AFRL-RW-EG-TR-2018-058

Solution of Optimal Control Problem for High-Speed Ascent and Reentry Flight

Anil V. Rao and William W. Hager

University of Florida
Office of Engineering Research
339 Well Hall
Gainesville, FL 32611

May 2018

Final Report

AIR FORCE RESEARCH LABORATORY
MUNITIONS DIRECTORATE
WEAPON ENGAGEMENT DIVISION
WEAPON DYNAMICS & CONTROLS BRANCH
EGLIN AIR FORCE BASE, FL 32542
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

AFRL-RW-EG-TR-2018-058 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

//SIGNED//

SHARON STOCKBRIDGE, DR-III
AFRL/RWWN Program Manager

//SIGNED//

ROBERT MURPHEY, DR-IV, PhD
Munitions Aerodynamics, Guidance, Navigation, and Control CTC Lead

REPORT DOCUMENTATION PAGE		*Form Approved* OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**		
1. REPORT DATE *(DD-MM-YYYY)* 18-05-2018	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From–To)*** 09-09-2015–18-05-2018
4. TITLE AND SUBTITLE Solution of Optimal Control Problem for High-Speed Ascent and Reentry Vehicles		**5a. CONTRACT NUMBER** FA8651-08-D-0108 0054
		5b. GRANT NUMBER N/A
		5c. PROGRAM ELEMENT NUMBER N/A
6. AUTHOR(S) Anil V. Rao and William W. Hager		**5d. PROJECT NUMBER** N/A
		5e. TASK NUMBER N/A
		5f. WORK UNIT NUMBER W0YZ
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Florida Office of Engineering Research 339 Well Hall Gainesville, FL 32611		**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Air Force Air Force Material Command AFRL—Eglin Research Site 101 West Eglin Blvd Eglin AFB FL, 32542-6864		**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RWWN
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RW-EG-TR-2018-058
12. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION A. Approved for public release: distribution unlimited 96TW-2018-0258.		
13.SUPPLEMENTARY NOTES Subject to Export Control Laws. Distribution Statement indicating authorized access is on the cover page and block 12 of this form. Data rights restrictions and availability of this report are shown on the Notice and Signature page. Report contains color.		

14. ABSTRACT

Advancements to a general-purpose computational framework is described for solving constrained nonlinear optimal control problems. The framework consists of the following three modules that operate synergistically to improve the accuracy and computational efficiency that can be achieved when solving constrained optimal control problems. The first module is a new class of efficient discretization methods called hp-adaptive Gaussian quadrature methods that transcribe the continuous optimal control problem to a finite-dimensional nonlinear optimization problem. The hp-adaptive methods have the feature that both the degree of the polynomial approximation and the number of mesh intervals can be adjusted to improve the accuracy in the approximation of the solution to the optimal control problem. The second module is a new approach to nonlinear optimization that employs conjugate gradient-based methods with a dual active set method to efficiently solve the non- linear optimization problem associated with the hp-adaptive method. The third module is a novel approach to algorithmic differentiation that produces an efficient derivative source code through a method that combines operator overloading with source transformation. The algorithmic differentiation provides the most accurate derivative possible for use with the gradient-based nonlinear optimization method. Each module in the framework is described and results are shown that demonstrate the effectiveness of the approach. The advancements described in this report include a new hp-adaptive mesh refinement method, a benchmarking study that provides comparisons between various hp-adaptive mesh refinement methods that have been developed in support of this framework, the demonstration of the framework on a high-speed ascent and entry problems, and further advancements to the optimization algorithms and convergence theory of the hp-adaptive methods.

15. SUBJECT TERMS

adaptive trajectory generation, adaptive trajectory reshaping, hp-adaptive mesh refinement, optimal control, optimal path planning, optimum-path-to-go trajectory reshaping

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** SHARON STOCKBRIDGE
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR	164	**19b. TELEPHONE NUMBER *with area code***
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			850-883-7940

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SUMMARY

Advancements to a general-purpose computational framework are described for solving constrained nonlinear optimal control problems. The framework consists of the following three modules, which operate synergistically to improve the accuracy and computational efficiency that can be achieved when solving constrained optimal control problems. The first module is a new class of efficient discretization methods called hp-adaptive Gaussian quadrature methods that transcribe the continuous optimal control problem to a finite-dimensional nonlinear optimization problem. The hp-adaptive methods have the feature that both the degree of the polynomial approximation and the number of mesh intervals can be adjusted to improve the accuracy in the approximation of the solution to the optimal control problem. The second module is a new approach to nonlinear optimization that employs conjugate gradient-based methods with a dual active set method to efficiently solve the non-linear optimization problem associated with the hp-adaptive method. The third module is a novel approach to algorithmic differentiation that produces an efficient derivative source code through a method that combines operator overloading with source transformation. The algorithmic differentiation provides the most accurate derivative possible for use with the gradient-based nonlinear optimization method. Each module in the framework is described, and results are shown that demonstrate the effectiveness of the approach. The advancements described in this report include a new hp-adaptive mesh refinement method, a benchmarking study that provides comparisons between various hp-adaptive mesh refinement methods that have been developed in support of this framework, the demonstration of the framework on a high-speed ascent and entry problems, and further advancements to the optimization algorithms and convergence theory of the hp-adaptive methods.

2 INTRODUCTION

Over the past two decades, the breadth of applications of optimal control has increased tremendously. While previously the subject of optimal control was considered applicable only in engineering, today optimal control is used not only in all branches of engineering (chemical, mechanical, electrical, biomedical, and aerospace), but also is used in application domains such as medicine, economics, and epidemiology. Because the specific problems that arise in these application areas can be quite complex and have no analytic solutions, it has become increasingly important to develop numerical methods that are capable of solving complex optimal control problems.

Without loss of generality, consider an optimal control problem in the so-called Bolza form [1]. The objective is to determine the state, $y(t) \in \mathbb{R}^n$, the control $u(t) \in \mathbb{R}^m$, the initial time, $t_0 \in \mathbb{R}$, and the terminal time, $t_f \in \mathbb{R}$, that minimize the objective functional

$$J = \mathcal{M}\big(y(t_0), t_0, y(t_f), t_f\big) + \int_{t_0}^{t_f} \mathcal{L}(y(t), u(t), t)dt \tag{1}$$

subject to the dynamic constraint

$$\dot{y}(t) = f(y(t), u(t), t) \tag{2}$$

the path constraint

$$c(y(t), u(t), t) \leq 0 \tag{3}$$

and the boundary condition

$$b\big(y(t_0), t_0, y(t_f), t_f\big) = 0 \tag{4}$$

where

$$\begin{gathered}
\mathcal{M}\colon \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R} \\
\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \\
f\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n \\
c\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^p \\
b\colon \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^q
\end{gathered}$$

The Bolza optimal control problem given in Equations (1) through (4) gives rise to the following first-order calculus of variations [2–4] conditions:

$$\dot{y} = \frac{\partial \mathcal{H}}{\partial \lambda} \tag{5}$$

$$\dot{\lambda} = -\frac{\partial \mathcal{H}}{\partial y} \tag{6}$$

$$u^* = \arg\min_{u \in \mathcal{U}} \mathcal{H} \tag{7}$$

The transversality conditions are given by

$$\lambda(t_0) = -\frac{\partial \mathcal{M}}{\partial y(t_0)} + v^T \frac{\partial b}{\partial y(t_0)},\ \lambda(t_f) = \frac{\partial \mathcal{M}}{\partial y(t_f)} - v^T \frac{\partial b}{\partial y(t_f)} \tag{8}$$

$$\mathcal{H}(t_0) = \frac{\partial \mathcal{M}}{\partial t_0} - v^T \frac{\partial b}{\partial t_0},\ \mathcal{H}(t_f) = -\frac{\partial \mathcal{M}}{\partial t_f} + v^T \frac{\partial b}{\partial t_f} \tag{9}$$

$$\begin{aligned} \mu_j(t) &= 0 \quad when\ c_j(y,u,t) < 0,\ j = 1, \cdots, p \\ \mu_j(t) &\leq 0 \quad when\ c_j(y,u,t) = 0,\ j = 1, \cdots, p \end{aligned} \tag{10}$$

where $\lambda(t) \in \mathbb{R}^n$ is the adjoint or costate, $\mu(t)$ is the path constraint multiplier, $\mathcal{H}(y,\lambda,\mu,u,t) = \mathcal{L} + \lambda^T f - \mu^T c$ is the control Hamiltonian, ν is the Lagrange multiplier associated with the boundary conditions, and $\mathcal{U}$ is the admissible control set. Equations (5) and (6) form what is classically known as a Hamiltonian system [2, 3]. Equation (7) is known as Pontryagin's Minimum Principle (PMP) [5] and is used to determine the optimal control as a function of the state and costate. The conditions in Equation (8) are called transversality conditions [2, 3, 6] on the boundary values of the costate, while the conditions given in Equation (10) are the complementary slackness conditions [7–9] on the path constraints. The Hamiltonian system, together with the original boundary conditions, the costate transversality conditions, and complementary slackness conditions, forms a Hamiltonian boundary-value problem (HBVP) [2, 3, 10]. Any solution $(y(t), \lambda(t), u(t), \mu(t), \nu)$ to the HBVP is called an extremal solution.

2.1 Numerical Methods Used in Optimal Control

Most optimal control problems of practical interest must be solved numerically because analytic solutions generally cannot be determined. Excellent surveys of numerical methods for optimal control can be found in References [11] and [12], while a survey of vehicular optimal control can be found in Reference [13]. Numerical methods for optimal control are divided based on the actual problem being solved and the method used to simulate the dynamics. The two different options for problems being solved are either the first-order necessary conditions that arise from variational calculus or an approximation of the optimal control problem itself. A method that attempts to find a solution to the first-order necessary conditions arising from variational calculus is called an indirect method, while a method that attempts to solve an approximation of the optimal control problem itself is called a direct method. An indirect method essentially solves a multiple-point boundary-value problem where the boundary conditions at either end are only partially known. A direct method transcribes the optimal control problem to a finite-dimensional nonlinear optimization problem; this nonlinear optimization problem is then solved numerically.

Once a solution approach has been chosen, the differential-algebraic equations associated with the problem being solved are simulated numerically using one of two types of simulation methods. In explicit simulation, the dynamics are integrated using a time-marching approach where the solution at a given time step is obtained from the solution of the dynamics at one or more previous time steps. This process of time-marching is then repeated by making progressively better approximations to the unknown boundary conditions until the known boundary conditions are satisfied. In implicit simulation, the solutions at all integration time steps, along with the boundary conditions, are obtained simultaneously without any notion of "time." Thus, the dynamics associated with an indirect method or a direct method can be simulated using either implicit or explicit simulation. While in the context of optimal control, direct methods solve an optimization problem, both indirect methods and direct methods have been developed more generally for the solution of a general boundary-value problem (that is, not specifically for an optimal control problem) of the form

$$\dot{y} = f(y(t), t),\ \ b\big(y(t_0), t_0, y\big(t_f\big), t_f\big) = 0 \tag{11}$$

where neither the initial conditions $y(t_0)$ nor the terminal conditions $y\big(t_f\big)$ are in general completely known. As a result, (11) must be solved while seeking simultaneously to any missing components of $y(t_0)$ and/or $y\big(t_f\big)$.

2.1.1 Explicit Simulation (Time-Marching)

In explicit simulation, typically performed by time-marching, the solution of the differential equation at a future time step is obtained using the solution at current and/or previous time steps. The most basic explicit simulation method is the shooting method [14]. In shooting, an initial guess is made of the unknown boundary conditions at one end of the interval. Using this guess, together with the known conditions at that endpoint, the differential equation in Equation (11) is integrated across the time interval of interest using a well-known time-marching method (for example, Euler or Runge-Kutta). Upon reaching the opposite end of the time interval $\big[t_0,\ t_f\big]$, the boundary conditions in Equation (11) are evaluated and an assessment is made as to how to update the unknown boundary conditions. The process of integrating the dynamics and updating the unknown conditions is repeated until the boundary conditions on Equation (11) are satisfied to within a specified accuracy tolerance.

The standard shooting method can present numerical difficulties due to instabilities in the dynamics. In order to overcome this difficulty, the multiple-shooting method [15] was developed. In a multiple-shooting method, the time interval $\left[t_0,\, t_f\right]$ is divided into K subintervals $S_{k=}[t_{k-1},\, t_k]$, where $t_k = t_f$ and $\bigcup_{k=1}^{K} S_{k=}\left[t_0,\, t_f\right]$. The shooting method is then applied over each subinterval $[t_{k-1},\, t_k]$ with the initial value of $y(t_{k-1}),\ k = 2, \cdots, K$ unknown. In order to enforce continuity, the conditions $y(t_k^-) = y(t_k^+),\ k = 1, \cdots, K-1$ must be recognized. These continuity conditions result in a vector-valued root-finding problem, where it is desired to satisfy the boundary conditions in Equation (11) in combination with

$$y(t_k^-) - y(t_k^+) = 0,\ k = 1, \cdots, K-1$$

Despite being of higher dimension, the multiple-shooting method represents an improvement over the standard shooting method, because the shorter integration intervals reduce the sensitivity to errors of the unknown terminal conditions.

2.1.2 Implicit Simulation (Collocation)

In implicit simulation, the boundary-value problem of Equation (11) is again solved by dividing the solution interval $\left[t_0,\, t_f\right]$ into mesh intervals $S_{k=}[t_{k-1},\, t_k]$; $k = 1, \cdots, K$, where $t_k = t_f$ and $\bigcup_{k=1}^{K} S_{k=}\left[t_0,\, t_f\right]$. In each mesh interval $S_{k=}$ the function $y(t)$ is approximated in terms of a set of basis functions $\psi_j^{(k)}(t)$, that is

$$y^{(k)}(t) \approx Y^{(k)}(t) = \sum_{j=1}^{J} c_j^{(k)} \psi_j^{(k)}(t) \tag{12}$$

This parameterization is used to integrate Equation (11) from t_{k-1} to one or more stage times $t_{ik} \in S_k$; $i = 1, \cdots, I$, where $t_{k-1} < t_{1k} < t_{2k} < \cdots < t_{(i-1),k} < t_{Ik} = t_k$ for all $i \in [1, \cdots, I]$ as follows

$$Y(t_{ik}) = Y(t_{k-1}) + \int_{t_{k-1}}^{t_{ik}} f(Y(\tau), \tau)\, d\tau,\ (i = 1, \cdots, I) \tag{13}$$

The integrals given in Equation (13) are then replaced by quadrature approximations of the form

$$Y_{ik} = Y_{k-1} + \sum_{j=1}^{I} A_{ij}^{(k)} f\big(Y_{jk}, t_{jk}\big), \qquad (i = 1, \cdots, I) \tag{14}$$

where Y_{jk}; $j = 1, \cdots, I$ are the values of the function approximation $Y(t)$ at the stage points t_{jk}; $j = 1, \cdots, I$ and $A_{ij}^{(k)}$; $i, j = 1, \cdots, I$ is the integration matrix associated with the particular integration (quadrature) rule that is used to integrate the dynamics in mesh interval S_k. Equation (14) can then be rearranged in the form

$$Y_{ik} - Y_{k-1} - \sum_{j=1}^{I} A_{ij}^{(k)} f\big(Y_{jk}, t_{jk}\big) = 0, \qquad (i = 1, \cdots, I) \tag{15}$$

which are called defect conditions that must be satisfied at all stage times in each mesh interval. In addition, assuming that the solution is continuous, continuity conditions $y(t_k^-) = y(t_k^+)$ must be satisfied at every mesh interval interface. The goal then is to solve for the coefficients $c_j^{(k)}$ in each mesh interval. As opposed to the seemingly similar explicit simulation method and multiple shooting, in implicit simulation, all of defect equations are found simultaneously by solving a large-scale algebraic system of the form $F(z) = 0$. Implicit simulation is often referred to as collocation, because the defect conditions in Equation (15) make the value of the function at each stage point equal to the quadrature approximation of the dynamics at those points.

2.1.3 Indirect Methods

The first-order necessary conditions given in Equations (5) through (10) constitute a two-point boundary-value problem. This boundary-value problem can be solved using either explicit or implicit simulation as described above. A typical explicit simulation method would be indirect shooting or indirect multiple-shooting, while a typical simultaneous indirect method would be indirect collocation. In an indirect shooting method [14], an initial guess is made for the unknown boundary conditions at one boundary. Using this guess, together with the known initial conditions, the Hamiltonian system in Equations (5) and (6) is integrated to the other boundary (that is, either forward from t_0 to $t_f$or backward from t_f to t_0) using a time marching method. Upon reaching t_f, the terminal conditions obtained from the numerical integration are compared to the known terminal conditions given in Equations (4) and (8). If the integrated terminal conditions differ from the known terminal conditions by more than a specified tolerance, the unknown initial conditions are adjusted and the process is repeated until the difference between the integrated terminal conditions and the required terminal conditions is less than some specified threshold. While the simplicity of simple shooting is appealing, this technique suffers from significant numerical difficulties due to ill-conditioning. The shooting method poses particularly poor characteristics when the optimal control problem is hypersensitive [16–20] (that is, when time interval of interest is long in comparison with the time-scales of the Hamiltonian system in a neighborhood of the optimal solution). In an indirect collocation method, the state and costate can be parametrized using piecewise polynomials, as shown in Equation (12). The collocation procedure leads to a large-scale root-finding problem $F(z) = 0$, where the vector of unknown coefficients z consists of the coefficients of the piecewise polynomial. This system of nonlinear equations is then solved using an appropriate root-finding technique.

2.1.4 Direct Methods

The most basic direct explicit simulation method for solving optimal control problems is the direct shooting method. The control is parameterized as

$$u(t) = U(t) \approx \sum_{i=1}^{m} \alpha_i \psi_i(t) \tag{16}$$

in which $\psi_i(t)$, $(i = 1, \cdots, m)$ are preassigned basis functions and α_i, $(i = 1, \cdots, m)$ are the parameters to be determined by optimization. The dynamics are then satisfied by integrating the differential equations using a time-marching integration method. Similarly, the cost function of Equation (1) is determined using a quadrature approximation that is consistent with the numerical integrator used to solve the differential equations. The NLP that arises from direct shooting then minimizes the cost subject to any path and interior-point constraints.

Direct shooting suffers from issues similar to those associated with indirect shooting in that instabilities arise due to the long time interval over which the time marching method is applied. Thus, in a manner similar to that for indirect methods, an alternate direct explicit simulation method is the direct multiple-shooting method. In this case, the time interval $[t_0, t_f]$ is divided into K subintervals. The aforementioned direct shooting method is then used over each subinterval $[t_{k-1}, t_k]$ with the values of the state at the beginning of each subinterval and the unknown coefficients in the control parameterization being unknowns in the optimization. In order to enforce continuity, the following conditions are enforced at the interface of each subinterval:

$$y(t_k^-) - y(t_k^+) = 0, \ (k = 1, \cdots, K-1) \tag{17}$$

The continuity conditions of Equation (17) result in vector root-finding problem where it is desired to drive the values of the difference $y(t_k^-) - y(t_k^+)$ to zero. It is seen that the direct multiple-shooting method increases the size of the optimization problem because the values of the state at the beginning of each subinterval are parameters in the optimization. Despite the increased size of the problem due to these extra variables, the direct multiple-shooting method is an improvement over the standard direct shooting method because the sensitivity to errors in the unknown initial conditions is reduced because integration is performed over significantly smaller time intervals.

2.1.5 Direct Collocation Methods

Over the past two decades implicit simulation direct collocation methods have become the de facto standard for the numerical solution of optimal control problems. A direct collocation method is an implicit simulation approach, where both the state and control are parameterized in terms of basis functions (generally piecewise polynomials) for implicit simulation. In the context of optimal control, fixed-order integration methods such as Runge-Kutta and Hermite-Simpson methods [21–28] have been employed. In direct collocation methods, the following nonlinear programming problem (NLP) arises from the approximation of the optimal control problem:

$$\min F(z) \;\; subject\ to \begin{cases} z_{min} \le z \le z_{max}, \\ g(z) \le 0. \end{cases}$$

As a rule, this NLP is large, containing tens or even hundreds of thousands of variables and constraints. The key feature of these problems, which makes them tractable, is the fact that the NLP is sparse. A variety of well-known NLP solvers such as SNOPT [29, 30], IPOPT, and [31] have been developed specifically for solving large sparse NLPs. In a typical fixed-order collocation method, such as a Runge-Kutta method, accuracy is improved by increasing the number of mesh intervals. A limitation of a fixed-order method is that, for complex problems, the only way to achieve convergence of the true optimal solution is to increase significantly the size of the mesh.

3 GENERAL METHODOLOGY (METHODS, ASSUMPTIONS, AND PROCEDURES)

The goal of this research is to advance a novel general-purpose computational framework for solving constrained nonlinear optimal control problems and to demonstrate the effectiveness of the framework on a problem of interest in high-speed ascent and entry. The first module is that of hp-adaptive Gaussian quadrature orthogonal collocation mesh refinement [32–45]. Earlier work conceived the idea of an hp-adaptive mesh refinement method, and some methods were developed during that early work. The advancement in this research is the development of a new mesh refinement method that provides accurate detection of discontinuities in solutions of optimal control problems using jump function approximations [46]. The second module of the module of the framework is the development of advanced optimization techniques for solving the sparse nonlinear programming problem that arises from the hp-adaptive discretization. Specifically, the hp-adaptive Gaussian quadrature collocation leads to a highly sparse nonlinear programming problem (NLP). The approach developed in this research is to develop new conjugate gradient methods that are able to quickly and accurately solve this sparse NLP. A key feature of this new approach is that it is possible to obtain solutions using only first derivatives for problems where previously only second derivative methods would work.

Being able to solve such problems using only first derivative reduces significantly the number of function evaluations required, thus, making it significantly more capable of rapidly solving a constrained optimal control problem. In the limited memory conjugate gradient algorithm, a basis for the subspace containing the recent prior iterates is stored. Whenever the solution at any iteration becomes trapped, a special subspace iteration is performed to obtain a descent direction orthogonal to the current subspace. Such an approach allows for rapid convergence because the iteration is able to escape the subspace in which it is trapped. As a result, not only is the conjugate gradient method much faster than previous methods, it is significantly more robust because it will continue to iterate when other algorithms fail outright. In this research algorithms that are capable of minimizing a mathematical programming problem with a nonlinear objective function subject to linear constraints are developed. The third module of this framework is the accurate and efficient generation of analytic derivatives for use with the new optimization techniques.

In this research a new algorithmic differentiation method is developed that generates derivative source code using a combined source transformation and operator overloading approach. The approach is found to generate highly efficient code that computes the derivative to machine precision accuracy. Furthermore, the algorithmic differentiation method developed in this research can be applied recursively to generate derivatives of any order. The three prongs of the framework are combined into a software testbed that enables testing the various modules independently and together.

The results of this research indicate that the combination of the hp-adaptive Gaussian quadrature orthogonal collocation methods together with advanced nonlinear programming methods and accurate and efficient derivative generation techniques greatly expand the range of optimal control problems that can be solved, thus enabling current-generation and next-generation applications that may arise in a wide variety of applications, including space and atmospheric flight mission planning, chemical process control, medical applications, and economics. A schematic of how the modules of the framework operate together is shown in Figure 1. The remainder of this report describes the advancements made to the framework shown in Figure 1.

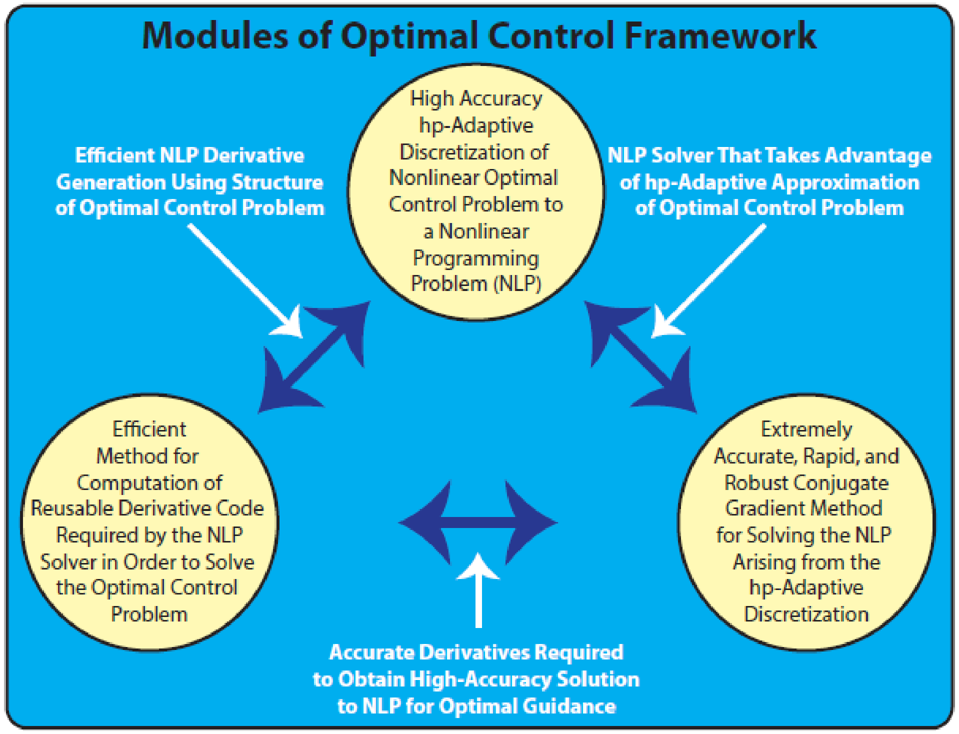

Modules of Optimal Control Framework
High Accuracy hp-Adaptive Discretization of Nonlinear Optimal Control Problem to a Nonlinear Programming Problem (NLP)
Efficient NLP Derivative Generation Using Structure of Optimal Control Problem
NLP Solver That Takes Advantage of hp-Adaptive Approximation of Optimal Control Problem
Efficient Method for Computation of Reusable Derivative Code Required by the NLP Solver in Order to Solve the Optimal Control Problem
Extremely Accurate, Rapid, and Robust Conjugate Gradient Method for Solving the NLP Arising from the hp-Adaptive Discretization
Accurate Derivatives Required to Obtain High-Accuracy Solution to NLP for Optimal Guidance

Figure 1. Three modules of optimal control framework.

4 OVERVIEW OF TECHNICAL RESULTS

The technical results provided in this report are divided into the following sections:

1. Formulation and Discretization of Continuous Optimal Control Problem.
2. New Mesh Refinement Method Employing Jump Function Approximations.
3. Performance Comparison of Various Mesh Refinement Methods.
4. Advancements in Nonlinear Optimization.
5. Application of Framework to Relevant High-Speed Ascent and Entry Problem.

Each of these various aspects of the research are now described.

4.1 Transformed Continuous Optimal Control Problem

Consider now the following change of independent variable:

$$t \equiv t(\tau, t_0, t_f) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{18}$$

The Bolza optimal control problem given in Equations. (1) through (4) can then be described as follows. Determine the state $y(\tau) \in \mathbb{R}^{n_y}$ and the control $u(\tau) \in \mathbb{R}^{n_u}$ on the domain $\tau \in [-1, +1]$, the initial time, t_0, and the terminal time t_f that minimize the cost functional

$$\mathcal{J} = \mathcal{M}\left(y(-1),\ t_0, y(+1), t_f\right) + \frac{t_f - t_0}{2}\int_{-1}^{+1} \mathcal{L}\left(y(\tau), u(\tau), t(\tau, t_0, t_f)\right) d\tau \tag{19}$$

subject to the dynamic constraints

$$\frac{dy}{d\tau} = \frac{t_f - t_0}{2} a\left(y(\tau), u(\tau),\ t(\tau, t_0, t_f)\right) \tag{20}$$

the inequality path constraints

$$c\left(y(\tau), u(\tau),\ t(\tau, t_0, t_f)\right) \leq 0 \tag{21}$$

and the boundary conditions

$$b\left(y(-1), t_0, y(+1), t_f\right) = 0 \tag{22}$$

4.1.1 Partitioning of Bolza Optimal Control Problem into a Mesh

In the hp discretization, the domain $\tau \in [-1, +1]$ is partitioned into a mesh consisting of K mesh intervals $S_k = [T_{k-1}, T_k]$, $k = 1, \cdots, K$, where $-1 = T_0 < T_1 < \cdots < TK = +1$. The mesh intervals have the property that $\cup_{k=1}^{K} S_k = [-1, +1]$. Let $y^{(k)}(\tau)$ and $u^{(k)}(\tau)$ be the state and control in S_k. The Bolza optimal control problem of Equations (19) through (22) can then be rewritten as follows.

Minimize the cost functional

$$\mathcal{J} = \mathcal{M}\left(y^{(1)}(-1),\ t_0,\ y^{(K)}(+1), t_f\right) + \frac{t_f\ -\ t_0}{2}\sum_{k=1}^{K}\int_{T_{k-1}}^{T_k} \mathcal{L}\left(y^{(k)}(\tau), u^{(k)}(\tau), t(\tau, t_0, t_f)\right) d\tau \tag{23}$$

subject to the dynamic constraints

$$\frac{dy^k(\tau)}{d\tau} = \frac{t_f - t_0}{2} a\left(y^{(k)}(\tau), u^{(k)}(\tau), t(\tau, t_0, t_f)\right),\ (k = 1, \cdots, K) \tag{24}$$

the path constraints

$$c\left(y^{(k)}(\tau), u^{(k)}(\tau), t(\tau, t_0, t_f)\right) \leq 0, (k = 1, \cdots, K) \tag{25}$$

and the boundary conditions

$$b\left(y^{(1)}(-1), t_0, y^{(K)}(+1), t_f\right) = 0 \tag{26}$$

Because the state must be continuous at each interior mesh point, it is required that the condition $y(T_k^-) = y(T_k^+)$, $(k = 1, \cdots, K-1)$ be satisfied at the interior mesh points $(T_1, \cdots, T_{K-1})$.

4.1.2 hp-Adaptive Legendre-Gauss-Radau Collocation

The multiple-interval form of the continuous-time Bolza optimal control problem discretized using collocation at Legendre-Gauss-Radau (LGR) points [36–39, 43]. In the LGR collocation method, the state of the continuous-time Bolza optimal control problem is approximated in $S_k, k \in [1, \cdots, K]$, as

$$y^{(k)}(\tau) \approx Y^{(k)}(\tau) = \sum_{j=1}^{N_{k+1}} Y_j^{(k)} \ell_j^{(k)}(\tau), \ell_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_{k+1}} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}} \tag{27}$$

where $\tau \in [-1,+1], \ell_j^{(k)}(\tau), j = 1,\cdots,N_{k+1}$, is a basis of Lagrange polynomials, $\left(\tau_1^{(k)},\cdots,\tau_{N_k}^{(k)}\right)$ are the Legendre-Gauss-Radau (LGR) [47] collocation points in $S_k = [T_{k-1}, T_k]$, and $\tau_{N_{k+1}}^{(k)} = T_k$ is a noncollocated point. Differentiating $Y^{(k)}(\tau)$ in Equation (27) with respect to τ gives

$$\frac{dY^{(k)}(\tau)}{d\tau} = \sum_{j=1}^{N_k+1} Y_j^{(k)} \frac{d\ell_j^{(k)}(\tau)}{d\tau} \tag{28}$$

The dynamics are then approximated at the N_k LGR points in mesh interval $k \in [1,\cdots,K]$ as

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} Y_j^{(k)} = \frac{t_f - t_0}{2} a\left(Y_i^{(k)}, U_i^{(k)}, t\left(\tau_i^{(k)}, t_0, t_f\right)\right), (i = 1,\cdots,N_k) \tag{29}$$

where

$$D_{ij}^{(k)} = \frac{d\ell_j^{(k)}\left(\tau_i^{(k)}\right)}{d\tau}, (i = 1,\cdots,N_k,\ j = 1,\cdots,N_k + 1)$$

are the elements of the $N_k \times (N_k + 1)$ Legendre-Gauss-Radau differentiation matrix [36] in mesh interval S_k, $k \in [1,\ldots,K]$. The LGR discretization then leads to the following nonlinear programming problem (NLP). Minimize the LGR quadrature approximation to the cost functional

$$\mathcal{J} \approx \mathcal{M}\left(Y_1^{(1)}, t_0, Y_{N_K+1}^{(K)}, t_f\right) + \sum_{k=1}^{K}\sum_{j=1}^{N_k} \frac{t_f - t_0}{2} \omega_j^{(k)} \mathcal{L}\left(Y_j^{(k)}, U_j^{(k)}, t\left(\tau_j^{(k)}, t_0, t_f\right)\right) \tag{30}$$

subject to the collocation equations

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} Y_j^{(k)} - \frac{t_f - t_0}{2} a\left(Y_i^{(k)}, U_i^{(k)}, t\left(\tau_i^{(k)}, t_0, t_f\right)\right) = 0, \qquad (i = 1, \cdots, N_k) \tag{31}$$

the discretized path constraints

$$c\left(Y_i^{(k)}, U_i^{(k)}, t\left(\tau_i^{(k)}, t_0, t_f\right)\right) \leq 0,\ \ (i = 1, \cdots, N_k) \tag{32}$$

and the discretized boundary conditions

$$b\left(Y_1^{(1)}, t_0, Y_{N_K+1}^{(K)}, t_f\right) = 0 \tag{33}$$

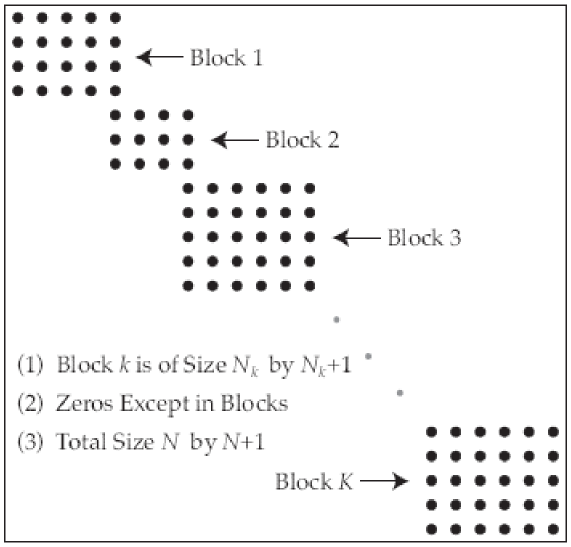
It is noted that the continuity in the state at the interior mesh points $(T_1, \cdots, T_{K-1})$ is enforced via the condition

$$Y_{N_k+1}^{(k)} = Y_1^{(k+1)}, (k = 1, \cdots, K-1) \tag{34}$$

Computationally, the constraint of Equation (34) is eliminated from the problem by using the same variable for both $Y_{N_k+1}^{(k)}$ and $Y_1^{(k+1)}$.

4.1.3 Sparse Structure of NLP Arising from LGR Collocation

The nonlinear programming problem (NLP) arising from the Legendre-Gauss-Radau collocation has a particular sparse structure that makes it efficient to solve. The sparse structure is seen in three different parts. First, a schematic of the composite Radau differentiation matrix D is shown in Figure 2 where it is seen that D has a block structure with nonzero elements in the row-column indices $\left(\sum_{l=1}^{k-1} N_l + 1, \cdots, \sum_{l=1}^{k} N_l, \sum_{l=1}^{k-1} N_l + 1, \cdots, \sum_{l=1}^{k} N_l + 1\right)$, where for every mesh interval $k \in [1, \cdots, K]$ the nonzero elements are defined by the LGR differentiation matrix. The second part of the sparse structure is seen in the NLP Jacobian shown in Figure 3. First, it is seen that the vast majority of the elements in the Jacobian are zero. The remaining (nonzero) elements appear in well-defined location in the Jacobian, making it possible to clearly identify the sparse structure. Finally, Figure 4 shows the sparse structure of the NLP Hessian, which would be used if the NLP was solved using a full Newton NLP solver. It is interesting to see that the Hessian is even sparser than the Jacobian. The overall structure of the Jacobian and the Hessian leads to a highly sparse NLP that can be solved efficiently using sparse nonlinear optimization techniques.



Block 1
Block 2
Block 3
(1) Block k is of Size N_k by N_k+1
(2) Zeros Except in Blocks
(3) Total Size N by $N+1$
Block K

Figure 2. Structure of composite Radau pseudospectral differentiation matrix where the mesh consists of K mesh intervals.

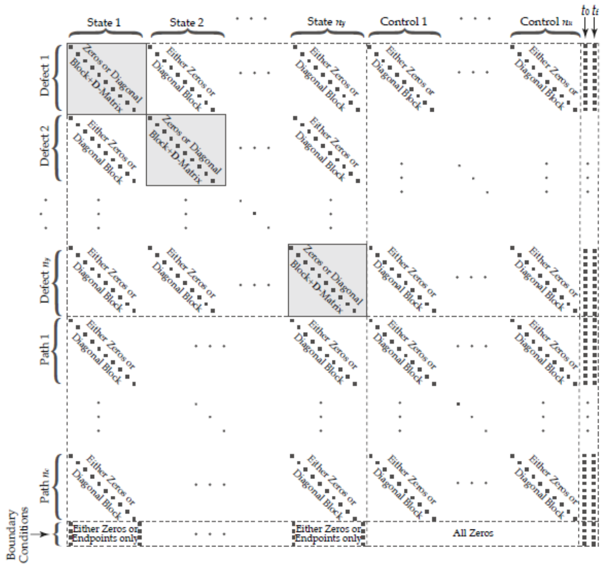


Figure 3. General Jacobian sparsity pattern for RPM.

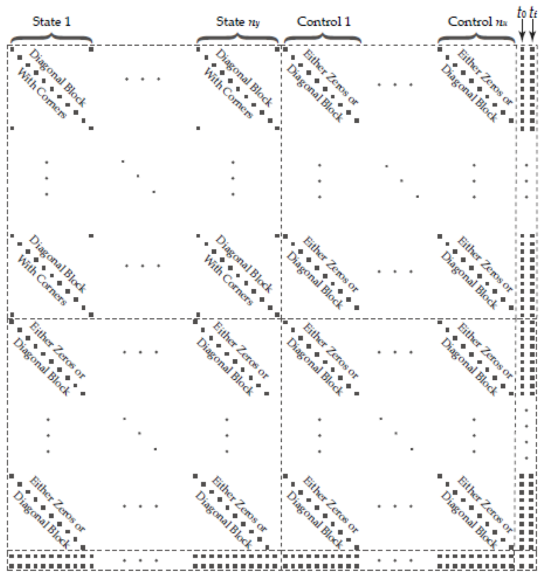


Figure 4. General Hessian sparsity pattern for RPM.

4.1.4 Approximation of Solution Error

In this section, the approach of Ref. [48] for estimating the relative error in the solution on a given mesh is reviewed. The relative error approximation derived in Ref. [48] is obtained by comparing two approximations to the state, one with higher accuracy. The key idea is that for a problem whose solution is smooth, an increase in the number of LGR points should yield a state that more accurately satisfies the dynamics. Hence, the difference between the solution associated with the original set of LGR points, and the approximation associated with the increased number of LGR points should yield an approximation of the error in the state.

Assume that the NLP of Equations (30) through (33) corresponding to the discretized Bolza optimal control problem has been solved on a mesh $S_k = [T_{k-1}, T_k], k = 1, \cdots$, with N_k LGR points in mesh interval S_k. Suppose that the objective is to approximate the error in the state at a set of $M_k = N_k + 1$ LGR points $\left(\hat{\tau}_1^{(k)}, \ldots, \hat{\tau}_{M_k}^{(k)}\right)$, where $\hat{\tau}_1^{(k)} = \tau_1^{(k)} = T_{k-1}$, and that $\hat{\tau}_{M_k+1}^{(k)} = T_k$. Suppose further that the values of the state approximation at the points $\left(\hat{\tau}_1^{(k)}, \ldots, \hat{\tau}_{M_k}^{(k)}\right)$ are denoted $\left(y\left(\hat{\tau}_1^{(k)}\right), \ldots, y\left(\hat{\tau}_{M_k}^{(k)}\right)\right)$. Next, let the control be approximated in S_k using the Lagrange interpolating polynomial

$$U^{(k)}(\tau) = \sum_{j=1}^{N_k} U_j^{(k)} \hat{\ell}_j^{(k)}(\tau)\,, \hat{\ell}_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}} \tag{35}$$

and let the control approximation at $\hat{\tau}_i^{(k)}$ be denoted $u\left(\hat{\tau}_1^{(k)}\right), 1 \le i \le M_k$. The value of the right-hand side of the dynamics at $(Y\left(\hat{\tau}_i^{(k)}\right), U\left(\hat{\tau}_i^{(k)}\right), \hat{\tau}_i^{(k)})$ is used to construct an improved approximation of the state. Let $\hat{Y}^{(k)}$ be a polynomial of degree at most M_k that is defined on the interval S_k. If the derivative of $\hat{Y}^{(k)}$ matches the dynamics at each of the Radau quadrature points $\hat{\tau}_i^{(k)}, 1 \le i \le M_k$, we then have

$$\hat{Y}^{(k)}\left(\hat{\tau}_j^{(k)}\right) = Y^{(k)}(T_{k-1}) + \frac{t_f - t_0}{2} \sum_{l=1}^{M_k} \hat{I}_{jl}^{(k)}\, a\left(Y^{(k)}\left(\hat{\tau}_l^{(k)}\right), U^{(k)}\left(\hat{\tau}_l^{(k)}\right), t\left(\hat{\tau}_l^{(k)}, t_0, t_f\right)\right), \quad j = 2, \cdots, M_k + 1, \tag{36}$$

where $\hat{I}_{jl}^{(k)}, j, l = 1, \cdots, M_k$, is the $M_k \times M_k$ LGR integration matrix corresponding to the LGR points defined by $\left(\hat{\tau}_1^{(k)}, \cdots, \hat{\tau}_{M_k}^{(k)}\right)$. Using the values $y\left(\hat{\tau}_l^{(k)}\right)$ and $\hat{y}\left(\hat{\tau}_l^{(k)}\right), \mathrm{l} = 1, \cdots, M_k + 1$, the absolute and relative errors in the i^{th} component of the state at $\left(\hat{\tau}_1^{(k)}, \cdots, \hat{\tau}_{M_k+1}^{(k)}\right)$ are then defined, respectively, as

$$\begin{aligned} E_i^{(k)}\left(\hat{\tau}_l^{(k)}\right) &= \left|\hat{Y}_i^{(k)}\left(\hat{\tau}_l^{(k)}\right) - Y_i^{(k)}\left(\hat{\tau}_l^{(k)}\right)\right|, \\ e_i^{(k)}\left(\hat{\tau}_l^{(k)}\right) &= \frac{E_i^{(k)}\left(\hat{\tau}_l^{(k)}\right)}{1 + \max\limits_{\substack{j \in [1, \cdots, N_k + 1] \\ k \in [1, \cdots, K]}} \left|Y_i^{(k)}\left(\tau_j^{(k)}\right)\right|}, \begin{bmatrix} l = 1, \cdots, M_k + 1 \\ i = 1, \cdots, n_y \end{bmatrix} \end{aligned} \tag{37}$$

The maximum relative error in S_k is then defined as

$$e_{max}^{(k)} = \max_{\substack{i \in [1, \cdots, n_y] \\ l \in [1, \cdots, M_k + 1]}} e_i^{(k)}\left(\hat{\tau}_l^{(k)}\right) \tag{38}$$

4.2 hp-Adaptive Mesh Refinement Using Jump Function Approximations

Optimal control problems often have nonsmooth solutions. Such discontinuities may arise in the form of jump discontinuities in the control (for example, an optimal control problem whose optimal control has a bang-bang structure) or discontinuities in the derivative of the state and/or the control. In order to improve the accuracy of a numerical solution to an optimal control problem whose solution is nonsmooth, the locations of such discontinuities need to be determined accurately.

This paper focuses on the development of a mesh refinement method, which accurately locates jump discontinuities in the control when solving an optimal control problem using a direct collocation method. The method developed in this paper employs a jump function approximation to identify the locations of control discontinuities. The jump function approximation is generated using the approximation of the control obtained at the collocation points on a given mesh for which the optimal control problem is approximated. The jump function approximation is then used to develop an iterative mesh refinement method where, on each mesh refinement iteration, the mesh is modified using the estimates of the locations of the discontinuities. The remainder of this section provides the mathematical background that serves as the basis of the method.

4.2.1 Motivation for New Mesh Refinement Method

Optimal control problems often have nonsmooth solutions. Such discontinuities may arise in the form of jump discontinuities in the control (for example, an optimal control problem whose optimal control has a bang-bang structure) or discontinuities in the derivative of the state and/or the control. In order to improve the accuracy of a numerical solution to an optimal control problem whose solution is nonsmooth, the locations of such discontinuities need to be determined accurately.

This paper focuses on the development of a mesh refinement method which accurately locates jump discontinuities in the control when solving an optimal control problem using a direct collocation method. The method developed in this paper employs a jump function approximation to identify the locations of control discontinuities. The jump function approximation is generated using the approximation of the control obtained at the collocation points on a given mesh for which the optimal control problem is approximated. The jump function approximation is then used to develop an iterative mesh refinement method where, on each mesh refinement iteration, the mesh is modified using the estimates of the locations of the discontinuities. The remainder of this section provides the mathematical background that serves as the basis of the method.

4.2.2 Jump Functions

Let $f(t)$ be an arbitrary function defined on $t \in [t_0, t_f\ ]$. The jump function, $f_j(t)$, that arises from $f(t)$ is defined as

$$f_j(t) = \lim_{\tau \to t^+} f(\tau) - \lim_{\tau \to t^-} f(\tau) \ \forall t \in \left(t_0, t_f\right) \tag{39}$$

From Equation (39) it is seen that a jump function, $f_j(t)$, of an underlying function, $f(t)$, is zero everywhere except at locations where the original function, $f(t)$, has jump discontinuities. Moreover, at the jump locations of $f(t)$, $f_j(t)$ takes on the value that equals the amount of the jump discontinuity itself. Using the definition of a jump function, the jump discontinuities can be located by observing where $f_j(t)$ is nonzero.

4.2.3 Approximation of Jump Functions

Now while in principle a jump function can be obtained using Equation (39), in practice the underlying function is not known because the solution is known only on a time series of data where the time points are the collocation points on the mesh for which the approximation of the solution to the optimal control problem was obtained. Thus, Equation (39) must be approximated using this time series of data. A possible way to approximate a jump function is by using a Fourier series approximation of $f(t)$. If the Fourier series approximation of a $2L$ periodic function, denoted $\hat{f}(t)$, is written as

$$f(t) \approx \hat{f}(t) = a_0 + \sum_{k=1}^{N} \left[a_k \cos\left(\frac{k\pi}{L} t\right) + b_k \sin\left(\frac{k\pi}{L} t\right) \right] \tag{40}$$

then the jump function approximation, denoted $\hat{f}_j(t)$, has the same $2L$ period and is defined as

$$f_j(t) \approx \hat{f}_j(t) = a_0 + \sum_{k=1}^{N} \sigma(k/N) \left[a_k \sin\left(\frac{k\pi}{L} t\right) - b_k \cos\left(\frac{k\pi}{L} t\right) \right] \tag{41}$$

where a_k and b_k are the same Fourier coefficients of Equation (40) and $\sigma(k/N)$ are concentration factors [49]. Concentration factors arise from the earlier work of Lukács [50, 51] where it was shown that the conjugate Fourier series

$$\tilde{\hat{f}}(t) = \sum_{k=1}^{N} \left[a_k \sin\left(\frac{k\pi}{L} t\right) - b_k \cos\left(\frac{k\pi}{L} t\right) \right] \tag{42}$$

converges to the jump function when multiplied by $-\pi/\log N$ as $N \to \infty$. A wider class of so-called "concentration factors" share the same convergence property when applied to the Fourier conjugate sum as written in Equation (41) [49]. Furthermore, these concentration factors accelerate convergence to the actual jump function, thereby making the jump function approximation computationally tractable because fewer terms in the series are required in order to obtain an accurate approximation of the jump function.

In this research, the following concentration factor is employed:

$$\sigma_{k,N}^{G} = -\frac{\pi}{Si(\pi)}\sin\left(\frac{\pi k}{N}\right), Si(\pi) = \int_0^{\pi}\frac{\sin t}{t}dt \approx 1.85194 \tag{43}$$

where Equation (43) is called the Gibbs concentration factor [49].

While obtaining the Fourier coefficients needed in Equation (41) can be done in many ways, it is important that an even Fourier approximation be used as opposed to a standard or odd Fourier approximation. The reasoning for needing an even Fourier approximation is due to the nature of the periodic behavior the standard, odd, and even Fourier approximations imply. Specifically, consider the numerical approximation of the solution to an optimal control problem on a given mesh where the solution data lies on the time interval $t \in [t_0, t_f\ ]$. A standard Fourier approximation of the form in Equation (40) will result in the Fourier approximation of the control having a period $T = 2L = t_f - t_0$. Due to this periodicity, artificial jumps may be present at the endpoints as illustrated in Figure 6. The same issue arises when using the odd Fourier approximation, $\hat{f}_{odd}(t)$, which is defined as

$$\hat{f}_{odd}(t) = \sum_{k=1}^{N} b_k \sin\left(\frac{k\pi}{L}t\right) \tag{44}$$

except now the period $T = 2L = 2(t_f - t_0)$. It is noted, however, that an even Fourier approximation, denoted $\hat{f}_{even}(t)$ and defined as

$$\hat{f}_{even}(t) = a_0 + \sum_{k=1}^{N} a_k \cos\left(\frac{k\pi}{L}t\right) \tag{45}$$

has the same period as the odd approximation but does not pose the risk of creating artificial jumps at t_0 or t_f. An even Fourier series approximation does not produce artificial jumps at the endpoints because the value of the function approximation at the start of any period is equal to the value of the function approximation at the end of the previous period.

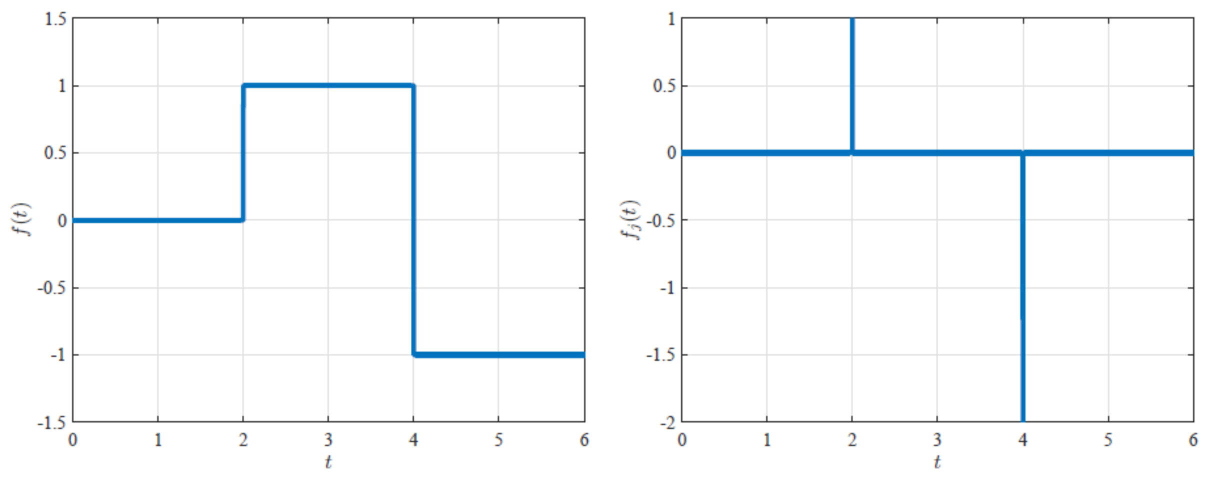
In order to see the behavior of a standard, odd, and even Fourier series approximation of a jump function, consider the following function:

$$f(t) = \begin{cases} 0, & 0 \le t < 2 \\ 1, & 2 \le t < 4 \\ -1, & 4 \le t \le 6 \end{cases} \tag{46}$$

The jump function $f_j(t)$ arising from the function $f(t)$ defined in Equation (46) is given as

$$f_j(t) = \begin{cases} 1, & t = 2 \\ -2, & t = 4 \\ 0, & otherwise \end{cases} \tag{47}$$

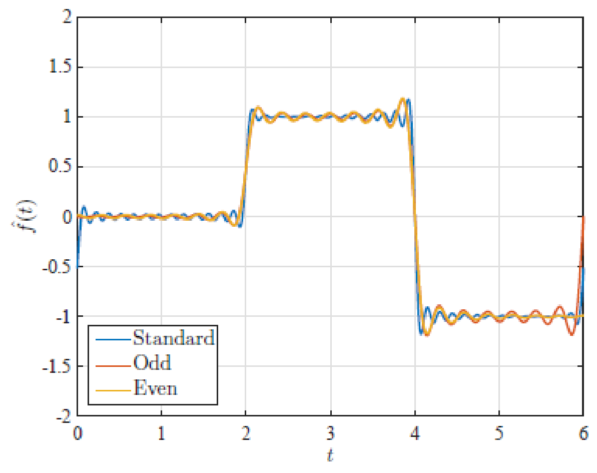
Figure 5 images (a) and (b) show, respectively, the functions $f(t)$ and $f_j(t)$ defined in Equations (46) and (47). Next, Figure 6a shows the standard, odd, and even Fourier series approximations of the function $f(t)$ defined in Equation (46), while Figure 6b shows the standard, odd, and even Fourier series approximations of the jump function $f_j(t)$ defined in Equation (47). It can be seen from Figure 6a that all three Fourier approximations provide a good approximation of $f(t)$. On the other hand, Figure 6b shows that the standard and odd Fourier approximations of the jump function $f_j(t)$ defined in Equation (47) produce artificial jumps at $t = t_0$ and $t = t_f$, while artificial jumps are not produced by the even Fourier approximation of $f_j(t)$. Because the even Fourier series approximation of a jump function does not produce artificial jumps, even Fourier approximations will be employed in the remainder of this paper to obtain the Fourier coefficients needed in the approximation of a jump function. Finally, for convenience, from this point forth the notation $\hat{f}(t)$ will be used to denote an even Fourier series approximation.



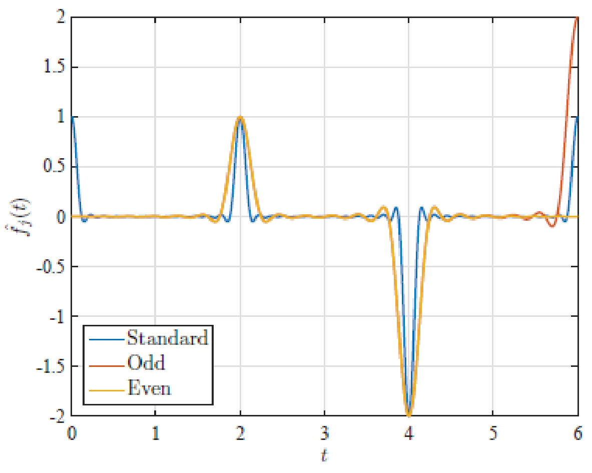
(a) Function, $f(t)$, defined in Eq. (46). (b) Jump function, $f_j(t)$, defined in Eq. (47).

Figure 5. Example function, $f(t)$, defined in Equation (46) alongside jump function, $f_j(t)$, defined in Equation 47.

Given that an even Fourier series approximation does not create artificial jumps when approximating a jump function, suppose now the jump function $f_j(t)$ given Equation (47) is approximated using an even N-term Fourier series, and let this jump function approximation be denoted $\hat{f}_j(t)$. Figure 5 shows the jump function approximation $\hat{f}_j(t)$ for $N = \{10, 20, 40\}$, where it is seen that the jump function approximation approaches the true jump function as N increases. Furthermore, the locations of the maxima and minima of the jump function approximation lie in close proximity, respectively, to the locations of the discontinuities of the actual function $f(t)$, and the values of the jump function approximation at these extremal points are in close proximity to the actual jump in the original function. It is also seen that the extrema in the jump function approximation tend to stay in the same location when a jump discontinuity is present, regardless of the value of N. Therefore, locating the maxima and minima in a jump function approximation can be used as a good estimate of the location of jump discontinuities of a function $f(t)$.



(a) Fourier approximation of $f(t)$ defined in Eq. (46).



(b) Jump function approximation of $f_j(t)$ defined in Eq. (47).

Figure 6. Standard, odd, and even Fourier approximations of $f(t)$ and their associated jump function approximations. Note that $\hat{f}_j(t)$ has no artificial jumps at the endpoints for the even approximation.

4.2.4 Even Fourier Series Approximation of Jump Functions Using Unevenly Spaced Data

In the example that was studied in Section 4.2.3, the underlying function, $f(t)$, was known. As a result, it was possible to obtain the Fourier coefficients of the jump function, $f_j(t)$, of $f(t)$, analytically. Note, however, that the solution obtained by solving the NLP that arises from the transcription of an optimal control problem via collocation leads to an approximation of the state and control at discrete (sampled) data points. Moreover, this discrete approximation is obtained at points that are not evenly spaced. Thus, any jump function approximation that would be used to determine the locations of discontinuities in the solution must be obtained using this discrete data. In this paper, an even Fourier approximation of unevenly spaced data that lies on $t \in [t_0, t_f]$ is obtained as follows. First, the unevenly spaced data is interpolated to $N + 1$ evenly spaced points on the time interval $t \in [t_0, t_f]$. Second, these interpolated data points are reflected about t_f (excluding the points at t_0 and t_f) to create a sampling of an even function with period $2(t_f - t_0)$. The Fast Fourier Transform (FFT) is then utilized to calculate the first N Fourier coefficients (excluding a_0) of the even Fourier approximation of the data on the period $2(t_f - t_0)$. The choice of N is somewhat arbitrary so long as the resulting Fourier approximation reasonably describes the original set of unevenly spaced data. Figure 8 images (a) and (b) show, respectively, the even Fourier series approximations of unevenly spaced data alongside the corresponding jump function approximations for $N = \{10, 20, 40\}$.

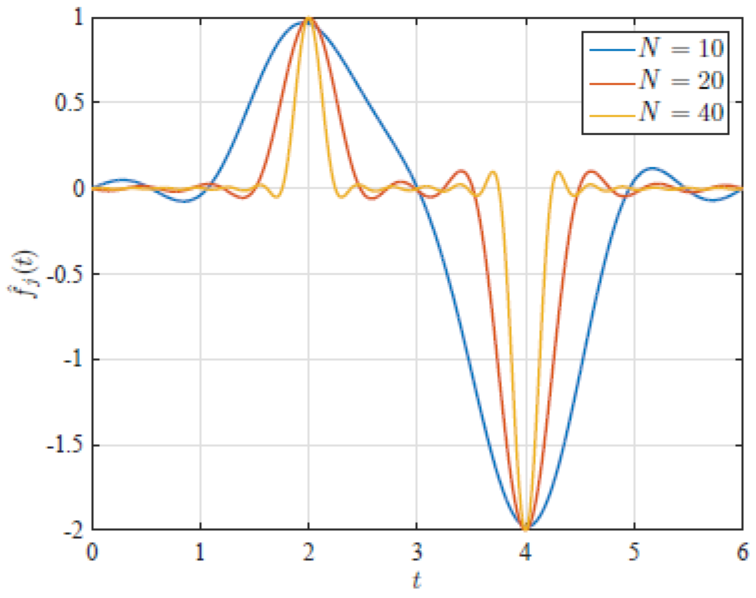
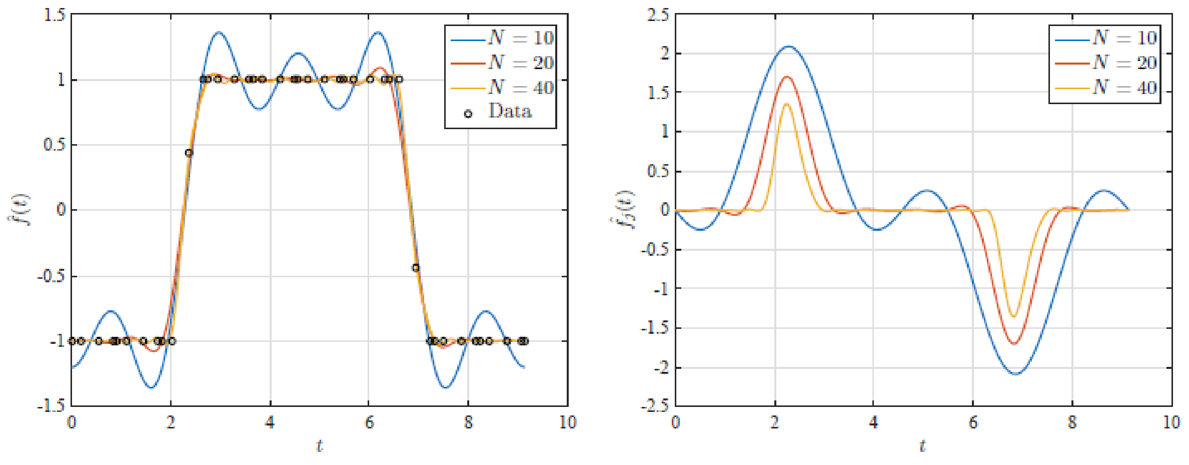


Figure 7. N-term approximation of jump function $f_j(t)$ given in Equation (47) using $N = \{10, 20, 40\}$. Note that the coefficients used in the jump function approximation correspond to an even Fourier approximation of funciton $f(t)$ defined in Equation (46).

Examining Figure 8, it is seen that the jump function approximation obtained using unevenly spaced data has similar features to the jump function approximation obtained in Figure 8 where the function $f(t)$ is known. Specifically, the global extrema of the jump function approximation shown in Figure 8 obtained using unevenly spaced data correspond closely with the locations of the discontinuities in the sampled function as is the case in Figure 8 where the function $f(t)$ is known. Moreover, these extrema locations do not tend to vary as N is increased. However, it is observed that the values of these extrema tend to decrease as N is increased. The decrease is due to the linear interpolation step when calculating the even Fourier series coefficients using the approach described previously. Despite this new drawback, these extreme points of each jump function approximation remain good estimates for the locations of jumps in the underlying function of the unevenly sampled data. The extrema of a jump function approximation $\hat{f}_j(t)$ is obtained by determining the zeros of the derivative of $\hat{f}_j(t)$, where the derivative of $\hat{f}_j(t)$ is given as

$$\frac{d\hat{f}_j(t)}{dt} = \sum_{k=1}^{N} \sigma\left(\frac{k}{N}\right)\frac{k\pi}{L}\left[a_k \cos\left(\frac{k\pi}{L}t\right) + b_k \sin\left(\frac{k\pi}{L}t\right)\right] \tag{48}$$



(a) Even Fourier approximations of unevenly spaced data.

(b) Jump function approximations of the unevenly spaced data in Fig. 8a.

Figure 8. N-term Fourier and jump function approximations of unevenly spaced data for $N = \{10, 20, 40\}$.

4.2.5 Mesh Refinement Method with Discontinuity Detection

The mesh refinement method used in this paper combines the hp-adaptive scheme of Ref. 45 with the approach described in Section 4.2.1 for approximating jump functions. This discussion is restricted to detecting discontinuities and does not provide a discussion of the mesh refinement method of Ref. 45.

Suppose that the NLP of Equations (30) through (33) arising from the LGR collocation method is solved on a mesh that has either been supplied (that is, an initial mesh) or a mesh that has been computed using a mesh refinement method (for example, the hp-adaptive method of Ref. 45). After solving the NLP on the initial mesh, the error is approximated using the error approximation method given in Section 4.1.4. For each mesh interval on the mesh where the maximum relative error tolerance, ϵ, is exceeded, the following discontinuity detection procedure is employed. First, for each component of the control, a jump function approximation is constructed using an even Fourier series approximation as described in Section 4.2.1. The global extrema of these jump function approximations are determined and are used as the basis for determining the existence of jump discontinuity. When a discontinuity is detected, the current mesh is refined by bracketing the discontinuity with three new mesh points (one at the estimated discontinuity location and two more surrounding the estimated discontinuity location). When no discontinuity is detected, the hp mesh refinement method of Ref. 45 is employed. The result of refining each mesh interval in the aforementioned manner leads to a new mesh. The optimal control problem is then approximated on this new mesh using the LGR collocation method given in Section 4.1.2 and the NLP of Equations (30) through (33) is solved. The process of constructing a new mesh using the hp-adaptive method of

Ref. 45 together with the aforementioned discontinuity detection method is repeated until the relative error tolerance ∈ is satisfied on every mesh interval.

4.2.5.1 Method for Obtaining Fourier Coefficients

Assume that at least one mesh interval exists on the current mesh for which the estimated relative error of the solution is larger than the relative error tolerance, ∈. Fourier coefficients corresponding to an even Fourier series approximation must then be calculated so that the jump function of each control component may be approximated. The necessary Fourier coefficients are generated from the set of estimated values of the control at the collocation points obtained by solving the NLP of Equations (30) through (33) on mesh M. The number of data points used to approximate the jump function can range from the data on a single mesh interval to the data on the entire mesh. It is desirable to use as many of the collocation points in the Fourier approximation as possible, because computing one set of Fourier coefficients for all K of the mesh intervals will be faster than producing K sets of Fourier coefficients with one set for each interval. Note, however, that the mesh fraction (ratio of the mesh interval time span to the total time span of the mesh) of individual mesh intervals may be widely different. Large differences between the mesh fractions cause the collocation data to be concentrated in some areas of the mesh and sparse in others, and such unevenly spaced data can result in a poor Fourier series approximation.

In this research, a grouping algorithm is developed such that mesh intervals are grouped together according to their mesh fractions. A group, Gg, is defined as a set of adjacent mesh intervals for which the following condition holds:

$$\frac{T_k - T_{k-1}}{\max\limits_{j\in\{K_g+1,\cdots,K_g+k_g\}}(T_j - T_{j-1})} \geq \rho_1, \forall k \in \{K_g + 1, \cdots, K_g + k_g\} \tag{49}$$

where

$$K_g = \sum_{i=1}^{g-1} k_i, g = 1, \cdots, G$$

and $\rho 1 \in [0, 1]$ is a user-defined threshold, kg are the number of mesh intervals in group Gg, G is the number of groups, $\sum_{i=1}^{G} k_i = \mathrm{K}$, and $\cup_{g=1}^{G} G_g = \{S1, \ldots, SK\}$.

Mesh intervals are grouped in the following manner. We start with one group containing all of the mesh intervals in the current mesh ($G_1 = \{S1, \ldots, SK\}$). If one or more of its members does not satisfy Equation (49), G_1 is split into G new groups, $G_1 = \{S1, \ldots, Sk1\}$, $G_2 = \{Sk1 + 1, \ldots, Sk1 + k2\}$, $G_G = \{SKG + 1, \ldots, SK\}$. Each new group contains kg adjacent mesh intervals, which either all satisfy or all do not satisfy Equation (49). Each of the new groups undergoes the same division process until Equation (49) is satisfied by all mesh intervals in each group.

In this research, $\rho_1 = 0.05$. Adjusting ρ_1 to be larger or smaller will ~~effect~~affect how many groups are created as well as the degree to which mesh interval's mesh fractions can vary within a group. For example, $\rho_1 = 1$ would cause each mesh interval to be assigned to its own distinct group if that particular mesh interval is not identical in length to its neighbors. Alternatively, $\rho_1 = 0.01$ results in each group containing mesh intervals with corresponding mesh fractions which are no more than two orders of magnitude apart.

After grouping is complete, we can produce an accurate even Fourier series approximation for any particular group. The data used in the Fourier approximation for a particular group, G_G, are the estimated control values at each of the collocation points corresponding to each mesh interval in the group as well as the endpoint of the final mesh interval in the group. The data is linearly interpolated to $N + 1$ evenly spaced points spanning $[T_{K_g}, T_{K_g} + k_g]$. The evenly spaced points (excluding the first and last points at T_{K_g} and $T_{K_g} + k_g$) are mirrored about $T_{K_g} + k_g$ to create a sampling of an even function with period 2($T_{K_g} + k_g - T_{K_g}$). The Fast Fourier Transform is then utilized to calculate the first N Fourier coefficients of the even approximation ($a_n = a_1, \ldots, a_N$, $b_n = 0$). The choice of N is somewhat arbitrary so long as the resulting Fourier approximation reasonably describes the control solution. Here, N is equal to the number of collocation points used in the even Fourier series approximation for the current group.

4.2.5.2 Method for Locating and Verifying Discontinuity Locations

Assume now that all of the mesh intervals on the current mesh have been divided into groups. Assume further that the maximum relative error tolerance, $\in$, is exceeded on a particular mesh interval, S_k, within a particular group, G_g. Assume once more that the first N Fourier coefficients of the even Fourier series approximation to the group's control collocation data have been calculated for each component of the control. The location of any existing jump discontinuity within S_k must be identified for each control component.

As a preliminary test, a jump in a particular control component, u, is deemed likely when the following criteria is met. There are two points, u_i^k and u_{i-1}^k, on the current mesh interval S_k in current group G_g that have the highest magnitude linear slope between them. The reasoning behind choosing u_i^k and u_{i-1}^k in this manner is due to the fact that the absolute value of the slope between two points on opposite sides of a jump discontinuity approaches infinity in the limit as those two points approach the discontinuity location from either side. Therefore, u_i^k and u_{i-1}^k are the most likely candidates to contain a jump discontinuity between them.

They have a relative difference

$$\Delta_r = \frac{u_i^k - u_{i-1}^k}{\left(\max_{k\in\{Kg+1,\cdots,Kg+kg\},i\in k}\left(u_i^k\right)\right) - \left(\min_{k\in\{Kg+1,\cdots,Kg+kg\},i\in k}\left(u_i^k\right)\right)} \tag{50}$$

where u_i^k is inclusive of the endpoint of the interval $u_i^k = u_i^{k+1}$. Note that division by zero in Equation (50) is possible only if u_i^k is constant for all $i \in k, k \in \{Kg + 1, \dots, Kg + kg\}$. In such a case, no jump discontinuity is likely. Therefore, we stop searching for jump discontinuities in that particular control component before the division by zero can occur. In all other cases, if the absolute value of Δ_r exceeds a user-set threshold $\rho_2 \in [0, 1]$, then the search for a jump discontinuity in that particular control component is continued.

The threshold applied to Δ_r in this research is $\rho_2 = 0.1$. Raising the value of ρ_2 helps limit the search for jump discontinuities to larger, more easily distinguishable jumps. The threshold also assists in avoiding unnecessary calculations when there are either no jump discontinuities or the jumps are too small to detect accurately.

Assuming that $\Delta_r \geq \rho_2$ for a particular control component, the location of the jump discontinuity and the value of the jump must be estimated. Earlier, in Section 6.1, it was shown that the location of the maximum or minimum of the jump function approximation, $\hat{f}_j(t)$, of Equation (41) can be a good approximation for the location and value of the jump in $f(t)$. We now seek to obtain an extremum of the control component's jump function approximation, denoted as $\hat{u}_j(t)$, within the current mesh interval.

An extremum of $\hat{u}_j(t)$ can be found by implementing Newton's method to find a zero for its derivative $\frac{d}{dt}\hat{u}_j(t)$. We use the midpoint between the times corresponding to u_i^k and u_{i-1}^k (the same points used to calculate Δ_r in Equation 50) as our initial guess. Once a zero of $\frac{d}{dt}\hat{u}_j(t)$ is obtained, the second derivative of the jump function approximation $\frac{d^2}{dt^2}\hat{u}_j(t)$ is used to verify that the concavity matches the jump, because we should converge to a local max if the jump is positive and a local min if the jump is negative. The process of locating an extremum of $\frac{d}{dt}\hat{u}_j(t)$ is done twice; once using all N Fourier coefficients, and once more using the first N (rounded up to the nearest integer) coefficients. The values and locations of each extremum are stored for further analysis. In the case where we are unable to converge to an extremum within S_k and with the correct concavity, the search for a jump discontinuity is ceased for that particular control component.

An example of locating an extremum of $\hat{u}_j(t)$ is depicted in Figure 9 along with visualizations for the first and second derivatives of $\hat{u}_j(t)$. As can be seen, the initial guess is quite close to the extremal point of $\hat{u}_j(t)$, which results in rapid convergence.

Let $\hat{u}_{j,N}(t)$ denote $\hat{u}_j(t)$ when N terms are used in the approximation. Assume now that we have located two respective extrema for $\hat{u}_{j,N}(t)$ and $\hat{u}_{j,\frac{N}{2}}(t)$. Let the locations of each extremum be denoted t_N^* and $t_{\frac{N}{2}}^*$. Three final criteria must be met in order to declare t_N^* a jump discontinuity location. The first criterion is

$$\frac{\left|\hat{u}_j(t_N^*)\right|}{\left(\max_{k\in\{K_g+1,\cdots,K_g+k_g\},i\in k}(u_i^k)\right)-\left(\min_{k\in\{K_g+1,\cdots,K_g+k_g\},i\in k}(u_i^k)\right)} \geq \rho 2 \tag{51}$$

where ρ_2 is the same threshold used previously on Δ_r.

The second and third criteria are used to verify that the jump function approximation is reliable. They are expressed, respectively, as

$$\frac{\left|\hat{u}_{j,N}(t_N^*)-\hat{u}_{j,\frac{N}{2}}\left(t_{\frac{N}{2}}^*\right)\right|}{\left|\hat{u}_{j,N}(t_N^*)\right|} \geq \rho 3 \tag{52}$$

and

$$\frac{\left|t_N^*-t_{\frac{N}{2}}^*\right|}{T_{K_{g+k_g}}-T_{K_g}} \geq \rho 4 \tag{53}$$

where T_{K_g} and $T_{K_{g+k_g}}$ are the endpoints of the current group and ρ_3 and ρ_4 are user-defined thresholds. In this research, $\rho_3 = 0.5$ and $\rho_2 = 0.02$. If the criteria of Equation (51) through (53) are all satisfied, then t_N^* is considered a reliable estimate for the location of a jump discontinuity in the current control component under investigation.

4.2.5.3 Mesh Refinement Actions

Assume now that a discontinuity has been detected on mesh interval S_k and its location identified to be t_N^*. The mesh interval is refined by splitting the mesh at t_N^* and at two adjacent points $t_N^* + \Delta t$ and $t_N^* - \Delta t$. The choice for Δt should be some function which scales with the length of the mesh interval $(T_k - T_{k-1})$. In this research, we choose

$$\Delta t = \frac{1.2}{N_k(T_k - T_{k-1}}\left(2\frac{\left|\hat{u}_{j,N}(t_N^*) - \hat{u}_{j,\frac{N}{2}}\left(t_{\frac{N}{2}}^*\right)\right|}{\left|\hat{u}_{j,N}(t_N^*)\right|}+1\right)$$

The adjacent mesh point located at $(t_N^* + \Delta t$ is omitted from the new mesh if $(t_N^* + \Delta t \geq T_k$. Similarly, the adjacent mesh point at $(t_N^* - \Delta t$ is omitted from the new mesh if $(t_N^* - \Delta t \leq T_{k-1}$. Should two separate discontinuities be identified on the same interval and their respective bracketing mesh points overlap, the overlapping mesh points are omitted from the new mesh and the midpoint between the two discontinuities is used instead.

On subsequent mesh iterations, the following caveat is used to keep the number of mesh intervals small (and thereby the size of the NLP as well). If a mesh interval that is known to contain a discontinuity, identified on the previous mesh $M-1$, does not meet the mesh error tolerance on the current mesh M, and that mesh interval is identified as having a discontinuity again; shrink the three original bracketing mesh points on mesh M around the new estimated discontinuity location and add one collocation point to the adjacent mesh intervals whose mesh fractions grow as their neighbor's mesh fraction shrinks. Recycling mesh points in this manner rather than creating three new mesh points each time a discontinuity is re-identified helps limit the growth of the NLP.

4.2.6 Mesh Refinement Algorithm

A summary of our mesh refinement algorithm appears below. The hp-adaptive scheme of Ref. 45 provides the shell to which our discontinuity detection algorithm is added. The mesh number is denoted by M and is incremented by one with each loop of the algorithm. M also corresponds to the number of mesh refinement iterations, because M is initialized at 0. This algorithm terminates when either the error tolerance is satisfied in Step 5 or when M reaches a prescribed limit M_{max}.

Mesh Refinement with Discontinuity Detection

Step 1: Set $M = 0$ and supply initial mesh, $S = \bigcup_k^K S_k = [-1, +1]\,, where\ \bigcap_k^K S_k = \emptyset$

Step 2: Solve Radau collocation NLP of Equations (30) through (33) on mesh M.

Step 3: Group adjacent mesh intervals such that Equation (49) is satisfied for each group G.

Step 4: Compute scaled error $e_{max}^{(k)}$ in $S_k, k = 1, \cdots, K$, using Equation (38).

Step 5: If $e_{max}^{(k)} \leq \in$ for all $k \in [1, \cdots, K]$ or $M > M_{max}$, then quit. Otherwise, proceed to Step 6.

Step 6: For every mesh interval $S_k, k \in [1, \cdots, K]$,

(a) if $e_{max}^{(k)} \leq \in$, locate any jump discontinuities in the control components on S_k and bracket them using the method of Section 6.5. If no discontinuities are found, refine using h or p as normal.

(b) if $e_{max}^{(k)} \leq \in$, determine if the mesh size can be reduced using the method of Ref. 45

Step 7: Increment M by one and return to Step 2.

4.2.7 Results and Discussion

The following example problems from the open literature are used to test the performance of the algorithm described in Section 4.2.6 when compared with the base algorithm of Ref. 45, which does not actively search for jump discontinuity locations. All solutions to the example problems contain jump discontinuities in at least one component of the control. All computation times (CPU times) shown are based on an average time obtained by solving each problem ten times.

The computations were performed using a MacBook Pro with a 2.8 GHz Intel Core i7 processor and 16 GB of RAM.

4.2.7.1 Minimum Control Effort Landing on the Moon

Consider the following optimal control problem. Minimize the objective functional

$$\min J = \int_0^{t_f} u(t)dt \tag{54}$$

subject to the dynamic constraints and boundary conditions

$$\begin{aligned} \dot{h}(t) = v(t), \left(h(0), h\left(t_f\right)\right) = (10,0) \\ \dot{v}(t) = -g + u(t), \left(v(0), v\left(t_f\right)\right) = (-2,0) \end{aligned} \tag{55}$$

and the control inequality constraint

$$0 \leq u(t) \leq 3 \tag{56}$$

where $(h(t), v(t)) \in R^2$ is the state, $u(t) \in \mathbb{R}$ is control, and g is a constant. The optimal control problem given in Equations (54) through (56) was solved using a mesh refinement relative error tolerance $\in = 10^{-6}$. It is known that the solution to the optimal control problem defined in Equations (54) through (56) has a bang-bang optimal control with a single control discontinuity at $t \approx 1.41$ where the control switches from its minimum allowable value to its maximum allowable value. The control solution is shown in Figure 10. The mesh histories using the hp-adaptive method of Ref. 45 without and with the discontinuity method developed in this paper are shown in Figure 11, while the corresponding final mesh characteristics and CPU times are shown in Table 1.

When discontinuity detection is applied, the problem solves on the second mesh, needing only a single mesh refinement iteration. Without discontinuity detection, however, the interval containing the discontinuity is cut into thirds on the first mesh refinement iteration, then consecutively halved twice on the second and third iterations of mesh refinement before the mesh error tolerance, ∈, is satisfied. Although the method of Ref. 45 does a good job of dividing only the nonsmooth mesh interval on each mesh refinement iteration, it divides in an evenly spaced manner without regard to the location of the discontinuity. As a result, as opposed to locating the discontinuity and dividing the interval using the method of Section 4.2.5.3, extra mesh refinement iterations are required. Moreover, as seen in Table 1, both the final mesh and the required CPU time without discontinuity detection are larger than the final mesh and the CPU time required with discontinuity detection.

Table 1. Final Mesh Characteristics for and Computation Times for Example 6.7.1.

	Without Discontinuity Detection	With Discontinuity Detection
Number of Meshes	4	2
Number of Mesh Intervals	7	6
Number of Collocation Points	23	22
CPU Time	0.253	0.154

4.2.7.2 Minimum Time Reorientation of a Robot Arm

Consider the following optimal control problem. Minimize the objective functional

$$\min J = t_f \tag{57}$$

subject to the dynamic constraints

$$\begin{array}{cc}
 & \left(x_1(0), x_1(t_f)\right) = (9/2, 0) \\
\dot{x}_1(t) = x_2(t), & \left(x_2(0), x_2(t_f)\right) = (0,0) \\
\dot{x}_2(t) = u_1(t)/L, & \\
\dot{x}_3(t) = x_4(t), & \left(x_3(0), x_3(t_f)\right) = (0, 2\pi/3) \\
\dot{x}_4(t) = u_2(t)/I_\theta, & \left(x_4(0), x_4(t_f)\right) = (0,0) \\
\dot{x}_5(t) = x_6(t), & \\
\dot{x}_6(t) = u_3(t)/I_\emptyset, & \left(x_5(0), x_5(t_f)\right) = (\pi/4, \pi/4) \\
 & \left(x_6(0), x_6(t_f)\right) = (0,0)
\end{array} \tag{58}$$

and the control inequality constraints

$$|u_i(t)| \leq 1, (i = 1,2,3) \tag{59}$$

where $(x_1(t),\ x_2(t), x_3(t), x_4(t), x_5(t),\ x_6(t)) \in \mathbb{R}^6$ is the state, $(u_1(t), u_2(t), u_3(t)) \in \mathbb{R}^3$ is the control, $I_\phi = ((L - x_1(t))^3 + x_1^3\,(t))/3$, and $I_\theta = I_\phi\, sin^2(x_5(t))$. The optimal control problem given in Equations (57) through (59) was solved using a mesh refinement relative error tolerance[1]$\epsilon = 10^{-8}$. It is known that each component of the optimal control for the example given by Equations (57) through (59) has a bang-bang structure with a total of five control discontinuities located at $t \approx \{2.28, 2.80, 4.57, 6.35, 6.86\}$. The control solution for the example given by Equations (57) through (59) is shown in Figure 12. The mesh histories using the hp-adaptive method of Ref. 45 without and with the discontinuity method developed in this paper are shown in Figure 13, while the corresponding final mesh characteristics and computation times are shown in Table 2.

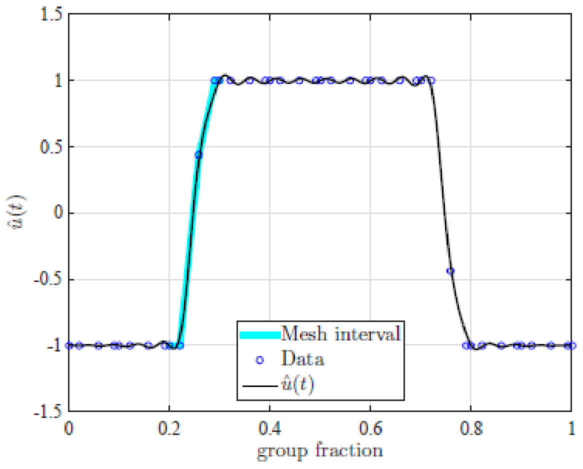
Table 2. Final Mesh Characteristics and Computation Times for Example 6.7.2.

	Without Discontinuity Detection	With Discontinuity Detection
Number of Meshes	7	4
Number of Mesh Intervals	33	22
Number of Collocation Points	182	118
CPU Time	1.780	0.909

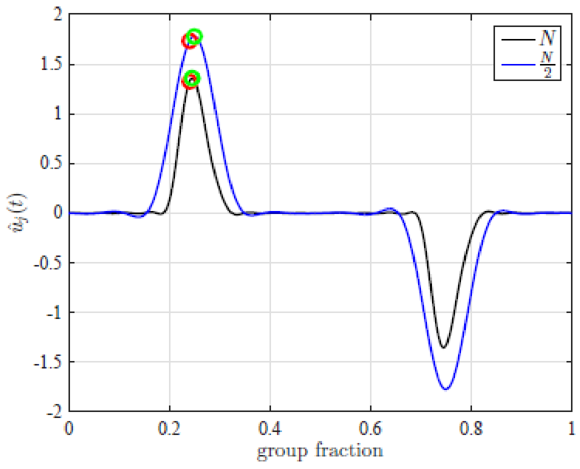
The discontinuity detection algorithm of this paper was able to successfully detect all five jump discontinuities on the first mesh refinement iteration. As can be seen in Figure 13(b), the mesh point triplets that bracket each discontinuity are successfully reassigned with each mesh refinement iteration. The reassignment of discontinuity bracketing mesh points has the visual effect of each mesh point triplet shrinking around the discontinuity location with each successive mesh refinement iteration. Reassigning mesh points in this manner has the desirable effect of keeping the number of mesh intervals on each new mesh smaller than it would have been had the mesh points not been reassigned. Table 2 indicates a more efficient use of mesh points and collocation points as well as a faster convergence to the solution when comparing the algorithm of this paper to the algorithm without jump discontinuity detection. In fact, the average computation time is nearly halved.

4.2.8 Conclusions

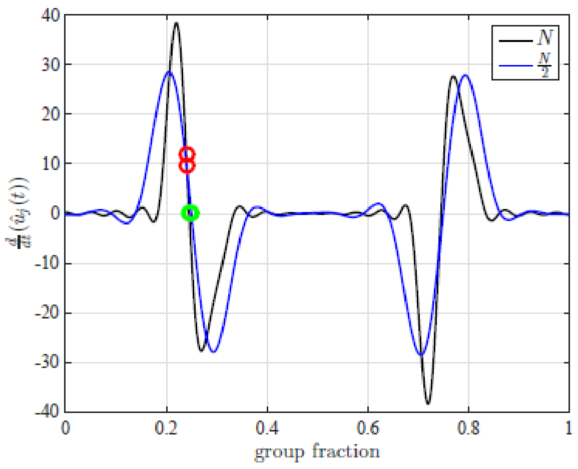
A mesh refinement method for optimal control has been developed that can detect jump discontinuities in control components. A jump function was defined and a method for approximating a jump function using an even Fourier series approximation was developed. The locations of discontinuities in the solution of an optimal control problem were then approximated by locating the extrema of the jump function approximation. A mesh refinement method was then described that employed discontinuity detection together with a previously developed adaptive mesh refinement method. The mesh was refined by bracketing the locations of jump discontinuities. The method was applied to two example, and the results indicate that a smaller final mesh size, fewer mesh refinement iterations, and less computation time were needed to solve the optimal control problem using collocation by including the discontinuity detection method when compared with excluding the discontinuity detection method.



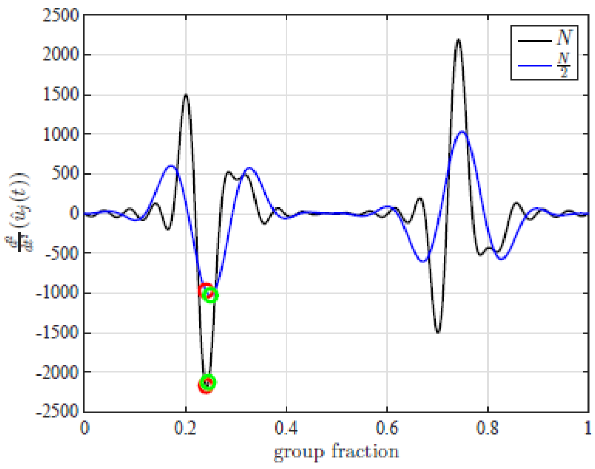
(a) Even Fourier approximation of control collocation data.



(b) N-term approximation of $u_j(t)$.



(c) First derivative of $\hat{u}_j(t)$.



(d) Second derivative of $\hat{u}_j(t)$.

Figure 9. Example of the mesh algorithm being implemented to find the maximum of the jump function on the current mesh interval. The red circles correspond to the initial guess, and the green circles correspond to the maximum of the jump function. Convergence to the maximum was achieved using Newton's method. Note that the x-axis is labeled "group fraction," which corresponds to the ratio of $T_k - T_{k-1}$ over $T_{k_g} - T_{k_1-1}$.

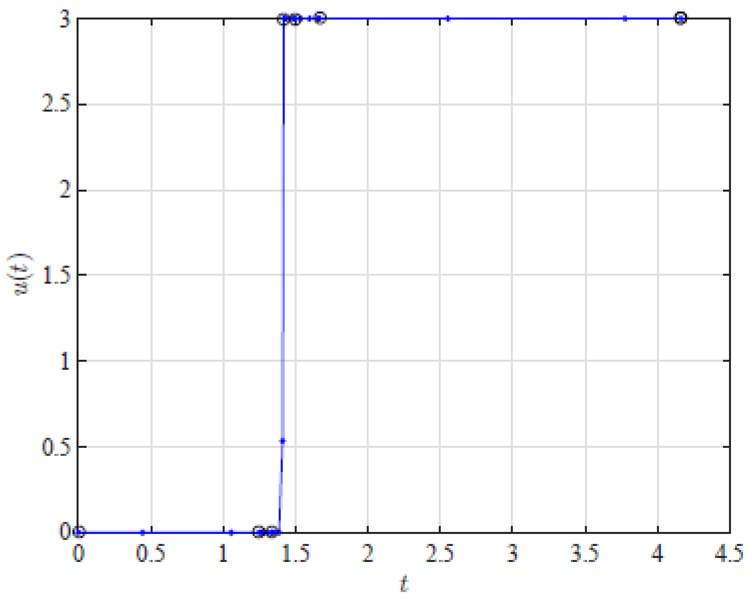
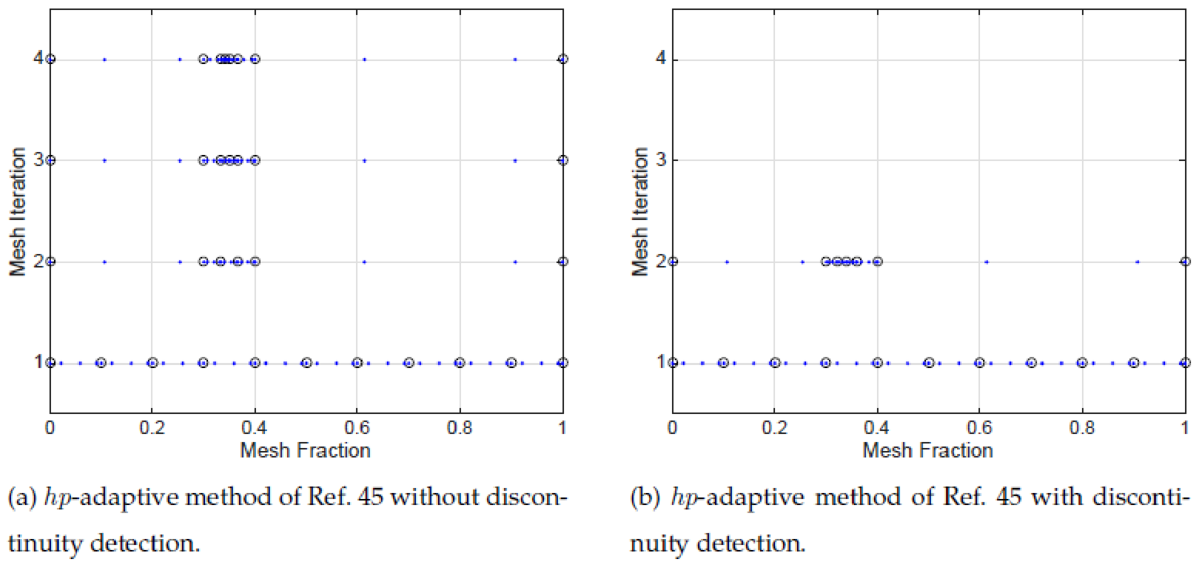


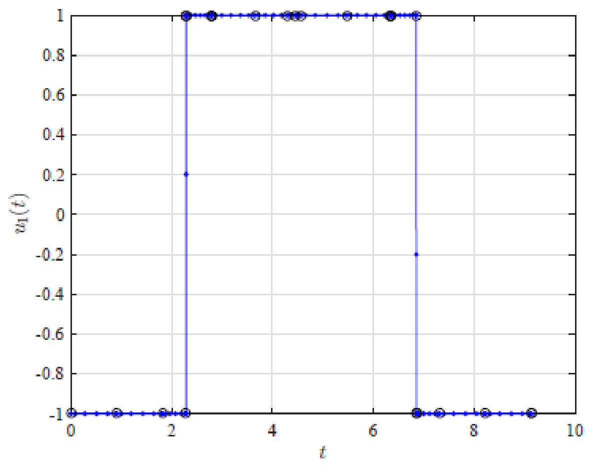
Figure 10. Control solution for Example 6.7.1.

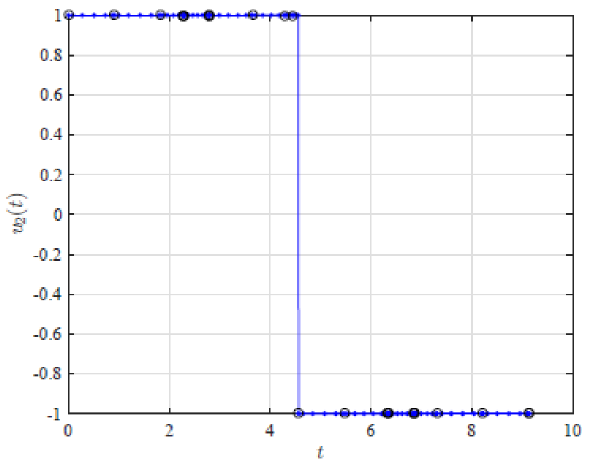


(a) hp-adaptive method of Ref. 45 without discontinuity detection.

(b) hp-adaptive method of Ref. 45 with discontinuity detection.

Figure 11. Mesh histories using *hp*-adaptive method of Ref. 45 without and with discontinuity detection for Example 6.7.1.


(a) Control component $u_1(t)$.


(b) Control component $u_2(t)$.

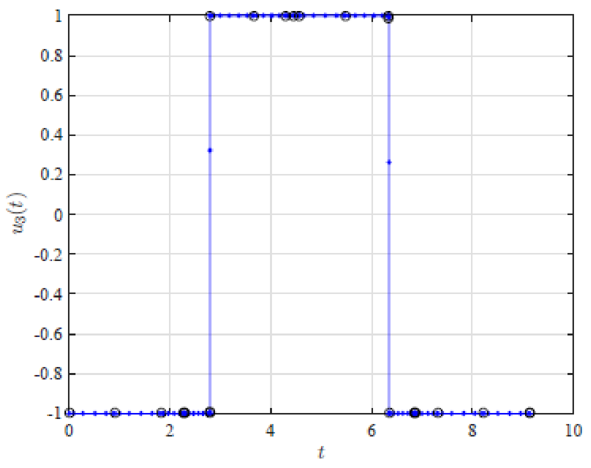

(c) Control component $u_3(t)$.

Figure 12. Control solution for Example 6.7.2.

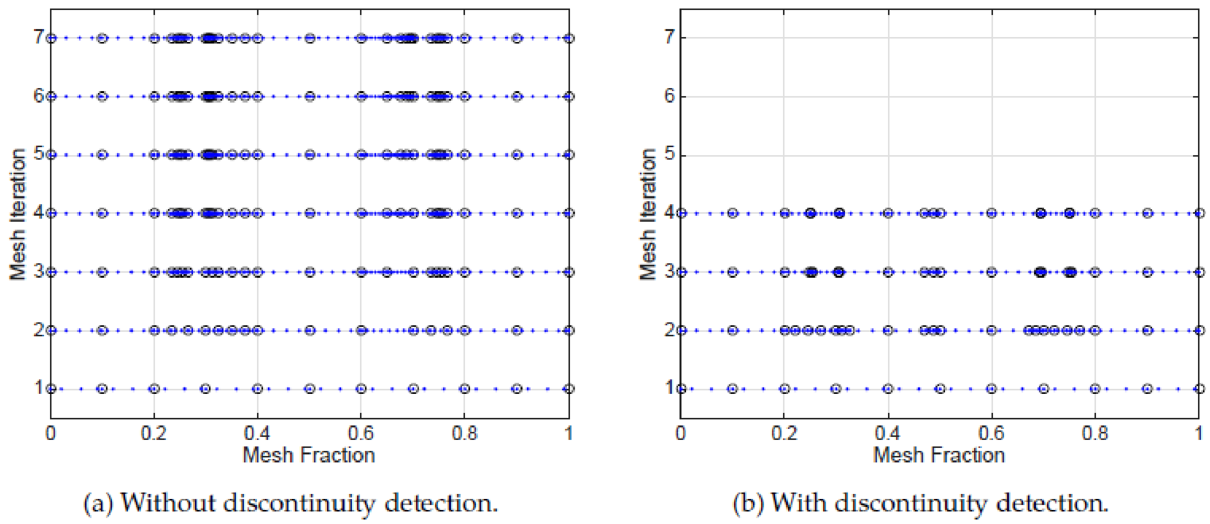


Figure 13. Mesh histories using *hp*-adaptive method of Ref. 45 without and with discontinuity detection for Example 6.7.2.

4.3 Performance of Various hp-Adaptive Methods

This section provides a summary of the performance of various hp-adaptive mesh refinement methods that have been developed over the past several years. The performance summary provides a way to better understand the behavior of the various hp-adaptive methods. Each mesh refinement method that is part of this summary employs a different approach to decide how to refine the mesh at each mesh refinement iteration. An overview of the approach used in each method is described and is followed by a comparison of the performance of the different methods. It is noted that the method of Ref. [46] is not included in the performance comparisons because this method is still being finalized.

4.3.1 Methods Used in Performance Analysis of hp-Adaptive Methods

The following four methods are used as part of the hp-adaptive performance summary. First, the hp-adaptive method of Ref. [41] estimates the curvature of the solution in the mesh interval. If $e\frac{(k)}{max}$ is smaller in each mesh interval than the relative error accuracy tolerance, $\in$, then no additional action is required. If $e\frac{(k)}{max} > \in$, one of two actions may take place: increase the approximating polynomial degree in the mesh interval or divide the mesh interval. For intervals where $e\frac{(k)}{max} > \in$, a criteria is used to determine whether the interval needs more collocation points or should be divided into subintervals. The criteria in this method is based on the ratio of the maximum and mean curvature of the state in each interval. If the ratio is greater than or equal to the user-specified threshold, r_{max}, the mesh interval is divided. If the ratio is less than r_{max}, the degree of the approximating polynomial is increased.

Next, the hp-adaptive method of Ref. [43] is a p, then h method. The polynomial degree is increased until the maximum degree is reached; then, the interval is divided. If $e\frac{(\mathrm{k})}{max}$ is smaller in each mesh interval than the relative error accuracy tolerance, ∈, then no additional action is required. If $e\frac{(\mathrm{k})}{max} >$∈, one of two actions may take place: increase the approximating polynomial degree in the mesh interval or divide the mesh interval and decrease the approximation polynomial degree. First, the method tries to add collocation points in the mesh to increase the approximating polynomial degree. The number of points added, P_q, is dependent on the current number of collocation points, N_q, $e\frac{(\mathrm{k})}{max}$, and ∈. If $N_q + P_q \leq N_{max}$, then the points are added. If $N_q + P_q > N_{max}$, then the mesh interval is divided. The number of collocation points wanted to approximate the polynomial in one mesh interval, $N_q + P_q$, is split evenly over the subintervals such that the number of collocation points in each subinterval is N_{min}.

Next, the hp-adaptive method of Ref. [44] estimates the smoothness of the solution in the mesh interval. If $e\frac{(\mathrm{k})}{max}$ is smaller in each mesh interval than the relative error accuracy tolerance, ∈, then no additional action is required. If the mesh iteration has $e\frac{(\mathrm{k})}{max} >$∈ in any mesh interval, one of two actions may take place in each mesh interval: collocation points are added to the mesh interval (increasing the approximating polynomial degree) or the mesh interval is divided. If mesh interval has $e\frac{(\mathrm{k})}{max} \leq$ ∈, the mesh size is reduced or maintained. For intervals where $e\frac{(\mathrm{k})}{max} >$∈, a criteria is used to determine whether the interval needs more collocation points or should be divided into subintervals. The criteria in this method is based on the ratio of current and previous meshes' local maxima of the second derivative of the state on the interior of the mesh interval. If the ratio is greater than or equal to the user-specified threshold, $\bar{R}$, for any of the state components, the mesh interval is divided. Each subinterval has the same number of collocation points as the previous interval before being divided. If the ratio is less than the threshold for all state components, the degree of the approximating polynomial is increased by adding collocation points. If $e\frac{(\mathrm{k})}{max} \leq$ ∈ in a mesh interval, the mesh size can be reduced or maintained. First, the method looks to reduce the degree of the approximating polynomial in each mesh interval. Using a power series representation of the polynomial approximation, negligible terms are found and used to determine the reduction in the degree of the approximating polynomial. After determining the polynomial degree in each mesh interval, adjacent mesh intervals can be merged. Merging of mesh intervals occurs when the polynomial degree of adjacent intervals is the same and the maximum relative error in the difference of the polynomials is smaller than ∈ where ∈ is the mesh refinement accuracy tolerance.

Next, the hp-adaptive method of Ref. [45] follows closely with the hp-adaptive method of Ref. [44]. The major differences in methodology are the criteria for determining whether a mesh interval is smooth or nonsmooth and the method for determining the number of collocation points and subintervals to add to a mesh interval. The hp-adaptive method of Ref. [44] uses the absolute value of the ratio of current and previous interior local maxima of the second derivative of the state while the hp-adaptive method of Ref. [45] uses the decay rate of the Legendre polynomial coefficients. Otherwise, the selection decision process is the same as the hp-adaptive method of Ref. [44].

4.3.2 Examples

Each example problem includes the problem statement, GPOPS-II setup, results, including the state and control solutions, summary of key computational characteristics, and mesh refinement histories, and a discussion of the results. In order to produce statistically relevant results, each example problem was run 20 times with each mesh refinement method. All results shown in this paper were obtained using MATLAB optimal control software GPOPS-II [48] using the NLP solver IPOPT [52]. The required first and second derivatives required by IPOPT were computed using the built-in sparse first and second finite-differencing method. All computations were performed on a 3.5 GHz Intel Core Xeon E5-2637V4 Dell Precision Tower 7910 running Linux with 192GB RAM and MATLAB R2017b.

4.3.3 Brachistochrone Problem

Minimize the cost functional:

$$J = t_f$$

subject to dynamic constraints

$$\begin{aligned} \dot{x} &= v \sin u \\ \dot{y} &= v \cos u \\ \dot{v} &= g \cos u \end{aligned}$$

and boundary constraints

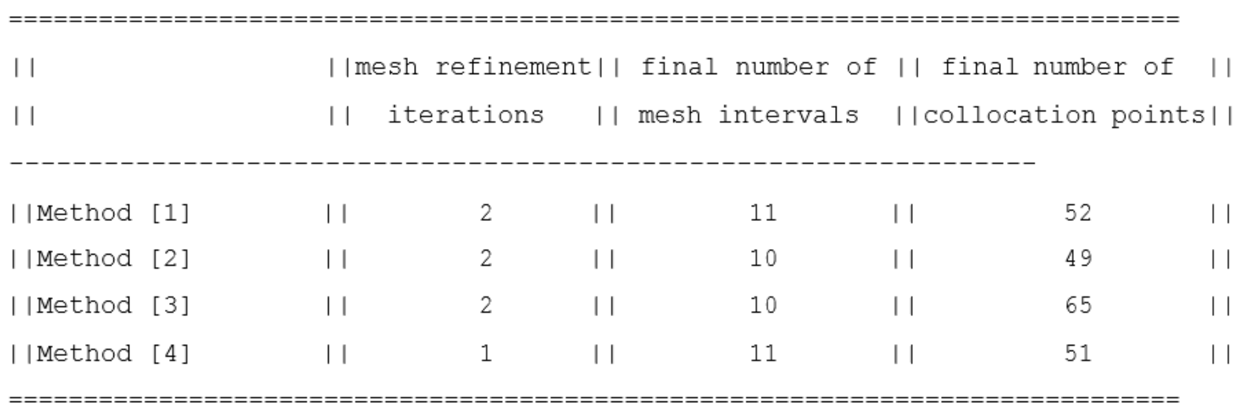
$$\begin{aligned} x(0) &= 0, \quad & x(t_f) &= 2 \\ y(0) &= 0, \quad & y(t_f) &= 2 \\ v(0) &= 0, \quad & v(t_f) &= Free \end{aligned}$$

4.3.3.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier   = 'sparseCD'
       +- derivative level = 'second'
       +- dependencies      = 'sparseNaN'
scales
       +- method = 'none'
method = 'RPM-Differentiation'
mesh
       +- tolerance  = 1.000000e-08
       +- max iterations = 50
       +- colpointsmin      =      3
       +- colpointsmax      = 20
initial mesh characteristics
+- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = mumps
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

4.3.3.2 Results

Table 3. Key performance measures for the brachistochrone problem.



```
=============================================================================
||                 ||mesh refinement|| final number of || final number of  ||
||                 ||  iterations   || mesh intervals  ||collocation points||
-------------------------------------------------------------------
||Method [1]       ||        2      ||        11       ||         52       ||
||Method [2]       ||        2      ||        10       ||         49       ||
||Method [3]       ||        2      ||        10       ||         65       ||
||Method [4]       ||        1      ||        11       ||         51       ||
=============================================================================
```

4.3.3.3 Discussion

The solution to this classic control problem when plotted in the x-y plane is the shape of a cycloid, which is smooth over all mesh intervals with no rapid changes, as seen in Figure 14. Due to the smoothness of the states and control, all of the mesh refinement methods are able to solve the problem with similar mesh sizes and within one mesh iteration of one another. The hp-adaptive method of Ref. [45] has the fastest time due to only having to refine the mesh once. The hp-adaptive method of Ref. [44] performed the worst, likely due to the addition of extra collocation points in comparison to the other methods.

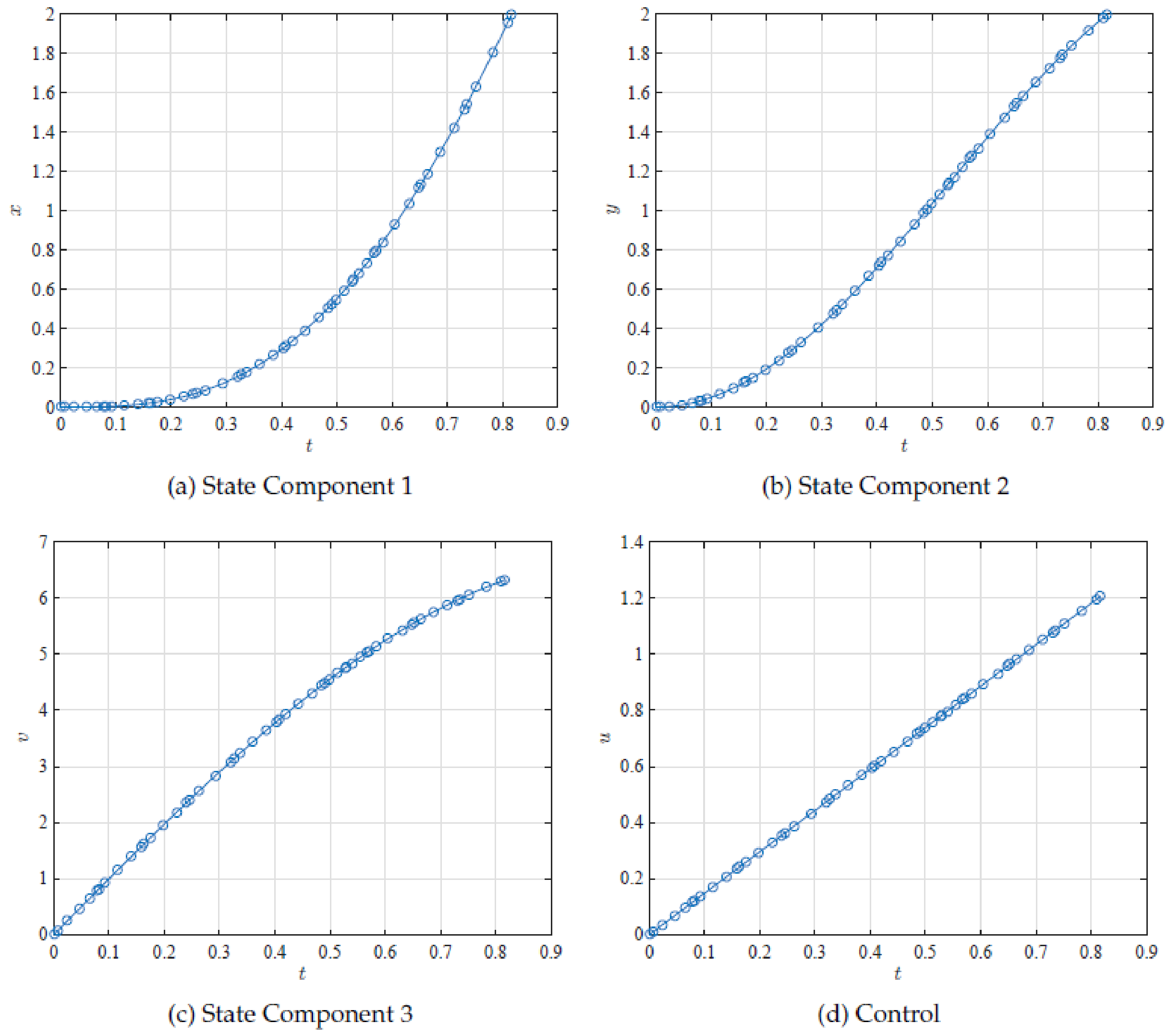


Figure 14. Problem state and control solution.

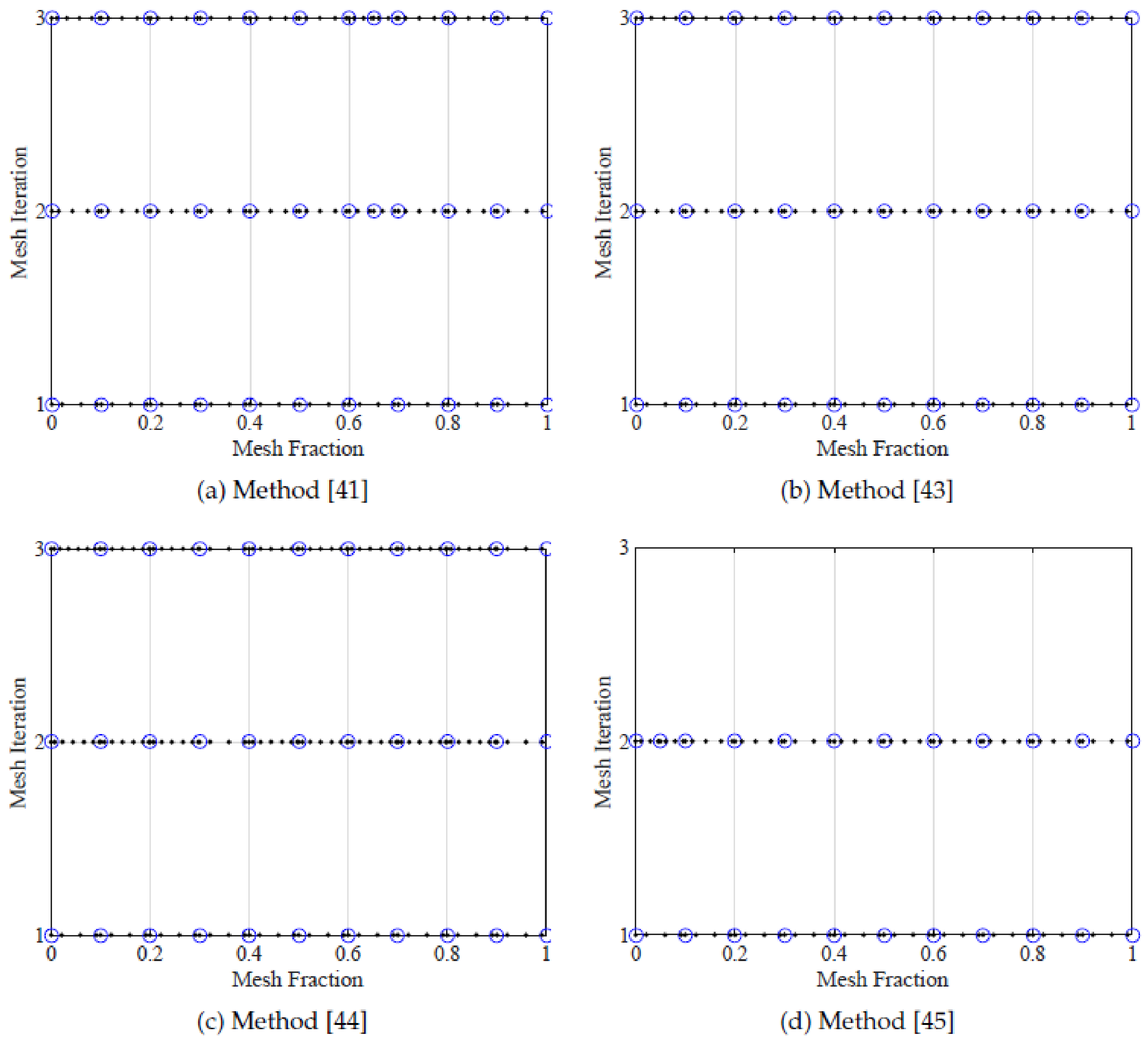


Figure 15. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

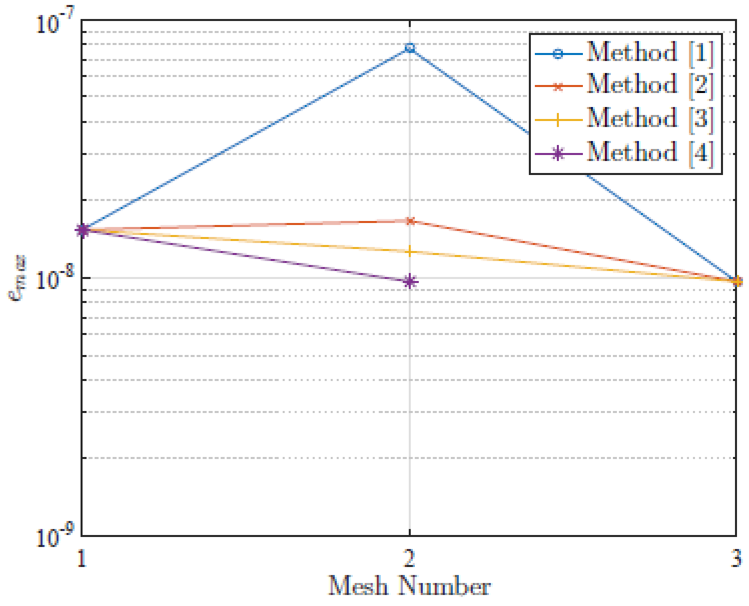


Figure 16. Mesh maximum relative error for each mesh.

4.3.4 Bryson-Denham Problem

Minimize the cost functional

$$J = \frac{1}{2}\int_0^1 u^2 dt$$

subject to the dynamic constraints

$$\begin{aligned}\dot{x} &= v \\ \dot{v} &= u\end{aligned}$$

the boundary conditions

$$\begin{aligned}x(0) = x(1) &= 0 \\ v(0) = -v(1) &= 1\end{aligned}$$

and the state inequality path constraint

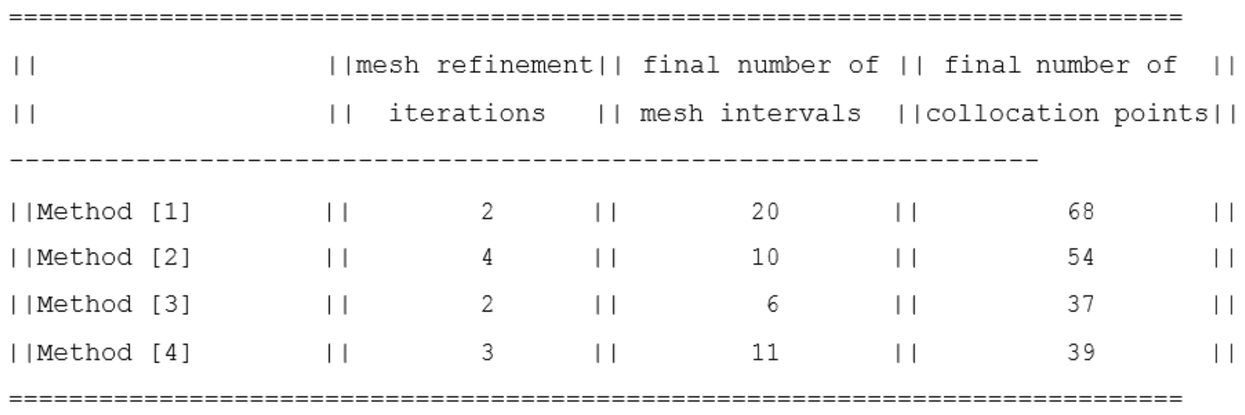
$$x \leq l$$

4.3.4.1 GPOPS-II Setup

```
setup details: derivatives
       +- supplier = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies = ’sparseNaN’
scales
       +- method = ’none’
method = ’RPM-Differentiation’
mesh
       +- tolerance = 1.000000e-08
       +- max iterations = 20
       +- colpointsmin = 3
       +- colpointsmax = 20
initial mesh characteristics
       +- Phase 1
       * fraction = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver = ipopt
       +- linear solver = ma57
       +- tolerance = 1.000000e-10
       +- max iterations = 2000
display level = 0
```

4.3.4.2 Results

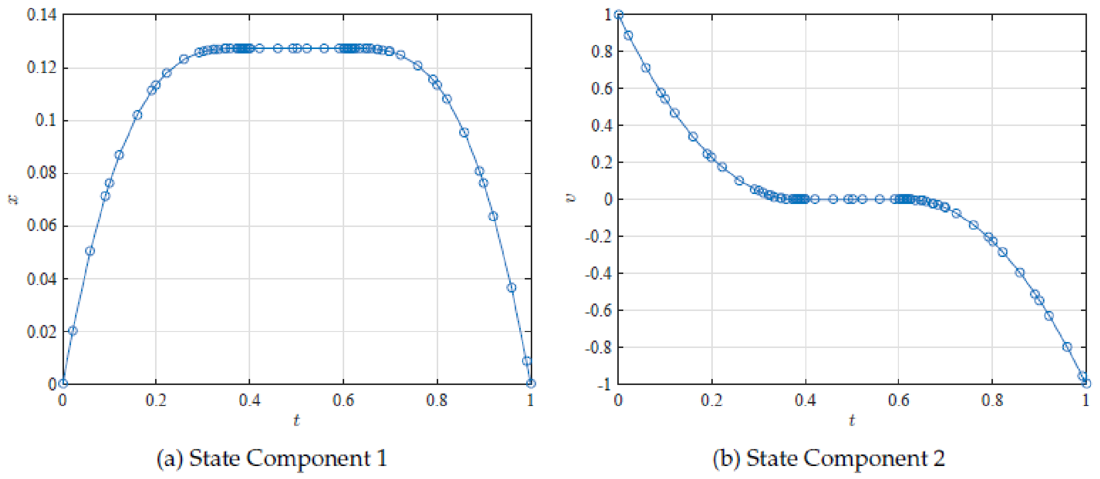
Table 4. Key performance measures for the bryson denham problem.



```
===========================================================================
||                ||mesh refinement|| final number of || final number of  ||
||                ||  iterations   || mesh intervals  ||collocation points||
------------------------------------------------------------------
||Method [1]      ||       2       ||       20        ||        68        ||
||Method [2]      ||       4       ||       10        ||        54        ||
||Method [3]      ||       2       ||        6        ||        37        ||
||Method [4]      ||       3       ||       11        ||        39        ||
===========================================================================
```

4.3.4.3 Discussion

Figure 17(c) shows two discontinuities in the first derivative of the control at approximately 0.333 and 0.667 seconds. Each mesh refinement's actions around this discontinuity can be seen in Figure 18. Each mesh refinement method is able to take refinement action near the discontinuities on the first mesh refinement, seen in Figure 18; however, because no mesh refinement method identifies the position of the discontinuity, the intervals where the discontinuities exist are either divided into equal subintervals or the polynomial degree of the interval is increased without dividing the interval. The hp-adaptive methods of Ref. [41] and Ref. [45] divide the mesh around the discontinuities, while the hp-adaptive methods of Ref. [43] and Ref. [44] increase the degree of the approximating polynomial. The hp-adaptive method of Ref. [44] is slightly faster than the hp-adaptive method of Ref. [41] due to the ability to reduce the mesh size on adjacent mesh intervals. The hp-adaptive method of Ref. [43] never divides the mesh intervals, able to approximate the derivative change in the control by just increasing the degree of the approximating polynomial. However, this takes more mesh iterations than the other methods, leading to the slowest cpu time, because the surrounding mesh intervals are not merged like the hp-adaptive method of Ref. [44].



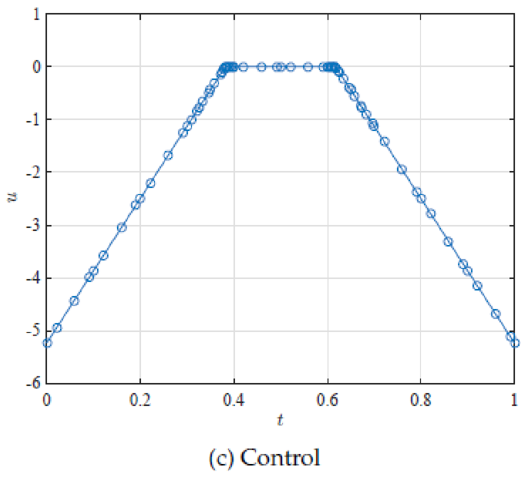
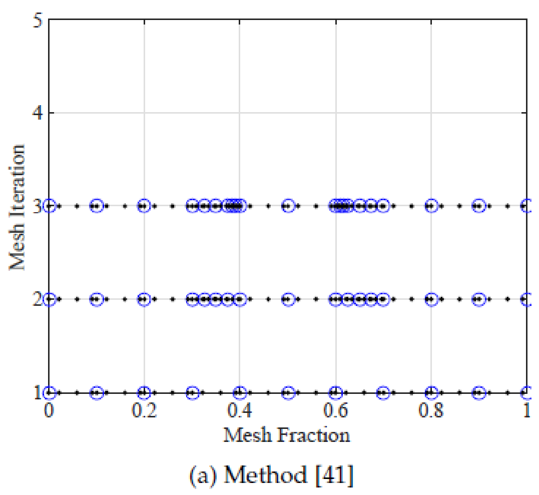
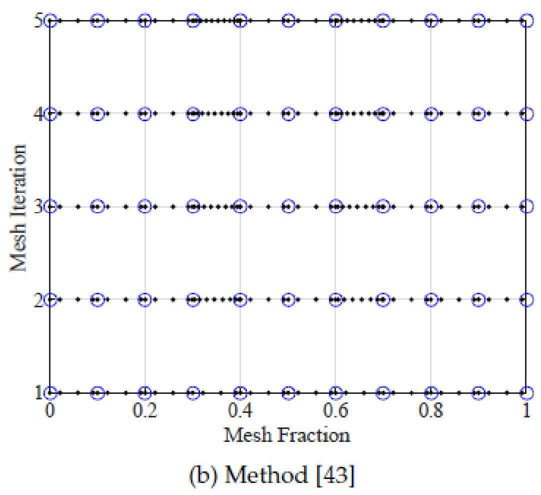
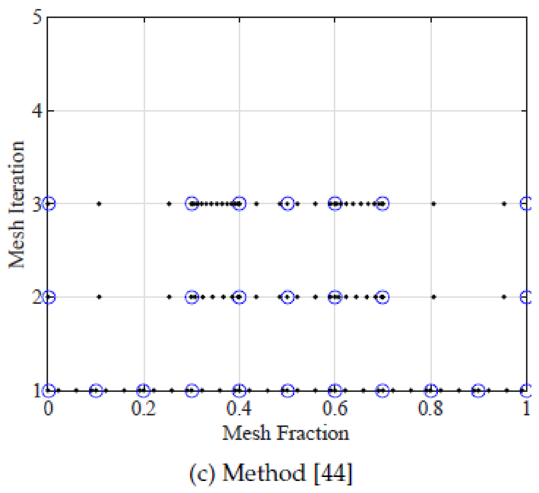


Figure 17. Problem state and control solution.







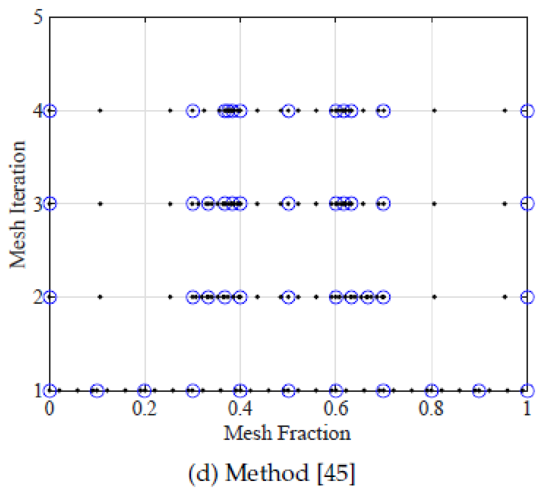


Figure 18. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

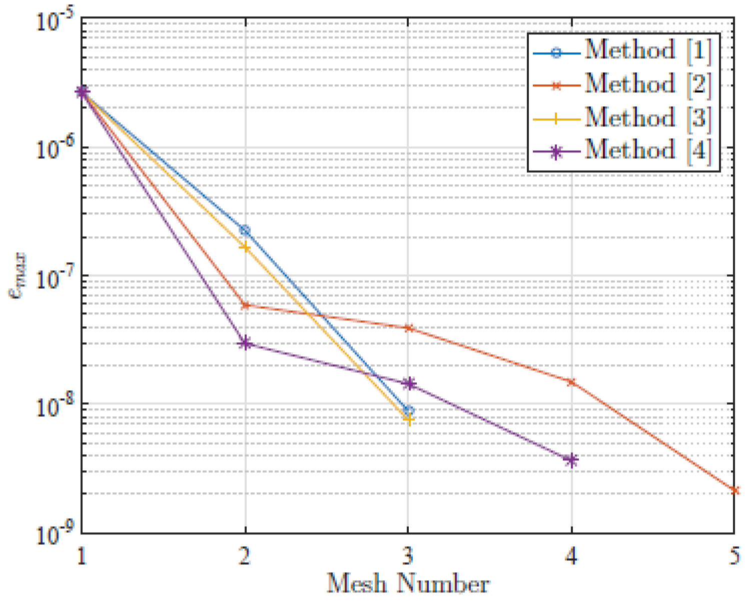


Figure 19. Mesh maximum relative error for each mesh.

4.3.5 Bryson Minimum Time-To-Climb Problem

Minimize the cost functional:

$$J = t_f$$

subject to the dynamic constraints

$$\begin{gathered}
\dot{h} = \sin\gamma \\
\dot{v} = \frac{T\cos a - D}{m} - \frac{\mu \sin\gamma}{r^2} \\
\dot{\gamma} = \frac{T\sin a + L}{mv} + \cos\gamma\left(\frac{v}{r} - \frac{\mu}{vr^2}\right) \\
\dot{m} = \frac{-T}{g_0 I_{sp}}
\end{gathered}$$

and the boundary conditions

$$\begin{array}{cc}
h(0) = 0 & h(t_f) = 19994.88\,m \\
v(0) = 129.314 & v(t_f) = 295.092\,m\,s^{-1} \\
\gamma(0) = 0\,deg & \gamma(t_f) = 0\,deg \\
m(0) = 19050.864\,kg & m(t_f) = Free
\end{array}$$

and the control inequality constraint

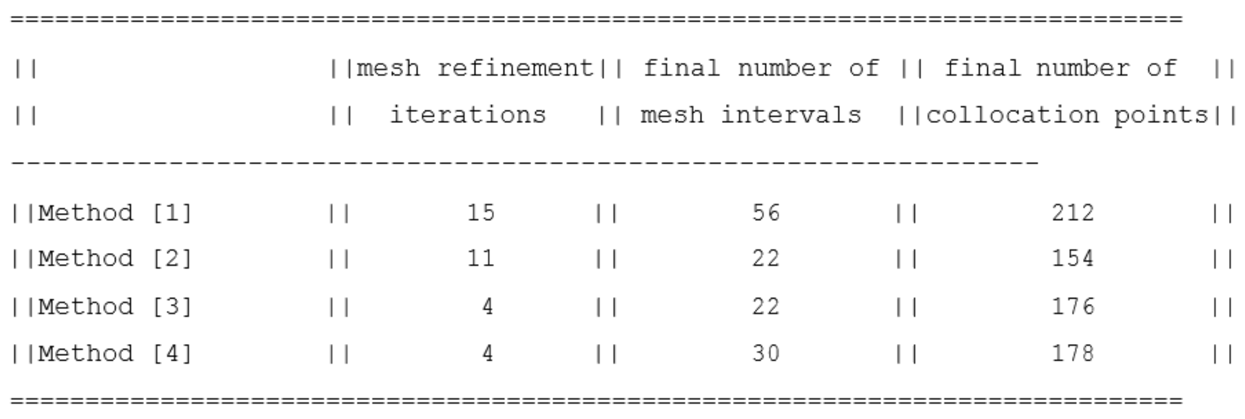
$$\frac{-\pi}{4} \le \alpha \le \frac{\pi}{4}$$

4.3.5.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier   = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies      = ’sparseNaN’
scales
       +- method = ’automatic-bounds’
method = ’RPM-Differentiation’
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 25
       +- colpointsmin      =      3
       +- colpointsmax      = 20
initial mesh characteristics
+- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = ma57
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

4.3.5.2 Results

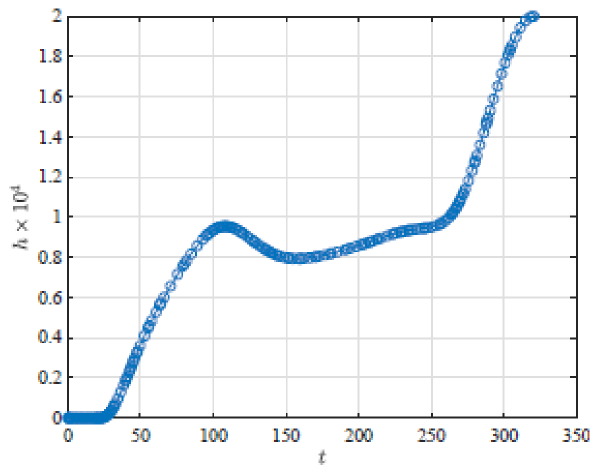
Table 5. Key performance measures for the Bryson minimum time to climb problem.



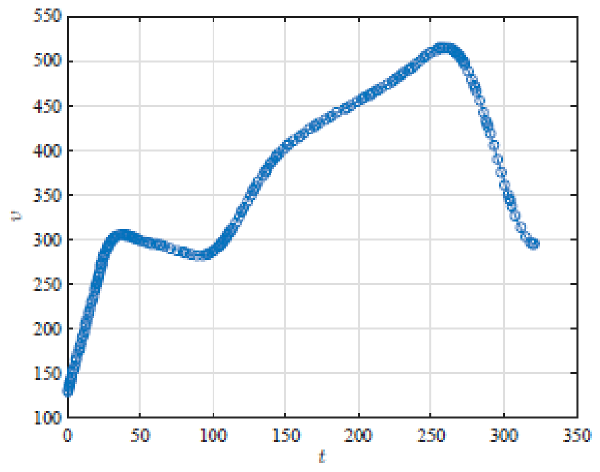
```
==========================================================================
||                 ||mesh refinement|| final number of || final number of  ||
||                 ||  iterations   || mesh intervals  ||collocation points||
-----------------------------------------------------------------
||Method [1]       ||       15      ||        56       ||        212       ||
||Method [2]       ||       11      ||        22       ||        154       ||
||Method [3]       ||        4      ||        22       ||        176       ||
||Method [4]       ||        4      ||        30       ||        178       ||
==========================================================================
```

4.3.5.3 Discussion

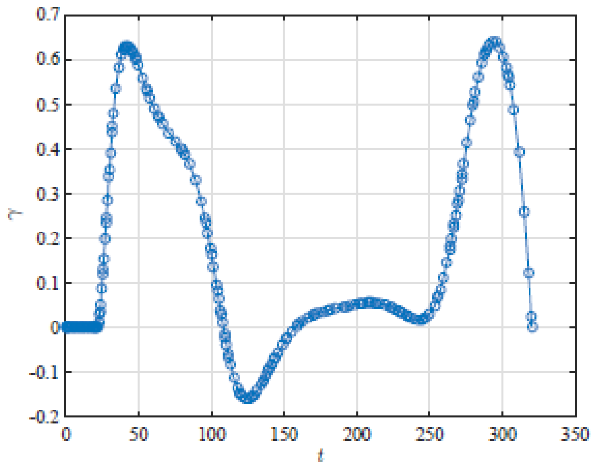
This problem has a discontinuity in the first derivative of the control, seen in Figure 20(e). Figure 21 shows how each mesh refinement method handles this type of discontinuity. The two mesh refinement methods of Ref. [44] and Ref. [45] solve the problem the fastest in four mesh iterations, as seen in Table 5. The hp-adaptive method of Ref. [44] first increases the polynomial degree before splitting the mesh intervals, which leads to a slightly faster cpu time than the hp-adaptive method of Ref. [45]. The hp-adaptive method of Ref. [41] has over thirty more mesh intervals than all the other methods because it is first an h method. Having significantly more mesh intervals than the other methods leads to the slowest cpu time. The hp-adaptive method of Ref. [43] uses the fewest amount of collocation points. However, this method is first a p method. Due to the method's algorithm, it adds collocation points before dividing mesh intervals. This addition of collocation points before dividing the mesh intervals at the location of the discontinuity leads to a longer computational time.



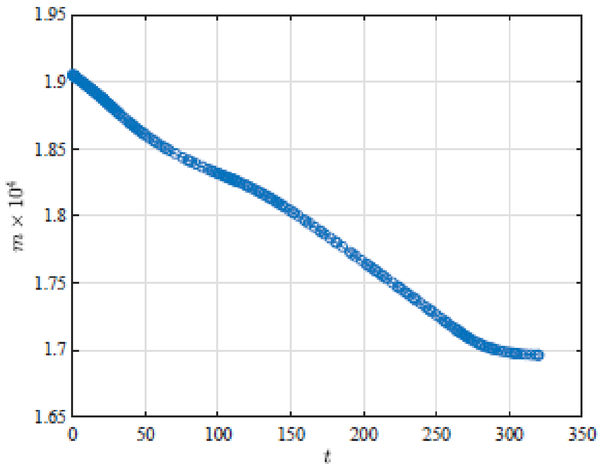
(a) State Component 1



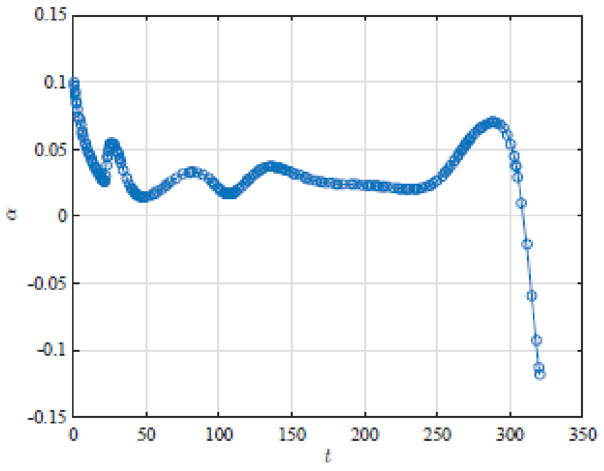
(b) State Component 2



(c) State Component 3



(d) State Component 4



(e) Control

Figure 20. Problem state and control solution.

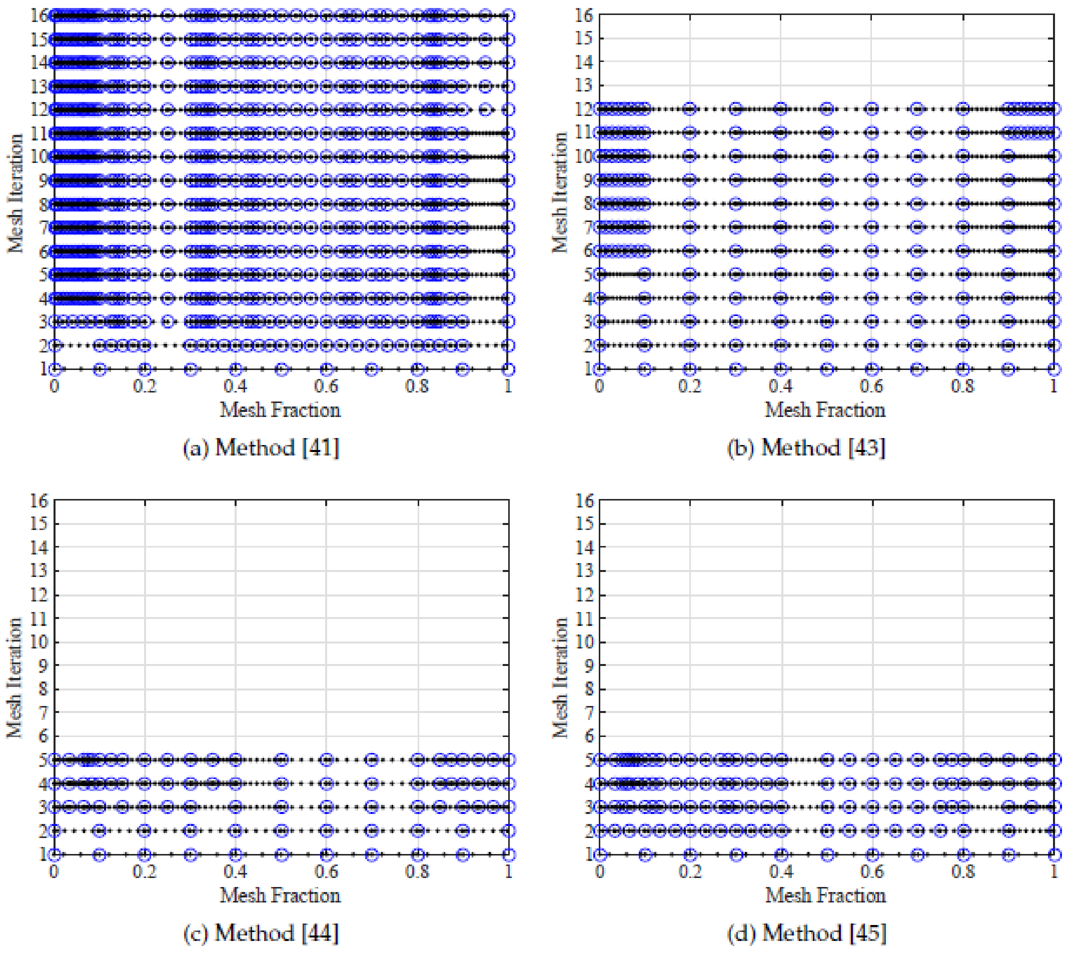


Figure 21. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

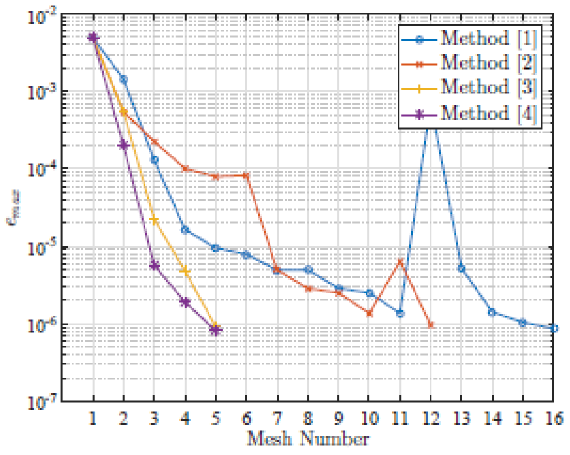


Figure 22. Mesh maximum relative error for each mesh.

4.3.6 Dynamic Soaring Problem

Minimize the cost functional:

$$J = 7\beta$$

subject to the dynamic constraints

$$\begin{gathered}
\dot{x} = v\cos\gamma\sin\psi + W_x \\
\dot{y} = v\cos\gamma\cos\psi \\
\dot{z} = v\sin\gamma \\
m\dot{v} = -D - mg\sin\gamma - m\dot{W}_x\cos\gamma\sin\psi \\
mv\dot{y} = L\cos\sigma - mg\cos\gamma + m\dot{W}_x\sin\gamma\sin\psi \\
mv\cos\gamma\dot{\psi} = m\dot{W}_x\cos\psi
\end{gathered}$$

and the boundary conditions

$$\begin{gathered}
x(0) = x(t_f) = 0 \\
y(0) = y(t_f) = 0 \\
z(0) = z(t_f) = 0 \\
v(0) = v(t_f) = 100 \\
\gamma(0) = \gamma(t_f) = 0 \\
\psi(t_f) = \psi(0) - 2\pi
\end{gathered}$$

where

$$\dot{W}_x = \beta v\sin\Upsilon$$

4.3.6.1 GPOPS-II Setup

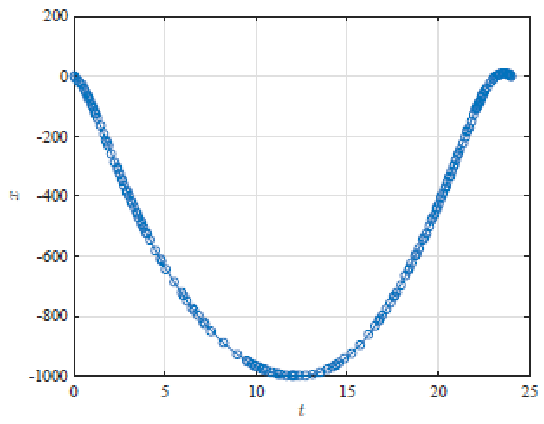
```
setup details:
derivatives
       +- supplier   = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies      = ’sparseNaN’
scales
       +- method = ’automatic-bounds’
method = ’RPM-Differentiation’
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 20
       +- colpointsmin      =      3
       +- colpointsmax      = 10
initial mesh characteristics
+- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = ma57
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

4.3.6.2 Results Discussion

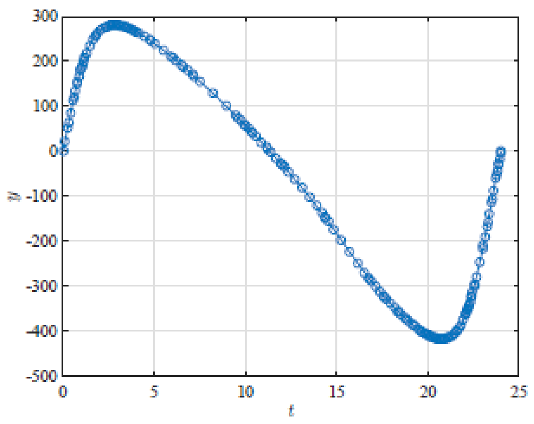
Figure 24(a) shows two discontinuities in the first derivative of the control. Figure 25 shows the mesh refinement history for each method. The hp-adaptive methods of Ref. [44] and Ref. [45] solve in three mesh refinement iterations by adding mesh points around the discontinuities. They have similar mesh sizes and solve in similar computational times. The hp-adaptive method of Ref. [41] solves in four mesh refinements but with a larger mesh size: thirty more mesh intervals and fifty more collocation points. The hp-adaptive method of Ref. [43] has one fewer mesh intervals than the hp-adaptive methods of Ref. [44] and Ref. [45] and over ten fewer collocation points. However, the hp-adaptive method of Ref. [43] is slower than the hp-adaptive methods of Ref. [44] and Ref. [45]. While the hp-adaptive method of Ref. [43] adds mesh intervals near the discontinuities, it does not identify the discontinuity. The hp-adaptive method of Ref. [44] is able to bracket and add mesh intervals around the discontinuities in its final mesh iteration, which allows it to solve the problem in one less iteration than the hp-adaptive method of Ref. [43].

Table 6. Key performance measures for the dynamic soaring problem.

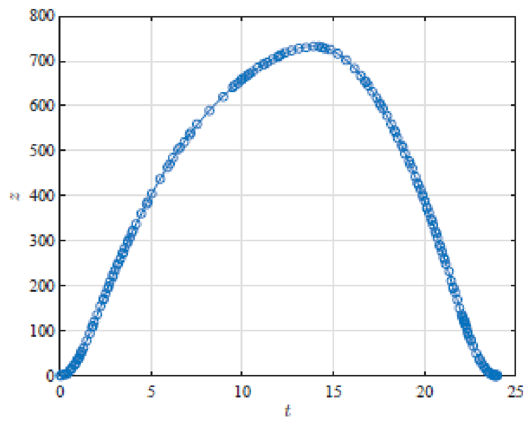
```
========================================================================
||                 ||mesh refinement|| final number of || final number of  ||
||                 ||  iterations   || mesh intervals  ||collocation points||
---------------------------------------------------------------
||Method [1]       ||        4      ||        47       ||        173       ||
||Method [2]       ||        4      ||        16       ||        110       ||
||Method [3]       ||        3      ||        17       ||        122       ||
||Method [4]       ||        3      ||        17       ||        125       ||
========================================================================
```



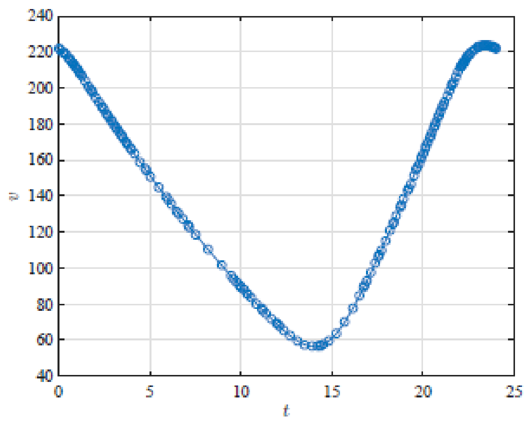
(a) State Component 1



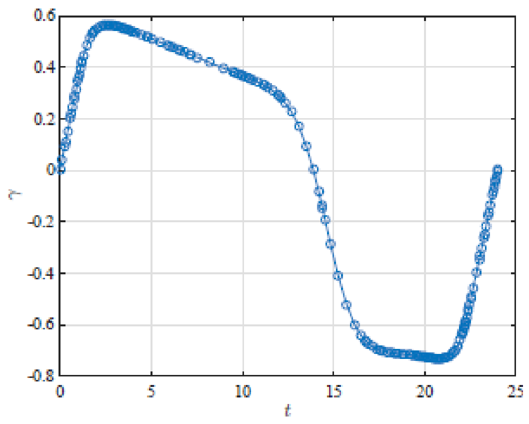
(b) State Component 2



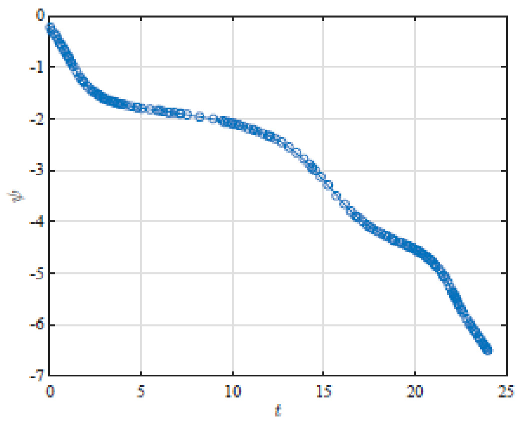
(c) State Component 3



(d) State Component 4



(e) State Component 5



(f) State Component 6

Figure 23. Problem state solution.

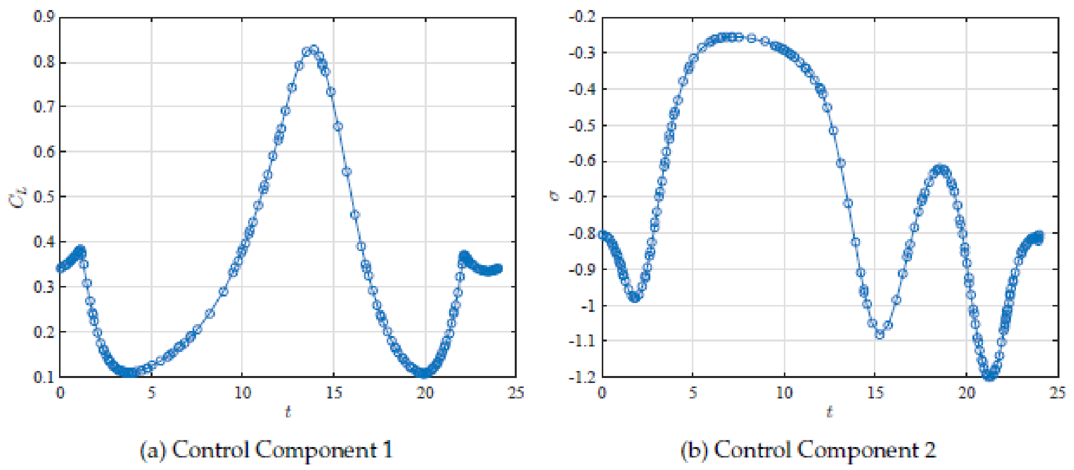


Figure 24. Problem control solution.

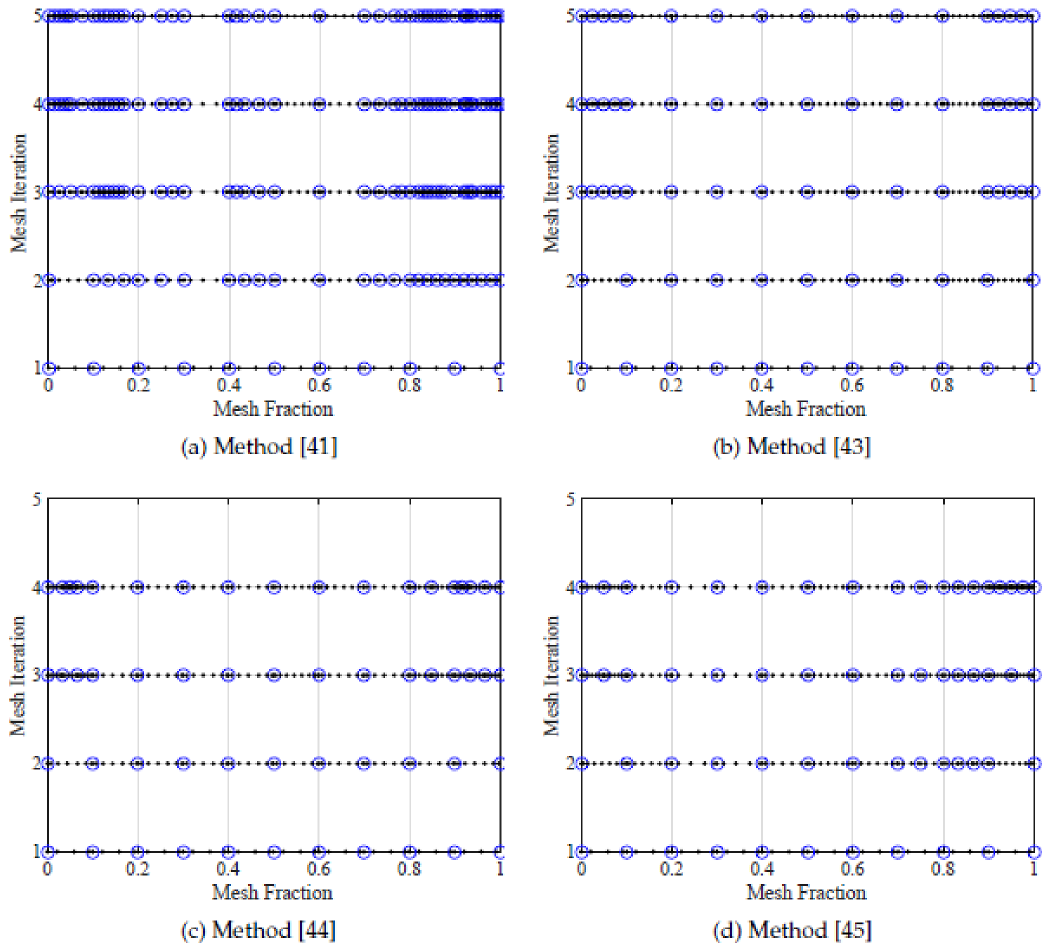


Figure 25. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

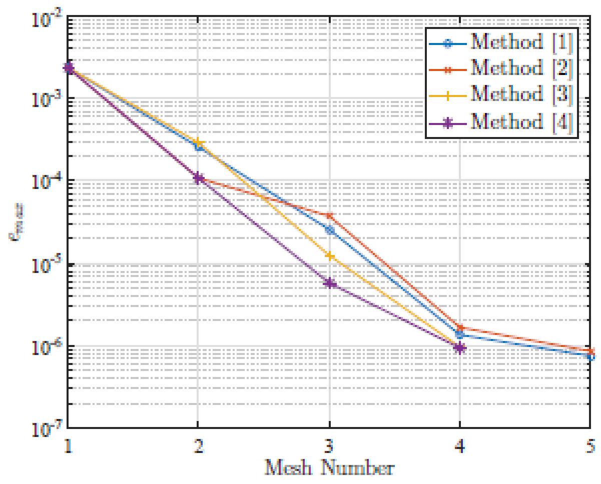


Figure 26. Mesh maximum relative error for each mesh.

4.3.7 Hypersensitive Problem

Minimize the cost functional:

$$J = \frac{1}{2}\int_0^{t_f} x^2 + u^2 \, dt$$

subject to the dynamic constraints

$$\dot{x} = -x^3 + u$$

and the boundary conditions

$$x(0) = 1.5$$
$$x\left(t_f\right) = 1$$
$$t_f = 10000$$

4.3.7.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier   = 'sparseCD'
       +- derivative level = 'second'
       +- dependencies      = 'sparseNaN'
scales
       +- method = 'none'
method = 'RPM-Differentiation'
mesh
       +- tolerance  = 1.000000e-07
       +- max iterations = 50
       +- colpointsmin      =      3
       +- colpointsmax      = 20
initial mesh characteristics
       +- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = ma57
       +- tolerance  = 1.000000e-10
       +- max iterations = 2000
display level =      0
```

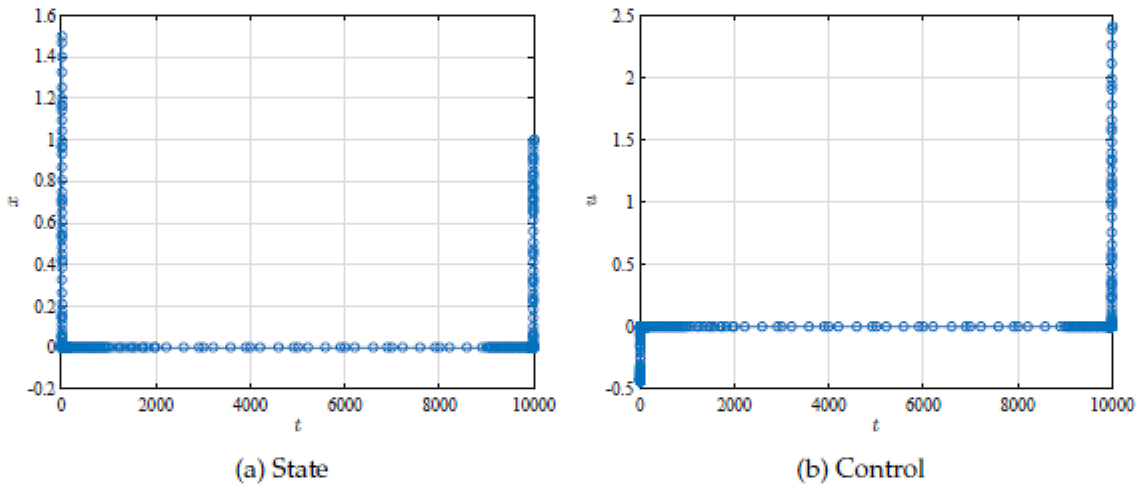


Figure 27. Problem state and control solution.

4.3.7.2 Results Discussion

The non-zero portions of the state and control solution to the hypersensitive problem lie on the intervals [0, 10] and [9990, 10000] respectively, seen in Figure 28. When compared to the entire problem domain on [0, 10000], the two non-zero state/control intervals each represent only 1 of the entire domain. Considering that the initial mesh has ten mesh intervals which each span $\frac{1}{2}$ of the entire problem domain, the hypersensitive problem is an excellent problem to test each mesh refinement method's capability to quickly and efficiently allocate collocation points in regions of interest (where the state and control are nonzero and changing with time). The top performers for this problem are the hp-adaptive methods of Ref. [44] and Ref. [45] due to their more efficient allocation of grid points near the initial and final time, as well as their ability to minimize the number of unnecessary grid points in the middle portion of the mesh (where both the state and control remain zero). The hp-adaptive method of Ref. [45] clocks in faster than the hp-adaptive method of Ref. [44] despite taking one more mesh refinement iteration. The faster computation time of the hp-adaptive method of Ref. [45] appears to be correlated with its smaller final mesh size (fewer mesh intervals and collocation points), which indicates that the corresponding NLP's being solved on each new mesh were generally smaller in size and more computationally tractable. The hp-adaptive methods of Ref. [41] and Ref. [43] had the largest final mesh sizes due to their inability to merge mesh intervals or decrease the polynomial degree on mesh intervals. The hp-adaptive method of Ref. [43] had a smaller final mesh size compared to the hp-adaptive method of Ref. [41] but took three more mesh refinement iterations to converge to the solution, resulting in the hp-adaptive method of Ref. [43] having the slowest CPU time out of the four methods. Therefore, h refinement is the preferred choice over p refinement in a problem like hypersensitive where extremely small domain mesh intervals are needed in precise locations to accurately capture the solution.

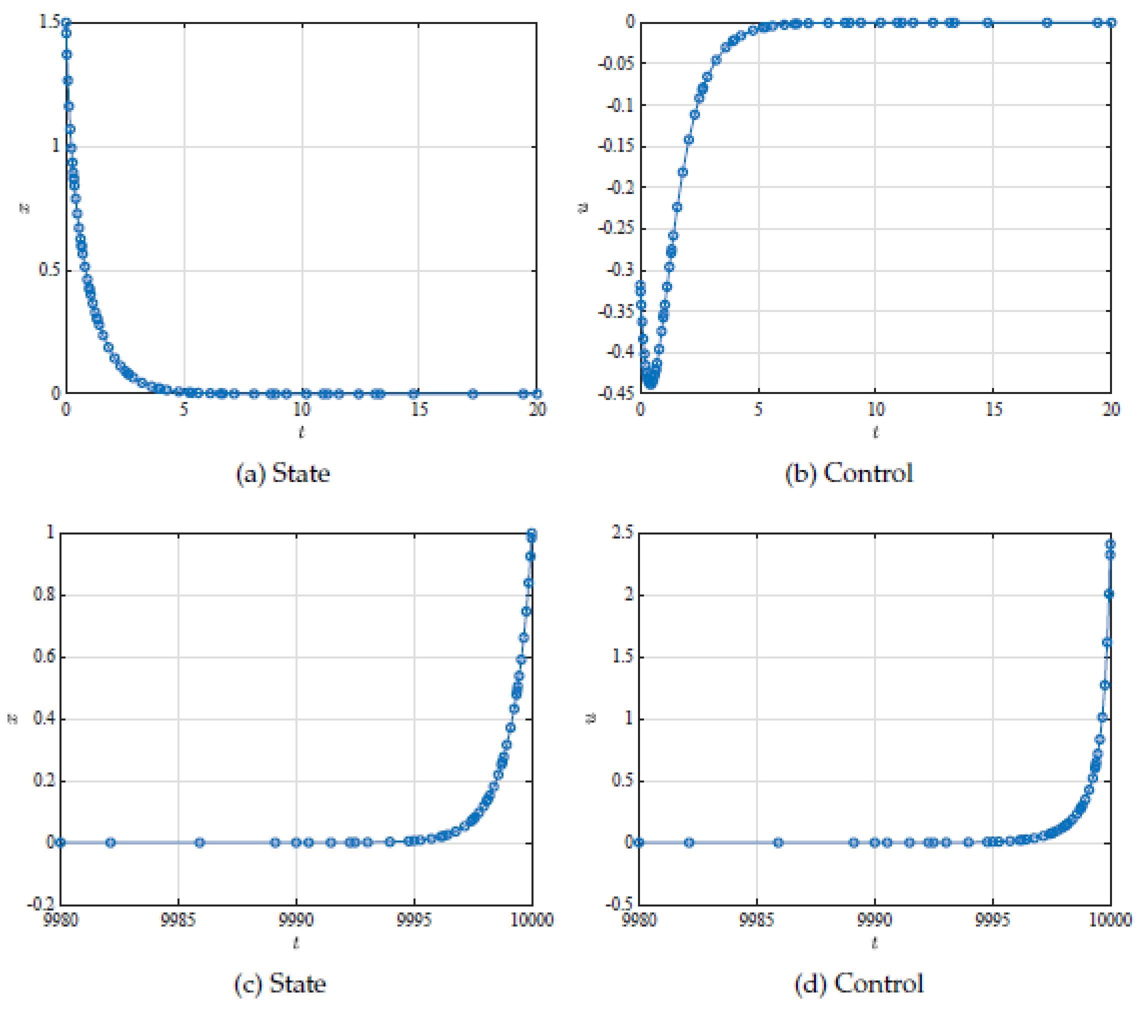
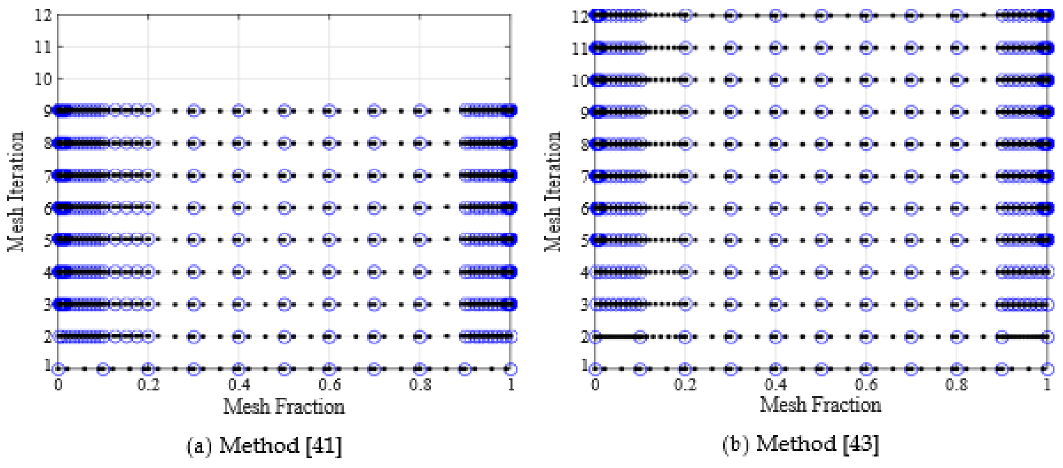


Figure 28. Close-up of state and control solution.

Table 7. Key performance measures for the hypersensitive problem.

	mesh refinement iterations	final number of mesh intervals	final number of collocation points
Method [1]	8	80	295
Method [2]	11	54	242
Method [3]	6	30	197
Method [4]	7	26	126



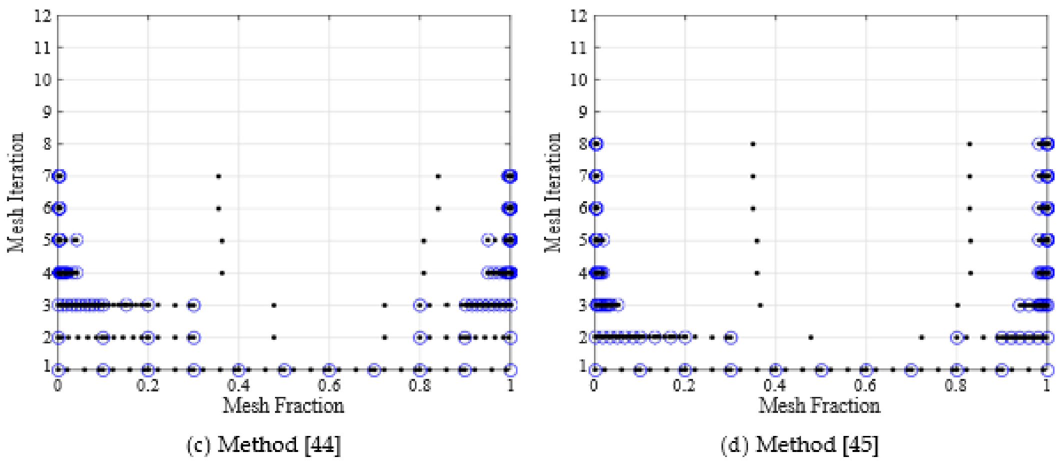


Figure 29. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

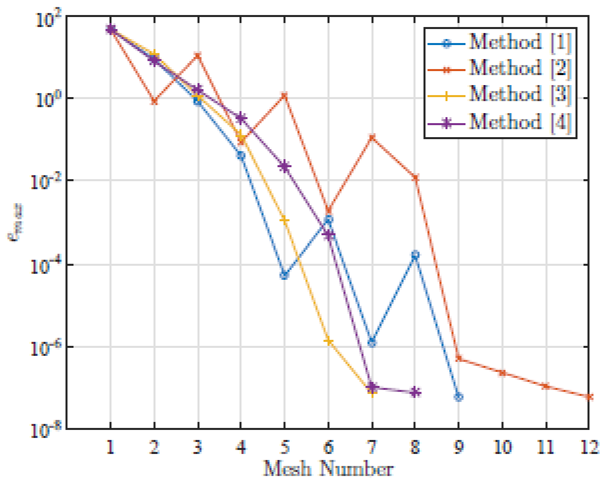


Figure 30. Mesh maximum relative error for each mesh.

4.3.8 Moon Lander Problem

Minimize the cost functional:

$$J = \int_0^{t_f} u \, dt$$

subject to the dynamic constraints

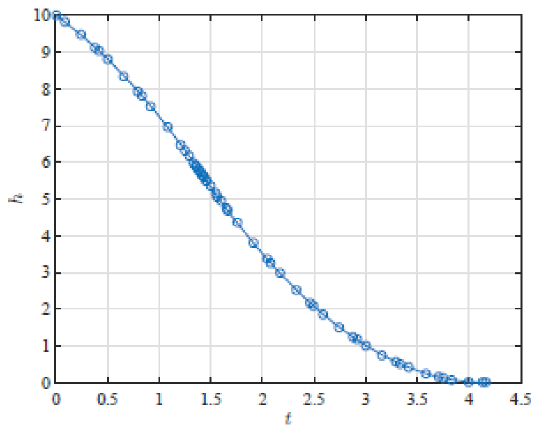
$$\dot{h} = v$$
$$\dot{v} = -g + u$$

and the boundary conditions

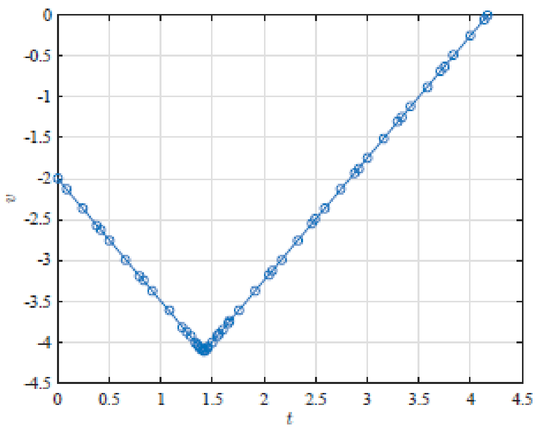
$$h(0) = 10, \qquad h(t_f) = 0$$
$$v(0) = -2, \qquad v(t_f) = 0$$

4.3.8.1 GPOPS-II Setup

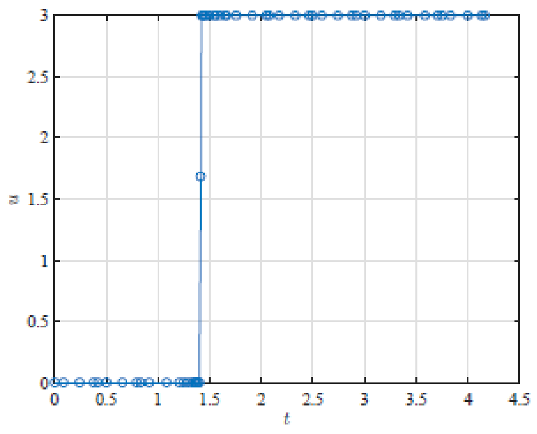
```
 setup details:
 derivatives
    +- supplier         = ’sparseCD’
    +- derivative level = ’second’
    +- dependencies     =’sparseNaN’
scales
    +- method = ’none’
 method = ’RPM-Differentiation’
 mesh
    +- tolerance      = 1.000000e-06
    +- max iterations = 20
    +- colpointsmin   = 3
    +- colpointsmax   = 20
initial mesh characteristics
    +- Phase  1
    * fraction= [1 1 1 1 1 1 1 1 1 1]/10
    * colpoints = [4 4 4 4 4 4 4 4 4 4]
 nlp
    +- solver         = ipopt
    +- linear solver  = mumps
    +- tolerance      = 1.000000e-07
    +- max iterations = 2000
display level =   0
```



(a) State Component 1



(b) State Component 2



(c) Control

Figure 31. Problem state and control solution.

Table 8. Key performance measures for the moon lander problem.

	mesh refinement iterations	final number of mesh intervals	final number of collocation points
Method [1]	2	15	54
Method [2]	3	10	46
Method [3]	3	5	27
Method [4]	3	7	23

4.3.8.2 Results Discussion

Although simple in its problem statement, the Moon Lander problem is deceivingly difficult for all four of the mesh refinement methods to solve efficiently. A single discontinuity in the solution for the control causes a discontinuity in the second state component derivative. Such non-smooth behavior on a particular mesh interval is difficult to approximate with a smooth polynomial approximation for the state. Therefore, the difficulty in solving this problem efficiently lies in splitting the mesh exactly at the discontinuity point so that the piecewise polynomial nature of the solution can be approximated using piecewise polynomials with the correct switch point.

The hp-adaptive method of Ref. [41] was most efficient in terms of the number of mesh refinement iterations needed to converge to the solution. However, the hp-adaptive method of Ref. [41] was least efficient in its allocation of collocation points to achieve solution accuracy, resulting in the largest final mesh size and one of the slowest computation times. The hp-adaptive methods of Ref. [44] and Ref. [45] had the most efficient use of collocation points in their final mesh, because of their ability to merge mesh intervals. Oddly enough, the hp-adaptive method of Ref. [43] was able to meet the mesh maximum relative error tolerance without ever splitting the mesh, but it took three iterations of mesh refinement to do so. Overall, the computation times of all four methods are comparable with one another, and the hp-adaptive method of Ref. [44] came out slightly faster than the rest.

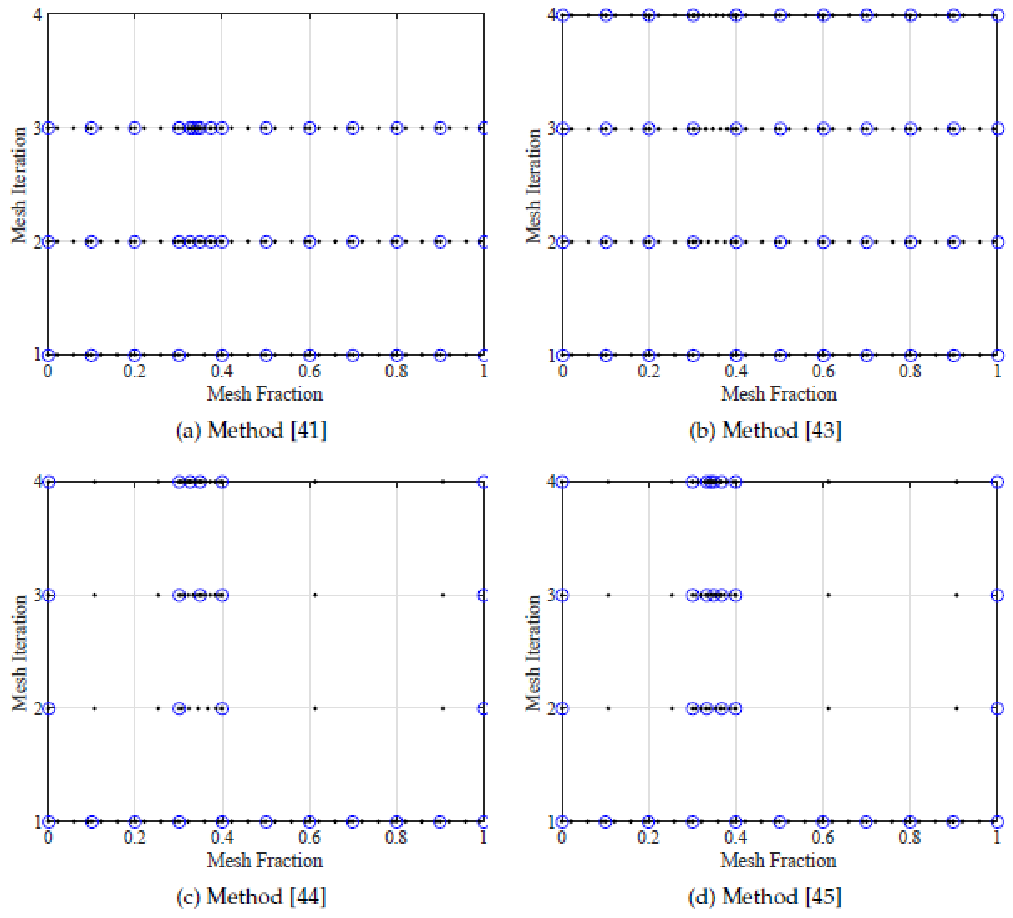


Figure 32. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

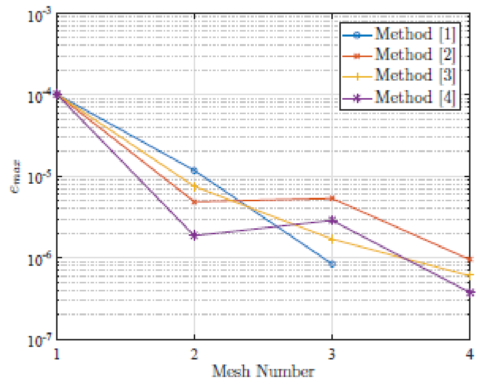


Figure 33. Mesh maximum relative error for each mesh.

4.3.9 Orbit-Raising Problem

Minimize the cost functional:

$$J = -r(t_f)$$

subject to the dynamic constraints

$$\begin{aligned}\dot{r} &= v_r \\ \dot{\theta} &= \frac{v_\theta}{r} \\ \dot{v}_r &= \frac{{v_\theta}^2}{r} - \frac{\mu}{r^2} + a u_1 \\ \dot{v_\theta} &= \frac{v_r v_\theta}{r} + a u_2\end{aligned}$$

the boundary conditions

$$\begin{array}{rr} r(0) = 1 & r(t_f) = Free \\ \theta(0) = 0 & \theta(t_f) = Free \\ & v_r(t_f) = 0 \\ v_r(0) = 0 & \\ v_\theta(0) = 1 & v_\theta(t_f) = \sqrt{\dfrac{\mu}{r(t_f)}} \end{array}$$

and the control equality constraint

$${u_1}^2 + {u_2}^2 = 1$$

where

$$\begin{aligned} a &= \frac{T}{m_0 + \dot{m}t} \\ T &= 0.1405 \\ m_0 &= 1 \\ \dot{m} &= 0.0749 \\ \mu &= 1 \end{aligned}$$

4.3.9.1 GPOPS-II Setup

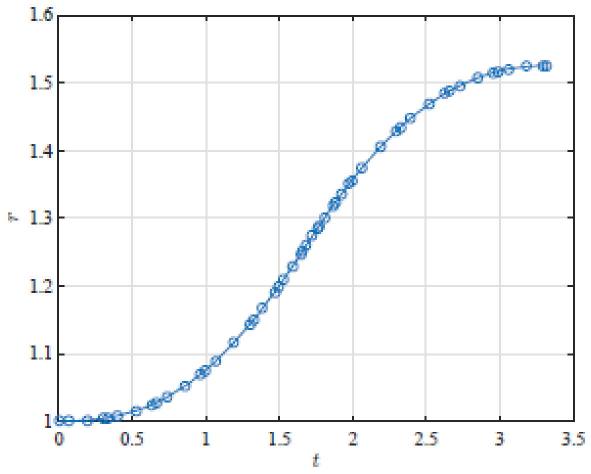
```
setup details:
derivatives
       +- supplier   = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies      = ’sparseNaN’
scales
       +- method = ’automatic-hybrid’
method = ’RPM-Differentiation’
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 25
       +- colpointsmin      =      3
       +- colpointsmax      = 10
initial mesh characteristics
       +- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = mumps
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

4.3.9.2 Results Discussion

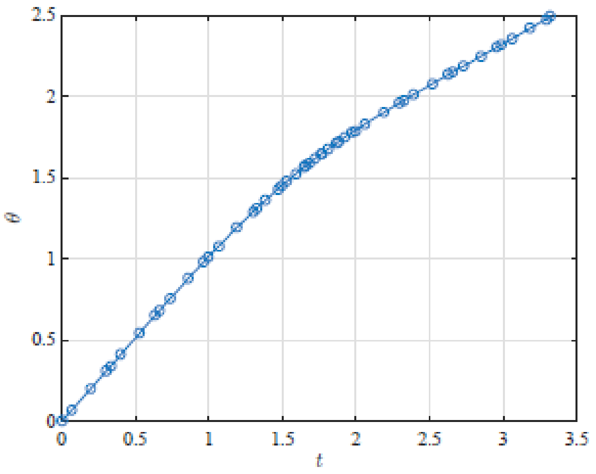
The solution to the orbit-raising problem is smooth and well-behaved for the state. However, the control experiences rapid changes in its first component as well as the first derivative of its second component near the halfway point of the maneuver, as seen in Figure 34. The simplicity of the solution is reflected in the performances of the mesh refinement methods. All four methods (excluding the hp-adaptive method of Ref. [41]) converged to the solution by simply increasing the polynomial approximation degree in the middle mesh intervals where the state and control change most rapidly. In the case of the hp-adaptive methods of Ref. [43] and Ref. [44], only one mesh refinement iteration was needed to achieve the desired solution accuracy. All four methods have comparable CPU times, but the hp-adaptive methods of Ref. [43] and Ref. [44] were slightly faster than the rest, because they used one fewer mesh refinement iteration, as seen in Table 9.

Table 9. Key performance measures for the orbit raising problem.

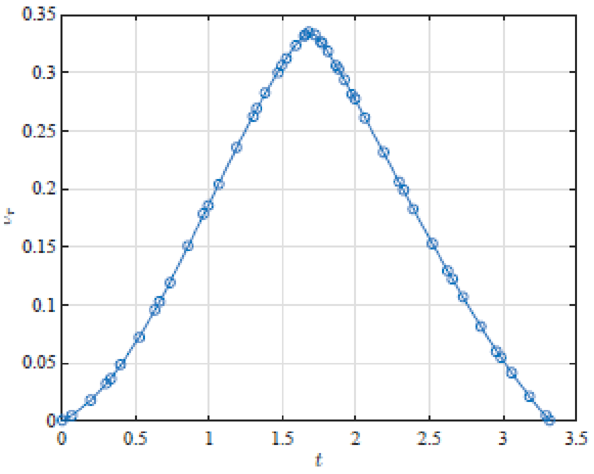
```
==============================================================================
||                  ||mesh refinement|| final number of || final number of  ||
||                  ||  iterations   || mesh intervals  ||collocation points||
-------------------------------------------------------------------
||Method [1]        ||       2       ||       13        ||        49        ||
||Method [2]        ||       1       ||       10        ||        45        ||
||Method [3]        ||       1       ||       10        ||        46        ||
||Method [4]        ||       2       ||       10        ||        45        ||
==============================================================================
```



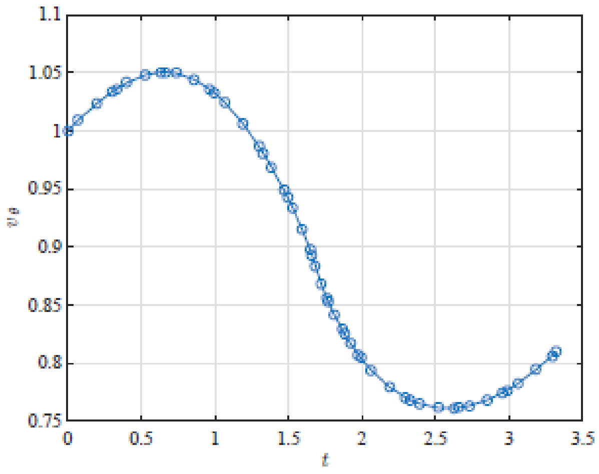
(a) State Component 1



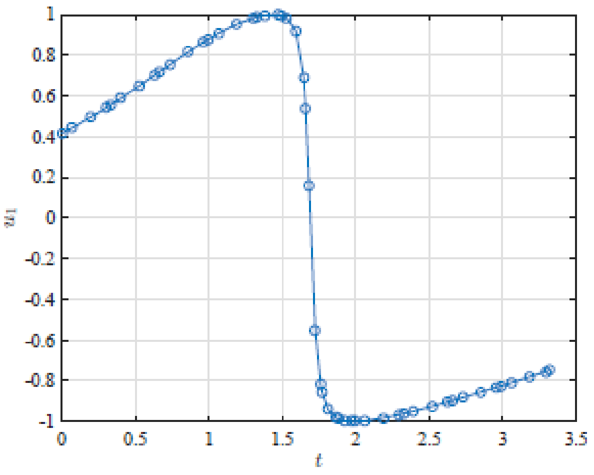
(b) State Component 2



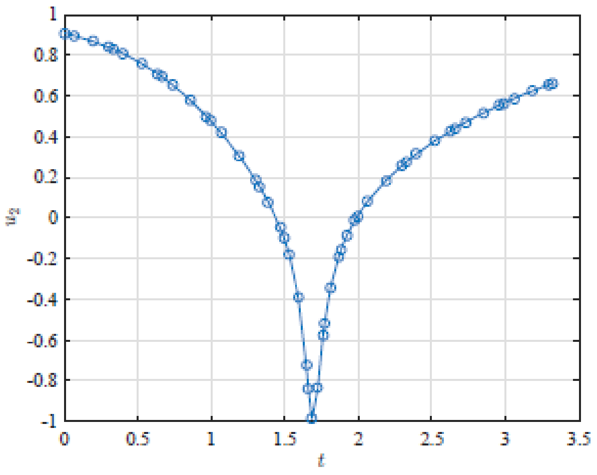
(c) State Component 3



(d) State Component 4

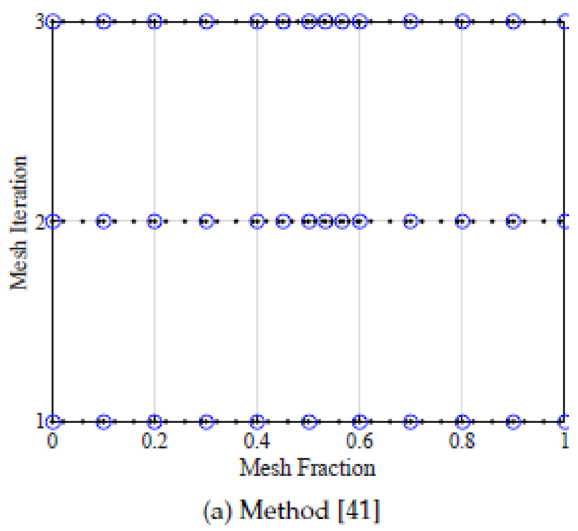


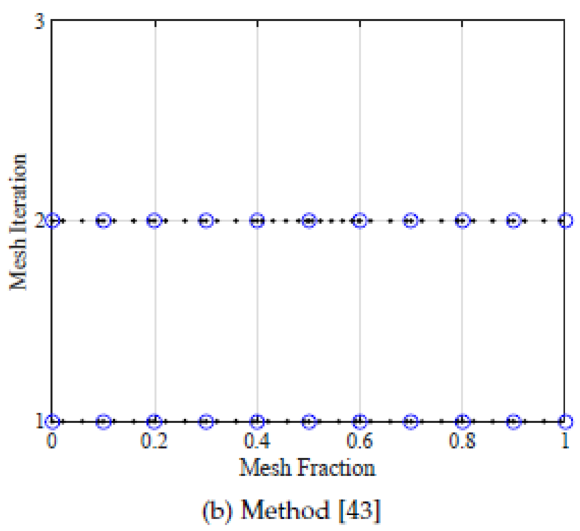
(e) Control Component 1

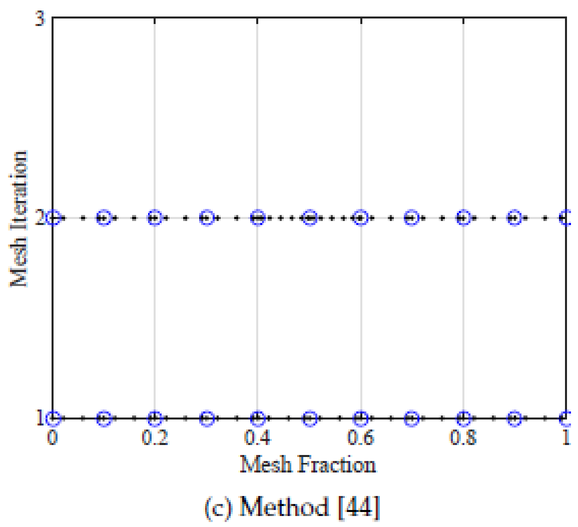


(f) Control Component 2

Figure 34. Problem state and control solution.


(a) Method [41]


(b) Method [43]


(c) Method [44]

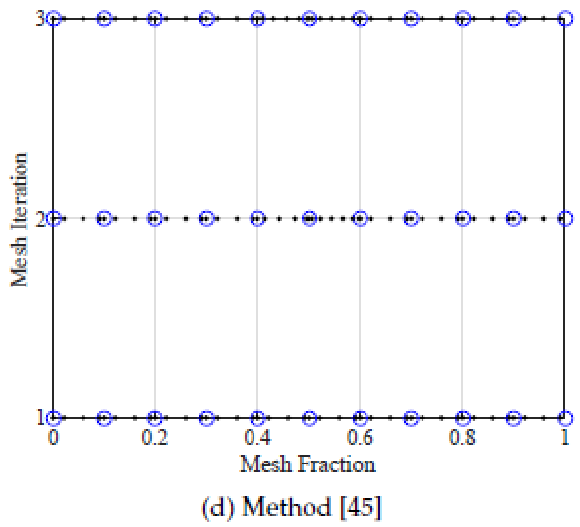

(d) Method [45]

Figure 35. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

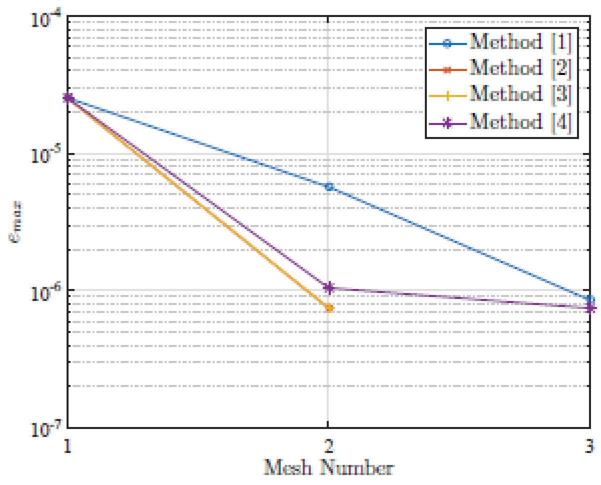


Figure 36. Mesh maximum relative error for each mesh.

4.3.10 Minimum-Time Reorientation of an Asymmetric Rigid Body

Minimize the cost functional:

$$J = t_f$$

subject to the dynamic constraints

$$\begin{aligned}
\dot{q}_1 &= \frac{1}{2}(\omega_1 q_4 - \omega_2 q_3 + \omega_3 q_2) \\
\dot{q}_2 &= \frac{1}{2}(\omega_1 q_3 + \omega_2 q_4 - \omega_3 q_1) \\
\dot{q}_3 &= \frac{1}{2}(-\omega_1 q_2 + \omega_2 q_1 + \omega_3 q_4) \\
\dot{\omega}_1 &= \frac{u_1}{I_x} - \left(\frac{I_z - I_y}{I_x}\right)\omega_2\omega_3 \\
\dot{\omega}_2 &= \frac{u_2}{I_y} - \left(\frac{I_x - I_z}{I_y}\right)\omega_1\omega_3 \\
\dot{\omega}_3 &= \frac{u_3}{I_z} - \left(\frac{I_y - I_x}{I_z}\right)\omega_1\omega_2
\end{aligned}$$

and the boundary conditions

$$\begin{gathered}
q_1(0) = 0, q_1(t_f) = \sin\left(0.5\frac{150\pi}{180}\right) \\
q_2(0) = 0, q_2(t_f) = 0 \\
q_3(0) = 0, q_3(t_f) = 0 \\
\omega_1(0) = 0, \omega_1(t_f) = 0 \\
\omega_2(0) = 0, \omega_2(t_f) = 0 \\
\omega_3(0) = 0, \omega_3(t_f) = 0
\end{gathered}$$

and the path constraint

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

where

$$\begin{gathered}
I_x = 5621 \\
I_y = 4547 \\
I_z = 2364
\end{gathered}$$

Note that the control consists of $\{q_4, u_1, u_2, u_3\}$, where q_4 corresponds to the scalar part of the quaternion $q = \{q_1, q_2, q_3, q_4\}$.

4.3.10.1 GPOPS-II Setup

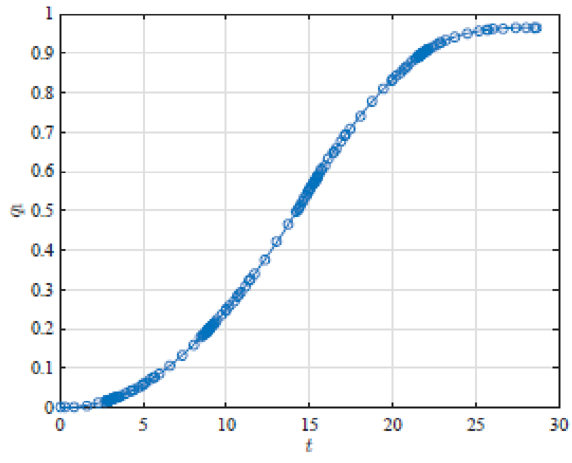
```
setup details:
derivatives
       +- supplier   = 'sparseCD'
       +- derivative level = 'second'
       +- dependencies      = 'sparseNaN'
scales
       +- method = 'none'
method = 'RPM-Integration'
mesh
       +- tolerance  = 1.000000e-08
       +- max iterations = 25
       +- colpointsmin      =      3
       +- colpointsmax      = 10
initial mesh characteristics
       +- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = ma57
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

4.3.10.2 Results Discussion

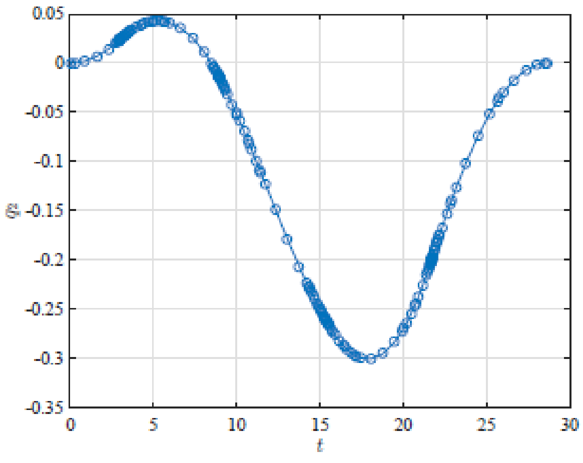
Five discontinuities in the components of the control cause five discontinuities in the first derivatives of the state components, as seen in Figures 37 and 38. All four mesh refinement methods spent the majority of their time refining the mesh intervals that contained the nonsmooth solution behavior. No matter which method was used, the mesh histories of each method, shown in Figure 39, show that mesh points tended to concentrate near the discontinuity locations as the intervals containing the discontinuities were split into smaller and smaller mesh intervals. Surprisingly, on the first two mesh refinement iterations, the hp-adaptive method of Ref. [43] behaved more like an h method than the hp-adaptive method of Ref. [41]. However, the unusual behavior for the hp-adaptive method of Ref. [43] resulted in it achieving the fastest computation time out of all four methods. The hp-adaptive method of Ref. [44] used the fewest number of refinement iterations to converge to the solution but was slower than the hp-adaptive methods of Ref. [43] and Ref. [45] due to its large mesh size.

Table 10. Key performance measures for the reorientation problem.

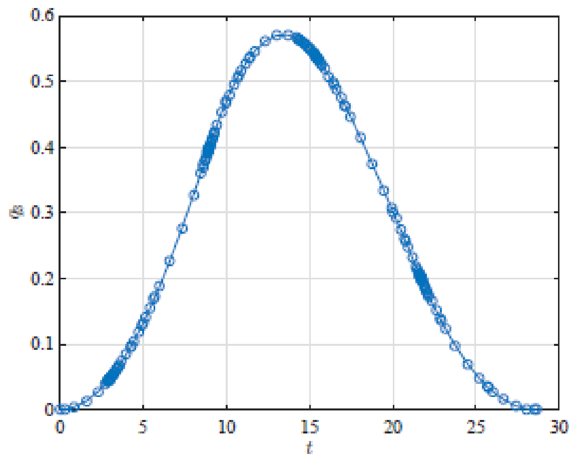
```
==========================================================================
||                  ||mesh refinement|| final number of || final number of  ||
||                  ||  iterations   || mesh intervals  ||collocation points||
-----------------------------------------------------------------
||Method [1]        ||       11      ||        49       ||        190       ||
||Method [2]        ||        7      ||        37       ||        183       ||
||Method [3]        ||        6      ||        37       ||        262       ||
||Method [4]        ||        7      ||        41       ||        181       ||
==========================================================================
```



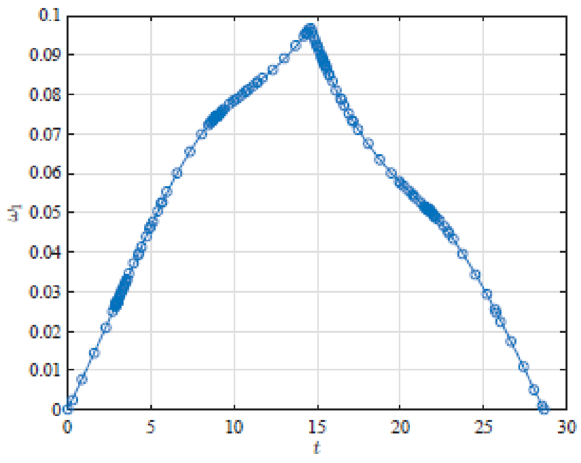
(a) State Component 1



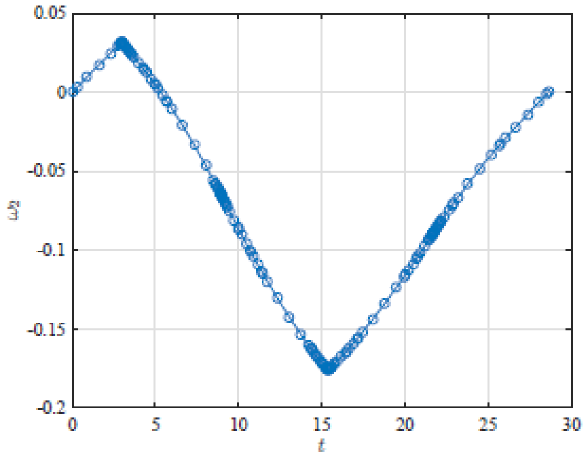
(b) State Component 2



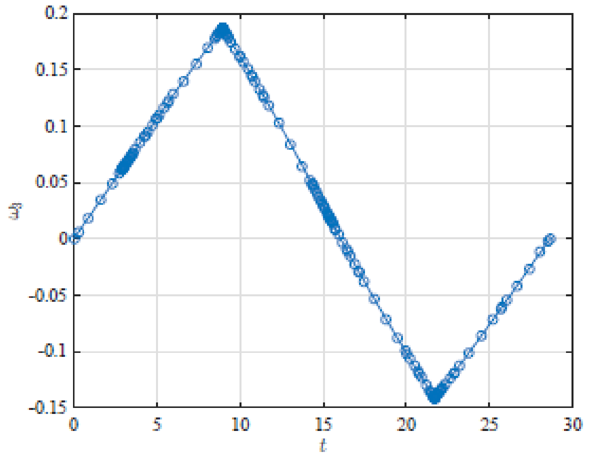
(c) State Component 3



(d) State Component 4

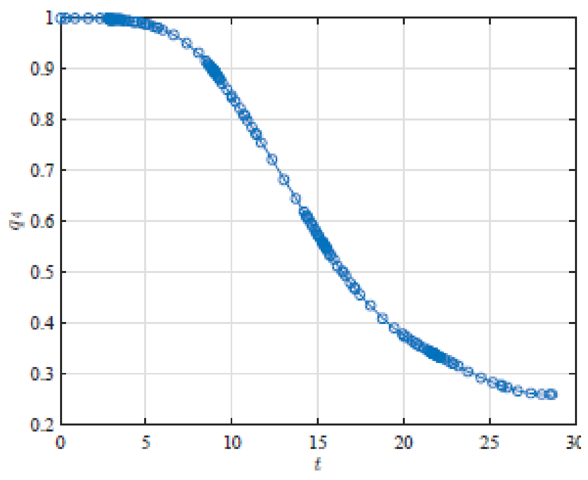


(e) State Component 5

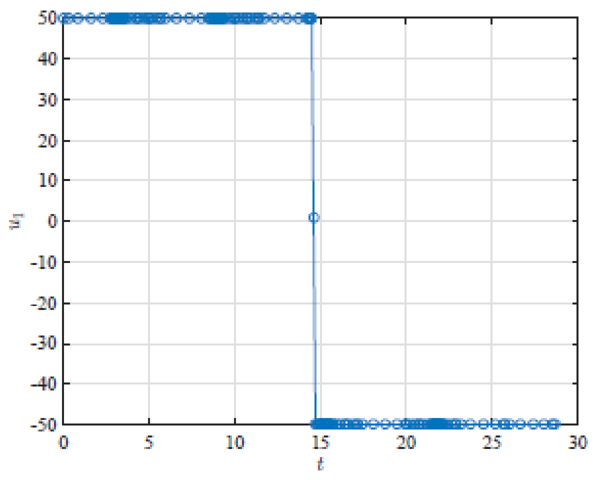


(f) State Component 6

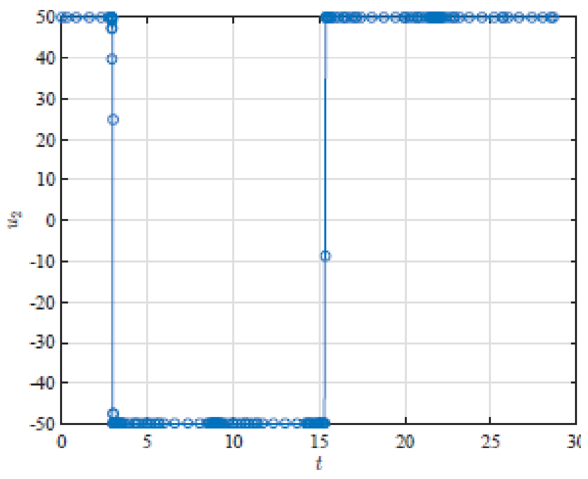
Figure 37. Problem state solution.



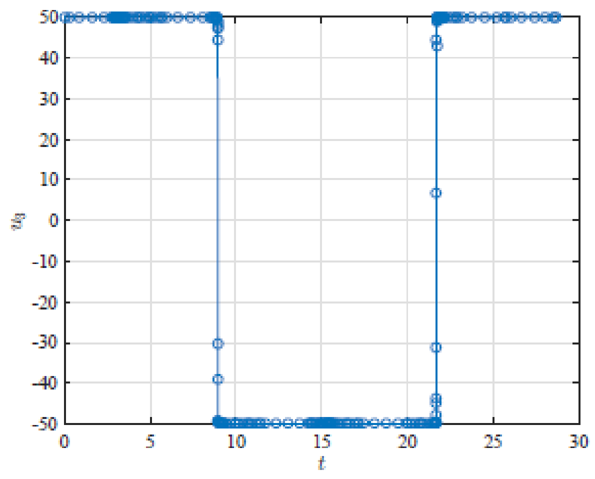
(a) Control Component 1



(b) Control Component 2



(c) Control Component 3



(d) Control Component 4

Figure 38. Problem control solution.

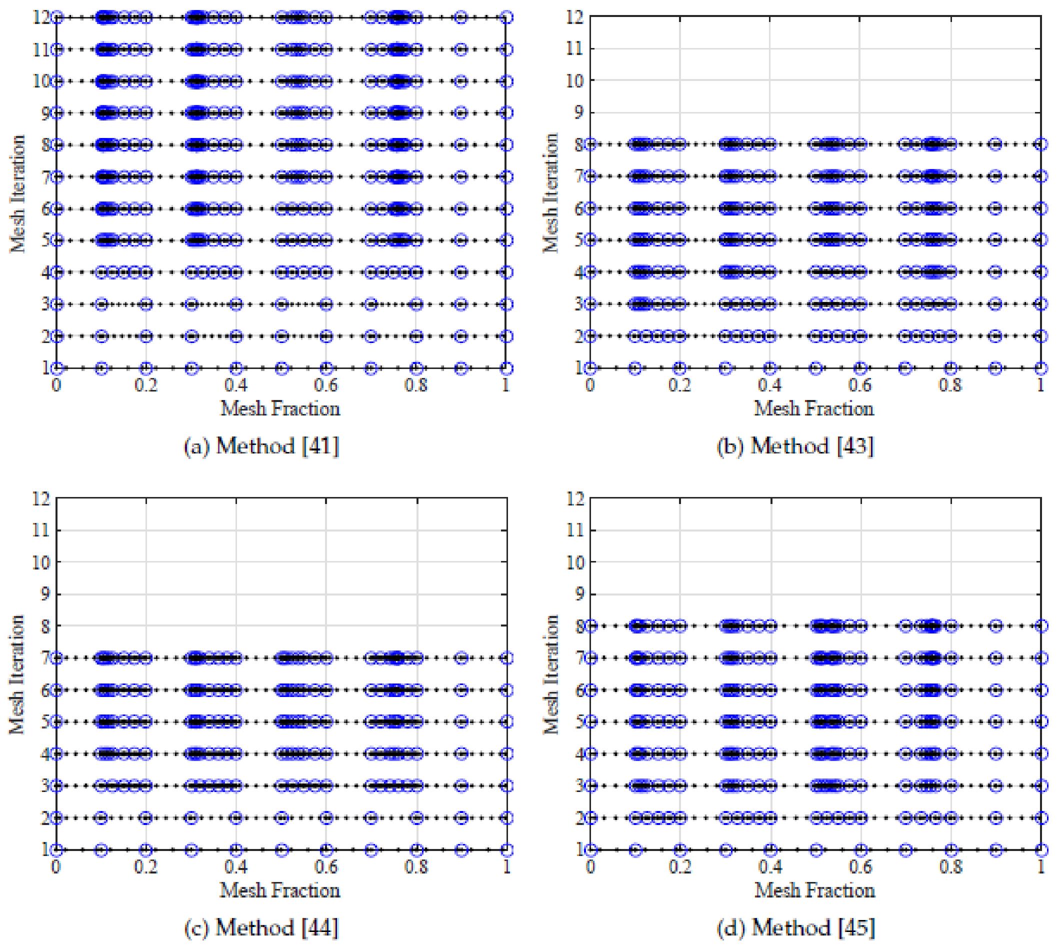


Figure 39. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

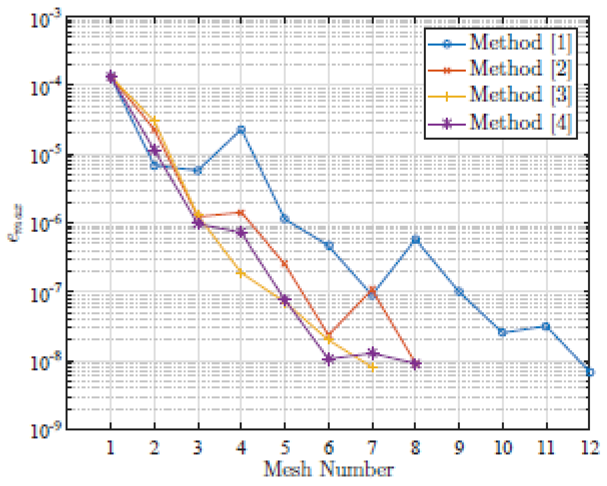


Figure 40. Mesh maximum relative error for each mesh.

4.3.11 Maximum Cross Range Reusable Launch Vehicle Entry

Minimize the cost functional:

$$J = -\phi(t_f)$$

subject to the dynamic constraints

$$\begin{aligned}
\dot{r} &= v \sin\gamma \\
\dot{\theta} &= \frac{v\cos\gamma\sin\psi}{r\cos\phi} \\
\dot{\phi} &= \frac{v\cos\gamma\cos\psi}{r} \\
\dot{v} &= -\frac{D}{m} g\sin\gamma \\
\dot{\gamma} &= \frac{L\cos\sigma}{mv} - \left(\frac{g}{v} - \frac{v}{r}\right)\cos\gamma \\
\dot{\psi} &= \frac{L\sin\sigma}{mv\cos\gamma} + \frac{v\cos\gamma\sin\psi\tan\phi}{r}
\end{aligned}$$

and the boundary conditions

$$\begin{gathered}
r(0) = 79248 + R_e, r(t_f) = 24384 + R_e m \\
\theta(0) = 0\ deg, \theta(t_f) = Free \\
\phi(0) = 0\ deg, \phi(t_f) = Free \\
v(0) = 7802.88, v(t_f) = 762\ m\ s^{-1} \\
\gamma(0) = -1\ deg, \gamma(t_f) = -5\ deg \\
\psi(0) = 90\ deg, \psi(t_f) = Free
\end{gathered}$$

where

$$g = \frac{\mu}{r^2}$$
$$D = 0.5pv^2SC_D$$
$$L = 0.5pv^2SC_L$$
$$\rho = \rho_0 exp\left(-\frac{r - R_e}{H}\right)$$
$$C_D = c_d(1) + c_d(2)a + c_d(3)a^2$$
$$C_L = c_{l(1)+}c_{l(2)a}$$

Important auxiliary data for this problem is contained in Table 11.

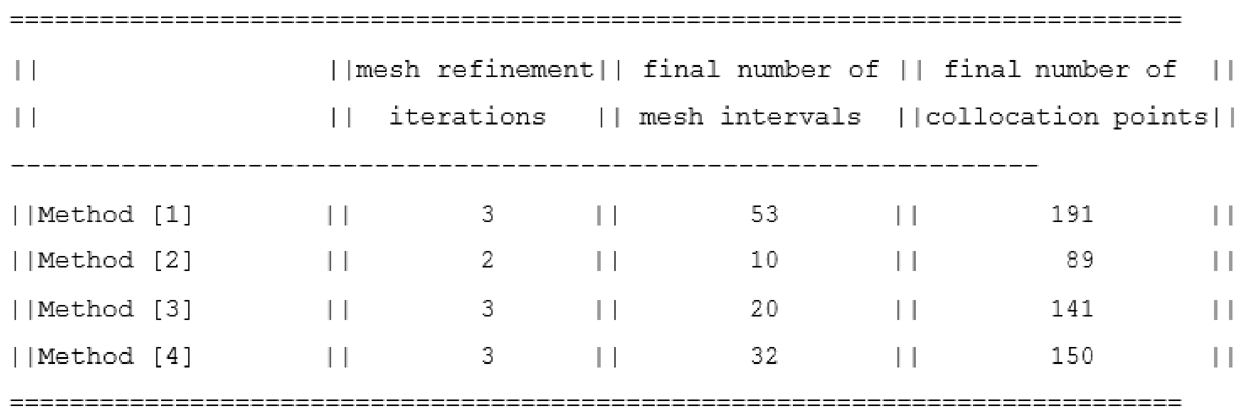
Table 11. Auxiliary data for the rlvEntry problem.

Symbol	Value
R_e	6371204 m
S	249.91 m^2
c_d	$[0.0785, -0.3529, 2.0400]^T$
c_l	$[-0.2070, 1.6756]^T$
H	7254 m
ρ_0	1.225571 $\frac{kg}{m^3}$
μ	3.986032×10^{14} $\frac{m^3}{s^2}$
m	92079 kg

4.3.11.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier   = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies      = ’sparseNaN’
scales
       +- method = ’automatic-bounds’
method = ’RPM-Integration’
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 20
       +- colpointsmin      =      3
       +- colpointsmax      = 20
initial mesh characteristics
       +- Phase      1
       * fraction    = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver = ipopt
       +- linear solver = ma57
       +- tolerance = 1.000000e-07
       +- max iterations = 2000
display level = 0
```

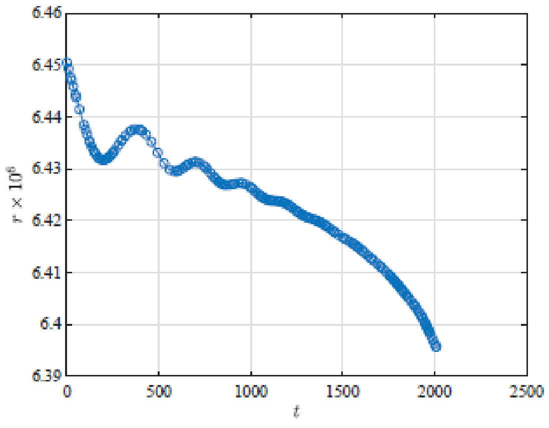
Table 12. Key performance measures for the rlv entry problem.



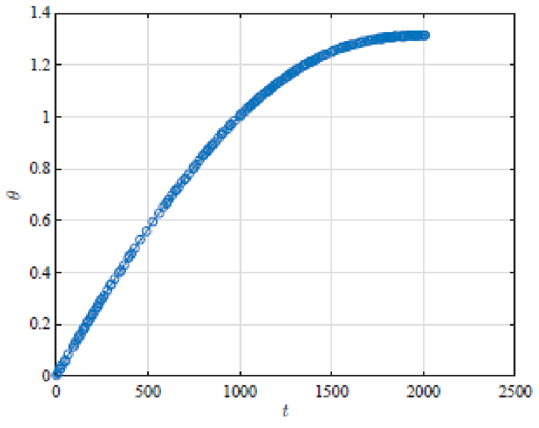
	mesh refinement iterations	final number of mesh intervals	final number of collocation points
Method [1]	3	53	191
Method [2]	2	10	89
Method [3]	3	20	141
Method [4]	3	32	150

4.3.11.2 Results Discussion

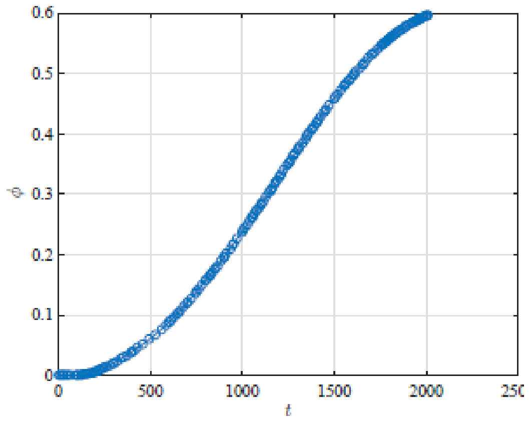
Both the state and control have smooth solutions; however, state components 1, 4, 5, and 6, as well as control component 1 all display oscillatory behavior (to varying degrees) which dampen out as the trajectory progresses. In addition, state component 5 and control component 1 dip down and back up rapidly at the end of the maneuver. Due to the smooth nature of the solution, the hp-adaptive method of Ref. [43] excelled over the other methods by using its p refinement approach. The hp-adaptive method of Ref. [43] did not split a single interval, converged in the fewest number of mesh refinement iterations, had the smallest final mesh size, and achieved the fastest cpu time as a result. The other three methods took one more refinement iteration than the hp-adaptive method of Ref. [43] and had many more mesh intervals and collocation points in their final mesh.



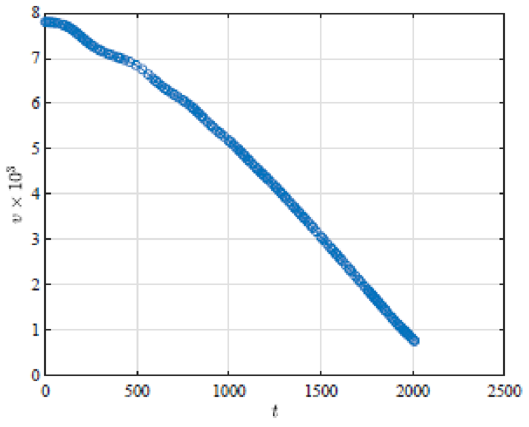
(a) State Component 1



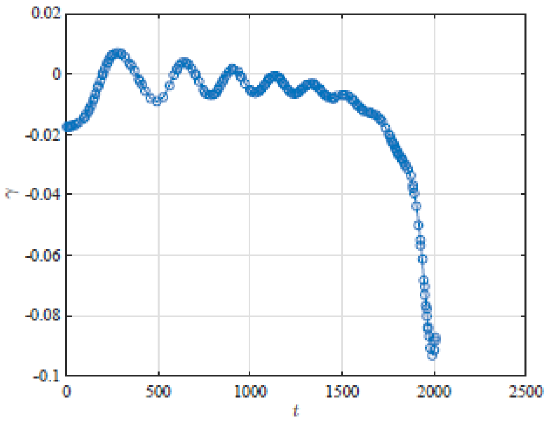
(b) State Component 2



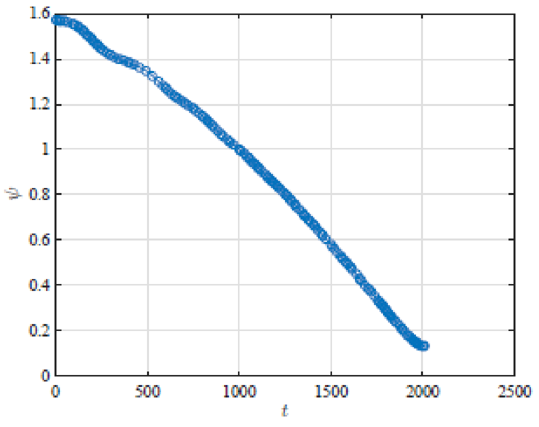
(c) State Component 3



(d) State Component 4

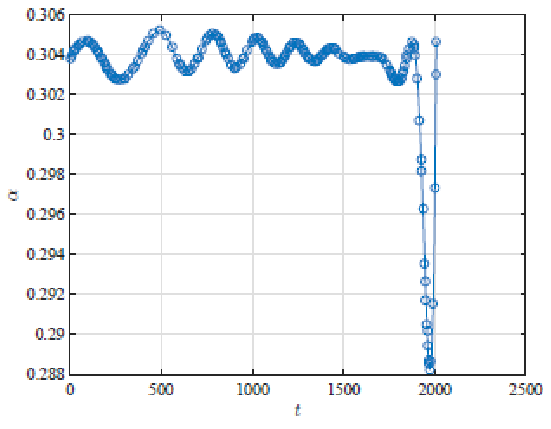


(e) State Component 5



(f) State Component 6

Figure 41. Problem state solution.


(a) Control Component 1

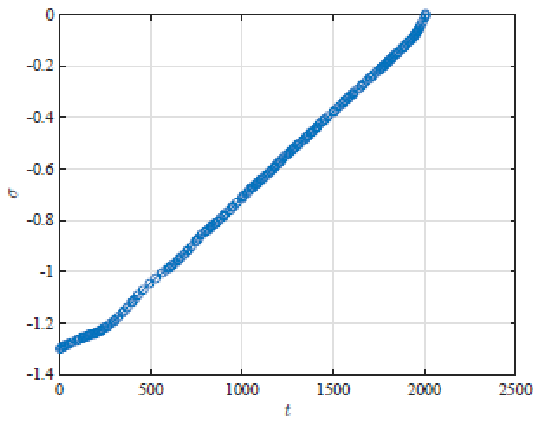
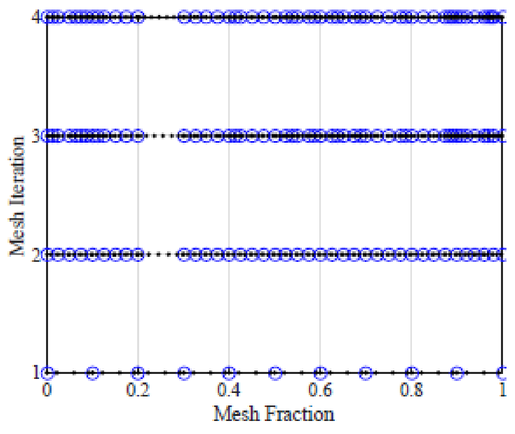
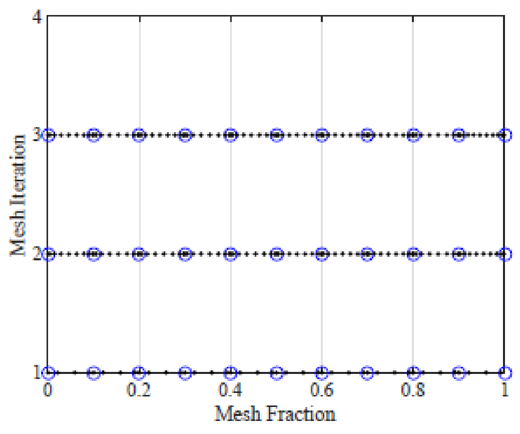
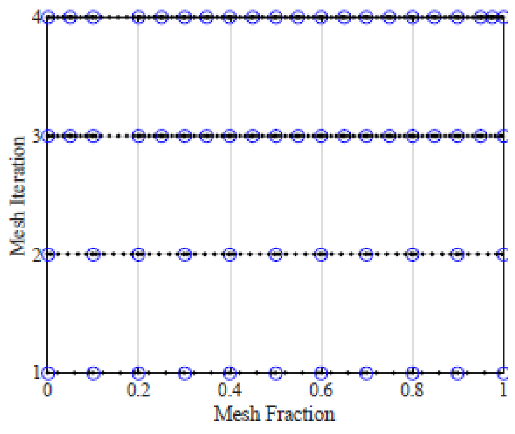

(b) Control Component 2

Figure 42. Problem control solution.


(a) Method [41]


(b) Method [43]


(c) Method [44]

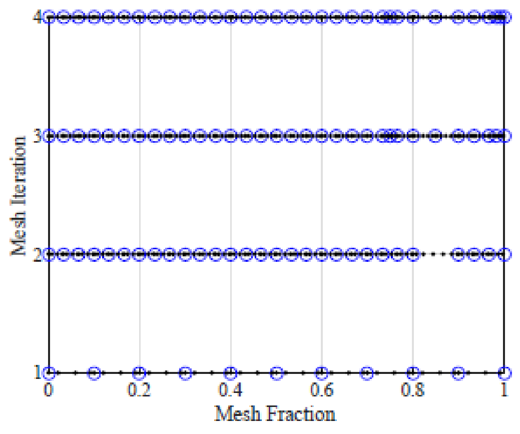

(d) Method [45]

Figure 43. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

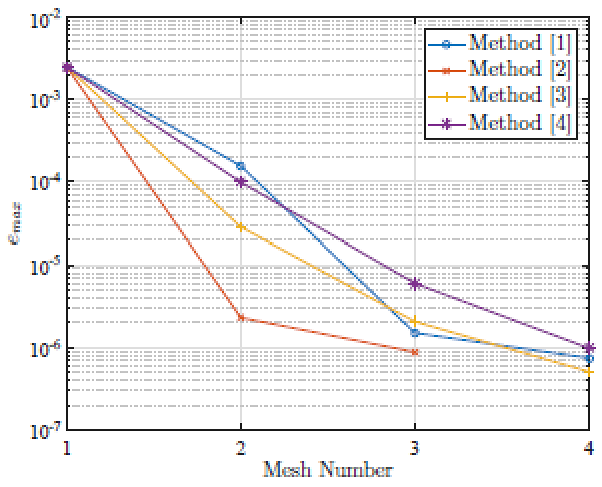


Figure 44. Mesh maximum relative error for each mesh.

4.3.12 Minimum-Time Reorientation of a Robot Arm

Minimize the cost functional:

$$J = t_f$$

subject to the dynamic constraints

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = \frac{u_1}{L}$$
$$\dot{x}_3 = x_4, \qquad \dot{x}_4 = \frac{u_2}{I_\theta}$$
$$\dot{x}_5 = x_6, \qquad \dot{x}_6 = \frac{u_3}{I_\phi}$$

and the boundary conditions

$$x_1(0) = \frac{9}{2},\ x_1(t_f) = \frac{9}{2}$$
$$x_2(0) = 0,\ x_2(t_f) = 0$$
$$x_3(0) =,\ x_3(t_f) = \frac{2\pi}{3}$$
$$x_4(0) =,\ x_4(t_f) = 0$$
$$x_5(0) =,\ x_5(t_f) = \frac{\pi}{4}$$
$$x_6(0) =,\ x_6(t_f) = 0$$

and the control inequality constraints

$$-1 \leq u_i \leq 1, (i = 1,2,3)$$

where

$$I_\phi = \frac{(L - x_1)^3 + {x_1}^3}{3}$$
$$I_\theta = I_\phi \sin^2(x_5)$$
$$L = 5$$

4.3.12.1 GPOPS-II Setup

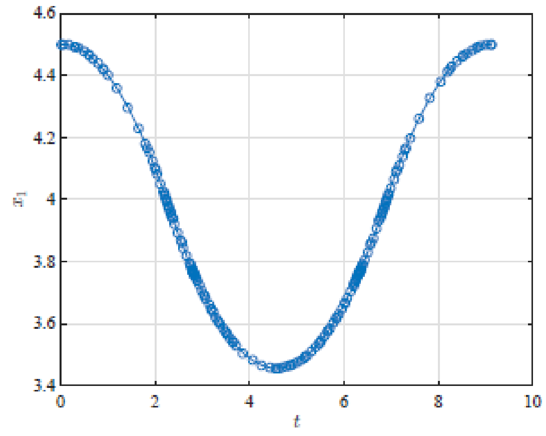
```
setup details:
derivatives
       +- supplier = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies = ’sparseNaN’
scales
       +- method = ’automatic-guess’
method = ’RPM-Differentiation’
mesh
       +- tolerance  = 1.000000e-08
       +- max iterations = 20
       +- colpointsmin = 3
       +- colpointsmax = 20
initial mesh characteristics
       +- Phase 1
       * fraction = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver = ipopt
       +- linear solver = ma57
       +- tolerance = 1.000000e-09
       +- max iterations = 2000
display level = 0
```

4.3.12.2 Results Discussion

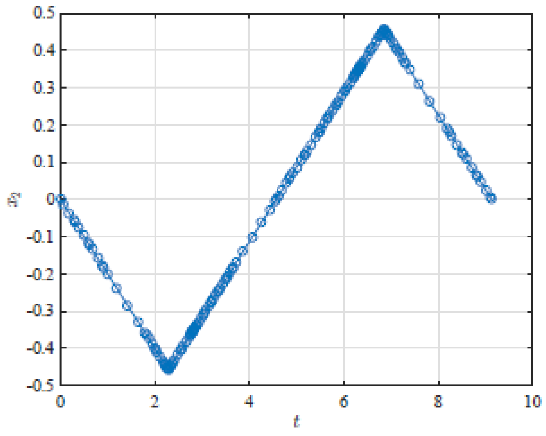
The robot arm problem has a bang-bang solution for each of the control components as seen in Figure 46. A total of five discontinuities in the control cause corresponding discontinuities in the state derivative, seen in Figure 45. In between discontinuity points, the state solution is smooth and relatively simple in its behavior. Therefore, the key to solving the robot arm problem efficiently lies in splitting the mesh at the five discontinuity locations. None of the methods is capable of estimating the discontinuity locations and taking targeted h refinement action at those locations. Instead, when using h refinement on a particular mesh interval, the original interval is evenly split into new intervals. Repeated application of evenly splitting intervals that contain a discontinuity results in mesh points concentrating around the discontinuity locations. The hp-adaptive method of Ref. [41] achieved the fastest computation time, illustrating the efficiency of h refinement for intervals which contain a discontinuity. The hp-adaptive methods of Ref. [44] and Ref. [45] were close behind the hp-adaptive method of Ref. [41] in their computation times, primarily using h-refinement as well on the nonsmooth mesh intervals. In contrast to the other methods, the hp-adaptive method of Ref. [43] utilized p refinement on most of its mesh refinement iterations. Showcasing what can go wrong in a p method approach, the hp-adaptive method of Ref. [43] took more than ten more mesh refinement iterations to achieve solution accuracy, and it was the slowest in terms of computation time as well. However, the hp-adaptive method of Ref. [43] did manage to use the fewest number of collocation points in its final mesh.

Table 13. Key performance measures for the robot arm problem.

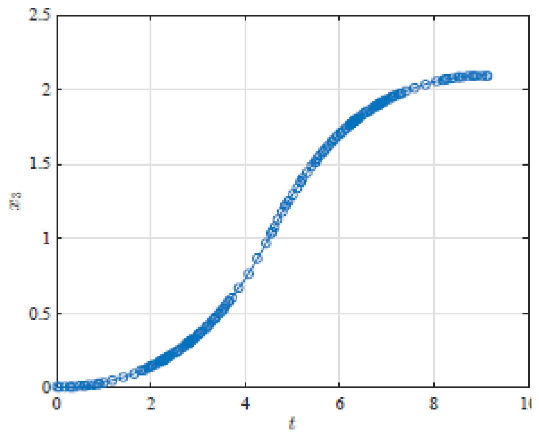
```
==========================================================================
||                 ||mesh refinement|| final number of || final number of  ||
||                 ||  iterations   || mesh intervals  ||collocation points||
-----------------------------------------------------------------
||Method [1]       ||        5      ||        51       ||        192       ||
||Method [2]       ||       17      ||        44       ||        175       ||
||Method [3]       ||        6      ||        33       ||        231       ||
||Method [4]       ||        6      ||        33       ||        182       ||
==========================================================================
```



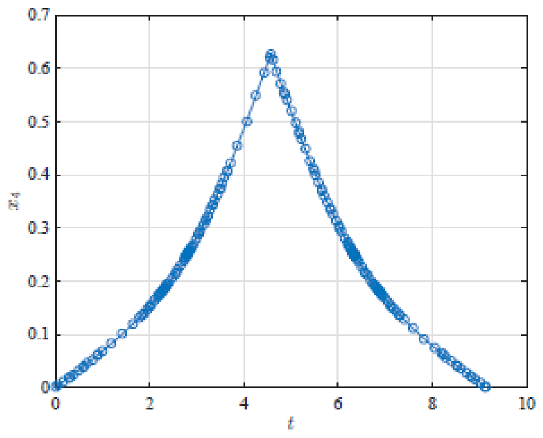
(a) State Component 1



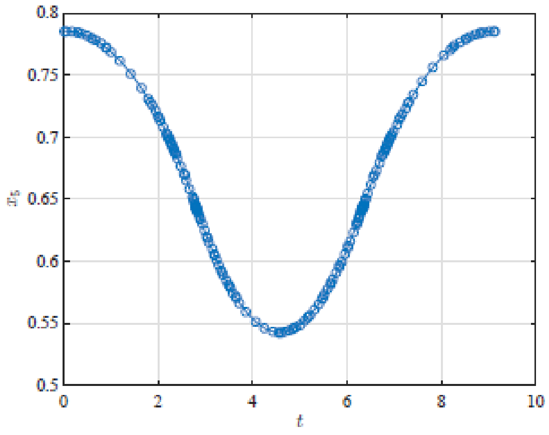
(b) State Component 2



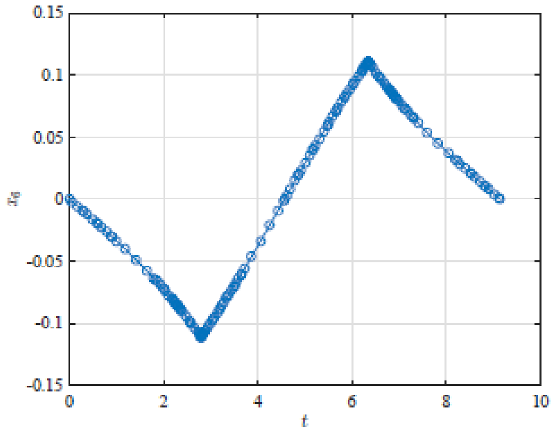
(c) State Component 3



(d) State Component 4

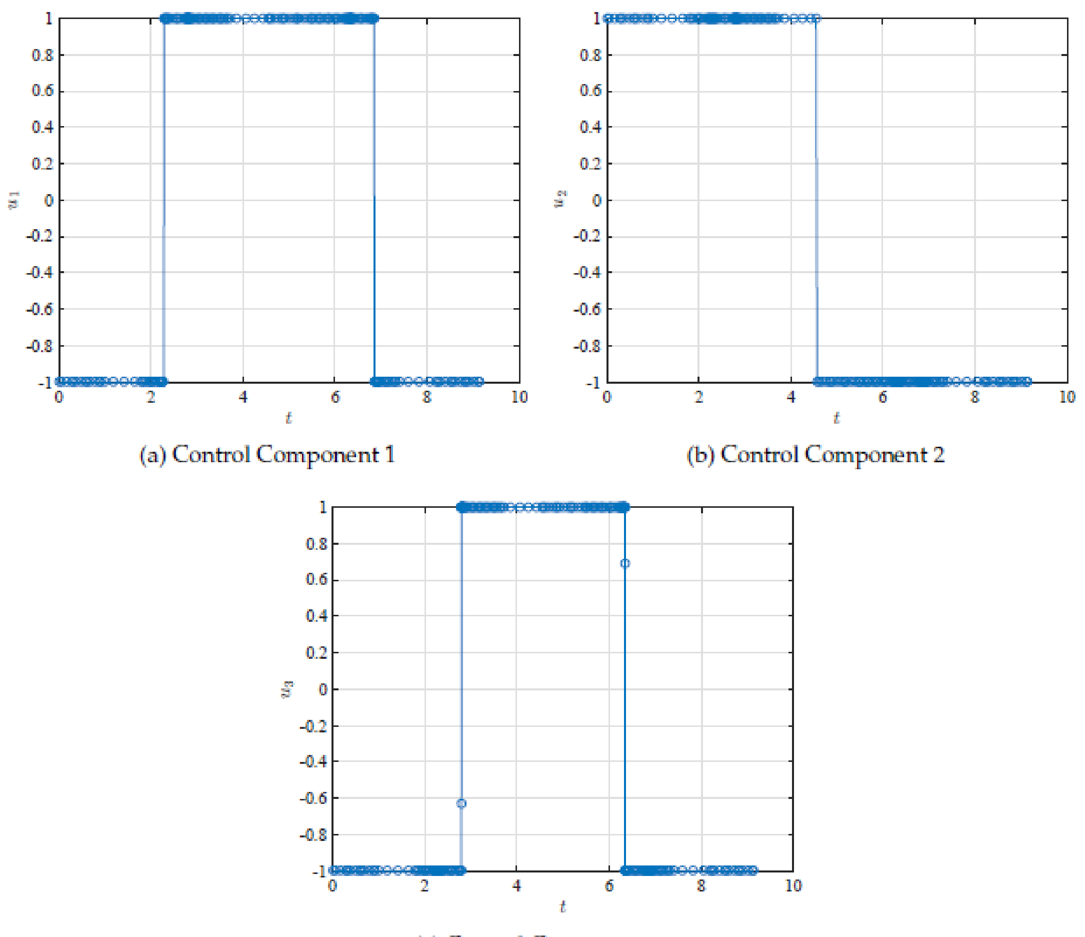


(e) State Component 5



(f) State Component 6

Figure 45. Problem state solution.

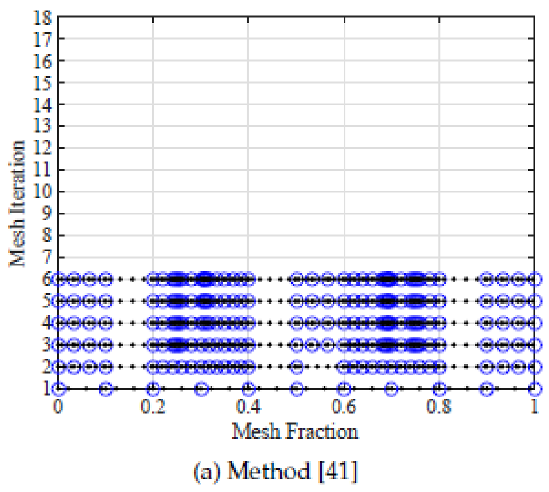


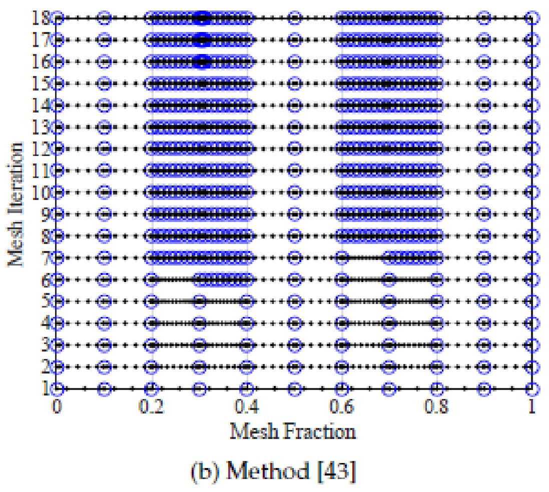
(a) Control Component 1

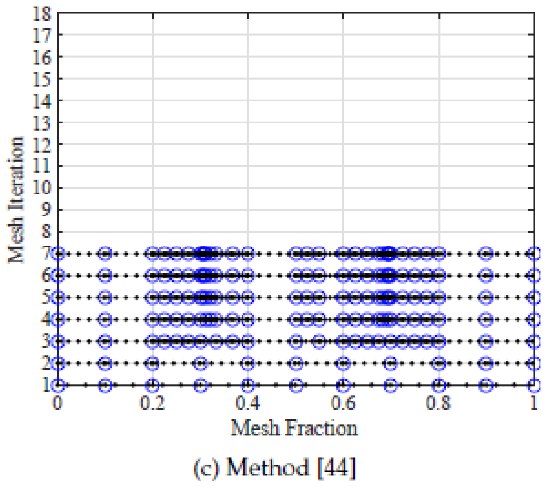
(b) Control Component 2

(c) Control Component 3

Figure 46. Problem control solution.







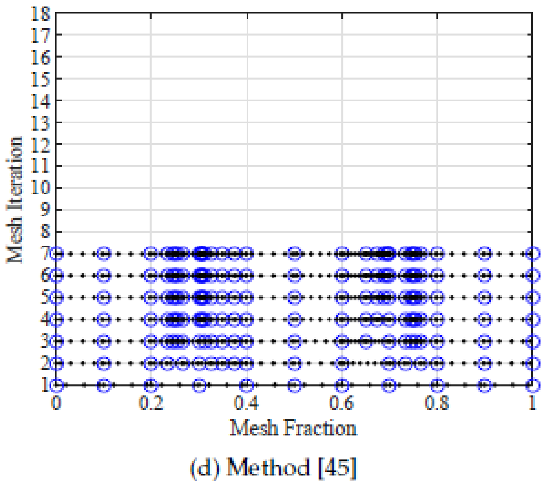


Figure 47. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

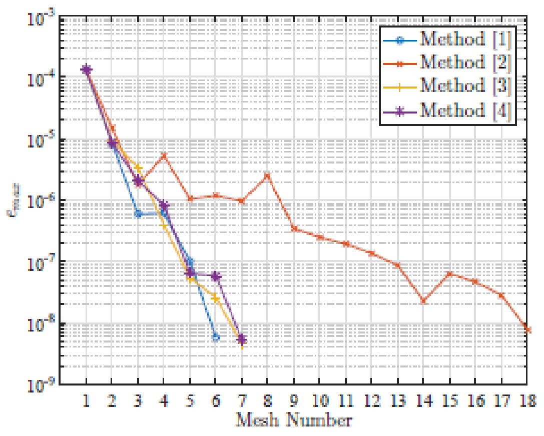


Figure 48. Mesh maximum relative error for each mesh.

4.3.13 Seywald Minimum Time-to-Climb

Minimize the cost functional:

$$J = t_f$$

subject to the dynamic constraints

$$\begin{gathered}\dot{h} = v \sin\gamma \\ \dot{v} = \frac{T - D}{m} - g \sin\gamma \\ \dot{\gamma} = g \frac{u - \cos\gamma}{v}\end{gathered}$$

the boundary conditions

$$\begin{gathered}h(0) = 0, \qquad h(t_f) = 19995\, m \\ v(0) = 129.314, \qquad v(t_f) = 295.092\, m\, s^{-1} \\ \gamma(0) = 0\, deg, \qquad \gamma(t_f) = 0\, deg\end{gathered}$$

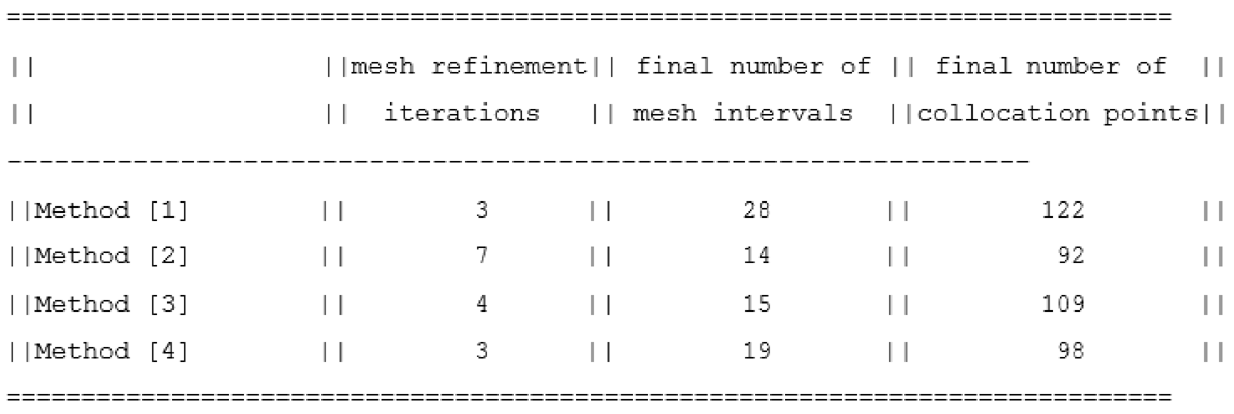
and the inequality path constraint

$$h \geq 0$$

4.3.13.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier = 'sparseCD'
       +- derivative level = 'second'
       +- dependencies = 'sparseNaN'
scales
       +- method = 'none'
method = 'RPM-Differentiation'
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 25
       +- colpointsmin = 4
       +- colpointsmax = 10
initial mesh characteristics
       +- Phase 1
       * fraction = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver = ipopt
       +- linear solver = ma57
       +- tolerance = 1.000000e-07
       +- max iterations = 2000
display level = 0
```

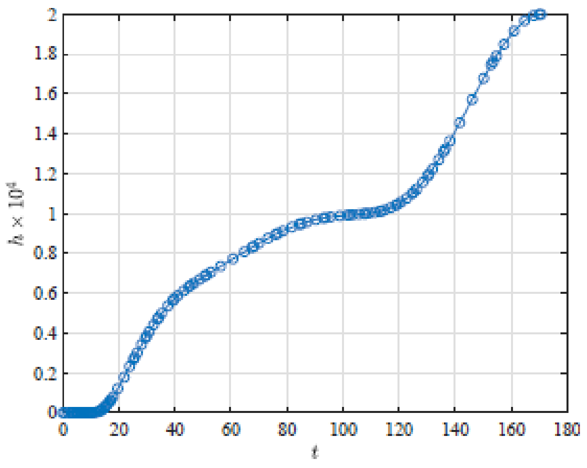
Table 14. Key performance measures for the seywald minimum time to climb problem.



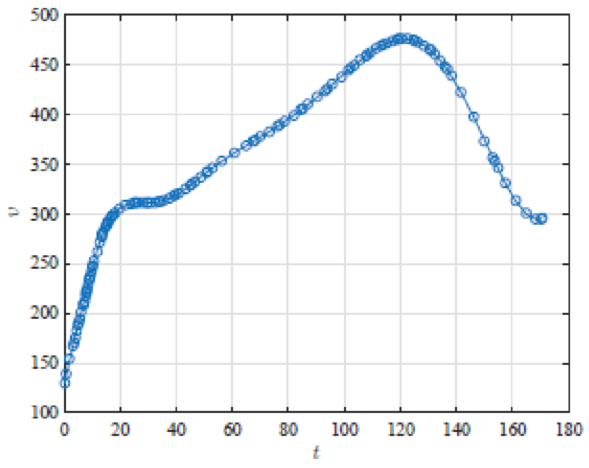
```
============================================================================
||                ||mesh refinement|| final number of || final number of  ||
||                ||  iterations   || mesh intervals  ||collocation points||
------------------------------------------------------------------
||Method [1]      ||       3       ||       28        ||       122        ||
||Method [2]      ||       7       ||       14        ||        92        ||
||Method [3]      ||       4       ||       15        ||       109        ||
||Method [4]      ||       3       ||       19        ||        98        ||
============================================================================
```

4.3.13.2 Results Discussion

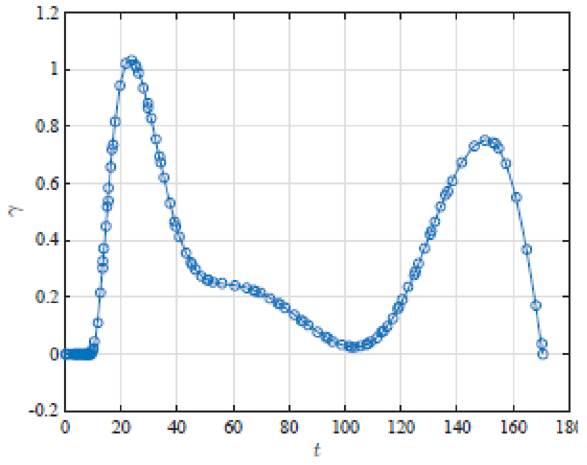
There is one key solution characteristic in this problem; there is one discontinuity in the first derivative of the control as seen in Figure 49. The hp-adaptive methods of Ref. [41] and Ref. [45] both solve the seywald minimum time-to-climb problem in three mesh refinement iterations; however, the hp-adaptive method of Ref. [45] solves slightly faster than the hp-adaptive method of Ref. [41]. This is due to the smaller mesh size, where the hp-adaptive method of Ref. [45] added collocation points instead of adding mesh intervals like the hp-adaptive method of Ref. [41], a primarily h method. The hp-adaptive method of Ref. [43] uses its collocation points most efficiently, using fewer points than the other methods. However, it takes the longest to solve because it takes longer for the method to divide the interval into enough subintervals to bracket the discontinuity.



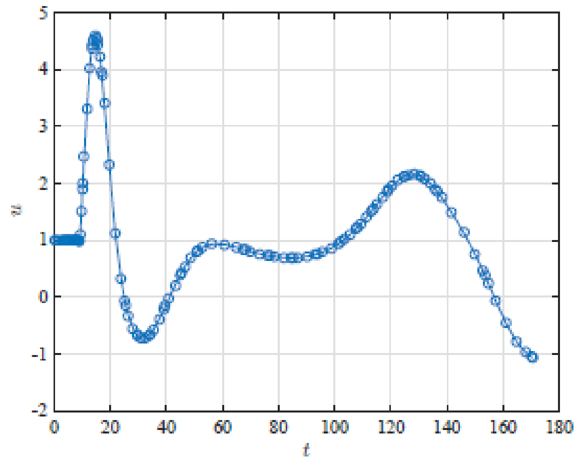
(a) State Component 1



(b) State Component 2



(c) State Component 3



(d) Control

Figure 49. Problem state and control solution.

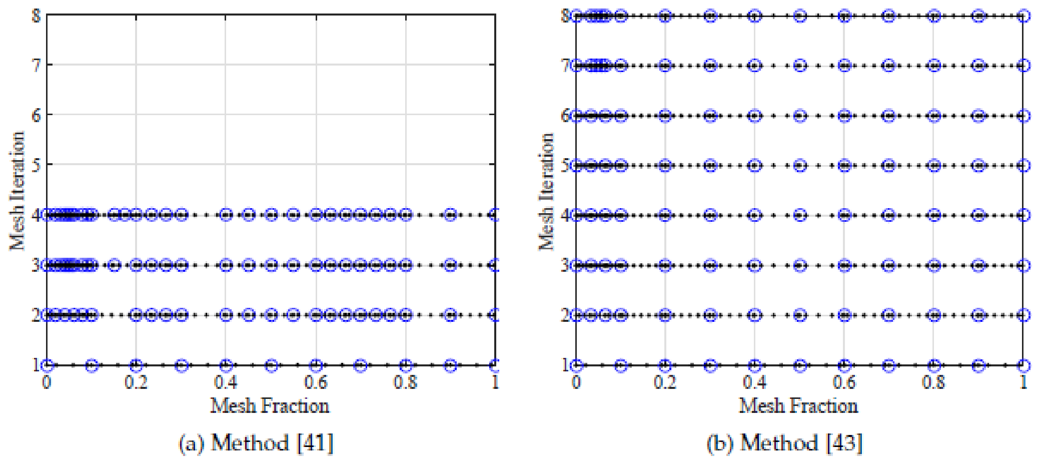


Figure 50. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

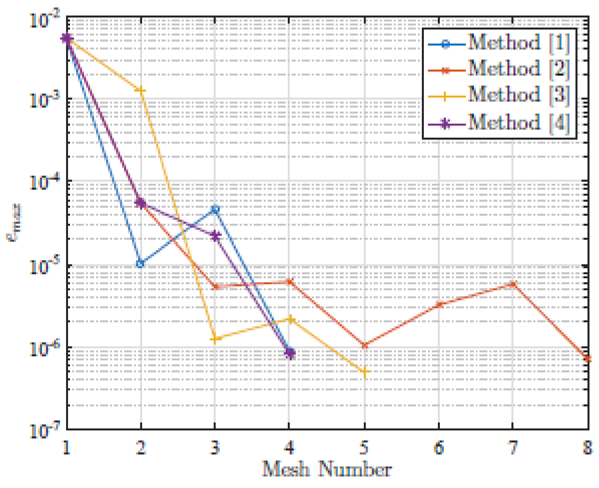


Figure 51. Mesh maximum relative error for each mesh.

4.3.14 Space Station Attitude Reorientation

Minimize the cost functional:

$$J = \frac{1}{2}\int_0^{t_f} u^T u \, dt$$

subject to the dynamic constraints

$$\begin{gathered}\dot{\omega} = J^{-1}\{T_{gg}(r) - \omega \times [J\omega + h] - u\} \\ \dot{r} = \frac{1}{2}[rr^T + I + r^{\otimes}][\omega - \omega(r)] \\ \dot{h} = u\end{gathered}$$

and the boundary conditions

$$\begin{gathered}\omega(0) = 0, \qquad h(t_f) = 19994.88\, m \\ r(0) = 129.314, \qquad v(t_f) = 295.092\, m\, s^{-1} \\ h(0) = 0, \qquad \gamma(t_f) = 0 \\ t_f = 1800\end{gathered}$$

and the control inequality constraint

$$\|h\|_2 \le h_{max}$$

4.3.14.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier   = 'sparseCD'
       +- derivative level = 'second'
       +- dependencies = 'sparseNaN'
scales
       +- method = 'automatic-bounds'
method = 'RPM-Integration'
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 25
       +- colpointsmin = 4
       +- colpointsmax = 20
initial mesh characteristics
       +- Phase 1
       * fraction = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver     = ipopt
       +- linear solver     = ma57
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

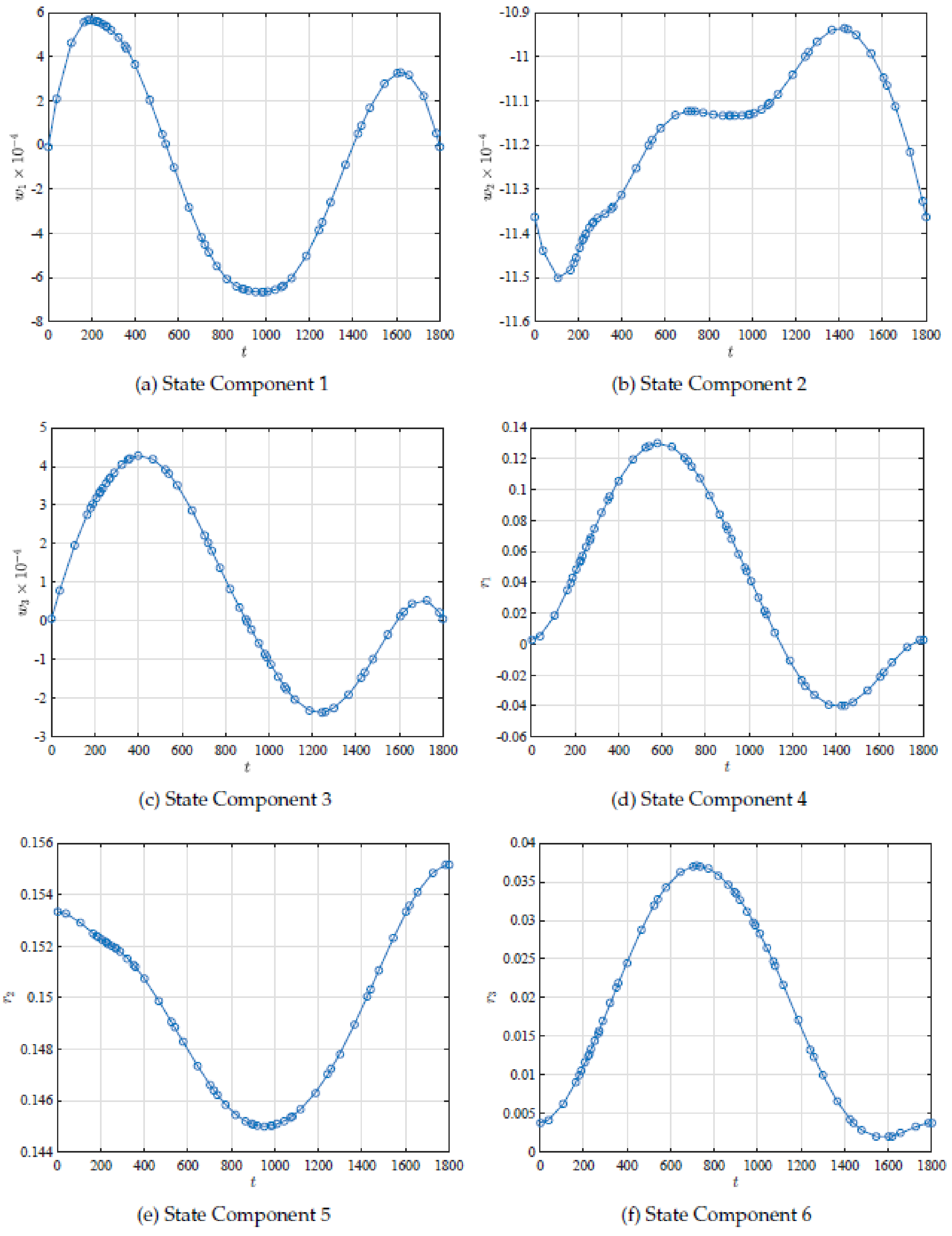
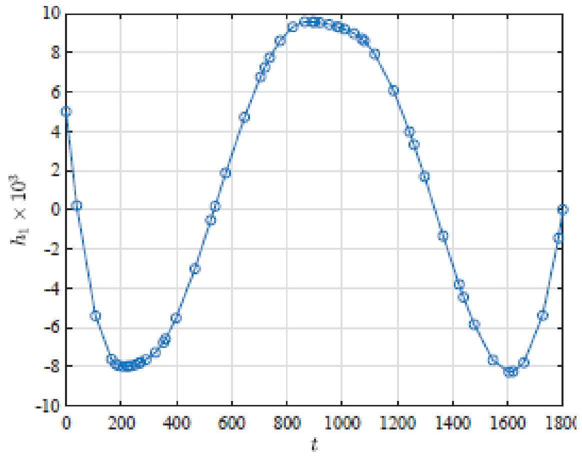
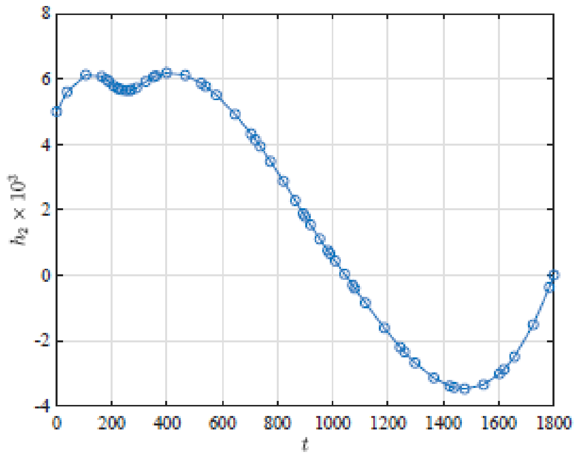


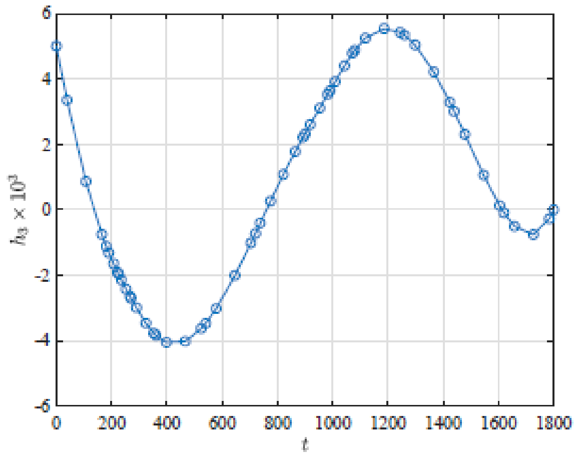
Figure 52. Problem state solution.



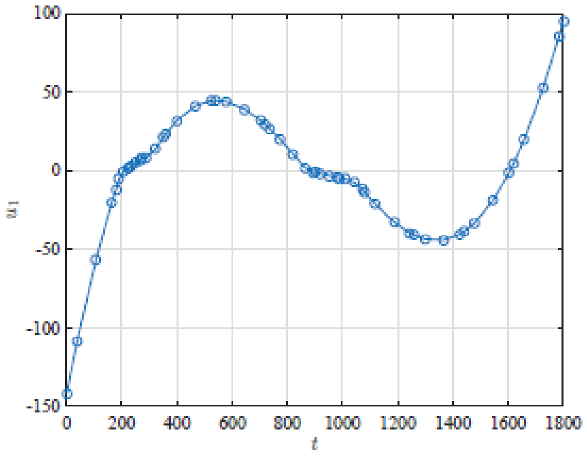
(a) State Component 7



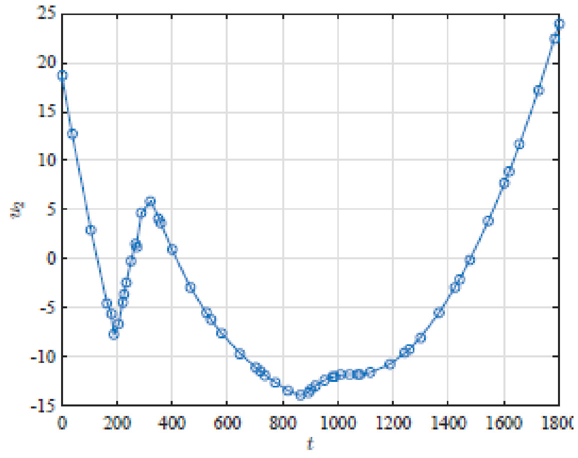
(b) State Component 8



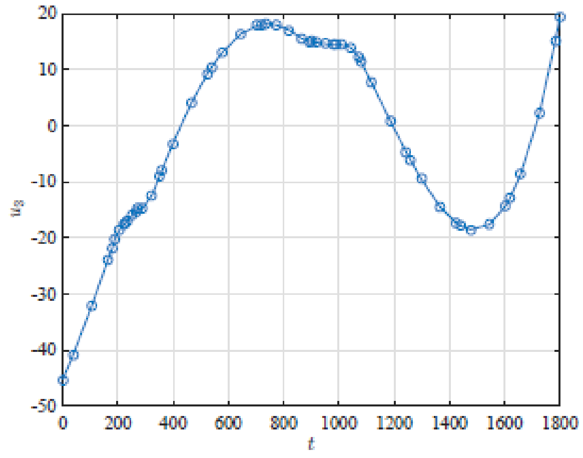
(c) State Component 9



(d) Control Component 1



(e) Control Component 2



(f) Control Component 3

Figure 53. Problem state and control solution.

Table 15. Key performance measures for the space station problem.

	mesh refinement iterations	final number of mesh intervals	final number of collocation points
Method [1]	2	13	54
Method [2]	1	10	46
Method [3]	1	10	49
Method [4]	2	14	56

4.3.14.2 Results Discussion

The solution of this problem is smooth, as seen in Figures 52 and 53. Therefore, a p method will be ideal. The hp-adaptive methods of Ref. [43] and Ref. [44] both solve the space shuttle problem in one mesh refinement. Both methods have similar mesh sizes and computational time; however, the hp-adaptive method of Ref. [44] is slightly faster than the hp-adaptive method of Ref. [43] even though the hp-adaptive method of Ref. [43] has fewer collocation points. The hp-adaptive methods of Ref. [41] and Ref. [45] solve the problem in two mesh refinements. They have similar mesh sizes and computational times. The difference in computational times for each method, seen in Table 15 as less than one second, is not significant due to the lack of key solution characteristics that might present difficulty in solving the problem.

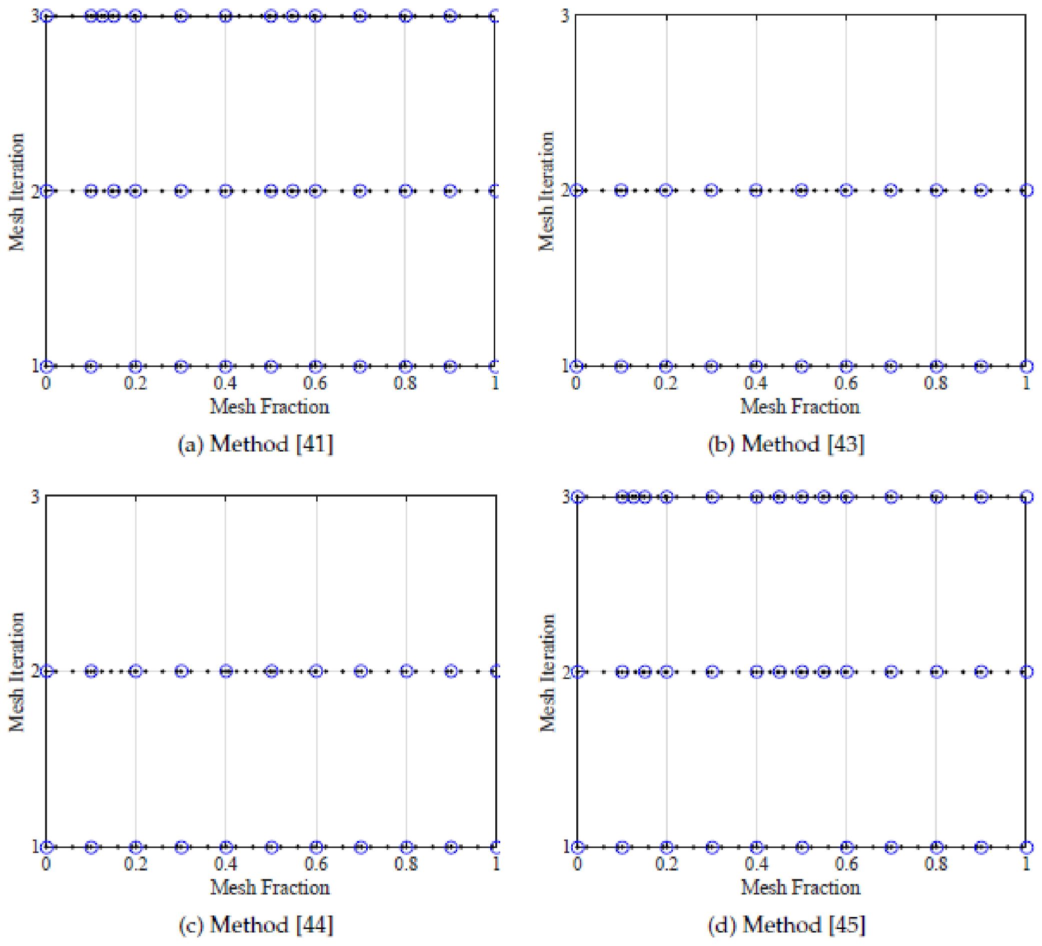


Figure 54. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

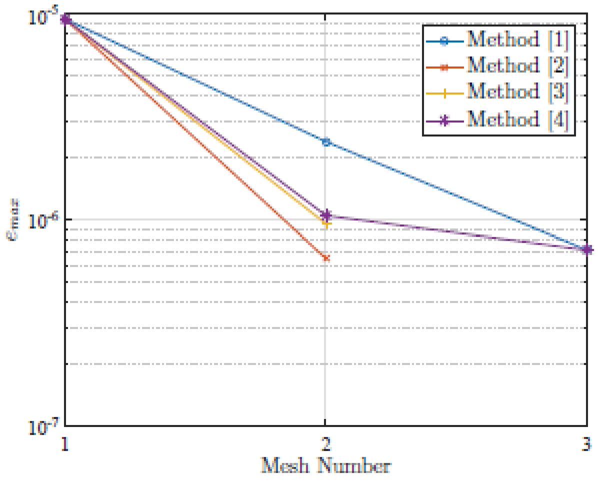


Figure 55. Mesh maximum relative error for each mesh.

4.3.15 Tuberculosis

4.3.15.1 Problem Statement

Minimize the cost functional:

$$J = \int_0^{t_f} \left(L_2 + I_2 + \frac{1}{2} B_1 u_1^2 + B_2 u_2^2 \right) dt$$

subject to the dynamic constraints

$$\dot{S} = \Lambda - \beta_1 S \frac{I_1}{N} - \beta^* S \frac{I_2}{N} - \mu S$$
$$\dot{L}_1 = \beta_1 S \frac{I_1}{N} - (\mu + k_1) L_1 - u_1 r_1 L_1 + (1 - u_2) p r_2 I_1 + \beta_2 T \frac{I_1}{N} - \beta^* L_1 \frac{I_2}{N}$$
$$\dot{I}_1 = k_1 L_1 - (\mu + d_1) I_1 - r_2 I_1$$
$$\dot{L}_2 = (1 - u_2) q r_2 I_1 - (\mu + k_2) L_2 + \beta^* (S + L_1 + T) \frac{I_2}{N}$$
$$\dot{I}_2 = k_2 L_2 - (\mu + d_2) I_2$$
$$\dot{T} = u_1 r_1 L_1 + u_2 (p + q) r_2 I_1 - \beta_2 T \frac{I_1}{N} - \beta^* T \frac{I_2}{N} - \mu T$$

the boundary conditions

$$S(0) = 19000$$
$$L_1(0) = 9000$$
$$L_2(0) = 500$$
$$I_1(0) = 1000$$
$$I_2(0) = 250$$
$$T(0) = 250$$

and the state equality path constraint

$$S + L_1 + I_1 + L_2 + I_2 + T - N = 0$$

4.3.15.2 GPOPS-II Setup

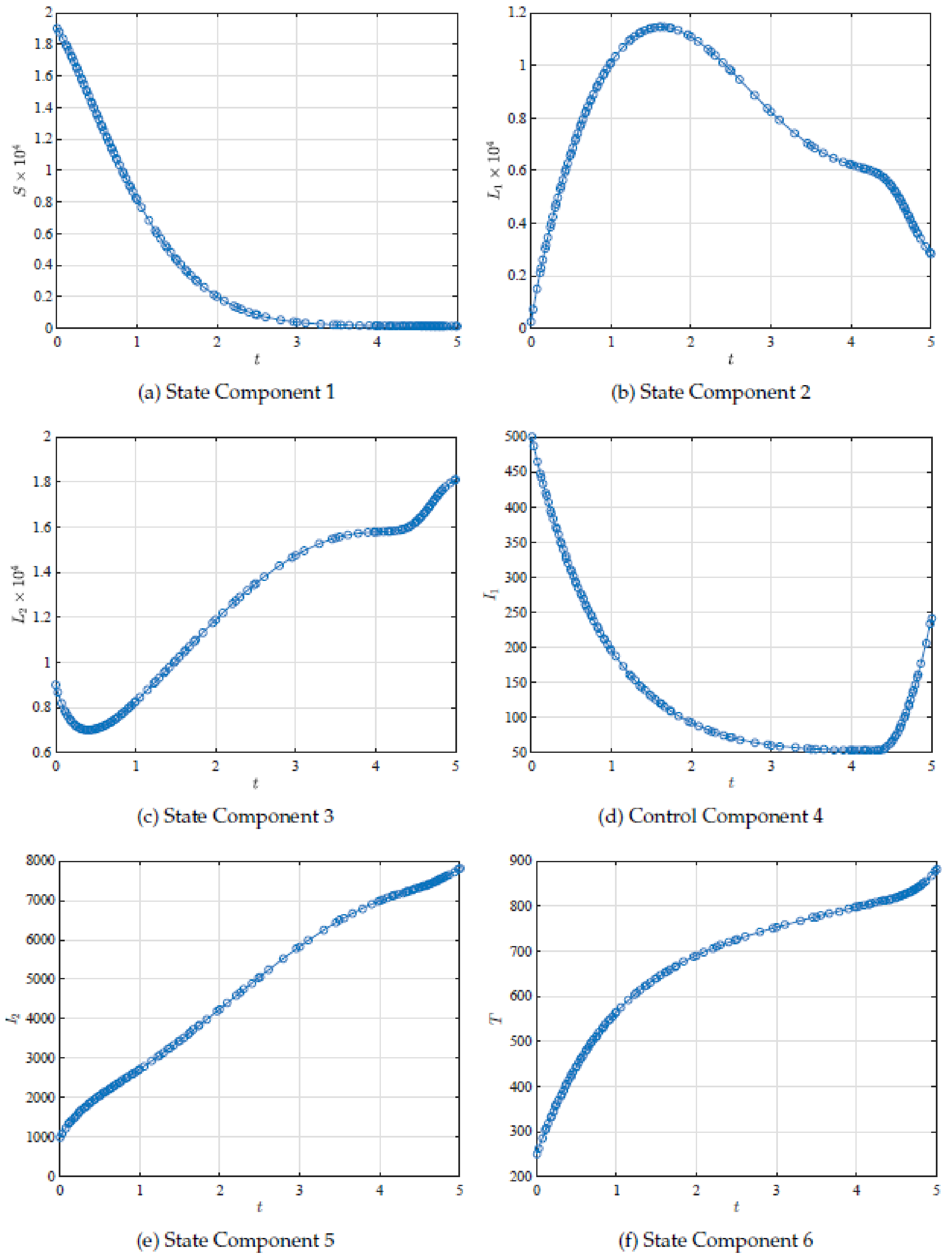
```
setup details:
derivatives
       +- supplier = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies = ’sparseNaN’
scales
       +- method = ’automatic-bounds’
method = ’RPM-Integration’
mesh
       +- tolerance = 1.000000e-06
       +- max iterations = 10
       +- colpointsmin = 3
       +- colpointsmax = 20
initial mesh characteristics
       +- Phase 1
       * fraction = [0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver = ipopt
       +- linear solver = mumps
       +- tolerance = 1.000000e-07
       +- max iterations = 2000
display level = 0
```

4.3.15.3 Results Discussion

Figure 57 shows the three discontinuities in the first derivative of the control. Each mesh refinement’s actions around these discontinuities can be seen in Figure 58. The hp-adaptive method of Ref. [44] solves the tuberculosis problem in two mesh refinement iterations. The hp-adaptive method of Ref. [43] has the smallest mesh size, takes four mesh iterations, and ranks third in computational time. This is due to being a primarily p method. The hp-adaptive methods of Ref. [44] and Ref. [45] divide the mesh intervals when the hp-adaptive method of Ref. [43] increases the degree of the approximating polynomial in the mesh interval that contains the discontinuity. While this leads to the hp-adaptive method of Ref. [43] having a smaller mesh size, the two other methods are able to converge to a solution quicker.

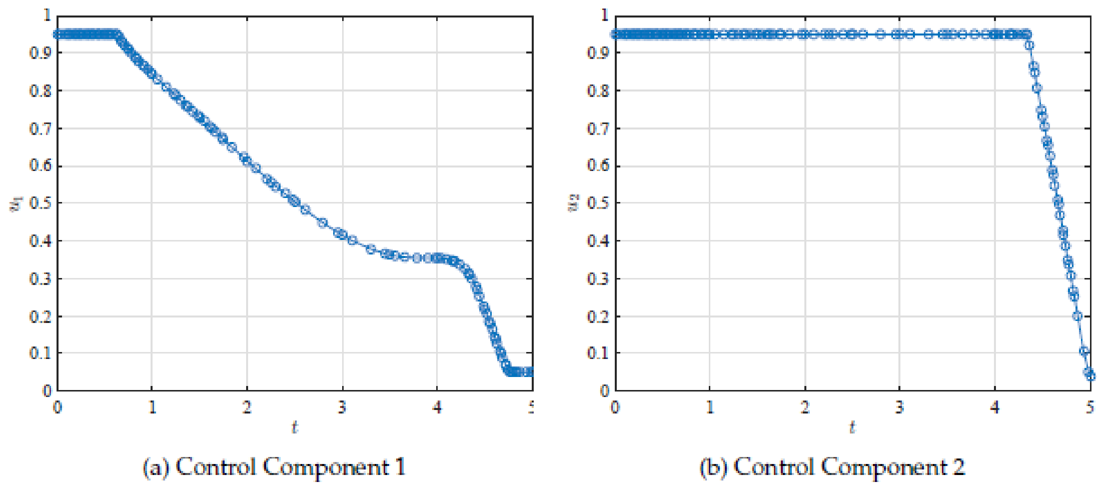
Table 16. Key performance measures for the tuberculosis problem.

	mesh refinement iterations	final number of mesh intervals	final number of collocation points
Method [1]	4	36	120
Method [2]	4	10	63
Method [3]	2	12	83
Method [4]	3	16	84



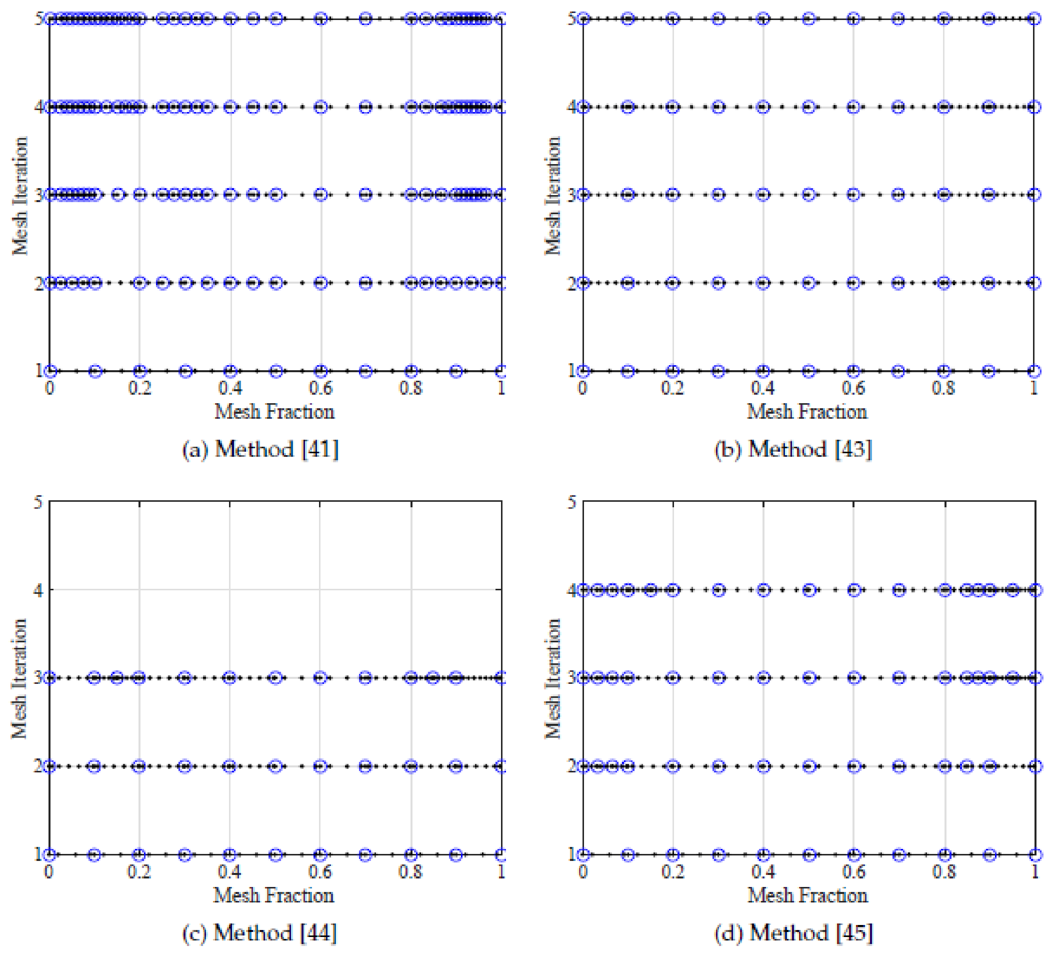
(a) State Component 1 (b) State Component 2 (c) State Component 3 (d) Control Component 4 (e) State Component 5 (f) State Component 6

Figure 56 Problem state solution.



(a) Control Component 1 (b) Control Component 2

Figure 57. Problem control solution.



(a) Method [41] (b) Method [43]

(c) Method [44] (d) Method [45]

Figure 58. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

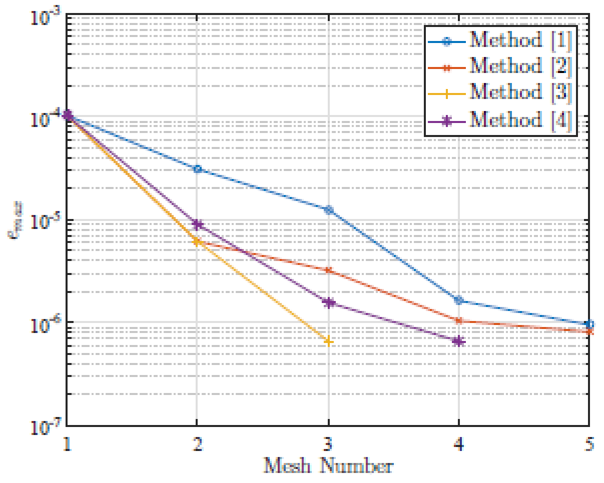


Figure 59. Mesh maximum relative error for each mesh.

4.3.16 Tumor Anti-Angiogenesis

Minimize the cost functional:

$$J = p(t_f)$$

Subject to the dynamic constraints

$$\dot{p} = -(\varsigma p)\log\left(\frac{p}{q}\right)$$
$$\dot{q} = q\left(b - \mu - dp^{2/3} - Gu\right)$$

the boundary conditions

$$p(0) = 0.5\left(\frac{b-\mu}{d}\right)^{\frac{3}{2}}$$
$$q(0) = 0.25\left(\frac{b-\mu}{d}\right)^{\frac{3}{2}}$$

the integral constraint

$$\int_0^{t_f} u(t)\, dt \leq 15$$

and the control inequality constraint

$$0 \leq u(t) \leq 75$$

where

$$\varsigma = 0.084\ day^{-1}$$

4.3.16.1 GPOPS-II Setup

```
setup details:
derivatives
       +- supplier = ’sparseCD’
       +- derivative level = ’second’
       +- dependencies = ’sparseNaN’
scales
       +- method = ’none’
method = ’RPM-Differentiation’
mesh
       +- tolerance  = 1.000000e-06
       +- max iterations = 25
       +- colpointsmin      =      3
       +- colpointsmax      = 10
initial mesh characteristics
       +- Phase 1
       * fraction = [1 1 1 1 1 1 1 1 1 1]/10
       * colpoints = [4 4 4 4 4 4 4 4 4 4]
nlp
       +- solver = ipopt
       +- linear solver = mumps
       +- tolerance  = 1.000000e-07
       +- max iterations = 2000
display level =      0
```

4.3.16.2 Results Discussion

The tumor anti-angiogenesis problem is a bit unusual, because the solution IPOPT yields on intermediate meshes do not always reflect the behavior of the actual solution to the problem.

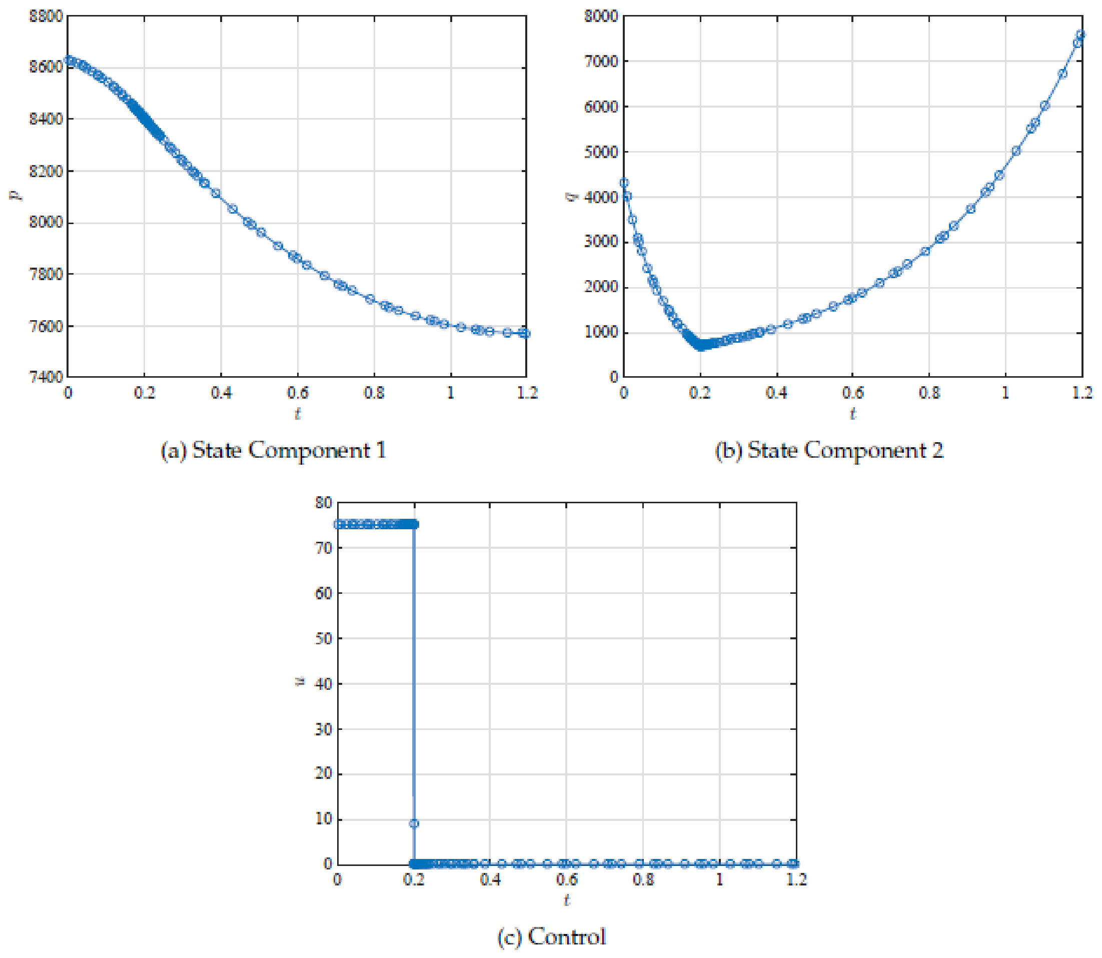
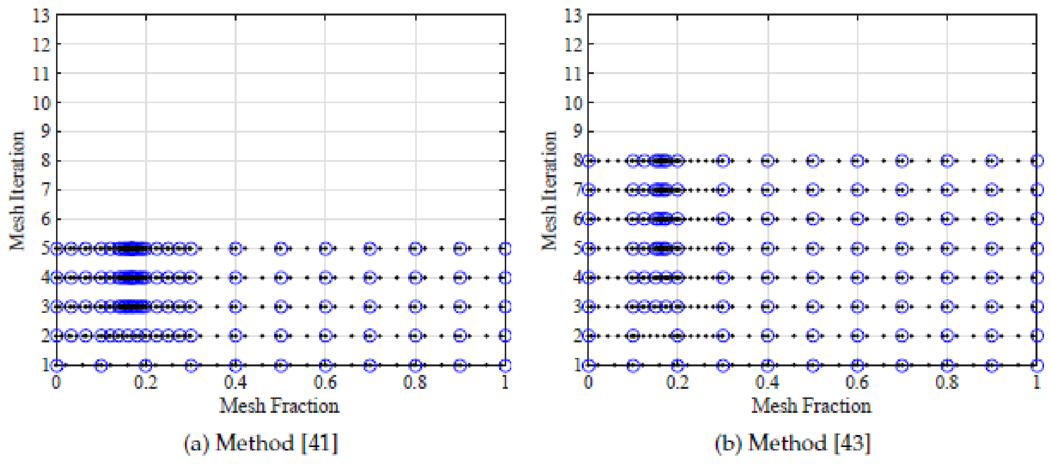


Figure 60. Problem state and control solution.

Nonetheless, it is an important problem, because it demonstrates the behavior of each mesh refinement method when less-than-perfect information is available from the NLP solution. Figure 60 shows a single discontinuity in both the control and the state derivative, which defines the key features of this problem. All four mesh refinement methods concentrate mesh points around the discontinuity location as the mesh refinements progress towards convergence, as seen in Figure 61. In particular, the hp-adaptive method of Ref. [41] concentrated mesh points around the discontinuity location much faster than the other methods and stands out as the fastest method for this problem. The hp-adaptive method of Ref. [41] used the least number of mesh refinement iterations, but had the largest final mesh size. The hp-adaptive methods of Ref. [43], Ref. [44], and Ref. [45] had similar (but slower) computation times when compared to the hp-adaptive method of Ref. [41], but they also had the most efficient usage of collocation points of the four methods. Finally, the hp-adaptive method of Ref. [45] took an unusually large number of mesh refinement iterations to converge to the solution, making it the slowest method overall.

Table 17. Key performance measures for the tumor anti-angiogenesis problem.

	mesh refinement iterations	final number of mesh intervals	final number of collocation points
Method [1]	4	31	103
Method [2]	7	16	78
Method [3]	5	15	73
Method [4]	12	21	94



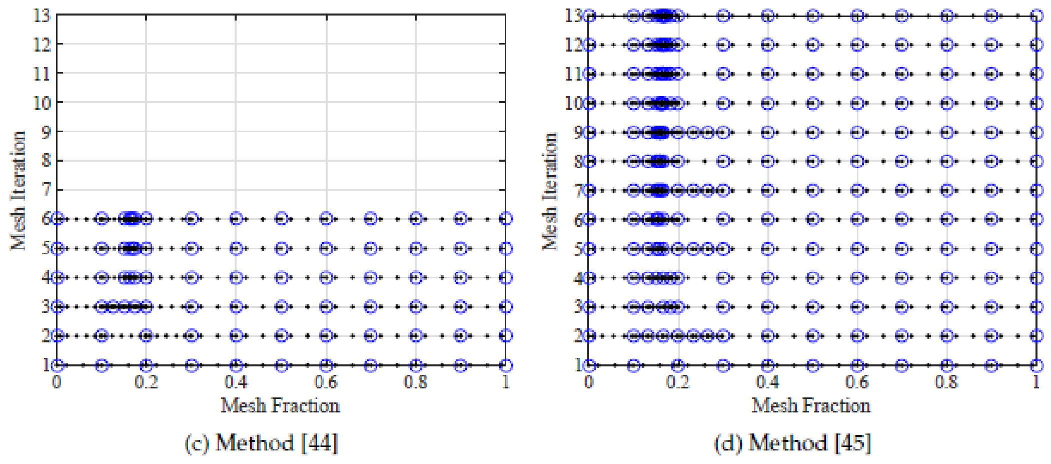


Figure 61. Mesh refinement history for each method. Note that the blue circles and black dots represent mesh points and collocation points, respectively.

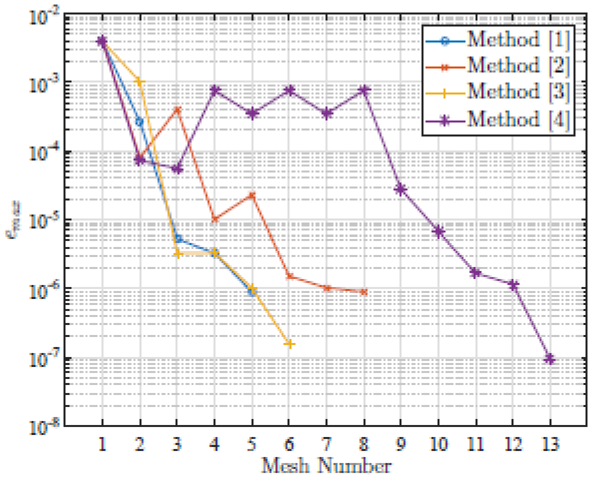


Figure 62. Mesh maximum relative error for each mesh.

4.3.17 Summary of Results

In Section 7.2, the relative performances of each mesh refinement method were discussed on a problem-by-problem basis. Here, general trends and recurring behaviors of the mesh refinement methods are explored. The origins of common features that arise in the solutions of the example problems in Section 4.3.2 are investigated as well. A few qualitative statements can be made regarding the final mesh size obtained using each of the methods. First, the hp-adaptive method of Ref. [41] generally had the largest final mesh size and the hp-adaptive method of Ref. [43] generally had the smallest final mesh size. Since the hp-adaptive method of Ref. [43] is primarily a p refinement method and the hp-adaptive method of Ref. [41] is primarily an h refinement method, this appears to indicate that p refinement tends to lead to a more efficient use of collocation points and mesh intervals. Another insight is the effectiveness of reducing the final mesh size by merging mesh intervals and lowering the polynomial approximation degree on intervals that already meet the maximum mesh relative error tolerance. Both the hp-adaptive methods of Ref. [44] and Ref. [45] actively seek to reduce the mesh size, and the impact is quite noticeable on problems like the hypersensitive problem. The computational performance of each method was largely problem-dependent, and each method had at least one problem where it converged to the solution the fastest. However, both of the hp-adaptive methods of Ref. [44] and Ref. [45] performed well in a wide breadth of problems (with the hp-adaptive method of Ref. [44] performing best overall) when compared with the other methods, and their final cpu times were typically on par with or better than those of the hp-adaptive methods of Ref. [43] or Ref. [41]. One exception is the tumor anti-angiogenesis problem, where the hp-adaptive method of Ref. [45] took an unusually large number of refinement iterations relative to the others in order to converge to a solution, thereby yielding a slow final computation time compared to the other methods. Gains in computational efficiency in each method appear to be a balancing act between keeping the mesh size on each iteration small (and therefore limit the growth in size of the NLP) and converging to the solution in the fewest number of iterations (keeping the number of NLPs that need to be solved small). Most important in the mesh size/number of refinements balancing act is converging to the solution in the fewest number of iterations, because the extra time needed to solve an extra NLP is generally greater than the time penalty incurred by solving a slightly larger NLP (but one less total number of NLPs solved). In fact, most of the example problems had the same method achieve the fastest computation time and the fewest number of mesh refinements used. Notable exceptions include the hypersensitive, moon lander, and minimum-time reorientation problems where the fastest computation time was achieved by the method with the second smallest number of mesh refinement iterations, but which had a smaller number of collocation points than the method that took the fewest number of mesh refinement iterations. On the Bryson-Denham, orbit-raising, and maximum cross range reusable launch vehicle entry problems, the hp-adaptive methods of Ref. [43] and Ref. [44] achieved the fastest convergence time, as well as the fewest number of mesh refinement iterations while using the smallest number of collocation points. In particular, the hp-adaptive method of Ref. [43] did so on the orbit-raising and maximum cross range reusable launch vehicle entry problems, which both have smooth solutions for the state (no discontinuities in any state component derivative or higher order derivative). The hp-adaptive method of Ref. [43]

appears to excel at solving problems with smooth solutions, but the method tends to get stuck repetitively p refining and then splitting on problems that contain discontinuities.

Discontinuities are a bane to mesh refinement methods generally, and arise quite frequently when solving optimal control problems. In fact, two common solution features that arise in nine out of the fourteen example problems investigated in this report are discontinuities in the control or control derivative. Discontinuities in the control caused corresponding discontinuities in the state first derivative, whereas discontinuities in the control derivative tended to cause discontinuities in the higher order derivatives of the state. Nonsmooth behavior in the state is not well approximated by a smooth polynomial approximation. For this reason, methods such as the hp-adaptive method of Ref. [43] (which tend to utilize p refinement on mesh intervals containing a nonsmooth state) tend to perform poorly when compared with methods that utilize h refinement on non-smooth intervals (assuming those methods are correctly able to identify the non-smooth interval). However, none of the four mesh refinement methods are capable of directly locating discontinuities. Therefore, each method’s h refinement approach is to split a particular mesh interval into evenly spaced parts rather than split directly at the discontinuity point (if it exists). Many h refinements may be needed before the piecewise smooth behavior of the state solution is well approximated by appropriately placed piecewise polynomials, causing mesh points to accrue near the discontinuity location(s). An accumulation of mesh points near a discontinuity is undesirable, because (theoretically) the solution could be approximated by splitting only once at the exact discontinuity location, thereby using a much smaller mesh to solve the problem to the same accuracy. Several factors can alert oneself to the possibility of a control discontinuity or control derivative discontinuity arising in the solution. Pontryagin’s Minimum Principle states

$$u^* = arg \min_{u \epsilon \mathcal{U}} \mathcal{H}(x^*, u, \lambda^*) \tag{60}$$

where $\{x^*, u^*, \lambda^*\}$ are the optimal state, control, and costate respectively, $\mathcal{H} = \mathcal{L} + \lambda^T a$ is the Hamiltonian, and $\mathcal{U}$ is the feasible set of solutions for the control. In the particular case where the Hamiltonian is linear in one or more control component(s), the Hamiltonian is minimized when u^* takes on its maximum or minimum feasible value depending on whether the coefficient in front of it is negative or positive (ignoring the case where the coefficient is zero). Therefore, the control solution structure for problems that have a Hamiltonian which is linear in one or more control components is oftentimes bang-bang. Observing the moon lander, minimum-time reorientation, robot arm, and tumor anti-angiogenesis problems, it is seen that the dynamics, a, and corresponding Hamiltonian were all linear in the control components that had bang-bang solution structures.

Another observed characteristic of the non-smooth control solutions is that the discontinuity points come paired with a switch on/off the bounds of the state, control, or path constraint. In the Bryson Denham, Bryson minimum time-to-climb, and Seywald minimum time-to-climb problems, the switching point where a state component transitions on or off of its upper or lower bounds is precisely located where the control first derivative discontinuity lies. Similarly, the tuberculosis problem has control derivative discontinuities where the control components transition on or off of one of its bounds, and the dynamic soaring problem has first derivative control discontinuities precisely where the path constraint transitions between active and inactive.

The prevalence of nonsmooth solutions to optimal control problems, as well as their negative effects on the performances of the mesh refinement methods discussed in this report calls for more research into methods for properly handling discontinuities when refining the mesh. Future mesh refinement methods must be able to precisely and accurately locate discontinuities in the control and control derivative so that targeted h refinement action can be taken to mitigate the effect of the discontinuity on solution accuracy. Combining a method such as the hp-adaptive method of Ref. [43] (which currently tends to perform well on problems with smooth solutions) with an h refinement at discontinuities algorithm has the potential to produce a "smart" mesh refinement algorithm. Such an algorithm would gain the benefit of convergence speed and small mesh size on smooth segments by using the hp-adaptive method of Ref. [43], but would not be hindered (or at least less hindered) by the presence of discontinuities in the solution. The hp-adaptive methods of Ref. [44] and Ref. [45] attempt such an approach (which may be the reason they tend to perform well on most of the example problems within this report), but both methods split non-smooth mesh intervals into evenly spaced sub-intervals rather than splitting the mesh directly at the discontinuity point.

4.3.18 Conclusions

Four mesh refinement methods used within the software GPOPS-II have been tested on a battery of problems to benchmark their performance and to identify key problem characteristics that influence the behaviors of each mesh refinement method. The hp-adaptive method of Ref. [43] tended to use collocation points most efficiently, obtaining the smallest final mesh size on many of the test problems. On the other hand, the hp-adaptive method of Ref. [41] frequently converged to the solution with the largest number of collocation points and/or mesh intervals. Both the hp-adaptive methods of Ref. [44] and Ref. [45] were effective in obtaining fast computation times and small final mesh sizes on the widest array of problems. It was observed that the hp-adaptive method of Ref. [43] tended to work best on problems with smooth solutions, and that the presence of discontinuities in the control or control derivative caused this method in particular to be hindered more than the rest. Non-smooth solution behavior and its root causes were discussed, and ideas for future mesh refinement methods that could handle discontinuities were discussed.

4.4 Application to High-Speed Ascent and Entry

The advancements of the computational framework for optimal control are now applied to a high-speed ascent and entry optimal control problem. The problem of Low-Earth Orbit (LEO) ascent and entry mission design has reemerged since the decommissioning of the U.S. Space Shuttle. A particular problem of interest is one where an entire mission, from launch to landing, is solved. Such a problem consists of powered ascent phases, a powered de-orbit phase, and unpowered entry phases. The primary focus during ascent is to minimize the fuel consumed (that is, maximize the payload to orbit) while a key consideration during entry is to maximize the time elapsed after de-orbit. Because each of these regimes of flight has distinctly different physical characteristics, determining complete ascent-entry mission plans is a computational challenge. Due to the importance of optimizing the motion during ascent and entry, such problems are posed as multiple-phase optimal control problems. Because of the complexity of such problems (nonlinear dynamics, phases of flight with different physical characteristics), they must be solved numerically. The problem of interest here is a combined ascent-entry optimal mission planning and is posed as a multiple-phase optimal control problem consisting of powered ascent phases, a powered de-orbit phase, and unpowered entry phases. The goal is to maximize a combination of the payload to orbit and the time elapsed since atmospheric entry after reaching orbit. The multiple-phase optimal control problem is solved using a Legendre-Gauss-Radau collocation method. The results provide insight into the structure of solutions for a mission that consists of a combination of ascent and entry phases and demonstrate the computational efficiency and accuracy of Legendre-Gauss-Radau collocation.

4.4.1 Ascent-Entry Optimal Mission Planning Problem

In this study, we consider an atmospheric flight vehicle that subtends both ascent and entry phases of flight. The goal is to steer the vehicle from an initial state that corresponds to the time at which the solid rocket motor has been jettisoned to a final TAEM state. In other words, to determine the state and control that steers the vehicle from the initial state to the terminal state while minimizing an objective function that is a linear combination of the payload mass and the time elapsed since atmospheric entry. The mission planning problem is divided into ascent phases consisting of a main engine burn that terminates at Main Engine Cutoff (MECO), a series of two Orbital Maneuvering System (OMS) burns that terminates at the operational orbit, a de-orbit burn followed by a free fall that terminates at atmospheric entry, and an atmospheric entry phase that terminates at the aforementioned TAEM condition. The remainder of this section provides the equations of motion, the specific vehicle model parameters, boundary conditions, path constraints, and phase sequencing that defines the mission.

4.4.2 Equations of Motion

The vehicle under consideration in this study is modeled as a point mass in motion over a spherical rotating Earth. The equations of motion for such a vehicle are modeled in spherical coordinates as [53]

$$
\begin{gathered}
\dot{r} = v \sin\gamma \\
\dot{\theta} = \frac{v\cos\gamma\cos\psi}{r\cos\phi} \\
\dot{\phi} = \frac{v\cos\gamma\sin\psi}{r} \\
\dot{v} = \frac{F_T}{m} - g\sin\gamma + \omega^2 r\cos\phi\,(\sin\gamma\cos(\phi) - \cos\gamma\sin\phi\sin\psi) \\
v\dot{\gamma} = \frac{F_N}{m}w_1 - g\cos\gamma + \frac{v^2}{r}\cos\gamma + 2\omega v\cos\phi\cos\psi + \omega^2 r\cos\phi\,(\cos\gamma\cos\phi + \sin\gamma\sin\phi\sin\psi) \\
v\dot{\psi} = \frac{F_N}{m\cos\gamma}w_2 - \frac{v^2}{r}\cos\gamma\cos\psi\tan\phi + 2\omega v(\tan\gamma\cos\phi\sin\psi - \sin\phi) - \frac{\omega^2 r}{\cos\gamma}\sin\phi\cos\phi\cos\psi \\
\dot{m} = -\frac{T}{g_0 I_{sp}}
\end{gathered}
\tag{61}
$$

where r is the geocentric radius, θ is the Earth-relative longitude, ϕ is the geocentric latitude, v is the Earth-relative speed, γ is the Earth-relative flight path angle, ψ is the heading angle (measured from due East), $g = \mu/r^2$ is the gravitational acceleration, g_0 is the sea-level gravitational acceleration, and I_{sp} is the engine specific impulse. The tangential and normal forces of the combined aerodynamic and propulsive forces, F_T and F_N, respectively, are given as

$$
\begin{aligned}
F_T &= T\cos\in - D \\
F_N &= T\sin\in + L
\end{aligned}
\tag{62}
$$

where T is the thrust, D is the drag, L is the lift (where it is noted that T is zero during non-powered phases), and $\in$ is the thrust angle of attack. The lift and drag forces are given as

$$
\begin{aligned}
L &= \frac{1}{2}pv^2 C_L S \\
D &= \frac{1}{2}pv^2 C_D S
\end{aligned}
\tag{63}
$$

where ρ is the atmospheric density, S is the vehicle reference area, and C_L and C_D are the coefficients of lift and drag, respectively. Models for the lift and drag coefficients were taken from Ref. 54 for the ascent portion of the mission, and from Ref. 21 for the reentry portion of the mission. For both ascent and entry the control is parameterized using the variables α, w_1, and w_2 where α is the angle of attack, $w_1 = \cos\sigma$, and $w_2 = \sin\sigma$ (where σ is the bank angle). Because σ is replaced in favor of (w_1, w_2), the path constraint

$$
w_1^2 + w_2^2 = 1 \tag{64}
$$

is imposed in order to ensure that (w_1, w_2) is a unit vector, and it is noted that

$$\sigma = \tan^{-1}(w_2, w_1) \tag{65}$$

where $\tan^{-1}(.,.)$ is the four-quadrant inverse tangent. It is noted for this study that $\epsilon = a$. Furthermore, the pressure, p, the density, ρ, and the speed of sound were obtained from a linear interpolation of the 1962 US Standard Atmosphere model [55]. The physical constants used in this study are listed in Table 20. Refer to Refs. 54 and 21 for lift, drag, and heating coefficients. Finally, it is noted that, when convenient, the abbreviation $x(t)$ will be used to denote the state, $(r(t), \theta(t), \phi(t), v(t), \gamma(t), \psi(t), m(t))$, while the abbreviation $u(t)$ will be used to denote the control $(\alpha(t), w_1(t), w_2(t))$ (or, equivalently, $(\alpha(t), \sigma(t))$, where $\sigma(t)$ is computed using Eq. 65).

Table 18. Constants.

Quantity	Value
A_e^m	$4.17055\ \mathrm{m}^2$
F_{vac}^o	$26690\ \mathrm{N}$
F_{vac}^m	$2090660\ \mathrm{N}$
g_0	$9.8066498\ \mathrm{m \cdot s^{-2}}$
H	$7254.2\ \mathrm{m}$
I_{sp}^m	$316\ \mathrm{s}$
I_{sp}^o	$455.15\ \mathrm{s}$
K_{max}^m	1.09
m_{ET}	$26535.1\ \mathrm{kg}$
R_e	$6.378166000 \times 10^6\ \mathrm{m}$
S	$249.9\ \mathrm{m}^2$
μ	$3.98600442 \times 10^{14}\ \mathrm{m^3 \cdot s^{-2}}$
ρ_0	$1.2256\ \mathrm{kg \cdot m^{-3}}$
ω	$7.292115 \times 10^{-5}\ \mathrm{rad \cdot s^{-1}}$

4.4.3 Propulsion Model

The propulsion model used is a phase-dependent quantity. During the main engine burn (which consists of three main engines burning simultaneously) the thrust is given as [54]

$$T = 3(KF_{vac}^m - A_e^m p) \tag{66}$$

where K is the throttle setting, F_{vac}^{m} is the vacuum thrust of a single main engine, A_e^m is the area at the nozzle exit for one main engine, and p is the ambient atmospheric pressure. The throttle setting is adjusted so that the maximum sensed acceleration does not exceed 3g0 [54]. During the OMS burn, the propulsive force is given as

$$T = 2F_{vac}^{0} \tag{67}$$

where F_{vac}^{0} is the propulsive force from one OMS engine and it is assumed that the total thrust is produced by two OMS engines operating at full throttle $(K = 1)$.

4.4.4 Phases of Flight and Event Sequence

The following two missions are considered in this study. These missions are labeled Mission 1 and Mission 2. Mission 1 consists of four ascent phases. Mission 1 starts at a point where the solid rocket motors are jettisoned and terminates when the operational orbit is achieved. Mission 2 consists of eight phases. The first four phases of Mission 2 are the same as those of Mission 1, while the final four phases of Mission 2 consist of de-orbit and atmospheric entry phases. The de-orbit and entry portions of Mission 2 start at the operational orbit and terminate at the TAEM interface condition. In this section, each ascent and entry phase is described. Missions 1 and 2 are then constructed from the appropriate phases. The description of the different phases includes path constraints enforced during a phase and event constraints that are enforced at phase boundaries. Boundary and linkage conditions are discussed at the end of this section.

4.4.4.1 Phase 1: Second Stage to Main Engine Cutoff

The simulation of the flight starts the instant after the solid rocket motors are jettisoned and terminates at MECO when the external tank is jettisoned. The dynamic model for this phase is that given in Eq. (61) and uses the vehicle model given in Ref. 54. It is noted that the parameterization given in Eq. (61) is different from that of Ref. 54 in that Ref. 54 employs Earth-Centered Inertial (ECI) Cartesian coordinates and includes the gravity perturbations J_2, J_3, and J_4, while this study employs Earth-relative spherical coordinates with a spherical gravity model (that is, in this study J_2, J_3, and J_4 are ignored). Furthermore, in this study, altitude is calculated assuming the Earth is perfectly spherical instead of oblate. Because thrust is produced by three main engines, the propulsion model used in this phase is given by Eq. (66). Next, the aerodynamic model follows the trigonometric model described in Ref. 54, while a linear interpolation of the 1962 US Standard Atmosphere [55] is used for the pressure, the density, and the speed of sound. Furthermore, the conditions for MECO are expressed in terms of the state component r, the inertial speed, the inertial flight path angle, and the cosine of the orbital inclination. The constraint on the radius is imposed as a simple bound on r, while the remaining MECO conditions are imposed on the inertial speed, the inertial flight path angle, and the cosine of the orbital inclination.

4.4.4.2 Phase 2: First OMS Burn

The second phase of flight starts the instant after MECO when the external tank is jettisoned. Aerodynamic forces are considered negligible in this phase and, thus, the lift and drag are eliminated from Eq. (61). Both OMS engines are used during the first OMS burn and, thus, thrust is characterized Eq. (67). The equality path constraint of Eq. (64) is enforced during this phase as well.

4.4.4.3 Phase 3: Coasting Towards Final Orbit

Phase 3 is an unpowered exo-atmospheric phase that starts the instant that the first OMS burn is completed. Therefore, the lift, drag, and propulsive forces are zero and are eliminated from Eq. (61) in this phase. Finally, it is noted that this phase is uncontrolled.

4.4.4.4 Phase 4: Second OMS Burn

Phase 4 starts with the OMS engines burning a second time to achieve the final orbit conditions. Once again, because both OMS engines are used, the thrust is modeled using Eq. (67), while the lift and drag are ignored. Finally, the conditions for achieving the operational orbit are set using the same parameters as those used to describe the MECO conditions. Namely, r is a boundary condition, and the inertial speed, inertial flight path angle, and cosine of the orbital inclination are an event constraint. The values for the event constraint are computed from the state component values at the end of this phase.

4.4.4.5 Phase 5: Final Orbit Coasting Period

Phase 5 begins immediately after the desired payload orbit is achieved and the payload has been deployed. The OMS engines are inactive during this phase, thus T is eliminated in this phase. Aerodynamic forces are assumed to be negligible during this phase and are ignored. In addition, control is absent during this phase.

4.4.4.6 Phase 6: Entry OMS Burn

Phase 6 starts with a third OMS burn to de-orbit the vehicle so that it enters the Earth's atmosphere. To simplify the problem slightly, the direction of the burn is opposite the direction of the Earth relative velocity vector. Thus, the control is held constant at the values $\alpha = 180\, deg$, $w_1 = 1$, and $w_2 = 0$ during the entirety of the phase. Because the control is constant in this phase, the path constraint of Eq. (64) is omitted.

4.4.4.7 Phase 7: Coasting to Entry

Phase 7 starts immediately after the entry OMS burn is complete. The shuttle falls towards Earth until the entry conditions are met. During this time, no engines are active and, thus, the thrust is eliminated from Eq. (61). In addition, aerodynamic forces are still considered negligible during the entirety of the phase. The control is also absent during the phase.

4.4.4.8 Phase 8: Entry to Terminal Area Energy Management Condition

Phase 8 starts at atmospheric entry. No thrust is used during this phase, thus T is eliminated from Eq. (61). Instead, the vehicle behaves like a glider, and the vehicle is controlled using aerodynamic forces. The models used for this phase are taken from Ref. 21. In particular, atmospheric entry is considered with and without a constraint on the heating rate. When a heating rate constraint is imposed, the model for the heating rate (in MW$\cdot m^{-2}$) is given as [21]

$$q = q_a q_r \tag{68}$$

where

$$q_a = c_0 + c_1\hat{a} + c_2\hat{a}^2 + c_3\hat{a}^3 \tag{69}$$

$$q_r = 339.8062\sqrt{\rho}(0.0001v)^{3.07} \tag{70}$$

The re-entry conditions and TAEM conditions are similar to those used in Ref. 21 as well. Because the control is present during this phase, the path constraint of Eq. (64) is active. Furthermore, the heating path constraint is applied as

$$0 \leq q \leq q_{max} \tag{71}$$

where q_{max} is a user-defined variable, and q is the aerodynamic heating on the wing leading edge as defined in [21]. In this study, entry is considered both with and without this heating constraint. When no heating constraint is desired $q_{max} = \infty$.

4.4.4.9 Boundary Conditions and Linkage Conditions

The boundary conditions are described in terms of five sets. Namely, the initial, MECO, operational orbit, entry, and TAEM conditions. Note that the MECO and operational orbit conditions are split up into a boundary condition on r, and an event constraint on the inertial speed, inertial flight path angle, and the cosine of the orbital inclination. Table 21 summarizes the boundary conditions used in this study. Furthermore, linkage conditions of the form

$$x^{p+1}(t_j) - x^p(t_j) = 0 \tag{72}$$

where x is the state and 0 is the zero vector, are imposed between phases (where p is the phase number). These ensure that the state is continuous from one phase to another. Two exceptions are when the external tank is dropped at the end of Phase 1 and when the payload is dropped at the end of Phase 4. In the first case, the zero entry in Eq. 72 corresponding to the state component m is replaced with the negative of the mass of the external tank. In the second case, the mass of the payload is unknown at the start of the problem. Therefore, the linkage condition for the mass state component is expressed as a bounded value between 0 and $-\infty$. Finally, in order to prevent any individual phase from occurring in zero time, a minimum elapsed time of two seconds is imposed as

$$\Delta t_{min} \leq t_{j+1} - t_j \leq \infty \tag{73}$$

where, arbitrarily, $\Delta t_{min} = 2\ s$

Table 19. Boundary Conditions.

State Component	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
$r(t)$	r_0	r_1	-	-	r_4	-	-	r_7	r_8
$\theta(t)$	θ_0	-	-	-	-	-	-	-	θ_8
$\phi(t)$	ϕ_0	-	-	-	-	-	-	-	ϕ_8
$v(t)$	v_0	-	-	-	-	-	-	v_7	v_8
$\gamma(t)$	γ_0	-	-	-	-	-	-	-	γ_8
$\psi(t)$	ψ_0	-	-	-	-	-	-	-	-
$m(t)$	m_0	-	-	-	-	-	-	-	-

4.4.5 Optimal Control Problem

This study focuses on maximizing the weight of a payload being sent orbit for two missions using the shuttle as an example launch vehicle. The following constrained nonlinear optimal control problem arises for Mission 1 (where we recall that Mission 1 terminates when the vehicle achieves its operational orbit). Determine the state $x(\cdot)$ and the control $u(\cdot)$ that minimize the cost functional

$$J = -m(t_f) \tag{74}$$

where $m(t_f)$ is the final mass of the shuttle (including its payload), subject to the dynamic constraints of Eq. (61), the boundary conditions of Table 21, and the appropriate path and event constraints. It is noted that minimizing the negative of the shuttle's final mass has the same effect as maximizing the payload mass being sent to orbit.

A slightly different constrained nonlinear optimal control problem arises for Mission 2. Determine the trajectory $(r(t), \theta(t), \phi(t), v(t), \gamma(t), \psi(t))$ and the control $(\alpha(t), \sigma(t))$ that minimize the cost functional

$$J = W\big(m(t_5) - m(t_4)\big) + t_7 - t_8 \tag{75}$$

where W is a scaling factor (for the purposes of this study, $W = 2$), $m(t_5) - m(t_4)$ is the negative of the payload mass and $t_7 - t_8$ is the negative elapsed time during phase 8 (entry), subject to the dynamic constraints of Eq. (61), and the boundary conditions of Table 21, and the appropriate path and event constraints. In the case where an entry heating constraint is imposed, the additional path constraint of Eq. (68) is also enforced. Note that the addition of $t_7 - t_8$ is necessary, because there are many entry trajectories that satisfy the conditions above and minimize a cost functional that is dependent only on the weight of the payload. This addition has the effect of singling out a trajectory which also maximizes the amount of time spent in entry. Furthermore, this yields useful information, as the solution will indicate the earliest possible time the entry maneuver could begin.

4.4.5.1 Numerical Solutions of Mission 1 and Mission 2 Optimal Control Problems

Missions 1 and 2, as described above, were solved using the MATLAB optimal control software $\mathbb{GPOPS} - \mathbb{II}$ [48]. $\mathbb{GPOPS} - \mathbb{II}$ employs variable-order Legendre-Gauss-Radau (LGR) collocation where the optimal control problem is transcribed to a large sparse nonlinear programming problem (NLP). The NLP arising from the LGR transcription was solved using the NLP solver IPOPT [56] where the derivatives required by the NLP solver are obtained using sparse finite-difference approximations using the method developed in Ref. 57. The Mission 1 and Mission 2 planning problems were solved with a mesh refinement accuracy tolerance of 10^{-4}, and an NLP solver tolerance of 10^{-4}. For both missions the initial mesh in each phase consisted of ten evenly spaced mesh intervals and four collocation points per mesh interval were used in each phase. All computations performed in this study were performed using a MacBook Pro with a 2.8 GHz Intel Core i7 processor and 16 GB of RAM.

4.4.5.2 Results and Discussion

Both missions were solved using the following conditions, where appropriate. Initial conditions were adapted from Ref. 54:

$$\begin{gathered} r(t_0) = 6.424613588 \times 10^6\, m,\ V(t_0) = 1384.7\, m\, s^{-1} \\ \theta(t_0) = -120.74\, deg,\ \gamma(t_0) = 26.409\, deg \\ \phi(t_0) =,\ \psi(t_0) = 259\, deg \\ m(t_0) =,\ t_0 = 126.1\, s \end{gathered} \tag{76}$$

Conditions for MECO were also adapted from Ref. 54:

$$\begin{gathered} r(t_1) = 6.483730100 \times 10^6\, m \\ vI(t_1) = 7734.0\, m\, s^{-1} \\ \gamma I(t_1) = 0.65\, deg \\ i(t_1) = 98\, deg \end{gathered} \tag{77}$$

where v_I and γ_I are the inertial speed and inertial flight path angle, respectively. A 98 deg inclined circular operational orbit at an altitude of 203.720 km was chosen and is described as

$$\begin{gathered} r(t_4) = 6.581886000 \times 10^6\, m \\ vI(t_4) = \sqrt{\frac{\mu}{r(t_4)}}\, m\, s^{-1} \\ \gamma I(t_4) = 0\, deg \\ i(t_4) = 98\, deg \end{gathered} \tag{78}$$

Next, the entry conditions were adapted from Ref. 21 and are given as

$$\begin{gathered} r(t_7) = 6.457414000 \times 10^6\, m \\ v(t_7) = 7802.9\, m\, s^{-1} \end{gathered} \tag{79}$$

Finally, the TAEM interface is described in Ref. 21 as

$$\begin{gathered} r(t_8) = 6.402550000 \times 10^6\, m,\ V(t_8) = 762\, m\, s^{-1} \\ \theta(t_8) = 121.5\, deg,\ \gamma(t_8) = -5\, deg \\ \phi(t_8) = 35.3\, deg \end{gathered} \tag{80}$$

where the additional geocentric latitude and longitude requirements were added so that Mission 2 ends approximately 110 km NW of the Vandenburg Air Force Base landing strip.

The results of this study are divided into sections for Mission 1 and Mission 2. Key features of the optimal trajectory are discussed in each section, and computational characteristics are explored. A comparison of trajectories with and without an additional heating constraint during the final entry phase is presented for Mission 2.

4.4.5.3 Key Features of Mission 1

The components of the state and control for Mission 1 are shown in Figures 63 and 64, respectively. The optimal final mass (including payload) obtained was 109784 kg, and the total duration of the ascent is approximately 2548 seconds. MECO occurs at t = 489.9 s, and the two OMS burns last approximately 284 s and 66 s, respectively. Along the optimal path, the largest increase in speed is obtained during Phase 1. In particular, it is observed that Phase 1 consists of an initial rapid climb followed by a slight reduction in altitude while the speed continues to increase. Phases 2 through 4 then consist of two relatively short OMS burns (Phases 2 and 4) with a longer coasting period in Phase 3. The long coast enables the vehicle to ascend to the operational orbit altitude and enables circularization of the orbit in Phase 4 (the second OMS burn).

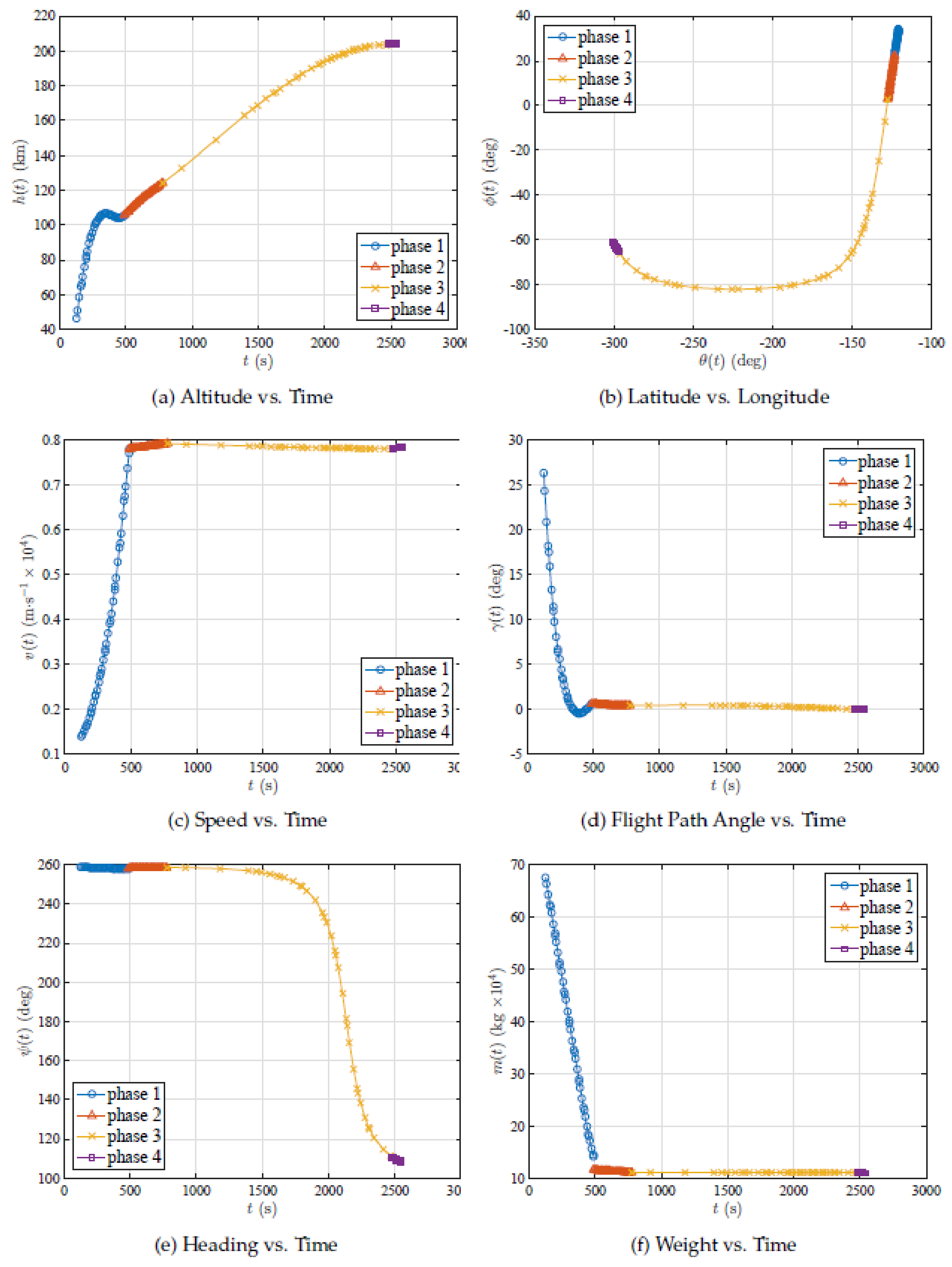
The control indicates the vehicle starts the mission upside down with its nose performing a single, slow "nod" maneuver during Phase 1. The two OMS burns each occur at roughly constant angle of attack and bank angle, indicating the thrust direction relative to the velocity is essentially constant throughout the OMS burn. It is noted that the values for the angle of attack and bank angle were arbitrarily chosen to be zero during Phase 3 where the shuttle is coasting.

The results obtained for Phase 1 are close to those obtained in Ref. 54 from which Phase 1 was adapted. In Ref. 54, the optimal MECO time is approximately 490.0 s with a final mass of 142246 kg. In this study, MECO occurs at 489.9 s with a final shuttle mass of 142347 kg. The slight difference is due to the fact that in this study a spherical Earth model is used while Ref. 54 uses an oblate Earth model.

Table 20 summarizes the performance of $\mathbb{GPOPS} - \mathbb{II}$ when solving Mission 1. In particular, Table 20 shows the total time taken to solve Mission 1 alongside the performance of the hp-adaptive mesh refinement. Note that the mesh relative error tolerance was set to 10^{-4} and the NLP tolerance was set to 10^{-4}. A default initial mesh was used with each phase divided into ten evenly spaced mesh intervals with four collocation points in each interval.

Table 20. Mission 1 Computational Characteristics.

Total Time	67.0 s			
Number of Meshes	2			
Phase	1	2	3	4
Mesh Intervals	10	10	9	10
Collocation Points	40	40	36	40



(a) Altitude vs. Time

(b) Latitude vs. Longitude

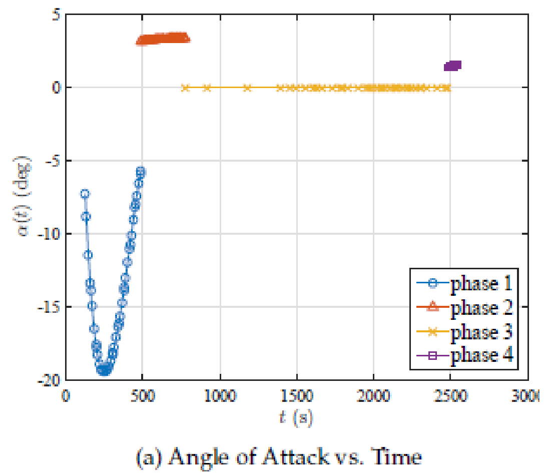
(c) Speed vs. Time

(d) Flight Path Angle vs. Time

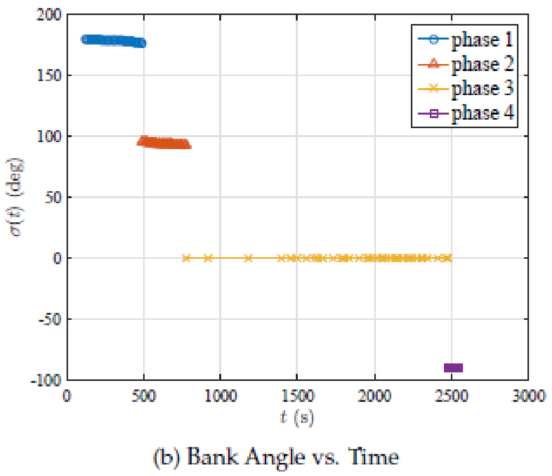
(e) Heading vs. Time

(f) Weight vs. Time

Figure 63. Mission 1: Components of the State.



(a) Angle of Attack vs. Time



(b) Bank Angle vs. Time

Figure 64. Mission 1: Components of the Control.

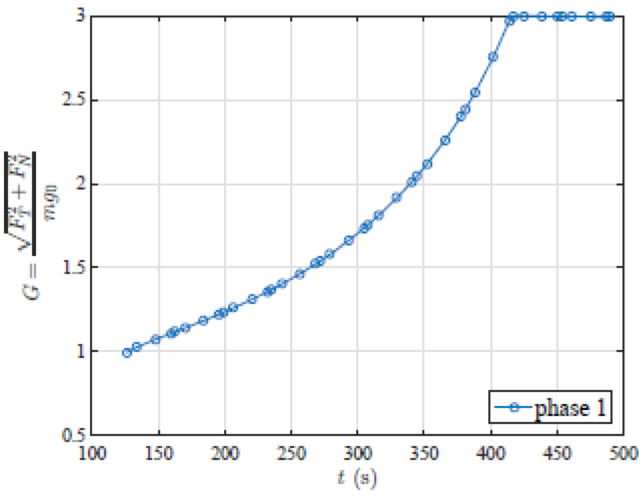


Figure 65. Mission 1: sensed acceleration during phase 1.

4.4.5.4 Key Features of Mission 2

The solution to Mission 2 without a heating rate constraint is shown in Figures 66 and 67, while the solution to Mission 2 with a heating rate constraint is shown in Figures 68 and 69. In the case where no heating rate constraint is imposed, the optimal trajectory yields a payload mass of approximately 11230 kg, while the solution with a maximum allowable heating rate of 0.795 MW$\cdot m^{-2}$ was 11476 kg. While it may seem contradictory that the payload mass is slightly larger when the heating rate constraint is imposed, it is important to remember that the objective functional for Mission 2 is a combination of payload mass and time elapsed since atmospheric entry. As a result, the goal is not solely to maximize the payload mass, and a tradeoff exists between the payload mass and the time elapsed since atmospheric entry. This tradeoff is seen by observing that the reentry phase of Mission 2 lasts (2814 s) when no heating rate constraint is imposed and lasts (2210 s) when the heating rate is constrained.

It is interesting to observe that the ascent portion of Mission 2 is essentially the same as the ascent portion of Mission 1. During the descent of Mission 2, the most interesting features occur during Phase 8 (atmospheric entry). Specifically, it is seen that the entry begins with a high angle of attack followed by a rapid decrease to an angle of attack of roughly 20 deg. The initial angle of attack is large in order to extend time of flight during entry. After the point where α = 20 deg, the angle of attack remains essentially constant for the remainder of the entry, experiencing only minor fluctuations. Next, when no heating rate constraint is imposed, the vehicle subtends large oscillations in altitude and flight path angle. On the other hand, when a heating constraint is imposed, the oscillations during entry are reduced significantly.

Although Mission 2 is a different problem from the entry problem given in Ref. 21, it is seen that the results for Phase 8 of Mission 2 are similar to those obtained in Ref. 21. Although the objective in Ref. 21 is to maximize the cross range, the solution in Ref. 21 shows a similar oscillatory behavior in altitude and flight path angle as the solution obtained in this study when no heating rate constraint is imposed. Finally, similar to the solution found in Ref. 21, the oscillations in the altitude and the flight path angle diminish when the heating rate is constrained.

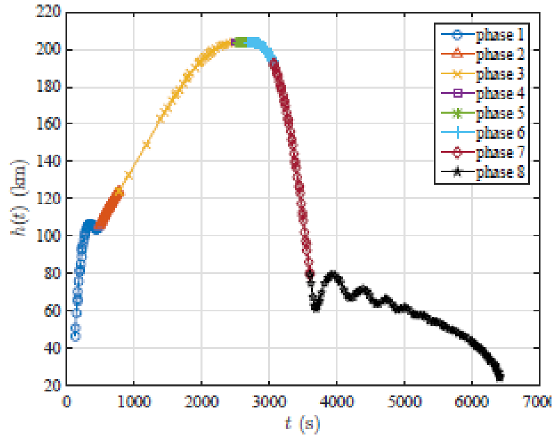
Tables 23 and 24 summarize the total time taken to solve the problem and the mesh characteristics of the solution. Note that the mesh relative error tolerance was set to 10^{-4} and the NLP tolerance was set to 10^{-4} for both cases. A default initial mesh was used with each phase divided into ten evenly spaced mesh intervals with four collocation points in each interval. Notice in Phases 1–7 (where the solution behavior is relatively smooth) the initial mesh intervals and collocation points remained relatively unchanged or were reduced. Phase 8 had a significant increase in the number of mesh intervals and number of collocation points, but this is understandable since the solution is highly oscillatory in this phase.

Table 21. Mission 2 ($q_{max} = \infty$) Computational Characteristics.

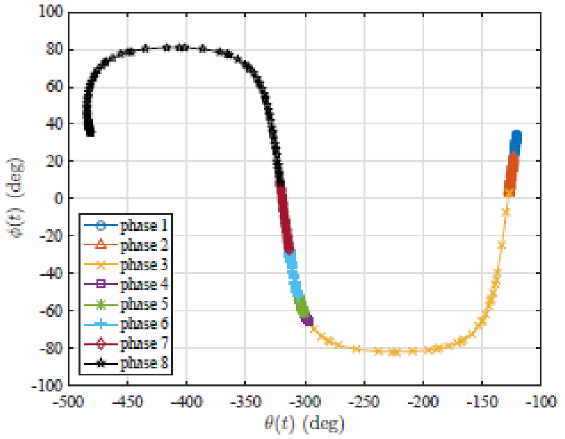
Total Time	512 s							
Number of Meshes	4							
Phase	1	2	3	4	5	6	7	8
Mesh Intervals	10	10	8	10	10	10	10	25
Collocation Points	40	40	33	40	40	40	40	100

Table 22. Mission 2 ($q_{max} = 0.795\ \mathrm{MW\ m^{-2} s^{-1}}$) Computational Characteristics.

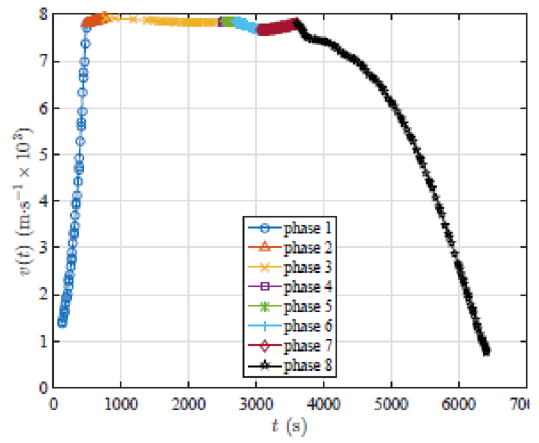
Total Time	1288 s							
Number of Meshes	4							
Phase	1	2	3	4	5	6	7	8
Mesh Intervals	10	10	9	10	11	10	10	25
Collocation Points	40	40	36	40	44	40	40	100



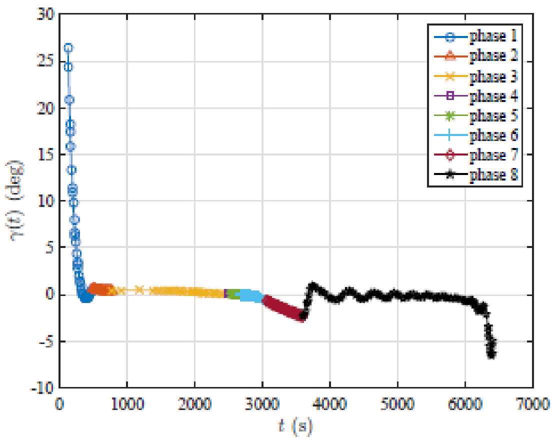
(a) Altitude vs. Time



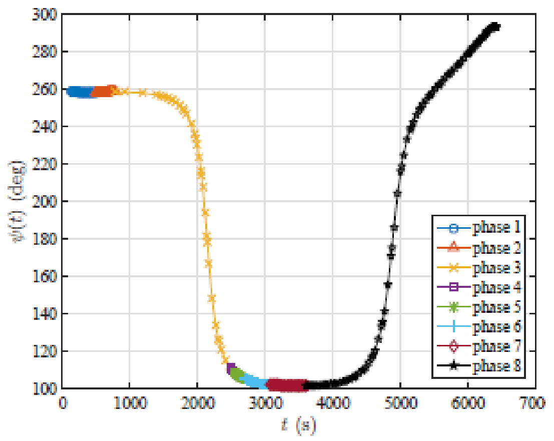
(b) Latitude vs. Longitude



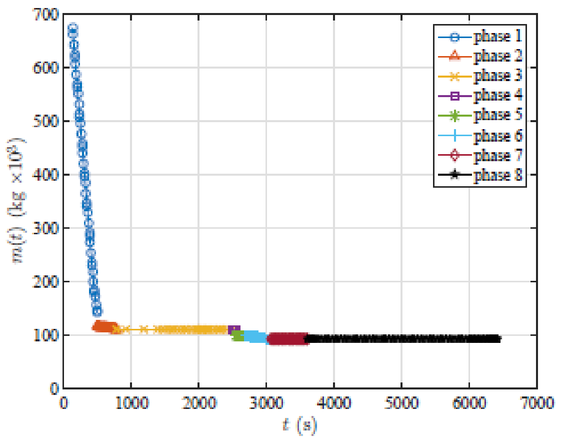
(c) Speed vs. Time



(d) Flight Path Angle vs. Time

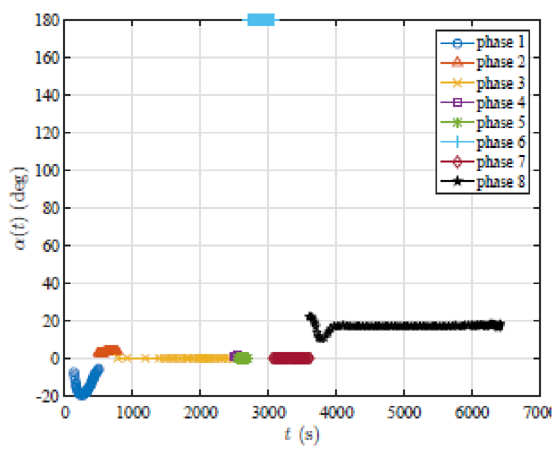


(e) Heading vs. Time



(f) Weight vs. Time

Figure 66.Mission 2 (No Heating Constraint): Components of the State.


(a) Angle of Attack vs. Time

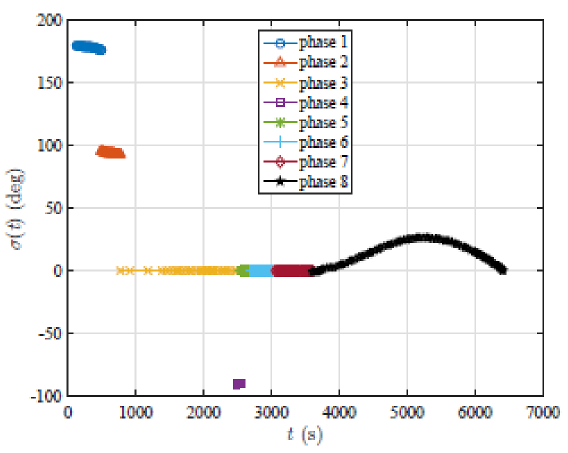
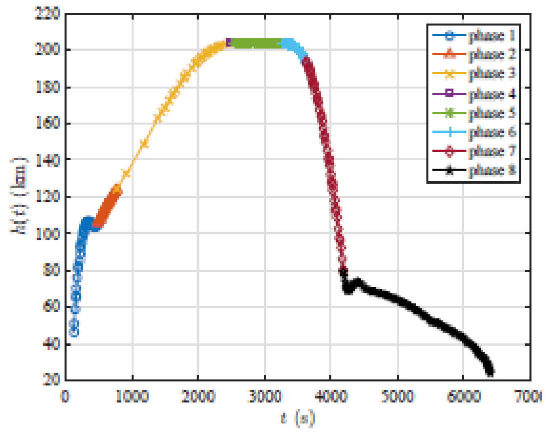
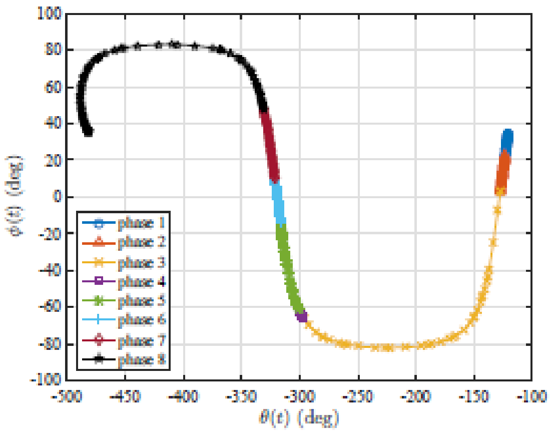

(b) Bank Angle vs. Time

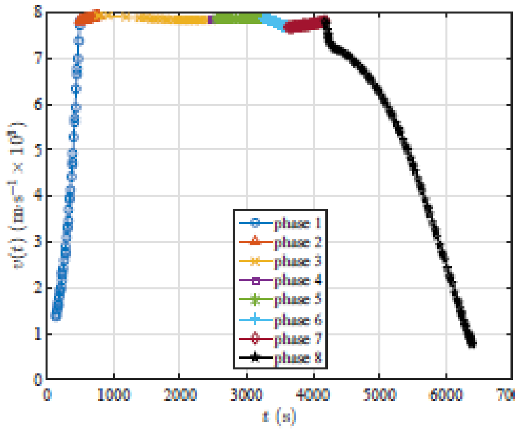
Figure 67. Mission 2 (No Heating Constraint): Components of the Control.



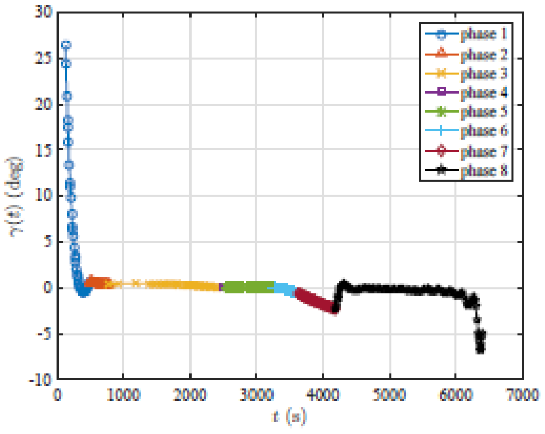
(a) Altitude vs. Time



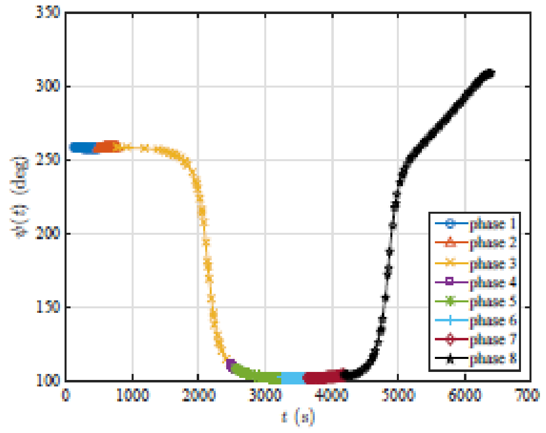
(b) Latitude vs. Longitude



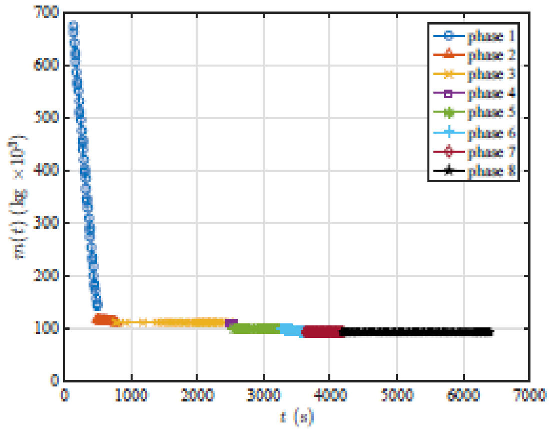
(c) Speed vs. Time



(d) Flight Path Angle vs. Time



(e) Heading vs. Time



(f) Weight vs. Time

Figure 68. Mission 2 (Heating Constraint Active): Components of the State.

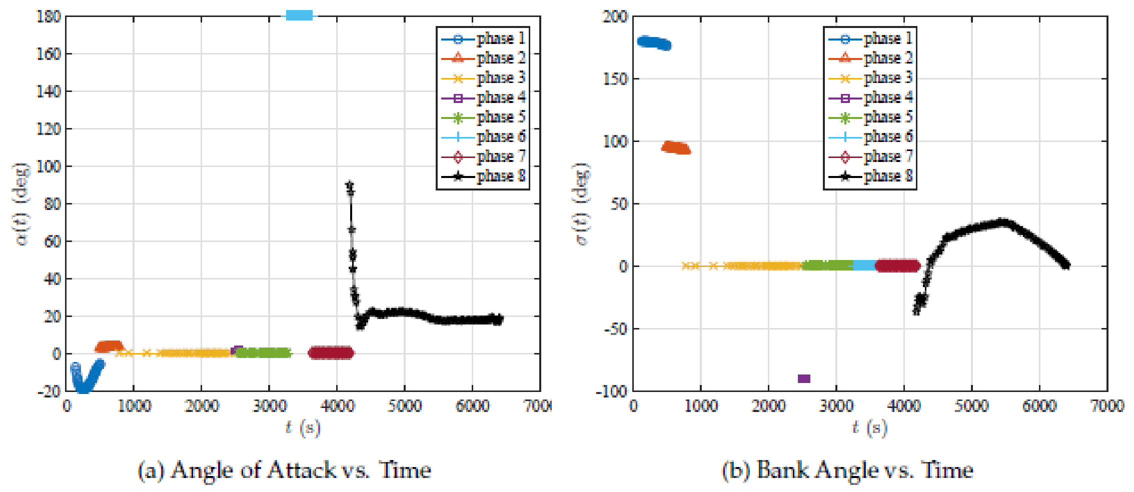


Figure 69. Mission 2 (Heating Constraint Active): Components of the Control.

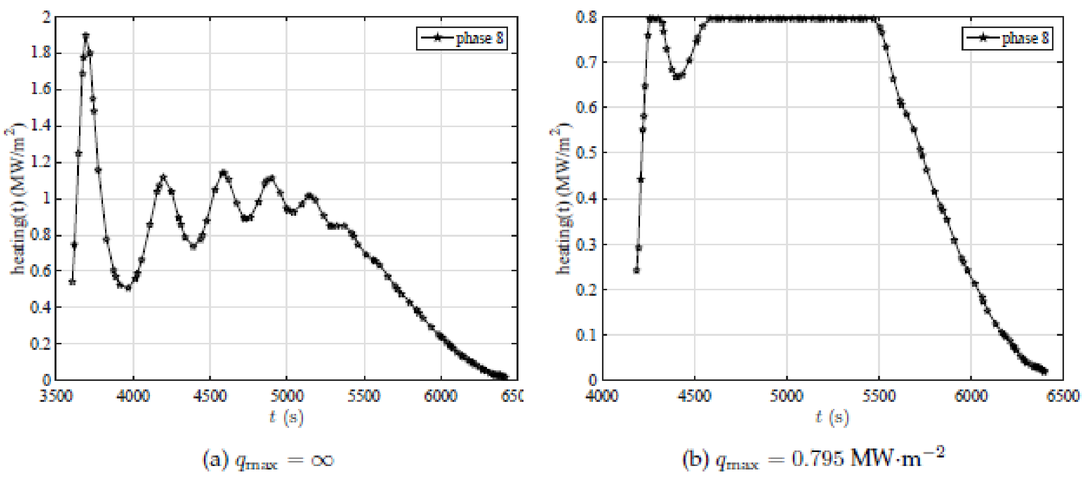


Figure 70.Heating vs. Time.

4.4.6 Conclusions

A combined ascent-entry optimal mission planning has been considered. The mission planning problem was posed as a multiple-phase optimal control problem consisting of powered ascent phases, a powered de-orbit phase, and unpowered entry phases. The goal was to maximize a combination of the payload to orbit and the time elapsed since atmospheric entry after reaching orbit. The multiple-phase optimal control problem was solved using a Legendre-Gauss-Radau collocation method. The results provide insight into the structure of solutions for a mission that consists of a combination of ascent and entry phases and demonstrate the computational efficiency and accuracy of Legendre-Gauss-Radau collocation.

APPENDIX

4.4.7 Limits on Variables

Table 23. Variable Lower and Upper Bounds (Phase 1).

Quantity	Lower Limit	Upper Limit
t_0 (s)	126.1	126.1
t_1 (s)	126.1	1000
$r(t)$ (m$\times 10^6$)	6.378166	6.500086
$\theta(t)$ (deg)	-480.74	-120.74
$\phi(t)$ (deg)	-89	34.1
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$ (kg)	118614	675703
$\alpha(t)$ (deg)	-30	0
$w_1(t)$	-1.3	1.3
$w_2(t)$	-1.3	1.3

Table 24. Variable Lower and Upper Bounds (Phase 2).

Quantity	Lower Limit	Upper Limit
t_2 (s)	126.1	2000
$r(t)$ (m$\times 10^6$)	6.483730	6.581986
$\theta(t)$ (deg)	-480.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$ (kg)	92079	181437
$\alpha(t)$ (deg)	1	90
$w_1(t)$	-1.3	1.3
$w_2(t)$	-1.3	1.3

Table 25. Variable Lower and Upper Bounds (Phase 3).

Quantity	Lower Limit	Upper Limit
t_3 (s)	126.1	4000
$r(t)$ (m$\times 10^6$)	6.483730	6.581986
$\theta(t)$ (deg)	-480.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$ (kg)	92079	181437

Table 26. Variable Lower and Upper Bounds (Phase 4).

Quantity	Lower Limit	Upper Limit
t_4 (s)	126.1	5000
$r(t)$ (m$\times 10^6$)	6.483730	6.581986
$\theta(t)$ (deg)	-480.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$ (kg)	92079	181437
$\alpha(t)$ (deg)	1	90
$w_1(t)$	-1.3	1.3
$w_2(t)$	-1.3	1.3

Table 27. Variable Lower and Upper Bounds (Phase 5).

Quantity	Lower Limit	Upper Limit
t_5 (s)	2000	10000
$r(t)$ (m$\times 10^6$)	6.483730	6.581986
$\theta(t)$ (deg)	-840.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$, (kg)	92079	136078

Table 28. Variable Lower and Upper Bounds (Phase 6).

Quantity	Lower Limit	Upper Limit
t_6 (s)	2000	11000
$r(t)$ (m$\times 10^6$)	6.378166	6.581986
$\theta(t)$ (deg)	-840.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$ (kg)	92079	136078

Table 29. Variable Lower and Upper Bounds (Phase 7).

Quantity	Lower Limit	Upper Limit
t_7 (s)	2000	16000
$r(t)$ (m$\times 10^6$)	6.378166	6.581986
$\theta(t)$ (deg)	-840.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	609.6	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	0	360
$m(t)$ (kg)	92079	136078

Table 30. Variable Lower and Upper Bounds (Phase 8).

Quantity	Lower Limit	Upper Limit
t_8 (s)	2000	20000
$r(t)$ (m$\times 10^6$)	6.378166	6.457414
$\theta(t)$ (deg)	-840.74	-120.74
$\phi(t)$ (deg)	-89	89
$v(t)$ (m$\cdot$s^{-1})	304.8	9144
$\gamma(t)$ (deg)	-89	89
$\psi(t)$ (deg)	-720	720
$m(t)$ (kg)	92079	113398
$\alpha(t)$ (deg)	8	90
$w_1(t)$	-1.3	1.3
$w_2(t)$	-1.3	1.3

4.4.8 Initial Guess

Table 33 contains the values used to generate the initial guess. Entries with "-" indicate there is no control present in either adjacent phase.

Table 31. Initial Guess Values.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t (s)	126.1	490	750	2250	2500	5000	6000	8000	11000
$r(t)$ (m$\times 10^6$)	6.424614	6.483730	6.516449	6.549167	6.581886	6.581886	6.581886	6.457414	6.402550
$\theta(t)$ (deg)	-120.74	-120.74	-120.74	-300.74	-300.74	-300.74	-300.74	-300.74	-481.50
$\phi(t)$ (deg)	34.1	14.1	-5.9	-5.9	14.1	34.1	54.1	75.3	35.3
$v(t)$ (m$\cdot$s^{-1})	1385	7734	7734	7734	7782	7782	7782	7803	762
$\gamma(t)$ (deg)	26.409	0.65	0.65	0.325	0	0	-3	-3	-5
$\psi(t)$ (deg)	259	259	259	159	159	159	159	159	259
$m(t)$ (kg)	675703	141294	110223	110223	108862	96615	92079	92079	92079
$\alpha(t)$ (deg)	0	0	0	0	0	-	-	45	15
$w_1(t)$	-1	-1	-1	1	1	-	-	-1	1
$w_2(t)$	0	0	0	0	0	-	-	0	0

4.5 Advances in Optimization Algorithms and Convergence Theory

An active set algorithm PASA was developed for solving general nonlinear optimization problems with polyhedral constraints. Phase 1 of the algorithm is the gradient projection method, while Phase 2 is any algorithm for solving a linearly constrained optimization problem. Rules were developed for branching between the two phases. Global convergence to a stationary point was established, while asymptotically PASA performs only Phase 2 when either a nondegeneracy assumption holds or the active constraints are linearly independent and a strong second-order sufficient optimality condition holds.

The target optimization problem has the form

$$\min\{f(x): x \in \Omega\},\ where\ \Omega = \{x \in \mathbb{R}^n: Ax \le b\} \tag{81}$$

Here f is a real-valued, continuously differentiable function, $A \in \mathbb{R}^{m\times n}$,$b \in \mathbb{R}^m$, and the polyhedron Ω are assumed to be nonempty. PASA has two phases: Phase one is the gradient projection algorithm, while phase two is any algorithm for solving a linearly constrained optimization problem over a face of the polyhedron. The gradient projection algorithm of phase one is robust in the sense that it converges to a stationary point under mild assumptions, but the convergence rate is often linear at best. When optimizing over a face of the polyhedron in Phase 2, the convergence is accelerated through the use of a superlinearly convergent algorithm based on conjugate gradients, a quasi-Newton update, or a Newton iteration. It is shown that the asymptotic convergence rate of PASA coincides with the convergence rate of the scheme used to solve the linearly constrained problem of Phase 2.

The gradient projection algorithm (GPA) used in PASA is shown in Figure 71. It takes a step along the negative gradient, computes the projection

$$P_\Omega(x) = arg\min\{\|x - y\| : y \in \Omega\} \tag{82}$$

onto the polyhedron, and then performs a line search along the line segment connecting the projection point to the iterate x_k to obtain the next iterate x_{k+1}.

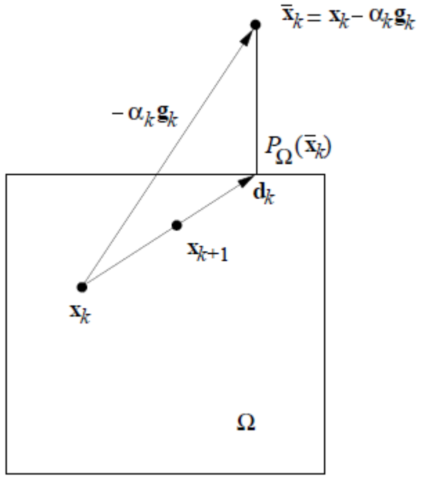

$\bar{\mathbf{x}}_k = \mathbf{x}_k - \alpha_k \mathbf{g}_k$
$-\alpha_k \mathbf{g}_k$
$P_\Omega(\bar{\mathbf{x}}_k)$
$\mathbf{d}_k$
$\mathbf{x}_{k+1}$
$\mathbf{x}_k$
Ω

Figure 71. Gradient projection algorithm.

The linearly constrained optimizer (LCO) in Phase 2 is any algorithm that satisfies the following three conditions:

F1. $x_k \in \Omega$ and $f(x_{k+1}) \leq f(x_k)$ for each k.

F2. $A(x_k) \subset A(x_{k+1})$ for each k, where $A(x) = \{i : (Ax - b)_i = 0\}$

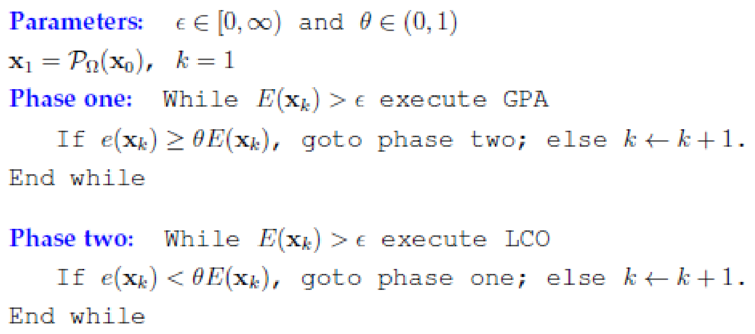
F3. $A(x_{j+1}) = A(x_j)$ for $j \geq k$, then $\lim_{j\to\infty} \inf e(x_j) = 0$

Here, e measures the distance to a stationary point on a face of the polyhedron corresponding to the active constraints; e vanishes only at a stationary point on the current face of the polyhedron. Besides the local error measure e, the algorithm also employs the global stationarity measure E given by

$$E(x) = \left\| P_{\Omega}\big(x - g(x)\big) \right\|$$

E vanishes only at a stationary point for the optimization problem.

A simplified version of PASA appears in Figure 72. Assuming $\theta = 1$, the idea of the algorithm is to use Phase 1 (gradient projection algorithm) until the global error E is smaller than the local error e. Then switch to Phase 2 (linear constrained optimizer), which continues until the local error e is smaller than the global error E.



```
Parameters:  $\epsilon \in [0,\infty)$ and $\theta \in (0,1)$
$\mathbf{x}_1 = \mathcal{P}_{\Omega}(\mathbf{x}_0)$,  $k = 1$
Phase one:  While $E(\mathbf{x}_k) > \epsilon$ execute GPA
    If $e(\mathbf{x}_k) \geq \theta E(\mathbf{x}_k)$, goto phase two; else $k \leftarrow k+1$.
End while

Phase two:  While $E(\mathbf{x}_k) > \epsilon$ execute LCO
    If $e(\mathbf{x}_k) < \theta E(\mathbf{x}_k)$, goto phase one; else $k \leftarrow k+1$.
End while
```

Figure 72. PASA Algorithm.

The paper [58] establishes strong convergence properties for PASA. For example, it is shown that PASA with $\in = 0$ either terminates in a finite number of iterations at a stationary point, or the global error tends to zero:

$$\lim_{k \to \infty} \inf \mathrm{E}(x_k) = 0$$

Moreover, at nondegenerate stationary points where the multipliers for active constraints do not vanish, only Phase 2 (the superlinearly convergent algorithm) is executed asymptotically. Essentially, this implies that the algorithm converges as fast as the algorithm used to solve the linearly constrained optimization problem.

Another result, similar to the result for nondegenerate stationary points, is the following: Theorem: If PASA with $\in = 0$ generates an infinite sequence of iterates converging to a local minimizer x^* where the active constraint gradients are linearly independent and the strong second-order sufficient optimality condition holds, and if f is twice continuously differentiable near x^*, then within a finite number of iterations, only Phase 2 (LCO) is executed.

To assess the behavior of the PASA in practice, a specific implementation was developed in which the projection algorithm PPROJ of [59] was used to implement the gradient projection method, the line search was implemented using the non-monotone techniques in [60], and the conjugate gradient algorithm CG_DESCENT [61] was used for the linear constrained optimizer. The two phases of PASA, unconstrained optimization and gradient projection, are tightly coupled since the projection used in the gradient projection algorithm became a preconditioner in the conjugate gradient algorithm to keep the iterates on the current face of the polyhedron.

A test set was solved that consisted of all the test problems in the CUTE library [62] that possess a nonlinear objective and both linear equality and inequality constraints. The problem dimensions extended up to 7564. For comparison, we solved this same test set using the well- established IPOPT algorithm [63] (Version 3.11) in the COIN-OR library. Since PASA currently does not utilize Hessian information, IPOPT was run in two different modes. In the quasi-Newton mode, IPOPT does not utilize Hessian information; instead, gradients are used to build a Hessian approximation. In its default mode, IPOPT evaluates the Hessian of the objective in each iteration, and then solves a symmetric linear system using the direct factorization method provided in Harwell subroutine MA57.

Figure 73 shows the performance profile for PASA in comparison with IPOPT. In a performance profile, the vertical axis gives the fraction P of problems for which any given method is within a factor τ of the best time. The top curve is the method that solved the most problems in a time that was within a factor τ of the best time. The percentage of the test problems for which a method is fastest is given on the left axis of the plot. The right side of the plot gives the percentage of the test problems that were successfully solved by each of the methods. In essence, the right side is a measure of an algorithm's robustness.

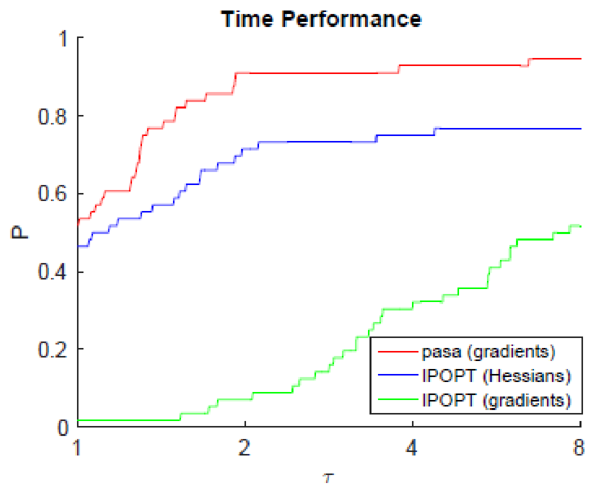


Figure 73. Running time performance for the CUTE test set.

As seen in Figure 73, PASA is much faster than IPOPT in either mode. The default mode for IPOPT based on Hessian information performed much better than the gradient-based quasi-Newton approach for this test set. In this test set, IPOPT with Hessian information failed to solve four out of the 56 problems for a failure rate of 7% (either an error message was generated stating that "restoration failed" or the code thought that the problem had been solved, but the solution was very inaccurate with 0 or 1 correct digits). In the quasi-Newton mode, there were five cases where restoration failed and two additional cases where the solution was very inaccurate. In contrast, PASA was able to solve all the problems in the test set to the specified six-digit error tolerance. A MATLAB interface has been developed that enables the use of PASA in conjunction with the optimal control software GPOPS-II.

Besides the development of an optimization algorithm, we have developed a theoretical convergence analysis for the hp-adaptive algorithm implemented in GPOP-II. The convergence analysis complements our practical experience with the algorithm by providing a strong theoretical foundation.

We assume that the control problem is written in the form

$$\begin{aligned} &\text{Minimize } C\big(x(1)\big) \\ &\text{Subject to } \dot{x}(t) = f\big(x(t), u(t)\big), u(t) \in \mathcal{U}, t \in \Omega, \\ &\qquad x(-1) = x_0, (x, u) \in C^1(\Omega; \mathbb{R}^n) \times C^0(\Omega; \mathbb{R}^m) \end{aligned} \tag{83}$$

where $\Omega = [-1, 1]$, the control constraint set $\mathcal{U} \subset \mathbb{R}^m$ is closed and convex with nonempty interior, the state $x(t) \in \mathbb{R}^n$, $\dot{x}$ denotes the derivative of x with respect to t, x_0 is the initial condition which we assume is given, $f: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, $C: \mathbb{R}^n \to \mathbb{R}, C^l(\Omega; \mathbb{R}^n)$

denotes the space of l times continuously differentiable functions mapping Ω to $\mathbb{R}^n$. It is assumed that f and C are at least continuous.

The first paper [64] analyzes collocation schemes of the following form: Let P_N denote the space of polynomials of degree at most N, let P_N denote the n-fold Cartesian product $P_N \times \cdots \times P_N$, and let $\tau_i, 1 \le i \le N$, denote N Gauss collocation points on the interval $[-1, 1]$. We analyzed the discretization of (81) given by

$$\begin{aligned} &\text{Minimize } C\big(x(1)\big) \\ &\text{Subject to } \dot{x}(\tau_i) = f(x(\tau_i), u_i), u_i \in \mathcal{U}, 1 \le i \le N, \\ &\qquad x(-1) = x_0, x \in P_N^n \end{aligned} \tag{84}$$

The first-order optimality conditions (Pontryagin minimum principle) for the continuous control problem (83) are equivalent to the existence of λ^* such that

$$\begin{gathered} \dot{\lambda}^*(t) = -\nabla_x H\big(x^*(t), u^*(t), \lambda^*(t)\big) \\ \lambda^*(1) = \nabla \mathrm{C}\big(x^*(1)\big) \\ N_u\big(u^*(t)\big) \ni -\nabla_u H\big(x^*(t), u^*(t), \lambda^*(t)\big) \end{gathered}$$

for all $t \in \Omega$, where H is the Hamiltonian defined by $H(x, u, \lambda) = \lambda^T f(x, u)$, ∇ denotes gradient, and N_u is the normal cone. It is shown in [64] that the first-order optimality conditions (Karush-Kuhn-Tucker conditions) for the discrete problem (84) are equivalent to the existence of $\lambda \in P_N^n$ such that

$$\begin{gathered} \dot{\lambda}(\tau_i) = -\nabla_x H\big(x(\tau_i), u_i \lambda(\tau_i)\big), \qquad 1 \le i \le N \\ \lambda(1) = \nabla \mathrm{C}\big(x(1)\big) \\ N_u(u_i) \ni -\nabla_u H\big(x(\tau_i), u_i \lambda(\tau_i)\big), \qquad 1 \le i \le N \end{gathered}$$

By comparing the discrete and continuous first-order optimality conditions, the following exponential convergence result is developed in [64].

Theorem. Suppose (x^*, u^*) is a local minimizer for the continuous problem (81) with $(x^*, \lambda^*) \in \mathcal{H}^\eta(\Omega; \mathbb{R}^n)$ for some $\eta \ge 2$. Then for N sufficiently large, the discrete problem (84) has a local minimizer $x^N \in P_N^n$ and $u \in \mathbb{R}^{mN}$, and an associated multiplier $\lambda^N \in P_N^n$ satisfying the KKT conditions; moreover, there exists a constant c independent of N and η such that

$$\max\{\|X^N - X^*\|_\infty, \|U^N - U^*\|_\infty, \|\Lambda^N - \Lambda^*\|_\infty\}$$
$$\leq \left(\frac{c}{N}\right)^{p-3/2} \left(\|x^*\|_{\mathcal{H}^p(\Omega;\mathbb{R}^n)} + \|\lambda^*\|_{\mathcal{H}^p(\Omega;\mathbb{R}^n)}\right), \qquad p := \min\{\eta, N+1\}$$

where the N and $*$ superscripts in the error bound refer to the discrete and continuous variables respectively evaluated at the Gauss collocation points, and $\mathcal{H}^P$ denotes a Sobolev space of order p.

In a numerical example for which the exact solution of (83) is known, we plot the log of the error versus the polynomial degree N in a log scale and obtain nearly straight lines, as shown in Figure 74, in agreement with the theorem.

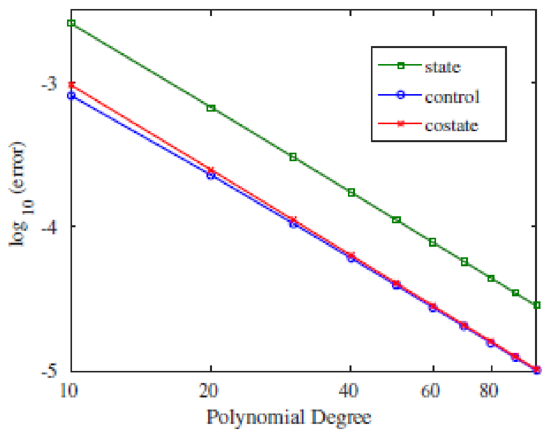


Figure 74. Logarithm of sup-norm error in state, control, and costate versus polynomial degree.

The second convergence theory paper [65] analyzes the actual hp-collocation scheme used in GPOPS-II. In this approach, explained earlier in this report, the time interval, which is now assumed to be $\Omega_0 = [0,1]$, is partitioned into K mesh intervals. For simplicity, we consider a uniform mesh of width $h = 1/K$. The continuous control problem is then reformulated as a series of K control problems, each on the interval $\Omega = [-1,1]$:

$$\left.\begin{array}{c} \textit{minimize } C\left(x_K(1)\right) \\ \textit{subject to } \dot{x}_k(\tau) = hf\left(x_k(\tau),\ u_k(\tau)\right), u_k(\tau) \in \mathcal{U}, \tau \in \Omega, \\ x_k(-1) = x_{k-1}(1),\ 1 \leq k \leq K, \\ (x_k, u_k) \in C^1(\Omega) \times C^0(\Omega) \end{array}\right\} \tag{85}$$

Let P_N denote the space of polynomials of degree at most N defined on the interval Ω, and let P_N^n denote the n-fold Cartesian product $P_N \times \cdots \times P_N$. The hp-collocation scheme corresponding to (85) is as follows:

$$\left.\begin{array}{c} minimize\ C\big(x_K(1)\big) \\ subject\ to\ \dot{x}_k(\tau_i) = hf(x_k(\tau_i),\ u_{ki}),\ 1 \le i \le N, u_{ki} \in \mathcal{U} \\ x_k(-1) = x_{k-1}(1),\ 1 \le k \le K, x_k \in \mathcal{P}_N^n \end{array}\right\} \quad (86)$$

To achieve continuity of the state variable continuous across the mesh points, it is more convenient to employ the Radau collocation points rather than the Gauss points. However, the Radau points are more difficult to analyze than the Gauss points due to the lack of symmetry. The first-order optimality conditions for (86) reduce to the following: There exists $\lambda_k \in P_{N-1}^n, 1 \le k \le K$, such that

$$\begin{gathered} \dot{\lambda}_k(\tau_i) = -h\nabla x^H\big(x_k(\tau_i), u_{ki}, \lambda_k(\tau_i)\big), 1 \le i < N, \\ \dot{\lambda}_k(1) = -h\nabla x^H\big(x_k(1), u_{kN}, \lambda_k(1)\big) + \big(\lambda_k(1) - \lambda_{k+1}(-1)\big)/\omega_N \\ where\ \lambda_{K+1}(-1) := \nabla C\big(x_K(1)\big) \\ N_{\mathcal{U}}(u_{ki}) \ni -\nabla u^H\big(x_k(\tau_i), u_{ki}, \lambda_k(\tau_i)\big), 1 \le i < N \end{gathered}$$

The convergence result established for the hp-collocation scheme was the following: Theorem. If (x^*, u^*) is a local minimizer for the continuous problem (81) with x^* and $\lambda^* \in \mathcal{PH}^\eta(\Omega_0)$ for some $\eta \ge 2$, then for N sufficiently large or for h sufficiently small with $N \ge 2$, the discrete problem has a local minimizer and associated multiplier satisfying the KKT conditions, and we have

$$\begin{gathered} \max\{\|X^N - X^*\|_\infty, \|U^N - U^*\|_\infty, \|\Lambda^N - \Lambda^*\|_\infty\} \\ \le h^{p-1}\left(\frac{c}{N}\right)^{p-1} |x^*|\mathcal{PH}^p(\Omega_0) + h^{q-1}\left(\frac{c}{N}\right)^{q-1.5} |\lambda^*|\mathcal{PH}^q(\Omega_0) \end{gathered}$$

where $p = min(\eta, N + 1)$, $q = min(\eta, N)$, and c is independent of h, N, and η.

Thus the convergence speed is again exponentially fast relative to the degree of the polynomials used on each interval, while the error also decreases at polynomial speed relative to width h of the mesh intervals.

5 REFERENCES

[1] Bliss, G. A., Lectures on the Calculus of Variations, University of Chicago Press, Chicago, IL, 1946.

[2] Athans, M. A. and Falb, P. L., Optimal Control: An Introduction to the Theory and Its Applications, Dover Publications, Mineola, New York, 2006.

[3] Kirk, D. E., Optimal Control Theory: An Introduction, Dover Publications, Mineola, New York, 2004.

[4] Bryson, A. E. and Ho, Y.-C., Applied Optimal Control, Hemisphere Publishing, New York, 1975.

[5] Pontryagin, L. S., Mathematical Theory of Optimal Processes, John Wiley & Sons, New York, 1962.

[6] Bryson, A. E., Desai, M. N., and Hoffman, W. C., “Energy-State Approximation in Performance Optimization of Supersonic Aircraft,” AIAA Journal of Aircraft, Vol. 6, No. 6, November–December 1969, pp. 481–488.

[7] Bazaraa, M. S., Sherali, H. D., and Shetty, C. M., Nonlinear Programming: Theory and Algorithms, Wiley-Interscience, 3rd ed., 2006.

[8] Bertsekas, D., Nonlinear Programming, Athena Scientific Publishers, Belmont, Massachusetts, 2004.

[9] Boyd, S. and Vandenberghe, L., Convex Optimization, Cambridge University Press, Cam- bridge, United Kingdom, 2004.

[10] Ascher, U. M., Mattheij, R. M., and Russell, R. D., Numerical Solution of Boundary-Value Problems in Ordinary Differential Equations, SIAM Press, Philadelphia, 1996.

[11] Betts, J. T., “Survey of Numerical Methods for Trajectory Optimization,” Journal of Guidance, Control, and Dynamics, Vol. 21, No. 2, March–April 1998, pp. 193–207.

[12] Rao, A. V., “A Survey of Numerical Methods for Optimal Control,” Advances in the Astronautics Sciences, edited by A. V. Rao, T. A. Lovell, K. A. Chan, and A. L. Cangahuala, Univelt Publishers, San Diego, April 2009.

[13] Limebeer, D. J. N. and Rao, A. V., “Faster, Higher, and Greener: Vehicular Optimal Control,” IEEE Control Systems Magazine, Vol. 35, No. 2, April 2015, pp. 36–56.

[14] Keller, H. B., Numerical Solution of Two Point Boundary Value Problems, SIAM, 1976.

[15] Stoer, J. and Bulirsch, R., Introduction to Numerical Analysis, Springer-Verlag, 2002.

[16] Rao, A. V., Extension of the Computational Singular Perturbation Method to Optimal Control, Ph.D. thesis, Princeton University, 1996.

[17] Rao, A. V. and Mease, K. D., “Dichotomic Basis Approach to solving Hyper-Sensitive Opti- mal Control Problems,” Automatica, Vol. 35, No. 4, April 1999, pp. 633–642.

[18] Rao, A. V. and Mease, K. D., “Eigenvector Approximate Dichotomic Basis Method for Solving Hyper-Sensitive optimal Control Problems,” Optimal Control Applications and Methods, Vol. 21, No. 1, January–February 2000, pp. 1–19.

[19] Rao, A. V., “Application of a Dichotomic Basis Method to Performance Optimization of Supersonic Aircraft,” Journal of Guidance, Control, and Dynamics, Vol. 23, No. 3, May–June 2000, pp. 570–573.

[20] Rao, A. V., “Riccati Dichotomic Basis Method for solving Hyper-Sensitive optimal Control Problems,” Journal of Guidance, Control, and Dynamics, Vol. 26, No. 1, January–February 2003, pp. 185–189.

[21] Betts, J. T., Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, SIAM Press, Philadelphia, 2nd ed., 2009.

[22] Hager, W. W., “Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System,” Numerische Mathematik, Vol. 87, 2000, pp. 247–282.

[23] Dontchev, A. L., Hager, W. W., and Malanowski, K., “Error Bounds for the Euler Approxi- mation and Control Constrained Optimal Control Problem,” Numerical Functional Analysis and Applications, Vol. 21, 2000, pp. 653–682.

[24] Dontchev, A. L., Hager, W. W., and Veliov, V. M., “Second-Order Runge-Kutta Approximations In Constrained Optimal Control,” SIAM Journal on Numerical Analysis, Vol. 38, 2000, pp. 202–226.

[25] Dontchev, A. L., Hager, W. W., and Veliov, V. M., “Uniform Convergence and Mesh independence of Newton’s method for Discretized Variational Problems,” SIAM Journal on Control and Optimization, Vol. 39, 2000, pp. 961–980.

[26] Dontchev, A. L. and Hager, W. W., “The Euler Approximation in State Constrained Optimal Control,” Mathematics of Computation, Vol. 70, 2001, pp. 173–203.

[27] Dontchev, A. L. and Hager, W. W., “A New Approach to Lipschitz Continuity in State Con- strained Optimal Control,” Systems and Control Letters, Vol. 35, 1998, pp. 137–143.

[28] Dontchev, A. L. and Hager, W. W., “Lipschitzian Stability for State Constrained nonlinear Optimal Control,” SIAM Journal on Control and Optimization, Vol. 36, 1998, pp. 696–718.

[29] Gill, P. E., Murray, W., and Saunders, M. A., “SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization,” SIAM Review, Vol. 47, No. 1, January 2002, pp. 99–131.

[30] Gill, P. E., Murray, W., and Saunders, M. A., User’s Guide for SNOPT Version 7: Software for Large Scale Nonlinear Programming, February 2006.

[31] Byrd, R. H., Nocedal, J., and Waltz, R. A., “KNITRO: An Integrated Package for Nonlinear Optimization,” Large Scale Nonlinear Optimization, Springer Verlag, 2006, pp. 35–59.

[32] Benson, D. A., Huntington, G. T., Thorvaldsen, T. P., and Rao, A. V., “Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method,” Journal of Guidance, Control, and Dynamics, Vol. 29, No. 6, November-December 2006, pp. 1435–1440.

[33] Huntington, G. T., Benson, D. A., and Rao, A. V., “Optimal Configuration of Tetrahedral Spacecraft Formations,” The Journal of the Astronautical Sciences, Vol. 55, No. 2, April-June 2007, pp. 141–169.

[34] Huntington, G. T. and Rao, A. V., “Optimal Reconfiguration of Spacecraft Formations Using the Gauss Pseudospectral Method,” Journal of Guidance, Control, and Dynamics, Vol. 31, No. 3, May-June 2008, pp. 689–698.

[35] Rao, A. V., Benson, D. A., Darby, C. L., Francolin, C., Patterson, M. A., Sanders, I., and Huntington, G. T., “Algorithm 902: GPOPS, A Matlab Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method,” ACM Transactions on Mathematical Software, Vol. 37, No. 2, April–June 2010, Article 22, 39 pages.

[36] Garg, D., Patterson, M. A., Darby, C. L., Francolin, C., Huntington, G. T., Hager, W. W., and Rao, A. V., “Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems via a Radau Pseudospectral Method,” Computational Optimization and Applications, Vol. 49, No. 2, June 2011, pp. 335–358. DOI: 10.1007/s10589–00–09291–0.

[37] Garg, D., Patterson, M. A., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., “A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods,” Automatica, Vol. 46, No. 11, November 2010, pp. 1843–1851. DOI: 10.1016/j.automatica.2010.06.048.

[38] Garg, D., Hager, W. W., and Rao, A. V., “Pseudospectral Methods for Solving Infinite- Horizon Optimal Control Problems,” Automatica, Vol. 47, No. 4, April 2011, pp. 829–837. DOI: 10.1016/j.automatica.2011.01.085.

[39] Kameswaran, S. and Biegler, L. T., “Convergence Rates for Direct Transcription of Optimal Control Problems Using Collocation at Radau Points,” Computational Optimization and Applications, Vol. 41, No. 1, 2008, pp. 81–126.

[40] Darby, C. L., Hager, W. W., and Rao, A. V., “An hp–Adaptive Pseudospectral Method for Solving Optimal Control Problems,” Optimal Control Applications and Methods, Vol. 32, No. 4, July–August 2011, pp. 476–502.

[41] Darby, C. L., Hager, W. W., and Rao, A. V., “Direct Trajectory Optimization Using a Variable Low-Order Adaptive Pseudospectral Method,” Journal of Spacecraft and Rockets, Vol. 48, No. 3, May–June 2011, pp. 433–445.

[42] Francolin, C. C., Hager, W. W., and Rao, A. V., “Costate Approximation in Optimal Control Using Integral Gaussian Quadrature Collocation Methods,”

Optimal Control Applications and Methods, Vol. 36, No. 4, July–August 2015, pp. 381–397.

[43] Patterson, M. A., Hager, W. W., and Rao, A. V., “A ph Mesh Refinement Method for Optimal Control,” Optimal Control Applications and Methods, Vol. 36, No. 4, July–August 2015, pp. 398–421.

[44] Liu, F., Hager, W. W., and Rao, A. V., “Adaptive Mesh Refinement for Optimal Control Using Nonsmoothness Detection and Mesh Size Reduction,” Journal of the Franklin Institute, Vol. 352, No. 10, 2015, pp. 4081–4106.

[45] Liu, F., Hager, W. W., and Rao, A. V., “Adaptive Mesh Refinement for Optimal Control Using Nonsmoothness Detection and Mesh Size Reduction,” IEEE Transactions on Control System Technology, Vol. Published Online in Early View, No. 10, June 2017, pp. 4081–4106.

[46] Miller, A. T., Hager, W. W., , and Rao, A. V., “A Preliminary Analysis of Mesh Refinement for Optimal Control Using Discontinuity Detection via Jump Function Approximations,” AIAA Guidance, Navigation, and Control Conference, Kissimmee, Florida, AIAA Paper 2018-0852, 8–12 January 2018.

[47] Abramowitz, M. and Stegun, I., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Dover Publications, New York, 1965.

[48] Patterson, M. A. and Rao, A. V., “GPOPS − II, A MATLAB Software for Solving Multiple- Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming,” ACM Transactions on Mathematical Software, Vol. 41, No. 1, October 2014, pp. 1:1–1:37.

[49] Gelb, A. and Tadmor, E., “Detection of Edges in Spectral Data,” Applied and Computational Harmonic Analysis, Vol. 7, No. 1, July 1999, pp. 101 – 135.

[50] Bary, N., Treatise of Trigonometric Series, Macmillan, New York, 1964.

[51] Zygmund, A., Trigonometric Series, Cambridge University Press, Cambridge, UK, 1959.

[52] Biegler, L. T. and Zavala, V. M., “Large-Scale Nonlinear Programming Using IPOPT: An Integrating Framework for Enterprise-Wide Optimization,” Computers and Chemical Engineering, Vol. 33, No. 3, March 2008, pp. 575–582.

[53] Vinh, N.-X., Busemann, A., and Culp, R. D., Hypersonic and Planetary Entry Flight Mechanics, University of Michigan Press, Ann Arbor, Michigan, 1980.

[54] Bauer, T., Betts, J., Hallman, W., Huffman, W., and Zondervan, K., “Solving the Optimal Control Problem Using a Nonlinear Programming Technique. II - Optimal Shuttle Ascent Trajectories,” Astrodynamics Conference, Guidance, Navigation, and Control and Co-located Conferences, AIAA, 1984.

[55] Sissenwine, N., Dubin, M., and Wexler, H., “The U.S. Standard Atmosphere, 1962,” Journal of Geophysical Research, Vol. 67, No. 9, August 1962.

[56] Biegler, L. T., Ghattas, O., Heinkenschloss, M., and van Bloemen Waanders, B., editors, Large-Scale PDE Constrained Optimization, Lecture Notes in Computational Science and Engineering, Vol. 30, Springer-Verlag, Berlin, 2003.

[57] Patterson, M. A. and Rao, A. V., “Exploiting Sparsity in Direct Collocation Pseudospectral Methods for Solving Continuous-Time Optimal Control Problems,” Journal of Spacecraft and Rockets,, Vol. 49, No. 2, March–April 2012, pp. 364–377.

[58] Hager, W. W. and Zhang, H., “An active set algorithm for nonlinear optimization with polyhedral constraints,” Sci. China Math., Vol. 59, 2016, pp. 1525–1542.

[59] Hager, W. W. and Zhang, H., “Projection onto a Polyhedron that Exploits Sparsity,” SIAM Journal on Optimization, Vol. 29, 2016, pp. 1773–1798.

[60] Hager, W. W. and Zhang, H., “A new active set algorithm for box constrained optimization,” SIAM Journal on Optimization, Vol. 17, 2006, pp. 526–557.

[61] Hager, W. W. and Zhang, H., “Algorithm 851: CG_DESCENT, A conjugate gradient method with guaranteed descent,” ACM Transactions on Mathematical Software, Vol. 32, 2006, pp. 113– 137.

[62] Gould, N. I. M., Orban, D., and Toint, P. L., “CUTEst: a Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization,” Computational Optimization and Applications, Vol. 60, 2015, pp. 545–557.

[63] Wächter, A. and Biegler, L. T., “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,” Mathematical Programming, Vol. 106, 2006, pp. 25–57.

[64] Hager, W. W., Liu, J., Mohapatra, S., Rao, A. V., and Wang, X.-S., “Convergence rate for a Gauss collocation method applied to constrained optimal control,” SIAM Journal on Control and Optimization, 2018, to appear.

[65] Hager, W. W., Hou, H., Mohapatra, S., and Rao, A. V., “Convergence rate for an hp-collocation method applied to constrained optimal control,” 2016, arXiv: 1605.02121.

6 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

A_e	Nozzle Exit Area, m^2
D	Drag, N
F_N	Normal Component of Aerodynamic and Propulsive Forces, N
F_T	Tangential Component of Aerodynamic and Propulsive Forces, N
F_{vac}	Vacuum Thrust, N
g	Acceleration Due to Earth Gravity, $m \cdot s^{-2}$
g_e	Sea Level Acceleration Due to Earth Gravity, $m \cdot s^{-2}$
H	Density Scale Height, m
h	Altitude, m
i	Inclination, rad or deg
I_{sp}	Vacuum Specific Impulse, s
K	Throttle Setting
L	Lift, N
m	Mass, kg
m_{ET}	Empty Mass of the External Tank, kg
r	Magnitude of the Position Vector, m
R_e	Radius of the Earth, m
S	Aerodynamic Reference Area, m^2
T	Thrust, N
V	Earth Relative Speed, $m \cdot s^{-1}$
W	Scaling Factor
α	Angle of Attack, rad or deg
γ	Flight Path Angle, rad or deg
θ	Longitude, rad or deg
μ_e	Earth Gravitational Parameter, $m^3 \cdot s^{-2}$
ρ	Density, $kg \cdot m^{-3}$
ρ_0	Sea Level Atmospheric Density, $kg \cdot m^{-3}$
σ	Bank Angle, rad or deg
φ	Geocentric Latitude, rad or deg
ψ	Heading, rad or deg
ω	Earth Rotation Rate, $rad \cdot s^{-1}$ or deg s^{-1}

LEO	Low Earth Orbit
ME	Main Engine
MECO	Main Engine Cut-Off
NLP	Nonlinear Programming Problem
OMS	Orbital Maneuvering System
TAEM	Terminal Area Energy Management

7 DISTRIBUTION LIST

Electronic copies will be delivered to the following. Primary distribution should be kept to a minimum as most users can obtain copies through DTIC. It may be desirable to distribute only a copy of the SF 298 to parties who may be interested.

AFRL-RW-EG-TR-2018-058

AFRL/RWWN -- Copies: 1

AFRL/RWOC (STINFO Officer) --- Copies: 1

DEFENSE TECHNICAL INFORMATION CENTER ------------------------------------ Copies: 1
8725 John J. Kingman Rd Ste 0944
Ft Belvoir, VA 22060-6218

GOVERNMENT WORK UNIT MANAGER FOR THE R&D CASE FILE---------- Copies: 1