

Development of a Generic Guidance Simulation Utilizing
Second Order Neighboring Optimal Control

by

Patrick Neil Hinchy

B.S.A.E. May 1995, Embry-Riddle Aeronautical University

A Thesis submitted to

The Faculty of

The School of Engineering and Applied Science
of the George Washington University in partial satisfaction
of the requirements for the degree of Master of Science

May 4, 1998

Thesis directed by

Dr. Daniel D. Moerder

Guidance and Control Branch

Flight Dynamics and Control Division

NASA Langley Research Center

Abstract

A numerical method is developed for implementing a second-order neighboring optimal control scheme. This feedback control scheme allows control of a system for following an optimal trajectory in the presence of modeling errors and external disturbances. The method is developed for single phase optimal control problems with no state inequality constraints. The algorithm is divided into two parts: calculation and storage of feedback gains, and utilization of the gains for controlling the plant during trajectory simulations.

Two example problems are examined for validation of the numerical implementation; a simple nonlinear problem and the maximum mass orbit insertion of a generic Single-Stage-to-Orbit launch vehicle. Results show that second-order feedback provides significant improvement over first-order feedback, but that certain numerical difficulties have to be overcome before it can be robustly applied to any generic optimal control trajectory.

Contents

Abstract	ii
Table of Contents	iii
Nomenclature	v
1 Introduction	1
2 Trajectory Optimization and Neighboring Optimal Control	4
2.1 Trajectory Optimization	4
2.1.1 Problem Statement	4
2.1.2 Direct Optimization Approach	6
2.2 Neighboring Optimal Control	8
2.2.1 Variational Development	8
2.2.2 Numerical Development	9
3 Zermelo Test Problem	19
3.1 General Problem Formulation	19
3.2 NOC Problem Formulation	20
3.3 Numerical Results	21
3.3.1 Gain Calculation	21
3.3.2 Simulation Performance	23
4 SSTO Orbit Insertion Problem	28
4.1 Problem Formulation	28
4.2 Numerical Results	31
5 Conclusions and Recommendations	40
References	42

Appendix A: Control Law Implementation	43
A.1 Simulation Logic	43
A.2 Guidance Logic	47
Appendix B: SSTO Model Improvements	52

Nomenclature

Roman Symbols

a	Speed of Sound, ft/s
A_e	Exit Area of SSME Derivative Engine, ft ²
c_D	Drag Coefficient
c_L	Lift Coefficient
C	State/Control Inequality Constraints
D	Drag, lbf
$flag012$	Flag Vector for Control Law Expansions
$flagMNI$	Flag for Time Indexing Method
g	Acceleration Due to Gravity, ft/s ²
g_0	Acceleration of Gravity at Sea Level, ft/s ²
ind	Integer Index Into Stored Trajectory Data
I_{sp}	Specific Impulse, s
J	Objective Function
L	Lift, lbs
m	Number of Controls
m	SSTO Vehicle Mass, slugs
M	Mach Number
n	Number of States
n_b	2-Norm of Terminal Condition Error
n_p	Number of Free Parameters
$N_{engines}$	Number of SSME Derivative Engines
N_t	Number of Discretization Intervals
$NDIV$	Integer Number of Timestep Subdivisions
NLP	Non Linear Programming
p	Optimization Parameter Vector
p	Atmospheric Pressure, lbs/ft ²
p_0	Atmospheric Pressure at Sea Level, lbs/ft ²

q	Dynamic Pressure, lbs/ft ²
\mathcal{P}	Free Parameter Inequality Constraints
r	Radial Distance from the Center of the Earth to Vehicle Center of Mass, ft
r_E	Radius of Earth, ft
R_{gas}	Gas Constant for Air
S	Reference Area, ft ²
t	Time, sec
T	Thrust, lbs
T_{vac}	Vacuum Thrust of SSME Derivatives Engine, lbs
\mathcal{T}	Temperature, °R
u	Control Vector
v	Airspeed, ft/s
v	Adjoined State and NOC Parameter Vectors
x	State Vector
z_s	NOC Static Parameter Vector

Greek Symbols

α	Attack-Angle, rad
γ	Flight-Path Angle, rad
ϵ	Numerical Tolerance
η	SSME Derivative Engine Throttle
θ	Longitude, rad
μ	Bank Angle, rad
ρ	Atmospheric Density, slugs/ft ³
ρ_0	Atmospheric Pressure at Sea Level, slugs/ft ³
τ	Trajectory Duration, sec
ϕ	Latitude, rad
Φ	Cost Function
ψ	Heading Angle, rad
Ψ	Boundary Conditions

ω Angular Velocity of the Earth, rad/s

Subscripts

0 Initial Value
f Final Value
in Index Value
TAB Tabular Aerodynamic Coefficient

Acronyms

DOF Degree Of Freedom
JIAFS Joint Institute for the Advancement of Flight Sciences
MNI Minimum Norm Indexing
NASA National Aeronautics and Space Administration
NLP Non-Linear Programming
NOC Neighboring Optimal Control
OTIS Optimal Trajectories via Implicit Simulation
PCA Point of Closest Approach
SSME Space Shuttle Main Engine
SSTO Single Stage To Orbit
SSV Single Stage Vehicle
WCC Wall Clock Correction

Chapter 1: Introduction

For many of today's aerospace vehicle systems, there is a strong need for achieving optimal performance in areas such as fuel consumption, payload carrying capability, and minimum time trajectories. For this reason, optimal control theory holds great potential for the aerospace industry.

One major problem with exploiting optimal control for such nonlinear systems is that solutions can be difficult to obtain, both analytically and numerically. Exact analytical solutions are normally impossible except for very simple systems. Since optimal control problems are nonlinear two-point boundary value problems, solutions using iterative numerical methods can have convergence problems. Also, calculating a numerical solution can be time consuming for even moderately complex problems.

The solution of a trajectory optimization problem consists of an optimal state trajectory and the associated optimal control history. These solutions are useful for characterizing and studying the performance potential of the system, but are not directly useful for controlling the system in cases where the plant is perturbed from the optimal trajectory. Thus, additional development is needed before an optimal control solution can form the basis of a control system.

For a system physically traversing a trajectory, the controller must compensate for modeling errors and disturbance inputs. The control history obtained from an optimal control solution does not reflect these. Correction for disturbances and modeling errors is most practically done by using a control signal which is a function of the current system state, known as feedback control.

There are three main approaches to providing feedback control for following an optimal trajectory. The first is to continuously recalculate the optimal control from the current state using numerical optimization methods. The main drawback of this method is that the reliability of convergence of the iterative algorithm used for the numerical optimization would be key to the success of the scheme. It is very difficult to guarantee convergence for all possible scenarios the plant might encounter.

A second approach is to derive approximations to optimal control laws which can

be implemented in feedback form. A considerable amount of work has appeared in the literature [1], in which the presence of fast and slow modes in the plant dynamics is exploited to break the optimal control problem into analytically tractable subproblems. An example of this is the longitudinal dynamics of an aircraft, with the fast short period mode and the slower phugoid mode. The formalism employed consists of identifying a small parameter with the fast dynamics and approximating the plant model, and hence the necessary conditions for optimal control, as a power series about zero in the small parameter. This is referred to as singular perturbation since the fast dynamics degenerate into algebraic constraints when the parameter is reduced to zero. In this approach, the small parameter is frequently assumed to have a nominal value of one, so that the accuracy of the approximation gives way to the convenience of simplifying the problem [1]. In any case, analytic solutions are extremely customized to the mission, requiring new analysis and code for each new problem.

A third method for providing feedback control to follow an optimal trajectory is called neighboring optimal control (NOC). The basic concept behind neighboring optimal control is to construct a Taylor series expansion of the control about the nominal control history, as a function of state perturbations. While this concept generally cannot be applied to optimal control solutions obtained using the calculus of variations, it is directly applicable to the discrete optimal control solutions considered in this document.

The use of direct numerical perturbation for synthesizing NOC control laws was reported and demonstrated by Seywald and Kumar [2]. Their approach was to calculate a family of state and control trajectories emanating from a linearly independent set of initial state perturbations which spanned the space containing the state vector. From these trajectories, the gradient of the control with respect to the state could be calculated at each instant in time by the matrix form of the chain rule. These Jacobian matrices comprised a time varying set of first-order feedback gains. This approach required that the perturbations be reinitialized when the collection of state vector histories emanating from the set of perturbations became rank deficient. This condition has been found to occur frequently in practice.

Wetzel [3] examined direct numerical calculation of NOC gains using a second-order expansion. In this work, the tabulation of trajectories emanating from sets of initial perturbations was abandoned. Instead, control perturbations were calculated directly from state perturbations at each instant. This was the equivalent of restarting Seywald and Kumar's technique at each instant. Because of numerical shortcomings, the examination of performance of second-order NOC was inconclusive.

The numerical approach used here is essentially the same technique used by Wetzel, except using improved algorithms to solve the optimal control problem. Additionally, an algorithm has been developed to implement the NOC feedback control scheme for simulating the traversal of the plant along the optimal trajectory in the presence of external disturbances. This document presents the development of a numerical method for implementing a second-order neighboring optimal control scheme. The method is developed for single phase optimal control problems with no state inequality constraints.

Two example problems are considered to illustrate the application of the NOC feedback scheme. The first is a nonlinear, minimum time problem involving steering a boat to a specific location in the presence of position dependent currents, known in the literature as the Zermelo problem. The second and more complex problem is a neighboring optimal control guidance scheme for the orbit insertion trajectory of a generic Single-Stage-to-Orbit launch vehicle.

The results of these tests validate the neighboring optimal approach, as well as the benefits of using a second-order expansion over a first-order expansion. While the numerical implementation needs refinements before it can be practically applied, the theory behind it is shown to be sound. Once these refinements have been made, neighboring optimal control will be a very useful tool for real-time simulation of optimal trajectories.

Chapter 2: Trajectory Optimization and Neighboring Optimal Control

In this chapter, the class of continuous-time optimal control problems considered in this document is defined, and theory for calculating expansions for synthesizing feedback controllers for trajectories in a neighborhood of the optimum is developed. The necessary conditions for optimality are briefly considered, and the temporally discretized problem formulation actually used for calculations is given. The neighboring optimal control problem is then defined in variational terms, and the computational limitations of the variational approach are discussed. Finally, the technique of calculating feedback gains by direct numerical differentiation of trajectories, as employed in this document, is given in detail.

2.1 Trajectory Optimization

As defined in [4], the objective of optimal control theory is to determine the control signals that will cause a process to satisfy some physical constraints and at the same time minimize (or maximize) some performance criterion.

2.1.1 Problem Statement

The typical optimal control problem, as it applies to most aerospace applications, is stated mathematically as follows. For a specified initial time t_0 , determine the initial states $x(0) \in \mathcal{R}^n$, control history $[u(\tau) \in \mathcal{R}^m \ 0 \leq \tau \leq 1]$, and free parameters $p \in \mathcal{R}^j$ to minimize the scalar performance index

$$\mathcal{J} = \Phi[x(1), p] \tag{2.1}$$

subject to the system dynamical constraints

$$\dot{x} - f(x, u, p) = 0, \text{ some } x(0) \text{ specified} \tag{2.2}$$

and to the boundary conditions

$$\Psi[x(1), p] = 0. \quad (2.3)$$

The control function $u(t)$ is assumed to be integrable on the interval $0 \leq t \leq 1$. Also, Φ , f , and Ψ are assumed to be twice differentiable in their arguments. These are combined into an augmented objective function for the system as follows:

$$\bar{\mathcal{J}} = \Phi[x(1), p] + \nu^T \Psi[x(1), p] + \int_0^1 \lambda^T [f(x, u, p) - \dot{x}] dt, \quad (2.4)$$

where ν and λ are Lagrange multiplier vectors.

Note that this problem formulation is nominally for unity duration. It is generalized to treat free time problems by treating the trajectory duration, τ , as an element of the p vector, and scaling the plant dynamics according to

$$\dot{x} = f(x, u, p) = \tau \tilde{f}(x, u, p_1), \quad p^T = [\tau \quad p_1^T]. \quad (2.5)$$

Using a development similar to that in [5], the variational necessary conditions for this problem are

$$\dot{x} = f(x, u, p), \quad x(0) \text{ specified}, \quad (2.6)$$

$$\dot{\lambda} = -f_x^T \lambda, \quad (2.7)$$

$$f_u^T \lambda = 0, \quad (2.8)$$

$$\lambda^T(1) = [\Phi_x + \nu^T \Psi_x]_{t=1}, \quad (2.9)$$

$$\Psi[x(1), p] = 0, \quad (2.10)$$

$$\int_0^1 \lambda^T f_p dt = - [\Phi_p + \nu^T \Psi_p]_{t=1}. \quad (2.11)$$

At this point, except for the simplest cases, the problem is solved by satisfying these necessary conditions using numerical techniques.

There are several difficulties with this solution approach. First, from equation 2.7, the adjoint dynamics have eigenvalues which are reflections about the imaginary axis of those associated with the linearization of the original system, as seen in equation 2.7. This means that, for a dissipative plant, the adjoint dynamics are unstable.

Also, the adjoint equations effectively double the number of states in the system. Finally, since this solution only satisfies the necessary conditions for a minimum, the solution may not actually be a local minimum.

2.1.2 Direct Optimization Approach

An alternative to the variational approach for solving optimal control problems is direct minimization of the objective function using a lumped-parameter approximation of the control function and state trajectory. In the formulation used in this document, the system dynamics are discretized into N_t equally spaced intervals via a midpoint integration rule to yield

$$\left. \begin{aligned} \eta(x_{k+1}, x_k, \bar{u}_k, p) &= x_{k+1} - x_k - \frac{1}{N_t} f(\bar{x}_k, \bar{u}_k, p) \\ &= 0 \end{aligned} \right\} \quad (2.12)$$

with

$$\left. \begin{aligned} t_k &= \frac{N_t \tau}{k}, \\ x_k &= x(t_k), \\ \bar{x}_k &= \frac{1}{2}(x_k + x_{k+1}), \\ \bar{u}_k &= u\left(\frac{t_{k+1} + t_k}{2}\right). \end{aligned} \right\} \quad (2.13)$$

These are concatenated to form the system of constraints

$$z(x_1, \dots, x_{N_t+1}; \bar{u}_1, \dots, \bar{u}_{N_t}, p) = \begin{bmatrix} \eta(x_2, x_1, \bar{u}_1, p) \\ \eta(x_3, x_2, \bar{u}_2, p) \\ \vdots \\ \eta(x_{N_t+1}, x_{N_t}, \bar{u}_{N_t}, p) \end{bmatrix} = 0. \quad (2.14)$$

Additional state/control inequality constraints of the form

$$\mathcal{C}(x, u, p) \leq 0, \quad (2.15)$$

are introduced, which are discretized to yield

$$\begin{bmatrix} c_i(\bar{x}_1, \bar{u}_1, p) \\ c_i(\bar{x}_2, \bar{u}_2, p) \\ \vdots \\ c_i(\bar{x}_{N_t}, \bar{u}_{N_t}, p) \end{bmatrix} \leq 0 \quad i = 1, \dots, n_c \quad (2.16)$$

or

$$\mathcal{C}_i \leq 0 \quad i = 1, \dots, n_c. \quad (2.17)$$

Parameter inequality constraints are implemented as

$$\mathcal{P}(p) \leq 0. \quad (2.18)$$

Constraints of the form (2.17, 2.18) were neglected in the variational development of 2.1.1, because they introduce additional complication in the calculation of time and costates in the necessary conditions [5]. This complication was not necessary to make the main point of 2.1.1, which is that the variational solution approach is impractical for many problems. Note that treatment of such constraints using the direct approach, described here, is a straightforward matter.

The boundary conditions are restated as

$$\Psi(x_{N_t+1}, p) = 0 \quad (2.19)$$

and concatenated with the system dynamics (2.12) to form the equality constraints

$$Z = \begin{bmatrix} z \\ \Psi \end{bmatrix} = 0. \quad (2.20)$$

The inequality constraints (2.17, 2.18) are concatenated to form

$$Y = \begin{bmatrix} \mathcal{C}_1 \\ \vdots \\ \mathcal{C}_{n_c} \\ \mathcal{P} \end{bmatrix} \leq 0. \quad (2.21)$$

Note that (2.21) is $N_t \cdot n_c + n_p$ constraints. Finally, the cost is expressed as

$$\mathcal{J} = \Phi(x_{N_t+1}, p). \quad (2.22)$$

The free variables can be arranged as

$$X^T = [x_1^T \ u_1^T \ x_2^T \ u_2^T \ \dots \ x_{N_t}^T \ u_{N_t}^T \ x_{N_t+1}^T \ p^T]. \quad (2.23)$$

Discretizing the problem in this manner reduces the optimal control problem to a constrained parameter optimization problem, which can be solved by generic nonlinear programming (NLP) methods [6].

2.2 Neighboring Optimal Control

2.2.1 Variational Development

The variational solution to a neighboring optimal control problem begins with the necessary conditions for the solution of a baseline optimal control problem, as given in Equations 2.6 through 2.11. However, to simplify the following development, the parameter vector will be held fixed, eliminating Equation 2.11.

Small perturbations in the initial state $\delta x(t_0)$ and the terminal conditions $\delta \Psi$ produce perturbations $\delta x(t)$, $\delta \lambda(t)$, $\delta u(t)$, and $d\nu$ found by linearizing (2.6) through (2.10) about the baseline trajectory.

$$\delta \dot{x} = f_x \delta x + f_u \delta u, \quad \delta x(0) \text{ specified}, \quad (2.24)$$

$$\delta \dot{\lambda} = -(f_x^T \lambda)_x \delta x - f_x^T \delta \lambda - (f_x^T \lambda)_u \delta u, \quad (2.25)$$

$$(f_u^T \lambda)_x \delta x + f_u^T \delta \lambda + (f_u^T \lambda)_u \delta u = 0, \quad (2.26)$$

$$\delta \lambda(1) = [(\Phi_{xx} + (\nu^T \Psi_x)_x) \delta x + \Psi_x^T d\nu]_{t=1}, \quad (2.27)$$

$$\delta \Psi = [\Psi_x \delta x]_{t=1}. \quad (2.28)$$

Equations 2.24 through 2.28 define a linear two-point boundary-value problem since the coefficient matrices are evaluated on the baseline optimal trajectory. If $(f_u^T \lambda)_u$ is nonsingular for $0 \leq t \leq 1$, equation 2.26 may be solved for $\delta u(t)$ in terms of $\delta \lambda(t)$ and $\delta x(t)$:

$$\delta u(t) = -(f_u^T \lambda)_u^{-1} [(f_u^T \lambda)_x \delta x + f_u^T \delta \lambda]. \quad (2.29)$$

The boundary-value problem may be solved using a backward sweep method as discussed in [5] to find $\delta \lambda(t)$ as a function of $\delta x(t)$. Substituting this relationship into (2.29) yields a feedback control law of the form

$$\delta u(t) = \Lambda \delta x(t). \quad (2.30)$$

The variational treatment of the neighboring optimal control problem here yields a first-order feedback control law. Extension of this approach to derive control expressions of second or higher order is not simple. An additional concern is the numerical difficulty of the variational solutions themselves, as discussed in Section 2.1.

2.2.2 Numerical Development

Numerical solution methods provide for a more practical implementation of a neighboring optimal control feedback law. There are two phases to implementing a numerical neighboring optimal control system: calculation and storage of the feedback gains and utilization of the gains for control calculation during trajectory simulations.

Numerical Gain Calculation

Once again, numerical calculation of neighboring optimal control feedback gains starts with the solution of a baseline optimal control problem. One starts with a simplified optimal control problem of the form

$$\mathcal{P}_c : \left. \begin{array}{l} \text{minimize } \Phi[x(t_f), p] \text{ subject to} \\ 0 = \dot{x} - f(x, u, p), \\ 0 = x(t_0) - x_0, \quad x_0 \text{ given,} \\ 0 = \Psi[x(t_f), p], \end{array} \right\} \quad (2.31)$$

where the final time is broken out of the parameter vector p , though it is understood that, computationally, it is treated as part of p during optimization. The reason for this prominence of t_f is its role in the feedback system, as discussed later. At this point the notation for the initial time is changed back to t_0 , to allow for examining specific time instants t_k in the following discussion.

If this problem has a locally unique solution, the solution is a final time t_f^* , and state and control trajectories, $\{x^*(t), u^*(t), t_0 = 0 \leq t \leq t_f^*\}$. In general, the constraints $\dot{x} - f(x, u)$ are nonholonomic, and the control history cannot be represented by a function of a finite number of parameters. However, if attention is restricted to the control at the instant t_0 and the terminal time t_f , the problem becomes

$$\mathcal{P}_c^0(t_0) : \left. \begin{array}{l} \text{determine } u_0^* = u^*(t_0), t_f^* \text{ such that} \\ \{x^* \times u^* \times t_f^*\} = \arg \min \Phi[x(t_f), p] \text{ subject to} \\ 0 = \dot{x} - f(x, u, p), \\ 0 = x(t_0) - x_0, \quad x_0 \text{ given,} \\ 0 = \Psi[x(t_f), p]. \end{array} \right\} \quad (2.32)$$

In \mathcal{P}_c^0 , $u_0^*(x_0)$ and $t_f^*(x_0)$ are well-defined algebraic functions. Thus, the behavior of the solution $\{u_0^* \times t_f^*\}$ can be examined for small perturbations of x_0 using Taylor series expansions. To second-order, the expansions for the elements of the control vector u_0^* , $k = 1, \dots, m$, and t_f^* are

$$\left. \begin{aligned} u_k^*(x_0 + \delta x_0) &= u_k^*(x_0) + \frac{\partial u_k^*}{\partial x_0} \delta x_0 + \frac{1}{2} \delta x_0^T \frac{\partial^2 u_k^*}{\partial x_0 \partial x_0^T} \delta x_0 + \text{H.O.T.}, \\ t_f^*(x_0 + \delta x_0) &= t_f^*(x_0) + \frac{\partial t_f^*}{\partial x_0} \delta x_0 + \frac{1}{2} \delta x_0^T \frac{\partial^2 t_f^*}{\partial x_0 \partial x_0^T} \delta x_0 + \text{H.O.T.}, \end{aligned} \right\} \quad (2.33)$$

which can be approximated by the finite difference expressions

$$\frac{\partial \eta}{\partial x_i} \approx \frac{\eta(x + \Delta_i) - \eta(x - \Delta_i)}{2\epsilon_i}, \quad (2.34)$$

$$\frac{\partial^2 \eta}{\partial x_i \partial x_j} \approx \frac{\eta(x + \Delta_{+i+j}) - \eta(x + \Delta_{+i-j}) - \eta(x + \Delta_{-i+j}) + \eta(x + \Delta_{-i-j})}{4\epsilon_i \epsilon_j}, \quad (2.35)$$

where

$$\left. \begin{aligned} \Delta_i &= \epsilon_i e_i, \\ \Delta_{\pm i \pm j} &= \pm \epsilon_i e_i + \pm \epsilon_j e_j, \end{aligned} \right\} \quad (2.36)$$

and e_i is the unit vector in the i^{th} state direction.

The first three terms in each of (2.33) provide a suboptimal feedback control law which is valid at time t_0 . To provide a control law over the entire trajectory, the partial derivatives in (2.33) can be calculated for a sequence of problems $\mathcal{P}_c^0(t_k)$ where $\{t_k, k = 0, \dots, N_t - 1\}$ is a discretization of the problem time line. At instants not directly represented in the set of solutions $\{\mathcal{P}_c^0(t_k)\}$, the feedback and nominal quantities can be approximated by interpolation in time. The resulting first and second-order feedback systems are

$$\left. \begin{aligned} u[x, t_{\text{in}}(t)] &= u_0^*(t_{\text{in}}(t)) + F_{u1}(t_{\text{in}}(t)) \delta x(x, t_{\text{in}}(t)), \\ t_f[x, t_{\text{in}}(t)] &= t_f^*(t_{\text{in}}(t)) + F_{t1}(t_{\text{in}}(t)) \delta x(x, t_{\text{in}}(t)), \\ \delta x(x, t_{\text{in}}(t)) &= x - x_0^*(t_{\text{in}}(t)), \end{aligned} \right\} \quad (2.37)$$

and

$$\left. \begin{aligned} u[x, t_{\text{in}}(t)] &= u_0^* + F_{u1} \delta x + \delta x^T F_{u2} \delta x, \\ t_f[x, t_{\text{in}}(t)] &= t_f^* + F_{t1} \delta x + \delta x^T F_{t2} \delta x, \\ t_{\text{in}}(t) &= t - F_{t1} \delta x - \delta x^T F_{t2} \delta x, \end{aligned} \right\} \quad (2.38)$$

where the arguments in (2.38) have been dropped for simplicity. Because the expression for t_{in} in (2.38) is not in closed form, it must be determined numerically. Methods for doing this are discussed in the description of the simulation method.

For the implementation being studied here, the baseline optimal control problem is discretized as discussed in Section 2.1. The resulting control discretization, when applied to $\{\mathcal{P}_c^0(t_k)\}$, greatly simplifies the problem, but the resulting optimized trajectories do not satisfy the Principle of Optimality from optimal control theory [5]. This principle states that subarcs of a given optimal state trajectory must coincide with solutions of control optimization problems restricted to those subarcs. Violation of this principle produces variation of the optimal final time t_f^* obtained from subarcs of the optimal trajectory, i.e.

$$t_f^*|_{\mathcal{P}_c^0(t_i)} \neq t_f^*|_{\mathcal{P}_c^0(t_k)} \quad i \neq k. \quad (2.39)$$

Naturally, there is also a variation in x^* . To account for this variation, instead of using an equally spaced temporal discretization, define

$$\bar{\Delta}_t^T = \left[(\Delta_t)_1 : (\Delta_t)_2 : \cdots : (\Delta_t)_{N_t} \right], \quad \left. \begin{array}{l} \\ (\Delta_t)_k = \frac{t_f - t_k}{N_t - k}. \end{array} \right\} \quad (2.40)$$

Because the baseline optimal trajectory changes with truncation, the baseline must be recalculated at each instant at which perturbations will be calculated. This gives rise to a “nominal” trajectory for the form

$$x^* = \left[x_0 : x^*(t_1)|_{\mathcal{P}_c^0(t_0)} : x^*(t_2)|_{\mathcal{P}_c^0(t_1)} : \cdots : x^*(t_f)|_{\mathcal{P}_c^0(t_{N_t-1})} \right]. \quad (2.41)$$

For many problems, it would be useful to be able to modify the control law to account for variations in static model parameters. For example, when controlling a launch vehicle, it would be useful to be able to modify the control law based on changes in payload mass, engine efficiency, launch site winds, and so on. This can be accomplished with the NOC control laws by adjoining the static parameter vector to the state vector,

$$v = \begin{bmatrix} x \\ z_s \end{bmatrix} \quad z_s \text{ is the static parameter vector,} \quad (2.42)$$

and performing the expansions with v instead of x .

The parameter correction can also be considered as a first-order correction of the state feedback control law:

$$\left. \begin{aligned} u(x, z_s, t) &= u(x, t) + \delta z_s^T \frac{du(x, t)}{dz_s}, \\ \frac{du}{dz_s} &= \frac{\partial u}{\partial z_s} + \frac{\partial^2 u}{\partial z_s \partial x^T} \delta x. \end{aligned} \right\} \quad (2.43)$$

This leads to the conclusion that it is possible to arbitrarily pick which states and parameters will be expanded to 0^{th} , 1^{st} , or 2^{nd} order.

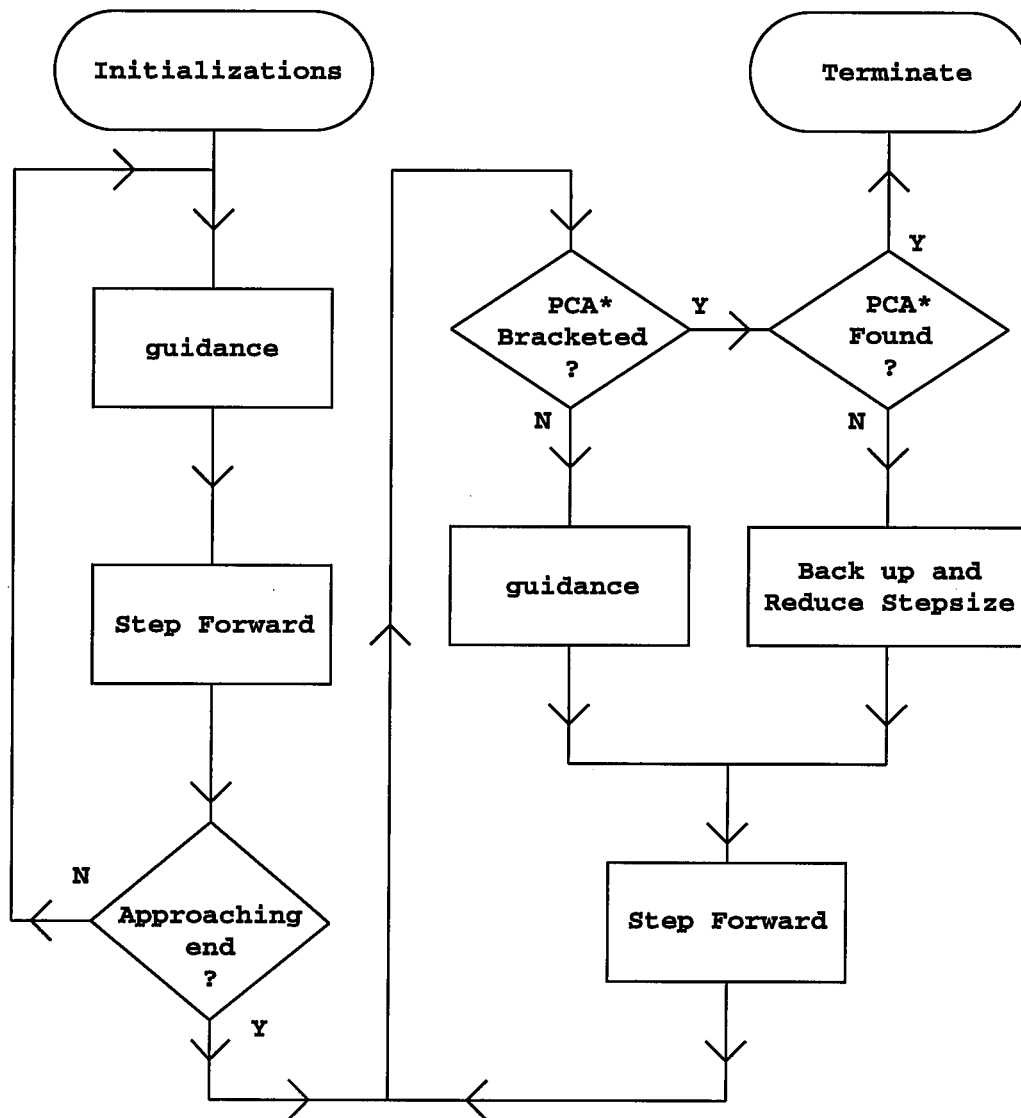
In terms of actual computation, a method has been developed which uses nonlinear programming methods to generate nominal and perturbed trajectory solutions, and saves key outputs for each perturbation in separate files. This is done to allow the calculations to be independently performed on a number of computers simultaneously. This reduces the workload on each individual computer, as well as the start-to-finish time of performing the calculations. The calculations are performed in two steps:

1. Calculate nominal trajectories beginning from i^{th} nodes, for $i = 1, \dots, Nt$. Store NLP gradient data, trajectories, and nonperturbed terms in the difference expressions (2.34,2.35).
2. For each $i = 1, \dots, Nt$, calculate perturbed terms in (2.34,2.35). Save up-to-date gradient data for i^{th} truncated trajectory, and perturbations of u and τ .

Control Law Implementation

The simulation method developed to implement the neighboring optimal control scheme has the structure represented by the flow chart in Figure 2.1. The following is a summary of the implementation method. For a more detailed description of the implementation, refer to Appendix A.

Implementation of the guidance law for simulation begins with initialization of the simulation variables. The control vector is then calculated using the guidance feedback law, and the plant is integrated forward in time. The first of four termination tests, represented graphically in Figure 2.2, is evaluated to determine if the terminal manifold is being approached. This test simply determines if the Euclidian norm of



*PCA: Point of Closest Approach to Terminal Condition

Figure 2.1: Simulation Logic Flowchart

the terminal condition error is less than some fixed value ϵ_0 . If this is not the case, the steps of calculating the control vector and integrating forward are continued until the terminal manifold is being approached.

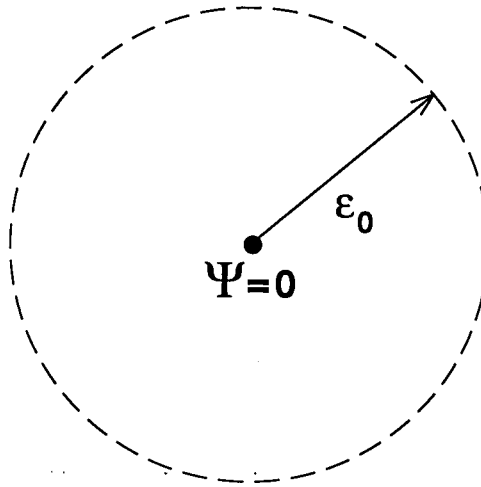


Figure 2.2: First Termination Test

If the first termination test has been passed, the next step is to determine if the Point of Closest Approach (PCA) to the terminal condition manifold is located between the current and previous plant positions. If not, continue calculating the control vector and integrating forward until it is.

Once the PCA is between the current and previous plant locations, the remaining three termination tests are performed. The first of these, represented by Figure 2.3, begins with fitting a quadratic Hermite polynomial between the current and previous plant locations, using state rate data from the previous location. A line search is then performed along the polynomial to find the location which minimizes the Euclidian norm of the terminal condition error. If this error norm is less than some ϵ_1 then this test has been passed, and the next test is performed. The next test, represented by Figure 2.4, begins at the location found from the previous test. A linesearch is performed in the direction of the gradient of the Euclidian norm of the terminal

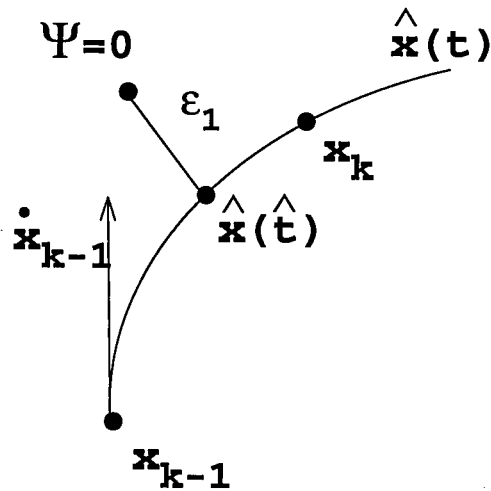


Figure 2.3: Second Termination Test

condition error, again looking for the position which minimized the Euclidian norm of the error. If this norm is less than some ϵ_2 then this test has been passed, and the final termination test is performed.

The final termination test, represented by Figure 2.5, begins with fitting a cubic Hermite polynomial between the current and previous plant locations, using state rate data at both locations. A minimization problem is then solved, which locates the position along the polynomial and the position on the terminal manifold which minimize the Euclidian norm of the terminal condition error. If this error norm is less than some ϵ_3 then the final termination test has been passed, and the simulation is ended.

If the PCA is between the current and previous locations, but does not satisfy all of the termination tests, the simulation is backed up to the previous location, the step size is halved, and the step is retaken. The simulation then returns to checking if the PCA is between the current and previous locations. The guidance logic, represented by the flowchart in Figure 2.6, begins by calculating the Minimum Norm Index (MNI)

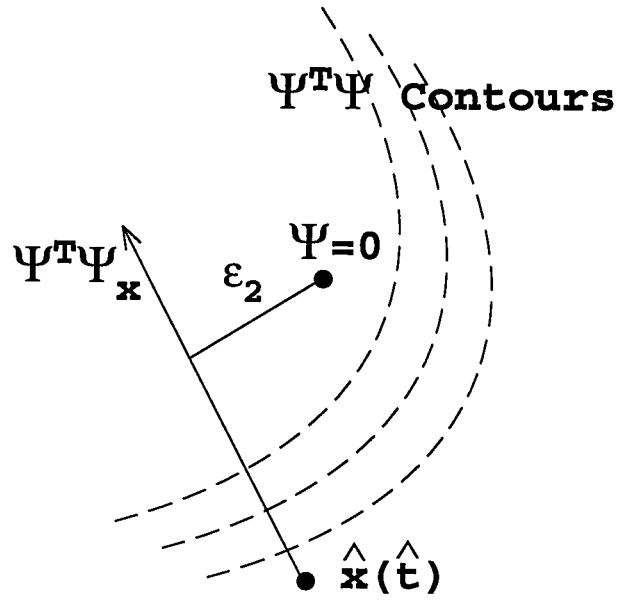


Figure 2.4: Third Termination Test

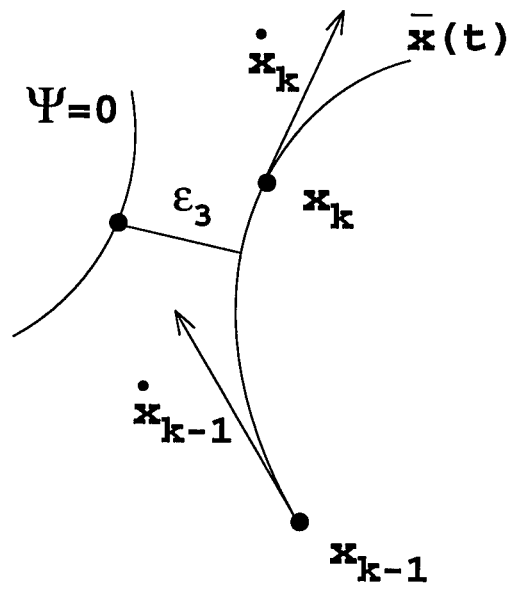


Figure 2.5: Final Termination Test

time:

$$t_{\text{MNI}} = \operatorname{argmin}_t \|x - x^*(t)\|. \quad (2.44)$$

This is done regardless of which time indexing method is being used, because it provides an easy method for determining when to re-interpolate the gain matrix elements, as well as a method for calculating the integration time step.

The major source of complication in the guidance routine comes from this calculation of index time, t_{in} , which is the time value used for performing table lookups for the guidance parameters. One approach attempts to estimate the time progressed into the nominal trajectory, using a Taylor series expansion for t_f to correct wall-clock time to describe the current instant. The other approach locates the time index value at which $x^*(t_{\text{in}})$ is closest to x as measured by the Euclidean norm. This latter approach is motivated by the idea that a Taylor expansion is most accurate for perturbations closest to the nominal value being expanded about. The former approach will be referred to as “wall-clock correction” (WCC), and the latter as “minimum norm indexing” (MNI), which was first examined in [3]. In the current implementation, the user can choose which of these two time indexing methods to use.

The next step in the guidance law is to calculate the integration time step. This is not used by the guidance law itself, but is simply passed to the simulation. It is calculated as the nominal timestep corrected for changes in the trajectory duration.

To reduce computation time, the feedback gain matrix elements are only periodically re-interpolated from the stored gain histories. This is done whenever the current state position passes from one stored trajectory point to the next.

The final step in the guidance logic is to use the index time and gain matrices to calculate the control values, using the NOC feedback law.

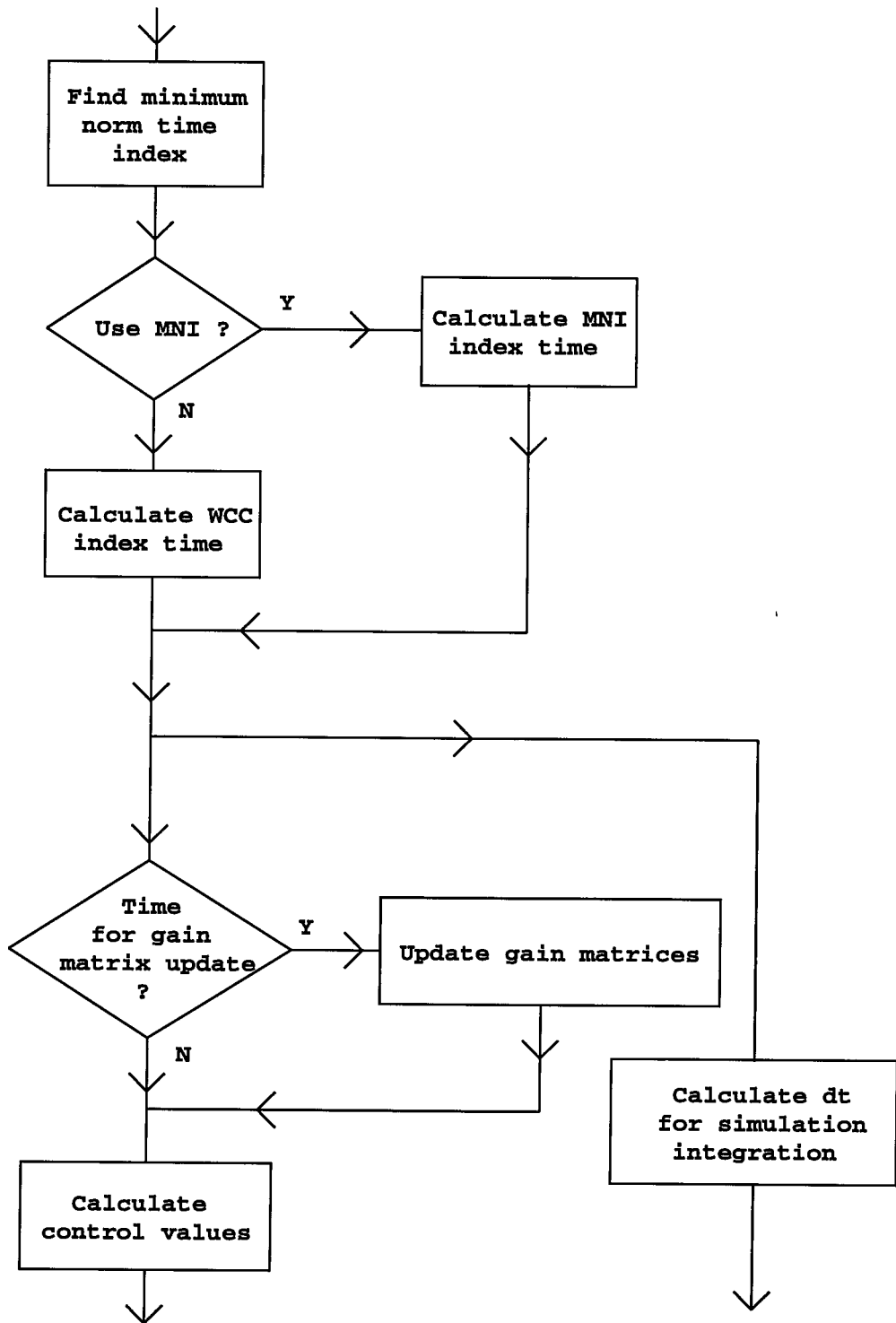


Figure 2.6: Guidance Logic Flowchart

Chapter 3: Zermelo Test Problem

In the course of developing a second-order neighboring optimal control feedback formulation, it is helpful to have a simple nonlinear example problem to test the gain calculation routines. This problem should have a small number of states and controls, with relatively simple nonlinear plant dynamics. The Zermelo problem of Bryson and Ho [5] is well suited for this purpose.

3.1 General Problem Formulation

In the Zermelo problem, a ship with constant speed V in the direction $\theta(t)$ relative to a fixed Cartesian coordinate system having x and y axes must travel through a region of strong position dependent currents. Specifically, the current is in the x -axis direction, and is negatively proportional to the y -axis position. The problem is to specify the control history $\theta(t)$ such that the ship is driven to the origin in minimum time given an initial starting position x_0 and y_0 . Since perturbation methods are being employed, it was decided to make the terminal condition a circle rather than a point, to make the perturbed problem near the end of the trajectory easier to solve. Basically, this is true because a circle provides a bigger target than a point. A formal problem statement is given as follows: Determine the control history $\theta(t)$ and final time t_f which minimize the performance index

$$\mathcal{J} = \Phi(x(1), p) = t_f, \quad (3.1)$$

subject to the equations of motion for the ship

$$\begin{bmatrix} \dot{x} - (V \cos(\theta) - Y_p y) \\ \dot{y} - V \sin(\theta) \end{bmatrix} = 0, \quad (3.2)$$

and to the boundary conditions

$$\Psi = \begin{bmatrix} x(0) - x_0 \\ y(0) - y_0 \\ x^2(1) + y^2(1) - R^2 \end{bmatrix} = 0, \quad (3.3)$$

where $x_0 = 4.0$, $y_0 = -4.0$, $V = 1.0$, and $Y_p = 1.0$. To determine just how sensitive the gains were to the size of the target circle, solutions were found for terminal radii of 0.5, 0.1, 0.05, 0.01, and 0.0 units.

3.2 Neighboring Optimal Control Problem Formulation

In addition to calculating feedback gains for states, gains were calculated for the velocity of the boat, V , and the water current velocity proportionality constant, Y_p . Since this is a test case, it was desired to vary everything that might possibly have an effect on the results. In addition to using different terminal radii, the following variations were made:

- Perturbation size: 1%, 3%, and 5% of the current nominal state value, with an absolute lower limit based on solution precision tolerance
- Node density: 50 and 20 discretization intervals

A comparison of the cost function values of trajectories from perturbed initial conditions using the feedback control law, versus the values from true optimal trajectories from those same initial conditions was also considered. This is done to compare performance between first and second-order feedback, to ensure that second-order feedback does indeed provide substantially improved performance over first-order feedback. Currently, the simulation uses a modified Adams-Peace integration scheme [7] to integrate the plant dynamics, whereas the NLP algorithm uses the implicit midpoint Euler discretization (2.12, 2.13). Therefore, when comparing feedback solutions to true optimal solutions, the feedback solution will be obtained using the NLP algorithm to solve an optimal control problem with the control history determined by the feedback control law, rather than using the simulation algorithm. This will ensure that all differences are due to the control law, not differences in the integration schemes.

3.3 Numerical Results

The first item to be considered is the untruncated nominal solution. The solution for a terminal radius of 0.0 with 50 discretization intervals is presented in Figure 3.1. The arrows represent values of the control variable, θ , along the trajectory. This solution agrees with the one in Bryson and Ho [5] for similar initial and final conditions.

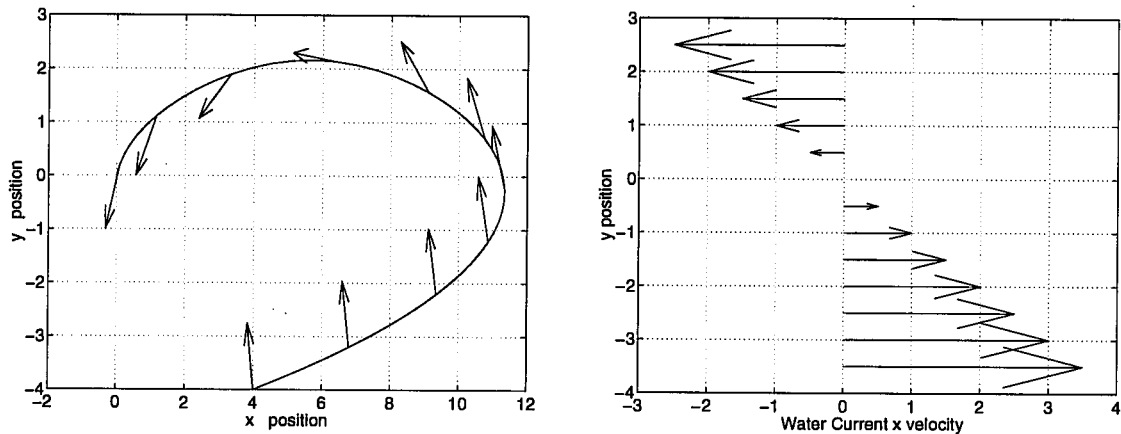


Figure 3.1: Nominal Trajectory, $R = 0.00$

3.3.1 Gain Calculation

In Figure 3.2 the effect of changing the number of discretization intervals on a typical gain history with 3% perturbations and a terminal radius $R = 0.50$ is presented. It can be seen that fewer intervals reduce the size of the peaks in the gain histories. In Figure 3.3 the effect of changing the size of the perturbations on a typical gain history with 50 discretization intervals and a terminal radius $R = 0.50$ is presented. It can be seen that the gains are nearly identical until close to the final time, possibly due to numerical problems associated with convergence. However, because this problem is close to the end of the trajectory, the error is acceptable for simulation. In Figures 3.4 and 3.5 the effect of the terminal radius size on typical gain histories with 3% perturbations and 50 discretization intervals is presented. Figure 3.4 is typical

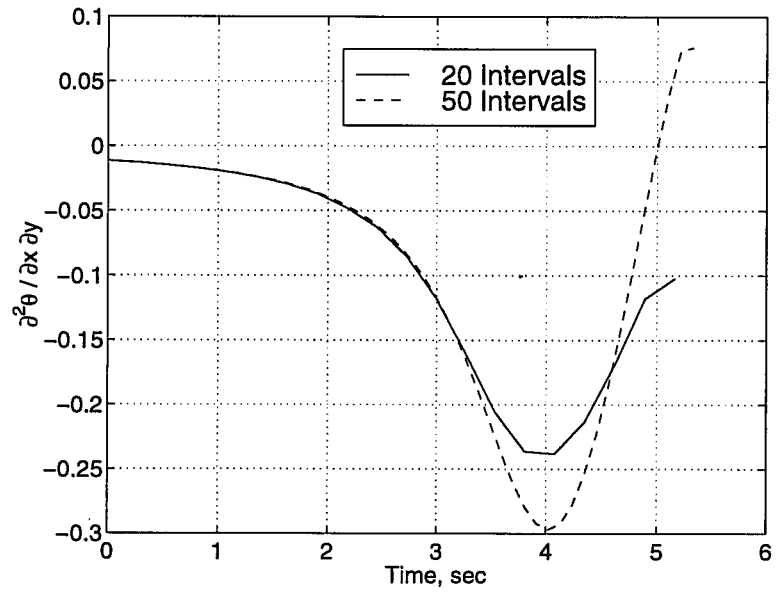


Figure 3.2: Effect of Number of Nodes on $\partial^2\theta/\partial x\partial y$

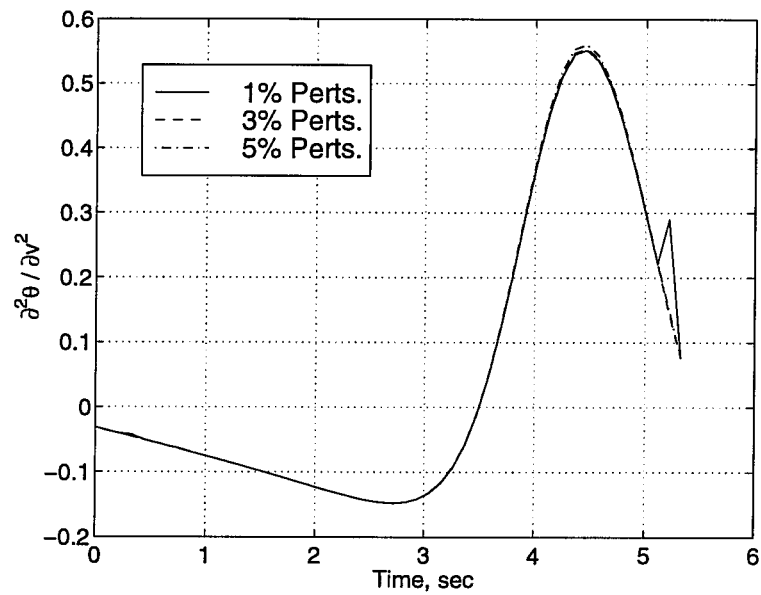


Figure 3.3: Effect of Perturbation Size on $\partial^2\theta/\partial v^2$

of the gains which get large at the end of the trajectory. As the terminal radius is decreased, the gain history comes closer to becoming infinite at the end of the trajectory. Interestingly, though, even for a terminal point condition ($r = 0.0$), the gains do not become infinite. This is because perturbations are done only to the next to last node. Figure 3.5 is typical of the gains which peak before the end of the trajectory, then fall off. The gain values at the end of the trajectory get larger as the terminal radius is decreased, but not infinitely so. For these gains, the values rapidly approach those for $r = 0.0$ as the terminal radius is decreased.

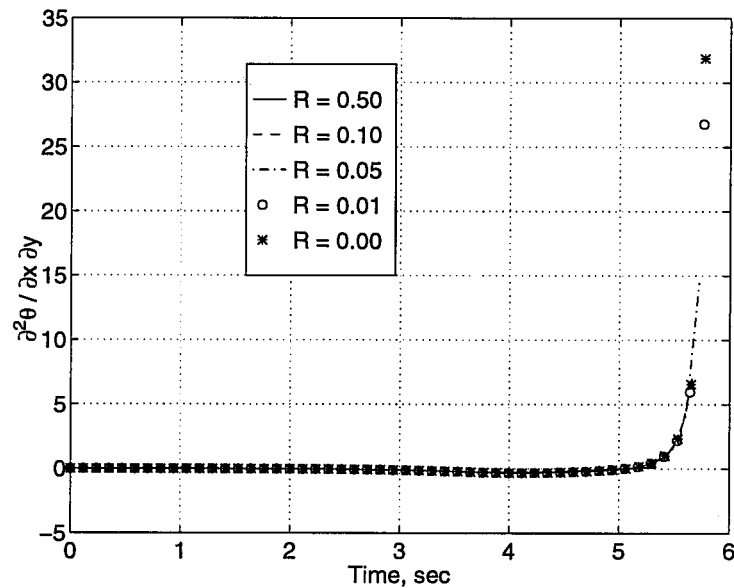


Figure 3.4: Effect of Terminal Radius on $\partial^2\theta/\partial x\partial y$

3.3.2 Simulation Performance

To determine the validity of the second-order NOC control scheme, first and second-order trajectory solutions for various initial condition perturbations are compared to optimal control solutions from those same initial conditions. Because the simulation algorithm uses a different integration scheme than the optimization algorithm, the following problem formulation was solved using the optimization method of Section 2.1.2

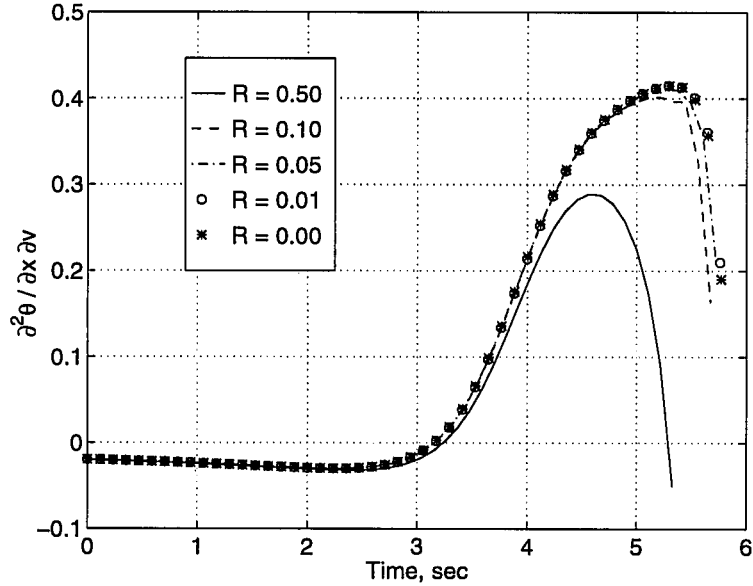


Figure 3.5: Effect of Terminal Radius on $\partial^2\theta/\partial x\partial v$

for determining the performance of the feedback control law:

$$\begin{aligned}
 \mathcal{P}_c : \quad & \text{minimize } \Psi[x(1), p] \text{ subject to} \\
 & 0 = \dot{x} - f(x, u, p), \quad x(0) \text{ given,} \\
 & 0 = u_k^* + (G_1 \delta v)_k + \frac{1}{2} \delta v^T (G_k)_2 \delta v \quad k = 1, \dots, m
 \end{aligned} \quad (3.4)$$

where

$$\begin{aligned}
 \delta v &= v - v^* \\
 v &= [x^T \quad z_s^T]^T
 \end{aligned}$$

This means that the controls are no longer considered as independent variables by the optimizer, but are instead related to the states by the NOC feedback scheme. In addition, the terminal boundary condition replaces the original cost as a function to be minimized, rather than being an equality constraint. Thus, any differences between solutions obtained for (3.4) and those obtained for the original optimal control problem are solely due to the sub-optimality of the feedback control law, rather than differences in integration schemes.

For the Zermelo problem, there are two indicators of optimality: the trajectory duration and the final plant position. To examine these, solutions were first found

for initial condition perturbations around a unit circle centered at the nominal initial state. Solutions were then studied for increasing perturbation size in the direction of worst performance found from the previous solution set.

Results from unit circle initial condition perturbations are presented in Figures 3.6 and 3.7. These indicate that the worst performance is obtained from perturbation along the y -axis, with $+y$ perturbations being slightly worse than $-y$ perturbations. It is also shown that as much as a factor of 10 improvement is achieved by going from first-order feedback to second-order feedback. While the magnitude of the first-order error is less than .04%, it should be remembered that this is only a simple example problem. Thus, the relative magnitudes of the errors are much more important than the absolute magnitudes.

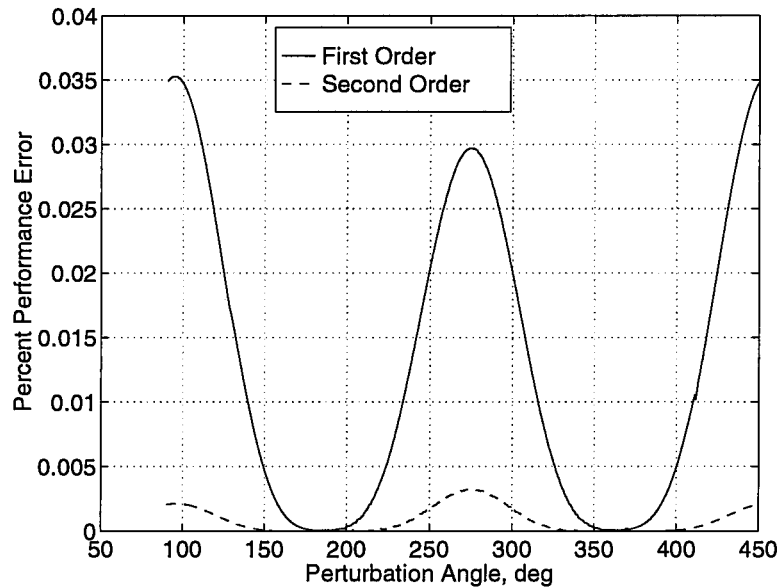


Figure 3.6: Trajectory Duration Error vs. Initial Condition Perturbation Angle

Results of increasing perturbation size along the worst-case perturbation direction, the $+y$ axis, are presented in Figures 3.8 and 3.9. It can be seen that as perturbation size is increased, performance degrades much less severely for second-order feedback than for first-order feedback. This confirms what would be expected from a second-order Taylor expansion, versus a first-order expansion.

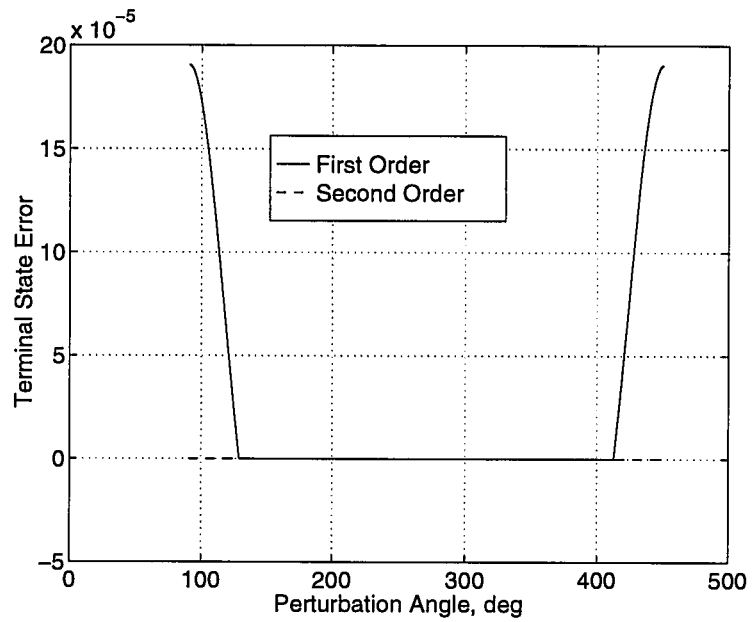


Figure 3.7: Terminal Condition Error vs. I.C. Perturbation Angle

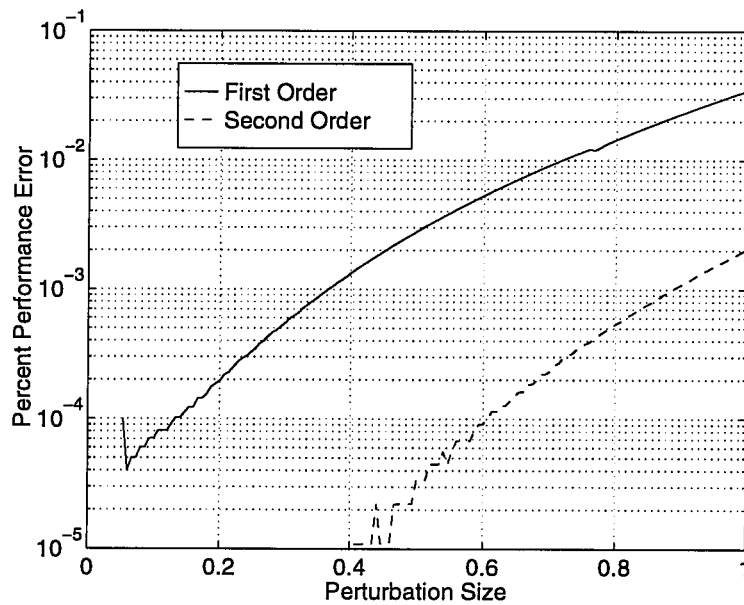


Figure 3.8: Trajectory Duration Error vs. I.C. Perturbation Size

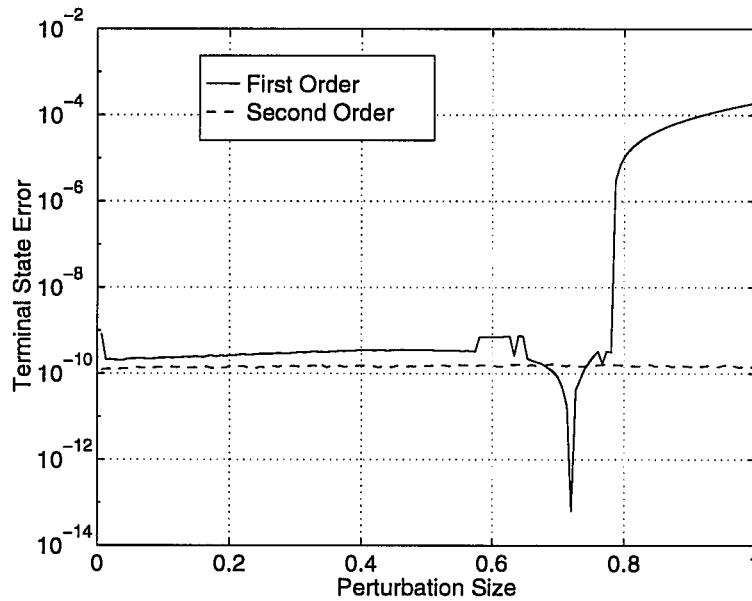


Figure 3.9: Terminal Condition Error vs. I.C. Perturbation Size

In summary, it was shown that even for a 25% perturbation of the initial conditions, the NOC feedback control law generates a trajectory with a performance index within 0.04 optimal performance. It was also shown that second-order feedback produces improved performance over first-order feedback. As expected, with second-order feedback, performance degradation as perturbation size is increased is much less severe than with first-order feedback.

Chapter 4: Single-Stage-To-Orbit Orbit Insertion Problem

In this chapter, the application of the neighboring optimal control technique to a more complex problem – feedback guidance for the launch and orbit insertion of a single-stage-to-orbit launch vehicle with rocket engines is presented. The performance goal for this problem is maximizing the mass lifted into orbit by the launch vehicle. Previously, abort-to-orbit studies were done for this vehicle by assuming the loss of an engine at some time after launch, then re-solving the optimal control problem [8]. Conceptually, using neighboring optimal control, this can be done simply by making the number of engines a static perturbation parameter and running simulations with the loss of an engine. Similarly, the specific impulse of the engines can be made another parameter, allowing changes in engine efficiency to be accommodated.

The state variables for the SSTO problem are longitude θ , latitude ϕ , altitude h , airspeed v , relative heading ψ , relative flight-path γ and mass m . The controls are attack angle α , bank angle μ and throttle η . The equations of motion assume that moment equilibrium is maintained by the control system. These equations also assume that a positive bank angle generates a heading to the north for a vehicle flying west to east, and are given in Appendix B.

The derivation of the equations of motion, as well as descriptions of the physical models used (gravity, atmosphere, propulsion), can be found in [8]. However, improvements have been made in the aerodynamics model since the writing of [8]. These improvements are detailed in Appendix B.

4.1 Problem Formulation

For the SSTO orbit insertion, the optimal control problem will be a maximum mass to orbit ascent trajectory for a vehicle with a full complement of engines. The target orbit is a 50 by 100 nmi orbit with an inclination of 51.6° .

If m_f is the mass of the vehicle at orbit insertion, the baseline trajectory is found

by minimizing the objective function

$$\mathcal{J} = -m_f. \quad (4.1)$$

Note that minimizing $-m_f$ is equivalent to maximizing m_f . The state equations for this problem are described in B.1. The boundary conditions for this problem are

$$\begin{bmatrix} v(0) - v_0 \\ h(0) - h_0 \\ m(0) - m_0 \\ \theta(0) - \theta_0 \\ \phi(0) - \phi_0 \\ \left(\frac{v(1)-v_f}{0.05v_f}\right)^2 + \left(\frac{h(1)-h_f}{0.05h_f}\right)^2 + \left(\frac{\gamma(1)-\gamma_f}{\pi/36}\right)^2 - 1 \\ i(1) - i_f \end{bmatrix} = 0, \quad (4.2)$$

where

$$\left. \begin{aligned} v_0 &= 100 \text{ ft/s,} \\ h_0 &= 15 \text{ ft,} \\ m_0 &= 2383.43 \text{ klbm,} \\ \theta_0 &= -80.71 \text{ deg,} \\ \phi_0 &= 28.47 \text{ deg,} \\ v_f &= 25852 \text{ ft/s,} \\ \gamma_f &= 0 \text{ deg,} \\ h_f &= 3.03805 \text{ ft,} \\ i_f &= 51.6 \text{ deg.} \end{aligned} \right\} \quad (4.3)$$

As shown in [9], the final inclination i_f can be found with

$$i_f = \cos^{-1} \left(\frac{v_f \cos \gamma_f \cos \phi_f \cos \psi_f + \omega(h_f + r_e) \cos^2 \phi}{\sqrt{(v_f \cos \gamma_f \cos \psi_f + (h_f + r_e) \omega \cos \phi)^2 + (v_f \cos \gamma_f \sin \psi_f)^2}} \right). \quad (4.4)$$

The initial flight-path and heading angles are designated as free variables, thus neglecting the brief and highly constrained trajectory arc required to clear the launch pad.

The state/control inequality constraints for this problem are

$$\left. \begin{aligned} 0.25 &\leq \eta \leq 1.09, \\ |\alpha| &\leq 45^\circ, \\ |F_N| &\leq 3.81232e + 05 \text{ lbf}, \\ |A| &\leq 3g_0, \end{aligned} \right\} \quad (4.5)$$

where A is the acceleration of the vehicle, g_0 denotes the gravitational acceleration at sea level, and F_N is the normal force, given by

$$F_N = 1/2\rho v^2 S c_N. \quad (4.6)$$

The normal force coefficient c_N is

$$c_N = c_D \sin \alpha + c_L \cos \alpha. \quad (4.7)$$

Since this is a free time problem, the duration of the trajectory τ is a free parameter. In the numerical calculations, the state equations B.1 are scaled by this parameter. The free parameter inequality constraint shown below was applied to guarantee that time will monotonically increase, but was obviously not active at the solution point.

$$\tau \geq 0. \quad (4.8)$$

The NOC problem was defined for perturbations in all of the states, and in two static parameters representing the efficiency of the propulsion system. These latter two parameters were η_{engines} and $\beta_{I_{sp}}$, which enter into the dynamics through the following relationships

$$\left. \begin{aligned} N_{\text{engines}} &= \eta_{\text{engines}}(N_{\text{engines}})_{\text{nominal}}, \\ I_{sp} &= \beta_{I_{sp}}(I_{sp})_{\text{nominal}}. \end{aligned} \right\} \quad (4.9)$$

The first parameter, η_{engines} , represents a falloff of both thrust and fuel consumption, as in the loss of one of the seven engines. The second parameter, $\beta_{I_{sp}}$, represents a decrease in engine efficiency.

To calculate the gain histories, perturbation sizes need to be chosen. Currently, perturbations are calculated as a percentage of the nominal value at the instant the

Table 4.1: Perturbation Percentage Values

Variable	Perturbation Percentage
v	1.00
γ	1.00
h	1.00
m	1.00
ψ	1.00
θ	0.25
ϕ	0.25
η_{engines}	1.00
$\beta_{I_{sp}}$	1.00

perturbation is being applied. If the nominal value is less than 1×10^{-8} , a fixed perturbation size of 1×10^{-8} is used. Table 4.1 presents perturbation percentages used for this problem. The numerical optimal control solutions, and hence the gain histories, were found using a forty interval time discretization.

4.2 Numerical Results

Results from first-order simulations will be examined first, followed by results from second-order simulations.

First-order feedback simulations using MNI time indexing were examined for the loss of one engine, as well as a reduction of engine I_{sp} by 10%. Results of these simulations were compared to optimal trajectories found under the same conditions. Trajectory comparison plots for the case of one engine out and the remaining engines at full efficiency are presented in Figure 4.1 . Only longitudinal variables are plotted, since the orbit insertion trajectory involves very little lateral motion. Results for all engines operating, but with reduced efficiency, were similar to the results shown.

To analyze the validity of the simulation, final time and final mass for various

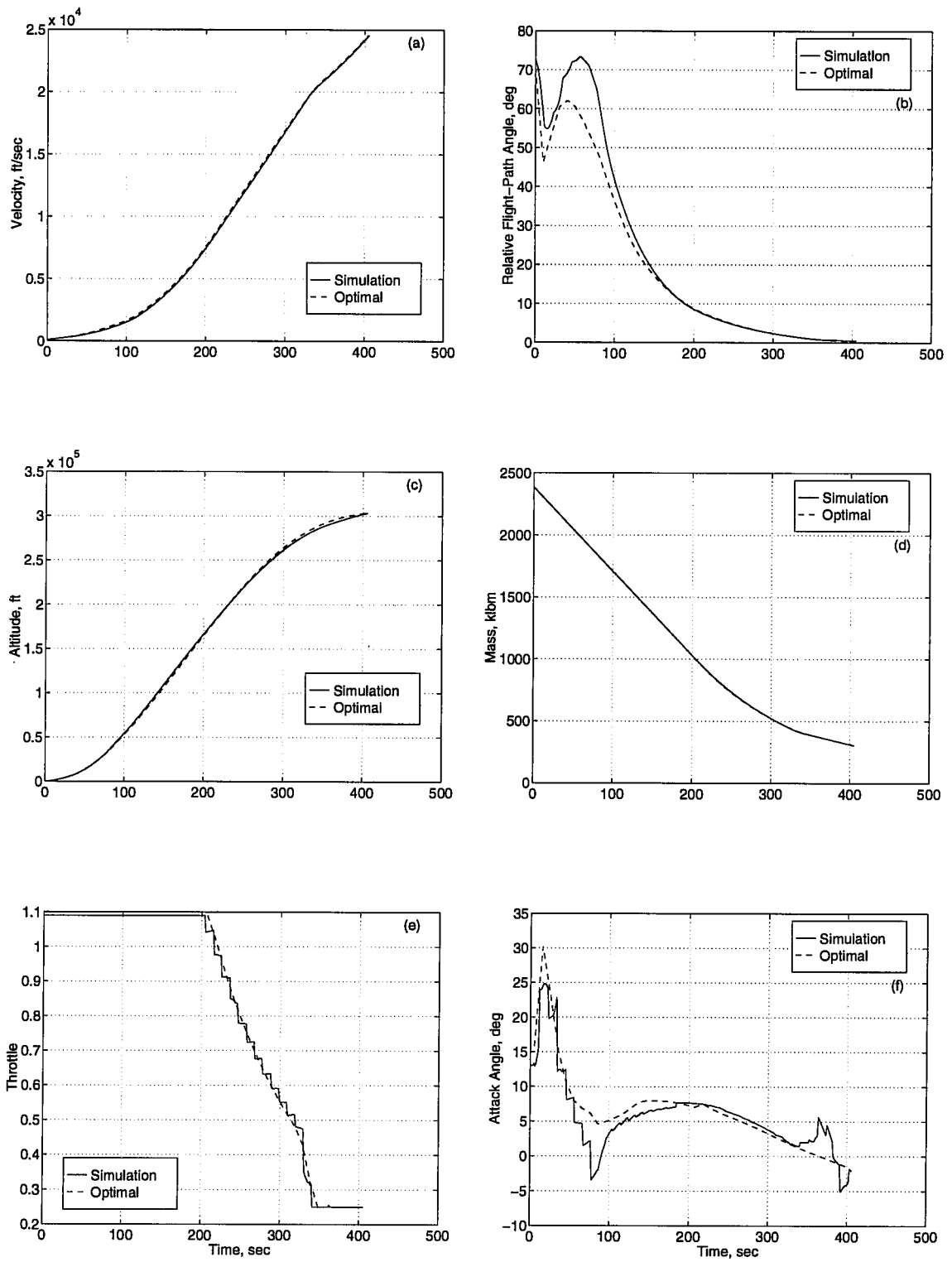


Figure 4.1: Engine Out Comparisons: (a) v , (b) γ , (c) h , (d) m , (e) η , (f) α

Table 4.2: Control Law Performance Reduction from Engine Loss

Engines Out	Final Time Reduction Error	Final Mass Loss Error
0	0%	0%
1	3.3%	0.51%

parameter perturbations are compared to optimal final time and final mass for these same perturbations. These comparisons give a measure of the optimality of the feedback control law as perturbation size is increased. The control law performance reduction when one engine goes out is presented in Table 4.2, while the control law performance reduction for loss of engine efficiency is presented in Figures 4.2 and 4.3. The expected performance dropoff for first-order feedback as perturbation size is increased is seen from these results.

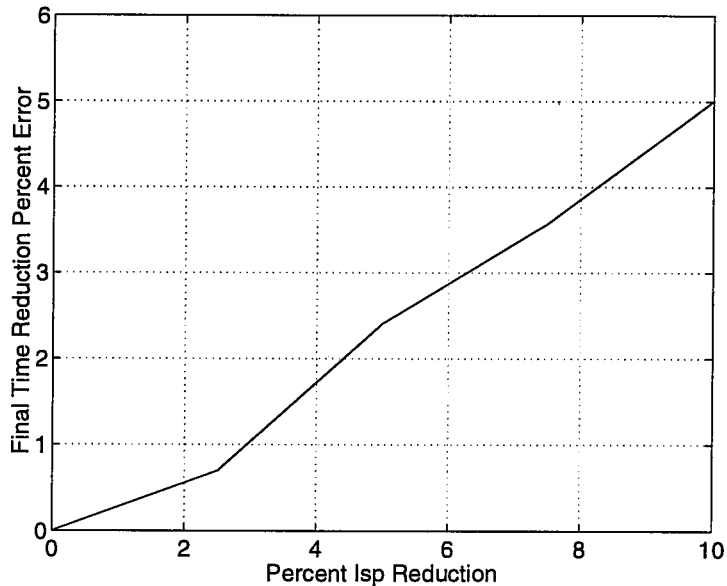


Figure 4.2: Final Time Reduction Error from Engine Efficiency Loss

For second-order feedback simulations, numerical difficulties in the solution approach for calculating the gains introduced apparently spurious variations, or “noise,” in the gain histories. As a worst case example, the gain history for changes in trajectory duration due to simultaneous changes in longitude and latitude is presented in

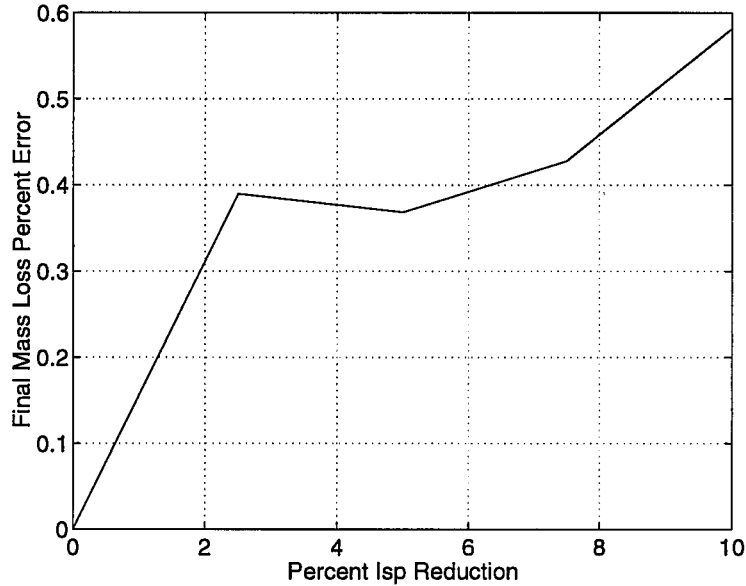


Figure 4.3: Final Mass Loss Error from Engine Efficiency Loss

Figure 4.4.

While first-order simulations were successful in spite of the noise in the gains, second-order simulations failed for no initial condition or parameter perturbations, unless the time correction was reduced to first order, as well as control feedback on at least three states. One possible cause for this problem may lie in the fact that the second derivatives are obtained via numerical second-order finite differences, which makes the noise in the second-order gain histories worse than that in the first-order gains. If these gains have significantly more “noise” than the first-order gains, index time calculations could be corrupted, in addition to the appearance of more noise in the feedback control corrections.

To compare the noise level of the second order gains to that of the first order gains, temporal second differences of the gains are monitored to provide a quantified measure of random, jerky variation in the gains. This measure is defined as:

$$J_2 = \sum_{k=2}^{N_t} \frac{\|\Delta^2 g_k\|_2}{\bar{g}_{k,w}}, \quad (4.10)$$

where g_k is the gain at the k^{th} sampling instant, w is a positive integer, and

$$\Delta^2 g_k = g_{k-1} - 2g_k + g_{k+1}, \quad (4.11)$$

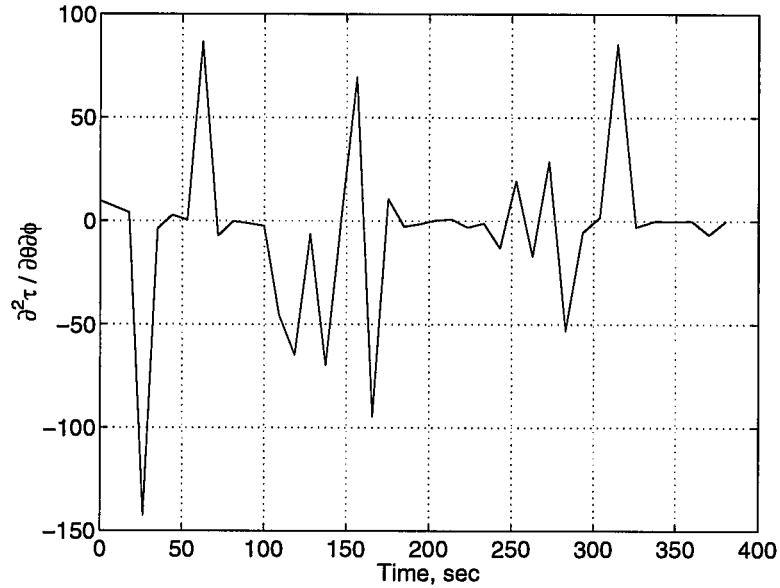


Figure 4.4: Second-Order Final Time Gain

$$\bar{g}_{k,w} = \max \left\{ \frac{1}{2w+1} \sqrt{\sum_{i=i_{\min}}^{i_{\max}} g_i^2}, \quad 0.01 \right\}, \quad (4.12)$$

where

$$\left. \begin{aligned} i_{\min} &= \begin{cases} 1 & \text{if } k \leq w, \\ (N_t + 1) - 2w & \text{if } k > N_t - w, \\ k - w & \text{else,} \end{cases} \\ i_{\max} &= i_{\min} + 2w. \end{aligned} \right\} \quad (4.13)$$

Histograms of J_2 for control and time gains, with $w = 5$, are presented in Figures 4.5 and 4.6. It can be seen from these plots that the second order gains have a larger maximum noise level than the first order gains, as well as having more gains with high noise levels.

The result of the noise in the gain histories can be seen in Figures 4.7 through 4.9. These plots are the control histories of a quasi-second-order feedback simulation with nominal values for initial conditions and parameters, overlaid with the optimal control histories. This is clearly an area in which future work is needed.

While the noise in the second-order feedback gains severely hampered the capability of performing trajectory simulations for significant parameter perturbations, it was

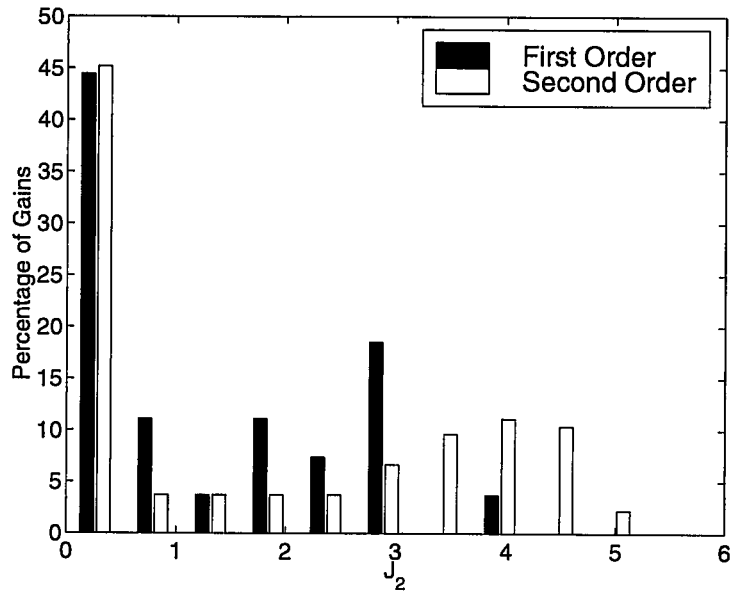


Figure 4.5: Control Gain Noise Level Histogram

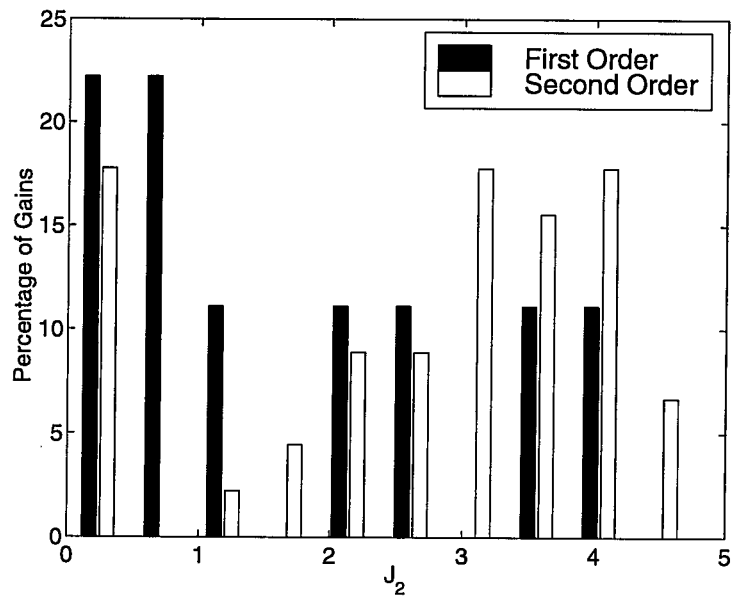


Figure 4.6: Time Gain Noise Level Histogram

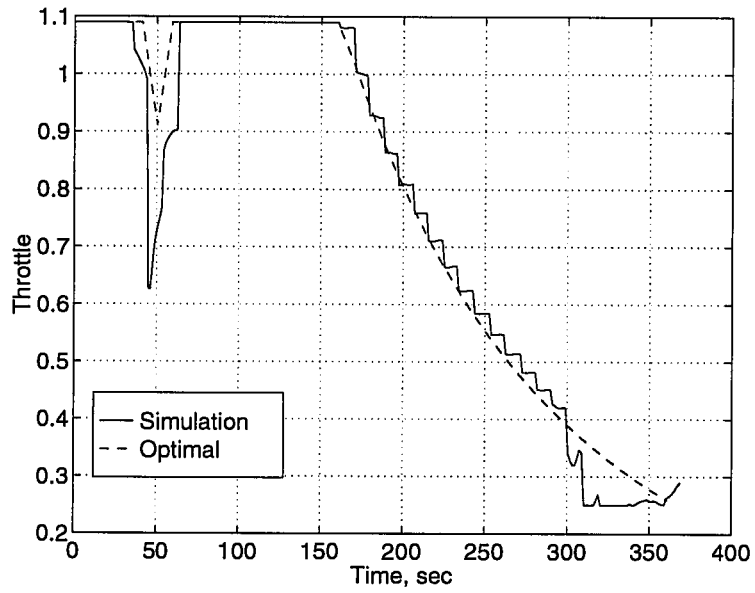


Figure 4.7: Throttle Simulation Control History

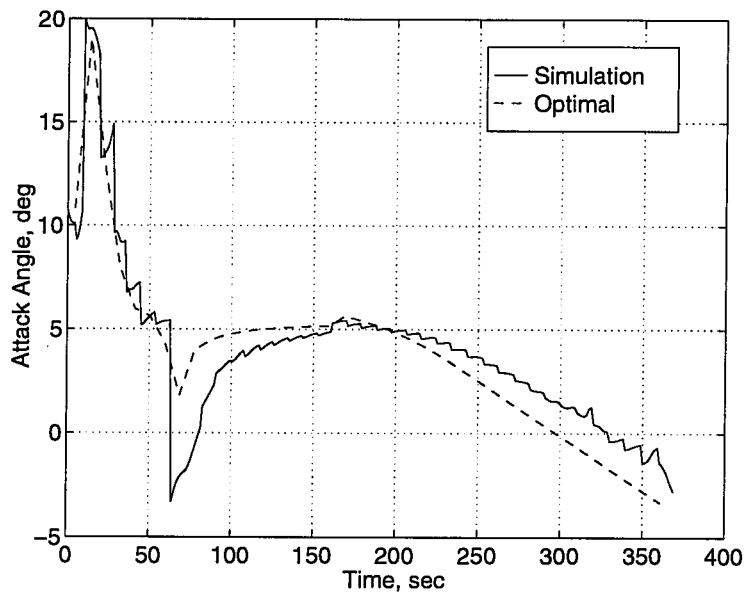


Figure 4.8: Attack Angle Simulation Control History

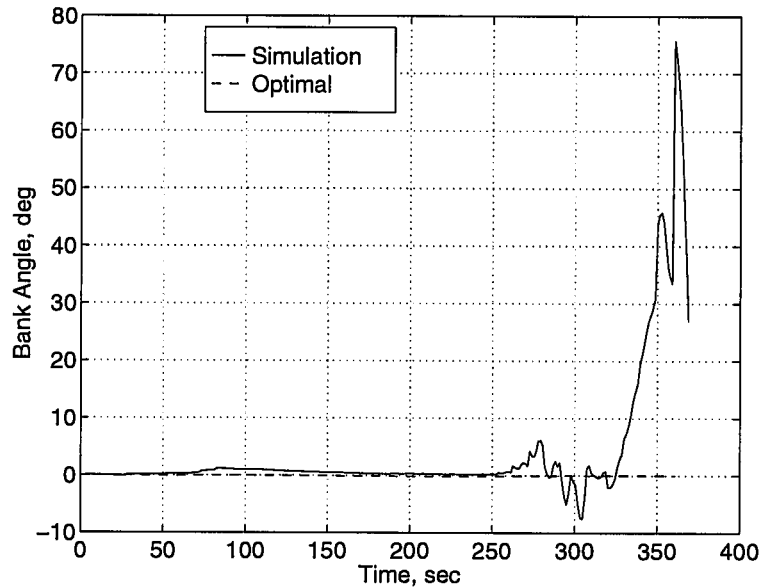


Figure 4.9: Bank Angle Simulation Control History

still possible to get some measure of whether second-order feedback would improve performance over first-order feedback if the noise could be removed from the gains. This potential for performance improvement was analyzed by using the same gain calculation procedure to calculate first and second-order gains for the cost function, final mass, as a function of I_{sp} perturbations at time $t = 0$. Optimal solutions for a series of I_{sp} perturbations were then found, and the final mass results were compared to the values predicted by first and second-order expansions at time $t = 0$.

The difference in percent errors of final mass for first and second-order expansions is presented in Figure 4.10. While this particular case shows less than a 0.25% improvement for a 10% specific impulse reduction for second-order feedback over first-order feedback, it does show that improvement exists, and that it increases as the perturbation size increases. This reinforces the idea that the performance advantage is there, if the numerical difficulties can be overcome to obtain it.

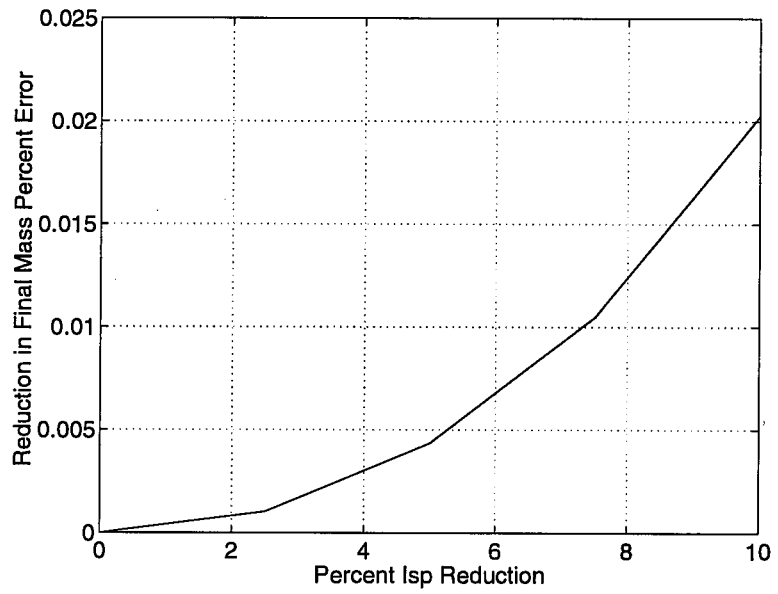


Figure 4.10: Performance Improvement from First to Second-Order

Chapter 5: Conclusions and Recommendations

In this document the examination of a numerical method of performing second-order neighboring optimal control (NOC) was presented. Numerical procedures were implemented for calculating NOC schemes, and a generic simulation with first and second-order NOC guidance was developed. Neighboring optimal control is one method of controlling a plant to follow an optimal trajectory in the presence of modeling errors or external disturbances. The synthesis and simulation was developed for single phase optimal control problems with no state inequality constraints, in which the performance index is purely a function of final state and/or final time. The validity of the control scheme was analyzed by applying it to two example problems: a simple problem, known as the Zermelo problem, and a more complex problem, the orbit insertion trajectory of a generic SSTO launch vehicle.

It was shown from results for the Zermelo test problem that the numerical NOC method examined here produces a valid sub-optimal control law. Even for a 25% perturbation of the initial conditions, the NOC feedback control law generates a trajectory with a performance index within 0.05% of the optimal performance. It was also shown that second-order feedback results produces improved performance over first-order feedback. As expected, performance degradation as perturbation size is increased is much less severe with second-order feedback.

It was shown from results for the SSTO orbit insertion problem that there is a need for improvement in the numerical solution approach employed in calculating the feedback gains. For a complex problem, the convergence sensitivity to initial condition perturbations of the numerical optimal control algorithm introduces excessive noise in the gain histories. The resulting gain histories produced a feedback control law which could be used only if time correction and control feedback on at least three states was performed only to first order. Obviously, opportunities for future work in this area include improvements in the numerical solution method employed in calculating the feedback gain histories. Improvement in this area is needed not only to

improve the solution accuracy, but also to reduce the time required to obtain solutions. There are several possibilities for altering the solution approach to reduce computation and improve solution accuracy, while still utilizing the same iterative solution code. Another approach would be to try to improve the efficiency of the iterative solution algorithms, or to replace them altogether. Also, more work is needed on finding improved methods for estimating the time index into the nominal trajectory.

Another area for further work is to add the ability to handle state inequality constraints. This requires the development of a method to handle state perturbations at points along active state constraint arcs, as well as modifying the control law implementation to be able to determine when the plant is on an active constraint arc.

References

- [1] Calise, A. J., Moerder, D. D., *Two Time Scale Output Feedback Regulation for Ill-Conditioned Systems*, Drexel University, 1986.
- [2] Seywald, Hans, "A Feedback Control Law for the Advanced Launch System," AIAA 91-2691, *AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, August 1991.
- [3] Wetzels, Todd A., *Development of a Finite-Differencing Neighboring Optimal Control Law and Application to the Optimal Landing of a Reusable Launch Vehicle*, Iowa State University, 1991.
- [4] Kirk, Donald E., *Optimal Control Theory: An Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 1970.
- [5] Bryson, A. E., Ho, Y., *Applied Optimal Control - Optimization, Estimation, and Control*, Hemisphere Publishing Corporation, New York, NY, 1975.
- [6] Gill, P. E., Murray, W., Saunders, M. A., Wright, M. H., "User's Guide for NPSOL (Version 4.0) : A Fortran Package for Nonlinear Programming," *Technical Report SOL 86-2*, Systems Optimization Laboratory, Department of Operations Research, Stanford University, January 1986.
- [7] Shampine, L. F., Gordon, M. K., *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.
- [8] Lyon, J. A., "Abort Performance for a Winged-Body Single-Stage to Orbit Vehicle," *NASA Contractor Report 4690*, NASA Langley Research Center, August 1995.
- [9] Hull, D. G., Speyer, J. L., "Optimal Reentry and Plane-Change Trajectories", *Journal of the Astronautical Sciences*, Vol. 30, No. 2, April-June, 1982

Appendix A: Control Law Implementation

In this appendix, the details of the method for implementing the neighboring optimal control law in trajectory simulations is presented. The software for this implementation was developed in the C programming language.

A.1 Simulation Logic

The simulation logic consists of the following steps:

1. Initialize the simulation variables. The following variables are initialized by the user:
 - n - number of states,
 - m - number of controls,
 - n_p - number of static parameters,
 - N_t - number of discretization intervals,
 - x_0 - initial state vector,
 - t_0 - simulation start time,
 - $flagMNI$ - flag indicating which index time calculation method to be used,
 - $flag012$ - flag vector indicating which states and parameters are to be expanded to first or second-order in the control law,
 - $NDIV$ - integer number of timestep subdivisions for determining the integration timestep,
 - $\epsilon_0, \dots, \epsilon_3$ - tolerances for termination logic.

In addition, the feedback gain histories are read in from data files, and the variable *ind_old* is set to -1 to ensure that the gain matrices are updated on the first call to **guidance**.

2. Call the function **guidance**. Given inputs of the current plant state $x = x_k$ and the current time $t = t_k$, the function implements the NOC feedback control law to output the control vector u_k and integration time step dt .
3. Set $x_{old} = x$ and $\Delta t = dt/NDIV$.
4. Step forward Δt by calling the function **ode** to integrate the plant forward in time.
5. Execute the first of four termination test functions to determine if the simulation has reached a stopping point. This first test, **Test0**, evaluates the terminal condition error at the current location. This error is

$$(n_b)_k = \|\Psi(x_k, p)\|, \quad (\text{A.1})$$

where $\|\cdot\|$ denotes the 2-norm. When

$$n_b \leq \epsilon_0 \quad (\text{A.2})$$

is satisfied, **Test0** has been passed. If this is the case, continue to step 6. Otherwise, return to step 2.

6. Test if the Point of Closest Approach (PCA) to the terminal condition manifold bracketed between the current and previous plant states. This point is defined as

$$\left. \begin{aligned} t^e &= \operatorname{argmin}_t \{ \Psi^T(x(t), p) \Psi(x(t), p) \}, \\ x^e &= x(t^e). \end{aligned} \right\} \quad (\text{A.3})$$

The location of this point is approximated by fitting an interpolation polynomial for the terminal condition error between the current and previous plant locations. More on this will be discussed shortly.

If the PCA is bracketed, go to step 10. Otherwise, got to step 7.

7. Call **guidance**.
8. Set $x_{old} = x$.

9. Call `ode` to integrate the plant forward in time by Δt . Return to step 6.
10. Execute the remaining three termination test routines to determine if the PCA has been found. These tests are as follows:

Test1: This test consists of the following steps:

- i. Fit quadratic Hermite polynomial $\hat{x}(t)$ to $\hat{x}_{k-1}, x_{k-1}, x_k$.
- ii. Determine

$$\hat{t} = \arg \min_{t_{k-1} \leq t \leq t_k} \|\Psi(\hat{x}(t), p)\|^2 \quad (\text{A.4})$$

via line search.

- iii. Expand \hat{n}_b^2 about \hat{x} for $x = \hat{x} + \delta\hat{x}$ satisfying $\Psi = 0$:

$$0 \approx \hat{\Psi}^T \hat{\Psi} + 2 \left(\hat{\Psi}^T \hat{\Psi}_x \right) \delta\hat{x} \quad (\text{A.5})$$

so that

$$\|\delta\hat{x}\| \geq \frac{\hat{n}_b^2}{2\|\hat{\Psi}^T \hat{\Psi}_x\|}, \quad (\text{A.6})$$

where the hats now denote evaluation at \hat{x} . As a worst-case error estimate, use the RHS of (A.6) in **Test1**:

$$\delta_1 = \frac{\hat{n}_b^2}{2\|\hat{\Psi}^T \hat{\Psi}_x\|} \leq \epsilon_1. \quad (\text{A.7})$$

If **Test1** is satisfied, perform **Test2**. Otherwise, the PCA has not been found.

Test2: This test consists of steps 1 and 2 from **Test1**, followed by a line search along the direction defined by $\hat{\Psi}^T \hat{\Psi}_x$:

- i. Define $\hat{s} = \hat{\Psi}^T \hat{\Psi}_x / \|\hat{\Psi}^T \hat{\Psi}_x\|$ and solve

$$\tilde{\alpha} = \arg \min_{-a_\alpha \leq \alpha \leq a_\alpha} \|\Psi(\hat{x} + \alpha\hat{s})\|^2 \quad (\text{A.8})$$

where $a_\alpha > 0$ is a user-specified parameter.

- ii. Define $\tilde{x} = \hat{x} + \tilde{\alpha}\hat{s}$ and expand:

$$0 \approx \tilde{\Psi}^T \tilde{\Psi} + 2 \left(\tilde{\Psi}^T \tilde{\Psi}_x \right) \delta\tilde{x} \quad (\text{A.9})$$

so that

$$\|\delta\tilde{x}\| \geq \frac{\tilde{\alpha}_b^2}{2\|\tilde{\Psi}^T\tilde{\Psi}_x\|}, \quad (\text{A.10})$$

where the tildes now denote evaluation at \tilde{x} .

- iii. Estimate the distance from the interpolated trajectory to the target as

$$\delta_2 = \|\delta\tilde{x} + \tilde{\alpha}\hat{s}\|. \quad (\text{A.11})$$

- iv. **Test2** is

$$\delta_2 \leq \epsilon_2. \quad (\text{A.12})$$

If **Test2** is satisfied, perform **Test3**. Otherwise, the PCA has not been found.

Test3: This test, the most stringent of the four, uses an enhanced estimate of the boundary condition error which is obtained by a numerically exact calculation of the minimum norm vector from a polynomial in t connecting x_{k-1} and x_k to a point on the locus $\{x : \Psi(x, p) = 0\}$. The sequence of computational steps are:

- i. Fit cubic Hermite polynomial $\bar{x}(t)$ to $\dot{x}_{k-1}, x_{k-1}, x_k, \dot{x}_k$.
- ii. The necessary conditions for the minimum norm $\delta\bar{x}$ are obtained from a stationary point for the Lagrangian function

$$\mathcal{L} = \frac{1}{2}\|\bar{x}(t) - x\|^2 - \lambda^T\Psi(x, p). \quad (\text{A.13})$$

These necessary conditions are

$$\left. \begin{aligned} 0 &= \mathcal{L}_t = \dot{\bar{x}}^T e, \\ 0 &= \mathcal{L}_x = -e - \Psi_x^T \lambda, \\ 0 &= \mathcal{L}_\lambda = -\Psi(x, p) \\ e &= \bar{x}(t) - x, \end{aligned} \right\} \quad (\text{A.14})$$

and are solved for $\{x^e, t^e, \lambda^e\}$ via a Newton-Raphson search on x, t, λ . The initial guesses for x and t are taken as \hat{t} and \tilde{x} , from **Test1** and

Test2, respectively, and the initial guess for λ is obtained by solving the second of (A.14) for λ :

$$\lambda = -(\Psi_x^T)^+ e \quad (\text{A.15})$$

where $(.)^+$ denotes the pseudoinverse.

iii. **Test3** is

$$\|e_k^e\| \leq \epsilon_3 \quad \text{AND} \quad \|e_k^e\| \leq \|e_{k+1}^e\| \quad (\text{A.16})$$

If **Test3** is satisfied, then the PCA has been found.

If the PCA has been found, go to step 13. Otherwise, go to step 11.

11. Set $\Delta t = \Delta t/2$ and $x = x_{old}$. This sets up re-integrating the previous step, but only half the distance.
12. Call **ode** to integrate the plant forward in time by Δt . Go to step 6.
13. Terminate the simulation at t^e and x^e from **Test3**.

A.2 Guidance Logic

The guidance logic is implemented in the function **guidance**, which performs the following steps:

1. Call the function **mni**. This function calculates the index time t_{in} which minimizes the Euclidean norm of $\delta x = x - x^*(t_{in})$, Where x is the current simulation state vector. The time index is found by bracketing the point of least $\|\delta x\|$ with three points about a trial point ind :

$$\left. \begin{aligned} x_- &= x^*[t^*(ind - 1)], \\ x_0 &= x^*[t^*(ind)], \\ x_+ &= x^*[t^*(ind + 1)], \end{aligned} \right\} \quad (\text{A.17})$$

such that

$$\left. \begin{aligned} \phi(x_0) &= \min\{\phi(x_-), \phi(x_0), \phi(x_+)\}, \\ \phi(y) &= \|x - y\|^2. \end{aligned} \right\} \quad (\text{A.18})$$

Then a quadratic is fitted to those three points and the inflection point is returned as the new time index:

$$t_{\text{MNI}} = \frac{\alpha(t_0 + t_+) + \beta(t_- + t_+) + \gamma(t_- + t_0)}{2(\alpha + \beta + \gamma)}, \quad (\text{A.19})$$

where $t_{\pm} = t^*(ind \pm 1)$, and

$$\left. \begin{aligned} \alpha &= \phi(x_-)/(t_- - t_0)(t_- - t_+), \\ \beta &= \phi(x_0)/(t_0 - t_-)(t_0 - t_+), \\ \gamma &= \phi(x_+)/(t_+ - t_-)(t_+ - t_0). \end{aligned} \right\} \quad (\text{A.20})$$

This function is called even if minimum norm time indexing is not being used, simply because the variable *ind* provides an easy method for determining when to re-interpolate the gain matrices, given in steps 6 and 7 below, as well as a method for calculating the integration time step, given in step 5 below.

2. Check the flag variable, *flagMNI*, to determine which time indexing scheme is to be used. This flag is set by the user.

If *flagMNI* is set, go to step 4. Otherwise, go to step 3.

3. Call the function **tinWCC**. In this function, the index time, t_{in} , is estimated as

$$t_{\text{in}} = t - \Delta t_f(t), \quad (\text{A.21})$$

where t is the current simulation time, and $\Delta t_f(t) = t_f(\Delta v) - t_f^*$ is a correction term due to the current state and the parameters being perturbed from the nominal optimal trajectory, and is calculated from one of

$$\left. \begin{aligned} t_f(\Delta v) &= t_f^* + \frac{\partial t_f}{\partial v} \Delta v, \\ t_f(\Delta v) &= t_f^* + \frac{\partial t_f}{\partial v} \Delta v + \frac{1}{2} \Delta v^T \frac{\partial^2 t_f}{\partial v \partial v^T} \Delta v, \end{aligned} \right\} \quad (\text{A.22})$$

where

$$v = \begin{bmatrix} x \\ z_s \end{bmatrix}, \quad v^*(t) = \begin{bmatrix} x^*(t) \\ z_{s, \text{nominal}} \end{bmatrix}, \quad \Delta v = v - v^*(t), \quad (\text{A.23})$$

depending on whether the user is implementing a first or second-order correction.

The t_{in} calculation can be refined at each k^{th} guidance update by repeating the calculations, forming the sequence

$$\left. \begin{aligned} [(t_{\text{in}})_0]_k &= t - \Delta t_f(t), \\ [(t_{\text{in}})_j]_k &= [(t_{\text{in}})_{j-1}]_k - \Delta t_f[(t_{\text{in}})_{j-1}] \quad j = 1, \dots, \end{aligned} \right\} \quad (\text{A.24})$$

which can be stopped at any j . Proceed to step 5.

4. Call the function **tinMNI**. This function calculates the index time from t_{MNI} using

$$(t_{\text{in}})_{\text{MNI}} = t_{\text{MNI}} - \left. \frac{\partial t_f}{\partial v} \right|_{t_{\text{MNI}}} [v - v^*(t_{\text{MNI}})], \quad (\text{A.25})$$

or its second-order counterpart.

5. Set the integration timestep, dt . This is the nominal timestep, $dt_{\text{nom}} = t^*(\text{ind}) - t^*(\text{ind} - 1)$, corrected for changes in the trajectory duration. First the final time ratio

$$\eta = \frac{t_f}{t_f^*} \quad (\text{A.26})$$

is computed. It is then limited to $0.1 \leq \eta \leq 3$ to ensure a reasonable step size.

The time step size is then computed from

$$dt = \eta dt_{\text{nom}}. \quad (\text{A.27})$$

6. Compare the value of ind to the value from the previous call, stored in ind_old . If they are not the same, go to step 7. Otherwise, go to step 8.
7. If the values ind and ind_old are not the same, this means that the simulation has passed from one point in the stored nominal trajectory to the next. Therefore, call the function **update_gains** to re-interpolate the feedback gain matrix values at time t_{in} from the stored gain histories, and set $\text{ind_old} = \text{ind}$.

In this version of the algorithm, the table lookups are performed using linear-quartic chamfer software which approximates linear interpolation, while preserving smoothness at the data points, since the plant integration and trajectory

termination software assume smoothness. In a real-time application, this could be changed to straight linear interpolation, for faster execution time.

8. Call the function `calc_u` to calculate the control vector for the next timestep. The user can specify whether the control will use either the first or the second-order feedback law, represented by:

$$u = u^*(t_{in}) + F_1(t_{in})V_1(t_{in}), \quad (\text{A.28})$$

$$u_i = u_i^*(t_{in}) + (F_1(t_{in})V_1(t_{in}))_i + \frac{1}{2}V_2^T(t_{in})(F_i)_2(t_{in})V_2(t_{in}) \quad i = 1, \dots, m, \quad (\text{A.29})$$

respectively. To form V_1 and V_2 , the x and z_s are concatenated as:

$$v = \begin{bmatrix} x \\ z_s \end{bmatrix}, \quad v^*(t_{in}) = \begin{bmatrix} x^*(t_{in}) \\ z_{s,nominal} \end{bmatrix}, \quad \Delta v = v - v^*(t_{in}), \quad (\text{A.30})$$

$$V_1 = \begin{bmatrix} \Delta v(\text{vind1}(1)) \\ \Delta v(\text{vind1}(2)) \\ \vdots \\ \Delta v(\text{vind1}(\text{nv1})) \end{bmatrix}, \quad V_2 = \begin{bmatrix} \Delta v(\text{vind2}(1)) \\ \Delta v(\text{vind2}(2)) \\ \vdots \\ \Delta v(\text{vind2}(\text{nv2})) \end{bmatrix}, \quad (\text{A.31})$$

where the `vind1` and `vind2` are indices of elements of v which are to be included in the feedback expansion. This means that the user is not required to feed back the entire v vector, nor required to expand all feedback quantities to second-order. However, all elements expanded to second-order must include a corresponding first-order feedback term.

For returning control values to the plant, the following assumptions are currently made:

- Guidance calculations are instantaneous, so that there is no need to compensate for computation lag.
- Control values are held constant between calls to **guidance**.

Because of these assumptions, and from experimentation, a zero-order hold is currently used for the guidance inputs:

$$u(\tau) = u(t_k) \quad t_k \leq \tau \leq t_{k+1}. \quad (\text{A.32})$$

Appendix B: SSTO Model Improvements

In this appendix, the equations of motion for the SSTO launch vehicle are given, and the improvements made to the aerodynamic model are explained.

The equations of motion for the SSTO, from [8], are

$$\begin{aligned}
 \dot{\theta} &= v \cos \gamma \cos \psi / (r \cos \phi), \\
 \dot{\phi} &= v \cos \gamma \sin \psi / r, \\
 \dot{h} &= v \sin \gamma, \\
 \dot{v} &= (T \cos \alpha - D) / m - g \sin \gamma + \omega^2 r \cos \phi (\sin \gamma \cos \phi - \cos \gamma \sin \phi \sin \psi), \\
 \dot{\psi} &= (T \sin \alpha + L) \sin \mu / (m v \cos \gamma) - \omega^2 r \sin \phi \cos \phi \cos \psi / (v \cos \gamma) \\
 &\quad + 2\omega (\tan \gamma \cos \phi \sin \psi - \sin \phi) / (v \cos \gamma) - (v/r) \cos \gamma \cos \psi \tan \phi, \\
 \dot{\gamma} &= (T \sin \alpha + L) \cos \mu / (m v) + (\omega^2 r / v) \cos \phi (\cos \gamma \cos \phi + \sin \gamma \sin \phi \sin \psi) \\
 &\quad + 2\omega \cos \phi \cos \psi + (v/r - g/v) \cos \gamma, \\
 \dot{m} &= -N_{engines} \eta T_{vac} / (g_0 I_{sp}).
 \end{aligned}
 \tag{B.1}$$

Originally, the lift and drag coefficients were modeled as linear and quadratic functions of angle of attack, respectively. This modeling produces large errors at high angles of attack. Therefore, the aerodynamic modeling was changed to

$$\begin{aligned}
 c_L(M, \alpha) &= a(M) + b(M)\alpha + c(M)\alpha^2 + d(M)\alpha^3 + e(M)\alpha^4 + f(M)\alpha^5, \\
 c_D(M, \alpha) &= a(M) + c(M)\alpha^2 + e(M)\alpha^4 + g(M)\alpha^6.
 \end{aligned}
 \tag{B.2}$$

For simplicity, the odd power coefficients in the c_D equation are set to zero to make the drag symmetric about $\alpha = 0$, in order to use $\alpha = 0$ as the minimum drag point. This also prevents the curve fitting routine from producing negative drag values near an attack angle of zero degrees.

In this model, the coefficients a through g are chosen so as to minimize the Euclidean norm of the error between the model and existing tabular data for the lift and drag coefficients at the table values. The Euclidean norm of the error between the lift model and the lift coefficient data ($c_{L_{TAB}}$) is described by

$$\mathcal{J}_L = \sum_j \frac{1}{2} [c_L(M, \alpha_j) - c_{LTAB}(\alpha_j)]^2 \quad -\frac{\pi}{2} \leq \alpha_j \leq \frac{\pi}{2}. \quad (\text{B.3})$$

This is minimized subject to the constraints that $c_L = 0$ at $\alpha = \pm 90^\circ$, and that the discriminant ($B^2 - 4AC$) of the third derivative of the equation be less than or equal to zero. This last constraint limits the equation to having only one curvature change, which helps the optimization routine to converge. Similarly, the Euclidean norm of the error between the drag model and the drag coefficient data (c_{DTAB}) is described by function

$$\mathcal{J}_D = \sum_j \frac{1}{2} [c_D(M, \alpha_j) - c_{DTAB}(\alpha_j)]^2 \quad -\frac{\pi}{2} \leq \alpha_j \leq \frac{\pi}{2}. \quad (\text{B.4})$$

Figures B.1 through B.4 illustrate the performance of the aerodynamic models in comparison to the tabular data. Surface plots of the aerodynamic models for the lift and drag coefficients are provided in Figures B.1 and B.2. In these figures the tabular data points are included as asterisks to provide an idea of how accurately the models approximate the tabular data. The shapes of these surfaces in the transonic region make modeling the surfaces with a simple analytic expression difficult.

Defining the error in the aerodynamic models as the difference between the modeled coefficients and the tabular coefficients, Figures B.3 and B.4 illustrate the absolute error in the aerodynamic models for the lift and drag coefficients respectively. The tabular data consists of three different subtables corresponding to subsonic, supersonic, and hypersonic Mach ranges. Each of the subtables are defined with different attack angles. Because the tabular data consists of three subtables, three different surfaces are required to describe the absolute error. These figures show that the aerodynamic models adequately approximate the tabular data.

Values for the polynomial coefficients a through g are defined in tables B.1 and B.2. The polynomial fits are used to generate homogeneous data tables of lift and drag versus attack angle and Mach number. This allows the use of in-house multivariable spline interpolation routines to interpolate the lift and drag coefficients over attack angle and Mach number. This should make it easier to update the aerodynamics in

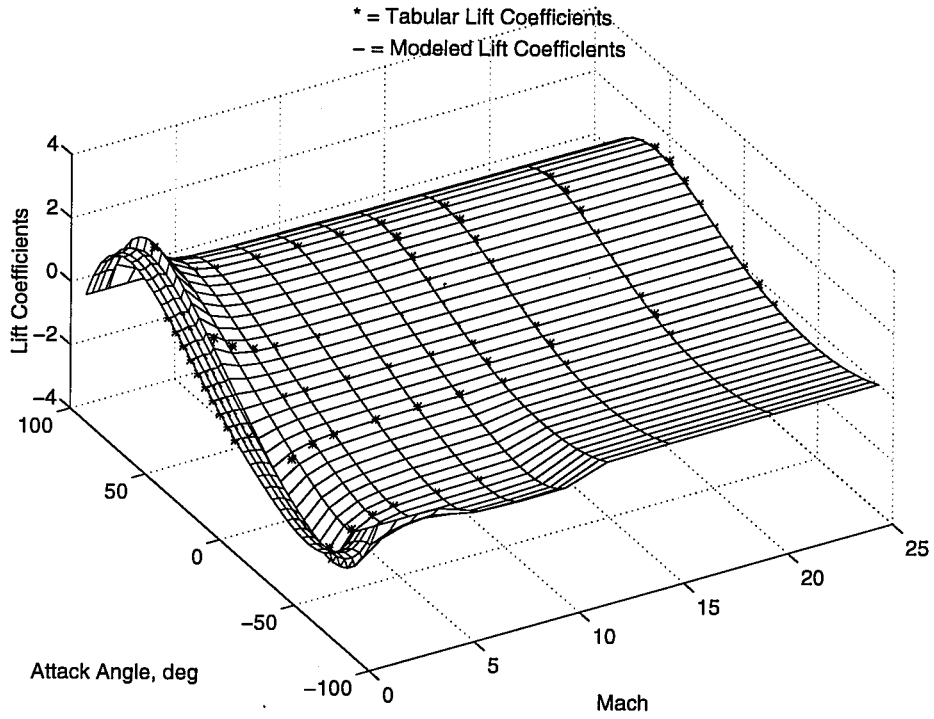


Figure B.1: Comparison of Model and Tabular Lift Coefficients

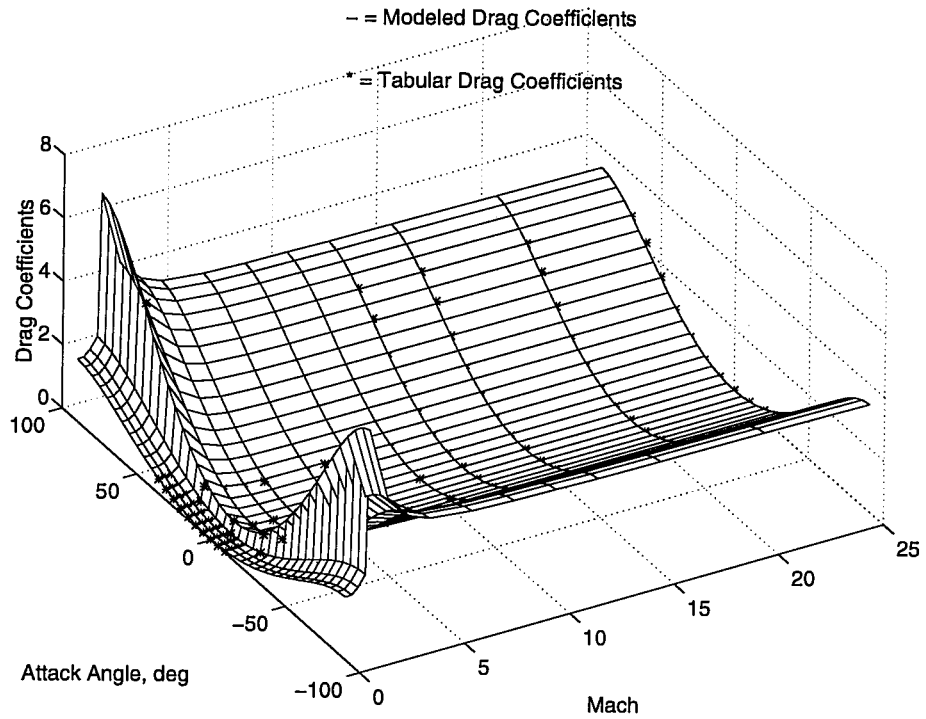


Figure B.2: Comparison of Model and Tabular Drag Coefficients

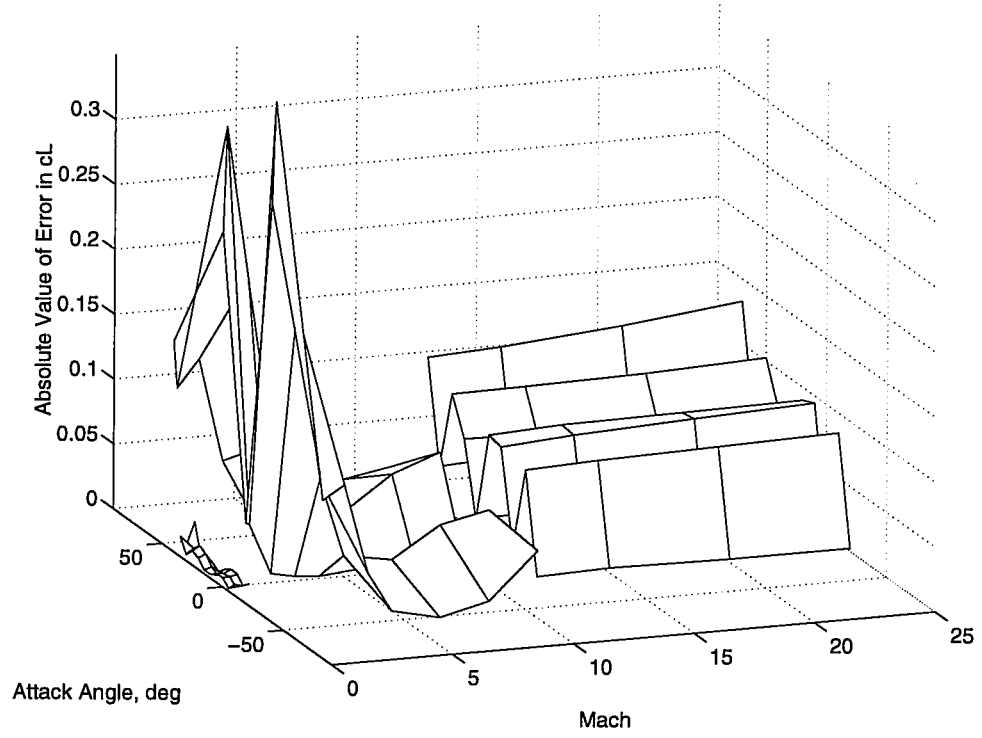


Figure B.3: Comparison of Model and Tabular Lift Coefficients

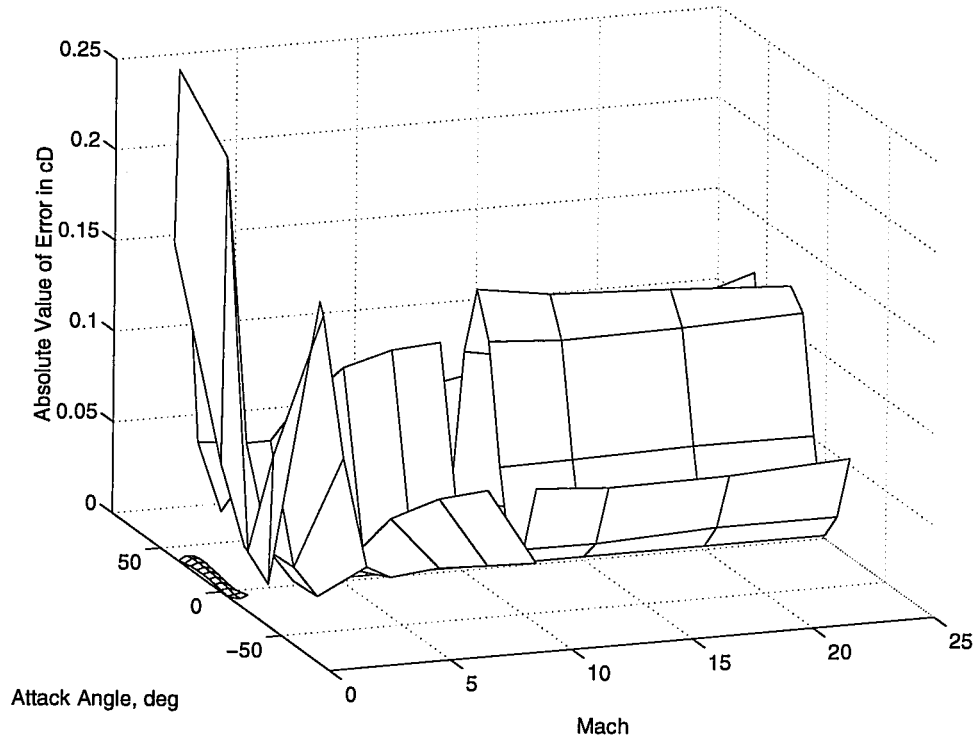


Figure B.4: Comparison of Model and Tabular Drag Coefficients

Table B.1: Coefficients of the Fifth Order Polynomial Lift Curves

<i>Mach</i>	<i>Polynomial Coefficients</i>					
	a	b	c	d	e	f
0.00	-0.0283	2.3791	0.5217	-0.0460	-0.2068	-0.3721
0.30	-0.0283	2.3791	0.5217	-0.0460	-0.2068	-0.3721
0.60	-0.0289	2.5156	0.5547	-0.0492	-0.2200	-0.3932
0.95	-0.0568	2.5580	0.2429	-0.0076	-0.0891	-0.4171
1.10	-0.0753	3.0560	0.3024	-0.0098	-0.1102	-0.4980
2.00	-0.0692	2.0617	0.2342	-0.0083	-0.0835	-0.3353
3.00	-0.0518	1.6673	0.1957	-0.0074	-0.0708	-0.2709
4.00	-0.0453	1.5208	0.1852	-0.0074	-0.0676	-0.2468
6.00	-0.0452	1.4207	0.2646	-0.0407	-0.1396	-0.1915
8.00	-0.0422	1.3849	0.2588	-0.0422	-0.1394	-0.1840
10.00	-0.0406	1.3678	0.2544	-0.0422	-0.1382	-0.1810
12.00	-0.0485	1.2901	0.6479	-0.3276	-0.2546	-0.0792
15.00	-0.0473	1.2693	0.6440	-0.3107	-0.2532	-0.0825
20.00	-0.0461	1.2577	0.6370	-0.3093	-0.2506	-0.0812
25.00	-0.0450	1.2584	0.6309	-0.3197	-0.2483	-0.0771

the future, should this prove necessary.

Table B.2: Coefficients of the Sixth Order Polynomial Drag Curves

<i>Mach</i>	<i>Polynomial Coefficients</i>			
	a	c	e	g
0.00	0.0277	1.1741	0.0000	-0.0643
0.30	0.0277	1.1741	0.0000	-0.0643
0.60	0.0388	1.2888	0.0000	-0.0706
0.95	0.0204	1.7740	-0.3595	0.0000
1.10	0.4071	4.1974	0.0000	-0.2298
2.00	0.0918	2.9723	-0.2242	-0.1022
3.00	0.0811	2.3463	0.0000	-0.1285
4.00	0.0767	2.1914	0.0000	-0.1200
6.00	0.0727	2.1144	0.0000	-0.1158
8.00	0.0710	2.0849	0.0000	-0.1142
10.00	0.0709	2.0732	0.0000	-0.1135
12.00	0.0720	2.0528	0.0000	-0.1124
15.00	0.0704	2.0443	0.0000	-0.1119
20.00	0.0710	2.0275	0.0000	-0.1110
25.00	0.0739	2.0025	0.0000	-0.1096