matrix given by

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial \chi_1} & \frac{\partial e_1}{\partial \chi_2} & \cdots & \frac{\partial e_1}{\partial \chi_n} \\ \frac{\partial e_2}{\partial \chi_1} & \frac{\partial e_2}{\partial \chi_2} & \cdots & \frac{\partial e_2}{\partial \chi_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_q}{\partial \chi_1} & \frac{\partial e_q}{\partial \chi_2} & \cdots & \frac{\partial e_q}{\partial \chi_n} \end{bmatrix} \tag{4.24}$$

The Jacobian matrix is computed through an adaptation of standard steepest descent back-propagation[31,59]. The first step in the output layer is to determine the sensitivity given by

$$\delta_3 = f_3'(z_3) \tag{4.25}$$

Using this sensitivity, the Jacobian can be populated according to

$$J = \frac{\partial e_q}{\partial \chi_n} \equiv \begin{cases} \frac{\partial e}{\partial w} = [\delta_3 a_2; \delta_3 a_1; \delta_3 P] \\ \frac{\partial e}{\partial b} = \delta_3 \end{cases} \tag{4.26}$$

Now, the weights for the output layer can be updated using Equation (4.21). For the next layer in backwards order, the sensitivy is processed by the weights, yielding

$$\delta_2^* = IW_3 \cdot \delta_3 + LW_{3,1} \cdot \delta_3 + LW_{3,2} \cdot \delta_3 P \tag{4.27}$$

which is used to find the sensitivity for the layer, given by

$$\delta_2 = \delta_2^* f_2'([z_2]) \tag{4.28}$$

As before, the Jacobian can be populated as

$$J = \frac{\partial e_q}{\partial \chi_n} \equiv \begin{cases} \frac{\partial e}{\partial w} = [\delta_2 a_1; \delta_2 P] \\ \frac{\partial e}{\partial b} = \delta_2 \end{cases} \tag{4.29}$$

The weights for the second hidden layer can be updated using Equation (4.21), and the backpropagation continues to the first hidden layer with the sensitivity processed by the weights,

$$\delta_1^* = IW_2 \cdot \delta_2 + LW_{2,1} \cdot \delta_2 \tag{4.30}$$

The layer sensitivity is then calculated as

$$\delta_1 = \delta_1^* f_1'([z_1]) \tag{4.31}$$

Finally, the first hidden layer Jacobian is constructed to be

$$J = \frac{\partial e_q}{\partial \chi_n} \equiv \begin{cases} \frac{\partial e}{\partial w} = \delta_1 P \\ \frac{\partial e}{\partial b} = \delta_1 \end{cases} \tag{4.32}$$

and the weights can be updated with Equation (4.21). The algorithm is exited when a desired output accuracy is met, the maximum number of epochs is reached, or the algorithm determines there is no more appreciable improvement. For the three networks in the NN module, the number of epochs (500) and the desired accuracy (10^{-10}) were set at values to force the algorithm to run until it could not improve.

4.3.3 Nominal Neural Network Case

A nominal case was developed using the NN algorithm, starting from the nominal initial conditions. A 1976 US Standard Atmosphere with no wind was used for the outer loop, as well as nominal thrust and aerodynamics. A guidance interval of one second was chosen to reduce the path deviation between pitch commands. The NN method can calculate much faster than the allotted interval, but this interval allows the end-to-end trajectory to run faster by having fewer exchanges between MATLAB and the POST2 outer loop. The algorithm produced a nominal case that follows closely along the optimized nominal case, as evidenced by the pitch profile in Figure 4.14. The three separate networks in the inner loop module are clearly defined in the pitch profile. The last ten seconds show the second network to be lowering the pitch based on current orbit altitudes and the final pitch from the closing network places the vehicle on the track toward MECO. The geodetic altitude profile in Figure 4.15 confirms the close correllation between the NN and optimal nominal cases. This method followed the nominal optimal case tightly because the NN is an approximation of optimization. The NN nominal case targeted the orbit constraints at 50.012×99.992 nmi with a geodetic altitude of 375,850 ft and an inertial flight path angle of 0.352°. The final injected weight for the NN algorithm was 50,416 lb, only 19 lb less than the optimized case.

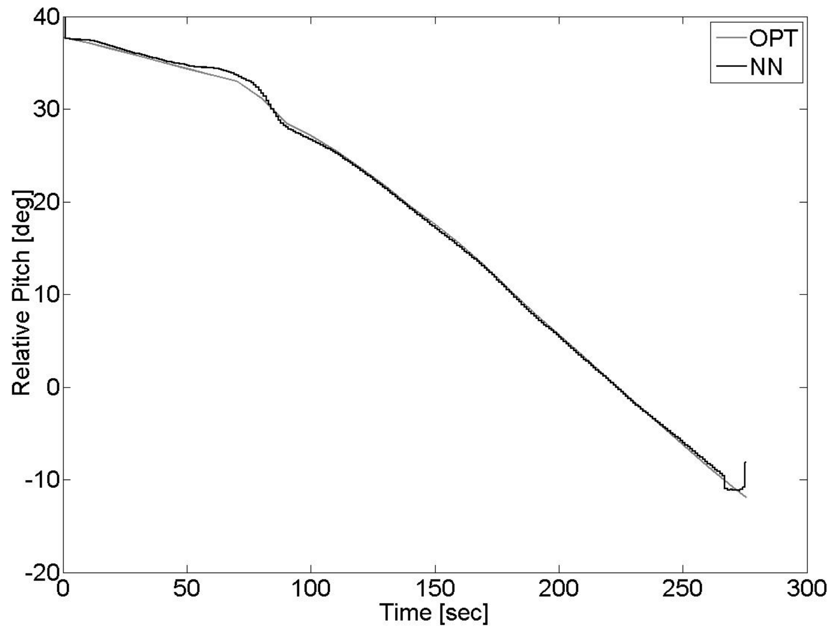


Figure 4.14: Relative pitch profile for nominal neural network case.

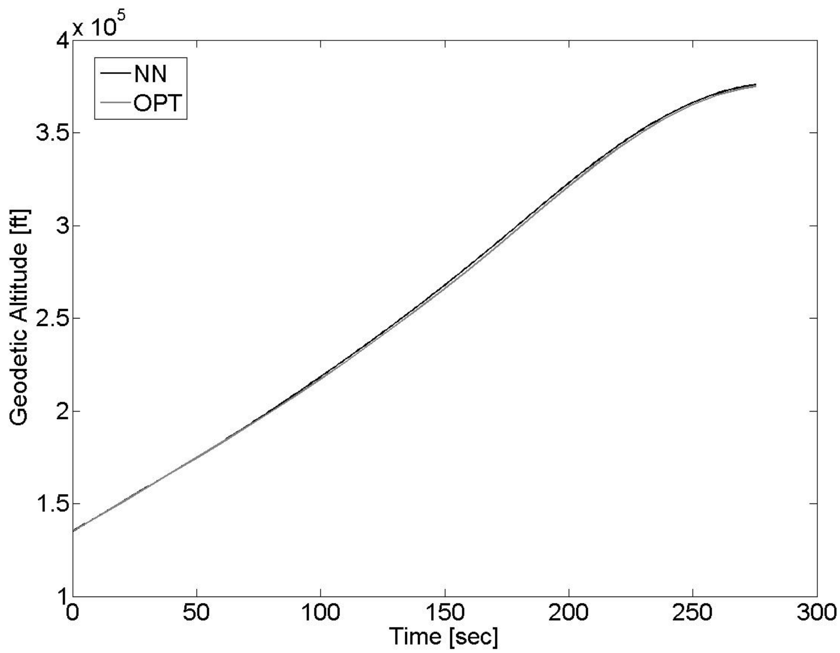


Figure 4.15: Geodetic altitude profile for nominal neural network case.

Chapter 5

Monte Carlo

In this chapter, an overview of the Monte Carlo simulation is presented with details on variable variations and random atmospheres. Monte Carlo results are presented for the POST2/NPSOL optimization, the predictor-corrector (P/C) algorithm and the neural network (NN) modification. Lastly, a brief discussion on the computation time for each method is presented.

5.1 Monte Carlo Variables

Monte Carlo simulation typically refers to any computational analysis using random sampling. The method used here is the variation of several trajectory simulation inputs to propagate the uncertainty through the model. The chosen Monte Carlo variables for this study are multipliers for C_D, C_L, C_M, and vacuum thrust, and variations in initial relative velocity, initial geodetic altitude, and initial relative flight path angle. The mean values and standard deviations are shown in Table 5.1 and are used to develop normal distributions for each variable. Monte Carlo analysis is important in guidance development because

Table 5.1: Monte Carlo variables with nominal values and standard deviations.

Variable	$\bar{q}$	σ
C_D	100%	10%
C_L	100%	10%
C_M	100%	10%
h_{gd}	135,000 ft	10,000 ft
v_r	6000 ft/s	125 ft/s
γ_r	7.5°	1.25°
T_{vac}	100%	1.5%

the guidance system must take into account variations in the initial state, thrust properties and aerodynamics. As a simple survey of the necessity of a guidance method, a set of cases has been developed for comparison with the nominal case. The initial relative flight path angle, relative velocity and geodetic altitude were allowed to be one of three values: $\bar{q} + \sigma$, $\bar{q}$, or $\bar{q} - \sigma$. All aerodynamic and thrust multipliers were 100%, and a 1976 US Standard Atmosphere was used. All 27 combinations of the three initial state variables were run with steering commanded by the optimized nominal angle of attack values given in Table 4.1. The pitch profiles in Figure 5.1 and the geodetic altitude profiles in Figure 5.2 show the 27 cases with the black line representing the nominal optimization case. While the pitch profile shows little variation, the geodetic altitude paths are vastly distributed. It is clear that when the variables are allowed to vary up to 3- and 4-σ variations and aerodynamic, thrust, and atmospheric variations are also included, the final dispersions may be much larger. Therefore, it is important to consider such dispersions when designing a guidance algorithm. The Monte Carlo set for this thesis consisted of 500 cases. The MATLAB "randn" function was used to develop the normally distributed sets for the variables in Table 5.1. The Monte Carlo atmospheres and winds were generated by GRAM99 with the initial date as J2000 and the initial location as the nominal initial longitude and latitude. The actual dispersions used are displayed in Appendix A.

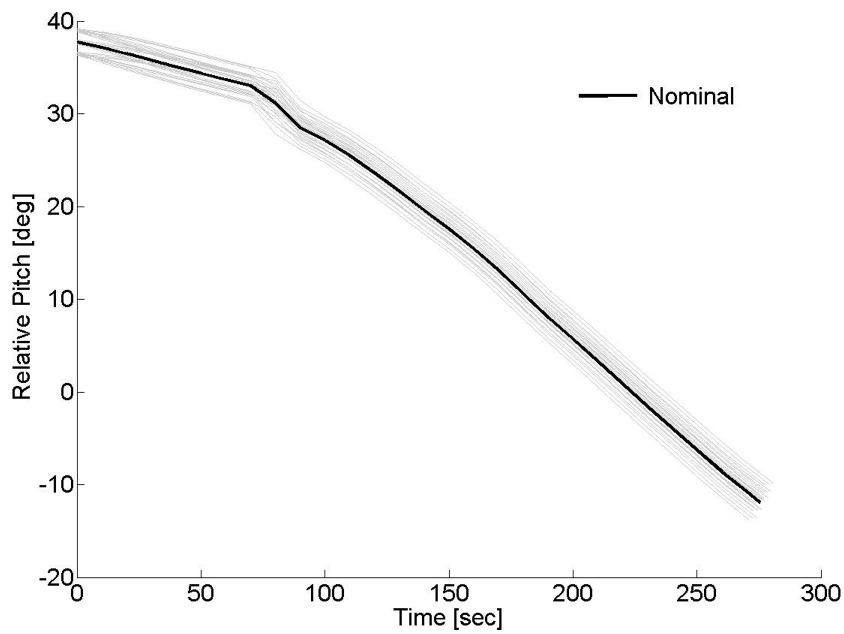


Figure 5.1: Pitch profile for trajectory simulations with off-nominal conditions using nominal angle of attack table.

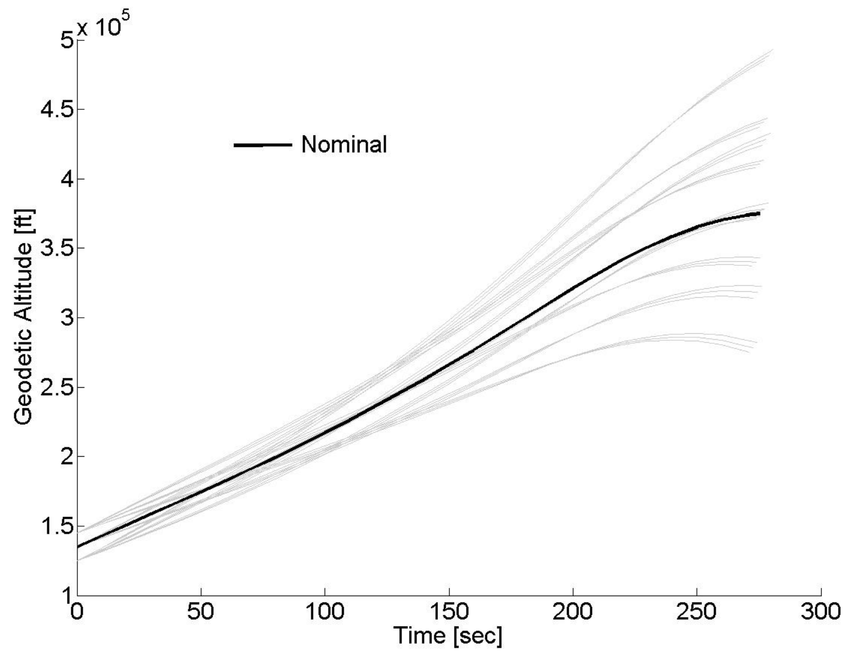
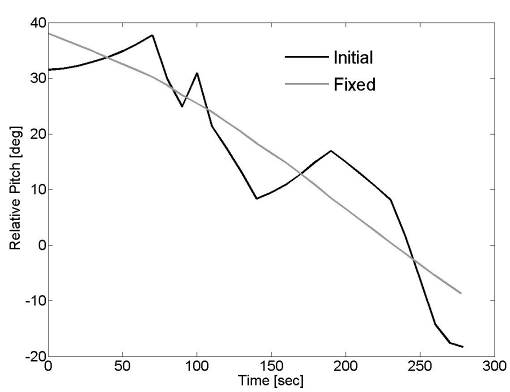


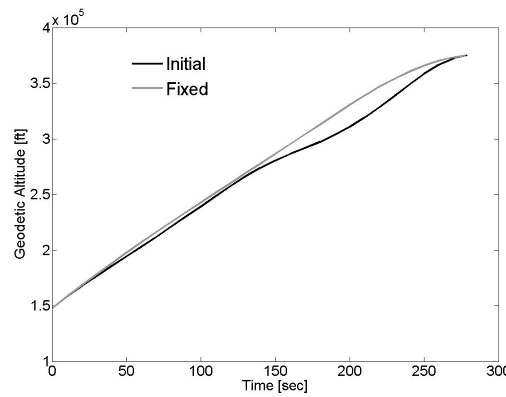
Figure 5.2: Geodetic altitude profile for trajectory simulations with off-nominal conditions using nominal angle of attack table.

5.2 Optimization Results

The 500 Monte Carlo cases were run using POST2/NPSOL optimization. Of the 500 cases, 458 reached acceptable solutions based on visual inspection of the pitch and altitude profiles. The remaining cases showed non-smooth curves for pitch, corresponding to uncharacteristic concavities in the altitude profiles. These cases required some adjustment to avoid non-optimal targeting solutions. Examples of non-optimal pitch and altitude profiles are compared with corrected optimal curves in Figures 5.3(a) and 5.3(b), respectively. While the two runs both targeted successfully, it is clear that the pitch profile is initially representative of major steering losses. The injected weight for the non-optimal solution was roughly 600 lb less than the optimal solution.



(a)



(b)

Figure 5.3: Relative pitch (a) and geodetic altitude (b) profiles for non-optimal and optimal targeted solutions.

The process of correcting non-optimal solutions began with adjustment of the angle of attack table bounds. The bounds were set to approximately $\pm 10^\circ$ of the nominal case. This bound adjustment helped NPSOL avoid non-optimal solutions and corrected about half of the cases. Another approach consisted of targeting without optimization first to reach a solution. Then, the initial control guesses were replaced with the final values from the targeting. Lastly, optimization was turned back on to reach an optimal solution. This method worked on the remaining cases producing the set of 500 optimized cases.

The final collection of pitch and altitude profiles can be seen in Figures 5.4(a) and 5.4(b). The altitude curves are smooth with very little concavity until the vehicle has exited the significant atmosphere. The pitch profile shows that most cases tended to decrease in pitch from start to MECO. The region between 50 and 100 seconds appears to be less smooth than

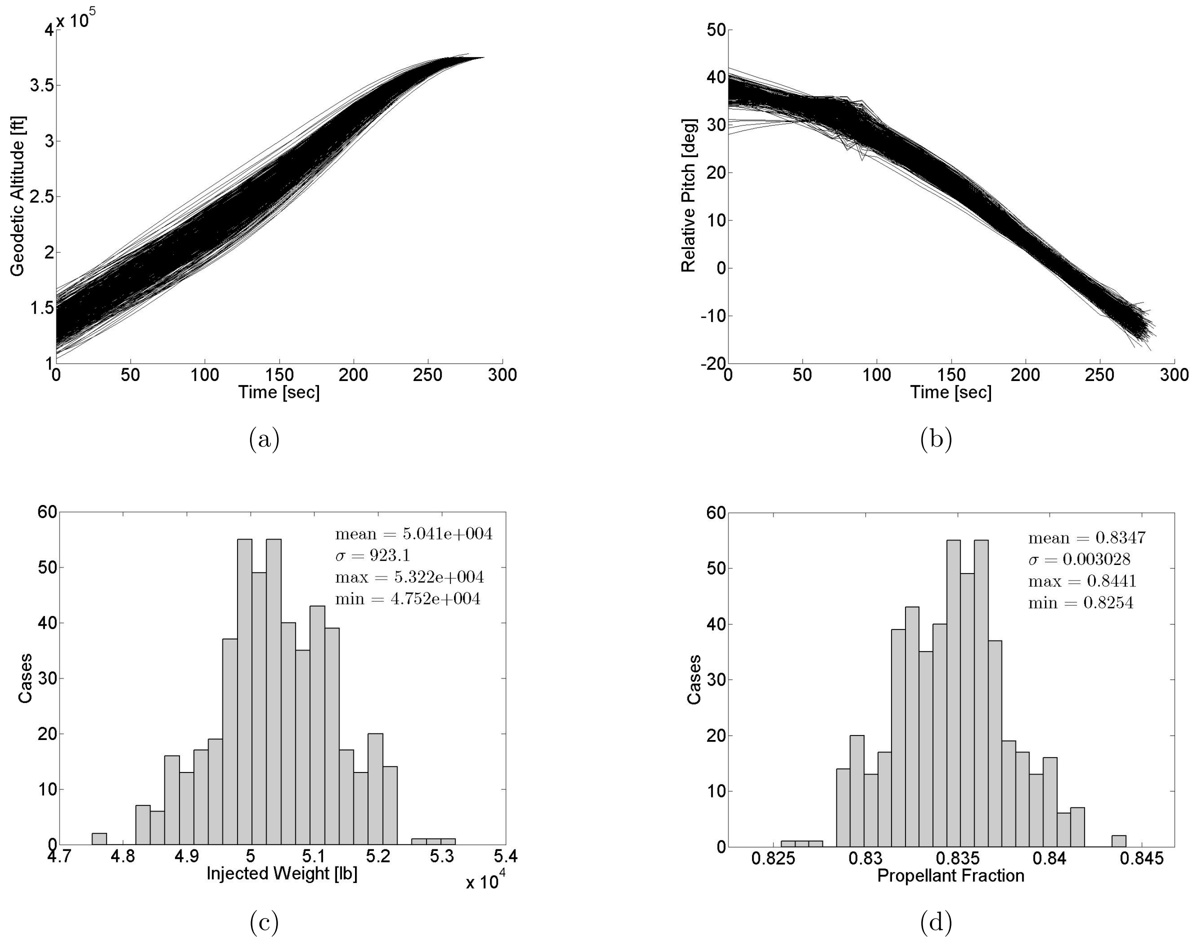


Figure 5.4: Geodetic altitude (a), relative pitch (b), injected weight (c), and propellant fraction (d) Monte Carlo results.

the remainder of the trajectory. This region is typically where the vehicle reaches the four closely packed angle of attack table values at velocities of 8.5, 9.0, 9.5, and 10.0 kft/s. The injected weight in Figure 5.4(c) shows a mean value within 25 lb of the nominal case with a range of approximately ±300 lb. Likewise, the propellant fraction distribution is shown in Figure 5.4(d). The final mean apsides altitudes for the Monte Carlo cases are shown in Figure 5.5 with a target area reflecting the ranges and tolerances for targeting. All 500 cases successfully targeted to within the desired target values given in Equation 4.9.

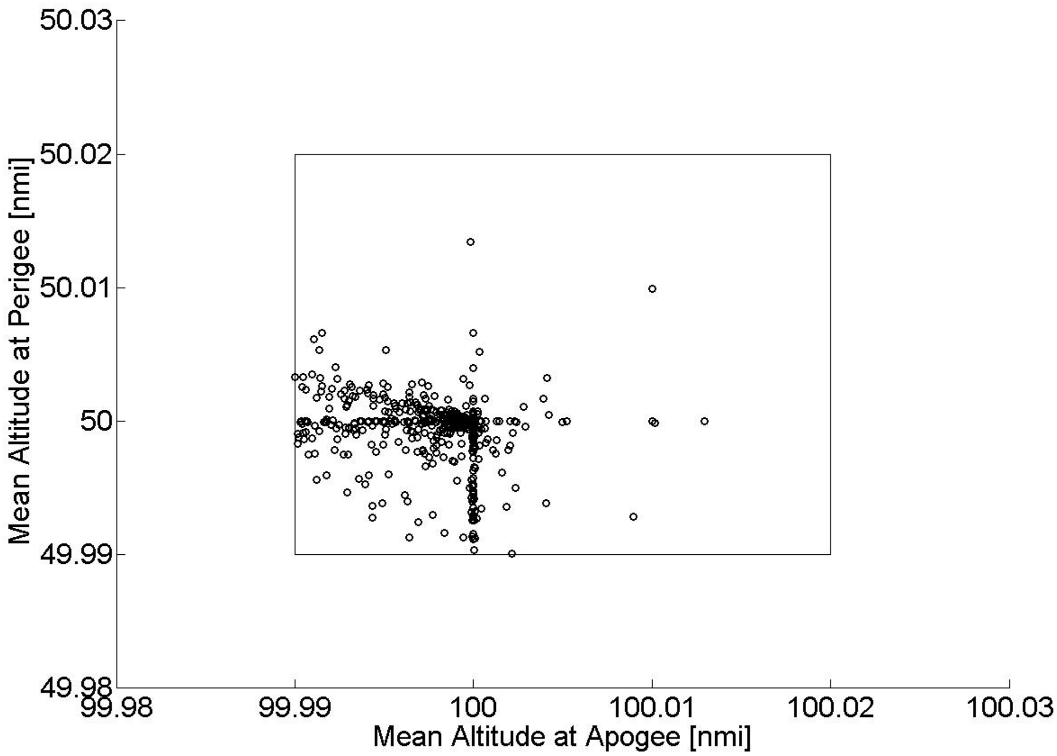


Figure 5.5: Final mean altitude of apsides for optimization Monte Carlo analysis.

5.3 Predictor-Corrector Results

The same atmosphere profiles and Monte Carlo variable values for state, thrust and aerodynamics were used for the corresponding P/C cases (i.e. Optimization case #22 has the same initial state, multipliers and atmosphere as P/C case #22). All 500 cases reached stable orbits at sufficient altitude and positive inertial flight path angle. The altitude profiles

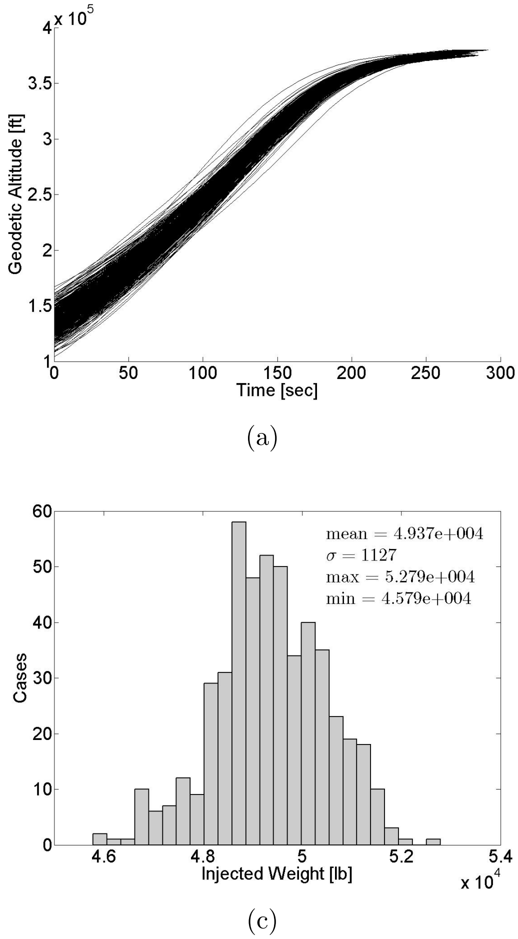


Figure 5.6: Geodetic altitude (a), relative pitch (b), injected weight (c), and propellant fraction (d) Predictor-Corrector Monte Carlo results.

in Figure 5.6(a) for the P/C cases show more concavity than the optimized cases because the initial pitch is higher. The curves also flatten sooner than the optimal cases because the pitch typically remains constant for a longer period leading up to MECO. The pitch profiles in Figure 5.6(b) show that characteristic constant pitch region. Just before MECO, several cases showed a single pitch-up command coming from the inclusion of the second pitch control in the final POST2 run. This final pitch command allows the algorithm to achieve a high accuracy in targeting. The injected weight in Figure 5.6(c) shows a mean value within 100 lb of the nominal case with a range of approximately ±350 lb. Likewise, the propellant fraction distribution is shown in Figure 5.6(d), with an average value of 0.838 and a standard deviation of 0.00369. The average weight difference between the P/C and

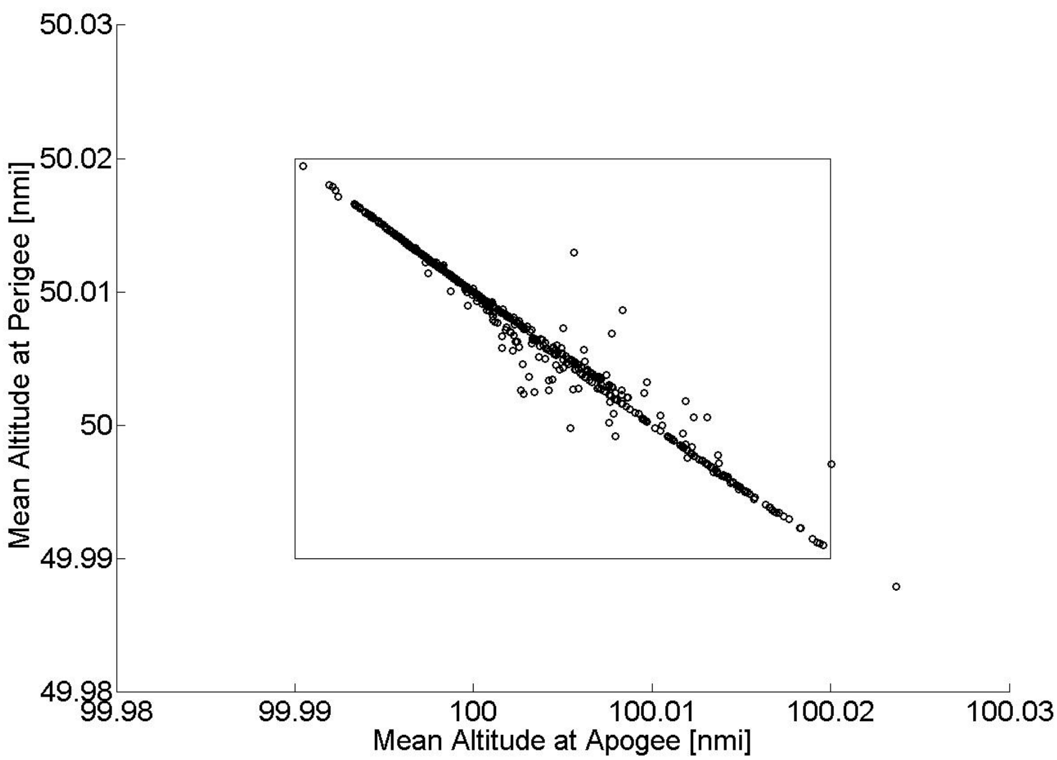


Figure 5.7: Final mean altitude of apsides for P/C Monte Carlo analysis.

optimized cases is approximately 1,040 lb, which corresponds to a ΔV loss of 232.8 ft/s. The final mean apsides altitudes for the Monte Carlo cases are shown in Figure 5.7 with a target area identical to the region for the optimization cases. The target area contains 99.6% of the P/C cases, with the only outliers within a double tolerance. While the P/C algorithm shows desirable accuracy, the performance comes at a cost of decreased injected weight. All 500 cases ran successfully without the need for assistance on any individual case. The P/C algorithm was robust in handling the given Monte Carlo variables, but no investigation has been performed to determine the capability to handle conditions far beyond 3-σ deviations.

5.4 Neural Network Results

For the NN modification Monte Carlo simulation, the same cases were used as in the optimization and P/C. Of the 500 cases, 98.2% reached reasonable orbits, while the remaining cases diverged until reaching the ±10° bounds and following the nominal profile. The altitude profiles in Figure 5.8(a) for the 98.2% NN cases are similar the optimized cases. The pitch profiles in Figure 5.8(b) are also similar to the optimized cases with the exception of the last pitch command for each trajectory. The last pitch command is necessary for accuracy in the targeted orbit.

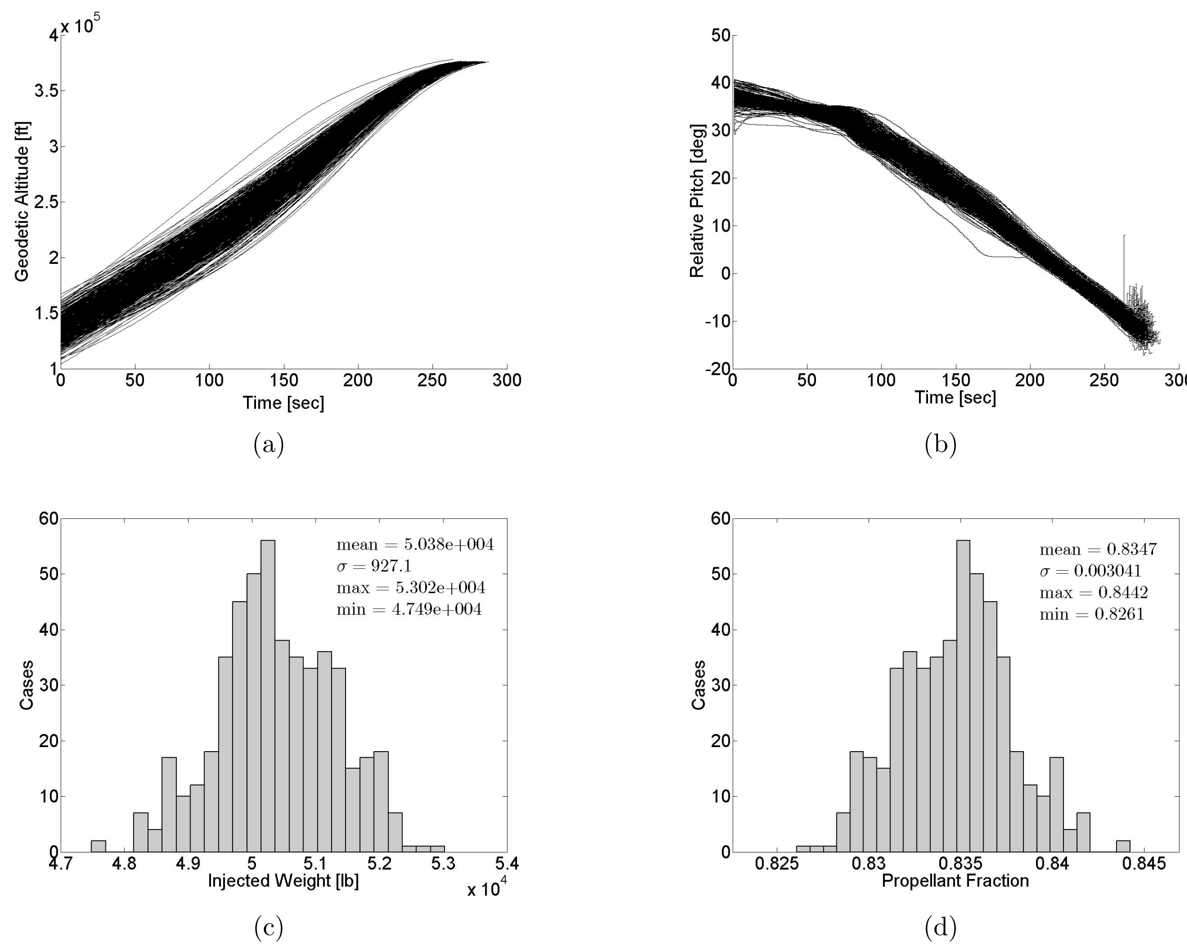


Figure 5.8: Geodetic altitude (a), relative pitch (b), injected weight (c), and propellant fraction (d) neural network Monte Carlo results for 98.2% of the cases.

The injected weight in Figure 5.8(c) shows a mean value of 50,380 lb with a range of approximately ±300 lb. Likewise, the propellant fraction distribution is shown in Figure 5.8(d), with an average value of 0.835 and a standard deviation of 0.00311. The average weight difference between the NN and optimized cases is approximately 30 lb, which corresponds to a ΔV loss of 6.6 ft/s. The NN algorithm showed a 1,010 lb (226 ft/s) advantage over the P/C algorithm. The final mean apsides altitudes for the Monte Carlo cases are shown in Figure 5.9. The successful cases targeted to within ±0.25 nmi for apogee and ±0.40 nmi for perigee. The NN algorithm does not provide the accuracy of the P/C method using an inner targeting loop. However, the NN produced injected weights very close to the optimized cases, using far less computation time.

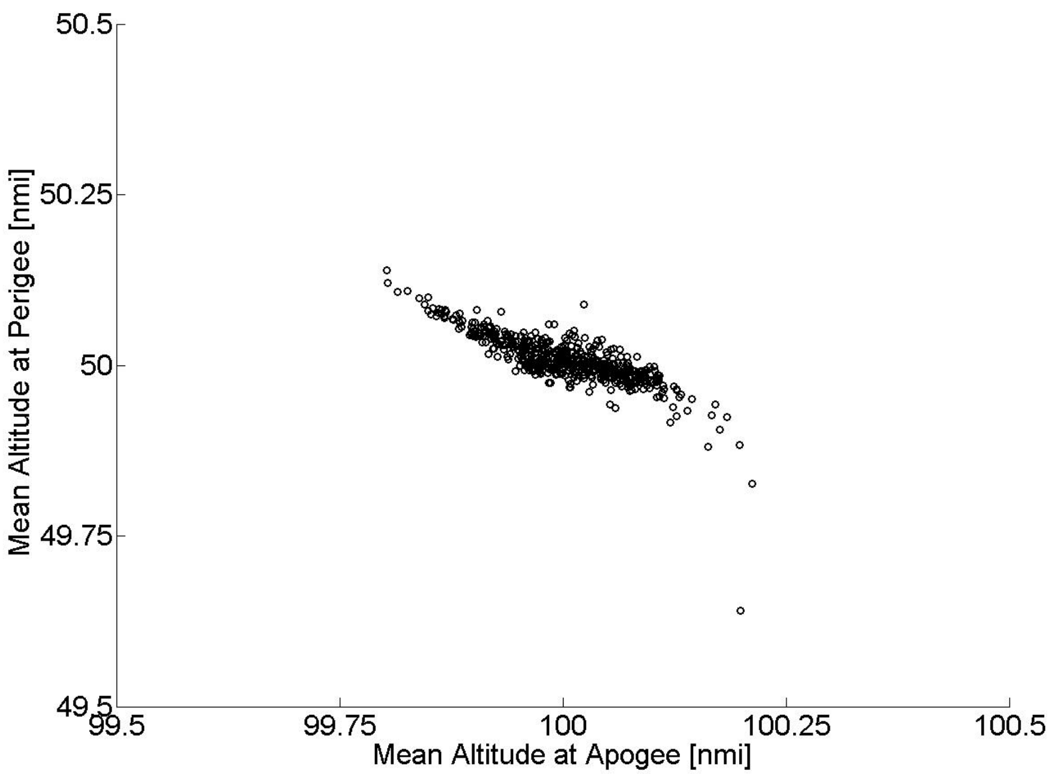


Figure 5.9: Final mean altitude of apsides for 98.2% of the cases for neural network Monte Carlo analysis.

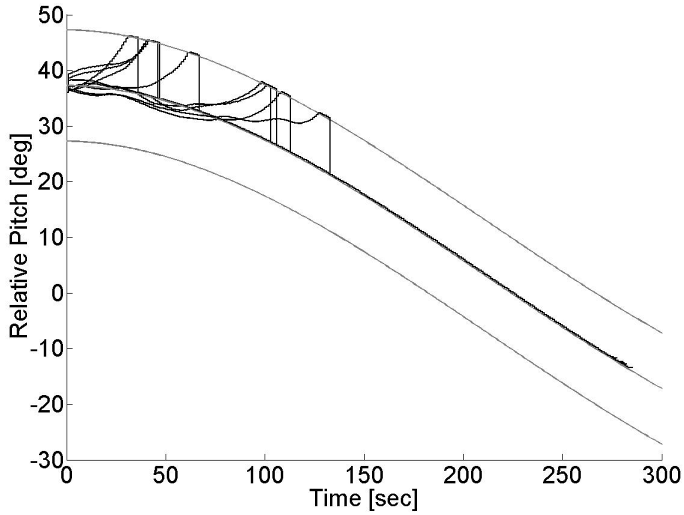


Figure 5.10: Failed neural network cases.

The cases that violated the boundaries on pitch were constrained to follow the nominal path. Figure 5.10 shows the failed cases and the divergence toward the upper bound. After following the nominal path to MECO, the failed cases all reached orbit. The orbits were considerably more eccentric on the order of 50 $\times$ 250 nmi, but wasted a great deal of mass along the way. Attempts to guide the vehicle back to the near-optimal path were unsuccessful because by the time the divergence was realized, the vehicle was too far from a path capable of targeting the desired orbit. An attempt was made to retrain the neural networks to 150%-σ so as to encompass a broader range of cases. The guidance algorithm was able to make the failed cases reach orbit within $\pm$5 nmi of the apsides altitudes. However, several of the cases that previously worked failed with the new networks. In addition, the cases that were successful showed significant reduction in targeting accuracy. The training set selection and subsequent NN performance shows that the algorithm is successful within most of the training domain, with a few outliers. The expanded training set showed characteristic increase in generalization within the training domain, though a significant decrease in accuracy resulted.

5.5 Computation Time

A goal of this research was to lower the computational cost of performing Monte Carlo simulations. The first step in comparing run times is to investigate the nominal optimization case. The variation in run time for this case is shown by the dashed line in Figure 5.11. For integration steps ≤ 1, the run time increases significantly. The P/C algorithm is near real-time with the trajectory. The trajectory time for the nominal optimization is represented by the solid line (not a function of the integration step). The intersection of the dashed and solid lines approximates the point at which the current P/C algorithm performs in the same time as the optimization – at an integration step size of approximately 0.8 sec.

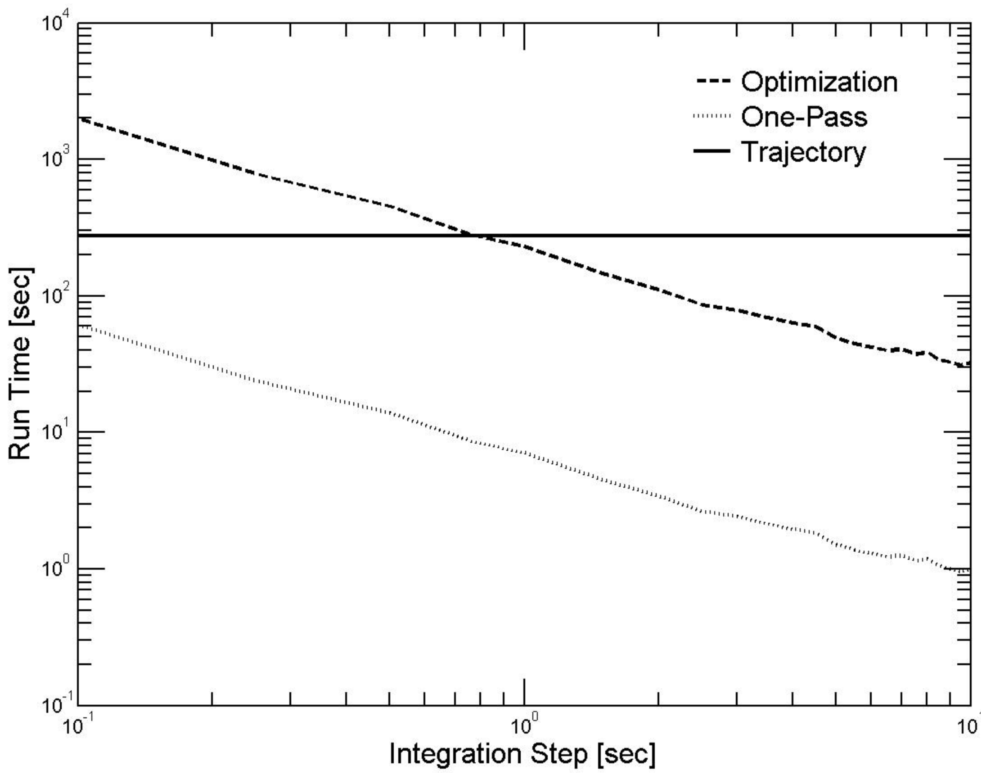


Figure 5.11: Optimization, one-pass, and trajectory times for various integration steps.

If fully implemented in the POST2 code, the P/C algorithm in the current formulation would run considerably faster. In an external state, the NN algorithm performs just slower than the computation time of one trajectory pass in POST2 with the difference coming only from the MATLAB-POST2 exchange. An implementation of the algorithm into POST2 would reduce the computation times to one trajectory pass as the network calculations are near-instantaneous. The run time for one pass of the trajectory is represented by the dotted line.

Implementation of the P/C algorithm would require considerable time and effort, but holds opportunities for improvement of the P/C design. Integrating the NN algorithm in to POST2, however, would not require an extensive process as a NN is basically a set of matrices. For the situation of a batch of 500 Monte Carlo cases, the P/C algorithm shows minimal computational advantage over the optimization as tailoring the algorithm for the specific problem can be difficult. A NN batch would require both time to run the Monte Carlo cases as well as training the networks. Development of 100 training cases is equivalent to running 100 optimized cases. The computation savings then comes from the ability to run 500 Monte Carlo cases an order of magnitude faster.

Chapter 6

Summary

Two candidate algorithms have been developed and evaluated for near-optimal injection weight for a rocket-based TSTO launch vehicle upper stage. Detailed descriptions of the methodologies behind the predictor-corrector (P/C) algorithm and the neural network (NN) modification were given. The Program to Optimize Simulated Trajectories 2 (POST2) was used for three degree-of-freedom trajectory analysis. The two algorithms have been compared to optimal solutions for 500 Monte Carlo cases including uncertainties in atmosphere, thrust, aerodynamics and initial state.

6.1 Conclusions

The algorithms developed in this thesis are designed for use within conceptual design tools. While design-oriented, the algorithms maintain the basic concepts of real vehicle guidance systems in handling uncertainties and generating pitch commands without specific knowledge of the exact atmosphere, thrust, and aerodynamics of flight. The algorithms drive POST2 externally using MATLAB, though both have potential for implementation into the POST2 source code. The P/C algorithm produced promising results for accurate orbit insertion but showed considerable degradation in injected weight. Computation time for the P/C proved to be a limiting factor in the algorithm design. The computation time for the external P/C algorithm was comparable to the trajectory time, which only holds an advantage over targeting and optimization for small integration steps. As an alternative to the inner loop targeting, a NN module was designed as an approximator to optimal cases in an attempt to decrease computation time. The NN produced pitch commands that closely followed counterpart optimization cases. Minimal ΔV-losses and rapid near-optimal simulations were highlights of the NN algorithm. Implementation of the algorithms into the POST2 code would further improve performance and allow for more complex inner loop designs. In conclusion, this study has developed two candidate methods for producing near-optimal trajectory guidance with potential computational savings over traditional targeting and optimization.

6.2 Future Work

The natural extension of the development of these algorithms would be implementation in the POST2 source code. The P/C algorithm would require special care when converting as the inner loop requires integration of the equations of motion. The benefit to such implementation is that the computation times would likely be reduced to allow for new inner loop control configurations. A general neural network guidance algorithm would not require as much effort because the NN computation is simple. However, full implementation of training algorithms would be very time intensive. Another extension of the work would involve the application of the algorithms to additional vehicle and trajectory formulations. While the algorithms are tailored toward the specific problem, the basic concepts are applicable to a broad range of problems.

Appendix A

Monte Carlo Dispersion Set

This appendix contains the actual distributions for the 500 Monte Carlo cases. The variable sets are developed using the mean and standard deviation values in Table 5.1 and the "randn" function in MATLAB. The atmospheric and wind dispersions are generated by GRAM99 with the initial date as J2000 and the initial location as the nominal initial longitude and latitude in Figure 3.8.

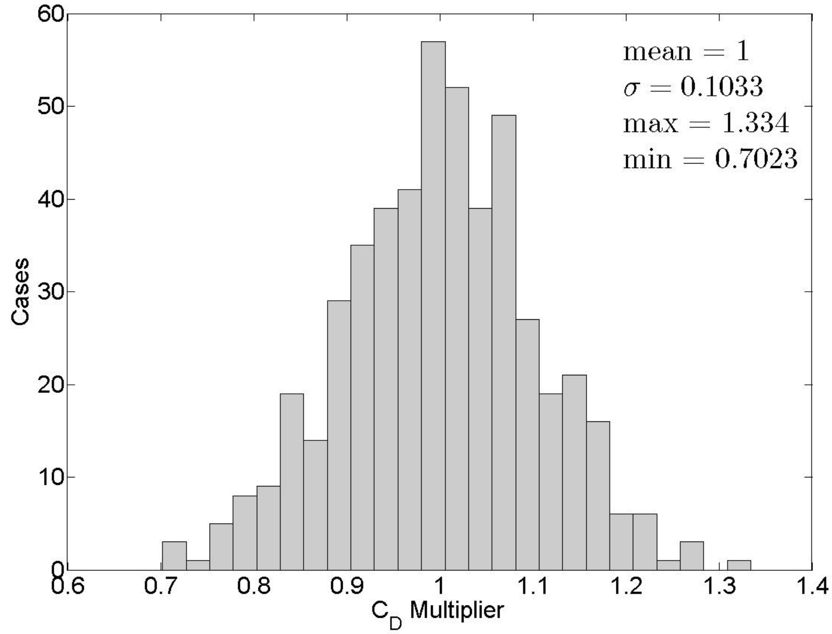


Figure A.1: Drag coefficient multiplier dispersion for 500 Monte Carlo cases.

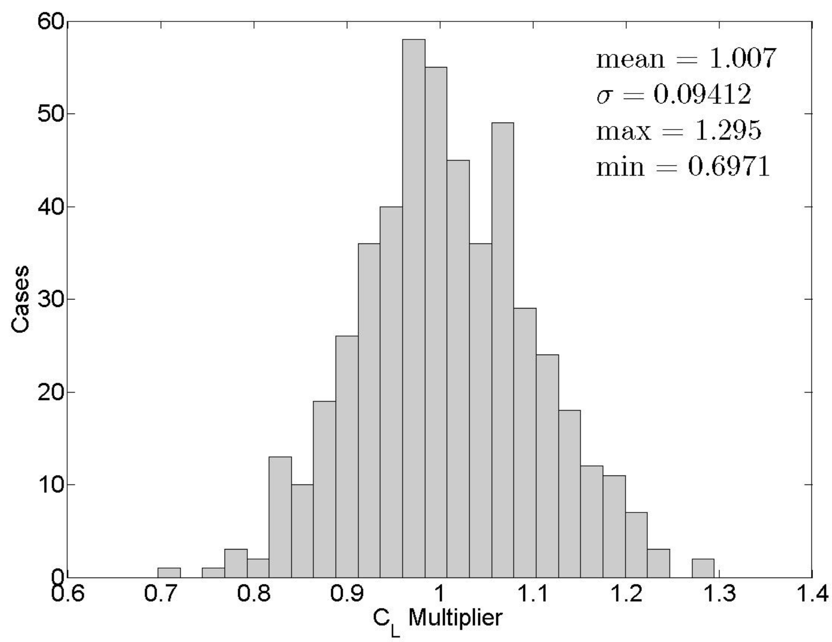


Figure A.2: Lift coefficient multiplier dispersion for 500 Monte Carlo cases.

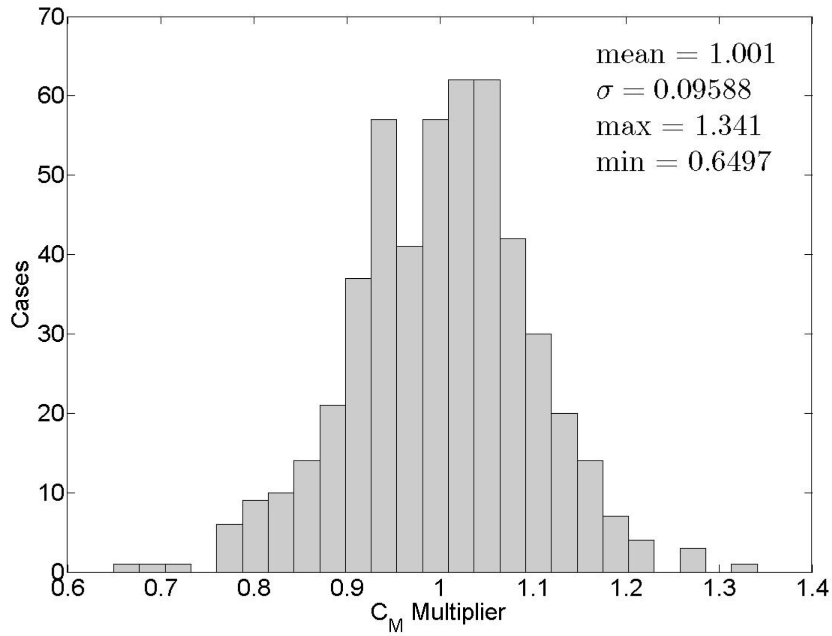


Figure A.3: Moment coefficient multiplier dispersion for 500 Monte Carlo cases.

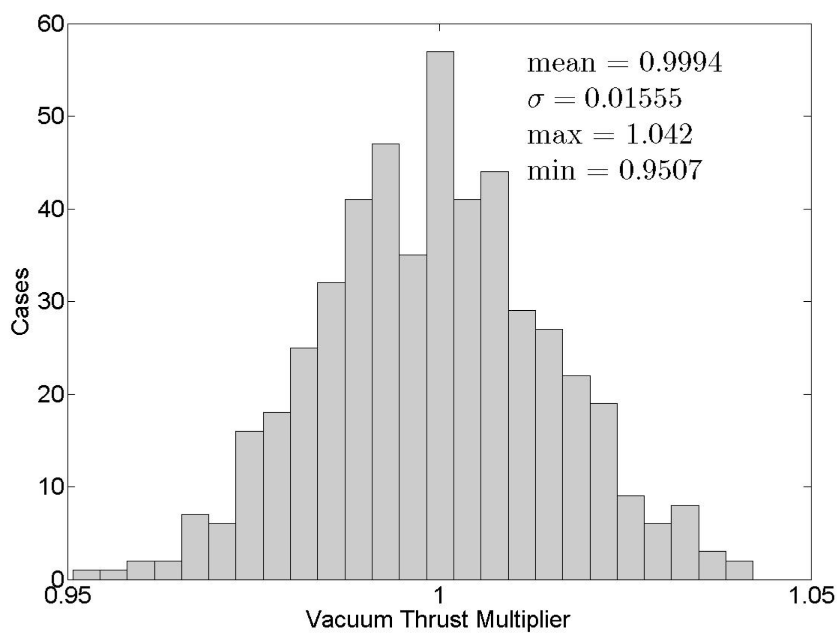


Figure A.4: Thrust multiplier dispersion for 500 Monte Carlo cases.

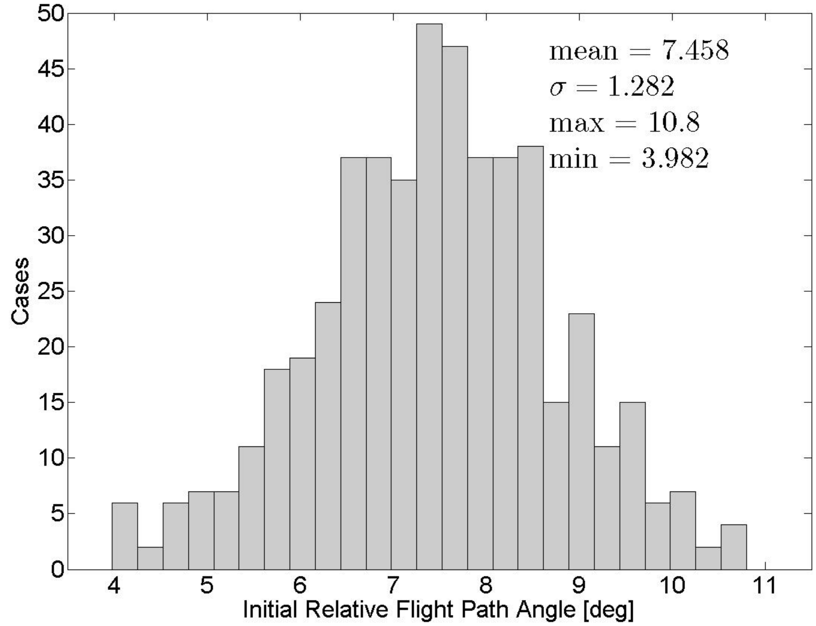


Figure A.5: Initial relative flight path angle dispersion for 500 Monte Carlo cases.

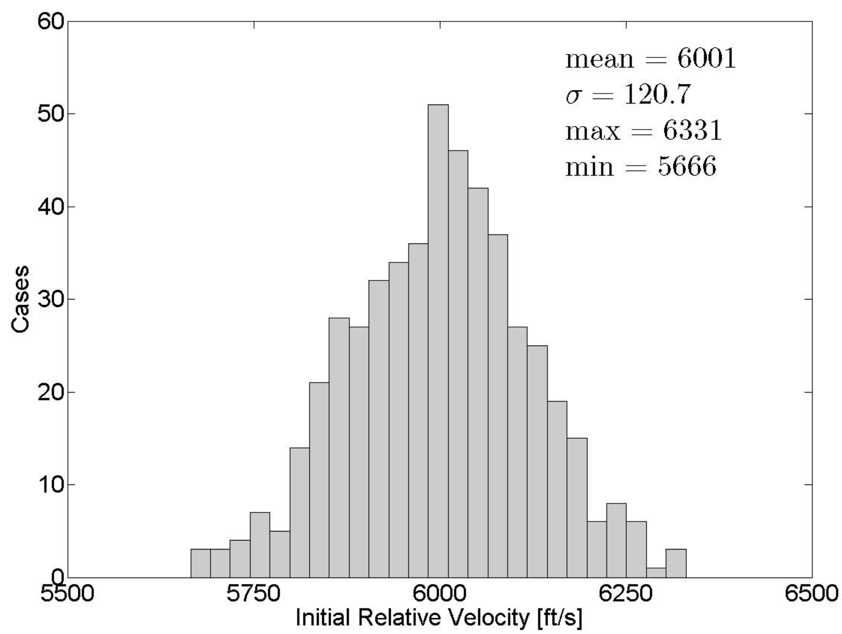


Figure A.6: Initial relative velocity dispersion for 500 Monte Carlo cases.

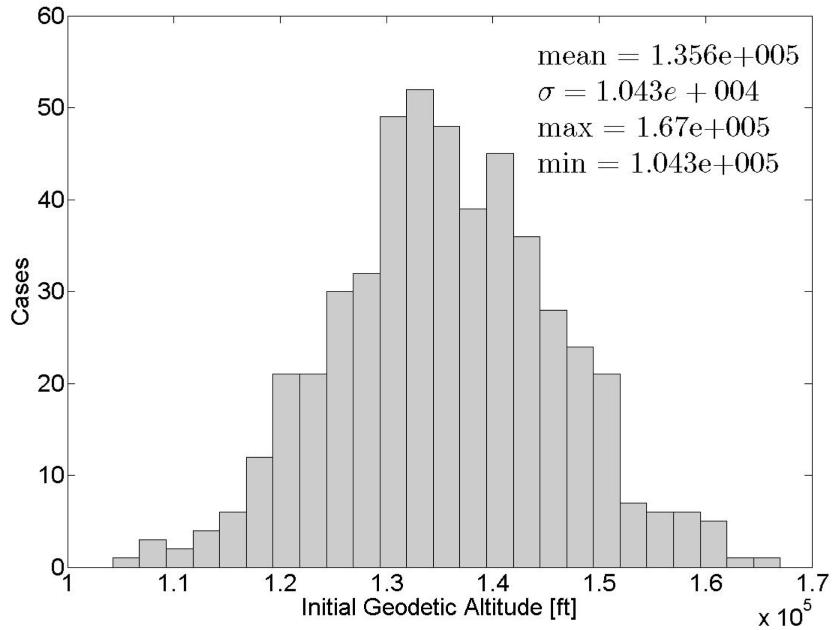


Figure A.7: Initial geodetic altitude dispersion for 500 Monte Carlo cases.

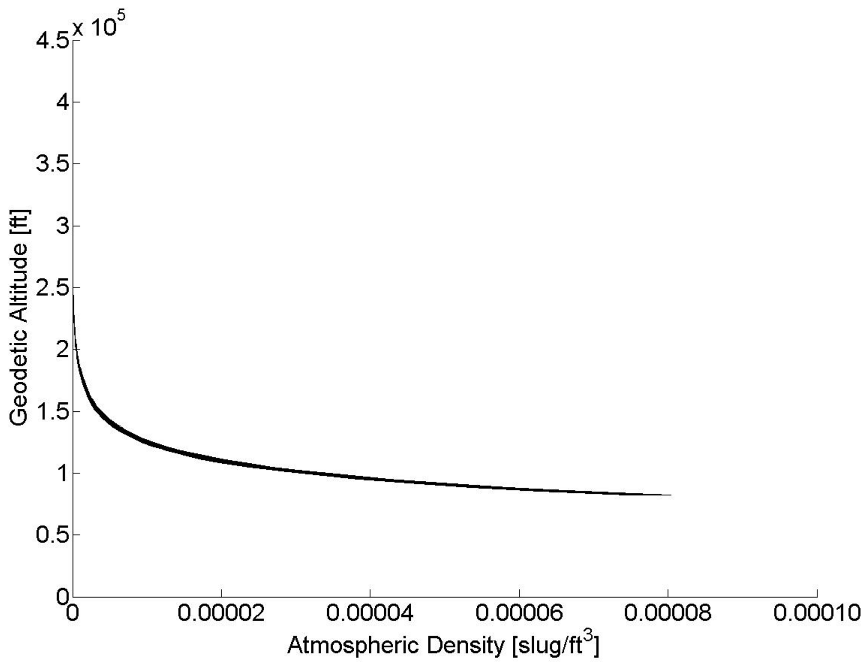


Figure A.8: GRAM99 atmospheric density dispersion for 500 Monte Carlo cases.

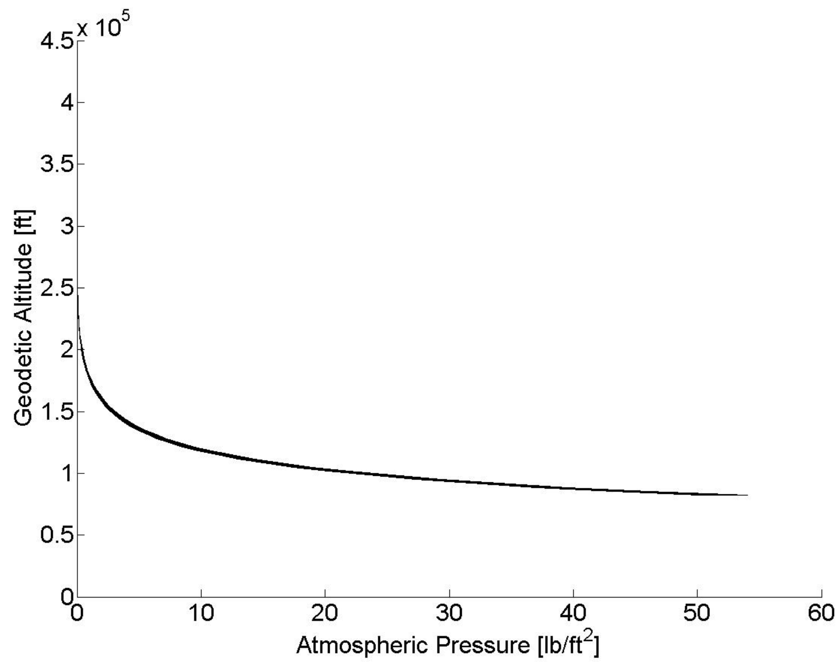


Figure A.9: GRAM99 atmospheric pressure dispersion for 500 Monte Carlo cases.

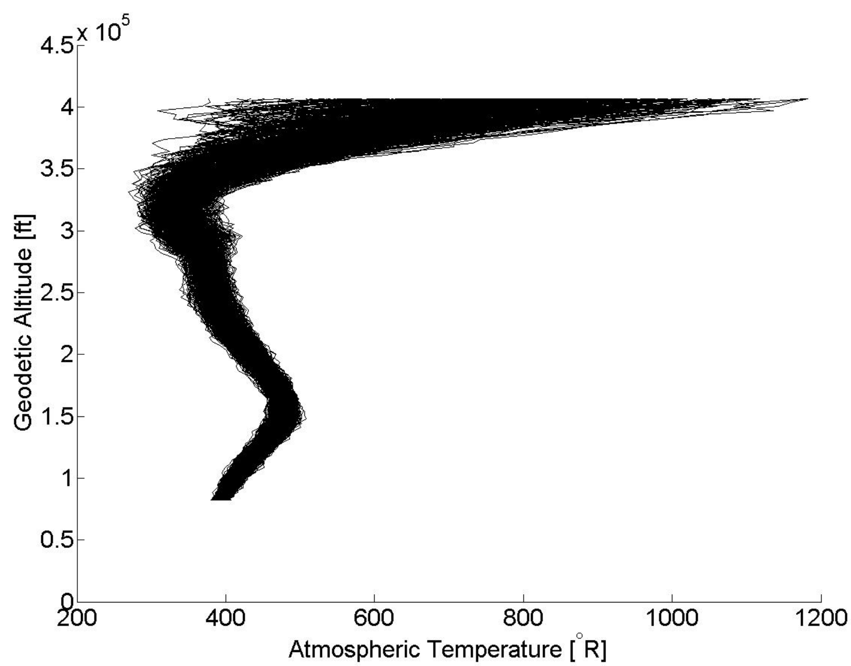


Figure A.10: GRAM99 atmospheric temperature dispersion for 500 Monte Carlo cases.

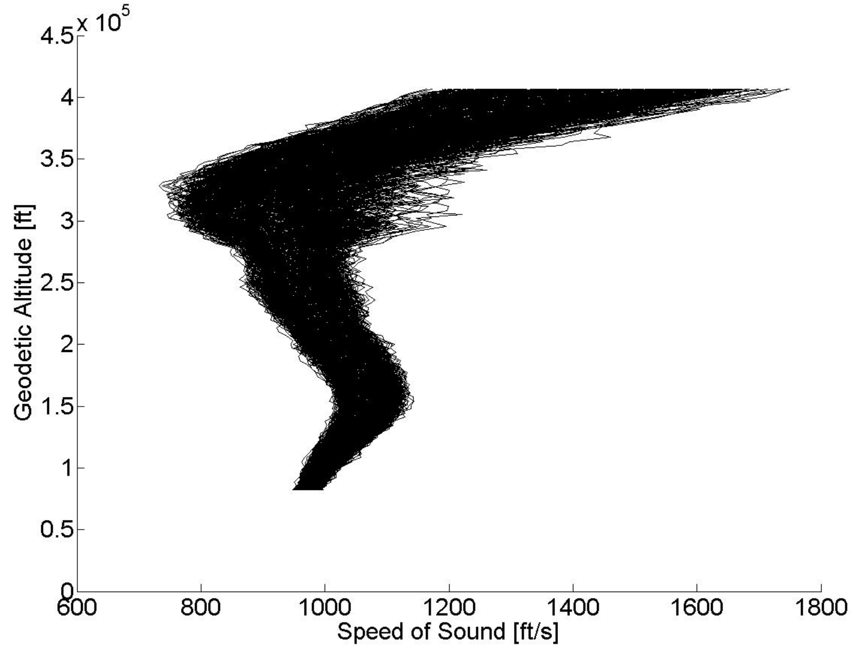


Figure A.11: GRAM99 speed of sound dispersion for 500 Monte Carlo cases.

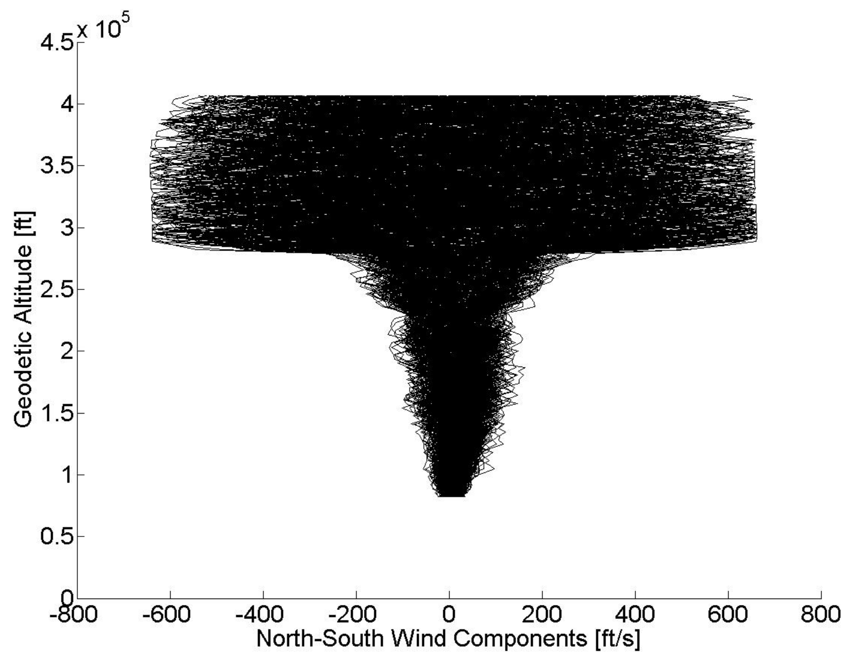


Figure A.12: GRAM99 north-south windspeed dispersion for 500 Monte Carlo cases.

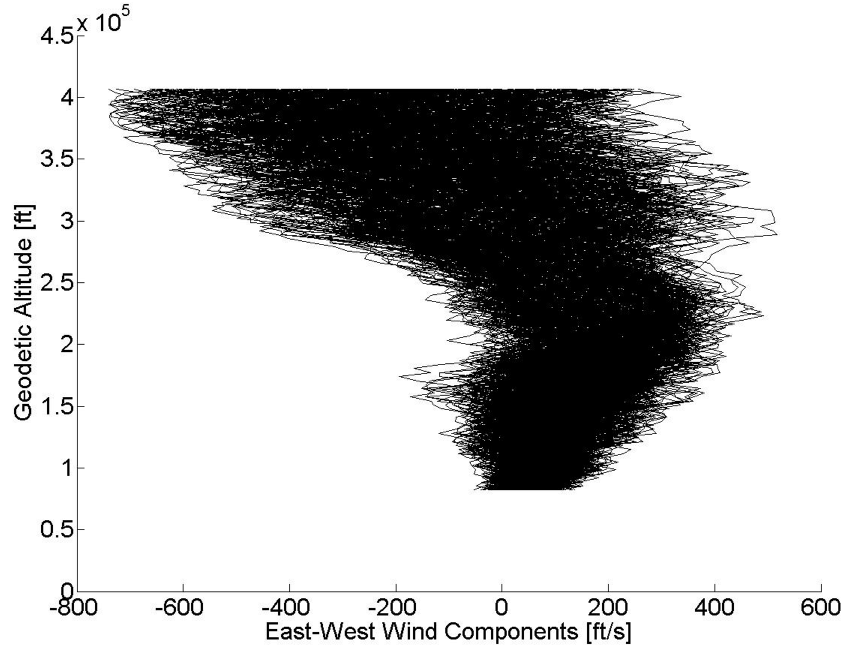


Figure A.13: GRAM99 east-west windspeed dispersion for 500 Monte Carlo cases.

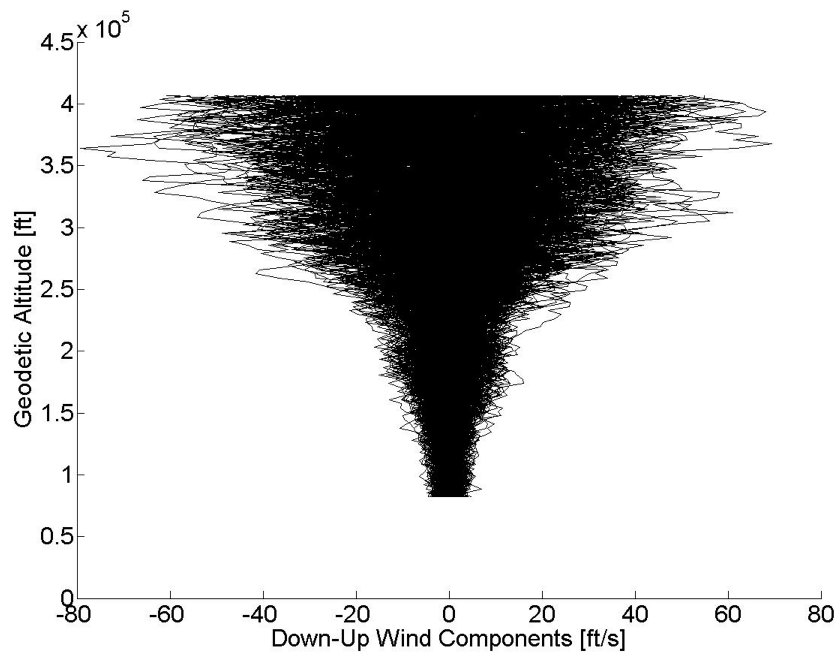


Figure A.14: GRAM99 down-up windspeed dispersion for 500 Monte Carlo cases.

Bibliography

[1] Brauer, G. L., Cornick, D. E., and Stevenson, R., "Capabilities and Applications of the Program to Optimize Simulated Trajectories," Tech. Rep. CR-2770, Martin Marietta Corporation, Denver, CO, 1977.

[2] Anderson, R. L., Aguirre, J. T., Martin, A. L., Striepe, S. A., Fisher, J. L., Brauer, G. L., and Engel, M. C., *Program to Optimize Simulated Trajectories (POST II) Volume I: Guide for New Users*, 2004, Version 1.1.6.G.

[3] Striepe, S. A., Powell, R. W., Desai, P. N., Queen, E. M., Brauer, G. L., Cornick, D. E., Olson, D. W., Peterson, F. M., Stevenson, R., Engel, M. C., Marsh, S. M., and Gromko, A. M., *Program to Optimize Simulated Trajectories (POST II) Volume II: Utilization Manual*, 2004, Version 1.1.6.G.

[4] Brauer, G. L., Cornick, D. E., Olson, D. W., Petersen, F. M., and Stevenson, R., "Program to Optimize Simulated Trajectories Volume I: Formulation Manual," Tech. Rep. MCR-87-583, NAS1-18147, Martin Marietta Corporation, Denver, CO, 1990.

[5] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming," Tech. Rep. 86-1, Systems Optimization Laboratory, Stanford University, Stanford, CA, 1998.

[6] Microsoft Corporation, *Microsoft Office Excel*, 2007.

[7] The Mathworks, *MATLAB R2007b*, 2007, Version 7.5.0342.

[8] Justus, C. G. and Johnson, D. L., *The NASA/MSFC Global Reference Atmospheric Model*, 1999, Version GRAM-99.

[9] TechnoSoft, Inc., Cincinnati, OH, *The Adaptive Modeling Language*.

[10] Demuth, H. and Beale, M., *Neural Network Toolbox For Use with MATLAB, User's Guide*, The MathWorks, Inc., 1999, Version 4.

[11] Hanson, J. M., "Advanced Guidance and Control Project for Reusable Launch Vehicles," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Denver, CO, 2000.

[12] Hanson, J. M., Coughlin, D. J., Dukeman, G. A., Mulqueen, J. A., and McCarter, J. W., "Ascent, Transition, Entry, and Abort Guidance Algorithm Design for X-33 Vehicle," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Boston, MA, 1998.

[13] Calise, A. J., Melamed, N., and Lee, S., "Design and Evaluation of a 3-D Optimal Ascent Guidance Algorithm," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 6, 2008, pp. 867–875.

[14] Chandler, D. C. and Smith, I. E., "Development of the Iterative Guidance Mode with Its Application to Various Vehicles and Missions," *AIAA Journal of Spacecraft and Rockets*, Vol. 4, No. 7, 1967, pp. 898–903.

[15] Horn, H. J., Chandler, D. C., and Buckelew, V. L., "Iterative Guidance Applied to Generalized Missions," *AIAA Journal of Spacecraft and Rockets*, Vol. 6, No. 1, 1969, pp. 4–8.

[16] Jaggers, R. F., "Asymmetrical Booster Ascent Guidance and Control System Design Study. Volume V: Space Shuttle Powered Explicit Guidance," Tech. Rep. CR-140191, NASA JSC, Houston, TX, 1974.

[17] Schleich, W. T., "The Space Shuttle Ascent Guidance and Control," *AIAA Guidance and Control Conference*, San Diego, CA, 1982, pp. 1–18.

[18] Su, S., *Advanced Guidance Algorithms for the Ares V Cargo Launch Vehicle*, Master's thesis, University of Missouri-Columbia, 2008.

[19] Nelson, D., *Qualitative and Quantitative Assessment of Optimal Trajectories by Implicit Simulation (OTIS) and Program to Optimize Simulated Trajectories (POST)*, Master's thesis, Georgia Institute of Technology, 2001.

[20] Queen, E. M. and O'Mara, T. M., "Comparison of Effects of Copropagated and Precomputed Atmosphere Profiles on Monte Carlo Trajectory Simulation," Tech. Rep. TM-102600, NASA Langley Research Center, Hampton, VA, August 1990.

[21] Hanson, J. M. and Hall, C. E., "Learning About Ares I from Monte Carlo Simulation," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, HI, 2008.

[22] Powell, R. W., Shaughnessy, J. D., Cruz, C. I., and Naftel, J. C., "Ascent Performance of an Air-Breathing Horizontal-Takeoff Launch Vehicle," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 4, 1991, pp. 834–839.

[23] Powell, R. W., Naftel, J. C., and Cruz, C. I., "Ascent Performance Issues of a Vertical-Takeoff Rocket Launch Vehicle," *Journal of Spacecraft and Rockets*, Vol. 28, No. 2, 1991, pp. 179–183.

[24] Powell, R. W., "Numerical Roll Reversal Predictor-Corrector Aerocapture and Precision Landing Guidance Algorithms for the Mars Surveyor Program 2001 Missions," *AIAA Atmospheric Flight Mechanics Conference*, Boston, MA, 1998.

[25] Lu, P., Sun, H., and Tsai, B., "Closed-Loop Endo-Atmospheric Ascent Guidance," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Monterey, CA, 2002.

[26] Graesslin, M., Telaar, J., and Schoettle, U., "Ascent and Reentry Guidance Concept Based on NLP-Methods," *Acta Astronautica*, Vol. 55, 2004, pp. 461–471.

[27] Jits, R. Y. and Walberg, G. D., "Blended Control, Predictor-Corrector Guidance Algorithm: An Enabling Technology for Mars Aerocapture," *Acta Astronautica*, Vol. 54, 2004, pp. 385–398.

[28] White, H., Stinchcombe, M., and Hornik, K., "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, 1989, pp. 359–366.

[29] White, H., "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," *Neural Networks*, Vol. 3, 1990, pp. 535–549.

[30] White, H. and Stinchcombe, M., "Approximating and Learning Unknown Mappings Using Multilayer Feedforward Networks with Bounded Weights," Economics working paper series, Department of Economics, UC San Diego, March 1990.

[31] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Representations by Back-Propagating Errors," *Nature*, Vol. 323, 1986, pp. 533–536.

[32] le Cun, Y., "A Theoretical Framework for Back-Propagation," *Proceedings of the 1988 Connectionist Models Summer School, Carnegie-Mellon University*, Pittsburgh, PA, 1988, pp. 21–28.

[33] le Cun, Y., "Generalization and Network Design Strategies," CRG-TR-89-4, Department of Computer Science, University of Toronto, June 1989.

[34] le Cun, Y. and Becker, S., "Improving the Convergence of Back-Propagation Learning with Second Order Methods," CRG-TR-88-5, Department of Computer Science, University of Toronto, September 1988.

[35] Battiti, R., "First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, Vol. 4, 1992, pp. 141–166.

[36] le Cun, Y., Bottou, L., Orr, G. B., and Muller, K.-R., "Efficient BackProp," *Lecture Notes in Computer Science, Volume 1524. Neural Networks: Tricks of the Trade*, Springer, 1998, pp. 9–50.

[37] de B. Harrington, P., "Sigmoid Transfer Functions in Backpropagation Neural Networks," *Analytical Chemistry*, Vol. 85, 1993, pp. 2167–2168.

[38] Duch, W. and Jankowski, N., "Survey of Neural Transfer Functions," *Neural Computing Surveys*, Vol. 2, 1999, pp. 163–212.

[39] Riedmiller, M. and Braun, H., "A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP Algorithm," *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, 1993, pp. 586–591.

[40] Hagan, M. T. and Menhaj, M. B., "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Transactions on Neural Networks*, Vol. 5, No. 6, 1994, pp. 989–993.

[41] Schaechter, D. B. and Rauch, H. E., "Applications of Neural Networks to Control Systems," *AIAA Guidance, Navigation and Control Conference*, Hilton Head Island, SC, 1992, pp. 378–384.

[42] Steinberg, M., "Potential Role of Neural Networks and Fuzzy Logic in Flight Control Design and Development," *AIAA Aerospace Design Conference*, Irvine, CA, 1992.

[43] Steck, J. E. and Rokhsaz, K., "Some Applications of Artificial Neural Networks in Modeling of Nonlinear Aerodynamics and Flight Dynamics," *AIAA Aerospace Sciences Meeting & Exhibit*, Reno, NV, 1997, pp. 378–384.

[44] Hagan, M. T. and Demuth, H. B., "Neural Networks for Control," *American Control Conference*, San Diego, CA, 1999, pp. 1642–1656.

[45] Plutowski, M. and White, H., "Selecting Concise Training Sets from Clean Data," *IEEE Transactions on Neural Networks*, Vol. 4, No. 2, 1993, pp. 305–318.

[46] Weller, P. and Avraam, M., "The Competitive Selection of Artificial Neural Network Training Sets Using an Arms Race Model," *10th Mediterranean Electrotechnical Conference*, Cyprus, 2000, pp. 465–468.

[47] Johnson, E. N., Calise, A. J., El-Shirbiny, H. A., and Rysdyk, R. T., "Feedback Linearization with Neural Network Augmentation Applied to X-33 Attitude Control," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Denver, CO, 2000.

[48] Johnson, E. N., Calise, A. J., and Corban, J. E., "Reusable Launch Vehicle Adaptive Guidance and Control Using Neural Networks," *AIAA Guidance, Navigation and Control Conference and Exhibit*, Montreal, Canada, 2001.

[49] ud Din Qazi, M., Linshu, H., and Elhabian, T., "Rapid Trajectory Optimization Using Computational Intelligence for Guidance and Conceptual Design of Multistage Space Launch Vehicles," *AIAA Guidance, Navigation and Control Conference and Exhibit*, San Francisco, CA, 2005.

[50] NOAA, NASA, and USAF, *US Standard Atmosphere 1976*, 1976, Washington, DC.

[51] Wertz, J. R. and Larson, W. J., editors, *Space Mission Analysis and Design*, Third Edition, Microcosm Press and Kluwer Academic Publishers, El Segundo, CA, 1999.

[52] Hill, P. and Peterson, C., *Mechanics and Thermodynamics of Propulsion*, Second Edition, Addison-Wesley Publishing Company, Reading, MA, 1992.

[53] Sutton, G. P. and Biblarz, O., *Rocket Propulsion Elements*, Seventh Edition, John Wiley & Sons, New York, 2001.

[54] Kinney, D. J., “Aero-Thermodynamics for Conceptual Design,” *42nd AIAA Aerospace Sciences Meeting and Exhibit*, Reno, NV, 2004.

[55] Cliff, E. M., “Trajectory Optimization Techniques,” AOE 5974 Class Notes, Virginia Polytechnic Institute and State University, Spring 2008.

[56] Bryson, A. E. and Ho, Y., *Applied Optimal Control*, Revised Printing, Hemisphere Publishing Corporation, Washington, DC, 1975.

[57] Kohler, W. and Johnson, L., *Elementary Differential Equations with Boundary Value Problems*, Pearson/Addison Wesley, 2004.

[58] Philen, M., “Numerical Methods,” AOE/MATH 4404 Class Notes, Virginia Polytechnic Institute and State University, Spring 2008.

[59] Fausett, L., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice Hall, New Jersey, 1994.