ℂGℙOℙS: A C++ Software for Solving Multiple-Phase Optimal Control Problems Using Adaptive Gaussian Quadrature Collocation and Sparse Nonlinear Programming

YUNUS M. AGAMAWI and ANIL V. RAO, University of Florida

A general-purpose C++ software program called ℂGℙOℙS is described for solving multiple-phase optimal control problems using adaptive direct orthogonal collocation methods. The software employs a Legendre-Gauss-Radau direct orthogonal collocation method to transcribe the continuous optimal control problem into a large sparse nonlinear programming problem (NLP). A class of *hp* mesh refinement methods are implemented that determine the number of mesh intervals and the degree of the approximating polynomial within each mesh interval to achieve a specified accuracy tolerance. The software is interfaced with the open source Newton NLP solver IPOPT. All derivatives required by the NLP solver are computed via central finite differencing, bicomplex-step derivative approximations, hyper-dual derivative approximations, or automatic differentiation. The key components of the software are described in detail, and the utility of the software is demonstrated on five optimal control problems of varying complexity. The software described in this article provides researchers a transitional platform to solve a wide variety of complex constrained optimal control problems.

CCS Concepts: • **Mathematics of computing → Solvers**; **Mathematical software performance**;

Additional Key Words and Phrases: Optimal control, direct orthogonal collocation, Gaussian quadrature, *hp* methods, numerical methods, C++, scientific computation, applied mathematics

ACM Reference format:
Yunus M. Agamawi and Anil V. Rao. 2020. ℂGℙOℙS: A C++ Software for Solving Multiple-Phase Optimal Control Problems Using Adaptive Gaussian Quadrature Collocation and Sparse Nonlinear Programming. *ACM Trans. Math. Softw.* 46, 3, Article 25 (July 2020), 38 pages.
https://doi.org/10.1145/3390463

1 INTRODUCTION

Optimal control problems arise in a wide variety of subjects including virtually all branches of engineering, economics, and medicine. Over the past few decades, the subject of optimal control has transitioned from theory to computations as a result of the increasing complexity of optimal control applications and the inability to solve them analytically. In particular, computational optimal control has become a science in and of itself, resulting in a variety of numerical methods and

This work was supported by the U.S. Office of Naval Research under grant N00014-15-1-2048, and the U.S. National Science Foundation under grants DMS-1522629 and DMS-1924762.
Authors' addresses: Y. M. Agamawi and Anil V. Rao, Aerospace and Mechanical Engineering, University of Florida, Gainesville, FL 32611-6250; emails: yunus.agamawi@gmail.com, anilvrao@ufl.edu.

0098-3500/2020/07-ART25 $15.00
https://doi.org/10.1145/3390463

corresponding software implementations of those methods. To date, most software implementations of optimal control involve direct transcription of a continuous optimal control problem to a nonlinear programming problem (NLP). The resulting NLP from discretizing the continuous optimal control problem may then be solved using well-established techniques. Examples of well-known software for solving optimal control problems include *SOCS* [10], *DIRCOL* [58], *GESOP* [44], *OTIS* [57], *MISER* [28], *PSOPT* [6], *GPOPS* [54], *ICLOCS* [21], *ACADO* [40], and $\mathbb{GPOPS}-\mathbb{II}$ [52].

Over the past few decades, direct collocation methods have become popular in the numerical solution of nonlinear optimal control problems. In a direct collocation method, the state and control are parameterized at an appropriately chosen set of discrete points along the time interval of interest. The continuous optimal control problem is then transcribed to a finite-dimensional NLP. The resulting NLP may then be solved using well-known software such as *SNOPT* [26], *IPOPT* [12], and *KNITRO* [13]. Direct collocation methods were originally developed as *h* methods (e.g., Euler or Runge-Kutta methods) where the time interval of interest is divided into a mesh and the state is approximated using a fixed-degree polynomial in each mesh interval. Convergence in an *h* method is then achieved by increasing the number and placement of the mesh points [11, 43, 62]. More recently, a great deal of research has been done in the class of direct *Gaussian quadrature orthogonal collocation* methods [8, 14, 19, 20, 23–25, 30, 41, 42, 45, 51, 54]. In a Gaussian quadrature collocation method, the state is typically approximated using a Lagrange polynomial where the support points of the Lagrange polynomial are chosen to be points associated with a Gaussian quadrature. Originally, Gaussian quadrature collocation methods were implemented as *p* methods using a single interval. Convergence of the *p* method was then achieved by increasing the degree of the polynomial approximation. For problems whose solutions are smooth and well behaved, a Gaussian quadrature orthogonal collocation method converges at an exponential rate [17, 36–39]. The most well-developed *p* Gaussian quadrature methods are those that employ either Legendre-Gauss (LG) points [8, 54], Legendre-Gauss-Radau (LGR) points [24, 25, 45], or Legendre-Gauss-Lobatto (LGL) points [19].

In this article, a new optimal control software called $\mathbb{CGPOPS}$ is described that employs *hp* direct orthogonal collocation methods. An *hp* method is a hybrid between an *h* and a *p* method in that both the number of mesh intervals and the degree of the approximating polynomial within each mesh interval can be varied to achieve a specified accuracy. As a result, in an *hp* method, it is possible to take advantage of the exponential convergence of a Gaussian quadrature method in regions where the solution is smooth and introduce mesh points only when necessary to deal with potential nonsmoothness or rapidly changing behavior in the solution. Originally, *hp* methods were developed as finite-element methods for solving partial differential equations [5, 33–35]. In the past few years, the problem of developing *hp* methods for solving optimal control problems has been of interest, and Refs. [15, 16, 47, 48, 50] provide examples of the benefits of using an *hp*-adaptive method over either a *p* method or an *h* method. This recent research has shown that convergence using *hp* methods can be achieved with a significantly smaller finite-dimensional approximation than would be required when using either an *h* or a *p* method.

It is noted that previously the software $\mathbb{GPOPS}-\mathbb{II}$ was developed as described in Ref. [52]. Although both the $\mathbb{GPOPS}-\mathbb{II}$ and $\mathbb{CGPOPS}$ software programs implement Gaussian quadrature collocation with *hp* mesh refinement, $\mathbb{CGPOPS}$ offers several advantages both in terms of computational efficiency, portability, and accuracy over $\mathbb{GPOPS}-\mathbb{II}$. First, $\mathbb{GPOPS}-\mathbb{II}$ is a MATLAB software program, whereas $\mathbb{CGPOPS}$ is a C++ software program. Furthermore, because $\mathbb{CGPOPS}$ is implemented in C++, it has the potential for improved computational efficiency and portability over MATLAB software such as $\mathbb{GPOPS}-\mathbb{II}$. Second, although $\mathbb{GPOPS}-\mathbb{II}$ employs both sparse finite-differencing and automatic differentiation using the software *ADiGator* [61], $\mathbb{CGPOPS}$ includes the following four derivative estimation methods:

central finite differencing, bicomplex-step [46], hyper-dual [22], and automatic differentiation [32]. Finite-difference derivative estimation is included because using finite difference makes it possible to compute NLP derivatives for the widest range of problems, whereas bicomplex-step and hyper-dual derivative estimation methods are included because they are significantly more accurate and more computationally efficient than either finite differencing or automatic differentiation (see Ref. [4] for a comprehensive analysis of these different differentiation methods). It is noted, however, that the bicomplex-step and hyper-dual derivative estimation methods are semi-automatic differentiation methods that employ source transformation and operator overloading. As a result, these methods are limited to problems where the functions are included in the overloaded library. Third, although $\mathbb{GPOPS}-\mathbb{II}$ is only capable of identifying the first-order derivative dependencies and over-estimates the dependencies of the second derivatives, $\mathbb{CGPOPS}$ is able to exactly identify both the first- and second-order derivative dependencies of the continuous optimal control problem functions when the derivatives are approximated using the hyper-dual method. The improvement in determining the dependencies at the level of second derivatives further improves computational efficiency over $\mathbb{GPOPS}-\mathbb{II}$. Finally, $\mathbb{CGPOPS}$ implements the Gauss quadrature collocation method in a fundamentally different way from the approached used in $\mathbb{GPOPS}-\mathbb{II}$. Specifically, $\mathbb{CGPOPS}$ includes the ability to divide the time interval into multiple domains. This multiple-domain approach is then used as the basis for a new bang-bang mesh refinement method as described in Ref. [2]. It is found for problems where the optimal control is bang-bang that the mesh refinement method described in Ref. [2] significantly improves accuracy and computational efficiency over previously developed mesh refinement methods as implemented in $\mathbb{GPOPS}-\mathbb{II}$.

The objective of this work is to describe a computationally efficient general-purpose C++ optimal control software that accurately solves a wide variety of constrained continuous optimal control problems. In particular, the software described in this article employs a differential form of the multiple-interval version of the LGR collocation method [23–25, 51]. The LGR collocation method is chosen for use in the software because it provides highly accurate state, control, and costate approximations while maintaining a relatively low-dimensional approximation of the continuous problem. The key components of the software are then described and the software is demonstrated on five examples from the open literature. Each example demonstrates different capabilities of the software. The first example is the hyper-sensitive optimal control problem taken from Ref. [55] and demonstrates the ability of the software to accurately solve a problem whose optimal solution changes rapidly in particular regions of the solution. The second example is the reusable launch vehicle entry problem taken from Ref. [11] and demonstrates the ability of $\mathbb{CGPOPS}$ to compute an accurate solution using a relatively coarse mesh. The third example is the space station attitude control problem taken from Refs. [11, 53] and demonstrates the ability of the software to generate accurate solutions to a problem whose solution is not intuitive. The fourth example is a free-flying robot problem taken from Refs. [11, 56] and shows the ability of the software to handle bang-bang optimal control problems using the novel bang-bang control mesh refinement method included in the software. The fifth example is a launch vehicle ascent problem taken from Refs. [7, 11, 54] that demonstrates the ability of the software to solve a multiple-phase optimal control problem. To validate the results, the solutions obtained using $\mathbb{CGPOPS}$ are compared against the solutions obtained using the software $\mathbb{GPOPS}-\mathbb{II}$ [52].

This article is organized as follows. In Section 2, the general multiple-phase optimal control problem is presented. In Section 3, the LGR collocation method that is used as the basis of $\mathbb{CGPOPS}$ is described. In Section 4, the key components of $\mathbb{CGPOPS}$ are described. In Section 5, the results obtained using the software on the five aforementioned examples are shown. In Section 6, a discussion of the capabilities of the software that are demonstrated by the results

obtained using the software is provided. In Section 7, possible limitations of the software are discussed. Finally, in Section 8, conclusions on the work described in this article are provided.

2 GENERAL MULTIPLE-PHASE OPTIMAL CONTROL PROBLEMS

The general multiple-phase optimal control problem that can be solved by $\mathbb{CGPOPS}$ is given as follows. Without loss of generality, consider the following general multiple-phase optimal control problem where each phase is defined on the interval $t \in [t_0^{(p)}, t_f^{(p)}]$. First, let $p \in \{1, \ldots, P\}$ be the phase number, where P is the total number of phases. Determine the state $\mathbf{y}^{(p)}(t) \in \mathbb{R}^{1\times n_y^{(p)}}$, the control $\mathbf{u}^{(p)}(t) \in \mathbb{R}^{1\times n_u^{(p)}}$, the integrals $\mathbf{q}^{(p)} \in \mathbb{R}^{1\times n_q^{(p)}}$, the start times $t_0^{(p)} \in \mathbb{R}$, and the terminus times $t_f^{(p)} \in \mathbb{R}$ in all phases $p \in \{1, \ldots, P\}$, along with the static parameters $\mathbf{s} \in \mathbb{R}^{1\times n_s}$ that minimize the objective functional

$$\mathcal{J} = \phi\left(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(P)}, \mathbf{s}\right), \tag{1}$$

subject to the dynamic constraints

$$\frac{d\mathbf{y}^{(p)}}{dt} \equiv \dot{\mathbf{y}}^{(p)} = \mathbf{a}^{(p)}\left(\mathbf{y}^{(p)}(t), \mathbf{u}^{(p)}(t), t, \mathbf{s}\right), \quad p \in \{1, \ldots, P\}, \tag{2}$$

the event constraints

$$\mathbf{b}_{\min} \leq \mathbf{b}\left(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(P)}, \mathbf{s}\right) \leq \mathbf{b}_{\max}, \tag{3}$$

the inequality path constraints

$$\mathbf{c}_{\min}^{(p)} \leq \mathbf{c}^{(p)}\left(\mathbf{y}^{(p)}(t), \mathbf{u}^{(p)}(t), t, \mathbf{s}\right) \leq \mathbf{c}_{\max}^{(p)}, \quad p \in \{1, \ldots, P\}, \tag{4}$$

the integral constraints

$$\mathbf{q}_{\min}^{(p)} \leq \mathbf{q}^{(p)} \leq \mathbf{q}_{\max}^{(p)}, \quad p \in \{1, \ldots, P\}, \tag{5}$$

and the static parameter constraints

$$\mathbf{s}_{\min} \leq \mathbf{s} \leq \mathbf{s}_{\max}, \tag{6}$$

where

$$\mathbf{e}^{(p)} = \left[\mathbf{y}^{(p)}(t_0^{(p)}), t_0^{(p)}, \mathbf{y}^{(p)}(t_f^{(p)}), t_f^{(p)}, \mathbf{q}^{(p)}\right], \quad p \in \{1, \ldots, P\}, \tag{7}$$

and the integral vector components in each phase are defined as

$$\begin{aligned} q_j^{(p)} \quad = \quad & \int_{t_0^{(p)}}^{t_f^{(p)}} g_j^{(p)}\left(\mathbf{y}^{(p)}(t), \mathbf{u}^{(p)}(t), t, \mathbf{s}\right) dt, \\ & j \in \left\{1, \ldots, n_q^{(p)}\right\}, p \in \{1, \ldots, P\}. \end{aligned} \tag{8}$$

It is important to note that the event constraints of Equation (3) contain functions that can relate information at the start and/or terminus of any phase (including any relationships involving any integral or static parameters), with phases not needing to be in sequential order to be linked. Moreover, it is noted that the approach to linking phases is based on well-known formulations in the literature, such as those given in Refs. [9, 11]. A schematic of how phases can potentially be linked is given in Figure 1.

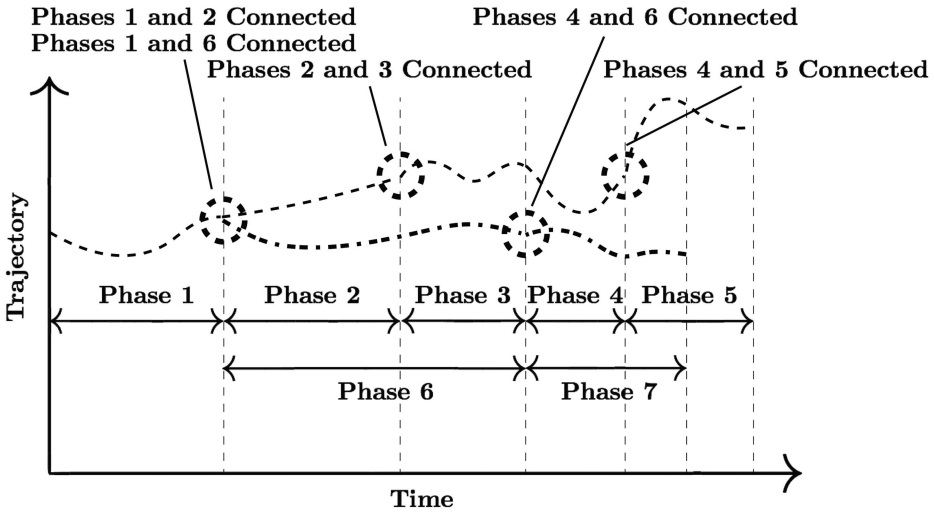


Fig. 1. Schematic of linkages for the multiple-phase optimal control problem. The schematic consists of seven phases where the termini of phases 1, 2, and 4 are linked to the starts of phases 2, 3, and 5, respectively, whereas the termini of phases 1 and 6 are linked to the starts of phases 6 and 4, respectively.

3 LGR COLLOCATION METHOD

As stated at the outset, the objective of this research is to provide researchers a computationally efficient general-purpose optimal control software for solving of a wide variety of complex constrained continuous optimal control problems using direct collocation. Although in principle any collocation method can be used to approximate the optimal control problem given in Section 2, in this research the LGR collocation method [23–25, 45, 47, 48, 50] is employed. It is noted that the NLP arising from the LGR collocation method has an elegant sparse structure that can be exploited as described in Refs. [3, 51, 52]. In addition, the LGR collocation method has a well-established convergence theory as described in Refs. [17, 36–39].

In the context of this research, a multiple-interval form of the LGR collocation method is chosen. In the multiple-interval LGR collocation method, for each phase p of the optimal control problem (where the phase number $p \in \{1, \ldots, P\}$ has been omitted to improve clarity of the description of the method), the time interval $t \in [t_0, t_f]$ is converted into the domain $\tau \in [-1, +1]$ using the affine transformation,

$$\begin{aligned} t &= \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}, \\ \tau &= 2\frac{t - t_0}{t_f - t_0} - 1. \end{aligned} \tag{9}$$

The domain $\tau \in [-1, +1]$ is then divided into K mesh intervals, $\mathcal{S}_k = [T_{k-1}, T_k] \subseteq [-1, +1]$, $k \in \{1, \ldots, K\}$ such that

$$\bigcup_{k=1}^{K} \mathcal{S}_k = [-1, +1], \quad \bigcap_{k=1}^{K} \mathcal{S}_k = \{T_1, \ldots, T_{K-1}\}, \tag{10}$$

and $-1 = T_0 < T_1 < \ldots < T_{K-1} < T_K = +1$. For each mesh interval, the LGR points used for collocation are defined in the domain of $[T_{k-1}, T_k]$ for $k \in \{1, \ldots, K\}$. The control is parameterized at the collocation points within each mesh interval. The state of the continuous optimal control

problem is then approximated in mesh interval $\mathcal{S}_k$, $k \in \{1, \ldots, K\}$, as

$$\mathbf{y}^{(k)}(\tau) \approx \mathbf{Y}^{(k)}(\tau) = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \ell_j^{(k)}(\tau), \quad \ell_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k+1} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}}, \tag{11}$$

where $\ell_j^{(k)}(\tau)$ for $j \in \{1, \ldots, N_k + 1\}$ is a basis of Lagrange polynomials, $(\tau_1^{(k)}, \ldots, \tau_{N_k}^{(k)})$ are the set of N_k LGR [1] collocation points in the interval $[T_{k-1}, T_k)$ in $\mathcal{S}_k$, $\tau_{N_k+1}^{(k)} = T_k$ is a noncollocated support point, and $\mathbf{Y}_j^{(k)} \equiv \mathbf{Y}^{(k)}(\tau_j^{(k)})$. Differentiating $\mathbf{Y}^{(k)}(\tau)$ in Equation (11) with respect to τ gives

$$\frac{d\mathbf{Y}^{(k)}(\tau)}{d\tau} = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \frac{d\ell_j^{(k)}(\tau)}{d\tau}. \tag{12}$$

The dynamics are then approximated at the N_k LGR points in mesh interval $k \in \{1, \ldots, K\}$ as

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} = \frac{t_f - t_0}{2} \mathbf{a}\left(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t\left(\tau_i^{(k)}, t_0, t_f\right), \mathbf{s}\right), \quad i \in \{1, \ldots, N_k\}, \tag{13}$$

where

$$D_{ij}^{(k)} = \frac{d\ell_j^{(k)}(\tau_i^{(k)})}{d\tau}, \quad i \in \{1, \ldots, N_k\}, j \in \{1, \ldots, N_k + 1\}$$

are the elements of the $N_k \times (N_k + 1)$ LGR differentiation matrix [24] in mesh interval $\mathcal{S}_k$, $k \in \{1, \ldots, K\}$, and $\mathbf{U}_i^{(k)}$ is the parameterized control at the i^{th} collocation point in mesh interval k. Finally, reintroducing the phase notation $p \in \{1, \ldots, P\}$, the phases of the problem are linked together by the event constraints

$$\mathbf{b}_{\min} \leq \mathbf{b}\left(\mathbf{E}^{(1)}, \ldots, \mathbf{E}^{(P)}, \mathbf{s}\right) \leq \mathbf{b}_{\max}, \tag{14}$$

where $\mathbf{E}^{(p)}$ is the endpoint approximation vector for phase p defined as

$$\mathbf{E}^{(p)} = \left[\mathbf{Y}_1^{(p)}, t_0^{(p)}, \mathbf{Y}_{N^{(p)}+1}^{(p)}, t_f^{(p)}, \mathbf{Q}^{(p)}\right] \tag{15}$$

such that $N^{(p)}$ is the total number of collocation points used in phase p given by

$$N^{(p)} = \sum_{k=1}^{K^{(p)}} N_k^{(p)}, \tag{16}$$

and $\mathbf{Q}^{(p)} \in \mathbb{R}^{1 \times n_q^{(p)}}$ is the integral approximation vector in phase p.

The aforementioned LGR discretization then leads to the following NLP. Minimize the objective function

$$\mathcal{J} = \phi\left(\mathbf{E}^{(1)}, \ldots, \mathbf{E}^{(P)}, \mathbf{s}\right), \tag{17}$$

subject to the defect constraints

$$\Delta^{(p)} = \mathbf{D}^{(p)} \mathbf{Y}^{(p)} - \frac{t_f^{(p)} - t_0^{(p)}}{2} \mathbf{A}^{(p)} = \mathbf{0}, \quad p \in \{1, \ldots, P\}, \tag{18}$$

the path constraints

$$\mathbf{c}_{\min}^{(p)} \leq \mathbf{C}_i^{(p)} \leq \mathbf{c}_{\max}^{(p)}, \quad i \in \{1, \ldots, N^{(p)}\}, p \in \{1, \ldots, P\}, \tag{19}$$

the event constraints

$$\mathbf{b}_{\min} \leq \mathbf{b}\left(\mathbf{E}^{(1)}, \ldots, \mathbf{E}^{(P)}, \mathbf{s}\right) \leq \mathbf{b}_{\max}, \tag{20}$$

the integral constraints

$$\mathbf{q}_{\min}^{(p)} \leq \mathbf{Q}^{(p)} \leq \mathbf{q}_{\max}^{(p)}, \quad p \in \{1, \ldots, P\}, \tag{21}$$

the static parameter constraints

$$\mathbf{s}_{\min} \leq \mathbf{s} \leq \mathbf{s}_{\max}, \tag{22}$$

and the integral approximation constraints

$$\boldsymbol{\rho}^{(p)} = \mathbf{Q}^{(p)} - \frac{t_f^{(p)} - t_0^{(p)}}{2} \left[\mathbf{w}^{(p)}\right]^{\mathsf{T}} \mathbf{G}^{(p)} = \mathbf{0}, \quad p \in \{1, \ldots, P\}, \tag{23}$$

where

$$\mathbf{A}^{(p)} = \begin{bmatrix} \mathbf{a}^{(p)}\left(\mathbf{Y}_1^{(p)}, \mathbf{U}_1^{(p)}, t_1^{(p)}, \mathbf{s}\right) \\ \vdots \\ \mathbf{a}^{(p)}\left(\mathbf{Y}_{N^{(p)}}^{(p)}, \mathbf{U}_{N^{(p)}}^{(p)}, t_{N^{(p)}}^{(p)}, \mathbf{s}\right) \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_y^{(p)}}, \tag{24}$$

$$\mathbf{C}^{(p)} = \begin{bmatrix} \mathbf{c}^{(p)}\left(\mathbf{Y}_1^{(p)}, \mathbf{U}_1^{(p)}, t_1^{(p)}, \mathbf{s}\right) \\ \vdots \\ \mathbf{c}^{(p)}\left(\mathbf{Y}_{N^{(p)}}^{(p)}, \mathbf{U}_{N^{(p)}}^{(p)}, t_{N^{(p)}}^{(p)}, \mathbf{s}\right) \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_c^{(p)}}, \tag{25}$$

$$\mathbf{G}^{(p)} = \begin{bmatrix} \mathbf{g}^{(p)}\left(\mathbf{Y}_1^{(p)}, \mathbf{U}_1^{(p)}, t_1^{(p)}, \mathbf{s}\right) \\ \vdots \\ \mathbf{g}^{(p)}\left(\mathbf{Y}_{N^{(p)}}^{(p)}, \mathbf{U}_{N^{(p)}}^{(p)}, t_{N^{(p)}}^{(p)}, \mathbf{s}\right) \end{bmatrix} \in \mathbb{R}^{N^{(p)} \times n_q^{(p)}}, \tag{26}$$

$\mathbf{D}^{(p)} \in \mathbb{R}^{N^{(p)} \times [N^{(p)}+1]}$ is the LGR differentiation matrix in phase $p \in \{1, \ldots, P\}$, and $\mathbf{w}^{(p)} \in \mathbb{R}^{N^{(p)} \times 1}$ are the LGR weights at each node in phase p. It is noted that $\mathbf{a}^{(p)} \in \mathbb{R}^{1 \times n_y^{(p)}}$, $\mathbf{c}^{(p)} \in \mathbb{R}^{1 \times n_c^{(p)}}$, and $\mathbf{g}^{(p)} \in \mathbb{R}^{1 \times n_q^{(p)}}$ correspond respectively to the functions that define the right-hand side of the dynamics, the path constraints, and the integrands in phase $p \in \{1, \ldots, P\}$, where $n_y^{(p)}$, $n_c^{(p)}$, and $n_q^{(p)}$ are respectively the number of state components, path constraints, and integral components in phase p. Finally, the state matrix, $\mathbf{Y}^{(p)} \in \mathbb{R}^{[N^{(p)}+1] \times n_y^{(p)}}$, and the control matrix, $\mathbf{U}^{(p)} \in \mathbb{R}^{N^{(p)} \times n_u^{(p)}}$, in phase $p \in \{1, \ldots, P\}$ are formed as

$$\mathbf{Y}^{(p)} = \begin{bmatrix} \mathbf{Y}_1^{(p)} \\ \vdots \\ \mathbf{Y}_{N^{(p)}+1}^{(p)} \end{bmatrix} \text{ and } \mathbf{U}^{(p)} = \begin{bmatrix} \mathbf{U}_1^{(p)} \\ \vdots \\ \mathbf{U}_{N^{(p)}}^{(p)} \end{bmatrix}, \tag{27}$$

respectively, where $n_u^{(p)}$ is the number of control components in phase p.

4 MAJOR COMPONENTS OF ℂGℙOℙS

In this section, we describe the major components of the C++ software ℂGℙOℙS that implement the aforementioned LGR collocation method. In Section 4.1, the large sparse NLP associated with the LGR collocation method is described. In Section 4.2, the structure of the NLP described in Section 4.1 is shown. In Section 4.3, the method for scaling the NLP via scaling of the optimal control problem is overviewed. In Section 4.4, the approach for estimating the derivatives required by the NLP solver is explained. In Section 4.5, the method for determining the dependencies of each optimal control function to provide the most sparse NLP derivative matrices to the NLP solver is presented. In Section 4.6, the *hp* mesh refinement methods that are included in the software

to iteratively determine a mesh that satisfies a user-specified accuracy tolerance are described. Finally, in Section 4.7, we provide a high-level description of the algorithmic flow of $\mathbb{CGPOPS}$.

4.1 Sparse NLP Arising from the Radau Collocation Method

The resulting NLP that arises when using LGR collocation to discretize the continuous optimal control problem is given as follows. Determine the NLP decision vector, $\mathbf{z}$, that minimizes the NLP objective function,

$$f(\mathbf{z}), \tag{28}$$

subject to the constraints

$$\mathbf{H}_{\min} \leq \mathbf{H}(\mathbf{z}) \leq \mathbf{H}_{\max} \tag{29}$$

and the variable bounds

$$\mathbf{z}_{\min} \leq \mathbf{z} \leq \mathbf{z}_{\max}. \tag{30}$$

It is noted that although the size of the NLP arising from the LGR collocation method changes depending upon the number of mesh intervals and LGR points used in each phase, the structure of the NLP remains the same regardless of the size of the NLP. Finally, in the sections that follow, the subscript ":" denotes either a row or a column, where the ":" notation is analogous to the syntax used in the MATLAB programming language.

4.1.1 NLP Variables. For a continuous optimal control problem transcribed into P phases, the NLP decision vector, $\mathbf{z}$, has the following form:

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}^{(1)} \\ \vdots \\ \mathbf{z}^{(P)} \\ s_1 \\ \vdots \\ s_{n_s} \end{bmatrix}, \quad \text{where } \mathbf{z}^{(p)} = \begin{bmatrix} \mathbf{Y}^{(p)}_{(:,1)} \\ \vdots \\ \mathbf{Y}^{(p)}_{(:,n_y^{(p)})} \\ \mathbf{U}^{(p)}_{(:,1)} \\ \vdots \\ \mathbf{U}^{(p)}_{(:,n_u^{(p)})} \\ (\mathbf{Q}^{(p)})^{\mathsf{T}} \\ t_0^{(p)} \\ t_f^{(p)} \end{bmatrix}, \tag{31}$$

$\mathbf{Y}^{(p)} \in \mathbb{R}^{[N^{(p)}+1]\times n_y^{(p)}}$ is the state approximation matrix (see Equation (27)), $\mathbf{U}^{(p)} \in \mathbb{R}^{N^{(p)}\times n_u^{(p)}}$ is the control parameterization matrix (see Equation (27)), $\mathbf{Q}^{(p)} \in \mathbb{R}^{1\times n_q^{(p)}}$ is the integral approximation vector, and $t_0^{(p)}$ and $t_f^{(p)}$ are scalars of the initial and final time, respectively, for phase $p \in \{1, \ldots, P\}$, and s_i for $i \in \{1, \ldots, n_s\}$ are the static parameters appearing throughout the entire problem.

4.1.2 NLP Objective and Constraint Functions. The NLP objective function, $f(\mathbf{z})$, is given in the form

$$f(\mathbf{z}) = \phi(\mathbf{E}^{(1)}, \ldots, \mathbf{E}^{(P)}, \mathbf{s}), \tag{32}$$

where $\mathbf{E}^{(p)}, p \in \{1, \ldots, P\}$, is the endpoint approximation vector defined in Equation (15), and the typical cost functional of a general multiple-phase optimal control problem has been turned simply

into a Mayer cost function by using the integral approximation vector, $\mathbf{Q}^{(p)}$, to approximate the Lagrange cost in each phase p. The NLP constraint vector, $\mathbf{H}(\mathbf{z})$, is given in the form

$$\mathbf{H}(\mathbf{z}) = \begin{bmatrix} \mathbf{h}^{(1)} \\ \vdots \\ \mathbf{h}^{(P)} \\ \mathbf{b} \end{bmatrix}, \text{ where } \mathbf{h}^{(p)} = \begin{bmatrix} \Delta^{(p)}_{(:,1)} \\ \vdots \\ \Delta^{(p)}_{(:,n_y^{(p)})} \\ \mathbf{C}^{(p)}_{(:,1)} \\ \vdots \\ \mathbf{C}^{(p)}_{(:,n_c^{(p)})} \\ (\boldsymbol{\rho}^{(p)})^{\mathsf{T}} \end{bmatrix}, \quad p = \{1, \ldots, P\}, \tag{33}$$

$\Delta^{(p)} \in \mathbb{R}^{N^{(p)} \times n_y^{(p)}}$, $\boldsymbol{\rho}^{(p)} \in \mathbb{R}^{1 \times n_q^{(p)}}$, and $\mathbf{C}^{(p)} \in \mathbb{R}^{N^{(p)} \times n_c^{(p)}}$ are respectively the defect constraint matrix, the integral approximation constraint vector, and the path constraint matrix in phase $p \in \{1, \ldots, P\}$, and $\mathbf{b} \in \mathbb{R}^{n_b \times 1}$ is the event constraint vector for the entire problem. The defect constraint matrix, integral approximation constraint vector, and path constraint matrix in phase p are defined by Equations (18), (23), and (25), respectively. It is noted that the constraints are divided into the equality defect and integral constraints

$$\begin{array}{rcl} \Delta^{(p)} & = & \mathbf{0}, \\ \boldsymbol{\rho}^{(p)} & = & \mathbf{0}, \end{array} \quad p \in \{1, \ldots, P\}, \tag{34}$$

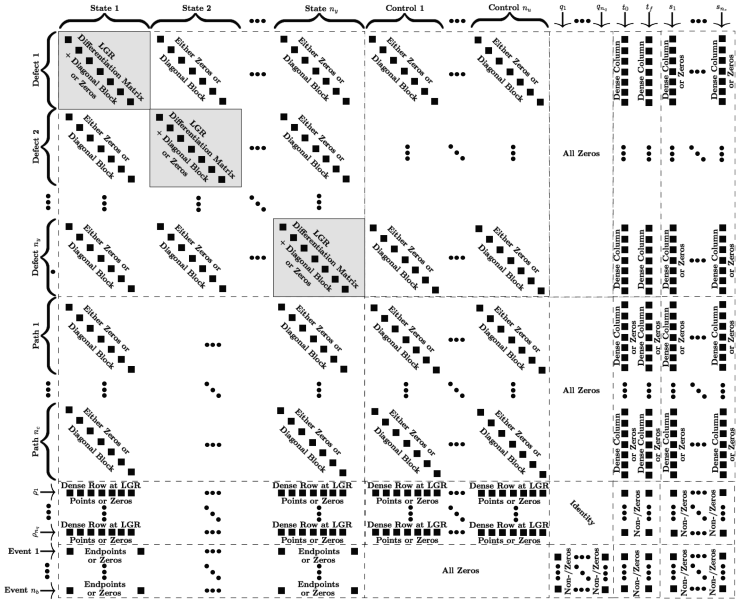
and the inequality discretized path and event constraints

$$\begin{array}{ccccc} \mathbf{c}^{(p)}_{\min} & \leq & \mathbf{C}^{(p)}_i & \leq & \mathbf{c}^{(p)}_{\max}, \\ \mathbf{b}_{\min} & \leq & \mathbf{b} & \leq & \mathbf{b}_{\max}. \end{array} \quad i \in \{1, \ldots, N^{(p)}\}, p \in \{1, \ldots, P\}, \tag{35}$$

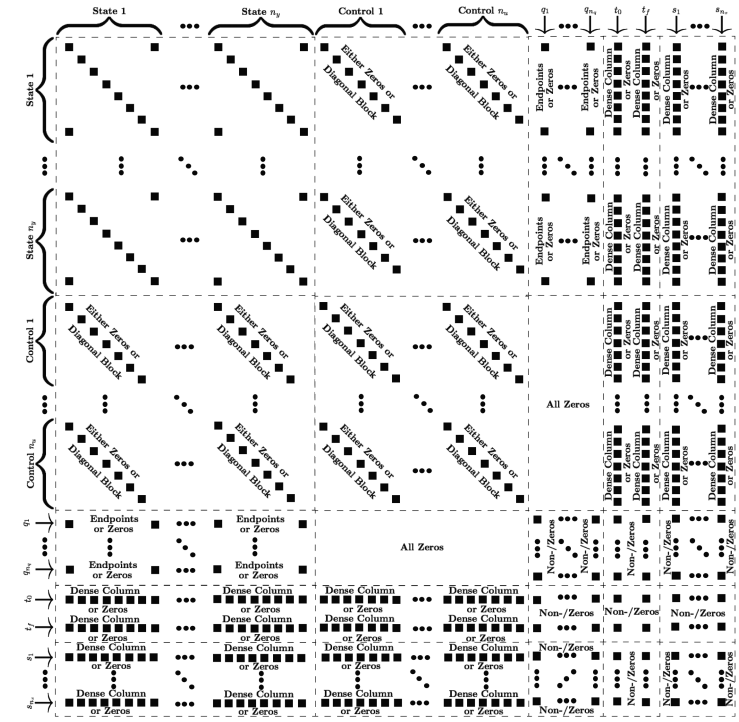
4.2 Sparse Structure of NLP Derivative Functions

The structure of the NLP created by the LGR collocation method is presented in detail in Refs. [3, 51]. Specifically, Refs. [51] and [3] respectively describe the sparse structure of the NLP for the differential form of the LGR collocation method for the single- and multiple-phase optimal control problem. As described in Section 4.1.1, the values of the state approximation coefficients at the discretization points, the control parameters at the collocation points, the initial time, the final time, and the integral approximation vector of each phase, as well as any static parameters of the problem, make up the NLP decision vector. The NLP constraint vector consists of the defect constraints and path constraints applied at each of the collocation points, as well as any integral approximation constraints, for each phase, and event constraints, as described in Section 4.1.2. The derivation of the NLP derivative matrices in terms of the original continuous optimal control problem functions is described in detail in Refs. [3, 51, 52] and is beyond the scope of this article. However, it is noted that the sparsity exploitation derived in Refs. [3, 51, 52] requires computing partial derivatives of the continuous optimal control problem functions on the first- and second-order derivative levels.

Examples of the sparsity patterns of the NLP constraint Jacobian and Lagrangian Hessian are respectively shown in Figure 2(a) and (b) for a single-phase optimal control problem. It is noted that for the NLP constraint Jacobian, the off-diagonal phase blocks relating constraints in phase i to variables in phase j for $i \neq j$ are all zeros. Similarly, for the NLP Lagrangian Hessian, the off-diagonal phase blocks relating variables in phase i to variables in phase j for $i \neq j$ are all zeros except for the variables making up the endpoint vectors that may be related via the objective



(a) NLP Constraint Jacobian



(b) NLP Lagrangian Hessian

Fig. 2. Example sparsity patterns for the single-phase optimal control problem containing n_y state components, n_u control components, and n_q integral components, as well as n_c path constraints, n_s static parameters, and n_b event constraints.

function or event constraints. The sparsity patterns shown in Figure 2 are determined explicitly by identifying the derivative dependencies of the NLP objective and constraints functions with respect to the NLP decision vector variables. It is noted that the phases are connected using the initial and terminal values of the time and state in each phase along with the static parameters.

4.3 Scaling of the Optimal Control Problem for the NLP

The NLP described in Section 4.1 must be well scaled for the NLP solver to obtain a solution. ℂ𝔾ℙ𝕆ℙ𝕊 includes the option for the NLP to be scaled automatically by scaling the continuous optimal control problem. The approach to automatic scaling is to scale the variables and the first derivatives of the optimal control functions to be $\approx O(1)$. First, the optimal control variables are scaled to lie on the unit interval $[-1/2, 1/2]$ and is accomplished as follows. Suppose that it is desired to scale an arbitrary variable $x \in [a, b]$ to $\tilde{x}$ such that $\tilde{x} \in [-1/2, 1/2]$. This variable scaling is accomplished via the affine transformation

$$\tilde{x} = v_x x + r_x, \tag{36}$$

where v_x and r_x are the variable scale and shift, respectively, defined as

$$\begin{aligned} v_x &= \frac{1}{b-a}, \\ r_x &= \frac{1}{2} - \frac{b}{b-a}. \end{aligned} \tag{37}$$

Every variable in the continuous optimal control problem is scaled using Equations (36) and (37). Next, the Jacobian of the NLP constraints can be made $\approx O(1)$ by scaling the derivatives of the optimal control functions to be approximately unity. First, using the approach derived in Ref. [11], in ℂ𝔾ℙ𝕆ℙ𝕊 the defect constraints are scaled using the same scale factors as were used to scale the state. Next, the objective function, event constraints, and path constraints scale factors are obtained by sampling the gradient of each constraint at a variety of sample points within the bounds of the unscaled optimal control problem and taking the average norm of each gradient across all sample points.

4.4 Computation Derivatives of NLP Functions

The NLP derivative functions are obtained by exploiting the sparse structure of the NLP arising from the *hp* LGR collocation method. Specifically, in Refs. [3, 51], it has been shown that by using the derivative form of the LGR collocation method, the NLP derivatives can be obtained by computing the derivatives of the optimal control problem functions at the LGR points and inserting these derivatives into the appropriate locations in the NLP derivative functions. In ℂ𝔾ℙ𝕆ℙ𝕊, the optimal control derivative functions are approximated using one of four types of derivative estimation methods: sparse central finite differencing, bicomplex-step derivative approximations, hyper-dual derivative approximations, and automatic differentiation.

4.4.1 Central Finite Difference. To see how the central finite-difference derivative approximation works in practice, consider the function $\mathbf{f}(\mathbf{x})$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is one of the *optimal control functions* (i.e., n and m are respectively the size of an optimal control variable and an optimal control function). Then, $\partial \mathbf{f}/\partial \mathbf{x}$ is approximated using a central finite difference as

$$\frac{\partial \mathbf{f}}{\partial x_i} \approx \frac{\mathbf{f}(\mathbf{x} + \mathbf{h}_i) - \mathbf{f}(\mathbf{x} - \mathbf{h}_i)}{2h}, \tag{38}$$

where $\mathbf{h}_i$ arises from perturbing the i^{th} component of $\mathbf{x}$. The vector $\mathbf{h}_i$ is computed as

$$\mathbf{h}_i = h_i \mathbf{e}_i, \tag{39}$$

where $\mathbf{e}_i$ is the i^{th} row of the $n \times n$ identity matrix and h_i is the perturbation size associated with x_i. The perturbation h_i is computed using the equation

$$h_i = h(1 + |x_i|), \tag{40}$$

where the base perturbation size h is chosen to be the optimal step size for a function whose input and output are $\approx O(1)$ as described in Ref. [27]. Second derivative approximations are computed in a manner similar to that used for first derivative approximations with the key difference being that perturbations in two variables are performed. For example, $\partial^2\mathbf{f}/\partial x_i \partial x_j$ can be approximated using a central finite-difference approximation as

$$\frac{\partial^2 \mathbf{f}(\mathbf{x})}{\partial x_i \partial x_j} \approx \frac{\mathbf{f}(\mathbf{x} + \mathbf{h}_i + \mathbf{h}_j) - \mathbf{f}(\mathbf{x} + \mathbf{h}_i - \mathbf{h}_j) - \mathbf{f}(\mathbf{x} - \mathbf{h}_i + \mathbf{h}_j) + \mathbf{f}(\mathbf{x} - \mathbf{h}_i - \mathbf{h}_j)}{4h_i h_j}, \tag{41}$$

where $\mathbf{h}_i$, $\mathbf{h}_j$, h_i, and h_j are as defined in Equations (39) and (40). The base perturbation size is chosen to minimize round-off error in the finite-difference approximation. Furthermore, it is noted that $h_i \to h$ as $|x_i| \to 0$.

4.4.2 Bicomplex-Step. To see how the bicomplex-step derivative approximation works in practice, consider the function $\mathbf{f}(\mathbf{x})$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is one of the *optimal control functions* (i.e., n and m are respectively the size of an optimal control variable and an optimal control function). Then, $\partial\mathbf{f}/\partial\mathbf{x}$ is approximated using a bicomplex-step derivative approximation as

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i} \approx \frac{\mathrm{Im}_1[f(\mathbf{x} + i_1 h \mathbf{e}_i)]}{h}, \tag{42}$$

where $\mathrm{Im}_1[\cdot]$ denotes the imaginary i_1 component of the function evaluated with the perturbed bicomplex input, $\mathbf{e}_i$ is the i^{th} row of the $n \times n$ identity matrix, and the base perturbation size h is chosen to be a step size that will minimize truncation error while refraining from encountering round-off error due to bicomplex arithmetic, which is described in detail in Ref. [46] and is beyond the scope of this article. It is noted that the imaginary component i_1 has the property $i_1^2 = -1$. Second derivative approximations are computed in a manner similar to that used for first derivative approximations with the key difference being that perturbations in two variables are performed in two separate imaginary directions. For example, $\partial^2\mathbf{f}/\partial x_i \partial x_j$ can be approximated using a bicomplex-step derivative approximation as

$$\frac{\partial^2 \mathbf{f}(\mathbf{x})}{\partial x_i \partial x_j} \approx \frac{\mathrm{Im}_{1,2}[f(\mathbf{x} + i_1 h \mathbf{e}_i + i_2 h \mathbf{e}_j)]}{h^2}, \tag{43}$$

where $\mathrm{Im}_{1,2}[\cdot]$ denotes the imaginary $i_1 i_2$ component of the function evaluated with the perturbed bicomplex input, where it is noted that $i_2^2 = -1$, and $i_1 i_2$ is a bi-imaginary direction distinct from either the i_1 or i_2 imaginary directions (i.e., $i_1 i_2 = i_2 i_1$).

4.4.3 Hyper-Dual. To see how the hyper-dual derivative approximation works in practice, consider the function $\mathbf{f}(\mathbf{x})$, where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is one of the *optimal control functions* (i.e., n and m are respectively the size of an optimal control variable and an optimal control function). Then, $\partial\mathbf{f}/\partial\mathbf{x}$ is approximated using a hyper-dual derivative approximation as

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i} = \frac{\mathrm{Ep}_1[f(\mathbf{x} + \epsilon_1 h \mathbf{e}_i)]}{h}, \tag{44}$$

where $\mathrm{Ep}_1[\cdot]$ denotes the imaginary ϵ_1 component of the function evaluated with the perturbed hyper-dual input, $\mathbf{e}_i$ is the i^{th} row of the $n \times n$ identity matrix, and the base perturbation size h is chosen to be unity because for first and second derivatives the hyper-dual arithmetic does not suffer from either truncation or round-off error (described in detail in Ref. [22] and beyond

the scope of this article). It is noted that the imaginary component ϵ_1 has the property of being nilpotent (i.e., $\epsilon_1^2 = 0$). Second derivative approximations are computed in a manner similar to that used for first derivative approximations with the key difference being that perturbations in two variables are performed in two separate imaginary directions. For example, $\partial^2\mathbf{f}/\partial x_i \partial x_j$ can be approximated using a hyper-dual derivative approximation as

$$\frac{\partial^2 \mathbf{f}(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\mathrm{Ep}_{1,2}[f(\mathbf{x} + \epsilon_1 h\mathbf{e}_i + \epsilon_2 h\mathbf{e}_j)]}{h^2}, \tag{45}$$

where $\mathrm{Ep}_{1,2}[\cdot]$ denotes the imaginary $\epsilon_1\epsilon_2$ component of the function evaluated with the perturbed hyper-dual input, where it is noted that ϵ_2 also has the property of being nilpotent (i.e., $\epsilon_2^2 = 0$), and $\epsilon_1\epsilon_2$ is a bi-imaginary direction distinct from either the ϵ_1 or ϵ_2 imaginary directions (i.e., $\epsilon_1\epsilon_2 = \epsilon_2\epsilon_1$).

4.4.4 Automatic Differentiation. In this section, the basis of automatic differentiation is discussed. As described in Ref. [49], automatic (algorithmic) differentiation may be derived from the unifying chain rule and supplies numerical evaluations of the derivative for a defined computer program by decomposing the program into a sequence of elementary function operations and applying the calculus chain rule algorithmically through the computer [32]. The process of automatic differentiation is described in detail in Ref. [32] and is beyond the scope of this article. It is noted, however, that the first- and second-order partial derivatives obtained using the Taylor series–based derivative approximation methods described in Sections 4.4.1 through 4.4.3 may be computed to machine precision using automatic differentiation. Specifically, ℂ𝔾ℙ𝕆ℙ𝕊 employs the well-known open source software ADOL-C [31, 59] to compute derivatives using automatic differentiation.

4.5 Method for Determining the Optimal Control Function Dependencies

It can be seen from Section 4.2 that the NLP associated with the LGR collocation method has a sparse structure where the blocks of the constraint Jacobian and Lagrangian Hessian are dependent upon whether a particular NLP function depends upon a particular NLP variable, as was shown in Refs. [3, 51]. The method for identifying the optimal control function derivative dependencies in ℂ𝔾ℙ𝕆ℙ𝕊 utilizes the independent nature of the hyper-dual derivative approximations. Specifically, since the imaginary directions used for hyper-dual derivative approximations are completely independent of one another, second-order derivative approximations only appear nonzero if the partial actually exists (same for first-order derivative approximations). For example, suppose that $\mathbf{f}(\mathbf{x})$ is a function where $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ and $\mathbf{x} = [x_1 \dots x_n]$. The hyper-dual derivative approximation of $\partial^2\mathbf{f}(\mathbf{x})/\partial x_i \partial x_j$ will only be nonzero if the actual $\partial^2\mathbf{f}(\mathbf{x})/\partial x_i \partial x_j$ exists and is nonzero. Given this knowledge of the exact correspondence of hyper-dual derivative approximations to the actual derivative evaluations, identifying derivative dependencies of optimal control problem functions with respect to optimal control problem variables becomes simple, as existing partial derivatives will have nonzero outputs when approximated by the hyper-dual derivative approximations, whereas nonexisting partial derivatives will simply be zero always. To ensure that derivative dependencies are not mistakenly missed due to a derivative approximation happening to equal zero at the point at which it is being evaluated for an existing nonzero partial derivative, the hyper-dual derivative approximations are evaluated at multiple sample points within the variable bounds. In this manner, the derivative dependencies of the optimal control problem functions can be easily identified exactly for the first- and second-order derivative levels. The computational expense of identifying the derivative dependencies in this manner is minimal, whereas the exact second-order derivative sparsity pattern that is obtained can significantly reduce the cost of

computing the NLP Lagrangian Hessian when compared to using an over-estimated sparsity pattern as done in $\mathbb{GPOPS}-\mathbb{II}$ [52].

4.6 Adaptive Mesh Refinement

In the past few years, the subject of adaptive mesh refinement has been of considerable study in the efficient implementation of Gaussian quadrature collocation methods. The work on adaptive Gaussian quadrature mesh refinement has led to several articles in the literature including those found in Refs. [15, 16, 29, 47, 48, 50]. $\mathbb{CGPOPS}$ employs the recently developed mesh refinement methods described in Refs. [2, 15, 47, 48, 50]. The mesh refinement methods of Refs. [50], [15], [47], [48], and [2] are respectively referred to as the *hp*-I, *hp*-II, *hp*-III, *hp*-IV, and *hp*-BB methods. In all five of the *hp*-adaptive mesh refinement methods, the number of mesh intervals, width of each mesh interval, and degree of the approximating polynomial can be varied until a user-specified accuracy tolerance has been achieved. When using any of the methods in $\mathbb{CGPOPS}$, the terminology *hp*-Method($N_{\min}$, $N_{\max}$) refers to a method whose minimum and maximum allowable polynomial degrees within a mesh interval are $N_{\min}$ and $N_{\max}$, respectively. All five methods estimate the solution error using a relative difference between the state estimate and the integral of the dynamics at a modified set of LGR points. The key difference between the five methods lies in the manner in which the decision is made to either increase the number of collocation points in a mesh interval or to refine the mesh. In Ref. [15], the degree of the approximating polynomial is increased if the ratio of the maximum curvature over the mean curvature of the state in a particular mesh interval is below a user-specified threshold. However, Ref. [50] uses the exponential convergence property of the LGR collocation method and increases the polynomial degree within a mesh interval if the estimate of the required polynomial degree is less than a user-specified upper limit. Similarly, Refs. [47, 48] employ a nonsmoothness criterion to determine whether an *h* or *p* method should be used for a given mesh interval while also utilizing mesh reduction techniques to minimize the size of the transcribed NLP in regions of the solution where such high resolution is not required. If a *p* method refinement is prescribed for a given mesh interval and the estimate of the polynomial degree exceeds the allowed upper limit, the mesh interval is divided into more mesh intervals (i.e., *h* method employed). Last, the mesh refinement method developed in Ref. [2] is designed for bang-bang optimal control problems and employs estimates of the switching functions of the Hamiltonian to obtain the solution profile. In $\mathbb{CGPOPS}$, the user can choose between these five mesh refinement methods. Finally, it is noted that $\mathbb{CGPOPS}$ has been designed in a modular way, making it possible to add a new mesh refinement method in a relatively straightforward way if it is so desired.

4.7 Algorithmic Flow of $\mathbb{CGPOPS}$

In this section, we describe the operational flow of $\mathbb{CGPOPS}$ with the aid of Figure 3. First, the user provides a description of the optimal control problem that is to be solved. The properties of the optimal control problem are then extracted from the user description from which the state, control, time, and parameter dependencies of the optimal control problem functions are identified. Subsequently, assuming that the user has specified that the optimal control problem be scaled automatically, the optimal control problem scaling algorithm is called and these scale factors are determined and used to scale the NLP. The optimal control problem is then transcribed to a large sparse NLP and the NLP is solved on the initial mesh, where the initial mesh is either user supplied or determined by the default settings in $\mathbb{CGPOPS}$. Once the NLP is solved, the NLP solution is analyzed as a discrete approximation of the optimal control problem and the error in the discrete approximation for the current mesh is estimated. If the user-specified accuracy tolerance is met, the software terminates and outputs the solution. Otherwise, a new mesh is determined using one of the supplied mesh refinement algorithms and the resulting NLP is solved on the new mesh.

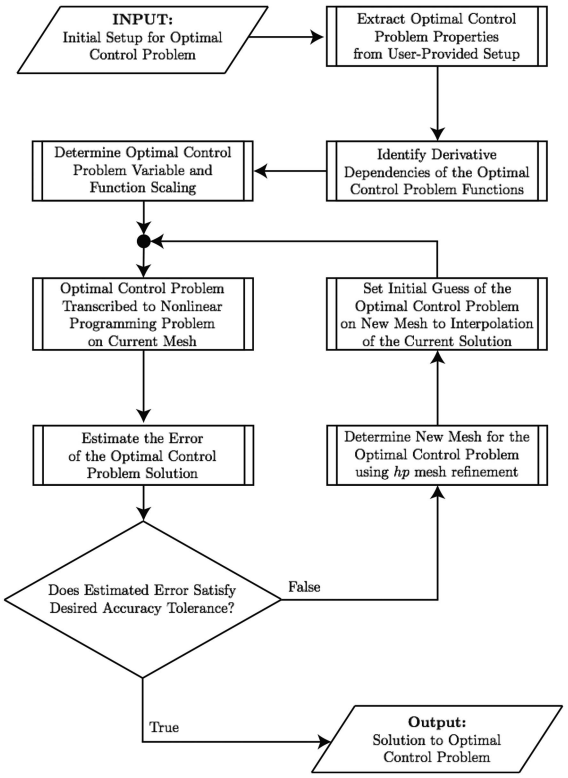
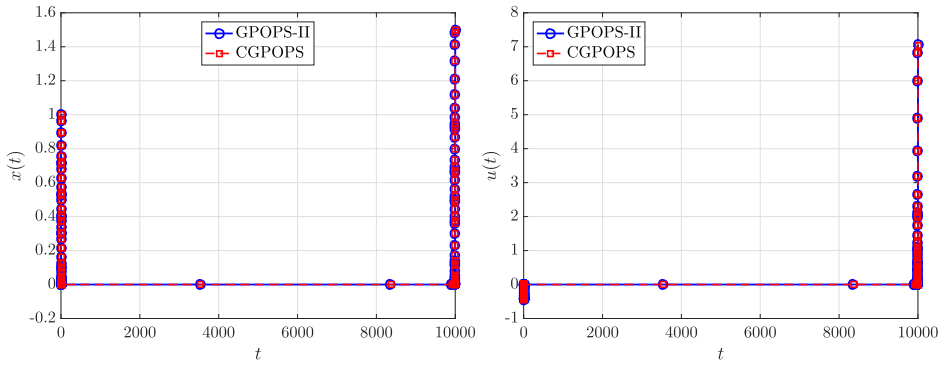


Fig. 3. Flowchart of ℂGℙOℙS algorithm.

5 EXAMPLES

ℂGℙOℙS is now demonstrated on five examples taken from the open literature. The first example is the hyper-sensitive optimal control problem taken from Ref. [55] and demonstrates the ability of ℂGℙOℙS to efficiently solve problems that have rapid changes in dynamics in particular regions of the solution. The second example is the reusable launch vehicle entry problem taken from Ref. [11] and demonstrates the efficiency of ℂGℙOℙS on a more realistic problem. The third example is the space station attitude optimal control problem taken from Refs. [11, 53] and demonstrates the efficiency of ℂGℙOℙS on a problem whose solution is highly nonintuitive. The fourth example is a free-flying robot problem taken from Ref. [11] and demonstrates the ability of ℂGℙOℙS to solve a bang-bang optimal control problem using discontinuity detection. The fifth example is a multiple-stage launch vehicle ascent problem taken from Refs. [7, 11, 54] and demonstrates the ability of ℂGℙOℙS to solve a problem with multiple phases.

All five examples were solved using the open source NLP solver IPOPT [12] in second derivative (full Newton) mode with the publicly available multifrontal massively parallel sparse direct linear solver MA57 [18]. All results were obtained using the differential form of the LGR collocation method and various forms of the aforementioned *hp* mesh refinement method using default NLP solver settings and the automatic scaling routine in ℂGℙOℙS. For ℂGℙOℙS, all first- and second-order derivatives for the NLP solver were obtained using hyper-dual derivative approximations as described in Section 4.4.3 with a perturbation step size of $h = 1$. All solutions obtained by ℂGℙOℙS are compared against the solutions obtained using the previously developed MATLAB software ℍℙOℙS − 𝕀𝕀 [52], which also employs *hp* LGR collocation methods. For ℍℙOℙS − 𝕀𝕀,



(a) $x(t)$ vs. t. (b) $u(t)$ vs. t.

Fig. 4. $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ solutions to Example 1 using *hp*-IV(3,10).

the first- and second-order derivatives for the NLP solver were obtained using the automatic differentiation software ADiGator [60] for all examples except the fifth example, which used sparse central finite differences. All computations were performed on a 2.9-GHz Intel Core i7 MacBook Pro running MAC OS-X version 10.13.6 (High Sierra) with 16-GB 2133-MHz LPDDR3 of RAM. C++ files were compiled using Apple LLVM version 9.1.0 (clang-1000.10.44.2). All m-scripts were executed using MATLAB version R2016a (build 9.0.0.341360). All plots were created using MATLAB version R2016a (build 9.0.0.341360).

5.1 Example 1: Hyper-Sensitive Problem

Consider the following optimal control problem taken from Ref. [55]. Minimize the objective functional

$$\mathcal{J} = \frac{1}{2}\int_0^{t_f} (x^2 + u^2)dt, \tag{46}$$

subject to the dynamic constraints

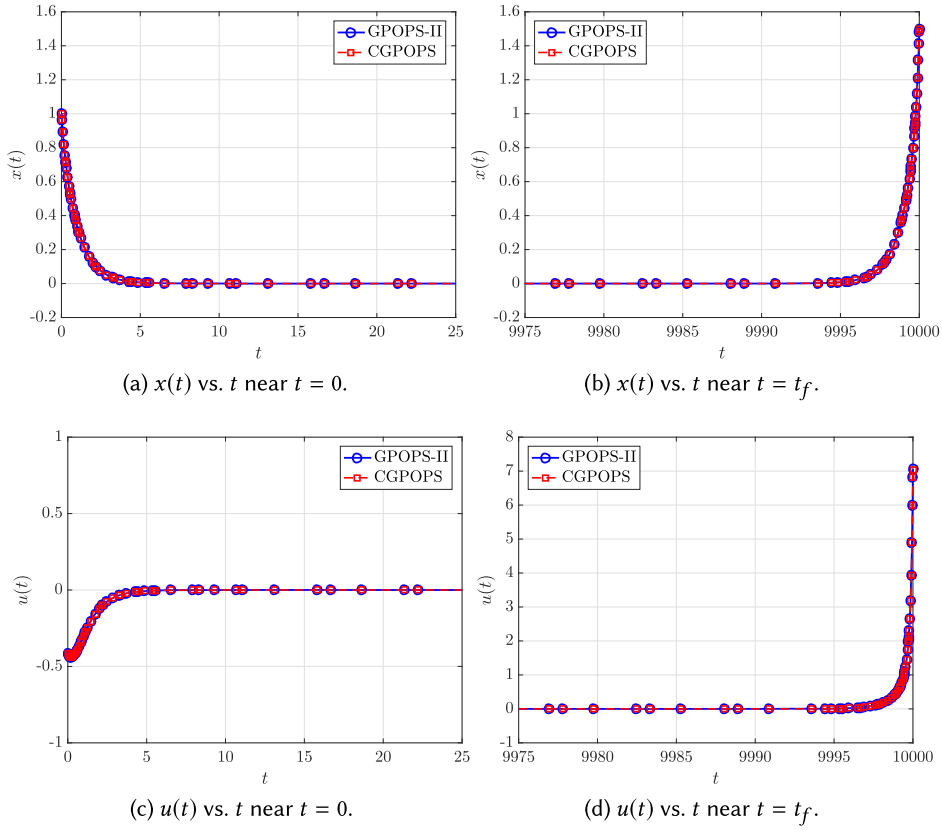
$$\dot{x} = -x^3 + u, \tag{47}$$

and the boundary conditions

$$x(0) = 1, \qquad x(t_f) = 1.5, \tag{48}$$

where $t_f = 10,000$. It is known for a sufficiently large value of t_f that the interesting behavior in the solution for the optimal control problem defined by Equations (46) through (48) occurs near $t = 0$ and $t = t_f$ (see Ref. [55] for details), whereas most of the solution is a constant. Given the structure of the solution, a majority of collocation points need to be placed near $t = 0$ and $t = t_f$.

The optimal control problem given in Equations (46) through (48) was solved using $\mathbb{CGPOPS}$ with the mesh refinement methods *hp*-I(3,10), *hp*-II(3,10), *hp*-III(3,10), and *hp*-IV(3,10) on an initial mesh of 10 evenly spaced mesh intervals with three LGR points per mesh interval. Furthermore, the NLP solver and mesh refinement accuracy tolerances were set to 10^{-7} and 10^{-6}, respectively. The solution obtained using $\mathbb{CGPOPS}$ with the *hp*-IV(3,10) method is shown in Figure 4 alongside the solution obtained using $\mathbb{GPOPS}-\mathbb{II}$ [52] with the *hp*-IV(3,10) method. It is seen that the $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ solutions are in excellent agreement. Moreover, the optimal objective obtained using both $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ was 3.3620559 to eight significant figures. Additionally, the computation time required by $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ to solve the optimal control problem was 0.2153 and 1.5230 seconds, respectively. To demonstrate how $\mathbb{CGPOPS}$ is capable of capturing the interesting features of the optimal solution, Figure 5 shows



(a) $x(t)$ vs. t near $t = 0$. (b) $x(t)$ vs. t near $t = t_f$.

(c) $u(t)$ vs. t near $t = 0$. (d) $u(t)$ vs. t near $t = t_f$.

Fig. 5. ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 solutions to Example 1 near $t = 0$ and $t = t_f$ using *hp*-IV(3,10).

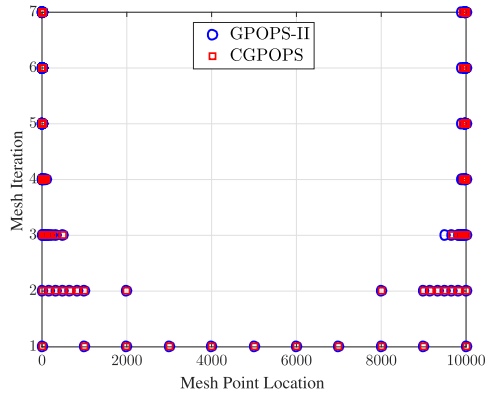


Fig. 6. ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 mesh refinement history for Example 1 using *hp*-IV(3,10).

the solution on the intervals $t \in [0, 25]$ (near the initial time) and $t \in [9975, 10000]$ (near the final time). It is seen that ℂ𝔾ℙ𝕆ℙ𝕊 accurately captures the rapid decay from $x(0) = 1$ and the rapid growth to meet the terminal condition $x(t_f) = 1.5$, with the density of the mesh points near $t = 0$ and $t = t_f$ increasing as the mesh refinement progresses. Additionally, Figure 6 shows the mesh

Table 1. Performance of $\mathbb{CGPOPS}$ on Example 1 Using *hp*-I(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-I(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-I(3,10)	Number of Collocation Points
1	28.27	31	28.27	31
2	4.090	67	4.090	67
3	7.060×10^{-1}	101	7.060×10^{-1}	101
4	1.661×10^{-1}	134	1.661×10^{-1}	134
5	1.476×10^{-2}	158	1.476×10^{-2}	158
6	1.139×10^{-3}	191	1.139×10^{-3}	191
7	7.557×10^{-7}	218	7.557×10^{-7}	218

Table 2. Performance of $\mathbb{CGPOPS}$ on Example 1 Using *hp*-II(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-II(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-II(3,10)	Number of Collocation Points
1	28.27	31	28.27	31
2	1.667	65	1.667	65
3	3.193	106	3.193	106
4	1.557×10^{-1}	140	1.557×10^{-1}	140
5	4.142×10^{-1}	165	4.142×10^{-1}	165
6	1.261×10^{-2}	185	1.261×10^{-2}	185
7	4.423×10^{-2}	204	4.423×10^{-2}	204
8	4.707×10^{-4}	209	4.707×10^{-4}	209
9	1.090×10^{-3}	226	1.090×10^{-3}	226
10	7.742×10^{-6}	247	7.742×10^{-6}	247
11	7.470×10^{-7}	250	7.470×10^{-7}	250

refinement history. Finally, Tables 1 through 4 show the approximation of the error in the solution on each mesh, where it is seen that the error approximation decreases with each mesh refinement iteration using any of the *hp* methods.

5.2 Example 2: Reusable Launch Vehicle Entry

Consider the following optimal control problem taken from Ref. [11] where the objective is to maximize the crossrange during the atmospheric entry of a reusable launch vehicle (where the numerical values in Ref. [11] have been converted from English units to SI units). Maximize the objective functional

$$\mathcal{J} = \phi(t_f), \tag{49}$$

subject to the dynamic constraints

$$\begin{aligned}
\dot{r} &= v\sin\gamma, & \dot{\theta} &= \frac{v\cos\gamma\sin\psi}{r\cos\phi},\\
\dot{\phi} &= \frac{v\cos\gamma\cos\psi}{r}, & \dot{v} &= -\frac{D}{m} - g\sin\gamma,\\
\dot{\gamma} &= \frac{L\cos\sigma}{mv} - \left(\frac{g}{v} - \frac{v}{r}\right)\cos\gamma, & \dot{\psi} &= \frac{L\sin\sigma}{mv\cos\gamma} + \frac{v\cos\gamma\sin\psi\tan\phi}{r},
\end{aligned} \tag{50}$$

Table 3. Performance of ℂ𝔾ℙ𝕆ℙ𝕊 on Example 1 Using *hp*-III(3,10)

Mesh Iteration Number	Estimated Error (ℂ𝔾ℙ𝕆ℙ𝕊) *hp*-III(3,10)	Number of Collocation Points	Estimated Error (𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀) *hp*-III(3,10)	Number of Collocation Points
1	28.27	31	28.27	31
2	5.207	22	5.207	22
3	5.848×10^{-1}	112	5.848×10^{-1}	112
4	9.156×10^{-2}	139	9.156×10^{-2}	142
5	5.732×10^{-3}	115	5.732×10^{-3}	112
6	9.927×10^{-5}	146	9.927×10^{-5}	146
7	2.451×10^{-5}	153	2.451×10^{-5}	153
8	8.237×10^{-7}	160	8.237×10^{-7}	160

Table 4. Performance of ℂ𝔾ℙ𝕆ℙ𝕊 on Example 1 Using *hp*-IV(3,10)

Mesh Iteration Number	Estimated Error (ℂ𝔾ℙ𝕆ℙ𝕊) *hp*-IV(3,10)	Number of Collocation Points	Estimated Error (𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀) *hp*-IV(3,10)	Number of Collocation Points
1	28.27	31	28.27	31
2	4.763	46	4.763	46
3	8.214×10^{-1}	52	8.214×10^{-1}	55
4	1.813×10^{-1}	55	1.813×10^{-1}	58
5	2.114×10^{-2}	61	2.114×10^{-2}	61
6	1.688×10^{-3}	87	1.688×10^{-3}	87
7	8.991×10^{-7}	106	8.991×10^{-7}	106

and the boundary conditions

$$\begin{array}{rclrcl} h(0) & = & 79248 \text{ km}, & h(t_f) & = & 24384 \text{ km}, \\ \theta(0) & = & 0 \text{ deg}, & \theta(t_f) & = & \text{Free}, \\ \phi(0) & = & 0 \text{ deg}, & \phi(t_f) & = & \text{Free}, \\ v(0) & = & 7.803 \text{ km/s}, & v(t_f) & = & 0.762 \text{ km/s}, \\ \gamma(0) & = & -1 \text{ deg}, & \gamma(t_f) & = & -5 \text{ deg}, \\ \psi(0) & = & 90 \text{ deg}, & \psi(t_f) & = & \text{Free}, \end{array} \tag{51}$$

where $r = h + R_e$ is the geocentric radius, h is the altitude, R_e is the polar radius of the Earth, θ is the longitude, ϕ is the latitude, v is the speed, γ is the flight path angle, ψ is the azimuth angle, and m is the mass of the vehicle. Furthermore, the aerodynamic and gravitational forces are computed as

$$D = \rho v^2 S C_D/2, \quad L = \rho v^2 S C_L/2, \quad g = \mu/r^2, \tag{52}$$

where $\rho = \rho_0 \exp(-h/H)$ is the atmospheric density, ρ_0 is the density at sea level, H is the density scale height, S is the vehicle reference area, C_D is the coefficient of drag, C_L is the coefficient of lift, and μ is the gravitational parameter.

The optimal control problem given in Equations (49) through (52) was solved using ℂ𝔾ℙ𝕆ℙ𝕊 with the *hp*-I(4,10), *hp*-II(4,10), *hp*-III(4,10), and *hp*-IV(4,10) methods on an initial mesh consisting of 10 evenly spaced mesh intervals with four LGR points per mesh interval. The NLP solver and mesh refinement accuracy tolerances were both set to 10^{-7}. The initial guess of the state was a

Table 5. Performance of $\mathbb{CGPOPS}$ on Example 2 Using *hp*-I(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-I(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-I(4,10)	Number of Collocation Points
1	2.463×10^{-3}	41	2.463×10^{-3}	41
2	9.891×10^{-5}	103	9.896×10^{-5}	103
3	3.559×10^{-6}	118	3.559×10^{-6}	118
4	3.287×10^{-7}	133	3.287×10^{-7}	133
5	8.706×10^{-8}	134	8.706×10^{-8}	134

Table 6. Performance of $\mathbb{CGPOPS}$ on Example 2 Using *hp*-II(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-II(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-II(4,10)	Number of Collocation Points
1	2.463×10^{-3}	41	2.463×10^{-3}	41
2	6.026×10^{-6}	193	6.023×10^{-6}	193
3	8.227×10^{-8}	261	8.227×10^{-8}	261

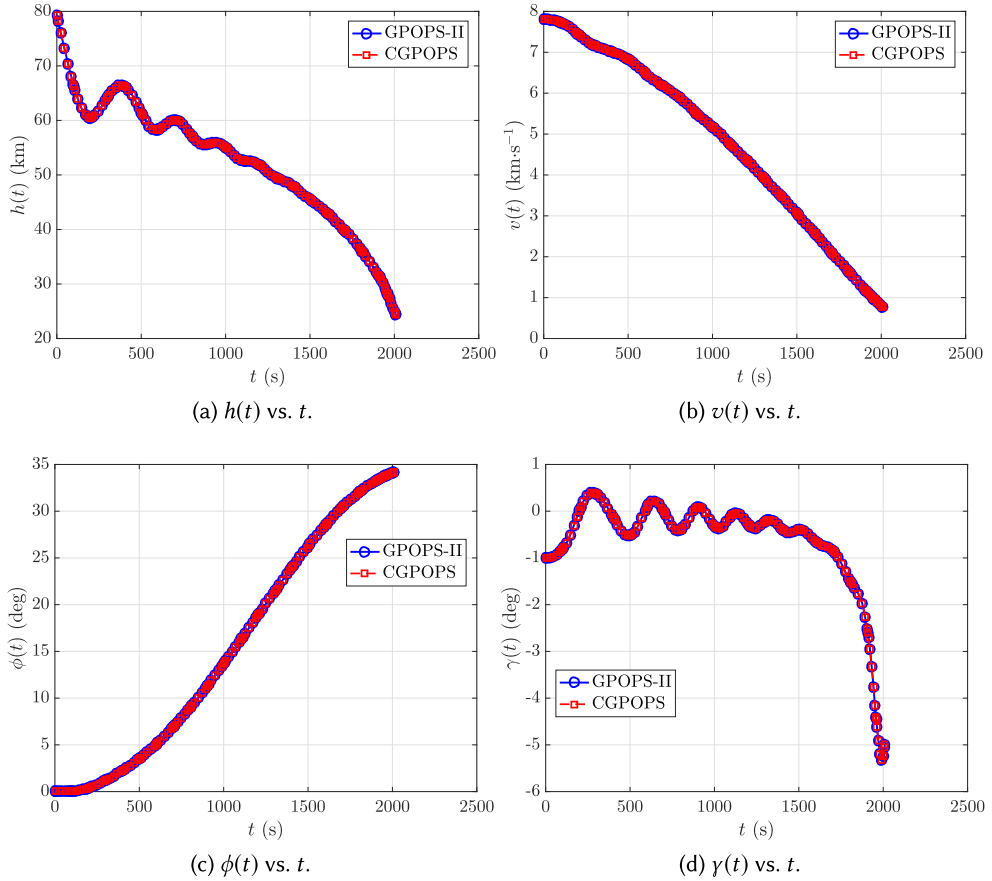
Table 7. Performance of $\mathbb{CGPOPS}$ on Example 2 Using *hp*-III(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-III(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-III(4,10)	Number of Collocation Points
1	2.463×10^{-3}	41	2.463×10^{-3}	41
2	2.850×10^{-5}	71	2.850×10^{-5}	71
3	2.065×10^{-6}	141	2.065×10^{-6}	141
4	8.887×10^{-8}	148	8.887×10^{-8}	148

Table 8. Performance of $\mathbb{CGPOPS}$ on Example 2 Using *hp*-IV(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-IV(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-IV(4,10)	Number of Collocation Points
1	2.463×10^{-3}	41	2.463×10^{-3}	41
2	2.364×10^{-5}	122	3.364×10^{-5}	122
3	3.286×10^{-7}	200	3.286×10^{-7}	192
4	9.561×10^{-8}	203	1.285×10^{-7}	194
5	—	—	9.561×10^{-8}	195

straight line over the duration $t \in [0, 1,000]$ between the known initial and final components of the state or a constant at the initial values of the components of the state whose terminal values are not specified, whereas the initial guess of both controls was zero. Tables 5 through 8 show the performance of both $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ on this example for the four *hp* methods, where the mesh refinement history is nearly identical using any of the *hp* methods. The solution obtained using $\mathbb{CGPOPS}$ with the *hp*-III(4,10) method is shown in Figures 7(a) through 8(b) alongside the



(a) $h(t)$ vs. t. (b) $v(t)$ vs. t.

(c) $\phi(t)$ vs. t. (d) $\gamma(t)$ vs. t.

Fig. 7. CGPOPS and GPOPS – II state solutions to Example 2 using *hp*-III(4,10).

solution obtained using the software GPOPS – II [52] with the *hp*-III(4,10) method, where it is seen that the two solutions obtained are essentially identical. Moreover, the optimal objective obtained using both CGPOPS and GPOPS – II was 0.59627639 to eight significant figures. Furthermore, Figure 9 shows the identical mesh refinement history of the two best performing methods. Finally, the computation time used by CGPOPS is approximately half the amount of time required by GPOPS – II to solve the optimal control problem, taking 0.9105 and 1.9323 seconds, respectively.

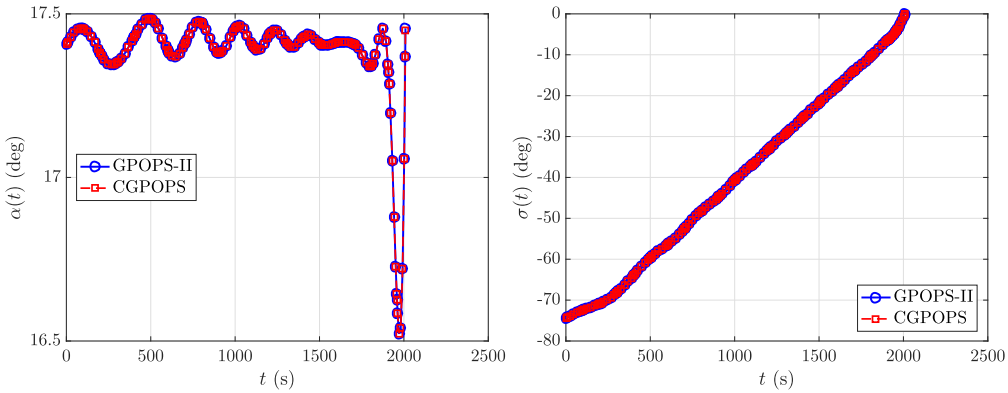
5.3 Example 3: Space Station Attitude Control

Consider the following space station attitude control optimal control problem taken from Refs. [11, 53]. Minimize the cost functional

$$\mathcal{J} = \frac{1}{2}\int_{t_0}^{t_f} \mathbf{u}^{\mathsf{T}}\mathbf{u}dt, \tag{53}$$

subject to the dynamic constraints

$$\begin{aligned} \dot{\boldsymbol{\omega}} &= \mathbf{J}^{-1}\left\{\boldsymbol{\tau}_{gg}(\mathbf{r}) - \boldsymbol{\omega}^{\otimes}\left[\mathbf{J}\boldsymbol{\omega} + \mathbf{h}\right] - \mathbf{u}\right\}, \\ \dot{\mathbf{r}} &= \frac{1}{2}\left[\mathbf{r}\mathbf{r}^{\mathsf{T}} + \mathbf{I} + \mathbf{r}^{\otimes}\right]\left[\boldsymbol{\omega} - \boldsymbol{\omega}_0(\mathbf{r})\right], \\ \dot{\mathbf{h}} &= \mathbf{u}, \end{aligned} \tag{54}$$



(a) $\alpha(t)$ vs. t. (b) $\sigma(t)$ vs. t.

Fig. 8. $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ control solutions to Example 2 using *hp*-III(4,10).

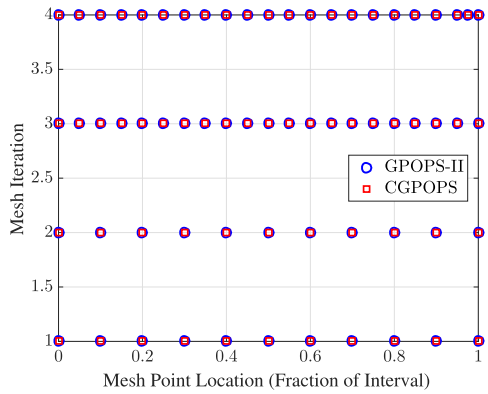


Fig. 9. $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ mesh refinement history for Example 2 using *hp*-III(4,10).

the inequality path constraint

$$\|\mathbf{h}\| \leq h_{\max}, \tag{55}$$

and the boundary conditions

$$\begin{aligned} t_0 &= 0, \quad t_f = 1800, \\ \boldsymbol{\omega}(0) &= \bar{\boldsymbol{\omega}}_0, \ \mathbf{r}(0) = \bar{\mathbf{r}}_0, \ \mathbf{h}(0) = \bar{\mathbf{h}}_0, \\ \mathbf{0} &= \mathbf{J}^{-1}\left\{\boldsymbol{\tau}_{gg}(\mathbf{r}(t_f)) - \boldsymbol{\omega}(t_f)^{\otimes}\left[\mathbf{J}\boldsymbol{\omega}(t_f) + \mathbf{h}(t_f)\right]\right\}, \\ \mathbf{0} &= \frac{1}{2}\left[\mathbf{r}(t_f)\mathbf{r}^{\mathsf{T}}(t_f) + \mathbf{I} + \mathbf{r}(t_f)^{\otimes}\right]\left[\boldsymbol{\omega}(t_f) - \boldsymbol{\omega}_0(\mathbf{r}(t_f))\right], \end{aligned} \tag{56}$$

where $(\boldsymbol{\omega}, \mathbf{r}, \mathbf{h})$ is the state and $\mathbf{u}$ is the control. In this formulation, $\boldsymbol{\omega}$ is the angular velocity, $\mathbf{r}$ is the Euler-Rodrigues parameter vector, $\mathbf{h}$ is the angular momentum, and $\mathbf{u}$ is the input moment (and is the control). Furthermore,

$$\begin{aligned} \boldsymbol{\omega}_0(\mathbf{r}) &= -\omega_{\text{orb}}\mathbf{C}_2, \quad & \boldsymbol{\tau}_{gg} &= 3\omega_{\text{orb}}^2\mathbf{C}_3^{\otimes}\mathbf{J}\mathbf{C}_3, \\ \omega_{\text{orb}} &= 0.6511\frac{\pi}{180}, \quad & h_{\max} &= 10000, \end{aligned} \tag{57}$$

Table 9. Performance of $\mathbb{CGPOPS}$ on Example 3 Using *hp*-I(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-I(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-I(4,10)	Number of Collocation Points
1	9.409×10^{-6}	41	9.409×10^{-6}	41
2	6.496×10^{-7}	47	6.496×10^{-7}	47

Table 10. Performance of $\mathbb{CGPOPS}$ on Example 3 Using *hp*-II(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-II(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-II(4,10)	Number of Collocation Points
1	9.409×10^{-6}	41	9.409×10^{-6}	41
2	2.389×10^{-6}	50	2.387×10^{-6}	50
3	7.125×10^{-7}	55	7.130×10^{-7}	55

Table 11. Performance of $\mathbb{CGPOPS}$ on Example 3 Using *hp*-III(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-III(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-III(4,10)	Number of Collocation Points
1	9.409×10^{-6}	41	9.409×10^{-6}	41
2	9.542×10^{-7}	50	9.559×10^{-7}	50

Table 12. Performance of $\mathbb{CGPOPS}$ on Example 3 Using *hp*-IV(4,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-IV(4,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-IV(4,10)	Number of Collocation Points
1	9.409×10^{-6}	41	9.409×10^{-6}	41
2	1.049×10^{-7}	53	1.046×10^{-7}	53
3	7.125×10^{-7}	57	7.130×10^{-7}	57

and $\mathbf{C}_2$ and $\mathbf{C}_3$ are the second and third column, respectively, of the matrix

$$\mathbf{C} = \mathbf{I} + \frac{2}{1 + \mathbf{r}^{\mathsf{T}}\mathbf{r}} \left(\mathbf{r}^{\otimes}\mathbf{r}^{\otimes} - \mathbf{r}^{\otimes} \right). \tag{58}$$

In this example, the matrix $\mathbf{J}$ is given as

$$\mathbf{J} = \begin{bmatrix} 2.80701911616 \times 10^{7} & 4.822509936 \times 10^{5} & -1.71675094448 \times 10^{7} \\ 4.822509936 \times 10^{5} & 9.5144639344 \times 10^{7} & 6.02604448 \times 10^{4} \\ -1.71675094448 \times 10^{7} & 6.02604448 \times 10^{4} & 7.6594401336 \times 10^{7} \end{bmatrix}, \tag{59}$$

whereas the initial conditions $\bar{\boldsymbol{\omega}}_0$, $\bar{\mathbf{r}}_0$, and $\bar{\mathbf{h}}_0$ are

$$\begin{aligned} \bar{\boldsymbol{\omega}}_0 &= \begin{bmatrix} -9.5380685844896 \times 10^{-6} \\ -1.1363312657036 \times 10^{-3} \\ +5.3472801108427 \times 10^{-6} \end{bmatrix}, \\ \bar{\mathbf{r}}_0 &= \begin{bmatrix} 2.9963689649816 \times 10^{-3} \\ 1.5334477761054 \times 10^{-1} \\ 3.8359805613992 \times 10^{-3} \end{bmatrix}, \quad \bar{\mathbf{h}}_0 = \begin{bmatrix} 5000 \\ 5000 \\ 5000 \end{bmatrix}. \end{aligned} \tag{60}$$

A more detailed description of this problem, including all of the constants $\mathbf{J}$, $\bar{\boldsymbol{\omega}}_0$, $\bar{\mathbf{r}}_0$, and $\bar{\mathbf{h}}_0$, can be found in Ref. [53] or [11].

The optimal control problem given in Equations (53) through (60) was solved using $\mathbb{CGPOPS}$ with the *hp*-I(4,10), *hp*-II(4,10), *hp*-III(4,10), and *hp*-IV(4,10) methods on an initial mesh consisting of 10 uniformly spaced mesh intervals and four LGR points per mesh interval. The NLP solver and mesh refinement accuracy tolerances were set to 10^{-7} and 10^{-6}, respectively. The initial guess was a constant over the time interval $t \in [0, 1800]$, where the constant was $(\bar{\boldsymbol{\omega}}_0, \bar{\mathbf{r}}_0, \bar{\mathbf{h}}_0)$ for the state and zero for the control. The essentially identical mesh refinement histories for $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ are shown in Tables 9–12. The state and control solutions obtained using $\mathbb{CGPOPS}$ are respectively shown in Figures 10 and 11 alongside the solution obtained using the optimal control software $\mathbb{GPOPS}-\mathbb{II}$ [52] with the *hp*-I(4,10). It is seen that the $\mathbb{CGPOPS}$ solution matches extremely well with the $\mathbb{GPOPS}-\mathbb{II}$ solution. Moreover, the optimal objective obtained using both $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ was 3.5867511×10^{-6} to eight significant figures. Finally, the computation time required by $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ to solve the optimal control problem was 0.5338 and 2.7696 seconds, respectively.

5.4 Example 4: Free-Flying Robot Problem

Consider the following optimal control problem taken from Refs. [11, 56]. Minimize the objective functional

$$\mathcal{J} = \int_0^{t_f} (u_1 + u_2 + u_3 + u_4) dt, \tag{61}$$

subject to the dynamic constraints

$$\begin{aligned} \dot{x} &= v_x, & \dot{y} &= v_y, \\ \dot{v}_x &= (F_1 + F_2)\cos(\theta), & \dot{v}_y &= (F_1 + F_2)\sin(\theta), \\ \dot{\theta} &= \omega, & \dot{\omega} &= \alpha F_1 - \beta F_2, \end{aligned} \tag{62}$$

the control inequality constraints

$$0 \le u_i \le 1, \quad (i = 1, 2, 3, 4), \qquad F_i \le 1, \quad (i = 1, 2), \tag{63}$$

and the boundary conditions

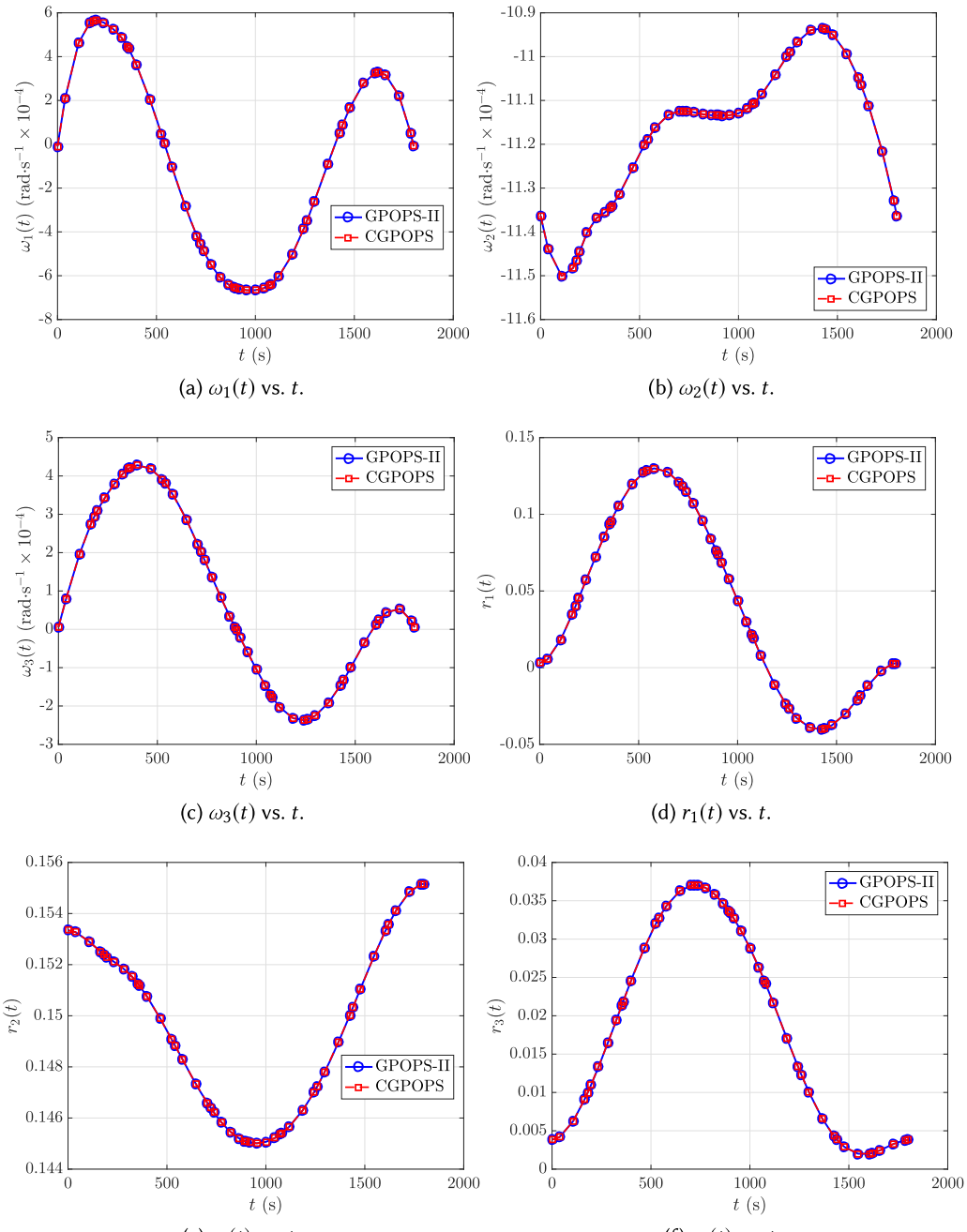
$$\begin{aligned} &x(0) = -10, \quad x(t_f) = 0, \quad y(0) = -10, \quad y(t_f) = 0, \\ &v_x(0) = 0, \quad v_x(t_f) = 0, \quad v_y(0) = 0, \quad v_y(t_f) = 0, \\ &\theta(0) = \tfrac{\pi}{2}, \quad \theta(t_f) = 0, \quad \omega(0) = 0, \quad \omega(t_f) = 0, \end{aligned} \tag{64}$$

where

$$F_1 = u_1 - u_2, \qquad F_2 = u_3 - u_4, \qquad \alpha = 0.2, \qquad \beta = 0.2. \tag{65}$$

It is known that the optimal control problem defined by Equations (61) through (65) is a bang-bang optimal control. Given the structure of the solution, the *hp*-BB(3,10) mesh refinement method [2] is also employed to solve this example.

The optimal control problem given in Equations (61) through (64) was solved using $\mathbb{CGPOPS}$ with the mesh refinement methods *hp*-I(3,10), *hp*-II(3,10), *hp*-III(3,10), *hp*-IV(3,10), and *hp*-BB(3,10)



(a) $\omega_1(t)$ vs. t.

(b) $\omega_2(t)$ vs. t.

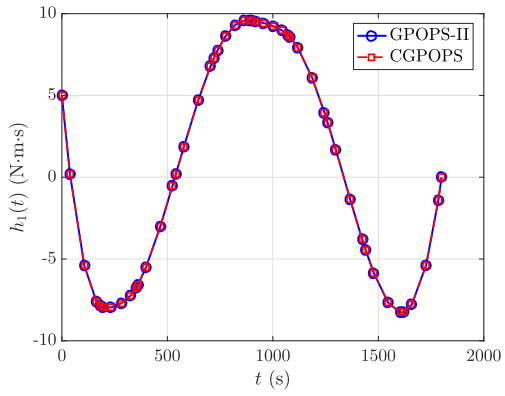
(c) $\omega_3(t)$ vs. t.

(d) $r_1(t)$ vs. t.

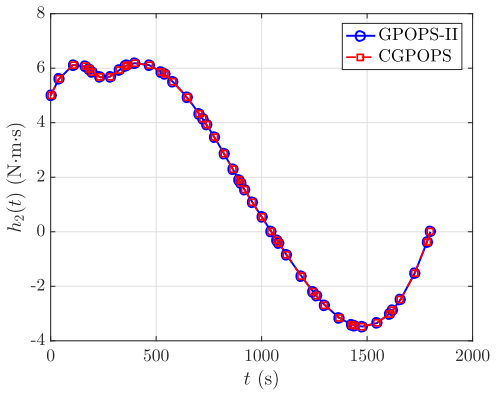
(e) $r_2(t)$ vs. t.

(f) $r_3(t)$ vs. t.

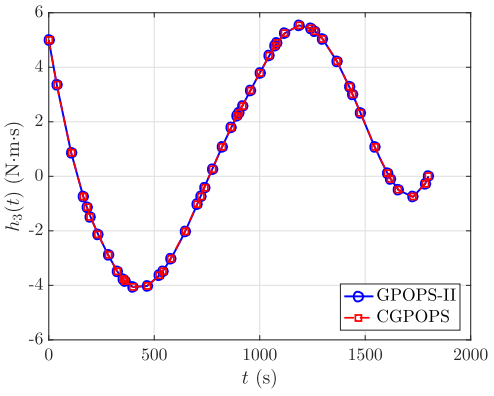
Fig. 10. ℂGℙOℙS and ℊℙOℙS – 𝕀𝕀 solutions to Example 3 using *hp*-I(4,10).



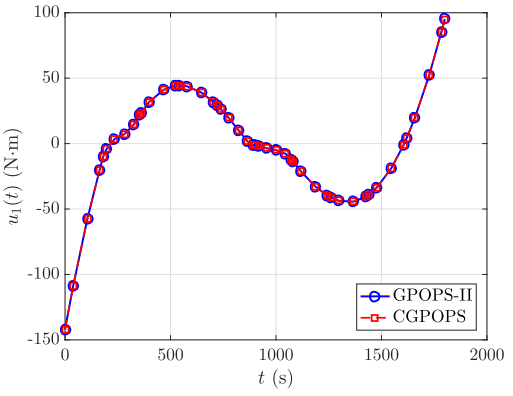
(a) $h_1(t)$ vs. t.



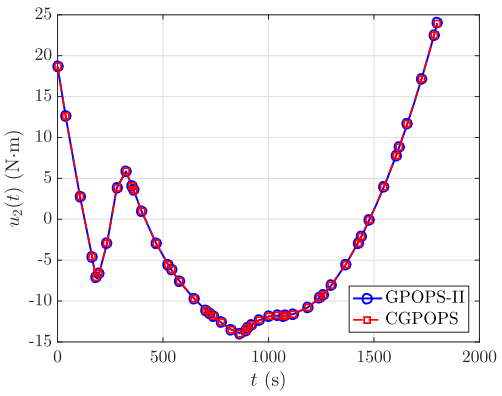
(b) $h_2(t)$ vs. t.



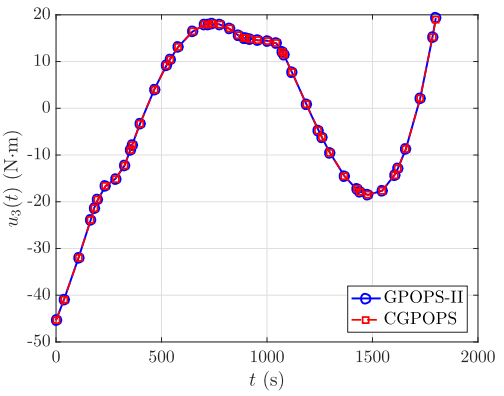
(c) $h_3(t)$ vs. t.



(d) $u_1(t)$ vs. t.



(e) $u_2(t)$ vs. t.



(f) $u_3(t)$ vs. t.

Fig. 11. $\mathbb{CGPOPS}$ and $\mathbb{GPOPS-II}$ solutions to Example 3 using hp-I(4,10).

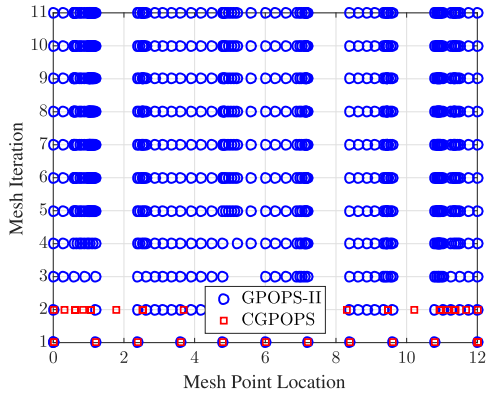


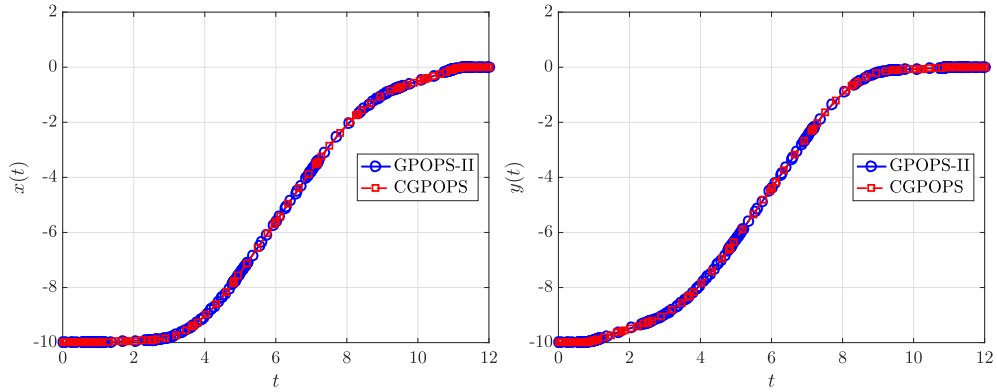
Fig. 12. ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 mesh refinement history for Example 4 using *hp*-BB(3,10) and *hp*-II(3,10), respectively.

on an initial mesh of 10 evenly spaced mesh intervals with five LGR points per mesh interval. Moreover, the NLP solver and mesh refinement accuracy tolerances were set to 10^{-9} and 10^{-7}, respectively. Figure 12 shows the mesh refinement history for ℂ𝔾ℙ𝕆ℙ𝕊 with the *hp*-BB(3,10) method and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 with the *hp*-II(3,10) method where ℂ𝔾ℙ𝕆ℙ𝕊 only requires a single mesh refinement iteration to attain the requested accuracy, whereas 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 takes nine mesh refinement iterations to attain that same accuracy. The solution obtained using ℂ𝔾ℙ𝕆ℙ𝕊 with the *hp*-BB(3,10) method is shown in Figures 13 and 14 alongside the solution obtained with 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 [52] with the *hp*-II(3,10) method. It is seen that the ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 solutions are in excellent agreement. Furthermore, the optimal objective obtained using ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 are 7.9101471 and 7.9101421, respectively, in agreement to six significant figures. Additionally, the computation time required by ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 to solve the optimal control problem was 0.6313 and 9.1826 seconds, respectively. To demonstrate how ℂ𝔾ℙ𝕆ℙ𝕊 is capable of accurately and efficiently capturing the bang-bang control profile of the optimal solution, Figure 14 shows the control solutions obtained using ℂ𝔾ℙ𝕆ℙ𝕊 employed with the *hp*-BB(3,10) mesh refinement method and 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 with the *hp*-II(3,10) mesh refinement method (where it is noted that the mesh refinement methods used were the most effective for that particular software program). It is seen that ℂ𝔾ℙ𝕆ℙ𝕊 accurately captures the switching times for all eight control discontinuities, whereas the solution obtained using 𝔾ℙ𝕆ℙ𝕊 – 𝕀𝕀 is less accurate near the discontinuities for the third and fourth control components (see Figure 14(c), (d), and (f)). Finally, Tables 13 through 17 show the estimated error on each mesh, where it is seen that the approximation of the solution error decreases with each mesh refinement iteration using any of the *hp* methods.

5.5 Example 5: Multiple-Stage Launch Vehicle Ascent Problem

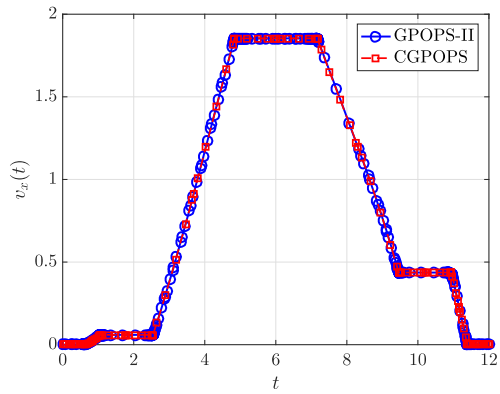
Consider the following four-phase optimal control problem where the objective is to steer a multiple-stage launch vehicle from the ground to the terminal orbit while maximizing the final mass of the vehicle [7, 11, 54]. The problem is modeled as a four-phase optimal control problem. Maximize the objective functional

$$\mathcal{J} = m\left(t_f^{(4)}\right), \tag{66}$$

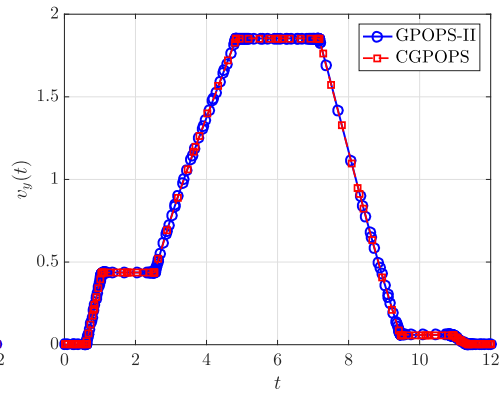


(a) $x(t)$ vs. t.

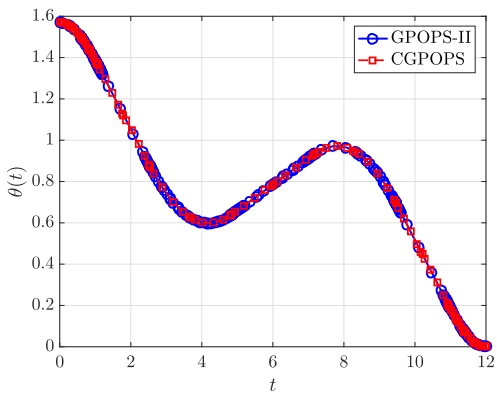
(b) $y(t)$ vs. t.



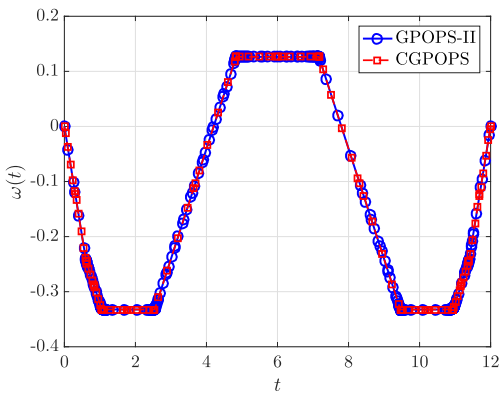
(c) $v_x(t)$ vs. t.



(d) $v_y(t)$ vs. t.

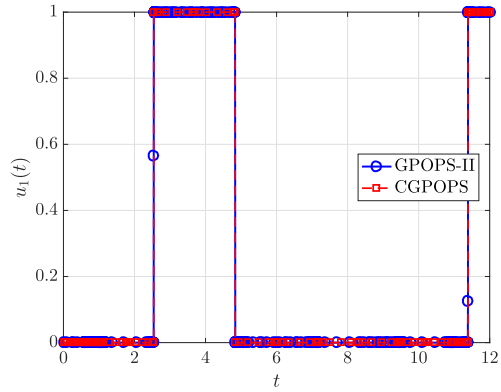


(e) $\theta(t)$ vs. t.

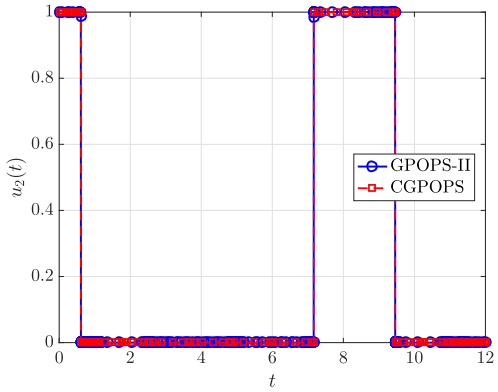


(f) $\omega(t)$ vs. t.

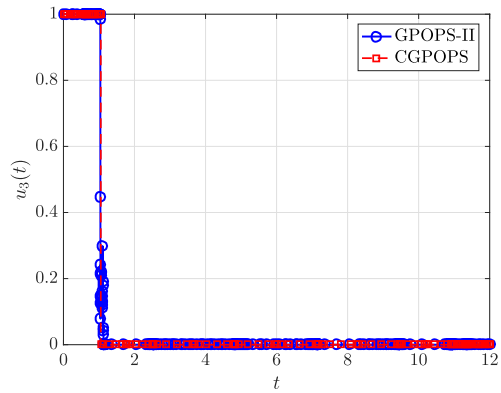
Fig. 13. $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$ state solutions to Example 4 using hp-BB(3,10) and hp-II(3,10), respectively.



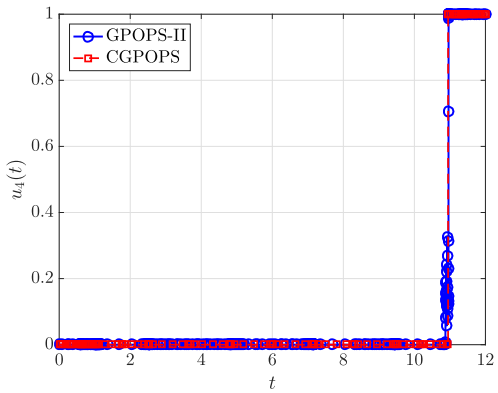
(a) $u_1(t)$ vs. t.



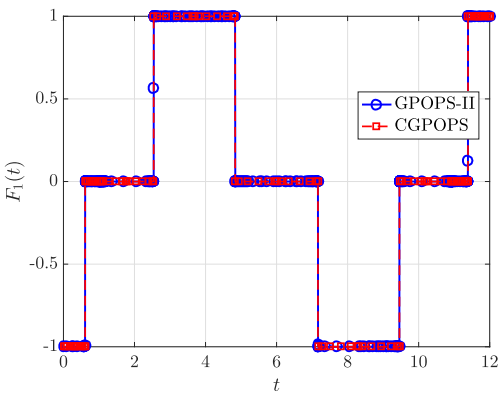
(b) $u_2(t)$ vs. t.



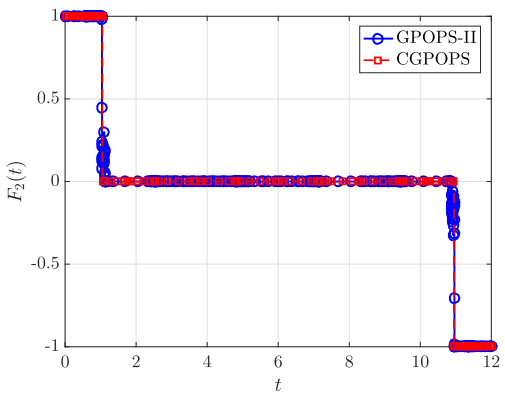
(c) $u_3(t)$ vs. t.



(d) $u_4(t)$ vs. t.



(e) $F_1(t)$ vs. t.



(f) $F_2(t)$ vs. t.

Fig. 14. CGPOPS and GPOPS – II control solutions to Example 4 using *hp*-BB(3,10) and *hp*-II(3,10), respectively.

Table 13. Performance of $\mathbb{CGPOPS}$ on Example 4 Using *hp*-I(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-I(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-I(3,10)	Number of Collocation Points
1	5.7636×10^{-4}	50	5.7636×10^{-4}	50
2	2.3428×10^{-4}	82	1.2977×10^{-4}	82
3	7.5065×10^{-5}	122	2.3256×10^{-4}	120
4	6.2091×10^{-5}	157	1.1175×10^{-5}	161
5	9.4236×10^{-6}	184	6.2093×10^{-5}	188
6	3.9835×10^{-6}	209	4.8405×10^{-6}	212
7	2.8105×10^{-6}	224	2.8104×10^{-6}	234
8	8.3276×10^{-7}	237	1.5139×10^{-6}	253
9	5.4493×10^{-7}	250	6.9960×10^{-7}	261
10	3.4339×10^{-7}	258	7.5178×10^{-7}	268
11	3.4145×10^{-7}	268	2.7108×10^{-7}	281
12	1.3458×10^{-7}	274	5.5799×10^{-7}	287
13	2.3812×10^{-7}	275	2.3815×10^{-7}	295
14	9.0332×10^{-8}	278	9.0299×10^{-8}	297

Table 14. Performance of $\mathbb{CGPOPS}$ on Example 4 Using *hp*-II(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-II(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-II(3,10)	Number of Collocation Points
1	5.7636×10^{-4}	50	5.7636×10^{-4}	50
2	2.3718×10^{-4}	98	1.1649×10^{-4}	98
3	6.4909×10^{-5}	162	9.3164×10^{-5}	146
4	2.1470×10^{-5}	219	1.1244×10^{-4}	207
5	9.3539×10^{-6}	263	3.2283×10^{-6}	267
6	1.0198×10^{-6}	297	3.5320×10^{-7}	302
7	1.7028×10^{-7}	310	2.3505×10^{-7}	320
8	9.8413×10^{-8}	315	1.3862×10^{-7}	322
9	—	—	1.0431×10^{-7}	325
10	—	—	9.5122×10^{-8}	328

subject to the dynamic constraints

$$\begin{aligned}\dot{\mathbf{r}}^{(p)} &= \mathbf{v}^{(p)},\\ \dot{\mathbf{v}}^{(p)} &= -\frac{\mu}{\|\mathbf{r}^{(p)}\|^3}\mathbf{r}^{(p)}+\frac{T^{(p)}}{m^{(p)}}\mathbf{u}^{(p)}+\frac{\mathbf{D}^{(p)}}{m^{(p)}}, \quad (p=1,2,3,4),\\ \dot{m}^{(p)} &= -\frac{T^{(p)}}{g_0 I_{sp}},\end{aligned} \tag{67}$$

the initial conditions

$$\begin{aligned}\mathbf{r}(t_0) &= \mathbf{r}_0 = (5605.2, 0, 3043.4)\times10^3 \text{ m},\\ \mathbf{v}(t_0) &= \mathbf{v}_0 = (0, 0.4076, 0)\times10^3 \text{ m/s},\\ m(t_0) &= m_0 = 301454 \text{ kg},\end{aligned} \tag{68}$$

Table 15. Performance of $\mathbb{CGPOPS}$ on Example 4 Using *hp*-III(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-III(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-III(3,10)	Number of Collocation Points
1	5.7636×10^{-4}	50	5.7636×10^{-4}	50
2	1.8489×10^{-4}	68	1.8489×10^{-4}	68
3	5.8497×10^{-5}	185	5.8497×10^{-5}	185
4	4.3708×10^{-6}	275	4.3709×10^{-6}	264
5	8.2894×10^{-7}	349	2.3747×10^{-6}	324
6	4.5337×10^{-7}	395	2.4780×10^{-7}	389
7	8.1069×10^{-8}	460	1.5231×10^{-7}	410
8	—	—	1.0142×10^{-7}	436
9	—	—	2.1817×10^{-7}	437
10	—	—	8.0985×10^{-8}	458

Table 16. Performance of $\mathbb{CGPOPS}$ on Example 4 Using *hp*-IV(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-IV(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-IV(3,10)	Number of Collocation Points
1	5.7636×10^{-4}	50	5.7636×10^{-4}	50
2	7.2614×10^{-5}	100	7.2614×10^{-5}	100
3	5.8350×10^{-5}	163	5.8350×10^{-5}	163
4	7.0276×10^{-6}	212	3.9712×10^{-6}	203
5	2.9097×10^{-6}	259	1.9372×10^{-6}	249
6	5.0338×10^{-7}	317	7.0224×10^{-6}	301
7	2.1987×10^{-7}	362	1.1880×10^{-6}	328
8	9.8979×10^{-8}	376	7.4092×10^{-7}	347
9	—	—	1.9947×10^{-7}	360
10	—	—	9.1526×10^{-8}	373

the interior point constraints

$$\begin{aligned} \mathbf{r}^{(p)}\left(t_f^{(p)}\right) - \mathbf{r}^{(p+1)}\left(t_0^{(p+1)}\right) &= \mathbf{0}, \\ \mathbf{v}^{(p)}\left(t_f^{(p)}\right) - \mathbf{v}^{(p+1)}\left(t_0^{(p+1)}\right) &= \mathbf{0}, \quad (p = 1, 2, 3), \\ m^{(p)}\left(t_f^{(p)}\right) - m_{\text{dry}}^{(p)} - m^{(p+1)}\left(t_0^{(p+1)}\right) &= 0, \end{aligned} \tag{69}$$

the terminal constraints (corresponding to a geosynchronous transfer orbit)

$$\begin{aligned} a\left(t_f^{(4)}\right) &= a_f = 24361.14 \text{ km}, \quad e\left(t_f^{(4)}\right) = e_f = 0.7308, \\ i\left(t_f^{(4)}\right) &= i_f = 28.5 \text{ deg}, \quad \theta\left(t_f^{(4)}\right) = \theta_f = 269.8 \text{ deg}, \\ \phi\left(t_f^{(4)}\right) &= \phi_f = 130.5 \text{ deg}, \end{aligned} \tag{70}$$

and the path constraints

$$\begin{aligned} |\mathbf{r}^{(p)}|_2 &\geq R_e, \\ \|\mathbf{u}^{(p)}\|_2^2 &= 1, \end{aligned} \quad (p = 1, \ldots, 4). \tag{71}$$

Table 17. Performance of $\mathbb{CGPOPS}$ on Example 4 Using *hp*-BB(3,10)

Mesh Iteration Number	Estimated Error ($\mathbb{CGPOPS}$) *hp*-BB(3,10)	Number of Collocation Points	Estimated Error ($\mathbb{GPOPS}-\mathbb{II}$) *hp*-II(3,10)	Number of Collocation Points
1	5.7636×10^{-4}	50	5.7636×10^{-4}	50
2	6.2675×10^{-9}	108	1.1649×10^{-4}	98
3	—	—	9.3164×10^{-5}	146
4	—	—	1.1244×10^{-4}	207
5	—	—	3.2283×10^{-6}	267
6	—	—	3.5320×10^{-7}	302
7	—	—	2.3505×10^{-7}	320
8	—	—	1.3862×10^{-7}	322
9	—	—	1.0431×10^{-7}	325
10	—	—	9.5122×10^{-8}	328

Table 18. Vehicle Properties for the Multiple-Stage Launch Vehicle Ascent Problem

Quantity	Solid Boosters	Stage 1	Stage 2
m_{tot} (kg)	19290	104380	19300
m_{prop} (kg)	17010	95550	16820
T (N)	628500	1083100	110094
I_{sp} (s)	283.3	301.7	467.2
Number of Engines	9	1	1
Burn Time (s)	75.2	261	700

In each phase, the quantities $\mathbf{r} = (x, y, z)$ and $\mathbf{v} = (v_x, v_y, v_z)$ respectively represent the geocentric position measured relative to an inertial reference frame and the inertial velocity measured in Earth-centered inertial (ECI) coordinates, μ is the gravitational parameter, T is the vacuum thrust, m is the vehicle mass, g_0 is the acceleration due to gravity at sea level, I_{sp} is the specific impulse of the engine, $\mathbf{u} = (u_x, u_y, u_z)$ is the thrust direction (and is the control), and $\mathbf{D} = (D_x, D_y, D_z)$ is the drag force. It is noted that the drag force is given as

$$\mathbf{D} = -\tfrac{1}{2} C_D S \rho \|\mathbf{v}_{\text{rel}}\| \mathbf{v}_{\text{rel}}, \tag{72}$$

where C_D is the drag coefficient, S is the vehicle reference area, $\rho = \rho_0 \exp(-h/H)$ is the atmospheric density, ρ_0 is the sea level density, $h = r - R_e$ is the altitude, $r = \|\mathbf{r}\|_2 = \sqrt{x^2 + y^2 + z^2}$ is the geocentric radius, R_e is the equatorial radius of the Earth, H is the density scale height, $\mathbf{v}_{\text{rel}} = \mathbf{v} - \boldsymbol{\omega} \times \mathbf{r}$ is the velocity as viewed by an observer fixed to the Earth expressed in ECI coordinates, and $\boldsymbol{\omega} = (0, 0, \Omega)$ is the angular velocity of the Earth as viewed by an observer in the inertial reference frame expressed in ECI coordinates. Furthermore, m_{dry} is the dry mass of phases 1, 2, and 3 and is defined as $m_{\text{dry}} = m_{\text{tot}} - m_{\text{prop}}$, where m_{tot} and m_{prop} are respectively the total mass and propellant mass of phases 1, 2, and 3. Finally, the quantities a, e, i, θ, and ϕ are respectively the semi-major axis, eccentricity, inclination, longitude of ascending node, and argument of periapsis. The vehicle data for this problem and the numerical values for the physical constants can be found in Tables 18 and 19, respectively.

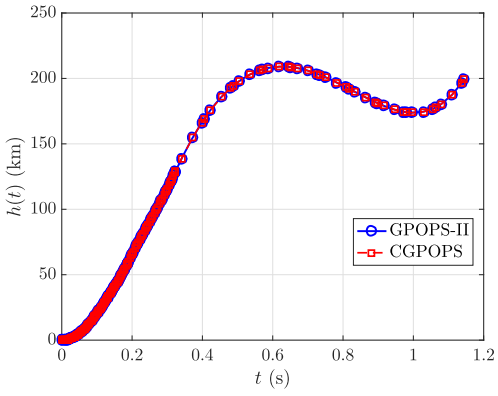
Table 19. Constants Used in the Multiple-Stage Launch Vehicle Ascent Problem

Constant	Value
Payload Mass	4164 kg
S	4π m^2
C_D	0.5
ρ_0	1.225 kg/m^3
H	7200 m
t_1	75.2 s
t_2	150.4 s
t_3	261 s
R_e	6378145 m
Ω	$7.29211585 \times 10^{-5}$ rad/s
μ	3.986012×10^{14} m^3/s^2
g_0	9.80665 m/s^2

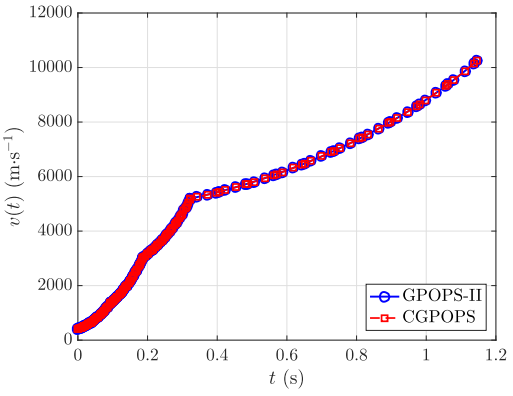
The multiple-stage launch vehicle ascent optimal control problem was solved using ℂ𝔾ℙ𝕆ℙ𝕊 with an initial mesh in each phase consisting of 10 uniformly spaced mesh intervals with four LGR points per mesh interval. The NLP solver and mesh refinement accuracy tolerances were set to 10^{-7} and 10^{-6}, respectively. The initial guess of the solution was constructed such that the initial guess of the position and the velocity in phases 1 and 2 was constant at $(\mathbf{r}(0), \mathbf{v}(0))$ as given in Equation (68), whereas in phases 3 and 4, the initial guess of the position and velocity was constant at $(\tilde{\mathbf{r}}, \tilde{\mathbf{v}})$, where $(\tilde{\mathbf{r}}, \tilde{\mathbf{v}})$ are obtained via a transformation from orbital elements to ECI coordinates using the five known orbital elements of Equation (70) and a true anomaly of zero. Furthermore, in all phases, the initial guess of the mass was a straight line between the initial and final mass, $m(t_0^{(p)})$ and $m(t_f^{(p)})$ $(p \in [1, \ldots, 4])$. Finally, in all phases, the guess of the control was constant at $\mathbf{u} = (0, 1, 0)$. The ℂ𝔾ℙ𝕆ℙ𝕊 solution is shown in Figure 15. In this example, the mesh refinement accuracy tolerance of 10^{-6} is satisfied on the initial mesh using both ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀, so no mesh refinement is necessary. The solution obtained using ℂ𝔾ℙ𝕆ℙ𝕊 matches closely with the solution obtained using the software 𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀 [52], where it is noted that the optimal objective obtained using ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀 are 7547.9729 and 7547.9739, respectively, agreeing to six significant figures. Finally, the computation time required by ℂ𝔾ℙ𝕆ℙ𝕊 and 𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀 to solve the optimal control problem was 2.9466 and 18.9401 seconds, respectively.

6 CAPABILITIES OF ℂ𝔾ℙ𝕆ℙ𝕊

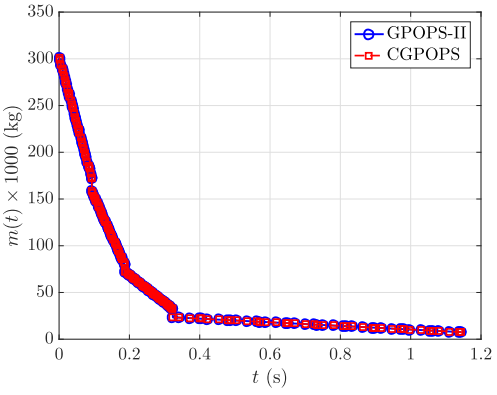
The five examples provided in Section 5 demonstrate the various capabilities of ℂ𝔾ℙ𝕆ℙ𝕊. First, the capabilities of the *hp* mesh refinement methods were demonstrated on the hyper-sensitive problem where the mesh was refined in the segments where the solution changed rapidly. Additionally, the ability of the *hp* methods to maintain a small mesh while satisfying a specified accuracy tolerance was shown in the reusable launch vehicle entry problem. Second, the flexibility of the software to achieve better performance by modifying the default settings of the mesh initialization and/or refinement process is demonstrated by the space station attitude control problem and free-flying robot problem. Third, all five examples demonstrate the increased computational efficiency of implementing the optimal control framework developed in Sections 2 and 3 in C++ as compared with the previous MATLAB software 𝔾ℙ𝕆ℙ𝕊 − 𝕀𝕀. In particular, the space station



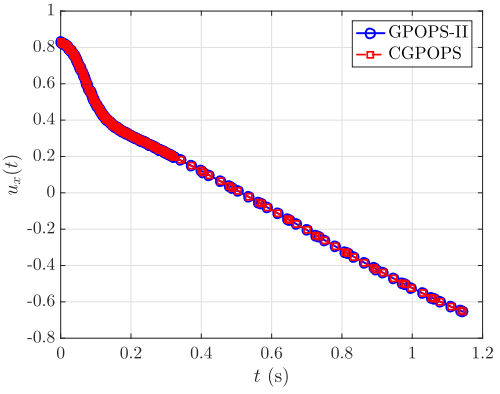
(a) $h(t)$ vs. t.



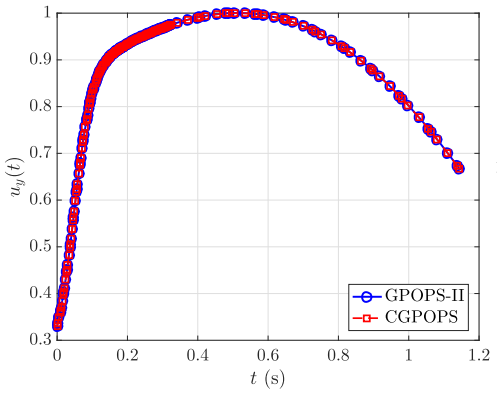
(b) $v(t)$ vs. t.



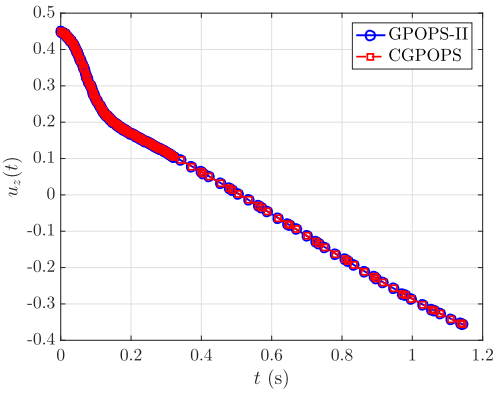
(c) $m(t)$ vs. t.



(d) $u_x(t)$ vs. t.



(e) $u_y(t)$ vs. t.



(f) $u_z(t)$ vs. t.

Fig. 15. Solution of Example 5 using $\mathbb{CGPOPS}$ and $\mathbb{GPOPS}-\mathbb{II}$.

attitude control example shows the computational benefits of using an exact NLP Lagrangian Hessian sparsity pattern (obtained by identifying the derivative dependencies using either hyper-dual or bicomplex-step derivative approximations as described in Section 4.5) as compared to the overestimated Hessian sparsity pattern employed in ℍℙOℙS − 𝕀𝕀. Next, because ℂGℙOℙS includes a newly developed mesh refinement method for problems whose solutions have a bang-bang structure, it is possible using ℂGℙOℙS to obtain an accurate solution to bang-bang optimal control problems much more efficiently than when using previously developed *hp* methods. In addition, the examples demonstrate the generality of the optimal control problem that can be formulated and solved using ℂGℙOℙS. The fact that ℂGℙOℙS is capable of solving the challenging benchmark optimal control problems shown in this article shows the general utility of the software on problems that may arise in different application areas.

7 LIMITATIONS OF ℂGℙOℙS

As with any software, ℂGℙOℙS has limitations. First, it is assumed that all functions used to formulate an optimal control problem of interest have continuous first and second derivatives. It is noted, however, that for some applications, the functions may have discontinuous derivatives while the functions themselves are continuous. In cases where the derivatives are discontinuous ℂGℙOℙS may have difficulty obtaining a solution because the NLP solver operates under the assumption that all first and second derivatives are continuous. Second, because ℂGℙOℙS is a direct collocation method, the ability to obtain a solution depends upon the NLP solver that is used. In particular, although the NLP solver IPOPT [12] used with ℂGℙOℙS may be effective for some examples, other NLP solvers (e.g., SNOPT [26] or KNITRO [13]) may be more effective than IPOPT for certain problems. Moreover, for problems with high-index path constraints, the constraint qualification conditions may not be satisfied when the mesh becomes extremely fine. In such cases, unique NLP Lagrange multipliers may not exist or, in some cases, these Lagrange multipliers may be unbounded. Furthermore, it may be difficult to obtain a solution to a poorly scaled problem. Finally, as is true for any optimal control software, optimal control problems whose solutions lie on a singular arc can create problems due to the inability to determine the optimal control along the singular arc. Moreover, the problems associated with a singular optimal control problem are exacerbated with mesh refinement. Thus, when solving a singular optimal control problem, it may be necessary to modify the original problem by including the higher-order optimality conditions that define the control on the singular arc.

8 CONCLUSION

A general-purpose C++ software program called ℂGℙOℙS has been described for solving multiple-phase optimal control problems using adaptive direct orthogonal collocation methods. In particular, the software employs an LGR quadrature orthogonal collocation where the continuous control problem is transcribed to a large sparse NLP. The software implements five previously developed adaptive mesh refinement methods that allow for flexibility in the number and placement of the collocation and mesh points to achieve a specified accuracy. In addition, the software is designed to compute all derivatives required by the NLP solver using one of four derivative estimation methods for the optimal control functions. The key components of the software have been described in detail, and the utility of the software is demonstrated on five benchmark optimal control problems. The software described in this article provides researchers a transitional platform upon which to solve a wide variety of complex constrained optimal control problems for real-time applications.

REFERENCES

[1] Milton Abramowitz and Irene Stegun. 1965. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, New York, NY.

[2] Yunus M. Agamawi, William W. Hager, and Anil V. Rao. 2019. Mesh refinement method for solving bang-bang optimal control problems using direct collocation. arXiv:1905.11895.

[3] Y. M. Agamawi and A. V. Rao. 2018. Exploiting sparsity in direct orthogonal collocation methods for solving multiple-phase optimal control problems. In *Proceedings of the 2018 Space Flight Mechanics Meeting*. https://doi.org/10.2514/6.2018-0724

[4] Yunus M. Agamawi and Anil V. Rao. 2020. Comparison of derivative estimation methods in optimal control using direct collocation. *AIAA Journal* 58, 1 (Jan. 2020), 341–354. https://doi.org/10.2514/1.J058514

[5] I. Babuska and M. Suri. 1994. The p and hp version of the finite element method, basic principles and properties. *SIAM Review* 36 (1994), 578–632. https://doi.org/10.1137/1036141

[6] V. M. Becerra. 2009. $\mathcal{PSOPT}$ *Optimal Control Solver User Manual*. University of Reading. http://www.psopt.org.

[7] D. A. Benson. 2004. *A Gauss Pseudospectral Transcription for Optimal Control*. Ph.D. Dissertation. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA.

[8] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao. 2006. Direct trajectory optimization and costate estimation via an orthogonal collocation method. *Journal of Guidance, Control, and Dynamics* 29, 6 (Nov.–Dec. 2006), 1435–1440. https://doi.org/10.2514/1.20478

[9] J. T. Betts. 1990. Sparse Jacobian updates in the collocation method for optimal control problems. *Journal of Guidance, Control, and Dynamics* 13, 3 (May–June 1990), 409–415. https://doi.org/10.2514/3.25352

[10] J. T. Betts. 1998. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics* 21, 2 (March–April 1998), 193–207. https://doi.org/10.2514/2.4231

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

[12] L. T. Biegler and V. M. Zavala. 2008. Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide optimization. *Computers and Chemical Engineering* 33, 3 (March 2008), 575–582. https://doi.org/10.1016/j.compchemeng.2008.08.006

[13] Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. 2006. KNITRO: An integrated package for nonlinear optimization. In *Large Scale Nonlinear Optimization*. Springer, Boston, MA, 35–59.

[14] Christopher L. Darby, Divya Garg, and Anil V. Rao. 2011. Costate estimation using multiple-interval pseudospectral methods. *Journal of Spacecraft and Rockets* 48, 5 (Sept.–Oct. 2011), 856–866. https://doi.org/10.2514/1.A32040

[15] C. L. Darby, W. W. Hager, and A. V. Rao. 2011. An hp-adaptive pseudospectral method for solving optimal control problems. *Optimal Control Applications and Methods* 32, 4 (July–Aug. 2011), 476–502. https://doi.org/10.1002/oca.957

[16] Christopher L. Darby, W. W. Hager, and Anil V. Rao. 2011. Direct trajectory optimization using a variable low-order adaptive pseudospectral method. *Journal of Spacecraft and Rockets* 48, 3 (May–June 2011), 433–445. https://doi.org/10.2514/1.52136

[17] Wenhao Du, Wanchun Chen, Liang Yang, and William W. Hager. 2019. Bounds for integration matrices that arise in Gauss and Radau collocation. *Computational Optimization and Applications* 74, 1 (Sept. 2019), 259–273. https://doi.org/10.1007/s10589-019-00099-5

[18] Iain S. Duff. 2004. MA57—A code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software* 30, 2 (April–June 2004), 118–144. http://doi.acm.org/10.1145/992200.992202

[19] G. Elnagar, M. Kazemi, and M. Razzaghi. 1995. The pseudospectral Legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Control* 40, 10 (1995), 1793–1796. https://doi.org/10.1109/9.467672

[20] G. Elnagar and M. Razzaghi. 1998. A collocation-type method for linear quadratic optimal control problems. *Optimal Control Applications and Methods* 18, 3 (1998), 227–235. https://doi.org/10.1002/(SICI)1099-1514(199705/06)18:3⟨227::AID-OCA598⟩3.0.CO;2-A

[21] Paola Falugi, Eric Kerrigan, and Eugene van Wyk. 2010. *Imperial College London Optimal Control Software User Guide (ICLOCS)*. Department of Electrical Engineering, Imperial College London, London, UK.

[22] J. Fike and J. Alonso. 2011. The development of hyper-dual numbers for exact second-derivative calculations. In *Proceedings of the 49th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. https://doi.org/10.2514/6.2011-886

[23] D. Garg, W. W. Hager, and A. V. Rao. 2011. Pseudospectral methods for solving infinite-horizon optimal control problems. *Automatica* 47, 4 (April 2011), 829–837. https://doi.org/10.1016/j.automatica.2011.01.085

[24] D. Garg, M. A. Patterson, C. L. Darby, C. Francolin, G. T. Huntington, W. W. Hager, and A. V. Rao. 2011. Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems via a Radau pseudospectral method. *Computational Optimization and Applications* 49, 2 (June 2011), 335–358. https://doi.org/10.1007/s10589-00-09291-0

[25] D. Garg, M. A. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington. 2010. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica* 46, 11 (Nov. 2010), 1843–1851. https://doi.org/10.1016/j.automatica.2010.06.048

[26] P. E. Gill, W. Murray, and M. A. Saunders. 2002. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review* 47, 1 (Jan. 2002), 99–131. https://doi.org/10.1137/S0036144504446096

[27] P. E. Gill, W. Murray, and M. H. Wright. 1981. *Practical Optimization*. Academic Press, London, UK.

[28] C. J. Goh and K. L. Teo. 1988. Control parameterization: A unified approach to optimal control problems with general constraints. *Automatica* 24, 1 (Jan. 1988), 3–18. https://doi.org/10.1016/0005-1098(88)90003-9

[29] Q. Gong, F. Fahroo, and I. M. Ross. 2008. Spectral algorithm for pseudospectral methods in optimal control. *Journal of Guidance, Control, and Dynamics* 31, 3 (May–June 2008), 460–471. https://doi.org/10.2514/1.32908

[30] Q. Gong, I. M. Ross, W. Kang, and F. Fahroo. 2008. Connections between the covector mapping theorem and convergence of pseudospectral methods. *Computational Optimization and Applications* 41, 3 (Dec, 2008), 307–335. https://doi.org/10.1007/s10589-007-9102-4

[31] Andreas Griewank, David Juedes, and Jean Utke. 1996. Algorithm 755: ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software* 22, 2 (1996), 131–167. https://doi.org/10.1145/229473.229474

[32] Andreas Griewank and Andrea Walther. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (2nd ed.). SIAM Press, Philadelphia, PA.

[33] W. Gui and I. Babuska. 1986. The h, p, and hp versions of the finite element method in 1 dimension. Part I. The error analysis of the p version. *Numerische Mathematik* 49 (1986), 577–612. https://doi.org/10.1007/BF01389733

[34] W. Gui and I. Babuska. 1986. The h, p, and hp versions of the finite element method in 1 dimension. Part II. The error analysis of the h and $h-p$ versions. *Numerische Mathematik* 49 (1986), 613–657. https://doi.org/10.1007/BF01389734

[35] W. Gui and I. Babuska. 1986. The h, p, and hp versions of the finite element method in 1 dimension. Part III. The adaptive $h-p$ version. *Numerische Mathematik* 49 (1986), 659–683. https://doi.org/10.1007/BF01389734

[36] William W. Hager, Hongyan Hou, Subhashree Mohapatra, Anil V. Rao, and Xiang-Sheng Wang. 2019. Convergence rate for a Radau hp collocation method applied to constrained optimal control. *Computational Optimization and Applications* 74, 1 (Sept. 2019), 275–314. https://doi.org/10.1007/s10589-019-00100-1

[37] W. W. Hager, H. Hou, and A. V. Rao. 2016. Convergence rate for a Gauss collocation method applied to unconstrained optimal control. *Journal of Optimization Theory and Applications* 169, 3 (2016), 801–824. https://doi.org/10.1007/s10957-016-0929-7

[38] W. W. Hager, H. Hou, and A. V. Rao. 2017. Lebesgue constants arising in a class of collocation methods. *IMA Journal of Numerical Analysis* 13, 1 (Oct. 2017), 1884–1901. https://doi.org/10.1093/imanum/drw060

[39] W. W. Hager, J. Liu, S. Mohapatra, A. V. Rao, and X.-S. Wang. 2018. Convergence rate for a Gauss collocation method applied to constrained optimal control. *SIAM Journal on Control and Optimization* 56, 2 (2018), 1386–1411. https://doi.org/10.1137/16M1096761

[40] B. Houska, H. J. Ferreau, and M. Diehl. 2011. ACADO toolkit—An open-source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods* 32, 3 (May–June 2011), 298–312. https://doi.org/10.1002/oca.939

[41] Geoffrey T. Huntington, David A. Benson, and Anil V. Rao. 2007. Optimal configuration of tetrahedral spacecraft formations. *Journal of the Astronautical Sciences* 55, 2 (April–June 2007), 141–169. https://doi.org/10.1007/BF03256518

[42] Geoffrey T. Huntington and Anil V. Rao. 2008. Optimal reconfiguration of spacecraft formations using the Gauss pseudospectral method. *Journal of Guidance, Control, and Dynamics* 31, 3 (May–June 2008), 689–698. https://doi.org/10.2514/1.31083

[43] D. Jain and P. Tsiotras. 2008. Trajectory optimization using multiresolution techniques. *Journal of Guidance, Control, and Dynamics* 31, 5 (Sept.–Oct. 2008), 1424–1436. https://doi.org/10.2514/1.32220

[44] C. Jansch, K. H. Well, and K. Schnepper. 1994. *GESOP—Eine Software Umgebung Zur Simulation Und Optimierung*. Proceedings des SFB.

[45] S. Kameswaran and L. T. Biegler. 2008. Convergence rates for direct transcription of optimal control problems using collocation at Radau points. *Computational Optimization and Applications* 41, 1 (2008), 81–126. https://doi.org/10.1007/s10589--007--9098--9

[46] G. Lantoine, R. P. Russell, and T. Dargent. 2012. Using multicomplex variables for automatic computation of high-order derivatives. *ACM Transactions on Mathematical Software* 38 (April 2012), Article 16, 21 pages.

[47] F. Liu, W. W. Hager, and A. V. Rao. 2015. Adaptive mesh refinement for optimal control using nonsmoothness detection and mesh size reduction. *Journal of the Franklin Institute* 352, 10 (Oct. 2015), 4081–4106. https://doi.org/10.1016/j.jfranklin.2015.05.028

[48] F. Liu, W. W. Hager, and A. V. Rao. 2018. Adaptive mesh refinement for optimal control using decay rates of Legendre polynomial coefficients. *IEEE Transactions on Control System Technology* 26, 4 (2018), 1475–1483. https://doi.org/10.1109/TCST.2017.2702122

[49] J. R. Martins and J. T. Hwang. 2013. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal* 51, 11 (Sept. 2013), 2582–2599. https://doi.org/10.2514/1.J052184

[50] M. A. Patterson, W. W. Hager, and A. V. Rao. 2015. A ph mesh refinement method for optimal control. *Optimal Control Applications and Methods* 36, 4 (July–Aug. 2015), 398–421. https://doi.org/10.1002/oca.2114

[51] M. A. Patterson and A. V. Rao. 2012. Exploiting sparsity in direct collocation pseudospectral methods for solving continuous-time optimal control problems. *Journal of Spacecraft and Rockets* 49, 2 (March–April 2012), 364–377. https://doi.org/10.2514/1.A32071

[52] Michael A. Patterson and Anil V. Rao. 2014. $\mathbb{GPOPS}-\mathbb{II}$, 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* 41, 1 (Oct. 2014), Article 1, 37 pages. https://doi.org/10.1145/2558904

[53] J. A. Pietz. 2003. *Pseudospectral Collocation Methods for the Direct Transcription of Optimal Control Problems*. Master's Thesis. Rice University, Houston, TX.

[54] Anil V. Rao, David A. Benson, Christopher L. Darby, Camila Francolin, Michael A. Patterson, Ilyssa Sanders, and Geoffrey T. Huntington. 2010. Algorithm 902: GPOPS, a MATLAB software for solving multiple-phase optimal control problems using the Gauss pseudospectral method. *ACM Transactions on Mathematical Software* 37, 2 (April–June 2010), Article 22, 39 pages. https://doi.org/10.1145/1731022.1731032

[55] Anil V. Rao and Kenneth D. Mease. 2000. Eigenvector approximate dichotomic basis method for solving hypersensitive optimal control problems. *Optimal Control Applications and Methods* 21, 1 (Jan.–Feb. 2000), 1–19. https://doi.org/10.1002/(SICI)1099-1514(200001/02)21:1⟨1::AID-OCA646⟩3.0.CO;2-V

[56] Y. Sakawa. 1999. Trajectory planning of a free-flying robot by using the optimal control. *Optimal Control Applications and Methods* 20 (1999), 235–248. https://doi.org/10.1002/(SICI)1099-1514(199909/10)20:5⟨235::AID-OCA658⟩3.0.CO;2-I

[57] W. G. Vlases, S. W. Paris, R. M. Lajoie, M. J. Martens, and C. R. Hargraves. 1990. *Optimal Trajectories by Implicit Simulation*. Technical Report WRDC-TR-90-3056. Boeing Aerospace and Electronics, Wright-Patterson Air Force Base, OH.

[58] O. von Stryk. 2000. *User's Guide for DIRCOL (Version 2.1): A Direct Collocation Method for the Numerical Solution of Optimal Control Problems*. Technical University, Munich, Germany.

[59] Andrea Walther, Andreas Griewank, and Olaf Vogel. 2003. ADOL-C: Automatic differentiation using operator overloading in C++. *Proceedings in Applied Mathematics and Mechanics* 2, 1 (2003), 41–44. https://doi.org/10.1002/pamm.200310011

[60] M. J. Weinstein and A. V. Rao. 2016. A source transformation via operator overloading method for the automatic differentiation of mathematical functions in MATLAB. *ACM Transactions on Mathematical Software* 42, 1 (May 2016), Article 11, 44 pages. https://doi.org/10.1145/2699456

[61] M. J. Weinstein and A. V. Rao. 2017. Algorithm: ADiGator, a toolbox for the algorithmic differentiation of mathematical functions in MATLAB. *ACM Transactions on Mathematical Software* 44, 2 (Oct. 2017), Article 21, 25 pages. https://doi.org/10.1145/3104990

[62] Y. Zhao and P. Tsiotras. 2011. Density functions for mesh refinement in numerical optimal control. *Journal of Guidance, Control, and Dynamics* 34, 1 (Jan.–Feb. 2011), 271–277. https://doi.org/10.2514/1.45852

Received May 2019; revised November 2019; accepted March 2020