Available online at www.sciencedirect.com

ScienceDirect

Journal of the Franklin Institute 352 (2015) 4081–4106

Journal of The Franklin Institute

www.elsevier.com/locate/jfranklin

Adaptive mesh refinement method for optimal control using nonsmoothness detection and mesh size reduction

Fengjin Liu[a], William W. Hager[b], Anil V. Rao[a,*]

[a]Department of Mechanical and Aerospace Engineering, University of Florida Gainesville, FL 32611, United States
[b]Department of Mathematics, University of Florida Gainesville, FL 32611, United States

Received 24 January 2015; received in revised form 22 April 2015; accepted 13 May 2015
Available online 23 May 2015

Abstract

An adaptive mesh refinement method for solving optimal control problems is developed. The method employs orthogonal collocation at Legendre–Gauss–Radau points, and adjusts both the mesh size and the degree of the approximating polynomials in the refinement process. A previously derived convergence rate is used to guide the refinement process. The method brackets discontinuities and improves solution accuracy by checking for large increases in higher-order derivatives of the state. In regions between discontinuities, where the solution is smooth, the error in the approximation is reduced by increasing the degree of the approximating polynomial. On mesh intervals where the error tolerance has been met, mesh density may be reduced either by merging adjacent mesh intervals or lowering the degree of the approximating polynomial. Finally, the method is demonstrated on two examples from the open literature and its performance is compared against a previously developed adaptive method.

1. Introduction

Over the past two decades, direct collocation methods have become popular in the numerical solution of nonlinear optimal control problems. In a direct collocation method, the state and the control are discretized at a set of appropriately chosen points in the time interval of interest. The continuous-time optimal control problem is then transcribed to a finite-dimensional nonlinear

*Corresponding author.
E-mail addresses: cliu121@ufl.edu (F. Liu), hager@ufl.edu (W.W. Hager), anilvrao@ufl.edu (A.V. Rao).

http://dx.doi.org/10.1016/j.jfranklin.2015.05.028

programming problem (NLP) and the NLP is solved using well known software [1,2]. Originally, direct collocation methods were developed as *h* methods (for example, Euler or Runge–Kutta methods) where the time interval is divided into a mesh and the state is approximated using the same fixed-degree polynomial in each mesh interval. Convergence in an *h* method is then achieved by increasing the number and placement of the mesh points [3–5]. More recently, a great deal of research has been done in the class of direct *Gaussian quadrature orthogonal collocation* methods [6–23]. In a Gaussian quadrature collocation method, the state is typically approximated using a Lagrange polynomial where the support points of the Lagrange polynomial are chosen to be points associated with a Gaussian quadrature. Originally, Gaussian quadrature collocation methods were implemented as *p* methods using a single interval. Convergence of the *p* method was then achieved by increasing the degree of the polynomial approximation. For problems whose solutions are smooth and well-behaved, a Gaussian quadrature collocation method has a simple structure and converges at an exponential rate [24–26]. The most well developed Gaussian quadrature methods are those that employ either Legendre–Gauss (LG) points [10,15], Legendre–Gauss–Radau (LGR) points [16,17,19], or Legendre–Gauss–Lobatto (LGL) points [6].

Many mesh refinement methods employing *h* or *p* direct collocation methods have been developed previously. Reference [27] describes what is essentially a *p* method where a differentiation matrix is used to identify switches, kinks, corners, and other discontinuities in the solution. References [28,29] locally refine the grids by splitting selected intervals according to some splitting criterion. Reference [5] develops a fixed-order method that uses a density function to generate a sequence of non-decreasing size meshes on which to solve the optimal control problem. References [30,31] (and the references therein) describe a dual weighted residual (DWR) method for mesh refinement and goal-oriented model reduction. The DWR method uses estimates of a dual multiplier together with local estimates of the residuals to adaptively refine a mesh and control the error in problems governed by partial differential equations. Finally, in Ref. [3] an error estimate is developed by integrating the difference between an interpolation of the time derivative of the state and the right-hand side of the dynamics. The error estimate developed in Ref. [3] is predicated on the use of a fixed-order method (for example, trapezoid, Hermite–Simpson, Runge–Kutta) and computes a low-order approximation of the integral of the aforementioned difference. Different from all of this previous research where the order of the method is fixed and the mesh can only increase in size, in the method of this paper varies the degree of the polynomial approximation and the mesh size can be reduced.

While *h* methods have a long history and *p* methods have shown promise in certain types of problems, both the *h* and *p* approaches have limitations. Specifically, achieving a desired accuracy tolerance may require an extremely fine mesh (in the case of an *h* method) or may require the use of an unreasonably large degree polynomial approximation (in the case of a *p* method). In order to reduce significantly the size of the finite-dimensional approximation, and thus improve computational efficiency of solving the NLP, *hp* collocation methods have been developed. In an *hp* method, both the number of mesh intervals and the degree of the approximating polynomial within each mesh interval are allowed to vary. Originally, *hp* methods were developed as finite-element methods for solving partial differential [32–36]. In the past few years the problem of developing *hp* methods for solving optimal control problems has been of interest [20,21,23]. References [20,21] describe *hp* adaptive methods where the error estimate is 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. It is noted that the approach of Refs. [20,21] creates a great deal of noise in the error estimate, thereby making these

approaches computationally intractable when a high-accuracy solution is desired. Furthermore, the error estimate of Refs. [20,21] does not take advantage of the exponential convergence rate of a Gaussian quadrature collocation method. On the other hand, Ref. [23] develops an error estimate based on the difference between the state interpolated on an increased number of Legendre–Gauss–Radau points in each mesh interval and the state obtained by integrating the dynamics on the solution using the interpolated state and control. Similar to the methods of Refs. [20,21], however,the method of Ref. [23] can only increase the size of the mesh.

As stated above, two key limitations of previous mesh refinement methods for optimal control are that the mesh can neither be decreased in size nor does the method attempt to detect discontinuities in the solution as the mesh refinement progresses. As a result, these previous methods may either create an unnecessarily large mesh. In addition, such methods place a larger than required number of mesh intervals near discontinuities or rapid changes in the solution. Both these limitations are addressed by the adaptive *hp* mesh refinement method described in this paper. The method of this paper is fundamentally different from any of these previously developed methods because it detects points where smoothness in the solution is lost and allows for reducing the size of the mesh. First, motivated by the approach similar to that of Ref. [37], nonsmoothness in the solution is determined by examining local maxima in the magnitude of the second derivative of the state within mesh intervals. Specifically, if a local maximum of the magnitude of the second derivative of the state is a user-specified factor greater than this second derivative at the same point on the previous mesh, then the mesh interval where this local maximum occurs is deemed to be a nonsmooth interval and the interval is divided. Mesh interval division in this manner then brackets the discontinuity within a narrow mesh interval. Outside of these intervals where the solution may be nonsmooth, the accuracy of the solution is improved by increasing the degree of the approximating polynomial. The method can reduce the size of the mesh either by combining mesh intervals or by reducing the degree of the approximating polynomial within a mesh interval. On mesh intervals where the error tolerance is satisfied, the degree of the approximating polynomial can be reduced when the high order terms in a power series expansion of the solution are sufficiently small. Similarly, adjacent mesh intervals can be combined into a single mesh interval when the degree of the polynomial approximation in these adjacent mesh intervals is essentially the same. The procedure for determining the intervals of nonsmoothness or for determining the mesh width is based on the solution of the collocated control problem on two meshes, one finer than the other, and an upper bound for the error in the collocation approximation given in Ref. [38]. Finally, it is noted that a preliminary version of the approach developed in this paper is given in Ref. [39].

This paper is organized as follows. In Section 2 the Bolza optimal control problem of interest in this research is described. In Section 3 the Legendre–Gauss–Radau collocation method used as the basis of the method of this paper is described. In Section 4 the mesh refinement method is described in detail. In Section 5 the method developed in Section 4 is demonstrated on two examples taken from the open literature. In Section 6 we provide a discussion of the method and the results obtained in the numerical examples. Finally, in Section 7 we provide conclusions of our research.

2. Bolza optimal control problem

Without loss of generality, consider the following general optimal control problem in the Bolza form. Determine the state $\mathbf{y}(\tau) \in \mathbb{R}^{n_y}$ and the control $\mathbf{u}(\tau) \in \mathbb{R}^{n_u}$ on the domain

$\tau \in [-1,+1]$, the initial time, t_0, and the terminal time t_f that minimize the cost functional

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

subject to the dynamic constraints

$$\frac{d\mathbf{y}}{d\tau} = \frac{t_f - t_0}{2} \mathbf{a}\big(\mathbf{y}(\tau), \mathbf{u}(\tau), t\big(\tau, t_0, t_f\big)\big), \tag{2}$$

the inequality path constraints

$$\mathbf{c}_{\min} \le \mathbf{c}(\mathbf{y}(\tau), \mathbf{u}(\tau), t(\tau, t_0, t_f)) \le \mathbf{c}_{\max}, \tag{3}$$

and the boundary conditions

$$\mathbf{b}_{\min} \le \mathbf{b}(\mathbf{y}(-1), t_0, \mathbf{y}(+1), t_f) \le \mathbf{b}_{\max}. \tag{4}$$

It is noted that the time interval $\tau \in [-1,+1]$ can be transformed to the time interval $t \in [t_0, t_f]$ via the affine transformation

$$t \equiv t\big(\tau, t_0, t_f\big) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}. \tag{5}$$

In the *hp* discretization, the domain $\tau \in [-1,+1]$ is partitioned into a *mesh* consisting of K *mesh intervals* $\mathcal{S}_k = [T_{k-1}, T_k]$, $k = 1, \ldots, K$, where $-1 = T_0 < T_1 < \cdots < T_K = +1$. The mesh intervals have the property that $\bigcup_{k=1}^{K} \mathcal{S}_k = [-1,+1]$. Let $\mathbf{y}^{(k)}(\tau)$ and $\mathbf{u}^{(k)}(\tau)$ be the state and the control in $\mathcal{S}_k$. The Bolza optimal control problem of Eqs. (1)–(4) can then be rewritten as follows. Minimize the cost functional

$$\begin{aligned}\mathcal{J} = {} & \mathcal{M}\big(\mathbf{y}^{(1)}(-1), t_0, \mathbf{y}^{(K)}(+1), t_f\big) \\ & + \frac{t_f - t_0}{2} \sum_{k=1}^{K} \int_{T_{k-1}}^{T_k} \mathcal{L}\big(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), t\big(\tau, t_0, t_f\big)\big)\, d\tau,\end{aligned} \tag{6}$$

subject to the dynamic constraints

$$\frac{d\mathbf{y}^{(k)}(\tau)}{d\tau} = \frac{t_f - t_0}{2} \mathbf{a}\big(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), t\big(\tau, t_0, t_f\big)\big) \quad (k = 1, \ldots, K), \tag{7}$$

the path constraints

$$\mathbf{c}_{\min} \le \mathbf{c}(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), t(\tau, t_0, t_f)) \le \mathbf{c}_{\max} \quad (k = 1, \ldots, K), \tag{8}$$

and the boundary conditions

$$\mathbf{b}_{\min} \le \mathbf{b}(\mathbf{y}^{(1)}(-1), t_0, \mathbf{y}^{(K)}(+1), t_f) \le \mathbf{b}_{\max}. \tag{9}$$

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

3. Legendre–Gauss–Radau collocation

The multiple-interval form of the continuous-time Bolza optimal control problem in Section 2 is discretized using collocation at Legendre–Gauss–Radau (LGR) points [16–19,23]. In the LGR collocation method, the state of the continuous-time Bolza optimal control problem is

approximated in $\mathcal{S}_k$, $k \in [1, \ldots, K]$, as

$$\mathbf{y}^{(k)}(\tau) \approx \mathbf{Y}^{(k)}(\tau) = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \ell_j^{(k)}(\tau), \quad \ell_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k+1} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}}, \tag{10}$$

where $\tau \in [-1, +1]$, $\ell_j^{(k)}(\tau)$, $j = 1, \ldots, N_k + 1$, is a basis of Lagrange polynomials, $\left(\tau_1^{(k)}, \ldots, \tau_{N_k}^{(k)}\right)$ are the Legendre–Gauss–Radau (LGR) [40] collocation points in $\mathcal{S}_k = [T_{k-1}, T_k)$, and $\tau_{N_k+1}^{(k)} = T_k$ is a noncollocated point. Differentiating $\mathbf{Y}^{(k)}(\tau)$ in Eq. (10) with respect to τ gives

$$\frac{d\mathbf{Y}^{(k)}(\tau)}{d\tau} = \sum_{j=1}^{N_k+1} \mathbf{Y}_j^{(k)} \frac{d\ell_j^{(k)}(\tau)}{d\tau}. \tag{11}$$

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

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} = \frac{t_f - t_0}{2} \mathbf{a}\left(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t\left(\tau_i^{(k)}, t_0, t_f\right)\right) \quad (i = 1, \ldots, N_k), \tag{12}$$

where

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

are the elements of the $N_k \times (N_k + 1)$ *Legendre–Gauss–Radau differentiation matrix* [16] in mesh interval $\mathcal{S}_k$, $k \in [1, \ldots, K]$. The LGR discretization then leads to the following nonlinear programming problem (NLP). Minimize the LGR quadrature approximation to the cost functional

$$\mathcal{J} \approx \mathcal{M}\left(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f\right) + \sum_{k=1}^{K} \sum_{j=1}^{N_k} \frac{t_f - t_0}{2} w_j^{(k)} \mathcal{L}\left(\mathbf{Y}_j^{(k)}, \mathbf{U}_j^{(k)}, t\left(\tau_j^{(k)}, t_0, t_f\right)\right) \tag{13}$$

subject to the collocation equations

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} - \frac{t_f - t_0}{2} \mathbf{a}\left(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t\left(\tau_i^{(k)}, t_0, t_f\right)\right) = \mathbf{0} \quad (i = 1, \ldots, N_k), \tag{14}$$

the discretized path constraints

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t(\tau_i^{(k)}, t_0, t_f)) \leq \mathbf{c}_{\max} \quad (i = 1, \ldots, N_k), \tag{15}$$

and the discretized boundary conditions

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f) \leq \mathbf{b}_{\max}. \tag{16}$$

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

$$\mathbf{Y}_{N_k+1}^{(k)} = \mathbf{Y}_1^{(k+1)} \quad (k = 1, \ldots, K-1). \tag{17}$$

Computationally, the constraint of Eq. (17) is eliminated from the problem by using the *same* variable for both $\mathbf{Y}_{N_k+1}^{(k)}$ and $\mathbf{Y}_1^{(k+1)}$.

4. Adaptive mesh refinement method

In this section the adaptive mesh refinement method of this paper is described. The description of the method is divided into five parts. First a review is provided for the approach of Ref. [23] for estimating the relative error in the solution on a given mesh. Next, the methods for both mesh interval division and polynomial degree increase are described. Finally, two approaches are described for reducing the size of the mesh.

4.1. Approximation of solution error

In this section the approach of Ref. [23] for estimating the relative error in the solution on a given mesh is reviewed. The relative error approximation derived in Ref. [23] is obtained by comparing two approximations to the state, one with higher accuracy. The key idea is that for a problem whose solution is smooth, an increase in the number of LGR points should yield a state that more accurately satisfies the dynamics. Hence, the difference between the solution associated with the original set of LGR points, and the approximation associated with the increased number of LGR points should yield an approximation of the error in the state.

Assume that the NLP of Eqs. (13)–(16) corresponding to the discretized Bolza optimal control problem has been solved on a mesh $\mathcal{S}_k=[T_{k-1},T_k]$, $k=1,\ldots,K$, with N_k LGR points in mesh interval $\mathcal{S}_k$. Suppose that the objective is to approximate the error in the state at a set of $M_k=N_k+1$ LGR points $\left(\hat{\tau}_1^{(k)},\ldots,\hat{\tau}_{M_k}^{(k)}\right)$, where $\hat{\tau}_1^{(k)}=\tau_1^{(k)}=T_{k-1}$, and that $\hat{\tau}_{M_k+1}^{(k)}=T_k$. Suppose further that the values of the state approximation at the points $\left(\hat{\tau}_1^{(k)},\ldots,\hat{\tau}_{M_k}^{(k)}\right)$ are denoted $\left(\mathbf{y}(\hat{\tau}_1^{(k)}),\ldots,\mathbf{y}(\hat{\tau}_{M_k}^{(k)})\right)$. Next, let the control be approximated in $\mathcal{S}_k$ using the Lagrange interpolating polynomial

$$\mathbf{U}^{(k)}(\tau)=\sum_{j=1}^{N_k}\mathbf{U}_j^{(k)}\hat{\ell}_j^{(k)}(\tau),\quad \hat{\ell}_j^{(k)}(\tau)=\prod_{\substack{l=1\\ l\neq j}}^{N_k}\frac{\tau-\tau_l^{(k)}}{\tau_j^{(k)}-\tau_l^{(k)}}, \tag{18}$$

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

$$\dot{\hat{\mathbf{Y}}}^{(k)}\left(\hat{\tau}_j^{(k)}\right)=\mathbf{Y}^{(k)}(\tau_{k-1})+\frac{t_f-t_0}{2}\sum_{l=1}^{M_k}\hat{I}_{jl}^{(k)}\mathbf{a}\left(\mathbf{Y}^{(k)}(\hat{\tau}_l^{(k)}),\mathbf{U}^{(k)}(\hat{\tau}_l^{(k)}),t(\hat{\tau}_l^{(k)},t_0,t_f)\right),$$
$$j=2,\ldots,M_k+1, \tag{19}$$

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

$$E_i^{(k)}\left(\hat{\tau}_l^{(k)}\right)=\left|\hat{Y}_i^{(k)}(\hat{\tau}_l^{(k)})-Y_i^{(k)}(\hat{\tau}_l^{(k)})\right|,$$
$$e_i^{(k)}\left(\hat{\tau}_l^{(k)}\right)=\frac{E_i^{(k)}(\hat{\tau}_l^{(k)})}{1+\max\limits_{\substack{j\in[1,\ldots,N_k+1]\\ k\in[1,\ldots,K]}}\left|Y_i^{(k)}(\tau_j^{(k)})\right|}\quad\left[\begin{array}{c}l=1,\ldots,M_k+1,\\ i=1,\ldots,n_y,\end{array}\right]. \tag{20}$$

The *maximum relative error* in $\mathcal{S}_k$ is then defined as

$$e_{\max}^{(k)} = \max_{\substack{i \in [1,\ldots,n_y] \\ l \in [1,\ldots,M_k+1]}} e_i^{(k)}\left(\hat{\tau}_l^{(k)}\right). \tag{21}$$

4.2. LGR collocation error bound for use in mesh refinement

It has been shown in Ref. [38] that, under suitable assumptions, the maximum difference between the LGR approximation $(\mathbf{y}, \mathbf{u})$ generated on a uniform mesh and the true solution $(\mathbf{y}^*, \mathbf{u}^*)$ evaluated on the same mesh satisfies an estimate of the form

$$\|\mathbf{y}-\mathbf{y}^*\|_\infty + \|\mathbf{u}-\mathbf{u}^*\|_\infty \leq \frac{ch^q}{N^{q-5/2}}, \tag{22}$$

where c is a constant, N is the number of LGR collocation points on each interval, h is the width of the mesh interval, q is the minimum of N and the number of continuous derivatives in the solution, and $\|\cdot\|_\infty$ denotes the sup-norm over the mesh points. Although Eq. (22) is an inequality that provides an upper bound for the error in the domain $[-1,+1]$, it is useful in developing a variable-order mesh refinement method that allows for changes in the width of each mesh interval, changes in the number of mesh intervals, and changes in the number of collocation points in each mesh interval.

4.3. Refining the mesh

After solving the NLP of Eqs. (13)–(16) on a given mesh, the maximum relative error estimate is computed in each mesh interval using Eq. (21). If in any mesh interval the estimated maximum relative error exceeds the mesh refinement accuracy tolerance ϵ, then the mesh interval is modified either by dividing it into smaller intervals or by increasing the degree of the approximating polynomial. As described earlier, a mesh interval is divided into subintervals when the solution is not sufficiently smooth in the mesh interval, while the polynomial degree in a mesh interval is increased when the solution is estimated to be sufficiently smooth in the mesh interval. The criterion for determining if a mesh interval is smooth or nonsmooth is based on whether the magnitude of the maximum second derivative of the state has increased by a specified factor from the previous mesh to the current mesh. In Section 4.3.1 an approach is developed for locating mesh intervals where the solution is nonsmooth, while Sections 4.3.2 and 4.3.3 provide methods for dividing a mesh interval (if the solution in the mesh interval is determined to be nonsmooth) or increasing the degree of the polynomial approximation (if the solution in the mesh interval is determined to be smooth).

4.3.1. Method for locating mesh intervals where solution is nonsmooth

Assume now that an optimal control problem has been solved on a mesh M using the previously described Radau collocation method and that $\mathbf{Y}^{(M)}(\tau) = [Y_1^{(M)}(\tau)\cdots Y_{n_y}^{(M)}(\tau)]$ is the state approximation that results from the solution on mesh M. Let τ_{ij}, $(i=1,\ldots,n_y,\ j=1,\ldots,L_i)$ be the values of $\tau \in [-1,+1]$ corresponding to the local maxima of $|\ddot{Y}_i^{(M)}(\tau)|$ that lie on the interior of mesh intervals [that is, τ_{ij}, $(i=1,\ldots,n_y,\ j=1,\ldots,L_i)$ are the interior local maxima of the absolute value of the i^{th} component of the state, $i \in [1,\ldots,n_y]$, on mesh M]. For compactness, let $P_{ij}^{(M)} = |\ddot{Y}_i^{(M)}(\tau_{ij})|$, $(i=1,\ldots,n_y, j=1,\ldots,L_i)$. Similarly, let $P_{ij}^{(M-1)}$ be the biggest interior local maximum of the function $|\ddot{Y}_i^{(M-1)}(\tau)|$ in the mesh interval on mesh $M-1$ that contains a

particular value τ_{ij} from mesh M. The solution in the mesh interval $\mathcal{S}_k$, $k \in [1, \ldots, K]$ on mesh M is considered to be nonsmooth if the condition

$$R_{ij} = \frac{P_{ij}^{(M)}}{P_{ij}^{(M-1)}} \geq \overline{R} \tag{23}$$

is satisfied for some $i \in [1, \ldots, n_y]$ and for some $j \in [1, \ldots, L_i]$ with $\tau_{ij} \in \mathcal{S}_k$, where $\overline{R}$ is a user-specified ratio.

The motivation for Eq. (23) is Ref. [37] where the function values on two uniform meshes, a fine mesh and a coarse mesh, are used to estimate low order derivatives of a function. If a maximum in the magnitude of the $(k+1)$th derivative on the fine mesh is much greater than corresponding $(k+1)$th derivative on the coarse mesh, then it is predicted that the kth derivative is discontinuous in an interval near the maximum. In the context of optimal control, the optimal control can be discontinuous at one or more switch points which often implies that the state has a discontinuous derivative. Thus, in a manner similar to that of Ref. [37], in the mesh refinement method of this paper the growth condition of Eq. (23) in the second derivative is used to determine if the solution in a mesh interval is nonsmooth.

4.3.2. Method for dividing a mesh interval

Assume now that the condition in Eq. (23) is satisfied, that is, $P_l^{(M)}/P_l^{(M-1)} > \overline{R}$ in a mesh interval $\mathcal{S}_k$ on the current mesh, and that the mesh interval needs to be divided. Treating Eq. (22) as an equality gives the relationship

$$e_k^{(M)} = \frac{c\left[h_k^{(M)}\right]^q}{\left[N_k^{(M)}\right]^{q-5/2}}. \tag{24}$$

Furthermore, assume for the ensuing mesh $M+1$ that it is desired to achieve a maximum relative error accuracy ϵ. Again, treating Eq. (22) as an equality gives

$$\epsilon = \frac{c\left[h_k^{(M+1)}\right]^q}{\left[N_k^{(M+1)}\right]^{q-5/2}}. \tag{25}$$

Since the mesh interval $\mathcal{S}_k$ is being divided, assume that $N_k^{(M+1)} = N_k^{(M)}$ (that is, the number of collocation points in each subinterval of $\mathcal{S}_k$ on mesh $M+1$ is the same as the number of collocation points in $\mathcal{S}_k$ on mesh M). Eqs. (24) and (25) can then be solved for the ratio $H = h_k^{(M+1)}/h_k^{(M)}$ as

$$H = \frac{h_k^{(M)}}{h_k^{(M+1)}} = \left(\frac{e_k^{(M)}}{\epsilon}\right)^{1/q}. \tag{26}$$

To obtain an estimate for q, consider the relative error on the mesh $M-1$, which is assumed to contain mesh M

$$e_k^{(M-1)} = \frac{c\left[h_k^{(M-1)}\right]^q}{\left[N_k^{(M-1)}\right]^{q-5/2}}. \tag{27}$$

The constant c is eliminated by forming the ratio of Eqs. (24) and (27), and then solve for q. The ratio H given in Eq. (26) is then used to determine the number of subintervals into which $\mathcal{S}_k$ should be divided. Specifically, the number of newly created subintervals must be at least $\lceil H \rceil$, the next largest integer greater than or equal to H. Now, because H can become large for certain problems, it is necessary to limit the growth in the number of subintervals. In this research the maximum number of subintervals into which $\mathcal{S}_k$ divided is based on the ratio of the relative error in the solution and the mesh refinement accuracy tolerance and is given as

$$H_{\max} = \lceil \log_N(e^{(k)}/\epsilon) \rceil, \tag{28}$$

where $\lceil \cdot \rceil$ is the next largest integer of the argument. The upper limit on the number of subintervals given in Eq. (28) is given as follows. First, when $e^{(k)} \gg \epsilon$ (say 10^6) the value of $H_{\max}$ will be typically in the range of 15–25. Next, it is seen that $H_{\max}$ will decrease to zero as $e^{(k)} \to \epsilon$. Thus, it will always be the case that $H_{\max}$ will provide a reasonable upper limit on allowable number of subintervals. Using Eq. (28) together with Eq. (27), the number of subintervals, denoted S, into which $\mathcal{S}_k$ is divided is given as

$$S = \min(\lceil H \rceil, H_{\max}). \tag{29}$$

4.3.3. Method for increasing the degree of the polynomial approximation

Suppose now that the error tolerance in a given mesh interval $\mathcal{S}_k$ has not been met and that the condition in Eq. (23) is not satisfied. In this case the solution in the mesh interval is regarded as smooth in $\mathcal{S}_k$ and, if possible, the degree of the polynomial approximation used on mesh $M+1$ is increased in order to reduce the solution error. Let $e_k^{(M)}$ denote the error on the interval $\mathcal{S}_k$ of mesh M. If the width of the mesh interval is the same on meshes M and $M+1$, then $h_k^{(M)} = h_k^{(M+1)}$ and Eqs. (24) and (25) together with the value of q computed in Section 4.3.2, can be used to solve for $N_k^{(M+1)}$ as

$$N_k^{(M+1)} = N_k^{(M)} \left(\frac{e_k^{(M)}}{\epsilon} \right)^{1/(q-5/2)}. \tag{30}$$

Now, in order to obtain a strict increase in the number of collocation points in $\mathcal{S}_k$ on mesh $M+1$, the result of Eq. (30) is replaced with

$$N_k^{(M+1)} = \left\lceil N_k^{(M)} \left(\frac{e_k^{(M)}}{\epsilon} \right)^{1/(q-5/2)} \right\rceil. \tag{31}$$

Finally, to ensure that the polynomial degree does not grow to an unreasonably large value, an upper limit $N_{\max}$ is set for the maximum allowable polynomial degree. If $N_k^{(M+1)} > N_{\max}$, then the mesh interval is divided into equally spaced subintervals with $N_k^{(M)}$ collocation points in each subinterval using the procedure of Section 4.3.2.

4.3.4. Reducing the number of collocation points in a mesh interval

In addition to the two approaches for increasing the size of the mesh as described in Sections 4.3.2 and 4.3.3, the mesh size can be decreased either by reducing the number of collocation points or by reducing the number of mesh intervals. Both these methods for mesh size reduction are now described.

Consider any mesh interval $\mathcal{S}_k = [T_{k-1}, T_k]$ where the accuracy tolerance ϵ has been satisfied. Suppose further that it is desired to determine if it is possible to reduce the degree of the

polynomial approximation of the state in $\mathcal{S}_k$ while retaining the same accuracy as has been attained using the current polynomial degree. The determination as to whether or not the polynomial degree can be reduced is based on the following power series representation of the polynomial approximation of the state in $\mathcal{S}_k$.[1] Let $\mu_k = (T_{k-1} + T_k)/2$, $h_k = (T_k - T_{k-1})/2$, and $\mathbf{Y}^{(k)}(\tau)$ be the midpoint of the mesh interval, the mesh interval half-width, and the state approximation, respectively, in mesh interval $\mathcal{S}_k$. Then the Lagrange polynomial representation of the ith component of the state approximation, $Y_i^{(k)}(\tau)$, is given as

$$Y_i^{(k)}(\tau) = \sum_{j=1}^{N_k+1} Y_{ij}\ell_j\left(\frac{\tau-\mu_k}{h_k}\right), \quad \ell_j(s) = \prod_{\substack{i=1\\ i\neq j}}^{N_k+1}\left(\frac{s-s_i}{s_j-s_i}\right), \tag{32}$$

where $-1 = s_1 < s_2 < \cdots < s_{N_k} < 1$ are the LGR points on the interval $[-1, +1]$ and $s_{N_k+1} = +1$. The polynomial $\ell_j(s)$ can then be written in the following form:

$$\ell_j(s) = \sum_{l=0}^{N_k} a_{lj}s^l, \tag{33}$$

where a_{lj} are coefficients that depend only on the LGR points and are computed as follows. First, suppose that $Q_j(s)$ is a power series whose roots are the same as those of $\ell_j(s)$, that is, $Q_j(s)$ has roots $\{s_k\}_{\substack{k=1\\k\neq j}}^{N_k+1}$ and has the following form:

$$Q_j(s) = \sum_{l=0}^{N_k} Q_{lj}s^l. \tag{34}$$

Then, because $\ell_j(s_j) = 1$,

$$\ell_j(s) = \frac{1}{Q_j(s_j)}Q_j(s) = \sum_{l=0}^{N_k}\frac{Q_{lj}}{Q_j(s_j)}s^l, \tag{35}$$

which implies that

$$a_{lj} = \frac{Q_{lj}}{Q_j(s_j)}. \tag{36}$$

It is noted that the coefficients a_{lj} depend only on N_k and, thus, only need to be computed once for each value of N_k after which they can be stored for future use on other intervals and meshes. Next, combining (32) and (33) gives

$$Y_i^{(k)}(\tau) = \sum_{l=0}^{N_k} b_{il}\left(\frac{\tau-\mu_k}{h_k}\right)^l, \quad b_{il} = \sum_{j=1}^{N_k+1} Y_{ij}a_{lj}. \tag{37}$$

Now, it follows from the definitions of mesh interval midpoint and the mesh interval half-width that $|\tau-\mu_k|/h_k \leq 1$ for $\tau \in \mathcal{S}_k$. Therefore, if the N_k^{th} degree term in Eq. (37) is dropped, the pointwise absolute error in mesh interval $\mathcal{S}_k$ is at most $|b_{iN_k}|$. In order to obtain an error estimate in mesh interval $\mathcal{S}_k$ that can be compared with the mesh refinement accuracy tolerance, however, it is necessary to normalize the coefficients b_{il}. The quantities used to normalize the coefficients b_{il} for each component of the state $i \in [1, \ldots, n_y]$ are similar to the manner in which the relative

[1] It is noted that Ref. [41] also employs a polynomial reduction procedure when approximating a function using Chebyshev polynomials. Specifically, in Ref. [41] if the coefficients of the highest terms in a Chebyshev polynomial expansion are negligible, then the grid is reduced by the number of negligible terms in the expansion.

error estimate in the solution from Eq. (21) is obtained and is given as

$$\beta_i = 1 + \max_{k \in [1,\ldots,K]} \max_{\tau \in \mathcal{S}_k} |Y_i^{(k)}(\tau)| \quad (i = 1, \ldots, n_y). \tag{38}$$

Then, starting with the highest power, all terms in Eq. (37) can continue to be removed (thus lowering the degree of the polynomial) until a coefficient b_{il}/β_i is found such that $|b_{il}|/\beta_i > \epsilon$. The aforementioned process of polynomial degree reduction is repeated for all components of the state $i \in [1, \ldots, n_y]$, resulting in reduced polynomial degrees $N_1^{(k)}, \ldots, N_{n_y}^{(k)}$. Then, the degree of the polynomial used for $\mathbf{Y}^{(k)}(\tau)$ in mesh interval $\mathcal{S}_k$ on the ensuing mesh is the one that corresponds to maximum of $(N_1^{(k)}, \ldots, N_{n_y}^{(k)})$. In other words, the reduced polynomial degree in mesh interval $\mathcal{S}_k$ is the one that corresponds to the largest of the reduced polynomial degrees over all components of the state. Finally, because the Radau collocation method requires at least one collocation point in each mesh interval, the polynomial approximation cannot be reduced to a constant but can only be reduced to a linear function in each mesh interval.

4.3.5. Merging mesh intervals

The second manner in which the mesh size can be reduced is by merging two adjacent mesh intervals into a single mesh interval. Before testing whether two subintervals can be merged, the highest powers in the polynomial approximation are eliminated when possible using the process described in Section 4.3.4. Next, two mesh intervals are joined into a single interval when the polynomials on the adjacent subintervals are roughly the same. First note that if $N_{k+1} \neq N_k$, then mesh intervals $\mathcal{S}_{k+1} = [T_k, T_{k+1}]$ and $\mathcal{S}_k = [T_{k-1}, T_k]$ cannot be merged because the degree of the polynomials in each mesh interval are different. The test for deciding when two intervals can be merged is the following: if the polynomial on the larger interval is extended into the smaller interval and if the pointwise difference between the original polynomial on the small interval and the extension from the large interval is at most ϵ, the state accuracy tolerance, then the mesh intervals $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ are merged to form a single interval.

By continuity, the polynomials on $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ are equal at T_k, the point where the intervals join. Typically, the difference between the original polynomial on the small interval and the extension from the large interval is largest at the end point, either T_{k-1} or T_{k+1}. If this difference is greater than ϵ, then the intervals should not be joined. On the other hand, if the difference is less than ϵ, then it is necessary to examine the polynomial difference over the entire smaller interval. It is possible to approximate the pointwise difference between the polynomials by evaluating the polynomials at more points on the smaller interval. An alternative to this approach, which is now described, is to derive an upper bound for this difference that is valid over the entire interval.

From Eq. (37) it is seen that the midpoint and mesh half-width of mesh interval $\mathcal{S}_k$ are different from the midpoint and mesh half-width of mesh interval $\mathcal{S}_{k+1}$. Consequently, using the representation in Eq. (37), it is difficult to bound the difference between the polynomial approximations in $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$. In order to bound the difference between these two polynomials, it is convenient to expand these two polynomials about the junction, T_k, between the mesh intervals $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$. These expansions are obtained by expressing the Lagrange basis in Eq. (32) in terms of the following two power series using the points $+1$ and -1:

$$\ell_j(s) = \sum_{l=0}^{N_k} (s-1)^l a_{lj}^{(k)}, \tag{39}$$

$$\ell_j(s) = \sum_{l=0}^{N_k} (s+1)^l a_{lj}^{(k+1)}. \tag{40}$$

The representations of the Lagrange polynomials given in Eqs. (39) and (40) can then be evaluated in an analogous manner to the approach used to compute the coefficients in Eq. (33). Furthermore, combining Eq. (32) with Eq. (39) in mesh interval $\mathcal{S}_k$ and with Eq. (40) in mesh interval $\mathcal{S}_{k+1}$, the ith component of the state approximations $\mathbf{Y}^{(k)}(\tau)$ and $\mathbf{Y}^{(k+1)}(\tau)$ in mesh intervals $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ is given, respectively, as

$$Y_i^{(k)}(\tau) = \sum_{l=0}^{N_k} c_{il}^{(k)} \left(\frac{\tau - T_k}{h_k}\right)^l, \quad c_{il}^{(k)} = \sum_{j=1}^{N_k+1} Y_{ij} a_{lj}^{(k)}, \tag{41}$$

$$Y_i^{(k+1)}(\tau) = \sum_{l=0}^{N_k} c_{il}^{(k+1)} \left(\frac{\tau - T_k}{h_{k+1}}\right)^l, \quad c_{il}^{(k+1)} = \sum_{j=1}^{N_k+1} Y_{ij} a_{lj}^{(k+1)}. \tag{42}$$

The representations of $Y_i^{(k)}(\tau)$ and $Y_i^{(k+1)}(\tau)$ in Eqs. (41) and (42) now differ only in that h_k appears in the denominator of Eq. (41) while h_{k+1} appears in the denominator of Eq. (42). In order to unify Eqs. (41) and (42) into a form such that the denominators are the same, let $\overline{h}_k = \min\{h_k, h_{k+1}\}$. The representations of the state approximation in mesh intervals $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ can then be written, respectively, as

$$Y_i^{(k)}(\tau) = \sum_{l=0}^{N_k} b_{il}^{(k)} \left(\frac{\tau - T_k}{\overline{h}_k}\right)^l, \quad b_{il}^{(k)} = c_{il}^{(k)} \left[\frac{\overline{h}_k}{h_k}\right]^l, \tag{43}$$

$$Y_i^{(k+1)}(\tau) = \sum_{l=0}^{N_k} b_{il}^{(k+1)} \left(\frac{\tau - T_k}{\overline{h}_k}\right)^l, \quad b_{il}^{(k+1)} = c_{il}^{(k+1)} \left[\frac{\overline{h}_k}{h_{k+1}}\right]^l. \tag{44}$$

Because the powers in both expansions are the same, the difference between the polynomials $Y_i^{(k)}(\tau)$ and $Y_i^{(k+1)}(\tau)$ is

$$Y_i^{(k)}(\tau) - Y_i^{(k+1)}(\tau) = \sum_{l=0}^{N_k} \left(b_{il}^{(k)} - b_{il}^{(k+1)}\right) \left(\frac{\tau - T_k}{\overline{h}_k}\right)^l.$$

Then, because $|\tau - T_k|/\overline{h}_k \le 2$ for τ in the smaller interval, it follows from the triangle inequality that the polynomial difference has the pointwise bound

$$\max\{|Y_i^{(k)}(\tau) - Y_i^{(k+1)}(\tau)|\} \le \sum_{l=0}^{N_k} 2^l |b_{il}^{(k)} - b_{il}^{(k+1)}|, \quad \tau \in \overline{\mathcal{S}}_k \tag{45}$$

where $\overline{\mathcal{S}}_k$ is the smaller of $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$. Now the result of Eq. (45) provides an upper bound on the maximum absolute difference between $Y_i^{(k)}(\tau)$ and $Y_i^{(k+1)}(\tau)$. In order to obtain a difference that can be compared with the mesh refinement accuracy tolerance, however, it is necessary to scale Eq. (45) appropriately to obtain an upper bound on the maximum *relative* difference between the polynomials $Y_i^{(k)}(\tau)$ and $Y_i^{(k+1)}(\tau)$. The quantities used to normalize each of the absolute differences $|Y_i^{(k)}(\tau) - Y_i^{(k+1)}(\tau)|$ are the values β_i $(i = 1, \ldots, n_y)$ given in Eq. (38) and which were used in the approach for polynomial reduction as described in Section 4.3.4. Scaling Eq. (45) by β_i gives the following relative difference between $Y_i^{(k)}(\tau)$ and $Y_i^{(k+1)}(\tau)$:

$$\frac{1}{\beta_i}\max\left\{|Y_i^{(k)}(\tau)-Y_i^{(k+1)}(\tau)|\right\}\le\frac{1}{\beta_i}\sum_{l=0}^{N_k}2^l\left|b_{il}^{(k)}-b_{il}^{(k+1)}\right|,\quad \tau\in\overline{\mathcal{S}}_k,\quad \left(i=1,\ldots,n_y\right). \tag{46}$$

Then, if

$$\frac{1}{\beta_i}\sum_{l=0}^{N_k}2^l\left|b_{il}^{(k)}-b_{il}^{(k+1)}\right|<\epsilon,\quad \tau\in\overline{\mathcal{S}}_k\quad \left(i=1,\ldots,n_y\right), \tag{47}$$

for *all* components of the state $i\in[1,\ldots,n_y]$ (where we recall that ϵ is the mesh refinement accuracy tolerance), the mesh intervals $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ are merged into a single mesh interval. Finally, it is noted that $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ cannot be merged if Eq. (47) is violated for any component of the state $i\in[1,\ldots,n_y]$.

4.3.6. Mesh refinement method

The mesh refinement method described in this paper is summarized in the following steps shown below:

I. Supply an initial mesh that consists of K mesh intervals $\mathcal{S}_k=[T_{k-1},T_k]$, $k=1,\ldots,K$, with N_k collocation points on each interval.
II. Solve the NLP of Eqs. (13)–(16) on the initial mesh.
III. If the maximum relative error in all mesh intervals is less than the mesh refinement accuracy tolerance, ϵ, then terminate.
IV. Generate a second mesh as follows:
   (i) Compute the maximum relative error given by Eq. (21).
   (ii) In every mesh interval where the maximum relative error of Eq. (21) is greater than the mesh refinement accuracy tolerance, ϵ, add three collocation points in the mesh interval.
   (iii) In every mesh interval where the maximum relative error of Eq. (21) is less than the mesh refinement accuracy tolerance, ϵ, use the mesh size reduction approaches given in Sections 4.3.4 and 4.3.5 to decrease the degree of the approximating polynomial or merge mesh intervals where possible.

For every mesh after the second mesh, employ the following steps.

(iv) Repeat the following steps until the mesh refinement accuracy tolerance ϵ is satisfied in every mesh interval $\mathcal{S}_k$ $(k=1,\ldots,K)$ or a specified maximum number of mesh refinement iterations is reached:
   (i) Solve the problem on the current mesh and estimate the maximum relative error $e_{\max}^{(k)}$ in each mesh interval.
   (ii) Increase or decrease the size of the mesh using steps (a)–(c) below:
      (a) For every mesh interval $k\in[1,\ldots,K]$ where $e_{\max}^{(k)}\ge\epsilon$, estimate the ratio R_{ij} for every component of the state using the approach in Section 4.3.1. If $R_{ij}>\overline{R}$ (where $\overline{R}$ is a threshold of significance of the ratio of second derivatives) for any component of the state, then divide the mesh interval into subintervals using the approach in Section 4.3.2. Otherwise, increase the degree of the polynomial approximation in mesh interval $k\in[1,\ldots,K]$ using the approach in Section 4.3.3.

(b) For every mesh interval $k \in [1, \ldots, K]$ where $e_{\max}^{(k)} < \epsilon$, determine if the number of collocation points in the mesh interval can be reduced using the approach of Section 4.3.4.

(c) For every pair of adjacent mesh intervals $k \in [1, \ldots, K]$ and $k+1 \in [1, \ldots, K]$ where $e_{\max}^{(k)} < \epsilon$ and $e_{\max}^{(k+1)} < \epsilon$, determine if these mesh intervals can be combined using the approach of Section 4.3.5.

(iii) Construct the new mesh, interpolate the solution from the previous mesh to this new mesh, and go to Step IV(i).

A schematic of the method is shown in Fig. 1.

5. Examples

In this section the mesh refinement method described in Section 4 is applied to two examples taken from the open literature. The first example is the robot arm optimal control problem taken from Ref. [42]. This example demonstrates the ability of the mesh refinement method to accurately determine regions of nonsmoothness in the problem on a problem whose solution contains multiple control discontinuities. The second example is the hyper-sensitive optimal control problem taken from Ref. [43]. This second example demonstrates the ability of the mesh refinement method to reduce the size of the mesh by eliminating unneeded mesh points and collocation points.

The following terminology is used in the examples. First, the method developed in this paper is called an *hp* method while the method of Ref. [23] (used for comparison) is called a *ph* method. Next, the notation *ph*-$(N_{\min}, N_{\max})$ refers to a variant of the aforementioned *ph* method where the number of collocation points in a mesh interval is allowed to vary between $N_{\min}$ and $N_{\max}$. Furthermore, the quantity M denotes the number of the *mesh refinement*, where $M=0$ corresponds to the initial mesh, and the quantities N and K denote the total number of LGR collocation points and the number of mesh intervals, respectively. Finally, it is noted that the *hp* method will maintain at least two collocation points in a mesh interval and does not require that an upper limit on the number of collocation points be specified.

All results shown in this paper were obtained using the MATLAB optimal control software $\mathbb{GPOPS}-\mathbb{II}$ [44] using the NLP solver *SNOPT* [1] using an optimality tolerance of 10^{-10} and a feasibility tolerance of 2×10^{-10}. All first derivatives for the NLP solver were obtained using the MATLAB automatic differentiation tool *AdiGator* [45]. In all results a mesh refinement accuracy tolerance $\epsilon = 10^{-6}$ was used with an initial mesh consisting of 10 uniformly spaced mesh intervals and the four collocation points per mesh interval, a second derivative ratio threshold $\overline{R} = 1.2$ (which is the same value used in Ref. [46]), and a maximum number of collocation points $N_{\max} = 14$ for the *hp* method. Furthermore, the initial guess for all examples is a straight line for variables with boundary conditions at both endpoints and is a constant for variables with boundary conditions at only one endpoint. All computations were performed on a 2.5 GHz Intel Core i7 MacBook Pro running MAC OS-X version 10.8.5 (Mountain Lion) with 16 GB 1333 MHz DDR3 of RAM and MATLAB Version R2012b (build 8.0.0.783). The central processing unit (CPU) times reported in this paper are 10-run averages of the execution time and exclude the time required to solve the NLP on the first mesh (because for any example the CPU time required to solve the problem on the initial guess is the same for any method and, thus, adds a constant to the total CPU time).

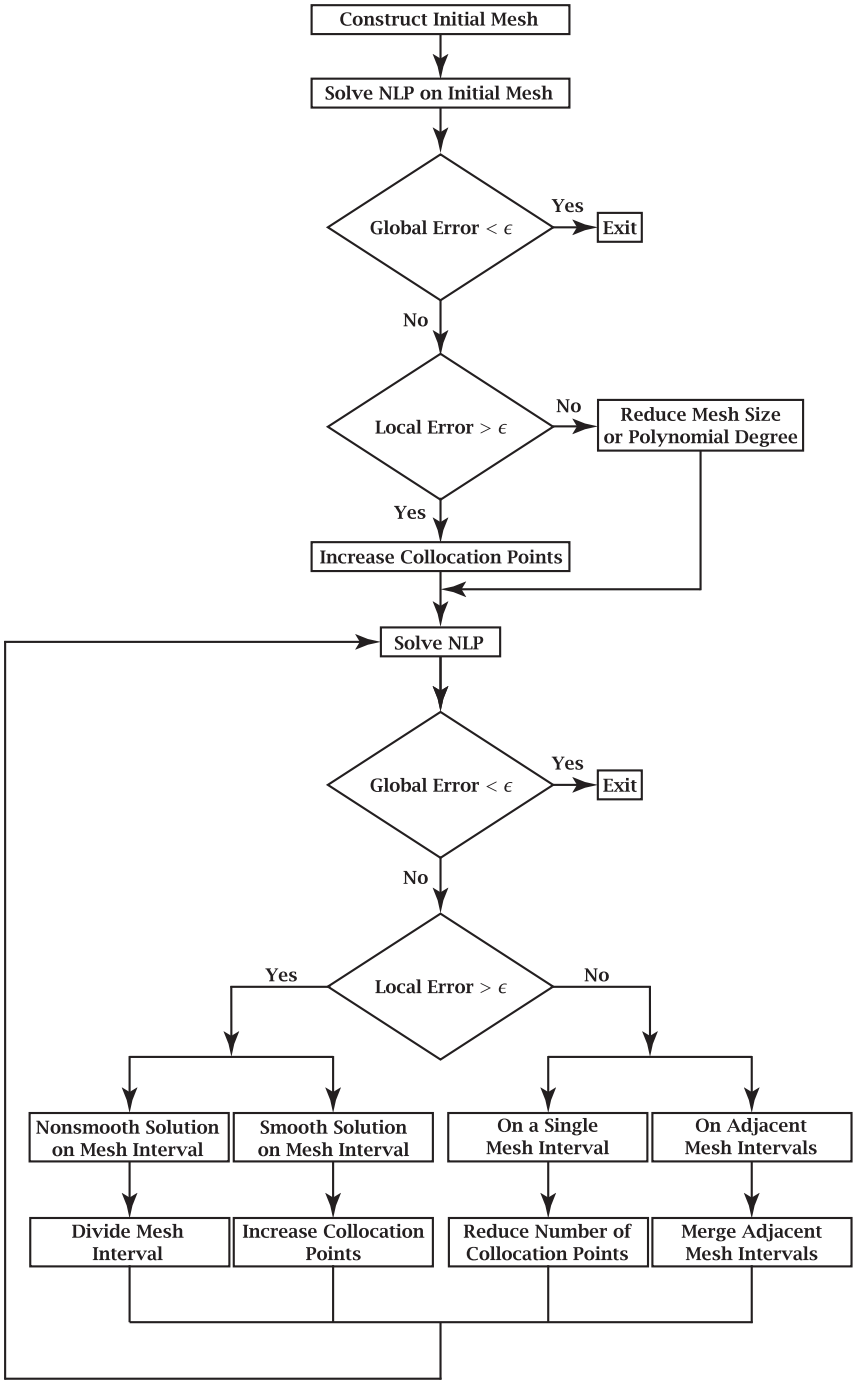

Fig. 1. Schematic of *hp* mesh refinement method.

5.1. Example 1

Consider the following minimum time reorientation of a robot arm taken from Ref. [42]. The objective is to minimize the cost functional

$$J = t_f \tag{48}$$

subject to the dynamic constraints

$$\begin{aligned} &\dot{y}_1 = y_2, \quad \dot{y}_2 = u_1/L, \\ &\dot{y}_3 = y_4, \quad \dot{y}_4 = u_2/I_\theta, \\ &\dot{y}_5 = y_6, \quad \dot{y}_6 = u_3/I_\phi, \end{aligned} \tag{49}$$

the control inequality constraints

$$-1 \le u_i(t) \le 1 \quad (i = 1, 2, 3), \tag{50}$$

and the boundary conditions

$$\begin{aligned} &y_1(0) = y_{10}, \quad y_1(t_f) = y_{1f}, \\ &y_2(0) = y_{20}, \quad y_2(t_f) = y_{2f}, \\ &y_3(0) = y_{30}, \quad y_3(t_f) = y_{3f}, \\ &y_4(0) = y_{40}, \quad y_4(t_f) = y_{4f}, \\ &y_5(0) = y_{50}, \quad y_5(t_f) = y_{5f}, \\ &y_6(0) = y_{60}, \quad y_6(t_f) = y_{6f}, \end{aligned} \tag{51}$$

where

$$I_\theta = \frac{((L-y_1)^3 + y_1^3)}{3} \sin^2(y_5), \quad I_\phi = \frac{((L-y_1)^3 + y_1^3)}{3}, \quad L = 5, \tag{52}$$

and

$$\begin{aligned} &y_{10} = 9/2, \quad y_{1f} = 9/2, \\ &y_{20} = 0, \quad y_{2f} = 0, \\ &y_{30} = 0, \quad y_{3f} = 2\pi/3, \\ &y_{40} = 0, \quad y_{4f} = 0, \\ &y_{50} = \pi/4, \quad y_{5f} = \pi/4, \\ &y_{60} = 0, \quad y_{6f} = 0. \end{aligned} \tag{53}$$

It is known for this problem that the optimal control has five discontinuities at $t \approx (2.286, 2.827, 4.570, 6.385, 6.855)$. Fig. 2a–c shows the three components of the optimal control obtained by solving the optimal control problem of Eqs. (48)–(51) where these control discontinuities are clearly seen. Next, Fig. 3a shows the evolution of the mesh points for this example where it is seen that the second derivative ratio, R, is greater than the threshold $\overline{R} = 1.2$ in segments that contain four of the five control discontinuities, while the fifth (middle) discontinuity is already accurately located due to the fact that a mesh point is located at $t \approx 4.570$. Furthermore, it is seen that the mesh continues to be refined only in neighborhoods of the discontinuities because the solution is smooth outside of these small segments. Thus, the error outside of the neighborhoods of the discontinuities is reduced by increasing the degree of the polynomial approximation and not by dividing mesh intervals. Moreover, it is seen that the mesh quickly progresses to a point where the accuracy tolerance is satisfied and stops after three ($M = 3$) mesh refinements.

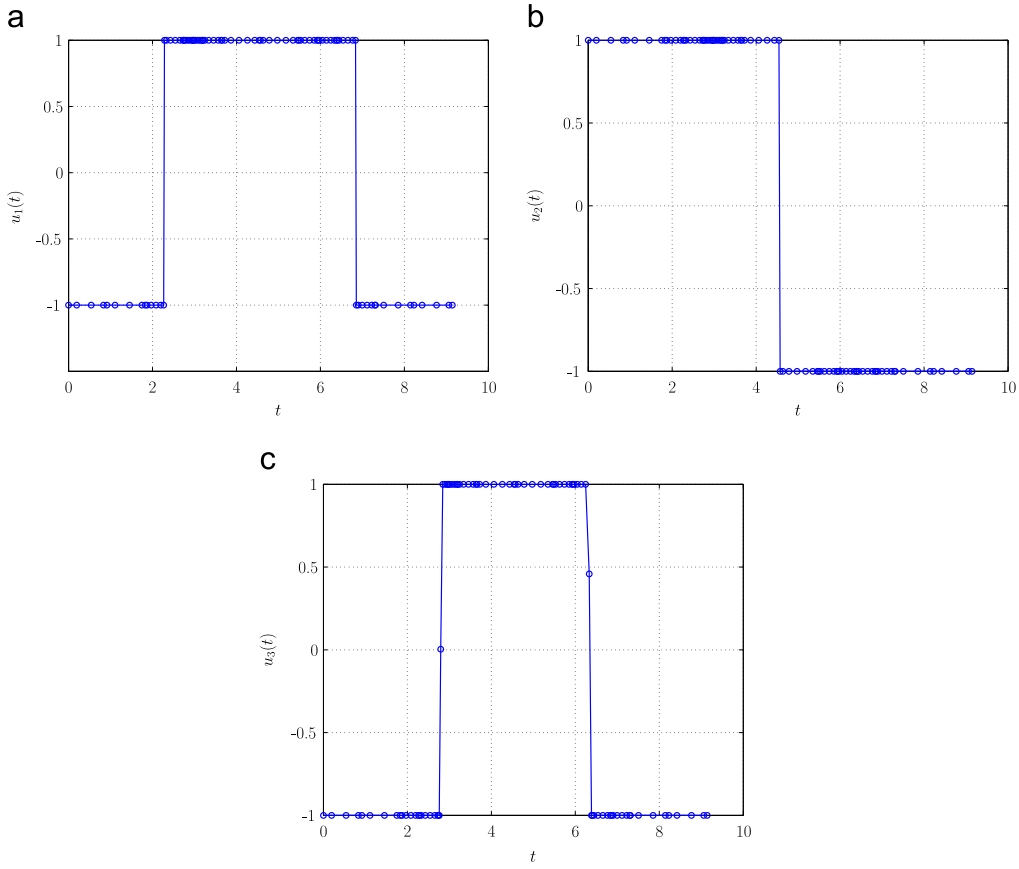

Fig. 2. Optimal control for example 1 using the *hp* method. (a) $u_1(t)$ vs. t. (b) $u_2(t)$ vs. t. (c) $u_3(t)$ vs. t.

Now, in order to demonstrate that the *hp* method correctly identifies all of the discontinuities even when none of the initial mesh points lie close to a discontinuity, consider now the *hp* mesh refinement using an initial mesh that consists of *nine* equally spaced mesh intervals. The mesh history for this initial mesh is shown in Fig. 3e. As alluded to earlier, this alternate initial mesh does contain a mesh point that lies close to a discontinuity. As the mesh refinement progresses from this initial mesh it is seen in Fig. 3e that the mesh interval $\mathcal{S}=[4,5]$ is found to be nonsmooth ($R>\overline{R}$) on the second mesh ($M=1$) and the *hp* method divides this mesh interval into two equally spaced mesh intervals in order to meet the accuracy tolerance. Furthermore, as with the initial 10-interval mesh, it is seen for the initial nine-interval mesh that the *hp* method adds mesh intervals that either closely surround or lie at the location of the five discontinuities and adds very few mesh points anywhere else on the interval $[0,t_f]$.

The *hp* mesh refinement method is now compared against various *h* methods and the previously developed *ph* mesh refinement method of Ref. [23]. First, it is seen from Fig. 3f that the CPU time required to solve the problem grows as the product of the number of collocation points and the number of mesh refinements, *NM*. In other words, as one might expect, the CPU time increases due to an increase in either the size of the mesh or an increase in the number of mesh refinement iterations required to meet the mesh refinement accuracy tolerance. An alternate

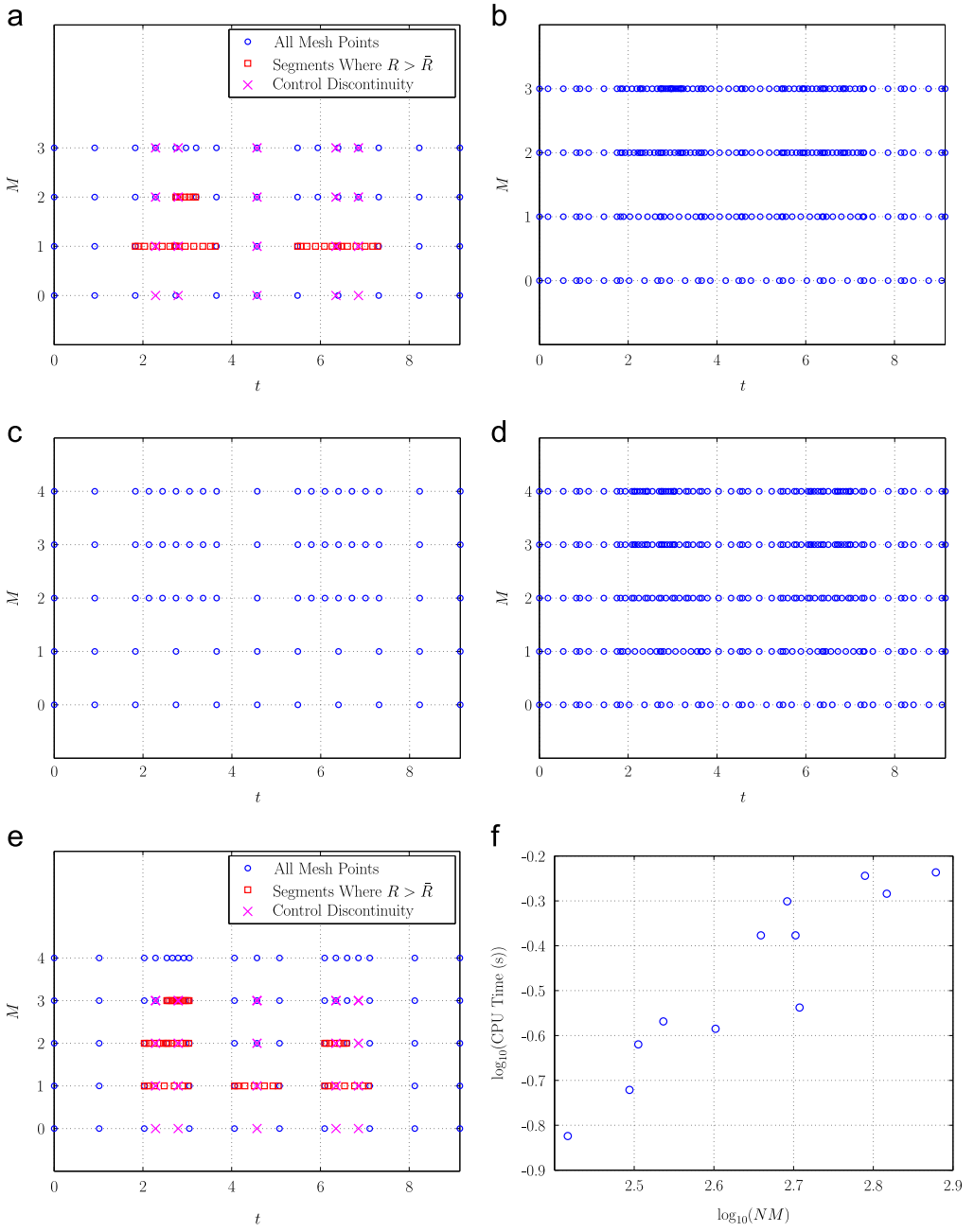

Fig. 3. Mesh refinement history of Example 1 when using the *hp* and $ph-(3,8)$ methods. (a) *hp* mesh point history. (b) *hp* collocation point history. (c) *ph* mesh point history. (d) *ph* collocation point history. (e) *hp* mesh point history starting from nine equally mesh intervals. (f) $\log_{10}$(CPU time) vs. $\log_{10}(NM)$.

Table 1
Mesh refinement results for Example 1 using *hp* and various $ph-(N_{\min}, N_{\max})$ methods.

Method	$N_{\min}$	$N_{\max}$	CPU time (s)	N	K	M
hp	–	14	0.15	87	15	3
h	2	2	0.29	170	81	3
h	3	3	0.26	100	32	4
h	4	4	0.24	80	20	4
ph	3	8	0.19	78	18	4
ph	3	10	0.42	76	19	6
ph	3	12	0.57	88	22	7
ph	3	14	0.58	84	18	9
ph	4	8	0.27	86	18	4
ph	4	10	0.50	82	16	6
ph	4	12	0.42	84	19	6
ph	4	14	0.52	82	16	8

view of this same trend in the CPU time is reflected in Table 1. Comparing Fig. 3f and Table 1, it is seen that the CPU time required by the *hp* method lies in the lower left-hand corner of the data shown in Fig. 3f. Next, Fig. 3c–d shows the mesh refinement history for both the *hp* and the $ph-(3,8)$ method (where the $ph-(3,8)$ method is the best performing of the *ph* methods on this example). First, it is seen that the $ph-(3,8)$ method introduces more mesh intervals during the first few mesh refinements when compared with the *hp* method. Furthermore, the $ph-(3,8)$ method takes one more mesh refinement to meet the accuracy tolerance and the mesh points are less concentrated near the discontinuities when compared with the *hp* method. Finally, Table 1 that while the total number of collocation points using the *hp* method is slightly larger using the *hp* method than it is for most of the *ph* methods, the *hp* method meets the mesh refinement accuracy tolerance in many fewer mesh refinements when compared with most of the *ph* methods. Finally, it is seen that the number of collocation points required by the *hp* method is significantly less than the number of collocation points required by either the $h-2$ or $h-3$ methods and is comparable in size to the mesh produced by the $h-4$ method. It is noted, however, that the $h-4$ method is still less computationally efficient than the *hp* method. In addition, the *hp* mesh refinement method of this paper is compared with the mesh refinement method used in the *Sparse Optimization Suite* (SOS) [47]. It is noted that 128 grid points and six mesh refinement iterations were required to solve the problem using (SOS) to meet a relative error accuracy tolerance $\epsilon = 10^{-6}$. Thus, the final SOS grid is approximately 1.5 times larger than the final mesh obtained using the method of this paper while SOS required approximately twice the number mesh iterations when compared with the method of this paper. Finally, it is noted that the SOS computation times are not compared with those obtained in this research because the SOS NLP solver is different from SNOPT (used in this research), SOS is written in FORTRAN 95 (whereas the work of this research was performed using MATLAB), and the machine on which SOS was used to perform the computations is completely different from the machine on which the computations in this research were performed.

5.2. Example 2

Consider the following hyper-sensitive optimal control problem taken from Refs. [23,43]. Minimize the cost functional

$$J = \frac{1}{2}\int_0^{t_f} \left(y^2 + u^2\right)\, dt \tag{54}$$

subject to the dynamic constraint

$$\dot{y} = -y + u \tag{55}$$

and the boundary conditions

$$\begin{aligned} y(0) &= 1.5, \\ y(t_f) &= 1, \end{aligned} \tag{56}$$

where t_f is fixed. The exact solution to the optimal control problem of Eqs. (54)–(56) is

$$\begin{bmatrix} y^*(t) \\ u^*(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1+\sqrt{2} & 1-\sqrt{2} \end{bmatrix} \begin{bmatrix} e^{t\sqrt{2}}c_1 \\ e^{-t\sqrt{2}}c_2 \end{bmatrix}, \quad \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \frac{1}{e^{-t_f\sqrt{2}} - e^{t_f\sqrt{2}}} \begin{bmatrix} 1.5e^{-t_f\sqrt{2}} - 1 \\ 1 - 1.5e^{t_f\sqrt{2}} \end{bmatrix}. \tag{57}$$

The exact solution for $t_f = 10{,}000$ is shown in Fig. 4a and b, while Fig. 4c and d shows the state in the initial decay segment $t \in [0, 50]$ and in the terminal growth segment $t \in [9950, 10, 000]$. It is seen that the solution has an initial rapid decay segment followed by a long constant middle segment and a rapid terminal growth segment. Next, Fig. 5a shows the mesh point history along with the regions where $R > \overline{R}$. Two key features of the *hp* method emerge from this example. First, on the early meshes it is found that the *hp* method correctly assesses that the solution is much less smooth ($R > \overline{R}$) near the initial and the terminal time, while the solution is smooth ($R < \overline{R}$) in the long constant middle segment. Next, the mesh interval merging procedure works well in that the *hp* method drastically reduces the size of the mesh in the middle constant segment while retaining the mesh points in the initial and terminal segments. Specifically, it is seen from Fig. 5a that, upon reaching the 5 th mesh refinement (that is, $M = 5$), the collocation points are concentrated in extremely small segments near $t = 0$ and $t = t_f$.

The *hp* mesh refinement method is now compared against various *h* methods and the previously developed *ph* mesh refinement method of Ref. [23]. Fig. 5a–d shows the mesh refinement history for both the *hp* and the $h-3$ method (where the $h-3$ is the best performing of the methods other than the *hp* method on this example). It is seen from Fig. 5c and d that the $h-3$ improves accuracy by increasing significantly the number of collocation points and the number of mesh intervals in the initial decay segment whereas the *hp* method improves accuracy by reducing the mesh size in these segments and concentrating the mesh points only in small segments near $t = 0$ and $t = t_f$. Next, Table 2a–c provides a comparison between the *hp* and *h* methods for increasing values of t_f. First, it is seen that the mesh size using the *hp* method is much smaller than either the *h* or the *ph* methods for all values of t_f. Next, it is seen that the *hp* mesh size remains the same as t_f increases, while the *h* and *ph* mesh sizes grow significantly as t_f increases. In fact, the difference in the computation times for different values of t_f using the *hp* method is due mostly to the increase in the number of mesh refinement iterations (where $M = (5, 6, 7)$ for $t_f = (10{,}000, 100{,}000, 1{,}000{,}000)$, respectively). More importantly, the gap between the *hp* computation time and the (*ph*,*h*) computation time grows significantly as t_f increases. The slow growth in computation time using the *hp* method is also seen in Table 2a–c

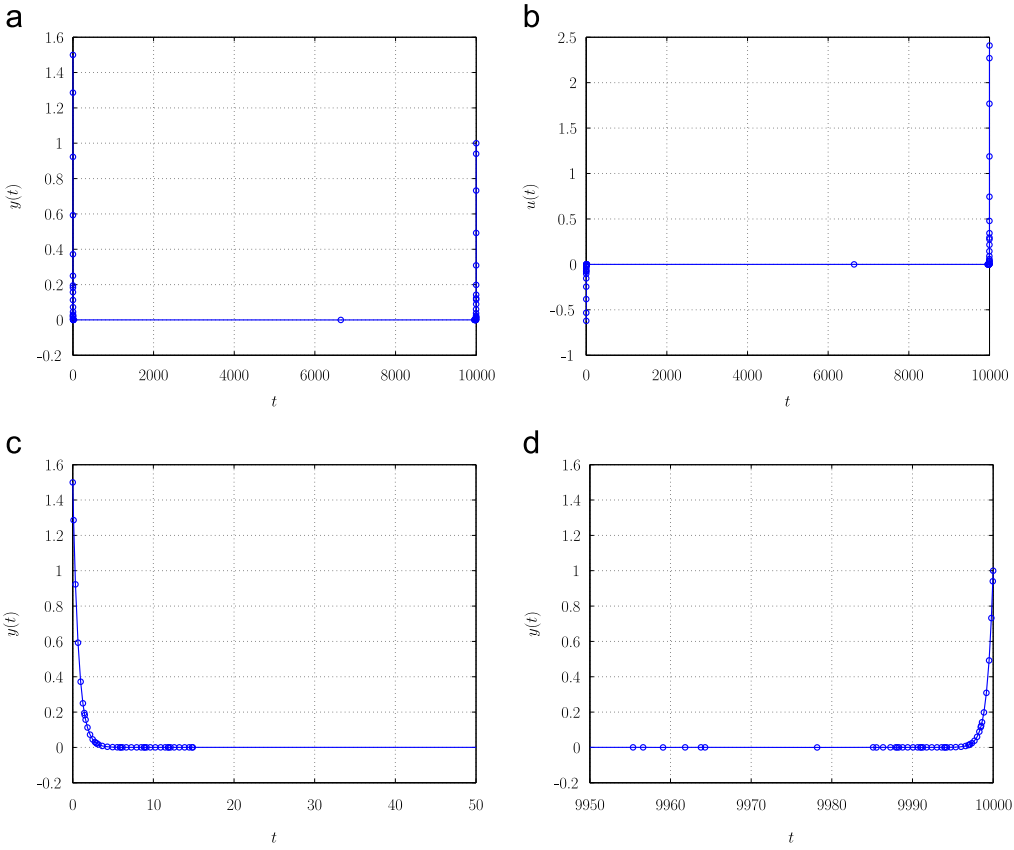

Fig. 4. State solution near $t=0$ and $t=t_f$ for Example 2 with $t_f=10{,}000$ using the *hp* method. (a) $y(t)$ vs. t for $t_f=10{,}000$. (b) $u(t)$ vs. t for $t_f=10{,}000$. (c) $y(t)$ vs. t for $t_f=10{,}000$ near $t=0$. (d) $y(t)$ vs. t for $t_f=10{,}000$ near $t=10{,}000$.

and in Fig. 5e where the *hp* computation times are $(4.92, 12.84, 18.68)$ s for $t_f = (10,000, 100,000, 1,000,000)$, respectively. The computation times using the *h* and *ph* methods, however, change by almost two orders of magnitude as t_f increases from 10,000 to 1,000,000. Finally, Table 2d provides a comparison between the *hp* mesh refinement method of this paper and the mesh refinement used in the optimal control software *Sparse Optimization Suite* (SOS). It is seen for $t_f=10{,}000$ that the final SOS mesh is three times as large as the mesh obtained using the *hp* method. Furthermore, for $t_f=100{,}000$ and $t_f=1{,}000{,}000$ the SOS meshes are approximately 5 and 22 times larger than the *hp* meshes.

6. Discussion

Each of the examples illustrates different features of the *hp* mesh refinement method. The first example demonstrates the ability of the method to accurately determine locations of discontinuities in the solution by predicting correctly the segments where the solution is not smooth. The second example highlights the ability of the *hp* method to significantly reduce the size of the mesh by eliminating collocation points and merging mesh intervals in regions where the solution does not change appreciably and shows that the method concentrates the collocation

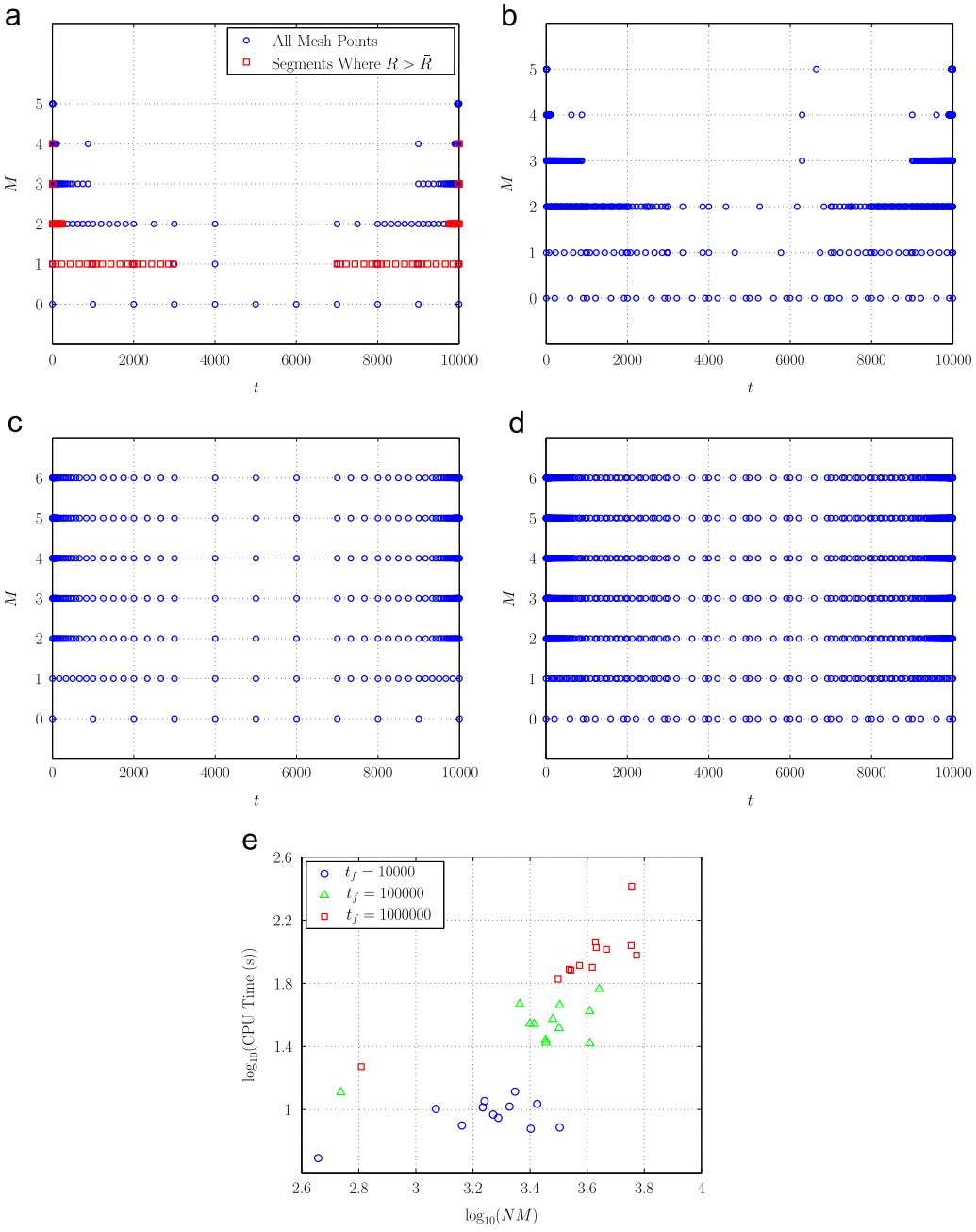

Fig. 5. Mesh refinement history of Example 2 when using the *hp* and $h-3$ methods. (a) *hp* mesh point history. (b) *hp* collocation point history. (c) $h-3$ mesh point history. (d) $h-3$ collocation point history. (e) $\log_{10}$(CPUtime) vs. $\log_{10}(NM)$.

Table 2
Mesh refinement results for Example 2 using *hp* and various ph–$(N_{\min}, N_{\max})$ methods.

(a) $t_f = 10,000$

Method	$N_{\min}$	$N_{\max}$	CPU time (s)	N	K	M
hp	–	14	4.92	91	15	5
h	2	2	7.69	638	315	5
h	3	3	7.55	421	139	6
h	4	4	10.85	380	95	7
ph	3	8	12.98	371	94	6
ph	3	10	11.32	348	77	5
ph	3	12	10.33	343	59	5
ph	3	14	10.09	294	39	4
ph	4	8	8.85	324	66	6
ph	4	10	9.30	311	56	6
ph	4	12	7.92	290	43	5
ph	4	14	10.44	304	39	7

(b) $t_f = 100,000$

Method	$N_{\min}$	$N_{\max}$	CPU time (s)	N	K	M
hp	–	14	12.84	91	17	6
h	2	2	58.10	732	362	6
h	3	3	42.07	508	168	8
h	4	4	26.34	452	113	9
ph	3	8	46.03	456	121	7
ph	3	10	37.49	431	101	7
ph	3	12	34.95	418	80	6
ph	3	14	46.66	385	56	6
ph	4	8	32.77	397	83	8
ph	4	10	34.79	371	69	7
ph	4	12	27.59	356	49	8
ph	4	14	26.53	357	48	8

(c) $t_f = 1,000,000$

Method	$N_{\min}$	$N_{\max}$	CPU time (s)	N	K	M
hp	–	14	18.68	92	17	7
h	2	2	260.31	814	401	7
h	3	3	95.02	593	196	10
h	4	4	109.37	568	142	10
ph	3	8	115.56	532	144	8
ph	3	10	103.54	517	129	9
ph	3	12	77.63	492	106	7
ph	3	14	67.04	450	70	7
ph	4	8	106.30	476	102	9
ph	4	10	79.74	461	90	9
ph	4	12	76.42	436	67	8
ph	4	14	82.06	416	57	9

(d) SOS mesh refinement summary

t_f	$N_{\min}$	$N_{\max}$	CPU time (s)	N	K	M
10,000	–	–	–	313	–	11
100,000	–	–	–	609	–	13
1,000,000	–	–	–	2284	–	15

points in the region where the solution changes rapidly. In contrast, the *ph* method of Ref. [23] does not allow for mesh size reduction and the mesh can only grow to satisfy the accuracy criterion. It is important to note that the performance of the *ph* method depends upon the choice of the parameters $N_{\min}$ and $N_{\max}$ while the performance of the *hp* method depends more strongly upon the choice of $\overline{R}$ and more weakly upon the choice of $N_{\max}$ (because the maximum number of collocation points is attained only in rare cases). Furthermore, the numerical results indicate that, for an appropriate choice of $\overline{R}$, the *hp* method can be more computationally efficient than the *ph* method for almost any choice of $N_{\min}$ and $N_{\max}$. Finally, it is noted that, as with any mesh refinement method, the performance of the *hp* method depends upon the initial mesh.

7. Conclusions

A variable-order adaptive mesh refinement method for solving optimal control problems has been developed. The method has the ability to both increase and decrease the mesh size. The mesh refinement is guided by a previously derived convergence rate. Mesh interval refinement is performed in regions where the solution is nonsmooth, while the polynomial degree is increased in regions where the solution is smooth. Furthermore, the size of the mesh can be decreased either by dropping the negligible terms in a power series representation of the state or by combining mesh intervals that share the same polynomial approximation. The method is described in detail and applied successfully to two examples from the open literature. The results obtained in this research show that the method outperforms fixed-order methods and a previously developed variable-order method.

Acknowledgments

The authors gratefully acknowledge support for this research from the U.S. Office of Naval Research under Grants N00014-11-1-0068 and N00014-15-1-2048, from the U.S. Defense Advanced Research Projects Agency under Contract HR0011-12-C-0011, and from the U.S. National Science Foundation under Grant CBET-1404767.

Disclaimer: The views, opinions, and findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] P.E. Gill, W. Murray, M.A. Saunders, SNOPT: an SQP algorithm for large-scale constrained optimization, SIAM Rev. 47 (January (1)) (2002) 99–131.

[2] L.T. Biegler, V.M. Zavala, Large-scale nonlinear programming using IPOPT: an integrating framework for enterprise-wide optimization, Comput. Chem. Eng. 33 (March (3)) (2008) 575–582.

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

[4] D. Jain, P. Tsiotras, Trajectory optimization using multiresolution techniques, J. Guid. Control Dyn. 31 (September–October (5)) (2008) 1424–1436.

[5] Y. Zhao, P. Tsiotras, Density functions for mesh refinement in numerical optimal control, J. Guid. Control Dyn. 34 (January–February (1)) (2011) 271–277.

[6] G. Elnagar, M. Kazemi, M. Razzaghi, The pseudospectral legendre method for discretizing optimal control problems, IEEE Trans. Autom. Control 40 (10) (1995) 1793–1796.

[7] G. Elnagar, M. Razzaghi, A collocation-type method for linear quadratic optimal control problems, Optim. Control Appl. Methods 18 (3) (1998) 227–235.
[8] F. Fahroo, I.M. Ross, Costate estimation by a legendre pseudospectral method, J. Guid. Control Dyn. 24 (2) (2001) 270–277.
[9] F. Fahroo, I.M. Ross, Direct trajectory optimization by a Chebyshev pseudospectral method, J. Guid. Control Dyn. 25 (1) (2002) 160–166.
[10] D.A. Benson, G.T. Huntington, T.P. Thorvaldsen, A.V. Rao, Direct trajectory optimization and costate estimation via an orthogonal collocation method, J. Guid. Control Dyn. 29 (November–December (6)) (2006) 1435–1440.
[11] G.T. Huntington, D.A. Benson, A.V. Rao, Optimal configuration of tetrahedral spacecraft formations, J. Astronaut. Sci. 55 (April–June (2)) (2007) 141–169.
[12] G.T. Huntington, A.V. Rao, Optimal reconfiguration of spacecraft formations using the gauss pseudospectral method, J. Guid. Control Dyn. 31 (May–June (3)) (2008) 689–698.
[13] Q. Gong, F. Fahroo, I.M. Ross, Spectral algorithm for pseudospectral methods in optimal control, J. Guid. Control Dyn. 31 (May–June (3)) (2008).
[14] Q. Gong, I.M. Ross, W. Kang, F. Fahroo, Connections between the covector mapping theorem and convergence of pseudospectral methods, Comput. Optim. Appl. 41 (December (3)) (2008) 307–335.
[15] A.V. Rao, D.A. Benson, C.L. Darby, C. Francolin, M.A. Patterson, I. Sanders, G.T. Huntington, Algorithm 902: GPOPS, a Matlab software for solving multiple-phase optimal control problems using the Gauss pseudospectral method, ACM Trans. Math. Softw. 37(April–June (2)) (2010), Article 22, 39 pp.
[16] D. Garg, M.A. Patterson, C.L. Darby, C. Francolin, G.T. Huntington, W.W. Hager, A.V. Rao, Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems via a Radau pseudospectral method, Comput. Optim. Appl. 49 (June (2)) (2011) 335–358, http://dx.doi.org/10.1007/s10589-00-09291-0.
[17] D. Garg, M.A. Patterson, W.W. Hager, A.V. Rao, D.A. Benson, G.T. Huntington, A unified framework for the numerical solution of optimal control problems using pseudospectral methods, Automatica 46 (November (11)) (2010) 1843–1851, http://dx.doi.org/10.1016/j.automatica.2010.06.048.
[18] D. Garg, W.W. Hager, A.V. Rao, Pseudospectral methods for solving infinite-horizon optimal control problems, Automatica 47 (April (4)) (2011) 829–837, http://dx.doi.org/10.1016/j.automatica.2011.01.085.
[19] S. Kameswaran, L.T. Biegler, Convergence rates for direct transcription of optimal control problems using collocation at Radau points, Comput. Optim. Appl. 41 (1) (2008) 81–126.
[20] C.L. Darby, W.W. Hager, A.V. Rao, An *hp*-adaptive pseudospectral method for solving optimal control problems, Optim. Control Appl. Methods 32 (July–August (4)) (2011) 476–502.
[21] C.L. Darby, W.W. Hager, A.V. Rao, Direct trajectory optimization using a variable low-order adaptive pseudospectral method, J. Spacecr. Rockets 48 (May-June (3)) (2011) 433–445.
[22] C.C. Francolin, W.W. Hager, A.V. Rao, Costate approximation in optimal control using integral gaussian quadrature collocation methods, Optim. Control Appl. Methods 36 (July–August (4)) (2015) 381–397.
[23] M.A. Patterson, W.W. Hager, A.V. Rao, A *ph* mesh refinement method for optimal control, Optim. Control Appl. Methods 36 (July–August (4)) (2015) 398–421.
[24] C. Canuto, M.Y. Hussaini, A. Quarteroni, T.A. Zang, Spectral Methods in Fluid Dynamics, Springer-Verlag, Heidelberg, Germany, 1988.
[25] B. Fornberg, A Practical Guide to Pseudospectral Methods, Cambridge University Press, New York, 1998.
[26] L.N. Trefethen, Spectral Methods Using MATLAB, SIAM Press, Philadelphia, 2000.
[27] Q. Gong, F. Fahroo, I.M. Ross, Spectral algorithm for pseudospectral methods in optimal control, J. Guid. Control Dyn. 31 (May–June (3)) (2008) 460–471.
[28] R. Munos, A. Moore, Variable resolution discretization in optimal control, Mach. Learn. 49 (November–December (2–3)) (2002) 291–323.
[29] L. Grune, An adaptive grid scheme for the discrete Hamilton–Jacobi–Bellman equation, Numer. Math. 75 (January (3)) (1997) 319–337.
[30] R. Rannacher, Adaptive finite element discretization of flow problems for goal-oriented model reduction, in: H. Choi, H. Choi, J. Yoo (Eds.), Computational Fluid Dynamics 2008, Springer, Berlin, Heidelberg, 2009, pp. 31–45.
[31] M. Besier, R. Rannacher, Goal-oriented space–time adaptivity in the finite element Galerkin method for the computation of nonstationary incompressible flow, Int. J. Numer. Methods Fluids 70 (November (9)) (2012) 1139–1166.
[32] I. Babuska, M. Suri, The p and hp version of the finite element method, an overview, Comput. Methods Appl. Mech. Eng. 80 (1990) 5–26.

[33] I. Babuska, M. Suri, The *p* and *hp* version of the finite element method, basic principles and properties, SIAM Rev. 36 (1994) 578–632.
[34] W. Gui, I. Babuska, The *h*, *p*, and *hp* versions of the finite element method in 1 dimension. Part I. The error analysis of the p version, Numer. Math. 49 (1986) 577–612.
[35] W. Gui, I. Babuska, The *h*, *p*, and *hp* versions of the finite element method in 1 dimension. Part II. The error analysis of the *h* and *h–p* versions, Numer. Math. 49 (1986) 613–657.
[36] W. Gui, I. Babuska, The *h*, *p*, and *hp* versions of the finite element method in 1 dimension. Part III. The adaptive *h–p* version, Numer. Math. 49 (1986) 659–683.
[37] R. Pachon, R. Platte, L.N. Trefethen, Piecewise smooth chebfuns, IMA J. Numer. Anal. 30 (2010) 898–916.
[38] H. Hou, Convergence analysis of orthogonal collocation methods for unconstrained optimal control (Ph.D. thesis), University of Florida, August 2013.
[39] F. Liu, W.W. Hager, A.V. Rao, An *hp* mesh refinement method for optimal control using discontinuity detection and mesh size reduction, in: 2014 IEEE Conference on Decision and Control, Los Angeles, CA, 15–17 December 2014, pp. 5868–5873.
[40] M. Abramowitz, I. Stegun, Handbook of Mathematical Functions with Formulas Graphs, and Mathematical Tables, Dover Publications, New York, 1965.
[41] Z. Battles, L.N. Trefethen, An extension of Matlab to continuous functions and operators, SIAM J. Sci. Comput. 25 (2004) 1743–1770.
[42] E. Dolan, J.J. More, T.S. Munson, Benchmarking Optimization Software with CPOPS 3.0, Technical Report, ANL/MCS-273, Argonne National Laboratory, Argonne, IL, 2004.
[43] A.V. Rao, K.D. Mease, Eigenvector approximate dichotomic basis method for solving hyper-sensitive optimal control problems, Optim. Control Appl. Methods 21 (January–February (1)) (2000) 1–19.
[44] M.A. Patterson, 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 Trans. Math. Softw. 41 (October (1)) (2015) 1:1–1:37.
[45] M.J. Weinstein, A.V. Rao, A source transformation via operator overloading method for the automatic differentiation of mathematical functions in MATLAB, ACM Trans. Math. Softw. 42 (1) (2015).
[46] R. Pachón, R. Platte, L.N. Trefethen, Piecewise-smooth chebfuns, IMA J. Numer. Anal. 30 (2010) 898–916.
[47] J.T. Betts, W.P. Huffman, Sparse Optimal Control Software-SOCS, Technical Report, MEA-LR-085, Boeing Information and Support Services, Seattle, Washington, July 1997.