Adaptive Mesh Refinement and Error Estimation Method for Optimal Control Using Direct Collocation

George V. Haman III* and Anil V. Rao†

University of Florida
Gainesville, FL 32611

Abstract

An adaptive mesh refinement method for numerically solving optimal control problems is developed using Legendre-Gauss-Radau direct collocation. In regions of the solution where the desired accuracy tolerance has not been met, the mesh is refined by either increasing the degree of the approximating polynomial in a mesh interval or dividing a mesh interval into subintervals. In regions of the solution where the desired accuracy tolerance has been met, the mesh size may be reduced by either merging adjacent mesh intervals or decreasing the degree of the approximating polynomial in a mesh interval. Coupled with the mesh refinement method described in this paper is a newly developed relative error estimate that is based on the differences between solutions obtained from the collocation method and those obtained by solving initial-value and terminal-value problems in each mesh interval using an interpolated control obtained from the collocation method. Because the error estimate is based on explicit simulation, the solution obtained via collocation is in close agreement with the solution obtained via explicit simulation using the control on the final mesh, which ensures that the control is an accurate approximation of the true optimal control. The method is demonstrated on three examples from the open literature, and the results obtained show an improvement in final mesh size when compared against previously developed mesh refinement methods.

1 Introduction

Direct collocation methods for solving optimal control problems have become increasingly popular over the past few decades. In a direct collocation method, the time domain of the optimal control problem is partitioned into a *mesh*. Within each mesh interval, the state is approximated using a set of basis or trial functions (typically, these basis functions are polynomials), and constraints are enforced at a specific set of points called *collocation points*. The resulting approximation transforms the original continuous-time optimal control problem into a finite-dimensional *nonlinear programming problem* (NLP) [1], and the NLP is solved using well-developed software (for example, SNOPT [2] or IPOPT [3]).

Direct collocation methods have historically been formulated as either h methods or p methods. In an h method, the order of the state approximation is fixed in each interval. Accuracy using an h

*Ph.D. Candidate, Department of Mechanical and Aerospace Engineering. Email: georgehaman@ufl.edu.
†Professor, Department of Mechanical and Aerospace Engineering. Email: anilvrao@ufl.edu.

method is then increased by adjusting the number and placement of mesh points. In a p method, the order of the state approximation varies in each interval, where the number and placement of mesh points are fixed. Accuracy using a p method is then increased by adjusting the order of the approximation in an interval. Various h and p methods that employ direct collocation have been developed, which include methods that utilize a density function to distribute mesh points [4] or a differentiation matrix to identify potential nonsmoothness in the solution [5]. While both h and p direct collocation methods have been used extensively, both approaches have limitations, specifically when obtaining high-accuracy solutions. An h method may require an extremely large number of intervals, whereas a p method may require an unreasonably high-degree polynomial approximation. In order to significantly reduce the size of the finite-dimensional approximation and improve the computational efficiency of solving the NLP, hp methods have been developed, which were originally employed as finite element methods for solving partial differential equations [6–10]. In an hp method, the number of intervals, placement of mesh points, and degree of the state approximation in each interval can vary. Furthermore, hp methods have gained considerable attention due to their robustness and ability to outperform both h and p methods in terms of computational efficiency and mesh size.

In recent years, the class of hp direct *Gaussian quadrature collocation* methods have been developed. In an hp Gaussian quadrature collocation method, the state is approximated using a basis of polynomials (typically, Lagrange or Chebyshev polynomials) whose support points include Gaussian quadrature points, and collocation is then performed at the Gaussian quadrature points [11–16]. The most well-developed Gaussian quadrature collocation methods employ collocation at the *Legendre-Gauss* (LG) points [11–13], *Legendre-Gauss-Radau* (LGR) points [12–14], or *Legendre-Gauss-Lobatto* (LGL) points [15,16]. Under certain assumptions of smoothness in the solution, Gaussian quadrature collocation methods that employ collocation at the LG or LGR points converge at an exponential rate as a function of the degree of the approximating polynomial [17–20]. Motivated by the desire to improve computational efficiency and accuracy, this research develops a novel hp Gaussian quadrature collocation method for optimal control driven by a novel error estimate.

In order to develop a practical hp method, an appropriate mesh must be chosen. Such a mesh is not known a priori and must be determined through an iterative process known as *mesh refinement*, where a new mesh is generated based on an estimate of the solution error on the current mesh and the NLP is then re-solved on the new mesh. Various hp Gaussian quadrature mesh refinement methods have been developed [21–26]. These hp methods typically utilize h refinement in regions of the solution that exhibit either nonsmooth or rapidly changing behavior and employ p refinement

in regions where the solution is changing in a smooth or slow manner, where refinement decisions are often influenced by user-specified parameters. Darby et al. [21] develops an *hp* method where *h* refinement is performed in an interval if the maximum residual of the dynamics midway between the collocation points exceeds a user-specified threshold; otherwise, the polynomial degree is adjusted by a constant. In a similar manner, Darby et al. [22] performs *h* refinement in regions of the solution where the ratio of the maximum to mean curvature in the state solution exceeds a user-specified curvature threshold; otherwise, the polynomial degree is adjusted using the ratio of the maximum error and desired error tolerance. On the other hand, Patterson et al. [23] introduces a *p*-then-*h* strategy for mesh refinement, where *p* refinement is exhausted in a mesh interval (that is, some user-specified maximum polynomial degree is exceeded) before the interval is uniformly split. In this method, the polynomial degree is adjusted based on the known exponential convergence rate of a Gaussian quadrature collocation method. Within a mesh interval, Liu et al. [24] estimates the decay rate of the state to be the decay rate of the coefficients of a Legendre polynomial expansion of the state, and *h* refinement is performed in an interval if a user-specified decay rate threshold is not exceeded. Moreover, Liu et al. [25] determines nonsmoothness in the solution by examining local maxima in the magnitude of the second derivative of the state within mesh intervals on two meshes. On the other hand, Miller et al. [26] develops a method to detect discontinuities in the control solution by employing an edge detection scheme based on jump function approximations, which can then be paired with any previously developed method for refinement in smooth regions of the solution. It is noted that the *hp* mesh refinement methods in Darby et al. [21], Darby et al. [22], and Patterson et al. [23] do not allow for mesh size reduction. As a result, the mesh may become unnecessarily large, and the large mesh can decrease the computational efficiency of solving the NLP. On the other hand, the methods described in Liu et al. [24,25] allow for mesh size reduction, which make it possible to obtain a smaller mesh while maintaining the desired solution accuracy.

The contributions of this paper are as follows. The first contribution of this paper is a new mesh refinement method for both increasing and decreasing the size of the mesh. This new *p*-then-*h* mesh refinement strategy differs from the method of Patterson et al. [23] in that the *p* refinement step adopts a conservative version of that described in Darby et al. [22] to avoid adding collocation points unnecessarily. Furthermore, different from the method of Liu et al. [24], the method of this paper can perform both *h* and *p* refinement on every mesh refinement iteration. In addition, a new *h*-then-*p* strategy for mesh size reduction is developed in this paper that differs fundamentally from the *p*-then-*h* reduction approach developed in Liu et al. [24]. Specifically, Liu et al. [24] employs power series approximations and requires the polynomial degree in adjacent intervals to be the same, whereas the new method developed in this paper for merging adjacent intervals ensures

that the explicit simulation (time-marching) solution obtained over adjacent intervals is in close agreement with the solution obtained via collocation in each interval. Consequently, the mesh size reduction method developed in this paper does not require that the polynomial degree be the same in adjacent intervals, which leads to greater mesh size reduction compared with the method of Liu et al. [24]. Moreover, the method of this paper attempts to decrease the degree of the polynomial approximation in an interval based on the maximum error, desired error tolerance, and current degree of the polynomial approximation. It is noted for completeness that preliminary versions of the approach developed in this paper are given in Haman and Rao [27,28].

A second contribution of this paper is a newly developed error estimate of the solution on a given mesh. Previously developed error estimates analyze the violation in the discretized differential-algebraic constraints (that is, the dynamic and path constraints) within a mesh interval [1,21–23,29], where error estimates are often obtained at a predetermined set of points (usually a midpoint). Betts and Huffman [29] and Betts [1] develop an error estimate for the state at the interval mid- and endpoints using a higher-order integration step (that is, by step-halving) for various discretization schemes (for example, trapezoid, Hermite-Simpson, and Runge-Kutta). On the other hand, Darby et al. [21] develops an error estimate based on the difference between an approximation of the time derivative of the state and the right-hand side of the dynamics midway between the collocation points. In addition, Darby et al. [22] extends the error estimate of Darby et al. [21] to include the violation in path constraints. Furthermore, Patterson et al. [23] calculates an error estimate at an increased number of LGR points within a mesh interval based on the difference between the Lagrange polynomial approximation of the state and an LGR quadrature integration of the state dynamics. Unlike these previous error estimates, the error estimate developed in this paper is obtained as follows. First, the dynamics are simulated in forward time across the mesh interval using the state at the start of the mesh interval. Second, the dynamics are simulated in backward time across the mesh interval using the state at the terminus of the mesh interval. During both forward and backward explicit simulations within a mesh interval, the control obtained via collocation is interpolated using a Lagrange polynomial. Then, two relative error estimates are obtained for the state within a mesh interval. The first relative error estimate is the maximum difference between the Lagrange polynomial approximation of the state in the mesh interval and the state obtained via forward explicit simulation in the mesh interval. The second relative error estimate is the maximum difference between the Lagrange polynomial approximation of the state in the mesh interval and the state obtained via backward explicit simulation in the mesh interval. The relative error estimate utilized in the mesh refinement method is then taken to be the maximum of these forward and backward relative error estimates. Using this approach for estimating the error ensures that the

solutions obtained via collocation and explicit simulation schemes are in agreement with one other on the final mesh. As a result, it is unnecessary to re-validate the solution via explicit simulation after the NLP is solved on the final mesh (as is often done after mesh refinement is complete in order to verify that the solution obtained from the NLP agrees with that obtained via explicit simulation).

This paper is organized as follows. Section 3 introduces the continuous-time Bolza optimal control problem. Section 4 describes the discretization of the Bolza optimal control problem given in Section 3 using the Legendre-Gauss-Radau collocation method. Section 5 describes the new relative error estimate along with the new mesh refinement method. Section 6 demonstrates the method developed in Section 5 on three examples taken from the open literature, where the key features of the new method are highlighted. In addition, Section 6 provides a comparison of the performance of the method developed in this paper against previously developed mesh refinement methods. Finally, Section 8 provides conclusions on this research.

2 Notation and Conventions

In this paper, the following vector-matrix notation and conventions are used. First, vectors and matrices appear in bold face (for example, $\mathbf{z}$ and $\mathbf{Z}$); whereas, scalars appear in regular weight (for example, z and Z). Second, all vectors are treated as *row* vectors (for example, $\mathbf{z} \in \mathbb{R}^n$ is given as $\mathbf{z} \equiv [z_1, \ldots, z_n]$); whereas, $\mathbf{Z} \in \mathbb{R}^{m\times n}$ denotes a matrix containing m rows and n columns. Third, a double subscript attached to a matrix denotes the scalar element in the particular row and column of the matrix (for example, $\mathbf{Z}_{ij} \in \mathbb{R}$ is the scalar element Z_{ij} in row i and column j of matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$). On the other hand, a single subscript attached to a matrix denotes the particular row of the matrix (for example, $\mathbf{Z}_j \in \mathbb{R}^n$ is row j of matrix $\mathbf{Z} \in \mathbb{R}^{m\times n}$ and is given as $\mathbf{Z}_j \equiv [Z_{j1}, \ldots, Z_{jn}]$).

Next, the following function notation and conventions are used in this paper. First, $z(t) \in \mathbb{R}$ denotes a scalar function z of the independent variable t, where $z(t_i) \in \mathbb{R}$ denotes the function $z(t)$ evaluated at $t = t_i$. Second, $\mathbf{z}(t) \in \mathbb{R}^n$ denotes a vector function $\mathbf{z}$ of the independent variable t and is given as $\mathbf{z}(t) \equiv [z_1(t), \ldots, z_n(t)]$ because all vectors are row vectors. Furthermore, $\mathbf{z}(t_i) \in \mathbb{R}^n$ denotes the function $\mathbf{z}(t)$ evaluated at $t = t_i$ and is given as $\mathbf{z}(t_i) \equiv [z_1(t_i), \ldots, z_n(t_i)]$. Similarly, $\mathrm{d}\mathbf{z}(t)/\mathrm{d}t \in \mathbb{R}^n$ denotes the derivative of $\mathbf{z}(t)$ with respect to t and is given as $\mathrm{d}\mathbf{z}(t)/\mathrm{d}t \equiv [\mathrm{d}z_1(t)/\mathrm{d}t, \ldots, \mathrm{d}z_n(t)/\mathrm{d}t]$. Moreover, $\mathrm{d}\mathbf{z}(t_i)/\mathrm{d}t \in \mathbb{R}^n$ denotes $\mathrm{d}\mathbf{z}(t)/\mathrm{d}t$ evaluated at $t = t_i$ and is given as $\mathrm{d}\mathbf{z}(t_i)/\mathrm{d}t \equiv [\mathrm{d}z_1(t_i)/\mathrm{d}t, \ldots, \mathrm{d}z_n(t_i)/\mathrm{d}t]$. Finally, $\mathbf{z}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^m$ is a vector function that maps $\mathbf{x} \in \mathbb{R}^n$ to $\mathbf{z}(\mathbf{x}) \in \mathbb{R}^m$; whereas, $\mathbf{z}(\mathbf{x}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ is a multivariate vector function that

maps $\mathbf{x} \in \mathbb{R}^n$ and $t \in \mathbb{R}$ to $\mathbf{z}(\mathbf{x}, t) \in \mathbb{R}^m$.

Lastly, the following general notation and conventions are used. First, let t (that is, time) be the independent variable of the original formulation of the optimal control problem. In addition, let $t \in [t_0, t_f]$ denote the corresponding time horizon, where t_0 and t_f are the initial and terminal time, respectively. Second, let τ be the independent variable of the single-interval formulation of the Bolza optimal control problem of Section 3, where $\tau \in [-1, +1]$ denotes the corresponding domain transformed from $t \in [t_0, t_f]$. Furthermore, in the multiple-interval formulation of the Bolza optimal control problem of Section 3, the domain $\tau \in [-1, +1]$ is partitioned into a K-interval *mesh* defined by the strictly monotonically increasing *mesh points* τ_k, $(k = 1, \ldots, K+1)$, where $\tau_0 = -1$ and $\tau_K = +1$. Within each mesh interval $\mathcal{S}_k = [\tau_{k-1}, \tau_k] \subseteq [-1, +1]$, $(k = 1, \ldots, K)$, the domain $\tau \in [\tau_{k-1}, \tau_k]$ is then transformed to the new independent variable $\zeta \in [-1, +1]$. Moreover, let the notation $x^{(k)}$ denote a quantity (in this case, x), variable, or function in mesh interval k; whereas, the notation x^* denotes the optimal value of a quantity (in this case, x), variable, or function. With regards to mesh refinement, let the notation $x^{[m]}$ denote a quantity (in this case, x), variable, or function on mesh refinement iteration m, where $m = 0$ corresponds to the initial mesh.

3 Bolza Optimal Control Problem

Without loss of generality, consider the following single-interval, continuous-time Bolza optimal control problem defined on the interval $\tau \in [-1, +1]$. Determine the state, $\mathbf{x}(\tau) \in \mathbb{R}^{n_x}$, the control, $\mathbf{u}(\tau) \in \mathbb{R}^{n_u}$, the initial time, $t_0 \in \mathbb{R}$, and the terminal time, $t_f \in \mathbb{R}$, that minimize the objective functional

$$\mathcal{J} = \mathcal{M}(\mathbf{x}(-1), t_0, \mathbf{x}(+1), t_f) + \alpha \int_{-1}^{+1} \mathcal{L}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \mathrm{d}\tau, \tag{1}$$

subject to the dynamic constraints

$$\frac{\mathrm{d}\mathbf{x}(\tau)}{\mathrm{d}\tau} - \alpha \mathbf{a}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) = \mathbf{0}, \tag{2}$$

the boundary conditions

$$\mathbf{b}(\mathbf{x}(-1), t_0, \mathbf{x}(+1), t_f) \leq \mathbf{0}, \tag{3}$$

and the inequality path constraints

$$\mathbf{c}(\mathbf{x}(\tau), \mathbf{u}(\tau), \tau) \leq \mathbf{0}, \tag{4}$$

where $\alpha \equiv \mathrm{d}t/\mathrm{d}\tau = (t_f - t_0)/2$ is the *time interval scaling factor*. The functions $\mathcal{M}$, $\mathcal{L}$, $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ are defined by the mappings

$$\begin{aligned}
\mathcal{M} &: \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \rightarrow \mathbb{R}, \\
\mathcal{L} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}, \\
\mathbf{a} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_x}, \\
\mathbf{b} &: \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \rightarrow \mathbb{R}^{n_b}, \\
\mathbf{c} &: \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_c},
\end{aligned}$$

where n_x, n_u, n_b, and n_c denote the number of states, controls, boundary conditions, and inequality path constraints, respectively. The affine transformations relating the original time interval, $t \in [t_0, t_f]$, and $\tau \in [-1, +1]$ are given by

$$\begin{aligned}
t \equiv t(\tau, t_0, t_f) &= \alpha\tau + \alpha_0, \\
\tau \equiv \tau(t, t_0, t_f) &= \frac{1}{\alpha}\left(t - \alpha_0\right),
\end{aligned} \tag{5}$$

where $\alpha_0 \equiv (t_f + t_0)/2$ is the *time interval scaling offset*.

Suppose now that the domain $\tau \in [-1, +1]$ is partitioned into a *mesh* consisting of K *mesh intervals* $\mathcal{S}_k = [\tau_{k-1}, \tau_k] \subseteq [-1, +1]$, $(k = 1, \ldots, K)$, where the *mesh points* are $-1 = \tau_0 < \tau_1 < \cdots < \tau_{K-1} < \tau_K = +1$ and

$$\bigcup_{k=1}^{K} \mathcal{S}_k = [-1, +1], \qquad \mathcal{S}_k \bigcap \mathcal{S}_{k+1} = \{\tau_k\}, \qquad (k = 1, \ldots, K-1). \tag{6}$$

The domain within each mesh interval is then mapped to the domain $\zeta \in [-1, +1]$, and the affine transformations relating $\tau \in [\tau_{k-1}, \tau_k]$ and $\zeta \in [-1, +1]$ in each mesh interval are given by

$$\begin{aligned}
\tau \equiv \tau(\zeta, \tau_{k-1}, \tau_k) &= \beta_k \zeta + \beta_{k0}, \\
\zeta \equiv \zeta(\tau, \tau_{k-1}, \tau_k) &= \frac{1}{\beta_k}(\tau - \beta_{k0}),
\end{aligned} \qquad (k = 1, \ldots, K), \tag{7}$$

where $\beta_k \equiv \mathrm{d}\tau/\mathrm{d}\zeta = (\tau_k - \tau_{k-1})/2$ is a *mesh interval scaling factor* and $\beta_{k0} \equiv (\tau_k + \tau_{k-1})/2$ is a *mesh interval scaling offset*. Finally, the Bolza optimal control problem defined by Eqs. (1)–(4) is redefined in multiple-interval form as follows. Determine the state, $\mathbf{x}^{(k)}(\zeta) \in \mathbb{R}^{n_x}$, and the control, $\mathbf{u}^{(k)}(\zeta) \in \mathbb{R}^{n_u}$, in mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$, the initial time, t_0, and the terminal time, t_f, that minimize the objective functional

$$\mathcal{J} = \mathcal{M}(\mathbf{x}^{(1)}(-1), t_0, \mathbf{x}^{(K)}(+1), t_f) + \alpha \sum_{k=1}^{K} \beta_k \int_{-1}^{+1} \mathcal{L}(\mathbf{x}^{(k)}(\zeta), \mathbf{u}^{(k)}(\zeta), \zeta)\mathrm{d}\zeta, \tag{8}$$

subject to the dynamic constraints

$$\frac{\mathrm{d}\mathbf{x}^{(k)}(\zeta)}{\mathrm{d}\zeta} - \alpha\beta_k \mathbf{a}(\mathbf{x}^{(k)}(\zeta), \mathbf{u}^{(k)}(\zeta), \zeta) = \mathbf{0}, \qquad (k = 1, \ldots, K), \tag{9}$$

the boundary conditions

$$\mathbf{b}(\mathbf{x}^{(1)}(-1), t_0, \mathbf{x}^{(K)}(+1), t_f) \leq \mathbf{0}, \tag{10}$$

the inequality path constraints

$$\mathbf{c}(\mathbf{x}^{(k)}(\zeta), \mathbf{u}^{(k)}(\zeta), \zeta) \leq \mathbf{0}, \qquad (k = 1, \ldots, K), \tag{11}$$

and the continuity constraints

$$\mathbf{x}^{(k)}(+1) - \mathbf{x}^{(k+1)}(-1) = \mathbf{0}, \qquad (k = 1, \ldots, K-1). \tag{12}$$

It is noted that the constraints of Eq. (12) ensure state continuity at every interior mesh point.

4 Legendre-Gauss-Radau Collocation

The multiple-interval formulation of the continuous-time Bolza optimal control problem defined by Eqs. (8)–(12) is discretized using collocation at the standard LGR points [12,14,30]. In mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$, the state is approximated using a basis of Lagrange polynomials, $\ell_j^{(k)}(\zeta)$, $(j = 1, \ldots, N_k + 1)$, as

$$\mathbf{x}^{(k)}(\zeta) \approx \mathbf{X}^{(k)}(\zeta) = \sum_{j=1}^{N_k+1} \ell_j^{(k)}(\zeta) \mathbf{X}_j^{(k)}, \qquad \ell_j^{(k)}(\zeta) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k+1} \frac{\zeta - \zeta_l^{(k)}}{\zeta_j^{(k)} - \zeta_l^{(k)}}, \tag{13}$$

where $\mathbf{X}_j^{(k)} \in \mathbb{R}^{n_x}$ denotes the state approximation at $\zeta_j^{(k)}$, $\{\zeta_1^{(k)}, \ldots, \zeta_{N_k}^{(k)}\} \in [-1, +1)$ is the set of N_k LGR collocation points, and $\zeta_{N_k+1}^{(k)} = +1$ is a noncollocated point. An advantage to employing a Lagrange polynomial approximation with LGR support points is that the classic Runge phenomenon is eliminated [31]. Differentiating $\mathbf{X}^{(k)}(\zeta)$ in Eq. (13) with respect to ζ gives

$$\frac{\mathrm{d}\mathbf{x}^{(k)}(\zeta)}{\mathrm{d}\zeta} \approx \frac{\mathrm{d}\mathbf{X}^{(k)}(\zeta)}{\mathrm{d}\zeta} = \sum_{j=1}^{N_k+1} \frac{\mathrm{d}\ell_j^{(k)}(\zeta)}{\mathrm{d}\zeta} \mathbf{X}_j^{(k)}, \qquad (k = 1, \ldots, K). \tag{14}$$

Within each mesh interval, evaluating $\mathrm{d}\ell_j^{(k)}(\zeta)/\mathrm{d}\zeta$ in Eq. (14) at the N_k LGR points gives the elements of the *LGR differentiation matrix*, $\mathbf{D}^{(k)} \in \mathbb{R}^{N_k \times (N_k+1)}$, as

$$\mathbf{D}_{ij}^{(k)} \equiv \frac{\mathrm{d}\ell_j^{(k)}(\zeta_i^{(k)})}{\mathrm{d}\zeta}, \qquad \begin{pmatrix} i = 1, \ldots, N_k \\ j = 1, \ldots, N_k + 1 \\ k = 1, \ldots, K \end{pmatrix}. \tag{15}$$

The discretization of the multiple-interval, continuous-time Bolza optimal control problem defined by Eqs. (8)–(12) is given as follows. First, the objective functional of Eq. (8) is approximated using a multiple-interval LGR quadrature. Next, the dynamic constraints and inequality path

constraints of Eqs. (9) and (11), respectively, are collocated at the LGR points in each mesh interval. Then, the boundary conditions of Eq. (10) are approximated at the boundary points, while the continuity constraints of Eq. (12) are enforced using the approximation of the state at the terminus of an interior mesh interval and the start of the next interior mesh interval. The resulting NLP is formally stated as follows. Determine the state, $\mathbf{X}_j^{(k)}$, $(j = 1, \ldots, N_k + 1)$, and the control, $\mathbf{U}_j^{(k)}$, $(j = 1, \ldots, N_k)$, in mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$, the initial time, t_0, and the terminal time, t_f, that minimize the objective function

$$\mathcal{J} = \mathcal{M}(\mathbf{X}_1^{(1)}, t_0, \mathbf{X}_{N_K+1}^{(K)}, t_f) + \alpha \sum_{k=1}^{K} \beta_k \sum_{i=1}^{N_k} w_i^{(k)} \mathcal{L}(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}, \zeta_i^{(k)}), \tag{16}$$

subject to the discretized dynamic constraints

$$\sum_{j=1}^{N_k+1} \mathbf{D}_{ij}^{(k)} \mathbf{X}_j^{(k)} - \alpha \beta_k \mathbf{a}(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}, \zeta_i^{(k)}) = \mathbf{0}, \qquad \begin{pmatrix} i = 1, \ldots, N_k \\ k = 1, \ldots, K \end{pmatrix}, \tag{17}$$

the discretized boundary conditions

$$\mathbf{b}(\mathbf{X}_1^{(1)}, t_0, \mathbf{X}_{N_K+1}^{(K)}, t_f) \leq \mathbf{0}, \tag{18}$$

the discretized inequality path constraints

$$\mathbf{c}(\mathbf{X}_i^{(k)}, \mathbf{U}_i^{(k)}, \zeta_i^{(k)}) \leq \mathbf{0}, \qquad \begin{pmatrix} i = 1, \ldots, N_k \\ k = 1, \ldots, K \end{pmatrix}, \tag{19}$$

and the discretized continuity constraints

$$\mathbf{X}_{N_k+1}^{(k)} - \mathbf{X}_1^{(k+1)} = \mathbf{0}, \qquad (k = 1, \ldots, K-1), \tag{20}$$

where $\mathbf{U}_i^{(k)} \in \mathbb{R}^{n_u}$ and $w_i^{(k)} \in \mathbb{R}$ are the control approximation at $\zeta_i^{(k)}$ and i^{th} LGR quadrature weight, respectively, in mesh interval $\mathcal{S}_k$. Because the class of problems considered in this work are only those with a continuous state profile, the discretized continuity constraints of Eq. (20) are included in the NLP formulation; furthermore, the constraints of Eq. (20) are implicitly satisfied by employing the same NLP decision variables for $\mathbf{X}_{N_k+1}^{(k)}$ and $\mathbf{X}_1^{(k+1)}$, $(k = 1, \ldots, K-1)$.

5 Adaptive Mesh Refinement Method

In this section, the adaptive mesh refinement method of this paper is described. First, Section 5.1 provides a brief overview of numerical methods for solving a set of ordinary differential equations (ODEs). Second, an estimate of the relative error in the solution on a given mesh is derived in Section 5.2, which guides the mesh refinement process of Section 5.3. Next, two methods for

increasing the size of the mesh are presented in Sections 5.3.1 and 5.3.2, which include increasing the degree of the polynomial approximation in an interval and dividing an interval. Then, two methods for decreasing the size of the mesh are presented in Sections 5.3.3 and 5.3.4, which include merging adjacent intervals and decreasing the degree of the polynomial approximation in an interval. Finally, Section 5.4 summarizes the adaptive mesh refinement method.

5.1 Numerical Simulation in a Mesh Interval

The dynamic constraints defined by Eq. (9) associated with the multiple-interval formulation of the continuous-time Bolza optimal control problem in Section 3 are represented by a set of ODEs. In order to obtain solutions to optimal control problems numerically, the use of numerical methods for solving the set of ODEs is required, which are categorized as either *collocation* (implicit simulation) or *time-marching* (explicit simulation) methods. As demonstrated by the LGR collocation method of Section 4, collocation schemes divide each mesh interval domain into steps, and the solution at every step in every mesh interval is obtained simultaneously; however, the objective of this section is to sequentially obtain the solution at each step in each mesh interval using a time-marching scheme. Regardless of the numerical method chosen, solving the set of ODEs can be posed in the form of an initial-value problem (IVP) given an initial condition or a terminal-value problem (TVP) given a terminal condition.

Suppose now that the NLP of Eqs. (16)–(20) is solved on a given mesh, resulting in the values of the state, $\mathbf{X}_j^{(k)}$, $(j = 1, \ldots, N_{k+1})$, and control, $\mathbf{U}_j^{(k)}$, $(j = 1, \ldots, N_k)$, in mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$. Given this discrete approximation, the objective is to obtain an error estimate in each mesh interval based on the differences between the Lagrange polynomial approximation of the state in Eq. (13) and state approximations obtained via *forward* and *backward* time-marching. In order to approximate the state via forward time-marching, an IVP must be formulated and solved. In mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$, the single-interval IVP is formulated as

$$\frac{\mathrm{d}\hat{\mathbf{X}}^{(k)}(\zeta)}{\mathrm{d}\zeta} = \alpha\beta_k \mathbf{a}(\hat{\mathbf{X}}^{(k)}(\zeta), \tilde{\mathbf{U}}^{(k)}(\zeta), \zeta), \qquad \hat{\mathbf{X}}^{(k)}(-1) = \mathbf{X}_1^{(k)}. \tag{21}$$

Suppose that explicitly simulating Eq. (21) from $\zeta = -1$ to $\zeta = +1$ in every mesh interval yields approximated values of the state, $\hat{\mathbf{X}}^{(k)}(\hat{\zeta}_j^{(k)})$, $(j = 1, \ldots, \hat{P}_k;\ k = 1, \ldots, K)$, at the $\hat{P}_k$ forward propagation points $\{\hat{\zeta}_1^{(k)}, \ldots, \hat{\zeta}_{\hat{P}_k}^{(k)}\} \in [-1, +1]$. Similarly, in order to approximate the state via backward time-marching, a TVP must be formulated and solved. In mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$, the single-interval TVP is formulated as

$$\frac{\mathrm{d}\check{\mathbf{X}}^{(k)}(\zeta)}{\mathrm{d}\zeta} = \alpha\beta_k \mathbf{a}(\check{\mathbf{X}}^{(k)}(\zeta), \tilde{\mathbf{U}}^{(k)}(\zeta), \zeta), \qquad \check{\mathbf{X}}^{(k)}(+1) = \mathbf{X}_{N_k+1}^{(k)}. \tag{22}$$

Suppose that explicitly simulating Eq. (22) from $\zeta = +1$ to $\zeta = -1$ in every mesh interval yields approximated values of the state, $\check{\mathbf{X}}^{(k)}(\check{\zeta}_l^{(k)})$, $(l = 1, \ldots, \check{P}_k;\ k = 1, \ldots, K)$, at the $\check{P}_k$ backward propagation points $\{\check{\zeta}_1^{(k)}, \ldots, \check{\zeta}_{\check{P}_k}^{(k)}\} \in [-1, +1]$. It is noted that the sets of points $\{\zeta_1^{(k)}, \ldots, \zeta_{N_k+1}^{(k)}\}$, $\{\hat{\zeta}_1^{(k)}, \ldots, \hat{\zeta}_{\hat{P}_k}^{(k)}\}$, and $\{\check{\zeta}_1^{(k)}, \ldots, \check{\zeta}_{\check{P}_k}^{(k)}\}$ are not all necessarily equivalent in mesh interval $\mathcal{S}_k$. In order to obtain solutions to Eqs. (21) and (22), a control function, $\tilde{\mathbf{U}}^{(k)}(\zeta)$, $(k = 1, \ldots, K)$, is required because time-marching schemes take integration steps to points other than the collocation points. In this work, a Lagrange polynomial approximation of the control is defined in mesh interval $\mathcal{S}_k$ as

$$\tilde{\mathbf{U}}^{(k)}(\zeta) = \sum_{j=1}^{N_k} \tilde{\ell}_j^{(k)}(\zeta)\mathbf{U}_j^{(k)}, \qquad \tilde{\ell}_j^{(k)}(\zeta) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k} \frac{\zeta - \zeta_l^{(k)}}{\zeta_j^{(k)} - \zeta_l^{(k)}}, \qquad (k = 1, \ldots, K), \tag{23}$$

where $\{\zeta_1^{(k)}, \ldots, \zeta_{N_k}^{(k)}\} \in [-1, +1)$ is the set of N_k LGR collocation points. Equation (23) differs from the state approximation in Eq. (13) as it does not include the noncollocated point. Finally, any numerical method for solving a set of ODEs (for example, `ode45` or `ode89` in `MATLAB`) can be employed to solve Eqs. (21) and (22). Using any such ODE solver, the sets of propagation points can either be specified by the user or determined by the ODE solver. Performance of the mesh refinement method described in this paper is impacted by the ODE solver and accuracy tolerance of the solver.

5.2 Relative Error Estimate in a Mesh Interval

In this section, an estimate of the relative error in the solution on a given mesh is derived. In the LGR collocation method, a uniquely defined function approximation is only available for the state; therefore, a relative error estimate is developed for the state only. The key idea is that numerically solving a set of ODEs via collocation and/or time-marching schemes should yield nearly identical results, barring any propagation error or discrepancy between the accuracy of the numerical methods. Thus, the differences between the Lagrange polynomial approximation of the state obtained by solving the NLP of Eqs. (16)–(20) and state approximations obtained by solving the IVP and TVP of Eqs. (21) and (22) via forward and backward time-marching, respectively, should yield an approximation of the error in the state. Consequently, using this newly developed error estimate ensures that the solutions obtained via collocation and explicit simulation schemes are in agreement with one another on the final mesh.

Suppose that the values of the state approximation given in Eq. (13) in mesh interval $\mathcal{S}_k$, $(k = 1, \ldots, K)$, at the forward and backward propagation points $\{\hat{\zeta}_1^{(k)}, \ldots, \hat{\zeta}_{\hat{P}_k}^{(k)}\}$ and $\{\check{\zeta}_1^{(k)}, \ldots, \check{\zeta}_{\check{P}_k}^{(k)}\}$ are denoted $\{\mathbf{X}^{(k)}(\hat{\zeta}_1^{(k)}), \ldots, \mathbf{X}^{(k)}(\hat{\zeta}_{\hat{P}_k}^{(k)})\}$ and $\{\mathbf{X}^{(k)}(\check{\zeta}_1^{(k)}), \ldots, \mathbf{X}^{(k)}(\check{\zeta}_{\check{P}_k}^{(k)})\}$, respectively. Then, the forward and backward *relative errors* in the i^{th} component of the state, $(i = 1, \ldots, n_x)$, at the

points $\{\hat{\zeta}_1^{(k)},\ldots,\hat{\zeta}_{\hat{P}_k}^{(k)}\}$ and $\{\check{\zeta}_1^{(k)},\ldots,\check{\zeta}_{\check{P}_k}^{(k)}\}$ are defined, respectively, as

$$\begin{aligned}\hat{e}_i^{(k)}(\hat{\zeta}_j^{(k)}) &= \gamma_i\left|\hat{X}_i^{(k)}(\hat{\zeta}_j^{(k)}) - X_i^{(k)}(\hat{\zeta}_j^{(k)})\right|, \\ \check{e}_i^{(k)}(\check{\zeta}_l^{(k)}) &= \gamma_i\left|\check{X}_i^{(k)}(\check{\zeta}_l^{(k)}) - X_i^{(k)}(\check{\zeta}_l^{(k)})\right|,\end{aligned} \qquad \begin{pmatrix} j=1,\ldots,\hat{P}_k \\ k=1,\ldots,K \\ l=1,\ldots,\check{P}_k\end{pmatrix}, \tag{24}$$

where the corresponding *error scaling factor*, $\gamma_i \in \mathbb{R}$, is defined such that

$$\gamma_i^{-1} = 1 + \max_{\substack{j\in\{1,\ldots,N_k+1\}\\ k\in\{1,\ldots,K\}}}\left|X_i^{(k)}(\zeta_j^{(k)})\right|, \qquad (i=1,\ldots,n_x), \tag{25}$$

and $X_i^{(k)}(x)$ denotes the i^{th} component of the state approximation $\mathbf{X}^{(k)}(\zeta)$ evaluated at the specified point (in this case, x). The choice of error scaling factors in Eq. (25) is made for the following two reasons. First, the constant in Eq. (25) is used to avoid division by zero, which could otherwise occur when a state component is zero and constant. Second, normalizing by the maximum absolute value of each state component brings all state component absolute errors to a similar order of magnitude for use in the mesh refinement process. To account for both relative errors in Eq. (24), the *maximum relative error* in mesh interval $\mathcal{S}_k$, $(k=1,\ldots,K)$, is defined as

$$e_{\max}^{(k)} = \max_{i\in\{1,\ldots,n_x\}}\left(\max_{j\in\{1,\ldots,\hat{P}_k\}}\hat{e}_i^{(k)}(\hat{\zeta}_j^{(k)}), \max_{l\in\{1,\ldots,\check{P}_k\}}\check{e}_i^{(k)}(\check{\zeta}_l^{(k)})\right). \tag{26}$$

5.3 Mesh Refinement in a Mesh Interval

The objective of the mesh refinement method is to meet the desired mesh tolerance, ϵ, in every mesh interval on the smallest mesh (that is, with the fewest total number of collocation points). For simplicity, let $\mathcal{K}^+$, $(\mathcal{K}^+ \subseteq \{1,\ldots,K\})$, denote the set of mesh intervals in which the desired mesh tolerance is not met (that is, $e_{\max}^{(k)} > \epsilon$, $(k\in\mathcal{K}^+)$); furthermore, let $\mathcal{K}^-$, $(\mathcal{K}^- \subseteq \{1,\ldots,K\})$, denote the set of mesh intervals in which the desired mesh tolerance is met (that is, $e_{\max}^{(k)} \leq \epsilon$, $(k\in\mathcal{K}^-)$). It is noted that $\mathcal{K}^+\cap\mathcal{K}^- = \emptyset$ and $\mathcal{K}^+\cup\mathcal{K}^- = \{1,\ldots,K\}$. The current mesh is refined only if $\mathcal{K}^+ \neq \emptyset$. First, the method attempts to increase the degree of the polynomial approximation in a mesh interval and/or divide a mesh interval into subintervals. Then, the method attempts to merge adjacent mesh intervals and/or decrease the degree of the polynomial approximation in a mesh interval.

5.3.1 p Refinement: Increasing the Polynomial Degree in a Mesh Interval

To reduce the maximum relative error, p refinement attempts to strictly increase the number of collocation points $N_k^{[m]}$ in mesh interval $\mathcal{S}_k$, $(k\in\mathcal{K}^+)$, on mesh m to

$$N_k^{[m+1]} = N_k^{[m]} + P_k^+, \qquad P_k^+ = \left\lceil \log_{10}\left(\frac{e_{\max}^{(k)}}{\epsilon}\right)\right\rceil, \tag{27}$$

where $N_k^{[m+1]}$ denotes the number of collocation points in $\mathcal{S}_k$ on the ensuing mesh, $\lceil \cdot \rceil$ denotes the ceiling function, and $P_k^+ \geq 1$ because $e_{\max}^{(k)} > \epsilon$. The choice of P_k^+ in Eq. (27) takes advantage of the exponential convergence rate obtained by employing a Gaussian quadrature collocation method, where the growth in the polynomial degree is related to the log of the error. Furthermore, using $\log_{10}$ helps to avoid adding collocation points unnecessarily relative to that of $\log_{N_k}$ (that is, for $N_k < 10$), which is employed by Patterson et al. [23]. Equation (27) also removes the arbitrary constant found in the p refinement step of Darby et al. [22] in order to keep the size of the mesh as small as possible. To ensure that the approximating polynomial degree in a mesh interval does not grow to an unreasonably large value, a user-specified upper limit $N_{\max} \geq 2$ is set for the maximum allowable polynomial approximation degree. If $N_{\max}$ is exceeded (that is, p refinement is exhausted), then the mesh interval is divided into subintervals using the method presented in Section 5.3.2.

5.3.2 h Refinement: Dividing a Mesh Interval into Subintervals

Suppose that evaluating Eq. (27) yields $N_k^{[m+1]} > N_{\max}$, which indicates that mesh interval $\mathcal{S}_k$, $(k \in \mathcal{K}^+)$, on mesh m must be divided into subintervals. Identical to the h refinement method described in Patterson et al. [23], it is desired to keep the predicted number of total collocation points on the ensuing mesh and employ $N_{\min}$ collocation points in each newly created subinterval, where $N_{\min}$ denotes a user-specified minimum allowable polynomial approximation degree such that $N_{\max} \geq N_{\min} \geq 2$. Thus, mesh interval $\mathcal{S}_k$, $(k \in \mathcal{K}^+)$, is divided into H_k uniformly spaced subintervals on the ensuing mesh, where

$$H_k = \max\left(2, \left\lceil \frac{N_k^{[m+1]}}{N_{\min}} \right\rceil\right). \tag{28}$$

The new subintervals $\hat{\mathcal{S}}_q = [\hat{\tau}_{q-1}^{(k)}, \hat{\tau}_q^{(k)}] \subset \mathcal{S}_k = [\tau_{k-1}, \tau_k]$, $(q = 1, \ldots, H_k;\ k \in \mathcal{K}^+)$, are created by defining the mesh points on the ensuing mesh as

$$\hat{\tau}_q^{(k)} = \tau_{k-1} + q\left(\frac{\tau_k - \tau_{k-1}}{H_k}\right), \qquad (q = 0, \ldots, H_k), \tag{29}$$

where $\tau_{k-1} = \hat{\tau}_0^{(k)} < \hat{\tau}_1^{(k)} < \cdots < \hat{\tau}_{H_k-1}^{(k)} < \hat{\tau}_{H_k}^{(k)} = \tau_k$.

5.3.3 h Reduction: Merging Adjacent Mesh Intervals

In addition to increasing the mesh size via p and h refinement, the mesh size can be decreased via either h or p reduction; however, h and p reduction are only attempted if the maximum relative error in at least one mesh interval does not satisfy the desired mesh tolerance (that is, $\mathcal{K}^+ \neq \emptyset$). The method developed in this research for h reduction utilizes state approximations obtained via forward

and backward time marching over the domain of both mesh intervals to determine if merging is possible, which extends the single-interval ideas of Sections 5.1 and 5.2 to adjacent mesh intervals. It is noted that this h reduction step differs significantly from that described in Liu et al. [24], where different polynomial approximations of the state are obtained from power series expansions about the shared mesh point.

First, let $\mathcal{Q}$, ($\mathcal{Q} \subseteq \mathcal{K}^-;\ K \notin \mathcal{Q}$), denote the set of adjacent mesh interval pairs that both satisfy the desired mesh tolerance. The goal is to merge adjacent mesh intervals $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$, ($\{k, k+1\} \in \mathcal{K}^-$), to form a single mesh interval $\bar{\mathcal{S}}_q = \mathcal{S}_k \cup \mathcal{S}_{k+1} = [\tau_{k-1}, \tau_{k+1}]$, ($q \in \mathcal{Q}$). For simplicity, let the following adjacent mesh interval transformations be

$$\begin{aligned} \chi(x) &\equiv \chi(x, \tau_{k-1}, \tau_k, \tau_{k+1}) = \zeta(\tau(x, \tau_{k-1}, \tau_k), \tau_k, \tau_{k+1}), \\ \xi(x) &\equiv \xi(x, \tau_{k-1}, \tau_k, \tau_{k+1}) = \zeta(\tau(x, \tau_k, \tau_{k+1}), \tau_{k-1}, \tau_k), \end{aligned} \qquad (k = 1, \ldots, K-1), \tag{30}$$

where $\chi(x)$ transforms a quantity (in this case, x) relative to $\zeta^{(k)} \in [-1, +1]$ to its corresponding value relative to $\zeta^{(k+1)} \in [-1, +1]$, and $\xi(x)$ provides the inverse transformation. It is noted that Eq. (30) utilizes the transformations in Eq. (7). Similar to Eqs. (21) and (22), an IVP and TVP must both be formulated and solved in order to approximate the state over the domain of the adjacent mesh intervals via forward and backward time-marching, which are defined as follows. First, the multiple-interval IVP in mesh interval $\bar{\mathcal{S}}_q$, ($q \in \mathcal{Q}$), is formulated as

$$\frac{\mathrm{d}\hat{\tilde{\mathbf{X}}}^{(q)}(\zeta)}{\mathrm{d}\zeta} = \alpha\beta_k \mathbf{a}(\hat{\tilde{\mathbf{X}}}^{(q)}(\zeta), \tilde{\mathbf{U}}^{(k)}(\zeta), \zeta), \qquad \hat{\tilde{\mathbf{X}}}^{(q)}(-1) = \mathbf{X}_1^{(k)}. \tag{31}$$

Suppose that explicitly simulating Eq. (31) from $\zeta = -1$ to $\zeta = \xi(\zeta_{N_{k+1}+1}^{(k+1)}) > +1$ (that is, forward from mesh interval $\mathcal{S}_k$ into $\mathcal{S}_{k+1}$) yields approximated values of the state, $\hat{\tilde{\mathbf{X}}}^{(q)}(\hat{\tilde{\zeta}}_j^{(q)})$, ($j = 1, \ldots, \hat{\tilde{P}}_q$), at the $\hat{\tilde{P}}_q$ extended forward propagation points $\{\hat{\tilde{\zeta}}_1^{(q)}, \ldots, \hat{\tilde{\zeta}}_{\hat{\tilde{P}}_q}^{(q)}\} \in [-1, \xi(\zeta_{N_{k+1}+1}^{(k+1)})]$ in mesh interval $\bar{\mathcal{S}}_q$, ($q \in \mathcal{Q}$). Next, the multiple-interval TVP in mesh interval $\bar{\mathcal{S}}_q$, ($q \in \mathcal{Q}$), is formulated as

$$\frac{\mathrm{d}\check{\tilde{\mathbf{X}}}^{(q)}(\zeta)}{\mathrm{d}\zeta} = \alpha\beta_{k+1} \mathbf{a}(\check{\tilde{\mathbf{X}}}^{(q)}(\zeta), \tilde{\mathbf{U}}^{(k+1)}(\zeta), \zeta), \qquad \check{\tilde{\mathbf{X}}}^{(q)}(+1) = \mathbf{X}_{N_{k+1}+1}^{(k+1)}. \tag{32}$$

Suppose that explicitly simulating Eq. (32) from $\zeta = +1$ to $\zeta = \chi(\zeta_1^{(k)}) < -1$ (that is, backward from mesh interval $\mathcal{S}_{k+1}$ into $\mathcal{S}_k$) yields approximated values of the state, $\check{\tilde{\mathbf{X}}}^{(q)}(\check{\tilde{\zeta}}_l^{(q)})$, ($l = 1, \ldots, \check{\tilde{P}}_q$), at the $\check{\tilde{P}}_q$ extended backward propagation points $\{\check{\tilde{\zeta}}_1^{(q)}, \ldots, \check{\tilde{\zeta}}_{\check{\tilde{P}}_q}^{(q)}\} \in [\chi(\zeta_1^{(k)}), +1]$ in mesh interval $\bar{\mathcal{S}}_q$, ($q \in \mathcal{Q}$). It is noted that the control in mesh interval $\mathcal{S}_k$ is extrapolated forward into mesh interval $\mathcal{S}_{k+1}$ in the multiple-interval IVP of Eq. (31), whereas the control in mesh interval $\mathcal{S}_{k+1}$ is extrapolated backward into mesh interval $\mathcal{S}_k$ in the multiple-interval TVP of Eq. (32).

In order to determine whether or not the adjacent mesh intervals can be merged, the relative error is estimated over the domain of the adjacent mesh intervals using the state approximations

obtained by solving the multiple-interval IVP and TVP of Eqs. (31) and (32), respectively. Because the state approximations given by Eq. (13) in the adjacent mesh intervals are not necessarily equivalent, the appropriate state approximations must be compared at the corresponding propagation points. Similar to Eq. (24), the relative errors in the i^{th} component of the state at the extended forward and backward propagation points $\{\hat{\zeta}_1^{(q)}, \ldots, \hat{\zeta}_{\hat{P}_q}^{(q)}\}$ and $\{\check{\zeta}_1^{(q)}, \ldots, \check{\zeta}_{\check{P}_q}^{(q)}\}$ in mesh interval $\bar{\mathcal{S}}_q$, $(q \in \mathcal{Q})$, are defined, respectively, as

$$\hat{e}_i^{(q)}(\hat{\zeta}_j^{(q)}) = \begin{cases} \gamma_i \left| \hat{X}_i^{(q)}(\hat{\zeta}_j^{(q)}) - X_i^{(k)}(\hat{\zeta}_j^{(q)}) \right| & \text{for} \quad -1 \leq \hat{\zeta}_j^{(q)} \leq +1, \\ \gamma_i \left| \hat{X}_i^{(q)}(\hat{\zeta}_j^{(q)}) - X_i^{(k+1)}(\chi(\hat{\zeta}_j^{(q)})) \right| & \text{for} \quad +1 < \hat{\zeta}_j^{(q)} \leq \xi(\zeta_{N_{k+1}+1}^{(k+1)}), \end{cases}$$

$$\check{e}_i^{(q)}(\check{\zeta}_l^{(q)}) = \begin{cases} \gamma_i \left| \check{X}_i^{(q)}(\check{\zeta}_l^{(q)}) - X_i^{(k)}(\xi(\check{\zeta}_l^{(q)})) \right| & \text{for} \quad \chi(\zeta_1^{(k)}) \leq \check{\zeta}_l^{(q)} < -1, \\ \gamma_i \left| \check{X}_i^{(q)}(\check{\zeta}_l^{(q)}) - X_i^{(k+1)}(\check{\zeta}_l^{(q)}) \right| & \text{for} \quad -1 \leq \check{\zeta}_l^{(q)} \leq +1, \end{cases} \tag{33}$$

$$(i = 1, \ldots, n_x;\ j = 1, \ldots, \hat{P}_q;\ \{k, k+1\} \in \mathcal{K}^-;\ l = 1, \ldots, \check{P}_q),$$

where $X_i^{(k)}(x)$ and $X_i^{(k+1)}(x)$ denote the i^{th} component of the respective state approximations $\mathbf{X}^{(k)}(\zeta)$ and $\mathbf{X}^{(k+1)}(\zeta)$ given by Eq. (13) evaluated at the specified point (in this case, x). To account for both relative errors in Eq. (33), the maximum relative error in mesh interval $\bar{\mathcal{S}}_q$, $(q \in \mathcal{Q})$, is defined as

$$\bar{e}_{\max}^{(q)} = \max_{i \in \{1, \ldots, n_x\}} \left(\max_{j \in \{1, \ldots, \hat{P}_q\}} \hat{e}_i^{(q)}(\hat{\zeta}_j^{(q)}), \max_{l \in \{1, \ldots, \check{P}_q\}} \check{e}_i^{(q)}(\check{\zeta}_l^{(q)}) \right). \tag{34}$$

It is noted that the adjacent mesh intervals cannot be merged if $\bar{e}_{\max}^{(q)} > \epsilon$, $(q \in \mathcal{Q})$.

Once the method determines all pairs of adjacent mesh intervals that can be merged, merging then occurs in ascending order based on the overall maximum relative error obtained via Eqs. (26) and (34) (that is, based on $\max(e_{\max}^{(k)}, e_{\max}^{(k+1)}, \bar{e}_{\max}^{(q)})$). If h reduction is performed, the number of collocation points $N_q^{[m+1]}$ in mesh interval $\bar{\mathcal{S}}_q$, $(q \in \mathcal{Q})$, on mesh $m+1$ is determined by

$$N_q^{[m+1]} = \max\left(N_k^{[m]}, N_{k+1}^{[m]}\right), \tag{35}$$

where the mesh point τ_k is removed on the ensuing mesh. It is important to note that the h reduction step presented in this research does not require the number of collocation points in adjacent mesh intervals to be equal, which can be a limiting factor enforced by the method described in Liu et al. [24]. It is possible that the merged region may not satisfy the desired error tolerance on the ensuing mesh (that is, in the region where h reduction is successfully performed on the current mesh) because of changes in the state and/or control solutions obtained from solving the NLP of Eqs. (16)–(20) on the current mesh versus the ensuing mesh. In this scenario, the proposed mesh refinement method proceeds as normal, where p or h refinement would be performed in the appropriate region on the ensuing mesh.

5.3.4 p Reduction: Decreasing the Polynomial Degree in a Mesh Interval

Before attempting p reduction, all possible mesh intervals are merged using the method described in Section 5.3.3. If mesh interval $\mathcal{S}_k$, $(k \in \mathcal{K}^-)$, is not merged, then p reduction attempts to decrease the number of collocation points $N_k^{[m]}$ on mesh m to

$$N_k^{[m+1]} = \max\left(N_{\min}, N_k^{[m]} - P_k^-\right), \qquad P_k^- = \left\lfloor \log_{10}\left(\frac{\epsilon}{e_{\max}^{(k)}}\right)^{1/\delta} \right\rfloor, \qquad \delta = N_{\min} + N_{\max} - N_k^{[m]}, \tag{36}$$

where $\lfloor \cdot \rfloor$ denotes the floor function, and $P_k^- \geq 0$ because $e_{\max}^{(k)} \leq \epsilon$. Different from the p reduction step described in Liu et al. [24] that utilizes a different polynomial approximation of the state obtained from a power series expansion about the mesh interval midpoint, the choice of P_k^- in Eq. (36) follows similar logic to that justifying the choice of P_k^+ in Eq. (27). In addition, to avoid removing too many collocation points, the parameter δ in Eq. (36) controls the acceptable order of magnitude difference between the current maximum relative error in a mesh interval and the desired mesh tolerance, where a larger number of collocation points already present in a mesh interval yields a smaller value for δ and vice versa. When $P_k^- = 0$, the polynomial approximation degree in mesh interval $\mathcal{S}_k$, $(k \in \mathcal{K}^-)$, cannot be reduced. Finally, Eq. (36) also ensures that the number of collocation points in a mesh interval is not decreased below the lower limit $N_{\min}$.

5.4 Summary of Adaptive Mesh Refinement Method

A summary of the method developed in this research is shown in Algorithm 1, which consists of three major components: numerical simulation, relative state error estimation, and mesh refinement in a mesh interval, as discussed in Sections 5.1–5.3, respectively. Mesh refinement in a mesh interval consists of four steps: p refinement, h refinement, h reduction, and p reduction, as discussed in Sections 5.3.1–5.3.4, respectively. Six user-specified parameters are required: an ODE solver, $\mathtt{s}$, an ODE solver tolerance, ϵ_{ODE}, a mesh tolerance, ϵ, a maximum number of mesh refinement iterations, $M_{\max}$, and minimum and maximum allowable numbers of collocation points, $N_{\min}$ and $N_{\max}$, respectively. As with any mesh refinement method, performance of the method depends on the method parameters, initial mesh, and mesh accuracy tolerance.

6 Examples

In this section, the mesh refinement method of Section 5 is demonstrated on three examples from the open literature. The first example is the minimum-time supersonic aircraft climb problem

Algorithm 1 Adaptive Mesh Refinement Method

Supply an initial mesh that satisfies Eq. (6). ▷ Section 3
Assign user-specified parameters: $\mathbf{s}$, ϵ_{ODE}, ϵ, $M_{\max}$, $N_{\min}$, $N_{\max}$. ▷ Section 5.4
$m \leftarrow 0$.
while $m < M_{\max}$ **do**
    $K \leftarrow$ number of mesh intervals on mesh m.
    $N_k \leftarrow$ number of collocation points in mesh interval $\mathcal{S}_k$ on mesh m.
    Solve the NLP on mesh m as follows: ▷ Section 4
    $\mathbf{X}^{(k)}_{1:N_k+1}, \mathbf{U}^{(k)}_{1:N_k}\ \forall\ k = 1 \ldots, K \leftarrow$ Solve Eqs. (16)–(20).
    for $k = 1, \ldots, K$ **do**
        Explicitly simulate the dynamics as follows: ▷ Section 5.1
        $\hat{\mathbf{X}}^{(k)}_{1:\hat{P}_k} \leftarrow$ Solve the IVP of Eq. (21) in $\mathcal{S}_k$.
        $\check{\mathbf{X}}^{(k)}_{1:\check{P}_k} \leftarrow$ Solve the TVP of Eq. (22) in $\mathcal{S}_k$.
        Compute the maximum relative error estimate as follows: ▷ Section 5.2
        $e^{(k)}_{\max} \leftarrow$ Solve Eq. (26) in $\mathcal{S}_k$.
    end for
    if $e^{(k)}_{\max} \leq \epsilon\ \forall\ k = 1, \ldots, K$ **then**
        exit
    end if
    Refine mesh m as follows: ▷ Section 5.3
    for $k = 1, \ldots, K$ **do**
        if $e^{(k)}_{\max} > \epsilon$ **then**
            Attempt p refinement in $\mathcal{S}_k$. ▷ Section 5.3.1
            if p refinement fails in $\mathcal{S}_k$ **then**
                Perform h refinement in $\mathcal{S}_k$. ▷ Section 5.3.2
            end if
        else
            if $k < K$ **and** $e^{(k+1)}_{\max} \leq \epsilon$ **then**
                Attempt h reduction in $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$. ▷ Section 5.3.3
                if h reduction fails in $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ **then**
                    Attempt p reduction in $\mathcal{S}_k$. ▷ Section 5.3.4
                end if
            else
                Attempt p reduction in $\mathcal{S}_k$. ▷ Section 5.3.4
            end if
        end if
    end for
    $m \leftarrow m + 1$.
end while

taken from Darby et al. [22] that uses the dynamic model provided in Seywald et al. [32]. This first example demonstrates how the method is able to handle a challenging real-world problem with an active state constraint and a smooth control profile. The second example is the minimum-time robot arm reorientation problem taken from Dolan et al. [33]. This second example demonstrates how the method is able to handle multiple discontinuities in the control. The third example is the

hyper-sensitive optimal control problem taken from Rao and Mease [34] and Darby et al. [22]. This third example addresses how the method may need to be modified when explicit simulation fails in either forward or backward simulation. All three examples demonstrate, to various extents, the ability of the method to effectively converge to an optimal solution on a smaller mesh that satisfies the desired mesh tolerance compared to the methods of Patterson et al. [23], Liu et al. [24], and Liu et al. [25].

The mesh refinement method developed in this paper is hereafter referred to as the *ph*s method, where s refers to the user-specified numerical simulation scheme. When comparing against various methods, the terminology *ph*, *hp*, and *hp*-Legendre is adopted to refer to the methods of Patterson et al. [23], Liu et al. [24], and Liu et al. [25], respectively, implemented with the error estimate of Patterson et al. [23]. Moreover, the terminology *ph**, *hp**, and *hp*-Legendre* is adopted to refer to the methods of Patterson et al. [23], Liu et al. [24], and Liu et al. [25], respectively, implemented with the error estimate of Section 5.2. The previously mentioned methods all require additional parameters. For all methods, combinations of parameter values of $N_{\min} \in \{2, 3, 4\}$ and $N_{\max} \in \{6, 8, \ldots, 14\}$ are examined. Similar to Liu et al. [24], the *hp* and *hp** methods use a parameter value of $\bar{R} = 1.2$, where $\bar{R}$ is the threshold of significance for the second derivative ratio. Similar to Liu et al. [25], the *hp*-Legendre and *hp*-Legendre* methods use a parameter value of $\bar{\sigma} = 0.5$, where $\bar{\sigma}$ is the threshold of significance for the decay rate of the Legendre coefficients. For the *ph*s, *ph**, *hp**, and *hp*-Legendre* methods, the medium and high accuracy MATLAB ODE solvers s $\in \{$ode45, ode89$\}$ are used to solve the single- and multiple-interval IVPs and TVPs of Sections 5.1 and 5.3.3 and demonstrate how the performance of the *ph*s method is impacted. The MATLAB ODE solvers are employed using an accuracy tolerance of $\epsilon_{\text{ODE}} = 1 \times 10^{-6}$ (that is, the relative and absolute error accuracy tolerances when using an ODE solver are set to ϵ_{ODE}, where the error is controlled relative to the norm of the solution). Table 1 summarizes the mesh refinement methods considered in this work with their respective error estimates and variant parameters. Note that a method variant includes all necessary parameters (for example, *ph*s-$(2, 6,$ ode45$)$ refers to a *ph*s method variant with $N_{\min} = 2$, $N_{\max} = 6$, and s = ode45). For mesh size analysis, let the quantities N and K denote the total number of collocation points and mesh intervals, respectively. For error analysis, let the quantities $e_{\max}^{[m]}$ and $\tilde{e}_{\max}^{[m]}$ denote the maximum relative errors obtained on mesh m via the error estimation methods of Section 5.2 and Patterson et al. [23], respectively.

All results are obtained with the MATLAB general-purpose optimal control software GPOPS-II [35] using the NLP solver SNOPT [2]. SNOPT is employed with a tolerance of $\epsilon_{\text{NLP}} = 1 \times 10^{-8}$, where first derivatives are supplied via a sparse central difference scheme. For all methods, a mesh refinement tolerance of $\epsilon = 1 \times 10^{-6}$ and a maximum number of mesh refinement iterations of

Table 1: Summary of the mesh refinement methods considered in this work.

Method Name	Error Estimation Method	Mesh Refinement Algorithm
ph-$(N_{\min}, N_{\max})$	Patterson et al. [23]	Patterson et al. [23]
ph^*-$(N_{\min}, N_{\max}, \mathtt{s})$	Section 5.2	Patterson et al. [23]
hp	Patterson et al. [23]	Liu et al. [24]
hp^*-$(\mathtt{s})$	Section 5.2	Liu et al. [24]
hp-Legendre	Patterson et al. [23]	Liu et al. [25]
hp-Legendre*-$(\mathtt{s})$	Section 5.2	Liu et al. [25]
$ph\mathtt{s}$-$(N_{\min}, N_{\max}, \mathtt{s})$	Section 5.2	Section 5.3

$M_{\max} = 40$ are used. For all examples, the initial mesh consists of 10 uniformly spaced intervals with $N_{\min}$ collocation points per mesh interval, and the initial guess is a straight line for variables with boundary conditions at both endpoints and constant for variables with boundary conditions at only one endpoint. Finally, all computations are performed on a 12-core Apple M3 Pro MacBook Pro running macOS Sonoma version 14.6.1 with 36 GB LPDDR5 of unified memory using `MATLAB` version R2024a (build 24.1.0.2603908). All reported computational run times are 10-run averages of the total time spent within SNOPT and total execution time.

6.1 Example 1: Minimum-Time to Climb of a Supersonic Aircraft

Consider the following minimum-time supersonic aircraft climb optimal control problem taken from Darby et al. [22] using the dynamic model provided in Seywald et al. [32]. Determine the state, $\mathbf{x}(t) = [h(t), v(t), \gamma(t)]$, the control, $u(t) = n(t)$, and the terminal time, t_f, that minimize the objective functional

$$\mathcal{J} = t_f, \tag{37}$$

subject to the dynamic constraints

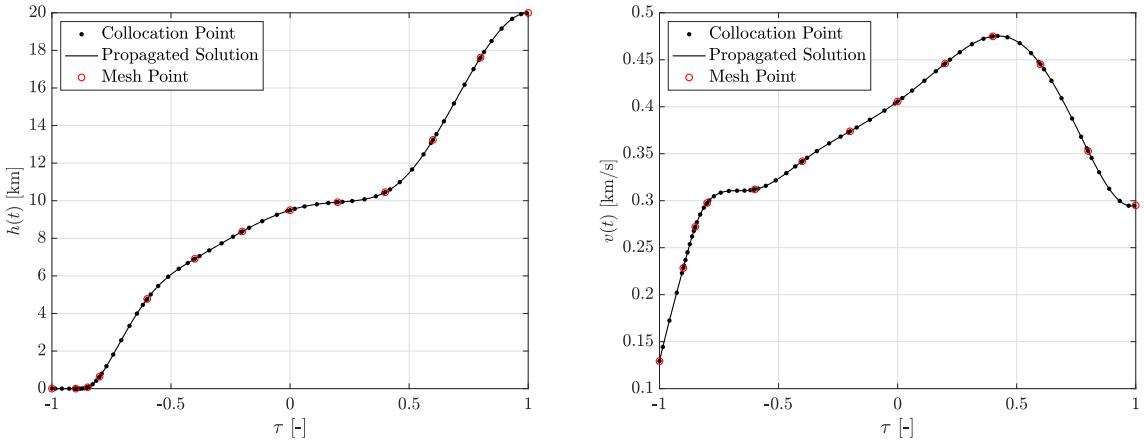
$$\dot{h} = v\sin(\gamma), \qquad \dot{v} = \frac{T - D}{m_v} - g\sin(\gamma), \qquad \dot{\gamma} = g\left(\frac{n - \cos(\gamma)}{v}\right), \tag{38}$$

the boundary conditions

$$\begin{aligned} h(0) &= 0, & v(0) &= 0.12931, & \gamma(0) &= 0, \\ h(t_f) &= 19.995, & v(t_f) &= 0.29509, & \gamma(t_f) &= 0, \end{aligned} \tag{39}$$

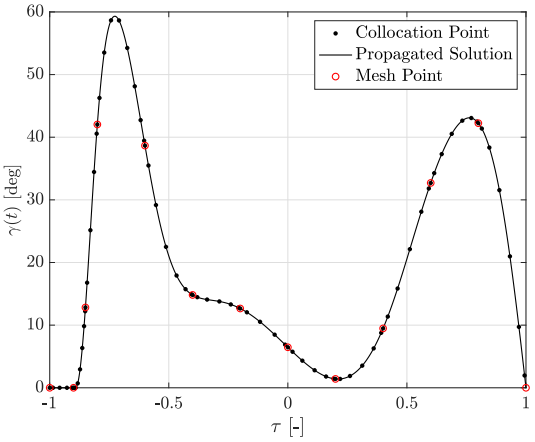
and the inequality path constraint

$$0 \le \gamma \le \frac{\pi}{2}, \tag{40}$$



(a) $h(t)$ vs. τ.

(b) $v(t)$ vs. τ.



(c) $\gamma(t)$ vs. τ.

Figure 1: Optimal state components for Example 1 using the phs-$(3, 10, \texttt{ode45})$ method.

where h is the altitude, v is the speed, γ is the flight path angle, u is the load factor, T is the thrust force, D is the drag force, m_v is the vehicle mass, and g is the acceleration due to gravity. A solution to this optimal control problem on $\tau \in [-1, +1]$ using the phs-$(3, 10, \texttt{ode45})$ method is shown in Figs. 1 and 2. It is noted that Figs. 1a–1c show the optimal state solutions obtained by solving the NLP of Eqs. (16)–(20) as well as the solutions obtained via explicit simulation using the MATLAB ODE solver `ode45`, whereas Fig. 2 shows the optimal control solution along with the interpolated control used in the explicit simulation. For this solution, an optimal final time of $t_f^* = 170.565775$ is obtained, and the corresponding mesh refinement history is shown in Fig. 3, where the phs-$(3, 10, \texttt{ode45})$ method takes five (5) iterations to satisfy the desired mesh tolerance. It is seen in Fig. 1c that the state constraint on the flight path angle as given in Eq. (40) is active from $\tau = -1$ to $\tau \approx -0.9$, and this active constraint arc is also seen in the altitude and load factor solutions of Figs. 1a and 2, respectively. To accurately capture this behavior, it is expected that

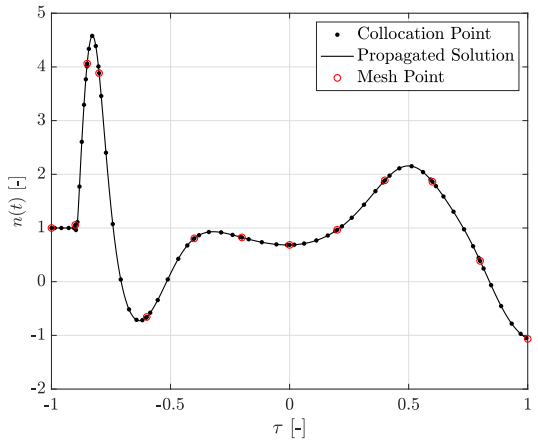


Figure 2: Optimal control, $n(t)$ vs. τ, for Example 1 using the phs-$(3, 10, \texttt{ode45})$ method.

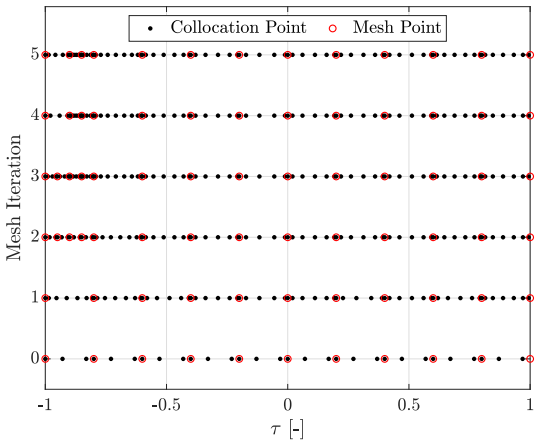


Figure 3: Mesh refinement history for Example 1 using the phs-$(3, 10, \texttt{ode45})$ method.

the phs method increases the density of the mesh near the activation and deactivation times of the path constraint. On the initial mesh ($m = 0$) of Fig. 3, the maximum error in the solution is $e_{\max}^{[0]} \approx 3 \times 10^{-3} > \epsilon$, which is largely due to a purposefully-naive initial mesh. Within the region of the active state constraint (that is, in $\tau \in [-1, -0.8]$), the first two mesh refinement iterations are driven by the error in the solution of the flight path angle. In this region in Fig. 3, the first mesh refinement iteration (that is, from mesh $m = 0$ to $m = 1$) adds collocation points to decrease the error in the solution, and during the second mesh refinement iteration (that is, from mesh $m = 1$ to $m = 2$), p refinement is exhausted. As a result, the mesh interval is split. Then, collocation points are appropriately added until the desired mesh tolerance is satisfied. For $\tau \in [-0.8, +1]$ on which the solution takes a smooth form (that is, when the state inequality constraint is inactive), the error in the solution is decreased by performing p refinement, and the mesh in $\tau \in [-0.6, +1]$ remains unchanged after the second mesh refinement iteration.

Next, the performance of the phs method is compared against the performance of the ph,

Table 2: Results for Example 1 using various mesh refinement methods.

$N_{\min}$	Method	N	K	M	$e_{\max}^{[m]}$	$\tilde{e}_{\max}^{[m]}$	Total Time [s] SNOPT	Total Time [s] CPU
2	None[a]	20	10	0	1.016×10^{-1}	9.693×10^{-2}	0.024	0.055
	hp[a]	107	21	4	1.228×10^{-6}	9.261×10^{-7}	0.365	0.532
	hp^*-(`ode45`)	114	22	4	6.195×10^{-7}	4.696×10^{-7}	0.411	0.978
	hp^*-(`ode89`)	114	22	4	6.193×10^{-7}	4.696×10^{-7}	0.403	1.840
	ph`s`-$(2, 10,$ `ode45`$)$	83	13	6	9.425×10^{-7}	7.140×10^{-7}	0.521	1.430
	ph`s`-$(2, 10,$ `ode89`$)$	83	13	6	9.114×10^{-7}	7.140×10^{-7}	0.527	2.883
3	None[a]	30	10	0	2.954×10^{-3}	2.570×10^{-3}	0.028	0.059
	hp[a]	103	16	4	1.200×10^{-6}	8.685×10^{-7}	0.388	0.557
	hp^*-(`ode45`)	99	14	5	5.655×10^{-7}	4.105×10^{-7}	0.478	1.037
	hp^*-(`ode89`)	99	14	5	5.655×10^{-7}	4.105×10^{-7}	0.474	1.809
	ph`s`-$(3, 10,$ `ode45`$)$	77	12	5	9.424×10^{-7}	7.429×10^{-7}	0.333	1.080
	ph`s`-$(3, 10,$ `ode89`$)$	77	12	5	9.462×10^{-7}	7.429×10^{-7}	0.349	2.381
4	None[a]	40	10	0	1.269×10^{-2}	1.010×10^{-2}	0.039	0.070
	ph-$(4, 12)$[a]	82	13	4	9.428×10^{-7}	7.395×10^{-7}	0.343	0.502
	ph^*-$(4, 12,$ `ode45`$)$	85	13	4	9.429×10^{-7}	7.395×10^{-7}	0.357	0.790
	ph^*-$(4, 12,$ `ode89`$)$	85	13	4	9.419×10^{-7}	7.395×10^{-7}	0.359	1.468
	ph`s`-$(4, 12,$ `ode45`$)$	77	12	5	9.425×10^{-7}	7.400×10^{-7}	0.375	1.049
	ph`s`-$(4, 12,$ `ode89`$)$	77	12	5	9.425×10^{-7}	7.400×10^{-7}	0.375	2.100

[a]The error estimate $e_{\max}^{[m]}$ of Section 5.2 is obtained with the `MATLAB` ODE solver `ode45`.

ph^*, hp, hp^*, hp-Legendre, and hp-Legendre* methods, and the error estimates obtained using the methods of Section 5.2 and Patterson et al. [23] are compared. Table 2 summarizes the results for Example 1 using various mesh refinement methods, where only the methods that obtain the smallest final mesh (that is, with the fewest total number of collocation points) are shown. In terms of final mesh size, the ph`s` method converges to the smallest mesh for every considered value of $N_{\min}$ when compared against that obtained using previously developed methods, where a 22%, 22%, and 6% decrease in total number of collocation points and a 38%, 14%, and 8% decrease in total number of

mesh intervals are observed for $N_{\min} = [2, 3, 4]$, respectively. The decrease in mesh size, however, typically requires the *ph*s method to perform more mesh refinement iterations to satisfy the desired mesh tolerance. The *ph*s method also requires the use of an explicit simulation scheme, which is typically more computationally expensive than the Gaussian quadrature integration scheme utilized in the *ph*, *hp*, and *hp*-Legendre methods. As a result, expected increases in total CPU times are observed when using the *ph*s method. In terms of error analysis, the error estimate $e_{\max}^{[M]}$ obtained in Section 5.2 is larger than the error estimate $\tilde{e}_{\max}^{[M]}$ obtained via the method of Patterson et al. [23] for all results shown in Table 2. For $N_{\min} = [2, 3]$, the final meshes obtained using the *hp* method do not satisfy $e_{\max}^{[M]} \leq \epsilon$, which suggests there is still a discrepancy between solutions obtained via collocation and time-marching schemes on these final meshes. Similar phenomena are observed in many other cases for Example 1 when using the *ph*, *hp*, and *hp*-Legendre methods but are not shown in Table 2. On the other hand, all final meshes obtained using the *ph*s method satisfy both $e_{\max}^{[M]} \leq \epsilon$ and $\tilde{e}_{\max}^{[M]} \leq \epsilon$, where the solutions obtained on the final mesh via collocation and time-marching are guaranteed to be in agreement with one another. Finally, the choice of `MATLAB` ODE solver did not impact the size of the final mesh for this example for all values of $N_{\min}$ considered, as shown in Table 2; however, using the `MATLAB` ODE solver `ode89` required larger computational times than `ode45`. To summarize for Example 1, the *ph*s mesh refinement method is able to outperform the *ph*, *ph**, *hp*, *hp**, *hp*-Legendre, and *hp*-Legendre* mesh refinement methods in terms of final mesh size.

6.2 Example 2: Minimum-Time Reorientation of a Robot Arm

Consider the following minimum-time robot arm reorientation optimal control problem taken from Dolan et al. [33]. Determine the state, $\mathbf{x}(t) = [\rho(t), \theta(t), \phi(t), \dot{\rho}(t), \dot{\theta}(t), \dot{\phi}(t)]$, the control, $\mathbf{u}(t) = [u_\rho(t), u_\theta(t), u_\phi(t)]$, and the terminal time, t_f, that minimize the objective functional

$$\mathcal{J} = t_f, \tag{41}$$

subject to the dynamic constraints

$$\ddot{\rho} = \frac{u_\rho}{L}, \qquad \ddot{\theta} = \frac{u_\theta}{I_\phi \sin^2(\phi)}, \qquad \ddot{\phi} = \frac{u_\phi}{I_\phi}, \tag{42}$$

the boundary conditions

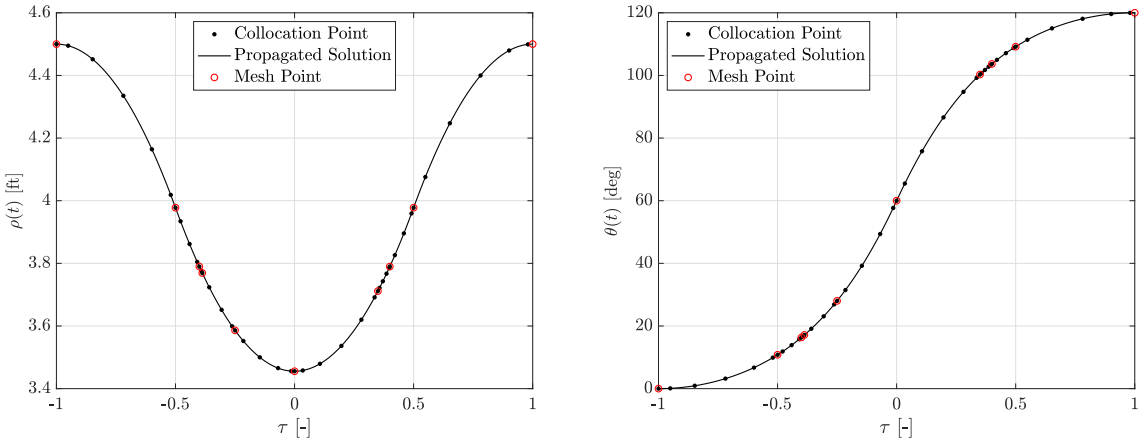
$$\begin{aligned} &\rho(0) = \frac{9}{2}, \quad \theta(0) = 0, \quad \phi(0) = \frac{\pi}{4}, \quad \dot{\rho}(0) = 0, \quad \dot{\theta}(0) = 0, \quad \dot{\phi}(0) = 0, \\ &\rho(t_f) = \frac{9}{2}, \quad \theta(t_f) = \frac{2\pi}{3}, \quad \phi(t_f) = \frac{\pi}{4}, \quad \dot{\rho}(t_f) = 0, \quad \dot{\theta}(t_f) = 0, \quad \dot{\phi}(t_f) = 0, \end{aligned} \tag{43}$$

and the inequality path constraints

$$\begin{aligned} 0 \leq \rho \leq L, \qquad -\pi \leq \theta \leq \pi, \qquad 0 \leq \phi \leq \pi, \\ -1 \leq u_\rho \leq 1, \qquad -1 \leq u_\theta \leq 1, \qquad -1 \leq u_\phi \leq 1, \end{aligned} \tag{44}$$

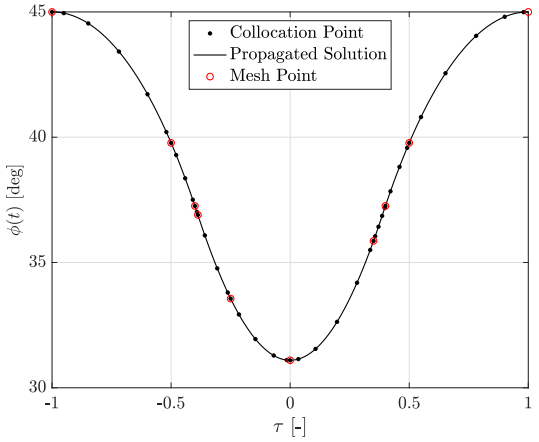
where $L = 5$ and $I_\phi \equiv I_\phi(\rho) = ((L-\rho)^3 + \rho^3)/3$. A solution to this optimal control problem on $\tau \in [-1, +1]$ using the phs-$(2, 6, \texttt{ode45})$ method is shown in Figs. 4 and 5. It is noted that Figs. 4a–4f show the optimal state solutions obtained by solving the NLP of Eqs. (16)–(20) as well as the solutions obtained via explicit simulation using the MATLAB ODE solver ode45, whereas Figs. 5a–5c show the optimal control solutions along with the interpolated controls used in the explicit simulation. For this solution, an optimal final time of $t_f^* = 9.140963$ is obtained, and the corresponding mesh refinement history is shown in Fig. 6, where the phs-$(2, 6, \texttt{ode45})$ method takes seven (7) iterations to satisfy the desired mesh tolerance. It is seen in Fig. 5 that the optimal control components all take on a bang-bang structure, where it is known that the five control discontinuities are located at $\tau \approx \{-0.5, -0.3882, 0, +0.3882, +0.5\}$. These discontinuity locations also correspond to the local extrema in the respective state components $\dot{\rho}$, $\dot{\theta}$, and $\dot{\phi}$ shown in Figs. 4d–4f, respectively. Because of the rapid changes in the solution, it is expected that the phs method increases the density of the mesh near the locations of the control discontinuities. On the initial mesh ($m = 0$) of Fig. 6, the maximum error in the solution is $e_{\max}^{[0]} \approx 1.5 \times 10^{-3} > \epsilon$, which is largely due to an initial mesh that does not accurately capture all of the discontinuity locations; however, due to the construction of the initial mesh consisting of 10 uniformly spaced intervals, the discontinuity in the control component u_θ at $\tau = 0$ is already identified. Mesh refinement is initially driven by the error in the solutions of ϕ and $\dot{\phi}$; however, p refinement is quickly exhausted in $\tau \in [-0.6, -0.2]$ and $\tau \in [+0.2, +0.6]$ during the second mesh refinement iteration (that is, from mesh $m = 1$ to $m = 2$) due to large state error estimates from the remaining inaccurately captured discontinuities. As a result, the appropriate intervals are split, and the discontinuities in the control component u_ρ at $\tau = \{-0.5, +0.5\}$ are then identified. It is important to note that the h refinement step of Section 5.3.2 does not perform any discontinuity detection; thus, new intervals may be formed in which discontinuities are not present. The h reduction step of Section 5.3.3 then helps remove unnecessary mesh intervals during the fourth and fifth mesh refinement iterations (that is, from mesh $m = 3$ to $m = 4$ and from $m = 4$ to $m = 5$, respectively) and, as a result, keeps the size of the mesh small. Finally, appropriate regions of the solution are further refined until the remaining two discontinuities in the control component u_ϕ at $\tau \approx \{-0.3882, +0.3882\}$ are accurately identified, and the desired mesh tolerance is satisfied.

Next, the performance of the various methods and error estimates is compared in a manner

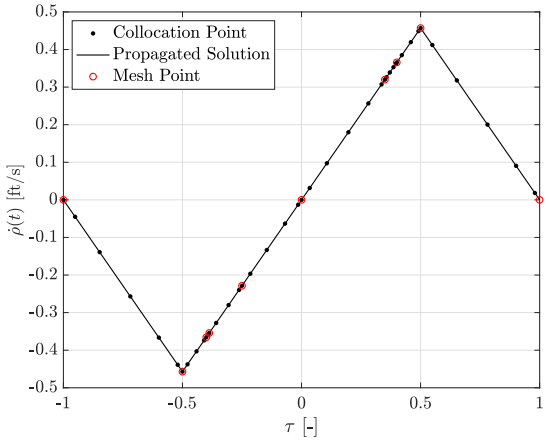


(a) $\rho(t)$ vs. τ.

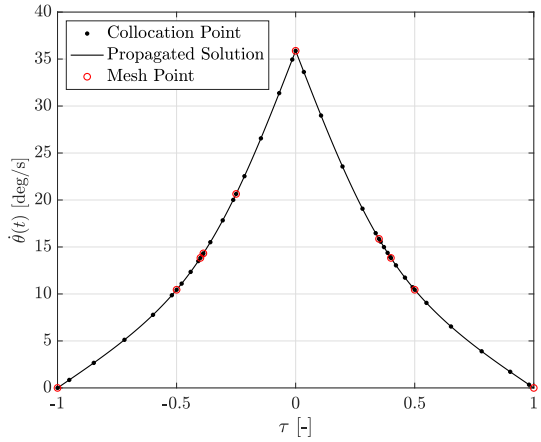
(b) $\theta(t)$ vs. τ.



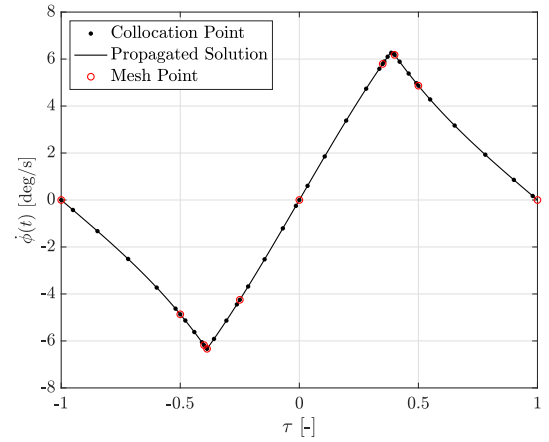
(c) $\phi(t)$ vs. τ.



(d) $\dot{\rho}(t)$ vs. τ.



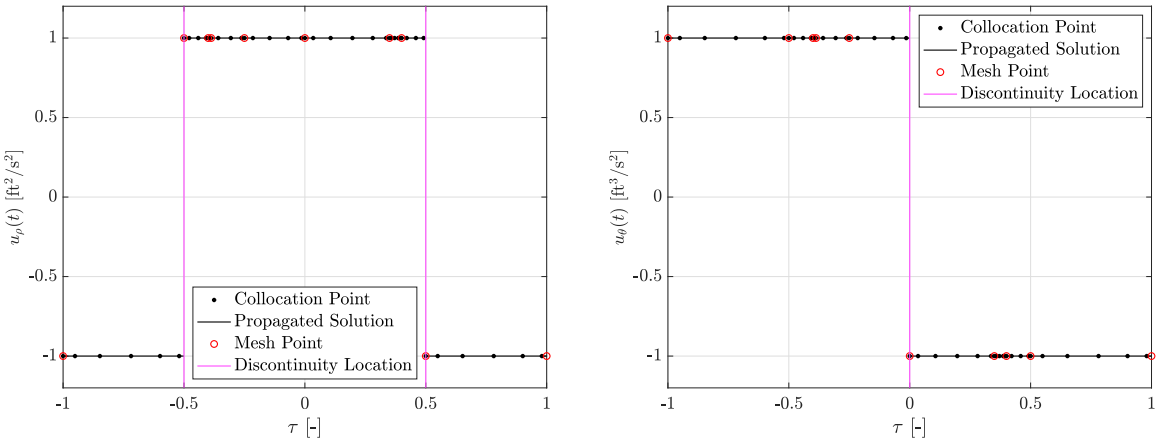
(e) $\dot{\theta}(t)$ vs. τ.



(f) $\dot{\phi}(t)$ vs. τ.

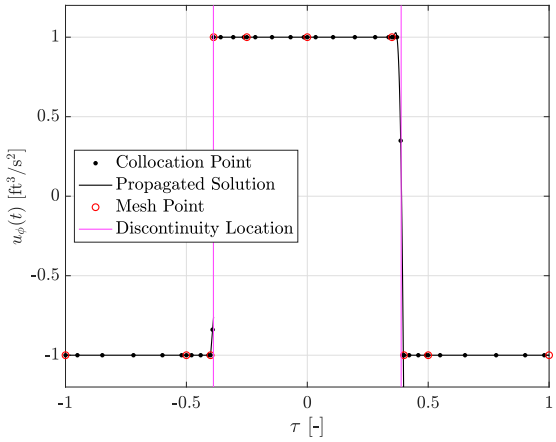
Figure 4: Optimal state components for Example 2 using the phs-$(2, 6, \texttt{ode45})$ method.

similar to that done for Example 1 in Section 6.1. Table 3 summarizes the results for Example 2, where only the methods that obtain the smallest final mesh are shown. In terms of final mesh size, the phs method converges to the smallest mesh for every considered value of $N_{\min}$ when



(a) $u_\rho(t)$ vs. τ.

(b) $u_\theta(t)$ vs. τ.



(c) $u_\phi(t)$ vs. τ.

Figure 5: Optimal control components for Example 2 using the *ph*s-$(2, 6,$ `ode45`$)$ method.

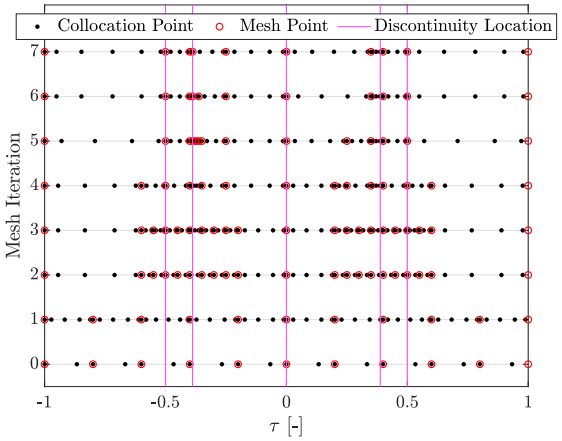


Figure 6: Mesh refinement history for Example 2 using the *ph*s-$(2, 6,$ `ode45`$)$ method.

compared against previously developed methods, where a 50%, 30%, and 38% decrease in total number of collocation points and a 47%, 42%, and 47% decrease in total number of mesh intervals

Table 3: Results for Example 2 using various mesh refinement methods.

$N_{\min}$	Method	N	K	M	$e_{\max}^{[m]}$	$\tilde{e}_{\max}^{[m]}$	Total Time [s] SNOPT	Total Time [s] CPU
2	None[a]	20	10	0	1.505×10^{-3}	1.125×10^{-3}	0.018	0.049
	hp[a]	84	17	4	2.479×10^{-7}	1.894×10^{-7}	0.140	0.307
	hp^*-(`ode45`)	84	17	4	2.479×10^{-7}	1.894×10^{-7}	0.139	0.508
	hp^*-(`ode89`)	84	17	4	2.468×10^{-7}	1.894×10^{-7}	0.138	0.992
	phs-$(2, 6,$ `ode45`$)$	42	9	7	6.615×10^{-7}	5.197×10^{-7}	2.806	3.432
	phs-$(2, 6,$ `ode89`$)$	42	9	7	6.671×10^{-7}	5.197×10^{-7}	2.823	4.310
3	None[a]	30	10	0	1.902×10^{-4}	1.690×10^{-4}	0.020	0.052
	ph-$(3, 10)$[a]	81	19	6	2.250×10^{-6}	3.498×10^{-7}	0.174	0.398
	ph^*-$(3, 8,$ `ode45`$)$	84	20	4	4.810×10^{-7}	3.770×10^{-7}	0.108	0.502
	ph^*-$(3, 8,$ `ode89`$)$	84	20	4	4.808×10^{-7}	3.770×10^{-7}	0.109	1.081
	phs-$(3, 10,$ `ode45`$)$	47	9	9	6.615×10^{-7}	5.197×10^{-7}	0.166	0.989
	phs-$(3, 10,$ `ode89`$)$	57	11	7	9.017×10^{-7}	6.845×10^{-7}	0.144	1.449
4	None[a]	40	10	0	1.661×10^{-4}	1.343×10^{-4}	0.024	0.057
	ph-$(4, 6)$[a]	71	17	6	1.193×10^{-6}	9.174×10^{-7}	0.111	0.332
	ph^*-$(4, 6,$ `ode45`$)$	72	17	7	7.536×10^{-7}	6.368×10^{-7}	0.129	0.746
	ph^*-$(4, 6,$ `ode89`$)$	72	17	7	7.555×10^{-7}	6.368×10^{-7}	0.130	1.595
	phs-$(4, 6,$ `ode45`$)$	44	9	5	9.420×10^{-7}	7.175×10^{-7}	0.090	0.525
	phs-$(4, 6,$ `ode89`$)$	44	9	5	9.420×10^{-7}	7.175×10^{-7}	0.092	1.033

[a]The error estimate $e_{\max}^{[m]}$ of Section 5.2 is obtained with the `MATLAB` ODE solver `ode45`.

are observed for $N_{\min} = [2, 3, 4]$, respectively. Again, the decrease in mesh size typically requires the phs method to perform more mesh refinement iterations to satisfy the desired mesh tolerance; however, the case for $N_{\min} = 4$ demonstrates that the phs method has the potential to also outperform previously developed methods in terms of computational efficiency. This example also reiterates trends observed in Section 6.1, specifically where the phs method satisfies $e_{\max}^{[M]} \leq \epsilon$ and $\tilde{e}_{\max}^{[M]} \leq \epsilon$ for every case shown in Table 3, but the ph, hp, and hp-Legendre methods are unable to guarantee satisfaction of $e_{\max}^{[M]} \leq \epsilon$. For $N_{\min} = 3$, performance of the phs method is impacted by the

choice of MATLAB ODE solver, where the use of ode89 requires a slightly larger final mesh compared to ode45. To summarize for Example 2, the *ph*s mesh refinement method again demonstrates a significant improvement over the *ph*, *ph**, *hp*, *hp**, *hp*-Legendre, and *hp*-Legendre* mesh refinement methods in terms of obtaining a smaller final mesh that satisfies the desired mesh tolerance.

6.3 Example 3: Hyper-Sensitive Problem

Consider the following variation of the hyper-sensitive optimal control problem taken from Rao and Mease [34] and Darby et al. [22]. Determine the state, $x(t)$, and the control, $u(t)$, that minimize the objective functional

$$\mathcal{J} = \frac{1}{2}\int_0^{t_f} (x^2 + u^2)\mathrm{d}t, \tag{45}$$

subject to the dynamic constraints

$$\dot{x} = -x^3 + u, \tag{46}$$

and the boundary conditions

$$x(0) = 1.5, \qquad x(t_f) = 1, \tag{47}$$

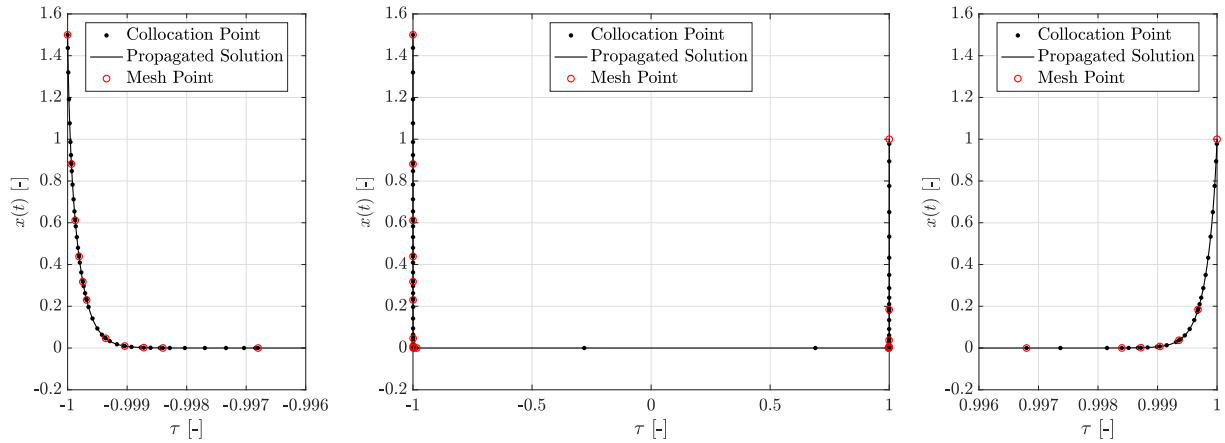
where a final time of $t_f = 10,000$ is used. For sufficiently large values of t_f, the resulting Hamiltonian boundary-value problem (HBVP) (that is, the HBVP obtained from deriving the first-order necessary conditions for optimality) is completely *hyper-sensitive* and the solution exhibits a *take-off*, *cruise*, and *landing* structure [34]. It is noted that the control is essentially zero during the cruise segment because the cruise segment lies in the neighborhood of an equilibrium point.

It is known that numerical difficulties often arise when obtaining solutions to the optimal control problem of Eqs. (45)–(47) using an explicit simulation method (for example, indirect or direct shooting) due to instabilities in the dynamics in either forward or backward time. Because the error estimate of Section 5.2 employs explicit simulation, it, too, may be prone to numerical difficulties similar to that of indirect or direct shooting. Furthermore, the optimal control problem of Eqs. (45)–(47) is considered *stiff* because the dynamics evolve on a time-scale that is significantly smaller than the horizon over which the problem is being solved [36]. In the case of the hyper-sensitive problem, the time-scale of the dynamics is approximately unity while the horizon is 10,000 time units. Consequently, the time horizon in this example is significantly larger than the time-scale of the dynamics, which makes this problem stiff from the perspective of optimal control. In this example, the objective is to highlight the impact of the initial mesh and ODE solver on the ability to employ the *ph*s method due to the reliance of the error estimate on explicit simulation.

To demonstrate the potential issues with the reliance of the error estimate of the *ph*s method on explicit simulation, consider the explicit simulation of the uncontrolled dynamics $\dot{x} = -x^3$. Using

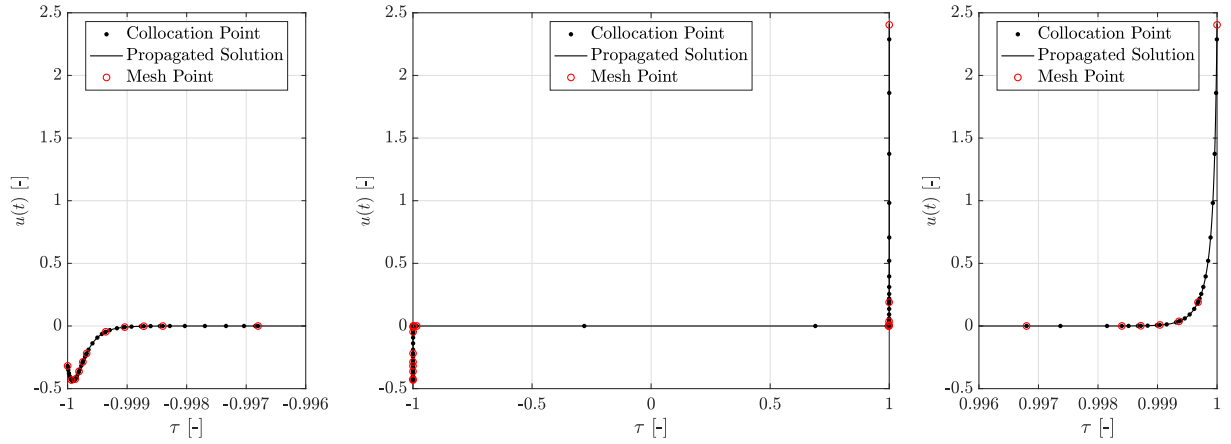
any terminal condition perturbed significantly from $x = 0$ to solve the TVP of Eq. (22), every MATLAB ODE solver fails in backward time because $\dot{x} = -x^3$ is unstable in backward time. As a workaround to the inability to perform explicit simulation in backward time on this example, the contribution to the maximum relative error estimate in Eq. (26) from the single-interval TVP of Eq. (22) is ignored in the mesh refinement method of Algorithm 1 for this example. A similar approach is applied to the h reduction step of Section 5.3.3, where the contribution to the maximum relative error estimate in Eq. (34) from the multiple-interval TVP of Eq. (32) is ignored. While it is not possible to include the contribution to the maximum relative error estimate from the solution of the TVP for this example, accurate error estimates are still obtained in each mesh interval by solving the single- and multiple-interval IVPs of Eqs. (21) and (31). Finally, it is noted that same approaches are utilized for this example when obtaining the error estimates using the ph^*, hp^*, and hp-Legendre* methods.

Using the phs-$(3, 12, \texttt{ode89})$ method, a solution to this optimal control problem on $\tau \in [-1, +1]$ is shown in Figs. 7 and 8. It is noted that Figs. 7a–7c show the optimal state solution obtained by solving the NLP of Eqs. (16)–(20) as well as the solution obtained via explicit simulation using the MATLAB ODE solver ode89, whereas Figs. 8a–8c show the optimal control solution along with the interpolated control used in the explicit simulation. For this solution, an optimal objective value of $\mathcal{J}^* = 1.330806$ is obtained, and the corresponding mesh refinement history is shown in Fig. 9, where the phs-$(3, 12, \texttt{ode89})$ method takes thirteen (13) iterations to satisfy the desired mesh tolerance. It is seen that the optimal state, shown in Fig. 7, attains the aforementioned take-off, cruise, and landing structure, which is reflected in the optimal control solution of Fig. 8. Because of the rapid initial decay in the take-off region and rapid terminal growth in the landing region, it is expected



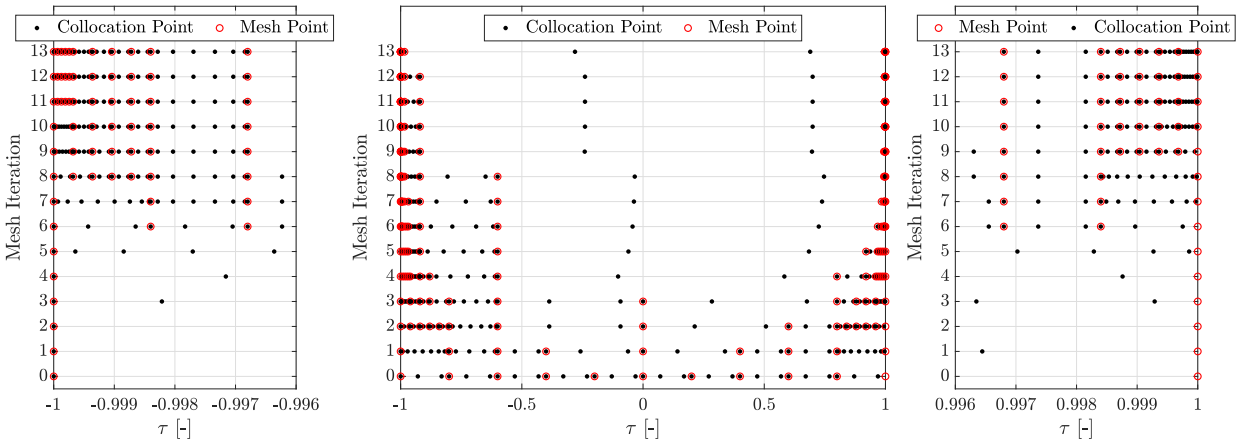
(a) Near $\tau = -1$. (b) Over $\tau \in [-1, +1]$. (c) Near $\tau = +1$.

Figure 7: Optimal state, $x(t)$ vs. τ, for Example 3 using the phs-$(3, 12, \texttt{ode89})$ method.



(a) Near $\tau = -1$. (b) Over $\tau \in [-1, +1]$. (c) Near $\tau = +1$.

Figure 8: Optimal control, $u(t)$ vs. τ, for Example 3 using the phs-$(3, 12, \texttt{ode89})$ method.



(a) Near $\tau = -1$. (b) Over $\tau \in [-1, +1]$. (c) Near $\tau = +1$.

Figure 9: Mesh refinement history for Example 3 using the phs-$(3, 12, \texttt{ode89})$ method.

that the phs method increases the density of the mesh in these regions to adequately capture the changes in the solution. As seen in Fig. 9a and 9c, the phs method is able to identify such regions and concentrate both mesh and collocation points in the regions near $\tau = -1$ and $\tau = +1$. In the constant portion of the solution in Fig. 9b (that is, not near $\tau = -1$ or $\tau = +1$), the phs method consecutively merges mesh intervals in order to reduce the size of the mesh in the interior region of the solution, while retaining an increased mesh density in the initial and terminal regions.

Next, the performance of the various methods and error estimates is compared in a manner similar to that done for Examples 1 and 2 in Sections 6.1 and 6.2, respectively. Table 4 summarizes the results for Example 3, where only the methods that obtain the smallest final mesh are shown. This example again reiterates trends observed in Sections 6.1 and 6.2. First, the phs method

Table 4: Results for Example 3 using various mesh refinement methods.

$N_{\min}$	Method	N	K	M	$e_{\max}^{[m]}$	$\tilde{e}_{\max}^{[m]}$	Total Time [s] SNOPT	Total Time [s] CPU
2	None[a]	20	10	0	4.787×10^{-1}	2.796×10^{2}	0.038	0.070
	hp[a]	155	32	6	1.164×10^{-6}	8.560×10^{-7}	0.466	0.721
	*hp**-(`ode45`)	172	38	7	2.931×10^{-7}	2.221×10^{-7}	0.673	1.578
	*hp**-(`ode89`)	172	38	7	2.931×10^{-7}	2.221×10^{-7}	0.669	2.640
	phs-$(2, 10,$ `ode45`$)$	93	24	11	7.804×10^{-7}	6.146×10^{-7}	0.470	2.141
	phs-$(2, 10,$ `ode89`$)$	93	24	11	7.798×10^{-7}	6.146×10^{-7}	0.480	4.442
3	None[a]	30	10	0	4.196×10^{-1}	9.570×10^{1}	0.031	0.061
	hp-Legendre[a]	101	25	6	8.952×10^{-7}	7.341×10^{-7}	0.979	1.218
	hp-Legendre*-(`ode45`)	100	24	6	7.444×10^{-7}	6.149×10^{-7}	0.327	0.925
	hp-Legendre*-(`ode89`)	100	24	6	7.484×10^{-7}	6.149×10^{-7}	0.324	1.563
	phs-$(3, 8,$ `ode45`$)$	91	21	12	8.320×10^{-7}	6.825×10^{-7}	0.606	1.765
	phs-$(3, 12,$ `ode89`$)$	87	18	13	7.545×10^{-7}	6.052×10^{-7}	0.572	5.022
4	None[a]	40	10	0	3.826×10^{-1}	4.742×10^{1}	0.058	0.088
	hp-Legendre[a]	107	22	6	7.925×10^{-7}	5.506×10^{-7}	0.400	0.637
	hp-Legendre*-(`ode45`)	109	23	7	8.082×10^{-7}	6.432×10^{-7}	0.490	1.162
	hp-Legendre*-(`ode89`)	109	23	7	8.083×10^{-7}	6.432×10^{-7}	0.492	1.853
	phs-$(4, 8,$ `ode45`$)$	94	20	10	8.703×10^{-7}	7.365×10^{-7}	0.772	1.795
	phs-$(4, 10,$ `ode89`$)$	96	20	15	8.316×10^{-7}	6.893×10^{-7}	0.709	2.960

[a]The error estimate $e_{\max}^{[m]}$ of Section 5.2 is obtained with the `MATLAB` ODE solver `ode45`.

converges to the smallest mesh for every considered value of $N_{\min}$ when compared against previously developed methods, where a 40%, 9%, and 10% decrease in total number of collocation points and a 25%, 13%, and 9% decrease in total number of mesh intervals are observed for $N_{\min} = [2, 3, 4]$, respectively. The *phs* method typically performs more mesh refinement iterations to satisfy the desired mesh tolerance, where the choice of `MATLAB` ODE solver impacts the performance of the *phs* method for $N_{\min} = [3, 4]$. Similar to the previous two examples, the error estimate $e_{\max}^{[M]}$ obtained using the method in Section 5.2 is larger than the error estimate $\tilde{e}_{\max}^{[M]}$ obtained using the

method of Patterson et al. [23] on the converged mesh for all results shown in Table 4, where less than an order of magnitude difference between the two error estimates is observed. On the other hand, the opposite is true on the initial mesh for this example, where a difference of roughly two to three orders of magnitude is observed. Through this, the error estimate of Patterson et al. [23] suggests that there is a much larger discrepancy between solutions obtained via collocation and explicit simulation than actually present, which guides the mesh refinement methods to increase the size of the mesh more than necessary in the appropriate regions. When using the error estimate of Section 5.2 in the *ph*s, *ph**, *hp**, and *hp*-Legendre* mesh refinement methods, explicit simulation of the dynamics of Eq. (46) in forward time does not fail at any point along the solution on any mesh refinement iteration when using either `ode45` or `ode89`. To summarize for Example 3, the *ph*s mesh refinement method again demonstrates an improvement over the *ph*, *ph**, *hp*, *hp**, *hp*-Legendre, and *hp*-Legendre* mesh refinement methods in terms of obtaining a smaller final mesh that satisfies the desired mesh tolerance.

7 Discussion

Each of the examples in Section 6 illustrates different features of the *ph*s mesh refinement method developed in Section 5 as well as highlights the performance of the novel error estimate derived in Section 5.2. The first example demonstrates the method on smooth state and control profiles, while converging to the desired mesh tolerance in a similar number of mesh refinement iterations when compared to previously developed methods. The second example shows how the method is able to handle multiple discontinuities in the control profile. The third example illustrates how the method handles stiff systems, specifically when explicit simulation fails in forward and/or backward time, and discusses potential workarounds to such issues. Finally, all examples demonstrate how the *ph*s method is able to converge to a smaller mesh that satisfies the desired mesh tolerance when compared against previously developed methods.

As with most computational methods, some limitations and trade-offs exist with the mesh refinement method developed in Section 5. Because the proposed method assumes the successful explicit simulation of the problem dynamics in forward and backward time, any failure of the explicit simulation scheme can limit the capabilities of the method. As demonstrated by the example in Section 6.3, the stiff system is unstable in backward time, which causes the explicit simulation scheme to fail. While explicit simulation in forward time is successful and the method is able to proceed nominally, this phenomena should be considered on a problem by problem basis. The most apparent trade-off that exists when using the *ph*s method is between mesh size and computational

time. All three examples presented in Section 6 demonstrate how the *ph*s method converges to a smaller mesh when compared to previously developed methods but often at the expense of a larger computational time; however, avenues to increase computational efficiency exist. From a software perspective, the current MATLAB implementation performs error estimate computations in chronological order (that is, mesh interval by mesh interval), as shown by Algorithm 1; however, parallelizing the process and/or using C/C++ could significantly reduce the computational burden. Similar avenues can be implemented for the refinement and reduction steps of the proposed method. Furthermore, any hardware upgrade could also increase computational efficiency (for example, by increasing the number of cores). Such improvements to the proposed method would be necessary for feasibility in real-time or near real-time applications.

8 Conclusions

An adaptive mesh refinement method for solving optimal control problems using Legendre-Gauss-Radau direct collocation has been described. In regions of the solution where the desired accuracy tolerance is not satisfied, the method increases the degree of the polynomial approximation within a mesh interval and/or the number of mesh intervals. The method also allows for mesh size reduction in regions of the solution where the desired accuracy tolerance is satisfied by either merging adjacent mesh intervals or decreasing the degree of the polynomial approximation within a mesh interval. In addition, the mesh refinement process is driven by a new relative error estimate in the state solution that is based on the differences between the Lagrange polynomial approximation of the state obtained via collocation and forward and backward explicit simulations of the dynamics in each mesh interval. Using the newly developed error estimate and mesh refinement process, the solutions obtained via collocation and explicit simulation schemes are guaranteed to be in agreement with one other on the final mesh. The method is demonstrated to three examples from the open literature that highlight various features of the method, where the method is able to handle an active state path constraint and multiple control discontinuities in the solution while the issues associated with failure of the explicit simulation scheme are addressed. The results of this research show that the size of the final mesh is smaller when compared with previously developed mesh refinement methods.

Acknowledgements

The authors gratefully acknowledge support for this research from the U.S. National Science Foundation under Grant CMMI-2031213 and the NASA Florida Space Grant Consortium under Grant 80NSSC20M0093.

References

[1] J. T. Betts. *Practical Methods for Optimal Control Using Nonlinear Programming*. SIAM Press, Philadelphia, PA, USA, 3rd edition, 2020. doi:10.1137/1.9781611976199.

[2] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131, January 2005. doi:10.1137/S0036144504446096.

[3] L. T. Biegler and V. M. Zavala. Large-Scale Nonlinear Programming Using IPOPT: An Integrating Framework for Enterprise-Wide Optimization. *Computers & Chemical Engineering*, 33(3):575–582, March 2009. doi:10.1016/j.compchemeng.2008.08.006.

[4] Y. Zhao and P. Tsiotras. Density Functions for Mesh Refinement in Numerical Optimal Control. *Journal of Guidance, Control, and Dynamics*, 34(1):271–277, January–February 2011. doi:10.2514/1.45852.

[5] Q. Gong, F. Fahroo, and I. M. Ross. Spectral Algorithm for Pseudospectral Methods in Optimal Control. *Journal of Guidance, Control, and Dynamics*, 31(3):460–471, May–June 2008. doi:10.2514/1.32908.

[6] W. Gui and I. Babuška. The h, p, and h-p Versions of the Finite Element Method in 1 Dimension. Part I. The Error Analysis of the p Version. *Numerische Mathematik*, 49:577–612, November 1986. doi:10.1007/BF01389733.

[7] W. Gui and I. Babuška. The h, p, and h-p Versions of the Finite Element Method in 1 Dimension. Part II. The Error Analysis of the h- and h-p Versions. *Numerische Mathematik*, 49:613–657, November 1986. doi:10.1007/BF01389734.

[8] W. Gui and I. Babuška. The h, p, and h-p Versions of the Finite Element Method in 1 Dimension. Part III. The Adaptive h-p Version. *Numerische Mathematik*, 49:659–683, November 1986. doi:10.1007/BF01389735.

[9] I. Babuška and M. Suri. The p- and h-p Versions of the Finite Element Method, An Overview. *Computer Methods in Applied Mechanics and Engineering*, 80(1–3):5–26, June 1990. doi:10.1016/0045-7825(90)90011-A.

[10] I. Babuška and M. Suri. The p- and h-p Versions of the Finite Element Method, Basic Principles and Properties. *SIAM Review*, 36(4):578–632, December 1994. doi:10.1137/1036141.

[11] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao. Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method. *Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, November–December 2006. doi:10.2514/1.20478.

[12] D. Garg, M. A. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington. A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods. *Automatica*, 46(11):1843–1851, November 2010. doi:10.1016/j.automatica.2010.06.048.

[13] D. Garg, W. W. Hager, and A. V. Rao. Pseudospectral Methods for Solving Infinite-Horizon Optimal Control Problems. *Automatica*, 47(4):829–837, April 2011. doi:10.1016/j.automatica.2011.01.085.

[14] D. Garg, M. A. Patterson, C. L. Darby, C. Froncolin, G. T. Huntington, W. W. Hager, and A. V. Rao. Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method. *Computational Optimization and Applications*, 49(2):355–358, June 2011. doi:10.1007/s10589-009-9291-0.

[15] G. Elnagar, M. A. Kazemi, and R. Razzaghi. The Pseudospectral Legendre Method for Discretizing Optimal Control Problems. *IEEE Transactions on Automatic Control*, 40(10):1793–1796, October 1995. doi:10.1109/9.467672.

[16] F. Fahroo and I. M. Ross. Costate Estimation by a Legendre Pseudospectral Method. *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, March–April 2001. doi:10.2514/2.4709.

[17] W. W. Hager, H. Hou, and A. V. Rao. Convergence Rate for a Gauss Collocation Method Applied to Unconstrained Optimal Control. *Journal of Optimization Theory and Applications*, 169(3):801–824, June 2016. doi:10.1007/s10957-016-0929-7.

[18] W. W. Hager, H. Hou, and A. V. Rao. Lebesgue Constants Arising in a Class of Collocation Methods. *IMA Journal of Numerical Analysis*, 37(4):1884–1901, October 2017. doi:10.1093/imanum/drw060.

[19] W. W. Hager, J. Liu, S. Mohapatra, A. V. Rao, and X.-S. Wang. Convergence Rate for a Gauss Collocation Method Applied to Constrained Optimal Control. *SIAM Journal on Control and Optimization*, 56(2):1386–1411, January 2018. doi:10.1137/16M1096761.

[20] W. W. Hager, H. Hou, S. Mohapatra, A. V. Rao, and X.-S. Wang. Convergence Rate for a Radau hp Collocation Method Applied to Constrained Optimal Control. *Computational Optimization and Applications*, 74(1):275–314, May 2019. doi:10.1007/s10589-019-00100-1.

[21] C. L. Darby, W. W. Hager, and A. V. Rao. An hp-Adaptive Pseudospectral Method for Solving Optimal Control Problems. *Optimal Control Applications and Methods*, 32(4):476–502, July–August 2011. doi:10.1002/oca.957.

[22] C. L. Darby, W. W. Hager, and A. V. Rao. Direct Trajectory Optimization Using a Variable Low-Order Adaptive Pseudospectral Method. *Journal of Spacecraft and Rockets*, 48(3):433–445, May–June 2011. doi:10.2514/1.52136.

[23] M. A. Patterson, W. W. Hager, and A. V. Rao. A *ph* Mesh Refinement Method for Optimal Control. *Optimal Control Applications and Methods*, 36(4):398–421, July–August 2015. doi:10.1002/oca.2114.

[24] F. Liu, W. W. Hager, and A. V. Rao. Adaptive Mesh Refinement Method for Optimal Control Using Nonsmoothness Detection and Mesh Size Reduction. *Journal of the Franklin Institute*, 352(10):4081–4106, October 2015. doi:10.1016/j.jfranklin.2015.05.028.

[25] F. Liu, W. W. Hager, and A. V. Rao. Adaptive Mesh Refinement Method for Optimal Control Using Decay Rates of Legendre Polynomial Coefficients. *IEEE Transactions on Control Systems Technology*, 26(4):1475–1483, July 2018. doi:10.1109/TCST.2017.2702122.

[26] A. T. Miller, W. W. Hager, and A. V. Rao. Mesh Refinement Method for Solving Optimal Control Problems with Nonsmooth Solutions Using Jump Function Approximations. *Optimal Control Applications and Methods*, 42(4):1119–1140, July–August 2021. doi:10.1002/oca.2719.

[27] G. V. Haman III and A. V. Rao. Leveraging a Mesh Refinement Technique for Optimal Libration Point Orbit Transfers. In *Proceedings of the AIAA SCITECH 2024 Forum*, January 2024. doi:10.2514/6.2024-0208.

[28] G. V. Haman III and A. V. Rao. An Error Estimation and Mesh Refinement Method Applied to Optimal Libration Point Orbit Transfers. In *Proceedings of the 2024 American Controls Conference*, July 2024. doi:10.23919/ACC60939.2024.10644689.

[29] J. T. Betts and W. P. Huffman. Mesh Refinement in Direct Transcription Methods for Optimal Control. *Optimal Control Applications and Methods*, 19(1):1–21, January–February 1998. doi:10.1002/(SICI)1099-1514(199801/02)19:1%3C1::AID-OCA616%3E3.0.CO;2-Q.

[30] M. A. Patterson and A. V. Rao. Exploiting Sparsity in Direct Collocation Pseudospectral Methods for Solving Optimal Control Problems. *Journal of Spacecraft and Rockets*, 49(2):354–377, March–April 2012. doi:10.2514/1.A32071.

[31] G. T. Huntington. *Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control Problems*. Ph.D. dissertation, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, USA, 2007. doi:1721.1/42180.

[32] H. Seywald, E. M. Cliff, and K. H. Well. Range Optimal Trajectories for an Aircraft Flying in the Vertical Plane. *Journal of Guidance, Control, and Dynamics*, 17(2):389–398, March–April 1994. doi:10.2514/3.21210.

[33] E. D. Dolan, J. J. More, and T. S. Munson. Benchmarking Optimization Software with COPS 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory, Argonne, IL, USA, 2024. doi:10.2172/834714.

[34] A. V. Rao and K. D. Mease. Eigenvector Approximate Dichotomic Basis Method for Solving Hyper-Sensitive Optimal Control Problems. *Optimal Control Applications and Methods*, 21(1):1–19, January–February 2000. doi:10.1002/(SICI)1099-1514(200001/02)21:1%3C1::AID-OCA646%3E3.0.CO;2-V.

[35] M. A. Patterson and A. V. Rao. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using *hp*-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Transactions on Mathematical Software*, 41(1):1:1–1:37, October 2014. doi:10.1145/2558904.

[36] D. O. Anderson and P. V. Kokotovic. Optimal Control Problem Over Large Time Intervals. *Automatica*, 23(3):355–363, May 1987. doi:10.1016/0005-1098(87)90008-2.