HIGHER-ORDER METHODS FOR DETERMINING OPTIMAL CONTROLS AND THEIR SENSITIVITIES

A Thesis

by

CHRISTOPHER MATHEW MCCRATE

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

MASTER OF SCIENCE

May 2010

Major Subject: Aerospace Engineering

HIGHER-ORDER METHODS FOR DETERMINING OPTIMAL CONTROLS AND THEIR SENSITIVITIES

A Thesis

by

CHRISTOPHER MATHEW MCCRATE

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

MASTER OF SCIENCE

May 2010

Major Subject: Aerospace Engineering

ABSTRACT

Higher-Order Methods for Determining Optimal Controls and Their Sensitivities.

(May 2010)

Christopher Mathew McCrate, B.S., The University of Missouri

Chair of Advisory Committee: Dr. Srinivas R. Vadali

The solution of optimal control problems through the Hamilton-Jacobi-Bellman (HJB) equation offers guaranteed satisfaction of both the necessary and sufficient conditions for optimality. However, finding an exact solution to the HJB equation is a near impossible task for many optimal control problems. This thesis presents an approximation method for solving finite-horizon optimal control problems involving nonlinear dynamical systems. The method uses finite-order approximations of the partial derivatives of the cost-to-go function, and successive higher-order differentiations of the HJB equation. Natural byproducts of the proposed method provide sensitivities of the controls to changes in the initial states, which can be used to approximate the solution to neighboring optimal control problems. For highly nonlinear problems, the method is modified to calculate control sensitivities about a nominal trajectory. In this framework, the method is shown to provide accurate control sensitivities at much lower orders of approximation. Several numerical examples are presented to illustrate both applications of the approximation method.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Srinivas R. Vadali, for his encouragement and support over the last two years. I have greatly benefited from his dedication to my academic and professional successes. I would also like to thank my committee members, Dr. John L. Junkins and Dr. Aniruddha Datta, for their time and assistance in reviewing the technical aspects of this thesis.

I am grateful to Boeing and the Texas Space Grant Consortium for the fellowships awarded to me during my graduate studies. I am also thankful for the financial support provided by NASA Cooperative Agreement NNX07AC44A.

I would like to acknowledge Drs. James Turner, Raktim Bhattacharya, Martin Berz, Pierluigi Di Lizia, and Walt Williamson for their contributions to this thesis. I truly appreciate all the time and effort they spent sharing their technical expertise with me. Thanks also to Karen Knabe and Lisa Willingham for helping out with all of the office work.

Finally, I am grateful to my family, whose love and support made all of this possible.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

Optimal control theory provides a systematic approach for determining the most efficient way to control a dynamical system. The theory is applicable to a broad range of dynamical systems, spanning all fields of engineering. Over the years, the application of optimal control theory to linear systems has been researched extensively, and is largely understood. However, much work remains in the development of optimal control methods suitable for highly nonlinear systems, such as those found in Aerospace applications.

Several distinct approaches have been developed to solve nonlinear optimal control problems. In many instances, optimal control problems are solved through a direct solution of the necessary conditions via discrete-approximations, e.g., collocation [1,2] or projection methods [3]. Indirect methods typically convert the necessary conditions into a two-point boundary-value problem (TPBVP) [4]. Both the direct and indirect approaches consider only the necessary but not the sufficient conditions for optimality. Conversely, the solution to the Hamilton-Jacobi-Bellman (HJB) equation provides a guarantee of closed-loop stability and the satisfaction of both the necessary and sufficient conditions for optimality [5]. Unfortunately, finding an analytical solution to the HJB equation is impossible for many nonlinear optimal control problems.

Many approaches have been developed to obtain approximate solutions to the

This thesis follows the style of *Journal of Guidance, Control, and Dynamics.*

monomial expansions [8,9], orthogonal functions [10], radial basis functions [11], and neural networks [12]. Methods based on monomial expansions minimize local approximation errors and are best suited for systems with polynomial nonlinearities. Global approximation methods based on finite-differences [13], finite-elements [14], and finite-volume [15] have also been employed to solve the HJB equation. Such methods are applicable to more general nonlinear systems. Recent advances, e.g., level-set methods [13], have been made for evaluating the spatial derivatives via finite-differences and approximating discontinuous cost functions. In general, these global approximation methods provide uniform approximations over a wider domain by allowing variable levels of discretization over different parts of the computational domain. However, practical implementations of these methods have been typically limited to lower-order systems, due to the enormous computational burden of performing discretization over the entire domain.

In many situations, it is beneficial to know the optimal control required to take a system from a variety of initial conditions to some set of fixed terminal conditions. To achieve this, a family of neighboring solutions, often referred to as a field of extremals [4], must be calculated. Both direct and indirect methods generate an open-loop solution to the optimal control problem, i.e., a solution for only one set of initial conditions. It has been widely recognized that if the open-loop optimal control problem can be solved in near-real-time, via either direct or indirect methods, then the solution provides extremal feedback controls. This is the motivation behind the development of model predictive control methods [16]. However, model predictive control methods require a near-real-time solution to a nonlinear constrained minimization problem at each time step, which can be extremely computationally demanding.

Recently, there has been an interest in the computation of a field of extremals via higher-order sensitivity methods [17,18]. Such methods generate sensitivities of the controls to changes in the initial states, which allows for the immediate determination of neighboring optimal controls without the burden of re-solving the optimal control problem. Furthermore, these methods can be extended to account for changes in other conditions, e.g., constant parameters in the system dynamics and terminal constraints.

This thesis develops a general method for approximating the solution to the HJB equation for a class of nonlinear optimal control problems. The approximated solution provides both the nominal open-loop solution and the higher-order control sensitivities required to generate a field of extremals. The HJB equation is approximated locally, instead of over a large domain, which greatly reduces the computational burden associated with the approximation. To begin the developments of this thesis, Sections 1.1, 1.2, and 1.3 introduce the fundamental concepts of optimal control theory relevant to this research.

1. 1 Optimal Control Problem Statement

To motivate the developments of this thesis, consider the following optimal control problem.

Minimize:

$$\mathcal{J} = \phi\left[x\left(t_f\right)\right] + \int_{t_0}^{t_f} L(x, u, t)\, d\tau \tag{1.1}$$

Subject to:

$$\dot{x} = f(x, u, t) \tag{1.2}$$

$$\psi[x(t_f)] = 0 \tag{1.3}$$

where $\mathcal{J}$ is the cost function (performance index), $\phi[x(t_f)]$ is the final state penalty function (soft terminal constraint), $\psi[x(t_f)] \in \Re^{p \leq n}$ is a vector function of hard terminal constraints, and $f(x, u, t)$ is a smooth, analytic, n-dimensional vector function with $x \in \Re^n$ and $u \in \Re^m$. It is assumed that the initial time t_0 and the initial states x_0 are given. The final time may or may not be specified, depending on the particular problem.

1.2 Two Point Boundary Value Problem

The optimal control problem established in Section 1.1 is transformed into a TPBVP using the principles of variational calculus [19]. To begin, the state dynamics and terminal constraints are augmented to the cost function through costates λ and terminal constraint Lagrange multipliers ν, respectively. The augmented cost function is written as:

$$\begin{aligned} \mathcal{J}_a = {} & \phi[x(t_f)] + \nu^T \psi[x(t_f)] \\ & + \int_{t_0}^{t_f} \{L(x, u, t) + \lambda^{\mathrm{T}}(f(x, u, t) - \dot{x})\}\, d\tau \end{aligned} \tag{1.4}$$

where $\nu \in \Re^p$ is a vector of terminal constraint Lagrange multipliers and $\lambda \in \Re^n$ is a vector of costates. Next, the Hamiltonian of the system is defined as:

$$H(x, u, \lambda, t) = L(x, u, t) + \lambda^T f(x, u, t) \tag{1.5}$$

Using this definition, the first variation of the augmented cost function [19] is written as:

$$\delta\mathcal{J}_a = \left[\frac{\partial\phi\left[x(t_f)\right]}{\partial x(t_f)} + \frac{\partial\psi\left[x(t_f)\right]}{\partial x(t_f)}^T \nu - \lambda\right]\delta x(t_f) + \psi\left[x(t_f)\right]^T \delta\nu \qquad (1.6)$$
$$+\int_{t_0}^{t_f}\left[\left(\frac{\partial H}{\partial x} + \dot{\lambda}\right)^T \delta x + \left(\frac{\partial H}{\partial \lambda} - \dot{x}\right)^T \delta\lambda + \left(\frac{\partial H}{\partial u}\right)^T \delta u\right] d\tau$$

First-order necessary conditions for optimality are established by requiring that the first variation of the augmented cost function is zero for arbitrary variations in $\delta x(t_f)$, $\delta\nu$, δx, $\delta\lambda$, and δu. The first-order necessary conditions are:

$$\lambda(t_f) = \frac{\partial\phi\left[x(t_f)\right]}{\partial x(t_f)} + \frac{\partial\psi\left[x(t_f)\right]}{\partial x(t_f)}^T \nu \qquad (1.7)$$

$$\psi\left[x(t_f)\right] = 0 \qquad (1.8)$$

$$\dot{x} = \frac{\partial H}{\partial \lambda} = f(x, u, t) \qquad (1.9)$$

$$\dot{\lambda} = -\frac{\partial H}{\partial x} = -L_x - f_x{}^T\lambda \qquad (1.10)$$

$$\frac{\partial H}{\partial u} = L_u + f_u\lambda = 0 \qquad (1.11)$$

Equations (1.7) and (1.8) provide terminal conditions that must be satisfied to achieve optimality. Equations (1.9) and (1.10) provide governing state and costate differential equations, and are often referred to as Euler-Lagrange equations [4]. For systems affine in the control, Equation (1.11) provides a convenient means to express the control as a function of the states and costates. If the control terms cannot be expressed as a function of the states and costates, they must be treated as additional states. In such a case, differential equations are found for the controls by splitting the time derivative of H_u into partial derivatives:

$$\frac{\partial H_u}{\partial t} + \frac{\partial H_u}{\partial x}\dot{x} + \frac{\partial H_u}{\partial \lambda}\dot{\lambda} + \frac{\partial H_u}{\partial u}\dot{u} = 0 \qquad (1.12)$$

This expression is rearranged to give:

$$\dot{u} = -H_{uu}{}^{-1}\left(\frac{\partial H_u}{\partial t} + \frac{\partial H_u}{\partial x}\dot{x} + \frac{\partial H_u}{\partial \lambda}\dot{\lambda}\right) \tag{1.13}$$

Treating the controls as additional states, Equation (1.11) becomes a constraint that must be achieved for optimality.

After the control terms have been dealt with, the challenge is to find the proper initial costates to drive the states and costates to the desired terminal conditions. This is often accomplished through the use of an iterative method, e.g., the shooting approach. In the shooting approach, the initial costate values are guessed and the Euler-Lagrange equations are integrated to the final boundary conditions. If the terminal conditions are not satisfied, the initial costate values are updated using Newton's method. This process is repeated until the initial costates have converged onto their proper values. Common shooting algorithms, e.g., *Matlab's fsolve.m*, utilize first-order finite-differencing derivatives in the application of Newton's method. Recently, there has been in interest in utilizing higher-order derivatives to provide faster convergence to the shooting algorithm [20].

Further insight can be gained by observing the value of the Hamiltonian along the optimal solution of the TPBVP. The time derivative of the Hamiltonian can be separated into partial derivatives as follows [19].

$$\dot{H} = H_x{}^T\dot{x} + H_u{}^T\dot{u} + H_\lambda{}^T\dot{\lambda} + H_t \tag{1.14}$$

Applying the first-order necessary conditions, the expression is reduced to:

$$\dot{H} = H_t \tag{1.15}$$

Therefore, for problems in which the Hamiltonian doesn't have any explicit time dependence, the Hamiltonian is a first integral of the optimal control problem [4]. This condition can be utilized to check the accuracy of the optimal solution.

1.3 Hamilton-Jacobi-Bellman Equation

The field of Dynamic Programming provides another means for solving optimal control problems in the form of the HJB equation. To begin this formulation, the cost-to-go (optimal value) function is defined as:

$$J(x,t) = \phi\left[x\left(t_f\right)\right] + \int_t^{t_f} L(x,u,t)\, d\tau \tag{1.16}$$

It is important to recognize the difference between the cost function $\mathcal{J}$ and the cost-to-go function J. The cost function measures the total cost of the entire optimal trajectory; whereas, the cost-to-go function measures the remaining cost associated with completing the optimal trajectory. Therefore, the cost-to-go function at the initial time is equal to the cost function.

The Hamilton-Jacobi-Bellman equation is a partial differential equation that defines the partial time derivative of the cost-to-go function. Solution of the HJB equation provides a guarantee of closed-loop stability and the satisfaction of both the necessary and sufficient conditions for optimality. The HJB equation is defined as[*]

$$-\frac{\partial J}{\partial t} = H(x, J_x, t) \tag{1.17}$$

[*] A full derivation of the HJB equation is shown in Appendix A

where J_x is the first-order partial derivative of the cost-to-go with respect to the states, and $H(x, J_x, t)$ is the Hamiltonian defined as:

$$H(x, J_x, t) = L(x, J_x, t) + {J_x}^{\mathrm{T}} f(x, J_x, t) \tag{1.18}$$

In this form of the Hamiltonian, all dependency on the control and costate terms has been removed. First, dependency on the controls is removed using the first-order necessary condition for optimality: $H_u = 0$, as discussed in Section 1.2. Next, the dependency on the costates is eliminated by utilizing the following relationship.

$$J_x = \lambda \tag{1.19}$$

This relationship is only valid when J is continuous and differentiable over the entire domain of x. For cases where J is not smooth over the entire domain, *viscosity* solutions must be considered. In this thesis, the smoothness of J will be assumed for all problems.

Equation (1.19) provides the necessary link between the HJB equation and the TPBVP. An exact solution to the HJB equation provides the optimal values of J_x, which are needed to solve the TPBVP. Unfortunately, finding an exact solution to the HJB equation is extremely difficult for most problems. Hence, some approximation scheme is necessary.

2. OPTIMAL CONTROL SOLUTION METHODOLOGY

As previously stated, the goal of this method is to obtain an open-loop solution to the TPBVP through a local approximation of the HJB equation. Section 2.1 provides a description of the finer details of this approximation, for a class of optimal control problems. Sections 2.2 and 2.3 extend the approximation method to handle optimal control problems formulated with hard terminal constraints and free final times. Finally, Sections 2.4 and 2.5 illustrate the application of the HJB equation approximation method to single dimension, linear and nonlinear optimal control problems, respectively.

2.1 Approximating the Hamilton-Jacobi-Bellman Equation

In Section 1.2, the cost-to-go function was shown to have an explicit dependence on the states of the system. Therefore, assuming the cost-to-go function continuous and differentiable, a finite-order of partial derivatives of the cost-to-go can be taken with respect to the states. In general, these partial derivatives will be defined as: J_x for the first order, J_{xx} for the second order, J_{xxx} for the third order, and so on.

From inspection of the HJB equation, it is evident that the values of J_x must be known in order to compute the time derivative of J. Furthermore, because J must be computed across the trajectory, the values of J_x are needed across the trajectory. Therefore, a differential equation governing J_x must be derived. This is accomplished, in the most general sense, by differentiating the HJB equation with respect to each of the states. Doing so creates a vector of partial differential equations (PDEs) that govern each of the first-order partial derivatives of the cost-to-go, which is given as:

$$-\frac{\partial J_x}{\partial t} = H_x(x, J_x, J_{xx}, t) \tag{2.1}$$

Looking at Equation (2.1), it is evident that the values of J_{xx} must be known in order to compute the time derivative(s) of J_x. Therefore, all of the PDEs contained within Equation (2.1) must be individually differentiated with respect to each state, forming a matrix of partial differential equations governing J_{xx}. Again, the new matrix of PDEs will contain the J_{xxx} terms. This cycle of dependency on the next higher-order partial derivatives will continue for infinitely many differentiations. Because it is infeasible to take infinitely many derivatives, the partial derivatives must be truncated at some order. At the order of truncation, all terms for the next higher-order of partial derivatives appearing in the PDEs will be dropped. In this unique structure, there exists a link between all of the partial derivatives of the cost-to-go, i.e., truncating the partial derivatives at the n^{th} order will affect the accuracy of the 1^{st} through $n\text{-}1^{th}$ orders of partial derivatives.

Once the PDEs have been derived to the desired order, they are solved by the *method of lines*. In this approach, the states are held constant at their initial values while the entire set of PDEs is integrated backwards in time from the terminal boundary conditions. Holding the states constant allows the PDEs to be solved as ordinary differential equations (ODEs), where time is the only independent variable allowed to change. The states are held at their initial values, because the goal of the method is to approximate controls and control sensitivities that are valid at the initial time, i.e., for the initial conditions.

Next, the terminal boundary conditions for the partial derivatives of the cost-to-go must be established. This procedure can vary, depending on the type of terminal boundary conditions applied to the system, i.e., whether or not hard terminal constraints are present. The developments of this section assume an absence of hard terminal constraints. Taking this into account, the cost-to-go function at the final time is defined as:

$$J(t_f) = \phi(x_0) \tag{2.2}$$

where $\phi(x_0)$ is the final state penalty function with the states fixed at their initial conditions. The necessary boundary conditions for the partial derivatives of the cost-to-go are found by differentiating Equation (2.2) with respect to the states, as shown below.

$$J_x(t_f) = \phi_x(x_0) \quad , \quad J_{xx}(t_f) = \phi_{xx}(x_0) \quad , \quad \ldots \tag{2.3}$$

After the PDEs have been integrated backwards in time, a large set of partial derivatives of the cost-to-go with respect to the states is obtained. These values are used in two different ways. The first-order partial derivatives represent the system costates, as defined by Equation (1.19). The higher-order partial derivatives of the cost-to-go function represent the sensitivities of the costates with respect to changes in the states. These sensitivities are used to adjust the costates in the event of a variation in the states (δx). This is accomplished by expanding J_x in a Taylor series about δx as:

$$\lambda^{\mathrm{n}} = J_x + J_{xx}\,\delta x + \frac{1}{2!}J_{xxx}\,\delta x^2 + \frac{1}{3!}J_{xxxx}\,\delta x^3 + \cdots \tag{2.4}$$

where λ^{n} represents the adjusted costates.

A significant drawback to this method is that the partial derivatives of the cost-to-go obtained from the back integration are only valid at the initial time. Therefore, the partial derivatives cannot be used as a form of feedback control valid for the duration of the trajectory. Instead, they must only be used to compute and adjust the costates at the initial time. After this initial computation, the TPBVP must be solved open-loop, by integrating the Euler-Lagrange equations to the terminal conditions. This drawback is a consequence of integrating the PDEs as ODEs, via the *method of lines*.

The advantage to this method is twofold. First, the optimal initial costates needed to solve the TPBVP are obtained without iteration; whereas, other solution methods require an unknown number of iterations to converge on the optimal solution. Secondly, the HJB approximation method automatically produces costate sensitivities, which are used to immediately compute guidance solutions about the nominal trajectory.

2.2 Extension to Terminally Constrained Problems

For problems subject to hard terminal constraints, additional steps must be taken to approximate the HJB equation. In such a problem, the cost-to-go function takes the form:

$$J(x,\nu,t) = \phi\left[x\left(t_f\right)\right] + \nu^T\psi\left[x\left(t_f\right)\right] + \int_t^{t_f} L(x,\mathrm{u},t)\, d\tau \tag{2.5}$$

Clearly, this form of the cost-to-go function has a dependence on both the states and the terminal constraint Lagrange Multipliers (ν). Furthermore, the proper values of ν are needed to initialize the integration of the cost-to-go function. Hence, a procedure for the proper selection of ν must be devised. The procedure is centered on satisfying the following condition for optimality [8]:

$$J_\nu^*(t) = \psi\left[x\left(t_f\right)\right] = 0 \tag{2.6}$$

where $J_\nu^*(t)$ is the optimal partial derivative of the cost-to-go with respect to ν. Although this condition is valid throughout the optimal trajectory, it must be enforced at the initial time. Again, this is because J_ν is only accurate at the initial time when integrated using the *method of lines*. The differential equations governing J_ν are derived by differentiating the HJB equation with respect to ν, as follows:

$$-\frac{\partial J_\nu}{\partial t} = H_\nu(x, J_x, J_{x\nu}, t) \tag{2.7}$$

Again, the cycle of dependency on the next higher order partial derivatives will occur in these PDEs. However, most of these partial derivatives will be a mixture of partial derivatives with respect to both x and ν. Because of this, it is convenient to treat each ν as an additional state.

As previously mentioned, values of ν are needed to start the integration. However, the integration must be completed before the proper values of ν can be found. To resolve this causality dilemma, an iterative procedure is implemented, beginning with a guess for the values of ν. The partial derivatives are then integrated via the *method of lines*, resulting in the initial time values of the partial derivatives. Next, $J_\nu(t_0)$ is expanded in a Taylor series about $\delta\nu$ as follows.

$$\delta J_\nu(t_0) = J_{\nu\nu}(t_0)\delta\nu + \frac{1}{2!}J_{\nu\nu\nu}(t_0)\delta\nu^2 + \frac{1}{3!}J_{\nu\nu\nu\nu}(t_0)\delta\nu^3 + \cdots \tag{2.8}$$

A reversion of series is performed on the above equation to provide the proper $\delta\nu$ values to satisfy Equation (2.8). Appendix B provides a description of the implementation of this reversion of series. The values of $\delta\nu$ calculated in the series reversion are then added to the previous values of ν. The updated values of ν are used to update the final time boundary conditions and the process is repeated. The number of iterations needed for convergence depends on the nonlinearity of the system, the series reversion order, and the accuracy of the initial guess.

Finally, additional steps must be taken to generate neighboring optimal trajectories for variations in the initial states (δx). In the terminally constrained problem, a variation in the initial states causes a variation in the terminal constraint Lagrange multipliers ($\delta\nu$). The adjusted initial costate (λ^n) is found by expanding the costates in a Taylor series about δx and $\delta\nu$ as follows.

$$\lambda^n = J_x + J_{xx}\,\delta x + J_{x\nu}\,\delta\nu + \frac{1}{2!}J_{xxx}\,\delta x^2 + J_{xx\nu}\,\delta x\delta\nu + \frac{1}{2!}J_{x\nu\nu}\,\delta\nu^2 + \cdots \tag{2.9}$$

Before this calculation is performed, the proper values of $\delta\nu$ must be calculated through a Taylor series about the state variations:

$$\delta\nu = \nu_x\delta x + \frac{1}{2!}\nu_{xx}(t_0)\delta x^2 + \frac{1}{3!}\nu_{xxx}\,\delta x^3 + \cdots \tag{2.10}$$

where ν_x, ν_{xx}, and ν_{xxx} are the first three orders of partial derivatives of the Lagrange multipliers with respect to the states. To find these partial derivatives, the Lagrange Implicit Function Theorem [17] is applied to the following condition.

$$J_\nu(t_0) = 0 \tag{2.11}$$

The partial derivative of the cost-to-go with respect to ν is implicitly defined as a function of both x and ν. Additionally, the terminal constraint Lagrange multiplier is implicitly defined as a function of x. Taking these functional relationships into account, successive total derivatives of $J_\nu(t_0)$ with respect to x are split into partial derivatives. The 1st and 2nd order expressions resulting from this process are shown below.

$$J_{\nu x}(t_0) + J_{\nu\nu}(t_0) \cdot \nu_x = 0 \tag{2.12}$$

$$J_{\nu xx}(t_0) + 2 \cdot J_{\nu\nu x}(t_0) \cdot \nu_x + J_{\nu\nu\nu}(t_0) \cdot \nu_x \cdot \nu_x + J_{\nu\nu}(t_0) \cdot \nu_{xx} = 0 \tag{2.13}$$

These equations are inverted to solve for the necessary partial derivatives, ν_x and ν_{xx}. A more detailed description of this process is given in Appendix E.

2.3 Extension to Free Final Time Problems

Some optimal control problems are formulated with a free final time. In such a problem, the final time becomes another variable to be optimized. As a result, these problems are subject to another terminal boundary condition [19].

$$H(t_f) + \frac{\partial\phi}{\partial t_f} + \frac{\partial\psi}{\partial t_f}^T \nu = 0 \tag{2.14}$$

For the class of optimal control problems considered here, it is assumed that the penalty function and terminal constraints have no explicit dependence on time. Therefore, the condition $H(t_f) = 0$ must be satisfied to achieve optimality. Furthermore, Section 1.2 showed that the total derivative of the Hamiltonian is constant throughout the trajectory for problems formulated with no explicit time dependence. Combining these two conditions, the Hamiltonian must be equal to zero throughout the optimal trajectory.

This condition for optimality is applied to the HJB approximation method in the following manner. The total time derivative of the Hamiltonian is separated into partial derivatives as:

$$\frac{dH}{dt} = \frac{\partial H}{\partial t} + \frac{\partial H}{\partial x}^{T} \dot{x} = 0 \tag{2.15}$$

This equation is rearranged to give the partial time derivative of the Hamiltonian as follows.

$$-\frac{\partial H}{\partial t} = \frac{\partial H}{\partial x}^{T} \dot{x} \tag{2.16}$$

This equation is added to the set of PDEs to be integrated via the *method of lines*. As with the other PDEs, the state terms in this equation must be fixed at their initial values during the integration.

To begin the solution process, a final time must be chosen for the integration routine. After the system is integrated, the optimal time is found by observing the time at which the Hamiltonian equals zero. If the Hamiltonian never equals zero, then optimal time is greater than the chosen integration time. However, some optimal control problems, particularly those with a minimum-fuel performance index, don’t have a finite optimal time. In such a problem, the Hamiltonian will approach zero as time approaches infinity.

2.4 Scalar, Linear Example

The following example demonstrates the HJB approximation methodology for a single state, linear optimal control problem with a quadratic performance index. The optimal control problem is solved in three different forms:

1) Fixed Final Time Problem
2) Terminally Constrained, Fixed Final Time Problem
3) Free Final Time Problem

All of the methodology developed in Sections 2.1, 2.2, and 2.3 will be demonstrated with these three forms of the linear quadratic problem.

2.4.1 Fixed Final Time Problem

Minimize:

$$\mathcal{J} = \frac{1}{2} W\big(x(t_f) - x_f\big)^2 + \frac{1}{2}\int_0^{t_f} (x^2 + u^2)\, dt \tag{2.17}$$

Subject to:

$$\dot{x} = x + u \tag{2.18}$$

$$x(0) = 1 \;\;,\;\; t_f = 2 \tag{2.19}$$

where W is a final state weight and x_f is the desired final state value. The values of W and x_f used in this problem are given below.

$$W = 100 \;\;,\;\; x_f = 4 \tag{2.20}$$

To begin the solution process, the problem is transformed into a TPBVP. First, the Hamiltonian is constructed as

$$H = \frac{1}{2}(x^2 + u^2) + \lambda(x + u) \tag{2.21}$$

where λ is the costate. Differentiating the Hamiltonian with respect to u provides a relationship between the costate and control.

$$u = -\lambda \tag{2.22}$$

This relationship is used to remove the control terms from the Hamiltonian. Next, first-order necessary conditions give the Euler-Lagrange equations and the final time boundary condition for the costate as:

$$\dot{x} = \quad x - \lambda \tag{2.23}$$

$$\dot{\lambda} = -x - \lambda \tag{2.24}$$

$$\lambda(t_f) = W\left(x(t_f) - x_f\right) \tag{2.25}$$

Replacing all of the costate terms in the Hamiltonian with J_x, the HJB equation is defined as:

$$-\frac{\partial J}{\partial t} = \frac{1}{2}x^2 - \frac{1}{2}{J_x}^2 + x\,J_x \tag{2.26}$$

Taking successive first and second order partial derivatives of the HJB equation with respect to x gives the following scalar differential equations.

$$-\frac{\partial J_x}{\partial t} = x - J_x\,J_{xx} + J_x + x\,J_{xx} \tag{2.27}$$

$$-\frac{\partial J_{xx}}{\partial t} = 1 - {J_{xx}}^2 + 2J_{xx} \tag{2.28}$$

Because this is a linear quadratic problem, all of the partial derivates higher than 2nd order will naturally be equal to zero for all time. Therefore, no approximation of the HJB equation is necessary and the values of J_x and J_{xx} will be solved for exactly. Additionally, Equation (2.28) is analogous to the *Riccati Equation*, which is given as:

$$-\dot{S} = Q - SBR^{-1}B^{\mathrm{T}}S + A^{\mathrm{T}}S + SA \tag{2.29}$$

where S is equivalent to J_{xx}, and the variables A, B, Q, and R all have a constant value of one. Because this equation is independent of state terms, the values of J_{xx} are valid throughout the integration.

The value of the cost-to-go function at the final time is given by:

$$J(t_f) = \frac{1}{2}W\left(x(t_f) - x_f\right)^2 \tag{2.30}$$

The boundary conditions for J_x and J_{xx} are derived by differentiating Equation (2.30) with respect to x, and fixing x at the initial condition.

$$J_x(t_f) = W\left(x(0) - x_f\right) \tag{2.31}$$

$$J_{xx}(t_f) = W \tag{2.32}$$

Finally, the differential equations governing J, J_x, and J_{xx} are integrated via the method of lines. Integration is performed in *MATLAB*, using a 4th order variable step *Runge-Kutta* routine.

The state and costate (J_x) results of this integration are represented by the dashed lines in Figure 2.1. For comparison purposes, the optimal open-loop solution to the TPBVP is represented by the solid lines. As expected, the value of J_x at the initial time is equal to the optimal initial costate, meaning that the correct initial costate was computed by the approximation method. Additionally, the figure shows that the condition $J_x = \lambda^*$ is only accurate at the points on the trajectory where $x^* = x_0$.

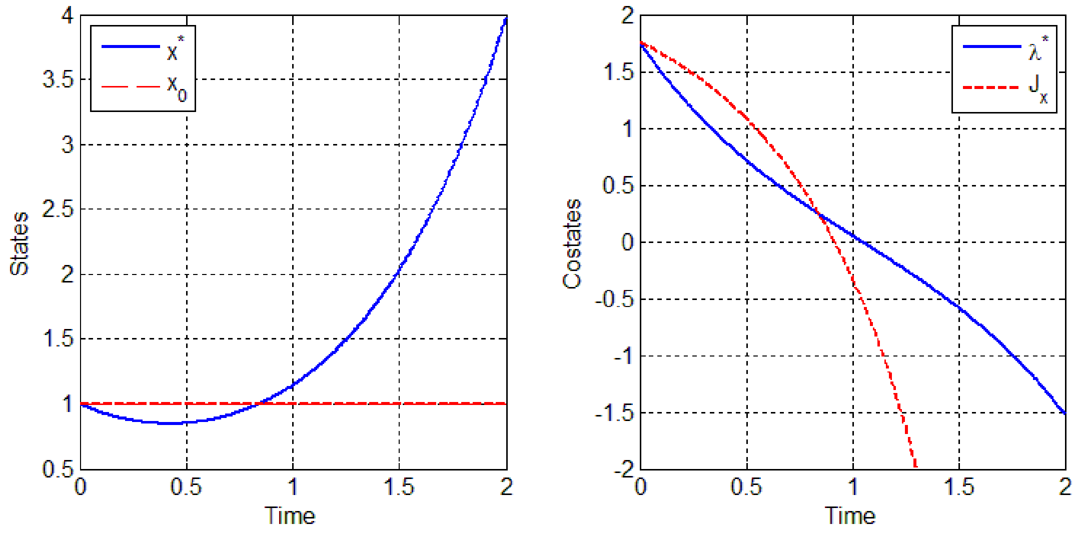

Figure 2.1 Approximation method results and optimal open-loop solutions to the linear problem.

Figure 2.2 shows neighboring optimal trajectories, approximated about the nominal solution, for the linear quadratic system. The neighboring trajectories are created by integrating the Euler-Lagrange equations from neighboring initial states ($x^* + \delta x$) and costates ($\lambda^* + \delta\lambda$). To begin the procedure, the initial states are perturbed over a range of ± 1. Then, the corresponding initial costates are perturbed using:

$$\delta\lambda = J_{xx}\,\delta x \tag{2.33}$$

An accurate value of J_{xx} ensures the neighboring trajectory is optimal.

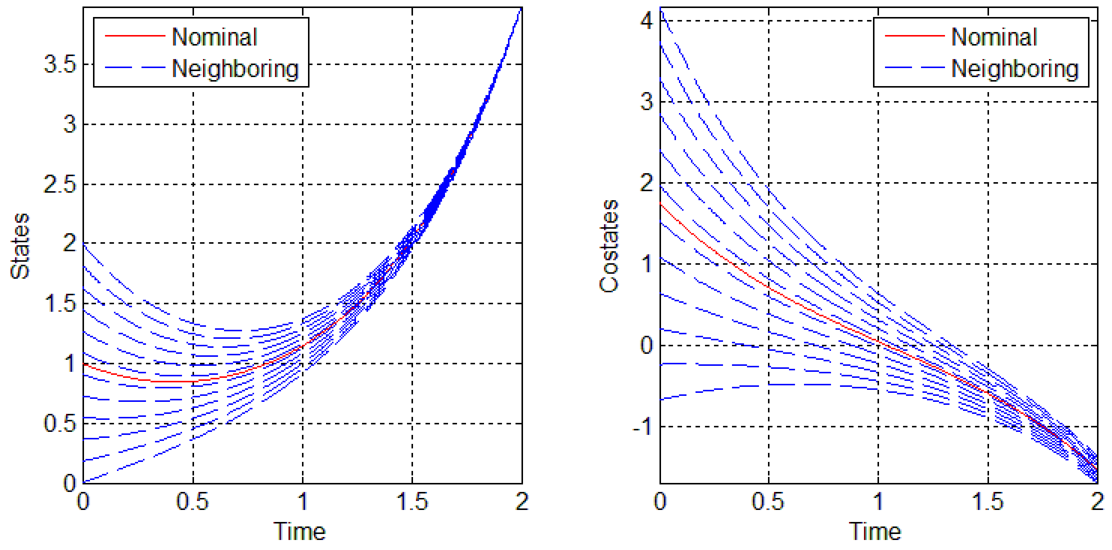

Figure 2.2 Neighboring optimal trajectories for the linear problem, considering initial state variations.

2.4.2 Terminally Constrained, Fixed Final Time Problem

Minimize:

$$\mathcal{J} = \frac{1}{2}\int_0^{t_f} (x^2 + u^2)\, dt \tag{2.34}$$

Subject to

$$\dot{x} = x + u \tag{2.35}$$

$$\psi[t_f] = x(t_f) - x_f \tag{2.36}$$

In this version of the linear problem, a hard terminal constraint is applied instead of a final state penalty function. If the problem is solved accurately, the terminal constraint will be satisfied, ensuring that the final state will exactly reach its desired final value (x_f).

The addition of the terminal constraint has no effect on the HJB equation[†], but does effect the cost-to-go function, which is now dependant on x and ν. Therefore, the HJB equation must now be differentiated with respect to both x and ν. This differentiation produces the following first and second order equations:

$$-\frac{\partial J_x}{\partial t} = x - J_x J_{xx} + J_x + x J_{xx} \tag{2.37}$$

$$-\frac{\partial J_\nu}{\partial t} = -J_x J_{x\nu} + x J_{x\nu} \tag{2.38}$$

$$-\frac{\partial J_{xx}}{\partial t} = 1 - J_{xx}{}^2 + 2J_{xx} \tag{2.39}$$

$$-\frac{\partial J_{x\nu}}{\partial t} = -J_{x\nu} J_{xx} + J_{x\nu} \tag{2.40}$$

$$-\frac{\partial J_{\nu\nu}}{\partial t} = -J_{x\nu} J_{x\nu} \tag{2.41}$$

The terminal boundary conditions for all five first and second order partial derivatives are given as:

$$J_x(t_f) = \nu \quad , \quad J_\nu(t_f) = \left(x(0) - x_f\right) \tag{2.42}$$

$$J_{xx}(t_f) = 0 \quad , \quad J_{x\nu}(t_f) = 1 \quad , \quad J_{\nu\nu}(t_f) = 0 \tag{2.43}$$

Finally, the process to solve the terminally constrained optimal control problem requires iterations. A list of the steps required to solve the problem is given below.

1) Guess a value for ν.
2) Integrate the 1st and 2nd order PDEs via the method of lines.
3) Update the value of ν with the following equation:

$$\nu^{i+1} = \nu^i - J_{\nu\nu}{}^{-1} J_\nu \tag{2.44}$$

4) Repeat Steps 2 and 3 until the value of ν converges.

[†] Consult Appendix A for an explanation of why the HJB equation is unaffected by terminal constraints

Because this problem is linear, the proper value of ν will be obtained on the first iteration.

The results for this solution process can be seen in Figure 2.3. Again, the HJB approximation results are plotted against the optimal solution to the TPBVP. The figure shows the values of J_x obtained on both iterations. In the first iteration, the simulation fails to meet the condition $J_x = \lambda^*$ at any point along the trajectory. This is not surprising, given that the value of ν used to start the integration was inaccurate. In the second iteration, the proper value of ν was used to start the integration. As a result, the proper value of J_x is obtained at the initial time. The condition $J_x = \lambda^*$ is also satisfied at another point along the trajectory. The significance of this intersection point will be discussed in the next section.

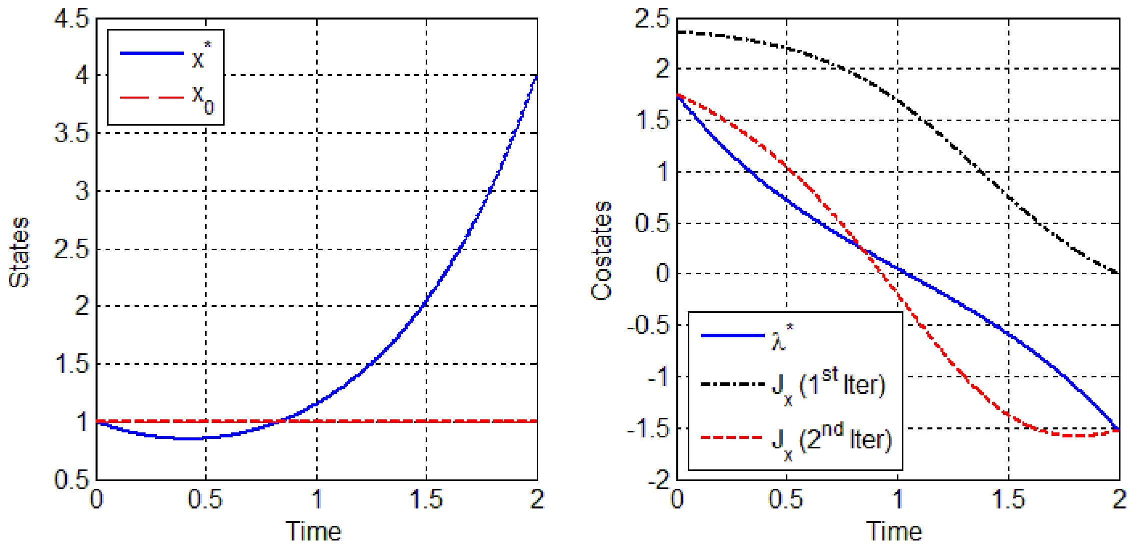

Figure 2.3 Approximation method results and optimal open-loop solutions for the terminally constrained, linear problem.

Figure 2.4 shows a field of extremals, about the nominal solution. Again, these neighboring solutions are generated by varying the initial state over a range of ± 1. Using the second order cost-to-go sensitivities, variations in the terminal constraint Lagrange multipliers and initial costates are computed as a function of the initial state variations, as shown below.

$$\delta\nu = -J_{\nu\nu}{}^{-1} J_{\nu x}\,\delta x \tag{2.45}$$

$$\delta\lambda = J_{xx}\,\delta x + J_{x\nu}\,\delta\nu \tag{2.46}$$

After the initial states and costates are varied, the Euler-Lagrange equations are integrated to the terminal conditions, providing the neighboring optimal trajectories.

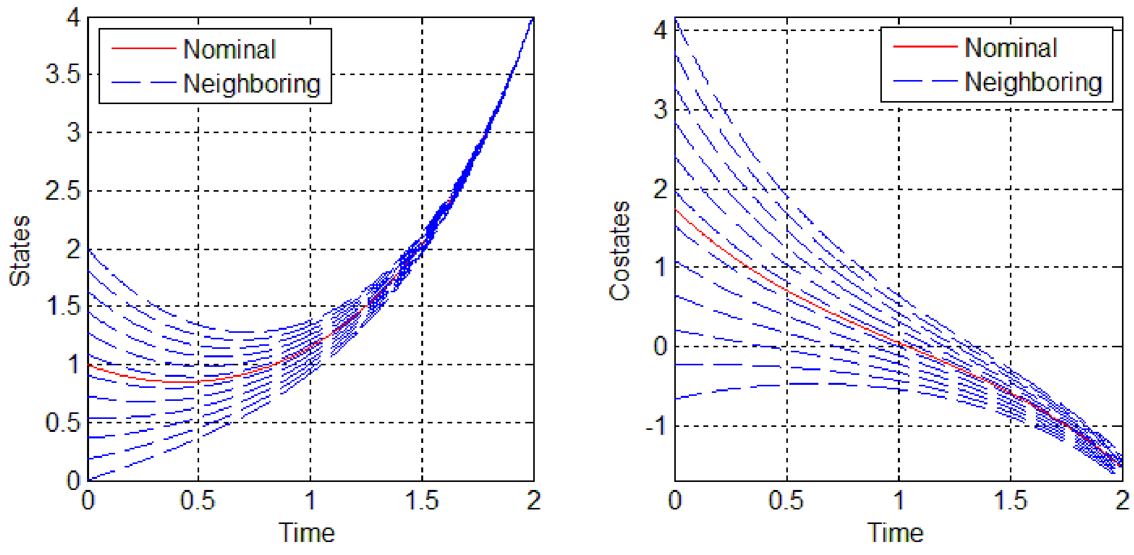

Figure 2.4 Neighboring optimal trajectories for the terminally constrained linear problem, considering initial state variations.

2.4.3 Free Final Time Problem

The free final time problem is analogous to the problem posed in Section 2.4.1, except for the unknown final time. The time history of the Hamiltonian will provide a condition for optimizing the final time. The differential equation governing the Hamiltonian is given as:

$$-\frac{\partial H}{\partial t} = (x - J_x\, J_{xx} + J_x + x\, J_{xx})(x - J_x) \tag{2.47}$$

The terminal boundary condition for the Hamiltonian is given as:

$$H(t_f) = \frac{1}{2}\, {x_0}^2 + \frac{1}{2}\, J_x(t_f)^2 - x_0\, J_x(t_f) \tag{2.48}$$

A final time of 2 seconds is used to integrate the differential equations. The cost-to-go and partial derivative results of the approximation method are analogous to those obtained in the fixed final time version of the problem. The difference between the two solutions will be the time at which the results are used. Instead of using the values at time $t = 0$, the values at some other time along the back integration will provide the optimal solution.

Figure 2.5 shows both the HJB approximation method results and the optimal open-loop solution. In the figure, the approximation method results are represented by the dashed line, and the optimal solution is represented by the solid line. From the figure, the condition for optimizing the final time is met at 1.023 seconds. Not only is the Hamiltonian equal to zero at this time, but the cost-to-go is also a minimum. Thus, the optimal final time is 0.977 seconds ($t_f^* = 2 - 1.023$). The optimal open-loop solution is obtained by integrating the Euler-Lagrange equations from 1.023 seconds forward to 2 seconds, using x_0 and $J_x(1.023)$. As the figure shows, $J_x(1.023)$ was the proper optimal initial costate for the optimal time problem.

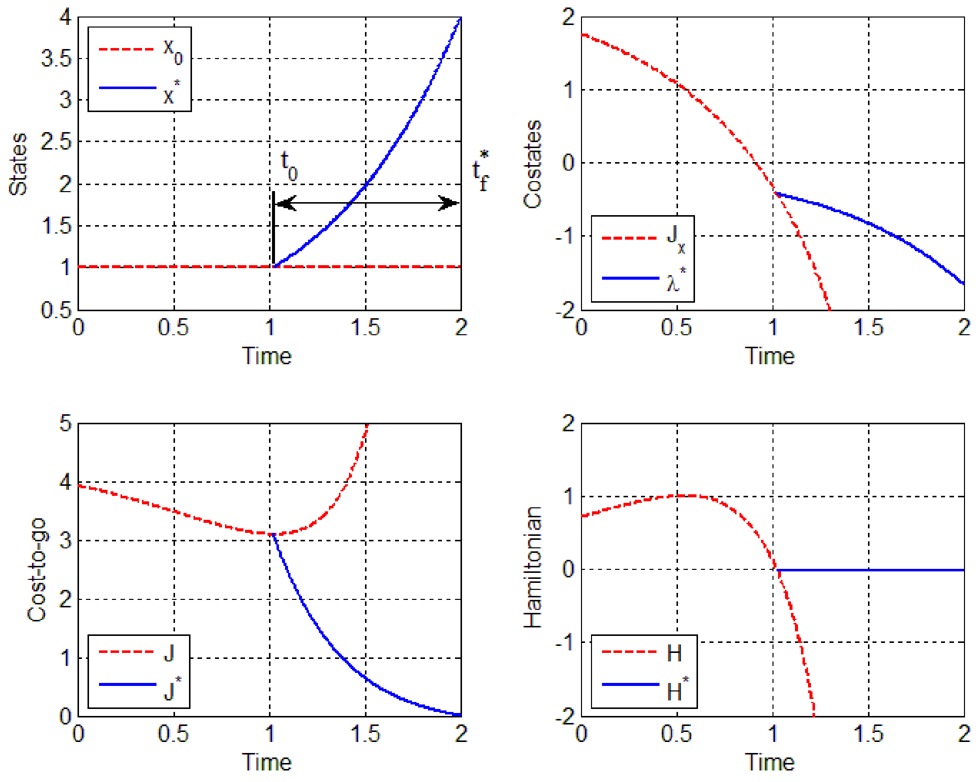

Figure 2.5 Approximation method results and optimal open-loop solutions to the free final time, linear problem.

The optimal time problem demonstrates an important point about the approximation method results, which has been ignored until now. Although the results are only valid at one point for a particular problem, each point is valid for a different optimal control problem. In other words, the approximation method results provide the initial costates needed to solve the open-loop optimal control problem formulated with any final time ranging from t_0 to t_f. In this manner, the method provides a means of generating a field of extremal controls, for variations in the problem time. This point is illustrated in Figure 2.6, where the problem time is varied over a range of ± 0.5 seconds about the optimal. As the figure shows, the optimal initial costate for each extremal path is located on the approximation method results (dashed line).

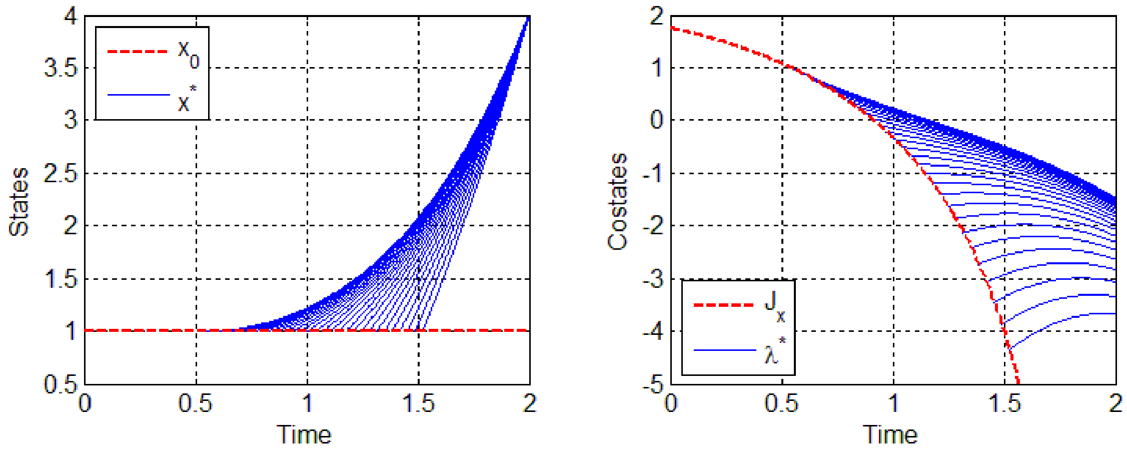

Figure 2.6 Neighboring optimal trajectories for the linear problem, considering variations in the problem time.

2.5 Scalar, Nonlinear Example

Minimize:

$$\mathcal{J} = \frac{1}{2}W\left(x\left(t_f\right) - x_f\right)^2 + \frac{1}{2}\int_0^{t_f} (x^2 + u^2)\, dt \tag{2.49}$$

Subject to:

$$\dot{x} = x + \varepsilon x^3 + u \tag{2.50}$$

$$x(0) = 1 \quad , \quad t_f = 2 \tag{2.51}$$

where ε is a parameter that is adjusted to change the nonlinearity of the system. The penalty function parameters used to simulate this problem are identical to those given for the linear problem.

$$x_f = 4 \quad , \quad W = 100 \tag{2.52}$$

The HJB equation for the nonlinear problem is constructed as follows.

$$-\frac{\partial J}{\partial t} = \frac{1}{2}x^2 - \frac{1}{2}{J_x}^2 + x\,J_x + \varepsilon x^3 J_x \tag{2.53}$$

Because this is a nonlinear problem, an unknown number of partial derivatives of the cost-to-go are needed to accurately approximate the HJB equation. The appropriate order of approximation depends on the nonlinearity of the system, i.e., the chosen value of ε. To illustrate this, Figure 2.7 shows the error in the initial costate (the difference between the optimal and approximated initial costates), calculated using 3rd through 6th order approximations, for increasing values of ε. Two main conclusions can be drawn from the figure. First, the accuracy of the approximation is reduced as the nonlinearity of the system increases, for all orders of approximation. Secondly, each additional order of approximation further improves the accuracy of the solution over the entire range of ε.

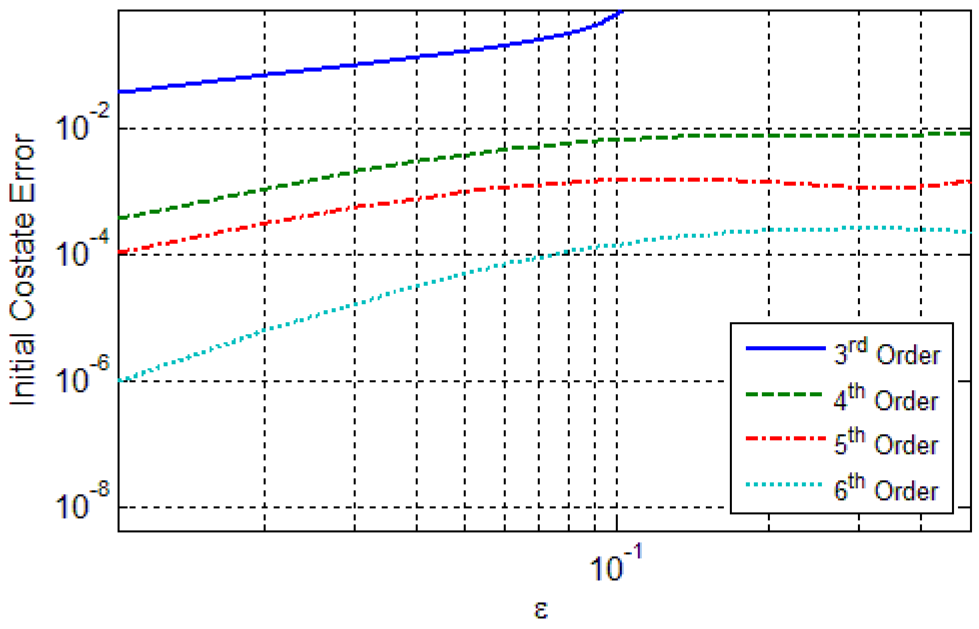

Figure 2.7 Initial costate error as a function of *ε*, for 3rd through 6th order approximations.

3. NUMERICAL EXAMPLES

The following section applies the HJB approximation methodology to nonlinear, multiple state, Aerospace-oriented optimal control problems. Section 3.1 provides an overview of the numerical implementation of the approximation method. Section 3.2 presents a solution for the optimal stabilization of a spacecraft. Finally, Section 3.3 presents a highly nonlinear, minimum-fuel, co-planar orbit transfer problem.

3.1 Numerical Implementation

The Aerospace-oriented problems addressed in this section represent a significant increase in dimensionality and complexity. Because of this, computer aided differentiation is utilized to greatly reduce the effort required for the numerical implementation of the approximation method. Computer aided differentiation is typically accomplished using either symbolic or automatic differentiation. For this research, both methods of differentiation were investigated.

First, a symbolic differentiation tool was developed in *Matlab*, using the *Maple symbolic toolbox for Matlab* [21]. The symbolic routine is employed to differentiate the HJB equation to an arbitrary order, with respect to an arbitrary number of independent variables. This produces a large volume of symbolic equations, which are automatically stored in *C* files. The *C* files are then dynamically linked to *Matlab* as Executable files (MEX-files) [22]. Storing the symbolic equations in *C* files offers two benefits. First, the symbolic code is stored in an optimized structure within the *C* file, which reduces the size of the file. Secondly, the *C* files are compiled before simulation and run much faster than regular *Matlab M*-files. This procedure is largely automated, eliminating most of the burden normally associated with symbolic differentiation. The resulting *C* file provides the differential equations needed to integrate the cost-to-go and its partial derivatives via the *method of lines*. All integrations are performed in *MATLAB*, using a 4th order variable step *Runge-Kutta* integration routine.

For problems formulated with state-space dimensions of two or more, the computational efficiency of the approximation method is greatly improved by exploiting the symmetry found in the tensor structured orders of partial derivatives of the cost-to-go function. Therefore, the aforementioned symbolic differentiation routine accounts for the symmetrical properties by only generating differential equations for unique partial derivatives of the cost-to-go function. A more detailed discussion of the properties of the tensor structured orders of partial derivatives, and the measures taken to handle them in *Matlab* is given in Appendix C.

Next, two automatic differentiation approaches were investigated. Automatic differentiation works by simultaneously deriving and evaluating partial derivatives in the background, during the integration routine. In the first approach considered, an Object-Oriented Cartesian Embedding Algorithm (OCEA) [23], developed by Dr. James Turner, was implemented in *FORTRAN*. OCEA operates by automatically invoking the chain rule of calculus. In its current form, OCEA is only capable of generating 1st through 4th order partial derivatives. In the second approach considered, automatic differentiation was performed using differential algebraic techniques within the *COSY Infinity* system

[24], developed by Dr. Martin Berz. Unlike OCEA, *COSY Infinity* is capable of performing differentiation to an arbitrary order.

When applied to moderately non-linear problems, all three approaches provided a fast and efficient means of solving the HJB approximation method. However, several of the examples presented in this research represent highly non-linear problems for which extremely high orders of differentiation are necessary. For these problems, the symbolic differentiation routine was found to be the best approach. OCEA was not utilized due to its inability to provide derivatives higher than 4th order. *COSY Infinity* was not selected because the speed and performance of the program tended to decline at high orders of approximation. Therefore, all of the examples presented in this thesis utilize the symbolic differentiation routine in *Matlab*.

3.2 Spacecraft Stabilization Problem

This section presents the optimal stabilization of a tumbling spacecraft [25]. The optimal control problem is stated as:

Minimize:

$$\mathcal{J} = \frac{1}{2}\int_0^{t_f} (\omega^T [I]\omega + u^T u)\ d\tau \tag{3.1}$$

Subject to:

$$\dot{\omega} = [I]^{-1}(u - [\tilde{\omega}][I]\omega) \tag{3.2}$$

$$\psi[t_f] = \omega(t_f) \tag{3.3}$$

where $\omega = [\omega_1, \omega_2, \omega_3]^{\mathrm{T}} \in \Re^3$ is a vector of angular velocities, $u \in \Re^3$ is a vector of control torques, and $[I] \in \Re^{3\times3}$ is the moment of inertia matrix. Additionally, $[\tilde{\omega}] \in \Re^{3\times3}$ is a vector cross product matrix of angular velocities [26], which is given as:

$$[\tilde{\omega}] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.4}$$

The moment of inertia (kg-m^2) for this asymmetric spacecraft is represented in the principal axis system as [25]:

$$[I] = \begin{bmatrix} 86.24 & 0 & 0 \\ 0 & 85.07 & 0 \\ 0 & 0 & 113.59 \end{bmatrix} \tag{3.5}$$

The initial angular velocities and final time for the problem are given as:

$$\omega = [-0.4 \quad 0.8 \quad 2]^T \;\; , \;\; t_f = 2 \tag{3.6}$$

The Hamiltonian of this system is constructed as:

$$H = \frac{1}{2}(\omega^T[I]\omega + u^T u) + \lambda^{\mathrm{T}}[I]^{-1}(u - [\tilde{\omega}][I]\omega) \tag{3.7}$$

where $\lambda = \left[\lambda_{\omega_1}, \lambda_{\omega_2}, \lambda_{\omega_3}\right]^{\mathrm{T}} \in \Re^3$ is a vector of costates. Differentiating the Hamiltonian with respect to the control produces the following linear relationship.

$$u = -[I]^{-1}\lambda \tag{3.8}$$

This relationship is used to remove u from the Hamiltonian. Next, the HJB equation is defined as:

$$-\frac{\partial J}{\partial t} = \frac{1}{2}\omega^T[I]\omega - {J_\omega}^T[I]^{-2}J_\omega - {J_\omega}^{\mathrm{T}}[I]^{-1}[\tilde{\omega}][I]\omega \tag{3.9}$$

where $J_\omega \in \Re^3$ is a vector of partial derivatives of the cost-to-go with respect to ω. For this problem, the cost-to-go is defined as:

$$J = \nu^T \omega\left(t_f\right) + \frac{1}{2}\int_t^{t_f} (\omega^T [I] \omega + u^T u)\ d\tau \tag{3.10}$$

where $\nu \in \Re^3$ is a vector of terminal constraint Lagrange multipliers. The most efficient way to deal with the Lagrange multipliers is to append them as additional states. Thus, the first two partial derivatives of the cost-to-go with respect to the "states" are given as:

$$J_x = \begin{bmatrix} J_\omega \\ J_\nu \end{bmatrix} \quad , \quad J_{xx} = \begin{bmatrix} J_{\omega\omega} & J_{\omega\nu} \\ J_{\omega\nu} & J_{\nu\nu} \end{bmatrix} \tag{3.11}$$

To accurately solve the TPBVP, the HJB equation is approximated to the 5th order. At the 5th order, there are 462 unique partial derivatives of J with respect to ω and ν. Therefore, 462 unique PDEs are derived using the symbolic differentiation routine. The conditions needed to start the integration are found by differentiating the cost-to-go at the final time with respect to ω and ν, and fixing the angular velocities at their initial conditions. The only non-zero boundary conditions are:

$$J = \nu^T \omega(0) \quad , \quad J_\omega = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{bmatrix} \quad , \quad J_\nu = \begin{bmatrix} \omega_1(0) \\ \omega_2(0) \\ \omega_3(0) \end{bmatrix} \quad , \quad J_{\omega\nu} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Because the proper values of ν are unknown, and iterative procedure is implemented. At each step, a fourth order series reversion process is applied to update ν. After three iterations (starting from $\nu = [0,0,0]^T$), the values of ν converge on the approximate solution. The results from each iteration, along with the optimal solution, are shown in Table 3.1.

Table 3.1 Spacecraft stabilization results

Iteration	ν_1	ν_2	ν_3	$\lambda_{\omega_1}(0)$	$\lambda_{\omega_2}(0)$	$\lambda_{\omega_3}(0)$
1	-2992.7	1447.2	12819.9	-67.9	134.0	449.1
2	-2987.8	1434.0	12803.2	-1507.5	2955.8	13065.4
3	-2987.8	1434.1	12803.3	-1509.2	2943.4	13053.3
Optimal	-2986.4	1430.6	12802.9	-1510.5	2939.9	13054.2

The table shows that even after convergence, error exists in both $\Lambda(0)$ and ν. This error stems from the truncation of the partial derivatives at the 5th order. To observe the effect this error has on the solution of the TPBVP, the problem is also solved using the optimal initial costates. The state and costate error between these two solutions is shown in Figure 3.1. As the Euler-Lagrange equations are integrated forward in time, the initial costate error propagates throughout the trajectory.

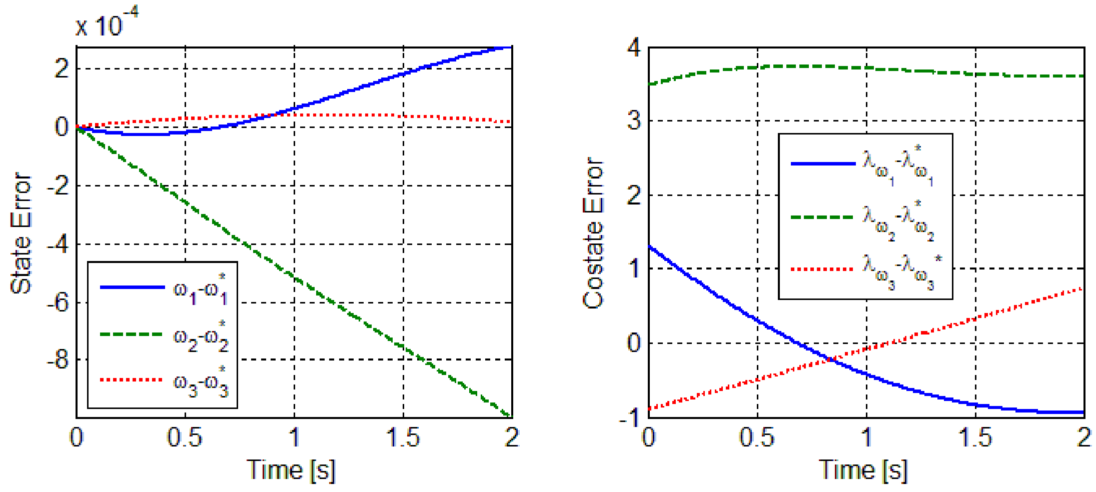

Figure 3.1 State and costate error for the terminally constrained, 5th order approximate spacecraft stabilization solution.

The motivation behind enforcing hard terminal constraints on the trajectory is to achieve extremely accurate results. However, due to the lack of feedback control scheme and the error associated with calculating the terminal constraint Lagrange multipliers, the approximation method fails to exactly satisfy the hard terminal constraints. Furthermore, the hard terminal constraint approach requires the full set of PDEs to be integrated for each iteration. This begs the question: Is the implementation burden of the terminal constraint method worth the inexact results it provides? To answer this question, the problem is re-solved without hard terminal constraints. Instead, a final state penalty function (soft terminal constraint) is used to stabilize the spacecraft. The final state penalty function takes the form:

$$\phi\left[x\left(t_f\right)\right] = \tfrac{1}{2}\omega^{\mathrm{T}} W \omega \qquad (3.13)$$

where $W \in \Re^{3\times 3}$ is a diagonal matrix of final state weights. The benefit of using the penalty function method is twofold. First, all dependency of the cost-to-go function on the terminal constraint Lagrange multipliers is eliminated. Because of this, the 5th order approximation method contains only 56 unique partial derivatives, instead of 462. Additionally, the solution process will no longer be iterative. Instead, a fixed number of equations are integrated one time to produce the optimal solution.

The effectiveness of the penalty function method is determined by three parameters: the nonlinearity of the system, the order of approximation applied to the HJB equation, and the size of the final state weight. The nonlinearity of the system is determined by the system dynamics and boundary conditions. The order of approximation has already been chosen (to match the order used in the terminal constraint method). This leaves only the size of the final state weights to be determined. To simplify the selection process, uniform weights are applied. The value of these weights is chosen to minimize the final angular velocities (maximize the stabilization performance). The variation in the final angular velocities is shown as a function of the weight values in Figure 3.2. The figure shows that as the weight values are increased, the effectiveness of the penalty function is increased. In fact, at extremely high final state weighting values, the penalty function method outperforms the terminal constraint method.

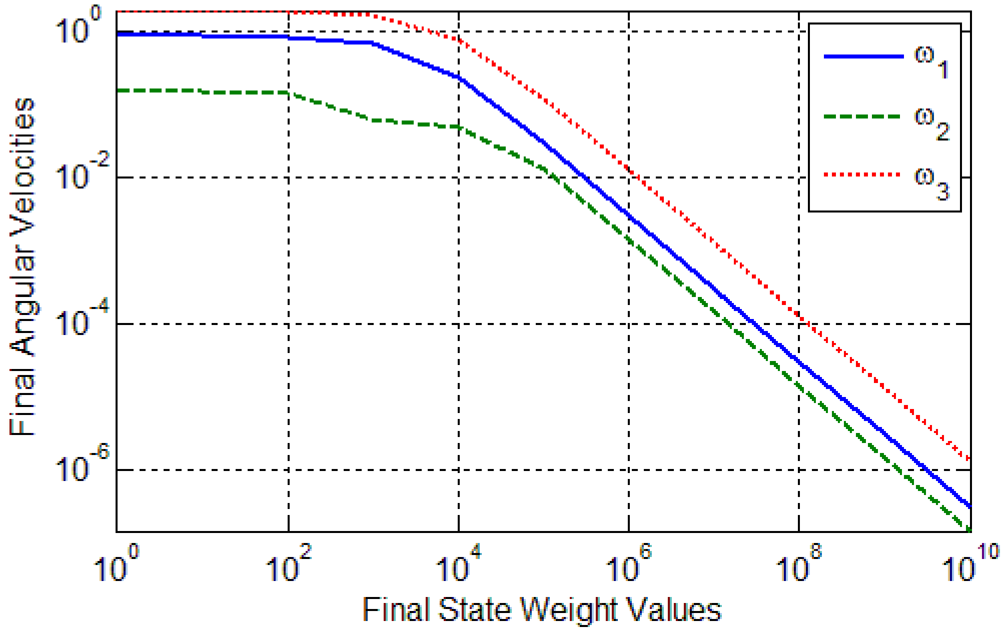

Figure 3.2 Final angular velocity as a function of the final state weight.

Based on Figure 3.2, the final state weight values are chosen to be 10^{10}. The final state weights in this problem are set very high because the problem was not formulated with non-dimensional variables. In general, it is best to scale the variables such that they are bounded by ±1. From the approximation method, the initial costates are calculated as:

$$\Lambda(0) = [-1510.39\ ,\ 2939.99\ ,\ 13053.78] \tag{3.14}$$

These initial costates are much closer to the optimal values, than those obtained in the terminally constrained solution method. However, some approximation error still exists in this method. To observe the propagation of the approximation error through this solution, the difference between the optimal and approximate solutions is shown in Figure 3.3.

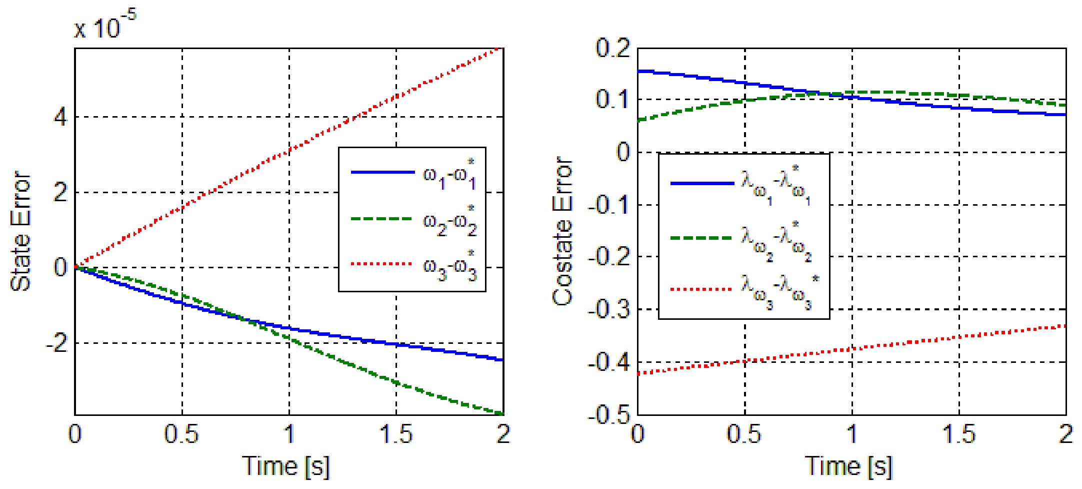

Figure 3.3 State and costate error for the penalized, 5th order approximate spacecraft stabilization solution

Overall, it is much more advantageous to solve this problem using the penalty function approach in place of hard terminal constraints. The penalty function approach is non-iterative, requires less derivation and integration, and can produce accurate results. The hard terminal constraint approach requires an unknown number of iterations to solve for the proper values of the terminal constraint Lagrange multipliers. Each iteration requires the integration of the full set of PDEs via the *method of lines*. This repetitive integration of a large set of PDEs poses an unfavorable computational burden. Therefore, the penalty function approach will be applied to solve the terminally constrained problems found in the remainder of this thesis.

3.3 Orbit Transfer Problem

This section presents the solution to a minimum-fuel, co-planar orbit transfer problem. Two distinct cases are presented to illustrate the approximation method. The system is defined in a heliocentric reference frame, and the dynamics are described in polar coordinates by the following equations [27].

$$\dot{r} = v \tag{3.15}$$

$$\dot{v} = \frac{w^2}{r} - \frac{\mu}{r^2} + u_r \tag{3.16}$$

$$\dot{w} = -\frac{vw}{r} + u_t \tag{3.17}$$

where r is the radial distance from the sun, v is the radial velocity, w is the tangential velocity, u_r and u_t are the radial and tangential thrust terms, and μ is the gravitational constant. The problem is non-dimensionalized such that the value of μ is 1. A minimum-fuel performance index is given as

$$\mathcal{J} = \frac{1}{2}\int_{t_0}^{t_f}\left(u_r{}^2 + u_t{}^2\right)\, d\tau \tag{3.18}$$

The problem is subject to the following terminal constraints.

$$\psi[t_f] = \begin{bmatrix} r(t_f) - r_f \\ v(t_f) - v_f \\ w(t_f) - w_f \end{bmatrix} \tag{3.19}$$

where r_f, v_f, and w_f are the desired values of the final states.

To begin the developments, the TPBVP is constructed. As a first step, the Hamiltonian is defined as follows.

$$H = \frac{1}{2}\left(u_r{}^2 + u_t{}^2\right) + \lambda_r(v) + \lambda_v\left(\frac{w^2}{r} - \frac{\mu}{r^2} + u_r\right) + \lambda_w\left(-\frac{vw}{r} + u_t\right) \tag{3.20}$$

Because the system is affine in the control, differentiating the Hamiltonian with respect to u_r and u_t provides the following linear relationships.

$$u_r = -\lambda_v \quad , \quad u_t = -\lambda_w \tag{3.21}$$

The costate differential equations are derived from the first-order necessary conditions for optimality as:

$$\dot{\lambda}_r = \lambda_v\left(\frac{w^2}{r^2} - \frac{2\mu}{r^3}\right) - \lambda_w\left(\frac{vw}{r^2}\right) \tag{3.22}$$

$$\dot{\lambda}_v = -\lambda_r + \lambda_w\left(\frac{w}{r}\right) \tag{3.23}$$

$$\dot{\lambda}_w = -\lambda_v\left(\frac{2w}{r}\right) + \lambda_w\left(\frac{v}{r}\right) \tag{3.24}$$

Finally, the HJB equation is constructed as:

$$-\frac{\partial J}{\partial t} = -\frac{1}{2}\left(J_v{}^2 + J_w{}^2\right) + J_r(v) + J_v\left(\frac{w^2}{r} - \frac{\mu}{r^2}\right) + J_w\left(-\frac{vw}{r}\right) \tag{3.25}$$

The HJB equation is approximated by differentiating the HJB equation a finite number of times. The number of differentiations needed to accurately solve the problem is dependent on the nonlinearity of the system. Therefore, the order of approximation will be addressed individually for each of the two orbit transfer cases presented below.

Once again, a final state penalty function is used in place of the hard terminal constraints. The penalty function is given as

$$\phi = \frac{1}{2}\psi\left[t_f\right]^{\mathrm{T}} W\,\psi\left[t_f\right] \tag{3.26}$$

Where $W \in \Re^{3\times3}$ is a diagonal matrix of final state weights. Again, this penalty function is being used because it provides a non-iterative method, for which fewer partial derivatives of the cost-to-go are required for each order of approximation. Using the penalty function approach, the cost-to-go and its non-zero partial derivatives at the final time are given as:

$$J\left(t_f\right) = \frac{1}{2}\psi[0]^{\mathrm{T}}\, W\,\psi[0] \quad , \quad J_x = W\,\psi[0] \quad , \quad J_{xx} = W \tag{3.27}$$

where $\psi[0]$ represents the terminal constraint function with the final states fixed at the initial state values.

3.3.1 Low Thrust Transfer

In the first case, a low thrust transfer is propagated over a long period of time. Due to the length of the trajectory, this problem is difficult to solve via the HJB approximation method. The boundary conditions for this case are shown in Table 3.2

Table 3.2 Boundary conditions for the low thrust transfer

Variable	Initial	Final	Units
Time	0	900	days
Radial Distance	1	1.25	AU
Radial Velocity	0	0	AU/TU
Tangential Velocity	1	$\sqrt{1/1.25}$	AU/TU

Before the problem is solved, two parameters must be chosen: the order of approximation and the value of the final state weights. To select these parameters, the approximation method is implemented for range of final state weights and orders of approximation. With each combination of parameters, the initial costates are computed. From these initial costates, an "approximation error" is determined as follows:

$$Approximation\ Error = mean(abs(\lambda_0 - \lambda_0^*)) \tag{3.28}$$

where λ_0 is the vector of approximate initial costates and λ_0^* is the vector of optimal initial costates (found using a shooting algorithm). Figure 3.4 shows the approximation error for the combinations of final state weights and approximation order considered for this study. Two main insights can be drawn from the figure. First, increasing the final state weight improves the accuracy of the approximation. However, at some point, an increase in the final state weight begins to diminish the accuracy of the approximation. The point at which this decline is observed is directly related to the order of approximation. In other words, increasing the approximation order increases the final state weight value at which a decline in performance is observed.

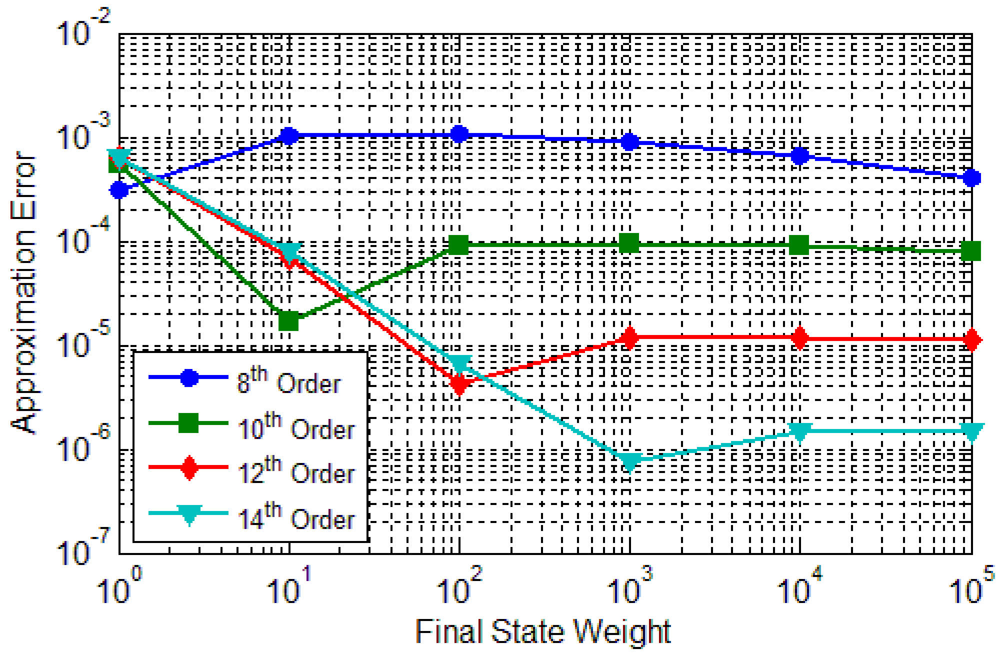

Figure 3.4 Terminal constraint error as a function of final state weight and approximation order for the low thrust transfer.

From this figure, the order of approximation is chosen to be 14th order and the final state weight value is chosen to be 1000. A larger order of approximation would yield more accurate results, but would require an enormous computational burden. With a 14th order approximation, 680 unique partial derivatives of the cost-to-go function exist. Therefore, the symbolic differentiation routine is charged with deriving and storing 680 PDEs that govern these partial derivatives.

The nominal trajectory is generated by integrating the Euler-Lagrange equations from the initial boundary conditions, using the initial costates obtained through the approximation method. The nominal trajectory is shown in Figure 3.5. This trajectory is sub-optimal, due to the truncation involved in the HJB approximation method and the application of soft terminal constraints.

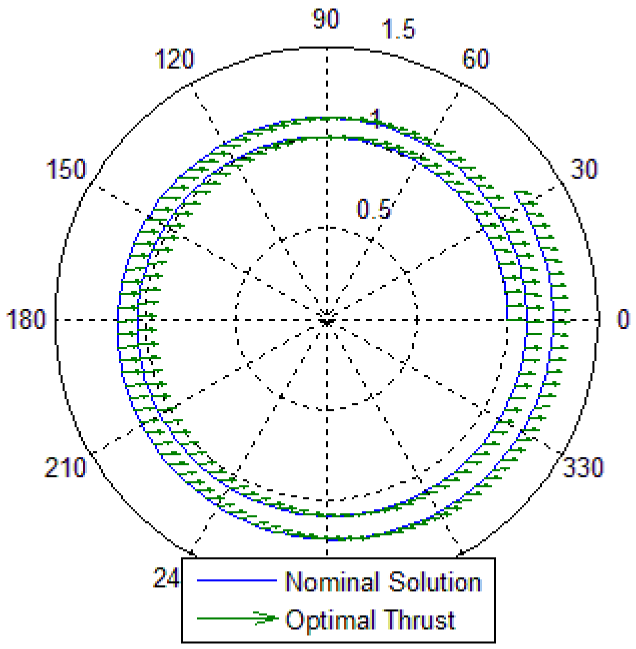

Figure 3.5 Approximate nominal solution for the low thrust transfer.

To observe the inaccuracy of the sub-optimal trajectory, it will be compared to the optimal trajectory for this problem. The optimal trajectory has no approximation error and is subject to hard terminal constraints. Figure 3.6 shows the state and costate errors between the optimal and sub-optimal trajectories. The figure shows a relatively low amount of state and costate error for this problem. Error propagation throughout the state trajectories is evident in Figure 3.6. Again, this propagation is attributed to the lack of a feedback control law.

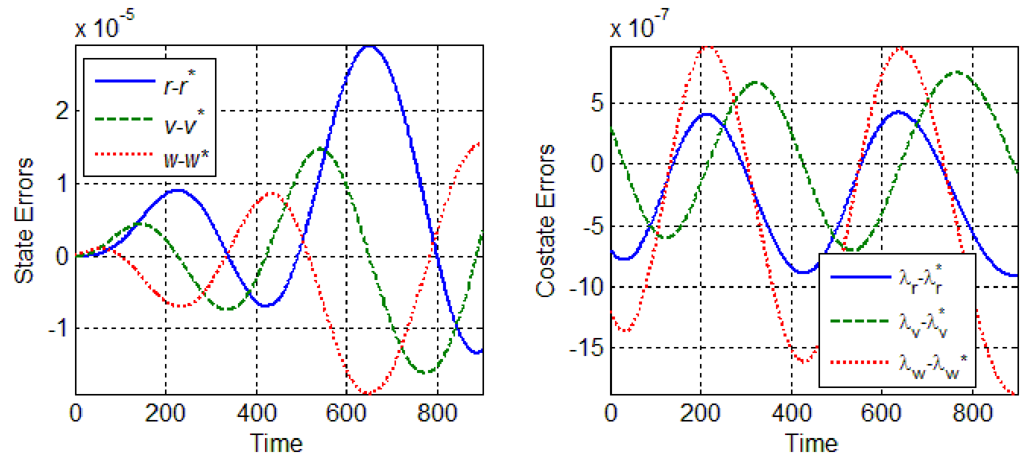

Figure 3.6 State and costate errors between the optimal and sub-optimal solutions to the low thrust transfer.

3.3.2 Earth to Mars Transfer

In the second case, the boundary conditions of the TPBVP were chosen to represent an Earth to Mars orbit transfer, for which both the initial and final orbits are circular. The boundary conditions are shown in Table 3.3.

Table 3.3 Boundary conditions for the Earth to Mars transfer

Variable	Initial	Final	Units
Time	0	210	days
Radial Distance	1	1.5	AU
Radial Velocity	0	0	AU/TU
Tangential Velocity	1	$\sqrt{1/1.5}$	AU/TU

Again, the "approximation error" is calculated for a variety of final state weights and orders of approximation, as shown in Figure 3.7. The same two insights drawn from Figure 3.4 are evident in this figure. However, because the Earth to Mars transfer represents a more non-linear problem, the point at which a decline in performance is observed occurs at lower values of the final state weight.

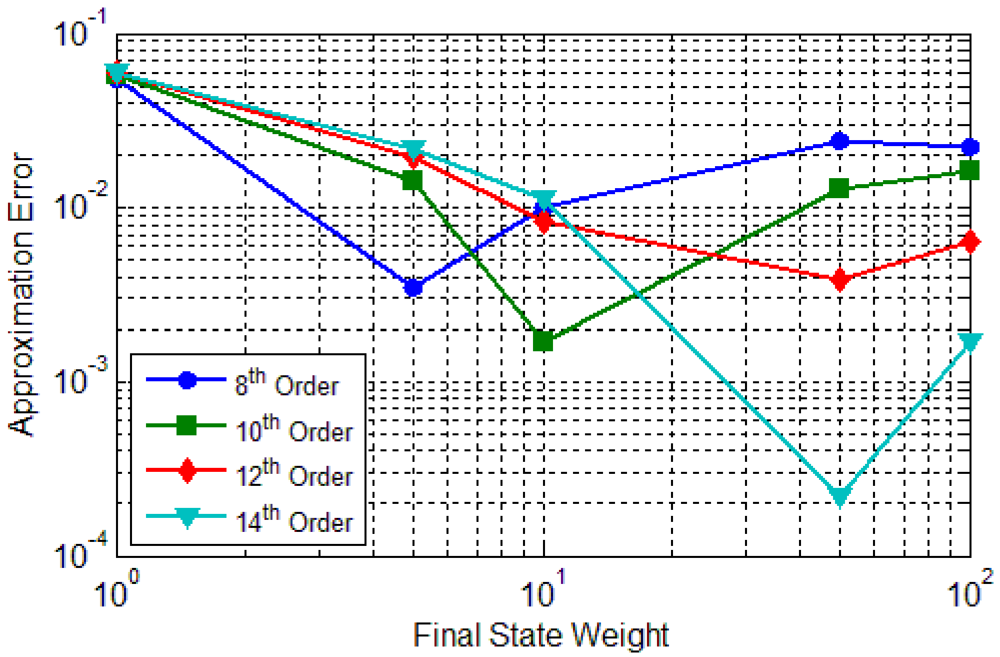

Figure 3.7 Terminal constraint error as a function of final state weight and approximation order for the Earth to Mars transfer.

From this figure, the order of approximation is chosen to be 14th order and the final state weight value is chosen to be 50. A larger order of approximation would yield more accurate results, but would require an enormous computational burden. The nominal trajectory is generated by integrating the Euler-Lagrange equations from the initial boundary conditions, using the initial costates obtained through the approximation method. The nominal trajectory is shown in Figure 3.8. This trajectory is sub-optimal, due to the truncation involved in the HJB approximation method and the application of soft terminal constraints.

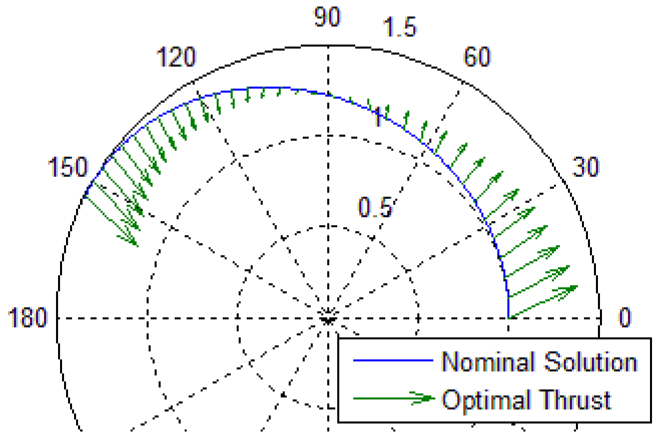

Figure 3.8 Approximate nominal solution for the Earth to Mars transfer.

To observe the inaccuracy of the sub-optimal trajectory, it will be compared to the optimal trajectory for this problem. The optimal trajectory has no approximation error and hard terminal constraints. Figure 3.9 shows the state and costate errors between the optimal and sub-optimal trajectories. At this approximation order, there is still a considerable amount of initial costate error, which leads to an error in the final states.

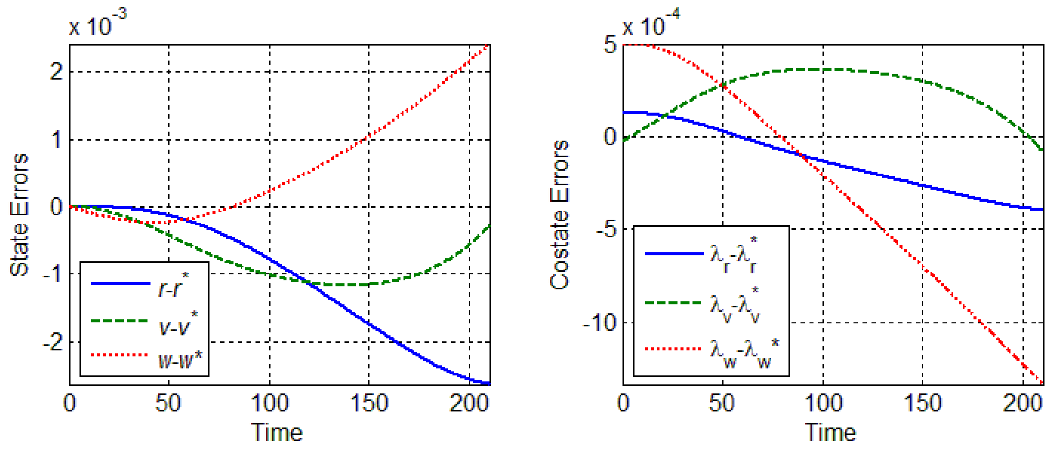

Figure 3.9 State and costate errors between the optimal and sub-optimal solutions for the Earth to Mars transfer.

As previously mentioned, the sensitivities of the initial costates to changes in the initial states are produced as a byproduct of the HJB approximation method. For a 14th order HJB approximation, 13 orders of sensitivities are produced. As outlined in Section 2.1, these sensitivities are used to produce a family of neighboring paths, about the nominal solution. To observe this benefit of the approximation method, the initial radial distance was varied over a range [0.9 1.1] AU. With each radial distance variation, the tangential velocity was altered such that the initial orbit remained circular. The costate sensitivities (up to 8th order) were used to update the initial costates. The 9th through 14 order sensitivities have virtually no effect on the costate update equation. Figures 3.10 through 3.13 show the solutions to the neighboring TPBVPs, using the updated costates.

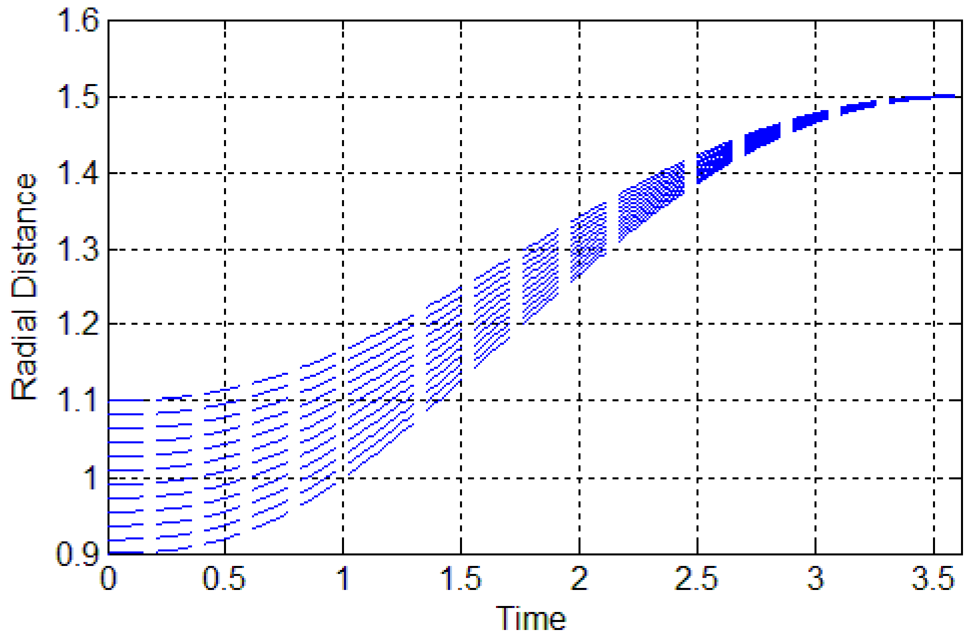

Figure 3.10 Neighboring radial distances for the Earth to Mars transfer.

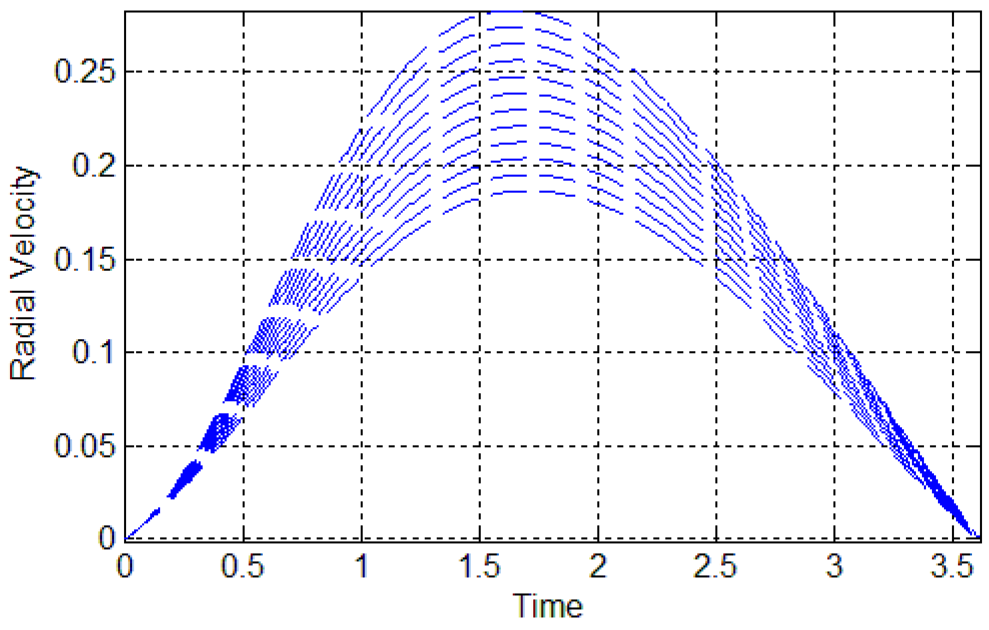

Figure 3.11 Neighboring radial velocities for the Earth to Mars transfer.

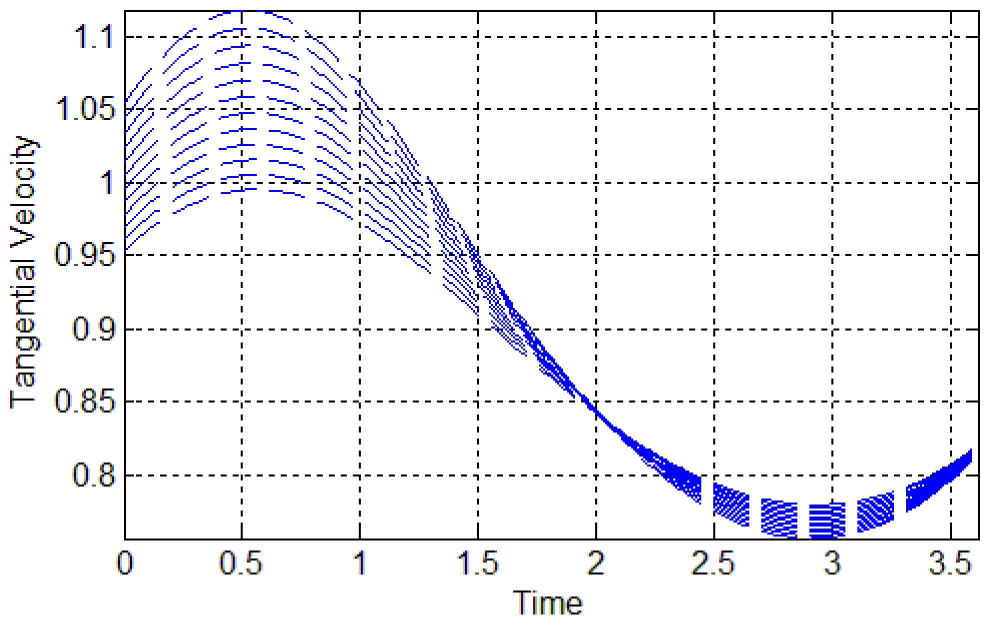

Figure 3.12 Neighboring tangential velocities for the Earth to Mars transfer.

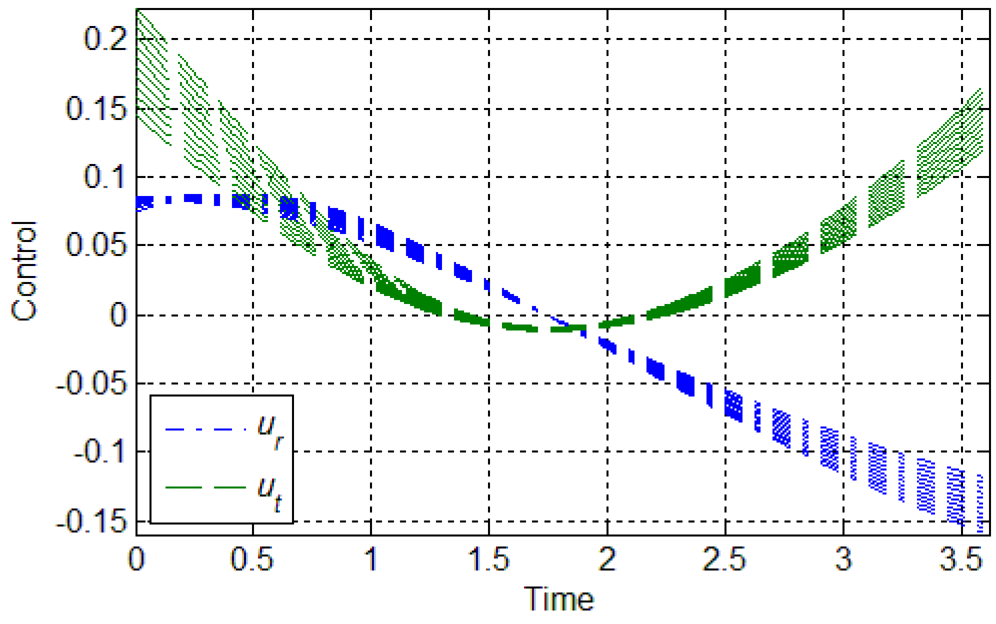

Figure 3.13 Neighboring controls for the Earth to Mars transfer.

4. HIGH-ORDER GUIDANCE SOLUTIONS

Section 3.3 demonstrated the need for extremely high orders of approximation to accurately solve a highly nonlinear optimal control problem. In the setting of a coplanar orbit transfer problem, with a state-space dimension of three, this approximation was feasible. However, as problem complexity and dimensionality increase, approximation to high orders becomes increasingly computationally expensive. Hence, it would be desirable to formulate a solution methodology for which lower orders of approximation would provide accurate solutions.

One such alternative is to utilize the HJB approximation methodology as a tool for generating corrections to the nominal control law. In this application, the method no longer produces the nominal optimal solution, but still provides the initial costate sensitivities needed to generate a field of extremals. As a result, the problem is significantly less nonlinear, which eliminates the need for extremely high orders of approximation. The general problem of producing control corrections to generate neighboring optimal trajectories is often referred to as the guidance problem [28].

Section 4.1 describes the application of the HJB approximation method to solve the guidance problem. Section 4.2 describes the application of an alternative guidance approach, which utilizes the Lagrange Implicit Function Theorem. In Section 4.3, the minimum-fuel, co-planar Earth to Mars transfer problem is re-solved in the guidance scheme. Finally, a three-dimensional re-entry guidance problem is solved in Section 4.4.

4.1 Perturbed HJB Equation Approximation

To implement the HJB approximation procedure about the nominal trajectory, the solution to the variation in the cost-to-go (δJ) is approximated. The equation governing this "perturbed" cost-to-go is referred to as the perturbed HJB (PHJB) equation, which is defined as[‡]

$$-\frac{\partial}{\partial t}(\delta J) = H(x^* + \delta x, \lambda^* + \delta\lambda) - H^* - H_x^{*T}\delta x - H_\lambda^{*T}\delta\lambda \tag{4.1}$$

where x^* are the nominal states, λ^* are the nominal costates, H^* is the nominal Hamiltonian, δx are the perturbed states, and $\delta\lambda$ are the perturbed costates. The nominal state, costate, and Hamiltonian values must be obtained *a priori*, using any of the available optimal control solution methods, e.g., the Pseudospectral method or any other direct approach. Again, a key relationship connecting the costates to the first order partial derivatives of the cost-to-go function is given as:

$$\delta\lambda = \delta \mathrm{J}_{\delta \mathrm{x}} \tag{4.2}$$

The solution to the PHJB equation is approximated in the same manner as the HJB equation, i.e., successive partial derivatives of the PHJB equation are taken with respect to the perturbed states. The first two partial derivatives are defined as:

$$-\frac{\partial}{\partial t}(\delta J_{\delta x}) = H_{\delta x}(x^* + \delta x, \lambda^* + \delta\lambda) - H_x^* - \delta J_{\delta x\delta x}\ H_\lambda^* \tag{4.3}$$

$$-\frac{\partial}{\partial t}(\delta J_{\delta x\delta x}) = H_{\delta x\delta x}(x^* + \delta x, \lambda^* + \delta\lambda) - \delta J_{\delta x\delta x\delta x}\ H_\lambda^* \tag{4.4}$$

Again, each PDE is dependent on the next higher order of partial derivatives. Thus, the partial derivatives must again be truncated at the *n*th order, which will affect the accuracy

[‡] A full derivation of the PHJB equation is provided in Appendix D

of the 1st through n-1th orders of partial derivatives. The symbolic PDEs governing the partial derivatives of the perturbed cost-to-go are again integrated via the method of lines, holding the perturbed states fixed at their initial conditions. Using soft terminal constraints in place of hard ones, the boundary condition for the perturbed cost-to-go is given as:

$$\delta J(t_f) = \phi[x^* + \delta x] - \phi[x^*] \tag{4.5}$$

Partial differentiation of this condition with respect to the perturbed states leads to the boundary conditions for the partial derivatives of the perturbed cost-to-go, which are needed to initialize the integration process.

The PHJB approximation is implemented about the nominal trajectory by assuming zero variation in the initial states, i.e., $\delta x = 0$. If the approximation is accurate, this implementation leads to the trivial solutions:

$$\delta J(0) = 0 \quad , \quad \delta J_{\delta x}(0) = 0 \tag{4.6}$$

These conditions provide a good check as to the accuracy of the approximation. Meanwhile, the 2nd through nth orders of partial derivatives of the perturbed cost-to-go provide the 1st through n-1th orders of sensitivities of the nominal costates to variations in the initial states. These costate sensitivities are used to approximate neighboring optimal initial costates, which are used to generate near-optimal solutions to neighboring TPBVPs.

In many problems, it is beneficial to calculate sensitivities of the initial costates to variations in certain parameters of the system. The PHJB method is modified to incorporate these sensitivities by treating the desired parameters as additional states. In doing this, the cost-to-go is assumed to be a function of both the states and desired system parameters. Thus, the PHJB equation is differentiated with respect to both the states and desired system parameters. The only discrepancy between the states and parameter sensitivities is in their relationship to the costates. The first order parameter sensitivities are not equivalent to the costates, and will not be used in the costate update equation.

4.2 Lagrange Implicit Function Theorem

In this solution method, the implicit function theorem is applied to the terminal conditions for optimality [17]. In doing so, equations containing the sensitivity of the initial costates to changes in the initial states are produced. These equations are then inverted, to provide solutions for the initial costate sensitivities. In this thesis, the LIFT method is only applied to optimal control problems with known initial states and linear constraints on the terminal states. Therefore, the methodology described in this section will be tailored to this specific class of problems.

To begin, the linear terminal constraint is given as follows.

$$\psi\left[X\left(t_f\right)\right] = X\left(t_f\right) - X_f = 0 \tag{4.7}$$

where $X\left(t_f\right)$ is the set of optimal states at the final time and X_f is a set of desired final state values. Again, the objective of the guidance solution is to correct the initial costates such that the terminal constraints remain satisfied in the event of changes in the initial states. In mathematical terms, this means that the total derivative of the terminal constraint with respect to the initial states should be equal to zero.

$$\frac{d}{dX_0}\psi[X(t_f)] = \frac{d}{dX_0}X(t_f) = 0 \tag{4.8}$$

Because $X(t_f)$ is an implicit function of the initial states and costates, the total derivative of $X(t_f)$ can be split into partial derivatives as follows.

$$\frac{d}{dx_0}X(t_f) = X(t_f)_{X_0} + X(t_f)_{\Lambda_0}\frac{d\Lambda_0}{dX_0} = 0 \tag{4.9}$$

This equation can be rearranged to solve for the sensitivities of the initial costates to changes in the initial states as follows.

$$\frac{d\Lambda_0}{dX_0} = -X(t_f)_{\Lambda_0}^{\;\;-1}\; X(t_f)_{X_0} \tag{4.10}$$

Higher order costate sensitivities are found by taking higher total derivatives of the terminal constraint with respect to the states. These total derivatives can also be split into partial derivatives, revealing similar expressions for the higher order costate sensitivities [17]. A more detailed description of this process is provided in Appendix E.

Before these costate sensitivities can be calculated, $X(t_f)_{X_0}$ and $X(t_f)_{\Lambda_0}$ must be calculated. These sensitivities are propagated throughout the nominal solution, starting from the following known initial conditions.

$$\begin{bmatrix} X(t_0)_{X_0} & X(t_0)_{\Lambda_0} \\ \Lambda(t_0)_{X_0} & \Lambda(t_0)_{\Lambda_0} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.11}$$

It is necessary to propagate the partial derivatives of both the states and costates, because of the coupling that occurs between the states and costates in the Euler-Lagrange equations, as seen below.

$$\frac{d}{dt}\begin{bmatrix} X(t) \\ \Lambda(t) \end{bmatrix} = \begin{bmatrix} H_\Lambda \\ -H_X \end{bmatrix} \tag{4.12}$$

Differentiating the Euler-Lagrange equations with respect to the states and costates produces differential equations that govern the state and costate sensitivities exactly. These equations are shown below for the first order.

$$\frac{d}{dt}\begin{bmatrix} X(t)_{X_0} & X(t)_{\Lambda_0} \\ \Lambda(t)_{X_0} & \Lambda(t)_{\Lambda_0} \end{bmatrix} = \begin{bmatrix} H_{\Lambda \mathrm{X}} & H_{\Lambda\Lambda} \\ -H_{\mathrm{XX}} & -H_{\mathrm{X}\Lambda} \end{bmatrix} \begin{bmatrix} X(t)_{X_0} & X(t)_{\Lambda_0} \\ \Lambda(t)_{X_0} & \Lambda(t)_{\Lambda_0} \end{bmatrix} \tag{4.13}$$

Again, successive derivatives of the Euler-Lagrange equations are taken to obtain differential equations for 2nd and higher order partial derivatives.

The LIFT method can also be extended to handle sensitivities of the initial costates with respect to system parameters. To do so, the total derivative of the terminal constraint with respect to the parameters is separated into partial derivatives and inverted.

$$\frac{d\Lambda_0}{dp} = -X(t_f)_{\Lambda_0}^{\ \ -1}\ X(t_f)_p \tag{4.14}$$

Before these control sensitivities can be calculated, $X(t_f)_p$ must be computed. These partial derivatives are propagated about the nominal solution, starting from the following known initial conditions.

$$\begin{bmatrix} X(t_0)_p \\ \Lambda(t_0)_p \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.15}$$

Finally, differentiating the Euler-Lagrange equations with respect to the system parameters provides differential equations that propagate the parameter sensitivities throughout the nominal solution.

$$\frac{d}{dt}\begin{bmatrix} X(t_0)_p \\ \Lambda(t_0)_p \end{bmatrix} = \begin{bmatrix} f_{\mathrm{X}} & f_{\Lambda} \\ g_{\mathrm{X}} & g_{\Lambda} \end{bmatrix} \begin{bmatrix} X(t_0)_p \\ \Lambda(t_0)_p \end{bmatrix} + \begin{bmatrix} f_p \\ g_p \end{bmatrix} \tag{4.16}$$

4.3 Orbit Transfer Guidance Problem

In this example, the orbit transfer problem is re-solved as a guidance problem, accounting for variations in all three of the initial states. The nominal boundary conditions represent the same Earth to Mars transfer generated in Section 3.3.2. These conditions can be seen again in Table 4.1.

Table 4.1 Nominal orbit transfer boundary conditions

Variable	Initial	Final	Units
Time	0	210	days
Radial Distance	1	1.5	AU
Radial Velocity	0	0	AU/TU
Tangential Velocity	1	$\sqrt{1/1.5}$	AU/TU

This example is presented to compare the performance of the PHJB method to the LIFT method. Both guidance methods are implemented in *Matlab*, using a modified version of the symbolic differentiation routine to derive and store the appropriate PDEs. Hence, both methods consider the minimum number of partial derivatives required for solution. Both methods are employed to produce the first four orders of control sensitivities. A fifth-order approximation is required by the PHJB method to produce these sensitivities, while the LIFT method only requires four orders of differentiation. Due to the difference in methodologies, a different number of partial derivatives must be integrated about the nominal trajectory to solve each method. Table 4.2 provides an overview of the number of unique partial derivatives required in each method.

Table 4.2 Number of required partial derivatives for the orbit transfer guidance problem, considering variations in the initial states

Order	PHJB method	LIFT method
1	6	36
2	10	126
3	15	336
4	21	756

As Table 4.2 shows, the PHJB method computes the costate sensitivities much more efficiently than the LIFT method. Fundamentally, both guidance methods operate by taking successive derivatives of the Hamiltonian. However, the methods differ in the form of the Hamiltonian used by each. In the PHJB equation, the Hamiltonian is formulated to be independent of the system costates. This is achieved by replacing the costates with J_x. Alternatively, the Euler-Lagrange equations are formulated with a dependence on both the system states and costates. Thus, partial derivatives of the Euler-Lagrange equations must be computed with respect to twice as many independent variables as the partial derivatives of the PHJB equation. Furthermore, the 1st order matrix of partial derivatives of the Euler-Lagrange Equations is non-symmetric, while the equivalent (2nd order) matrix of partial derivatives of the HJB equation is symmetric about the diagonal.

To compare the performance of each method, the orbit transfer guidance problem is solved for variations in the initial states. The initial costate correction is determined through a Taylor series expansion about the variations in the initial states, δx_0, which is given as:

$$\delta\Lambda(0) = \Lambda(0)_{\mathrm{x}_0}\delta x_0 + \frac{1}{2!}\Lambda(0)_{\mathrm{x}_0\mathrm{x}_0}\delta {x_0}^2 + \frac{1}{3!}\Lambda(0)_{\mathrm{x}_0\mathrm{x}_0\mathrm{x}_0}\delta {x_0}^3 + \cdots \tag{4.17}$$

For this test case, the initial state variations are given as:

$$\delta x_0 = [\ 0.1 \quad 0 \quad -0.0465\ ]^{\mathrm{T}}. \tag{4.18}$$

The altered initial states represent a circular orbit with a radius of 1.1 AU. Again, the PHJB method utilizes a final state penalty function, instead of hard terminal constraints. As a result, the non-zero boundary conditions needed to initialize the integration are given as:

$$\delta J_{\delta x \delta x}\left(t_f\right) = \begin{bmatrix} W & 0 & 0 \\ 0 & W & 0 \\ 0 & 0 & W \end{bmatrix} \tag{4.19}$$

Again, the proper final state weight value W is found by simulating the PHJB method for a range of final state weight values. Figure 4.1 shows the approximation error calculated for different combinations of final state weight and approximation order. The figure shows that the accuracy of the approximation method is improved by increasing both the approximation order, as well as, the final state weight value. For this example, a 5th order approximation is chosen, and the final state weight value is selected to be $1\text{x}10^5$. As previously mentioned, a 5th order PHJB approximation provides four orders of costate sensitivities.

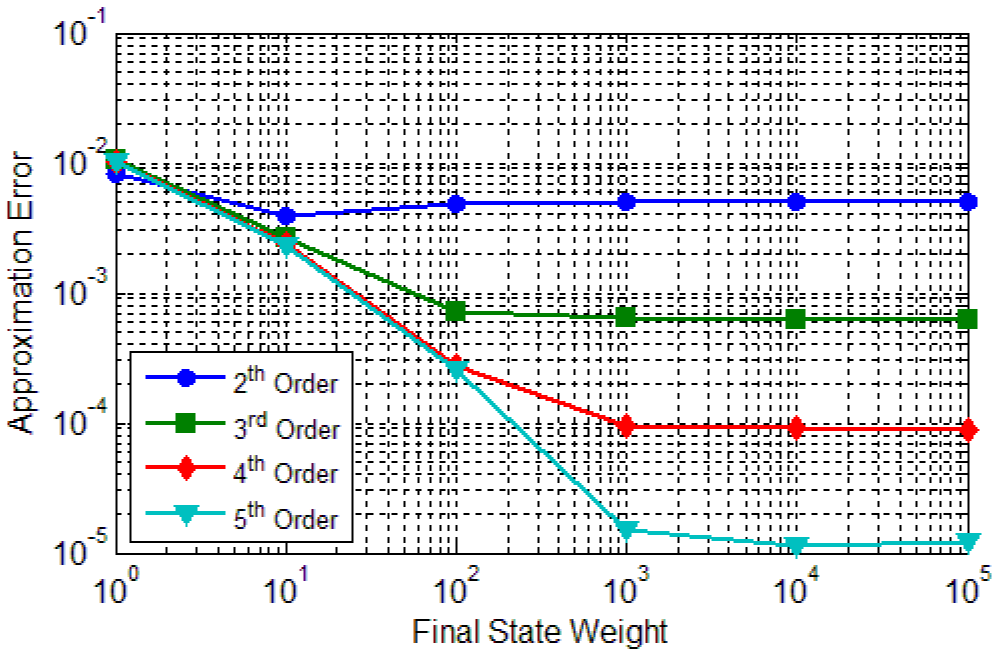

Figure 4.1 Approximation error as a function of the final state weight and approximation order for the orbit transfer guidance problem.

The results for each guidance method are shown in Table 4.3. The table shows that the two methods produce analogous results. In fact, the difference in the initial perturbed costates can be attributed to the tolerance chosen for the integration routine. The only significant difference between the two methods is in the runtime. The PHJB method runs approximately 5 times faster than the LIFT method. This significant savings in computational time is directly attributed to the reduction in the number of sensitivities required for solution.

Table 4.3 Orbit transfer guidance results

	PHJB Method	LIFT Method	Difference
$\delta\lambda_r(0)$	0.04082751	0.04082757	-5.9467×10^{-8}
$\delta\lambda_v(0)$	0.00181448	0.00181435	1.2670×10^{-7}
$\delta\lambda_w(0)$	0.04089636	0.04089652	-1.6053×10^{-7}
$r(t_f)$	1.50001442	1.50001237	2.0488×10^{-6}
$v(t_f)$	0.00012732	0.00012590	1.4201×10^{-6}
$w(t_f)$	0.81662828	0.81662879	-5.1628×10^{-7}
Cost	0.01377348	0.01377349	-1.0830×10^{-8}
Runtime (s)	0.17	0.85	0.68

Finally, a family of neighboring approximate trajectories is shown in Figures 4.2 through 4.5. These trajectories are generated using the costate sensitivities obtained from the PHJB method. However, if the trajectories generated with the LIFT method were to be plotted in the same figures, the two sets of trajectories would be indistinguishable. The neighboring trajectories are generated by varying the initial radius over a range [0.9 1.1] AU. With each radial distance variation, the tangential velocity was altered such that the initial orbit remained circular. The same family of neighboring trajectories was generated in Section 3.3, using the full 14th order HJB approximation. Comparing the two sets of neighboring trajectories, it is evident that the control sensitivities provided by the 5th order guidance solution are more accurate than those obtained as a byproduct of the full problem solution.

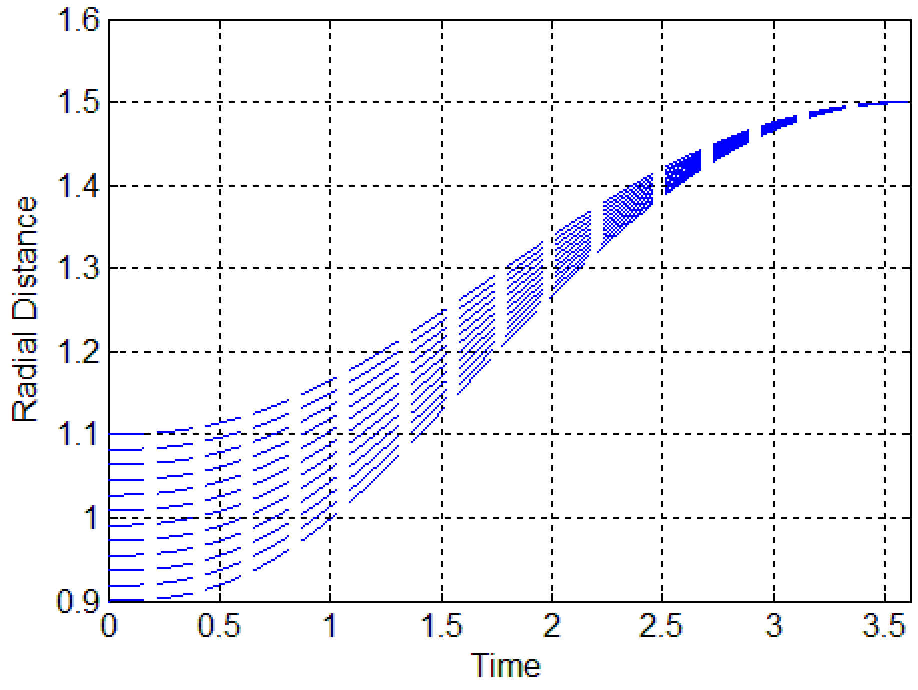

Figure 4.2 Neighboring radial distances for the orbit transfer guidance problem.

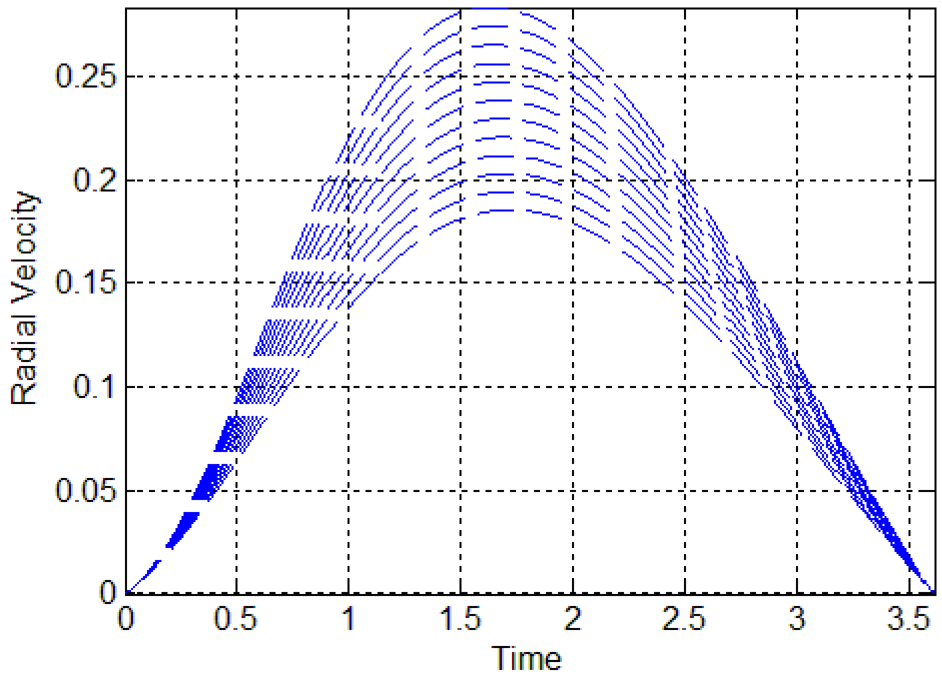

Figure 4.3 Neighboring radial velocities for the orbit transfer guidance problem.

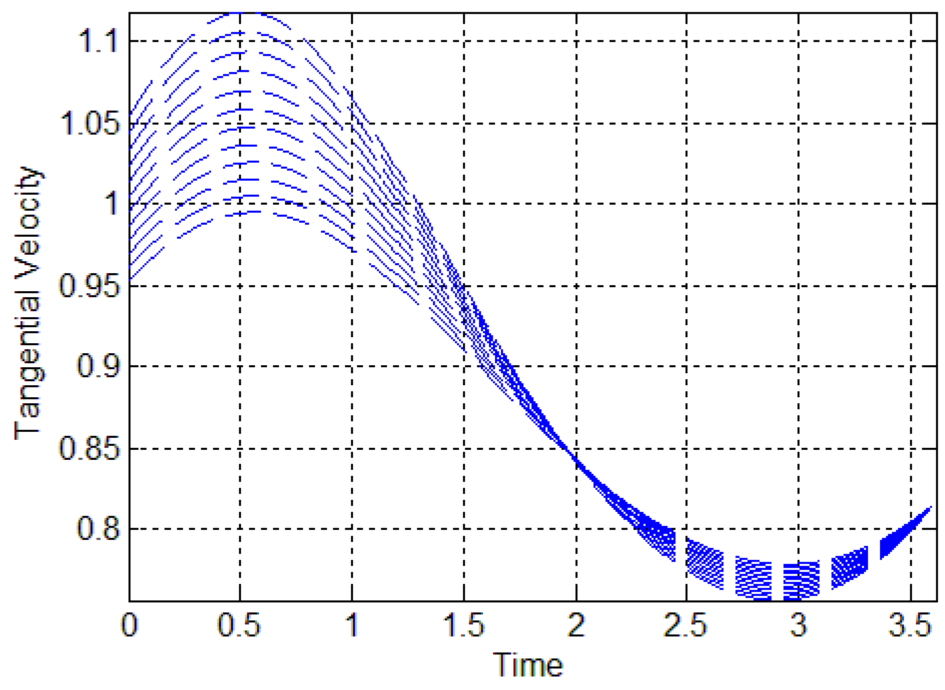

Figure 4.4 Neighboring tangential velocities for the orbit transfer guidance problem.

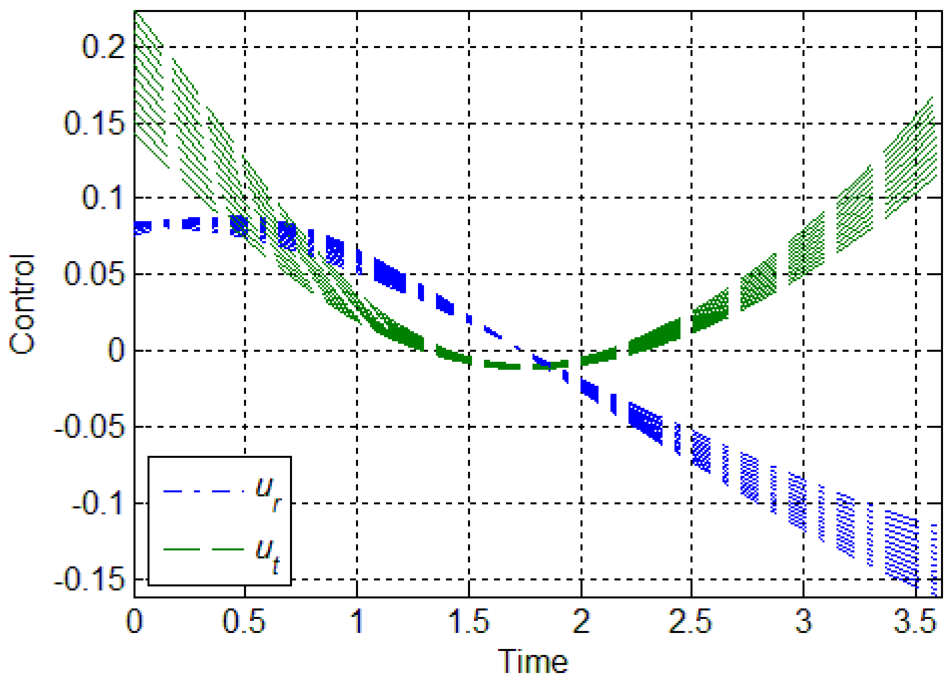

Figure 4.5 Neighboring controls for the orbit transfer guidance problem.

4.4 Re-entry Guidance Problem

This section presents the guidance solution for a three dimensional re-entry problem. Within the guidance framework, sensitivities of the initial costates to changes in the initial states, system parameters, and final states are computed. The motion of the re-entry vehicle is described by the following differential equations [29]:

$$\dot{r} = V sin(\gamma) \tag{4.20}$$

$$\dot{\theta} = \frac{V\cos(\gamma)\cos(\psi)}{r\cos(\phi)} \tag{4.21}$$

$$\dot{\phi} = \frac{V\cos(\gamma)\sin(\psi)}{r} \tag{4.22}$$

$$\dot{V} = -\frac{\mu}{r^2}\sin(\gamma) - D \tag{4.23}$$

$$\dot{\gamma} = -\frac{\mu}{r^2}\frac{cos(\gamma)}{V} + \frac{V}{r}cos(\gamma) + \frac{L}{V}cos(\beta) \tag{4.24}$$

$$\dot{\psi} = -\frac{V}{r}\frac{\cos(\gamma)cos(\psi)\sin(\phi)}{\cos(\phi)} - \frac{L}{V}\frac{sin(\beta)}{cos(\gamma)} \tag{4.25}$$

where r is the radial distance from earth, θ is the longitude, ϕ is the latitude, V is the vehicles velocity, γ is the flight path angle, ψ is the heading angle, and μ is the gravitational constant. The vehicle is controlled with the bank angle, β. Additionally, L and D represent the aerodynamic lift and drag per unit mass given by:

$$D = \frac{1}{2}C_D\rho S^* V^2 \quad , \quad L = \frac{1}{2}C_L\rho S^* V^2 \tag{4.26}$$

where C_D is the drag coefficient, C_L is the lift coefficient , S^* is the reference area per unit mass, and ρ is the density of earth's atmosphere. The vehicles mass is absorbed into the reference area term, and thus doesn't appear explicitly in the equations of motion. An exponential model for the atmospheric density is given as:

$$\rho = \rho_0 e^{-k(r-r_e)} \tag{4.27}$$

where ρ_0 is the density at sea level, k is the scale height of the atmosphere, and r_e is the radius of the earth. The objective of the controller is to minimize both the convective heating rate and the aerodynamically induced acceleration. Therefore, the performance index is a scaled combination of these quantities [29].

$$\mathcal{J} = \int_0^{t_f} \left[(L^2 + D^2)^{1/2} + 20\varepsilon\rho^{1/2} \left(\frac{V}{1000}\right)^3 \right] dt \tag{4.28}$$

where ε is a scaling factor applied to the convective heating rate quantity. The constant parameters used in this simulation [30] are given in Table 4.4 below.

Table 4.4 Simulation parameters for the re-entry problem

Parameter	Symbol	Value	Units
Density at sea level	ρ_0	2.7 x 10^{-3}	slug/ft^3
Atmospheric scale height	k	4.2 x 10^{-5}	1/ft
Gravitational constant	μ	1.4077 x 10^{16}	ft^3/s^2
Lift Coefficient	C_L	0.35	-
Drag Coefficient	C_D	1.3	-
Reference Area	S^*	0.3752	ft^2/slug
Scaling Factor	ε	1.0538 x 10^{-6}	deg

For brevity, the Hamiltonian, Euler-Lagrange equations, and HJB equation will not be shown. The formulation of these equations is straightforward, with the introduction of the system costates: $\Lambda = \left[\lambda_r, \lambda_\theta, \lambda_\phi, \lambda_V, \lambda_\gamma, \lambda_\psi\right]^{\mathrm{T}}$. Although the system isn't affine in the control, a relationship between the control and the costates can still be found through differentiation of the Hamiltonian with respect to the bank angle.

$$-\lambda_\gamma \frac{L}{V} sin(\beta) - \lambda_\psi \frac{L}{V cos(\gamma)} cos(\beta) = 0 \tag{4.29}$$

Rearranging this equation gives an expression for the bank angle in terms of the states and costates.

$$tan(\beta) = -\frac{\lambda_\psi}{\lambda_\gamma cos(\gamma)} \tag{4.30}$$

The boundary conditions for the nominal trajectory can be seen in Table 4.5. As the table shows, only the longitude, latitude, and velocity are terminally constrained. Because of this, three additional terminal conditions for optimality are given as:

$$\lambda_r(t_f) = 0 \;\;,\;\; \lambda_\gamma(t_f) = 0 \;\;,\;\; \lambda_\psi(t_f) = 0 \tag{4.31}$$

Table 4.5 Boundary conditions for the re-entry problem

Variable	Initial	Final	Units
Time	0	390	s
Altitude	400,000	-	ft
Longitude	0	0.33	rad
Latitude	0	-0.025	rad
Velocity	36,000	2,640	ft/s
Flight Path Angle	-6.5	-	deg
Heading Angle	0	-	deg

A nominal optimal solution is obtained from a shooting algorithm performed with the *Matlab* nonlinear equation solver *fsolve*. The shooting method was only able to converge on the optimal solution because it was given accurate guesses for the initial costates. The full optimal re-entry solution is shown in Figure 4.6.

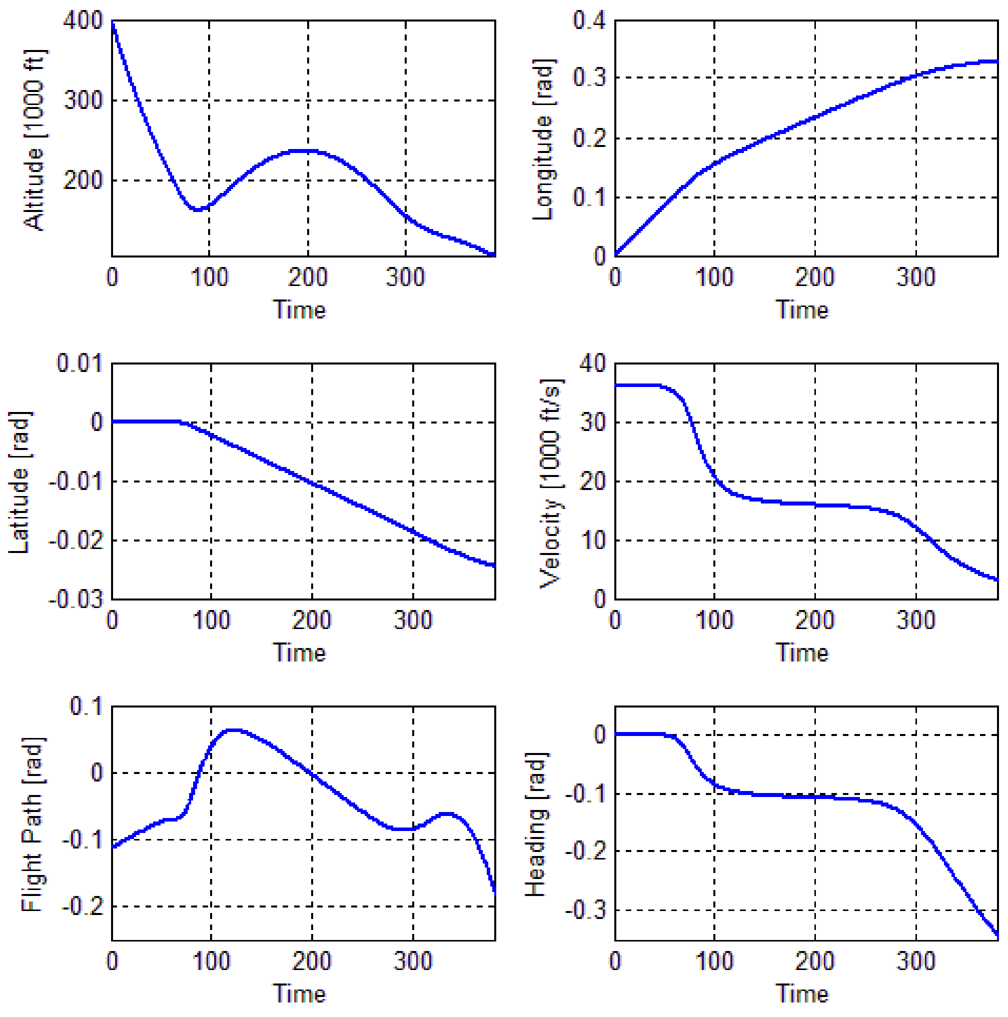

Figure 4.6 Nominal re-entry solution.

4.4.1 Initial State Variations

In this section, sensitivities of the initial costates to changes in the initial altitude and velocity are computed. Due to the size and complexity of the re-entry problem, the PHJB method provides only three orders of control sensitivities. In total, 203 unique control sensitivities are computed: 21 first-order, 56 second-order, and 126 third-order sensitivities. A number of neighboring trajectories are simulated, simultaneously varying the initial altitude ± 8000 ft and the initial velocity ± 2000 ft/s. To show the results in an illustrative manner, the set of initial altitude and velocity variations are chosen to form an ellipse around the nominal initial conditions. Figure 4.7 shows the guidance solutions for these initial state variations. The average error in the final velocity for these neighboring solutions is 14.6 ft/s.

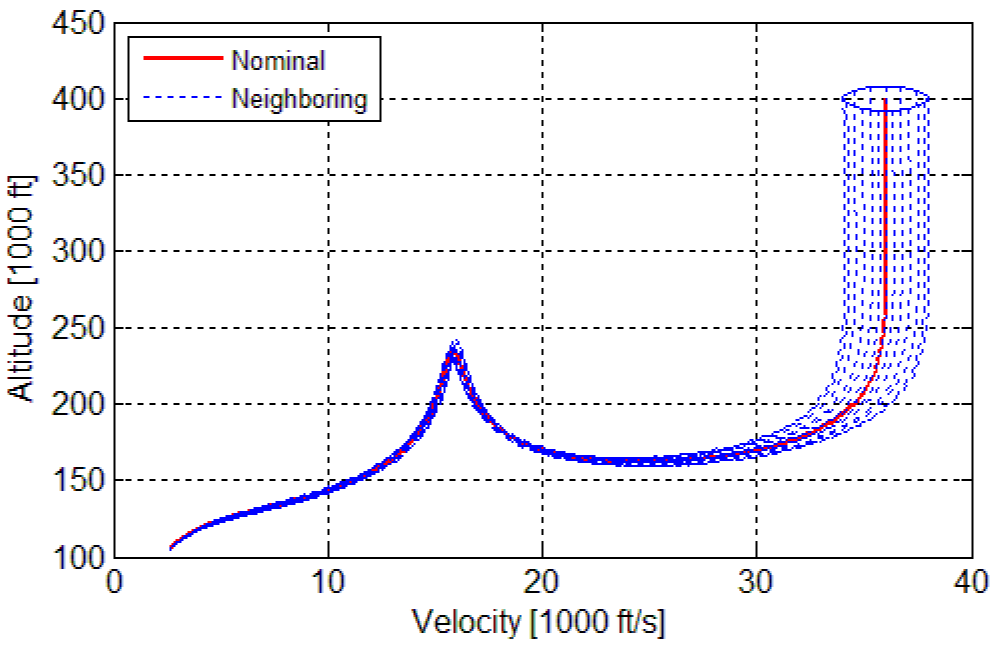

Figure 4.7 Neighboring re-entry profiles for initial altitude and velocity variations.

4.4.2 Parameter Variations

To demonstrate the use of parameter sensitivities, the re-entry guidance problem is solved for costate sensitivities with respect to atmospheric and aerodynamic parameters. Again, the PHJB method generates three orders of costate sensitivities. Considering one parameter variation at a time, a total of 330 unique control sensitivities are computed: 28 first order, 84 second order, and 210 third order sensitivities.

To begin, the effects of changes in the atmospheric density at sea level (ρ_0) and the atmospheric scale height (k) are analyzed. First, the density parameter is varied ± 20 percent of the nominal. Next, the atmospheric scale height is varied ± 5 percent of the nominal. Guidance results for these variations are shown in Table 4.6. The table shows that the error in the final velocity is less than 10 ft/s for these extreme cases. Additionally, neighboring re-entry profiles for variations in ρ_0 and k are shown in Figures 4.8 and 4.9, respectively. In these figures, the terminal constraint appears as a line because the final altitude is free to change.

Table 4.6 Re-entry guidance results for density and scale height variations

		$\delta\rho_0$		$\delta\kappa$	
	Nominal	-20%	+20%	-5%	+5%
$\beta(t_0)$ [deg]	147.0	149.2	145.2	142.2	151.2
$V(t_f)$ [ft/s]	2640.0	2647.9	2646.0	2636.7	2635.8

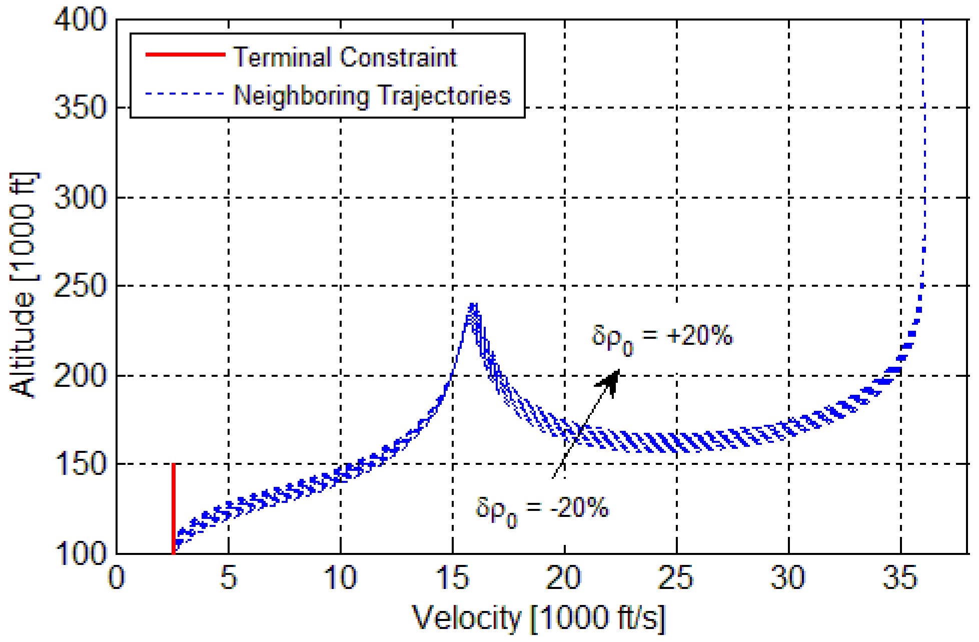

Figure 4.8 Neighboring re-entry profiles for variations in the reference density.

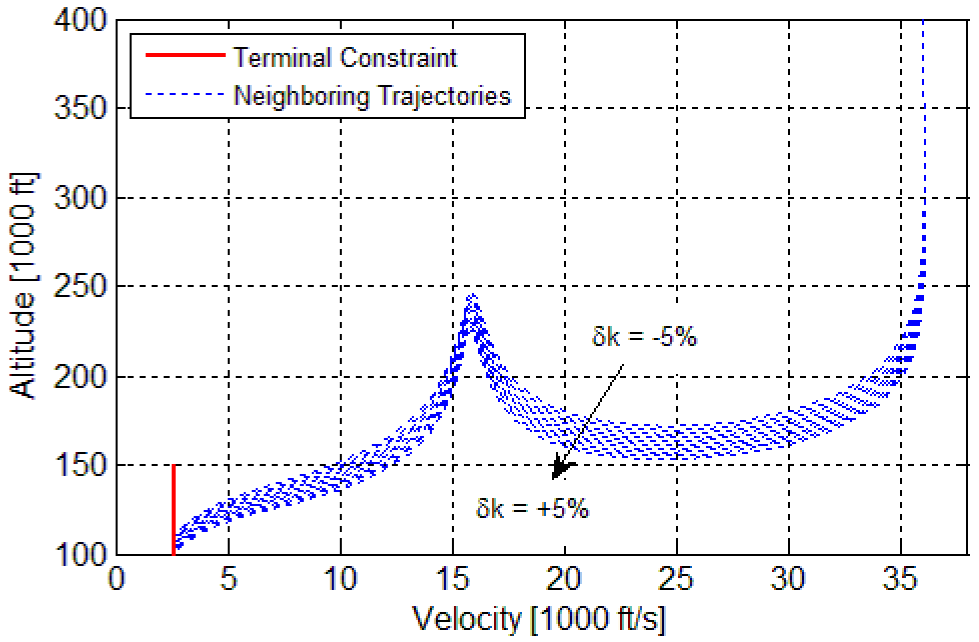

Figure 4.9 Neighboring re-entry profiles for variations in the scale height.

Next, the effects of changes in the lift and drag coefficients are analyzed. Each coefficient is varied ± 10 of the nominal. Guidance results for these variations are shown in Table 4.7. Additionally, neighboring re-entry profiles for variations in C_L and C_D are shown in Figures 4.10 and 4.11, respectively. Again, the PHJB method is able to provide accurate updated initial costates to account for the parametric variations.

Table 4.7 Re-entry guidance results for lift and drag coefficient variations

		δC_L		δC_D	
	Nominal	-10%	+10%	-10%	+10%
$\beta(t_0)$ [deg]	147.0	138.5	151.9	154.7	135.4
$V(t_f)$ [ft/s]	2640.0	2623.8	2623.0	2640.8	2633.6

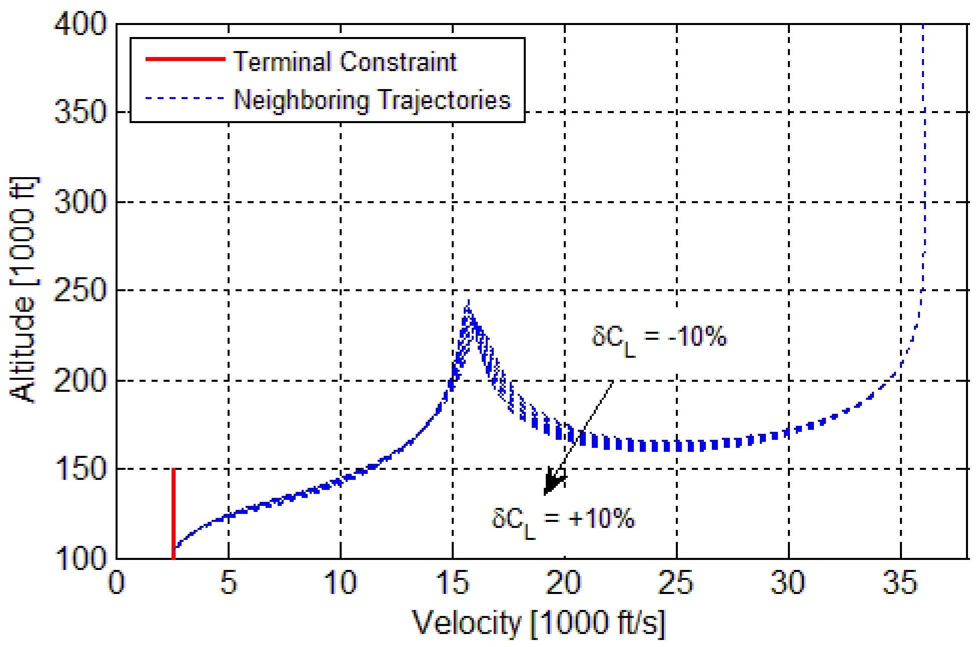

Figure 4.10 Neighboring re-entry profiles for variations in the lift coefficient.

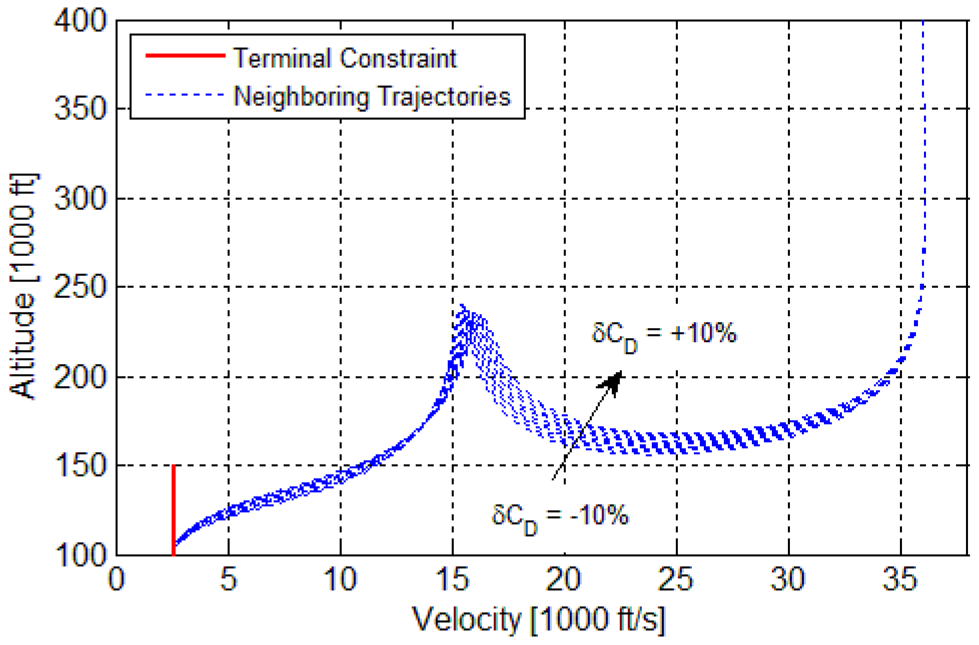

Figure 4.11 Neighboring re-entry profiles for variations in the drag coefficient.

4.4.3 Final State Variations

In this section, sensitivities of the initial costates to changes in the final velocity are computed. Again, the PHJB method generates three orders of costate sensitivities. Six neighboring trajectories are generated with the following desired final velocities:

$$V_f = [1.5,\ \ 2.0,\ \ 2.5,\ \ 3.0,\ \ 3.5\ ,\ \ 4.0] \qquad (4.32)$$

where the final velocities are given in 1000 ft/s. Figure 4.12 shows the neighboring trajectories generated using the PHJB method. To observe the effectiveness of the guidance results, Figure 4.13 shows a blown up view of the results near the final time. The six neighboring trajectories have an average final velocity error of 12.6 ft/s.

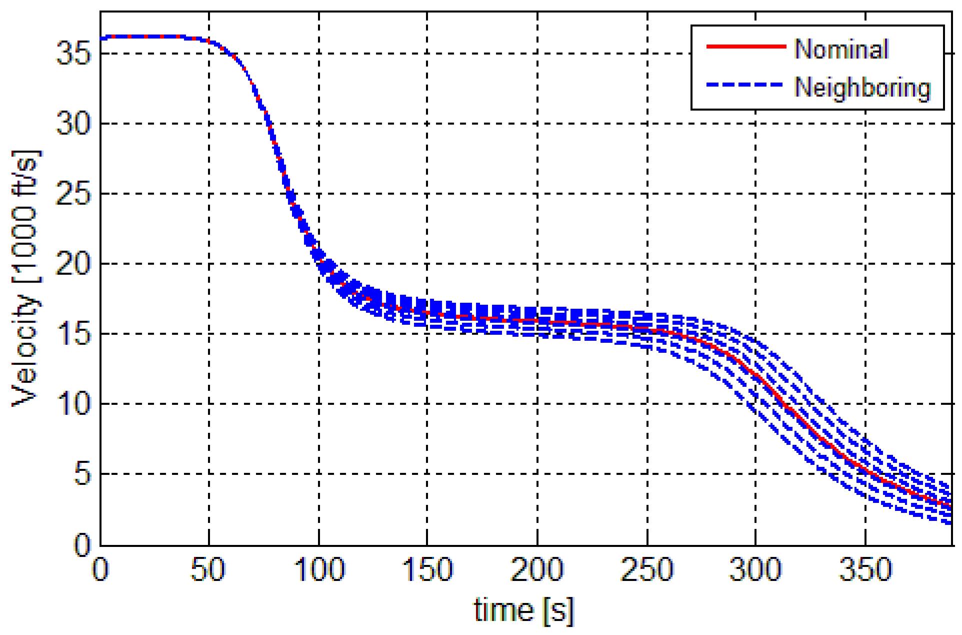

Figure 4.12 Nominal and neighboring trajectories for final velocity variations.

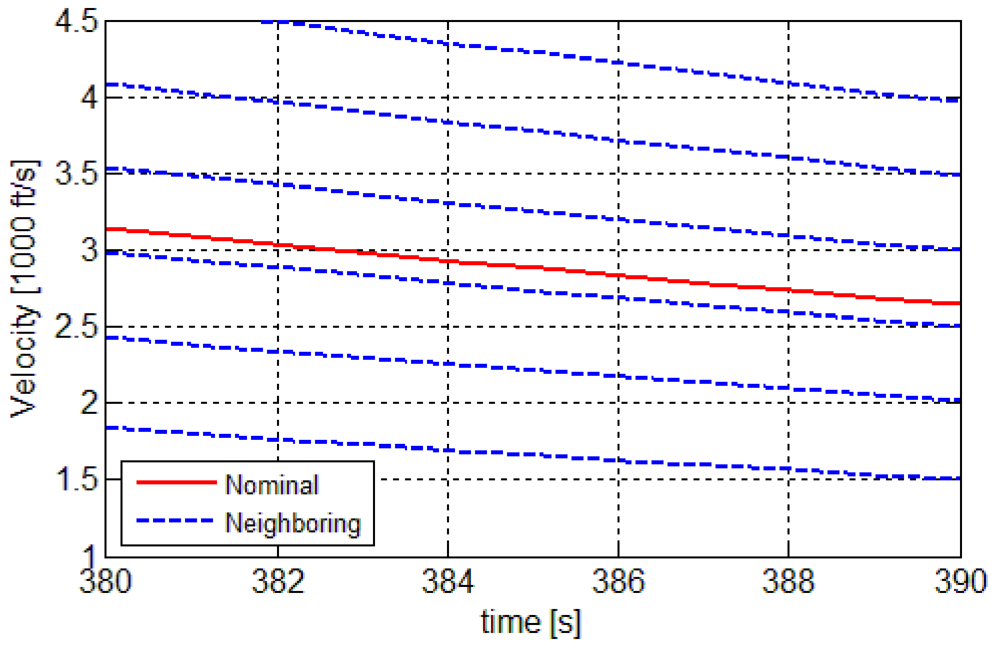

Figure 4.13 A blown up view of near final time nominal and neighboring trajectories for final velocity variations.

5. SUMMARY AND CONCLUSIONS

This thesis presented a non-iterative method for solving finite-horizon optimal control problems involving nonlinear dynamical systems. Analytical partial differentiation of the Hamilton-Jacobi-Bellman equation with respect to the states led to an approximate solution of the cost-to-go function and its associated sensitivities. First order cost-to-go sensitivities provided the nominal open-loop solution to the optimal control problem, while higher order sensitivities provided a means of generating a family of extremals about the nominal trajectory. The method was extended to handle terminally constrained problems, as well as, free final time problems.

The approximation method was shown to be well suited for moderately nonlinear problems with soft terminal constraints. Extremely high orders of approximation were shown to be needed to accurately solve highly nonlinear problems. The addition of hard terminal constraints was shown to be unfavorable because it imposed the need for an iterative solution process and introduced a second source of approximation error.

Next, an alternative approach for generating a family of extremals was presented. This new approach approximated the solution to the Hamilton-Jacobi-Bellman equation about the nominal solution. As a result, the nonlinearity of the problem was greatly reduced, which allowed for accurate solutions with much lower orders of approximation. For comparison, the Lagrange Implicit Function Theorem was applied as another means of generating a family of extremals. The two approaches were found to provide analogous results when applied at equivalent orders. However, it was shown that the

Hamilton-Jacobi-Bellman equation approximation method was more efficient to perform, allowing for higher orders of control sensitivities.

REFERENCES

[1] Betts, John T., *Practical Methods for Optimal Control using Nonlinear Programming*, SIAM, Philadelphia, 2001, pp. 162–165.

[2] Fahroo, F., and Ross, I.M., “Direct Trajectory Optimization by a Chebyshev Pseudospectral Method,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, January–February 2002, pp. 160–166.

[3] Singh, B., Bhattacharya, R., and Vadali, S. R., “Verification of Optimality and Costate Estimation using Hilbert Space Projection,” *AIAA Journal of Guidance, Control, and Dynamics* , Vol. 32, No. 4, 2009, pp. 1345-1344.

[4] Bryson, A.E., and Ho, Y.-C., *Applied Optimal Control*, Hemisphere, Washington D.C., 1975, pp. 47–50 and 128–136.

[5] Bellman, R. E., and Dreyfus, S. E., *Applied Dynamic Programming,* Princeton University Press, Princeton, NJ, 1962, pp. 180–204.

[6] Speyer, J.L., and Crues, E.Z., “Approximate Optimal Atmospheric Guidance Law for Aeroassisted Plane-Change Maneuvers,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 5, 1990, pp. 792–802.

[7] Xin, M., Balakrishnan, S.N., Stansbery, D.T., and Ohlmeyer, E.J., "Nonlinear Missile Autopilot Design with Theta-D Technique," *AIAA Journal of Guidance, Control and Dynamics*, Vol. 27, No. 3, 2004, pp. 406–417.

[8] Vadali, S.R., and Sharma, R., “Optimal Finite-Time Feedback Controllers for Nonlinear Systems with Terminal Constraints,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 29, No. 4, 2006, pp. 921–928.

[9] Park, C., and Scheeres, D. J., “Solutions of Optimal Feedback Control Problem with General Boundary Conditions using Hamiltonian Dynamics and Generating Functions,” *Automatica*, Vol. 42, 2006, pp. 869–875.

[10] Beard, R., Saridis, G., and Wen, J., “Galerkin Approximation of the Generalized Hamilton-Jacobi-Bellman Equation,” *Automatica*, Vol. 33, 1997, pp. 2159–2177.

[11] Huang, C.-S., Wang, S., Chen, C. S., and Li, Z.-C., “A Radial Basis Collocation Method for Hamilton-Jacobi-Bellman Equations,” *Automatica*, Vol. 42, 2006, pp. 2201–2207.

[12] Cheng, T., Lewis, F. L., and Murad, A., "A Neural Network Solution for Fixed-Final Time Optimal Control of Nonlinear Systems," *Automatica*, Vol. 43, 2007, pp. 482–490.

[13] Osher, S., and Fedkiw, R., *Level Set Methods and Dynamic Implicit Surfaces*, Springer-Verlag, New York, 2003, pp. 47–59.

[14] Kumar, M., Chakravorty, S., and Junkins, J. L., "Computational Nonlinear Control," *AIAA Journal of Guidance, Control and Dynamics*, Vol. 32, No. 3, 2009, pp. 1050–1055.

[15] Richardson, S., and Wang, S., "Numerical Solution of Hamilton-Jacobi-Bellman Equations by an Exponentially Fitted Finite Volume Method 1," *Optimization*, Vol. 55, Nos. 1&2, Feb 2006, pp. 121–140.

[16] Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, Vol. 36, 2000, pp. 789–814.

[17] Junkins, J.L., Turner, J.D., and Majji, M., "Generalizations and Applications of the Lagrange Implicit Function Theorem," *The F. Landis Markley Astronautics Symposium of the American-Astronautical-Society*, AAS 08-302, Cambridge, MD, 2008, pp. 723–744.

[18] Di Lizia, P., Armellin, R., Ercoli-Finzi, A., and Berz, M., "High-order Robust Guidance of Interplanetary Trajectories Based on Differential Algebra," *Journal of Aerospace Engineering, Sciences and Applications*, Vol. 1, No. 1, 2008, pp. 43–57.

[19] Lewis, F.L. and Syrmos, V.L., *Optimal Control*, 2 ed., John Wiley & Sons, New York, 1995, pp. 131–135.

[20] Griffith, D.T., Turner, J.D., Vadali, S.R., and Junkins, J.L., "Higher Order Sensitivities for Solving Nonlinear Two-Point Boundary Value Problems," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, AIAA-2004-5404, Providence, RI, 2004.

[21] MapleSoft, "*Maple* Toolbox for *MATLAB*," http://www.maplesoft.com/products/MapleMATLAB/, 2010.

[22] MathWorks, "Product Support: MEX Files Guide," http://www.mathworks.com/support/tech-notes/1600/1605.html, 2010.

[23] Turner, J.D., "Automated Generation of High-Order Partial Derivative Models," *AIAA Journal*, Vol. 41, No. 8, August 2003, pp. 1590–1599.

[24] Berz, M., and Makino, K., “COSY Infinity Version 9,” Technical Report MSUHEP 06083, Michigan State University, East Lansing, MI, 2006.

[25] Sharma, R., “A Series Solution Framework for Finite-Time Optimal Feedback Control, H-Infinity Control and Games,” Ph.D. Thesis, Texas A&M University, College Station, 2008.

[26] Schaub, H., and Junkins, J.L, *Analytical Mechanics of Space Systems*, American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 2003, pp. 77.

[27] McCrate, C. M., and Vadali, S. R, “Simultaneous Computation of Optimal Controls and Their Sensitivities,” *AAS/AIAA Astrodynamics Specialist Conference*, Pittsburgh, PA, AAS 09-406, 2009.

[28] Di Lizia, P., “Robust Space Trajectory and Space System Design Using Differential Algebra,” Ph.D. Thesis, Politecnico di Milano, 2008.

[29] Tapley, B.,D., and Williamson, W. E., “Comparison of Linear and Riccati Equations Used to Solve Optimal Control Problems,” *AIAA Journal*, Vol. 10, No. 9, 1972, pp. 1154–1159.

[30] Williamson, W. E., “Optimal Three Dimensional Re-entry Trajectories for Apollo Type Vehicles,” AMRL Rept. 1013, University of Texas, Austin, TX, 1970.

APPENDIX A

In this appendix, a derivation of the Hamilton-Jacobi-Bellman equation and the costate differential equation from the cost-to-go function is presented. To begin, assume the cost-to-go function takes the form:

$$J(x,\nu,t) = \phi\left[x\left(t_f\right)\right] + \nu^T\psi\left[x\left(t_f\right)\right] + \int_t^{t_f} L(x,u,t) \qquad \text{(A.1)}$$

For this derivation, the relationship $J_x = \lambda$ is assumed. As Equation (A.1) shows, the cost-to-go is a function of the states, terminal constraint Lagrange multipliers, and time. To construct the HJB equation, the total time derivative of Equation (A.1) is taken. On the left hand side of the equation, the total time derivative is separated into partial derivatives with respect to the time, states, and terminal constraint Lagrange multipliers. On the right hand side of the equation, the second fundamental theorem of calculus is applied to simplify the expression. The resulting equation is given as:

$$\frac{\partial J}{\partial t} + J_x\frac{dx}{dt} + J_\nu\frac{d\nu}{dt} = -L(x,u,t) \qquad \text{(A.2)}$$

Because ν is constant along the trajectory, $\frac{d\nu}{dt}$ will be equal to zero and will drop out of the equation. Thus, the form of the HJB equation is the same with or without an applied terminal constraint. The same logic holds when including a dependence on some constant system parameter as well. Next, Equation (A.2) is rearranged give:

$$-\frac{\partial J}{\partial t} = L(x,u,t) + J_x\, f(x,u,t) \qquad \text{(A.3)}$$

Utilizing the relationship $J_x = \lambda$, the right hand side of Equation (A.3) is equivalent to the Hamiltonian, thus giving the recognizable form of the HJB equation:

$$-\frac{\partial J}{\partial t} = \min_u H(x, u, J_x, t) \tag{A.4}$$

$$-\frac{\partial J}{\partial t} = H(x, J_x, t) \tag{A.5}$$

Since we have assumed the relationship $J_x = \lambda$, the costate differential equation can be derived from the HJB equation in the following manner.

$$\dot{\lambda} = \frac{d}{dt}(J_x) = J_{xx}\frac{dx}{dt} + \frac{\partial J_x}{\partial t} \tag{A.6}$$

First, $\frac{\partial J_x}{\partial t}$ is derived by taking the partial derivative of the HJB equation with respect to x, as shown below.

$$\frac{\partial J_x}{\partial t} = -L_x - J_x f_x - J_{xx}\frac{dx}{dt} \tag{A.7}$$

Substituting this expression into Equation (A.6) gives:

$$\dot{\lambda} = -L_x - J_x f_x \tag{A.8}$$

Finally, relating the right hand side of Equation (A.8) to the Hamiltonian, the costate differential equation is written in the recognizable form:

$$\dot{\lambda} = -\frac{\partial H}{\partial x} \tag{A.9}$$

APPENDIX B

In this appendix, a 4th order reversion of series procedure is described. The series reversion process is applied to the following Taylor series expansion:

$$\delta J_\nu = J_{\nu\nu}\ \delta\nu + \frac{1}{2!} J_{\nu\nu\nu}\ \delta\nu^2 + \frac{1}{3!} J_{\nu\nu\nu\nu}\ \delta\nu^3 + \frac{1}{4!} J_{\nu\nu\nu\nu\nu}\ \delta\nu^4 \qquad \text{(B.1)}$$

where:

$$\delta J_\nu = J_\nu^* - J_\nu = -J_\nu \qquad \text{(B.2)}$$

The aim of the reversion process is to obtain the proper perturbed terminal constraint Lagrange multiplier (δν) as a function of the partial derivatives of the cost-to-go function ($J_\nu, J_{\nu\nu}, J_{\nu\nu\nu}, J_{\nu\nu\nu\nu}, J_{\nu\nu\nu\nu\nu}$).

This is accomplished by solving expanding J_ν in 1st through 4th order Taylor series expansions, as shown below.

$$\delta J_\nu = J_{\nu\nu}\ \delta\nu \qquad \text{(B.3)}$$

$$\delta J_\nu = J_{\nu\nu}\ \delta\nu + \frac{1}{2!} J_{\nu\nu\nu}\ \delta\nu^2 \qquad \text{(B.4)}$$

$$\delta J_\nu = J_{\nu\nu}\ \delta\nu + \frac{1}{2!} J_{\nu\nu\nu}\ \delta\nu^2 + \frac{1}{3!} J_{\nu\nu\nu\nu}\ \delta\nu^3 \qquad \text{(B.5)}$$

$$\delta J_\nu = J_{\nu\nu}\ \delta\nu + \frac{1}{2!} J_{\nu\nu\nu}\ \delta\nu^2 + \frac{1}{3!} J_{\nu\nu\nu\nu}\ \delta\nu^3 + \frac{1}{4!} J_{\nu\nu\nu\nu\nu}\ \delta\nu^4 \qquad \text{(B.6)}$$

Each of these equations is inverted to solve for the first-order δν term. The equations are solved in ascending Taylor series expansion order, because the solution to each equation utilizes the values of δν obtained by the previous equation. To help illustrate this, a subscript will be given to each δν which corresponds to the order of Taylor series expansion considered. With this in mind, the update equations are given as:

$$\delta\nu_1 = [J_{\nu\nu}]^{-1}\,[\delta J_\nu] \tag{B.7}$$

$$\delta\nu_2 = [J_{\nu\nu}]^{-1}\left[\delta J_\nu - \tfrac{1}{2!}J_{\nu\nu\nu}\ (\delta\nu_1)^2\right] \tag{B.8}$$

$$\delta\nu_3 = [J_{\nu\nu}]^{-1}\left[\delta J_\nu - \tfrac{1}{2!}J_{\nu\nu\nu}\ (\delta\nu_2)^2 - \tfrac{1}{3!}J_{\nu\nu\nu\nu}\ (\delta\nu_2)^3\right] \tag{B.9}$$

$$\delta\nu_4 = [J_{\nu\nu}]^{-1}\left[\delta J_\nu - \tfrac{1}{2!}J_{\nu\nu\nu}\ (\delta\nu_3)^2 - \tfrac{1}{3!}J_{\nu\nu\nu\nu}\ (\delta\nu_3)^3 - \tfrac{1}{4!}J_{\nu\nu\nu\nu\nu}\ (\delta\nu_3)^4\right] \tag{B.10}$$

The accuracy of $\delta\nu$ increases with each additional order of Taylor series expansion. Therefore, $\delta\nu_4$ is used to update the terminal constraint Lagrange multipliers.

To implement these equations in *Matlab*, the partial derivatives of the cost-to-go function must be left in their natural structure as tensors. A procedure for dealing with these tensors is developed in Appendix E.

APPENDIX C

This appendix provides a description of the properties of multivariable partial derivatives, and how they are exploited for efficiency. For problems formulated with state-space dimensions of two or more, the orders of partial derivatives of the cost-to-go assume the following structure: 1st order partial derivatives are vectors, 2nd order partial derivatives are matrices, and 3rd through *n*th order partial derivatives are tensors containing 3 through *n* indices, respectively.

In multivariable calculus, the order of differentiation has no effect on the partial derivatives, i.e., the partial derivatives possess the commutative property. For example, given an arbitrary number of states, the commutative property is represented on 2nd and 3rd order partial derivatives as:

$$J_{x_i x_j} = J_{x_j x_i} \tag{C.1}$$

$$J_{x_i x_j x_k} = J_{x_j x_i x_k} = J_{x_i x_k x_j} = J_{x_j x_k x_i} = J_{x_k x_i x_j} = J_{x_k x_j x_i} \tag{C.2}$$

This creates symmetry within the structure of the multivariable partial derivatives, starting at the 2nd order. With each additional order of derivation, this symmetry incorporates more and more terms. With this in mind, a scheme is devised to exploit the symmetry of these partial derivatives, which dramatically reduces the computational effort and storage requirements necessary to implement the HJB approximation method.

The devised scheme relies on treating each independent variable separately, thus disassembling the overall structure of the partial derivatives. As individual terms, conditions are enforced to ensure no repetitive partial derivatives are calculated.

Specifically, each independent variable is given an index number, and partial differentiation is only allowed to occur in ascending order, i.e.,

$$J_{x_i x_j} \ , \ \forall \ \ i \leq j \tag{C.3}$$

$$J_{x_i x_j x_k} \ , \ \forall \ \ i \leq j \leq k \tag{C.4}$$

$$J_{x_i x_j x_k x_l} \ , \ \forall \ \ i \leq j \leq k \leq l \tag{C.5}$$

where i, j, k, l are the index numbers of the variables. By only allowing differentiation in ascending index order, all of the repetitive terms are eliminated. Once the partial derivatives have been derived, they are concatenated in into a single column, again in ascending order.

$$PD = \left[J_{x_i} \, , J_{x_j} \ \ldots , J_{x_i x_i} \, , J_{x_i x_j} \, , \ldots , J_{x_i x_i x_i} \, , J_{x_i x_i x_j} \, , \ldots \right]^{\mathrm{T}} \tag{C.6}$$

Within the symbolic differentiation routine, the PDEs governing the partial derivatives are differentiated and stacked in the same manner. Thus, the unique partial derivatives are passed to and from the numerical integration routine (*ode*45) using this structure.

APPENDIX D

In this appendix, a derivation of the perturbed Hamilton-Jacobi-Bellman (PHJB) equation is presented. To begin assume some initial state perturbations $\delta x(t_0)$. These initial perturbations will generate perturbations $\delta x(t)$ governed by:

$$\delta \dot{x} = f(x^* + \delta x, u^* + \delta u) - f^* \tag{D.1}$$

where the $*$ superscript indicates the nominal values. Next, assuming the problem is formulated with soft terminal constraints, the perturbed cost-to-go function is given by the following expression.

$$\begin{aligned}\delta J = \phi\left[x^*(t_f) + \delta x(t_f)\right] - \phi^* \\ + \int_t^{t_f} \{H(x^* + \delta x, u^* + \delta u, \lambda^*) - H^* - H_x^* \delta x - H_u^* \delta u\} dt\end{aligned} \tag{D.2}$$

To construct the PHJB equation, the total time derivative of Equation (D.2) is taken. On the left hand side of the equation, the total time derivative is separated into partial derivatives with respect to the time and perturbed states. On the right hand side of the equation, the second fundamental theorem of calculus is applied to simplify the expression. The resulting equation is given as:

$$\frac{\partial \delta J}{\partial t} + \frac{\partial \delta J}{\partial \delta x}^{\mathrm{T}} \delta \dot{x} = -\{H(x^* + \delta x, u^* + \delta u, \lambda^*) - H^* - H_x^* \delta x - H_u^* \delta u\} \tag{D.3}$$

To simplify this equation, the following relationships are exploited:

$$\delta J_{\delta x} = \delta \lambda \quad ; \quad H_u^* = 0 \quad ; \quad \delta \dot{x} = f(x^* + \delta x, u^* + \delta u) - f^* \, ; \tag{D.4}$$

$$H(x^* + \delta x, u^* + \delta u, \lambda^* + \delta \lambda) = H(x^* + \delta x, u^* + \delta u, \lambda^*) + \delta \lambda^{\mathrm{T}} f(x^* + \delta x, u^* + \delta u)$$

Making use of these relationships, Equation (D.3) is reduced to:

$$-\frac{\partial \delta J}{\partial t} = H(x^* + \delta x, u^* + \delta u, \lambda^* + \delta \lambda) - H^* - H_x^* \delta x - H_\lambda^* \delta \lambda \tag{D.5}$$

From Equation (D.4), the perturbed Hamiltonian is defined as:

$$\delta H = H(x^* + \delta x, u^* + \delta u, \lambda^* + \delta\lambda) - H^* - H_x^*\delta x - H_\lambda^*\delta\lambda \qquad \text{(D.6)}$$

A second order necessary condition for optimality is given as:

$$\frac{\partial \delta H}{\partial \delta u} = 0 \qquad \text{(D.7)}$$

For systems affine in the control, Equation (D.7) will provide a means of expressing the control terms as a function of the costates. The control terms are removed and the final form of the PHJB equation is given as:

$$-\frac{\partial \delta J}{\partial t} = H(x^* + \delta x, \lambda^* + \delta\lambda) - H^* - H_x^*\delta x - H_\lambda^*\delta\lambda \qquad \text{(D.8)}$$

Since we have assumed the relationship $\delta J_{\delta x} = \delta\lambda$, the perturbed costate differential equation can be derived from the PHJB equation in the following manner.

$$\delta\dot{\lambda} = \frac{d}{dt}(\,\delta J_{\delta x}) = \frac{\partial}{\partial t}(\delta J_{\delta x}) + \frac{\partial \delta\lambda}{\partial \delta x}\delta\dot{x} \qquad \text{(D.9)}$$

First, $\frac{\partial}{\partial t}(\delta J_{\delta x})$ is derived from the PHJB equation, as shown below.

$$\frac{\partial}{\partial t}(\delta J_{\delta x}) = -\left\{H_x(x^* + \delta x, \lambda^* + \delta\lambda) - H_x^* + \frac{\partial \delta\lambda}{\partial \delta x}(H_\lambda - H_\lambda^*)\right\} \qquad \text{(D.10)}$$

Recognizing that $\delta\dot{x} = H_\lambda - H_\lambda^*$, Equation (D.9) can be substituted back into Equation (D.8) revealing the following equation:

$$\delta\dot{\lambda} = -H_x(x^* + \delta x, \lambda^* + \delta\lambda) + H_x^* \qquad \text{(D.11)}$$

Finally, it is worth noting the connection between δH, $\delta\dot{x}$, and $\delta\dot{\lambda}$:

$$\delta\dot{x} = \frac{\partial \delta H}{\partial \delta\lambda} \quad , \quad \delta\dot{\lambda} = -\frac{\partial \delta H}{\partial \delta x} \qquad \text{(D.12)}$$

APPENDIX E

In this appendix, the implementation of the Lagrange Implicit Function Theorem is described up to the 4th order. In this thesis, the implicit function theorem is used in two cases:

1) To calculate the sensitivities of the terminal constraint Lagrange multipliers with respect to the states.
2) To calculate the sensitivities of the initial costates with respect to the initial states.

In both situations, the implicit function theorem is utilized the same manner. However, the constraint equation used in each case is different. These constraint equations are given for each case as:

1) $J_\nu(x, \nu(x)) = 0$
2) $\psi(x, \lambda(x)) = 0$

To avoid unnecessary repetition, the application of the implicit function theorem is demonstrated only once. To begin, assume the following general constraint equation:

$$M(x, \lambda(x)) = 0 \tag{E.1}$$

The first four total derivatives of M with respect to x are taken and split into their respective partial derivatives as follows:

$$\frac{d}{dx}M \;= M_x + M_\lambda\left(\frac{d\lambda}{dx}\right) \tag{E.2}$$

$$\frac{d^2}{dx^2}M = M_{xx} + 2M_{\lambda x}\left(\frac{d\lambda}{dx}\right) + M_{\lambda\lambda}\left(\frac{d\lambda}{dx}\right)^2 + M_\lambda\left(\frac{d^2\lambda}{dx^2}\right) \tag{E.3}$$

$$\frac{d^3}{dx^3}M = M_{xxx} + 3M_{\lambda xx}\left(\frac{d\lambda}{dx}\right) + 3M_{\lambda\lambda x}\left(\frac{d\lambda}{dx}\right)^2 + M_{\lambda\lambda\lambda}\left(\frac{d\lambda}{dx}\right)^3 + 3M_{\lambda x}\left(\frac{d^2\lambda}{dx^2}\right) + 3M_{\lambda\lambda}\left(\frac{d\lambda}{dx}\right)\left(\frac{d^2\lambda}{dx^2}\right) + M_{\lambda}\left(\frac{d^3\lambda}{dx^3}\right) \tag{E.4}$$

$$\frac{d^4}{dx^4}M = M_{xxxx} + 4M_{\lambda xxx}\left(\frac{d\lambda}{dx}\right) + 6M_{\lambda\lambda xx}\left(\frac{d\lambda}{dx}\right)^2 + 4M_{\lambda\lambda\lambda x}\left(\frac{d\lambda}{dx}\right)^3 + M_{\lambda\lambda\lambda\lambda}\left(\frac{d\lambda}{dx}\right)^4 + 6M_{\lambda xx}\left(\frac{d^2\lambda}{dx^2}\right) + 12M_{\lambda\lambda x}\left(\frac{d\lambda}{dx}\right)\left(\frac{d^2\lambda}{dx^2}\right) + 6M_{\lambda\lambda\lambda}\left(\frac{d\lambda}{dx}\right)^2\left(\frac{d^2\lambda}{dx^2}\right) + 4M_{\lambda\lambda}\left(\frac{d\lambda}{dx}\right)\left(\frac{d^3\lambda}{dx^3}\right) + 4M_{\lambda x}\left(\frac{d^3\lambda}{dx^3}\right) + 3M_{\lambda\lambda}\left(\frac{d^2\lambda}{dx^2}\right)^2 + M_{\lambda}\left(\frac{d^4\lambda}{dx^4}\right) \tag{E.5}$$

These four equations are easily inverted to solve for the first four orders of sensitivities of λ with respect to x. However, many of the partial derivative terms in these equations are naturally represented as tensors.

To handle these equations in *Matlab*, the tensors are represented as matrices through a stacking operation. The tensors are stacked using a built-in concatenation function (*cat.m*). To illustrate the stacking procedure, assume the following:

$$M = [A, B]^{\mathrm{T}} \quad , \quad x = [x_1, x_2]^{\mathrm{T}} \quad , \quad \lambda = [\lambda_1, \lambda_2]^{\mathrm{T}} \tag{E.6}$$

From this, $M_{\lambda x} \in \Re^{2\times2\times2}$ is given as:

$$M_{\lambda x}(:,:,1) = [M1] = \begin{bmatrix} A_{x_1\lambda_1} & A_{x_2\lambda_1} \\ B_{x_1\lambda_1} & B_{x_2\lambda_1} \end{bmatrix} \tag{E.7}$$

$$M_{\lambda x}(:,:,2) = [M2] = \begin{bmatrix} A_{x_1\lambda_2} & A_{x_2\lambda_2} \\ B_{x_1\lambda_2} & B_{x_2\lambda_2} \end{bmatrix} \tag{E.8}$$

The stacked version of this tensor is formed with the command: $cat(2, M1, M2)$. The resulting matrix takes the form[§]:

$$M_{\lambda x}^{S} = [\,[M1] \quad [M2]\,] = \begin{bmatrix} A_{x_1\lambda_1} & A_{x_2\lambda_1} & A_{x_1\lambda_2} & A_{x_2\lambda_2} \\ B_{x_1\lambda_1} & B_{x_2\lambda_1} & B_{x_1\lambda_2} & B_{x_2\lambda_2} \end{bmatrix} \tag{E.9}$$

[§] The superscript S denotes a stacked tensor

In Equation (E.5), $M_{\lambda x}$ is multiplied with $\frac{d\lambda}{dx} \in \Re^{2\times 2}$. Before this multiplication can take place using the stacked tensor, the structure of $\frac{d\lambda}{dx}$ must be altered using a Kronecker product, as follows:

$$I_2 \otimes \frac{d\lambda}{dx} = \begin{bmatrix} \frac{d\lambda}{dx} & 0 \\ 0 & \frac{d\lambda}{dx} \end{bmatrix} \tag{E.10}$$

where $I_2 \in \Re^{2\times 2}$ is an identity matrix. Using this notation, the equivalence between the stacked and non-stacked tensor multiplications is given as:

$$\left(M_{\lambda x} \frac{d\lambda}{dx}\right)^S = M_{\lambda x}^S \left(I_2 \otimes \frac{d\lambda}{dx}\right) \tag{E.11}$$

Close attention must be paid to the order in which x and λ are differentiated, e.g., $M_{\lambda x}^S \neq M_{x\lambda}^S$. The difference between these stacked tensors is in the placement of individual terms. The manner in which the Kronecker product is applied can be used to compensate for this dissimilar structure, e.g.,

$$M_{\lambda x}^S \left(I_2 \otimes \frac{d\lambda}{dx}\right) = M_{x\lambda}^S \left(\frac{d\lambda}{dx} \otimes I_2\right) \tag{E.12}$$

Similarly, $M_{\lambda\lambda}^S$ is multiplied with $\frac{d\lambda}{dx}^2$ in the following manner.

$$\left(M_{\lambda\lambda} \frac{d\lambda}{dx}^2\right)^S = M_{\lambda\lambda}^S \left(\frac{d\lambda}{dx} \otimes \frac{d\lambda}{dx}\right) \tag{E.13}$$

As the rank of the tensors increases, additional Kronecker products are used as follows:

$$\left(M_{\lambda xx} \frac{d\lambda}{dx}\right)^S = M_{\lambda xx}^S \left[I_2 \otimes \left(I_2 \otimes \frac{d\lambda}{dx}\right)\right] \tag{E.14}$$

$$\left(M_{\lambda xxx} \frac{d\lambda}{dx}\right)^S = M_{\lambda xxx}^S \left\{I_2 \otimes \left[I_2 \otimes \left(I_2 \otimes \frac{d\lambda}{dx}\right)\right]\right\} \tag{E.15}$$

VITA

Name:	Christopher Mathew McCrate
Address:	Texas A&M University Department of Aerospace Engineering H.R. Bright Building, Ross Street – TAMU 3141 College Station, TX 77843-3141
Email Address:	chris.mccrate@gmail.com
Education:	B.S., Mechanical Engineering, The University of Missouri, 2008 M.S., Aerospace Engineering, Texas A&M University, 2010